Credit Card Fraud Detection

with Machine Learning

Duc Le, M.S

Interactive Data Analysis

Computational & Data Sciences

Chapman University

Abstract

The following report summarizes the approach to tackling the issue within imbalanced dataset when it comes to applying classification methods. The problem was solved by applying the Synthetic Minority Oversampling Technique (SMOTE) to the imbalanced dataset prior to training it on desired classification models, which in this case are Random Forest and Neural Network. The results achieved were much more meaningful after SMOTE was applied to the dataset.

Keywords: *Credit Card Fraud. Machine Learning. SMOTE. Random Forest. Neural Network.*

Introduction

The two main research questions are:

- How to overcome the imbalance nature of the dataset, also efficiently and accurately classify whether a transaction is fraudulent or not?

- What variables are deemed important when it comes to classifying fraudulent activities?

The Dataset

The selected dataset was acquired from Kaggle and contained 6,362,620 rows by 11 columns. The dataset was however generated by the Paysim Simulator, thus does not actually contain real transactions. Due to the restrictions and confidentiality issues in finance, there was no way of acquiring data of real transactions. However, the simulator

uses aggregated data to generates samples that closely resemble the normal operation of transactions and injects malicious behavior for later evaluation on the performance of fraud detection methods.

The data contains a total of 11 columns such as:

- Step: maps a unit of time in the real world. For example, 1 step is 1 hour of time.

- Type: types of transaction, i.e.: Cash In/Out, Debit, Payment and Transfer.

- Amount: amount per transaction.

- nameOrig: name of the person who started the transaction.

- OldbalanceOrg: initial balance before the transaction.

- NewbalanceOrg: new balance after the transaction.

- nameDest: the recipient of the transaction. The labels that start with "C" indicate a Customer, whereas a starting "M" means Merchant.

- oldBalanceDest: initial balance of the recipient before the transaction.

- newBalanceDest: new balance of the recipient after the transaction.

- isFraud: the target variable. The indication of whether the activity is fraudulent or not (dichotomous).

- isFlaggedFraud: whether the transaction was flagged as fraud.

The dataset contains 6,354,407 fraudulent activities to only 8,213 non-frauds. Thus, only 0.12% of all transactions are fraudulent. Obviously, this limits the credibility of the overall accuracy achieved from any classification algorithms.

Data Pre-Processing

Fortunately, the dataset contains no NA's, thus no sample imputation or omission was needed. In terms of features selection, "Step", "nameOrig, "isFlaggedFraud" and "old

and new BalanceDest" were omitted from the model. The reasoning behind this decision was "Step" does not provide any meaningful interpretation to the data. If anything, it provides some context on each simulation and how it relates to the real world, which is not really needed if we were to have data from the real world. Including all the names of the people who started the transactions would just be counterproductive as it will lead to overfitting in the models. Although there were technically no NA's in the data, there was a large amount of unavailable data for the "old" and "new" balanceDest columns were marked as 0's. Lastly, "isFlaggedFraud" is non-essential as all values in the column are 0 (meaning none of the activities were flagged at all). In addition, for the purpose of mitigate the computation expense, the training data was slightly modified. All fraudulent accounts were reserved but they are accompanied by only 10% of the non-fraud activities. Thus, resulting in a training dataset consisted of 444,811 non-fraud to 5721 fraud samples (still only 1.2% are frauds).

Exploratory Data Analysis

The data is divided into a 70-30 training-testing split. As mentioned above, the pre-SMOTE training data contains about ~450k samples (Figure 1). 5721 of which are non-frauds which account for about 1.2% of all transactions. Despite the slight modification, the training data remains heavily imbalanced thus SMOTE will be required in order to achieve good accuracy and precision.
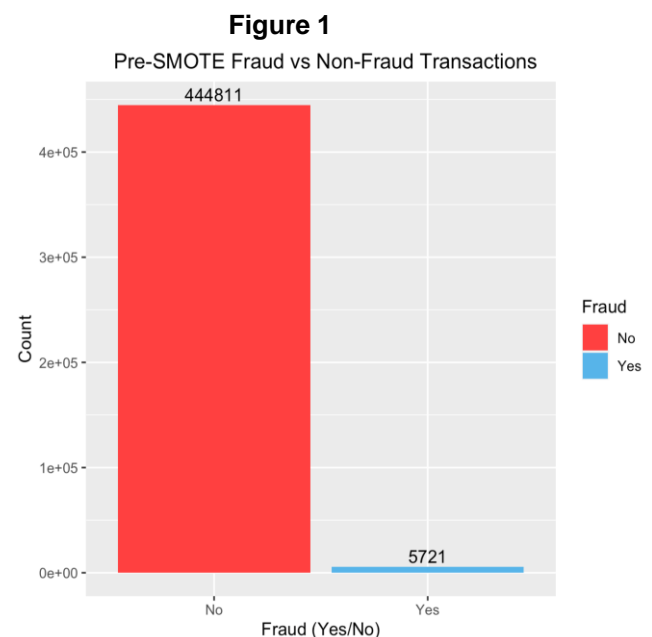
Figure 1

