Classifying Fraudulent Transactions on Credit Cards

Ryan Gascoigne-Jones

1 Introduction and Dataset

Credit card companies want to protect their customers. This is to retain their customer base and increase their attractiveness to new customers. A good way to do this is to detect fraudulent transactions and stop them before they are approved. This means the companies don't have to pay customers back, meaning they save more money [2]. The easiest and most efficient way to do this is to develop a model which can classify a transaction as fraudulent or non-fraudulent. The fraudulent transactions can then be handled by asking the cardholder for extra verification of their identity and intent to complete the transaction. This model should detect fraudulent transactions to a high degree. It should prioritise reducing false negatives, as this would leave the company liable to any charges. Whereas false positives would only be a minor inconvenience to customers (as long as it is a small proportion of total transactions) and will not cost the company any money.

The dataset used contains transactions made by credit cards in 2 days during September 2013 by European cardholders [1]. It is comprised of 284,807 anonymised transactions total. There is only a total of 492 fraudulent transactions, which make the dataset highly imbalanced. The entire dataset contains only numerical data and contains a total of 31 features. These include the monetary amount and time of transaction. Along with whether the transaction was fraudulent or not (0 for genuine, 1 for fraudulent). The other 28 features are abstractly named as the actual features could not be published due to data privacy laws.

2 Classification and Data Preprocessing

Classification was chosen as the ML technique as the model needed to predict a binary variable. There was no need to overcomplicate this, as the systems designed to handle this would only class a transaction as genuine or fraudulent; meaning if a continuous value probability of fraud was given, it would likely be classified itself using a threshold. For the purposes of this model, any outliers on the fringes of being classified as fraudulent will be ignored in favour of detecting the majority of fraudulent transactions, which would save the credit card companies the most money.

Firstly, the time and amount values were scaled from absolute values to values more similar to the other features, this was done using robust scaling.

Due to the dramatic data imbalance, a balanced sub-sample of the dataset needs to be used for the training of models. To do this, the set of genuine transactions was randomly undersampled to create a sub-sample training dataset of 984 total samples, which had 50% fraudulent and 50% genuine transactions. This is quite a small dataset meaning the model produced won't be as accurate as it could be and will more than likely overfit it.

A twin sub-sample dataset was also created that was cleaned of extreme outliers on certain features that were chosen, to test whether this would improve a models accuracy on unseen data.

3 Feature Selection

To work out which features to use, a correlation matrix of all the features was generated (see Figure. 1). This helped me to determine the 3 most positively correlated features to the class, were V2, V4 and V11. With the 3 most negatively correlated features to the class being V10, V12, V14. The distributions of these features were then analysed with respect to what class data points were in (see Figure. 2). V11 lacks extreme outliers on the fringes of where a classifying threshold could be placed, this is good for training a model. V4 has a larger range than V11 and does have some outliers past the threshold, but the majority of its fraudulent values lie further from the interquartile range (IQR) of the genuine values. V14 had a very concentrated IQR, but did have some outliers on the threshold. V12 has a larger range but does have less outliers on the threshold. The V2 and V10 feature's fraudulent values were less differentiable to genuine values than the other features, with fraudulent values lying almost completely within the range of the genuine ones. For these reasons the features V14 was chosen to be used as part of the model, alongside either V11 and V4 which both have their own respective advantages. These 2 features will both be tested on preliminary models to determine which is best.

4 Models

Firstly, a logistic regression model was applied to V11 and V14 using a train-test split of 80/20 (for confusion matrix see appendices). This had a precision of 93.7%, accuracy of 91.3% anda recall of 89%. Another similar model used the dataset of cleaned values (see appendices), which doesn't include extreme outliers, this got a slightly higher precision at 94% but other metrics were lower, with recall being 87% and an accuracy of 89.8%. This shows that using cleaned data can result in a slightly more precise model, but at the detriment of recall, which is bad for this model when false negatives should be kept to a minimum. The logistic regression model was also applied to V4 and V14 (see appendices), which resulted in a better model, with a precision of 94.7%, recall of 91% and an accuracy of 92.9%. However, this was more liable to variance due to outliers and often gave very different data each time it was ran.

