"Empirical Validation of Process Metrics to Check the Predictive Performance of Classification and Ensemble Methods"

A PROJECT REPORT

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

MASTER OF TECHNOLOGY
IN
DATA SCIENCE

Submitted By

Rohit Ramchandani
(2K21/DSC/09)

Under the supervision of

Prof. Ruchika Malhotra
Head of Department (Software Engineering)
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, 2023

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

DECLARATION

I, Rohit Ramchandani, 2K21/DSC/09 student of M.Tech (DSC), hereby declare that the project entitled "***Empirical validation of process metrics to check the predictive performance of classification and ensemble methods***" 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 Data Science, 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 **Rohit Ramchandani**

Date: June, 2023 (**2K21/DSC/09)**

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

CERTIFICATE

I hereby certify that the project entitled "***Empirical validation of process metrics to check the predictive performance of classification and ensemble methods***" which is submitted by Rohit Ramchandani (2K21/DSC/09) 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 Data Science, 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

HOD and Professor

Department of Software Engineering

ACKNOWLEDGEMENT

I am very thankful to **Prof. Ruchika Malhotra** (Head of Department, Professor, 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.

Rohit Ramchandani

2K21/DSC/09

ABSTRACT

Software defects have always been considered a major problem in the software industry and for software engineers, early detection improves software performance and reduces faults, time, and cost. In order to predict defects in software, many researchers have been used classification and ensemble techniques. Different dataset produces different results. In this research, we have evaluated the prediction accuracy of classification and ensemble approaches using 3 distinct models: combined model of static code and process metrics, model containing process metrics, and model containing static code metrics. In simple terms, we can say that these 3 models have different independent variables and dependent variables are the actual values of bugs which is the same. We have used NB, LR, KNN, SVM, DT as classification approaches and stacking, voting, bagging, and boosting as ensemble approach for implementation. The dataset was gathered from the publicly available repository. AUC metric was used to examine the prediction performance of classification and ensemble techniques. Additionally, the statistical significance of the results obtained from various models was assessed using the Friedman and Nemenyi post hoc test. The result of this study demonstrates that the use of process metrics in predicting the defects in software produces effective outcomes.

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
WMC	Weighted Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Objects

RFC	Response For a Class
LOC	Lines of Code
LCOM	Lack of Cohesion in Methods
NPM	Number of Public Methods
DAM	Data Access Metric
MOA	Measure of Aggregation
MFA	Measure of Functional Abstraction
IC	Inheritance Coupling
CAM	Cohesion Among Methods
CBM	Coupling Between Methods
AMC	Average Method Complexity
Ca	Afferent Coupling
Ce	Efferent Coupling

CHAPTER 1

INTRODUCTION

1.1 Introduction

Software development life cycle consist of software testing which is time consuming and resource consuming. The aim of the testing procedure is to deliver software that is completely error-free and meets the needs of all stakeholders. Finding software flaws is a necessary but expensive process. Testing adds to the overall project budget because it is an expensive process. The effectiveness and quality of the software improve when defects are correctly predicted at an early stage. Additionally, accurate defect prediction aids in controlling the project's budget.

1.2 Software Defect Prediction

Defects prediction and proneness in the software are always considered a major problem in the software industry and for software engineers. In classical methods, previous experience with the defective or non-defective software is required while detecting the software defect in a software application. Defects in the software applications have been predicted by using different classification techniques by different researchers as of now but the results of the methods changed with the dataset, so they lack the property of robustness for defect prediction in an un- known software application. However, ensemble techniques for defect detection in software applications will be very efficient, as there will be an advantage of using various techniques on a specific dataset to predict software defects with much more accurate than using a single technique, for the reasons stated above. An ensemble learning software defect prediction model will allow the software to extensively identify and rectify software defects using soft voting and hard voting. This ability makes the soft- ware run more effectively and reduces faults, time, and cost.

1.3 Motivation

Different software metrics are used by software defect prediction (SDP) models to make predictions. Numerous studies have been conducted in the past using static code metrics to identify different system design elements. Numerous techniques, including LR [6], NB [10][4], SVM [20], ANN [21], KNN [23], and DT[30] have been suggested in the past to demonstrate the correlation between static code metrics and defect proneness. Investigating process metrics is necessary in the area of defect prediction since both the studies Dejaeger *et al*. [20] and Okutan and Yildiz [2] highlight the same. The field of software defect prediction has witnessed a lot of literature reviews published. The majority of them employ static code metrics, while only a limited number of studies demonstrate the connection between defect proneness and process metrics. There is no clear-cut conclusion provided by the authors some claim that process metrics outperform static code metrics, while others contend that the reverse is true.

1.4 Objective

The main objective of this thesis is to examine the effectiveness of process metrics in predicting outcomes. To assess and compare different techniques, we will construct a defect prediction model in this thesis and evaluate their performance based on AUC values, bar graphs, and statistical test. A heterogeneous ensemble learning model (bagging, stacking, boosting, and voting) is introduced and the comparison of the ensemble model is made with all the classifiers i.e., DT, LR, KNN, NB, SVM which have been used to build the model. In this thesis, AUC has been used so to determine whether the ensemble algorithm is accurate or not in determining whether the software is defective or not defective. We analyze the results of each technique using distinct models. Further we will analyze, if we use process metrics, which techniques gives better results and can we use process metrics to determine defect proneness. Additionally, we will evaluate the performance of selected process metrics by considering different combinations of static code and process metrics to determine the most effective one.

1.5 Thesis Structure

The structure of this research is as follows: The related work that has previously been done in the area of software defect prediction is presented in Section 2. The proposed work is presented in Section 3, which goes over the dataset used, dependent and independent variables used, and static code metrics and process metrics used. Research Methodology is presented in Section 4, which goes over the performance evaluation metric used, the classification and ensemble strategies employed, the statistical test employed, and how these strategies will be put into practice. The output of each model using each classification method and ensemble technique is shown in Section 5. The discussion of the results is presented in Section 6, which includes bar graphs and statistical tests. Section 7 of this study presents its conclusion and future work.

CHAPTER 2

LITERATURE SURVEY

A lot of research has been conducted to develop software defect prediction models using different techniques including Machine Learning techniques and ensemble learning techniques. Numerous studies have been conducted to develop software defect prediction models using various defect prediction techniques, but the majority of these studies use static code metrics as independent variables, while only a small number of them use process metrics.

Related Work:

According to a systematic literature review [7] that included 106 papers published between 1991 and 2011, OOM were used as independent variables in 49% of the research, conventional source code metrics in 27% of the research, and process metrics in 24% of the papers. The prediction results of object-oriented metrics and process metrics outperform those of conventional source code metrics. Delta metrics, code churn metrics, history metrics, and developer metrics are the process metrics used in this literature review.

According to some studies, process metrics did not perform well when used in the early phases of software development. The results of the experiments conducted have demonstrated that process metrics perform better in the post-release stage of software development.

Two new process metrics were introduced [29]:

a. Life cycle-based management process metric

b. History changes process metric.

On the basis of the traits of the development process, these metrics were created. They discuss the efficiency of process metrics during the analysis, coding, and design stages of the

development life cycle. The findings of this study demonstrate that combining process metrics and code metrics improves defect prediction and error rates are reduced.

An experiment is performed in which the authors took into account both open-source and commercial projects. They looked at models that included all SC metrics and one process metric. This study demonstrates that process metrics, particularly NDC and NML process metrics, significantly contribute to improving defect prediction [15]. A combined model comprising of all static code metrics and all process metrics is suggested for better prediction results [25].

Komalasari and Candra [1] used a combination of product metrics, process metrics and profile metrics to build classification model. They used NB, LR, and RF techniques to predict the defects. The result shows that combination of these three metrics gives effective result in RF model.

Choudhary *et al.* [8] proposed new change metrics and analyzed the effect of existing and proposed change metrics in software defect prediction on eclipse dataset. They used decision tree, KNN and random forest as classification technique and precision, recall and F-measure as performance evaluation metrics. They analyzed that combined model of change and code metrics gives better performance as compared to model that have individual set of metrics on eclipse dataset.

According to Ghotra *et al.* [5], the choice of classifier can lead to an increase or decrease in prediction model accuracy of up to 30%. Perreault *et al.* [16] conducted a study on five NASA datasets, comparing the effectiveness of Artificial Neural Networks, LR, NB, SVMs, and KNN. However, the study did not provide a clear explanation of which strategy was the most effective.

Also, Panichella *et al*. [3] demonstrate that despite comparable prediction accuracy, the predictions of different classifiers are highly interconnected. The model can be trained to

utilize massive volumes of data either from the project under being observed or from a comparable project that has not been observed yet (cross-project strategy).

Hussain *et al.* [22] in their study conducted a comparison of three ensemble techniques utilizing five base classification techniques (J48, LR, NB, Voted-Perceptron, and SVM) in the Weka tool for software defect prediction. The results of their study indicate that Stacking outperforms all the selected ensemble methods.

Rhmann *et al*. [27] analyzed the performance of hybrid search-based algorithms for software defect prediction using change metrics. They used android dataset and recall and precision as performance evaluation measure. The result shows that hybrid search-based algorithms perform better as compared to machine learning techniques.

CHAPTER 3

SYSTEM DESIGN

3.1 Proposed Architecture

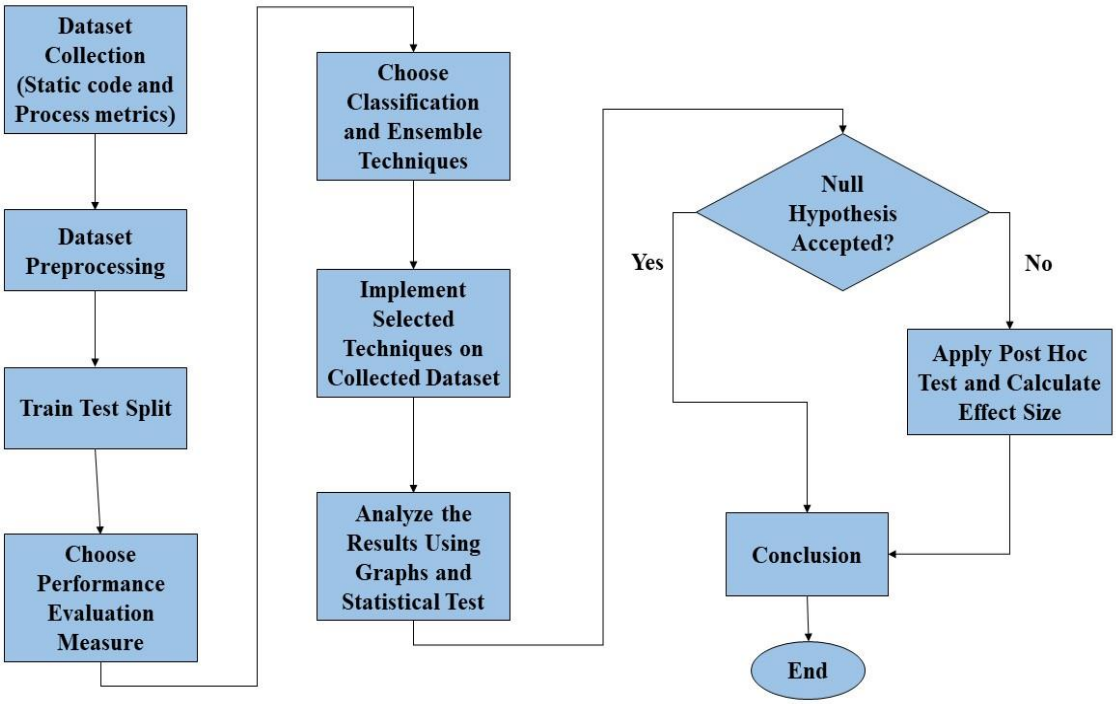

Fig 3.1 Architecture of Proposed Methodology

This chapter defines the methodology used for the implementation Figure 3.1 displays the methodology proposed and along with it independent and dependent metrics or variables i.e No. of Bugs is a dependent variable or metric and Weighted method per class is the independent variable or metric which have been taken into account for this project and dataset's collection.

3.2 Dependent and Independent Variables

Independent metrics or variables which have been considered in this project are static code metrics such as WMC, DIT, NOC, CBO etc. and process metrics such as NR, NDC etc. and

the dependent metrics or variable is defect count or its likelihood which has been defined as the inclination of predicting defects in the class i.e. defective and non-defective.

3.2.1 Static Code Metrics

The design complexity and the size of software system are defined by the static code metrics, they have been widely used in creating a defect detection model. The static code metrics used in this project are WMC, NOC, LOC, DIT, LCOM, CBO, RFC, NPM, DAM, LCOM3, MFA, CAM, MOA, CBM, IC, AMC, Ca, Max (CC), Ce, and Avg (CC). The static code metrics' definition can be found in an independent report provided by Jureczko and Madeyski [17]. This information is openly accessible as open-source data.

3.1.2 Process Metrics

The quality and effectiveness of the system are determined by process metrics. These metrics provide additional descriptive information about defective modules. They are obtained from two sources:

a) The developer's experience
b) The software change history

The process metrics utilized in this thesis are as follows:

1. **NR**

NR is an acronym for "Number of Revisions," which refers to the quantity of modifications made to a Java class throughout the development stages of the analyzed software release. This metric provides insight into the frequency of amendments made to the class during its evolution.

2. **NDC**

The acronym NDC represents Number of Distinct Committers, which refers to the count of developers who have contributed changes to a Java class throughout the development of a specific software release. This metric provides insight into the quantity of individuals involved in modifying the examined release.

3. **NML**

The acronym NML represents Number of Modified Lines, which calculates the quantity of source code lines that have been added or removed from a Java class. This metric takes into account every change submitted throughout the software's evolution in the examined release.

4. **NDPV**

NDPV represents the acronym for "Number of Defects in Previous Versions," which quantifies the quantity of issues addressed in a Java class throughout the progression of the preceding software release.

3.2 Empirical Data Collection

We have collected the data for 4 java-based projects (Ant, jEdit, Xalan, and Xerces). The dataset is collected from the publicly available repository [9] which consist of both SC and process metrics. We chose java project datasets since they contain static code metrics such as WMC, DIT, LOC, LCOM etc. and process metrics such as NR, NDC, NML, and NDPV [17]. Table 3.1 lists the datasets that were used in this study:

Table 3.1 Dataset Used

Project Name	**Number of Instances**	**Defective Instances**	**Non-Defective Instances**
Ant 1.6	524	92	432
Ant 1.7	1065	166	899
jEdit 4.0	606	75	531
jEdit 4.1	644	79	565
Xalan 2.6.0	1170	411	759
Xalan 2.7.0	1194	898	296
Xerces 1.3.0	545	69	476
Xerces 1.4.4	671	437	234

CHAPTER 4

METHODOLOGY

We have used the following steps:

- Obtain datasets with static code and process metrics from publicly accessible repositories.
- Perform some preprocessing operations on datasets.
- Choose some classification and ensemble techniques.
- Choose performance evaluation measures to evaluate the prediction performance.
- Analyze the performance based on chosen evaluation measure.
- Validation of results using statistical test.

4.1 Dataset Preprocessing

Datasets obtained from publicly accessible repositories have some missing values, which impair the performance of the created model, hence preprocessing is done on datasets to avoid this type of problem, such as eliminating data points with missing values because they are extremely small in count or replacing missing values with mean value or median value or with some user constant. Since there are some string data type variables which is an invalid input to the classifier so we use a filter method to convert the datatype into compatible data type.

4.2 Classification Techniques

Classification techniques are algorithms in machine learning utilized for predicting the class or category of a provided input, relying on a set of features or attributes. These techniques acquire knowledge from labeled training data, discerning patterns, and relationships and subsequently apply this acquired knowledge to classify new instances that have not been

previously encountered into predefined classes. In this, the model is first trained using training data, and then the data for the invalid dataset is predicted.

In this study, we use NB [4], LR [6], DT [30], SVM [20], and KNN [23] technique for developing the prediction model. For implementation, Waikaito Environment for Knowledge Analysis (WEKA) machine learning tool is used. The classification of machine learning techniques is depicted in Figure 4.1.