Pre-SMOTE Fraud vs Non-Fraud Transactions

Figure 2

Looking at Figure 2, with outliers removed, we can see that most transactions are done on a smaller scale. The distribution of transaction amount is heavily skewed to the left. Intuitively, this makes sense because every day commercial purchases and other use of money are rarely done on a six, seven figure scale.
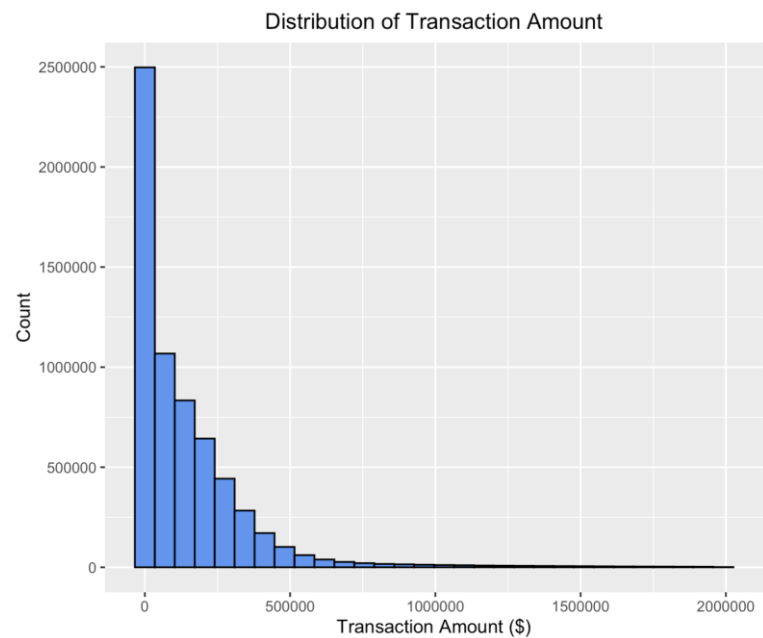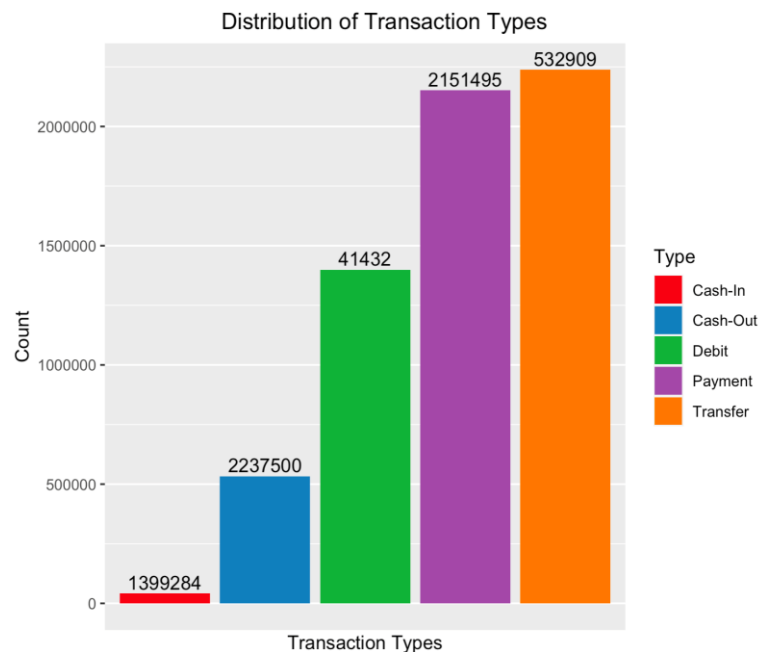
Figure 3

Figure 3 provides an insight to the distribution of transaction types. Transfers and forms of payment dominate the dataset as transactions done with debit come in third. It is interesting to see the discrepancy between recorded transactions of cash-in vs. cash-out. However, this could also be the result of my modification random sampling.

