A Model for Mapping Graduates' Skills to Industry Roles Using Machine Learning Techniques: A Case of Software Engineering

By

Fullgence Mwachoo Mwakondo

P80/98728/2015

Supervisors:

1. **Dr. Lawrence Muchemi**
2. **Prof. Elijah Omwenga**

Thesis Presented for the Award of Degree of Doctor of Philosophy in Computer Science

School of Computing and Informatics

University of Nairobi, Kenya

DECLARATION

I hereby declare that this thesis is my own work and has, to the best of my knowledge, not been submitted to any other institution of higher learning.

Signature: .... **Date:** 12/10/2017 ....

Name: Fullgence Mwachoo Mwakondo **Registration Number:** P80/98728/2015

This thesis has been submitted for examination towards fulfillment for the award of degree of Doctor of Philosophy in Computer Science with my approval as the supervisor.

Signature: .... **Date:** ....

Name: Dr. Lawrence Muchemi

School of Computing and Informatics,

University of Nairobi, Kenya

Signature: .... **Date:** ....

Name: Prof. Elijah Isanda Omwenga

School of Computing and Informatics,

University of Nairobi, Kenya

Abstract

Despite rapid development in information technologies, a practical way of mapping graduates' skills to industry roles is a challenge. Attempts have been made by posing this as a multi-classification problem and solving using machine learning techniques. However, existing approaches seem not to embrace attributes and machine learning structures relevant to the problem, and hence, their results may not be reliable. For example, although occupational industry roles in the organizations are structured hierarchically, many studies have approached this problem using flat instead of hierarchical methods. Either relevant attributes or hierarchical structure that correctly reflects hierarchy of industry roles, or both, are unknown for an effective model for mapping graduates' skills to industry roles.

Currently, hierarchical method has not been applied in skills mapping to industry roles despite its many benefits vis-à-vis flat method. However, in other areas where it has been used, classification approach contradicts underlying structure of the problem thus resulting in multiple label prediction problems. As a result, this study presents an investigation that posed skills mapping to industry roles as a hierarchically structured multiclass problem where a machine learning structure that correctly reflects the hierarchy of industry roles was applied. The aim being to demonstrate using a case how to build a machine learning model for mapping graduates' skills to hierarchically structured industry roles. This was achieved by establishing both underlying structural characteristic of industry roles, as concepts required for target classes, that correctly reflects the hierarchy of industry roles and concepts appropriate as attributes for hierarchical machine learning purpose, before building and evaluating the mapping model. The model is based on the underlying taxonomic structure whose basic approach is to correctly reflect the hierarchical structure of industry roles. Literature analysis of three theoretical frameworks provided a basis for establishing appropriate attributes for machine learning investigation after which hierarchical classification strategy was designed to generate the model before its prototype was constructed. Experimental design was adopted using four machine learning techniques (Logistic Regression, K-Nearest Neighbor, SVM, and Naïve Bayes). A benchmark dataset and 113 Software Engineering employees' skills profile data collected using stratified random sampling from various software development firms in Nairobi were involved in the investigation. Experiments to evaluate performance and validity of the model were designed using repeated 5-fold cross validation procedure. Performance reported on carefully selected benchmarks on multi-classification method was adopted for validation of results.

Findings revealed five appropriate attributes for building a model for mapping skills to industry roles and the best model was SVM induced with an average generalization performance accuracy of 67% across three datasets. On benchmark dataset, our model registered performance accuracy of 85% better than 82% reported by a selected benchmark on similar dataset. These results seem to be fairly consistent with results achieved by similar hierarchical models as reported in other problem domains such as proteins (53.3%) and music (61%).

In conclusion, the research objective was fulfilled with the following contributions, namely conceptual model, ML architecture for the model, software prototype, hierarchical mapping framework, research findings, datasets and literature survey which will benefit researchers in general (students, universities and industry) and specially the government in developing an effective policy for training evaluation that ensures graduates are relevant to the industry.

Keywords: Hierarchical Classification, Industry-Academia Gap, Problem-solving, Skills Mapping

Acknowledgement

As I approach towards the end of this journey, I would like to take this great moment to remember all those who have contributed directly and indirectly towards this end. Indeed, a PhD journey is one that is full of ups and downs, mountains and valleys, rivers and rocks, oceans and forests and every kind of person you meet around is as useful as a friend, and a friend indeed. It is upon this I would like to remember you all whom I gripped your hands and lifted me up again when I almost fell into despair. Your encouragement, support, patience, tolerance, understanding, consolation, consultancy, advice, opinion, suggestion, counseling, guidance, is what I greatly needed during this moment for a little forward movement towards the end.

First, I would like to express my thanks to my supervisors, Dr Lawrence Muchemi and Prof Elijah I. Omwenga for their endless support and guidance throughout this study. I am greatly indebted to Prof Omwenga without whom this idea would not have grown to maturity and with whose guidance shaped the whole idea into a research proposal worth a PhD research. Prof you have done a great job worth to mention. To Dr Muchemi, you have been my great inspiration, ensuring my determination and strength of mind is always high, to carry on with the journey. I feel greatly owed to you, especially when I remember great meetings with you that raised my spirits high. Daktari, I am grateful for the job well done.

Secondly, I greatly express my thanks to University of Nairobi for granting me this great opportunity to study in this great institution, especially School of Computing and Informatics for their great resources, seminars and supervisions. I would like, also, to express my gratitude to all my classmates for their valuable advice and support including Hadullo and especially Omondi for data collection support; all the lecturers in the school for all their endless support during my study, including Prof Waiganjo, Dr Oboko, Dr Opiyo, Prof Okello, Dr Mbura, to mention just a few; and a special friend indeed Dr Collins for your great support during data collection.

Thirdly, my great appreciations go to my colleagues in my home department at Technical University of Mombasa (TUM) including Dr Mvurya Mgala for proof reading, and especially TUM as an institution for financially supporting this study through staff development scholarship award towards tuition fees. I would like to say without this scholarship my studies would have been very difficult. Others worth to remember and appreciate for their dramatic help include National Council of Science, Technology and Innovation (NACOSTI). NACOSTI have been very instrumental in supporting the research funding part of this study without which would have been difficult to collect and analyze data and present results. Thank you all.

Finally, to my own family especially my wife, Selestinah, my children, my dad and mum, my sisters, family friends, I wish to thank them all for their endless support. I wish to say the list is too long to fit into this space. However, the whole list is in my heart and I wish you all “many thanks” for this great work. God bless you all.

Dedication

I would like to dedicate this thesis to my family, dad, and mum for their patience and great support during my study.

Table of Contents Page

List of Tables Page

List of Figures Page

List of Abbreviations and Acronyms

AI	Artificial Intelligence
AL	Academic Librarians
ANN	Artificial Neural Network
DBMS	Database Management Systems
e-CF	European e-Competence Framework
CRESST	Center for Research on Evaluation, Standards, and Student Testing
GPA	Grade Point Average
HKCS	Hong Kong Computer Society
ICT	Information and Communication Technology
IDC	International Data Corporation
IS	Information Systems
IST	Innovation Science & Technology
IT	Information Technology
ITCA	Industry Training Advisory Committee
LTU	Long Term Unemployment
LR	Logistic Regression
NAS	National Academy of Sciences
OECD	Organization for Economic Co-operation and Development
SE	Software Engineering
SWEBOK	Software Engineering Body of Knowledge

Definitions of Terms

Competence

This refers to a proven ability to use or apply knowledge, skills and attitudes for achieving observable results in a work or study situations.

Knowledge

This refers to a body of facts, principles, theories and practices that is related to a field of work or study which is assimilated through learning or training.

Learning outcomes

These are statements of what a learner knows, understands and is able to do on completion of a learning process, which are defined in terms of knowledge, skills and competence.

Qualification

It is a formal outcome of an assessment and validation process which is obtained when a competent body determines that an individual has achieved learning outcomes to given standards. It is a standard declaring the amount of learning outcome achieved by a learner.

Industry Role

It is a job title in an industry occupation.

Skills

This is the ability to apply knowledge and use know-how to complete tasks and solve problems. Skills are described as cognitive (involving the use of logical, intuitive and creative thinking) or practical (involving manual dexterity and the use of methods, materials, tools and instruments)

Skills Mapping

This is a mechanism for matching a set of related skills with known industry roles for the purpose of prediction. This process links industry jobs with highly skilled workforce and involves use of analytical methods, such as machine learning, to determine graduate's right match of knowledge, skills and their levels for performing jobs efficiently.

CHAPTER 1: INTRODUCTION

1.1 Background to the Study

International Labor Organization Global Employment Trends (2015) indicate rapid growth of Long Term Unemployment (LTU) which is as a result of increased unemployment rate currently standing at 13 per cent, originally at 5.6 and 6.2 per cent in 2007 and 2010 respectively (Jantawan & Tsai, 2013). In Europe, number of unemployed persons went up from 30.6 million in 2007 to 47 million in 2010, while LTU went up from 8.5 million to 14.9 million in the same period (Junankar, 2011). These correspond to an increment ratio of 1.5359 and 1.7529 respectively.

Empirical studies indicate that unemployment problem relates to either workers unable to match their skills to requirements of advertised jobs (Kaminchia, 2014), or employers unable to find workers with important skills, especially both before and after economic recession of 2008 to 2010. Large companies have the highest trouble (30% before and 25% after recession), than smaller companies (19% before and 17% before recession) (Perron, 2011).

In Kenya, the number of unemployed persons increased from 1.8 million in 1998/99 to 1.9 million in 2005/2009 (Kaminchia, 2014). Empirical studies indicate that unemployment problem relates to workers unable to match their skills to the requirements of advertised jobs (Kaminchia, 2014). This situation has posed serious psychological and socio-economic challenges to the unemployed persons including loss of skills through human capital depreciation, loss of motivation, self-respect and dignity, and finally leading to poverty, terrorism, riots, divorce, illness and death (Kaminchia, 2014). According to McCowan *et al.* (2016), the economic survey of 2014 in the Republic of Kenya indicates the youth (15-35 years) who form 35% of Kenyan population have the highest unemployment rate of 67%.

However, LTU wouldn't be a trouble if characteristics of each kind of job, level of education and skills, and experience were precisely known by the new graduates; if search strategies followed by graduates improved search intensity and efficiency; if matching the characteristics employers sought against characteristics of applicants was made possible to predict suitability for employment much earlier before the applicants faced the employer and before duration of unemployment was used as a signal of quality of work productivity. Suitability for employment of skilled graduates in the industry is a challenge not only because of the effect of LTU, but due to increased skills variation among both graduates and industry roles, emanating from the industry academia gap (Quintin, 2011).

For instance, employers often describe their staffing requirements in terms of job profiles and/or competences while academia expresses the characteristics of their graduates through certifications and qualifications. Although creation of job profiles and the concept of competence are ways of communicating the knowledge and skills characteristics required by industry (CWA16458, 2012) to stakeholders and specifically academia, Aggarwal *et al*. (2015) indicates that mapping graduates' skills to job profiles is not easy

In the academia, education and training are key activities that ensure supply of qualified practitioners in the industry (Show, 2000; Shkoukani, 2013a). However, many education and training providers in the academia have certifications that lack transparency in content (Korte *et al.*, 2013) and have resulted not only to increased qualifications mismatch but also skill variations between individuals with same qualifications (Quintin, 2011). This has been evidenced by revelation of recent studies (Cihan & Kalipsiz, 2014; Shkoukani, 2013b; Cope *et al.*, 2000) that employers are not satisfied with knowledge and skills of new graduates. In fact, there is an obvious difference between the industry needs and the actual supply from the academia hence causing a mismatch gap between academia and industry (Tamayko, 1998, Shkoukani, 2013a).

1.1.1. Causes of the Gap between Academia and Industry

The issues causing industry-academia gap have been studied widely with an obvious aim of sending a strong signal of warning to academia and these issues have ranged from curriculum to assessment.

1) Curriculum Issues

There are three types of curriculum: planned, delivered, and experienced curricula (Kenny & Desmarais, 2010). Planned curriculum refers to what is intended or planned for the learner while delivered curriculum refers to what is taught by the teacher to the learner and experienced curriculum consists of what is learned or experienced by the learner during or after learning. According to Kenny & Desmarais (2010), the three types of curricula are layered. Planned curriculum, which is at the lower level, affects the delivered curriculum, while delivered curriculum, which is in the second level, affects the experienced curriculum. Since planned curriculum is the foundation for the other two and experienced curriculum is the product, then the gaps in planned curriculum affects the experienced curriculum causing the industry to raise alarm.

Recent studies (Moreno *et al.*, 2012) suggest there are mismatch gaps between *defacto* curriculum and the knowledge expected by the industry. McCowan *et al.* (2016) have associated all this with decline of funding in public universities by the government hence forcing universities to cut cost by focusing on less expense aspects of curricula. As a result, academic curricula are mostly theory based, heavily governed by knowledge components and rarely include problem solving skills, best practices, interpersonal skills, and leadership skills (Lee & Han, 2008; Kichenham *et al.*, 2005). According to McCowan *et al.* (2016), universities are forced to focus less expense areas such as theoretical aspects, knowledge aspects (factual, conceptual and procedural) and very little on expensive aspects such as practical skills.

Besides, Moreno *et al.* (2012) revealed that some topics in the domain body of knowledge are totally ignored. This is in agreement with previous studies (Lethbridge *et al*, 2007; Gargi & Varma, 2008) that cited the same views. Higher order cognitive skills such as application, analysis and evaluation which are important for problem solving are rarely part of the curriculum.

Many similar undergraduate degree programs curricula in different universities have different emphasis on domain content knowledge and skills. For example, a survey conducted in 1998 shows that there are over 77 graduate software engineering programs all over the world each with different career and content emphasis for SE skills (Shaw, 2000). Or, even some undergraduate programs contain more than one domain skills in one curriculum, such as in computer science where most undergraduate SE education is enshrined within computer science degree programs as SE course and related SE courses (Cihan & Kalipsiz, 2014).

2) Pedagogy issues

The traditional lecture-based teaching method and large classroom enrollments are not effective for teaching. Lecture-based teaching method is only suitable for imparting theoretical knowledge hence denying learner's application of knowledge and skills through practical training (Shaw, 2000). The lecture-based model has been shown by Jackson and Posser (1989) as cited by Cope *et al* (2000) to be effective in transferring knowledge from lecturer to students but ineffective in promoting conceptual understanding. According to Gargi & Varma (2008), large number of students enrolled in each class is too high for effective classroom teaching. McCowan *et al.* (2016) observes low quality of training as a result of this.

While some topics are prescribed very little time, others are taught in more depth than required in the industrial practice (Lethbridge *et al.*, 2000; Kichenham *et al* 2005; Surakka, 2007). Further, there is a complain of inadequate time to cover the curriculum as provided by the domain's body of knowledge. For example, according to Gargi & Varma (2008), SE course is often taught as a one semester course in most computer science programs of which it means 2-3 hours of teaching per week for about 14-16 weeks.

3) Resource related issues

The resources available in many institutions are not sufficient to model quality professionals as per the industry requirements. Poor educational infrastructure such as under-equipped computer labs denies students practical exposure. Findings of a study carried out by Shkoukani (2013b) in Indian Universities reveals that there are no well equipped laboratories, adequate tools and software development experienced teachers towards producing well qualified SE graduate (Shkoukani, 2013b). Bondesson (2004) observes lack of qualified teachers resulting to professional experience limiting learners to theoretical aspects only (Bondesson, 2004).

4) Assessment issues

Assessments, especially in projects, are not done effectively to provide sufficient evaluation of the learners' skills capacity or learning outcomes or to check if students used practices, tools or techniques appropriately. Sometimes, projects assigned to students are not assessed throughout each step but at the end during presentation hence giving a grade that merely reflects presentation alone (Shkoukani, 2013a, 2013b). Besides, since there is a one or two semester gap between attending the training of the course and applying the training skills in the project, the learners are likely to forget the knowledge.

Also, most projects are academic in nature and do not represent the issues of scale and complexity of real world and are very poor in soft skills. For instance, findings in a study by Cihan & Kalipsiz (2014) reveals that soft skills are more important than hard skills for the success of projects, and therefore, there is close relationship between success of projects and soft skills.

5) Industry issues

There are rapid changes in the industry resulting in a growing demand for both professionals and products especially in the ICT sector. However, Ellis *et al.* (2002) note that the number of professionals is not growing at the rate equal to the growth rate of industry demand. For instance, in the ICT sector, the few software engineers available do not meet the SE industry needs (Kolding &

Ahorlon, 2009); while Moreno *et al.*, (2012) observe that the newly graduated software engineers have a problem of matching the industry skill profiles.

6) New observations

In the modern age, matching of skills to industry roles could be achieved using information technologies. However, the current study has observed little efforts towards use of appropriate methods and as a result causing the industry academia gap.

1.1.2. Effects of the Industry-Academia gap

1) Effect of curriculum issues on graduates

The curriculum issues described above have resulted into a pool of graduates with diverse domain skills. Their diversity is around a number of attributes with different levels that determine their skills (Norwood & Briggeman, 2010) such as depth of understanding, level of skill competence or problem solving skills, general capabilities of the student, etc. (Shaw, 2000; Shkoukani, 2013a). These variations have rendered graduates a challenge in matching their domain skills with the existing industry needs (Shkoukani, 2013a). Determining which roles they are likely to fit in the industry based on their skills is not easy. There is significant amount of diversity among graduates and among industry jobs (Norwood & Briggeman, 2010). Although Show (2000) recommends that education and training should prepare student differently for different industry roles, it is expensive.

2) Effect of industry issues on industry practitioners

In order to cope with challenges of evolving industry sector, many companies have structured their needs into a number of professional roles. The job descriptions of these roles capture the requirements relevant to their industry needs (HKCS, 2011). For example, evolving SE industry produces new applications that must have new SE requirements of being autonomous, extensible, flexible, robust, reliable and capable of being remotely monitored and controlled. According to Shkoukani & Lail (2012), this demands new SE approaches whose nature is different from that of classical approaches. This leads to new SE roles with new competences which are significantly different from those of classical SE.

These variations of industry needs into diverse professional roles have rendered industry a challenge in matching their requirements with the available graduate skills (Shkoukani, 2013a). Determining whether a graduate has the skills level relevantly needed by a given company is not easy due to the diverse skills of these graduates. Furthermore, graduates possessions of these skills are not directly observable (Norwood & Briggeman, 2010).

1.1.3. Towards bridging the gap

There is need to bridge the gap between industry and academia. Thompson *et al.* (2007) observe that academia Industry interaction is vital to bridging the gap through partnership in research projects and curriculum development and review. This can lead to production of skillful graduates compatible with industry requirements. However, employers in the industry describe and communicate their staffing requirements in terms of job profiles and/or competences, while academia communicates the skills and knowledge characteristics of their graduates in terms of certifications and qualifications (CWA16458, 2012). This communication breakdown has possibly led to a mismatch between skills possessed by graduates and skills required by the industry (Quintin, 2011).

Besides, there are many institutions providing undergraduate degree programs with similar names leading to certifications that are different or similar, but producing graduates with different qualifications or competences. According to Korte *et al.*, (2013), this is as a result of either or both lack of transparency in the content of different courses or different entry points for new students in different training institutions. Ideally, individuals with the same certification and qualifications should portray same level of competence. However, this cannot be guaranteed because individuals differ in the ability to acquire knowledge and skills (Quintin, 2011; Plant & Hammond, 2004; Kraiger *et al.*, 1993) leading to differences between individuals in skill levels and types they possess (Handel, 2012).

To acquire knowledge and skills, intellectual abilities are essential prerequisites that are needed (Winterton *et al.*, 2005). While academia provides this knowledge and skills through training, there is no direct control on the amounts the learner finally acquires or transfers apart from the learner's abilities (Handel, 2012). Though studies have also shown there are other factors that influence the acquisition and transfer of knowledge and skills including academic staff capability, infrastructure, domain course content, specific requirements, etc (Shkoukani, 2013a), which is clearly evidenced in individuals with same qualifications but have varied competences (Quintin, 2011). Consequently, there is a challenge to employers in screening through the qualification mix of many individuals with similar qualifications (Quintin, 2011; Korte *et al.*, 2013) for the required skills needed for the jobs during recruitment.

According to Thompson *et al.* (2007), the industry has a picture of the knowledge, skills and abilities that a new graduate should possess for each role, and these are the skills that employers seek from graduates, like problem solving skills, communication skills, leadership skills, ability to work well with others etc (Griffin, 2008; Sutherland *et al.*, 2009; Norwood & Briggeman, 2010). Although many studies have singled out problem solving as one of the key skills that employers seek (NACE, 2006; Hansen & Hansen, 2007, Texas A & M, 2007), there is little research about how this skill is assessed (Norwood & Briggeman, 2010). The signals employers use to measure graduates for problem solving skills like performance in interviews, previous leadership positions and internship, are not ideal for measuring problem solving skills (Norwood & Briggeman, 2010).

Problem solving is a cognitive process that includes goal-oriented thinking and involves the use of previously acquired knowledge, skills and understanding to meet the demands of an unfamiliar situation (Krulik & Rudnik, 1996; Baker & Mayer, 1999; Orhun, 2003; Wirth & Klieme, 2011). Research findings indicate that knowledge and skills acquired during class lectures are the most important variables that increase performance in problem solving skills (Robertson, 1990; Orhun, 2003). Further, the thresholds and certification levels for these skills vary differently for different domain roles in the industry (Shkoukani & Lail, 2012; Korte *et al*, 2013) but the precise levels and kind of skills demanded by each role are poorly understood (Handel, 2012).

There is a challenge in assessing problem solving competence in the traditional classroom education and training where evaluation is limited only to the learning objectives. More often, classroom grades are used to indicate knowledge and skills acquired in class lectures hence signal problem solving skills. However, grades alone are not sufficient to indicate problem solving skills due to issues in section 1.1.1 of this chapter. Furthermore, there is a significant variation in grading from grader to grader (Srikant & Aggarwal, 2014).

Although, apart from classroom, there are other forms of education and training such as online, self study, and on job training (CWA16458, 2012), still the challenge remains, and most often there are three issues in problem solving competence assessment which Baker & Mayer (1999) characterize as follows: what to test (product or process?), how to test (routine or non-routine problems?), where to test (separate skills in isolated situation or integrated skills in authentic context situation?). For example, education for software engineers is confounded with education for other non-software engineers (Show, 2000).

Problem solving competence is multidimensional (Wirth & Klieme, 2011), and consists of at least two aspects: analytic and dynamic. Analytic aspect of problem solving competence is strongly related to intelligence while dynamic aspect is neither related to intelligence nor school-related literacy. Moreover, problem solving competence is more of knowledge transfer than retention, more of meaningful learning than rote learning, and more of qualitative learning than quantitative learning (Baker & Mayer, 1999; Wirth & Klieme, 2011). Therefore, evaluation of problem solving competence requires assessment methods that are not only valid and efficient (Mayer, 2002; Kraiger *et al.*, 1993) but also cognitive, skill-based and affective.

Evaluation of problem solving competence should not be done quantitatively in the traditional classroom way because of its multidimensionality nature, but instead qualitatively relative to industry roles' competences. Dimensions for problem solving competence should be used as signals for problem solving skills that employers should use for different industry roles and they should be founded on strong cognitive abilities that enable them to adapt in case of unexpected changes or problems (Plant & Hammond, 2004). The scales for these dimensions should be derived from the respective industry roles requirements.

While employers seek insight on current and future personnel needs, job seekers, parents, and students seek to not only know which job prospects look favorable but also understand the requirements in terms of education, training and other characteristics (Handel, 2012). Besides, with ever increasing unemployment trends in the world and decreasing capacity of most economies to create employment opportunities, employability of young and productive graduates from universities is at threat of LTU if something is not done to reverse the trend.

Likewise, with ever increasing pool of qualification mix of new graduates from universities each year, employers are at risk of not only taking longer to search the pool but also selecting graduates whose skills do not match their needs. Conventionally, Bharthvajan (2013) has observed that employers select employees with the right match to efficiently perform jobs based on qualifications before they interview them. However, the relationship of this technique to selection of employees with adequate performance is not even 10% correct (Bharthvajan, 2013).

As an alternative, many employers have converted to skills mapping. Skills mapping is a mechanism that links highly skilled graduates with industry jobs. This involves use of analytical methods to determine graduate's right match of knowledge, skills and their levels for performing jobs efficiently. Analytical methods in skills mapping are vital in ensuring high performance of highly skilled workforce from academia in the industry jobs. Computationally, skills mapping problem could be viewed as a pattern recognition problem where evaluation of such problems using technology is the essence of Artificial Intelligence (AI).

In AI, such problems are tackled using two broad approaches, either searching techniques or modeling techniques. Searching techniques involve applying a process with search conditions to look for the solution of the problem through a set of possibilities, where solution is a path from current state to goal state. Modeling techniques involve creating a general model to represent the natural phenomena then using either knowledge based systems or data driven methods such as machine learning (ML) techniques to estimate or learn unknown parameters of the model. Due to wide availability of data globally, data driven methods, such as ML techniques, are gaining traction.

ML is one of the major branches of Artificial Intelligence (AI) that is concerned with designing programs (ML algorithms) that attempt to make computers behave intelligently by being able to sense, remember, learn, and recognize patterns (Leeuwen, 2004). Currently, major areas of ML research include speech recognition, computer vision, bio-surveillance, robotic control, and data mining. The first three are concerned with pattern recognition, while the last two relate to adapting based on self-collected data and knowledge discovery respectively. The current study relates to the area of pattern recognition but in the focused area of skills mapping to industry roles.

Pattern recognition, according to Basu *et al.* (2010), is the study of how machines can observe the environment, learn to distinguish patterns of interest in their background, and make reasonable decisions about the categories of patterns. The pattern recognition problem is posed as a classification task where the classes are either predefined or are learned based on similarities of patterns. To solve such kind of problems a suitable classification method and algorithm to learn the classifier are needed. This study relates to pattern recognition where classes are predefined as industry roles. As a result, skills mapping problem can be viewed computationally as a pattern recognition problem where a feature space of diverse skills graduates requires a classifier to map to a set of possible classes of industry roles.

Recently, research on skills mapping using ML techniques has been active as observed in the works of Chien & Chen (2008) in mapping demographic profile of employees to retention and performance in the job; Jantawan & Tsai (2013) in mapping demographic profile of employees to employment status; Korte *et al.* (2013) in mapping certification knowledge and skills content to industry roles; Srikart & Aggarwal (2014) in mapping programming skills of an employee to software developer's ability to solve problems; Shashidhar *et al.* (2015) in mapping skills to SE industry roles. This is as a result of wide availability of data globally where data driven methods are gaining traction. Figure 1.1 illustrates skills mapping using classifiers.

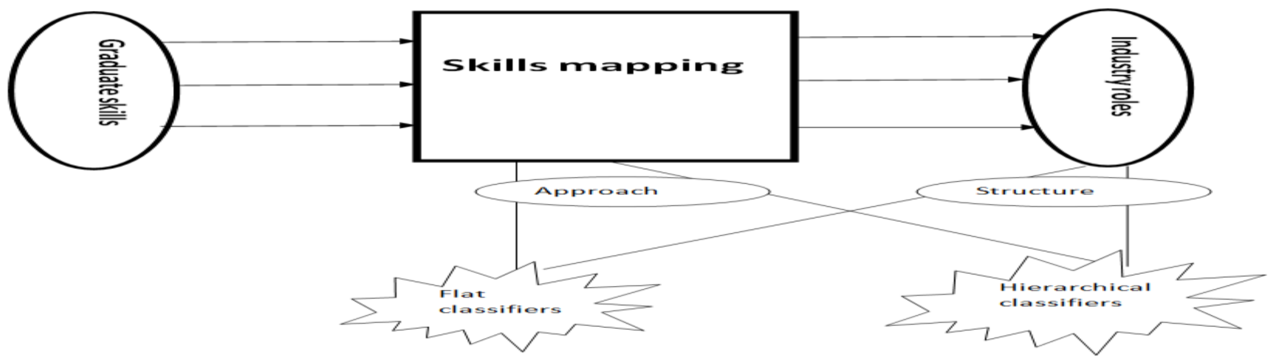


Figure 1.1: Skills mapping using flat classifiers and hierarchical classifiers (adapted from Chien & Chen, 2008; KIM, 2009)

However, there seems to be a broad way of establishing ML attributes where some are not relevant either to industry roles performance or across occupational industry domains. Besides, there seems to be two lines of thought for skills mapping (Chien & Chen, 2008; Jantawan & Tsai, 2013; Korte *et al.*, 2013; Shashidhar *et al.*, 2015), classification or regression. There is need to make clear which one is relevant. Classification is where job performance skills are classified into various known finite range of industry roles' classes before skills of graduate are matched with these classes. This process results to matching graduate's skills to only known and finite industry roles.

Regression is where skills thresholds for various industry roles are predefined on a continuous scale before skills a graduate possesses are determined whether they meet the thresholds of various roles (Srikart & Aggarwal, 2014). This process results in matching of graduate's skills to infinite number of industry roles, both known and unknown. Due to the need to assist graduates and employers match correctly skills to available and known industry roles and predict job suitability or performance capability, classification approach seemed as the only approach that was viable to achieve this goal because of: 1) its ability to produce known class label predictions, and 2) its state of the art classification models that improve accuracy of results.

However, existing ML classification models for skills mapping are based on flat classifiers, despite possibility of underlying structure of industry roles being hierarchical as observed in organizational structures. Flat classifiers are classification models whose underlying structure of target classes ignore relationships between classes and predict only leaf classes. Apart from their inability to handle non-mandatory leaf class prediction problems, either they commit more serious errors or are not as accurate as hierarchical classifiers (Silla & Freitas, 2011; Merschmann & Freitas, 2013).

On the other hand, Wu *et al.*, (2005) note that hierarchical classifiers are designed for classification problems whose classes are naturally organized in a hierarchically structured class taxonomy. Two types of underlying ML structures used for hierarchical ML are top-down and Directed-Acyclic Graph (DAG) trees. Unlike flat, hierarchical classifiers are flexible in representing underlying structure of the problem and hence likely to achieve better accuracy levels. Despite these benefits, they have not been applied in skills mapping to industry roles. However, in other domains where they have been applied, underlying ML structure of classes not only contradicts underlying structure of the problem but also the results have been subject to multiple class labels problem hence may not be reliable.

Consequently, the real challenge in skills mapping is how to map graduates' skills to underlying hierarchical structure of industry roles as reflected by the four types of structures used to organize industry roles, namely functional, geographical, product, and matrix (Malone, 2011). Analysis of these four organization structures against the two ML structures (trees) available for hierarchical ML revealed no tree could be used to describe all four organization structures at once. Ideally, top-down tree is suited well for only functional, geographic, and product structures while DAG tree is suited well for only matrix structure. So, we do not know a ML methodology that maps skills to a hierarchical tree that correctly reflects the hierarchy of industry roles.

1.2 Statement of the Problem

The problem of mapping graduates' skills to industry roles using machine learning techniques has remained a challenge due to both non-relevant attributes and lack of appropriate machine learning structure that correctly reflects the hierarchy of industry roles. This situation may cause poor matching of graduates' skills to industry roles and possibly lead to a mismatch problem. The mismatch problem has negative impact not only to graduates of low job satisfaction but also to employers of high employee turnover and low productivity.

Despite rapid development in information technologies, a practical way of mapping graduates' skills to industry roles is a challenge. This is evidenced by large number of graduates holding jobs that do not make best use of their skills, 70% in Sub Saharan Africa; 35% in Europe (ILO, 2015). Although attempts have been made by posing this as a multi- classification problem and solving using machine learning techniques, existing approaches use both a broad range of non-relevant attributes that are industry domain dependent and flat classifiers whose classification methodology does not correctly reflect the hierarchy of industry roles (Srikat & Aggarwal, 2014; Shashidhar *et al.*, 2015), and hence, their results (82% and 60% respectively) may not be reliable.

Currently, flat classifiers used for skills mapping either may not be accurate or commit more serious errors than their hierarchical counterparts. Hence, exposing not only graduates to a threat of low job satisfaction but also employers to the risk of low productivity and high employee turnover. Although hierarchical classifiers are more accurate than flat classifiers, they have not been used in skills mapping. However, in other domains where they have been used, the underlying machines learning structure contradicts the underlying structure of the problem, and have often resulted in possibly unreliable results.

Therefore, we do not know an effective machine learning model with relevant attributes that maps graduates' skills to industry roles and that correctly reflects the hierarchy of industry roles. Our main challenge is, therefore, to develop a machine learning model with both relevant attributes and underlying machine learning structure that correctly matches the hierarchy of industry roles. The skills mapping model will benefit not only graduates by providing both feedback on job suitability and credentials to signal employability but also employers by providing an easy way to filter candidates before interviews.

1.3 Objectives

1.3.1. General Objective

To build a data driven model using machine learning for mapping graduates' skills to hierarchically structured industry roles.

1.3.2. Specific Objectives

1) To establish concepts appropriate as machine learning attributes for mapping graduates skills to occupational industry roles
2) To establish structural characteristic of concepts that correctly reflect the hierarchy of industry roles required as target classes for machine learning process
3) To build using these concepts an appropriate machine learning model that maps graduates' skills to hierarchically structured industry roles
4) To evaluate the performance and validity of the machine learning mapping model

1.4 Research Questions

1) What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?
2) What is the structural characteristic of concepts that correctly reflects the hierarchy of industry roles required as target classes for machine learning purpose?
3) How do we build using these concepts an appropriate machine learning model for mapping graduates' skills to hierarchically structured industry roles?
4) How do we evaluate performance and validity of the machine learning mapping model?

1.5 Scope

The study investigated the content of undergraduate training programs and industry roles' requirements in a given occupational domain. The undergraduate content related to domain curriculum coverage, competence skills tested as reflected in the exams past papers and student performance in domain related subjects. Industry role requirements related to job descriptions/competence requirements for various categories of domain job titles. The research was conducted in Kenya and a case of Software Engineering was used as an industry domain.

1.6 Significance of the study

The findings of this study are expected to benefit universities, industry, the government, and students. This is in attempt to reduce both low job satisfaction and long term unemployment that is one of the causes of social and economic pain both in Kenya and around the world. More specifically, Universities and the government as stakeholders in education and training will get a better understanding of the gap between the academia and industry and can use this information to plan on how to bridge the gap using the mapping model.

On the other hand, the industry will benefit by getting evaluation tool for revealing information on graduates' suitability for employment which they can use for decision making when filtering candidates for interview. Finally, students will benefit by being able to get an insight on the industry roles they are suitable at, hence empowering them to conduct informed search for jobs and lead to the right job fix. Right job fix is the ultimate goal the researcher intends to achieve in order to lower the risk of low job satisfaction, high employee turnover and low productivity.

The expected results of this study constitute a number of products that would contribute significantly both in the world of knowledge and research. These include: 1) a conceptual model for tackling the problem of mapping graduates' skills to hierarchically structured industry roles, 2) a machine learning model for predicting new graduates' suitability to industry roles, 3) a taxonomic structure that is friendly to hierarchical classification methodology, 4) a framework for mapping industry roles to hierarchically structured class taxonomy 5) machine learning datasets for experimenting hierarchical classification algorithms, 6) a software prototype that can be used by both academia and industry in assessing graduates' skills vis-à-vis industry roles during training and recruitment respectively.

1.7 Assumptions of the study

The following assumptions were made in the study:

1) Entry level occupational industry roles have different requirements for skills proficiency levels
2) Content coverage in the exam paper directly reflects content coverage during training.
3) Questions model in the exam paper reflects competencies tested during training.
4) Student class performance in domain technical subjects reflects the level of competence required to perform technical tasks.

1.8 Thesis Overview

The rest of this thesis is organized as follows: chapter 2 presents a detailed review of literature focusing first on trends of knowledge and skills required by industry, then a mismatch gap between industry and academia, followed by evaluation frameworks and methods of knowledge and skills competences, then a review of machine learning and its relevance to automatic skills mapping, and finally analysis of theoretical frameworks that form the basis for derivation of the conceptual model. Chapter 3 outlines the research methodology adopted while chapter 4 presents modeling results and findings. Chapter 5 presents the software methodology adopted in the design and implementation of the software prototype of the proposed machine learning model. Chapter 6 presents both the evaluation results and discussions of the evaluation findings while chapter 7 concludes by highlighting not only the main contributions and limitations but also achievements of this study.

CHAPTER 2: LITERATURE REVIEW

2.0 Introduction

Employment suitability of skilled graduates in the industry has become a challenge due to both increased skills variation among graduates and among industry roles, and evidenced by the industry academia mismatch gap. This chapter not only reviews background literature on knowledge and skills trends in the industry and academia, the industry academia mismatch gap, evaluation of graduates skills through mapping to industry roles, and machine learning techniques but also examines how skills mapping problem can be viewed computationally as a pattern recognition problem where Machine Learning (ML) can play an important role in addressing the challenge.

This chapter is organized as follows: Section 2.1 presents a review of knowledge and skills trends. Section 2.2 reviews issues of industry academia gap. Section 2.3 provides a review of evaluation and mapping of graduates' knowledge, skills, and competences, an introduction to ML classification methods and algorithms, reviews the past, present, and proposed techniques. Section 2.4 & 2.5 review mapping models. Section 2.6 presents a synopsis of literature review. Section 2.7 outlines the theoretical frameworks for skills evaluation. Section 2.8 concludes the chapter with a summary.

2.1 Trends

Although this section reviews trends in the industry with a special focus on Software Engineering (SE), the researcher remains optimistic that same trends can be generalized in other domains. It is equally important to note that the objective is to generally show how research studies are biased towards skills trends in the industry at the expense of skills trends in the academia towards industry roles and hence failing to effectively highlight the industry academia mismatch gap.

Globally, extensive research has been made in the area of ICT trends towards industry roles. Houghton (2012) highlights these digital trends and attributes these exponential changes in industry technology to the pressure exerted by industry roles' demand due to expansion in population, improvements in human wealth and health, and climate change. The relationship between ICT trends and demand to industry roles can be likened with the famous Moore's law, which predicted that the number of transistors of an affordable C.P.U would double every two years, such that every time population doubles so is the demand for industry roles and change in technology. In addition, Walter (2005) highlights the Kryder's law that predicts doubling of hard drive storage space in every 1-2 years.

However, according to Kanellos (2003) the current chip technology used on C.P.U and hard drives is based on silicon technology which is now approaching the limit of physics of shrinking the size of transistors on the chip. According to Kaku (2012), unless a new technology is developed to replace the silicon technology then both the Moore's law and Kryder's law are going to collapse. Kaku cites that a number of technologies to replace silicon have been proposed including nanotechnology which will be used to produce protein computers, DNA computers, Optical computers, Molecular computers and Quantum computers.

On the other hand, there is increasing criticality of software within systems and this has put an increasing demand not only for software products but also manpower onto 21st century systems (Boehm, 2005). According to Boehm (2005), systems and software engineering processes will evolve significantly over the next decades in order to address the need to design and develop not only software products but also industry roles that incorporate new technologies. He highlights the following eight trends in SE industry: integration, usability, dependability, rapidity, connectivity, interoperability, complexity, and autonomy. These trends predict a lot of job requirements changes expected in the industry that academia should take into account when preparing graduates.

As a result, educational institutions are currently experiencing a lot of challenges to change the way they educate software developers due to the way software evolves and is developed in the industry (show, 2000). According to Show, education for software developers that is currently emphasizing on content taught in the traditional way and inspired by closed-shop development model of software has failed to produce the supply and quality of developers needed to satisfy the growing demand of software. He underlines four key challenges facing educators for software developers which are: education for software developers should prepare students differently for different roles, infuse a stronger engineering attitude in curricula, help students stay current in the face of rapid change, and establish credentials that accurately reflect ability.

To address these challenges, educators need to understand which skills are important for software developers and their changing trends so that they can align their curricula accordingly. Surakka (2005) analyzed the trends of job advertisements to find out the most common technical skills sought in various software developers roles and identified five common skills for software developers: platform skills, database skills, networking skills, distributed technology skills and programming skills. According to Surakka, over the past 35 years the technical requirements for software developers have changed significantly, the number of required individual skills has increased and duties of software developers have also changed.

There is little research evidence in Kenya (0 studies) and Africa (12 studies), (outside Africa (600 studies)) relating to graduates' destinations after university, interventions in universities to improve employability and their effectiveness, and attributes that promote performance in the job (McCowan *et al.*, 2016). A lot of research is focused only on trends in the industry while trends in knowledge and skills covered during training in the academia towards industry roles still remain unnoticed.

In conclusion, trends in the industry indicate significant evolution of technologies that demand strong problem solving skills and, equally, evolution of skills requirements for professionals (Show, 2000; Boehm, 2005; Surakka, 2005; Houghton, 2012). Long term trends have been towards jobs requiring more education and cognitive skills, but the precise levels and kinds of skills are poorly understood by graduates (Handel, 2012). Currently, there is no study that indicates the trend of problem solving skills transferred to and acquired by graduates during training towards industry roles.

2.2 Industry Academia Gap

ILO (2015) reveals a large number of graduates holding jobs that do not make best use of their skills (70% in Sub Saharan Africa; 35% in Europe). Therefore, this section reviews literature and studies that have previously worked on the industry academia mismatch with a special focus on the methods used or proposed to evaluate or bridge the gap. The aim is to propose an improved method for bridging the mismatch gap that is more promising than previous methods.

A study by OECD (2012) reveals unemployment rate of ICT specialists all over the world was on a gradual increase, with 2% in 2007 and 6% in 2010, 2012). IDC study in 2009 in 13 European Union countries, observes that graduates are educated but not trained in the commercial world; they do not have the latest and appropriate technology skills; they have a good foundation but do not have skills for the market (Kolding & Ahorlon, 2009). Most of the graduates from school do not have skills for technologies that are used or required in the industry.

Moreno *et al.* (2012) reveal that curricula in the academia do not deliver all or the minimum knowledge and skills prescribed by the industry. They evaluated the relationship between SE education and industry needs using career space report of 2001 as a source of industrial needs, while SE2004 curriculum guideline for undergraduates and SE2009 curriculum guideline for graduates as a source for SE education. They examined whether the two curricula provided knowledge that was useful for performing tasks identified by career space report that related to software and application development, software architecture and design, and IT business consultancy. They observed that neither of the curricula delivered the knowledge of all tasks, and therefore were some gaps in the curricula. However, they did not indicate the minimum required by the industry.

Saiedian (2002) in his study, bridging academic education and industrial needs, observes key issues that are identified by researchers as challenges between education and industry, and proposes to bridge the mismatch gap through industry academia collaboration. Among these being reluctance of education community to introduce component-based principles of templates, specification and reasoning in introductory undergraduate classes either because they are too difficult for freshmen to understand or they might displace other principles taught in introductory courses.

Shkoukani (2012) proposes a model to find the mismatch gap between academia and industry that consists of three independent variables and one dependent variable. The dependent variable consists of well qualified graduates, while independent variables include solid courses and resources availability, academic staff capabilities and properties, and well equipped laboratories and adequate tools. The findings indicate that there are no qualified SE graduates. Hence, there is a mismatch between industry and academia. However, their study did not include student academic capabilities as this also may equally contribute to graduate qualification.

Ludi & Collofello (2001) observe a mismatch between academic projects and industry prescribed knowledge and skills for real projects. They analyzed the gap between the knowledge and skills learned in projects and those required in real projects. Their technique involved mapping a SE project course to SWEBOK content and Bloom's taxonomies' skills. The findings reveal, although most of the SWEBOK topics are covered to some extent, there exist several gaps between the level of knowledge expected from SWEBOK and the project course. However this study was limited to project course which is only one source of SE skills.

We conclude that although many studies reveal there is a mismatch gap between academia and industry, none has been able to show that one of the underlying causes of the gap is poor evaluation of problem solving skills of graduates by the industry and academia (Ludi & Collofello, 2001; Saiedian, 2002; Kolding & Ahorlon, 2009; Shkoukani, 2012; Moreno *et al*, 2012; OECD, 2012; McCowan, 2016). Studies on evaluation of graduates' skills indicate problem solving skill is poorly evaluated (Griffin, 2008; Sutherland *et al.*, 2009; Norwood & Briggeman, 2010) hence causing industry academia mismatch gap.

Our attempt was to solve the mismatch problem between industry and academia through evaluating and mapping not only content knowledge and skills gained during training, but also academic capability of the student to learning, towards job performance competences. We also put great focus on the industry minimum requirements of knowledge and skills to perform the industry roles.

2.3 Evaluation and Mapping of Graduate's Knowledge, Skills, and Competences

The aim of this section is to highlight not only how graduate skills in the industry and academia are evaluated, but also what kind of skills and competences that are evaluated and sought for by the industry. This is import because it can provide insight on the fundamental components or attributes that the industry seeks from graduates. Competence is a useful concept in bridging the mismatch gap between industry and academia. Sandberg (2000) defines competence as attributes possessed by workers, typically represented as knowledge, skills, and abilities and personal traits, required for effective work performance. Employers usually describe their job requirements in terms of competences, while academia provides qualifications and certification tests as evidence of knowledge and skills acquired during training (CWA1654, 2012).

Extensive efforts have been made to evaluate graduates' skills through mapping qualifications and certifications to job competences in the industry but with no success. For example, Korte *et al.* (2013) produces a prototype of a model to map certifications based competences to competences in the industry jobs. Although the mapping method is not clearly shown in the study, they report a challenge of a reliable formula to combine competences in order to understand the overall capability of the graduate.

There is also confusion among students and graduates in understanding employers' preferences, with some being underestimated or overestimated by students (Hansen & Hansen, 2007). For example, Belcheir (1996) as cited by Norwood & Briggeman (2010) reveals that Boise State University understood properly the importance of communication skills to employers but overemphasized the role of problem solving skills and underemphasized the value of interpersonal skills. Again, showing there is a problem with the reliability of the formula to predict employers' preferences.

Quintin (2011) in their study in OECD countries reveal 25% of workers are overqualified while 20% are under-qualified. They further reveal a challenge to employers in screening job competences through graduates with same formal qualifications, as workers with same qualification level may portray different degrees of competence. Competence assessment methods used for graduates by employers in the industry are different and most common are interviews, grades, and awards. However, many of them do not express the actual worker's value or attribute that organizations prefer, but instead only signal those values or attributes. A survey by Norwood & Briggeman (2010) reveal that interview is the most used method by employers to signal every attribute they prefer of a graduate, then followed by others like grades, course taken, major, etc.

Most studies seek to know methods and competences employers prefer to assess graduates. Sutherland *et al.* (2009) reveal five competences that must be offered side by side with content knowledge during training: problem solving, critical thinking, communication, collaboration, and adaptive learning. They show that learning based on content knowledge only encourages memorization at the expense of deep conceptual understanding of core ideas, generalizable principles, and knowledge that can be applied in new situations.

Since universities offer flexible degrees with diverse experiences as learning outcomes, they use a wide range of assessment methods including formal examination, laboratory reports, problem-solving exercises, presentations, and project work. However, there is no adequate cross check made to ensure that some learning outcomes are not over tested at the expense of others which may not be tested at all (Karl *et al.*, 2009). Although Colvin (2007) cite that some courses taught by different professors may vary in content and emphasis, Karl *et al.* (2009) reveal that the cognitive skill level examined by exam questions remains relevant to the cognitive skills. But still, assessment of examinations tends to vary from grader to grader because there is no underlying framework of reference.

We conclude, therefore, that a number of issues that may arise in the evaluation and mapping of graduates' skills: Content knowledge evaluation may not be adequate, and therefore we may need to also evaluate competences (Sutherland *et al.*, 2009); Qualifications and certifications alone may not adequately communicate graduates' skill possession (Quintin, 2011); Manual grading may be subjective (Colvin, 2007; Karl *et al.*, 2009); there may not be reliable formula to combine competences to predict and indicate overall capability of graduate (CWA1654, 2012).

As a way forward, there was need to explore a number of strategies that could provide focus to deep understanding of the solution requirements based on the existing knowledge. For example, to perform job tasks properly in the industry core technical knowledge (content knowledge) received during training and experience are key requirements, although experience is acquired with practice on the job (Moreno *et al.*, 2012). Sutherland *et al.* (2009) note that learning content knowledge alone makes it difficult to apply the knowledge in unfamiliar context away from the context in which it was learned, and this would promote memorization. However, if the goal is to apply the knowledge in unfamiliar context outside classroom, such as in the job, then content knowledge should be accompanied by some competences that promote deep understanding and generalizable principles.

2.3.1. Relationship between Content Knowledge and Competences

2.3.1.1. Communication

Content knowledge is required to provide logic and evidence to explain a task i.e. a good command of content knowledge is required to do so. Baker & Mayer (1999) posit that one of the cognitive tasks of content understanding is explanation which involves illustrating an argument by applying the relevant prior knowledge and writing in an organized way that avoids misconceptions (Mayer, 2002).

2.3.1.2. Collaboration

Collaboration helps in sharing, clarifying, and distributing content knowledge among peers. Collaboration cannot occur without communication. Both communication and collaboration increase understanding, retention, and expression of content knowledge (Mayer, 2002). They both add value to content knowledge.

2.3.1.3. Critical thinking

Critical thinking refers to deep thinking required to tightly connect discrete pieces of content knowledge to produce integrated content knowledge. Mayer (2002) outlines four types of knowledge as factual, conceptual, procedural, and meta-cognitive. While factual and procedural are low level knowledge, conceptual and meta-cognitive are higher level knowledge that involve connecting pieces of knowledge together to enhance or demonstrate better content understanding (Baker & Mayer, 1999; Mayer, 2002).

2.3.1.4. Adaptive learning

Adaptive thinking refers to the ability to actively use ones cognitive resources to regulate ones thinking in order to improve understanding of integrated content knowledge with an aim of creating new content. According to Mayer (2002), meta-cognitive involves knowing strategies for doing tasks, knowing demands for tasks, and knowing ones' own capabilities towards a task. This promotes creating new content or strategies.

2.3.1.5. Problem solving

Problem solving involves application of integrated content knowledge in a new context. Problem solving involves critical thinking in relation to a problem while adaptive learning controls and regulates thinking about a problem. So, critical thinking and adaptive learning support problem solving.

We conclude, from this analysis, that communication and collaboration are subordinate to content knowledge. Likewise, critical thinking and adaptive learning are subordinate to problem-solving. Therefore, the most important and useful relationship to evaluate is content knowledge and problem solving relationship. Robertson (1990) reveals high correlation between conceptual understanding of content knowledge and transfer of problem solving skills. In the study, they claim that concept understanding is the main predictor of performance in transfer problems and there is a cognitive structure associated with that successful performance. This is in concurrence with earlier studies that also reveal that cognitive connections within a person's memory structure promote understanding and enhance performance on transfer problems (Ausubel, 1968; Gagne & White, 1978).

However, the index Robertson (1990) uses for understanding is not clearly understood what it reveals and therefore cannot be interpreted. This is because the index is a very poor predictor of performance in familiar problems in the written exam, and also is not correlated with overall performance in the written exam.

2.3.2. Skills Evaluation Frameworks

Content knowledge is usually the main source of domain-specific knowledge (declarative knowledge and procedural knowledge, also known as domain-specific strategies). Content knowledge can be evaluated using the body of knowledge provided in the academic discipline or competence framework provided in the industry.

Each academic discipline has a body of knowledge that all graduates ought to acquire during training (Calvin, 2007). Krishnan (2009) characterizes every academic discipline with a body of accumulated specialist knowledge referring to their object of research. In their study, on analysis of the gap between the knowledge and skills learned in academic course project and those required in real projects, Ludi & Collofello (2001) used Software Engineering Body of Knowledge guide (SWEBOK) as the framework to evaluate the industry academia mismatch gap.

Problem solving can be evaluated using competence framework which provides skill areas, competences, and proficiency levels to which every certification or qualification can be mapped (Korte *et al.*, 2013). This then splits problem-solving into three dimensions: skill area, competence, proficiency level. Each problem-solving area consists of a number of skill areas, and each skill area requires a number of domain specific competences. Now, each competence is scaled into several proficiency levels. Korte *et al*. (2013), in their study, use e-Competence Framework (e-CF) to evaluate the skill value of industry based certifications.

Therefore, competence framework defines a set of skill-based competences needed by all students entering the industry profession. Some frameworks that may be relevant to this study have been described below. Since the domain of academic librarians was used as a validation case for our model there was need to also discuss its framework.

2.3.2.1. SWEBOK Guide

The industry accepted SE knowledge and skills required of a qualified software engineer are provided under the Software Engineering Body of Knowledge (SWEBOK) curriculum guideline. There are two versions of SWEBOK guide for both undergraduate and graduate students. These SE curriculum guidelines are provided in SE2004 and GSWE2009 under the joint effort of IEEE/ACM for both graduate and undergraduate students respectively (SE, 2004; GSWE, 2009). The two curriculum guidelines are used as *defacto* standards for the knowledge and skills expected of a professional software engineer (Merono *et al.*, 2012), and they constitute planned curriculum (Pideaux, 2003; Kenny & Desmarais, 2010).

According to Abran *et al.*, (2006), the purpose of SWEBOK guide is to describe what portion of the body of knowledge is generally accepted and to provide topical access to it. The actual body of knowledge already exists in published literature provided as reference materials in the guide. SWEBOK is just a guide that can assist in the development of curriculum as each Knowledge Area (KA) is decomposed into topics, and knowledge depths of each topic are rated using Bloom's Taxonomy (Ludi & Collofello, 2001).

SWEBOK guide is a joint product of a continued collaboration between industry, academia and standard setting bodies all over the world (Ludi & Collofello, 2001). Abran *et al.* (2006) cite that so as to get a worldwide consistent view of SE, the first version 2001 guide was developed through a process that engaged about 500 reviewers from 42 countries while the second version 2004 guide engaged over 120 reviewers from 21 countries from North America, Pacific Rim, and Europe.

SWEBOK guide provides the following ten Knowledge Areas (KA) that define the SE profession for undergraduates, and considered as core knowledge for all software engineers (Ludi & Collofello, 2001):

1) Software Configuration Management
2) Software Construction
3) Software Design
4) Software Engineering Infrastructure
5) Software Engineering Management
6) Software Engineering Process
7) Software Evaluation and Maintenance
8) Software Quality Analysis
9) Software Requirements Analysis
10) Software Testing

According to Abran *et al.* (2006), the reference material for each KA is provided in the form of book chapters, referenced papers or other recognized sources of authoritative information. Further, the guide recognizes eight related disciplines that software engineers should have knowledge from and each KA description may make reference to. Since it is a result of a process of domain experts review and validation, SWEBOK is not only a good foundation for creating SE curriculum (Ludi & Collofello , 2001; Abran *et al.*, 2006) but also for creating a skills mapping model for software engineers in this study.

2.3.2.2. European e-Competence Framework

European e-Competence Framework (e-CF) is a common European framework for ICT professionals in all industry sectors created in 2008 (version 1) and 2010 (version 2). It provides a reference of 40 competences as required and applied at ICT workplace, using a common language for competences, skills and proficiency levels that can be understood across Europe (www.ecompetences.eu). It is an implementation of the European Qualification Framework (EQF) for application in the ICT sector by all stakeholders.

Korte, *et al.* (2013) cite that EQF is the overall qualification framework for the European countries agreed in 2008 and its intention is to help make national qualifications more transferable across Europe by relating national qualification systems to a common reference framework. The e-CF, on the other hand, is an effort of the need for standardization and guidance to ICT practitioners (students or experienced) in their performance, training and development in European countries (CWA16458). Basically, e-CF is used to support the definition of jobs, training courses, qualifications, career paths, certifications etc in the ICT sector.

The e-CF framework provides a three dimensional views, namely skill areas, competences, and proficiency levels to which every certification can be mapped. The 40 competences of the framework are classified according to five main ICT business areas and relate to EQF. To support e-CF application within multiple environments, a series of case studies have been carried out including the following:

1) e-CF for ICT professional self-assessment
2) e-CF for assessment and career tools

Although e-CF has been used successfully to create European ICT job profiles, the following challenges have been reported:

1) how to combine competences using a reliable formula to indicate the overall capability of a candidate
2) how to verify the competences claimed by ICT professionals
3) e-CF is a high level description of competences and does not take into account the granularity levels of individual job competences
4) e-CF requires the combined use with other frameworks or educational achievements

2.3.2.3. Professional Knowledge and Skill Base (BPKSB) Framework

The knowledge and skills necessary for academic librarians are captured in the Body of Professional Knowledge described as Professional Knowledge and Skills Base (PKSB). PKSB was created by the Chartered Institute of Library and Information Professionals (CILIP) and, according to Nagata *et al*. (2006), describes the knowledge base that distinguishes information professionals in three concentric circles. The framework describes a total of 11 areas of knowledge and skills necessary for professional academic librarians as outlined below:

1) Traditional services

2) Books and libraries
3) New services
4) Organization of information
5) Collection building
6) Library standards and networks
7) Information flow/publishing industry
8) Communication
9) IT technology
10) Business administration
11) Foreign languages

The framework divides the knowledge and skills areas into three groups, core schema (1-5), application environment (6-7), and generic and transferable services (8-11).

We conclude, from the above analysis that frameworks need to be used as references for skill evaluation (Srikant & Aggarwal, 2014) in order to reduce assessment variation from grader to grader. However, frameworks provide skill transparency only but not the entire solution to variation problem from grader to grader, cost of hiring graders, or evaluation time wasted during grading. And therefore, automatic skill evaluation can greatly provide a reliable solution and formula for combining competences to predict overall capability of a graduate during both training and recruitment processes of industry and academia (Srikant & Aggarwal, 2014).

2.3.3. Automatic Skills Mapping

Variations in assessment from grader to grader has made automatic skills evaluation and mapping a hot topic of keen interest both in the recruitment process of industry and training process of academia (Srikant & Aggarwal, 2014). This is an attempt to greatly lower the cost of hiring, reduce time wasted and provide a standard way of graduate assessment. Due to wide availability of data globally, data driven methods, such as machine learning techniques, have become popular. Machine learning classification methods and algorithms may provide a reliable formula for combining competences to indicate or predict overall capability of a graduate.

2.3.3.1. Machine Learning Classification Methods

Machine Learning (ML) is one of the major branches of Artificial Intelligence (AI) that is concerned with designing programs (ML algorithms) that attempt to make computers behave intelligently by being able to sense, remember, learn, and recognize patterns (Leeuwen, 2004). Through the years, major branches of ML have emerged including symbolic learning by Hunt *et al*. (1966), neural networks by Rosenblatt (1962), and statistical learning by Nilsson (1965). In each of the ML branches there has been a rapid development of ML algorithms, although majority of them face so many challenges. ML algorithms are designed to analyze a known data set so as to discover and extract knowledge rules from the data set through building a classifier that can map or predict group membership of unknown data instances.

Machine learning problem can be defined as the problem of improving some measure of performance when executing some task, through some kind of training experience (Jordan & Mitchell, 2015). Task can be of assigning a label to an item, performance to be improved could be accuracy (or speed) of doing this task and training experience could be historical data of the item with labels. Traditionally, the task can be modeled as a function (f), where learning problem is to improve the accuracy of the function and training experience consists of a sample data of known input-output pairs (x,y) of the function.

In many machine learning setups, the goal is to learn the function *f* such that:

$$f: x \longrightarrow y \qquad \text{eqn (1)}$$

Where x ϵ X are inputs while y ϵ Y are outputs. The goal of learning *f* is to improve its performance accuracy through function approximation or optimization procedures and is achieved using various machine learning algorithms. Conceptually, machine learning algorithms are viewed to be searching through a large space of candidate functions that optimize the performance metric, guided by the training experience (Jordan & Mitchell, 2015). Depending on the kind of output (discrete or continuous) the candidate function is called a classifier or regression function respectively.

ML is used to solve problems through a number of methods including segmentation, feature extraction, classification, clustering, regression, modeling, etc. There are three main categories of machine learning methods: supervised, unsupervised, and reinforced learning methods. Classification is one of the machine learning methods used to predict group membership for data instances (Mehtani, 2011). The groups, also known as classes, are either predefined (supervised classification) or are learned based on similarities (unsupervised classification) or rewards (reinforced classification) (Basu *et al.*, 2010; Raschka, 2015).

2.3.3.2. Supervised Classification Method

This is the construction of a classification procedure from a set of data for which the true classes are known (Mitchie *et al.*, 1994), and is sometimes referred to as supervised learning, pattern recognition or discrimination. The main objective of supervised classification method is to establish a classification rule from a given correctly classified data, or to construct a learning model from labeled training data set so as to be able to classify new objects with unknown labels (Mehra & Gupta, 2013). Supervised classification methods are further sub-divided into parametric and non-parametric depending on whether the data follows a specific distribution or not.

The supervised classification method (also known as supervised machine learning) consists of the following main elements (Kotsiantis, 2007):

1) Identification of required data
    - Involves identifying the most informative features
    - Methods which can be used include: experts, brute-force
2) Data pre-processing
    - Involves removing noisy features to enhance learning from very large data set
    - Methods used include: instance selection, features subset selection
3) Algorithm selection
    - Involves comparing two or more supervised learning algorithms
    - Methods used include: statistical comparisons, paired t-test
4) Training
    - Involves teaching the model with a sample of existing correctly classified cases
    - Methods used include: Artificial Intelligence (AI), Neural Networks, Statistical techniques, Support Vector Machines (SVM)
5) Evaluation
    - Involves running the trained model with a set of classified cases it has never seen before so as to see whether it will classify correctly or not

2.3.3.3. Unsupervised Classification Method

This is the construction of a classification procedure from a set of data for which the true classes are unknown but are inferred from the data set (Mitchie *et al.*, 1994), and is sometimes known as clustering. This method can be viewed as aiming to identify natural groups or classes or clusters in the data.

2.3.3.4. Reinforced Classification Method

This is the construction of a classification system that improves its performance through interaction with the environment (Raschka, 2015). This method can be viewed as aiming to establish a classification rule based on a reward signal in the environment. Reinforcement learning is related to *supervised* learning where instead of the correct ground truth label or value, we have a measure of how well the classification action was measured by a *reward* function.

2.3.4. Machine Learning Algorithms

ML algorithms are usually designed around a particular paradigm for the learning process which must be clear about the learner, domain, goal, representation, algorithmic technology, data source, training scenario, prior knowledge, success criteria, and performance (Leeuwen, 2004). As a result, Kotsiantis (2007) indicate that classifier design must be based on assumptions made about the classification problem and the training sample used to teach the classifier. This is because the predictive power of the classifier is largely dependent on the quality and size of the training sample. However, determining the termination point for the training is still a challenge (Figueroa, *et al.*, 2012) and this can lead to over fitting.

Preliminary survey revealed three categories of supervised machine learning algorithms/techniques: Logical/symbolic techniques (Artificial Intelligence), Perception-based techniques (Neural Networks), Statistics-based techniques (statistical methods), and support vector machines technique. However, many ML algorithms suffer challenges in terms of algorithmic approach, data representation, computational efficiency, and quality of the resulting classifier (Kotsiantis, 2007). This triggered review of various ML algorithms to reveal various ways they could be improved.

2.3.4.1. Back Propagation Algorithm

Under neural networks, Rosenblatt (1962) developed a basic delta learning rule for Single Layered Neural Network (SFNN). Minsky and Papert (1969) proved that this rule could not solve non-linear problems. Rumelhart *et al.* (1986) developed the Back Propagation Algorithm (BPA) for Multi-Layered Feedforward Neural Networks (MLFNN). BPA is based on gradient descent search method that it uses to adjust the connection weights in MLFNNs. According to Kononenko (2001), although

BPA is well known for its accuracy, it suffers a problem of slow convergence and local minimum problem.

Survey by Vora and Yaguik (2013), reveal extensive research proposing various ways of solving BPA problems and improving its performance such as replacing its gradient descent method with momentum and delta-bar-delta method; modifying coefficient of correlation between prior weight change and downhill momentum factor; choosing a network weight upgrade rule; choosing a series of weight vector over learning phase; multiplying connecting weight by a factor; changing the derivative of the learning function; updating learning rate and inertia factor dynamically; summing linear and non-linear quadratic errors of the output neurons; adjusting learning rate and momentum factor at each iteration; introducing activation function of neurons in hidden layer in each training pattern; combining non-linear regression with; training hidden and output layers independently; combining linear least squares with gradient descent; combining BPA with genetic algorithm.

Even though several variations and different techniques have been suggested to improve performance of BPA none guarantees a global solution and, therefore, the problem of slow convergence and local minimum is yet to be solved.

2.3.4.2. Support Vector Machines Algorithm

Support Vector Machines (SVM) is a learning algorithm invented by Vladmir Vapnik in 1995 and is used in many fields of pattern recognition and classification of data. SVM is based on convex quadratic programming. Although SVM has emerged as a good classification technique and has achieved excellent generalization performance in a variety of applications, it suffers a problem of bad memory utilization and long training time as the number of training examples increase (Wang, 2015).

Recent survey by Wang (2015) reveals extensive research in SVM, and a variety of ways for improving the SVM problems have been proposed. These include decomposition-based approaches that consider a small subset of variables in each training iteration; alpha seeding approaches; adding a constant to the objective function; using conjugate gradient scheme; informative instance selection for training; incremental learning; using Finite Newton Method.

Given N input elements and two disjointed output classes, the goal of SVM is to take the input elements, learn them, and predict if each of them belongs to one of the two classes.

Given a training set, S= {(x1,y2),(x2,y2),….(xN,yN)}, SVM learning algorithm involves building a model that maps new instances of X to Y. Geometrically, the function, f, represents the hyperplane of all possible planes that are able to correctly classify the input elements.

For linear model, f is given by,

$$f(x) = (w.x) + b \qquad \text{Eqn (2)}$$

Where x ϵ X are inputs while y ϵ Y are outputs.

Finding the function f involves modeling of this hyperplane by learning two parameters w and b that maximize the distance between the nearest points of the two classes, i.e. that make f(x) = 0. These nearest points between the two classes are called support vectors and the distance between each point in each class and the hyperplane is called functional margin(Y) and is given by,

$$Y_i = y_i(w.x_i) + b \qquad \text{Eqn (3)}$$

The nearest points of each of the two classes are those points that optimize $Y_i = 1$, and distance between these nearest points of the two classes is given by the sum of their functional margins. Therefore, SVM learning involves finding an optimal separation hyper-plane that maximizes this sum of the two margins i.e. $Y_i >=1$, and its parameters (w, b) minimized to the lowest level possible. The solution to the above dual optimization can be summarized using the equation below which gives the hyperplane.

$$f(x) = \sum^N x_i y_i (x.x_i) + b \qquad \text{Eqn (4)}$$

For non-linear model, f is given by,

$$f(x) = \sum^N w_i \phi_i(x_i) + b \qquad \text{Eqn (5)}$$

Where $\Phi : X \longrightarrow y$ is a non-linear mapping from a high dimensional input space to a high dimensional output space. The solution to the above dual optimization can be summarized using the equation below which gives the hyperplane.

$$f(x) = \sum^N x_i y_i (\phi(x) . \phi(x_i)) + b \qquad \text{Eqn (6)}$$

Given the original input space points, calculate $(\phi(x). \phi(x_i))$ product directly in the feature space and then map the point in the feature space. To do this an instrument known as kernel is needed. There are a few functions that can be considered as kernel i.e.

1. Linear kernel, $K(x_i,x_j) = (x_i.x_j)$

2. Polynomial, $K(x_i,x_j) = [(x_i.x_j)+1]^d$ where d is not zero
3. Gaussian, $K(x_i,x_j) = e^{-||xi-xj||^2/2\Omega^2}$

2.3.4.3. Naïve Bayesian Algorithm

The theoretical basis of the naïve Bayesian algorithm and its variants was first developed by Thomas Bayes in 1964. Naïve Bayesian algorithm assumes underlying probabilistic model and allows us to capture uncertainty about the model using maximum likelihood method. Although it is simple and very powerful, naïve Bayes algorithm does not work well if there is dependency between predictor variables.

Survey by Bielza & Larranaga (2014), reveals extensive variants and extensions of the naïve Bayesian classifier focusing towards detecting and handling dependency between predictor variables such as the m-estimate of probabilities that significantly improved the performance of Bayesian classifier; a semi-naïve Bayesian classifier that detects dependency between attributes; fuzzy discretization of continuous attributes within the naïve Bayesian classifier; a recursive Bayesian classifier that uses naïve Bayesian classifier in the nodes of decision trees; explicit searching of dependences between attributes in the naïve Bayesian classifier; relaxing conditional independence assumption by allowing each predictor variable to depend on at most one other predictor in addition to the class; allowing each predictor variable to have a maximum of k parent variables apart from the class variable.

Given X (x_1, x2,…, xn) input attributes and W (w_1, w2,…, wm) disjointed output classes, where X and W are dependent the goal of naiveBayes is to take the input attributes, learn them, and predict conditional probability of W given X. naïve Bayes is based on the Bayesian theory which uses the knowledge of prior events to predict the future events. According to Bayesian theorem, if w_j is a hypothesis that is made over an event and x_i is the data set describing the event then:

$$P(w_j|x_i) = P(x_i|w_j).P(w_j) / P(x_i) \qquad \text{Eqn (7)}$$

Where P (w_j) is prior probability of hypothesis w_j, P (x_i) is prior probability of data x_i, P ($x_i|w_j$) is conditional probability of x_i given w_j, and P ($w_j|x_i$) is conditional probability of w_j given x_i.

Suppose x_i is a feature vector of a sample of instances i (i=1,2,…,n) and w_j be notation of class j (j=1,2,…,m), then probability of observing sample x_i given that it belongs to class w_j is called conditional probability of x_i and is given by $P(x_i|w_j)$.

Also:

$$P(x_i|w_j) = \frac{\text{No. of times } x_i \text{ appears in all instances of class j}}{\text{Total count of all values of features of instances in class } w_j} \quad \text{Eqn (8)}$$

The general probabilities of encountering a class w_j are given by counting all instances of class w_j then dividing by the total count of all instances in the training dataset and are called prior probabilities and denoted by $P(w_i)$

Also:

$$P(w_j) = \frac{\text{Count of all instances of class } w_j}{\text{Total count of all instances in the training dataset}} \quad \text{Eqn (9)}$$

While the general probabilities of observing an instance x_i independent from class labels are given by adding probabilities of x_i given w_j and probabilities of x_i given not in w_j

Also:

$$P(x_i) = P(x_i|w_j).\ P(w_j) + P(x_i|\sim w_j).\ P(\sim w_j) \quad \text{Eqn (10)}$$

However, we make some assumption that x_i are independent and identically distributed so that x_i are independent and drawn from similar probability distribution such as normal probability distribution. Hence, using this probability distribution we can easily calculate prior probabilities of $x_{i.}$

2.3.4.4. Logistic Regression Algorithm

This is a linear and binary classification algorithm that can easily be extended to multiclass classification through the one versus the rest (OvR) technique (Raschka, 2015). The principle of logistic regression is based on the ratio of probabilities of two mutually exclusive events (y=1, y=0), where probability of y=1 is *p* and probability of y=0 is *1-p*. Then, the ratio of these probabilities also known as odd ratio is given by:

$$\text{odd ratio} = p/(1-p) \quad \text{Eqn (11)}$$

Logit probability of *p* is the logarithm of the odds ratio and is given by:

$$\text{Logit}\ (p\ (y=1)) = \log\ (p/(1-p)) \quad \text{Eqn (12)}$$

Logit function takes in values in the range (0, 1) and transforms them to the entire range of real numbers, which we can use to express the relationship between feature values and the log odd-ratio as follows:

$$\text{Logit } (p \text{ (y=1|x)}) = w_0x_0 + w_1x_1 + w_2x_2 + \ldots\ldots W_mx_m \qquad \text{Eqn (13)}$$

From Eqn (13) p(y=1|x) is the conditional probability that a particular instance belongs to class 1 given its features x which is obtained by getting the inverse of eqn (13). This inverse of logit function is given a follows:

$$\text{Inverse (logit } (p \text{ (y=1|x)))} = 1/(1- e^{-\text{logit } (p \text{ (y=1|x))}}) \qquad \text{Eqn(14)}$$

$$L(z) = 1/(1- e^{-z}) \qquad \text{Eqn (15)}$$

Where $z = w_0x_0 + w_1x_1 + w_2x_2 + \ldots\ldots W_mx_m$

L(z) which is the inverse function is called the logistic regression model

2.3.4.5. K-Nearest Neighbor (KNN) Algorithm

This is one of the algorithms which does not learn any discriminative function from the data but memorizes the training data instead (Raschka, 2012). The algorithm uses distance metric to find the instances in the training dataset that are closest or most similar to the new instance that needs to be classified. The class label of the new instance is then determined by a majority vote among its nearest neighbors. Its procedure can be summarized by the following steps:

i) Choose a number, *k, as* a distance metric.

ii) Find *k* nearest neighbors of the new instance that needs to be classified.

iii) Assign the class label by majority vote.

While the main merit of this algorithm is the ability to immediately adapt as we collect new training data, its downside is the computational complexity for classifying new instances that grows linearly with the number of instances in the training dataset in the worst-case scenario, unless the dataset has very few features.

We conclude, from the above analysis, that the classification methodology applied on a particular problem depends on the data, the model of the data, and the expected results of analysis (Bedzek, 1981).

2.3.5. Advanced ML Methods and Algorithms

2.3.5.1. Extreme Machine Learning

Extreme Machine Learning (EML) is the state of the art ML algorithm for learning a Single hidden Layer Feedforward Neural Network (SLFNN) where the hidden nodes are randomly initiated and then fixed without iteratively tuning (Huang *et al.,* 2014). The only free parameters needed to be learned are the connection weights between the hidden layer and output layer.

According to Huang *et al.* (2014), ELM is based on three learning principles: 1) learning capability i.e. can fit perfectly to any training data set so long as the number of hidden neurons is large enough and no larger than the number of distinct training samples (Huang *et al.*, 2006); 2) universal approximation capability i.e. ELM parameters are randomly generated instead of being learned and therefore does not require the activation function to be continuous or differentiable (Huang & Chen, 2007, 2008; Huang *et al.*, 2006). 3) generalization performance i.e. ELM has a relatively low VC dimension and Lin *et al.*(2012) show that the VC dimension of ELM is equal to its number of hidden neurons with probability one.

One of the major problems with ELM is demand for more neurons than conventional neural networks in order to achieve a matched performance, hence resulting in longer running time during testing. A recent review of ELM trends by Gao *et al.* (2014) reveals various proposals of ELM variants to solve the ELM problems. Incremental ELM proposes getting rid of insignificant neurons dynamically during training process using pruning techniques; Parsimonious ELM proposes recursive orthogonal least squares to perform forward selection and backward elimination of hidden neurons; tuning most of the output weights to zero using a sparse Bayesian approach.

2.3.5.2. Deep Learning

Deep Learning (DL) is a set of algorithms in ML that attempt to learn in multiple levels of modeling corresponding to different levels of abstractions in the model (Li & Yu, 2013). Key aspects that are common among these algorithms are: 1) models consisting of multiple layers, and 2) methods for learning feature representation at successively higher and more abstract layers.

Most traditional ML algorithms are based on shallow structured architectures that are effective in solving simple and well-constrained problems. However, more complicated real-world applications involving natural signals such as human speech, natural sound and language, natural images and visual scenes, are more difficult to be handled by such shallow architectures, hence calling for deep learning architectures.

Deep learning techniques are divided into two: supervised learning techniques (also known as deep discriminative models) and unsupervised learning techniques (also known as deep generative models). Deep discriminative models include Deep Neural Networks (DNN), Recurrent Neural

Networks (RNN), and Convolution Neural Networks (CNN), while deep generative models include Restricted Boltzmann Machines (RBM), Deep Beliefs Networks (DBN), and Deep Boltzmann Machines (DBM).

2.3.6. Multiclass Classification Classifiers

In ML the problem of classification is encountered in various areas such as in medicine to identify the disease of a patient or in industry to decide whether a defect has appeared or not, or whether the temperature is low, medium or high (Mehra & Gupta, 2013). In all these situations, multiclass classification is the major problem (Aly, 2005; Mehra & Gupta, 2013).

Multiclass classification is a case of the classification problem where there are many distinct classes while binary classification is a case of the classification problem where there are only two distinct classes. Many of the basic ML algorithms were developed to solve the binary classification problem (i.e. two classes case). However, majority of ML algorithms can be naturally extended to solve the multiclass classification problem (i.e. multiclass case). Extensible algorithms use different techniques such as codeword for output neurons (neural networks), adding additional parameters and constraints to the optimization problem to handle the separation of various classes (SVM). Though, a few of ML algorithms require converting the multiclass classification problem into a set of binary classification problems (Aly, 2005; Mehra & Gupta, 2013).

A survey on multiclass classification methods (Aly, 2005; Mehra & Gupta, 2013), reveals a number of methods various researchers have proposed to solve the multiclass classification problem including decomposition and hierarchical methods, apart from extensible methods. Decomposition methods involve splitting and include one-versus-all (OVA) that results to the number of binary classifiers equal to the number of classes in the multiclass classification problem, all-versus-all (AVA) that requires K(K-1)/2 binary classifiers for a classification problem with K classes, and error-correcting-output-code (ECOC) results to several binary classifiers.

2.3.6.1. Hierarchical Classifiers

Hierarchical methods involve arranging classes hierarchically into a tree and using a simple classifier at each node. According to the two surveys (Kumar *et al*., 2002; Vural & Dy, 2004; Chen *et al.*, 2004), a method that uses K-1 binary classifiers to classify K-classes problem has been proposed in the literature. Mehra & Gupta (2013) experiments with all the available multiclass classification methods on various data sets, and the results reveal that no any single method is perfect across all the data sets. The conclusion is, any one of the method can be used depending on the need. But their conclusion was based on experimental results focusing on accuracy alone. However, looking at the survey literature, multiclass classification is still a major problem area for research (Aly, 2005; Mehra & Gupta, 2013), and the following are some of the major issues:

1) If outputs corresponding to two or more classes are very close to each other those points are labeled as unclassified (OVA)
2) Memory requirement is very high in tune of the square of the total amount of training samples (OVA, AVA, ECOC)
3) Unbalanced training sample sizes i.e. ratio of training sample of one class to rest of the classes is 1:K-1 (OVA,AVA, ECOC, Hierarchical)
4) Large number of classifiers i.e. for OVA (K classifiers), AVA (K(K-1)/2 classifiers), ECOC (N classifiers where N>K), Hierarchical (K-1 classifiers).

Silla & Freitas (2011), in their survey of hierarchical classification across different application domains, define three criteria that distinguish hierarchical classification methods: 1) hierarchical structure (tree or DAG), 2) depth of classification hierarchy (mandatory or non mandatory leaf node prediction at any level of hierarchy), 3) hierarchical structure transverse (flat, big-bang, or top-down). Major types of multiclass classifiers based on the above criteria are flat, big bang (also global), and local classifiers as outlined in the following subsections.

2.3.6.2. Flat classifiers

These are classifiers that ignore class relationships and predict only the leaf nodes, and also known as bottom-up classifiers in some literatures. One disadvantage with these classifiers is inability to handle non-mandatory leaf node prediction problems (Silla & Freitas, 2011; Merschmann & Freitas, 2013). In industry roles classification problems, where some roles are intermediate to some high level roles, some employees are assigned to intermediate (non-leaf nodes) and some to high level roles (leaf nodes) and therefore during classification there is need for non-mandatory leaf node prediction. For a problem with K classes, we need K classifiers, one for each class.

2.3.6.3. Big bang classifiers

These are classifiers that handle the entire class hierarchy by being able to classify both leaf and non leaf nodes using one classifier. They are also known as global classifiers. Although their prediction accuracy is pretty good, they lack the kind of modularity for local training (Silla & Freitas, 2011; Merschmann & Freitas, 2013).

2.3.6.4. Local classifiers

Also known as top-down classifiers, local classifiers have the ability to use local information at each level of hierarchy to create a classifier. They apply different approaches for using local information and building a classifier around that information: 1) local classifier per node 2) local classifier per parent node 3) local classifier per hierarchy level.

i) Local classifier per node approach

This approach creates one binary classifier for each class node in the hierarchy except the root node. Has a disadvantage of allowing classes to be assigned to classes in distinct branches in the hierarchy, hence can lead to class membership inconsistency (Silla & Freitas, 2011; Merschmann & Freitas, 2013). Several inconsistency removal methods are available.

ii) Local classifier per parent node approach

This approach creates a classifier for each parent node in the class hierarchy with the aim of distinguishing its child nodes.

iii) Local classifier per level approach

This approach creates a classifier for each level of the class hierarchy. Has same disadvantage as local classifier per node approach of allowing classes to be assigned to classes in distinct branches in the hierarchy, hence can lead to class membership inconsistency and requires post processing procedure to correct the inconsistency (Merschmann & Freitas, 2013).

We conclude, from the above review, that hierarchical classifier is the only classifier that respects the hierarchical structure of the class taxonomy in a classification problem. It is also evident that local classifiers are synonymous to topdown classification approach (Merschmann & Freitas, 2013; Silla & Freitas, 2011). Also, despite the nature of some classification problems being bottom-up, there is little research towards bottom-up hierarchical classifiers.

2.4. Models for Skills Mapping using Machine Learning

Chien & Chen (2008) built a classification model for improvement of employee selection by predicting both retention and performance of new job applicants. They used flat ML classification structure and a total of seven demographic attributes. Although performance of their model was good (80%), the target concepts for mapping were broad. For each role, graduates were mapped not only as either 'can perform' or 'can't perform' but also as either 'retainable or unretainable', hence in two layered labels. Prediction label was a combination of layer1 (can perform or can't perform) and layer2 (retainable or unretainable) labels. This way, it was possible to have more than one industry role with similar labels hence multiple label prediction problems. Besides, their target classes were hierarchically related and, hence, better accuracy could have been achieved using hierarchical classifier despite the fact that the class labels were not directly industry roles.

Also, Jantawan & Tsai (2013) presented a classification model for predicting graduate's employability. They attempted to predict whether a graduate twelve month after graduation would be employed, unemployed, or undetermined, based on twenty one demographic attributes that influenced graduate employability identified from actual data collected from graduates twelve month after graduation. They used Bayesian and decision tree and flat ML classification structure to generate their model. Although performance of their model was good (98%), the target concepts were broad and were mapping graduate's skills as either employed or unemployed. Whereas target concepts were too broad and therefore not specific to industry roles, most of their ML attributes were not relevant to problem solving skills.

Equally, Shashidhar *et al*. (2015) developed a classification model to predict employability by mapping graduate's skills to software engineer's role. Their underlying ML classification structure was flat with a total of four attributes for machine learning. Although performance of their model was good (82%) and their ML attributes were relevant to problem solving skills, their target concepts for mapping were broad and were mapping graduate's skills as either satisfactory or unsatisfactory. Besides, it was possible to have more than one industry role with similar labels hence multiple label prediction problems.

Srikant & Aggarwal (2014) presented a model to map graduate's skills to programmer competences. Their approach involved mapping graduate's program for skills based on two layered steps: 1) program logic that was evaluated for best programming practices; 2) complexity of the program that was evaluated for execution time. An average score of the two steps was mapped to five competence levels defined by domain experts. They used ridge, SVM, and Random Forest to generate their model based on regression method. Their underlying ML structure was flat with an average performance of 60% for SVM model. Although their ML attributes were relevant to problem solving skills, they were just too specific for programmers only and hence domain dependent.

Table 2.2 provides a summary of analysis for some of the most important properties of models in related literature where broad range of attributes and flat ML classification structure were dominant. Our dilemma was whether ML methods used in the past were adequate, and whether attributes and ML classification structure used were relevant to industry roles.

Table 2.2: Summary analysis of related ML skills mapping models

Author/ work	**Year**	**Method**	**Type of Attributes**	**Number of attributes**	**Classification Structure**	**Performance**	**Target class**	**Target class construct**	**Nature of attributes to problem solving**
Chien & Chen	2008	Classification	Demographic profile	7	Flat	80%	engineers	broad	Non relevant
Jantawan & Tsai	2013	Classification	Demographic profile	21	Flat	98%	employee	broad	Non relevant
Korte *et al.*	2013	Classification	Qualifications	9	Flat	Not given	Multiple roles	specific	relevant
Srikart & Aggarwal	2014	Regression	Programming practices	6	Flat	60%	programme r	specific	Domain specific
Shashidhar *et al.*	2015	Classification	English,Logical, Program,Quant	4	Flat	82%	Software engineers	broad	relevant

In summary, models for mapping problem solving skills to industry roles in an attempt to bridge industry academia mismatch gap have been proposed (Chien & Chen, 2008; Korte *et al.*, 2013; Srikant & Aggarwal, 2014; Shashidhar *et al.*, 2015). However, either their target classes are too broad or their attributes are domain specific and not relevant to problem solving skills for effective performance in the industry role. Besides, there is very little research in skills mapping especially towards improving graduates employability using machine learning techniques (McCowan *et al.*, 2016).

We conclude that a mapping model is unknown that has relevant attributes and that takes advantage of both the hierarchical nature of industry roles and the natural mobility of employees in the industry organizational hierarchy. Mapping models using flat machine learning structure that are currently used are either inaccurate or commit more serious errors (Silla & Freitas, 2011; Merschamann & Freitas, 2013).

2.5. Models using Hierarchical Machine Learning Structure

Models using hierarchical machine learning structure for their target classes have not been reported in skills mapping. However, in other domains there is evident effort towards hierarchical machine learning. Barbedo & Lopes (2007) organized musical genre in a hierarchical structure and used musical signals as machine learning attributes to predict the genre of music. They used the conventional top-down tree as the hierarchical ML structure. They applied bottom-up multi-classification approach on the conventional top-down tree where they reported performance result of 61%. Besides, they analyzed the performance of their model along various levels of the structure and reported 87%, 80%, 72%, and 61% at level 1, level 2, level 3, and level 4 respectively. However, their work suffered multiple class labels problem as a result of bottom-up classification method applied to a top-down structured problem.

Clare & King (2003) organized gene functions in a hierarchical structure and used various features of genes as their machine learning attributes to predict the function of a gene. They also used the conventional top-down tree as the hierarchical ML structure. They applied top-down multi-classification approach where they reported performance result of 53.3%. Besides, they analyzed the performance of their model along various levels of the structure and reported 56.4%, 46.3%, 23.1%, and 7.9% at level 1, level 2, level 3, and level 4 respectively.

We conclude that choice and design of an effective classifier model is dependent upon: 1) assumptions made about the classification problem and 2) the problem structure (Kotsiantis, 2007; Silla & Freitas, 2011; Merschamann & Freitas, 203).

2.6. Synopsis of Literature Review

Literature review reveals not much has been done in the area of mapping graduates' skills to industry roles using machine learning techniques. There are several potential areas for improvement ranging from ML attributes, classification method, to ML structure. For example, one of the major problematic issues in multi-classification is a classification approach that contradicts the underlying hierarchical structure of class taxonomy. This formed some of the gaps we focused to address through development of appropriate concepts for ML attributes, structure, and model required to achieve effective mapping of graduates' skills to industry roles. Therefore, theoretical literature analysis was necessary to provide concepts to characterize the mapping problem and ML structure before the state of the art classification methodology that reflects organization of industry roles was proposed.

2.7 Theoretical and Conceptual Frameworks.

A framework is an essential supporting idea around which a research problem is modeled and solved. Two common frameworks around which a research problem is solved are theoretical and conceptual frameworks (Green, 2014). While theoretical framework refers to existing theory or theories used to provide essential explanatory support for the solution to the research problem, conceptual framework is an essential concept developed by the researcher and derived from the existing theory or theories to help provide explanatory support for the solution to the research problem.

