"SOFTWARE DEFECT PREDICTION USING HOMOGENEOUS AND HETEROGENEOUS ENSEMBLE TECHNIQUES"

DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY IN

SOFTWARE ENGINEERING

***Submitted* By**

Yankit Kumar (2K19/SWE/16)

Under the guidance

Of

Dr. Manoj Kumar
Assistant Professor, DTU

Department of Computer Science & Engineering
Delhi Technological University, Delhi

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
SHAHBAD DAULATPUR, DELHI-110042

MAY-2021

DECLARATION

I hereby declare that the work presented in this report entitled "SOFTWARE DEFECT PREDICTION USING HOMOGENEOUS AND HETEROGENEOUS ENSEMBLE TECHNIQUES", in fulfillment of the requirement for the award of the MASTER OF TECHNOLOGY degree in Software Engineering submitted in Computer Science Department at DELHI TECHNOLOGICAL UNIVERSITY, New Delhi, is an authentic record of my own work carried out during my degree under the guidance of Dr. Manoj Kumar.

The work reported in this has not been submitted by me for the award of any other degree or diploma.

Date:

Yankit Kumar
(2K19/SWE/16)

Place: Delhi

CERTIFICATE

This is to certify that Yankit Kumar (2K19/SWE/16) has completed the project titled "Comparing the predictive performances of different classification algorithm and ensemble techniques" under my supervision in partial fulfillment of the MASTER OF TECHNOLOGY degree in Software Engineering at DELHI TECHNOLOGICAL UNIVERSITY.

[illegible]

Dr. Manoj Kumar
(Supervisor)
(Assistant Professor),
Dept. of Computer Science & Engineering,
Delhi Technological University

ACKNOWLEDGEMENT

I am very thankful to **Dr. Manoj Kumar** (Assistant Professor, Department of Computer Science) and all the faculty members of the Department of Computer Science at DTU. They all provided us with immense support and guidance for the project.

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

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

Yankit Kumar
Roll No. 2K19/SWE/16

ABSTRACT

Many researchers have already been working in the field of defect prediction in software using some machine learning algorithms. Their results vary from dataset to dataset. These algorithms give inconsistent output for predicting defects in a random software project. Researchers have not decided which machine learning algorithm is best suitable for correctly predicting the defects in software so recent developments in machine learning introduce ensembling methods to predict defects. Ensembling takes the advantages of different techniques to give a better prediction of defects compared to individual base models.

The major objective for work is building the ensemble of various classification methods to predict the defects in the given software module and compare the results of the ensemble with an individual classification technique. We have used naive bayes classifiers, logistic regression, k- nearest neighbors, support vector machine, decision trees for implementation and then choose the best three classification techniques to build the ensemble, and data sets are collected from publicly available repositories. Here we have used heterogeneous ensemble techniques such as voting, stacking and homogeneous ensemble techniques such as bagging and boosting for prediction. Also heterogeneous version of bagging and boosting is used. All the six techniques are implemented and compared using the various performance metrics. Area under ROC curve (AUC) is used to analyze the prediction performance and to check the statistical significance of the results of different models, Friedman test is used. The results show that the ensemble method improves the prediction performance as compared to individual classifiers.

INDEX

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SDP	Software Defect Prediction
AUC	Area Under the curve
ROC	Receiver Operating Characteristics
WMC	Weighted Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Object
RFC	Response for a Class
LCOM	Lack of Cohesion in Methods
CA	Afferent Coupling
CE	Efferent Coupling
NPM	Number of Public Methods
LCOM3	Lack of Cohesion in Methods version 3
LOC	Lines of Code
DAM	Data Access Metric
MOA	Measure of Aggregation
MFA	Measure of Functional Aggregation
CAM	Cohesion Among Methods
IC	Inheritance Coupling
CBM	Coupling Between Methods
AMC	Average Method Complexity
max (CC)	Maximum McCabe’s Complexity
avg (CC)	Average McCabe’s Complexity
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

Software testing is a resource and time-consuming task in the software development lifecycle. The main motive of the testing process is to deliver defect-free products to the customers and meet all the requirements of the stakeholders. The small budget for testing process leads to the development of a prediction model which helps in finding defective parts of the software so that there is no need to test all modules of the software.

Software defect prediction (SDP) is the process of finding defective modules in the software using some software metrics to enhance the quality of the software and also this process reduces the number of modules to be tested. If the software is complex then it is difficult to produce it without any faults. So there is a need to predict the faults at an early stage so that there will be no increment in the budget of the software and the customers get quality products. This leads to building the defect prediction model.

Recent developments in machine learning introduce an ensemble approach which combines different techniques to predict the number of defects and gives better prediction results compared to individual techniques. According to Mendis Moreira et al[7], "*Ensemble learning is a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models (ensemble) is integrated in some way to obtain the final prediction.*" The ensemble method improves the prediction performance of the base models. Due to the benefits of the ensemble approach, it is widely used in classifying the modules of the software into faulty and non-faulty modules.

Regression and classification methods are commonly used in the field of software defect prediction. Regression methods are used to predict the total number of defects in the software [8]. Various classification algorithms have been used in the field of defect prediction [1] which includes statistical and machine-learning techniques. Discriminant analysis [2], logistic regression [3], factor analysis [4], fuzzy classification [5], classification trees [5], Bayesian network [6] etc. comes under classification algorithms.

If we use ensemble methods, it has some advantages over base learning techniques and leads to improvement in the prediction performance [7]. If the dataset is very large, then the cost of computing the global minima is large in case of individual fault prediction methods. So to overcome this problem, the ensemble method combines the local minima of base techniques to improve the overall performance of the given input function [9].

In this study, we present a system that makes use of ensemble methods to predict the faults in the software modules. We analyze the results of five classification techniques and choose those three techniques which give better AUC values on most of the dataset. These three techniques work as base learning techniques to build ensemble methods.

1.1 Software Defect Prediction

Defect prediction encompasses the prediction in which it referred to defets in software modules. In this, module is known as the main unit of a system such as function or class.

Any software is composed of many of OOPS and function oreiented which is associated to many large number of classes.

To ensure this, there are many project related software attributes for each class as well.

It is termed as centre for application of quality assurance (QA) techniques to assess the classes' inherent quality. For these techniques, it add-ons inspections, unit, static source code analyzers, unit testing etc. A documentation of whole results in this QA is known as defect log. We can also, use this to grow our domain, understand the principles, major human errors.

If the information contained in the data provides not only a precise account of the encountered faults (i.e., the "bugs"), but also a thorough description of static code features such as lines of code (LOC), complexity measures (e.g., McCabe's cyclomatic complexity), and other suitable object-oriented design metrics. There are three very good reasons to study defect predictors learned from static code attributes: they are easy to use, widely used, and useful to use.

1.2 Ensemble Learning

Ensemble techniques comes out as to be an meta-classifier which are combination of many of machine learning models by applying that to it for one prescient model which overall improves predictions , decline of predisposition or may even cancels the difference of stacking.

Ensemble learning is a technique that consists of classifiers in which an indicator is built for utilization of combination of classifiers which may or may not be of one kind while,

By measuring the weighted vote or at the midpoint of aftermath, in their yiels. New info. Focuses on the ordered ones. Various students are utilized to ensemble model to make it as a base model.

Ensemble model capacity is dependent upon the sum of better base class capacity which makes that engaging in the capacity of ensemble models in support of frail student's exhibition and even more to make it as solid theory which produces unquestionably progressively exact predictions. Along with this the base students in ensemble learning are traditionally assigned as feeble.

1.3 Organisation Of Dissertation

This dissertation is organized as follows: Chapter 1 includes introduction and related work done. In Chapter 2 – Proposed work, basics ensemble techniques, ensemble techniques used in project, classification techniques used and their comparison is being explained. Chapter 3 - Experimental layout explains about the proposed architecture, dataset used, dataset pre processing, performance metrics used and statistical tests used. Chapter 4 – Results includes the discussion about the results obtained using box plots, AUC values, statistical tests and graphs. The dissertation ends with conclusion and references being written.

CHAPTER 2

LITERATURE REVIEW

The writing examines that have been overviewed proposes widely compelling models for the prediction of deformities. The underlying work in prediction of software deserts concentrate chiefly about the utilization of measurable procedures. The outline of the examinations that were utilized in this exploration are talked about beneath.

2.1Related Work

Aleem et al.[11] analyzed and compare the prediction performance of 11 machine learning techniques such as Naive bayes, Multilayer perceptron, support vector machines, Adaboost, Bagging, Random forest etc. using 15 NASA dataset which was downloaded from PROMISE repository. The result of this study shows that the Bagging and Support vector machine gives better prediction performance.

A comparative study of ensemble classifiers done by Wang et al.[12] shows that voting and random forest gives better classification results as compared to stacking, naive bayes, Adaboost and bagging.

A study done by Perreault et al.[13] on five NASA dataset compared naive bayes, support vector machines, artificial neural networks, logistic regression and k nearest neighbor but there is no clear explanation about which technique is best.

Hussain et al.[14] in his study compared the three ensemble techniques that uses five base classification techniques such as naive bayes, logistic regression, J48, Voted-Perceptron, and support vector machine in Weka tool for SDP. The result of this study shows that Stacking gives better results among all the ensemble techniques used. A defect prediction model using ensemble method was presented by Mısırlı, Bener, and Turhan (2011)[15]. In this, they build ensembles using three learning techniques: naive bayes, voting feature intervals and artificial neural networks. This study results in improvement of the prediction accuracy using ensemble methods.

Aljamaan and Elish (2009)[16] analyzed the bagging and boosting ensemble methods and compared their performances with some other individual classification techniques. The results show that bagging and boosting improved prediction accuracy over most of the individual classification techniques.

A study done by Elish, Aljamaan, and Ahmad (2015)[17] shows that ensemble method gives correct prediction results on considered dataset. They used two publicly available datasets to analyze the ensemble methods for predicting software maintenance and changing efforts.

All the studies done in this field shows the prediction capabilities of ensemble method. Most of them show that ensemble methods improve the prediction accuracy results as compared to individual technique. Other works available in the field of prediction using ensemble methods are given in Zheng(2010)[18], Wang et al.(2011)[20] and Twala(2011)[19].

CHAPTER 3

PROPOSED WORK

3.1 Ensemble Method

An ensemble is a machine learning model where combination from two or more models' prediction is made.

The predictions made by the ensemble members' may be combined using statistics, such as the mode or mean, or by more sophisticated methods that learn how much to trust each member and under what conditions.

Ensemble learning techniques are known for their long histories for showing a better level of performance that too in a variety of various machine learning applications. The domains of all of these applications consist of regression and classification problems. The popular model of Random forest along with the gradient boosting model is a very well-known ensemble model, in which a combination of some weak learners is used for building up an ensemble. In these type of models, the collection made up of weak learners is homogeneous, this implies that all those weak learners that are of same type are to be grouped together in order to show their combined strength. Here, we will show how a heterogeneous collection made from weak learners can be used to build a hybrid learning model in ensemble. All these different types of ML algorithms can be grouped together for doing this task so as to work on any particular classification problem.

Ensemble methods greatly increase computational cost and complexity. This increase comes from the expertise and time required to train and maintain multiple models rather than a single model.

3.1.1 Motivation Behind Ensemble Method

The inspiration driving utilizing the ensemble AI methods are because of a few reasons.

Ensemble models have been demonstrated exceptionally successful to inspire the precision and the presentation of the models.

Some AI methods play out a neighborhood search as opposed to finding the worldwide optima, which frequently gets caught in nearby optima. For instance, the calculation for the choice tree utilizes a parting rule for ravenous strategies to develop the tree. On the other hand, an ensemble worked from a few distinctive beginning stages by running a nearby pursuit typically will in general give a superior prediction of the genuine unlabeled example than any of the individual classifiers taken independently.

A learning calculation can be seen as looking for a space H of theory so as to recognize the best speculation in space. Nonetheless, the factual issue emerges that the measure of information accessible to prepare the model is excessively little contrasted with the size of the speculation space. Without adequate information, a wide range of theories can be found in a learning calculation in H which when utilized with preparing information for the most part gives a similar precision. By making an ensemble of all these exact models, the calculation can have 10 weighted-normal of their votes and gives a decrease in the probability of choosing the improper classifier for prediction.

3.1.2 Why an ensemble method is used?

An ensemble method is used for two main reasons:

1. Performance: Better predictions and better thus performance can be achieved through ensemble method in contrast of non-ensemble method.
2. Robustness: The spread or dispersion of the predictions can be reduced by ensemble methods and so the performance of the model can be improved.

Through Ensemble method better predictive performance can be achieved for a problem of predictive modelling as compared to a single predictive model.

To achieve better performance, the ensemble method adds the bias and thus reduces the variance of the prediction error.

Another advantage of ensemble method that is not discussed frequently is they provide better robustness and reliability even if the performance of model is average.

3.2 Ensemble Techniques Used in Project

Various researchers have evaluated the performance of classification techniques for the software fault prediction. Therefore, there is none common permission among the researchers that a classification approach is the most suited for the fault prediction. Recent developments in the area of machine learning have brought the concept of ensemble learning to improve the performance of the prediction model.

