Proceedings of the Third International Conference on Smart Systems and Inventive Technology (ICSSIT 2020)
IEEE Xplore Part Number: CFP20P17-ART; ISBN: 978-1-7281-5821-1

Influence of Optimizing XGBoost to handle Class Imbalance in Credit Card Fraud Detection

Dr. C. Victoria Priscilla
Associate Professor and Head
PG Department of Computer Science
S.D.N.B. Vaishnav College for Women
University of Madras
Chennai, India
aprofvictoria@gmail.com

D. Padma Prabha
Assistant Professor
Department of Computer Applications
Madras Christian College
University of Madras
Chennai, India
padmaprabha@mcc.edu.in

***Abstract*— XGBoost is one of the popular machine learning models used in the domains like fraud detection as well as to tackle the class imbalance that creates overfitting if not handled properly. Digital transactions are encouraged by financial institutions to maintain data integrity. Credit card payment is one of the prevalent means of transactions carried out in both online and offline purchases. Consequently, the risk of fraudulent activities are increased during these financial transactions. This creates awareness among the researchers about the need for an efficient method to detect fraud accomplishments. This paper proposes an optimized XGBoost (OXGBoost) approach to handle class imbalance in the datasets without using resampling techniques. In this proposed approach, RandomizedSearchCV hyperparameter optimization technique is applied to find the optimal parameters of XGBoost. The data sampling techniques are integrated with XGBoost to increase the efficiency of the model. The experiment was performed based on the two real-world credit card datasets. The findings of the experiment proved that the integration of data sampling does not have an impact on the efficiency of XGBoost. Based on the comparison, the proposed approach has outperformed the higher accuracy.**

Keywords—Credit card, fraud detection, class imbalance, optimization, XGBoost, hyperparameter, parameter tuning.

I. INTRODUCTION

E-commerce has taken its position in the global market, hence electronic payments are initiated by the financial institution to their customers during purchase transactions. The popular payment methods like credit or debit cards, online transactions, e-wallet, etc. are used. This creates an alarm of money laundering in the financial system during online transactions. The Nilson report is one of the leading newsletters conveying the global statistics about the payment industry [1] published that the worldwide fraud loss by 2023 is expected to reach $35.67 billion and in 2027 it is predicted to reach $71.593 trillion (Nilson report, issue-1146, Pg.8). Machine learning algorithms for credit card fraud detection (CCFD) that helps the financial community to identify the fraud transaction but the important challenge in identifying these fraudulent transactions is the existence of class imbalance in the data pattern.

Class imbalance is the disproportion of positive and negative class in the dataset. Many real-world application domains such as fraud detection, detection of oil spills in radar images, telecommunication, credit loan payment default, tax payment, medical diagnostics, churn prediction, customer retention, etc. have recognized class imbalance problem [2]. Due to this irregular distribution of data, the learning algorithms cannot perform well on the data as they are biased towards the majority class. The skewed distribution of data emphasis on majority class during the training process may lead to classify high false negative rates. Data sampling methods redistribute the data evenly to improve the efficiency of minority class considered as most important as majority class. This can be handled by oversampling the minority class by generating synthetic instances during training phase or resampling the dataset before applying towards the model to produce better performance for any imbalanced data [3]–[7]. Previous studies on CCFD suggest ensemble methods to attain higher accuracy in the detection of fraud samples.

Ensemble learning [8] is a supervised learning coined by Nilsson for classification of data. The idea behind this method is to train the base learners and combining the predicted outcome to a single result. Bagging and Boosting are widely used ensemble techniques. *Bagging (Bootstrap Aggregation*) is a meta algorithm coined by Breiman that creates multiple base models separately and resampling is done with different training data taken from the original training data by row sampling with replacement and finally, the results are integrated by majority voting [9]. To avoid overfitting central limit theorem is used in [10] to average all the decision trees created. Boosting is a combination of weak learners that are created sequentially to produce accurate results. Each model is trained iteratively and the performance of the previous model influence the new model to correct the incorrect instances created by the previous learner. After each iteration, the learners are weighted based on their performance and finally, the boosting weight predicts the model's performance.