Conceptual framework is derived from theoretical framework and is also sometimes known as research framework, research model or research paradigm or conceptual model. Conceptual framework, also conceptual model, specifies variables that will have to be explored in the investigation and identifies relationships between those variables. Therefore, we derived our conceptual model from concepts of existing models for training evaluation that served as the theoretical framework.

The rest of this section attempts to answer systematically the following questions:

1. What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?
2. What is the structural characteristic of concepts that correctly reflects the hierarchy of industry roles required as target classes for machine learning purpose?
3. How do we build using these concepts an appropriate machine learning model for mapping graduates' skills to hierarchically structured industry roles?

2.7.1. Models for Training Evaluation

The purpose of education and training is to improve knowledge, increase skills, and change attitudes of a person in order to improve the fit between the person and job requirements. This can only be achieved through learning and evaluation. Learning is achieved through thinking (cognitive) or doing (psychomotor) or feeling (affective). Hence, the three domains of learning: cognitive learning, psychomotor learning, and affective learning.

The purpose of training evaluation is two folded: to determine whether training objectives were achieved and whether the achievement of these objectives can result into enhanced performance on the job. To achieve this, several evaluation models have been developed to explain the theory behind evaluation. This study is hinged on three theoretical models. These are the Kirkpatrick's (1959) model of training evaluation, the CRESST model of learning evaluation attributed to Baker & Mayer (1999), and the Kraiger's (1993) theory of cognitive learning. These theories have been widely used in describing learning outcomes (O'Neil *et al.*, 2005).

2.7.2. Kirkpatrick's Model of Training Evaluation

Kirkpatrick (1959) produced a training evaluation model that focused on four stages of assessment as shown in Fig.2.1. Stage 1 is *reaction* that assesses learners' satisfaction and how they react to the learning program. Stage 2 is *learning* that assesses the extent to which learners' improved knowledge, increased skills, and changed attitudes. Stage 3 is *transfer* which assesses the extent to which learners' change in behavior and applies what they learn in the job. Stage 4 is *result* and assesses the extent to which the company benefits as a result of training the learner. These stages are hierarchically layered and the difficult of measuring the training performance increases as you move up from stage 1 to stage 4. Many fields have relied on this model or its adaptations for many years (Leake & Parry, 2003).

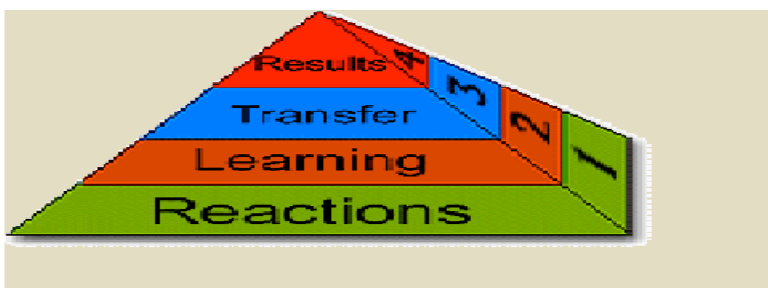


Figure 2.1: Training evaluation stages (adapted from Kirkpatrick, 1959**)**

According to Kirkpatrick (1959), the trainee must learn the content knowledge (stage 2 *learning*) before applying or transferring it to the job (stage 3 *transfer*). Hence, we conclude that learning must begin with acquisition of content knowledge that is relevant to the job and, therefore, evaluation should focus on assessing the relevance of content knowledge acquired.

This current study is basically concerned with stage 2 of the model. However, Kirkpatrick's model does not explain clearly effective measures and variables for assessing learning outcomes at stage 2. In fact, research suggests (Leake & Parry. 2003) that employees transfer very little of what they learn in training (about 10-20%), hence raising curiosity to know whether any learning occurs, and which learning outcomes enhance performance in the job and how can they be measured. Consequently, Leake & Parry (2003) suggest that certain attributes can be used to predict and improve transfer of learning. These are: 1) motivation 2) self-efficacy 3) personality 4) expectations 5) control 6) ability 7) quality of training 8) relevancy of content to the job. That is why, therefore, it became necessary to incorporate the CRESST model to shed more light on the types of learning outcomes that enhance performance in the job.

2.7.3. CRESST Model for Learning

Baker & Mayer (1999) came up with CRESST (Center for Research on Evaluation, Standards, and Student Testing) model of learning evaluation which is a micro-view for stage 2 of Kirkpatrick's model. According to Baker & Mayer, to assess a student in any field it is important to design performance tasks that represent the type of learning intended in terms of broad subject matter topics, item formats, and types of cognitive demands expected to attain success.

In their CRESST model, Baker & Mayer (1999) identified five families of cognitive demands that can be used as a framework for designing teaching, learning, and testing as shown in Fig.2.2. As a result, the CRESST model is composed of 1) *content understanding* 2) *problem solving* 3) *self-regulation* 4) *collaboration/teamwork* 5) *communication skills*. Problem solving is the core outcome of this model. Problem solving is a cognitive process that includes goal-oriented thinking and involves the use of prior or previously acquired knowledge, skills and understanding to meet the demands of an unfamiliar situation (Krulik & Rudnik, 1996; Baker & Mayer, 1999; Orhun, 2003; Wirth & Klieme, 2011).

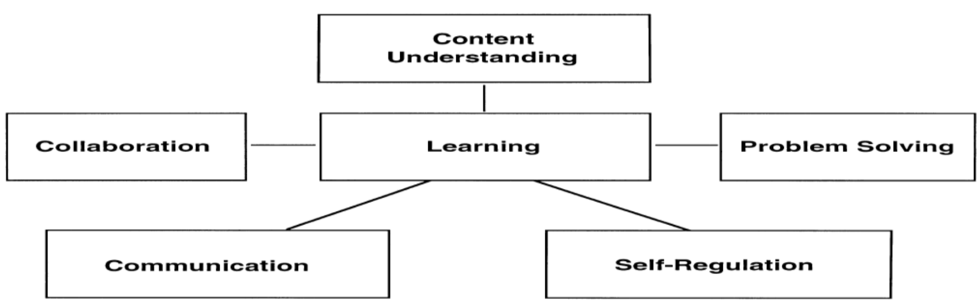


Figure 2.2: CRESST model for learning (adapted from Baker & Mayer, 1999)

Consequently, problem solving provides the learner with the capacity to apply or transfer content knowledge learned to the job (new situation). According to Anderson *et al.* (2001) as quoted by

O'Neil *et al.* (2005), problem solving transfer involves applying a specific set of cognitive processes to a specific set of knowledge types. Baker & Mayer (1999) further observes that problem solving is a family that is a superset of other families, and consists of: content understanding, problem solving strategies, and self-regulation. Self-regulation comprises of motivation and metacognition, while problem solving strategies comprises of domain dependent and domain independent aspects.

Domain dependent (specific) aspect of problem solving strategies involves the specific content knowledge, specific procedural knowledge in the domain, domain specific cognitive strategies, and domain specific discourse (Baker & Mayer, 1999). On the other hand, domain independent (general) aspect of problem solving is static and is very strongly related to intelligence (reasoning) (Wirth & Klieme, 2011). Motivation comprises of two components: effort and self-efficacy.

Furthermore, Baker & Mayer (1999) observe that each family consists of a set of cognitive tasks which can be used as a skeleton for the design of instruction and testing, and this forms a skeletal structure. Each cognitive task in the skeletal structure will have a set of core cognitive demands. The skeletal structures in each family will be instantiated in content domains so as to form structurally similar models that can be applied across domains, like science, mathematics, or social sciences.

A number of training evaluations have been conducted using these evaluation models. Common measures that are used to assess these learning outcomes are multiple-question test, essays, and knowledge maps. A survey conducted by O'Neil *et al.* (2005) reveals that assessment of problem solving is the most popular and is assessed using performance measures, followed by content understanding which is assessed using knowledge maps measures. Collaboration is rarely assessed and is not explicitly measured.

Hence, from CRESST model we conclude that the core learning outcome that is fundamental in enhancing performance on the job is problem solving competence. In order to be able to apply content knowledge to the job, problem solving competence is needed. Besides, problem solving competence is multi-dimensional consisting of: - content understanding dimension, intelligence (domain independent) dimension, and technical (domain dependent) dimension, and self-regulation dimension.

Therefore, evaluation of learning should focus in evaluating problem solving competence along the three dimensions. Although the CRESST model is very clear about the outcomes of learning and their various aspects, it is silent about what to test, how to test, and where to test. It does not provide the possible set or range of cognitive tasks or demands needed for each learning outcome and how to assess them. This makes it difficult to evaluate problem solving competence unless we understand the measures for evaluation; hence it was also necessary to look at the cognitive theory of training evaluation to see more about various measures of evaluation.

2.7.4. Cognitive Theory For Training Evaluation

Cognition is a term that describes quantity and type of knowledge and the relationship between knowledge elements. In the context of training evaluation, cognition involves acquisition, organization and application of knowledge (Kraiger *et al.*, 1993). The purpose of training evaluation is two folded: to determine whether training objectives were achieved and whether the achievement of these objectives can result into enhanced performance on the job. To achieve this purpose, Kraiger *et al.* (1993), proposed a classification scheme for the learning outcomes that could be used as a guide for developing a training evaluation model.

Kraiger *et al.* (1993) assumed that learning outcomes are multidimensional and therefore can be evident from changes in cognitive, skill or affective capacities. Consequently, they proposed three learning outcomes: cognitive, skill-based, and affective-based outcomes. They further proposed assessment measures and techniques corresponding to the learning outcomes categories. Cognitive outcomes consists of verbal knowledge measures (measure of amount and accuracy of acquired knowledge), knowledge organization measures (measure of mental models for knowledge retention), and cognitive strategies measures (measure of meta-cognition for skills on self regulation of own's cognition).

While verbal knowledge could be measured directly using speed tests (measures amount of knowledge) and power test (measures accuracy of knowledge), knowledge organization and cognitive strategies require measures that test higher order thinking skills (critical thinking) that promote creation of mental models for knowledge retention.

Skill-based outcomes consist of compilation measures (measure of proceduralization, generalization and discrimination of verbal knowledge during practice), and automacity measures (measure of automatic reaction after a long practice). Both compilation and automaticity require measures that test hands-on performance.

Affective-based outcomes consist of attitude (measure of internal state that influences choice of personal actions) and motivation (measure of internal state that influences behavior). Both attitude and motivation, although require measures that test internal states, they are highly dynamic. Fig.2.3 below shows the learning outcomes as proposed by Kraiger *et al.* (1993).

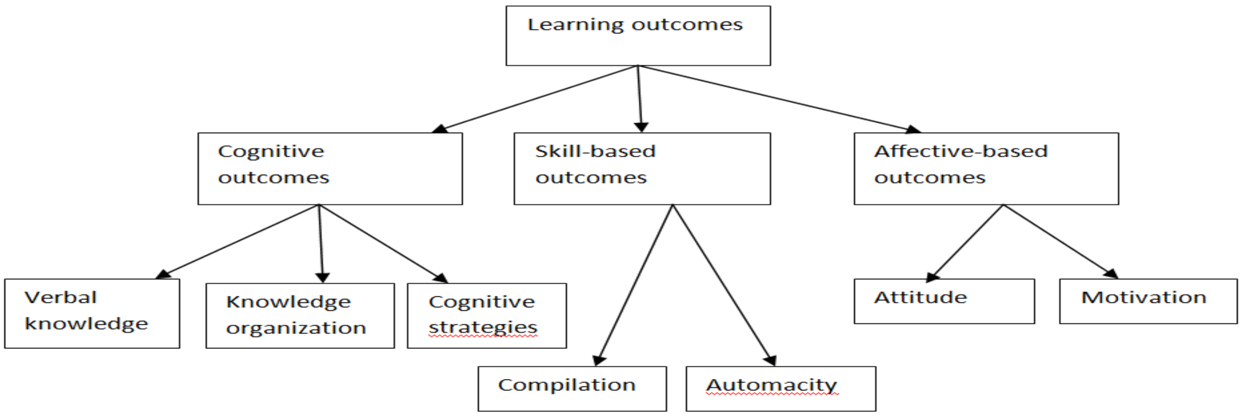


Figure 2.3: Learning outcomes as per Kraiger *et al.* (1993).

Classification of learning outcomes was originally proposed by Bloom *et al.* (1956). According to Bloom, cognitive outcomes beyond recall or recognition of verbal knowledge are legitimate learning outcomes and proposed taxonomy of cognitively based learning outcomes where they came up with six levels of cognitive abilities (intellectual abilities or competence skills) needed during and after learning: Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation. The six levels indicate the increasing level of thinking difficulty starting with knowledge upward to synthesis, to evaluation. According to Mayer (2002), there are a total of 19 types of cognitive processes that can be classified into the six levels or categories of Bloom's taxonomy.

Bloom intended his work to benefit assessment experts who were developing new ways to measure what learners learned. By correlating assessment questions to Bloom's cognitive levels of abilities or skills, test developers can be assured that their questions promote both knowledge retention and critical thinking. However, according to Kraiger *et al.* (1993), Bloom's taxonomy is one-dimensional i.e. is based only on cognitive domain and hence he extended it into three domains. Bloom's taxonomy is well recognized and widely used system in the design and assessment of education components. Fig.2.4 shows the six Bloom's levels of cognitive domain.

Figure 2.4: Cognitive levels (competence skills level) as per Bloom *et al.* (1956).

Therefore, from Kraiger's theory we conclude that there are three categories of learning outcomes hence evaluation measures: cognitive-based, skill-based, and affective-based. Cognitive-based measures include verbal knowledge measures, knowledge organization measures, and cognitive strategies measures. Verbal knowledge is a measure of *relevant* amount and *accuracy* of knowledge acquired during training and can be signaled from content knowledge coverage and grades scored as indicated in achievement tests respectively at the end of the course.

Traditionally, knowledge and skill acquisition during training is assessed through achievement tests (Kraiger *et al.*, 1993) administered at the end of training season. In the context of this study, *relevant* amount of knowledge would be measured relative to the domain body of knowledge and *accuracy* of knowledge would be measured in terms of domain dependent aspects of problem solving. Since performance in technical subjects (or skill related subjects) that provide technical skill required for the industry role is a measure and predictor for problem solving skills (Kraiger *et al.*, 1993), this could be used as a signal for *accuracy* of knowledge and skills acquired during training.

Knowledge organization and cognitive strategies are measures of knowledge retention for *durability* or transfer required for critical thinking, which are promoted through Bloom's competence skills as covered in test items. Achievement tests use items that require learners to apply a particular cognitive process to a particular type of knowledge (Mayer, 2002) or that test higher order thinking skills to assess student ability to apply acquired knowledge and skills in situations inside and outside school (Kellaghan & Greaney, 2003). There are 19 types of cognitive processes that can be classified into six major categories: knowledge, comprehension, application, analysis, synthesis and evaluation (Bloom *et al.*, 1956; Mayer, 2002).

Verbal knowledge is necessary for higher order skills development and task performance at early stages of training, but in advanced training stages tasks behaviors become internalized and performance levels for tasks will be influenced as much as psychomotor differences and general intellectual abilities (Kraiger *et al.*, 1993). Individual's academic capability is an index directly relevant to training and employment opportunities, and performance Grade Point Average (GPA) in high school and university undergraduate level is a good predictor of student *capacity* for the industry role (Richardson & Abraham, 2012).

Skill-based outcomes can only be measured after the graduate is assigned the industry role because they require hands-on performance measures. Finally, affective-based outcomes (*attitude and motivation*) are highly dynamic and influence graduate choice of industry role, and therefore were used as confounded variables in the proposed model.

2.7.5. Discussion Summary of Training Evaluation Models

Table 2.3 below captures a summary of how the three models contributed to the derivation of the proposed research variables and their proposed measures of evaluation.

Table 2.3: Learning outcomes and their measures (Kirkpatrick, 1956; Baker & Mayer, 1999; Kraiger *et al.*, 1993)

Leaning outcomes	**Theoretical Models for Analysis Learning**			**Proposed variables for evaluation of learning transfer**	**Source of student assessment information**	**Evaluation framework**
	Kirkpatrick's model	**CRESST model (learning outcome)**	**Kraiger's model (measures)**			
Content knowledge	Relevance to job	Prior knowledge		Possession of Relevant Content knowledge	Domain exam questions (qualitative)	Body of knowledge
Problem solving competence		Content understanding	Cognitive strategies (processes)	Understanding of Content (Cognitive skills)	Domain exam questions (qualitative)	Cognitive skills framework
		(Domain independent) Intelligence	Knowledge organization	Intellectual ability to learn (Academic capacity)	Student GPA (qualitative)	High school and undergraduate GPA
		(domain dependent) Technical	Verbal knowledge	Ability to perform with precision and speed (Technical skills)	Domain subjects performance (quantitative)	Performance grades in domain technical skills subjects

Content knowledge is one that is taught in class by teachers. Teachers are known to align their teaching to the demands of examinations and studies have shown considerable evidence that a change in the content area examined results into a shift in the content to which students are exposed (Madaus & Kellaghan, 1992; Eisemon, 1990) as cited by Kellaghan & Greaney (2003). Since graduate's skills are influenced by individual's capability and subject content coverage (Kraiger *et al.*, 1993), analysis of the examination test items and student performance can provide insights into the nature and level of knowledge and skills learned at the end of the course (Kellaghan & Greaney, 2003).

Student performance GPA, in high school and university undergraduate, is a good predictor of individual's academic capability and is an index directly relevant to training and employment opportunities (Geiser & Sentelices, 2007; Richardson & Abraham, 2012). Fig.2.5 below summarizes how the proposed mapping model variables are related to and derived from the Kraiger's conceptual model. The yellow balloons represent one of the four independent factors in the model while the green balloon represents the confounding factors.

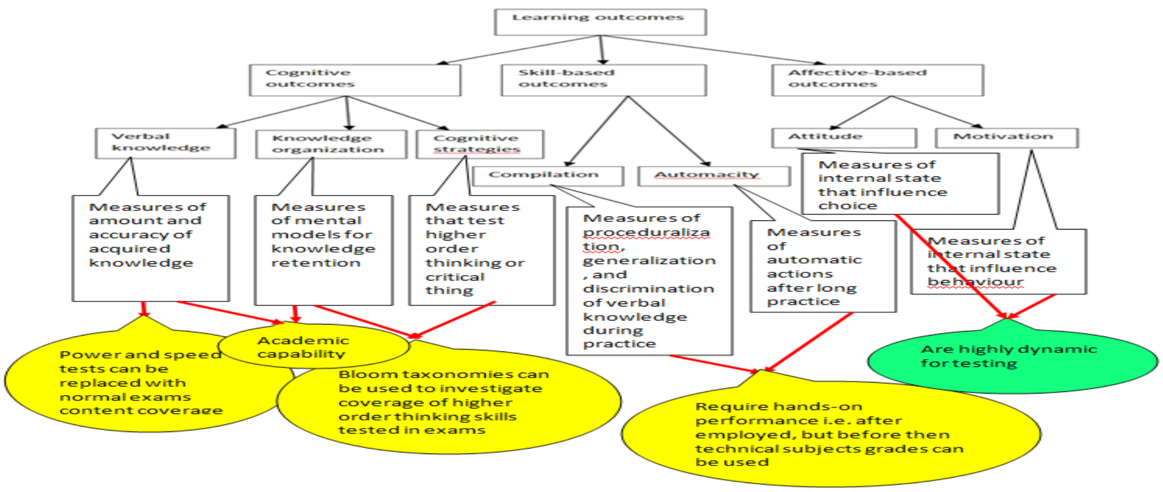


Figure 2.5: Deriving variables of the proposed mapping model from Kraiger's conceptual model (Kraiger *et al.*, 1993).

2.7.6 Conceptual Framework for the Proposed Mapping Model

The study's conceptual model was based on three theoretical models: Kirkpatrick's model, CRESST model, and Kraiger's cognitive theory for training evaluation. The study hypothesized that the problem solving competence requirement of an industry role could be determined by five cognitive factors: Content knowledge, technical skills, cognitive skills, academic capacity of individual's ability and Attitude-Motivational factors. Therefore, content knowledge, cognitive skills, technical skills, and academic capacity are independent factors or variables and industry role is the dependent variable as shown in the proposed conceptual model. Fig.2.6 shows the proposed conceptual mapping model.

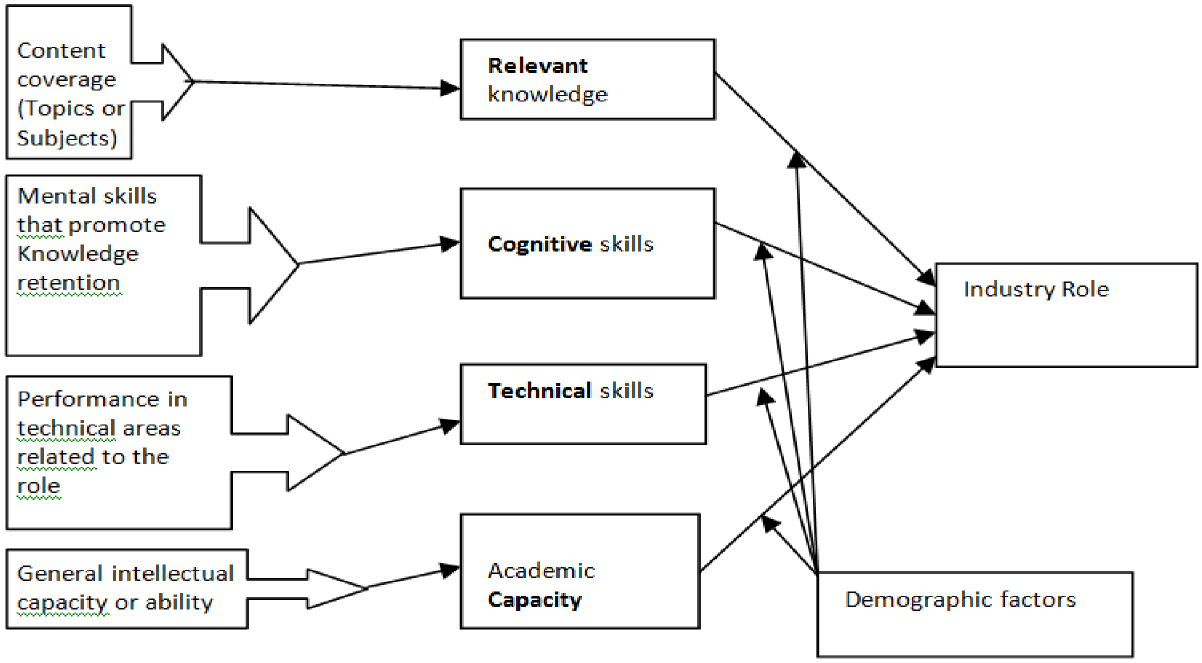


Figure 2.6: The conceptual model for proposed mapping model as adapted from training evaluation model (Kirkpatrick, 1956), learning evaluation model (Baker & Mayer, 1999), training evaluation model (Kraiger *et al*., 1993).

Choice of industry roles may also be affected by attitude and motivation associated with demographic factors. Demographic factors that have been known to influence motivation and attitude include environmental factors, physiological factors, and psychological factors. Environmental factors relate to location and specialization of the job which may be closely correlated to university of study and type of bachelor's degree for an inexperienced graduate. Physiological factors are related to the physical systems of the person which may be correlated to age. Psychological factors are related to internal motives that make a person to seek for more success or achievement and this may be correlated to grading system used to reward academic performance.

The above categories of demographic factors may influence not only the way the graduate is attracted to an industry role but also the way the employer selects a graduate for an industry role and, therefore, they were captured in the conceptual model generally as confounding factors.

2.7.7 Automatic Skills Mapping using the Proposed Mapping Model

The conceptual model of the proposed mapping model was used to describe each industry role concept as a function of the independent factors defined in the conceptual model. Basically, the concept of industry roles was linked to the concept of occupation which is a collection of jobs, sufficiently similar in work performed and grouped under a common label known as occupational title (NOC, 2011). Some occupational titles are broad while others are specializations within occupational area.

Traditionally, four types of structures are used to organize industry roles in any organization, namely functional, geographical, product, and matrix (Malone, 2011). Fig.2.7a presents the four types of structures used to organize industry roles. Therefore, occupational titles, and hence industry role concepts, are predefined, are structured hierarchically, are associated with a certain skill level (as explained in section 2.3.2) and occupational mobility of employees is vertical and upward. Computationally, skills mapping problem can be viewed as a pattern recognition problem and modeled as a ML task for mapping skills to predefined roles in the hierarchical structure using a suitable traditional design methodology for problem solving, such as bottom-up or top-down.

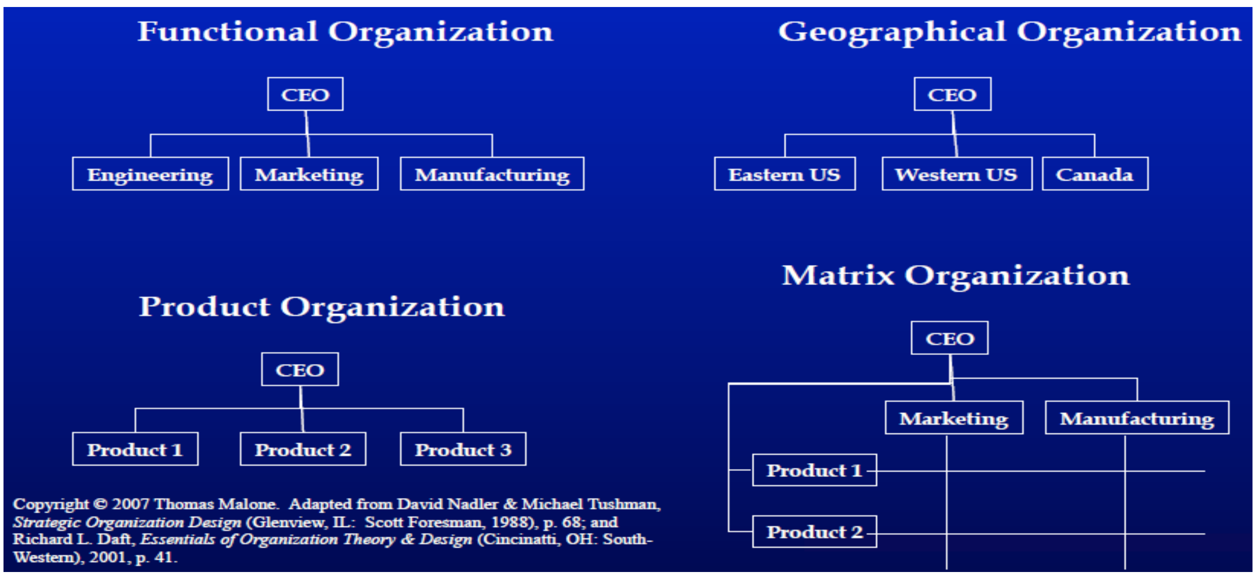


Figure 2.7a: Organization Structures for Industry Roles (Malone, 2011)

2.7.7.1 Top-Down Versus Bottom-Up Approaches

1) Top-down Approach

In top-down approach, a problem is split repeatedly into smaller units and each unit is further split over and over again until the resulting smaller problem unit is manageable. The aim is to solve the problem progressively from generality to specifics where the underlying problem is described hierarchically using a tree structure that is asymmetric and transitive (Silla & Freitas, 2011). As a problem solving approach, top-down involves solving the problem from a known starting state (defined objective/requirements) to an unknown end state (solution or technical basis that satisfies the objective/requirements) where repeated decomposition is a divide and conquer strategy towards the unknown state (technical basis). The objective or requirements must be global and delegatable to individual lower level components (Crespi *et al.*, 2005) where the aim is to satisfy these known requirements through unknown solutions.

In the classification problem, top-down method's objective is to first predict the most generic class (generic level) then it relies on the predicted class to select the next level class where the only valid candidate classes are children of the previous level predicted class, and this is repeated in each level until the most specific class is predicted. Fig.2.7 presents two common types of hierarchical machine learning taxonomic structures/trees that model and support top-down approach as tree (top-down) and directed acyclic graph (DAG) as presented by Silla & Freitas (2011).

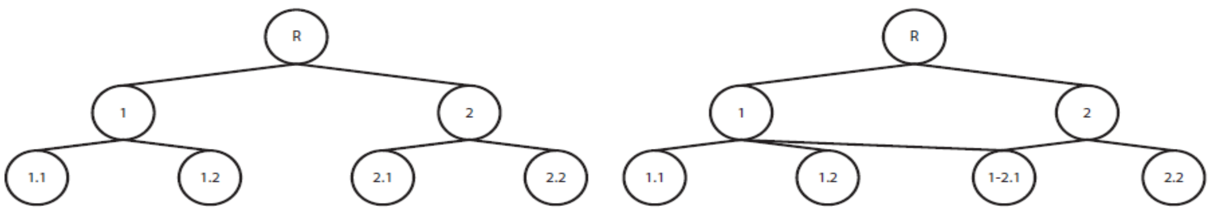


Figure 2.7b: Tree structure (left-side diagram) and DAG structure (right-side diagram)

According to Silla & Freitas (2011), the underlying structure of most hierarchical classification problems are based on tree or DAG structures whose "IS-A" relationship is asymmetric, anti-reflexive, transitive, and has the following properties:

1) The only one greatest element R is the root of the tree.
2) For every class c_i ; $c_j \epsilon C$; *if* c_i *is related to* c_j *then* c_j *is not related to* c_i.
3) For every class $c_i \epsilon C$; c_i *is not related to* c_i.
4) For every class c_i; c_j ; $c_k \epsilon C$; c_i *is related to* c_j *and* c_j *is related to* c_k *imply* c_i *is related to* c_k.

Currently, classification problems with the above structures have been solved successfully using top-down approaches. However, not all problems have such kind of structures and, therefore, top-down approach may not be suitable for them.

2) Bottom-up Approach

In bottom-up approach, the problem solution is derived in the reverse order of top-down approach (Barbedo & Lopes, 2007). The main idea is to analyze large volumes of individual pieces of information so as to find relationships and patterns that can help to generalize into a meaningful solution. Ideally, the aim is to solve the problem progressively and incrementally from the most specific and basic aspects to the most complex and general aspect. This approach begins with lower level local processing and works towards higher level global processing, where lower level specific/basic items are analyzed to provide information that helps to generalize into meaningful and complex higher level items (Amir, 2014; Maloof, M.A, 1999).

As a problem solving approach, bottom-up involves solving the problem from a known end state (solution/technical reality) to an unknown goal state (objective/requirements) where known solutions are agglomerated in a more flexible way to satisfy unknown (or variety of realistic) requirements (Crespi *et al.*, 2005). In the classification problem, bottom-up method's objective is to first select the technical basis (lower level) for describing/modeling classifier objects (higher level) that whose prediction results are agglomerated in a more flexible way to satisfy a number of unknown or realistic user requirements represented by class concepts.

Both top-down and bottom-up can be viewed as complementary approaches for mapping predetermined requirements (top) to available possible solutions (bottom) which can be approached from either side. While top-down begins with requirements then followed by stepwise refinement down to technical basis for implementation, bottom-up begins with technical basis of implementation and attempts to reach up the requirements by constructing higher level services and components on already existing implementations (Crespi *et al.*, 2005). Both approaches are valid and constitute different ways of thinking which have been used widely to develop computing models such as operating systems, computer games, banking systems among many others.

However, it is important to note that the underlying structure of some problems, such as skills mapping, may not fit well to top-down approach. Besides, applying a bottom-up method on the traditional taxonomic tree structures, as defined by Silla & Freitas (2011), leads to either class inconsistency or multiple label classification problems as revealed by Barbedo & Lopes (2007). As a result, for a bottom-up solution to work effectively, a suitable taxonomic structure must be defined that promotes or facilitates bottom-up processing and results to a single class label prediction. Consequently, part of the contribution of this study was to propose not only a machine learning architecture for a skills mapping model but also a taxonomic structure that is bottom-up friendly.

2.7.7.2 Proposed Taxonomy

Hierarchical classification is a special type of structured classification problem. Structured classification is a problem where there is some structure (hierarchical or not) among the classes and the output of the classification algorithm is defined over a class taxonomy. Wu *et al.* (2005) defined a class taxonomy as a tree structured regular concept hierarchy defined over a partially order set (C, R), where C is a finite set that lists all classes in the application domain and the relation, R, represents the “IS-A” relationship. According to Silla & Freitas (2011), most hierarchical classification problems are based on: 1) trees or DAG structure whose “IS-A” relationship is asymmetric, anti-reflexive, and transitive, 2) flat or multi-class classifiers that are multi-label.

However, underlying structure of skills mapping problem may not fit well to top-down approach. This is because occupational titles, and hence industry roles, are structured hierarchically according to the organizational structure. This is evident from the hierarchical nature of most organizational structures including functional, product, geographical, and matrix organization structures (Malone, 2011). Each occupation is associated with a certain skill level which varies increasingly upward in the hierarchy, from lower skilled occupations to higher skilled occupations.

Further, occupational mobility of employees is vertical and upward i.e. employees start with occupational roles at entry level positions and progress to increasingly higher skilled occupational roles. Occupational roles at higher levels of the hierarchy are characterized by higher levels of responsibility, accountability, and subject matter expertise gained through formal education or extensive experience in lower skilled occupational roles (NOC, 2011). Occupational mobility may be through promotion or appointment. Unlike promotion where an existing employee progresses upward the occupational ladders based on observed job performance and experience, in appointment an employee (new or existing) does not necessarily start at the lower levels occupational roles but can be appointed to any occupational role at any level based on performance predicted from their academic qualifications.

Therefore, skills mapping involves classifying a set of skills into one of predefined industry occupational roles in the hierarchy. Since the natural occupational mobility of employees is upward, then the classification strategy that fits well with this phenomenon is bottom-up approach. However, the above two machine learning structures in Fig. 2.7b are top down oriented and may not be fit to not only represent the skills mapping problem but also work well with bottom-up approach.

Analysis of these four organization structures against the two ML structures (trees) available for hierarchical ML revealed no tree could be used to describe all four organization structures at once. Ideally, top-down tree is suited well for only functional, geographic, and product structures while DAG tree is suited well for only matrix structure. Therefore, way forward was to create a ML structure suitable to represent hierarchy of industry roles uniformly across the four possible organization structures and in a way that also obeys the natural mobility of employees along the organization structure, which is practically from bottom to top.

Literature (CWA16458, 2012) provided a clue that all industry roles are characterized by three dimensions, namely main competence, specific competence, and proficiency. In the present study, a tree was created that represented the three dimensions graphically and was proposed as the machine learning structure suitable to achieve the research goal. Fig. 2.8 shows the proposed bottom-up friendly taxonomic structure (BFTS) that represents the hypothetical structural organization of classes as per the structured classification problem and classification assumptions in this method.

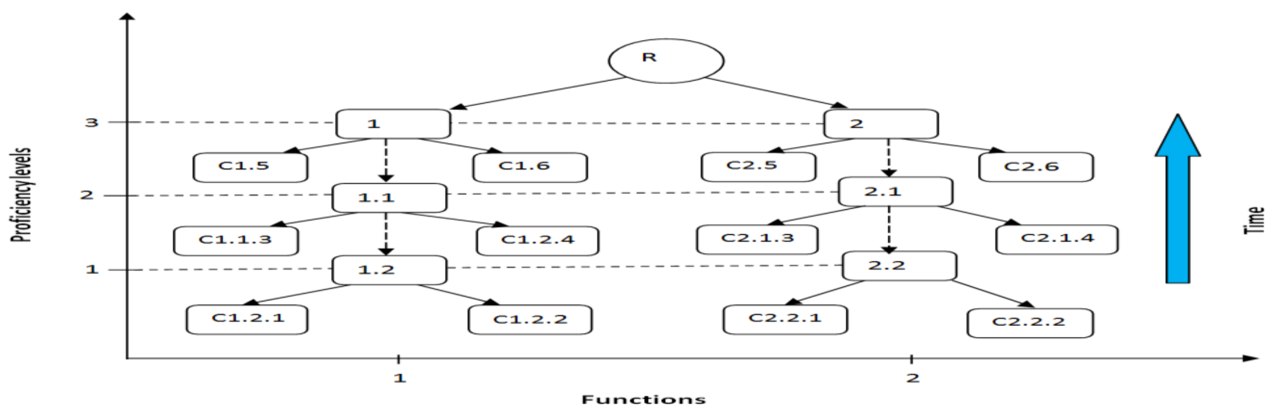


Figure 2.8: Bottom-up friendly taxonomic structure.

Figure 2.8 illustrates hierarchical structure with two branches (may be more), each branch with three levels, a total of twelve leaf node classes (C1.5, C1.6, C1.1.3, C1.2.4, C1.2.1, C1.2.2, C2.5, C2.6, C2.1.3, C2.1.4, C2.2.1, and C2.2.2), and a total of six parent nodes (1, 1.1, 1.2, 2, 2.1, and 2.2), and root node (R). Leaf nodes represent specific competences, non-leaf nodes represent main competences of individual roles while the upward arrow indicates the direction of employees' occupational mobility with time based on proficiency.

However, although the proposed taxonomic structure "IS-A" relationship is asymmetric and ant-reflexive as in Sillas & Freitas (2011) definition of "IS-A" relationship, it departs away from this definition by being anti-transitive with the following properties:

1) The only one greatest element R is the root of the tree.

2) For every class c_i *;* $c_j \epsilon C$*; if* c_i *is related to* c_j *then* c_j *is not related to* c_i*.*

3) For every class $c_i \epsilon C$*;* c_i *is not related to* c_i*.*

4) For every class c_i*;* c_j *;* $c_k \epsilon C$*;* c_i *is related to* c_j *and* c_j *is related to* c_k *does not imply* c_i *is related to* c_k*.*

In the context of skills mapping, the above proposed taxonomic structure represents the hypothetical structural organization of occupational industry roles' problem, and reflects not only the natural mobility of employees upward the occupational ladders but also promises effective bottom-up mapping of graduate skills to industry roles that does not result to multiple label prediction problem. As per the assumptions of the current skills mapping problem, each branch represents an occupational function which refers to a skills category; each level represents proficiency which refers to a skills level, non-leaf node represents main competence which refers to a main skill type, while each leaf node represents specialty of industry role which refers to a specific skill type.

However, while each specialty is a member of a proficiency category, relationship between proficiency categories is one of peer to peer where one category follows the other. As a result, these concepts have been applied in subsequent discussion of the proposed machine learning architecture. The main difference between the proposed taxonomic structure and the traditional tree structure is eminent at the levels/non-leaf nodes where the former adopts peer-to-peer and the later adopts parent-child relationships.

While in the traditional structure lower level parents are decompositions of higher level parents, this is not the case in the proposed structure as each level is a category that indicates superiority of skills proficiency. However, to be able to explore the proposed taxonomic structure from bottom to top as it is natural with employee mobility in the organizational hierarchy, there was need of a special type of architecture for the skills mapping model.

2.7.7.3 Proposed Machine Learning Architecture for Skills Mapping Model

Modeling skills mapping problem computationally involves abstracting it from the problem space and defining the computational theory and tools needed to solve it (Vernon, 2009). Although attempts have been made by posing this problem as a multi-class classification problem and solving using machine learning theory (Jantawan & Tsai, 2013; Chien & Chen, 2008), existing studies have approached this problem using top-down instead of bottom-up method, hence are not sufficient and their results may not be reliable.

Currently, bottom-up method has not been applied in skills mapping to industry roles, and therefore part of the contribution of this study was to propose bottom-up ML architecture needed to generate the automatic skills mapping model that promises reliable results. Fig. 2.9a illustrates a machine learning architecture of a model for skills mapping to industry roles by exploring the proposed taxonomic structure in Fig. 2.8 that represents the problem. The mapping model consists of a number of objects that are hierarchically arranged to progressively group industry role constructs before selecting the best.

At each level, different kind of objects are triggered to generate specific type of information that is jointly used at higher levels for further processing and this continues up to the highest level where the most promising class is predicted. The model objects at lower levels gather local information about the demographic characteristic of the problem structure (height, width, siblings, evaluation objects) which they then pass to higher levels whose objects collect further local information about the potential function, proficiency and specialty before this information is subjected to higher level global processing to reveal or predict the industry role class label. The industry role's label is described in terms of function, specialty, and proficiency.

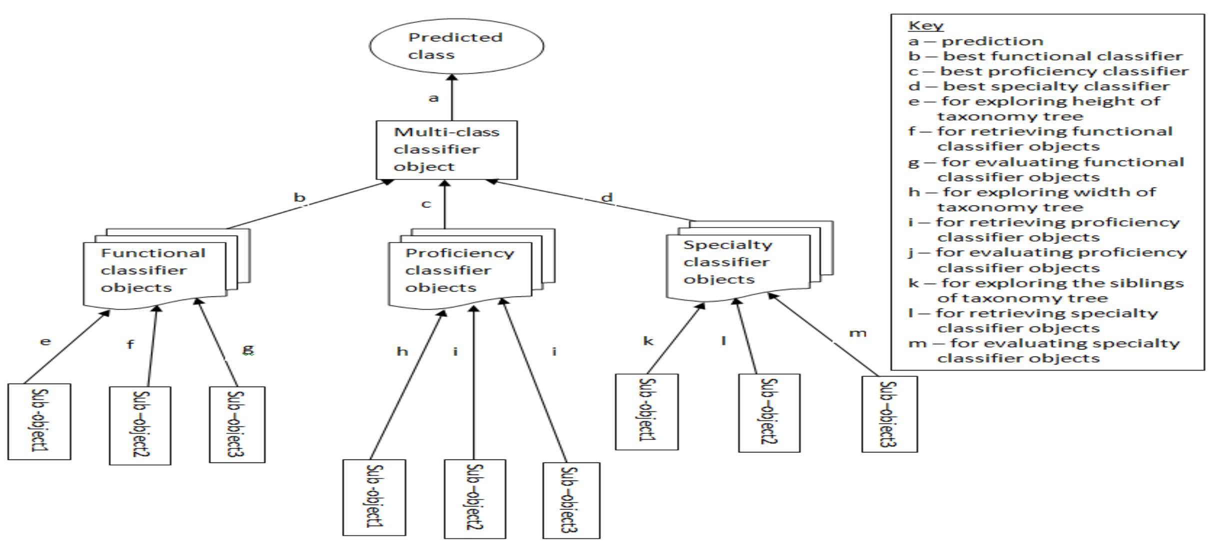


Figure 2.9a: Machine Learning Architecture for the Model

Based on the problem formulation, the appropriate classification methodology was: 1) Multiclass (i.e. many classes), 2) Hierarchical (i.e. several levels), 3) Bottom-up (i.e. vertically upward the levels), 4) Supervised (i.e. trained with predefined classes). The multi-class classifier comprised a collection of binary classifiers or objects organized methodically into layers that were activated from bottom to top.

2.7.7.4 Basic Architecture of Model's Classifier Objects

Machine learning is one of the commonly representatives of bottom-up analysis where various types of data are analyzed to reveal relationships and patterns (Wirsch, 2014). As a result, the underlying structure of each machine learning object is based on bottom-up method as per Fig. 2.6 of proposed conceptual framework. Fig. 2.9b shows the basic architecture of each classifier objects indicated in Fig. 2.9a.

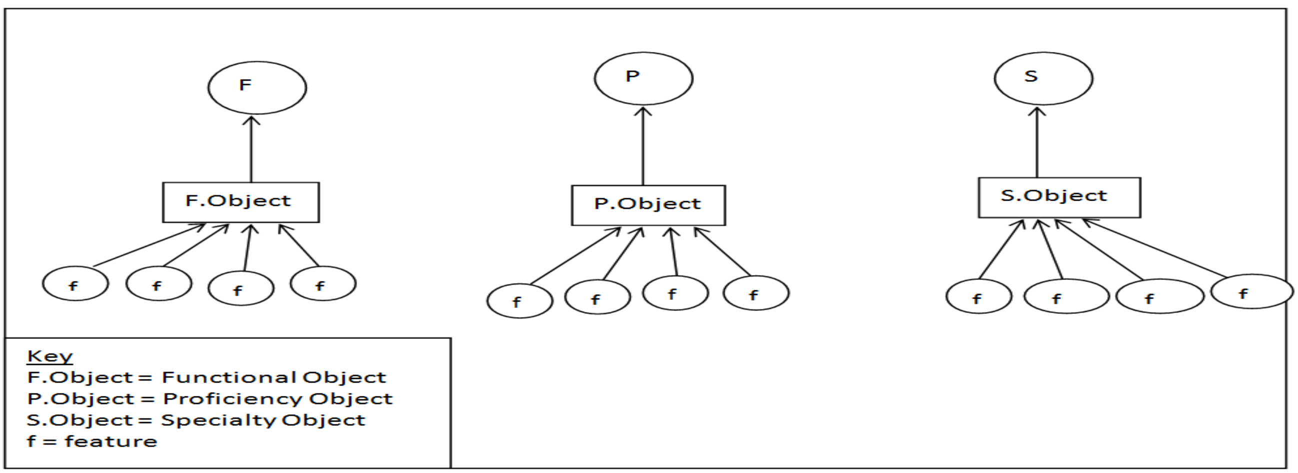


Figure 2.9b: Machine Learning Architecture for the Model's Objects

2.7.7.5 Choice of ML Algorithm for the Model's Classifier Objects

Key questions when choosing machine learning algorithm is not about whether or not a learning algorithm is superior to the others, but how significantly it outperforms others on a given application problem under certain known conditions (Kononenko, 2001). Perhaps the best and simplest approach could be to estimate the accuracy needed on the problem and choose the one that appeared to be most accurate. However, accuracy alone is not sufficient (Kononenko, 2001). The trend in the improvement of classifier performance is the concept of combining two or more learning algorithms.

This is currently popular among researchers with the ultimate goal of generating more certain, precise and accurate results.

Criteria for selecting learning algorithm are characterized by: 1) accuracy 2) speed of learning 3) number of parameters 4) transparency (ease to understand a method) 5) results interpretability 6) incremental learning (Stefanowski, 2010). Some of the best known algorithms include:1) decision trees(DT), 2) rule-learners(RL), 3) Neural Networks(NN), 4) K-Nearest Neighbor(KNN), 5) Support Vector Machine(SVM), 6) Naïve Bayes(NB) 7) Logistic Regression (LR)) and can be divided into three groups based on assessment against the six criteria: group A (DT,RL), group B (LR,NB), and group C (NN, KNN ,SVM).

While group A members have similar operational profile and strongly conform to quick learning, fewer parameter handling, good transparency and high interpretability, group C members (although have similar operational profile) have low learning speeds, many parameters therefore poor parameter handling, poor transparency, and poor interpretability. However, group C is superior in high accuracy and good incremental learning while group A is very poor in those aspects. To moderate between these two extreme groups there is group B which conveniently harmonizes the two groups by taking half of the good features of either side (of group A and B) while taking the average of each feature of the remaining half.

Although group B members have joint added advantage of dealing with over fitting dangers, members here complement each other on the speed of classification, tolerance of noise and missing values. Therefore group B and C stood out as better candidates to use in the current research for constructing the classifier. Table 2.4 outlines a summary of criteria that could guide selection of the machine learning technique in each category.

Based on the analysis in Table 2.4 the study proposed to train the model using Naïve Bayes and SVM learning algorithms. The selection was guided first by good incremental learning (ability to refine its learned rules), then ability to deal with missing data and noise in data, and finally ability to accurately perform. In skills mapping, skills requirements for industry roles gradually migrate as environmental factors, such as technology, change. This demands the model to accumulate these changes and refine its learning rules without necessarily requiring retraining of the model. Therefore, the model needs to have a very good incremental learning property. As a result, group A ML algorithms were technically removed from any further consideration.

Besides, skills mapping model should be able to work with data collected in the field which is likely to have missing data or non-relevant values (also known as outliers) without necessarily requiring replacing them with meaningful values. However, K-Nearest Neighbors and Neural Networks algorithms are very poor at tolerating missing values and noise data while SVM and Naïve Bayes handle this easily by ignoring them (Kononenko, 2001; Kotsiantis, 2007). As a result, this improves the classification speed of the model and therefore K-Nearest Neighbors and Neural Networks algorithms were dropped from further consideration.

Finally, to ensure the right people are placed in the right job, the model must have very high performance accuracy. SVM algorithm is highly associated with high performance accuracy which should be an important property of the model. In contrast, naïve Bayes and Logistic Regression have a moderate accuracy. However, naïve Bayes has been used widely as a benchmark algorithm in many other studies (Kononenko, 2001) as opposed to Logistic Regression. Therefore, to ensure our work is able to compare with results achieved in other related work for validity purposes, Logistic Regression was dropped in favor of naïve Bayes algorithm.

Table 2.4: Features of main categories of machine learning algorithms (Kotsiantis, 2007)

	TYPES OF MACHINE LEARNING ALGORITHMS		
	GROUP A	***GROUP B***	***GROUP C***
TYPE OF FEATURES	***Rule-Learners(RL), Decision Trees(DT)***	***Naïve Bayes(NB), Logistic Regression (LR)***	***K-Nearest Neighbour(KNN) Neural Networks(NN), Support Vector Machines(SVM)***
GOOD	a) Quick learning b) Fewer/Good parameter handling c) Good transparency d) High interpretability	a) Quick learning b) Fewer/Good parameter handling c) Good transparency d) High interpretability e) Good incremental learning	a) High accuracy b) Good incremental learning
BAD	a) Low accuracy b) Poor incremental learning	**a)** Low accuracy	b) Slow learning c) Many parameter handling d) Bad transparency e) Low interpretability

2.7.7 Synopsis of Theoretical Concepts Development

Lessons learnt from this section related to our research questions 1 & 2 included:

1) Concepts appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles must be based on strong cognitive theoretical frameworks. Fig. 2.9c and Table 2.5 have summarized how these proposed concepts were derived and operationalised

Table 2.5: Operationalization of Conceptual Framework concepts

Concept	Indicator	Variable	Measure-ment
1.Relevant content knowledge	Domain Body of Knowledge	Topic areas of Body of knowledge	Scale
2.Cognitive skills	Cognitive skills areas	Skills areas of Bloom's Taxonomy	Scale
3.Technical skills	Domain technical subjects	Domain technical subjects	Scale
4.Academic capacity	School GPA	Average GPA high school and university	Scale
5.Industry role	Occupational industry roles	Occupational industry roles	Nominal
6. Demographic factors as Confounding factors	Environmental factors:	University of study	Nominal
		Bachelor's Degree type	Nominal
		Location of 'O' level study	Nominal
	Physiological factors:	age	Nominal
		gender	Nominal
	Psychological factors:	'O' level grading system	Nominal
		'O' level results	Nominal
		Degree grading system	Nominal
		Bachelor's Results	Nominal

respectively.

In the present study proposed concepts were approached from two cognitive dimensions, namely knowledge and skills, and were derived from three cognitive theories. A total of 13 concepts were revealed as follows: 4 as independent and 9 as confounding factors. The validity of these concepts was to be investigated empirically and confirmed.

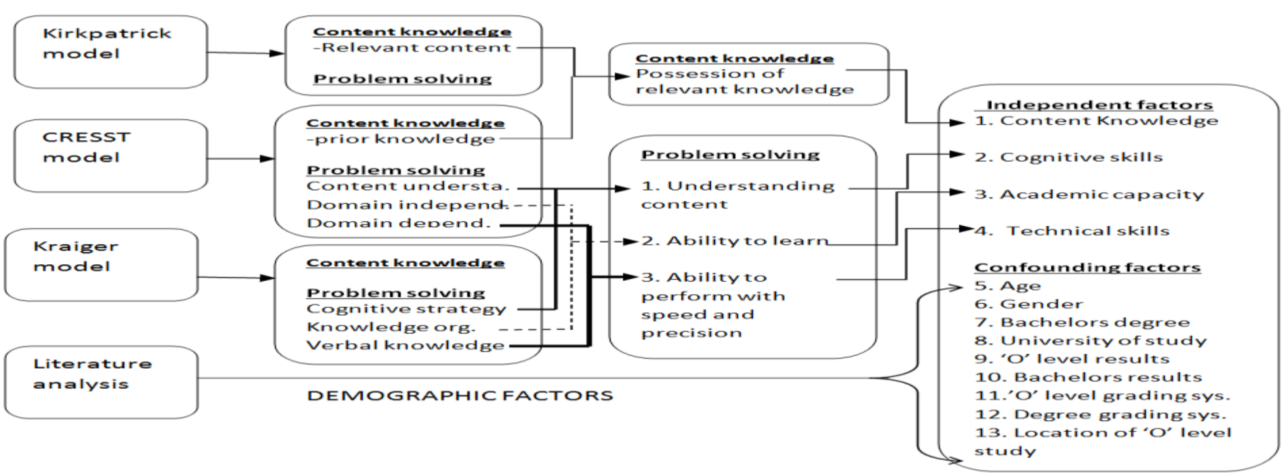


Figure 2.9c: Development of conceptual model

2) While structural characteristics of concepts to be used as target classes for machine learning can be flat or hierarchical, underlying nature of the problem greatly determines this. In the present chapter, indeed literature analysis revealed occupational industry roles were structured hierarchical and their underlying fundamental dimensions were identified as: main competence, specific competence, and proficiency levels. The validity of this hierarchical structure would be investigated empirically and confirmed.
3) Three issues that greatly affect the design and building of classifier models that learn from observations are: 1) Input that consists of a sample of data instances described by a number of attributes which may be of different data types but also parameter values 2) the type of feedback for observational learning which can be of three types: supervised, unsupervised, and reinforcement, and 3) the way the solution is to be represented which depends on feedback output (Lavesson, 2006; Vernon, 2009). While representation of the solution in supervised learning depended on whether the desired output was discrete or continuous, thus it could be represented using a classifier or regression function respectively, in the present case it was a classifier.

2.8 Summary

This chapter has presented detailed review of literature on background information on trends of industry roles requirements, mismatch gap of academia skills and industry roles, related studies, and state of the art technology that guided in providing answers to the research questions. Trends on evolution of industry roles requirements were observed towards jobs requiring more education and cognitive skills. Besides, a mismatch of graduates' skills and industry roles was noted as the main problem between academia and industry, and whose underlying cause was poor mapping of graduates' problem solving skills to industry roles.

However, approaches of related studies towards this mismatch problem indicated models with a broad range of machine learning attributes that either are not relevant to industry roles performance or are not usable across occupational domains. After careful literature review, three learning and evaluation theories provided concepts to support explanation of our solution on this aspect of the problem. Although state of the art technology indicated potential of a better technique to describe the solution using a structure that could represent industry roles, it needed some modification to correctly reflect the hierarchy of industry roles across occupational domains. In summary, the chapter culminated with a proposed conceptual model of the mapping model and a proposed machine learning structure that correctly reflects the structure of industry roles.

CHAPTER 3: RESEARCH METHODOLOGY

3.0 Introduction

This chapter discusses the research philosophy, research strategy including research methods, research design model, and data analysis and presentation. The chapter is organized as follows: Section 3.1 discusses research philosophy, section 3.2 outlines research design, section 3.3 describes the research framework, section 3.4 highlights research methods, section 3.5 describes the methodology for developing the model, and section 3.6 concludes the chapter with a summary.

3.1 Research Philosophy

Research philosophy relates to the development of knowledge and the nature of knowledge. It is a belief system or view about the world that guides the investigation. Philosophical views about the world assumed by the researcher during investigation are described under broad philosophical paradigms such as epistemology, ontology, and axiology.

Epistemology is a term that refers to the theory of knowledge that provides a philosophical support for accepting knowledge discovery and especially how to ensure the adequacy and legitimacy of investigation for the discovered knowledge (KIM, 2009). Hirschheim (1985) reiterates that knowledge is acquired through an inquiry process. At one time, Greeks classified knowledge into two types, i.e. *doxa* (what is believed to be true) and *episteme* (what is known to be true), and method of inquiry involved transformation of *doxa* to *episteme.* Therefore, science as a method of inquiry is considered as the process of transforming what is believed to be true to what is known to be true.

The agreed set of conventions in science is the scientific method, and therefore, for anything to be called scientific knowledge must conform to the scientific method. However, philosophical questions were raised on how to know that something was true. This led to a major difference in opinion over the nature of truth and how to arrive at it through the scientific investigation (Easterbrook *et al.*, 2007).

Consequently, ontology is a term that refers to the nature of reality in terms of the way the world objects operate and provides a philosophical support for accepting knowledge discovery (Saunders, *et al*., 2009). Two popular aspects of ontology are objectivism and subjectivism. Objectivism assumes that social entities exist in the external reality outside the mind of a social actor, while subjectivism assumes that social phenomena exist in the mind of the social actors and is created from the perceptions and consequence actions of the social actors.

Two well known philosophies based on the above two paradigms are positivism and interpretivism. Positivists believe that reality is fixed and can be observed and described from an objective point of view. It advocates the use of highly structured methodology that facilitates replicability, repeatability and generalization. Also, it assumes that the researcher is independent of, and neither affects nor is affected by, the subjects of research. On the other hand, interpretivists believe that reality is too complex and in order to understand it without losing rich insights of its complexity, some kind of subjective interpretation and intervention must be involved (Levin, 1988). Science is based on a strict conception of positivism, an epistemology which posits beliefs and scrutinizes them through empirical testing (Hirschheim, 1985).

In computing, Alavi & Carlson (1992) reviewed 902 Information Systems (IS) research articles and revealed that all empirical studies were positivist in approach. Orlikwoski & Baroudi (1991) as cited by Bolan & Mende (2004) reveal 96.8% of the use of positivists approach in IS based research journals in US. Further, there is reliable evidence that positivism has had successful association with physical and natural sciences in which computer science belongs (Hirschheim, 1985). Hirschheism (1985) summarizes five pillars of positivisms which provide a link between our study and positivism.

1) Unity of scientific method

Scientific method is the only valid and accepted approach for knowledge generation. Our study embraced the conventions of scientific method which include replicability and generalization.

2) Search for casual relationships

We had a desire to find regularities and casual relationship among the elements of study. We attempted to understand regularities and casual relationship between graduate skills and industry roles.

3) Belief in empiricism

We believed that valid data is one that was experienced from the senses and extraordinary experiences, conscious or unconscious arrangement of apparatus, and subjective perceptions were not to be acceptable.

4) Science and its process is value free

Science and its processes are value-free and as a result the undertaking of this study had no relationship or connection with political, ideological or moral beliefs.

5) Foundation of science is based on logic and mathematics

Logic and mathematics provided the basis for quantitative analysis which is an important tool for searching casual relationships. We sought for the casual relationship between graduate skills and industry roles experimentally, where the end product was a law-like generalizations derived through quantitative analyses.

Since the basic idea was to come up with credible findings, we had a clear understanding that graduates were observable objects that were real while industry roles were observable social phenomena. Hence, there was an assurance that the data collected would lead to credible findings. Moreover, knowledge and skills imparted to graduates in academia are realities that exist separate from the graduates who benefit from that reality. This is because the description of the content of knowledge and skills intended for the graduates is well documented in the curricula and the extent to which they are delivered to the graduates is well expressed in the exams papers administered to the students, hence this content is an objective entity.

Although industry roles with similar role names in different organizations may have job descriptions that are different, the role names are just a creation of the social actors who create them. Ideally, the underlying functional requirements are realities that exist separate from the social actors who occupy them and may distinguish industry roles objectively. This is why the researcher believed the relationship between graduates' skills and underlying functional requirements of industry roles were fixed and could be observed and described from an objective point of view.

As a result, a structured methodology that was used in this study promised objectivity to search and reveal any regularities and casual relationship between graduates' skills and industry roles. The structured search and analysis not only enabled the variables that were relevant to the relationship to be explored but also enabled the precise relationship to be described and manipulated in order to observe its behavior. This is to say, the ontological position taken by this study was that of objectivism. And, because positivism is associated with the notion of observable social reality and phenomena that are considered to be real where the assumption made is only real objects can produce credible data, this is clear evidence that this study fitted well to this approach.

To collect this kind of credible data there was need to develop a research design that would enable relevant research hypotheses to be defined and tested using factual data. Factual data was collected with instruments that ensured same questions were asked to the respondents in exactly the same way. Table 3.1 presents a spectrum of research design methods as described by Travis (1999) as cited by

Bolan & Mende (2004) which formed part of the elements of the research strategy to achieve this goal.

Table 3.1 Taxonomy of research methods (Bolan & Mende, 2004)

Scientific (Positivistic)	**<=Spectrum of Research Methods=>**	**Interpretivistic**
Theorem Proof		Subjective / Argumentative
Laboratory Experiments		
Field Experiments		Grounded Theory
Surveys		Action Research
Case Studies		Futures Research, Role/Game Playing
Forecasting		
Simulation		Ethnography / Ethnomethodology

In research, for the results to be credible it is important to know the role of the researcher's values in the research process. This is relevant to both the researcher and research stakeholders. To the researcher, this may raise the issue of individual's honesty in the research process, awareness of value judgments when drawing conclusions and this may help in deciding what is appropriate ethically and answering queries in case they are raised about a decision.

Axiology is a term that is widely used to refer to the study of judgments about values in terms of the role values play in making judgment about a decision and provides a philosophical support for accepting knowledge discovery (Saunders, *et al*., 2009). The role of the researcher's values adopted in this research process related to issues of scientific honesty and ethics to be observed in the selection of data, giving credit where it was due, and avoidance of issues of scientific misconduct. NAS (1995) provided a guideline that was used both to raise awareness of value judgment when drawing scientific conclusion and to caution the researcher on issues of scientific misconduct, ethics and honesty adopted in this study.

3.2 Research Design

Research design is a logic of enquiry or plan or blueprint for an investigation towards obtaining answers to research questions within a caution of controlled variance. While no research design is more superior to all others in all research areas (Benbasat *et al.*, 1987), selection of the design was influenced by the nature of the research topic, goals of the researcher and the paradigmatic assumptions.

With respect to specific goals, paradigmatic assumptions played important role especially when positivism is highly associated with deductive and quantitative approaches. Deductive approach enabled not only the formulation and testing of hypotheses but also identification of the possible results, research method for obtaining the results as well as a validation strategy appropriate for the research results (Shaw, 2002).

Indeed, our design was a mixed methods research design that focused on providing a research purpose for each research question and a plan by which the research purpose was to be achieved. This enabled to reveal the appropriate methods and procedures that were suitable to help collect and analyze data so as to provide research answers. The research approach adopted corresponds roughly to the three major categories of scientific methods consisting of observe, formulate, and evaluate (Glass, 1995).

In computer science, literature reveals the corresponding research approaches are descriptive (also known as characterization), formulative (also known as design), and evaluative respectively (Ramesh *et al.,* 2002; Glass *et al.,* 2004). Descriptive research is concerned with a systematic process of describing systems or situations or groups so as to portray accurately the underlying characteristics, formulative research is concerned with formulating models, processes, or algorithms so as to explore new insights into a phenomenon, while evaluative is concerned with evaluating models or systems or algorithms deductively, or interpretively, or critically so as to test casual relationship between variables. Formulative is the most widely used approach with 79.15% followed by evaluative and descriptive with 10.98% and 9.88 respectively based on a survey by Ramesh *et al.* (2002).

Based on the type of result expected for each research question, appropriate research approaches for obtaining the results as well as to validate the results were determined. In computer science, research results may be of the type of model (qualitative, empirical, analytical, and descriptive), procedure or technique, notation or tool, answer or judgment, or report (Shaw, 2002). Table 3.2 summarizes the characterization of our study's research questions as adapted from Shaw (2002), and proposed research design for each.

Therefore, the research methods applied in the study were determined by the nature and character of research questions, expected results and type of results validation in the study. Shaw (2002) provided a guideline for describing the character of research questions in computer science by outlining the

Table 3.2a: Characterization of research objectives (adapted from Shaw (2002))

Research objective/question	Criteria for characterization of research objectives			Research type
	Question type	Results expected	Method expected to validate results	
1) What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?	Generalization/ characterization (exploratory)	Qualitative model	Evaluation & Analysis	Descriptive (result) & Experimental (validation)
2) What is the structural characteristic of concepts that correctly reflect the hierarchy of industry roles required as target classes for machine learning?	Generalization/ characterization (descriptive)	Qualitative model	Analysis	Descriptive (both)
3) How do we build using these concepts, an appropriate machine learning model for mapping graduates' skills to hierarchically structured occupational industry roles?	Design (formulative)	Empirical model	Evaluation & Analysis	Experimental (both)
4) How do we evaluate performance and validity of the mapping model?	Evaluation	Answer/ judgment	Evaluation & Analysis	Experimental (result) & Descriptive (validation)

approach types of research questions, types of research results, and types of validation and illustrated how this could be used as a guide to choose a research design.

Using Shaw's model, we were able to characterize the research questions and concluded that two questions (1 & 2) were of the type generalization/characterization, one (question 3) was of design type, and the remaining one (question 4) was of the type evaluation. Further, one question (4) result was of the type analysis/judgment and another (question 3) of the type empirical model, while the other two (questions 1 & 2) results were of the type qualitative model. Finally, the type of validation for three questions' (1, 3, & 4) results was of type evaluation/analysis and one (question 2) was only analysis. The proposed research types for each were based on the four research purpose i.e. exploratory, descriptive, diagnostic, and experimentation.

3.2.1 Synopsis of Research Design

1.) To establish concepts appropriate as machine learning attributes for mapping graduates skills to occupational industry roles

There was need for a research design that provides a way to analyze literature and identify concepts appropriate as machine learning attributes for mapping skills to industry roles then experimental evaluation to determine valid attributes. The type of result expected to be a qualitative model, namely conceptual model to be established and validated through literature analysis and experimental evaluation respectively. Based on this requirement, appropriate research designs to collect the data for providing answers were literature review/analysis and experimental designs.

a) Literature Review/Analysis

Based on literature review, we were able to identify candidate attributes that determine ones performance in a particular industry role.

b) Experimental design

From the candidate attributes selected during literature review in a), we were able to conduct feature selection experiments to determine the most relevant attributes.

2.) To establish structural characteristic of concepts that correctly reflect the hierarchy of industry roles required as target classes for machine learning process

There was need for a research design that provides a way to analyze and identify structural characteristic of industry roles then collect data to analyze concepts to be used as target classes for machine learning purpose. The type of result expected to be a qualitative model, namely hierarchical machine learning structure to be established and validated through literature and descriptive analyses respectively. Based on this strategy requirement for research question 2, the most appropriate research designs to collect the data for providing answers were literature review/analysis and descriptive survey designs.

a) Literature Review/Analysis

Based on literature, we were able to identify the most appropriate structure for organizing industry roles for machine learning purpose

b) Descriptive Survey

From the candidate attributes selected during literature review in 1a) and structure identified in 2a), we were able to prepare dataset for machine learning.

3.) To build using these concepts an appropriate machine learning model that maps graduates' skills to hierarchically structured industry roles

There was need for a research design that provides a way to design a mapping model through predictive modeling and experimental analyses to optimize the model then experimentally evaluate to get the best fit model. The type of result expected to be an empirical model, namely machine learning model to be built and validated through experimental analyses and experimental evaluations respectively. Based on this strategy requirement for research question 3, the most appropriate research design to collect the data for providing answers was experimental design.

a) Experimental Design

From the dataset prepared in 2b), we were able to conduct algorithm selection experiments and algorithm optimization experiments to build the machine learning model.

4.) To evaluate the performance and validity of the machine learning mapping model

There was need for a research design that provides a way to build a model's prototype and experimentally evaluate model's prediction performance then analyze performance vis-à-vis other related models. The type of result expected to be a judgment, namely performance result to be validated through comparative literature analysis. Based on this strategy requirement for research question 4, the most appropriate research designs to collect the data for providing answers were literature review/analysis and experimental designs.

a) Literature Review/Analysis

Based on literature we were able to identify appropriate benchmark models and their performance properties.

b) Experimental Design

From the machine learning model developed in 3a) and benchmark models identified in 4a), we were able to conduct performance evaluation experiments and benchmark comparisons to evaluate performance and validity of the model.

3.2.2 Literature Review/Analysis

A model captures relevant features of a phenomenon and these features a derived from theoretical literature, as elaborated by Onweugbuzie *et al.* (2012), which forms the foundation of the study. Therefore, this research method was vital in formulating the research models using explorative variables. This was after enough evidence was gathered that literature analysis is widely used in computer science (Glass *et al.*, 2002, 2004; Ramesh *et al.*, 2004; Vessey, 2001; Zelkowitz & Wallace, 1997) as revealed in a survey by Holz *et al.* (2006).

This design method was applied to three of our research questions, first, second and fourth. First research question required to establish concepts appropriate as machine learning attributes for mapping graduates' skills to industry roles. The design method was used to select appropriate conceptual literature on theories related to learning outcomes and to collect literature data on appropriate concepts as learning outcomes that promoted performance in the job. Qualitative analysis on literature data collected helped to analyze similarities between these theories and relationships towards job performance before constructing the proposed conceptual model (Dodig-Crnkovic, 2002).

Second research question required to establish structural characteristic of concepts that correctly reflected the hierarchy of industry roles required as target classes for machine learning. In this question, literature analysis was used to identify appropriate conceptual literature on frameworks related to describing or organizing industry roles and to collect literature data on appropriate dimensions for describing or organizing industry roles across companies. Qualitative analysis on literature data collected helped to analyze structural elements and their relationships towards describing industry roles across companies before constructing a hierarchical structure for machine learning that correctly describes industry roles.

Fourth research question required to evaluate performance and validity of the model. As a result, literature analysis was used to identify appropriate empirical literature on machine learning models for skills mapping and to eventually collect relevant information pertaining to their performance parameters. Qualitative analysis was used to compare present study's model with other literature models before validating its performance. Table 3.2b outlines a summary of qualitative activities of literature analysis under each target research question.

Table 3.2b: Literature search design

Activity	**Research 1**	**Research 2**	**Research 4**
1. Search	Learning theories	Industry roles' frameworks	Skills mapping models
2. Identify	Relevant theories	Relevant frameworks	Relevant ML models
3. Select	Relevant concepts	Relevant dimensions	Relevant performance parameters
4. Construct	Proposed conceptual model	proposed hierarchical structure	Comparison criteria

3.2.2 Survey

Sufficient evidence was gathered revealing that surveys have been used successfully in computer science (Glass *et al.,* 2002, 2004; Ramesh *et al*., 2004; Vessey, 2001; Alavi & Carlson, 1992; Orlikowski, 1991; Farhoomand, 1999; Hamilton & Ives, 1982; Vogel, 1984) as surveyed by Holz *et al.* (2006). Mathers *et al.* (2007) reveal survey can be cross-sectional or longitudinal and provide detailed description of each.

Then, survey method was applied to research question two where the objective was to collect data that was used to describe structural characteristic of industry roles. The basic idea was that the model must be learned and tested with data on employees profile and applied to recently graduated university students in the academia who were unemployed. Graduate employees who have been holding industry roles were a source of primary data that was collected through survey. Equally, exams past papers of the relevant subjects in the respective domains were a source of primary data for unemployed graduates where the model would be deployed.

Descriptive survey was employed in executing this goal under research question two where industry roles concepts needed to be determined, and the survey was conducted by designing two samples and questionnaire instruments for each. One instrument was designed as a survey questionnaire to collect data from employees' sample so as to establish employees industry roles concepts to be used as target classes for machine learning while the other as an analysis questionnaire to collect data from exams past papers to characterize institution in academia towards industry roles.

The exact details of sample design and instrument validation are provided in sections 3.3.2 and 3.3.3 respectively. Table 3.2c provides a characterization of the survey design that guided the current study's survey. The main justification for using survey in this study included: 1) need to collect data from a wide number of variables, such as 17 variables in this case, 2) need to collect data retrospectively for a phenomenon that has been in existence for while, such as graduates who have been in employment for a while and exam past papers that were done a while ago, only survey could achieve this.

Descriptive analysis was used to characterize industry roles according to the data collected based on attributes of each reference framework identified during literature review. As a result, to provide focus towards the research question under investigation using this research design six aims were set: 1) to establish various industry roles that could be used as target classes for machine learning, 2) to analyze central tendency characteristics of these industry roles as potential target classes, 3) to analyze class boundaries of industry roles as potential target classes, 4) to test significance of class differences of industry roles, 5) to establish academia bias towards these industry roles, 6) to establish underlying structure of industry roles.

Table 3.2c: Characterization of research survey design

Decision Activity A	Employees Population
Type of survey	Cross-sectional
Data collection method	Questionnaire
Target population	Software Engineer Employees
Type of information being sought	Specific job title, skills and knowledge demands for work
Sampling method	Multi-stage: Stage1: software firms Stage2:employees
Sampling technique	Random simple: stage1 Stratified: stage2 Stratum1:lead employee Stratum2:normal employee
Sample size	187

Decision Activity B	Past Exam population
Type of survey	Cross-sectional
Data collection method	Questionnaire
Target population	SE Exam past papers not more than 10years old
Type of information being sought	Specific exam year, knowledge and skills tested in exam
Sampling method	Multi-stage: Stage1:universities Stage2:past exam papers
Sampling technique	Random simple: stage1 Stratified: stage2 Stratum1:degree program Stratum2:exam paper
Sample size	25

3.2.3 Laboratory experiment

In Software Engineering and generally in computing, there is a predefined way of carrying out experiments. Pfleeger (1995) has elaborately defined the six steps to follow as: 1) conception, 2) design 3) preparation 4) execution 5) analysis 6) dissemination and decision making. Besides, Wohlin *et al*. (2003) outlines basic principles that should be observed before an experiment is conducted. Following these guidelines, laboratory experiments enabled the researcher to validate the proposed conceptual model generated in research question one as well as to build and evaluate the machine learning model as per the research questions three and four respectively.

Through experiments the model performance was evaluated by evaluating results obtained with the model. The model was experimented with three kinds of datasets. First dataset was manually created from employees profile data collected from survey conducted in the domain of Software Engineering. Second dataset was a benchmark dataset derived from literature. Third dataset was manually created from employees profile data collected from survey in the domain of academic librarians. Basically, the experiments were guided by three questions: 1) what is the performance of the model in mapping graduates' skills to industry roles? 2) How do we ensure the validity of the results? Answers from these experimental questions enabled the researcher to provide answers to three research questions, research question 1, 3 & 4. Using Pfleeger's (1995) strategy Table 3.2d outlines design characterization that was generated for the experiments.

Table 3.2d: Characterization of research experimental design (adapted from Pfleeger (1995))

Step	Design element	Research Question 1	Research Question 3	Research Question 4
1. Conception	Research question	What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?	How do we build an appropriate machine learning model for mapping graduates' skills to hierarchically structured occupational industry roles?	How do we evaluate the performance and validity of the machine learning model?
	Experiment objective	**Exp. A:** To select relevant features for the model	**Exp. B:** To select relevant parameter values for the model **Exp. C:** To estimate generalization performance of the model	**Exp.D:** To evaluate model performance using three different datasets
1. Design	Hypothesis	H_{0A}: All features are equally relevant for better performance of the model	H_{oB}: Any parameter value induces better performance in the model H_{oC}: All induction algorithms induce equal performance to the model	H_{0D}: There is no significant performance difference of the classifier model in different industry domains
	Experimental unit	Graduate Employees (Software Engineering Domain Field & Lit)	Graduate Employees (Software Engineering Domain Field & Lit)	Graduate Employees (SE(Field & Lit) Academic Librarians(Field)
	Experimental subjects	ML Models (filter algorithms)	ML Models (induction algorithms)	ML Models
	Dependent (response) variable	Performance (accuracy)	Performance (accuracy)	Performance (accuracy, precision, recall, f1_score)
	Independent (state) variables	Features, parameters, algorithms	Features, parameters, algorithms	Features, algorithms
2. Preparation & Execution	Data preprocessing	Training dataset, test dataset	Training dataset, test dataset	Training dataset, test data set
	Randomization	6-10 random trials	6-10 random trials	6-10 random trials
	Local control	5-fold cross-validation	5-fold cross-validation	5-fold cross-validation
3. Analysis	Pre-Analysis	Selection of analysis technique	Selection of analysis technique	Selection of analysis technique
	Main-Analysis (Model Evaluation)	**Evaluation of model accuracy using benchmark (dataset2):** Approach : Hypothesis testing Technique :ANOVA, Paired sample T Test Significance value: 0.05	**Evaluation of model accuracy using Field & benchmark (dataset2):** Approach : Hypothesis testing Technique :ANOVA, Paired sample T Test Significance value: 0.05	**Evaluation of model performance differences in three datasets: Accuracy, Precision, Recall, F_score** Approach : Hypothesis testing Technique : Paired sample T Test Significance value: 0.05

The need to use laboratory experiments was as a result of the following reasons: 1) need to manipulate one or more variables as observed in the data collected, such as in this case knowledge and skills variables, 2) need to identify precise relationships between small numbers of variables, such as in this case knowledge-cum-skills variables and industry role variable.

3.3 Research Framework

Research framework operationalized the research design to provide answers systematically to the main research question: How do we build a data driven model using machine learning for mapping graduate's skills to hierarchically structured industry roles? There were several approaches in machine learning and especially in data mining which could be used to operationalize the research design, but one that was considered significantly important and also widely used in data mining and was both technological and sector independent was Cross Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM aims at making projects less costly, more reliable, more manageable, faster and, most importantly, more repeatable (Wirth & Hipp, 2000).

The six main phases of CRISP-DM model are:

1) Business understanding – understanding objectives and requirements from business view.
2) Data understanding – familiarizing with data quality and interesting subsets
3) Data preparation – constructing dataset from initial raw data
4) Modeling – selecting modeling techniques and parameters for model building and assessment
5) Evaluation – assessment of model results
6) Deployment – generating a report or implementing a repeatable data mining process

However, Guruler & Istanbullu (2014) note that CRISP-DM model is highly recommended for technical projects that follow a structured plan-do-check-act (PDCA) cycle and therefore to achieve optimized quality and success in data mining projects they recommended combining the two. PDCA cycle is a quality-driven approach to change and problem-solving that consist four phases:

1) Plan – identify and define the problem
2) Do – develop and test a potential solution
3) Check – measure how effective the tested solution is and whether can be improved
4) Act – implement the improved solution

Therefore, to ensure high quality and reliable results CRISP-DM model and PDCA cycle were combined and presented in a research framework with ten systematic stages as shown in Figure 3.1 and described below:

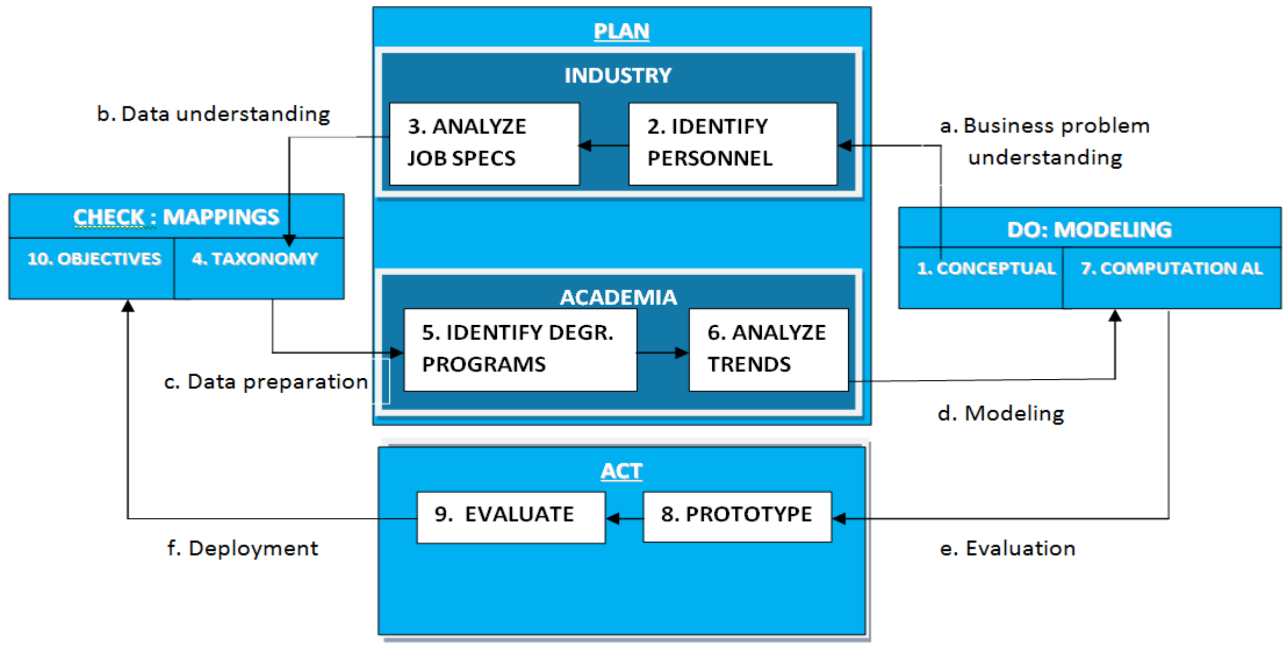


Figure 3.1: Research framework as adapted from Guruler & Istanbullu (2014)

While PDCA provides the underlying blueprint for the research design, CRISP-DM provides the operational activities to realize the end product of the research as follows:

a. Business problem domain understanding – understanding objectives and requirements from business point of view. This was achieved through three operational activities: identifying in the literature 1) factors that promote performance and productivity in the job, 2) various industry roles occupied by personnel in a given occupational domain and 3) bachelors degree programs in the academia that provide a source of skills towards these occupational industry roles.

b. Data understanding – familiarizing with data quality and interesting subsets. This was achieved through two operational activities: data collection and analysis of 1) identified job specifications for occupational industry roles and 2) trends towards those roles in the academia so as to establish institutions' biases towards industry roles. The aim of this task was to verify validity of the initial assumptions of the current study that industry roles are hierarchically structured and there was skills bias in various institutions in the academia towards these industry roles.

c. Data preparation – constructing dataset from initial raw data. This involved transforming data and mapping it into the proposed taxonomic structure and selecting the most meaningful features using standard machine learning techniques.

d. Modeling – selecting modeling techniques and parameters for model building and assessment. This involved computational modeling by simplifying the phenomenon of interest to be studied where the best model was selected.

e. Evaluation – assessment of model results. This involved creating a prototype of the model and assessing its performance.

f. Deployment – generating a report or implementing a repeatable data mining process. This involved mapping the evaluation results to the original objectives so that conclusions could be drawn.

3.4 Research Methods

Research methods refer to schemes, procedures, algorithms and techniques that were used to perform research operations that included data collection, data analysis, and results evaluation. Three categories of research methods were applied: 1) data collection or sampling methods, 2) data analysis methods, and 3) evaluation methods.

3.4.1 Sampling

Data collection, also known as sampling, focused on availing data for the study. Depending on the research design method, the source of data could be literature review where literature analysis was used as data collection method, survey where questionnaires were used as data collection methods, experiments where experimental observations were used as data collection method, and case study where a variety of data collection methods could be employed. In the current sub-section, the focus was data collection methods that availed data from the population of study.

1) Target Population

We targeted two populations in one domain of occupation: past exam papers of degree programs in the academia and graduate employees, both belonging to the same domain. A domain expert was used to create a checklist that was used to filter the industry firms and universities' degree programs from which the target personnel and exam past papers populations were formed (Refer to 3.5.2.1).

2) Sampling Method

The goal was to use a sampling method that would make the study as representative of sources of knowledge and skills as possible and that truly reflects the multi-university environment in the academia across the country. Therefore, multi-stage sampling technique was applied to draw the two samples i.e. each sample created in two stages. For employees, sampling of industry firms was performed (stage 1), before employees were sampled from each firm (stage 2). For exam past papers, sampling of degree programs was done (stage 1), before exam past papers were sampled from each degree program (stage 2). Refer to sub-section 3.5.2.1 for specific details.

For employees' sample, simple random sampling was used to generate stage-1 sample of the firms and stratified random sampling was applied to select stage-2 sample of employees (so that each firm contributes employees to employees' sample). For exam past papers' sample, simple random sampling was applied to select stage-1 sample of universities where the required bachelor's degree programs were offered and stratified random sampling was applied to select stage-2 sample of exam past papers (so that each degree program sampled contributes to exam past papers' sample).

Three types of questionnaires were designed, two to collect data from employees (ordinary employees and head of department/sections separately) and one from exam past papers. Employees' questionnaire was used to collect data for various job titles and their requirements in terms of content knowledge, cognitive skills, technical skills, and academic capacity. Analysis questionnaire was used on exam past paper to collect data on cognitive skills and content knowledge.

For each exam past paper, each question was split into two parts i.e. verb and topic parts. The verb part was used as the indicator for the cognitive skills, while the topic part was used as the indicator for the content knowledge. Bloom's taxonomy has been used as a reference framework for extracting cognitive skills from each question's verb part, while domain's body of knowledge has been used as a reference framework for extracting content knowledge from the topic part of the question. Dalton and Smith (1986) verb list was used to map the verb part of each question to Bloom's taxonomy. The marks awarded to the question were recorded as the value for the cognitive skill as well as knowledge type of the question. Two domain experts have been used to evaluate the past exam papers and their results were correlated.

For each employee's job title, requirements for content knowledge, cognitive skill, technical skill, and academic capacity have been assessed in a set of lickert scale type of competence item/sub-variable matrix whose score range from 1=least important to 12=most important.

The three questionnaires details are summarized below:

1) Questionnaire to collect data from each employee (inexperienced). Details collected were:

- Personal information (gender, age, university of study, degree program, year of graduation)
- Academic performance (secondary school performance, undergraduate performance, domain area subjects' performance)
- Domain area industry requirements (job title, job activities, domain area knowledge demands, Cognitive skills demands,).

2) Questionnaire to analyze and collect data from each past exam paper. Details collected were:

- Exam information (university and year of administration, degree program name, number of questions, total marks allocated, duration)
- Exam content (knowledge area covered and rating, cognitive skills covered and rating).

3) Questionnaire to collect data experienced personnel (Leader or expert). Details collected were:

- Firm/Department information (Regional size, staff size, products or services delivered)
- Domain area job titles (graduate entry level titles, minimum entry grades, job activities, title knowledge area rating, title cognitive skills rating, title technical skills rating)

3.4.1.1 Reliability and Validity of Research Instrument

The data reliability, internal-consistency reliability coefficients for all completed questionnaire (during both pre-test and actual survey) were determined using Cronbach's alpha. The questionnaire was administered to the same respondents two times. After the fast administration, some time was allowed to elapse, long enough to eliminate response by remembering the responses in the first administration. The scores on the two sets were then correlated and reported.

The validity of the instrument was achieved through a pilot study using a section of the respondents in each of the cases of the study before the actual study was conducted. This was necessary to determine whether the respondents would find the questions in the questionnaire precise and concise to the subject of the study. Any questions found ambiguous to the study was restructured to make the instrument more valid.

Ethical issues or norms are important in research because they tend to deal with and discourage cheating through falsifying and fabricating. Ethical issues are important in promoting truthfulness, honesty, social responsibility and integrity in research (Shamoo *et al*, 2009).This research adopted the following ethical principles as adapted by (Shamoo *et al*, 2009; NAS, 1995):

i. Honesty – was achieved through citing relevant sources of information as used in the research
ii. Objectivity- was achieved by following the format of research as provided by the School of Computing and Informatics, University of Nairobi.
iii. Integrity –was achieved by ensuring the research design and data was valid and reliable through validating research instruments.
iv. Legality – was achieved through complying with the laws governing research in Kenya by acquiring permission and authorization to conduct research from National Council of Science and Technology (NCST) of Kenya which is the board in charge of research in Kenya.