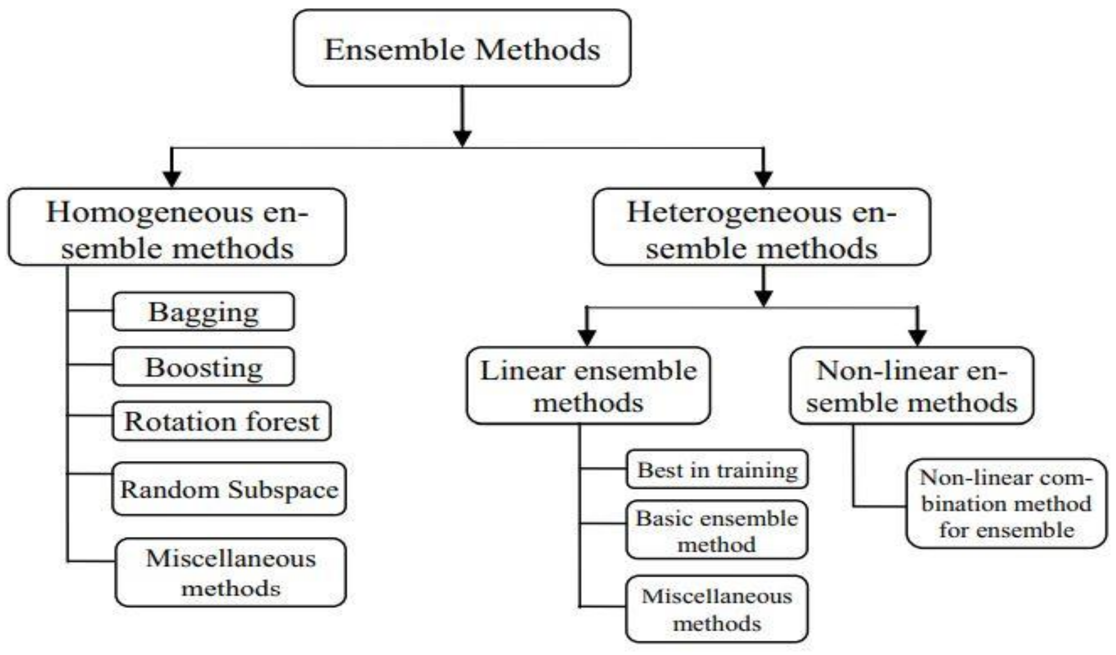

Fig 1 Classification of ensemble techniques

The central idea of ensemble is to combine the prediction outputs of several learning techniques such that the overall performance of the decision is improved as compared to the individual techniques output. Ensemble method can be of two types: homogeneous ensemble and heterogeneous ensemble. In a homogenous ensemble, learning techniques are of the same type such as bagging, boosting, etc. In heterogeneous ensembles, different learning techniques are used.

Here in this study we are using four ensemble techniques to combine the prediction result of some classification technique in order to get better prediction accuracy. The four ensemble techniques used are:

1. Stacking:

Stacking is a way of ensembling classification or regression models that consists of two-layer estimators. The first layer consists of all the base models or models for individual techniques that are used to predict the outputs on the test datasets. The second layer consists of a Meta-Classifier or Regressor which takes all the predictions of base models as an input and generates new predictions. Fig 2 shows the architecture of stacking methods.

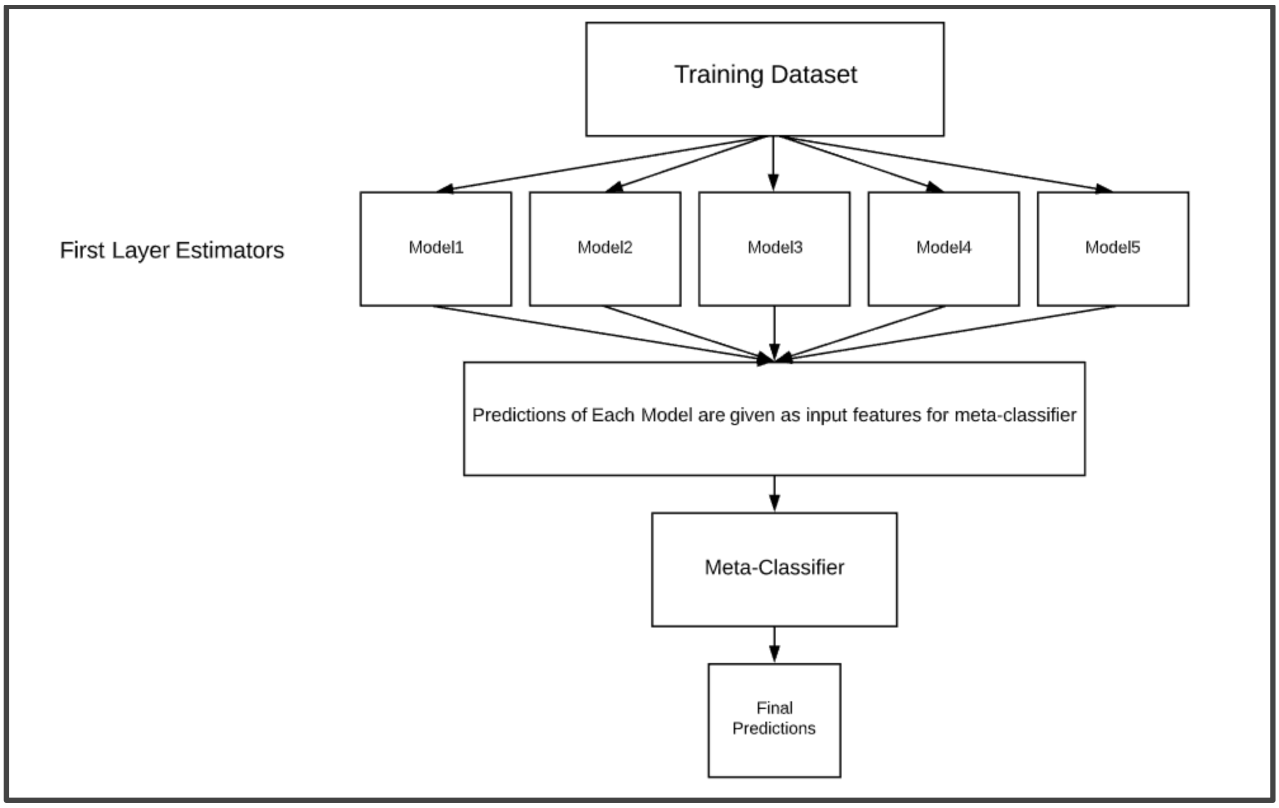

Fig 2 Architecture of stacking method

2. Voting:

Voting is another way of ensemble technique used for classification models. In this the results are combined on the basis of majority voting and on the basis of probability values. There are two types of voting:

a) Hard voting:

In hard voting, the results of individual technique are combined on the basis of majority voting for better prediction performance.

b) Soft voting:

Soft voting is applicable where individual techniques result in probabilities for the outcome. This technique gives the best result by calculating the average probabilities of the individual technique probabilities. Here we are using soft voting. Fig 3 shows the architecture of the voting method.

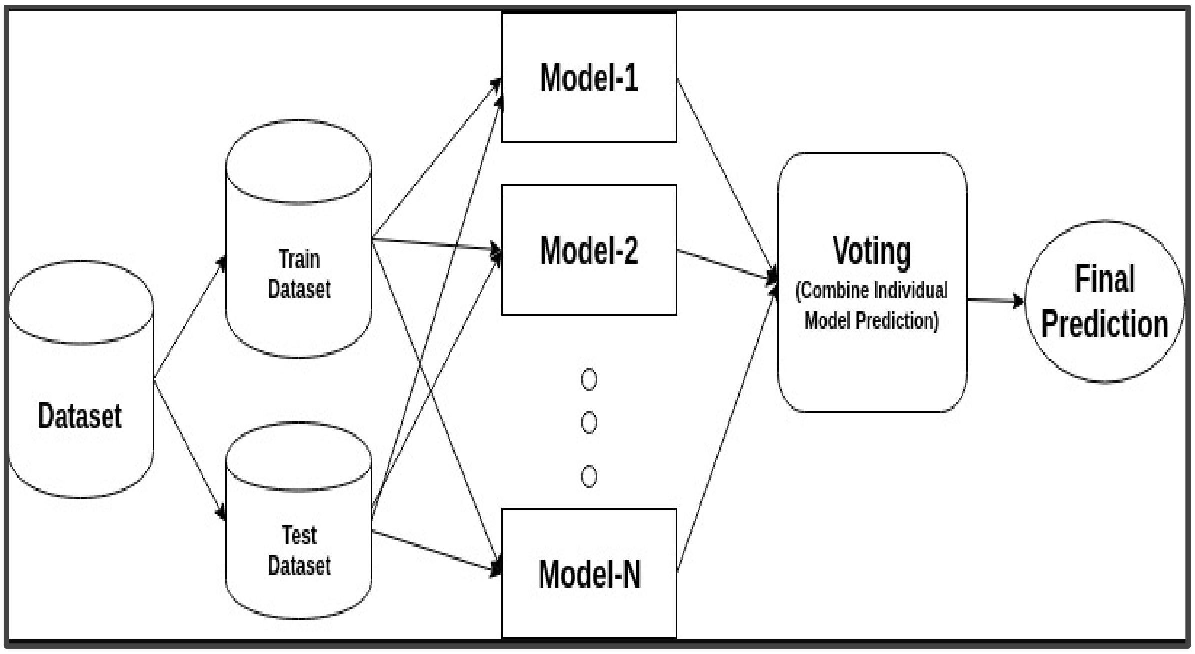

Fig 3 Architecture of voting method

3. Bagging

Bagging is a condensing for Bootstrap Aggregating. Packing was presented by Breiman. The thought of sacking is straightforward, that is, the ensemble comprises of classifiers that depend on the preparation set's bootstrap copies. The yields of the individual classifier are consolidated utilizing the blend rule of larger part casting a ballot.

In this bootstrap testing is utilized in which subset of information focuses are chosen aimlessly from the space of information focuses with names. The fundamental hidden guideline is that the examples are gotten with substitution, so we can say that an information point that has been gotten before has equivalent likelihood of being picked again like other information focuses which were not gotten before.

These examples or we can say that the bootstrapped tests are then sent to an aggregator which tallies the vote that how much vote a class is having. The class with dominant part casts a ballot is viewed as the anticipated mark for that obscure example.

Bagging is classified into two types as shown in Fig.4, i.e., bootstrapping and aggregation. Bootstrapping is a sampling technique where samples are derived from the whole population (set) using the replacement procedure.

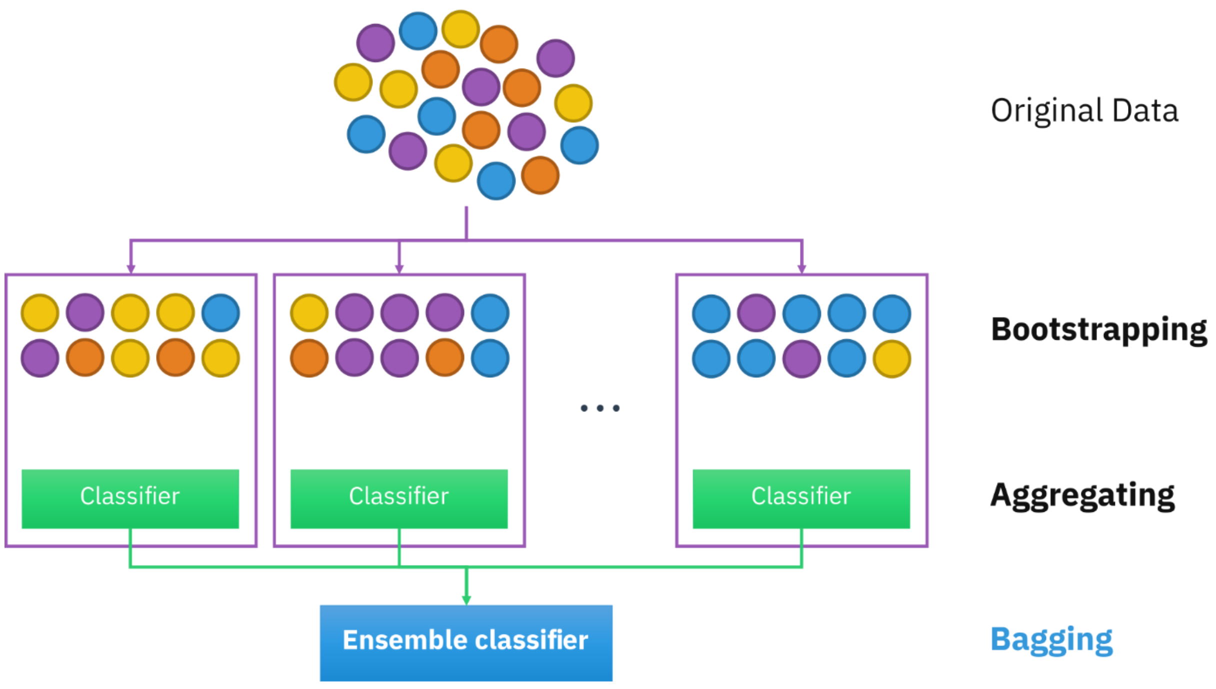

Fig 4 Types of Bagging

The sampling with replacement method helps make the selection procedure randomized. The base learning algorithm is run on the samples to complete the procedure.

Aggregation in bagging is done to incorporate all possible outcomes of the prediction and randomize the outcome. Without aggregation, predictions will not be accurate because all outcomes are not put into consideration. The aggregation is, therefore, based on the probability bootstrapping procedures or on the basis of all outcomes of the predictive models.

Bagging algorithm

Given a training data set D containing m examples, bootstrap drawing method draws a sample of training examples D_i, by selecting m examples in uniform random with replacement. It comprises two phases namely Training phase and Classification Phase.

