International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Software Defect Projections Performance by Class Rebalancing

Ranjeetsingh Suryawanshi[1], Amol Kadam[2]

Submitted: 15/11/2023 **Revised:** 27/12/2023 **Accepted:** 07/01/2024

Abstract: Data is the most precious asset to any tech firm since it has ability to totally revolutionize an industry. However, great availability does not imply perfection; there are numerous issues with the current quality of publicly available datasets. Class imbalance is one such problem, it basically means that one class label appears more frequently than others, and it occurs practically everywhere, such as in medical diagnosis, natural language processing, fraud detection, and so on. This issue is extremely serious since it undermines the potential of existing machine learning models by lowering their ability to identify newer data that has never been seen before. Many strategies have been proposed to address this problem, including oversampling and under sampling. These procedures, however, either overgeneralize or over diversify the data points, making them unproductive. The ensemble technique is another solution with good outcomes but high complexity. However, to address this challenge more effectively, we propose a model that uses mean absolute deviation to find the best features that have the most impact on the outcome, thus lowering dimensionality, and the adaptive synthetic data creation technique to balance the data. The model prioritizes recall over accuracy, which is a crucial parameter when dealing with data imbalances. The findings of this model appear to be quite promising, with an accuracy of nearly 85% and a recall of 89%. The model's AUC curve is 0.93, demonstrating the model's ability to correctly detect positive and negative instances.

Keywords: *Class Imbalance, Cost sensitive learning, Median absolute deviation, Sampling methods, Software Defect Prediction*

1. Introduction

Predictive systems are in high demand right now because of the rise of machine learning. Every industry aspires to automate time-consuming operations, ranging from healthcare and education to geography and logistics. Software is one such popular industry, and the need of the hour is to create a model that can correctly identify bad programmes before they are released for usage. Algorithms such as Decision Trees, Random Forest, and Support Vector Machine are currently in use to build general models.

Though these are excellent models, the most significant impediment to their effectiveness is a lack of data, or, in other words, 'class imbalance,' in which the occurrence of one label is significantly more than that of the other. This problem is not limited to the software area, but also to other fields. As an example, consider credit card fault detection, where the occurrence of faults is extremely low in comparison to the majority class. One of the most difficult problems in online fraudulent transaction detection is class imbalance with overlap. To avoid being discovered, scammers have tried their hardest to make a fraudulent transaction seem as authentic as possible. Based on the divide-and-conquer theory, this research suggests a unique hybrid method to address the problem of class imbalance with overlap.

The problem of class imbalance causes the model to over fit and become less accurate when confronted with fresh data points that it has never seen before. As a result, there is a need to create a model that outperforms current methodologies while also being more functional.

Datasets used for software defect prediction have a greater number of non-faulty than defective cases. To mitigate this issue, oversampling techniques are often used to create additional, artificially faulty instances. Current methods either produce almost identical cases, leading to an excessive generalization. provide the innovative oversampling method known as Complexity-based OverSampling method, which can achieve low likelihood of false alarm and high probability of detection[1].

Due to biassed learning and poor fault prediction, models trained on unbalanced data provide erroneous predictions. Ruchika Malhotra et al found that Ensemble approaches are the preferred choice for software quality prediction modelling among developers and researchers. Software practitioners and developers will benefit from early defect detection and reduced testing costs and effort when the class imbalance problem is handled with Random OverSampling and Aglomerative Hierarchical Clustering[2].

[1] *Bharati Vidyapeeth Deemed To Be University, College of Engineering, India. ORCID ID: 0009-0009-5984-3401*

[2] *Bharati Vidyapeeth Deemed To Be University, College of Engineering, India. ORCID ID: 0000-0002-2350-4397*

** Corresponding Author Email: ranjeetsinghsuryawanshi@gmail.com*

To exclude many majority samples from the original dataset as well as a few minority class outliers, an anomaly detection model should be first trained on the minority samples. After that, the leftover samples can be combined to create an overlapping subset with a lower learning interference from the majority class and minority class than the original dataset, as well as a low imbalance ratio[3].

2. Related Work

Despite the great contributions for software defect prediction made by several scholars and academicians, The NASA research states that if a flaw is done during the requirement phase and is not corrected, it will cause substantially greater repairing costs when it reaches the testing and coding phases[4].

