An Hybrid Approach for Cost Effective Prediction of Software Defects

Satya Srinivas Maddipati[1]
Research Scholar, CSE Department
Koneru Lakshmaiah Education Foundations
Guntur, India

Malladi Srinivas[2]
Professor, CSE Department
Koneru Lakshmaiah Education Foundations
Guntur, India

***Abstract*—Identifying software defects during early stages of Software Development life cycle reduces the project effort and cost. Hence there is a lot of research done in finding defective proneness of a software module using machine learning approaches. The main problems with software defect data are cost effective and imbalance. Cost effective problem refers to predicting defective module as non defective induces high penalty compared to predicting non defective module as defective. In our work, we are proposing a hybrid approach to address cost effective problem in Software defect data. To address cost effective problem, we used bagging technique with Artificial Neuro Fuzzy Inference system as base classifier. In addition to that, we also addressed Class Imbalance & High dimensionality problems using Artificial Neuro Fuzzy inference system & principle component analysis respectively. We conducted experiments on software defect datasets, downloaded from NASA dataset repository using our proposed approach and compared with approaches mentioned in literature survey. We observed Area under ROC curve (AuC) for proposed approach was improved approximately 15% compared with highly efficient approach mentioned in literature survey.**

Keywords*—Cost effective problem; principle component analysis; adaptive neuro fuzzy inference system; area under ROC curve

I. Introduction

Software Development process involves Requirement specification, Design, Implementation and Testing. During each phase of software development, reviews will be conducted to assess the progress and quality of software. The quality of software depends on defects found in the software. Defect is a condition that doesn't meet user requirement, specified in requirement specification. If a defect is found during late stages of software development i.e. during software maintenance, the penalty is very high. To reduce this penalty, the defective proneness must be identified in advance [27].

According to Boehm, the cost of fixing errors increased gradually as the software development progress. If we consider cost of fixing error during requirement phase as 1 unit, then the cost of fixing error in design phase will be 3-8 units, implementation phase will be 7 to 16 units, integration & testing phase will be 21 to 78 units and maintenance phase will be 29 to more than 1500 units. This motivates application of machine learning techniques in early stage identification of software defects [28]. Fig. 1 shows soft escalation of defect resolving during various phases of software development life cycle.

A. Machine Learning Techniques

Various Machine learning techniques such as K nearest neighbours, Support Vector machines, Decision Trees, Bayesian Networks and etc. are used to identify software defects.

B. Approaches for Software Defect Prediction (SDP)

1) Decision trees: Decision Trees are used as early classifier techniques for software defects. In a decision tree, the attribute with less impurity value is selected as root node. There are three measures for impurity 1) Entropy 2) Gini Index 3) Misclassification error. Decision tree will output whether the module is defective prone or not, based on input attributes like IO Comments, Cyclometric complexities etc. Fig. 2 shows Decision Tree constructed on cm1 dataset.

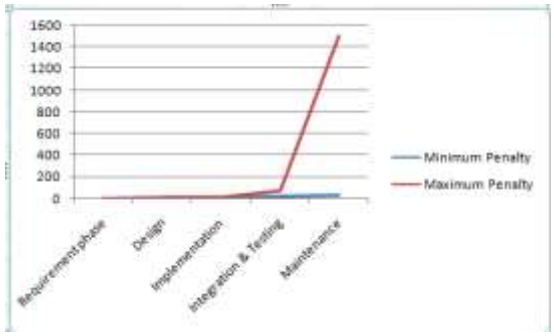

Fig. 1. Cost Escalation for Defect Solving during Phase of Software Development.

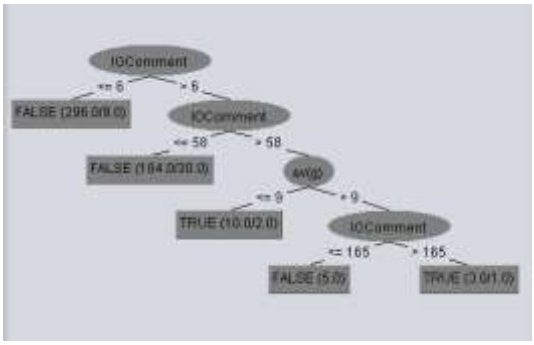

Fig. 2. Decision Tree on CM1 Dataset.

2) Bayesian Classifiers: Bayesian classifier uses Baye's theorem to classify unknown sample. It comes under lazy classifier. According to Bayes theorem, Conditional probability $P(Y=y_i/X=x_i)$ is defined as:

$$P(Y = yi/X = xi) = \frac{P(X = xi/Y = yi) * P(Y = yi)}{P(X = xi)}$$

There are two types of Bayesian classifiers: 1) Naive Baye's classifier 2) Bayesian Belief Networks.

