International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-5, March 2020

Software Defect Prediction using Support Vectorised Data and Intelligent Techniques

Kovuru Vijaya Kumar, Ch GVN Prasad

Abstract*: Software enhances the working capability of any business. Developing such a software entrusts the developing organization to build defect free software. In this context we have used PC1 dataset(NASA dataset) which has sufficient parameters for analysis. Intelligent techniques using different methodologies have been applied exhaustively on the PC1 data to find out the best intelligent technique for software defect. As the PC1 data is highly imbalanced data, there was biasness in the prediction of the intelligent techniques. Hence, to overcome this issue, in this paper we tried to propose best balancing method along with the intelligent technique to predict the software defect accurately.

Keywords*: Software Defect Prediction, Decision Tree (DT), Support Vectorised data, Logistic Regression(LR), synthetic minority over-sampling technique (SMOTE).

I. INTRODUCTION

Software has become one of the basic part of almost every device which is used in today's world. Due to the steep decrease in the hardware prices and evolving of advanced hardware configurations, most of the people are welcoming new electronic gadgets in their daily life for fulfilling their needs. The software developing industries too have come in high demand and a need for successful software has emerged. Literature shows that only few modules contribute to most of the software defects [19,20]. Because of these few modules software failure occurs and leads to customer dissatisfaction and hikes the cost of maintenance [21]. Software fault prediction models come handy to address these type of problems provided enough data with sufficient features is there for predicting. With the help of these models software engineers can concentrate on enhancing the quality of a software and utilize the resources efficiently [22].

They can build a successful software provided they have been given early warning predictions regarding the software failures by the software prediction models. Every software development organization intends to identify the low quality design in the early stage and tries to improvise it before it is too costly to fix it [23].

Revised Manuscript Received on March 30, 2020.
* Correspondence Author
Kovuru Vijaya Kumar*, Computer Science and Engineering department, Rayalaseema University, Kurnool, Andhra Pradesh, India. Email: vijay657@yahoo.co.in
Ch GVN Prasad, Department of Computer Science & Engineering, Sri Indu College of Engg & Tech, Hyderabad. India

EMAIL: PRASADCH204@GMAIL.COM

The objective of this paper is to identify how effectively the software defect prediction can be done using Machine learning techniques for predicting defect-prone software in the context of PC1 NASA datasets. As the dataset is highly unbalanced data, additionally, we also address, by doing what type of preprocessing can we get good prediction results for this type of data.

The remaining part of the paper is organized in the following way. Section II gives the literature survey related to defect prediction in software. In Section III overview of the data description and data preparation is presented. The brief outline of techniques applied are discussed in Section IV, followed by the architecture of the proposed model and it's experimental setup is described in Section V. Section VI describes results and discussions. Finally, Section VII gives the conclusion of the paper.

II. LITERATURE SURVEY

Software defect prediction (SDP) aims to predict the defect proneness of new software modules with the historical defect data of a software so that its quality can be improved. Lot of different techniques, methods and models have been proposed for SDP problem.

The literature regarding the empirical studies which evaluate the proposed SDP models is also available in good numbers. However, every method has got its own unique advantages and disadvantages, so coming up with a reliable SDP model still remains challenging [3]. David Bowes et al. worked on the uncertainty of the classifiers like, SVM, Random Forest, RPart and Naïve Bayes in the context of SDP and has done the analysis of sensitivity[8].

Li. et al. proposed two stage classification of the SDP by using multiple classifiers and three way decision making[11]. Z. Li et al. worked on categorization of SDP method into four categories, namely prediction algorithms based on Machine learning, SDP that manipulates the data, empirical studies and prediction based on the effort [20]. Libo Li et al. observed that the classifiers results depend on the software project. According to him choosing the best classifier is difficult, so different dimensions has to be considered to overcome the potential bias in unbalanced data[5]. It has been observed that accuracy is affecting if data is having high dimension.

R. jayanthi et al. applied PCA(principle component analysis) for dimension reduction and finally used k-NN, SVM, Naïve Bayes and LDA for SDP[18].

