International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Application of Two-Stage Data Pre-processing Approach for Software Fault Prediction

Sajna P[1], Shahad P[2]

[1, 2]MEA Engineering College, State Highway 39, Nellikkunn-Vengoor, Perinthalmanna, Malappuram

Abstract: ***Software fault prediction is a valuable exercise in software quality assurance to better allocate limited testing resources. Classification is one of the effective strategies for predicting software errors. The classification models are trained based on data sets obtained by historical repositories of mining software. In this project, a new Two-stage data preprocessing approach is applied with classification models such as Naive Bayes, Decision Tree, Knn Classifier and SVM to improve the prediction accuracy of each classification model. The data preprocessing approach in two stages incorporates both the selection of features and the reduction of instances. Specifically, in the feature selection stage, first relevance analysis is done, second, a threshold-based clustering method is proposed, termed novel threshold-based clustering algorithm, to drive redundancy control. In the instance reduction stage, random sampling is applied to maintain the balance between defective and non defective instances. To demonstrate this project chose real-world software project dataset, such as Eclipse.***

Keywords: Data Mining, Fault Detection, Prediction Algorithms, Data preprocessing, Classifiers

1. Introduction

Software fault prediction is a hot research topic in software engineering. It can allocate the limited test resources effectively by predicting the fault proneness of software modules. Classification is one of the prevalent methods used for software fault prediction.

Due to the increased dependency of modern system on software-based system, software reliability and quality has become the primary concern during the software development. It is very difficult to produce fault-free software due to the problem complexity, complexity of human behaviors, and the resource constrains. System failures due to the software failure are common and results in undesirable consequences, which will affect both reliability and safety of the system.

A software system consists of various modules and, in general, it is known that a small number of software modules are responsible for majority of the failures. Moreover, it is also known that early identification of faulty module can help in producing software of quality and reliability more cost effectively.

A software fault is an error, mistake, failure, or defect in a computer program or system that produces unexpected results, or causes it to behave in unintentionally. Software fault prediction is the process of locating defective modules in software. SFP helps us to improves software quality and test efficiency by constructing predictive models from code attributes to enable timely identification of error-prone modules, predicting software failures also helps us in planning, Monitoring and control and prediction of defect density. Control the quality oft he software. The result of Software

Defect Prediction, ie, the number of defects left in a software system, the results can be used as an important measure for the software developer and can be used to control the overall software process. Detecting software faults prior to system development may reduce software maintenance costs. Early software fault prediction helps to improve software quality and to achieve high software reliability.

Software fault prediction is a hot research topic in the domain of software engineering. It can allocate the limited test resources effectively by predicting the fault proneness of software modules. Classification is one of the prevalent methods used for software fault prediction. Its main task is to categorize modules, represented by a set of software code metrics, into two classes: fault-prone modules (FP), or non-fault prone modules (NFP). For a specific classification model, the each classifier must be trained in advance on the basis of training data historical repositories of mining software, such as records of changes in Software configuration management, error reporting on error tracking Systems, and emails from developers.

Data mining involves the general process of extracting knowledge from large amounts of data. The different types of data mining techniques are used, such as regression, classification, and associations. Emphasis is placed on the classification technique, which consists in classifying the data in a predefined class to its predictive characteristics. The result of an each classification technique is a model which will make it possible to classify future data points based on a set of specific characteristics in an automated way. In the literature, there are many classification techniques, some of the most commonly used being C4.5, k-nearest neighbor (KNN), Naïve Bayes (NB) and Support Vector Machines (SVM).

One of the most important aims of fault prediction is to detect fault-prone modules as early as possible in the software development life cycle. Design and code metrics have been successfully used for predicting fault-prone modules. In recent years, researchers have found that the quality of software datasets had a serious effect on the performance of predicting software faults. Issues concerned with data quality include biased datasets, noise a large number of features, and class imbalance. One is the high

dimensionality problem caused by too many unnecessary features, and the other is the class imbalance problem caused by superfluous instances of some classes. To solve these problems we design a novel two-stage data preprocessing approach which combines both feature selection and instance reduction.

