Fraud Detection Dataset Report

Dr. Kherad Pisheh

Amirreza Safehian
Farzam Taghipour

Introduction

Fraud detection dataset containing 284807 records of financial transactions presents up 30 features for each transaction and defines that each transaction is fraud or not fraud based on those 30 features.
With a simple quick look at dataset, First thing that collects attention is that this dataset contains only 492 fraud records, So it's an skewed dataset.
To deal with skewness, we have multiple choices. First but not least we can get more data if it's available and of course effort able. Second we can split non-fraud transactions with size of fraud ones and combine them together to make a vary small (about 1000 record) sub-dataset which is balanced and is not suffering from skewness. Third, we can change the way we look at data and alternate our approach and use methods like precision and recall. Obviously first approach is not valid for us right now, so we get to second one and gets down to create a sub-dataset of given dataset.

Solution:

First thing we should take care is that we should shuffle data so that our algorithm doesn't get use to order of data. Then we create our sub-dataset as shown below:

```
fraud_transactions = df[df.Class == 1]
non_fraud_transactions = df[df.Class == 0][:508]
Data = np.concatenate((fraud_transactions_array, non_fraud))
```

Now we have our desirable data variable that contains 1000 records. Next step is to try to classify these data with usual linear and non-linear algorithms like: Linear Regression, Logistic Regression, Linear SVM, SVM with kernel and measure how well they perform. Next step is to create a sequential neural network and compare these algorithms performance.

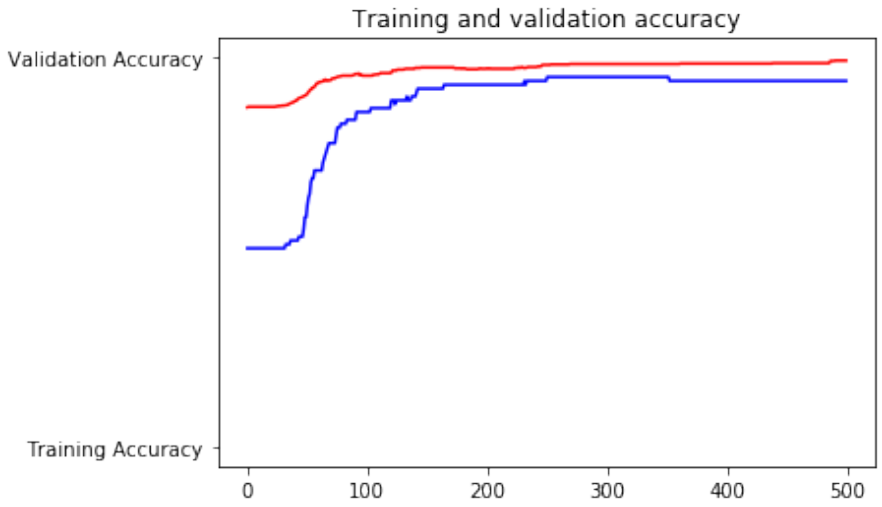
With defining each classifier with appropriate parameters, we got to these results:

```
[Linear Regression] --> Accuracy over validation set: 0.66
[Logistic Regression] --> Accuracy over validation set: 0.93
[Linear SVC] --> Accuracy over validation set: 0.81
[SVC with BRF kernel] --> Accuracy over validation set: 0.87
```

As shown above, according to the fact that this dataset is binary, a logistic regression model performs better that other approaches, even better than SVM and Linear SVM, this shows that our dataset is not linearly separate able.

