“Comparative Analysis of Filter Feature Selection Algorithms for Bug Prediction using multiple classifiers”

A PROJECT REPORT

SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted By

Harshit Arora

(2K19/SWE/06)

Under the supervision of

Dr. Ruchika Malhotra
Associate Professor
Department of Computer Science & Engineering
Delhi Technological University, Delhi

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

MAY, 2021

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Harshit Arora, 2K19/SWE/06 student of M.Tech (SWE), hereby declare that the project entitled **"Comparative Analysis of Filter Feature Selection Algorithms for Bug Prediction using multiple classifiers"** which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Software Engineering, has not been previously formed the basis for any fulfilment of the 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 Dr. Ruchika Malhotra.

Place: Delhi **Harshit Arora**

Date: 10th May, 2021 **(2K19/SWE/06)**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the project entitled **"Comparative Analysis of Filter Feature Selection Algorithms for Bug Prediction using multiple classifiers"** which is submitted by Harshit Arora (2K19/SWE/06) to the Department of Computer Science & Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Software Engineering, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere.

Place: Delhi **Dr. Ruchika Malhotra**

Date: **SUPERVISOR**

Head of Department

Dept. of Software Engineering

ACKNOWLEDGEMENT

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

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

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

Harshit

Date: **Harshit Arora**

(2K19/SWE/06)

Abstract

Software is a set of instructions with the sole purpose of defining functionality. It helps to make our life easier by doing heavy computation. That is precisely why developing software has become essential in the modern day. Many researchers are working on bug proneness of software using different approaches from manual testing to automation. In automation, Machine Learning algorithms are used to detect any flaw in the software. Their results vary from dataset to dataset. These algorithms give inconsistent output for predicting bugs in a random software project.

Software Bug Prediction is the process of classifying a new module as buggy or not using some historical data. Using Software bug prediction, the number of modules to be tested decreases drastically. As software size increases daily, developing a classification model becomes challenging due to the massive amount of data to be processed. To that end, feature selection can be used to reduce the dimensionality of data.

Feature selection is the process of reducing the feature space of a system under observation by using the evaluation criteria to select N relevant features from the original set. This reduced feature set helps in increasing the accuracy as well as the throughput of the models.

In this study, we analyzed the prediction performance of various classifiers based on multiple ranked search-based feature selection algorithms (filter algorithms). In other terms, we can say that all the different feature selection algorithms are used with each classifier to check the model's prediction power. We have used Naive Bayes classifiers, Logistic Regression, K-Nearest Neighbours, Bayesian Network, Random Forest, Decision Tree, MultiLayer Perceptron and four types of ensemble classifier (Voting, Stacking, Bagging and Boosting) for implementation, and data sets are collected from the PROMISE repository, which is publicly available. The area under the ROC curve (AUC) is used to analyze the prediction performance. Friedman test is used, To check the statistical significance of the results of the different models. This study shows that the Feature Selection Algorithms improve the performance of SBP Models, and Correlation Attribute Feature Evaluation and Symmetrical Uncertainty gives effective results. Also, Bagging Ensemble gives the best results in all classifiers studied.

CONTENTS

List of Figures

List of Tables

List of symbols and abbreviations

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

OR	One R
RF	Relief F
SU	Symmetrical Uncertainty
CR	Correlation
CS	Chi Square
LOG	Logistic Regression
BN	Bayesian Network
NB	Naive Bayes
KNN	K-Nearest Neighbour
RFo	Random Forest
DT	Decision Tree
MLP	MultiLayer Perceptron
VE	Voting Ensemble
SE	Stacking Ensemble
BE	Bagging Ensemble
BoE	Boosting Ensemble
RFE	Recursive Feature Elimination
CFS	Correlation Based Feature Selection
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

1.1 General

Software testing is a resource and time-consuming task in the software development lifecycle. The end goal of the testing process is to deliver error-free software that meets all stakeholders' requirements. Continuous changes, strict deadlines, and the need to ensure the correct behaviour of the functionality are some challenges faced by developers consistently. However, limited time and workforce are threats to practical testing. Therefore, instead of testing the whole software for bugs, allocating all the resources to the bug-prone classes will be easier if we know them.

Bug Prediction Model, which predicts the software component which needs to be tested more extensively and is more likely to have bugs, offers an effective solution to the above problem. **Software Bug Prediction (SBP)** is one of the most assisting activities in the **Testing Phase** of SDLC. SBF models use multiple software metrics data collected either from previous versions of the same system (within-project approach) or from the metric data of other systems (cross-project approach).

Ghotra *et al.* established that the accuracy of a prediction model can increase or decrease up to 30% depending upon the classifier used [15]. Also, Panichella *et al.* prove that the predictions of different classifiers are highly interdependent despite similar prediction accuracy [28].

The model can be trained using sufficiently large data from the project under observation (within-project strategy) or using data from a similar project, not under observation (cross-project strategy). But the main problem with the dataset is the higher dimensionality of the metrics, including redundant or irrelevant metrics. Higher dimensionality of the dataset will lead to higher costs for building and testing the systems. For the foregoing reasons, a variety of feature selection methods were proposed to alleviate this issue of high dimensionality by eliminating irrelevant and redundant features [39].

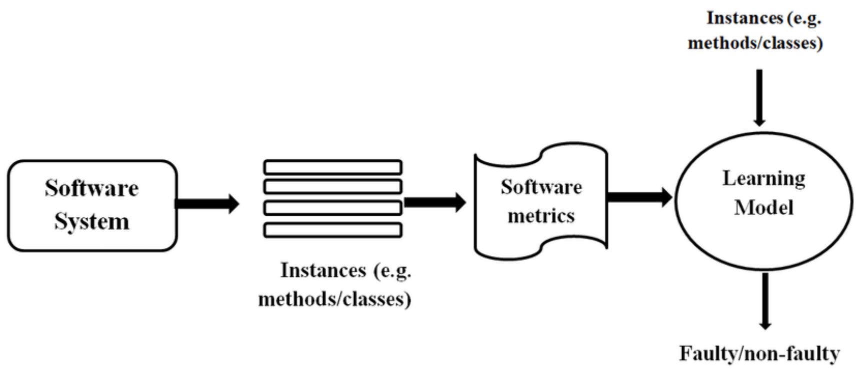

Instances (e.g. methods/classes)
Software System
Instances (e.g. methods/classes)
Software metrics
Learning Model
Faulty/non-faulty

Fig.1.1 Software Bug Prediction Steps

Software Bug Prediction (SBP) Model predicts the software component that needs to be analyzed more for the bugs as the bug probability is higher in that component. The bug prediction model is a supervised method in which a set of independent variables are selected (predictor) and are used to predict the value of the dependent variable (bug proneness of the component) using one or more ML Classifiers.

Feature selection is the process of selecting N relevant features from the original set to optimally reduce the feature space according to the evaluation criteria and system under observation. Datasets with thousands of features are not uncommon in such applications. All features may be necessary for some problems, but for some target concepts, only a small subset of features is usually relevant [27].

The process of Feature Selection is divided into four parts:

1. Subset Generation
2. Evaluation
3. Stopping Criteria
4. Validation

1.2 Problem Formulation

During the testing phase, it is not feasible to test the complete product, nor is it possible to perform 100% testing. So to find the bug proneness of a particular function (area), we use a bug prediction model, which can help identify weak areas. Based on this problem following question have been identified:

1. What models are used to predict the proneness of a function?
2. What datasets are available to test these models?
3. Do the current models incorporate ensemble techniques?

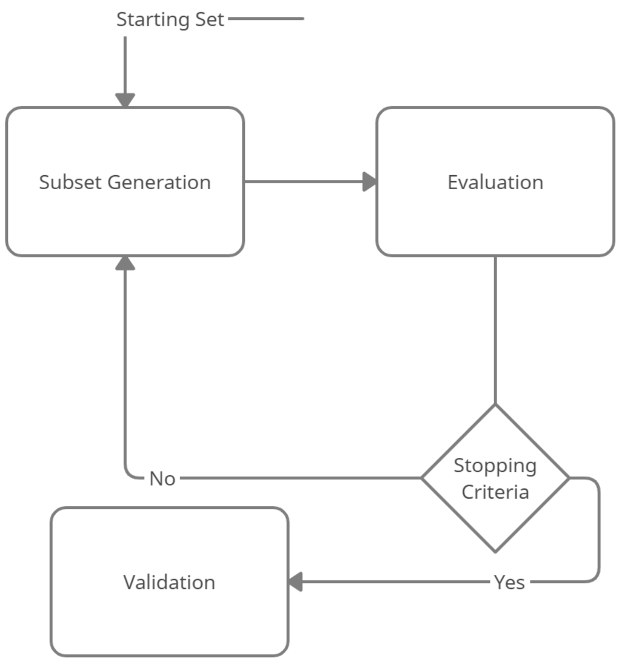


Fig. 1.2 Feature Selection Process

1.3 <u>Objectives of the Project</u>

This study evaluates the performance of Filter Feature Selection Algorithms and compares their predictive power with different classifiers for SBP. The main focus of this study is on the following research questions:

1. Which Feature Selection Algorithm gives the best results?
2. What is the effect of Feature Selection Algorithms on Fault Proneness Models?
3. Which classifier or ensemble techniques perform the best?

Therefore, in this study, we will build bug prediction models using multiple classification techniques and then compare and analyze each classification technique's results based on AUC values [20]. We will analyze the results of each classification technique based on different Feature Selection Algorithms.

CHAPTER 2

LITERATURE REVIEW

In this section, information related to the variety of research papers concentrating on software bug prediction using different methodologies, challenges present in bug prediction, and which techniques can perform better, etc. has been listed. This section provides a brief insight into the previous works done in the field of software bug prediction.

Many researchers 6have been trying to build software bug prediction models using different bug prediction techniques which deliver better performance, but most of them use static code metrics as independent variables, and few of them use feature selection to analyze metrics.

Most of the previous work of bug prediction models is done using predictors such as CK metrics, HIstory Based Metrics.

1. Individual Classifiers for Bug prediction

Several Classifier can be used to build a bug prediction model like Logistic Regression (LOG), Support Vector Machines (SVM), Radial Basis Function Network (RBF), Multi-Layer Perceptron (MLP), Bayesian Network (BN), Decision Tree (DTree), and Decision Tables (DTables). But there is no clear winner from previous studies as to which is the best classifier to predict bug proneness. Their performance depends upon the predictor we selected and the dataset used.

Lessman et al. [21] experimented with 22 classification models. The top 17 models were statistically similar to each other on ten publicly available software development data sets from the NASA repository. Later Shepperd et al. [34] found that the NASA dataset used was noisy and biased.

2. Within- vs Cross-Project Bug Prediction

The dataset can be collected in two ways:

a. Within-Project: Dataset only contains Historical data of the system under observation. It can only be applied to mature projects, where sufficient amount of project history is available. This strategy is preferred due to the homogeneity of data.

b. Cross-Project: Dataset contains data from other systems that are similar to the system under observation. It is applied to those projects where a sufficient amount of project history is not available. So a dataset can only be constructed with data from other systems.

3. Individual Feature Selection for Bug Prediction

Feature selection is the process of selecting **N** features from the original set of features to decrease the dimensionality of the feature pool and increase the performance while lowering the cost and time to build the prediction model.

Types of Feature Selection methods are listed below:

a. **Wrapper Method**: These methods carry out the selection process keeping in mind the classification algorithms that are going to be used. Wrapper methods use the predictor as a black box and the predictor performance as the objective function to evaluate the variable subset [9].

b. **Filter Method**: These methods carry out the selection process without considering the algorithm used for classification. Filter methods are faster than wrapper methods and result in a better generalization because they act independently of the classification algorithm.

c. **Embedded method**: These methods encompass the benefits of both the wrapper and filter methods by including interactions of features but also maintaining reasonable computational cost. Embedded methods are iterative because they take care of each iteration of the model training process and carefully extracts those features that contribute the most to the training for a particular iteration [3].

4. Homogenous and Heterogeneous Ensemble

Ensemble models have been demonstrated exceptionally successfully to inspire the precision and the presentation of the models. An ensemble consists of a set of individually trained classifiers, such as neural networks or decision trees, whose predictions are combined when classifying new instances [29]. Ensemble takes place in two steps:

a. Model Training: Training each individual classifier with the same training dataset but using different subsets.

b. Model Combination: Combining the power of all the trained classifiers using one of the combining techniques (Averaging or Voting).

There are two types of ensemble techniques:

a. Homogeneous Ensemble: It consists of classifiers having a single-type base learner. Example bagging or boosting.
b. Heterogeneous Ensemble: It consists of classifiers having different base learning algorithms. Example stacking or voting of bagged classifiers.

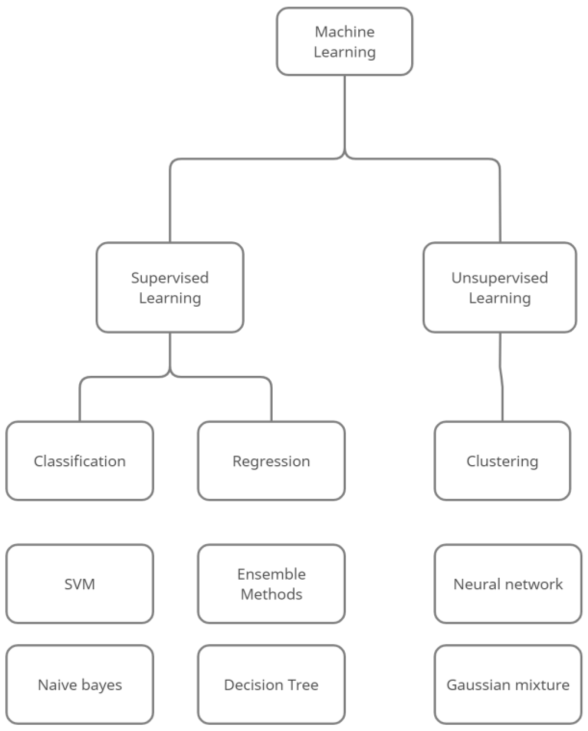


Fig. 2.1 ML Techniques

An Empirical Comparison is made by Zhou et al. [39] consisting of 32 feature selection methods and the result of this study shows that feature selection algorithms significantly improve the bug prediction performance. Also, Wrapper and filter feature selection algorithms give the best result compared to clustering-based, but they tend to take more time to select features.

Shivaji et al. [35] conducted research showing that the performance of bug prediction models is increased by eliminating 90% of the original features. Also, NOVAKOVIC et al. [27] compares the performance of 5 filter feature selection algorithms concluding that to rank the algorithms based on their performance, one will need to keep indices to check the best feature subset is selected and a bigger dataset is needed with more classifiers.

CHAPTER 3

THEORETICAL CONCEPTS

This section presents the basic theoretical concepts required to understand the key processes and working of the experiment studied in this project. This section familiarizes the concept of data preprocessing, features, feature selection algorithms and ensemble techniques. It also induces the idea of working with individual classifiers. The concepts introduced in this section help to understand the proposed architecture for software bug prediction models used.

3.1 **Features**

A feature is an individual measurable property or characteristic of a phenomenon being observed [38]. Data objects are described by many features, which captures the essence of the object under consideration.

Features are also known as Variables, characteristics, attributes, etc.

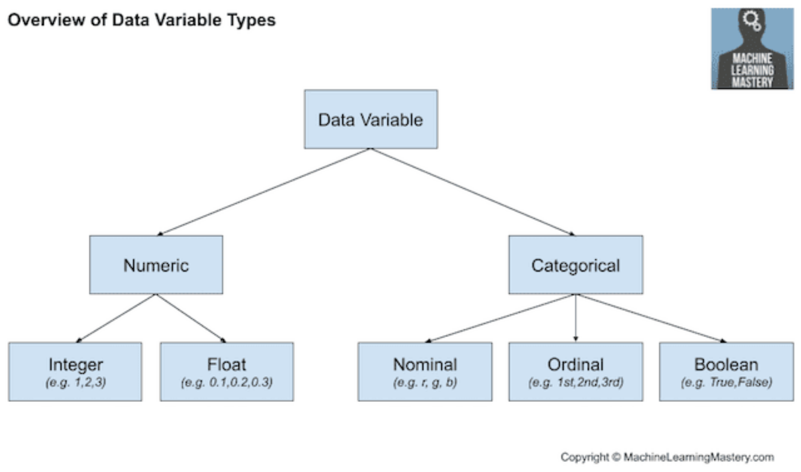


Fig. 3.1 Data Variable Types [5]

There are 2 types of features:

1. Categorical - values taken from a defined set. Example: Days of the week.
2. Numerical - values are continuous or integer-valued. Example: Speed of the car.