The XGBoost (Extreme Gradient Boosting) is a popular winning algorithm in the Kaggle [11] platform for both classification and regression problems. Chen and Guestrin developed an ensemble tree boosted sparsity-aware algorithm

[12] to handle sparse data. Overfitting of data is prevented through regularizing the objective. It is designed for parallel processing which is much faster compared with GBM. The missing values are handled by itself in each node. XgBoost split the nodes till the maximum depth then tree pruning is done backward to remove the splits when negative loss is encountered. The learning rate of the model is increased once applied as parallel processing. The base of this model was developed from Gradient Tree Boosting [13] or Gradient Boosting Machine(GBM); a tree boosting technique to reduce the loss function using additive learning of the weak learners. From GBM, Ke *et al.* developed an algorithm named Light GBM [14] to obtain information gain with small dataset and Prokhorenkova *et al.* formulated Catboost [15] algorithm to solve the target leakage issue existed in GB algorithms. Even though LightGBM and Catboost proved its efficiency to solve the existing drawbacks found in GBM, XGBoost outperforms with significant features. [16] applied imbalance XGBoost with integrated weighted and focal loss to handle class imbalance. Classification through optimization identifies the optimal parameters to fit the algorithm for greater accuracy by different optimization techniques [16]–[19].

This paper intends the influence of optimization in the XGBoost classifier using hyperparameter tuning to enhance the classification rate between the binary targets in an extremely imbalanced dataset. The goal of the experiment is to handle imbalance without combining any resampling techniques to the proposed model. Two real-world credit card fraud dataset with high imbalance ratio was experimented and identified the exemplary performance of optimized XGBoost model. The rest of the paper is structured as Section II reviews the related studies conducted by the researchers, Section III describes the mathematical derivatives of XGBoost, Section IV present the experimental analysis which comprises three subdivisions, Section V discusses the results obtained from the experiment and section VI concludes with future insights.

II. Related Work

The imbalance in class can be handled by four common methodologies namely data level, algorithmic level, cost-sensitive learning, and Ensemble learning [20]. The most popularly used approach to handle class imbalance is the resampling of majority and minority classes. To maximize the overall performance conventional *data level* sampling techniques are used to redistribute the class evenly through oversampling, under sampling, and combined sampling [21].

Oversampling is randomly generating new instances of minority class by reproducing the existing minority samples that could create overfitting [22]. This problem was handled by Chawla et al. proposing SMOTE [23] to synthetically generate minority instances that are randomly selected by k-nearest neighbors. He et al. introduced ADASYN [24] to modify the distribution of data and minimize the bias by assigning weights to create a synthetic minority class. [25] proposed KMFOS a cluster based model with noise filtering, here the clusters of minority instances are selected and new minority instances are generated. Later [26] combined a metaheuristic SMOTE with Firefly that sets data bounds of minority instances by selecting the amount of minority class then oversampled randomly by selected data points. [27] designed a framework to deal with noise present in the data by noise filtering and oversampling. The abstention-SMOTE algorithm created by [28] to find abstention in minority instances and synthetically generating the minority samples to redistribute the data. [22] presented an oversampling technique to increase the minority class by combining k-means clustering and SMOTE. [18] experimented by integrating DBSCAN and SMOTE to produce DSMOTE by clustering minority instance into three groups and finally remove noise from minority samples to generate more meaningful data.

Undersampling resamples the data by reducing the majority class randomly can cause missing the most significant information needed for classification. Thi Thu *et al.* constructed a nearest neighbor algorithm to find the nearest positive class called MASI [4]. A small proportion of majority samples are under sampled by [29] investigated through KNN approach to make the data less skewed with three different versions of NearMiss. [30] Cleaned the overlap between the two classes and majority samples are minimized by finding the distance of instance pairs using nearest neighbor algorithm. A new approach was introduced to combine oversampling and under sampling to increase accuracy. Batista *et al.* experimented with SMOTEENN and SMOTETomek to balance the class distribution [31]. Many more methods have been explored by the researchers in this category.