Training Phase:

1. Initialize the parameters

2. D = {Φ}

3. H= the number of classification

4. For k=1 to h

5. Take a bootstrap sample S from training set S

6. Build the classifier D using S as a training set

7. D = DUD$_i$

8. Return D

Classification Phase:

1. Run D$_1$, D2 ...Dk on the input k

2. The class with a maximum number of the vote is chosen as the label for X.

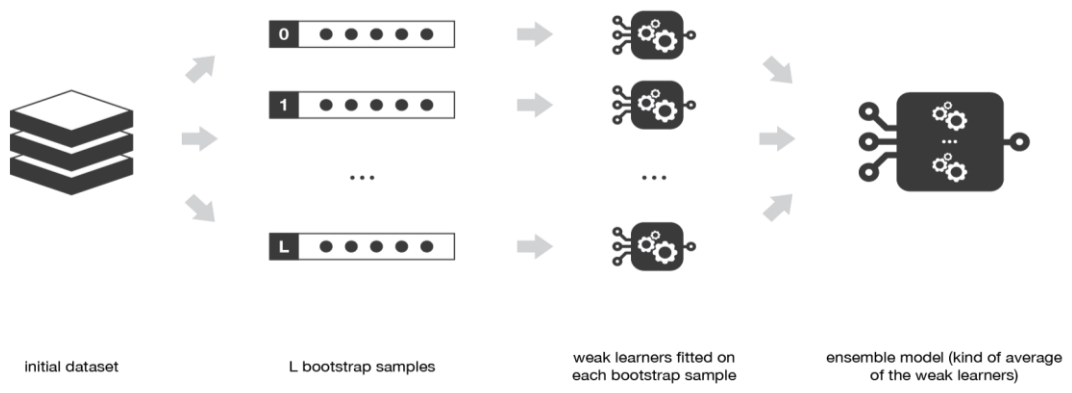
0
1
L
initial dataset
L bootstrap samples
weak learners fitted on each bootstrap sample
ensemble model (kind of average of the weak learners)

Fig 5 Algorithm of Bagging

Advantages and Disadvantages

Advantages:

- When weak learners are aggregated, they perform better than a single learner over the entire set and thus have less overfitting.
- Bagging helps is removing variance for data set having high variance and low bias.
- Bagging can be performed in parallel, as each separate bootstrap can be processed on its own before combination

Disadvantages:

- For a dataset having high bias, bagging will also lead to high bias for its aggregate
- Loss of interpretability of a model.
- Depending on the dataset, bagging can be computationally expensive.

4. Boosting

The fundamental thought behind the working of ensemble model is to improve the prescient intensity of the model by including each classifier iteratively in turn. At a specific stage when a classifier enters an ensemble then it is prepared on an informational collection haphazardly examined from the preparation informational index. Test appropriation begins with consistently stable mode and meets its development towards expanding the prediction of troublesome information focuses.

Boosting unites powerless students. On the other hand, we can say base students are made utilizing AI calculations with an alternate appropriation to frame a solid classifier with solid rules. Each time a fundamental learning calculation is applied, it produces another standard. At the end of the day, boosting is an iterative strategy in which the primary calculation is prepared all in all dataset and the resulting calculations are built by fitting the residuals of the main calculation, accordingly giving more noteworthy load to the perceptions inadequately anticipated by the past model.

AdaBoost is a variation of boosting ensemble inclining technique. AdaBoost is abbreviation for Adaptive Boosting. At first we start by allocating equivalent loads to all the information focuses. On the off chance that there is any off-base prediction i.e., the blunder of prediction because of the main calculation of essential characterization, at that point we pay more regard for those perceptions with blunder of prediction. At that point comes the utilization of the next calculation for learning the base.

3.3 Classification Techniques Used

Classification is defined as predicting the binary class variables (dependent data) from a given set of independent data such as classifying email as spam or not spam or we can say that classification of many module as consisting of faults or not.

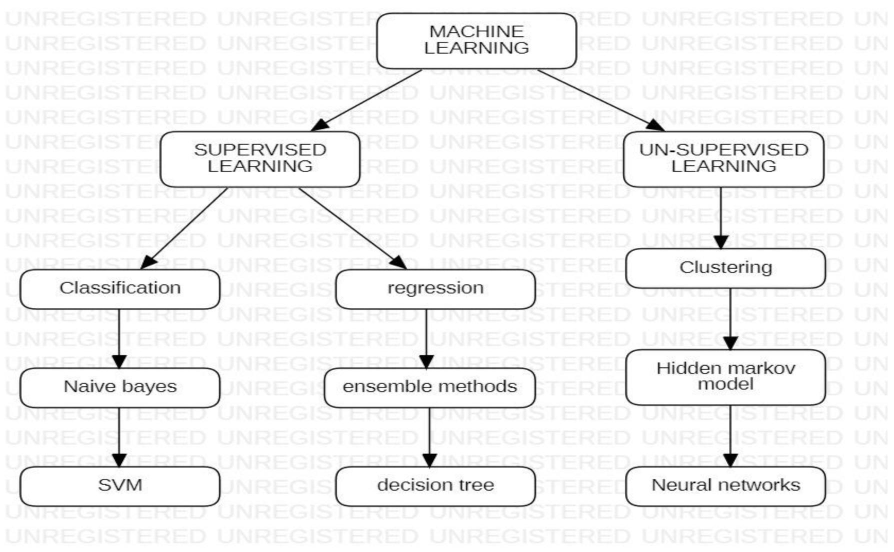

Fig 6 Classification of Machine Learning Techniques

In this study, we are using five different classifier techniques to predict the results. For implementation, we are using weka tool. All the techniques have their parameter value as default available in WEKA . Here we are also using a ten - fold cross-validation technique for splitting the original dataset into training and testing dataset. Five classification technique which we have chosen for implementation are as follows:

1. Naive Bayes:

Naive Bayes is classification method which is based on mathematical theorems which is termed as Bayes with an assumption of independence among predictors. In other words, NB classifier takes that when an specific side is present within class which is not enclosed to the presence in others.It is a simple technique and gives better accuracy results[6],[21]. It is based on one assumption that the value of one feature is not dependent on another feature value. Naive bayes theorem is a supervised learning algorithm. Supervised algorithm further classified into two parts: Classification and Regression. Naive bayes come under classification algorithms.

$$P(F/c)=(P(F)*P(c/F))/P(c)$$

Where

P(F/c) is the posterior probability.

P(F) is the class prior probability.

P(c) is the predictor prior probability.

P(c/F) is the likelihood.

Given F is the set of feature values or independent variables and c is the dependent variable or class variable having values either 0 or 1.0 value indicates not faulty and 1 indicates faulty modules.

2. Logistic Regression:

Logistic regression is a widely used statistical technique used for predicting the faulty modules or predicting the dependent variable using independent variables. Here binary Logistic regression is used to build the model as the dependent variable has binary values either 0 or 1. It models the probability of classification anomalies with 2 possible outputs. It's an extension that works as a linear regression model for classification anomalies.

In this first data is fit into the linear regression model as linear regression outputs continuous variables, so logistic regression makes use of the logistic sigmoid function to transform this output into probability value and this probability is mapped to target categorical dependent variable. A detailed description of logistic regression is given by Basili et al.(1996)[22] and Hosmer and Lemeshow(1989)[23]. Based on categories, there are 3 types of logistic regression:

a) Binary

b) Multinomial

c) Ordinal

3. Support Vector Machines:

It is another simple algorithm used for both regression and classification. It is a highly preferred technique as it gives the best accuracy with less computation. It divides the whole dataset into two parts by constructing an N-dimensional hyperplane. The hyperplane is created in the way to divide the means of 1-category in dependent value on 1-manor way and another category of the dependent variable on another side [25]. Vectors that are nearer to the hyperplane are called support vectors. Support vector machines have also been used in face recognition, medical diagnosis, and text classification [24].

4. K- nearest neighbor:

It is another simple Ml that could be using in classification and regression problems. KNN follows two properties: Lazy learning and Non parametric learning. Lazy learning means no specification about how to choose training data and non parametric learning means no there is no assumption about the data. It considers the k most similar instances to classify an instance by calculating the euclidean distance between instances[26]. KNN has also been used in pattern recognition, data mining, and intrusion detection.

5. Decision Tree:

In this, we use the REP tree which means Reducing within Erroneous inside of Pruning tree. It is the decision tree learning based technique & uses regression logic as it creates trees for every iteration and chooses the best one among all. It used the methods from C4.5 or J48 algorithm. It has also been used in intrusion detection. A study done by Zhao and Zhang [27] shows that the J48 or C4.5 algorithm produces decision tree classifier by the recursive division of the data, and using the Depth-first strategy, decision trees are grown.

3.4 Comparison of Classification Techniques Used

Table I shows the result of these five fault prediction techniques on the basis of AUC values. The result shows that in 53.33% cases logistic regression gives better prediction performance. Fig 7 shows the box plot analysis of these five fault prediction techniques. Choose the best 3 techniques among these 5 and they will be considered as base techniques for ensemble. From table I and fig 7 it can be seen that logistic regression, naive bayes and knn gives the best result.

Project name	Logistic regression	Naïve bayes	KNN	Decision tree	Support vector
ant1.3	0.686	**0.769**	0.704	0.640	0.515
ant1.4	**0.669**	0.624	0.627	0.549	0.500
ant1.6	**0.814**	0.809	0.721	0.765	0.621
ant1.7	**0.814**	0.806	0.712	0.782	0.624
Camel1.0	0.613	**0.743**	0.596	0.434	0.500
Camel1.2	0.628	0.564	**0.649**	0.613	0.508

Ivy1.1	0.669	0.667	0.637	0.648	**0.678**
Ivy1.4	0.494	**0.660**	0.507	0.419	0.500
Jedit4.0	**0.776**	0.741	0.708	0.695	0.567
Jedit4.1	**0.823**	0.773	0.742	0.721	0.654
Synapse1.0	0.672	**0.747**	0.649	0.422	0.500
Synapse1.1	**0.718**	0.716	0.709	0.677	0.660
Synapse1.2	0.748	0.756	**0.765**	0.700	0.651
Xalan2.4	**0.760**	0.742	0.659	0.729	0.498
Xalan2.6	**0.804**	0.786	0.788	0.798	0.707

Table I AUC values using different classification techniques

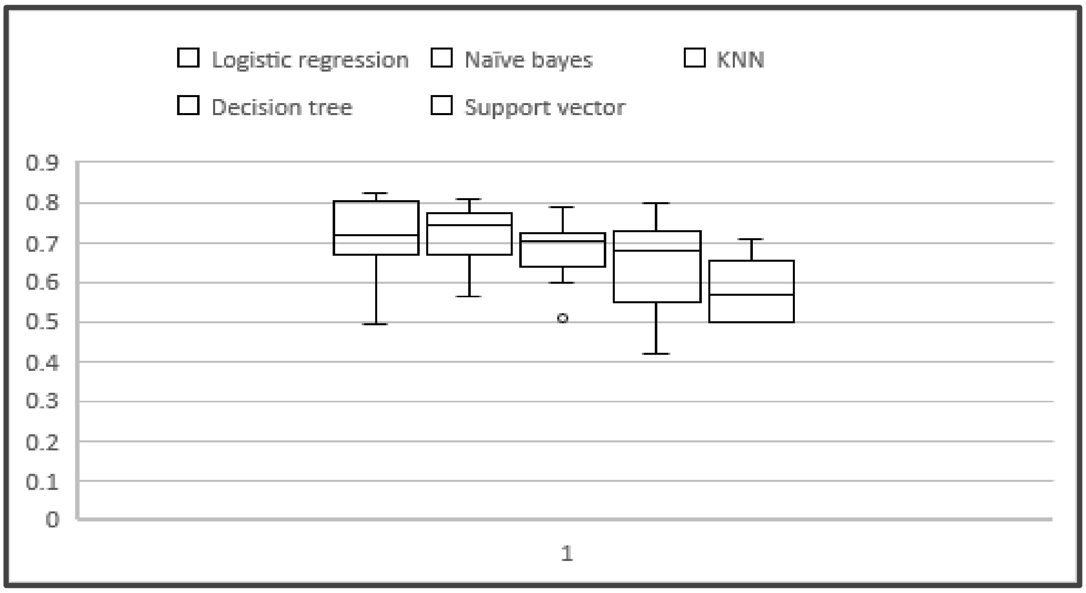

Fig 7 Box plots of five classification techniques

CHAPTER 4

EXPERIMENTAL LAYOUT

4.1 Proposed Architecture

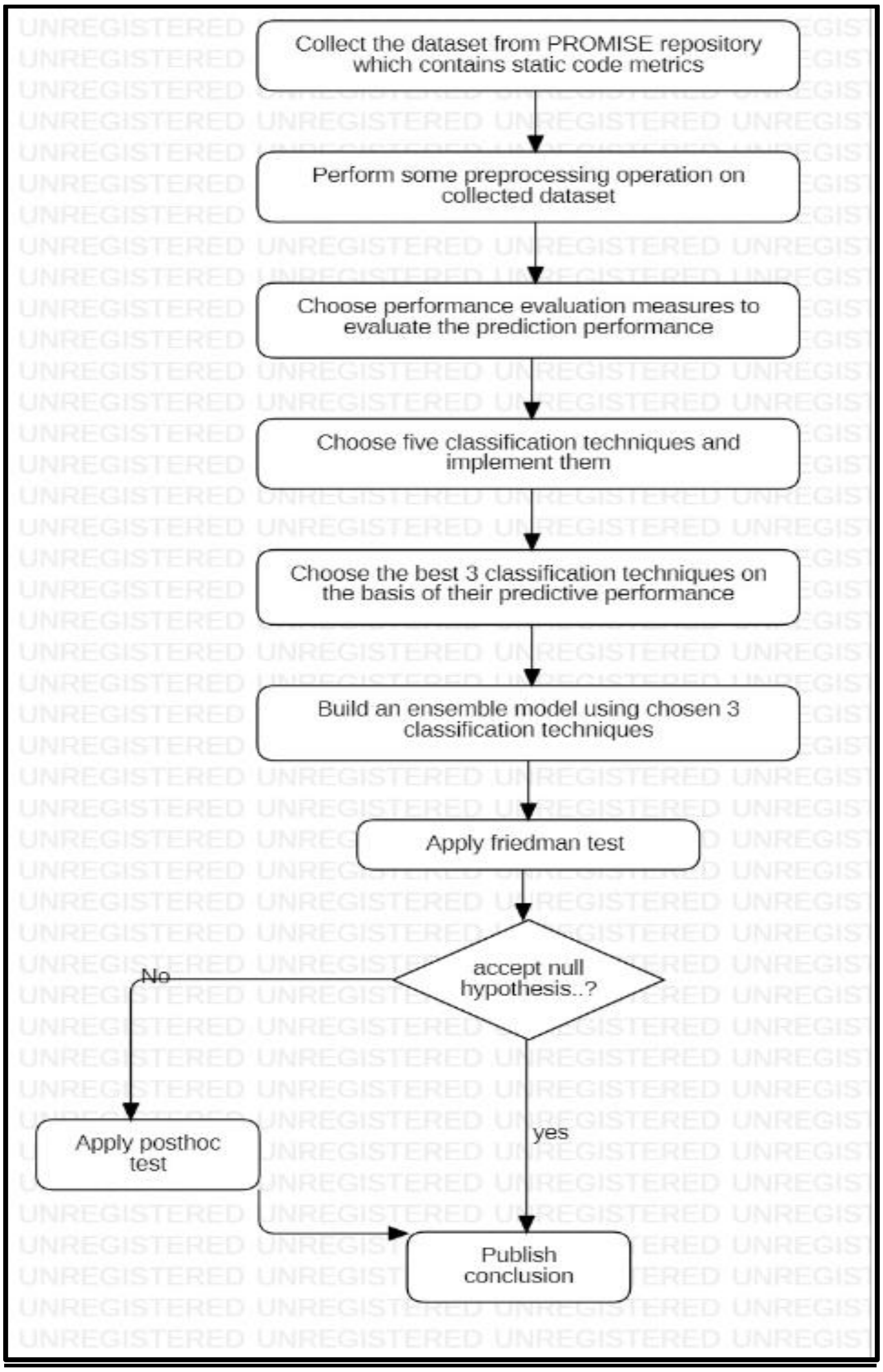
Collect the dataset from PROMISE repository which contains static code metrics
Perform some preprocessing operation on collected dataset
Choose performance evaluation measures to evaluate the prediction performance
Choose five classification techniques and implement them
Choose the best 3 classification techniques on the basis of their predictive performance
Build an ensemble model using chosen 3 classification techniques
Apply friedman test
accept null hypothesis..?
No
Apply posthoc test
yes
Publish conclusion

Fig 8 Proposed Architecture

4.2 Software fault dataset

In this section, we define independent and dependent variables used in this study and empirical data collection of the dataset used in this study.