3.2 **Data Preprocessing**

The process of converting raw data into an efficient and usable format using data mining techniques. This process reduces the prediction time for our models by selecting the relevant features for bug prediction and disposing of the rest.

The process of Data Preprocessing includes:

1. Data Cleaning: The process of filling the missing information or removing the noisy data from the dataset is called data cleaning. There are multiple ways to handle cleaning, like Fill the missing values, clustering, etc.
2. Data Transformation: This process turns the data into a suitable form for the mining process. Steps include are:
    a. Normalization: Getting all the values in a specific range
    b. Attribute Selection: Construction of new attribute from older one for mining purpose
    c. Discretization: Replacing the raw value of a numeric attribute by interval/conceptual levels.
    d. Concept Hierarchy Generation: Converting low-level hierarchy attributes to a higher level.
3. Data Reduction: It helps to reduce the size of data that we process for our models to amplify the prediction speed and accuracy of the models. Steps include are:
    a. Data Cube Aggregation: Application of Aggregation operation to generate data cube.
    b. Attribute Subset Selection: Forming a subset of those features with higher relevance to our goal feature.
    c. Numerosity Reduction: It is a process of storing a model of data instead of complete data.
    d. Dimensionality Reduction: Process of reducing the size of data by encoding mechanism. The methods can be lossy or lossless.

3.3 Feature Selection

Feature Selection is the process of selecting a subset of input attributes from a given set of inputs in order to minimize the computational cost and improve the prediction abilities of the predictive model to be developed. The aim of feature selection is to remove the redundant and irrelevant inputs for the predictive model. Feature Selection is closely related to the dimensionality reduction process. Feature Selection focuses on adding or deleting data from a dataset whereas dimensionality reduction focuses on projecting the data to generate entirely new sets of inputs.

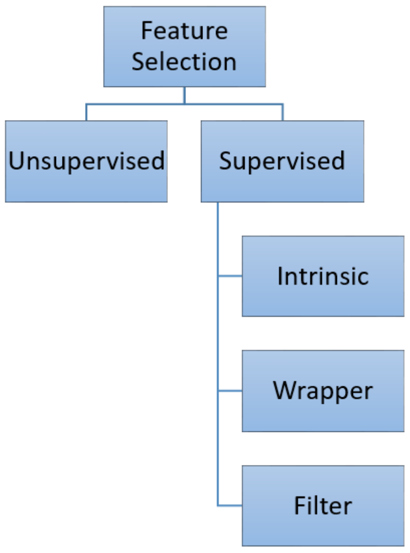


Fig. 3.2 Feature Selection Types

There are two ways of performing feature selection:

a. Unsupervised: Remove redundant predictors without utilizing the target variable. Example - Correlation Based Feature Selection (CFS)
b. Supervised: Remove irrelevant predictors using the target variable as guiding parameter for selection. Example - Recursive Feature Elimination (RFE)

Supervised Feature Selection is further divided into three types:

a. Wrapper: Perform search to select a subset of features that performs well. Example - RFE
b. Filter: Select subset of features based on the relation with the target variable. Examples - Feature Importance Methods
c. Intrinsic: Algorithm that automatically selects the feature during model training. Example - Decision Tree

3.4 Classification

Classification is the process of predicting or assigning class/label to an unknown input using the set of inputs provided to the predictive model during training. There are two types of classification:

a. Binary Classification: In this type of classification, only two classes are available for assignment, i.e., input belongs to either class available. Example - Spam Classification.
b. Multi-class Classification: In this type of classification, more than two classes are available for assignment, i.e., the input can belong to one of the available classes. Example - Plant Species Classification

3.5 Ensemble Technique

Ensemble strategies appear to be meta-calculations that are a combination of a few methods of machine learning into one prescient model to improve predictions (casting a ballot), decline predisposition (boosting), or decline difference (sacking). With the help of an ensemble of classifiers, one can achieve the better predictive power of the model developed. The member classifiers of an ensemble may or may not be of the same type and may or may not be trained on the same training data (Homogenous or Heterogeneous Ensemble). The main goal of Ensemble Techniques:

1. Performance: Increases the prediction capabilities of the developed model.
2. Robustness: Reduces the overall dispersion of prediction and model performance.
3. Low Bias, Low Variance: Provides a way to decrease the variance while increasing the model's performance.

There are many types of Ensemble like BAGGING, BOOSTING, VOTING, STACKING, RANDOM FOREST, etc.

3.6 Software Bug Prediction

Software deformity (or deficiency) prediction is viewed as one of the most practical and furthermore useful devices which let us know whether a specific module is having imperfection or not. Software professionals consider it to be an essential stage for guaranteeing the nature of the procedure or the item which is to be created. It made light of an exceptionally pivotal job in achieving the cases in the software industry that it can't meet the necessities in the spending plan and on schedule.

Today, software can be huge and to test the complete software is not feasible in terms of time as well as according to cost perspective. Error could be present anywhere in the code so to distinguish which modules are defective and which are not plays an important role in reducing the overall cost of the software. Software bug prediction models helps to find these bug prone modules as early as possible to increase efficiency, accuracy and durability of software and to reduce the cost of building the software.

CHAPTER 4

PROPOSED MODEL

This section presents the proposed model being utilized in this project. This section familiarizes the architecture of the model used. It also helps to understand the comparison of multiple feature selection algorithms and classification algorithms.

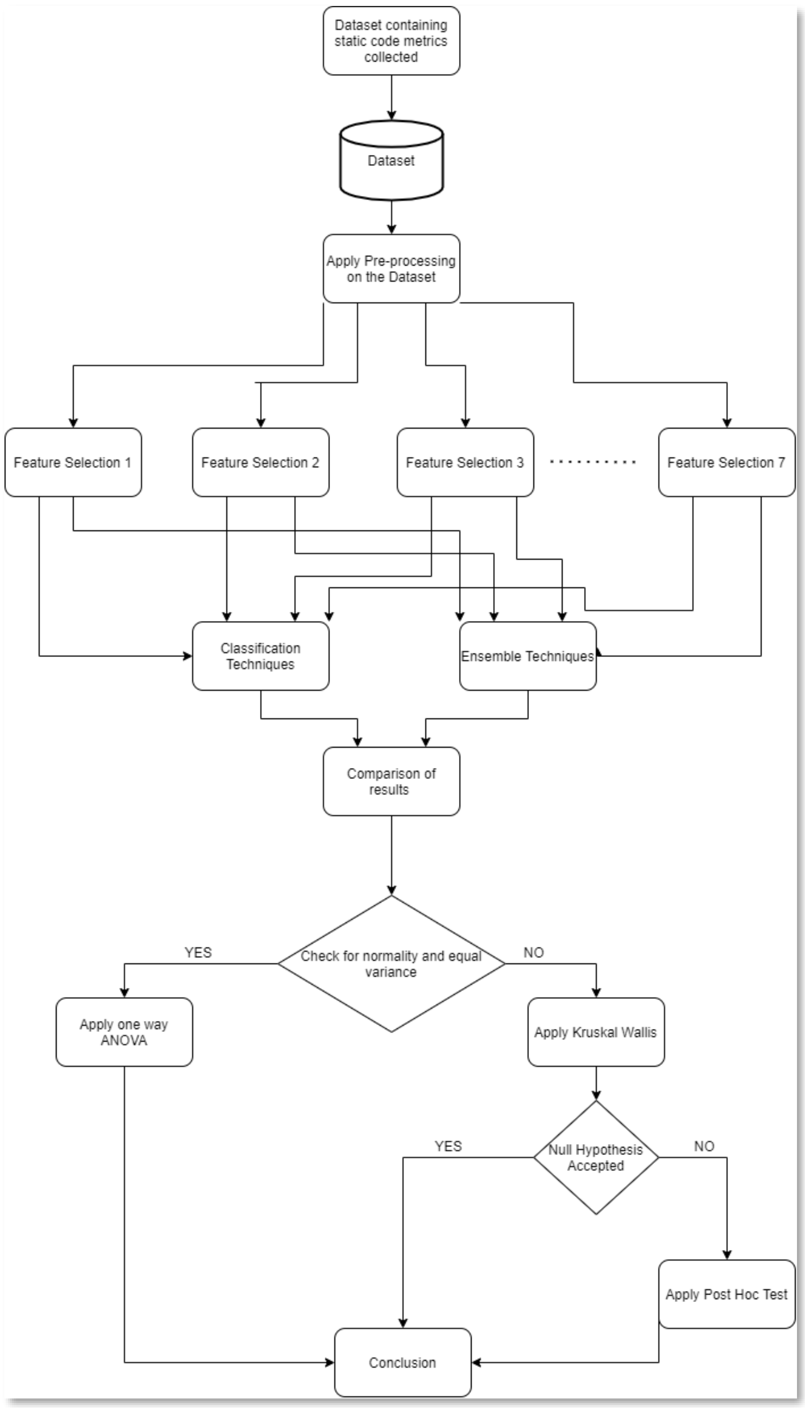


Fig. 4.1 Proposed Architecture

In this section, static code metrics are used as dependent and independent variables and are defined below. Also, datasets are collected using empirical data collection methods.

4.1 Dependent and Independent Variables:

Bug proneness is defined as the probability of finding bugs in the class. Independent variables used in this study are static code metrics, and the dependent variable used in this study is **bug proneness**. Table I shows the static code metrics used in this study. Jureczko and Madeyski[19] defined each of the static metrics given in table 4.1.

Table 4.1: Static Code Metrics Details

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

4.2 Empirical Data Collection:

The dataset is collected from 9 projects. All are java based projects. Ckjm [23] program and the BugInfo [13] tool was used by Madeyski and Jureczko [1] to collect metrics from project repositories. Bug is the dependent metric used in this study, the rest are independent. The datasets which have been used in this study are shown in table 4.2.

Table 4.2: Dataset Details

Project name	Version	Description
Ant	1.6, 1.7	Java-based build tool.
Camel	1.4, 1.6	Open-source integration framework based on known Enterprise Integration Patterns.
Ivy	1.4,2.0	Dependency Manager.
Lucene	2.4	Full text search library in java.
POI	2.5, 3.0	Java- based tool to create and maintain API.
Synapse	1.2	Enterprise service bus
Tomcat	6.0	Open source implementation of jsp
Velocity	1.6	Template Builder
Xalan	2.5,2.6	XSLT processor

CHAPTER 5

EXPERIMENTAL SETUP

5.1 **Dataset**

In the experiment, 15 projects from the PROMISE repository are taken and are then cleaned using the WEKA tool.

5.2 **Dataset Preprocessing**

Datasets are downloaded from PROMISE repositories, which may contain some missing data or noise, which affect the generated model's performance. Preprocessing is done to avoid these problems, such as removing unwanted metrics like Version field, ID field, Project name metric, etc. Using one of the WEKA tool filters, we can convert the bug metric from "numeric to nominal" as software's bug field contains binary values either 0 or 1.

5.3 **Feature Selection Algorithm**

Entropy is the foundation of GR, IG and SU which is considered as the measure of unpredictability of the system. The entropy of y is:

$$H(Y) = \sum_{y \epsilon Y} p(y) log_2(p(y))$$

Where p(y) is a marginal probability density function of random variable Y.

Suppose the observed values of Y in the training data set S is partitioned on the basis of second feature X, Y's entropy with respect to the partitions produced by X is less than the entropy of Y prior to partitioning. In that case, there is a relationship between features Y and X. The entropy of Y after the partition produced by X is then [27]:

$$H(Y|X) = -\sum_{x \epsilon X} p(x) \sum_{y \epsilon Y} p(y|x) log_2(p(y|x))$$

In this study we have used seven ranker methods for feature selection:

a. **Gain Ratio (GR) attribute Evaluation:** It is the non-symmetrical measure that compensates for the bias of the IG [16]. GR is given by:

$$GR = \frac{IG}{H(X)}$$

Due to this normalization of IG, the GR will lie between [0, 1].

Higher the value of GR, higher the relation between X and Y. GR favors variables with fewer values [27].

b. **Information Gain (IG) attribute Evaluation:** Entropy is a measure of impurity in a training set S. We can define a standard by considering additional information about Y provided by X that represents the amount by which Y’s entropy decreases [16]. This measure is known as IG. It is given by:

$$IG = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

IG is symmetrical in nature. The information gained about Y after considering X is equal to the information gained about X after considering Y [27]. IG’s problem is that it is biased towards features with more values, even if they do not contain anything relevant.

c. **Symmetrical Uncertainty (SU) attribute evaluation:** The Symmetrical Uncertainty measures counterbalance for the inherent bias of IG by dividing it by the sum of the entropies of X and Y. It is given by:

$$SU = 2\frac{IG}{H(Y) + H(X)}$$

SU takes values, which are standardized to the range [0, 1] because of the correction factor 2.

SU = 1 means that one feature completely predicts while SU = 0 indicates that X and Y are uncorrelated [27]. SU is biased towards features with fewer values similar to GR.

d. **Relief-F (RF) attribute Evaluation:** In this, measurement of feature’s worth is done by repeated sampling of an instance and the value of the given feature for the nearest instance of the same and different class is also considered. To distinguish among the classes, this technique assigns a weight to each feature which is based on the ability of the feature and then those features are selected whose weight is greater than a user defined threshold [27].

e. **One-R (OR) attribute Evaluation:** It is useful for determining a baseline performance for other techniques. OR builds one rule for each attribute in the training set and selects the rule with the smallest error. All numerically valued features are treated as continuous and then divide the range of values into several disjoint intervals. Missing values are treated as legitimate values by considering them as “missing”. Novaković *et al.* [27] further explained OR’s functioning.

f. **Correlation (CR) attribute Evaluation:** Correlation is used to find good features that are strongly correlated to the class concept but are not redundant to any other relevant feature. The problem of attribute selection requires a suitable measure of correlations between attributes and a sound procedure to select attributes based on this measure [36]. It's value ranges from [-1,1]. Closer to +1 depicts positive correlation between the two features and -1 depicts negative correlation between the features. There are two types of correlation approaches:

   a. Based on **Classical Linear Correlation.**
   b. Based on **Information Theory.**

   Formula for first approach is given below:

$$Correlation(r) = \frac{N(\sum XY) - \sum X \sum Y}{\sqrt{[N(\sum X^2 - (\sum X)^2)][N(\sum Y^2 - (\sum Y)^2)]}}$$

   Where X and Y are Relevant features.

g. **Chi Square (CS) Attribute Evaluation:** Chi Square is used when the feature is categorical. It determines if the degree of association between two categorical samples would reflect their real association in population [11]. It assumes that the observed value of a variable will match the expected value of the variable (null hypothesis). Its

   Formula for Chi Square is given below:

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

   where:

   c = Df

   O = Observed Value(s)

   E = Expected Value(s)

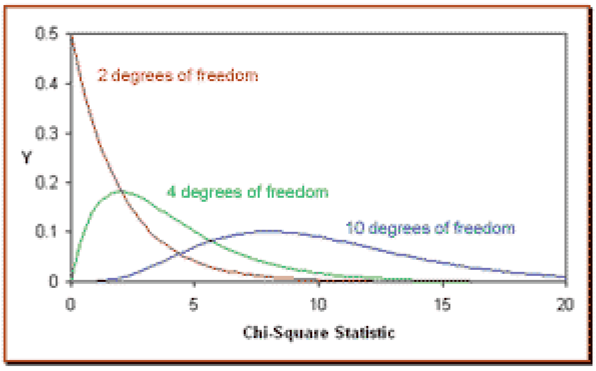


Fig. 5.1 Chi Square distribution for different Df[14]

5.4 **Classification and Ensemble Technique**

In this study, we are using seven different classifiers and four ensemble techniques to build the prediction model. For implementation, we use the WEKA machine learning tool. There can be some string data type variables due to which there can be some error, so we use a filtered classifier that provides the facility of all the classifiers with additional functionality. We choose meta filtered classifier first. After this, we choose any classification technique as classifier and filter as unsupervised.attribute.string to a word vector. All the parameter values are initialized with the default values in WEKA. Here we are also using a 10- fold cross-validation technique for splitting the dataset into testing and training.

Seven classification and four ensemble techniques which we have chosen for implementation are as follows:

a. **Naive Bayes:** It is a supervised classification technique. This classification technique gives better results and is simple to understand and implement [12]],[18]. It is based on one assumption that one feature's value is not dependent on another feature value. NB classifier technique follows **Bayes theorem**:

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

   P(F/c) is the posterior probability.

   P(F) is the class prior probability.

   P(c) is the predictor prior probability.

   P(c/F) is the likelihood.