The *algorithmic level* approaches from previous studies are, [32] proposed entropy-based fuzzy SVM (EFSVM) by allocating fuzzy membership for minority samples giving more importance to the minority class. To filter the majority samples clustering tree algorithm was constructed by [33] to analyze the feature of data and then to construct the tree. The incorporation of Biased Support Vector Machine (BSVM) and Weighted-SMOTE proposed by [34] to balance the variance between majority and minority samples. According to [7], [19] *cost-sensitive learning* performs well by increasing misclassified cost weight for minority samples decreasing the weights for majority samples to solve the class imbalance, but it is difficult to assign the exact misclassification cost. To enhance efficiency Yang *et al.* suggested to select informative sample based on probability using density based under sampling and done a cost sensitive learning method adjusting the weights of the samples [6]. To escalate the efficiency of huge dataset [35] combined cost sensitive decision tree was proposed with two adaptive techniques by selecting adaptive cut points and by removing redundant attributes. Besides SVM, Artificial Neural Network and K-Nearest Neighbor are popularly used cost sensitive methods by easily adjusting the misclassification cost [20].

Ensemble learning has proved as the best technique for class prediction of any real-world binary classification problem, where the credit card frauds are identified from a huge volume of transactional data as the fraudulent transactions are very

less. Various studies were done by integrating ensemble techniques with traditional sampling techniques to improve performance. Raghuwanshi and Shukla, worked on kernelized ELM [36] to estimate the underbagging ensemble technique by substituting weights for the training data. [3] implemented the self-paced ensemble (SPE) by harmonizing hardness through under sampling. A novel approach ROSE a bootstrap method proposed by Lunardon *et al.* to generate new artificial samples that are not identified earlier [37]. Ensemble algorithm RotEasy was experimented in [30] to reduce the memory storage and to increase the execution speed using ensemble pruning. Researchers have combined boosting algorithms with data sampling techniques to increase the accuracy rate [7], [21], [38], [39]. Combination of boosting model with data sampling increase the computation time as boosting models have regularization parameters to avoid overfitting.

III. Mathematical Derivative of XGBoost

This section explains the mathematical modeling of XGBoost derivation introduced by [12]. XGBoost model is built by additive tree boosting method with two derivatives namely gradient g_i and hessian h_i independently creates the boosting tree to handle class imbalance [16]. Initially, the objective function O measures the fitness of the model for training data obtained by the combination of training loss L and regularization term Ω [40] then $O(\theta) = L(\theta) + \Omega(\theta)$, the loss function L measured by mean squared error is $L(\theta) = \sum_i (y_i - \hat{y}_i)^2$ and θ represents the parameters. According to Chen and Guestrin, the regularization objective equation for training features x_i and target y_i, the tree ensemble with K number of trees is given as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{1}$$

Where f is the functional space and $\mathcal{F}$ is the set of possible classification and regression trees(CART). The optimized regularized objective equation is

$$O(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2}$$

now consider the additive training for tree boosting, the optimized objective function is defined as

$$O = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i) \tag{3}$$

the f_i function has the structure and leaf scores of the tree, the difference between the predicted $\hat{y}_i$ and actual y_i is computed by the convex loss function l and additive training t for i iterations learns and create a new tree each time, then the prediction value $\hat{y}_i$ for each t is written as

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{4}$$

Now assume $\hat{y}_i^{(t)}$ as the prediction of i^{th} instance for t iteration then minimize the objective function by greedily adding f_t in (2)

$$O^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{5}$$

Mean squared error loss function applied with first-order expansion, then the objective function is generalized as

$$O^{(t)} = \sum_{i=1}^{n} \left(y_i - \left(\hat{y}_i^{(t-1)} + f_t(x_i)\right)\right)^2 + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} \left[2\left(\hat{y}_i^{(t-1)} - y_i\right) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) \tag{6}$$

the second-order Taylor expansion for the loss function is given by

$$O^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{7}$$

Where $g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$, after the removal of constant terms, the final objective equation for a new tree is

$$O^{\sim(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega\left(f_t\right) \tag{8}$$

the objective equation is re-formulated with $I_j = \{i | q(x_i) = j\}$ is the set of indices to the j^{th} leaf node and expanding the complexity of the tree $\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ in (8) where w is the vector scores of leaf, q is the function assigned to each data of the corresponding leaf and T the number of leaves. Hence the structure score is

$$O^{\sim(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \tag{9}$$

given $q(x)$ after reducing the (9) the best objective function with weight w_j^* of j^{th} leaf for a good tree structure is