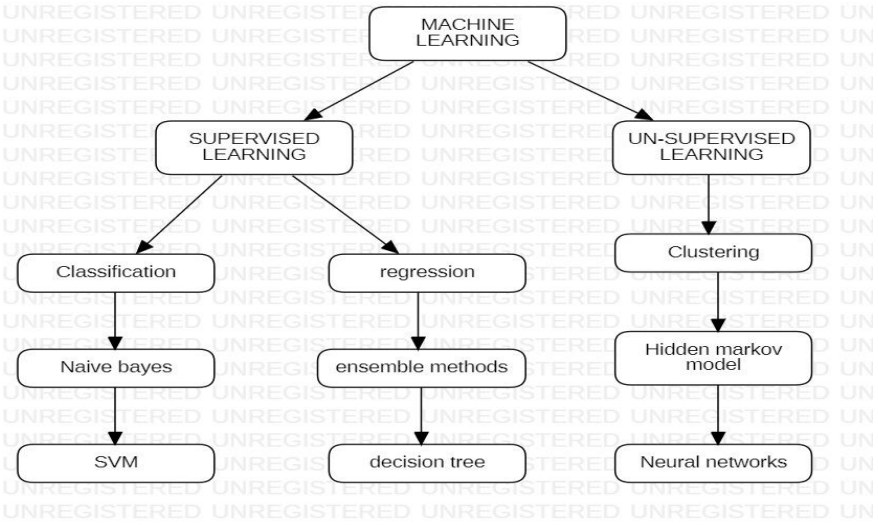

Fig 4.1 Machine Learning Techniques

The five classification techniques selected for implementation are as stated below:

4.2.1 NB

NB is a classification algorithm based on Bayes' theorem and assumes that the features are conditionally independent of each other given the class. It is a simple yet powerful algorithm commonly used for text classification and other tasks with high-dimensional feature spaces. Despite its "naive" assumption of feature independence, NB often performs well in practice and can be computationally efficient [4][10].

The algorithm is called Naive Bayes because it assumes that the occurrence of a specific feature in a class is unrelated to the presence or absence of other features. This simplifies the calculations and allows the algorithm to make predictions efficiently. Some common variations of NB include:

- **Gaussian NB:** It assumes that the numerical features follow a Gaussian (normal) distribution.

- **Multinomial NB:** It is often used for text classification tasks with discrete features, such as word counts.
- **Bernoulli NB:** It is similar to Multinomial NB but assumes binary features, often used for binary text classification tasks.

It performs well in situations where the independence assumption holds reasonably well and when there is a relatively large number of features compared to the size of the training dataset. Naive Bayes can handle categorical and numerical features, and it is particularly suitable for text classification tasks.

4.2.2 LR

LR is a statistical classification algorithm used to predict the probability of categorical outcomes based on one or more independent variables. Despite its name, it is a classification algorithm rather than a regression algorithm. It represent the association between the independent variables and the likelihood of a certain outcome using a logistic function, also known as the sigmoid function.

It estimates the coefficients or weights for each independent variable in the dataset. The algorithm applies an optimization algorithm, such as maximum likelihood estimation, to fit the model to the training data. It uses a logistic or sigmoid function to map the linear combination of the independent variables and their coefficients to a value between 0 and 1. Basili *et al*. [26] and Hosmer and Lemeshow [6] provides an in-depth explanation of LR.

It is a widely used algorithm for binary classification tasks, particularly when interpretability and probabilistic predictions are desired. It is commonly applied in various domains, including healthcare, finance, and social sciences.

4.2.3 SVM

SVM is a powerful supervised machine learning algorithm used for classification and regression purposes. Its objective revolves around discovering an ideal hyperplane that effectively distinguishes data points belonging to distinct classes or predicts a continuous target variable based on the maximum margin principle.

The main idea behind SVM is to transform the input data into a higher-dimensional feature space and identify a hyperplane that optimally maximizes the separation between the classes.

The margin represents the space between the hyperplane and the nearest data points of each class [18]. By maximizing the margin, SVM aims to achieve better generalization and robustness to new data.

SVM can map the input features into a higher-dimensional space using a technique called the kernel trick. The kernel function allows SVM to implicitly operate in the higher-dimensional feature space without explicitly computing the transformations. Commonly used kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid [28]. We have used linear kernel function.

SVM is widely used in various domains, such as text categorization, image recognition, bioinformatics, and finance [20]. It is particularly effective when dealing with complex datasets with a clear separation between classes or a need for nonlinear decision boundaries.

4.2.4 KNN

KNN is a supervised machine learning algorithm used for both classification and regression purposes. KNN is also known as lazy learning technique. KNN makes predictions based on the similarity between the new data point and its neighboring data points in the training dataset.

The basic idea behind KNN is to classify a new data point or predict its value by examining the k nearest data points in the feature space. The "nearest" neighbors are determined based on a distance metric, such as Euclidean distance or Manhattan distance [23].

KNN is commonly used in various applications such as recommender systems, image recognition, and anomaly detection. It can be particularly useful when dealing with datasets that have clear patterns and local structures.

4.2.5 DT

A DT is a supervised machine learning algorithm used for both classification and regression purposes. A tree-like model is generated by analyzing the data's characteristics, depicting decisions and their potential outcomes. Each internal node in the tree corresponds to a specific feature or attribute, while each leaf node indicates a class label or a predicted value. The main idea behind a DT is to recursively partition the data based on the feature values to create a tree structure that can make predictions or classifications. At each internal node of

the tree, a decision is made based on a feature, and the data is split into branches corresponding to different feature values. This process continues until a predetermined condition is fulfilled, such as attaining the maximum tree depth or when all instances in a leaf node are of same class.

In this thesis, we use the REPTree. It is a decision tree algorithm that focuses on reducing errors through a pruning technique. It is commonly used for classification purpose. REPTree builds a decision tree in a recursive manner, similar to other decision tree algorithms, such as C4.5 or ID3 [30]. REPTree is a useful algorithm for classification tasks, particularly when the goal is to reduce overfitting and improve generalization. It is often applied in various domains, including healthcare, finance, and social sciences.

4.3 Ensemble Techniques

The objective is to aggregate the prediction results of various learning approaches so that the overall performance of the decision is enhanced [24]. The ensemble model improves the performance of the individual model for example it improves the performance of the decision tree by reducing variance in the model. They are classified as either homogeneous or heterogeneous ensembles. In a homogeneous ensemble, similar type of learning techniques like bagging, boosting, and others are employed [13]. Different types of learning techniques are used in heterogeneous ensembles. We built a defect prediction model using voting, stacking, bagging, and boosting in this study.

Ensemble techniques are machine learning methods that combine the predictions of multiple individual models, known as base models or weak learners, to improve the overall predictive performance [24]. By leveraging the diversity and collective wisdom of multiple models, ensemble techniques aim to achieve better generalization, reduce overfitting, and enhance prediction accuracy. The categories of ensemble techniques are depicted in fig 4.2.

Ensemble techniques are widely used in various domains and have achieved success in competitions and real-world applications. Machine learning tasks, such as classification, regression, and anomaly detection, can benefit from the application of these techniques across a broad spectrum. The choice of the ensemble technique depends on the specific problem, data characteristics, and desired tradeoffs between performance and interpretability.

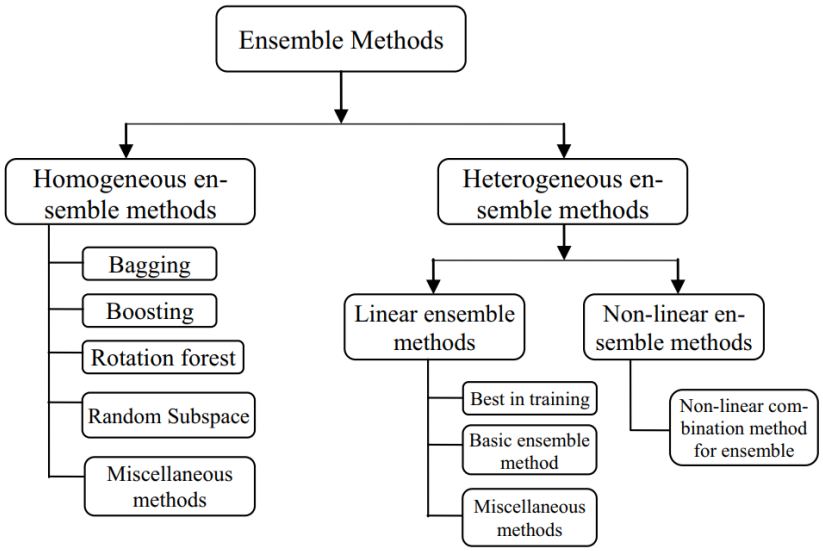

Fig 4.2 Types of Ensemble Techniques

NB, SVM, KNN, LR, and DT (Reduced Error Pruning Tree) are used as base classification techniques to generated ensemble model. Here are the selected ensemble techniques we have opted for implementation:

4.3.1 Stacking

Stacking, in the context of machine learning, refers to a technique where multiple models, known as base models or learners, are aggregated to improve predictive results. It is a form of ensemble learning, which leverages the strengths of different models to make more accurate predictions.

In a stacking ensemble, the base models are trained on the same dataset, and their predictions are then combined using another model called a meta-learner or a stacking model. The meta-learner takes the predictions of the base models as input and learns how to best combine them to produce the final prediction. The stacking architecture is depicted in figure 4.3.

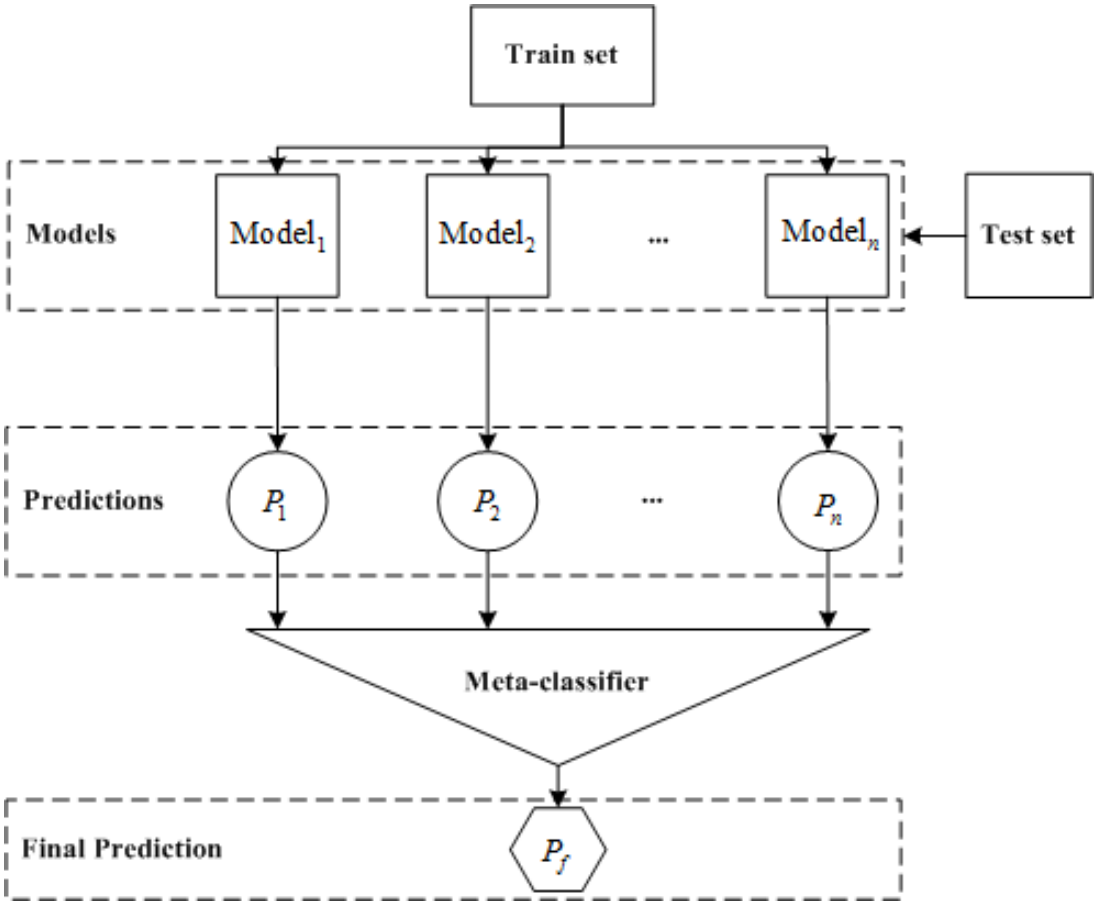

Fig 4.3 Structure of Stacking Technique

4.3.2 Voting

Voting, in the context of machine learning, refers to a technique where multiple models or classifiers are used to make predictions, and the final prediction is determined by aggregating the individual predictions through a voting process. It is another form of ensemble learning, which aims to improve the overall accuracy and robustness of predictions. Voting is classified into two types:

1. **Hard Voting**

   Hard voting, also called deterministic voting, considers only the class labels predicted by each model. The final prediction is determined by selecting the class label that occurs most frequently among the models. The architecture of hard voting is depicted in fig 4.4.

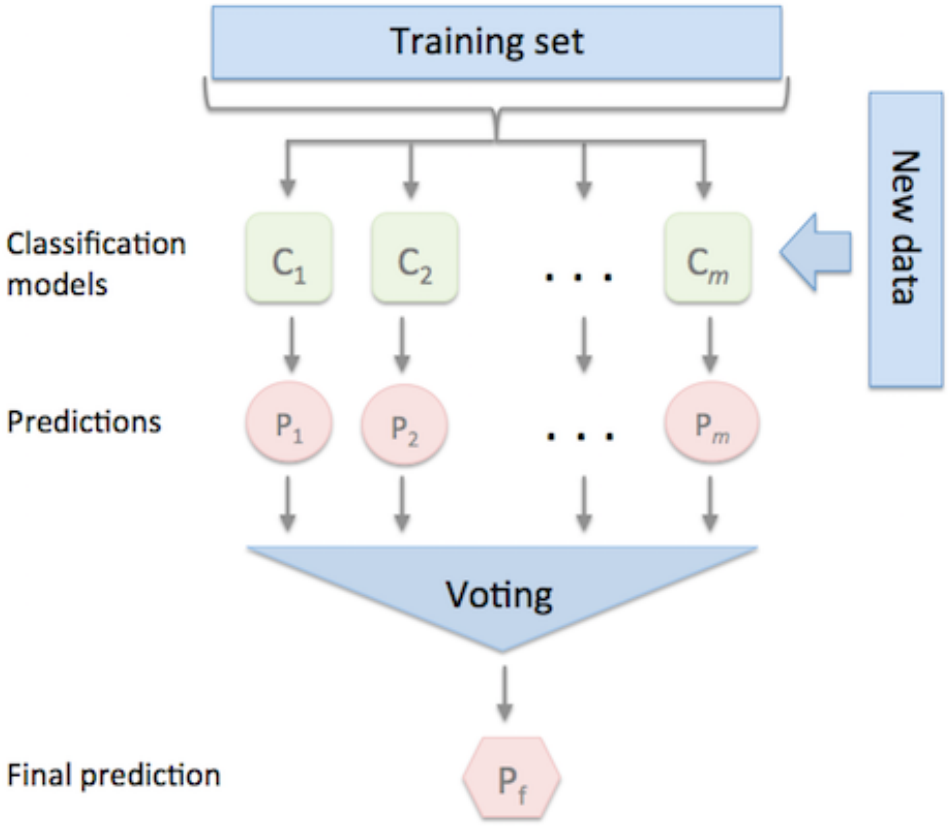

Fig 4.4 Structure of Hard Voting Technique

2. **Soft Voting**

Soft voting, also known as probabilistic voting, takes into account the probabilities or confidence scores assigned by each model for each class label. The final prediction is determined by averaging the probabilities across all models and selecting the class label with the highest average probability. Fig 4.5 shows the architecture of soft voting.

