Credit Card Fraud Detection using Machine Learning Algorithms

Roya Latifi
School of Economics
University of California, Los Angeles
Los Angeles, California 90095
Email: royalatifi@g.ucla.edu

Yiran Sun
School of Economics
University of California, Los Angeles
Los Angeles, California 90095
Email: yiran8sun@g.ucla.edu

Maxwell Speill
School of Economics
University of California, Los Angeles
Los Angeles, California 90095
Email: mspeil@g.ucla.edu

***Abstract*—This paper analyzes the ability of 6 unique machine learning models to predict fraudulent credit card transactions. We use credit card transaction data from the UCI machine learning repository. Model performance is evaluated using Recall score as well as AUC. These metrics were picked due to the importance of correctly classifying as many true fraudulent transactions as possible, which is far more important than not misclassifying a legitimate transaction. We find that an optimized Feedforward Neural Network performs the strongest in terms of Recall, while a Linear Discriminant Analysis performs strongest in terms of AUC.**

***Index Terms*—Machine learning, Probability, Classification, KNN, Neural Networks, Support vector machines, Random Forest**

I. Introduction

Nowadays, a majority of purchases made each day are done so using credit cards in place of cash, many of them online. This means that transactions can be made faster and more seamless than ever before, greatly increasing the ease with which people can acquire things that they want. There is also a dark side to this technological development however, as it also creates a much greater possibility for fraudulent transactions. Due to the great risk, much emphasis must be placed on identifying and preventing fraudulent transactions. The power of machine learning lends itself readily to this dask. Able to shift through and gain insights from vast amounts of data exponentially faster than any human could, machine learning models have great potential to ultimately prove themselves victorious in the battle against fraudulent credit card purchases.

A. Task Description

We develop 6 different machine learning models to classify transactions as either fraudulent (positive) or legitimate (negative). Similar to the task of screening a patient for cancer, in this case a false negative is potentially far more detrimental than a false positive as misclassification of a fraud case is more expensive than misclassification of a non-fraud case. If a legitimate transaction is flagged as fraudulent, the worst that will happen is that the person making this purchase will get an extra notification on their phone asking something along the lines of "This transaction was flagged fraudulent. Was it you?" and will just have to hit "Yes" and the transaction will go through regardless. If a fraudulent transaction is missed, however, it could mean the theft of thousands of dollars or more, which can have serious negative consequences on the victims life. Because of this, we chose to primarily focus on our model's AUC and recall scores rather than precision or accuracy since the percentage of actual positives that were correctly classified (recall) is more important than the percentage of predicted positives that were correctly classified (precision) in our case.

II. Data

A. Dataset description and Data Processing

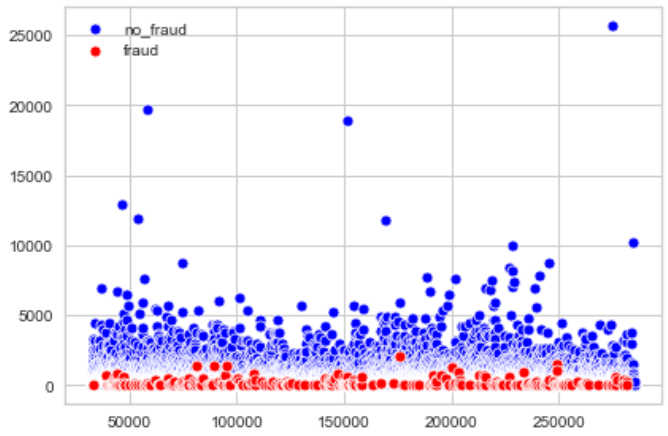
We utilized a dataset, taken from the UCI Machine Learning Repository, of 284,807 credit card transactions made in Europe over two days in September 2013. Of these, 492 transactions were fraudulent, making up only .17% of the data. This is another reason we chose to use recall and AUC over accuracy. Because such a vast majority of the transactions were not fraudulent, many of the models accuracy scores were meaninglessly high by just classifying every transaction as legitimate.