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \tag{10}$$

the scoring function to measure a good tree structure q is

$$O^{\sim(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{11}$$

A greedy algorithm is applied to start from a single leaf to split into two leaves and the score evaluated can be used to split candidates. I_L and I_R are the left and right nodes after split. Hence the loss reduction after a split is

$$O_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_j} h_R + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] + \gamma \tag{12}$$

IV. Experimental Analysis

The experimental setup of the proposed OXGBoost for CCFD is explained in Fig 1. The historical dataset is pre-processed and this huge dataset is distributed in the ratio of 70% for the training phase and 30% for the testing phase. The XGBoost alternately integrated with the data sampling methods SMOTE, ADASYN, TomekLink, NearMiss, SMOTENN, SMOTETomek, and then the proposed OXGBoost is also trained and tested. The results were compared using performance metrics and finally proposed OXGBoost with good accuracy is identified. In this experiment, the threshold value is set to 0.5. The parameter setting section explains how the optimal parameters are identified for the experiment.

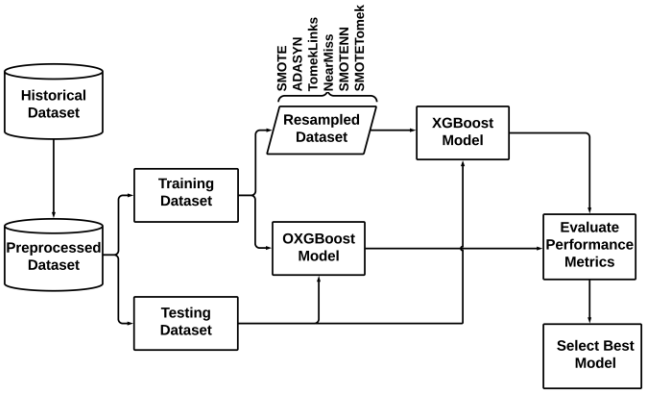


Fig. 1. Process flow of the proposed approach for CCFD.

A. Dataset Specification

Data preprocessing transforms the real-world data that are inconsistent and incomplete to meaningful processed data by cleaning and transformation techniques. For every machine learning algorithm, the quality of data is important as this may reduce the performance of the classifier. The irrelevant and redundant features are discarded to increase the efficiency and decrease the training time of the model. This research work has used two real-world datasets available at Kaggle, where dataset 1 [41] an online transactional data provided by VESTA an e-commerce payment service company to facilitate the machine learning programmers to overcome the greatest loss by frauds happening around the word. This dataset was published by IEEE-CIS. The data is split into two components as identity and transaction which was then combined by the index column named TransactionID. Dataset 2 [42] a European credit card transactional data, except the features 'Time' and 'Amount' all the other features have been PCA transformed with no identity. The quantitative information about the datasets is given in Table I.

TABLE I. Summary of the Experimental Dataset

Dataset	Total No. of Samples	No. of Attributes	Majority Samples	Minority Samples	Imbalance Ratio[a]
Dataset 1	590540	435	569877	20663	27.58
Dataset 2	284807	31	284315	492	577.88

[a] IR= Number of Majority samples/ Number of Minority samples

B. Parameter Setting

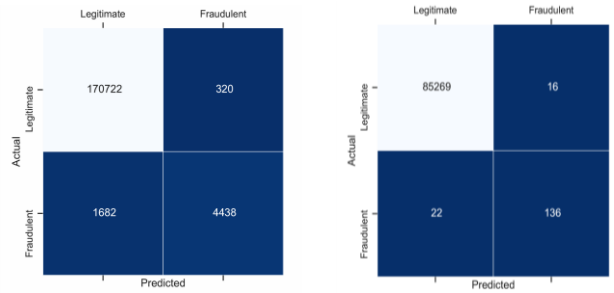
XGBoost has a collection of parameters that are categorized into the general, booster, learning task, and command line [40]. To attain optimal solution hyper-parameter tuning is done to extract the optimal parameters using RandomizedSearchCV available in 'scikit-learn' python library. The parameters are optimized using 5 fold cross-validation technique and the number of iteration performed is 5 for the two different datasets. The boosting parameters *learning_rate* controls the weight of newly added trees, *gamma* returns the minimum loss to create portions of the leaf node, *max_depth* does not have a limit, if it increases there is a chance of overfitting, *min_child_weight* is the minimum sum of sample weight, if the value assigned is larger the algorithm performs better. *scale_pos_weight* parameter handles the imbalance between majority and minority samples. The parameters are optimized and analyzed using two evaluation metric AUC and logloss. Table II lists the tuned parameter values applied to optimize the classifier for the two datasets.