3.4.2 Data Analysis and Presentation

Data analysis focused on establishing relationships between the data and the unknowns. The choice of data analysis method depended on the nature of the unknowns, namely qualitative or quantitative. For example, to establish relationship between theoretical concepts in various theories, literature data required qualitative analyses methods such as qualitative comparison analysis technique; to establish descriptive summaries in a population of study, survey data (questionnaire or interview collected) required quantitative analyses methods such as descriptive statistics techniques.

We had four unknowns which were largely quantitative and these were the independent variables, namely Content Knowledge, Cognitive skill, Technical skill, and Academic Capacity. Each was assessed in a set of lickert scale type of sub-variable matrix (competence item) whose score range from 1-12; least important to most important. Each sub-variable score on each competence item was then aggregated to a total score which was then divided by the maximum possible score of all competence items then multiplied by twelve to reduce the sub-variable score to an index ranging between 1 and 12. An average index was then calculated from values of all sub-variable scores to give overall index for each independent variable (refer to section 3.7.1).

Thus, to achieve the objectives of data analysis, data collected through the questionnaires was pre-processed using Excel spreadsheets before creating the data files and subjecting the data to actual analysis.

3.4.2.1 Data Pre-Processing

In order to use the proposed model, data was modeled according to the four independent variables. The industry role requirements for each variable were captured and calculated using the tables described below. With the exception of Capacity variable where the table content type is indicated as column headings High School GPA and Undergraduate GPA, the rest derived their content type from reference frameworks.

The reference framework for content knowledge variable was based on the specific domain's body of knowledge, for technical skills variable it was based on specific domain's competence framework, while for cognitive skills variable it was based on Bloom's taxonomy as the cognitive framework. Data collected from industry experts, heads of sections, was used for reliability validation, while data collected from employed graduates and past exam papers was used as values for individual cases. Excel worksheet was used in the preprocessing of data. The researcher proposed to use the coding scheme described below.

1) **Content knowledge (Relevance):** Under each content type (subject or topic) a value on the scale of 1 to 12 was used to indicate the level of importance for each requirement, where 1 = least important, 12 = most important. The totals were then calculated for each content type, i, and a ratio r_i (i=1,..,n) was calculated and rounded off to a whole number ranging from 1 to 12. An average was then calculated from all r for each content type to get an index value R for the role. Table 3.4 illustrates the layout for calculating the content knowledge index.

Table 3.4: Computing the Content Knowledge Index

Role/Career:	Relevant Content Required: (either Topics or Subjects denoted by C)				
Requirements	C 1	C 2	C 3	----------------	C n
a					
b					
-					
Possible Total(T)					
Calculated total(t)					
r=t*12/T					

2) **Cognitive skills (Durability):** Under each core skills area (subject or topic or competence) a value on the scale of 1 to 12 was used to indicate the level of importance for each requirement, where 1 = least important, 12 = most important. The totals were then calculated for each content

type, i, and a ratio d_i (i=1,..,n) was calculated and rounded off to a whole number ranging from 1 to 12. An average was then calculated from all d for each content type to get an index value D for the role. Table 3.5 illustrates the layout for calculating the cognitive skills index.

Table 3.5: Computing the Cognitive Skills Index

Role/Career:	Core Skills Areas Clusters Required: (either Topics or Subjects or Competences denoted by C)				
Requirements	C 1	C 2	C 3	----------------	C n
a					
b					
c					
-					
Possible Total(T)					
Calculated total(t)					
d=t*12/T					

3) **Technical skills (Accuracy):** Under each core area (subject or topic or competence) a value on the scale of 1 to 12 was used to indicate the level of importance for each requirement, where 1 = least important, 12 = most important. The totals were then calculated for each content type, i, and a ratio a_i (i=1,..,n) was calculated and rounded off to a whole number ranging from 1 to 12. An average was then calculated from all, a, for each content type to get an index value A for the role. Table 3.6 illustrates the layout for calculating the technical skills index.

Table 3.6: Computing the Technical Skills Index

Role/Career:	Core Areas Cluster Points: (either Topics or Subjects or Other denoted by C)				
Requirements	C 1	C 2	C 3	----------------	C n
a					
b					
c					
-					
Possible Total(T)					
Calculated total(t)					
a=t*12/T					

4) **Academic capacity (Capacity):** Individual's capacity for each role/career was derived from both high school and undergraduate Grade Point Average (GPA) which each was converted to decimal number where 1 = E, 2 = D-, 3 = D, 4 = D+, 5 = C-, 6 = C, 7 = C+, 8 = B-, 9 = B, 10 =

B+, 11 = A-, 12 = A. Then, an average was then calculated from the two to get an index value C for the role capacity index value. Table 3.7 shows the layout for calculating the academic capacity index.

Table 3.7: Computing the Capacity Index

	High school GPA	Undergraduate GPA
Grades Points		

3.4.2.2 Creating the Data Files

After the above pre-processing the data was then entered into SPSS software version 6, case by case, and stored in four separate files under the following variables:

1) Employees' data file

This file was used to store data from employees' questionnaires under the following variables (see table 3.8).

Table 3.8: Employee data variables description

VARIABLE NAME	VARIABLE DISCRIPTION
1. Gender	Gender
2. Agebracket	Age
3. Olevelstudyregion	O level study region
4. Ogradingsystem	O level grading system
5. Oresults	O level results
6. Bachelordegree	Name of first degree
7. Bacheloruniversity	First degree university of study
8. Graduationyear	Graduation year
9. Bachelorgradingsystem	First degree grading system
10. Bachelorresults	First degree results
11. Entryleveljobtitle	Entry level Job title
12. Entryleveljobyearofappointment	Entry level Job year of appointment
13. Firstjobdescription	First/entry level job description
14. Currentjobtitle	Current job title
15. Currentjobyearofappointment	Current Job year of appointment
16. currentjobdescription	Current Job description
17. DSubjectstudyyear	Domain area Subject year of study
18. DSubjectscore	Domain area Subject score
19. Jobactivitycategory	Job activity category
20. KAratingRI	Relevancy Index calculated from Knowledge Area ratings
21. CSratingDI	Durability Index calculated from Cognitive Skills ratings

2) exam past paper file

This file was used to store data from domain area exam past paper questionnaires under the following variables (see table 3.9).

Table 3.9: Exam past paper data variables description

VARIABLE NAME	VARIABLE DISCRIPTION
1. Papercode	Subject code for the exam paper
2. Universityname	University name
3. Examyear	Exam year
4. Examduration	Exam duration
5. Totalmarks	Total marks
6. Yearofstudy	Year of study
7. Coursename	Degree program name
8. Totalquestions	Total number of questions
9. BoKnowledgerating	Body of knowledge area ratings (several variables depending on domain area)
10. Krating	Knowledge Acquisition rating
11. Crating	Comprehension rating
12. Anrating	Application rating
13. Aprating	Analysis rating
14. Srating	Synthesis rating
15. Erating	Evaluation rating

3) Industry firm file

This file was used to store the details of the firm or department from which the employees were sampled under the following variables (see table 3.10).

Table 3.10: Firm data variables description

VARIABLE NAME	VARIABLE DISCRIPTION
1. Ownership	Ownership of the firm
2. Staffsize	Number of staff related to the domain area
3. Totaljobcategories	Total number of job categories
4. productsdelivered	Name of product or service delivered by the firm
5. Entryleveljobcategories	List of entry level job categories

The three files were then be used to perform the following analyses on the data collected and stored in SPSS format:-

1) Demographic characteristics analyses
2) Industry role requirements analyses.
3) Trends analysis.

All data analyses were conducted using SPSS software version 16 while experimental analyses have been done using python 4.3 version.

3.4.2.3 Demographic Characteristics Analysis

The purpose of this section was to analyze the demographic characteristic of the sample. The data was analyzed quantitatively to reveal the demographic characteristics of the two survey samples and experimental samples as follows: 1) Frequency distribution table and chart for employees sample to reveal types of bachelor's degree program, gender and industry roles distribution, 2) Frequency distribution table for exams past papers sample to reveal types of bachelor's degree program, year of study, number of questions and total marks distribution, 3) Frequency distribution table for experiment samples to reveal classes distribution.

3.4.2.4 Industry Role Requirements Analysis

The purpose of this section was to analyze competence (content knowledge, cognitive skills, technical skills and academic capability) requirements for each job title so as to reveal knowledge and skills landscape for various industry roles based on our conceptual model (CWA16458, 2012). This was important not only to many stakeholders who have interest in the recruitment and development, education and training, and qualifications and certifications of professionals (Korte *et al*, 2013) but also to provide transparency in validating our assumption that occupational industry roles are distinct and therefore are feasible for machine learning classification.

This was achieved through the following quantitative analyses: 1) Frequency distribution charts for employees sample to reveal types of entry level industry roles, job activities and their proportions, 2) Factor analysis for employees sample to reduce data redundancy using principle component method. Factor analysis, as a statistical procedure for identifying underlying variables (called factors) that explain most variation using fewer variables observed in the original data and principle component method, was used to achieve this, 3) Descriptive statistical analyses for employees sample to reveal central tendency, dispersion values for each independent variable and eventually calculate class boundaries and the index vector for each industry role.

The index vector consists of four values: The minimum index value (Mn), maximum index value (Mx), average index value (Iv) and relative index value (I_R) of each predictor variable (content knowledge, cognitive skills, technical skills, and academic capacity) in each role category as shown in table 3.11. 5) Significance tests analyses for employees sample to test differences between industry roles.

Table 3.11: Role Categories' Indexes minimum and maximum values

Role category name	Content Knowledge (Relevancy Index)				Cognitive skills (Durability Index)				Technical skills (Accuracy Index)				Academic capacity (Capacity Index)			
	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R
1.																
2.																
……… ….																

3.4.2.5 Trend Analysis

The purpose of this section was to analyze trends of knowledge and skills transferred during training in the academia so as to reveal biases towards industry roles among institutions of academia (Topno, 2012). Jones *et al.* (2009) provides a green light that examinations are key tools to evaluate in order to determine whether the test questions' items contain the knowledge and skills desired of learners at the end of training.

The data stored in the exam past paper file was analyzed to show and compare the index values for cognitive skills and knowledge content for each academia institution sampled using the following quantitative analysis procedure: 1) Descriptive statistical analyses for exam past paper sample to reveal central tendency and dispersion values for content knowledge and cognitive skills for each institution, 2) Box plot charts to graphically present the index means and inter-quartile range for various industry roles, 3) Reference lines representing central tendency value for content knowledge and cognitive skills of each institution superimposed on the box plot charts.

3.4.3. Evaluation Methods

Evaluation focused to establish the relevance of research results obtained. For example in literature analysis, triangulation was applied to establish legitimation of the results, while in survey, experiments, or case study, statistical significance tests techniques were used. In the present study, the main focus was to build a classifier model and therefore evaluation was significantly needed to determine its performance and validity. To evaluate the performance of the classifier model a number of experiments were performed according to experimental design described in section 3.1. The choice of evaluation method depended on the choice of evaluation criteria, also known as performance measure. Chapter 6 discusses the evaluation methods and metrics chosen.

3.5. Methodology for Developing the Mapping Model

3.5.1 Problem Domain Understanding

The most important task was to first understand the problem domain, namely mismatch of skills and industry roles. To provide focus towards this problem and narrow down the scope, the problem was treated somehow as an evaluation challenge in the academia where evaluation was limited to learning objectives instead of evaluation of learning outcomes that promote performance and productivity in the job. Based on this in mind and the wide availability of data in a society where data driven methods are gaining traction, the researcher posed a research question towards solution to this problem: how do we build a data-driven model using machine learning to map graduates' skills to hierarchically structured industry roles. However, to answer this question several questions needed to be answered as outlined in section 1.4.

This was done within the context of a selected case of industry occupation.

3.5.1.1 A Case of Software Engineering

We selected the domain of Software Engineering (SE) as a typical case of occupational industry roles. This domain has been widely studied in research literature (Moreno *et al.*, 2012; Shashidhar *et al.*, 2015). Software Engineering (SE) is an industry occupation concerned with development of software that is reliable, efficient and economical. Software developers or engineers refer to the entire community of people involved in software development or working in the SE industry. The universally recommended source of knowledge and skills for software engineers is known as Software Engineering Body of Knowledge (SWEBOK).

Equally, kind of technical skills required of software developers were revealed by a study carried out by Surakka (2005) which grouped these skills into five categories: platform skills, programming skills, networking skills, database skills and distributed technology skills. Software developers are trained along with other ICT practitioners through a number of degree programs such as Computer science, Information Technology, Software Engineering, Mathematics and Computer science.

3.5.1.2 Mismatch of skills and industry roles

A review conducted on literature (Ludi & Collofello, 2001; Saiedian, 2002; Kolding & Ahorlon, 2009; Shkoukani, 2012; Moreno *et al*, 2012; OECD, 2012; McCowan, 2016) revealed indeed there was a problem of skills mismatch between graduates produced by academia and industry roles requirements. Since literature (Griffin, 2008; Sutherland *et al.*, 2009; Norwood & Briggeman, 2010) indicates problem solving skill is poorly evaluated, poor approaches for skills mapping to industry roles may have contributed partly to this situation.

This problem may have rendered both graduates and employers difficult in matching graduates' skills with industry roles. There was need to focus the study to its main goal, namely to build an effective machine learning model for mapping graduate's skills to matching industry roles in hierarchically structured class taxonomy so as to be able to predict suitable industry roles for new graduates based on their skills.

At this point there was need to understand issues that affected the design and building of such computational models that learn from observations. Lavesson (2006) outlined these issues as: 1) the input 2) the type of feedback, and 3) the way the solution is to be represented. Input consists of a sample of data instances described by a number of attributes which may be of different data types, feedback for observational learning could be of three types: supervised, unsupervised, and reinforcement, while representation of the solution in supervised learning depends on whether the desired output is discrete or continuous and thus could be represented using classifier or regression function respectively.

Clearly, out of these three issues three observations were made. First, there was need to select appropriate attributes for describing industry roles instances to be used for machine learning. Secondly, based on feedback requirements of this problem where employees with known industry roles were needed to learn the model, then this was conducted as a supervised learning problem (Lavesson, 2006). Thirdly, since the prediction output of the model was to be discrete then it was addressed as a classifier model.

The first observation focused on the first research question: what concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles? To answer this question a systematic investigation was required to identify and analyze theories for evaluating learning outcomes so as:

1) To establish their underlying concepts that promotes performance in the job.
2) To identify suitable frameworks in academia that are suitable for assessing these concepts.

Figure 3.5.1 provides a logical plan that was used to conduct this type of investigation whose findings were fundamental in providing answers to the above research question.

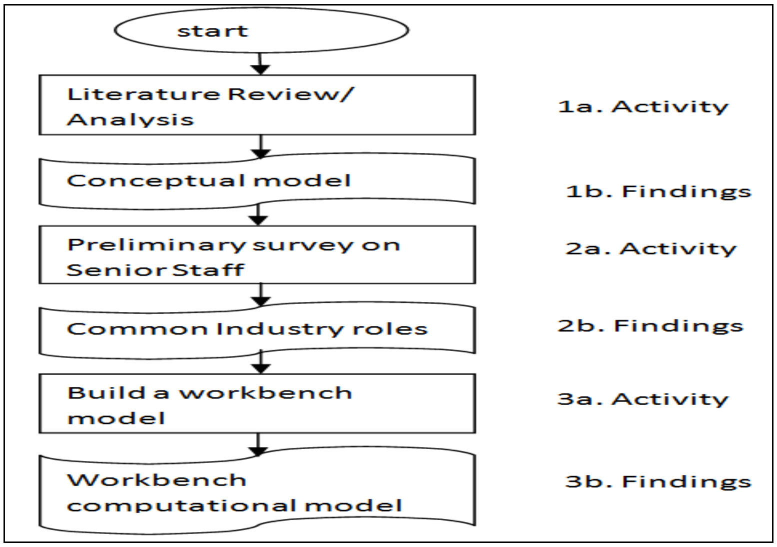


Figure 3.5.1: Understanding problem domain

Activity 1a: Literature Review/Analysis

Literature review was conducted that helped to provide information on concepts that were used to develop the conceptual model of the problem. Problem modeling involved looking at the domain to identify the issue that needed to be addressed and the problem to be solved, and understanding the theoretical issues by which we could model the problem (Vernon, 2009). Keywords in the abstract section of this study were used to select journals for the literature review.

Initially, the keywords guided the searching of literature, then refined using the following questions: What learning outcomes enhance performance in the job? How do we evaluate learning outcomes? Which evaluation methods are common for each learning outcome identified? While literature was the main source, between study literature analysis method was preferred, literature on three theories for learning evaluation were compared and contrasted, complementarity and development techniques of literature analysis were used to achieve representation while triangulation was used to achieve legitimation (Onmuegbuzie *et al.*, 2012). The end product of this activity was the conceptual model within which the concepts to be used as attributes for machine learning were proposed.

Findings 1b: Conceptual Model

Three theoretical models for evaluating learning outcomes were identified, namely Kirkpatrick model, CRESST model, and Kraiger's model. Their underlying concepts were analyzed to reveal ones that promoted performance in the job, and their relationships were represented in the conceptual model. The conceptual model has been presented in Fig. 2.6 and its proposed concepts were operationalized using frameworks that provided indicators that were used to derive the variables for collecting data as shown in Table 2.3.

Activity 2a: Preliminary Survey on Senior staffs

One instrument was designed as a survey questionnaire to collect data from senior employees sample so as to establish employees' industry roles concepts to be used as target classes for machine learning. The instrument was designed to collect data from senior staffs so as to provide insight on three issues.

1) Degree programmes that were the main source of software developers
2) Occupational role names for software developers
3) Main competence areas for occupational industry roles
4) Hierarchical relationship between main competence areas

Findings 2b: Common industry roles

Six broad industry role names were identified for software developers and three degree programs were identified as their main source, namely Computer Science, Information Technology, and Software Engineering. Three main competence areas for software developers were identified and their hierarchical relationship from bottom to top in terms of their skills superiority was established as follows respectively, namely software programmer, software designer and software project manager.

Activity 3a: Build a workbench computational model

A workbench computational model was built to test the feasibility of the study. Computational modeling involved abstracting the problem from the problem space and modeling it computationally by identifying the computational theory and tools needed to solve the problem (Vernon, 2009). The end product was a comprehensive definition of the data input, explicit techniques to process/transform/analyze the data and the information to be produced as output.

3.5.2 Data Understanding

This phase was key not only in familiarizing with the quality of data that was to be used to build the mapping model but also to provide answer to the second research question: what is the structural characteristic of concepts that correctly reflect the hierarchy of industry roles required as target classes for hierarchical machine learning purpose? To answer this question an investigation that needed data collection was launched to:

1) Establish employees' industry roles for entry level jobs in the domain industry that could be used as target classes for machine learning.
2) Analyze job specifications for the occupational industry roles as potential target classes so as to establish their central tendency characteristics.
3) to analyze class boundaries of industry roles as potential target classes so as to establish their uniqueness characteristics,
4) Test significance of the assumption that class boundaries of occupational industry roles were real.
5) establish academia bias towards these industry roles,
6) Establish underlying structure of industry roles.

3.5.2.1 Data Collection

Figure 3.5.2a illustrates the complete data collection process that was started with a preliminary survey on the target populations that led to the identification of three sources that were used as the source of data.

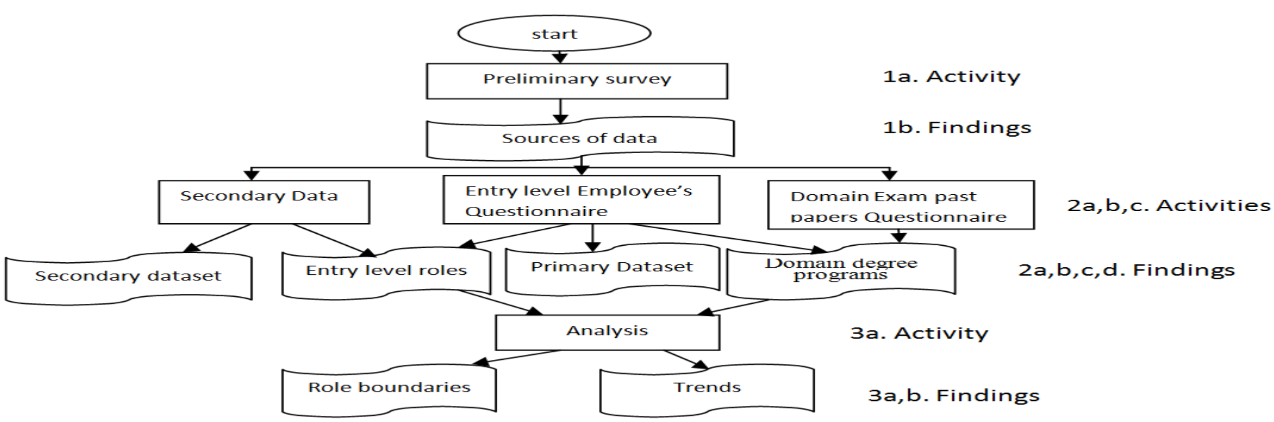


Figure 3.5.2a: Data collection

Activity 1a: Preliminary Survey on Target Populations

Two websites provided the source for software houses as one of our target population (www.kenya-information-guide.com, 2015; www.softkenya.com, 2015). Most software houses are based or centralized in Nairobi because it is the business hub of Kenya and East Africa. Identifiable addresses, physical location and phone number or email address were used as the criteria to locate reachable software houses. Researcher's preliminary survey revealed about 43 software houses working on software development related activities with identifiable addresses, physical location and phone number or email address. These had an average of 10 software developers and a total of about 430 developers. . See Appendix D and Appendix E for sampling frames.

Also, Commission of University Education (CUE) website and other two related websites (www.cue.or.ke, 2015; www.softkenya.com, 2015; www.businesslist.co.ke, 2015) provided the source of university programmes providing SE training courses. This research adopted the list of universities together with their accredited programmes provided by CUE on their website dated 2th February, 2015.

A total of 43 universities (private/public) with a total of 87 bachelor's degree programs in at least computer science or Information Technology or Software Engineering which offered Software Engineering as a course were identified. There was at least one exam for SE each academic year for each of the 87 degree programs. The current study was targeting SE past exam papers from each of the 87 degree programs in a period not more than 10 years, hence a total of at least 870 SE exam past papers. The universities' population excluded university colleges, because it was assumed that the degree programs and exams they offered were originally provided by the mentor university and would cause duplication if included.

1) **Sample Design for the Case**

Sample of software developers was created by first sampling the software houses (as sampling units), then from each sampled software house a second sampling procedure was conducted by selecting the software developers employed (as sampling units). The software developers sample consists of two strata i.e. job entry level software developers (inexperienced) and head of section or department software developers (experienced). 50% (about 22 firms) of the target software houses were selected for stage one sample. This resulted to a total of about 220 software developers from which stage two sample was generated by selecting 22 (department or section head for each firm) experienced developers and 165 inexperienced developers. This gives a sample size of about 187 (44%) software developers of the total population.

Sample of SE exam past papers has been created by first sampling universities where SE related degree programs are offered, then from the sampled universities a stratified random sample of SE exam past papers was created. Bachelor's degree programs of the 43 universities were sampled to create a sample of SE exam past papers by selecting one degree program per sampled university. A total of 5 universities offering the required bachelors' degree programs (12% of the total population of 43 universities) were used to generate stage two sample of SE exam past papers. This resulted to a total of at least 50 exam past papers from which stage two sample was generated by selecting 25 (3% of the total population) SE exams past papers administered in the period of less than 10 years, 5 exam past papers from each of the degree programs.

2) Sampling for the Case

For software developers sample, simple random sampling technique was used to generate stage-one sample of the software houses and stratified random sampling was applied to select stage-two sample of software developers. For SE exam past papers' sample, simple random sampling was applied to select stage-1 sample of universities where the required bachelor's degree programs are offered and stratified random sampling was applied to select stage-2 sample of exam past papers administered in the period less than 10 years (so that each degree program sampled contributes to exam past papers' sample).

3) Data Collection for the Case

Three types of questionnaires were used to collect data, from experienced software developers (experts), inexperienced software developers (recently employed graduates) and SE exam past papers. The three questionnaires details are provided in the appendix (refer to appendix C).

Activity 2a: Secondary Data

After carefully searching for a dataset that would suit the purpose of this method, AMEO2015, one of the datasets listed by Aggarwal *et al.* (2015) was selected as baseline to validate our method. The dataset was downloaded from the web link http://research.aspiringminds.com/resources/. The dataset contains data related to entry level engineers, including software engineers. The dataset has 38 attributes and 3998 instances. AMEO2015 is a dataset comprising cognitive skills test scores (AMCAT test scores), biodata details and employability outcomes of job seekers.

AMEO is an acronym for Aspiring minds Employability Outcomes which is a research affiliated group with the following research objectives: 1) to determine combination of skills needed for various jobs in the market, 2) to provide feedback to candidates on their job suitability, gaps in their skill set for a particular job, and ways for them to improve upon, 3) to provide job credentials to candidates to signal employability, 4) to provide an easy way for companies to filter high quality candidates and provide interview opportunities for them.

In our study, the dataset was carefully analyzed to produce a benchmark dataset. This included the following steps:

1) Filtering out all non-software engineers' data records. Specialization column of the data set was used where all non-Computer Science and non-Information Technology data records were removed.
2) Filtering out all trainees and senior software engineers' data record. Designation column was used to remove any data record that implied a trainee or senior software developer/engineer.
3) Filtering out columns or attributes that were not relevant to our study, such as date of joining, job city, personality attributes, salary, etc. Attributes that correlated to our data collection variables in the questionnaire were retained.
4) Deriving data values for variables that were not directly represented in the dataset, such as age was derived from date of birth and date of joining columns, Relevant content knowledge was derived from domain column, cognitive skills was derived from average of English, Logical, and Quant columns, Technical skills was derived from computer programming column, academic Capacity was derived from average of 12percentage (High school exam grade) and collegeGPA columns.
5) Selecting industry roles whose names clearly indicated a well defined software engineer's role. General names such as software engineers and software developers were ignored.
6) Computing the weights for each of the independent variables for all the industry roles selected.

Findings 2a: Secondary Data

A total of 13 variables were derived from the dataset with 279 data records (instances) and 12 well defined industry roles. Figure 3.5.2b shows a snapshot of the benchmark dataset where the codes

1	GENDER	AGE	LOLE	GSOLE	ROLE	BDGREE	UNIVERSITY	GSBDEGREE	RBACHELORS	R	D	A	C	CLASS
2	2	2	2	1	3	2	44	2	3	11	1.3	6.7	9	1
3	2	1	2	2	3	2	51	2	4	10.1	1	7.3	10.5	1
4	2	2	2	1	3	1	1621	2	4	7.5	1.2	5.9	10.5	1
5	2	2	2	2	3	2	8297	2	4	10.9	1.4	5.3	10.5	1
6	2	2	2	1	3	1	8350	2	3	1	1.1	5.6	9	1
7	2	2	2	2	4	1	9737	2	4	11.4	1.1	6.2	12	1
8	2	2	2	1	3	1	10185	2	4	12	1.3	8.8	10.5	1
9	2	2	2	1	3	1	10932	2	3	1.7	1	4.9	9	1
10	2	2	2	1	4	1	13344	2	4	11	1.5	6.7	12	1
11	2	2	2	1	3	2	350	2	3	1.9	1.9	1.1	4.5	2
12	2	2	2	7	4	1	893	2	3	3	2.3	1.5	5.3	2
13	2	2	2	0	4	2	3838	2	3	2.7	1.8	1.3	5.3	2
14	1	2	2	2	4	1	4973	2	4	0.1	1.5	1	6	2
15	2	2	2	1	3	1	5147	2	3	2.6	2.5	1.3	4.5	2
16	1	1	2	2	2	2	5508	2	4	2.7	2.1	0.8	4.5	2
17	2	2	2	16	2	2	6567	2	3	1.5	1.7	1	3.8	2
18	2	2	2	1	4	1	6907	2	3	3	2.8	1.5	5.3	2
19	2	2	2	7	3	2	7134	2	3	0.3	1.8	0.7	4.5	2

Figure 3.5.2b: SE Benchmark dataset

adopted in the class columns represented the following industry roles extracted: 1:ios developer(9), 2:data analyst (14), 3:android developer(23), 4:java developer(40), 5:programmer(12), 6:software test engineer(42), 7:systems administrator(9), 8:network engineer(8), 9:php developer(19), 10:web developer(32), 11:programmer analyst(51), 12:test engineer(19). Table 3.5.2a describes the main sources of the benchmark dataset attributes relative to the original secondary dataset.

Activity 2b: Entry level Employee's Questionnaire

A survey Questionnaire with 17 items was used to collect data from 113 software engineers in the industry (after data management process). Table 3.5.2b describes the structural characteristic of the employee's questionnaire.

Table 3.5.2a: Description of the benchmark dataset

NO.	ATTRIBUTES	DESCRIPTION	SOURCE (Column name in the original dataset)
1	GENDER	Gender	GENDER
2	AGE	Age	DOB (Date of Birth)
3	LOLE	Place of O-level Study	CollegeCityTier (2=1, 0=1)
4	GSOLE	Grading System of O-level	12 Grade Exam Board (High School Exam. Board)
5	ROLE	Results for O-level	12 Grade Exam Results (High School Results- 4 classes)
6	BDGREE	Type of Bachelor's Degree	Specialization
7	UNIVERSITY	University of Study for Bachelors	CollegeID
8	GSBDEGREE	Grading System for Bachelors	CollegeTier
9	RBACHELORS	Results for Bachelors	CollegeGPA (grouped into 4 classes)
10	R	Relevant Content Knowledge	Domain (converted to out of 12 points = x12)
17	D	Cognitive Skills	Average (Logical, English, Quant) X12/1000
12	A	Technical skills	ComputerProgramming (X12/1000)
13	C	Intellectual Capacity	Average (12 Grade Exam Result, CollegeGPA)
14	Class	Target Industry Role	Designation (Job title)

Table 3.5.2b: characteristics of employee's questionnaire

NO	ATTRIBUTES	VALUES	DESCRIPTION
1	GENDER	{Male, Female}	Gender
2	AGE	{20-24, 25-29, 30-34, 35-39, 40 and above}	Age
3	LOLE	{ Local, Abroad}	Place of O-level Study
4	GSOLE	{Grades, Points, Marks}	Grading System of O-level
5	ROLE	{Less than 4, 5-7, 8-10, 11 and above }	Results for O-level
6	BDGREE	{Computer Science, Information Technology, Software Engineering, Other}	Type of Bachelor's Degree
7	UNIVERSITY	{UON, KU, JKUAT, MOI, EGERTON, Strathmore, KEMU, Daystar, Nazarene, Maseno, Other}	University of Study for Bachelors
8	GSBDEGREE	{Grades, Points, Marks}	Grading System for Bachelors
9	RBACHELORS	{Less than 4, 5-7, 8-10, 11 and above }	Results for Bachelors
10	FIRSTJOB	{Software Architect, Analyst Programmer, Test Engineer, Web Programmer, Mobile Programmer, System programmer, Project manager, Other }	First Appointed Job
11	CURRENTJOB	{Software Architect, Analyst Programmer, Test Engineer, Web Programmer, Mobile Programmer, System programmer, Project manager, Other }	Current Job
12	CHANGEDJOB	{NO, YES}	Current Job Is Different From First Job
13	ATTRACTOR	{Passion, Salary, Ambition, Qualification, Other}	Enticing Factor to Current Job
14	SEEXAM	{100%, 75%, 50%, 25%, 0%}	Se Content In Exam
15	Technical Skills	{interval value}	Index of Technical Skills Components
16	Relevant Knowledge	{interval value}	Index of Content Knowledge Components
17	Cognitive skills	{interval value}	Index of Cognitive Skills Components

1) **Data Pre-processing**

Data collected through the questionnaires was pre-processed and stored in four separate files under the variables described in section 3.4.2. To demonstrate the applicability of this method, SE was used as a case study where SWEBOK content, Bloom's taxonomy, Surakka's technical skills, and Student GPA were used as reference framework as follows:

1) **Content knowledge:** SWEBOK content's topics were used to calculate the index as shown in table 3.12. (for detailed description refer to section 3.4.1 and Table 3.4)

Table 3.12: Computing the Content knowledge Index for the case study

SE role:	SWEBOK Content				
Requirements	Topic 1	Topic 2	Topic 3	----------------	Topic n
a					
b					
Possible Total(T)					
Calculated total(t)					
r=t*12/T					

2) **Cognitive skills:** Bloom's taxonomy was used as coding scheme for cognitive skill areas: K=Knowledge, C=Competence, A=Application, A=Analysis, S=Synthesis, E=Evaluation as shown in Table 3.13. These will be used to compute the index (for detailed descriptions refer to section 3.4.1 and Table 3.5).

Table 3.13: Computing Cognitive Skills Index for the case study

SE ROLE:	Bloom's Competence skills					
Role Requirement	K	C	A	A	S	E
a						
b						
-						
Possible Total(T)						
Calculated total(t)						
r=t*12/T						

3) **Technical Skills:** The Surakka's technical skills for software developers were used to compute index as shown in Table 3.14 (for detailed descriptions refer to section 3.4.1 and Table 3.6).

Table 3.14: Computing Technical Skills Index for the case study

Subjects (Technical skills)	Database (skills)	Networking (skills)	Distributed (skills)	Programming (skills)	Platform (skills)
Grades					
value					

4) **Academic Capacity:** Student capacity for each role was derived from both high school and undergraduate Grade Point Average (GPA) as shown in Table 3.15 (for detailed descriptions refer to section 3.4.1 and Table 3.7).

Table 3.15: Computing Capacity Index for the case study

	High school GPA	Undergraduate GPA
Grades		
Value		

2) Demographic characteristics of SE sample

Figure 4.1.2 in chapter four reveals the results for this stage where seven software engineers' roles were identified as software architect, analyst programmer, test engineer, web programmer, mobile programmer, system administrator, and project manager.

Activity 2c: Domain exam past papers

The purpose of this activity was to identify undergraduate programs through which domain professionals were trained and identify exams past papers for the domain core subject. Descriptive survey methods, especially data collection method using questionnaires was used to collect content knowledge and cognitive skills transferred during learning through a selected set of exam past papers. This stage was achieved through data collection on population 2: Exams past papers. Table 3.5.2b describes the structural characteristic of the exam past papers' questionnaire.

Table 3.5.2c: characteristics of exam past papers questionnaire

NO.	ATTRIBUTES	VALUES	DESCRIPTION
1	EXAMCODE	{Numeric}	Exam Paper Code
2	DEGREENAME	{Computer sciences, Information technology, Software engineering, other}	Degree Name
3	UNIVERSITY	{UON, KU, JKUAT, MOI, EGERTON, Strathmore, KEMU, Daystar, Nazarene, Maseno, Other}	University Name
4	EXAMYEAR	{2009,2010,2011,2012,2013,2014, other}	Exam Calendar Year
5	EXAMDURATION	{1,2,3,4,5 or more}	Exam Durations In Hours
6	STUDYYEAR	{1st,2nd,3rd,4th,5th}	Year Of Study
7	EXAMQUESTIONS	{3,4,5,6,7 or more}	Exam No Of Questions
8	TOTALMARKS	{interval value}	Exam Total Marks
9	SR	{interval value}	Software Requirements
10	SD	{interval value}	Software Design
11	SP	{interval value}	Software Processes
12	ST	{interval value}	Software Testing
13	SCONF	{interval value}	Software Configuration
14	SMAINT	{interval value}	Software Maintenance
15	SI	{interval value}	Software Infrastructure
16	SQ	{interval value}	Software Quality
17	SMGT	{interval value}	Software Management
18	SCONS	{interval value}	Software Construction
19	KNOWLEDGE	{interval value}	Knowledge
20	COMPREHENSION	{interval value}	Comprehension
21	APPLICATION	{interval value}	Application
22	ANALYSIS	{interval value}	Analysis
23	SYNTHESIS	{interval value}	Synthesis

Activity 3a: Analysis of Role Boundaries and Trends

The purpose of this activity was not only to identify various entry level job titles that referred to the occupational domain area but also their boundaries and trends or bias in the academia towards industry roles. Their boundaries along each meaningful attribute were important characteristic in distinguishing unique industry roles while their trends in academia were important characteristic in distinguishing bias of graduates' skills from various academia institutions. Descriptive survey methods were used to analyze content knowledge, cognitive skills, technical skills and academic capability requirements for each job title and establish distinction among industry roles.

For each industry role a minimum (Mn) and maximum (Mx) value under each variable was established, then an average (I_V) was calculated for each variable before it was ranked (I_R) against other roles. Table 4.1.4b in chapter four presents results for this activity.

Further, descriptive statistics analyses were conducted to reveal the central tendency and dispersion values of each independent variable for all industry roles. The most important aspect of this activity was to determine boundaries among revealed industry roles and to test whether class differences between these industry roles was significant. To achieve this purpose a research hypothesis was defined and investigated as follows:

H_{02A}: There are no significant boundary differences between industry roles/potential target classes

To approach this research hypothesis, the four main qualitative variables were classified into two different ways with the help of a 2 by 2 matrix as shown in Table 3.5.2d. One way classified them as either knowledge (content knowledge & academic capacity) or skill type (technical skills & cognitive skills), and the other way classified them as either domain specific (content knowledge & technical skills) or domain general (academic capacity & cognitive skills). Table 3.5.2d shows a two way classification of the independent variables. After which, four research hypotheses were defined and investigated in order to answer this research question as follows:

Table 3.5.2d: Two way classification of independent variables

Variable type	Knowledge	Skill
Domain specific	Content Knowledge	Technical skills
Domain general	Academic capacity	Cognitive skills

Hypothesis 1(H_{01}):

H_0: There are no significant domain specific knowledge differences between industry roles in the same occupation

H_a: There are significant domain specific knowledge differences between industry roles in the same occupation

For this hypothesis, content knowledge variable was used as the test variable and we reject the null hypothesis when the test statistic value (*P*) is less than significance value (.05), otherwise we accept the null hypothesis.

Hypothesis 2(H_{02}):

H_0: There are no significant domain general knowledge differences between industry roles in the same occupation

H_a: There are significant domain general knowledge differences between industry roles in the same occupation

For this hypothesis academic capacity variable was used as the test variable and we reject the null hypothesis when the test statistic value (*P*) is less than significance value (.05), otherwise we accept the null hypothesis.

Hypothesis 3(H_{03}):

H_0: There are no significant domain specific skill differences between industry roles in the same occupation

H_a: There are significant domain specific skill differences between industry roles in the same occupation

For this hypothesis technical skills variable was used as the test variable and we reject the null hypothesis when the test statistic value (*P*) is less than significance value (.05), otherwise we accept the null hypothesis.

Hypothesis 4(H_{04}):

H_0: There are no significant domain general skill differences between industry roles in the same occupation

H_a: There are significant domain general skill differences between industry roles in the same occupation

For this hypothesis cognitive skills variable was used as the test variable and we reject the null hypothesis when the test statistic value (*P*) is less than significance value (.05), otherwise we accept the null hypothesis. Finally, the hypothesis testing results were appended in the two way classification table and interpreted accordingly.

Equally, in this activity descriptive survey methods were used to analyze content knowledge, and cognitive skills administered in the domain core subject exam past papers for each selected academia institution and establish trends/bias towards industry roles. Descriptive statistics analyses were conducted to reveal the central tendency and dispersion values of each of the two independent variables for all institutions and compared these values relative to industry roles revealed. The core aim was to show how different academia institutions were biased towards these industry roles requirements. Table 4.1.6 in chapter presents a summary of the counts of the trending industry roles in each university as revealed by analysis results in chapter four.

3.5.3 Data Preparation

Before the process of building the machine learning model was started a thorough cleaning activity was conducted to put the data into the appropriate shape that promised optimal and reliable results. This involved addressing the following issues: 1) missing data 2) categorical data 3) standard scale for all features 4) selecting meaningful features 5) hierarchical mapping of target classes.

1) Missing data

Two ways for handling missing data according to Raschka (2015) are: 1) eliminating sample instances or data features with missing values and, 2) imputing missing values. The former has several disadvantages such as by removing either many instances ends with reducing the sample size and thus affecting the reliability of the results or too many features we lose valuable information that the classifier model to discriminate between classes. The later is often used where the missing value is estimated using a number of interpolation techniques such as mean imputation. Mean imputation is an interpolation technique where the mean value of the entire feature column is used to replace the missing values of that column. The most convenient way to achieve this in python is to use the imputer class of scikit-learn and is implemented as shown below (Raschka, 2015).

```
>>> from sklearn.preprocessing import Imputer
>>> imr = Imputer(missing_values='NaN', strategy='mean', axis=0)
>>> imr = imr.fit(df)
>>> imputed_data = imr.transform(df.values)
```

In the current study, some feature columns had missing values as well as the target class column. This was as a result of unfilled values in some questionnaires where some employees did not fill simply because they did not have adequate information about the questionnaire item or simply an error of omission during the filling process. To cope up with this problem, the researcher removed all records whose target class values were missing and it was concluded that they were incomplete while for the other feature columns the imputation technique described above was applied.

2) Categorical features

Categorical features have their data values in discrete group or categories. Categorical data values can be nominal (unordered set) or ordinal (ordered set). To ensure that the learning algorithms interpreted categorical data values correctly, we needed to map categorical data and class labels to integers (Raschka, 2015). As observed under the data collection section, most of the features were categorical and therefore they needed some transformation to integer values. In the current study, this was conducted manually.

3) Standard scale for all features

Majority of the machine learning and optimization algorithms work well when features are put on the same scale (Raschka, 2015). However, decision trees and random forests are the only machine learning techniques that do not care for feature scaling. Two common approaches for scaling features observed in literature were normalization and standardization. Normalization refers to scaling all features to a range of 0 to 1. To normalize data a min-max scaling formula that could be applied to each feature is as shown below:

$$X^i_{norm} = \frac{X^i - X_{min}}{X_{max} - X_{min}} \qquad \text{Eqn(16)}$$

Where X^i_{norm} is the new normalized value of an instance value X^i in a feature column where X_{min} is the minimum and X_{max} is the maximum value. The min-max scaling procedure could be implemented as follows:

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> mms = MinMaxScaler()
>>> X_train_norm = mms.fit_transform(X_train)
>>> X_test_norm = mms.transform(X_test)
```

On the other hand, standardization is a way of centering the feature around the mean 0 and standard deviation 1 so that the feature column values take the form of a normal distribution which makes it easier to learn the weights (Raschka, 2015). Therefore, unlike normalization, standardization was likely to maintain useful information about outliers and make the learning algorithm less sensitive to it. To standardize any instance value X^i the formula below was applied:

$$X^i_{std} = \frac{\bar{X}^i - \mu_x}{\sigma_x} \qquad \text{Eqn(17)}$$

Where X^i_{std} is the new standardized value while μ_x and σ_x are the feature column mean and standard deviation respectively. The standardization procedure was implemented as follows (Raschka, 2015):