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

b. **K- nearest neighbor:** It is another simple, non-parametric and supervised machine learning technique used in classification and regression problems. Also known as the lazy learning technique, KNN considers the k most similar instances to classify an instance by calculating the euclidean distance between instances [37]. It has also been used in pattern recognition, data mining, and intrusion detection.

c. **Bayesian Network:** Bayesian Network classifier technique is a supervised as well as an unsupervised classification technique and is used along with a search technique to build the prediction model as it helps in ordering the metrics on their importance in predicting the bugs. The K2 search algorithm is the most famous search algorithm used with BN to build the prediction model.

A Bayesian network is a DAG graph with E edges and V Vertices representing joint probability distribution of a set of variables. The probability is given by:

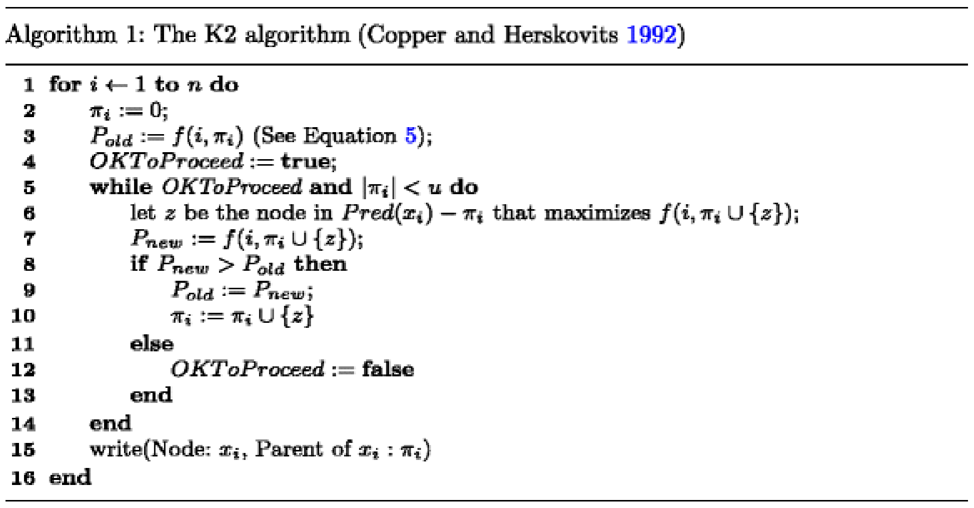
$$P(X) = \prod_{i=1}^{n} P(X_i | X_{i+1}, \ldots, X_n)$$

Given the parents of Xi, other variables are independent from Xi, so we can write the joint probability distribution as :

$$P(X) = \prod_{i=1}^{n} P(X_i | a_{X_i})$$

On the other hand, Bayes' rule is used to calculate X_i's posterior probability in a Bayesian network based on the evidence information present. We can calculate probabilities either towards or causes to effects (P(Xi|E)) or from effects to causes (P(E|Xi)) [1].

The K2 algorithm is given by Cooper and Herskovits which heuristically searches for the most probable Bayesian network structure.



Algorithm 1: The K2 algorithm (Copper and Herskovits 1992)

```
for $i \leftarrow 1$ to $n$ do
    $\pi_i := 0$;
    $P_{old} := f(i, \pi_i)$ (See Equation 5);
    $OKToProceed :=$ true;
    while $OKToProceed$ and $|\pi_i| < u$ do
        let $z$ be the node in $Pred(x_i) - \pi_i$ that maximizes $f(i, \pi_i \cup \{z\})$;
        $P_{new} := f(i, \pi_i \cup \{z\})$;
        if $P_{new} > P_{old}$ then
            $P_{old} := P_{new}$;
            $\pi_i := \pi_i \cup \{z\}$
        else
            $OKToProceed :=$ false
        end
    end
    write(Node: $x_i$, Parent of $x_i$ : $\pi_i$)
end
```

Fig 5.2. K2 Algorithm [1]

d. **Random Forest:** Random Forest is a supervised classification technique. In this, the algorithms generated a large number of decision trees as an ensemble [40]. The Decision tree with the highest vote becomes the model predictor. Two conditions for Random Forest are:

   i. A model with some actual signal should be built so that model does better than random guessing with those features

ii. The predictions made by the individual trees need to have low correlation with each other.

e. **Logistic Regression:** Logistic Regression is a supervised classification technique which is a widely used statistical technique used to predict dependent variables with the help of independent variables. Here binary Logistic regression is used to build the model as the dependent variable has binary values either 0 or 1. The classification takes place in 2 steps:

   i. First data is fit into the linear regression model as linear regression outputs continuous variables, so logistic regression makes use of the logistic sigmoid function to transform this output into probability value.

   ii. The probability is mapped to the target dependent variable categorically.

   A detailed description of logistic regression is given by Basili et al. (1996) [4] and Hosmer and Lemeshow (1989) [17].

f. **Decision Tree:** A Decision Tree is a non-parametric supervised classification technique. Its structure is like a flow chart where the internal nodes (decision nodes) describe the test on the features, and the leaf nodes describe a decision (label for classification). The main goal of DT is:

   1. Generalize beyond the training sample so that unseen samples could be classified with as high accuracy as possible [33].
   2. Classifying training sample as accurately as possible.
   3. Easily modifiable to accommodate more training samples when available.
   4. To have a simple structure.

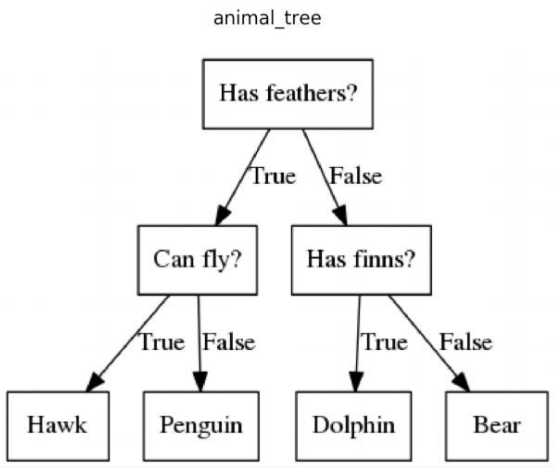


Fig. 5.3 Decision Tree example [10]

Keeping point as mentioned earlier, the design of the DT classifier can be broken into the following tasks [2] [25] [26]:

1. The appropriate choice of the tree structure.
2. The choice of feature subsets to be used at each internal node.
3. The choice of the decision rule or strategy to be used at each internal node [33].

g. **MultiLayer Perceptron:** MLP is a supervised classification technique and is a part of a feed-forward neural network. The neurons of MLP are trained with a back-propagation learning algorithm to give approximate values of continuous functions. Also, it can help to solve non-linearly separable problems. The classifier consists of three layers:

   1. Input Layer: The data flow starts from this layer.
   2. Hidden Layer: Performs all the computation of MLP, and the number of hidden layers can be arbitrary.
   3. Output Layer: Prediction and classification are performed here.

   The main use of MLP classifier is for pattern classification, prediction, approximation and Recognition.

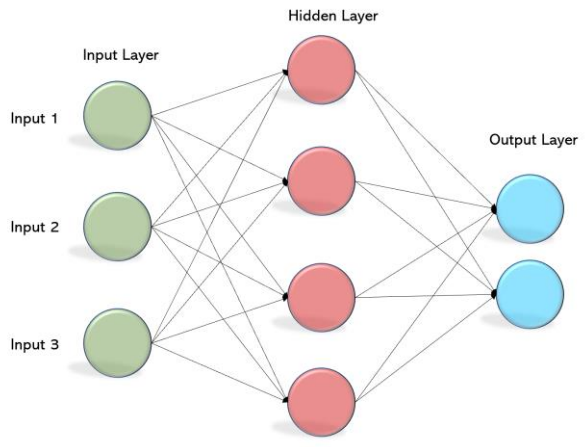


Fig. 5.4 MLP classifier structure [24]

h. **Ensemble Techniques:** The ensemble combines more than one classifier of the same or different types to achieve better accuracy for the model. In this study, an ensemble of aforementioned techniques is taken. Four types of Ensemble Techniques Used in this study:

   a. **Voting**: It combines the classification result of all the member classifiers to create a better classifier. Voting can be used for classification as well as regression. For classification, it uses the majority vote rule, i.e., more than half of the member classifiers should agree, whereas, for regression, it uses the average of all member classifier's results. We can also consider the Voting Ensemble as a meta-model as it could be used with any collection of existing trained machine learning models, and the existing models do not need to be aware that they are being used in the ensemble [7].

b. **Stacking**: It uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms [6]. There are two levels in stacking ensemble architecture:

    i. Level 0 - It contains base models (2 or more) fitted on training data, and their results are compiled for level 1.

    ii. Level 1 - It contains the meta-model, which learns how to best combine the results of the base models (Level 0).

    The prediction (output) of level 0 is fed to level 1 as input which can be real values in case of regression and probability value or class labels in case of classification.

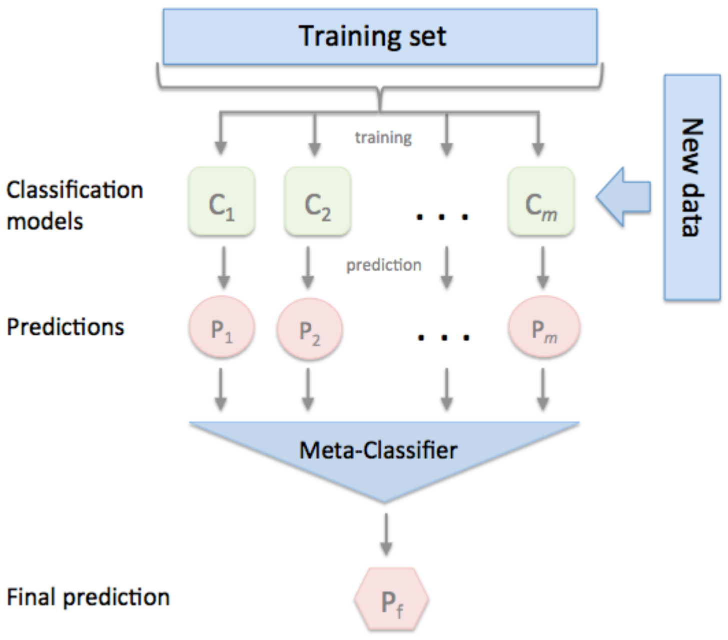


Fig. 5.5 Stacking Ensemble Working [31]

Base models can be a combination of any classification models used for that particular study. For Meta classifier, classifier that is commonly used is (but not compulsorily):

I. For Classification: Logistic Regression

II. For Regression: Linear Regression

c. **Bagging**: Bagging is short for Bootstrap Aggregation. It often considers homogeneous weak learners. Bagging learns these models parallelly and combines them using an averaging process. It generally prefers to generate an ensemble of learners having less variance than its individual components. We use Bootstrapping for Bagging Ensemble but in WEKA tool the ensemble is already developed. We use Bootstrap sampling to obtain subsets to train our base models. Bootstrap sampling is the process of using increasingly large random samples until you achieve diminishing returns in predictive accuracy. Each sample is used to train a separate decision tree, and the results of each model are aggregated [8].

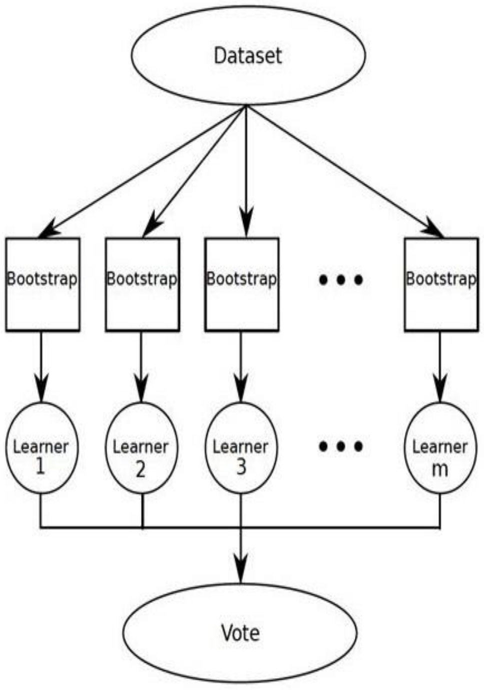


Fig. 5.6 Bagging working

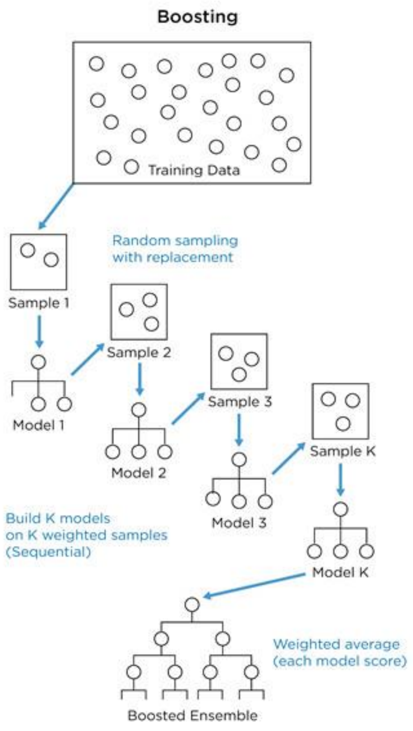

Boosting
Training Data
Random sampling with replacement
Sample 1
Model 1
Sample 2
Model 2
Sample 3
Model 3
Sample K
Model K
Build K models on K weighted samples (Sequential)
Weighted average (each model score)
Boosted Ensemble

Fig. 5.7 Boosting Working [8]

d. **Boosting**: Boosting considers homogenous weak learners to combine the results of weak learners following a deterministic strategy. It learns about the classifiers sequentially (base model depending upon previous one). AdaBoost is the most used Boosting technique. The main goal of Boosting is to fit several individual classifiers and average out their prediction to generate a model with lower bias and less variance.

5.5 **Performance Evaluation Measure**

In this study, we use **AUC (area under ROC curve)** to evaluate the prediction performance. Although the ROC (Receiver Operating Characteristics) curve is the accurate measure for prediction performance [20] [30], it does not give the numeric values to discriminate between the results, so AUC is the better choice for measuring prediction performance. AUC represents the worthiness of model predictions (Fig. 26). It is the degree of how superior a model is capable of discerning between positive and negative occurrences.

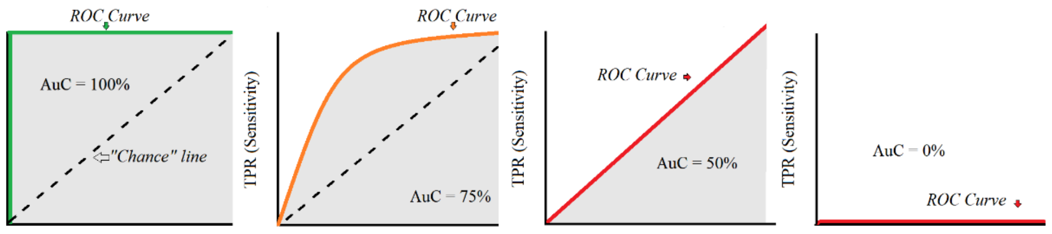


Fig. 5.8 AUC and ROC

1.0 value in AUC means model prediction is 100% accurate and 0.5 means model prediction is worthless for unknown instances prediction. AUC value is the average of all threshold values. A detailed description of how to calculate AUC values is given in Dejaeger et al. [20].

5.6 **Statistical Test:**

In this study, we use Kruskal Wallis Test to check whether there is a significant difference between the predictive performances of various classification techniques or not.

1. Kruskal Wallis Test:

It is a non-parametric test that means this test doesn't require that data should be normal. Kruskal Wallis test is used to compare the predictive performances of three or more independent samples. Independent samples mean all the sample values are defined on different datasets. The test is a non-parametric version of the one-way ANOVA test. Kruskal Wallis is the same as the Wilcoxon Mann Whitney Test as both are ranking methods; the main difference is that the Wilcoxon Mann Whitney test considers two data samples, but the Kruskal Wallis test is applied in case of three or more than three samples.

In this test, we compare the calculated H-statistics value with the tabulated chi-square value to check whether our null hypothesis is accepted or rejected.
H-statistics is calculated using the given formula:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{T_i^2}{n_i} - 3(N+1)$$

2. **Nemenyi Test:**

This test is utilized to contrast numerous procedures and each other when the sample sizes are equivalent. It is a post hoc test as it is applied if there is a rejection of the null hypothesis when we use Kruskal Wallis or Friedman test. This test is the same as the Bonferroni test as both are post hoc tests. However, the main difference is that in the Nemenyi test, we compare multiple techniques with each other, and in Bonferroni, we compare all the techniques with one control technique.