Naive Baye's classifier: In Naive Baye's classifier, the given unknown sample is considered as 'X'. The classifier finds the posterior probability P (Defect=Yes/X) and P(Defect=No/X) for given sample 'X'.

If P(Defect=Yes/X) > P(Defect=No/X), then classifier outputs the sample 'X' as defective. Otherwise it outputs the given sample 'X' as non defective. The drawback with Naive Baye's classifier is, it assumes the target variable (Defect) is independent on input variables.

Bayesian Belief Networks: In Bayesian Belief Networks, There are two components: 1) Direct Acyclic Graph (DAG); 2) Probability table DAG encodes the relationship between attributes into a graph. Probability table comprises of posterior probabilities dependent on their parent attributes. Fig. 3 represents the DAG, constructed on cm1 dataset.

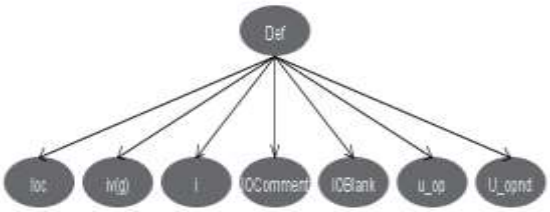

Fig. 3. Directed Acyclic Graph for Software Defect Prediction.

3) Support vector machines: Support vector machines are one of the popular classifier technique for regression and classification problems. Binary Support Vector Machines solves classification tasks while support vector regression solves regression tasks. Software defect prediction is a classification problem and hence Binary Support Vector Machines are used to classify the module as defective or non-defective. In support vector machines, there exist a boundary function that classify sample. There are various types of boundary functions like Linear, Polynomial, Radial basis, ANOVA and etc.

Polynomial Support Vector Machine:

Defect = -0.0062 * (normalized) loc+ 0.0043 * (normalized) iv(g) +0.0044 * (normalized) i+0.012 * (normalized) IOComment + -0.0021 * (normalized) IOBlank + 0.0004 * (normalized) u_op +-0.0054 * (normalized) U_opnd - 1.0005.

Multi Layer Perceptrons: Multi layer perceptrons are the neural networks which comprises Processing units, called neurons, organized in multiple layers. These neurons are having computing capabilities on inputs, receiving from previous layers, and propagate output to the next layers. These neurons are connected by weighted edges. Each neurons applies activation functions on the inputs along with threshold and produces output signals. There are various activation functions like Threshold, sigmoid, Tangible and etc.

Table I illustrates the architecture of multi layer perceptrons along with nodes, connections and their weights.

Artificial Neuro Fuzzy Inference System:

Artificial Neuro Fuzzy Inference System: ANFIS is a five layered architecture used for classification tasks. Satya srinivas et al. [26] proposed Artificial Neuro Fuzzy Inference System for Software defect prediction.

ANFIS generates Sugeno Fuzzy Inference system as output for classification task. In ANFIS input attributes are fuzzified and target attribute is defuzzified. Intially subtractive clustering method is used to generate Sugeno fuzzy inference system. The premise and consequent parameters in Sugeno Fuzzy inference system are trained used training data. Here training rate parameter must be set to appropriate value. Setting high training rate parameter converges the ANFIS model into unstable state. Setting low training rate parameter creates high complexity model.

4) Cost effective learning: Misclassifying some class samples results high penalty compared to misclassification of other classes. For example, in software Defect prediction, misclassifying defective module as non defective imposes high penalty compared to misclassifying non defective module as defective. If a defect was found during later stages of software development, it imposes high penalty and hence pronable defective module should not be misclassified as non defective even though non defective module was misclassified as defective. This error cost escalation was shown in Fig. 1.

5) Ensemble learning: Ensemble learning is the process of constructing multiple classifiers and combining them to improve the accuracy for classification problems. Some of the ensembling techniques are Simple voting, Average voting, Bagging, Boosting and etc. In simple voting, each classifier will vote for an output value. The output, value with high number of votes, considered as actual output. In average voting, the average value of output of each classifier is considered as actual output. This technique is suitable for regression tasks. In Bagging, the dataset is sampled into equal size subsets of data and a classifier is constructed with each subset. Finally each classifier will vote for output value. Bagging and Boosting techniques improves the performance of classifiers by constructing multiple classifiers.

In Bagging, classifiers are constructed in sequence. The samples which are incorrectly classified are given with higher weight for construction of next classifier. This procedure is repeated until required accuracy obtained or maximum numbers of classifiers were constructed.

