Machine Learning Engineer Nanodegree

Capstone Project: Credit Card Fraud Detection

Mena Gabara

10.11.2019

Definition

- ### Project Overview

  Credit card fraud is a wide-ranging term for theft and fraud committed using or involving a payment card, as a fraudulent source of funds in a transaction.The purpose may be to obtain goods without paying, or to obtain unauthorized funds from an account. It is also an adjunct to identity theft. According to the United States Federal Trade Commission, while the rate of identity theft had been holding steady during the mid 2000s, it increased by 21 percent in 2008. However, credit card fraud, that crime which most people associate with ID theft, decreased as a percentage of all ID theft complaints for the sixth year in a row. Although incidences of credit card fraud are limited to about 0.1% of all card transactions, they have resulted in huge financial losses as the fraudulent transactions have been large value transactions. In 1999, out of 12 billion transactions made annually, approximately 10 million—or one out of every 1200 transactions—turned out to be fraudulent. Also, 0.04% (4 out of every 10,000) of all monthly active accounts were fraudulent. It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

  And that is where machine learning comes in handy, such behaviour could be detected and millions of dollars could be saved.

- ### Problem Statement

  The goal here is to detect whether a new transaction is fraud or not. This works by feeding the system with previous data -that has unique and common features like: time, class(fraud, not fraud), amount.. etc - to collect data which would help in accurately classifying new transaction.

It's more important to detect fraud than classifying a normal transaction as fraud.

First, analysis the data and then find a way to balance the data. Then building different supervised model to choose among them the best one with highest score . and Finally evaluation metric techniques will help in comparing between the scores of the implemented model to choose among them the best model with highest score.

- **Metrics**

Hence the goal is to correctly classify fraud transactions. Classifying some fraud cases as normal seems more crutual than classifying a normal case as a fraud.

TP = True Positive. Fraud transactions are predicted as fraud.

TN = True Negative. Normal transactions are predictd as normal.

FP = False Positive. Normal transactions are predicted as fraud.

FN = False Negative. Fraud transactions are predicted as normal.

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F1 &= \frac{2 \times precision \times recall}{precision + recall} \\ accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \end{aligned}$$

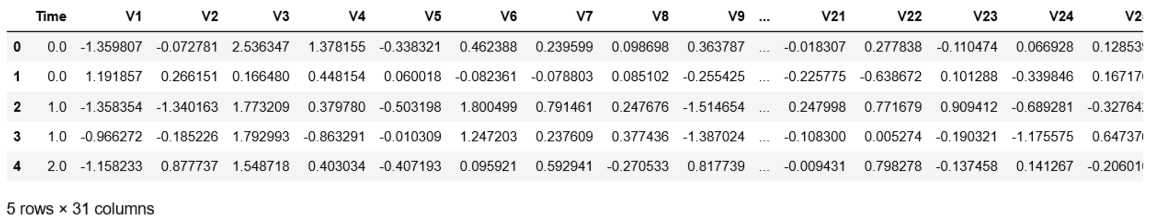
Hence Recall-Precision approach does not care about true negatives. so using these two would be much better in our case here, as there are more normal cases than fraud cases. and instead of using two values for evaluating AUPRC will be used to calculate the accuracy of the score by plotting precision against recall. The area under precision-recall curve will measure and evaluate the performance of the model, given that the closer the score to 1 the more accurate the model is.

Analysis

- **Data Exploration**

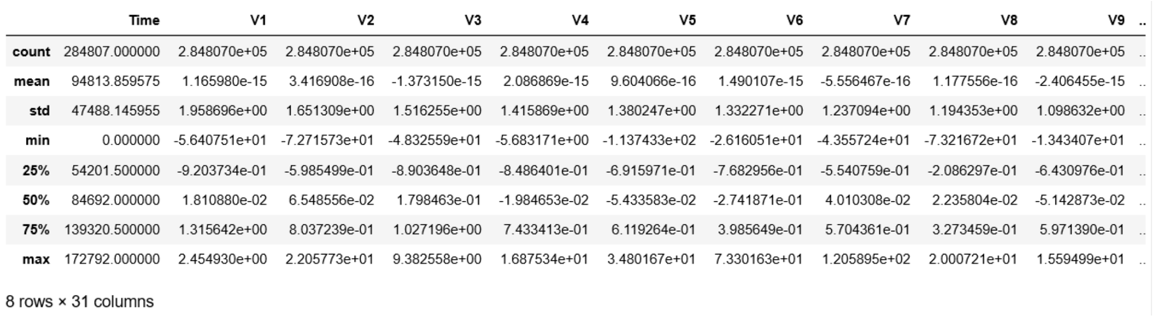
The dataset used is the one provided by kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud/data#
The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation.
Features V1, V2, ... V28 are the principal components obtained with PCA,

