Weighted Cross-Entropy to tackle Overlapping in Fraud Detection

Claire Verdier, Stephane Perriot, Arnault Pachot

Weighted Cross-Entropy to tackle Overlapping in Fraud Detection

CLAIRE VERDIER

OpenStudio

STEPHANE PERRIOT

OpenStudio

ARNAULT PACHOT

OpenStudio

This paper proposes a new loss function for unbalanced binary data sets with overlapping classes. This loss function is a weighted cross-entropy binary function that takes as its argument the distance between an observation of the majority class and an observation of the nearest minority class. The distance is calculated using the nearest neighbor algorithm. The quality of this loss function is measured with the AUC-ROC metric on a fraud detection data set for energy saving certificates. In addition, the reduction in the workload for human verification is calculated. The new loss function improves the result. A statistical test confirms the improvement of the model using the new loss function.

CCS CONCEPTS •

Additional Keywords and Phrases: Fraud Detection, Weighted Cross-Entropy, High Class Imbalance, Data Overlap

1 INTRODUCTION

To fight global warming, the French government has introduced energy saving certificates, and energy producers are responsible for financing energy renovation programs. Under the French law, everyone can benefit from financial support for energy renovations that reduce energy consumption. As in many areas, there are people who want to benefit from this aid without doing any energy work. Fraudsters seek to reproduce files that are similar to real files. Thus, the data sets have regions where fraudulent and non-fraudulent files overlap. In addition, the fraud rate in the data sets is less than 5%. Unbalanced data sets have to be dealt with in many domains, such as the financial domain with bank card fraud [13,16], the medical field with the detection of disease, automobile fraud and even telecommunications [1]. We also are faced with unbalanced classes that overlap. The requirement is to minimize the number of files to be checked while maximizing the number of fraudulent files detected. In the context of decreasing the workload with an unbalanced and overlapping data set, this study proposes a weighted cross-entropy method, with the distance between an evaluated file and the nearest known fraud obtained through the 1-nearest neighbor algorithm. Cases that are strongly predicted to be fraudulent are then checked. In this case, it does not matter whether non-fraudulent files are mis-detected as fraudulent, since each suspicious file is examined by a person. The aim is therefore to reduce the workload.

To predict the files, we use *LightGBM*, which is a gradient boosting algorithm. According to [4,10], gradient boosting is the best algorithm for predicting fraudulent files.
The rest of the paper is divided as follows. Section 2 describes the different methods used by the community to deal with unbalanced data sets. Section 3 introduces distance weighted cross-entropy. Section 4 describes the data set used and the results obtained.

