# An Ensemble Framework for Software Defect Prediction

[1]Shiwang Agarwal

**ABSTRACT:** Defect Prediction is a very necessary tool in the maintenance and durability of software. It increases durability, reduces cost and most important efforts. It increases the efficiency of the software. In this paper, a verifiable study is done to analyse the performance and durability of software. Three open source software projects are taken in this paper. Cross-validation techniques i.eKfold, Stratified K fold Repeated stratified K fold are applied for training data. Smote, Random Under-Sampling are applied for balancing imbalance data. Ensembling of KNN and Decision Tree classifiers to form an ensemble model, which is analyzed through different performance measures. Later comparing the performance of the ensemble model with the performance of each classifier used in ensemble modeling. The finding infers that Ensemble model performed better than individual classifier.

**Keywords --** machine learning, k-fold cross validation, data balancing, Voting Classifier

## 1 INTRODUCTION

**Defect** prediction is an essential tool in software development. It ensures software durability, increases software efficiency, decreases maintenance cost, increases reliability by analyzing bugs in software.

Our paper revolves around one question i.e Does the proposed ensemble model performs better than the performance of individual classifiers.
In this research data preprocessing is done on dataset , on which k-fold cross-validation techniques like Stratified Kfold, Kfold techniques are applied for training the data , this training data and testing data is balanced through SMOTE, RUC, after balancing the training and testing data we move to ensemble model, where different models i.e KNN and Decision Tree are ensembled to form an ensemble model, explanation of classifiers are shown in Table 1. The paper is divided into different sections: Section 2 gives an overview of the literature survey. Section 3 explains the proposed methodology. Section 4 gives an overview of results and discussion. Section 5 provides a conclusion.

- [1] graduated in computer science and engineering in Dr.A.P.J Abdul Kalam Technical University,India,shiwang999@gmail.com

TABLE 1 CLASSIFIERS USED IN DEFECT PREDICTION

| CLASSIFIER | DESCRIPTION |
|---|---|
| KNN | K-Nearest Neighbors is used for finding k nearest neighbors to the data point to be categorized. |
| Decision Tree | It constructs tree-like a model and selects the best possible solutions through the help of the Gini index and entropy index. |

## 2 LITERATURE SURVEY

Ron Kohavi. [1] worked on cross-validation techniques to improve the performance of the software. Sriparna Saha. [2] worked on multiple classifiers by ensembling them through voting classifier to improve the accuracy of the software. Thomas Mandl.[3] worked named entity for effectiveness in cross-language information. Quan Zheng[4] worked on cross defect prediction using unsupervised learning to analysis accuracy. Yutao Ma. [5] worked on software defect prediction by balancing data using different techniques. Diri, B.[6] worked on proper review of software fault prediction. Sevim, U.[7] deals with Software Fault Prediction of Unlabeled data. P. Chandra.[8] deals with techniques of machine learning to deal with software related issues. Mrinal Rawat.[9] deals with ml techniques for quality improvement of the software. Gong, L[10] deals with noise occurring in software using machine learning techniques A. [12] deals with different models for dealing with effort estimation and software defect prediction. Monden, A[13] deals with ensemble techniques for software defect prediction through machine learning. Fong, S. [14] deals with balancing data through Oversampling and undersampling techniques. Mishra, A.[15] deals with experience in analyzing and predicting fault-prone software modules using performance measures. Zimmermann.[16] Deals with cross-project software defect prediction through machine learning.

## 3 PROPOSED METHODOLOGY

It deals with collecting data in the form of datasets, preprocessing the data, training the data through cross-validation , balancing the data, ensembling different models to form an ensemble model. It contains step by step process shown in Fig 1.

Fig 1. PROPOSED METHODOLOGY

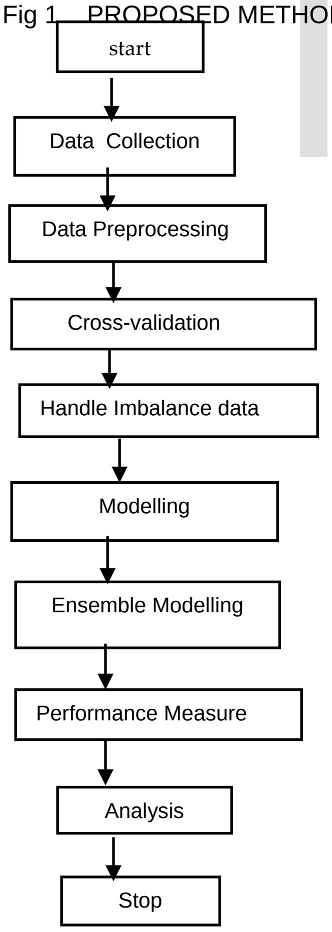


TABLE 2. DESCRIPTION OF DATASET

| S.NO | NAME OF DATASET | TOTAL LABELS | TOTAL NUMBER OF BUGS |
|---|---|---|---|
| 1 | Xerses-1.2 | 441 | 72 |
| 2 | Xerses-1.3 | 454 | 70 |
| 3 | Xerses-1.4 | 589 | 432 |