Then, a K-Nearest Neighbour classification model was applied to V11 and V14 (see appendices). Originally the value of k used was set to 5, resulting in a model precision of 95.9% and an accuracy of 90.4%, but only having a recall of 86.1%. Then a cross-validation optimiser was ran for 30 iterations, finding the optimum value of k to be 8 and that the original value of 5 was already a very good starting point, this resulted in the new model using k=8 having a slightly increased precision but slightly reduced recall. I then applied this model to V4 and V14, for k=5 this gave a precision of 94.9% and an accuracy of 89.8%, with a model recall of 82.4%. The cross-validation optimiser found the optimum value of k to be 17, but again this resulted in a very similar model with all metrics only increasing marginally.

Next a classifying perceptron model (see appendices) was applied to V11 and V14, it originally ran for 200 iterations, with a mutation value of 0.05 per iteration. This produced a model with 90% for precision, accuracy and recall. The same perceptron was tested using V4 and V14 values and resulted in a less precise model at 88.2% and slightly less accurate at 89.3%.

I then attempted to optimise this perceptron by changing the number of iterations and the mutation values, using V11 and V14 as these provided the best starting point. The most precise model created had a precision of 92.6% and an accuracy of 92.4% with a recall of 91.6%, this was found using 100 iterations and a mutation value of 0.1. I also experimented with perceptrons with a very high number of iterations, with one model using 500 iterations and a mutation value of 0.025. This created a model with a recall of 97.8% with only 2 false negatives predicted. However, this came at the detriment of a precision of 81.2% and accuracy of 88.3% with a lot of false positives, making up 10% of the total test transactions; this would be highly annoying to customers as it would flag a lot of transactions. Another problem with this experimental perceptron is that every time it was ran, it produced a wildly different model, this ended up creating a model with 93.6% precision, 93.6% recall, and 92.9% accuracy. However, this cannot be used in practice as it is too inconsistent and is very dependent on luck.

A Gaussian Naive Bayes algorithm was also applied with the model created using V11 and V14 features resulting in a precision of 92%, accuracy of 88.8% and a recall of 84.2%, the V4 and V14 version had a higher precision, at 95.6%, accuracy, at 91.4%, but a lower recall, at 86.7%.

5 Conclusion

The best model that could be consistently applied seemed to be logistic regression using the V4 and V14 features. This resulted in high precision, accuracy and recall with only 9 false negatives. This does however mean 9% of fraudulent transactions would be missed, which is not a great for the company as these would most likely result in lost revenue. The perceptron can result in more accurate models with much higher recall, but this comes at the cost of a very high variance while training. The K-NN and Naive Bayes models had low recall, making them the least helpful for this application.

The models have a high variance due to the massive undersampling necessary, this is because the dataset has such a small number of fraudulent transactions and these must all be used for training for maximum model fitness. Stratified sampling would also be impossible as the model will be heavily biased towards classifying something as genuine, the opposite of the main priority of the model. This would also result in a tiny amount of fraudulent transactions in the training dataset meaning the model will overfit massively. The better models produced have a low bias, meaning they would be quite good when being applied to real world data. When the dataset was cleaned of extreme outliers, the model was more precise but also seemed to overfit the dataset more and would likely have a reduced effect with other unseen data. The models could be vastly improved if the dataset was expanded, allowing for a greater range of genuine transactions to be included in the training and test data, resulting in a more well fitting model. A more recent dataset would also be more helpful in classifying transactions made nowadays, as credit card transactions have become more frequent and more online since this dataset was collected.

