International Journal of Advanced Science and Technology
Vol. 29, No. 6, (2020), pp. 2129 - 2141

Ensemble Based Classification for Class Imbalanced Credit Card Fraudulent Data

S.Priya*, Siddharth Agarwal, Pankaj Thakur, Annie Uthra

Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Kattankulathur
**priyas3@srmist.edu.in,annieu@gmail.com,pankajpream524@gmail.com,*
sid7549@hotmail.com

Abstract

Knowledge extraction from imbalanced data has been receiving increasing interest in recent years. Most of the real world problems including credit card transaction frauds, disease prediction and so on have huge data instances but the number of positive instances in those datasets is far lesser than the number of negative instances. Suppose 99% of the data tuples come from the majority class but only 1% of the instances are from the minority class, and our classifier classifies each data tuple as majority instance, then the accuracy of our model will of course be 99% but this classifier is of no use as it doesn't detect any minority instance which is the main purpose of the classifier. This is why the data needs to be balanced first in order to train our classifiers. The balanced data can now be used to train our classification model. There are so many classification algorithms present, and each of them has their own pros and cons. So, in order to achieve higher accuracy and better results various individual classification algorithms including Naive Bayes, Decision Trees and Random Forests are used. After getting the results of the above mentioned classifiers, ensemble models have been applied that includes Voting Classifier, XGBoost and ADABoost. It is seen that the efficiency of ensemble classifiers in data classification is much higher than that of individual algorithms.

Keywords: *Class Imbalance, Ensemble Classifier, Sampling, Machine Learning.*

1. Introduction

Many of the real world problems have their datasets that exhibit class imbalance. Imbalanced datasets are the ones in which the number of one or more classes is much lesser than others. Imbalanced datasets can be intrinsic i.e. the domain of the dataset might be biased towards one class and very few cases belong to the other classes. Also, the data collection method might be inefficient that leads to imbalanced datasets. The class with the highest number of examples is known as the majority class and the class with very few examples in the dataset is known as the minority class. In imbalanced datasets, rare events are very hard to predict. Some of the examples of events that form imbalanced datasets are credit card frauds, telecom frauds, medical diagnosis, and network intrusion detections. Rare events are very hard to predict because these events occur only once or twice in about a thousand records.

Most of the standard classifiers including SVMs (Support Vector Machine)and decision trees work well on balanced datasets. While facing imbalanced datasets, the classification results are fine for the majority class but not so good in case of the minority classes [1]. In a decision tree classifier, it's nodes represent the attributes of the data instances, values of those attributes are represented by the edges, and the leaf nodes are the class labels. Due to the imbalanced class problem, the decision tree classifier needs to create more tests in order to distinguish the minority classes from majority classes. This may lead to over fitting of the data hence reducing the accuracy of the classifier. For a binary classification problem, the imbalance ratio or the ratio of number of data instances in the minority class to the number of data instances in the majority class is taken into consideration. In some cases, the ratio can be very drastic like 0.001 or even worse. We have used the credit card fraud dataset which has an imbalance ratio of 0.0017.

To overcome the class imbalance issue, we can either add new samples or remove some of the existing samples. The process of adding new samples in the set of existing samples is known as Oversampling. The process of removing samples from the existing samples is known as Under sampling techniques. Through various experiments conducted by various scientists and researchers, it has been found that Oversampling is generally better in giving better classification results than other under sampling techniques. Under sampling may lead to loss of those data instances which might be quite important to the training of our classifier. Currently there are various Oversampling and under sampling methods present that have proved to be useful in removing the class imbalance problem from our dataset. The major Oversampling algorithms include SMOTE [2](Simple Minority Oversampling Technique), Borderline SMOTE, ADASYN (Adaptive Synthetic Sampling for Imbalanced Data), MSMOTE (Modified Simple Minority Oversampling Technique) and so on. Oversampling means generating synthetic data for the minority class while Under sampling means removing data from the majority class. The major under sampling techniques include Near Miss algorithm, Random under sampling. For the classification task, rather than relying on only one algorithm, we can use the ensemble model to get better predictive accuracy. The main idea behind the ensemble method is weighing several individual classification algorithms, and combining them in order to design a classification model that outperforms the individual classifier. The prediction error decreases when the ensemble model is used in place of relying on any single model. The Ensemble model aggregates more than one model of same or different algorithms that are trained on the same dataset or set of different datasets. There are various ensemble methods that include bagging, boosting etc. Examples of Ensemble models include Voting Classifier, ADABoost, XGBoost, Gradient tree Boosting etc.

The rest of this paper is organised as follows. Section II discusses about the previous work. The proposed work is discussed in Section III. The implementation results and discussions are given in Section IV and Section V followed by conclusion in Section VI.

