Software Effort Estimation for Improved Decision Making

[1]S M Nazmuz Sakib (sakibpedia@gmail.com)

S M Nazmuz Sakib is an eLearning expert and done more than 500 MOOCs or Massive Open Online Courses and experienced as an instructor in sites like Udemy. He has completed his BSc in Business Studies from School of Business And Trade, Switzerland with CGPA 4 in the scale of 4 and 97.06% grade marks on an average. He is also a certified Google IT Support Professional, Google Data Analytics Professional and IBM Customer Engagement Specialist Professional.

ABSTRACT

Software development effort estimation is an active area in arena of software project management. Ranging from expert judgement to machine learning techniques various parametric and non-parametric methods were proposed with the aim of improving accuracy of estimation so that upcoming projects are completed within constraints of schedule and budgets.
Nowadays, software development organizations use machine learning techniques in different areas to improve decision-making process so that their performance is boosted. In this dissertation, with the goal of increasing the accuracy in effort estimates, we applied programming models in an environment of software development organizations. We collected empirical data from two organizations and constructed a consolidated data sets. The programming models applied in this study are K-Means clustering, Support Vector Machines using polynomial kernel, Random Forest, Linear Regression, K Nearest Neighbor and Neural Networks using ORANGE tool.
The obtained results demonstrate the use of data mining and machine learning techniques in general increases the accuracy of predictions with lesser error magnitude as compared to experts. Moreover, we recommend application of programming models in comparable environment of software development organizations to get reliable and more generalized predictions for decision making.

[1]Graduate of BSc in Business Studies, School of Business And Trade; Pilatusstrasse 6003, 6003 Luzern, Switzerland
[1]Student of BSc in Civil Engineering, Faculty of Science and Engineering, Sonargaon University; 147/I, Green Road, Panthapath, Dhaka
[1]Student of LLB(Hon's), Faculty of Law, Dhaka International University; House # 4, Road # 1, Block - F, Dhaka 1213
[1]Student of BSc in Physiotherapy, Faculty of Medicine, University of Dhaka; Nilkhet Rd, Dhaka 1000

Chapter 1

Introduction

Software development effort estimation is an imperative process that lies further down in the area of software project management. Inaccuracy in predictions lead a software project to failure. As one of the prime missions of software industry is to reduce the estimation error so that successful completion of project is ensured. Therefore, the area of software effort estimation is active and is considered as an important for researchers and industry. Therefore, the objective of this chapter is to present the overview of the research topic, then highlight the challenges and motivations for this study. Afterwards, we defined problem statement and highlighted the contributions for selected area of research. Then, we introduced the applied research methodology which has also been depicted in Figure. Finally, thesis structure which is also represented in Figure for better understanding and summary of this chapter is provided at the end.

1.1 Overview of Research

Software development effort estimation is considered as a basic activity underneath the wide-ranging methods of Software Project Management. During the past three decades plethora of work has been done in arena of Software development effort estimation. However, none of the prevailing practices outperform in all milieus [1]. As various studies has been steered to improve effort estimation all around the work. To the best of our knowledge research work has been ended in the region of Islamabad-Pakistan. Therefore, the ambition of this research work is to amplify accurateness in predictions of software development effort.

Accurate prediction of software development effort is very challenging. Accurate predictions are demanding to complete software project successfully [2]. Various studies have been conducted with the intention of bringing accuracy to predictions [3]. Software development organizations are worried about completing projects successfully as the urge for software projects is increasing every single day [4]. With the same intentions projects managers predict effort but most of the time it is either underestimated or over estimated. These estimations could cause stern complications [5] which are related to budget, schedule. The difficulties in meeting schedule and budget boundaries leads a project to fail badly under provided resources [6].

The overestimation of effort is considered as one of the main problems. This leads towards a compromise in developing a project with quality and ultimately, testing a product could not be performed properly. However, underestimation is another serious issue in field of estimation. It leads towards the allocation of resources. For underestimated projects more resources are allocated to project [7]. To overcomes challenges in making effort predictions, a lot of work has been done for the last three decades. A lot of researchers have conducted studies in order to bring accuracy in predictions.

Furthermore, Boehm in 1981 proposed a technique known as Expert Judgement [8]. Even now, Expert Judgment is considered as most widely used technique in software development organizations as addressed in work of Khan B et al [9] and Mallidi and Sharma [10]. In some additional studies a thumb rule was used for prediction purpose. Then some models such as PRICE, SLIM [11], COCOMO [12], Function points [13] etc. were proposed with the intent of bringing accuracy in predictions. Similarly, the application of data mining and its impact overestimation were investigated in different studies as highlighted in systematic literature review from 1990-2019 [14]. According to general perception machine learning techniques were adopted for increasing accuracy in predictions due the reason it delivers predictions after completing numerous rounds [2].

We have seen plenty of work to bring accuracy in software development effort predictions. Consequently, we noted the use of machine learning techniques such as Logistic Regression [15], Linear Regression [16], Support Vector Machine, Decision Trees [17], K-Nearest Neighbor [18], Neural Network [19], Na¨ıve Bayes [20], Fuzzy Logic [21] and many other. Then combination of two or more techniques were used to implemented such as Multi-layer Perceptron (MLP) and Genetic Algorithm (GA) [22]. Similarly, another hybrid method with combination of ensemble-based technique based on expert estimation. Another hybrid method with the combination of analogy-based estimation with Fuzzy logic [23] and many other similar methods were acknowledged from literature analysis.

However, none of the existing techniques fit in all environments. Most of the proposed solutions were tested over publicly available sets such as COCOMO [18], NASA [14], ISBSG [24], Desharnais, Maxwell and Miyazakil etc. [19]. Some of the studies investigated application of machine learning techniques over their known data sets which were collected from software development industries from different regions of the world. Furthermore, it has been released the importance of bringing accuracy of estimation in industry [25] and academic research [21]. Moreover, from the analysis of existing techniques we figured out, none of the existing technique fit in all environment. The reason behind is availability of data sets and features.

The prediction highly depends on the type of data set [26].

As we have seen the effect of applying machine learning algorithms in different environments, but none of these techniques was adopted in all different conditions. In addition, from the analysis of literature and survey steered in software industry it became prominent these algorithms were mostly analyzed over public available data sets. Now, researchers and software development industry are more active to bring accuracy in software effort estimation. On that account, the motive of this study is to boost accuracy of software development effort estimation from software industry positioned region of Islamabad-Pakistan.

This research work lies in the area of software development effort estimation. Over the last three decades ample of research works have been done in order to bring accuracy in software effort predictions. However, none of the methods outperform in all environments. Moreover, to the best of our knowledge no similar work has been conducted to improve effort predictions for the software development organizations developed in Islamabad-Pakistan. Consequently, with the aim of increasing accuracy in predictions, this study has two objectives. First, we identified the strengths and weaknesses of existing software effort estimation technique. Secondly, our leading purpose of this study is to improve the effort estimation process in software development organizations located in Islamabad-Pakistan.

To achieve the goal of bringing accuracy of predictions we applied combination of data mining and machine learning to the data available. First, we applied machine learning algorithms such as K- nearest neighbor, neural network, support vector machine, Linear regression and random forest that has extensively been used for improving predictions. Then, we replicated in work done by H karna et al [25] which uses data mining technique acknowledged as K-Means clustering earlier than application of machine learning algorithm. In the succeeding phase, we compared the accuracy of machine learning algorithms with and without use of K-Means Clustering. At the end, for a new project, we use combination of expert judgement and machine learning algorithm to predict effort of software project.

Firstly, we applied machine learning algorithms over the data set which was collected from two software development organizations sited in region of Islamabad Pakistan. We collected empirical data from a survey conducted for two software organizations. We collected data of 38 projects P1-P38. Out of which P1-P28 were employed for training and P29-P38 were used for testing algorithms. We selected these projects on the basis of project size such as Small, Medium and Large. Moreover, we considered the projects which are developed within organization. We have not considered outsourced projects. Secondly, we performed K-means clustering over data set to identify the difference of clustering over prediction. Finally, in the thirds phase we analyzed the impact of using the similar techniques for upcoming projects. We realized machine learning when applied to the data sets used in this study increases accuracy of estimation as compared to those of experts. Overall, for new projects, the models used in this study utilized the input from experts and then machine learning algorithms produce predictions of effort for software project. Thus, we conclude the use of machine learning algorithms in general increases the accuracy of effort and help project managers to allocate reasonable resources to project for successful completion within constraints of schedule and budget.

1.2 Application of Research

In the area of software development effort estimation, this study aims to improve the accuracy of effort estimation of software development projects for software organizations located in Islamabad. In direction to propose a method of estimation to organizations, this study first identifies the maximum used techniques from literature. With respect to research, this study has following applications:

1. This research is applicable in the organizations which involve software development such as software industry.
2. This research is also helpful for Business analysts, project managers and Estimators.
3. In context of research, this research is applicable in the area of effort estimation for comparison and generalization of results.
4. This research is applicable in all fields which involve projects and their effort estimation is unresolved.

1.3 Problem Statement

In Software development, accurate effort estimation is significant for project manager [18]. The problems of overestimation and underestimation could cause serious complications [15] such as schedule, budget and would ultimately lead towards project failure [27], ranging from expert estimation [8, 28] to machine learning techniques, [29, 30, 31, 32] none of these method fit in different environments [1]. Furthermore, the trend of increasing estimation accuracy within software organizations has been realized in literature [25, 33, 34]. Therefore, the motivation of this

study is to increase accuracy of estimation and provide decision support system for two software development organizations located in region of Islamabad-Pakistan. Based on problem statement, we developed following research questions:

RQ1: What are the strengths and weaknesses of existing software effort estimation techniques?

RQ2: How could Effort Predictions in Software Development Organizations be improved?

1.4 Research Motivation

Software development effort estimation is considered as a challenging process in area of software project management. As underestimation and overestimation of project are the main challenges in prediction of effort. However, if the effort of project is not predicted closer to actual effort, project manager is unable to control the flow of project for successful delivery. Furthermore, the accuracy of predicting effort is poor in software industry located in Islamabad- Pakistan. Moreover, we have not seen considerable work done in the region of Islamabad-Pakistan for improvement of predictions.

1.5 Research Objectives

The objective of this research is to deliver decision support system for software development organizations by estimating effort and assign reasonable resources to project prior to starting phase. To meet this we divided our work into following:

1. We proposed a framework for improvement of effort prediction.
2. To overcome problems enforced by inaccurate estimations.
3. To conduct an experimental study for improving up accuracy.
4. To help project manager allocate reasonable resources for upcoming projects.

1.6 Research Contributions

The primary contribution of this research work is to improve software development effort estimation for the software organizations located in region of Islamabad Pakistan. To this, first we identified the problem of effort estimation is the selected

region. Then, proposed a framework which includes designing questionnaire, data collection, data analysis and then application of machine learning and data mining. To the best of our knowledge none of the similar work has been done in the selected environment. The secondary objective of this research work is to identify strengths and weaknesses of existing techniques to analyze the impact of machine learning over effort estimation.

First of all we have critically analyzed the literature to identify the strength and weakness of the state-of-the-art effort estimations techniques and we have utilized the identified limitations to answer our research question1. Further, we have proposed a model for effort estimation that employs the effectiveness of K-mean clustering (well-known supervised learning algorithm). Proposed model is validated in the software industry of Islamabad Pakistan. By validating the proposed model we affirm that this model can be efficiently used to measure the effort of projects. Broadly, this research work has following contributions in the area of software development effort estimation.

1. Identified the strengths and weaknesses of existing techniques for effort estimation.
2. Propose a model with the combination of unsupervised and supervised learning techniques to estimate effort estimation of software projects within the organizations located in region of Islamabad- Pakistan.
3. The model predicts effort which would help the project members to take decisions regarding resource allocation to each new project for successful completion of project within budget and time.

Thus, the results express improvement in effort estimation with the combination of expert judgement and machine learning algorithms. The proposed framework could be adopted for predicting accurate estimation for software development organizations.

1.7 Significance of Research

Software development effort estimation is one of the most significant process in the field of software engineering. As inaccurate predictions in software development organizations have shaped a massive problem for software engineers and project managers to complete projects within offered budget and schedule. As the existing practices of expert judgement have not been able to produce robust predictions

in software industry. Therefore, the application of machine learning algorithms in over the data sets collected from software industry located in Islamabad-Pakistan could improve prediction accuracy.

1.8 Research Methodology

The objective of this research work is to improve effort estimation for software development organizations. To achieve the main aim, we first explored the area of software project management and highlighted the importance of effort estimation for successful delivery of project. After gaining relevant knowledge, we adopted a tool known questionnaire as done in H karna et al [25] for data collection. We collected data from two software development organizations and analysis was performed using ORANGE tool. Finally, we applied machine learning algorithms over data set to predict effort and then analyzed the results using different performance measures. The overall methodology is presented in Figure 1.1.

1.8.1 Topic Selection

In the first phase we studied software project management in details and highlighted the most important area in it. From the analysis of problems in different area of project management, we identified software effort allocation to a project is the trickiest process and is performed at the start of process. We also realized software development effort estimation is one the foremost concern of industry and researchers. To bring accuracy in predictions over the past three decades a plethora of work has been done but unfortunately, none of the method outclassed in all set-ups. After the analysis of gaps and challenges that were identified from literature, we selected our research topic.

1.8.2 Literature Review

To collect relevant data for effort estimation, we surfed different digital libraries. We collected data from Google Scholar, IEEE and Wiley were explored, and appropriate data is retrieved. The most relevant studies were studied and provided in Chapter 3 of this dissertation with details.

1.8.3 Development of Framework

Based on literature, we identified strength and weakness of existing literature. After the detailed study of literature, we developed a conceptual framework for

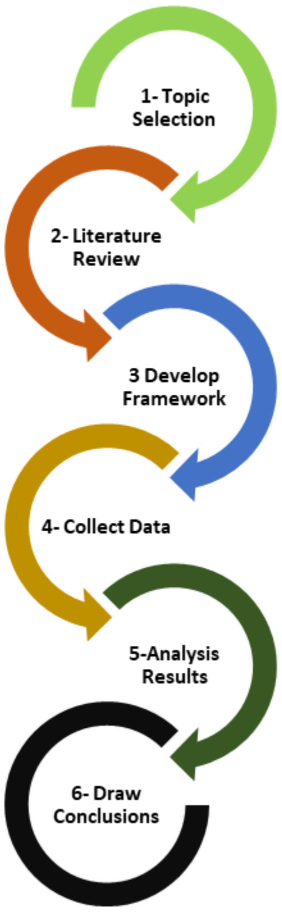


Figure 1.1: Research Methodology

effort prediction of software projects. The components of Conceptual Framework are provided in Chapter 4 of this dissertation. After attaining knowledge about the topic and highlighting the issues of previous frameworks we proposed a framework for effort estimation.

1.8.4 Data Collection

This section is related to data collection process. At this phase we selected two organizations located in Islamabad-Pakistan. We collected data of 38 already completed software projects P1-P38. We selected 28 projects for training and then 10 projects P29-P38 for testing purpose. We used a tool named as questionnaire for data collection. There are two types of variable involved in this study first one is numeric values and the second is categorical variable which could also be named as ordinal values. The details of data set is provided in chapter 4 of this description.

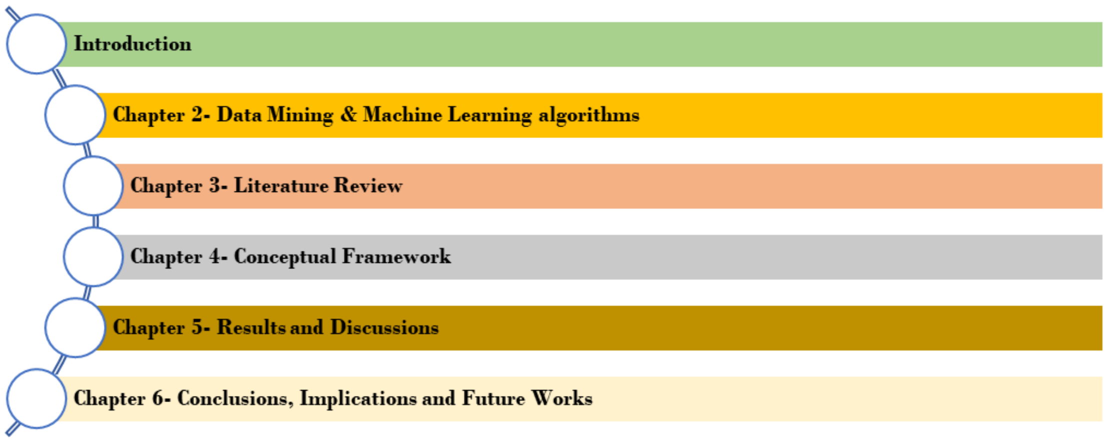

Introduction
Chapter 2- Data Mining & Machine Learning algorithms
Chapter 3- Literature Review
Chapter 4- Conceptual Framework
Chapter 5- Results and Discussions
Chapter 6- Conclusions, Implications and Future Works

Figure 1.2: Thesis Structure

1.8.5 Analysis of Result

We performed all experiments in ORANGE tool. After results were produced, we analyzed the result using different validation methods used in effort estimation. We applied CHI- square and ANOVA to understand trends of data set and Pearson correlation to find the relationship between independent and target variable.

1.8.6 Drawing Conclusions

At the end, conclusions were drawn after performing analysis of formed results. To reach a single conclusion, we performed experiments in two main steps. First one comprises of application of machine learning and in step we applied clustering before applying machine learning. Furthermore, we have analyzed the results using different evaluation measures to come up with one solution.

1.9 Thesis Structure

The remaining sections of his dissertation is ordered in following style: Machine Learning algorithm have been presented in Chapter 2. Chapter 3 constitutes of Literature Review. Conceptual framework and results & discussions are part of Chapter 4 and Chapter 5. Finally, in Chapter 6, we provided Conclusions and Future works. The structure of this dissertation is provided in Figure 1.2.

1.9.1 Chapter 2- Data Mining & Machine Learning Algorithms

This chapter presents the details of data mining techniques and machine learning algorithms that were used in this study for the prediction of effort.

1.9.2 Chapter 3- Literature Review

This chapter provides the literature review in two section. First section provides the application of data mining in the field of software engineering. Then, in second section, we explain data mining and machine learning when applied in the area of software development effort estimation.

1.9.3 Chapter 4-Conceptual Framework

This chapter defines the proposed conceptual framework and explain each component of this framework in detail.

1.9.4 Chapter 5- Results and Discussions

This chapter provides in depth interpretation of the results. The validation measures and provides the discussions.

1.9.5 Chapter 6- Conclusions & Future Work

This is the last chapter of this dissertation which provides the conclusion and guide the future perspective to the attentive researchers in area of software development effort estimation

Chapter 2 Preliminary Studies

Chapter 1 has provided the introduction of research topic. The objective of this chapter is to provide in depth view of the algorithms used in this study. The used algorithms are K-Means clustering, Support Vector Machine, K-Nearest Neighbor, Random Forest, Neural network and Linear regression. We also described Expert Judgement and the evaluation measures used in this study.

2.1 Software Development Effort Estimation

Software Development Effort Estimation is a process of predicting work hours required to complete software project successfully. It is usually expressed in person month, man-hour or work-hours. As we have seen in literature, software estimation techniques such as thump rule, expert based judgement, checkpoints, seer, Price-S, Function points, COCOMO methods [35], Analogy based estimation, top down, bottom up approaches, price to win were proposed for estimation. But, accuracy of predictions seemed to decline with increasing size and complexity of software projects. Due to this problem of parametric methods researchers have shifted towards application of machine learning for estimation as we have in [36]. Further, machine learning is applied for situations where we have increased size of data set and we want to improve the performance [37]. Thus, with aim of increasing accuracy of estimates random forest trees were applied for bring up accuracy in predictions. In addition, use of case-based reasoning [38], and combination of functional point with neural network, CBR and regression is implemented in [39]. The combination of regression with analogy based estimation is noted in [40] with improved estimates. The application of machine learning has not stopped here; more researchers have applied different machine learning algorithms.

Major type of machine learning techniques for effort estimation involve use of concept learning (CL), artificial intelligence (AI), decision trees (DT), artificial neural networks (ANN), instance based learning (IBL) and analytical learning [41]. So, the use of more techniques such as neural network has been investigated and proved to be feasible technique for bottom up data [42]. With the increased number of experiments for applying data mining and machine learning has continued with investigation of different techniques repeatedly in different environment such as application of case based reasoning, rule induction and artificial neural networks for Canadian dataset is reported in [43]. We again noticed an experiment using genetic algorithm over Desharnais dataset which is tested for software development effort estimation [44].

Moreover, the effectiveness of transfer learning has been investigated for environment of Tukutuku datasets [45]. Similarly, with many other methods, ensemble based effort estimation is another machine learning based technique for estimation purpose. This effectiveness of this technique was investigated in [46, 47]. Similarly, like many other researchers' fuzzy logic also considered effective in combination with analogy based approach [48]. In addition, from the systematic literature review, it was noted that 8 types of machine learning techniques were under investigation till year 2010 [49].

Even today the process of effort estimation has not been generalized for all types of software development projects and researchers are keenly interested in finding one solution. Therefore, as seen in [50] where use of gradient boosting and deep learning is analyzed. Similarly another method known as Neural network is tested proves to be a better model for estimation [2]. Moreover in a systematic literature review till year 2018, we realized the importance and applicability of machine learning algorithms could help in improving effort estimation of software projects
[24].

From the overview presented above, we concluded numerous techniques have been proposed such as Expert judgement, parametric models and then era of machine learning for effort estimation had been started long time ago. With the use of data mining we have seen application of machine learning has widely been used to bring accuracy in estimation. But still, none of these work in all environments. Thus, extending the research in arena of software development effort estimation, this study has applied data mining and machine learning algorithms over the data set collected for organizations located in Islamabad-Pakistan.

To meet the objectives of this research work, we applied several machine learning and data mining techniques. In this chapter, first we explained Expert Judgement Estimation and then we provided in depth view of all the selected methods such as supervised and unsupervised learning algorithms. At the end we have provided details of evaluation measures which are used for validation of research.

2.1.1 Effort Estimation using Expert Judgement and Machine Learning

To apply combination of expert judgement and machine learning, this section helps us in gaining deep understanding of expert judgement and then machine learning.

2.1.1.1 Expert Judgement

Boehm in 1981 proposed a method known as expert judgement for software development effort estimation [8]. Expert judgement is a quantification step and result in effort estimates [51]. The meaning of quantification step is to provide a numeric value in terms of hours/ days or months. This number is then used to allocate resources to individual.

Expert judgement has two categories: structured [52] and unstructured. The unstructured estimation technique is not reliable and yields inaccurate results [53]. Unstructured effort estimation is purely based on intuition, knowledge of estimator. Unstructured estimation is not supported by researchers for reason of high inaccurate estimations. Furthermore, the structured expert-based judgement is preferable as it is mixture of multiple methods such as using checklists for making estimates. Moreover, work break down structure and Delphi method are used by experts to produce estimates without using any other parametric model [54].

Usually expert judgement is based on historical data [55]. It also depends on motivation, experience, knowledge [9]. Most of the time effort of software projects is estimated by using this technique. There is no better evidence that these techniques perform work well [56, 57]. Different machine learning techniques from expert judgement to artificial intelligence techniques, the estimation accuracy remains different in all conditions [58]. However, expert judgement remains highly inconsistent [59]. The degree of error in expert judgement is conscious and unconsciousness [60]. Due to the above mentioned reasons, in this study expert judgement is used with combination of machine learning techniques.

2.1.1.2 Machine Learning Algorithms & Data Mining Techniques

In this Section, we explained in detail the machine learning, the types and techniques which are used for the estimation of effort.

2.1.1.3 What is machine learning?

Starting from learning, which has been defined in dictionary as: *“to gain knowledge, or understanding of, or skill in, by study, instruction, or experience,”* and *“modification of a behavioral tendency by experience”* [55]. According to Tom Mitchell, machine learning is defined as *”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”* [56].

The statistical learning has applications in different arena such as finance, science and industry [57]. Some very basic example of statistical learning are as follows:

- Identify risk factors in cancer on the basis of demographic information.
- Prediction of price in stock market.
- Estimation related to amount of glucose etc.

Furthermore, the use of machine learning for effort estimation has been used for prediction of effort estimation as seen in [58] where fuzzy logic was implemented to predict effort based on factors such as complexity, size and developer characteristics. In addition, from [59] we have noted the work is done using machine learning to increase the precision and reliability using confidence intervals. However, estimating software project with respect to schedule and budget using machine learning techniques such as decision trees, support vectors, radial basis function and principle component analysis is studied in [35]. Even today, machine learning algorithms are applied to improve effort estimation in different environments such as Random forest is applied to increase accuracy of COCOMO II in [36].

2.1.1.4 Why do we use machine learning?

Machine Learning applied over two conditions. First, when there are multiple features, and all are of equal importance. Second, the size of available data is large, and human cannot analyses the trends and patterns of data set. We can also say machine learning solves complex problems by increasing learning experience and adaptability of features [60]. Therefore, for prediction over such conditions, machine learning algorithms are applied to get better and suitable results.

2.1.1.5 What are the types of machine learning algorithms?

Learning is extensive domain. When combined with machine learning, it is divided into multiple sub fields to solve wide range of problems. The learning paradigm is classified into four parameters [60].

1. Supervised and unsupervised
2. Active and passive learners
3. Helpfulness of the teacher
4. Online and batch learning protocols

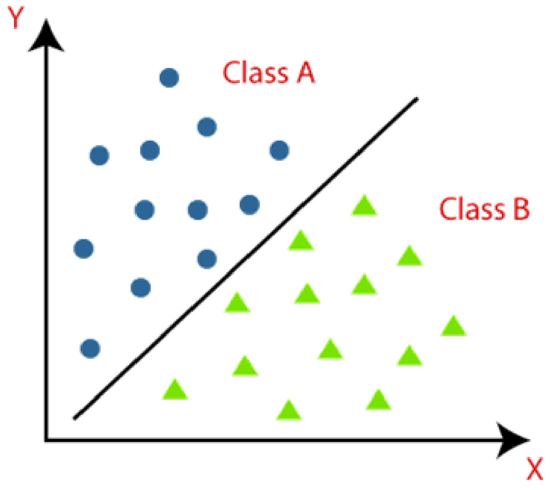


Figure 2.1: Classification

2.1.1.6 Supervised Learning and Unsupervised Learning Techniques

In this dissertation, we are concerned about supervised and unsupervised learning. Rest of the learning methods are out of the scope for this research work. As, we have defined above learning depends on the environment and the learners. Thus, in order to understand the difference between both, let's first take an example of spam e-mail detection and anomaly detection. For the first example, we have labelled data set with two labels spam and not spam. The learners are trained on this data set. However, for anomaly detection, all the learners are trained with the data set with no labels. The task of learners is to detect the unusual messages. As we know learning is process of gaining experience to increase expertise. For unseen training example, supervised learning with identify the missing information in new examples and classify it. In above example, the classification in terms of spam and not spam. In contrast to supervised learning, unsupervised learning is trains classifier using labelled data and forms suitable clusters based on characteristics.