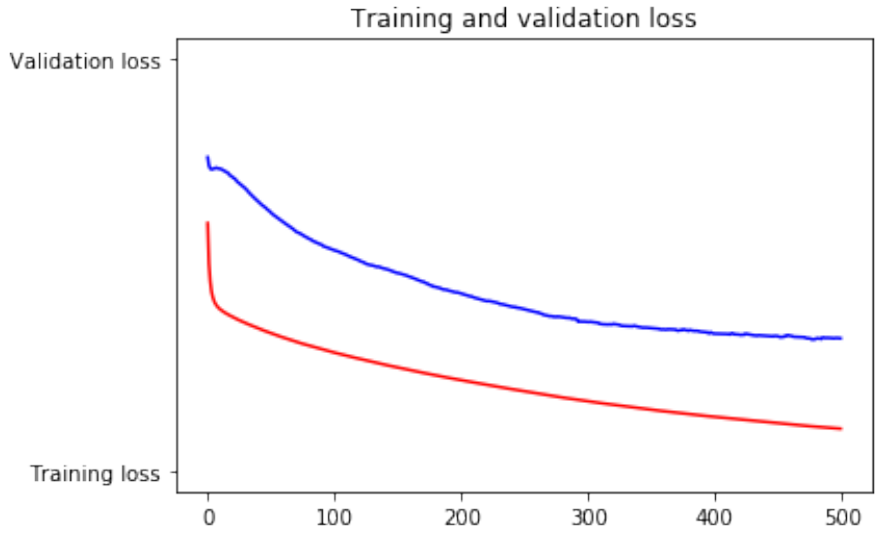
After these approaches we constructed a sequential model and used 3 fully connected layers with 16, 32, 8 neurons. At first try we tried SGD optimizer but the result wasn't better than regular logistic regression. SO we switched to mush advanced optimizers like RMSProp and Adam. For each optimizer model trained for 500 epochs and learning rate of 0.005, Let's see the differences:

SGD Optimizer accuracy (500 epochs, learning rate = 0.005)



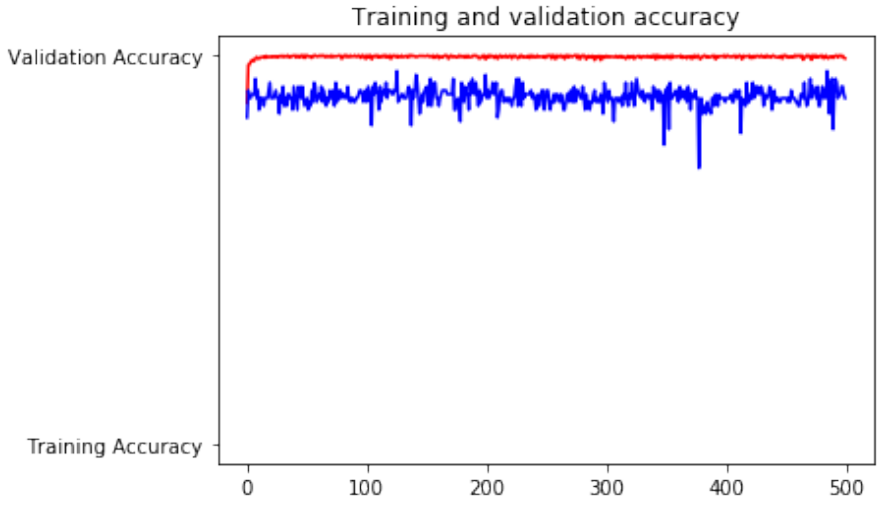
***Red* line: training accuracy**
***Blue* line: validation accuracy**

SGD Optimizer loss (500 epochs, learning rate = 0.005)



