# *"Comparative Analysis of Classification and Ensemble Methods for Predicting Software Fault Proneness using Process Metrics"*

A PROJECT REPORT

SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF

**MASTER OF TECHNOLOGY**
**IN**
**SOFTWARE ENGINEERING**

Submitted By

**Anjali Bansal**
**(2K19/SWE/01)**

Under the supervision of

**Prof. Ruchika Malhotra**
Head of Department (Software Engineering)
Associate Dean IRD, DTU
Department of Software Engineering
Delhi Technological University, Delhi


**DEPARTMENT OF SOFTWARE ENGINEERING**
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

**June, 2021**

**DEPARTMENT OF SOFTWARE ENGINEERING**
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

# DECLARATION

I, Anjali Bansal, 2K19/SWE/01 student of M.Tech (SWE), hereby declare that the project entitled "Comparative Analysis of Classification and Ensemble Methods for Predicting Software Fault Proneness using Process Metrics" which is submitted by me to Department of Software Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Software Engineering, has not been previously formed the basis for any fulfilment of requirement in any degree or other similar title or recognition.

This report is an authentic record of my work carried out during my degree under the guidance of Prof. Ruchika Malhotra.

Place: Delhi **Anjali Bansal**

Date: June, 2021 (**2K19/SWE/01)**

**DEPARTMENT OF SOFTWARE ENGINEERING**
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

# CERTIFICATE

I hereby certify that the project entitled "**Comparative Analysis of Classification and Ensemble Methods for Predicting Software Fault Proneness using Process Metrics**" which is submitted by Anjali Bansal (2K19/SWE/01) to Department of Software Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Software Engineering, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere.

Place: Delhi

Date:

**Prof. Ruchika Malhotra**

**SUPERVISOR**

**Professor**

**Associate Dean IRD, DTU**

**Department of Software Engineering**

# ACKNOWLEDGEMENT

I am very thankful to **Prof. Ruchika Malhotra** (Head of Department, Professor, Associate Dean IRD, DTU, Department of Software Engineering) 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.

**Anjali Bansal**

**2K19/SWE/01**

# ABSTRACT

Various researchers have worked in the subject of software defect prediction to group the modules into defective or non-defective classes. But most of the previous studies done in this field utilize static code metrics to find the predicted value. The principal motive of this study is to evaluate the impact of process metrics on fault prediction performance using various classification techniques and ensemble techniques. In this study, we have analyzed the prediction performance of several classification and ensemble techniques based on three models: models that solely contain process metrics, models that solely contain static code metrics, and models containing different combinations of both metrics. In other terms, we can say these three models work as independent variables and dependent variables are actual bug values. We have used Naive Bayes classifiers, Logistic Regression, Support Vector Machines, K-Nearest Neighbors, and Decision Trees for implementation, and data sets are collected from publicly available repositories. We have also used four ensemble techniques: Stacking, Voting, Bagging, and Boosting to evaluate the impact of process metrics on fault prediction performance. We have also analyzed which process metrics give the best result among all selected process metrics. We have analyzed the prediction performance based on AUC (Area under ROC) performance measure and we have also used Friedman test with Nemenyi post hoc test to check whether the predictive performance of various classification techniques and ensemble techniques differ significantly. The result of this study shows that the use of process metrics in fault prediction gives effective results. In most of the cases, NR metric is effective when combined with static code metrics. If we consider combined model of 2 process metrics with static code metrics then combined model of NR, NDC metric with static code metrics gives effective result. If we consider combined model of 3 process metrics with static code metrics then combined model of NR, NDC, NDPV metric with static code metrics gives effective result.

# INDEX

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| Abbreviations | Full Form |
|---|---|
| **SFP** | Software Fault Prediction |
| **AUC** | Area Under the Curve |
| **ROC** | Receiver Operating Characteristics |
| **WEKA** | Waikato Environment for Knowledge Analysis |
| **SC** | Static Code Metrics |
| **NR** | Number of Revisions |
| **NDC** | Number of Distinct Committers |
| **NML** | Number of Modified Lines |
| **NDPV** | Number of Defects in Previous Version |
| **NB** | Naïve Bayes |
| **LR** | Logistic Regression |
| **SVM** | Support Vector Machine |
| **KNN** | K Nearest Neighbor |
| **DT** | Decision Tree |

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

Software testing is detecting defects in the early stages of the software development life cycle. It is an essential and costly process to find the faults in the software. As testing is a costly process, it increases the overall project budget. Correct prediction of faults at an early stage leads to improvement in the effectiveness and quality of the software. Also, the correct prediction of faults helps in keeping the budget of the project under control. Many approaches have been suggested to identify the defects in the beginning phase of the development life cycle. This leads to the development of a fault prediction model which can predict the faults and group them in non-faulty or faulty classes.

## 1.2 Software Fault Prediction

Software fault prediction (SFP) models use different software metrics for prediction. These models use software metrics as their quality data. Previously there has been a lot of work done in this field that uses static code metrics to find various design aspects of the system. Many fault prediction models use static code metrics as quality attributes in classifying the modules. Many methods such as Logistic Regression [25], Naive Bayes [15][3], Support Vector Machines, Artificial Neural Network [25], K- Nearest Neighbor, Decision Tree, Stacking, Voting, Bagging and Boosting have been proposed in the past which shows the relationship between static code metrics and fault proneness.

## 1.3 Motivation

According to Dejaeger *et al*. [18], As of late, a few researchers directed their study toward another subject of excitement, i.e., the addition of data different from static code variables into defect prediction models such as data on inter-module connections and requirement metrics. The connection to the more ordinarily utilized static code variables remains anyway dubious. Two process metrics were used in this study but there can be other process metrics as well.

According to Okutan and Yildiz [1], As a forthcoming heading, we intend to refine our exploration to incorporate other process and software metrics in our model to uncover the connections among them and to decide the most important in fault prediction. We accept that instead of managing with a huge arrangement of software metrics, concentration on the nest ones will enhance the quality in fault prediction studies.

Both studies show the requirement to explore the process metrics in the domain of fault prediction. There are many literature reviews published in the domain of software fault prediction (e.g. [28],[6]). Most of them use static code metrics and a few studies show the relationship between process metric and fault proneness. There is no conclusive result given by authors, some authors say that process metric outperforms static code metrics but some authors say that static code metrics outperforms process metric. This thesis focuses on analyzing the predictive accuracy of process metrics as compared to static code metrics.

### 1.4 Research Questions

This thesis is mainly focused on the following research questions:

RQ1: Is process metric as effective as static code metrics in checking fault proneness?

RQ2: Which classification technique gives best result for combined model?

RQ3: To check whether ensemble techniques improve the prediction performance as compared to individual classification techniques used or not?

RQ4: Which process metric is more effective among all selected process metrics?

Thus, in this thesis, we will build a fault prediction model with the help of various classification techniques and then compare and analyze the outcome of each classification technique based on AUC values [16]. Also, we will build fault prediction models using various ensemble techniques which all the individual classification techniques as their base classification techniques. We will analyze the outcome of each technique based on several combinations of static code metrics and process metrics and also, we analyze results to find which process metric is effective in fault prediction.

### 1.5 Thesis Structure

This thesis is classified into various chapters that are as follows: Chapter 2 presents similar work which has been done in the field of software fault prediction previously. Chapter 3 presents an experimental design that describes the independent and dependent variables used in this study and also describes the dataset used in this study. In Chapter 4, research

methodology is given which describes the performance evaluation measure used, classification techniques used, ensemble techniques used, how we will implement these classification techniques, and ensemble techniques. In Chapter 5, the results of all the models using various classification techniques and ensemble techniques are given. Chapter 6 presents the result analysis which describes the results based on statistical tests, box plots, and a comparative analysis will be done to check the effectiveness of process metrics on fault proneness using various classification techniques and ensemble techniques. The conclusion of this study is described in Chapter 7.

# CHAPTER 2

# LITERATURE REVIEW

Plenty of work has been done to build software fault prediction models utilizing distinct fault prediction techniques but most of them use static code metrics as independent variables and few of them use process metrics as independent variables. This chapter examines all the past work in the field of software fault prediction which makes use of process metrics as their independent variable and also examines the previous work done which makes use of ensemble techniques in building software fault prediction models.

## 2.1 Related Work

A systematic literature review is done by Radjenovic *et al*. [6] consist of 106 papers published between 1991 and 2011 and the result of this study shows that in 49% of papers object-oriented metrics were used, in 27% of papers traditional source code metrics were used and in 24% of paper process metrics were used as independent variables. Process metrics and object-oriented metrics prediction results outperform the prediction result of source code metrics. In this literature review, code churn metrics, delta metrics, history, and developer metrics were used as process metrics. Delta metrics are concerned with different software versions whereas code churn metrics are concerned with changes in the code.

Some studies have indicated that when process metrics were used in the beginning phase of software development, they have not performed well. Results of the conducted experiments have shown that in the post-release phase of software development, process metric gives better outcomes.

Xia *et al*. [32] introduced two new process metrics: lifecycle-based management process metric and history change process metric. These metrics are based on the characteristics of the development process. They describe the effectiveness of process metrics during requirement analysis, designing, and coding. The result of this study shows that a combined model of process metrics and code metrics gives better results in defect prediction and decreases the error rate.

There are two sources to gather the process metrics: the developer's experience and software change history. Metrics gathered from software change history can be classified into two types: code churn metrics and delta metrics. Delta metrics keep track of various versions of the software and code churn metrics monitor changes in the code.

Kamei *et al*. [31] analyzed the fault density prediction based on code churn metrics and cyclomatic complexity measures. The results of this study showed that code churn metrics outperforms cyclomatic complexity measures.

Moser *et al*. [24] analyzed the Eclipse project for defect detection which consist of java files. The result showed that process metrics based on change history give the best result when contrasted with static code metrics. Hall *et al*. [28] suggested a combined model of process metrics and static code metrics for better prediction results.

Madeyski and Jureczko [19] conducted an experiment in which they used the same process metrics which we are using in this study. They considered open source and industrial projects in their research. They examined models that contain 1 process metric and all static code metrics. This study shows that the process metric improves defect prediction and gives a notable contribution and NDC and NML process metrics improve the defect prediction.

Aleem *et al*. [2] analyzed and compare the prediction outcomes of 11 machine learning techniques such as Multilayer Perceptron, Naive Bayes, 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 Bagging and Support Vector Machine gives better prediction performance.

A study done by Elish, Aljamaan, and Ahmad (2015) [8] shows that the ensemble method gives correct prediction results on the considered dataset. They analyzed the ensemble approaches for predicting change efforts and maintenance of software using two publicly available datasets.

A study was done by Perreault *et al.* [21] on five NASA datasets compared Artificial Neural Networks, Logistic Regression, Naive Bayes, Support Vector Machines, 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 use five base classification techniques such as Logistic Regression, Naive Bayes, J48, Voted-Perceptron,

and Support Vector Machine in Weka tool for software defect prediction. This study shows that Stacking gives better results among all the ensemble techniques used.

Rahman *et al.*[9] analyzed 85 versions of 12 projects to check the performance, stasis and stability of different combination of metrics. They have used 14 process metrics and 4 classification techniques(J48, Naïve Bayes, Logistic Regression, Support Vector Machine) to build the prediction model. They concluded that process metrics are more useful than code metrics and process metrics have low stasis means their values can change from release to release.

Alshehri *et al.*[30] analyzed 3 versions of the eclipse project to compare 3 different classification techniques. The results in this study are based on different combinations of change metrics and static code metrics and also, they analyzed the result based on a reduced set of change metrics. They concluded that if we are choosing the G score as a performance evaluation measure then J48 outperforms Logistic Regression and Naïve Bayes in all cases.

# CHAPTER 3

# EXPERIMENTAL DESIGN

## 3.1 Proposed Architecture

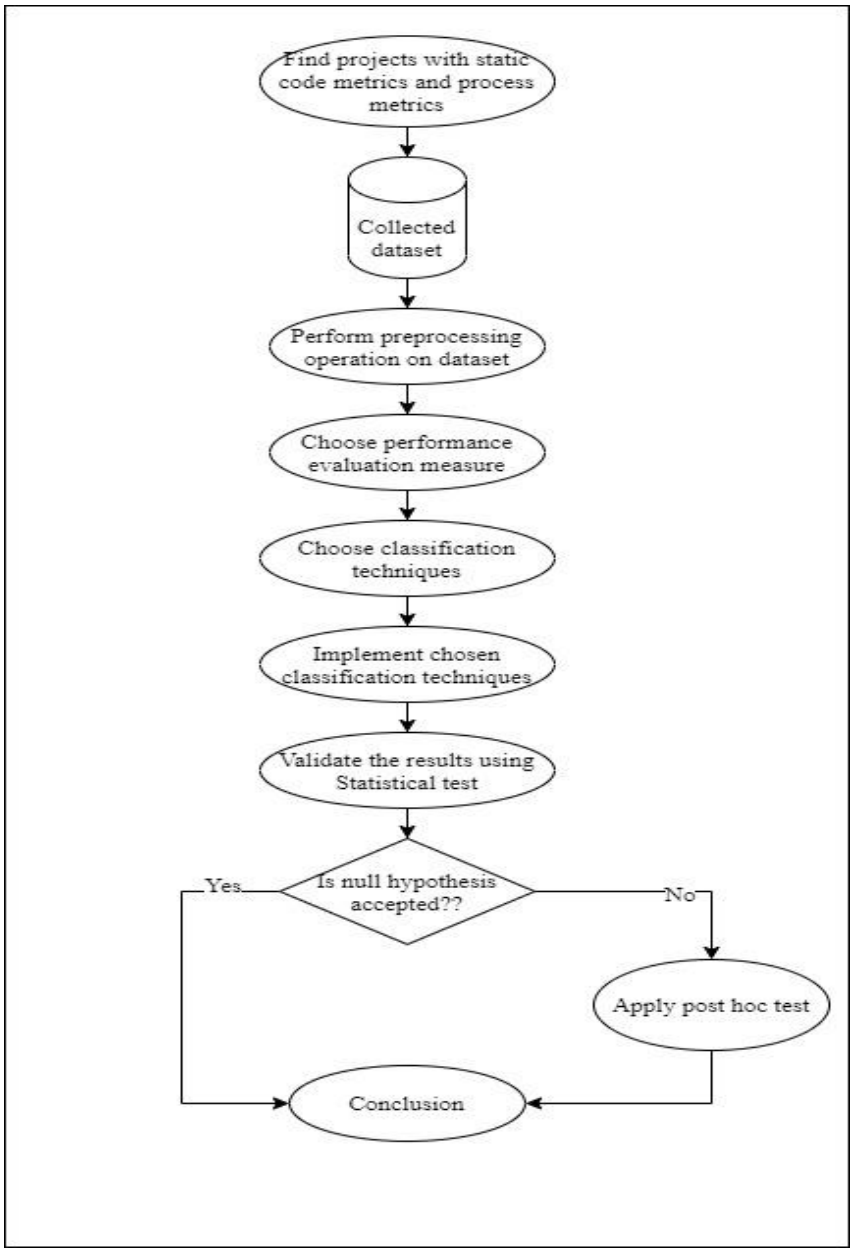


**Fig 3.1 Proposed Architecture**

In this chapter, we define the proposed architecture which we are following for implementation. Figure 3.1 shows the proposed architecture. Also, we define about independent and dependent metrics used in this study and empirical data collection of the dataset. In other terms, we can say that in this chapter we are defining the software metrics which we used in this work.

### 3.2 Dependent and Independent Variables

Independent variables that we are considering are static code metrics and process metrics, and the dependent variable is fault proneness which is defined as the likelihood of finding faults in the class.

#### 3.2.1 Static Code Metrics

Static code metrics define the design complexity and size of the software system. These metrics are widely used in building fault prediction models. Table 3.1 shows the static code metrics used in this study. The definition of the metrics of the table 3.1 is given in a separate report given by Jureczko and Madeyski [17] and also this report is available online.

**Table 3.1 Static code metrics details**

| | |
|---|---|
| **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** | Lines of Code |

### 3.1.2 Process Metrics:

Process metrics determine the quality and effectiveness of the system. Process metrics consist of more descriptive details about faulty modules These metrics are found from two sources:

a) Developer's experience

b) Software change history

Process metrics used in this study are as follows:

1. **NR**

NR stands for Number of Revisions. This metric describes the number of amendments of a java class during the evolution of examined release of the software.

2. **NDC**

NDC stands for Number of Distinct Committers. This metric describes the number of developers who submitted their modifications in a java class during the evolution of examined release of the software.

3. **NML**

NML stands for Number of Modified Lines. This metric computes the number of lines of source code that were added or taken out from the java class. Each of the submitted changes during the evolution of examined release of the software is considered.

4. **NDPV**

NDPV stands for Number of Defects in Previous Versions. This metric gives the number of faults fixed in a java class during the evolution of the past release of the software.

### 3.2 Empirical Data Collection

The dataset is collected from 5 projects. All are java-based projects. To maintain the consistent measurement of static code metrics, all projects chosen were java based. Madeyski and Jureczko [19] maintained a metric repository [11] in which all the datasets are available having both process metrics and static code metrics. Two tools have been used by Madeyski and Jureczko [19] to extract metrics from the metric repository. To extract the static code metrics, the CKJM tool [13] was used, and to extract process metrics BUGInfo [12] tool was used. The datasets which have been used in this research are shown in table 3.2.

**Table 3.2 Dataset details**

| **Dataset Name** | **Description** | **# Modules** | **#Faulty Modules** |
|---|---|---|---|
| Ant 1.4 | Java- based build tool | 265 | 40 |
| Ant 1.5 | | 401 | 32 |
| Ant 1.6 | | 524 | 92 |
| Ant 1.7 | | 1065 | 166 |
| Jedit 4.0 | Content editor written in java and free to run on any platform that supports java | 606 | 75 |
| Jedit 4.1 | | 644 | 79 |
| Synapse 1.1 | Enterprise service bus | 230 | 60 |
| Synapse 1.2 | | 269 | 86 |
| Xalan 2.5.0 | Maintenance of libraries that makes use of XSLT standard stylesheet to transform XML documents | 945 | 387 |
| Xalan 2.6.0 | | 1170 | 411 |
| Xalan 2.7.0 | | 1194 | 898 |
| Xerces 1.2.0 | | 515 | 71 |

| | | | |
|---|---|---|---|
| Xerces 1.3.0 | Parser that supports XML 1.0 | 545 | 69 |
| Xerces 1.4.4 | | 671 | 437 |

# CHAPTER 4

# RESEARCH METHODOLOGY

## 4.1 Dataset Preprocessing

Datasets from the publicly available repositories may contain some noise or there can be some missing values that 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 the unique id field, version field, class field, project name, etc. In this, we have used one filter "replacing missing values with user constant" available in the weka tool to fill the missing values if any. There can be incompatibility in the datatype of independent variable due to which there can be some error during classification. Some of the process metrics have datatype as nominal. If we directly apply classification algorithm on nominal data type fields, it leads to error. So, to remove this error, firstly we apply "nominal to string" filter and then one more filter "string to word vector" to convert nominal value into numerical value as there is no method by which we can convert nominal values to numerical. We have also used one more filter of the 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.

## 4.2 Classification Techniques

The process of predicting class or categorical data from a set of independent data is known as classification. Mathematically we can say that classification is the mapping of a function from ana input variable (independent variable) to an output variable (dependent variable). Fig 4.1 shows the classification of machine learning techniques.