TABLE I. MULTI LAYER PERCEPTRONS NODES, CONNECTIONS AND WEIGHTS

Node	Inputs	Weights	Node	Inputs	Weights	Node	Inputs	Weights
0	Threshold	-3.79506	2	Attrib u_op	4.04982	4	Attrib u_op	3.31857
0	Node 2	4.53581	2	Attrib U_opnd	4.93275	4	Attrib U_opnd	4.17034
0	Node 3	3.66528	3	Threshold	-4.9707	5	Threshold	-9.06268
0	Node 4	5.20758	3	Attrib loc	2.20562	5	Attrib loc	-3.81299
0	Node 5	3.18853	3	Attrib iv(g)	-7.07055	5	Attrib iv(g)	-1.35707
1	Threshold	3.79522	3	Attrib i	-6.27414	5	Attrib i	-0.84483
1	Node 2	-4.50871	3	Attrib IOComment	-2.75564	5	Attrib IOComment	-4.81926
1	Node 3	-3.66711	3	Attrib IOBlank	-1.19511	5	Attrib IOBlank	3.59625
1	Node 4	-5.23767	3	Attrib u_op	0.39515	5	Attrib u_op	-5.56998
1	Node 5	-3.18676	3	Attrib U_opnd	3.76011	5	Attrib U_opnd	-1.61555
2	Threshold	-2.83258	4	Threshold	-2.79212		Class FALSE	
2	Attrib loc	4.66117	4	Attrib loc	4.39579		Input	
2	Attrib iv(g)	0.10108	4	Attrib iv(g)	2.53554		Node 0	
2	Attrib i	-1.00049	4	Attrib i	-0.01669		Class TRUE	
2	Attrib IOComment	-8.27215	4	Attrib IOComment	-9.35516		Input	
2	Attrib IOBlank	0.286789	4	Attrib IOBlank	0.769326		Node 1	

In this paper, we are applying hybrid approach to overcome cost effective problem in SDP. Section II presents literature survey on SDP. In Section III, we designed methodology using hybrid approach for SDP. Section IV Presents the results by applying proposed methodology on SDP.

II. LITERATURE SURVEY

Yan Naung Soe et al. proposed Random Forest algorithm on Software Defect Prediction and compared the performance of Random forest algorithm with other machine learning techniques. They concluded that maximum accuracy is 99.59 and minimum accuracy is 85.96[1]. Taek Lee et al. proposed micro interaction metrics, such as browsing events, file editing, for prediction of software defects and observed high accuracy by combining these metrics with existing metrics in cost effective manner [2]. Fei Wu et al. proposed a cost-sensitive local collaborative representation (CLCR) approach for software defect prediction and concluded that accuracy has been increased with proposed approach [3]. Jinsheng Ren et.al proposed asymmetric kernel principle component analysis for solving class imbalance problem in software defect prediction. They evaluated the validity of their proposed model using *F*-measure, Friedman's test, and Tukey's test [4].

Ayse Tosun et al. proposed decision threshold optimization on Naive Bayes classifier to find best threshold that separate defective and non-defective samples in software defect data [5]. Ming Cheng et al. proposed semi supervised approach for identification of software defects. Their proposed model evaluates the confidence probability of unlabelled sample to predict class labels. They considered different misclassification cost to improve classifier performance [6]. Igor Ibarguren et al proposed consolidated tree construction that ensembles weights of misclassification in training of classifier. They showed that consolidated tree construction performs better than other rule based classifiers [7]. Yuanxun Shao et.al proposed weighted associative classification for addressing imbalance problem in software defect prediction. They determined weights of features using correlation analysis. They proved GMean measure has been increased with their approach [8]. Shuo Feng et al. proposed complexity based over sampling technique to address data imbalance problem in identification of software defects [9]. Rakesh Rana et al. proposed Bayesian Inference method for software defect prediction to analyse inflow distribution of defects. This technique has been used for early detection of software defects in large software projects [10].