A comparison between critical distance and pairwise difference of average ranks occurs to check whether the null hypothesis is accepted or rejected. We can calculate the critical distance value using the given formula:

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

CHAPTER 6

RESULTS

Q ***Which feature selection algorithm gives the best result?***

To compare the feature selection algorithms for Bug Prediction Performance, we analyze some models using multiple classification Techniques:
Model-1: contains features selected from Gain Ratio Attribute Evaluation.
Model-2: contains features selected from Information Gain Attribute Evaluation.
Model-3: contains features selected from Relief F Attribute Evaluation.
Model-4: contains features selected from One R Attribute Evaluation.
Model-5: contains features selected from Symmetrical Uncertainty Attribute Evaluation.
Model-6: contains features selected from Correlation Attribute Evaluation.
Model-7: contains features selected from Chi Squared Attribute Evaluation.
In this section we analyze the results of all the seven models using different classifiers on the basis of AUC values.

The performance of different feature selection algorithms with different models are shown in the following tables:

Naive Bayes Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using NB are presented in this section. Table 6.1 shows the AUC values of all the seven models implemented using NB classification techniques. This table shows that in 40% of cases, CR gives better AUC values, in 33.3% cases, CS gives better AUC values, in 26.7 % cases, IG gives better AUC values, in 20% cases, RF gives better AUC values, in 13.3% cases, GR and SU gives better AUC values, and in 6.7 % cases, OR gives better AUC values.

If we statistically analyze the results of Table III using Kruskal Wallis at the 0.05 significance level, then the results show that the calculated H-value is 2.3078, and the p-value is 0.8893. $\chi 2$ value at 0.05 and k=6 is 12.5916, which is greater than the calculated H-value, so there is no significant difference between all the models implemented using NB.

Table 6.1: AUC values using NB

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	**0.825**	**0.825**	0.803	0.816	**0.825**	0.823	**0.825**
ANT 1.7	0.793	0.794	0.76	0.824	0.794	**0.825**	0.815
CAMEL 1.4	0.651	0.634	0.648	0.605	0.634	**0.689**	0.634
CAMEL 1.6	0.623	0.624	0.619	0.575	0.609	0.58	**0.644**
IVY 1.4	0.569	0.569	0.6	0.605	0.569	**0.693**	0.569
IVY 2.0	0.788	0.793	0.74	0.754	0.788	0.8	**0.805**
LUCENE 2.4	0.699	0.691	0.724	0.727	0.682	**0.731**	0.691
POI 2.5	0.726	0.819	**0.825**	0.806	0.815	0.79	0.817
POI 3.0	0.666	0.805	**0.813**	0.765	0.805	0.8	0.757
SYNAPSE 1.2	0.735	0.735	0.705	**0.768**	0.735	0.73	0.73
TOMCAT 6.0	0.794	**0.803**	0.755	0.748	**0.803**	**0.803**	**0.803**
VELOCITY 1.6	0.764	**0.766**	0.765	0.759	0.763	**0.766**	**0.766**
XALAN 2.4	**0.745**	0.74	0.723	0.651	0.74	0.686	0.744
XALAN 2.5	0.596	0.612	**0.635**	0.624	0.596	0.611	0.612
XALAN 2.6	0.784	**0.797**	0.795	0.774	0.793	0.79	**0.797**

Logical Regression Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Logistic Regression are presented in this section. Table 6.2 shows the AUC values of all the seven models implemented using Logistic Regression classification techniques. This table shows that in 40% of cases CS gives better AUC values, in 33.3 % cases CR gives better AUC values, in 20 % cases IG and SU gives better AUC values, in 13.3% cases OR gives better AUC values, and in 6.7% cases GR and RF gives better AUC values.

If we statistically analyze the results of table IV using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 2.5701 and the p-value is 0.8605. χ^2 value at 0.05 and k=5 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Logistic Regression.

Table 6.2: AUC values using LOG

Project name/ Feature Selection algorithms	**GR**	**IG**	**RF**	**OR**	**SU**	**CR**	**CS**
ANT 1.6	0.836	0.836	0.789	**0.837**	0.836	0.835	0.836
ANT 1.7	0.823	0.821	0.754	0.828	0.821	0.826	**0.83**
CAMEL 1.4	0.708	0.704	0.665	0.632	**0.704**	0.684	**0.704**
CAMEL 1.6	0.664	0.649	0.628	0.628	0.667	0.656	**0.67**
IVY 1.4	0.648	0.648	0.654	0.484	0.648	**0.772**	0.648
IVY 2.0	0.783	**0.799**	0.75	0.791	0.783	0.788	0.788
LUCENE 2.4	0.738	**0.757**	0.755	0.747	0.754	0.752	**0.757**
POI 2.5	0.706	0.797	0.786	0.785	0.805	**0.814**	0.807
POI 3.0	0.803	0.814	0.805	0.789	0.81	**0.818**	0.777
SYNAPSE 1.2	0.748	0.748	0.716	**0.789**	0.748	0.744	0.744
TOMCAT 6.0	0.806	**0.812**	0.772	0.799	**0.812**	**0.812**	**0.812**
VELOCITY 1.6	0.78	0.769	**0.801**	0.757	0.764	0.769	0.769
XALAN 2.4	0.772	0.779	0.729	0.611	0.779	0.685	**0.787**
XALAN 2.5	0.601	0.592	0.624	0.61	0.601	**0.625**	0.592
XALAN 2.6	**0.805**	0.795	0.79	0.779	**0.805**	0.799	0.795

Bayesian Network Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using NB are presented in this section. Table 6.3 shows the AUC values of all the seven models implemented using NB classification techniques. This table shows that in 33.3% of cases, CR and SU gives better AUC values, in 26.6 % cases, CS gives better AUC values, in 20 % cases, IG gives better AUC values, and in 13.3% cases, RF, OR and GR gives better AUC values.

If we statistically analyze the results of table V using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 2.3704 and the p-value is 0.8827. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using BN.

Table 6.3: AUC values using BN

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.806	0.806	0.777	0.794	0.806	**0.825**	0.806
ANT 1.7	0.815	0.807	**0.873**	0.815	0.807	0.798	0.806
CAMEL 1.4	**0.669**	0.651	0.596	0.586	0.651	0.659	0.651
CAMEL 1.6	0.591	0.653	0.49	0.509	0.652	0.639	**0.679**
IVY 1.4	0.423	0.423	0.412	0.423	0.423	**0.46**	0.423
IVY 2.0	0.764	0.774	0.638	**0.775**	0.764	0.772	0.773
LUCENE 2.4	0.679	0.684	0.682	0.689	**0.691**	0.679	0.648
POI 2.5	0.854	0.851	**0.878**	0.853	0.867	0.826	0.864
POI 3.0	0.836	0.836	0.833	**0.861**	**0.861**	0.801	0.844
SYNAPSE 1.2	**0.762**	**0.762**	0.612	0.757	**0.762**	0.728	0.728
TOMCAT 6.0	0.783	**0.79**	0.706	0.739	**0.79**	**0.79**	**0.79**
VELOCITY 1.6	0.714	**0.731**	0.697	0.71	0.728	**0.731**	**0.731**
XALAN 2.4	0.765	0.774	0.699	0.641	0.774	0.659	**0.78**
XALAN 2.5	0.64	0.635	0.64	0.645	0.64	**0.648**	0.635
XALAN 2.6	0.803	0.8	0.785	0.792	**0.808**	0.662	0.8

K Nearest Neighbor Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using KNN (K=3) are presented in this section. Table 6.4 shows the AUC values of all the seven models implemented using KNN classification techniques. This table shows that in 33.3% of cases GR gives better AUC values, in 20 % cases IG, RF and CS gives better AUC values, in 6.7% cases CR and OR gives better AUC values, and in 0% cases SU gives better AUC values.

If we statistically analyze the results of table VI using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 2.0208 and the p-value is 0.9178. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using KNN (K=3).

Table 6.4: AUC values using KNN

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.733	0.733	0.681	0.702	0.737	0.729	**0.753**
ANT 1.7	0.714	**0.728**	0.629	0.701	0.728	0.718	**0.758**
CAMEL 1.4	**0.651**	0.611	0.58	0.606	0.611	0.632	0.611
CAMEL 1.6	**0.673**	0.635	0.592	0.647	0.637	0.627	0.638
IVY 1.4	0.648	0.648	0.678	**0.68**	0.648	0.667	0.626
IVY 2.0	0.706	**0.732**	0.728	0.728	0.706	0.71	0.719
LUCENE 2.4	**0.709**	0.704	0.649	0.654	0.699	0.659	0.704
POI 2.5	0.834	0.854	0.863	0.843	0.869	**0.877**	0.862
POI 3.0	0.836	**0.847**	0.83	0.844	0.843	0.818	0.825
SYNAPSE 1.2	0.727	0.727	0.674	0.721	0.727	0.705	**0.74**
TOMCAT 6.0	**0.732**	0.709	0.688	0.61	0.709	0.709	0.716
VELOCITY 1.6	0.657	0.718	**0.719**	0.684	0.687	0.718	0.718
XALAN 2.4	**0.722**	0.688	0.679	0.687	0.688	0.61	0.668
XALAN 2.5	0.631	0.629	**0.688**	0.667	0.631	0.68	0.629
XALAN 2.6	0.765	0.777	**0.778**	0.773	0.772	0.772	0.777

Random Forest Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Random Forest are presented in this section. Table 6.5 shows the AUC values of all the seven models implemented using Random Forest classification techniques. This table shows that in 40% of cases GR gives better AUC values, in 33.3% cases, IG gives better AUC values, in 26.7% cases, SU gives better AUC values, in 20% cases, RF and CS gives better AUC values, in 13.3% cases OR gives better AUC values, and in 6.7% cases CR gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 1.8699 and the p-value is 0.9313. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Random Forest.

Table 6.5: AUC values using RFo

Project name/ Feature Selection algorithms	**GR**	**IG**	**RF**	**OR**	**SU**	**CR**	**CS**
ANT 1.6	0.797	0.797	0.797	0.797	0.797	**0.811**	0.797
ANT 1.7	**0.826**	**0.826**	0.74	0.813	**0.826**	0.807	0.808
CAMEL 1.4	0.67	**0.672**	0.623	0.595	**0.672**	0.658	**0.672**
CAMEL 1.6	0.659	**0.699**	0.647	0.628	0.698	0.64	0.682
IVY 1.4	**0.72**	**0.72**	0.604	0.691	**0.72**	0.667	**0.72**
IVY 2.0	**0.803**	0.781	0.762	0.781	**0.803**	0.801	0.801
LUCENE 2.4	**0.763**	0.757	0.72	0.728	0.751	0.746	0.757
POI 2.5	0.877	0.878	**0.893**	0.857	0.89	0.874	0.864
POI 3.0	0.863	0.87	0.851	**0.878**	0.869	0.835	0.868
SYNAPSE 1.2	0.755	0.755	0.761	**0.793**	0.755	0.786	0.786
TOMCAT 6.0	**0.799**	0.797	0.708	0.753	0.797	0.797	0.797
VELOCITY 1.6	0.757	0.746	**0.769**	0.743	0.756	0.746	0.746
XALAN 2.4	**0.777**	0.748	0.753	0.719	0.748	0.668	0.759
XALAN 2.5	0.611	0.681	**0.731**	0.716	0.683	0.707	0.681
XALAN 2.6	0.794	**0.812**	0.811	0.796	0.797	0.792	**0.812**

Decision Tree Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Decision Tree are presented in this section. Table 6.6 shows the AUC values of all the seven models implemented using Decision Tree classification techniques. This table shows that in 26.7% of cases RF gives better AUC values, in 20% cases, IG, OR, CR and CS gives better AUC values, in 13.3% cases, SU gives better AUC values, and in 6.7% cases GR gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 0.8131 and the p-value is 0.9917. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Decision Tree.

Table 6.6: AUC values using DT

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.764	0.764	0.714	0.764	0.764	**0.782**	0.764
ANT 1.7	0.745	0.734	0.723	**0.753**	0.734	0.737	0.744
CAMEL 1.4	0.64	**0.642**	0.593	0.582	**0.642**	0.611	**0.642**
CAMEL 1.6	0.612	0.615	0.575	0.586	0.598	0.595	**0.637**
IVY 1.4	0.436	0.436	0.439	**0.45**	0.436	0.433	0.436
IVY 2.0	0.67	0.629	**0.689**	0.588	0.67	0.617	0.617
LUCENE 2.4	0.676	0.676	0.671	0.648	0.644	**0.69**	0.676
POI 2.5	0.838	0.825	**0.861**	0.809	0.841	0.823	0.835
POI 3.0	0.812	**0.837**	0.797	0.804	0.712	0.808	0.827
SYNAPSE 1.2	0.674	0.674	0.672	**0.746**	0.674	0.732	0.732
TOMCAT 6.0	0.715	**0.734**	0.595	0.73	**0.734**	**0.734**	**0.734**
VELOCITY 1.6	0.717	0.648	**0.728**	0.687	0.681	0.648	0.648
XALAN 2.4	**0.731**	0.706	0.599	0.655	0.706	0.634	0.712
XALAN 2.5	0.644	0.631	**0.698**	0.684	0.644	0.662	0.631
XALAN 2.6	**0.787**	0.752	0.76	0.777	0.767	0.767	0.752

MultiLayer Perceptron Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using MultiLayer Perceptron are presented in this section. Table 6.7 shows the AUC values of all the seven models implemented using MultiLayer Perceptron classification techniques. This table shows that in 26.7% of cases RF and OR gives better AUC values, in 20% cases, GR gives better AUC values, in 13.3% cases, CS gives better AUC values, in 6.7% cases, IG and CR gives better AUC values, and in 0% cases SU gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 0.9278 and the p-value is 0.9882. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using MultiLayer Perceptron.

Table 6.7: AUC values using MLP

Project name/ Feature Selection algorithms	**GR**	**IG**	**RF**	**OR**	**SU**	**CR**	**CS**
ANT 1.6	0.809	0.809	0.766	**0.821**	0.809	0.784	0.809
ANT 1.7	0.805	0.802	0.737	0.807	0.802	0.806	**0.814**
CAMEL 1.4	**0.705**	0.696	0.625	0.609	0.696	0.683	0.696
CAMEL 1.6	0.633	0.623	0.634	**0.639**	0.607	0.621	0.631
IVY 1.4	0.626	0.626	0.671	0.532	0.626	**0.733**	0.626
IVY 2.0	0.769	**0.773**	0.752	0.771	0.769	0.701	0.764
LUCENE 2.4	0.716	0.724	**0.753**	0.718	0.739	0.723	0.724
POI 2.5	0.818	0.835	**0.845**	0.825	0.804	0.815	0.843
POI 3.0	0.808	0.801	0.782	0.811	0.802	0.784	**0.817**
SYNAPSE 1.2	0.729	0.729	0.757	**0.761**	0.729	0.718	0.718
TOMCAT 6.0	0.787	0.791	0.786	**0.796**	0.791	0.791	0.791
VELOCITY 1.6	0.758	0.754	**0.775**	0.734	0.75	0.754	0.754
XALAN 2.4	**0.789**	0.782	0.702	0.68	0.782	0.775	0.785
XALAN 2.5	0.615	0.6	**0.678**	0.667	0.615	0.589	0.6
XALAN 2.6	**0.807**	0.795	0.796	0.782	0.804	0.801	0.795

Voting Ensemble Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Voting Ensemble are presented in this section. Table 6.8 shows the AUC values of all the seven models implemented using Voting Ensemble classification techniques. This table shows that in 33.3% of cases CR gives better AUC values, in 26.7% cases, GR gives better AUC values, in 20% cases, CS and IG gives better AUC values, and in 13.3% cases RF, OR and SU gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 1.8283 and the p-value is 0.9348. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Voting Ensemble.

Table 6.8: AUC values using VE

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.818	0.818	0.81	0.814	0.818	**0.835**	0.818
ANT 1.7	**0.833**	0.832	0.779	0.822	0.832	0.822	0.82
CAMEL 1.4	**0.707**	0.688	0.659	0.646	0.688	0.688	0.688
CAMEL 1.6	0.701	0.708	0.658	0.677	0.695	0.673	**0.723**
IVY 1.4	0.67	0.67	0.589	0.643	0.67	**0.725**	0.67
IVY 2.0	0.806	0.805	0.771	0.8	0.806	**0.81**	0.808
LUCENE 2.4	0.757	**0.758**	0.746	0.738	0.756	0.751	**0.758**
POI 2.5	0.697	0.697	0.689	**0.715**	0.705	0.707	0.696
POI 3.0	0.851	**0.863**	0.847	0.861	0.862	0.852	0.853
SYNAPSE 1.2	0.782	0.782	0.766	**0.814**	0.782	0.798	0.798
TOMCAT 6.0	**0.815**	**0.815**	0.767	0.785	**0.815**	**0.815**	**0.815**
VELOCITY 1.6	0.765	0.767	**0.796**	0.756	0.766	0.767	0.767
XALAN 2.4	0.804	0.774	0.753	0.729	0.774	**0.786**	0.785
XALAN 2.5	0.685	0.68	**0.733**	0.72	0.685	0.714	0.68
XALAN 2.6	**0.819**	0.814	0.816	0.815	**0.819**	0.815	0.814

Stacking Ensemble Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Stacking Ensemble are presented in this section. Table 6.9 shows the AUC values of all the seven models implemented using Stacking Ensemble classification techniques. This table shows that in 33.3% of cases SU gives better AUC values, in 26.7% cases, IG gives better AUC values, in 20% cases, CR and CS gives better AUC values, and in 13.3% cases GR, RF and OR gives better AUC values.
If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level, then results show that the calculated H-value is 1.5887 and the p-value is 0.9534. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Stacking Ensemble.

