Machine Learning Engineer Nanodegree

Capstone Project: Credit Card Fraud Detection Model

Femi Ogunbode

September 16, 2017.

I. Definition

Project Overview

Credit card fraud is the use of a payment card, either debit or credit obtained fraudulently or illegally to carry out fraudulent transactions.

Credit card fraud is a problem that has been plaguing financial industries for years, for example in my country Nigeria, ATM's (Debit Card) had the highest percentage of fraud volume in 2016 with 49.6% which accounts for half of all fraudulent transactions on electronic payment channels.[1]

Solving this problem could bring about reduced volume of fraudulent transactions in financial industries hereby saving them a huge volume of money lost and this problem can be solved using cutting edge predictive analytics where machine learning algorithms are used to detect fraud patterns and determine future probabilities and trends.

Problem Statement

Given a transaction, it is often difficult know which is fraudulent or genuine

The problem before us is a binary classification problem, because the problem arises from deciding whether or not a transaction is fraudulent and the goal is to create a model that accurately predicts whether a transaction is fraudulent or not.

To solve this problem, a prediction model would be developed using several machine learning algorithms on this dataset to see which will give the best performance which is judged against a pre-defined evaluation metric. These algorithms include Naïve Bayes Classifier, Logistic Regression, Multi-Layer Perceptron and Decision Trees.

[1] The Nigeria Electronic Fraud Forum "A Changing Payments Ecosystem: The Security Challenge" *Annual Report, Pg. 16* (2016)

Since the features made available are a reduced dimension of the original dataset, no other form of reduction would be made going forward. All features provided would be used in building the model.

Metrics

In a binary classification problem such as this, a model classifies examples as either positive (fraudulent) or negative (genuine). The decision made by the model, either positive or negative can be represented in a structure known as **confusion matrix**. This confusion matrix has four elements that define it, contextually they are:

- True Positive (TP) – An example where a transaction is fraudulent and is classified correctly as fraudulent.
- False Positive (FP) – An example where a transaction is genuine and is classified as incorrectly as fraudulent.
- True Negative (TN) – An example where a transaction is fraudulent but is classified incorrectly as genuine.
- False Negative (FN) – An example that is genuine and is classified correctly as fraudulent.

In most binary classification problems, accuracy is used as a metric for evaluating models and is given as follows:

$$\text{Accuracy} = \frac{True\ Positive + True\ Negative}{Total\ Observations}$$

Accuracy is a measure of how well a model accurately predicts true examples as positive and false examples as negative.

Now in situations like this where there is a class imbalance ratio (negative examples outnumber positive examples by a very wide margin), the performance of a model is not reflected by accuracy because there are not enough training points for the positive class (fraudulent transactions).

Given the class imbalance ratio in the dataset, there are other evaluation metrics that take such class imbalance into consideration:

I. Precision
II. Recall

Recall measures the fraction of true positives that are correctly labelled while Precision measures that fraction of examples classified as positive that are truly positive.

Precision and Recall are defined as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

Since Precision & Recall don't account for true negatives (cases where a transaction is fraudulent but classified as genuine by the model) the choice for using these two elements) seems reasonable given that there are lot more cases of genuine transactions than fraudulent.[2]

Finally, a common practice is to plot precision against recall to get a better sense of how a model is performing instead of having to use two values for evaluation, hence **the Area Under the Precision-Recall Curve (AUPRC)** would be used to judge the performance of the model, and the **AUPRC** is gotten by plotting the Precision against Recall and measuring the area under this curve.[3]

The closer to 1 the AUPRC is, the better the model is. A model with AUPRC of 1 implies a perfect classifier.

[2] Jesse Davis & Mark Goadrich "The Relationship Between Precision-Recall and ROC Curves" *Proceedings of the 23rd International Conference on Machine Learning (ICML).* (2006)

[3] Jesse Davis & Mark Goadrich "The Relationship Between Precision-Recall and ROC Curves" *Proceedings of the 23rd International Conference on Machine Learning (ICML).* (2006)

II. Analysis

Data Exploration

The dataset to be used in solving this problem is an anonymized set of credit card transactions labelled as fraudulent or genuine. Due to confidentiality issues, the original features and more background information about the data were not provided.

The dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

The dataset contains 31 numerical features. The first 28 features are labelled V1, V2….V28 and these are principal components obtained with Principal Components Analysis of the raw/original data.

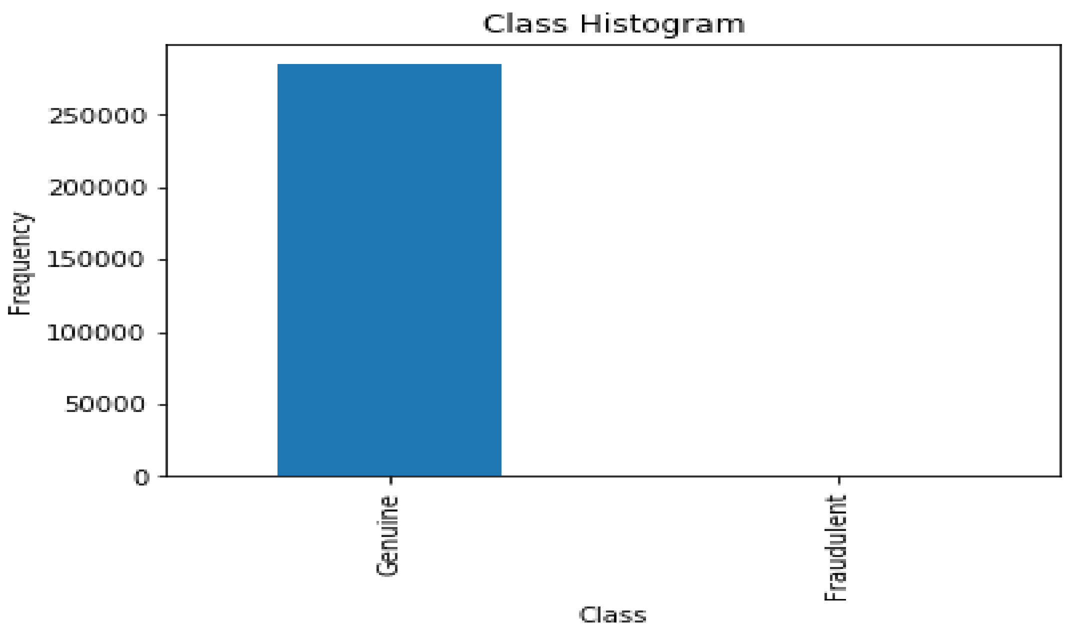
The only features which have not been transformed with PCA are:

I. Time - contains the seconds elapsed between each transaction and the first transaction in the dataset
II. Amount - transaction amount

The last feature to be discussed is 'Class' which is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Exploratory Visualization

Fig 1. A plot showing the ratio of fraudulent transactions to genuine transactions, from this visualization it is obvious that fraudulent cases are grossly misrepresented and may do some harm while training our model.



Algorithms and Techniques

Due to the nature of the problem four machine learning algorithms have been selected, out of these just one would be chosen for the final model.

Logistic Regression

The logistic regression algorithm is a popular algorithm used in classification problems. This algorithm works by learning a numerical weight for each feature and a constant term from our training dataset. Given a new point from the test dataset, it multiplies each feature by the corresponding learnt weight and adds all the product together with the learnt constant term.

The result of this operation is then fit into a logit function which scales it down to a number between 0 & 1. The final number gotten can be interpreted as probabilities with results lesser than 0.5 indicating a prediction of false and results greater than or equals to 0.5 indicating a prediction of true.

The following parameters can be tuned to optimize the logistic regression classifier:

I. Penalty - specifies the kind of regularization option: L1 OR L2
II. C – Inverse of regularization strength, smaller values specify stronger regularization.

