ETH zürich

Credit Card Fraud Detection using Machine Learning

Rayan Takka	20-909-511
Shujun Wang	19-772-250

July 20, 2021

363-1098-00L Business Analytics FS2021

Prof. Dr. Stefan Feuerriegel

ABSTRACT

With the rapid development of the global economy and the rise of the Internet, the use of credit cards has become increasingly popular. However, credit card fraud has become a major problem in the financial sector which causes huge financial losses to credit card holders and banks. Therefore, it is imperative to establish a reasonable and effective credit card fraud identification mechanism. Recently, machine learning techniques have been widely applied to credit card fraud detection and have shown a good performance. This paper presents a thorough study of machine learning methods for the credit card fraud detection problem and compares their performances on a European credit cards transaction dataset. Experimental results show great performance of the proposed machine learning models and demonstrate the effectiveness of machine learning models for credit card fraud problem.

Keywords: credit card fraud, detection techniques, machine learning, under-sampling

Contents

1. Introduction

This first section introduces some facts behind our topic and shows our motivation as well as the main challenges we face in this project.

1.1 Background

Credit cards are very popular all over the world among users thanks to their security, speed and convenience, making them one of the most important payment methods in modern life, especially in developed countries such as in Europe or in North America, where the credit card business system is quite well developed. Currently, there are 2.8 billion credit cards in use worldwide and the number of cards per capita in the United States has reached 2.9 in 2017.[1] In the meanwhile, the significant increase in credit card issuance and usage has also brought an increase in the number of fraudulent credit card transactions worldwide. According to Merchant Savvy[2] Global payments fraud has tripled from 2011 to 2020, rising from \$9.84 billion to \$32.39 billion. The problem of credit card frauds not only causes credit card holders and banks to suffer huge financial losses, but also negatively affects the security reputation of banks. Therefore, it is urgent to establish an effective credit card fraud identification mechanism.

At the present stage, credit card fraud identification adopts a manual identification mechanism supplemented by computer. In recent years, due to the surge of huge transaction volume, high requirements on data processing methods are expected, and the traditional method of credit card fraud identification can no longer meet the actual needs. In addition, the datasets of credit card transaction records naturally have a serious data imbalance problem, i.e., the volume of legitimate transactions far exceeds the volume of fraudulent transactions, posing a great obstacle to behavior identification and putting forward higher requirements for credit card fraud identification mechanisms.

1.2 Challenges of credit card fraud identification

Among the established approaches, the practice of credit card fraud identification based on

[1] https://shiftprocessing.com/credit-card/
[2] https://www.merchantsavvy.co.uk/payment-fraud-statistics/

machine learning has two main challenges: the extreme imbalance of labeled data and the trade-off between cost and effectiveness.

1.2.1 Extreme imbalance in labeled data

The imbalance in labeled data is one of the main problems in credit card fraud detection, i.e., only a very small fraction of credit card transaction data is fraudulent and the majority is not. Training machine learning classifiers with imbalanced annotated data, without targeted processing, will inevitably lead to a bias towards the majority class in the data set, or even to a unilateral classifier, which will negatively affect the model training. Labeled data imbalance is not unique to the credit card fraud problem but is also common in all kinds of practical application datasets. A reasonable solution to the dataset imbalance problem in credit card fraud prediction will provide a reliable reference for solving similar problems in a wide range of real-world applications.

1.2.2 Cost-Effectiveness tradeoff

In the practical application of machine learning methods, the tradeoff between cost and effectiveness directly affects the model evaluation criteria and training methods. From the perspective of machine learning classifiers, each correct classification and each misclassification between categories represent one event with the same weight. In practice, however, each transaction is unique in that it is a different amount of transactions generated by a different entity, so the cost of monitoring each credit card fraud event is a variable potential loss rather than a fixed error cost. In this case, it is necessary to build error cost measurement models based on machine learning models and further consider the needs of practice to ensure the maximization of the model cost-utility ratio and feasibility.

2. Experimental Dataset

One main focus of this study is to determine the performance of classifiers for credit card fraud detection on a dataset having a varied number of samples and features. For this purpose, we use a dataset called European dataset. This dataset is highly imbalanced. This problem is common to the credit card fraud datasets where fraud transactions are very less compared to normal transactions. Further information about this dataset and data pre-processing methods are given below.