The two main concepts for supervised learning to gain experience are classification and regression [60]. alternatively, unsupervised learning technique are based on relationship between data variables [56]. There are multiple algorithms, however in this section we would explain the algorithms which are most widely used in area of effort estimation and research community supports use of such algorithms.

2.1.1.7 Classification

Classification problem in machine learning is based on discrete valued output i.e. either 0 or 1. [61, 62] The classification of two groups is presented in Figure 2.1.

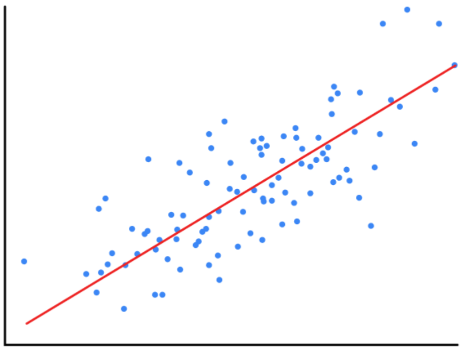


Figure 2.2: Regression

2.1.1.8 Regression

The most used tool to understand the relationship of variable of dependent (Y) and independent variables (x1......xn) which are part of any system [63]. The aim of this analysis is to find the function which finds of the target variable by considering input variables. In regression we produce a continuous valued output [64]. Moreover, this function should always have the minimum possible error for input variables. However, to minimize the error, another parameter is used to find the difference between the predicted and actual values. Furthermore, the sum of ı can be reduced by using least square method with the objective of find best possible function [65]. The concept of simple regression is presented in Figure 2.2.

In the next sub-sections, we have explained supervised learning techniques such as Support Vector Machine, Linear Regression, Random Forest, K nearest neighbor and Neural Networks. Afterwards, we have explained unsupervised learning techniques such as clustering. The clustering technique that was employed in this research work is known as K-means clustering.

2.1.2 Support Vector Machine

Vapnik proposed support vector machine algorithm [72]. This approach is revolves around training data. First it considers complete data set and then smaller subsets are considered in training models. Furthermore, SVM could be adopted for both regression and classification problem solving [63, 73]. Support Vector machine is implemented in conditions when we have nonlinear data (see Figure 2.3, adopted from [74]).

Therefore, SVM forms nonlinear optimization function to generate a convex function to reach global optimum for solving any problem. Moreover, instead of providing probabilistic solutions to any problems, this algorithm results in forming

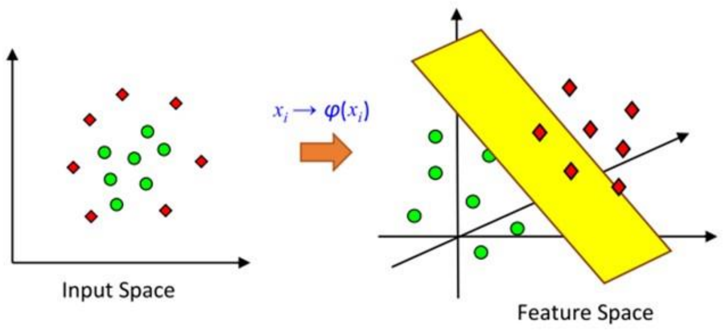


Figure 2.3: Non-Linear Data points [74]

some decision. Instead of using the optimal boundary SVM defines hyper planes. In SVM we select decision boundary when margin is maximized [73].The concept of using margins was first introduced by Tong and Koller [74]. The margin is stated as the perpendicular distance between data points which is closet to decision boundary. Whenever the margin is maximized, the decision boundary is located by support vectors. These support vectors are the closet subsets of data points. The Figure 2.4 is adopted from [73]. It explains the margin and maximized margin with circles denoting data points in a subset which are also known as support vectors.

2.1.2.1 SVM with linearly separable data

When using SVM for classification, there are two to do so. First one described in this section is two class classification. The linear model used in two class classification is expressed in Equation.

$$y(x) = w^t\theta(x) + b \tag{2.1}$$

There are N values in training set from x1......xn with target values t1......tn where tn belongs to -1,1. The data points of training set are classified with respect to function y(x). Assuming that the feature space is linearly separable and only option for b and w exists if the function is in the form of Equation 2.1.

This equation should satisfy following conditions (referred to equation 2.2, 2.3)

$$y(x_n) < 0 for t_n = +1 \& y(x_n) > 0 \tag{2.2}$$

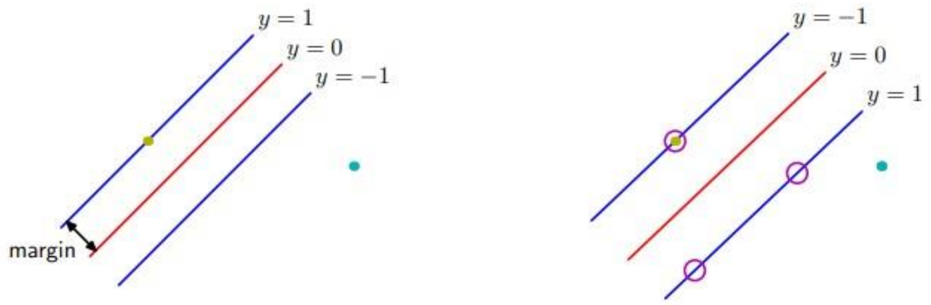


Figure 2.4: SVM: Margin [73]

$$y(x_n) < 0 for t_n = +1 \& y(x_n) > 0 \tag{2.3}$$

However, for all training examples which are also known as data point following equation 2.4 should be followed.

$$t_n = -1, t_n y(x_n) > 0 \tag{2.4}$$

As described above, SVM finds the maximum margin boundary for decision boundary. The maximum margin is can be a motivation for computational learning theory also named as statistical learning. Furthermore, to find the best suitable hyperplane, distance should be calculated for maximized [69] with the use of following Equation 2.5 given below:

$$d(w,b;x) = \frac{|(w^t x + b - 1) - (w^t x + b + 1)|}{||w||} \tag{2.5}$$

The meaning of maximizing margin is to minimize the vector w which is multidimensional. This can also be inscribed as equation 2.6 below:

$$Min_{w,b} = \frac{1}{2} w^t w \tag{2.6}$$

This is according to the given constraints between the margin of two classes. To ensure the constraints, we can a Lagrange multiplier(α) See equation 2.7 below

$$L_p(w,b,\alpha) = \frac{1}{2} w^t w - \sum_{i=1}^{N} \alpha \{y_i [w^t x_i + b] - 1\} \tag{2.7}$$

Therefore, to find the point at which slope is equal to zero, which is also known as saddle point the following equations 2.8 & **??** should be used.

$$\frac{\partial L}{\partial w} = 0 \rightarrow w_0 = \sum_{i=1}^{N} \alpha_i x_i y_i \tag{2.8}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.9}$$

In case of a corresponding input data (x_i) is a support vector, the (α) is not equal to zero [69, 75]. These support vectors are used to define the boundary of class.
Adding the values from equation 2.8 and 2.9 into 2.7, we get the following equation 2.10 which is subjected to following limitations.

$$MaxL_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j=1}^{N} [y_i y_j] \alpha_i \alpha_j x_i^T x_j \tag{2.10}$$

with respect to equation 2.11

$$\alpha_i \geq 0 \& \sum_{i=1}^{N} [\alpha_i y_i] = 0 \tag{2.11}$$

This equation is used to find the vectors and their input data. The decision function also known as hyper plan w can be calculated from this equation. Another parameter b which is known as bias can be calculated from equation 2.12 below.

$$b_0 = \frac{1}{N} \sum_{S=1}^{N} (y_s - w^T x_s) \tag{2.12}$$

2.1.2.2 Two class Classification with linearly Non- separable data

There are conditions when have linear data which cannot be separated due to feature similarity within data set [69]. in this situation linear function could not perform well. Therefore, the distance $_i$ should be calculated. This is the distance between the margin and bad classified data. The penalty function can be written as (referred to equation 2.13):

$$F(\epsilon) = \sum_{i=1}^{N} \epsilon_i \tag{2.13}$$

As a result, the nonlinear function from equation 2.6 will be as following Equation 2.14.

$$Min_{w,b} = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \epsilon_i \tag{2.14}$$

w.r.t equation 2.15

$$y_i(w^T w + b) \geq= 1 - \epsilon_i \tag{2.15}$$

In the above equation the parameter C is used to minimize error in classification and maximize the margin. This is also known as “trade-off” parameter [76]. The equation for soft margin w.r.t constrains is written as equation 2.16:

$$MaxL_d\alpha = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j=1}^{N} [y_i y_j] \alpha_i \alpha_j x_i^T x_j \tag{2.16}$$

w.r.t constraints given in equation 2.17

$$0 \leq \alpha_i \leq C; \sum_{i=1}^{N} [\alpha_i y_i] = 0 \tag{2.17}$$

where α as constraint tries to adjust its value equal to or it should be less than parameter c.

2.1.2.3 Kernel used in SVM

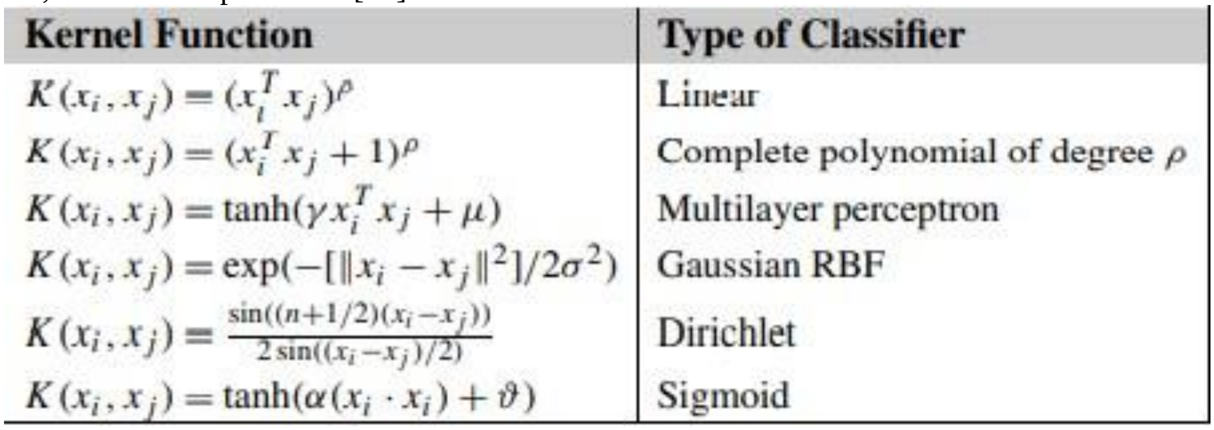
Even when the best hyperplane is found, SVM for nonlinear data would not work well. Therefore, in order to increase generalization of model, input data and its mapping with high dimensional dot product is calculated. This is known as Hilbert space [77]. The concept of nonlinear data is presented in Figure 2.3. The inner product after the selection of kernel can be calculated as equation 2.18.

$$\theta(x_i,x_j) = K(x_i,x_j) \tag{2.18}$$

Thus, to solve this equation for nonlinear data following equation 2.19 defines constraints for kernel function g(x) must be satisfied.

$$\int K(x_i,x_j) g(x_i) g(x_j) dx_i x_j \geq 0 \tag{2.19}$$

Furthermore, the type and equations for different kernels are presented in Figure 2.5, which is adopted from [69].



Kernel Function	Type of Classifier
$K(x_i,x_j) = (x_i^T x_j)^\rho$	Linear
$K(x_i,x_j) = (x_i^T x_j + 1)^\rho$	Complete polynomial of degree ρ
$K(x_i,x_j) = \tanh(\gamma x_i^T x_j + \mu)$	Multilayer perceptron
$K(x_i,x_j) = \exp(-[\lVert x_i - x_j \rVert^2]/2\sigma^2)$	Gaussian RBF
$K(x_i,x_j) = \frac{\sin((n+1/2)(x_i - x_j))}{2\sin((x_i - x_j)/2)}$	Dirichlet
$K(x_i,x_j) = \tanh(\alpha(x_i \cdot x_i) + \vartheta)$	Sigmoid

Figure 2.5: Kernel Function, [69]

With respect to kernel the equation of hyper plane is given in Equation [**?**].

$$d(x) = \sum_{i=1}^{N} [y_i \alpha_i K(x,x_i) + b] \quad (2.20)$$

2.1.2.4 SVM with Linear Regression

Support vector machine also work as a regressive model. In case of Linear decision function, the equation 2.21 if given below:

$$f(x) = w^T x + b \quad (2.21)$$

in the above equation x is the vector which is used to predict the target variable Y with use of n-dimensional feature space with weight w and the bias parameter b. As we have already described classical regression in above sections. The difference of applying SVM with regression lies in the decision function where another parameter using this parameter is presented in Figure 2.7 which is adopted from [69]. This shows SVM when implemented for regression avoids or ignores the sensitivity parameter(reference to Figure 2.6) and make use of slack term ξ to find best hyperplane [73, 78]. Therefore, the objective function L_p is given in equation 2.22 & 2.23. The purpose of this equation is to find best weights and reduce the risks.

$$L_p = \frac{1}{2}||W||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i') \quad (2.22) \text{ w.r.t}$$

$$y_i - w^T x - b \leq \xi_i + \epsilon, y_i + w^T x + b \leq \xi_i + \epsilon, \xi_i, x_i \geq 0 \quad (2.23)$$

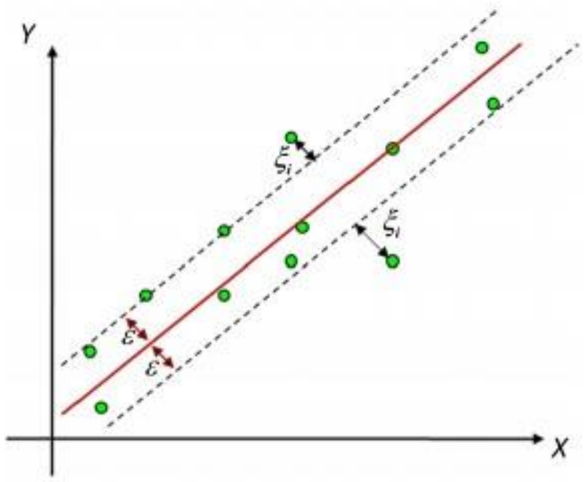


Figure 2.6: insensitivity parameter and slack variables, [70]

The restrictions in above equation show that is any error is less than is out of objective function. The concept explained above is the insensitivity theory described by Vapnik [69, 72]. Furthermore, just like classification, for solving

0 optimization problems the Langrage multipliers (α_i, α_i) are used to follow the similar conditions given in equation 2.24 & 2.25:

$$\frac{\partial l}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N}(\alpha_i - \alpha_i')X_1 \quad (2.24)$$

$$\frac{\partial l}{\partial b} = 0 \rightarrow \sum_{i=1}^{N}(\alpha_i - \alpha_i') = 0 \quad (2.25)$$

Thus, from the above equations, the equation for SVR is presented in Equation 2.26.

$$L_d = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^{'})X_i^T X_j(\alpha_i - \alpha_i^{'}) + \sum_{i=1}^{N}((\alpha_i - \alpha_i^{'})y_i - (\alpha_i - \alpha_i^{'})\epsilon) \quad (2.26)$$

w.r.t equation 2.27 given below

$$\geq 0(\alpha_i - \alpha_i^{'}) \leq C \quad (2.27)$$

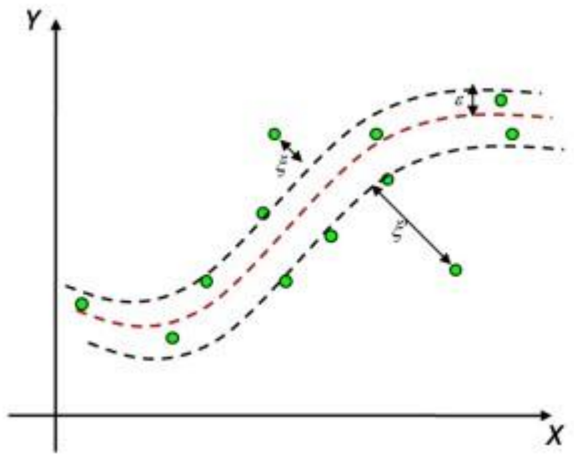


Figure 2.7: SVM for Non-Linear Data, [70]

However, when there are nonzero langrage multipliers, the bias parameter and weighing parameter were determined by one of the following equations 2.28.

$$-y_i + w^T x_i + b + \epsilon = 0; y_i - w^T x_i - b + \epsilon = 0; (\alpha_i, \alpha_i^{'}) < C \quad (2.28)$$

2.1.2.5 SVM with Nonlinear Regression

To run SVM for nonlinear data, the basic concept remains the same as described for linear data. However, the high dimensional mapping in Hilbert space has to be fixed. The insensitive margins for nonlinear data are presented in Figure 2.9.

Thus, a method for optimal weighing vector w for regression in this case be written as given in equation 2.29:

$$w = \sum_{i=1}^{N}(\alpha_i - \alpha_i^{0})\varphi(x_i) \quad (2.29)$$

Furthermore, need to use the kernel as we have no information for φ in Hilbert space. Thus, the final equation in this case is given in equation 2.20 below.

$$y_i = \sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i 0)\varphi(x_i^T)\varphi(x_j) + b = \sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i 0)K(x_i,x_j) + b \quad (2.30)$$

However, if we use kernel trick there is no need to calculate weighting vector, all we need to find is parameter b. This is calculated by using following equation 2.31.

$$b = y_i - \sum_{i,j=1}^{NSV}\alpha_i y_i K(x_i,x_j) \quad (2.31)$$

General steps for executing SVM for classification & Regression:

Following are the steps which are executed for running SVM in classification and Regression.

1. Preparation of pattern matrix

2. Kernel Selection
3. Parameter Selection
4. Execution of training algorithms
5. Classification/ regression

Thus, in the process of performing regression using SVM in orange tool, we have fixed the parameter values as which are provided in chapter 4. In the next section we explain another supervised learning technique known as Linear Regression.

2.1.3 Linear Regression

Another statistical used for the analysis of independent variables also known as explanatory variables and target variable which is dependent variable [66]. Let's suppose the input vector (X) belongs to subset of R^d. The label associated to input vector is Y. To get the linear function of h: $R^d \rightarrow R$. For some feature d, the linear function is given in Figure 2.8, adopted from [79].

The hypothesis of linear regression is represented in Figure 2.8 [79].

$$y = h_\theta(x) = \theta_0 + \theta_1 x \tag{2.32}$$

In the above equation, the function is formed in which all input values are mapped to their corresponding target value (y). To measure accuracy of hypothesis that we have formed, we use cost function. The average of the results is calculated with X as an input and y being output. Therefore, the cost junction for linear regression is presented in Equation 2.33.

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(y_i' - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \tag{2.33}$$

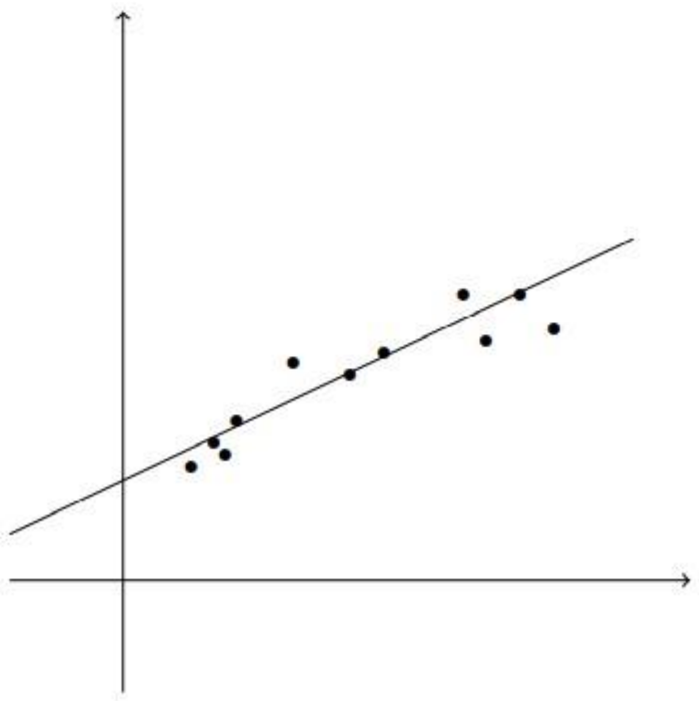


Figure 2.8: Linear Regressive function for single feature,[79]

Where $h_\theta(x_i) - y_i$ means difference between actual and predicted values. The function is also known as mean of squares of this term.

Moreover, gradient descent is used to estimate features of hypothesis function. We take smaller steps based on value of learning rate (α) on cost function to reach steepest point.

The gradient descent of algorithm is repeated till it reaches to convergence. Following is equation used for gradient descent (See equation 2.34 & 2.35): Apply gradient descent for θ_o & θ_1 {

$$\theta_o = \theta_o - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i) \tag{2.34}$$

$$\theta_1 = \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)x_i \tag{2.35}$$

}

Linear Regression using Multiple Features

The cost function for multiple variables remain same (Referred to equation 2.33). The gradient descent also remains same. The only difference is it has to run several

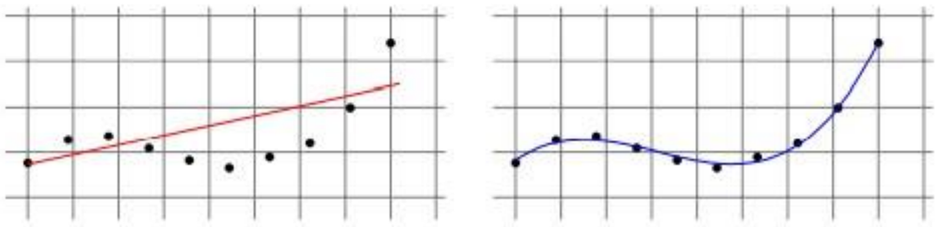


Figure 2.9: Regression with single and n-dimensional polynomial features [66]

times till n features. The general form of gradient descent is presented in equation 2.36.

$$\theta_j = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i).x_j \qquad for j := 0 \ldots n \tag{2.36}$$

Whenever we have multiple features they may be on different scale or must be nonlinear. There different ways to handles such features. One is Feature normalization and another is Normal equations. In feature normalization and mean normalization, we scale the variable by the division of range of input values to mean or standard deviation. Another technique is polynomial regression. (See Figure 2.9, adopted from [66]). In this technique we combine multiple features to make one useful feature. Furthermore, to find the optimum value of theta we use normal equation instead of gradient descent. The general form of normal equation is given in Equation 2.37.

$$\theta = (X^T X)^{-1} X^T y \tag{2.37}$$

When using normal equation , feature scaling is not required [80]. To find the linear fitting of linear regression, elastic net regression is used [81]. It is applied in conditions where we have highly correlated features [82]. This method is based on regularization. It combines L1 and L2 in linear form from ridge(Tikhonov regularization) and Lasso(least absolute shrinkage and selection operator) regression [81]. To improve accuracy elastic net regression finds the highly correlated variables in the model. It uses these variables and adds a penalty from ridge regression to find best estimates by improving accuracy of model.

To reduce over fitting ridge in elastic net shrinks the coefficients of regression. It does not perform selection of covariates rather it just adjusts the values to fit model

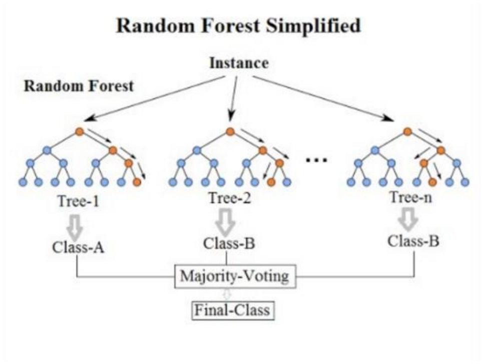


Figure 2.10: Random Forest [79]

in certain circumstances. However, in Lasso the variables are selected, and their value of coefficient is shrunk with respect to some threshold value. Both ridge and lasso fit the model by keeping value less than threshold value. Therefore, elastic net regression combines both penalties. First it finds the highly correlated variables and the tries to adjust the value of least square distance less than some value. Lasso finds one important variable and ignores all other therefore, ridge in elastic net adds a quadratic term which is used to fix the limitation of lasso. Thus, the general form of elastic net regression is derived as Equation 2.38 & 2.39.

$$||\beta||_1 = \sum_{j=1}^{p} |\beta| \tag{2.38}$$

$$B\hat{} = argmin_{\beta}(||y - X\beta||)^2 + \lambda_2||\beta||^2 + \lambda_1||\beta||_1 \tag{2.39}$$

In order to run linear regression using multiple variables in orange tool. We performed different settings of parameters which is presented in chapter 4. In the next chapter, we presented Random forest.

2.1.4 Random Forest

Random forest is a machine learning technique used for classification and regression purpose [18]. It works under the principle of decision trees [82]. Decision trees usually have three nodes such as leaf node, then internal nodes and finally decision or root node (See Figure 2.10 adopted from [79]). In decision trees the branches are the results of root and internal nodes. The hierarchies in each division represent the classifications in decision trees. Different methods such as entropy, classification error etc. are used to find the depth level of trees. The two problems of over fitting and under fitting are caused by variance and bias. Variance is error of classifier

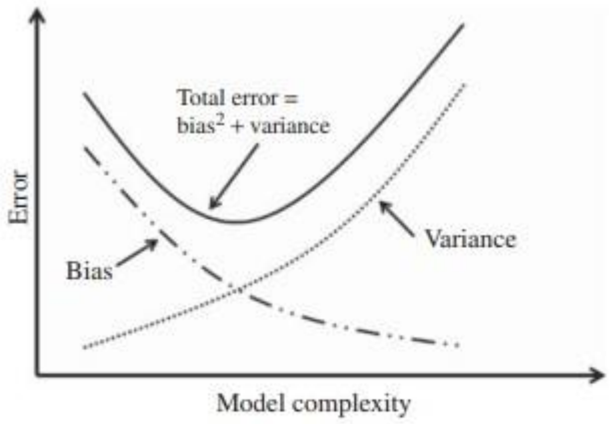


Figure 2.11: Variance and Bias, [82]

due to its variability. However, bias is error of difference between the predictions and actual values. The effect of reducing bias is presented in Figure 2.11 which is adopted from [82]. This figure shows if we reduce the variance resultantly it increases bias.
Bootstrapping also known as bagging is an ensemble method which is used to improve classification accuracy of an algorithm. Furthermore, a technique known as Random Forest is proves improvements of bagging. It de-correlates trees by using small tweaks. Let's suppose we have one highly strong variables and rest are moderately stronger variables. In bagged trees, the highly correlated variable is considered, and rest of the variables are not given any importance. As a result, trees which are formed acts as highly related trees. Therefore, random forest is developed by Ho [83]. Moreover, Random forest splits on the basis of random selection of variables. On average, the Equation 2.40 shows random selection and it is not inclined towards highly correlated variables.

$$\frac{n-r}{n} \tag{2.40}$$

Where n is number of features and r is random splitting of trees. However, for regression problems with random forests, the v/3 (v corresponds to variables) rounds should be selected for minimum node size of 5 [83]. Moreover, random forest in orange tool is executed by setting no of trees, no of splits in each tree and depth of trees. Thus, parameter setting is given in chapter 4. In the next section we explain another supervised learning technique named as Neural network.