A predictive model's training and testing may be done with fewer features by using the feature selection approach. The main goals of lowering the feature count are to save computing expenses and avoid overfitting issues to improve model performance. In addition to using the elbow technique and inertia to determine the ideal number of clusters, K-means clustering is utilized to determine class labels[5].

Regardless of the classifier being used, Mike Wasikowski et al observe a considerable improvement in performance when we limit the number of features in the data set to about the same order of magnitude as the samples. When learning from high-dimensional unbalanced data, feature selection is important for getting the best potential outcomes[6].

Table 1. Comparative study for class imbalance in software defect prediction.

Approach	*Methods used*	*Performance metrics*	*Advantages*	*Limitation or Future work*
Data Level	US-PONR Under sampling [12]	AUC, MCC	Minimises the overfitting noise introduced by the creation of minority class samples	As noise datasets are generated for studies, artificial noise may be added.
	Neighbourhood based Under-Sampling [13]	AUC	maximize the visibility of minority data points and prevent data loss	The experiment may be repeated using a bigger dataset.
	SVMSMOTE Oversampling [14]	F Measure, AUC	get ideal border closer to the learnt boundary	The problem of overfitting occurs
	Random oversampling and SMOTE [15]	AUC, MCC	creates precise criteria that are additionally utilised to increase the classification's accuracy	In the future, the suggested method will be assessed on several, sizable datasets (such as the Prop dataset).
Algorithm level	convolutional neural network (CNN) and gated recurrent unit (GRU) [10]	AUC, MCC, AUCPR	SMOTE Tomek approach combined with CNN and GRU models gives good performance of SDP on datasets with unbalanced class distributions	Can be tested with more dataset
Cost sensitive	data level and algorithm level [16]	GMean, Balance, and AUC	It tries to minimize the total cost of misclassifications	Cost is not precisely known, must use approximations or ratios of proportionate

Testing engineers may make efficient use of testing resources without going overboard by anticipating the modules that are likely to have defects. Bejjanki et al proposed approach for creating new samples by figuring out the centroid of every feature of minority class samples which will reduce class imbalance. When implemented with commonly used machine methods, class imbalance reduction outperforms SMOTE and K-means SMOTE. When it comes to accuracy, precision, and specificity, KNN outperforms other classifiers, but logistic regression excels in recall, F-measure, and geometric mean[7].

Results from the defect prediction system trained on noisy and unbalanced data are inconsistent and disappointing. Performance of traditional SDP models decreases when noise level in datasets rises and the learning process starts misclassifying the true class. The imbalanced dataset caused the classifiers to overfit, which produced disappointing performance results, even if the sampling strategy was not employed over the standard baseline models[8].

Predictions made by models that are developed using unbalanced datasets are unreliable due to their bias result. As a result, the model becomes less efficient[9].

Poor classifications result when there is an imbalance in the class distribution; although accuracy may be good, the model is unable to identify data examples in the minority class[10].To rectify the inaccurate assessment provided by cross-validation, the Ming Tan et al applies and modifies online change classification. They also use resampling approaches and updatable classification to enhance performance. The evaluation of this work on six open-source and one commercial project demonstrates that the precision may be increased by 12.2-89.5%, or 6.4-34.8 percentage points, by using both resampling approaches and updatable categorization[11].

[12][13][14][15][10][16]

There are significant approaches that are helpful in comprehending the unbalanced learning issues and that can be developed in three ways.1) Sampling 2) Cost-Sensitive Learning 3) Ensemble Learning

(1) Sampling: In sampling, various techniques may be used to modify or create a balanced data set.

Over-Sampling: In this category, two oversampling techniques were taken into consideration. In the first, known as random oversampling, the tiny class is randomly oversampled until it has the same number of instances as the other class. Focused oversampling is the second technique, wherein the minor class is oversampled with data that occurs around the borders between the idea and its opposite.

Undersampling : In the first, known as random undersampling, elements from the larger class are randomly removed until the size of another class is equal. The second method, focused undersampling, involves removing features that are farther away[17].

