Major Project
Report

Software Effort Prediction Using Machine Learning Techniques

Submitted in partial fulfillment of
the requirements for the award of the degree of

Master of Technology
in
Software Engineering

Submitted by
Chandan Kumar Yadav
2K15/SWE/06

Under the guidance of
Dr. Ruchika Malhotra
Associate Head & Assistant Professor
Department of CSE

Department of Computer Science & Engineering
Delhi Technological University
Delhi, India – 110042

June 2017

Certificate

Department of Computer Science & Engineering

Delhi Technological University

This is to certify that report entitled "**Chandan Kumar Yadav**" has completed the project titled "**Software Effort Prediction using Machine Learning Techniques**" under my supervision in partial fulfillment of the Master of Technology degree in Software Engineering at Delhi Technological University.

Dr. Ruchika Malhotra
Associate Head & Assistant Professor
Department of Computer Science & Engineering
Delhi Technological University
Delhi -110042
(Project Guide)

Declaration

We hereby declare that the thesis work entitled "**Software Effort Prediction using Machine Learning Techniques**" which is being submitted to **Delhi Technological University** , in partial fulfillment of requirements for the award of degree of Master of Technology (Software Engineering) is a bonafide report of thesis carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Chandan Kumar Yadav
2K15/SWE/06

Acknowledgements

I am very thankful to **Dr.Ruchika Malhotra** (Associate Head & Assistant Professor, Computer Science & Eng. Dept.) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by me.

I would also like to express my gratitude to the university for providing the laboratories, infrastructure, testing facilities and environment which allowed me to work without any obstructions.

I would also like to appreciate the support provided by our lab assistants, seniors and peer group who aided me with all the knowledge they had regarding various topics.

CHANDAN KUMAR YADAV
M. Tech. in Software Engineering
Roll No. 2K15/SWE/06

Abbreviations

LSR	**L**east **S**quares **R**egression
CFS	**C**orrelation Based **F**eature **S**ub selection
REP	**R**educes **E**rror **P**runing
RBF	**R**adial **B**asis **F**unction
CBR	**C**ase **B**ased **R**easoning
PCA	**P**rincipal **C**omponent **A**nalysis
MLP	**M**ultilayer **P**erceptron
ANN	**A**rtficial **N**eural **N**etwork
PNN	**P**robabilistic **N**eural **N**etworks
SVM	**S**upport **V**ector **M**achine
SLIM	**S**oftware **L**ifecycle **M**anagement
PRED	**P**rediction
CART	**C**lassification **A**nd **R**egression **T**ree
ASMA	**A**ustralian **S**oftware **M**etrics **A**ssociation
MART	**M**ultiple **A**dditive **R**egression **T**ree
WEKA	**W**aikato **E**nvironment for **K**nowledge **A**nalysis
MMRE	**M**ean **M**agnitude of **R**elative **E**rror
KSTAR	**K**orea **S**uperconducting **T**okamak **A**dvanced **R**esearch
COCOMO	**C**onstructive **C**ost **M**odel

Abstract

Precise estimation of software effort is a crucial task in the software engineering domain. The effort is the most important factor which affects the budget of a project. Therefore, software effort estimation is very crucial and there is continuously a necessity to improve its accuracy as much as possible. For a quality software effort, both over-estimation as well as under-estimation may lead to very dangerous consequences. Therefore, it is very important to determine the best technique which can give exact results for software effort estimation. In this study, we analyze several machine learning (ML) techniques like bagging, linear regression, KStar, M5Rules, RIP (Reduces Error Pruning) Tree and Multilayer Perceptron (MLP) in order to develop models to predict software effort. Two different datasets i.e. China dataset and Albrecht dataset have been used in our research. Results of machine learning algorithms can be different from dataset to dataset. Multilayer perceptron has shown good performance for China dataset and REP Tree shown for Albrecht dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In any software industry, for quality software project, estimating accurate software effort is the most significant research area in the field of software development. For software effort estimation, getting an estimate of the person-months and the time expected to complete the project is crucial. Although software effort estimation is playing most important role in the field of software project development, there has been very minor improvement in the last thirty or forty years. One of the most important factor for failure is inaccurate estimate of required resources. Even though lots of software effort estimation models are available, it is still required to research novel models for improving the accuracy of such estimation because the problem for effort estimation and accuracy remain same. So the construction of software effort prediction models has motivated to estimate the accurate software effort as much as possible.

For a development of software artefact, software effort prediction is required to estimate the specified effort. Actually, ML technique considers an only historical data set which hold the various historical software project. These projects are expressed by features with their values to characterize those project. The features with similar values may produce almost the similar software project efforts. The main task of using ML techniques is to learn the inherent patterns of feature value and their relations with project efforts

and predicting the effort for new projects. Regarding software effort prediction, recent studies of effort provide a detailed review of several studies. The prediction techniques are classified into 3 general categories [1] :

- Expert judgment: This type of technique has been used widely. This technique estimates the software effort on the basis of expert experience for similar projects. The accuracy of such estimates highly depends on the ability of the expert and degree to what extent expert experience involved in which new project concurs. Expert judgment is always complicated to evaluate but it can be effective for effort estimation when an adjustment factor for algorithm models involved [2].

- Algorithmic: The algorithmic models is the most popular in literature. Generally, Software size, function point, source lines of code are used as principal effort driver. Examples of this models are COCOMO model, Function Points Analysis, and SLIM model. These models are also known as parametric models because these models predict the software effort by a fixed formulas from the historical dataset.

- Machine learning: The machine learning approaches have been used as a compliment for both expert judgment and algorithmic models in the last decade. These approaches include ANN, Fuzzy logic, CBR, regression trees, SVM etc. The advantage of this approaches, it models the complex set of relationship between effort and independent variable. It is used for those difficult problems where a result must be learned from historical project data.

There are a number of ML techniques recommended in the literature like bagging, linear regression, KStar, M5Rules, RIP Tree and Multilayer Perceptron. But it is very difficult to find which one of the ML technique is superior over the other techniques using multiple data sets. Hence, intense studies are required to draw well-formed, generalized and widely acceptable conclusions based on the experimental evidence. Thus, the following research questions

are addressed in this work:

- RQ1: What is the performance of ML techniques for software effort prediction? In this question, the performance of ML techniques is assessed for developing software effort estimation models on two data sets. The performance is evaluated using four performance metrics namely, Mean Magnitude of Relative Error (MMRE) and Prediction (PRED) at level 0.25, 0.50 and 0.75 respectively.

- RQ2: Which is the best and the worst ML techniques for software effort prediction? In this research question, we determine ML techniques which give the best and the worst results corresponding to each data set investigated in the study.

The objective of this thesis is to empirically evaluate the accuracy of different machine learning techniques in order to predict software development effort. The results obtained from the empirical studies will help to improve the results which obtained from past studies. In order to this, several ML techniques being used in our research such as bagging, linear regression, KStar, M5 rules, multilayer perceptron and REP Tree. These techniques are the modern trends in the field of effort estimation. For results, ML techniques are applied on China dataset and Albrecht dataset consisting of 499 and 24 projects respectably.

The rest of the thesis has been divided into various chapter. In chapter 2, related work done for the research have been explained in brief. In chapter 3, the research methodology and various ML techniques for software effort estimation have been discussed. In chapter 4, whole research background has been explained. First of all the 2 data sets used in our research work have been explained. After that, the method used for cross-validation has been described. The tool which is used for calculation of results has been

explained. In chapter 5, the results after application of different algorithms on the dataset have been computed. The future work have been explained in chapter 6. Finally, all the references used in the research have been mentioned [3].

Chapter 2

Related Work

Software effort estimation plays very crucial task in calculation of the development cost of a software project. Various methods like Empirical techniques, algorithmic effort estimation, regression techniques, theory based techniques and machine learning have been presented and many models have been discussed in previous study. The understanding and controlling of critical variables that affects software effort is very essential job in software project management. According to Subramaniam et al. [4], inferred that software effort is significantly affected by various adjustment variables like complexity of software, platform used and type of program. The adjustment variables like complexity of software and reliability are used by COCOMO I, COCOMO II and Function Points for computation of adjusted estimate of software cost and effort. Original model developers have stated most of these adjustment variables and in subsequent studies they have been modified by other researchers. The arguable point among researchers have been the usage of minimum set of adjustment variable and also subsequently making the updates of adjustment variables so that differences in software projects could be reflected. Other adjustment variables for task assignment patterns have been suggested in intermediate COCOMO which is proposed to improve the accuracy of software effort estimation.

According to Smith et al.[5], 4-task assignment factors i.e. intensity, concurrency, fragmentation and team size have been taken into consideration for their impact on software effort development. All these factors improved the estimations of intermediate COCOMO I model. These factors along with unadjusted function points helps in achieving a better effort estimation model which results in improved predictive ability as compared to COCOMO model.

Various analogy based estimations have been done for software effort prediction. The analogy based estimation compares the project whose effort estimation is needed with some historical project for similarity and similarity is computed using distance metrics like Eucledian, Minkowiski, manhattan distance etc. Chiu and Huang [6], have suggested an adjusted analogy based effort estimation model which makes use of Genetic Algorithm to adjust the software effort based on various distance metrics. In the conventional Euclidean distance, the feature of the projects have the same weight or unweight there by significance of every feature does not need into consideration. The paper by Tosun et al.[7], has proposed another novel approach of improving the estimation accuracy with the help of a new feature weight assignment algorithm which gives better consequences as compared to previous research. Here a statistical technique called Principal Component Analysis (PCA) has used to implement the two weighted assignment heuristics.

The machine learning (ML) techniques have been widely used in effort estimation. The research by Finnie and Witting [8] [9], have analysed the capability of 2 artificial intelligence (AI) methodologies i.e. "Artificial Neural Networks" (ANN) and "Case Based Reasoning" (CBR) to make advancement in application evaluation models, utilize the same data set which is "Australian Software Metrics Association" (ASMA). Additionally, the capability of ANN and CBR, gives premise to improvement in application estimation models rather than relapse models. AI models are prepared for giving sufficient evaluation models. The performance of both models is largely de-

pendent on training data, and the degree to which appropriate data of the project is available. Other than ANN and CBR, various ML approaches like "Classification and Regression Trees" (CART) and "Multiple Additive Regression Tree" (MART) have been published [10]. The paper by Elish [11], has performed a comparative analysis of MART as model of software effort estimation with respect to recently existing models i.e. support vector regression models with linear regression, RBF kernels, RBF neural networks and linear regression. The resulting effort estimation with MART proves to have better accuracy.

