APRENDIZAJE DE MAQUINA-002
Prof. FERNANDO ESPONDA

Final Project Report

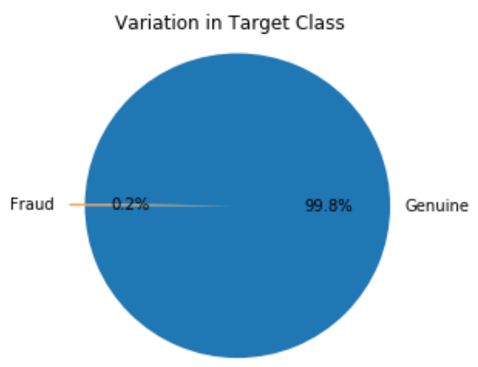
Done By:
Edwin Peter
177019

Introduction:
This project describes the usage of different classification techniques on credit card data. The data is taken from Kaggle [1]. The datasets contain transactions made by credit cards. The aim of this project is to train a classification model to identify fraud or genuine credit card transactions. The data presented is highly skewed, where 492 frauds out of 284,807 transactions are fraud. The attributes have been anonymized due to confidentiality and we are left with variables that have already undergone a PCA transformation, leaving 28 principal components and 2 attributes that are not anonymized: Time and Amount. 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. 'Amount' is the transaction amount. The target class takes on values of 1 in case of fraud and 0 otherwise. Having such an imbalanced dataset, there is a high risk of predicting all transactions as genuine when there could potentially be fraudulent cases. The interest of banks is to minimize fraud cases because it is physically impossible to manually look through each transaction to determine if a transaction is fraud. Thereby, we train models to help determine the nature of the transaction and lower the amount of physical checks conducted on these transactions.

Methodology:
Summary Statistics
The first step is to understand the variables in the dataset. A summary statistics is generated for each column.

	Time	V1	V2	V3	…	V27	V28	Amount	Class
count	284807	284807	284807	284807	…	284807	284807	284807	284807
mean	94813.85	1.17E-15	3.42E-16	-1.37E-15	…	-3.67E-16	-1.22E-16	88.34	0.0017
std	47488.14	1.95	1.65	1.51	…	0.40	0.33	250.12	0.0415
min	0	-56.4	-72.7	-48.3	…	-22.5	-15.4	0	0
25%	54201.5	-0.92	-0.59	-0.89	…	-0.07	-0.052	5.6	0
50%	84692	0.018	0.065	0.17	…	0.001	0.011	22	0
75%	139320.5	1.31	0.80	1.02	…	0.091	0.078	77.165	0
max	172792	2.45	22.0	9.38	…	31.6	33.84	25691.16	1

Summary statistics describe that V1 to V28 are not well scaled, where there is a wide range for each variable. Moreover, looking at Time and Amount which are relatively large compared to variables V1 to V28, scaling needs to be done across all these variables to make sure the (Euclidean) calculation of classification algorithms fit and contributes approximately proportionately to the final distance.



Understanding the Target Class
Looking at the target class, we determine that the fraud cases are highly under-represented in the dataset, representing a 0.2% of the entire 284,807 transactions. As mentioned previously, this causes a problem of the Accuracy Paradox [2], where “Predictive Models with a given level of Accuracy may have greater Predictive Power than Models with higher Accuracy.” Thereby, if we do not adjust the data set, we will naturally see that any classification algorithm trained would

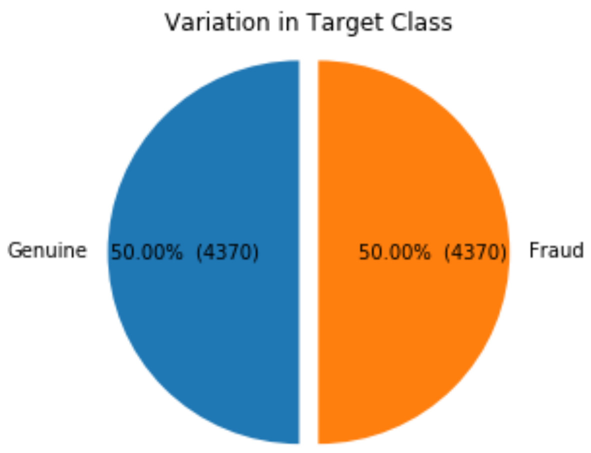
have a superb accuracy rate (~99%). This is because the algorithm is fitted and trained better to identify the genuine transactions instead of the fraudulent ones, which defeats the purpose for banks to train an algorithm. Therefore, through this, we need to determine a way to teach our machine learning algorithm to identify the fraudulent transactions as much as the genuine transactions. Accordingly, since we are trying to minimize the fraud transactions, we look at using recall and accuracy as a metric to judge the effectiveness of the models that we will train.