2. Literature Survey

To overcome the issue of class imbalance, various oversampling and under sampling techniques are present. SMOTE is one of the most widely used oversampling techniques. SMOTE is an oversampling technique that is used to create synthetic minority datasets. It has been observed that for high dimensional data under sampling outperforms SMOTE, but for low dimensional data SMOTE works just fine. SMOTE works by selecting two of the minority instance points, drawing a straight line between them and then selecting a random value between 0 and 1 and multiplying the distance between those two points by the selected random number. This gives us a new point on that line. The selected point is the new data instance of the minority class. Borderline-SMOTE is an extension of the SMOTE algorithm that only oversamples the data points that have majority of their neighboring instances from the majority class. These data points are considered to be DANGER points. For detecting minority instances, Borderline-SMOTE tends to have better F1 scores than SMOTE and other oversampling techniques [3]. ADASYN is an improved version of SMOTE. It works similarly as SMOTE but instead of making the synthetic data points being linearly correlated to the older data points, it adds variance to these, i.e. the synthetic data points are much more scattered than before.

Random under sampling [4] is an under sampling technique that randomly removes data points from the majority class to make the number of instances in both the classes same. Near Miss[5] is an under sampling technique that removes those data points from the majority class that have the least average distance from the minority class. In other words, those instances of the majority which are closest to the minority class are removed.

XGBoost [6] stands for eXtreme Gradient Boosting. It is based on ensemble classifiers' boosting technique. The XGBoost implements a lot of features to give a fast and scalable classifier. After testing XGBoost, ADABoost and GBM ensemble classifiers [7] on different datasets, XGBoost has proved to be the best ensemble classifier. ADABoost and Voting Classifiers are also able to provide good results in handling imbalanced data. Another important and widely used classifier is the Logistic Regression [8] Classifier. The Logistic Regression Classifier uses sigmoid function to calculate the class values. The output of the classifier is discreet unlike the output of the linear regression models whose output is continuous. The Naive Bayes Classifier [9] is a classification model that works on the basis of Bayes theorem in probability. The classifier works best in the case of independent features. Decision Tree Classifier [10], on combining with oversampling technique, SMOTE is proved to be highly efficient in detecting the class labels of the imbalanced datasets. Random Forest classifiers [11] are collections of decision trees in which the number of classifiers and the number of features can be defined.

3. Proposed Work

The imbalanced data is split into train and test. Individual classifiers and Ensemble models are trained on the imbalanced data. Oversampling and under sampling techniques are used to balance the train data and train the individual classifiers and Ensemble models. Following this, the class labels of the test data is predicted using the classifiers and ensemble models and their results recorded. The proposed architecture is shown in Fig.1. Various classification algorithms, oversampling techniques and ensemble models used in our architecture is discussed as follows.

The SMOTE algorithm generates synthetic data instances of the minority class in order to make the number of minority data instances i.e. the rare events equal to the number of majority data instances. Either this or the ratio can be specified in which the minority and majority instances are required. SMOTE takes the feature vector of two minority class instances, then it takes a random value between 0 and 1, multiplies the difference of feature vectors of the minority instance with the random value and adds it to the feature vector of the instance with lower value instance. By doing this, we get a new instance whose feature vector lies between the previous two minority instances.

The Borderline SMOTE algorithm also generates synthetic data tuples of the minority class in order to increase the number of minority classes. This method is an implementation of SMOTE and only takes the borderline data instances of the minority class. The borderline data instances are the ones that have more number of majority instances as their neighbors' than the minority instances. After getting the danger set, we apply the same steps as in SMOTE [12] to get synthetic minority datasets. Naive Bayes is a powerful algorithm for Predictive Modeling which is based on Bayes Theorem. It is not a single algorithm but a family of algorithms where all of these algorithms share one common working principle and each and every pair of features being classified are independent of each other. Training is fast because only the probability of each class and the probability of each class given different input values need to be calculated. Dataset is divided into two parts which are the feature matrix and response vector, in which the feature matrix contains all the rows of the dataset in which each vector consists of the value of dependent features and response vector contains the output for each row of feature matrix. Each feature makes an equal and independent contribution to the output class label. Bayes theorem is stated mathematically as:

$$P(A|B) = (P(B|A) * P(A))/P(B) \qquad (1)$$

Logistic Regression can be used for binary classification of datasets. The logistic function (or the sigmoid function) is used by this classifier. The function takes input values and gives output as a number between 0 and 1. The output of the function gives an S-shaped curve whose values lie between 0 and 1.