Dependent and Independent Variables:

Independent variables used in this study are static code metrics and the dependent variable used in this study is fault proneness. Fault proneness is defined as the probability of finding faults in the class. Static code metrics used in this study are given in table II:

WMC	Weighted Method per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Objects
RFC	Response for a Class
LCOM	Lack of Cohesion in Methods
LCOM3	Lack of Cohesion in Methods version 3
NPM	Number of Public Methods
DAM	Data Access Metric
MOA	Measure of Aggregation
MFA	Measure of Functional Abstraction
CAM	Cohesion Among Methods
IC	Inheritance Coupling
CBM	Coupling Between Methods
AMC	Average Method Complexity
Ca	Afferent Coupling
Ce	Efferent Coupling
Max (CC)	Maximum McCabe’s Complexity
Avg (CC)	Average mccabe's Complexity
LOC	Line of code

Table II Static code metrics details

The definition in aforementioned metric is given in a separate report given by Jureczko and Madeyski[10] and also this report is available online.

Empirical Data Collection:

The dataset is collected from 6 projects. All are java based projects which can be downloaded from PROMISE repository which is publicly available. The datasets which have been used in this study are shown in table III.

4.3 Dataset Preprocessing

Datasets from the publicly available repositories may contain some noise or there can be some missing values which can affect the performance of the generated model so to avoid this type of problem some preprocessing is to be done on datasets such as removing unique id field, version field, class field etc. In this we have used one filter "replacing missing values with user constant" to fill the missing values if any. We have also used one more filter of weka tool to convert the bug field data type from "numeric to nominal" as software can be faulty or non faulty. In other terms we can say that the bug field contains binary values either 0 or 1.

Project name	Version	Description
ant	1.3,1.4,1.6,1.7	Java-base-build-tool
camel	1.0,1.2	a rule-based framework written in Java
ivy	1.1,1.4	Dependency Manager
jEdit	4.0.1,4.1.1,4.3.1.1	Java-bases cross-platform using-text editor
synapse	1.1.1,1.2.1	Enterprise service bus
xalan	2.5.0.1,2.6.0.1,2.7.0.1	XSLT processor

Table III Datasets used

4.4 Performance Evaluation Measure Used

In this study, we use AUC (area under ROC curve) to evaluate the prediction performance. Although the ROC (Receiver Operating Characteristics) curve is the accurate measure for prediction performance [28], it does not give the numeric values to discriminate between the results so AUC is the better choice for measuring prediction performance.AUC value is the average of all threshold values. AUC closer to 1 shows better prediction performance and closer to 0 shows poor prediction performance. A detailed description of how to calculate AUC values is given in Dejaeger et al [28].

4.5 Statistical Test Used

In this section, we describe statistical tests used in this study. Statistical tests are used to check whether there is a significant difference between the predictive performance of various techniques. We have used Friedman test and one post hoc test(nemenyi test).

4.5.1 Friedman Test:

The Friedman test is the nonparametric version of the one-way ANOVA with repeated measures. It is used to test for differences between groups when the dependent variable being measured is ordinal. This test is also applicable when the data violates the assumption which are required to run the one way ANOVA test

$$\chi^2 = \frac{12}{nk(k+1)} \sum_{i=1}^{k} R_i^2 - 3n(k+1)$$

The null hypothesis for this test is that there is no significant difference between the performances of all techniques. The alternate hypothesis is that the treatments have significant differences.

4.5.2 Nemenyi Test:

Nemenyi test is a post-hoc test which intends to find the groups of data that are alike after a statistical test with multiple comparisons (such as the Friedman test or kruskal wallis test)which rejects the null hypothesis. This test makes this test pair-wise tests for performance.

It compares all the algorithms pairwise and is based on the absolute difference of the average rankings of the classifiers.

In a significant level α the test determines the critical difference [CD]. if the difference b/w the average ranking of 2 algo is above than CD.then null hypothesis that the algo having same throughput is rejected. The function nemenyi test compares the critical difference with all the pairwise differences.

$$CD = q_\alpha \sqrt[2]{\frac{k(k+1)}{6n}}$$

CHAPTER 5

RESULTS

5.1 Discussion on Results through Box Plots and AUC values

This part represents the result of the ensemble methods answered in sec-3. I have shown results in the form using AUC values. We also performed analysis using box plot and statistical tests.

Table IV represents the AUC values obtained after applying a stacking ensemble method using naive bayes, logistic regression and k nearest neighbor on a given 15 software fault dataset. The result of table IV shows that in 60% cases stacking method gives better results, in 26.66% cases naive bayes gives better results and in 13.33% cases logistic regression gives better results.

Project name	Logistic regression	Naïve bayes	KNN	Stacking
ant1.3	0.686	**0.769**	0.704	0.723
ant1.4	**0.669**	0.624	0.627	0.628
ant1.6	0.814	0.809	0.721	**0.825**
ant1.7	0.814	0.806	0.712	**0.817**
camel1.0	0.613	**0.743**	0.596	0.467
camel1.2	0.628	0.564	0.649	**0.659**
ivy1.1	0.669	0.667	0.637	**0.675**
ivy1.4	0.494	**0.660**	0.507	0.510
jEdit4.0	0.776	0.741	0.708	**0.790**
jEdit4.1	**0.823**	0.773	0.742	0.816
synapse1.0	0.672	**0.747**	0.649	0.723
synapse1.1	0.718	0.716	0.709	**0.747**
synapse1.2	0.748	0.756	0.765	**0.770**
xalan2.4	0.760	0.742	0.659	**0.782**
xalan2.6	0.804	0.786	0.788	**0.813**

Table IV AUC values for stacking and base classification technique

Table V represents the AUC values obtained after applying a voting ensemble method using naive bayes, logistic regression and knn on a given 15 software fault dataset. We are using soft voting as we are considering the average of probabilities for combining the result of base techniques. The result of table V shows that in 66.66% cases the voting method gives better results, in 26.66% cases naive bayes gives best results and in 6.66% cases logistic regression gives best results.

Project name	**Logistic regression**	**Naïve bayes**	**KNN**	**Voting**
ant1.3	0.686	**0.769**	0.704	0.750
ant1.4	0.669	0.624	0.627	**0.674**
ant1.6	0.814	0.809	0.721	**0.820**
ant1.7	0.814	0.806	0.712	**0.824**
camel1.0	0.613	**0.743**	0.596	0.711
camel1.2	0.628	0.564	0.649	**0.671**
ivy1.1	0.669	0.667	0.637	**0.697**
ivy1.4	0.494	**0.660**	0.507	0.559
jEdit4.0	0.776	0.741	0.708	**0.794**
jEdit4.1	**0.823**	0.773	0.742	0.811
synapse1.0	0.672	**0.747**	0.649	0.728
synapse1.1	0.718	0.716	0.709	**0.771**
synapse1.2	0.748	0.756	0.765	**0.796**
xalan2.4	0.760	0.742	0.659	**0.785**
xalan2.6	0.804	0.786	0.788	**0.813**

Table V AUC values for voting and base classification technique

Table VI represents the AUC values for stacking and voting obtained from 15 different software fault dataset. As in xerces 2.6 the AUC values for both stacking and voting are the same so we are not considering that value. The result of this table shows that in 85.71% cases voting gives better results and in 14.28% cases the stacking method gives the best result as compared to the stacking method.

Project name	**Voting**	**Stacking**
ant1.3	**0.750**	0.723
ant1.4	**0.674**	0.628
ant1.6	0.820	**0.825**
ant1.7	**0.824**	0.817
camel1.0	**0.711**	0.467
camel1.2	**0.671**	0.659
ivy1.1	**0.697**	0.675
ivy1.4	**0.559**	0.510
jEdit4.0	**0.794**	0.790
jEdit4.1	0.811	**0.816**
synapse1.0	**0.728**	0.723
synapse1.1	**0.771**	0.747
synapse1.2	**0.796**	0.770
xalan2.4	**0.785**	0.782
xalan2.6	0.813	0.813

Table VI AUC values for stacking and voting

Fig 9 represents the box plot analysis of stacking and classification techniques used in building stacking ensemble models. Analysis of fig 9 shows that stacking gives the best result followed by naive bayes and logistic regression. And there is one outlier in case of knn. Fig 10 represents the box plot analysis of voting and classification techniques used in building voting ensemble models. Analysis of fig 10 shows that voting gives the best result followed by naive bayes and logistic regression. And there is one outlier in case of knn. Fig 11 represents the box plot analysis of voting and stacking. Analysis of fig 11 shows that voting gives better performance prediction as compared to stacking. Box plots give the lowest, first quartile, the peak, and third quartile results in the sample test. The centroid of the boxplot gives the middle results for the sample.

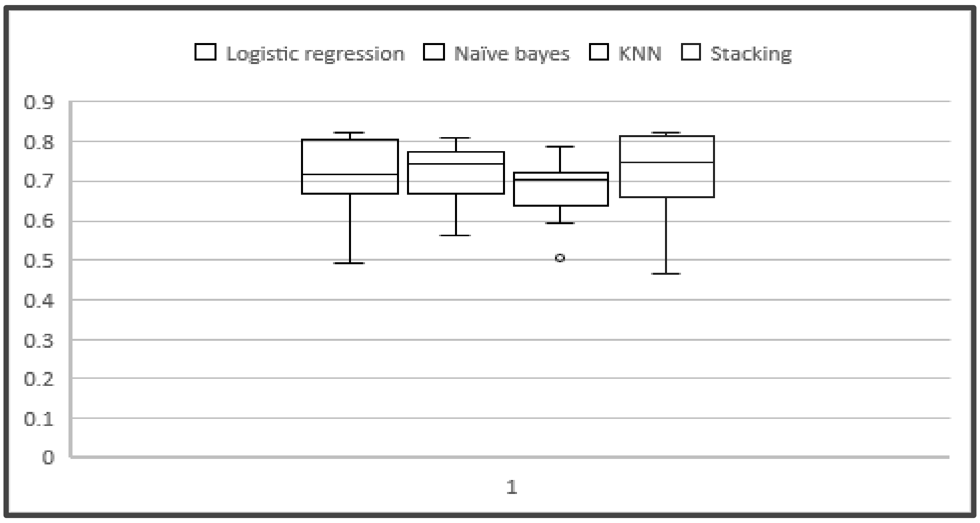

Fig 9 Box plot of stacking and base techniques

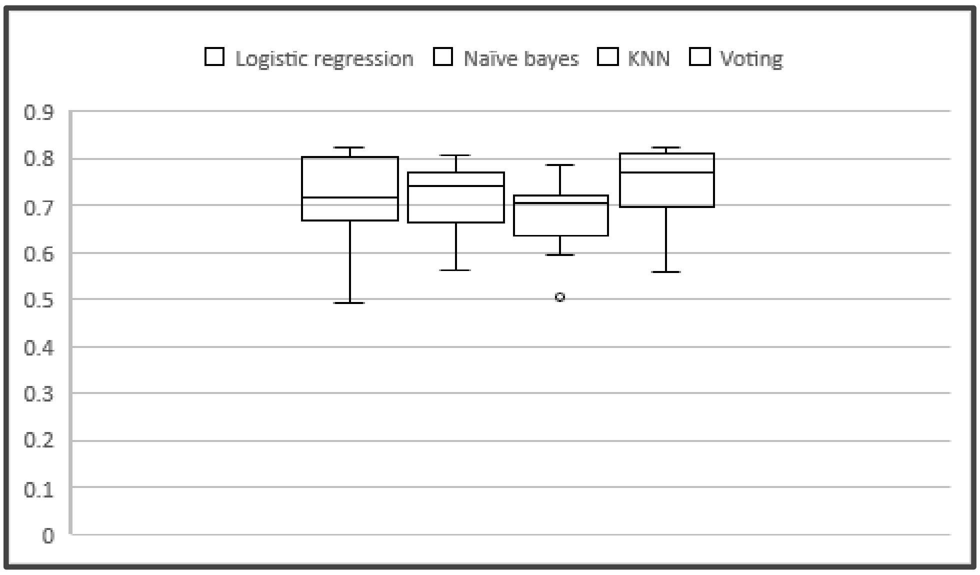

Fig 10 Box plot of voting and base technique

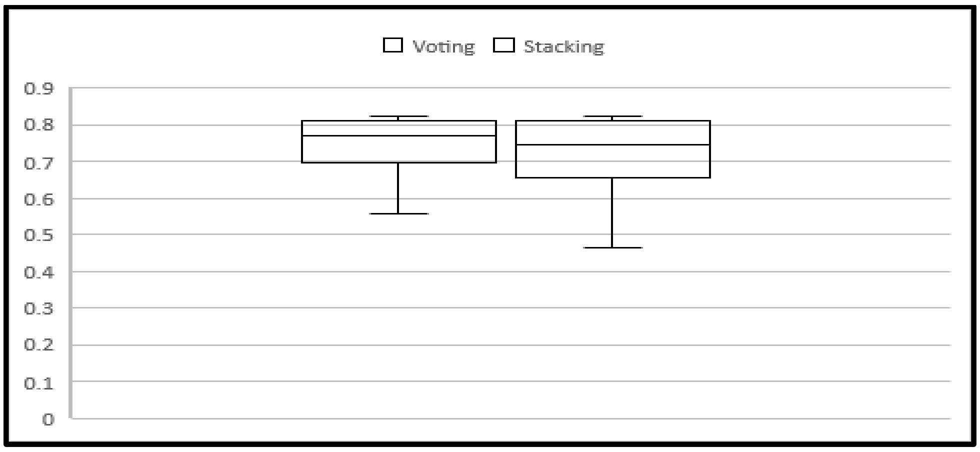

Fig 11 Box plot of voting and stacking

Table VII represents the AUC values obtained after applying a homogenous bagging ensemble method using naive bayes, logistic regression and k nearest neighbor on a given 15 software fault dataset. It can be seen from the table that all the three classification techniques gave almost same AUC values for a given dataset; there is not much difference between the results obtained. For Synapse 1.2 dataset, the AUC values obtained is same for all the three techniques and for Synapse 1.0, Naïve bayes and KNN gave same results for AUC values.