SMOTE to the rescue!

Synthetic Minority Oversampling TEchnique (SMOTE) is a popular oversampling method. For this data set, we use the imbalanced-learn package in Python to do a SMOTE and a Random Under Sampler. SMOTE allows us to create new minority class transactions using current instances. The simplified explanation of the algorithm is as such: Choose a minority instance, take K nearest neighbours and create new instances between the instance and its neighbours. Through this process, we up-sample the dataset to include an equal proportion of fraud and genuine transactions. However, it must be noted that this creates specific instances of fraud transactions that are like the current ones. Using SMOTE would make the algorithm perform poorer on fraud transactions with variables that are different than that of the training set. Therefore, we must be careful in the implementation and if possible, subject the algorithm training to more instances of fraud. On the other hand, due to the majority class being over-represented, we take a smaller proportion of it to aid in the speed of training and because we do not necessarily aim to 'detect' genuine transactions. We do this through a Random Under Sampler. An under-sampling combined with an oversampling creates a new balanced data set which will be used to train the models.

Data Pre-processing

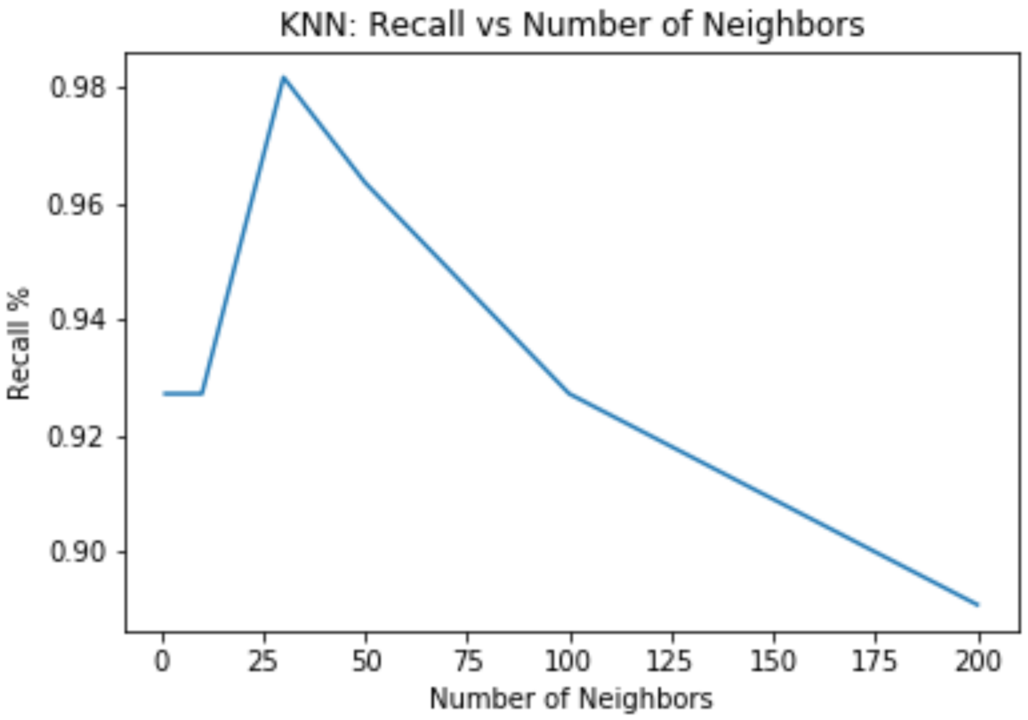
We first scale the data such that all the columns are proportionally represented in the dataset. Subsequently, we split the dataset into a Train set and Test set. The rationale to do so is to have an entirely new set of transactions that the model has not seen to test the accuracy of the model with. Using the train set, we conduct an over and under sampling using SMOTE and Random Under Sampler as previously mentioned. The result is a 50/50 representation of fraud and genuine transactions in the data set. This new data set will now be used for the training of different classification models.