Synthetic Minority Oversampling Technique (SMOTE)   **Figure 3**

SMOTE is a popular solution to solving the

discrepancy issue in imbalanced data. Using

SMOTE, we were able to synthetically

generate new samples by observing the

characteristics of the minority set. SMOTE

grants the target variable a nice 55-45 ratio,

thus creating a much more balanced dataset

ready for classification (see Figure 3).

The new training set now contains ~440k

samples, ~205k of which are now frauds.
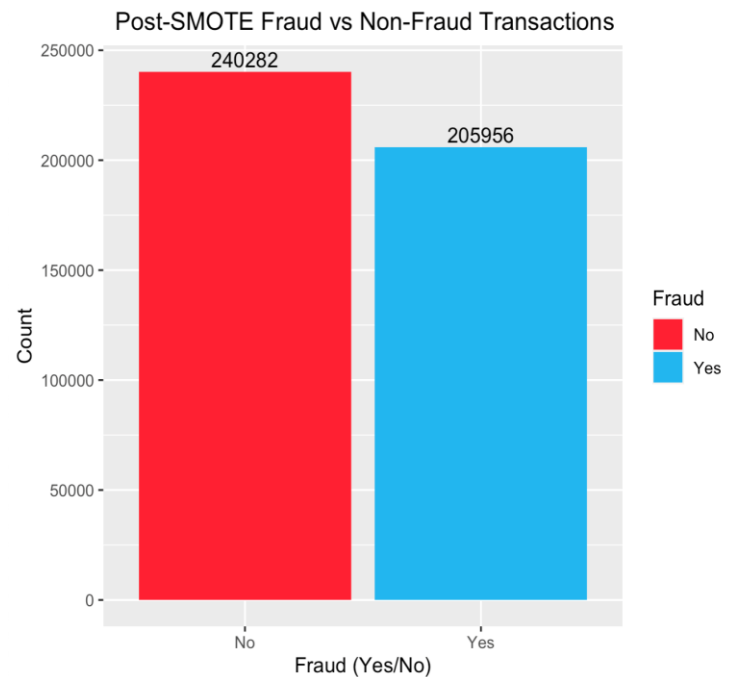
Classification Models

Neural Network (Keras)

The first classification model attempted is a neural network using the Keras interface.

The neural network was chosen due to its nature to learn well if given sufficient data

(~440k+ of training data). The network is made up of 8 total layers, including the input

and output layers. The output layer uses Softmax as its activation function since we are

doing classification. LeakyRELU and Hyperbolic Tan (Tanh) are sprinkled through the

rest of the layers, one of which contained a dropout rate of 0.1. A variety of

combinations of activation functions were tested for this neural network including the

popular Sigmoid and RELU functions. RELU was found not to be very effective even

though it was much less computationally expensive when compared to Tanh or

Sigmoid. Tanh was chosen over Sigmoid due to Tanh providing stronger gradients and

its ability to overcome certain issues the Sigmoid function cannot. After many trials, it

was evident that the network performed best when dominated by Tanh. The neural

network uses Stochastic Gradient Descent as its optimizer and Binary Cross Entropy as

its loss function since we are doing binary classification. Finally, the network uses a 70-

30 validation split and has 50 total epochs.

Figure 4. Pre-SMOTE NN

The constructed neural network will be
used to train on both pre-SMOTE and
post-SMOTE data with identical
parameters. Figure 4 displays how the
network performs throughout 50 epochs.
Both the model's training and validation
accuracy were able to stabilize almost
instantaneously. Both accuracies started
above 0.99 and never dipped below that
mark after the first epoch. However, we

should take the accuracy with a grain of salt. This should not come as a surprise as the

network does not have to work hard to learn as it is training on a heavily imbalanced

dataset after all.