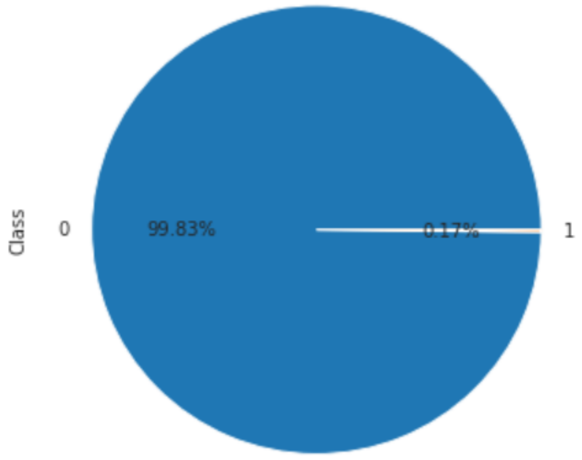
2.1 Data Description

This dataset courtesy of Machine Learning Group of Université Libre de Bruxelles is retrieved from Kaggle[3]. It contains two days of transaction data of European cardholders in September 2013. This dataset contains 284,807 samples and 31 features, of which 492 transactions are fraudulent, accounting for only 0.173% of all transactions. Fig. 1 shows the imbalance problem of this dataset. Due to the privacy of customer information and the sensitivity of transactional details, all except 'Time' and 'Amount' features in the dataset are principal component analysis (PCA) transformed, as shown in Table 1.

Table 1 Attributes of European dataset

Feature	**Description**
Time	Time in seconds passed starting from the first sample in the dataset
Amount	Transaction amount
Class	0 – not fraud; 1- fraud

Figure 1 Percentage of each Class



2.2 Data Exploratory Analysis

Fig. 2 shows the correlation matrix of the dataset. This matrix explains that class is independent of both the amount and time when the transaction was made. It is clear from that matrix that the class of the transaction depends mostly on PCA applied attributes.

[3] https://www.kaggle.com/mlg-ulb/creditcardfraud

Figure 2 Correlation matrix of Attributes

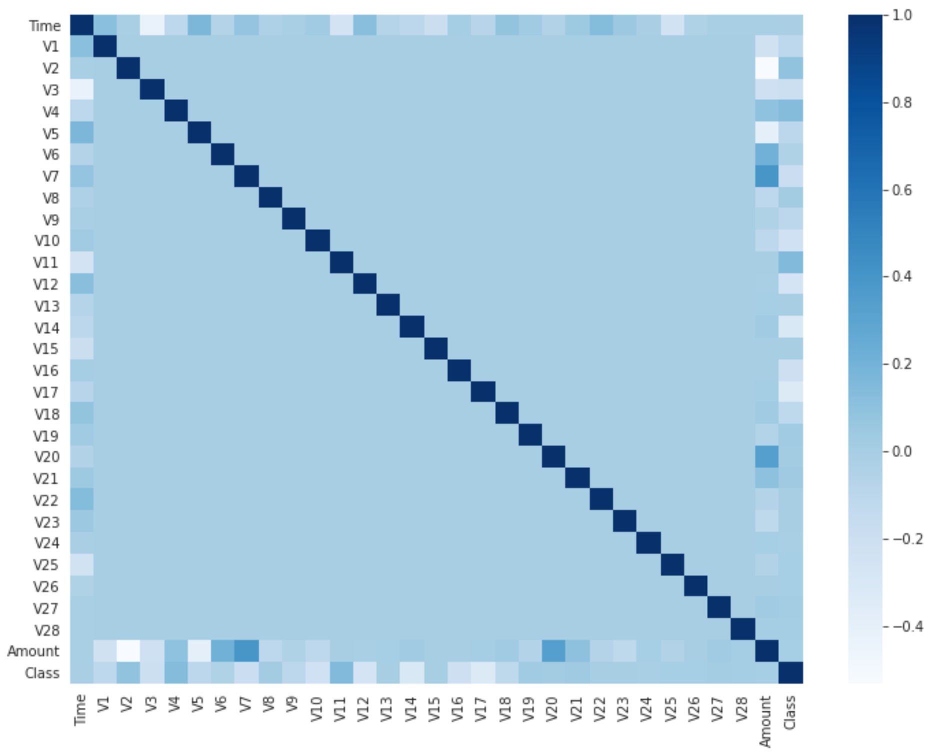
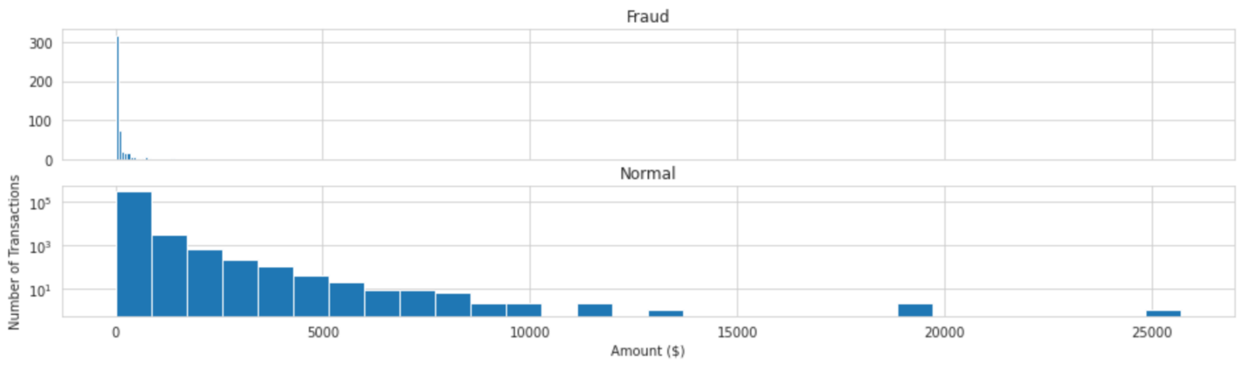