Project name	**Logistic regression**	**Naïve bayes**	**KNN**
Ant 1.3	0.800	0.816	0.840
Ant 1.4	0.719	0.539	0.691
Ant 1.6	0.786	0.794	0.726
Ant 1.7	0.826	0.808	0.786
Camel1.0	0.949	0.926	0.941
Camel1.2	0.662	0.648	0.651
Ivy1.1	0.612	0.594	0.639
Ivy1.4	0.921	0.883	0.912
Jedit4.0	0.787	0.777	0.790
Jedit4.1	0.814	0.801	0.778
Synapse1.0	0.859	0.847	0.847
Synapse1.1	0.797	0.752	0.729
Synapse1.2	0.746	0.746	0.746
Xalan2.4	0.849	0.816	0.811
Xalan2.6	0.738	0.711	0.719

Table VII AUC values for Homogeneous Bagging

Table VIII represents the AUC values obtained after applying a homogeneous boosting ensemble method using naive bayes, logistic regression and k nearest neighbor on a given 15 software fault dataset. Out of 15 datasets, logistic regression performed better than naïve bayes and KNN in 11 datasets. In 3 datasets Logistic regression and KNN gave same AUC values, i.e. they gave same performance for these 3 datasets. For 5 datasets Naïve bayes and KNN gave the same results for AUC values.

Project name	**Logistic regression**	**Naïve bayes**	**KNN**
Ant 1.3	0.792	0.761	0.816
Ant 1.4	0.719	0.556	0.674
Ant 1.6	0.803	0.794	0.740
Ant 1.7	0.822	0.811	0.774
Camel1.0	0.946	0.929	0.929
Camel1.2	0.657	0.646	0.657
Ivy1.1	0.675	0.594	0.648
Ivy1.4	0.901	0.892	0.892
Jedit4.0	0.797	0.764	0.784
Jedit4.1	0.821	0.769	0.769
Synapse1.0	0.859	0.834	0.853
Synapse1.1	0.801	0.747	0.747
Synapse1.2	0.742	0.726	0.742
Xalan2.4	0.853	0.817	0.853
Xalan2.6	0.737	0.715	0.724

Table VIII AUC values for Homogeneous Boosting

Table IX represents the AUC values for Homogeneous Bagging and Homogeneous Boosting obtained from 15 different software fault dataset. The table is made by combining the above two tables, so that comparison between the two techniques can be made.

Project Name	BAGGING			BOOSTING		
	Logistic regression	KNN	NAÏVE BAYES	Logistic regression	KNN	NAÏVE BAYES
Ant 1.3	0.800	0.816	0.84	0.792	0.760	0.816
Ant 1.4	0.719	0.539	0.691	0.719	0.556	0.674
Ant 1.6	0.786	0.795	0.726	0.803	0.795	0.741
Ant 1.7	0.827	0.808	0.787	0.823	0.811	0.774
Camel1.0	0.950	0.926	0.941	0.947	0.929	0.929
Camel1.2	0.663	0.648	0.651	0.658	0.646	0.658
Ivy1.1	0.613	0.595	0.64	0.676	0.595	0.649
Ivy1.4	0.921	0.884	0.913	0.900	0.892	0.892
Jedit4.0	0.788	0.778	0.791	0.797	0.765	0.784
Jedit4.1	0.814	0.801	0.779	0.821	0.769	0.769
Synapse1.0	0.860	0.847	0.847	0.860	0.834	0.854
Synapse1.1	0.797	0.752	0.73	0.802	0.748	0.748
Synapse1.2	0.746	0.746	0.746	0.742	0.727	0.742
Xalan2.4	0.849	0.816	0.812	0.853	0.817	0.853
Xalan2.6	0.739	0.712	0.72	0.738	0.715	0.724

Table IX AUC values for Homogeneous Boosting vs. Homogeneous Bagging

Table X represents the AUC values obtained after applying a heterogeneous bagging ensemble method using naive bayes, logistic regression and k nearest neighbor on a given 15 software fault dataset.

The result of table shows that in 13 datasets out of 15, heterogeneous bagging gave better results than basic techniques. KNN never performed better than all the other three techniques. Logistic regression and Naïve bayes performed better in case of two datasets i.e. Ant 1.6 and Synapse 1.0 respectively.

Project name	Logistic regression	Naïve bayes	KNN	Bagging
Ant 1.3	0.686	0.769	0.704	**0.783**
Ant 1.4	0.669	0.624	0.627	**0.670**
Ant 1.6	**0.814**	0.809	0.721	0.811
Ant 1.7	0.814	0.806	0.712	**0.822**
Camel1.0	0.613	0.743	0.596	**0.766**
Camel1.2	0.628	0.564	0.649	**0.665**
Ivy1.1	0.669	0.667	0.637	**0.708**
Ivy1.4	0.494	0.660	0.507	**0.688**
Jedit4.0	0.776	0.741	0.708	**0.797**
Jedit4.1	0.823	0.773	0.742	**0.830**
Synapse1.0	0.672	**0.747**	0.649	0.728
Synapse1.1	0.718	0.716	0.709	**0.777**
Synapse1.2	0.748	0.756	0.765	**0.811**
Xalan2.4	0.760	0.742	0.659	**0.774**
Xalan2.6	0.804	0.786	0.788	**0.816**

Table X AUC values for bagging and base classification technique

Table XI represents the AUC values obtained after applying a heterogeneous boosting ensemble method using naive bayes, logistic regression and k nearest neighbor on a given 15 software fault dataset. The result of table shows that in 7 cases out of 15, heterogeneous boosting gave better results than individual basic technique.

Project name	**Logistic regression**	**Naïve bayes**	**KNN**	**AdaBoost**
Ant 1.3	0.686	**0.769**	0.704	0.703
Ant 1.4	0.669	0.624	0.627	**0.671**
Ant 1.6	**0.814**	0.809	0.721	0.807
Ant 1.7	**0.814**	0.806	0.712	0.784
Camel1.0	0.613	**0.743**	0.596	0.687
Camel1.2	0.628	0.564	0.649	**0.665**
Ivy1.1	0.669	0.667	0.637	**0.689**
Ivy1.4	0.494	**0.660**	0.507	0.573
Jedit4.0	0.776	0.741	0.708	**0.792**
Jedit4.1	**0.823**	0.773	0.742	0.789
Synapse1.0	0.672	**0.747**	0.649	0.656
Synapse1.1	0.718	0.716	0.709	**0.775**
Synapse1.2	0.748	0.756	0.765	**0.772**
Xalan2.4	0.760	0.742	0.659	**0.780**
Xalan2.6	**0.804**	0.786	0.788	0.798

Table XI AUC values for bagging and base classification technique

Table XII represents the AUC values for Heterogeneous Bagging and Heterogeneous Boosting obtained from 15 different software fault dataset. The table is made by combining the above two tables, so that comparison between the two techniques can be made.

Project Name	**BAGGING**	**BOOSTING**
Ant 1.3	**0.783**	0.703
Ant 1.4	**0.670**	**0.671**
Ant 1.6	0.811	0.807
Ant 1.7	**0.822**	0.784
Camel1.0	**0.766**	0.687
Camel1.2	**0.665**	**0.665**
Ivy1.1	**0.708**	**0.689**
Ivy1.4	**0.688**	0.573
Jedit4.0	**0.797**	**0.792**
Jedit4.1	**0.830**	0.789
Synapse1.0	0.728	0.656
Synapse1.1	**0.777**	**0.775**
Synapse1.2	**0.811**	**0.772**
Xalan2.4	**0.774**	**0.780**
Xalan2.6	**0.816**	0.798

Table XII AUC values for Boosting vs. Bagging

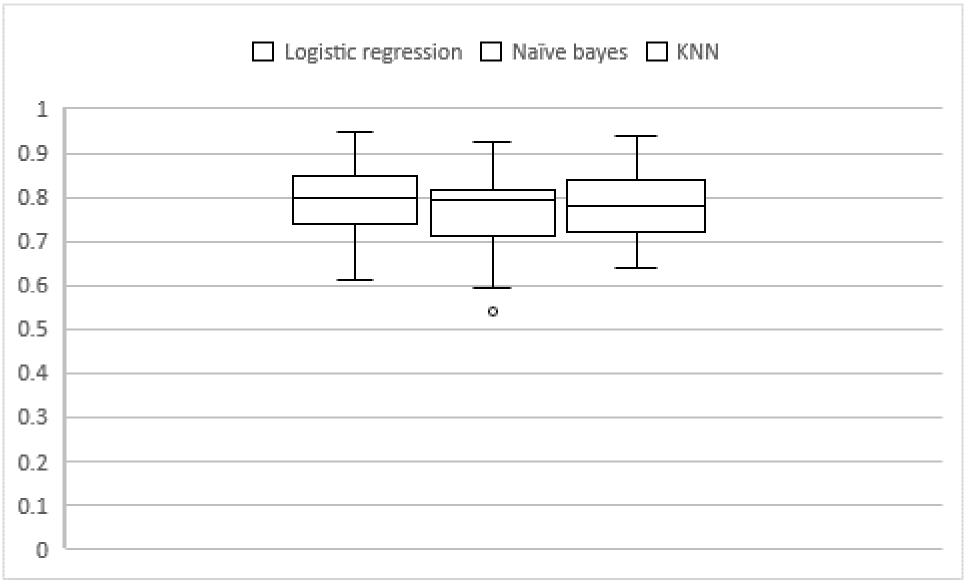

Fig. 12. Box Plot for Homogeneous Bagging

Fig 12 represents the box plot analysis of Homogeneous bagging ensemble models. Analysis of fig 12 shows that KNN gives the best result followed by Naïve Bayes and logistic regression. And there is one outlier in case of Naïve Bayes.

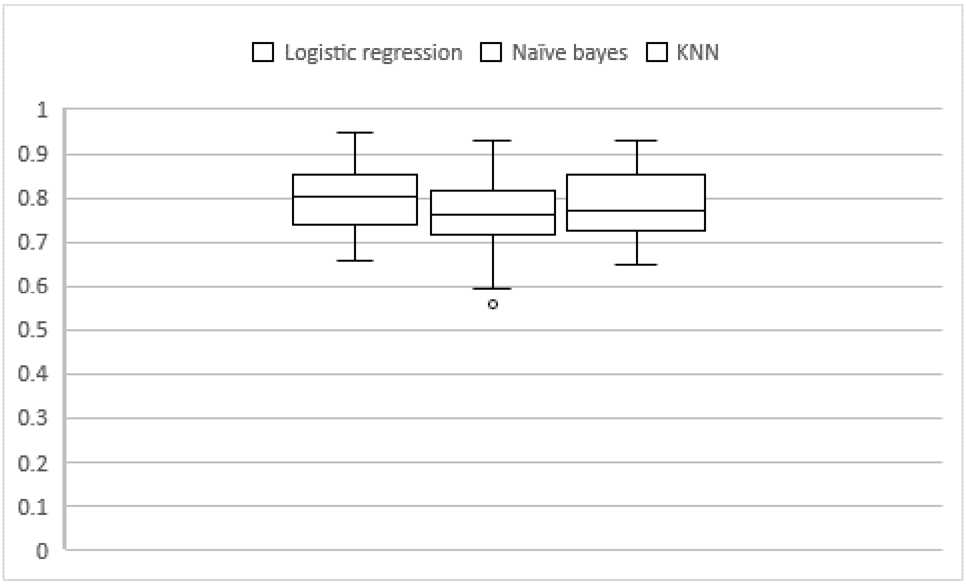

Fig.13 Box Plot for Homogeneous Boosting

Fig 13 represents the box plot analysis of Homogenous Boosting ensemble models. Analysis of fig 13 shows that Logistic regression gives the best result followed by naive bayes and KNN. And there is one outlier in case of Naïve Bayes.