We design experiments based on real-world software project Eclipse to demonstrate the effectiveness of our approach. Based on this thesis study, we found that the two-stage approach can greatly reduce both the number of features and the number of instances of original datasets, without sacrificing the prediction the performance of the classifier built after. Compared with other commonly used data preprocessing methods, our approach has presented consistently better performance over different classification models. Numerous researchers have successfully used classification models to categorize software modules into faulty and non-faulty, based on the features (i.e., metrics) collected from each module by mining software development repositories. These classification models require training data collected from previous projects where faulty modules have been identified. First, design the research questions for the empirical study. Second, describe the datasets used in the experiments. Third, put forward the experimental design based on the research questions. Last introduce the performance measure used in the research.

To evaluate the effectiveness of our proposed approach, design and perform a series of experiments using datasets collected from real-world software project, including the Eclipse projects, which is commonly used by researchers in software fault prediction. The Eclipse datasets are obtained from the PROMISE data repository.

2. Related Works

In this section, first we briefly introduce the background of software fault prediction. Second, we describe Two stage data preprocessing approach and finally we describe some classifiers for prediction purposes.

Defective modules in operational software may cause failures, increase costs of development and maintenance, and decrease customer satisfaction. Many prediction models have been proposed for identifying defect-prone software modules. Software metrics can be used for fault prediction in software projects. Performance of SFP is determined by either the choice of suitable classification algorithm or quality of datasets. It also depends on the selection of features from the large datasets.

In this paper, we present our two-stage data preprocessing approach and apply this approach to various classification models. The Two stage data preprocessing approach can combine both feature selection and instance reduction to eliminate irrelevant and redundant elements in the training data. By combing the feature selection and instance reduction, a balanced and effective dataset can be constructed to improve the quality of training data for classification models.

3. The Two-stage Data Preprocessing Approach

Two-stage data preprocessing approach for software fault prediction, which combines feature selection and instance reduction to eliminate both irrelevant and redundant units in software datasets. In particular, for the feature selection, we propose an improved version of our previous method[1].

First, feature ranking is used to eliminate irrelevant features. Second, a novel threshold-based clustering algorithm is used to remove redundant features. For the instance reduction, because the similarity among NFP instances (i.e., software modules) is usually high random sampling is applied to reduce the NFP instances. By the above two-phase preprocessing, we could get a balanced, high quality dataset for training the classification models, which would improve the performance of fault prediction [2].

In the framework, Fig.2.1, the feature selection (FS) stage performs feature ranking and threshold-based clustering in sequence, while the instance reduction (IR) stage uses random sampling to reduce the NFP instances. Here, FS is performed before IR (denoted as FS, IR) in our approach.

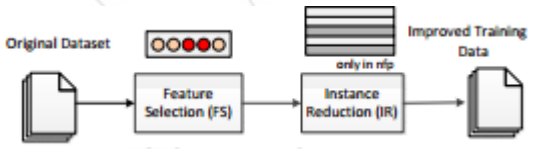

Figure 2.1: Framework of TSPP [2]

For feature selection, we need to keep as much information as possible to select the valuable features, and after eliminating instances the ranking of the features may be changed. On the other hand, as random sampling is used for instance reduction in our approach, the effect of feature selection on it can be ignored. The first step is performing relevance analysis to remove irrelevant features, and the second step is performing redundancy control to eliminate redundant features. Feature ranking is used for the first step, where individual features are ranked according to the importance (i.e., weights) in differentiating instances of different classes (i.e., FP or NFP). The subset of features from the top- of the ranking list is selected. To weight the relevance of a feature to the class, we should measure the correlation between them. The commonly used measures can be categorized into three groups: entropy based (such as information gain, gain ratio, and symmetric uncertainty), statistical based (such as chi-square), and instance based (such as Relief and ReliefF). In our method, we choose one representative from each group, each of which has been proven good in software fault prediction [2][3].