2 LITERATURE REVIEW

In the presence of unbalanced classes, the loss caused by the frequency of classes may dominate the total loss and cause instability during learning. To overcome this problem, the machine learning community has developed two types of technique. The first is the SMOTE technique [5]. This is a mixture of the undersampling of the majority class and the oversampling of the minority class. This technique eliminates some observations from the majority class and generates observations from the minority class. Undersampling applied to the existing methods can lead to information loss and degradation of the overall performance, but, according to [16], data rebalancing is effective in delivering good results. SMOTEBoost is a combination of SMOTE and a boosting procedure. Nitesh V. Chawla et al. [6] rely on the SMOTE technique to construct this variant. SMOTEBoost performs better than SMOTE for extremely or moderately unbalanced data sets. Another method derived from SMOTE, called MSMOTE [9], relies on the *k*-nearest neighbor algorithm to perform oversampling. Another method to counter the problem of class imbalance is to weight the loss function. The most common choice for the loss function for classification is cross-entropy. Cross-entropy gives equal importance to each class in the data set. It misclassifies the minority classes. The method developed to overcome this problem is to add a weight. This weight is often posed as the inverse of the class frequency. Chen Wang et al. [17] propose a weighted cross-entropy for a binary classification with unbalanced classes in the case of medical data sets. The aim of this loss function is to increase the misclassification penalty for the minority class. Zhenchuan Li et al. [11] propose a hybrid method and a weighted cross-entropy for credit card fraud detection. They separate the data set into two parts: a non-overlapping set and an overlapping set. All observations in the non-overlapping subset are considered to be non-fraudulent. For the overlapping subset, they use dynamic weighted entropy whose dynamic weighting is based on the signal-to-noise ratio. Shyam Prasad Adhikari et al. [2] propose a distance weighted cross-entropy to detect forest trails in images. Their objective is to detect a forest trail in a forest. They weight each pixel by its distance from the nearest forest road pixel. A pixel representing a forest road is weighted by the maximum distance between a non-forest pixel and a forest pixel. Yashiang Ho and Samuel Wookey [8] propose a new loss function to solve unbalanced class problems such as those that occur in finance. The real-world-weight cross-entropy weights each class with a different weight to make it easier to classify the minority class. To compensate for extremely unbalanced data sets, the focal loss is introduced [12]. In practice, the authors add an unbalanced parameter. Many authors, like those presented above, modify the loss function to fit the model to a complex data set. Jonah Mushava and Michael Murray [14] use the weighted focal loss function and change the link function. The link function is a function that transforms the prediction given by the algorithm between 0 and 1. This facilitates interpretation. Generally, the link functions in loss functions are symmetrical. According to Mushava and Murray [14], it makes more sense to use an asymmetric link function for unbalanced and complex data sets. They then propose using the quartile function of the generalized extreme value distribution. To deal with unbalanced overlapping classes, Artit Sagoolmuang and Krung Sinapiromsaran [15] propose a loss function that they call class overlapping-balancing entropy (OBE).

This function puts more weight on observations that are outside the overlapping regions. It therefore focuses on misclassifications in non-overlapping areas. Chujai et al. [7] establish a completely different method to counteract the problem of an unbalanced class distribution and the overlapping of observations between the two classes. They split each subset of the data set into k clusters using the k-means algorithm. They apply the support vector machine (SVM) algorithm to each cluster created.

3 DISTANCE WEIGHTED LOSS FUNCTION

To define the *distance weighted loss function*, we use, as a basis, the weighted binary cross-entropy function [17]. This function is given by the following equation:

$$H_{XGBoost} = -\sum_{i=1}^{N} \alpha y_i \log \hat{y}_i + (1 - y_i) \log \hat{y}_i \quad (1)$$

N is the number of training examples. y_i is the target label for training example i. $\hat{y}_i$ is the prediction for training example i. α is the weight. If α is greater than 1, the fraud penalty and the importance of the correction of the model weights is increased, whereas if α is smaller than 1, the error and therefore the correction of the model weights is reduced. The weighted cross-entropy function weights the minority class by the same weight. The majority class is weighted by 1. This function does not differentiate between the different files in the majority class. It is therefore important for this function that the majority class is properly classified. Importance is given to the classification of the minority class. With a single weight, the overlap between the majority and the minority class is not taken into account. We recall that the error is acceptable in our situation, since after the algorithm is applied, there is verification by a person. Thus, if two files are very similar and one is fraudulent, it is acceptable for the non-fraudulent file to be a suspect file, as long as this facilitates the detection of other fraudulent files. With this in mind, the idea is to weight the non-fraudulent records by their distance from the nearest fraudulent record, calculated using the 1-nearest neighbor algorithm. Only fraudulent files are chosen as neighbors. The distance-weighted cross-entropy can thus be defined:

$$H = -\sum_{i=1}^{N} y_i \log \hat{y}_i + \alpha_i (1 - y_i) \log \hat{y}_i \quad (2)$$

where the α_i are a discontinuous function that take the distance *1*-nearest neighbour (NN) as their argument. We optimize the parameters α_i with the *Optuna* Python package. We decided to separate the distance into deciles, and then for each interval between two deciles to assign a value of α_i. One could equally decide to divide the distances into quartiles or percentiles instead.