SMOTE Synthetic Minority Over-sampling Technique: Instead of oversampling with replacement, author provide an oversampling strategy where the minority class is oversampled by producing artificial instances. Rather than working in data space, this approach operates in feature space, producing synthetic instances[18].Because SMOTE creates the same amount of synthetic data samples for each actual minority instance without considering the instances that are nearby, there is a higher likelihood of overlapping between classes[19].

ADASYN Adaptive Synthetic Sampling: The foundation of ADASYN is the concept of adaptively creating minority data samples based on their spreads, for minority class samples that are more challenging to learn, additional synthetic data is produced. In addition to minimizing the learning bias brought about by the initial unbalanced spread of data, the ADASYN approach may adaptively move the decision boundary to concentrate on the samples that are challenging to learn[20].

(2) Cost-sensitive learning: These methods establish a matrix known as the cost matrix. To enable the efficient learning of unbalanced data sets, the cost matrix manages discrete faults or instances. This suggests that the inherent characteristics of unbalanced distributions cannot be altered by cost-sensitive learning strategies. Distinct cost matrices are used to detect inaccurate defect classification based on the properties of imbalanced data[21].

(3) Imbalanced Ensemble Learning: Shuo Wang et al developed three ensemble models, known as UnderBagging, OverBagging, and SMOTEBagging, respectively, and each used bagging to integrate each individual classifier. UnderBagging and OverBagging involves resampling examples, building each classifier in the ensemble iteratively using a subset, and selecting the class with the highest vote total When building a subset, SMOTEBagging includes the creation phase for synthetic instances. Two factors need to be determined, per SMOTE: the number of k nearest neighbours and the degree of oversampling from the minority class[22].

3. Methodology

Figure 1 shows proposed model for software defect prediction using mean absolute deviation to find the best features and ADASYN for synthetic data generation. The data repository used in proposed method is the software defect repository, a dataset made publicly available by NASA that gives software metrics and labels whether there is a defect in the written code or not based on various factors like the number of control flow branches, the complexity of the program, number of comments, etc. This data has already been checked and cleaned.

After loading this data, feature selection is applied to the dataset using the concept of median absolute deviation on each feature to select the 10 best ones, hence reducing the dimensionality (this is usually very helpful when the dataset is huge and confusing). Next, to deal with the minority label, the technique of ADASYN (Adaptive Synthetic Data Generation) is used to generate synthetic data and balance the numbers of yes and no.

This new dataset is divided for training data for creating model and testing data for evaluation of model.Random forest algorithm is used to train the model, and performance metrics are then calculated. The training and testing results are compared to understand the extent to which the class imbalance issue could be addressed.

Finally, all this is visualized in the form of a receiver operating characteristic curve, which further helps us understand the results of the proposed model

International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Table 2. Abbreviations used in this article

Abbreviated word	Full form of word
US-PONR	Undersampling Propensity Score Matching Based Oversampling and Noise Reduction
AUC	Area under the ROC curve
MCC	Matthew's correlation coefficient
AUCPR	Area under precision-recall curve
SMOTE	Synthetic Minority Over-sampling Technique
ADASYN	Adaptive Synthetic Sampling
MAD	Median Absolute Deviation