TABLE II. OPTIMAL PARAMETERS OBTAINED FROM HYPER-PARAMETER TUNING

Parameter	Dataset 1	Dataset 2
n_estimators	400	400
min_child_weight	1	3
max_depth	12	15
learning_rate	0.15	0.3
gamma	0.0	0.1
colsample_bytree	0.7	0.4

C. Model Evaluation using Performance Metrics

The skewness of the data is inclined over the majority class, therefore the accuracy of the model archives higher concentration towards the majority class. Yet accuracy being the common evaluation metric this misleads to improper information for an extreme imbalance in binary classification until it is balanced. Low precision and recall indicate the poor performance of the classifier. The performance of the classifier is estimated by the evaluation metrics obtained from two class confusion matrix given in Table III. Here negative indicates legitimate class and positive indicates fraudulent class. The performance metrics balanced accuracy, G-mean, f-measure, Precision, recall, and AUC are attained from the outcome of True Positive (t_p), False Positive(f_p), True Negative(t_n), and False Negative(f_n) from the confusion matrix. The pictorial confusion matrix of the proposed model for the two datasets is given in Fig. 2.



(a) (b)

Fig. 2.Confusion matrix of the proposed model (a) Dataset1 (b) Dataset2

TABLE III. CONFUSION MATRIX

		Predicted	
		Legitimate(0)	***Fraudulent(1)***
Actual	***Legitimate(0)***	True Negatives(t_n)	False Positive (f_p)
	Fraudulant(1)	False Negatives (f_n)	True Positive (t_p)

t_p : Number of samples that are predicted as fraudulent that are truly fraudulent.

f_p : Number of samples that are predicted as fraudulent that are truly legitimate.

t_n : Number of samples that are predicted as legitimate that are truly legitimate.

f_n : Number of samples that are predicted as legitimate that are truly fraudulent.

The following evaluations define the metrics adopted to measure the performance of the classifier.

True Negative Rate (TNR) is the percentage of actual negatives predicted as True negatives from the overall negative values.

$$\frac{t_n}{t_n + f_p} \tag{13}$$

Precision - Attaining high precision indicates the minimal number of false positives which means predicting legitimate as fraudulent.

$$\frac{t_p}{t_p + f_p} \tag{14}$$

Recall or TPR - Attaining high recall indicates the low number of false negatives which means predicting fraudulent as legitimate.

$$\frac{t_p}{t_p + f_n} \tag{15}$$

Balanced Accuracy is used when the dataset is imbalanced. It is the average of sensitivity (TPR) and specificity (TNR).

$$\frac{1}{2}\,(TPR + TNR) \tag{16}$$

G-mean - As the minority samples are ignored by the accuracy score, G-mean can be a good accuracy score for binary classification which measures the accuracy discretely for each class.

$$\sqrt{TPR \times TNR} \tag{17}$$

V. RESULTS AND DISCUSSION

The effectiveness of the proposed work has shown good accuracy by optimizing the parameters of XGBoost using RandomizedSearchCV method for two real-world credit card fraud datasets. Moreover, experiment was conducted to find the progress in efficiency of XGBoost when combined with data level approach to handle the class imbalance in data. As the combinational approach doesn't express the desired accuracy, tuning the default parameter values of XGboost builds the model more accurately and efficiently. Table IV. summarizes the outcome of the proposed model with other combinational methods based on its performance.

Fig. 3, indicates the ROC-AUC curve analysis representing the model performance in the classification of data between the true positive and false positive rate, the AUC values of OXGBoost is interpreted as 0.966 and 0.981 respectively, higher than XGBoost for both the datasets. Hence the performance of the proposed model achieves good predictions between the fraudulent and legitimate samples.