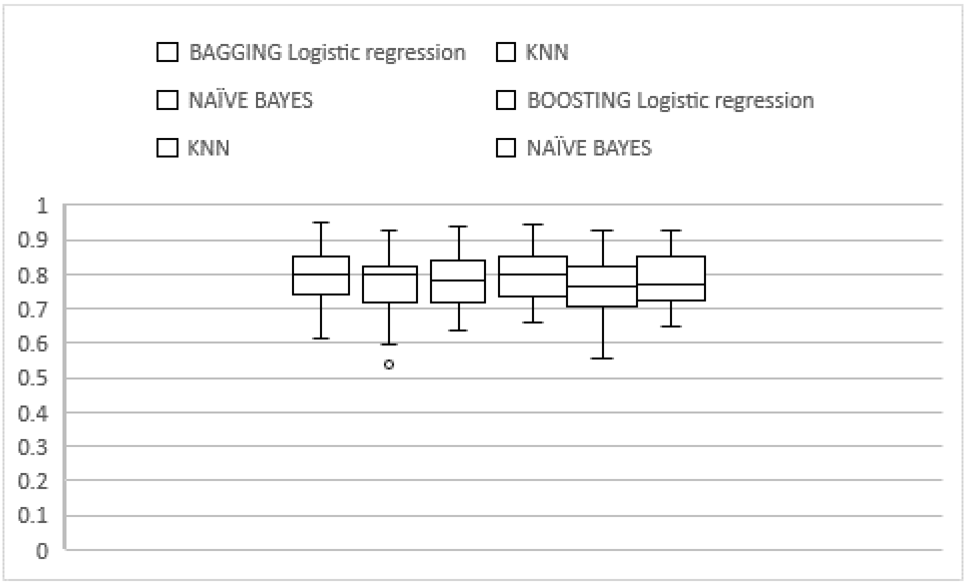

Fig. 14 Box Plot for Homogeneous Boosting vs. Homogeneous Bagging

Fig 14 represents the box plot analysis of Homogeneous Bagging and boosting. Analysis of fig 14 shows that Logistic regression gives better performance prediction as compared to others. And there is one outlier in case of Naïve Bayes in homogeneous bagging.

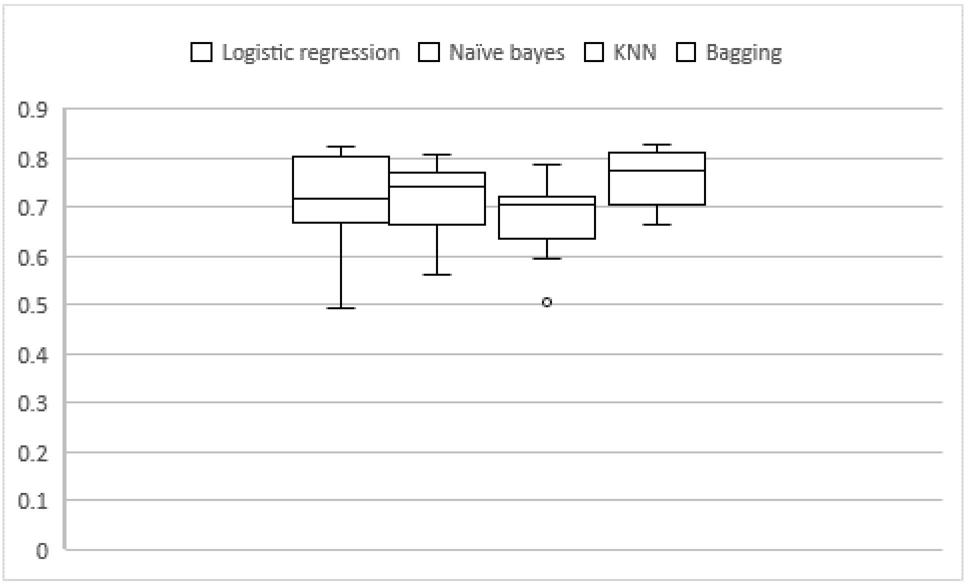

Fig. 15 Box Plot for Bagging

Fig 15 represents the box plot analysis of heterogeneous bagging. Analysis of fig 15 shows that bagging gives better performance prediction as compared to individual base classification technique. And there is one outlier in case of KNN.

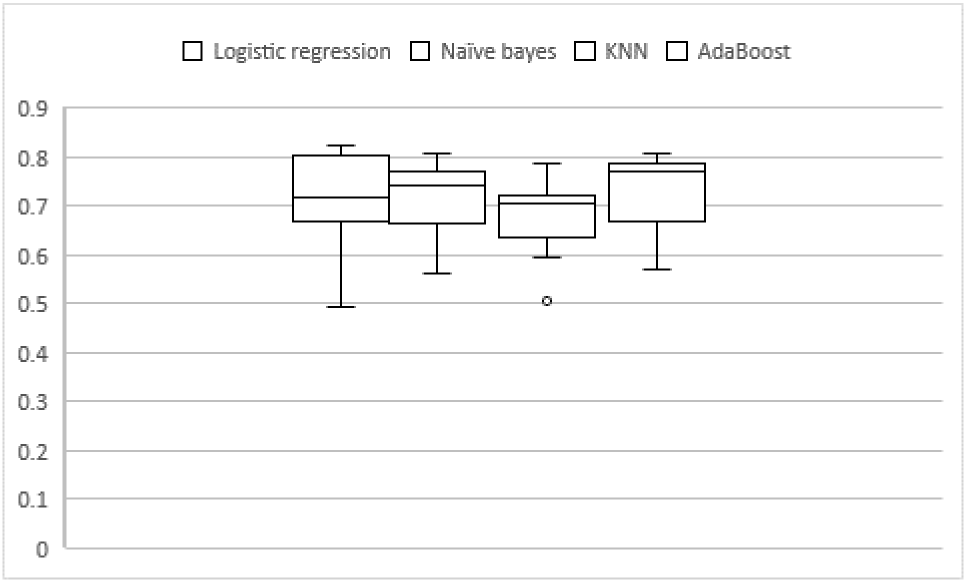

Fig.16 Box Plot for Boosting

Fig 16 represents the box plot analysis of Heterogeneous boosting. Analysis of fig 16 shows that boosting gives better performance prediction as compared to individual base classifiers. There is one outlier in case of KNN.

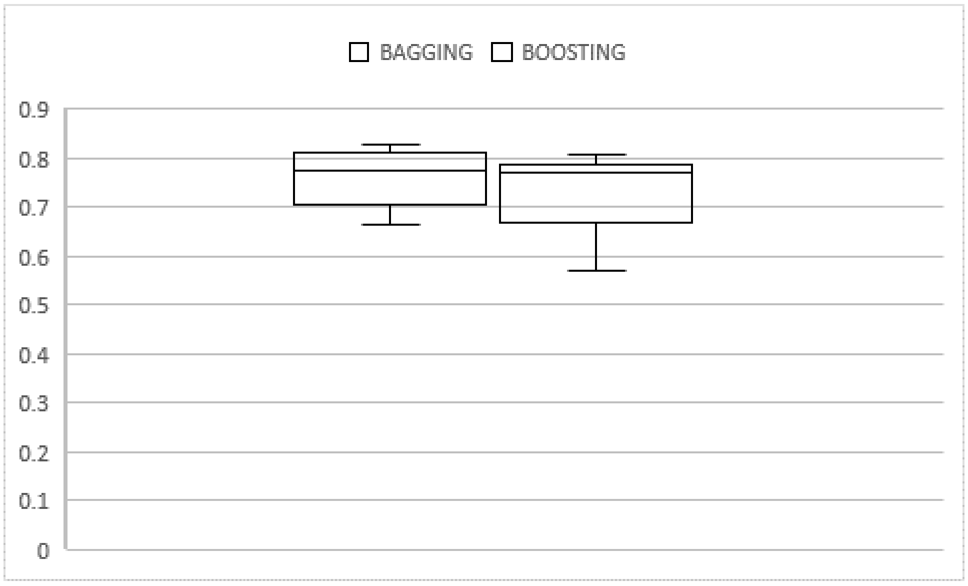

Fig. 17 Box Plot for Boosting vs. Bagging

Fig 17 represents the box plot analysis of heterogeneous boosting and heterogeneous bagging. Analysis of fig 17 shows that bagging gives better performance prediction as compared to boosting. Box plots give the lowest, first quartile, the peak, and third quartile results in the sample test. The centroid of the boxplot gives the middle results for the sample.

5.2 <u>Discussion On Results Through Statistical Tests</u>

If we statistically analyze the results of all the tables then results of table I shows that calculated χ^2- value=30.9867 and χ^2 value at (4,0.05)=9.488 so there is significant difference between the performances of techniques of table I. As when their an significance of alikeness b/w the performance in techniques so, will apply nemenyi post-hoc which is post hoc test.The result of nemenyi test on table I shows that Decision tree and support vector machines significantly outperforms logistic regression at significance level 0.05, Decision tree and support vector machines significantly outperforms naive bayes at significance level 0.05 and support vector machines significantly outperforms knn at significance level 0.05.

Results of table IV shows that calculated χ^2-value=16.04 and χ^2 value at (3,0.05)=7.815 so, when an significance of alike between the performancement of techniques of table IV. As there is a significant difference between the performances of techniques so will apply nemenyi post-hoc which is post hoc test. The result of nemenyi test on table IV shows that significant alike in performance of knn and stacking at significant levels at 0.05.

Results of table V shows that calculated χ^2-value=20.92 and χ^2 value at (3,0.05)=7.815 so there is a significant difference between the performance of techniques of table V. As there is a significant difference between the performances of techniques so will apply nemenyi test which is post hoc test. The result of nemenyi test on table V shows that when an significant values differ in performances of knn & voting there an significant values differ by the performances in naive bayes and voting at significance level 0.05.

Results of table VI shows that calculated χ^2-value=6.667 and χ^2 value at (1,0.05)=3.841 so, there is a significance of difference b/w the power of performance in stacking and voting.

Results of table VII shows that calculated χ^2 value at (3,0.05)=8.6333 so there is a significant difference between the performance of techniques of table VII. As there is a significant difference between the performances of techniques so will apply nemenyi test which is post hoc test. The result of nemenyi test on table VII shows that when a significant value differ in performances of knn & homogeneous bagging there an significant values differ by the performances in naive bayes and homogeneous bagging at significance level 0.05.

Results of table VIII shows that calculated χ^2 value at (3,0.05)=18.6333 so there is a significant difference between the performance of techniques of table VIII. As there is a significant difference between the performances of techniques so will apply nemenyi test which is post hoc test. The result of nemenyi test on table VIII shows that when an significant values differ in performances of knn & homogeneous boosting there an significant values differ by the performances in naive bayes and homogeneous boosting at significance level 0.05.

Results of table IX shows that calculated χ^2 value at (3,0.05)=24.0571 so, there is a significance of difference b/w the power of performance in homogeneous bagging and boosting.

Results of table X shows that calculated χ^2 value at (3,0.05)=26.44 so there is a significant difference between the performance of techniques of table X. As there is a significant difference between the performances of techniques so will apply nemenyi test which is post hoc test. The result of nemenyi test on table X shows that when an significant values differ in performances of knn & heterogeneous bagging there an significant values differ by the performances in naive bayes and heterogeneous bagging at significance level 0.05.

Results of table XI shows that calculated χ^2 value at (3,0.05)=12.2 so there is a significant difference between the performance of techniques of table XI. As there is a significant difference between the performances of techniques so will apply nemenyi test which is post hoc test. The result of nemenyi test on table XI shows that when an significant values differ in performances of knn & Heterogeneous Boosting there an significant values differ by the performances in naive bayes and Heterogeneous Boosting at significance level 0.05.

Results of table XII shows that calculated χ^2 value at (1,0.05)=6.6667 so, there is a significance of difference b/w the power of performance in Heterogeneous Bagging and boosting.

5.3 Discussion On Results Through Graphs Obtained

With an ensemble technique is useful, it must provide greater than, the initial participating technique in the ensemble method. Hence, with this part we have to measure performance of given ensemble technique with initial learning method. The results of differ fault prediction methods compares by using AUC results. The result of table I shows that among five fault prediction techniques logistic regression gives the good result followed by naive bayes and knn. The result of table IV shows that stacking gives good prediction accomplishment with comparison to an individual technique. Result of table V shows that voting gives great prediction production with comparison to an individual technique. The result of table VI shows that voting gives gives prediction production with comparison in stacking method. The box plot analysis shows that there is one outlier in case of knn. Statistical analysis shows that the null hypothesis is rejected then we, apply nemenyi test. The result shows that performance of stacking and voting is significantly different from naive bayes.

Fig 18 represents the comparison of average AUC values of various classification on the basis of AUC values and the last column of this figure shows the average value and the result of this diagram shows that naive bayes, logistic regression and knn gives the best prediction performance on the basis of average of AUC values.

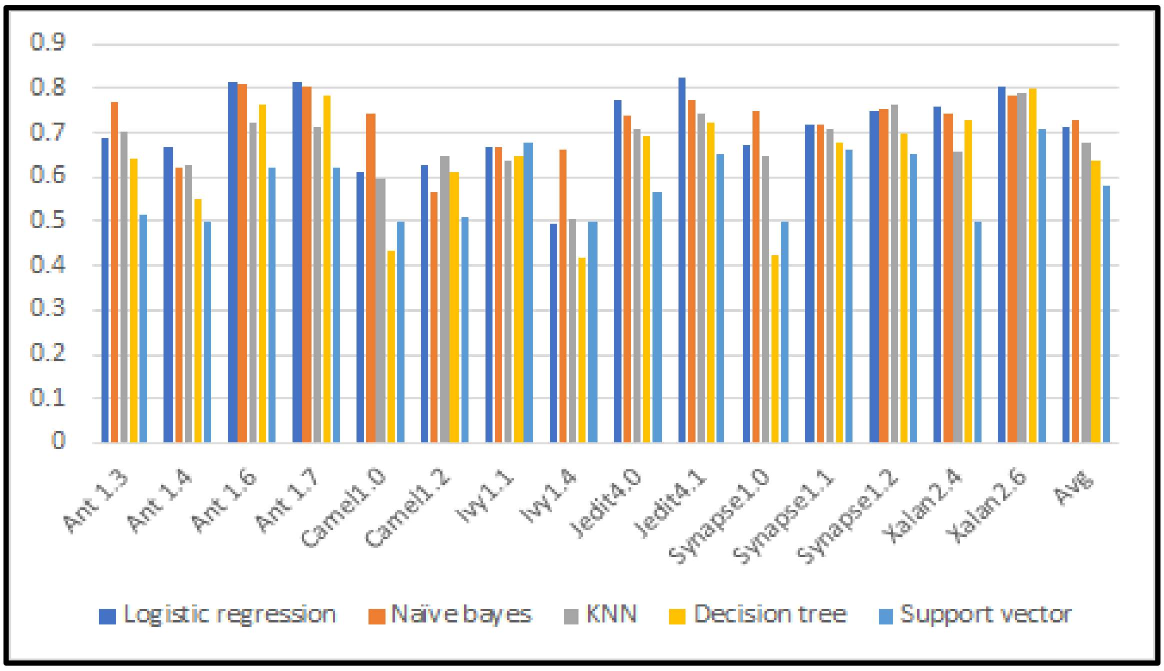

Fig 18 Collate of average AUC values between various classification techniques

Fig 19 represents the comparison of average AUC values between stacking and individual technique on the basis of AUC values and the end column of this figure shows the average value. The output of this figure show that stacking performs better than individual technique on the basis of average values of AUC values.