Below is a sample of the data and data statistics respectively:



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601

5 rows × 31 columns



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	..
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	..
mean	94813.859575	1.165980e-15	3.416908e-16	-1.373150e-15	2.086869e-15	9.604066e-16	1.490107e-15	-5.556467e-16	1.177556e-16	-2.406455e-15	..
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	..
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	..
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	..
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	..
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	..
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	..

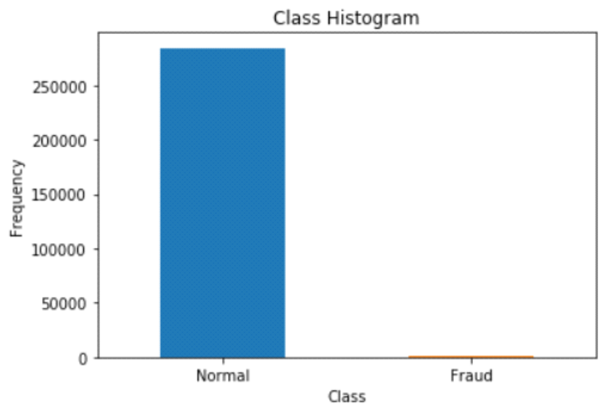
8 rows × 31 columns

And as seen from the data sample, the only features which have not been transformed with PCA are 'Time' and 'Amount'.
Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.
The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

- **Exploratory Visualization**

The below graph shows the number of normal and fraud transactions. And as seen the ratio of fraud to normal is way too small. the imbalance of the data may make the model falsy trained. So finding a way to skew the data is highly needed to have a well-trained model.

```
Normal transactions: 284315
Fraud transactions: 492
```



- **Algorithms and Techniques**

Due to the nature of the problem (a binary classification problem). the choice was to select among different algorithms which suits best with the stated problem and putting into consideration the computational time. So decision tree and logistic regression were chosen to compare between them the best model.

Decision Tree: It is a supervised machine learning technique for inducing a decision tree from training data. It is a predictive model which is a mapping from observations about an item to conclusions about its target value. In the tree structures, leaves represent classifications (also referred to as labels), nonleaf nodes are features, and branches represent conjunctions of features that lead to the classifications.

Genrally, the following parameters will be checked their best value to feed the algorithm with:

- criterion: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
- max_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split

samples.

- min_samples_leaf: The minimum number of samples required to be at a leaf node.

- class_weight: Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

Logistic Regression: It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

- penalty: Used to specify the norm used in the penalization. The ‘newton-cg’, ‘sag’ and ‘lbfgs’ solvers support only l2 penalties. ‘elasticnet’ is only supported by the ‘saga’ solver. If ‘none’ (not supported by the liblinear solver), no regularization is applied.

- class_weight: : Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

- C: Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

After removing outliers and choosing the best parameters using GridSearchCV, and splitting the data into test and train. trainning data will fit into the model and score will be calculated.

- ## Benchmark

In order to be able to mesaure how accurate the model is, some benchmark models need to be taken into consideration and after searching the internet in what other have accomplished especially in kaggleAUPRC should not be less than 80%. and since the case is very sensetive a lower score can not be taken into consideration as a good model.

And by comparing our models with these ones for example:

https://www.kaggle.com/surentharbe/credit-card-fraud-detection-using-svm-100-auprc
https://www.kaggle.com/shaliniyaramada/calculating-auc-and-auprc-for-credit-card-fraud our model seems to perform well .

Methodology

- ### Data Preprocessing

First, Column "Amount" was normalized to have values from -1 to 1; since it was not in line with other features.
Then, column "Time" was dropped to reduce the noise.
Last thing was balancing the data, following the oversampling techniques to have an equal size of fraud and normal values which would help in training the model.