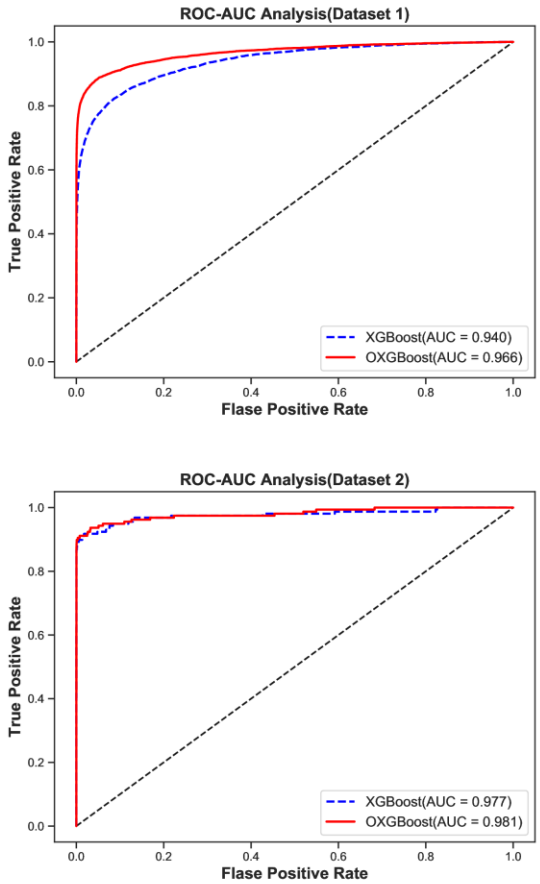


Fig. 3. ROC analysis curve for XGBoost and OXGBoost for the two different datasets.

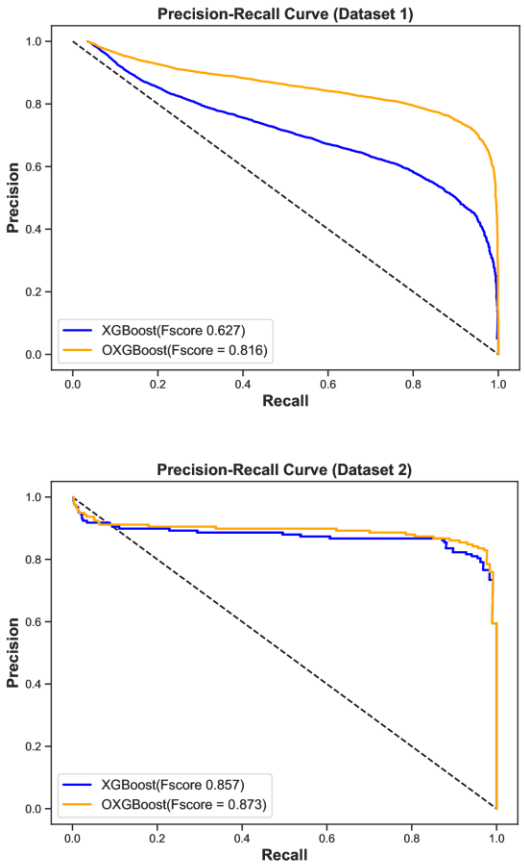


Fig. 4. Precision-Recall analysis curve for XGBoost and OXGBoost for the two different datasets.

Fig. 4, visualizes the precision-recall curve of the proposed OXGBoost compared with XGBoost. Both the datasets performed well with the proposed approach. The result obtained by the f-score shows the average between precision and recall. Therefore, optimization improves the test accuracy of the proposed model.

TABLE IV. PERFORMANCE OF OXGBOOST AND XGBOOST COMBINED WITH SAMPLING ALGORITHMS.