We reiterate that the prediction probability $\hat{y}_i$ is defined by subscript $\hat{y}_i = \sigma(z_i)$ with z_i being the prediction given by the XGBoost algorithm and σ() representing the sigmoid function.

The partial first derivative is shown below:

$$\frac{\partial H}{\partial z_i} = -\alpha_i^{1-y_i}(y_i - \hat{y}_i) \quad (3)$$

The partial second derivative is:

$$\frac{\partial^2 H}{\partial z_i^2} = \alpha_i^{1-y_i}\hat{y}_i(1 - \hat{y}_i) \quad (4)$$

4 EXPERIMENTS AND RESULTS

4.1 The data set and its labeling potential

The data set comes from seven files extracted from a company database. One variable in the data set indicates whether the company carrying out the work has confirmed the renovation work. This variable is our variable of interest. In the database, 40,000 of the 174,000 observations have been verified. By default, the variable is marked as not evaluated. The observations for which this variable has been verified are selected for the training set and the test set. Each file for which the construction company did not respond is considered a fraudulent case. Of the 40,000 records in the data set, only 293 files are identified as fraudulent. The minority class represents 0.7325% of the data set. The two classes are therefore highly unbalanced.

Principal component analysis (PCA) is performed with two components. The explained variance is 93.78%. As can be seen in Figure 1, the majority class and the minority class overlap in one region.

The idea is that observations in the neighborhood of the frauds located in (160;0) can be considered as files for which there is a high suspicion of fraud if this makes it easier to detect fraudulent files belonging to this neighborhood.

Thus, the data set is quite unbalanced, with overlapping classes.

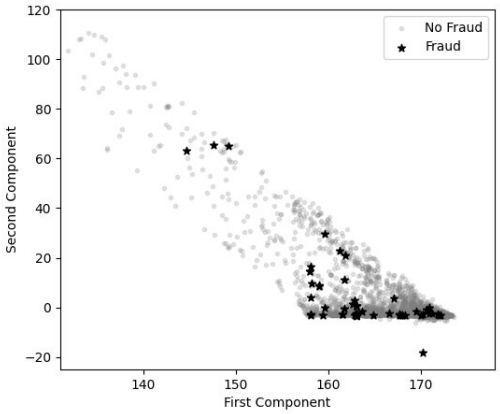

Figure 1: Representation of a part of the data set

4.2 Evaluation

In the literature, it is common to use the confusion matrix or accuracy ($Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$) for evaluation. However, in order to calculate these metrics, a threshold must be set. We do not yet have a predefined threshold. In addition, the team carrying out the checks has not yet defined its workload reduction target. This is why we use a workload reduction chart in the following. Furthermore, in the presence of an unbalanced data set, it is not appropriate to use these metrics. It is preferable to use the ROC (Receiver Operating Characteristics) curve. From this, we can decide on a threshold. We evaluate the quality of the algorithm using the AUC-ROC (Area Under the Curve – Receiver Operating Characteristics) metric. This metric calculates the area under the curve of the ROC metric. It represents the performance of the classifier for all possible ratios. In addition, we compare the curves plotted by the recall rate of the model. The recall rate is defined as:

$$Rappel = \frac{\text{True positive}}{\text{True positive + False negative}} \quad (5)$$

By visually comparing the curves we can see the quality of the models for each threshold. In order to see the difference between the workload reduction curves, as well as the difference between the recall per workload curves, we apply a Wilcoxon test with a significance level of 5%. In addition, we adapt the permutation test to see if the AUC metric of the proposed model is significantly different from the AUC of the model proposed in the library *LightGBM*. Let u_i be the prediction scores of the model proposed in this article and v_i the prediction scores of the model presented in the library *LightGBM*. Let X_i be an i.i.d. variable which follows a Bernoulli distribution with parameter $\frac{1}{2}$. We define the following equations:

$$\tilde{u}_i = u_i X_i + v_i(1 - X_i) \quad (6)$$

$$\tilde{v}_i = v_i X_i + u_i\left(1 - X_i\right) \quad (7)$$

We then calculate the AUC value with these new groups of prediction scores. We define x_i.

$$x_i = \left|AUC\left(\tilde{u}_i\right) - AUC(\tilde{v}_i)\right| \quad (8)$$

The above experiment is repeated N times. From the x_i obtained, we calculate the probability that the x_i values are greater than the difference between the two AUCs obtained from the two models. This gives a p-value and a way to tell if the difference is significant.