C. Manjula et al. used modified GA for dimension reduction with the help of deep neural network for SDP. They modified GA with the help of new chromosome design technique and fitness function that alter the DNN technique using adaptive auto-encoder computation [13]. Tiejian Wang et al. proposed Multiple kernel learning method which maps the SDP data to a greater-dimensional space and also used ensemble learning to improvise the performance of the classifiers[10].

Most of the software defect prediction techniques focus on complexity software metrics of the defective code. These techniques do not capture different levels of source code semantics which are vital elements for constructing a good prediction models[14]. So, Hoa Khanh Dam et al. developed a novel prediction model that can learn the features that represent a source code automatically and can be used for prediction. They applied Long Short Term Memory (LSTM) and deep learning techniques[14] for capturing features like structure of programs and their semantic information.

Jian. Li. et al. Proposed a framework called Defect Prediction via Convolutional Neural Network (DP-CNN), that considers the programs based on Abstract Syntax Trees [7]. The nature of the software prediction data such as presence of noise, unbalanced, and high dimensional data has an effect to a large extent on classifiying models . K. Bashir et al. Proposed a joint framework to improve Software defect prediction models with the help of Iterative-Partition Filter to overcome the abnormality in the data[17].

In a data set, when the samples of one class is more than the other class, this type of problem is known as data imbalance problem. Lately, many methods were proposed to tackle this problem of imbalanced data[21]. Song et al categorized different SDP models into broadly five groups they are, Ensemble, Sub-sampling, special purpose, cost sensitive and imbalanced ensemble groups[6]. AdaBoost, Over-sampling methods are the one widely used to handle class imbalance problem [1]. But, these may also generate unnecessary noise also.

Laradji et al., had worked on the SDP using greedy forward selection feature selection method along with ensemble method to obtain the defect prediction[2]. L. Gong et al. proposed Cluster-based Over-sampling with noise filtering (KMFOS) approach to handle class imbalance problem in the context of Software Defect Prediction [21]. Mahmood et al., conducted a study of replicability as well as reproducibility of defect prediction in research areas to show the importance of any topic behind the replicability[12]. Tong et al worked on SDP using dual-stage ensembles and encoders [16]. Multi objective effort aware SDP model called as Just in Time(JIT) software defect predictor has been proposed by Chen et al. They have used logistic regression to build this JIT software defect predictor[15].

Czibula et al.,[4] proposed a novel classification model based on relational association rules mining which is an extension of ordinal association rules that describe numerical orderings between attributes that commonly occur over a dataset. Omer et al.,[9] used the software defect features as pairs to propose Feature Dependent Naive Bayes method.

In recent years, a number of many Machine learning techniques have been proposed for software fault prediction. But, unfortunately none of the papers have addressed how optimal the data has to be prepared before being supplied to any machine learning technique and which method would give good prediction result for a well-balanced software defect prediction data. So, we tried to explore this particular part of software defect prediction with optimal data, accuracy and recall.

III. DATA PREPARATION AND DESCRIPTION

The dataset used in this study was the mission critical NASA software projects data, which is publicly accessible from PROMISE Software Engineering Repository. The dataset contains total 10885 number of instances, each instances contains 21 software metrics and the associated dependent Boolean variable which has the following two values: Defective (whether or not the module has any defects) or Non-Defective. The 21 independent metrics which are further divided into, 5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, and a branch-count.

It has been observed through visualization that, there is wide variation among the values of the each attributes, normalization of the data have been adopted. Normalization of data is done using the below mentioned equation,

$$va_i = \frac{(va_i - \min_a)}{(\max_a - \min_a)} \quad (1)$$

where va_i indicates individual value of attribute a, $\min_a$ and $\max_a$ indicate minimum and maximum values of that attribute.The dataset is highly imbalanced that contained 93% non-fault and 7% fault instances. If we supply this data to the intelligence techniques it may lead to biased decision. Hence, balancing has to be done, in such a way that the classifier's decision capability is not effected by the major class.