```
>>> from sklearn.preprocessing import StandardScaler
>>> stdsc = StandardScaler()
>>> X_train_std = stdsc.fit_transform(X_train)
>>> X_test_std = stdsc.transform(X_test)
```

In the present study, preliminary results indicated scaling through standardization produced better results than scaling using normalization. As a result, scaling through standardization was adopted.

4) Hierarchical mapping of target classes.

The goal of this stage was to map the industry roles into the proposed taxonomic structure. Mapping the industry roles to the proposed taxonomic structure involved merging duplicate industry roles or separating industry roles with similar names but different requirements. As result of variation of definition of industry roles in various industry firms, some roles may have elements from more than one role and this might endanger intra-class similarity and inter-class dissimilarity which is an important requirement in classification (Chien & Chen, 2008).

To improve on this, a procedure was devised to divide the dataset into several classes in which the intra-class similarity was maximized while the inter-class similarity was minimized (Chien & Chen, 2008). The original employees' data that contained, among other attributes describing the industry roles, first appointed role after attaining university bachelor's degree as well as current role of the employee was vital in achieving this. This procedure helped also to harmonize role names and boundaries derived from various firms. The procedure was conducted in three steps as described below.

Step 1: Branch Mapping

The aim was to identify industry roles as set members with almost similar characteristics and isolate them into separate branches. Figure 3.5.3a presents a summary of this procedure. First, the mean was calculated for all the features that were core in describing the industry roles. Along each feature, industry roles were partitioned into two sets based on each feature's mean to give two partition sets of industry roles i.e. upper and lower sets. The two sets (i.e. upper or lower) for all the features were listed to get a list of sets. Each set in the list was cross-examined against each feature's two partition sets (i.e. upper or lower) to check whether all role members of the list set were contained jointly in the feature's partition sets. If all members were contained in either one of the feature's partition sets then a score of 1 was noted otherwise 0. This process was repeated for each listed set across all features, and a sum was calculated by adding the scores for all the features.

Therefore, role members of a listed set that occurred frequently and consistently in majority of feature partition sets would constitute a possibly separate branch and was evidenced by high sum of scores. If two or more sets tied with highest score each was noted as candidate for isolation into a branch set only if the sets were disjoint, else only the one with the highest total sample size was isolated.

This process was repeated after removing the isolated branch set from the listed sets and all its members from the remaining listed sets. However, any of the subsequent candidates must have both their set cardinalities and highest scores exceed both the cardinality and highest score of the original set. This was to minimize chances of many branches with very few industry roles. The process was optimal if all remaining listed sets' cardinalities were less than the cardinality of the original branch set. This process revealed new branches and appropriate names were identified for each branch.

Step 2: Mapping Instances to Proposed Taxonomy's Branches

The aim of this step was to identify and isolate instances of the dataset into specific branches based on first and current appointment role values. Figure 3.5.3b shows a summary of this procedure. Before isolation procedure, cross-tabulation of values of the first and current appointment roles was conducted and the following assumptions were noted when doing the isolation using the cross-tabulation technique:

1) employees originally holding industry roles (as first role) belonging to one branch and were still holding those roles currently (as current roles) in the same branch were considered permanently

affiliated to that branch while those converted to other roles in a different branch may be considered to have left permanently

2) Employees who converted from previous roles in one branch and were currently holding industry roles belonging to another second branch were now affiliated towards that second branch

3) If one of the first appointment roles in a branch overlapped or coincided with the branch name then it was removed as a possible name of industry role and its employees who converted to other roles in a different branch were assumed to still belong to the original branch and were redistributed to the original branch roles accordingly.

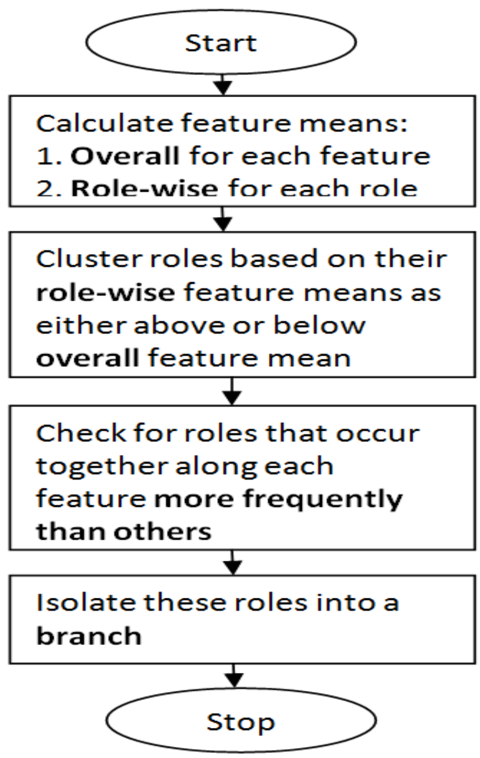


Figure 3.5.3a: Branch Mapping Framework

Step 3: Mapping to Proposed Taxonomy's Hierarchies

The aim was to categorize industry roles into levels in the hierarchy. First, with the help of domain experts, the levels in the hierarchy were identified based on superiority of functionality in the domain, with the most superior at the top and the least superior at the bottom. Secondly, the industry roles were categorized along the levels and their count scores extracted from the cross-tabulation described in step 2. A two by two table was used to relate industry roles in each level with branches by splitting each first appointment role total in the cross-tabulation into respective branches based on the assumptions.

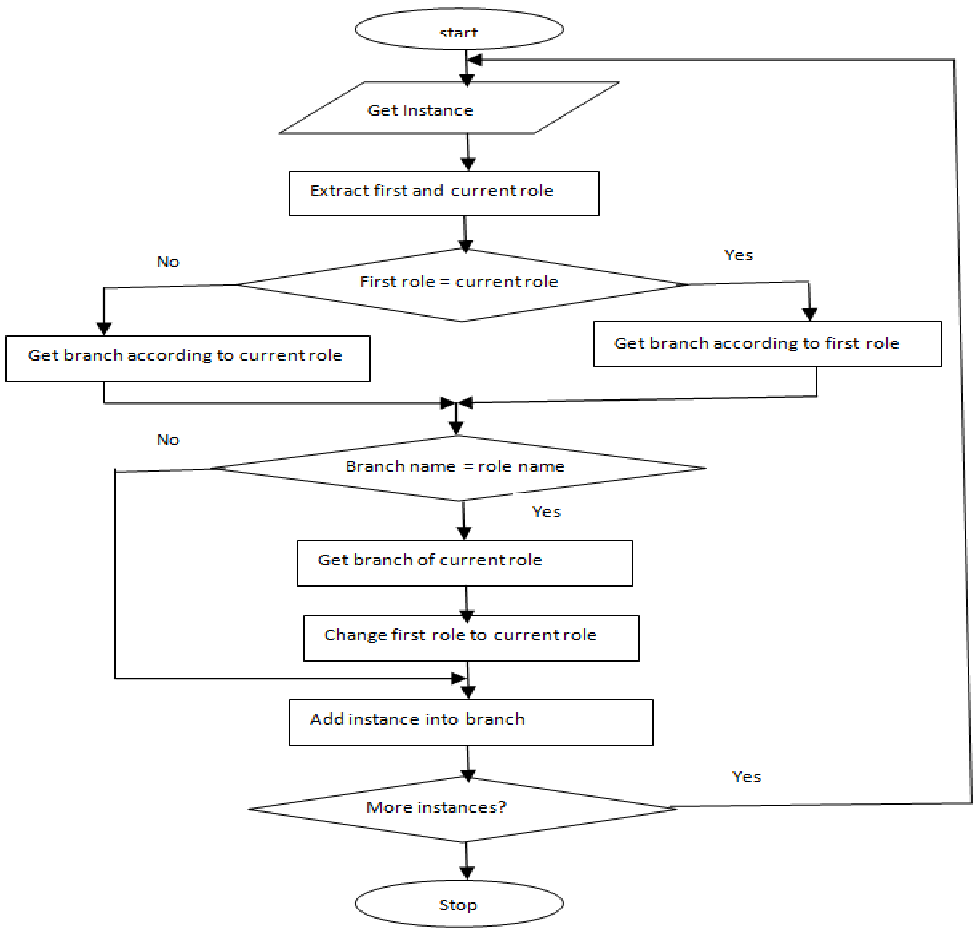


Figure 3.5.3b: Instances Mapping Framework

1) Mapping software engineers raw data to the proposed taxonomy

This procedure was applied to raw data with the original seven industry roles and resulted into twelve distinct industry roles. Fig. 4.2.1 in chapter four illustrates the mapping of 12 industry roles for software engineers into the proposed taxonomic structure using our method. The 12 distinct industry roles have been coded as follows: 1: mobile system manager, 2: mobile project manager, 3: mobile architect designer 4: mobile web designer 5: mobile analyst programmer 6: mobile test programmer 7: desktop system manager 8: desktop project manager 9: desktop architect designer 10: desktop web designer 11: desktop analyst programmer 12: desktop test programmer.

5) Selecting meaningful features

This was one of the most common and important methods applied to data preprocessing so as to improve the performance of the classifier model (Chang, 2009). Real-world classification tasks contain irrelevant or redundant features that may compromise the accuracy of the classifier model. As a result, many feature subset selection approaches were developed to help reduce dimensionality problem (Raschka, 2015). Feature subset selection, as a process of removing irrelevant or redundant features from the original feature set, offered many benefits such as reducing the cost of gathering data for training or testing or even reducing the time for creating the classification model (Chang, 2009).

In the current study, so as to ensure a specific feature subset was optimal, an evaluation strategy was needed. As a result, feature subset selection process was approached as a search problem and was conducted in four stages: 1) starting point for the search space, 2) a generation rule with search strategies to generate the next candidate feature subset, 3) an evaluation function to evaluate the generated feature subset, 4) stopping criterion to determine when to stop the selection process (Chang, 2009). Figure 3.5.3c illustrates the procedure for feature selection.

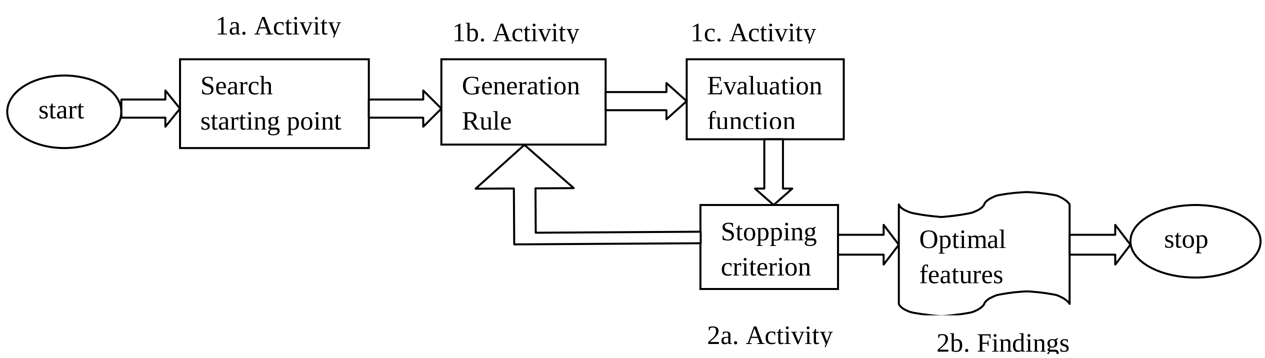


Figure 3.5.3c: Selecting meaningful features

Activity 1a: Search starting point

At the search starting point, a decision was made whether to start with zero features (sequential forward selection method, where features are added successively as evaluation progresses) or start with all the features (sequential backward method, where features are eliminated from the original feature set successively as evaluation progresses). Sequential backward method was selected because it is simple and widely used in machine learning pattern classification methods and, most importantly, previous studies have shown that the technique produces better classification accuracy than sequential forward method (Witten & Frank, 2005). The simple steps for sequential backward method were outlined as follows (Raschka, 2015):

1. **Initialize the algorithm with k = d, where d is the dimensionality of the full feature space X_d.**
2. **Determine the feature x− that maximizes the criterion x− =kargmaxJ(X_x−x) where xϵ X_k.**
3. **Remove the feature x− from the feature set: X_k – 1 = X_k – 1 = X_x− x−, k=k-1**
4. **Terminate if k equals the number of desired features, if not, go to step 2.**

Activity 1b: Generation rule

A rule that generated a subset of features to be assessed based on a certain search strategy was to be selected. Common search strategies for the subsets in the feature space are: 1) exhaustive search (search all possible subsets, becomes difficult as the number of attributes increases), 2) greedy search (search that begins in one direction, top or bottom, and progresses by adding or eliminating a feature to or from the current subset, search terminates when no feature improves on the current subset), 3) best first search (search that keeps a list of subsets evaluated so far and sorted in order of performance measure) , 4) beam search (like best first search but truncates its list to a specified fixed number), 5) genetic algorithm search (search based on evolution or natural selection theory). In the present study, our approach used sequential backward method whose search strategy was greedy search (Witten & Frank, 2005).

Activity 1c: Evaluation function

When selecting a good feature subset, two fundamentally different evaluation approaches that we came across were: independent assessment based on the general characteristics of data and assessment using machine learning algorithm that would be used for the learning of the classifier model (Witten & Frank, 2005). The former are called filters while the later wrappers. Filter approach was used in the current study. Filters assess the features according to their prediction ability using two approaches: ranking method (ranking features according to some predictive measure then the best subset is made of high ranking features) or space search method (maximizing a predetermined cost function where features that maximize this function make up the optimal subset).

Features were selected that other evidence, including more general models fitted into the full dataset, suggest would be important predictors of industry roles as applied by Clive & Joan (2000). The same approach was used successfully elsewhere (Ramaswami and Bhaskaran, 2009; Mgala, 2016) and this informed our decision to use it. However, Mgala (2016) used Information Gain, ReliefF, and Gain Ratio filter algorithms for feature selection where their technique was ranking and comparison across the three algorithms. This was slightly different from current study where Logistic Regression (LR), K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) were preferred. Instead of feature ranking, however, the current study used space search where each algorithm searched for the best feature combination subsets that produced the best performance level.

The best feature subsets results from each algorithm were compared to determine features that were widely selected. This approach, unlike elsewhere (Mgala, 2016), ensured that each feature was popular among the participating algorithm where simple majority was used as a criterion for popularity. Unpopular features were removed. In case of more than one candidate feature subsets, evaluation was conducted with each subset and the one that gave better results was selected as the best feature subset for that particular algorithm. This procedure was conducted with one dataset, namely SE benchmark dataset, through an experimental procedure whose objective was clearly stated as shown below:

Objective: To select valuable feature subset likely to induce optimal accuracy to the model

Procedure:

- Split (ratio 80:20) dataset into 2: train, test sets
- Divide features into subsets using combinations of 2 to all features
- Train and test 3 filter algorithms on each subset
- Get the best subset for each algorithm
- Select features that appear in at least two of these 3 best subsets

Activity 2a: Stopping criterion

The criteria of removing a feature at each iteration was defined as “Remove the feature that maximized the difference in performance of the classifier model after and before the removal of this particular feature”.

Findings 2b: Optimal Features

Section 4.2.6.1 presents results for the current activity of selecting meaningful features. The optimal number of features from the original 13 feature set was then concluded as 5 features. The aim was to determine optimal features that generated optimal performance to the classifier model with the ultimate focus to investigate appropriate features that enabled the model achieve appropriate performance to serve its purpose. In order to investigate whether these generated features were likely to induce optimal performance significantly to our classifier model a research hypothesis was defined to be tested as follows:

H_{01A}: All features are equally relevant for better performance of the classifier model

3.5.4 Modeling and Selecting the best classifier model using the best feature subsets

This phase involved building the machine learning model and ensuring the model was appropriate to serve its purpose. The phase was vital in providing answer to the third research question : how do we build an appropriate machine learning model for mapping graduates' skills to hierarchically structured occupational industry roles? To answer this question it required the following three activities:

1) Design of machine learning algorithm,
2) Algorithm optimization through induction algorithm and parameter selection
3) Model evaluation through estimation of its generalization performance.

Generally, overall implementation of the classifier was achieved using python technology due to its richness in ML resources and simplicity. Fig. 3.5.4a illustrates a typical work flow diagram for using machine learning in predictive modeling.

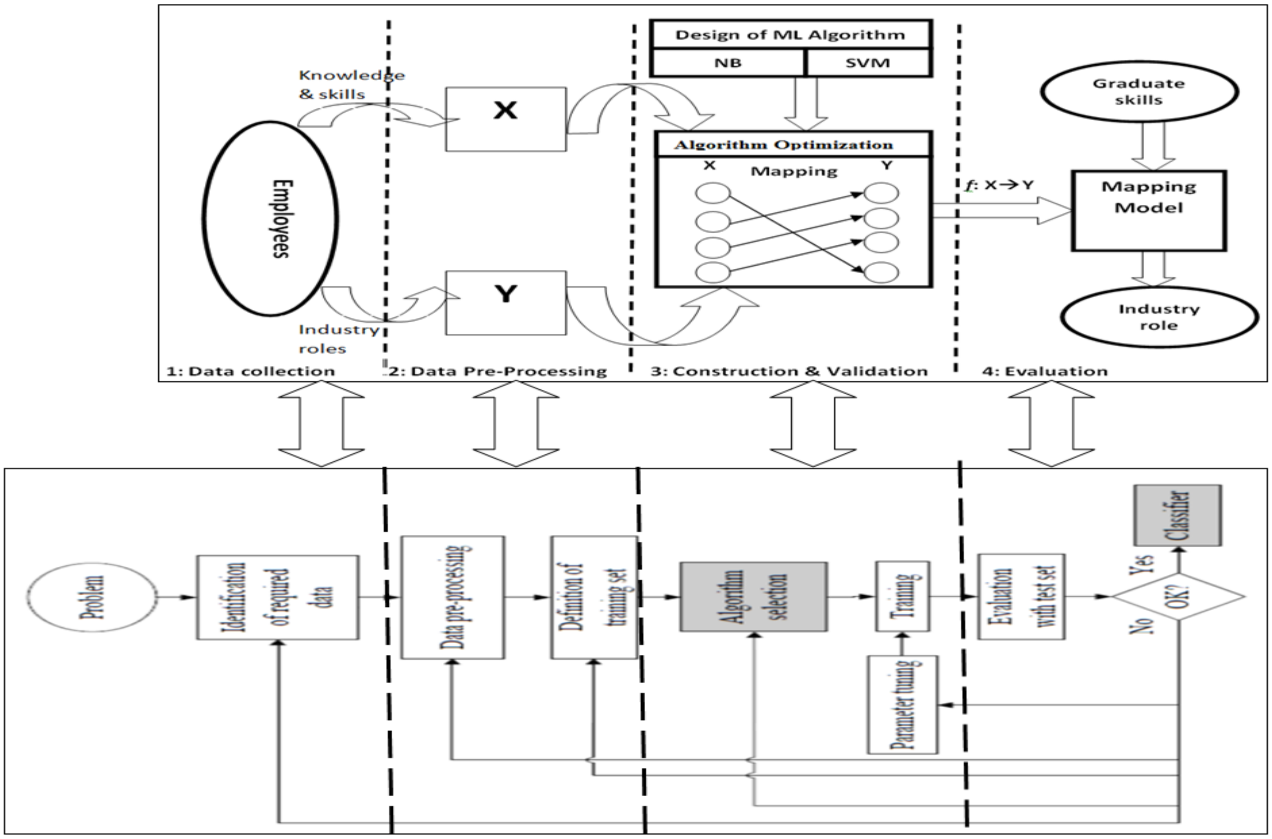


Figure 3.5.4a: Workflow framework for predictive modeling using machine learning (adapted from Raschka, 2015)

3.5.4.1. Design of Machine learning algorithm for the classifier model

Design and building of such a computational model that learns from observations required three considerations, namely: input, feedback process, and output (Lavesson, 2006). Thus, the design architecture of the classifier model consists of three elements: 1) input, the various materials or resources that the model requires to accomplish its purpose and these constitutes three items: employee's data, occupational domain's roles, and the taxonomic structure.

As revealed in Fig. 2.9b *f* represents features or knowledge and skills attributes of industry roles whose data values, for the purpose of building the classifier objects of the model, were derived from graduate employees in the industry holding various roles through data collection stage as emphasized in Fig. 3.5.4a. 2) process, the ML logic that the model applies to transform the input materials or resources into required form and this comprises the ML architecture as given in Fig. 2.9a, 3) output, the prediction result generated by the process. Fig. 3.5.4b outlines the design architecture for the classifier model. This design architecture was eventually converted into a design algorithm.

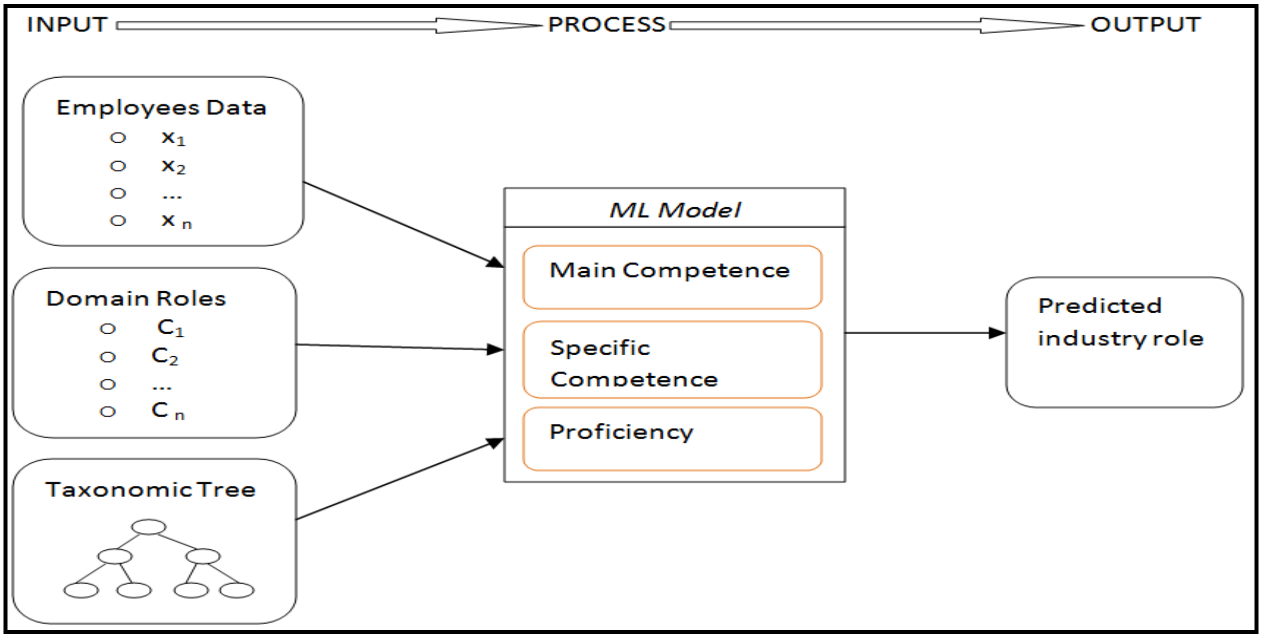


Figure 3.5.4b: Design architecture

Building of the classifier model's algorithm was conducted using meaningful features that were selected in section 3.5.3.

3.5.4.2. Algorithm optimization

This process helped to validate the classifier model by ensuring it had appropriate valid properties to serve its purpose.

a) Through selection of appropriate induction algorithm

This activity involved selecting appropriate machine learning technique for the classifier model. Raschka (2015) observes that choosing an induction algorithm for a particular classification problem required experience because each algorithm has its own quirks and is based on certain assumptions. As a result, it is recommended to compare performance of at least two learning algorithms before selecting the best classifier model for the problem (Drummond, 2006).

In the present study, two machine learning techniques, naïve Bayes and support vector machines were selected in the construction of the classifier algorithm to implement the architecture and learn the model. Section 2.7.7.5 describes the criteria for choosing the two algorithms. Evaluation experiments were conducted with each of the induction algorithms on the classifier model where generalization performance of each was determined. An induction algorithm that induced better performance was selected as the best induction algorithm. This procedure was conducted with two datasets, namely SE benchmark and SE field datasets, through an experimental procedure whose objective was clearly stated as shown below:

Objective: To select induction algorithm likely to induce optimal accuracy to the model

Procedure: 5-fold cross-validation

- Split dataset into 2: train, test sets
- Divide train set into samples of increasing size intervals of 20%
- Split each sample into 2: train, test sets
- Train and test induction algorithms on each sample
- Plot the train and test accuracy of respective samples
- Observe the behavior of accuracy difference as the sample grows
- Split train set into five folds
- Alternately, train with 4 folds and test with 1 fold both induction algorithms simultaneously and ten times
- Get the means in each test fold

In order to investigate whether the induced performance of our model by each induction algorithm was significantly better than the other, a research hypothesis was defined and tested as follows:

H_{03A}: All induction algorithms induce equal generalization performance to the model

b) Through parameter tuning

This was achieved through parameter tuning using validation curves. Only one algorithm was involved in this, namely as per the results of selection of the best induction algorithm in (a) above. The aim was to determine parameter values that generated better performance to the classifier model with the ultimate focus to investigate appropriate values that enabled the model achieve appropriate performance to serve its purpose.

To investigate this, a research hypothesis was defined as follows:

H_{o3B}: Any parameter value induces better performance to the model

Three trials of an experiment were conducted under each dataset whose findings were important in selecting the best parameter values of the classifier model. Figure 3.5.4c illustrates the parameter tuning procedure that was adopted in the experiment.

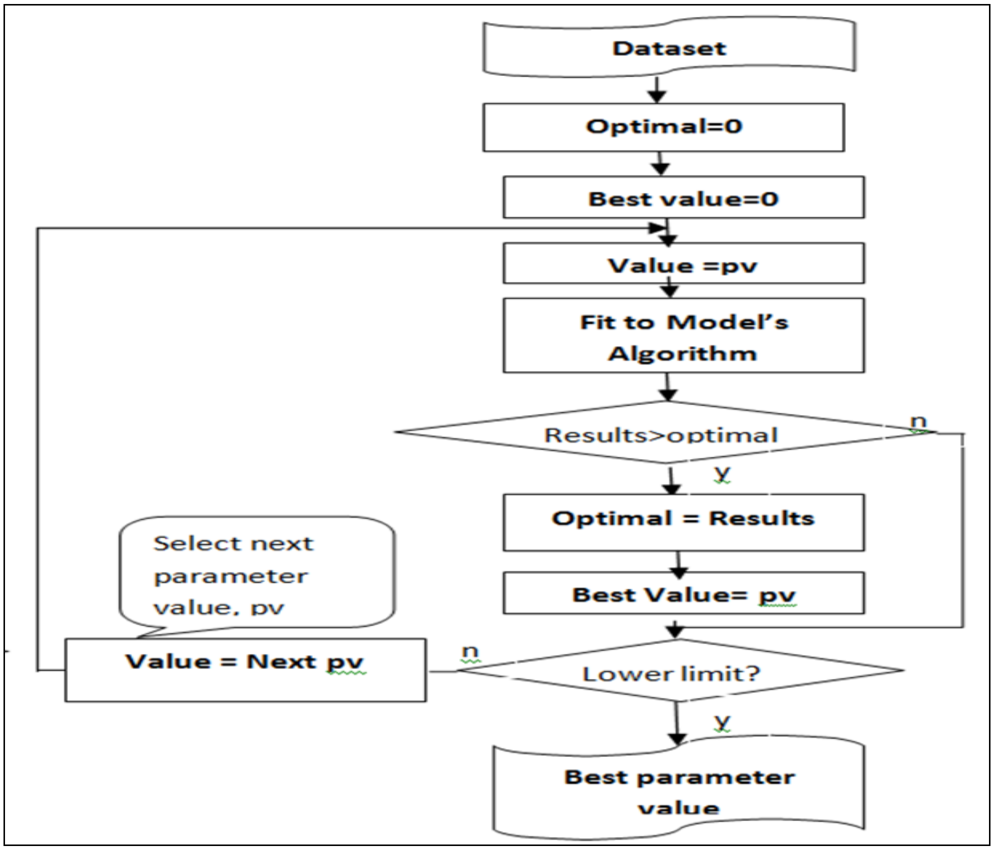


Figure 3.5.4c: Algorithm optimization through validation curve

3.5.4.3. Model validation

This was conducted through a number of experiments and the focus was to estimate the generalization performance of the classifier model. In supervised learning, classification is conducted in two phases, namely training and prediction phase. In the training phase, a learning algorithm trains by observing known data then generates the best classifier that is used to classify new data of same kind. In the present study, this was achieved through cross-validation technique where the best performing model was selected, and this involved a number of activities as described in the diagram below. Fig. 3.5.4d illustrates the model validation process as adopted from Care & King (2003).

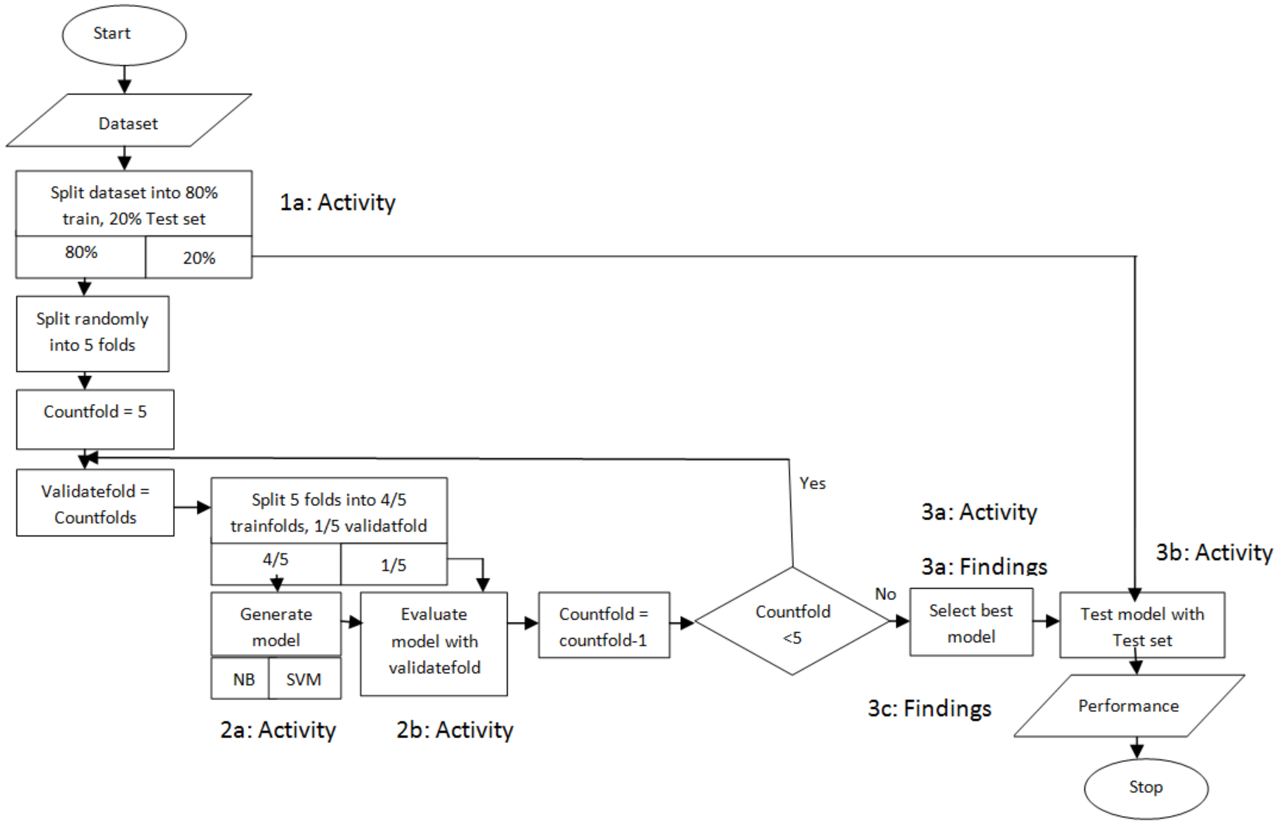


Figure 3.5.4d: Model validation & Evaluation (adapted from Clare & King (2003)**)**

Activity 1a: Splitting Dataset

This involved partitioning the dataset into three sets: training set and testing set where the most common practice is to split in the ratio of 60:40, 70:30. 80: 20, and 90: 10 (Raschka, 2015). It was noted that splitting the dataset amounts to withholding valuable information which could otherwise be beneficial to the learning algorithm while at the same time the smaller the test size the more inaccurate the generalization error (Raschka, 2015).

In order to balance this trade-off, stratified random sampling was adopted to ensure each target class was maintained in either of the two splits to safeguard against poor generalization error. Further, to ensure little information was withheld in the test set which could be valuable to our learning algorithm, a split ratio of 80:20 was selected. In practice, 80:20 split ratio is beneficial to large datasets, however, in case of smaller datasets, as is the case in the current study, cross-validation technique guarantees better results (Raschka, 2015). Consequently, in the current study, 5-fold cross-validation technique was applied to split further the training set into two, train (64%) and validate set (16%), so that together with test set (20%) we got a total of three split sets.

Although 10-fold splitting is recommended, 5-fold was adopted as a result of smaller frequencies of less than 10 in some target classes. While the training set was used to fit the data and learn various classifier models, validate set was used to select the best performing classifier model, and the test set was used as an ultimate test to the model before it was ready to release in the real world. Fig 3.5.4e describes the dataset splitting process.

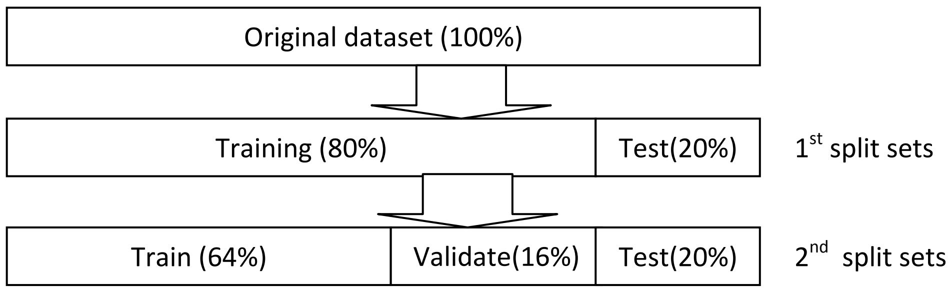


Figure 3.5.4e: Splitting datasets (adapted from Raschka, 2015)

Activity 2a: Generating Model

The classifier model was generated through two learning algorithms, namely naïve Bayes and SVM. Raschka (2015) provided a guiding principle used to select the two learning algorithms, that no single classification model enjoys superiority over others since each classification algorithm used to generate the model has its own inherent biases and assumptions. The best practice is to make assumption about the classification task and use a handful of classification algorithms for comparative analyses.

Each candidate classification algorithm selected to generate the model correlates closely with the main classification assumption made in the present study that occupational industry roles are distinct to each other and their predictors are independently identical. Consequently, the element of independence of identical predictors is the basis of naïve Bayes while the element of distinct classes that are separable is the basis of SVM. Choice criteria for the two algorithms was given in 2.7.7.5.

Generation of the model involved training and tuning iterative processes. Training involved making the learning algorithms learn a map function from features to target classes by analyzing data in the feature set. Tuning involved making the learning algorithms find optimal hyperparameter values that generated satisfactory generalization performance (Raschka, 2015). These twin iterative processes were conducted under well designed experiments and generated a variety of models with different performance levels that demanded careful evaluation strategy. This is because in each iteration the learning algorithms improved their performance of classification as a result of the learning experience derived from data and the result would be a new candidate classifier model.

Activity 2b: Evaluate Model with validate set

In the current study, evaluation of each candidate classifier model was important in three ways: 1) to assess the extent to which the type of parameter tuning affected performance of generated model, 2) to assess generalization performance of individual candidate classifier models with respect to their generalization error, 3) to enable compare performance between various candidate classifier models. A widely used measure of performance, namely accuracy where number of correctly classified samples are determined, was selected (Raschka, 2015). The aim was to evaluate performance of each individual candidate classifier generated at each iteration of training using validate set.

However, performance of a classifier may be affected by the bias in partitioning dataset into training set and validate set. Practically, in k-fold cross-validation one fold is set aside as a validation set and whose choice may affect performance estimate of the candidate model. To ensure an estimate performance that is less sensitive to partitioning and choice of validation set effect, repeated k-fold is recommended (Raschka, 2015). As a result of activity 1a (splitting dataset), the current work adopted repeated 5-fold cross-validation where each fold was used as a validation set alternately thus amounting to five iterations. The classification accuracy for each fold used as a validation set in each iteration were then used to calculate the average performance of each candidate model (Raschka, 2015).

Activity 3a: Select the Best Model

The question we should ask is, how do we know which model performs well on the final test dataset and real world data? Each classification algorithm is based on certain assumptions which may differ from algorithm to algorithm in terms of number of features or samples, amount of noise in the dataset, and whether the target classes are linearly separable or not (Raschka, 2015). Further, each algorithm may have different classifiers depending on its different configurations (Lavesson, 2006).

An experimental comparison between classifiers of the selected algorithm, was conducted.

Activity 3b: Evaluate Model with Test set

To determine whether the model would perform well in the real world data, a test set, that had not been seen by the model before was adopted.

3.5.5. Model Evaluation

Model evaluation was conducted to establish generalization suitability and validity of the model. In the present study, experimental approach was adopted for evaluation where two questions guided the process: 1) what is the performance of the model in mapping graduates' skills to industry roles? 2) How do we ensure the validity of the results? Answers from these experimental questions enabled the researcher to provide answers to the last research question: how do we evaluate performance and validity of the mapping model?

To investigate this question a research hypothesis was defined as follows:

H_{o4A}: There is no significant performance difference of the model in different industry domains

The findings from a number of experimental trials helped to investigate the above hypothesis. Sokolova & Lapalme (2009) provided source for performance measures for evaluating classifier models where apart from accuracy and miscalculation errors, precision, recall, and f1-score were adopted. Classification accuracy was preferred in this study because it has been reported widely in many machine learning studies (Raschka, 2015).

However, Raschka (2015) notes that a lot of caution has to be taken because model accuracy is only a useful metric to quantify performance of the model in general. In light of this fact, there was a desire to use performance measures that would provide insight into the quality of the model in terms of committing more serious errors, such as precision, recall, and f-score as elaborated by Sokolova & Lapalme (2009). To achieve this kind of evaluation a prototype software system for mapping graduates' skills to industry roles based on the model was developed. This helped to not only evaluate the model's performance but also compare its performance with other models in literature.

3.6. Summary

This chapter has presented a detailed analysis and design of the research methodology adopted in this study, ranging from research philosophy, research strategies, research designs and methods. The research philosophy was selected based on two philosophical assumptions: epistemology and ontology. Philosophical assumptions helped to locate the philosophical paradigm, positivism, in which the research methodology was placed. A carefully selected approach was used to design a research strategy for each research question before finally deciding on the appropriate research design for each.

The first specific research question was approached using literature review and experimental designs which provided important concepts that formed the basis of data collection and analyses in the second question. The second research question was largely approached using descriptive survey design where data was collected and analyzed to reveal either boundaries between concepts used as target classes or whether the classes are separable as required in machine learning classification. Research questions three and four were both largely approached using experimental design where evaluation of classifier model performance and validity was necessary.

A framework to operationalize the research process was designed where a number of hypotheses were defined. In summary, five research hypotheses were placed at the center of investigation where a concrete research methodology was put into action to provide proof to either accept or reject the hypotheses. Table 3.6 presents a summary of how research was operationalized.

Table 3.6: Operationalization of research methodology

Research Question	Research hypothesis	Research methodology
RQ1: What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?	H_{01A}: All features are equally relevant for inducing better performance to the classifier model	Literature Review/Analysis Experimental Design
RQ2: What is the structural characteristic of concepts that correctly reflect the hierarchy of industry roles required as target classes for machine learning?	H_{02A}: There is no significant boundary differences between industry roles/potential target classes	Descriptive Survey Design
RQ3: How do we build, using these concepts, an appropriate machine learning model for mapping graduates' skills to hierarchically structured occupational industry roles?	H_{o3B}: Any parameter value induces better performance in the model H_{o3C}:All induction algorithms induce equal generalization performance to the model	Experimental Design
RQ4: How do we evaluate the performance and validity of the machine learning model?	H_{04A}: There is no significant performance difference of the model in different industry domains	Literature Review/Analysis Experimental Design

CHAPTER 4: MODELING RESULTS AND FINDINGS

4.0. Introduction

Data analysis results have been grouped into sections. Section 4.1 presents descriptive analysis results while section 4.2 presents experiments analysis results. Discussion provides interpretation of the results and is presented in section 4.3.

4.1. Descriptive Results and Findings

4.1.1 Population description

Tables 4.1.1a and 4.1.1b describe the demographic characteristics of exam past papers' sample and employees' sample.

Table 4.1.1a: Demographic characteristics of exam past papers sample

Variable	Category	Frequency	Percentage (%)
1. Degree program	BSc. Computer science	15	62.5%
	BSc. IT	9	37.5%
2. Year studied	Second year	4	16.7%
	Third year	10	47.7
	Fourth year	5	20.8%
	Second and third year	5	20.8%
3. Number of questions	Four	5	20.8%
	Five	14	58.3%
	Eight	1	4.2%
	Ten	4	16.7%
4. Total exam marks	90	5	20.8%
	110	14	58.3%
	160	3	12.5%
	170	1	4.2%
	180	1	4.2%

4.1.2. Proportions of job entry industry roles

Figure 4.1.2 presents pie chart results showing common industry roles undertaken by software engineers in the industry at job entry level after graduation and their proportions (%) as revealed by the survey.

Findings #1:

Figure 4.1.2 reveals that while 'web programmer [WP]' (25.66%) and 'analyst programmer [AP]' (23.39%) were very popular at job entry level 'project manager [PM]' (3.54%) was not. The assumption behind this is that experience is highly demanded in this position than the rest and yet this experience may not be available to entry level graduates.

Table 4.1.1b: demographic characteristics of employees' sample

Variable	Category	Frequency	Percentage (%)
1. Gender	Male	77	68.1%
	Female	36	31.9%
2. Bachelor's degree	BSc. Computer science	32	28.3%
	BSc. IT	55	48.7%
	BSc. Software engineering	22	19.5%
	Others	4	3.5%
3. Attractor to job	Passion	31	27.4%
	Salary	33	29.2%
	Ambition	33	29.2%
	Qualification	7	6.2%
	Other	9	8.0%
4. % of classroom learnt content tested in exam	100%	4	3.5%
	75%	73	64.6%
	50%	33	29.2%
	25%	2	1.8%
	0%	1	9.0%

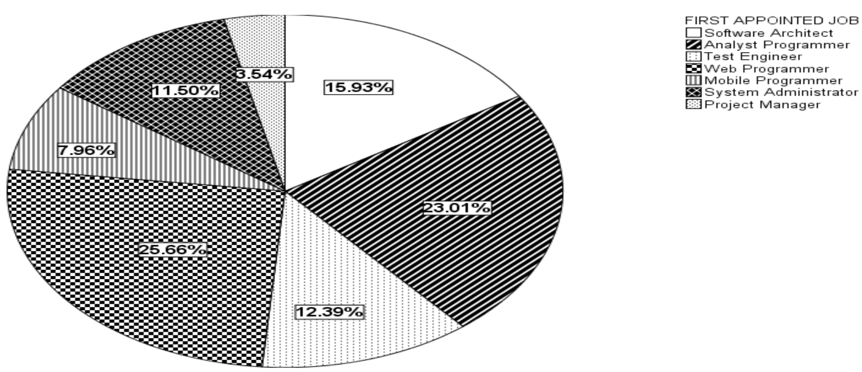


Figure 4.1.2: industry roles for software engineers

4.1.3. Proportions of job entry level role performance activities

Figure 4.1.3 presents a bar graph showing frequency analysis results of a total of 17 role performance activities (RPA) performed by software engineers in various industry roles at job entry level as revealed by the survey. The results reveal RPA 'design data base' is the highest performed (11%) while 'manage project workflows' is the least (2%). Table 4.1.3 presents results showing two types of competences for software engineers as main competences and specialization area for specific competences. Three main competences are 'Design [D]', 'Coding [P]', and 'Manage [M]' and their prevalence proportions (%) indicated for each industry role. The results indicate, for example in 'Software Architecture [SA], design competence is more demanded (prevalence of 50%) than coding (prevalence of 33.2%) and manage (prevalence of 16.8%). Two specialization areas of specific competences are 'Mobile Developer' and 'Desktop Developer' and their proportion numbers in the sample data are indicated for each industry role. The results indicate out of 19 'Software Architecture [SA] for example, 4 have specialized as 'Mobile Developers' and 15 as "Desktop Developers'. The results also indicate that the overall software engineers' demand for 'Coding [P]' is higher (prevalence of 42.72%) than 'Design [D]' (prevalence of 36%) and 'Manage' (prevalence of 21.2.8%).

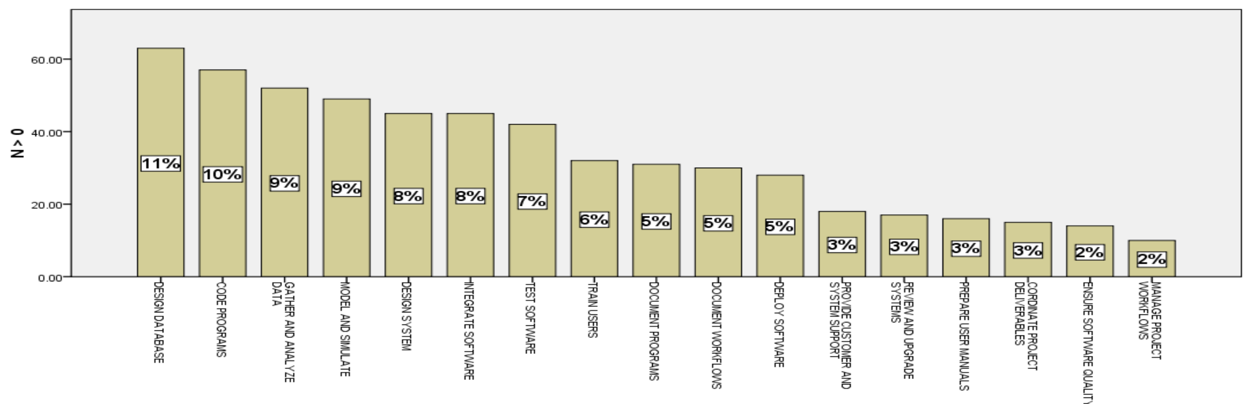


Figure 4.1.3: Role performance activities for software engineers' industry roles

Table 4.1.3: Prevalence of competences in each industry role

	TYPE	SE Industry Roles							TOTAL [Rank]
		SA	AP	TE	WP	MP	SAD	PM	
Main Competence [% prevalence]	D (design)	50.00	22.12	42.16	40.51	34.58	17.61	28.21	36.00 [2]
	P (coding)	33.20	61.06	52.61	40.51	42.36	29.55	34.29	42.72 [1]
	M (manage)	16.80	16.81	5.22	18.99	23.05	52.84	37.50	21.28 [3]
Specific competence [numbers]	Mobile Developers	5	14	8	15	-	1	1	44
	Desktop Developers	14	15	6	16	-	12	6	69

Findings #2:

Figure 4.1.3 and Table 4.1.3 reveal that occupational industry roles have similar job performance activities/competences but different levels of emphasis where some are more emphasized in one role but less emphasized in other roles. Also, software engineers demand more of programming than management skills.

4.1.4. Central tendency measures

Both mean and mode were used to describe the central tendency of the independent variables. However, before further analyses were conducted, reduction of data redundancy using principle component analysis method was performed on the study's data file. Table 4.1.4a presents a rotated component matrix result indicating the uncorrelated factors of the data. A total of 24 original sub-variables for analysis were reduced to 13 components or factors, hence considerably reducing data complexity with little loss of accuracy information of only 13.71%. The 13 components represent 13 sub variables that were used to assess respondents' perception on the four factors that could be used to determine graduates suitability for various industry roles as indicated in the research model's input variables and as described in this section.

Table 4.1.4a: Rotated Component Matrix for principle component analysis

Rotated Component Matrix[a]

	Component												
	1	2	3	4	5	6	7	8	9	10	11	12	13
SOFTWARE REGUIREMENT	**.872**	.076	.125	.136	.034	-.040	-.020	.030	.084	.075	.046	-.140	-.081
SOFTWARE DESIGN	.807	.270	.196	.122	.095	-.074	-.121	-.086	.047	-.146	-.036	.052	.040
SOFTWARE PROCESS	.628	.314	.302	-.029	.219	.088	-.021	-.058	.266	-.086	.000	.151	-.031
CONCEPT UNDERSTANDING	.206	**.869**	.162	.126	.153	.010	-.039	-.102	.062	.019	.012	.076	.035
CONCEPT REMEMBERING	.254	.740	.111	.122	.153	.025	-.010	.030	.427	-.026	-.014	.020	-.017
CONCEPT ANALYSIS	.155	.687	.223	.539	.148	-.052	-.019	-.025	.051	.025	-.085	-.065	-.062
SOFTWARE CONFIGURATION	.263	.129	**.754**	.176	.125	.187	-.113	-.098	.078	-.063	-.111	.090	-.015
SOFTWARE TESTING	.443	.129	.715	.123	.011	-.073	.050	-.023	.146	-.083	.114	-.001	-.047
SOFTWARE MAINTENANCE	.003	.237	.659	.405	.295	-.096	-.066	-.089	-.075	-.037	-.121	-.038	.059
SOFTWARE INFRASTRUCTURE	.090	.146	.617	.086	.615	-.158	-.007	.077	-.038	.026	-.074	-.093	-.017
CONCEPT JUDGEMENT	.048	.380	.290	**.753**	.013	-.085	-.079	-.048	.049	-.037	.006	-.096	.056
CONCEPT MODELING	.127	.065	.159	.742	.188	-.022	-.075	-.153	.157	.019	.133	-.061	.245
SOFTWARE QUALITY	.038	.253	.188	.102	**.832**	.100	-.040	-.055	.110	-.028	.145	-.026	.110
SOFTWARE CONSTRUCTION	.377	.034	.137	.452	.552	-.030	-.103	-.065	.216	-.207	-.155	.052	-.070
SOFTWARE MANAGEMENT	.366	.101	.033	.532	.548	-.043	.023	.072	.040	-.027	-.228	.095	-.165
DATABASE SCORE	-.067	.024	-.028	-.076	.058	**.941**	-.006	.030	.049	.030	.147	.014	.074
OPERATING SYSTEM CORE	.032	-.093	.084	-.047	-.212	.571	.486	.388	-.138	.201	-.140	.089	-.131
SE PROJECT SCORE	-.119	-.034	-.093	-.110	-.020	.026	**.890**	-.005	-.036	.053	.224	.096	.184
NETWORKING	-.057	-.070	-.096	-.125	.000	.084	.019	**.927**	-.023	.017	.150	.132	.099
CONCEPT APPLICATION	.229	.287	.069	.175	.118	.016	-.065	-.043	**.846**	.025	-.097	-.039	-.059
RESULTS FOR OLEVEL	-.075	.012	-.091	-.028	-.059	.075	.074	.027	.016	**.938**	.102	.085	.196
PROGRAMMING SKILLS	.028	-.030	-.081	.028	.011	.128	.198	.153	-.086	.111	**.861**	.174	-.088
DISTRIBUTED SYSTEMS SCORE	-.019	.046	.019	-.081	-.016	.034	.099	.133	-.023	.082	.150	**.941**	.035
RESULTS FOR BACHELORS	-.068	-.004	-.016	.147	.027	.046	.153	.095	-.056	.204	-.084	.038	**.902**

Extraction Method: Principal Component Analysis.
Rotation Method: Varimax with Kaiser Normalization.

a. Rotation converged in 18 iterations.

Table 4.1.4b presents summarized results showing the calculated index vector for each industry role where Mn, Mx, Iv, and I_R represent minimum index value, maximum index value, average index value, and relative index value.

Table 4.1.4b: Class boundaries for various industry roles

Role category name	Content Knowledge (Relevance Index)				Cognitive skills (Durability Index)				Technical skills (Accuracy Index)				Academic capacity (Capacity Index)			
	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R	Mn	Mx	I_V	I_R
Project Manager (PM)	8.44	8.51	**8.5**	**3**	9.06	9.51	**9.5**	**2**	0	9.53	**9.525**	**7**	8.84	above	**9**	**1**
Mobile Programmer (MP)	8.06	8.08	**8.074**	**6**	9.51	above	**9.815**	**1**	10.01	10.03	**10.022**	**3**	8.77	8.84	**8.833**	**2**
System Administrator (SAD)	8.58	above	**8.718**	**1**	8.99	9.06	**9.051**	**3**	10.06	above	**10.342**	**1**	8.26	8.77	**8.769**	**3**
Test Engineer (TE)	8.08	8.43	**8.429**	**5**	less	8.01	**8**	**7**	9.88	10.01	**10**	**4**	8.07	8.26	**8.25**	**4**
Web Programmer (WP)	less	8.06	**8.057**	**7**	8.01	8.25	**8.241**	**6**	9.55	9.88	**9.876**	**5**	7.56	8.07	**8.069**	**5**
Analyst Programmer (AP)	8.43	8.44	**8.436**	**4**	8.25	8.49	**8.487**	**5**	10.03	10.06	**10.058**	**2**	7.09	7.56	**7.558**	**6**
Software Architect (SA)	8.51	8.58	**8.574**	**2**	8.49	8.99	**8.981**	**4**	9.53	9.55	**9.545**	**6**	0	7.09	**7.083**	**7**

Independent Variable1 – Relevant content knowledge that promotes enhanced performance in the industry role.

Out of the original 10 sub-variables only three are uncorrelated i.e. 1) software requirement 2) software configuration, and 3) software quality. Figures 4.1.4a, 4.1.4b, and 4.1.4c present bar graph results showing comparison of average content required of various knowledge areas to perform each industry role. Mode has been used as the measure of central tendency and the results reveal knowledge content type 'software requirements' and 'software quality' are least relevant to 'analyst programmer' while 'software configuration' is least relevant to 'project manager'. However, 'software requirements' and 'software configuration' are highly relevant to 'systems administrator' while 'software quality' is most relevant to 'test engineer'.

Finally, the content knowledge index has been calculated by getting the average of the three sub-variables and the mean has been used as the measure of central tendency. Figure 4.1.4d presents bar graph results showing comparison of the means for the content knowledge index of the various industry roles. Y axis of this figure represents the average of the three subvariables referred in this section and denoted as meanR. The results indicate ‘systems administrator’ has the highest content knowledge index (8.718) while ‘web programmer’ has the least content knowledge index (8.057).

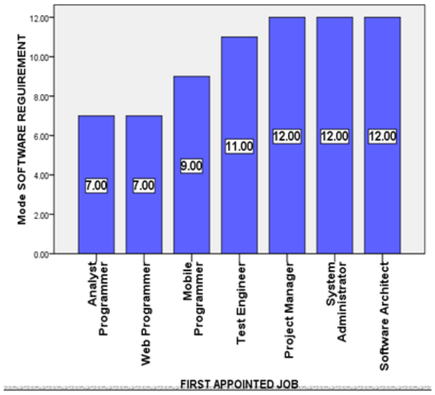


Figure 4.1.4a: Average software requirements knowledge content required for each industry role

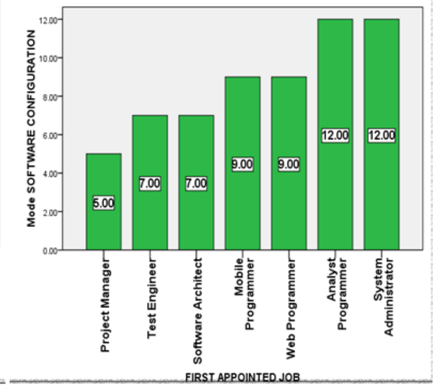


Figure 4.1.4b: Average software configuration knowledge content required for each industry role.

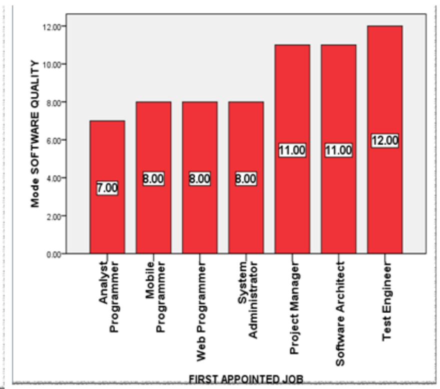


Figure 4.1.4c: Average software quality knowledge content required for each industry role

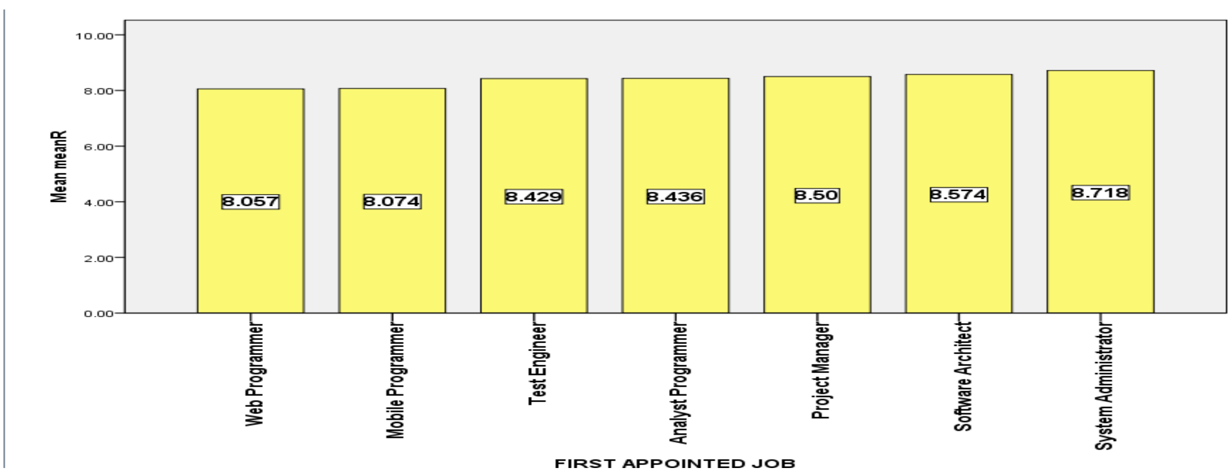


Figure 4.1.4d: Average Content knowledge index for each industry role

Independent Variable2 – Cognitive skills that promote prolonged retention of relevant knowledge required to perform the industry role.

Out of the original 6 sub-variables only three are uncorrelated i.e. 1) concept understanding 2) concept application, and 3) concept judgment. Figure 4.1.4e, 4.1.4f, and 4.1.4g present bar graph results showing comparison of average level required of various types of cognitive skills to perform each industry role. Again, mode has been used as the measure of central tendency and results indicate industry role ‘analyst programmer’ demands highest levels of skill type ‘concept understanding’ and ‘concept application’, while ‘test engineer’ and ‘project manager’ demand levels for these skill types are the lowest.

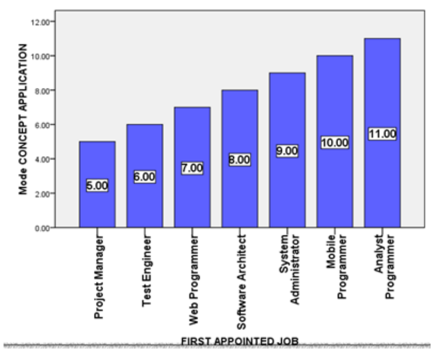


Figure 4.1.4e: concept application skill required for each industry role

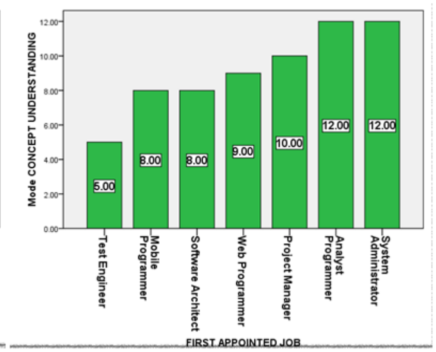


Figure 4.1.4f: concept understanding skill required for each industry role

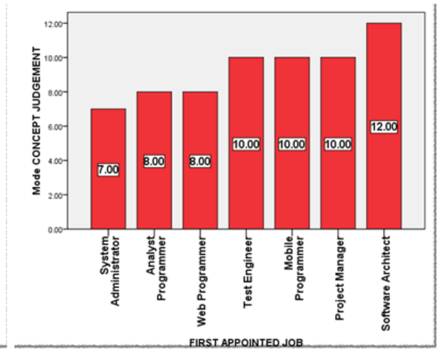


Figure 4.1.4g: concept judgment skill required for each industry role

However, ‘concept judgment’ demand levels are very high for ‘software architect’ and very low to ‘systems administrator’. Finally, the cognitive skills index has been calculated by getting the average of the three sub-variables and the mean has been used as the measure of central tendency. Figure 4.1.4h presents bar graph results showing comparison of the means for the cognitive index of the various industry roles. Y axis of this figure represents the average of the three sub-variables referred in this section and denoted as meanD. The results indicate ‘mobile programmer’ have the highest cognitive skills index (9.815) while ‘test engineer’ have the least cognitive skills index (8.0).

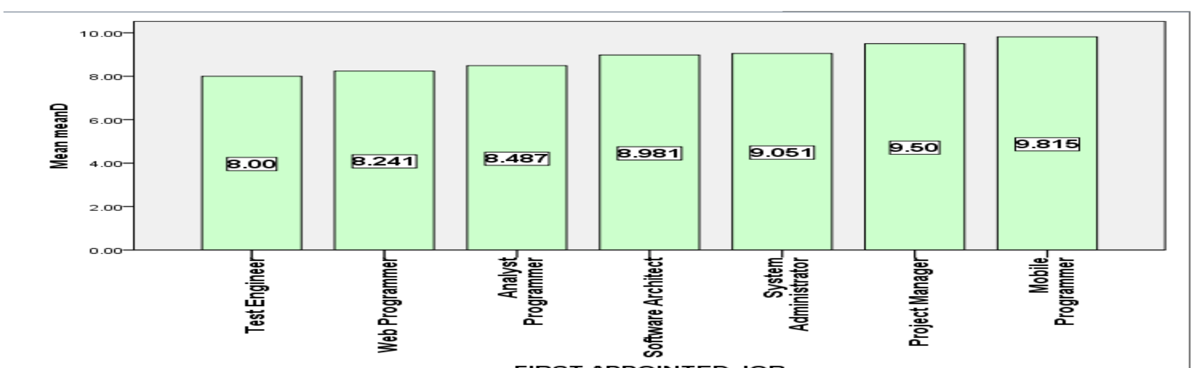


Figure 4.1.4h: Average cognitive skills index for each industry role

Independent Variable3 – Technical skills that promote precision of performance results in the industry role.

Out of the original 6 sub-variables five are uncorrelated i.e. 1) SE project 2) database skills 3) programming skills 4) networking skills, and 5) distributed skills. Figure 4.1.4i presents bar graph results showing comparison of average level required of various types of technical skills to perform each industry role. Again, mode has been used as the measure of central tendency and results indicate industry roles ‘analyst programmer’, ’test engineer’, ‘web programmer’, and ‘mobile programmer’ have similar demand levels of all skill types while the rest reveal some variations. Finally, the technical skills index has been calculated by getting the average of the five sub-variables and the mean has been used as the measure of central tendency. Figure 4.1.4k presents bar graph results showing comparison of the means for the technical skills index of the various industry roles. The results indicate ‘systems administrator’ has the highest technical skills index (10.342) while ‘project manager’ has the least technical skills index (9.525).

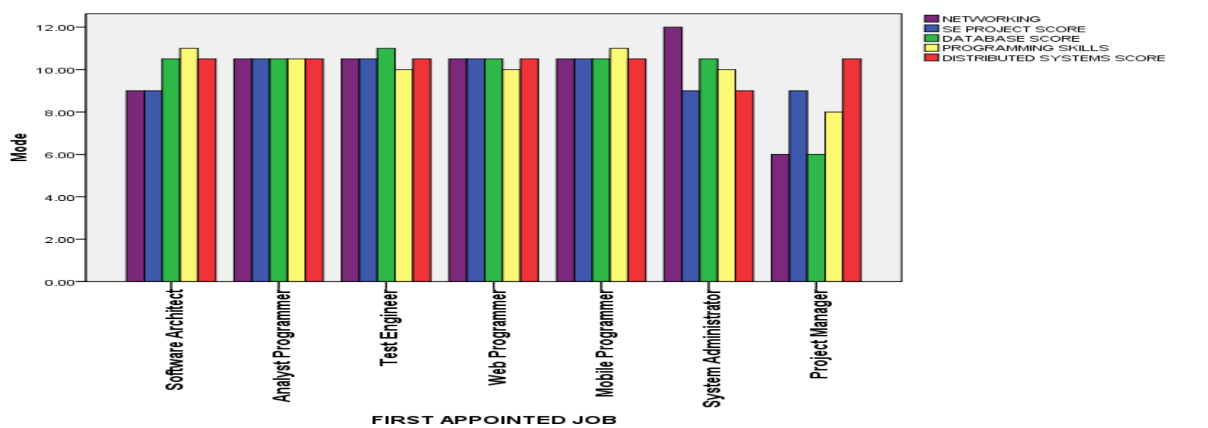


Figure 4.1.4i: Average Technical skills required to perform each industry role

Independent Variable4 – Intellectual content that promotes capacity to perform the industry role

All the two original sub-variables are uncorrelated i.e. ‘O’ level Aggregate points and Bachelors final grade. Figure 4.1.4j presents bar graph results showing comparison of average level required of various types of intellectual content to perform each industry role. Again, mode has been used as the measure of central tendency and results indicate only industry roles ’test engineer’ and ‘web programmer’ have their content type values paired different while the rest reveal their pairs are tying. However, it is important to note that there are two blocks of ties, lower and upper. Industry roles ‘software architect’ and ‘analyst programmer’ have the lowest similar tie, while ‘project manager ’,’ systems administrator and ’mobile programmer’ have the highest similar tie.

Finally, the academic capacity index has been calculated by getting the average of the paired sub-variables and the mean has been used as the measure of central tendency. Figure 4.1.4l presents bar graph results showing comparison of the means for the academic capacity index of the various industry roles. The results indicate ‘project manager’ have the highest academic capacity index (9.0) while ‘software architect’ have the least academic capacity index (7.083).

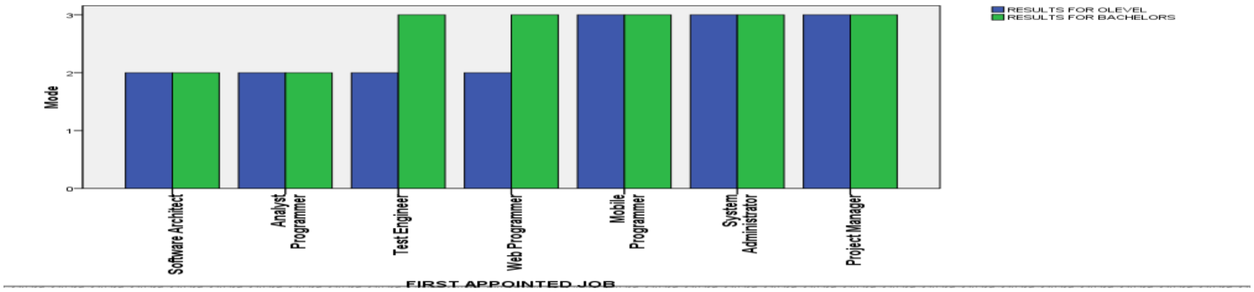


Figure 4.1.4j: Average Intellectual capacity required to perform each industry role

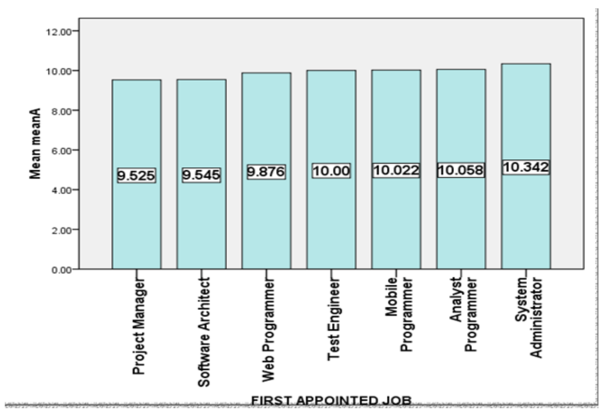


Figure 4.1.4k: Average Technical skills Index required for each industry role

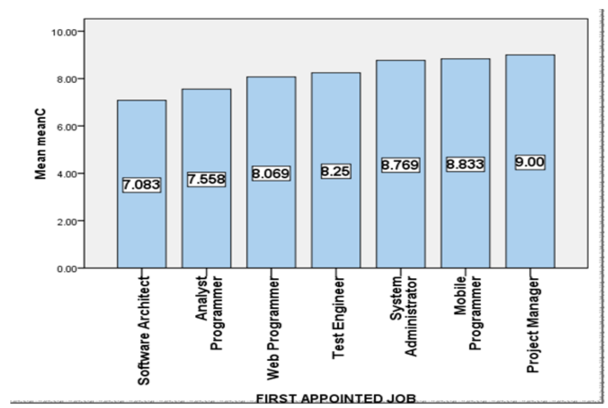


Figure 4.1.4l: Average Academic capacity Index required for each industry role

4.1.5. Hypothesis Testing Results

Table 4.1.5a: presents results of validity test to data that indicates normality of data and homogeneity of group variance in the data. Two types of data (actual values based data and factor values based data) have been scrutinized for validity before they could be adopted in subsequent analysis. The findings in Table 4.1.5a reveal while all the variables of factor based data pass the test for homogeneity of variance, most test variables of actual data do not pass the test. Moreover, both types of data do not meet all the three conditions of normality. Therefore, the tests in this section were conducted with the later data type. Table 4.1.5b: presents results of non-parametric test for multiple independent samples that have been conducted using factor values derived during data redundancy process, to test the research hypotheses.

Table 4.1.5a: Tests of data validity

Type of validity	**Type of test**	**Type of data**	**content knowledge**	**Cognitive skills**	**Technical skills**	**Academic capacity**
1.Homogeneity of group variances Accept if >0.1	**(Levene test)** (Equality of variances) **Hypothesis:** Are variances between the groups equal?	**Actual**	0.172	0.054	0.804	0.077
			yes	no	yes	no
		Factors	0.364	0.265	0.432	0.159
			yes	yes	yes	yes
2.Normality of data Accept if difference not more than 1	(mean ≈ trimmed mean≈ median) **Hypothesis:** **Which groups test positive?**	**Actual**	all	all	all	all
		Factors	all	all	all	all
Accept if when rounded is 0	(Skewness ≈ kurtosis ≈ 0) **Hypothesis:** **Which groups test positive?**	**Actual**	SAD	none	TE, AP	WP
		Factors	none	WP,TE	none	none
Accept if greater than 0.05	(kolmogorov-smirnov test) **Hypothesis:** **Is this data a good fit to a normal distribution?**	**Actual**	0.146	0.607	0.150	0.003
			yes	yes	yes	no
		Factors	0.995	0.466	0.966	0.903
			yes	yes	yes	yes

Table 4.1.5b presents significance test results for the first set of four hypotheses defined in the research design as given chapter3 section 3.5.2) and restated below:

Hypothesis 1(H_{01}):

H_0: There are no significant domain specific knowledge differences between industry roles in the same occupation

H_a: There are significant domain specific knowledge differences between industry roles in the same occupation

Hypothesis 2(H_{02}):

H_0: There are no significant domain general knowledge differences between industry roles in the same occupation

H_a: There are significant domain general knowledge differences between industry roles in the same occupation

Hypothesis 3(H_{03}):

H_0: There are no significant domain specific skill differences between industry roles in the same occupation

H_a: There are significant domain specific skill differences between industry roles in the same occupation

Hypothesis 4(H_{04}):

H_0: There are no significant domain general skill differences between industry roles in the same occupation

H_a: There are significant domain general skill differences between industry roles in the same occupation

Table 4.1.5b: Tests[b] of hypotheses results

	Hypothesis 1	Hypothesis 2	Hypothesis 3	Hypothesis 4
N	109	109	109	109
Median	.0279	-.0525	.0464	-.0005
Chi-Square	2.441	16.151	1.866	13.109
df	6	6	6	6
Asymp. Sig.	.875	.013	.932	.041

b. Grouping Variable: FIRST APPOINTED JOB

The results indicate while Hypothesis 1 results (χ^2=2.441, p=0.875) and Hypothesis 3 results (χ^2=1.866, p=0.932) imply we accept the null hypotheses, Hypothesis 2 results (χ^2=16.151, p=0.013) and Hypothesis 4 results (χ^2=13.109, p=0.041) imply we reject the null hypotheses. Table 4.1.5c presents a cross tabulation of the hypothesis testing results.

Table 4.1.5c: Hypothesis decision results

Variable type	KNOWLEDGE	SKILL
DOMAIN SPECIFIC	**Hypothesis 1 =** Accept	**Hypothesis 3 =** Accept
DOMAIN GENERAL	**Hypothesis 2 =** Reject	**Hypothesis 4 =** Reject

Findings #3:

Table 4.1.5c reveals that domain specific knowledge and skills for occupational industry roles were similar while their domain general knowledge and skills were different in each role.

4.1.6. Trend analysis results

Figure 4.1.6a presents bar graph results showing comparison of average content knowledge Index values while Figure 4.1.6b presents bar graph results showing comparison of average cognitive skills index values for various universities in the academia both derived from their exam past papers. Results reveal although ‘KCA’ university has the highest content knowledge index value, its cognitive skills index value is the lowest. While ‘UON’ university has the highest cognitive skills index value, ‘JKUAT’ university has the lowest content knowledge index value.

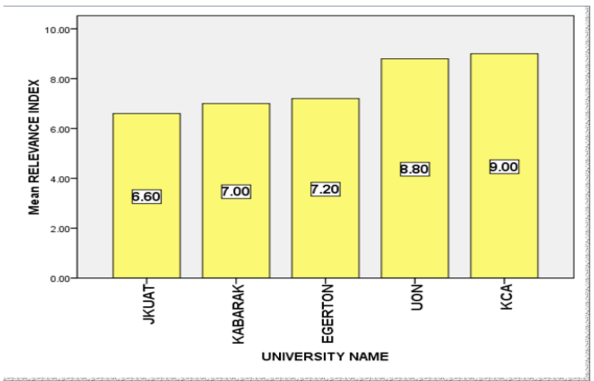


Figure 4.1.6a: Content knowledge Index derived from academia

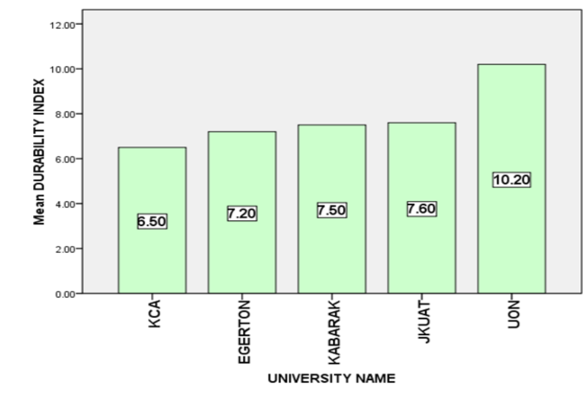


Figure 4.1.6b: Cognitive skills Index derived from academia

Figure 4.1.6c shows box-plot results of the content knowledge index value requirements for various industry roles represented using boxes and content knowledge index values for various universities represented using reference lines. The reference line represents the minimum content knowledge index values expected by various universities. The results reveal that while universities ‘KCA’ and ‘UON’ are trending in all industry roles, ‘JKUAT’ is only trending in only three industry roles i.e. ‘software architect’, ‘mobile programmer’, and ‘project manager’.

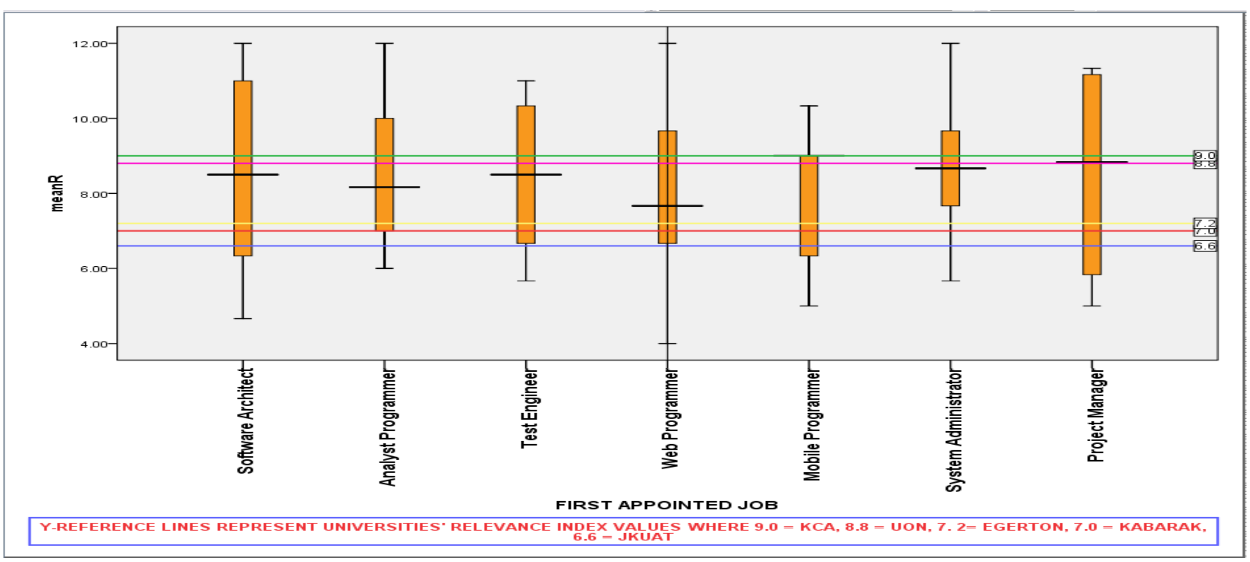


Figure 4.1.6c: Comparison of Average Content Knowledge Index of Academia and Industry roles

Figure 4.1.6d shows box-plot results of the cognitive skills index value requirements for various industry roles and cognitive skills index values for various universities represented using reference lines. The results reveal that only ‘UON’ is trending in all industry roles, while ‘KCA’ and ‘EGERTON’ are only trending in only one and two industry roles respectively i.e. ‘analyst programmer’ for ‘KCA’, while for ‘EGERTON’ are ‘analyst programmer’, and ‘web programmer’.

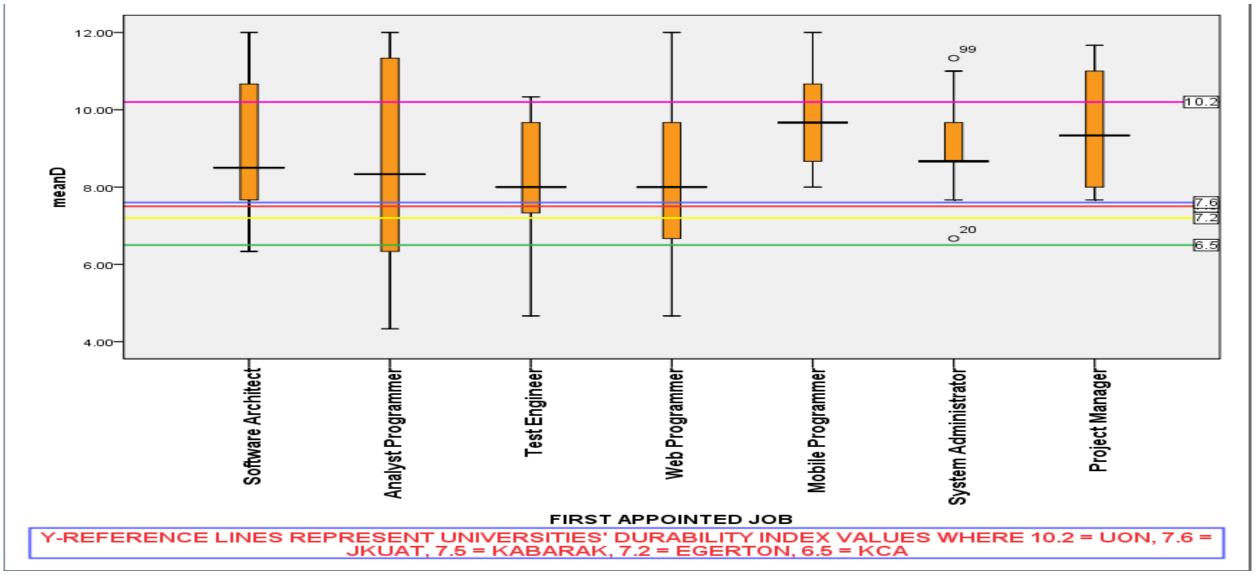


Figure 4.1.6d: Comparison of Average Cognitive Skills Index of Academia and Industry roles

Table 4.1.6 presents a summary of the counts of the trending industry roles in each university as revealed by figure 4.1.6a and 4.1.6b analysis results.

Table 4.1.6: Summary of trending industry roles in the academia

University name	Counts of roles in Content knowledge Index	Counts of roles in Cognitive skills Index	Average counts per university	Percentage (%)
1. UON	7	7	7	100%
2. JKUAT	3	3	3	42.9%
3. Kabarak	6	3	4.5	64.3%
4. Egerton	5	2	3.5	50%
5. KCA	7	1	4	57.1%
Average counts per variable	5.6	3.2		
Percentage (%)	80%	45.7%		62.86%

Findings #4:

Table 4.1.6 reveals that academia was able to meet knowledge requirements of 80% of industry roles while only 45% of industry roles had their skills requirements fulfilled. Academia institutions had different biases towards industry roles’ requirements.

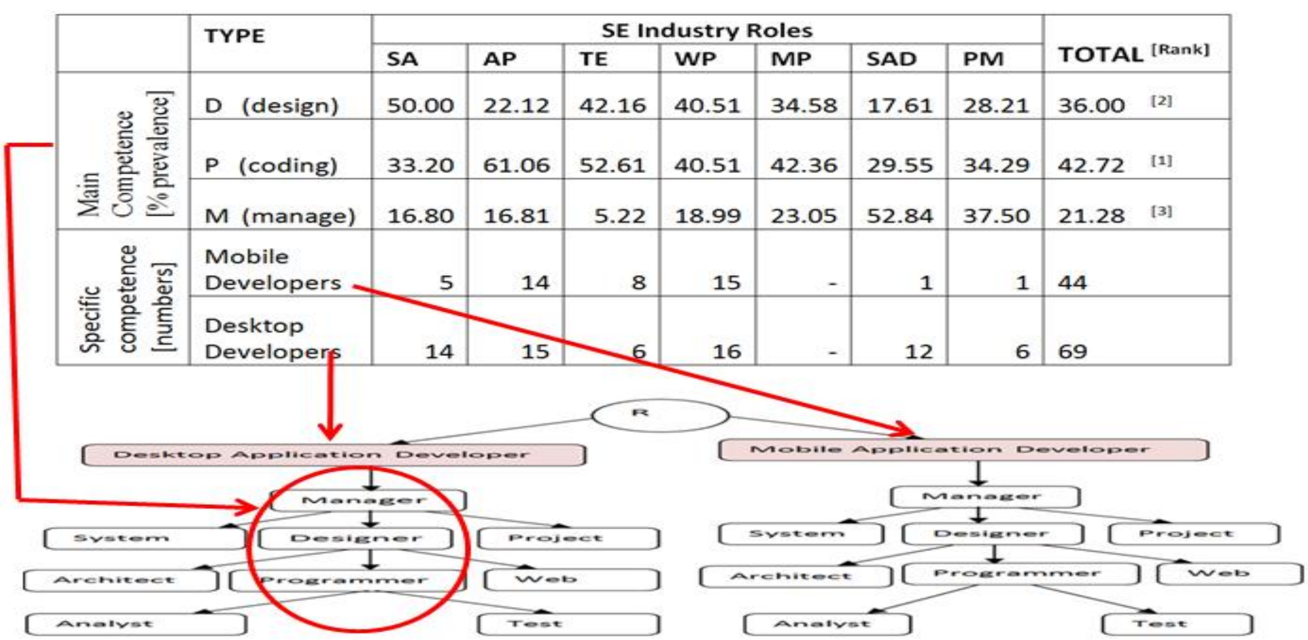
4.2. Experimental Results and Findings for Feature Selection and Algorithm Selection

4.2.1 Introduction

Three types of datasets with known demographic descriptions were used i.e. research (dataset1), benchmark (dataset2), and validation (dataset3). Hypotheses for experimental analyses were tested for significance using either analysis of variance (ANOVA) or paired sample T tests. Significance level of 0.05 was used. Three ML algorithms used for the feature selection experiments were Logistic Regression (LR) whose parameter was (c = 1.0), K-Nearest Neighbor (KNN) whose parameter was (k = 4), and Support Vector Machines (SVM) whose parameters were (kernel='gamma=0.0', C=1.0, random_state=0). The parameters were selected through preliminary trials that produced the best training results. Two ML algorithms used for the algorithm selection experiments were naïve Bayes and SVM whose parameter tuning was explicitly determined.

4.2.2 Taxonomic description of Software Engineers' Industry roles (dataset1)

Figure 4.2.1 illustrates mapping of 12 roles for software engineers from Table 4.1.3 into the proposed taxonomic structure using our method. The 12 roles have been coded as follows: 1: mobile system manager, 2: mobile project manager, 3: mobile architect designer 4: mobile web designer 5: mobile analyst programmer 6: mobile test programmer 7: desktop system manager 8: desktop project manager 9: desktop architect designer 10: desktop web designer 11: desktop analyst programmer 12: desktop test programmer.



	TYPE	SE Industry Roles							TOTAL [Rank]
		SA	AP	TE	WP	MP	SAD	PM	
Main Competence [% prevalence]	D (design)	50.00	22.12	42.16	40.51	34.58	17.61	28.21	36.00 [2]
	P (coding)	33.20	61.06	52.61	40.51	42.36	29.55	34.29	42.72 [1]
	M (manage)	16.80	16.81	5.22	18.99	23.05	52.84	37.50	21.28 [3]
Specific competence [numbers]	Mobile Developers	5	14	8	15	-	1	1	44
	Desktop Developers	14	15	6	16	-	12	6	69

Figure 4.2.1: The Taxonomy for Software Engineers' Industry roles

4.2.3 Taxonomic description of Academic Librarians' Industry roles (dataset3)

Figure 4.2.2 illustrates the mapping of 7 industry roles for academic librarians into the proposed taxonomic structure using our method. The 7 distinct industry roles have been coded as follows: 1: Reference librarian, 2: Circulation librarian, 3: Digital media librarian 4: Multi-service librarian 5: Acquisition & cataloguing librarian 6: Africana librarian 7: Information literacy librarian.

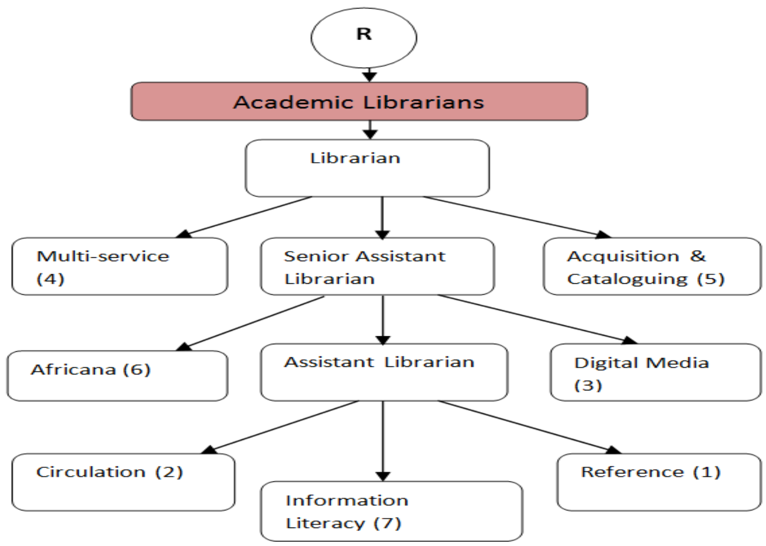


Figure 4.2.2: The Taxonomy for Academic Librarians' roles

Findings #5:

Figures 4.2.1 and 4.2.2 reveals that entry level occupational industry roles were both branched into functional areas and each functional branch was hierarchical with different levels of skills demand (proficiency) and different types of skills at various levels.

4.2.4 Experiment Datasets Description

Table 4.2.4 describes the demographic characteristics of the three datasets that have been used for experimental purpose. Dataset1 represents software engineering employees' profile data while dataset2 represents extract of the AMEO2015 data that has been used as a benchmark and dataset3 represents academic librarians' profile data that has been used for model validation.

Table 4.2.4: Demographic characteristics of experiment datasets

Dataset	Attributes	Instances	Classes	Levels
1. Dataset1	18	113	12	3
2. Dataset2	18	279	12	3
3. Dataset3	14	50	7	3

4.2.5 Class Sizes in the Experiment Datasets

Table 4.2.5 presents table results showing distribution of class instances in the three datasets as revealed by the experiment. While in dataset1 class number 10 has the largest number of instances of 16 and the lowest number of instances in a class is 1, in dataset2 class number 9 has the highest number of instances of 75 and 10 is the lowest number of instances in a class. In dataset3, classes number 1 and 5 have the highest number of instances of 9, 4 is the lowest number of instances in a class.

Table 4.2.5: Distribution of class instances in the datasets

Class-codes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total
Instances (dataset1)	1	1	5	15	14	8	12	6	14	16	15	6	-	-	-	-	-	-	**113**
Instances (dataset2)	9	14	23	40	13	42	9	8	19	32	51	19							**279**
Instances (dataset3)	9	8	7	7	9	4	6	-	-	-	-	-	-	-	-	-	-	-	**50**

Findings #6:

Table 4.2.5 reveals that dataset1 had classes with smaller sizes such as class 1 and class 2 whose sizes were both one. Such class sizes would not be useful for machine learning that required the class instances to be partitioned into training and test set. Therefore such classes were eliminated in the subsequent experiments with this dataset.

4.2.6 Model Building Results and Findings

A total of three experiments were conducted with an overall aim of building the best model. The aims and design elements of the individual experiments have been summarized as shown in Table 4.2.6.1a:

1) Experiment A: To select meaningful features for the model

   Three algorithms (Logistic Regression, K-Nearest Neighbors, and SVM) were used experimental subjects. Out of the features generated by each of the three algorithms, features that appeared in at least two of these algorithms were selected.

2) Experiment B: To select parameter values for the model.

   A range of parameter values was purposively chosen for the algorithm selected in experiment C. Out of the range select a parameter value that renders the model the best performance

3) Experiment C: To select the best model with the smallest generalization error

Two induction algorithms (Naïve Bayes and SVM) were used as experimental subjects. Out of the two induction algorithms used, algorithm that gave the smallest generalization error was selected.

Table 4.2.6.1a presents the planning of the experiments while the detailed results for these experiments have been presented in sections 4.2.6.1, 4.2.6.2, and 4.2.6.3 respectively.

Table 4.2.6.1a: Model Building Experiments' Designs

	Experiment A	**Experiment B**	**Experiment C**
Conception/Objective	To select valuable features for the model	To select relevant parameter values for the model	To estimate generalization error of the model
Design 1.Experimental units 2.Experimental subjects 3.Dependent variable 4.Independent Variable	1. Graduate employees skills 2.ML model's Algorithms 3.Performance (accuracy) 4.Number of features	1. Graduate employees skills 2.ML model's Algorithms 3.Performance (accuracy) 4.Parameter values	1. Graduate employees skills 2.ML model's Algorithms 3.Performance (accuracy) 4.Sample size
Preparation & Execution	1.Split dataset into three: Training set, Validation set, Testing set 2.apply 5-fold cross validation 3.Select features using Sequential backward selection method	1.Split dataset into three: Training set, Validation set, Testing set 2.Apply 5-fold cross validation 3.Apply purposive sampling to values	1.Split dataset into three: Training set, Validation set, Testing set 2.Apply 5-fold cross validation 3.Apply progressive sampling
Analysis	Compare features that give the best accuracy for the model	Compare parameter values that give the best accuracy for the model	Compare generalization performance of the model by the two induction algorithms
Criteria of selection	Out of the features generated by each of the three algorithms, select the one that appears in at least two of these algorithms	Out of a range purposively chosen, select a parameter value that renders the model higher performance	Out of the two induction algorithms used, select algorithm that gives the smallest generalization error

4.2.6.1 Feature Selection using SE Benchmark Dataset (Experiment A)

Initially, benchmark dataset (dataset2) had a total of 13 features excluding the class feature after which feature selection was applied and reduced the features to 5. Initially, features were selected that other evidence, including more general models fitted into the full dataset, suggest would be important predictors of industry roles as applied by Clive & Joan (2000). In the present study, three machine learning algorithms, namely logistic regression (LR), K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) were used for this process. Through sequential backward selection method the three algorithms, namely logistic regression (LR), KNN, and SVM(kernel='gamma', C=1.0, random_state=0) functions were applied on the benchmark dataset (see Figure:

4.2.6.1a,b,&c) and resulted into a range of 4 feature subsets for each of the respective algorithms that gave an optimal performance accuracy (validation =0.80%, test=0.78%) , (validation =0.84%, test=0.71%) and (validation =0.90%, test=0.85%) respectively. Therefore, the best features that gave optimal results to each algorithm as evidenced by Figures 4.2.6.1a,b,&c, in increasing order of importance, were:

Logistic regression = {Age, D, A, C}; KNN = { Age, R, D, A, }; SVC = { R, D, A, C}

```
================FEATURE SELECTION USING  THIS ALGORITHM=========: LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
=================================================================
==DATASET USED WAS===: C:\Program Files
(x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\EMPLOYEES\ARCHIVE\BenchMark_2minusrdac.csv
=================================================================
OVERALL BEST SUBSET: SCORE: (1, 10, 11, 12) 0.836734693878
Test results for all subsets are: [0.63265306122448983, 0.67346938775510201, 0.73469387755102045, 0.77551020408163263,
0.77551020408163263, 0.77551020408163263, 0.79591836734693877, 0.79591836734693877, 0.81632653061224492, 0.83673469387755106,
0.81632653061224492, 0.55102040816326525, 0.26530612244897961]
 best_features: [1, 10, 11, 12]
 Codes and Labels of Relevant_Features: [1, 10, 11, 12]
Index(['AGE', 'D', 'A', 'C'], dtype='object')
Training accuracy with all features: 0.810256410256
Test accuracy with all features: 0.642857142857
Training accuracy with selected features: 0.8
Test accuracy with selected features: 0.785714285714
=============================END===================================

**** End of process output ****
```

Figure 4.2.6.1a: Logistic Regression algorithm run results

```
================FEATURE SELECTION USING  THIS ALGORITHM=========: KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
           metric_params=None, n_neighbors=4, p=2, weights='uniform')
=================================================================
==DATASET USED WAS===: C:\Program Files
(x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\EMPLOYEES\ARCHIVE\BenchMark_2minusrdac.csv
=================================================================
OVERALL BEST SUBSET: SCORE: (1, 9, 10, 11) 0.959183673469
Test results for all subsets are: [0.40816326530612246, 0.51020408163265307, 0.61224489795918369, 0.63265306122448983,
0.7142857142857143, 0.81632653061224492, 0.83673469387755106, 0.89795918367346939, 0.89795918367346939, 0.93877551020408168,
0.95918367346938771, 0.87755102040816324, 0.63265306122448983]
 best_features: [1, 9, 10, 11]
 Codes and Labels of Relevant_Features: [1, 9, 10, 11]
Index(['AGE', 'R', 'D', 'A'], dtype='object')
Training accuracy with all features: 0.661538461538
Test accuracy with all features: 0.404761904762
Training accuracy with selected features: 0.841025641026
Test accuracy with selected features: 0.714285714286
=============================END===================================

**** End of process output ****
```

Figure 4.2.6.1b: K-Nearest Neighbor algorithm run results

```
================FEATURE SELECTION USING  THIS ALGORITHM=========: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3,
gamma=0.0,
  kernel='linear', max_iter=-1, probability=False, random_state=0,
  shrinking=True, tol=0.001, verbose=False)
=================================================================
==DATASET USED WAS===: C:\Program Files
(x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\EMPLOYEES\ARCHIVE\BenchMark_2minusrdac.csv
=================================================================
OVERALL BEST SUBSET: SCORE: (9, 10, 11, 12) 0.959183673469
Test results for all subsets are: [0.81632653061224492, 0.87755102040816324, 0.91836734693877553, 0.93877551020408168,
0.93877551020408168, 0.93877551020408168, 0.93877551020408168, 0.93877551020408168, 0.93877551020408168, 0.95918367346938771,
0.87755102040816324, 0.79591836734693877, 0.59183673469387754]
 best_features: [9, 10, 11, 12]
 Codes and Labels of Relevant_Features: [9, 10, 11, 12]
Index(['R', 'D', 'A', 'C'], dtype='object')
Training accuracy with all features: 0.928205128205
Test accuracy with all features: 0.738095238095
Training accuracy with selected features: 0.902564102564
Test accuracy with selected features: 0.857142857143
=============================END===================================