Table 6.9: AUC values using SE

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.813	0.813	0.786	0.798	0.813	**0.825**	0.813
ANT 1.7	0.825	**0.828**	0.771	0.812	**0.828**	0.813	0.826
CAMEL 1.4	**0.694**	0.675	0.655	0.622	0.675	0.673	0.675
CAMEL 1.6	0.673	**0.687**	0.659	0.657	0.686	0.637	0.676
IVY 1.4	0.53	0.53	0.533	0.504	0.53	**0.621**	0.53
IVY 2.0	0.762	0.754	0.728	0.723	0.762	0.741	**0.766**
LUCENE 2.4	0.758	**0.763**	0.741	0.738	**0.763**	0.745	**0.763**
POI 2.5	0.876	0.879	0.885	0.859	**0.892**	0.877	0.87
POI 3.0	0.855	0.86	0.844	**0.868**	0.864	0.839	0.856
SYNAPSE 1.2	0.773	0.773	0.755	**0.806**	0.773	0.786	0.786
TOMCAT 6.0	0.753	**0.797**	0.658	0.75	**0.797**	**0.797**	**0.797**
VELOCITY 1.6	0.751	0.728	**0.777**	0.741	0.748	0.728	0.728
XALAN 2.4	**0.78**	0.766	0.761	0.698	0.766	0.763	0.771
XALAN 2.5	0.677	0.675	**0.729**	0.712	0.677	0.714	0.675
XALAN 2.6	0.815	0.813	0.815	0.81	**0.82**	0.808	0.813

Bagging Ensemble Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Bagging Ensemble are presented in this section. Table 6.10 shows the AUC values of all the seven models implemented using Bagging Ensemble classification techniques. This table shows that in 26.7% of cases IG, SU and CR gives better AUC values, in 20% cases, CS gives better AUC values, in 13.3% cases, GR and OR gives better AUC values, and in 6.7% cases RF gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 2.0695 and the p-value is 0.9132. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Bagging Ensemble.

Table 6.10: AUC values using BE

Project name/ Feature Selection algorithms	GR	IG	RF	OR	SU	CR	CS
ANT 1.6	0.823	0.823	0.812	0.82	0.823	**0.828**	0.823
ANT 1.7	0.832	**0.834**	0.776	0.826	**0.834**	0.828	0.831
CAMEL 1.4	**0.713**	0.696	0.668	0.647	0.696	0.686	0.696
CAMEL 1.6	0.701	0.716	0.667	0.675	0.708	0.684	**0.721**
IVY 1.4	0.663	0.663	0.628	0.671	0.663	**0.716**	0.663
IVY 2.0	0.808	0.806	0.775	0.816	0.808	**0.82**	0.816
LUCENE 2.4	0.763	**0.772**	0.75	0.753	0.768	0.759	**0.772**
POI 2.5	0.87	0.875	0.886	0.87	**0.887**	0.87	0.875
POI 3.0	0.848	**0.869**	0.847	0.858	0.868	0.851	0.853
SYNAPSE 1.2	0.787	0.787	0.767	**0.814**	0.787	0.797	0.797
TOMCAT 6.0	0.827	**0.828**	0.768	0.789	**0.828**	**0.828**	**0.828**
VELOCITY 1.6	0.758	0.763	**0.786**	0.753	0.76	0.763	0.763
XALAN 2.4	**0.803**	0.785	0.753	0.753	0.785	0.786	0.784
XALAN 2.5	0.69	0.691	0.741	**0.746**	0.724	0.723	0.722
XALAN 2.6	0.816	0.817	0.816	0.817	**0.822**	0.818	0.818

Boosting Ensemble Analysis:

The results of analyzing all the seven models to check which models give the best result for bug prediction using Boosting Ensemble are presented in this section. Table 6.11 shows the AUC values of all the seven models implemented using Boosting Ensemble classification techniques. This table shows that in 33.3% of cases SU gives better AUC values, in 26.7% cases, GR gives better AUC values, in 20% cases, RF gives better AUC values, in 13.3% cases, CR and CS gives better AUC values, and in 6.7% cases IG and OR gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 0.8456 and the p-value is 0.9908. χ^2 value at 0.05 and k=6 is 12.5916 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Boosting Ensemble.

Table 6.11: AUC values using BoE

Project name/ Feature Selection algorithms	**GR**	**IG**	**RF**	**OR**	**SU**	**CR**	**CS**
ANT 1.6	0.821	0.821	0.791	0.809	0.821	**0.829**	0.821
ANT 1.7	0.815	**0.827**	0.778	0.81	**0.827**	0.807	0.802
CAMEL 1.4	**0.713**	0.681	0.654	0.636	0.681	0.687	0.681
CAMEL 1.6	0.673	0.71	0.669	0.67	0.687	0.663	**0.727**
IVY 1.4	0.68	0.68	0.596	0.723	0.68	**0.748**	0.68
IVY 2.0	**0.798**	0.783	0.783	0.788	**0.798**	0.777	0.796
LUCENE 2.4	0.754	0.753	0.736	0.737	**0.768**	0.738	0.753
POI 2.5	0.857	0.865	**0.892**	0.864	0.883	0.875	0.867
POI 3.0	0.845	0.848	0.838	0.858	**0.865**	0.843	0.854
SYNAPSE 1.2	0.782	0.782	0.766	**0.809**	0.782	0.797	0.797
TOMCAT 6.0	**0.814**	0.797	0.756	0.769	0.797	0.797	0.797
VELOCITY 1.6	0.743	0.746	**0.785**	0.729	0.745	0.746	0.746
XALAN 2.4	**0.777**	0.774	0.763	0.744	0.744	0.766	**0.777**
XALAN 2.5	0.668	0.677	**0.737**	0.694	0.668	0.694	0.677
XALAN 2.6	0.801	0.808	0.809	0.799	**0.811**	0.803	0.808

Q ***What is the effect of feature selection algorithms on fault proneness models?***

Previous studies have shown that the effect of feature selection algorithms on fault proneness models is positive. The power of the models has increased, and faultier modules, functions or classes can be accurately classified as faulty or not.

The comparison done by Zhou et al. clearly shows the increase in power of bug prediction models with wrapper and filter methods showing better results than clustering-based feature selection algorithms.

These models' power can be increased by 90 by eliminating redundant or less useful features.

Q ***Which classifier or ensemble technique performs the best?***

To compare the classifiers for Bug Prediction Performance, we analyze some models using multiple feature selection algorithms Techniques:

Model-1: classification using Bayesian Network.

Model-2: classification using Naive Bayes.

Model-3: classification using Logistic Regression.
Model-4: classification using Random Forest.
Model-5: classification using K-Nearest Neighbor.
Model-6: classification using Decision Tree.
Model-7: classification using MultiLayer Perceptron.
Model-8: classification using Voting Ensemble.
Model-9: classification using Stacking Ensemble.
Model-10: classification using Bagging Ensemble.
Model-11: classification using Boosting Ensemble.
In this section we analyze the results of all the eleven models using different feature selection algorithms on the basis of AUC values.

The performance of different classifiers with different feature selection algorithms are shown in the following tables:

Gain Ratio Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Gain Ratio are presented in this section. Table 6.12 shows the AUC values of all the eleven models implemented using Gain Ratio classification techniques. This table shows that in 46.7% of cases BE gives better AUC values.
If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 16.0398 and the p-value is 0.0985. χ^2 value at 0.05 and k=10 is 18.307 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Gain Ratio.

Information Gain Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Information Gain are presented in this section. Table 6.13 shows the AUC values of all the eleven models implemented using Information Gain classification techniques. This table shows that in 60% of cases BE gives better AUC values.
If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 15.9929 and the p-value is 0.09984.

χ^2 value at 0.05 and k=10 is 18.307 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Information Gain.

Relief F Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Relief F are presented in this section. Table 6.14 shows the AUC values of all the eleven models implemented using Relief F classification techniques. This table shows that in 33.3% of cases BE gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 15.4822 and the p-value is 0.1154. χ^2 value at 0.05 and k=10 is 18.307 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Relief F.

Table 6.12: AUC values using GR

Project name/ Classifier	**BN**	**NB**	**LOG**	**RFo**	**KNN**	**DT**	**MLP**	**VE**	**SE**	**BE**	**BoE**
ANT 1.6	0.806	0.825	**0.836**	0.797	0.733	0.764	0.809	0.818	0.813	0.823	0.821
ANT 1.7	0.815	0.793	0.823	0.826	0.714	0.745	0.805	**0.833**	0.825	0.832	0.815
CAMEL 1.4	0.669	0.651	0.708	0.67	0.651	0.64	0.705	0.707	0.694	**0.713**	**0.713**
CAMEL 1.6	0.591	0.623	0.664	0.659	0.673	0.612	0.633	**0.701**	0.673	**0.701**	0.673
IVY 1.4	0.423	0.569	0.648	**0.72**	0.648	0.436	0.626	0.67	0.53	0.663	0.68
IVY 2.0	0.764	0.788	0.783	0.803	0.706	0.67	0.769	0.806	0.762	**0.808**	0.798
LUCENE 2.4	0.679	0.699	0.738	**0.763**	0.709	0.676	0.716	0.757	0.758	**0.763**	0.754
POI 2.5	0.854	0.726	0.706	**0.877**	0.834	0.838	0.818	0.697	0.876	0.87	0.857
POI 3.0	0.836	0.666	0.803	**0.863**	0.836	0.812	0.808	0.851	0.855	0.848	0.845
SYNAPSE 1.2	0.762	0.735	0.748	0.755	0.727	0.674	0.729	0.782	0.773	**0.787**	0.782
TOMCAT 6.0	0.783	0.794	0.806	0.799	0.732	0.715	0.787	0.815	0.753	**0.827**	0.814
VELOCITY 1.6	0.714	0.764	**0.78**	0.757	0.657	0.717	0.758	0.765	0.751	0.758	0.743
XALAN 2.4	0.765	0.745	0.772	0.777	0.722	0.731	0.789	**0.804**	0.78	0.803	0.777
XALAN 2.5	0.64	0.596	0.601	0.611	0.631	0.644	0.615	0.685	0.677	**0.69**	0.668
XALAN 2.6	0.803	0.784	0.805	0.794	0.765	0.787	0.807	**0.819**	0.815	0.816	0.801

Table 6.13: AUC values using IG

Project name/ Classifier	BN	NB	LOG	RFo	KNN	DT	MLP	VE	SE	BE	BoE
ANT 1.6	0.806	0.825	**0.836**	0.797	0.733	0.764	0.809	0.818	0.813	0.823	0.821
ANT 1.7	0.807	0.794	0.821	0.826	0.728	0.734	0.802	0.832	0.828	**0.834**	0.827
CAMEL 1.4	0.651	0.634	**0.704**	0.672	0.611	0.642	0.696	0.688	0.675	0.696	0.681
CAMEL 1.6	0.653	0.624	0.649	0.699	0.635	0.615	0.623	0.708	0.687	**0.716**	0.71
IVY 1.4	0.423	0.569	0.648	**0.72**	0.648	0.436	0.626	0.67	0.53	0.663	0.68
IVY 2.0	0.774	0.793	0.799	0.781	0.732	0.629	0.773	0.805	0.754	**0.806**	0.783
LUCENE 2.4	0.684	0.691	0.757	0.757	0.704	0.676	0.724	0.758	0.763	**0.772**	0.753
POI 2.5	0.851	0.819	0.797	0.878	0.854	0.825	0.835	0.697	**0.879**	0.875	0.865
POI 3.0	0.836	0.805	0.814	**0.87**	0.847	0.837	0.801	0.863	0.86	0.869	0.848
SYNAPSE 1.2	0.762	0.735	0.748	0.755	0.727	0.674	0.729	0.782	0.773	**0.787**	0.782
TOMCAT 6.0	0.79	0.803	0.812	0.797	0.709	0.734	0.791	0.815	0.797	**0.828**	0.797
VELOCITY 1.6	0.731	0.766	**0.769**	0.746	0.718	0.648	0.754	0.767	0.728	0.763	0.746
XALAN 2.4	0.774	0.74	0.779	0.748	0.688	0.706	0.782	0.774	0.766	**0.785**	0.774
XALAN 2.5	0.635	0.612	0.592	0.681	0.629	0.631	0.6	0.68	0.675	**0.691**	0.677
XALAN 2.6	0.8	0.797	0.795	0.812	0.777	0.752	0.795	0.814	0.813	**0.817**	0.808

Table 6.14: AUC values using RF

Project name/ Classifier	BN	NB	LOG	RFo	KNN	DT	MLP	VE	SE	BE	BoE
ANT 1.6	0.777	0.803	0.789	0.797	0.681	0.714	0.766	0.81	0.786	**0.812**	0.791
ANT 1.7	**0.873**	0.76	0.754	0.74	0.629	0.723	0.737	0.779	0.771	0.776	0.778
CAMEL 1.4	0.596	0.648	0.665	0.623	0.58	0.593	0.625	0.659	0.655	**0.668**	0.654
CAMEL 1.6	0.49	0.619	0.628	0.647	0.592	0.575	0.634	0.658	0.659	0.667	**0.669**
IVY 1.4	0.412	0.6	0.654	0.604	**0.678**	0.439	0.671	0.589	0.533	0.628	0.596
IVY 2.0	0.638	0.74	0.75	0.762	0.728	0.689	0.752	0.771	0.728	0.775	**0.783**
LUCENE 2.4	0.682	0.724	**0.755**	0.72	0.649	0.671	0.753	0.746	0.741	0.75	0.736
POI 2.5	0.878	0.825	0.786	**0.893**	0.863	0.861	0.845	0.689	0.885	0.886	0.892
POI 3.0	0.833	0.813	0.805	**0.851**	0.83	0.797	0.782	0.847	0.844	0.847	0.838
SYNAPSE 1.2	0.612	0.705	0.716	0.761	0.674	0.672	0.757	0.766	0.755	**0.767**	0.766
TOMCAT 6.0	0.706	0.755	0.772	0.708	0.688	0.595	**0.786**	0.767	0.658	0.768	0.756
VELOCITY 1.6	0.697	0.765	**0.801**	0.769	0.719	0.728	0.775	0.796	0.777	0.786	0.785
XALAN 2.4	0.699	0.723	0.729	0.753	0.679	0.599	0.702	0.753	0.761	0.753	**0.763**
XALAN 2.5	0.64	0.635	0.624	0.731	0.688	0.698	0.678	0.733	0.729	**0.741**	0.737
XALAN 2.6	0.785	0.795	0.79	0.811	0.778	0.76	0.796	**0.816**	0.815	**0.816**	0.809

One R Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using One R are presented in this section. Table 6.15 shows the AUC values of all the eleven models implemented using One R classification techniques. This table shows that in 53.3% of cases BE gives better AUC values.

If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 13.7615 and the p-value is 0.1842. χ^2 value at 0.05 and k=10 is 18.307 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using One R.