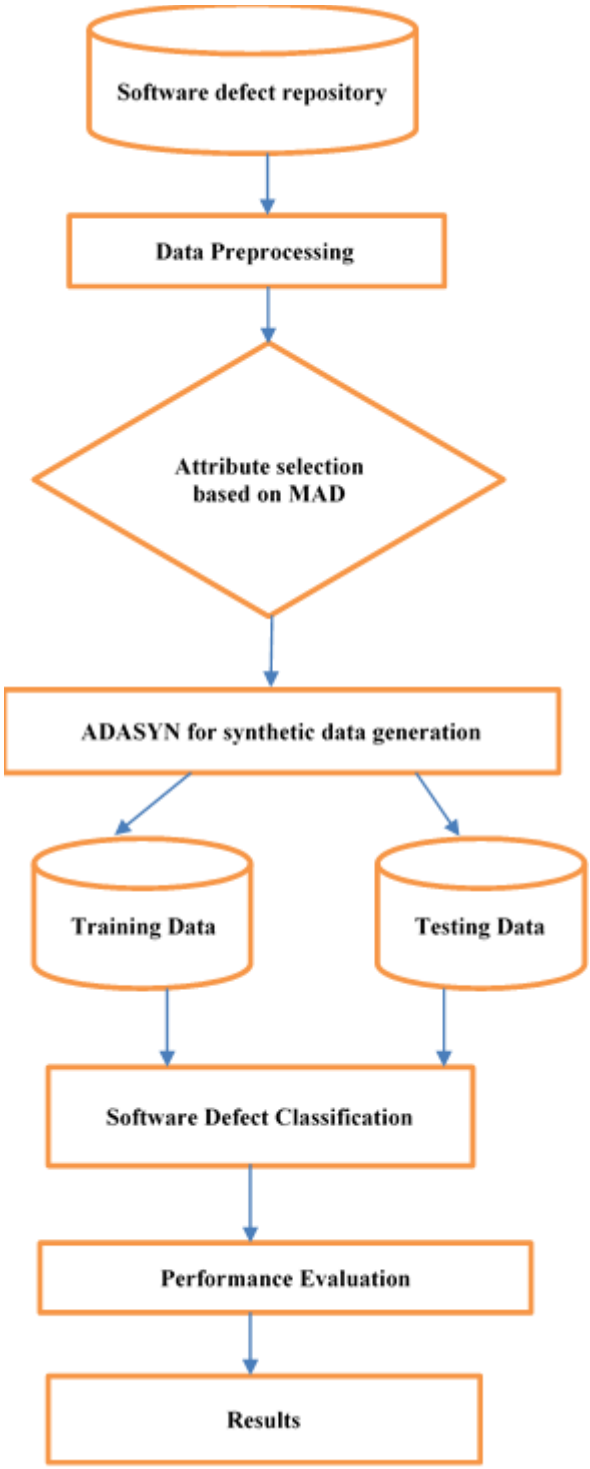

Fig 1. Software defect prediction proposed framework

Median Absolute Deviation (MAD):

Median Absolute Deviation or MAD is a robust measure of statistical dispersion. Mathematically it is calculated by taking median of absolute deviation of data points from the median. MAD is less sensitive to outliers and has high resistance against noisy data (which is common in imbalanced dataset) than other measures of dispersion like standard deviation and variance which are commonly preferred for attribute selection. Attribute selection using MAD helps to reduce dimensionality of dataset by retaining relevant features. The underlying concept of using MAD as a factor for attribute selection is, the attributes with highest MAD are most informative and contribute the most to the prediction of target variable. MAD as a scoring function helps in the identification of attributes that exhibit significant variability (has a larger spread of data), which is very crucial for distinguishing different classes especially when we must deal with the issue of class imbalance. Attributes with high MAD scores are more likely to help in building a better predictive model as they can capture essential patterns and characteristics. In simple words we can say that MAD will help us identify the attributes that are

most discriminatory between the majority and minority classes which have the biggest differences in values between the two classes. For example, we are trying to identify a patient who has a particular disease or not, the attributes that are most discriminatory between majority and minority classes will be symptoms, test results, risk factors, etc. These attributes will more accurately talk about which patients are more likely to have the disease due to their larger spread than other attributes. Mathematically MAD can be expressed as,

$$MAD = median(|x_i - x'|)$$

where

x_i *is the datapoint*

x'*is the median of the dataset*

For evaluation We will consider a sample feature from the JM1[23] dataset of NASA software defect repository named: BRANCH_COUNT (first 15 values already arranged in an ascending order)
The values are as follows:
[29, 39, 47, 65, 67, 163, 175, 187, 240, 338, 344, 405, 464, 503, 826]

Median of 15 observations is $\left(\frac{15+1}{2}\right) th\ observation = 8th\ observation$

Median = 187

Absolute differences:

- |826 - 187| = 639
- |29 - 187| = 158
- |405 - 187| = 218
- |240 - 187| = 53
- |464 - 187| = 277
- |187 - 187| = 0
- |344 - 187| = 157
- |47 - 187| = 140
- |163 - 187| = 24
- |67 - 187| = 120
- |503 - 187| = 316
- |175 - 187| = 12
- |39 - 187| = 148
- |338 - 187| = 151
- |65 - 187| = 122

Arranging all these values again in ascending order:

[0, 12, 24, 53, 120, 122, 140, 148, 151, 157, 158, 218, 277, 316, 639]

Now we calculate median of all these deviation values:

Median Absolute Deviation = 148. This will give top 10 best feature from data.

