ABSTRACT

Software reliability is an indispensable part of software quality and is one among the most inevitable aspect for evaluating quality of software product. Software Industry endures various challenges in developing highly reliable software. We use the ensemble methods and machine learning techniques for software reliability predictions and evaluate them based on selected performance criteria. Ensemble learning, a machine learning paradigm where multiple base learners are trained to solve the problem. Here a set of hypotheses are constructed and combined to give improved results. Software reliability modelling based on test data is done to estimate whether the current reliability level meets the requirements for the product software reliability modelling also provides possibility to predict the reliability of the modules in a software.

In this proposed work we have used ensemble methods on various machine learning approaches and study the ensemble performance using bagging, boosting and stacking. Using ensembling, a module in a software is classified whether to have faults or not based on certain attributes of the software module and predictions of the next failure in the software given the Mean Time Between Failure.

Contents

Chapter 1

INTRODUCTION

Software reliability is defined as “The ability of the software to perform its required function under stated conditions for a stated period of time”. With the rapid growth and increasing complexity of the software, the reliability of the software is hard to achieve. Reliability is one of the important and crucial aspect and attribute of the software quality. According to ANSI, software reliability is ‘‘The probability of failure free operation of a computer program for a specified period in a specified environment’’ (Quyoum et al. 2010). In this proposed work, the model has been used for predicting and estimating the number of errors in the software. Also, classification in also done in this work to classify the error to a desired class of output. The primary goal of software reliability modelling is to find out the probability of a system failing in given time interval or the expected time span between successive failures.

Machine Learning (ML) techniques have proved to be successful in predicting better results that statistical methods and can be used for prediction and classification of software failures more accurately and precisely. ML is an approach which is focused on learning automatically and allows computers to evolve and predict and classify the system behaviour based on past and the present failure data. Thus, it is quite natural to know that which method tends to work well for a given failure dataset and up to what extent quantitatively (Aggarwal et al. 2006; Goel and Singh et al. 2009).

In this proposed work, we present an ensemble of various ML techniques such as Bagging, Boosting & Stacking for predicting and classifying software reliability based on two industrial datasets. Thereafter, we check about the accuracy and performances of ML based models in predicting and classifying software reliability when applied to past failure week data.

Chapter 2

Work Objective

Business applications which are critical in nature require reliable software, but developing such software is a key challenge which software industry faces today. With the increasing complexity of the software these days, the reliability of the software is hard to achieve. Ensemble method has been used for predicting, estimating and classifying number of errors remaining in the software. The primary goal of software reliability modelling is to find out the probability of a system failing in given time interval or the expected time span between successive failures. In this work, ML techniques used for predicting software reliability prediction and classification are SVM, KNN, Random Forest, Decision Tree, Linear Regression, Logistic Regression, Bayesian Ridge Regression, Lasso Regression, ElasticNet Regression, ANN, Naïve Bayes algorithm, SVR, bagging, boosting & stacking. The mentioned machine learning methodologies are used together on two different datasets.

- **Prediction:**
  For predicting software reliability, the dataset of successive failures of the software's is used and different above-mentioned ML techniques is used for prediction of the failure time of the software reliability dataset. The value is predicted based on the Mean Time Between Failure (MTBF) using a single feature dataset representing the meantime between failures in chronological order. To perform prediction, the cumulative MTBF is calculated for bagging and hence, the subsequent errors are predicted. In prediction, set of hypotheses are combined to give the better accuracy and improved result.

- **Classification:**
  In classification problem, based on certain parameters of a module in a software like cyclomatic complexity, essential complexity, design complexity and number of lines etc. , the module of a software needs to be classified whether it would have one or more reported defects or not. Using Ensemble learning for the classification problem is based on different types of classifiers mention above.

Chapter 3

Literature Survey

Several ML techniques have been proposed and applied in the literature for software reliability modelling. Some of the techniques are Generic Programming, Gene Expression Programming, Artificial Neural Network, Decision Trees, Support Vector Machines, Feed Forward Neural Network, Fuzzy models, Generalized Neural Network etc (Malhotra et al. 2011; Xingguo and Yanhua 2007; Hua Jung 2010; Karunanithi et al. 1992; Singh and Kumar 2010d; Eduardo et al. 2010).

3.1 The Connectionist Model

The artificial neural networks, also known as connectionist models, represent an innovative technology that is rooted in many disciplines. It essentially tries to develop the underlying mapping function of the process by "learning" based on a series of input patterns.

However, despite its highly successful applications in such diverse fields as modelling, process identification and control, pattern recognition, speech and signal processing, classification, etc., its use in reliability data analysis and prediction is not widespread. A relatively novel technique in software reliability growth prediction(Karunanithi et al. 1992), have demonstrated that neural networks models performed better than parametric models and that the types of network architectures can significantly influence the predictive performance. In recent years, there have been some revived interests in exploring the use of neural models for software quality and reliability.

For an input stream $\{X_t\}$ consisting of history of the random variable being studied, under the assumption of no external variables, the general neural network structure is essentially a nonlinear model

$$Y_{t+1} = F(X_t, X_{t-1}, X_{t-2}\ldots\ldots\ldots\ldots\ldots X_{t-p}) \qquad [eq^n 1]$$

where Y_{t+r} denotes the output target, F is the underlying mapping function, and there are p+ 1 input neurons. From a statistical perspective, in time series forecasting, the stochastic behaviour of the output is given by the conditional expectation

$$E[Y_{t+1} \mid X_t, X_{t-1}, X_{t-2}\ldots\ldots\ldots\ldots\ldots X_{t-p}], \qquad [eq^n 2]$$

which is the minimum mean square error predictor. One of the most popular neural network models is the multilayer feedforward network.

Suppose that there are k input neurons connected to a single neuron i. A simple form of this dynamic feedforward neural model has the following relationship:

$$U_j(t+1) = F\{\sum_{i=1}^{n} w_{ji} G[u_i(t)]\} \qquad [eq^n 3]$$

where $u_j(t)$ is the activation of neuron j at time t; G is the threshold function; w_{ji} is the connection weight from the input neuron i to neuron j in the next layer; F defines the type of transfer function under consideration. We can easily extend this concept to a multilayer network structure consisting of many neurons in the hidden layers. It can also be shown that for any feedforward network with an input vector X of n neurons, i.e., $\{x_1, x_2, \ldots, x_n\}$, and a hidden layer of k neurons, there exists a network output function g(X, Θ) that can provide an accurate approximation to any function of input vector scaled between 0 and 1. Under the assumption of linear threshold function, the generated output signal will be

$$g(X,\Theta) = F\{\sum_{j=0}^{k} v_j f(\sum_{i=0}^{n} w_{ji} x_i)\} \qquad [eq^n 4]$$

where Θ represents the overall weight vector, and v_j denotes the connecting weights between the output neuron and neuron j in the hidden layer.