Decision Tree

Decision tree is a machine learning algorithm with the goal of predicting a target variable by learning simple decision rules inferred from the data features.

Here the max_depth parameter which specifies the maximum depth of the tree is the major parameter for tuning.

Gaussian Naïve Bayes

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems.

Naïve Bayes stems from the Bayes theorem which involves computing the probability of an event based on the probabilities of certain related events. While the Bayes theorem fails at using more than feature, Naïve Bayes allows for multiple features and the “naïve” in Naïve Bayes implies these features are independent of each other.

Gaussian Naïve Bayes is simply an extension of the Naïve Bayes algorithm to numerical/continuous features, most commonly assuming a Gaussian distribution. The Gaussian Naïve Bayes algorithm works well out of the box and tuning it’s parameter is rarely ever necessary.

Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a neural network, which consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

Its multiple layers and non-linear activation makes it suitable for data that is non linearly separable.

For this project the MLP would have just 1 hidden layer consisting of 21 nodes with a relu activation function. The output layer would consist of just one node with a sigmoid activation function that outputs numbers between 0 & 1 to represent the probability of predicting a value for fraudulent transactions.

Benchmark

For this project, considering the imbalance class ratio, accuracy would not be used to judge the model since it can be misleading. Instead, as a benchmark the model should have an Area Under the Precision-Recall Curve (AUPRC) score of 74% or greater.
This score was gotten from a simple logistic regression classifier I built, this serves as benchmark towards building a better model. The intuition behind using the AUPRC Score has been explained earlier on.

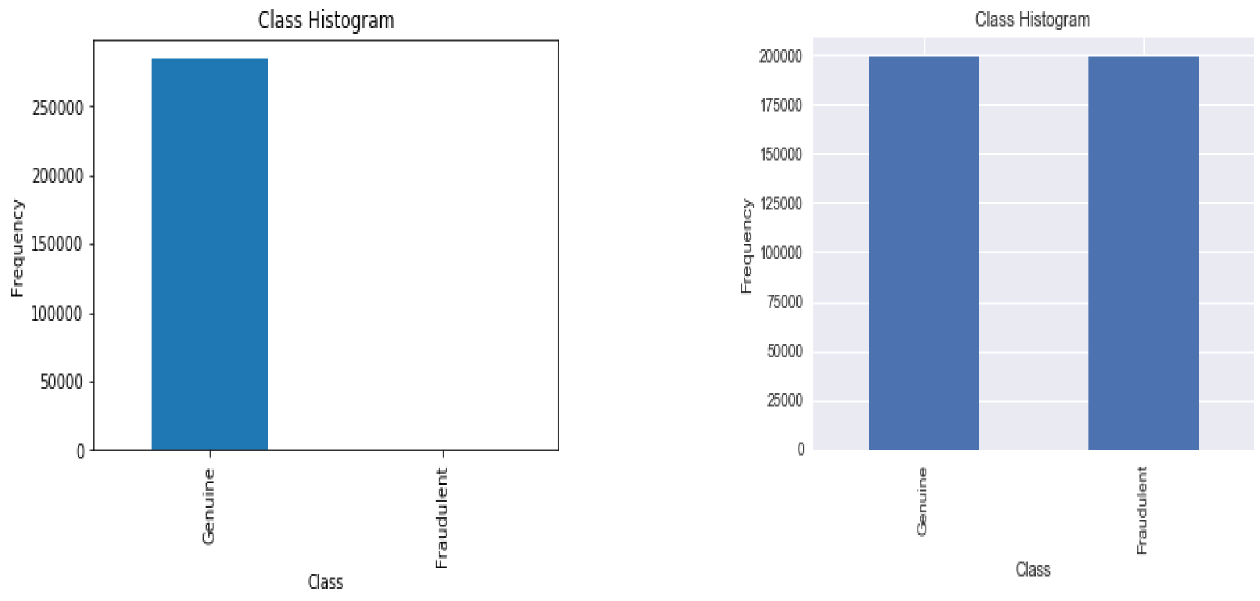
III. Methodology

Data Pre-processing

The first step taken in pre-processing the dataset was **Normalization.** The Normalization procedure was applied only on the **Amount** Feature since it wasn't on the same scale as the other features.