The Genetic Algorithm (GA) have been used a lot for accurately estimating the effort. According to Burgess and Lefley [12], the ability of "Genetic Programming" has been used for estimation of effort evaluation and comparison has also been ensured with the ANN, Linear LSR etc. Here Desharnais dataset of 81 software project is used for comparison and obtained results are totally dependent on the used fitness function.

The paper by Braga et al. [13], has proposed a method which provides effort estimation as well as a confidence interval on the basis of machine learning. The authors have suggested for using robust confidence intervals. This confidence interval doesnt depend on a probability distribution of the errors in the training set. Numerous experiments have been done with 2 datasets of software projects: "Desharnais and NAS", which are expected to analyse ML techniques to making accurate effort estimation. After simulations, the suggested method was able for making confidence intervals which are very important for a client of the effort estimation system. The paper by Martin et al. [14] **Lopez-Martin2006** introduced an improved Fuzzy Logic Model for measurement of software development effort and suggested a new methodology for effort estimation. The paper by Elish [11], the comparative study has been carried out to make software effort prediction systems with the help of ML methods such as ANNs, CBR and RI (Rule Induction). Ac-

cording to this paper comparison of the software effort prediction systems have been carried out with respect to configurability, accuracy and explanatory value.

The paper by Bibi and Stamelos [15], has proposed a number of estimation rules for taking a decision to choose an appropriate machine learning techniques in terms of causality, uncertainty, accuracy, dynamic updates, handling of missing values, comprehensibility, sensitivity and applicability for software development effort estimation. Hence the author proposed several rules to select an appropriate estimation method according to his need and for the research of the estimator. The paper by Gallego et al. [16], the author has shown one of the biggest challenges in front of developers, regarding prediction of development effort to the software system, in terms of size of a project, complexity, developer abilities, and other metrics.

The paper by Pendharkar [17], made the suggestion of a "Probabilistic Neural Networks" (PNN) methodology for simultaneous estimation of the values of development parameters, i.e. effort or size and for the probability that the estimated value of the parameter will be more than its actual value. According to Radlinski and Hoffmann [18], the author has made a comparison of the accuracy of predictions for software effort in taking into account several ML techniques. Here the most important purpose is to explore the stability of this prediction by studying if certain techniques, get a same level of accuracy for dissimilar data sets. The results of the prediction accuracy show variation to a dataset used for every ML technique.

The paper by Bisi and Goyal [19], stated that for highly reliable and quality software, resource management is necessary. Indicative measures of software quality and reliability help to predict the resource which is required for prediction of specified software effort. Inaccurate effort prediction may lead to poor reliability and cost overruns. Due to overestimation of

effort, software development resources may lead to wastage while due to underestimation of effort, causes may be poor quality of software, associated penalties and schedule delays. Hence ANN architecture has been proposed with logarithmic activation function by adding some additional input layer. PSO(particle swarm optimization) is used for software effort perdition. For evaluation of effort prediction, genetics algorithm is applied which optimize the no. of hidden neurons in the hidden layer.

Estimation of effort is considered as a key in cost evaluation of software. There are various effort evaluation techniques that are available, for example algorithmic, empirical, theory-based, ML and regression. Numerous research work has been carried out using ML techniques. In recent years, for effort estimation various ML techniques like ANN, Bayesian Network, Case based reasoning, fuzzy logic, GA, SVM(support vector machine), regression tree have been proposed. We have carried out comparative study of several ML techniques like bagging, linear regression, KStar, M5Rules, RIP Tree and Multilayer Perceptron for predicting the software effort on different datasets.

Chapter 3

Research Methodology

3.1 Outlier Analysis

The extreme values that does not lie within the cluster of other observations are known as outliers. The outlier analysis is an important step performed before training of ML algorithms because if they are not removed the model may take more training time, less accurate model and hence poorer results. In our study, we have found two types of outlier known as Univariate and Multivariate. For identification of multivariate outlier, we can use Mahalanobis Jack knife distance, which is computed on the basis of distance between a data point in multidimensional space of every remark from the centre of mean [20] [21]. Here we have tested the effect of univariate outliers and multivariate outliers. If by eliminating one of the univariate outlier, the importance of the metric gets altered, i.e. if the fault metric get affected by the outlier then outlier has to be removed. Also, if the outlier presence or absence affects the significance of the one or more independent variable then that outlier has also to be eliminated. Further detailing of the outlier analysis are presented in [22].

3.2 Machine Learning Techniques

In our research, we have used some modern machine learning techniques available in WEKA tool. Here LR and various modern ML techniques like bagging, KStar, M5Rules, RIP Tree and MLP methods that have been successfully applied in various fields [23] [24].

3.2.1 Bagging

Bagging, a name comes from"bootstrap aggregation". It was very effective and simplest technique of ensemble learning. According to Breiman, meta-algorithm is one of the special case for model averaging which was initially proposed for classification and generally applies to decision tree models but at recent time it is used for regression or classification. Bagging uses different versions of a training data set in train a different model with the help of bootstrapping, i.e. sampling with replacement. For single output, output of all models is combined by voting (in case of classification) or averaging (in case of regression). Bagging is very effective if we used unstable nonlinear models because a minor change in the training data sets can cause an important change in the model.

So we can say that bagging just like a bootstrap aggregation which works as a technique of cumulative accuracy that frequently samples from particular dataset along with a uniform probability distribution. A sample size of each bootstrap is same as initial dataset since sampling is finished through substitution.

Certain instances which may seem numerous times in the identical training dataset, whereas others may possibly eliminate from the given training data set [20] [3]. Gathering of multiple predictors is the greatest significant feature of bagging. Bagging advance the machine learning algorithms which

are used for statistical classification and regression by improving their stability and accuracy. It decreases variance and avoids over-fitting [25].

3.2.2 Linear Regression (LR)

Linear Regression is a methodology which analyse the relationship between independent and dependent variables which is denoted as X and Y respectively. LR is one of the statistical method which is used for data regression with the help of dependent variables. Dependent variables have only continuous values while independent variables may have either categorical or continuous values. We can say in a different way, LR is a technique, used for prediction of the dependent variable on the basis of the values of independent variables and it is also used in the case of prediction of some continuous quantity. In the case of single independent variable, linear regression is known as simple linear regression. For this, LR discovers a line which decreases the summation of the squares of the vertical separations of the available points from that line. Means, LR is a procedure of approximating the conditional predictable value of single variable of Y which has given by the values of a number of extra variable of X [25].

Fundamentals

To begin with Linear Regression (LR), we must be known about some essential ideas of measurements. Such that

- Correlation (r), which describes the relationship between 2 variables and its values lies between [-1, +1].
- Variance (σ^2), which measure the spread of dataset.

- Standard deviation (σ), which measure the spread of dataset in the form of ($\sqrt{variance}$)
- Normal distribution
- Residual (i.e. Error), which is calculated as {Actual value - Predicted value}

Linear Regression Line

If we apply linear regression, our main objective is to fit a line which is closest to the majority of the points by distribution in the sample space. Thus we always try to decrease the separation (i.e. Error) between fitted line and data points.

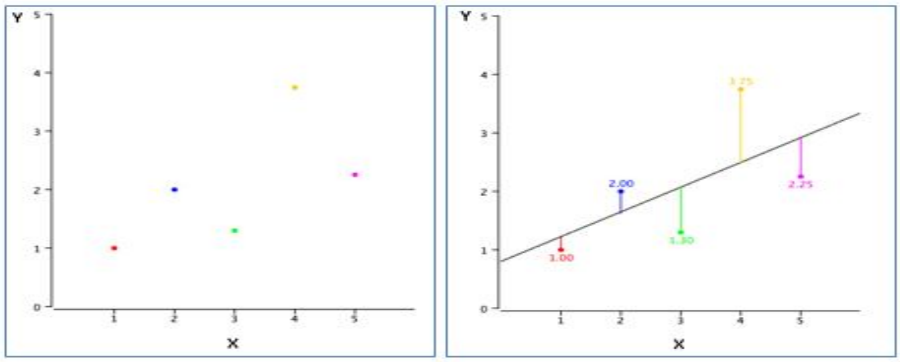

Figure 3.1: Linear Regression

In the left figure, the various data points are denoted by dots and line shown in the right figure, is denoted an approximate line. These lines describe the relationship between x and y axis. For finding this type of approximate line, we always prefer to use linear regression. For a single value of the dependent (Y) and independent (X) variable, the relation shown in between X and Y is given in the form of following linear equation.i.e.

$$Y = B0 + B1X \tag{3.1}$$

Where,
0 = Constant term which intercept y-axis.
1 = Coefficient of relationship between X and Y.

- B0 and B1 are two parameters, which specify the line and are to be estimated with the help of data sets.
- For the known values of Y1, Y2, ...X1, X2, ..., the least squares criterion is used.

Here B0 is used as intercept at y-axis through approximate line. In machine learning, it denoted as bias which added to the offset of all predictions that we make. B1 show the slope of a line or we can say, how the value of X is translated into the value of Y before the addition of bias.

3.2.3 KStar (K*)

KStar (K*) algorithm is a case based type classifier i.e. the class of test case depends on the class of those training case which is like it, which is determined through certain similar function. KStar is different from other case based learners since KStar measures distance function which is based on entropy. KStar works based on related procedures will have associated classification. This approach is slightly slower to other in evaluation but suitable for prediction.

3.2.4 REP(Reduces Error Pruning) Tree

It is one of the fastest decision tree learners which depend on the principle of calculating the "Information Gain" (IG) with Entropy (E) as well as reducing error arising due to variance. REP Tree generates multiple trees with the help of logic of regression tree in different iterations. Afterward, it always tries to pick the best one from available spawned trees. REP Tree makes a regression tree or decision tree with the help of variance or information gain

and REP tree uses reduced error pruning with back-fitting to prune these trees.

3.2.5 Multilayer Perceptron

Basically, a multilayer perceptron (MLP) is a feed-forward artificial neural model. There is single hidden layer lies between input-output layers. Its functionality is mapping the set of inputs to a particular set of outputs means data flow in one direction from input to the output layer. Each join is a neuron leaving the input joins and each join has an activation function (Y) corresponding to it. MLP is applying a back propagation algorithm for training the neural network. MLP is a revision of the standard linear perceptron and can solve the problem which is not linear-separable.