### 3.1 Data Collection

Data Collection is a method of collecting data through different sources in the form of datasets in .csv format and use them for prediction. In this paper we have taken 3 datasets named xerses 1.2, xerses 1.3, xerses 1.4 from www.openscience.us/repo as shown in Table 2.

### 3.2 Data preprocessing

Data preprocessing deals with the processing of data. In this, we can convert any string feature into integer through Label Encoder and Labels into 0's and 1's through OneHot Encoder. Data preprocessing is used to reduce noise before using it in an algorithm. In the dataset we are provided with the different feature set, to make easy to understand for the machine we use data preprocessing where we are converting a string, float into integer format. Here we use numpy library, pandas library to deal with data.

### 3.3 Cross-validation

In cross-validation, we deal with preprocessed data coming from the dataset. In cross-validation data is divided into training and testing data. In it we divide data into n-folds i.e (n-1) for training and rest for testing purpose, this process is done until all features of the dataset are used for training and testing purpose. It is different from normal data split method as it trains and tests on each and every feature of the dataset. There are different methods used for K-fold cross-validation those are K Fold, Stratified K fold.

- K Fold: It is used for dividing training and testing data. It does not deal with uniform distribution, it is depicted in below Table 1 for n-splits=4. It is represented as:

  K-fold(n_splits=4, random-state=12) (1)

  as shown in Table 3 where it is explained through an example. Here n_splits states that splitting is done into 4 segments with 3 segments for training and rest for testing.
- Stratified K Fold: It is used for dividing training and testing data. It deals with uniform distribution , which is depicted in below Table 1 for n-splits=4. It is represented as:

  Stratified KFold(n_splits=4,random-state=12) (2)

  as shown in Table 4 where it is explained through an example .Here n_splits states that splitting is done into 4 segments with 3 segments for training and rest for testing.

TABLE 3. K-FOLD

| Feature | Total | Sample1 | Sample2 | Sample3 |
|---|---|---|---|---|
| Boys | 5 | 2 | 1 | 3 |
| Girls | 6 | 2 | 3 | 1 |

TABLE 4. STRATIFIED K-FOLD

| Feature | Total | Sample1 | Sample2 | Sample3 |
|---|---|---|---|---|
| Boys | 5 | 3 | 3 | 3 |
| Girls | 6 | 1 | 1 | 1 |

## 3.4 Handle Imbalance data

Handling Imbalance data is a very necessary requirement in defect prediction , we encounter with a large amount of data without being balanced , training and testing data were taken are not in proper proportion leading false prediction , inaccurate prediction. To deal with this issue we encounter with various techniques in handling imbalance dataset.

SMOTE: SMOTE stands for Synthetic Minority over-sampling technique. It avoids overfitting issue where an exact replica of minority instance(for testing purpose) are added with majority instances to form complete balance data from preprocessed data set. Procedure to deal with SMOTE are:

- Identify the feature vector and nearest neighbor.
- Take the difference between the two.
- Multiply difference with random no between 0 and 1.
- Identify new point on line segment by adding random no to feature vector.
- Repeat process for identified feature vector.

## 3.5 Modeling

It deals with classifying the data point to which class it belongs. It deals with preprocessed data which is later split into training and testing data through 7:3 ratio i.e 70% data for training and 30% for testing.There are many techniques to deal with it, those are KNN ,Decision tree, etc.

- KNN*:* K nearest neighbor is one of the finest technique to classify data point to which category it belongs. It is a classifier which classifies data point through nearest neighbor principle. All data points are plotted in the x-y plane and new data point is to categorize to which class it belongs, for it which we take k=3 i.e 3 nearest neighbors to an unknown data point will be selected and then majority data point nearest to unknown data point will classify unknown data point. To calculate it we encounter with Manhattan distance and Euclidean distance.

  Euclidean distance: ( d(p,q)^2)=((q1-p1)^2)+((q2-p2)^2) (3)
- Decision Tree: Decision Tree is one of the finest technique to classify data point to which category it belongs. It forms tree-like structure when the feature is selected on the basis of their Gini index and entropy index values , feature with low value will be taken for the root node and then to the subsequent node.

