Abstract: In recent decades, as the size of the software products increases, the prediction of software metrics also increasing exponentially. The main objective of the quality software product is to significantly detect the software defects with minimum time and cost. It aims to make a meaningful impact on memory space and cost. A realistic assessment of metrics has significance importance in the prediction of software defects. Traditional software detection models are implemented in order to assess the metrics in the source code level. Neural Network (NN), SVM, Naïve Bayes, logistic regression approaches as well as statistical approaches are used to assess the software metrics for defect prediction. These models are implemented to classify and cluster the software metrics in different software databases. Also, these approaches are used to enhancing the software quality to a great extent. Most of the traditional software defect prediction models are focused on Within-Company Defects Prediction (WCDP). One of the major limitations is that, lack of training information in the early phases of software testing process. Thus, the proposed feature selection based prediction model is used to assess the Cross-Company Defects Prediction (CCDP) on multiple defect databases. Proposed clustering based defect prediction model is used to group the new defects in the cross company defects for defect prediction. Experimental results proved that the proposed model has high computational accuracy compared to the existing software defect prediction models in terms of time and memory is concerned.

Keywords: Software defect prediction, classification model, Feature selection measures.

1. Introduction

During the last two decades, different feature selection methods and classification models have been used for prediction of different defect datasets. Today, with the exponential growth of technology, it is merely impossible to use these conventional methods for defect prediction due to the high dimensional features and imbalance properties. Both machine learning and conventional methods suffer from this problem. This problem can be resolved by either decreasing the number of variables or increasing the number of training datasets. Sample-feature ratio must be more than 5:1 every time. Machine learning algorithms can be categorized into three broad types, they are:- Supervised machine learning, Unsupervised machine learning and Reinforcement machine learning. Supervised learning consists of a prescient provider which provides the labelled training dataset as input to the algorithm and produces output after mapping. But on the contrary, for unsupervised machine learning only training datasets are given as input without labels. Some of the examples of unsupervised learning are:- Self Organizing feature Maps, hierarchical clustering, k-means clustering, and so on. All machine learning algorithms used for cancer prediction come under supervised learning. Most widely applicable algorithms are:- Artificial Neural Networks, Decision Trees, Genetic algorithms, Linear Discriminant Analysis, k-Nearest Neighbour etc.

Feature sub-set selection approaches basically extract subset of features from the large feature space. The software quality has been measured using static and dynamic software defects analysis. To predict the fault in software, there should be a good understanding of the static and dynamic code metrics. In this static code metrics, few of them are highly relevant features with the defects, but others have less number. So, it is not a mandatory to use minimal defect features; the chosen features should be detected and employed in the prediction process. Classification technique is considered as one of the efficient technique among all traditional software defect detection models.

Before this, credibility theory was never implemented during the process of software defects predictions and detections. Series of experiments are carried out in order to evaluate the defect hypothesis and a new approach is introduced which is known as credibility theory based Naïve Bayes classification technique for limited small datasets. The objective of this approach is to achieve target of Cross-Project Defect Prediction. Apart from this, proposed technique is capable to discriminate data distribution among source and target data. The credibility factor plays very important role in order to evaluate the degree of reweighting source data. It also analyses the extent of transfer knowledge starting from target to
source. This technique assists the source data in order to include all the features of a new or small-scale application. The attributes are modified and adjusted during this process and the data distribution remains same.

Traditional feature clustering models are basically executed in two phases. In the initial phase, the process of feature clustering is carried out on single product defects. Here, a new k-medoids clustering approach is presented which completely depends upon Correlation measure. In the subsequent phase, this framework carries out feature extraction process. At first, all the instances are partitioned into k numbers of clusters. All relevant and useful features are extracted from each individual cluster. This selection process requires fuzzy cluster relevance measure for feature subset selection. According to the empirical validation process, it can be showed that this framework achieves better performance with the help of relative feature ranking scheme.

The final outcomes of the above framework are given below:-

(1) The process of feature clustering is capable of reducing the redundancy of the selected feature subset. It is possible through the process of Information Gain or Chi-Square.

