Credit Card Fraud Detection

Guilherme Righetto

[1]Campinas – SP – Brasil

guilherme.righetto@ifood.com.br

1. Introduction

A fraudulent transaction Transaction fraud occurs when a stolen payment card or data is used to generate an unauthorized transaction Credit card fraud is an unauthorized purchase by the card owner, which can happen in several ways, such as card information theft, card cloning, card exchange scams, etc. When this happens, the customer usually opens an order for a chargeback on a specific purchase and the financially responsible needs to reimburse the purchase price, in which the loss is usually left to the financially responsible for selling the item.

In this way, companies have a great interest in containing this type of problem in order to avoid losses and guarantee reliability in their business. However, unfortunately, it is usually not a very easy task, even with huge advances in the anti-fraud area, the fraudsters are improving their methods too, trying even more to simulate a transaction that looks exactly like a real transaction. Using this motivation, businesses need to scale this evaluation to predict if a transaction is fraudulent or non-fraudulent.

Currently, are several solution to attempt to prevent the problem of fraud transactions in companies, such as: creating fixed rules, manually analyzing the data, inserting more validation mechanisms and etc. However, we are living in the big data era, which the data volume is so big that the solutions mentioned in are not viable, both economically speaking, and in relation to performance. For this reason, most companies are betting on the creation of machine learning models, as they have more accurate results and benefit from large volumes of data, and it is for these reasons that in this work we will describe a method to try to achieve good performance in the identification fraudulent transactions.

The motivation of this work comes in two ways, first to try to contain as many fraudulent transactions as possible, taking into account that blocking non-fraudulent transactions brings a great deal of damage to the business, sometimes more than a fraudulent transaction. And second, trying to build a data model without using a lot of data in training, saving processing and consequently time.

1.1. Goals

The main objectives of this work are:

- Create a binary classification model
- Identify the best set of techniques to create a final solution
- Compare the final results of the steps

2. Dataset

The dataset used in this work is public and was published in the Kaggle [1]. The data in the dataset represents the sets of fraudulent and non-fraudulent transactions carried out

[1]https://www.kaggle.com/mlg-ulb/creditcardfraud. Available in 04-06-2020

with a credit card. The dataset was created and analyzed during a collaboration between Worldline and Machine Learning Group at Université Libre de Bruxelles.

The dataset has a total of 284,807 transactions, 492 of which are fraudulent, which represents a standard feature in relation to the fraud detection problem, in which the number of classes (fraudulent/non-fraudulent) is normally very unbalanced. In addition, the main columns are anonymized, in which a PCA transformation was used to perform this task, with columns V1, V2, ...,V28 columns transformed by the PCA, 'Amount' with the transaction value, 'Time' with the seconds between one transaction and another, and finally, the 'Class' column with a value of 1 (fraudulent transactions) or 0 (non-fraudulent transactions).

3. Method

The method for building the classification model will be described in this chapter. The Figure 1 shows all the steps of the method developed in this work.

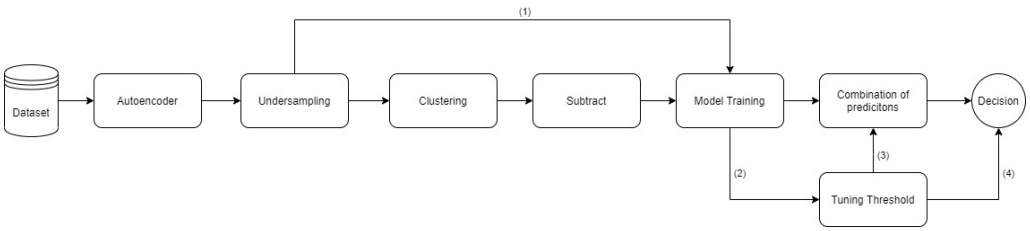

Figure 1. A flowchart with the proposed method.

From the dataset, an auto-encoder was created to extract the vector from each sample. Subsequently, under-sampling was performed to balance the classes of the training set. In the next step we built a cluster with only the negative class to extract the centroid, in which, from the centroid, a subtraction of the characteristic vectors was carried out, consequently generating more samples. Then, to train the `XGBoost` model we perform an optimization of hyper-parameters using Bayesian's optimization. After that we realizes the thresholds' optimization and combine the predictions to build the final prediction. In the end we calculate the metrics to evaluate the model.