3.2 Statistical efficiency Measures

To validate the proposed reliability prediction models have used several efficiency measures (Jaiswal-Malhotra et al. 2016). The precise view of overall prediction methodology adopted is being illustrated through a flow char Fig. 1. To conduct this experiment, foremost requisite is the selection of independent and dependent variables. The dependent variable which have been used in this research paper(Jaiswal-Malhotra et al. 2016) was failure rate and the independent variable used was time interval in terms of weeks.

For the corresponding failures, testing time in terms of weeks is chosen to be independent variable.

Su and Huang (2006) had applied neural network for predicting software reliability. Pai and Hong (2006) performed experiments using SVMs for forecasting software reliability. Kumar and Singh (2012) discusses about the usage of machine learning techniques like cascade correlation neural network, decision trees and fuzzy inference system to predict the reliability of software products.

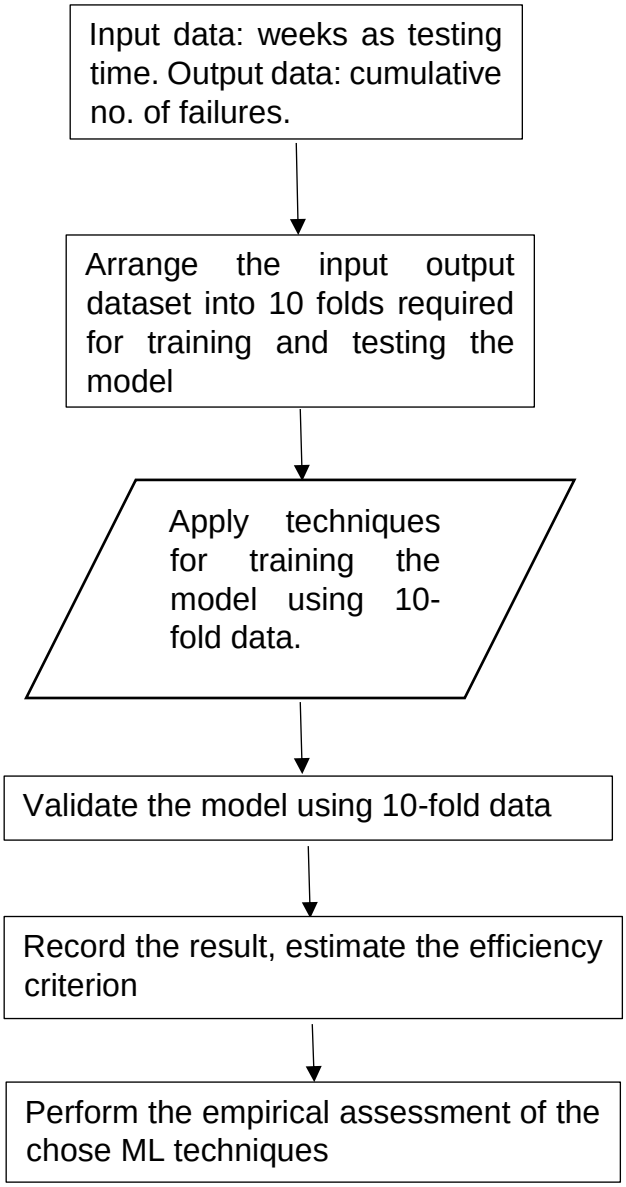

Fig. 1. Flowchart of software reliability prediction methodology adopted.

Chapter 4

Work Background

In this section, it is described about the two dataset that is used in this proposed work.

For prediction of time of failure, dataset used: Musa, J.D: Software reliability data, IEEE Computer Society Repository(1979). Failure data during system testing phase of various projects collected at Bell Telephone Laboratories, Cyber security and Information Systems Information Analysis Centre(CSIAC) by John D. Musa are considered.

For classification, the software reliability classification dataset used : Source : NASA, then the NASA Metrics Data Program, http://mdp.ivv.nasa.gov. Mike Chapman, Galaxy Global Corporation

1. *loc* : McCabe's line count of code
2. *v(g)* : McCabe "cyclomatic complexity"
3. *ev(g)* : McCabe "essential complexity"
4. *iv(g)* : McCabe "design complexity"
5. *n* : Halstead total operators + operands
6. *v* : Halstead "volume"
7. *l* : Halstead "program length"
8. *d* : Halstead "difficulty"
9. *i* : Halstead "intelligence"
10. *e* : Halstead "effort"
11. *b* : Halstead value
12. *t* : Halstead's time estimator
13. *lOCode* : Halstead's line count
14. *lOComment* : Halstead's count of lines of comments
15. *lOBlank* : Halstead's count of blank lines
16. *lOCodeAndComment* : IO lines
17. *uniq_Op* : unique operators
18. *uniq_Opnd* : unique operands
19. *total_Op* : total operators
20. *total_Opnd* : total operands
21: *branchCount* : of the flow graph
22. *prediction* : {false,true} module has one or more reported defects or not

Chapter 5

ENSEMBLING METHODS & MACHINE LEARNING TECHNIQUES

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent algorithms alone. Machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives. Supervised learning algorithms are most commonly described as performing the task of searching through a hypothesis space to find a suitable hypothesis that will make good predictions with a problem. Even if the hypotheses space contains hypotheses that are very well-suited for a problem, it may be very difficult to find a good one. Ensembles combine multiple hypotheses to form a better hypothesis. The term ensemble is usually reserved for methods that generate multiple hypotheses using the same base learner. The broader term of multiple classifier systems also covers hybridization of hypotheses that are not induced by the same base learner.

Evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model, so ensembles may be thought of to compensate for poor learning algorithms by performing a lot of extra computation. Fast algorithms such as decision trees are commonly used in ensemble methods (for example, random forests), although slower algorithms can benefit from ensemble techniques as well. An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to over-fit the training data more than a single model would, but in practice, some ensemble techniques (especially bagging) tend to reduce problems related to over-fitting of the training data.

Empirically, ensembles tend to yield better results when there is a significant diversity among the models. Many ensemble methods, therefore, seek to promote diversity among the models they combine. Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees). Using a variety of strong learning algorithms, however, has been shown to be

more effective than using techniques that attempt to *dumb-down* the models to promote diversity.

While the number of component classifiers of an ensemble has a great impact on the accuracy of prediction, there is a limited number of studies addressing this problem. *A prior* determining of ensemble size and the volume and velocity of big data streams make this even more crucial for online ensemble classifiers. Mostly statistical tests were used for determining the proper number of components. More recently, a theoretical framework suggested that there is an ideal number of component classifiers for an ensemble such that having than this number of classifiers would deteriorate the accuracy. It is called "the law of diminishing returns in ensemble construction." Their theoretical framework shows that using the same number of independent component classifiers as class labels gives the highest accuracy.

5.1 TYPES OF ENSEMBLE METHOD

There are different types of ensemble methods that can be used in a machine learning model. But in this project generally three ensemble methods have been used to develop this project.

5.1.1 Bootstrap Aggregating (Bagging)

Bootstrap aggregating, often abbreviated as *bagging*, involves having each model in the ensemble vote with equal weight. To promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy.