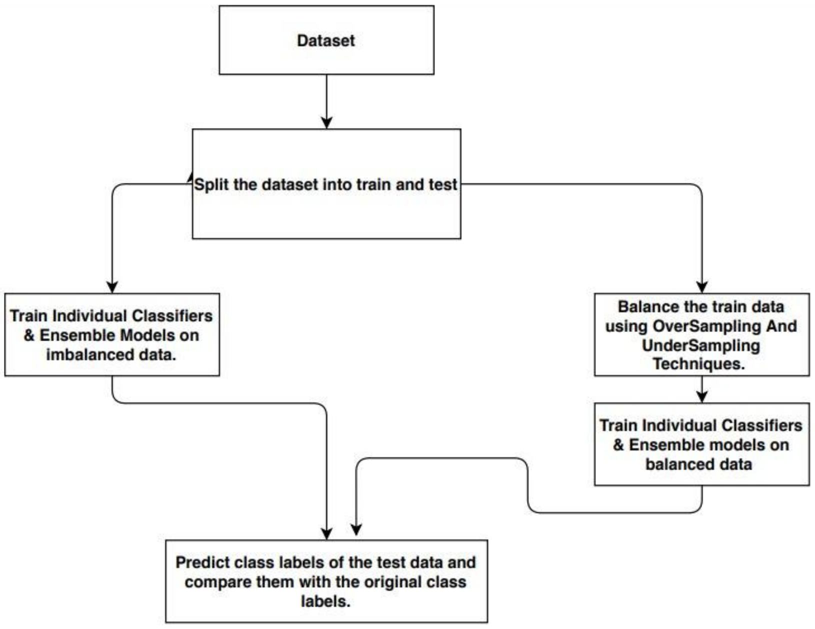
Dataset
Split the dataset into train and test
Train Individual Classifiers & Ensemble Models on imbalanced data.
Balance the train data using OverSampling And UnderSampling Techniques.
Train Individual Classifiers & Ensemble models on balanced data
Predict class labels of the test data and compare them with the original class labels.

Fig. 1: Proposed Architecture

$$Sigmoid(x) = \frac{1}{1-e^{x}} (2)$$

Decision tree classifier is used to classify our data tuples after training our decision tree classifier model with train dataset. The decision tree forms a tree like structure with each node representing a parameter of the given dataset. The decision tree is constructed in a top down manner. The most important feature is chosen as the root node of the tree and the subsequent nodes are features that best split the set of items.

Random Forest is a simple Supervised Machine Learning Technique that constructs multiple decision trees. The final decision is made based on the outcome of the majority decision tree. Here a dataset is classified as row sampling and feature sampling and it is going to be trained in the multiple decision tree and may be some feature sampling and row sampling is common to the other decision tree and all the samples of the dataset are trained with the model and it gives some results in binary and continuous results of regression problems. Whenever we pass some test data in a binary way it gives some result by checking with the entire decision tree. According to Bagging, finally aggregations are done and select the answer with majority votes. Random forest includes flexibility and reduces the high variance to the low variance by combining all decision trees with respect to majority votes. High variance is reduced to the low variance and even in the change of dataset it does not impact more on our result.

XGBoost is also known as sequential ensemble Technique. The main feature of this algorithm is scalability that leads to fast learning through parallel computing and performs efficient memory usage. It creates a sequential decision tree and the first decision tree takes a sample of records and classify as true or false and those records and classified results are known as 1st weak classifier and records classifies as false their weight of that sample is updated and updated records and similar weight records are passed to the next sequential decision tree and this is going to be continued. All the weak classifiers are going to be combined and get the final classifier. It checks the model as classified with 0 or 1 and by seeing the model with maximum number of output are going to be considered by the model. In XGBoost it gives high bias and low variance and by using the decision tree up to some depth it combines multiple weak classifiers and it reduces the high bias into low bias.

In the ADABoost [13] model, the output of our other classification models is combined into a sum after being weighted. This sum represents the final output of the classification model. In ADABoost, the Decision Tree is created with only one depth and for every one feature we create one stump. On seeing entropy and gini index stumps with lesser entropy is going to be selected. Check the wrongly classified sample and calculate the total error and performance of the stump where

$$Performance = \frac{1}{2} * log(\frac{1-total\ error}{total\ error})(3)$$

Then update the weight and our new sample weight.

$$New\ Sample\ Weight = weight * e^{performance} \quad (4)$$

$$Correctly\ Classified\ Sample\ Weight = weight * e^{-performance} \quad (5)$$

After this we get normalized value and updated weight, calculate mean of updated value and by dividing the value with the mean we will get normalized Value. Now, on working with normalized Value, create a new dataset and it selects wrong data and trains the model and divides it into buckets of normalized value and majority votes between all the stumps. We are combining weak learners to make it stronger.

The voting classifier [14] is a type of bagging ensemble model that combines different machine learning algorithms and tests the test data on all the models, then performs voting on the results of all these models and the final output is given as the class that achieves the majority of these votes. The weights of votes of each model are kept the same.

4. Implementation

4.1 Dataset

We have used the credit card fraud dataset as it is highly class imbalanced. Our dataset consists of 2, 84,807 instances of data, each representing one credit card transaction. The dataset has been taken from kaggle. The dataset is already normalized and the dimensionality reduction has already been carried out on the dataset using PCA (Principal Component Analysis) [15].

The dataset is highly imbalanced with only 0.17% of the instances belonging to the positive class i.e. fraud and rest of the 99.83% of the instances belongs from the negative class i.e. genuine transactions as shown in Fig.2. The imbalance ratio of our dataset is 0.0017. The dataset is split in order to get the training set and test set. The dataset is divided into two parts in the ratio 4:1. The train dataset is then used to train all our models and then use the test dataset to predict class labels of the test dataset. The performance metrics Accuracy, F1 score, True Positive Rate and ROC-AUC score are used to check the efficiency of our classifiers on imbalanced data.