Results:

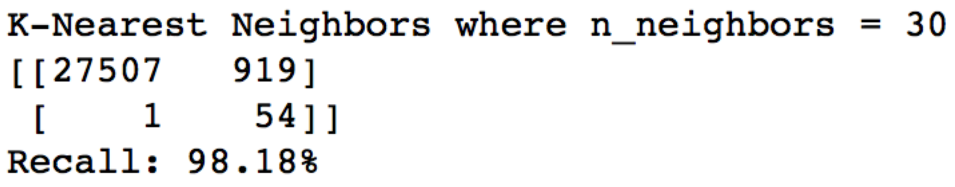
K-Nearest-Neighbours:

For training of the model, we will use a variety of classification algorithms and justify the best threshold to use for each of these models. The first model use is the K-nearest neighbours (KNN). The rationale for using KNN is because it is the simplest classification algorithm yet it can give a highly competitive result with other more complex algorithms. Thereby, KNN will be used as our baseline model and its results will be compared with the other algorithms implemented later. In KNN, the threshold being experimented with is the number of nearest neighbours, K. To get the optimal value of K, we plot the validation error curve to get the optimal value of K. The metric used in this curve is the recall rate, calculated



by the True Negative/ False Positive + True Negative, which essentially calculates how the percentage the model predicted it was fraud when it was fraud, compared to when it was a fraud but predicted by the model as non-fraud. As such, the threshold was varied from 1 to 200 and the results of recall plotted on a graph.

Looking at the graph, a suitable K to be selected would be between the range of 25 to 35, where K = 30 proved the best result of having a recall: 98.18% and a confusion matrix as such:

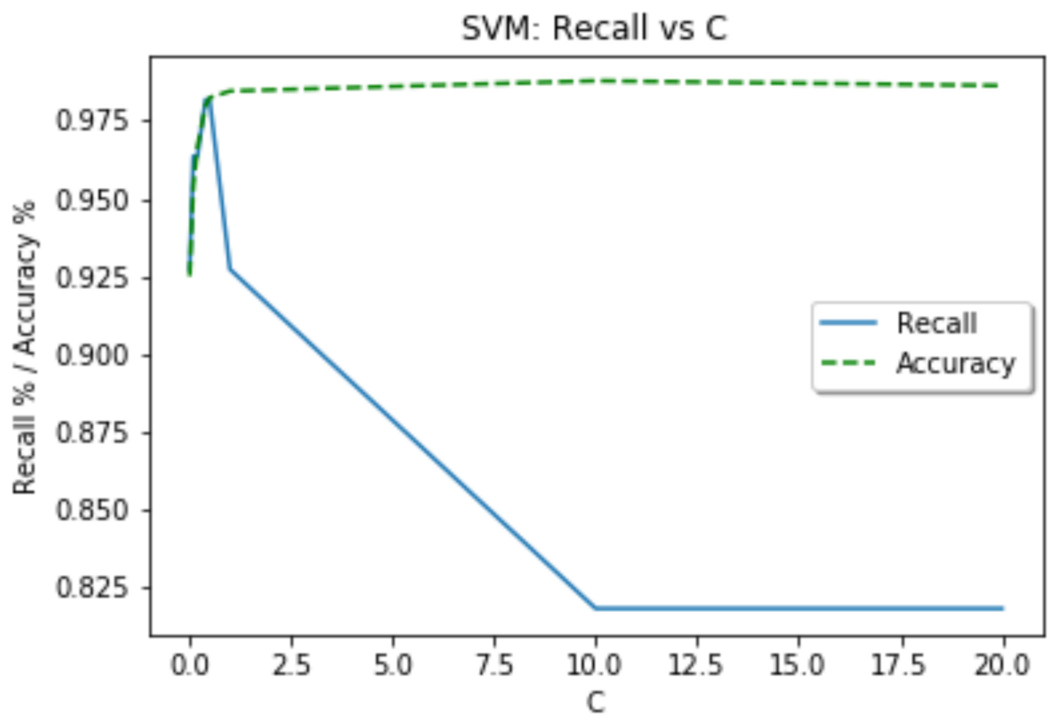


```
K-Nearest Neighbors where n_neighbors = 30
[[27507   919]
 [    1    54]]
Recall: 98.18%
```

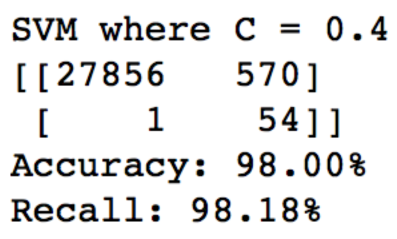
This is a relatively decent score where the classification algorithm only wrongly identified 1 transaction in the test set as non-fraudulent when it was in fact fraudulent, but identified correctly 54 other fraud cases. This is rather impressive despite the amount of training data provided being relatively small. Therefore, for KNN, the model selected is where number of neighbours is 30, achieving a Recall of 98.18%.