(2) Irrespective of FC-Relevance measure implementation, feature clustering is capable to enhance the performance of defect predictors.

(3) Mostly the above framework uses Information Gain FC-Relevance measure.

Fuzzy Integral Based on Mutual Information for Software Defect Prediction:

Most of the softwares can be modified and repaired after the development process, but this modification is much more costly. Hence, it is suggested to determine all defects at the time of early development. Mostly, 80% of defects are included within 20% of software modules. This skewness is most complicated issue in the defect identification process. A large number of modules are actually non-defective and the classifier has the responsibility to detect only defective modules. It is considered as a common problem in the process of machine learning and the complete process is known as learning from imbalanced datasets. The dataset which is completely skewed towards majority class are needed to pre-process before applying the classifiers to normalize the minority classes. In certain cases, minority class can be detected along with the defective instances. The defective instances of minority class are much higher as compared to the majority class. As a result, most of the classification models generates high accuracy in case of majority class, whereas less accuracy in case of minority class. A novel technique to handle the imbalance property is known as stratification-based resampling technique (SMOTE). Here, the emphasis is given on SMOTE approach which is basically an approach responsible for oversampling minority-class examples. It considers the synthetic patterns of minority class and discards the duplication of already existing ones.

Most of the existing software defect detection models are broadly categorized into two types, those are:- static prediction model and dynamic prediction model. The static prediction technique involves complex metrics along with software scale in order to detect the fault-prone components. PCA based decision tree approach is a special type of static software defect classification approach. This approach is used to reduce the dimensions of software metric in large feature space. Additionally, it resolves all the issues of data redundancy and it also obtains a set of low-dimensional linear non-redundant data. This scheme is implemented in order to discriminate the defective classes from the non-defective ones. It is considered as an advanced machine learning approach which is generally used to resolve data modeling issues. It relates primary software metrics with the overall probability of availability of defects.

Improved quality software can be generated by proper testing and debugging process. Mostly, developers never have sufficient resources in order to put an extra effort in the testing and debugging phases. With presence of enough training data, all of these Multiple machine learning approaches can be merged with each other in order to develop an efficient classification technique known as ensemble model. Ensemble classification model analyses the applicability of existing hybrid approaches for defect prediction.

The main limitations of the traditional ensemble classification models on high dimension datasets are given below.

- This approach is not suitable for high dimensional features and sparse datasets.
- Feature selection is one of the major issue in the traditional classification models. As the size of the attributes increases , it is difficult to classify the test samples with atleast 10% of attributes.
• Large amount of data are essential part of machine learning schemes like deep learning. It is very difficult and hectic to analyze, evaluate and manage such a vast amount of data.

2. Related Works

The traditional EM algorithm can be modified and extended in order to develop an optimized algorithm known as Optimized- EM learning technique. Optimized EM learning is used to learn the parameters of hierarchical statistical approaches. This technique involves numbers of iterations with E-step and M-step. Apart from this, it also assists the backpropagation technique to optimize the initialization objective function. During the E-step, inferences for the latent model variables are predicted using the efficient sampling techniques. This method is used to extract samples from shared latent variables. Weights are associated with all samples and it denotes the relevance of samples with original posterior distribution. The quality of the resulted outcomes of selection measure completely depends upon the distance among the initialization and the targeted distributions. If the initiation point and the target point are very far from each other, then it will become more difficult for the samples to reach the exact target point [2].

X. Xia, et.al, presented a new method and termed it as HYDRA [2]. It is basically a Massively Compositional Model for Cross-Project Defect Prediction. The labeled data from a target project is referred to as training target data. In HYDRA, there are two phases: 1) genetic algorithm (GA) phase and 2) ensemble learning (EL) phase. In the GA phase, they first build a classifier for each source project. All the data are merged with the training target data. Next, they created a GA classifier by assigning different weights to the multiple classifiers. The genetic algorithm is responsible to find out the best weights which optimize F-score on the training target data.