Guisheng Fan et.al proposed attention based recurrent neural networks for software defect prediction. Their experimental results shows that the proposed model increases F1 score by 14% and AUC by 7% [11]. Sushant Kumar et al. proposed Deep representation and ensemble learning for Software defect prediction. They conducted experiments on 12 NASA Dataset repositories. Among 12 datasets, F Measure has been increased for 8 datasets, ROC values has been increased for 6 datasets, PRC values has been increased for 12 datasets and MCC values has been increased for 11 datasets[12]. Rodrigo et al. proposed ensemble of clustering using Particle Swarm Optimization for prediction of Software defects and concluded that prediction quality has been increased [13]. Shamsul Huda et al. proposed ensemble over sampling algorithm for prediction of software defects [14]. Shanthini. et al. proposed Ensemble SVM approach for prediction of software defects[15]. Nageswara Rao et.al proposed Ensemble Bayesian networks for prediction of Software Defects and proved that their proposed model have high true positive rate compared to traditional methods [16]. Steven Young et al. proposed deep super learner for Just in time defect prediction. They used bagging of random forests

and concluded that F1 score was improved for 5 of 6 projects [17]. Arvinder Kaur et al investigated different ensemble techniques such as Boosting, Bagging and Rotation forest in prediction of software defects. They conducted Wilcoxon signed rank test to prove ensemble techniques outperforms traditional techniques in generalization of results [18]. Thanh Tung et al proposed ensemble model by combining sampling technique with common classification technique to improve the performance of classifier [19].

Jaroslaw Hryszko et al investigated the effect of Software defect in modules on Quality assurance of Software. Their investigation proved that quality assurance cost can be reduced by 30% with their proposed approach [20]. Kazuya Tanaka et al focused on usage of auto-sklearn tool that automatically selects appropriate prediction model for data pre processing and classification in software defect prediction. This tool presents random forest is the best model in various machine learning techniques [21]. Pradeep Singh proposed stacking based framework, in which he combined class balancing technique SMOTE with ensemble classifiers to predict software defects. He concluded that the accuracy of stacking based model increased compared to traditional approaches used in their literature survey [22]. Haitao He et al proposed Ensemble RIPPER classifier for software defect prediction. In their research, they applied Principle component analysis for dimensionality reduction, Adaptive Synthetic sampling for balancing the dataset and RIPPER model for classification. They concluded that classification error has been reduced with their proposed model [23]. Zhiqiang Li et al proposed ensemble multiple kernel correlation alignment for heterogeneous defect prediction and they concluded ensemble approach outperforms remaining competing methods [24]. Xin Xia et al proposed Hybrid model reconstruction (HYDRA) approach for Software defect prediction. It consists of two phase's Genetic algorithm followed by Ensemble learning. They concluded that HYDRA improves F1 score of Zero-R base classifier [25].

In prediction of software defects, some researchers addressed class imbalance problem and someone addressed high dimensionality problem. But In this research work, we are addressing cost effective problem in SDP.

III. METHODOLOGY

In this paper, we are proposing Ensemble approach of Adaptive Neuro Fuzzy Inference system for prediction of Software defects for cost effective learning. In step 1, we are performing Synthetic Minority oversampling technique (SMOTE) to balance the dataset. In step 2, Dimensionality reduction will be performed to reduce the dataset. Here, we are proposing Principle component analysis (PCA) for dimensionality reduction. In step 3, multiple ANFIS classifiers will be constructed for ensemble approach. In step 4, Aggregation will be performed on votes given by multiple ANFIS classifiers and it produces the actual output.

In our research work, we considered data from NASA dataset repository. The dataset is neither noisy nor in complete but imbalanced. To remove imbalance, we are applying SMOTE technique and to overcome for high dimensionality problem we are applying PCA. Fig. 4 represents the proposed methodology for SDP.

Algorithm:

Step 1: Apply Synthetic Minority Over Sampling Technique for Class Balance.

- 1.1 Choose a random sample from minority class.
- 1.2 Identify k-nearest neighbours from chosen sample
- 1.3 For each neighbour sample
  - 1.3.1 construct a line from chosen sample to nearest neighbour
  - 1.3.2 Add more number samples by picking of points on the line
- 1.4 Repeat steps 1.1 to 1.3 until two classes samples are equal.

Step 2: Apply Dimensionality reduction using PCA

- 2.1 Perform Z score normalization on data.
  Z-score = $(x_i - \mu)/\sigma$
- 2.2 Create a covariance matrix for eigen decomposition.
- 2.3 select principle components with high relevance.

Step 3: Construct classifier using Artificial Neuro Fuzzy Inference system

- 3.1 Fuzzify input variable
- 3.2 Apply membership function on input variable
- 3.3 Calculate weighted average
- 3.4 calculate contribution of each fuzzy rule
- 3.5 Output sum of all incoming signals.

Step 4: Repeat steps 1 to Step 3 for multiple times (Possibly odd number of times).