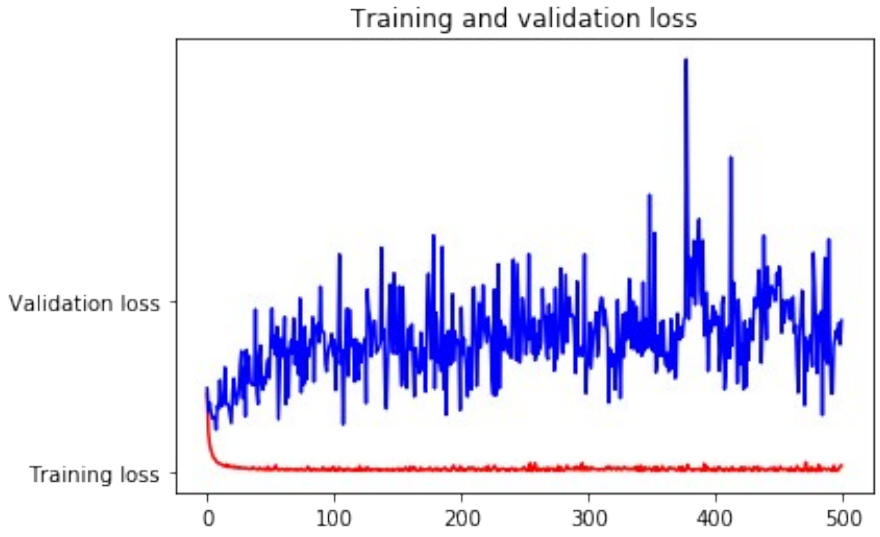
***Red* line: training accuracy**
***Blue* line: validation accuracy**

RMSProp Optimizer accuracy (500 epochs, learning rate = 0.005)



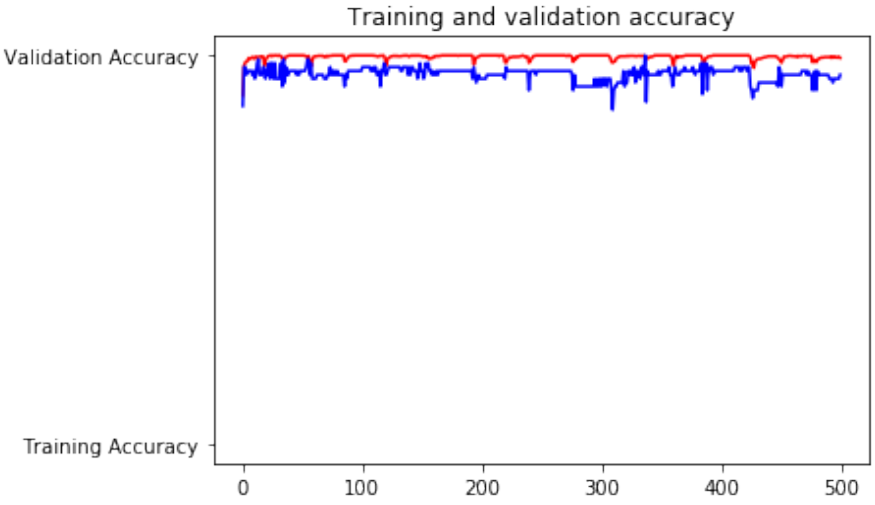
***Red* line: training accuracy**
***Blue* line: validation accuracy**

RMSProp Optimizer loss (500 epochs, learning rate = 0.005)



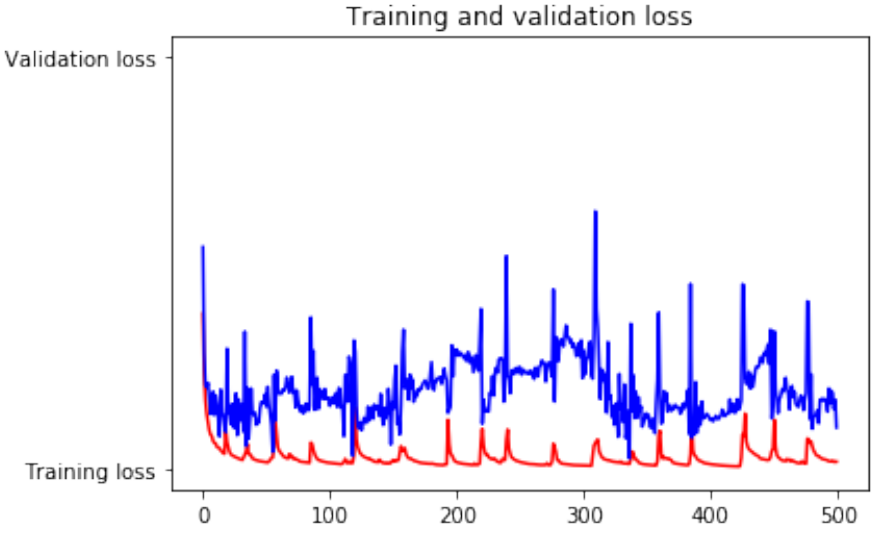
***Red* line: training accuracy**
***Blue* line: validation accuracy**

