# Machine Learning Engineer Nanodegree

## Capstone Proposal

Ramakrishnan A
June 2019

## Domain Background

Credit card fraud is a wide-ranging term for theft and fraud committed using or involving a payment card, such as a credit card or debit card, as a fraudulent source of funds in a transaction. 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. [1] The number of frauds and types of scams are increasing and it looks to become higher in the future. So, the need to detect fraudulent transactions is assuming higher importance than other activities.
The Kaggle site https://www.kaggle.com/mlg-ulb/creditcardfraud/kernels has a variety of solutions involving Deep Learning and Machine Learning techniques.

## Problem Statement

The objective of this model will be to identify whether a transaction is fraudulent or genuine. This should happen during the time the transactions is happening. The commonly used techniques for this problem, given that it involves categorization are Support Vector Machines (SVM) or Random Forest Classifier or Decision Trees. Every transaction has a set of data points that describe it and is classified as 'Fraud (1)' or 'Normal (0)'.

I am also curious about this problem as I want to try using Neural Networks instead of ML techniques and measure the performance.

## Datasets and Inputs

I started working on this problem after I found the dataset in kaggle [3]. (The next paragraph is copy-paste from Kaggle)

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. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, 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-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

## Solution Statement

I would like to try using DL techniques for predicting whether a transaction is fraudulent or not. I would like to build a Sequential Model with Relu and Sigmoid activation functions as they mimic a classification problem the best. I intend to start with a simple neural network which takes in all the inputs and then move to a neural network where I manage the data (remove unnecessary variables etc) and compare the results.
Then I intend to compare this with the RandomForest classifier model that I will build to begin with and compare results.
I expect the imbalance of data available for fraudulent and genuine transactions to have a higher effect in the neural network than in the Random Forest classifier.

## Benchmark Model

I will build a model with Randomforest to compare the results with the DL Model that I will use. I will split the training/testing set at 0.8.

```
We will also look at
Accuracy
Error_rate
Specificity
Sensitivity
```

For the randomForest model and the DL Models, which can be derived from the confusion matrix values.

I would also like to compare with 2 models:
https://www.kaggle.com/currie32/predicting-fraud-with-tensorflow/ - Built using tensorflow with an accuracy of slightly more than 98%
https://www.kaggle.com/yuridias/credit-card-fraud-detection-knn-naive-bayes - Using Naive Bayes and KNN; Accuracy is around 98 & 99% respectively

The results from the first model are as follows:

```
____________________
Epoch: 0 Acc = 0.98181 Cost = 76205.75000 Valid_Acc = 0.98244 Valid_Cost =
8314.01758
Epoch: 1 Acc = 0.98349 Cost = 66124.74219 Valid_Acc = 0.98543 Valid_Cost =
7283.07812
Epoch: 2 Acc = 0.98232 Cost = 62798.66016 Valid_Acc = 0.98402 Valid_Cost =
7260.79736
Epoch: 3 Acc = 0.98372 Cost = 62235.78906 Valid_Acc = 0.98599 Valid_Cost =
7295.44336
Epoch: 4 Acc = 0.98558 Cost = 60142.89062 Valid_Acc = 0.98754 Valid_Cost =
7369.91943

_________________________
The results from the second model are as follows:
KNN:
## Confusion Matrix and Statistics
##
##            Reference
## Prediction     0     1
##          0 14212    28
##          1     0     0
```

```
##
##                Accuracy : 0.998
##                  95% CI : (0.9972, 0.9987)
```

Naive Bayes:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13894     5
##          1   318    23
##
##                Accuracy : 0.9773
##                  95% CI : (0.9747, 0.9797)
```

## Evaluation Metrics

As mentioned in the comments, I will use ROC AUC score as the metric to compare the models.
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

We will also look at
Accuracy
Error_rate
Specificity
Sensitivity
For the DL Model.

## Project Design

The project will follow the steps mentioned below:

1. Exploratory analysis to understand the data;
    a. Preprocessing of data
        i. Look for any missing values and use mean as the replacement, if required
        ii. Drop any data that might be unnecessary or shows collinearity
        iii. Normalize the data (especially the amount data)
        iv. Probably drop the time column as we don't have a reference of the starting time
2. Train the neural network with the necessary parameters;
    a. Build using the Sequential Model
    b. Initially start with 16, 64, 256, 128 & 32 units with relu as the activation for all layers except the last one. The last layer will use Sigmoid.
    c. BatchNormalization and Dropouts will be added as necessary.
3. Will also build a Random Forest Model for comparison
    a. Estimators will be 100
    b. Beyond that I will use all the default parameters.

4. Evaluation and compare with the benchmark models.
    a. Plot the AUC Curves and compare the results

## References

[1] https://en.wikipedia.org/wiki/Credit_card_fraud
[2] https://pdfs.semanticscholar.org/0419/c275f05841d87ab9a4c9767a4f997b61a50e.pdf
[3] https://www.kaggle.com/mlg-ulb/creditcardfraud
[4] https://www.kaggle.com/yuridias/credit-card-fraud-detection-knn-naive-bayes