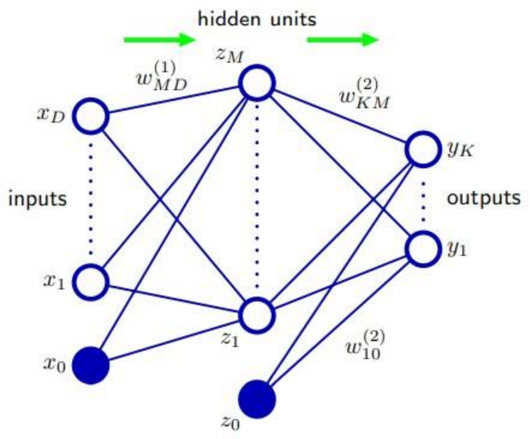


Figure 2.12: Neural Network Structure [73]

2.1.5 Neural Network

Neural network is interconnection of nonlinear elements with the weights. The elements along with their weighted sum [61]. The simplest form of neural network is presented in Figure 2.12 (which is adopted from[73]). The first layer is the input layer. Next is the layer of hidden units. We can have multiple hidden layers. Finally, the last layer contains output units or nodes. The one of the important elements in neural networks in threshold value.
The linear functions are easy to implement. These functions work by adding weighted input. This is then compared with threshold value. Figure 2.13, which is adopted from [61]. This is threshold logic unit (TLU). The output remains in 1 or 0. The output depends on the threshold value. Each element is known as perceptron, Adaline and neuron. The input vector having multiple features (n-dimensional) is represented by

$$X = (x_1,x_2........x_n) \tag{2.41}$$

The elements present in X is real- valued number. These are specific to binary numbers 1 and 0. The weights for X are given in TLU are given in W= $(w_1, w_2 \ldots\ldots w_n)$.
Furthermore, if TLU remains the output to 1 is it follows following condition presented in Equation 2.42. If this is not satisfied the output remains 0. The weighted sum is calculated as dot product X.W. it is sometimes represented as $X^t W$.

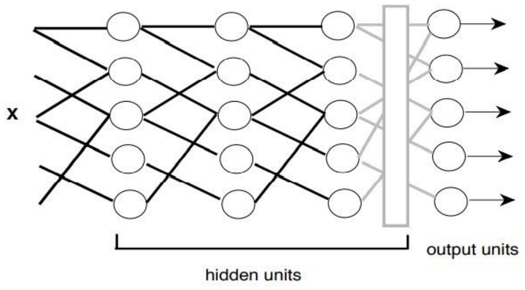


Figure 2.13: Forward propagation Neural Network [61]

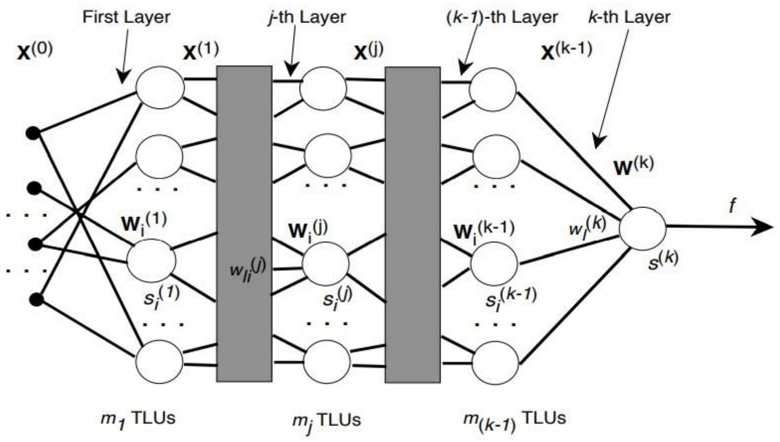


Figure 2.14: Multi-Layer Neural Network

$$\sum_{i=1}^{n} x_i w_i \geq 0 \tag{2.42}$$

The structure of forward propagation is presented in Figure 2.13. The forward propagation neural network is simple neural network in which we have no access to previous layers. The derivative term of previous layer becomes zero and we cannot trace back. In forward propagation, j^{th} term receives input from $j\text{-}1^{th}$ term of network. The multi layered neural network is presented in Figure 2.14, adopted from [61]. We have fixed values of α and no of iterations for application of neural networks. The detail is provided in chapter 4. In the next section we explain another supervised learning technique known as K nearest Neighbor.

2.1.6 K nearest Neighbor

K-nearest neighbor is a memory-based classification classifier [63]. This tool is used to identify the hidden patterns [84]. For any target x_0, KNN finds the nearest

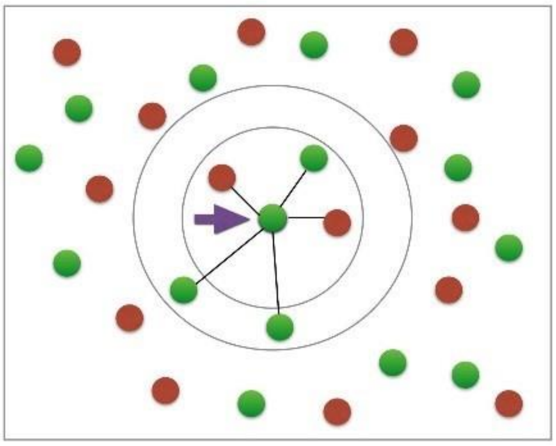


Figure 2.15: K Nearest Neighbor [85]

neighbor such that the closest training point be $x_{(r)}$, where r = {1,2......k}. Next step of KNN is to find the majority of nearest neighbors and their vote. This concept is presented in Figure 2.15, adopted from [63].

If we have real valued data, we find the Euclidean distance between the data points. Thus, we calculate distance between two points. The Equation 2.43 represents formula of Euclidean distance.

$$d_{(i)} = ||x_{(i)} - x_0|| \tag{2.43}$$

The decision boundary of KNN classifier is determined locally and is based on value of k [85]. Furthermore, KNN when applied for classification returns the mode of k labels however, in case of solving regression problem it returns mean of k labels [66, 67]. Moreover, if we choose smaller vale of k, we do not get robust results and higher value k of k presents produce low noise and smooth boundaries. To overcome these issues weighted method is used over all variables [86]. To use KNN in orange tool, we selected the uniform weighted method which assigns equal weightage to all the neighbors and selected the value of k to 3. The process of training data with unlabeled data set on the basis of characteristics is known as supervised learning. In supervised learning as described in previous section, we group data into clusters based on pattern and characteristics of data set. Therefore, the K-Means clustering is applied on data set of effort estimation. In the following subsection, we explain the clustering process specifically K-means clustering.

2.1.7 K Means Clustering

The process of grouping similar objects in same group is known as clustering[61, 87] and class boundaries are undefined and statistical [88]. The clusters are represented in Figure 2.16 (adopted from [89]). There are different clustering algorithms such as hierarchical clustering, K- means [90] and C means. The algorithm used in this study is K-Means algorithm. The working of K means is presented below. **Steps to form Clusters** The steps to allocate clusters are given below:

Step 1: User has to select number of clusters and the centroid for each cluster.

Step 2: Distance between data points is calculated. The item we are predicting takes the minimum distance between data points and centroid. Step 3: Repeat the step to find centroid again till the user requirements are satisfied.

Step 4: If we find that the desired clusters are not formed, repeat process from step 2 till you achieve desired results.

Similarity between two Numeric values

The formula to calculate similarity between two numeric variables [91] is presented in equation below (referred to equation 2.44).

$$D(X,Y) = |x(n+1) - x(n)| + |y(n+1) - y(n) \qquad (2.44)$$

However, to find similarity based on categorical variables, we first convert the data into matrix and then we calculate the Euclidean distance the motive behind using this technique is its similitude with human understanding [25]. The primary step in K-Means Clustering is to adopt number of clusters which is represented by K. The subsequent step deals with picking centroid for each cluster. Simplest way is to select k randomly from the given data points, we may have multiple iterations to get accurate centroid for each cluster. Usually, distance metrics known as Euclidean distance is widely used in clustering phase but we have other measures too [88].

Cluster Quality

Furthermore, to examine cluster superiority and assigning quality of cluster, the projected method use Silhouette Index as presented in studies [25, 92]. The Equation 2.45 demonstrates how silhouette Index is calculated for analyzing the quality

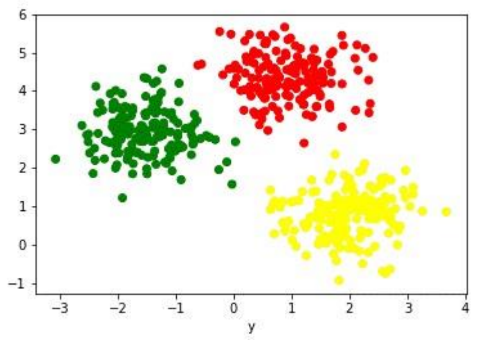


Figure 2.16: K-Mean Clustering [89]

of cluster. We intend to select K= {3,5,7} for clustering process. We have selected ORANGE tool for the application of K-means clustering algorithm. Further detail about parameter setting is given in chapter 4. To access the cluster quality equation 2.45 presents the formula.

$$Average \quad Silhouette \quad Score = \frac{(B-A)}{(max(A,B))} \qquad (2.45)$$

where A is considered as average distance to each project and B is measured as average distance to projects in all other clusters. In the next section, we have explained most widely used evaluation measures in arena of software development effort estimation. Furthermore, we have applied all these measures explained in next section.

2.1.7.1 Accessing Accuracy of Effort Estimates

The following evaluation measures are used to validate results as researchers have done before. The evaluation measures along with techniques are presented in 3.

The most widely and repeatedly used measures are discussed below:

Absolute Error

The difference between actual and estimated values is known as Absolute Error [93]. The formula is presented in equation 2.46.

$$abs.err = x_1 - x_2 \qquad (2.46)$$

Where x_1 is predicted value and x_2 is actual value.

Magnitude of Absolute Error

Magnitude of absolute error [94] is calculated as (referred to equation 2.47)

$$MAE = \frac{1}{T}\sum_{i=1}^{T} |y_i^{\wedge} - y_i| \qquad (2.47)$$

[y_i^] predicted value and y_i is actual value.

Magnitude of Mean Relative Error

The sample average MRE is known as MRE [93]. It is calculated as referred to equation 2.48).

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{2.48}$$

Mean Relative Error

The equation to calculate mean relative error [93] is given in below. (See equation 2.49)

$$MRE_i = \frac{|y_1 - y_i^{\wedge}|}{y_i} \tag{2.49}$$

where, y_i is predicted value and $y^{\wedge}_i$ is actual value.

Relative Error

The formula to calculate Relative error [93] is given in equation 2.50.

$$Rel.err = \left(\frac{x_1 - x_2}{x_1}\right) * 100 \tag{2.50}$$

Where x_1 is predicted value and x_2 is actual value.

Mean Squared Error

Mean squared error [94] is defined in equation 2.51.

$$MSE = \frac{\sum_{i=1}^{T}(y_i^{\wedge} - y_i)^2}{T} \tag{2.51}$$

y_i is predicted value and $y^{\wedge}_i$ is actual value.

Prediction (.25)

Let's considers the average fraction of the MRE's off by no more than x as defined by [93]. The formula to calculate Pred is given in equation 2.52.

$$PRED(x) = \frac{1}{N}\sum_{I=1}^{N} 1 \quad ifMRE \leq x \quad else0, where \quad MMRE \leq 0.25 \tag{2.52}$$

Root Mean Squared Error

The root mean squared error [94] is defined in equation 2.53.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T}(y_i^{\wedge} - y_1)^2}{T}} \tag{2.53}$$

where y_i is predicted value and $y^{\wedge}_i$ is actual value.

2.1.8 Summary

The chapter has explained the methods and algorithms that were employed in this study. We provided details about linear regression, k nearest neighbor, support vector machine, K-means clustering, neural networks and random forests. In the next chapter, we would present the related studies of software development effort estimation.

Chapter 3 Related Work

The objective of this chapter is to review related work in the arena of software development effort estimation. The main focus is to find the answer of first research question by critically analyzing the strengths and weaknesses and to come up with the limitations.

3.1 Related Work

Software development effort estimation is an active area for researchers since 1960s. This process falls under software project management. The aim of this chapter is to highlight limitations of previously proposed solution. The effort estimation techniques are generally grouped into three categories i.e. parametric, non-parametric and machine learning based models. Thus, in this section, First we have analyzed the strengths and weakness of parametric and non-parametric models. Second, we analyzed machine learning based methods for effort estimation of software projects.

3.1.1 Parametric and non-parametric Estimation methods

Software Development estimation is an active area for last three decades. There were plethora of studies conducted to fix the problem of effort estimation. Therefore, the aim of this section is to analyse strength and weaknesses of existing literature for expert based estimation, parametric and non-parametric models.

The non-algorithmic techniques are based on inference and comparing things analytically. These techniques require information of already completed projects.[97] These techniques are expert judgement and analogy-based estimation.

Barry Boehm in 1987, proposed a method named as Cocomo[98]. This model is based on parametric model and produce effort and duration of software projects. It is important to accurately measure the cost drivers which are used for making predictions. In 1983, Galorath, Inc. of El Segundo, California proposed a model named as Seer model which was developed on the basis of Jensen model [99]. This model takes input in form of size, personnel, environment, complexity and constraint. As a result of processing, it produces results for cost, effort, schedule, maintenance, risk and reliability of projects.

Boehm [8] in 1981 proposed a technique known as expert judgement. Ballay states expert as “person who has the knowledge” and judgement is a process in which a person continues to practice. The quantification step of expert judgement is based on the knowledge and on spot decision of experts [100]. However, there is no statistically collected method for expert judgement. Rather it is based on spontaneous decisions which could be either due to political pressure or some unfair measures [101]. The expert judgement is considered beneficial because it uses the knowledge gained from previous experience. But sometimes it could be challenging to use expert judgement due to following reasons. First, the deficient knowledge due to less experience which is a basic reason for underestimating an effort of project [9]. This technique is based on another method known as Delphi method [102]. This method was proposed in late 1940s. There are multiple rounds in which participants provide their estimates and with consultation they reach to a single effort estimate. Another method which comes under expert judgement is known as Work break down structure. This method is a bottom up estimation in which each individual item is considered and used for making estimation. Another non algorithmic technique is known as analogy-based estimation [9]. This technique is based on case-based reasoning. This estimation is performed either of system level or making estimates for smaller systems known as sub-systems. This technique is based on selection of right analogy, then making boundaries between similarities and dissimilarities. Then it is very important to examine the quality of analogy. After the detail analysis of these steps the quantification step provides

estimates for each project [97]. The main goal is to allocate effort to project to fill needs of time and budget [102]. This techniques uses similarity measures such as Euclidean distance between the previously completed projects and new projects
[103].
After non algorithmic techniques various methods have been proposed which are known as algorithmic techniques. The examples of such methods are explained below. Additionally, in 1977, at RCA another method names as Price-S [104] was developed as a tool which uses functional points to estimate project size for input. The equation of this tool was not released. This model was used to estimate schedule and cost of software projects. However, with evolving software solutions they were updating their tool. The model named as Putnam's Software Lifecycle Model (SLIM) [105] was proposed by Larry Putnam in 1970s. This model supports model such as Function point analysis. However, this model need data for previously completed project. In case of unavailability of data, this model requires answering questions to make a Rayleigh distribution curve. Thus, if the information of previously complete projects is not available or correct this model would not provide better results of estimation.
Another algorithmic model named as Checkpoints[106] was developed from studies named as Capers Jones'. The input of size in this method is made by function point analysis. This method is used for prediction at four levels such as task, project, activity and phase. This method help users to perform benchmark analysis to make effort estimates [102]. Function Point [107] is another method which eliminates the problem of line of code by producing the estimates of size and complexity or project. This method has nothing to do with the language of program [102]. unfortunately, this method is not utilized by many estimation methods.
This section has provided information related to methods which do not involve machine learning. However, from the literature review we concluded none of the above presented method works equally good in all environments [102]. Therefore, in the next section, we presented a literature review for effort estimation using machine learning techniques.

3.1.2 Machine Learning based Estimation methods

The trend in estimation has changed and researchers are investigating the use of data mining and machine learning for the purpose. Therefore, as seen in Kumar S et al [38] machine learning algorithms are applied to publicly available data sets named as Desharnais and COCOMO over ORANGE tool for K- Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF) and Neural Networks(NN) and Python for the application of back propagation neural network algorithm. This study reported the use of back propagation algorithm to be the best among on given data sets for the successful completion of projects which involves estimates related to staff, cost and time. The produced effort estimated were evaluated using Mean Magnitude of Relative Error (MMRE) and Adjusted square (R^2). The main limitation of this study was it was implemented and tested over publicly available data sets. Furthermore, this study has used only two data sets. Therefore, we are unaware of the application of the proposed methodology in different environments.
Further, in work done by H karna et al [39] we studied the use of data mining method such as K- Nearest Neighbor (KNN) which is extended version of Neural Networks was applied on agile projects. This study suggested application of data mining for the purpose of estimation in industry as this experiment was conducted with industrial data set. The method is this study used Estimation error (EE), Magnitude of relative error (MRE), Mean Magnitude of Relative Error (MMRE), Pred(x) as evaluation measures

for the validation of results. The main limitation of this study was the application over one single large agile project.
Similarly, use of machine learning algorithm has been addressed in work of Arsalan F [108] and model is presented. The model uses Random Forest, REPTree, Gussian Processes, Linear Regression, M5P, ZeroR, Decision Table, Input Mapped Classifier, KStar, Multilayer Perceptron, IBK, Additive Regression, and SMOreg algorithms. As a result, Random Forest outperforms on Usp05-ft dataset while Kstar, REPTree and Additive Regression outperformed all other over Usp05 data set. Therefore, the results encourage estimators to apply machine learning for making accurate predictions. The results are evaluated using Adjusted square (R^2), MAE, RMAE, RAE, and RMSE. This study is limited to two mentioned data sets and the inputs to model need to be improved for more reliable predictions.
To predict the effort estimate as done in various studies, differential evolution algorithm for feature optimization and its effectiveness is studied in work of Benala T et al [29]. This study has reported the increase in estimation with use differential algorithm over partial swarm optimization, analogy based, genetic algorithm, functional link artificial neural networks etc. The proposed method was tested on data from real world and publicly available datasets. The estimated are evaluated with the use of MdMRE, PRED (0.25), MMRE, SA, and Δ. Furthermore, for the improvement of effort estimation the feature selection technique in work done by Abran S [109] uses entropy based method. This study has reported the positive influence of entropy-based method for numeric data. However, the effect of entropy on categorical data variables remain unidentified. Furthermore, the effect of entropy was tested for seven data sets still this is not enough of generalization of results. Thus, more data sets should be utilized for making comprehensive results.
Alternatively, Principal Component Analysis, a data mining technique was used for feature reduction and then classifiers such as K- Nearest Neighbor (KNN), decision trees (DT) and Na¨ıve Bayes were applied to make prediction in work of Nejad and Tavoli [110]. This study supports the use of PCA with any of these techniques but specially KNN, which performs best in current scenario of NASA and COCOMO data sets. In future, this study could be implemented to test other machine learning classifiers for effort prediction. The analysis of results was made on precision, accuracy and recall. This study has main focus of reducing the size of input to model. So, one of the most main tasks is to decide which feature would be eliminated from input.
Also, the study presented by Minku and Yao [111] provides a Dynamic Cross company Learning (DCL) method to analyses effect of machine learning on effort prediction. This study reported the positive influence of DCL which is a weighing method on accuracy of predictions by considering the recent projects. This study was based on two datasets from NASA and COCOMO and three data sets. This method does not involve the practitioners to identify and choose any previous estimation method. Moreover, the results produced by DCL are evaluated using Mean Absolute Error (MAE). DCL method, however, needed investigation for industrial data sets.
In addition, to improve accuracy of COCOMO dataset using gaussian function, results presented in work of Shankar G [112] proposed fuzzy approach with gaussian function with use of Support Vector Regression over K- Nearest Neighbor (KNN), Linear regression (LR) and artificial neural network (KNN). The results showed that SVR and ANN performs better in most cases than other techniques. The results proposed by each model are evaluated using Root Relative error, relative absolute error, mean absolute error, root mean squared error, mean relative error and correlation coefficient. This

study however needs more attention and should be tested for data set having different characteristics.
Moreover, the use of data mining techniques such as decision trees and bagging using WEKA tool were testified in work of Bedi S R et al [113]. The results produced by both techniques are validated by mean absolute error, correlation coefficient, relative absolute error, root mean squared error, and root relative absolute error. The data set used in this study was Promise repository data set. The comparison of estimates was made on effort and months basis. However, both the methods should be used for other data sets also. This would bring the generalized results as dependable estimation still remains challenging in different environments.
Using fuzzy C means with neural network & optimizers algorithms such as artificial bee colony (ABC), modified cuckoo search (MCS) and hybrid ABC-MCS algorithms was studied in work of Azath H et al [30] and informed increase in performance is observed with the specified approach. the experiments were conducted over two large real-world data sets. The performance of algorithm is evaluated using mean magnitude of relative error and mean absolute relative error. The problems of underestimation and overestimation leads software projects towards failure. In case of under estimation, the cost and delivery are affected. In contrast, the overestimated projects are reason for financial loss and outbidding issues inside an organization.
Thereafter, in work done by Chhabra and Singh [114] proposes a model using fuzzy logic combined with intermediate COCOMO to identify effect of cost parameters. Fuzzy logic is considered beneficial when there is missing information and prediction is at risk. This method also leads to increase in estimation over publicly available datasets named as COCOMO and NASA. Moreover, the results are analyzed by using MRE, MMRE and Pred. The same method could be extended for functional point analysis and other models. However, there is need of hybrid method which are able to improve accuracy of predictions.
A new method using fuzzy logic was proposed in work of Nassif A et al [115] for estimation of effort. This technique uses three fuzzy models Sugeno with constant output, Mamdani, and Sugeno with linear output. They described use of Sugeno would enhance accuracy on data set named as e International Software Benchmarking Standards Group (ISBSG). The results were evaluated sing standard measures such as effect size, standardized accuracy and mean balanced relative error. This study further removes a cover from effect of data heterogeneity and outliers' impact on effort estimation using fuzzy logic. According to this study, there is great impact and unclean data sets containing outliers badly effect estimation accuracy.
To bring rightness in effort estimations researchers are dealing with project's missing data was investigated in work done by Zhang W et al. [116] The techniques used for data imputation are Bayesian Regression, Linear regression, M5' regression, Support Vector Machine and BREM. This study informed BREM outperformed all other techniques. This experimentation was conducted for CSBSG (Chinese Software Benchmarking Standard Group) and ISBSG datasets. The limitations of this study lie between the application over more data sets and other performance measures need to be implemented for evaluation of model.
Likewise, Vijay et al [117] in their work analyzed the effectiveness of using fuzzy logic when dealing with data. This method presented in this study uses fuzzy based method in functional point analysis and quality factors. Thus, fuzzy logic was implemented to deal with software size by using triangular fuzzy set and at the end this study has reported increase in accuracy of estimation for real world data set collected from

software industry. The results generated by proposed method are evaluated with VAF and MMRE. The primary focus of this study was to put functional and nonfunctional properties of software project in form to effort estimation. Another work using fuzzy logic for data imputations was reported in work of Abanane I et al [118]. This study, therefore, investigated and reported that use of FA-KP-1 would be beneficial to deal with categorical data imputations and thus, it would in return be able to increase accuracy of estimation. The missing mechanisms such as NIM, MCAR and MAR are used with four data sets to perform analysis.

Similarly, to deal with multiple imputations of data the work has been done by Abran and Bala [119]. This study advises use of multiple imputation techniques instead of single imputation due to reason that it brings accuracy to predictions. The data sets from ISBSG are selected for this study because they have many missing values such as lines of codes, resource level, maximum team size etc. This study has used adjusted R2, Pred(x) and MMRE. The problem of data imputation causes misleading and biased results. Therefore, this problem should be addressed and sorted. Additionally, the work done by Tanveer B et al [33] proposed a hybrid method for estimation of agile projects. The use of hybrid model with Gradient boosted trees would increase estimation accuracy of software projects. This study has been implemented on data set collected from Insiders Technologies GmbH, a German software company. The results are evaluated using Pred (x) and MMER.

Meanwhile, the work using multivariate linear regression and deep structured multi-layer perceptron using different optimization algorithm was studied in work of Resmi and Vijayalakshmi [19]. This study proves the use of the classification techniques along with clustering provide better results as compare analogy based estimation without machine learning. The data sets from Promise repository named as Cocomonasa60, Cocomo81, and Cocomonasa93, ALBRECHT, DESHARNAIS, Miyazaki1, Kemerer and MAXWELL are used for this study. Moreover, this study has used following evaluation measures: Classification accuracy, correlation coefficient, prediction and MMRE. The limitation of this paper was identified as no preprocessing was performed prior to application of data mining and machine learning.

Similarly, usage of neural network in amalgamation with evolutionary techniques for effort estimation was investigated in work of Khazaiepoor M et al. [120] Their results indicate the effectiveness of techniques on all selected datasets such as Cocomo, Desharnais, Albrecht, Maxwell, ISBSG and China. These data sets contain few records except for one data set named as CHINA. The results are evaluated using MMRE, MdMRE and Pred(x). Moreover, the results show that the prediction over china data set are better than other data sets.

Further, the investigation of hybrid technique using neural network based fuzzy logic with incremental data based clustering algorithm and then bootstrapping smoothing was noted in work of Souza et al [121]. The practice of this technique not only reduces the execution time but also reduces accuracy in predictions. The results are evaluated using Root mean squared error (RMSE). Feature selection and correlated feature selection remains an important area in software development effort estimation and we need to address this problem for data density algorithm. In continuation more work has been done by Souza et al [122] with fuzzy regularized neural network for effort prediction of software projects. This study supports the usage of their proposed technique for estimation of project prior to starting phase. They used a real-world data set for evaluation of proposed model. The results proposed by this study are not interpreted because they are formed by a black box problem.