With the limitation of the accuracy metric, we tested the network on the testing set and

analyzed the resulting confusion matrix for a higher-level analysis. Figure 5 is the

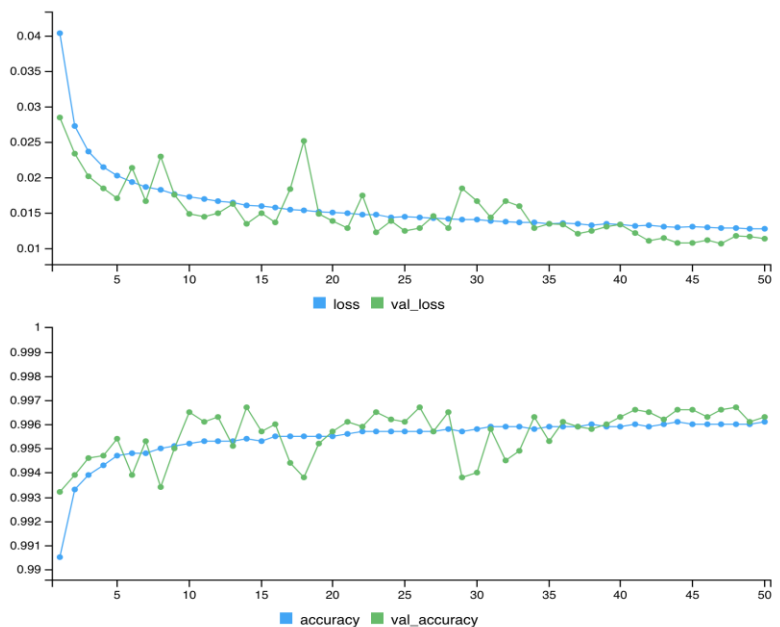confusion matrix acquired after testing the network on unseen data.

Figure 5

To avoid any confusion (no pun-intended), the smaller percentages represent the classification rates, whereas the larger numbers represent the distribution of the labels. As we can see, the neural network had no trouble classifying the non-fraud activities, as it achieved a 99.7% sensitivity. Surprisingly, its specificity was 81.5% which was terrific given the nature of the imbalanced labels.
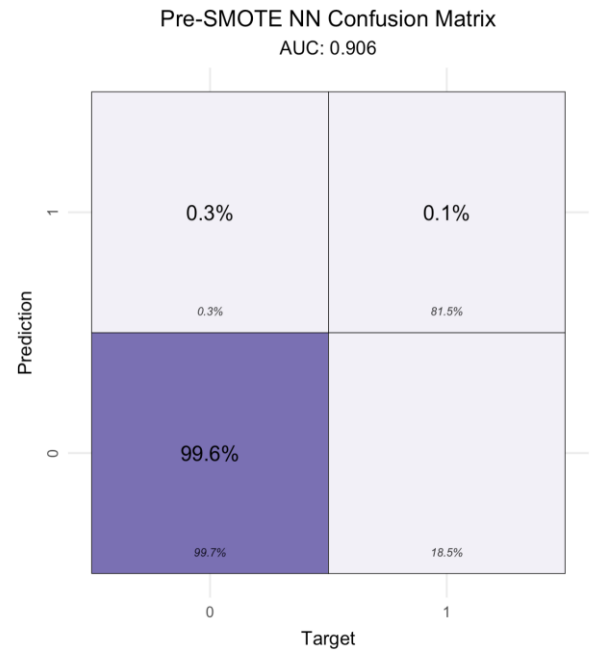
Pre-SMOTE NN Confusion Matrix
AUC: 0.906