In Sampling with replacement given a dataset length of d and then k number of datasets are created where a set of data tuples are randomly picked from the dataset in which we get 66% non-redundant data and 33% are redundant data. The whole dataset is applied for testing the model and then the model predicts or classifies the data and then it gives us the output. For single instance of the prediction of problem Lag method is used to generate a set of lagged and combined value for the machine learning prediction problem. The lag Algorithm in this work is as follows:

Algorithm 1 Lagging

- *Input: Dataset, lag sequence(k, e.g.: k=3 or 4 or 5 etc)*
- *Algorithm:*
    1. *Length = length of the input dataset*
    2. *for i in 0 to Length-k+1:*
    3. *Store data from i to i+k index of the dataset in the list z*
    4. *return z*
- *Output: array with lagged value from the original dataset in this case z list*

Bagging is implemented as follows in this project: Considering a dataset D having d number of instances as a classification problem, the pseudo code for Bagging is as follows:

Algorithm 2 Bagging

- *Input: Dataset (K train dataset D_1 to D_k by sampling with replacement), d number of instances*
- *Training:*

    1. for i in 1 to k
    2. Di = lagged the dataset using Lag algorithm[Algorithm 1]
    3. For j in 1 to d
    4. r = random number between 1 to d
    5. Store D[r] in Di
    6. Return Di

- *Output: D[ ]*

- *Train K Classifiers on respective dataset bases on a model:*
  Here different classifiers models are used such as SVM, Naïve Bayes, Decision Tree, Random Forest, KNN, ANN, Logistics regression.
  Then:

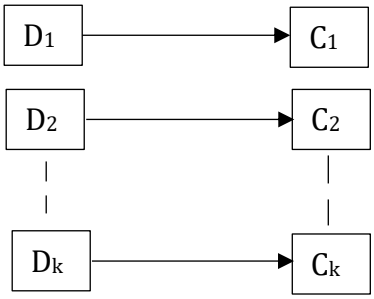

Fig. 2. *Training of different ML model on sample with replacement dataset*

- *To classify an instance, that instance is classified with K classifiers and maximum voting is done to classify the instance:*

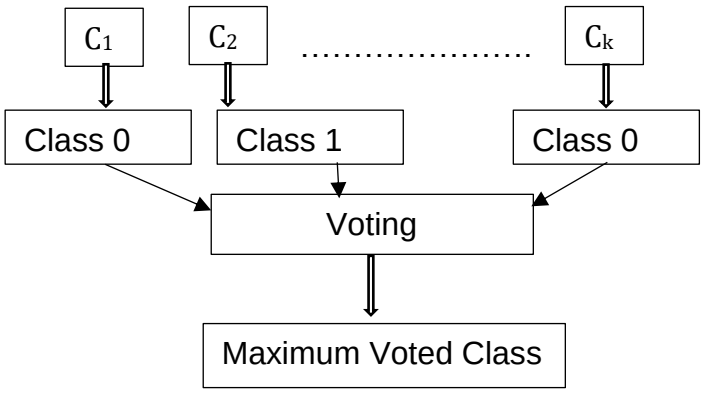

Fig. 3. *Classification on different instance & voting*

5.1.2 Boosting

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models' mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. By far, the most common implementation of boosting is Adaboost, although some newer algorithms are reported to achieve better results. In this work Adaboost algorithm is used to improve the accuracy of the weak classifiers. A boosting algorithm create an ensemble of classifiers, each one gives a weighted vote.

Algorithm 3 Boosting

- *Input:*
  D (set of d training tuples), K (number of round-one classifier is generated per round), a classification learning scheme
- *Method:*
  1. *The weight of each tuple in D is initialized to 1/d*
  2. *For i = 1 to k do // for each round:*
  3. *Sample D with replacement acc. to the tuple weights*
  4. *Training set D_i is used to derive a model M_i*
  5. *Error(M_i) = $\sum_{j=1}^{d} w_j$*err(X_j)/sum(w)*
  6. *if Error(M_i) > 0.5 then*
  7. *Go back to step 3 & try again*
  8. *End if*
  9. *For each tuple in Di that was correctly classified*
  10. *Multiply weight of tuple with Error(M_i)/(1-Error(M_i))*
  11. *Normalize the weight of each tuple*
  12. *End For*
- *To use the ensemble to classify tuple, X:*
  1. *Initialize weight of each class to 0*
  2. *For i = 1 to k to // for each classifier:*
  3. *$W_i = log \frac{1-error(Mi)}{error(Mi)}$*
  4. *C = Mi(X); // get class prediction for X from Mi*
  5. *Add W_i to weight for class c*
  6. *End for*
  7. *Return the class with the largest weight;*
- *Output: A composite Model*

5.1.3 Stacking

Stacking (sometimes called *stacked* generalization) involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although, in practice, a logistic regression model is often used as the combiner. The stacking model is represented as shown in Fig. 4.

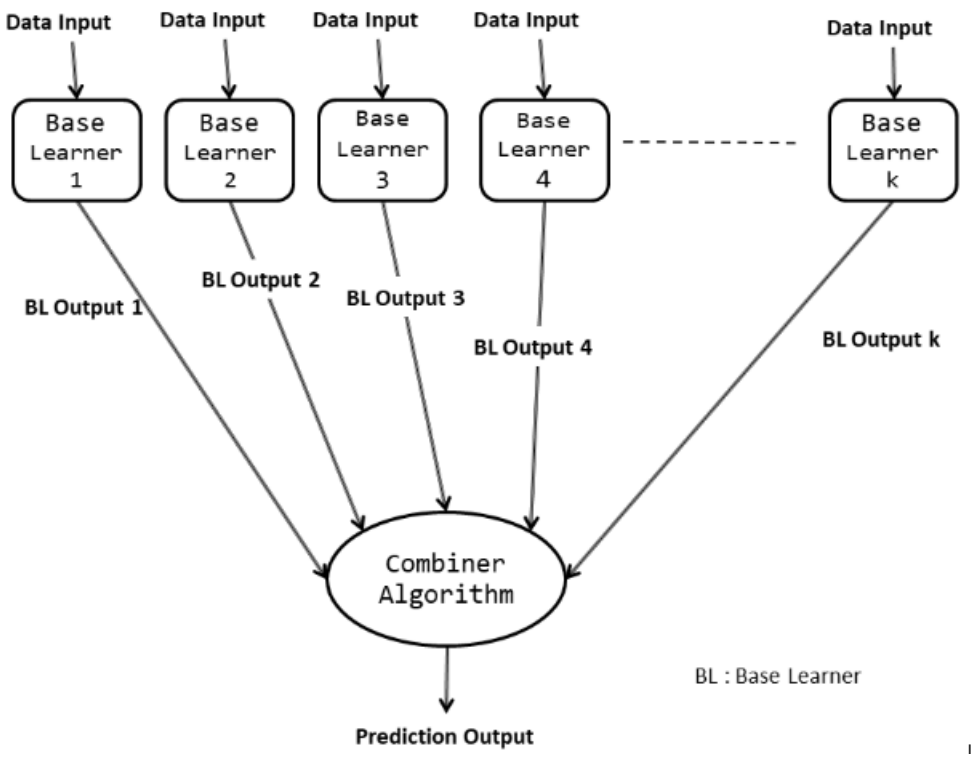

Fig.4 . *Stacking Model*

Stacking typically yields performance better than any single one of the trained models. It has been successfully used on both supervised learning tasks (regression, classification and distance learning) and unsupervised learning (density estimation). It has also been used to estimate bagging's error rate. It has been reported to out-perform Bayesian model-averaging. The two top-performers in the Netflix competition utilized blending, which may be a form of stacking.