A. Balancing Techniques

The balancing methods such as Over-sampling, Under-sampling using Synthetic Minority Oversampling Technique (SMOTE) and support vectorized balancing of data have been applied during cleaning and data preparation.

i. **SMOTE:** SMOTE is a method in which the synthetic samples are generated randomly which closely resemble the class that contains less instances. This way it creates a balance between the two classes of the data.

ii. **Under-sampling:** Under-sampling is a method where few instances that belong to class of majority are randomly removed from the data set so that there is a striking balance between the two classes. For example,

a. *Quarter-undersample* data means that few instances belonging to majority class are eliminated so that they become one quarter of its original size.

b. Similarly, for *Half-undersample* data means that few instances belonging to majority class are eliminated so that they become half of its original quantity, so the balancing between the two classes is normalized.

c.In the similar lines, we have *three-quarter-undersample* data where, the majority class's three-fourth data is taken instead of the complete majority class.

iii. **Over-Sampling:** In this paper over-sampling is done as follows, the synthetic samples which closely resemble the minority class are generated with the help of SMOTE technique in the following quantities,

a.In *Double-over-sampling*, the whole set of minority class is replicated once(i.e., it is doubled).

b. Similarly, for T*riple-oversampled* data, it is tripled.

c.In *Quadruple-oversampled* data the data is quadrupled.

d. In *Quintuple-oversampled* data the whole minority class is quintupled in quantity with the help of SMOTE technique.

iv. Support Vector Data: Support vectorized data is that data which contains only support vectors that have been extracted using various kernels of support vector machine. For this purpose, we have used three kernels namely, Linear, Sigmoid and RBF kernels[28].

After the datasets are balanced using one of the above-mentioned methods, intelligence techniques are applied. All the experiments have been carried out using Hold-out method by taking a ratio of 70:30. Where for training seventy percent component is used and for testing the 30 percent component is used.

B. Metrics used for measuring the predictions

The prediction capability and the quality of the various classifiers are measured using Recall, Precision, accuracy and F1-score whose equations and definition are given as follows[11]:

Recall gives the quantity of the Truepositives, that are correctly identified.

$$\text{Re}call = \frac{(Truepositive)}{(Truepositive + Falsenegative)} \quad (2)$$

Similarly, we compute the Precision which gives the quality of the true positives that is given as,

$$\text{Pr}ecision = \frac{(Truepositive)}{(Truepositive + Falsepositive)} \quad (3)$$

Accuracy gives the combination of true positives and true negatives that are correctly identified out of all ratios. This is given as

$$Accuracy = \frac{(Truepositive + Truenegatives)}{(Truepositive + Falsepositive + Truenegatives + Falsenegatives)} \quad (4)$$

F1-score or simply F-score is also used to measure the balancing strike between the first two metrics. It is given as,

$$F - score = 2 * \frac{(\text{Re}call * \text{Pr}ecision)}{(\text{Re}call + \text{Pr}ecision)} \quad (5)$$

All the experiments have been carried out on a full-scale and the reading having been noted down meticulously.

IV. INTELLIGENT TECHNIQUES

The following intelligent techniques have been used with various datasets prepared as described in section III.

i. Decision Tree: Decision tree is one of the machine learning technique which classifies the given data with the help of the entropy and information gain function. It basically constructs a tree out of the given data, which contains the important conditional variables at the inner nodes and the root, and the class variables at the leaf. For more details the reader may refer to [24].

ii. Random Forest: Random forest is nothing but a collection of decision trees which individually predict the values of the classes and collectively give the result based upon the majority trees decision. This can be considered as the one of the hybrid model. In [25], the more details of the random forest can be obtained.

iii. KNN: K- Nearest Neighbor technique is one of the earliest classification technique. This works on the principle of "k" number of nearest neighbors, where if majority of the "k" nearest neighbors say that a specific instance belongs to certain class, then that instance is classified into that class. [26] gives the detailed functioning of KNN classifier.

iv. Logistic Regression: Logistic regression is another popular classifier that is widely used for classification problems that works on the similar lines of curve fitting but with the help of probabilities of the each instance belonging to specific class. More detail of its working methodology can be seen in [27].

v. Support Vector Machine: Support vector Machine(SVM) tries to construct a hyperplane for the given data in n-dimensional space with the help of various kernels like, linear, RBF and sigmoidal, etc., With the help of SVM, we can identify a subset of instances that lie on the border of the hyperplane that can clearly distinguish between the two sets of classes, called as "Support Vectors".