**** End of process output ****
```

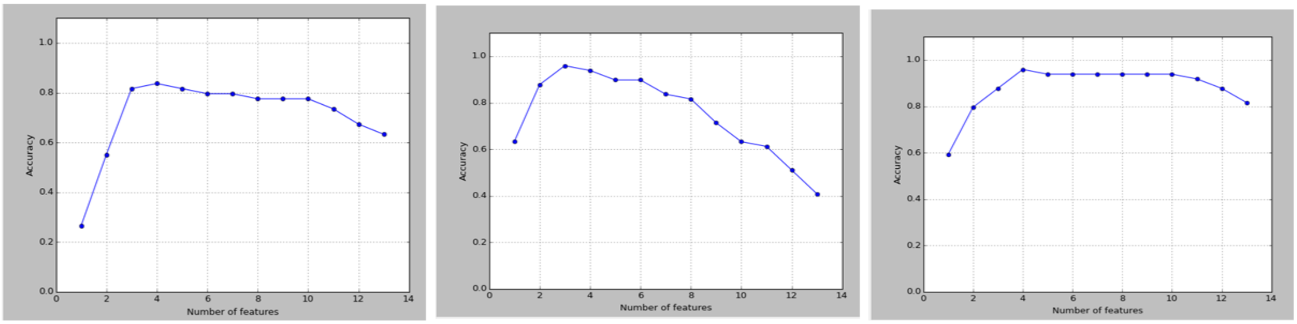
Figure 4.2.6.1c: SVM algorithm run results

In the present study, comparison was conducted and features that were popular in at least two algorithms were selected as true candidates for the best features while the rest were marked as false. Table 4.2.6.1b presents results of comparative analysis of features' subsets for the three algorithms where Y (yes) was used to mark a feature selected by an algorithm, otherwise a dash (-). A true/false score was used to analyze the features along the columns where a feature with at least two Ys was scored true otherwise false.

Those algorithms whose features had been scored false, hence marked for removal, were further analyzed to study performance impact of removing each feature both in isolation and in combination. Caution was taken to ensure core features of the model were carefully removed and analysis was conducted on the impact of adding a feature in other algorithms where it was not selected, especially the core features marked for removal. Popular features that did not exist in other algorithms, were added unconditionally into the subsets of these algorithms. In the present study, all three algorithms were affected through adding popular features, namely LR (feature 'R'), KNN (feature 'C') and SVM (feature 'age'). The overall impact in performance for removing or adding new features was determined.

For logistic regression (LR), the impact of adding 'R' was a loss in performance of -0.01 (0.78 to 0.77). For KNN, the impact of adding 'C' was a gain in performance of +0.12 (0.71-0.83) . For SVC, the impact of adding 'age' was 0.00 (0.85-0.85) . In conclusion, the addition of these popular features would result to a total gain in performance of +0.11 as shown in Table 4.2.6.1b. As a result, a total of five features from the original 13 were selected as optimal features for further analyses, namely: age, R (relevant knowledge), D (cognitive skills), A (technical skills), C (capacity). Table 4.2.6.1b shows cross analysis of features selected by the three algorithms.

Figure 4.2.6.1d shows general performance behavior of each algorithm when fitted with the selected four feature dataset while Figure 4.2.6.1e shows general performance behavior of our model under each induction algorithm when fitted with the all features dataset where the result seem to be consistent with previous observations.



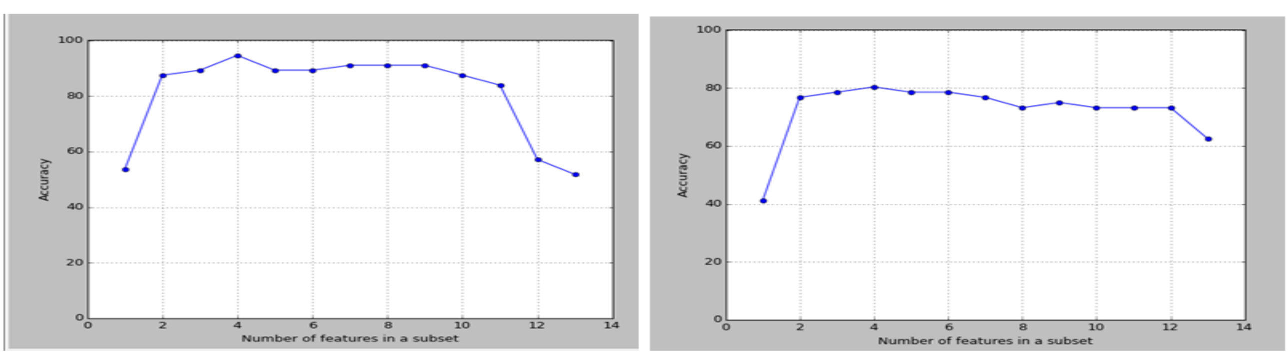
a) Logistic regression b) KNN (neighbors = 4) c) SVC(kernel='linear',c=1.0)

Figure 4.2.6.1d: Sequential Backward Selection of features (LR, KNN, SVM) in SE benchmark dataset.

Table 4.2.6.1b: Analysis of relevant features in SE benchmark dataset

	1,Age	2.R (Relevant content)	3.D (Cognitive)skills	4.A (Technical skills)	5..C (Academic capacity)	**6.Gender**	**7.Bachelors degree**	**8.University of study**	**9.'O' level results**	**10.Bachelors results**	**11.'O' level grading system**	**12.Degree grading system**	**13.location of 'O' Level**	**Accuracy with old features (O)**	**Accuracy with new true features(S)**	**Difference (S – O)**
LR	Y	-	Y	Y	Y	-	-	-	-	-	-	-	-	0.78	0.77	-0.01
KNN	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	0.71	0.83	+0.12
SVM	-	Y	Y	Y	Y	-	-	-	-	-	-	-	-	0.85	0.85	0.00
	True	True	True	True	True	**false**	**false**	**false**	**false**	**false**	**false**	**false**	**false**	OVERALL LOSS		0.11

Further experiments were conducted using our model on all, and 5 features and the results were as shown in Table 4.2.6.1c. Further analysis was conducted to test whether model's performance difference was significant.



a) Naïve Bayes model's feature selection b) SVM model's feature selection

Figure 4.2.6.1e: Selection of features using our model in SE benchmark dataset.

Table 4.2.6.1c: Model performance with all and only selected features in SE benchmark dataset

	Validation Test (naïve Bayes) %		*Validation Test (SVM) %*	
	All features (t_a)	***Selected features (t_s)***	***All features (t_a)***	***Selected features (t_s)***
Fold1	36.59	85.37	75.61	85.37
Fold2	51.43	74.29	74.29	88.57
Fold3	38.24	79.41	85.29	88.24
Fold4	40.63	75.00	87.50	87.5
Fold5	53.33	80.00	93.33	93.33
Mean	**44.04**	**78.81**	**83.20**	**88.60**

Testing whether the difference of group means (folds) was significant using ANOVA:

Table 4.2.6.1c presents validation test results showing a trade-off between model's performance with all features and selected features both under naïve Bayes and SVM based constructs of the model. Two groups were defined, namely all features' and selected features' groups. The results reveal a possible difference between the two scenarios under both constructs of the model. The mean difference under naïve Bayes construct of the model was 34.77 (78.81-44.04) while under SVM was 5.4 (88.60-83.20). To be sure the difference was not due to any other factor but only difference in number of features, ANOVA test was conducted to rule out the effect of group(fold) to group (fold). For this type of test to be valid, conditions for ANOVA that must be satisfied, homogeneity of group variance and normality of data, were checked.

Table 4.2.6.1d presents results for ANOVA analysis for both kinds of model constructs investigated through 10 trials of 5-fold cross-validation experiments. The results indicate the feature sets variances were equal for naïve Bayes based model while not equal for SVM based model and, in fact, means of the two feature sets scores were different in either case and, therefore, the seemingly difference between the two models in Table 4.2.6.1c was real, was due to effect of variation of feature set. For SVM based model Welch and Brown-Forsythe values are 0.000 for both.

Table 4.2.6.1d: ANOVA results (effect of feature selection) in SE benchmark dataset

Type of validity	**Type of test**	**Model**	**p-value**	**Decision**
1.Homogeneity of group variances Accept if p>0.1	**(Levene test -** Equality of variances) **Hypothesis:** Are variances between the groups equal?	**naiveBayes**	0.250	*ACCEPT*
		SVM	0.021	*REJECT*
2.Difference of group means Accept if p>0.05	(**F test** - Equality of group means) **Hypothesis:** Are group means equal?	**naiveBayes**	0.000	*REJECT*
		SVM	0.000	*REJECT*

Findings #7:

Table 4.2.6.1d reveals that reduction of features improved the performance of our model. The change in performance was significant. Slightly better performance could be achieved with fewer features, hence reducing the computational demand in terms of time and computational power. For this dataset, out of 13 features only 5 features produced optimal results, namely Age, R, D, A, C.

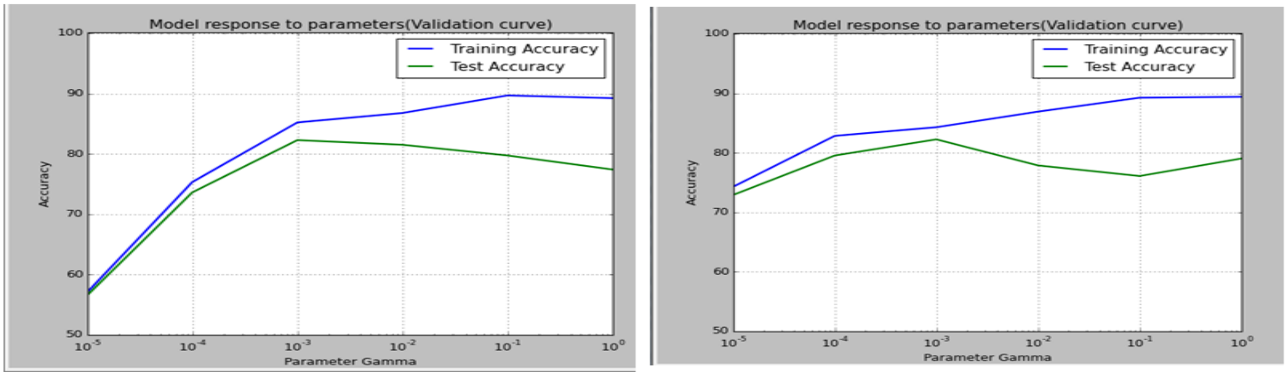
4.2.6.2 Selecting Parameter values using SE BenchMark Dataset (Experiment B)

Table 4.2.6.2a presents results of our model performance under various combination of gamma and complexity parameter, while kernel parameter was held constant at value equal 'Gaussian'. The findings reveal that the model was optimal at gamma value at most 0.01 and complexity value at least 100. Gamma parameter was varied at intervals of 10^n in the range of n (-5 to 0) while complexity was varied at intervals of 10^n in the range of n (-5 to 3). This gave us insight into the relevant values of gamma and complexity for our experiment to select the right values. Figure 4.2.6.2 presents graphical results showing validation curves for the SVM model under various gamma parameter settings in the range of { 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0} for two relevant values of complexity, namely complexity = 1000 and gamma = 10000. These results indicate that our model showed very little improvement under complexity greater than 1000 while the optimal value gamma was 0.001.

Table 4.2.6.2b presents experimental results with various parameter values for gamma and complexity to show a trade-off between relevant and non-relevant parameter values. Further analysis was conducted to determine whether model's performance difference was significant between model with relevant and non-relevant parameter values.

Table 4.2.6.2a: Analysis of relevant parameter values using SE benchmark dataset

	Complexity									
gamma	**0.00001**	**0.0001**	**0.001**	**0.01**	**0.1**	**1**	**10**	**100**	**1000**	**10000**
0.00001	18.3	18.3	18.9	17.6	20.3	18.9	20.2	18.9	44.1	66.7
0.0001	17.6	20.8	19.6	18.9	18.9	18.3	20.8	44.7	66.1	89.5
0.001	18.3	20.2	20.8	17.6	20.3	18.9	45.4	66.6	87.0	87.4
0.01	24.0	24.0	23.4	24.0	24.0	43.0	64.8	87.1	87.6	87.2
0.1	24.0	24.0	24.0	24.0	36.9	65.1	84.7	86.3	84.3	82.7
1	23.4	23.4	23.4	23.4	43.7	79.9	82.9	78.4	79.1	79.2



a) SVC (Complexity = 1000)   b) SVC (Complexity = 10000)

Figure 4.2.6.2: Validation curve for SVM model in SE benchmark dataset

Table 4.2.6.2b: Model performance under various relevant parameter values.

	Non-Relevant gamma and complexity values	**Relevant gamma and complexity values**
	C =0.1, gamma = 0.1	**C=1000, gamma=0.001**
F1	31.71	80.49
F2	31.43	85.71
F3	38.24	82.35
F4	34.38	81.25
F5	36.67	90.0
Mean	**34.48**	**83.96**

Testing whether the difference was significant using ANOVA procedure

Table 4.2.6.2b presents validation test results showing a trade-off between model's performances under various parameter values under SVM based constructs of the model. The focus of this test was between relevant and non-relevant parameter values, hence two groups. The results reveal a possible difference between the two scenarios under this constructs of the model. The mean difference of the model was 49.48 (83.96-34.48). To be sure the difference was not due to any other factor but only difference in parameter values, ANOVA test was conducted. For this type of test to be valid, conditions for ANOVA were checked (homogeneity of group variance and normality of data). Two models, one treated with non-relevant parameter values (gamma = 0.1 and complexity = 0.1) and another with relevant parameter values (gamma = 0.001 and complexity =1000) were used in the investigation.

Table 4.2.6.2c presents results for ANOVA analysis for 10 iterations of 5-fold cross-validation experiments that were conducted to investigate performance change. The results indicate the ,model's performance variances were equal across parameter set values and, in fact, the average model performance scores under the two parameter sets were different. Therefore, the seemingly

Table 4.2.6.2c: ANOVA results (effect of parameter values) in SE benchmark dataset

Type of validity	Type of test	Model	p-value	Decision
1.Homogeneity of group variances Accept if p>0.1	**(Levene test -** Equality of variances) **Hypothesis:** Are variances between the groups equal?			
		SVM	0.760	*ACCEPT*
2.Difference of group means Accept if p>0.05	**(F test** - Equality of group means) **Hypothesis:** Are group means equal?			
		SVM	0.000	*REJECT*

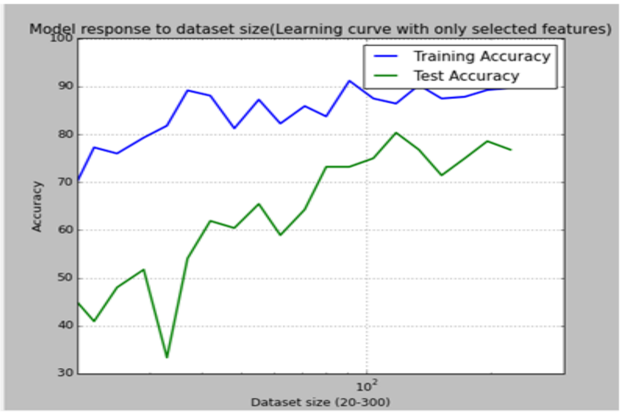
difference between the two models in Table 4.2.6.2b was real, which means it was not due to effect of any other factor but parameter variations in the model.

Findings #8:

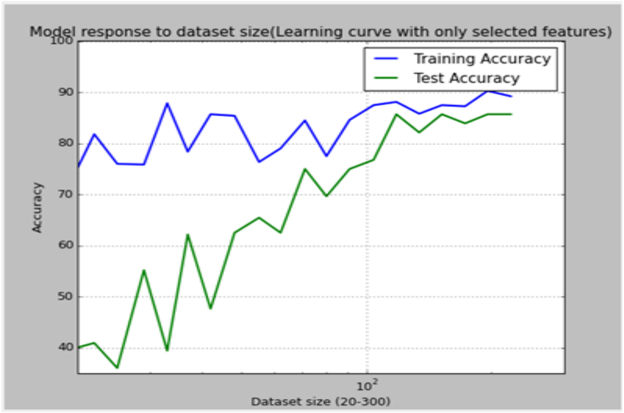
Table 4.2.6.2c reveals that parameter values of SVM improved performance of our model, especially when gamma was at 0.001 and complexity was at least 1000. The change in performance was significant at p=0.05.

4.2.6.3 Estimating generalization error of model using SE Benchmark dataset (Experiment C)

Figure 4.2.6.3a presents graphical results showing learning curves for the two models under various sample sizes starting from sample size of 20. The results reveal that while training and test accuracy



a) Naïve Bayes Model's learning curve



b) SVM Model's learning curve

Figure 4.2.6.3a: Learning Curves for Naïve Bayes and SVM models in SE benchmark dataset

curves for SVM were almost converging as sample sizes increased, for naïve Bayes model the gap between the two curves still remained large. The results also indicate SVM model required a sample size, of about 190 to achieve optimal performance and smaller generalization error, while naive Bayes with sample size less than 120 readily achieved optimal performance. The results also indicate SVM has the smallest generalization error compared to naïve Bayes model at their optimal performance levels.

To investigate this behavior further the two models were experimented under similar conditions then the results were compared. This involved fitting and testing both models with similar training and validate sets respectively through 10 iterations of 5-fold cross-validation. Table 4.2.6.3a presents results of this experiment that indicated there was a difference in mean performance between SVM (78.77) and naïve Bayes (63.93) models which suggested that SVM model was better than naïve Bayes. Further investigation was conducted to test whether the difference (14.84) was real and significant. This test was conducted using paired sample T test procedure.

Table 4.2.6.3a: 10 iterations of 5-fold cross validation tests in SE benchmark dataset

Test fold		***5-Fold cross validation accuracy tests (%)***	
		Naïve Bayes	***SVM***
Fold_1	Mean	60.81	73.30
	N	10	10
	Std. Deviation	3.38	3.65
Fold_2	Mean	63.00	77.78
	N	10	10
	Std. Deviation	3.70	4.21
Fold_3	Mean	66.69	80.18
	N	10	10
	Std. Deviation	5.92	6.04
Fold_4	Mean	63.35	81.90
	N	10	10
	Std. Deviation	5.49	2.63
Fold_5	Mean	65.79	80.70
	N	10	10
	Std. Deviation	6.18	5.81
Total	**Mean**	63.93	78.77
	N	50	50
	Std. Deviation	5.30	5.42

Testing whether the difference was significant using Paired Sample T test procedure

Table 4.2.6.3a presents validation test results showing a trade-off between model's performance under both naïve Bayes and SVM based constructs. The focus of this test was between naïve Bayes and SVM, hence two paired variables. Table 4.2.6.3a indicates a potential difference of 14.84 (78.77-63.93) in the overall mean performance A paired sample T test was conducted to test the hypothesis that model performance difference was not significant. For this type of test to be valid, conditions for tests were checked (homogeneity and normality of data). Table 4.2.6.3b presents results based on 10 iterations of 5-fold cross-validation tests. The results indicate the difference was real and significant.

Table 4.2.6.3b: Paired Sample T Tests for Model selection using SE benchmark dataset

	Pair	**Paired differences**					**t**	**df**	**Sig(2-tailed)**	**RESULT**
		Mean	**Std. dev.**	**Std. error mean**	**95% confidence interval for difference**					
					lower	**upper**				
Paired	naiveBayes-svm	-14.84	6.988	.988	-16.83	-12.86	-15.02	49	.000	*REJECT*

Findings #9:

General performance indicated that SVM model (78.77%) was better than naïve Bayes model (63.93%), this was revealed by cross-validation results in Table 4.2.6.3a. Table 4.2.6.3b confirmed that the performance difference was real and significant at p=0.05.

4.2.6.4 Selecting Parameter values using SE Field Dataset (Experiment B)

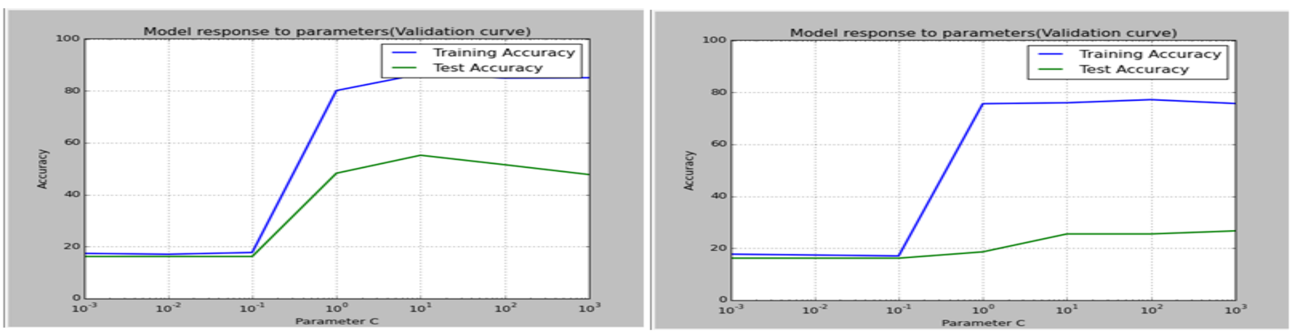
Table 4.2.6.4a presents results of our model performance under various combination of gamma and complexity parameter, while kernel parameter was held constant at value equal 'Gaussian'. The findings reveal that the model was optimal at gamma value at least 0.1 and complexity value at least 10. Gamma parameter was varied at intervals of 10^n in the range of n (-5 to 0) while complexity was varied at intervals of 10^n in the range of n (-5 to 3). This gave us insight into the relevant values of gamma and complexity for our experiment to select the right values.

Figure 4.2.6.4 presents graphical results showing validation curves for the SVM model under various complexity parameter settings in the range of { 0.001, 0.01, 0.1, 1.0, 10.0, 100.0 } for both relevant values of gamma, namely gamma = 0.1 and gamma = 1.0. These results indicate that our model shows better results under gamma = 0.1 than when gamma = 1.0 and we experimented further with all relevant values of

Table 4.2.6.4a: Analysis of relevant parameter values using SE field dataset

	Complexity								
gamma	**0.00001**	**0.0001**	**0.001**	**0.01**	**0.1**	**1**	**10**	**100**	**1000**
0.00001	18.7	18.7	18.7	18.7	18.7	18.7	18.7	18.7	37.8
0.0001	37.8	37.8	37.8	37.8	37.8	37.8	37.8	37.8	55.1
0.001	55.1	55.1	55.1	55.1	55.1	55.1	55.1	55.1	57.8
0.01	57.8	57.8	57.8	57.8	57.8	57.8	60.2	60.4	60.4
0.1	60.4	60.4	60.4	60.4	60.4	60.4	60.4	60.4	60.4
1	60.4	60.4	60.4	60.4	60.4	60.4	60.4	60.4	60.4

complexity parameter where gamma = 0.1. Table 4.2.6.4b presents experimental results with various parameter values for gamma and complexity to show a trade-off between relevant and non-relevant parameter values. Further analysis was conducted to determine whether model's performance difference was significant between model with relevant and non-relevant parameter values.



a) SVC (gamma = 0.1)   b) SVC (gamma = 1.0)

Figure 4.2.6.4: Validation curve for SVM model using SE field dataset

Table 4.2.6.4b: Model performance under various relevant parameter values.

	Non-Relevant(at gamma = 0.0001)	**Relevant Complexity values (at gamma = 0.1)**		
	C =0.0001	**C =10**	**C = 100**	**C = 1000**
F1	11.7	64.7	52.9	58.8
F2	12.5	56.2	68.7	50.0
F3	13.3	53.3	53.3	53.3
F4	18.1	63.6	63.6	54.5
F5	37.5	87.5	62.5	75.0
Mean	**18.6**	**65.0**	**60.2**	**58.3**

Testing whether the difference was significant using ANOVA procedure

Table 4.2.6.4b presents validation test results showing a trade-off between model's performance under various parameter values under SVM based constructs of the model. The focus of this test was between relevant and non-relevant parameter values, hence two groups. The results reveal a possible difference between the two scenarios under this constructs of the model. The mean difference of the model was 46.6 (65.0-18.6) to 39.7 (58.3-18.6). To be sure the difference was not due to any other factor but only difference in parameter values, ANOVA test was conducted. For this type of test to be valid, conditions for ANOVA were checked (homogeneity of group variance and normality of data). Two models, one treated with non-relevant parameter values (gamma = 0.0001 and complexity = 0.0001) and another with relevant parameter values (gamma = 0.1 and complexity =10) were used in the investigation.

Table 4.2.6.4c presents results for ANOVA analysis for 10 iterations of 5-fold cross-validation experiments that were conducted to investigate performance change. The results indicate the ,model's performance variances are equal across parameter set values and, in fact, the average model performance scores under the two parameter sets are different. Therefore, the seemingly difference between the two models in Table 4.2.6.4b was real, was not due to effect of any other factor but parameter variations in the models.

Table 4.2.6.4c: ANOVA results (effect of parameter values) in SE field data

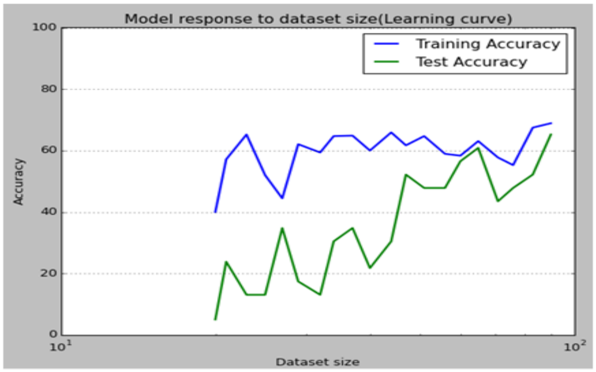
Type of validity	Type of test	Model	p-value	Decision
1.Homogeneity of group variances Accept if p>0.1	**(Levene test -** Equality of variances) **Hypothesis:** Are variances between the groups equal?	**SVM**	0.673	*ACCEPT*
2.Difference of group means Accept if p>0.05	(**F test** - Equality of group means) **Hypothesis:** Are group means equal?	**SVM**	0.000	*REJECT*

Findings #10:

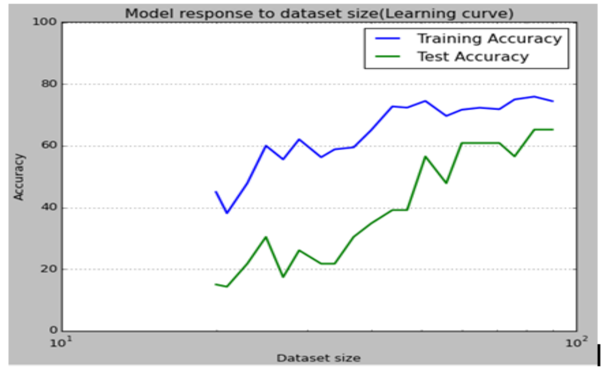
Table 4.2.6.4c reveals that parameter values of SVM improved performance of our model, especially when gamma was at least 0.1 and complexity was at least 10. The change in performance was significant at p=0.05.

4.2.6.5 Estimation of generalization error of the model using SE Field dataset (Experiment C)

Figure 4.2.6.5a presents graphical results showing learning curves for the two models under various sample sizes starting from sample size of 20. The results reveal that while training and test accuracy curves for naïve Bayes were almost converging as sample sizes increased, for SVM model the gap between the two curves still remained large. The results also indicate SVM model required a bigger sample size, i.e. greater than 100, to achieve optimal performance and smaller generalization error, while naive Bayes with sample size less than 100 readily achieved optimal performance.



naïve Bayes model's learning curve



SVM-model's learning curve

Figure 4.2.6.5a: Learning Curves for Naïve Bayes and SVM models in SE field data

To investigate this behavior further, the two models were experimented under similar conditions then the results were compared. This involved fitting and testing both models with similar training and validate sets respectively through 10 iterations of 5-fold cross-validation. Table 4.2.6.5a presents results of the experiment that indicate there was a difference in mean performance between the two,

Table 4.2.6.5a: 10 iterations of 5-fold cross validation tests in SE field dataset

Test fold		***5-Fold cross validation accuracy tests (%)***	
		Naïve Bayes	***SVM***
Fold_1	Mean	49.51	55.86
	N	10	10
	Std. Deviation	7.54	9.59
Fold_2	Mean	48.89	59.41
	N	10	10
	Std. Deviation	11.37	4.99
Fold_3	Mean	56.22	58.09
	N	10	10
	Std. Deviation	7.22	10.65
Fold_4	Mean	51.22	53.11
	N	10	10
	Std. Deviation	10.54	8.46
Fold_5	Mean	56.84	57.48
	N	10	10
	Std. Deviation	13.97	12.70
Total	**Mean**	**52.54**	**56.79**
	N	**50**	**50**
	Std. Deviation	**10.56**	**9.48**

SVM (56.7) and naïve Bayes (52.5) models, which suggested that SVM model was better than naïve Bayes. Further investigation was conducted to test whether the difference was real and significant. This test was conducted using paired sample T test procedure.

Testing whether the difference was significant using Paired Sample T test procedure

Table 4.2.6.5a presents validation test results showing a trade-off between model's performance under both naïve Bayes and SVM based constructs. The focus of this test was between naïve Bayes and SVM, hence two paired variables. Table 4.2.6.5a indicates a potential difference of 4.25 (56.79-52.54) in the overall mean performance A paired sample T test was conducted to test the hypothesis that model performance difference was not significant. For this type of test to be valid, conditions for tests were checked (homogeneity and normality of data). Table 4.2.6.5b presents results based on 10 iterations of 5-fold cross-validation tests. The results indicate the difference was real and significant.

Table 4.2.6.5b: Paired Sample T Tests for Model selection using SE field dataset

	Pair	**Paired differences**					**t**	**df**	**Sig(2-tailed)**	**RESULT**
		Mean	**Std. dev.**	**Std. error mean**	**95% confidence interval for difference**					
					lower	**upper**				
Paired	naiveBayes-svm	-4.254	14.39	2.03	-8.34	-.163	-2.09	49	.042	*REJECT*

Findings #11:

General performance indicated that SVM model (56.79%) was better than naïve Bayes model (52.54%), this was revealed by cross-validation results in Table 4.2.6.5a. Table 4.2.6.5b confirmed that the performance difference was real and significant at p=0.05.

4.3. Discussion of Modeling Findings

Descriptive results and findings were crucial in providing foundation for building the classifier model while experimental results and findings were crucial in building the classifier model. They both provided crucial information that was needed to execute those two processes respectively. Besides, they were both vital in answering research questions under investigation, namely research question 1, 2 and 3.

4.3.1. Discussion of Descriptive Findings

4.3.1.1. Concepts as target classes for machine learning process

Findings#1, #2 and #6 were crucial in discovering industry roles concepts that formed the basis of creating target classes for machine learning. While findings#1 & #2 revealed the concepts as raw which were initially 7, findings#6 later on revealed the refined form of these concepts as 12. Finding#6 also revealed the distribution of these concepts that was important in deciding how to handle class imbalances during training process of machine learning.

4.3.1.2. Characteristics of target classes for machine learning process

The choice and design of machine learning methodology depends on: 1) structure of the problem and 2) assumptions about the learning problem (Kotsiantis, 2007; Silla & Freitas, 2011; Merschamann & Freitas, 2013). As a result, findings#2 was crucial in discovering that these concepts had similar structural elements (job activities/skills) but different levels of emphasis. Further, findings#5 discovered the structural relationship among these concepts that was crucial in deciding the machine learning approach suitable for building the classifier model, in this case hierarchical classification approach.

The fundamental assumption in the present study that occupational industry roles have different requirements for problem solving skills was put in the form of a hypothesis under research question 2: **H_{02A}: There is no significant boundary differences between concepts to be used as potential target classes for machine learning.** Findinsgs#3 was crucial in rejecting this hypothesis. Finding#4 was important in revealing that learning institutions have different biases towards these concepts. This was crucial in designing the prototype software to handle graduates from different learning institutions differently when deployed in the real world.

4.3.2. Discussion of Experimental Findings

4.3.2.1. Selection of meaningful features

Findings#7 was related to determination of not only the number of features that would maximize performance of the classifier model but also whether the improved performance was significant. Findings #7 revealed 5 features out of 13 were able to induce better performance results for the classifier model equivalent to performance that could be achieved with 13 features. Besides, there was significant improvement in performance leading to a conclusion that reduction of features has a number of benefits to the classifier model, including lowering demand for computational resources and reducing the processing time. The findings revealed R (Relevant content), D (Cognitive skills), A (Technical skills), C (Intellectual Capacity) and 'Age' as the only 5 features out of 13 which were able to induce optimal performance to the classifier model and this performance improvement was significantly better than that of 13 feature model.

The implication of these findings provided insight not only into which features should be included in the subsequent investigations but also to accept or reject the hypothesis posed in research question 1: **H_{01A}: All features are equally relevant for inducing better performance in the classifier model.** The outcome based on these findings was to reject the hypothesis at significance level, p=0.05. These findings' explanation was that when more than five features were used, the summary feature space dimension became too large causing performance of the model to start decreasing and while when less than five features were used essential information was lost that caused accuracy to decline (Barbedo & Lopes, 2006).

4.3.2.2. Selection of the best induction algorithm for the model

The main focus of this experiment was to estimate the generalization performance of each of the two models generated by each machine learning algorithm and select the best. Both findings#9 and #11 were key in revealing this fact where both concurred that the general performance of the SVM classifier model was much better than that of naïve Bayes and in fact the difference between the two was significant. Based on these findings, SVM was more likely to generalize its performance to unseen data in the real world better than naïve Bayes classifier model. As a result, it was selected as the best induction algorithm for classifier model. Also, the two findings were in concurrency in rejecting a hypothesis posed in the research question 3 that: **H_{o3C}: All induction algorithms induced equal generalization error.**

4.3.2.3. Selection of the best parameter values

Both finding#8 and finding#10 were related to investigation towards parameter tuning, although through different datasets with different landscapes. Coincidentally, both findings agreed that parameter tuning of SVM improved performance of the classifier model significantly. However, parameter values that induced the best performance of the classifier model were dataset dependent. The implication of these findings in this investigation suggested that in every different dataset we needed to tune the parameter values for the best performance. Also, these findings provided key evidence that was used to reject the hypothesis paused in research question 3 that: **H_{03B}: Any parameter value induces optimal performance in the model.**

4.3.3. Discussions Conclusion of Modeling Findings

The conclusion relates to the research questions which were subject to investigation, namely research questions 1, 2 and 3.

RQ1: What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles?

Based on the findings in the present study, it is important to note when developing classifier models for mapping skills to industry roles that appropriate attributes that are valid for machine learning are content knowledge, cognitive skills, technical skills, academic capacity, and age. Table 4.3a illustrates method followed to arrive at the findings.

Table 4.3a: Method followed to answer research question 1

METHOD	FINDINGS
1. Literature analysis	Obtained 13 concepts: 1. Independent factors (4 concepts) 2. Confounding factors (9 concepts)
2. Evaluation of three filter algorithms using benchmark dataset [Experiment]	Obtained meaningful concepts under each algorithm
3. Analysis of three algorithms' results for relevant concepts	Established 5 relevant concepts appearing in at least two results of the three algorithms: 1. Independent factors (4 concepts) 2. Confounding factors (1 concepts)
4. Use the relevant concepts to develop the conceptual model	Obtained a validated conceptual model (**OUTCOME**)

RQ2: What is the structural characteristic of concepts required as target classes for machine learning process of mapping graduates' skills to industry roles?

Based on the findings in the present study, it is important to note when developing classifier models for mapping skills to industry roles that target classes for machine learning are industry roles concepts which are distinct, and therefore, should be approached using supervised classification approach. Class distributions of these concepts are imbalanced, and therefore, they need stratified sampling during machine learning process of building the classifier model.

Besides, structural relationship among these concepts is hierarchical, and therefore, the process of building the classifier model should be approached using hierarchical machine learning approach. Finally, when designing software to deploy for real world use, the underlying biases of different learning institutions towards these concepts should be known so that the software can handle graduates from different institutions differently. Table 4.3b illustrates method followed to arrive at the findings.

Table 4.3b: Method followed to answer research question 2

METHOD	FINDINGS
1. Literature analysis	Obtained three dimensions: 1. Main competence 2. Specific competence 3. Proficiency
2. Analysis of data collected [Descriptive]	Established relationships between industry roles 1. Main roles [Programmer, Designer, Manager] 2. Specific roles within main roles [total of 12] 3. Skill levels among main roles [3 levels]
3. Graphically represent relationships	Obtained hierarchical structure (**OUTCOME**)

RQ3: How do we build an appropriate machine learning model for mapping graduates' skills to hierarchically structured occupational industry roles?

Based on the findings and outcomes of research hypotheses that were tested, three things are key in building machine learning model for mapping graduates skills to industry roles, namely selection of appropriate features, tuning parameters of the model to appropriate values, and selecting induction algorithm that induces appropriate generalization performance to the model. These three are key determinants of the final performance of the model. Table 4.3c illustrates method followed to arrive at the findings.

Table 4.3c: Method followed to answer research question 3

METHOD	FINDINGS
1. Data collection [Survey]	Obtained 78.9% response rate - Out of 190 questionnaires 150 were returned
2. Data preprocessing	Obtained cleaned and scaled data - Out of 150 records, 37 with missing values removed - Out of 17 variables, 11 were digitized 6 discretized - All variables were standardized
3. Construction	Obtained design of mapping model - Design architecture - Design of Algorithm
4. Evaluation of two induction algorithms using data collected and benchmark dataset [Experiment]	Established the best induction algorithm for the model - SVM
5. Evaluation of parameter values of the best induction algorithm [Experiment]	Established the best parameter values for the model [Kernel = gamma (values>0.1), complexity = 0.0001 to 1000]
6. Building model using the best induction algorithm and the best parameter values	Obtained the ML mapping model (**OUTCOME**)

4.4. Summary

This chapter has presented results of the study, both descriptive and experimental, and a detailed discussion of the major research findings. For purpose of clarity, the results have been presented using not only tables and but also graphs. The statistical analysis procedures have been carefully selected based on preliminary tests results for data validity. The final research findings have been carefully drawn from both descriptive and experimental results after detailed discussion of the results.

In summary, the results findings discussed in this chapter have literally provided answers to three research questions posed in this study. What concepts are appropriate as machine learning attributes for mapping graduates' skills to occupational industry roles? This was the first research question which was answered through experiment A where five features were selected as relevant for the ML model. What is the structural characteristic of concepts that correctly reflects the hierarchy of industry roles required as target classes for machine learning purpose? This was the second research question which was answered through descriptive analysis where the hierarchical structure was conceptualized and described. How do we build using these concepts an appropriate machine learning model for mapping graduates' skills to hierarchically structured industry roles? This was the third research question which was answered through experiment C and B. whereas experiment C provided the appropriate induction algorithm to use when building the ML model, experiment B provided appropriate parameter values for that induction algorithm.

CHAPTER 5: PROTOTYPE DESIGN AND IMPLEMENTATION

5.0. Introduction

Design and implementation of a software prototype can be a complex task, especially, if an organized approach is not followed. This chapter presents an elaborate description of the design and implementation aspects of the software prototype for the skills mapping model. The chapter is organized into three sections as follows: section 5.1 discusses the background and details of prototype development methodology, section 5.2 highlights the computing resources utilized, and section 5.3 closes the chapter with a summary.

5.1. Prototype Development Methodology

Prototype development methodology, as applied in this study, is a reference model for software development process that provides a common basis for standards, description of major functions involved in the software development, and an insight into important features necessary for common understanding and focus.

Ideally, software prototype development is part of a broader field known as software engineering where several software development process models are presented, such as waterfall, prototyping, Rapid Application Development (RAD) and evolutionary models. Generic software engineering activities which are executed within different software development models include requirements specification, software design, software implementation, software validation. These activities may be carried out linearly, or iteratively, or cyclically, or a combination of these, depending on the assumptions behind the software development methodology adopted.

5.1.1. Choice of Prototype Development Methodology

Since we did not have detailed requirements for the customer, there was need for a customer driven model. A process model that generates the first version of the usable product quickly and subsequently to be used not only to solicit for more requirements from customers but also to keep the customer happy with a working version that keeps them busy as we incrementally improve on it. This suggested two important principles in software development, i.e. incrementality and reusability. Incrementality principle ensured easier to make small changes to a working system than to rebuild the system while reusability ensured standard components that are flexible to changes.

As a result the design methodology that promised to fulfill this desire was an incremental model. The incremental model combines elements of linear sequential model (applied repetitively) with the iterative philosophy of prototyping (Pressman, 2001). It applies linear sequences in a staggered fashion to deliver software in small but usable pieces, called "increments". Each increment builds on those "increments" that have already been delivered. When an incremental model is used, the first increment is often called the "core product", which addresses only the basic requirements, but many supplementary features (some known, others unknown) remain undelivered.

Figure.5.1 presents the stages of the incremental model followed in this study. The model's activities were done in successive iterations, each of which ended with the delivery of a new version of an increment (P) that was usable, until the product's final version is delivered.

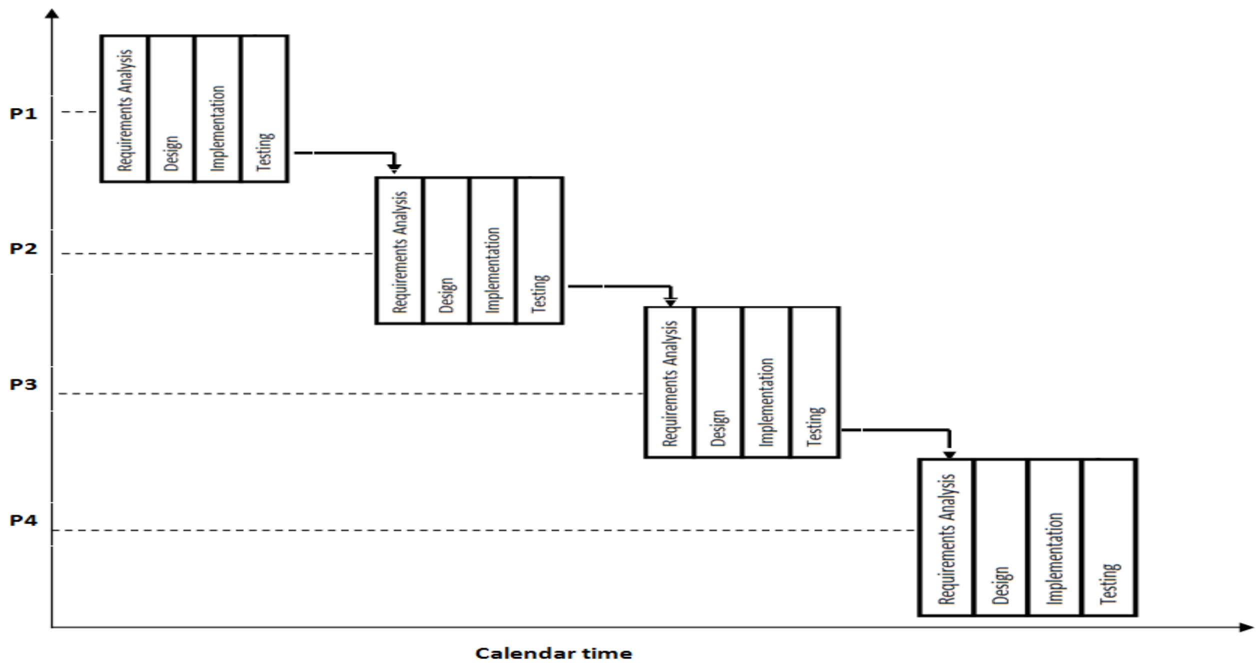


Figure 5.1: Incremental model adapted from (Pressman, 2001)

Incremental model includes the following advantages: 1) Customer value can be delivered with each increment, so system functionality is available early, 2) Early increments act as a prototype to help elicit requirements for later increments, 3) Lower risk of overall project failure, and 4) The highest priority system services tend to receive the most testing. Besides, incremental model fulfills all the typical characteristics that are commonly used as a criteria for choosing a software process model such as: 1) Visibility i.e. easy for an external assessor to determine the progress made, 2) Reliability i.e. how good the process is at detecting errors before they appear in a product, 3) Robustness i.e.

how well the process is in coping with unexpected change, 4) Maintainability i.e. easy to change so as to take account of changed circumstances, 5) Rapidity i.e. how fast a system can be produced.

Incremental model is mostly important when staffs are unavailable for a complete implementation by the deadline that has been established for the project. The basic idea is, if the core product is well received, then additional staff (if required) can be added to implement the next increments where early increments can be implemented with fewer people.

Initially, rapid prototyping was applied where a laboratory prototype was designed and used to investigate on the initial set of the skills mapping software requirements. The laboratory prototype was then incrementally developed and tested for maturity until it became a field prototype. The field prototype was derived by adding a better user interface to the laboratory prototype, before it was ready to be tested with the real world data collected from the real environment. After the field prototype was successfully tested with the real data, it was then considered as the final requirements specification for the production version (Kemboi, 2013).

The rest of this section highlights each of the Software Engineering activity as applied in the software prototype methodology of the current study.

5.1.2. Requirements Analysis

This involves an elicitation activity which resulted into an initial set of requirements specifications. The initial set was the basis for the design of the preliminary research prototype. The requirements specifications were revised every time the research prototype evolved. This version of the specifications was the basis for developing the lab prototype. At the end of the lab prototype development and testing, the requirements specifications were again revised. This second revision was the basis for developing the field prototype. The final requirements specifications were then produced after the implementation and testing of the field prototype. The final set of requirements specifications were the basis for the production of the software prototype developed.

Traditionally, the requirements for any software will be manifested by a number of analysis models such as data models, functional models, and behavioral models (Pressman, 2001). In the current study, the plan section of the research design model in Fig. 3.1 of this study indicates the source of requirements where both industry and academia were target grounds for requirements elicitation. As a result of analysis of data derived from these two areas, initial data model for the prototype was constructed.

1) Use case model

A use case model which describes the function of the system as viewed by its users, developers, and testers, was developed as the initial specification of the skills mapping model's requirements . Fig. 5.2 presents the functional model in the form of a use case model.

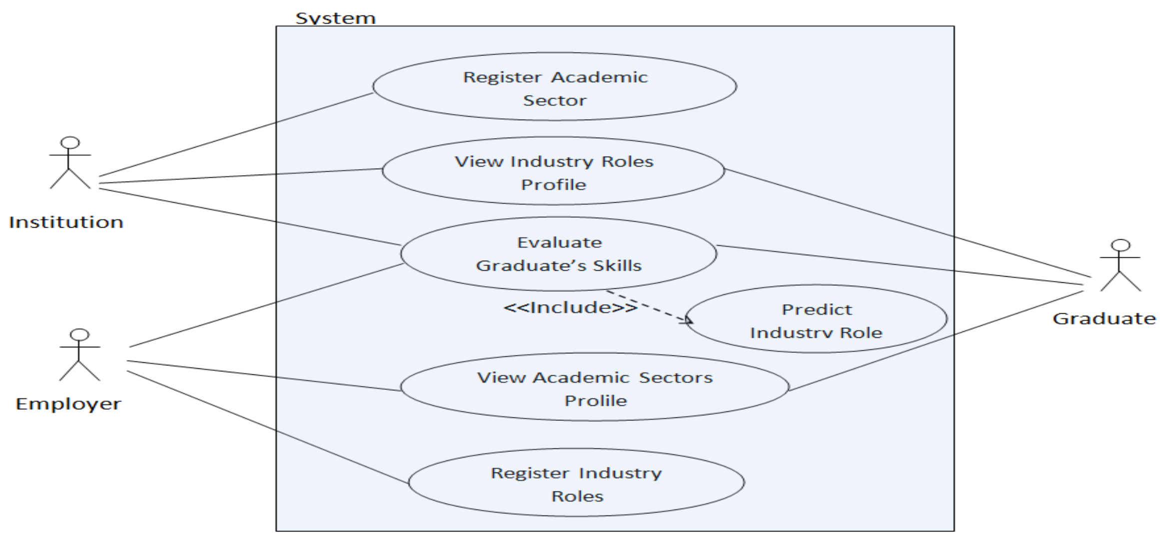


Figure 5.2: Use Case Model

The use case model envisaged three kind of users for the model prototype i.e. employer, graduate, and university institution. Employers should be able to register industry roles available in various sectors in which they operate, clearly indicating their minimum skills and knowledge index values requirements. Also, they should be able to view academic sector profiles for various institutions based on their knowledge and skills content in the exams each year they examine. Finally, employers should be able to evaluate new graduates on industry roles suitability.

Likewise, institutions should be able to register their academic profiles for sectors in which their degree programs are based. Where for each sector, each year they should record knowledge and skills indices derived from their exam's content administered to students. Also, they should be able view industry roles profiles for various sectors based on knowledge and skills minimum indices required by industry. Finally, institutions should be able to evaluate their graduates on industry roles suitability before they graduate so as to assess themselves against industry requirements.

Graduates, as well should be able to evaluate themselves against industry roles requirements to determine their suitability for employment. They should, also, be able to view industry role requirements for various sectors in industry as well as view academic performance profiles in various sectors for various institutions.

2) **Class model**

Also, a data model, which also describes the information requirements of the domain, was developed as the initial specification of the skills mapping model. Fig. 5.3 presents the data model in form of a class model.

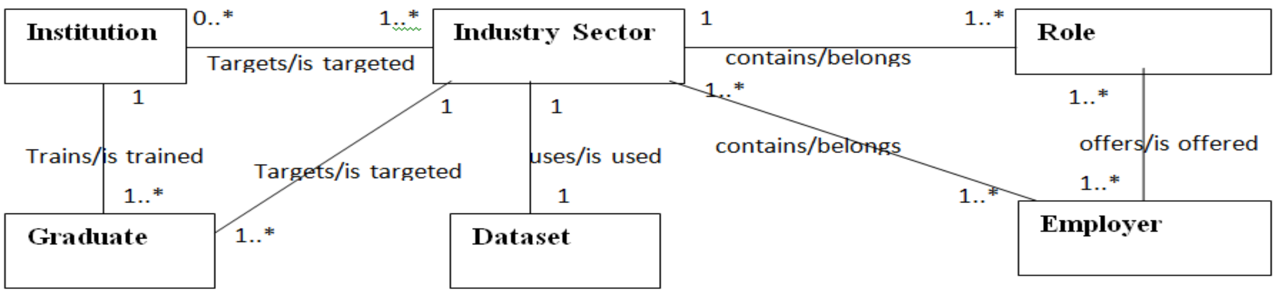


Figure 5.3: Class Model

5.1.3. Design

A preliminary design was constructed just after initial set of requirements specifications was determined, and a preliminary mapping model was specified. This design was the basis for the research prototype which was used to produce the requirements specifications for the lab prototype. Another cycle of the design was done after the lab prototype specifications were determined, and more elaboration on the mapping model was conducted. This cycle was repeated after the implementation and testing of the lab prototype to produce the design of the field prototype. Once the field prototype was implemented and tested, the final design of the software system was issued.

The analysis models produced during requirements specification feed into the design task where four design models required to complete the design specification were produced. The four design models are architectural design, data design, interface design, and component design (Pressman. 2001). A wireframe design was produced to guide the overall design process of the four design models. Fig. 5.3b illustrates the wire-design for the software prototype where interfaces were defined for the following types of users, namely EMP= Employer, GRAD = Graduate, COLL= college institution, ADM = Administrator for the software, and RESU= Results for all users.

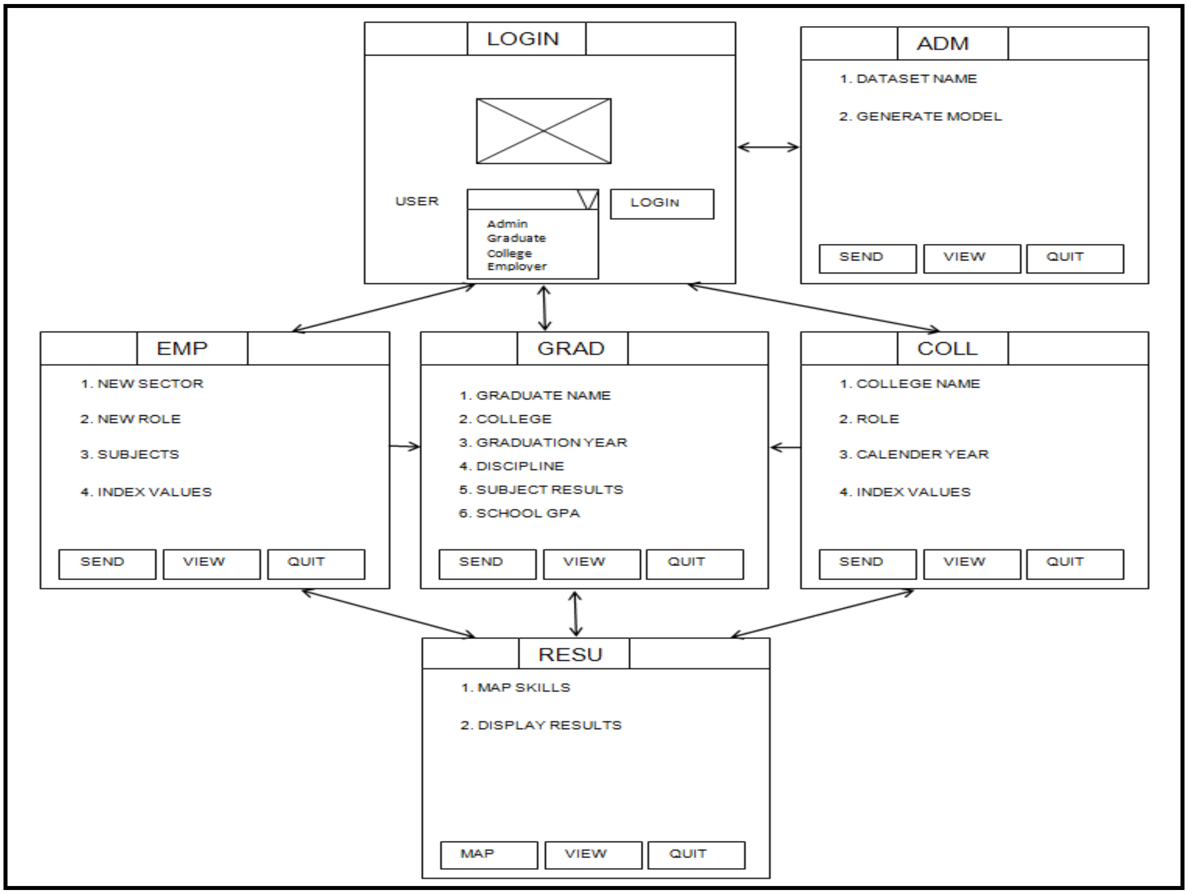


Figure 5.3b: Wire-frame for the Prototype software Design

1) Architectural design

Architectural design defines the relationship between major structural elements of the software and the design pattern that is appropriate to achieve the requirements. Layering is a strategy that is often used to divide a system into subsystems where two approaches used to guide the layering styles are either responsibility or reuse driven (Ojo & Estevez, 2005). In the current study, a preliminary architectural design of the research prototype was based on responsibility driven layering of the basic system elements i.e. input, process, and output. Thus, each layer was designed to fulfill a specific role.

The input component was designed as a data source system to provide input data to the prototype while the output component was designed as a dashboard subsystem that presents the results of the prototype to the user. The process component is the core function of the prototype and was designed as a machine learning subsystem that provides transformative function for mapping skills to industry roles. Each layer makes use of the services provided by the lower layers. Fig. 5.4 presents the skeleton for the architectural design of the research prototype.

5.1.3.1. Data Source Subsystem

The data source consists of two components: a) Database and b) Dataset. The two components realize or implement an interface that they export for the other subsystems to access them i.e. Sinterface and Dinterface. Fig. 5.5 presents the high level design of the subsystem.

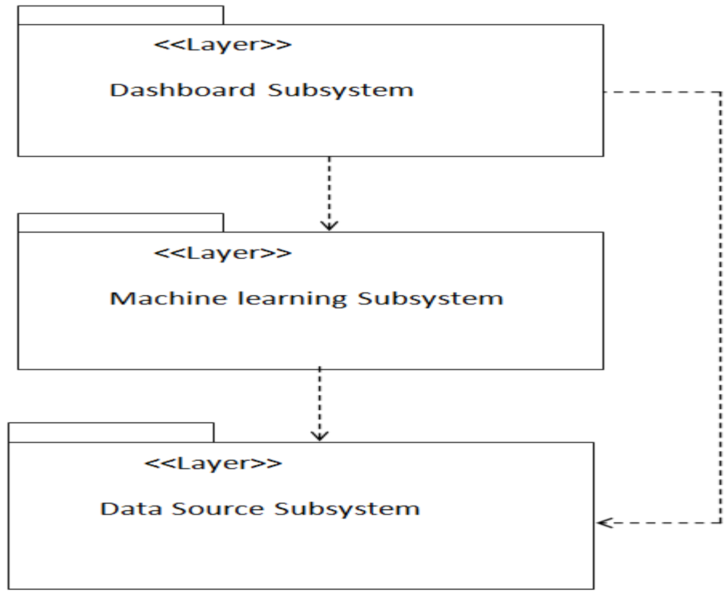


Figure 5.4: Architectural design model for the prototype

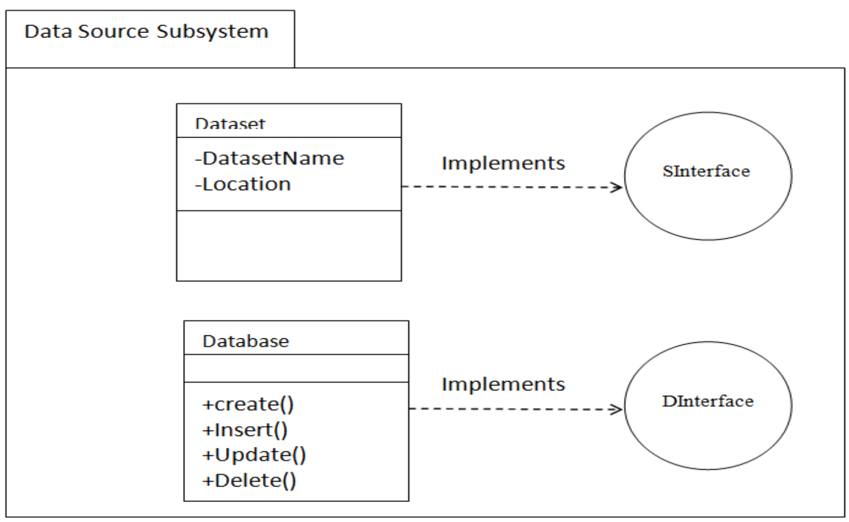


Figure 5.5: Components of the data source subsystem

i) Database

A database is an organized collection of central data and was used in this study to store user-centered data generated at the dashboard, as well as a source of data to feed into the machine learning model to produce predictions. The database was designed based on the class model identified during data requirements analysis and variables of the conceptual mapping model shown in Fig.2.6. Relational data model was selected as a basis of deriving the database design whose main components are related tables storing industry-academia requirements data, such as sectors' table, roles' table, institutions' table, institution/sector index table, and dataset table as shown in Fig.5.6.

Fig.5.6 was derived from the class model in Fig.5.3 by removing 'Graduate' and 'Employer' classes in the model and also resolving the many to many relationship between 'Industry sector' and 'Institution' by creating an extra class 'Institution sector indices' to store sector indices derived from academic institutions. The decision to remove was arrived at after an assumption that they are both external actors whose data may not be needed to be stored in the system as indicated by the use case model. However, although 'Institution' is also an external actor, it is quite important to store its data because each academic sector registered in the system would be associated with a particular institution and it is important to store academic institutions' indices for various industry sectors.

The complete set of attributes for the database was determined using the data collected and analyzed during descriptive analysis stage.

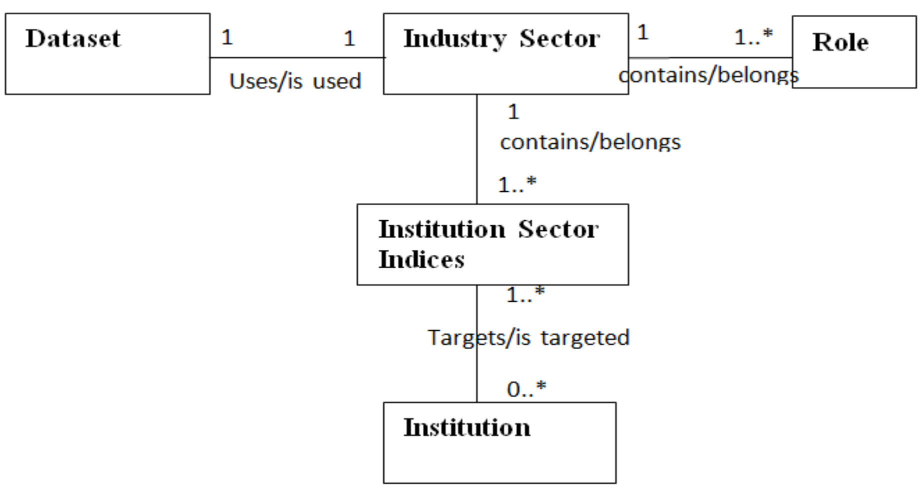


Figure 5.6: Database Model

Table 5.1 shows the detailed description of each table indicating the purpose, fields, data type, data width, and primary key.

Table 5.1: Detailed description for database model tables

Table	Purpose	Fields Data type (Data Width)	Primary/ Foreign Key
Sector	To store academia subject details for each industry sector.	ID Integer(AutoIncrement), NAME CHAR(50), SUBJECT1 CHAR(50), SUBJECT2 CHAR(50), SUBJECT3 CHAR(50), SUBJECT4 CHAR(50), SUBJECT5 CHAR(50), SUBJECT6 CHAR(50), SUBJECT7 CHAR(50)	ID (Primary Key)
Role	To store index details for each sector in the industry	ROLEID INTEGER (AUTOINCREMENT), NAME CHAR(50, SECTORID INTEGER, RI CHAR(10), DI CHAR(10), AI CHAR(10), CI CHAR(10), FOREIGN KEY(SECTORID) REFERENCES SECTOR(ID)	ROLEID – Primary Key SECTORID – Foreign Key
Institution	To store name details for institutions in the academia	ID INTEGER(AUTOINCREMENT), NAME CHAR(50)	ID – Primary Key
Institution/ sector Index	To store institutions' yearly indexes for each sector in the academia	ID INTEGER (AUTOINCREMENT), INSTID INTEGER, SECTORID INTEGER, YEAR CHAR(10), RI CHAR(10), DI CHAR(10), FOREIGN KEY(INSTID) REFERENCES INSTITUTION(ID), FOREIGN KEY(SECTORID) REFERENCES SECTOR(ID)	ID – Primary Key INSTID- Foreign Key SECTORID- Foreign Key
Dataset	To store training dataset for each sector in the industry	ID INTEGER (AUTOINCREMENT), NAME CHAR(50), PATH CHAR(50), SECTORID INTEGER, FOREIGN KEY(SECTORID) REFERENCES SECTOR(ID)	ID- Primary Key SECTORID- Foreign Key

ii) Dataset

A dataset, which is a collection of data that is stored in a specific format, was used for the purpose of learning and evaluating the machine learning model. The data set was used as input to the machine learning subsystem where a machine learning model is generated. In the current study, the data set was derived from the data collected after pre-processing. The pre-processing was conducted using Ms Excel spreadsheet where the predictors' and class values were defined before getting converted into a .csv text file. Originally, the structure of the dataset consisted of attributes derived both from the questionnaire and some computed attributes. Table 5.2 shows the original set of the dataset attributes.

Table 5.2: original set of the dataset attributes

NO.	ATTRIBUTES	VALUES	DESCRIPTION
1	GENDER	{Male, Female}	Gender
2	AGE	{20-24, 25-29, 30-34, 35-39, 40 and above}	Age
3	LOLE	{ Local, Abroad}	Place of O-level Study
4	GSOLE	{Grades, Points, Marks}	Grading System of O-level
5	ROLE	{Less than 4, 5-7, 8-10, 11 and above }	Results for O-level
6	BDGREE	{Computer Science, Information Technology, Software Engineering, Other}	Type of Bachelor's Degree
7	UNIVERSITY	{UON, KU, JKUAT, MOI, EGERTON, Strathmore, KEMU, Daystar, Nazarene, Maseno, Other}	University of Study for Bachelors
8	GSBDEGREE	{Grades, Points, Marks}	Grading System for Bachelors
9	RBACHELORS	{Less than 4, 5-7, 8-10, 11 and above }	Results for Bachelors
10	FIRSTJOB	{Software Architect, Analyst Programmer, Test Engineer, Web Programmer, Mobile Programmer, System programmer, Project manager, Other }	First Appointed Job
11	CURRENTJOB	{Software Architect, Analyst Programmer, TestEngineer, Web Programmer, Mobile Programmer, System programmer, Project manager, Other }	Current Job
12	CHANGEDJOB	{NO, YES}	Current Job Is Different From First Job
13	ATTRACTOR	{Passion, Salary, Ambition, Qualification, Other}	Enticing Factor to Current Job
14	SEEXAM	{100%, 75%, 50%, 25%, 0%}	Se Content In Exam
15	R	{interval value}	Index of Content Knowledge Components
16	D	{interval value}	Index of Cognitive Skills Components
17	A	{interval value}	Index of Technical Skills Components
18	C	{interval value}	Index of Academic Capacity Components

5.1.3.2. Machine learning subsystem

This is the transformative function and core component that forms the predictive engine of the prototype. While the transformative function involves mapping skills to industry roles, two main activities involved to achieve this were learning and classification. Therefore, design of this component involved designing the algorithm for machine learning and classification. Fig.5.7 presents the design model for machine learning and classification. This was derived from a section of the class model in Fig.5.3 where the section consisting of 'Industry sector', 'Role', and 'Dataset' classes was extracted as the main classes in the interaction. In order to align the classes with respect to the new roles they were to play, some of the classes were renamed, such as 'Industry sector' was renamed as 'Algorithm', while 'Dataset' as 'Model'.

Moreover, two machine learning techniques, naïve Bayes and support vector machines were adopted as the basis for designing the algorithms to implement the architecture and learn the model. This is because of their good incremental learning ability and assurance of high accuracy in either cases of small or large dataset where each of them is good at. Further, these techniques are widely used in supervised learning and belong to two different families of learning algorithms i.e. instance-based and kernel machines, as described in Table 2.4 in the literature review. As a result, two classes, 'SVM' and 'naiveBayes', were introduced in a generalization relationship with the 'Algorithm' class in the design. Other classes in this subsystem include 'Model' class that stores the model object generated by the 'ML Algorithm' class and 'Role' class that stores the hierarchical structure of industry roles of each sector.

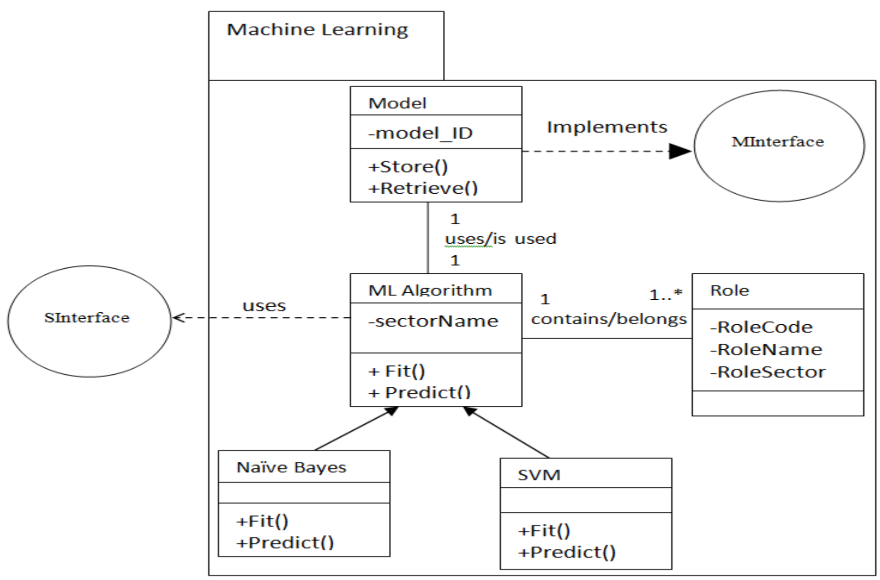


Figure 5.7: Design model for machine learning subsystem

Fig.5.7 illustrates the design model for the algorithms that was adopted to learn the model. This subsystem relies on an interface, 'Sinterface', created by the data source subsystem to realize its behavior and implements an interface, 'Minterface', which it exports for other subsystems to use. The 'Minterface' enables the learned model to be accessed and used by other subsystems, such as the Dashboard subsystem, while 'Sinterface' enables the machine learning subsystem to access the dataset to be used for learning the model.

Basically, the ML algorithm was designed such that there were two core methods, 'fit' and 'predict'.

1) <u>Fit Algorithm explanation</u>

This method is responsible for fitting the data into the model to learn or estimate the parameters. Fig. 5.7a outlines algorithm of the 'fit' method. This algorithm takes in the taxonomic tree in which the industry roles are organized and the dataset containing graduate employees details to be learned. The algorithm is able to group the dataset content based on their dependent values according to the various sections of the taxonomic tree such as sub-tree, non-leaf nodes, leaf nodes, or tree heights. The algorithm is able to learn how items of the dataset belonging to various leaf nodes look like, if they belong to known non-leaf nodes and various non-leaf nodes are distinguished by their height levels in the tree or sub-trees. Finally, the algorithm is able to store the learned knowledge rules for that particular dataset. Therefore, the key aspects of this algorithm are: 1) input 2) learning 3) storing the learned knowledge rules.

```
Fit(taxonomy_tree, dataset)
 1_Get taxonomy_tree's height/levels
 1_Get subtrees/functions
 1_For each subtree/function
        2_Get subtree's leaf nodes/classes
        2_Get other subtrees' leaf nodes/classes
        2_Create subtree's (function) classifier object
 1_For each subtree's non-leaf nodes/proficiencies
        2_Get leaf children
        2_Get other non-leaf nodes' leaf children
        2_Create non-leaf node's (proficiency) classifier object
 1_For each subtree's leaf nodes/specialties
        2_Get leaf node/class
        2_Get siblings
        2_Create leaf node's (specialty) classifier object
 1_Store classifier objects in a data structure object
                    1:Function objects
                              2:Proficiency objects (ordered by taxonomic_tree's height/levels)
                              2:Specialty objects (ordered by taxonomic_tree's height/levels)
```

Figure 5.7a: Fit Method's Algorithm

2) **<u>Predict Algorithm explanation</u>**

This method is responsible for the prediction function of the model. Fig. 5.7b outlines the algorithm of the 'predict' method. The algorithm takes in an instance of unemployed graduate's data and taxonomic tree for industry roles in which the graduate is seeking for employment. The algorithm uses the knowledge rules generated by the 'fit' algorithm to decide the role for which the graduate is suitable. The key aspects for this algorithm are: 1) input tree and graduate data 2) load the knowledge rules from the store 3) search for the appropriate knowledge rules to process the graduate data 4) use the rules to decide the industry role suitable for the graduate.

<u>Predict(taxonomy_tree, data)</u>

1_Load classifier objects

1_Get taxonomy_tree's width/subtrees/functions

1_For each subtree/function

    2_Get function classifier objects

    2_Predict data's function

    2_Select function of classifier object that predicts +ve

1_For each subtree's non-leaf nodes/proficiencies ordered in ascending order of levels

    2_Get corresponding order's proficiency classifier objects

    2_Predict data's proficiency

    2_Select proficiency of classifier object that predicts +ve

1_Get current non-leaf node's specialty classifier objects

    2_ Get specialty classifier object

    2_Predict data's specialty

    2_Select specialty of classifier object that predicts +ve

1_Report industry role = function+specialty+proficiency

Figure 5.7b: Predict Method's Algorithm

5.1.3.3. Dashboard Subsystem

The main purpose of this is to link the user of the prototype with the prediction engine using interactive user interfaces. This was based on a class model in Fig.5.3 where the 'Graduate' and 'Employer' classes were used to design two categories of user interfaces. While 'Graduate' class was used to produce 'GraduateUI' class, 'Employer' class was used to produce 'EmployerUI' class. Fig

5.8 presents the design model for the dashboard subsystem. These two have all their functionalities similar except for only two, 'RegisterIndustryRoles' and 'RegisterAcademicSectors'. Fig.5.9 presents design model for the user interfaces. The dashboard subsystem uses two interfaces, 'Minterface' from the machine learning subsystem and 'Dinterface' from the data source subsystem. 'Dinterface' enables this subsystem to access and use the database while 'Minterface' enables the subsystem to access and use the mapping model.

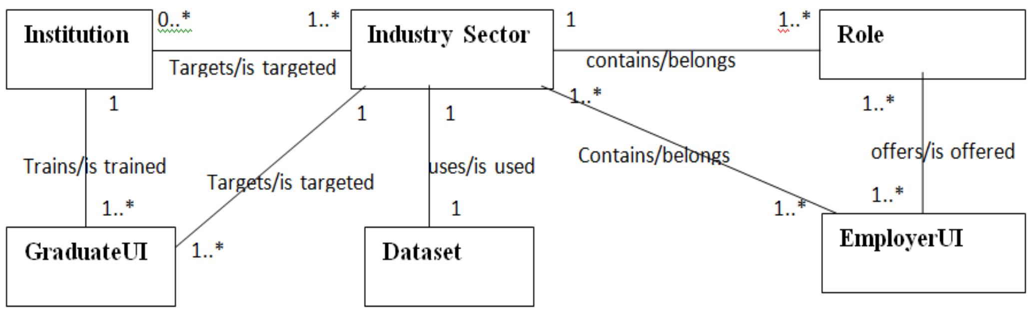


Figure 5.8: Design model for the Dashboard Subsystem.

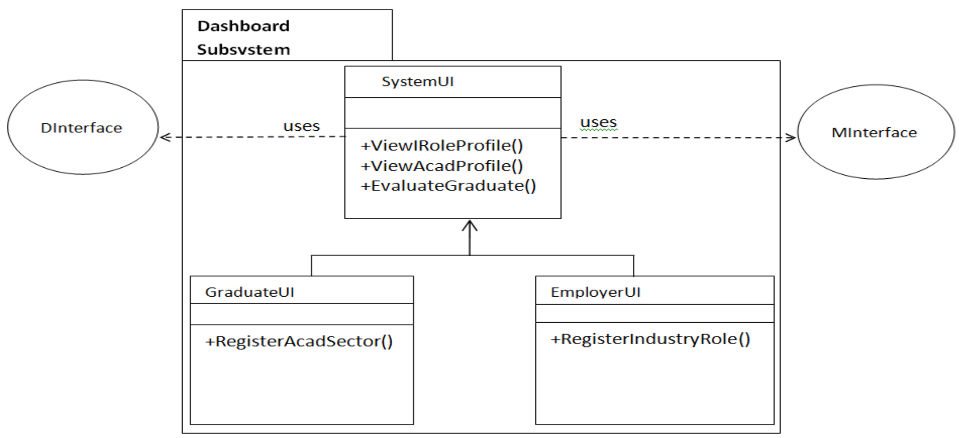


Figure 5.9: Design model for user interfaces

5.1.4. Implementation and Testing

The implementation process of the prototype for the mapping model was conducted by first reviewing and evaluating existing machine learning techniques and this resulted into choosing the most generally applicable techniques that would be suitable to support the building of the research prototype. Further, python was identified as the most common platform for machine learning implementations and was reviewed to determine how it could be used with additional technologies to implement the prototype. Finally, the construction of the prototype was implemented using python technology and as WEMA (Where Employers Meet Academia) platform being the preferred name for the prototype. The WEMA platform was tested and validated using data collected, and a presentation of how it operates including analysis of its performance was conducted. Fig.5.10 shows the welcome screen of the prototype implementation.

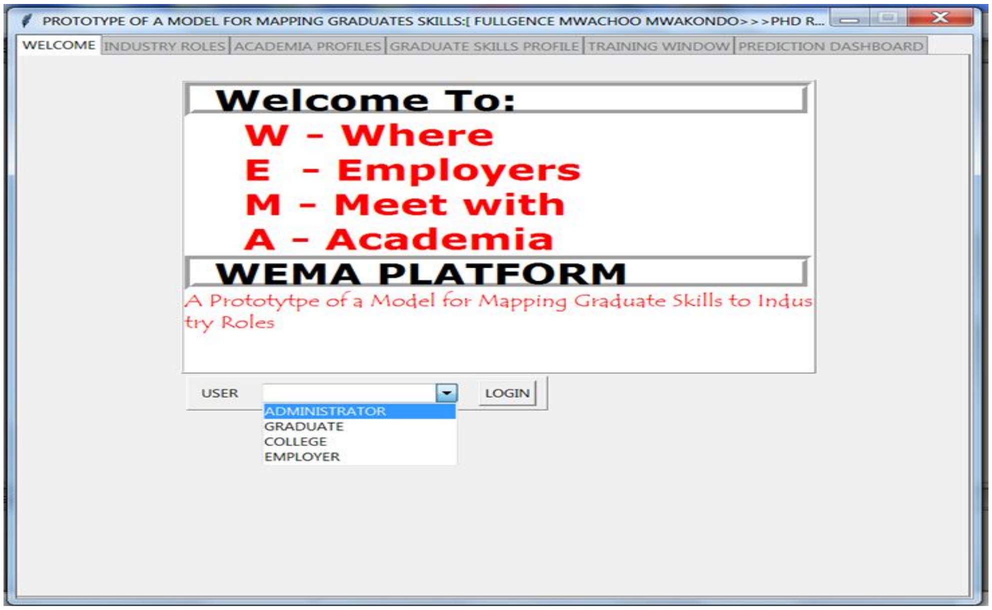


Figure 5.10: welcome screen for the prototype implementation

A number of system elements designed in the previous section were eventually implemented as describe below:

5.1.4.1. Implementation of Data source subsystem

(1) Database Class

The database design was implemented using SQL Technology that is already integrated in python as SQLite. This involved implementing the SQLite class where several of its methods were used. The connection method of the SQLite class is a very useful method for accessing the implemented database and, therefore, was used to implement the 'Dinterface'. Fig. 5.11 presents a snapshot of the code that was used to implement the database class.

<u>Code segment explanation</u>

The code illustrates that the model uses a number of concepts that are key to its operations. These concepts are stored in a number of tables that are related and sqlite technology was used to implement and store this relationship in a database object. The database object was defined using object-oriented concept known as class. Therefore, the key aspects of this code are: 1) class 2) sqlite 3) tables 4) relationships.

```
class database():
        def __init__(self):
                #self.master = master
                self.value = None
                #THIS CREATES A NEW DATABASE
        def createDatabase(self):
                conn = sqlite3.connect('prototype.db')
                #print("Opened database successfully")
                conn.execute("DROP TABLE SECTOR")
                conn.execute("CREATE TABLE SECTOR(ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME  CHAR(50)  NOT NULL,SUBJECT1 CHAR(50),SUBJECT2 CH
                #print("SECTOR Table created successfully")

                conn.execute("DROP TABLE ROLE")
                conn.execute("CREATE TABLE ROLE(ROLEID INTEGER PRIMARY KEY AUTOINCREMENT,NAME  CHAR(50)  NOT NULL,SECTORID  INTEGER,RI CHAR(1
                #print("ROLE Table created successfully")


                conn.execute("DROP TABLE INSTITUTION")
                conn.execute("CREATE TABLE INSTITUTION(ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME  CHAR(50)  NOT NULL)")
                #print("INSTITUTION Table created successfully")


                conn.execute("DROP TABLE INST_SECTOR_INDEX")
                conn.execute("CREATE TABLE INST_SECTOR_INDEX(ID INTEGER PRIMARY KEY AUTOINCREMENT,INSTID INTEGER,SECTORID INTEGER,YEAR CHAR(1
                #print("ACADEMIAINDEX Table created successfully")
```

Figure 5.11: Database class code segment

(2) Dataset class

The dataset was implemented directly as a text file and to be stored as .csv format where the python csv class was used for implementation. The read method of the csv class was used to access and retrieve the dataset and, therefore was used as the implementation of the 'Sinterface'.

5.1.4.2. Implementation of Machine learning subsystem

(1) Role class

The implementation involved coding classes mapped on the taxonomic structure. Classes on the leaf nodes were coded with non-negative integers while internal nodes were coded serially with negative integers. The levels of the taxonomic structure were coded from 0 as topmost and downwardly. Then the whole taxonomic structure was represented using a python data dictionary noting the structural relationships in terms of level, parent class, and child classes. Fig.5.12 presents a segment of the dictionary data structure that was used to implement the taxonomic structure and its implementation code.

Code segment explanation

This code segment illustrates how the taxonomic tree mentioned in Fig.5.7a&b was implemented using data structure in python technology called data dictionary. The structure contains a number of data items that represent codes for industry roles which were arranged methodically according to order described by the following structure:

{classCode:[[levelcode], [parentClasscode], [childsClasscodes]], classCode:[[levelcode], [parentClasscode], [childsClasscodes]],……….}.

```
#THIS IS THE TAXONOMY FOR EMPLOYEE DATASET
STRUCTURE2 = {-1:[ [0],[],[-2,1,2],[1]],-4:[[0],[],[-5,7,8],[1]],-2:[ [1],[-1],[-3,3,4],[1]],-5:[[1],[1],[-6,9,10],[1]],-3:[ [2],[-2],[5,6],[1]],
                                      4:[[2],[-2],[],[1]],5:[ [3],[-3],[],[1]],6:[[3],[-3],[],[1]],7:[ [1],[-4],[],[1]],8:[[1],[-4],[],[1]], 9:[ [
```

Figure 5.12: Role class code segment

(2) ML algorithm class

The overall implementation of this class's generalization relationship was achieved using the concept of polymorphism. The 'fit' method was implemented as 'classify' method while the 'predict' method was implemented as 'classifyinstance' method in two separate classes. The two classes are 'svmRootclassfier' and 'naiveRootclassifier' for svm and naïveBayes algorithms respectively. Fig. 5.13a presents segment codes of the 'svmRootclassfier'classes' . A number of python technologies were plugged in to realize the purpose of the code such as svmpy, numpy, pickle. The final implementation of the algorithm was then trained or fitted with data to learn the mapping model.

<u>Code segment explanation</u>

The code segment illustrate how the 'Fit ' and 'Predict' algorithms mentioned in Fig. 5.7a&b were implemented and most importantly the algorithm for the model as described in 3.5.4b. The important aspect of this code is to show how 'Fit' algorithm was implemented using class method called 'classify' while 'Predict' algorithm using 'classifyinstance' method. Besides, the code illustrates that the model algorithm was implemented as 'SVMclassifier' object that was defined using classs concept of python technology.

```python
#THIS CLASS IS FOR SVM CLASSIFIERS ONLY
class SVMRootclassifier():
        def __init__(self):
                #self.master = master
                self.value = None

        def loadCsv(self,filename):
                self.filename=filename
                lines = csv.reader(open(filename, "r"))
                dataset = list(lines)
                for i in range(len(dataset)):

                        dataset[i] = [float(x) for x in dataset[i]]

                return dataset

        #THIS CREATES A HIERARCHICAL MULTI-CLASSIFIER
        def classify(self,mainTree,trainingSet):
              svm = SVMRootclassifier()
              TreePredictors = {}

              self.trainingSet = trainingSet
              self.mainTree = mainTree

              trainset = []
              trainset = list(trainingSet)
              #Trainfile = list(trainingSet
              maintrees = svm.getMainTrees(mainTree)
              mKey = maintrees.keys()
              otherTrees = []
              AllTrees = []
              value = []
              mainTreePredictor = {}
              for mKeys, classTree in maintrees.items():
                      for mK, trees in classTree.items():
        #THIS CLASSIFIES A WHOLE DATASET
        def classifyInstance(self,classifier,classTree,data):
                svm = SVMRootclassifier()

                self.classifier = classifier
                self.classTree = classTree
                self.data = data

                tree = classTree
                testdata = data
                #mKey =list(TreePredictors.keys())
                #print('TreePredictors keys:',mKey)
                mKey =list(classifier.keys())
                #print('TreePredictors keys:',mKey)
                #TreePredictorTree = {}
                CellPredictorTree = {}
                NodePredictorTree = {}

                #predictiondata = data
                predictiondata = []
                correctClassified = {}
                incorrectClassified = {}
                predictionresult = -1
                for i in range(len(testdata)):
                                  X = testdata[i]
```

Figure 5.13: ML Algorithm class code segment

(3) Model class

The pickle class was used to implement this class directly where its damp method was used to store the model object while its load method was used to retrieve the object. Thus, the load method of pickle class was used as the implementation of 'Minterface' for making the model accessible to other subsystems. Fig. 5.14a &b presents the code segments for store and retrieve methods of the model class.

Code segment explanation

This code illustrates how the classifier object generated by the 'Fit' algorithm is stored (Fig. 5.14a) in a folder using the pickle class of the python technology. This enables this classifier to be copied and used elsewhere away from the training dataset environment. Hence Fig. 5.14b is a code segment that illustrates how the 'Predict' algorithm loads the classifier object from store.

```
#THIS CHOOSES AND STORES THE BEST CLASSIFIER
if result>bestaccuracy:
    bestaccuracy = result
    pickle_out = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\BNclassifier.pickle','wb')
    pickle.dump(classifier1,pickle_out)
    pickle_out.close()
```

Figure 5.14a: Model class store code segment

```
#RETRIEVE THE CLASSIFIER
pickle_in = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\SVMclassifier.pickle','rb')
tp=pickle.load(pickle_in)
```

Figure 5.14b: Model class retrieve code segment

5.1.4.3. Implementation of Dashboard subsystem

This provides a simplified mode for the user to interact with the system. Its implementation was conducted using a python graphical user interface class called tkinter. Tkinter is a python module for creating a rich graphical user interface. Two separate user interface classes indicated on the design were implemented in such a way to suit the requirements of three primary users of the system as illustrated in the use case model presented in Fig.5.2, and these are academia institutions, industry employers, and graduates. To address these needs, their functions as indicated in the use case model were portioned into primary and secondary ones, where 'RegisterAcdemicSector' and 'RegisterIndustryRole' are primary to 'Institution' and 'Employer' users respectively while the rest are secondary functions to all users.

Primary functions can only be executed by the specific target users while secondary any user can execute. Tabbed windows were adopted in the implementation of the multi-user interface where there is a tab window for each primary user and two subsidiary windows for viewing prediction results and learning/selecting the learning model. Fig 5.15a presents segment code for the implementation of the systemUI class as 'gui' class while Fig. 5.15b to 5.15f presents sample views windows of various user interfaces.

<u>Code segment explanation</u>

The code segment illustrates how the user interface of the model prototype was implemented. A tabbed window was implemented using GUI technology in python known as 'tkinter' also 'ttk'. The tabs were created using several layered 'frames' of 'ttlinter'. Each user of the prototype was given access to the model through a specific tab.

```
class gui:
        def __init__(self,root):

            self.rolVar = StringVar()
            self.Var = StringVar()
            self.secVar = StringVar()
            self.pathVar = StringVar()
            self.listVar = StringVar()
            self.saveVar = StringVar()
            self.editVar = StringVar()
            self.editVar.set('SAVE_NEW')
            #tab 1 begins from here INDUSTRY ROLES INDEX DETAILS
            tab1 = ttk.Frame(nb)

            list = [random.randint(1,100) for x in range(50)]
            list2 = []
            frametab1 = Frame(tab1, relief=RAISED, borderwidth=1)#MAIN FRAME FOR THIS TAB
            frametab1.pack(fill=BOTH, expand=1)

            frame1 = Frame(frametab1, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE LISTBOX

            frame2 = Frame(tab1, relief=RAISED, borderwidth=1)#SUBFRAME FOR COMMAND BUTTONS

            lblRHEADING = Label(frametab1, text="      INDUSTRY ROLES INDEX DETAILS")
            lblRHEADING.grid( row=1, column=20, columnspan=30, sticky=W, pady=4, padx=5)
            #tab 2 begins from here INSTITUTION INDEX DETAILS
            tab2 = ttk.Frame(nb)

            frametab2 = Frame(tab2, relief=RAISED, borderwidth=1)
            frametab2.pack(fill=BOTH, expand=1)


            frame1 = Frame(frametab2, relief=RAISED, borderwidth=1)
            frame2 = Frame(tab2, relief=RAISED, borderwidth=1)

            frame3 = Frame(frametab2, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE LISTBOX

            lblIHEADING = Label(frametab2, text="      ACADEMIA (INSTITUTIONS) INDEX DETAILS")
            lblIHEADING.grid( row=1, column=20, columnspan=30, sticky=W, pady=4, padx=5)
            #tab 3 begins from here GRADUATE INDEX DETAILS
            tab3 = ttk.Frame(nb)


            #list = [random.randint(1,100) for x in range(50)]

            frametab3 = Frame(tab3, relief=RAISED, borderwidth=1)#MAIN FRAME FOR THIS TAB
            frametab3.pack(fill=BOTH, expand=1)

            frame1 = Frame(frametab3, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE LISTBOX

            frame2 = Frame(tab3, relief=RAISED, borderwidth=1)#SUBFRAME FOR COMMAND BUTTONS

            lblGHEADING = Label(frametab3, text="  GRADUATE'S SKILLS INDEX DETAILS")
            lblGHEADING.grid( row=1, column=10, columnspan=30, sticky=W, pady=4, padx=5)

          #tab 4 begins from here  TRAINING WINDOW
            tab4 = ttk.Frame(nb)

            frametab4 = Frame(tab4, relief=RAISED, borderwidth=1)
            frametab4.pack(fill=BOTH, expand=1)

            frameA = Frame(frametab4, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE TEXTAREA
            frameB = Frame(frametab4, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE TEXTAREA
            frameC = Frame(frametab4, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE PROGRESSBAR
            frame2 = Frame(tab4, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE COMMAND BUTTONS

            #COMBOBOXES FOR SUBSECTOR, DATASETS, AND TRAINING ALGORITHMS
            lblTcombo1 = Label(frametab4, text="SUBSECTOR/DISCIPLINE")
            lblTcombo1.grid( row=3, column=10, sticky=W, pady=4, padx=5)

            self.comboTSECTOR=ttk.Combobox(frametab4,values=LISTSECTOR,state='readonly',textvariable = self.secVar)
            self.comboTSECTOR.bind("<<ComboboxSelected>>",self.ListDatasetMethod)
            self.comboTSECTOR.grid( row=4, column=10, sticky=W, pady=4, padx=5)
          #tab 5 begins from here PREDICTION RESULTS
            tab5 = ttk.Frame(nb)

            frametab5 = Frame(tab5, relief=RAISED, borderwidth=1)
            frametab5.pack(fill=BOTH, expand=1)

            frameA = Frame(frametab5, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE TEXTAREA
            frameB = Frame(frametab5, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE TEXTAREA
            frame2 = Frame(frametab5, relief=RAISED, borderwidth=1)#SUBFRAME FOR THE COMMAND BUTTONS