The novel threshold-based clustering algorithm (NTC)[39] is used for the second step. Algorithm shows the details of NTC. At first, features are grouped into clusters using a pre-specified threshold . To cluster these features, the algorithm starts by computing the similarity between each pair of features, and constructs a-nearest neighbor graph over the features. The nodes of the graph correspond to the features, and a link between two features and exists if the similarity is

no less than . After that, the feature which has the most compact neighborhood (i.e., the average similarity of its neighbor set is the highest) is selected, while all the remaining neighbors are removed. Finally, the graph is reconstructed without the selected feature and its neighbors. This procedure is repeated until all the features are either selected or removed. The methods for computing the similarity between any pair of features. To measure the similarity between any pair of features, we choose the non-linear similarity measure Symmetric Uncertainty (SU), and the commonly used linear similarity measure Cosine Similarity (COS). Both the SU and the COS measures range from 0 to 1. The greater the measure, the more likely the two features are similar. The algorithm is rather effective, and can be optimized. The most time-consuming part is the computation of the similarities between pairs of the features.

In the instance reduction stage, we apply random sampling to reduce the NFP instances. Random sampling is a simple but effective technique for instance reduction . It may bear the risk of losing information by removing valuable instances, because all the instances are treated similarly. However, during our experiments, we found that the similarities among the NFP instances are usually higher than 0.9 on average, no matter which function is used to compute them. Because the datasets used in our experiments are commonly used by other researchers, we think it is reasonable to reduce the NFP set by random sampling without replacement, which leads to the simplest implementation. To keep balance between the FP and NFP classes, we set the FP/NFP ratio as the terminal condition of sampling. In the experiments, the FP/NFP ratio is set to 35pers and 65pers according to the recommendation by Khoshgoftaar et al [5].

Algorithm 1 The Algorithm TC Used for Feature Selection

Input:
The original given dataset $T(f_1, f_2, \cdots, f_m, l)$ with m features.
Here $l \in \{FP, NFP\}$ denotes the class label
The number of features k selected by feature ranking
The similarity threshold β for clustering
Output:
The selected feature subset F_{out}
/*==== Part 1: Irrelevant Features Removal =====*/
1: **for** $i = 1$ to m **do**
2: Computing the relevance r_i between f_i and the class using SU measure
3: **end for**
4: Sorting the feature set based on r_i and returning the top k features in F_{tk}
5: $F_{out} = F_{tk}$
/* ==== Part 2: Redundant Features Elimination ===*/
6: **for** each pair of features (f_i, f_j) in F_{out} **do**
7: Computing $SU(f_i, f_j)$
8: **end for**
9: **for** each feature $f_i \in F_{out}$ **do**
10: **for** each $f_j \in F_{out} \setminus f_i$ **do**
11: **if** $SU(f_i, f_j) >= \beta$ **then**
12: Adding f_j into C_i
13: **end if**
14: **end for**
15: **end for**
16: **while** C is not empty **do**
17: **for** each $C_i \in C$ **do**
18: $AS_i = \frac{\sum_{f_j \in C_i} SU(f_i, f_j)}{|C_i|}$
19: **end for**
20: $C_{i'} = \arg\max_{C_i \in C} AS_i$
21: Removing all the features in $C_{i'}$ from F_{out}
22: $C = C \setminus C_{i'}$
23: Updating all the rest clusters in C, removing both $f_{i'}$ and features in $C_{i'}$
24: **end while**
25: **return** F_{out}

Figure 2.2: NTC algorithm [2]

Instance Reduction Methods: For the instance reduction stage, we use random under-sampling (RUS), which randomly discards instances from the NFP class to build a more balanced dataset. To make a comparison, we also implement random over-sampling (ROS), which randomly duplicates instances in the FP class until the dataset reaches a balanced ratio [4].

For instance reduction, the random under-sampling technique is proved useful, and can be applied to other feature selection methods with profitable results, while the over-sampling technique is not suitable, at least for software datasets. Random sampling is effective in the experiments, because we have found that similarities among instances in the datasets are usually high (e.g., the average instance similarity measured by SU is greater than 0.9 in almost all the datasets)

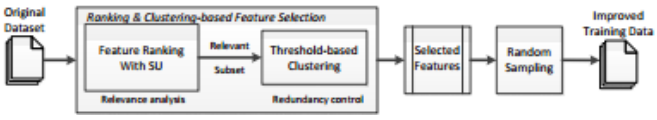

Figure 2.3: Details of TSPP [2]