Step 5: find number of votes for each class from multiple classifiers

Step 6: Output the class variable based on number of votes(High number of votes)

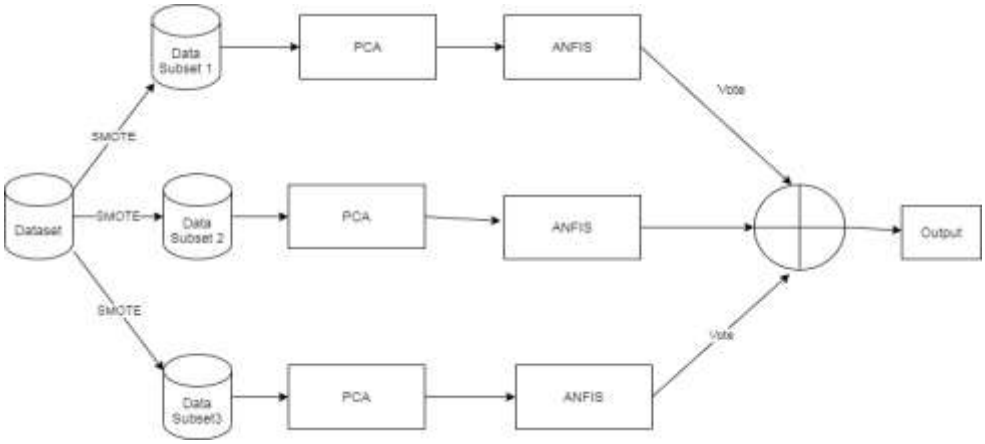

Fig. 4. Proposed Methodology for Software Defect Prediction.

IV. Experimentation and Results

In this work, we addressed class imbalance problem using Synthetic Minority Oversampling Technique. After balancing the data, we applied principle component analysis for dimensionality reduction. In Table II, we are projecting few principle component values constructed on cm1 dataset, downloaded from NASA dataset repository.

The reduced dataset is used to construct classifier using Adaptive Neuro Fuzzy Inference System. In this work, we applied AdaBoost Ensemble learning technique with ANFIS as base classifier. The performance of a classifier, constructed from imbalance data, can be measured using AuC (Area under ROC Curves). Receiver Operating Characteristics curves are constructed by plotting True positive rate against False positive rate. Area under this ROC curve is considered as a performance metric in our research work.

We applied cost sensitive approach to our classifier. In cost sensitive approach, the cost values are derived from imbalance nature of data. We found cost sensitive approach improves the performance of classifier. We applied our proposed model on various software defect datasets cm1, pc1, kc1 and jm1. In the Table III, We are comparing the AuC values of our proposed model with results of methods proposed in literature survey. Fig. 5 to 16 compares the ROC curves of various techniques discussed in literature survey with proposed methodology.

TABLE II. Principle Component Values on Software Defect Prediction

Attribute	PC1	PC2	PC3	PC4	PC5
loc	0.246323	0.031239	-0.08786	0.161044	-0.04483
V.g	0.243657	0.057173	0.103265	0.084628	-0.0588
ev.g.	0.205853	0.053904	0.118336	0.050775	-0.11766
iv.g.	0.233961	0.073498	0.029074	0.145675	0.010602
n	0.250283	-0.00128	-0.06655	-0.00328	0.006294
v	0.250585	0.043622	-0.04154	0.067938	0.037045
l	-0.09998	0.657384	-0.0066	0.191268	0.294755
d	0.211336	-0.14493	0.234525	-0.34617	-0.30322
i	0.193738	-0.11365	-0.60919	0.121661	0.10463
e	0.219659	0.167325	0.392074	0.132239	-0.0035
b	0.247381	0.132791	-0.07955	-0.00411	-0.01886
t	0.219659	0.167332	0.392072	0.132234	-0.0035
IOCode	0.199326	0.034844	0.132263	-0.30874	0.501051
IOComment	0.209861	0.048167	-0.12006	0.257712	-0.14999
IOBlank	0.188142	-0.05663	-0.02223	-0.46712	0.52168
locCC	-0.01344	0.623327	-0.27367	-0.513	-0.40025
u_op	0.218739	-0.21843	0.018721	-0.27123	-0.27324
U_opnd	0.238652	-0.03448	-0.31957	0.088945	0.053283
total_op	0.249928	-0.00192	-0.06993	0.008577	-0.01499
total_opnd	0.248372	0.000253	-0.06084	-0.0222	0.039162
branchCount	0.243334	0.040645	0.048155	0.08437	-0.06212