            #TEXTAREAS FOR SECTOR ROLES AND RESULTS PANE
            lblPNAMES = Label(frametab5, text="GRADUATE NAMES LIST")
            lblPNAMES.grid( row=4, column=0, sticky=W, pady=4, padx=5)
```

Figure 5.15a: GUI class code segment

1) Employer user Interface

Employer is the primary user of this user interface while the rest of the users are secondary users. The primary function is ‘RegisterIndustryRoles’ which was implemented through a number of menu items indicated on the screen shot. The rest of the users can only scroll through by clicking sector names and industry roles to view the underlying details.

2) Institution user Interface

Institution is the primary user of this user interface while the rest of the users are secondary users. The primary function is ‘RegisterAcademicSectors’ which were implemented through a number of menu items indicated on the screen shot. The rest of the users can only scroll through by clicking sector names, academic institutions and academic years to view the underlying details.

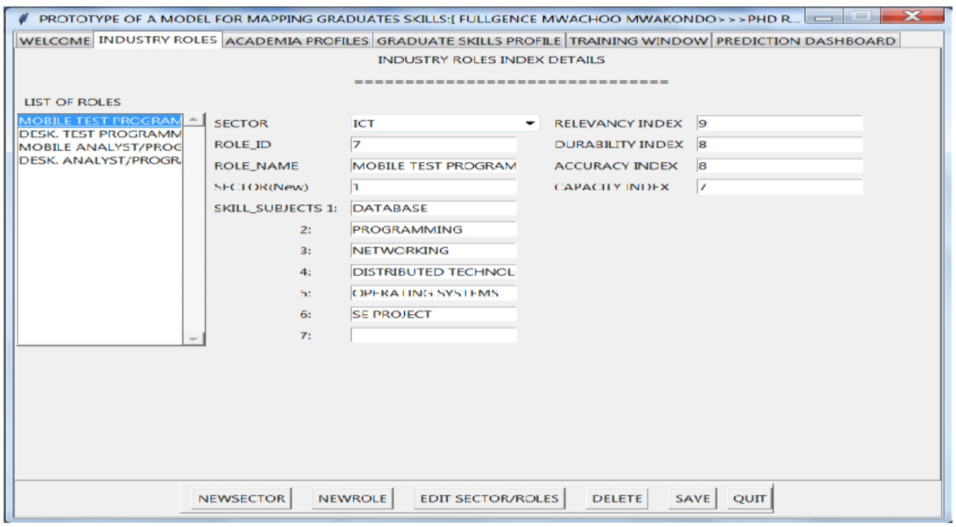


Figure 5.15b: Employer user interface

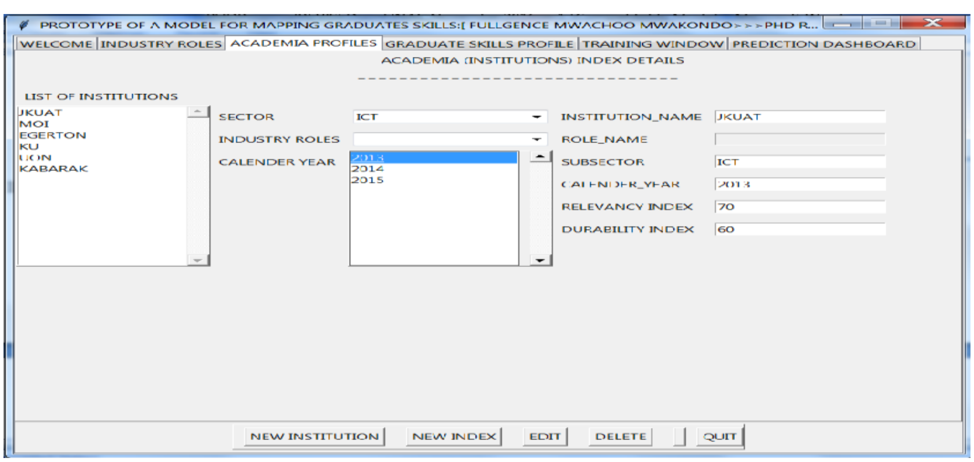


Figure 5.15c: Institution user interface

3) Graduate user interface

All are secondary users of this user interface. This is part of the secondary function of 'EvaluateGraduate' which was implemented through a number of menu items indicated on the screen shot. All the users can interact with this interface using the commands indicated on the screenshot.

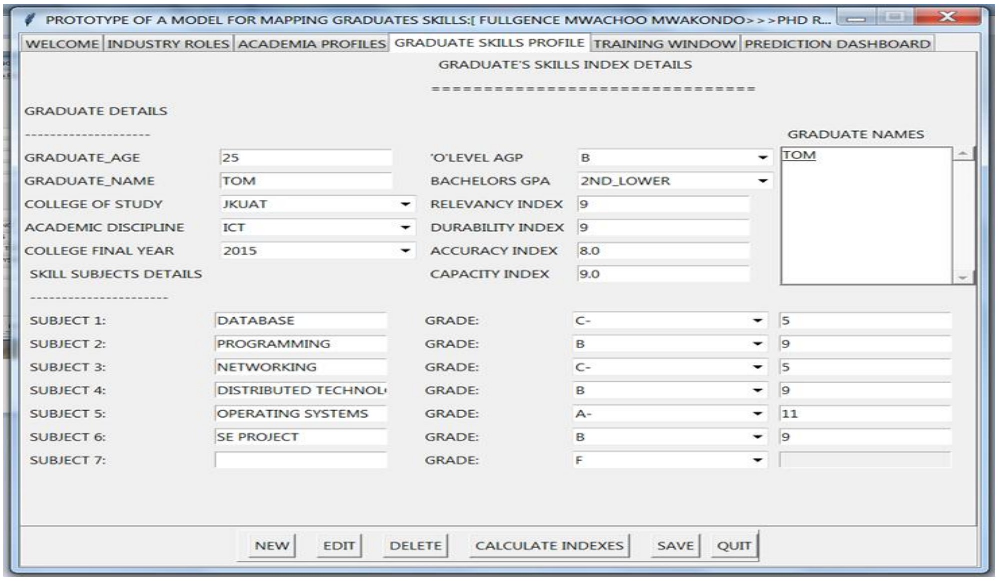


Figure 5.15d: Graduate user interface

4) Training and model selection user interface

This user interface was created specifically for system administrator as the primary user where the primary function is to 'RegisterIndustrySectorDatasets' implemented through menu items indicated on the screenshot. System administrator may be an employer regulator in the industry market. All other users of this user interface are secondary users. However, all the users can interact with this interface through clicking both sectors and training algorithms to select as well as using the 'train' commands indicated on the screenshot.

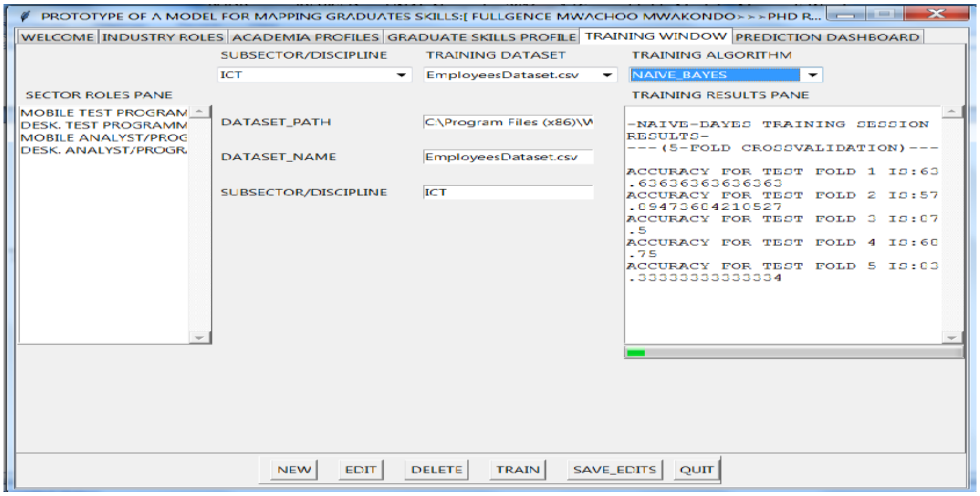


Figure 5.15e: Training and model selection user interface

5) Prediction results

All are secondary users of this user interface. This is part of the secondary function of 'EvaluateGraduate' which was implemented through a number of menu items indicated on the screen shot. All the users can interact with this interface using the commands indicated on the screenshot.

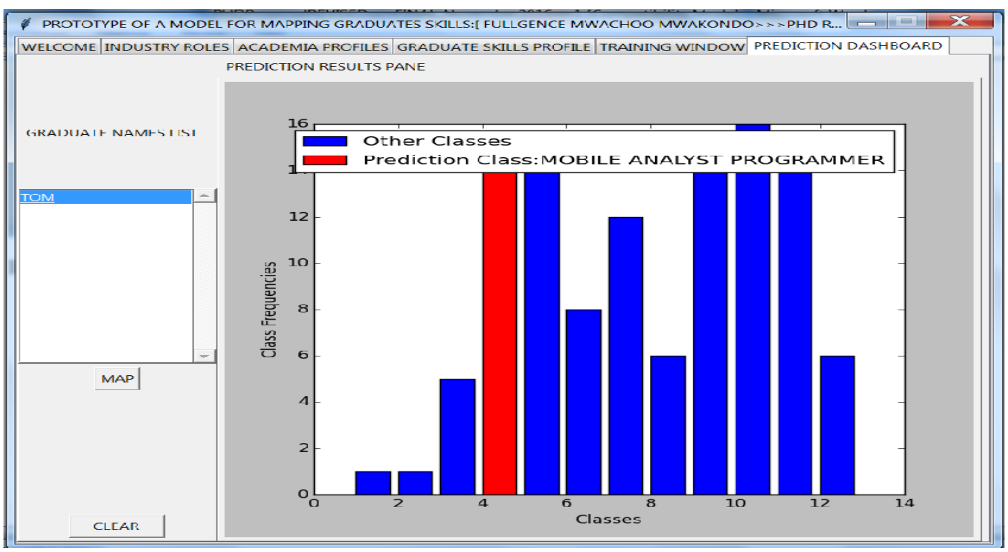


Figure 5.15f: Prediction results user interface

Finally, a number of python libraries were used in the implementation of the user interface. Visualization system that is capable of representing the data using graphical symbols was adopted and was implemented using a python library known as matplotlib. Matplotlib is a python module for data visualization capable of creating most kinds of charts, plots, and graphs and also rendering them on the screen using a canvas system. Data analysis feature that was capable of modeling the data was also adopted and was implemented using python library known as pandas. Pandas is a python module for data analysis which at the core of its data analysis has a powerful datasheet known as dataframe that is capable of modeling the data into rows and columns.

5.2. Computing and Development Resources

A number of computing resources were adopted and applied to produce the implementation of the software prototype at various points of design.

1) Hardware platform

Hewlett-Packard computer was used for the project and whose processor and memory specifications were Intel Core i5 CPU, 2,53 GHz speed and 4.0 GB of memory size.

2) Software operating system platform

A 64-bit Microsoft window’s operating software version 7 was the driving force behind the development platform providing the necessary computing resources such as storage.

3) Software development environment

A 64-bit Python software version 3.4.3 provided the development environment where both programming and database activities were realized ranging from the overall code editor, debugging, to testing. Python was identified as one of the most common platform for machine learning implementations with rich programming resources and was reviewed to determine how it could be used with additional technologies to implement the prototype. The following were some of the many python resources exploited during the development.

a. SQLite module was used to implement the database component of the prototype. SQLite is a version of SQL Technology that is already integrated in python as sqlite3 library class module. The sqlite3 class has several of its methods used, such as the connection method is a very useful in creating and accessing the implemented database
b. Python’s csv class module was used for implementation of the dataset for machine learning. The read method of the csv class was used to access and retrieve the dataset.
c. Python’s dictionary data structure was very useful in implementing the proposed taxonomical structure for the machine learning architecture.
d. Python’s svmpy class was used to implement SVM machine learning technique.

e. Python's numpy was very useful in implementing the naïve Bayes machine learning technique. This is a library for processing n-dimensional arrays.
f. Python's pickle library class was very useful in implementing the storage and retrieval of the generated machine learning models' objects.
g. Python's tkinter library class was used to implement the graphical user interface for the prototype.
h. Python's pandas' library class was used for numerical and statistical data analyses.
i. Python's sklearn library was used for both data preprocessing, feature selection and feature extraction purposes during machine learning. This library provided a number of essential algorithms that were key in implementing machine learning techniques such as cross-validation, naïve Bayes, support vector machines, Linear discriminant analysis, principal component analysis, scaler and many other algorithms.
j. Python's matplotlib library class was used for graphical data analyses especially in the production of high quality 2D graphics.

5.3. Summary

This chapter has outlined the methodology and software processes adopted in building the software prototype for the mapping model. An incremental methodology was adopted where four basic software processes were iteratively applied to generate the final prototype. The outcome of each process was emphasized with diagrams that illustrated the important aspects of the prototype. In summary, the chapter has demonstrated using the prototype that a prediction model for mapping graduates skills to industry roles in a practical way is feasible. Table 5.3 presents a summary of the main aspects of the mapping model's prototype and their implementation implication.

Table 5.3: Model 's design and implementation summary

Model's components and Design	Design Implementation	Software Development Resource
1.Data source subsystem -Database design model -Dataset design structure	- Database class - Dataset file	- python sqlite3 class - python's csv library class
2.Machine learning subsystem - Machine learning design model	- ML Algorithm class - Models class - Role class	- python's pandas, sklearn, matplotlib, svmpy, numpy library classes - python's pickle library class - Python's data dictionary
3.Dashboard subsystem - Dashboard design model - User Interface design model	- Gui class	- python's tkinter library class

CHAPTER 6: MODEL EVALUATION AND DISCUSSIONS

6.0. Introduction

This chapter presents a comprehensive description of evaluation results and discussion of the findings. The chapter has been organized into four sections as follows: Section 6.1 presents background to evaluation methods. Section 6.2 presents evaluation results using Research dataset (SE field data). Section 6.3 presents evaluation results using Benchmark dataset (SE literature data). Section 6.4 presents evaluation results using Validation dataset (AL field data). Section 6.5 provides a discussion and interpretation of the research findings. Finally, Section 6.5 concludes the chapter with a summary.

6.1. Background to Evaluation Methods

Evaluation in machine learning is needed to evaluate not only the ability of a classifier model (Lavesson, 2006) but also its generalization performance (Kahavi, 1995). There are many evaluation methods which have been categorized as either empirical (evaluate classifiers using portions of known data which have not been seen before by the classifier) or theoretical (evaluate classifiers using training data only or combined with other theoretical measures of generalization performance). We aimed to evaluate whether the model would perform well in the real world, and this required a portion of known data which had not been seen before by the classifier to provide a test situation that emulated the real world data.

As a result, our evaluation focused on empirical evaluation methods. Empirical methods divide the data into two subsets, training and test set, where training set is used to learn or generate the model while test set is used for evaluation its performance. Hence, we used training set to generate the classifier model and test set to test its performance. However, performance could be determined through a number of metrics. And, therefore, the type of performance metric used depends on the specific evaluation method employed (Lavesson, 2006).

We consider briefly some of the candidate evaluation methods and their types of performance metrics.

1) Vapnik Chervonenkis (VC) Evaluation Method

This is an evaluation method for algorithm that consists of a combination of theoretical measures of algorithm’s capacity to select the best classifier based on its inductive bias (also known VC dimension) and training error of a classifier generated by the algorithm (also known as empirical risk). This combination is called the VC bound which is the actual or expected risk, namely an estimate of the classifier's error on unseen instances of test data. This evaluation method is theoretical and depends on an algorithm with particular configurations. According to this method all algorithms have theoretical values calculated for VC dimension (Lavesson, 2006). Therefore, the evaluation metric for this method is VC bound.

2) Minimum Description Length (MDL) Evaluation Method

This is an evaluation method based on theoretical measure of classifier's complexity or simplicity which is related to classifiers length. Classifiers length is a theoretical indicator of existence of regularities in data that have been compressed using fewer symbols by the classifier than the symbols needed to describe the data literally. It is not only difficult to calculate the length of a classifier but also there is no guarantee that MDL will choose the most accurate classifier (Lavesson, 2006). Therefore, the evaluation metric is classifier's length.

3) Structural Risk Minimization (SRM) Evaluation Method

This is an algorithm evaluation method that is classifier dependent and based on VC dimension. The aim of SRM is to find a classifier with minimal empirical risk and low VC bound from a series of classifiers organized in structured subsets. The use of SRM is limited to algorithms with which one can create nested subsets of classifiers (Lavesson, 2006).

4) Bootstrap (BS) Evaluation Method

This is a classifier evaluation method that is based on statistical method of sampling with replacement where instances are sampled from the data to create the training set. To create a training set of size n involves sampling with replacement from the data n times. The instances that were never sampled are set aside for evaluation purposes. It is possible to have some instances repeated in the training set. This method is only suitable with large datasets. The main evaluation metric is the measure of performance, accuracy.

5) Cross Validation (VC) Evaluation Method.

This is an evaluation method that focuses on partitioning data into two mutually exclusive subsets, namely training and test set. The main evaluation metrics are measures of performance (accuracy/error) and measures of cost of misclassification (precision, recall, f1_score). There are several variants of CV, namely hold-out, leave-one-out(Jack-Knife), and k-fold cross-validations.

6.1.1. Choice of Evaluation Metrics and Method

According to Lavesson (2006), there are many evaluation metrics for measuring a variety of quality measures of a classifier, such as metrics for measuring performance (accuracy, errors), metrics for measuring complexity (VC dimensions), metrics for measuring similarity and misclassification cost (precision, recall, f1_score) or metrics for measuring sensitivity. Our choice of evaluation metric was based on the assumption that effective evaluation for job suitability of a graduate before employment improves not only performance but also productivity in the job.

Since misplacement of people in the job results into a negative impact such as low job satisfaction hence low productivity and high employee turnover, our aim was to get a model that should be able to place the right people in the right job (also known as accuracy). Therefore, our focus was to measure not only the classification accuracy but also misclassification cost of the classifier model. The risk or cost associated with misclassification errors can greatly harm not only the organization's productivity but also graduate's performance. Misclassification errors include placing either the right people in the wrong job (also known as false negative) or wrong people in the right job (also known as false positive). As a result our desired metrics for performance evaluation were accuracy, precision, and recall.

A number of evaluation methods were reviewed but only two turned out to be empirical, namely cross validation (CV) and bootstrap (BS). CV and BS have been studied widely and conclusions drawn indicate that while BS has high bias and low variance, CV has low bias and high variance which is the opposite of BS (Lavesson, 2006). High bias implies our model will not be complex enough to capture well the underlying pattern in the training data and hence will suffer from low performance on unseen data. CV provides a better technique for finding an acceptable bias-variance tradeoff than BS (Raschka, 2015).

The recommendation in literature has been CV with 10-folds as the standard (Kohavi, 1995). However, in situations where there is unequal class proportions stratified k-fold CV is better than the standard CV with 10-folds in yielding better bias-variance trade-off. Besides, stratified k-fold CV applies a resampling technique without replacement on the dataset that renders it the advantage of yielding a lower variance estimate of the model performance than other variants of CV. Since our focus was a model with low bias and low variance, and all our datasets had unequal class proportions then stratified 5-fold CV was a better option. 5-fold was adopted so as to ensure each class was represented in each fold through stratification in the data set where some classes had frequencies as low as 5.

6.1.2. Stratified K-fold Cross-Validation Evaluation Method

The method is appropriate where we have unequal class proportions in the dataset. It is a special form of K-fold CV method that uses a resampling technique without replacement to partition the dataset into several mutually exclusive subsets where all are used for learning the classifier except one subset that is used for evaluation purposes. The training and evaluation are repeated until all subsets have been used once for evaluation purposes. Each subset is called a fold and to ensure that each class is properly represented in each fold a special configuration called stratified folds is employed. It is widely used in machine learning due to its ability to yield better bias-variance trade-off. Its main evaluation metrics are measures of performance (accuracy/error) and measures of cost of misclassification (precision, recall, f1_score).

6.1.3. Evaluation Metrics

One important source of information for deriving accuracy, precision and recall values was noted as the confusion matrix. A confusion matrix was defined as an n by n matrix, where n is the number of classes, which displays the number of correct and incorrect predictions made by the model compared with the actual classification in the test data.

6.1.3.1. Accuracy

This is the probability of a classifier to correctly classify a randomly selected instance (Kohavi, 1995). It is the most widely used measure for performance currently in practice (Lavesson, 2006). In the present study, accuracy was used to capture the average and the best performance of the classifier model under cross validation evaluation. A single accuracy estimate is meaningless without confidence (Kohavi, 1995) about quality of its performance. In the present study, accuracy was used to measure performance of classifier model generated.

6.1.3.2. Precision

The precision is the ratio TP/(TP + FP) where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0. In the present study, precision was used to conduct further investigation to reveal the performance quality of the model.

6.1.3.3. Recall

The recall is the ratio TP/(TP + FN) where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0. In the present study, recall was used to conduct further investigation to reveal the performance quality of the model.

6.1.3.4. F1-score

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

F1 = 2 * (precision * recall) / (precision + recall)

In the multi-class and multi-label case, this is the weighted average of the F1 score of each class.

In the present study, f1_scores were used to conduct further investigation to reveal the performance quality of the model.

6.2. Experimental Evaluations Results

We adopted stratified 5-fold cross validation. The aim was to evaluate whether the model would perform well in the real world, and this required a portion of known data which had not been seen before by the classifier to provide a test situation that emulated the real world data. As a result, using accuracy alone as the performance metric would have only indicated the general performance of the model to correctly predict class labels over all predictions, but would not have given enough information on the quality of the model towards critical or important classes. And that was why precision, recall, and f1_score were used to conduct further investigation to reveal the performance quality of the model.

One experiment was repeated on three datasets to evaluate performance of the model. Although our main focus was to evaluate the SVM model selected in chapter 4, we felt necessary to further monitor its behavior by comparing with the naïve Bayes model. This was to confirm beyond reasonable doubt about its capacity. Table 6.1 illustrates the planning of the experiment while the sections that follow present details of evaluation results.

Table 6.1: Evaluation Experiment design

	Conception/ Objective	Design	Preparation & Execution	Analysis
Experiment D	To evaluate performance and validity of the machine learning model	**1.Experimental units**: Graduate Employees' skills **2.Experimental subjects**: ML models **3. Dependent variable**: accuracy,precision,recall,f1-score **4. Independent variables:** feature subsets	1.Split dataset into three: Training set, Validation set, Testing set 2.Apply 5-fold cross validation 3.Apply 6-10 iterations	**Evaluate model using three datasets** **-**compare performance, per class, per level and across other models in literature Approach : Hypothesis testing Technique : Paired sample T Test Test variable: Machine learning technique Significance value: 0.05

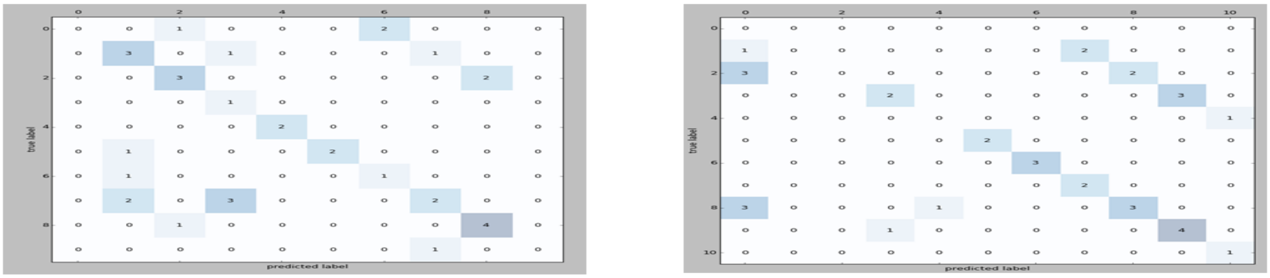
6.2.1. Experimental Evaluation using Software Engineers Field Data (Research Dataset)

Initially, the dataset had 113 instances but two classes (1 & 2) had sizes of only one, so they were dropped. The remaining 111 instances were split into two, training and test set, in the ratio of 80:20. Stratified random sampling was applied to ensure each was represented. This resulted with a test set size of 28 instances (about 25%). Table 6.2.1a presents the class distribution of the test set derived from the Research dataset (SE field data). From Table 6.2.1a it is clear that class sizes were imbalanced in the original dataset. The training set was subjected to 5-fold cross-validation where 5 instances of classifier models were generated. One instance of classifier model with the best training results was selected for subsequent evaluation that generated a confusion matrix for the model.

From the confusion matrix various performance metrics' values were extracted such as accuracy, precision, recall, and f1_score. This experiment was repeated for each induction algorithm, namely naïve Bayes and SVM, hence two models. Fig. 6.2.1a presents graphical results showing confusion matrix for the two models while Fig.6.2b presents bar graph results showing comparative analysis of the performances of the two classifier models along the four evaluation metrics. For both models' results accuracy and recall values seemed to be equal. However, SVM classifier model was in all aspects better than naïve Bayes model as expected and in fact its precision seemed to be the highest. Further analysis was conducted to confirm whether these performance differences between the two models were significant.

Table: 6.2.1a Class distribution of test set for the SE field dataset (Research dataset)

Class	3	4	5	6	7	8	9	10	11	12	**TOTAL**
Size	1	3	3	2	3	2	3	4	3	2	28



Naïve Bayes model's confusion matrix

SVM-model's confusion matrix

Figure 6.2.1a: Confusion matrices for Naïve Bayes and SVM models using (Research dataset) dataset1

The confusion matrix for each model in Fig. 6.2.1a shows classes that were correctly classified along the principal diagonal while the classes below were falsely classified as correct and the classes above were falsely classified as incorrect.

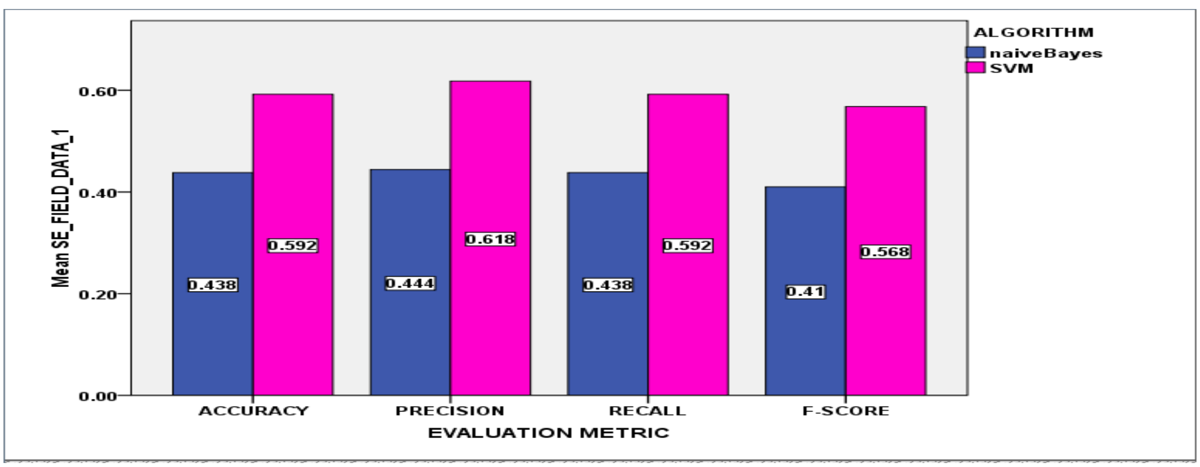


Figure 6.2.1b: Bar graph comparative analysis of the two models using (Research dataset) dataset1

Testing whether the differences between the two models were significant

The aim of this investigation was to find out whether performance difference between the two models was real. Therefore, the focus of this test was between naïve Bayes and SVM, hence two paired variables. Fig.6.2.1b indicates a potential difference between the two along all performance metrics. A paired sample T test was conducted to test the hypothesis that model performance difference was not significant. For this type of test to be valid, conditions for tests were checked (homogeneity and normality of data). Table 6.2.1b presents results based on 10 iterations of 5-fold cross-validation evaluation where we rejected the hypothesis at p=0.05. The results indicate the difference was real and significant.

Table 6.2.1b:Paired Sample T Tests for Model Evaluation using SE field dataset (Research)

	Pair	Paired differences					t	df	Sig(2-tailed)	RESULT
		Mean	Std. dev.	Std. error mean	95% confidence interval for difference					
					lower	upper				
Pair 1	accuracyNB_R - accuracySVM_R	-.017	.14135	.04470	-.1181	.0841	-.380	9	.713	*REJECT*
Pair 2	precisionNB_R - precisionSVM_R	-.077	.17366	.05492	-.2012	.04723	-1.402	9	.194	*REJECT*
Pair 3	recallNB_R - recallSVM_R	-.017	.14135	.04470	-.1181	.08411	-.380	9	.713	*REJECT*
Pair 4	fscoreNB_R - fscoreSVM_R	-.057	.15370	.04860	-.1669	.05295	-1.173	9	.271	*REJECT*

Based on these results it was clear that SVM model was the best as expected. Further analysis of its performance per class was also investigated. Fig.6.2.1c presents bar graph results showing performance per class of the selected classifier model.

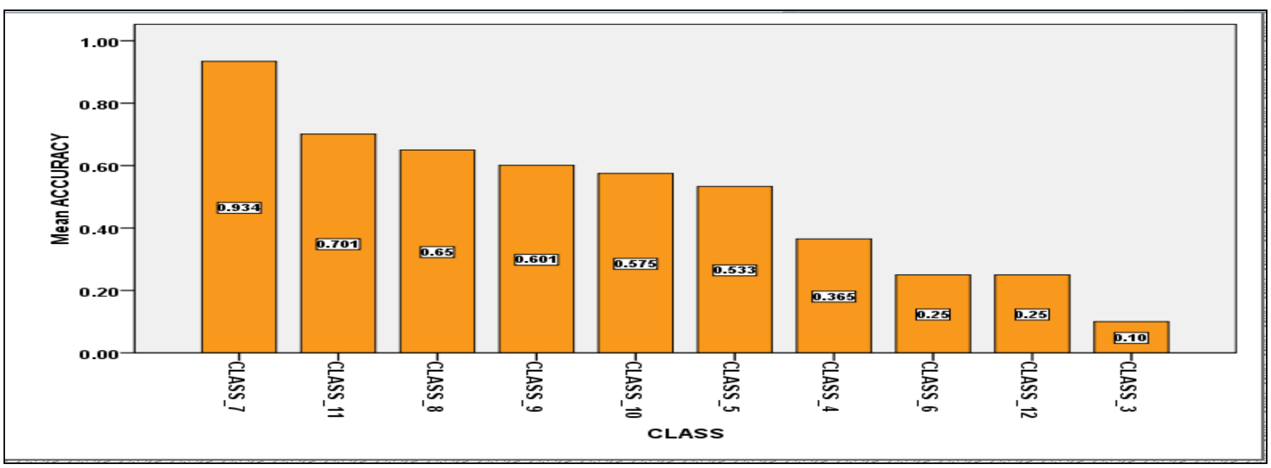


Figure 6.2.1c: Class performance accuracies of the selected model using (Research dataset) dataset1

Findings #12

Fig.6.2.1b reveals that SVM classifier model was significantly the best (accuracy=59.2% against 43.8% for naïve Bayes) for this dataset (as expected from chapter 4) and its performance per class was fairly good in some classes (class7 =93.4%) and fairly poor in other classes (class3=10%). However, its ability not to label negative classes as positive was more or less the same as its ability to find all positive classes correctly (precision =.61.8%, recall = 59.2%).

6.2.2. Experimental Evaluation using Software Engineers Benchmark Dataset (Literature)

The dataset had 279 instances which were split into two, training and test set, in the ratio of 80:20. Stratified random sampling was applied to ensure each class was represented. This resulted with a test set size of 60 instances (about 21%). Table 6.2.2a presents the class distribution of the test set derived from the Benchmark dataset (SE literature data). From Table 6.2.2a it is clear that class sizes were also imbalanced in the original dataset. The training set was subjected to 5-fold cross-validation where 5 instances of classifier models were generated.

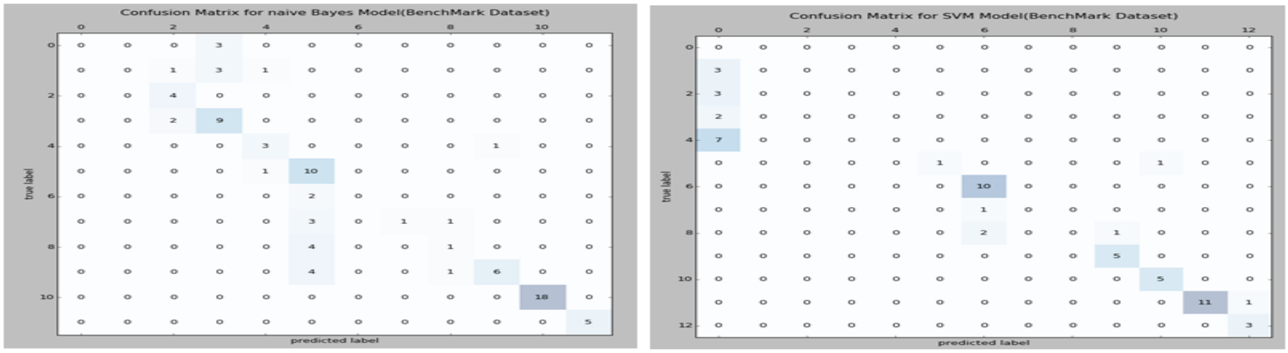
One instance of classifier model with the best training results was selected for subsequent evaluation that generated a confusion matrix for the model. From the confusion matrix various performance metric values were extracted such as accuracy, precision, recall, and f1_score. This experiment was repeated for each inducer algorithm, namely naïve Bayes and SVM, hence two models. Fig.6.2.2a presents graphical results showing confusion matrix for the two models while Fig.6.2.2b presents bar graph results showing comparative analysis of the performances of the two classifier models along the four evaluation metrics.

For both models' results, accuracy and recall values seemed to be equal. However, SVM classifier model was in all aspects similar as naïve Bayes model. In fact, further analysis was conducted to confirm whether this observation between the two models was significant.

Table: 6.2.2a Class distribution of test set for the Benchmark dataset

Class	1	2	3	4	5	6	7	8	9	10	11	12	**TOTAL**
Size	2	3	5	8	3	9	2	2	4	7	11	4	**60**

The confusion matrix for each model in Fig. 6.2.2a shows classes that were correctly classified along the principal diagonal while the classes below were falsely classified as correct and the classes above were falsely classified as incorrect.



a) Naïve Bayes Model's confusion matrix

b) SVM Model's confusion matrix

Figure 6.2.2a: Confusion matrices for Naïve Bayes and SVM models using Benchmark dataset (dataset2)

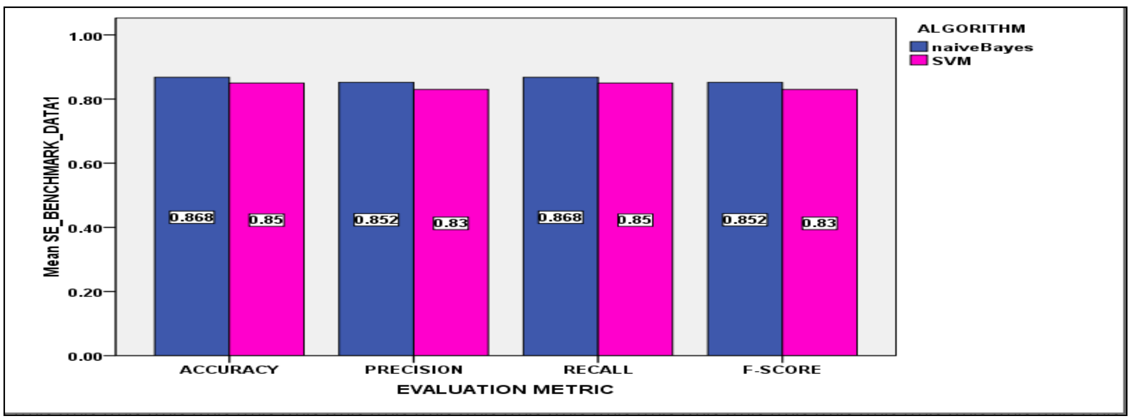


Figure 6.2.2b: Bar graph comparative analysis of the two models using (Benchmark dataset) dataset2

Testing whether there were any significant differences between the two models

The aim of this investigation was to find out whether there was any performance difference between the two versions of the model. Therefore, the focus of this test was between naïve Bayes and SVM, hence two paired variables. Fig.6.2.2b indicates a potential of no difference between the two along all performance metrics. A paired sample T test was conducted to test the hypothesis that model performance difference was not significant. For this type of test to be valid, conditions for tests were checked (homogeneity and normality of data). Table 6.2.2b presents results based on 10 iterations of 5-fold cross-validation tests where we accepted the hypothesis at p=0.05. The results indicate there was no difference that was significant.

Based on these results the model selected in chapter 4, namely SVM model, was further analyzed of its performance per class in the current dataset. Fig.6.2.2c presents bar graph results showing performance per class of the selected classifier model under the current dataset.

Table 6.2.2b:Paired Sample T Tests for Model Evaluation using Benchmark dataset

	Pair	Paired differences					t	df	Sig(2-tailed)	RESULT
		Mean	Std. dev.	Std. error mean	95% confidence interval for difference					
					lower	upper				
Pair 1	accuracyNB - accuracySVM	-.130	.03333	.01054	-.1538	-.1061	-12.33	9	.000	*ACCEPT*
Pair 2	precisionNB - precisionSVM	-.140	.05312	.01680	-.1780	-.1020	-8.334	9	.000	*ACCEPT*
Pair 3	recallNB - recallSVM	-.130	.03333	.01054	-.1538	-.1061	-12.33	9	.000	*ACCEPT*
Pair 4	fscoreNB - fscoreSVM	-.152	.04492	.01420	-.1841	-.1198	-10.70	9	.000	*ACCEPT*

Figure 6.2.2c: Class performance accuracies of the selected model using (Benchmark dataset) dataset1

Findings #13

Fig.6.2.2c reveals the performance per class of the selected model, namely SVM model, was excellently good in some classes (100% accuracy for 4 classes) and fairly poor in other classes (5% accuracy for class7). However, its ability not to label negative classes as positive was more or less the same as its ability to find all positive classes correctly which were equally excellent (precision=83%, recall=85%).

6.2.3. Experimental Evaluation using Academic Librarians' Field Data (Validation Dataset)

The aim of this investigation was to find out the behavior of our classifier model in other related areas different from SE with an ultimate goal to evaluate its applicability across domains. Table 6.2.3a presents 5-fold cross-validation tests results of the model performance in the current dataset where SVM model was run with parameter settings adopted in dataset1 (section 6.2.1). However, SVM model needed parameter tuning within the current dataset.

Table 6.2.3a: Model Evaluation performance using AL dataset

	Naïve Bayes	**SVM (0.1,1000)**
	5-feature	5-feature
F1	62.5	62.5
F2	71.4	71.5
F3	66.6	50.0
F4	33.3	66.6
F5	50.0	50.0
Mean	**56.7**	**60.1**

Findings #14

Table 6.2.3a reveals that in the current dataset (validation dataset) 5 feature subset, similar to the ones selected in dataset2 (benchmark), SVM induced the best results for the model (60.1%) as expected.

6.2.3.1. Parameter Selection using Academic Librarians' Dataset (Experiment B)

Both gamma and complexity values were varied in a range of {0.00001 to 1} and {0.00001 to 10000} incrementally in multiples of 10. Table 6.2.3b indicates gamma value of 0.1 and complexity values of at least 10 appeared to be optimal. Fig.6.2.3a presents results of a validation curve(a) that confirmed these findings.

Finding #15

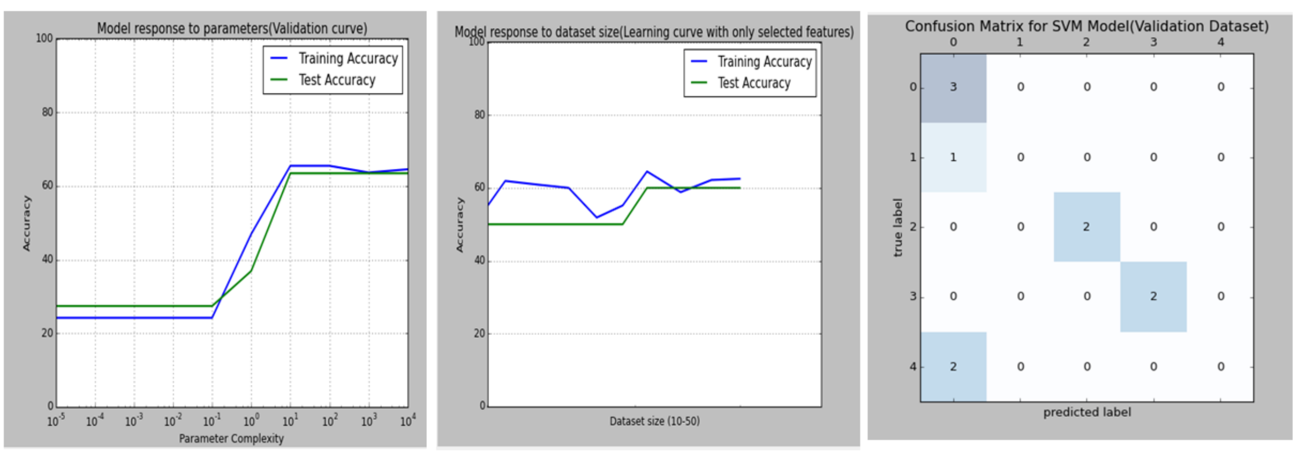
Table 6.2.3b reveals that in the current dataset parameter settings that induced the best performance in the model (63.4%) were gamma=0.1 and complexity of at least 10.

Table 6.2.3b: Analysis of relevant parameter values in AL dataset

	Complexity									
gamma	**0.00001**	**0.0001**	**0.001**	**0.01**	**0.1**	**1**	**10**	**100**	**1000**	**10000**
0.00001	27.3	27.3	27.3	27.3	27.3	27.3	27.3	27.3	27.3	45.2
0.0001	27.3	27.3	27.3	27.3	27.3	27.3	27.3	27.3	41.9	60.1
0.001	27.3	27.3	27.3	27.3	27.3	27.3	27.3	39.4	60.1	53.9
0.01	27.3	27.3	27.3	27.3	27.3	27.3	39.4	60.1	63.4	60.1
0.1	27.3	27.3	27.3	27.3	27.3	39.4	63.4	63.4	63.4	63.4
1	27.3	27.3	27.3	27.3	27.3	42.2	45.5	42.2	50.5	51.4

6.2.3.2. Estimating generalization error of the model using Academic Librarians' Dataset (Experiment C)

Generalization performance of the model was studied using different dataset sizes at increments of 10. Fig.6.2.3a presents results of a learning curve (b) that indicate the generalization error of the selected model, namely SVM model, progressively reduced as sample size increased. This was an indication that the classifier model was able to generalize its performance very well in the current dataset.



a) Validation Curve b) Learning Curve c) Confusion Matrix

Figure 6.2.3a: Learning performance behavior of selected model using (Academic Librarians' dataset) dataset3

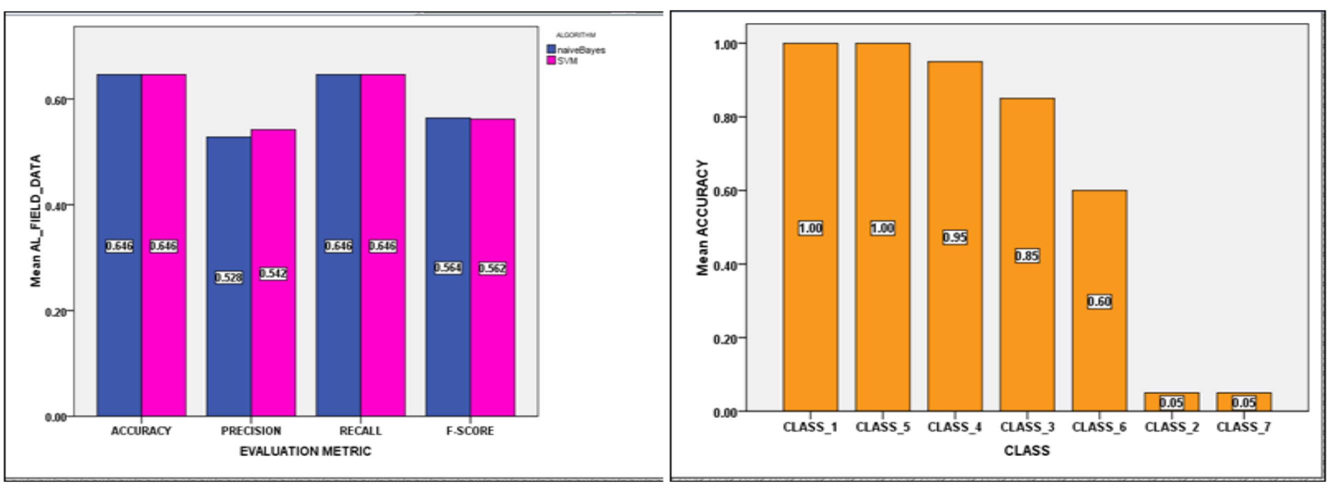
6.2.3.3. Evaluating model using Academic Librarians' Test set (Experiment D)

The evaluation dataset had 50 instances which were split into two, training and test set, in the ratio of 80:20. Stratified random sampling was applied to ensure each class was represented. This resulted with a test set size of 13 instances (about 26%). Table 6.2.3a presents the class distribution of the test set derived from the Validation dataset (Academic Librarians' field data). From Table 6.2.3a it is clear that class sizes were also imbalanced in the original dataset. The training set was subjected to 5-fold cross-validation where 5 instances of classifier models were generated.

One instance of classifier model with the best training results was selected for subsequent evaluation that generated a confusion matrix for the model. From the confusion matrix various performance metric values were extracted such as accuracy, precision, recall, and f1_score. This experiment was repeated for each induction algorithm, namely naïve Bayes and SVM, hence two models. Fig.6.2.3a presents graphical results showing confusion matrix (c) for the two models while Fig.6.4b presents bar graph results showing comparative analysis of the performances of the two classifier models along the four evaluation metrics. For both models' results, accuracy and recall values seemed to be equal. However, SVM classifier model was in all aspects similar as naïve Bayes model. This behavior of the models was also observed when evaluating with the benchmark dataset.

Table: 6.2.3c Class distribution of test set for the AL dataset

Class	1	2	3	4	5	6	7	TOTAL
Size	2	2	2	2	2	1	2	13



a) Model's Performance Comparison

b) Class performance of selected model

Figure 6.2.3b: Class performance accuracies of the selected model using (AL dataset) dataset3

Findings #16

Fig.6.2.3b reveals that the two models seemed to be equally the same along most performance metrics except precision where SVM model (0.542) seemed to outdo naïve Bayes model (0.528) as expected. The performance per class of the selected model, namely SVM model, was excellently good in some classes (100% for classes1&5) and fairly poor in other classes (5% for classes 2&7). However, its ability not to label negative classes as positive was not as good as its ability to find all positive classes correctly which was equally good (precision = 54.2%, recall = 64.5%).

6.3. Comparative analysis

Comparative analysis was necessary to reveal not only the dataset in which the model performed best but also hierarchical level as well as class where the model performed best. Table 6.3a presents performance results across the three datasets while Table 6.3b presents performance results along hierarchical levels across the three datasets. In each case, the model reported equal performance in both accuracy and recall. On average, model performance seemed to improve upward the hierarchy levels.

Table 6.3a: Comparison of performance across three datasets

Performance Metric	Research Dataset	Benchmark Dataset	Validation Dataset	Mean
accuracy	0.59	0.85	0.65	**0.69**
precision	0.62	0.83	0.54	**0.66**
recall	0.59	0.85	0.65	**0.69**
F1_score	0.57	0.83	0.56	**0.65**

Table 6.3b: Comparison of performance along hierarchical levels across datasets

	Research Dataset		**Benchmark Dataset**		**Validation Dataset**		
level	**classes**	**average**	**classes**	**average**	**classes**	**average**	**Mean**
1	7,8	0.79	1,2,7,8	0.53	4,5	0.98	**0.77**
2	3,4,9,10	0.41	3,4,9,10	0.95	3,6	0.73	**0.69**
3	5,6,11,12	0.43	5,6,11,12	0.82	1,2,7	0.37	**0.54**
Mean		**0.54**		**0.77**		**0.69**	**0.67**

Model's performance seemed to be very high in the benchmark dataset as a result of having more instances whose classes had very high accuracies (class 10 & 11) and fewer instances whose classes had very low accuracies (class 7 & 8). This was not the case with other two datasets where a distribution difference of classes with very high and very low accuracies was not high. Explanation behind this could be differences in sources of data and their data collection techniques. While our data was collected through questionnaires and from the Kenyan population, the benchmark data was collected through a carefully designed assessment tool that improves the accuracy of the data.

Table 6.3c: Reported performance across other related models in literature

Performance accuracy	Current (2017) {Industry roles}	Clare & King (2003) {Proteins}	Barbedo & Lopes (2006) {Music}
Type of model	SVM	Decision Tree	K-NN
Number of datasets experimented with	3	12	1
Reported performance accuracy			
Level 1	77	56.4	87
Level 2	69	46.3	80
Level 3	54	23.1	72
Level 4	-	7.9	61
Average per level	67	33.4	75
Reported evaluation	69	53.3	61

Findings #17

Table 6.3b reveals that model performance depends on the distribution differences in the dataset of class instances with very high and very low accuracies. In Benchmark dataset where performance was 85%, high accuracy (100%) class (class11) had the highest number of instances (size=11) while low accuracy (5%) class (class7) had the lowest number of instances (size=2). In Research dataset where performance was 59%, high accuracy (93.4%) class (class7) had moderate number of instances (size=3) while low accuracy (5%) class (class3) had moderate number of instances (size=1). In Validation dataset where performance was 65%, high accuracy (100%) class (class1&5) had moderate number of instances (size=2) while low accuracy (5%) class (class2&7) had moderate number of instances (size=2).

Model performance in both Research and Validation datasets seemed to be fairly good (59% and 65% respectively). Model performance seemed to improve upward in the hierarchical levels of the taxonomy consistent with other related models in literature.

6.4. Discussion of Evaluation Findings

Evaluation results and findings were crucial not only in establishing the best generalization performance and the best classifier model but also in understanding the behavior of the classifier model from what was expected and how it compared with other models in literature. Besides, these findings were eventually used to answer the last research question: how do we evaluate performance and validity of the mapping model?

6.4.1. The best generalization performance of the classifier model

Findings #12, #13, #14, & #15 were crucial in establishing the best generalization performance of the classifier model using the selected inducer algorithm. According to findings #13 & #14 the best classifier model seemed to show excellent performance behavior (along all four performance metrics adopted) in all the three datasets. However, it performed differently in the first dataset (research) and approximately similar performance behavior in the second (benchmark) and third (validation) datasets. This could possibly be attributed to class distribution differences where dataset2 and 3 have more or less similar distribution of 'bad' (less) and 'good' (more) classes, unlike in dataset 1 where 'bad' classes are more than 'good' classes. In this case, classes with very high accuracies are 'good' while classes with very low accuracies are 'bad'.

The best generalization performance was calculated as an average performance across the three datasets as indicated in Table 6.3a&b. In this case, along hierarchical levels the best average accuracy performance of the model was 67% and, therefore, we can confidently claim that the best performance of our model was 67%.

6.4.2. To compare model performance under two industry domains

Likewise, to compare model's performance findings 12# and #16 equally played an important role. The two results of the classifier model seemed to show more or less similar performance behavior (along all four performance metrics adopted). Precision as a measure of the ability of the model not to label a negative outcome as positive is a very crucial measure of the classifier model. This is because the original goal of the study was to build a model that would improve productivity and performance in the job.

Table 6.4 presents comparative results for the model contrasting the two cases as extracted from SVM model results in Fig. 6.2.1b and 6.2.3b. These results indicate that in both cases the precision values were good (SE case precision = 61%; AL case precision = 54%). The two findings signal strongly the confirmation and acceptance of the hypothesis posed in research question four: **H_{04A}: There is no significant performance difference of the model in different industry domains**

Low precision would have meant the model would be prone to act against this objective. This was enough reason to confirm SVM model as the best classifier.

Besides, along the three datasets SVM model was able to show a consistent superior performance over naïve Bayes model along all the performance metrics. There was clear evidence that SVM model performance per class was 100% accurate for about 30% of the target classes in a dataset, namely 28% in dataset2 (2 out of 7 classes) and 33% in dataset3 (4 out of 12 classes).

Table 6.4: Comparison of performance measures in each Case in the study

Variable	Case 1: Software Engineers	Case 2: Academic Librarians
Accuracy	0.59	0.64
Precision score	0.61	0.54
Recall score	0.59	0.64
F-score	0.59	0.56

6.4.3. Performance Comparison with other Classifier Models in Literature

Our model seemed to compare well with other hierarchical classifier models in literature. Model performance seemed to improve upward in the hierarchical levels of the taxonomy consistent with other models in literature (Clare & King, 2003; Barbedo & Lopes, 2006). Based on the model's performance as revealed in findings #17, average level performance across the three datasets was better than average level performance of Clare & King's model (2003) across 12 datasets. However, the model's average level performance was slightly lower compared to Barbedo & Lopes's model (2006). Although, Barbedo & Lopes's model results (2006) were based on only one dataset which we did not know its distribution.

Nevertheless, in the present study the best average performance achieved by the model was in dataset2 (benchmark) whose results were much better compared to Barbedo & Lopes's model (2006). Although there was no evidence whether there was use of hierarchical approach, Shashidhar *et al.* (2015) built a classifier model using the same Benchmark dataset and achieved a performance of 82% which was slightly lower compared to the performance level achieved using the same Benchmark dataset (85%) by the classifier model produced in the current study.

6.5. Discussions Conclusion of Evaluation Findings

The main focus of this discussion was not only to demonstrate the appropriateness of the classifier model to serve its purpose by evaluating its performance but also to assess the validity of the classifier model by comparing its evaluation results across other related models in literature. From this discussion it was clear that the appropriate performance of was achieved using SVM model at an average accuracy of 67% across three datasets. Its performance on three datasets was fairly good 59% (SE field data), 65% (AL dataset) and 85% (Benchmark dataset was better)

The validity of the model was demonstrated experimentally where the results of the model performance under different industry domains were compared. Lastly, the discussion also clearly demonstrated comparative performance of the classifier model against other literature models, especially hierarchical models, was considerably better than most of them. These findings were crucial in providing answer to the fourth research question: **RQ4: How do we evaluate performance and validity of the mapping model?**

Table 6.5 presents a summary of the outline research method that contributed towards answering the research.

Table 6.5: Method followed to answer research question 4

METHOD	FINDINGS
1. Building model's prototype	Obtained prototype of the mapping model
2. Evaluation of prototype using data collected and benchmark dataset	Obtained average generalization performance of 67% for the model
3. Evaluation of prototype using only benchmark dataset [Experiment]	Obtained average generalization performance of 85% for the model
4. Comparison of prototype results on benchmark dataset with related models results on same benchmark dataset	Obtained better results :85% (current model) against 82% (related model) (**OUTCOME**)
5. Reporting related performance in other non-industry role domains	Obtained : 53.3% on protein dataset (Clare & Kings, 2003) and 61% on music dataset (Barbedo & Lopes, 2006) (**OUTCOME**)

6.6. Discussion of Results Validity

The results presented for discussion in this section have been carefully planned and generated under the assumption that credibility of any study and its findings depends on not only the research methodology applied but also the validity of its results. Therefore, it would be improper to discuss the results without assessing how valid the results are. Wohlin *et al* (2003) highlights four types of results validity concerns and observes that it is important to assess how valid the results are before they are presented for discussion. The four types are internal validity, external validity, construct validity and conclusion validity. This section outlines each one of them and how they have been addressed in the current study.

6.6.1. Internal Validity

This kind of validity was concerned with factors that might have affected the dependent variable without the researcher's knowledge (Wohlin *et al*.,2003). In the current study, several factors that would have affected the results outside the original four dependent variables considered in the conceptual framework were noted during the experiments. These included 'age' (this refers to age of the graduate), 'university' (this refers to the University of Study for the graduate), and 'bachelor 's degree' (this refers to the type of degree program the graduate enrolls).

However, one more factor that would have affected the results adversely was variation of industry roles' definition in various industry firms where some roles' definitions would have either similar names but different requirements or requirements elements from more than one role (Chien & Chen, 2008). To address this issue, two frameworks (Fig.3.5,3a & 3.5.3b in section 3.5) that were designed to harmonize role names and role boundaries were adopted and are part of the contribution of this study. Before the frameworks were applied as described in section 3.5 of research methodology, there was need to maximize intra-role similarity and minimize inter-class similarity with the aim to avoid the model under-fitting the data (Raschka, 2015; Chien & Chen, 2008).

A case is comparative in nature where there is contrasting of results generated from either case. To avoid bias and ensure internal validity, a valid basis for assessing the results was adopted and this involved organizing the study in a way that facilitated comparison of results. Three common strategies for organizing a study to facilitate this comparisons consists of comparing results using 1) sister projects, 2) company baseline projects, and 3) differently treated components of a single project (Kitchenham & Lesley, 1995). Alternative occupational domains and benchmark studies in literature were closely related to these strategies, and thus the present study employed strategy 1&2 to facilitate comparative analysis of results. Table 6.4 presents comparative results for the model contrasting the two cases as extracted from SVM results in Fig. 6.2.1b and 6.2.3b.

6.6.2. External validity

This was concerned with ability to generalize results of the experiments over the entire target population. Again, Wohlin *et al.* (2003) raises concerns that the problem and the participants in the study have to be representative of the target population for the results to reach the threshold for generalization. Clearly, this was an issue of research design. The overall research question of this study was answered through a case study research design. Case studies have been known to be generalizable to theoretical propositions and not to population universes, because they do not represent a sample (Yin, 1994).

However, the choice for this design was driven by not only the explanatory nature of the question but also the contemporary nature of the event where the relevant behaviors could not be manipulated. To address this issue, multiple cases approach was adopted where a case of software engineers was used as the primary case to produce the research dataset while a case of academic librarians was used as a secondary case to generate the validation dataset. In the present study, multiple case approach offered greater validity to the case study findings because each case was considered as a replication that was used to confirm the findings consistency for generalization (Easterbrook *et al.*, 2007). The adoption of case approach was also important in avoiding the scale-up problem (Kitchenham & Lesley, 1995).

The biggest challenge was in the selection of the cases, because case samples are not based on variables that are manipulated but on variables that represent typical situations and this was central to the issue of external validity for this study (Kitchenham & Lesley, 1995). Common approach adopted consisted of describing cases based on significant characteristics and using these as state variable information to select a case. Demographic characteristics results of Table 4.1.1a and Table 4.2.4 helped to identify four characteristics to represent typical cases, namely gender (variation), bachelors degree (several types), University of study (at least one) , and industry roles (several). Table 6.6.1 presents description of the two typical cases.

6.6.3. Construct validity

This was concerned with the relationship between the concepts and theories behind the study and what was measured and affected (Wohlin *et al.*, 2003). Construct validity tried to establish correct operational measures for the concepts being studied (Yin, 1994). This involved defining concepts clearly before measurements were conducted and justifying measures adopted for such concepts. This study derived its concepts from three existing models for training evaluation that served as its theoretical framework as explained in section 2.7. The justification of each framework used as measure for each concept was clearly illustrated and summarized in Table 2.2.

Table 6.6.1: Description of Typical Situation in each Case

	Case 1: Software Engineers			Case 2: Academic Librarians		
Variable	**Category**	**Frequency**	**Percent**	**Category**	**Frequency**	**Percent**
1. Gender	Male	77	68.1%	Male	18	36.0%
	Female	36	31.9%	Female	32	64.0%
2. Bachelor's degree	BSc. Computer science	32	28.3%	BSc. Library science	3	6.0%
	BSc. IT	55	48.7%	BSc. Information science	13	26.0%
	BSc. Software engineering	22	19.5%	BSc. Library & Information science	27	54.0%
	Others	4	3.5%	Others	7	14.0%
3. Industry roles	Software Architect	18	15.9%	System Librarian	1	2.0%
	Analyst Programmer	26	23.0%	Reference Librarian	9	18.0%
	Test Engineer	14	12.4%	Information Literacy Librarian	6	12.0%
	Web Programmer	29	26.7%	Circulation Librarian	8	16.0%
	Mobile Programmer	9	7.9%	Africana Librarian	3	6.0%
	Systems Administrator	13	11.5%	Digital Media Librarian	7	14.0%
	Project Manager	4	3.5%	Multi-Purpose Librarian	7	14.0%
				Other	9	18.0%
4. University	UoN	14	12.4%	UoN	5	10.0%
	Kenyatta	9	8.0%	Kenyatta	21	42.0%
	Moi	4	3.5%	Moi	11	22.0%
	Egerton	9	8.0%	Egerton	1	2.0%
	KEMU	11	9.7%	KEMU	1	2.0%
	Daystar	10	8.8%	Daystar	1	2.0%
	Maseno	1	0.9%	Other	10	20.0%
	JKUAT	27	23.9%			
	Strathmore	8	7.1%			
	Nazarene	12	10.6%			
	Other	8	7.1%			

6.6.4. Conclusion validity

This kind of validity related to the possibility to draw correct conclusions regarding the relationship between treatments (independent variable) and outcome of an experiment (dependent variable) (Wohlin *et al.*, 2003). This tried to establish the power of the tests and the reliability of the measurements. The aim was to reduce errors and biases in the study so that if the same procedure was repeated by a second investigator would be able to arrive at the same findings and conclusion (Yin, 1994). One approach adopted was to make research methodology steps as operational as possible as evidenced in the research design (refer to section 3.3) and the verification of conditions for each statistical test procedure.

6.7. Summary

This chapter has presented experimental evaluation results of the study, and a detailed discussion of the major research findings. The climax of this discussion has culminated with validation of the mapping model through not only a holistic multi-case design but also a theoretical replication approach. For purpose of clarity, the results have been presented using not only tables and but also graphs. The statistical analysis procedures have been carefully selected based on preliminary tests results for data validity. The final research findings have been carefully drawn from both descriptive and experimental results after details discussion of the results and findings. In summary, the results and findings discussed in this chapter have provided answers to the fourth research question posed in this study.

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

7.0 Introduction

This chapter has been structured into sections: Section 7.1 presents the conclusion and future research. Section 7.2: highlights the research contributions. Section 7.3: presents the research limitations. Section 7.4: outlines the benefits and achievements of the study. Finally, section 7.5: highlights relevant research publications generated by this study.

7.1 Conclusion and Future Research

7.1.1 Conclusion

This study set out to investigate whether skills profile from employed graduates could be used to develop a machine learning model to map graduates' skills to industry roles that are hierarchically structured and the applicability of machine learning techniques in improving prediction of graduates' performance and productivity towards industry jobs. This was as a result of not only the glaring risk that graduates were facing of long term unemployment but also the growing dissatisfaction by industry over graduates' productivity as a result of poor evaluation of graduates' skills vis-à-vis industry job competence requirements in a practical way. To address this problem an investigation was launched to build a model for mapping graduate's skills to matching industry roles using machine learning techniques. The challenges towards this study were:

1) Lack of appropriate concepts to be used as machine learning attributes to predict performance of new graduates towards industry roles
2) Lack of understanding of characteristics of relevant concepts to be used as target classes for machine learning process for mapping graduates' skills to industry roles
3) Lack of a valid and effective machine learning model for predicting graduates' performance towards industry roles

The above challenges formed the basis of the research objectives which were operationalized through research questions and hypotheses to be answered. Initially, relevant literature was reviewed to understand the problem and its domain. A number of systematic questions, such as: what learning outcomes are looked for in the job industry; what learning outcomes enhance performance in the job; how we evaluate learning outcomes; what evaluation approaches are commonly used in related work. The literature derived from these questions was reviewed where three theories that are commonly used to evaluate learning outcomes were jointly analyzed to produce the conceptual framework while literature on related work was used to refine the final research questions in an attempt to achieve each of the corresponding objective.

We consider each of the objectives and the extent to which they were achieved.

1) To establish concepts appropriate as machine learning attributes for mapping graduates skills to occupational industry roles

Initially, the investigation to answer this research question was launched through literature review and analysis before empirical analysis refined these concepts. The main focus of literature review and analysis was to review and analyze literature so as to identify: 1) theories for evaluating learning outcomes, 2) underlying concepts of these theories that promote performance in the job, 3) suitable cognitive frameworks that could be used to assess these concepts in the academia. Three theoretical models for evaluating learning outcomes were identified, namely Kirkpatrick model, CRESST model, and Kraiger's model. Their underlying concepts were analyzed to reveal ones that promoted performance in the job, and their relationships were represented in the conceptual model (Fig.2.6). The proposed concepts in the conceptual model were operationalized using frameworks that provided indicators used to derive the variables for collecting data as shown in Table 2.5.

i) Selecting meaningful features for building the model

Findings #7 was related to determination of not only the number of features that would optimize performance of the classifier model but also whether the improved performance was significant. Findings #7 revealed 5 features out of 13 were able to induce the best performance results for the classifier model equivalent to performance that could be achieved with 13 features. Reduction of features had a number of benefits to the classifier model, including lowering demand for computational resources and reducing the processing time. The 5 features out of 13 were able to induce best performance of the classifier model and this performance improvement was significantly better than that of 13 feature model.

The findings revealed the number of valuable features as, namely R (Relevant content knowledge), D (Cognitive skills), A (Technical skills), C (Intellectual Capacity) and 'Age'. The implication of this findings provided insight not only into which features should be included in the subsequent investigations but also to accept or reject the hypothesis posed in research question 1: **H_{01A}: All features are equally relevant for better performance of the classifier model.** The outcome based on these findings was to reject the hypothesis at significance level, p=0.05. These findings' explanation was that when more than five features were used, the summary feature space dimension became too large causing performance of the model to start decreasing while when less than five features were used essential information was lost that caused accuracy to decline. These findings concurred with others in literature (Barbedo & Lopes, 2006).

2) To establish characteristics of concepts required as target classes for hierarchical machine learning purpose

Descriptive survey research design was adopted to answer this research question where the main focus was: 1) to establish concepts to be used as target classes for the machine learning process 2) to establish characteristics of these concepts in terms of their structural elements, structural relationship amongst them and relationship of academia towards these concepts. These issues were important to understand not only to help select the appropriate approach for building the machine learning model and design appropriate features for the prototype to handle new graduates from diverse institutions of academia but also to verify the research assumption that occupational industry roles were unique and hierarchical. To verify research assumption a hypothesis was defined, tested and provided results to answer the question. The findings towards this research question were organized according to the two main focuses as summarized below:

i) Establishing concepts to be used as target classes for machine learning process

Findings#1 and #6 were crucial in discovering industry roles concepts that formed the basis of creating target classes for machine learning. While findings#1 revealed the concepts as raw which were initially 7, findings#6 later on revealed the refined form of these concepts as 12. Finding#6 also revealed the distribution of these concepts that was important in deciding how to handle class imbalances during training process of machine learning for building the model.

ii) Establishing characteristics of target classes for machine learning process

The choice and design of machine learning methodology depends on: 1) structure of the problem and 2) assumptions about the learning problem (Kotsiantis, 2007; Silla & Freitas, 2011; Merschamann & Freitas, 2013). As a result, findings#2 was crucial in discovering that these concepts had similar structural elements (job activities/skills) but different levels of emphasis. Further, findings#5 discovered the structural relationship among these concepts that was crucial in deciding the machine learning approach suitable for building the classifier model, in this case hierarchical classification approach.

The fundamental assumption in the present study that occupational industry roles have different requirements for problem solving skills was put in the form of a hypothesis: **H_{02A}: There is no significant boundary differences between concepts to be used as potential target classes for machine learning.** Findinsgs#3 was crucial in rejecting this hypothesis. Hypothesis results suggested that occupational industry roles were unique and demanded a unique capacity to apply content knowledge learned during training. These results concurred with other findings that industry roles were becoming more and more diversified and therefore workers were required to be empowered to apply domain specific knowledge and skills differently in different industry roles (Chien & Chen, 2008).

Findings#4 was important in revealing that learning institutions have different biases towards these concepts. This was crucial in designing the model's prototype software to handle graduates from different learning institutions differently when deployed in the real world. Findings#4, concurred with other studies that, either methods used during training were only able to impart theoretical knowledge to the learners hence denying them application of knowledge and skills through practical training (Shaw, 2000, McCowan *et al*., 2016) or some areas were prescribed very little time, or were taught in more depth than others (Lethbridge *et al*, 2000; Kichenham *et al* 2005; Surakka, 2007).

Based on the findings in the study, it is important to note that, when developing classifier models for mapping skills to industry roles, target classes for machine learning are industry roles concepts which are distinct, and therefore, should be approached using supervised classification approach. Class distributions of these concepts could be imbalanced, and therefore, they may need stratified sampling during machine learning process of building the classifier model. Besides, structural relationship among these concepts is hierarchical, and therefore, the process of building the classifier model should be approached using hierarchical machine learning approach.

Finally, when designing software for the model to deploy for real world use, the underlying biases of different learning institutions towards these concepts should be known so that the model could handle graduates from different institutions differently.

3) To build using these concepts a machine learning model that maps graduates' skills to hierarchically structured industry roles

Experimental research design was adopted to answer the research question where the model was built through training and testing experimental processes. The main focus of these experimental processes was: 1) to select appropriate machine learning algorithm required to build the model 2) to select appropriate parameter values for the machine learning algorithm,. These two issues were key in building the machine learning model for mapping graduates skills to industry roles and were used as experimental objectives in the experiment design.

Because the nature of true experiments should not only be objective and repeatable but also be characterized by testing claims. Two hypotheses each for the two experimental objectives were defined, tested and their results utilized to answer the research question. Again, the findings towards this research question were organized according to the two main focuses as summarized below:

i) Selecting the best machine learning algorithm for building the model

This main focus of this experiment was to estimate the generalization performance of each of the two models generated by each machine learning algorithm and possibly help select the best induction algorithm. Both findings#9 and #11 were key in revealing this where both concurred that the general performance of the SVM classier model was much better than that of naïve Bayes and in fact the difference between the two was significant. Based on these findings, SVM was more likely to generalize its performance to unseen data in the real world better than naïve Bayes classifier model. As a result, it was selected as a candidate for the best classifier model.

Also, the two findings were key in rejecting a hypothesis posed in the research question that: **H_{o3C}: All induction algorithms induce equal generalization performance to the model.** These findings concur with other findings in literature that prediction performance of a classification methodology applied on a particular problem depends on the data, the induction algorithm for the model, and the expected results of analysis (Bedzek, 1981). Also, classifiers behave differently in many different datasets because induction algorithms that generated them have different internal biases and imposed different assumptions about the data (Raschka, 2015), but can be proved equivalent when applied on certain datasets (Mitchell, 2006).

ii) Selecting best parameter values for building the model

Both finding#8 and finding#10 were related to investigation towards parameter tuning, although through different datasets with different landscapes. Coincidentally, both findings agreed that parameter tuning of SVM improved performance of the classifier model significantly. However, parameter values that induced optimal performance of the classifier model were dataset dependent. The implication of these findings in this investigation suggested that in every different dataset we needed to tune the parameter values for optimal performance.

Also, these findings provided key evidence that was used to reject the hypothesis paused in the research question that: **H_{o3B}: Any parameter value produces better performance in the model.** These findings concurred with observations in literature that default parameter values in the libraries of induction algorithms may not induce better performance of a model (Raschka, 2015). Besides, parameter tuning is sometimes more important than even choosing an induction algorithm (Lavesson, 2006).

Based on the findings and outcomes of research hypotheses that were tested, two things (among others) were key in building machine learning model for mapping graduates skills to industry roles, namely selecting induction algorithms that induce appropriate generalization performance and tuning parameters of the model to appropriate values, and. These two were among the key determinants of the final performance of the model.

4) To evaluate the validity of the model

To answer this question, experimental research design and literature analysis were adopted where the model was evaluated through training and testing experimental processes then its results compared with other similar models in literature. The main focus of these processes was:1) to determine the generalization performance of the best classifier model selected for mapping graduates' skills to industry roles, 2) to compare performance of the classifier model with other models in literature. The two issues were important in understanding the properties or behavior of the classifier model from what was expected and how it compared with other models in literature.

But one that was key was experimental evaluation to establish the generalization performance of the best classifier model. Thus, it was handled as a true experiment and characterized by a testing claim, where a hypothesis was defined, tested and results utilized to answer the research question. Performance of the model on carefully selected benchmark was compared with performance of other models on the same dataset, while comparison was also made with other models using same approach and not necessarily on the same dataset. Again, the findings towards this research question were organized according to the two main focuses as summarized below:

i) To establish the generalization performance of the best classifier model

Findings #12, #13, & #14 were crucial in establishing the generalization performance of the best classifier model using best induction algorithm. According to findings #12, #13 & #14 the best classifier model seemed to show excellent performance behavior (along all four performance metrics adopted) in all the first, second, and third datasets where the average accuracy performance was 67%/. Along the three datasets, SVM model was able to show a consistent performance over naïve Bayes model (that was dropped in chapter 4) as expected along all the four performance metrics. Its performance per class was 100% accurate for about 30% of the target classes in a dataset (2 out 7 classes (28%) in dataset2 and 4 out of 12 classes (33%) in dataset3).

Besides, our model seemed to perform equally better in both dataset2 (SE field data) and datset3 (AL field data) and yet they belong to different occupational domains. This was a clear indicator of the ability of the classifier model to generalize in other occupational domains not covered in the study.

The findings#12 & #16 and Table 6.4 were key in signaling acceptance of a hypothesis posed in the research question that: **H_{o4A}: There is no significant performance difference of the classifier model in different industry domains.**

In the present study, the best generalization performance was calculated as an average performance across the three datasets as indicated in Table 6.3a&b. In this case, along hierarchical levels the best average performance of the model was 67% and, therefore, we can confidently claim that the best accuracy performance of our model was 67%.

ii) To compare performance of the best classifier model with other models in literature

Our model seemed to compare well with other hierarchical classifier models in literature. Model performance seemed to improve upward in the hierarchical levels of the taxonomy consistent with other models in literature (Clare & King, 2003; Barbedo & Lopes, 2006). Based on the model's performance as revealed in findings #18, average level performance (67%) across the three datasets was better than average level performance (33.5%) of Clare & King's model (2003) across 12 datasets. However, the model's average level performance was slightly lower compared to 75% that of Barbedo & Lopes's model (2006). Although, Barbedo & Lopes's model results (2006) were based on only one dataset which we did not know its distribution. Nevertheless, in the present study the best average performance achieved by the model was in dataset2 (benchmark) whose results were much better compared to Barbedo & Lopes's model (2006).

Although there was no evidence whether there was use of hierarchical approach, Shashidhar *et al.* (2015) built a classifier model using the same SE Benchmark dataset and achieved a performance of 82% which was slightly lower compared to the performance level achieved using the same Benchmark dataset (85%) by the classifier model produced in the current study. Assuming their classifier model was flat then our model was evidence that hierarchical models are more accurate than flat models as claimed in literature and hence concurs with other findings (Silla & Freitas, 2011; Merschamann & Freitas, 2013). To the best of our knowledge this was the first classifier model to pose skills mapping to industry roles problem as a hierarchical multiclass classification problem that was solved successfully.

7.1.2 Future Research

This model will greatly help to alleviate the risks facing graduates and employers due to effects of industry academia gap, such as employing graduates who do not match their needs, and taking longer to search the ever growing pool of new graduates with qualification mix. SVM and naïve Bayes were used to extract the rules for the model. SVM was adopted due to its high level of accuracy while naïve Bayes was chosen due to its ability to produce good results quickly while both of them are widely used in skills mapping. However, in order to improve on the reliability of the model, the following future research is highly recommended.

1) Testing this approach using other alternative machine learning techniques such as decision trees and neural networks.
2) Although the applicability of this approach in other alternative industry domains has been implied, it is important future research is conducted in these domains to confirm this so that more experiments to be performed with cases in other domains.
3) To ensure a good match between skills acquired in education and those required in the labor market, more investigation is needed to identify both individual and training attributes that predict transfer of learning.

7.2 Research Contributions

The key objective of this thesis was to investigate a model for mapping graduate's skills to industry roles using machine learning techniques so as to improve prediction performance for both employability and productivity. This was approached by using employees' data to model the relationship between employees' academic profile and work requirements. Useful strategies were applied in designing the model which would possibly accrue numerous benefits not only to the evaluation and recruitment processes of both academia and industry but also to the whole fraternity of researchers. These useful strategies culminated in making a significant contribution to the world of research.

Making a significant contribution implies adding to knowledge or contributing to the discourse by providing evidence to substantiate a conclusion that's worth making (Petre & Rugg (2010). Wobbrock & Kientz (2016) outlined seven types of research contributions in computing and these are theoretical, empirical, methodological, dataset, artifact, survey, and opinion.

The rest of this section presents the main contributions of this study as per Wobbrock & Kientz (2016) model.

7.2.1 Theoretical contributions

Theoretical contribution may be in the form of new or improved models, frameworks, concepts or principles that inform what we do, why we do it, and what we expect from it. They are evaluated based on their novelty, soundness, and power to describe, predict and explain (Wobbrock & Kientz , 2016). Based on this observation, this study has made the following theoretical contributions whose contribution to knowledge was analyzed using Whetten framework (1989):

1) Conceptual framework

Conceptual framework for studying skills mapping to industry roles problem was one of the major theoretical contributions. The framework identified factors that were appropriate to predict performance in the job and indicated the logical relationship between them (refer to chapter 2, Figure 2.6). The need for this framework was as a result of a missing tool for training evaluation that enhances both employability of graduates and performance in the job. For purposes of graduate employability, the conceptual framework was applied in understanding both factors that were key in differentiating between occupational industry roles from the academic point of view and the trends or biases in the academia towards these occupational industry roles.

The former was important especially in the design and implementation of the curriculum where there is need to identify core factors that target general aspects of occupational industry roles and the factors that target specific aspects of industry roles. The later was important especially to the academic institutions in evaluating their progress in enhancing graduates employability towards occupational industry roles.

The findings derived from analyses based on this framework have not only important significance to theory especially in explaining why we have qualification mismatch among graduates of same bachelor's degree program whether from same or different universities but also several implications both to the academia and industry in terms of: 1) coverage of domain specific knowledge and skills; 2) approval of curriculum by domain experts and stakeholders; 3) selection of undergraduate students; 4) emphasis of the right levels of thinking skills.

In summary, for many years the researchers and stakeholders both in academia and industry have been struggling to come up with a framework that could describe, explain, and bridge the gap between industry and academia. This is what this conceptual framework has done theoretically and is a contribution that is significant to skills mapping researchers who would want to describe, explain, and bridge industry academia gap by using this conceptual framework as a research model.

2) Taxonomic structure

Taxonomic structure that is not only friendly to bottom-up classification methodology but also based on functional organizational structure was developed. This indicated the logical and hierarchical relationship between nodes that reduced multiple label prediction problem (refer chapter 2, Figure 2.8). This approach where the classification taxonomic structure was derived from functional organizational structure has not been applied anywhere in machine learning literature. The approach renders skills mapping to industry roles practically relevant and reliable, where skills mapping is performed according to not only the natural structure of industry roles but also the natural mobility of employees in the organizational structure, and this promotes single label prediction.

The significance of this approach lies in partitioning the industry roles in three natural dimensions adopted in many organizations (i.e. functional, proficiency, and specialty) when considering employability of personnel. In summary, this is a positive contribution to the body of knowledge in machine learning based skills mapping where none had existed. The significance of this contribution extends to researchers in skills mapping to industry roles who can use this taxonomic structure to organize classes as required in supervised machine learning. The implication of this finding is that for relevant and reliable results in skills mapping, this kind of taxonomic structure is vital. The original class taxonomy for hierarchical classification was defined by Wu *et al.* (2005) as a tree structure with two properties, anti-reflexive and transitive. Further, Silla & Freitas (2011) extended the definition to include asymmetric properties.

However, these properties by both are biased towards top-down approach to hierarchical multi-class classification where the assumption is a child node naturally belongs to not only the immediate parent node but also all other ancestor nodes up the hierarchy. This makes it difficult to apply bottom-up navigation without leading to multiple labels. As a result, to make the class taxonomy compatible with the currently proposed bottom-up method so as to promote integrity and validity of this method, transitive property of the taxonomy was reviewed as follows: For every class $c_i; c_j ; c_k \in C$; *c_i is related to c_j and c_j is related to c_k does not imply c_i is related to c_k.*

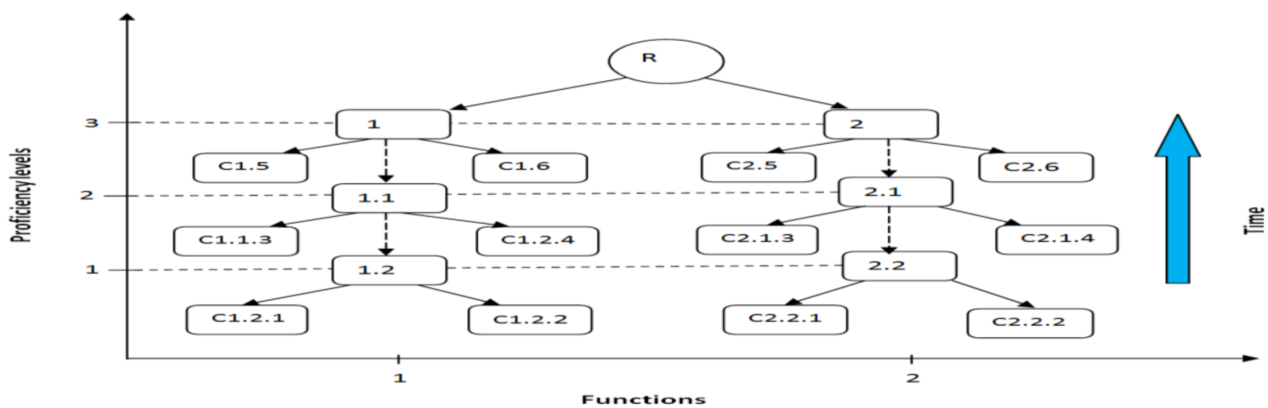


Figure 2.8: Bottom-up friendly taxonomic structure (repeat)

Figure 2.8 illustrates hierarchical structure with two branches (may be more), each branch with three levels, a total of twelve leaf node classes (C1.5, C1.6, C1.1.3, C1.2.4, C1.2.1, C1.2.2, C2.5, C2.6, C2.1.3, C2.1.4, C2.2.1, and C2.2.2), and a total of six parent nodes (1, 1.1, 1.2, 2, 2.1, and 2.2), and root node (R). Leaf nodes represent specialized individual roles while the upward arrow indicates the direction of employees' occupational mobility with time.

In the context of skills mapping, the above proposed taxonomic structure represents the hypothetical structural organization of occupational industry roles' problem, and reflects not only the natural mobility of employees upward the occupational ladders but also promises effective bottom-up mapping of graduate skills to industry roles that does not result to multiple label prediction problem. As per the assumptions of the current skills mapping problem, each branch represents an occupational function which refers to a skills category, each level or non-leaf node represents a skills proficiency which refers to a skills level, while each leaf node represents specialty of industry role which refers to a skills type. However, while each specialty is a member of a proficiency category, relationship between proficiency categories is one of peer to peer where one category follows the other.

The main difference between the proposed taxonomic structure and the traditional tree structure is eminent at the levels/non-leaf nodes where the former adopts peer-to-peer and the later adopts parent-child relationships. While in the traditional structure lower level parents are decompositions of higher level parents, this is not the case in the proposed structure as each level is a category that indicates superiority of skills proficiency. However, to be able to explore the proposed taxonomic structure from bottom to top as it is natural with employee mobility in the organizational hierarchy, there was need of a special type of architecture for the skills mapping model, which was clearly another contribution in the study.

3) Architecture of the ML based model for skills mapping

Architecture of the machine learning based model for mapping graduates' skills to industry roles was another theoretical contribution (refer to chapter 2, section 2.7.5, Figure 2.8a&b). This architecture was significant for the machine learning model to not only browse the taxonomic structure but also produce single-label prediction results. This architecture has been and will be the backbone for producing software prototype as revealed in chapter 5. The need for this architecture was as a result of a missing model for training evaluation that predicts both employability of graduates and performance in the job without multiple label results. Figure 2.9a illustrates the building blocks of the bottom-up machine learning architecture of the model.

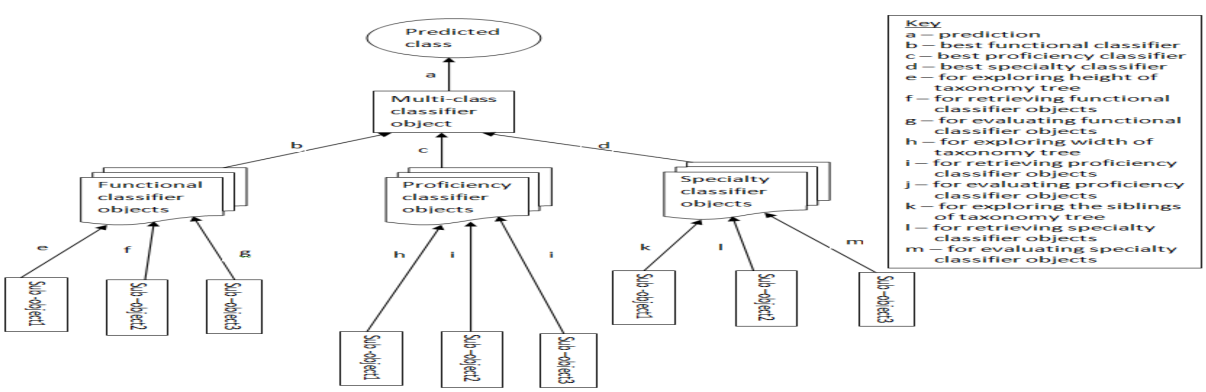


Figure 2.9a: Machine Learning Architecture for the Model (repeat)

4) Theoretical Knowledge

Besides, Whetten (1989) provided a comprehensive elaboration of a framework for analyzing what a theoretical contribution to knowledge was, and this was based on four elements of a theory: 1) Element that relates to the factors that should be considered as part of the explanation of a phenomena of interest (what), 2) Element that relates to relationship between factors that describe casual nature of the theory (how), 3) Element concerned with the justification of the selection of factors and their proposed casual relationship (why) and 4) Element concerned with the range of the theory in terms of the limitations placed on the propositions generated from the theoretical model (who,where,when). Each of these elements can be evaluated against a certain criteria to establish their correctness, practicality, reasonableness, and generalizability respectively. Fig.7.2 summarized the analysis to knowledge contribution.

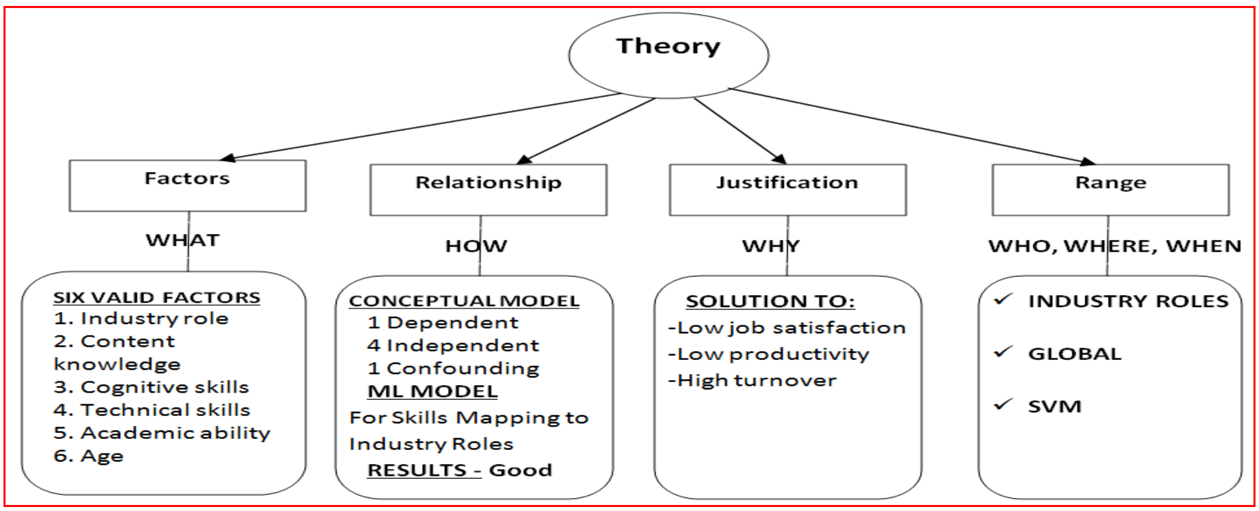


Figure 7.2: Analysis of contribution to knowledge

In the present study, the existing theory under investigation was on prediction of job performance originally by Schmidt & Hunter (1992). Using Whetten (1989) framework, analysis was conducted to bring out clearly the contribution to theoretical knowledge and was guided by two themes or arguments or lines of thoughts that justify a new and valid contribution: 1) How factors either added as new or removed as redundant in existing models affected accepted relationship between variables, and 2) The reason why a theory either does not work in a new setting or does work when it was not expected to. The main independent variables in the present hierarchical mapping model for skills mapping were content knowledge, cognitive skills, technical skills, and academic capacity.

There is compelling evidence in the body of literature suggesting personality traits, language, cognitive skills, domain skills, and job knowledge are important attributes for predicting job suitability for a job seeker (Schmidt & Hunter, 1992, Chang & Xi, 2009, Shashidhar *et al*, 2015). However, it was not known or expected whether 1) these factors would remain valid in a new setting where industry roles were considered as structured hierarchically 2) how new sub-variables of each category would affect this known relationship between these main factors.

A slightly similar empirical study by Shashidhar was based on two categories of four sub-variables: cognitive skills (English comprehension, logical ability, Quantitative ability) and domain skills (Programming skills), different sub-variables from the ones used in this study. Table 7.1 has summarized the results. The observation reveals as the number of sub-variables increases, the strength of relationship between these factors slightly improved as indicated. The improvement was attributed to the hierarchical approach applied and was expected. However, the addition of new sub variables that were easy to work with in academia did not compromise the relationship that describe the casual nature of the theory, and was the greatest contribution to theoretical knowledge of the study. We therefore, conclude that our contribution to knowledge was a hierarchical mapping model for skills mapping to industry roles.

Table 7.1: Summary of analysis of theoretical knowledge impact

	What factors				Results
	Cognitive skills	Domain skills	Knowledge	Academic capacity	
Shashidhar *et al*, 2015	-English, -logical ability -Quantitative ability	- Programming skills			80-82%
Current study	-Recall -Comprehension -Application -Analysis -Synthesis -Evaluation	-Programming -Database -Operating systems -Networking -Distributed	-Reqments Analy. -Sys. Design -Dev. Process -Project Mgt -Configuration mgt	-High School GPA -College GPA	83-85%
Change	**addition**	**addition**	**addition**	**Addition**	**improve**

7.2.2 Methodological contributions

Methodological contribution were in the form of new or improved methods that inform how we discover, measure, analyze, create or build things. They improve research or practice and are evaluated based on their utility, reproducibility, reliability, and validity (Wobbrock & Kientz , 2016). Based on this observation, this study has made the following contributions:

1) Research framework for operationalizing the study (refer to chapter 3, section 3.3, Figure 3.1)
2) Mapping frameworks for grouping roles into logical classes based on their underlying functional requirements and promoting maximum intra-class similarity and minimum inter-class similarity (refer to chapter 3).

7.2.3 Dataset contributions

Dataset contribution was in the form of new and useful corpus, accompanied by an analysis of its characteristics that would enable the research community to perform evaluations against shared benchmarks by new algorithms or systems or methods (Wobbrock & Kientz , 2016). They are valued based on the extent they supply useful and representative corpus against which to test and measure. As a result, this study was able to generate three types of datasets for hierarchical multi-class classification problems whose characteristics were well described (refer to chapter 3&4).

1) Research dataset (dataset1) stores data for software engineers' field
2) Benchmark dataset (dataset2) is an extract from AMEO2015 dataset which is an Engineers dataset that is famous in the machine learning industry
3) Validation dataset (dataset3) stores data for academic librarians' field data

7.2.4 Empirical contributions

Empirical contributions were in the form of findings based on systematically observed data both from experiments and data collection (Wobbrock & Kientz , 2016). These were evaluated based on the importance of their findings and soundness of their methods. In this case, the study revealed very compelling findings that are relevant to the contemporary problem facing both the academia and industry. The discussion of these findings were well supported with validity claims (refer to section 6.6)

1) Research findings in research question #1 revealed there is significant difference in both knowledge and skills among occupational industry roles.
2) Research findings in research question #2 revealed that the trends towards industry roles were not uniform among universities
3) Research findings in research question #3 revealed that prediction performance of the mapping model was affected by both machine learning technique used for the model induction.

4) Research findings in research question #4 revealed that indeed generalization of the mapping model across industry domains was practically feasible and valid

7.2.5 Artifact contributions

Artifact contributions were inventions in the form of systems, tools, techniques, or architectures that showed how to accomplish either new things formerly impossible or things formerly possible but now more easily (Wobbrock & Kientz , 2016). These enabled to make new explorations or facilitate new insights. In this study, the main artifact produced was a software prototype for mapping graduates' skills to industry roles.

1) Software prototype in the name of WEMA (Where Employers Meet Academia) was developed as a platform where employers and academia (students and university administrators) meet to interact with the mapping model.

7.2.6 Survey contributions

Survey contribution was in the form of review and synthesis conducted in a research field with the goal to reveal trends, themes, and gaps in literature. In this study a thorough literature review was conducted and was able to reveal the gap, namely ineffectiveness of hierarchical classifiers to map graduates' skills to industry roles.

7.3 Research Limitations

Although this approach has numerous benefits it has the following limitations.

1) It depends on evaluation of the currently employed graduates. The skill requirements of the industry roles derived from incumbents may not correspond exactly to the levels they are holding, with some being overqualified or under qualified, or due to change in entry requirements for the occupation after they were employed.

7.4 Benefits and Achievements

Skills mapping as a mechanism that links industry job (entry-level or on-demand) with a highly skilled workforce (Johnson, C. & Simpson, T., P-Tech Brooklyn) was directly informed by actual job requirements and was the lynchpin for connecting the best employment opportunities to a series of rigorous classroom learning objectives. It reduces the risk of hiring overqualified or under-qualified graduate employees. Hiring overqualified or under-qualified workers may result into: 1) industry compensating these positions at a higher rate than necessary, 2) workers likely to leave if they find a more appropriate position, 3) high potential graduates likely to be left out of consideration for jobs they could perform brilliantly.

As a result, the mapping model generated by this study has numerous benefits not only to evaluation processes of academia but also recruitment processes of industry as outlined here.

1) The approach lowers the cost of hiring by empowering employers to practice direct hiring, rather than hiring through recruitment agencies which can sometimes be very expensive.

2) This approach, also, focuses to reduce evaluation time wasted during recruitment. Matching of the vector of characteristics employers seek against characteristics of new graduates or applicants will make possible to predict probability of success of the worker within few seconds of waiting rather than long interviews.

3) In addition, it provides a standard way of graduate assessment by promoting evidence based decision making rather than the employer using duration of unemployment as a signal of the quality of the worker.

4) This approach can, also, promote improvement of job search strategies followed by new graduates, by increasing search intensity and efficiency. Large database of up-to-date job requirements can be searched and analyzed online.

The following achievements have evidently marked the success of this study:

1) A hierarchical method that uses fewer classifiers (K-1) than popular methods, such as one against one approach (K(K-1))/2 and one against all approach (K classifiers).

2) A hierarchical method that registers fairly better performance accuracy (65-67%) than the benchmark method (61%)

3) Empirical findings to be used as a basis of deciding in future the methods, tools, and techniques to apply when developing an automatic skills mapping to industry roles software.

4) Extension of the list provided by Silla & Freitas (2011) on taxonomic structures for hierarchical classification.

7.5 Relevant Research Publications

Mwakondo FM, Muchemi L & Omwenga EI. “Proposed Model for Predictive Mapping of Graduate’s Skills to Industry Roles Using Machine Learning Techniques”. *The International Journal of Engineering And Science (IJES) Vol.5, Issue 4, PP -15-24, 2016* .

Mwakondo FM, Muchemi L & Omwenga EI. “Trends towards Predictive Mapping of Graduate’s Skills to Industry Roles: A Case Study of Software Engineering”. *British Journal of Education, Society & Behavioral Science Vol.18, Issue 1, PP -1-17, 2016* .

Mwakondo FM, Muchemi L & Omwenga EI. “Automatic Mapping of Graduate’s Skills to Industry Roles using Machine Learning Techniques: A Case Study of Software Engineering”. *International Journal of Computer Science & Technology Vol.9, Issue 4, PP -111-118, 2016* .

References

Aly, M.(2005).Survey on Multiclass Classification Methods

Aggarwal, V., Srikant, S., & Nisar, H. (2015). A dataset comprising AMCAT test scores, biodata details, and employment outcomes of job seekers.

Amoui, M., Salehie, M., & Tahvildari, L. (2009).Temporal Software Change Prediction Using Neural Networks. *International Journal of Software Engineering and Knowledge Engineering, January,8,2009*

Barbedo J.G.A, & Lopes A. (2007).Automatic Genre Classification of Musical Signals". *Journal on Advances in Signal Processing Volume 2007, Article ID 64960, 12 pages doi:10.1155/2007/64960*

Basu, J., Bhattacharyya, D. & Kim, T. (2010). Use of Artificial Neural Network in Pattern Recognition. *International Journal of Software Engineering and Its Applications Vol. 4, No. 2, April 2010*

Benbasat, I. (1984).An Analysis of Research Methodologies in *The Information Systems Research Challenge*, F. Warren McFarlan (ed.), Harvard Business School Press, Boston, Massachusetts1, 984, pp 47-85.

Bharthvajan, R. (2013). Competency Mapping. *International Journal of Innovative Research in Science,Engineering and Technology Vol. 2, Issue 11, November 2013*

Bigelow, K. (2009). Lecture Notes for AI (CS-482) Fall 2009, Lecture 24

Bloom, B.S. (Ed.), Engelhart, M.D., Furst, E.J., Hill, W.H., & Krathwohl, D.R. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain*. New York: David McKay.

Boehm, B. (2005). Some Future Trends and Implications for Systems and Software Engineering Processes. Retrieved February 20, 2013 from http://csse.usc.edu/csse/TECHRPTS/2005/usccse2005-507/usccse2005-507.pdf

Bondesson, T.(2004).Software Engineering Education Improvement: An Assessment of a Software Engineering Programme Thesis

Cannady, J. (1998).Artificial neural networks for misuse detection. Retrieved February 17, 2013 from http://csrc.nist.gov/nissc/1998/proceedings/paperF13.pdf

Chang, H.(2009).Employee Turnover: A Novel Prediction Solution with Effective Feature Selection Wseas Transactions on Information Science and Applications Issue 3, Volume 6, March 2009

Chien C., Chen L.(2008). Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with Applications 34 (2008) 280–290* .Retrieved February 17, 2016 from: http://www.sciencedirect.com

Clare A. and King R. D. (2003).Predicting gene function in Saccharomyces cerevisiae". *Bioinformatics Vol. 19 Suppl. 2, pp. ii42–ii49, 2003*

Cope, C., Staehr, and L. & Horan, P. (2000) .Towards Establishing the Best Ways to Teach and Learn about IT. The Challenge of IT Education in the 21st Century

Dalton, J. and Smith, D. (1986). Extending Children's Special Abilities - Strategies for primary classrooms. Australia: Ministry of Education Victoria, 1986, 36-7.

doITinKenya.(2011).Kenya National ICT Survey Results, 2011

Dodig-Crnkovic, G. (2002). Scientific Methods in Computer Science. *Proc. Conf. for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, (2002)*

EasterBrook, S., Singer, J., Story, A. & Damian, D. (2007).Selecting Empirical Methods for Software Engineering Research Oct 24-25, 2007. Retrieved February 15, 2013 from http://www.springerlink.com/index/q78020282234148r.pdf

Ellis, H., Moreno, A., Mead, N. & Seidman, S. (2002).Industry/University Software Engineering Collaborations for the Successful Reeducation of Non-Software Professionals

Garg, K. & Varma, V. (2008): People issues relating to software engineering education and training in India, ACM, India. *Software Engineering Conference, Proceedings of the 1st conference on India software engineering conference, pp. 121-128.*

Ghezzi, C. & Mandrioli, D. (2006). Challenges of Software Engineering Education. Springer-Verlag Berlin Heidelberg, 2006

Green, H. (2014). Use of theoretical and conceptual frameworks in qualitative research. *Nurse Researcher. 21, 6, 34-38.*

Hämäläinen, H., Ikonen, J. & Porras, J. (2011).A Tool for Visualizing Skill Requirements in ICT Job Advertisements. *7th E-learning Conference, e-Learning'11 (E-Learning and the Knowledge Society), Bucharest, Romania, August 25-26th, 2011, pp. 254-259*

Handel, M. (2012). Trends in Job Skills Demands in OECD Countries.

Harb, H. M. and Moustafa, M. A. (2012). Selecting optimal subset of features for student performance model. *Int J Comput Sci, (9):253{262.*

Haryanto, I. , Setiawan, J. & Budiyono, A. (2007). Structural damage detection using randomized trained neural networks. *ICIUS Bali, Indonesia, Oct 24-25, 2007*. Retrieved February 15, 2013 from http://www.springerlink.com/index/q78020282234148r.pdf

Harshito, T. (2012). Evaluation of Training and Development: Analysis of Various Models. *IOSR Journal of Business and Management, ISSN: 2278-487x. Vol. 5, Issue 2, pp. 16-22.* Retrieved June 15 2015 from www.iosrjournals.org

Hirschheim, R. (1985). Chapter2: Information Systems Epistemology. An Historical Perspective

HKCS (2011).Development of a Certification Roadmap for IT Professional Certification" Project, Aug 2011

Holz, H., Applin. A., Joyce, D., Purchase, H., Haberman, B, & Reed, C. (2006). Research Methods in Computing: What are they and how do we teach them? *ITiCSE, Bologna, Italy. 2006, 576.*

Houghton, R.S. (2012).Thinking & Teaching Tools for Digital Thought, August 2012. Retrieved September 20, 2013 from http://www.wcu.edu/ceap/houghton/readings/technology_trends.html

Hunt, E., Martin, J., & Stone, P.(1966). Experiments in Induction. New York: Academic Press, 1966.

IST-Africa Consortium. (2012). Guide to ICT Policy in IST-Africa Partner Countries v2.2 20 April 2012.

Jantawan, B. & Tsai, C.(2013).Application of Data Mining to Build Classification Model for Predicting Graduate Employment. *International Journal of Computer Science and Information Security, Vol. 11, No.4, October, 2013.*

Jones, K., Harland, J., Reid, J. & Bartlett, R. (2009). Relationship between Examination Questions and Bloom's Taxonomy. *39th ASEE/IEEE Frontiers in Education Conference October 18 - 21, 2009, San Antonio, TX.*

Johnson, C. & Simpson, T. (P-Tech Brooklyn) Pathways in Technology Early College High School (9-14) Model Development: Skills Mapping Process Guide

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects *Science Vol. 349, Issue 6245.*

Kaku, M. (2012).Tweaking Moore's Law: Computers of the Post-Silicon Era. Big Think. http://bigthink.com/ideas/42825

Kaminchia, S.(2014). Unemployment in Kenya: Some economic factors affecting wage Employment. *African Review of Economics and Finance Vol. 6, No. 1, June 2014*

Kanellos, M. (2003).Intel Scientists Find Wall for Moore's Law.ZDNet. Retrieved February 15, 2013 from http://www.zdnet.com/news/intel-scientists-find-wall-for-moores-law/133066

Kashorda, M. (2007). *Emerging Trends in Information and Communications Technology Education in Kenyan Universities.* **In** M. Kashorda, F. Acosta and C. Nyandiere (*eds*). ICT Infrastructure, Applications, Society and Education: Proceedings of the Seventh Annual Strathmore University ICT Conference. Strathmore University Press: Nairobi

Kellaghan, T. & Greaney, V. (2003). Monitoring Performance: Assessment and Examinations in Africa. *Association for the Development of Education in Africa ADEA Biennial Meeting 2003 (Grand Baie, Mauritius, December 3-6, 2003)*

Kenny, N. & Desmarais, S. (2010). A Guide to Developing and Assessing Learning Outcomes at the University of Guelph

Keshtkar, F., Burkett, C., Li, H. & Graesser, A. (2014). Using Data Mining Techniques to Detect the Personality of Players in an Educational Game. *Studies in Computational Intelligence, Vol. 524,2014.*

Kitchenham, B., Pickard, L. & Pfleeger, S.L. (1995). Case Studies for Method and Tool Evaluation

Kolding, M. & Ahorlon, M. (2009).Post Crisis: e-Skills Are Needed to Drive Europe's Innovation Society. Retrieved September 20, 2013 from http://ec.europa.eu/enterprise/sectors/ict/files/idc_wp_november_2009_en.pdf

Korte W., Husing T., Hendriks, L. & Dirkx, J. (2013). Towards a European Quality Label for ICT Industry Training and Certification. *Final Report.* 2013.

Kotsiantis, S.B.(2007). Supervised Machine Learning: Review of Classification Technques. *Informatica, 2007, Vol 31, pp. 249-268*

Kumar, S, Ghosh, J. & Crawford, M. (2002). Hierarchical fusion of multiple classifiers for hyper spectral data analysis, Pattern Analysis& Applications, 5:210-220, 2002.

Leeuwen, J.V. (2004).Chpater1: Approaches to Machine Learning. Algorithms in Ambient Intelligence. Kluwer Academic Publishers (2004) Ed.

Lenth, R. (2001). Some Practical Guidelines for Effective Sample Size Determination. *The American Statistician, August 2001, Vol. 55, No. 3.*

Ludi, S. & Collofello, J. (2001).An Analysis of The Gap Between The Knowledge And Skills Learned In Academic Software Engineering Course Projects And Those Required In Real Projects

Mathers, N., Fox, N,. & Hunn, A. (2007). Surveys and Questionnaires. The NIHR RDS for the East Midlands / Yorkshire & the Humber, 2007.

Mead, N., Tobin, L., Couturiaux, S. (1996).Best Training Practices within the Software Engineering Industry. Retrieved September 20, 2013 from http://www.sei.cmu.edu/reports/96tr034.pdf

Mehra, N. & Gupta, S. (2013). Survey on Multiclass Classification Methods. *International Journal of Computer Science and Information Technologies, Vol. 4 (4), 2013, 572 - 576*

Merschmann, L.H. C. & Freitas, A.A. (2013). An Extended Local Hierarchical Classifier for Prediction of Protein and Gene Functions.

Mgala, M. (2016). Investigating Prediction Modeling of Academic Performance for Students in Rural Schools in Kenya, *PhD Thesis.*

Mitchie, D., Spiegelhalter, D.J., Taylor, C.C. (1994). Machine Learning, Neural Networks and Statistical Classification

Moreno, A., Sanchez-Segurab, M., Medina-Dominguezb, F. & Carvajal, L. (2012).Balancing software engineering education and industrial needs

Nagata, H., Toda, S., Itsumura, H., Koyama, K., Saito, Y., Suzuki, M., & Takahashi, N. (2006). Body of professional knowledge required for academic librarians in Japan. In C. Khoo, D. Singh & A.S. Chaudhry (Eds.), *Proceedings of the Asia-Pacific Conference on Library & Information Education & Practice 2006 (A-LIEP2006), Singapore, 3-6 April 2006* (pp. 316-327). Singapore: School of Communication & Information, Nanyang Technological University.

NAS. (1995).On Being a Scientist: *Responsible Conduct in Research*, Second Edition (1995)

NOC. (2011). Human Resource & Skills Development Canada. *National Occupational Classification 2011 Report*

Norwood, B. & Briggemen, B. (2010). Assessing the College Graduate: How Employers Measure Graduates' Possession of General Skills.

OECD. (2012). ICT Skills and Employment; New Competences and Jobs for Greener and Smarter Economy. OECD Digital Economy Papers, No.198. http://dxdoi.org/10.1787/5k994f3prlr5_en

Ojo, A. & Estevez, E. (2005). Object Oriented Analysis & Design with UML. *e-Macao Report 19, Version 1.0*

Onwuegbuzie, A.J, Leech, N.L, & Collins, K.M.T. (2012). Qualitative Analysis Techniques for the Review of the Literature . *Qualitative Report 2012, Vol 17.*

Orhun, N. (2003).Effects of Some Properties 5. Grade Students on the Performance of Mathematical Problem Solving

Payne, C. & Payne, J. (2000). Early Identification of the Long Term Unemployed. PSI Research Discussion Paper 4, *PSI Report No. 874*

Perron R. (2011). Employer Expections and Experiences. Findings, Training and Keeping Qualified Workers. 2011.

Pideaux, D. (2003). ABC of learning and teaching in medicine: curriculum design. *British Medical Journal* 326:268- 270.

Pillai, S. Curriculum Design and Development http://www.unom.ac.in/asc/pdf/curriculum design and development-1.pdf

Pressman, R. S. (2001). Software engineering: a practitioner's approach.—5th ed. *McGraw-Hill series in computer science index. ISBN 0-07-365578-3*

Quintin G. (2011). Right for the job: Overqualified or Underqualified.

Raschka, S. (2015). Python Machine Learning. *Packt Publishing, Copyright 2015*

Refaeilzadeh, P., Tang, L., & Liu, H. (2008). Cross-Validation, Arizona State University. Retrieved November 16, 2014 from http://www.leitang.net/papers/ency-cross-validation.pdf

Richardson, M. & Abraham, C. (2012). Psychological Correlates of University Students' Academic Performance: A Systematic Review and Meta-Analysis. *Psychological Bulletin 2012, Vol. 138, No. 2, pp. 353–387*

Saidian, H. (2002).Bridging Academic Software Engineering Education and Industrial Needs. *Computer Science Education 2002, Vol. 12, No. 1-2, pp. 5-9*

Saunders, M, Lewis, P., & Thomhill, A. (2009). Research for Business Studies, 5th Edition

Schmidt, F.L and Hunter, J. E. (1992).Development of a causal model of processes determining job performance. *Current Directions in Psychological Science, pp. 89–92,1992*.

Shamoo, A. & Rensik, D. (2009). *Responsible Conduct of Research, 2nd ed. (New York: Oxford University Press).*

Shashidhar, V., Srikant, S., Aggarwal, V.(2015). Learning Models for Personalized Educational Feedback and Job Selection. *Proceedings of the 32 nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015*

Shaw, M. (2000).Software Engineering Education Roadmap. The Future of Software Engineering. Anthony Finkelstein (Ed.), ACM Press 2000ACM E-Store: http://store.acm.org/acmstore

Shaw, M. (2002). What Makes Good Research in Software Engineering? *International Journal for Software Tools and Technology Transfer Vol.4, no.1, pp. 1-7*: http://store.acm.org/acmstore

Shkoukani, M. (2013a).Proposed Model to Find the Gap between Academic Supply and Industry Demand in Software Engineering Field in Jordan. *International Journal of Advanced Computational Engineering and Networking, ISSN (p): 2320-2106, Volume-1, Issue-2, April-2013*

Shkoukani, M. (2013b).The Ability to Provide Well Qualified Software Engineering Graduates to the Software Industry Case Study: Jordanian Universities. *International Journal of Engineering Science and Technology (IJEST) ISSN: 0975-5462 Vol. 5 No.03 March 2013*

Shkoukani, M. & Lail, R. (2012).The Importance of Restructuring Software Engineering Education Strategies In Order To Minimize The Gap Between Academic Supply And Industry Demand In Software Engineering Field, *International Journal of Reviews in Computing, Vol.11, pp. 26-31*.

Silla, C.N & Freita, A.A. (2011).A Survey of Hierarchical Classification across different Application Domains. *Data Mining and Knowledge Discovery ·January 2011*

Software Engineering, GSwE2009 (2009). Curriculum Guidelines for Graduate Degree Programs in Software Engineering. Integrated Software & Systems Engineering Curriculum (ISSEC) Project.

Software Engineering, SE2004 (2004). Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A Volume of the Computing Curricula Series.

Sokolova, M. & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing and Management 45 (2009) 427–437

Srikant, S., Aggarwal, V. (2014).A system to grade computer programming skills using machine learning

Stefanowski, J. (2010). Data Mining -Evaluation of Classifiers .Lecture 4: 2010, Institute of Computing Sciences Poznan University of Technology Poznan, Poland. Retrieved November 16, 2014 from http://www.cs.put.poznan.pl/jstefanowski/sed/DM-4-evaluatingclassifiersnew.pdf

Surakka, S. (2005).Trend Analysis of Job Advertisements: What Technical Skills Do Software Developers Need?

Thompson, j., Noro, M. & Edwards, H. (2007).Introduction to the Workshop: Bridging the University/Industry Gap. *Proceedings Workshop on Software Engineering Education. A workshop collocated with 14th Asia-Pacific Software Engineering Conference (APSEC'07) Nagoya, Japan, December 4, 2007*

Tomayko, J. (1998). Forging a discipline: An outline history of software engineering education. *Annals of Software Engineering* 6 (1998) 3-18

Tutorialspoint.com. "Design Patterns in Java Tutorial", *Simply Easy Learning*. www.tutorialspoint.com

Walter, C. (2005).Kryder's Law. *Scientific American*. Retrieved September 20, 2013 from http://www.sciam.com/article.cfm?id=kryders-law.

Winterton, J., Delamere, F. & Stringfellow, E. (2005). Typology of Knowledge, Skills and Competences: Clarification of the concept and prototype.

Wirsch, A. (2014). Analysis of Top-down Bottom-up Data Analysis Framework and Software Architecture Design. Retrieved on 3rd December 2016 from *web.mit.edu/smadnick/www/wp/2014-08.pdf*

Witten, I.H. and Frank, E. (2005) Data Mining: Practical machine learning tools and techniques. 2nd edition Morgan Kaufmann, San Francisco.

Wobbrock, J.O & Kientz, J.A. (2016). Research contributions in human-computer interaction. *Digital Library, ACM, 2016*. https://interactions.acm.org/archive/view/may-june-2016/research-contribution-in-human-computer-interaction.

Wohlin, C. & Regnel, B.(1999).Achieving Industrial Relevance in Software Engineering Education, Proceedings Conference on Software Engineering Education & Training, pp. 16-25, New Orleans, Lousiana, USA, 1999.

Wu, F., Zhang, J., Honavar, V. (2005) Learning classifiers using hierarchically structured class taxonomies. *In: Proc. of the Symp. on Abstraction, Reformulation, and Approximation, Springer, 313-320, vol 3607* www.businesslist.co.ke. “Computers Companies in Kenya”. Retrieved on February 20, 2015. From: http://www.businesslist.co.ke/category/computers

www.kenya-information-guide.com. "Nairobi-The Center for Business and Economic Activities in Kenya". Retrieved on February 20, 2015. From:http://www.kenya-information-guide.com/nairobi-business.html

Vernon, D. (2009). Software Engineering 2: Module 514 Course Notes, Khalifa University

Vural, V. & Dy, J.G. (2004). A hierarchical method for multi-class support vector machines. *In Proceedings of the Twenty-First International Conference on Machine Learning, 105-112, 2004*

Yin, R., K. (1994). Case Study Research Design and Methods. *Sage Publications, Beverly Hills, California*

APPENDIX A: TIME SCHEDULE & BUDGET

Proposed Research Time Frame

		TIME IN MONTHS										
S/NO	**ACTIVITY**	**Sept-2014-Marc-2015**	**Apr-2015 July-2015**	**Aug-2015-Sept.-2015**	**Oct-2015-Jan.-2016**	**Febt-2015 - Marc-2016**	**Apr.-2016-May.-2016**	**June-2016-Aug.-2016**	**Sept-2016-Dec-2016**	**Jan.-2017-Apr.-2017**	**May-2017-June.-2017**	**TOTAL(MONTHS)**
1	Proposal Writing/ Approval											
2	DEVELOP initial MODEL											
3	Data collection for industry roles											
4	Analyzing job title/roles' specs & DIFFERENCES											
5	Data collection for degree programs											
6	Analyzing academia TRENDS											
7	Prototyping MODEL											
8	Phase 1: EVALUATE MODEL											
9	Phase 2: EVALUATE MODEL											
10	Report Writing/ Presentation											
	DURATION (MONTHS)	**7**	**4**	**2**	**4**	**2**	**2**	**3**	**4**	**4**	**2**	**34**

Budget (Kenya Shillings (KSh.))

NO.	ITEM	QUANTITY	UNIT COST	TOTAL COST
1.	Laptop	1	80,000	80,000
2.	Stationery Rim	20	500	10,000
3.	Internet Modem	2	5,000	10,000
4.	Internet data bundle(1GB)	100	1,000	100,000
5.	Printing copies	1000	30	30,000
6.	Binding (copy)	8	5000	40,000
7.	Transport(trips)	10	100,000	1,000,000
8.	Tuition Fee			838,000
	TOTAL			**2,088,000**

APPENDIX B: LETTER TO THE RESPONDENTS

University of Nairobi,
School of Computing and Informatics,
P.O Box 30197,
Nairobi.
4th June, 2015.

Dear Respondent,

COLLECTION OF RESEARCH DATA

I am a PhD student at the University of Nairobi, School of Computing and Informatics.

In order to fulfill the degree requirement, I am undertaking a research study in the area of Software Engineering. You have been selected to form part of this study. This is therefore to kindly request you to assist me collect the data by filling out the accompanying questionnaire, which I will collect from your premises.

The information provided will be used exclusively for academic and research purposes only. This will be kept in strict confidence. Kindly answer all questions. In case of any queries pertaining to this research, please do not hesitate to contact me on mobile phone: 0725-133-239 or email: mwakondopoly@gmail.com.

Thank you for your help.

Fullgence M. Mwakondo	Dr. Lawrence Muchemi	Prof. Elijah Omwenga
Candidate	Supervisor	Supervisor

APPENDIX C: QUESTIONNAIRES

Analysis Questionnaire A (for Exam Past Paper)

PART A: EXAMINATION INFORMATION

Please respond by ticking in the appropriate boxes or providing the appropriate information required.

1. What is the university name of the examination paper?

   Nairobi ☐ Kenyatta ☐ JKUAT ☐ Moi ☐

   Egerton ☐ Strathmore ☐ KEMU ☐ Daystar ☐

   If other, specify________________________________

2. What is the administration year of the examination paper?

   2014 ☐ 2013 ☐ 2012 ☐ 2011 ☐ 2010 ☐ 2009 ☐

   If other, specify________________________________

3. What is the time duration allocated for the examination paper?

   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 or more ☐

   If other, specify________________________________

4. What is the total mark allocated for the examination paper?

   60 ☐ 70 ☐ 80 ☐ 90 ☐ 100 ☐

   If other, specify________________________________

5. Which year of study is the examination paper administered?

   First ☐ Second ☐ Third ☐ Fourth ☐ Fifth ☐

6. What is the name of the undergraduate programme for which the examination was administered?

   Please specify________________________________

7. What is the number of the main questions in the examination paper?

   3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 or more ☐

   If other, specify________________________________

<u>PART B:</u> EXAMINATION CONTENT (KNOWLEDGE AND SKILLS WEIGHTS)

8. For each question in the exam paper fill in the marks allocated to each of its sections against the software development area tested.

Questions Details		Software development areas marks allocated									
Question number	Question sections	Software Requirements	Software Design	Software Process	Software Testing	Configuration Management	Software Maintenance	Software Infrastructure	Software Quality	Software Management	Software Construction
Q1	P1										
	P2										
	P3										
	P4										
Q2	P1										
	P2										
	P3										
	P4										
Q3	P1										
	P2										
	P3										
	P4										
Q4	P1										
	P2										
	P3										
	P4										
Q5	P1										
	P2										
	P3										
	P4										
If others, specify below and rate accordingly											

9. For each question in the exam paper fill in the marks allocated to each of its sections against the mental activities tested.

Question number	Mental activities		Question sections marks allocated					
Q1	Question sections		P1	P2	P3	P4	P5	P6
	Mental activities	Knowledge						
		Comprehension						
		Application						
		Analysis						
		Synthesis						
		Evaluation						
Q2	Question sections		P1	P2	P3	P4	P5	P6
	Mental activities	Knowledge						
		Comprehension						
		Application						
		Analysis						
		Synthesis						
		Evaluation						
Q3	Question sections		P1	P2	P3	P4	P5	P6
	Mental activities	Knowledge						
		Comprehension						
		Application						
		Analysis						
		Synthesis						
		Evaluation						
Q4	Question sections		P1	P2	P3	P4	P5	P6
	Mental activities	Knowledge						
		Comprehension						
		Application						
		Analysis						
		Synthesis						
		Evaluation						
Q5	Question sections		P1	P2	P3	P4	P5	P6
	Mental activities	Knowledge						
		Comprehension						
		Application						
		Analysis						
		Synthesis						
		Evaluation						

REFERENCE LIST

Knowledge Examples: **list, define, tell, identify, label, collect, tabulate, quote, name, state**

Comprehension Examples: **summarize, describe, interpret, contrast, associate, distinguish, estimate, discuss**

Application Examples: **apply, calculate, complete, illustrate, solve, modify, relate**

Analysis Examples: **separate, order, explain, classify, arrange, divide, compare, select**

Synthesis Examples: **combine, integrate, modify, rearrange, substitute, plan, create, design, invent, compose, formulate, rewrite, develop**

Evaluation Examples: **assess, choose, rank, grade, recommend, select, judge, support, conclude**

Questionnaire B (for Employed Software developers)

Questionnaire code……………
Date …………………………………
Time…………………………………

QUESTIONNARE

Questionnaire (for Employed Software developers)

This questionnaire is part of a study on Software development companies in Nairobi County**.** Your participation in this study is voluntary. The questions will purely be used to satisfy an academic requirement only, and not for any statistical study. We will not identify you as an individual. The researcher would be most grateful if you give your views by answering the questions below. Please, first answer the background questions and then complete the rest of the survey. Be assured that Confidentiality of information solicited is guaranteed.

Thank you

Instructions: Please read the questions and answer them by either filling in the blank Spaces or ticking the check boxes [/] or tables

<u>PART A:</u> PERSONAL BACKGROUND INFORMATION

Please respond by ticking in the appropriate boxes or providing the appropriate information required.

1. What is your gender?

   Male [ ] Female [ ]

2. Which of the following brackets does your age fall (in years)?

   20-24 [ ] 25-29 [ ] 30-34 [ ] 35-39 [ ] 40 or more [ ]

3. Where did you study for your 'O' level education?

   Local [ ] Abroad [ ]

4. Which system was used to grade your 'O' level results?

   Grades [ ] Points [ ] Marks [ ]

5. Which of the following brackets does your overall 'O' level education result fall?

   If Grades,

   Less or equal D+ [ ] C- *to* C+ [ ] B- *to* B+ [ ] A- *and above* [ ]

   If Points,

   Less or equal 4 [ ] 5 *to* 7 [ ] 8 *to* 10 [ ] 11 *and above* [ ]

   If marks,

   Less or equal 44% [ ] 45% *to* 59% [ ] 60% *to* 74% [ ] 75% *and above* [ ]

If other, specify___________________________________

6. What is the area of your undergraduate degree?

Computer Science ☐ Information Technology ☐ Software Engineering ☐

If other, specify____________________________

7. What is the university name of your undergraduate degree?

Nairobi ☐ Kenyatta ☐ JKUAT ☐ Moi ☐

Egerton ☐ Strathmore ☐ KEMU ☐ Daystar ☐

If other, specify___________________________________

8. What is the graduation year for your Bachelor's degree?

2014 ☐ 2013 ☐ 2012 ☐ 2011 ☐ 2010 ☐ 2009 ☐

If other, specify___________________________________

9. Which system was used to grade your undergraduate degree final result?

Grades ☐ Points ☐ Marks ☐

10. Which of the following brackets does your overall bachelor's degree final result fall?

If Grades,

Less or equal D+ ☐ C- *to* C+ ☐ B- *to* B+ ☐ A- *and above* ☐

If Points,

Less or equal 4 ☐ 5 *to* 7 ☐ 8 *to* 10 ☐ 11 *and above* ☐

If marks,

Less or equal 44% ☐ 45% *to* 59% ☐ 60% *to* 74% ☐ 75% *and above* ☐

If other, specify___________________________________

11. What is the title of your first (entry-level) Software development job appointment in the industry after graduating and current job title? Select from the table, or specify, and fill years of appointment for both.

		Tick only job categories that apply to you									
Tick V Appointment types and dates		Software architect/designe	Analyst/ programmer	Test analyst/ engineer	Web developer/ programmer	Mobile application developer/progra	Systems admin/ programmer	Project manager	Others, specify		
A	First Software Development job category appointment (Tick only one)	☐	☐	☐	☐	☐	☐	☐			
B	Year of Appointment in A, specify in cell										
C	Current Software Development job	☐	☐	☐	☐	☐	☐	☐			

	category appointment (Tick only one)										
D	Year of Appointment in B, specify in cell										

12. What inspired you to join the current Software development job?

Passion ☐ Salary☐ Ambition☐ Qualification ☐ If other, specify________

<u>PART B:</u> SOFTWARE DEVELOPMENT BACKGROUND INFORMATION

Please respond by ticking in the appropriate boxes or providing the appropriate information required.

13. Which year of your study did you study Software Engineering subject?

First ☐ Second ☐ Third ☐ Fourth☐ Fifth☐

Specify the year____________________________

14. To what extend do you think the software engineering exam paper reflected the content covered in class during training?

100% ☐ 75%☐ 50% ☐ 25%☐ 0% ☐

15. What grade did you score in the following Software development related subjects?

O = One unit		**T = Two units**				**M = More than 2**		**X=unit not done**		
Subject taught in one unit		Subject taught in two unit e.g. I, II, or advanced				Subject taught in more than two units		Subject not taught at all		
Subject Name	No. of units	Mark one grade for each unit				Subject Name	No. of units	Mark one grade for each unit		
		Unit 1	Unit 2	Other units				Unit 1	Unit 2	Other units
Software Development Project	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E			Operating Systems	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	
Database	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E			Structured Programming	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	

Distributed Systems	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E			Object Oriented Programming	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	
Networking	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E			Web-based Programming	☐ O ☐ T ☐ M ☐ X	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	

16. Which of the following activities is associated with your current job title? On a scale of 1=(less important) to 12=(most important), rate the relative importance of each of the following Software development areas on each of your job activities ticked.

Choose and tick job activities		Software development areas (fill their rated values along column)									
Tick V	Job activities below	Software Requirements	Software Design	Software Process	Software Testing	Configuration Management	Software Maintenance	Software Infrastructure	Software Quality	Software Management	Software Construction
1 ☐	Gathering and analyzing requirements										
2 ☐	Modeling and simulating software										
3 ☐	Designing database										
4 ☐	Designing systems										
5 ☐	Software Programming										
6 ☐	Integrating software										
7 ☐	Documenting programs										
8 ☐	Deploying software										
9 ☐	Testing software										
10 ☐	Training users										
11 ☐	Preparing manuals and user guides										
12 ☐	Documenting workflows										
13 ☐	Managing project workflows										
14 ☐	Coordinating project deliverables										
15 ☐	Ensuring software quality										
16 ☐	Providing customer and system support										
17 ☐	Upgrading and reviewing systems										
	If others, specify and rate accordingly:										

	____________________										

17. On a scale of 1=(less thinking demand) to 12= (very high thinking demand), rate the relative mental demand for each of your job activities (selected above) in terms of the following mental activities.

Tick only job activities as selected above		Mental activities (fill their rated values)					
Tick V	Job/Role Activities	Applying concepts	Remembering concepts	Understanding concepts	Analyzing concepts	Judging concepts	Modeling concepts
1 ☐	Gathering and analyzing requirements						
2 ☐	Modeling and simulating software						
3 ☐	Designing database						
4 ☐	Designing systems						
5 ☐	Software Programming						
6 ☐	Integrating software						
7 ☐	Documenting programs						
8 ☐	Deploying software						
9 ☐	Testing software						
10 ☐	Training users						
11 ☐	Preparing manuals and user guides						
12 ☐	Documenting workflows						
13 ☐	Managing project workflows						
14 ☐	Coordinating project deliverables						
15 ☐	Ensuring software quality						
16 ☐	Providing customer and system support						
17 ☐	Upgrading and reviewing systems						
	If others, specify below and rate accordingly:						

Questionnaire C (for Industry Experts)

Questionnaire code..............
Date ....
Time...

QUESTIONNARE

Questionnaire (for Software Development Head of Section)

This questionnaire is part of a study on Software development companies in Nairobi County**.** Your participation in this study is voluntary. The questions will purely be used to satisfy an academic requirement only, and not for any statistical study. We will not identify you as an individual. The researcher would be most grateful if you give your views by answering the questions below. Please, first answer the background questions and then complete the rest of the survey. Be assured that Confidentiality of information solicited is guaranteed.

Thank you

Instructions: Please read the questions and answer them by either filling in the blank

Spaces or ticking the check boxes [/] or tables

<u>PART A:</u> SOFTWARE FIRM BACKGROUND INFORMATION

Please respond by ticking in the appropriate boxes or providing the appropriate information required.

1. What is the ownership status of your firm?

   Local ☐ Foreign ☐ Both ☐

2. What is the number of software development staff in your firm in Kenya?

   1-5 ☐ 6-10 ☐ 11-15 ☐ 16-20 ☐ 20 or more ☐

3. What is the number of job title categories for software development in your firm in Kenya?

   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 or more ☐

4. What type of software products or service does your firm provide?

   ☐ Mobile applications ☐ Desktop applications ☐ Web applications

   ☐ Multipurpose applications if others, specify____________________________

5. Which of the following ICT job categories are offered as graduate level in your firm?

	Tick only job category offered in your firm (one or many)							
Tick ☑ V	☐	☐	☐	☐	☐	☐	☐	☐
	Software architect /designer	Test analyst /engineer	Mobile application developer /programmer	Project manager	Analyst/application programmer	Web developer /programmer	Systems admin /programmer	If others, specify ____
REQUIRMENT TYPE	**MINIMUM ENTRY REQUIRMENTS (Tick for each selected job category above)**							
Type of entry **GE**=Graduate Entry **GP**=Graduate Promotion **X**=Non-graduate	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X	☐ GE ☐ GP ☐ X
Secondary school grade (**A** = A- and Above **B** = B-, B, B+ **C** = C-, C, C+ **D** = D-, D, D+ **E** = E and Below)	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E	☐ A ☐ B ☐ C ☐ D ☐ E
Bachelors degree type (**1** = Computer Science **2** = IT, **3** = Any of the above, **4** = Any degree type	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4	☐ 1 ☐ 2 ☐ 3 ☐ 4
Degree Grade (**F** = First class, **U** = second Upper, **L** = second Lower, **P** = Pass, **A** = Any of the above)	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A	☐ F ☐ U ☐ L ☐ P ☐ A
Grade Quality (**S** = Strong, **W**= Weak)	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W	☐ S ☐ W

6. Which of the following job activities are associated with each of the job categories offered in your firm? Tick cells below the job category offered.

Tick V		Tick only job categories that apply in your firm as selected above									
Tick Job activities below that apply to the selected job category		Software architect/designer	Analyst/ programmer	Test analyst/ engineer	Web developer/ programmer	Mobile application developer/programmer	Systems admin/ programmer	Project manager	Others		
1	Gathering and analyzing requirements										
2	Modeling and simulating software										
3	Designing database										
4	Designing systems										
5	Software Programming										
6	Integrating software										
7	Documenting programs										
8	Deploying software										
9	Testing software										
10	Training users										
11	Preparing manuals and user guides										
12	Documenting workflows										
13	Managing project workflows										
14	Coordinating project deliverables										
15	Ensuring software quality										
16	Providing customer and system support										
17	Upgrading and reviewing systems										
	If others, specify and rate										

7. On a scale of 1=(less important) to 12= (most important), rate the relative importance of each of the following software development areas on each of the job categories offered in your firm.

V Tick Job categories below as they apply in your firm.		Software development areas									
		Software Requirements	Software Design	Software Process	Software Testing	Configuration Management	Software Maintenance	Software Infrastructure	Software Quality	Software Management	Software Construction
1	Software architect/developer										
2	Analyst/ programmer										
3	Test analyst/ engineer										
4	Web developer/ programmer										

5 ☐	Mobile application developer/programmer									
6 ☐	Systems admin/ programmer									
7 ☐	Project manager									
	If others, specify below and rate accordingly:									

8. On a scale of 1=(less thinking demand) to 12= (very high thinking demand), rate the relative mental demand of each of the following mental activities for each of the job categories offered in your firm.

V Tick Job categories below as they apply in your firm.		Mental activities					
		Applying concepts	Remembering concepts	Understanding concepts	Analyzing concepts	Judging concepts	Modeling concepts
1 ☐	Software architect/developer						
2 ☐	Analyst/ programmer						
3 ☐	Test analyst/ engineer						
4 ☐	Web developer/ programmer						
5 ☐	Mobile application developer/programmer						
6 ☐	Systems admin/ programmer						
7 ☐	Project manager						
	If others, specify below and rate accordingly:						

9. On a scale of 1=(less thinking demand) to 12= (very high thinking demand), rate the relative importance of each of the following skills for each of the job categories offered in your firm.

V Tick Job categories below as they apply in your firm.		Software development skills							
		Database skills	Programming skills	Distributed skills	Networking skills	Platform skills	Others, specify		
1 ☐	Software architect/developer								
2 ☐	Analyst/ programmer								
3 ☐	Test analyst/ engineer								
4 ☐	Web developer/ programmer								
5 ☐	Mobile application developer/ programmer								
6 ☐	Systems admin/ programmer								
7 ☐	Project manager								
	If others, specify and rate:								

10. Is there a hierarchical organization structure that describes the software development job categories in your firm?
Yes ☐ No ☐

If Yes, provide the structure by sketching below or attach printed copy.

APPENDIX D: SE EXAMS PAST PAPERS SAMPLING FRAME

ACCREDITED UNIVERSITIES AND ACADEMIC PROGRAMMES			
	Sunday, February 01, 2015		
No.	INSTITUTION	DETAILS OF SOFWARE ENGINEERING OFFERRING DEGREE PROGRAMMES	
		No.	DEGREE PROGRAMMES
1	UNIVERSITY OF NAIROBI	1	Bachelor of Science in Computer Science
2	MOI UNIVERSITY	2	Bachelor of Science (Computer Science)
		3	Bachelor of Science (Information sciences)
		4	Bachelor of science in Computer Engineering
		5	Bachelor of Science (Informatics)
3	KENYATTA UNIVERSITY	6	Bachelor of Science in Computer Science
		7	Bachelor of science in Computer Engineering
		8	Bachelor of Science in Information Technology
		9	Bachelor of Information Technology
4	EGERTON UNIVERSITY	10	Bachelor of Science in Applied Computer Science
		11	Bachelor of Science in Computer Science
		12	Bachelor of Science in Software Engineering
5	JKUAT	13	Bachelor of Business Information Technology
		14	Bachelor of Science in Computer Science
		15	Bachelor of Science in Computer Technology
		16	Bachelor of Science in Information Technology
6	MASENO UNIVERSITY	17	Bachelor of Science in Computer Science
		18	Bachelor of Science in Information Technology
7	MASINDE MULIRO UNIVERSITY OF SCIENCE AND TECHNOLOGY	19	Bachelor of Science in Information Technology
		20	Bachelor of Science in Computer Science
8	DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY	21	Bachelor of Business Information Technology
		22	Bachelor of Science in Computer Science
		23	Bachelor of Science in Information Technology
9	CHUKA UNIVERSITY	24	Bachelor of Science (Computer Science)
10	TECHNICAL UNIVERSITY OF KENYA	25	Bachelor of Technology (Business Information Technology)
		26	Bachelor of Technology (Information Technology)
		27	Bachelor of Technology (Computer Technology)
11	TECHNICAL UNIVERSITY OF MOMBASA	28	Bachelor of Mathematics & Computer Science
		29	Bachelor of Science in Information Technology
		30	Bachelor Technology in Inform. & Communication Technology
12	PWANI UNIVERSITY	31	Bachelor of Science (Computer Science)
13	KISII UNIVERSITY	32	Bachelor of Applied Computer Science
		33	Bachelor of Computer Science
		34	Bachelor of Business Information Management
		35	Bachelor of Software Engineering
14	UNIVERSITY OF ELDORET	36	Bachelor of Science in Computer Science
		37	Bachelor of Science in Informatics
		38	Bachelor of Science in Information Technology
15	MAASAI MARA UNIVERSITY	39	Bachelor of Science (Computer Science)
16	JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE AND TECHNOLOGY	40	Bachelor of Science (Business Information Systems)
		41	Bachelor of Science (Information Communication Technology)
17	LAIKIPIA UNIVERSITY	42	Bachelor of Science (Computer Science)
18	SOUTH EASTERN KENYA UNIVERSITY	43	Bachelor of Information Technology
		44	Bachelor of Science (Computer Science)

19	**MERU UNIVERSITY OF SCIENCE AND TECHNOLOGY**	45	Bachelor of Business Information Technology
		46	Bachelor of Science in Computer Science
		47	Bachelor of Science in Computer Technology
		48	Bachelor of Science in Information Technology
		49	Bachelor of Science in Mathematics and Computer Science
20	**MULTIMEDIA UNIVERSITY OF KENYA**	50	Bachelor of Information Technology
		51	Bachelor of Science and Business Information Technology
		52	Bachelor of Science and Information Technology
		53	Bachelor of Science Computer Science
		54	Bachelor of Science Computer Technology
		55	Bachelor of Science Mathematics & Computer
21	**UNIVERISTY OF KABIANGA**	56	Bachelor of Science in Computer Science
22	**KARATINA UNIVERSITY**	57	Bachelor of Science in Computer Science
		58	Bachelor of Science in Information Technology
23	**UNIVERSITY OF EASTERN AFRICA BARATON**	59	Bachelor of Business Information Technology
		60	Bachelor of Science in Software Engineering
24	**CATHOLIC UNIVERSITY OF EAST AFRICA**	61	Bachelor of Science in Computer Science
25	**DAYSTAR UNIVERSITY**	62	Bachelor of Science in Applied Computer Science
26	**UNITED STATES INTERNATIONAL UNIVERSITY**	63	Bachelor of Science in Information Science and Technology
27	**AFRICA NAZARENE**	64	Bachelor of Science in Computer Science
		65	Bachelor of Business and Information Technology
28	**KENYA METHODIST UNIVERSITY**	66	Bachelor of Science in Mathematics and Computer Science
		67	Bachelor of Business Information Technology
29	**ST PAUL'S UNIVERSITY**	68	Bachelor of Business Information Technology
		69	Bachelor of Science in Computing and Information Systems
30	**STRATHMORE UNIVERSITY**	70	Bachelor of Science in Informatics
		71	Bachelor of Business Information Technology
31	**KABARAK UNIVERSITY**	72	Bachelor of Science in Computer Science
		73	Bachelor of Business and Information Technology
		74	Bachelor of Science in Information Technology
32	**MOUNT KENYA UNIVERSITY**	75	Bachelor of Business Information Technology
34	**KCA UNIVERSITY**	76	Bachelor of Science in Information Technology
		77	Bachelor of Business Information Technology
35	**KIRIRI WOMEN'S UNIVERSITY OF SCIENCE AND TECHNOLOGY**	78	Bachelor of Science in Computer Science
36	**GRETSA UNIVERSITY**	79	Bachelor of Science in Computer Science
37	**PRESBYTERIAN UNIVERSITY OF EAST AFRICA**	80	Bachelor of Science in Computer Science
38	**INOORERO UNIVERSITY**	81	Bachelor of Information and Communication Technology
39	**THE EAST AFRICAN UNIVERSITY**	82	Bachelor of Computer Science and Information Technology
		83	Bachelor of Business Information Technology
40	**RIARA UNIVERSITY**	84	Bachelor of Science in Computer Science
41	**PIONEER UNIVERSITY**	85	Bachelor of Science in Information Technology
42	**UMMA UNIVERSITY**	86	Bachelor of Science in Computer Science
43	**ZETECH UNIVERSITY**	87	Bachelor of Science in Information Technology
TOTAL OF 43 UNIVERSITIES		**87**	**PROGRAMMES**

APPENDIX E: SOFTWARE DEVELOPERS' SAMPLING FRAME

Software Houses in Kenya (source: www.softkenya.com)

S/NO	COMPANY NAME	TELEPHONE	FAX/MOBILE	EMAIL
1	Abacus Computer Systems Ltd.	– 2 – 213740/ 214450/ 312491	215 – 2 – 221321	sales@abacuscom.com
2	Afritech Solutions Ltd.	+254 020-2129035		
3	Alphabit Technologies	020 2470510	0750220736	info@alphabitkenya.com
4	Andest Bites	254-020-2394420	254-0733720619,0724164346	info@andestbites.com
5	Bridge Ict		0726178724	
6	Bunduz Creative		254723571032	fraogongi@yahoo.co.uk
7	Compulynx	+254-20-3747060	+ 254-20-3747280	sales@Lynxafrica.com
8	Copycat Limited	+254 20 3970000/ +254 20 534008-15/ +254 20 3970000	+254 20 652276/ 554249	info@copycatltd.com sales@copycatltd.com
9	Daniche Solutions	0202605564		info@daniche.co.ke
10	Designjobs Interactive Media	+254 020 245 3230		
11	Digital Horizons Ltd	+254 20 2062457,	+254 722 305680	info@dhkenya.com
12	East Africa Data Handlers Ltd	+254-20-3751400/ 3751402	+254-722435163, +254-720-776840 / +254-726-643116 Fax: +254-20-3751400/ 3751402	info@datarecovery.co.ke
13	Ebits Online	(254) 20 2384022	(254) 721 985408 (254) 738 168248	Email: info@ebitsonline.com
14	Empire Microsystems Ltd	254-(020)-352 5210 , 254-(020)-247 2011	(+254) 723 782 505 (+254) 721 815 466 (+254) 727 709 772	info@empire.co.ke
15	Endeavour Africa Kenya	+254 (20) 375 2451 / 239 4959	+254 (734) 446 600 / (714) 446 600 Fax: +254 (20) 375 2458	info@endeavourafrica.com
16	Enfinite Solutions Limited	020-2603710		
17	Enterprise Information Management Solutions (EIM)	+254-20-2730900	+254-20-2731058	info@eimsolutions.co.ke
18	ESRI Eastern Africa	+254 (0) 20 2713630, 2713631, 2713632	+254 (0) 722 521341, 733 568381 Fax: +254 (0) 20 2713633	sales@esriea.co.ke
19	Extend Limited	Tel: 0202329194, 0202329195		info@extend.co.ke
20	Footprint Computer Solutions Limited	254 020 2727510/2727511	254 020 2727512	info@footprintebusiness.com
21	Freelance Web Developer	+254733438933		
22	Freepac Tech	+254-20-4452691	0720 405 201 ,0721 617049	:info@frepactech.com,Sales@frepactech.com
23	Gem Multimedia Ltd	+254 721 818 345 / 254 202 777 847		info@gem.co.ke
24	MAGNUM	254724348990		
25	Octagon Data System	+254-020-2719733/2738708,	+25420-2730675	info@octagon.co.ke
26	Octopus ICT Solutions Ltd.	0206007423		info@octopusict.com
27	Passive Software Technologies Limited	+ 254 020 2485696		sales@softwares.co.ke
28	Peak and Dale Solutions Ltd	020 2216522	0722216522	info@peakanddale.com
29	Pinecrest Studios		0727163765	Emungai@pinecrest.co.ke
30	Rapid Applications Developers	0206760918		info@rad.co.ke
31	Snettscom | innovative web solutions		0723934017, 0725562184	info@snetts.com
32	Softlink Options	+254 (020) 3559522	+254 – 0722810084	felista@softlinkoptions.com
33	Software Technologies Ltd	+ 254 20 7122971/2/3 Fax: + 254 20 7122991		marketing@stl-horizon.com
34	Symbiotic Media Consortium	+254 20 359 6305		business@symbiotic.co.ke
35	Symphony	(+254) 20 – 4455000	(+254) 722 – 205456/7, (+254) 733 – 605739/40 Fax: (+254) 20 – 4453067/8	info@symphony.co.ke (General) enquiries@symphony.co.ke (
36	Synfotech Technologies Kenya		0722270423	

37	Techbiz Ltd	+254-20-2724916	+254-20-2724919	info@technic.co.ke
38	TK Professional Computer Services	0602030707	0725417111	tkcomputersp@gmail.com
39	Track and Trace Kenya Ltd.	254 20 2042628	254 720 844 638 Fax: 254 20 2250969	info@trackntrace.co.ke
40	Web Professional Services	0726-476-620	0726-476-620	gsimiy@gmail.com
41	WebSoft Development	254 (20) 249 2470	254 722 407 837	info@websoftdevelopment.com
42	WebSpaceKenya IT Solutions	254202384600	254-724-557 399	info@webspacekenya.com
43	ZeboTech Business Solutions	(254) 02-2177372	(254)771047405	ict@zebotech.co.ke

APPENDIX F: RESEARCH PERMIT

NACOSTI

NATIONAL COMMISSION FOR SCIENCE, TECHNOLOGY AND INNOVATION

Telephone:+254-20-2213471,
2241349,3310571,2219420
Fax:+254-20-318245,318249
Email:dg@nacosti.go.ke
Website: www.nacosti.go.ke
when replying please quote

9th Floor, Utalii House
Uhuru Highway
P.O. Box 30623-00100
NAIROBI-KENYA

Ref. No. **NACOSTI/P/16/9782/14472**

Date:

7th November, 2016

Fullgence Mwachoo Mwakondo
University of Nairobi
P.O. Box 30197-00100
NAIROBI.

RE: RESEARCH AUTHORIZATION

Following your application for authority to carry out research on "*A model for mapping graduates skills to industry roles using machine learning techniques, a case study of software engineering,*" I am pleased to inform you that you have been authorized to undertake research in **Nairobi County** for the period ending **7th November, 2017.**

You are advised to report to **the County Commissioner and the County Director of Education, Nairobi County** before embarking on the research project.

On completion of the research, you are expected to submit **two hard copies and one soft copy in pdf** of the research report/thesis to our office.

BONIFACE WANYAMA
FOR: DIRECTOR-GENERAL/CEO

Copy to:

The County Commissioner
Nairobi County.

The County Director of Education
Nairobi County.

National Commission for Science, Technology and Innovation is ISO 9001: 2008 Certified

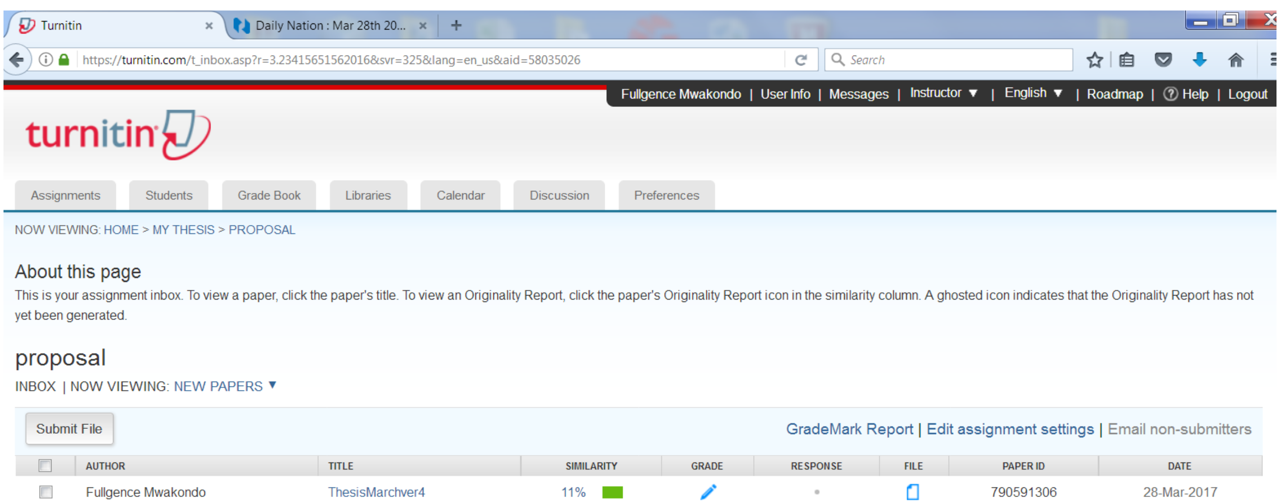
APPENDIX G: TURNIT REPORT

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Fullgence Mwakondo
Assignment title:	proposal
Submission title:	ThesisMarchver4
File name:	PHDProposalREVISEDuonFINAL_M..
File size:	4.65M
Page count:	166
Word count:	51,508
Character count:	288,849
Submission date:	28-Mar-2017 12:51PM
Submission ID:	790591306



APPENDIX H: SE BENCHMARK DATASET

'GENDER'	'AGE'	'LOLE'	'BDGREE'	'ROLE'	'GSOLE'	'GSBDEGREE'	'UNIVERSITY'	'RBACHELORS'	'R'	'D'	'A'	'C'	'CLASS'
2	2	2	1	3	2	44	2	3	11	1.3	6.7	9	1
2	1	2	2	3	2	51	2	4	10.1	1	7.3	10.5	1
2	2	2	1	3	1	1621	2	4	7.5	1.2	5.9	10.5	1
2	2	2	2	3	2	8297	2	4	10.9	1.4	5.3	10.5	1
2	2	2	1	3	1	8350	2	3	1	1.1	5.6	9	1
2	2	2	2	4	1	9737	2	4	11.4	1.1	6.2	12	1
2	2	2	1	3	1	10185	2	4	12	1.3	8.8	10.5	1
2	2	2	1	3	1	10932	2	3	1.7	1	4.9	9	1
2	2	2	1	4	1	13344	2	4	11	1.5	6.7	12	1
2	2	2	1	3	2	350	2	3	1.9	1.9	1.1	4.5	2
2	2	2	7	4	1	893	2	3	3	2.3	1.5	5.3	2
2	2	2	0	4	2	3838	2	3	2.7	1.8	1.3	5.3	2
1	2	2	2	4	1	4973	2	4	0.1	1.5	1	6	2
2	2	2	1	3	1	5147	2	3	2.6	2.5	1.3	4.5	2
1	1	2	2	2	2	5508	2	4	2.7	2.1	0.8	4.5	2
2	2	2	16	2	2	6567	2	3	1.5	1.7	1	3.8	2
2	2	2	1	4	1	6907	2	3	3	2.8	1.5	5.3	2
2	2	2	7	3	2	7134	2	3	0.3	1.8	0.7	4.5	2
2	2	2	1	3	2	7626	2	3	1.5	2.1	0.8	4.5	2
1	2	2	2	3	1	9128	2	3	1.1	1.9	1.1	4.5	2
1	2	2	1	3	2	9256	2	2	2.7	2.5	1	3.8	2
1	2	2	2	4	1	9769	2	3	1.5	2	1	5.3	2
2	2	2	4	3	2	11759	2	3	2.4	2.2	1.2	4.5	2
2	2	2	2	3	2	55	2	3	1.5	0.6	1.3	3	3
1	1	2	2	4	2	137	2	3	2.4	0.6	1.3	3.5	3
2	2	2	1	3	2	172	2	3	1.8	0.6	1.6	3	3
2	2	2	1	3	1	184	2	3	1.4	0.7	1.4	3	3
2	2	2	2	3	1	184	2	3	1.2	0.5	1.5	3	3
2	2	2	2	4	2	236	2	3	0.6	0.6	1.7	3.5	3
2	2	2	25	2	2	272	2	3	1.4	0.6	1.3	2.5	3
1	2	2	1	4	2	429	2	4	2.1	0.8	1.6	4	3
2	2	2	2	3	1	982	2	3	0.6	0.5	1.4	3	3
2	2	2	2	4	2	3725	2	3	2	0.7	1.3	3.5	3
2	2	2	1	3	1	5904	2	3	1.4	0.6	1.3	3	3
2	1	2	1	3	2	5904	2	3	0.9	0.6	1.3	3	3
1	2	2	2	3	2	6294	2	3	1.5	0.5	1.3	3	3
2	2	2	1	2	1	6741	2	3	2.2	0.6	1.7	2.5	3
2	2	2	1	2	2	7376	2	3	1.3	0.7	1.3	2.5	3
1	2	2	4	4	2	10051	2	3	1.5	0.7	1.3	3.5	3
2	2	1	1	4	1	11127	1	3	2.2	0.8	1.9	3.5	3
1	2	2	2	4	1	11516	2	3	0.9	0.7	1.5	3.5	3
2	2	2	2	3	2	11664	2	4	1.5	0.5	1.1	3.5	3
1	1	1	1	4	1	12232	1	3	1.3	0.8	1.3	3.5	3
2	2	2	2	3	1	13147	2	4	1.8	0.7	1	3.5	3
2	2	2	2	4	1	15401	2	3	1.8	0.7	1.6	3.5	3
2	2	2	1	3	1	16347	2	4	2.2	0.8	1.6	3.5	3
2	2	1	1	4	1	47	1	4	3.8	0.9	2.3	3	4
2	2	1	1	4	1	47	1	4	3.7	0.9	2.2	3	4
2	2	2	2	4	1	52	2	3	3.4	0.8	2.1	2.6	4
2	2	2	1	3	2	96	2	3	1.4	0.7	1.9	2.3	4
2	2	2	2	4	1	108	2	4	4	0.9	2.6	3	4
2	2	2	0	3	1	220	2	3	0.3	0.5	1.4	2.3	4
2	1	2	2	4	1	272	2	4	3.8	0.8	2.3	3	4
2	2	2	2	4	2	315	2	3	3.5	0.7	2.1	2.6	4
2	1	2	2	4	1	434	2	4	1.4	0.5	1.3	3	4
2	2	2	2	3	1	439	2	3	1.5	0.7	1.5	2.3	4
2	2	2	4	4	1	485	2	4	3.9	0.9	2.5	3	4
1	2	2	14	4	2	974	2	3	2.9	0.7	1.9	2.6	4
2	2	2	2	4	1	1111	2	4	3.4	0.7	1.9	3	4
2	2	2	2	3	1	1219	2	3	3	0.7	2	2.3	4
2	2	2	1	3	2	1995	2	2	3.1	1	1.9	1.9	4
2	2	2	2	4	1	3668	2	3	3.9	0.6	1.7	2.6	4
2	2	2	12	3	1	3741	2	3	3.8	0.9	2.3	2.3	4
1	1	2	2	4	1	3931	2	4	3.9	0.8	2.5	3	4
2	2	2	2	3	2	4793	2	3	3.6	0.5	2	2.3	4
2	2	2	2	4	2	5338	2	3	3	0.8	2.1	2.6	4
2	2	2	1	3	1	5400	2	3	3.1	0.7	1.9	2.3	4
1	2	2	1	4	1	6874	2	4	3	0.8	2.2	3	4
1	2	2	2	4	1	7269	2	4	3.5	0.8	2.1	3	4
2	2	2	1	3	1	7376	2	3	1.1	0.7	1.4	2.3	4
2	2	2	1	2	1	7564	2	3	3.5	0.7	2.1	1.9	4
2	2	2	7	2	2	7627	2	3	3.3	0.8	2	1.9	4
1	1	2	2	4	1	9198	2	4	3.4	0.8	1.9	3	4
1	2	2	2	3	1	11000	2	3	1.9	0.6	1.6	2.3	4
2	1	2	2	3	1	11630	2	3	3.4	0.7	1.4	2.3	4
2	2	2	1	4	1	11759	2	3	3.4	0.7	2.1	2.6	4
1	2	2	2	4	1	11788	2	3	0.6	0.5	1.4	2.6	4
2	1	2	2	3	1	12515	2	3	0.2	0.5	2	2.3	4
2	2	2	20	3	1	12515	2	3	0.6	0.7	1.8	2.3	4
2	1	2	2	4	1	12867	2	3	1	0.7	2	2.6	4
2	2	2	1	3	1	13543	2	3	1.8	0.7	1.6	2.3	4
2	2	2	2	3	1	13697	2	4	2.7	0.6	1.8	2.6	4
2	2	2	2	3	2	14587	2	3	3.4	0.7	1.8	2.3	4
2	2	2	2	3	2	15041	2	3	2.1	0.7	1.7	2.3	4
2	2	2	2	3	1	15863	2	3	3.5	0.8	2.1	2.3	4
1	1	2	2	4	1	16183	2	3	1.9	0.5	1.7	2.6	4
2	2	2	1	3	2	350	2	3	1.7	1.6	0.9	1.8	5
2	2	2	2	3	2	431	2	3	0.8	1.1	0.7	1.8	5
2	2	2	1	3	2	914	2	3	1.4	2.2	0.8	1.8	5
2	2	2	23	2	2	993	2	2	0.6	1.3	0.6	1.2	5
2	2	2	2	4	2	1111	2	3	1	1.1	0.5	2.1	5
2	1	2	2	3	1	1282	2	2	1	1.5	0.7	1.5	5
2	1	2	2	4	1	1759	2	4	1.5	1.9	1	2.4	5
2	3	2	1	3	1	2673	2	3	1	1.6	0.7	1.8	5
2	2	2	2	3	1	4795	2	3	2	1.7	1.1	1.8	5
2	2	2	2	4	1	5752	2	3	1.3	1.3	0.9	2.1	5
1	2	2	2	4	1	6996	2	3	0.4	1.5	0.5	2.1	5
2	2	2	1	2	1	7428	2	3	1.3	1.5	0.8	1.5	5
2	2	2	1	3	2	8310	2	3	1.5	1.8	0.8	1.8	5
2	2	2	28	3	2	44	2	3	0.4	0.7	0.5	1.5	6
2	2	2	1	3	1	53	2	3	0.5	0.9	0.5	1.5	6
2	2	2	27	4	2	64	2	3	0.3	0.9	0.5	1.8	6
2	2	2	2	3	1	67	2	3	1	1	0.6	1.5	6
1	2	2	1	4	2	165	2	4	0.5	0.8	0.6	2	6
2	2	2	4	3	2	172	2	3	0.7	0.9	0.7	1.5	6
2	2	2	1	3	2	255	2	3	0.5	0.9	0.5	1.5	6
2	2	2	1	4	1	272	2	3	0.8	0.9	0.6	1.8	6
1	2	2	1	3	1	387	2	3	0.9	0.8	0.7	1.5	6

2	2	2	1	3	1	387	2	3	0.3	0.8	0.5	1.5	6
2	2	2	0	4	2	429	2	4	0.5	0.8	0.5	2	6
2	2	2	6	3	2	547	2	3	0.7	0.7	0.6	1.5	6
1	2	1	1	4	1	849	1	4	0.5	1.1	0.9	2	6
2	3	2	1	2	1	1228	2	2	0.7	1	0.6	1	6
2	1	2	2	4	1	1843	2	3	0.2	0.8	0.6	1.8	6
2	2	2	24	3	2	2774	2	3	0	0.7	0.5	1.5	6
1	2	2	2	3	2	3579	2	3	0.2	0.7	0.4	1.5	6
2	2	2	0	4	2	3666	2	3	0.4	0.8	0.5	1.8	6
2	3	2	1	3	2	4023	2	3	0.7	0.9	0.6	1.5	6
2	2	2	18	3	2	4042	2	3	0.3	0.9	0.5	1.5	6
2	1	2	2	4	2	4805	2	3	0.7	0.8	0.5	1.8	6
1	2	2	1	2	2	6741	2	3	0.3	0.7	0.4	1.3	6
2	2	2	2	2	2	6835	2	3	0.5	0.7	0.4	1.3	6
1	2	2	1	3	2	7770	2	4	0.5	1	0.5	1.8	6
2	2	2	2	4	2	8011	2	3	0.3	0.5	0.5	1.8	6
2	2	2	7	3	1	8718	2	2	0.5	0.8	0.5	1.3	6
2	2	2	2	4	1	9122	2	3	0.5	1	0.5	1.8	6
2	1	2	2	4	2	9748	2	3	0.8	0.8	0.6	1.8	6
2	2	2	2	4	1	9803	2	4	0.2	0.8	0	2	6
1	2	2	1	3	1	9837	2	4	0.1	0.9	0.5	1.8	6
2	1	2	1	3	1	10971	2	3	0.5	0.8	0.6	1.5	6
1	2	2	17	3	2	11637	2	3	1	0.7	0.7	1.5	6
2	1	2	1	3	1	11759	2	4	0.5	0.9	0.7	1.8	6
2	2	2	2	4	1	11852	2	3	1	0.9	0.7	1.8	6
1	1	2	1	4	2	12061	2	3	0.5	0.8	0.5	1.8	6
2	1	2	2	3	2	12497	2	3	0.7	0.8	0.5	1.5	6
2	2	2	2	3	1	13473	2	3	0.8	1	0.6	1.5	6
1	2	2	1	4	2	13478	2	4	0.7	1	0.6	2	6
2	2	2	2	3	2	14662	2	3	0.2	0.8	0.4	1.5	6
2	2	2	2	4	2	16097	2	3	0.6	0.7	0.6	1.8	6
1	2	2	2	2	1	17168	2	3	0.7	0.7	0.6	1.3	6
2	2	2	1	4	1	17205	2	2	0.4	0.8	0.6	1.5	6
2	2	2	2	3	2	279	2	3	1.1	0.5	0.5	1.3	7
1	2	2	0	3	2	1019	2	4	0.3	0.4	0.4	1.5	7
2	2	2	0	3	2	1802	2	3	0.8	0.6	0.5	1.3	7
2	2	2	2	4	1	4557	2	3	0.5	0.5	0.3	1.5	7
1	2	2	1	4	1	6874	2	4	1.3	0.6	0.4	1.7	7
1	2	2	0	4	2	8011	2	3	0.4	0.6	0.4	1.5	7
2	2	2	1	4	1	11154	2	4	1.3	0.8	0.6	1.7	7
1	1	2	2	4	2	11788	2	4	0.5	0.6	0.4	1.7	7
2	2	2	1	4	2	13543	2	3	0.3	0.6	0.3	1.5	7
2	2	2	26	3	2	129	2	3	0.4	0.4	0	1.1	8
2	3	2	2	3	2	1995	2	2	0.2	0.4	0.3	0.9	8
1	2	2	24	3	2	4501	2	3	0.3	0.5	0.4	1.1	8
2	2	2	2	4	2	4566	2	3	0.2	0.4	0.3	1.3	8
1	2	2	2	4	2	7500	2	4	0.5	0.4	0.5	1.5	8
2	2	2	2	4	1	9129	2	3	0.6	0.5	0.4	1.3	8
1	2	2	0	3	2	11425	2	3	0.8	0.4	0.5	1.1	8
2	2	2	1	4	1	17935	2	3	0.5	0.6	0.5	1.3	8
1	2	1	1	4	2	34	1	3	1.3	0.6	0.6	1.2	9
1	2	2	2	3	1	51	2	3	0.8	0.5	0.5	1	9
2	1	2	1	3	1	184	2	3	0.4	0.5	0.6	1	9
2	2	2	2	3	2	184	2	3	0.5	0.5	0.4	1	9
2	2	2	1	2	1	429	2	3	0.7	0.5	0.5	0.8	9
2	2	2	1	4	1	527	2	3	1.2	0.6	0.6	1.2	9
2	2	2	2	3	1	2921	2	3	1	0.6	0.6	1	9
2	2	2	2	3	1	4217	2	2	1.1	0.4	0.3	0.8	9
1	1	2	2	4	1	5812	2	4	1.4	0.6	0.6	1.3	9
2	2	2	2	2	1	6545	2	2	0.5	0.4	0.4	0.7	9
2	2	2	2	3	2	7299	2	4	1.1	0.5	0.6	1.2	9
1	1	2	7	3	1	8718	2	4	1.4	0.5	0.6	1.2	9
2	2	2	4	3	2	8776	2	3	0.5	0.7	0	1	9
2	2	2	1	3	1	10859	2	3	0.5	0.5	0.6	1	9
2	1	2	2	3	1	11659	2	4	0.5	0.5	0.4	1.2	9
2	2	2	2	3	1	11759	2	4	1	0.5	0.6	1.2	9
2	2	2	2	4	1	12187	2	4	0.6	0.5	0.5	1.3	9
1	2	2	1	3	1	14351	2	4	0.9	0.6	0.5	1.2	9
2	1	2	2	3	2	16213	2	3	0.5	0.4	0.5	1	9
2	2	2	1	3	2	55	2	3	0.2	1	0.9	0.9	10
1	1	2	28	3	1	57	2	3	0.6	1	0.8	0.9	10
2	2	2	22	2	1	64	2	3	1.1	1	1.1	0.9	10
2	2	2	1	3	2	175	2	4	0.5	0.9	0.8	0.9	10
2	2	2	1	3	2	314	2	3	1.1	1.3	1.1	0.9	10
2	2	2	1	4	2	350	2	2	1.1	1.1	1.2	0.8	10
2	2	2	1	4	1	382	2	3	0.4	0.8	0.7	0.9	10
1	2	2	2	3	2	485	2	3	0.1	1.2	0.9	1.1	10
2	2	2	1	3	1	1018	2	3	1.2	1.1	1	1.1	10
2	2	2	1	4	1	1759	2	4	0.9	0.9	1.2	1.1	10
2	1	2	1	3	1	1940	2	3	1.1	1.1	1	0.9	10
2	1	2	2	3	1	2783	2	3	0.2	0.9	0.7	0.9	10
1	2	1	2	4	1	2988	1	4	0.2	1.2	0.9	1.2	10
1	2	2	2	3	2	3717	2	3	1	1.1	1	0.9	10
2	2	2	7	3	2	4043	2	1	0.8	1	0.9	0.6	10
2	2	2	1	2	1	4319	2	4	0.2	1.1	1.1	1.1	10
2	2	2	2	3	1	4417	2	4	0.3	0.8	0.8	0.9	10
2	2	2	1	3	1	4501	2	3	0.9	1.1	1	0.9	10
2	2	2	2	4	1	5081	2	3	0.3	0.8	0.7	0.9	10
2	2	2	2	4	2	5815	2	3	0.7	1	0.9	1.1	10
1	2	2	1	2	2	5904	2	3	0.9	1.3	1	1.1	10
1	2	2	22	2	2	6345	2	3	0.5	0.6	0.8	0.8	10
2	2	2	1	4	1	6545	2	3	1.1	1.2	1.1	1.1	10
2	2	2	1	3	1	6624	2	3	0.4	0.7	0.7	0.9	10
2	2	2	2	3	1	7783	2	3	0.8	0.9	1	0.9	10
2	2	2	2	3	1	8203	2	2	0.7	0.9	0.9	0.8	10
2	2	2	2	2	1	8949	2	3	0.9	1	1	0.8	10
2	2	2	1	4	2	10930	2	3	0.4	1	1.2	1.1	10
2	2	2	2	4	2	11603	2	3	-1.2	1.1	0	1.1	10
2	2	2	1	3	1	11759	2	3	0.8	1.1	1.2	0.9	10
2	2	2	1	2	1	14720	2	3	1	0.9	1	0.8	10
2	2	2	2	3	2	16687	2	3	0.3	0.9	0.7	0.9	10
2	2	2	1	3	1	28	2	3	5.8	6.7	3.6	0.8	11
2	3	2	4	4	2	137	2	3	3.8	5.5	2.7	1	11
1	2	2	1	3	1	184	2	4	3.6	5.4	2.6	1	11
2	2	2	0	3	2	252	2	4	3.4	5.3	2.6	1	11
2	2	2	1	3	1	272	2	4	5.2	7.1	3.2	1	11
1	2	2	4	4	1	272	2	3	2.2	7.9	3.2	1	11
2	1	2	1	4	1	272	2	4	5.9	6.8	3.7	1.1	11
2	1	2	1	4	1	272	2	4	5.7	6.9	3.2	1.1	11
2	1	2	1	4	1	272	2	4	6	6.3	3.4	1.1	11
2	2	2	24	4	1	462	2	3	5.5	6.3	3.2	1	11
1	2	2	0	4	2	501	2	3	3.4	7	2.6	1	11
2	1	2	2	4	1	883	2	3	5.1	6.5	3.2	1	11

2	2	2	2	3	2	1087	2	3	2.7	6.6	2.4	0.8	11
2	1	2	22	4	1	1237	2	4	2.2	6.8	2.5	1.1	11
1	2	2	2	4	2	1428	2	3	2.9	5.6	2.2	1	11
1	2	2	22	4	2	1764	2	4	3.6	5.6	2.6	1.1	11
2	2	2	4	4	1	1906	2	4	5.4	8.4	3.3	1.1	11
2	2	2	2	3	2	1906	2	4	4.8	7.2	3	1	11
2	2	2	1	3	1	2009	2	4	1.5	5.6	3.1	1	11
1	1	2	2	3	1	2041	2	3	2.9	3.6	2.4	0.8	11
1	2	2	1	4	1	2925	2	4	5.9	5.1	3.9	1.1	11
1	2	2	0	4	2	3076	2	4	5.6	6.4	3.4	1.1	11
2	2	2	11	4	2	3076	2	4	5.7	6.2	3.5	1.1	11
1	2	2	0	4	2	3076	2	4	5.3	6.3	3.2	1.1	11
2	2	1	2	3	2	3136	1	3	5.5	5.8	3.4	0.8	11
2	2	2	1	3	1	3449	2	3	4.9	6.1	3.1	0.8	11
1	2	2	0	4	2	3670	2	4	1.5	5.5	2	1.1	11
2	2	2	1	3	1	3905	2	3	4.5	5.7	3.2	0.8	11
2	2	2	4	3	2	4439	2	3	3.8	7.1	2.6	0.8	11
2	2	2	7	3	2	4439	2	3	4.8	7	3	0.8	11
2	2	2	1	4	1	4971	2	3	5.1	7.3	3.4	1	11
2	2	2	1	3	1	5056	2	4	4.5	7.5	3.7	1	11
2	3	2	0	3	2	5400	2	3	5.5	6.7	3.4	0.8	11
2	2	2	1	4	2	5400	2	3	5.9	6.9	3.7	1	11
1	2	2	1	4	2	5812	2	4	5.7	7.7	3	1.1	11
2	2	2	17	4	2	6857	2	3	0.5	4.2	2	1	11
2	2	2	2	4	2	6884	2	4	2.2	6	3.5	1.1	11
1	2	2	2	4	1	6948	2	4	2.9	6.6	2.9	1.1	11
2	2	2	2	3	2	8116	2	3	5.7	6.4	2.7	0.8	11
2	2	1	1	3	2	8195	1	4	4.4	6.4	2.9	1	11
1	2	1	1	4	2	8351	1	3	5.1	8.2	3.4	1	11
2	2	2	21	4	2	8818	2	3	5.4	6.2	3.3	1	11
2	1	2	2	4	1	9173	2	4	2.2	7.2	2.9	1.1	11
2	2	2	2	3	1	9508	2	3	5.7	6.7	2.8	0.8	11
1	2	2	1	3	2	11302	2	2	1.3	5.2	2	0.7	11
2	2	2	2	4	2	11651	2	3	5.5	6.6	3	1	11
1	2	2	1	4	1	12289	2	4	3.6	5	2.6	1.1	11
2	2	1	2	4	1	15022	1	3	3.8	5.6	2.6	1	11
2	2	1	2	4	1	15022	1	3	3.8	6.9	3.1	1	11
1	2	2	1	4	1	15645	2	4	3.8	6.4	3.5	1.1	11
2	2	2	2	3	1	17470	2	3	2.9	4.7	2	0.8	11
2	1	2	1	3	2	387	2	3	1.3	2.3	0.5	0.8	12
2	2	2	1	3	1	1155	2	3	1.4	2.7	0.6	0.8	12
2	2	2	2	2	2	1618	2	3	0.4	2.3	0.7	0.6	12
2	2	2	2	4	1	1754	2	3	1.3	3.5	0.6	0.9	12
1	2	2	14	3	2	4056	2	2	1.3	2.4	0.7	0.6	12
2	2	2	2	4	2	4554	2	3	1.1	4	0.7	0.9	12
1	2	2	2	4	1	4569	2	3	1.2	2.6	0.7	0.9	12
2	2	1	1	4	1	6290	1	3	1.7	3.9	0.7	0.9	12
1	2	2	1	3	2	6874	2	3	0.4	3.8	0.7	0.8	12
2	2	2	1	3	1	7557	2	3	1.1	3.7	0.9	0.8	12
1	2	2	4	4	1	8161	2	4	1.6	3.1	0.9	1	12
2	2	2	1	3	2	8350	2	3	1.4	3.4	0.7	0.8	12
2	2	2	2	3	1	8888	2	3	0.2	2.4	0.4	0.8	12
1	2	2	2	3	2	9141	2	3	1.1	3	0.7	0.8	12
2	1	2	1	4	1	10932	2	3	1.1	3.3	0.8	0.9	12
2	2	1	1	4	1	11127	1	4	0.8	3.8	0.8	1	12
2	2	2	2	3	1	11467	2	3	1.7	2.8	0.6	0.8	12
2	2	2	19	3	1	12289	2	3	0.6	3	0.6	0.8	12
2	2	2	2	3	1	15051	2	4	1.6	3.4	0.9	0.9	12

APPENDIX I: SE FIELD DATASET

'GENDER'	'AGE'	'LOLE'	'BDGREE'	'ROLE'	'GSOLE'	'GSBDEGREE'	'UNIVERSITY'	'RBACHELORS'	'R'	'D'	'A'	'C'	'CLASS'
2	1	1	1	2	1	1	2	2	3.9	1.9	1.8	0.9	3
2	1	1	1	2	2	2	2	2	4.7	2.2	1.4	0.9	3
1	3	2	2	3	3	3	2	2	5.2	2	1.6	1.1	3
2	2	1	1	2	1	9	1	2	5.7	2.9	1.7	0.9	3
1	4	1	3	2	1	7	3	3	1.5	9.7	3.4	3.8	3
2	1	1	1	2	2	4	2	2	1	1	2.1	1.2	4
1	1	1	1	4	3	2	1	4	1	1.7	2.1	2.4	4
2	2	1	1	2	2	3	1	3	0.9	1	1.8	1.5	4
2	1	1	1	3	2	3	1	3	1	1.2	1.9	1.8	4
2	1	1	1	2	1	1	2	3	1.4	1.4	2.1	1.5	4
2	3	1	1	2	2	3	1	3	1.7	1.5	2.2	1.5	4
1	1	1	1	4	3	3	2	2	1.3	1	2	1.8	4
2	1	1	1	2	1	1	1	4	1.7	2	1.8	1.8	4
2	1	1	1	3	1	9	2	3	1.5	1.6	2.1	1.8	4
2	2	1	2	2	3	10	1	2	1.4	1.6	1.9	1.2	4
2	2	1	1	2	1	9	1	2	0.6	2	2	1.2	4
2	2	2	1	3	1	10	2	3	1.3	1.8	1.8	1.8	4
1	4	1	1	3	3	7	3	3	1.3	1.5	1.9	1.8	4
1	2	1	1	2	3	7	1	3	1.5	8.7	3.4	3.8	4
1	3	1	2	3	3	8	3	3	0.9	8	3.5	4.5	4
1	3	1	1	3	2	2	1	2	1.6	1.7	4.8	1.3	5
2	3	1	2	2	3	3	2	2	1.5	1.3	5.3	1	5
2	2	1	1	2	3	7	2	3	1.9	2.3	4.4	1.3	5
2	3	1	2	2	1	8	3	2	2.8	2.1	4.8	1	5
2	2	1	3	2	3	7	2	2	2.1	1.7	4.8	1	5
1	4	1	2	2	2	4	2	3	1.8	1.4	5.3	1.3	5
2	1	1	1	2	1	1	1	4	3	2.4	4.5	1.5	5
2	2	1	2	2	2	4	2	3	1.8	1.5	4.8	1.3	5
2	3	1	3	3	2	8	3	3	1.6	1.1	5.4	1.5	5
1	2	1	3	3	2	3	3	3	1.8	1.1	5.1	1.5	5
2	2	1	2	3	2	2	2	2	1.8	0.9	5.3	1.3	5
2	2	1	2	3	2	10	2	3	1.5	10	3.2	4.5	5
2	3	2	1	4	1	10	1	3	1.5	8.7	3.7	5.3	5
2	2	1	1	4	1	11	3	3	0.8	12	3.4	5.3	5
1	3	1	2	2	3	5	2	2	2.2	1.4	2.3	1.5	6
1	3	1	2	2	3	8	2	2	1.5	1.2	2.3	1.5	6
2	2	1	1	3	2	3	1	3	1.3	1	2.5	2.3	6
2	2	1	3	2	2	5	1	3	2.1	1.5	2.7	1.9	6
1	3	1	1	2	3	6	2	3	1.8	1.1	2.5	1.9	6
2	2	1	1	3	3	8	3	2	2.1	0.7	2.4	1.9	6
1	4	1	2	2	1	8	2	2	2.1	1.4	2.4	1.5	6
2	3	1	2	3	1	5	2	3	1.5	1	2.7	2.3	6
1	1	1	2	3	2	10	1	3	9.7	2.2	11.5	3	7
2	1	1	1	2	2	9	2	2	7.3	2.6	9.6	2	7
2	1	1	1	4	2	3	1	2	8.7	3.2	10.5	3	7
2	1	1	1	4	3	3	1	2	10	2.9	10.7	3	7
2	3	1	1	2	1	9	1	3	7.7	2.9	11.1	2.5	7
2	3	1	1	3	2	10	1	3	9.3	3.3	9.8	3	7
2	2	1	1	4	2	6	1	3	7	3.1	10.8	3.5	7
2	4	1	3	3	2	10	2	4	8.7	2.9	10.1	3.5	7
2	2	1	1	2	2	6	1	1	5.7	3.1	10	1.5	7
1	5	1	2	4	2	3	2	4	7.7	3.8	9.9	4	7
2	2	1	1	3	1	3	1	2	8.7	2.9	9.2	2.5	7
2	2	1	1	3	1	3	1	4	12	3.7	10	3.5	7
2	4	1	1	3	2	3	1	3	2.2	3.9	1.5	9	8
1	2	1	1	3	1	5	1	3	3.8	5.9	1.3	9	8
2	1	1	1	3	1	1	1	3	3.7	5.2	1.4	9	8
2	1	1	1	3	4	1	1	3	1.7	4.2	1.3	9	8
2	1	1	1	2	2	3	1	3	1.7	12	2.9	3.8	8
2	1	1	1	3	2	3	2	3	1.7	10.7	3.1	4.5	8
2	2	1	2	3	2	2	3	3	2.7	1.9	1.8	1.3	9
2	2	1	2	2	3	7	2	2	2.4	2.7	1.5	0.9	9
2	2	1	3	3	1	1	3	3	3.2	1.6	1.7	1.3	9
2	3	1	2	2	3	5	2	2	3	1.6	1.7	0.9	9
1	3	1	1	2	2	3	2	2	3.5	1.9	1.5	0.9	9
1	1	1	1	2	2	9	1	3	4.9	2.1	1.5	1.1	9
2	1	1	1	2	1	1	1	2	4.5	2.5	1.8	0.9	9
1	3	1	1	2	2	6	1	2	5.5	2.3	1.7	0.9	9
2	1	1	3	2	2	7	1	2	4	2.1	1.5	0.9	9
2	3	1	1	2	4	9	1	3	6	3	1.6	1.1	9
2	1	1	1	3	1	1	1	3	3	1.7	1.2	1.3	9
1	1	2	1	4	1	10	1	2	6	2.9	1.6	1.3	9
2	1	1	1	2	1	1	1	3	5.5	3	1.6	1.1	9
2	2	1	1	2	2	7	1	3	3.9	2.3	1.6	1.1	9
2	1	2	1	2	2	9	2	1	1.4	1.8	1.8	0.9	10
2	2	1	1	2	2	3	3	3	1.1	1.3	2	1.5	10
2	2	1	3	4	2	3	3	3	1.1	1.5	2.1	2.1	10
2	1	1	3	3	2	9	3	4	0.9	1.3	2.2	2.1	10
2	2	1	2	2	3	3	2	3	1	1.2	1.8	1.5	10
1	3	1	2	2	2	3	2	2	1.4	1.7	1.7	1.2	10
2	2	1	2	2	2	5	2	3	1	1.2	2.1	1.5	10
2	3	1	1	4	1	2	3	3	0.8	1	2	2.1	10
2	2	1	1	2	3	8	1	2	1.4	2	2.2	1.2	10
2	3	1	3	4	1	2	3	2	1.4	1.6	2.1	1.8	10
1	2	1	1	2	2	3	1	2	1	0.9	1.9	1.2	10
2	2	1	1	4	1	4	3	3	1.1	1.1	1.9	2.1	10
2	3	1	2	3	2	2	2	3	1	1.1	2.1	1.8	10
2	1	1	1	2	4	1	2	3	0.9	1.1	1.6	1.5	10
1	2	2	2	3	2	6	2	2	0.9	0.8	2.1	1.5	10
1	3	2	2	2	2	8	2	3	1	1.2	2.1	1.5	10
2	2	1	2	3	2	5	1	2	1.7	1.4	5.2	1.3	11
2	1	1	1	3	1	1	1	4	2.2	1.3	5.6	1.8	11
1	1	1	1	3	1	3	1	3	1.6	1.5	5.4	1.5	11
2	3	1	2	1	3	3	2	2	2.6	1.7	4.5	0.8	11
2	3	1	1	2	2	7	2	3	2.6	2.3	5	1.3	11
1	2	1	2	3	2	9	2	2	2.5	1.9	4.2	1.3	11
2	1	1	2	3	1	1	1	3	2	1.9	5.4	1.5	11
1	2	1	2	2	3	7	1	2	2.4	2.3	4.9	1	11
1	2	1	2	2	2	6	2	2	2.3	2.3	5.4	1	11
1	2	1	1	2	2	3	2	3	1.6	1.1	5	1.3	11
1	2	1	2	2	1	1	2	2	2.4	2.3	4.1	1	11
2	1	1	1	2	2	6	1	2	2.5	1.6	5.7	1	11
2	3	1	1	2	2	7	1	3	1.8	1.3	5.1	1.3	11
2	2	1	1	4	2	3	1	4	3	2.4	5.9	2	11
1	1	1	1	3	2	8	2	2	2.3	1.3	5.5	1.3	11
2	2	1	2	3	2	8	2	3	1.3	1.5	2.5	2.3	12
2	1	1	3	4	2	6	1	4	1.7	1.1	2.9	3	12
2	2	1	1	3	2	2	2	3	1.1	0.7	2.6	2.3	12

1	1	1	1	4	2	9	1	4	1.9	1.3	2.1	3	12
1	3	1	2	3	1	5	2	3	1.1	1	2.7	2.3	12
1	1	1	1	2	4	9	2	2	1.7	1.1	2.6	1.5	12

APPENDIX J: ACADEMIC LIBRARIANS FIELD DATASET

'GENDER'	'AGE'	'LOLE'	'BDGREE'	'ROLE'	'GSOLE'	'GSBDEGREE'	'UNIVERSITY'	'RBACHELORS'	'R'	'D'	'A'	'C'	'CLASS'
1	4	1	1	3	3	2	1	3	3	4	1	2	3
1	4	1	3	2	2	4	1	2	1	1	1	1	7
1	3	1	1	2	3	2	1	2	8	1	9	1	1
2	2	1	1	2	2	4	1	3	1	3	2	1	2
1	2	1	1	3	3	2	3	3	3	4	1	2	3
1	5	1	1	2	3	2	1	2	2	2	1	1	7
1	3	1	1	3	3	2	1	3	3	3	1	2	3
1	5	1	1	3	3	2	3	3	8	1	9	1	1
1	2	1	1	2	3	2	2	3	9	0	11	1	1
1	3	1	2	2	2	2	2	2	2	2	1	1	7
2	3	1	1	2	2	4	1	2	9	0	11	1	1
1	3	1	1	3	3	10	2	2	3	1	1	2	3
1	3	1	1	2	3	4	1	2	1	1	1	1	7
1	3	1	1	2	3	2	3	3	10	1	9	1	1
2	5	1	1	1	3	2	1	3	2	2	1	1	7
1	2	1	1	3	3	2	1	3	2	2	5	5	5
1	2	1	1	3	3	2	3	3	2	1	5	5	5
1	2	1	1	3	3	2	1	3	3	3	1	2	3
1	5	1	1	3	2	10	1	3	1	2	2	2	2
1	5	1	1	3	4	1	1	3	3	3	1	2	3
1	5	1	1	4	1	10	1	4	1	1	1	2	7
1	2	1	1	2	2	10	1	3	1	2	3	1	2
1	5	1	1	2	3	10	3	3	1	2	2	1	2
2	2	1	1	3	3	2	1	3	10	1	11	1	1
2	4	1	1	3	2	4	1	3	3	1	4	5	5
2	4	1	1	2	3	4	1	3	1	2	2	1	2
2	4	1	1	2	2	2	3	2	4	1	3	2	4
2	4	1	1	3	3	4	1	3	4	1	4	3	4
1	2	1	2	3	3	2	1	3	4	1	3	3	4
2	5	1	1	2	4	10	1	3	2	2	2	1	2
1	5	1	2	4	4	1	3	3	1	3	2	2	2
1	1	1	2	2	1	2	2	2	8	1	9	1	1
2	5	1	1	3	2	4	3	3	2	2	5	5	5
2	5	1	2	3	4	1	3	3	2	2	5	5	5
1	5	1	2	4	4	1	3	3	3	0	5	5	5
1	2	1	1	3	3	5	1	3	12	1	11	1	1
1	4	1	2	4	2	4	3	3	1	1	4	5	5
2	5	1	2	4	4	1	1	2	1	2	3	2	2
2	5	1	2	4	4	8	2	2	1	5	2	2	6
1	4	1	2	2	3	2	1	3	3	2	2	2	3
1	5	2	2	4	2	10	3	3	9	1	10	2	1
2	5	1	2	3	3	2	2	3	1	5	2	2	6
1	3	1	1	3	2	7	2	3	5	1	4	3	4
1	2	1	1	3	3	2	2	3	5	1	3	3	4
2	5	1	2	4	3	2	2	3	2	1	3	5	5
2	3	1	1	4	2	4	3	3	1	10	2	11	6
1	5	2	2	2	3	10	2	3	1	5	1	2	6
2	5	1	2	4	3	4	3	3	4	1	3	4	4
1	5	2	2	2	1	10	2	3	5	1	3	3	4
2	5	2	1	3	3	10	3	3	3	2	6	5	5

APPENDIX K: PYTHON SAMPLE CODE FOR THE PROTOTYPE