The algorithm of stacking is listed below:

Algorithm 4 Stacking

- *Input: Dataset*
- *Classification:*
  1. *BaseLearners = [Knn, Svm, Ann, Logistic……]*
  2. *Combiner = [Random Forest]*
  3. *X_train, X_test, Y_train, Y_test = test_train_split(Dataset)*
  4. *Model = [ ]*
  5. *For function in BaseLearners:*
  6. *Model.append(function.train(X_train, Y_train)*

7. *Train_set = [ ]*
8. *For i in range(length(X_train)):*
9. *Train = [ ]*
10. *For m in Model:*
11. *Train.append(m.predict(X_train[i]))*
12. *Combiner.train(Train_set, Y_train)*

- *Prediction:*
  1. *Train_set_x = [ ]*
  2. *For m in model:*
  3. *Train_set_x.append(m.predict(x))*
  4. *Output = Combiner.predict(Train_set_x)*
- *Output: Individual Accuracy array of each method, Stacking accuracy*

5.1.4 Bayesian Parameter Averaging

Bayesian parameter averaging (BPA) is an ensemble technique that seeks to approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space and combining them using Bayes' law. Unlike the Bayes optimal classifier, Bayesian model averaging (BMA) can be practically implemented. Hypotheses are typically sampled using a Monte Carlo sampling technique such as MCMC. For example, Gibbs sampling may be used to draw hypotheses that are representative of the distribution. It has been shown that under certain circumstances, when hypotheses are drawn in this manner and averaged according to Bayes' law, this technique has an expected error that is bounded to be at most twice the expected error of the Bayes optimal classifier. Despite the theoretical correctness of this technique, early work showed experimental results suggesting that the method promoted over-fitting and performed worse compared to simpler ensemble techniques such as bagging; however, these conclusions appear to be based on a misunderstanding of the purpose of Bayesian model averaging vs. model combination. Additionally, there have been considerable advances in theory and practice of BMA. Recent rigorous proofs demonstrate the accuracy of BMA in variable selection and estimation in high-dimensional settings, and provide empirical evidence highlighting the role of sparsity-enforcing priors within the BMA in alleviating overfitting.

5.1.5 Bayesian Model Combination

Bayesian model combination (BMC) is an algorithmic correction to Bayesian model averaging (BMA). Instead of sampling each model in the ensemble individually, it samples from the space of possible ensembles (with model weightings drawn randomly from a Dirichlet distribution having uniform parameters). This modification overcomes the tendency of BMA to converge toward giving all the weight to a single model. Although BMC is somewhat more computationally expensive than BMA, it tends to yield dramatically better results. The results from BMC have been shown to be better on average (with statistical significance) than BMA, and bagging.

The use of Bayes' law to compute model weights necessitates computing the probability of the data given each model. Typically, none of the models in the ensemble are exactly the distribution from which the training data were generated, so all of them correctly receive a value close to zero for this term. This would work well if the ensemble were big enough to sample the entire model-space, but such is rarely possible. Consequently, each pattern in the training data will cause the ensemble weight to shift toward the model in the ensemble that is closest to the distribution of the training data. It essentially reduces to an unnecessarily complex method for doing model selection.

The possible weightings for an ensemble can be visualized as lying on a simplex. At each vertex of the simplex, all the weight is given to a single model in the ensemble. BMA converges toward the vertex that is closest to the distribution of the training data. By contrast, BMC converges toward the point where this distribution projects onto the simplex. In other words, instead of selecting the one model that is closest to the generating distribution, it seeks the combination of models that is closest to the generating distribution.

The results from BMA can often be approximated by using cross-validation to select the best model from a bucket of models. Likewise, the results from BMC may be approximated by using cross-validation to select the best ensemble combination from a random sampling of possible weightings.

5.1.6 Bayes Optimal Classifier

The Bayes Optimal Classifier is a classification technique. It is an ensemble of all the hypotheses in the hypothesis space. On average, no other ensemble can outperform it. Naive Bayes Optimal Classifier is a version of this that assumes that the data is conditionally independent on the class and makes the computation more feasible. Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true. To

facilitate training data of finite size, the vote of each hypothesis is also multiplied by the prior probability of that hypothesis. The Bayes Optimal Classifier can be expressed with the following equation:

$$y = \arg\max_{c_j \in C} \sum_{h_i \in H} P(c_j \mid h_i) P(T \mid h_i) P(h_i) \qquad [eq^n 5]$$

Where y: predicted class, C: of all possible classes, H: hypothesis space, P: probability, and T: training data.

As an ensemble, the Bayes optimal classifier represents a hypothesis that is not necessarily in H. The hypothesis represented by the Bayes Optimal Classifier, however, is the optimal hypothesis represented by the Bayes Optimal classifier, however, is the optimal hypothesis in ensemble space. This formula can be restated using Bayes' theorem, which says that the posterior is proportional to the likelihood times the prior:

$$P(h_i \mid T) \alpha P(T \mid h_i) P(h_i) \qquad [eq^n 6]$$

Hence,

$$y = \arg\max_{c_j \in C} \sum_{h_i \in H} P(c_j \mid h_i) P(h_i \mid T) \qquad [eq^n 7]$$

5.2 Machine Learning Technique

There are different regression techniques for prediction like Linear regression, Logistic regression, Support Vector regression, Ridge Regression, Lasso regression, ElasticNet regression etc and for classification different ML classification algorithms are used such as Decision Tree, Random Forest, SVM, ANN, KNN, Naïve Bayes etc.

5.2.1 Linear Regression

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable, the process is called simple linear regression. For more than one explanatory variable is called multiple linear regression. In Linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data.

5.2.2 Logistic Regression

Logistic regression is estimating the parameters of a logistic model (Statistical model that, in its basic form, uses a logistic function to model a binary dependent variable; many more complex extensions exist); it is a form of binomial regression.

5.2.3 Support Vector Machine

In machine learning, Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

5.2.4 Ridge Regression

Ridge regression is a technique for analysing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large, so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. It is hoped that the net effect will be to give estimates that are more reliable.

5.2.5 Lasso Regression

Lasso is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces.

5.2.6. ElasticNet Regression

In the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L_1 and L_2 penalties of the lasso and ridge methods.