Table 6.15: AUC values using OR

Project name/ Classifier	**BN**	**NB**	**LOG**	**RFo**	**KNN**	**DT**	**MLP**	**VE**	**SE**	**BE**	**BoE**
ANT 1.6	0.794	0.816	**0.837**	0.797	0.702	0.764	0.821	0.814	0.798	0.82	0.809
ANT 1.7	0.815	0.824	**0.828**	0.813	0.701	0.753	0.807	0.822	0.812	0.826	0.81
CAMEL 1.4	0.586	0.605	0.632	0.595	0.606	0.582	0.609	0.646	0.622	**0.647**	0.636
CAMEL 1.6	0.509	0.575	0.628	0.628	0.647	0.586	0.639	**0.677**	0.657	0.675	0.67
IVY 1.4	0.423	0.605	0.484	0.691	0.68	0.45	0.532	0.643	0.504	0.671	**0.723**
IVY 2.0	0.775	0.754	0.791	0.781	0.728	0.588	0.771	0.8	0.723	**0.816**	0.788
LUCENE 2.4	0.689	0.727	0.747	0.728	0.654	0.648	0.718	0.738	0.738	**0.753**	0.737
POI 2.5	0.853	0.806	0.785	0.857	0.843	0.809	0.825	0.715	0.859	**0.87**	0.864
POI 3.0	0.861	0.765	0.789	**0.878**	0.844	0.804	0.811	0.861	0.868	0.858	0.858
SYNAPSE 1.2	0.757	0.768	0.789	0.793	0.721	0.746	0.761	**0.814**	0.806	**0.814**	0.809
TOMCAT 6.0	0.739	0.748	**0.799**	0.753	0.61	0.73	0.796	0.785	0.75	0.789	0.769
VELOCITY 1.6	0.71	**0.759**	0.757	0.743	0.684	0.687	0.734	0.756	0.741	0.753	0.729
XALAN 2.4	0.641	0.651	0.611	0.719	0.687	0.655	0.68	0.729	0.698	**0.753**	0.744
XALAN 2.5	0.645	0.624	0.61	0.716	0.667	0.684	0.667	0.72	0.712	**0.746**	0.694
XALAN 2.6	0.792	0.774	0.779	0.796	0.773	0.777	0.782	0.815	0.81	**0.817**	0.799

Symmetrical Uncertainty Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Symmetrical Uncertainty are presented in this section. Table 6.16 shows the AUC values of all the eleven models implemented using Symmetrical Uncertainty classification techniques. This table shows that in 60% of cases BE gives better AUC values.

If we statistically analyze the results of table 6.16 using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 19.0802 and the p-value is 0.03926. χ^2 value at 0.05 and k=10 is 18.307 which is less than the calculated H-value, so there is a significant difference between all the models implemented using Symmetrical Uncertainty. As the results differ significantly, we will apply Nemenyi test to check the pairwise comparison of the techniques. The results show that there exist significant difference between BN and BE, NB and BE, LOG and DT, RFo and DT, KNN and VE, KNN and BE, KNN and BoE, DT and VE, DT and SE, DT and BE, DT and BoE, and MLP and BE.

Table 6.16: AUC values using SU

Project name/ Classifier	**BN**	**NB**	**LOG**	**RFo**	**KNN**	**DT**	**MLP**	**VE**	**SE**	**BE**	**BoE**
ANT 1.6	0.806	0.825	**0.836**	0.797	0.737	0.764	0.809	0.818	0.813	0.823	0.821
ANT 1.7	0.807	0.794	0.821	0.826	0.728	0.734	0.802	0.832	0.828	**0.834**	0.827
CAMEL 1.4	0.651	0.634	**0.704**	0.672	0.611	0.642	0.696	0.688	0.675	0.696	0.681
CAMEL 1.6	0.652	0.609	0.667	0.698	0.637	0.598	0.607	0.695	0.686	**0.708**	0.687
IVY 1.4	0.423	0.569	0.648	**0.72**	0.648	0.436	0.626	0.67	0.53	0.663	0.68
IVY 2.0	0.764	0.788	0.783	0.803	0.706	0.67	0.769	0.806	0.762	**0.808**	0.798
LUCENE 2.4	0.691	0.682	0.754	0.751	0.699	0.644	0.739	0.756	0.763	**0.768**	**0.768**
POI 2.5	0.867	0.815	0.805	0.89	0.869	0.841	0.804	0.705	**0.892**	0.887	0.883
POI 3.0	0.861	0.805	0.81	**0.869**	0.843	0.712	0.802	0.862	0.864	0.868	0.865
SYNAPSE 1.2	0.762	0.735	0.748	0.755	0.727	0.674	0.729	0.782	0.773	**0.787**	0.782
TOMCAT 6.0	0.79	0.803	0.812	0.797	0.709	0.734	0.791	0.815	0.797	**0.828**	0.797
VELOCITY 1.6	0.728	0.763	0.764	0.756	0.687	0.681	0.75	**0.766**	0.748	0.76	0.745
XALAN 2.4	0.774	0.74	0.779	0.748	0.688	0.706	0.782	0.774	0.766	**0.785**	0.744
XALAN 2.5	0.64	0.596	0.601	0.683	0.631	0.644	0.615	0.685	0.677	**0.724**	0.668
XALAN 2.6	0.808	0.793	0.805	0.797	0.772	0.767	0.804	0.819	0.82	**0.822**	0.811

Correlation Attribute Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Correlation Attribute are presented in this section. Table 6.17 shows the AUC values of all the eleven models implemented using Correlation Attribute classification techniques. This table shows that in 53.3% of cases BE gives better AUC values.

As the results differ significantly, we will apply Nemenyi test to check the pairwise comparison of the models. The results show that there exists significant difference between BN and BE, LOG and DT, KNN and VE, KNN and BE, KNN and BoE, DT and VE, DT and BE, DT and BoE, MLP and VE, and MLP and BE

Table 6.17: AUC values using CR

Project name/ Classifier	**BN**	**NB**	**LOG**	**RFo**	**KNN**	**DT**	**MLP**	**VE**	**SE**	**BE**	**BoE**
ANT 1.6	0.825	0.823	**0.835**	0.811	0.729	0.782	0.784	**0.835**	0.825	0.828	0.829
ANT 1.7	0.798	0.825	0.826	0.807	0.718	0.737	0.806	0.822	0.813	**0.828**	0.807
CAMEL 1.4	0.659	**0.689**	0.684	0.658	0.632	0.611	0.683	0.688	0.673	0.686	0.687
CAMEL 1.6	0.639	0.58	0.656	0.64	0.627	0.595	0.621	0.673	0.637	**0.684**	0.663
IVY 1.4	0.46	0.693	**0.772**	0.667	0.667	0.433	0.733	0.725	0.621	0.716	0.748
IVY 2.0	0.772	0.8	0.788	0.801	0.71	0.617	0.701	0.81	0.741	**0.82**	0.777
LUCENE 2.4	0.679	0.731	0.752	0.746	0.659	0.69	0.723	0.751	0.745	**0.759**	0.738
POI 2.5	0.826	0.79	0.814	0.874	**0.877**	0.823	0.815	0.707	**0.877**	0.87	0.875
POI 3.0	0.801	0.8	0.818	0.835	0.818	0.808	0.784	**0.852**	0.839	0.851	0.843
SYNAPSE 1.2	0.728	0.73	0.744	0.786	0.705	0.732	0.718	**0.798**	0.786	0.797	0.797
TOMCAT 6.0	0.79	0.803	0.812	0.797	0.709	0.734	0.791	0.815	0.797	**0.828**	0.797
VELOCITY 1.6	0.731	0.766	**0.769**	0.746	0.718	0.648	0.754	0.767	0.728	0.763	0.746
XALAN 2.4	0.659	0.686	0.685	0.668	0.61	0.634	0.775	**0.786**	0.763	**0.786**	0.766
XALAN 2.5	0.648	0.611	0.625	0.707	0.68	0.662	0.589	0.714	0.714	**0.723**	0.694
XALAN 2.6	0.662	0.79	0.799	0.792	0.772	0.767	0.801	0.815	0.808	**0.818**	0.803

Chi Square Analysis:

The results of analyzing all the eleven models to check which models give the best result for bug prediction using Chi Square are presented in this section. Table 6.18 shows the AUC values of all the eleven models implemented using Chi Square classification techniques. This table shows that in 46.7% of cases BE gives better AUC values.
If we statistically analyze the results of table VII using Kruskal Wallis at 0.05 significance level then results show that the calculated H-value is 16.3288 and the p-value is 0.0906. χ^2 value at 0.05 and k=10 is 18.307 which is greater than the calculated H-value, so there is no significant difference between all the models implemented using Chi Square.

Table 6.18: AUC values using CS

Project name/ Classifier	BN	NB	LOG	RFo	KNN	DT	MLP	VE	SE	BE	BoE
ANT 1.6	0.806	0.825	**0.836**	0.797	0.753	0.764	0.809	0.818	0.813	0.823	0.821
ANT 1.7	0.806	0.815	0.83	0.808	0.758	0.744	0.814	0.82	0.826	**0.831**	0.802
CAMEL 1.4	0.651	0.634	**0.704**	0.672	0.611	0.642	0.696	0.688	0.675	0.696	0.681
CAMEL 1.6	0.679	0.644	0.67	0.682	0.638	0.637	0.631	0.723	0.676	0.721	**0.727**
IVY 1.4	0.423	0.569	0.648	**0.72**	0.626	0.436	0.626	0.67	0.53	0.663	0.68
IVY 2.0	0.773	0.805	0.788	0.801	0.719	0.617	0.764	0.808	0.766	**0.816**	0.796
LUCENE 2.4	0.648	0.691	0.757	0.757	0.704	0.676	0.724	0.758	0.763	**0.772**	0.753
POI 2.5	0.864	0.817	0.807	0.864	0.862	0.835	0.843	0.696	0.87	**0.875**	0.867
POI 3.0	0.844	0.757	0.777	**0.868**	0.825	0.827	0.817	0.853	0.856	0.853	0.854
SYNAPSE 1.2	0.728	0.73	0.744	0.786	0.74	0.732	0.718	**0.798**	0.786	0.797	0.797
TOMCAT 6.0	0.79	0.803	0.812	0.797	0.716	0.734	0.791	0.815	0.797	**0.828**	0.797
VELOCITY 1.6	0.731	0.766	**0.769**	0.746	0.718	0.648	0.754	0.767	0.728	0.763	0.746
XALAN 2.4	0.78	0.744	**0.787**	0.759	0.668	0.712	0.785	0.785	0.771	0.784	0.777
XALAN 2.5	0.635	0.612	0.592	0.681	0.629	0.631	0.6	0.68	0.675	**0.722**	0.677
XALAN 2.6	0.8	0.797	0.795	0.812	0.777	0.752	0.795	0.814	0.813	**0.818**	0.808

CHAPTER 7

DISCUSSION ON RESULTS

In this section, we analyse the results using box plots. We analysed different types of feature selection to check the Bug prediction power of the model. When using the Bayesian network, Naive Bayes, Voting Ensemble and Bagging Ensemble, Correlation Attribute Evaluation gives better results than the rest of the feature selection algorithms. When the Kruskal Wallis Test was applied, the result concluded that there is no significant difference between the feature selection algorithms using any classifiers.

Naive Bayes analysis:

Fig 7.1 represents the box plot using NB. We can see that CR Attribute Evaluation performed better than all other Attribute Evaluation algorithms.

Logistic Regression analysis:

Fig 7.2 represents the box plot using LOG. We can see that CS Attribute Evaluation performed better than all other Attribute Evaluation algorithms.

Bayesian Network analysis:

Fig 7.3 represents the box plot using BN. We can see that CR and SU Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 2 models, 1 outlier in GR model and 1 outlier in IG model only.

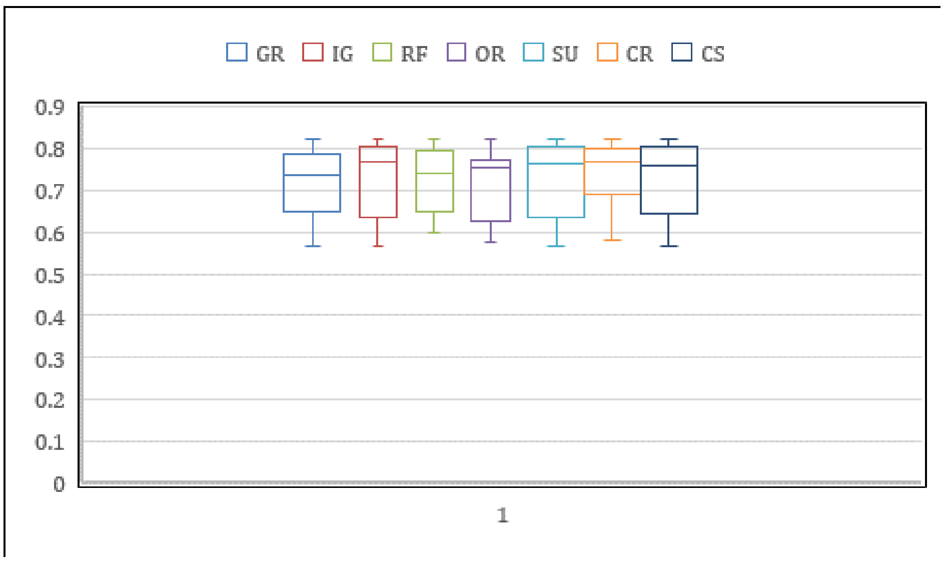


Fig 7.1. Box plot of NB

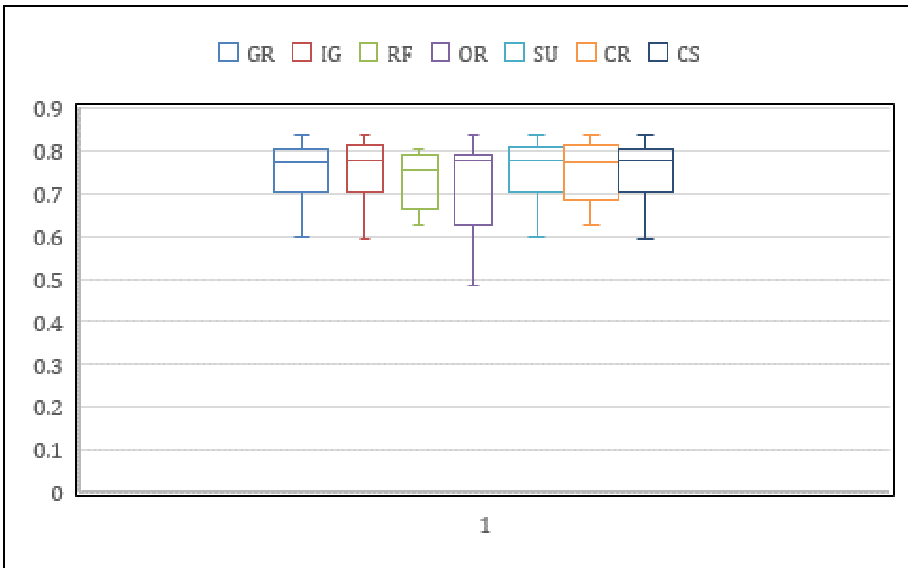


Fig 7.2. Box plot of LOG

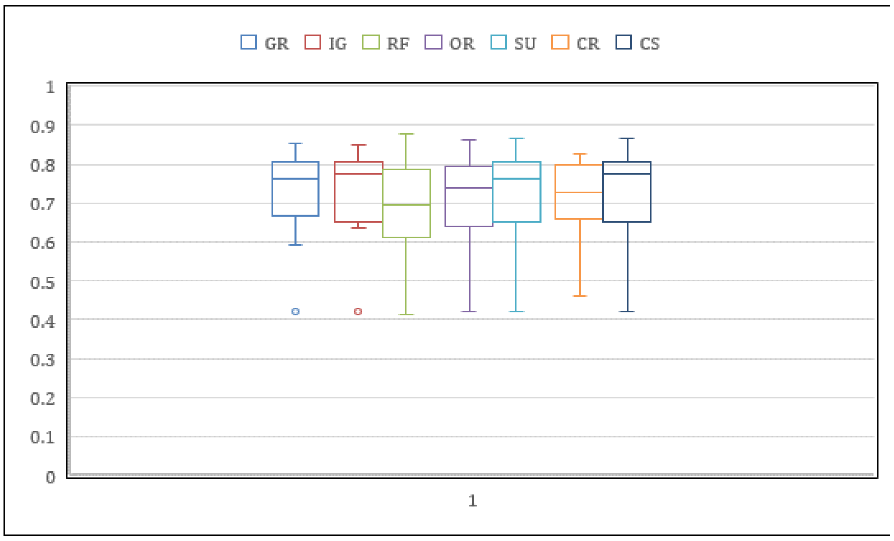


Fig 7.3. Box plot of BN

K Nearest Neighbor analysis:

Fig 7.4 represents the box plot using KNN. We can see that GR Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 3 models, 1 outlier in RF model, 1 outlier in OR and 1 outlier in CR model only.

