Credit Card Detection

26/04/2020

Guilherme Righetto

Overview

In this work, all the steps proposed to create a binary model will be described in order to detect fraud in credit card transactions. The idea is to try to create something new and compare different results obtained by each stage of the process, trying to identify whether the technique used in that stage really brings any advantage or not.

Introduction

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, as frauds are adaptable and more and more the fraudster tries to simulate a non-fraudulent transaction, and since most businesses need to scale this evaluation of fraudulent transaction or not, it is usually used Machine Learning to try to solve this problem, and for that reason the proposal of this work will be developed.

I currently work with credit card fraud detection, and consequently understand about some difficulties and goals that a classifier has. In this way, the implementation of this work has as a fundamental basis the use of few samples and of not doing any work in relation to the feature engineer and consequently iterating over the results faster, as well as evaluating some techniques.

Goals

1. Create a binary classification model
2. Compare the final results of the steps
3. Identify the best set of techniques to create a final solution

Dataset

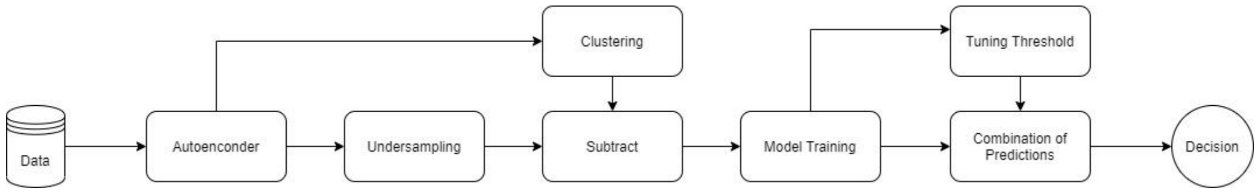
This work will use a dataset published in the kaggle of fraudulent and non-fraudulent transactions carried out with a credit card. The dataset was created and analyzed during a collaboration between Worldline and Machine Learning Group at Université Libre de Bruxelles.

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).

Link: https://www.kaggle.com/mlg-ulb/creditcardfraud

Method

The suggested proposal to solve the problem is to create a binary classifier using the method described below:



Data

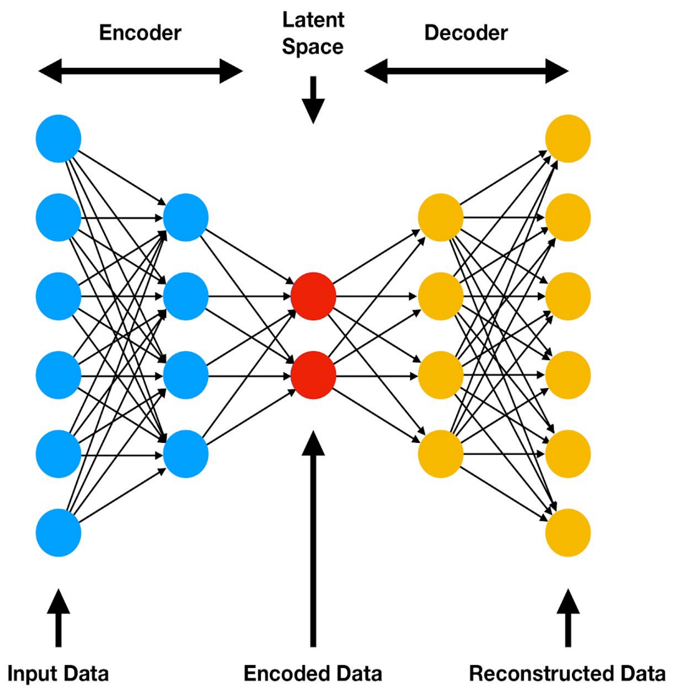
In this step we will divide the dataset between three subsets called train, dev and test. The divided proportions will be 80%, 10% and 10% respectively, in which the training subset will be balanced the number of samples maintaining the same proportion of fraudulent and non-fraudulent samples. The dev subset will be used to perform any optimization, such as tuning hyperparameters and threshold. Meanwhile, the test subset will be used for the final evaluation of the method compared to the dev.

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 image below, performing the training and validation of the network only with non-fraudulent samples.

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.



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.

Clustering

Clustering will be used in the non-fraudulent samples in which they were used to train the autoencoder model. The purpose of this step is to select a sample from each of the N clusters created by the k-means algorithm and subsequently subtract the characteristic vector from each sample in the dataset with the selected samples from each cluster. The number of clusters created will be analyzed by the elbow method and the justification for this approach will be better described in the next step.

Subtract

In this step we will subtract the characteristic vector from each sample with the samples found in the previous step. As only 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.

Model Training

In this step, some sorting algorithm will be chosen, probably having a robust probability calibration so as not to interfere with the results of the next step. In addition, in this step we will perform the tuning of the hyperparameters using a Bayesian method in order to extract the best performance for solving the problem.

Tuning Threshold

The objective of this step is to analyze the predicted probabilities of each class and try to optimize the cut-off threshold to define a fraudulent class or not. In this step, a technique called Youden Index will be used, which uses the False Positive Rate and True Positive Rate metrics to calculate the ideal threshold.

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

Decision

In this step, we will perform the calculation of the metrics described in the next chapters to evaluate the performance of the proposed method. It is worth noting that we will evaluate several paths of the method, for example: Do not perform the subtract step and evaluate/compare the result obtained.

Benchmark

The works selected as a comparison were extracted from kernels performed by Kaggle users using the mentioned dataset. It is worth noting that in some results the number of samples used was different.

1. Kernel 1: https://www.kaggle.com/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets
2. Kernel 2: https://www.kaggle.com/joparga3/in-depth-skewed-data-classif-93-recall-acc-now

Kernel	F1	Recall	Precision	FPR	FNR
1	74,29%	66.33%	84.42%	0.0%	33.67%
2	72.80%	61.90%	88.35%	0.0%	38.10%

Metrics

In this work, several metrics will be used to evaluate the model, since the dataset used has unbalanced classes.

I. F1

F1 = 2*(Recall * Precision) / (Recall + Precision)

II. Recall

Recall = TP/TP+FN

III. Precision

Precision = TP/TP+FP

IV. False Positive Rate

False Positive Rate = FP/FP+TN

V. False Negative Rate

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.

Project Design

Finally, the project will be developed in the python3 language and the main libraries used will be: scikit-learn, optuna, pandas and keras or pytorch. After the project is finalized and corrected, the idea is to create a simple article on Medium.