4.3 Feature engineering

We have a table with 44 initial variables. The aim of feature engineering is to generate the maximum amount of information that can be introduced into the gradient boosting algorithm *LightGBM*. The unordered categorical variables are transformed into hot vectors. We also construct variables from public tables. We create the variable for the measurement of the geographical distance between the location of the renovations and the location of the business. To reduce the time dependency of the data set, date variables are used to create a variable consisting of the days of the week of those dates. In addition, the difference between two dates is calculated to create another time independent variable. For example, the initial dates are the registration of the author or the creation of the application. Thus 78 different variables are generated. Missing values are replaced by the mean or median, depending on the distribution of the variable. From the 78 variables, we use ridge regression with the regularization parameter of 0.5 and the Sequential Features Selector with the backwards method. Thus, we select 34 variables. All the variables are normalized. Indeed, the *LightGBM* model has two regularization hyper-parameters. To use these parameters, it is necessary to normalize the variables. Furthermore, the 1-NN is not influenced by the normalized variables.

4.4 Unsupervised learning as additional feature

Unsupervised learning creates clusters with an unlabeled data set. In our case, it is used to perform outlier detection. Following Zhao et al. [18], we use the scores given by *Isolation Forest* and *Local Outlier Factor* to generate two new variables. Final list of features is given in Table .

Table 1: Final list of features

#	Name	Values
1	FraudLabel	{0,1}
2	ManGender	{0,1}

3	CustomerType_CL	{0,1}
4	CP_equal	{0,1}
5	DistTravEtabl	[0,1]
...	...	...
15	InscriptionDay	[0,1]
16	CreationFileDay	[0,1]
17	CostEstimateSignatureDay	[0,1]
18	StartWorkDay	[0,1]
19	CostEstimatesWE	[0,1]
20	CreationDayDiff	[0,1]
21	CostEstimatesDiff	[0,1]
22	StartWorkDayDiff	[0,1]
23	EndWorkDayDiff	[0,1]
24	BillDayDiff	[0,1]
...	...	...
33	LO_score	[0,1]
34	IF_score	[0,1]

4.5 Results

With the library *Optuna* [3], we optimize the hyper-parameters of *LightGBM*, such as the parameter *bagging_fraction* which controls the size of the sample to be taken at each iteration. The values of the hyper-parameters chosen for the model are given in Table 2.

Table 2: Values of hyper-parameters of *LightGBM*

Names	Values
Number of iterations	2379
Bagging_fraction	0.475073
Bagging_freq	15
Feature_fraction	0.685061
Lambda_L1	1.9558e-08
Lamba_L2	2.706e-04
Learning_rate	0.01657
Max_bin	738
Min_data_in_leaf	23
Min_gain_o_split	1.281322
Num_leaves	139
Boosting_type	Gbdt
Num_threads	os.cpu.count()
Seed	0
Verbosity	-1

The parameter *number_threads* defines the number of cores chosen by the algorithm. The value *os.cpu_count* allows the number of cores available on the computer to be chosen. For the *LightGBM* model, three more parameters are added to define the loss function and the weighting of the loss function. These three parameters are described in Table 3.

Table 3: Values of hyper-parameters of *LightGBM for Cross Entropy XGBoost*

Names	Values
Metric	Binary_logloss
Objective	Binary
Is_unbalanced	True

We optimize these parameters for the *LightGBM* model and keep these parameter values for the proposed model. In addition, the 10 α parameters that weight the majority class according to its position in the distance histogram are also optimized. Table 4 gives the hyper-parameter values of the weighted cross-entropy. We also tested with other quartiles.