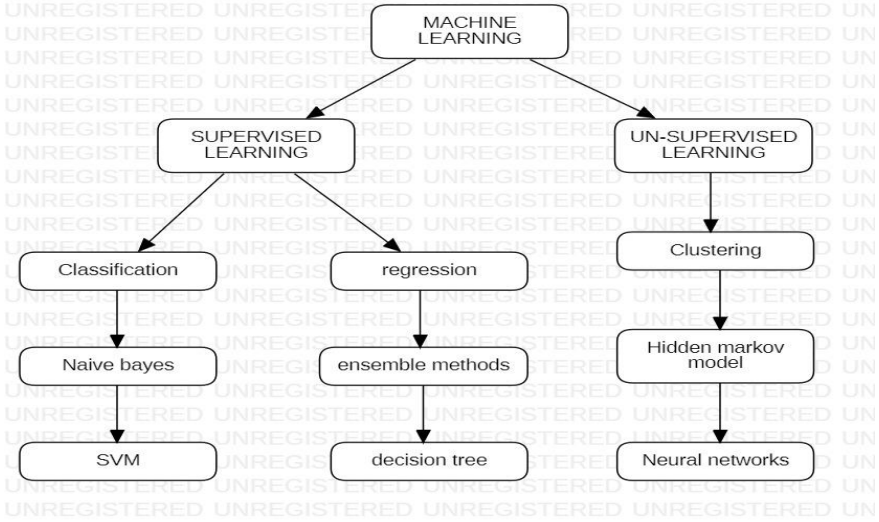


**Fig 4.1 Machine Learning Techniques**

In this research, we are using five classification techniques to build the prediction model. For implementation, we use weka machine learning tool. There can be some string data type variables due to which there can be some error so we use a filtered classifier that provides the facility of all the classifiers with additional functionality. We choose meta.filtered classifier firstly. After this, we choose any classification technique as classifier and filter as unsupervised. attribute. String to a word vector. All the techniques have their parameter values as the default values available in the weka tool. Here we are also using a 10- fold cross-validation technique for dividing the dataset into two parts: testing and training.

Five classification technique which we have chosen for implementation are as follows:

### 4.2.1 Naive Bayes

It is a supervised learning algorithm as both input and output are given to predict the value of output variable. Supervised algorithms are further classified into two parts: Classification and Regression. Naive bayes algorithms come under classification algorithms. It is a simple technique and gives better accuracy results[5],[15]. In this technique, the value of one variable is not dependent on another variable value.

There are 3 types of naive bayes model:

a) Gaussian naive bayes
b) Multinomial naive bayes
c) Bernoulli naive bayes

Naive Bayes classification technique follows Bayes theorem. In this, we find the posterior probability.

$$P(F/c) = (P(F)*P(c/F))/P(c) \quad (4.1)$$

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.

### 4.2.2 Logistic Regression

It is a widely used statistical technique applied for predicting the value of dependent variables by using independent variables. Here we are using binary Logistic regression to build the model as the dependent variable has binary values of either 0 or 1. 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. In logistic regression, the model should not be correlated means all the independent variables should be independent of each other. A detailed description of logistic regression is given by Basili *et al*. (1996) [4] and Hosmer and Lemeshow (1989) [10].

On the basis of categories, logistic regression can be classified into the following types:

a) Binary
b) Multinomial
c) Ordinal

### 4.2.3 Support Vector Machines

It is another simple algorithm used for both regression and classification. It is a powerful and flexible supervised machine learning algorithm . 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 such a way that it divides the values of one class of the dependent variable on one side and another class value of the dependent variable on another side[26]. Vectors that are nearer to the hyperplane are

called support vectors. Support vector machines can handle continuous and categorical variables.

This technique have also been used in face recognition, medical diagnosis, text classification[29].

Support vector machine kernels are used in the implementation of support vector machines. These kernels are divided into the following parts:

a) Linear Kernel
b) Polynomial Kernel
c) Radial Basis Function Kernel

In this study, we have used linear kernels to predict the fault proneness using support vector machines.

### 4.2.4 K Nearest Neighbor

It is another simple and clustering techniques used for both regression and classification problems. It is also known as the lazy learning technique as there is no specification for choosing training data, the whole dataset is considered as training data. It thinks about the k most similar instances to classify an instance by calculating the Euclidean distance between instances [27]. It is a nonlinear and versatile technique and the computation cost of this technique is high as whole data is considered as training data. In pattern recognition, data mining, and intrusion detection, K Nearest Neighbor techniques has been used.

### 4.2.5 Decision Tree

In this, we use the REP tree which means reduced error pruning tree. It comes under decision tree learner and uses regression logic as it creates trees for every iteration and chooses the best one among all. REPTree makes use of information gain to produce classification and regression trees and pruning is done with the help of a reduced error pruning algorithms. It uses the methods from C4.5 or J48 algorithm. It has also been used in intrusion detection. A study done by Zhao and Zhang[33] shows that the C4.5 algorithm or J48 produces decision tree arrangement for a given dataset by the recursive division of the data, and using the depth first strategy, decision trees are grown.

## 4.3 Ensemble Techniques

The performance of classification algorithms for software fault prediction has been accessed by a number of researchers. Therefore, there is no clear agreement in the literature that a

classification approach is the best for fault prediction. Recent advances in the field of machine learning have introduced the concept of ensemble learning to improve the prediction model's performance. The ensemble's core idea is to integrate the prediction outputs of several learning techniques such that the decision's overall performance is improved compared to the output of the individual techniques[7]. Ensemble model improves the performance of individual model by reducing bias and variance. Fig 4.2 shows the types of ensemble techniques. The 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 [20]. In heterogeneous ensembles, different learning techniques are used. Here in this study, we are using two homogeneous ensemble techniques to combine the prediction result of some classification techniques to get better prediction accuracy.

Ensemble methods follows 3 step procedure:

1. Generation: In this step, all the individual base models are generated using classification techniques for the given dataset.
2. Pruning: In this step, a subset of individual models is selected among all the generated individual models. One major advantage of this step is that ensemble of selected models gives better results as compare to ensemble of all the generated models.
3. Integration: In this step, all the selected models in the pruning steps are combined to make ensemble.

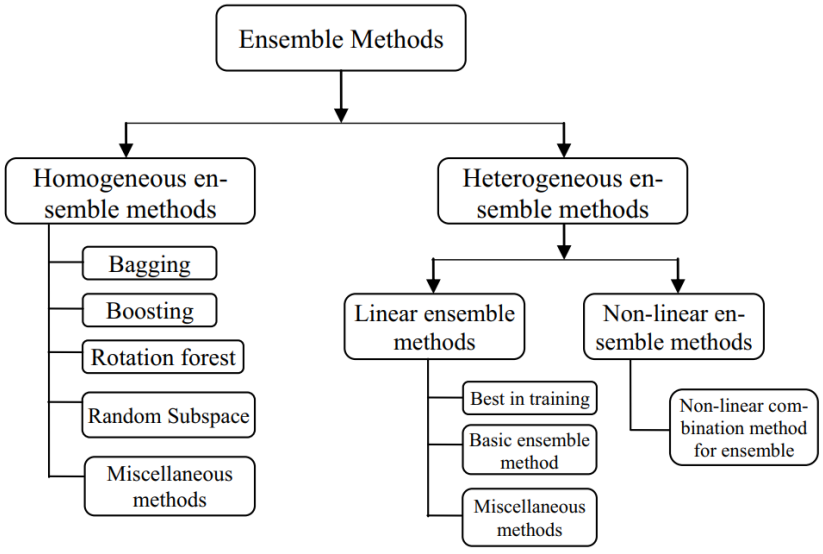


**Fig 4.2 Types of Ensemble Techniques**

Base classification techniques which we are using here to build an ensemble are Naïve Bayes, Support Vector Machine, K Nearest Neighbor, Logistic Regression, and Decision Tree (Reduced Error Pruning Tree). Ensemble techniques which we have chosen here for implementation are as follows:

### 4.3.1 Stacking

Stacking is a method of putting together classification or regression models with two-layer estimators. The base models or models for individual techniques that are used to predict the outputs based on test datasets make up the first layer. All the base classifiers work as Level 0 classifier. The second layer is made up of a Meta-Classifier or Regressor which accepts all of the base models prediction as an input and generates new ones. Here in this study, we have used logistic regression as meta classifier which is Level 1 classifier. Fig 4.3 shows the architecture of stacking method.

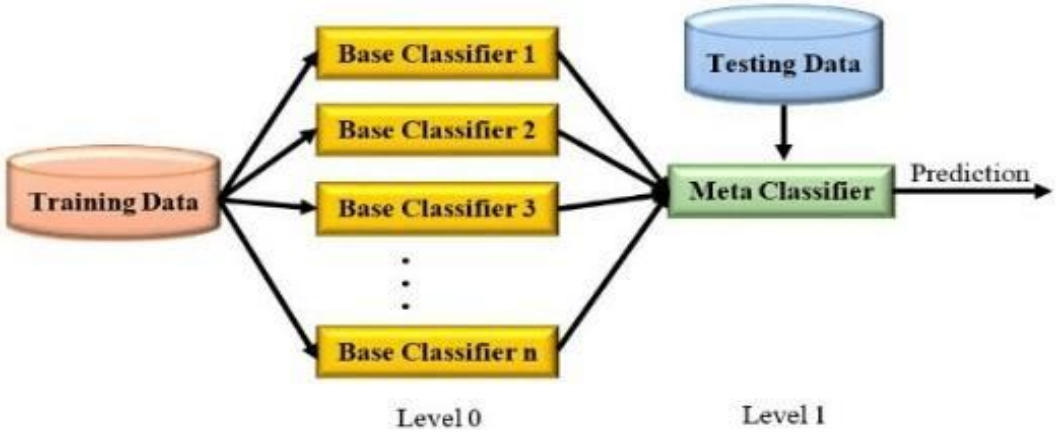


**Fig 4.3 Stacking architecture**

### 4.3.2 Voting

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

1. **Hard Voting**

   Fig 4.4 shows the architecture of hard voting in which selection of prediction outputs is done on the basis of majority voting. In fig 4.4, Class A has two votes so final predicted output is Class A.

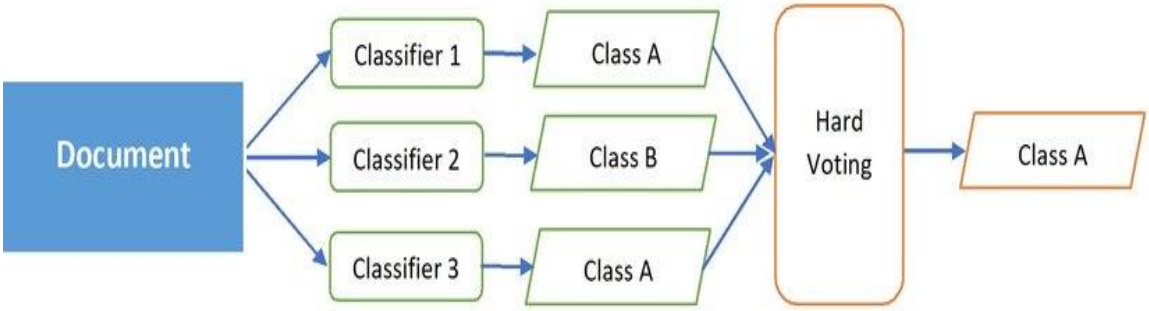


**Fig 4.4 Hard voting architecture**

2. **Soft Voting**

Fig 4.5 shows the architecture of soft voting in which selection of prediction outputs is done on the basis of average of probabilities. In fig 4.5, Class A and Class B has some probabilities for all the classifiers. Now Class A has average probability of 0.39 and Class B has average probability of 0.61 which is greater so final predicted output is Class B.

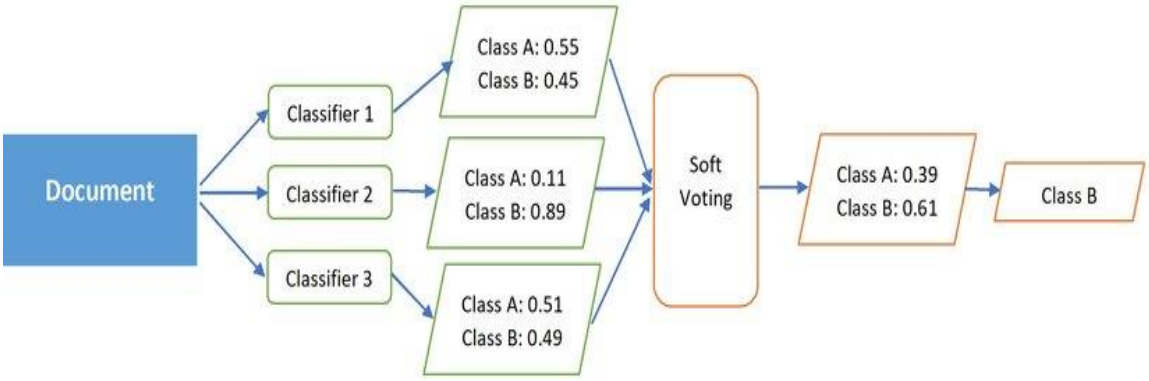


**Fig 4.5 Soft voting architecture**

### 4.3.3 Bagging

Bagging is the combination of bootstrapping and aggregating. Bagging stands for Bootstrap AGGregation. It helps in reducing variance and overfitting. Fig 4.6 shows the bagging architecture. In this type of ensemble, firstly bootstraps are created by choosing the training datasets by using replacement policy. After that, individual classification techniques are used to build the model and find the prediction for individual techniques. For combining the results and choosing the best prediction among all prediction outputs, voting is applied to get the final prediction output.

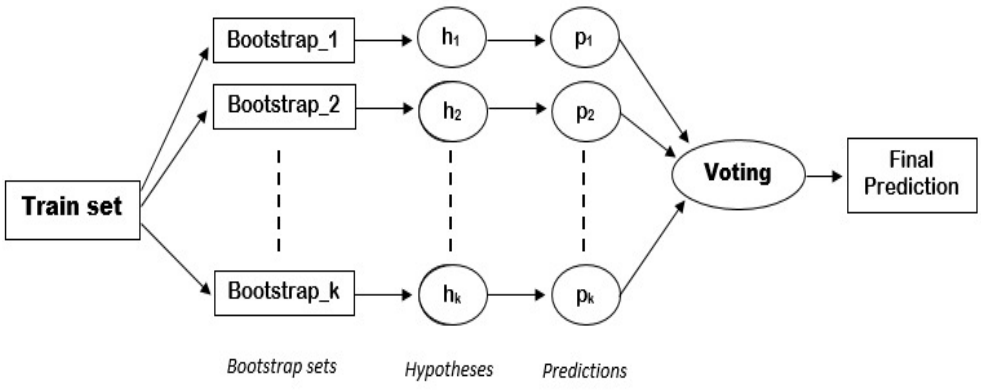


**Fig 4.6 Bagging architecture**

### 4.3.4 Boosting

Boosting is an ensemble method which enhance the capability of weak learners and convert them into strong learner. In this method, we train the weak learners sequentially while in bagging we train the learners parallelly. In boosting methods, the individual models can have unequal importance. Boosting method helps in reducing the bias. Prediction of each models are combined using voting method to get final prediction result. There are different types of boosting algorithms:

1. Adaboost
2. XGBoost
3. Gradient boosting
4. LightGBM

Here we have used Adaboost technique. Figure 4.7 shows boosting architecture. The results of Prediction 1, Prediction 2…., Prediction N are combined using voting method to get final prediction results.

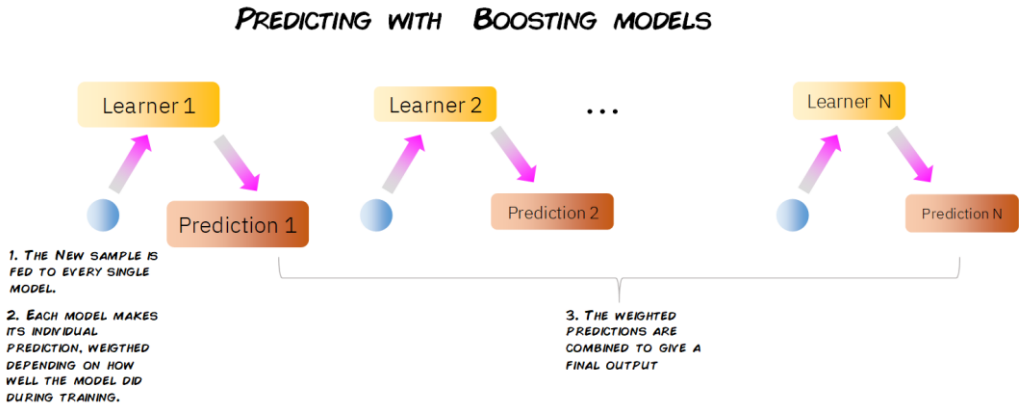


**Fig 4.7 Boosting architecture**

### 4.4 Performance Evaluation Measure

If there is a class imbalance problem in the dataset and we are trying to do classification then it is a difficult task to choose the correct performance evaluation measure. In such a scenario, AUC is used to estimate the prediction performance. Although the ROC (Receiver Operating Characteristics) curve is the accurate measure for prediction performance[18][16], it does not give the numeric values to discriminate between the results so AUC is the better choice for measuring prediction performance. Fig 4.8 represents confusion matric in which TP represents True Positive, FP represents False Positive, TN represents True Negative, and FN represents False Negative. AUC is a summarization of the ROC curve. It plots the curve between two parameters:

1. TPR: It stands for true positive rate and is defined as follows:

$$TPR=TP/(TP+FN) \quad (4.2)$$

2. FPR: It stands for false positive rate and is defined as follows:

$$FPR=FP/(FP+TN) \quad (4.3)$$

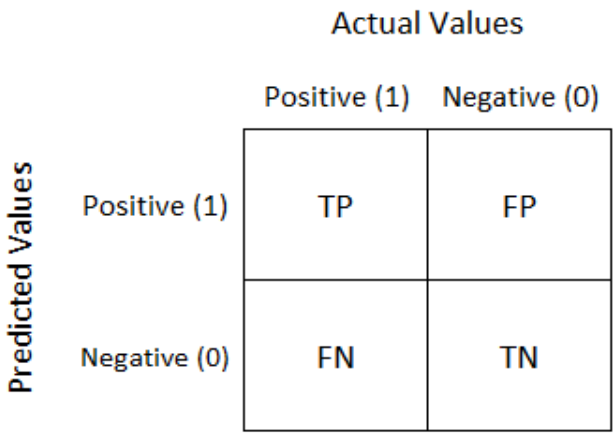


**Fig 4.8 Confusion matrix**

AUC value is the summarization of ROC curve. Value of AUC nearer to 1 shows better prediction results and nearer to 0 shows poor prediction results. AUC value is scale-invariant and classification threshold invariant. A detailed description of how to calculate AUC values is given in Dejaeger *et al* [18].

### 4.5 Statistical Analysis

We have used Friedman test with Nemenyi test to check whether the predictive performance of several classification and ensemble techniques differ significantly or not.

- **Friedman Test**

It is a non-parametric test that means this test doesn't require that data should be normal. This test follows a ranking method. Over multiple datasets, this test assigns some ranks to a set of classification techniques. The Null and alternative hypothesis for Friedman test is defined as follows:

Null Hypothesis ($H_0$): Performances of the various techniques do not differ significantly.

Alternative Hypothesis ($H_a$): Performances of the various techniques differ significantly.

Friedman test is utilized to check whether the performance of different techniques differs significantly or not. In this test, we compare the calculated $\boldsymbol{\chi}^2$-statistics value with the tabulated chi-square value to check whether the null hypothesis is accepted or rejected. We can calculate the $\boldsymbol{\chi}^2$-statistics value using the given formula.

$$\chi 2\text{- statistics} = \frac{12}{n(n+1)} \sum_{i=1}^{k} R_i^2 - 3n(k+1) \qquad (4.4)$$

- **Nemenyi Test**

This test is utilized to contrast numerous procedures and each other when the sample sizes are equivalent. It is a Post hoc test as it is applicable when there is a rejection of the null hypothesis when we use the Kruskal Wallis or Friedman test. In this, a comparison between critical distance and pairwise difference of average ranks takes place to check whether the null hypothesis is accepted or rejected. We can calculate the critical distance value using the given formula.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \qquad (4.5)$$

- **Effect Size**

Effect size in statistics is the measure of the importance of differences between two groups or samples. It is not the same as statistical significance[19]. There is no direct method to calculate effect size if we are applying the Friedman test, so we apply Kendall's W coefficient (Coefficient of Cordonance). Kendall's W coefficient can be calculated as:

$$W = \frac{\chi 2 - statistics}{n(k-1)} \qquad (4.6)$$

This coefficient value talks about the agreement between the techniques used and the coefficient value lies between 0 and 1. If the coefficient value is 0.1 to 0.2, it will be considered as a small effect size, if the coefficient value is 0.3 to 0.4, it will be considered as

medium effect size and if the value of effect size exceeds 0.5 then it will be considered as large effect size.