The paths described in the Figure 1 with the numbers (1,2,3,4) may or may not be used, these are the differences between the evaluations that are compared.

3.1. Dataset

In this step, we will not perform any more specific treatment, such as evaluating correlations, creating new features, replacing null values, etc. Because the dataset used is the result of a PCA and in relation to the creation of features, the idea of the proposed method is exactly to eliminate the need to work with the characteristics and transfer this process to autoencoder. The only pre-processing performed in this step is the normalization of the characteristics to be in the range of 0 to 1 using the MinMax approach, since the column 'Amount' and 'Time' are not normalized.

3.2. Autoencoder

An autoencoder is a type of artificial neural network that aims to learn a representation for a set of data, and is also used to reduce dimensionality. In addition, the reconstruction of

the input sample is also learned, where the autoencoder tries to generate from the reduced coding a representation as close as possible to its original input, while removing the noise presented in the input samples.

The autoencoder stage is the main stage of the project, with the objective of further discretizing the sample space and consequently facilitating the learning of the classification model. Therefore, to perform this task we will use the latent space of the created network, shown in the Figure 2, performing the training and validation of the network only with non-fraudulent samples.

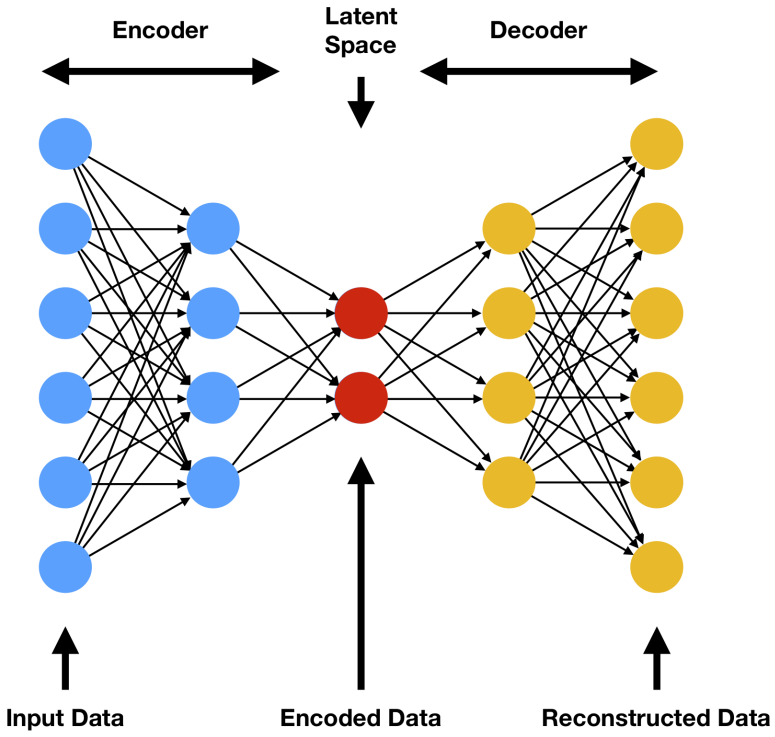

Figure 2. Autoencoder.

As previously described, the motivation for the development of this work is not to use many samples to perform the training of the model, thus, only a portion of the non-fraudulent samples will be used. Subsequently, with the model already trained, it will be applied to all samples in the dataset, including fraudulent samples, in which the network itself will discretize the sample space by learning only about non-fraudulent samples.

3.3. Undersampling

In this step we will carry out the balancing of the classes at random after all samples are predicted for the latent vector of the model described above. This technique is used for the purpose of facilitating the learning of the classifier, that is, it does not learn just one class because of imbalance and also for the motivation of using only a few samples in training.

3.4. Clustering

Clustering will be used for non-fraudulent samples resulting from the undersampling step. The objective of this step is to select the centroid of each of the N clusters created by the

k-means algorithm. And then subtract each sample from the dataset by each centroid selected when creating the clustering model. The number of clusters created will be optimized by the elbow method for each experiment performed. In the next section, the step of subtracting the feature vectors will be better described.