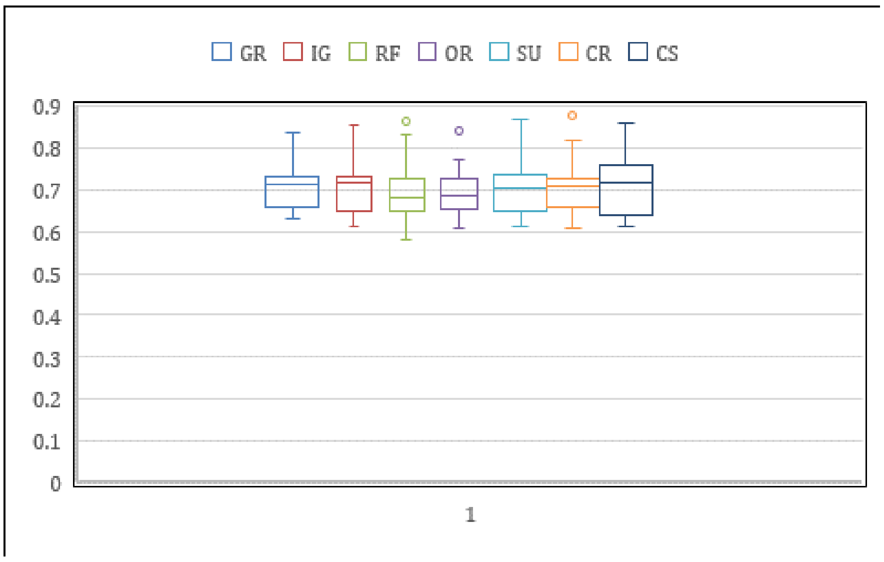


Fig 7.4. Box plot of KNN

Random Forest analysis:

Fig 7.5 represents the box plot using RFo. We can see that GR Attribute Evaluation performed better than all other Attribute Evaluation algorithms.

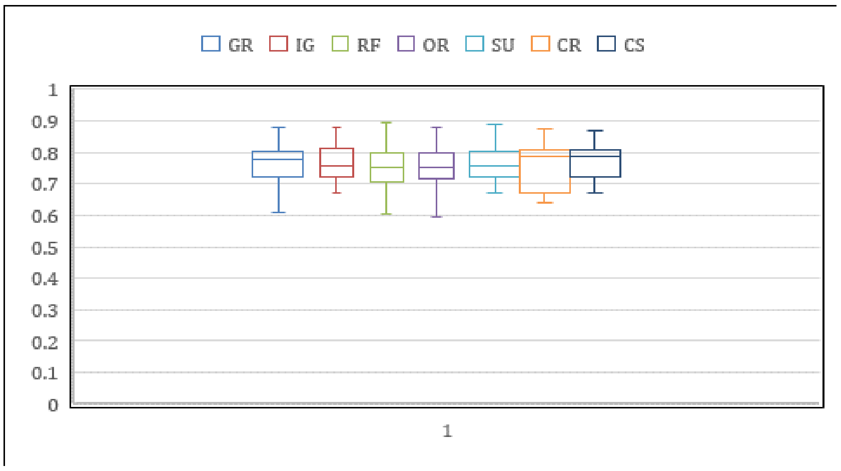


Fig 7.5. Box plot of RFo

Decision Tree analysis:

Fig 7.6 represents the box plot using DT. We can see that RF Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 4 models, 1 outlier in GR model, 1 outlier in IG, 1 outlier in SU and 1 outlier in CS model only.

MultiLayer Perceptron analysis:

Fig 7.7 represents the box plot using MLP. We can see that RF and OR Attribute Evaluation performed better than all other Attribute Evaluation algorithms.

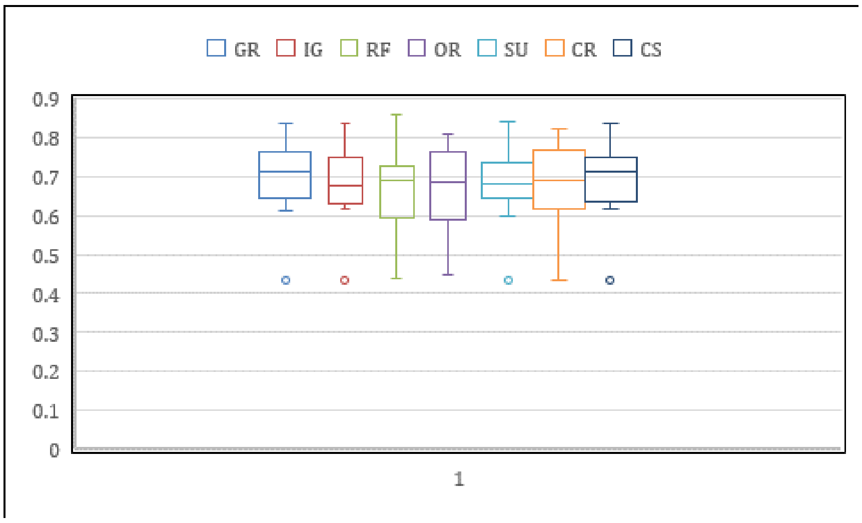


Fig 7.6. Box plot of DT

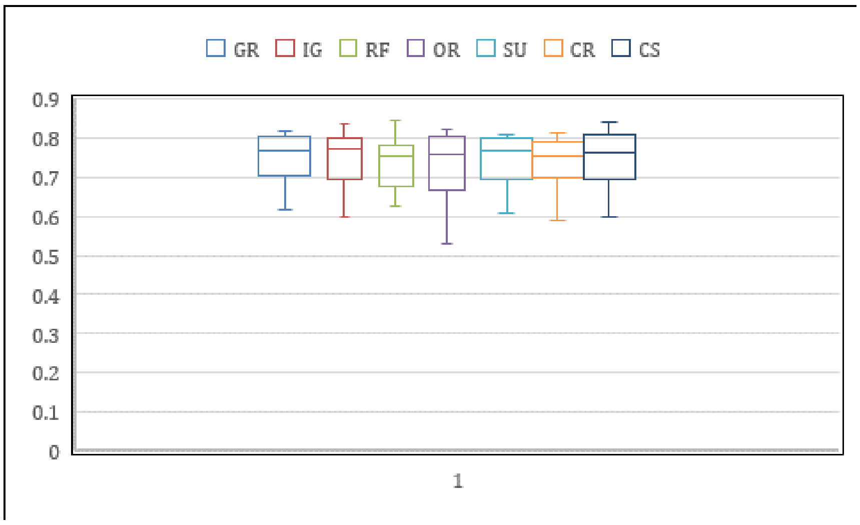


Fig 7.7. Box plot of MLP

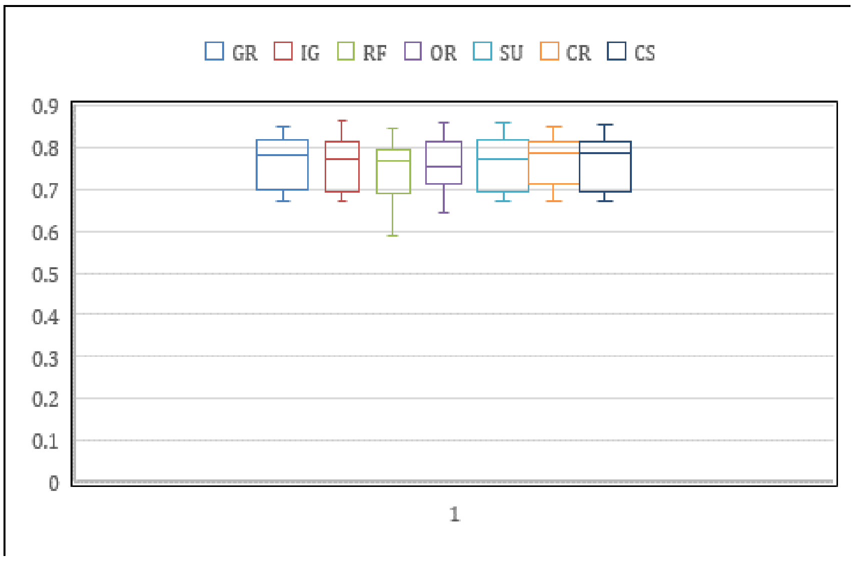


Fig 7.8. Box plot of VE

Voting Ensemble analysis:

Fig 7.8 represents the box plot using VE. We can see that CR Attribute Evaluation performed better than all other Attribute Evaluation algorithms.

Stacking Ensemble analysis:

Fig 7.9 represents the box plot using SE. We can see that SU Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 1 model, 1 outlier in OR model only.

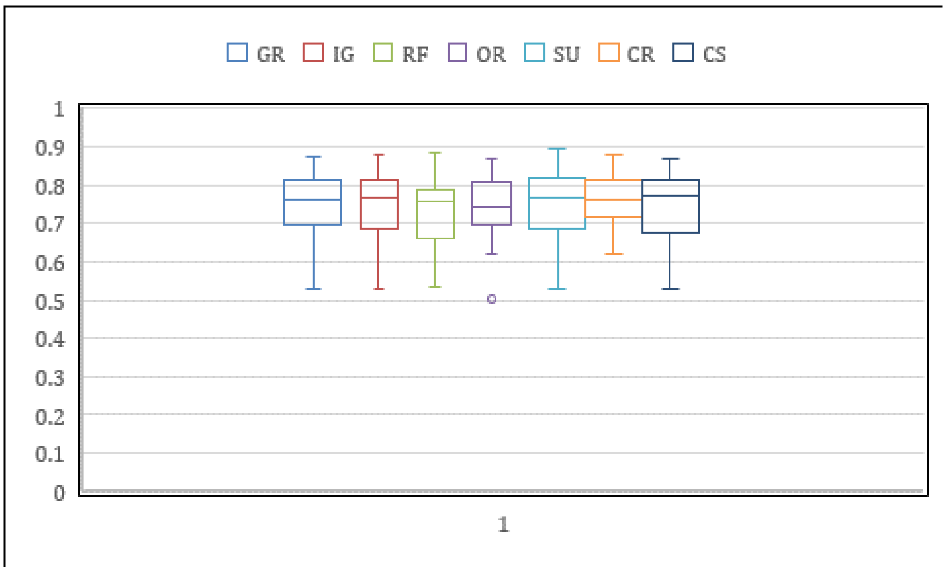


Fig 7.9. Box plot of SE

Bagging Ensemble analysis:

Fig 7.10 represents the box plot using BE. We can see that IG, SU and CR Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 1 model, 1 outlier in RF model only.

Boosting Ensemble analysis:

Fig 7.11 represents the box plot using BoE. We can see that SU Attribute Evaluation performed better than all other Attribute Evaluation algorithms and there is an outlier in 1 model, 2 outliers in RF model only.

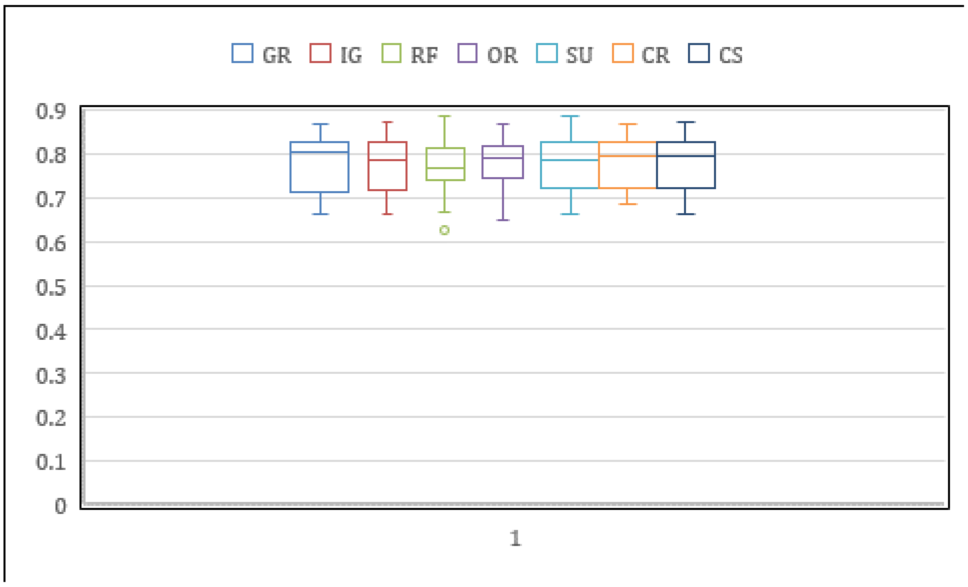


Fig 7.10. Box plot of BE

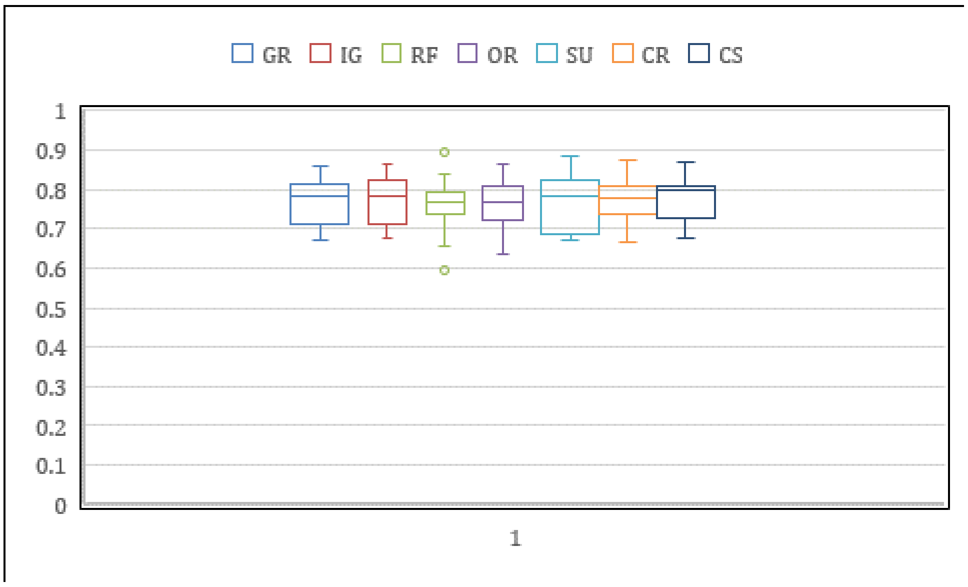


Fig 7.11. Box plot of BoE

We also analysed different types of classifiers and ensemble techniques to check the Bug prediction power of the model. Bagging Ensemble gives better results with all seven feature selection algorithms as compared to rest of the classifiers and ensemble techniques. When the Kruskal Wallis Test was applied, the result concluded that there is a significant difference between classifiers using SU and CR feature selection algorithm.

Gain Ratio Analysis:

Fig 7.12 represents the box plot using GR. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 2 models, 1 outlier in BN model and 1 outlier in DT model only.

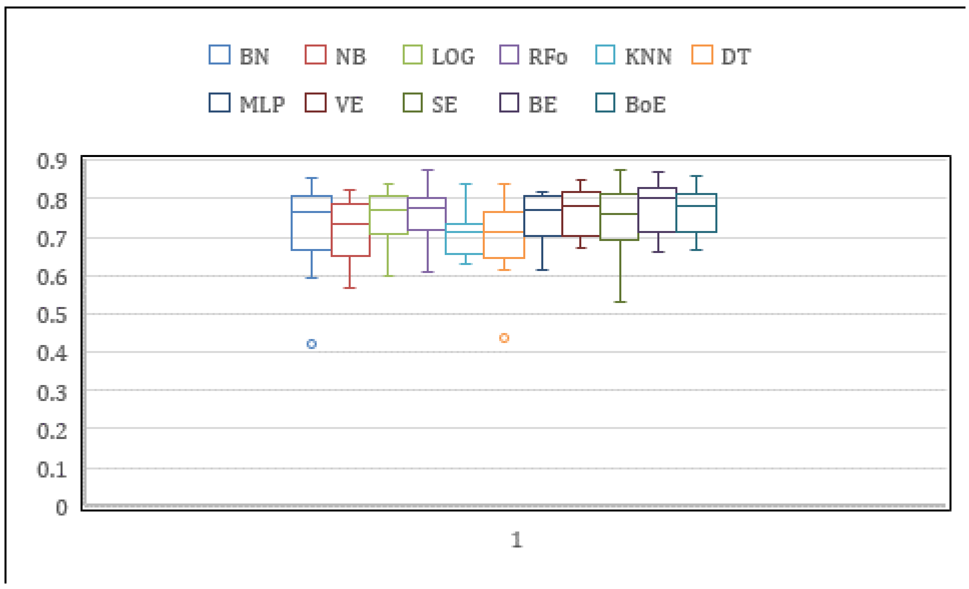


Fig 7.12. Box plot of GR

Information Gain Analysis:

Fig 7.13 represents the box plot using IG. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 2 models, 1 outlier in BN model and 1 outlier in DT model only.