Table 4: Hyper-parameter values

#	Quantile interval	Distance interval	α' values
α_1	[0%,10%]	[0, 3.22]	4.787912
α_2	(10%, 20]	(3.22,3.52]	4.457169
α_3	(20%,30%]	(3.52,3.74]	2.264252
α_4	(30%,40%]	(3.74,3.96]	0.298478
α_5	(40%, 50%]	(3.96,4.2]	0.66746
α_6	(50%,60%]	(4.2,4.5]	3.5460604
α_7	(60%,70%]	(4.5,4.95]	4.727532
α_8	(70%,80%]	(4.95,5.58]	4.38085
α_9	(80%,90%]	(5.58,7.76]	1.7592849
α_{10}	(90%,100%]	(7.76, 275.78]	1.853092

However, the more precise the quartile, the more hyper-parameters we have to optimize. For example, if we divide into percentiles, we have to optimize 100 hyper-parameters. The percentile separation increases the accuracy, but produces a large number of hyper-parameters to optimize. The computation time to optimize these 100 hyper-parameters will therefore increase. The distance weighted cross-entropy is compared with the weighted cross-entropy implemented in the *LightGBM*. The weighting parameter for the cross-entropy of *LightGBM* is chosen by the parameter *is_unbalanced*. The following metrics are obtained and are given in Table .

Table 5: Comparison of values of metrics

Cross-Entropy Type	AUC-ROC
Weighted Cross-Entropy	0.949
1-NN Weighted Cross-Entropy	0.964

According to AUC, weighting the cross-entropy by a function that takes the distance generated by the 1-NN algorithm as an argument improves the performance of the algorithm relative to a single weight. Furthermore, we have seen that the AUC metric is a global metric. Note that, to plot the graph, the files are ranked according to their prediction score obtained with the algorithm. The files are listed in descending order. A score close to 1 means that the file is strongly suspected of being a fraudulent file, whereas a score close to 0 establishes that

the file is weakly suspected of being fraudulent. From Figure 2, it can be seen that the proposed model improves the detection of fraudulent files. By checking 27% of the files most strongly suspected of being fraudulent, all the fraudulent files are detected, whereas with the old model, 68% of the files most strongly suspected to be fraudulent had to be checked to detect all the fraudulent files.