4. Software Fault Prediction Models

A.Naive Bayes Software Defect Prediction Model

Na¨ıve Bayes(NB) is a very effective machine learning method. A NB model treats defect prediction as binary classification, it trains and constructs predictor by analyzing historical data of software modules, based on predictor it will make decision whether new module has defects or not [9]. In [10] proposed Naive Bayes Prediction (NBP) model chooses software module to be object unit of training and prediction. Let A= a1,a2,...,an be set of metrics attribute set, there is a vector M: (a1,w1),(a2,w2),...,(an,wn) to denote a software module, where 'a' is metrics and 'w' is weight of 'ai' . We train the classifier using module data sets with category tag, and compute the defective probability of a new module which will make an alert when it exceeds a threshold.

If we define the category notion of software module is C∈(Cd,Cn) where Cd is defective category and Cn is non-defective category. Then according to Bayesian theory, the probability that software module is defective will be computed by,

$$P[C_d \mid M] = \frac{P[M \mid C_d] \times P[C_d]}{\sum_{c \in C} P[C] \times P[M \mid C]} \qquad (2)$$

Classifier will classify software module M to Cd when the ratio greater than threshold. Estimation of P[M |c] is a key problem of NBP model .To solve this problem we use Multi-variants Gauss Naïve Bayes [10]. When attribute value of metrics is real-valued, Multi-variants Gauss Naïve Bayes (MvGNB) estimates $P[M|c]$ using ,

$$P[M \mid c] = \prod_{i=1}^{n} g(w_i; \mu_{i,c}, \sigma_{i,c}) \qquad (3)$$

Where we supposed that each attribute follows a normal distribution 'g' in each category 'c', and the mean (μ) and typical deviation (σ) of each distribution are estimated from the training data sets. In[11]derived the Weighted NaïveBayes method, and then describe three heuristics for feature weight assignment. Used three heuristics in order to

estimate the weights of features based on their relative importance. Two novel heuristics are introduced for this purpose.

We have evaluated our approach on Weighted Naïve Bayes predictor, which is an extension of standard Naïve Bayes. To the best of our knowledge, the weighted features approach is a novel one in defect prediction literature. We observed linear methods for feature weighting lack the ability to improve the performance of Naïve Bayes [11].

B. Support Vector Machine model

SVM are useful tools for performing data classification. and have been successfully used in applications. SVM constructs an N-dimensional hyperplane that optimally separates the data set into two categories. The purpose of SVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the dependent variable on one side of the plane and the cases with the other category on the other side of the plane [12]. The support vectors are the vectors near the hyperplane. The SVM modeling finds the hyperpalne that is oriented so that the margin between the support vectors is maximized. When the given points are separated by a nonlinear region, SVM classifier handles this by using a kernel function in order to map the data into a different space when a hyperplane can be used to do the separation. Decision stump achieves slightly higher classification accuracy, the precision and f measure is much lower. Elish et al. [12] and [13] stated that the performance of Support Vector Machines (SVM) is generally better than, or at least is competitive against the other statistical and machine learning models in the context of four NASA datasets. In [14] Proposed ensemble SVM. The bagging algorithm creates an ensemble of models for a learning scheme where each model gives an equally weighted prediction. In this study, the Support Vector Machine is constructed and 10-fold cross validation technique is applied and evaluated error rate from the mean square error. Secondly, bagging is performed with Support Vector Machine to obtain a very good generalization performance.

C. Decision tree model

The Decision tree is one of the classification techniques which is done by the splitting criteria. The decision tree is a flow chart like a tree structure that classifies instances by sorting them based on the feature (attribute) value[15][16]. Each node in a decision tree represents a feature in an instance to be classified. All branches denote an outcome of the test, each leaf node hold the class label. The instances are classified from starting based on their feature value. Decision tree generates the rule for the classification of the data set. Decision trees use feature values for the classification of instances. A feature in an instance that has to be classified is represented by each node of the decision tree, while the assumption values taken by each node is represented by each branch. The classification of instances is performed by following a path through the tree from root to leaf nodes by checking feature values against rules at each node. The root node is the node that best divides the training data.