# CHAPTER 5

# EXPERIMENTAL RESULTS

To check whether process metrics outperform static code metrics in case of fault prediction results, we analyze the following models using various classification and ensemble techniques:

Model-1: model that solely contains static code metrics.

Model-2: model that solely contains process metrics.

Model-3: combination of process and static code metrics.

In this section, we analyze the results of all three models using various classifiers and ensemble techniques based on AUC values. We will analyze three other models also using only those classification and ensemble techniques which performs better in combined model.

Model-4: combination of 1 process metric and all static code metrics

Model-5: combination of 2 process metrics and all static code metrics

Model -6: combination of 3 process metrics and all static code metrics and

Using these 3 models, we can identify which process metric is useful or which pair of process metric is more useful in predicting the fault proneness.

## 5.1 Naive Bayes Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using naive bayes are presented in this section. Table 5.1 shows the experimental results of all the three models implemented using naive bayes classification techniques. This table shows that in 53.33% of cases combination of both metrics gives better AUC values, in 40 % cases process metric gives better AUC values and in 6.67% cases static code metric gives better AUC values. As Naïve Bayes gives better prediction performance in combined model so we will also analyze model 4, model 5 and model 6 using Naïve Bayes classification technique.

**Table 5.1 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using NB**

| Dataset Name | Combined | Static Code Metric | Process Metric |
|---|---|---|---|
| **Ant1.4** | 0.746 | 0.668 | **0.789** |
| **Ant1.5** | **0.831** | 0.805 | 0.800 |
| **Ant1.6** | **0.862** | 0.841 | 0.840 |
| **Ant1.7** | **0.889** | 0.846 | 0.876 |
| **Jedit4.0** | 0.902 | 0.826 | **0.914** |
| **Jedit4.1** | **0.926** | 0.837 | 0.906 |
| **Synapse1.1** | 0.694 | **0.707** | 0.475 |
| **Synapse1.2** | **0.761** | 0.754 | 0.694 |
| **Xalan2.5.0** | 0.723 | 0.621 | **0.732** |
| **Xalan2.6.0** | **0.865** | 0.822 | 0.810 |
| **Xalan2.7.0** | 0.973 | 0.932 | **0.991** |
| **Xerces 1.2.0** | 0.737 | 0.543 | **0.773** |
| **Xerces1.3.0** | **0.830** | 0.786 | 0.650 |
| **Xerces1.4.4** | **0.806** | 0.755 | 0.713 |

Table 5.2 shows the experimental results of model 4 implemented using the naïve bayes classification technique. This table shows that NR metrics is more useful in predicting the fault proneness as in 71.42% of cases the combination of NR metrics with all static code metrics gives better AUC values.

**Table 5.2 AUC values of models which contain combination of SC and 1 process metric using NB**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.762** | 0.705 | 0.738 | 0.661 |
| **Ant1.5** | **0.823** | 0.821 | **0.823** | 0.805 |
| **Ant1.6** | **0.862** | 0.850 | 0.860 | 0.840 |
| **Ant1.7** | **0.876** | 0.865 | 0.867 | 0.853 |
| **Jedit4.0** | **0.879** | 0.868 | 0.867 | 0.833 |
| **Jedit4.1** | 0.887 | **0.889** | 0.888 | 0.866 |
| **Synapse1.1** | 0.702 | 0.705 | 0.705 | **0.708** |

| | | | | |
|---|---|---|---|---|
| **Synapse1.2** | 0.757 | 0.761 | **0.763** | 0.755 |
| **Xalan2.5.0** | 0.708 | 0.666 | **0.709** | 0.634 |
| **Xalan2.6.0** | **0.863** | 0.834 | 0.858 | 0.833 |
| **Xalan2.7.0** | **0.977** | 0.952 | 0.953 | 0.941 |
| **Xerces1.2.0** | **0.680** | 0.608 | 0.634 | 0.648 |
| **Xerces1.3.0** | **0.822** | 0.817 | 0.809 | 0.796 |
| **Xerces1.4.4** | **0.828** | 0.789 | 0.784 | 0.756 |

Table 5.3 shows the experimental results of model 5 implemented using naïve bayes classification technique. This table shows that NR metric combining with NDC metric gives better prediction performance as in 57.14% of cases combination of static code metrics, NR metrics and NDC metric gives better AUC values.

**Table 5.3 AUC values of models which contain combination of SC and 2 process metrics using NB**

| **Dataset Name** | **NR+NDC+SC** | **NR+NML+SC** | **NR+NDPV+SC** | **NDC+NML+SC** | **NDC+NDPV+SC** | **NML+NDPV+SC** |
|---|---|---|---|---|---|---|
| **Ant1.4** | **0.766** | 0.743 | 0.758 | 0.753 | 0.700 | 0.733 |
| **Ant1.5** | 0.831 | 0.824 | 0.824 | **0.832** | 0.820 | 0.823 |
| **Ant1.6** | **0.864** | 0.860 | 0.861 | 0.862 | 0.849 | 0.858 |
| **Ant1.7** | **0.885** | 0.878 | 0.880 | 0.878 | 0.869 | 0.872 |
| **Jedit4.0** | **0.900** | 0.882 | 0.881 | 0.890 | 0.870 | 0.869 |
| **Jedit4.1** | **0.912** | 0.889 | 0.901 | 0.911 | 0.904 | 0.904 |
| **Synapse1.1** | 0.700 | 0.697 | 0.703 | 0.694 | **0.705** | **0.705** |
| **Synapse 1.2** | 0.763 | 0.757 | 0.758 | **0.768** | 0.763 | **0.765** |
| **Xalan2.5.0** | 0.711 | 0.712 | 0.714 | 0.713 | 0.676 | **0.717** |
| **Xalan2.6.0** | 0.864 | 0.864 | **0.865** | **0.865** | 0.842 | 0.862 |
| **Xalan2.7.0** | **0.982** | 0.968 | 0.980 | 0.959 | 0.954 | 0.955 |
| **Xerces1.2.0** | 0.685 | 0.678 | **0.738** | 0.653 | 0.696 | 0.711 |
| **Xerces1.3.0** | **0.840** | 0.807 | 0.829 | 0.834 | 0.825 | 0.817 |
| **Xerces1.4.4** | **0.840** | 0.795 | 0.826 | 0.803 | 0.787 | 0.784 |

Table 5.4 shows the experimental results of model 6 implemented using naïve bayes classification technique. This table shows that NR metric combining with NDC metric and NDPV metric gives better prediction performance as in 78.57% of cases combination of static code metrics, NR metrics, NDC metric and NDPV metric gives better AUC values.

**Table 5.4 AUC values of models which contain combination of SC and 3 process metrics using NB**

| Dataset Name | NR+NDC+ NML+SC | NR+NDC+N DPV+SC | NR+NML+N DPV+SC | NDC+NML+N DPV+SC |
|---|---|---|---|---|
| **Ant1.4** | 0.750 | **0.763** | 0.740 | 0.747 |
| **Ant1.5** | **0.831** | **0.831** | 0.823 | **0.831** |
| **Ant1.6** | **0.863** | **0.863** | 0.859 | 0.861 |
| **Ant1.7** | 0.886 | **0.888** | 0.881 | 0.881 |
| **Jedit4.0** | 0.900 | **0.901** | 0.883 | 0.892 |
| **Jedit4.1** | 0.913 | **0.925** | 0.904 | **0.925** |
| **Synapse1.1** | 0.694 | 0.700 | 0.697 | **0.702** |
| **Synapse1.2** | 0.760 | 0.764 | 0.758 | **0.770** |
| **Xalan2.5.0** | 0.716 | 0.717 | 0.720 | **0.721** |
| **Xalan2.6.0** | 0.864 | **0.866** | 0.865 | 0.864 |
| **Xalan2.7.0** | 0.972 | **0.983** | 0.969 | 0.960 |
| **Xerces1.2.0** | 0.683 | **0.739** | 0.733 | 0.724 |
| **Xerces1.3.0** | 0.825 | **0.845** | 0.814 | 0.842 |
| **Xerces1.4.4** | 0.808 | **0.837** | 0.794 | 0.801 |

### 5.2 Logical Regression Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using logistic regression are presented in this section. Table 5.5 shows the experimental results of all the three models implemented using the logistic regression classification techniques. Table 5.5 consists of AUC values. This table shows that in 86.66% cases static code metrics give better AUC values, in 6.66% case process metric, and in the remaining 6.66% cases combined model of both process and static code metric gives better AUC values.

**Table 5.5 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using LR**

| Dataset Name | Combined | Static Code Metric | Process Metric |
|---|---|---|---|
| **Ant1.4** | 0.631 | **0.751** | 0.450 |
| **Ant1.5** | 0.635 | **0.844** | 0.596 |
| **Ant1.6** | 0.725 | **0.848** | 0.589 |
| **Ant1.7** | 0.726 | **0.856** | 0.698 |
| **Jedit4.0** | 0.694 | **0.819** | 0.687 |
| **Jedit4.1** | 0.612 | **0.856** | 0.212 |
| **Synapse1.1** | 0.689 | **0.717** | 0.493 |
| **Synapse1.2** | 0.461 | **0.737** | 0.329 |
| **Xalan2.5.0** | 0.731 | **0.733** | 0.677 |
| **Xalan2.6.0** | 0.833 | **0.844** | 0.771 |
| **Xalan2.7.0** | 0.982 | 0.956 | **0.989** |
| **Xerces1.2.0** | **0.604** | 0.564 | 0.523 |
| **Xerces1.3.0** | 0.604 | **0.751** | 0.552 |
| **Xerces1.4.4** | 0.853 | **0.867** | 0.764 |

### 5.3 Support Vector Machine Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using a support vector machine are presented in this section. Table 5.6 shows the experimental results of all the three models implemented using the support vector machine classification technique. Table 5.6 consists of AUC values. This table shows that in 66.67% cases combination of both metrics gives better AUC values, in 33.33% cases process metric gives better AUC values and in 13.33% cases static code metric gives better AUC values. As Support Vector Machine gives better prediction performance in combined model so we will also analyze model 4, model 5 and model 6 using Support Vector Machine technique.

**Table 5.6 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using NB**

| Dataset Name | Combined | Static Code Metric | Process Metric |
|---|---|---|---|
| **Ant1.4** | 0.579 | 0.500 | **0.589** |
| **Ant1.5** | **0.616** | 0.500 | 0.553 |
| **Ant1.6** | **0.676** | 0.606 | 0.647 |
| **Ant1.7** | **0.694** | 0.620 | 0.669 |
| **Jedit4.0** | 0.735 | 0.538 | **0.748** |
| **Jedit4.1** | **0.740** | 0.609 | 0.723 |
| **Synapse1.1** | **0.671** | 0.635 | 0.499 |
| **Synapse1.2** | 0.601 | **0.628** | 0.546 |
| **Xalan2.5.0** | **0.651** | 0.575 | 0.629 |
| **Xalan2.6.0** | **0.784** | 0.695 | 0.684 |
| **Xalan2.7.0** | **0.981** | 0.843 | **0.981** |
| **Xerces1.2.0** | 0.540 | 0.500 | **0.553** |
| **Xerces1.3.0** | **0.594** | 0.519 | 0.548 |
| **Xerces1.4.4** | **0.849** | 0.766 | 0.679 |

Table 5.7 shows the experimental results of model 4 implemented using support vector machine technique. This table shows that NR metric is more useful in predicting fault proneness as in 57.14% of cases combination of NR metrics with static code metrics gives better AUC values.

**Table 5.7 AUC values of models which contain combination of SC and 1 process metric using SVM**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.595** | 0.543 | 0.575 | 0.527 |
| **Ant1.5** | 0.540 | 0.500 | **0.556** | 0.500 |
| **Ant1.6** | 0.634 | 0.606 | **0.650** | 0.617 |
| **Ant1.7** | 0.656 | 0.636 | **0.666** | 0.617 |
| **Jedit4.0** | **0.751** | 0.569 | 0.633 | 0.558 |
| **Jedit4.1** | **0.707** | 0.662 | 0.701 | 0.666 |
| **Synapse1.1** | **0.674** | 0.655 | 0.662 | 0.627 |

| | | | | |
|---|---|---|---|---|
| **Synapse1.2** | 0.640 | **0.645** | 0.601 | 0.622 |
| **Xalan2.5.0** | **0.649** | 0.627 | 0.642 | 0.581 |
| **Xalan2.6.0** | 0.774 | 0.728 | **0.777** | 0.719 |
| **Xalan2.7.0** | **0.981** | 0.881 | 0.883 | 0.850 |
| **Xerces1.2.0** | 0.493 | 0.500 | **0.539** | 0.498 |
| **Xerces1.3.0** | **0.576** | 0.527 | 0.572 | 0.511 |
| **Xerces1.4.4** | **0.847** | 0.846 | 0.845 | 0.767 |

Table 5.8 shows the experimental results of model 5 implemented using support vector machine technique. This table shows that NR metric combining with NML metric gives better prediction performance as in 50% of cases combination of static code metrics, NR metrics and NML metric gives better AUC values.

**Table 5.8 AUC values of models which contain combination of SC and 2 process metrics using SVM**

| **Dataset Name** | **NR+NDC+SC** | **NR+NML+SC** | **NR+NDPV+SC** | **NDC+NML+SC** | **NDC+NDPV+SC** | **NML+NDPV+SC** |
|---|---|---|---|---|---|---|
| **Ant1.4** | 0.608 | 0.601 | **0.610** | 0.558 | 0.518 | 0.558 |
| **Ant1.5** | 0.540 | **0.616** | 0.540 | 0.587 | 0.500 | 0.554 |
| **Ant1.6** | 0.637 | **0.683** | 0.636 | 0.653 | 0.612 | 0.650 |
| **Ant1.7** | 0.662 | **0.693** | 0.663 | 0.687 | 0.646 | 0.680 |
| **Jedit4.0** | **0.751** | 0.735 | **0.751** | 0.640 | 0.583 | 0.638 |
| **Jedit4.1** | 0.711 | **0.726** | 0.715 | 0.687 | 0.695 | 0.694 |
| **Synapse1.1** | **0.677** | 0.663 | 0.654 | 0.659 | 0.655 | 0.665 |
| **Synapse1.2** | **0.648** | 0.595 | 0.645 | 0.610 | 0.645 | 0.601 |
| **Xalan2.5.0** | **0.648** | 0.654 | 0.656 | 0.634 | 0.627 | 0.636 |
| **Xalan2.6.0** | 0.770 | 0.789 | 0.773 | **0.782** | 0.740 | 0.778 |
| **Xalan2.7.0** | **0.981** | **0.981** | **0.981** | 0.883 | 0.882 | 0.883 |
| **Xerces1.2.0** | 0.493 | 0.529 | 0.507 | 0.523 | 0.498 | **0.537** |
| **Xerces1.3.0** | 0.591 | **0.601** | 0.569 | 0.566 | 0.527 | 0.566 |
| **Xerces1.4.4** | 0.846 | **0.851** | 0.847 | 0.846 | 0.849 | 0.845 |

Table 5.9 shows the experimental results of model 6 implemented using support vector machine technique. This table shows that NR metric combining with NML metric and NDPV metric gives better prediction performance as in 64.28% of cases combination of static code metrics, NR metrics, NML metric and NDPV metric gives better AUC values.

**Table 5.9 AUC values of models which contain combination of SC and 3 process metrics using SVM**

| Dataset Name | NR+NDC+NML+SC | NR+NDC+NDPV+SC | NR+NML+NDPV+SC | NDC+NML+NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | 0.570 | **0.592** | 0.591 | 0.556 |
| **Ant1.5** | 0.600 | 0.540 | **0.601** | 0.571 |
| **Ant1.6** | 0.677 | 0.637 | **0.682** | 0.652 |
| **Ant1.7** | 0.690 | 0.666 | **0.693** | 0.691 |
| **Jedit4.0** | 0.735 | **0.751** | 0.735 | 0.638 |
| **Jedit4.1** | 0.734 | 0.720 | **0.747** | 0.694 |
| **Synapse1.1** | **0.679** | 0.669 | 0.668 | 0.651 |
| **Synapse1.2** | 0.610 | **0.648** | 0.604 | 0.610 |
| **Xalan2.5.0** | 0.646 | 0.655 | **0.656** | 0.643 |
| **Xalan2.6.0** | 0.784 | 0.774 | **0.786** | 0.779 |
| **Xalan2.7.0** | **0.981** | **0.981** | **0.981** | 0.883 |
| **Xerces1.2.0** | 0.516 | 0.500 | **0.544** | 0.530 |
| **Xerces1.3.0** | **0.600** | 0.591 | 0.594 | 0.552 |
| **Xerces1.4.4** | 0.849 | 0.846 | **0.851** | 0.846 |

### 5.4 K Nearest Neighbor Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using the K nearest neighbor is presented in this section. Table 5.10 shows the experimental results of all the three models implemented using K Nearest Neighbor technique. Table 5.10 consists of AUC values. This table shows that in 60% of cases process metric gives better AUC values, combined model of both metrics gives better AUC value in 20% of cases and model that solely contain static code metrics also give better AUC values in 20% of cases.

**Table 5.10 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using KNN**

| Dataset Name | Combined | Static Code Metric | Process Metric |
|---|---|---|---|
| **Ant1.4** | 0.657 | 0.651 | **0.762** |
| **Ant1.5** | 0.665 | 0.647 | **0.728** |
| **Ant1.6** | 0.691 | 0.728 | **0.778** |
| **Ant1.7** | 0.719 | 0.719 | **0.785** |
| **Jedit4.0** | 0.753 | 0.774 | **0.868** |
| **Jedit4.1** | 0.746 | 0.710 | **0.820** |
| **Synapse1.1** | 0.621 | **0.695** | 0.428 |
| **Synapse1.2** | 0.699 | **0.719** | 0.630 |
| **Xalan2.5.0** | 0.705 | 0.708 | **0.733** |
| **Xalan2.6.0** | **0.817** | 0.801 | 0.811 |
| **Xalan2.7.0** | **0.991** | 0.924 | 0.990 |
| **Xerces1.2.0** | 0.700 | 0.694 | **0.714** |
| **Xerces1.3.0** | 0.679 | **0.759** | 0.614 |
| **Xerces1.4.4** | **0.893** | 0.847 | 0.812 |

### 5.5 Decision Tree Analysis

The results of analyzing all three models to check the relationship between fault proneness and process metrics using a decision tree is presented in this section. Table 5.11 shows the experimental results of all the three models implemented using the decision tree classification technique. Table 5.11 consists of AUC values. This table shows that in 80% of cases combination of both metrics gives better AUC values, in 20% cases process metric gives better AUC values and in 13.33% of cases static code metric gives better AUC values.

**Table 5.11 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using DT**

| Dataset Name | Combined | Static Code Metric | Process Metric |
|---|---|---|---|
| **Ant1.4** | **0.676** | 0.580 | 0.629 |
| **Ant1.5** | **0.679** | 0.646 | 0.535 |

| | | | |
|---|---|---|---|
| **Ant1.6** | **0.813** | 0.809 | 0.771 |
| **Ant1.7** | **0.846** | 0.810 | 0.814 |
| **Jedit4.0** | 0.817 | 0.717 | **0.871** |
| **Jedit4.1** | **0.873** | 0.783 | 0.790 |
| **Synapse1.1** | **0.683** | 0.630 | 0.494 |
| **Synapse1.2** | 0.695 | **0.701** | 0.612 |
| **Xalan2.5.0** | **0.797** | 0.734 | 0.717 |
| **Xalan2.6.0** | **0.872** | 0.848 | 0.848 |
| **Xalan2.7.0** | 0.987 | 0.914 | **0.989** |
| **Xerces1.2.0** | **0.732** | 0.659 | 0.673 |
| **Xerces1.3.0** | **0.763** | 0.712 | 0.603 |
| **Xerces1.4.4** | **0.909** | 0.893 | 0.795 |

Table 5.12 shows the experimental results of model 5 implemented using decision tree technique. This table shows that NR metric is more useful in predicting fault proneness as in 35.71% of cases combination of NR metrics with static code metrics gives better AUC values.