Fig. 3 shows normal transactions and fraudulent transactions over amount, indicating a strong aggregation of consumption amount distribution at low amounts, which could provide some reference value for the following evaluation and selection of our models.

Figure 3 Transaction amount distribution



2.3 Data Pre-processing

The straightforward way to solve the class imbalance problem is on data-level, like over- or under-sampling that is used to balance the classes before applying. According to Drummond[3], using under-sampling performs better than over-sampling on solving imbalanced dataset problems, so we adopt the under-sampling technique. With this technique, we adjust the largest class to match the number of samples of the under-represented class. We randomly pick an amount of non-fraudulent transactions that is equal to the number of fraudulent transactions in the dataset. We use the same set of feature variables, the same label variables and the same set of training and testing datasets in all algorithms. Only the training dataset (70%) is used for model training, and the test dataset (30%) is completely independent from the training dataset for the independent testing of model prediction.

3. Experimental Study

In this section, we report our experimental study that we performed with selected machine learning algorithms and provide a detailed description of the design of experiments followed by the results and a discussion.

3.1 Machine Learning Algorithm

In our experiments, we decided to apply 5 different classification machine learning algorithms: logistic regression, random forest, support vector machine, k - nearest neighbor and neural networks.

3.1.1 Logistic Regression

Logistic regression models relationships between a set of variables. These variables can be dichotomous (yes/no), categorical or continuous. The outcome of this method is binary (Yes or No). In our case, the outcome will simply be if the considered transaction is fraudulent or not.

The model used will consider one binary response variable Y associated to the transaction type (fraudulent or not) and a set of variables $x_1, x_2, x_3, x_4 ... x_{28}$ called the predictors. The first assumption made is that these variables are linearly related to the logit of the binary response variable as it can be seen below.

$$ln(\frac{p}{1-p}) = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{28} x_{28}$$

Where p is defined as $p = P(Y = 1)$ and β_i, $i \in [1, 28]$ are the model's parameters.

Given our training example (x_i^j, y^j), j being the samples index, the next step is to compute the likelihood (assuming independence of training examples) as defined below.

$L(a)=\prod_j p(y^j | x_i^j; a)$, where a is defined as $a = (X^T X)^{-1} X^T y$. with $X = (x_i^j)_{(i,j)}$ and $y = (y^j)_j$.

Our strategy will be to maximize the logarithm of the defined likelihood. Thus, we will run a gradient ascent algorithm to maximize the log likelihood. The algorithm implemented is described in the following pseudocode:

Given a, $\{(x_i^j; y^j)\}$, we apply:

```
● Initialize $a$ = <1,...,1> $^T$
● Perform **feature scaling** on the examples' attributes
● Repeat until convergence :
    ○ for each j = 0, .., n:  $a_j' = a_j + \sum_j (y^j - p(y^j | x_i^j; a) x_i^j$
    ○ for each j = 0, .., n:  $a_j = a_j$
● Output $a$
```

In our case we directly used logistic regression by importing it from sklearn.linear_model.