The application of fuzzy logic has not ended here. Thus, more work has been seen in work of Kaur I et al [123], where Neuro fuzzy logic using COCOMO 81 was implemented over MATLAB tool. Their study supports use of proposed method for cost estimation as it produces better estimates. The experiments were performed over NASA data set with application of Cocomo 81 model. The fuzzy logic basically named as neuro fuzzy logic was implemented for Cocomo 81. However, it could also be implemented for other parametric models such as Slim, Functional point analysis etc.
The application of Real Time Extreme Learning Machine (RT-ELM) for effort estimation was investigated then reported in work of Pillai K et al [31]. This proves usage of their approach for publicly available data sets. This technique has online sequential learning algorithm to learn from all new recently added projects. The estimates produced by this technique shows that radial basis function and the new additive hidden nodes are not dependent on data. Furthermore, the results are validated from industry and evaluation measures such as RMSE, Correlation, Kurtosis, Skewness, IQR, mean, median, maximum and standard deviation.
However, to analyses effect of support vector regression (SVR) to estimate effort during maintenance phase is studied in work of Garcia-Florina A et al [17]. This study reports use of SVR with polynomial kernel performs best. This study was conducted on five data sets which are collected from ISBSG data sets. The limitations of this study are linked to few tested data sets. Furthermore, this study is conducted for projects which are under maintenance. This method needs to be tested for all type of projects.
In the study of Carvalho et al [124] we observed ensemble regression methods using bagging applied on Linear Regression (B-LR), Ridge Regression (B-RI), Robust Regression (B-RR), Lasso Regression (B-LA), Robusta, Lasso and Linear meta-predictor (ST-LR), Stacking with Ridge, Stacking with Linear, Stacking with Linear, Robusta, Lasso and Robusta meta-predictor (ST-RR), Lasso and Ridge meta predictor (ST-RI), Lasso meta-predictor (ST-LA) and Stacking with Linear, Robusta, Ridge. Consequently, this study as a result of their experimentation, specified use of their proposed ensemble regression method to be best among previously generated methods. The predictions are evaluated using Mean Absolute Residual (MAR). This study is limited to smaller data set with few features.
An ensemble method using different solo algorithms to form one stack based stable ranking method based on publicly available industrial datasets could improve estimation accuracy of software projects as mentioned in work of Phannachitta and Matsumoto [125]. The results had shown the stack with combination of ordinary least square regression, adaBoost, bagging, analogy-based estimation, and bagging provides promising results. The total of 13 data sets which are extracted from Promise repository are used in this study. To evaluate performance of model, this study has utilized MAE, RSD, MBRE, LSD, MBRE and MMER.
Alternatively, in another work of Thamarai and Murugavalli [126] Genetic algorithm based on Expert Judgement for effort prediction was investigated with a technique named as Modified Genetic Algorithm-Simulated Annealing (MGASA). This technique outperformed all other techniques on NASA dataset in context of software effort estimation. The main challenges in effort estimation are selection of features and components of projects. Moreover, the results are evaluated using Relative Error (RE), Mean Magnitude of Relative Error (MMRE), Magnitude of relative Error (MRE) and Pred (Percentage of prediction). This methodology needs to be tested for more data sets do that the application of this model would yield efficient and reliable effort estimates.

Additionally, in study conducted by Khatoon and Kaur [127] the use of genetic algorithm for optimization of COCOMO parameters was evaluated. This study proves practice of this approach could be valuable in attaining accuracy over real world data sets having characteristics like NASA data sets.
In studies, of Xia T et al [128], we have seen the tool OIL, based on analogies, which was tested over publicly available dataset for estimation and optimization of features. These study support using CART and FLASH as they outperformed others. The outlier method should be change as it is not effective and does not contribute to improve estimates. Effectiveness of model is to be tested on more projects so that results are improved. The results are evaluate using IQR which is inter quartile range, Standardized accuracy (SA) and MRE. While the same tool in [129] was evaluated using Magnitude of Actual Residual, MRE, SA and MAE. This study has used hyperparameter optimization. The limitations are linked to use of few classifiers, data preprocessing which is process of selection of features.
Similarly, with intention of increasing accuracy in making predictions, a new method using Deep Neural network was proposed in work of Menash S et al. [130] This technique uses Bellwether moving window with Tri-weight function on three data sets(Desharnais, ISBSG and Kitchenham). This study shows effectiveness of technique over new projects in a window. The results are evaluated using Cliff's δ effect size, Brunner's test at 5% asymptotic significance level, MAE and Yuen's test. However, a new idea which supports the usefulness of software analytics for effort prediction was perceived in work of Hassan A et al. [131] Extending their work for use of Bellwether to estimate effort and other some other areas is investigated in Krishna R et al [132]. The predictions are evaluated using standardized accuracy (SA).
The use of hybrid method which combines Particle swarm optimization technique with case based reasoning over two datasets (Desharnais and Maxwell) was investigated in work of Wu D et al [133]. This study supported using combination instead of using one as generated result produce lesser error magnitude. The predictions made by model are evaluated using MdMRE, MMRE and Pred(x). The results have proved the weighted method to be better than unweighted Case Based Reasoning (CBR).
However, use of hybrid morphological perceptron for effort predictions was studied in work of Bilgaiyan S et al [134]. To increase accuracy in predictions, this method has used CMPSO algorithm for improvement and optimization of DEF parameters. The accuracy of this technique was presented over five publicly available datasets. These are Albercht, KotenGray, Cocomo, Kermer and Desharnais data set. The estimates made for all the project inside data set are evaluated using Evaluation Function (EF), MMRE and Pred(x).
Further, indicative -based optimization for effort estimation using MATLAB was seen in work of Alsalman and Ali [135]. This uses technique known as Cat swarm estimation. The use of this technique over NASA dataset produces estimate near to actual effort of software projects. This method is evaluated with MRE, RMSE, MORE and MAE. This study is limited to one data set; therefore, the results are not generalized. Also, use of particle swarm optimization algorithm with COCOMO model had increased accuracy over dataset of Turkish company in work of Langsari K et al [136]. This method has an ability to take incomplete inputs and could easily deal with them. The method was evaluated using MMRE. Additionally, use of data mining for feature selection and then effort estimation is presented in work of Jodpimai P et al [137]. This study used thirty eight software projects which were collected from two software organizations. The evaluation of model is performed by MBRE, MdBRE, MIBRE, MdiBRE, 10-Fold cross

validation, Inversed Balanced Relative Error (IBRE) and Balanced Relative Error (BRE). This technique proved itself to be better for re-estimating software project.
More, the use of Analogy based estimation (ABE) with combination of other techniques were proposed in this study of Bardsiri [138] This study was conducted on two different data sets. First is ISBSG data set and second one is collected by students. Moreover, both the data sets are evaluated using Pred (x), MdMRE, and MMRE. Results shows proposed technique known as ABEM outperform other developed models. With increased number of variables and complexity of software project, the estimation remains challenging task.
To dealing with combinations of algorithms, in the work of Malgonde and Chari [139]. proposed an ensemble-based technique on the basis of expert estimation by using different machine learning techniques. Their results disclosed, ensemble predictor outperformed extra trees, random forest, average and other machine learning algorithms. RMSE, MBE and MAE were used for evaluation of models. The data set was collected from information available on Project management system. This method does not synchronize with new technologies.
Another ensemble method proposed in work of Pospieszny et al [32] uses SVM, MLP and GLM. This method was tested over ISBSG dataset and their results indicate towards the effectiveness of using ensemble techniques for estimation at prior stages. The predictions were evaluated using Pred (x) and MMRE. The use of machine learning techniques such as Deep learning algorithm and Gradient boosting machine is investigated in work of Phannachitta P [50] for effort estimation. As a result of systematic comparison this paper suggest using machine learning for effort estimation of software projects.
Additionally, for selection of features to improve accuracy of prediction is evaluated in work of Fernandez-Deigo M et al [140]. This study performed cross validation and MMRE as average value for cross validation to ensure the validity of their results in which they reported the use of K-NN work well in case-based reasoning. The main of this work to select features and rank them in two categories. These are continuous and numeric features. However, feature selection and weighing them properly could bring accuracy in predictions. The techniques proposed in work of Bardsiri A [141] uses a hybrid technique which outperforms all other techniques. The method was evaluated with MdMRE, MMRE and Pred (x). However, in study conducted by Tariq S et al, [24] we noted the use M5P and linear regressions for selection of attributes to make effort estimates. Further, work of H karna et al [142] use of SVM and ANN could improve accuracy as compared to other machine learning techniques. This study also reported application of MMRE is used in most of the studies for accuracy predictions. We have provided the highly relevant papers from related work in Table 3.1.

Table 3.1: Effort Estimation with Machine Learning

Authors	Description of Research	Methodology	Data set used	Evaluation Measures	Limitations
Barry Boehm [8]	The use of expert estimation in estimation of effort.	Nonparametric	NA	NA	Based on Single technique.
Kumar and Behera [18]	Machine Learning based prediction	•SVM •KNN •NN •RF	•Desharnais •COCOMO'81	MMRE	Two data sets only

Resmi and Vijayalakshmi [19]	Analogy based estimation based on clustering	•K-Means •DMLP •MLR •Firefly •Analogy based fuzzy logic	•Cocomo81 •Cocomonasa60 •Cocomonasa93 •Deshnaris •ALBRECHT •Kemerer •Miyazakil •MAXWELL	•Correlations •MMRE •accuracy •Pred (25)	Data preprocessing

Continued on next page

Table 3.1 – Effort Estimation with Machine Learning

Authors	Description of Research	Methodology	Data set used	Evaluation Measures	Limitations
Tariq S et al [24]	To select predictor and eliminate out-liers	•LR •M5P	•ISBSG •real data	MMRE	No outliers' inclusion and exclusion.

Pospieszny P [32]	Ensemble methods for estimation	•SVM •MLP •GLM	•ISBSG	•Crossvalidation •ME •MAE •RMSE •MSE •MMRE •Pred •MM •R •MBRE	Limitations related to data sets.
Farrukh Arsalan [108]	Using WEKA tool for prediction	•RF •DT •Gussian Processes •LR •MLP	•Upsp05 •Upsp05-tf	•R2 •MAE •RMAE •RRSE •RA E	Applied on two publicly available datasets

Continued on next page

Table 3.1 – Effort Estimation with Machine Learning

Authors	Description of Research	Methodology	Data set used	Evaluation Measures	Limitations
Nejad and Tavoli [110]	PCA classification (Rapid Miner)	•PCA •KNN •DT •Na¨ıve bayes	•COCOMO81 •NASA 93	•Accuracy •Precision •Recall	two publicly available data sets.

Minku and Yao [111]	Weights provided to learners by DCL and predictive performance	•MLP •RT •Bag+RT •k-NN	•ISBSG2000 •ISBSG2001 •ISBSG •Nasa60 •Coc81 •Nasa93	•Random holdout •crossvalidation •MAE	Proper parameter setting is required
Bedi R et al [111]	Data mining using WEKA tool	•Bagging •DT	PROMISE	•MAE •correlation •RAE •RMSE •RRAE	Leads to use of few data sets.
Ali and Gravino [14]	Use of SVM and ANN for improvement in accuracy	SLR	NASA	MMRE	keyword selection

3.2

Outcomes of Related work

The aim of this section is to highlight challenges in area of software development effort estimation identified from previous sections.

- Expert judgment is misleading due to spontaneous decisions, less knowledge and experience and political pressures.
- This ultimately leads to underestimation.
- Another challenge in making estimation is to fulfil the restriction of budget and constraint.
- Data of previously completed projects is input to most of the models therefore, it should be updated continuously.
- The schedule estimates should be very accurate in order to estimate cost properly.
- The information of size, personnel, environment, complexity and constraint is considered as important for accurate effort estimation. Therefore, it should be made compulsory to gain and verify the information which is input to most of the models.
- Cocomo model is depend on the time estimates at each level. If the estimator is unable to provide the accurate time period for each stage the estimation is not correct and is biased. • Individual assessment in functional point analysis could provide inaccurate and unreliable estimates.
- The accurate number of staff and cost should be allocated to project for successful completion.
- The generalization of models is needed as most of the studies either use publicly available data sets or data set of single software development organization.
- The inputs to model should be given a look and more is required in this direction.
- Effect of entropy method on categorical variables need to be addressed and more data sets should be employed for comprehensive results.

- For techniques which utilize feature selection, it is important to decide which feature is selected because the selected features have impact on estimation. • Accurate and dependable estimation still remains a challenge during estimation.
- Noisy data sets are difficult to handle and ultimately, these kind of data sets effect effort estimation badly.
- The problems of underestimation and overestimation leads software projects towards failure. In case of under estimation, the cost and delivery are affected. In contrast, the overestimated projects are reason for financial loss and outbidding issues inside an organization.
- Incorrect estimates disturb the planned budgets for completion of software projects.
- Data heterogeneity, data imputations and outliers are main considered as main cause of estimation error.
- The problem of data imputation causes misleading and biased results.
- The most important challenge in software development effort estimation is to bring transparency in it. • The optimization of cost drivers for parametric optimization is looked-for the accurate estimations.
- Pre-processing is necessary and important step. If it is ignored the overall estimation accuracy remains effected.

- Identification of features and correlated features have a significant impact overestimation.
- The main challenges in effort estimation are selection of features and components of projects.
- On time completion and delivery of software projects remain challenging task.
- Effort estimation is difficult process because of increasing complexity, addition of new variables, change and unusual nature of projects.
- Effort estimation remains challenging due to new technological improvements in organization.

3.3 Summary

This chapter presented dimensions of software development effort estimation such as estimation with machine learning and estimation with parametric and non-parametric methods. We further highlighted the limitations of previously related studies. In the next chapter, we present proposed methodology.

Chapter 4 Proposed Model

The prior chapters of this dissertation have provided detailed knowledge and background of software development effort estimation. These chapter have also defined the recent research patterns in arena of effort estimation and how we have formulated the problem statement for this research work. The proposed framework is designed with the aim of improving software development effort estimation for organizations located in region of Islamabad-Pakistan.

4.1 Proposed Framework

4.1.1 A machine learning based framework for software development effort estimation

Software development effort estimation is one of the vital process in software development. Making accurate prediction prior to starting the project has a significant impact on successful completion of project. However, effort estimation remains challenging and unresolved problem for last three decades. The research community could not come up with one broad solution to solve all estimation problems. With the aim of improving software development effort estimation, we have analyzed literature and identified the unsolved challenges in arena of software development effort estimation.

Effort estimation of software projects are sometimes misleading due to spontaneous decisions, less knowledge and experience and political pressures mostly from the managerial staff on experts. This pressure is sometimes caused by the customer as they want some giant task to be done in small duration of weeks. Due to this, the schedule and budget plans are disturbed and leads project towards failure. Effort estimates are made in a realistic manner. The estimates should be supportable to develop credible strategies for successful completing software project. These effort estimates are then taken by project manager of higher authorities to develop complete cost plan.

Data of previously completed projects is input to most of the models therefore, it should be updated continuously. The information of size, personnel, environment, complexity and constraint is considered as important for accurate effort estimation. Therefore, it should be made compulsory to gain and verify the information which is input to most of the models. The generalization of models is needed as most of the studies either use publicly available data sets or data set of single software development organization. Moreover, it is important to understand the importance of inputs provided to models for estimation. Identification of features and correlated features should be given position as they have significant impact overestimation.

Thus, one most important challenge in software development effort estimation is to bring transparency in it so that in time completion and delivery of software project is ensured even with changing unusual nature of projects. Another important factor which effects effort estimation is new technological improvements in organization. All the above-mentioned issues in estimation leads projects towards in two directions. First is underestimation of projects and second corresponds to overestimation. The problems of underestimation and overestimation leads software projects towards failure. In case of under estimation, the cost and delivery are affected. In contrast, the overestimated projects are reason for financial loss and outbidding issues inside an organization.

To the best of our knowledge, there was no work done to improve effort estimation of software projects for software development organizations located in Islamabad Pakistan. Therefore, there is a need of a model which is capable of minimizing error magnitude for estimation in region of Islamabad-Pakistan. Thus, a more efficient and reliable model is replicated and is applied for estimation in similar environment.

4.1.2 Conceptual Framework

Software development effort estimation is one of the vital process in software development. Making accurate prediction prior to starting the project has a significant impact on successfully completion of project. Therefore, there is a need of a model which is capable of minimizing error magnitude for estimation in region of Islamabad-Pakistan. Thus, a more efficient and reliable model is replicated [25] and is applied for estimation in similar environment.

In the proposed framework the first step is was to design a questionnaire. After formation of questionnaire, we have collected data from two software development organizations located in Islamabad-Pakistan. The collection process was full of challenges however, we succeeded to collect information of thirty eight software development projects. Thereafter, we performed pre-processing and analyzed the properties of variables present in data set.

The next step of this conceptual framework is to apply data mining techniques on data set to form clusters. The results produced by clustering are input to modelling phase in which we applied several machine learning algorithms. At

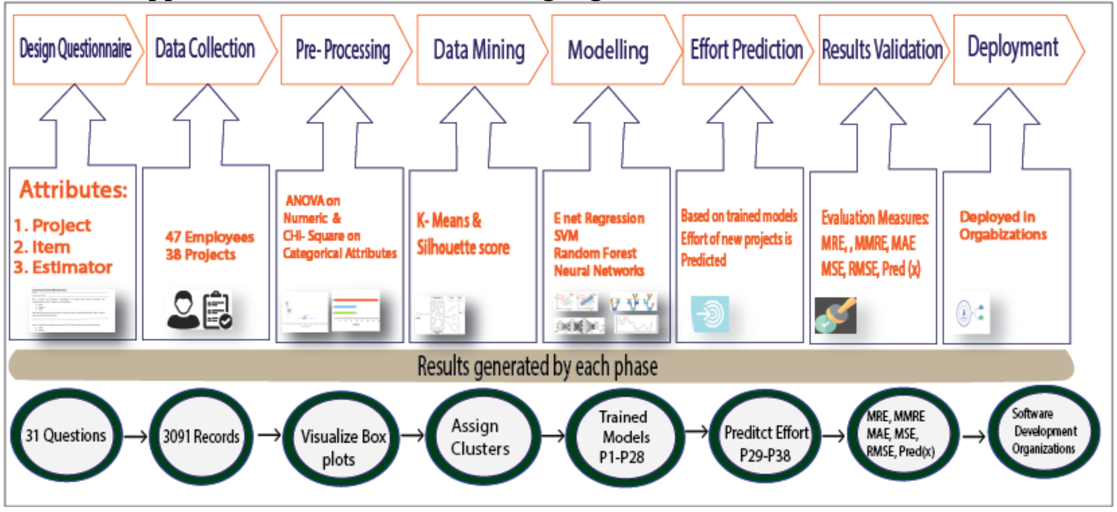


Figure 4.1: Conceptual Framework for Software Development Effort Estimation

this stage, we have produced effort estimates of project. It is very important to evaluate results generated by machine learning algorithms. Thus, the next phase is evaluation phase, where we have used measures to validate the produced results. When we were satisfied with the generated results, we have deployed this methodology to two selected organization which have provided data for this research work. Furthermore, the proposed framework is depicted in Figure 4.1.

The components of framework are explained in following sections.

4.1.2.1 Design Questionnaire

The questionnaire is designed based on 21 variables depicted in Table 4.1 used in this study. The variables are grouped into three parts as presented in Table 4.1. First one is linked with the project characteristics, second with the estimator and third one is related to work items which are extracted from work break down structure. These three types of variables were used to construct a data set. The answers of questions are given in numeric or categorical type. The design questionnaire is provided in chapter 5.

4.1.2.2 Data Collection

We gathered data from two software organizations in region of Islamabad-Pakistan. Mainly the questionnaire is based on variables which are used for making effort estimation. The variables are selected on the basis of work done by Karna et al [25].

The questionnaire contains a demographic section which contains information like name, organization name etc. which was not included in study. However, the

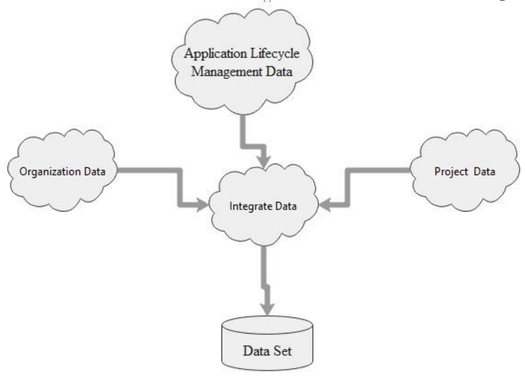


Figure 4.2: Data Collection Process

Table 4.1: Features and Target Variable used to build model

Variables	**Project**	**Estimator**	**Item**
Used as Predictor	Size	Level	Phase
	Volume	Company Experience	Area
	Development Method	Estimation Experience	Item Size
	Duration	Role and their Responsibilities	Activity
	Precedence	Total Experience	Priority
	Turnover	Technical Competence	Estimated Effort
	Complexity Estimated Effort	Organizational Competence	Severity
Target Variable		Actual Effort	

rest of project variables and the questionnaire was adopted from work of Karna et al [25].

The selected software development organizations were working on same scale and have more than 50 employees. Respondents filled these questionnaires for 38 already completed software projects which are categorized as small, medium and large. We used random sampling procedure [144] for data collection from organizations. As a result, 47 respondents (mostly software engineers and project managers) participated in survey. Combining the results of survey into a data set comprising 3091 instances collected from the data of 28 projects which were used for training. Rest of the projects P29-P38 are used for testing phase. This data collection process is given in Figure 4.2.

4.1.2.3 Pre-Processing

In this phase the variables are analyzed using ORANGE tool. The description of Orange tool has been provided in chapter 5. Following the visualization from [121] we used box plots for visualization of variables. The variables are categorized into numeric and categorical types. For numeric variables' ANOVA Test is used to calculate variance between mean of groups [145] such as Project volume, total experience, company experience, item actual effort, item estimated effort etc. variables which contains three groups of data for small, medium and large projects. However, for the categorical variables we have used Chi-Square. This test is used to find the independence of variables [146]. The results of these statistical testing show variables are significant as their p- values are less than 0.05 and therefore, they would have giant impact on estimation. We also performed Pearson correlation analysis to analyze relationship of variables. The pre-processing of each attribute that will be used for building models and eventually lead towards prediction phase are presented in results section. (referred to chapter 5) The equation to calculate degree of freedom for categorical variables is given below:

$$df = (r - 1)(c - 1) \tag{4.1}$$

4.1.2.4 Data Mining

At Data mining phase, we apply data mining technique known as clustering. The procedure of grouping elements of the basis of similarity is known as Clustering [147]. The motive behind using this technique is its similitude nature with human understanding . The primary step in K-Means Clustering is to adopt number of clusters which is represented by K. The subsequent step deals with picking centroid for each cluster. Simplest way is to select k randomly from the given data points, we may have multiple iterations to get accurate centroid for each cluster. Usually, distance metrics known as Euclidean distance is widely used in clustering phase but we have other measures too. So, we applied euclidean distance as it is used to measure the smallest distance between object and centroid to assign the cluster. Moreover, euclidean distance is the smallest distance in all dimensions. Furthermore, to examine cluster superiority and assigning quality of cluster, the projected method use Silhouette Index as presented in studies [25].

Moreover, to run K-means in Orange tool, We have select K= 3,5,7 for clustering process. The clustering process is presented in Figure 4.3. After clustering, next

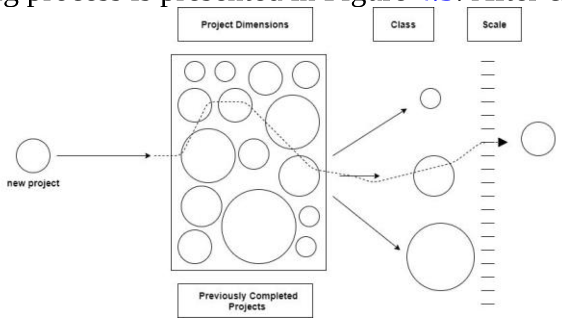


Figure 4.3: Project Clustering

phase is to apply machine learning technique to build models that would later predict effort of software projects. The results of this phase are given in 5.

4.1.2.5 Modelling

At this stage we performed experiments in two stages. First stage involve the model formation without taking input from data mining phase. At second stage we performed experimentation with the input taken from K-Means algorithm. Whenever the size of data set is large, and we have more than more of features we apply machine learning over it for prediction purpose. These techniques are used either for classification or regression purpose. We apply supervised learning techniques in proposed framework, such as Neural Networks, Linear Regression, Support Vector Machine and Random Forest using ORANGE tool. The techniques are used to train model and used for making predictions in different areas including software development effort estimation. Further in this section, we have explained the setting of machine learning algorithms and the values of parameter for running experimentation.

Algorithm 1: Support Vector Machine
In the process of performing regression using Support Vector Machine in orange tool, we have fixed the parameter values as: Cost=4.0, Regression loss to 2.80. The tolerance value and iteration limit are 0.0001 and 100 respectively. Furthermore, the polynomial kernel is used with values of g,c,d as 0.03, 0.31, 2.0. The effort estimations made by Support Vector Machine are Presented in Chapter 5.
Algorithm 2: Linear Regression
In order to run linear regression using multiple variables in orange tool. We performed different settings of parameters, but the best results were seen with elastic net regression. The value of L1, L2 and α are 0.46, 0.054 and 0.013 respectively.
The effort estimations made by Linear Regression are Presented in Chapter 5.
Algorithm 3: Random Forest
Random forest in orange tool is executed by setting no of trees, no of splits in each tree and depth of trees. Thus, we set value of parameters as: no of trees=2, no of split at each level=6, depth limit=3 and do not split trees more than=3.The effort estimations made by Random Forest are Presented in Chapter 5.
Algorithm 4: Neural Network
We have fixed values of α = 0.007 and no of iterations to 20000. Furthermore, we selected hidden layers to 4. To run forward propagation neural network, we have selected identity activation function. The effort estimations made by Neural Network are Presented in Chapter 5.
Algorithm 5: K-Nearest Neighbour
To use K-Nearest Neighbour in orange tool, we selected the uniform weighted method which assigns equal weightage to all the neighbors and selected the value of k to 3. The effort estimations made by K-Nearest Neighbour are Presented in Chapter 5.

4.1.2.6 Effort Prediction

At the effort prediction phase, we aim to test the accuracy of machine learning algorithms for new- unseen projects. Thus, we selected software projects (P29-P38). These projects contains small, medium and large sized projects. The effort estimates of these 10 projects are presented in Chapter 5.

4.1.2.7 Results Validation

At the stage of results validation, we have utilized the evaluation measures presented in chapter 2. These evaluation measures are Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Prediction (Pred), Absolute Error (AE), Mean Relative Error (MRE) and Mean Magnitude of Relative Error (MMRE) [149]. The evaluation of results for algorithms are presented in Chapter 5.