It has another additional responsibility for reducing training error in approximation of the generalization error. There are not enough instances in training target data which can be divided into training and validation sets. In the EL phase, they integrated multiple iterations of GA. For each and every iteration, they built individual GA classifier. Then, every individual classifier is associated with certain weights. Weights are assigned depending upon the prediction error rate of the training target data. The numbers of instances are also increased in source projects as well as the training target data. This type of situation occurs when the process of classification is not carried out properly by the GA classifier which was built in the original initial iteration.

[3], developed an advanced classifier ensemble framework in order to classify huge amount of multimedia big data. There exist a numbers of different classification approaches for different datatypes. A single classifier can’t give optimized results in all types of datasets. Hence, an ensemble learning approaches have become very much popular. Generally accuracy of the ensemble learning approaches depend on base classifiers and data dimensionality. In this work, an advanced classifier ensemble framework is developed along with a group of predefined base classifiers. The main objective of this approach is to classify the test instances based on the majority voting of multiple base classifiers.

[4] proposed a model to resolve the issues of evolutionary unsampling for imbalanced big data classification [2]. Most of the applications require appropriate classification approaches in order to classify big data. The dimensionality issue of microarray data has become more complex in case of imbalanced big data classification process. A proper solution for imbalanced classification is evolutionary ensemble approaches. In this paper, an optimized MapReduce approach is implemented to classify imbalanced data. The whole process can be broadly divided into two phase MapReduce. In the initial phase, a decision tree is learnt on every individual map after successful completion of pre-processing and prediction of test samples using trained dataset. A windowing approach is applied in order to enhance the system’s performance. In future, this model will be extended to implement hybrid oversampling or undersampling techniques instead of EUS in order to enhance the system’s performance.

Z. A. Rana, et.al proposed an enhanced method known as improving recall of software defect prediction models using association mining [5]. Since many years, software metrics play vital role in the process of software defect detection. The existing association relationship among software product metrics and software defects are considered and analyzed in order to enhance the overall performance of NB classifier. They presented a preprocessing technique which is responsible for discretization of data in order to evaluate the associations of software metrics and defects. Initially, all the data are partitioned class-wise (into Pt and Pf). It is responsible for producing frequent itemsets for every individual partition.

The itemset has numbers of abnormalities and also known as focused itemset. Depending upon real itemset, they introduced a new pre-processing technique which is
responsible for setting real items those are missing in partition Pf only. These changed data has significant role during the development process of Naïve Bayes classifier. It is also responsible for detection of defective software modules. In the evaluation phase, the performance of NB model with ten bins is considered. It can be noticed that, this performance is not much satisfactory. It may either increase or decrease with respective to the inclusion of missing itemsets. The stability of the above technique is analyzed through the development of NB classifier having different numbers of bins. Recall has the responsibility to analyze various bins in order to conclude whether the performance is improved or not. Re-labeling of bins to missing values can enhance the prediction rate of D modules up to 40%. It also results false positive errors which is less than 14% observed in case of the AR4 dataset. The outcomes are generated by considering the data gathered from real projects. In future, this technique can be implemented in many software companies. Additionally, various binning technique can also be implemented to test the above approach and its efficiency. In the present work, equal numbers of bins are used for every individual variable.

Q. Song, et.al, proposed a general software defect-proneness prediction framework [7]. It involves an unbiased and comprehensive comparison technique. We can select various learning mechanisms for different types of datasets. It can be said that, no approach is allowed to dominate in all datasets. A minor modification during the evaluation phase may influence the resulted outcomes significantly. The above suggested technique is very much efficient for real world implementation irrespective of the nature of data. Here we can take one simple example, both balanced and imbalanced can influence the overall performance of the above learning technique. There are certain cases, where data is skewed.