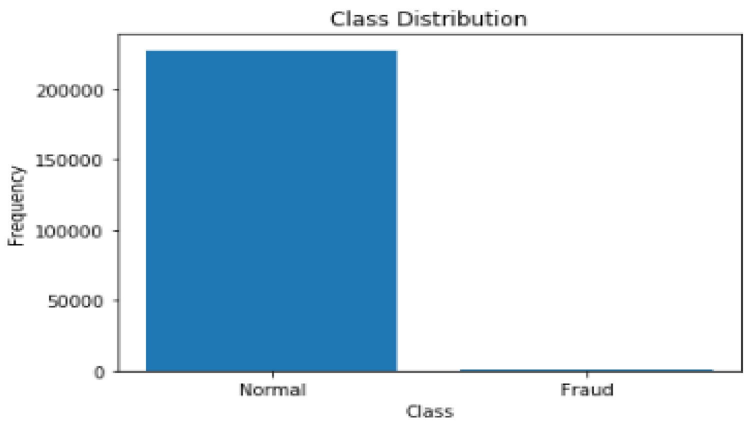

Fig.2: Number of instances in both the classes before oversampling.

The classifiers used are Naive Bayes, Decision Tree, Logistic Regression and Random Forest Classifiers. The train dataset is then oversampled using SMOTE in order to make the number of minority class instances to be equal to that of majority class instances as shown in Fig.3. After balancing the class distribution of our dataset, we use our classifiers to predict the class labels of the test dataset. Following this, Borderline SMOTE and ADASYN are used for oversampling the minority instances and their results are recorded. Under sampling techniques, Near Miss and Random under sampling are also used to balance the train dataset. Under sampling removes data tuples from the majority class and makes the number of majority class instances equal to that of minority class. After applying under sampling on the train dataset, we train our classifiers to finally predict the class labels of our test dataset.

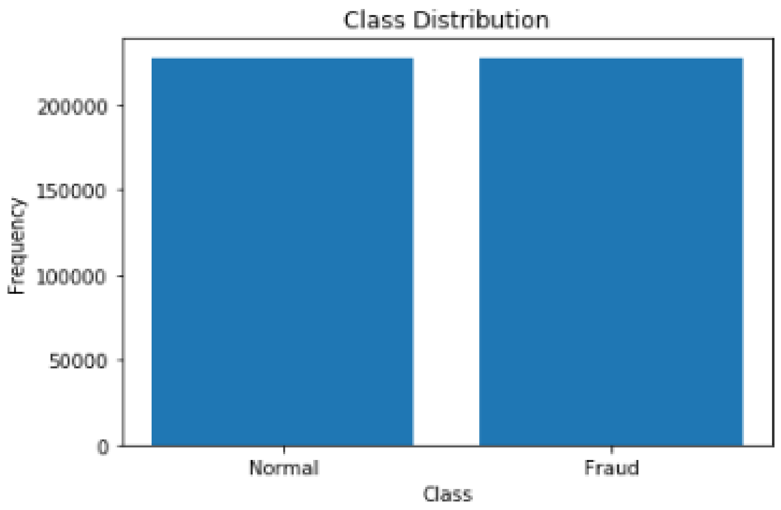

Fig.3: Number of instances in both the classes after oversampling.

4.2 Model Training

The naive bayes algorithm is used to make classification of the test data into positive and negative classes after training the models with original data and oversampled data. We train two Naive Bayes models, one with original train data and the other with the oversampled train data. After training both the models, we get the accuracy of both the models by predicting the class labels of the Xtest dataset. After getting the predicted output labels from our models, we get the accuracy of both the models using confusion matrix. Similarly, Logistic Regression classifier, Decision Tree Classifier and Random Forest Classifier is trained on the original train data and the oversampled train data and then predict the class labels of Xtest data using these models. Along with these we will keep using confusion matrix to get the accuracy of all these models.

After testing the efficiency of the above algorithms individually, we applied ensemble models on our dataset to find out its efficiency in classification. We have applied various ensemble models including XGBoost, ADABoost, and Voting Classifier. These ensemble models tend to give higher accuracy than all the individual models. We performed tests on all the given classification models with different parameters. The ratio of train and test data is set at 4:1. After applying oversampling and under sampling separately, the number of minority instances and the number of majority instances are made equal.