3.1.2 Random Forest

Random Forest is a machine learning algorithm that leverages the power of multiple decision trees for making decisions.

The idea is to randomly create decision trees. Each tree will contain nodes that will generate an output by workin on a small subset of features. At the end, all the results are combined to generate the final output of the random forest.

In more details, for each tree in the forest, a bootstrap is selected from our training set S and we denote by S^i the i^{th}bootstrap.Then, we run a slightly modified version of the decision tree algorithm :

For each node of each tree: a subset f ⊆ F of the features is selected. This is the main difference from a classic decision tree where we could be examining all possible feature-splits. The next step is to split on the best feature in f. Thus, we drastically speed up the learning of the tree by narrowing the set of features[4]

In our case we directly used the implemented algorithm Random Forest Classifier by importing it sklearn.ensemble.

3.1.3 Support Vector Machine

SVMs are a family of machine learning algorithms that solve classification, regression and anomaly detection problems. In our case, we'll use it to classify the transaction's class.

The method used by SVM is to find a separating line (or hyperplane) between data of two different classes. It takes the data as an input and outputs a line that separates those classes if possible.
According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Then, we compute the distance between the line and the support vectors. This distance is called the margin. Our final goal is to maximize that margin.

To apply the svm algorithm, we directly build the model thanks to the SVC function from the library sklearn.svm.

3.1.4 k - Nearest Neighbor

k - Nearest neighbor algorithms is one of the most classic machine learning algorithms and

[4] Matthew B., *Random Forests*, University of Wisconsin, Computer Science Department.

has been mostly used in the field of pattern recognition. Nowadays, the kNN algorithm is less popular than it used to be but it is still widely used in practice, and is still highly recommended in classification projects as a predictive performance benchmark when trying to develop more sophisticated models.

The underlying concept behind kNN is relatively easy to understand: it is an algorithm for supervised learning that simply stores the labeled training examples during the training phase .Then, it considers the k nearest neighbors when predicting a class label. The simplest incarnation of the kNN model is to predict the target class label as the class label that is most often represented among the k most similar training examples for a given query point. In other words, the class label can be considered as the “mode” of the k training labels.

We can write in pseudocode the following naive implementation of the kNN algorithm:

```
For a given point x from the training sample:
1. Determine $d(x, x_i)$ ; where d denotes the distance between the points defined by the chosen metrics.
2. Sort the calculated n Euclidean distances in non-decreasing order. Fix $k$ as a positive integer and take the first $k$ distances from this sorted list.
3. Search the $k$-points corresponding to these $k$-distances.
4. Let $k_i$ denotes the number of points belonging to the ith class among $k$ points
05. If $k_i > k_j$, $\forall\, i \neq j$, put x in class i
```

In our case, we’ll implement the algorithm with the function KNeighborsClassifier imported from sklearn.

3.1.5 Neural Networks

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data. Its particularity is that it can adapt to changing input. Therefore, the network generates the best possible result without having to redesign the output criteria after any change.

In our case, we used a simple NN made of 3 connected layers. The neural network takes a vector of length 28 as input. This represents the information related to each transaction, i.e. each line with 28 columns from the dataset. For each transaction, the final layer will output a probability distribution and classify either as not fraudulent (0) or fraudulent (1).

3.2 Evaluation Measures

There are many measures to evaluate model performance. We choose several common measures and in order to calculate them, we compare the confusion matrix (Table 2) of each method that can be defined by:

- True Negative (TN): The amount of correctly classified transactions by the model of no fraud transactions.
- False Negative (FN): The amount of incorrectly classified transactions as fraud cases, but the actual label is no fraud .
- False Positive (FP): Lower Left Square: The amount of incorrectly classified transactions as no fraud cases, but the actual label is fraud.
- True Positive (TP): Lower Right Square: The amount of correctly classified by our model of fraud transactions

Table 2 Confusion matrix

Predicted	**Actual**	
	Normal (0)	Fraud (1)
Normal (0)	TN	FN
Fraud (1)	FP	TP

To further understand the evaluation metrics, consider the equations for Accuracy, Precision, Recall, and the F1 Score given below.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