3.5. Subtract

In this step we will subtract the characteristic vector from each sample with the centroid found in the previous step. As only the non-fraudulent samples will be used, the objective is to further discretize the space of the characteristics and to be able to combine the predictions at the end of the evaluation.

Therefore, for non-fraudulent samples, ideally, the feature vector should be close to zero, and for fraudulent samples that is different from zero. In addition, it is worth noting that this approach increases the number of samples in the dataset in relation to how many clusters were created in the previous step.

To facilitate the understanding and motivation of this approach, we can make a comparison, imagine that you are classifying images and they have a lot of information, in this case you decided to divide the samples into blocks to facilitate the learning of each part of the image and then combine the predictions of each block. The only difference in this case is that we will not break the feature vector.

3.6. Model Training

In this step, the classification algorithm `XGBoost` will be used, because in addition to being a robust algorithm and used in several competitions, it normally has calibrated probabilities, which is ideal so that it does not interfere with the result of the optimization of the thresholds. In addition, in this step, we will adjust the hyperparameters using a Bayesian method to extract the best performance to solve the problem.

`XGBoost` is a decision tree based algorithm that uses the ensemble technique called boosting to optimize the creation of more robust models and uses a gradient gradient approach to optimize the final predictions.

Gradient boosting is an approach that at each iteration a new model is created trying to predict the residuals of the previous model, that is, it tries to predict errors from the previous model, using all the models created to make the final forecast and the downward gradient is used to minimize the loss by creating new models.

3.7. Tuning Threshold

The objective of this step is to analyze the predicted probabilities of each class and try to optimize the cut-off limit to define the fraudulent class. In this step, brute force optimization will be used trying to achieve the best possible performance for a given metric. Therefore, several thresholds are tested within a range and the one that has the best performance in relation to the metrics is chosen.

3.8. Combination of Predictions

The combination of the predictions is a consequence of the Subtract step, as we need to combine the predictions to compose the original sample and have only one final result. There are several ways to accomplish this combination, and in this work the following methods will be used:

- **Sum**: Performs the sum of the probabilities of the predictions and the class is chosen in relation to the highest value in the index
- **Average**: Performs the average of the prediction probabilities and chooses from the threshold selected in the previous step
- **Vote**: The class is chosen for each subsample using the optimized threshold and the class with the highest quantity is chosen

3.9. Decision

In this step, we will perform the calculation of the metrics to evaluate the performance of the method. In addition, some experiments will be carried out evaluating different ways of the method to identify the best solution.

3.9.1. Metrics

The metrics used in this work were:

- **F1**: 2*(Recall * Precision) / (Recall + Precision)
- **Recall**: TP/TP+FN
- **Precision**: TP/TP+FP
- **False Positive Rate**: FP/FP+TN
- **False Negative Rate**: FN/FN+TP

Although the metrics are correlated, all of them will be used to be able to carry out a more accurate analysis of all the possible problems found in the problem of detecting credit card fraud. It is worth mentioning that the metrics F1, Recall and Precision, the higher the value is better and the metrics False Positive Rate and False Negative Rate the lower value is better.

4. Experiments

In this chapter, each experiment will be described specifying which techniques were applied to each experiment. However, in all experiments, `5-kfolds` were used to evaluate the execution of each performed experiment, and the metrics explored in the previous chapter were also calculated.

In addition, we will describe all the steps with common settings for all experiments. The first step is to perform the normalization of all the characteristics because we will use algorithms that need the normalized variables. In this work, the `MinMaxScaler` normalization approach was used, leaving all characteristics with values from 0 to 1. Subsequently, the creation of the autoencoder model was performed, which was developed using the Keras framework and the network architecture present in the Table 1 was built through experimentation to get better results.

Next, we have the optimization of the hyperparameters and training of the model, which from the training samples, internal `3-kfolds` were used to find the best hyperparameters, using the Optuna library, with the average of the F1 metrics of the executions. Finally, the optimization of the threshold, which was also performed with internal `3-kfolds` and from the probabilities, the precision metric was calculated for the 0 to

Layer	Quantity	Activation	Parameters
Dense 1	25	Tanh	*L1 regularization = 10e-5*
Dropout 1	-	-	*p =0.5*
Dense 2	20	Tanh	-
Dropout 2	-	-	*p =0.5*
Dense 3	15	Tanh	-
Dropout 3	-	-	*p =0.5*
Dense 4	20	Tanh	-
Dropout 4	-	-	*p =0.5*
Dense 5	25	Tanh	-
Dense 6	30	ReLu	-

Table 1. Configuration of the architecture used in autoencoder.