SVM:
The next classification algorithm used is Support Vector Machines (SVM). In SVM, we vary the threshold C. The C parameter is used to tell the SVM optimization how much to avoid misclassifying each training example. Using a very small value of C will cause the optimizer

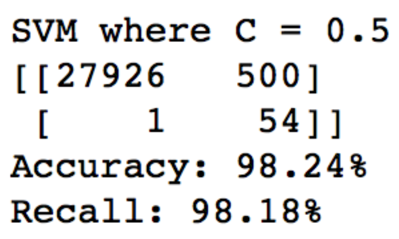
to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. On the other hand, using a larger value of C will cause the SVM optimization to choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Different values of C, ranging from 0.01 to 20 are varied to test how well SVM performs.



Looking at the graph of Recall vs C, we see that C peaks its performance when C is between 0.4 and 0.5. The selection of threshold is not conclusive to say which we should pick. Thereby, in this specific instance, we justify it by picking the one with the better accuracy. This is calculated by the number of genuine cases that have been predicted as fraud compared to those that are genuine and predicted as genuine. By picking a model that is better able to classify the genuine cases, the end goal is that we do not have to spend time and additional resources to look through 70 (570 – 500 based on the confusion matrix below) transactions to determine that these were mistakenly classified as fraud when they were genuine.

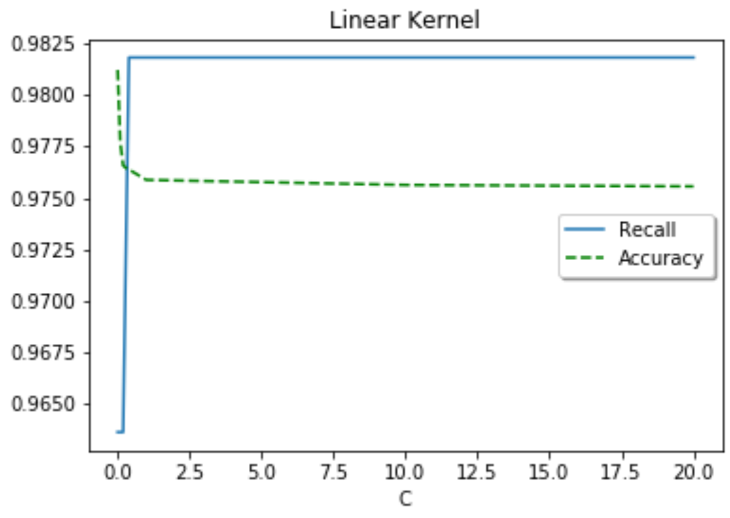


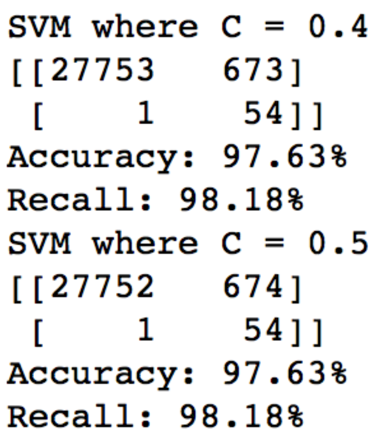
```
SVM where C = 0.4
[[27856   570]
 [    1    54]]
Accuracy: 98.00%
Recall: 98.18%
```



```
SVM where C = 0.5
[[27926   500]
 [    1    54]]
Accuracy: 98.24%
Recall: 98.18%
```