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

It can be seen that *Precision* is related to positive prediction values. Reducing the number of FPs increases Precision, so for cases where the cost of having FPs is high, Precision is an appropriate metric. *Recall* is correlated with TP. Reducing the number of FNs increases recall, and high recall is often achieved for problems where FNs are costly. For credit card fraud detection, there is a need to balance between FP and FN. Predicting all samples as fraud will have high recall but low accuracy, precision, and F1 score, while predicting all samples as non-fraud will result in high accuracy but zero recall and undefined precision and F1 score. In Section 3.3, we will show the results of all four of these metrics.

3.3 Results

We conducted the experiments with all five machine learning classification algorithms that are described before. The obtained results are summarized in Table 3. According to the results we found, for all algorithms except SVM, an accuracy, precision and a F1 score higher than 90%. We use accuracy as the main evaluation criterion rather than others because we are now using post-sampling category-balanced data, and accuracy has a better evaluation effect. Based on our study with the results, we conclude that of all these algorithms, Neural Network is the most eligible one to be used in credit card fraud classification.

Table 3 Performance of different methods

Model	**Accuracy**	**Recall**	**Precision**	**F1 Score**
LR	0.9122	0.8581	0.9621	0.9071
RF	0.9189	0.8581	0.9769	0.9137
SVM	0.8649	0.8243	0.8971	0.8592
kNN	0.9189	0.8784	0.9559	0.9155
NN	0.9223	0.8784	0.9630	0.9187

We also search for the optimal threshold to better evaluate the model performance in the experiment of Logistic Regression. We show our results through the confusion matrix (Fig. 3[5]) and the Precision-Recall Curve (Fig. 4) setted with different thresholds. From these two figures, we can see that the smaller the threshold value, the larger the recall value, and the more the model can find out the number of credit card frauds, but in exchange, the number of FP is also larger. As the threshold value increases, the recall value gradually decreases, the precision value gradually increases, and the number of FP decreases. The strength of the model against credit card frauds is controlled by adjusting the model thresholds, setting smaller thresholds if you want to find out more credit card frauds, and vice versa, setting larger thresholds.

Figure 3 Confusion matrix

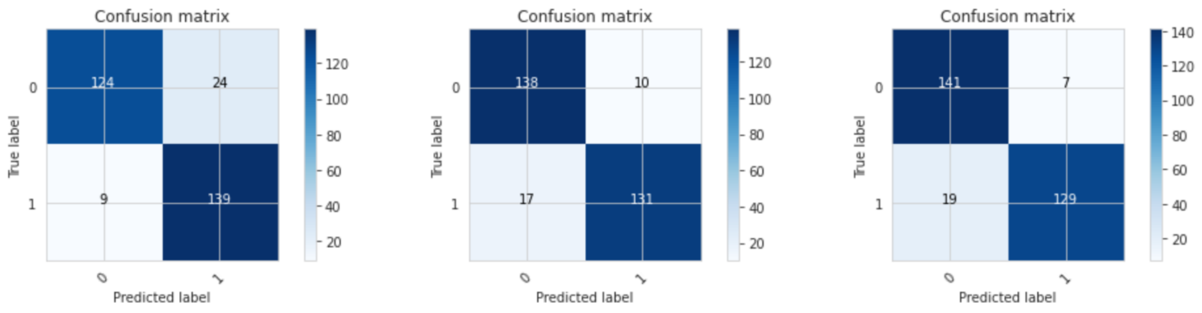
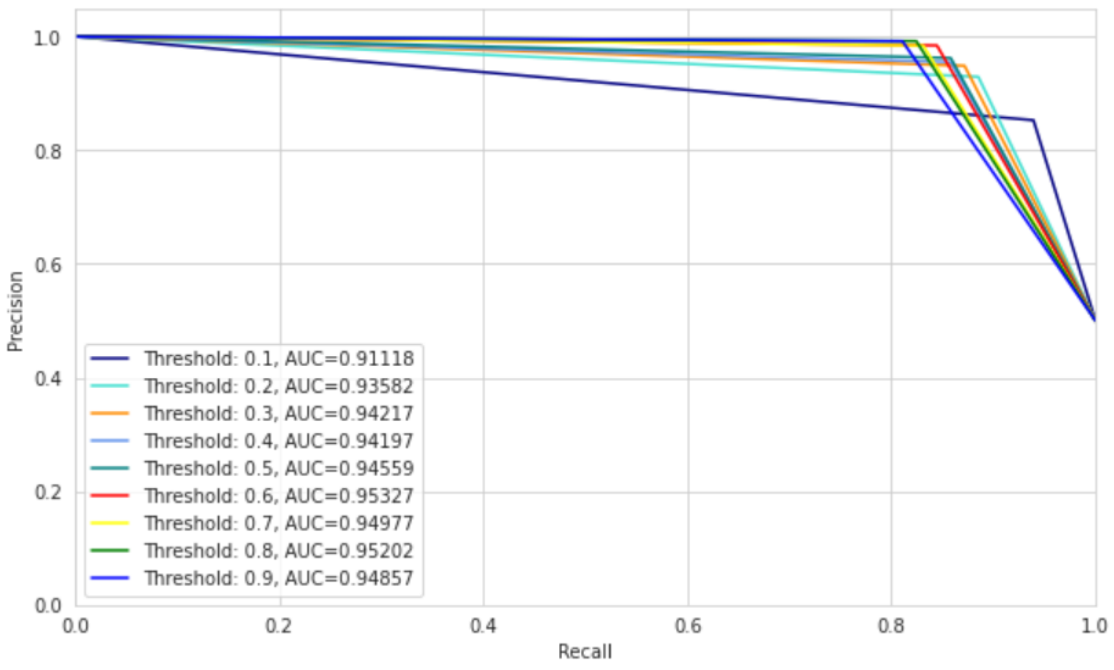