1 thresholds with the step of 0.001, and the threshold was chosen with the greatest precision. The decision to perform the optimization by precision was due to decrease the number of false positive rate.

Now that we have defined the common settings for all experiments, Table 2 presents all the steps performed in each experiment, specifying the possible possibilities for evaluating the proposed method. Each experiment will be better detailed below.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Step 1	MinMaxScaler	MinMaxScaler	MinMaxScaler	MinMaxScaler
Step 2	Autoencoder	Autoencoder	Autoencoder	Autoencoder
Step 3	Undersampling	Undersampling	Undersampling	Undersampling
Step 4	Clustering	Clustering	Clustering	-
Step 5	Subtract	Subtract	Subtract	-
Step 6	Model Training	Model Training	Model Training	Model Training
Step 7	Tuning Threshold	-	-	Tuning Threshold
Step 8	Combine Predictions (Vote)	Combine Predictions (Sum)	Combine Predictions (Mean)	-
Step 9	Decision	Decision	Decision	Decision

Table 2. Experiment definition overview

Experiments 1,2 and 3 have the clustering stage and consequently the sample subtraction stage. Because, as previously mentioned, the idea is to take the centroid of each cluster and then subtract the samples. In this way, Image 3 presents the result of `K-means` for the three experiments, in which the `K-means++` approach was used to define the centroids, thus, we realized that Experiment 1 will multiply the number of samples by 7, Experiment 2 by 5 and Experiment 3 by 7 because of the number of centroids chosen by the elbow approach and the subtraction of the samples.

In addition, only experiments 1 and 4 have the optimization of thresholds as they are the only experiments that preserve the structure of the prediction of the probabilities, being possible to perform the optimization of the thresholds, since the sum of each pre-

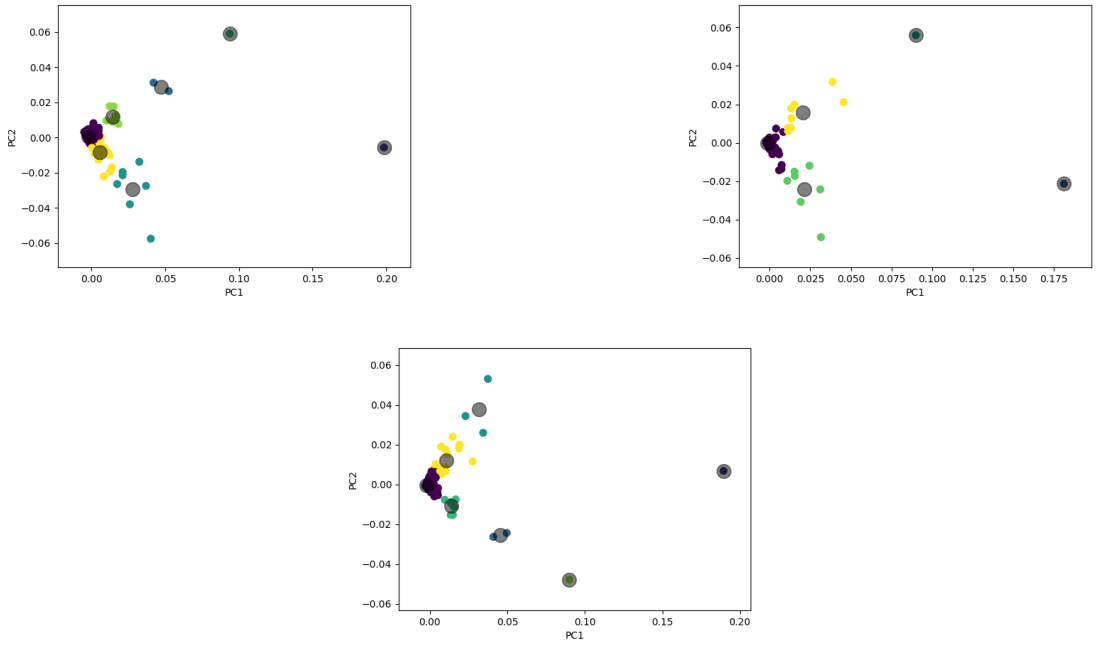

Figure 3. Result of `K-means` of an execution of each experiment 1,2 and 3 respectively.

diction of a sample is always 100% , while in the case of the combination of predictions by sum and average, the sum of the probabilities of a sample can be greater than 100%.