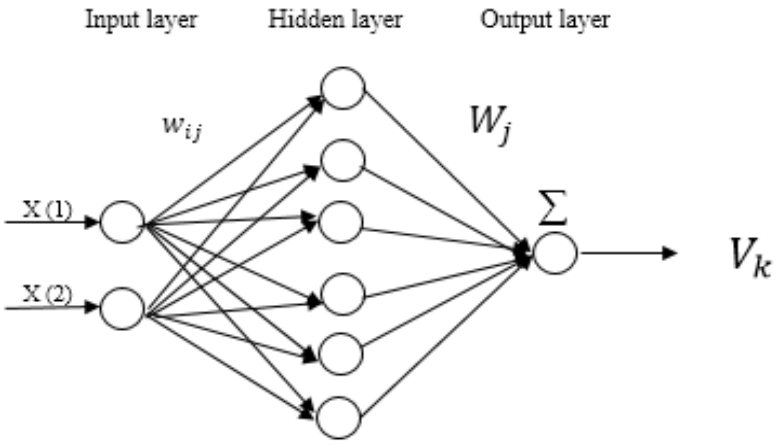

Figure 3.2: Multilayer Perceptron

3.2.6 M5Rules

M5Rules makes a decision list on behalf of regression problems via divide & conquer. In every iteration, M5Rules constructs a model tree with the help

of M5 and get to the "best" leaf into a rule. The approach of making rules from model trees is called M5Rules. In its work flow, a learner for the tree is operated to complete training set of data and after that, a pruned tree is found then the best branch is prepared into the rule and after that tree is thrown out. All instances which are covered by that rule are deleted from the dataset. The procedure is applied to remaining instances recursively and ends after covering every instance via one/more rules. This is a fundamental divide & conquer approach to learning rules and it is "best" branch as a rule.

Chapter 4

Research Background

4.1 Feature Sub Selection Method

We have used China dataset and Albrecht dataset which are taken from an available repository known as promise data repository. The China data set includes 19 feature in which it contains one dependent variable and eighteen independent variable and Albrecht dataset includes 8 feature in which it contain one dependent variable and seven independent variables. For making simple and efficient model, some independent variables are removed, which are playing less significant role in effort prediction from available independent variables. For reduction of independent variables, recently, there are so many techniques are available but we prefer to use a most significant technique called Feature sub selection which available in WEKA [26]. In China dataset, if we use Correlation Based Feature Sub selection (CFS), 19 independent variables are reduced to 10 independent variables in which only one variable is dependent variable and remaining 9 variables are independent variables and in Albrecht dataset, the 8 independent variables are reduced to 3 independent variables in which only one variable is dependent variable and remaining 2 variables are independent variables. Thus, CFS is used in dataset, to select the best one predictors out of available independent variables [27] [20]. So we always try to search for best combination of independent variable among all

possible combinations of independent variables. In available feature, effort is only dependent variable. Software effort is described as the work done by the software supplier from the starting point of specification to the last delivery at customer side, measured in hours [20].

4.2 Independent and Dependent Variables

In our research, software development effort is considered as a dependent variable. Actually, software effort is described as the work done by the software supplier from the starting point of the specification of the last delivery at customer side which measured in hours. Recently, most of the independent variables are categories in the China dataset and the Albrecht dataset [28]. The all independent variables are summarized in Table 1 and Table 2 for China dataset and Albrecht dataset respectively. In available data sets, to incorporate the independent variables based on correlation, CFS is used to select the best predictors among independent variables [11]. We always try to search the best combination of independent variable among all possible combinations of independent variables. CFS assesses the best of a subset of independent variables by considering the predictive ability of each feature accompanied by the degree of redundancy, present between them.

4.3 Empirical Data Collection

The dataset which we have used in our research, consists of 19 drivers for China and 8 for Albrecht of effort for effort prediction model. Here for our research, we used the descriptive statistics of six independent variables for China and two for Albrecht, chosen by CFS method are shown in Table 1 and Table 2. The mean value of the effort is found to be 3921 for China and 14.43 for Albrecht dataset which put in Table 1 and Table 2.

Table 4.1: China Dataset Statistics

Variables	Minimum	Maximum	Mean	Standard Deviation
ID	1	499	250	144.19
Output	0	2455	113.601	221.19
Interface	0	1572	24.23	85
Added	0	13580	360.35	829.84
Duration	1	84	8.71	7.34
N_effort	31	54620	4277.64	7071.25
Effort	26	54620	3921	6480.85

Table 4.2: Albrecht Dataset Statistics

Variables	Minimum	Maximum	Mean	Standard Deviation
Output	12	112	40.270	26
AdjFP	199	1572	540.501	340.59
Effort	0.5	61.2	14.43	13.67

4.4 Data Set

For our research purpose we have used two datasets namely China dataset and Albrecht dataset which has taken from promise dataset [28], are describes as follows:

4.4.1 Dataset 1: China Dataset

It consists of 19 attributes and 499 instances. Here 70% set of data is used for training and 30% set of data is used for validation. Correlation feature selection (CFS) technique is employed to choose the finest predictors among independent variables which are ID, Output, Interface, Added, Duration, N_effort in sets of data [20] [27] in the WEKA tool [26]. All the independent variables correspond to function point method.

4.4.2 Dataset 1: Albrecht Dataset

It consists of 8 attributes and 24 instances. Here 70% set of data is used for training and 30% set of data is used for validation. Correlation feature selection (CFS) technique is employed to choose the finest predictors among independent variables which are Output and AdjFP in sets of data [20] [27] in the WEKA tool [26]. The outlier is removed for finding the best result. All the independent variables correspond to function point method.

4.5 Estimation accuracy measures

In this work, we have employed the most effective and standard measures for estimation accuracy i.e. MMRE and PRED at intensity level 0.25, 0.50 and 0.75 respectively. Previous research carried out in this field have also used these two for performance measures. MMRE and PRED(x) value for a dataset consisting of N observations can be calculated in the following manner.

- MMRE: It is an extremely common condition used to estimate software cost estimation models. The MRE (magnitude of relative error) for every outcome i can be achieved as follows:

$$MRE_i = \frac{|\ Actual\ Effort_i - Predicted\ Effort_i\ |}{Actual\ Effort_i} \tag{4.1}$$

  MMRE can be obtained through the summation of MRE over N outcome:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{4.2}$$

- PRED (x): It can be expressed as the average fraction of the MREs off by no more than x as:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 1, & \text{if } MRE_i \leq x. \\ 0, & otherwise\ . \end{cases} \tag{4.3}$$

If

$$PRED(25) = \frac{M}{N} \tag{4.4}$$

Here M is the no. of outcomes whose MRE is less than or equal to 0.25, and N is the total number of outcomes for a particular dataset. In the same way, MRE of PRED(50) and PRED(75) is less than or equal to 0.50 and 0.75 respectively. Finally, the estimated accuracy is proportionate to PRED(x) and inversely proportionate to MMRE.

4.6 Cross validation

4.6.1 10-cross validation method

In this method, we obtained 10 equal size subsamples by partitioning the original sample randomly. Here in 10 sub-samples, 9 are using as training data and 1 is using for model testing. This method is repeated ten times and each one of the ten subsamples are using once as per validation data for testing. Result obtained from these 10 repetitions is averaged to get a single value. The main advantage of this method is that all the sub-samples are

used for both training and validation.

4.7 Tool used for result calculation

We have used WEKA tool in our research work. WEKA tool has embedded machine learning algorithms and it provides automatic pre-processing for our data and makes them more suitable as an input to our machine learning algorithms. Feature sub-selection technique has been used, given in WEKA [26] to decrease some independent variables. Other prominent features provided by WEKA tool are:

- Classification
- Clustering
- Association rule extraction
- Selection of attributes
- Visualization

4.8 Analyzed Model in WEKA

In existing study, we have found some limitations and from our study, we analyzed that most of the researcher does not follow the preprocessing step in their evaluation. Before preprocessing, researcher removes the missing and noisy data from their data sets. Besides these limitations, attribute selection is the another important limitation because the presence of unnecessary attributes may have create a problem in memory and affect our output results. So overcome from these limitations, we follow some basic steps. These are:

1. Input data in the form of data sets.
2. Preprocess the data
    (i) By removing noisy and missing data.
    (ii) By conversion.
    (iii) By removing outlier.
3. Apply attribute selection.
4. Apply machine learning techniques.
5. Result evaluation.

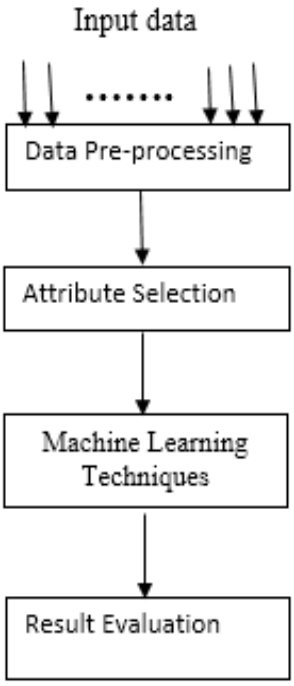

Figure 4.1: Analyzed model in WEKA

4.9 Classification of China dataset

4.9.1 Linear Regression

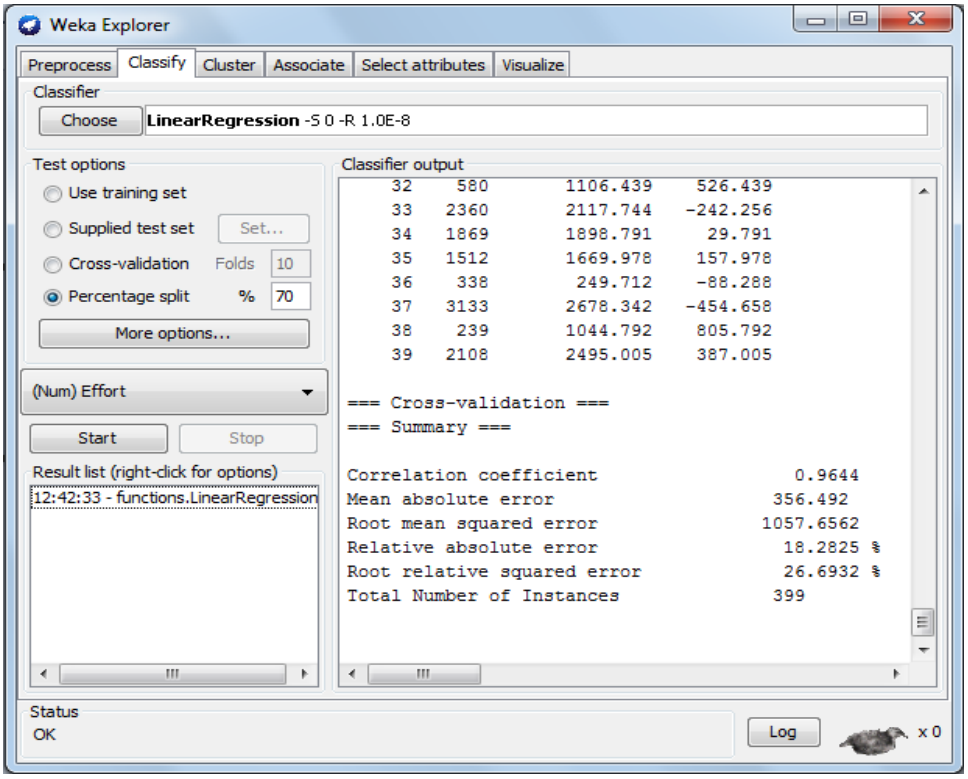

Figure 4.2: Linear Regression result in WEKA

Here we classify 399 instances of the china dataset. We apply the one of the machine learning technique known as linear regression for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9644 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.18 and 0.26 respectively.