Figure 4 Precision-Recall Curve



[5] We only show the confusion matrix with threshold equals to 0.1,0.2,0.3 here due to the page limit.

4. Conclusion and Future Work

Our main goal during this project was to apply different machine learning methods in order to compare their performance on a transaction dataset. The first issue was related to the imbalance of the dataset. The chosen solution was to apply undersampling to finally get a balanced dataset that can be used in order to apply the machine learning methods. Finally, by undersampling, it was possible to use the accuracy score as a good metric to compare the different methods' performance.

However, other solutions could have been used to evaluate the algorithms' performance on our dataset. For example, it would have been possible to consider a large number of samples (> 20 000) that will thus be more representative of the initial dataset. Given the class imbalance ratio of this large number of samples, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC) since measuring classic accuracy is not meaningful for unbalanced classification.

In terms of how to choose the optimal threshold, it depends in reality on the comparison of marginal profit and marginal cost of the company's business. When the model threshold value is smaller, it could identify more cardholders facing credit card frauds, but as the number of misclassifications increases, it not only increases the workload of the post-credit team, but also reduces the consumption experience of customers misclassified as having their credit cards stolen, which leads to lower customer satisfaction. A model threshold is optimal when it allows the business to balance marginal profit and marginal cost. There are of course exceptions to this rule. A financial crisis is often accompanied by an increased chance of loan default or credit card frauds, and financial institutions will be more willing to guard the bottom line of risk at all costs.

There are also some shortcomings in this paper. First, since the used data is derived from a PCA transformed dataset, all features are not known except for time and the transaction amount. Thus, the interpretability of the credit card fraud detection model is relatively poor. On the other hand, instead of keeping the default hyperparameters for each model, it is possible to tune them in order to understand their impact on the models.

References

[1] Nguyen, T. T., Tahir, H., Abdelrazek, M., & Babar, A. (2020). Deep learning methods for credit card fraud detection. *arXiv preprint arXiv:2012.03754*.

[2] Dornadula, V. N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. *Procedia computer science*, *165*, 631-641.

[3] Drummond, C., & Holte, R. C. (2003, August). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II* (Vol. 11, pp. 1-8). Washington DC: Citeseer.

[4] Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M. S., & Zeineddine, H. (2019). An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access*, *7*, 93010-93022.

[5] Delamaire, L., Abdou, H., & Pointon, J. (2009). Credit card fraud and detection techniques: a review. *Banks and Bank systems*, *4*(2), 57-68.

[6] Maniraj, S. P., Saini, A., Ahmed, S., & Sarkar, S. (2019). Credit card fraud detection using machine learning and data science. *International Journal of Engineering Research and*, *8*(09).