Karim O. Elish, et.al, developed new scheme in order to predict all defects of software modules efficiently [16]. It is basically a prediction scheme that has the responsibility to identify defect-prone software modules. This approach uses the concept of support vector machines for the prediction purpose. The resulted prediction performance is compared along with other eight statistical and machine learning approaches. It also considers and includes four NASA datasets. The above proposed approach is compared with two numbers of statistical classification schemes (Logistic Regression (LR) and K-Nearest Neighbour (KNN) ). Again, it is compared with another two numbers of neural networks approaches (Multi-layer Perceptions (MLP) and Radial Basis Function (RBF)).
Filtering Data:

**Input:** High dimensional dataset $HD^1, HD^2, ..., HD^n$

**Output:** Filtered Datasets

**Procedure:**

- Load dataset $HD^1, HD^2, ..., HD^n$
- For each attribute $A(i)$ in the $HD^1, HD^2, ..., HD^n$
  - For each instance value $I(j)$ in the $A(i)$
    - if(isNum($I(j)$) && $I(j)==null$) then
      - $I(j)=\sum_{j\in A(i)}((I(j))-\mu_{A(i)}))/(\text{Max}_{A(i)}-\text{Min}_{A(i)})$     -----(1)
    - end if
- if(isNominal($A(i)$) && $A(i)(I)==null$) then
  - $I(j)=\text{PriorProb}(A(i),\text{class(m)})$;     ----(2)
- Here, mth class of the missing value is used to find the prior probability in place of missing value
  - end if

**Hybrid Feature Selection Measure**

In the proposed work, we have used different feature selection measures such as Hybrid FS, Information Gain FS, Gain Ratio FS to evaluate attribute selection measure in the proposed decision tree model.

The computation formula used to measure the feature selection is given in equation (3) as

$$
\text{CFSM}(D, D_{\text{attribute}}) = \sqrt{\sum_{i,j} \sum_{m} \left( \sum_{m} \left[ \frac{D_{ij}}{D_m} - \frac{D_{ij}}{D_m} \right] \right)^2 \sum \text{Corr}(D, D_{\text{attribute}})}
$$

---(3)

Where, $D_i$ is the $i$th class instances.

$D_m$ is the remaining $m$th class instances.

Proposed attribute selection measure is computed using (3),(4) and (5) as

$$
\text{HFSM}(A) = \frac{\sqrt{\text{GainRatio}(A) * \text{CFSM}(D, D_{\text{attribute}})}}{\text{IG}(A)}
$$

---(6)

**Proposed Feature Selection Based Ensemble Classifier**

**Input:** Ranked Features Data as $FData$;

**Output:** Disease prediction

**Procedure:**

- Read cancer disease dataset as $CDat$
- For each Feature $CData [i]$ in $CData$
  - For each instance $I(A_i)$ in $A_i$
    - Divide the data instances of $FA(D_i)$ into ‘k’ independent sets.
    - Select classifier $C_{\text{split}}$
    - Load training features and instances
      - a) Construct $N$ subset of trained data and $N$ subset of test data sampling with replacement.
      - b) In the tree growing phase, each and every node select $k$ features at random from $N$, compute for best split computation using equation (6)
      - c) Sort the $k$ individual trees according to class labels.
d). Select the majority voting available in each tree using ensemble learning.

End while

Calculate misclassified rate and statistical f-measure, accuracy and true positive rates;

Done

4. Experimental Results

In this section, we have executed our proposed model on NASA and Promise software defect datasets and compared the results with traditional defect prediction models.

NASA Metrics Data Program, it is publicly available for verifying, refuting and improving predictive models of software engineering. KC1 is a C++ system implementing storage management for receiving and processing ground data. The dataset consists of the McCabe and Halstead features extractors of the code. The measures are module based.

The probability of detection is proportional to the effort; thus, higher rate of detection, more effort is required. Probability of false alarm decreases with increase in detection. This linkage can be observed in receiver operating curve (ROC).