4.9.2 Bagging

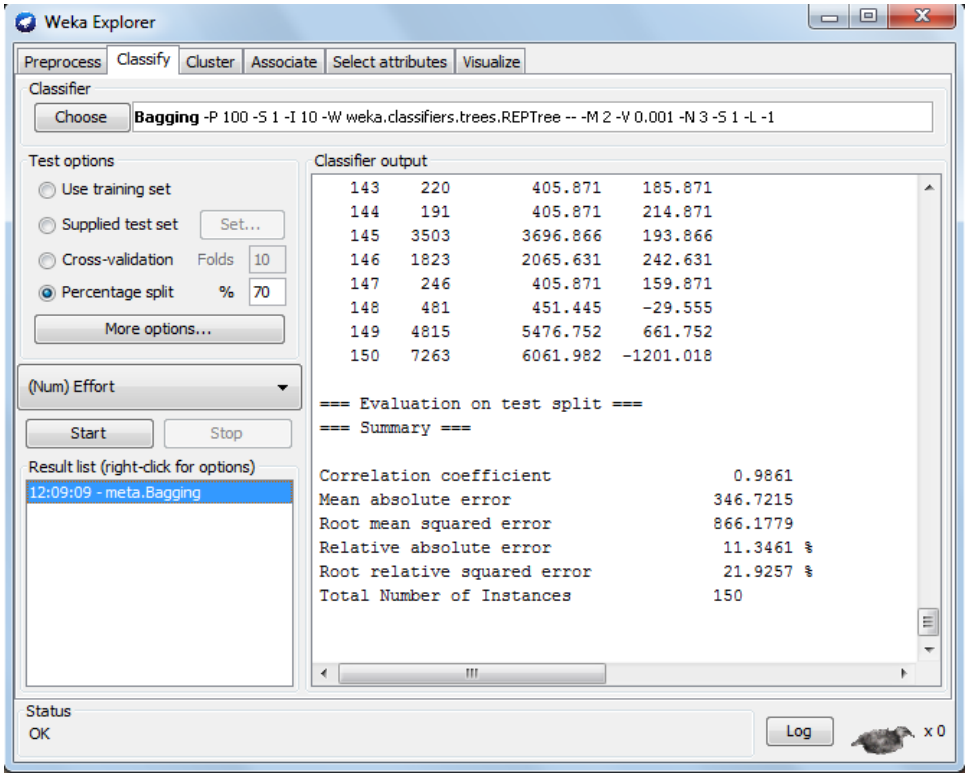

Figure 4.3: Bagging result in WEKA

Here we classify 150 instances of the china dataset. We apply the one of the machine learning technique known as bagging for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9861 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.11 and 0.21 respectively.

4.9.3 KStar

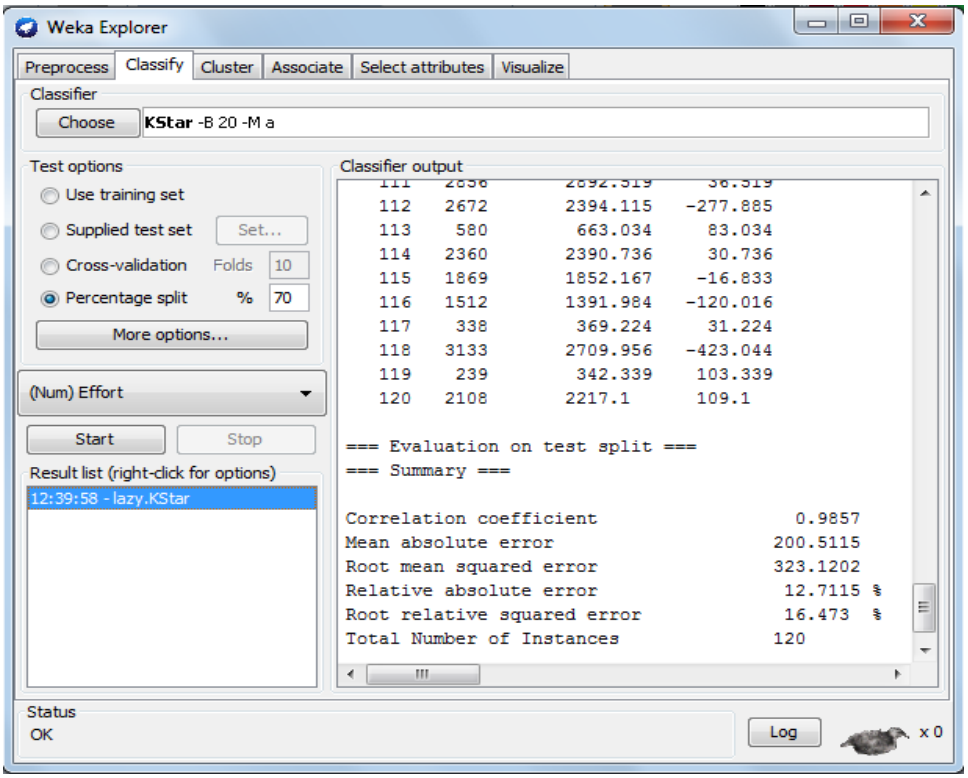

Figure 4.4: KStar result in WEKA

Here we classify 120 instances of the china dataset. We apply the one of the machine learning technique known as KStar for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9857 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.12 and 0.16 respectively.

4.9.4 M5Rules

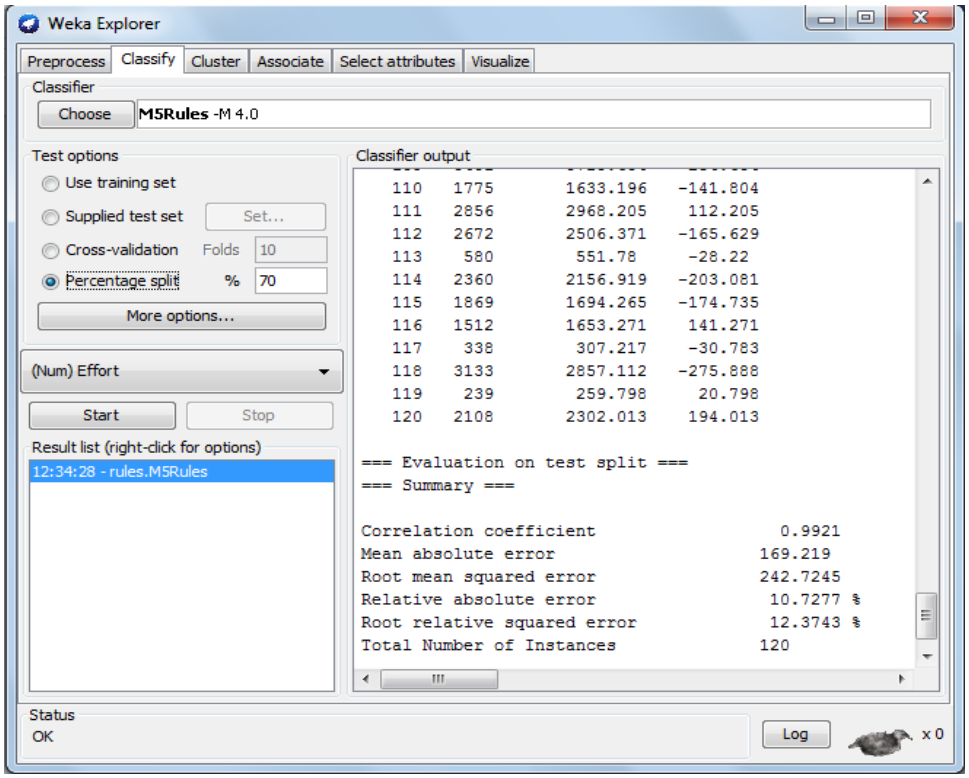

Figure 4.5: M5Rules result in WEKA

Here we classify 120 instances of the china dataset. We apply the one of the machine learning technique known as M5 Rules for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9921 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.10 and 0.12 respectively.

4.9.5 Multilayer Perceptron

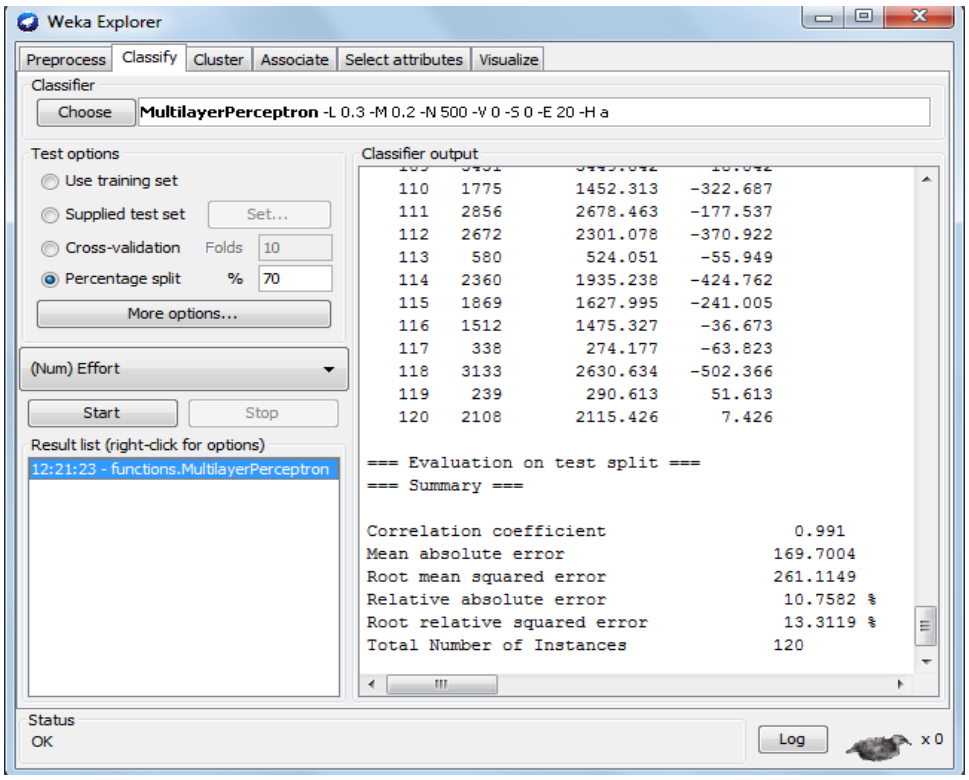

Figure 4.6: Multilayer Perceptron result in WEKA

Here we classify 120 instances of the china dataset. We apply the one of the machine learning technique known as Multilayer Perceptron for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.991 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.10 and 0.13 respectively.