Three basic algorithms are widely used that are ID3, C4.5, and CART [18].ID3 is an iterative Dichotomer 3. It is an older decision tree algorithm introduced by Quinlan Ross in 1986. The basic concept is to make a decision tree by using the top-down greedy approach. C4.5 is the decision tree algorithm generated by Quinlan. It is an extension of ID3 algorithm.. C4.5 algorithm is widely used because of its quick classification and high precision. CART stands for Classification Regression Tree introduced by Bremen. The property of CART is that it is able to generate the regression tree.

The C4.5 can be referred as the statistic Classifier. This algorithm uses gain radio for feature selection and to construct the decision tree[17]. It handles both continuous and discrete features. C4.5 algorithm is widely used because of its quick classification and high precision. C4.5 based technique that uses information entropy to build the decision tree. At each node of the decision tree, a rule is chosen by C4.5 such that it divides the set of training samples into subsets effectively. The C4.5 algorithm is an inductive supervised learning system which employs. decision trees to represent a quality model. C4.5 is a descendent of another induction program[17]

D. KNN model

The K-Nearest Neighbor (NN) is the simplest method of machine learning. It is a type of instance base learning in which object is classified based on the closest training example in the feature space[19][1]. It implicitly computes the decision boundary however it is also possible to compute the decision explicitly. So the computational complexity of K NN is the function of the boundary complexity .The k-NN algorithm is sensitive to the local structure of the data set. The special case when k = 1 is called the nearest neighbor algorithm. The best choice of k depends upon the data set; larger values of k reduce the effect of noise on the classification but make boundaries between classes less distinct. The various heuristic techniques are used to select the optimal value of K. KNN has some strong consistent results. As the infinity approaches to data, the algorithm is guaranteed to yield an error rate less than the Bayes error rate.

Nearest neighbour (a.k.a., lazy-learning) techniques are another category of statistical techniques. Nearest neighbor learners take more time in the testing phase, while taking less time than techniques like decision trees, neural networks, and Bayesian networks during the training phase .In this paper, we study the KNN nearest neighbor technique. KNN considers the K most similar training examples to classify an instance. KNN computes the Euclidean distance to measure the distance between instances . We find K = 8 to be the best-performing K value of the five tested options (i.e., 2, 4, 6, 8, and 16).

5. Experimental Design

In this section, we design experiments to demonstrate the effectiveness of our approach. First, we design the research

questions for the empirical study. Second, we describe the datasets used in the experiments. Third, we put forward the experimental design based on the research questions. Last we introduce the performance measure used in our research.

A. Research Questions

1) To validate whether the novel threshold-based clustering algorithm NTC of the feature selection stage can improve the performance of the classification models built after.
2) What are the effects of the characteristics of individual datasets, including level of instance sufficiency, and temporal issues, on the final performance of the approach.
3) Is our two-stage approach better to improve the performance of the classification models built after.

B. Dataset

To evaluate the effectiveness of our proposed approach, design and perform a series of experiments using datasets collected from real-world software project, including the Eclipse projects, which is commonly used by researchers in software fault prediction. The Eclipse datasets are obtained from the PROMISE data repository.

For the Eclipse datasets, three releases of Eclipse are collected with instances measured at the Java class level. Each of the Eclipse datasets contains 198 features, including the code complexity measures (such as LOC, cycloramic complexity, and number of classes), the syntax tree based measures, and many others.

Before the experiments, we perform the following treatments to the datasets. 1) Remove all the non-numeric measures. 2) Transform the post-release faults measure (which counts the number of faults in the post release versions) into the binary class label. In particular, those containing one or more faults are labeled as FP, whereas those with zero faults are labeled as NFP. 3) Remove the measures which have only one distinct value. After performing these treatments, each Eclipse dataset leaves 155 features.

C. Proposed system SFP framework

The data preprocessing software ensure to improve the quality of data mining as it is designed by considering scalability characteristics. With the evolution of distributed computing, the databases were inherently distributed across the globe. The core need in the current industrial environment is to extract information from the huge, complex and dynamic data through data mining techniques. However, existing data mining tools are not effective in efficiently processing the dynamic and inconsistent data.