4.2 Performance Metrics

The performance metrics of the classification models are generally the prediction accuracy. But, in case of imbalanced dataset, the model is biased in favor of the majority class, and the model classifies every data tuple as an instance of the majority class, the accuracy of the model will surely be very high but still, it is of no use. The rare events in our imbalanced dataset may even be treated as noise (outliers) or the noise might be misclassified as a rare event of a minority class. This is why, Accuracy cannot be treated as a good performance metric. This is why F1 score, True Positive Rate and ROC-AUC score are used as performance metrics [16]. F1 score is calculated using the precision and recall values of the predicted results from the confusion matrix. Confusion matrix has four values, namely True Positive, True Negative, False Positive and False Negative. Precision value of a classifier is calculated as:

$$Precision = \frac{True\ \ Positive}{True\ \ Positive\ \ + False\ \ Positive} \tag{6}$$

It can also be written as:

$$Precision = \frac{True\ \ Positive}{Total\ \ Predicted\ \ \ as\ Positive} \tag{7}$$

Similarly, the Recall value can be calculated as:

$$Recall = \frac{True\ \ Positive}{True\ \ Positive\ \ + False\ \ Negative} \tag{8}$$

It can also be written as:

$$Recall = \frac{Total\ \ Positive}{Total\ \ Actual\ \ Positive} \tag{9}$$

The F1 score is a balance between both Precision and Recall and is very effective in case of imbalanced datasets. The F1 score can be calculated as:

$$F1\ score = 2 * \frac{Precision\ \ \ * Recall}{Precision\ \ \ + Recall} \tag{10}$$

The True Positive Rate is the same as Recall and is also known as Sensitivity. The TPR is an important performance metric in terms of imbalanced classes. ROC-AUC (Receiver Operator Characteristics - Area under Curve) score is calculated by plotting the ROC curve by using Sensitivity and Specificity. The ROC curve is plotted using Sensitivity on the y-axis and (1-Specificity) on the x-axis. Specificity can be calculated as:

$$Specificity = \frac{True\ \ Negative}{True\ \ Negative\ \ + False\ \ Positive} \tag{11}$$

5. Results and Discussion

The Classifiers such as Naive Bayes, Logistic Regression, Decision Tree and Random Forest are trained on the imbalanced dataset and the highest AUC Score is 0.87 and is obtained by the Decision Tree Classifier. After balancing the dataset by using SMOTE, ADASYN, Borderline SMOTE, Near Miss and Random under sampling separately, and training the aforementioned classifiers, the highest AUC Score is 0.95 which is obtained by Random Forest Classifier when trained using the dataset which is balanced using SMOTE. The ROC-AUC scores of the individual classifiers are shown in Table I and Fig. 4.

The aforementioned classifiers have the highest F1 score of 99.94% which is obtained by Random Forest when trained on imbalanced dataset. After balancing the dataset by using the aforementioned oversampling and under sampling techniques, the Random Forest Classifier obtains the highest F1 score of 99.95% when the dataset is balanced using SMOTE. The F1 scores of the individual classifiers is shown in Table II and Fig. 5.

The aforementioned classifiers have the highest True Positive Rate of 72.16% which is obtained by Random Forest when trained on imbalanced dataset. After balancing the dataset by using the aforementioned oversampling and under sampling techniques, the

Linear Regression Classifier obtains the highest True Positive Rate of 92.68% when the dataset is balanced using SMOTE. The True Positive Rates of the individual classifiers is shown in Table III and Fig.6.

Table 1: ROC-AUC scores of individual classifiers on imbalanced and balanced data

ROC-AUC	Naive Bayes	Logistic Regression	Decision Tree	Random Forest
IMBALANCED	0.81	0.86	0.87	0.78
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM				
SMOTE	0.88	0.91	0.92	0.95
ADASYN	0.86	87	0.88	0.91
BORDERLINE	0.84	0.9	0.9	0.93
NEARMISS	0.77	0.67	0.8	0.87
RANDOM	0.85	0.88	0.92	0.93

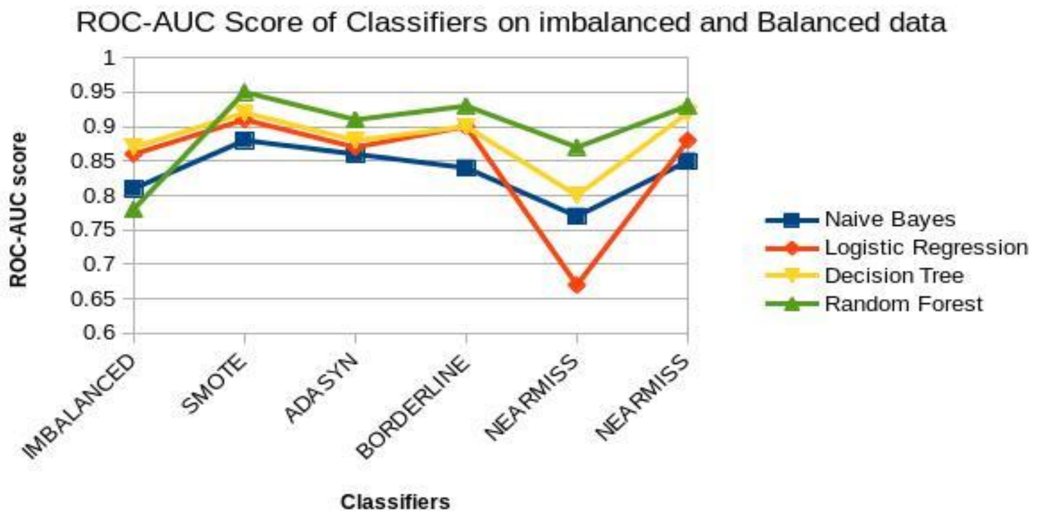

Fig.4: ROC-AUC Score of Classifiers on imbalanced and Balanced data

TABLE II: F1 scores of individual classifiers on imbalanced and balanced data

F1 Score	Naive Bayes	Logistic Regression	Decision Tree	Random Forest
IMBALANCED	99.51	99.92	99.9	99.94
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM				
SMOTE	99.5100	97.41	99.79	99.95