4.9.6 REP Tree

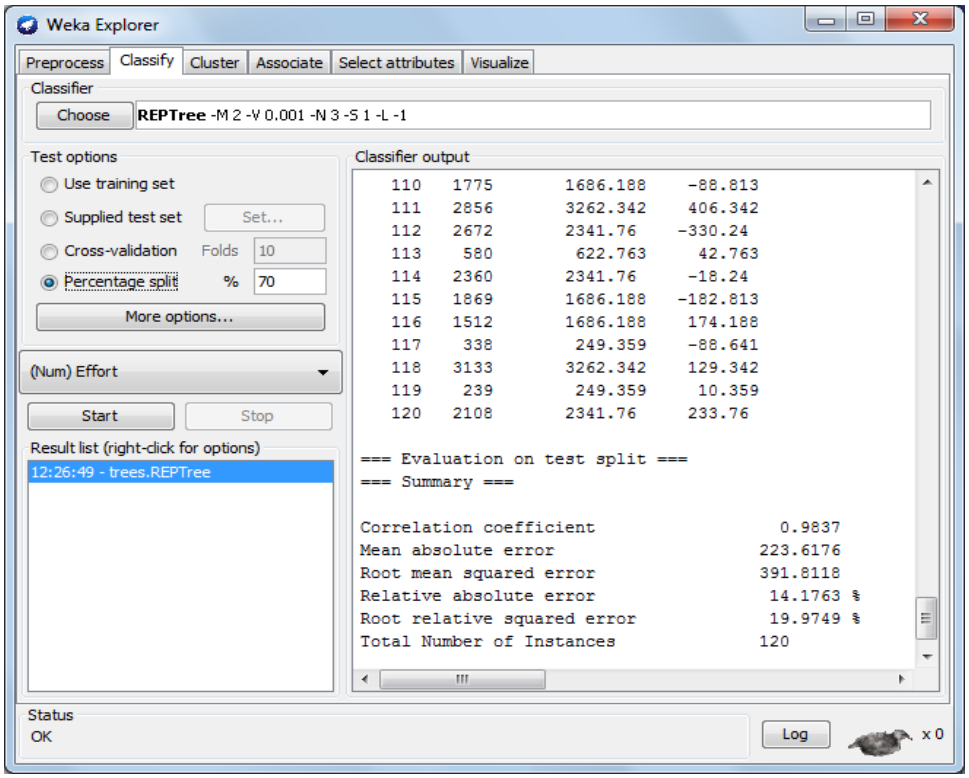

Figure 4.7: REP Tree result in WEKA

Here we classify 120 instances of the china dataset. We apply the one of the machine learning technique known as REP Tree for classification of the china dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9837 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.14 and 0.19 respectively.

4.10 Classification of Albrecht dataset

4.10.1 Linear Regression

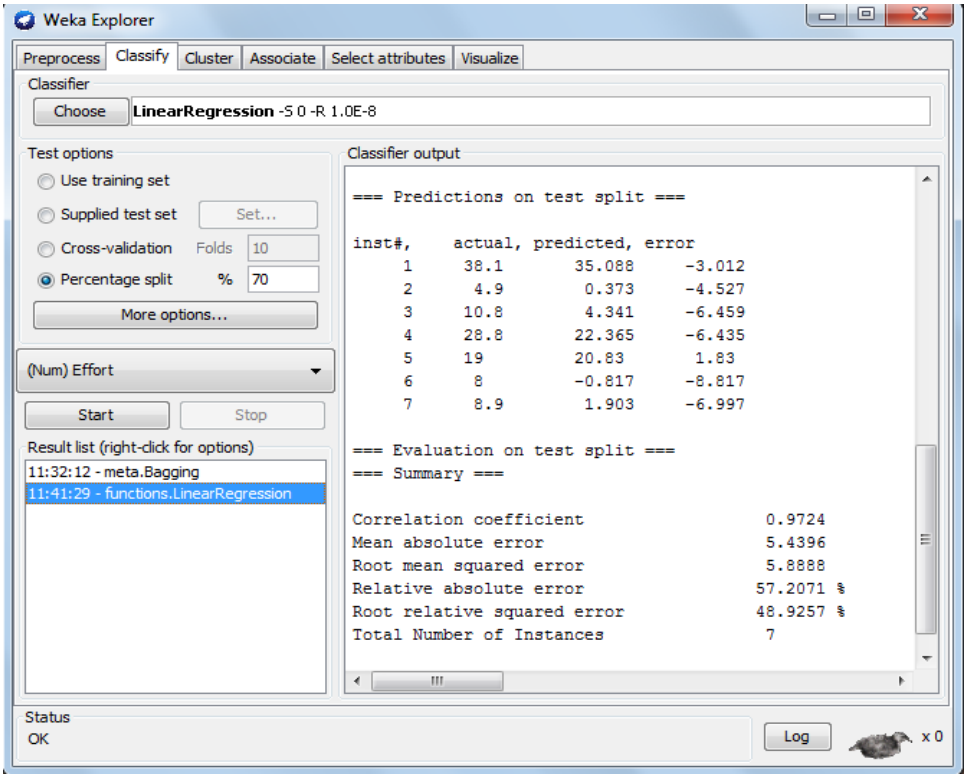

Figure 4.8: Linear Regression result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as linear regression for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.9724 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.57 and 0.48 respectively.

4.10.2 Bagging

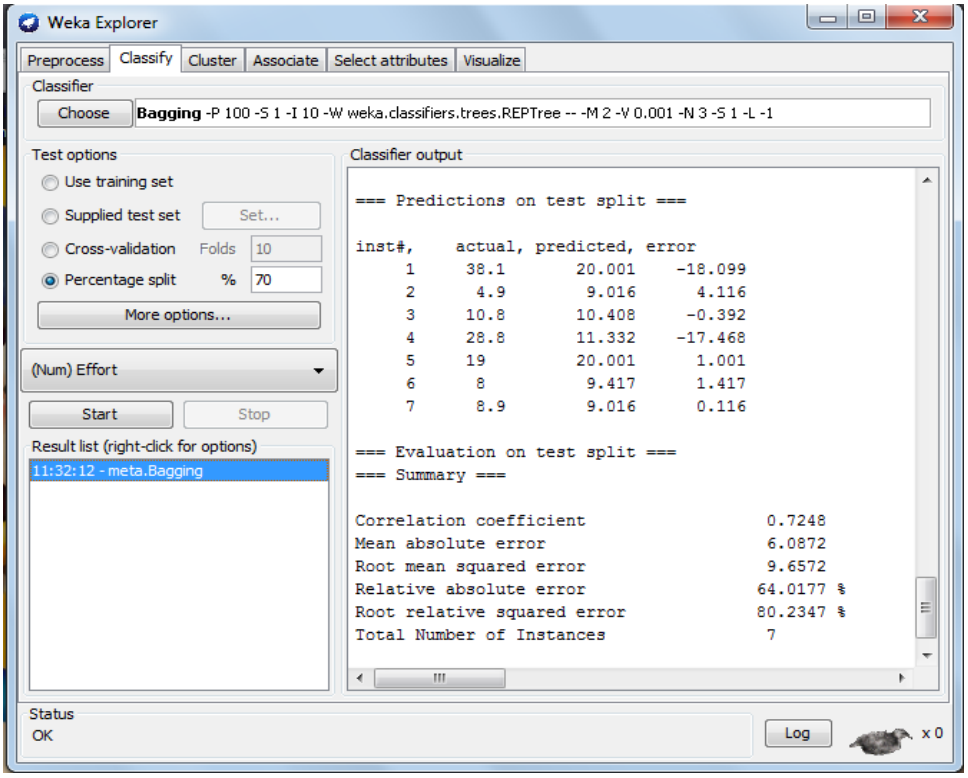

Figure 4.9: Bagging result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as bagging for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.7248 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.64 and 0.80 respectively.

4.10.3 KStar

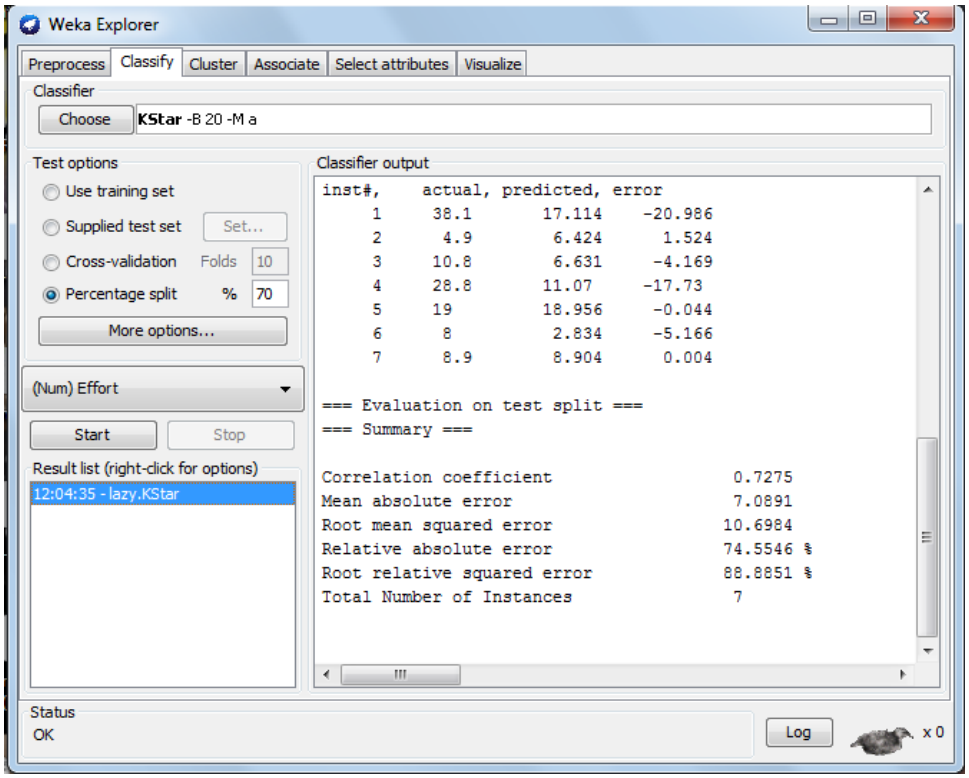

Figure 4.10: KStar result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as Kstar for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.7275 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.74 and 0.88 respectively.