  Gini index=1-((probability of"yes")^2)-((probability of "no")^2 (4)

## 3.6 Ensemble Modelling

In modeling, we have seen how classifier classifies but to increase the performance of the software we deal with ensemble modeling technique , where we take 2 or more classifiers and perform the task.The technique which we used in this paper is Voting Classifier with voting="hard" . Voting Classifier deals with majority voting i.e it will consider responses from all classifiers and select the result according to majority count . Syntax to represent it is through: VotingClassifier(estimators=[m1,m2],voting='hard')

Here estimators take various models and voting=hard means majority vote will be considered as output. Ensemble modeling helps to increase the performance of software, taking best individual classifiers i.e KNN and Decision Tree and merging them to form a new classifier which is Voting Classifier. Voting Classifier contains parameter named voting ="hard" and voting ="soft", where voting="soft" deals with probability and provide result after dealing with probability , and voting ="hard" deals with max voting, i.e provide result by analyzing result of all classifiers and give result having majority votes.

## 3.7 Performance Measure

After dealing with data preprocessing, K fold cross-validation, balancing the data and classifying data, we encounter with a performance measure. There are various performance measure those are accuracy,precision, recall,f1 score,AUC curve. These performance measures are calculated through confusion matrix and classification report. Explanation of each is explained in Table 5.

TN: True Negative means Observation done is positive and prediction done is negative.

TP: True Positive means Observation done is positive and prediction done is positive

FP: False Positive means Observation done is False and prediction done is Positive.

FN: False negative means Observation done is False and prediction done is negative.

TABLE 5. PERFORMANCE MEASURES

| PERFORMANCE MEASURE | DESCRIPTION |
|---|---|
| Recall | True Positive / (True Positive+False Negative) |
| F1_score | (2*Precision*Recall)/ (Precision +Recall) |
| Precision | True Positive/(True Positive+False Positive) |

## 4 RESULTS & DISCUSSION

Results of two classifiers named KNN and Decision Tree have been taken. Results have shown good performance at n-splits =20 in Stratified K Fold.In this paper we have taken 3 datasets , Label Encoder and One Hot Encoder are used to preprocess the data , to make machine easily understand the data, after that we have used Stratified K Fold with n-splits=20 to split data into training and testing , using Stratified K Fold we have better result than K Fold as it provides symmetrical distribution of data. After training and testing data through cross-validation next step was to balance features used for training and testing set, to do this there is need of algorithm to deal with it, to do this SMOTE(Synthetic Minority over-sampling technique) performed better result than other techniques like ROC, etc .

After dealing with balanced training and testing features, the next step was modeling where KNN and Decision Tree classifiers were used for classification. Ensemble technique is used to ensure it provides better performance than simple individual model i.e KNN and decision tree and results show that it performed better than individual classifiers.The accuracy of Voting Classifier(ensemble technique) was 90.48 as shown in Table 6 , the accuracy of KNN was 82.34 as shown in Table 7 and accuracy of Decision Tree was 87.89 as shown in Table 8.

AUC value of Voting Classifier was 0.944, AUC value of KNN is 0.8 and for Decision Tree was 0.77, looking after all these data in table 5 ,Voting Classifier i.e ensemble technique performed better than other classifiers i.e KNN and Decision Tree , Table 9 depicts it completely.

TABLE 6. CLASSIFICATION REPORT ON VOTING CLASSIFIER

| Training | Testing | Cross-validation | no of splits | Imbalance data | ensemble technique | Accuracy | Precision | Recall | F1-score | AUC | Support |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | Voting Classifier | 90.48 | 0.94 | 0.94 | 0.94 | 0.944 | 21 |
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | Voting Classifier | 80 | 0.84 | 0.84 | 0.84 | 0.844 | 22 |
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | Voting Classifier | 89.29 | 0.89 | 0.89 | 0.89 | 0.833 | 28 |