Each data point contained 33 variables. The first was its class label, 1 for a fraudulent transaction, 0 for a legitimate transaction. Next were the variables "Time" and "Amount." We decided to drop "Time," which simply referred to the number of seconds that had passed since the first transaction in the two day window from which the data was collected. This variable could not in any way be applied to any future transactions not from this data set, and so it was not helpful in building a model that could be used to make classifications in the future. We kept "Amount" which referred to transaction amount and since it ranged from 0 to 25,691.16, we normalized it to improve model performance. The other 30 variables were simply listed as V1 through V30, and had already been preprocessed for machine learning usage. The data had been normalized, and any description of what each variable original corresponded to had been lost. This could be due to the potentially confidential nature of many credit card purchases. There were no missing values in the dataset.

B. *Data exploration*

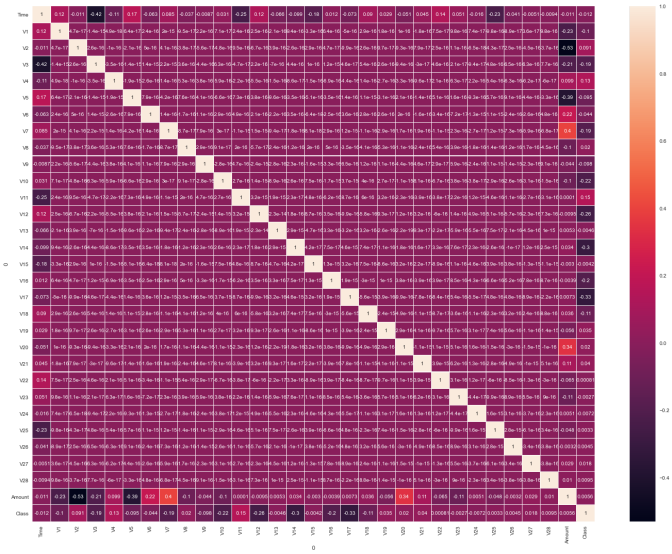
The only variable we explored visually by itself was "Amount," as this was the only variable we were able to assign any meaning to.

Figure 1



From figure 1 we see that the majority of the fraudulent transactions were relatively small, the largest being $2,500.

Figure 2



From figure 2, the correlation matrix, we see that there is weak to no correlation between any two of the variables other than Amount or the class label, meaning we likely do not have redundant features.

III. Model Implementation

A. *Evaluation metrics*

As mentioned previously, the primary metrics utilized to assess model performance were Recall and AUC, with Recall taking precedence. We split our data in half to create our training and testing sets, and then further split the training set into ⅘ and ⅕ for a training set and validation set, respectively.

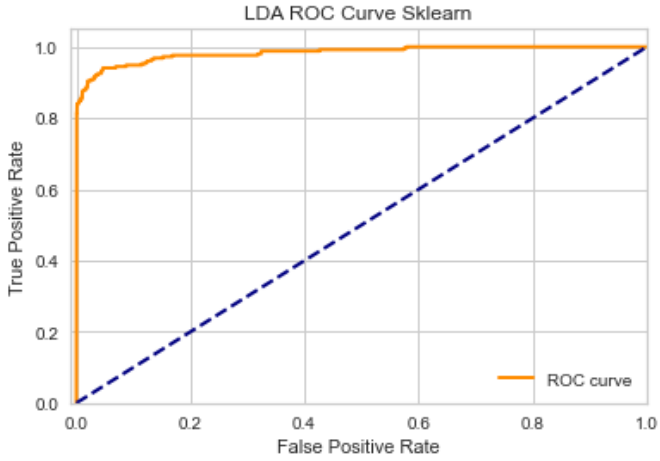
For each model, we optimized hyper-parameters by training over the training set and evaluating performance over the validation set. Once each model's hyper-parameters were optimized, we evaluated each model's performance over the testing set and reported the results.