	0.3%	0.1%
	0.3%	81.5%
	99.6%	
	99.7%	18.5%

Prediction / Target

Figure 6. Post-SMOTE NN

As we run the same neural network through the new post-SMOTE training data, it is evident that the validation accuracy struggles to converge as the model is having a harder time learning the new labels. Despite the training accuracy's immediate convergence, the validation accuracy seems to fluctuate until after the ~26th epoch, where it showed some signs of stabilization. Although it could be concerning to see the volatile showings of validation accuracy, this may be a classic case of bias-variance tradeoff in machine learning.
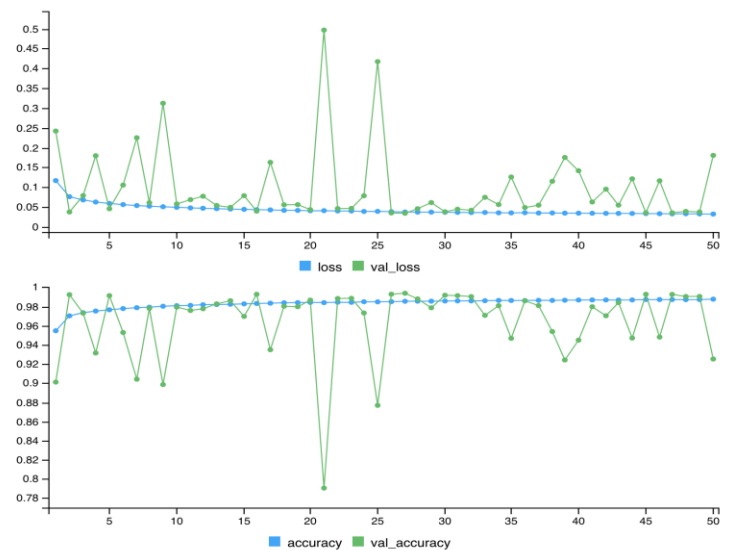
The AUC slight dropped to 0.875 from 0.906. The specificity significantly increased to 99.6%, a big climb from 81.5%. This near-perfection specificity came at a cost of our sensitivity, however. The true positive rate dropped by nearly 25% as we witness a rise in the false negative rate. The post-SMOTE neural network is much more careful and
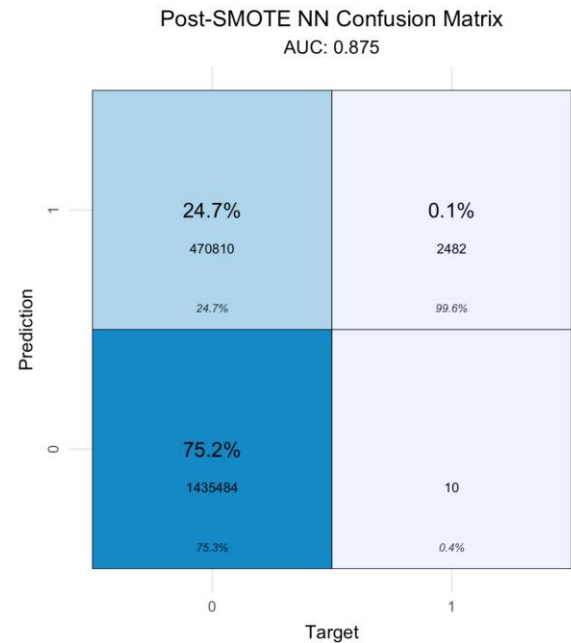
paranoid than the previous, maybe a little bit too much. This issue will require further investigation and parameters tuning. As of now, given the context of the problem, which is to accurately identify fraudulent transactions, this may be an unideal tradeoff we can afford. The two networks individually took about 24 minutes per run.
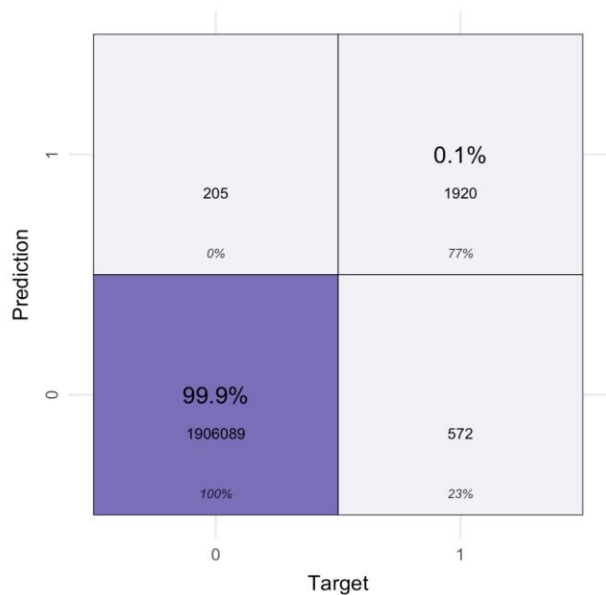
Random Forest