In this paper, we have extracted these support vectors to reduce the dimensionality of the data set vertically.

V. ARCHITECTURE OF THE MODEL

The structure or the architecture of the proposed model can be viewed below.

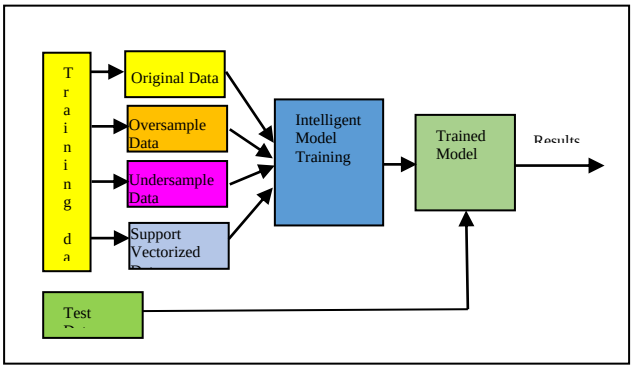

Fig. 1. Architecture of the proposed model

First the whole dataset is taken and divided into two parts with the ratio of 70:30. The larger part is taken for training and the least part is used for testing. Now, the training data is utilized in four different ways as mentioned in the data description section with the help of its copies.

Now, first the whole training data called as Original Data which has been obtained after partitioning is used for training different intelligent models with various parameters and fine-tuned to give optimal efficiency thus obtained trained model's best results have been tabulated in Table- I after supplying the test data.

Similarly, in the second case the training data is taken and oversampled in various proportions as mentioned in the data description section and the same intelligent models are trained and their best results have been tabulated in Table- II to Table- V after supplying the test data to the trained model.

In the third case, the training data is balanced using the under-sampling techniques as described in the Section III and the intelligent models are trained and their best results obtained from trained model on the test data have been tabulated in Table- VI to Table- VIII.

Finally, the training data is balanced using Support vectors that are obtained using various kernels like Linear, Sigmoid and RBF. Thus obtained data is used for training the intelligent models and the best results given by the trained model on the test data have been tabulated from Table-IX to Table- XI.

A. EXPERIMENTAL SETUP

Complete experiments were done using phython spider on the google colab platform. .

All the experiments were done using a method called as holdout method. In this method the data is partioned into two parts. Normal standard ratio of partitioning is 30:70. The thirty percent part is used for testing and the seventy percent part is used for training.

Due to the imbalance in the data, the balancing or equal proportion of classes has to be maintained for getting the optimal results and good learning of the machine learning technique. So, various methods of balancing have been adopted as discussed in the Section III. Full-fledged experiments have been conducted using the four intelligent techniques explained in Section IV and various combinations of the balanced data are used to find the efficient one.

VI. RESULTS AND DISCUSSION

The results obtained using test data after comprehensive runs of the experiments are tabulated from Table-I to Table-XI. The measures considered for evaluation of the results are F-score, Accuracy, Recall and Precision. The definitions of these measure are given Section III.B.

i. Results of Original data

The detail description of the Table-I is given in the following paragraph.

Table- I: Original data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	73.82	0.35	0.32	0.33
RF	81.84	0.23	0.54	0.32
KNN	79.94	0.24	0.44	0.31
LR	81.26	0	0	0

Table-I gives the results of the test data supplied to a trained model, whose training is done with original data. Since the data is largely imbalanced, it can be observed from the table that, the recall and precision are very poor. So, it can also be observed that even the F-Score is very poor. The accuracy of almost all models are good. This is due to high ratio of the true negative samples in the data.

ii. Results of the Oversampling (OS)

Data oversampling is done using SMOTE technique. Oversampling is done in various compositions which are shown below.

We can see that Table-II represents the results obtained from test data and models trained on double oversampled data. It could be observed from the tables that there is slight improvement in the results for almost all models. But, the overall results in this table are also not satisifiable. Hence, the F-Score is also very poor.