**Table 5.12 AUC values of models which contain combination of SC and 1 process metric using DT**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.654** | 0.601 | 0.572 | 0.549 |
| **Ant1.5** | 0.644 | **0.734** | 0.603 | 0.700 |
| **Ant1.6** | 0.779 | 0.800 | 0.794 | **0.801** |
| **Ant1.7** | 0.778 | 0.802 | 0.798 | **0.828** |
| **Jedit4.0** | **0.856** | 0.836 | 0.845 | 0.761 |
| **Jedit4.1** | 0.786 | **0.872** | 0.847 | 0.768 |
| **Synapse1.1** | 0.641 | 0.643 | 0.681 | **0.694** |
| **Synapse1.2** | 0.698 | 0.691 | 0.723 | **0.730** |
| **Xalan2.5.0** | 0.767 | 0.765 | **0.779** | 0.722 |
| **Xalan2.6.0** | **0.868** | 0.865 | 0.862 | 0.837 |
| **Xalan2.7.0** | **0.987** | 0.961 | 0.958 | 0.947 |
| **Xerces1.2.0** | 0.657 | **0.672** | 0.635 | 0.658 |

| | | | | |
|---|---|---|---|---|
| **Xerces1.3.0** | 0.710 | 0.693 | **0.757** | 0.689 |
| **Xerces1.4.4** | **0.939** | 0.902 | 0.892 | 0.863 |

Table 5.13 shows the experimental results of model 5 implemented using decision tree technique. This table shows that NR metric combining with NDC metric gives better prediction performance as in 28.57% of cases combination of static code metrics, NR metrics and NDC metric gives better AUC values.

**Table 5.13 AUC values of models which contain combination of SC and 2 process metrics using DT**

| Dataset Name | NR+NDC+SC | NR+NML+SC | NR+NDPV+SC | NDC+NML+SC | NDC+NDPV+SC | NML+NDPV+SC |
|---|---|---|---|---|---|---|
| **Ant1.4** | **0.763** | 0.709 | 0.644 | 0.715 | 0.590 | 0.541 |
| **Ant1.5** | 0.713 | 0.537 | **0.728** | 0.704 | 0.703 | 0.715 |
| **Ant1.6** | **0.832** | 0.828 | 0.800 | 0.819 | 0.790 | 0.798 |
| **Ant1.7** | 0.830 | 0.826 | 0.804 | **0.841** | 0.827 | 0.823 |
| **Jedit4.0** | 0.843 | 0.869 | 0.823 | 0.816 | **0.872** | 0.801 |
| **Jedit4.1** | 0.862 | 0.868 | 0.811 | 0.863 | **0.897** | 0.824 |
| **Synapse1.1** | 0.633 | **0.722** | 0.624 | 0.710 | 0.693 | 0.695 |
| **Synapse1.2** | 0.695 | 0.719 | 0.704 | 0.725 | 0.715 | **0.740** |
| **Xalan2.5.0** | 0.768 | **0.794** | 0.772 | 0.787 | 0.780 | 0.774 |
| **Xalan2.6.0** | 0.857 | 0.855 | 0.849 | 0.862 | **0.867** | 0.860 |
| **Xalan2.7.0** | 0.990 | **0.993** | 0.992 | 0.961 | 0.965 | 0.961 |
| **Xerces1.2.0** | 0.658 | 0.668 | **0.718** | 0.688 | 0.694 | 0.673 |
| **Xerces1.3.0** | **0.787** | 0.697 | 0.769 | 0.772 | 0.763 | 0.756 |
| **Xerces1.4.4** | **0.947** | 0.917 | 0.931 | 0.896 | 0.923 | 0.879 |

Table 5.14 shows the experimental results of model 6 implemented using decision tree technique. This table shows that NR metric combining with NML metric and NDPV metric gives better prediction performance as in 50% of cases combination of static code metrics, NR metrics, NML metric and NDPV metric gives better AUC values.

**Table 5.14 AUC values of models which contain combination of SC and 3 process metric using DT**

| Dataset Name | NR+NDC+NML+SC | NR+NDC+NDPV+SC | NR+NML+NDPV+SC | NDC+NML+NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.690** | 0.597 | 0.603 | 0.543 |
| **Ant1.5** | 0.600 | **0.695** | 0.658 | 0.659 |
| **Ant1.6** | 0.810 | 0.814 | **0.831** | 0.821 |
| **Ant1.7** | 0.811 | **0.832** | 0.784 | 0.813 |
| **Jedit4.0** | 0.816 | 0.833 | **0.858** | 0.828 |
| **Jedit4.1** | **0.862** | 0.803 | 0.811 | 0.847 |
| **Synapse1.1** | 0.673 | 0.641 | **0.683** | 0.682 |
| **Synapse1.2** | **0.743** | 0.716 | 0.727 | 0.693 |
| **Xalan2.5.0** | **0.785** | 0.769 | 0.768 | 0.764 |
| **Xalan2.6.0** | 0.848 | 0.854 | **0.873** | 0.868 |
| **Xalan2.7.0** | 0.989 | 0.985 | **0.992** | 0.951 |
| **Xerces1.2.0** | 0.718 | 0.718 | **0.728** | 0.722 |
| **Xerces1.3.0** | 0.718 | 0.737 | **0.740** | 0.731 |
| **Xerces1.4.4** | 0.933 | **0.941** | 0.933 | 0.890 |

### 5.6. Stacking Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using naive bayes are presented in this section. Table 5.15 shows the experimental results of all the three models implemented using stacking ensemble technique. Table 5.15 consists of AUC values. This table shows that in 71.42 % of cases combination of both metrics gives better AUC values, and in 28.57% cases static code metric gives better AUC values.

**Table 5.15 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using stacking**

| Dataset Name | Combined | Static Code Metrics | Process Metrics |
|---|---|---|---|
| **Ant1.4** | **0.767** | 0.716 | 0.760 |
| **Ant1.5** | 0.772 | **0.802** | 0.706 |
| **Ant1.6** | **0.864** | 0.857 | 0.850 |

| | | | |
|---|---|---|---|
| **Ant1.7** | **0.872** | 0.850 | 0.856 |
| **Jedit4.0** | **0.894** | 0.821 | 0.888 |
| **Jedit4.1** | **0.913** | 0.815 | 0.911 |
| **Synapse1.1** | 0.711 | **0.752** | 0.453 |
| **Synapse1.2** | 0.788 | **0.789** | 0.664 |
| **Xalan2.5.0** | **0.797** | 0.792 | 0.733 |
| **Xalan2.6.0** | **0.898** | 0.867 | 0.824 |
| **Xalan2.7.0** | **0.993** | 0.968 | 0.991 |
| **Xerces1.2.0** | **0.797** | 0.725 | 0.777 |
| **Xerces1.3.0** | 0.756 | **0.777** | 0.719 |
| **Xerces1.4.4** | **0.942** | 0.918 | 0.817 |

Table 5.16 shows the experimental results of model 4 implemented using stacking technique. This table shows that NR metric is more useful in predicting fault proneness as in 57.14% of cases combination of NR metrics with static code metrics gives better AUC values.

**Table 5.16 AUC values of models which contain combination of SC and 1 process metric using stacking**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.797** | 0.749 | 0.649 | 0.724 |
| **Ant1.5** | 0.786 | **0.843** | 0.786 | 0.787 |
| **Ant1.6** | **0.881** | 0.873 | 0.834 | 0.853 |
| **Ant1.7** | 0.882 | **0.888** | 0.824 | 0.864 |
| **Jedit4.0** | 0.861 | **0.896** | 0.872 | 0.837 |
| **Jedit4.1** | 0.903 | **0.904** | 0.809 | 0.851 |
| **Synapse1.1** | 0.737 | 0.733 | 0.711 | **0.749** |
| **Synapse1.2** | 0.777 | **0.809** | 0.779 | 0.780 |
| **Xalan2.5.0** | **0.821** | 0.805 | 0.801 | 0.795 |
| **Xalan2.6.0** | **0.900** | 0.885 | 0.891 | 0.880 |
| **Xalan2.7.0** | **0.994** | 0.981 | 0.979 | 0.970 |
| **Xerces1.2.0** | **0.787** | 0.759 | 0.692 | 0.782 |
| **Xerces1.3.0** | **0.859** | 0.849 | 0.769 | 0.794 |

| | | | | |
|---|---|---|---|---|
| **Xerces1.4.4** | **0.950** | 0.934 | 0.938 | 0.913 |

Table 5.17 shows the experimental results of model 5 implemented using stacking technique. This table shows that NDC metric combining with NDPV metric gives better prediction performance as in 50% of cases combination of NDC metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.17 AUC values of models which contain combination of SC and 2 process metrics using stacking**

| Dataset Name | NR+NDC+SC | NR+NML+SC | NR+NDPV+SC | NDC+NML+SC | NDC+NDPV+SC | NML+NDPV+SC |
|---|---|---|---|---|---|---|
| **Ant1.4** | 0.743 | 0.750 | 0.778 | 0.749 | **0.812** | 0.662 |
| **Ant1.5** | 0.783 | 0.773 | 0.782 | 0.775 | **0.795** | 0.769 |
| **Ant1.6** | **0.885** | 0.857 | 0.878 | 0.825 | 0.863 | 0.796 |
| **Ant1.7** | 0.883 | 0.859 | 0.886 | 0.824 | **0.891** | 0.823 |
| **Jedit4.0** | 0.835 | 0.864 | 0.846 | 0.866 | **0.887** | 0.839 |
| **Jedit4.1** | 0.905 | 0.911 | 0.916 | 0.898 | **0.918** | 0.844 |
| **Synapse1.1** | 0.729 | 0.704 | 0.707 | 0.726 | **0.742** | 0.730 |
| **Synapse1.2** | 0.767 | 0.789 | 0.778 | 0.756 | 0.817 | **0.819** |
| **Xalan2.5.0** | **0.825** | 0.799 | 0.823 | 0.808 | 0.824 | 0.813 |
| **Xalan2.6.0** | **0.904** | 0.901 | **0.904** | 0.898 | 0.887 | 0.894 |
| **Xalan2.7.0** | 0.991 | **0.993** | 0.992 | 0.983 | 0.982 | 0.980 |
| **Xerces1.2.0** | 0.791 | 0.750 | 0.793 | 0.761 | **0.835** | 0.727 |
| **Xerces1.3.0** | **0.840** | 0.784 | 0.830 | 0.781 | 0.828 | 0.750 |
| **Xerces1.4.4** | 0.945 | 0.945 | **0.947** | 0.935 | 0.933 | 0.937 |

Table 5.18 shows the experimental results of model 6 implemented using stacking technique. This table shows that NR metric combining with NDC metric and NDPV metric gives better prediction performance as in 78.57% of cases combination of NR metric, NDC metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.18 AUC values of models which contain combination of SC and 3 process metric using stacking**

| Dataset Name | SC+NR+NDC +NML | SC+NR+NDC +NDPV | SC+NR+NML+ NDPV | SC+NDC+NML +NDPV |
|---|---|---|---|---|
| **Ant1.4** | 0.786 | **0.812** | 0.796 | 0.677 |
| **Ant1.5** | 0.809 | **0.820** | 0.774 | 0.811 |
| **Ant1.6** | 0.863 | **0.886** | 0.849 | 0.845 |
| **Ant1.7** | 0.853 | **0.890** | 0.867 | 0.835 |
| **Jedit4.0** | 0.843 | **0.898** | 0.846 | 0.844 |
| **Jedit4.1** | 0.902 | 0.912 | **0.914** | 0.865 |
| **Synapse1.1** | 0.731 | 0.732 | 0.694 | **0.737** |
| **Synapse1.2** | 0.753 | **0.772** | 0.766 | 0.762 |
| **Xalan2.5.0** | 0.802 | **0.822** | 0.806 | 0.792 |
| **Xalan2.6.0** | 0.899 | **0.900** | **0.900** | 0.894 |
| **Xalan2.7.0** | 0.993 | 0.993 | **0.995** | 0.981 |
| **Xerces1.2.0** | 0.738 | **0.811** | 0.764 | 0.780 |
| **Xerces1.3.0** | 0.754 | **0.844** | 0.775 | 0.768 |
| **Xerces1.4.4** | 0.949 | **0.950** | 0.948 | 0.938 |

### 5.7 Voting Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using naive bayes are presented in this section. Table 5.19 shows the experimental results of all the three models implemented using stacking ensemble technique. Table 5.19 consists of AUC values. This table shows that in 71.42% of cases combination of both metrics gives better AUC values, in 14.28% of cases process metrics gives better AUC values, and in 14.28% cases static code metric gives better AUC values.

**Table 5.19 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using voting**

| Dataset Name | Combined | Static Code Metrics | Process Metrics |
|---|---|---|---|
| **Ant1.4** | 0.806 | 0.760 | **0.811** |
| **Ant1.5** | **0.832** | **0.832** | 0.772 |
| **Ant1.6** | **0.872** | 0.857 | 0.856 |

| | | | |
|---|---|---|---|
| **Ant1.7** | **0.877** | 0.862 | **0.877** |
| **Jedit4.0** | 0.890 | 0.845 | **0.911** |
| **Jedit4.1** | **0.916** | 0.855 | 0.908 |
| **Synapse1.1** | 0.745 | **0.797** | 0.480 |
| **Synapse1.2** | 0.764 | **0.794** | 0.651 |
| **Xalan2.5.0** | **0.792** | 0.759 | 0.735 |
| **Xalan2.6.0** | **0.899** | 0.869 | 0.832 |
| **Xalan2.7.0** | **0.997** | 0.963 | 0.989 |
| **Xerces1.2.0** | **0.812** | 0.741 | 0.797 |
| **Xerces1.3.0** | **0.818** | 0.794 | 0.711 |
| **Xerces1.4.4** | **0.943** | 0.897 | 0.817 |

Table 5.20 shows the experimental results of model 4 implemented using voting technique. This table shows that NR metric is more useful in predicting fault proneness as in 78.57% of cases combination of NR metric with static code metrics gives better AUC values.

**Table 5.20 AUC values of models which contain combination of SC and 1 process metric using voting**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.836** | 0.829 | 0.760 | 0.767 |
| **Ant1.5** | **0.881** | 0.864 | 0.802 | 0.828 |
| **Ant1.6** | **0.888** | 0.884 | 0.849 | 0.856 |
| **Ant1.7** | **0.885** | **0.885** | 0.840 | 0.870 |
| **Jedit4.0** | **0.923** | 0.912 | 0.860 | 0.855 |
| **Jedit4.1** | **0.932** | 0.921 | 0.884 | 0.871 |
| **Synapse1.1** | 0.782 | **0.801** | 0.777 | 0.796 |
| **Synapse1.2** | 0.802 | **0.830** | 0.736 | 0.795 |
| **Xalan2.5.0** | 0.797 | **0.805** | 0.781 | 0.764 |
| **Xalan2.6.0** | **0.897** | 0.878 | 0.888 | 0.882 |
| **Xalan2.7.0** | **0.995** | 0.979 | 0.979 | 0.971 |
| **Xerces1.2.0** | **0.812** | 0.789 | 0.698 | 0.781 |
| **Xerces1.3.0** | **0.871** | 0.843 | 0.779 | 0.810 |

| | | | | |
|---|---|---|---|---|
| **Xerces1.4.4** | **0.954** | 0.939 | 0.937 | 0.902 |

Table 5.21 shows the experimental results of model 5 implemented using voting technique. This table shows that NR metric combining with NDC metric gives better prediction performance as in 50% of cases combination of static code metrics, NR metrics and NDC metric gives better AUC values.

**Table 5.21 AUC values of models which contain combination of SC and 2 process metrics using voting**

| Dataset Name | NR+NDC+SC | NR+NML+SC | NR+NDPV+SC | NDC+NML+SC | NDC+NDPV+SC | NML+NDPV+SC |
|---|---|---|---|---|---|---|
| **Ant1.4** | **0.846** | 0.802 | 0.840 | 0.777 | 0.818 | 0.762 |
| **Ant1.5** | **0.874** | 0.847 | 0.872 | 0.817 | 0.858 | 0.799 |
| **Ant1.6** | **0.892** | 0.856 | 0.886 | 0.859 | 0.880 | 0.853 |
| **Ant1.7** | **0.892** | 0.854 | 0.890 | 0.846 | 0.889 | 0.841 |
| **Jedit4.0** | **0.924** | 0.889 | 0.923 | 0.865 | 0.911 | 0.846 |
| **Jedit4.1** | 0.934 | 0.911 | 0.929 | 0.894 | **0.937** | 0.875 |
| **Synapse1.1** | 0.785 | 0.744 | 0.783 | 0.757 | **0.803** | 0.776 |
| **Synapse1.2** | 0.807 | 0.760 | 0.807 | 0.753 | **0.830** | 0.740 |
| **Xalan2.5.0** | 0.805 | 0.791 | 0.805 | 0.782 | **0.808** | 0.783 |
| **Xalan2.6.0** | **0.902** | 0.898 | 0.899 | 0.890 | 0.887 | 0.889 |
| **Xalan2.7.0** | 0.994 | **0.997** | 0.996 | 0.982 | 0.979 | 0.980 |
| **Xerces1.2.0** | 0.790 | 0.765 | 0.818 | 0.736 | **0.843** | 0.744 |
| **Xerces1.3.0** | 0.865 | 0.798 | **0.868** | 0.780 | 0.843 | 0.786 |
| **Xerces1.4.4** | **0.954** | 0.947 | 0.953 | 0.937 | 0.938 | 0.935 |

Table 5.22 shows the experimental results of model 6 implemented using voting technique. This table shows that NR metric combining with NDC metric and NDPV metric gives better prediction performance as in 92.85% of cases combination of NR metric, NDC metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.22 AUC values of models which contain combination of SC and 3 process metric using voting**

| Dataset Name | SC+NR+NDC+ NML | SC+NR+NDC +NDPV | SC+NR+NML +NDPV | SC+NDC+NML +NDPV |
|---|---|---|---|---|
| **Ant1.4** | 0.809 | **0.840** | 0.813 | 0.776 |
| **Ant1.5** | 0.838 | **0.865** | 0.838 | 0.814 |
| **Ant1.6** | 0.874 | **0.890** | 0.862 | 0.861 |
| **Ant1.7** | 0.878 | **0.891** | 0.860 | 0.845 |
| **Jedit4.0** | 0.894 | **0.924** | 0.885 | 0.853 |
| **Jedit4.1** | 0.916 | **0.943** | 0.912 | 0.895 |
| **Synapse1.1** | 0.753 | **0.787** | 0.741 | 0.756 |
| **Synapse1.2** | 0.765 | **0.809** | 0.764 | 0.758 |
| **Xalan2.5.0** | 0.790 | **0.810** | 0.795 | 0.787 |
| **Xalan2.6.0** | 0.898 | **0.901** | 0.900 | 0.888 |
| **Xalan2.7.0** | 0.997 | 0.995 | 0.997 | **0.982** |
| **Xerces1.2.0** | 0.750 | **0.840** | 0.801 | 0.782 |
| **Xerces1.3.0** | 0.798 | **0.863** | 0.808 | 0.794 |
| **Xerces1.4.4** | 0.949 | **0.951** | 0.945 | 0.936 |

### 5.8 Bagging Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using naive bayes are presented in this section. Table 5.23 shows the experimental results of all the three models implemented using stacking ensemble technique. Table 5.23 consists of AUC values. This table shows that in 64.28% of cases combination of both metrics gives better AUC values, in 14.28% of cases process metrics gives better AUC values, and in 21.42% cases static code metric gives better AUC values.