4.10.4 M5Rules

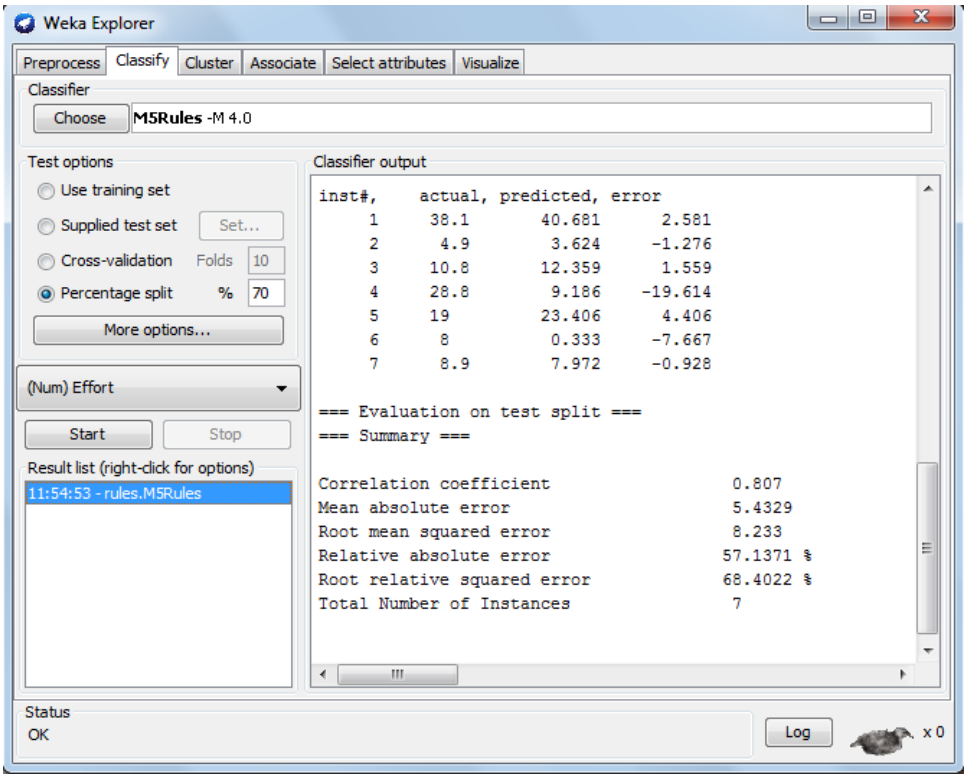

Figure 4.11: M5Rules result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as M5 Rules for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.807 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.57 and 0.68 respectively.

4.10.5 Multilayer Perceptron

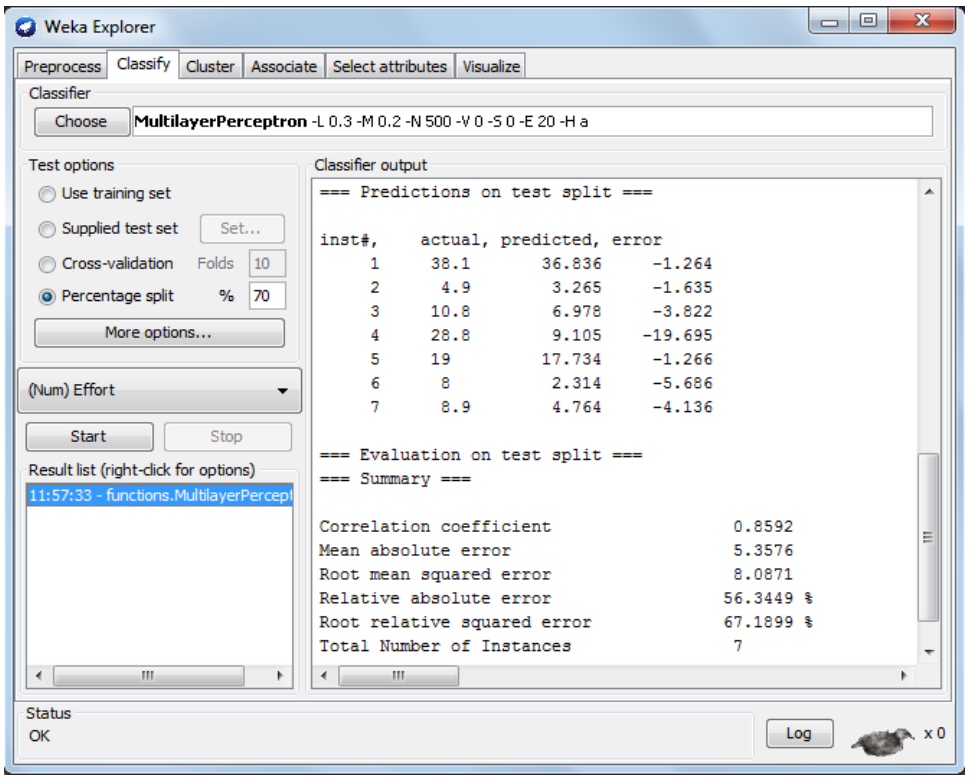

Figure 4.12: Multilayer Perceptron result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as Multilayer Perceptron for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.8592 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.56 and 0.67 respectively.

4.10.6 REP Tree

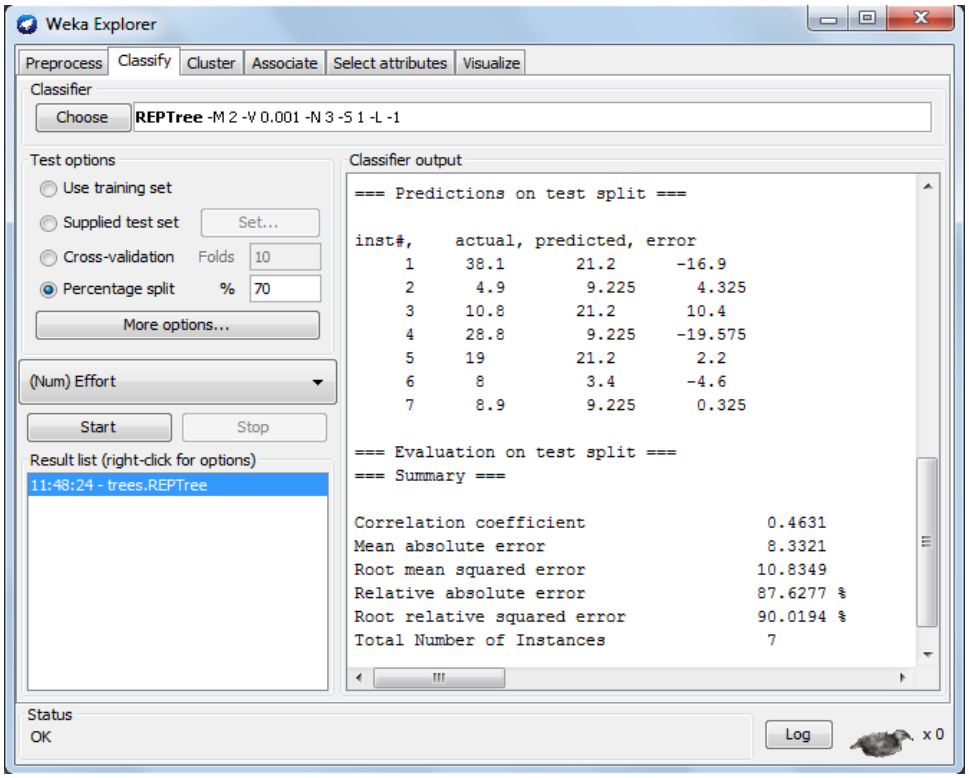

Figure 4.13: REP Tree result in WEKA

Here we classify 7 instances of the Albrecht dataset. We apply the one of the machine learning technique known as REP Tree for classification of the Albrecht dataset. After classification we find predicted value with respect to actual value. Here we also apply Percentage split in which 70% set of data is used for training and 30% set of data is used for validation. Also, after classification we find the most value of correlation coefficient i.e. 0.4631 and the low value of error in the form of relative absolute error and root relative squared error i.e. 0.87 and 0.90 respectively.

Chapter 5

Results and discussion

We were obtained result after the application of different machine learning algorithms on the two selected datasets which explained below with the help of tables and after that, we have explained the result in a detailed discussion [29].

5.1 Discussion of result with China dataset

Table 5.1: Analysis of result with China dataset

ML Techniques	MMRE	PRED(25)	PRED(50)	PRED(75)
Linear Regression	0.225266	0.763527	1	1
Bagging	0.168925	0.866667	0.953333	0.973333
KStar	0.129468	0.908333	0.975	0.991667
M5 Rules	0.123708	0.91984	0.963928	0.97996
Multilayer Perceptron	0.094517	0.958333	1	1
REP Tree	0.205582	0.764706	0.958824	0.970588