Table- II: Double oversampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	74.37	0.38	.85	0.35
RF	81.75	0.24	0.84	0.33
KNN	79.94	0.24	0.84	0.31
LR	81.26	0.0	0.81	0.0

In Table-III we can see the oversampling has an effect in improving the performance of the models. Here we can see that the recall has improved considerably from the previous table, henceforth the F-Score has also improved. It must be noted that Decision tree has been consistently performing well for oversampled trained data upto double oversampled, but in Table-III we can see that Random Forest is overtaking the top position among all other models.

Table- III: Triple oversampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	67.52	0.64	0.84	0.63
RF	83.73	0.67	0.86	0.72
KNN	75.57	0.66	0.84	0.62
LR	69.42	0.01	0.69	0.02

From Table- IV we can see the improvement in results is significant and this time again Random forest is performing well over the quadruple oversampled training models. The F-Score has also improved considerably. But, as our main concern is recall, we can see that the KNN has also given good recall value equivalent to Random forest, but due to its low precision compared to Random forest it is ranking next to Random forest.

Table- IV: Quadruple oversampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	79.80	0.76	0.83	0.76
RF	86.31	0.82	0.87	0.83
KNN	76.95	0.82	0.85	0.75
LR	60.59	0.06	0.60	0.12

Table-V gives the results of the models trained on the quintuple oversampled data, where we can see again Random forest has toped the models in the table. KNN also has given good recall and precision due to which its F-Score is also good and it ranked second by beating Decision tree.

Hence, it is evident from all the oversampling data that the quintuple oversampled data has significantly improved the learning capability of the machine learning models by giving enough ratio of samples between two classes. It can been seen from the Fig. 2 that among all the oversampling techniques employed, the quintuple oversampled trained models are giving good results and recall values also for majority of the techniques used in this paper.

Table- V: Quintuple oversampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	81.82	0.83	0.83	0.82
RF	88.39	0.88	0.88	0.88
KNN	79.34	0.90	0.87	0.88
LR	63.27	0.44	0.60	0.54

Since, recall is our main concern, the immediate below figure shows the recall values of the various techniques used. Except the Logistic Regression all other techniques are doing good predictions for the quintuple oversampled data.

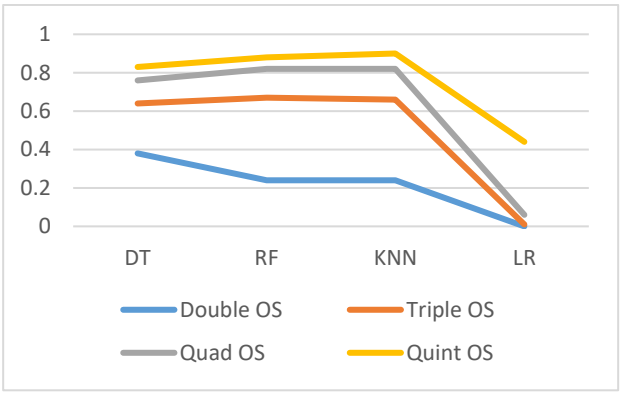

Fig. 2. **Recall values for oversampling.**

iii. Results of the Undersampling (US)

Another type of balancing the imbalanced data is using Undersampling techniques. Here, we have adopted random undersampling techniques for all undersamplings. Various types of undersampling techniques used described as follows.

Table- VI: Quarter-undersampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	61.08	0.60	0.61	0.60
RF	66.12	0.64	0.66	0.65
KNN	61.31	0.60	0.61	0.61
LR	56.27	0.19	0.54	0.31

Table- VI gives the results of the undersampled data that are obtained on the test data after training the models using the Quarter-undersampled data. It could be observed from the table that the models have performed very poor, but better than Original data trained models and Double oversampled data. Here, too it is clear from the table that Random forest has top the list of intelligent techniques.