4.1.2.8 Deployment

At the last stage, we would deploy the presented model in two selected software development organizations in region of Islamabad-Pakistan for accurate effort estimation of software projects. This model was applied with the intention of improving effort estimation of new projects for reasonable allocation of resources for

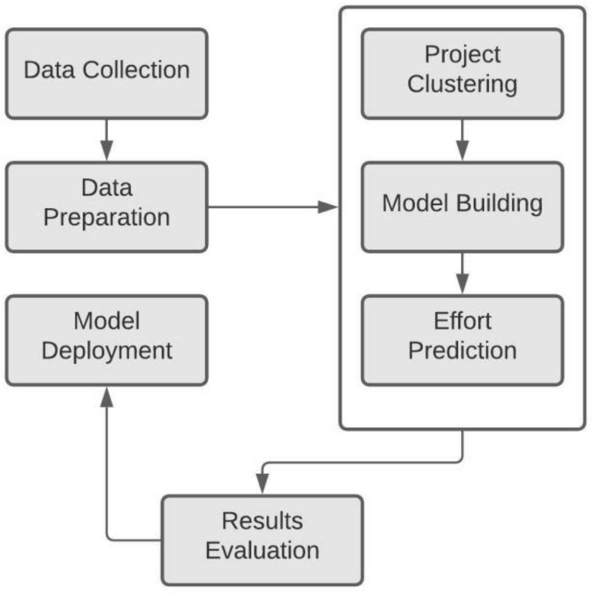


Figure 4.4: Flow of Proposed Model

successful in time completion of project.

4.1.3 Putting Model into Work

The aim of this study is to propose a framework which improve software development effort estimation in region of Islamabad-Pakistan. The proposed model uses machine learning and data mining algorithms for effort estimation of upcoming software projects. First phase of model is data collection. Although data is collected by conducting surveys in this work but here for understanding of model we have we selected a publicly available data set named as Desharnais Data Set [150] as an example. Another reason behind using this data set is that some variables of Desharnais variable and the variables used in study are common.

After the data is loaded through file widget from orange tool. We have connected the input data set to the K-means widget from clustering module. At this phase, the clusters based on similarity are produced as a result. We selected three optimum number for clusters that are 3 and 5 for this data because we have total of 81 projects. The cluster formation phase therefore, this step is very important as, these clusters are used during model building for effort estimation.

At the next stage, we have formed models using machine learning algorithms. We have used Support Vector Machine, Neural Networks, Random Forest, Linear Regression and K- Nearest Neighbour algorithms for model formation. For the application of each algorithm in orange tool we connected widgets for each of the algorithm and model is trained with setting parameter value. After the training phase is completed, we have selected three projects from Desharnais data set for testing phase which is effort prediction phase. Further, at next stage we have evaluated results using Root Mean Squared Error, Mean Squared Error, Mean Relative Error, Absolute Error, Mean Magnitude of Relative Error and Pred (x). This models evaluated is important for check whether the produced estimates are guessed by the model or they are predicted. At the last stage, we could deploy this model to the environments which are similar to Desharnais settings.

The work flow of model which is described above is presented in Figure 4.4. The work flow consists of five basic steps, first one is data collection. Next, we have applied pre-processing in data preparation phase. Once we have analysed the variables and data set. We move onto the Project clustering. After cluster formation, we build the model and Effort of new projects is predicted. Thereafter, once we are done with prediction, we evaluate the results using above mentioned measures. Last phase is deployment, in which we apply the model into software development organizations.

4.1.4 summary

This chapter has presented a framework for software development effort estimation. It provided a detailed information of model which is based on machine learning.

In the next chapter, we have explained the obtained results.

Chapter 5 Results and Discussions

The objective of this chapter is to provide the results of experiments done for prediction of software development effort estimation. We developed a model by merging data mining and machine learning techniques. For the purposes of prediction, data sets were collected from two software development organizations in the region of Islamabad-Pakistan. At the end, we have compared the results with similar study.

5.1 Description of Data set

The purpose of this section is to explain data sets which are used to conduct experimentation for software development effort estimation. Thus, we formed two data sets. One is used for training another is used for testing of models. The following subsections explain both data sets.

5.1.0.1 Data set used for Training Model

In the first phase, data is collected from two organizations separately. Then data of both organizations are combined, and consolidated data set was formed. The data set contain project related information. The variables are either numerical or categorical. (See Table 5.1 for variables and their data types)

Overall data set consists data of 38 already completed projects from two software development organizations. We have used 28 projects (P1-P28) for training purpose and four software development projects (P29-P38) for testing the performance of model.

Combing the information of each single task for all projects we formed a data set of 3091 records. The software projects utilized for training contains 9 small sized projects, 13 medium and 6 large projects. The classification of project size is based on organizations internal documentation. The projects of small size are within the range of 1-50 hours, medium sized projects are between 50-102 hours and large projects are in 102+ hours. The Actual Effort for all three classes is given in Table 5.2. The variables are adopted from work done by H karna et al [25]. Consequently, the data set contains information in three parts, first one is linked to the project which contains information of project such as Size (Small, Medium & Large), Volume (Implementation workhours), Duration (Short, Medium & Long),

Complexity (Nominal, High & Very High), Development Method (Iterative & Sequential), Precedence (True & False) and Turnover (None or Low, Medium and High). Second part is linked to Estimator related variables such as Role and

Table 5.1: Categories of Variable

Type	Categorical	Numeric
Item	Phase Activity Area Severity Completion	Item Size Priority Estimated Effort Actual Effort
Estimator	Role and their &responsibilities Organizational Competence Technical Competence Estimation Experience	Level Total Experience Company Experience
Project	Size Duration	Actual Effort Estimated Effort

	Turnover Complexity Development Method	Volume

their Responsibilities (Software Engineer, Project Manager, Solution Architect, Quality Manager & Configuration Manager), Level (Junior, Advanced & Expert), Experience divided further into Company Experience, Total Experience and Total Experience all expressed in numeric values.

Then the competence of estimator is again divided into Organizational and Technical competence expressed in categorical values (Basic, Intermediate, Advanced & Expert). Finally, the last attribute is related to Item (Each single task till project completion) such as Phase (Initiation, Definition, Design, Implementation. Operation, & Termination), Activity (Design, Quality, Management, Documentation, Implementation, System Test, Configuration, Installation and Integration & Acceptance), Area (Project Management, Configuration Management, Documentation, System & Quality Management), Item Size (Small, Medium Large and Very Large), Priority (True & False), Severity (Low, Medium & High), Estimated Effort and Actual Effort represented with numeric values in work-hours [h]. Furthermore, we have summarized description of all variables in Table 5.3.

The project used for clustering and then model building consists of 28 projects. The actual and estimates made by Estimators of these projects are presented in bar chart in Figure 5.1, where red lines show project estimate effort and blue lines show project actual effort.

Table 5.2: Project Size w.r.t Work-hours

Project	Effort in Work-hours[h]
Small	1+50
Medium	50-102
Large	102++

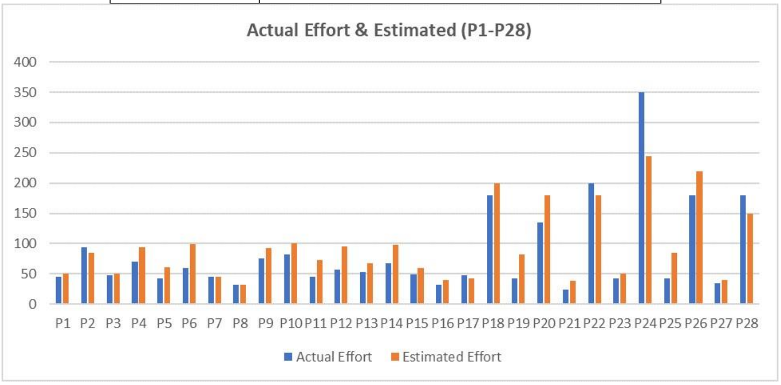


Figure 5.1: Actual and Estimated Effort of projects used in Training phase

Table 5.3: Description of Variables.

Variable	Sub-Variable	Description

Project	Size	The size of software projects represented as categorical value which is defined by organizational internal rules. Type: Categorical; Values: [Small, Medium and Large]
	Volume	The applied effort in implementation phase for project completion. Type: Numeric; Values: [Hours]
	Actual Effort	The actual effort which is written down after project is completed Type: Numeric; Values:[Hours]
	Estimated Ef-fort	The estimated effort which is provided by project managers or estimators at the start of project. Type: Numeric; Values:[Hours]
	Duration	Overall time required the complete a project.

Continued on next page

Table 5.3 – continued from previous page

Variable	Sub-Variable	Description
		Type: Categorical; Values: [Small, Medium and Large]
	Turnover	Extent to which project members leave in between with respect to project team size. Type: Categorical; Values: [None: Low; Medium and High]
	Precedence	Highlight if any similar project is present. Type: Categorical; Values: [True, False]
	Complexity	Identifies the difficulty level of project. Type: ordinal; Values: [Normal; High and Very High]
	Development Method	Method adopted to complete software project. Type: Categorical; Values: [Sequential, Iterative]
Item	Activity	what type of activity associated with each item. Type: Categorical; Values [Documentation, Quality, Implementation, Management, Test, Design, Acceptance]
	Phase	The phase with which item is linked to. Type: Categorical; Values: [Initiation, Definition, Design, Implementation, Operation and Termination]

Table 5.3 – continued from previous page

	Area	area associated with each item. Type: Categorical; Values: [Documentation, Project Management, Configuration Management, Quality Management, Installation, Integration, and System]
	Priority	The sequence of execution for all items.

Continued on next page

Variable	Sub-Variable	Description
		Type: Numeric; Values: [1,2,3]
	Severity	The impact of one item over other items. Type: Categorical; Values: [Low, Medium, High]
	Estimated Effort	The effort estimates associated with Work Items before starting a project. Type: Numeric; Values:[Hours]
	Similarity	Information related to similar work that has already been done. Type: Categorical; Values:[None, Low, Medium and High]
	Actual Effort	The effort estimates associated with Work Items after completing a project. Type: Numeric; Values:[Hours]
	Item Size	Work hours associated with each item. Type: Categorical; Values: [Small, Medium and Large] Size up till 8 [h] are labelled as small Size up till 8-16 [h] as Medium, size up till 16-32 [h] as Large

Table 5.3 – continued from previous page

Variable	Sub-Variable	Description
		Type: Categorical; Values: [Junior, Advanced and Senior]
	Organizational Competence	Competence level of estimator within an organization. Type: Categorical; Values: [Basic, Intermediate, Advanced and Expert]
	Estimator's Experience	Experience in estimation given in years Type: Numeric; Values: [Years]
	Total Experience	Total employment years. Type: Numeric; Values: [Years]
	Technical Competence	Extent of technically Competent Type: Categorical; Values: [Basic, Intermediate, Advanced and Expert]
Estimator	Role and their Responsibilities	Responsibility given to all people associated to project. Type: Categorical; Values: [Software Engineer, Project Manager, Quality Manager, Configuration manager, Solution Architect]
	Organization's Experience	work experience of estimator within the organization. Type: Numeric; Values: [Years]
	Level	Rank given to estimator with respect to experience.

Continued on next page

In the next section, we would show the results of statistical testing applied on data set to analyze characteristics for further model building phase.

5.1.1 Data set used for Testing Model

Table 5.3 – continued from previous page

The data set used in testing phase consists of software projects from P29- P38. Three projects P31, P36 & P38 are categorized as small sized projects. Four medium sized projects (P30, P32, P35 & P37) and three large projects (P29, P33 & P34) are used for testing machine learning algorithms.

In the data set used for testing the information related to Project actual effort, Item size, turnover, Item actual and estimated effort are not provided. So that model is tested over actual data that is provided at early stages of project. The input provided to model is based on Estimators.

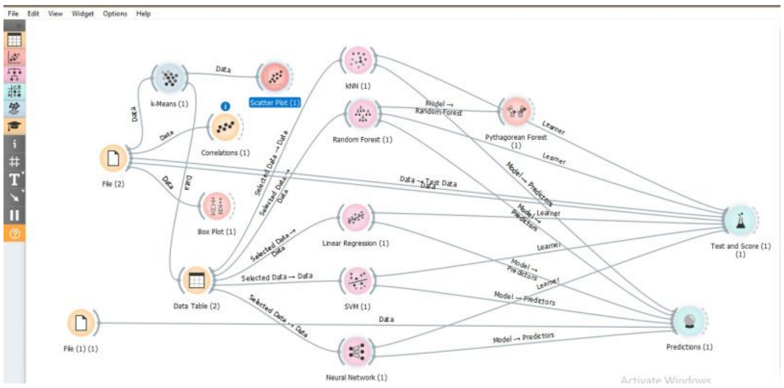


Figure 5.2: Orange tool Model Formation

5.2 Implementation Details

Orange is a visual programming tool based on Python 3 data mining library. After data collection, the first step visualization and then application of programming models for building predictive models, we have adopted an open source software named as ORANGE Tool [143]. It contains user defined components known as Widgets. These widgets contain all the function which are required for model building and evaluation purpose. Further, the selected component-based framework is preferred by academicians and researchers [148].

We have performed the experiments in steps. For the analysis of data set, we have used Chi- Square, Anova and Pearson correlation analysis. Further, at next stage, we have used K-Means clustering, Support Vector Machine, Neural Network, Random Forest, Linear Regression and K-Nearest Neighbour. The application of these algorithms and model formation can be visualized in Figure 5.2.

In addition, we performed this experimentation with processor: Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.20 GHz with 8.0 GB RAM and 64-bit operating system, x64-based processor.

5.3 Results of Pre-processing

In this phase, we applied two tests over data sets, first one is Chi-Square which is applied for categorical variables to analyze their degree of freedom [151, 146]. However, to analyze the variation between means of two or more groups, we applied ANOVA test [152, 147]. The threshold p – value set is 0.05. If the value of variable is less than 0.05, this indicates their significance. The results and visualization of

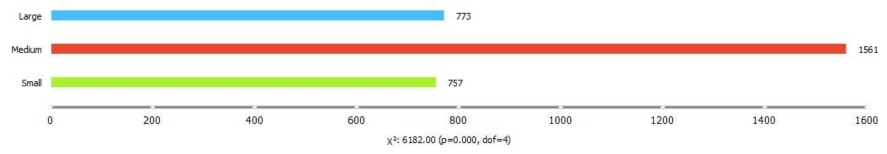


Figure 5.3: Project Size

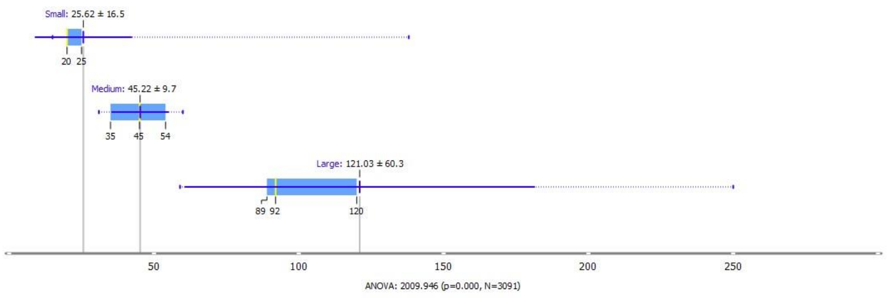


Figure 5.4: Project Volume

statistical testing is given in Box plots in this section. Figure 5.3- Figure 5.28 represents box plot for each test.

5.3.0.1 Project Related Variables

The first variable in this category in Project Size. Project size is divided into three groups small, medium and large based on total number of work hours. The project size is categorical variable and Chi-square is applied on it. The value of chi is 6182.00. however, the degree of freedom is calculated as 4. The Figure 5.3 represents the number of as small, medium and large from the work-items which were used in this study. There are 773 work-items of large projects. 1561 from medium sized and 757 for small projects. Project volume (See Figure 5.4)is another variable linked to project related variables. This is numeric type variable and Anova is applied to compare the mean between groups. There are three formed groups are small, medium and large. The p-value is 0.00 for N=3091.

Another variable used in this study is Project Duration, which is categorized in three groups as short, medium and large. The calculated degree of freedom is 4. This variable is used to analyze the duration of already completed projects. The project duration variable is represented in Figure 5.5. Project Actual Effort is the most important numeric type variable. This variable is used to allocate clusters to projects. The difference between mean of groups is calculated using ANOVA test. The p value of Anova is 0.00. Later we use this variable to analyze the difference

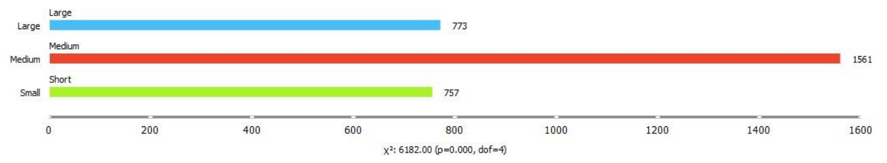


Figure 5.5: Project Duration

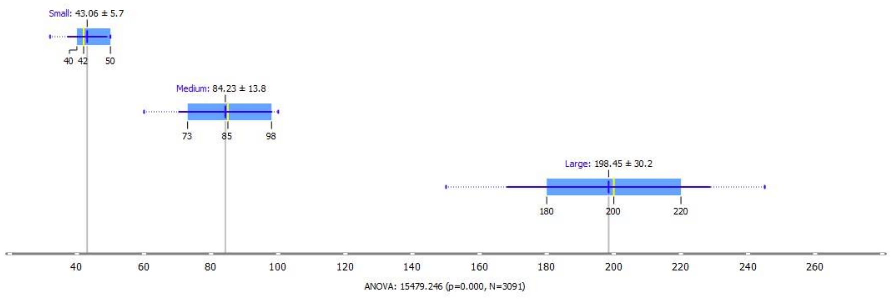


Figure 5.6: Project Actual Effort

between actual and estimated effort of a project. The box plot of project actual variable is presented in Figure 5.6.

Another variable used for allocation of cluster is Project Estimated Effort. This variable is used to calculate difference between actual and estimated variable. The difference between three groups is calculated and represented in Box plot (See Figure 5.7).

5.3.0.2 Item Related Variables

The second type or variables used in this study are Item related variable. Item as already defined are small tasks used to execute a software development project. These items combine to form a project. The variables associated to each item are explained in this section.

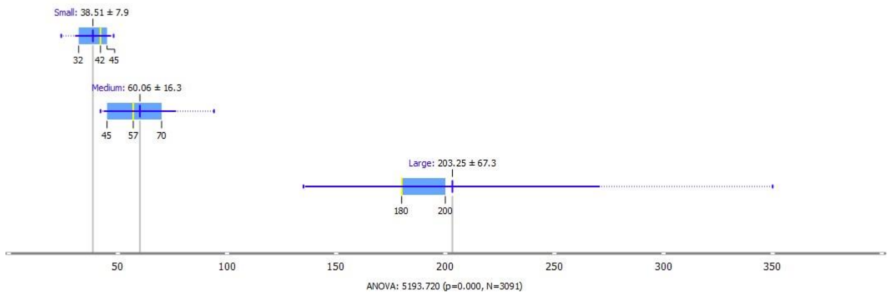


Figure 5.7: Project Estimated Effort

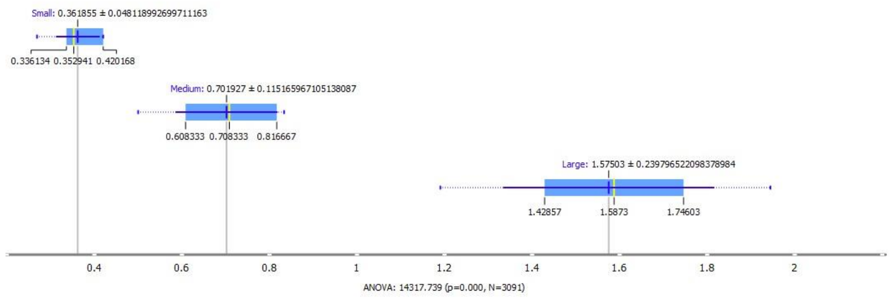


Figure 5.8: Item estimated effort

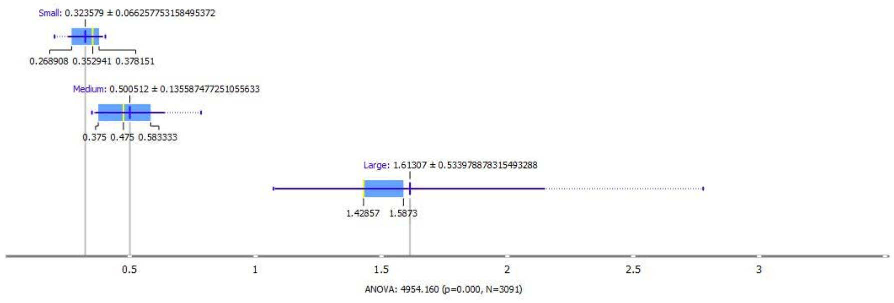


Figure 5.9: Item Actual Effort

Item estimated effort is a numeric type variable for three groups small, medium and large. The difference between each group is shown in Figure 5.8. The Anova test calculates P-value of item estimated effort as 0.00. This variable is used for clustering process. Second type of variable related to Item is Item actual effort. The variable is numeric type variable having P- value of 0.00 calculated by performing Anova test. The variable is also used in clustering process. The difference in groups is calculated and represented in Figure 5.9. Project Item Size is a categorical variable. The variable is used as a predictor for model building phase. Chi-value for this variable us calculated as 5490.05. the box plot representation is given in Figure 5.10. The item is grouped in three categories based on the work hours. Item within range of 1-8 hours is marked as

small, while between 8-16 hours is considered as medium sized item. However, item having more than
16 hours is considered as Large sized item. Development method of software

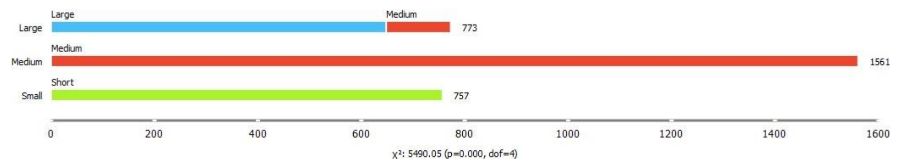


Figure 5.10: Item Size

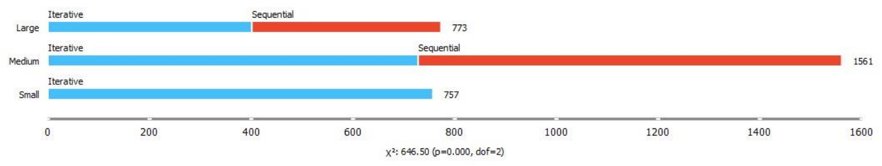


Figure 5.11: Item Development Method

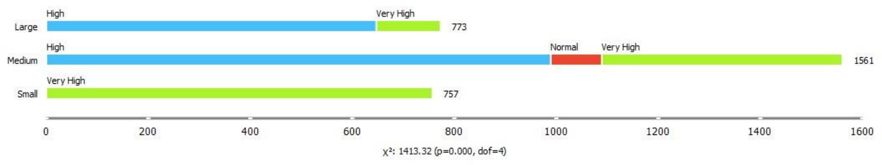


Figure 5.12: Complexity

projects is divided into two categories. These are Iterative and sequential. The projects of three categories small, medium and Large are completed using any one methodology. The chi- vale for this variable is calculated as 646.50 and degree of freedom is noted as 2. The box plot representation of the variable is presented in Figure 5.11.

Software development projects based on their complications is considered as complexity. Since the software projects are different from each other so there is a chance of difficulty in completing project successfully. The variable complexity is divided into three groups i.e. normal, high and very high. There are three categories of projects as already explained in above sections. The box plot representation for each group and their complexity is represented in Figure 5.12. The chi vale for Project complexity is 1413.32 and degree of freedom is calculated as
4.

Project Turnover is another variable. This means the percentage employee leave or change their project during execution. The type of project turnover is categorical. This variable is very important as it depends on the individual resource working on project. The employee turnover is considered for each project and their work item. The chi value for this variable is 923.68 with degree of freedom 4. The box plot representation for this variable is provided in Figure 5.13.

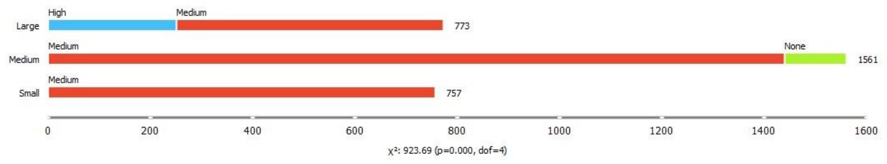


Figure 5.13: Turnover

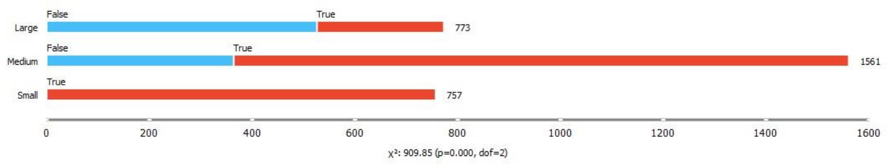


Figure 5.14: Precedence

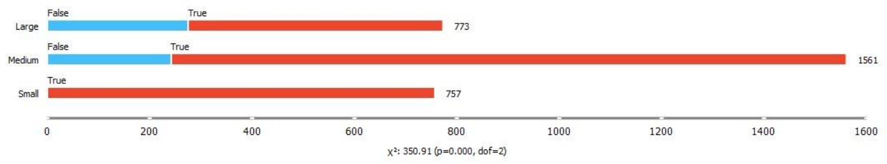


Figure 5.15: Completion

Precedence is another categorical variable associated to Item. This variable is used to understand the similarity between new item and previously available items. There two categories are True and False. The chi-value for precedence is 909.85 with degree of freedom as 2. The box plot for precedence is given in Figure 5.14.

Another variable associated to item is Completion this variable identifies whether the item is completed within estimates of not. If the item takes longer duration then estimated time, then we mark it as False otherwise it is True. The item completion is considered as categorical variable with chi value 350.91 and degree of freedom 2. The box plot representation for all projects is given in Figure 5.15.

Work items are associated to another variable known as Phase. The area of a categorical variable divided in 5 categories. These are Definition, Design, Implementation. Installation, Operation. These categories are considered for all work items in this study. The chi value of completion is 1237.16 and degree of freedom is calculated as 8. The box plot representation for Phase is given in Figure 5.16. The area associated with item are Configuration Management, Project Management, Documentation, System, Quality Management. These areas are together form a variable known as Area which is associated to item. This variable is a

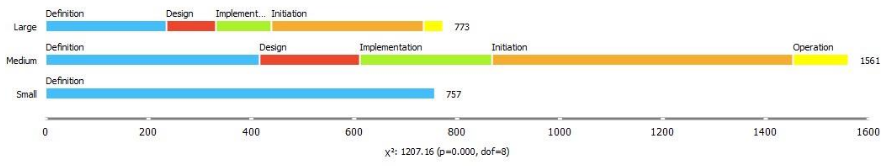


Figure 5.16: Phase

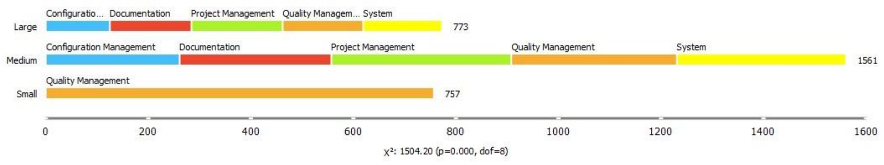


Figure 5.17: Area

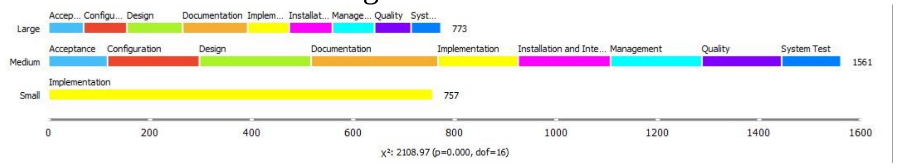


Figure 5.18: Activity

categorical variable with chi value of 1504.20 and degree of freedom as 8. the box plot representation for this variable is given in Figure 5.17.