To overcome the existing system drawbacks we apply two stage data preprocessing method in training dataset to improve the quality of dataset. The quality of software datasets had serious effect on the performance of predicting software faults. Issues concerned in data quality include biased datasets , noise , a large number of features, and class imbalance. Fig 3.1 shows the proposed system work and Fig 3.2 shows architecture diagram of proposed system SFP model respectively

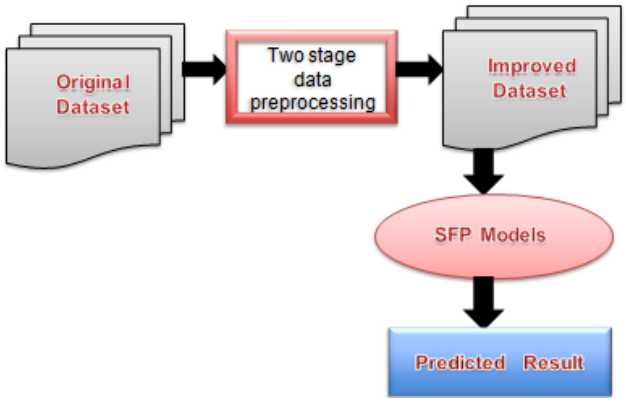

Figure 3.1: Proposed system SFP framework

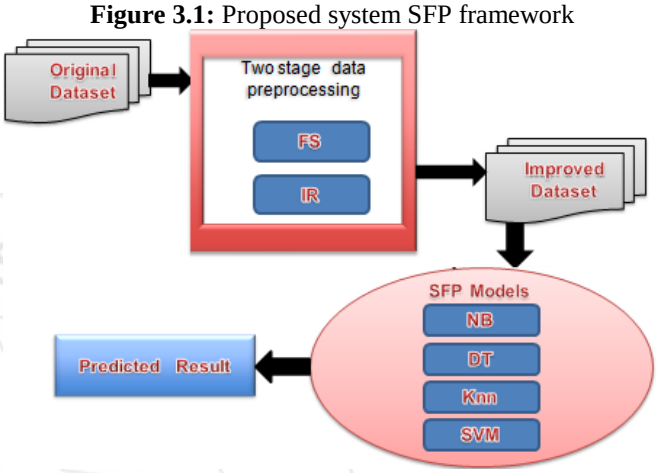

Figure 3.2: Architecture diagram of Proposed system.

To study the research questions, 6 different preprocessing schemes for the feature selection stage are designed according to the combinations of the relevance measures and similarity measures. Combined with options for the instance reduction stage, these schemes are applied with four different classification models to comprehensively study the effects on the performance of software fault prediction. We combine TSPP with four different classification models, which are commonly used in software fault prediction. The models are Naive Bayes (NB), C4.5 decision tree (C4.5), k-Nearest Neighbors ($k = 1$, denoted as IB1) and Support vector machine(SVM).

All the experiment results are averaged over 10 ×10-fold cross validation, which means 10-fold cross validation repeated 10 times in each experiment. A 10-fold cross validation means that the dataset is equally di- vided into 10 parts, and instances of each part are used in turn as the testing set, while the remaining instances are used as the training set [2], [11], [22], [49], [61]. The performance measure is averaged over the 10 folds. To further overcome the effect of randomness, the 10-fold cross validation is repeated 10 times, and the grand average value over the 10 repetitions is used as the final performance measure.

D. Prediction Performance Measures

In order to evaluate performance of these models, we compared prediction results. We choose the AUC measure to evaluate the prediction performance. The results demonstrate the potential of our approach in enhancing the prediction performance of the classifiers built thereafter.

i. Confusion Matrix

A confusion matrix is a visualization tool that reports the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). TN represents the fault free modules correctly classified. FP refers to fault free modules incorrectly labelled as faulty modules. FN corresponds to faulty modules incorrectly labelled as fault free modules. TP refers to modules that are correctly classified as faulty modules [5].