5.2.7 ANN

ANN or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. The Neural network itself isn't an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems learn to performs tasks by

considering examples, generally without being programmed with any task-specific rules. ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and the signal additional artificial neurons connected to it.

5.2.8 KNN

The k-nearest neighbors algorithm is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space.

5.2.9 Naïve Bayes Classifier

Naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumption between the features. Naïve Bayes classifiers are highly scalable, requiring several parameters linear in the number of variables in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

5.2.10 Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks. That operate by constructing multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

5.2.11 Decision Tree

Decision Tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Chapter 6

RESULT ANALYSIS

In this project of Software reliability prediction using ensembling learning approach 3 different ensembling machine learning approaches are used to predict and classify the dataset. In which bagging, boosting & stacking is implemented using python programming language given software reliability prediction & classification dataset. In software reliability prediction dataset given a single row of failure time of the software indexed in a file, where sample with replacement technique is used to build another dataset and train the model on that dataset and after that the prediction is done using the original dataset and similarly different rounds is used for the iteration purpose to improve the accuracy of the ensembling model. While predicting the output, different machine learning regression models are used to predict the output and then find the error rate in the model.

6.1 Bagging

The prediction models or the regression methods those are used in this project are Linear Regression, Decision Tree regression, Ridge regression, Lasso regression, ElasticNet regression, Random Forest regression, Support vector regression and others. After predicting the dataset and finding the error; the error value is very less in each model. We can also say that the error is tends to zero.

For each of the regression method the error rate is very less and for some regression like the logistic regression and support vector regression the error rate is bit high. The comparison between actual failure interval values versus various regression model and the bagging model is shown in Fig. 5, 6 & 7.

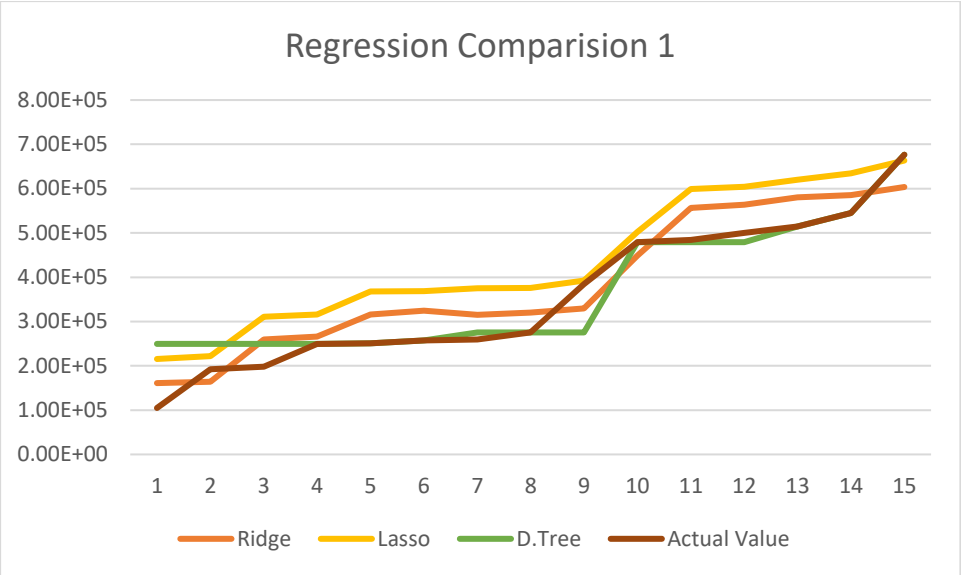

Fig.5. *Regression model comparison between Ridge regression, lasso regression, Decision Tree regressor and the Actual time interval.*

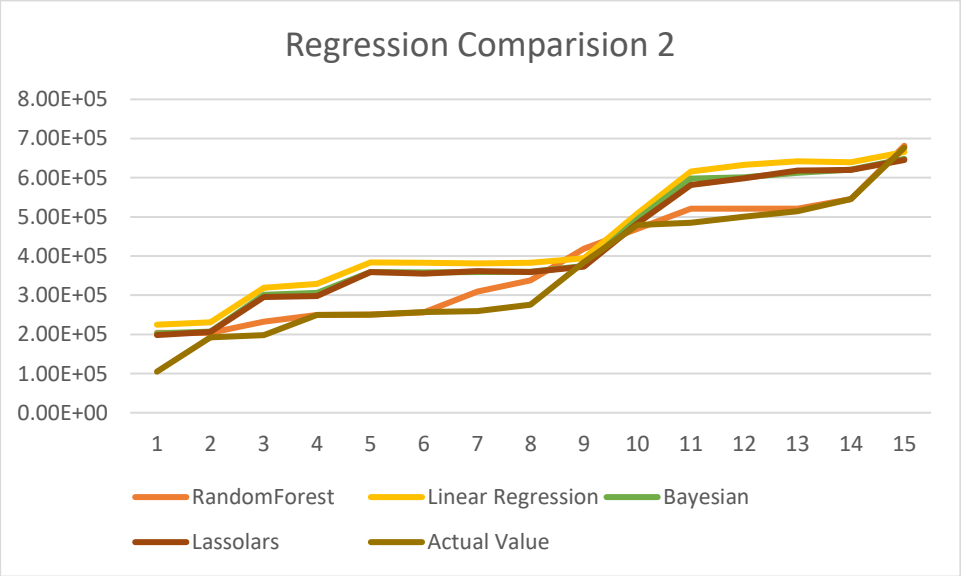

Fig.6. *Regression model comparison between Random Forest regression, linear regression, Bayesian Ridge regressor, LassorLars Regressor and the Actual time interval.*

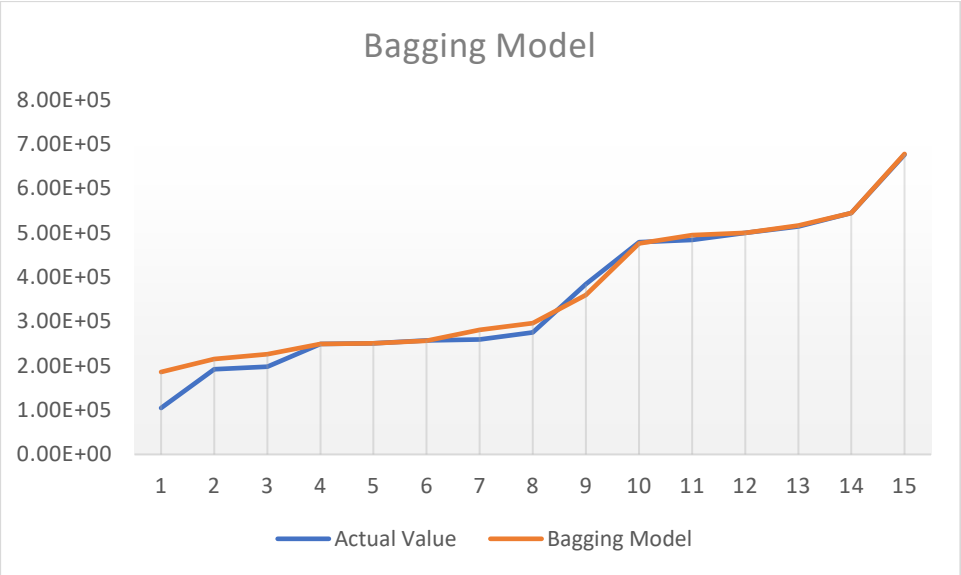

Fig.7. *Actual failure time interval and Bagging model failure time interval comparison.*

Similarly, in the bagging approach for the classification 7 different models or machine learning classifiers are used to classify the sample with replaced dataset and then the original dataset is used to get the output class and then finding the final output voting approach is used to find the final output value. While displaying the output in the bagging approach accuracy of the model is printed also the confusion matrix and maximum value, minimum value, mean value and final bagging accuracy is printed which gives a clear idea about the classification of each model. The visualization of the accuracy of each model while where the iterator range is 7 is visualized below in Fig.8.