**Table 5.23 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using bagging**

| Dataset Name | Combined | Static Code Metrics | Process Metrics |
|---|---|---|---|
| **Ant1.4** | 0.808 | 0.769 | **0.817** |
| **Ant1.5** | 0.850 | **0.858** | 0.784 |
| **Ant1.6** | **0.894** | 0.868 | 0.859 |

| | | | |
|---|---|---|---|
| **Ant1.7** | **0.900** | 0.875 | 0.877 |
| **Jedit4.0** | 0.917 | 0.850 | **0.920** |
| **Jedit4.1** | **0.938** | 0.881 | 0.906 |
| **Synapse1.1** | 0.764 | **0.790** | 0.494 |
| **Synapse1.2** | 0.794 | **0.811** | 0.660 |
| **Xalan2.5.0** | **0.803** | 0.779 | 0.740 |
| **Xalan2.6.0** | **0.906** | 0.876 | 0.834 |
| **Xalan2.7.0** | **0.997** | 0.967 | 0.992 |
| **Xerces1.2.0** | **0.814** | 0.759 | 0.804 |
| **Xerces1.3.0** | **0.843** | 0.831 | 0.731 |
| **Xerces1.4.4** | **0.959** | 0.905 | 0.823 |

Table 5.24 shows the experimental results of model 4 implemented using bagging technique. This table shows that NR metric is more useful in predicting fault proneness as in 78.57% of cases combination of NR metric with static code metrics gives better AUC values.

**Table 5.24 AUC values of models which contain combination of SC and 1 process metric using bagging**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | **0.840** | 0.815 | 0.799 | 0.768 |
| **Ant1.5** | **0.882** | 0.869 | 0.841 | 0.851 |
| **Ant1.6** | **0.896** | 0.891 | 0.868 | 0.867 |
| **Ant1.7** | **0.892** | **0.892** | 0.883 | 0.881 |
| **Jedit4.0** | **0.918** | 0.909 | 0.891 | 0.858 |
| **Jedit4.1** | **0.942** | 0.931 | 0.913 | 0.900 |
| **Synapse1.1** | 0.779 | **0.785** | 0.775 | 0.784 |
| **Synapse1.2** | 0.802 | **0.819** | 0.794 | 0.808 |
| **Xalan2.5.0** | 0.806 | **0.808** | 0.789 | 0.785 |
| **Xalan2.6.0** | **0.904** | 0.890 | 0.899 | 0.887 |
| **Xalan2.7.0** | **0.996** | 0.979 | 0.980 | 0.971 |
| **Xerces1.2.0** | 0.800 | 0.791 | 0.771 | **0.807** |
| **Xerces1.3.0** | **0.870** | 0.868 | 0.827 | 0.839 |

| | | | | |
|---|---|---|---|---|
| **Xerces1.4.4** | **0.956** | 0.943 | 0.947 | 0.911 |

Table 5.25 shows the experimental results of model 5 implemented using bagging technique. This table shows that NR metric combining with NDC metric gives better prediction performance as in 57.14% of cases combination of static code metrics, NR metrics and NDC metric gives better AUC values.

**Table 5.25 AUC values of models which contain combination of SC and 2 process metrics using bagging**

| Dataset Name | NR+NDC+SC | NR+NML+SC | NR+NDPV+SC | NDC+NML+SC | NDC+NDPV+SC | NML+NDPV+SC |
|---|---|---|---|---|---|---|
| **Ant1.4** | **0.844** | 0.813 | 0.833 | 0.810 | 0.806 | 0.791 |
| **Ant1.5** | **0.870** | 0.855 | 0.866 | 0.862 | 0.864 | 0.840 |
| **Ant1.6** | **0.895** | 0.893 | **0.895** | 0.874 | 0.888 | 0.869 |
| **Ant1.7** | **0.896** | 0.893 | 0.895 | 0.884 | **0.896** | 0.887 |
| **Jedit4.0** | **0.923** | 0.915 | 0.919 | 0.901 | 0.909 | 0.894 |
| **Jedit4.1** | **0.946** | 0.936 | 0.945 | 0.926 | 0.940 | 0.918 |
| **Synapse1.1** | 0.776 | 0.766 | 0.776 | 0.770 | **0.778** | 0.775 |
| **Synapse1.2** | 0.806 | 0.791 | 0.804 | 0.785 | **0.821** | 0.792 |
| **Xalan2.5.0** | 0.805 | 0.799 | **0.812** | 0.792 | **0.812** | 0.793 |
| **Xalan2.6.0** | **0.906** | **0.906** | 0.905 | 0.899 | 0.897 | 0.901 |
| **Xalan2.7.0** | 0.996 | 0.997 | **0.998** | 0.982 | 0.980 | 0.981 |
| **Xerces1.2.0** | 0.796 | 0.775 | **0.838** | 0.780 | 0.831 | 0.809 |
| **Xerces1.3.0** | **0.871** | 0.836 | 0.869 | 0.858 | 0.865 | 0.837 |
| **Xerces1.4.4** | 0.955 | **0.959** | 0.954 | 0.945 | 0.943 | 0.946 |

Table 5.26 shows the experimental results of model 6 implemented using bagging technique. This table shows that NR metric combining with NDC metric and NDPV metric gives better prediction performance as in 78.57% of cases combination of NR metric, NDC metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.26 AUC values of models which contain combination of SC and 3 process metrics using bagging**

| Dataset Name | NR+NDC+NML+SC | NR+NDC+NDPV+SC | NR+NML+NDPV+SC | NDC+NML+NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | 0.814 | **0.836** | 0.810 | 0.806 |
| **Ant1.5** | 0.854 | **0.866** | 0.851 | 0.861 |
| **Ant1.6** | 0.893 | **0.898** | 0.890 | 0.871 |
| **Ant1.7** | **0.899** | 0.898 | 0.894 | 0.888 |
| **Jedit4.0** | **0.920** | **0.920** | 0.912 | 0.898 |
| **Jedit4.1** | 0.938 | **0.947** | 0.937 | 0.928 |
| **Synapse1.1** | 0.766 | **0.773** | 0.764 | 0.765 |
| **Synapse1.2** | 0.793 | **0.807** | 0.789 | 0.785 |
| **Xalan2.5.0** | 0.800 | **0.810** | 0.803 | 0.795 |
| **Xalan2.6.0** | **0.907** | 0.905 | 0.906 | 0.900 |
| **Xalan2.7.0** | **0.998** | **0.998** | 0.997 | 0.982 |
| **Xerces1.2.0** | 0.778 | **0.841** | 0.814 | 0.812 |
| **Xerces1.3.0** | 0.841 | **0.867** | 0.835 | 0.859 |
| **Xerces1.4.4** | **0.961** | 0.955 | 0.958 | 0.946 |

### 5.9 Boosting Analysis

The results of analyzing all the three models to check the relationship between fault proneness and process metrics using naive bayes are presented in this section. Table 5.27 shows the experimental results of all the three models implemented using stacking ensemble technique. Table 5.27 consists of AUC values. This table shows that in 71.42% of cases combination of both metrics gives better AUC values, and in 28.57% cases static code metric gives better AUC values.

**Table 5.27 AUC values of combined model, model solely containing static code metrics and model solely containing process metrics using boosting**

| Dataset Name | Combined | Static Code Metrics | Process Metrics |
|---|---|---|---|
| **Ant1.4** | **0.822** | 0.781 | 0.759 |
| **Ant1.5** | 0.835 | **0.852** | 0.763 |
| **Ant1.6** | 0.869 | **0.871** | 0.843 |

| | | | |
|---|---|---|---|
| **Ant1.7** | **0.894** | 0.873 | 0.857 |
| **Jedit4.0** | **0.908** | 0.857 | 0.901 |
| **Jedit4.1** | **0.902** | 0.844 | 0.877 |
| **Synapse1.1** | 0.732 | **0.758** | 0.506 |
| **Synapse1.2** | 0.753 | **0.796** | 0.547 |
| **Xalan2.5.0** | **0.815** | 0.783 | 0.718 |
| **Xalan2.6.0** | **0.899** | 0.875 | 0.830 |
| **Xalan2.7.0** | **0.993** | 0.967 | 0.992 |
| **Xerces1.2.0** | **0.801** | 0.744 | 0.772 |
| **Xerces1.3.0** | **0.819** | 0.801 | 0.657 |
| **Xerces1.4.4** | **0.949** | 0.904 | 0.822 |

Table 5.28 shows the experimental results of model 5.28 implemented using boosting technique. This table shows that NR metric is more useful in predicting fault proneness as in 50% of cases combination of NR metric with static code metrics gives better AUC values.

**Table 5.28 AUC values of models which contain combination of SC and 1 process metrics using boosting**

| Dataset Name | NR+SC | NDC+SC | NML+SC | NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | 0.831 | **0.843** | 0.802 | 0.783 |
| **Ant1.5** | 0.847 | 0.841 | 0.798 | **0.854** |
| **Ant1.6** | **0.894** | 0.885 | 0.865 | 0.863 |
| **Ant1.7** | **0.890** | 0.884 | 0.857 | 0.871 |
| **Jedit4.0** | **0.921** | 0.906 | 0.887 | 0.851 |
| **Jedit4.1** | **0.940** | 0.929 | 0.877 | 0.887 |
| **Synapse1.1** | 0.765 | 0.773 | 0.761 | **0.775** |
| **Synapse1.2** | 0.785 | **0.834** | 0.748 | 0.792 |
| **Xalan2.5.0** | 0.808 | **0.811** | 0.800 | 0.788 |
| **Xalan2.6.0** | **0.900** | 0.881 | 0.892 | 0.880 |
| **Xalan2.7.0** | **0.995** | 0.982 | 0.982 | 0.973 |
| **Xerces1.2.0** | 0.795 | **0.809** | 0.737 | 0.790 |
| **Xerces1.3.0** | 0.822 | **0.855** | 0.780 | 0.825 |

| | | | | |
|---|---|---|---|---|
| **Xerces1.4.4** | **0.954** | 0.941 | 0.939 | 0.901 |

Table 5.29 shows the experimental results of model 5 implemented using boosting technique. This table shows that NDC metric combining with NDPV metric gives better prediction performance as in 50% of cases combination of NDC metric, NDPV metric and static code metrics gives better AUC values. Also, NR metric combining with NDPV metric gives better prediction performance as in 50% of cases combination of NR metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.29 AUC values of models which contain combination of SC and 2 process metrics using boosting**

| Dataset Name | NR+NDC +SC | NR+NM L+SC | NR+ND PV+SC | NDC+NM L+SC | NDC+ND PV+SC | NML+N DPV+SC |
|---|---|---|---|---|---|---|
| **Ant1.4** | 0.824 | 0.814 | 0.825 | 0.799 | **0.835** | 0.795 |
| **Ant1.5** | **0.851** | 0.834 | 0.834 | 0.795 | 0.839 | 0.802 |
| **Ant1.6** | **0.901** | 0.863 | 0.891 | 0.867 | 0.884 | 0.864 |
| **Ant1.7** | 0.889 | 0.876 | **0.896** | 0.856 | 0.883 | 0.869 |
| **Jedit4.0** | 0.920 | 0.898 | **0.922** | 0.890 | 0.914 | 0.878 |
| **Jedit4.1** | 0.934 | 0.889 | **0.936** | 0.892 | 0.926 | 0.876 |
| **Synapse1.1** | 0.759 | 0.731 | 0.755 | 0.749 | **0.776** | 0.746 |
| **Synapse1.2** | 0.815 | 0.745 | 0.793 | 0.754 | **0.839** | 0.747 |
| **Xalan2.5.0** | **0.826** | 0.807 | **0.826** | 0.802 | 0.821 | 0.806 |
| **Xalan2.6.0** | **0.906** | 0.902 | 0.903 | 0.891 | 0.886 | 0.889 |
| **Xalan2.7.0** | 0.996 | **0.998** | 0.996 | 0.982 | 0.982 | 0.981 |
| **Xerces1.2.0** | 0.791 | 0.743 | 0.839 | 0.744 | **0.841** | 0.782 |
| **Xerces1.3.0** | 0.852 | 0.827 | 0.842 | 0.797 | **0.861** | 0.779 |
| **Xerces1.4.4** | 0.954 | 0.952 | **0.955** | 0.937 | 0.942 | 0.934 |

Table 5.30 shows the experimental results of model 6 implemented using boosting technique. This table shows that NR metric combining with NDC metric and NDPV metric gives better prediction performance as in 92.85% of cases combination of NR metric, NDC metric, NDPV metric and static code metrics gives better AUC values.

**Table 5.30 AUC values of models which contain combination of SC and 3 process metrics using boosting**

| Dataset Name | NR+NDC+NML+SC | NR+NDC+NDPV+SC | NR+NML+NDPV+SC | NDC+NML+NDPV+SC |
|---|---|---|---|---|
| **Ant1.4** | 0.819 | **0.823** | 0.813 | 0.802 |
| **Ant1.5** | **0.823** | **0.823** | 0.814 | 0.796 |
| **Ant1.6** | 0.867 | **0.899** | 0.871 | 0.864 |
| **Ant1.7** | 0.877 | **0.901** | 0.884 | 0.865 |
| **Jedit4.0** | 0.906 | **0.922** | 0.905 | 0.880 |
| **Jedit4.1** | 0.895 | **0.937** | 0.898 | 0.891 |
| **Synapse1.1** | 0.730 | **0.760** | 0.727 | 0.754 |
| **Synapse1.2** | 0.765 | **0.813** | 0.758 | 0.748 |
| **Xalan2.5.0** | 0.817 | **0.822** | 0.810 | 0.806 |
| **Xalan2.6.0** | **0.902** | **0.902** | 0.898 | 0.891 |
| **Xalan2.7.0** | 0.994 | **0.996** | **0.996** | 0.983 |
| **Xerces1.2.0** | 0.750 | **0.811** | 0.792 | 0.810 |
| **Xerces1.3.0** | 0.794 | **0.857** | 0.834 | 0.803 |
| **Xerces1.4.4** | **0.953** | 0.952 | **0.953** | 0.934 |

# CHAPTER 6

# DISCUSSION ON RESULTS

We analyzed various models to check the effectiveness of process metrics on fault proneness using various classifiers and ensemble techniques. We also investigated some other models for that classifiers in which combination model gives better prediction performance to check which process metric or which pair of process metric is useful in prediction. The results of the experimental analysis show that the combination model gives better prediction results when we are using Naïve Bayes classifiers, Decision Trees and Support Vector Machines. Also, the result of all ensemble techniques shows that combination model gives better prediction results. NR metric, NDC metric and NDPV metric are most effective process metric in almost all the analyzed cases. In this section, we are analyzing the AUC results using box plots and statistical test. We are also calculating the effect size if there is a rejection of null hypothesis.

## 6.1 Naive Bayes Analysis

If we statistically analyze the results of table 5.1 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 7.429 and the p-value is 0.024. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.265 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain static code metric just.

Fig 6.1 represents the box plot for table 5.1. We can see that combined model of process metrics and static code metrics outperforms the models solely containing process metrics and model solely containing static code metrics.

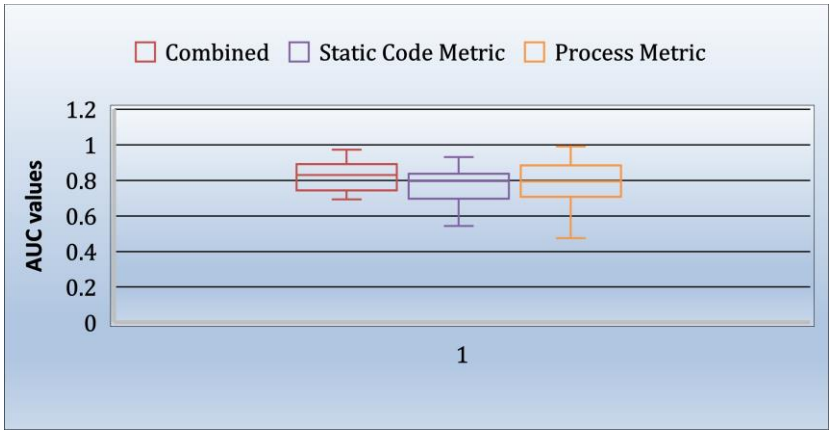


**Fig. 6.1 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using NB**

If we statistically analyze the results of table 5.2 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 19.108 and the p-value is 0.0002. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pair differs significantly with an effect size of 0.455 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NDPV models, also there exist a significant difference between SC+NML model and SC+NDPV model.

Fig 6.2 represents the box plot for table 5.2. We can see that combination of NR metric with statis code metric outperforms other pair of process metric and static code metrics.

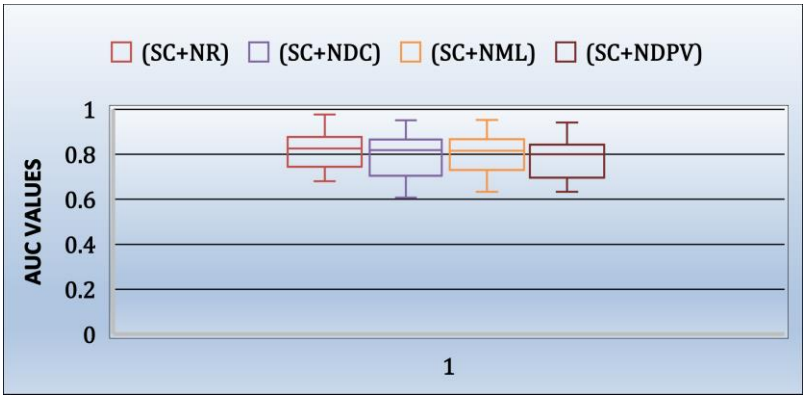


**Fig 6.2 Box plots of models which contain combination of SC and 1 process metric using NB**

If we statistically analyze the results of table 5.3 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 22.37 and the p-value is 0.0004. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.319 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+SC model and NR+NML+SC model, NR+NDC+SC model and NDC+NDPV+SC model, and NR+NDPV+SC model and NDC+NDPV+SC.

Fig 6.3 represents the box plot for table 5.3. We can see that combination of NR metric with NDC metric and statis code metric outperforms other pair of process metric and static code metrics.

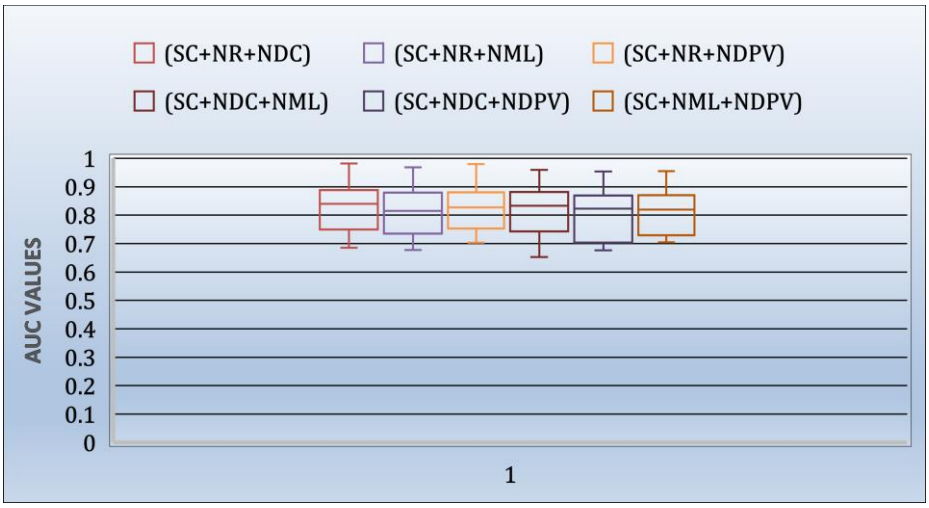


**Fig 6.3 Box plots of models which contain combination of SC and 2 process metrics using NB**

If we statistically analyze the results of table 5.4 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 17.75 and the p-value is 0.0004. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.423 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to

check the pairwise comparison of the models which contains 3 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+NML+SC model and NR+NDC+NDPV+SC model, NR+NDC+NDPV+SC model and NR+NML+NDPV+SC model, and NR+NML+NDPV+SC model and NML+NDC+NDPV+SC model.

Fig 6.4 represents the box plot for table 5.4. We can see that combination of NR metric with NDC metric, NDPV metric and statis code metric outperforms other pair of process metric and static code metrics.