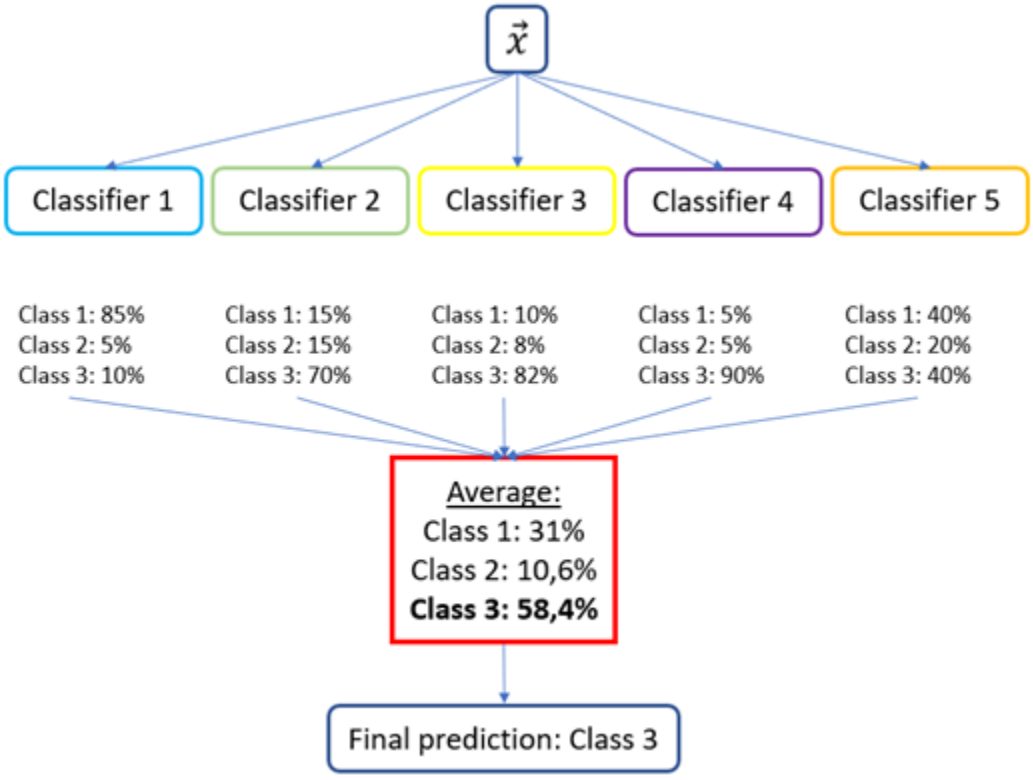

Fig 4.5 Structure of Soft Voting Technique

4.3.3 Bagging

Bagging, short for bootstrap aggregating, is a machine learning approach that involves training multiple models using distinct subsets of the training data and then merging their predictions to form a final prediction. It is a popular ensemble learning method that aims to improve the accuracy and stability of predictions. The bagging architecture is depicted in figure 4.6.

The bagging process typically involves the following steps:

- **Bootstrap Sampling:** The training dataset is randomly sampled with replacement to create multiple subsets of data, called bootstrap samples. Each bootstrap sample has the same size as the original dataset, but some instances may be repeated while others may be excluded.
- **Base Model Training:** A base model, such as a decision tree or a neural network, is trained on each bootstrap sample independently. Each model in the ensemble learns from a slightly different version of the training data.

- **Prediction Aggregation:** Once the base models are trained, they are used to make predictions on new, unseen data. For classification tasks, the final prediction is often determined by majority voting, where the class label that receives the most votes among the base models is selected. For regression tasks, the predictions from the base models are typically averaged to obtain the final prediction.

Some popular bagging algorithms include Random Forests, which use decision trees as base models, and Bagging of Neural Networks, which employ neural networks as base models.

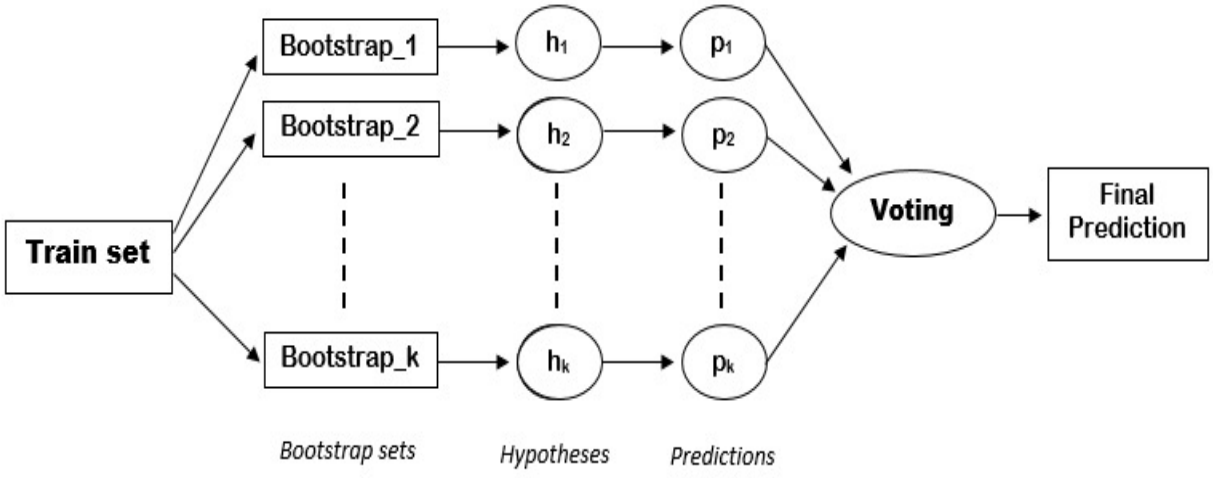

Fig 4.6 Structure of Bagging Technique

4.3.4 Boosting

Boosting is a machine learning technique that combines multiple weak models, often referred to as weak learners or base learners, to create a strong predictive model. Unlike bagging, where base models are trained independently, boosting sequentially trains base models in an adaptive manner, with each subsequent model focusing on the instances that were misclassified by the previous models. The overall goal of boosting is to improve the predictive accuracy by giving more weight to the instances that are difficult to classify correctly. Commonly used boosting algorithms are Adaboost, XGBoost, Gradient Boosting, and LightGBM. Here, we use Adaboost algorithms. The boosting architecture is depicted in Figure 4.7.

The boosting process typically involves the following steps:

- **Base Model Training:** A weak base model, such as a decision tree or a simple linear model, is trained on the initial dataset. The weak model typically performs slightly better than random guessing.
- **Instance Weighting:** Each instance in the training dataset is assigned an initial weight. Initially, all instances have equal weights, but as the boosting process progresses, the weights are adjusted based on the performance of the previous models.
- **Sequential Model Training:** The base model is trained on the weighted training data. The model focuses on the instances that were misclassified or have higher weights, aiming to improve the accuracy for those instances.
- **Instance Weight Update:** After training the base model, the weights of the instances are updated based on their classification results. Misclassified instances are assigned higher weights, making them more influential in the subsequent model training. Correctly classified instances may have their weights reduced.
- **Model Combination:** The base models are combined by assigning weights to their predictions. The weights are typically determined based on the performance of each model during training. The final prediction is made by aggregating the weighted predictions of all the base models.
- **Final Prediction:** The boosted model is then used to make predictions on new, unseen data. The base models generate predictions, and these predictions are combined using the assigned weights to produce the final prediction.

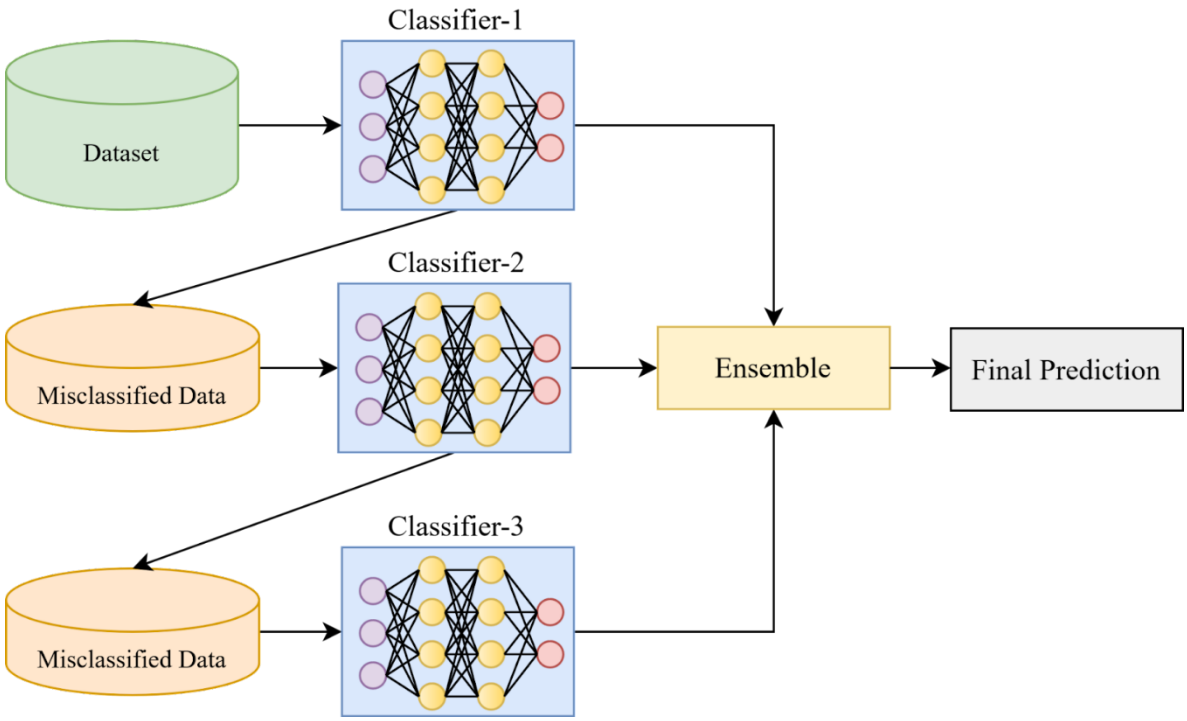

Fig 4.7 Structure of Boosting Technique

4.4 Performance Evaluation Measure

We employ AUC, to evaluate the prediction performance of machine learning techniques and ensemble techniques. The AUC is a valuable metric for evaluating classification models, especially in scenarios where class imbalance exists or when the cost of false positives and false negatives is not equal [12][14]. A higher AUC suggests better discriminative power and the ability of the model to correctly rank instances from positive and negative classes. The AUC serves as a summary of the ROC curve, illustrating the relationship between two parameters, namely TPR and FPR. We can find values of TPR and FPR with the help of a confusion matrix. A confusion matrix is a table-like presentation that summarizes the results of a classification model's performance on a specific set of test data. It offers a comprehensive breakdown of the predicted class labels compared to the actual class labels, allowing for the evaluation of different aspects of the models performance. Fig 4.8 represents confusion matrix.

$$TPR = \frac{TP}{TP+FN} \quad (4.2)$$

$$FPR = \frac{FP}{FP+TN} \quad (4.3)$$

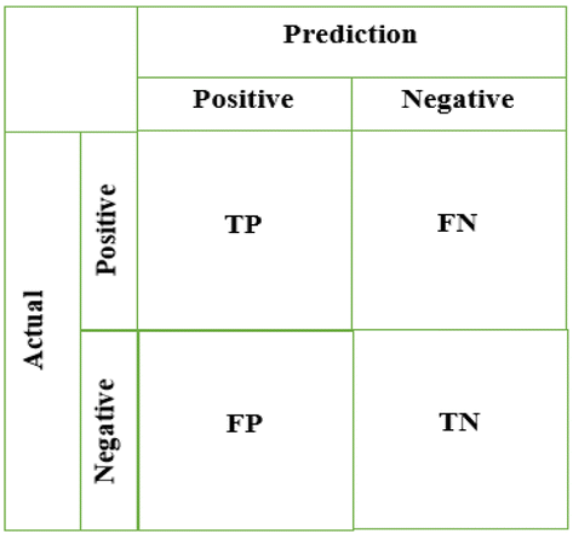

Fig 4.8 Confusion matrix

The AUC represents the area under the ROC curve, which ranges from 0 to 1. A higher AUC value indicates better performance of the classifier in distinguishing between positive and negative instances. An AUC of 1 represents a perfect classifier, while an AUC of 0.5 indicates a classifier that performs no better than random guessing. Dejaeger *et al* [14] presented a detailed description on how to calculate AUC value.

4.5 Statistical Test

We employ Friedman test with Nemenyi test to check whether the prediction results are statistically significant or not [11]. Friedman test is a non parametric test which indicates that the data don't need to be normal. Ranking system is used in this test. This test assigns ranks to the chosen techniques. Nemenyi test is a post hoc test which is used if there is a rejection of null hypothesis. This test is used to identify the pairwise difference between the techniques.

- **Friedman Test**

The Null and alternative hypothesis statements for the Friedman test can be restated as follows:

- Null Hypothesis (H_0): There is no significant difference in the performances of the various techniques.

- Alternative Hypothesis (H_a): There is a significant difference in the performances of the various techniques.

The Friedman test is a non-parametric statistical test used to determine whether there are significant differences among multiple treatments or conditions [19]. It is typically applied when the data are measured on an ordinal scale and assumptions for parametric tests, such as normality or equal variances, are violated. It's important to note that the Friedman test assesses the overall differences among the treatments but does not provide information about the direction or nature of the differences.

In this test, we compare the calculated $\boldsymbol{\chi}^2$-statistics value with the tabulated $\boldsymbol{\chi}^2$ value. 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**

The Nemenyi test, also known as the Nemenyi-Damico-Wolfe-Dunn test, is a post-hoc test commonly used in conjunction with the Friedman test or other non-parametric tests for multiple comparisons. It helps identify specific pairs of treatments or conditions that differ significantly from each other after finding a significant overall difference among the groups. The Nemenyi test provides a pairwise comparison of treatments, allowing researchers to identify specific treatments that differ significantly from each other. It is commonly used when there are more than two treatments or conditions and the goal is to determine which treatments are significantly different, rather than just identifying the overall differences. In this, we calculate CD.

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

- **Effect Size**

Effect size is a statistical measure that quantifies the magnitude or strength of a relationship or difference between variables or groups [19]. Effect size is useful because it allows researchers to move beyond statistical significance and assess the practical or real-world significance of their findings. While statistical significance indicates whether an observed effect is likely to have occurred by chance, effect size provides information about the magnitude or impact of the effect.

Commonly used effect size measure are Cohen's d, Kendall's W, Pearson's r, etc. In this study, Kendall's W coefficient (also known as the Coefficient of Cordonance) is utilized. Its calculation is as follows:

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

Kendall's W coefficient ranges from 0 to 1, with higher values indicating greater agreement among the techniques. The coefficient measures the extent to which the rankings assigned by different techniques are consistent or concordant. A coefficient of 1 indicates perfect agreement, meaning that all techniques have assigned the same rankings. A coefficient of 0 indicates no agreement, meaning that the rankings assigned by the techniques are completely random.

CHAPTER 5

EXPERIMENTAL RESULTS

This section demonstrates the experimental result found after applying classification and ensemble learning techniques to all the considered datasets. Note that we have used AUC to evaluate the prediction performance of chosen techniques. We analyze three models to check the effectiveness of process metrics on prediction performance of classification and ensemble method:

- Combined model of SC and process metrics
- Model containing SC metrics
- Model containing process metrics

Table 5.1 represents the AUC values for all considered datasets for all classification (NB, KNN, DT, SVM, LR) and ensemble learning techniques (stacking, voting, bagging, and boosting) in combined model. According to table 5.1, ensemble techniques outperform classification techniques and bagging gives the better prediction results as compared to all the selected techniques.