	Dataset 1		Dataset 2	
Classifier	*G-Mean*	*BCR*	*G-Mean*	*BCR*
XGB	0.7760	0.6978	0.9139	0.9176
XGB+SMOTE	0.7873	0.8098	0.9310	0.9333
XGB+ADASYN	0.7784	0.8030	0.9311	0.9365
XGB+Tomeklink	0.8019	0.7768	0.9139	0.9176
XGB+Nearmiss	0.7526	0.7679	0.7084	0.7448
XGB+SMOTENN	0.7980	0.8179	0.9377	**0.9396**
XGB+SMOTETomek	0.7809	0.8048	0.9311	0.9365
OXGB	**0.8444**	**0.8569**	**0.9344**	**0.9335**

From the results, it is observed that there is no significant improvement in the performance, when XGBoost is integrated with sampling techniques. As XGBoost algorithms are based on tree boosting this avoids overfitting of imbalanced data. It was observed that optimizing strategy increases the performance of XGBoost and handles the class imbalance in skewed datasets.

VI. CONCLUSION

It is important to identify fraud occurrences in credit card transactions. There is a need in developing more efficient methods for CCFD to reduce financial loss. This paper presented the influence of optimization in the XGBoost model for handling class imbalance in the dataset effectively by itself. The best parameters are identified using RandomizedSearchCV method available in the scikit-learn package in python. The mathematical derivative of XGBoost is discussed and the experiment was conducted with two real-world imbalanced datasets by integrating different sampling methods. Our findings proved that the proposed OXGBoost can achieve higher accuracy for extremely imbalanced data without sampling. In future, different optimization techniques can be experimented and identified.

REFERENCES

[1] Nilson Report, "The Nilson Report Newsletter Archive," The Nilson Report, 2019. https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1164.

[2] H. Ye, L. Xiang, and Y. Gan, "Detecting Financial Statement Fraud Using Random Forest with SMOTE," IOP Conf. Ser. Mater. Sci. Eng., vol. 612, no. 5, 2019.

[3] Z. Liu et al., "Self-paced ensemble for highly imbalanced massive data classification," Proc. - Int. Conf. Data Eng., vol. 2020-April, pp. 841–852, 2020.

[4] T. N. Thi Thu, L. N. Thi, N. T. Thuy, T. N. Thi, and N. C. Trung, "Improving Classification by Using MASI Algorithm for Resampling Imbalanced Dataset," Int. J. Intell. Syst. Appl., vol. 11, no. 10, pp. 33–41, 2019.

[5] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," Inf. Sci. (Ny)., vol. 513, pp. 429–441, 2020.

[6] K. Yang et al., "Hybrid Classifier Ensemble for Imbalanced Data," IEEE Trans. Neural Networks Learn. Syst., vol. 31, no. 4, pp. 1387–1400, 2020.

[7] F. Rayhan et al., "MEBoost: Mixing estimators with boosting for imbalanced data classification," Int. Conf. Software, Knowl. Information, Ind. Manag. Appl. Ski., vol. 2017-Decem, pp. 1–6, 2018.

[8] Nilsson Nils J., Learning machines; foundations of trainable pattern-classifying systems / Nils J. Nilsson. McGraw-Hill Book Company New York.

[9] L. Breiman, "Bias, variance, and arcing classifiers," 1996.

[10] S. Simske, Meta-analytics: consensus approaches and system patterns for data analysis. Morgan Kaufmann, 2019.

[11] "Kaggle Competitions." [Online]. Available: https://www.kaggle.com/competitions.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[13] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Ann. Stat., pp. 1189–1232, 2001.

[14] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," in Advances in neural information processing systems, pp. 3146–3154, 2017.

[15] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in Advances in neural information processing systems, pp. 6638–6648, 2018.

[16] C. Wang, C. Deng, and S. Wang, "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost," Pattern Recognit. Lett., vol. 136, pp. 190–197, 2020.

[17] A. A. Taha and S. J. Malebary, "An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine," IEEE Access, vol. 8, no. February, pp. 25579–25587, 2020.

[18] X. Xu, W. Chen, and Y. Sun, "Over-sampling algorithm for imbalanced data classification," J. Syst. Eng. Electron., vol. 30, no. 6, pp. 1182–1191, 2019.

[19] H. He, W. Zhang, and S. Zhang, "A novel ensemble method for credit scoring: Adaption of different imbalance ratios," Expert Syst. Appl., vol. 98, pp. 105–117, 2018.

[20] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, Learning from Imbalanced Data Sets. 2018.