Next, the **SMOTE** algorithm was used to balance out the class ratio. It works by constructing new points from the minority class until it evens out the deficit in the class ratio.

Fig 2. Distribution of the two classes before and after the SMOTE algorithm was used the balance the dataset.



Implementation

The implementation stage involved creating a training and predicting pipeline. This stage involved testing four algorithms to see which best suits the problem. The following algorithms were used;

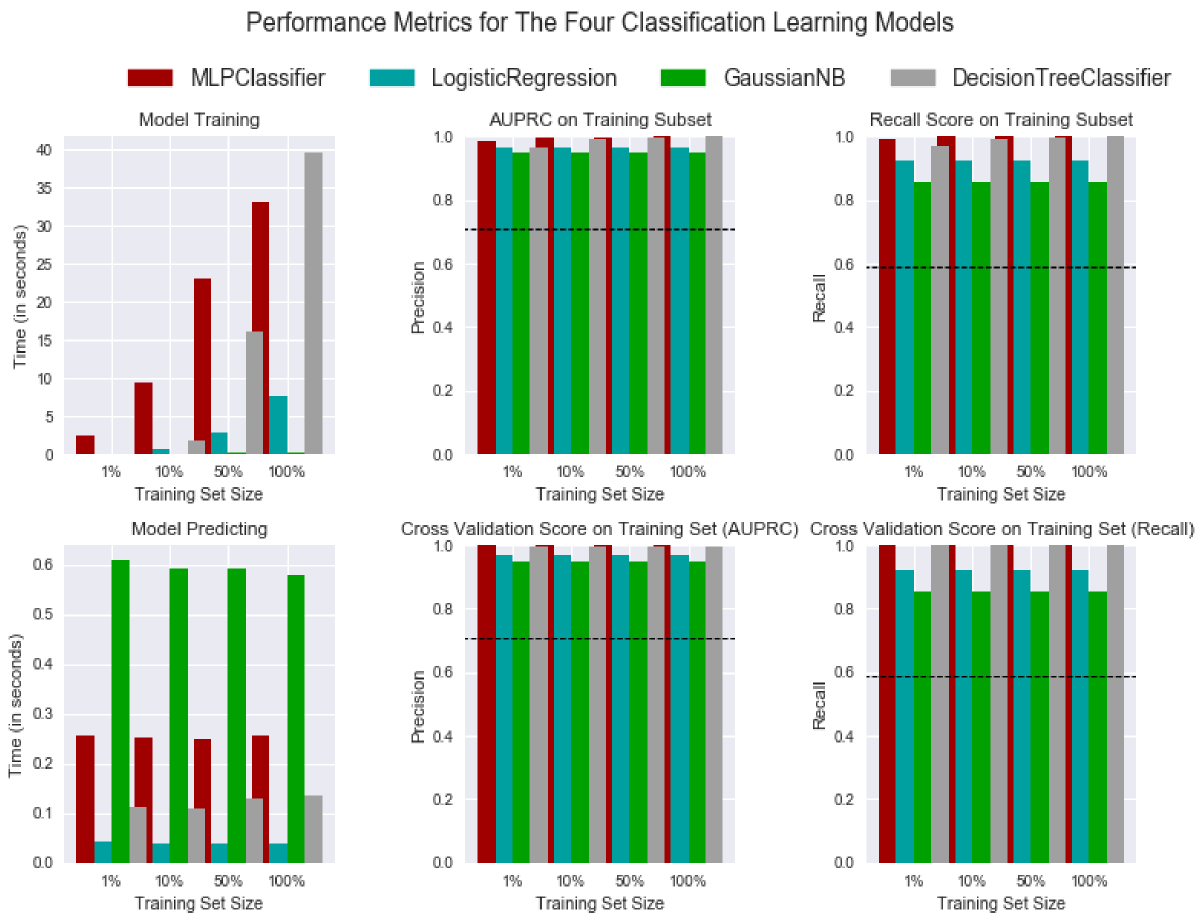
I. Decision Trees
II. Multi-Layer Perceptron
III. Logistic Regression
IV. Gaussian Naïve Bayes