```
import tkinter.filedialog
from tkinter import *
#from ScrolledText import *
import tkinter.ttk as ttk
import pandas as pd
import dill as pickle
#import pickle
import random
import sqlite3
import svmpy
import logging
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import itertools
import argh
import csv
import decimal
import math
import copy
import time
import threading
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from sklearn.preprocessing import StandardScaler
splitRatio = 0.80
splitRatio2 = 0.10
#THIS CLASS IS FOR SVM CLASSIFIERS ONLY
class SVMRootclassifier():
    def __init__(self):
        #self.master = master
        self.value = None

    def loadCsv(self,filename):
        self.filename=filename
        lines = csv.reader(open(filename, "r"))
        dataset = list(lines)
        for i in range(len(dataset)):
            dataset[i] = [float(x) for x in dataset[i]]
        return dataset

    def splitDataset2(self, dataset, splitRatio):# splits the dataset into two: training and test dataset
        self.dataset=dataset
        self.splitRatio=splitRatio
        freq = self.getClassDistribution(dataset)
        trainSet = []
        copy = []
        #print('key','frequency','trainsize')
        for keys,frequency in freq.items():
            trainSize = int(frequency * splitRatio)
            #print(keys,  frequency,  trainSize)
            fold = []
            trainFold = []
            for k in range(len(dataset)):
                vector1=dataset[k]
                if (vector1[-1]==keys):
                    fold.append(vector1)
            #print('frequency:len(fold):trainSize', frequency,len(fold),trainSize)
            while len(trainFold) < trainSize:
                index = random.randrange(len(fold))
                trainFold.append(fold.pop(index))
                #print('fold',  fold)
            #trainSet.append(trainFold)
            trainSet= trainSet + trainFold
            #copy.append(fold)
            copy = copy + fold
            #print('copy',  copy)
        return [trainSet, copy]

    def splitDataset(self, dataset, splitRatio):# splits the dataset into two: training and test dataset
        self.dataset=dataset
        self.splitRatio=splitRatio
        trainSize = int(len(dataset) * splitRatio)
        trainSet = []
        copy = list(dataset)
        while len(trainSet) < trainSize:
            index = random.randrange(len(copy))
            trainSet.append(copy.pop(index))
        return [trainSet, copy]

    #TAKES IN REQUIRED CLASSES AND FILTERS THE DATASET TO REMAIN WITH INSTANCES OF ONLY THESE       CLASSES
    def refineDataset(self, dataset, mergeclasses):# removes unwanted classes from dataset
```

```
        self.dataset=dataset
        self.classValue = mergeclasses
        dataset2 = []
        for i in range(len(dataset)):
                vector1=dataset[i]
                if vector1[-1] in mergeclasses:
                        dataset2.append(vector1)
        return dataset2

#CLASSES AND FILTERS THE DATASET TO REMAIN WITH INSTANCES OF ONLY THESE CLASSES
def getClassDistribution(self,dataset):
        self.dataset = dataset
        #distinctcopy = list(dataset)
        dataset2 = {}
        counts = {}
        #print('trainSet',  dataset)
        for i in range(len(dataset)):
               vector1 = dataset[i]
               #print('vector1',vector1)
               #print('vector1[-1]',vector1[-1])
               if (vector1[-1] in counts):
                        counts[vector1[-1]] += 1
               else:
                        counts[vector1[-1]]=1
        return counts

def getClassAccuracy(self,testFile,correctClassified,incorrectClassified):
        self.testFile = testFile
        self.correctClassified = correctClassified
        self.incorrectClassified = incorrectClassified
        classAccuracy = {}
        for classValue,freq in testFile.items():
              if classValue in correctClassified:
                 n = freq
                 x = correctClassified[classValue]
                 y = x * 100/n
              else:
                 y = 0.0
              classAccuracy[classValue] = y
        return classAccuracy

def getClassCodes(self,dataset,parentclass):
        self.dataset=dataset
        dataset2 = []
        self.parentclass = parentclass
        #print('parentclass',parentclass)
        for i in range(len(dataset)):
               vector1 = dataset[i]
               vector = dataset[i]
               value = float(parentclass)
              # print('parentclass',type(value),type(vector1[-1]))
               if (vector1[-1] == value):
                        #print(i,'class for 1',vector1[-1],value)
                        vector1[-1] = 1
                        vector1[-1]=float(vector1[-1])
                        dataset2.append(vector1)
               else:
                        #print(i,'class for -1',vector1[-1],value)
                        vector1[-1] = -1
                        vector1[-1]=float(vector1[-1])
                        dataset2.append(vector1)
        return dataset2

def separateByRootClass(self,dataset,mergeclass,parentclass):# merges and then separates instances
        #into distinct classes
        self.dataset=dataset
        self.mergeclasses = mergeclass
        self.parentclass = parentclass
        separated = {}
        dataset2 = []
        dataset3 = []
        #print('parentclass,mergeclass,dataset',parentclass,mergeclass)
        positives = mergeclass[1]
        negatives = mergeclass[-1]
        #print('positives=:',positives)
        #print('negatives=:',negatives)
        for k in range(len(dataset)):
                vector1=list(dataset[k])
                if vector1[-1] in negatives:
                        vector1[-1] = -1
                        vector1[-1]=float(vector1[-1])
                        dataset2.append(vector1)
                if vector1[-1] in positives:
                        vector1[-1] = 1
                        vector1[-1]=float(vector1[-1])
                        dataset2.append(vector1)
        return dataset2
```

```
def svmTrainer(self,data,num_samples, num_features,para):
        #self.k = k
        self.data = data
        self.num_samples=num_samples
        self.num_features=num_features
        samples = []
        value = []
        labels = []
        k = num_features
        #print('num_features',num_features)
        #print('data',data)
        #THIS SEPARATES SAMPLE CASES AND LABELS
        for i in range(len(data)):
                vector = data[i]
                #print('data,class',data[i],vector[-1])
                value = vector[-1]
                #samples.append(vector[0:77])#THIS STANDS FOR RANGE OF ATTRIBUTES
                #samples.append(vector[0:19])#THIS STANDS FOR RANGE OF ATTRIBUTES
                samples.append(vector[0:k])#THIS STANDS FOR RANGE OF ATTRIBUTES
                labels.append(value)
        #print('size',num_samples,num_features)
        sample = np.matrix(samples).reshape(num_samples,num_features)
        label = np.matrix(labels).reshape(-1,1)
        #print('label',label)
        #trainer = svmpy.SVMTrainer(svmpy.Kernel.linear(), para) #this is training with linear kernel
        #trainer = svmpy.SVMTrainer(svmpy.Kernel.gaussian(0.1), para) #this is training with linear kernel
        trainer = svmpy.SVMTrainer(svmpy.Kernel.gaussian(1.0), para) #this is training with linear kernel
        #trainer = svmpy.SVMTrainer(svmpy.Kernel._polykernel(0.1,1.0), para) #this is training with linear kernel
        #print('trainer',trainer)
        #print('label1')
        predictor = trainer.train(sample, label)
        #print('label2')
        #print('predictor')
        return predictor

def Predict(self,predictor, X):
        self.predictor = predictor
        self.X = X
        #print('X',X,predictor)
        #print('predictor',predictor)
        result = []
        #vector = X[0:len(X)-1]
        vector = X
        result.append(predictor.predict(vector))
        #result.append(svmpy.SVMPredictor.predict(vector))
        return result

def getPredictions(self, predictor, testdata):
        self.predictor = predictor# summary of mean and std dev of each atribute in each class
        self.testdata = testdata
        predictions = []
        test = []
        for i in range(len(testdata)):
                #test = testSet[i]
                test.append(testdata[i])
        #print('The testset is:',testdata)
        for i in range(len(test)):
                vector = test[i]
                result = self.Predict(predictor, vector)
                #print('Testset prediction is:',test[i],result)
                predictions.append(result)
        return predictions

def getAccuracy(self, testdata, predictions):
        self.testdata=testdata
        self.predictions=predictions
        correct = 0
        #print('Testset prediction is:',testdata,predictions)
        for i in range(len(testdata)):
                #print('Testset prediction is:',testdata[i],predictions[i])
                vector = testdata[i]
                value1= predictions[i]
                value2 = vector[-1]
                #print('Testset prediction is:',value1[0],value2)
                if (value1[0] == value2):
                        #print('Testset prediction is:',vector[-1],predictions[i])
                        correct += 1
        return (correct/float(len(testdata))) * 100.0

def getLevelNodes(self, classTree,level):
        self.classTree = classTree
        self.level = level
        parent = []
        childs = []
        levelNodes = {}
        for classValue,instances in classTree.items():
```

```
            for i in range(len(instances)):
                classlev = instances[0]
                classlevel = classlev[0]
                parent = instances[1]
                childs = instances[2]
                if (classlevel == level): #only for this level
                    levelNodes[classValue] = [parent,childs]
        return levelNodes

    def getTreeDepth(self, classTree):
        self.classTree = classTree
        depth = 0
        for classValue,instances in classTree.items():
            for i in range(len(instances)):
                classlevel = instances[0]
                level = classlevel
                if(len(level) > 0):
                    #print(type(level[0]),type(depth))
                    #print(level[0],depth)
                    if (int(level[0]) > depth): #check leve of the current node
                        depth = level[0]

        return depth

    def getChildrenOf(self, classvalue,classTree):
        self.classvalue = classvalue
        self.classTree = classTree
        childs = []
        #print('class value is',classvalue)
        for classValue,instances in classTree.items():
            #print('classValue,classvalue',classValue,classvalue)
            if (classValue == classvalue):
                #print('instances',instances)
                #for i in range(len(instances)):
                    #parent = instances[1]
                    childs = instances[2]
                    #print('instances',instances[2])
        #print('childs',childs)
        return childs

    def getSubTrees(self, classTree):
        self.classTree = classTree
        childs = []
        subTrees = {}
        height = self.getTreeDepth(classTree)
        top = 0
        if (height > 0):
            topNodes= self.getLevelNodes(classTree,top)
            #print('THESE ARE TOPNODES',topNodes)
            for classValue,instances in topNodes.items():
                if (classValue < 0):
                    nextparent = classValue
                    #print('nextparent,classvalue',nextparent,classValue)
                    #childs = self.getChildrenOf(nextparent,classTree)
                    while nextparent<0:
                        classes = []
                        childs = self.getChildrenOf(nextparent,classTree)
                        #print('childs of: ',nextparent,'are:',childs)
                        nextparent = 0
                        for i in range(len(childs)):
                            if (childs[i]>0):
                                classes.append(childs[i])
                            else:
                                nextparent=childs[i]
                        if classValue not in subTrees:
                            subTrees[classValue] = []
                        subTrees[classValue].append(classes)
        return subTrees

    def getMainTrees(self, classTree):
        self.classTree = classTree
        childs = []
        mainTree = {}
        Tree = {}
        for classValue,instances in classTree.items():
            #if (classValue < 0):
            maintreeid = instances[3]
            #print('maintreeid:',maintreeid)
            if maintreeid[0]>0:
                if maintreeid[0] not in mainTree:
                    mainTree[maintreeid[0]] = []
                    Tree = {}
                    classes = []
                    classes = [instances[0],instances[1],instances[2]]
                    Tree[classValue] = []
                    Tree[classValue] = classes
                    mainTree[maintreeid[0]] = Tree
```

```
                else:
                    classes = []
                    classes = [instances[0],instances[1],instances[2]]
                    Tree = mainTree[maintreeid[0]]
                    if classValue not in Tree:
                        Tree[classValue] = []
                    Tree[classValue] = classes
                    mainTree[maintreeid[0]] = Tree
                #print('maintree:',mainTree)
        return mainTree

    def orderByParents(self, classNodes):
        self.classNodes = classNodes
        print(classNodes)
        orderedByParent = {}
        parentList = []
        for classValue,instances in classNodes.items():
            #for i in range(len(instances)):
            parent = instances[0]
            #print('parent',parent[0])
            if(len(parent)!=0):
                if parent[0] not in orderedByParent:
                    #parentList = classValue
                    orderedByParent[parent[0]]=[]
                parentList=orderedByParent[parent[0]]
                #print('parentList',parentList)
                if classValue not in parentList:
                    if (len(parentList)==0):
                        parentList = [classValue]
                    else:
                        parentList.append(classValue)
                    #print('classValue',classValue)
                    #print('parentList',parentList)
                    orderedByParent[parent[0]] = parentList
                    #print('orderedByParent',orderedByParent)
        return orderedByParent

    def getParentNode(self, childnode,classTree):
        self.childnode = childnode
        self.classTree = classTree
        parent = []
        for classValue,instances in classTree.items():
            if (childnode in instances):
                parent = [classvalue]
                continue
        return parent

    #THIS CREATES A HIERARCHICAL MULTI-CLASSIFIER
    def classify(self,mainTree,trainingSet,para):
        svm = SVMRootclassifier()
        TreePredictors = {}
        self.para = para
        self.trainingSet = trainingSet
        self.mainTree = mainTree
        trainset = []
        trainset = list(trainingSet)
        #Trainfile = list(trainingSet
        maintrees = svm.getMainTrees(mainTree)
        mKey = maintrees.keys()
        otherTrees = []
        AllTrees = []
        value = []
        mainTreePredictor = {}
        for mKeys, classTree in maintrees.items():
            for mK, trees in classTree.items():
                value = mK
                #print('mK:',mK)
                if mK > 0:
                    value =[mK]
                    AllTrees = AllTrees + value
                    otherTrees = otherTrees + value
        #print('ALLTREES and mKey :',AllTrees,mKey)
        treeno = 0
        #FOR EACH MAIN TREE
        for mKeys, classTree in maintrees.items():
            #print('CLASSTREES:',classTree)
            #COUNT MAIN TREES
            treeno = treeno + 1
            depth = svm.getTreeDepth(classTree)#GET DEPTH/HEIGHT OF EACH MAIN TREE
            subtrees = svm.getSubTrees(classTree)#GET SUBTREES/BRANCHES OF EACH MAIN TREE
            Key = list(subtrees.keys())#GET THE SUBTREE ID'S
            #print('SUBTREES and Key:',subtrees,Key)
            trainset = list(trainingSet)
            Allclasses = []
            otherClasses = []
            TreePredictorTree = {}
            CellPredictorTree = {}
```

```
NodePredictorTree = {}
#'''
#FOR EACH SUBTREE/BRANCH OF THE MAIN TREE GET ALL THE CLASSES
for Keys, cells in subtrees.items():
    for i in range(len(cells)):
        Allclasses = Allclasses + cells[i]
        otherClasses = otherClasses + cells[i]
#print('ALL CLASSES SET :',Allclasses,type(otherClasses))
order = 0
#FOR EACH SUBTREE/BRANCH OF THE MAIN TREE GET NODE AND CELL CLASSIFIERS
for Keys, cells in subtrees.items():
    if len(Key)>1:
      order = order + 1#COUNT SUBTREES/BRANCHES
    cell1 = []
    cellorder = 0
    CellPredictorTree = {}
    NodePredictorTree = {}
    #''
    #IN EACH SUBTREE/BRANCH CELL
    #trainingSet3=trainset
    #print('len(trainingSet3):len(trainingSet3[0])-1',len(trainingSet3),len(trainingSet3[0])-1)
    for i in range(len(cells)):
        cellorder = cellorder + 1#COUNT CELLS IN EACH SUBTREE
        nodes = cells[i]
        #CREATE NODE PREDICTOR
        #print('NODES:',nodes)
        if len(nodes) == 2:#IF ONLY TWO LEAF NODES IN EACH CELL CREATE NODE CLASSIFIER FOR EACH
            #trainset = svm.loadCsv(filename)
            #trainset = list(Trainfile)
            trainingSet3=list(trainset)
            #print('len(trainingSet3)',len(trainingSet3))
            currentnode = []
            othernode = []
            currentnode = [nodes[0]]
            othernode = [nodes[1]]
            mergeclass = {1:currentnode,-1:othernode}
            #print('1:currentnode,-1:othernode',currentnode,othernode)
            trainingSet3, testSet3 = svm.splitDataset2(trainingSet3, splitRatio)
            trainingSet3 = svm.separateByRootClass(trainingSet3,mergeclass,nodes[0])
            if len(trainingSet3) > 0:
                testSet3 = svm.separateByRootClass(testSet3,mergeclass,nodes[0])
                #print('len(trainingSet3)',len(trainingSet3))
                cases = len(trainingSet3)
                                                  #print('len(trainingSet3):len(trainingSet3[0])',len(trainingSet3),len(trainingSet3[0]))
                features = len(trainingSet3[0])-1
                predictor = svm.svmTrainer(trainingSet3,cases,features,para)
                dataframe = pd.DataFrame(testSet3)
                array1 = dataframe.values
                X = []
                X = array1[:,0:features]
                predictions3 = svm.getPredictions(predictor,X)
                accuracy3 = svm.getAccuracy(testSet3, predictions3)
                NodePredictor = {currentnode[0]:[currentnode+othernode,predictor,accuracy3]}
            #print('PREDICTION ACURACY FOR NODES:',nodes,accuracy3)
        else:#IF ONLY ONE LEAF NODE IN EACH CELL CREATE NODE CLASSIFIER FOR ONLY ONE NODE
            predictor=[]
            accuracy3='100%'
            NodePredictor = {nodes[0]:[nodes,predictor,accuracy3]}
            #print('PREDICTION ACURACY FOR NODES:',nodes[0],': IS:',accuracy3)
        #print('THE NODE PREDICTOR :',NodePredictor)
        if (order not in NodePredictorTree):
                NodePredictorTree[order] = []
        NodePredictorTree[order].append(NodePredictor)#STORE NODE CLASSIFIERS   ACCORDING TO THEIR CELL NUMBER
        #print('NODE PREDICTOR TREE:',NodePredictorTree)
        #CREATE HIERARCHICAL CELL CLASSIFIERS
        CellPredictor = []
        if cellorder<=len(cells)-1:#CREATE ONE AGAINST ALL(REMIANING CELLS) CELL CLASSIFIERS
            #trainset = svm.loadCsv(filename)
            #trainset = list(Trainfile)
            #trainset = copy.copy(Trainfile)
            trainingSet2=list(trainset)
            othercells = cell1 + cells[i]
            currentcell = cells[i+1]
            cell1 = cells[i]
            mergeclass = {1:currentcell,-1:othercells}
            trainingSet2, testSet2 = svm.splitDataset2(trainingSet2, splitRatio)
#print('getClassDistribution:trainset2',svm.getClassDistribution(trainingSet2),mergeclass)
            #print('getClassDistribution:cellsTrainfile',svm.getClassDistribution(Trainfile))
            trainingSet2 = svm.separateByRootClass(trainingSet2,mergeclass,i)
            #print('getClassDistribution:cellsTrainfile',svm.getClassDistribution(Trainfile))
            testSet2 = svm.separateByRootClass(testSet2,mergeclass,i)
            cases = len(trainingSet2)
            features = len(trainingSet2[0])-1
            predictor = svm.svmTrainer(trainingSet2,cases,features,para)
            dataframe = pd.DataFrame(testSet2)
            array1 = dataframe.values
            X = []
```