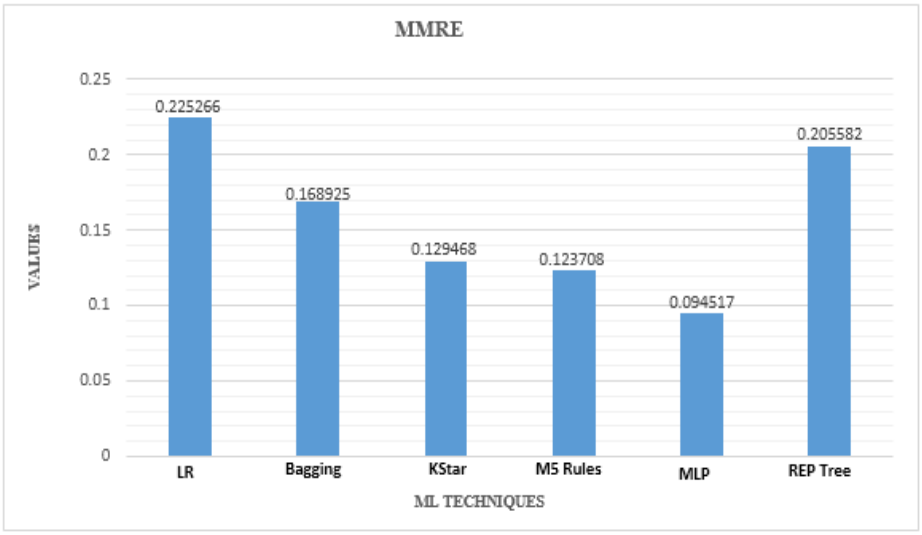

Figure 5.1: MMRE values for China dataset

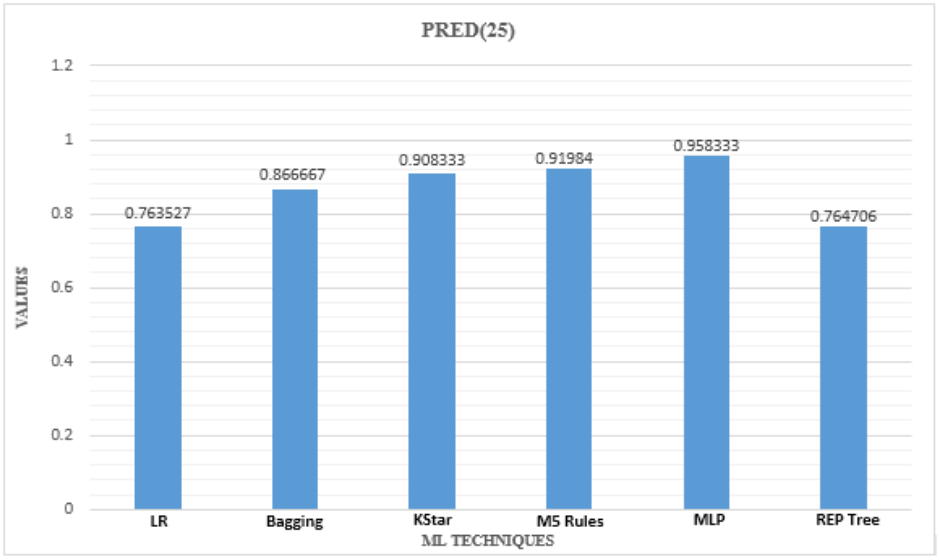

Figure 5.2: PRED(25) values for China dataset

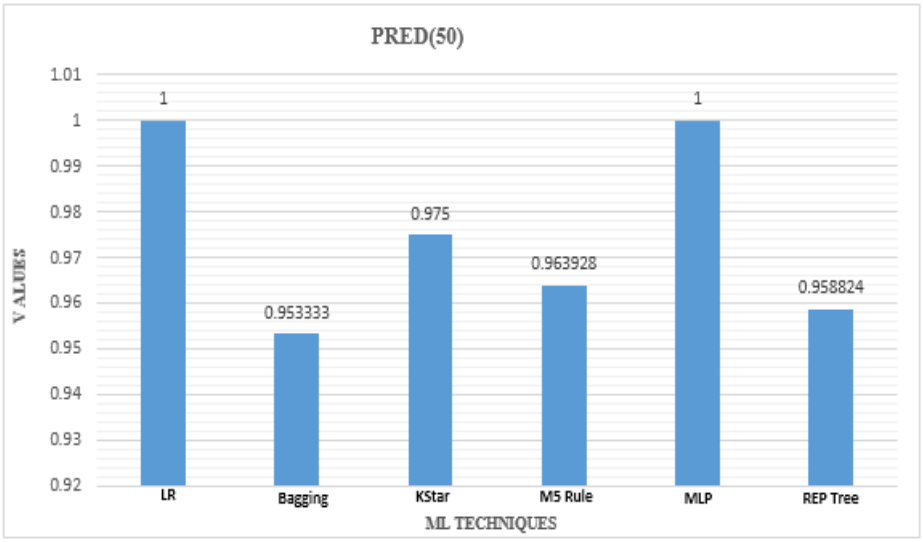

Figure 5.3: PRED(50) values for China dataset

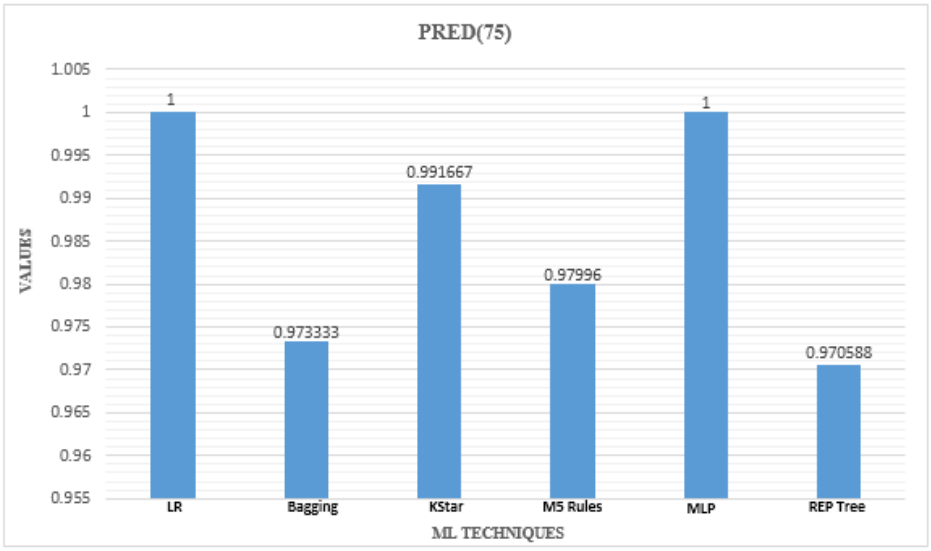

Figure 5.4: PRED(75) values for China dataset

Here we can see from above evaluation, mean magnitude of relative error is less for multilayer perceptron and M5 Rules which is 0.094517 and 0.123708 respectively. Also in terms of PRED(prediction) value, MLP and

M5 Rules give good result compared to other techniques having the maximum PRED(25) Value of 0.958333 and 0.91984, and also PRED(50) & PRED(75) value is 1 for MLP and PRED(50) & PRED(75) value are 0.963928 and 0.97996 for M5 Rule respectively. We know that for best accuracy, MMRE should be minimum and PRED(x) should be maximum. Hence on the basis of above result MMRE for multilayer perceptron is very low and Prediction is maximum, compare to other ML technique for China dataset. So multilayer perceptron is best ML technique for software effort prediction on china dataset. M5 Rules is also better ML technique which gives less MMRE and more PRED but not better than multilayer perceptron.

5.2 Discussion of result with Albrecht dataset

Table 5.2: Analysis of result with Albrecht dataset

ML Techniques	MMRE	PRED(25)	PRED(50)	PRED(75)
Linear Regression	0.544157	0.428571	0.428571	0.571429
Bagging	0.314408	0.571429	0.714286	0.857143
KStar	0.344807	0.428571	0.714286	1
M5 Rules	0.349732	0.571429	0.714286	0.857143
Multilayer Perceptron	0.378093	0.285714	0.714286	1
REP Tree	0.31212	0.428571	0.714286	0.857143