**Generated Patterns:**

- \( \text{lines}\_\text{removed} < 135.6 \text{ AND lines}\_\text{removed} < 678.0 \text{ -> lines}\_\text{added} <= 404.4 \)
- \( \text{lines}\_\text{removed} < 135.6 \text{ -> filetype} != \text{i18n} \text{ -> external} != 1 \)
- \( \text{lines}\_\text{added} <= 1011.0 \text{ AND lines}\_\text{removed} <= 135.6 \text{ -> external} != 1 \)
- \( \text{lines}\_\text{added} <= 404.4 \text{ -> filetype} != \text{images} \)
- \( \text{lines}\_\text{added} < 1011.0 \text{ AND external} != 1 \text{ AND lines}\_\text{removed} < 271.2 \)
- \( \text{lines}\_\text{removed} < 135.6 \text{ -> filetype} != \text{images} \)
- \( \text{lines}\_\text{added} < 1011.0 \text{ -> filetype} != \text{documentation} \text{ AND lines}\_\text{added} < 808.8 \)
- \( \text{lines}\_\text{removed} < 135.6 \text{ AND lines}\_\text{removed} <= 678.0 \text{ -> lines}\_\text{added} <= 404.4 \)
- \( \text{lines}\_\text{added} < 202.2 \text{ AND lines}\_\text{removed} <= 135.6 \text{ AND lines}\_\text{removed} < 271.2 \)
- \( \text{lines}\_\text{added} <= 404.4 \text{ -> filetype} != \text{images} \)
- \( \text{lines}\_\text{added} < 1011.0 \text{ -> filetype} != \text{documentation} \text{ AND lines}\_\text{added} < 606.59 \)
- \( \text{lines}\_\text{removed} <= 135.6 \text{ AND lines}\_\text{removed} <= 678.0 \)
- \( \text{lines}\_\text{added} <= 404.4 \)
- \( \text{lines}\_\text{added} <= 404.4 \text{ AND lines}\_\text{added} <= 606.59 \)
- \( \text{lines}\_\text{removed} <= 678.0 \text{ AND lines}\_\text{removed} <= 135.6 \text{ -> filetype} != \text{i18n} \)
- \( \text{lines}\_\text{removed} <= 678.0 \text{ AND lines}\_\text{removed} <= 135.6 \text{ AND external} != 1 \text{ AND lines}\_\text{added} <= 202.2 \)
- \( \text{filetype} != \text{documentation} \)
- \( \text{lines}\_\text{removed} < 135.6 \text{ -> external} != 1 \)
- \( \text{lines}\_\text{removed} < 271.2 \text{ -> filetype} != \text{images} \)
- \( \text{lines}\_\text{removed} < 542.4 \text{ AND lines}\_\text{removed} < 271.2 \)
- \( \text{lines}\_\text{added} <= 606.59 \)
- \( \text{lines}\_\text{removed} < 1011.0 \text{ -> lines}\_\text{removed} < 135.6 \)
- \( \text{lines}\_\text{removed} < 678.0 \text{ AND lines}\_\text{removed} <= 135.6 \text{ AND lines}\_\text{removed} < 271.2 \)
- \( \text{filetype} != \text{documentation} \text{ -> lines}\_\text{removed} <= 678.0 \)
- \( \text{filetype} != \text{images} \text{ AND filetype} != \text{unknown} \)
- \( \text{lines}\_\text{added} <= 202.2 \)
- \( \text{lines}\_\text{removed} < 542.4 \text{ AND lines}\_\text{removed} < 135.6 \text{ -> lines}\_\text{added} <= 606.59 \)
- \( \text{lines}\_\text{added} <= 404.4 \text{ -> lines}\_\text{removed} < 135.6 \)
- \( \text{external} != 1 \text{ -> lines}\_\text{added} <= 202.2 \)
- \( \text{external} != 1 \text{ -> lines}\_\text{added} <= 202.2 \)
- \( \text{lines}\_\text{added} < 606.59 \text{ AND lines}\_\text{added} < 808.8 \)
- \( \text{lines}\_\text{removed} <= 406.79 \)
- \( \text{lines}\_\text{removed} <= 678.0 \text{ AND external} != 1 \)
- \( \text{lines}\_\text{removed} <= 271.2 \text{ -> filetype} != \text{images} \)
- \( \text{lines}\_\text{added} < 1011.0 \text{ -> lines}\_\text{removed} < 406.79 \)
- \( \text{lines}\_\text{removed} <= 406.79 \text{ AND lines}\_\text{removed} <= 678.0 \)
- \( \text{lines}\_\text{added} <= 404.4 \)

