Credit Card Fraud Detection using Machine Learning and Neural Networks

**Abstract – *With the rise of online payment credit cards have had a huge rule in our daily life and economy for the past two decades and it is important task for companies to identify fraud and non-fraud transactions. Multiple methods have been suggested for this problem and they each have their own pros and drawbacks. In this paper we will apply machine learning algorithms and artificial neural networks on the real-world dataset that is taken from Kaggle [1]. Our main goal is to detect all fraud cases. Moreover, we will compare the results of different linear, ensemble, voting and other methods from open-source libraries as well as with methods done in previous papers in this field.***

**Keywords** *Credit Card Fraud, Supervised Machine Learning, Artificial Neural Networks, Imbalanced classification.*

**Introduction**

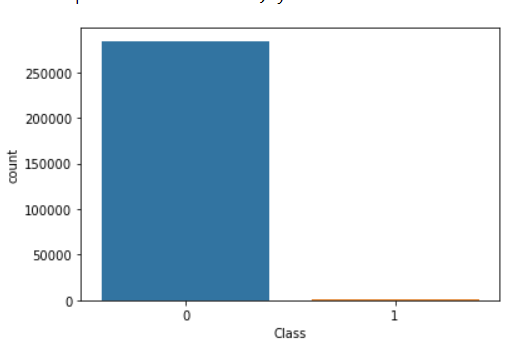
In the modern world, credit cards are important part of our life as people receive their salary, do their shopping, pay their bills with the help of credit cards. Only in one day there are more than 1 billion credit card transactions are made according to The Nilson Report. For the fraudsters who are eager to steal it can be another opportunity. There is plethora of methods which scammers use. Only in 2018, without even presence of card more than 400 million dollars were stolen. Credit card frauds are most common type of identity theft, occurring 41% of all identity theft reports. Moreover, for the most part police cannot investigate on the credit card fraud due to its international nature.

Credit card fraud detection’s goal is to decide if the given transaction is fraudulent or not according to the previous transaction data. Now the challenge in this type of dataset is that, when you want to train a model while measuring the accuracy the results will be higher than 90% even if the model labels all transactions as non-fraud and the reason for that is because these kinds of datasets are highly imbalanced. For example, in the data that we will use only 492 transactions are fraud and 284315 transaction are not fraud. This means roughly 0.17 percent of all transactions.

In this paper I used multiple supervised learning algorithms, deep learning models and compared their ROC\_AUC score, F1-Score, Precision and Accuracy on the real-world dataset.

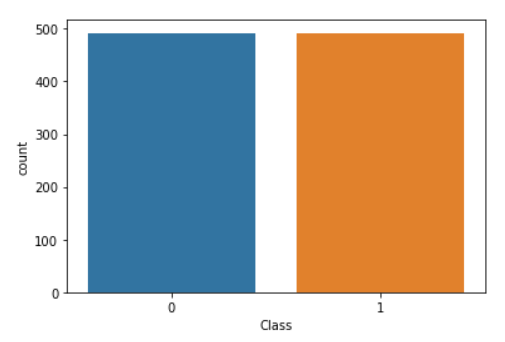
**Materials and methods**

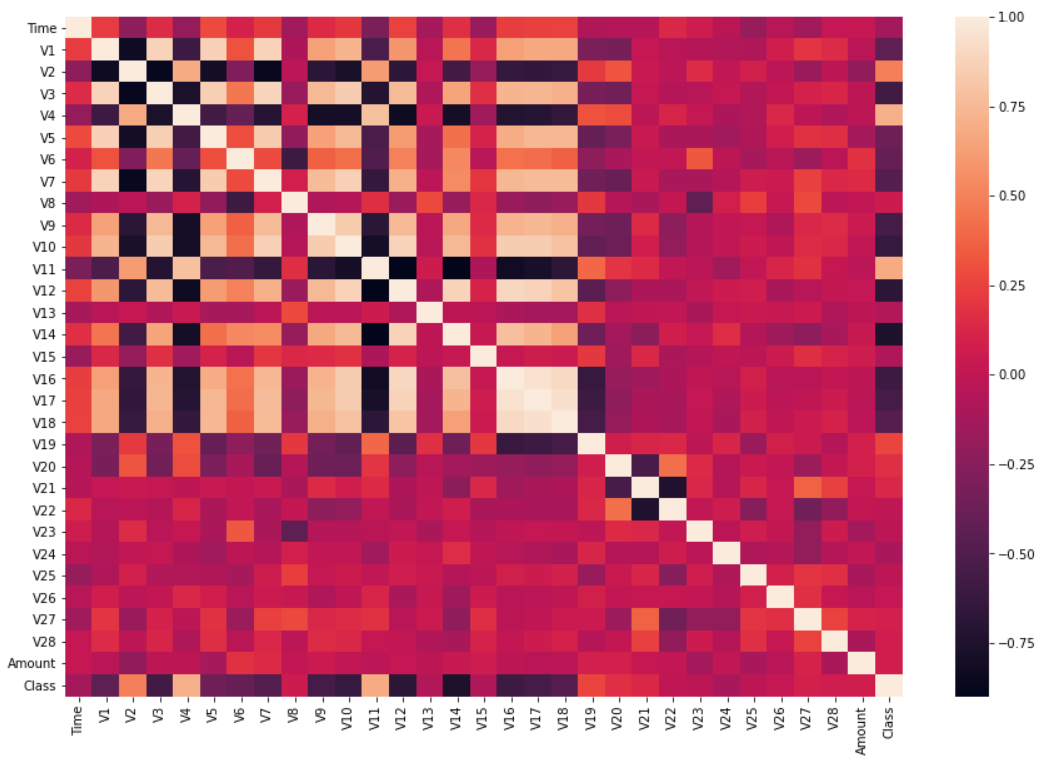
The dataset that we have picked is one of the most famous datasets in Kaggle and it contains transactions made by credit cards in September 2013 by European cardholders. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions and the figure below shows it visually:

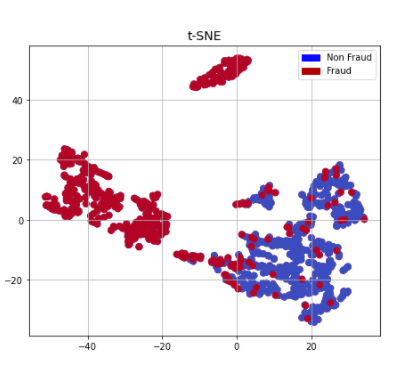
****

If we use **accuracy = (TP+TN)/ (TP+TN+FP+FN***)* to calculate how well our model works this method would not be efficient. For instance, in our case it is enough to label all rows as non-frauds and our accuracy will be more than 99%.

There are few things we could do: Over sampling, under sampling, Generating Synthetic Samples, Using Tree algorithms, using penalized models. Experimenting on these methods we found out that using tree algorithms on under sampled data gave us the best results. Here is how our under sampled data looks like.

****

After under sampling we have left with 492 fraud ad 492 non fraud rows. In the dataset we have total 31 columns. 28 of them is labelled from V1-V28 and there is Time, Amount and Class (fraud or non-fraud) which target column. The correlation between them is like this.  The interesting pattern we need here is in the last row (Class) and we can see that columns V10, V14, V12 have negative correlation with class variable. By using only those three columns and using dimensionality reduction we were able to find some interesting patterns in the data.



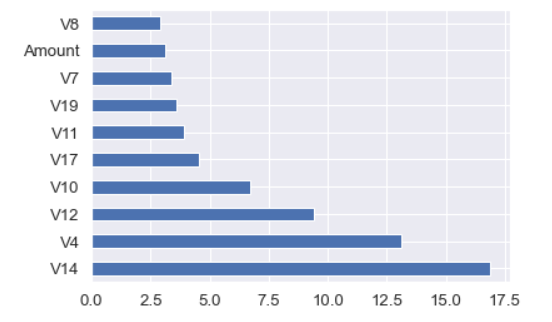
**Results**

Based on the experiments we decided to divide data to 80-20 train test split ratio. Below in the table you can witness Recall, ROC AUC, F1 scores of different machine learning algorithms combined with stratified cross validations gave.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | ROC AUC | Recall Score | F1 score |
| Gradient Boosting | 0.986 | 0.960 | 0.93 |
| Random Forest | 0.991 | 0.948 | 0.97 |
| Logistic Regression | 0.976 | 0.975 | 0.68 |
| Logistic Regression with Bagging | 0.979 | 0.983 | 0.69 |
| LGBMClassifier | 0.992 | 0.962 | 0.962 |
| CatBoostClassifier | 0.993 | 0.958 | 0.969 |
| TensorFlow model | 0.977 | 0.897 | 0.921 |

From the results above in the table we can conclude Tree Based Algorithms performed better.

CatBoostClassifier has the highest ROC score followed Light GBM and Random Forest. I did not have good results with XGBClassifier and LGB is 7 times faster that it. Logistic Regression had the highest recall but a very low f1 score. Like in ROC Cat boost and Light GBM had highest f1 scores.



When comes to feature selection columns V14, V4, and V12 were three most useful columns.

**Conclusion**

Without knowing the columns real names, it was difficult to perform feature engineering. Selecting less columns was not good for the overall results. So eventually we feed all columns to the model.