Adam Optimizer accuracy (500 epochs, learning rate = 0.005)



***Red* line: training accuracy**
***Blue* line: validation accuracy**

Adam Optimizer accuracy (500 epochs, learning rate = 0.005)



***Red* line: training accuracy**
***Blue* line: validation accuracy**

Normalizing Data:

After using data without normalizing and reaching above results, now we are going to see what happens if we normalize our data columns (except class column that represents fraud / not-fraud).
To normalize our dataset, we are going to use below formula:

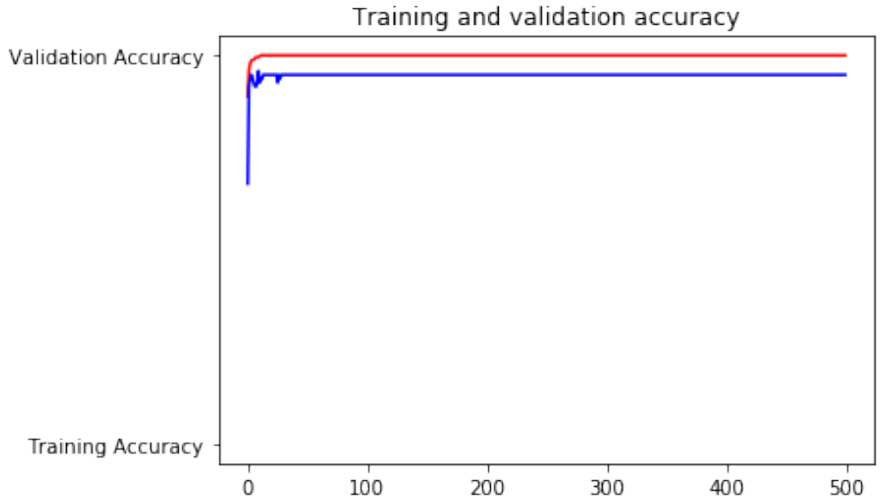
$$z = \frac{x_i - \mu}{\sigma}$$

After normalizing our data frame and re-executing the script, The result changed in as strange pattern! Linear regression and logistic regression's performance didn't get improved but for linear SVM and SVM with kernel, accuracy increased a lot. (from 81 to 95 percentages)

```
[Linear Regression] --> Accuracy over validation set: 0.6
[Logistic Regression] --> Accuracy over validation set: 0.94
[Linear SVC] --> Accuracy over validation set: 0.94
[SVC with BRF kernel] --> Accuracy over validation set: 0.95
```

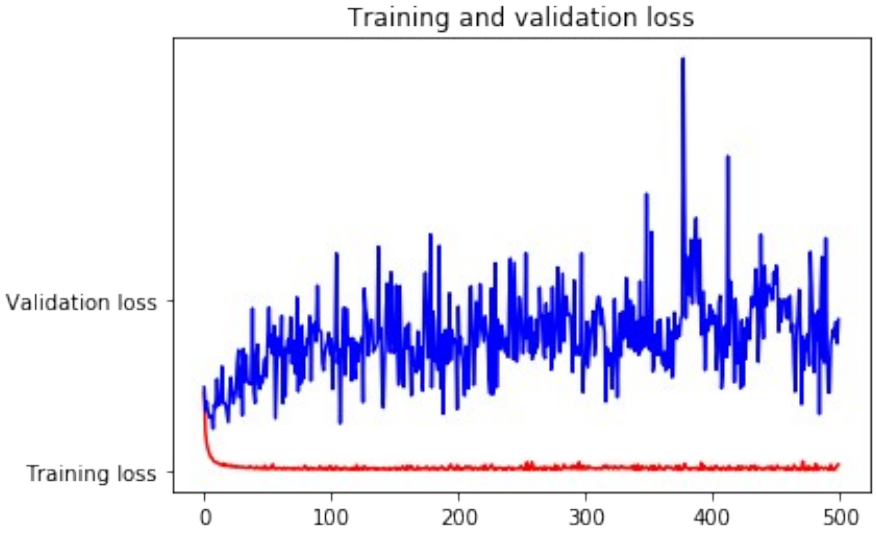
Also neural network with RMSProp optimizer, the accuracy over train set reached 1.0 after few epochs (12 – 14) and validation accuracy reached 98%.