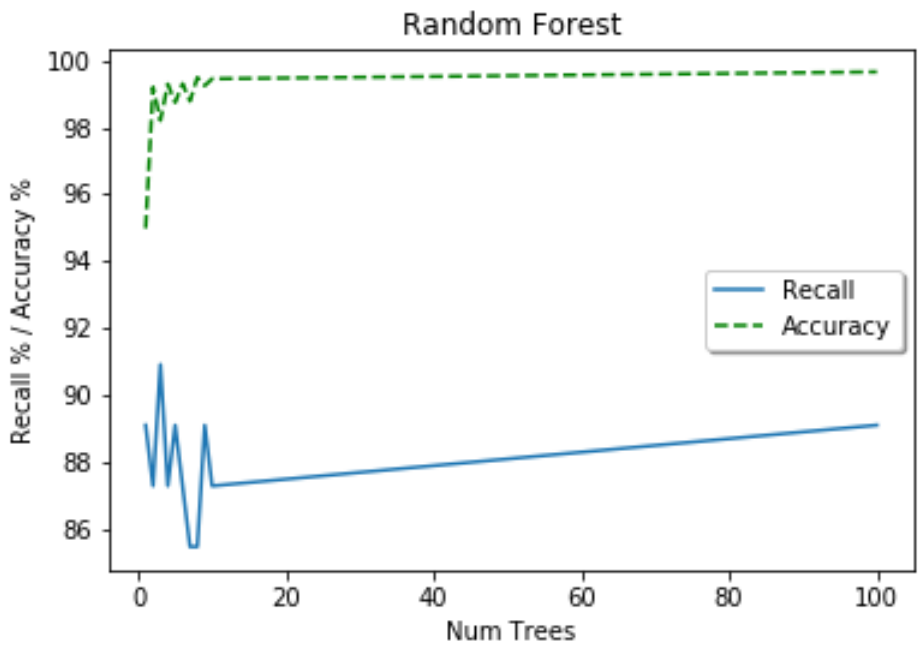
The above results were generated using a radial basis function kernel. To test if other kernels work equally well, I replicated the experiment with a linear kernel (below). Similarly, the results showed that the range of 0.4 to 0.5 showed the highest Recall and accuracy, but were lower compared to that of the 'rbf' kernel. Therefore, the final model for SVM is selected where the kernel is 'rbf' and the C value is 0.5, achieving an accuracy of 98.24% and recall of 98.18% on the test set.



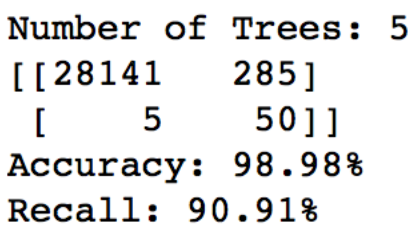


```
SVM where C = 0.4
[[27753   673]
 [    1    54]]
Accuracy: 97.63%
Recall: 98.18%
SVM where C = 0.5
[[27752   674]
 [    1    54]]
Accuracy: 97.63%
Recall: 98.18%
```

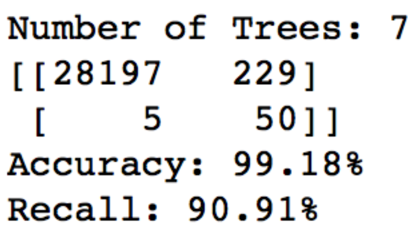
Random Forest:
The last classification algorithm used is the Random Forest algorithm. Random forest algorithm is a supervised classification algorithm. In this classification algorithm, we vary the number of trees created as an optimization parameter. The more trees used in the forest, the more robust the model. In the same way in the random forest classifier, the higher the number of trees in the forest, the higher the accuracy results.



Looking at the performance of random forest, we see that the optimal number of trees seem to be between 5 to 7, achieving an estimated accuracy of 99% and recall of around 90%. This shows that despite random forest being an ensemble of decision trees, it is not computationally expensive to train and can be competitively efficient to other algorithms with just a few trees being trained.



```
Number of Trees: 5
[[28141   285]
 [    5    50]]
Accuracy: 98.98%
Recall: 90.91%
```



```
Number of Trees: 7
[[28197   229]
 [    5    50]]
Accuracy: 99.18%
Recall: 90.91%
```

Comparing KNN, SVM and Random Forest:
Looking at the 3 classification models trained, we see that there is usually a trade-off between accuracy and recall. Accuracy here refers to the number of genuine transactions predicted as genuine by the model. To achieve a high recall, as seen in KNN and SVM, a lower accuracy would be recorded. On the contrary, random forests seems to be better at accuracy than recall, meaning that it would be better at detecting a genuine transaction whereas KNN and SVM are better at detecting fraud transactions. As mentioned earlier, the difficulty in ensuring a high recall value could also be due to the usage of SMOTE that makes the models less robust to looking for fraudulent transactions. It is therefore important to be careful when resampling the data and if possible, look at using more data to train (which is not possible in this case).

Conclusion:
In conclusion, the best model in this scenario would be SVM with a high accuracy and recall compared to the other models. In the context of the bank, it would be useful for them to be able to identify fraudulent transactions easily, as seen in SVM (only misidentifying 1) and being able to track down small amounts of transactions that the model considers as fraud to be manually checked. For future works, it would be wise to ensemble these methods described above so that we can achieve both a high accuracy and recall.

Bibliography
[1] https://www.kaggle.com/dalpozz/creditcardfraud/data
[2] https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b