```
                X = array1[:,0:features]
                predictions2 = svm.getPredictions(predictor,X)
                accuracy2 = svm.getAccuracy(testSet2, predictions2)
                CellPredictor ={len(cells)-cellorder:[currentcell,othercells,predictor,accuracy2]}
                #print('ACCURACY FOR CELL PREDICTION',currentcell,' IS: %=', accuracy2)
            #print('THE CELL PREDICTOR IS:',CellPredictor)
            if (order not in CellPredictorTree):
                CellPredictorTree[order] = []
            if len(CellPredictor) > 0:
                CellPredictorTree[order].append(CellPredictor)
            #print('CELL PREDICTOR TREE:',CellPredictorTree)
            #''
        #CREATE SUBTREE CLASSIFIERS
        if (order<=len(Key)-1):
            #trainset = svm.loadCsv(filename)
            #trainset = Trainfile
            trainingSet1= list(trainset)
            currentTree = []
            for i in range(len(cells)):
                currentTree = currentTree + cells[i]
            for i in range(len(currentTree)):
                otherClasses.remove(currentTree[i])
            others = []
            for j in range(len(otherClasses)):
                others.append(otherClasses[j])
            ThisLevelmergeclass = {1:currentTree,-1:others}
            trainingSet1, testSet1 = svm.splitDataset2(trainingSet1, splitRatio)
            trainingSet1 = svm.separateByRootClass(trainingSet1,ThisLevelmergeclass,Keys)
            testSet1 = svm.separateByRootClass(testSet1,ThisLevelmergeclass,Keys)
            cases = len(trainingSet1)
            features = len(trainingSet1[0])-1
            predictor = svm.svmTrainer(trainingSet1,cases,features,para)
            dataframe = pd.DataFrame(testSet1)
            array1 = dataframe.values
            X = []
            X = array1[:,0:features]
            predictions1 = svm.getPredictions(predictor,X)
            accuracy1 = svm.getAccuracy(testSet1, predictions1)
            TreePredictor = {1.0:currentTree,-1.0:others,0.0:[predictor,accuracy1],}
            #print('ACCURACY FOR SUBTREE PREDICTION',currentTree, 'IS: %=', accuracy1)
            #print('THE SUBTREE PREDICTION:',TreePredictor)
        if (order not in TreePredictorTree):
                    TreePredictorTree[order] = []
        TreePredictorTree[order].append([TreePredictor,CellPredictorTree,NodePredictorTree])
#'''
#print('TreePredictorTree:',TreePredictorTree)

#CREATE TREE CLASSIFIERS
if (treeno<=len(mKey)-1):
        #trainset = svm.loadCsv(filename)
        #trainset = Trainfile
        trainingSet1=list(trainset)
        currentMainTree = []
        for mK, trees in classTree.items():
            if mK > 0:
                value = [mK]
                currentMainTree = currentMainTree + value
        for i in range(len(currentMainTree)):
            otherTrees.remove(currentMainTree[i])
        others = []
        for j in range(len(otherTrees)):
            others.append(otherTrees[j])
        ThisLevelmergeclass = {1:currentMainTree,-1:others}
        trainingSet1, testSet1 = svm.splitDataset2(trainingSet1, splitRatio)
        trainingSet1 = svm.separateByRootClass(trainingSet1,ThisLevelmergeclass,mKeys)
        testSet1 = svm.separateByRootClass(testSet1,ThisLevelmergeclass,mKeys)
        cases = len(trainingSet1)
        features = len(trainingSet1[0])-1
        predictor = svm.svmTrainer(trainingSet1,cases,features,para)
        dataframe = pd.DataFrame(testSet1)
        array1 = dataframe.values
        X = []
        X = array1[:,0:features]
        predictions2 = svm.getPredictions(predictor,X)
        accuracy2 = svm.getAccuracy(testSet1, predictions2)
        mainTreePredictor = {1.0:currentMainTree,-1.0:others,0.0:[predictor,accuracy2],}
        #print('ACCURACY FOR MAIN TREE PREDICTION IS:', accuracy2)
else:
        #print('TWIN1 IS:treeno,len(mKey)',treeno,len(mKey) )
        if (len(mKey)==1):
            predictor=[]
            accuracy2='100%'
            mainTreePredictor = {1.0:[],-1.0:[],0.0:[accuracy2],}
            #mainTreePredictor = {mKeys:[predictor,accuracy2]}
#print('MAIN TREE PREDICTOR IS:', mainTreePredictor)
#'''
if (mKeys not in TreePredictors):
```

```
                    TreePredictors[mKeys] = []
                TreePredictors[mKeys].append([mainTreePredictor,TreePredictorTree])
        #print('ALL MAIN TREES PREDICTORS ARE:', TreePredictors)
        '''
        for tree, trees in TreePredictors.items():
                TreePredictorList = trees[0]
                mainTreePredictor =  TreePredictorList[0]

                mainPredictors = mainTreePredictor[1]
                print('Key: 1.0', tree,mainPredictors)
                mainPredictors = mainTreePredictor[-1]
                print('Key: -1.0', tree,mainPredictors)
        '''
        #pickle_out = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\multiclassifier.pickle','wb')
        #pickle.dump(TreePredictors,pickle_out)
        #pickle_out.close()
        return TreePredictors
#THIS CLASSIFIES A WHOLE DATASET
def classifyInstance(self,classifier,classTree,data):
        svm = SVMRootclassifier()
        self.classifier = classifier
        self.classTree = classTree
        self.data = data
        tree = classTree
        testdata = data
        #mKey =list(TreePredictors.keys())
        #print('TreePredictors keys:',mKey)
        mKey =list(classifier.keys())
        #print('TreePredictors keys:',mKey)
        #TreePredictorTree = {}
        CellPredictorTree = {}
        NodePredictorTree = {}
        #predictiondata = data
        predictiondata = []
        correctClassified = {}
        incorrectClassified = {}
        predictionresult = -1
        for i in range(len(testdata)):
                        X = testdata[i]
                        vector = X[0:len(X)-1]
                        #print('testdata[i]',testdata[i])
                        T = 0
                        while (T <len(mKey)):#CHECK IN EACH MAIN TREE IN WHICH THE INSTANCE BELONGS
                                treeno = mKey[T]#GET CLASSIFIER NUMBER
                                #mainTreePredictorList = TreePredictors[treeno]
                                mainTreePredictorList = classifier[treeno]
                                mainTreePredictor =  mainTreePredictorList[0]
                                #print('mainTreePredictor[0]', mainTreePredictor[0])
                                mainPredictors = mainTreePredictor[0][0.0]
                                if (len(mainPredictors)> 1):#CASE OF MORE THAN ONE TREE
                                        mainPredictor = mainPredictors[0]
                                        mainTreeResult =  svm.Predict(mainPredictor,vector)#MAKE PREDICTION
                                        if (mainTreeResult[0] ==1.0)and (T <=(len(mKey)- 2)):#IF 1 GET SUBTREE CLASSIFIER
                                            TreePredictorTree = mainTreePredictor[1]
                                            #print('FOR MAINTREE NO:', treeno)
                                            T = len(mKey)+1#END THE LOOP
                                        else:#IF -1
                                            if (mainTreeResult[0] == -1.0)and (T >= (len(mKey) - 2)):#CHECK WHETHER IT IS SECOND LAST
                                                treeno = mKey[T+1]#GET GET THE ONLY LAST AND END THEN LOOP
                                                #mainTreePredictorList = TreePredictors[treeno]
                                                mainTreePredictorList = classifier[treeno]
                                                mainTreePredictor =  mainTreePredictorList[0]
                                                #print('mainTreePredictor[0]', mainTreePredictor[0])
                                                TreePredictorTree = mainTreePredictor[1]
                                                #print('(this is second last)FOR MAINTREE NO:', treeno)
                                                T = len(mKey)+1 #END THE LOOP
                                            else:#IF NOT SECOND LAST (mainTreeResult[0] == -1.0)and (T < len(mKey) - 2)
                                                T = T + 1 #LOOP AGAIN
                                else:#CASE OF ONLY ONE TREE
                                         TreePredictorTree = mainTreePredictor[1]
                                         T = len(mKey)+1 #END THE LOOP
                        Key = list(TreePredictorTree.keys())
                        N = len(Key)
                        #print('TreePredictorTree', TreePredictorTree)
                        K = 0
                        while (K < len(Key)):#CHECK IN EACH SUBTREE THE CELL IN WHICH THE INSTANCE BELONGS
                                subtreeno = Key[K]
                                TreePredictorList = TreePredictorTree[subtreeno]#TreePredictorList IS A LIST OF ONLY ONE ELEMENT I.E. THIS SUBTREE
                                TreePredictorTr  = TreePredictorList[0]#TreePredictorTr IS A LIST OF THREE DICTIONARIES OF THIS SUBTREE PREDICTORS I.E.[{SUBTREE},{CELLS},{NODES}]
                                TreePredictor  = TreePredictorTr[0]# TreePredictor IS A DICTIONARY OF THIS SUBTREE PREDICTOR
                                CellPredictorList = TreePredictorTr[1] #CellPredictorList IS A DICTIONARY OF THIS SUBTREE CELL PREDICTORS
                                NodePredictorList  = TreePredictorTr[2]#NodePredictorList IS A DICTIONARY OF THIS SUBTREE NODE PREDICTORS
                                CellPredictors = CellPredictorList[subtreeno]#CellPredictors IS A LIST OF THIS SUBTREE'S CELL PREDICTORS
                                Trpredictor = TreePredictor[0.0]#Trpredictor IS A PREDICTOR OF THIS CURRENT SUBTREE
                                #print('Trpredictor[0]',Trpredictor[0])
                                result1 = svm.Predict(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
                                #print('subtree result1',result1)
```

```
if (result1[0] == 1.0):#IF CURRENT SUBTREE PREDICTED YES
    #GET CELL PREDICTORS
    X = len(CellPredictors)
    #print('CellPredictor:for result1=1',CellPredictors)
    if (X>0):#IF THERE ARE CELL PREDICTORS
        cellpredictorskeys = []
        cellpredictor = {}
        i=0
        while i<X:#WHILE THERE ARE CELL PREDICTORS
            predictor = CellPredictors[i]
            for Keys, cells in predictor.items():
                predictorkey = Keys
                cellpredictorskeys.append(predictorkey)
                cellpredictor[predictorkey] = []
                cellpredictor[predictorkey].append(predictor[predictorkey])
            i=i+1
        cellpredictorskeys.sort()#SORT THEM IN THE ORDER THEY WILL BE WORKED ON
        count=X
        for Keys, cells in cellpredictor.items():
            count=count-1 #COUNT CELL PREDICTORS BOTTOM UP
            #print('Cell:',cells)
            cell = cells[0]
            currentcell = cell[0]
            othercells = cell[1]
            cellpredictor = cell[2]
            accuracy2 = cell[3]
            result2 =  svm.Predict(cellpredictor,vector)
            #print('Cell result2',result2[0])
            if (result2[0] == 1.0):  #IF CELL RESULT IS 1 SELECT THE FIRST CELL'S NODE PREDICTORS
                #GET NODE PREDICTOR FOR THIS SUBTREE
                NodePredictors = NodePredictorList[subtreeno]
                #print('NodePredictors',NodePredictors)
                X = len(NodePredictors)
                nodepredictorskeys = []
                nodepredictor = {}
                i=0
                while i<X:
                    predictor = NodePredictors[i]
                    for Keys, nodes in predictor.items():
                        predictorkey = Keys
                        nodepredictorskeys.append(predictorkey)
                        nodepredictor[predictorkey] = []
                        nodepredictor[predictorkey].append(predictor[predictorkey])
                    i=i+1
                nodepredictorskeys.sort()
                #print('nodepredictor',nodepredictor)
                nodes = nodepredictor[currentcell[0]]
                #print('nodes',nodes)
                node = nodes[0]
                nodepair = node[0]
                if len(nodepair)==2:
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  svm.Predict(nodepredictor,vector)
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]
                else:
                        predictionresult = nodepair[0]
                #print('Node result(+ve)',vector,predictionresult)
                break
            else:#IF CELL RESULT IS -1 SELECT THE OTHER CELL'S NODES
                if (count==0):#IF THIS IS THE LAST CELL PREDICTOR FOR THIS SUBTREE
                    if len(othercells)==1:#IF THERE IS ONLY ONE NODE IN THIS CELL
                        predictionresult = othercells[0]
                    else:#IF THERE IS MORE THAN ONE(TWO) NODES IN THIS CELL
                        NodePredictors = NodePredictorList[subtreeno]
                        X = len(NodePredictors)
                        nodepredictorskeys = []
                        nodepredictor = {}
                        i=0
                        while i<X:
                            predictor = NodePredictors[i]
                            for Keys, nodes in predictor.items():
                                predictorkey = Keys
                                nodepredictorskeys.append(predictorkey)
                                nodepredictor[predictorkey] = []
                                nodepredictor[predictorkey].append(predictor[predictorkey])
                            i=i+1
                        nodepredictorskeys.sort()
                        nodes = nodepredictor[othercells[0]]
                        #print('nodes',nodes)
                        node = nodes[0]
                        nodepair = node[0]
                        if len(nodepair)==2:
```

```
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  svm.Predict(nodepredictor,vector)
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]
                  else:
                        predictionresult = nodepair[0]
                  #print('Node result(+ve)',vector,predictionresult)
                  break
          K = N
      else:#IF THERE ARE NO CELL PREDICTORS
          #GET NODE PREDICTORS
              NodePredictors = NodePredictorList[subtreeno]
              X = len(NodePredictors)
              nodepredictorskeys = []
              nodepredictor = {}
              i=0
              while i<X:
                  predictor = NodePredictors[i]
                  for Keys, nodes in predictor.items():
                      predictorkey = Keys
                      nodepredictorskeys.append(predictorkey)
                      nodepredictor[predictorkey] = []
                      nodepredictor[predictorkey].append(predictor[predictorkey])
                  i=i+1
              nodepredictorskeys.sort()
              currentcell = nodepredictorskeys[0]
              #print('currentcell',currentcell)
              nodes = nodepredictor[currentcell]
              #print('nodes',nodes)
              node = nodes[0]
              nodepair = node[0]
              if len(nodepair)==2:#CHECK IF THERE ARE TWO NODES IN A CELL
                  nodepredictor = node[1]
                  accuracy = node[2]
                  result3 =  svm.Predict(nodepredictor,vector)#PREDICT ONE OF THE NODES
                  #print('Node result',result3[0])
                  if (result3[0] == 1.0):
                      predictionresult = nodepair[0]
                  else:
                      predictionresult = nodepair[1]
                  K = N
                  break
              else:#IF THERE IS ONLY ONR NODE IN A CELL
                      predictionresult = nodepair[0]
                      K = N
  else:#IF CURRENT SUBTREE NOT PREDICTED
          if (K == N-1):#CHECK IF ONLY ONE SUBTREE REMAINING
            subtreeno = Key[K]
            TreePredictorList = TreePredictorTree[subtreeno]
            TreePredictorTr  = TreePredictorList[0]
            TreePredictor  = TreePredictorTr[0]
            CellPredictorList = TreePredictorTr[1]
            NodePredictorList  = TreePredictorTr[2]
            CellPredictors = CellPredictorList[subtreeno]
            X = len(CellPredictors)
            cellpredictorskeys = []
            cellpredictor = {}
            i=0
            while i<X:
                predictor = CellPredictors[i]
                for Keys, cells in predictor.items():
                    predictorkey = Keys
                    cellpredictorskeys.append(predictorkey)
                    cellpredictor[predictorkey] = []
                    cellpredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            cellpredictorskeys.sort()
            count=X
            for Keys, cells in cellpredictor.items():
              count=count-1
              #print('if current subtree not predicted,Cell:',cells)
              cell = cells[0]
              currentcell = cell[0]
              othercells = cell[1]
              cellpredictor = cell[2]
              accuracy2 = cell[3]
              result2 =  svm.Predict(cellpredictor,vector)
              #print('if current subtree not predicted,Cell result2',result2[0])
              if (result2[0] == 1.0): #IF CELL PREDICTION IS TRUE
                 #GET NODE PREDICTOR
                  NodePredictors = NodePredictorList[subtreeno]
                  X = len(NodePredictors)
                  nodepredictorskeys = []
```

```
                                nodepredictor = {}
                                i=0
                                while i<X:
                                  predictor = NodePredictors[i]
                                  for Keys, nodes in predictor.items():
                                    predictorkey = Keys
                                    nodepredictorskeys.append(predictorkey)
                                    nodepredictor[predictorkey] = []
                                    nodepredictor[predictorkey].append(predictor[predictorkey])
                                  i=i+1
                                nodepredictorskeys.sort()
                                nodes = nodepredictor[currentcell[0]]
                                #print('if current subtree not predicted,nodes',nodes)
                                node = nodes[0]
                                nodepair = node[0]
                                if len(nodepair)==2:
                                  nodepredictor = node[1]
                                  accuracy = node[2]
                                  result3 =  svm.Predict(nodepredictor,vector)
                                  #print('if current subtree not predicted,Node result',result3[0])
                                  if (result3[0] == 1.0):
                                    predictionresult = nodepair[0]
                                  else:
                                    predictionresult = nodepair[1]
                                else:
                                    predictionresult = nodepair[0]
                                #print('if current subtree not predicted,Node result(+ve)',vector,predictionresult)
                                break
                              else:#IF CELL PREDICTION IS FALSE
                                if (count==0):
                                  if len(othercells)==1:
                                    predictionresult = othercells[0]
                                  #print('if current subtree not predicted,Node result(-ve)',vector,predictionresult)
                                  break
                            K = N
                          else:
                            K = K + 1
               #print('Prediction result for this vector:',vector,predictionresult)
               y = float(predictionresult)
               predictiondata.append([y])
               #predictiondata[i][-1] = float(predictionresult)
               #print('len(testdata),len(predictiondata):',len(testdata),len(predictiondata))
               if vector[-1] == y:
                  if (vector[-1] in correctClassified):
                      #print('count before is:',correctClassified[vector[-1]])
                      count=correctClassified[vector[-1]]
                      correctClassified[vector[-1]] = count+1
                      #print('count after is:',correctClassified[vector[-1]])
                      #print('Yes1')
                  else:
                      correctClassified[vector[-1]] = 1
                      #print('Yes2')
               else:
                  if (vector[-1] in incorrectClassified):
                      count=incorrectClassified[vector[-1]]
                      incorrectClassified[vector[-1]] = count+1
                      #print('No1')
                  else:
                      incorrectClassified[vector[-1]] = 1
                      #print('No2')
      #print('correctClassified:',correctClassified)
      #print('incorrectClassified:',incorrectClassified)
      testFileDistribution = svm.getClassDistribution(testdata)
      classAccuracy = svm.getClassAccuracy(testFileDistribution,correctClassified,incorrectClassified)
      #print('classAccuracy is:',classAccuracy)
      overallaccuracy = self.getAccuracy(testdata,predictiondata)
      #print('THE ACCURACY FOR THIS CLASSIFICATION IS=%:',svm.getAccuracy(testdata,predictiondata))
      return overallaccuracy
#THIS CLASSIFIES ONE INSTANCE AT A TIME USING A STORED TRAINED CLASSIFIER LOADED FROM PICKLE
def classifyOneInstance(self,classifier,classTree,data):
      self.classTree = classTree
      self.data = data
      self.classifier = classifier
      tree = classTree
      testdata = data
      '''
      #RETRIEVE THE CLASSIFIER
      pickle_in = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\SVMclassifier.pickle','rb')
      tp=pickle.load(pickle_in)
      TreePredictors = tp
      '''
      TreePredictors = classifier
      mKey =list(TreePredictors.keys())
      CellPredictorTree = {}
      NodePredictorTree = {}
      TreePredictorTree = {}
      #predictiondata = data
```

```
predictiondata = []
svm = SVMRootclassifier()
vector = testdata
T = 0
while (T <len(mKey)):#CHECK IN EACH MAIN TREE IN WHICH THE INSTANCE BELONGS
    treeno = mKey[T]#GET CLASSIFIER NUMBER
    mainTreePredictorList = TreePredictors[treeno]
    mainTreePredictor =  mainTreePredictorList[0]
    #print('mainTreePredictor[0]', mainTreePredictor[0])
    mainPredictors = mainTreePredictor[0][0.0]
    if (len(mainPredictors)> 1):#CASE OF MORE THAN ONE TREE
            mainPredictor = mainPredictors[0]
            mainTreeResult =  svm.predict(mainPredictor,vector)#MAKE PREDICTION
            if (mainTreeResult[0] ==1.0)and (T <=(len(mKey)- 2)):#IF 1 GET SUBTREE CLASSIFIER
                TreePredictorTree = mainTreePredictor[1]
                #print('FOR MAINTREE NO:', treeno)
                T = len(mKey)+1#END THE LOOP
            else:#IF -1
                if (mainTreeResult[0] == -1.0)and (T >= (len(mKey) - 2)):#CHECK WHETHER IT IS  ECOND LAST
                  treeno = mKey[T+1]#GET GET THE ONLY LAST AND END THEN LOOP
                  mainTreePredictorList = TreePredictors[treeno]
                  mainTreePredictor =  mainTreePredictorList[0]
                  #print('mainTreePredictor[0]', mainTreePredictor[0])
                  TreePredictorTree = mainTreePredictor[1]
                  #print('(this is second last)FOR MAINTREE NO:', treeno)
                  T = len(mKey)+1 #END THE LOOP
                else:#IF NOT SECOND LAST (mainTreeResult[0] == -1.0)and (T < len(mKey) - 2)
                  T = T + 1 #LOOP AGAIN
    else:#CASE OF ONLY ONE TREE
            TreePredictorTree = mainTreePredictor[1]
            T = len(mKey)+1 #END THE LOOP
Key = list(TreePredictorTree.keys())
N = len(Key)
#print('TreePredictorTree', TreePredictorTree)
K = 0
while (K < len(Key)):#CHECK IN EACH SUBTREE THE CELL IN WHICH THE INSTANCE BELONGS
    #print('K',K)
    subtreeno = Key[K]
    TreePredictorList = TreePredictorTree[subtreeno]#TreePredictorList IS A LIST OF ONLY ONE ELEMENT I.E. THIS SUBTREE
    TreePredictorTr  = TreePredictorList[0]#TreePredictorTr IS A LIST OF THREE DICTIONARIES OF THIS SUBTREE PREDICTORS I.E.[{SUBTREE},{CELLS},{NODES}]
    TreePredictor  = TreePredictorTr[0]# TreePredictor IS A DICTIONARY OF THIS SUBTREE PREDICTOR
    CellPredictorList = TreePredictorTr[1] #CellPredictorList IS A DICTIONARY OF THIS SUBTREE CELL PREDICTORS
    NodePredictorList  = TreePredictorTr[2]#NodePredictorList IS A DICTIONARY OF THIS SUBTREE NODE PREDICTORS
    CellPredictors = CellPredictorList[subtreeno]#CellPredictors IS A LIST OF THIS SUBTREE'S CELL PREDICTORS
    Trpredictor = TreePredictor[0.0]#Trpredictor IS A PREDICTOR OF THIS CURRENT SUBTREE
    result1 = svm.Predict(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
    #print('Trpredictor[0]',Trpredictor[0])
    #result1 = nB1.getPredictions(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
    #print('subtree result1',result1)
    if (result1[0] == 1.0):#IF CURRENT SUBTREE PREDICTED YES
      #GET CELL PREDICTORS
      X = len(CellPredictors)
      #print('CellPredictor:for result1=1',CellPredictors)
      if (X>0):#IF THERE ARE CELL PREDICTORS
          cellpredictorskeys = []
          cellpredictor = {}
          i=0
          while i<X:#WHILE THERE ARE CELL PREDICTORS
            predictor = CellPredictors[i]
            for Keys, cells in predictor.items():
              predictorkey = Keys
              cellpredictorskeys.append(predictorkey)
              cellpredictor[predictorkey] = []
              cellpredictor[predictorkey].append(predictor[predictorkey])
            i=i+1
          cellpredictorskeys.sort()#SORT THEM IN THE ORDER THEY WILL BE WORKED ON
          count=X
          for Keys, cells in cellpredictor.items():
            count=count-1 #COUNT CELL PREDICTORS BOTTOM UP
            #print('Cell:',cells)
            cell = cells[0]
            currentcell = cell[0]
            othercells = cell[1]
            cellpredictor = cell[2]
            accuracy2 = cell[3]
            result2 =  svm.Predict(cellpredictor,vector)
            #print('Cell result2',result2[0])
            if (result2[0] == 1.0):  #IF CELL RESULT IS 1 SELECT THE FIRST CELL'S NODE PREDICTORS
              #GET NODE PREDICTOR FOR THIS SUBTREE
              NodePredictors = NodePredictorList[subtreeno]
              #print('NodePredictors',NodePredictors)
              X = len(NodePredictors)
              nodepredictorskeys = []
              nodepredictor = {}
              i=0
              while i<X:
                predictor = NodePredictors[i]
```

```
                for Keys, nodes in predictor.items():
                    predictorkey = Keys
                    nodepredictorskeys.append(predictorkey)
                    nodepredictor[predictorkey] = []
                    nodepredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            nodepredictorskeys.sort()
            #print('nodepredictor',nodepredictor)
            nodes = nodepredictor[currentcell[0]]
            #print('nodes',nodes)
            node = nodes[0]
            nodepair = node[0]
            if len(nodepair)==2:
                nodepredictor = node[1]
                accuracy = node[2]
                result3 =  svm.Predict(nodepredictor,vector)
                #print('Node result',result3[0])
                if (result3[0] == 1.0):
                    predictionresult = nodepair[0]
                else:
                    predictionresult = nodepair[1]

            else:
                    predictionresult = nodepair[0]
            #print('Node result(+ve)',vector,predictionresult)
            break
        else:#IF CELL RESULT IS -1 SELECT THE OTHER CELL'S NODES
            if (count==0):#IF THIS IS THE LAST CELL PREDICTOR FOR THIS SUBTREE
                if len(othercells)==1:#IF THERE IS ONLY ONE NODE IN THIS CELL
                    predictionresult = othercells[0]
                else:#IF THERE IS MORE THAN ONE(TWO) NODES IN THIS CELL
                    NodePredictors = NodePredictorList[subtreeno]
                    X = len(NodePredictors)
                    nodepredictorskeys = []
                    nodepredictor = {}
                    i=0
                    while i<X:
                        predictor = NodePredictors[i]
                        for Keys, nodes in predictor.items():
                            predictorkey = Keys
                            nodepredictorskeys.append(predictorkey)
                            nodepredictor[predictorkey] = []
                            nodepredictor[predictorkey].append(predictor[predictorkey])
                        i=i+1
                    nodepredictorskeys.sort()
                    nodes = nodepredictor[othercells[0]]
                    #print('nodes',nodes)
                    node = nodes[0]
                    nodepair = node[0]
                    if len(nodepair)==2:
                        nodepredictor = node[1]
                        accuracy = node[2]
                        result3 =  svm.Predict(nodepredictor,vector)
                        #print('Node result',result3[0])
                        if (result3[0] == 1.0):
                            predictionresult = nodepair[0]
                        else:
                            predictionresult = nodepair[1]

                    else:
                            predictionresult = nodepair[0]
                    #print('Node result(+ve)',vector,predictionresult)
                    break
    K = N
else:#IF THERE ARE NO CELL PREDICTORS
    #GET NODE PREDICTORS
        NodePredictors = NodePredictorList[subtreeno]
        X = len(NodePredictors)
        nodepredictorskeys = []
        nodepredictor = {}
        i=0
        while i<X:
            predictor = NodePredictors[i]
            for Keys, nodes in predictor.items():
                predictorkey = Keys
                nodepredictorskeys.append(predictorkey)
                nodepredictor[predictorkey] = []
                nodepredictor[predictorkey].append(predictor[predictorkey])
            i=i+1
        nodepredictorskeys.sort()
        currentcell = nodepredictorskeys[0]
        #print('currentcell',currentcell)
        nodes = nodepredictor[currentcell]
        #print('nodes',nodes)
        node = nodes[0]
        nodepair = node[0]
        if len(nodepair)==2:#CHECK IF THERE ARE TWO NODES IN A CELL
```

```
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  svm.Predict(nodepredictor,vector)#PREDICT ONE OF THE NODES
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]
                    K = N
                    break
                else:#IF THERE IS ONLY ONR NODE IN A CELL
                        predictionresult = nodepair[0]
                        K = N
else:#IF CURRENT SUBTREE NOT PREDICTED
        if (K == N-1):#CHECK IF ONLY ONE SUBTREE REMAINING
            subtreeno = Key[K]
            TreePredictorList = TreePredictorTree[subtreeno]
            TreePredictorTr  = TreePredictorList[0]
            TreePredictor  = TreePredictorTr[0]
            CellPredictorList = TreePredictorTr[1]
            NodePredictorList  = TreePredictorTr[2]
            CellPredictors = CellPredictorList[subtreeno]
            X = len(CellPredictors)
            cellpredictorskeys = []
            cellpredictor = {}
            i=0
            while i<X:
                predictor = CellPredictors[i]
                for Keys, cells in predictor.items():
                    predictorkey = Keys
                    cellpredictorskeys.append(predictorkey)
                    cellpredictor[predictorkey] = []
                    cellpredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            cellpredictorskeys.sort()
            count=X
            for Keys, cells in cellpredictor.items():
                count=count-1
                #print('if current subtree not predicted,Cell:',cells)
                cell = cells[0]
                currentcell = cell[0]
                othercells = cell[1]
                cellpredictor = cell[2]
                accuracy2 = cell[3]
                result2 =  svm.Predict(cellpredictor,vector)
                #print('if current subtree not predicted,Cell result2',result2[0])
                if (result2[0] == 1.0): #IF CELL PREDICTION IS TRUE
                    #GET NODE PREDICTOR
                    NodePredictors = NodePredictorList[subtreeno]
                    X = len(NodePredictors)
                    nodepredictorskeys = []
                    nodepredictor = {}
                    i=0
                    while i<X:
                        predictor = NodePredictors[i]
                        for Keys, nodes in predictor.items():
                            predictorkey = Keys
                            nodepredictorskeys.append(predictorkey)
                            nodepredictor[predictorkey] = []
                            nodepredictor[predictorkey].append(predictor[predictorkey])
                        i=i+1
                    nodepredictorskeys.sort()
                    nodes = nodepredictor[currentcell[0]]
                    #print('if current subtree not predicted,nodes',nodes)
                    node = nodes[0]
                    nodepair = node[0]
                    if len(nodepair)==2:
                        nodepredictor = node[1]
                        accuracy = node[2]
                        result3 =  svm.Predict(nodepredictor,vector)
                        #print('if current subtree not predicted,Node result',result3[0])
                        if (result3[0] == 1.0):
                            predictionresult = nodepair[0]
                        else:
                            predictionresult = nodepair[1]
                    else:
                            predictionresult = nodepair[0]
                    #print('if current subtree not predicted,Node result(+ve)',vector,predictionresult)
                    break
                else:#IF CELL PREDICTION IS FALSE
                    if (count==0):
                        if len(othercells)==1:
                            predictionresult = othercells[0]
                        #print('if current subtree not predicted,Node result(-ve)',vector,predictionresult)
                        break
            K = N
        else:
```

```
                K = K + 1
        #print('Prediction result for this vector:',vector,predictionresult)
        y = float(predictionresult)
        return  y
```

```
import tkinter.filedialog
from tkinter import *
#from ScrolledText import *
import tkinter.ttk as ttk
import pandas as pd
import dill as pickle
#import pickle
import random
import sqlite3
import logging
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import itertools
import csv
import decimal
import math
import copy
import time
import threading
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from sklearn.preprocessing import StandardScaler
splitRatio = 0.80
splitRatio2 = 0.10
class naiveRootclassifier():
    def __init__(self):
        #self.master = master
        self.value = None

    def loadCsv(self,filename):
        self.filename=filename
        lines = csv.reader(open(filename, "r"))
        dataset = list(lines)
        for i in range(len(dataset)):
            #print(dataset[i])
            dataset[i] = [float(x) for x in dataset[i]]
        return dataset

    def scaler(self,datatrain,datatest):
        self.datatrain = datatrain
        self.datatest = datatest
        stand = StandardScaler()
        dataframe1 = pd.DataFrame(datatrain)
        array1 = dataframe1.values
        n1 = len(datatrain[0])-1
        m1 = []
        set1 = []
        X1 = array1[:,0:n1]
        Y1 = array1[:,n1]
        X1_std = stand.fit_transform(X1)
        count =0
        for i in X1_std:
            #print("before",i)
            #print("before",Y[count])
            m1 = list(i)
            m1.append(Y1[count])
            set1.append(m1)
            #print("after",m)
            count = count+1
        dataframe2 = pd.DataFrame(datatest)
        array2 = dataframe2.values
        n2 = len(datatest[0])-1
        m2 = []
        set2 = []
        X2 = array2[:,0:n2]
        Y2 = array2[:,n2]
        X2_std = stand.transform(X2)
        count =0
        for i in X2_std:
            #print("before",i)
            #print("before",Y[count])
            m2 = list(i)
            m2.append(Y2[count])
            set2.append(m2)
            #print("after",m)
            count = count+1
        return set1,set2

    def splitDataset2(self, dataset, splitRatio):# splits the dataset into two: training and test dataset
```

```
        self.dataset=dataset
        self.splitRatio=splitRatio
        freq = self.getClassDistribution(dataset)
        trainSet = []
        copy = []
        #print('dataset',dataset)
        #print('frequency:',freq)
        for keys,frequency in freq.items():
            trainSize = int(frequency * splitRatio)
            #print(keys,  frequency,  trainSize)
            fold = []
            trainFold = []
            for k in range(len(dataset)):
                vector1=list(dataset[k])
                if (vector1[-1]==keys):
                    fold.append(vector1)
            #print('frequency:len(fold):trainSize', frequency,len(fold),trainSize)
            while len(trainFold) < trainSize:
                index = random.randrange(len(fold))
                trainFold.append(fold.pop(index))
                #print('fold',  fold)
            #trainSet.append(trainFold)
            trainSet= trainSet + trainFold
            #copy.append(fold)
            copy = copy + fold
            #print('copy',  copy)
        return [trainSet, copy]

def splitDataset(self, dataset, splitRatio):# splits the dataset into two: training and test dataset
        self.dataset=dataset
        self.splitRatio=splitRatio
        trainSize = int(len(dataset) * splitRatio)
        trainSet = []
        copy = list(dataset)
        while len(trainSet) < trainSize:
             index = random.randrange(len(copy))
             trainSet.append(copy.pop(index))
        return [trainSet, copy]

#TAKES IN REQUIRED CLASSES AND FILTERS THE DATASET TO REMAIN WITH INSTANCES OF ONLY THESE CLASSES
def refineDataset(self, dataset, mergeclasses):# removes unwanted classes from dataset
        self.dataset=dataset
        self.classValue = mergeclasses
        dataset2 = []
        for i in range(len(dataset)):
              vector1=dataset[i]
              if vector1[-1] in mergeclasses:
                  dataset2.append(vector1)
        return dataset2

#CLASSES AND FILTERS THE DATASET TO REMAIN WITH INSTANCES OF ONLY THESE CLASSES getClassDistirbution
def getClassDistribution(self,dataset):
        dataset2 = {}
        counts = {}
        #print('trainSet',  dataset)
        for i in range(len(dataset)):
             vector1 = dataset[i]
             #print('vector1',vector1)
             #print('vector1[-1]',vector1[-1])
             if (vector1[-1] in counts):
                  counts[vector1[-1]] += 1
             else:
                  counts[vector1[-1]]=1
        return counts

def getClassAccuracy(self,testFile,correctClassified,incorrectClassified):
        self.testFile = testFile
        self.correctClassified = correctClassified
        self.incorrectClassified = incorrectClassified
        classAccuracy = {}
        for classValue,freq in testFile.items():
            if classValue in correctClassified:
              n = freq
              x = correctClassified[classValue]
              y = x * 100/n
            else:
              y = 0.0
            classAccuracy[classValue] = y
        return classAccuracy

def getClassCodes(self,dataset,parentclass):
        self.dataset=dataset
        dataset2 = []
        self.parentclass = parentclass
        #print('parentclass',parentclass)
        for i in range(len(dataset)):
             vector1 = dataset[i]
```

```
            vector = dataset[i]
            value = float(parentclass)
           # print('parentclass',type(value),type(vector1[-1]))
            if (vector1[-1] == value):
                #print(i,'class for 1',vector1[-1],value)
                vector1[-1] = 1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)
            else:
                #print(i,'class for -1',vector1[-1],value)
                vector1[-1] = -1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)
        return dataset2

def separateByClass(self,dataset,mergeclass):#this separates instances into distinct classes i.e isolates class instances
        self.dataset=dataset
        self.mergeclass = mergeclass
        separated = {}
        dataset2 = []
        dataset3 = []
        positives = mergeclass[1]
        negatives = mergeclass[-1]
        for k in range(len(dataset)):
            vector1=list(dataset[k])
            #print('vector1=dataset[k]:k',k)
            if vector1[-1] in negatives:
                vector1[-1] = -1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)
            if vector1[-1] in positives:
                vector1[-1] = 1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)
        return dataset2

# merges and then separates instances into two distinct classes
def separateByRootClass(self,dataset,mergeclass):
        #MERGECLASS IS A DICTIONARY CONTAINING TWO KEYS EACH WITH CLASSES BELONGING TO EACH KEY I.E. {1:positivenodes, -1:negativenodes}
        self.dataset=dataset
        self.mergeclass = mergeclass
        separated = {}
        dataset2 = []
        dataset3 = []
        positives = mergeclass[1]
        negatives = mergeclass[-1]
        for k in range(len(dataset)):
            vector1=list(dataset[k])
            if vector1[-1] in negatives:
                vector1[-1] = -1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)

            if vector1[-1] in positives:
                vector1[-1] = 1
                vector1[-1]=float(vector1[-1])
                dataset2.append(vector1)
        for i in range(len(dataset2)):
            vector = dataset2[i]
            if (vector[-1] not in separated):
                separated[vector[-1]] = []
            separated[vector[-1]].append(vector)
        return separated

def summarizeByClass(self,dataset,mergeclass):# produces attribute-based means and std. devs for each class in the dataset
        self.dataset=dataset
        self.mergeclass = mergeclass
        separated = self.separateByRootClass(dataset,mergeclass)
        summaries = {}
        #separated is a dictionary containing instances grouped into positives and negatives
        for classValue, instances in separated.items():
            summaries[classValue] = self.summarize(instances)
        #print('summaries are',summaries)
        #summaries is a dictionary containing mean and standard deviation of each attribute but grouped according to classes
        return summaries

def summarize(self,instances):#calculates mean and std. dev. for each attribute in the given instances set
        self.instances=instances
        summaries = [(self.mean(attribute), self.stdev(attribute)) for attribute in zip(*instances)]
        del summaries[-1]
        return summaries

def mean(self,numbers):
        self.numbers=numbers
        #print('attribute values are:',numbers)
        return sum(numbers)/float(len(numbers))
```

```
def stdev(self,numbers):
        #numbers here is a sequence values of one attribute
        self.numbers=numbers
        avg = self.mean(numbers)
        if len(numbers) > 1:
          variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
        else:
          variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers))
        sdt = math.sqrt(variance)
        if sdt == 0.0:
          #print('numbers:',numbers,'mean is:',avg, 'std is:', sdt,)
          sdt = 0.1
        return sdt

def calculateProbability(self,x, mean, stdev):#calcualates conditional probability of an attribute instance given attribute mean and std dev
        self.x=x
        self.mean=mean
        self.stdev=stdev
        #print('The value of x is:',x,'mean is:',mean,'std. dev. is:',stdev)
        exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
        prob = (1 / (math.sqrt(2*math.pi) * stdev)) * exponent
        return prob

def calculateClassProbabilities(self,summaries, instanceVector):#calculates probabilitie of each instance towards each each class
        self.summaries=summaries
        self.instanceVector=instanceVector
        probabilities = {}
        #print('summaries :',summaries)
        #print('input vector :',inputVector)
        for classValue, attributeSummaries in summaries.items():
                #probabilities = {}
                probabilities[classValue] = 1
                #attributeSummaries is an array of paired mean and std deviation for each attribute
                for i in range(len(attributeSummaries)):#FOR EACH ATTRIBUTE
                        mean, stdev = attributeSummaries[i]
                        x = instanceVector[i]#X IS VALUE OF A GIVEN ATTRIBUTE i
                        #GET PROBILITY OF THIS ATTRIBUTE AND MULTIPLY BY PROBABILITIES OF OTHER ATTRIBUTES IN EACH CLASS TO GET PROBALITY OF THE CLASS
                        probabilities[classValue] = probabilities[classValue]*self.calculateProbability(x, mean, stdev)
        #probabilities is  DICTIONARY CLASS PROBABILITIES
        return probabilities

def predict(self,summaries, inputVector):
        self.summaries=summaries
        self.inputVector=inputVector
        #print('summaries:',summaries)
        probabilities = self.calculateClassProbabilities(summaries, inputVector)
        #print('The probabilities are:',probabilities)
        #print('The input vector is:',inputVector)
        bestLabel, bestProb = None, -1
        for classValue, probability in probabilities.items():
                if bestLabel is None or probability > bestProb:
                        bestProb = probability
                        bestLabel = classValue
        #print('bestLabel:',bestLabel)
        return [bestLabel, bestProb]

def getPredictions(self, summaries, testSet):
        self.summaries=summaries# summary of mean and std dev of each atribute in each class
        self.testSet=testSet
        predictions = []
        #print('The summary is:',summaries)
        #print('len(testSet):',len(testSet))
        #print('testSet:',testSet)
        for i in range(len(testSet)):
                #print('testSet[i]:',testSet[i])
                result, prob = self.predict(summaries, testSet[i])
                predictions.append(result)
        return predictions

def getAccuracy(self, testSet, predictions):
        self.testSet=testSet
        self.predictions=predictions
        correct = 0
        result = 0
        for i in range(len(testSet)):
                #print('testSet[i]:predictions[i]',testSet[i],predictions[i])
                if testSet[i][-1] == predictions[i]:
                        #print('correct:before',correct)
                        correct += 1
                        #print('correct:after',correct)
                if (len(testSet)>0):
                        result = correct/float(len(testSet)) * 100.0
        return result
'''
def Predict(self,predictor, X):
        self.predictor = predictor
        self.X = X
```

```
        #print('X',X,predictor)
        result = []
        vector = X[0:len(X)-1]
        result.append(predictor.predict(vector))
        #result.append(svmpy.SVMPredictor.predict(vector))
        return result

def getPredictions(self, predictor, testdata):
        self.predictor = predictor# summary of mean and std dev of each atribute in each class
        self.testdata = testdata
        predictions = []
        test = []
        for i in range(len(testdata)):
                #test = testSet[i]
                test.append(testdata[i])

        #print('The testset is:',testdata)
        for i in range(len(test)):
                vector = test[i]
                result = self.Predict(predictor, vector)
                #print('Testset prediction is:',test[i],result)
                predictions.append(result)
        return predictions

def getAccuracy(self, testdata, predictions):
        self.testdata=testdata
        self.predictions=predictions
        correct = 0
        #print('Testset prediction is:',testdata,predictions)
        for i in range(len(testdata)):
                #print('Testset prediction is:',testdata[i],predictions[i])
                vector = testdata[i]
                value1= predictions[i]
                value2 = vector[-1]
                #print('Testset prediction is:',value1[0],value2)
                if (value1[0] == value2):
                        #print('Testset prediction is:',vector[-1],predictions[i])
                        correct += 1
        return (correct/float(len(testdata))) * 100.0
'''
def getLevelNodes(self, classTree,level):
        self.classTree = classTree
        self.level = level
        parent = []
        childs = []
        levelNodes = {}
        for classValue,instances in classTree.items():
               for i in range(len(instances)):
                        classlev = instances[0]
                        classlevel = classlev[0]
                        parent = instances[1]
                        childs = instances[2]
                        if (classlevel == level): #only for this level
                                levelNodes[classValue] = [parent,childs]
        return levelNodes

def getTreeDepth(self, classTree):
        self.classTree = classTree
        depth = 0
        for classValue,instances in classTree.items():
               for i in range(len(instances)):
                        classlevel = instances[0]
                        level = classlevel
                        if(len(level) > 0):
                                #print(type(level[0]),type(depth))
                                #print(level[0],depth)
                                if (int(level[0]) > depth): #check leve of the current node
                                        depth = level[0]

        return depth

def getChildrenOf(self, classvalue,classTree):
        self.classvalue = classvalue
        self.classTree = classTree
        childs = []
        #print('class value is',classvalue)
        for classValue,instances in classTree.items():
               #print('classValue,classvalue',classValue,classvalue)
               if (classValue == classvalue):
                        #print('instances',instances)
                        #for i in range(len(instances)):
                                #parent = instances[1]
                                childs = instances[2]
                                #print('instances',instances[2])
        #print('childs',childs)
        return childs
```

```
def getSubTrees(self, classTree):
        self.classTree = classTree
        childs = []
        subTrees = {}
        height = self.getTreeDepth(classTree)
        top = 0
        if (height > 0):
              topNodes= self.getLevelNodes(classTree,top)
              #print('THESE ARE TOPNODES',topNodes)
              for classValue,instances in topNodes.items():
                     if (classValue < 0):
                            nextparent = classValue
                            #print('nextparent,classvalue',nextparent,classValue)
                            #childs = self.getChildrenOf(nextparent,classTree)
                            while nextparent<0:
                                 classes = []
                                 childs = self.getChildrenOf(nextparent,classTree)
                                 #print('childs of: ',nextparent,'are:',childs)
                                 nextparent = 0
                                 for i in range(len(childs)):
                                       if (childs[i]>0):
                                             classes.append(childs[i])
                                       else:
                                             nextparent=childs[i]

                                 if classValue not in subTrees:
                                       subTrees[classValue] = []
                                 subTrees[classValue].append(classes)
        return subTrees

def getMainTrees(self, classTree):
        self.classTree = classTree
        childs = []
        mainTree = {}
        Tree = {}
        for classValue,instances in classTree.items():
              #if (classValue < 0):
              maintreeid = instances[3]
              #print('maintreeid:',maintreeid)
              if maintreeid[0]>0:
                   if maintreeid[0] not in mainTree:
                         mainTree[maintreeid[0]] = []
                         Tree = {}
                         classes = []
                         classes = [instances[0],instances[1],instances[2]]
                         Tree[classValue] = []
                         Tree[classValue] = classes
                         mainTree[maintreeid[0]] = Tree
                   else:
                         classes = []
                         classes = [instances[0],instances[1],instances[2]]
                         Tree = mainTree[maintreeid[0]]
                         if classValue not in Tree:
                              Tree[classValue] = []
                         Tree[classValue] = classes
                         mainTree[maintreeid[0]] = Tree
                   #print('maintree:',mainTree)
        return mainTree

def orderByParents(self, classNodes):
        self.classNodes = classNodes
        print(classNodes)
        orderedByParent = {}
        parentList = []
        for classValue,instances in classNodes.items():
               #for i in range(len(instances)):
               parent = instances[0]
               #print('parent',parent[0])
               if(len(parent)!=0):
                     if parent[0] not in orderedByParent:
                           #parentList = classValue
                           orderedByParent[parent[0]]=[]

                     parentList=orderedByParent[parent[0]]
                     #print('parentList',parentList)
                     if classValue not in parentList:
                           if (len(parentList)==0):
                                 parentList = [classValue]
                           else:
                                 parentList.append(classValue)
                           #print('classValue',classValue)
                           #print('parentList',parentList)
                           orderedByParent[parent[0]] = parentList
                           #print('orderedByParent',orderedByParent)
        return orderedByParent

def getParentNode(self, childnode,classTree):
```

```
        self.childnode = childnode
        self.classTree = classTree
        parent = []
        for classValue,instances in classTree.items():
            if (childnode in instances):
                parent = [classvalue]
                continue
        return parent

#THIS CREATES A HIERARCHICAL MULTI- CLASSIFIER USING NAIVE BAYES APPROACH
def classify(self,mainTree,trainingSet):
    nB1 = naiveRootclassifier()
    nB2 = naiveRootclassifier()
    nB3 = naiveRootclassifier()
    nB4 = naiveRootclassifier()
    #TreePredictorTree = {}
    TreePredictors = {}
    #mainTreePredictor = {}
    #TreePredictor = {}
    self.trainingSet = trainingSet
    self.mainTree = mainTree
    trainset = list(trainingSet)
    #print("trainset:",len(trainset))
    maintrees = nB1.getMainTrees(mainTree)
    mKey = maintrees.keys()
    otherTrees = []
    AllTrees = []
    value = []
    for mKeys, classTree in maintrees.items():
            for mK, trees in classTree.items():
                value = mK
                #print('mK:',mK)
                if mK > 0:
                    value =[mK]
                    AllTrees = AllTrees + value
                    otherTrees = otherTrees + value
    #print('ALLTREES and mKey :',AllTrees,mKey)
    treeno = 0
    #FOR EACH MAIN TREE
    for mKeys, classTree in maintrees.items():
            #print('CLASSTREES:',classTree)
            #COUNT MAIN TREES
            treeno = treeno + 1
            depth = nB1.getTreeDepth(classTree)#GET DEPTH/HEIGHT OF EACH MAIN TREE
            subtrees = nB1.getSubTrees(classTree)#GET SUBTREES/BRANCHES OF EACH MAIN TREE
            Key = list(subtrees.keys())#GET THE SUBTREE ID'S
            #print('SUBTREES and Key:',subtrees,Key)
            trainset = list(trainingSet)
            #print('TRAIN SET :',len(trainset))
            Allclasses = []
            otherClasses = []
            TreePredictorTree = {}
            CellPredictorTree = {}
            NodePredictorTree = {}
            #'''
            #FOR EACH SUBTREE/BRANCH OF THE MAIN TREE GET ALL THE CLASSES
            for Keys, cells in subtrees.items():
                for i in range(len(cells)):
                    Allclasses = Allclasses + cells[i]
                    otherClasses = otherClasses + cells[i]
            #print('ALL CLASSES SET :',Allclasses,type(otherClasses))
            order = 0
            #FOR EACH SUBTREE/BRANCH OF THE MAIN TREE GET NODE AND CELL CLASSIFIERS
            for Keys, cells in subtrees.items():
                if len(Key)>1:
                    order = order + 1#COUNT SUBTREES/BRANCHES
                cell1 = []
                cellorder = 0
                CellPredictorTree = {}
                NodePredictorTree = {}
                #''
                #IN EACH SUBTREE/BRANCH CELL
                #trainingSet3=trainset
                #print('len(trainingSet3):len(trainingSet3[0])-1',len(trainingSet3),len(trainingSet3[0])-1)
                for i in range(len(cells)):
                    cellorder = cellorder + 1#COUNT CELLS IN EACH SUBTREE
                    nodes = cells[i]
                    #CREATE NODE PREDICTOR
                    #print('NODES:',nodes)
                    if len(nodes) == 2:#IF ONLY TWO LEAF NODES IN EACH CELL CREATE NODE CLASSIFIER FOR EACH
                        nB1 = naiveRootclassifier()
                        #trainset = nB1.loadCsv(filename)
                        trainingSet3=list(trainset)
                        #print("trainingSet3:",len(trainingSet3))
                        currentnode = []
                        othernode = []
                        currentnode = [nodes[0]]
```

```
                othernode = [nodes[1]]
                mergeclass = {1:currentnode,-1:othernode}
                #print('len(trainingSet3):',len(trainingSet3))
                trainingSet3, testSet3 = nB1.splitDataset2(trainingSet3, splitRatio)
                #trainingSet3 = nB1.separateByRootClass(trainingSet3,mergeclass)
                #print('len(trainingSet3):',len(trainingSet3))
                testSet3 = nB1.separateByClass(testSet3,mergeclass)
                #print('getClassDistribution:trainset2',nB1.getClassDistribution(trainingSet3))
#print('getClassDistribution:trainset2',nB1.getClassDistribution(trainingSet3),mergeclass)
                summary3 = nB1.summarizeByClass(trainingSet3,mergeclass)
                #print('len(trainingSet3):',len(trainingSet3))
                features = len(trainingSet3[0])-1
                dataframe = pd.DataFrame(testSet3)
                array1 = dataframe.values
                X = []
                X = array1[:,0:features]
                predictions3 = nB1.getPredictions(summary3,X)
                accuracy3 = nB1.getAccuracy(testSet3, predictions3)
                NodePredictor = {currentnode[0]:[currentnode+othernode,summary3,accuracy3]}
                #print('PREDICTION ACURACY FOR NODES:',nodes,accuracy3)
            else:#IF ONLY ONE LEAF NODE IN EACH CELL CREATE NODE CLASSIFIER FOR ONLY ONE NODE
                predictor=[]
                accuracy3='100%'
                NodePredictor = {nodes[0]:[nodes,predictor,accuracy3]}
                #print('PREDICTION ACURACY FOR NODES:',nodes[0],': IS:',accuracy3)
            #print('THE NODE PREDICTOR :',NodePredictor)
            if (order not in NodePredictorTree):
                    NodePredictorTree[order] = []
            NodePredictorTree[order].append(NodePredictor)#STORE NODE CLASSIFIERS ACCORDING TO THEIR CELL NUMBER
            #print('NODE PREDICTOR TREE:',NodePredictorTree)
            #CREATE HIERARCHICAL CELL CLASSIFIERS
            CellPredictor = []
            if cellorder<=len(cells)-1:#CREATE ONE AGAINST ALL(REMIANING CELLS) CELL CLASSIFIERS
                nB2 = naiveRootclassifier()
                #trainset = nB2.loadCsv(filename)
                trainingSet2=list(trainset)
                #print('getClassDistribution:trainset1',nB1.getClassDistribution(trainingSet2))
                othercells = cell1 + cells[i]
                currentcell = cells[i+1]
                cell1 = cells[i]
                mergeclass = {1:currentcell,-1:othercells}
                trainingSet2, testSet2 = nB2.splitDataset2(trainingSet2, splitRatio)
                #trainingSet2 = nB2.separateByRootClass(trainingSet2,mergeclass)
                testSet2 = nB2.separateByClass(testSet2,mergeclass)
#print('getClassDistribution:trainset2',nB1.getClassDistribution(trainingSet2),mergeclass)
                summary2 = nB2.summarizeByClass(trainingSet2,mergeclass)
                features = len(trainingSet2[0])-1
                dataframe = pd.DataFrame(testSet2)
                array1 = dataframe.values
                X = []
                X = array1[:,0:features]
                predictions2 = nB2.getPredictions(summary2,X)
                accuracy2 = nB2.getAccuracy(testSet2, predictions2)
                CellPredictor ={len(cells)-cellorder:[currentcell,othercells,summary2,accuracy2]}
                #print('ACCURACY FOR CELL PREDICTION',currentcell,' IS: %=', accuracy2)
            #print('THE CELL PREDICTOR IS:',CellPredictor)
            if (order not in CellPredictorTree):
                CellPredictorTree[order] = []
            if len(CellPredictor) > 0:
                CellPredictorTree[order].append(CellPredictor)
            #print('CELL PREDICTOR TREE:',CellPredictorTree)
            #''
        #CREATE SUBTREE CLASSIFIERS
        #print("order",order)
        if (order<=len(Key)-1):
            nB3 = naiveRootclassifier()
            #trainset = nB3.loadCsv(filename)
            trainingSet1=list(trainset)
            currentTree = []
            for i in range(len(cells)):
                currentTree = currentTree + cells[i]
            for i in range(len(currentTree)):
                otherClasses.remove(currentTree[i])
            others = []
            for j in range(len(otherClasses)):
                others.append(otherClasses[j])
            mergeclass = {1:currentTree,-1:others}
            trainingSet1, testSet1 = nB3.splitDataset2(trainingSet1, splitRatio)
            #trainingSet1 = nB1.separateByRootClass(trainingSet1,mergeclass)
            testSet1 = nB3.separateByClass(testSet1,mergeclass)
            summary1 = nB3.summarizeByClass(trainingSet1,mergeclass)
            features = len(trainingSet1[0])-1
            dataframe = pd.DataFrame(testSet1)
            array1 = dataframe.values
            X = []
            X = array1[:,0:features]
            predictions1 = nB3.getPredictions(summary1,X)
```

```
                accuracy1 = nB3.getAccuracy(testSet1, predictions1)
                TreePredictor = {1.0:currentTree,-1.0:others,0.0:[summary1,accuracy1],}
                #print('ACCURACY FOR SUBTREE PREDICTION',currentTree, 'IS: %=', accuracy1)
                #print('THE SUBTREE PREDICTION:',TreePredictor)
            if (order not in TreePredictorTree):
                    TreePredictorTree[order] = []
            TreePredictorTree[order].append([TreePredictor,CellPredictorTree,NodePredictorTree])
        #'''
        #print('TreePredictorTree:',TreePredictorTree)

        #CREATE TREE CLASSIFIERS
        if (treeno<=len(mKey)-1):
            nB4 = naiveRootclassifier()
            #trainset = nB4.loadCsv(filename)
            trainingSet1=list(trainset)
            currentMainTree = []
            for mK, trees in classTree.items():
                if mK > 0:
                    value = [mK]
                    currentMainTree = currentMainTree + value
            for i in range(len(currentMainTree)):
                otherTrees.remove(currentMainTree[i])
            others = []
            for j in range(len(otherTrees)):
                others.append(otherTrees[j])
            mergeclass = {1:currentMainTree,-1:others}
            trainingSet1, testSet1 = nB4.splitDataset2(trainingSet1, splitRatio)
            #trainingSet1 = nB1.separateByRootClass(trainingSet1,mergeclass)
            testSet1 = nB4.separateByClass(testSet1,mergeclass)
            summary1 = nB4.summarizeByClass(trainingSet1,mergeclass)
            features = len(trainingSet1[0])-1
            dataframe = pd.DataFrame(testSet1)
            array1 = dataframe.values
            X = []
            X = array1[:,0:features]
            predictions2 = nB4.getPredictions(summary1,X)
            accuracy2 = nB4.getAccuracy(testSet1, predictions2)
            mainTreePredictor = {1.0:currentMainTree,-1.0:others,0.0:[summary1,accuracy2],}
            #print('ACCURACY FOR MAIN TREE PREDICTION IS:', accuracy2)
        else:
            #print('TWIN1 IS:treeno,len(mKey)',treeno,len(mKey) )
            if (len(mKey)==1):
                predictor=[]
                accuracy2='100%'
                mainTreePredictor = {1.0:[],-1.0:[],0.0:[accuracy2],}
                #mainTreePredictor = {mKeys:[predictor,accuracy2]}

        #print('MAIN TREE PREDICTOR IS:', mainTreePredictor)
        #'''
        #if (mKeys not in classifier):
        if (mKeys not in TreePredictors):
                TreePredictors[mKeys] = []
        TreePredictors[mKeys].append([mainTreePredictor,TreePredictorTree])
    #print('ALL MAIN TREES PREDICTORS ARE:',len(TreePredictors))
    #print('ALL MAIN TREES PREDICTORS ARE:', TreePredictors)
    '''
    for tree, trees in TreePredictors.items():
        #TreePredictorList = trees[0]
        mainTreePredictor =  trees[0]

        mainPredictors = mainTreePredictor[0][0.0]
        print('Key: 1.0', tree,len(mainPredictors))
        #mainPredictors = mainTreePredictor[-1]
        print('Key: -1.0', tree,mainPredictors)
    '''
    #pickle_out = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\multiclassifier.pickle','wb')
    #pickle.dump(TreePredictors,pickle_out)
    #pickle_out.close()
    #print('ALL MAIN TREES PREDICTORS ARE:', TreePredictors.keys())
    return TreePredictors
#THIS CLASSIFIES WHOLE DATASET USING NAIVE BAYES CLASSIFIERS
def classifyInstance(self,classifier,classTree,data):
    self.classifier = classifier
    self.classTree = classTree
    self.data = data
    #print('TreePredictors keys:',TreePredictors.keys())
    tree = classTree
    testdata = data
    #mKey =list(TreePredictors.keys())
    #print('TreePredictors keys:',mKey)
    mKey =list(classifier.keys())
    #TreePredictorTree = {}
    CellPredictorTree = {}
    NodePredictorTree = {}
    #predictiondata = data
    predictiondata = []
    correctClassified = {}
```

```
incorrectClassified = {}
predictionresult = -1
for i in range(len(testdata)):
        #mainTreePredictorList = {}
        #mainTreePredictor = {}

        nB1 = naiveRootclassifier()
        X = testdata[i]
        vector = X[0:len(X)-1]
        T = 0
        while (T <len(mKey)):#CHECK IN EACH MAIN TREE IN WHICH THE INSTANCE BELONGS
            treeno = mKey[T]#GET CLASSIFIER NUMBER
            #mainTreePredictorList = TreePredictors[treeno]
            mainTreePredictorList = classifier[treeno]
            mainTreePredictor =  mainTreePredictorList[0]
            mainPredictors = mainTreePredictor[0][0.0]
            #print('mainPredictors', mainPredictors)
            if (len(mainPredictors)> 1):#CASE OF MORE THAN ONE TREE
                    mainPredictor = mainPredictors[0]
                    #print('mainPredictor', mainPredictor)
                    mainTreeResult =  nB1.predict(mainPredictor,vector)#MAKE PREDICTION
                    if (mainTreeResult[0] ==1.0)and (T <=(len(mKey)- 2)):#IF 1 GET SUBTREE CLASSIFIER
                        TreePredictorTree = mainTreePredictor[1]
                        #print('FOR MAINTREE NO:', treeno)
                        T = len(mKey)+1#END THE LOOP
                    else:#IF -1
                        if (mainTreeResult[0] == -1.0)and (T >= (len(mKey) - 2)):#CHECK WHETHER IT IS SECOND LAST
                            treeno = mKey[T+1]#GET GET THE ONLY LAST AND END THEN LOOP
                            #mainTreePredictorList = TreePredictors[treeno]
                            mainTreePredictorList = classifier[treeno]
                            mainTreePredictor =  mainTreePredictorList[0]
                            #print('mainTreePredictor[0]', mainTreePredictor[0])
                            TreePredictorTree = mainTreePredictor[1]
                            #print('(this is second last)FOR MAINTREE NO:', treeno)
                            T = len(mKey)+1 #END THE LOOP
                        else:#IF NOT SECOND LAST (mainTreeResult[0] == -1.0)and (T < len(mKey) - 2)
                            T = T + 1 #LOOP AGAIN
            else:#CASE OF ONLY ONE TREE
                    TreePredictorTree = mainTreePredictor[1]
                    #print('CASE OF ONLY ONE TREE',TreePredictorTree[2])
                    #mainTreePredictor =  mainTreePredictorList[1]
                    T = len(mKey)+1 #END THE LOOP
        Key = list(TreePredictorTree.keys())
        N = len(Key)
        #print('TreePredictorTree', TreePredictorTree)
        #print('Key', Key)
        K = 0
        while (K < len(Key)):#CHECK IN EACH SUBTREE THE CELL IN WHICH THE INSTANCE BELONGS
            #TreePredictorTr = {}
            #TreePredictor = {}
            #Trpredictor = {}
            subtreeno = Key[K]
            TreePredictorList = TreePredictorTree[subtreeno]#TreePredictorList IS A LIST OF ONLY ONE ELEMENT I.E. THIS SUBTREE
            TreePredictorTr  = TreePredictorList[0]#TreePredictorTr IS A LIST OF THREE DICTIONARIES OF THIS SUBTREE PREDICTORS I.E.[{SUBTREE},{CELLS},{NODES}]
            TreePredictor  = TreePredictorTr[0]# TreePredictor IS A DICTIONARY OF THIS SUBTREE PREDICTOR
            #print('TreePredictor ',TreePredictor)
            CellPredictorList = TreePredictorTr[1] #CellPredictorList IS A DICTIONARY OF THIS SUBTREE CELL PREDICTORS
            NodePredictorList  = TreePredictorTr[2]#NodePredictorList IS A DICTIONARY OF THIS SUBTREE NODE PREDICTORS
            CellPredictors = CellPredictorList[subtreeno]#CellPredictors IS A LIST OF THIS SUBTREE'S CELL PREDICTORS
            Trpredictor = TreePredictor[0.0]#Trpredictor IS A PREDICTOR OF THIS CURRENT SUBTREE
            result1 = nB1.predict(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
            #print('Trpredictor[0]',len(Trpredictor[0]))
            #result1 = nB1.getPredictions(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
            #print('subtree result1',result1)
            if (result1[0] == 1.0):#IF CURRENT SUBTREE PREDICTED YES
              #GET CELL PREDICTORS
              X = len(CellPredictors)
              #print('CellPredictor:for result1=1',CellPredictors)
              if (X>0):#IF THERE ARE CELL PREDICTORS
                  cellpredictorskeys = []
                  cellpredictor = {}
                  i=0
                  while i<X:#WHILE THERE ARE CELL PREDICTORS
                     predictor = CellPredictors[i]
                     for Keys, cells in predictor.items():
                        predictorkey = Keys
                        cellpredictorskeys.append(predictorkey)
                        cellpredictor[predictorkey] = []
                        cellpredictor[predictorkey].append(predictor[predictorkey])
                     i=i+1
                  cellpredictorskeys.sort()#SORT THEM IN THE ORDER THEY WILL BE WORKED ON
                  count=X
                  for Keys, cells in cellpredictor.items():
                    count=count-1 #COUNT CELL PREDICTORS BOTTOM UP
                    #print('Cell:',cells)
                    cell = cells[0]
                    currentcell = cell[0]
```

```
            othercells = cell[1]
            cellpredictor = cell[2]
            accuracy2 = cell[3]
            result2 =  nB1.predict(cellpredictor,vector)
            #print('Cell result2',result2[0])
            if (result2[0] == 1.0):  #IF CELL RESULT IS 1 SELECT THE FIRST CELL'S NODE PREDICTORS
                #GET NODE PREDICTOR FOR THIS SUBTREE
                NodePredictors = NodePredictorList[subtreeno]
                #print('NodePredictors',NodePredictors)
                X = len(NodePredictors)
                nodepredictorskeys = []
                nodepredictor = {}
                i=0
                while i<X:
                    predictor = NodePredictors[i]
                    for Keys, nodes in predictor.items():
                        predictorkey = Keys
                        nodepredictorskeys.append(predictorkey)
                        nodepredictor[predictorkey] = []
                        nodepredictor[predictorkey].append(predictor[predictorkey])
                    i=i+1
                nodepredictorskeys.sort()
                #print('nodepredictor',nodepredictor)
                nodes = nodepredictor[currentcell[0]]
                #print('nodes',nodes)
                node = nodes[0]
                nodepair = node[0]
                if len(nodepair)==2:
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  nB1.predict(nodepredictor,vector)
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]
                else:
                        predictionresult = nodepair[0]
                #print('Node result(+ve)',vector,predictionresult)
                break
            else:#IF CELL RESULT IS -1 SELECT THE OTHER CELL'S NODES
                if (count==0):#IF THIS IS THE LAST CELL PREDICTOR FOR THIS SUBTREE
                    if len(othercells)==1:#IF THERE IS ONLY ONE NODE IN THIS CELL
                        predictionresult = othercells[0]
                    else:#IF THERE IS MORE THAN ONE(TWO) NODES IN THIS CELL
                        NodePredictors = NodePredictorList[subtreeno]
                        X = len(NodePredictors)
                        nodepredictorskeys = []
                        nodepredictor = {}
                        i=0
                        while i<X:
                            predictor = NodePredictors[i]
                            for Keys, nodes in predictor.items():
                                predictorkey = Keys
                                nodepredictorskeys.append(predictorkey)
                                nodepredictor[predictorkey] = []
                                nodepredictor[predictorkey].append(predictor[predictorkey])
                            i=i+1
                        nodepredictorskeys.sort()
                        nodes = nodepredictor[othercells[0]]
                        #print('nodes',nodes)
                        node = nodes[0]
                        nodepair = node[0]
                        if len(nodepair)==2:
                            nodepredictor = node[1]
                            accuracy = node[2]
                            result3 =  nB1.predict(nodepredictor,vector)
                            #print('Node result',result3[0])
                            if (result3[0] == 1.0):
                                predictionresult = nodepair[0]
                            else:
                                predictionresult = nodepair[1]

                        else:
                                predictionresult = nodepair[0]
                        #print('Node result(+ve)',vector,predictionresult)
                        break
        K = N
    else:#IF THERE ARE NO CELL PREDICTORS
        #GET NODE PREDICTORS
                NodePredictors = NodePredictorList[subtreeno]
                X = len(NodePredictors)
                nodepredictorskeys = []
                nodepredictor = {}
                i=0
                while i<X:
                    predictor = NodePredictors[i]
```

```
                for Keys, nodes in predictor.items():
                    predictorkey = Keys
                    nodepredictorskeys.append(predictorkey)
                    nodepredictor[predictorkey] = []
                    nodepredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            nodepredictorskeys.sort()
            currentcell = nodepredictorskeys[0]
            #print('currentcell',currentcell)
            nodes = nodepredictor[currentcell]
            #print('nodes',nodes)
            node = nodes[0]
            nodepair = node[0]
            if len(nodepair)==2:#CHECK IF THERE ARE TWO NODES IN A CELL
                nodepredictor = node[1]
                accuracy = node[2]
                result3 =  nB1.predict(nodepredictor,vector)#PREDICT ONE OF THE NODES
                #print('Node result',result3[0])
                if (result3[0] == 1.0):
                    predictionresult = nodepair[0]
                else:
                    predictionresult = nodepair[1]
                K = N
                break

            else:#IF THERE IS ONLY ONR NODE IN A CELL
                    predictionresult = nodepair[0]
                    K = N

else:#IF CURRENT SUBTREE NOT PREDICTED
        if (K == N-1):#CHECK IF ONLY ONE SUBTREE REMAINING
            subtreeno = Key[K]
            TreePredictorList = TreePredictorTree[subtreeno]
            TreePredictorTr  = TreePredictorList[0]
            TreePredictor  = TreePredictorTr[0]
            CellPredictorList = TreePredictorTr[1]
            NodePredictorList  = TreePredictorTr[2]
            CellPredictors = CellPredictorList[subtreeno]
            X = len(CellPredictors)
            cellpredictorskeys = []
            cellpredictor = {}
            i=0
            while i<X:
                predictor = CellPredictors[i]
                for Keys, cells in predictor.items():
                    predictorkey = Keys
                    cellpredictorskeys.append(predictorkey)
                    cellpredictor[predictorkey] = []
                    cellpredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            cellpredictorskeys.sort()
            count=X
            for Keys, cells in cellpredictor.items():
              count=count-1
              #print('if current subtree not predicted,Cell:',cells)
              cell = cells[0]
              currentcell = cell[0]
              othercells = cell[1]
              cellpredictor = cell[2]
              accuracy2 = cell[3]
              result2 =  nB1.predict(cellpredictor,vector)
              #print('if current subtree not predicted,Cell result2',result2[0])
              if (result2[0] == 1.0): #IF CELL PREDICTION IS TRUE
                 #GET NODE PREDICTOR
                NodePredictors = NodePredictorList[subtreeno]
                X = len(NodePredictors)
                nodepredictorskeys = []
                nodepredictor = {}
                i=0
                while i<X:
                    predictor = NodePredictors[i]
                    for Keys, nodes in predictor.items():
                        predictorkey = Keys
                        nodepredictorskeys.append(predictorkey)
                        nodepredictor[predictorkey] = []
                        nodepredictor[predictorkey].append(predictor[predictorkey])
                    i=i+1
                nodepredictorskeys.sort()
                nodes = nodepredictor[currentcell[0]]
                #print('if current subtree not predicted,nodes',nodes)
                node = nodes[0]
                nodepair = node[0]
                if len(nodepair)==2:
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  nB1.predict(nodepredictor,vector)
                    #print('if current subtree not predicted,Node result',result3[0])
```

```
                        if (result3[0] == 1.0):
                            predictionresult = nodepair[0]
                        else:
                            predictionresult = nodepair[1]
                    else:
                            predictionresult = nodepair[0]
                    #print('if current subtree not predicted,Node result(+ve)',vector,predictionresult)
                    break
                else:#IF CELL PREDICTION IS FALSE
                    if (count==0):
                        if len(othercells)==1:
                          predictionresult = othercells[0]
                        #print('if current subtree not predicted,Node result(-ve)',vector,predictionresult)
                        break
            K = N
        else:
            K = K + 1

    #print('Prediction result for this vector:',vector,predictionresult)
    y = float(predictionresult)
    predictiondata.append(y)
    #predictiondata[i][-1] = float(predictionresult)
    #print(vector[-1],'...',y)

    if vector[-1] == y:
        if (vector[-1] in correctClassified):
            #print('count before is:',correctClassified[vector[-1]])
            count=correctClassified[vector[-1]]
            correctClassified[vector[-1]] = count+1
            #print('count after is:',correctClassified[vector[-1]])
            #print('Yes1')

        else:
            correctClassified[vector[-1]] = 1
            #print('Yes2')
    else:
        if (vector[-1] in incorrectClassified):
            count=incorrectClassified[vector[-1]]
            incorrectClassified[vector[-1]] = count+1
            #print('No1')
        else:
            incorrectClassified[vector[-1]] = 1
            #print('No2')


#print('correctClassified:',correctClassified)
#print('incorrectClassified:',incorrectClassified)
testFileDistribution = nB1.getClassDistribution(testdata)
classAccuracy = nB1.getClassAccuracy(testFileDistribution,correctClassified,incorrectClassified)
#print('classAccuracy is:',classAccuracy)
#print('len(testdata),len(predictiondata):',len(testdata),len(predictiondata))
overallaccuracy = nB1.getAccuracy(testdata,predictiondata)
#print('THE ACCURACY FOR THIS CLASSIFICATION IS=%:',nB1.getAccuracy(testdata,predictiondata))
return  overallaccuracy


#THIS CLASSIFIES ONE INSTANCE AT A TIME USING A STORED TRAINED CLASSIFIER LOADED FROM PICKLE
def classifyOneInstance(self,classifier,classTree,data):

    self.classTree = classTree
    self.data = data

    tree = classTree
    testdata = data
    '''
    #RETRIEVE THE CLASSIFIER
    pickle_in = open('C:\Program Files (x86)\WinPython-64bit-3.4.3.5\PythonEditor\PYPE-2.9.4\EXPERIMENTDATA\PROTEIN\BNclassifier.pickle','rb')
    tp=pickle.load(pickle_in)

    TreePredictors = tp
    '''
    TreePredictors = classifier

    mKey =list(TreePredictors.keys())
    CellPredictorTree = {}
    NodePredictorTree = {}
    #TreePredictorTre = {}
    #predictiondata = data
    predictiondata = []

    nB1 = naiveRootclassifier()
    vector = testdata
    T = 0
    while (T <len(mKey)):#CHECK IN EACH MAIN TREE IN WHICH THE INSTANCE BELONGS
        treeno = mKey[T]#GET CLASSIFIER NUMBER
        mainTreePredictorList = TreePredictors[treeno]
```

```
        mainTreePredictor =  mainTreePredictorList[0]

        #print('mainTreePredictor[0]', mainTreePredictor[0])
        mainPredictors = mainTreePredictor[0][0.0]
        if (len(mainPredictors)> 1):#CASE OF MORE THAN ONE TREE
                mainPredictor = mainPredictors[0]
                mainTreeResult =  nB1.predict(mainPredictor,vector)#MAKE PREDICTION
                if (mainTreeResult[0] ==1.0)and (T <=(len(mKey)- 2)):#IF 1 GET SUBTREE CLASSIFIER
                    TreePredictorTree = mainTreePredictor[1]
                    #print('FOR MAINTREE NO:', treeno)




                    T = len(mKey)+1#END THE LOOP
                else:#IF -1
                    if (mainTreeResult[0] == -1.0)and (T >= (len(mKey) - 2)):#CHECK WHETHER IT IS SECOND LAST
                      treeno = mKey[T+1]#GET GET THE ONLY LAST AND END THEN LOOP
                      mainTreePredictorList = TreePredictors[treeno]
                      mainTreePredictor =  mainTreePredictorList[0]
                      #print('mainTreePredictor[0]', mainTreePredictor[0])
                      TreePredictorTree = mainTreePredictor[1]
                      #print('(this is second last)FOR MAINTREE NO:', treeno)




                      T = len(mKey)+1 #END THE LOOP
                    else:#IF NOT SECOND LAST (mainTreeResult[0] == -1.0)and (T < len(mKey) - 2)
                      T = T + 1 #LOOP AGAIN
        else:#CASE OF ONLY ONE TREE
                TreePredictorTree = mainTreePredictor[1]
                T = len(mKey)+1 #END THE LOOP

    Key = list(TreePredictorTree.keys())
    N = len(Key)
    #print('TreePredictorTree', TreePredictorTree)
    K = 0
    while (K < len(Key)):#CHECK IN EACH SUBTREE THE CELL IN WHICH THE INSTANCE BELONGS
        subtreeno = Key[K]
        TreePredictorList = TreePredictorTree[subtreeno]#TreePredictorList IS A LIST OF ONLY ONE ELEMENT I.E. THIS SUBTREE
        TreePredictorTr  = TreePredictorList[0]#TreePredictorTr IS A LIST OF THREE DICTIONARIES OF THIS SUBTREE PREDICTORS I.E.[{SUBTREE},{CELLS},{NODES}]
        TreePredictor  = TreePredictorTr[0]# TreePredictor IS A DICTIONARY OF THIS SUBTREE PREDICTOR
        CellPredictorList = TreePredictorTr[1] #CellPredictorList IS A DICTIONARY OF THIS SUBTREE CELL PREDICTORS
        NodePredictorList  = TreePredictorTr[2]#NodePredictorList IS A DICTIONARY OF THIS SUBTREE NODE PREDICTORS
        CellPredictors = CellPredictorList[subtreeno]#CellPredictors IS A LIST OF THIS SUBTREE'S CELL PREDICTORS
        Trpredictor = TreePredictor[0.0]#Trpredictor IS A PREDICTOR OF THIS CURRENT SUBTREE
        result1 = nB1.predict(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE
        #print('Trpredictor[0]',Trpredictor[0])
        #result1 = nB1.getPredictions(Trpredictor[0],vector)#THIS IS PREDICTING THE CURRENT SUBTREE

        #print('subtree result1',result1)
        if (result1[0] == 1.0):#IF CURRENT SUBTREE PREDICTED YES
          #GET CELL PREDICTORS
          X = len(CellPredictors)
          #print('CellPredictor:for result1=1',CellPredictors)
          if (X>0):#IF THERE ARE CELL PREDICTORS
              cellpredictorskeys = []
              cellpredictor = {}
              i=0
              while i<X:#WHILE THERE ARE CELL PREDICTORS
                predictor = CellPredictors[i]
                for Keys, cells in predictor.items():
                  predictorkey = Keys
                  cellpredictorskeys.append(predictorkey)
                  cellpredictor[predictorkey] = []
                  cellpredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
              cellpredictorskeys.sort()#SORT THEM IN THE ORDER THEY WILL BE WORKED ON
              count=X
              for Keys, cells in cellpredictor.items():
                count=count-1 #COUNT CELL PREDICTORS BOTTOM UP
                #print('Cell:',cells)
                cell = cells[0]
                currentcell = cell[0]
                othercells = cell[1]
                cellpredictor = cell[2]
                accuracy2 = cell[3]
                result2 =  nB1.predict(cellpredictor,vector)
                #print('Cell result2',result2[0])
                if (result2[0] == 1.0):  #IF CELL RESULT IS 1 SELECT THE FIRST CELL'S NODE PREDICTORS
                  #GET NODE PREDICTOR FOR THIS SUBTREE
                  NodePredictors = NodePredictorList[subtreeno]
                  #print('NodePredictors',NodePredictors)
                  X = len(NodePredictors)
                  nodepredictorskeys = []
                  nodepredictor = {}
```

```
                i=0
                while i<X:
                    predictor = NodePredictors[i]
                    for Keys, nodes in predictor.items():
                        predictorkey = Keys
                        nodepredictorskeys.append(predictorkey)
                        nodepredictor[predictorkey] = []
                        nodepredictor[predictorkey].append(predictor[predictorkey])
                    i=i+1
                nodepredictorskeys.sort()
                #print('nodepredictor',nodepredictor)
                nodes = nodepredictor[currentcell[0]]
                #print('nodes',nodes)
                node = nodes[0]
                nodepair = node[0]
                if len(nodepair)==2:
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  nB1.predict(nodepredictor,vector)
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]

                else:
                        predictionresult = nodepair[0]
                #print('Node result(+ve)',vector,predictionresult)
                break
            else:#IF CELL RESULT IS -1 SELECT THE OTHER CELL'S NODES
                if (count==0):#IF THIS IS THE LAST CELL PREDICTOR FOR THIS SUBTREE
                    if len(othercells)==1:#IF THERE IS ONLY ONE NODE IN THIS CELL
                        predictionresult = othercells[0]

                    else:#IF THERE IS MORE THAN ONE(TWO) NODES IN THIS CELL
                        NodePredictors = NodePredictorList[subtreeno]
                        X = len(NodePredictors)
                        nodepredictorskeys = []
                        nodepredictor = {}
                        i=0
                        while i<X:
                            predictor = NodePredictors[i]
                            for Keys, nodes in predictor.items():
                                predictorkey = Keys
                                nodepredictorskeys.append(predictorkey)
                                nodepredictor[predictorkey] = []
                                nodepredictor[predictorkey].append(predictor[predictorkey])
                            i=i+1
                        nodepredictorskeys.sort()
                        nodes = nodepredictor[othercells[0]]
                        #print('nodes',nodes)
                        node = nodes[0]
                        nodepair = node[0]
                        if len(nodepair)==2:
                            nodepredictor = node[1]
                            accuracy = node[2]
                            result3 =  nB1.predict(nodepredictor,vector)
                            #print('Node result',result3[0])
                            if (result3[0] == 1.0):
                                predictionresult = nodepair[0]
                            else:
                                predictionresult = nodepair[1]

                        else:
                                predictionresult = nodepair[0]
                        #print('Node result(+ve)',vector,predictionresult)
                        break
        K = N
    else:#IF THERE ARE NO CELL PREDICTORS
        #GET NODE PREDICTORS
            NodePredictors = NodePredictorList[subtreeno]
            X = len(NodePredictors)
            nodepredictorskeys = []
            nodepredictor = {}
            i=0
            while i<X:
                predictor = NodePredictors[i]
                for Keys, nodes in predictor.items():
                    predictorkey = Keys
                    nodepredictorskeys.append(predictorkey)
                    nodepredictor[predictorkey] = []
                    nodepredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            nodepredictorskeys.sort()
            currentcell = nodepredictorskeys[0]
            #print('currentcell',currentcell)
            nodes = nodepredictor[currentcell]
```

```
                #print('nodes',nodes)
                node = nodes[0]
                nodepair = node[0]
                if len(nodepair)==2:#CHECK IF THERE ARE TWO NODES IN A CELL
                    nodepredictor = node[1]
                    accuracy = node[2]
                    result3 =  nB1.predict(nodepredictor,vector)#PREDICT ONE OF THE NODES
                    #print('Node result',result3[0])
                    if (result3[0] == 1.0):
                        predictionresult = nodepair[0]
                    else:
                        predictionresult = nodepair[1]
                    K = N
                    break

                else:#IF THERE IS ONLY ONR NODE IN A CELL
                        predictionresult = nodepair[0]
                        K = N

else:#IF CURRENT SUBTREE NOT PREDICTED
            if (K == N-1):#CHECK IF ONLY ONE SUBTREE REMAINING
            subtreeno = Key[K]
            TreePredictorList = TreePredictorTree[subtreeno]
            TreePredictorTr  = TreePredictorList[0]
            TreePredictor  = TreePredictorTr[0]
            CellPredictorList = TreePredictorTr[1]
            NodePredictorList  = TreePredictorTr[2]
            CellPredictors = CellPredictorList[subtreeno]
            X = len(CellPredictors)
            cellpredictorskeys = []
            cellpredictor = {}
            i=0
            while i<X:
                predictor = CellPredictors[i]
                for Keys, cells in predictor.items():
                    predictorkey = Keys
                    cellpredictorskeys.append(predictorkey)
                    cellpredictor[predictorkey] = []
                    cellpredictor[predictorkey].append(predictor[predictorkey])
                i=i+1
            cellpredictorskeys.sort()
            count=X
            for Keys, cells in cellpredictor.items():
                count=count-1
                #print('if current subtree not predicted,Cell:',cells)
                cell = cells[0]
                currentcell = cell[0]
                othercells = cell[1]
                cellpredictor = cell[2]
                accuracy2 = cell[3]
                result2 =  nB1.predict(cellpredictor,vector)
                #print('if current subtree not predicted,Cell result2',result2[0])
                if (result2[0] == 1.0): #IF CELL PREDICTION IS TRUE
                    #GET NODE PREDICTOR
                    NodePredictors = NodePredictorList[subtreeno]
                    X = len(NodePredictors)
                    nodepredictorskeys = []
                    nodepredictor = {}
                    i=0
                    while i<X:
                        predictor = NodePredictors[i]
                        for Keys, nodes in predictor.items():
                            predictorkey = Keys
                            nodepredictorskeys.append(predictorkey)
                            nodepredictor[predictorkey] = []
                            nodepredictor[predictorkey].append(predictor[predictorkey])
                        i=i+1
                    nodepredictorskeys.sort()
                    nodes = nodepredictor[currentcell[0]]
                    #print('if current subtree not predicted,nodes',nodes)
                    node = nodes[0]
                    nodepair = node[0]
                    if len(nodepair)==2:
                        nodepredictor = node[1]
                        accuracy = node[2]
                        result3 =  nB1.predict(nodepredictor,vector)
                        #print('if current subtree not predicted,Node result',result3[0])
                        if (result3[0] == 1.0):
                            predictionresult = nodepair[0]
                        else:
                            predictionresult = nodepair[1]
                    else:
                            predictionresult = nodepair[0]
                    #print('if current subtree not predicted,Node result(+ve)',vector,predictionresult)
                    break
                else:#IF CELL PREDICTION IS FALSE
                    if (count==0):
```

```
                if len(othercells)==1:
                    predictionresult = othercells[0]
                #print('if current subtree not predicted,Node result(-ve)',vector,predictionresult)
                break
        K = N
    else:
        K = K + 1

#print('Prediction result for this vector:',vector,predictionresult)
y = float(predictionresult)
return  y
```

BIOGRAPHY

The author of this PhD thesis, Mr. Fullgence M. Mwakondo, is working at the Technical University of Mombasa (TUM) as an assistant lecturer in the Institute of Computing and Informatics. He has over 10 years experience in teaching at higher institutions of learning. He completed his BSc. Mathematics and Computer science at Jomo Kenyatta University of Agriculture and Technology, and MSc. (Information Technology) at Masinde Muliro University of Science and Technology. He is currently pursuing a PhD in Computer science in the school of computing & Informatics at the University of Nairobi.