Appling Adaptive Synthetic sampling as follows:

1. It will calculate the distance of each minority class sample nearest to its neighbour.
2. Then it selects minority class samples with highest distances to its nearest neighbour.
3. For the selected minority class sample, a randomly selected minority class sample is chosen and new data is generation by taking a random value between the minority class sample and the nearest neighbour.

Steps 2 and 3 are repeated until suitable number of data points are generated.

This unique and adaptive approach makes the technique more suitable for class imbalance issues.

Figure 2 shows the frequency of labels in the dataset before any modification.

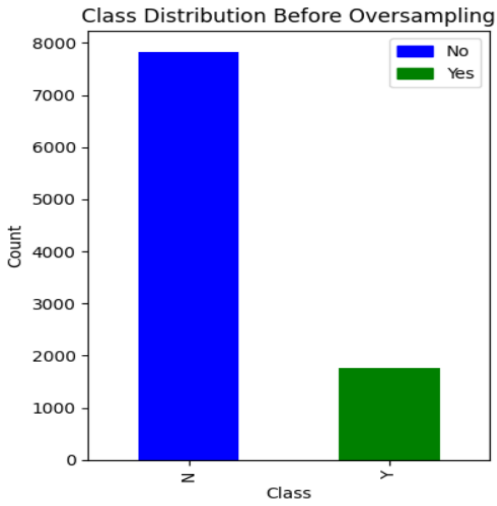

Fig 2. Class distribution before Oversampling (ADASYN)

Pseudocode

Load the dataset from 'JM1.csv'

For each column in the dataset:

If column datatype is 'object':

Encode the column using Label Encoding

Define a custom MAD scoring function (mad_score)

Use SelectKBest to select the most informative features based on MAD: X_selected = SelectKBest(score_func=mad_score, k=10).fit_transform(X, Y)

\\Apply ADASYN for oversampling:

X_over, y_over = ADASYN.fit_resample(X_selected, y)

\\Split oversampled data into training and testing sets:

Xtrain, Xtest, ytrain, ytest

\\Create a Random Forest Classifier

\\Get predicted probabilities for positive class for training and testing data:

y_train_prob = rf.predict_proba(X_train)[:, 1]

y_test_prob = rf.predict_proba(X_test)[:, 1]

\\Calculate ROC curve values:

Calculate AUC (Area Under the ROC Curve) for training and testing data

\\Evaluate classifier performance:

Calculate training accuracy, precision, and recall

Calculate testing accuracy, precision, and recall

4. Result Discussion

Figure 3 shows the frequency of labels after synthetic data generation:

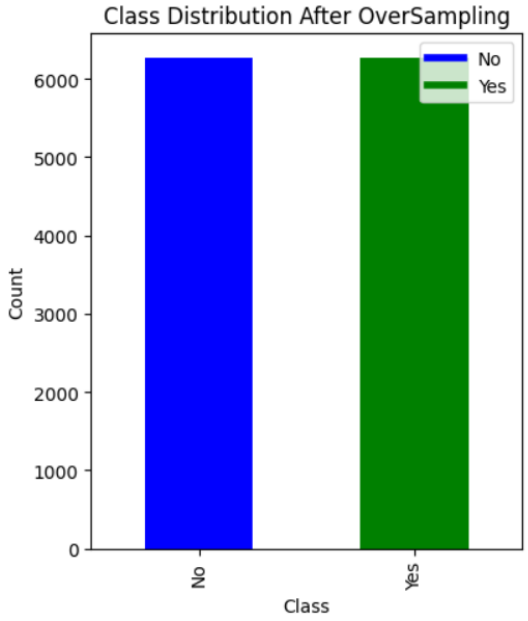

Fig 3. Class Distribution after Oversampling (ADASYN)