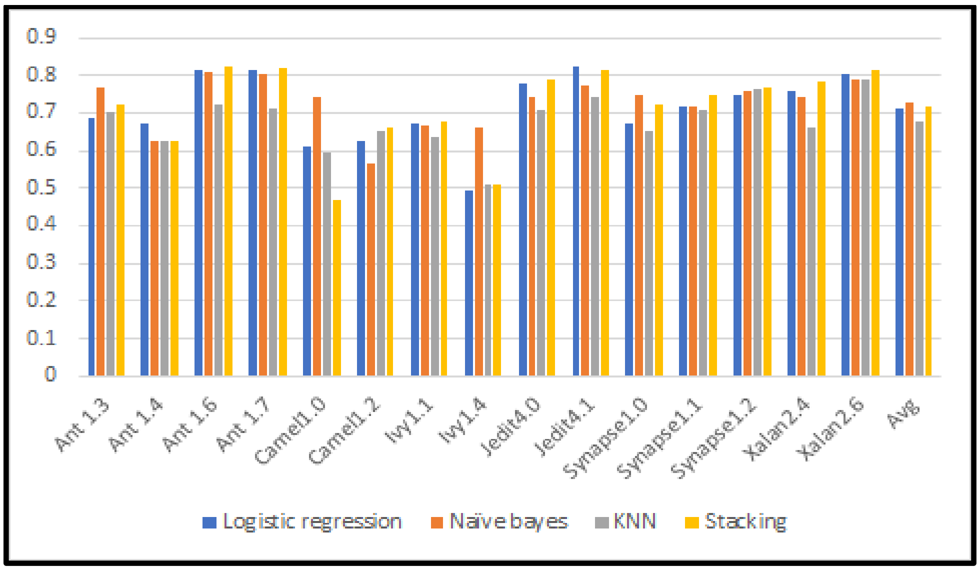

Fig 19 Collate of average AUC values between stacking and individual technique

Fig 20 represents the comparison of average AUC values between voting and individual technique on the basis of AUC value and the last column of this figure shows the average value. Results of this figure show that voting performs better than individual technique on the basis of average values of AUC values.

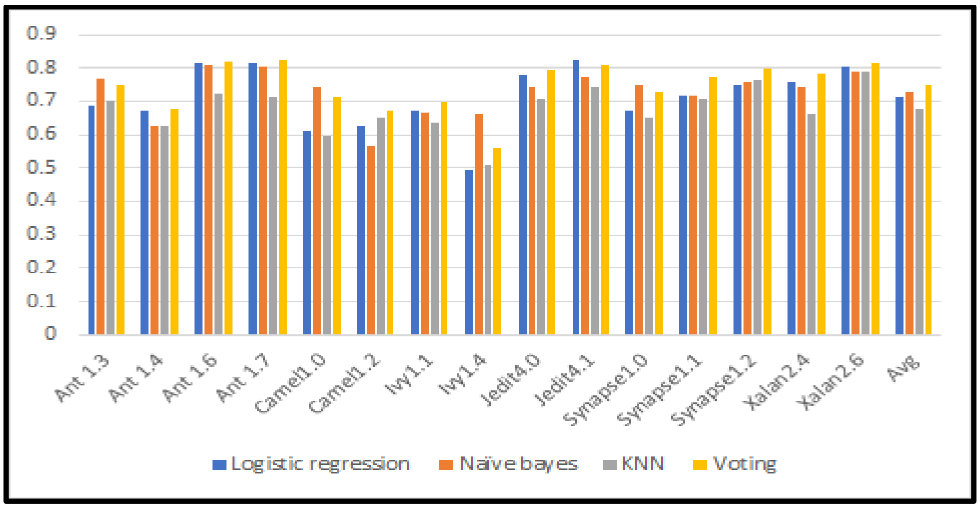

Fig 20 Collate of average AUC values between voting and individual technique

Fig 21 represents the comparison of average AUC values between voting and stacking on the basis of AUC value and the last column of this figure shows the average value. Results of this figure show that voting performs better than stacking on the basis of average values of AUC values.

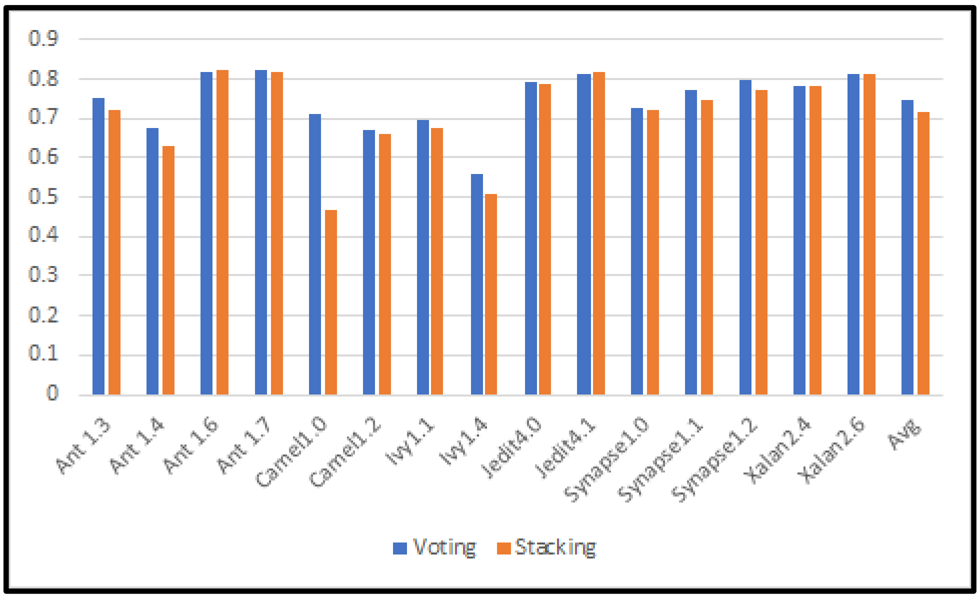

Fig 21 Collate of average AUC values between voting and stacking

Fig 22 represents the comparison of average AUC values for Homogeneous Bagging. It can be seen from the graph that all the three classification techniques gave almost same AUC values for a given dataset; there is not much difference between the results obtained.

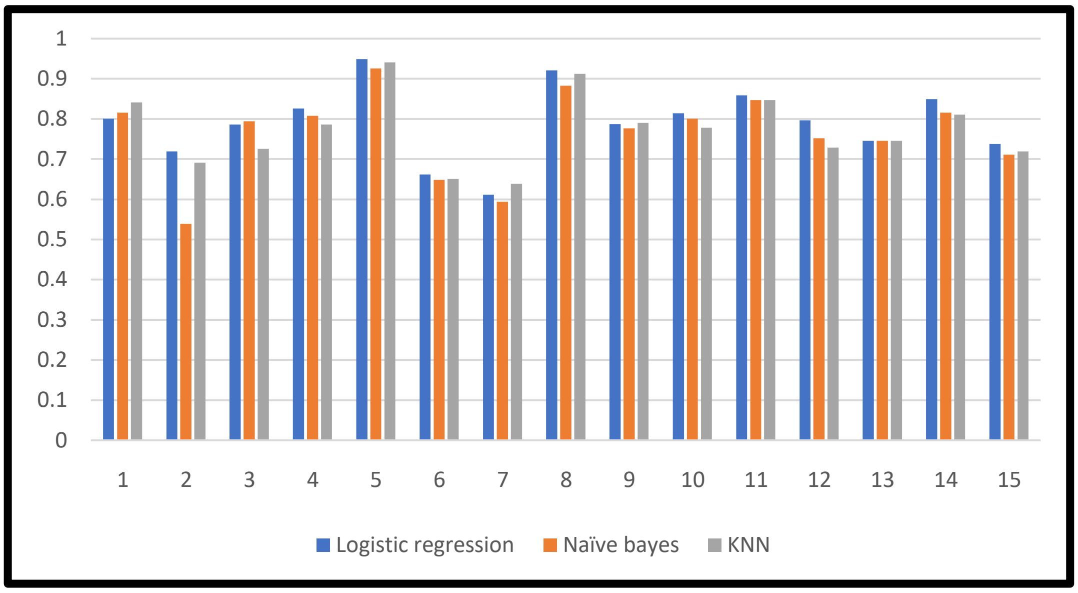

Fig 22 Collate of average AUC values for Homogeneous Bagging

Fig 23 represents the comparison of average AUC values for Homogeneous Boosting. From the graph obtained, it can be seen out of 15 datasets, logistic regression performed better than naïve bayes and KNN in 11 datasets.

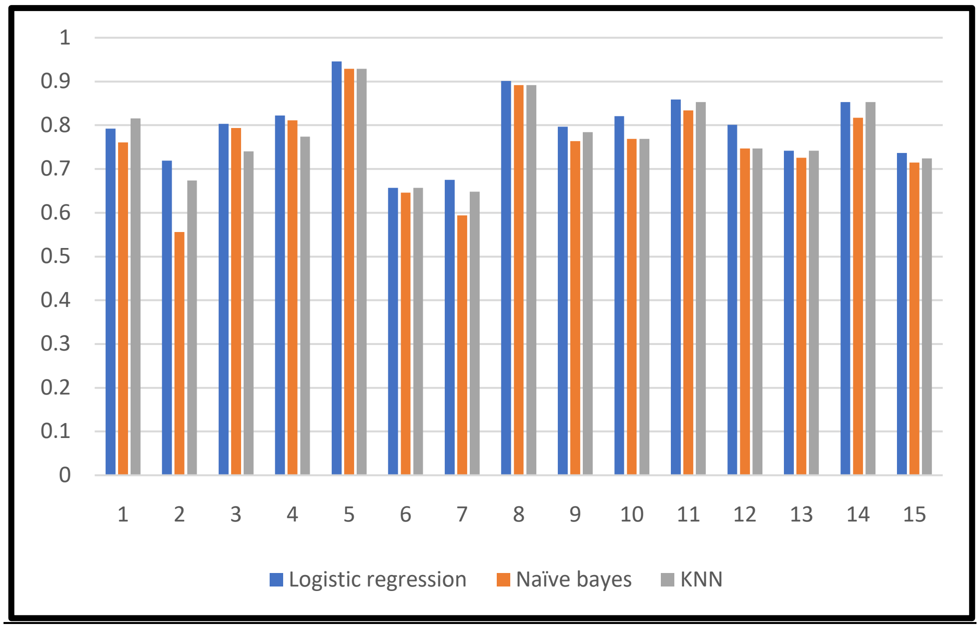

Fig 23 Collate of average AUC values for Homogeneous Boosting

Fig 24 represents the comparison of average AUC values between Homogeneous Bagging and Homogeneous Boosting on the basis of AUC value.

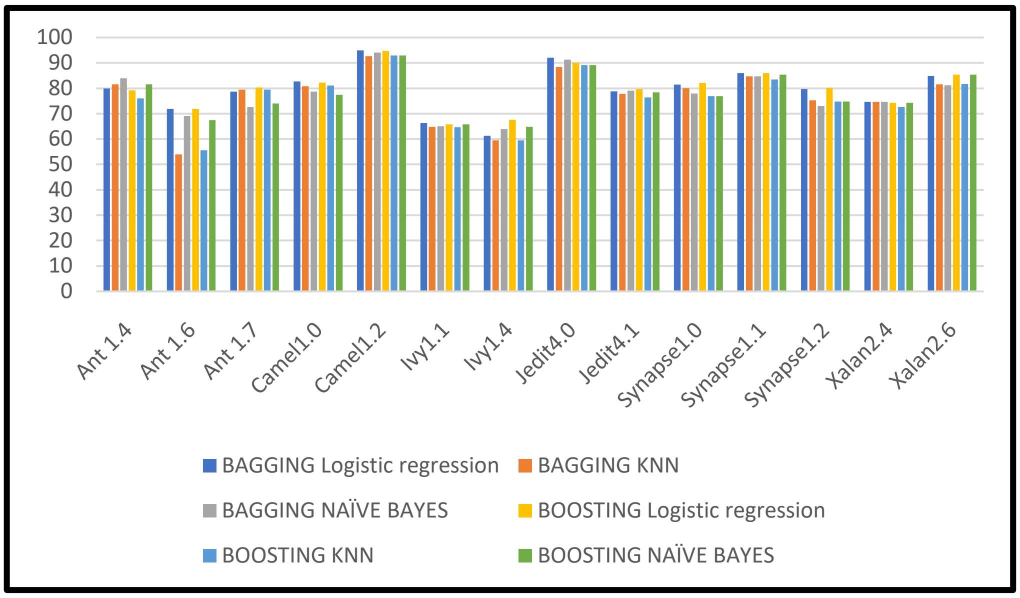

Fig 24 Collate of average AUC values for Homogeneous Boosting and Homogeneous Boosting

Fig 25 represents the comparison of average AUC values between heterogeneous bagging and individual technique on the basis of AUC values. The output of this figure show that bagging performs better than individual technique on the basis of average values of AUC values.