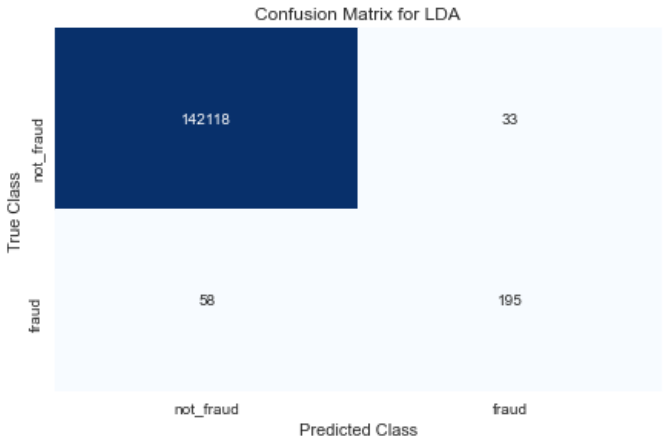
B. *Models*

1. *Linear Models*

Four different linear models were compared: Ridge Regression, LASSO Regression, Stochastic Gradient Descent (SGD), and Linear Discriminant Analysis (LDA). Ridge Regression was optimized with an alpha learning rate of 1.6, LASSO was optimized with an alpha learning rate of .0013, SGD was optimized with an alpha learning rate of .001, and LDA was optimized with the Singular Value Decomposition solver.

Once each linear model was optimized using the validation set, each was run over the testing set to determine the best overall model. All AUC scores were above .97, with SGD having the highest at .985133. However, LDA had the highest Recall by far, and thus is the linear model we ultimately deemed best.

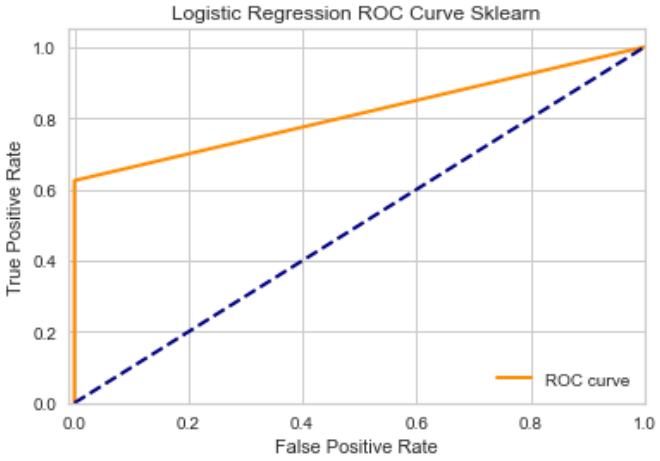


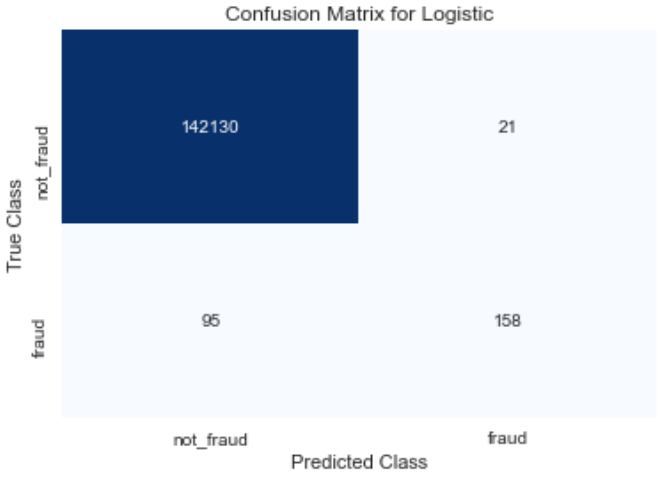


2. *Logistic Regression*

When optimizing the Logistic Regression model, multiple combinations of solvers and penalizations were tested. These were: simple Logistic Regression, Logistic Regression with Ridge penalization and SAG solver, Logistic Regression with Ridge penalization and liblinear solver, Logistic Regression with LASSO penalization and liblinear solver, Logistic Regression with LASSO penalization and SAGA solver, and Logistic Regression with Elastic-Net penalization.