[21] C. Zhang and X. Zhang, "An effective sampling strategy for ensemble learning with imbalanced data," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10363 LNAI, pp. 377–388, 2017.

[22] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," Inf. Sci. (Ny)., vol. 465, pp. 1–20, 2018.

[23] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[24] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," Proc. Int. Jt. Conf. Neural Networks, no. July 2008, pp. 1322–1328, 2008.

[25] L. Gong, S. Jiang, and L. Jiang, "Tackling Class Imbalance Problem in Software Defect Prediction through Cluster-Based Over-Sampling with Filtering," IEEE Access, vol. 7, pp. 145725–145737, 2019.

[26] P. Kaur and A. Gosain, "FF-SMOTE: A Metaheuristic Approach to Combat Class Imbalance in Binary Classification," Appl. Artif. Intell., vol. 33, no. 5, pp. 420–439, 2019.

[27] G. Rekha, A. K. Tyagi, and V. Krishna Reddy, "A novel approach to solve class imbalance problem using noise filter method," Adv. Intell. Syst. Comput., vol. 940, pp. 486–496, 2020.

[28] C. Zhang, Y. Chen, X. Liu, and X. Zhao, "Abstention-SMOTE: An over-sampling approach for imbalanced data classification," ACM Int. Conf. Proceeding Ser., pp. 17–21, 2017.

[29] I. Mani and I. Zhang, "kNN approach to unbalanced data distributions: a case study involving information extraction," in Proceedings of workshop on learning from imbalanced datasets, vol. 126, 2003.

[30] Q. Y. Yin, J. S. Zhang, C. X. Zhang, and N. N. Ji, "A novel selective ensemble algorithm for imbalanced data classification based on exploratory undersampling," Math. Probl. Eng., vol. 2014, no. ii, 2014.

[31] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explor. Newsl., vol. 6, no. 1, pp. 20–29, 2004.

[32] D. Gupta, B. Richhariya, and P. Borah, "A fuzzy twin support vector machine based on information entropy for class imbalance learning," Neural Comput. Appl., vol. 31, no. 11, pp. 7153–7164, 2019.

[33] Y. Zhang, G. Liu, L. Zheng, and C. Yan, "A hierarchical clustering strategy of processing class imbalance and its application in fraud detection," Proc. - 21st IEEE Int. Conf. High Perform. Comput. Commun. 17th IEEE Int. Conf. Smart City 5th IEEE Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2019, pp. 1810–1816, 2019.

[34] Hartono, O. S. Sitompul, Tulus, and E. B. Nababan, "Biased support vector machine and weighted-SMOTE in handling class imbalance problem," Int. J. Adv. Intell. Informatics, vol. 4, no. 1, pp. 21–27, 2018.

[35] X. Li, H. Zhao, and W. Zhu, "A cost sensitive decision tree algorithm with two adaptive mechanisms," Knowledge-Based Syst., vol. 88, pp. 24–33, 2015.

[36] B. S. Raghuwanshi and S. Shukla, "Class imbalance learning using UnderBagging based kernelized extreme learning machine," Neurocomputing, vol. 329, pp. 172–187, 2019.

[37] N. Lunardon, G. Menardi, and N. Torelli, "ROSE: A package for binary imbalanced learning," R J., vol. 6, no. 1, pp. 79–89, 2014.

[38] Z. Wu, W. Lin, and Y. Ji, "An Integrated Ensemble Learning Model for Imbalanced Fault Diagnostics and Prognostics," IEEE Access, vol. 6, pp. 8394–8402, 2018.

[39] J. Gong and H. Kim, "RHSBoost: Improving classification performance in imbalance data," Comput. Stat. Data Anal., vol. 111, pp. 1–13, 2017.

[40] "Introduction to Boosted Trees — xgboost 1.2.0-SNAPSHOT documentation." https://xgboost.readthedocs.io/en/latest/tutorials/model.html.

[41] "IEEE-CIS Fraud Detection | Kaggle." https://www.kaggle.com/c/ieee-fraud-detection/data.

[42] "Credit Card Fraud Detection | Kaggle." https://www.kaggle.com/mlg-ulb/creditcardfraud.