Similarly, while we print the confusion matrix of each model there are four possible values where 2 out of 4 options are correct and others are incorrect. The accuracy value of each model depends upon the confusion matrix of each model and the accuracy can be calculated by using the confusion matrix. If there is a plot between each value of the confusion matrix in each model, then the line graph is shown as in Fig.9.

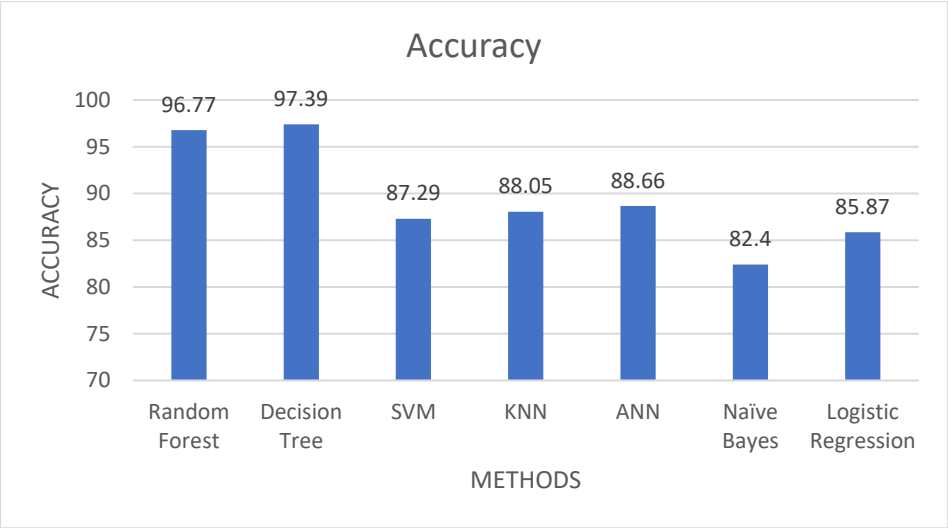

Fig.8. *Accuracy of the different machine learning classification model*

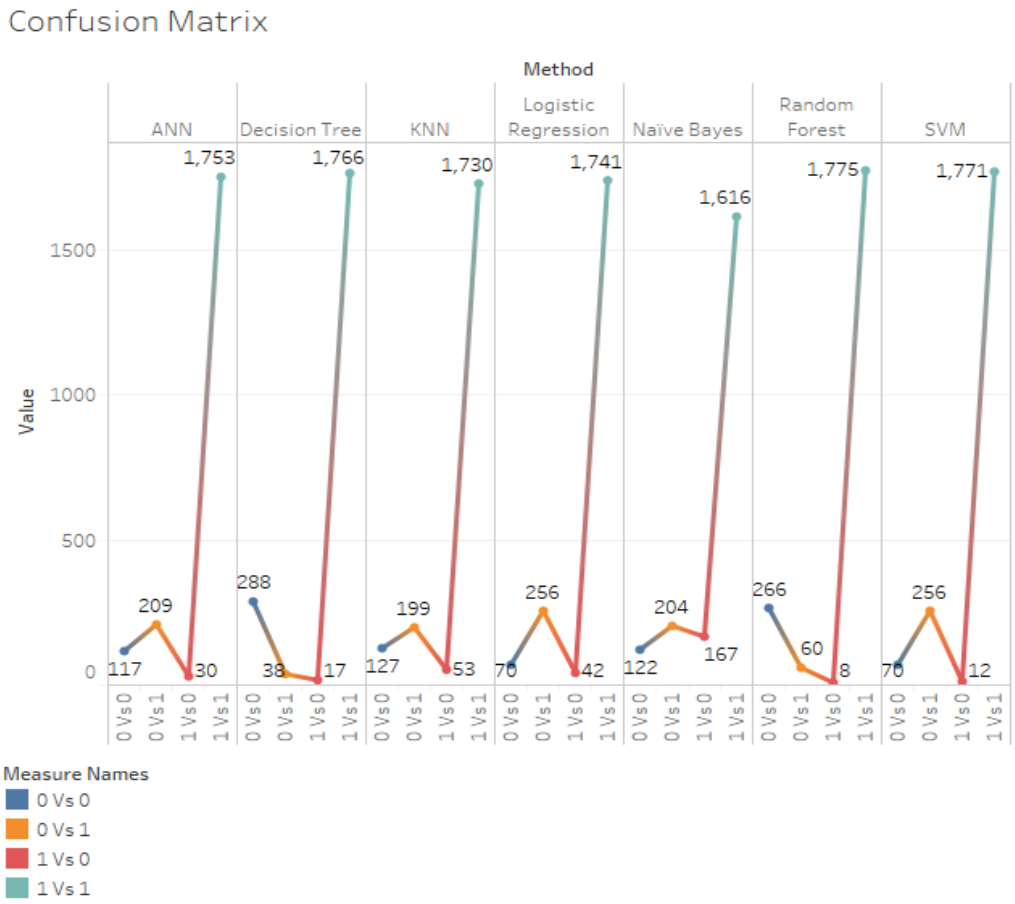

Fig.9. *Confusion matrix visualization of different machine learning models*

6.2 Boosting:

Adaboost algorithm is used to implement boosting over software reliability prediction to improve the accuracy of the dataset where for each model we are creating multiple instance of the boosting where each instance will act as a neuron; the weight will be initialized in very small amount and then output will be predicted and again the weight will be adjusted and similar thing continues. At last the model gives us the individual accuracy of each model and the boosted accuracy of each model for each iteration. And we can also observe the following graph to observe the individual accuracy of each model for each iteration and boosting accuracy of each model for each iteration in the Fig. 10, 11, 12 & 13.