RMSProp Optimizer accuracy (500 epochs, learning rate = 0.005, normalized)



***Red** line: training accuracy*
***Blue** line: validation accuracy*

RMSProp Optimizer loss (500 epochs, learning rate = 0.005, normalized)



Red line: training accuracy
Blue line: validation accuracy

Optimizer	Train Accuracy	Validation accuracy
SGD	0.9577778	0.93
RMSProp	0.9988889	0.96
Adam	0.9411111	0.95

Overall results (Not normalized)

Optimizer	Train Accuracy	Validation accuracy
SGD	0.99666667	0.91
RMSProp	1.0	0.98
Adam	1.0	0.98

Overall results (Normalized)

Conclusion:

According to overall results table we can get into conclusion that normalizing dataset is a big deal and effects the neural network's performance a lot and also using SGD is quiet good and Adam performs good over our dataset although it takes more time to converge to global optimum than others. But RMSProp performs pretty good both on training dataset and validation set. So in this case it is better to use RMSProp.
But, We still use only 1000 records of our huge dataset. To solve this we need to handle the problem we a new approach called ***Precision and Recall***.

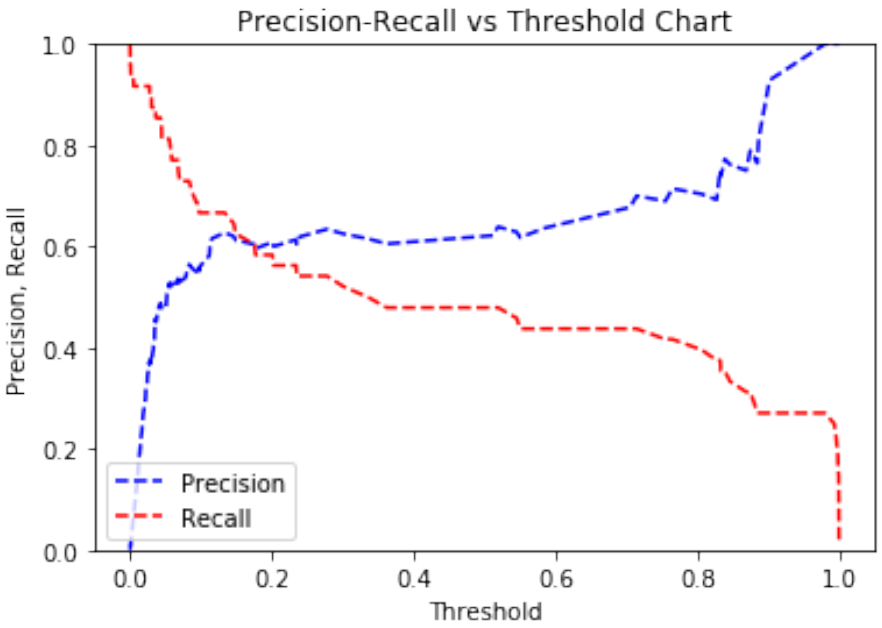
Second Approach
(Using Precision and recall)