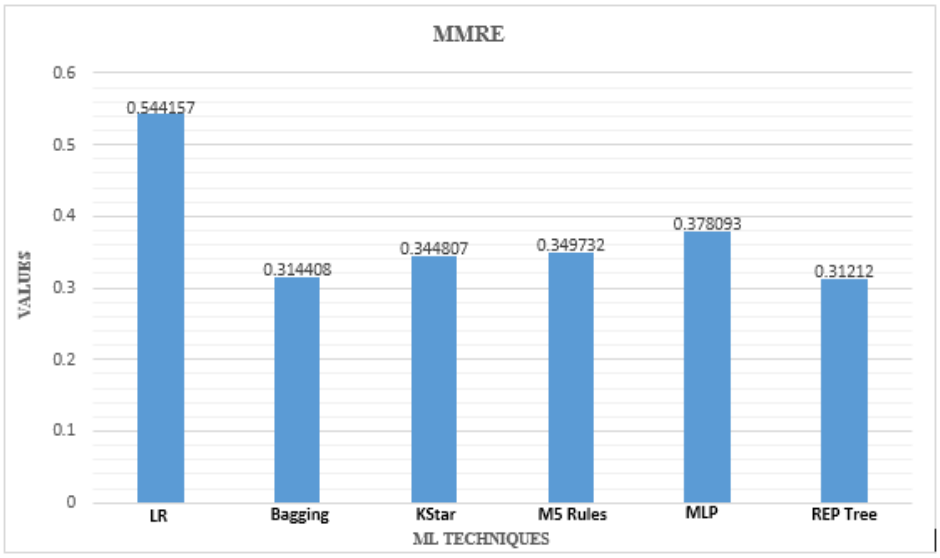

Figure 5.5: MMRE values for Albrecht dataset

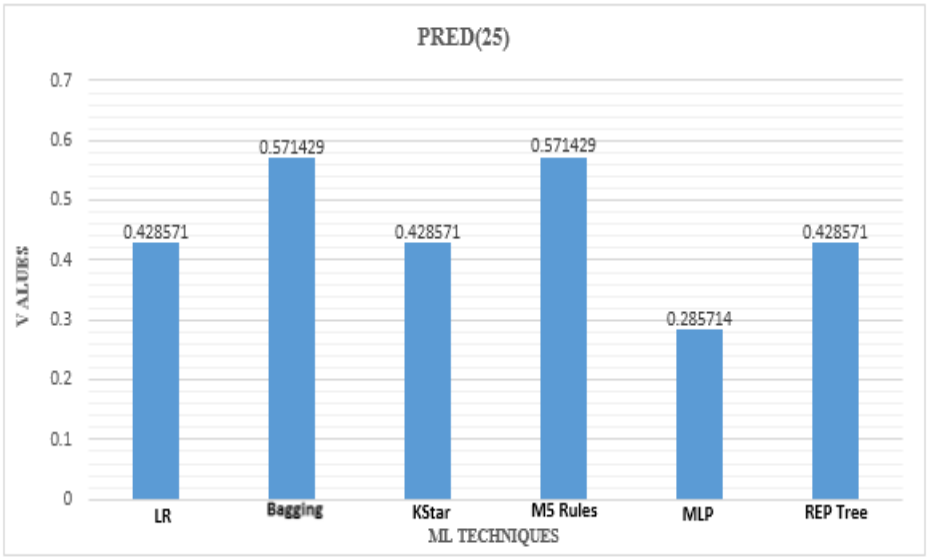

Figure 5.6: PRED(25) values for Albrecht dataset

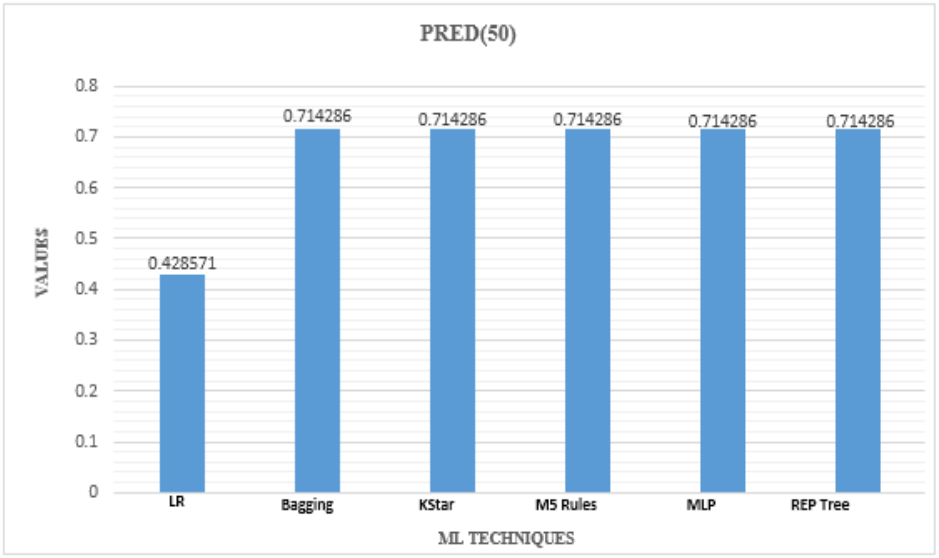

Figure 5.7: PRED(50) values for Albrecht dataset

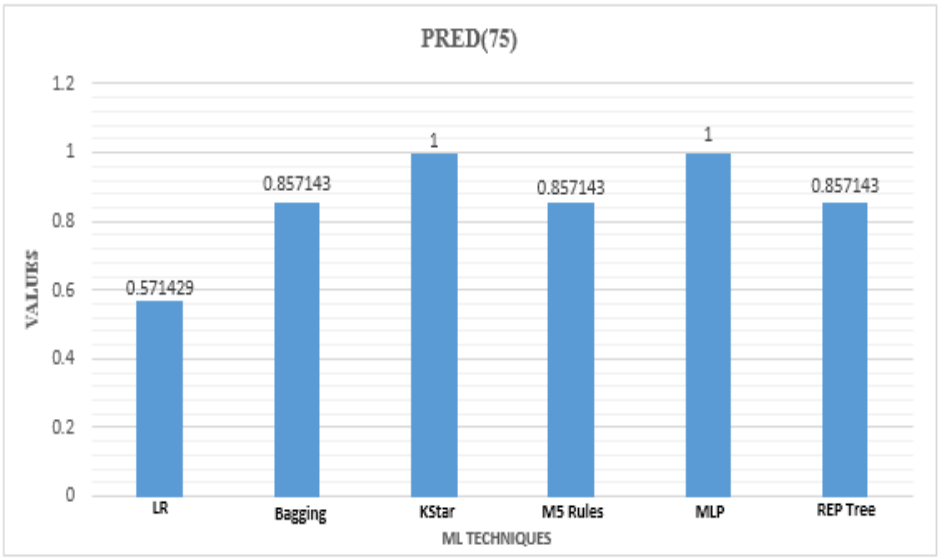

Figure 5.8: PRED(75) values for Albrecht dataset

Here we can see from above evaluation, mean magnitude of relative error is less for REP Tree and Bagging which is 0.31212and 0.314408 respectively. Also in terms of PRED(prediction) value, REP Tree and Bagging give good

result compared to other techniques having the maximum PRED(25) Value of 0.428571 and 0.571429, and also PRED(50) & PRED(75) value are 0.714286 and 0.857143 for REP Tree and PRED(50) & PRED(75) value are 0.714286 and 0.857143 for Bagging respectively. We know that for best accuracy, MMRE should be minimum and PRED(x) should be maximum. Hence on the basis of above result MMRE for REP Tree is very low and Prediction is maximum, compare to other ML technique for Albrecht dataset. So REP Tree is best ML technique for software effort prediction on Albrecht dataset. Bagging is also better ML technique which gives less MMRE and more PRED but not better than REP Tree.

5.3 Threats to validity

If we perform empirical analysis, it is very essential to consider the various threats to validity of the obtained results and conclusion [30]. So it can be, the possible source of bias to the results of the study is threats. This section express the various threats to validity of the study.

5.3.1 Threats to internal validity

The threats to internal validity comes with the causal effect of the independent variable on effort proneness attribute. In order to determine this effect, it is necessary to perform experiment where the independent variable are controlled to examine the causal effect on effort attribute. But it is very difficult to perform this experiment. Hence, our study was not to find the cause effect. So this threat exists in this study.

5.3.2 Threats to External validity

External validity takes into account the degree to which the outcome of the study are generalizable. As our study deals with the in-built machine learning algorithms of WEKA, the results would be easily acceptable. Also, the data-set is from open source repository which would help the replication of the study. However the result should be verified on other data mining tool like WEKA so that generalizability of the outcomes could be improved.

Chapter 6

Conclusion and Future Work

After examining the results with different datasets with the help of table and bar graph and using different machine learning techniques on original datasets, it can be inferred that the different datasets shown changed outcomes with altered techniques. The outcome that comes out depends on the data type to a great extent. Also, we can deduce that multilayer perceptron has shown good performance for China dataset and REP Tree shown for Albrecht dataset. So, as per our research Multilayer perceptron and REP Tree is good for estimating the software effort. The MMRE value of multilayer perceptron for China dataset and REP Tree for Albrecht dataset are 0.094517 and 0.31212 respectively.

In the future, we can perform this entire study for some software used in industry. Also, we can use evolutionary algorithms or techniques like a genetic algorithm, particle swarm optimization and bacterial foraging on the same data and check out whether there is any improvement shown in performance or not. We can also check out whether our project is economically feasible or not by estimating the cost based on predicted effort.

References

[1] I. F. de Barcelos Tronto, J. D. S. da Silva, and N. SantAnna, "An investigation of artificial neural networks based prediction systems in software project management," *Journal of Systems and Software*, vol. 81, no. 3, 2008.

[2] A. R. Gray, S. G. MacDonell, and M. J. Shepperd, "Factors systematically associated with errors in subjective estimates of software development effort: The stability of expert judgment," in *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403)*, 1999.

[3] R. Malhotra and A. Jain, "Software Effort Prediction using Statistical and Machine Learning Methods," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 2, no. 1, 2011.

[4] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," *Empirical Software Engineering*, vol. 11, no. 4, 2006.

[5] R. K. Smith, J. E. Hale, and A. S. Parrish, "An empirical study using task assignment patterns to improve the accuracy of software effort estimation," *IEEE Transactions on Software Engineering*, vol. 27, no. 3, 2001.

[6] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *Journal of Systems and Software*, vol. 80, no. 4, 2007.

[7] A. Tosun, B. Turhan, and A. B. Bener, "Feature weighting heuristics for analogy-based effort estimation models," *Expert Systems with Applications*, vol. 36, no. 7, 2009.

[8] G. Finnie, G. Wittig, and J.-M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models," *Journal of Systems and Software*, vol. 39, no. 3, 1997.

[9] G. R. Finnie and G. E. Wittig, "Ai tools for software development effort estimation," in *Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96)*, ser. SEEP '96, Washington, DC, USA: IEEE Computer Society, 1996.

[10] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," *Software Quality Journal*, vol. 14, no. 2, 2006.

[11] M. O. Elish, "Improved estimation of software project effort using multiple additive regression trees," *Expert Syst. Appl.*, vol. 36, no. 7, Sep. 2009.

[12] C. J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? a comparative evaluation," *Information and Software Technology*, vol. 43, no. 14, 2001.

[13] P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, "Software effort estimation using machine learning techniques with robust confidence intervals," in *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, vol. 1, 2007.

[14] C. L. Martin, J. L. Pasquier, C. M. Yanez, and A. G. Tornes, "Software development effort estimation using fuzzy logic: A case study," in *Sixth Mexican International Conference on Computer Science (ENC'05)*, 2005.

[15] S. Bibi and I. Stamelos, "Selecting the appropriate machine learning techniques for the prediction of software development costs," in *Artificial Intelligence Applications and Innovations: 3rd IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI) 2006, June 7–9, 2006, Athens, Greece*, I. Maglogiannis, K. Karpouzis, and M. Bramer, Eds. Boston, MA: Springer US, 2006.

[16] J. J. C. Gallego, D. Rodríguez, M. Á. Sicilia, M. G. Rubio, and A. G. Crespo, "Software project effort estimation based on multiple parametric models generated through data clustering," *Journal of Computer Science and Technology*, vol. 22, no. 3, 2007.

[17] P. C. Pendharkar, "Probabilistic estimation of software size and effort," *Expert Syst. Appl.*, vol. 37, no. 6, Jun. 2010.

[18] L. Radlinski and W. Hoffmann, "On predicting software development effort using machine learning techniques and local data on predicting software development effort using machine learning techniques and local data," Jul. 2017.

[19] M. Bisi and N. K. Goyal, "Software development efforts prediction using artificial neural network," *IET Software*, vol. 10, no. 3, 2016.

[20] R. Malhotra, A. Kaur, and Y. Singh, "Application of machine learning methods for software effort prediction," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, May 2010.

[21] J. F. Hair Jr., R. E. Anderson, R. L. Tatham, and W. C. Black, *Multivariate Data Analysis (4th Ed.): With Readings*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[22] R. Pincus, "Barnett, v., and lewis t.: Outliers in statistical data. 3rd edition. j. wiley & sons 1994, xvii. 582 pp., 49.95," *Biometrical Journal*, vol. 37, no. 2, 1995.

[23] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, 1995.

[24] E. N. Regolin, G. A. de Souza, A. R. T. Pozo, and S. R. Vergilio, "Exploring machine learning techniques for software size estimation," in *23rd International Conference of the Chilean Computer Science Society, 2003. SCCC 2003. Proceedings.*, 2003.

[25] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, 1996.

[26] *Weka 3 - data mining with open source machine learning software in java*, `http://www.cs.waikato.ac.nz/ml/weka/`, (Accessed on 07/20/2017).

[27] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.

[28] G. BOETTICHER, "The promise repository of empirical software engineering data," *http://promisedata.org/repository*, 2007.

[29] Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," *Software quality journal*, vol. 18, no. 1, 2010.

[30] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.