- ### Implementation

After balancing the data i added both lists of fraud and normal to new_data list with shuflle option to make it more reliable in training phase. Both normal data and balanced data were splitted into train and test, where the train data size was 30% of the data. Then it was the time for determine which algorithm to use for normal data before balancing. i chose decision tree and logistic regression, but before using them i had to know which paramter i should use, so for decision tree the options were between:

```
param_grid = {

      "criterion": ["gini", "entropy"],

      "max_depth": list(range(2,6,1)),

      "min_samples_leaf": list(range(5,9,1)),

      'class_weight' : ['balanced', None]

      }
```

and for logistic regression the choices were among:

```
param_grid = {

                 'penalty' : ['l1','l2'],

                 'class_weight' : ['balanced', None],
'C' : [0.1, 1, 10, 100]

}
```

And by using GridSearchCV and put into consideration the scoring techniques the would be used latter in the code scoring value was make_scorer(average_precision_score) . and fir each train value into each model and get the best parameter set which was:

(criterion='gini', max_depth=5, min_samples_leaf=8) for decision tree, and (C= 10, penalty= 'l1') for logisticc regression. After that i needed to know how each model do as per the score for recall, precision and f1score.

Both models scores 100%. which means overfitting. So i decided to re-follow the previous steps but with the balanced dataset this time.

First the best parameters was:

(class_weight='balanced', criterion='gini', max_depth=5,min_samples_leaf=6) for decision tree, and (C=.1,   penalty='l1') for logistic regression. This time the score was 97% for decision tree and 95% for logistic regression. Both models performs very well but i take the model with the highest score to evaluate which is decision tree.

- **Refinement**

First i tried to implement the model with the normal data, then i found out the overfitting issue. so after a lot of reading i decided to implement an oversampling method to balance the data. then i tried it again with decision tree and the score dowgraded to 97% which was high but not overfitting.

Second thing was my attempt to use random forest, but it was taking to much time to find the parameters so i chose decision tree instead alongside with logistic regression.

Then was building each model with random parameters which result in low score. so i decided to use grid search to come up with best parameters to use while building each algorithm.

which was:

```
param_grid = {
    "criterion": ["gini", "entropy"],
    "max_depth": list(range(2,6,1)),

    "min_samples_leaf": list(range(5,9,1)),

    'class_weight' : ['balanced', None]

    }
```

for decision tree and for logistic regression the choices were among:

```
param_grid = {

            'penalty' : ['l1','l2'],

            'class_weight' : ['balanced', None],

            'C' : [0.1, 1, 10, 100]

            }
```

Also the scoring value i added as a last modification to thr grid search as my models was still getting lower score than i expected.

Results

- ### Model Evaluation and Validation

The chosen model was decision tree with the balanced data and parameters: (class_weight='balanced', criterion='gini', max_depth=5,min_samples_leaf=6).

The model achieved an average score of 97% for precision, recall and f1score.

and scored 95% for AUPRC and 95% for recall (detecting fraud data). So It can be said that this model is capable of detecting fraud transaction and could be a base for a more enhanced model. Also the training and testing score is near to each other. The model is robust and can be used in detecting new data.

- ### Justification

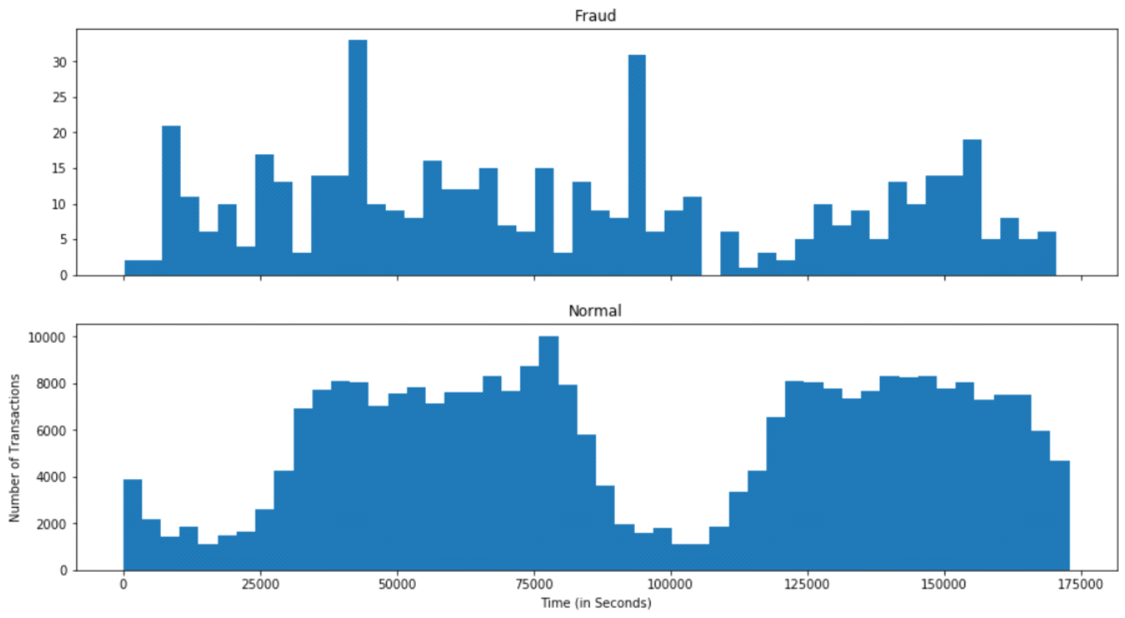
Comparing the AUPRC which scored (95%) with the benchmark models that scored (100% and 97%). we can see that the model is successfully started to detect fraud transactions and near to the score of benchmark models. so with more anhancments it could achieve better result.