For such dataset that class weights are not balanced, also called skewed dataset, even if the accuracy is very very high, we can not be sure that the model performed well enough. For example our dataset has 492 records for non-fraud transactions and 284315 for fraud ones. The ratio of non-fraud to fraud transactions is 0.001730475. So if the model had the accuracy of 0.999 we can not be sure if it did a good job or not. To solve this we use precision , recall and F1 score to determine how good is our prediction. To do so, we are going to use our Logistic Regression model and define a threshold variable for its classification. For each prediction it gives us the output of Sigmoid function which is always between 0 and 1. So for classification if the probability of precision is lower than the threshold, it is classified 0 otherwise it will be 1.

After re-declaring X, y (now X, y re-presents the whole dataset) and training our logistic regression model on it it reached 0.9988413328183702 accuracy over test set, But how good it is?

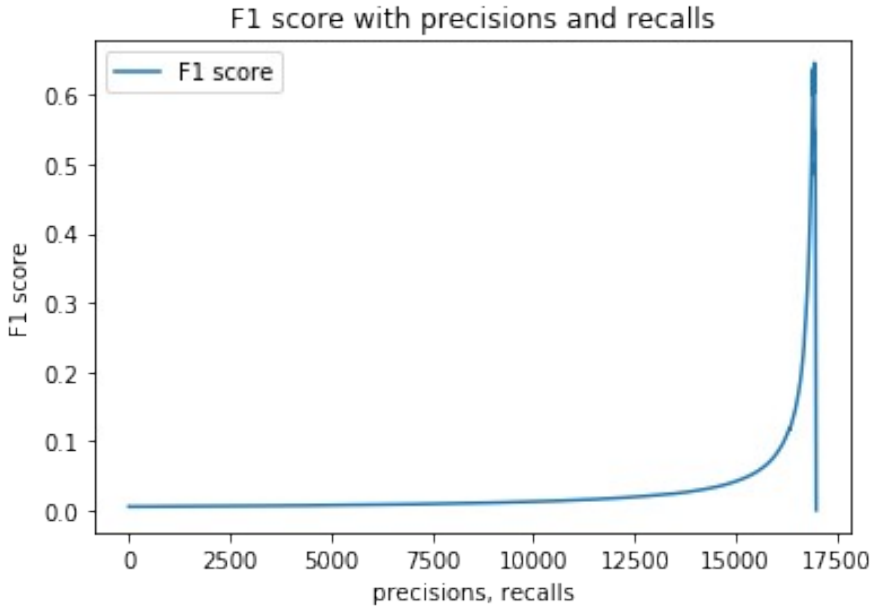
To calculate precision and recall, we need to calculate number of true positive, true negative, false positive and false positives. For that we need to predict probabilities of given x_test dataset via our logistic regression model and use precision_recall_curve() function that returns precision, recall and thresholds. To compute these when the class indicates 1 (Fraud transaction), we feed second element of predicted probabilities to algorithm, shown below:

Precision Recall values over each Threshold



Now, we can compute F1-score for each precision and recall. The following plot demonstrates how F1-score changes over each threshold.

F1-Score over Thresholds



Now that we have F1-score lists over thresholds, we know that maximum values of F1-scores is ***0.6465*** and appropriate precision value is ***0.6275*** and recall value is: ***1.0*** and the threshold is ***0.133614.***
Now we set the threshold for logistic regression classifier to be 0.133614 and we get accuracy of ***0.998665*** with F1-score of ***0.59574469***.

Final conclusion:

In this project we dealt with skewed dataset of fraud in credit cards and tried to make out path trough that. Our first approach was to create a sub-dataset of given original dataset and made predictions. But in this case we actually used only 1000 records of data and that can not be generalized. Then we tried to predict over whole dataset but according to the fact that dataset is skewed, our predictions accuracy is not reliable. To solve that we tried to use precision recall and F1-score concepts and compute them for our predictions. At last we could find best parameters for our logistic regression model and find the best threshold to classify the predicted probabilities.