Table 5.1 AUC values in combined model

	NB	KNN	DT	SVM	LR	Stacking	Voting	Bagging	Boosting
Ant 1.6	0.862	0.691	0.813	0.676	0.725	0.864	0.872	0.894	0.869
Ant 1.7	0.889	0.719	0.846	0.694	0.726	0.872	0.877	0.900	0.894
jEdit 4.0	0.902	0.753	0.817	0.735	0.694	0.894	0.890	0.917	0.908
jEdit 4.1	0.926	0.746	0.873	0.740	0.612	0.913	0.916	0.938	0.902
Xalan 2.6.0	0.865	0.817	0.872	0.784	0.833	0.898	0.899	0.906	0.899
Xalan 2.7.0	0.973	0.991	0.987	0.981	0.982	0.993	0.997	0.997	0.993
Xerces 1.3.0	0.830	0.679	0.763	0.594	0.604	0.756	0.818	0.843	0.819
Xerces 1.4.4	0.806	0.893	0.909	0.849	0.853	0.942	0.943	0.959	0.949

Table 5.2 represents the AUC values for all considered datasets for all classification (NB, KNN, DT, SVM, LR) and ensemble learning techniques (stacking, voting, bagging, and boosting) in model containing SC metrics. According to table 5.2, ensemble techniques

outperform classification techniques and bagging gives the better prediction results as compared to all the selected techniques.

Table 5.2 AUC values in model containing SC metrics

	NB	KNN	DT	SVM	LR	Stacking	Voting	Bagging	Boosting
Ant 1.6	0.841	0.728	0.809	0.606	0.848	0.857	0.857	0.868	0.871
Ant 1.7	0.846	0.719	0.810	0.620	0.856	0.850	0.862	0.875	0.873
jEdit 4.0	0.826	0.774	0.717	0.538	0.819	0.821	0.845	0.850	0.857
jEdit 4.1	0.837	0.710	0.783	0.609	0.856	0.815	0.855	0.881	0.844
Xalan 2.6.0	0.822	0.801	0.848	0.695	0.844	0.867	0.869	0.876	0.875
Xalan 2.7.0	0.932	0.924	0.914	0.843	0.956	0.968	0.963	0.967	0.967
Xerces 1.3.0	0.786	0.759	0.712	0.519	0.751	0.777	0.794	0.831	0.801
Xerces 1.4.4	0.755	0.847	0.893	0.766	0.867	0.918	0.897	0.905	0.904

Table 5.3 represents the AUC values for all considered datasets for all classification (NB, KNN, DT, SVM, LR) and ensemble learning techniques (stacking, voting, bagging, and boosting) in model containing process metrics. According to table 5.3, ensemble techniques outperform classification techniques and bagging gives the better prediction results as compared to all the selected techniques.

Table 5.3. AUC values in model containing process metrics

	NB	KNN	DT	SVM	LR	Stacking	Voting	Bagging	Boosting
Ant 1.6	0.840	0.778	0.771	0.647	0.589	0.850	0.856	0.859	0.843
Ant 1.7	0.876	0.785	0.814	0.669	0.698	0.856	0.877	0.877	0.857
jEdit 4.0	0.914	0.868	0.871	0.748	0.687	0.888	0.911	0.920	0.901
jEdit 4.1	0.906	0.820	0.790	0.723	0.712	0.911	0.908	0.906	0.877
Xalan 2.6.0	0.810	0.811	0.848	0.684	0.771	0.824	0.832	0.834	0.830
Xalan 2.7.0	0.991	0.990	0.989	0.981	0.989	0.991	0.989	0.992	0.992
Xerces 1.3.0	0.650	0.614	0.603	0.548	0.552	0.719	0.711	0.731	0.657
Xerces 1.4.4	0.713	0.812	0.795	0.679	0.764	0.817	0.817	0.823	0.822

According to table 5.1, table 5.2, and table 5.3, we can analyze that all the selected techniques except logistic regression and KNN performs better in combined model. Therefore, we proceed to examine three additional models, utilizing only the techniques that demonstrate superior performance in the combined model.

- Combined model of SC and 1 process metric
- Combined model of SC and 2 process metrics
- Combined model of SC and 3 process metrics

By employing these three additional models, we can determine which process metric is more effective in predicting the defect proneness.

Table 5.4 represents the AUC values for all considered datasets for NB technique in combined model of static code and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of SC and process metrics as in 87.5% of cases, combining NR with all SC metrics yields superior AUC values.

Table 5.4 AUC values for NB in combined model of static code and 1 process metric

	SC+NR	SC+NDC	SC+NML	SC+NDPV
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
Xalan2.6.0	0.863	0.834	0.858	0.833
Xalan2.7.0	0.977	0.952	0.953	0.941
Xerces1.3.0	0.822	0.817	0.809	0.796
Xerces1.4.4	0.828	0.789	0.784	0.756

Table 5.5 represents the AUC values for all considered datasets for NB technique in combined model of SC and 2 process metrics. It can be observed that combined mode of static code, NDC, and NR outperforms other combinations of SC and process metrics as in 87.5% of cases, combining NR and NDC with all SC metrics yields superior AUC values.

Table 5.5 AUC values for NB in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 5.6 represents the AUC values for all considered datasets for NB technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NDC, NR, and NDPV outperforms other combinations of static code and process metrics as in all the considered datasets, combining NR, NDPV, and NDC metrics with all SC metrics yields superior AUC values.

Table 5.6 AUC values for NB in combined model of static code and 3 process metrics

	SC+NDC+ NR+NML	SC+NDC+ NR+NDPV	SC+NML+ NR+NDPV	SC+NML+ NDC+NDPV
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
Xalan2.6.0	0.864	0.866	0.865	0.864
Xalan2.7.0	0.972	0.983	0.969	0.960
Xerces1.3.0	0.825	0.845	0.814	0.842
Xerces1.4.4	0.808	0.837	0.794	0.801

Table 5.7 represents the AUC values for all considered datasets for SVM technique in combined model of SC and 1 process metric. It can be observed that combined model of

static code and NR outperforms other combinations of SC and process metrics as in 62.5% of cases, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.7 AUC values for SVM in combined model of static code and 1 process metric

	SC+NR	SC+NDC	SC+NML	SC+NDPV
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
Xalan2.6.0	0.774	0.728	0.777	0.719
Xalan2.7.0	0.981	0.881	0.883	0.850
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 represents the AUC values for all considered datasets for SVM technique in combined model of SC and 2 process metrics. It can be observed that combined model of static code, NR, and NML outperforms other combinations of SC and process metrics as in 75% of cases, combining NR and NML metrics with all SC metrics yields superior AUC values.

Table 5.8 AUC values for SVM in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 represents the AUC values for all considered datasets for SVM technique in combined model of SC and 3 process metrics. It can be observed that combined model of

static code, NR, NML, and NDPV outperforms other combinations of SC and process metrics as in 75% of cases, combining NR, NML, and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.9 AUC values for SVM in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	SC+NDC+NR +NDPV	SC+NML+NR +NDPV	SC+NML+NDC +NDPV
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
Xalan2.6.0	0.784	0.774	0.786	0.779
Xalan2.7.0	0.981	0.981	0.981	0.883
Xerces1.3.0	0.600	0.591	0.594	0.552
Xerces1.4.4	0.849	0.846	0.851	0.846

Table 5.10 represents the AUC values for all considered datasets for DT technique in combined model of SC and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of static code and process metrics as in 50% of cases, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.10 AUC values for DT in combined model of static code and 1 process metric

	SC+NR	SC+NDC	SC+NML	SC+NDPV
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
Xalan2.6.0	0.868	0.865	0.862	0.837
Xalan2.7.0	0.987	0.961	0.958	0.947
Xerces1.3.0	0.710	0.693	0.757	0.689
Xerces1.4.4	0.939	0.902	0.892	0.863

Table 5.11 represents the AUC values for all considered datasets for DT technique in combined model of SC and 2 process metrics. It can be observed that combined model of

SC, NR & NDC and combined model of SC, NDC & NDPV outperforms other combinations of SC and process metrics as in 37.5% of cases, combining NR and NDC with all SC metrics yields superior AUC values and also in other 37.5% of cases, combining NDC and NDPV with all SC metrics yields superior AUC values.

Table 5.11 AUC values for DT in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 5.12 represents the AUC values for all considered datasets for DT technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NR, NML, and NDPV outperforms other combinations of SC and process metrics as in 62.5% of cases, combining NR, NML and NDPV with all SC metrics yields superior AUC values.

Table 5.12 AUC values for DT in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	SC+NDC+NR +NDPV	SC+NML+NR +NDPV	SC+NML+NDC +NDPV
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
Xalan2.6.0	0.848	0.854	0.873	0.868

Xalan2.7.0	0.989	0.985	0.992	0.951
Xerces1.3.0	0.718	0.737	0.740	0.731
Xerces1.4.4	0.933	0.941	0.933	0.890

Table 5.13 represents the AUC values for all considered datasets for stacking technique in combined model of SC and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of SC and process metrics as in 62.5% of cases, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.13 AUC values for stacking in combined model of static code and 1 process metric

	SC+NR	**SC+NDC**	**SC+NML**	**SC+NDPV**
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
Xalan2.6.0	0.900	0.885	0.891	0.880
Xalan2.7.0	0.994	0.981	0.979	0.970
Xerces1.3.0	0.859	0.849	0.769	0.794
Xerces1.4.4	0.950	0.934	0.938	0.913

Table 5.14 represents the AUC values for all considered datasets for stacking technique in combined model of SC and 2 process metrics. It can be observed that combined model of SC, NDC, and NDPV outperforms other combinations of SC and process metrics as in 50% of cases, combining NDC and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.14 AUC values for stacking in combined model of static code and 2 process metrics

	SC+NR +NDC	**SC+NR +NML**	**SC+NR +NDPV**	**SC+NDC +NML**	**SC+NDC +NDPV**	**SC+NML +NDPV**
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
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 5.15 represents the AUC values for all considered datasets for stacking technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NDC, NR, and NDPV outperforms other combinations of SC and process metrics as in 75% of cases, combining NR, NDC and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.15 AUC values for stacking in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	**SC+NDC+NR +NDPV**	**SC+NML+NR +NDPV**	**SC+NML+NDC +NDPV**
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
Xalan2.6.0	0.899	0.900	0.900	0.894
Xalan2.7.0	0.993	0.993	0.995	0.981
Xerces1.3.0	0.754	0.844	0.775	0.768
Xerces1.4.4	0.949	0.950	0.948	0.938

Table 5.16 represents the AUC values for all considered datasets for voting technique in combined model of SC and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of SC and process metrics as in all of the considered datasets, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.16 AUC values for voting in combined model of static code and 1 process metric

	SC+NR	SC+NDC	NML+SC	NDPV+SC
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
Xalan2.6.0	0.897	0.878	0.888	0.882
Xalan2.7.0	0.995	0.979	0.979	0.971
Xerces1.3.0	0.871	0.843	0.779	0.810
Xerces1.4.4	0.954	0.939	0.937	0.902

Table 5.17 represents the AUC values for all considered datasets for voting technique in combined model of SC and 2 process metrics. It can be observed that combined model of SC, NR, and NDC outperforms other combinations of SC and process metrics as in 62.5% of cases, combining NR and NDC metrics with all SC metrics yields superior AUC values.

Table 5.17 AUC values for voting in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 5.18 represents the AUC values for all considered datasets for voting technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NDC, NR, and NDPV outperforms other combinations of SC and process metrics as in 87.5% of cases, combining NR, NDC and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.18 AUC values for voting in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	SC+NDC+NR +NDPV	SC+NML+NR +NDPV	SC+NML+NDC +NDPV
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
Xalan2.6.0	0.898	0.901	0.900	0.888
Xalan2.7.0	0.997	0.995	0.997	0.982
Xerces1.3.0	0.798	0.863	0.808	0.794
Xerces1.4.4	0.949	0.951	0.945	0.936

Table 5.19 represents the AUC values for all considered datasets for bagging technique in combined model of SC and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of SC and process metrics as in all of the considered datasets, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.19 AUC values for bagging in combined model of static code and 1 process metric

	SC+NR	SC+NDC	SC+NML	SC+NDPV
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
Xalan2.6.0	0.904	0.890	0.899	0.887
Xalan2.7.0	0.996	0.979	0.980	0.971
Xerces1.3.0	0.870	0.868	0.827	0.839
Xerces1.4.4	0.956	0.943	0.947	0.911

Table 5.20 represents the AUC values for all considered datasets for bagging technique in combined model of SC and 2 process metrics. It can be observed that combined model of SC, NDC, and NR outperforms other combinations of SC and process metrics as in 75% of cases, combining NR and NDC metrics with all SC metrics yields superior AUC values.

Table 5.20 AUC values for bagging in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 5.21 represents the AUC values for all considered datasets for bagging technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NDC, NDPV and NR outperforms other combinations of SC and process metrics as in 62.5% of cases, combining NR, NDC and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.21 AUC values for bagging in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	SC+NDC+NR +NDPV	SC+NML+NR +NDPV	SC+NML+NDC +NDPV
Ant1.6	0.893	0.898	0.890	0.871
Ant1.7	0.899	0.898	0.894	0.888
Jedit4.0	0.910	0.920	0.912	0.898
Jedit4.1	0.938	0.947	0.937	0.928
Xalan2.6.0	0.907	0.905	0.906	0.900
Xalan2.7.0	0.998	0.998	0.997	0.982
Xerces1.3.0	0.841	0.867	0.835	0.859
Xerces1.4.4	0.961	0.955	0.958	0.946

Table 5.22 represents the AUC values for all considered datasets for boosting technique in combined model of SC and 1 process metric. It can be observed that combined model of SC and NR outperforms other combinations of SC and process metrics as in 87.5% of cases, combining NR metrics with all SC metrics yields superior AUC values.

Table 5.22 AUC values for boosting in combined model of static code and 1 process metric

	SC+NR	SC+NDC	SC+NML	SC+NDPV
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
Xalan2.6.0	0.900	0.881	0.892	0.880
Xalan2.7.0	0.995	0.982	0.982	0.973
Xerces1.3.0	0.822	0.855	0.780	0.825
Xerces1.4.4	0.954	0.941	0.939	0.901

Table 5.23 represents the AUC values for all considered datasets for boosting technique in combined model of SC and 2 process metrics. It can be observed that combined model of SC, NDPV and NR outperforms other combinations of SC and process metrics as in 50% of cases, combining NR and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.23 AUC values for boosting in combined model of static code and 2 process metrics

	SC+NR +NDC	SC+NR +NML	SC+NR +NDPV	SC+NDC +NML	SC+NDC +NDPV	SC+NML +NDPV
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
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
Xerces1.3.0	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 5.24 represents the AUC values for all considered datasets for boosting technique in combined model of SC and 3 process metrics. It can be observed that combined model of SC, NDC, NDPV, and NR outperforms other combinations of SC and process metrics as in 87.5% of cases, combining NR, NDC, and NDPV metrics with all SC metrics yields superior AUC values.

Table 5.24 AUC values for boosting in combined model of static code and 3 process metrics

	SC+NDC+NR +NML	SC+NDC+NR +NDPV	SC+NML+NR +NDPV	SC+NML+NDC +NDPV
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
Xalan2.6.0	0.902	0.902	0.898	0.891
Xalan2.7.0	0.994	0.996	0.996	0.983
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

The predictive performance of classification and ensemble methods have been analysed based on whisker plots, bar charts and statistical tests in this section. AUC values for the model containing both the SC metric and process metric have been displayed in Fig 6.1. using the bar charts for all the datasets under consideration. For the classification and ensemble methods NB, KNN, DT, SVM, LR, Stacking, Voting, Bagging, and Boosting considering all the datasets the mean AUC values are 0.882, 0.786, 0.86, 0.756, 0.753, 0.891, 0.901, 0.919, 0.904. The results of Table 5.1 have also been analysed using statistical test such as Friedman test and it can be observed that the performances of the selected methods differ significantly, so post hoc test has been applied for pairwise comparison and it can be incurred from the outcomes that the difference between the performances of SVM and bagging and LR and bagging is significant.