ADASYN	99.4600	97.65	99.95	92.41
BORDERLINE	99.6000	98.7	99.9	99.95
NEARMISS	90.51	76.98	81.72	96.82
RANDOM	99.45	93.98	95.08	98

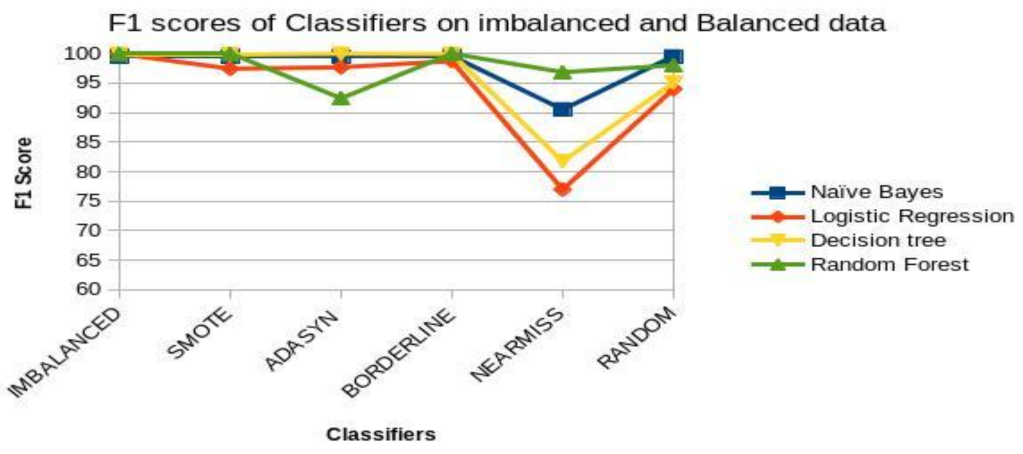

Fig.5: F1 scores of Classifiers on imbalanced and Balanced data

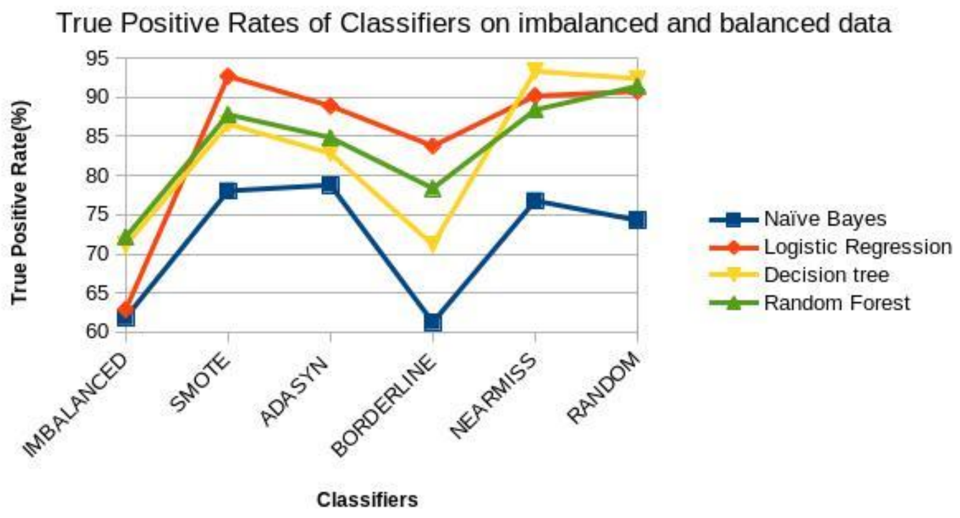

Fig.6: True Positive Rates of Classifiers on imbalanced and balanced data

TABLE III: True Positive Rates of individual classifiers on imbalanced and balanced data

TPR	Naive Bayes	Logistic Regression	Decision tree	Random Forest
IMBALANCED	61.86	62.89	71.13	72.16
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM				

SMOTE	78.05	92.68	86.59	87.8
ADASYN	78.79	88.89	82.83	84.85
BORDERLINE	61.26	83.78	71.17	78.38
NEARMISS	76.79	90.18	93.33	88.39
RANDOM	74.29	90.75	92.38	91.43

The Ensemble models such as XGBoost, ADABoost, and Voting Classifier are trained on the imbalanced dataset and the highest ROC-AUC score is 0.90 and is obtained by the XGBoost Ensemble Model. After balancing the dataset by using the aforementioned oversampling and under sampling techniques, the highest AUC score is 0.95 and is obtained by ADABoost when the dataset is balanced using ADASYN. The ROC-AUC scores of the ensemble classifiers is shown in Table IV and Fig.7.