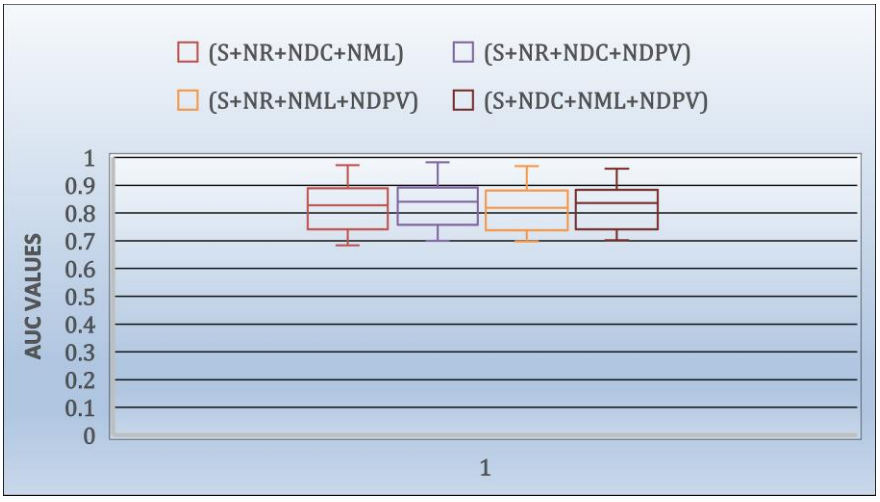


**Fig 6.4 Box plots of models which contain combination of SC and 3 process metrics using NB**

## 6.2 Logistic Regression Analysis

If we statistically analyze the results of table 5.5 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 19 and the p-value is 7.4e-05. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.678 which indicates that the difference among the models has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which solely contain process metric, and model which solely contain static code metric.

Fig 6.5 represents the box plot for table 5.5. We can see that models solely containing static code metrics outperforms models solely containing process metrics and combined model of process metric and static code metrics and there is one outlier in the combination model.

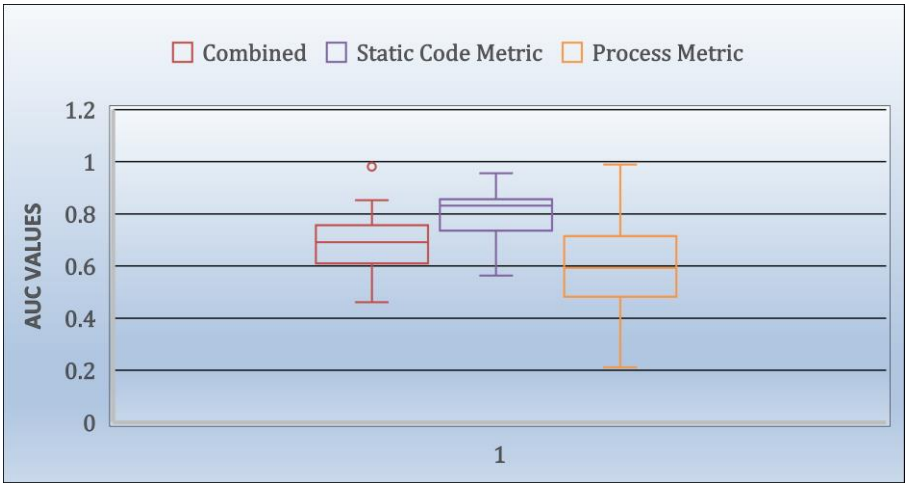


**Fig. 6.5 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using LR**

### 6.3 Support Vector Machine Analysis

If we statistically analyze the results of table 5.6 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 12.47 and the p-value is 0.0019. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.445 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain static code metric just.

Fig 6.6 represents the box plot for table 5.6. We can see that the combined model of process metric and static code metrics outperforms models solely containing process metrics and models solely containing static code metrics and there is one outlier in combined model and model containing just process metric.

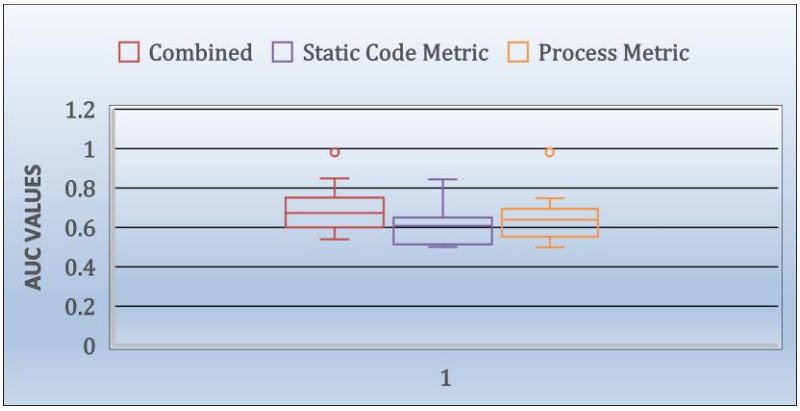


**Fig. 6.6 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using SVM**

If we statistically analyze the results of table 5.7 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 23.84 and the p-value is 2.68e-05. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pair differs significantly with an effect size of 0.567 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NDC model, SC+NR model and SC+NDPV models, and SC+NML model and SC+NDPV model.

Fig 6.7 represents the box plot for table 5.7. We can see that the combination of NR metric with statis code metrics outperforms other pairs of process metric and static code metrics.

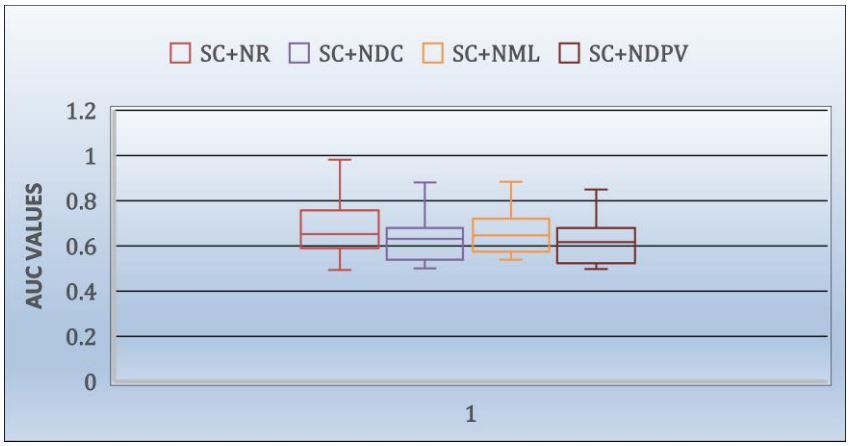


**Fig 6.7 Box plots of models which contain combination of SC and 1 process metric using SVM**

If we statistically analyze the results of table 5.8 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 22.40 and the p-value is 0.0004. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.32 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NML+SC model and NDC+NDPV+SC model, and NR+NDPV+SC model and NDC+NDPV+SC model.

Fig 6.8 represents the box plot for table 5.8. We can see that combination of NR metric with NML metric and statis code metric outperforms other pair of process metric and static code metrics and there is one outlier in the pair of NR metric, NML metric and static code metric.

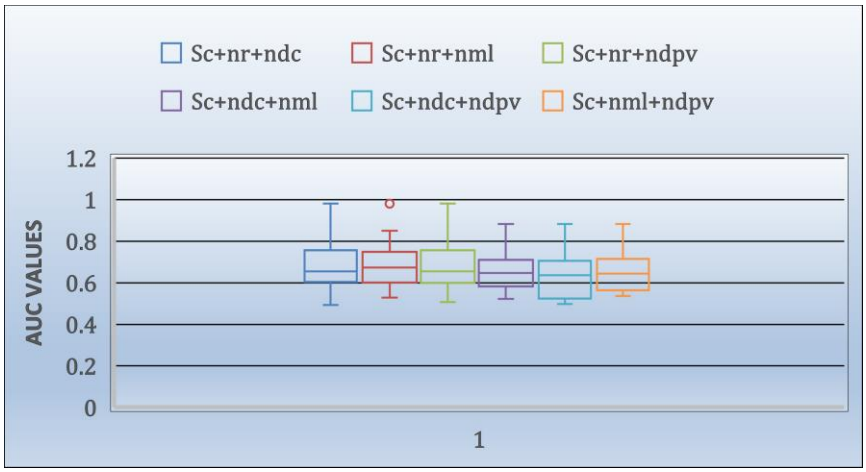


**Fig 6.8 Box plots of models which contain combination of SC and 2 process metrics using SVM**

If we statistically analyze the results of table 5.9 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 13.736 and the p-value is 0.0032. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.327 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 3 process metrics and all static

code metrics. The results show that a significant difference exists between the performances of the NR+NML+NDPV+SC model and NDC+NML+NDPV+SC model.

Fig 6.9 represents the box plot for table 5.9. We can see that combination of NR metric with NML metric, NDPV metric and statis code metric outperforms other pair of process metric and static code metrics and there is one outlier in the pair SC+NR+NDC+NML.

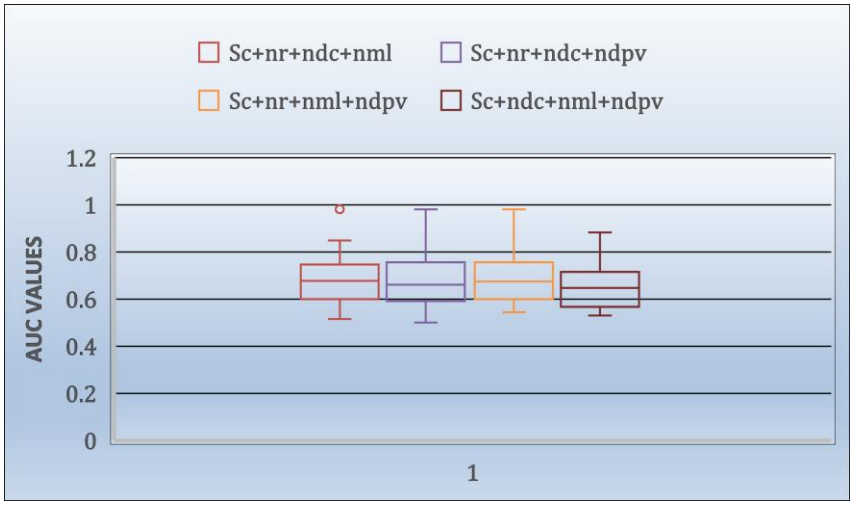


**Fig 6.9 Box plots of models which contain combination of SC and 3 process metrics using SVM**

## 6.4 K Nearest Neighbor Analysis

If we statistically analyze the results of table 5.10 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 2.072 and the p-value is 0.354. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is greater than the calculated $\chi^2$-statistics so the null hypothesis is accepted.

Fig 6.10 represents the box plot for table 5.10. We can see that the models solely containing process metric outperforms models solely containing static code metric and combined model of process metrics and static code metrics. There is one outlier in combined model, one outlier in model that solely contains static code metric and one outlier in model that solely contains process metric.

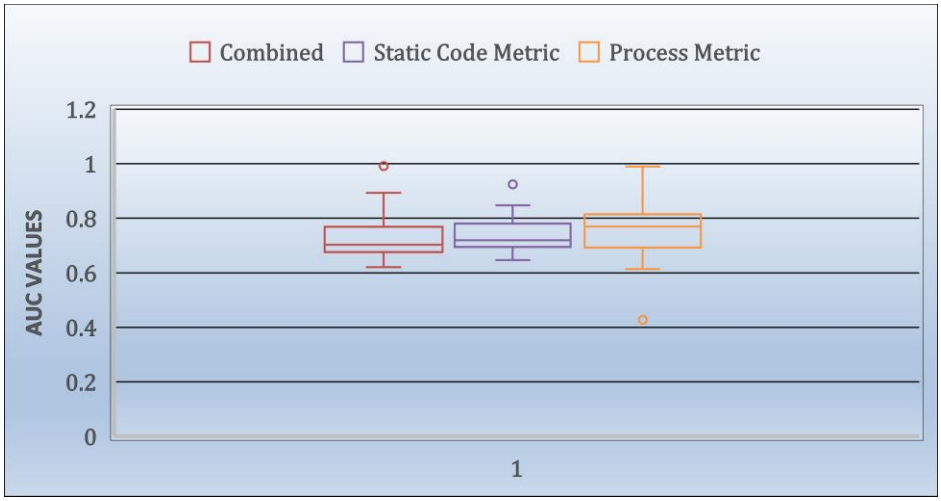


**Fig. 6.10 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using KNN**

## 6.5 Decision Tree Analysis

If we statistically analyze the results of table 5.11 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 13.199and the p-value is 0.0013. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.471 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain static code metric just, and the combined model and model which contain process metric just. Fig 6.11 represents the box plot for table 5.11. We can see that the combined model of process metric and static code metrics outperforms models solely containing process metrics and models solely containing static code metrics.

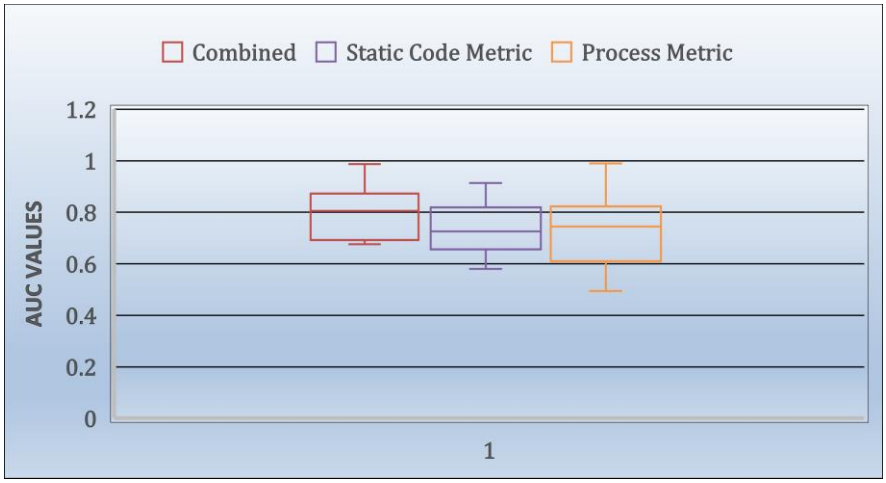


**Fig. 6.11 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using DT**

If we statistically analyze the results of table 5.12 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 1.971 and the p-value is 0.578. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is greater than the calculated $\chi^2$-statistics so null hypothesis is accepted.

Fig 6.12 represents the box plot for table 5.12. We can see that the combination of NR metric with statis code metrics outperforms other pairs of process metric and static code metrics.

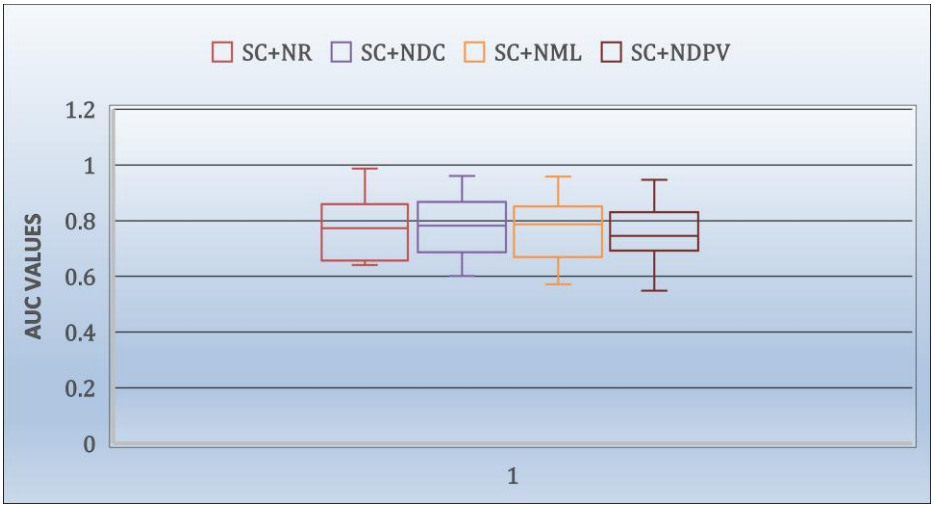


**Fig 6.12 Box plots of models which contain combination of SC and 1 process metric using DT**

If we statistically analyze the results of table 5.13 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 5.286 and the p-value is 0.3819. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is greater than the calculated $\chi^2$-statistics so null hypothesis is accepted.

Fig 6.13 represents the box plot for table 5.13. We can see that combination of NR metric with NDC metric and static code metric outperforms other pair of process metric and static code metrics.

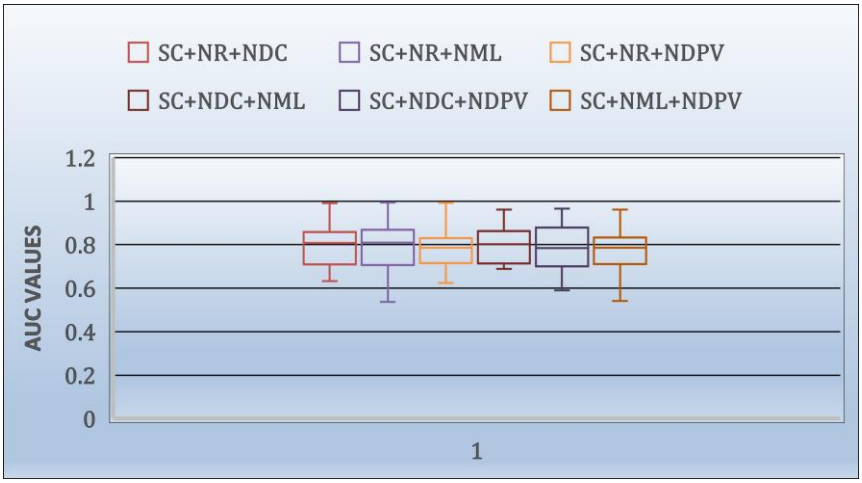


**Fig 6.13 Box plots of models which contain combination of SC and 2 process metrics using DT**

If we statistically analyze the results of table 5.14 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 4.630 and the p-value is 0.2009. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is greater than the calculated $\chi^2$-statistics so the null hypothesis is accepted.

Fig 6.14 represents the box plot for table 5.14. We can see that combination of NR metric with NDC metric, NML metric and statis code metric outperforms other pair of process metric and static code metrics.

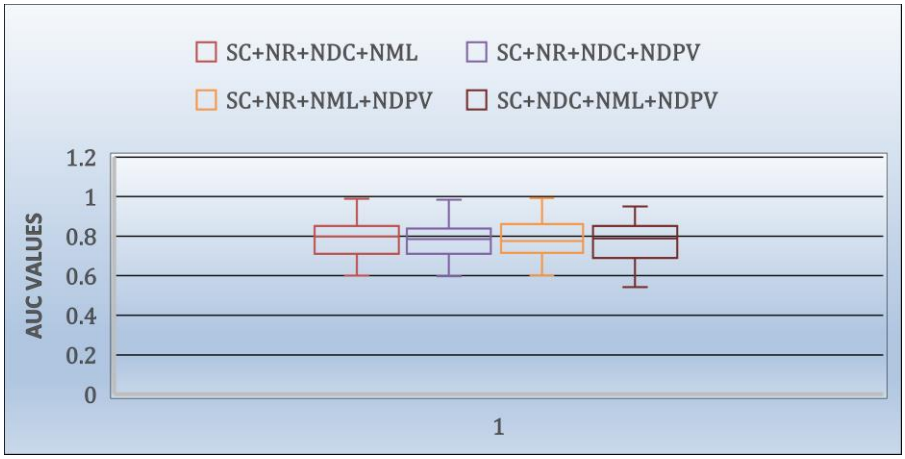


**Fig 6.14 Box plots of models which contain combination of SC and 3 process metric using DT**

### 6.6 Stacking Analysis

If we statistically analyze the results of table 5.15 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 12 and the p-value is 0.0024. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.428 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain process metric just.

Fig 6.15 represents the box plot for table 5.15. We can see that the combined model of process metric and static code metrics outperforms models solely containing process metrics and models solely containing static code metrics and there is one outlier in model solely containing process metrics.