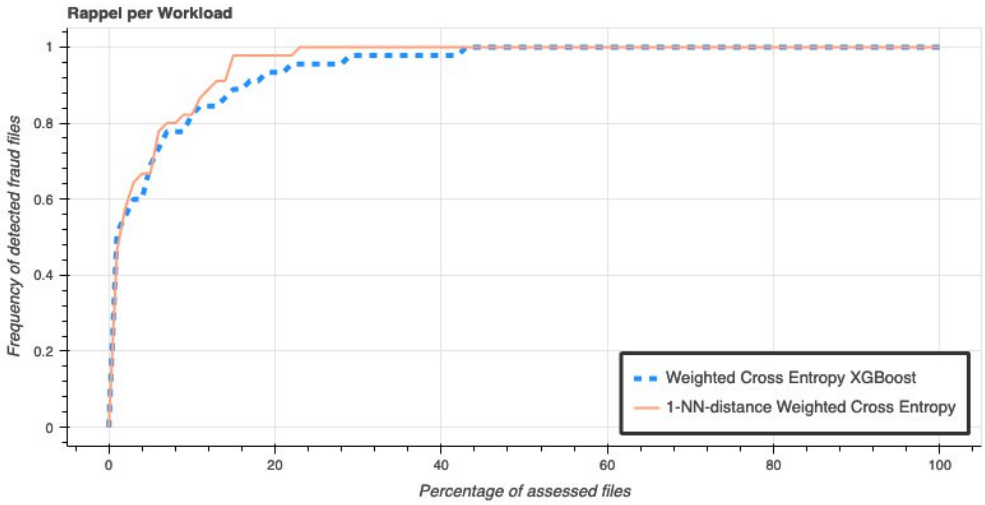

Figure 2: Rappel per Workload

On the other hand, we are interested in reducing the workload. Moreover, the monitoring team has not yet decided how many files they want to check after the algorithm. This is why we create Figure 3, which represents the number of files checked on the abscissa and the reduction in workload on the ordinate. To calculate this curve, we subtracted the recall of our algorithm from the recall if we randomly chose the files to be evaluated. It should be noted that Figure 3 is a performance indicator for workload reduction. It should in no way help in the decision of which threshold to set. It can be seen that the model with distance-weighted cross-entropy 1-NN produces a better workload reduction. With the Wilcoxon test, we can say that, with a confidence level of 95%, the curves of the recall per workload, in Figure 2, are on average significantly different. Similarly, with a confidence level of 95%, the workload reduction curves shown in Figure 3 are, on average, significantly different. Furthermore, the permutation test that was adopted allows us to say that, with 95% confidence, the values for the AUC metric are significantly different.

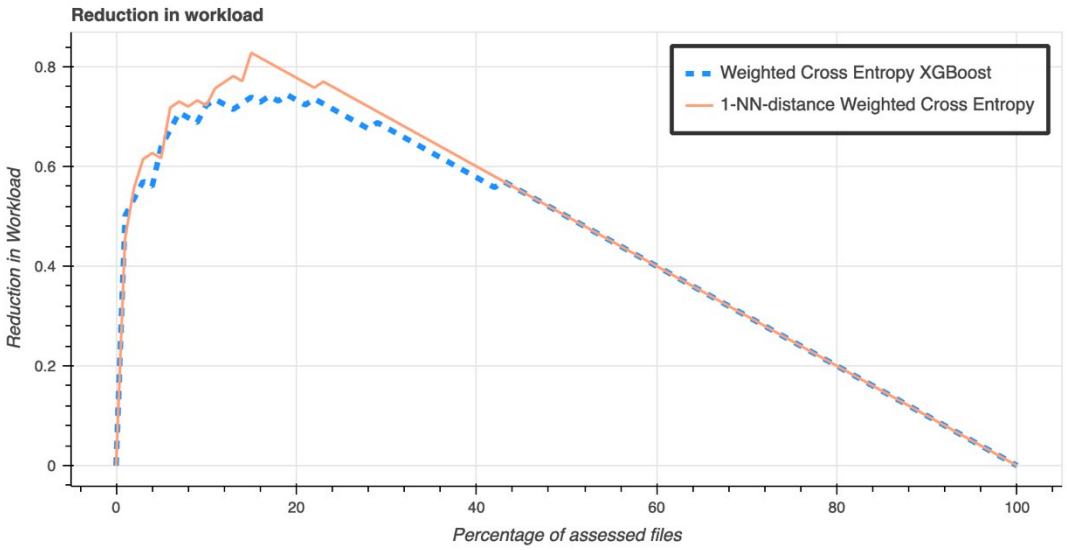

Figure 3: Reduction in Workload

5 CONCLUSION

In this paper, we propose a weighted loss function to solve the binary classification problem with unbalanced and overlapping classes. This function penalizes the observations of the majority class according to their distances from the observations of the minority class. In other words, it does not penalize the minority class. This method does not increase the probability of detecting a new fraud far away the existing frauds in the training set, as is the case with the other methods. This model helps the generalization by minimizing errors, so the model has more tolerance. The results prove that the method proposed in this paper works well with the *LightGBM* algorithm and improves its performance. In order to avoid data drift, the machine learning system must be regularly trained by adding additional verified files. The next step is to compare the different loss function transformation methods on several standard fraud detection data sets.