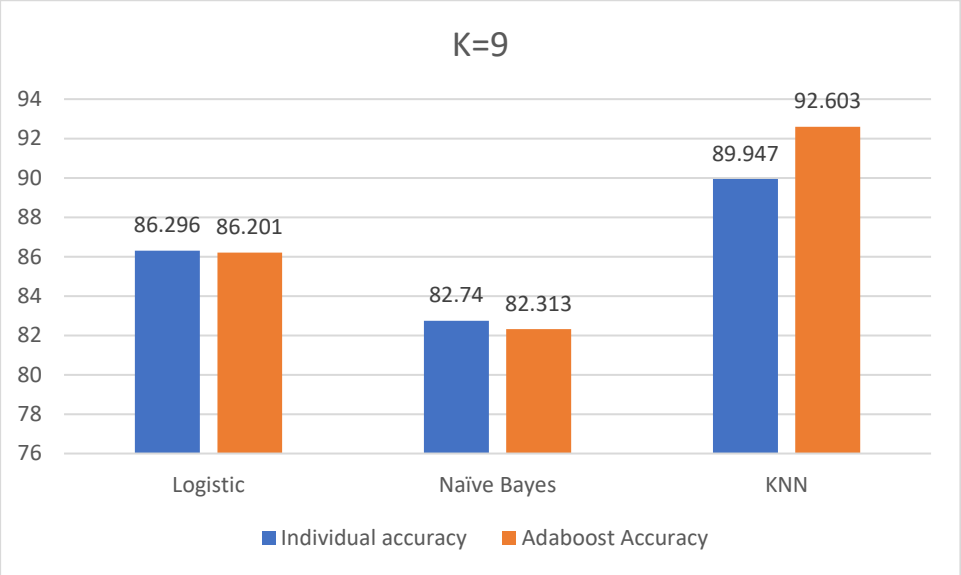

Fig.10. *Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=9*

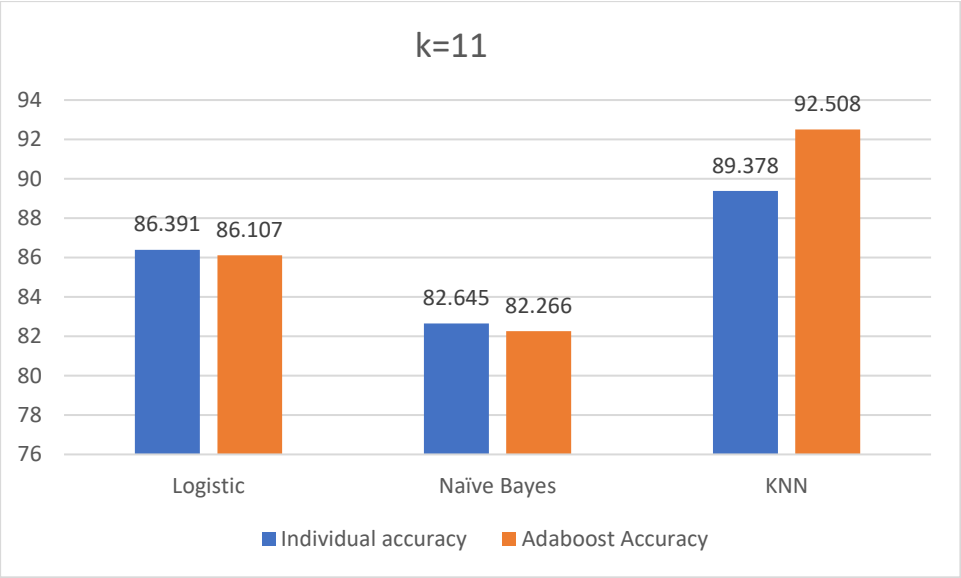

Fig.11. *Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=11*

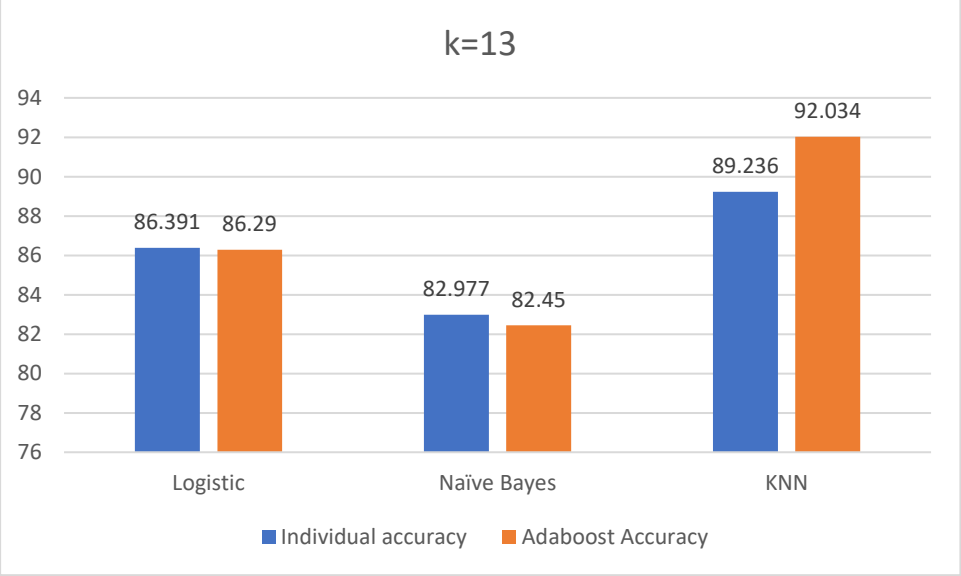

Fig.12. *Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=13*

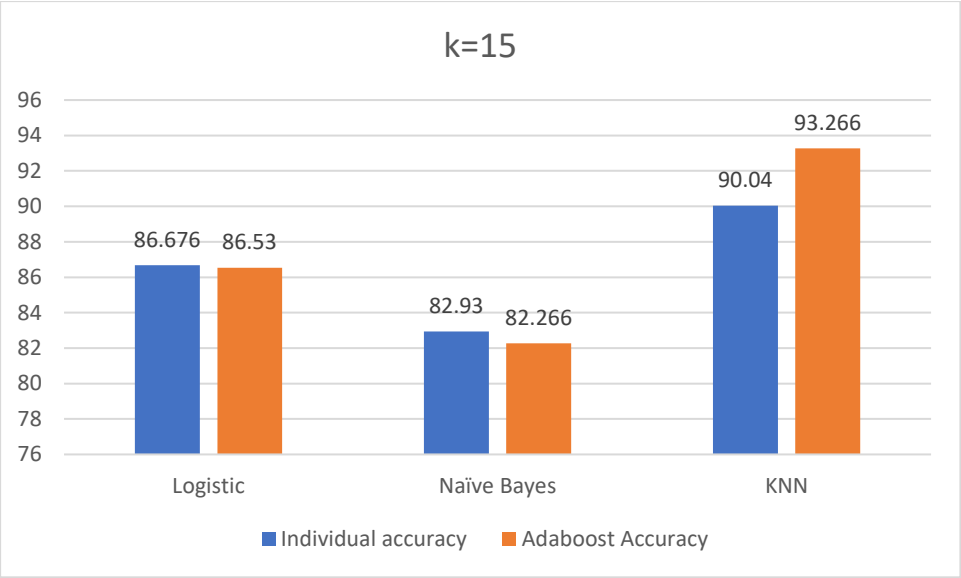

Fig.13. *Comparison between boosting accuracy and individual accuracy of 3 models for iterative range k=15*

As we can see from the above graph that for different value of K the result is different and for some k the accuracy is higher in increasing k.