And if we see in the area of the problem, this is considered a good precentage as well.

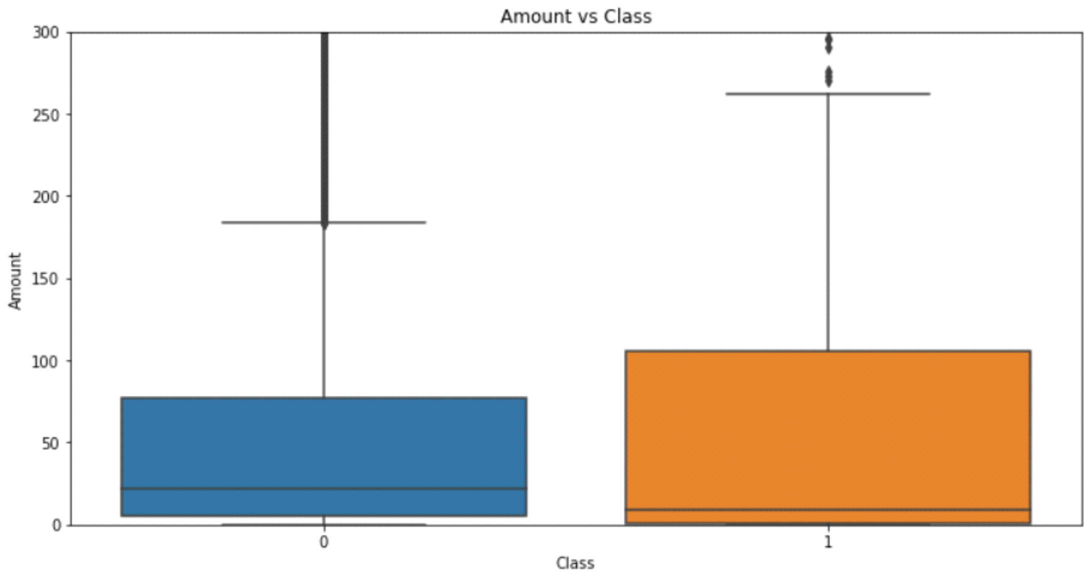
Conclusion

- **Free-Form Visualization**

As shown in the below graph at time near 4500 seconds the number of fraud transaction is higher than normal transaction, also at time near 9500 the normal transaction is too low and fraud occured alot at same period.



In second graph, it is noted that fraud happens more in amount compared to normal transactions.



- **Reflection**

In this project i tried to make a predicted software that could to detect fraud transactions in credit card, the problem was that the portion of fraud data is almost nothing compared to normal transaction, but i run into the following phase to find the best technique:

- Load the data from Kaggle.
- Exploratory Data Analysis.
- Data Pre-processing which includes dropping outliers(Time), normalizing the denormalized data (normalize amount column values to range from -1 to 1 to be like the rest of features) and data balancing(have an equal number of rows for both normal and fraud transactions using oversampling) .
- Split the data into train and test (this goes two times one for the normal dataset and once with balanced data).
- Build different supervised learning models(decision tree and logistic regression) and evaluate the best paramaters for each one by fitting the training data into it and evaluate the best.
- plotting the score for selected the algorithm using confusion metrics to see how the algorithm do.

- **Improvement**

There are many things to do to obtain better result and improve our final model like:

- Try other data balancing techniques like SMOTE.
- Try more complex algorithms.
- Add more parameters to param_grid to grid search to choose from which could make a better model.