**Generated Patterns:**

- \( \text{DIT} >= 0.0 \text{ -> Bug-count} != \text{false} \)
- \( \text{RFC} <= 862.0 \text{ AND NOC} <= 38.0 \text{ -> Bug-count} != \text{false} \)
- \( \text{DIT} <= 8.0 \text{ AND Bug-count} != \text{false} \text{ -> NPM} <= 214.0 \)
- \( \text{RFC} <= 862.0 \text{ -> DIT} <= 8.0 \)
- \( \text{DIT} >= 0.0 \text{ AND RFC} <= 862.0 \text{ -> Bug-count} != \text{false} \)
- \( \text{DIT} >= 0.0 \text{ AND CBO} <= 125.0 \text{ -> WMC} <= 351.0 \)
- \( \text{LOC} <= 5317.0 \text{ AND DIT} <= 8.0 \text{ AND NPM} <= 214.0 \)
- \( \text{RFC} >= 0.0 \)
- \( \text{LCOM} >= 0.0 \text{ -> DIT} <= 8.0 \)
- \( \text{CBO} <= 125.0 \text{ AND RFC} <= 862.0 \text{ -> Bug-count} != \text{false} \)
- \( \text{CBO} <= 125.0 \text{ AND RFC} <= 862.0 \text{ -> WMC} <= 351.0 \)
- \( \text{RFC} <= 862.0 \text{ -> DIT} <= 8.0 \)
- \( \text{RFC} <= 8.0 \text{ -> LCOM} <= 100.0 \)
- \( \text{RFC} <= 8.0 \text{ -> NOC} <= 38.0 \)
- \( \text{WMC} <= 351.0 \text{ -> NOC} <= 38.0 \)
- \( \text{NOC} <= 38.0 \text{ -> RFC} <= 0.0 \)
- \( \text{NOC} <= 38.0 \text{ AND Bug-count} != \text{false} \text{ -> NPM} <= 214.0 \)
- \( \text{NOC} <= 38.0 \text{ AND Bug-count} != \text{false} \text{ -> RFC} <= 862.0 \)
- \( \text{NOC} <= 38.0 \text{ -> LCOM} <= 100.0 \)
- \( \text{NOC} <= 38.0 \text{ -> DIT} <= 8.0 \)
RFC >= 0.0 AND RFC <= 862.0 → DIT <= 8.0
NPM <= 214.0 AND RFC <= 862.0 → WMC <= 351.0
NOC <= 38.0 → DIT <= 8.0
LOC <= 5317.0 → WMC <= 351.0
NOC <= 38.0 → Bug-count != false
Bug-count != false → LOC <= 5317.0
LOC < 5317.0 → DIT >= 0.0
LOC < 5317.0 → WMC < 351.0
NPM < 128.39 → Bug-count != false
RFC < 862.0 → NOC <= 38.0
LOC < 5317.0 → CBO <= 125.0
NOC <= 38.0 → Bug-count != false
NPM < 214.0 → NOC <= 38.0
NOC < 38.0 → Bug-count != false
NPM < 214.0 AND RFC <= 862.0 → Bug-count != false
WMC <= 351.0 AND LCOM >= 0.0 → NOC <= 38.0
LOC < 5317.0 → RFC <= 862.0 → Bug-count != false
LOC < 5317.0 → RFC <= 862.0 → DIT >= 0.0
LOC < 5317.0 → RFC <= 862.0 → WMC < 351.0
LOC < 5317.0 → RFC <= 862.0 → NOC <= 38.0
LOC < 5317.0 → RFC <= 862.0 → Bug-count != false

Table 1: Performance analysis of proposed model with the traditional models for classification Accuracy measure