Various oversampling techniques, along with the proposed model, have been applied to the software defect dataset. We can clearly observe that accuracy cannot be the only measure of how well the model has been trained, i.e., though the base machine learning model (trained on a random forest classifier) may have an accuracy of 0.82804, the recall is extremely low, suggesting it is unable to correctly identify the positives, which is obvious due to the issue of class imbalance. This is not a good sign for a machine learning model. Hence, we go for oversampling methods to balance the number of positives and negatives. The standard oversampling technique is SMOTE (Synthetic Minority Oversampling Technique), but we have opted for ADASYN (Adaptive Synthetic Oversampling Technique). Firstly, because ADASYN generates synthetic data points based on difficulty level, unlike SMOTE, which merely works on the concept of generating minority classes based on nearest neighbour only, Secondly, from the results, we can see that SMOTE may have a little higher accuracy than ADASYN, but it is suitably compensated by a higher recall. But all these techniques are very straightforward, which may not give better results. This is where our proposed model helps to give exceptionally good results. High accuracy, precision, and recall suggest it can correctly identify both positive and negative samples from the dataset, hence making it a more progressive solution over traditional methods.

A similar conclusion can be inferred from the ROC (Receiver Operating Characteristic curve) curves of all these methods. Figure 4 clearly shows the robustness of the proposed model, which has an AUC (Area Under the Curve) value of 0.93, which shows it can comfortably identify positive and negative instances and will also accurately identify newer data points

better than other models, which only rely upon simple oversampling, undersampling, or ensemble models.

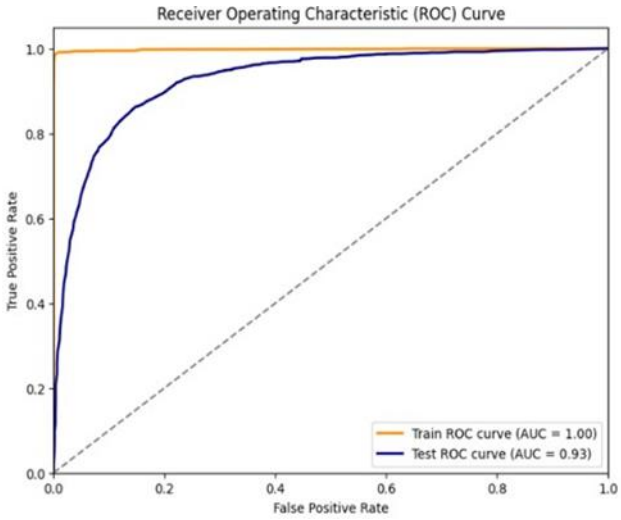

Fig 4. ROC Curve for MAD + ADASYN

Figures 5, 6, and 7 also give us a sense of comparison between the various techniques. From AUC value of Simple ML Model ADASYN and SMOTE in Figures 5, 6, and 7 shows that it correctly identifies positive and negative instances for training data but for test data its AUC value for Simple ML Model 0.76, ADASYN 0.74 and for SMOTE 0.74 which is low score compared to MAD + ADASYN value.