Table- VII: Half-undersampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	64.34	0.47	0.74	0.46
RF	73.21	0.46	0.77	0.53
KNN	69.67	0.44	0.76	0.48
LR	68.03	0.00	0.68	0.01

As it can be seen form the Table- VII the Half-undersampled data has shown an deteriorating effect in terms of recall, precision and F-score. Similarly, it could be observed from Table- VIII that the Tree-quarter-undersampled data has also given very poor performance. The results of the Table- VIII are more deteriorating than the results of the Table- VII. Hence, it can be clearly understood that the undersampling technique is not suitable particularly for this type of data.

Table- VIII: Three-quarter-undersampled data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	71.38	0.38	0.81	0.39
RF	78.02	0.30	0.81	0.40
KNN	76.17	0.34	0.81	0.41
LR	75.68	0.00	0.76	0.00

The overall perforamance on the undersampling methods is depicted in a graph below in Fig. 3. Which shows how the intelligent models are performing on this type of data. It is evident from the Fig. 3. that the undersampling techniques are not suitable for this type of data.

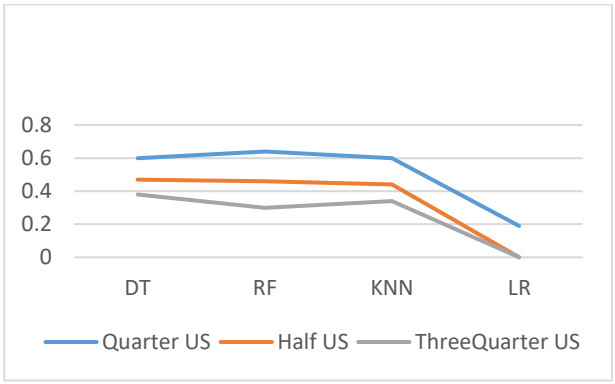

Fig. 3. Recall values for undersampling.

iv. *Results of the Support Vectorized data*

The Support Vector Machine is used on the raw/original data with various kernels and the support vectors thus obtained are extracted and used as a data for training different intelligent techniques. The tables following shows the results of the various intelligent techniques when they have been trained using support vectorized data.

Table- IX shows the results of the intelligent techniques which have been trained using the Sigmoid kernel generated support vectors. It can be seen from the table that the results are pretty good compare to all other methods which we had explored till now. There were total 1026 support vectors have been extracted out of which 528 were of non-defect class and 498 were of defect class. It could be observed from the table that almost all the techniques have performed well except LR. The Decision Tree has fared well compared to all other intelligent techniques used.

Table- IX: Sigmoid kernel based support vector data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	88.98	.90	.90	.89
RF	91.42	0.87	.89	.91
KNN	90.74	0.86	.88	.90
LR	41.81	.86	.00	.59

Next, we tried to extract support vectorsusing Linear kernel where we obtained 1266 support vectors of which 642 were of non-defect and 624 were defect class support vectors. Though we have obtained more number of support vectors, the performance of the intelligent techniques which has been trained using these data are poor compared to the techniques which had been trained using sigmoid kernel. In this method, Random Forest has given good result, but, was not even closer to any of the three techniques which have given good result when trained using sigmoid kernel. The results using Liner kernel are shown in Table- X.

Table- X: Linear kernel based support vector data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	66.74	.66	.67	.66
RF	70.53	.66	.69	.69
KNN	68.56	.68	.69	.68
LR	50.63	.77	.53	.60

Table- XI shows the results of the intelligent techniques which has been trained on the support vectorized data extracted using the RBF kernel. The results of the these techniques could be seen to be very poor compared to the techniques trained using other support vectorized data methods.

Table- XI: RBF kernel based support vector data

Algorithm	Metric			
	Accuracy(%)	*Recall*	*Precision*	*F-Score*
DT	59.08	.60	.63	.58
RF	62.41	.61	.65	.60
KNN	58.77	.59	.62	.57
LR	54.44	.03	.54	.06

Fig. 4 shows the overall results of the intelligent techniques traind over different support vector data. We can see that the support vectors extracted using sigmoid kernel has contributed significantly clearly distinguishing the instances between the two classes. Where as other support vectors extracted using linear and RBF kernels were not so efficient in distinguishing the classes, hence their performance was poor over the performance of intelligent techniques trained on sigmoid kernel.