Project Item is linked to another variable known as Activity. This is a categorical type variable with Design, Quality, Management, Documentation, Implementation, System Test, Configuration Installation and integration and Acceptance as categories. The chi value calculated for this variable is 2108.98 with degree of freedom 16. The box plot is provided in Figure 5.18.

Based on the importance of item, a number is allocated to each item. Numbers from 1-3 are assigned to item. The number 1 indicates the work item is most important and should be executed at first and 3 shows the least important item. Thus, priority is numeric variable. The between mean of three groups of projects is calculated using Anova test. The test resulted p value of 0.109. Box plot representation of Priority is given in Figure 5.19.

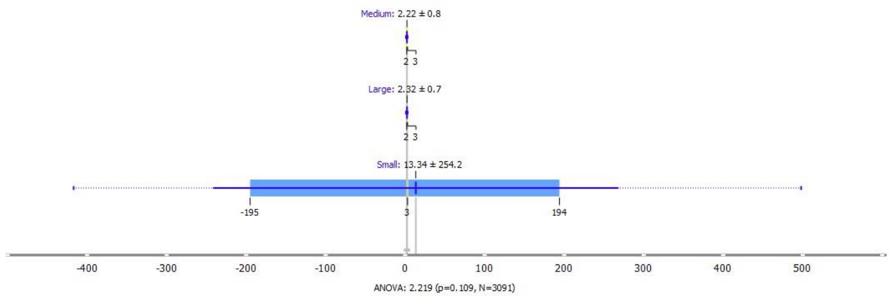


Figure 5.19: Priority

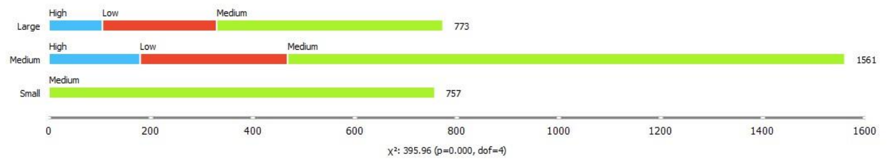


Figure 5.20: Severity

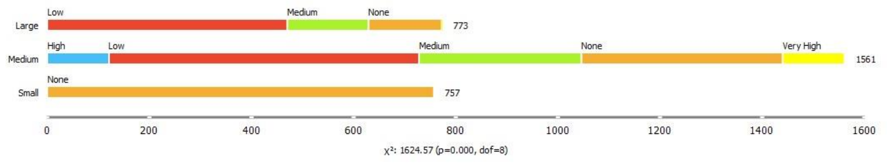


Figure 5.21: Similarity

Another variable related to item is its Severity. This categorical type variable has assigned three categories Low, Medium and High. The chi-value for severity is calculated as 395.96 and degree of freedom as 4. The box plot is given in Figure 5.20. The last variable related to item is Similarity. The three categories are Low, Medium and High. These are assigned to each work item. The similarity is checked on three bases i.e. Low, Medium and High. The categorical variable is tested using chi-square and chi-value is 1624.57 while degree of freedom is 8. The box plot is given in Figure 5.21.

5.3.0.3 Estimator Related Variables

The third kind of variable is linked to the estimator. Estimator is the one who assigns effort estimated to project. The first categorical variable in this type is Role and their responsibilities. This variable used to analyze the responsibilities of Software engineer, project Manager, Quality manager and other people. The chi value is 1772.08 and degree of freedom is calculated as 8. The box plot representation is provided in Figure 5.22.

Another variable associated with estimator is level. The estimator might belong to Junior, Advanced and Senior category. The chi value for Level is 2134.04 with

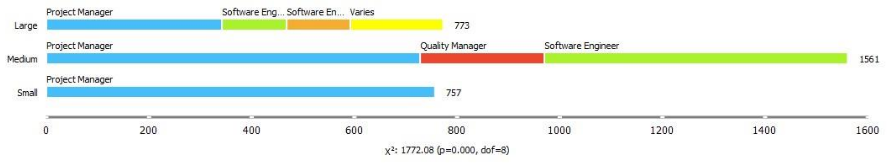


Figure 5.22: Role and their responsibilities

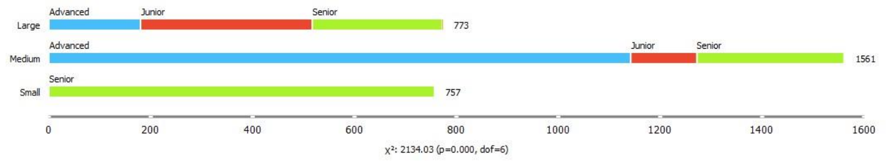


Figure 5.23: Level

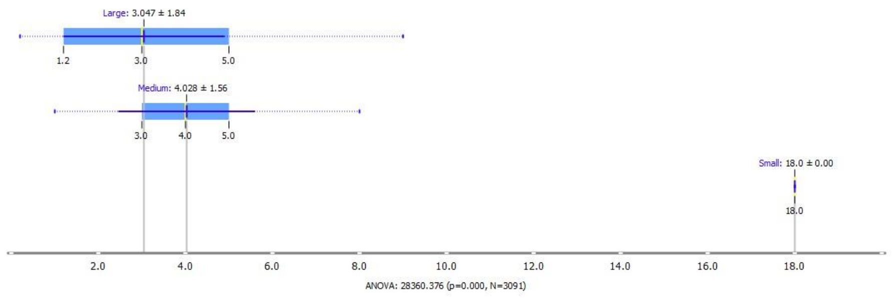


Figure 5.24: Total Experience

degree of freedom 6. The box plot is presented in Figure 5.23.

Experience of estimator is one of the important variables. This is numeric type variable. The difference between groups of three projects for total experience of estimated is represented in box plots (See Figure 5.24). The p value is 0.00. Then another variable known as Organizational Experience is considered as numeric type variable. The difference between groups of is plotted in Figure 5.25. The p value as a result of Anova test is 0.00.

Another numeric variable is known as Estimation experience. The total experience of estimator in making estimates of effort is considered in this variable. The box plot represented in Figure 5.26 shows p value of 0.00.

Technical Competence (See Figure 5.27) is a categorical variable linked estimator.

The categories of Technical Competence are Basic, Intermediate, Advanced and

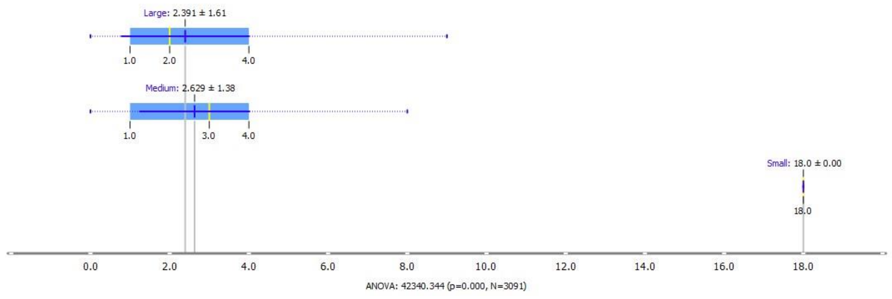


Figure 5.25: Organizational Experience

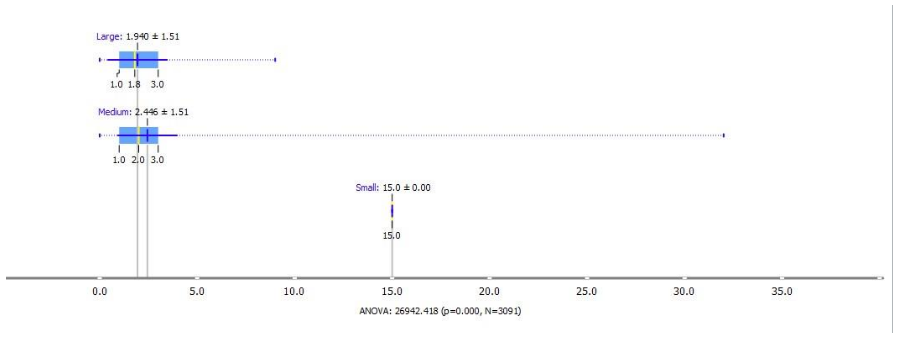


Figure 5.26: Estimation Experience

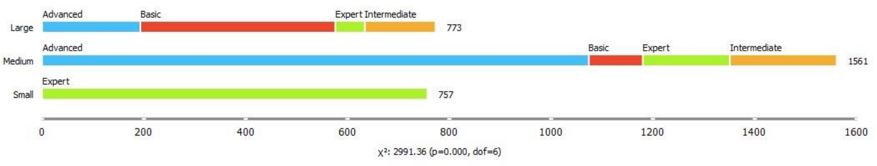


Figure 5.27: Technical competence

Expert. The chi value is calculated as 299.36 for degree of freedom as 6.

Organizational Experience (See Figure 5.28)is a categorical variable linked estimator. The categories of Organizational experience are Basic, Intermediate, Advanced and Expert. The chi value is calculated as 32293.83 for degree of freedom as 8.

5.3.1 Correlation Analysis

Correlation analysis is performed to analyses the relationship between variables. The target variable in this study is project estimated effort. Therefore, we apply Pearson correlation to analyses the impact of variables on project estimated effort.

Pearson Correlation is used to identify three relationships such as no relation, positive or negative correlation as stated in work of Kim S et al. [148] The value closer to 1 indicate positive relation, value near to -1 indicate negative relation.

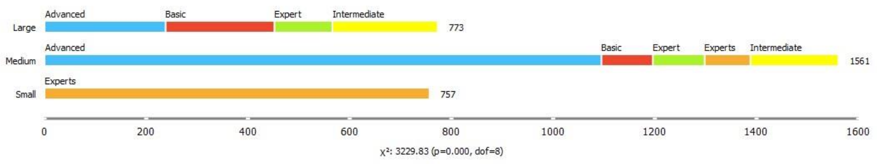


Figure 5.28: Organizational Competence

Table 5.4: Pearson Correlation Analysis

Variable 1	**Variable 2**	**Variable 3**
Project Estimated Effort	Item Estimated Effort	+1.000
	Item Actual Effort	+0.918
	Project Actual Effort	+0.921
	Project Volume	+0.921
	Estimation Experience	-0.434
	Total Experience	-0.440
	Organizational Experience	-0.404
	Priority	-0.036

However, value close to 0 indicate no relation. Table 5.4 shows the relation between these variables.

The Table 5.4 above shows variables names as Item estimated effort, Item Actual Effort, Project Actual Effort and Project Volume are positively correlated to Project Estimated effort. However, variables named as Estimation Experience, Total Experience, Priority and Organizational Experience are slightly negatively correlated.

5.4 Experimental Results

5.4.1 Project Clustering

In this study, we applied K-Means Clustering for grouping similar projects in one group. The results of Clustering phase prior to model building phase is provided in Table 5.5. The project items (also known as tasks) are combined and Silhouette score for each project is calculated to analyze the quality of items in a project. Then, the projects are grouped in clusters based on their similarity.

This table represents clusters assigned to projects and their silhouette scores. Values closer to 0 indicate poor quality while value closer to 1 indicate good quality projects which are far from their neighboring clusters.

The results of clustering phase also show the projects categorized as small are grouped in one cluster C3, medium in C1 are Large projects in C2 cluster when K=3. As soon as we increase value of K by 5, we noticed projects are divided into 5 groups or clusters. All other projects remain in same cluster however, project no
24. is a part of separate cluster-C5. In another iteration, value of K is changed to 7. Consequently, we noticed formation of two more clusters C6 and C7. Further grouping shows the division of project of smaller size are again divided. However,

Table 5.5: Cluster and Silhouette score of each project

When K=3			**When K=5**			**When K=7**		
P.ID	**Cluster**	**Silhouette Score**	**P.ID**	**Cluster**	**Silhouette Score**	**P.ID**	**Cluster**	**Silhouette Score**
2	C1	0.61	1	C1	0.67	1	C1	0.67
4	C1	0.60	3	C1	0.69	3	C1	0.69
5	C1	0.62	7	C1	0.68	7	C1	0.69
6	C1	0.63	8	C1	0.68	8	C1	0.68
9	C1	0.61	16	C1	0.62	16	C1	0.62
10	C1	0.62	17	C1	0.68	17	C1	0.60
11	C1	0.63	2	C2	0.58	21	C1	0.63
12	C1	0.63	4	C2	0.60	23	C1	0.69

13	C1	0.62	5	C2	0.59	2	C2	0.54
14	C1	0.63	6	C2	0.61	4	C2	0.53
15	C1	0.63	9	C2	0.60	5	c2	0.53
19	C1	0.62	10	C2	0.62	6	C2	0.65
25	C1	0.61	11	C2	0.62	20	C3	0.55
18	C2	0.55	12	C2	0.61	26	C3	0.57
20	C2	0.56	13	C2	0.63	28	C3	0.59
22	C2	0.58	14	C2	0.62	9	C4	0.54
24	C2	0.61	15	C2	0.61	10	C4	0.55
26	C2	0.60	19	C2	0.61	11	C4	0.55
28	C2	0.57	25	C2	0.61	12	C4	0.57
1	C3	0.64	18	C3	0.56	13	C4	0.58
3	C3	0.65	20	C3	0.57	14	C4	0.58
7	C3	0.67	22	C3	0.55	15	C4	0.56
8	C3	0.68	26	C3	0.54	19	C4	0.57
16	C3	0.69	28	C3	0.58	25	C4	0.56
17	C3	0.68	21	C4	0.65	24	C5	0.63
21	C3	0.67	23	C4	0.69	18	C6	0.59
23	C3	0.54	27	C4	0.68	22	C6	0.56
27	C3	0.64	24	C5	0.65	27	C7	0.68

Project 24 remains in the same Cluster, C5while projects 18 and 22 are part of C6. Further division shows project no 27 is a part pf new formed cluster, C7.

However, the Silhouette score calculated for each cluster is presented in Table 5.6. This silhouette score for each cluster is less than 1 this indicated the formed clusters are highly cohesive and thus far away from their neighboring clusters. Thus, the formed clusters are highly cohesive and hence would have a huge impact on effort prediction. Whenever a new project comes, it is placed inside the cluster which has similar characteristics. Therefore, for testing purpose we selected four projects P29-P38.

The results of clustering over all taken projects is given in Figure 5.30. These Figures also shows projects within a same cluster are closer to each other.

5.4.2 Relationship between Cluster Quality & No of Items

The projects P1-P29 are used to form clusters in this research work. To analyze the quality of cluster and their number of items we plotted a line graph (see Figure 5.29). This x- axis represents number of Clusters, y- axis shows no of items

Table 5.6: Caption

Value of K	**Silhouette Score**
K=3	0.409
K=5	0.395
K=7	0.302

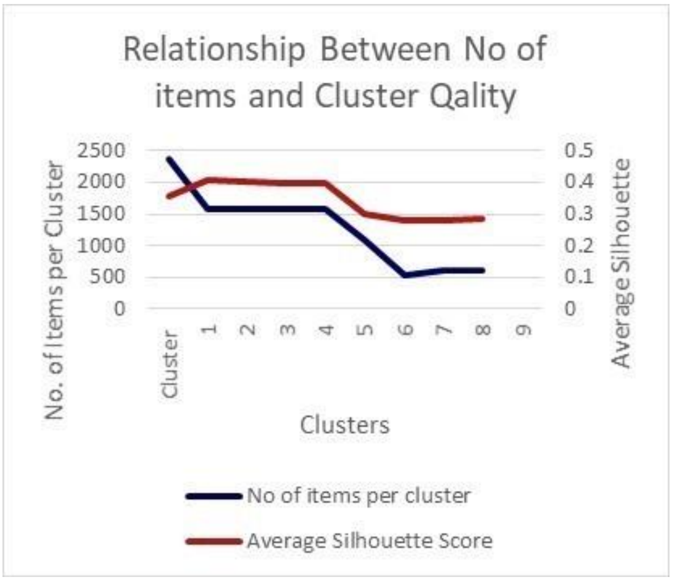


Figure 5.29: relationship between no of clusters and average silhouette score

and third dimension is a representation of Average Silhouette score. The graph represents increase in number of clusters would decrease number of items and ultimately the quality of formed clusters is reduced. Thus, the optimal number of clusters are K= 3,5,7.

Where red line indicates the relationship between no of clusters and average silhouette score. The blue line indicates towards the relationship between Cluster and their number of items.

In the next phase we utilized the results of clustering to form model over projects P1-P28. .

5.4.3 Effort Modelling

This phase takes the results of clustering phase. The projects within clusters are used to build model for making effort estimates. The projects P29-P38 are used to validate the built models. The projects and their neighboring projects are given in Table 5.7.

This table shows number of items of projects (P29-P38) and Total number of projects in data set. These project number of items plays very significant role in model building as they defined the quality of clusters. We performed modelling over Orange tool. The model building phase uses P1-P28 projects and 21 variables as predictors and one target variable named as actual effort.

The project P29, P22 and P34 are categorized as larger project having 135 items, projects P30, P32, P35 & P37 are categorized as Medium sized projects having 45 items. Finally, P31, P36 & P38 having 30 items are categorized as Smaller sized project. When K= 3, the projects with characteristics of P29 are P28, P26, P22 & P18. Further, these projects collectively contain 540 items. However, for K=5 the neighboring projects are reduce to single project P26 forming 135 items. Moreover, when K in increased to value of 7, the neighboring project is P22 with 135 work items. Similarly, medium sized projects used for testing are P30, P32, P35 & P37. When K= 3, the projects near to P30 are P25 with 90 work items. However, when we change value of K to 5 the close project us P25 with 56 work items. Further when value of K is 7, there is no project closer to it. Furthermore, P32 has closer projects P11, P12, P13, P14 & P19 with 280 work items for K=3. Changing value of K from 3 to 5, we see a reduction in number of projects to one project i.e. P25 with 56 work items. For K= 7, we see no closer project to P32. More, for project P35, when K= 3, the closer project is P10. Further changing value of k to 5 and then 7, we see P25 and P18 as closer projects having 56 work items. For project P37, we see closer projects as P5, P11 & P15 with168 work items. However, when we change value of k to 5, we noticed only P15 as closer projects with 56 items and no closer project when K=7.

Table 5.7: Closest Projects and Items in data set

Projects		**K=3**		**K=5**		**K=7**	
Project	**Items**	**data set Projects**	**Items**	**data set Projects**	**Items**	**data set Projects**	**Items**
P29	135	P28, P26, P22, P18	540	P26	135	P22	135
P30	45	P9, P2	90	P25	90	–	–
P31	30	–	–	–	–	–	–
P32	56	P11, P12, P13, P14, P19	280	P25	56	–	–
P33	135	P18	135	P28, P22	270	–	–
P34	135	P20	135	P26	135	P18	135
P35	56	P10	56	P25	56	P18	56
P36	30	P4	30	P11, P19	60	–	–
P37	56	P5, P11, P15	168	P15	56	–	–
P38	30	–	–	P13	30	–	–

The three projects P31, P36 & P38 are small sized projects used for testing purpose. For the project P31, we see no closer projects for K = {3,5,7}. However, for P36 we see P4 nearer to it with 30 work items when K= 3. However, for K =5 we see P11 & P19 as closer projects with 60 work items. Further changing k to 7.

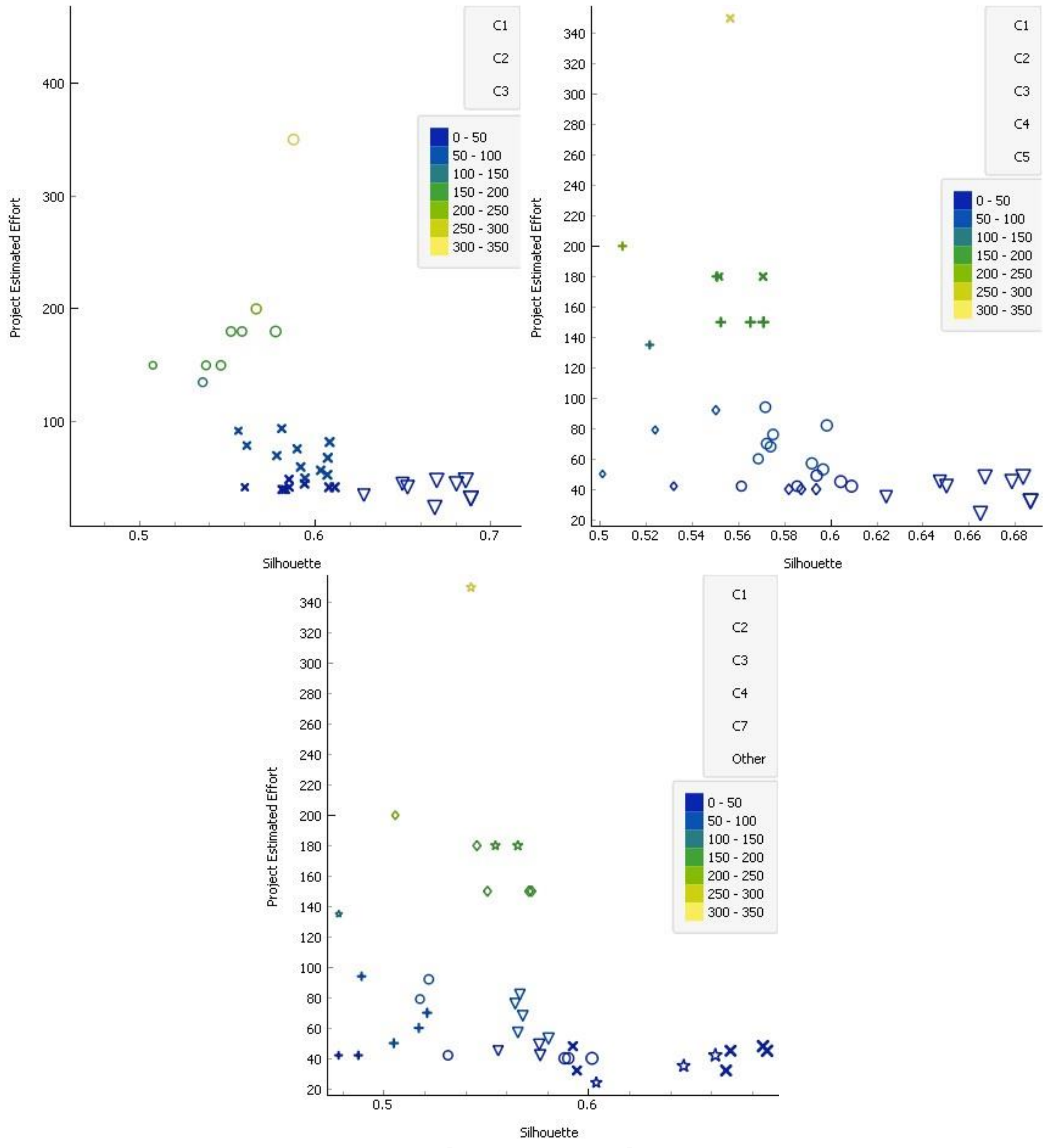


Figure 5.30: Clusters assigned to Projects

We see no nearer projects to P31. For project P38, we see one project closer to it for K=5 only. The closer project is P13 with 30 work items.

Thus, the optimal number of clusters for model building phase are K= {3, 5} and in some cases K=7. Increasing number of clusters would affect modelling process badly and ultimately would not generate accurate predictions. In the next phase, we would represent the results of prediction phase.

5.4.4 Effort Prediction

This phase generates results based on training. The projects P29-P38 are used for prediction and testing the presented model. In this phase, the effort of an actual projects is estimated using Machine Learning models such as Random Forest, Support Vector Regression, Linear Regression, Neural Networks and K- nearest neighbor. Table 5.8 contains the actual and predictions for projects P29-P38. At the end these estimates are compared with estimated made by above mentioned machine learning algorithm. .

Table 5.8: Actual and Estimators estimation

Project	**Estimators**	**Actual Effort**

P29	115.0	150.0
P30	52.0	79.0
P31	22.0	40.0
P32	72.0	65.0
P33	280.0	180.0
P34	120.0	193.0
P35	60.0	90.0
P36	30.0	45.0
P37	62.0	80.0
P38	28.0	44.0

5.4.4.1 Prediction before Clustering

In this phase, predictions made by machine learning algorithm such as Support Vector Machine, Neural Networks, Random Forests, Linear Regression and KNearest Neighbor before application of K- Means Clustering are presented. The algorithms prediction of effort for software development projects is given in Table

Table 5.9: Effort Prediction with Machine Learning

Project	SVM	LR	NN	KNN	RF
P29	144.0	129.0	97.0	185.0	185.0
P30	66.0	77.0	89.0	68.0	64.0
P31	52.0	75.0	89.0	42.0	64.0
P32	78.0	89.0	89.0	42.0	124.0
P33	148.0	163.0	113.0	154.0	180.0
P34	167.0	164.0	104.0	180.0	180.0
P35	66.0	91.0	95.0	57.0	82.0
P36	81.0	81.0	95.0	154.0	42.0
P37	81.0	82.0	97.0	154.0	68.0
P38	86.0	92.0	94.0	144.0	92.0

5.8. The estimated effort of projects P29-P38 for each machine learning algorithm before clustering is presented in Table 5.9. The predictions made by Support Vector Machine before prediction for P29-P38 are 144.0, 66.0, 52.0, 78.0, 148.0, 167.0, 66.0, 81.0, 81.0 and 86.0. Similarly, Linear Regression predicts effort for these projects as 129.0, 77.0, 75.0, 89.0, 163.0, 164.0, 91.0, 81.0, 82.0 and 92.0. Moreover the predictions made by NN and KNN for these ten projects of small, medium and large size are 97.0, 89.0, 89.0, 89.0, 113.0, 104.0, 95.0, 95.0, 97.0, 94.0 and 185.0, 68.0, 42.0, 42.0, 154.0, 180.0, 57.0, 154.0, 154.0, 144.0. Finally, the predictions made by KNN are 185.0, 64.0, 64.0, 124.0, 180.0, 82.0, 42.0, 68.0 and
92.0. These estimates are used determine the effect of clustering on predictions.