ii. ROC curve

AUC-ROC is used as a performance metrics [5][12][19]. A receiver operating characteristic (ROC) curve can be represented equivalently by plotting the probability of detection (PD) vs. probability of false alarms (PF).ROC curves can be beneficial for finding accuracy of predictions.ROC curve is divided in two different regions defined as follows. Risk incompatible region with high PD and high PF, is beneficial for safety critical systems as identification of faults is more important than validating false alarms. Cost incompatible region defines low PD and low PF, this region is beneficial for the organizations having limited Verification Validation budgets. Negative region with low PD high PF is also preferred for some of the software projects.ROC analysis can easily avoid this risk.AUC is area under the ROC curve. Higher AUC values indicate that the classifier is on average more to the upper left region of the graph.

iii. Accuracy

Accuracy is also known as correct classification rate. It can be find out as the ratio of the number of modules correctly predicted to the total number of modules. The accuracy is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

iv. Precision

Precision is the proportion of the examples which truly have class x / Total classified as class x. Precision gives positive predictive values and it process values or product quality or exactness. So basically high precision stated the accurate results and it takes all relevant data but returns only topmost results.

Precision = TP/ (True Positive + False Positive)

v. Recall

Recall gives sensitivity of problem and it process values or product quantity or completeness. It returned most relevant and part of the documents that are relevant as result from the query. In other words, modules that are really recognize as difficult to maintain from the total number of modules.

Recall = TP/ (True Positive + False Negative)

v. F-Measure

It is a combined measure for precision and recall.
F-Measure = 2*Precision*Recall / (Precision + Recall).

E. Results

An increasing number of studies use machine learning approaches to predict where faults are likely to occur in code. The performance of models is measured in a range of ways that makes cross comparison very complex. The external validation of models is rarely demonstrated. We found that combination of two stage data preprocessing approach and classification models are greatly improve the prediction accuracy of software fault prediction. Table 1: shows the comparison result of fault prediction models without Two stage datapreprocessing approach and Table 2: shows that comparison result of software fault prediction models with Two stage data preprocessing approach.

Table 1: Comparison Results of Algorithms without TSPP

SFP Model	Precision	Recall	F-Measure	AUC	Accuracy
NB Model	0.83	0.838	0.836	0.652	83.77
DT Model	0.741	0.861	0.797	0.617	86.1
KNN Model	0.789	0.812	0.812	0.621	84.14
SVM Model	0.814	0.87	0.899	0.695	87.05

Table 2: Comparison Results of Algorithms with TSPP

SFP Model	Precision	Recall	F-Measure	AUC	Accuracy
NB Model	0.85	0.898	0.891	0.737	85.54
DT Model	0.87	0.83	0.825	0.728	89.99
KNN Model	0.84	0.861	0.876	0.711	88.87
SVM Model	0.90	0.939	0.943	0.782	91.02

6. Conclusion

Defect prediction is based on good data mining model. In this work different data mining algorithms used for defect prediction. We provide a two-stage data preprocessing approach, which incorporates both feature selection and instance reduction, to improve the quality of software datasets used by classification models for software fault prediction.

We systematically design experiments based on the Eclipse dataset. The results demonstrate the potential of our approach in enhancing the prediction performance of the classifiers built thereafter to comparing previous methods. Experiments reveal that SVM achieves best performance Our most important finding is that there is no single data mining technique that is more powerful or suitable for all type of projects. Multiple classifiers were combined by majority voting of experts to get more accurate result.

References

[1] J. Chen, S. Liu, X. Chen, Q. Gu, and D. Chen, "Empirical studies on feature selection for software fault prediction," in Proc. Asia-Pacific Symp. Internetware, 2013, pp. 163–166.
[2] J. Chen, S. Liu, W. Liu, X. Chen, Q. Gu, and D. Chen, "A two-stage data preprocessing approach for software fault prediction," in Proc. Int. Conf. Software Security and Reliability, 2014, pp. 20–29.
[3] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in Proc. Int. Conf. Mach. Learn., 2003, vol. 3, pp. 856–863.