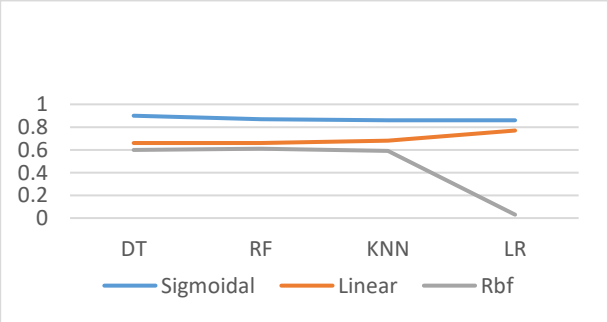

Fig. 4. Recall values for support vectorized data.

v. Comparision among all methods

At the outset when we compare different data balance handling methods used in the context of software defect prediction along with different intelligent techniques used for predicting the software defect, it could be observed that the support vectorized data using sigmoidal kernel has been consistant with respect to recall across all the intelligent techniques used compared to other balancing techniques like undersampling and oversampling.

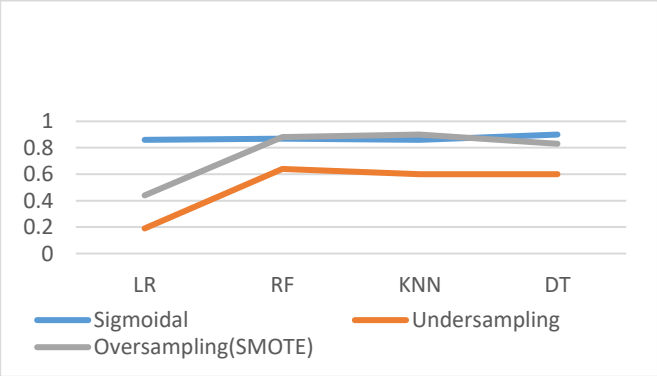

Fig. 5. Overall recall values.

The oversampling technique is ranking as second best balancing method among the balancing methods.

VII. CONCLUSION

For the Software Defect Prediction problem lot of machine learning techniques and hybrid techniques have been proposed. But, none of them have addressed the usage of the reduced data and mostly using support vectorized data for predicting the software defect, which takes very less quantity of data for training the intelligent techniques optimally and reduces their learning time without compromising on the accuracy of these techniques. In this way our paper is unique and addresses the above mentioned point and provides a new angle of dealing with the highly imbalanced data. The experimental analysis shows that the support vectorized data extracted using sigmoidal kernel gives good data using which most of the intelligent techniques can predict accurately.

ACKNOWLEDGMENT

The authors would like to thank the PROMISE Software Engineering Repository for providing the PC1 software defect prediction data set.