TABLE IV: ROC-AUC scores of ensemble classifiers on imbalanced and balanced data

AUC	XGBOOST	ADABOOST	VOTING CLASSIFIER
IMBALANCED	0.9	0.87	0.87
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM			
SMOTE	0.91	0.93	0.92
ADASYN	0.92	0.95	0.92
BORDERLINE	0.92	0.93	0.89
Near Miss	0.88	0.78	0.85
Random	0.93	0.93	0.93

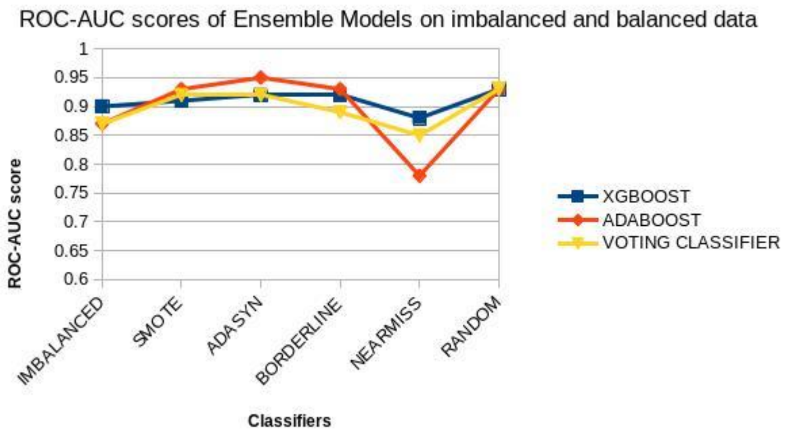

Fig.7: ROC-AUC scores of Ensemble Models on imbalanced and balanced data

The ensemble classifiers have the highest F1 score of 99.95% which is obtained by XGBoost when trained on imbalanced dataset. After balancing the dataset by using the aforementioned oversampling and under sampling techniques, the ADASYN ensemble Classifier obtains the highest F1 score of 99.95% when the dataset is balanced using ADASYN. The F1 scores of the ensemble classifiers is shown in Table V and Fig.8.

The aforementioned ensemble classifiers have the highest True Positive Rate of 79.46% which is obtained by XGBoost when trained on the imbalanced dataset. After balancing the dataset by using the aforementioned oversampling and under sampling techniques, the XGBoost Classifier obtains the highest True Positive Rate of 95.56% when the dataset is balanced using Near Miss. The True Positive Rates of the ensemble classifiers is shown in Table VI and Fig.9.

TABLE V: F1 scores of ensemble classifiers on imbalanced and balanced data

F1 Score	XGBOOST	ADABOOST	VOTING CLASSIFIER
IMBALANCED	99.95	99.93	99.92
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM			
SMOTE	99.94	99.49	99.78
ADASYN	99.95	99.49	99.86
BORDERLINE	99.97	99.86	99.9
Near Miss	88.47	77.22	88.4
Random	98.22	97.16	99.31