[4] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in Working Notes of the ICML Workshop on Learning From Imbalanced Data Sets, 2003, vol. 2.
[5] T. M. Khoshgoftaar, K. Gao, and A. Napolitano, "An empirical study of feature ranking techniques for software quality prediction," Int. J. Softw. Eng. Knowl. Eng., vol. 22, no. 2, pp. 161–183, 2012.
[6] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
[7] Zhong, Shi, Taghi M. Khoshgoftaar, and Naeem Seliya. "Unsupervised Learning for Expert-Based Software Quality Estimation." *HASE*. 2004.
[8] Zhong, Shi, Taghi M. Khoshgoftaar, and Naeem Seliya. "Analyzing software measurement data with clustering techniques." *IEEE Intelligent Systems* 19.2 (2004): 20-27.
[9] Jiang, Yue, Bojan Cukic, and Tim Menzies. "Cost curve evaluation of fault prediction models." *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*. IEEE, 2008.
[10] Kaur, Deepinder, et al. "A clustering algorithm for software fault prediction." *Computer and Communication Technology (ICCCT), 2010 International Conference on*. IEEE, 2010.
[11] Bishnu, Partha S., and Vandana Bhattacherjee. "Software fault prediction using quad tree-based k-means clustering algorithm." *IEEE Transactions on knowledge and data engineering* 24.6 (2012): 1146-1150.
[12] Patil, Swapna M., and R. V. Argiddi. "Study Of Fault Prediction Using Quad Tree Based K-Means Algorithm And Quad Tree Based EM Algorithm."
[13] Varade, Swati, and Madhav Ingle. "Hyper-quad-tree based k-means clustering algorithm for fault prediction." *International Journal of Computer Applications* 76.5 (2013).
[14] Lessmann, Stefan, et al. "Benchmarking classification models for software defect prediction: A proposed framework and novel findings." *IEEE Transactions on Software Engineering* 34.4 (2008): 485-496.
[15] Wang Tao, LI Wei-hua "Naïve Bayes Software Defect Prediction Model"School of ComputerScience and Technology,Northwestern Polytechnical University, Xi'an, China.*water@snnu.edu.cn*
[16] Turhan, Burak, and Ayse Basar Bener. "Software Defect Prediction: Heuristics for Weighted Naïve Bayes." *ICSOFT (SE)*. 2007.
[17] Elish, Karim O., and Mahmoud O. Elish. "Predicting defect-prone software modules using support vector machines." *Journal of Systems and Software* 81.5 (2008): 649-660.
[18] Singh, Yogesh, Arvinder Kaur, and Ruchika Malhotra. "Software fault proneness prediction using support vector machines." *Proceedings of the world congress on engineering*. Vol. 1. 2009.
[19] Shanthini, A., G. Vinodhini, and R. M. Chandrasekaran. "Bagged SVM Classifier for Software Fault Prediction." *International Journal of Computer Applications* 62.15 (2013).
[20] Chen, Mike, et al. "Failure diagnosis using decision trees." *Autonomic Computing, 2004. Proceedings. International Conference on*. IEEE, 2004.
[21] Dietterich, Thomas G. "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization." *Machine learning* 40.2 (2000): 139-157.
[22] Sharma, Seema, Jitendra Agrawal, and Sanjeev Sharma. "Classification through machine learning technique: C4. 5 algorithm based on various entropies." *International Journal of Computer Applications* 82.16 (2013).
[23] Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1.1 (1986): 81-106.
[24] Wilson, D. Randall, and Tony R. Martinez. "Reduction techniques for instance-based learning algorithms." *Machine learning* 38.3 (2000): 257-286.
[25] Bradley, Andrew P. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern recognition* 30.7 (1997): 1145-1159.

Author Profile

Sajna P. received the B.Tech. degrees in Computer Science and Engineering from MEA Engineering college Perinthalmanna, Malappuram During 2009-2013 . Right now she is pursuing her M. Tech degree in computer science at MEA Engineering College, Kerala from 2015 to 2017. Her research interests lie in Data Mining.

Shahad P, Assistant Professor, MEA Engineering College, Perinthalmanna. Received the B.Tech degree in Computer Science & Engineering from Maharaja Prithvi Engineening College Coimbatore. and M.tech degree in Computer Science from College of Enginnering, at MEA Engineering College Perinthalmanna. His research interests includes Data Mining.