REFERENCES

1. Wang, Shuo, and Xin Yao. "Using class imbalance learning for software defect prediction." *IEEE Transactions on Reliability* 62.2 (2013): 434-443.
2. Laradji, Issam H., Mohammad Alshayeb, and Lahouari Ghouti. "Software defect prediction using ensemble learning on selected features." *Information and Software Technology* 58 (2015): 388-402.
3. Shepperd, Martin, David Bowes, and Tracy Hall. "Researcher bias: The use of machine learning in software defect prediction." *IEEE Transactions on Software Engineering* 40.6 (2014): 603-616.
4. Czibula, Gabriela, Zsuzsanna Marian, and Istvan Gergely Czibula. "Software defect prediction using relational association rule mining." *Information Sciences* 264 (2014): 260-278.
5. Li, Libo, Stefan Lessmann, and Bart Baesens. "Evaluating software defect prediction performance: an updated benchmarking study." *arXiv preprint arXiv:1901.01726* (2019).
6. Song, Qinbao, Yuchen Guo, and Martin Shepperd. "A comprehensive investigation of the role of imbalanced learning for software defect prediction." *IEEE Transactions on Software Engineering* (2018).
7. Li, Jian, et al. "Software defect prediction via convolutional neural network." *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017.
8. Bowes, David, Tracy Hall, and Jean Petrić. "Software defect prediction: do different classifiers find the same defects?." *Software Quality Journal* 26.2 (2018): 525-552.
9. Arar, Ömer Faruk, and Kürşat Ayan. "A feature dependent Naive Bayes approach and its application to the software defect prediction problem." *Applied Soft Computing* 59 (2017): 197-209.
10. Wang, Tiejian, et al. "Multiple kernel ensemble learning for software defect prediction." *Automated Software Engineering* 23.4 (2016): 569-590.
11. Li, Weiwei, Zhiqiu Huang, and Qing Li. "Three-way decisions based software defect prediction." *Knowledge-Based Systems* 91 (2016): 263-274.
12. Mahmood, Zaheed, et al. "Reproducibility and replicability of software defect prediction studies." *Information and Software Technology* 99 (2018): 148-163.
13. Manjula, C., and Lilly Florence. "Deep neural network based hybrid approach for software defect prediction using software metrics." *Cluster Computing* 22.4 (2019): 9847-9863.
14. Dam, Hoa Khanh, et al. "A deep tree-based model for software defect prediction." *arXiv preprint arXiv:1802.00921* (2018).
15. Chen, Xiang, et al. "MULTI: Multi-objective effort-aware just-in-time software defect prediction." *Information and Software Technology* 93 (2018): 1-13.
16. Tong, Haonan, Bin Liu, and Shihai Wang. "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning." *Information and Software Technology* 96 (2018): 94-111.
17. Bashir, Kamal, et al. "Enhancing software defect prediction using supervised-learning based framework." *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. IEEE, 2017.
18. Jayanthi, R., and Lilly Florence. "Software defect prediction techniques using metrics based on neural network classifier." *Cluster Computing* 22.1 (2019): 77-88.
19. Fenton, Norman E., and Niclas Ohlsson. "Quantitative analysis of faults and failures in a complex software system." *IEEE Transactions on Software engineering* 26.8 (2000): 797-814.
20. Koru, A. Güneş, and Jeff Tian. "An empirical comparison and characterization of high defect and high complexity modules." *Journal of Systems and Software* 67.3 (2003): 153-163.
21. Koru, A. Gunes, and Hongfang Liu. "Building effective defect-prediction models in practice." *IEEE software* 22.6 (2005): 23-29.
22. 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.
23. Malhotra, Ruchika, and Ankita Jain. "Fault prediction using statistical and machine learning methods for improving software quality." *Journal of Information Processing Systems* 8.2 (2012): 241-262.
24. Friedl, Mark A., and Carla E. Brodley. "Decision tree classification of land cover from remotely sensed data." *Remote sensing of environment* 61.3 (1997): 399-409.
25. Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." *R news* 2.3 (2002): 18-22.
26. Peterson, Leif E. "K-nearest neighbor." *Scholarpedia* 4.2 (2009): 1883.
27. Wright, Raymond E. "Logistic regression." (1995).
28. Furey, Terrence S., et al. "Support vector machine classification and validation of cancer tissue samples using microarray expression data." *Bioinformatics* 16.10 (2000): 906-914.

AUTHORS PROFILE

Kovuru Vijaya Kumar, received his B. Tech (Computer Science &Engineering) degree from Sri Venkateswara University, Tirupati and M.Tech (Computer Science &Technology) degree from GITAM College of Engineering, Andhra University. He worked as Assistant Professor at various engine ering colleges in Hyderabad. He has 7 years of teaching experience. He is currently research scholar at Department of Computer Science and Engineering, Rayalaseema University, Kurnool, Andhra Pradesh, India. His area of interest includes Data Warehousing & Data Mining, Artificial Intelligence, Compiler Design and Network Security.

Dr. Ch GVN Prasad, M.Tech, Ph.D(Experience - 23 years ; 12 years IT industry (8 years in National Informatics Centre, Govt. of India, as Scientist and Software Analyst in AT&T in US) and 11 years Teaching as Professor and HOD of CSE dept). He is currently working as Professor, in Department of Computer Science & Engineering in Sri Indu College of Engg & Tech. Guided many UG, PG and Phd projects as supervisor. Published several papers in international and national journals. Research areas include Data Mining, Image Processing, Neural Networks and Network Security.