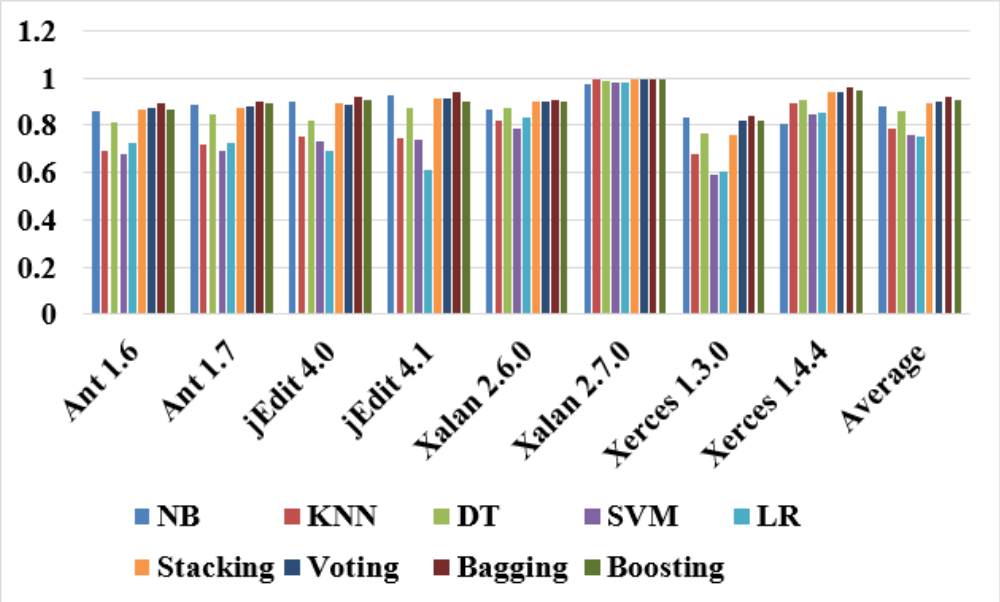

Fig 6.1 Bar chart of AUC values in combined model

AUC values for the model containing SC metrics have been displayed in Fig 6.2 using the bar charts for all the datasets under consideration. For the classification and ensemble methods NB, KNN, DT, SVM, LR, Stacking, Voting, Bagging, and Boosting considering all the datasets the mean AUC values are 0.831, 0.783, 0.811, 0.649, 0.849, 0.859, 0.867, 0.881,

0.874. The results of Table 5.2 have also been analysed using statistical test such as Friedman test and it can be observed that the performances of the selected methods differ significantly, so post hoc test has been applied for pairwise comparison and it can be incurred from the outcomes that the difference between the performances of SVM and bagging and SVM and boosting is significant.

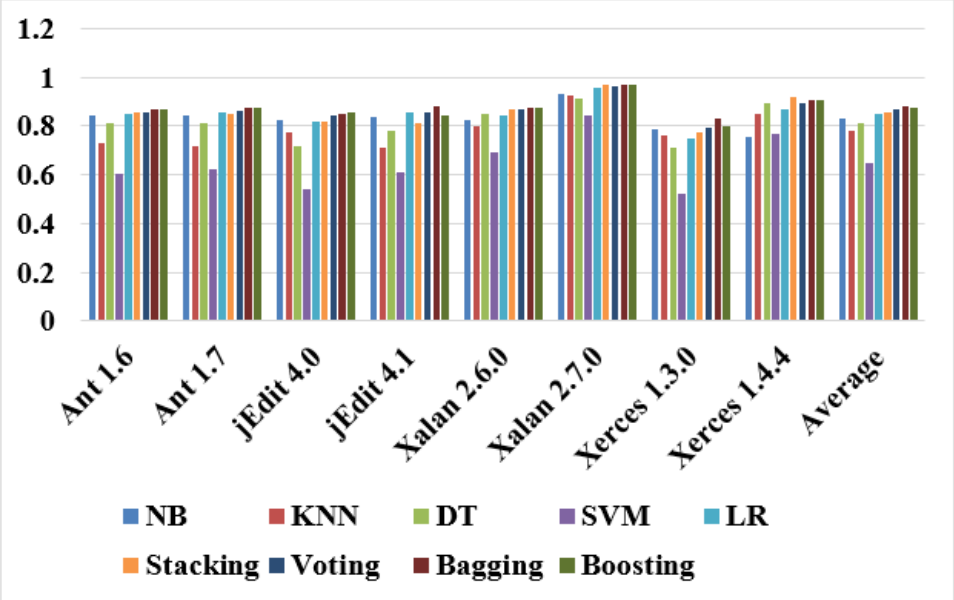

Fig 6.2 Bar chart of AUC values in model containing SC metrics

AUC values for the model containing process metric have been displayed in Fig 6.3 using the bar charts for all the datasets under consideration. For the classification and ensemble methods NB, KNN, DT, SVM, LR, Stacking, Voting, Bagging, and Boosting considering all the datasets the mean AUC values are 0.837, 0.809, 0.810, 0.709, 0.720, 0.857, 0.862, 0.867, 0.847. The results of Table 5.3 have also been analysed using statistical test such as Friedman test and it can be observed that the performances of the selected methods differ significantly, so post hoc test has been applied for pairwise comparison and it can be incurred from the outcomes that the difference between the performances of SVM and bagging and LR and bagging is significant.

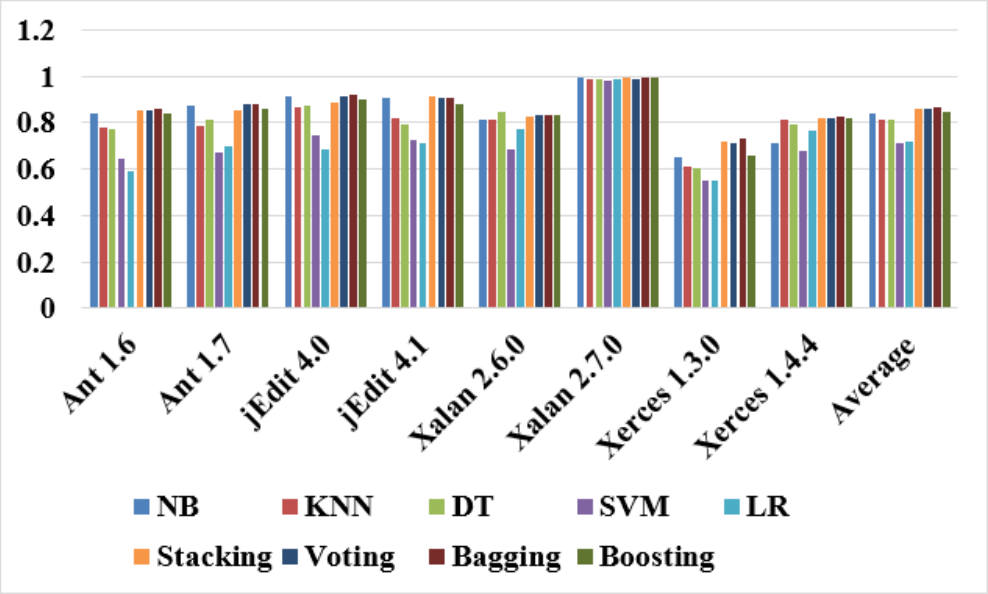

Fig 6.3 Bar chart of AUC values in model containing process metrics

We conducted an analysis on various models to assess the impact of process metrics on defect proneness. Different classifiers and ensemble techniques were employed for this purpose. Furthermore, we explored additional models within each classifier and ensemble technique to determine which combination of process metrics yields improved prediction performance. The experimental results indicate that the combination model produces superior predictions when utilizing NB, DT, and SVM classifiers. Moreover, the results from all ensemble techniques demonstrate that the combination model leads to improved prediction outcomes. Notably, the process metrics NR, NDC, and NDPV consistently display the highest effectiveness across most analyzed cases. Fig 6.1, fig 6.2, and fig 6.3 indicates that combination of process and static code metrics outperform the individual model of static code metrics and process metrics. In this section, we utilize bar charts and statistical tests to analyze the AUC results, and in cases where the null hypothesis is rejected, we calculate the effect size. Table 5.4 has been analyzed statistically, the outcome shows that the calculated p-value is 0.002 and χ2-statistics is 19.108 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.455 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the

performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR metrics and model containing SC+NDPV & also model containing SC+NML metrics and model containing SC+NDPV will have a significant difference between them. For table 5.4 the bar chart has been represented by Figure 6.4. It can be observed that combination of static code and NR outperforms other combinations of static code and process metrics.

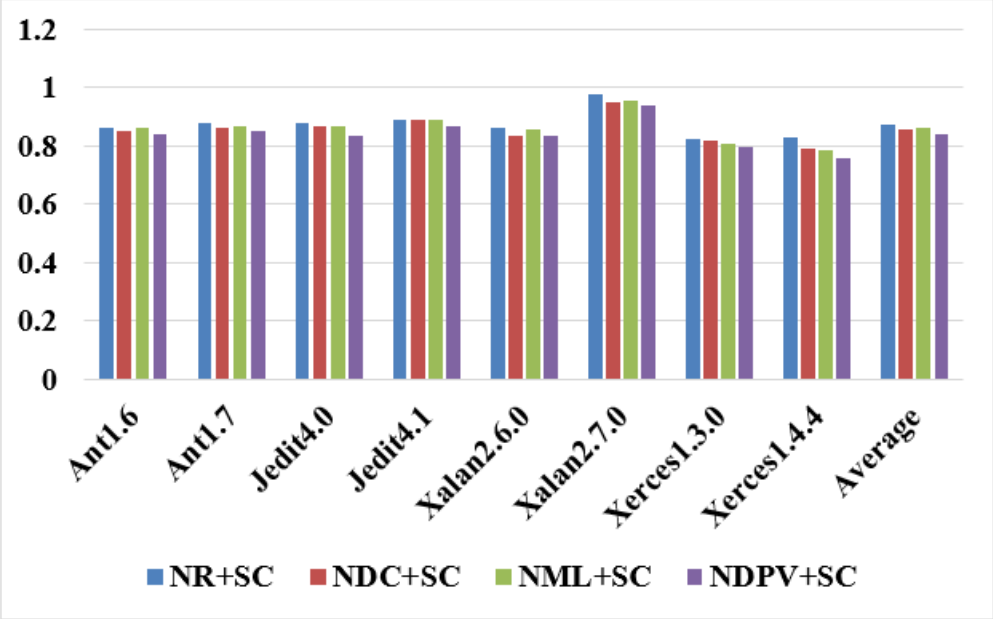

Fig 6.4 AUC values for NB in combined model of static code and 1 process metric

Table 5.5 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0004 and χ2-statistics is 22.37 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.319 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+NDC & model containing NR+SC+NML, model containing NR+SC+NDC & model containing NDC+SC+NDPV, and also model containing NDPV+SC+NR & model containing NDC+SC+NDPV will have a significant difference between them. For table 5.5

the bar chart has been represented by Figure 6.5. It can be observed that combined model of static code, NR and NDC outperforms other combinations of static code and process metrics.

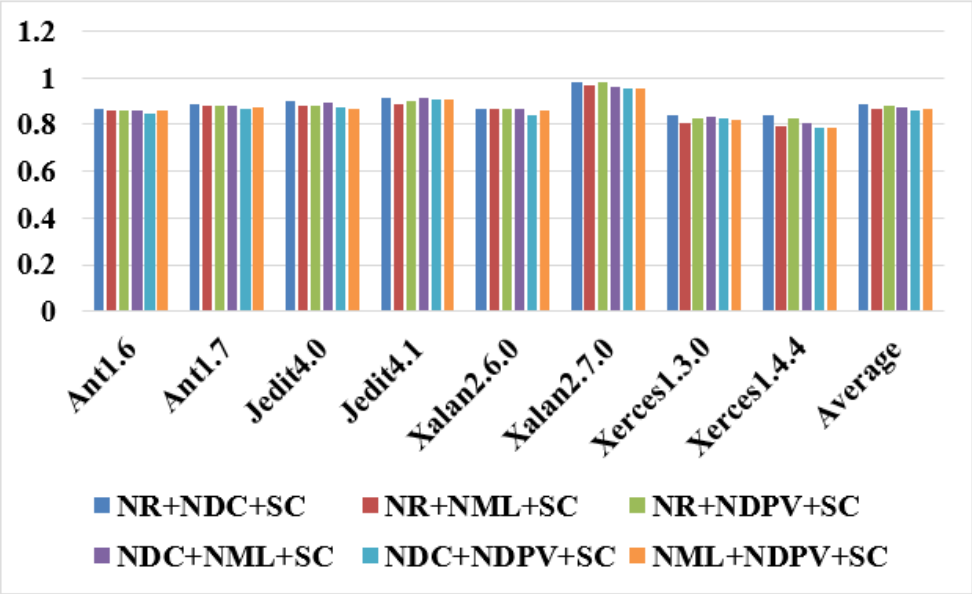

Fig 6.5 AUC values for NB in combined model of static code and 2 process metrics

Table 5.6 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0004 and χ2-statistics is 17.75 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.423 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+NML+SC+NDC & model containing NR+NDPV+SC+NDC, model containing NR+NDPV+SC+NDC & model containing NR+NDPV+SC+NML, and also model containing NR+NDPV+SC+NML & model containing NML+NDPV+SC+NDC will have a significant difference between them. For table 5.6 the bar chart has been represented by Figure 6.6. It can be observed that combined model of static code, NR, NDPV, and NDC outperforms other combinations of static code and process metrics.

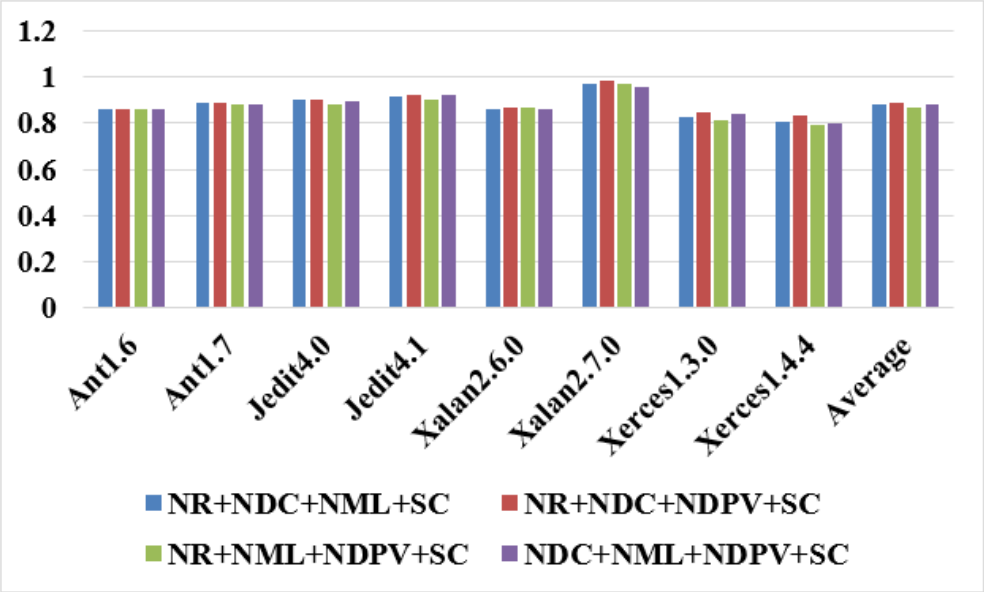

Fig 6.6 AUC values for NB in combined model of static code and 3 process metrics

Table 5.7 has been analyzed statistically, the outcome shows that the calculated p-value is 2.68e-05 and χ2-statistics is 23.84 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.567 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR & model containing SC+NDC, model containing SC+NR & model containing SC+NDPV, and also model containing SC+NML metrics & model containing SC+NDPV will have a significant difference between them. For table 5.7 the bar chart has been represented by Figure 6.7. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

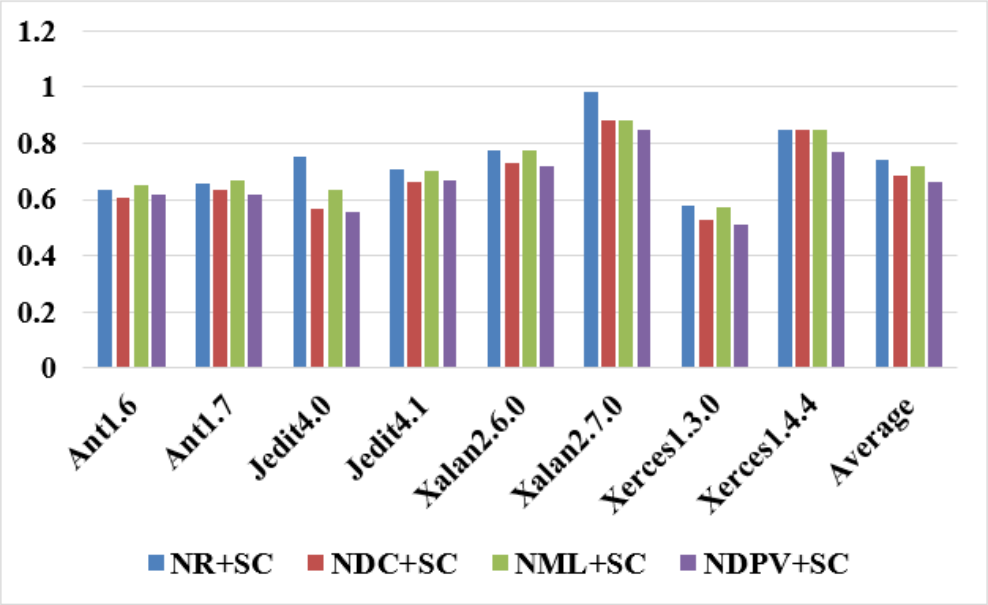

Fig 6.7 AUC values for SVM in combined model of and 1 process metric

Table 5.8 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0004 and χ2-statistics is 22.40 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.32 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+ NML & model containing NDC+SC+NDPV and model containing NR+SC+NDPV & model containing NDC+SC+NDPV will have a significant difference between them. For table 5.8 the bar chart has been represented by Figure 6.8. It can be observed that combined model of static code, NML, and NR outperforms other combinations of static code and process metrics.