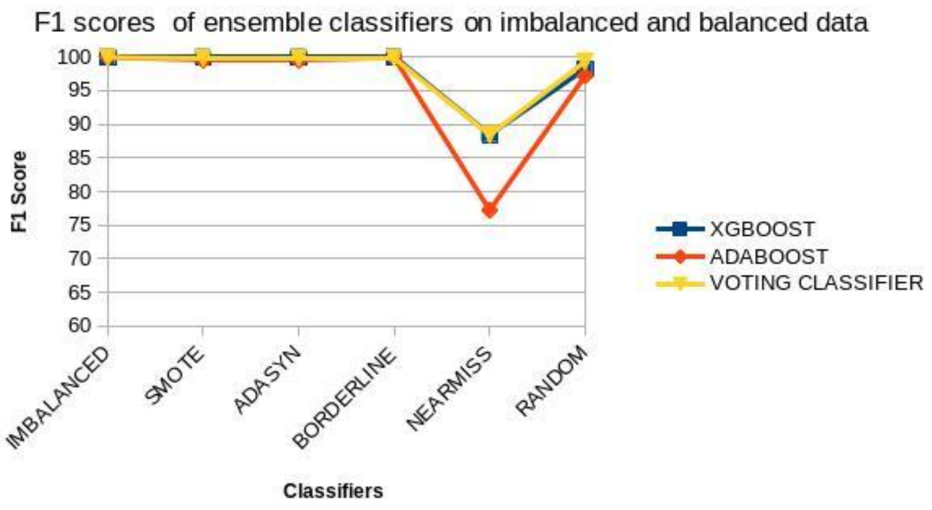

Fig.8: F1 scores of Ensemble Models on imbalanced and balanced data

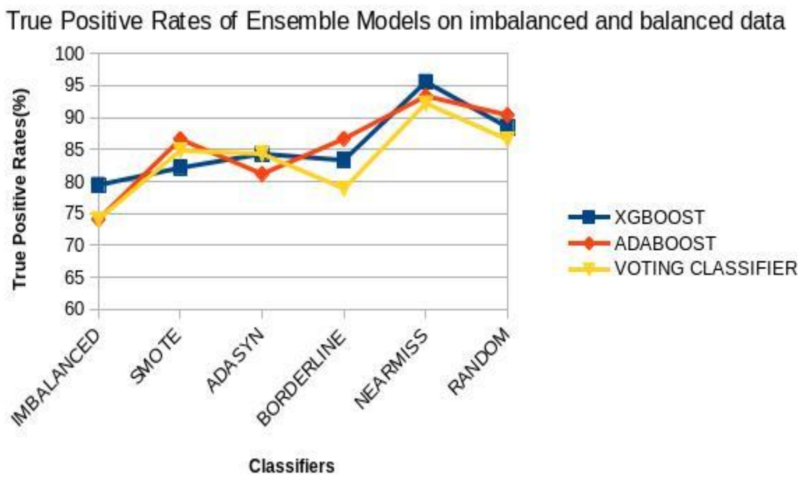

Fig.9: True Positive Rates of Ensemble Models on imbalanced and balanced data

TABLE VI: True Positive Rates of Ensemble Models on imbalanced and balanced data

TPR	XGBOOST	ADABOOST	VOTING CLASSIFIER
IMBALANCED	79.46	74.11	74.11
BALANCED BY OVERSAMPLING AND UNDER SAMPLING ALGORITHM			
SMOTE	82.14	86.61	84.82
ADASYN	84.31	81.18	84.31
BORDERLINE	83.33	86.67	78.89
Near Miss	95.56	93.33	92.22
Random	88.46	90.38	86.54

6. Conclusion

By looking at the results of all the algorithms that are used, it is seen that by removing the class imbalance and making the dataset balanced, the accuracy of our models have increased significantly. The True Positive rate of Naive Bayes classifier goes up by around 17%. The logistic regression classifier has shown an increase of about 28% in True Positive Rate. In the Decision Tree classifier model, the True Positive Rate increases by 17%. The Random Forest Classifier performs very well in classifying as it is also based on the ensemble model. The True Positive Rate of the Random Forest Classifier goes up by 20% and remains high at 91.43%. The ensemble classifiers XGBoost, ADABoost and Voting Classifiers show the best performance results in terms of ROC-AUC Score, F1 score and True Positive Rates. It has been seen that balancing the data using oversampling techniques helps the classifiers achieve better classification results than under sampling techniques. Also, the Ensemble models have outperformed the individual classifiers in most cases by providing better classification results.

REFERENCES

[1] Haixiang, Guo, et al. "Learning from class-imbalanced data: Review of methods and applications." Expert Systems with Applications 73 (2017): 220-239.

[2] Blagus, R., Lusa, L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 14, 106 (2013).

[3] Han H., Wang WY., Mao BH. (2005) Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang DS., Zhang XP., Huang GB. (eds) Advances in Intelligent Computing. ICIC 2005. Lecture Notes in Computer Science, vol 3644. Springer, Berlin, Heidelberg.

[4] Mishra, Satwik, “Handling Imbalanced data: SMOTE vs. random undersampling.”International Research of Engineering and Technology(IRJET)(2017).

[5] Yen, Show-Jane, and Yue-Shi Lee. "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset." Intelligent Control and Automation. Springer, Berlin, Heidelberg, 2006. 731-740.

[6] T. Chen, T. He, M. Benesty, et al., Xgboost: extreme gradient boosting, R package version 0.4-2 (2015).

[7] Gómez-Ríos, Anabel, Julián Luengo, and Francisco Herrera. "A study on the noise label influence in boosting algorithms: AdaBoost, GBM and XGBoost." International Conference on Hybrid Artificial Intelligence Systems. Springer, Cham, 2017.

[8] Dreiseitl, Stephan, and Lucila Ohno-Machado. "Logistic regression and artificial neural network classification models: a methodology review." Journal of biomedical informatics 35.5-6 (2002): 352-359.

[9] Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. 2001.

[10] Safavian, S. Rasoul, and David Landgrebe. "A survey of decision tree classifier methodology." IEEE transactions on systems, man, and cybernetics 21.3 (1991): 660-674.

[11] Khoshgoftaar, Taghi M., Moiz Golawala, and Jason Van Hulse. "An empirical study of learning from imbalanced data using random forest." 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007). Vol. 2. IEEE, 2007.

[12] Ganganwar, Vaishali. "An overview of classification algorithms for imbalanced datasets." International Journal of Emerging Technology and Advanced Engineering 2.4 (2012): 42-47.

[13] Schapire, Robert E. "Explaining adaboost." Empirical inference. Springer, Berlin, Heidelberg, 2013. 37-52.

[14] Bauer, Eric, and Ron Kohavi. "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants." Machine learning 36.1-2 (1999): 105-139.

[15] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2.1-3 (1987): 37-52.

[16] Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).