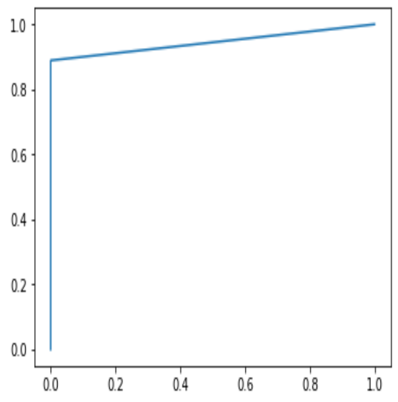
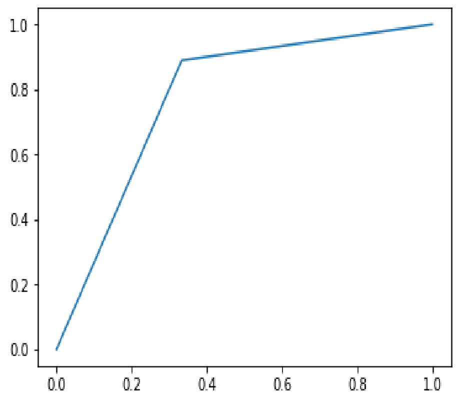
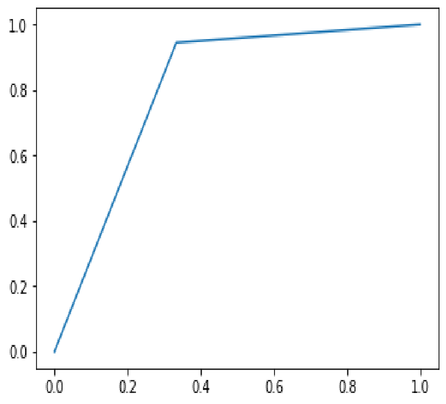
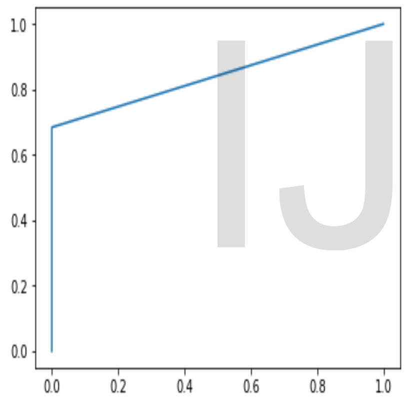
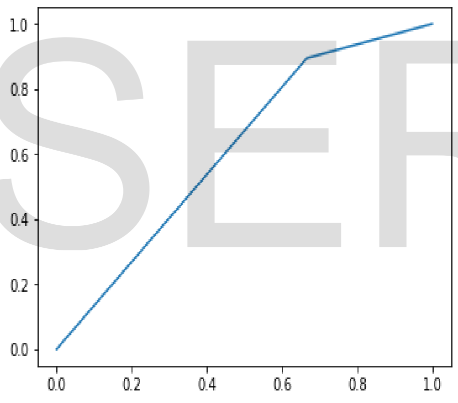
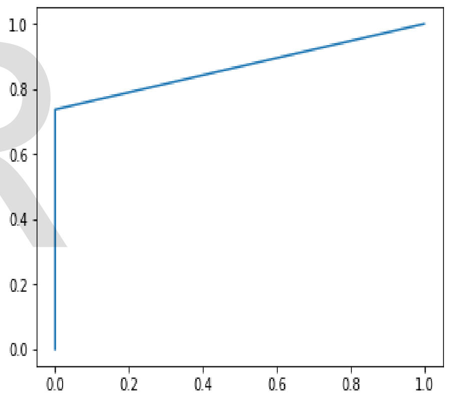
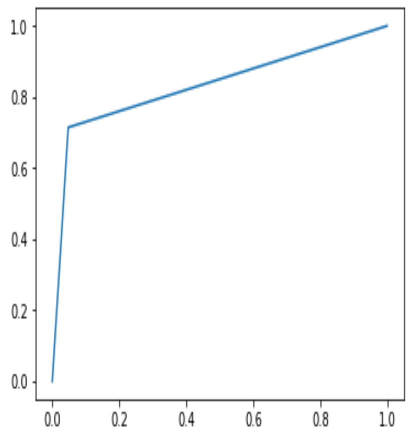
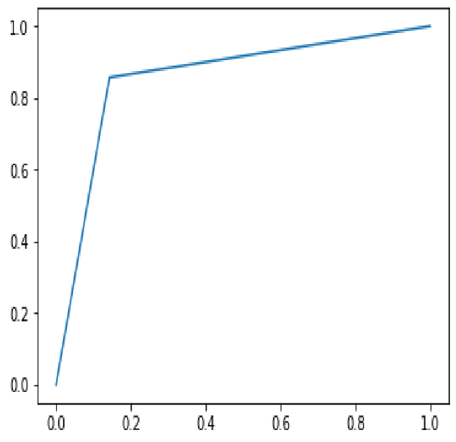
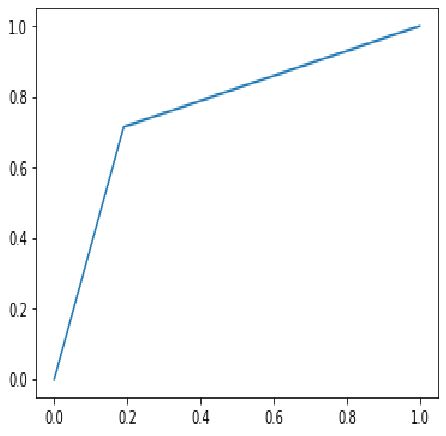
TABLE 7. CLASSIFICATION REPORT ON KNN