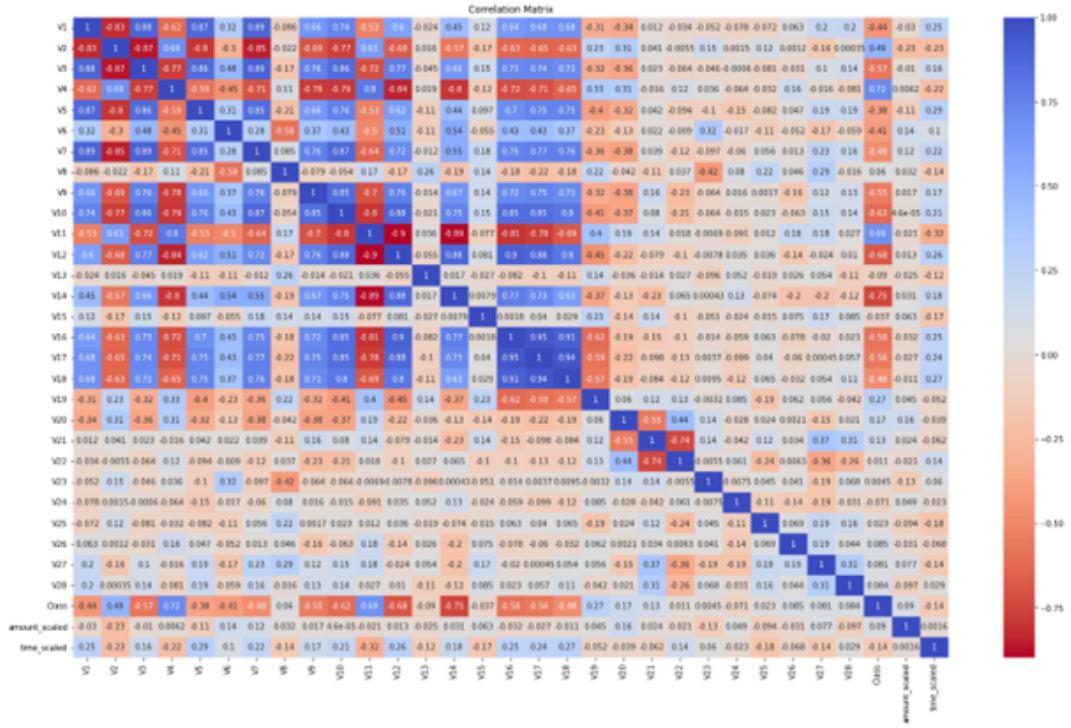


Figure 1: This shows the correlation between the values of a features, we are particularly interested in the correlation between the value of features and the Class, indicating whether it will be helpful for using in a model for prediction.

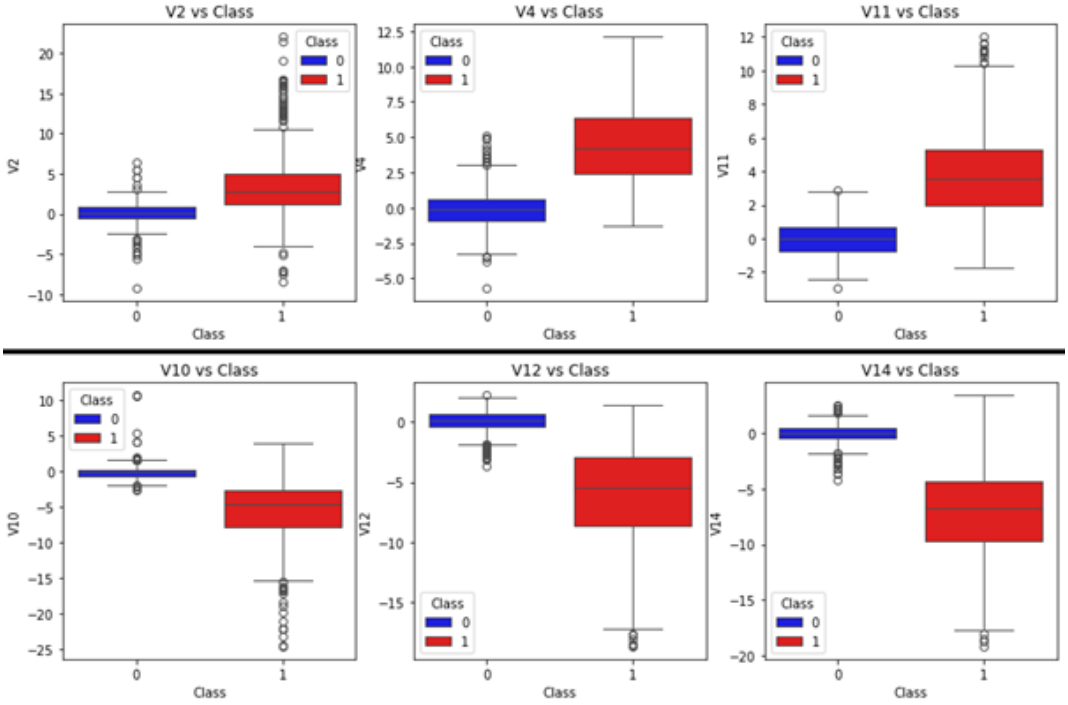


Figure 2: This shows the distribution of certain features being considered for selection. The blue distribution is the data of interest, as it shows how differentiable fraudulent transactions data is to genuine transactions.