All these binary classification algorithms were initialized. Afterwards the training dataset was broken into four different portions, 1%, 10%, 50% and 100%. The purpose of breaking the dataset into different portions was to track the performance of each algorithm as the training set size increased.

The following metrics were used to measure performance as the data set scaled;

I. Time taken to train the model.
II. Time taken to make predictions on the train and cross validation set.
III. Area Under the Precision Recall Curve (AUPRC) on the train and cross validation set.
IV. Recall score on the train and cross validation set.

Fig 3. Performance of all four algorithms and how they scaled across different portions of the dataset.



Though all four algorithms performed well, the Logistic Regression algorithm was chosen and here’s why. The MLPClassifier and DecisionTreeClassifier performed too well by getting an almost perfect AUPRC and Recall Score on the training set, this indicates that they were too complex for the problem considering that no special parameter was chosen.

This leaves us with the Gaussian Naïve Bayes and Logistic Regression models. Across board the Logistic Regression performed better, hence Logistic Regression was selected.

Refinement

After the Logistic Regression algorithm was chosen, hyper parameter tuning was performed to optimize the model. Grid Search was used to find the optimal parameters for the “C” and “Penalty” values.

Afterwards both the optimized and optimized model returned an AUPRC of 0.9680. This implies that the Area Under Precision Recall Curve of both models was 0.9680 which is a very good score.

IV. Results

Model Evaluation and Validation

During the optimization/refinement stage of the project, the Logistic Regression algorithm was tested on a cross validation set and got a cross validation score of 0.9680 with AUPRC as the evaluation metric.

The parameters of the final model were;

I. C – 1
II. Penalty – L2

To verify the robustness of the model, the model was used to predict fraudulent transactions in the test dataset. The result from this prediction on the test set was an AUPRC of 0.97 and a Recall Score of 0.92.

A recall score of 0.92 implies that the final model predicted 92% of the fraudulent transactions in the test dataset correctly.

Justification

Comparing the final model (AUPRC-0.97) with the benchmark model (AUPRC-0.71), the final model did outperform the benchmark with an increase of 21% in the AUPRC of the final model.

Though In the global context of credit card fraud detection this model might not be a silver bullet but it is definitely a great starting point to a larger project that seeks to tackle credit card fraud detection globally.

V. Conclusion

Free-Form Visualization

Fig 4. Area Under the Precision Recall Curve of the Benchmark Model.