TABLE III. AUC VALUES OF VARIOUS MODELS WITH AND WITHOUT COST SENSITIVE TECHNIQUES

Dataset	CM1		PC1		KC1		JM1	
Base Classifier	**without Cost Sentive**	**With Cost sensitive**	**without Cost Sentive**	**With Cost sensitive**	**without Cost Sentive**	**With Cost sensitive**	**without Cost Sentive**	**With Cost sensitive**
J-48	0.53	0.56	0.49	0.53	0.48	0.54	0.49	0.55
Random Forest	0.53	0.74	0.51	0.71	0.5	0.72	0.51	0.69
SVM	0.49	0.55	0.48	0.53	0.5	0.53	0.51	0.57
K-NN	0.51	0.54	0.5	0.55	0.49	0.56	0.51	0.55
MLP	0.5	0.55	0.49	0.56	0.47	0.58	0.49	0.56
ANFIS	0.69	0.74	0.68	0.73	0.69	0.75	0.71	0.75

ROC Curves

J-48 (Decision Tree)

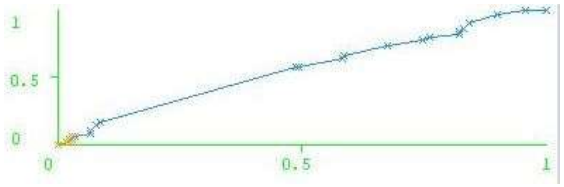

Fig. 5. ROC Curve using J-48 without Cost Sensitive.

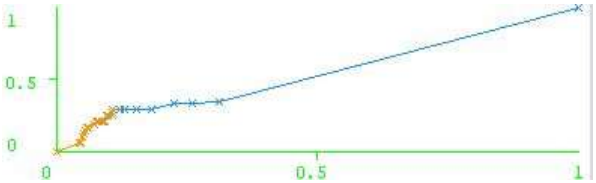

Fig. 6. ROC Curve using J-48 with Cost Sensitive.

Random Forest (RF)

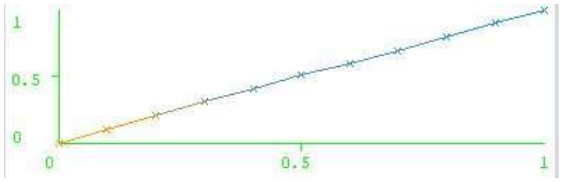

Fig. 7. ROC Curve using RF without Cost Sensitive.

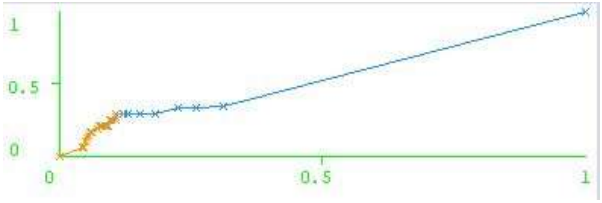

Fig. 8. ROC Curve using RF with Cost Sensitive.

Support Vector Machines (SVM)

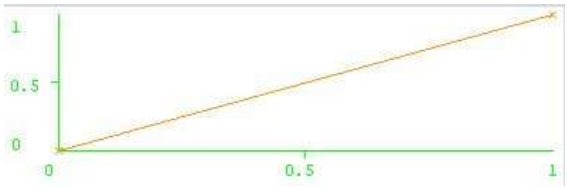

Fig. 9. ROC Curve using SVM without Cost Sensitive.

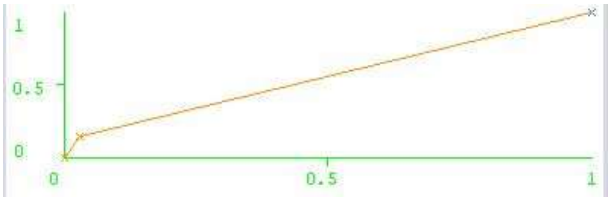

Fig. 10. ROC Curve using SVM with Cost Sensitive.

K Nearest Neighbour (KNN)

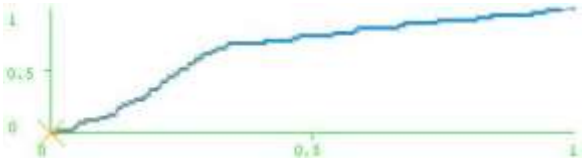

Fig. 11. ROC Curve using KNN without Cost Sensitive.

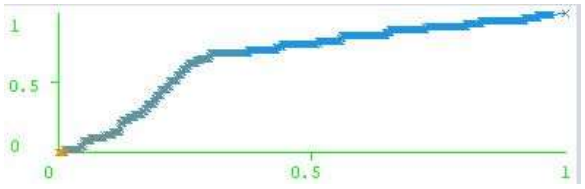

Fig. 12. ROC Curve using KNN with Cost Sensitive.