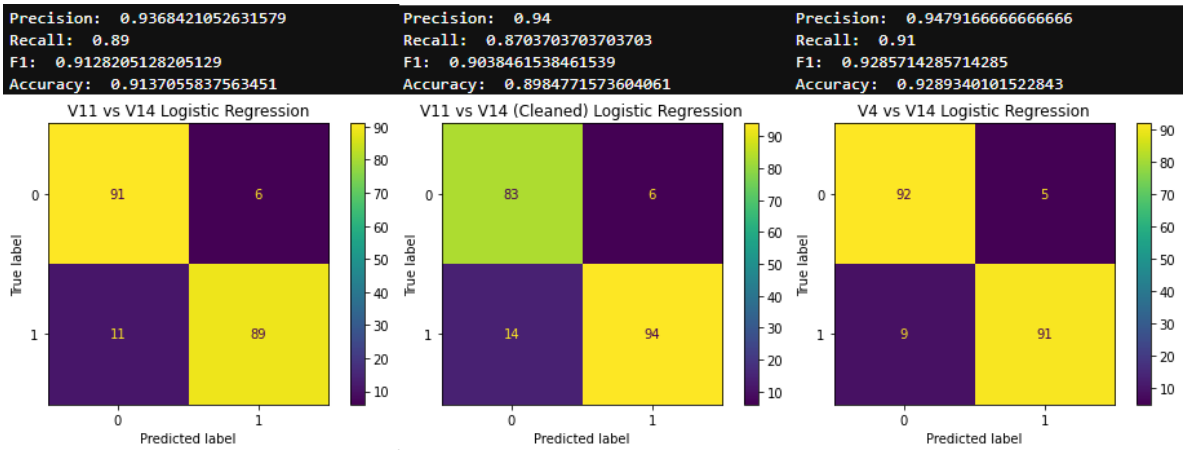
References

[1] Credit Card Fraud Detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. (accessed Dec. 5, 2023).

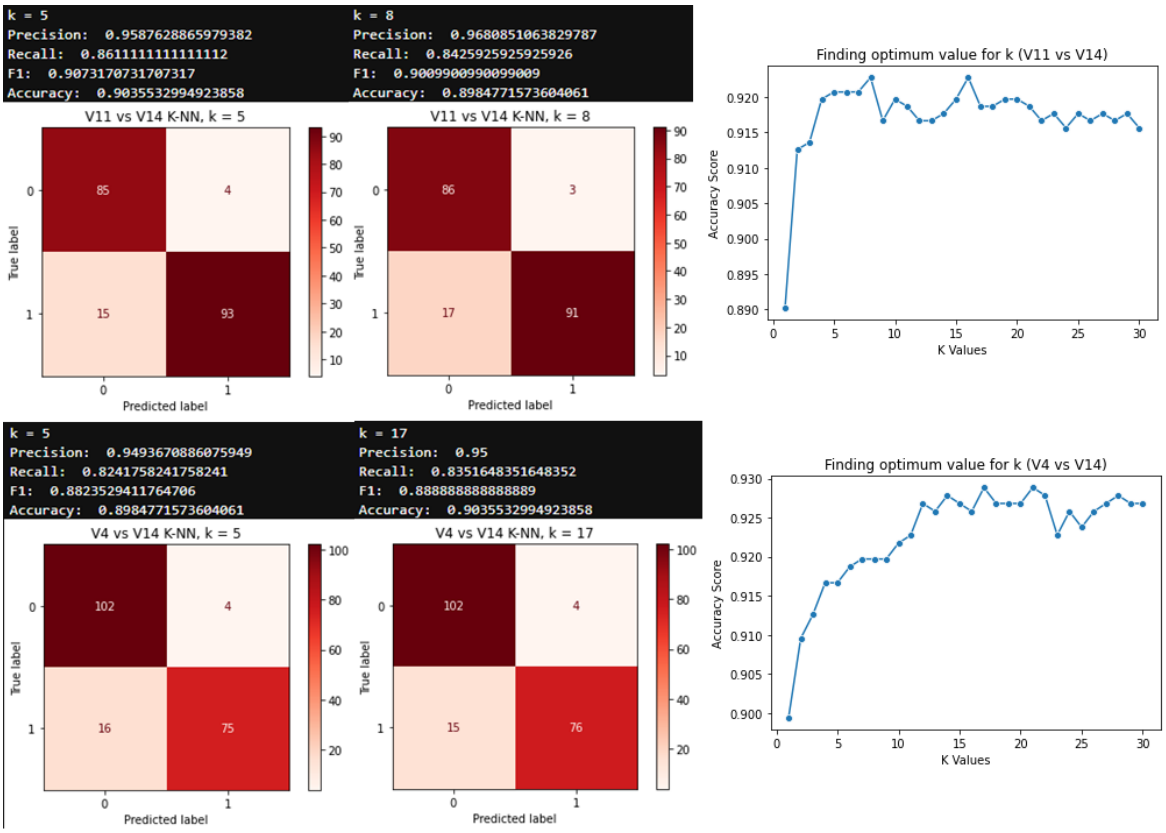
[2] Daniel Bukszpan. How Credit Card Companies Detect Fraud. https://www.cnbc.com/id/46907307, March 2012. Section: US: News (accessed Dec. 5, 2023).