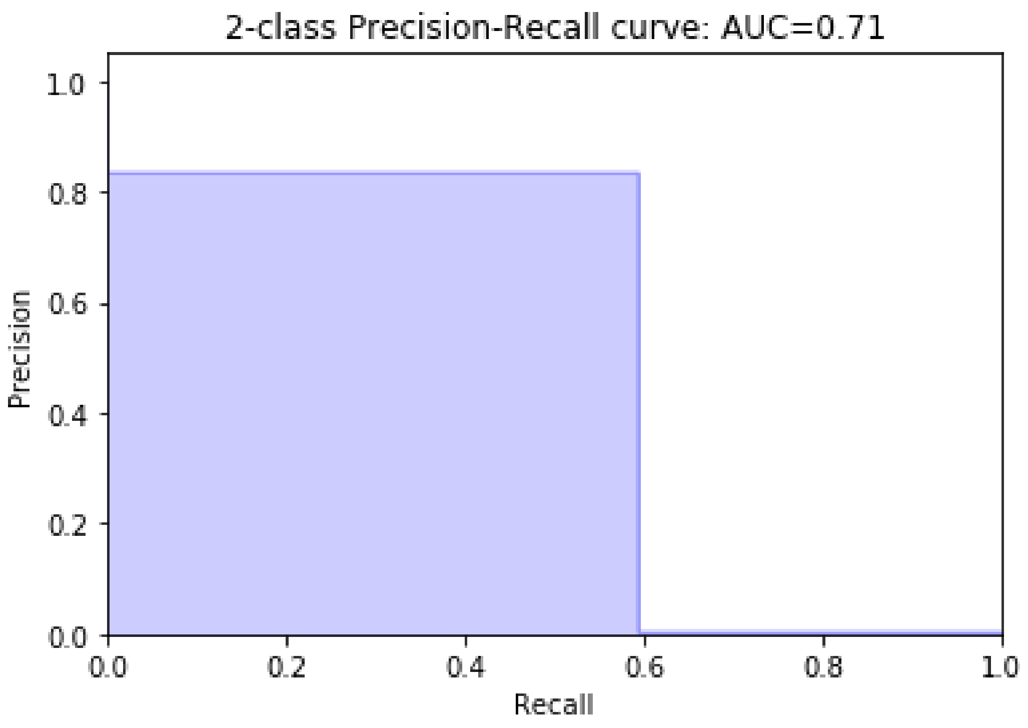


Fig 5. Area Under the Precision Recall Curve of the Final Model

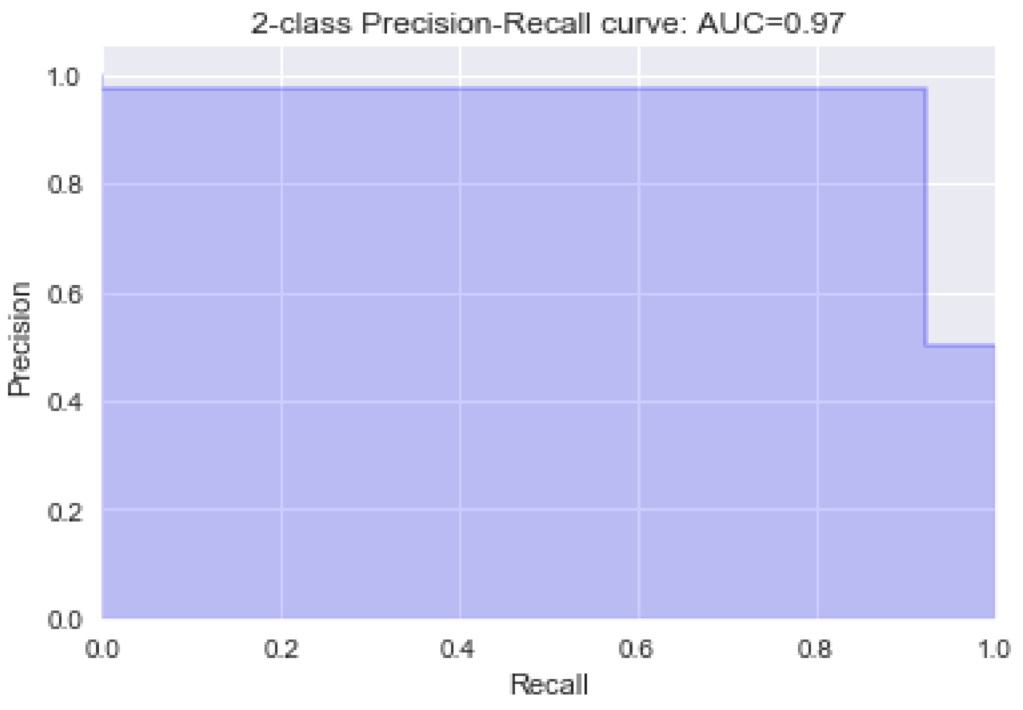
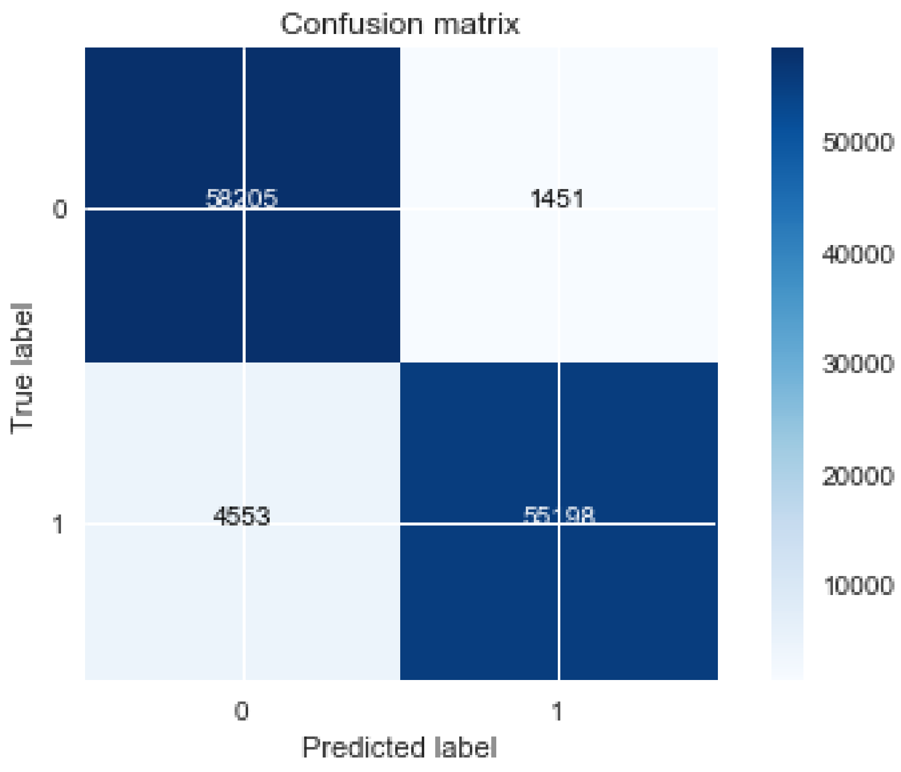