Multi Layer Perceptrons (MLP)

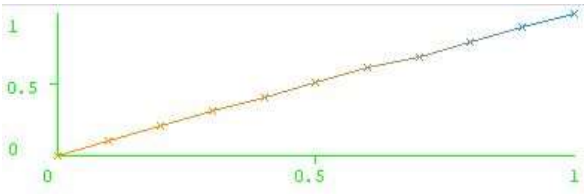

Fig. 13. ROC Curve using MLP without Cost Sensitive.

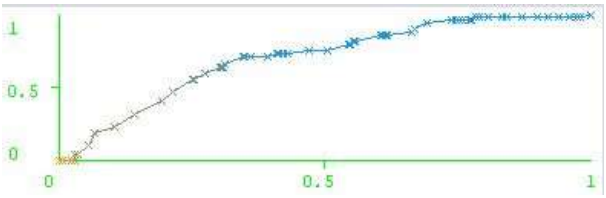

Fig. 14. ROC Curve using MLP with Cost Sensitive.

Adaptive Neuro Fuzzy Inference System (ANFIS)

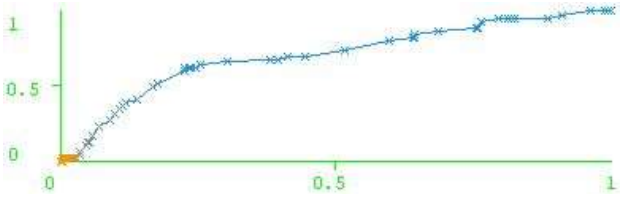

Fig. 15. ROC Curve using ANFIS without Cost Sensitive.

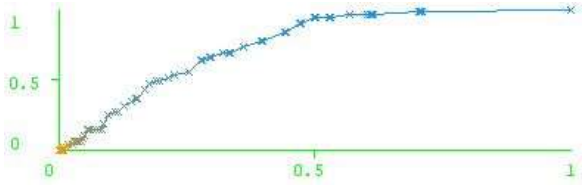

Fig. 16. ROC Curve using ANFIS with Cost Sensitive.

V. Conclusion and Future Work

In this work, we proposed an hybrid approach cost effective problems in software defect prediction. To reduce number of dimensions, we applied Principle Component Analysis. Ensemble ANFIS were constructed for cost effective learning of software defects. We compared the performance of proposed models, with algorithms specified in literature survey, using AuC values. Our proposed model got approximately 15% high Auc values over all datasets. As a future work, we can improve the AuC values by addressing High dimensionality, Class Imbalance, Cost effective Problems in SDP.

References

[1] Yan Naung Soe , Paulus Insap Santosa, Rudy Hartanto, "Software Defect Prediction Using Random Forest Algorithm", 2018 12th South East Asian Technical University Consortium (SEATUC), DOI: 10.1109/SEATUC.2018.8788881.

[2] Taek Lee, Jaechang Nam, Donggyun Han, Sunghun Kim, Hoh Peter In," Developer Micro Interaction Metrics for Software Defect Prediction", IEEE Transactions on Software Engineering,42(11).

[3] Fei Wu; Xiao-Yuan Jing; Xiwei Dong; Jicheng Cao; Baowen Xu; Shi Ying "Cost-Sensitive Local Collaborative Representation for Software Defect Prediction", 2016 International Conference on Software Analysis, Testing and Evolution (SATE), DOI: 10.1109/SATE.2016.24.

[4] Jinsheng Ren, Ke Qin, Ying Ma, Guangchun Luo," On Software Defect Prediction Using Machine Learning",Journal of Applied Mathematics,2014.

[5] Ayse Tosun, Ayse Bener, "Reducing false alarms in software defect prediction by decision threshold optimization", 2009 3rd International Symposium on Empirical Software Engineering and Measurement. DOI: 10.1109/ESEM.2009.5316006.

[6] CHENG Ming, WU Guoqing, YUAN Mengting and WAN Hongyan, "Semi-supervised Software Defect Prediction Using Task-Driven Dictionary Learning", Chinese Journal of Electronics,25(6).

[7] Igor Ibarguren, Jesus M.P´erez ´, Javier Mugerza , Daniel Rodriguez, Rachel Harrison" , The Consolidated Tree Construction Algorithm in Imbalanced Defect Prediction Datasets", 2017 IEEE Congress on Evolutionary Computation (CEC), DOI: 10.1109/CEC.2017.7969629.

[8] Yuanxun Shao,Bin Liu,Shihai Wang,Guoqi Li," Software defect prediction based on correlation weighted class association rule mining", Knowledge-Based Systems,196.

[9] Shuo Feng, Jacky Keung, Xiao Yu, Yan Xiao, Kwabena Ebo Bennin, Md Alamgir Kabir, Miao Zhang, "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction", Information and Software Technology,129.