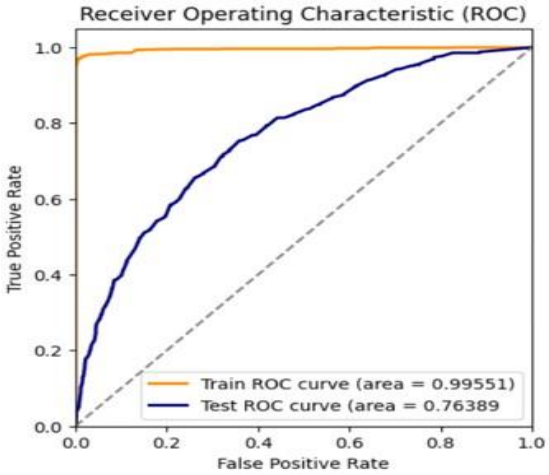

Fig 5. ROC for Simple ML Model

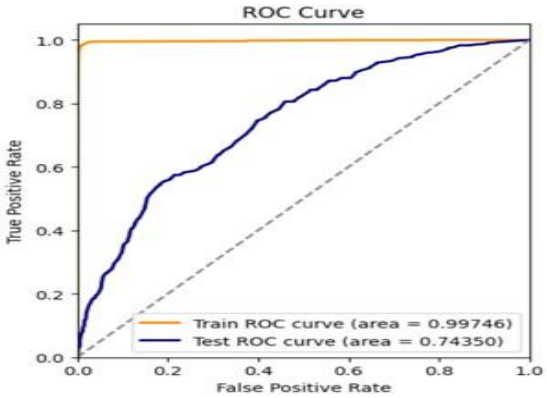

Fig 6. ROC Curve for SMOTE

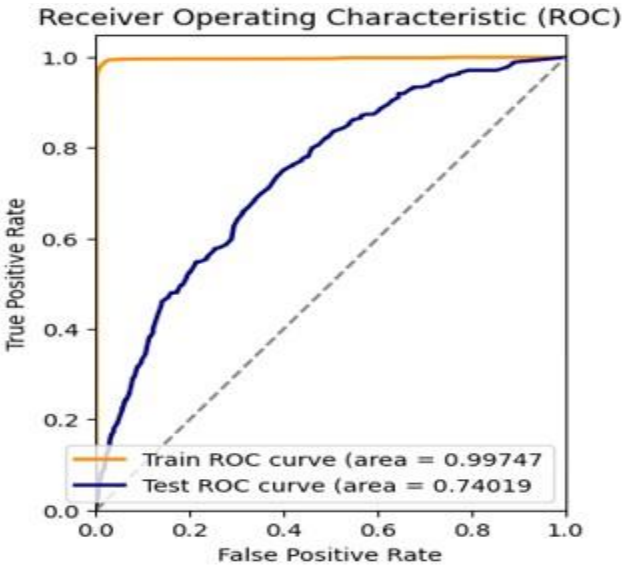

Fig 7. ROC Curve for ADASYN

Table 3. Comparative result after balancing class data for JM1 dataset

Sr. No.	Technique	Accuracy	Precision	Recall
1.	Simple Machine Learning Model	0.82804	0.58929	0.18857
2.	SMOTE	0.78583	0.41689	0.43714
3.	ADASYN	0.77749	0.40494	0.46857
4.	MAD + ADASYN	0.85497	0.82364	0.89440

Table 3 shows comparison result for Simple Machine Learning Model, SMOTE, ADASYN, MAD + ADASYN for NASA's JM1

Dataset. From result we conclude recall value using MAD+ADASYN has been significantly improved compared to other technique which helps to identify positive and negative instances correctly and will also identify newer data points better than other models.

5. Conclusion

This paper addressed issue of class imbalance using mean absolute deviation to find the best features and ADASYN for synthetic data generation. This work demonstrate that recall is more important than accuracy while handling class imbalance. We have studied various ways to handle class imbalance like sampling, Cost-sensitive learning, Ensemble Learning. A recall of 89% and an accuracy of around 85% suggest that our model's results are quite positive. The AUC curve for the model is 0.93, indicating that it can accurately identify both positive and negative events. Future work can be extended software defect prediction on commercial projects.

6. References and Footnotes

Author contributions

Ranjeetsingh Suryawanshi: Data collection and analysis, developing methodology, Design and Development of an application.. **Amol Kadam:** Reviewing and Editing of the article.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] S. Feng *et al.*, "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction," *Inf. Softw. Technol.*, vol. 129, no. April, p. 106432, 2021, doi: 10.1016/j.infsof.2020.106432.

[2] R. Malhotra and J. Jain, "Predicting defects in imbalanced data using resampling methods: An empirical investigation," *PeerJ Comput. Sci.*, vol. 8, pp. 1–34, 2022, doi: 10.7717/peerj-cs.573.

[3] Z. Li, M. Huang, G. Liu, and C. Jiang, "A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection," *Expert Syst. Appl.*, vol. 175, no. July 2020, p. 114750, 2021, doi:

10.1016/j.eswa.2021.114750.

[4] S. S. Rathore and S. Kumar, "Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study," *Appl. Intell.*, vol. 51, no. 12, pp. 8945–8960, 2021, doi: 10.1007/s10489-021-02346-x.