Appendices

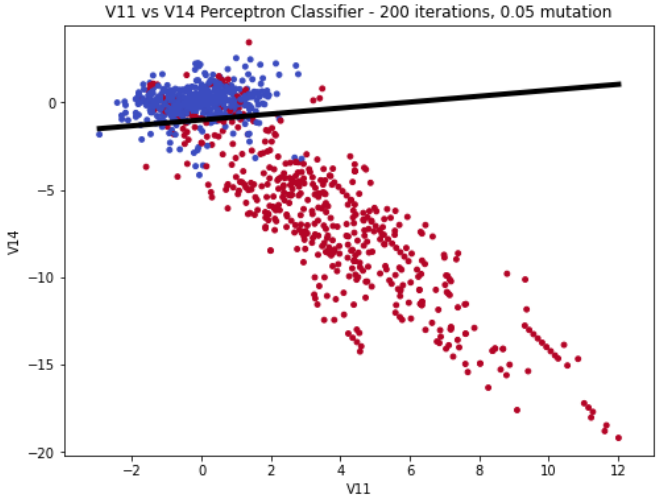
Logistical Regression confusion matrices



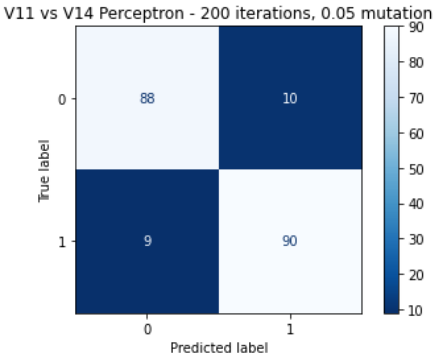
K-Nearest Neighbour confusion matrices and k optimiser graph