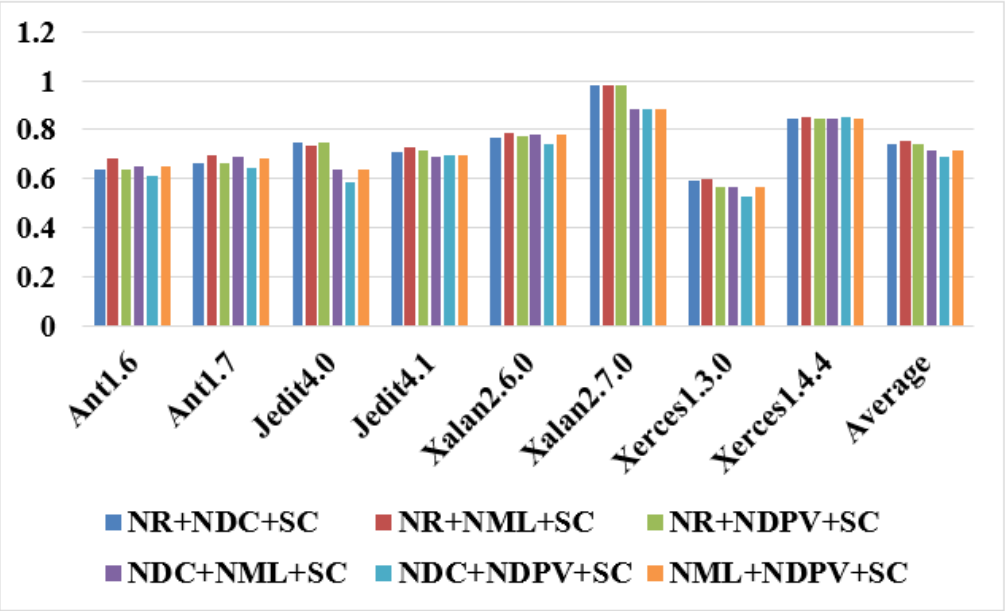

Fig 6.8 AUC values for SVM in combined model of static code and 2 process metrics

Table 5.9 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0032 and χ2-statistics is 13.736 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 2 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected, so with an effect size of 0.32 at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NDPV+NR+NML & model containing SC+NML+NDC+NDPV.

For table 5.9 the bar chart has been represented by Figure 6.9. It can be observed that combined model of static code, NML, NDPV, and NR outperforms other combinations of static code and process metrics.

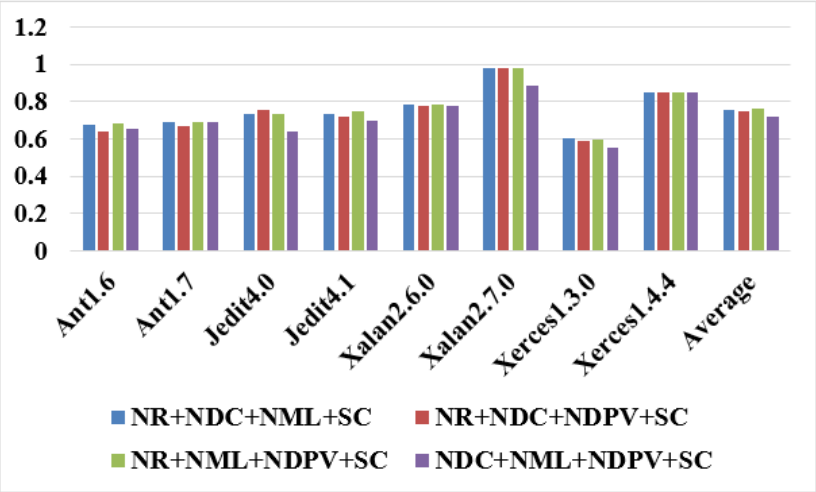

Fig 6.9 AUC values for SVM in combined model of static code and 3 process metrics

Table 5.10 has been analyzed statistically, the outcome shows that the calculated p-value is 0.578 and χ2-statistics is 1.971 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., greater than the χ2-statistics calculated, hence the null hypothesis is accepted. For table 5.10 the bar chart has been represented by Figure 6.10. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

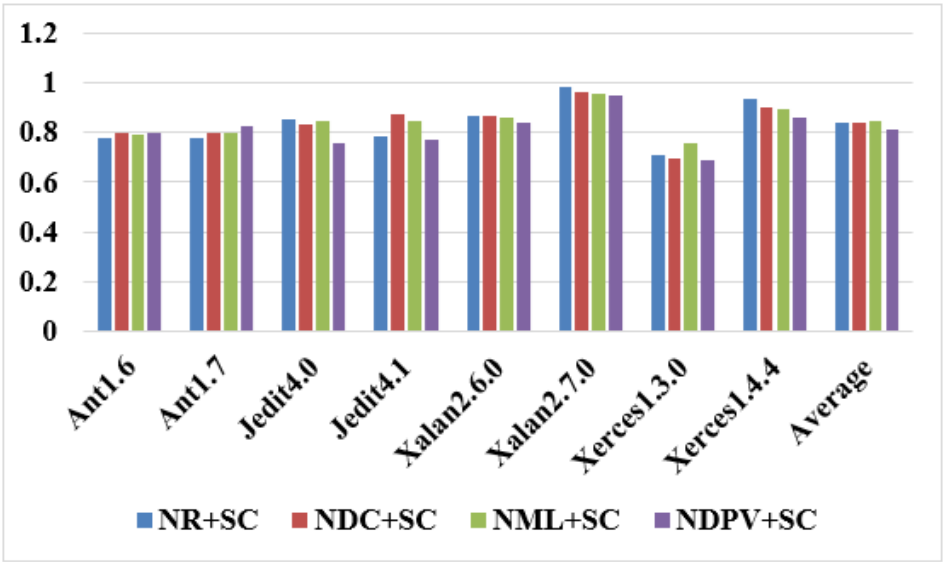

Fig 6.10 AUC values for DT in combined model of static code and 1 process metric

Table 5.11 has been analyzed statistically, the outcome shows that the calculated p-value is 0.3819 and χ2-statistics is 5.286 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., greater than the χ2-statistics calculated, hence the null hypothesis is accepted. For table 5.11 the bar chart has been represented by Figure 6.11. It can be observed that combined model of static code, NDC, and NR outperforms other combinations of static code and process metrics.

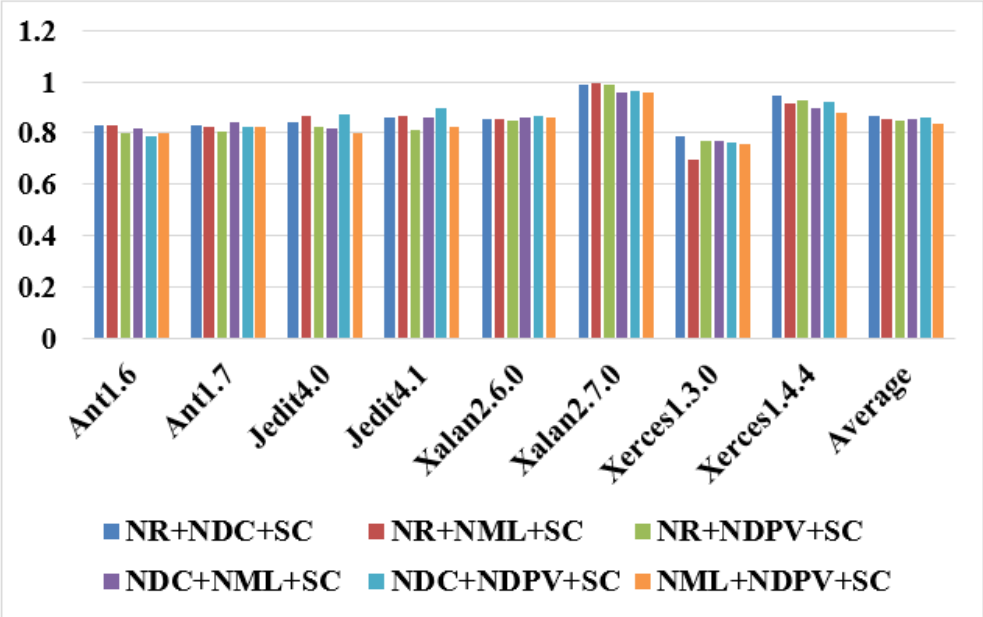

Fig 6.11 AUC values for DT in combined model of static code and 2 process metrics

Table 5.12 has been analyzed statistically, the outcome shows that the calculated p-value is 0.2009 and χ2-statistics is 4.630 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., greater than the χ2-statistics calculated, hence the null hypothesis is accepted. For table 5.12 the bar chart has been represented by Figure 6.12. It can be observed that combined model of static code, NML, NDPV, and NR outperforms other combinations of static code and process metrics.

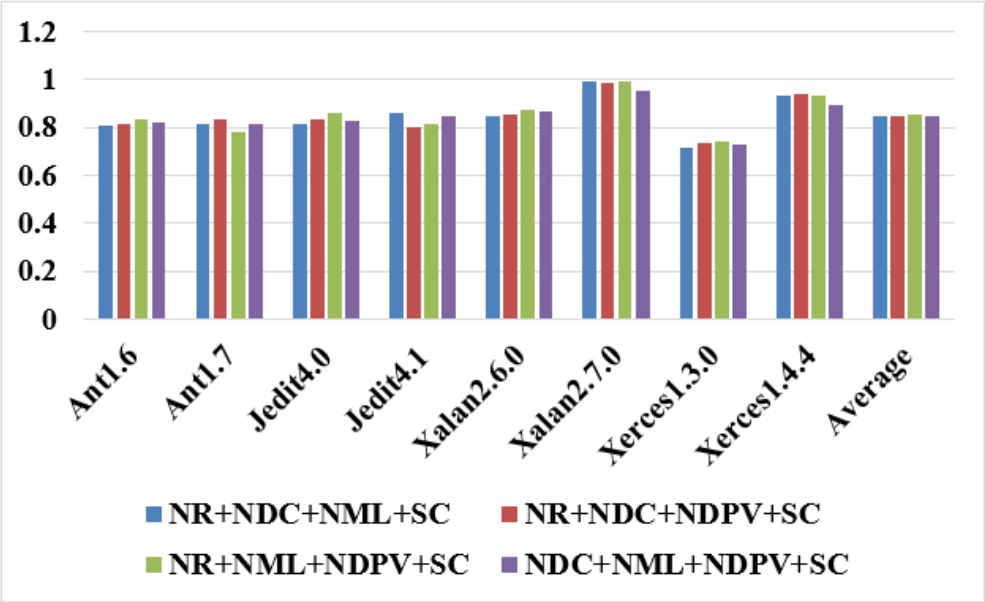

Fig 6.12 AUC values for DT in combined model of static code and 3 process metrics

Table 5.13 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0015 and χ2-statistics is 15.345 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.365, at least one of the model's performances differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR & model containing SC+NML and model containing SC+NDC & model containing SC+NML will have a significant difference between them. For table 5.13 the bar chart has been represented by Figure 6.13. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

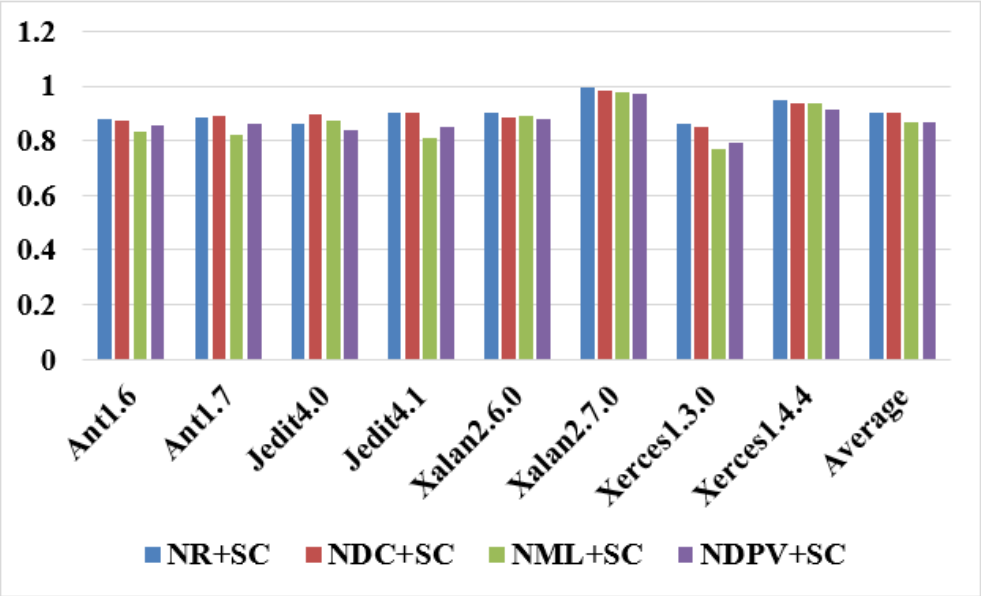

Fig 6.13 AUC values for stacking in combined model of static code and 1 process metric

Table 5.14 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0006 and χ2-statistics is 21.567 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.308, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+NDPV & model containing NML+SC+NDPV and model containing NDC+SC+NDPV & model containing NML+SC+NDPV will have a significant difference between them. For table 5.14 the bar chart has been represented by Figure 6.14. It can be observed that combined model of static code, NDPV, and NDC outperforms other combinations of static code and process metrics.