<table>
<thead>
<tr>
<th>Datasize</th>
<th>KNN</th>
<th>Predictive Bug detection</th>
<th>Regression Based Bug prediction</th>
<th>Proposed Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>#500</td>
<td>0.8700</td>
<td>0.8512</td>
<td>0.7980</td>
<td>0.9732</td>
</tr>
<tr>
<td>#1000</td>
<td>0.9100</td>
<td>0.8675</td>
<td>0.8141</td>
<td>0.9625</td>
</tr>
<tr>
<td>#1500</td>
<td>0.8500</td>
<td>0.8272</td>
<td>0.8451</td>
<td>0.9264</td>
</tr>
<tr>
<td>#2000</td>
<td>0.8610</td>
<td>0.8195</td>
<td>0.8422</td>
<td>0.9834</td>
</tr>
<tr>
<td>#5000</td>
<td>0.8937</td>
<td>0.8402</td>
<td>0.8601</td>
<td>0.9783</td>
</tr>
</tbody>
</table>

The comparative between the classification accuracy of proposed model with the traditional models is shown in table 1. From the table it is observed that the accuracy rate improves as the noise level increases in the training data.

Performance analysis of proposed approach with traditional methods for classification Accuracy measure

<table>
<thead>
<tr>
<th>Multiple datasets</th>
<th>SVM</th>
<th>Naïve BN</th>
<th>Random Forest</th>
<th>Proposed Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>#500</td>
<td>0.8586</td>
<td>0.8374</td>
<td>0.8318</td>
<td>0.983</td>
</tr>
<tr>
<td>#1000</td>
<td>0.8782</td>
<td>0.8643</td>
<td>0.9098</td>
<td>0.973</td>
</tr>
<tr>
<td>#1500</td>
<td>0.8854</td>
<td>0.8788</td>
<td>0.9183</td>
<td>0.935</td>
</tr>
<tr>
<td>#2000</td>
<td>0.8485</td>
<td>0.9145</td>
<td>0.8879</td>
<td>0.939</td>
</tr>
<tr>
<td>#5000</td>
<td>0.8976</td>
<td>0.8659</td>
<td>0.9289</td>
<td>0.9898</td>
</tr>
</tbody>
</table>

Table 5.1, describes the different phases and its performance. From the table it is observed that the accuracy rate improves on an average of 15% when preprocessed with the proposed methodology.

5. CONCLUSION

In this paper, a new feature selection based ensemble model was proposed on cancer microarray disease datasets. Most of the conventional classification techniques deal with limited attributes and small datasets. Proposed model is one of the ensemble learning models, which is capable to handle datasets with large number of attributes.

Thus, the proposed feature selection based prediction model is used to assess the Cross-Company Defects Prediction (CCDP) on multiple defect databases. Proposed clustering based defect prediction model is used to group the new defects in the cross company defects for defect prediction. Experimental results proved that the proposed model has high computational accuracy compared to the existing software defect prediction models in terms of time and memory is concerned. Experimental results show that proposed model has high computational efficiency in terms of accuracy and true positive rate. In future, this work can be extended to large number of instances with high dimensional feature set and noise instances.

References


workshop on Software quality (pp. 35-40).
ACM.results, limitations, new approaches, 

(2009, August). Cross-project defect prediction: a 
large-scale experiment on data vs. domain vs. 
process. In Proceedings of the 7th joint meeting of 
the European software engineering conference and 
the ACM SIGSOFT Symposium on the foundations 
of software engineering (pp. 91-100). ACM.

(2011). A general software defect-proneness 
prediction framework. IEEE Transactions on 

August). REMI: Defect prediction for efficient API 
testing. In Proceedings of the 2015 10th Joint 
Meeting on Foundations of Software Engineering 
(pp. 990-993). ACM.

the evaluation of defect prediction models. In 
Proceedings of the 5th International Conference on 
Predictor Models in Software Engineering (p. 7). 
ACM.

Y., & A.Bener,(2010). Defect prediction from static 
code features current results, limitations, new 
approaches. Automated Software Engineering, 
17(4), 375-407.