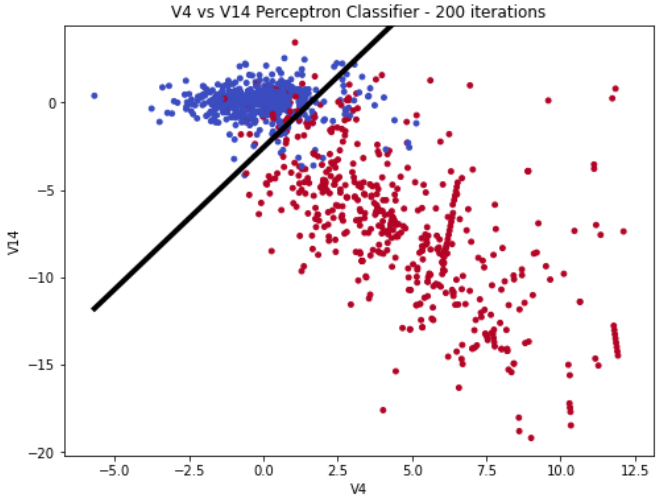


Perceptron classifier and confusion matrices

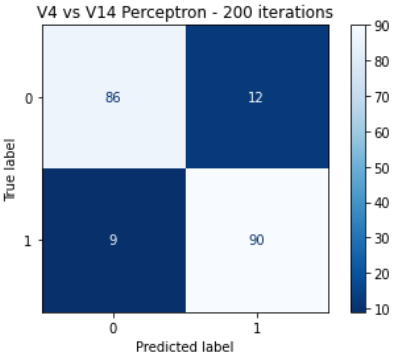



```
Precision:  0.9
Recall:  0.9090909090909091
F1:  0.9045226130653266
Accuracy:  0.9035532994923858
```

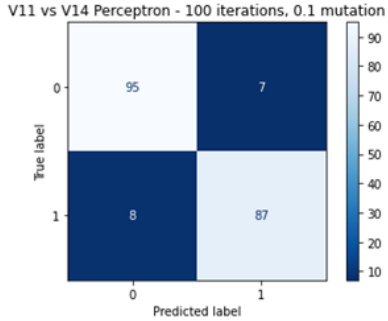




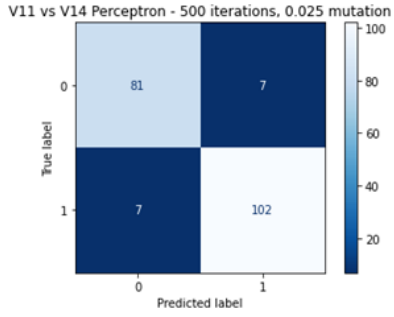

```
Precision:  0.8823529411764706
Recall:  0.9090909090909091
F1:  0.895522388059701
Accuracy:  0.8934010152284264
```



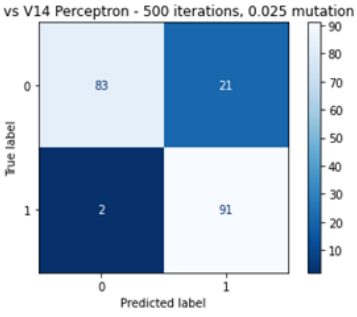

```
Precision:  0.925531914893617
Recall:  0.9157894736842105
F1:  0.9206349206349206
Accuracy:  0.9238578680203046
```




```
Precision:  0.9357798165137615
Recall:  0.9357798165137615
F1:  0.9357798165137615
Accuracy:  0.9289340101522843
```

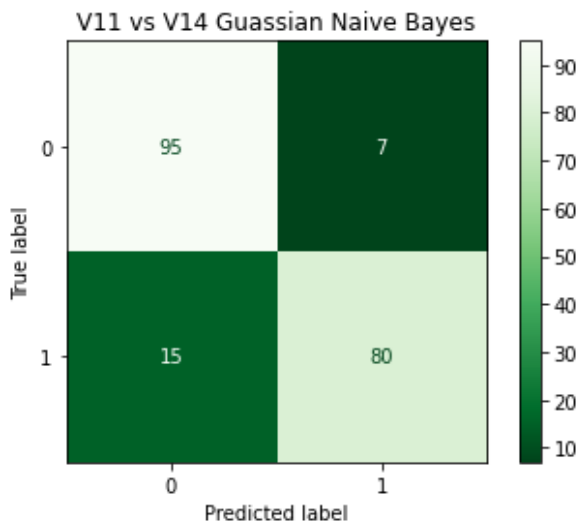



```
Precision:  0.8125
Recall:  0.978494623655914
F1:  0.8878048780487805
Accuracy:  0.883248730964467
```



Gaussian Naive Bayes confusion matrices

```
Precision:  0.9195402298850575
Recall:  0.8421052631578947
F1:  0.879120879120879
Accuracy:  0.8883248730964467
```




```
Precision:  0.9555555555555556
Recall:  0.8686868686868687
F1:  0.9100529100529101
Accuracy:  0.9137055837563451
```