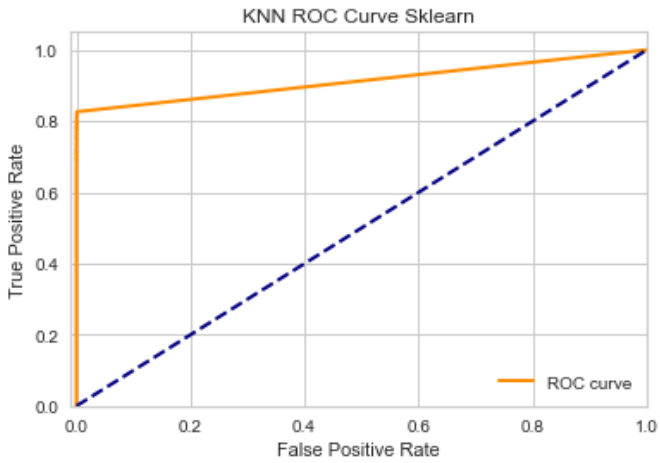
Simple, Ridge-SAG, LASSO-SAGA, and Elastic-Net all tied for the highest validation Recall scores of .62222. LASSO-SAGA had the highest validation AUC by a hair, with a score of .977322, and was thus deemed the optimal Logistic Regression model.

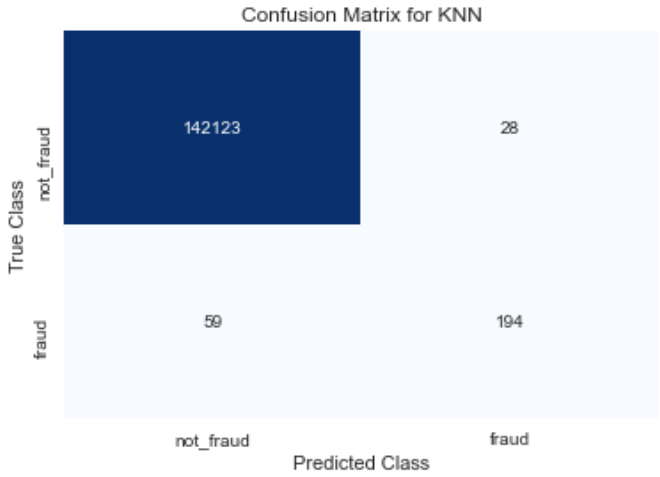




3. *K-Nearest Neighbors*

The optimized KNN model was with a K value equal to 4. This model produced a validation AUC of .93322