5.4.4.2 Prediction after Clustering

The predictions made by machine learning algorithms such as Linear regression, Support vector machine, neural networks, random forest and k nearest neighbor after application of K-means clustering algorithm over data set are presented in this section.

5.4.4.3 Support Vector Machine

The prediction made by Support Vector Machine after application of K-Means clustering are presented in this section (referred to Table 5.10). The predictions made for Projects P29-P38 by Support vector machine with K=3 are 154.0, 62.0, 78.0, 57.0, 157.0, 187.0, 65.0, 78.0, 82.0 and 92.0. Moreover when value of K was changed to 5 i.e. K=5, we get effort predictions as 15.0, 65.0, 73.0, 60.0, 166.0, 184.0, 68.0, 81.0, 92.0 and 92.0. Furthermore, with an iteration we reached K=7 with effort predictions as 150.0, 65.0, 73.0, 58.0, 158.0, 185.0, 67.0, 81.0, 82.0 and 92.0. These predictions are made using K = {3,5,7}. We have noticed the estimates remain same for P29, P36 & P38 for K= {5,7}. However, the predictions are different for other projects. We noticed a slight change in predictions for all values of K.

5.4.4.4 K-Nearest Neighbour

The effort predictions produced by K-Nearest neighbor after clustering is performed are presented in this section. We selected K= {3,5,7} as optimal number of clusters for model building and evaluation phase. The predictions using KNearest Neighbour are given in Table 5.11. The prediction made in case of K=3 with KNN for Projects P29-P38 are predicted as 82.0, 68.0, 42.0, 42.0, 180.0, 180.0, 42.0, 42.0 and 42.0. With an iterations, we set K=5 and predicted effort

Table 5.10: SVM for K=3,5,7

Cluster	**K=3**	**K=5**	**K=7**
Project	**SVM**	**SVM**	**SVM**
P29	154.0	150.0	150.0
P30	62.0	65.0	65.0
P31	78.0	73.0	73.0
P32	57.0	60.0	58.0
P33	157.0	166.0	158.0
P34	187.0	184.0	185.0
P35	65.0	68.0	67.0
P36	78.0	81.0	81.0
P37	82.0	92.0	82.0
P38	92.0	92.0	92.0

Table 5.11: KNN for K=3,5,7

Cluster	**K=3**	**K=5**	**K=7**
Project	**KNN**	**KNN**	**KNN**
P29	82.0	82.0	82.0
P30	68.0	68.0	68.0
P31	42.0	42.0	41.0
P32	42.0	42.0	42.0
P33	180.0	180.0	180.0
P34	180.0	180.0	180.0
P35	82.0	82.0	82.0
P36	42.0	42.0	42.0
P37	42.0	42.0	68.0
P38	42.0	42.0	42.0

for these projects was recorded as 82.0, 68.0, 42.0, 42.0, 180.0, 180.0, 82.0, 42.0, 42.0 and 42.0. Moreover, with final value of K i.e. K=7, we have seen predictions as 82.0, 68.0, 41.0, 42.0, 180.0, 180.0, 82.0, 42.0, 68.0 and 42.0. The values show that for all the projects the prediction remain same for all three clusters. Therefore, we can say there is no effect of forming multiple clusters using K-Means over prediction of K-nearest algorithm.

5.4.4.5 Neural Networks

Forward propagation neural networks is applied for making predictions. The algorithm is tested over K= {3,5,7}. The predictions proposed by neural networks are given in Table 5.12. The prediction of Projects P29-P38 for algorithm known as neural network with K=3 are noted as 97.0, 91.0, 95.0, 90.0, 106.0, 98.0, 89.0,
93.0, 97.0 and 88.0. However the prediction for K=5 and K=7 are 96.0, 91.0, 92.0, 91.0, 109.0, 103.0, 94.0, 95.0, 93.0, 93.0 and 95.0, 88.0, 92.0, 89.0, 97.0, 97.0, 87.0,

Table 5.12: NN for K=3,5,7

Cluster	**K=3**	**K=5**	**K=7**
Project	**NN**	**NN**	**NN**
P29	97.0	96.0	95.0
P30	91.0	91.0	88.0
P31	95.0	92.0	92.0
P32	90.0	91.0	89.0
P33	106.0	109.0	97.0
P34	98.0	103.0	97.0
P35	89.0	94.0	87.0
P36	93.0	95.0	90.0
P37	97.0	93.0	91.0
P38	88.0	93.0	86.0

Table 5.13: RF for K=3,5,7

Cluster	**K=3**	**K=5**	**K=7**
Project	**RF**	**RF**	**RF**
P29	154.0	80.0	111.0
P30	55.0	68.0	75.0
P31	41.0	35.0	75.0
P32	55.0	68.0	75.0
P33	106.0	99.0	112.0
P34	98.0	213.0	135.0
P35	89.0	95.0	62.0
P36	93.0	95.0	112.0
P37	97.0	95.0	48.0
P38	88.0	209.0	112.0

90.0, 91.0, 86.0 respectively. We further noticed change in prediction for different clusters. For example P29, P37, P33 have different values for K= {3,5,7}.

5.4.4.6 Random Forest

The effort predictions using Random Forest trees are provided in Table 5.13. The predicted effort for ten projects P29-P38 for K={3,5,7} with Random forest are presented in this section. The predictions for K=3 are 154.0, 55.0, 41.0, 55.0, 106.0, 98.0, 89.0, 93.0, 97.0 and 88.0. With K=5, we noted predictions as 80.0, 68.0, 35.0, 68.0, 99.0, 213.0, 95.0, 95.0, 95.0 and 209.0. However the effort predicted for K=7 is noted as 111.0, 75.0, 75.0, 112.0, 135.0, 62.0, 112.0, 48.0 and 112.0. Moreover, we analysed that results show the predictions remain constant for K=5 and K=7 for Project P29. However, for P30 the predictions are reduced with increased number of clusters. Further for P30 the predictions are almost same for K={3,5&7}. Random Forest for P32 changes with changed values of K= 3,5,7.

Table 5.14: LR for K=3,5,7

Cluster	**K=3**	**K=5**	**K=7**
Project	**LR**	**LR**	**LR**
P29	130.0	129.5	129.0
P30	81.0	80.0	80.0
P31	89.0	89.0	89.0
P32	74.0	72.0	75.0
P33	148.0	148.0	148.0
P34	164.0	165.0	164.0
P35	90.0	91.0	91.0
P36	81.0	81.0	81.0
P37	86.0	86.0	81.0
P38	86.0	86.0	86.0

5.4.4.7 Linear Regression

The predictions made by machine learning algorithm named as linear regression for P29-P38 are presented in this section. The Table 5.14 shows these predictions. The prediction for K=3 are 130.0, 81.0, 89.0, 74.0, 148.0, 164.0, 90.0, 81.0, 86.0 and 86.0. However, when we changed value of K to 5, i.e. K=5, we get effort predictions as 129.0, 80.0, 89.0, 72.0, 148.0, 165.0, 91.0, 81.0, 86.0 and 86.0. Furthermore, we have seen predictions for K=7 as 129.0, 80.0, 89.0, 75.0, 148.0, 164.0, 91.0, 81.0, 81.0 and 81.0. We noticed no change in prediction for P30, P31, P33, P36 & P38 for K= {3,5,7}. However, for all other projects the predictions are changed for example for project P29, P32 etc.

5.4.4.8 Model outperformed for each project

As a result of modelling phase, the model selected for each project (P29-P38) is given in Table 5.15. These models are selected because they produce estimates closer to actual project effort with K={3,5,7}. When K=3, the models for project P29-P38 are Support Vector Machine, Linear Regression, Support Vector Machine,
Support Vector Machine, K Nearest Neighbour, Support Vector Machine, Linear Regression, K Nearest Neighbour, Linear Regression AND K Nearest Neighbour. However, for K=5 & 7, we seen outperforming models are the same except for project P32 for K=7 where K Nearest Neighbour and Support Vector Machine both perform equally well.

Table 5.15: Models for Each Project

Cluster	**K=3**	**K=5**	**K=7**
Project	**Model**	**Model**	**Model**
P29	SVM	SVM	SVM
P30	LR	LR	LR
P31	SVM	SVM	SVM
P32	SVM	SVM	SVM/KNN
P33	KNN	KNN	KNN

P34	SVM	SVM	SVM
P35	LR	LR	LR
P36	KNN	KNN	KNN
P37	LR	LR	LR
P38	KNN	KNN	KNN

Table 5.16: Validation of estimates Before clustering

	AE	MAE	MMRE	Pred (.25)	MSE	RMSE
SVM	200.0	20.0	0.029371	70	546.0	7.380
LR	215.0	21.5	0.358127	60	698.1	8.355
NN	414.0	41.4	0.542304	40	2361.0	15.36
RF	217.0	42.1	0.339472	70	3084.5	17.56
KNN	426.0	21.7	0.697484	50	819.70	9.053

5.4.5 Results Validation

The effort predictions made by models are validated using techniques such as Absolute error, Relative error, Mean Absolute Error, Mean Relative Error, Mean Squared Error, Root Mean squared Error, Mean Magnitude of Relative Error and Pred(x).

5.4.5.1 Before Clustering

In Table 5.16, we represent the predictions made by machine learning algorithms before applying clustering. We noted a magnitude of absolute error between 20.0 to 42.1 for machine learning algorithms. The minimum error of 20.0 was recorded for Support Vector Machine and Neural Network predicts with maximum error among the above-mentioned algorithms. Further models such as

5.4.5.2 After Clustering

The objective of this section is to explain predictions of machine learning algorithms after clustering was performed. The Table 5.17, presents results for K=3, using Support Vector Machine, Neural Network, Random Forest, K-Nearest Neighbour and Linear Regression. We noted the minimum error for K-Nearest

Table 5.17: Validation of estimates for When K=3

K=3	AE	MAE	MMRE	Pred(.25)	MSE	RMSE
SVM	204.0	20.4	0.360082	50	640.0	8.90
LR	225.0	22.5	0.367969	60	784.7	8.85
NN	424.0	42.4	0.545846	40	2563.4	16.0
RF	262.0	26.2	0.235700	60	1985.6	14.1
KNN	160.0	16.0	0.169833	70	826.0	16.0

Table 5.18: Validation of estimates for When K=5

K=5	AE	MAE	MMRE	Pred(.25)	MSE	RMSE
SVM	193.0	19.3	0.348890	50	581.5	7.625
LR	223.0	22.3	0.364886	70	779.7	8.830
NN	421.0	42.1	0.550435	40	2466.7	15.71
RF	425.1	42.5	0.643485	60	1985.0	14.09
KNN	160.0	16.8	0.169833	50	682.0	8.260

Neighbour. The Mean Absolute Error for K-Nearest Neighbour is 16.0 however, the Neural networks again produced much greater error of Mean Absolute Error equal to 42.4.

Similarly, Table 5.18, represents the prediction error for K= 5. These results show K-Nearest Neighbour be the best model among all other model. This model produces results with mean absolute error of 16.8. When we change value of K to 5, we see both neural networks and random forest do not perform well.
Furthermore, in Table 5.19, setting value of K to 7, we noticed the reduction in mean absolute error for K-Nearest Neighbour, and Support Vector Machine remains same for K=3 and K=7. The results show, if we change value of K to 7, the estimation error for K-Nearest Neighbour is reduced.

5.4.5.3 Expert Judgement

To analyses the predictions of Estimator judgments we calculated error rate and used different evaluation measure. We noted a magnitude of absolute error of

Table 5.19: Validation of estimates for When K=7

K=7	**AE**	**MAE**	**MMRE**	**Pred(.25)**	**MSE**	**RMSE**
SVM	220.0	19.3	0.358494	70	601.5	7.775
LR	222.0	22.2	0.363769	70	787.0	8.87
NN	420.0	42.0	0.523372	30	2461.0	16.21
RF	409.0	40.9	0.576322	30	2177.1	14.75
KNN	134.0	14.1	0.001373	80	566.5	7.52

Table 5.20: Validation of estimates made by expert judgements

Est.	AE	MAE	MMRE	Pred (.25)	MSE	RMSE
	339.0	40.2	0.332189	40	2515.8	15.8612

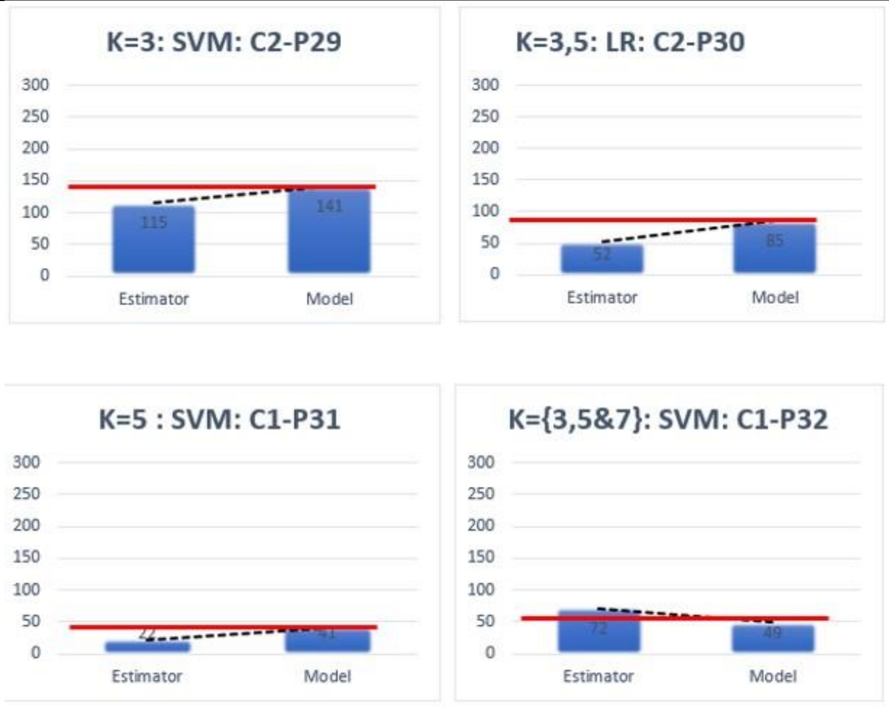


Figure 5.31: Comparison of Estimator and Model Estimates- P29-P32

339 hrs. for Estimator judgments. Table 5.20 represents results of all evaluation measures used for Estimators. we noticed a huge difference between actual and predictions of effort.

5.5 Comparison of Results

The aim of this section is to perform a comparison between the results produced by machine learning algorithms for software development organizations in area of Islamabad-Pakistan with the work done by H karna et al [25].

For comparison, we provided the analyzed difference between the estimator and models in Figure 5.31 & Figure 5.32 for ten projects (P29-P38) which we have to test the models generated by machine learning algorithms. The red horizontal line in these figures represent actual effort of projects. The vertical bars are labelled as Estimator and Expert. These bar shows the effort predicted by estimators and models. Moreover, the black dotted line is a trend line. This dotted line shows the difference in accuracy of estimation. If we see project P33, the red horizontal

Software Effort Estimation	Results
Application of Data Mining and Machine Learning algorithms [25]	K= 3 & 5 (Linear Regression , Classification and Regression Trees)
Effort estimation using machine learning for Organizations located in Islamabad Pakistan	**K= 5& 7 (Support Vector Machine, Linear Regression & K- Nearest Neighbor)**

Table 5.21: Comparison of [25] with proposed method

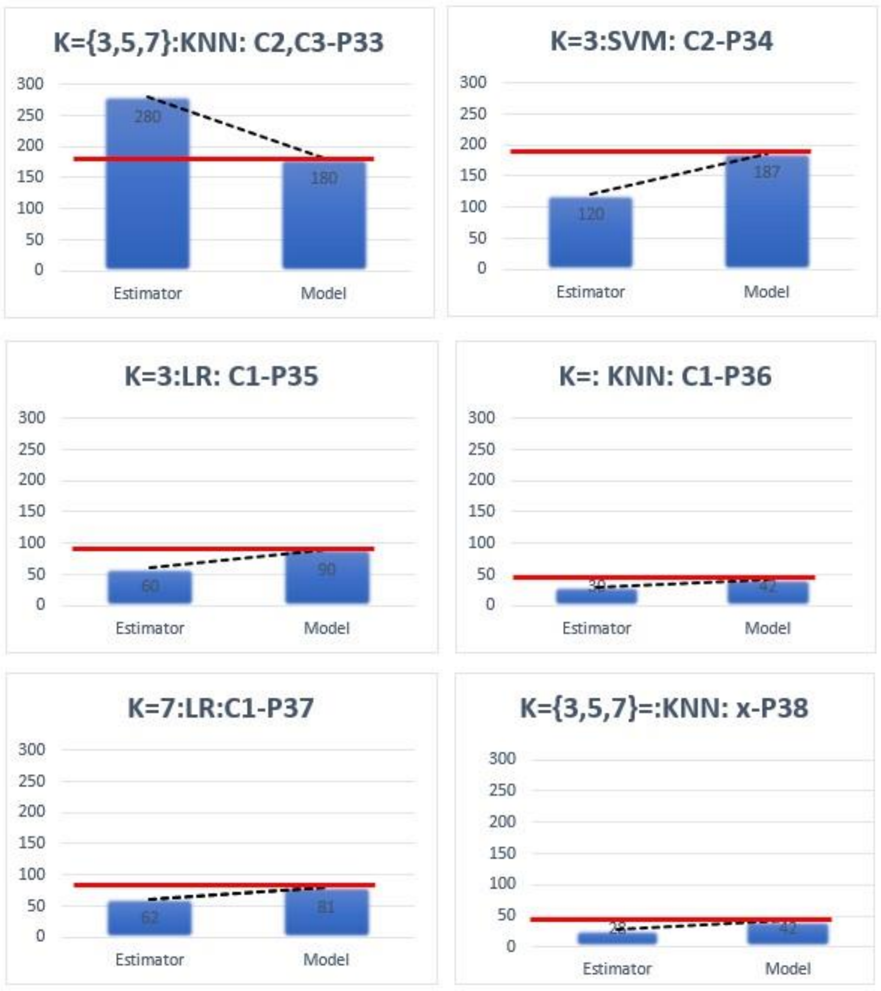


Figure 5.32: Comparison of Estimator and Model Estimates- P33-P38

line shows actual effort of 180 work hours. However, estimators tend towards overestimation and predicts 280 work hours and model predict 180 work hours which is exactly same. Thus, trend line for project P33 shows the decrease in estimates for P33. Furthermore, if we see Figure 5.31 & Figure 5.32 we can see it clearly that models tend to increase the accuracy of predictions for projects categorized as small, medium and large.

From comparison of predictions (See Table 5.21), we have noticed the same technique in both studies improves effort estimation in general. Moreover, the models produce improved results for K = 3 & 5 with Linear Regression , Classification and Regression Trees in work of H karna et al [25]. Furthermore, the results of this study shows improved result for K=5 with Support Vector Machine, Linear Regression & K- Nearest Neighbor. made by Estimators with those of machine learning algorithms, we determined the combination of machine learning algorithm and Estimator judgements produced better and robust results as compared to the situations when they are applied individually. Thus, we concluded that machine learning and data mining when applied for effort estimation improves the accuracy of predictions.

5.6 Discussions

The aim of this study is first to identify the strengths and weaknesses of existing techniques. Then, to improve effort predictions of software projects in software development organizations in region of Islamabad-Pakistan. To meet objectives, following research questions were answered:

RQ1: What are the strengths and weaknesses of Existing techniques?

From the existing literature (See Chapter 3), we analyzed from Estimator judgement to parametric methods, none of the techniques outperformed in all environments. Therefore, trend is shifted towards application of machine learning techniques such as Linear Regression, Artificial Neural Networks, K- Nearest Neighbor, Support Vector Machine, Fuzzy logic, and then hybrid techniques by combination of two or more techniques. We studied the predictions made by machine learning algorithms in different environments by none of the above-mentioned techniques outperformed in all environments. We also studied the evolution measures used for validation. The most used evaluation measures are Root Mean Squared Error, Mean Squared Error, Absolute Error, Relative Error, Mean Relative Error, Magnitude of Mean Relative Error, Pred (.25). We also recognized the performance

of machine learning techniques were patterned using Publicly available data sets. However, some studies formed data set after collecting it from different organizations. Therefore, we concluded the use of machine learning techniques improve effort prediction for different environments.

RQ2: How could effort prediction in software organizations be improved?

To answer this question, we collected data from two software organizations in region of Islamabad- Pakistan. We gathered data through survey and were able to collect information of 38 projects. The data set was divided into two parts: training set (P1-P28) and test set (P29-P38). Then, we performed experiments in two steps. In first step, we applied machine learning techniques such as Neural Networks, K- Nearest Neighbor, Support Vector Machine, Elastic net regression and Random Forest. In the second step, we applied K-means clustering over test set to form suitable clusters. We then applied these machine learning techniques to analyze the impact of K-means Clustering over effort prediction. We also noted the predictions made by Estimators. These values were recorded to identify the difference between Estimator judgement and when Estimator judgement is combined with machine learning techniques.

From the analysis of error magnitude for test, we concluded, machine learning techniques when combine with Estimator judgement provide better results as compared to using them individually. We also concluded the minute difference in prediction with and without clustering.

5.6.1 Summary

This chapter has explained the results produced by application of data mining and machine learning. In general, we have seen the improvement in accuracy of estimation with application of the algorithms. In the next chapter, we elaborate conclusions and future works of this study.

Chapter 6 Conclusion and future work

The previous chapters of this dissertation has provided the concept of effort estimation, how machine learning is used for effort estimation. We then provided experimental support for solution we have provided. At last we have presented results and how they are validated. Thus, the objective of this section if to provide conclusions, limitations and future work of this study.

6.1 Conclusion

In this dissertation, we have provided introduction of software development effort estimation in chapter 1. Chapter 2 has provided the preliminary studies for deep understanding. Further, Chapter 3 has provided Related work in two sections for the arena of software development effort estimation. Thereafter, in chapter 4, we proposed a framework for software development effort estimation for the software development organizations located in Islamabad-Pakistan. At the end, we have validated the results produced by model in chapter 5.

So, we come to conclusion that software development Effort estimation is necessary process to complete projects successfully. We applied machine learning algorithms and data mining technique for the effort estimation. We also compared the application of data mining techniques when applied with machine learning algorithm and when machine learning is applied individually. From the comparison presented in this study, we concluded both the methods improve accuracy of estimation when compared to humans.

6.2 Limitations

Setting up standard conclusions with inside the identical surroundings is a tough task. Primarily, due to the fact the method relies upon at the applicable variables that are like variables determined in work completed by [25, 154]. Despite that, if the work achieved with inside the study might be performed for different environments of software development companies. That would, consequently, assist in deducing results.

Another difficulty of this study could be using this research work is the data utilized is gathered from companies of comparable environment. In the identical context, we might need to accumulate data from greater companies however right here this wasn't viable because of disinclination in sharing the organizational data.

This research reviews the utility of data mining and machine learning algorithms to bring improvement in error magnitude of effort estimation within organizations. This study is performed inside software organizations located in Pakistan. The purpose of this research work is to carry perfection and increase reliability of software effort estimation process.

We have used 38 actual finished projects for this study. Out of these, we've got decided on four projects to check the constructed models. Thereafter, we've got implemented data mining and machine learning strategies on 28 projects with the goal of constructing predictive models. The formed models are then applied to estimate effort of ten projects (P29-P38). We investigated and concluded the use of models produce dependable and green effects for effort estimation.

This research proves we can use software engineering data to solve problems with machine learning, data mining and sometimes their combination also proves to be best among all other methods. However, the concealed patterns with inside data could stay unrevealed if data mining strategies become now no longer used. The use of programming version affords regular development in estimation of effort estimates.

This recommends the usage of comparable models for different software development agencies could be useful for minimizing the definite errors during calculating estimates. Therefore, this technique may be implemented to any software development company with the improved and updated data sets.
The research findings propose the utility of data mining and machine learning algorithms to construct the predictive model inside the experimental surroundings of software development organizations. In future, this study may be prolonged both by including extra capabilities to extrude the models for gaining extra dependable predictions. The outcomes show validity the applied methods. The obstacles of this study are already interpreted. We inspire research students to increase the work with goal of extending model. We additionally propose using similar methods to find comparable and better ways for problem solving.

6.3 Future work

In future, we aim to increase this work through making use of different clustering strategies and learners on similar and increased data set to generalize results. We further intend to use the same strategy for the estimation of work item with respect to the factors affecting each work item.

Chapter 7
References

[1] S. El Koutbi, A. Idri, and A. Abran, “Empirical evaluation of an entropybased approach to estimation variation of software development effort,” *Journal of Software: Evolution and Process,* vol. 31, no. 3, p. e2149, 2019.

[2] P. S. Kumar, H. Behera, A. Kumari, J. Nayak, and B. Naik, “Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades,” *Computer Science Review,* vol. 38, p. 100288, 2020.

[3] R. Shukla and A. Misra, “Software maintenance effort estimation-neural network vs regression modeling approach,” *Int. J. Futur. Comp. Applic.(Accepted, 2010),* 2010.

[4] I. F. de Barcelos Tronto, J. D. S. da Silva, and N. Sant’Anna, “Comparison of artificial neural network and regression models in software effort estimation,” in *2007 International Joint Conference on Neural Networks.* IEEE, 2007, pp. 771–776.

[5] Y. Mahmood, N. Kama, and A. Azmi, “A systematic review of studies on use case points and expert-based estimation of software development effort,” Journal of Software: Evolution and Process, p. e2245, 2020.

[6] Y. Mahmood, N. Kama, A. Azmi, and M. Ali, “Improving estimation accuracy prediction of software development effort: A proposed ensemble model,” in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE).* IEEE, 2020, pp. 1–6.

[7] K. E. Rao and G. A. Rao, “Ensemble learning with recursive feature elimination integrated software effort estimation: a novel approach,” *Evolutionary Intelligence,* pp. 1–12, 2020.

[8] B. Barry et al., “Software engineering economics,” *New York,* vol. 197, 1981.

[9] Mallidi, R.K. and Sharma, M., Empirical Study on Software Estimation Techniques.

[10] B. Khan, W. Khan, M. Arshad, and N. Jan, “Software cost estimation: Algorithmic and non-algorithmic approaches,” *International Journal of Data Science and Advanced Analytics,* vol. 2, no. 2, pp. 1–5, 2020.

[11] L. H. Putnam, “A general empirical solution to the macro software sizing and estimating problem,” *IEEE transactions on Software Engineering,* no. 4, pp. 345–361, 1978.

[12] R. C. Tausworthe, “Deep space network software cost estimation model,” 1981.

[13] A. J. Albrecht and J. E. Gaffney, “Software function, source lines of code, and development effort prediction: a software science validation,” *IEEE transactions on software engineering,* no. 6, pp. 639–648, 1983.