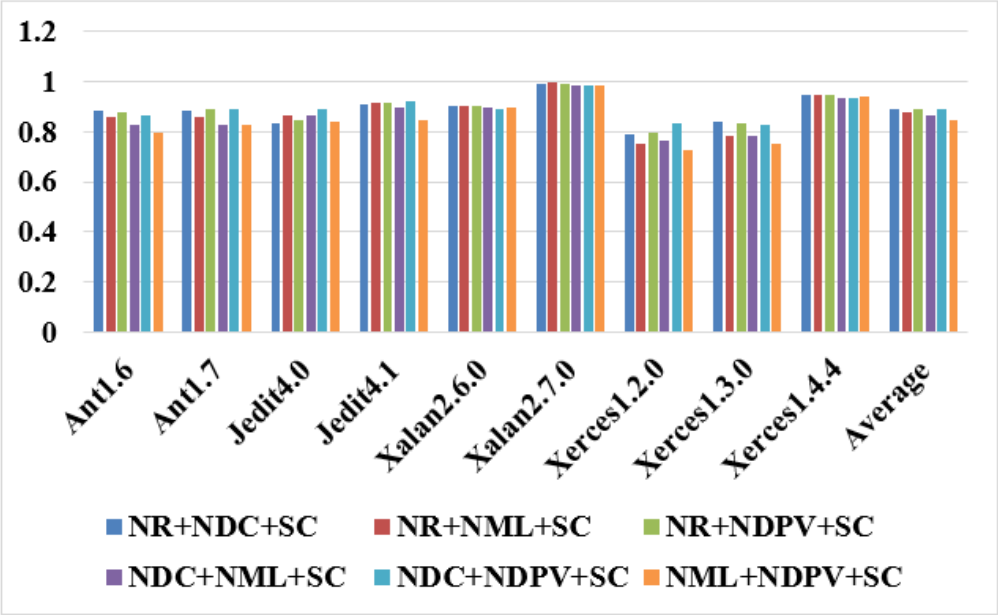

Fig 6.14 AUC values for stacking in combined model of static code and 2 process metrics

Table 5.15 has been analyzed statistically, the outcome shows that the calculated p-value is 9.38e-05 and χ2-statistics is 21.239 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.505, at least one of the model's performances differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NDC+NR+NDPV & model containing SC+NML+NDC+NDPV will have a significant difference between them.

For table 5.15 the bar chart has been represented by Figure 6.15. It can be observed that combined model of static code, NR, NDPV, and NDC outperforms other combinations of static code and process metrics.

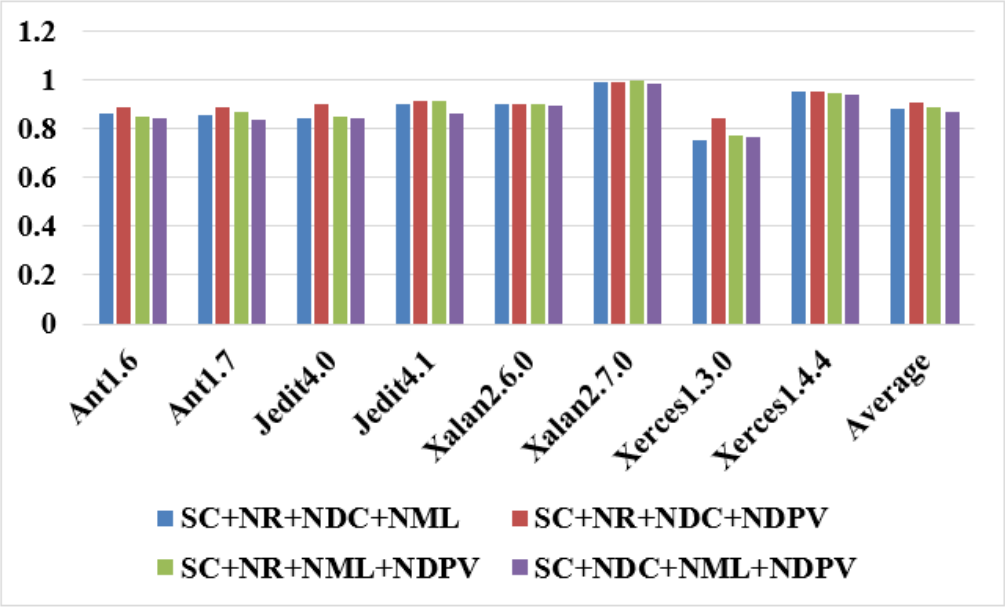

Fig 6.15 AUC values for stacking in combined model of static code and 3 process metrics

Table 5.16 has been analyzed statistically, the outcome shows that the calculated p-value is 3.99e-06 and χ2-statistics is 27.804 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.662, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR & model containing SC+NML, model containing SC+NR & model containing SC+NDPV, model containing SC+NDC & model containing SC+NDPV, and model containing SC+NDC & model containing SC+NML will have a significant difference between them. For table 5.16 the bar chart has been represented by Figure 6.16. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

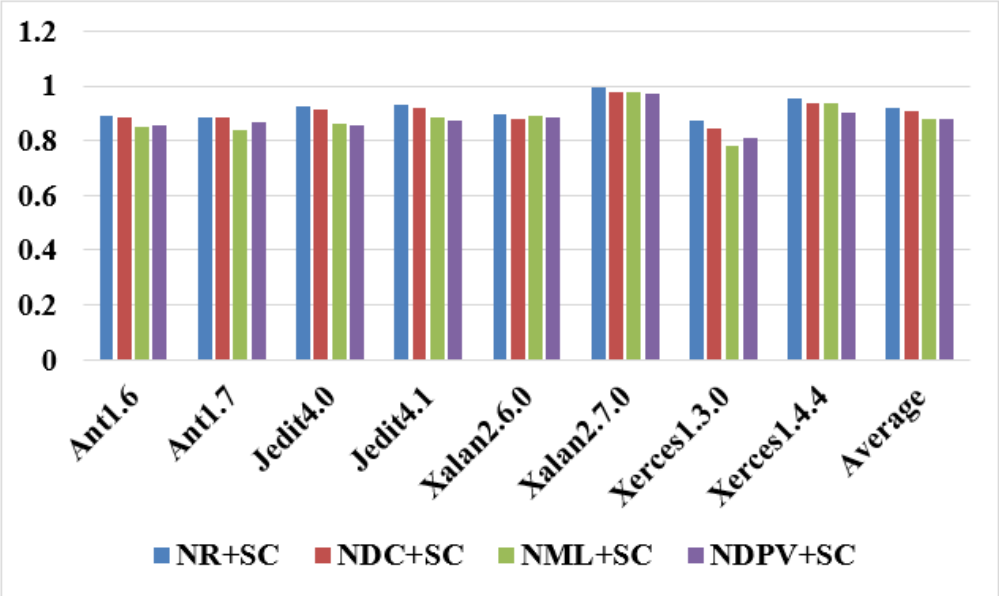

Fig 6.16 AUC values for voting in combined model of static code and 1 process metric

Table 5.17 has been analyzed statistically, the outcome shows that the calculated p-value is 3.77e-09 and χ2-statistics is 47.868 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.684, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+NDC & model containing NR+SC+NML, model containing NR+SC+NDC & model containing NDC+SC+NML, model containing NR+SC+NDC & model containing NML+SC+NDPV, model containing NR+SC+NML & model containing SC+NDC+NML, model containing SC+NR+NML and model containing NML+SC+NDPV, model containing NDC+SC+NML & model containing NDC+SC+NDPV, and model containing NDC+SC+NDPV & model containing NML+SC+NDPV will have a significant difference between them. For table 5.17 the bar chart has been represented by Figure 6.17. It can be observed that combined model of static code, NDC, and NR outperforms other combinations of static code and process metrics.

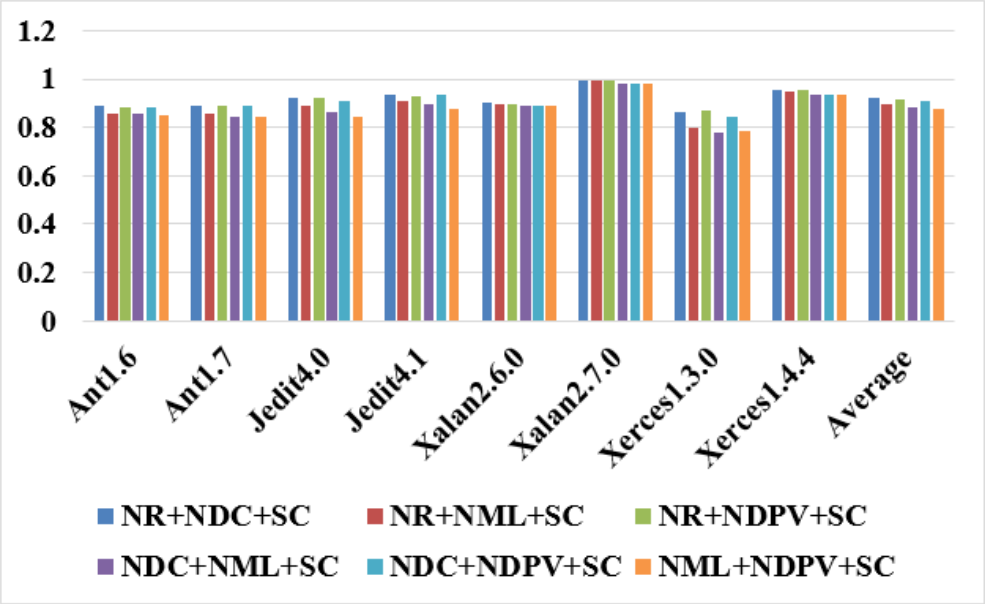

Fig 6.17 AUC values for voting in combined model of static code and 2 process metrics

Table 5.18 has been analyzed statistically, the outcome shows that the calculated p-value is 1.50e-06 and χ2-statistics is 29.826 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.710, at least one of the model's performances differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NDC+NR+NML & model containing SC+NML+NDC+NDPV, model containing SC+NDC+NR+NDPV & model containing SC+NML+NR+NDPV, and model containing SC+NDC+NR+NDPV & model containing SC+NML+NDC+NDPV will have a significant difference between them.

For table 5.18 the bar chart has been represented by Figure 6.18. It can be observed that combined model of static code, NDC, NDPV, and NR outperforms other combinations of static code and process metrics.

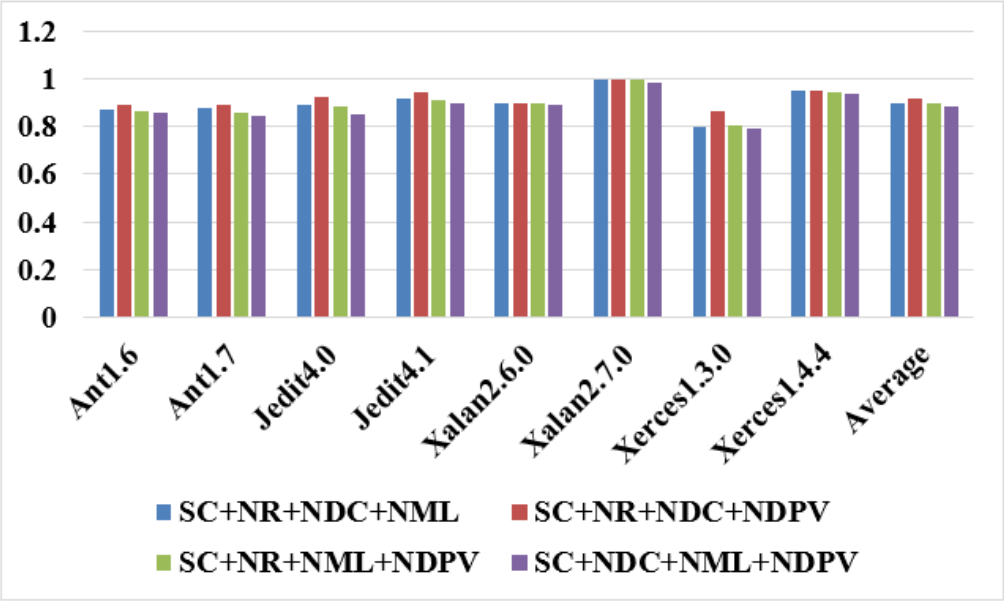

Fig 6.18 AUC values for voting in combined model of static code and 3 process metrics

Table 5.19 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0001 and χ2-statistics is 20.61 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.490, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR & model containing SC+NML, model containing SC+NR & model containing SC+NDPV, and model containing SC+NDC & model containing SC+NDPV will have a significant difference between them. For table 5.19 the bar chart has been represented by Figure 6.19. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

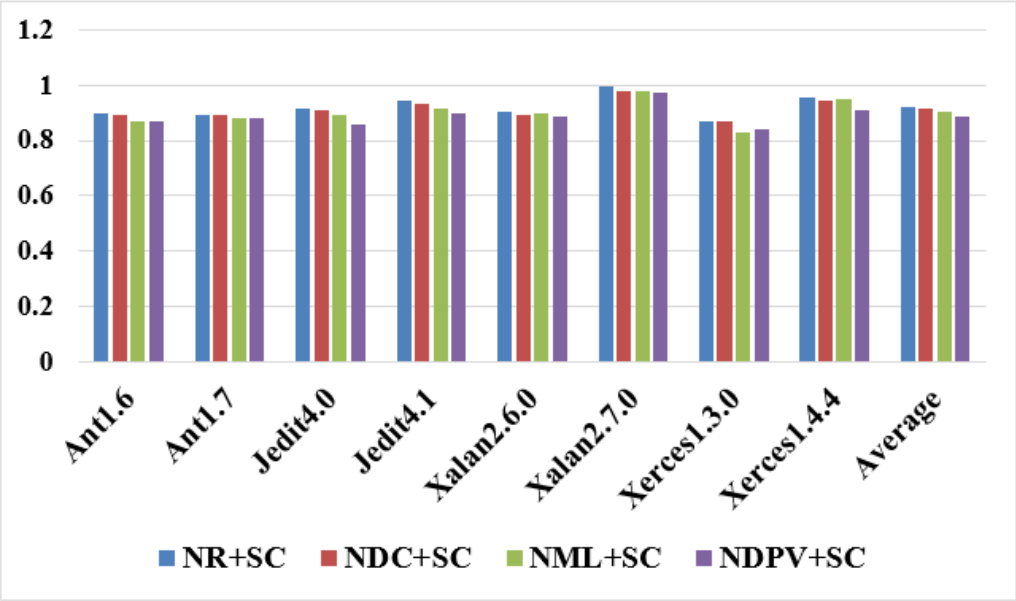

Fig 6.19 AUC values for bagging in combined model of static code and 1 process metric

Table 5.20 has been analyzed statistically, the outcome shows that the calculated p-value is 1.069e-06 and χ2-statistics is 35.74 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.511, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+NDC & model containing NDC+SC+NML, model containing NR+SC+NDC & model containing NDPV+SC+NML, model containing NR+SC+NDPV & model containing NML+SC+NDC, and model containing NR+SC+NDPV & model containing NML+SC+NDPV will have a significant difference between them. For table 5.20 the bar chart has been represented by Figure 6.20. It can be observed that combined model of static code, NDC, and NR outperforms other combinations of static code and process metrics.

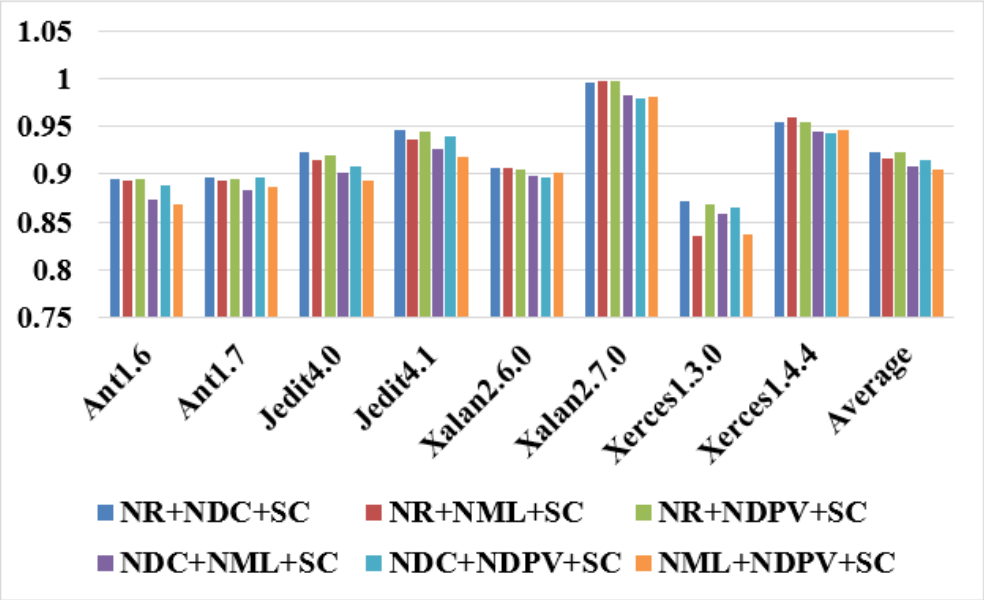

Fig 6.20 AUC values for bagging in combined model of static code and 2 process metrics

Table 5.21 has been analyzed statistically, the outcome shows that the calculated p-value is 1.50e-06 and χ2-statistics is 29.826 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.710, at least one of the model's performances differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NDC+NR+NML & model containing SC+NDC+NR+NDPV, model containing SC+NDC+NR+NML & model containing SC+NML+NDC+NDPV, model containing SC+NDC+NR+NDPV & model containing SC+NML+NR+NDPV, and model containing SC+NDC+NR+NDPV & model containing SC+NML+NDC+NDPV will have a significant difference between them. For table 5.21 the bar chart has been represented by Figure 6.21. It can be observed that combined model of static code, NDC, NDPV, and NR outperforms other combinations of static code and process metrics.