[10] Rakesh Rana , Miroslaw Staron , Christian Berger , Jorgen Hansson , Martin Nilsson , Wilhelm Meding , Analyzing Defect Inflow Distribution and Applying Bayesian Inference Method for Software Defect Prediction in Large Software Projects, The Journal of Systems & Software (2016), doi: 10.1016/j.jss.2016.02.015.

[11] Guisheng Fan, Xuyang Diao, Huiqun Yu, Kang Yang and Liqiong Chen,"Software Defect Prediction via Attention-Based Recurrent Neural Network", Scientific Programming,2019.

[12] Sushant Kumar Pandey, Ravi Bhushan Mishra, Anil Kumar Tripathi,"BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques",Expert Systems with Applications,144,2020.

[13] Rodrigo A. Coelho; Fabrício dos R.N. Guimarães; Ahmed A.A. Esmin," Applying Swarm Ensemble Clustering Technique for Fault Prediction Using Software Metrics", 2014 13th International Conference on Machine Learning and Applications, DOI: 10.1109/ICMLA.2014.63.

[14] Shamsul Huda, Kevin Liu, Mohamed Abdelrazek, Amani Ibrahim, Sultan Alyahya, Hmood Al-Dossari and Shafiq Ahmad," An ensemble Oversampling Model for Class Imbalance Problem in Software Defect Prediction", SPECIAL SECTION ON SOFTWARE STANDARDS AND THEIR IMPACT IN REDUCING SOFTWARE FAILURES,2018.

[15] Shanthini. A, R M Chandrasekaran, "Analyzing the Effect of Bagged Ensemble Approach for Software Fault Prediction in Class Level and Package Level Metrics", International Conference on Information Communication and Embedded Systems (ICICES2014), DOI: 10.1109/ICICES.2014.7033809.

[16] Nageswara Rao Moparthi, Dr. N. Geethanjali," Design and implementation of hybrid phase based ensemble technique for defect discovery using SDLC software metrics", 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), DOI: 10.1109/AEEICB.2016.7538287.

[17] Steven Young; Tamer Abdou; Ayse Bener," A Replication Study: Just-in-Time Defect Prediction with Ensemble Learning", 2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE).

[18] Arvinder Kaur and Kamaldeep Kaur," Performance Analysis of Ensemble Learning for Predicting Defects in Open Source Software", 2014 International Conference on Advances in Computing,Communications and Informatics (ICACCI).

[19] Thanh Tung Khuat, My Hanh Le, "Ensemble learning for software fault prediction problem with imbalanced data", International Journal of Electrical and Computer Engineering (IJECE),9(4),2019.

[20] Jaroslaw Hryszko, Lech Madeyski, ” Cost Effectiveness of Software Defect Prediction in an Industrial Project”, Foundations of Computing and Decision Sciences,43(1),2018.

[21] Kazuya Tanaka; Akito Monden; Zeynep Yücel,” Prediction of Software Defects Using Automated Machine Learning”, 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD),DOI: 10.1109/SNPD.2019.8935839.

[22] Pradeep Singh,” Stacking based approach for prediction of faulty modules”, 2019 IEEE Conference on Information and Communication Technology (CICT).

[23] Haitao He, Xu Zhang, Qian Wang, Jiadong Ren, Jiaxin Liu, Xiaolin Zhao, Yongqiang Cheng,” Ensemble MultiBoost Based on RIPPER Classifier for Prediction of Imbalanced Software Defect Data”, IEEE Access,7.

[24] Z. Li, X. Jing, X. Zhu and H. Zhang, "Heterogeneous Defect Prediction Through Multiple Kernel Learning and Ensemble Learning," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, 2017, pp. 91-102, doi: 10.1109/ICSME.2017.19.

[25] X. Xia, D. Lo, S. J. Pan, N. Nagappan and X. Wang, "HYDRA: Massively Compositional Model for Cross-Project Defect Prediction," in IEEE Transactions on Software Engineering, vol. 42, no. 10, pp. 977-998, 1 Oct. 2016, doi: 10.1109/TSE.2016.2543218.

[26] Boehm, B. W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[27] Satya Srinivas Maddipati, Dr. G Pradeepini,Dr. A Yesubabu,” Software Defect Prediction using Adaptive Neuro Fuzzy Inference System”, International Journal of Applied Engineering Research ,ISSN 0973-4562 ,Volume 13, Number 1 (2018) pp. 394-397.

[28] R Anand, Dr. K David, Dr.S. Stanley Sagayaraj,”Identifing the impact of Defects among the Defect types in Software Development Proects”, ICSTM, May 2015,pp. 146-152.