[14] A. Ali and C. Gravino, “A systematic literature review of software effort prediction using machine learning methods,” *Journal of Software: Evolution and Process,* vol. 31, no. 10, p. e2211, 2019.

[15] A. BaniMustafa, “Predicting software effort estimation using machine learning techniques,” in *2018 8th International Conference on Computer Science and Information Technology (CSIT).* IEEE, 2018, pp. 249–256.

[16] M. O. Elish, “Improved estimation of software project effort using multiple additive regression trees,” *Expert Systems with Applications,* vol. 36, no. 7,pp. 10 774–10 778, 2009.

[17] A. Garcıa-Floriano, C. Lopez-Martın, C. Yanez-Marquez, and A. Abran, “Support vector regression for predicting software enhancement effort,” *Information and Software Technology,* vol. 97, pp. 99–109, 2018.

[18] P. S. Kumar and H. Behera, “Estimating software effort using neural network: An experimental investigation,” in *Computational Intelligence in Pattern Recognition.* Springer, 2020, pp. 165–180.

[19] V. Resmi and S. Vijayalakshmi, “Analogy-based approaches to improve software project effort estimation accuracy,” *Journal of Intelligent Systems,* vol. 29, no. 1, pp. 1468–1479, 2019.

[20] A. E. Hassan, A. Hindle, P. Runeson, M. Shepperd, P. Devanbu, and S. Kim, “Roundtable: What’s next in software analytics,” *IEEE software,* vol. 30, no. 4, pp. 53–56, 2013.

[21] M. Azzeh, D. Neagu, and P. I. Cowling, “Fuzzy grey relational analysis for software effort estimation,” *Empirical Software Engineering,* vol. 15, no. 1, pp. 60–90, 2010.

[22] M. Aghazadeh and F. Soleimanian Gharehchopogh, “A new hybrid model of multi-layer perceptron artificial neural network and genetic algorithms in web design management based on cms,” *Journal of AI and Data Mining,* vol. 6, no. 2, pp. 409–415, 2018.

[23] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, “Software development effort estimation using regression fuzzy models,” *Computational intelligence and neuroscience*, vol. 2019, 2019. Online. Available: https//doi.org/10.1155/2019/ 8367214

[24] S. Tariq, M. Usman, and A. C. Fong, “Selecting best predictors from large software repositories for highly accurate software effort estimation,” *Journal of Software: Evolution and Process,* p. e2271.

[25] H. Karna, L. Vickovi´c, and S. Gotovac, “Application of data mining methods for effort estimation of software projects,” *Software: Practice and Experience,* vol. 49, no. 2, pp. 171–191, 2019.

[26] K. E. Rao and G. A. Rao, “Ensemble learning with recursive feature elimination integrated software effort estimation: a novel approach,” *Evolutionary Intelligence*, pp. 1–12, 2020.

[27] Y. Mahmood, N. Kama, A. Azmi, and M. Ali, “Improving estimation accuracy prediction of software development effort: A proposed ensemble model,” in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE).* IEEE, 2020, pp. 1–6

[28] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *Journal of Systems and Software,* vol. 70, no. 1-2, pp. 37–60, 2004.

[29] T. R. Benala and R. Mall, “Dabe: Differential evolution in analogy-based software development effort estimation,” *Swarm and Evolutionary Computation,* vol. 38, pp. 158–172, 2018.

[30] H. Azath, M. Mohanapriya, and S. Rajalakshmi, “Software effort estimation using modified fuzzy c means clustering and hybrid abc-mcs optimization in neural network,” *Journal of Intelligent Systems,* vol. 29, no. 1, pp. 251–263, 2018.

[31] K. Pillai and M. Jeyakumar, “A real time extreme learning machine for software development effort estimation.” *Int. Arab J. Inf. Technol.,* vol. 16, no. 1, pp. 17–22, 2019.

[32] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, “An effective approach for software project effort and duration estimation with machine learning algorithms,” *Journal of Systems and Software,* vol. 137, pp. 184– 196, 2018.

[33] I. SI, B. Tanveer, A. M. Vollmer, S. Braun, and N. b. Ali, “An evaluation of effort estimation supported by change impact analysis in agile software development,” *Journal of Software: Evolution and Process,* vol. 31, no. 5, p. e2165, 2019.

[34] K. Rak, Z. Car, and I. Lovrek, “Effort estimation model for software development projects based on use case reuse,” *Journal of Software: Evolution and Process,* vol. 31, no. 2, p. e2119, 2019.

[35] B. Baskeles, B. Turhan, and A. Bener, “Software effort estimation using machine learning methods,” in *2007 22nd international symposium on computer and information sciences.* IEEE, 2007, pp. 1–6.

[36] I. C. Suherman, R. Sarno et al., “Implementation of random forest regression for cocomo ii effort estimation,” in *2020 International Seminar on Application for Technology of Information and Communication (iSemantic).* IEEE, 2020, pp. 476–481.

[37] D. Zhang and J. J. Tsai, “Machine learning and software engineering,” *Software Quality Journal,* vol. 11, no. 2, pp. 87–119, 2003.

[38] T. Mukhopadhyay, S. S. Vicinanza, and M. J. Prietula, “Examining the feasibility of a case-based reasoning model for software effort estimation,” *MIS quarterly,* pp. 155–171, 1992.

[39] G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, “A comparison of software effort estimation techniques: using function points with neural networks, casebased reasoning and regression models,” *Journal of systems and software,* vol. 39, no. 3, pp. 281–289, 1997.

[40] F. Walkerden and R. Jeffery, “An empirical study of analogy-based software effort estimation,” *Empirical software engineering,* vol. 4, no. 2, pp. 135–158, 1999.

[41] N. ARC, “2000 monterey workshop on modelling software system structures in a fastly moving scenario,” 2000.

[42] G. Boetticher, “Using machine learning to predict project effort: Empirical case studies in data-starved domains,” in *First international workshop on modelbased requirements engineering,* 2001.

[43] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster, “An investigation of machine learning based prediction systems,” *Journal of systems and software,* vol. 53, no. 1, pp. 23–29, 2000.

[44] Y. ZHOU and Y. LI, “Analysis of nitrogen implantation to improve the channel mobility of 4h-sic n-mosfet,” *Research & Progress of SSE,* no. 2, p. 2, 2016.

[45] E. Kocaguneli, T. Menzies, and E. Mendes, “Transfer learning in effort estimation,” *Empirical Software Engineering,* vol. 20, no. 3, pp. 813–843, 2015.
[46] E. Kocaguneli, T. Menzies, and J. W. Keung, “On the value of ensemble effort estimation,” *IEEE Transactions on Software Engineering,* vol. 38, no. 6, pp. 1403–1416, 2011.
[47] L. L. Minku and X. Yao, “Ensembles and locality: Insight on improving software effort estimation,” *Information and Software Technology,* vol. 55, no. 8, pp. 1512–1528, 2013.
[48] S. Malathi and S. Sridhar, “A classical fuzzy approach for software effort estimation on machine learning technique,” *arXiv preprint arXiv:1112.3877,* 2011.
[49] J. Keung, E. Kocaguneli, and T. Menzies, “Finding conclusion stability for selecting the best effort predictor in software effort estimation,” *Automated Software Engineering,* vol. 20, no. 4, pp. 543–567, 2013
[50] P. Phannachitta, “On an optimal analogy-based software effort estimation,” *Information and Software Technology,* p. 106330, 2020. *Journal of Software: Evolution and Process,* p. e2271.
[51] E. Praynlin and P. Latha, “Software development effort estimation using anfis,” *International Information Institute (Tokyo). Information,* vol. 17, no. 4, p. 1325, 2014.
[52] H. J. Pasman and W. J. Rogers, “How to treat expert judgment? with certainty it contains uncertainty!” *Journal of Loss Prevention in the Process Industries,* p. 104200, 2020.
[53] P. Faria and E. Miranda, “Expert judgment in software estimation during the bid phase of a project–an exploratory survey,” in *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement.* IEEE, 2012, pp. 126–131.
[54] A. Priya Varshini and K. Anitha Kumari, “Predictive analytics approaches for software effort estimation: A review,” *Indian Journal of Science and Technology,* vol. 13, no. 21, pp. 2094–2103, 2020.
[55] M. Jørgensen, “What we do and don’t know about software development effort estimation,” *IEEE software,* vol. 31, no. 2, pp. 37–40, 2014.
[56] K. Molokken and M. Jorgensen, “A review of software surveys on software effort estimation,” in *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* IEEE, 2003, pp.
[57] M. Jorgensen, “Practical guidelines for expert-judgment-based software effort estimation,” *IEEE software,* vol. 22, no. 3, pp. 57–63, 2005.
[58] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development: a systematic literature review,” in *Proceedings of the 10th international conference on predictive models in software engineering,* 2014, pp. 82–91.
[59] S. Grimstad and M. Jørgensen, “Inconsistency of expert judgment-based estimates of software development effort,” *Journal of Systems and Software,* vol. 80, no. 11, pp. 1770–1777, 2007.
[60] K. Moløkken-Østvold and M. Jørgensen, “Group processes in software effort estimation,” *Empirical Software Engineering,* vol. 9, no. 4, pp. 315–334, 2004.

[61] N. J. Nilsson, “Introduction to machine learning. an early draft of a proposed textbook (1998), Department of Computer Science,” *USA; Stanford University*, vol. 56, no. 2, pp. 387–99, 2005.

[62] S. University. (2020) Machine learning. Accessed: 2020-27-09. [Online]. Available: https://www.coursera.org/learn/machine-learning/supplement/1O0Bk/unsupervised-learning

[63] A. Ng. Machine Learning. Accessed: 2020-27-09. Online.Available: https://www.coursera.org/learn/machine-learning/supplement/1O0Bk/ unsupervised-learning

[64] T. Hastie, R. Tibsharani, and J. Friedman, “Springer series in statistics the elements of,” *Math. Intell*, vol. 27, no. 819, pp. 83–85, 2009.

[65] A. Gray and S. MacDonell, “Applications of fuzzy logic to software metric models for development effort estimation,” in *1997 Annual Meeting of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No. 97TH8297).* IEEE, 1997, pp. 394–399.

[66] P. L. Braga, A. L. Oliveira, and S. R. Meira, “Software effort estimation using machine learning techniques with robust confidence intervals,” in *7th international conference on hybrid intelligent systems (HIS 2007).* IEEE, 2007, pp. 352–357.

[67] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[68] O. Harrison, “Machine learning basics with the k-nearest neighbors algorithm,” *Towards Data Science. September*, vol. 10, 2018. Accessed: 202027-09. Online. Available: https://towardsdatascience.com/machine-learningbasics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[69] O. Kramer, “Dimensionality Reduction with Unsupervised Nearest Neighbors,” *Intell. Syst. Ref. Libr.*, vol. 51, pp. 13–23, 2013, doi: 10.1007/978-3-64238652-7.

[70] R. Gholami and N. Fakhari, “Support vector machine: principles, parameters, and applications,” in *Handbook of Neural Computation. Elsevier*, 2017, pp. 515–535.

[71] A. Ng. (2020) Machine learning offered by stanford. Accessed: 2020-27-09. [Online]. Available: https://www.coursera.org/learn/machine-learning

[72] H. Riesen, Kaspar and Bunke, “Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR),” *springer*, pp. 287–297, 2008.

[73] V. Vapnik, *The nature of statistical learning theory.* Springer science & business media, 2013.

[74] C. M. Bishop, “Pattern recognition and machine learning: springer new york,” 2006.

[75] S. Tong and D. Koller, “Restricted bayes optimal classifiers,” in *AAAI/IAAI*, 2000, pp. 658–664.

[76] S. Abe, *Support vector machines for pattern classification.* Springer, 2005, vol. 2.

[77] T. Zhang, “An introduction to support vector machines and other kernelbased learning methods,” *AI Magazine*, vol. 22, no. 2, p. 103, 2001.

[78] R. Singh, A. Kainthola, and T. Singh, “Estimation of elastic constant of rocks using an anfis approach,” *Applied Soft Computing*, vol. 12, no. 1, pp. 40–45, 2012.

[79] H.-J. Lin and J. P. Yeh, “Optimal reduction of solutions for support vector machines,” *Applied Mathematics and Computation,* vol. 214, no. 2, pp. 329–335, 2009.

[80] S. University. (2020) Week 1 lecture notes, ml:introduction, what is machine learning? Accessed: 2020-27-09.Online. Available: https://www.coursera.org/learn/machine-learning/resources/JXWWS

[81] Andrew Ng. (2020) Week 2 lecture notes, ml:linear regression with multiple variables, *Coursera* Accessed: 2020-27-09. Online. Available: https://www.coursera.org/learn/machine-learning/resources/QQx8l

[82] Wikipedia. (2020) Elastic net regularization. Accessed: 2020-27-09. [Online. Available: https://en.wikipedia.org/wiki/Elastic_net_regularization

[83] M. Pecht, “Prognostics and health management of electronics,” *Encyclo-pedia of structural health monitoring,* 2009.

[84] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition,* vol. 1. IEEE, 1995, pp. 278–282.

[85] E. University, IEEE 2005 Symposium on Computational Intelligence and games. IEEE, 2005.

[86] C. D. Manning, H. Sch¨utze, and P. Raghavan, *Introduction to information retrieval.* Cambridge university press, 2008. Online. Available: https://nlp.stanford.edu/IR-book/html/htmledition/k-nearest-neighbor-1.html

[87] O. F. Ertu˘grul and M. E. Ta˘gluk, “A novel version of k nearest neighbor: ¨ Dependent nearest neighbor,” *Applied Soft Computing,* vol. 55, pp. 480– 490, 2017.

[88] U. N. Dulhare, K. Ahmad, and K. A. B. Ahmad, *Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications.* John Wiley & Sons, 2020.

[89] A. K. Nandi and H. Ahmed, Condition Monitoring with Vibration Signals: *Compressive Sampling and Learning Algorithms for Rotating Machines.* John Wiley & Sons, 2020.

[90] cmdline. (2019) Implementing k-means clustering in python from scratch. Accessed: 2020-27-07. [Online]. Available: https://cmdlinetips.com/2019/ 05/k-means-clustering-in-python/

[91] M. Kalra, N. Lal, and S. Qamar, “K-Mean Clustering Algorithm Approach for Data Mining of Heterogeneous Data,” *Lect. Notes Networks Syst.*, vol. 10, pp. 61–70, 2018, doi: 10.1007/978-981-10-3920-1 7.

[92] C. Outline, “Prediction in medicine - The data mining algorithms of predictive analytics,” *Pract. Predict. Anal. Decis. Syst. Med. Informatics Accuracy Cost-Effectiveness Healthc. Adm. Deliv. Incl. Med. Res.*, pp. 239–259, 2014, doi: 10.1016/b978-0-12-411643-6.00015-6.

[93] A. Zimmermann, “Method evaluation, parameterization, and result validation in unsupervised data mining: A critical survey,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 10, no. 2, p. e1330, 2020.

[94] J. Yu, H. Huang, and S. Tian, “Cluster validity and stability of clustering algorithms,” in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR).* Springer, 2004, pp. 957–965.

[95] M. Korte and D. Port, “Confidence in software cost estimation results based on mmre and pred,” in *Proceedings of the 4th international workshop on Predictor models in software engineering*, 2008, pp. 63–70.
[96] L. L. Minku and X. Yao, “How to make best use of cross-company data in software effort estimation?” in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 446–456.
[97] M. El Bajta, “Analogy-based software development effort estimation in global software development,” in *2015 IEEE 10th International Conference on Global Software Engineering Workshops*. IEEE, 2015, pp. 51–54.
[98] T. Menzies, S. Williams, O. Elrawas, D. Baker, B. Boehm, J. Hihn, K. Lum, and R. Madachy, “Accurate estimates without local data?” *Software Process: Improvement and Practice*, vol. 14, no. 4, pp. 213–225, 2009.
[99] R. Jensen, “An improved macrolevel software development resource estimation model,” in *5th ISPA Conference*, 1983, pp. 88–92.
[100] A. K. Bardsiri and S. M. Hashemi, “Software effort estimation: a survey of well-known approaches,” *International Journal of Computer Science Engineering (IJCSE)*, vol. 3, no. 1, pp. 46–50, 2014.
[101] R. T. Hughes, “Expert judgement as an estimating method,” *Information and software technology*, vol. 38, no. 2, pp. 67–75, 1996.
[102] B. Boehm, C. Abts, and S. Chulani, “Software development cost estimation approaches—a survey,” *Annals of software engineering*, vol. 10, no. 1-4, pp. 177–205, 2000.
[103] M. Shepperd and C. Schofield, “Estimating software project effort using analogies,” *IEEE Transactions on software engineering*, vol. 23, no. 11, pp. 736–743, 1997. doi: Online. Available: https://doi.org/10.1109/32.637387
[104] R. Park, “The central equations of the price software cost model,” in *4th COCOMO Users Group Meeting*, 1988
[105] L. H. Putnam and W. Myers, *Measures for excellence: reliable software on time, within budget*. Prentice Hall Professional Technical Reference, 1991.
[106] C. Jones, *Applied software measurement: global analysis of productivity and quality*. McGraw-Hill Education Group, 2008.
[107] A. J. Albrecht, “Measuring application development productivity,” in *Proc. Joint Share, Guide, and IBM Application Development Symposium*, 1979, pp. 83-92, 1979.
[108] F. Arslan, “A review of machine learning models for software cost estimation,” 2019.
[109] S. El Koutbi, A. Idri, and A. Abran, “Empirical evaluation of an entropybased approach to estimation variation of software development effort,” *Journal of Software: Evolution and Process*, vol. 31, no. 3, p. e2149, 2019.
[110] A. Saberi Nejad and R. Tavoli, “A method for estimating the cost of software using principle components analysis and data mining,” *Journal of Electrical and Computer Engineering Innovations (JECEI)*, vol. 6, no. 1, pp. 33–42, 2017.
[111] L. L. Minku and X. Yao, “Which models of the past are relevant to the present? a software effort estimation approach to exploiting useful past models,” *Automated Software Engineering*, vol. 24, no. 3, pp. 499–542, 2017.
[112] G. Shankar and L. K. Sharma, “Comparative study for software cost estimation with data mining techiques.” vol. 07, no. 05, pp. 1968–1971, 2019.

[113] R. S. Bedi and A. Singh, “Software effort estimation analysis using data mining techniques.” pp. 34–38.

[114] S. Chhabra and H. Singh, “Optimizing design of fuzzy model for software cost estimation using particle swarm optimization algorithm,” *International Journal of Computational Intelligence and Applications,* vol. 19, no. 01, p. 2050005, 2020.

[115] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, “Software development effort estimation using regression fuzzy models,” *Computational intelligence and neuroscience,* vol. 2019, 2019.

[116] W. Zhang, Y. Yang, and Q. Wang, “Using bayesian regression and em algorithm with missing handling for software effort prediction,” *Information and software technology,* vol. 58, pp. 58–70, 2015.

[117] J. F. Vijay, “Enrichment of accurate software effort estimation using fuzzybased function point analysis in business data analytics,” *Neural Computing and Applications,* vol. 31, no. 5, pp. 1633–1639, 2019.

[118] I. Abnane, A. Idri, and A. Abran, “Fuzzy case-based-reasoning-based imputation for incomplete data in software engineering repositories,” *Journal of Software: Evolution and Process, p. e2260,* 2020.

[119] A. Bala and A. Abran, “Impact analysis of multiple imputation on effort estimation models with the isbsg repository of software projects,” *Softw. Meas. News,* vol. 23, no. 1, pp. 17–34, 2018.

on cocomo to increase the accuracy of software cost estimation,” *Journal of Advances in Computer Engineering and Technology,* vol. 4, no. 1, pp. 27–40, 2018.

[120] M. Khazaiepoor, A. K. Bardsiri, and F. Keynia, “A dataset-independent model for estimating software development effort using soft computing techniques,” *Applied Computer Systems,* vol. 24, no. 2, pp. 82–93, 2019.

[121] P. V. de Campos Souza, A. J. Guimaraes, V. S. Araujo, T. S. Rezende, and V. J. S. Araujo, “Incremental regularized data density-based clustering neural networks to aid in the construction of effort forecasting systems in software development,” *Applied Intelligence,* vol. 49, no. 9, pp. 3221–3234, 2019.

[122] P. V. d. C. Souza, A. J. Guimaraes, V. S. Araujo, T. S. Rezende, and V. J. S. Araujo, “Regularized fuzzy neural networks to aid effort forecasting in the construction and software development,” *arXiv preprint arXiv:1812.01351,* 2018.

[123] I. Kaur, G. S. Narula, R. Wason, V. Jain, and A. Baliyan, “Neuro fuzzy—cocomo ii model for software cost estimation,” *International Journal of Information Technology,* vol. 10, no. 2, pp. 181–187, 2018.

[124] H. Carvalho, “Ensemble regression models for software development effort estimation: A comparative study,” *International Journal of Software Engineering & Applications (IJSEA),* vol. 11, no. 3, 2020.

[125] P. Phannachitta and K. Matsumoto, “Model-based software effort estimation–a robust comparison of 14 algorithms widely used in the data science community,” *INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING INFORMATION AND CONTROL,* vol. 15, no. 2, pp. 569– 589, 2019.

[126] I. Thamarai and S. Murugavalli, “Modified genetic algorithm-simulated annealing based software effort and cost estimation,” *International Journal of Pure and Applied Mathematics,* vol. 118, no. 16, pp. 777–793, 2018.

[127] A. Khatoon and R. Kaur, “Optimization estimation parameters of cocomo model ii through genetic algorithm,” 2018.

[128] T. Xia, J. Chen, G. Mathew, X. Shen, and T. Menzies, “Why Software Effort Estimation Needs SBSE,” pp. 1–15, 2018, [Online]. Available: http://arxiv.org/abs/1804.00626.

[129] T. Xia, R. Krishna, J. Chen, G. Mathew, X. Shen, and T. Menzies, “Hyperparameter optimization for effort estimation,” *arXiv preprint arXiv:1805.00336*, 2018.

[130] S. Mensah, J. Keung, S. G. MacDonell, M. F. Bosu, and K. E. Bennin, “Investigating the significance of the bellwether effect to improve software effort prediction: Further empirical study,” *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1176–1198, 2018.

[131] A. E. Hassan, A. Hindle, P. Runeson, M. Shepperd, P. Devanbu, andS. Kim, “Roundtable: What’s next in software analytics,” *IEEE software*, vol. 30, no. 4, pp. 53–56, 2013.

[132] R. Krishna and T. Menzies, “Bellwethers: A baseline method for transfer learning,” *IEEE Transactions on Software Engineering*, vol. 45, no. 11, pp. 1081–1105, 2018.

[133] D. Wu, J. Li, and C. Bao, “Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation,” *Soft Computing*, vol. 22, no. 16, pp. 5299–5310, 2018.

[134] S. Bilgaiyan, K. Aditya, S. Mishra, and M. Das, “A swarm intelligence based chaotic morphological approach for software development cost estimation,” *International Journal of Intelligent Systems and Applications*, vol. 10, no. 9, p. 13, 2018.

[135] F. Alsalman and A. Ali, “Estimation effort using developed cat swarm optimization,” in *Proceedings of the 9th International Conference on Information Systems and Technologies*, 2019, pp. 1–7.

[136] K. Langsari, R. Sarno et al., “Optimizing effort parameter of cocomo ii using particle swarm optimization method,” *Telkomnika*, vol. 16, no. 5, pp. 2208–2216, 2018.

[137] P. Jodpimai, P. Sophatsathit, and C. Lursinsap, “Re-estimating software effort using prior phase efforts and data mining techniques,” *Innovations in Systems and Software Engineering*, vol. 14, no. 3, pp. 209–228, 2018.

[138] AA. Khatibi Bardsiri, “A new combinatorial framework for software services development effort estimation,” *International Journal of Computers and Applications*, vol. 40, no. 1, pp. 14–24, 2018.

[139] O. Malgonde and K. Chari, “An ensemble-based model for predicting agile software development effort,” *Empirical Software Engineering*, vol. 24, no. 2, pp. 1017–1055, 2019.

[140] M. Ferna´ndez-Diego and F. Gonza´lez-Ladro´n-de-Guevara, “Application of mutual information-based sequential feature selection to isbsg mixed data,” *Software Quality Journal*, vol. 26, no. 4, pp. 1299–1325, 2018.

[141] A. Khatibi Bardsiri, “An intelligent model to predict the development time and budget of software projects,” *International Journal of Nonlinear Analysis and Applications*, vol. 11, no. 2, pp. 85–102, 2020. prediction using machine learning methods,” *Journal of Software: Evolution and Process*, vol. 31, no. 10, p. e2211, 2019.

[142] H. Karna, S. Gotovac, and L. Vickovi´c, “Data mining approach to effort modeling on agile software projects,” *Informatica*, vol. 44, no. 2, 2020.

[143] I. Etikan and K. Bala, “Sampling and sampling methods,” *Biometrics & Biostatistics International Journal*, vol. 5, no. 6, p. 00149, 2017.

[144] M. J. Blanca, R. Alarc´on, J. Arnau, R. Bono, and R. Bendayan, “Nonnormal data: Is anova still a valid option?” *Psicothema*, vol. 29, no. 4, pp. 552–557, 2017.

[145] V. A. Vieira, *Experimental Designs Using ANOVA*, Thomson/Brooks/Cole Belmont, CA, 2 vol. 15, no. 2. 2011.

[146] G. Beranek, W. Zuser, and T. Grechenig, “Functional group roles in software engineering teams,” in *Proceedings of the 2005 workshop on Human and social factors of software engineering*, 2005, pp. 1–7.

[147] E. Alpaydin, *Introduction to machine* learning. MIT press, 2020.

[148] A. Bartschat, M. Reischl, and R. Mikut, “Data mining tools,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1309, 2019.

[149] A. Idri, S. Mbarki, and A. Abran, “Validating and understanding software cost estimation models based on neural networks,” in *Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications*, 2004. IEEE, 2004, pp. 433–434.

[150] M. P. S. Ansolabehere and A. Lee. (2014) Precinct-level election data. vi. Accessed: 2020-15-5. [Online]. Available: http://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff

[151] Q. Liu and R. C. Mintram, “Preliminary data analysis methods in software estimation,” *Software quality journal*, vol. 13, no. 1, pp. 91–115, 2005

[152] R. Jeffery, M. Ruhe, and I. Wieczorek, “Using public domain metrics to estimate software development effort,” in *Proceedings Seventh International Software Metrics Symposium*. IEEE, 2001, pp. 16–27.

[153] S. Kim, W. M. Lively, and D. B. Simmons, “An effort estimation by uml points in early stage of software development.” in *Software Engineering Research and Practice*. Citeseer, 2006, pp. 415–421. Online. Available: http://ww1.ucmss.com/books/LFS/CSREA2006/SER5194.pdf

[154] T. E. Ayyıldız and H. C. Terzi, “Case Study on Software Effort Estimation,” *International Journal of Information and Electronics Engineering*, vol. 7, no. 3, pp. 103–107, 2017.