If we statistically analyze the results of table 5.16 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 15.345 and the p-value is 0.0015. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis.

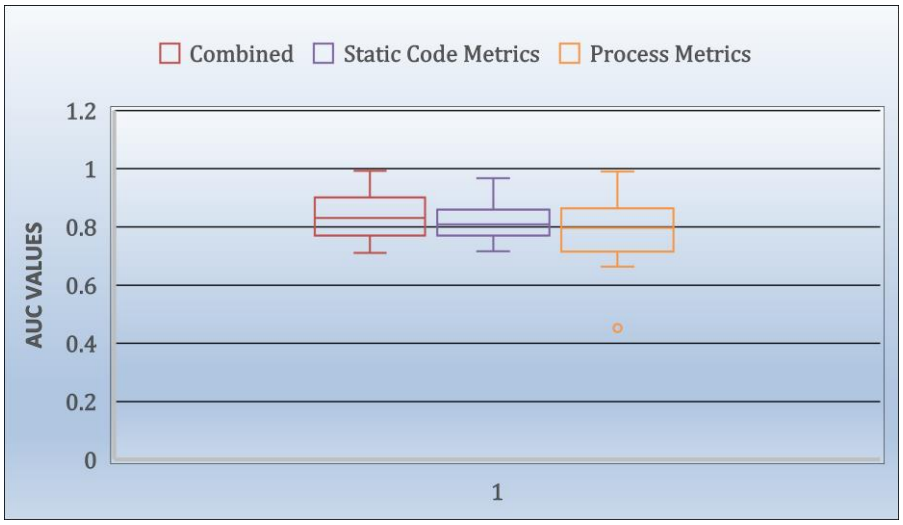


**Fig 6.15 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using stacking**

Hence the performance of at least one of the pair differs significantly with an effect size of 0.365 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NML model, and SC+NDC model and SC+NML model.

Fig 6.16 represents the box plot for table 5.16. We can see that the combination of NR metric with static code metrics outperforms other pairs of process metric and static code metrics.

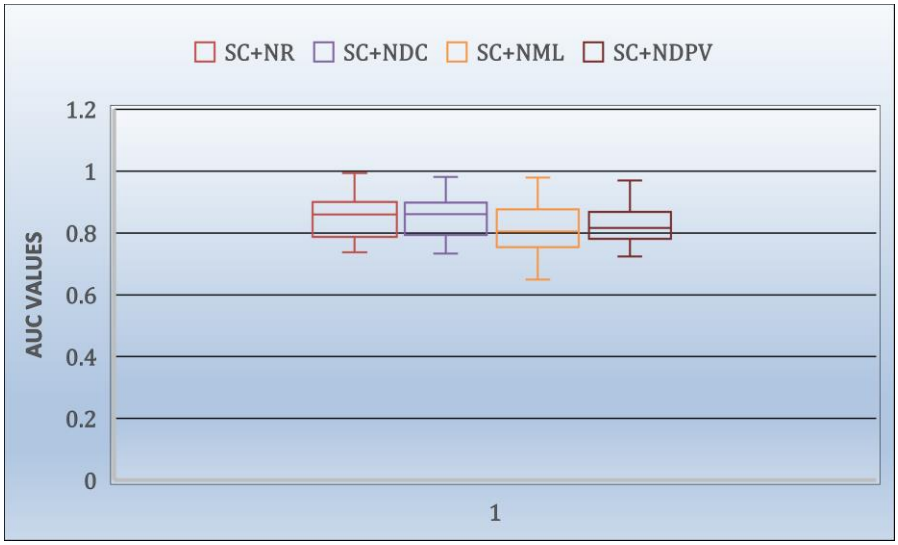


**Fig 6.16 Box plots of models which contain combination of SC and 1 process metric using stacking**

If we statistically analyze the results of table 5.17 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 21.567 and the p-value is 0.0006. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.308 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the SC+NR+NDPV model and SC+NML+NDPV model, and SC+NML+NDPV model and SC+NDC+NDPV model.

Fig 6.17 represents the box plot for table 5.17. We can see that combination of NDC metric with NDPV metric and statis code metric outperforms other pair of process metric and static code metrics.

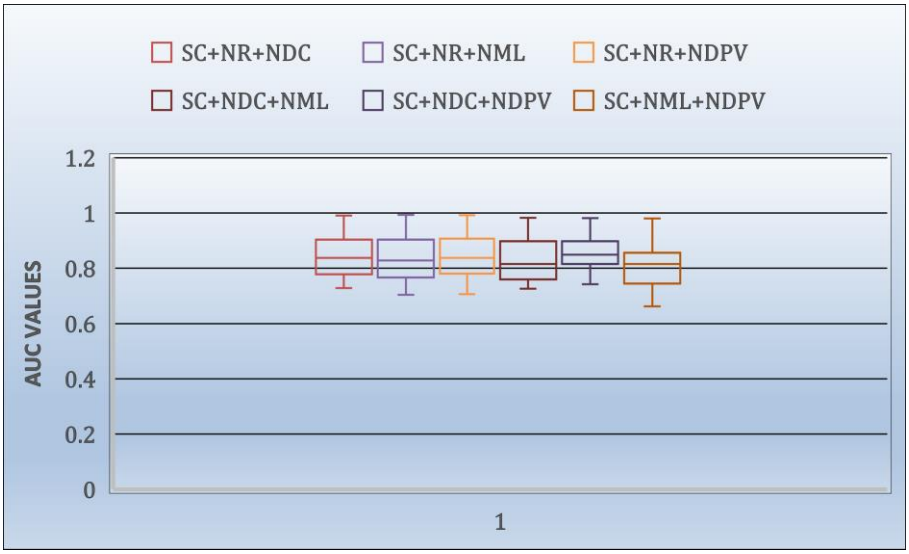


**Fig 6.17 Box plots of models which contain combination of SC and 2 process metrics using stacking**

If we statistically analyze the results of table 5.18 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 21.239 and the p-value is 9.38e-05. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.505 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain 3 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+NDPV+SC model and NDC+NML+NDPV+SC model.

Fig 6.18 represents the box plot for table 5.18. We can see that combination of NR metric with NDC metric, NDPV metric and static code metric outperforms other pair of process metric and static code metrics.

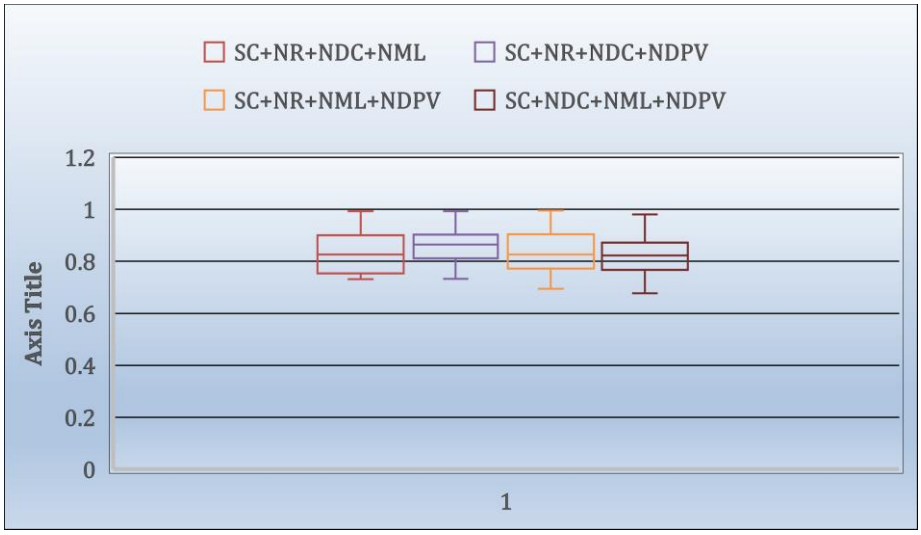


**Fig 6.18 Box plots of models which contain combination of SC and 3 process metric using stacking**

## 6.7 Voting Analysis

If we statistically analyze the results of table 5.19 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 9.148 and the p-value is 0.0103. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.326 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain just process metric, and the combined model and model which contain just static code metric. Fig 6.19 represents the box plot for table 5.19. We can see that the combined model of process metric and static code metrics outperforms models solely containing process metrics and models solely containing static code metrics and there is one outlier in model solely containing process metrics.

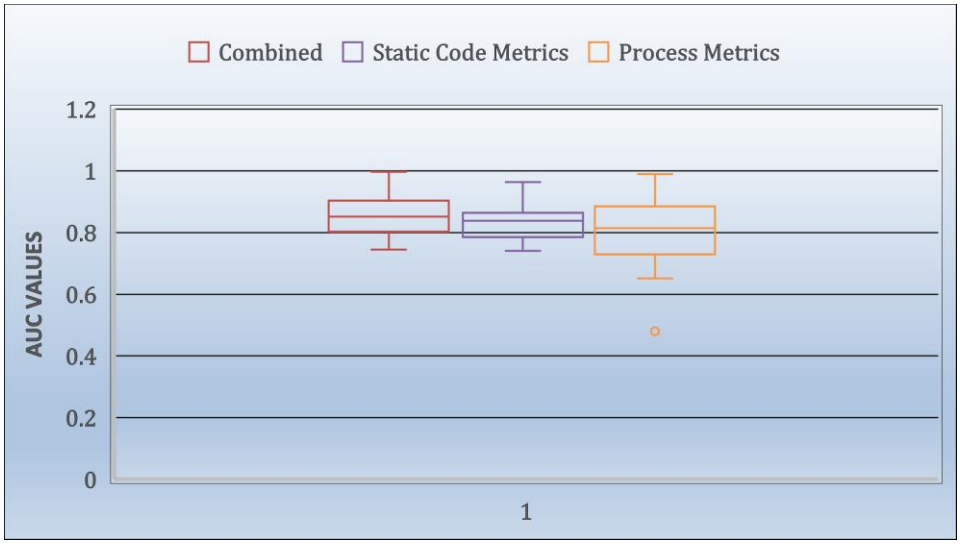


**Fig 6.19 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using voting**

If we statistically analyze the results of table 5.20 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 27.804 and the p-value is 3.99e-06. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pair differs significantly with an effect size of 0.662 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NML model, SC+NR model and SC+NDPV, SC+NDC model and SC+NDPV model, and SC+NDC model and SC+NML model.

Fig 6.20 represents the box plot for table 5.20. We can see that the combination of NR metric with static code metrics outperforms other pairs of process metric and static code metrics.

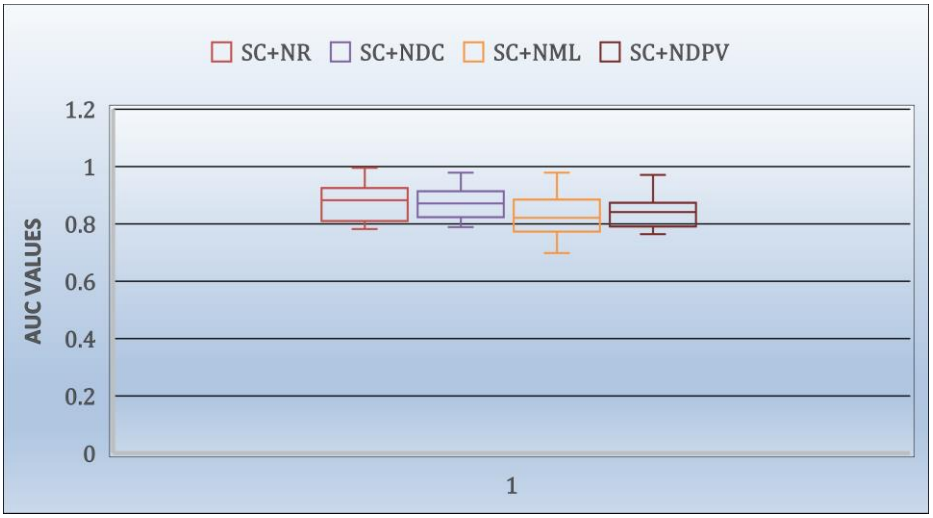


**Fig 6.20 Box plots of models which contain combination of SC and 1 process metric using voting**

If we statistically analyze the results of table 5.21 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 47.868 and the p-value is 3.77e-09. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.684 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+SC model and NR+NML+SC model, NR+NDC+SC model and NDC+NML+SC model, NR+NDC+SC model and NML+NDPV+SC model, NR+NML+SC model and NDC+NML+SC model, NR+NML+SC model and NML+NDPV+SC model, NDC+NML+SC model and NDC+NDPV+SC model, and NML+NDPV+SC model and NDC+NDPV+SC model.

Fig 6.21 represents the box plot for table 5.21. We can see that combination of NR metric with NDC metric and static code metric outperforms other pair of process metric and static code metrics.

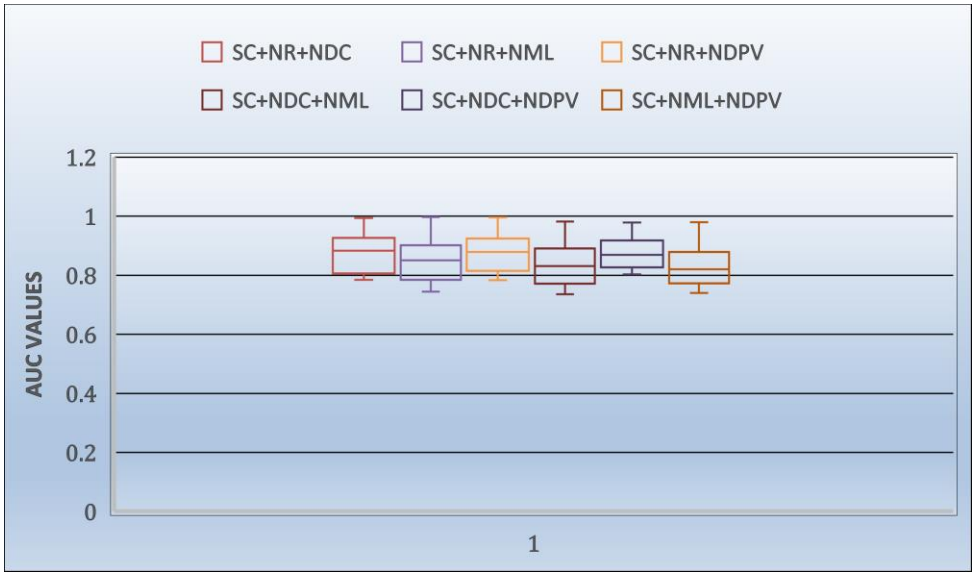


**Fig 6.21 Box plots of models which contain combination of SC and 2 process metrics using voting**

If we statistically analyze the results of table 5.22 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 29.826 and the p-value is 1.50e-06. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.710 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 3 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+NML+SC model and NDC+NML+NDPV+SC model, NR+NDC+NDPV+SC model and NR+NML+NDPV+SC model, and NR+NDC+NDPV+SC model and NDC+NML+NDPV+SC model.

Fig 6.22 represents the box plot for table 5.22. We can see that combination of NR metric with NDC metric, NDPV metric and static code metric outperforms other pair of process metric and static code metrics.

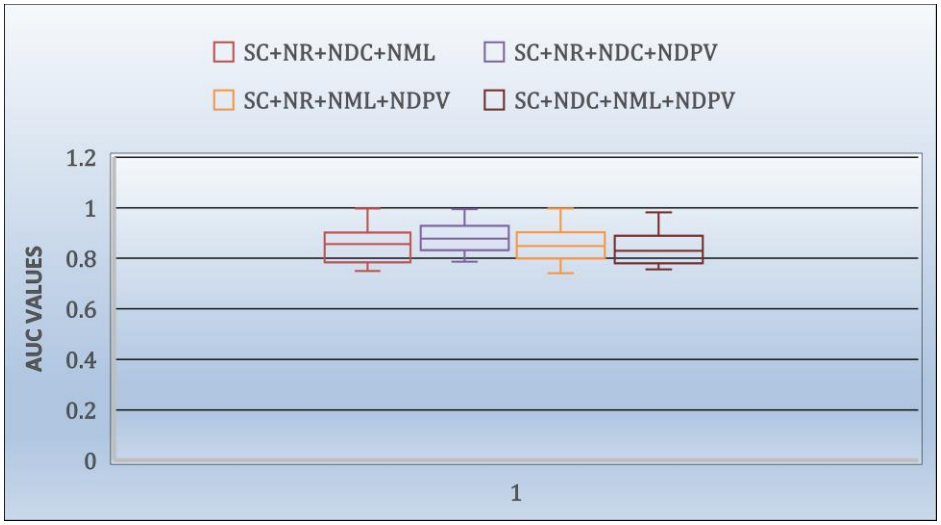


**Fig 6.22 Box plots of models which contain combination of SC and 3 process metric using voting**

## 6.8 Bagging Analysis

If we statistically analyze the results of table 5.23 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 9 and the p-value is 0.011. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.321 which indicates that the difference among the models has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain just process metric, and the combined model and model which contain just static code metric.

Fig 6.23 represents the box plot for table 5.23. We can see that the combined model of process metric and static code metrics outperforms models that solely contains process metrics and models that solely contains static code metrics and there is one outlier in model that solely contains process metrics.

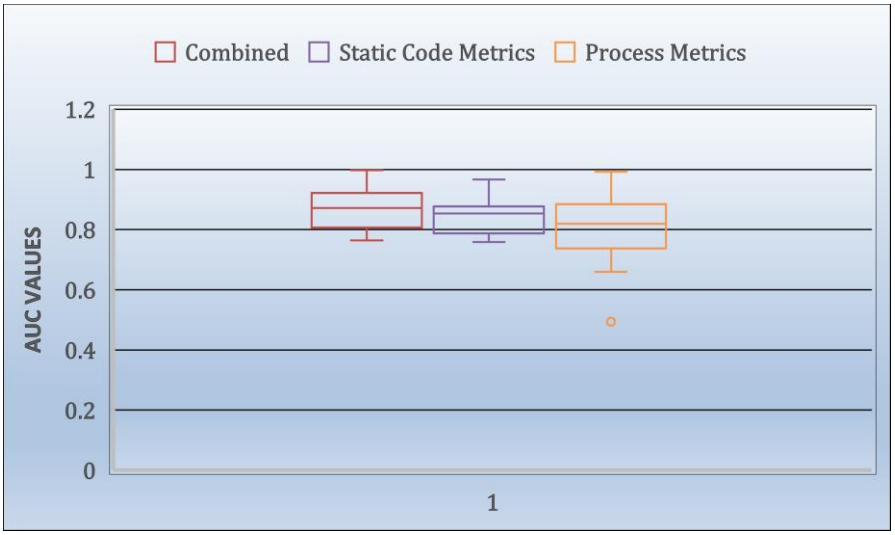


**Fig 6.23 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using bagging**

If we statistically analyze the results of table 5.24 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 20.61 and the p-value is 0.0001. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pair differs significantly with an effect size of 0.490 which indicates that the difference among the pairs has a small effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NML model, SC+NR model and SC+NDPV, and SC+NDC model and SC+NDPV model.

Fig 6.24 represents the box plot for table 5.24. We can see that the combination of NR metric with static code metrics outperforms other pairs of process metric and static code metrics.

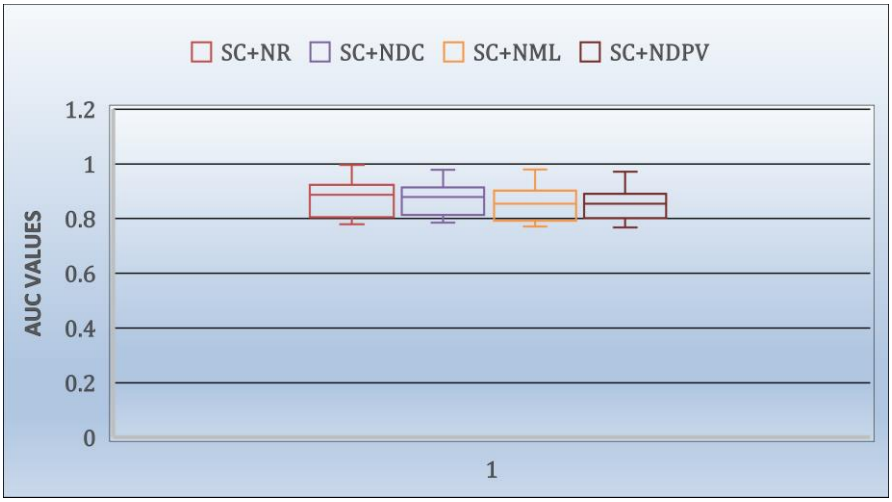


**Fig 6.24 Box plots of models which contain combination of SC and 1 process metric using bagging**

If we statistically analyze the results of table 5.25 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 35.74 and the p-value is 1.069e-06. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.511 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+SC model and NDC+NML+SC model, NR+NDC+SC model and NML+NDPV+SC model, NR+NDPV+SC model and NDC+NML+SC model, and NR+NDPV+SC model and NML+NDPV+SC model.