Fig 6. Confusion Matrix for the prediction on the test set.

The final model predicted 58,205 fraudulent transactions correctly, 1,451 fraudulent transactions incorrectly, 55,198 genuine transactions correctly and 4,553 genuine transactions incorrectly.



Reflection

This project can be broken down into phases based on my encounter;

I. Getting the dataset from kaggle - this was the easiest phase for me as the dimension of the data had already been reduced into Principal Components.
II. Exploratory data analysis - much wasn't done concerning this due to the fact that most of the features were principal components of the original data.
III. Model Selection – Choosing out of four algorithms was a bit confusing because of the resulting complexity of the Multi-Layer Perceptron and Decision Tree algorithm which gave an almost perfect score but then this might lead to overfitting.

Also I found the interpretation of this model to be difficult due to the anonymity of features which if was present, might explain the problem better. The most difficult phase for me was picking the right algorithm to use in building the final model.

Finally, embarking on this project has taught me important lessons about problems that may arise from solving a real life problem. Such lessons include, being able to work with data at an abstracted level, there is no perfect algorithm we can only find a very good one that suits the problem and sometimes the simple model is the best option!

Improvement

As stated earlier in the justification section, in the global context of credit card fraud detection this model might not be a silver bullet but it is definitely a great starting point to a larger project that seeks to tackle credit card fraud detection globally.

Improvement on solving the problem of credit card fraud detection would be industry/environment specific but some general rules apply, like a proper and detailed exploratory data analysis of variables that might explain the outcome of a transaction.

Also more data could be collected, which would put the complexity and strength of advanced deep learning techniques to good use in building a more robust and accurate algorithm.