| Training | Testing | Cross-validation | no of splits | Imbalance data | Classifier | Accuracy | Precision | Recall | F1-score | AUC | support |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | KNN | 82.34 | 0.9 | 0.9 | 0.9 | 0.8 | 21 |
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | KNN | 86.3 | 0.77 | 0.77 | 0.77 | 0.86 | 22 |
| Xerses 1.2 | Xerses 1.2 | Stratified Kfold | 20 | SMOTE | KNN | 89.54 | 0.79 | 0.79 | 0.79 | 0.761 | 28 |

TABLE 8. CLASSIFICATION REPORT ON DECISION

| Training | Testing | Cross-validation | no of splits | Imbalance data | ensemble technique | Accuracy | Precision | Recall | F1-score | AUC | support |
|---|---|---|---|---|---|---|---|---|---|---|---|
| xerses 1.2 | xerses 1.2 | Stratified Kfold | 20 | Smote | Decision Tree | 87.89 | 0.86 | 0.86 | 0.86 | 0.777 | 21 |
| xerses 1.3 | xerses 1.3 | Stratified Kfold | 20 | Smote | Decision Tree | 85.63 | 0.82 | 0.82 | 0.82 | 0.614 | 22 |
| xerses 1.4 | xerses 1.4 | Stratified Kfold | 20 | Smote | Decision Tree | 92.67 | 0.86 | 0.86 | 0.86 | 0.857 | 28 |

TABLE 9. ROC CURVE

| Dataset | Voting Classifier | Decision Tree | KNN |
|---|---|---|---|
| Xerses1.2 |  |  |  |
| Xerses1.3 |  |  |  |
| Xerses1.4 |  |  |  |

## 5 CONCLUSION

Software defect prediction is necessary for good performance of the software. In this paper we have taken 3 datasets named xerses1.2, xerses 1.3 ,xerses 1.4 from www.openscience.us/repo ,data is preprocessed through label encoder, after that data is split into training and testing sets through cross-validation technique named Stratified K Fold, balancing of data is done through SMOTE technique which further is used for classification through individual classifiers i.e KNN and Decision tree and performance is compared with Voting classifier(ensemble modeling technique).

The results indicate that Voting classifier performed better than individual classifiers at n-splits=20 during cross-validation.

## 6 REFERENCES

[1] Ron Kohavi, “The study of cross-validation for accuracy estimation” (1995)

[2] Sriparna Saha , Asif ekbal,"Combining multiple classifiers using voting classifier”(2013).
[3] Thomas Mandl . “ The effect of named entities on effectiveness in cross-language info”(2005)
[4] Quan Zheng, Ahmed E. Hassan, “cross defect prediction using unsupervised classifier”(2016)

[5] Yutao Ma, Bing Li, “Cross project defect prediction using imbalance features” (2014)

[6] Diri, B.,Catal, C, “A proper review of software fault prediction”(2009)

[7] Diri, B., Sevim ,U., “ Software Fault Prediction of Unlabeled Program Modules”(2009)

[8] P. Chandra, D. Sharma , " Software Defect Prediction Through Techniques of ML" 2018.

[9] Mrinal Rawat, “Machine learning techniques for software defect prediction for quality improvement “(2012)

[10] Gong, L. ,Wu, R., “working on noise in software defect prediction”,(2011)

[11] R. Robbes, M. D’Ambros,“analyzing software defect prediction approaches”.(2012).

[12] Butcher, A., Menzies, T., Marcus, A. “Different models for dealing with effort estimation and software defect prediction”.(2011)

[13] Monden, A. ,Uchigaki, S., “Ensembling Approach for Cross-Project Fault Prediction” (2012).

[14] Fong, S., Yap, B., Rani, K.,”Application for Oversampling, Undersampling, for Handling Imbalanced Datasets”.(2014).

[15] Mishra, A., Yu, L., “Experience in analyzing and predicting fault-prone software modules using performance measures”.(2012)

[16] Zimmermann, T. Murphy, B. , Nagappan, N., Gall, H. ,” Dealing with cross-project software defect prediction”(2009)