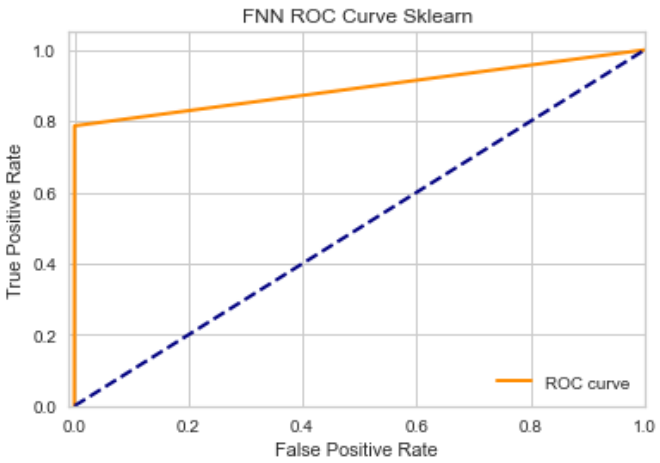


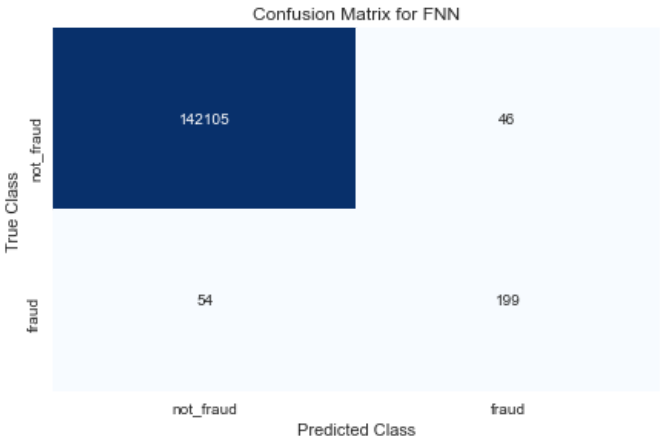


4. *Feedforward Neural Network*

When developing our FNN models, 4 different combinations of number of layers, neurons, and activation functions were tried. Model 1 was 3 layers, going from input→ 256→ 128→ 64→ output, and had a sigmoid activation function at each layer. Model 2 was also 3 layers, going from input→ 8→ 10→ 12→ output, with ReLu activation functions at each layer except the output layer, which still utilized a sigmoid activation function. Model 3 was 4 layers, input→ 8→ 40→ 42→ 44→ output, with ReLu→ ReLu→ Linear→ ReLu→ Sigmoid for activation functions. Lastly, Model 4 was again 3 layers, input→ 28→ 10→ 12→ output, again with ReLu activation at each layer except for the output, which was sigmoid activation.

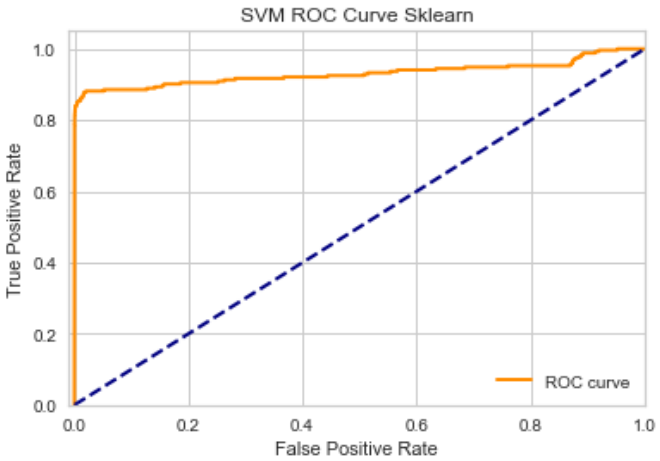
Model 1 performed the strongest in terms of both AUC and Recall. Its validation AUC score was .9332, with a validation Recall score of .8667

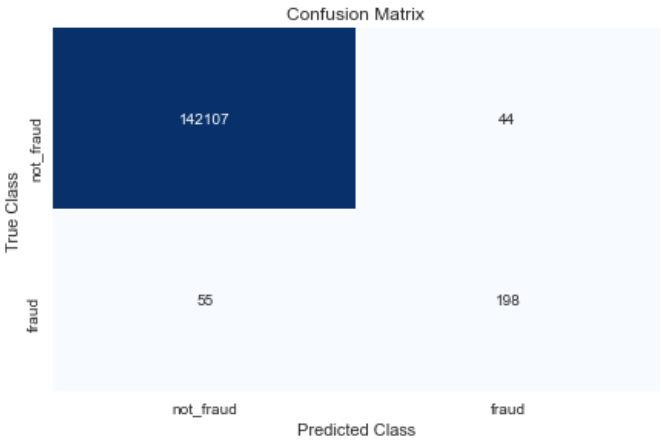




5. *Support Vector Machine*

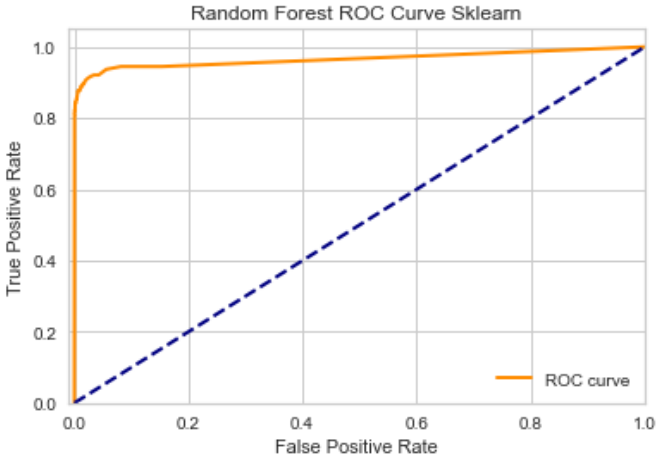
For our SVM model, we first tested each kernel type and determined that the Linear kernel performed best over many C values. We then optimized C over a range of values, and determined that a C of 5 produced the highest validation AUC of .9554.