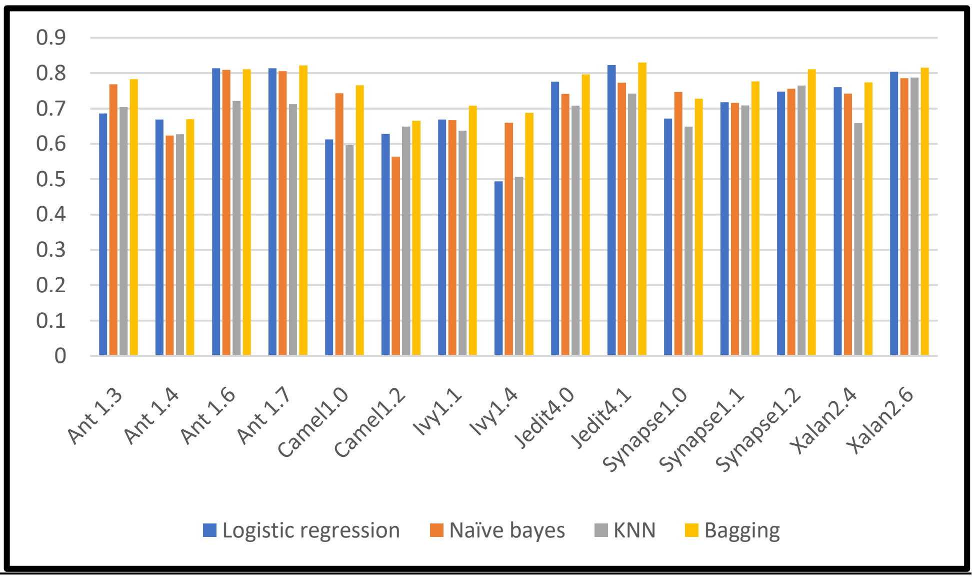

Fig 25 Collate of average AUC values for Heterogeneous Bagging

Fig 26 represents the comparison of average AUC values between boosting and individual technique on the basis of AUC values. The output of this figure show that boosting performs better than individual technique on the basis of average values of AUC values.

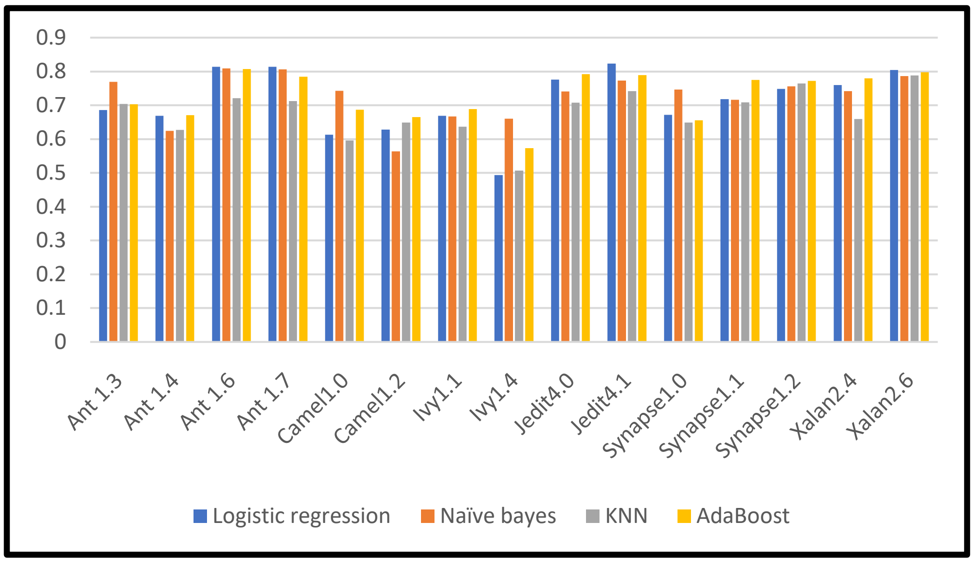

Fig 26 Collate of average AUC values for Heterogeneous Boosting

Fig 27 represents the comparison of average AUC values between bagging and boosting on the basis of AUC value. Results of this figure show that bagging performs better than boosting on the basis of average values of AUC values.

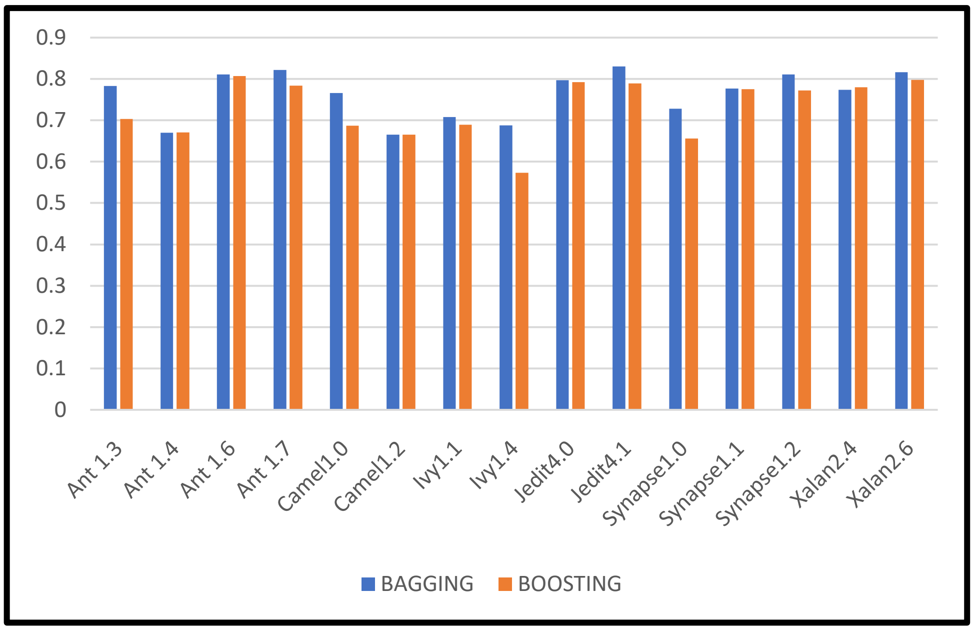

Fig 27 Collate of average AUC values for Heterogeneous Boosting vs. Heterogeneous Bagging

CHAPTER 6

CONCLUSION

The main motive of this study was to examine the various classification techniques and ensemble techniques in order to find which method gives better fault prediction performance. Thus we employed five classification techniques (naive bayes, logistic regression, decision tree, KNN, and support vector machine). Out of these 5, we selected the best 3 techniques on the basis of AUC values. The results show that naive bayes, logistic regression and knn gives better prediction performance out of 5 classification techniques. Then we employed six ensemble techniques (Heterogeneous stacking and voting, Homogeneous Bagging and Boosting and Heterogeneous Bagging and Boosting). And if we compare the results of ensemble techniques and individual techniques, ensemble gives the best result. Here we are also comparing which ensemble technique gives better prediction performance. Result shows that the voting method gives better prediction performance as compared to the stacking method and out of Bagging and Boosting (Both homogenous and heterogeneous), bagging performs better.

REFERENCES

[1] Rathore, S. S., & Kumar, S. (2016a). A decision tree logic based recommendation system to select software fault prediction techniques. Computing, 1-31.

[2] McLachlan, G.(2004). Discriminant analysis and statistical pattern recognition: volume 544. John Wiley & Sons.

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R.(2013). An introduction to statistical learning. Springer.

[4] Child, D.(1990). The essentials of factor analysis, Cassell Educational.

[5] Quinlan, J. R.(1987). Simplifying decision trees. International Journal of Man-Machine Studies, 27(3), 221-234.

[6] I. Rish. An empirical study of the naive bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial engineering. Volume 3 (pp 41-46).

[7] Mendes-Moreira, J., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: A survey. ACM Comput. Surv. (CSUR) 45(1), 10 (2012).

[8] Kalaivani, N. and Beena, R. (2018) Overview of Software Defect Prediction Using Machine Learning Algorithms. International Journal of Pure and Applied Mathematics, 118, 3863-3873.

[9] Dietterich, T. G.(2000a). Ensemble methods in machine learning. In multiple classifier system (pp. 1-15). Springer.

[10] Jureczko, M., & Madeyski, L. (2011c). Software product metrics used to build defect prediction models. Report SPR 2/2014, Faculty of Computer Science and Management, Wroclaw University of Technology.

[11] Aleem, S., Capretz, L. and Ahmed, F. (2015) Benchmarking Machine Learning Technologies for Software Defect Detection. International Journal of Software Engineering & Applications, 6, 11-23.

[12] Wang, Tao, Li, W., Shi, H. and Liu, Z. (2011) Software Defect Prediction Based on Classifiers Ensemble. Journal of Information & Computational Science, 8, 4241-4254.

[13] Perreault, L., Berardinelli, S., Izurieta, C., and Sheppard, J. (2017) Using Classifiers for Software Defect Detection. 26th International Conference on Software Engineering and Data Engineering, 2-4 October 2017, Sydney, 2-4.

[14] Hussain, S., Keung, J., Khan, A. and Bennin, K.(2015) Performance evaluation of ensemble methods for software fault prediction: An experiment. Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference, 2, 91-95.

[15] Mısırlı, A. T., Bener, A. B., & Turhan, B.(2011). An industrial case study of classifiers ensembles for locating software defects. Software Quality Journal, 19(3), 515-536.

[16] Aljamaan, H., Elish, M. O., et al.(2009). An empirical study of bagging and boosting ensembles for identifying faulty classes in object oriented software. In CIDM'09. IEEE Symposium on computational intelligence and data mining (pp. 187-194).

[17] Elish, M. O., Aljamaan, H., & Ahmad, I.(2015) Three empirical studies on predicting software maintainability using ensemble methods. Soft Computing, 1-14.

[18] Zheng, J.(2010). Cost sensitive boosting neural networks for software defect prediction. Expert Systems with Applications, 37(6), 4537-4543.

[19] Twala, B.(2011) Predicting software faults in large space system using machine learning techniques. Defence Science Journal, 61(4), 306-316.

[20] Wang, T., Li, W., Shi, H., & Liu, Z.(2011) Software defect prediction based on classifiers ensemble. Journal of Information & Computational Science, 8(16), 4241-4254.

[21] C. Catal, U. Sevim, and B. Diri. Practical development of an eclipse-based software fault prediction tool using Na¨ıve Bayes algorithm. Expert Systems with Applications, 38(3):2347 – 2353, 2011.

[22] Basili, V., Briand, L. and Melo, W. (1996) 'A validation of object-oriented design metrics as quality indicators', IEEE Transactions on Software Engineering, Vol. 22, No.10, pp.751–761.

[23] Hosmer, D. and Lemeshow, S. (1989) Applied Logistic Regression, John Wiley & Sons.

[24] Wang, X., Bi, D. and Wang, S. (2007) 'Fault recognition with labeled multi-category', 3rd Conference on Natural Computation, Haikou, China.

[25] Sherrod, P. (2003) 'DTreg predictive modeling software'.

[26] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967.

[27] Y. Zhao and Y.Zhang, "Comparison of Decision Tree Methods for Finding Active Objects," National Astronomical Observatories, Advances of Space Research, 2007.

[28] K. Dejaeger, T. Verbraken, and B. Baesens. Toward comprehensible software fault prediction models using Bayesian network classifiers. IEEE Transactions on Software Engineering, 39(2):237–257, 2013.

[29] T. Gyimothy, R. Ferenc and 1. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," IEEE Transactions on Software Engineering. vol.31, pp. 897-910, 2005.

[30] C. Catal, B. Diri and B. Ozumut, "An artificial immune system approach for fault prediction in object-oriented Software," In 2nd International Conference Dependability Compution System., pp. 1-8, 2007.

[31] Y. Zhou and H. Leung, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High Severity Faults," IEEE Transactions on Software Engineering, vol. 32, no.10, pp. 771-784, 2006.

[32] U P. Singh and A. Chug. "Software defect prediction analysis using machine learning algorithms." In International Conference of Computing and Data Science, 2017, pp. 775 781.

[33] Z.Sun, Q. Song, X. Zhu, "Using coding-based ensemble learning to improve software defect prediction, IEEE Transactions on Systems, vol.43, no.6, 2012, pp. 313-325. 13 T. Wang, W. Li, H. Shi and Z. Liu. "Software defect prediction based on classifiers ensemble. Journal of Information Systems, vol.8. no 16, 2011, pp.4241-4254.

[34] A. Kaur and K. Kamaldeep. "Performance analysis of ensemble learning for predicting defects in open source software, International Conference on Advances in Communication and Informatics, 2014, pp. 219-225.

[35] P. Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms In International Conference of Computing and Data Science, 2017, pp. 775-781

[36] H. Shamsul, L. Kevin and M. Abdelrazek, "An ensemble oversampling model for class imbalance problem in software defect prediction," IEEE Access 6, 2018.

[37] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction, International Conference of Computing and Data Science vol. 62, no. 2, 2013, pp. 434-443.

[38] A. Kaur, K. Kaur and D. Chopra "An empirical study of software entropy based bug prediction using machine learning." International Conference on Distributed systems, vol.8, no.2, 2017, pp. 599-616.

[39] C. Seiffert, T.M. Khoshgoftaar and J. V. Hulse, "Improving software-quality predictions with data sampling and boosting." IEEE Transactions on Systems Man and Cybernetics Part A. vol. 39, no. 66, pp. 1283-1294, 2009.

[40] W. Dai, Y.E. Shao, C.J.J Lu, "Incorporating feature selection method into support vector regression for stock index forecasting." Neural Computing and Applications, vol. 23, no.6. pp. 1551-561, 2012.

[41] W. Chen and C.J. Lin. "Combining SVMs with various feature selection strategies, IEEE Transactions on Software Engineering, 2005.

[42] C.L. Huang, M.C. Chen and C.J. Wang, "Credit scoring with a data mining approach based on support vector machines," Expert Systems with Applications, vol. 33, no. 4, pp. 30C. Cortes and V. Vapnik, "Support-vector networks." Machine Learning, vol. 20, no.3,847-856, 2007

[43] S.K. Shevade, S.S Keerthi CK Bhattacharyya and R.K. Murthy, Improvements to the SMO Algorithm for SVM Regression, IEEE Transactions on Neural Networks, vol II.

[44] R.Malhotra, S. Shukla. G.Sawhney, Assessment of Defect Prediction Models using Machine Learning Techniques for Object Oriented Systems." 2016 5th International Conference on Reliability. Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Sep 7-9, 2016. AllT, Amity University Uttar Pradesh, Noida India.

[45] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software." Applied Soft Computing, vol. 49, pp. 1034-1050.