Fig 6.25 represents the box plot for table 5.25. We can see that combination of NR metric with NDC metric and static code metric outperforms other pair of process metric and static code metrics.

If we statistically analyze the results of table 5.26 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 29.826 and the p-value is 1.50e-06. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of

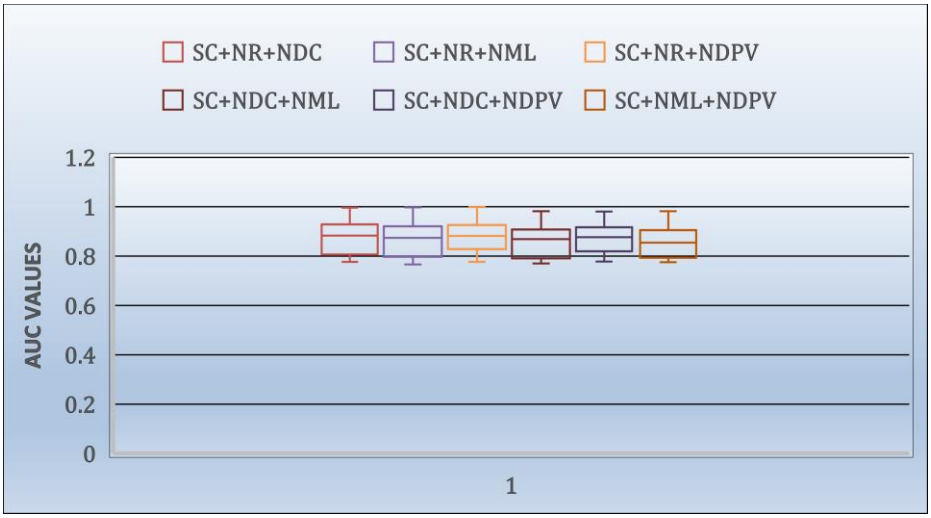


**Fig 6.25 Box plots of models which contain combination of SC and 2 process metrics using bagging**

0.710 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 3 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+NML+SC model and NR+NDC+NDPV+SC model, NR+NDC+NML+SC model and NDC+NML+NDPV+SC model, NR+NDC+NDPV+SC model and NDC+NML+NDPV+SC model, and NR+NDC+NDPV+SC model and NR+NML+NDPV+SC model.

Fig 6.26 represents the box plot for table 5.26. We can see that combination of NR metric with NDC metric, NDPV metric and static code metric outperforms other pair of process metric and static code metrics.

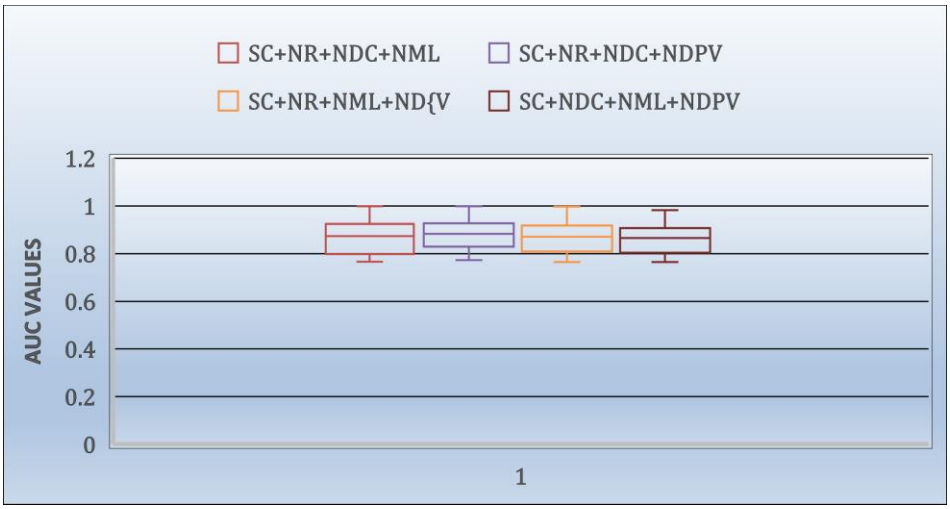


**Fig 6.26 Box plots of models which contain combination of SC and 3 process metrics using bagging**

### 6.9 Boosting Analysis

If we statistically analyze the results of table 5.27 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 14.285 and the p-value is 0.0007. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 2 is 5.99 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the models differs significantly with an effect size of 0.510 which indicates that the difference among the models has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models. The results show that a significant difference exists between the performances of the combined model and model which contain just process metric.

Fig 6.27 represents the box plot for table 5.27. We can see that the combined model of process metric and static code metrics outperforms models that solely contains process metrics and model that solely contains static code metrics.

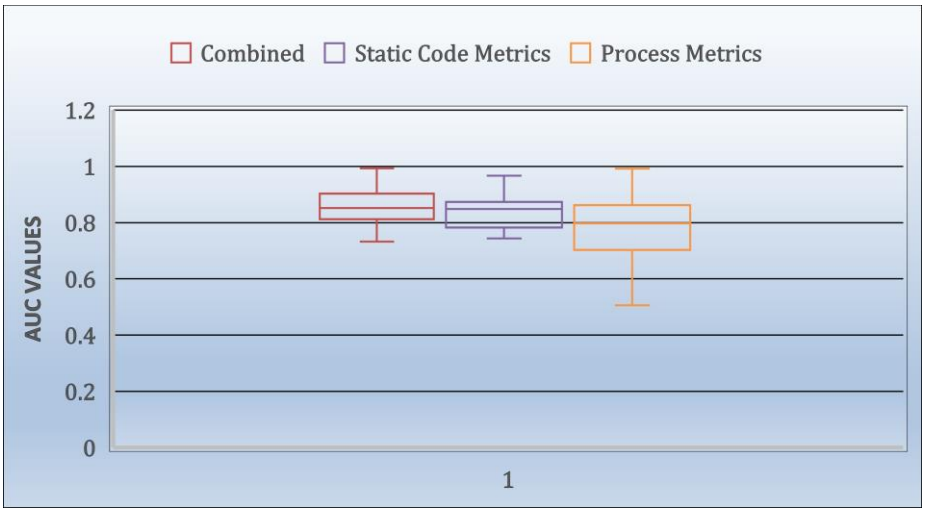


**Fig 6.27 Box plots of combined model, model solely containing static code metrics and model solely containing process metrics using boosting**

If we statistically analyze the results of table 5.28 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 18.625 and the p-value is 0.0003. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pair differs significantly with an effect size of 0.444 which indicates that the difference among the pairs has a small effect on the prediction

performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contain all static code metric and 1 process metric. The results show that a significant difference exists between the performances of the SC+NR model and SC+NML model, SC+NR model and SC+NDPV, and SC+NDC model and SC+NML model.

Fig 6.28 represents the box plot for table 5.28. We can see that the combination of NR metric with static code metrics outperforms other pairs of process metric and static code metrics.

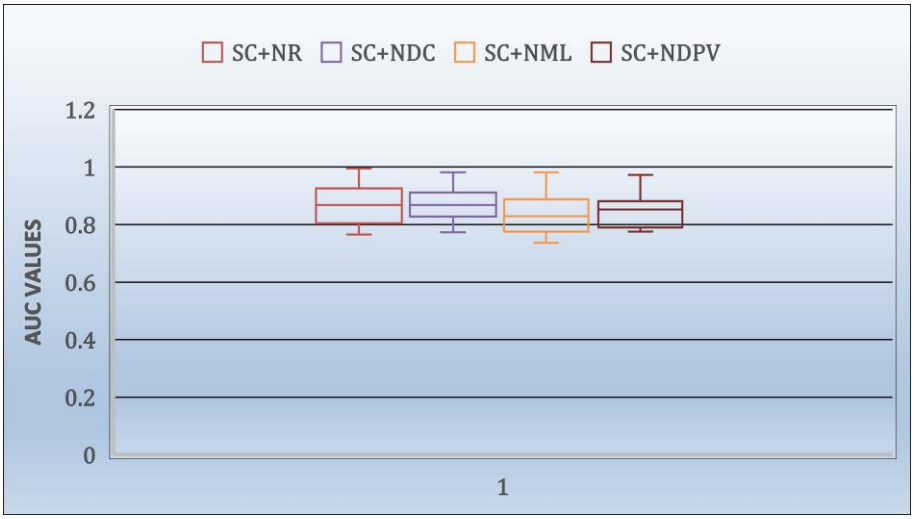


**Fig 6.28 Box plots of models which contain combination of SC and 1 process metrics using boosting**

If we statistically analyze the results of table 5.29 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 45.04 and the p-value is 1.423e-08. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 5 is 11.070 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.643 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 2 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+SC model and NR+NML+SC model, NR+NDC+SC model and NML+NDC+SC model, NR+NDC+SC model and NML+NDPV+SC, NR+NML+SC model and NR+NDPV+SC model, NR+NDPV+SC model and NDC+NML+SC model, NR+NDPV+SC model and NML+NDPV+SC model, NDC+NML+SC model and NDC+NDPV+SC model, and NDC+NDPV+SC model and NML+NDPV+SC model.

Fig 6.29 represents the box plot for table 5.29. We can see that combination of NR metric with NDC metric and static code metric outperforms other pair of process metric and static code metrics.

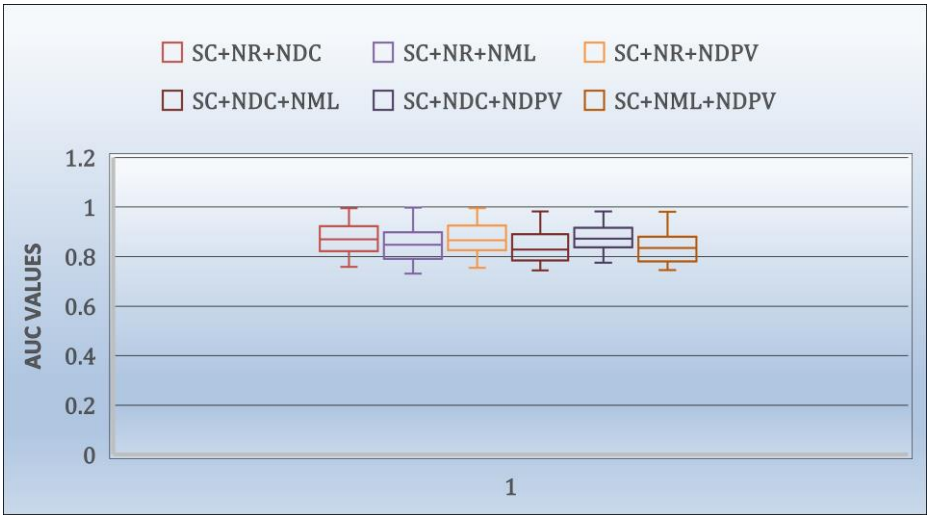


**Fig 6.29 Box plots of models which contain combination of SC and 2 process metrics using boosting**

If we statistically analyze the results of table 5.30 using Friedman test at the 0.05 significance level, then results show that the calculated $\chi^2$-statistics is 24.860 and the p-value is 1.65e-05. From the chi-square table, $\chi^2$ value at significance level 0.05 and degree of freedom 3 is 7.815 which is less than the calculated $\chi^2$-statistics so there is a rejection of the null hypothesis. Hence the performance of at least one of the pairs differs significantly with an effect size of 0.592 which indicates that the difference among the pairs has a large effect on the prediction performance. As the results differ significantly, we will apply a Nemenyi post hoc test to check the pairwise comparison of the models which contains 3 process metrics and all static code metrics. The results show that a significant difference exists between the performances of the NR+NDC+NML+SC model and NR+NDC+NDPV+SC model, NR+NDC+NDPV+SC model and NDC+NML+NDPV+SC model, and NR+NDC+NDPV+SC model and NR+NML+NDPV+SC model.

Fig 6.30 represents the box plot for table 5.30. We can see that combination of NR metric with NDC metric, NDPV metric and static code metric outperforms other pair of process metric and static code metrics.

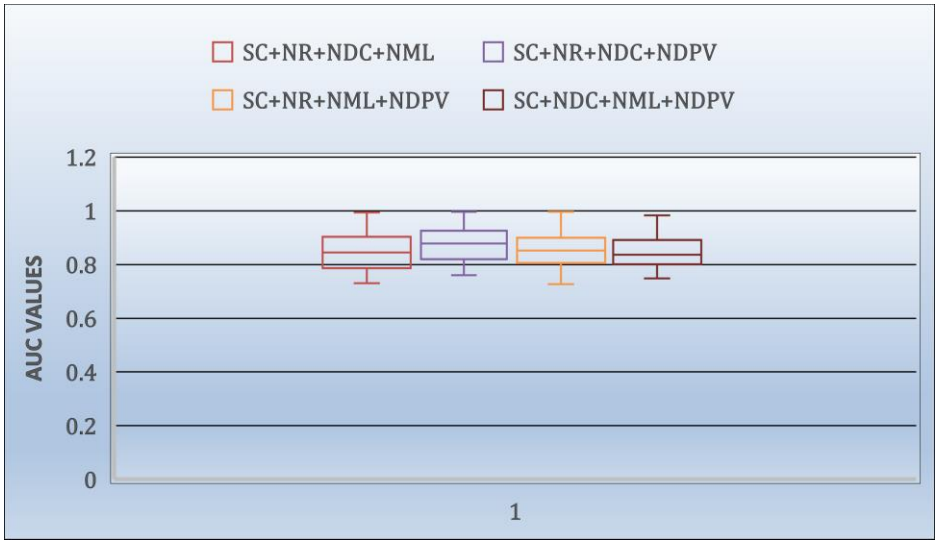


**Fig 6.30 Box plots of models which contain combination of SC and 3 process metrics using boosting**

# CHAPTER 7

# CONCLUSION AND FUTURE WORK

## 7.1 Conclusion

Various types of models exist and are researched in the field of software fault prediction. Many of them utilize static code metrics to predict faulty modules. Here we are analyzing the effectiveness of process metrics on fault prediction results using different classification techniques and ensemble techniques. We have analyzed the results based on AUC values, boxplots and statistical test (Friedman test with Nemenyi post hoc test). We can conclude that when we are using logistic regression, models containing only static code metrics give the best result. When we are using K nearest neighbor, models containing only process metrics give the best result, and combination of process metric and static code metric gives the best result in the case of naive bayes, decision tree, support vector machines, stacking, voting, bagging and boosting. The answer to the research questions are as follows:

**RQ1: Is process metric as effective as static code metrics in checking software fault proneness?**

Process metrics are good predictors as the results of most of the classification techniques and ensemble techniques show that the combination of process metric and static code metric gives better prediction results based on experimental analysis, boxplot analysis and statistical analysis.

**RQ2: Which classification technique gives best outcome for combined model?**

As compare to other classification techniques, the results of Naïve Bayes, Support Vector Machine and Decision tree shows the better prediction outcome for combined model based on experimental analysis, boxplot analysis and statistical analysis.

**RQ3: To check whether ensemble techniques improve the prediction performance as compared to individual classification techniques or not?**

As compare to individual classification technique results, ensemble techniques (Stacking, Voting, Bagging and Boosting) gives better AUC values for combined model in almost all the analyzed dataset.

**RQ4: Which process metric is more effective among all selected process metrics?**

To answer this research question, we have analyzed the prediction results of model 4, model 5 and model 6 based on AUC values, box plots and Friedman test with Nemenyi post hoc test. The results are as follows:

If we analyze the results of model 4 which is a combined model of 1 process metric and all static code metric then the result of most of the techniques based on experimental analysis, boxplot analysis and statistical analysis shows NR metric is more effective.

If we analyze the model 5 which is a combined model of 2 process metric and all static code metric then the result of most of the techniques based on experimental analysis, boxplot analysis and statistical analysis shows NR+NDC pair is more effective.

If we analyze the model 6 which is a combined model of 3 process metric and all static code metric then the result of most of the techniques based on experimental analysis, boxplot analysis and statistical analysis shows NR+NDC+NDPV pair is more effective.

## 7.2 Future Work

This work can be extended to analyzing other process metrics using classification techniques and ensemble techniques. Also, we can predict number of defects using regression technique considering same dataset used in this work. Also, instead of bugs, we can consider effort or maintainability as dependent variable.

# CHAPTER 9

# REFERENCE


[1] A. Okutan and O. T. Y. Software defect prediction using Bayesian networks. Empirical Software Engineering, 19(1):154–181, 2014

[2] 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.

[3] B. Diri, C. Catal, U. Sevim. Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm. Expert Syst. Appl., 38 (2011), pp. 2347-2353.

[4] 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.

[5] 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.

[6] D. Radjenovic, M. Heri cko, R. Torkar, and A. Zivkovi c. Software fault prediction metrics: A systematic literature review. Information and Software Technology, 55(8):1397 – 1418, 2013.

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

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

[9] F. Rahman and P. Devanbu, "How, and why, process metrics are better", 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 2013, pp. 432-441, doi: 10.1109/ICSE.2013.6606589.

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

[11] https://madeyski.e-informatyka.pl/tools/software-defect-prediction

[12] https://kenai.com/projects/buginfo

[13] http://www.spinellis.gr/sw/ckjm

[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] 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).

[16] J. Hanley, B.J. McNeil. The meaning and use of the area under a Receiver Operating Characteristic ROC curve Radiology, 143 (1982), pp. 29-36.

[17] 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.

[18] 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.

[19] L. Madeyski and M. Jureczko. Which process metrics can significantly improve defect prediction models? An empirical study. Software Quality Journal, 23(3):393–422, 2015.

[20] 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).

[21] 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.

[22] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. IEEE Transactions on Software Engineering, 37(3):356–370, 2011.

[23] R. Malhotra. Empirical Research in Software Engineering. Chapman & Hall/CRC:978-1-4987-1972-8, 2015.

[24] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In Proceedings of the 30th International Conference on Software Engineering (ICSE'08), New York, NY, USA, 2008. ACM.

[25] S. Dreiseitl, L. Ohno-Machado. Logistic Regression and Artificial Neural Network Classification models: a methodology review. J. Biomed. Inform., 35 (2002), pp. 352-359.

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

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

[28] T. Hall, S.Beecham, D. Bowes, D.Gray, and S.Counsell, “A systematic literature review on fault prediction performance in software engineering”, Software Engineering, IEEE Transactions, Volume 38, Issue 6, pp 1276– 1304, Nov.-Dec. 2012.

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

[30] Y. A. Alshehri, K. Goseva-Popstojanova, D. G. Dzielski and T. Devine, “Applying Machine Learning to Predict Software Fault Proneness Using Change Metrics, Static Code Metrics, and a Combination of Them”, SoutheastCon 2018, St. Petersburg, FL, USA, 2018, pp. 1-7, doi: 10.1109/SECON.2018.8478911.

[31] Y. Kamei, H. Sato, A. Monden, S. Kawaguchi, H. Uwano, M. Nagura, K. I. Matsumoto, and N. Ubayashi. An empirical study of fault prediction with code clone metrics. In Proceedings of the 21st International Workshop on Software Measurement and 6th International Conference on Software Process and Product Measurement, 2011.

[32] Y. Xia, G. Yan, and H. Zhang. Analyzing the significance of process metrics for TT&C software defect prediction. In Proceedings of the 5th IEEE International Conference on Software Engineering and Service Science, 2014.

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