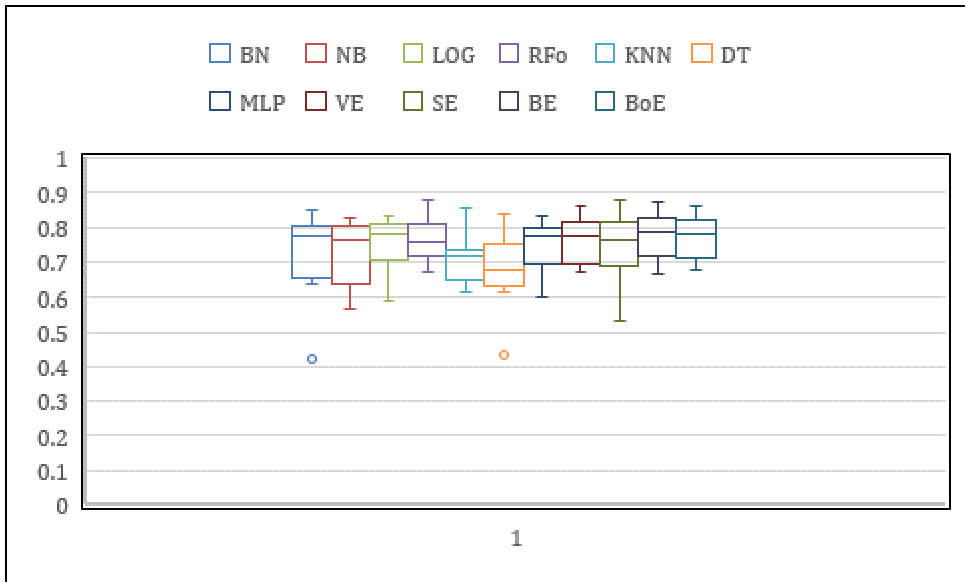


Fig 7.13. Box plot of IG

Relief F Analysis:

Fig 7.14 represents the box plot using RF. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 2 models, 1 outlier in the BE model and 1 outlier in the BoE model only.

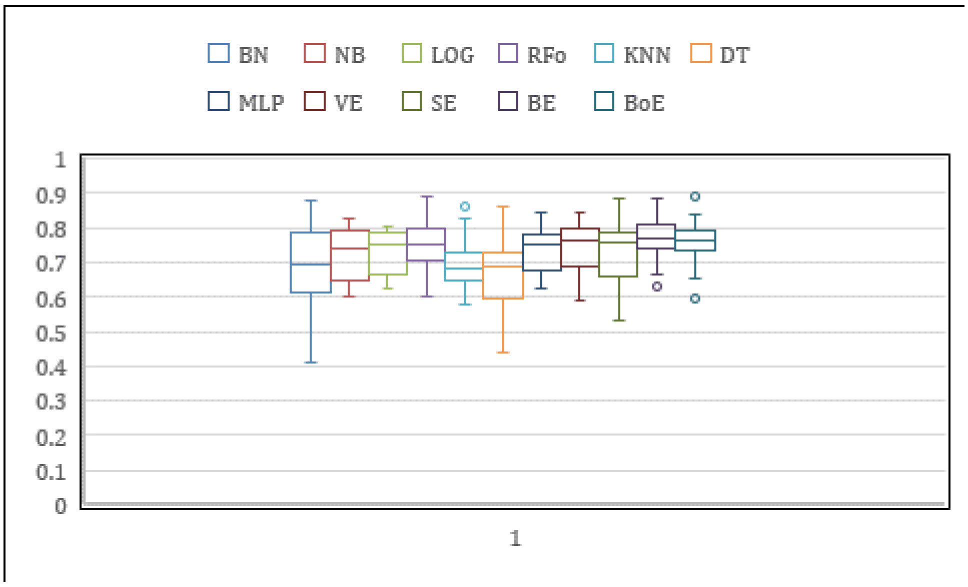


Fig 7.14. Box plot of RF

One R Analysis:

Fig 7.15 represents the box plot using OR. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 1 model, 1 outlier in SE model only.

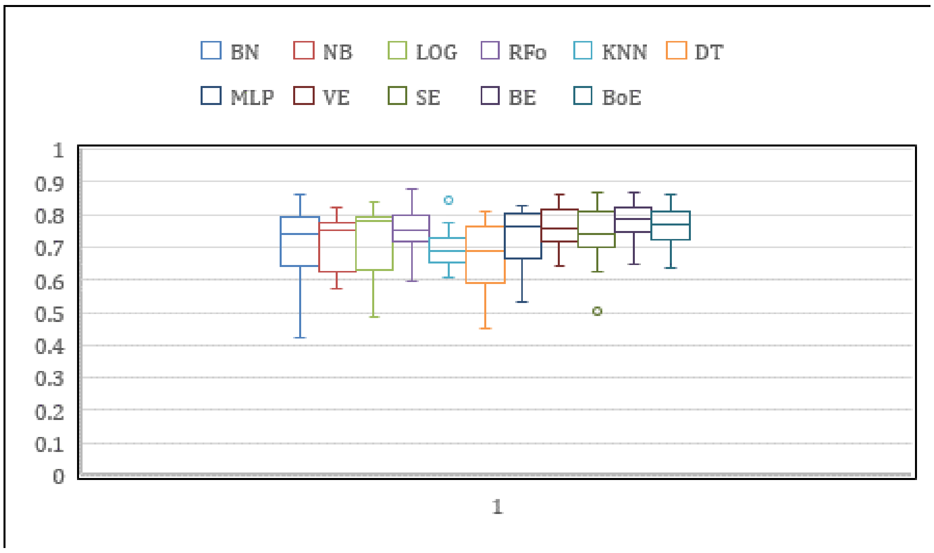


Fig 7.15. Box plot of OR

Symmetrical Uncertainty Analysis:

Fig 7.16 represents the box plot using SU. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 1 model, 1 outlier in DT model only.

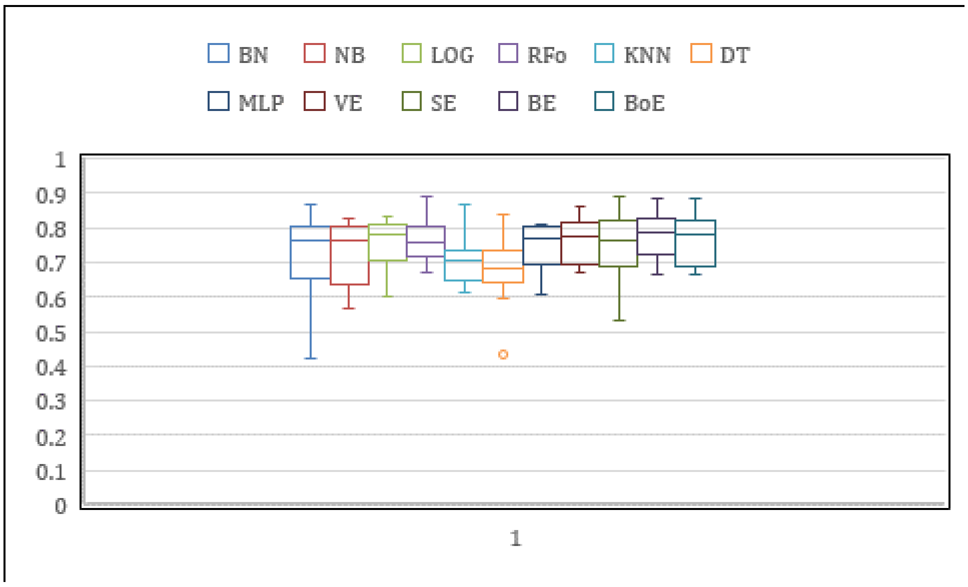


Fig 7.16. Box plot of SU

Correlation Attribute Analysis:

Fig 7.17 represents the box plot using CR. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 1 model, 1 outlier in KNN model only.

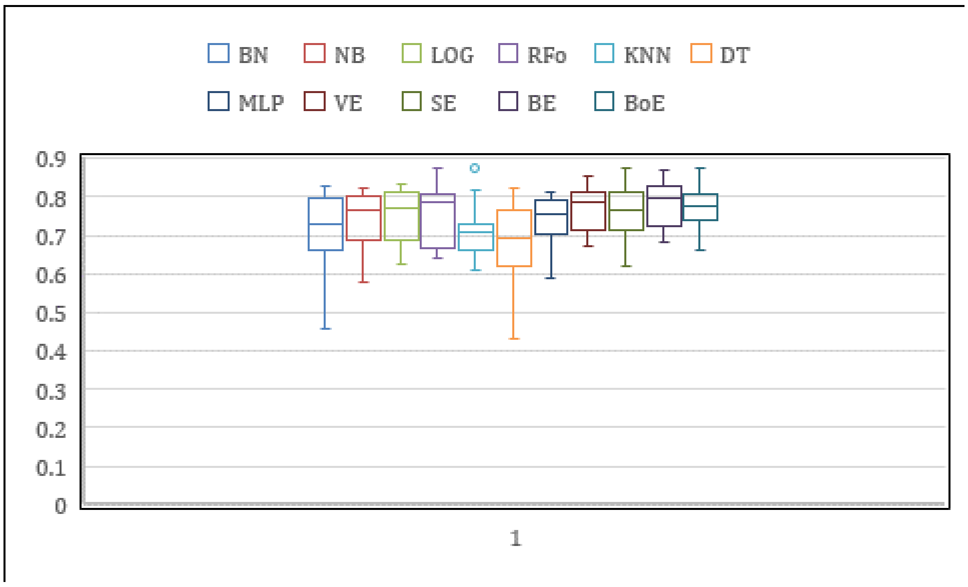


Fig 7.17. Box plot of CR

Chi Square Analysis:

Fig 7.18 represents the box plot using CS. We can see that BE performed better than all other classifiers and ensemble techniques and there is an outlier in 1 model, 1 outlier in DT model only.

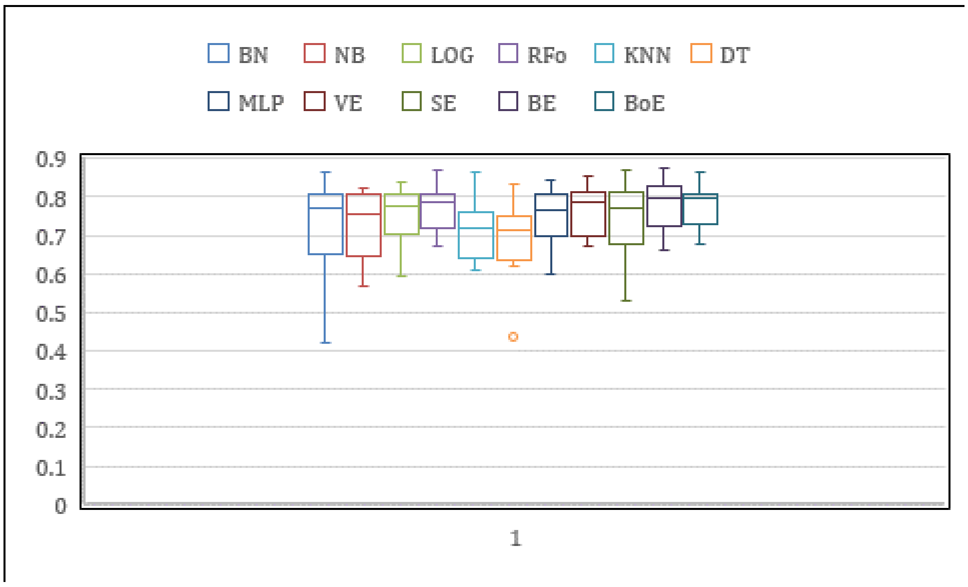


Fig 7.18. Box plot of CS

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Many models exist and give good results for Software Bug Prediction. Here we are analyzing the effect of feature selection algorithms on bug proneness using multiple classification techniques. The following conclusion has been made:

- The performance of Software Bug Prediction models is increased by 90% using Feature Selection algorithms.
- In this study, we concluded that Correlation Attribute Evaluation and Symmetric Uncertainty Attribute Evaluation are better predictors as compared to other filter feature selection algorithms.
- Chi Square, One R and Information Gain Attribute Evaluation performs the worst among the filter feature selection algorithms.
- We can also conclude that Bagging Ensemble Techniques gives the best result with every filter feature selection algorithm used individually.
- While using SU and CR feature selection algorithms, there exist significant difference between various techniques.

The future scope of this project is as follows:

- This study can further be improved by using more classifiers and feature selection algorithms that may or may not be of filter type.
- New Ensemble techniques can also be used to check its effect on bug proneness.
- Also ensemble of feature selection techniques can be used for comparison.

REFERENCES

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

[2] 777-> A. V. Kulkarni and L. N. Kanal, "An optimization approach to hierarchical classifier design," Proc. 3rd Int. Joint Conf. on Pattern Recognition, San Diego, CA, 1976.

[3] A. (2020, December 02). Feature Selection Techniques in Machine Learning. Retrieved September 09, 2020, from https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

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

[5] Brownlee, J. (2020, August 20). *How to Choose a Feature Selection Method For Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[6] Brownlee, J. (2020a, April 10). *Stacking Ensemble Machine Learning With Python*. Machine Learning Mastery. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

[7] Brownlee, J. (2020a, April 17). *How to Develop Voting Ensembles With Python*. Machine Learning Mastery. https://machinelearningmastery.com/voting-ensembles-with-python/#:%7E:text=A%20voting%20ensemble%20

[8] Budzik, J. (n.d.). *Many Heads Are Better Than One: The Case For Ensemble Learning*. KDnuggets. Retrieved June 1, 2021, from https://www.kdnuggets.com/2019/09/ensemble-learning.html

[9] Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16-28

[10] Chowdary, D. H. (2020, May 28). *Decision Trees Explained with a Practical Example*. Towards AI — The Best of Tech, Science, and Engineering. https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53

[11] Chugh, A. (2018, December 24). *ML | Chi-square Test for feature selection*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-chi-square-test-for-feature-selection/

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

[13] Diomidis Spinellis. Tool writing: A forgotten art? IEEE Software, 22(4):9–11, July/August 2005. (doi:10.1109/MS.2005.111).

[14] Gajawada, S. K. (2019, October 4). *Chi-Square Test for Feature Selection in Machine learning*. Medium. https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223

[15] Ghotra, B., McIntosh, S., & Hassan, A. E. (2015, May). Revisiting the impact of classification techniques on the performance of defect prediction models. In Proceedings of the 37th International Conference on Software Engineering-Volume 1 (pp. 789-800). IEEE Press

[16] Hall, M.A., and Smith, L.A., "Practical feature subset selection for machine learning", Proceedings of the 21st Australian Computer Science Conference, 1998, 181–191.

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

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

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

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

[21] Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 34(4), 485-496.

[22] Madeyski, L. (2011). Software Defect Prediction. Retrieved November 12, 2020, from https://madeyski.e-informatyka.pl/tools/software-defect-prediction/

[23] Mohanty, A. (2019, May 15). *Multi layer Perceptron (MLP) Models on Real World Banking Data*. Medium. https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

[24] M. W. Kurzynski, "Decision rules for a hierarchical classifier," Pattern Recognition Lett. vol. 1, 305-310, (1983)

[25] M. W. Kurzynski, "The optimal strategy of a tree classifier," Pattern Recognition vol. 16, 81-87 (1983).

[26] Novaković, J. (2016). Toward optimal feature selection using ranking methods and classification algorithms. Yugoslav Journal of Operations Research, 21(1).

[27] Panichella, A., Oliveto, R., & De Lucia, A. (2014, February). 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). IEEE.

[28] 555->Petrakova, A., Affenzeller, M., & Merkurjeva, G. (2015). Heterogeneous versus homogeneous machine learning ensembles. *Information Technology and Management Science*, *18*(1), 135-140.

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

[30] Raschka, S. (2020). *StackingClassifier - mlxtend*. Mlxtend. http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/#:%7E:text=Stacking%20is%20an%20ensemble%20learning,models%20via%20a%20meta-classifier.&text=The%20meta-classifier%20can%20either,or%20probabilities%20from%20the%20ensemble

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

[32] Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, *21*(3), 660-674.

[33] Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data quality: Some comments on the nasa software defect datasets. IEEE Transactions on Software Engineering, 39 (9), 1208-1215.

[34] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim. Reducing features to improve code change-based bug prediction. IEEE Transactions on Software Engineering, 39(4): 552-569, 2013.

[35] Tiwari, R., & Singh, M. P. (2010). Correlation-based attribute selection using genetic algorithms. *International Journal of Computer Applications*, *4*(8), 28-34.

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

[37] Wikipedia contributors. (2021, May 15). *Feature (machine learning)*. Wikipedia. https://en.wikipedia.org/wiki/Feature_(machine_learning)

[38] Xu, Z., Liu, J., Yang, Z., An, G., & Jia, X. (2016, October). The impact of feature selection on defect prediction performance: An empirical comparison. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 309-320). IEEE.

[39] Yiu, T. (2019, August 14). Understanding Random Forest. Retrieved September 05, 2020, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2