[5] C. L. Prabha and N. Shivakumar, "Software Defect Prediction Using Machine Learning Techniques," *Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020*, pp. 728–733, 2020, doi: 10.1109/ICOEI48184.2020.9142909.

[6] M. Wasikowski and X. W. Chen, "Combating the small sample class imbalance problem using feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1388–1400, 2010, doi: 10.1109/TKDE.2009.187.

[7] K. K. Bejjanki, J. Gyani, and N. Gugulothu, "Class imbalance reduction (CIR): A novel approach to software defect prediction in the presence of class imbalance," *Symmetry (Basel).*, vol. 12, no. 3, 2020, doi: 10.3390/sym12030407.

[8] S. K. Pandey and A. K. Tripathi, "An empirical study toward dealing with noise and class imbalance issues in software defect prediction," *Soft Comput.*, vol. 25, no. 21, pp. 13465–13492, 2021, doi: 10.1007/s00500-021-06096-3.

[9] S. Pandey and K. Kumar, "Software Fault Prediction for Imbalanced Data: A Survey on Recent Developments," *Procedia Comput. Sci.*, vol. 218, pp. 1815–1824, 2022, doi: 10.1016/j.procs.2023.01.159.

[10] N. A. A. Khleel and K. Nehéz, "A novel approach for software defect prediction using CNN and GRU based on SMOTE Tomek method," *J. Intell. Inf. Syst.*, vol. 60, no. 3, pp. 673–707, 2023, doi: 10.1007/s10844-023-00793-1.

[11] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online Defect Prediction for Imbalanced Data," *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 99–108, 2015, doi: 10.1109/ICSE.2015.139.

[12] H. Shi, J. Ai, J. Liu, and J. Xu, "Improving Software Defect Prediction in Noisy Imbalanced Datasets," *Appl. Sci.*, vol. 13, no. 18, 2023, doi: 10.3390/app131810466.

[13] S. Goyal, "Handling Class-Imbalance with KNN (Neighbourhood) Under-Sampling for Software Defect Prediction," *Artificial Intelligence Review*, vol. 55, no. 3. pp. 2023–2064, 2022. doi: 10.1007/s10462-021-10044-w.

[14] T. A. Ruchika Malhotra, Vaibhav Agrawal, Vedansh Pal, "Support Vector based Oversampling Technique for Handling Class Imbalance in Software Defect Prediction," *2021 11th Int. Conf. Cloud Comput. Data Sci. Eng.*, vol. 978-1–6654, pp. 1078–1083, 2021.

[15] N. A. A. Khleel and K. Nehéz, "Software defect prediction using a bidirectional LSTM network combined with oversampling techniques," *Cluster Comput.*, vol. 0123456789, 2023, doi: 10.1007/s10586-023-04170-z.

[16] R. Malhotra and J. Jain, "Predicting defects in object-oriented software using cost-sensitive classification," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1022, no. 1, 2021, doi: 10.1088/1757-899X/1022/1/012112.

[17] N. Japkowicz and S. Stephen, "The class imbalance problem A systematic study fulltext.pdf," vol. 6, pp. 429–449, 2002.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique Nitesh," *J. Artif. Intell. Res.*, vol. 16, no. Sept. 28, pp. 321–357, 2002, [Online]. Available: https://arxiv.org/pdf/1106.1813.pdf%0Ahttp://www.snopes.com/horrors/insects/telamonia.asp

[19] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.

[20] S. He, H., Bai, Y., Garcia, E., & Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *IJCNN 2008.(IEEE World Congr. Comput. Intell. (pp. 1322– 1328)*, no. 3, pp. 1322– 1328, 2008.

[21] A. Balaram and S. Vasundra, "Sampling-based Software Prone Technique for an Optimal Prediction of Software Faults," *Indian J. Comput. Sci. Eng.*, vol. 13, no. 4, pp. 981–991, 2022, doi: 10.21817/indjcse/2022/v13i4/221304009.

[22] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," *2009 IEEE Symp. Comput. Intell. Data Mining, CIDM 2009 - Proc.*, pp. 324–331, 2009, doi: 10.1109/CIDM.2009.4938667.

[23] Promise Software Engineering Repository. Available online: http://promise.site.uottawa.ca/SERepository/datasets-page.html