The Random Forest model requires less work to construct and achieved significantly better results for both training datasets. The model builds a maximum of 50 trees and uses only 2 variables per tree to evaluate. The algorithm is much less computationally expensive than the neural network. Each run time was approximately 4-5 minutes and perform significantly better. Both testing runs were able to achieve a near-perfection true positive rate. The difference lies in their specificities; the model trained on the Pre-SMOTE data only achieved a 77% specificity whereas the latter model was able to obtain a staggering true negative rate of 98.8%. See Figure 8.1 and 8.2 for more details.
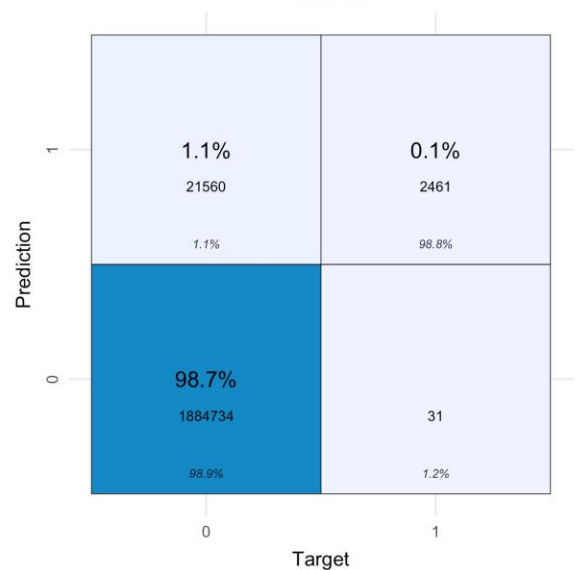
Shown below are the confusion matrices for the Pre-SMOTE and Post-SMOTE testing

runs:

Figure 8.1   **Figure 8.2**

Pre-SMOTE RF Confusion Matrix
AUC: 0.885

Post-SMOTE RF Confusion Matrix
AUC: 0.885

Lastly, we were able to gain insights on the importance of each predictor variable using

the built-in "varImp" function of R's

randomForest package. The results are

measured by the Mean Decrease in Accuracy

and rank the "Type" of transaction (transfers,

debit, payment, etc.) as the most important

variable for classification. Coming in 2nd is the

amount per transaction and followed by the

old & new balance of the account.
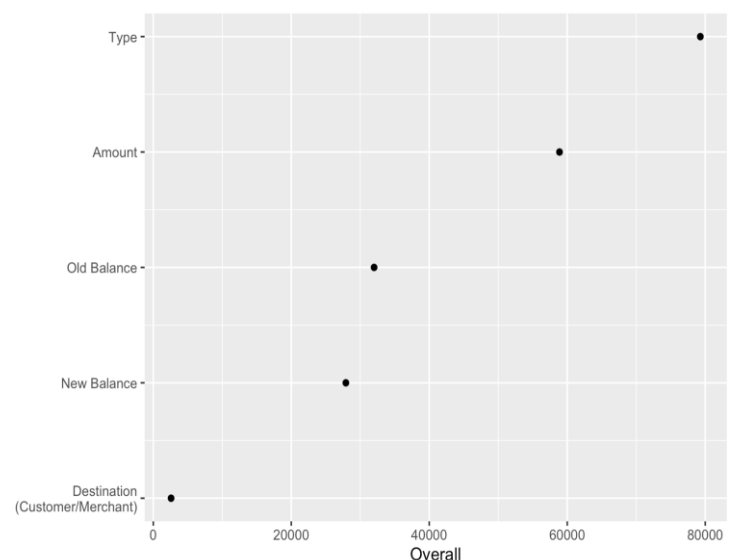
Conclusion

Revisiting the proposed research questions, for the 1st, we can conclude that at the current scale, a combination of Random Forest and SMOTE does an excellent job in classifying the target variables. Although the non-SMOTE imbalanced dataset still achieved respectable classification rates, the post-SMOTE data proves to be exceptionally superior, having a near-perfect sensitivity and specificity.

With Random Forest, we were also able to extract the most important predictors for detecting fraud transactions, which answers the 2nd proposed research question. The Mean Decrease in Accuracy shows that when attempting to flag the transaction, it is imperative to pay attention to its Type and Amount.

Future investigation could be to explore the sustainability of the Random Forest model. Despite its remarkable classification rate, I am interested to see how its precision, accuracy as well as efficiency scales as the dataset scales. Keep in mind that the training data was only a small subset of the original data. In addition, I am curious to analyze the inner workings of the neural network and see whether the network can perform better with some parameters tuning. Given a larger dataset (perhaps the original dataset), could the neural network be the more efficient and versatile classification algorithm for this specific problem? Knowing the nature of neural networks, I do expect this to be the case, but this hypothesis will be up for investigation.