REFERENCES

[1] Aisha Abdallah, Mohd Aizaini Maarof, & Anazida Zainal (2016). Fraud detection system: A survey. Journal of Network and Computer Applications, 68, 90-113. https://doi.org/10.1016/j.jnca.2016.04.007.

[2] Shyam Prasad Adhikari & Hyongsuk Kim. (2020). Distance weighted loss for forest trail detection using semantic line. Advanced Concepts for Intelligent Vision Systems: 20th International Conference, Auckland, New Zealand, Proceedings. Springer-Verlag, Berlin, Heidelberg, 302-311. https://doi.org/10.1007/978-3-030-40605-9_26.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. (2019). Optuna: A next-generation hyperparameter optimization framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining (KDD '19), 2623-2631. https://doi.org/10.1145/3292500.3330701.

[4] Yung-Chia Chang, Kuei-Hu Chang, & Guan-Jhih Wu. (2018). Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. Applied Soft Computing, 73, 914-920. 10.1016/j.asoc.2018.09.029

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, & W. P. Kegelmeyer. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357.

[6] N. V. Chawla, A. Lazarevic, L. O. Hall, & K. W. Bowyer. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds) Knowledge Discovery in Databases: PKDD 2003. Lecture Notes in Computer Science, 2838. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39804-2_12.

[7] P. Chujai, K. Chomboon, K. Chaiyakhan, K. Kerdprasop, & N. Kerdprasop (2017). A cluster based classification of imbalanced data with overlapping regions between classes. Proceedings of the International MultiConference of Engineers and Computer Scientists, 1, 15-17.

[8] Y. Ho and S. Wookey. (2020). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. IEEE Access, 8, 4806-4813. doi: 10.1109/ACCESS.2019.2962617.

[9] S. Hu, Y. Liang, L. Ma, & Y. He. (2009). MSMOTE: Improving classification performance when training data is imbalanced. Second International Workshop on Computer Science and Engineering, 13-17. doi: 10.1109/WCSE.2009.756.

[10] K. Huang. (2020, November). An optimized lightgbm model for fraud detection. Journal of Physics: Conference Series, 1651, 1, 012111.

[11] Zhenchuan Li, Mian Huang, Guanjun Liu, & Changjun Jiang. (2021). A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection. Expert Systems with Applications, 175, 114750, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.114750.

[12] T. Y. Lin, P. Goyal, R. Girshick, K. He, & P. Dollár, P. (2017). Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision, 2980-2988.

[13] S. P. Maniraj, A. Saini, S. Ahmed, & S. Sarkar. (2019). Credit card fraud detection using machine learning and data science. International Journal of Engineering Research, 8(9), 110-115.

[14] Jonah Mushava & Michael Murray (2022). A novel XGBoost extension for credit scoring class-imbalanced data combining a generalized extreme value link and a modified focal loss function. Expert Systems with Applications, 202, 117233. https://doi.org/10.1016/j.eswa.2022.117233.

[15] A. Sagoolmuang & K. Sinapiromsaran (2020). Decision tree algorithm with class overlapping balancing entropy for class imbalanced problem. International Journal of Machine Learning and Computing, 10(3), 444-451.

[16] D. Varmedja, M. Karanovic, S. Sladojevic, M. Arsenovic and A. Anderla (2019). Credit card fraud detection – Machine learning methods. 18th International Symposium INFOTEH-JAHORINA (INFOTEH), 1-5. doi: 10.1109/INFOTEH.2019.8717766.

[17] Chen Wang, Chengyuan Deng, & Suzhen Wang. (2020). Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. Pattern Recognition Letters, 136. 10.1016/j.patrec.2020.05.035.

[18] Y. Zhao & M. K. Hryniewicki (2018). XGBOD: Improving supervised outlier detection with unsupervised representation learning. 2018 International Joint Conference on Neural Networks (IJCNN), 1-8. doi: 10.1109/IJCNN.2018.8489605.