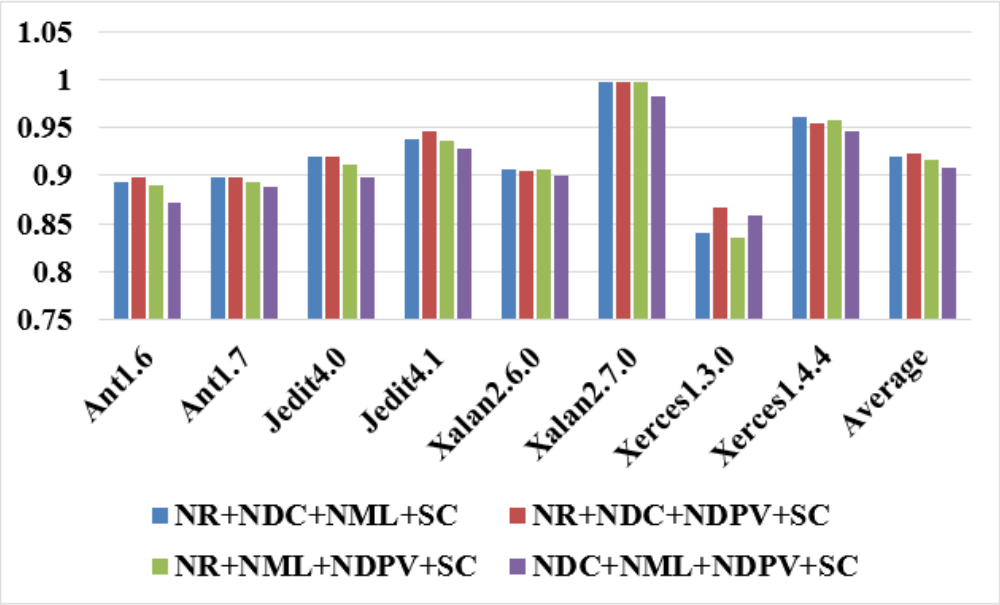

Fig 6.21 AUC values for bagging in combined model of static code and 3 process metrics

Table 5.22 has been analyzed statistically, the outcome shows that the calculated p-value is 0.0003 and χ2-statistics is 18.625 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.444, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NR & model containing SC+NML, model containing SC+NR & model containing SC+NDPV, and model containing SC+NDC & model containing SC+NML will have a significant difference between them. For table 5.22 the bar chart has been represented by Figure 6.22. It can be observed that combined model of static code and NR outperforms other combinations of static code and process metrics.

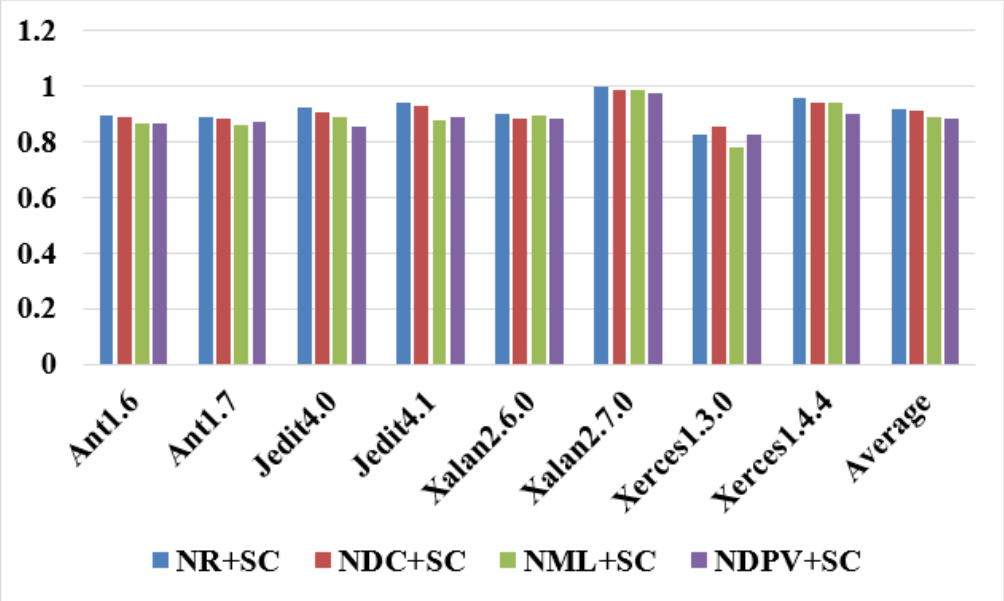

Fig 6.22 AUC values for boosting in combined model of static code and 1 process metric

Table 5.23 has been analyzed statistically, the outcome shows that the calculated p-value is 1.423e-08 and χ2-statistics is 45.04 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 5 at 0.05 significance level which is 11.070 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.643, at least one of the model's performance differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing NR+SC+NDC & model containing NR+SC+NML, model containing NR+SC+NDC & model containing NDC+SC+NML, model containing NR+SC+NDC & model containing NML+SC+NDPV, model containing NDC+SC+NDPV & model containing NML+SC+NDPV, model containing NDC+SC+NML & model containing NDC+SC+NDPV, model containing NR+SC+NDPV & model containing NML+SC+NDPV, and model containing NR+SC+NML & model containing NR+SC+NDPV will have a significant difference between them. For table 5.23 the bar chart has been represented by Figure 6.23. It can be observed that combined model of static code, NDC, and NR outperforms other combinations of static code and process metrics.

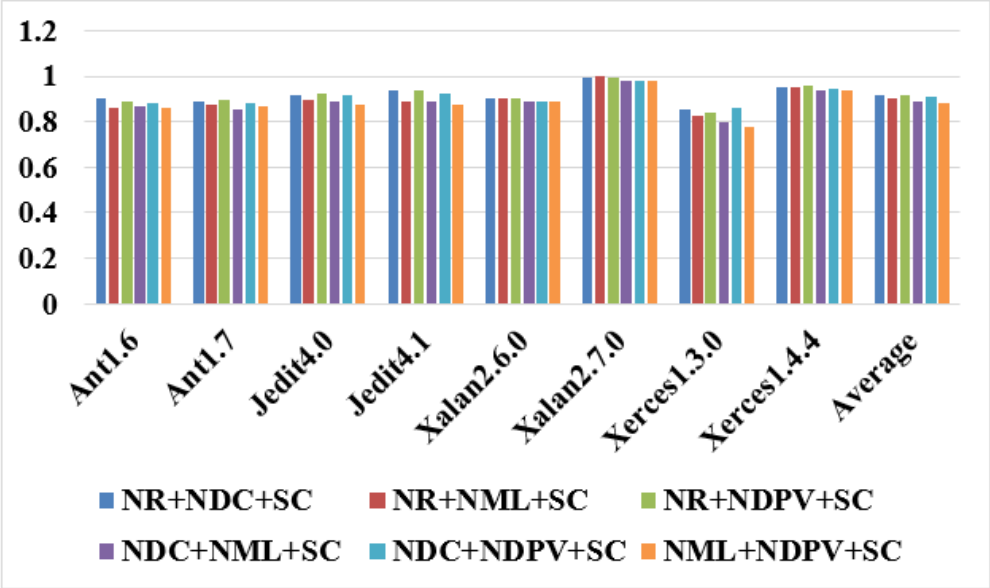

Fig 6.23 AUC values for boosting in combined model of static code and 2 process metrics

Table 5.24 has been analyzed statistically, the outcome shows that the calculated p-value is 1.65e-05 and χ2-statistics is 24.860 at 0.05 significance taking Friedman test into consideration, and when compared to χ2 value with degree of freedom as 3 at 0.05 significance level which is 7.815 i.e., lower than the χ2-statistics calculated, hence the null hypothesis is rejected. So, with an effect size of 0.592, at least one of the model's performances differ considerably thus indicating the difference amongst the models will slightly effect the performance of prediction, further Nemenyi post hoc test is applied to check the pairwise model's comparison, as the results differs significantly, and it is determined that model containing SC+NDC+NR+NML & model containing SC+NDC+NR+NDPV, model containing SC+NDC+NR+NDPV & model containing SC+NML+NDC+NDPV, and model containing SC+NDC+NR+NDPV & model containing SC+NML+NR+NDPV will have a significant difference between them.

For table 5.24 the bar chart has been represented by Figure 6.24. It can be observed that combined model of static code, NDC, NDPV, and NR outperforms other combinations of static code and process metrics.

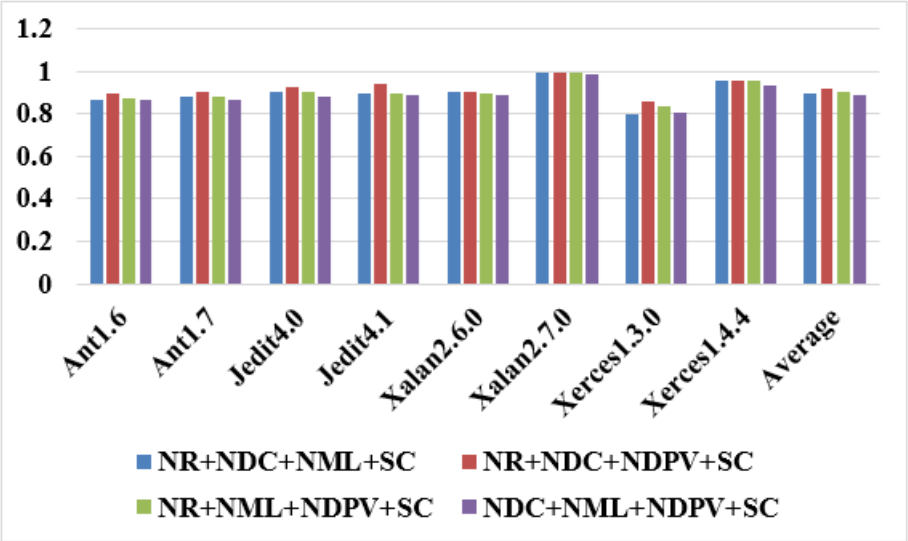

Fig 6.24 AUC values for boosting in combined model of static code and 3 process metrics

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In the area of software defect prediction numerous models have been existing and are explored. A wide range of those use static code metrics to detect defective software codes or programs. In this project the efficiency of process metrics or variable have been analyzed using classification and ensemble methods on the basis of AUC values, bar graphs, and statistical test.

It can be concluded that bagging outperforms all the selected classification and ensemble methods. All the selected techniques (NB, SVM, LR, KNN, DT, stacking, bagging, voting, and boosting) except logistic regression and KNN performs better in combined model. We can also conclude that ensemble methods produce better results as compared to the classical classification algorithms.

In case of combined model of SC and 1 process metrics, NR is most effective process metrics. In case of combined model of SC and 2 process metrics, NR+NDC is most effective. In case of combined model of SC and 3 process metrics, NR+NDC+NDPV is most effective.

7.2 Future Scope

Other process metrics can also be analyzed using classification and ensemble techniques. The number of defects can be predicted using regression methods while considering the same datasets which have been used in this research. Effort or maintainability can also be considered as dependent variables instead of bugs.

REFERENCE

[1] A. Komalasari and M. Z. C. Candra, "Improving Defect Prediction Using Combination of Software Metrics," In *2022 International Conference on Data and Software Engineering (ICoDSE)*, pp. 89–94, 2022.

[2] A. Okutan and O. T. Yildiz, "Software defect prediction using Bayesian networks," *Empirical Software Engineering*, vol. 19, no. 1, pp:154–181, 2014.

[3] A. Panichella, R. Oliveto, and A. de Lucia, "Cross-project defect prediction models: L'union fait la force," *In 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pp. 164–173, 2014.

[4] B. Diri, C. Catal, and U. Sevim, "Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm," *Expert Syst. Appl*, vol. 38, no. 3, pp. 2347-2353, 2011.

[5] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," *In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, pp. 789–800, 2015.

[6] D. Hosmer and S. Lemeshow, "Applied Logistic Regression," John Wiley & Sons, 1989.

[7] D. Radjenovic, M. Hericko, R. Torkar, and A. Zivkovic, "Software fault prediction metrics: A systematic literature review," *Information and Software Technology*, vol. 55, no. 8, pp:1397 – 1418, 2013.

[8] G. R. Choudhary, S. Kumar, K. Kumar, A. Mishra, and C. Catal, "Empirical analysis of change metrics for software fault prediction," *Computers & Electrical Engineering*, vol. 67, pp. 15–24, 2018.

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

[10] I. Rish, "An empirical study of the naive bayes classifier," *In IJCAI 2001 workshop on empirical methods in artificial engineering*, vol: 3, pp 41-46, 2001.

[11] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[12] J. Hanley and B.J. McNeil, "The meaning and use of the area under a Receiver Operating Characteristic ROC curve," *Radiology*, vol. 143, no. 1, pp. 29-36, 1982.

[13] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *ACM Computing Surveys. (CSUR)*, vol. 45, no. 1, p. 10, 2012.

[14] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using Bayesian network classifiers," *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp:237–257, 2013.

[15] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? An empirical study," *Software Quality Journal*, vol. 23, no. 3, pp:393–422, 2015.

[16] L. Perreault, S. Berardinelli, C. Izurieta, and J. Sheppard, "Using Classifiers for Software Defect Detection," *26th International Conference on Software Engineering and Data Engineering*, Sydney, pp:2-4, 2017.

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

[18] P. Sherrod, "DTreg predictive modeling software," 2003.

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

[20] R. Malhotra and A. Bansal, "Use of Support Vector Machine to Check Whether Process Metrics are as Good as Static Code Metrics," In *Topical Drifts in Intelligent Computing: Proceedings of International Conference on Computational Techniques and Applications (ICCTA 2021)*, vol. 426, p. 35, 2022.

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

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

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

[24] T. G. Dietterich, "Ensemble methods in machine learning," *In multiple classifier systems, Springer*, pp:1-15, 2000a.

[25] T. Hall, S.Beecham, D. Bowes, D.Gray, and S.Counsell, "A systematic literature review on fault prediction performance in software engineering", *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp:1276– 1304, 2012.

[26] V. Basili, L. Briand, and W. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no.10, pp:751–761, 1996.

[27] W. Rhmann, B. Pandey, G.Ansari, and D. K. Pandey, "Software Fault Prediction Based on Change Metrics Using Hybrid Algorithms: An Empirical Study," *Journal of King Saud University*, vol. 32, no. 4, pp:419–424, 2020.

[28] X. Wang, D. Bi, and S. Wang, S., "Fault recognition with labeled multi-category," *3rd Conference on Natural Computation*, Haikou, China, 2007.

[29] 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.

[30] Y. Zhao and Y. Zhang, "Comparison of Decision Tree Methods for Finding Active Objects," *Advances of Space Research*, vol. 41, no. 12, pp:1955, 1959, 2008.

LIST OF ACCEPTED PAPERS

[1] Malhotra, R. Ramchandani, R. (2022) A Comparative Study of Homogeneous and Heterogeneous Ensemble Techniques for Prediction of Software Faults. In: Proceedings of Communications in Computer and Information Science (CCIS), Springer, MIND 2022. (Scopus Indexed) (Accepted and Presented).

[2] Malhotra, R. Ramchandani, R. (2023) How Process Metrics Effect the Prediction Performance of Classification and Ensemble Methods. In: 5th International Conference on Advances in Computing, Communication Control and Networking- ICAC3N23 (IEEE Xplore, Scopus Indexed) (Accepted).