When analyzing experiment 4, we noticed that it skips some steps, because as previously mentioned the idea is to validate the results by going through different steps of the method. Because the goal is to understand if a given technique brings some improvement in the performance of the model or not. In the next chapter, the results obtained by each experiment will be presented and discussed.

5. Results

The Table 3 presents the results obtained from each experiment using the metrics mentioned above, remembering that each experiment was performed 5 times through cross-validation.

	Experiment 1	**Experiment 2**	**Experiment 3**	**Experiment 4**
F1	74.62 +- 6.55	47.37 +- 27.5	66.45 +- 21.5	**77.31 +- 6.5**
Recall (%)	78.85 +- 3.22	80.48 +- 1.95	80.08 +- 1.6	79.66 +- 2.62
Precision (%)	72.49 +- 14.7	41.51 +- 32.4	62.62 +- 24.7	75.54 +- 10.14
False Positive Rate (%)	0.0 +- 0.0	0.0 +- 0.0	0.0 +- 0.0	0.0 +- 0.0
False Negative Rate (%)	21.14 +- 3.22	19.51 +- 1.95	19.91 +- 1.6	20.33 +- 2.62

Table 3. Results

Observing the results, we realized that the experiments with the best performance are Experiment 1 and Experiment 2. A possible justification for the best performance

of these two cases is the execution of the optimization of the thresholds that manages to extract the best result for that particular model to the maximum.

An important point to be mentioned is that experiments 2 and 3 did not obtain such significant results, in which, it demonstrates that the technique of clustering and subtraction of the samples does not bring so many gains to the performance of the model. Another point that justifies this statement is that experiment 4 has very similar performance to experiment 1.

5.1. Comparisons

Follows Table 4 of comparison of results with other works that used the same dataset. Below is a brief description of the techniques used in other works.

- **Kaggle Kernel 1**[2]: In this work the author used the technique of balancing SMOTE classes (Oversampling) and for classification a neural network.
- **Kaggle Kernel 2**[3]: In this work the author performed the classification using Logistic Regression and performs the threshold optimization for the final definition of the class.

	Work 1	**Work 2**	**This Work**
F1	74,29	72.8	77.31
Recall (%)	66.33	61.90	79.66
Precision (%)	84.42	88.35	75.54
False Positive Rate (%)	0.0	0.0	0.0
False Negative Rate (%)	33.67	38.10	20.33

Table 4. Comparisons with other works

When comparing the results of this work with the others, we noticed an improvement in relation to the Recall and F1 metrics. However, each job has strengths, in this case, the precision of the other jobs is better than this job. In conclusion, we cannot determine which method is better than the other, depends on the business goals.

6. Conclusions

From the results we can conclude that the method works best when the threshold is optimized and that the clustering and subtraction tenets of the characteristic vector do not bring much gain to the final performance. Another point, which we noticed in the development and execution of the project, is that the auto encoder has extremely specific characteristics for the training set, thus overfitting the model.

In addition, the execution of the Keras framework has a difficulty in ensuring the reproducibility of the experiments by running in parallel on the CPU or by running on the GPU. Thus, we were unable to conclude that this would be an ideal technique, as it would be necessary to include more regularization techniques to reduce overfitting and perform a deeper analysis on the latent vector generated.

[2]Available at: kaggle.com/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets
[3]Available at: kaggle.com/joparga3/in-depth-skewed-data-classif-93-recall-acc-now

7. Future works

During this work, we observed some questions that may be relevant and can be investigated in the future.

- **Other datasets:**: To better evaluate the method it would be interesting to apply it to other datasets to compare performances. Especially in datasets that have no previous treatment in the data.
- **Other architectures**: Test other architectures for the auto encoder and better analyze the results of the latent vector of characteristics.
- **Oversampling**: Create another study on how to improve performance if we implement oversampling instead of undersampling.