6.3 Stacking:

In stacking, to train machine learning algorithms with training dataset and then the new dataset is generated with these models. This new dataset is used as the input for the combiner machine learning algorithm.

In stacking the sub-models produce different predictions. All these sub model's prediction is combined to generate a new dataset and then that is used for the combiner model. For different iteration of stacking different models are used in stacking and the accuracy value of the stacking is represented in the Fig. 14.

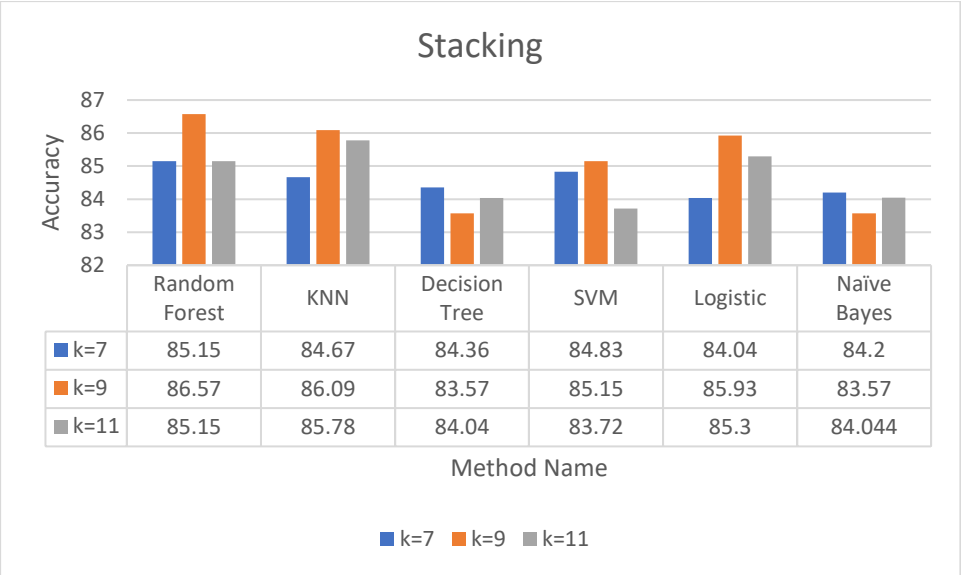

Fig.14. *Comparison between stacking accuracy of different machine learning model for different iterative range*

Thus, from the above graph we conclude that for different value of k in different model different model give different accuracy. And from the above graph we can clearly see the variance of the accuracy for different models for different k.

Chapter 7

Conclusion

Different machine learning prediction and classification algorithms are used in ensemble learning for software reliability prediction. As we have seen that in the bagging approach of the classification the output is classified in different class and then voting method is used to get the final output. In prediction problem we are getting very less error rate for prediction. In boosting model, we have taken weak machine learning model and then by using AdaBoost algorithm the accuracy of the model is improved. In stacking we are combining different base learner algorithm and then predicting the dataset and finally we are using combiner algorithm to predict the output. All the output value is stored in xlsx file and tableau software is used to visualize the data or the output that we got from the Ensemble method for software reliability prediction.

In bagging method of classification, we can see that for k=7 the decision tree gives a highest performance among all other machine learning model. In boosting method, for different value of k the output of KNN model is improved and the accuracy value is increased by maximum 1%. For stacking for different value of k i.e. 7,9 or 11 KNN model gives highest accuracy among all.

Acronyms

- **MTBF:** Mean Time Between Failures
- **ANN:** Artificial Neural Network
- **KNN:** K-Nearest Neighbours
- **ML:** Machine Learning
- **BMC:** Bayesian Model Combination
- **BPA:** Bayesian Parameter Averaging
- **SVM:** Support Vector Machine
- **SVR:** Support Vector Regression
- **CSIAC:** Cyber security and Information Systems Information Analysis Centre
- **IEEE:** Institute of Electrical & Electronics Engineers
- **ANSI:** American National Standard Institute

Bibliography

[1] Aljahdali SH, Buragga KA (2008) Employing four ANNs paradigms for software reliability prediction: an analytical study. ICGST AIML J 8(2):1687–4846

[2] Karunanithi N, Whitley D, Malaiya Y (1992) Prediction of software reliability using connectionist models. IEEE Trans Softw Eng 18(7):563–574

[3] Quyoum A, Dar MD, Quadr SMK (2010) Improving software reliability using software engineering approach-a review. Int J Comput Appl 10(5):0975-8887

[4] Aggarwal KK, Singh Y, Kaur A, Malhotra R (2006) Investigating the effect of coupling metrics on fault proneness in object-oriented systems. Softw Qual Prof 8(4):4-16

[5] Goel B, Singh Y (2009) An empirical analysis of metrics. Softw Qual Prof 11(3):35-45

[6] Malhotra R, Kaur A, Singh Y (2011) Empirical validation of object oriented metrics for predicting fault proneness at different severity levels using support vector machines. Int J Syst Assur Eng Manag 1(3):269–281. doi:10.1007/s13198-011-0048-7

[7] Xingguo L, Yanhua S (2007) An early prediction method of softwarereliability based on support vector machine. In: Proceedingsinternational conference on wireless communications, networking and mobile computing (WiCom'07), pp 6075–6078

[8] Hua Jung L (2010) Predicting software reliability with support vector machines. In: Proceedings of 2nd international conference on computer research and development (ICCRD'10), Kuala Lumpur,pp 765–769

[9] Machine Learning Mastery, https://machinelearningmastery.com/implmenting-stacking-scratch-python/

[10] Kdnuggets, https://www.kdnuggets.com/2016/02/ensemble-methods-techniques-produce-improved-machine-learning.html

[11] Oregon State University, http://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf

[12] Towards Data Science, https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

[13] Ensemble Learning-Wikipedia, https://en.wikipedia.org/wiki/Ensemble_learning

[14] Toptal, https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

[15] Stats and Bots, https://blog.statsbot.co/ensemble-learning-d1dcd548e936

[16] Analytics Vidhya, https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

[17] Introduction into ensemble method-Ensembling-Coursera, https://www.coursera.org/lecture/competitive-data-science/introduction-into-ensemble-methods-MJKCi

[18] Ensemble Method-Scikit-learn, http://scikit-learn.org/stable/modules/ensemble.html

[19] Ensemble Learning in Python-DataCamp, https://www.datacamp.com/community/tutorials/ensemble-learning-python

[20] Frontline Solvers, https://www.solver.com/data-mining-ensemble-methods

[21] Machine learning & Data Mining Ensemble methods, http://www.cs.toronto.edu/~rsalakhu/CSC411/notes/lecture_ensemble1.pdf