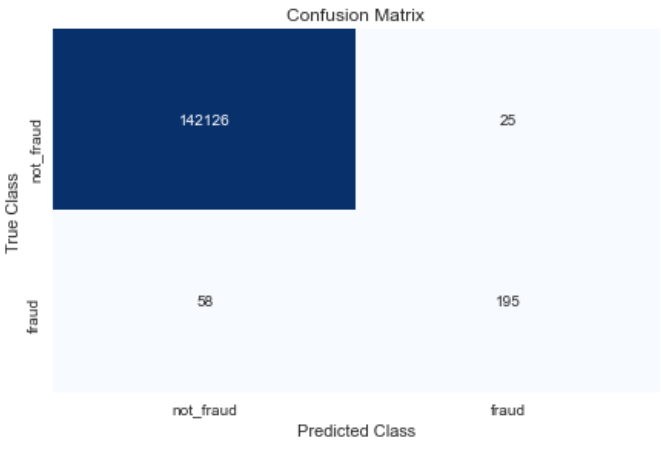




6. *Random Forest*

Random Forest was optimized over 2 hyper-parameters, number of estimators and max depth. A number of estimators of 2000, with no max depth gave the optimal validation AUC of .9842.





IV. Results and Analysis

Once each model's hyperparameters had been optimized, the next step was to determine the highest performer from among them. We ran each of the models over the testing set,

and received the results seen in table 1.

Table 1

Model	AUC Score	Recall	Precision	Accuracy
Linear Discriminant Analysis	0.983247	0.770751	0.855263	0.999361
Logistic Regression	0.812179	0.624506	0.882682	0.999185
K Nearest Neighbors	0.912884	0.766798	0.873874	0.999389
Feedforward Neural Network	0.893119	0.786561	0.812245	0.999298
Support Vector Machine	0.93085	0.782609	0.818182	0.999305
Random Forest	0.965758	0.770751	0.886364	0.999417

As previously noted, all models have an absurdly high Accuracy score, making comparison based on this metric nearly meaningless. This is, again, due to how unbalanced the data set is between the classes, with only .17% of the transactions being fraudulent. With so few true positives, most predictions will be negative, and with most ground truth labels also being negative, the overall accuracy will always be extremely high.

Based on AUC score, the LDA model is the clear winner, with an AUC of .983247, as seen boxed in blue. The Random Forest model also had a relatively high AUC of .965758, with the SVM model coming in third. The Logistic Regression model performed comparatively poorly, with the lowest AUC value by a significant margin.

In terms of Recall, arguably the most important metric in this scenario due to the incredible risk associated with missing a fraudulent transaction, the FNN model was the highest performer, with a recall of .786561, boxed in green. The SVM model also performed quite well in terms of Recall with a score of .782609. The far and away worst performer over the testing set was again the Logistic Regression, with an abysmal Recall of .624506. As it turns out, the Logistic Regression does happen to have the second highest Precision score, and the FNN model, which had the highest Recall score, in fact has the lowest of all Precision scores.

All things considered, we would recommend that a credit card company looking to employ the most effective anti-fraud system possible should do so with an FNN model, as it will catch the highest portion of truly fraudulent transactions. If customers end up complaining about the program over-classifying transactions as fraudulent, and are annoyed by too many notifications on their phone asking to verify transactions that they had in fact made legitimately, the credit card company could consider transitioning to an LDA model. This model may miss a few more truly fraudulent transactions, but will annoy customers less with fewer mis-classified legitimate transactions and thus fewer phone notifications. While we believe it to be more important to catch as many fraudulent transactions as possible, this will ultimately come down to a judgement call by the organization, hopefully with input from their customers.

V. Conclusion, Challenges, and Future Work

In this paper we developed 6 different models for classifying credit card transactions as fraudulent or legitimate. The models were trained on a dataset of 284,807 credit card transactions made in Europe over two days in September 2013. After optimizing each model's hyper-parameters over the validation set, we employed the final models over the testing set to compare performance. The LDA model performed the best in terms of AUC, while the FNN model performed the best in terms of Recall. Which performance metric is deemed most important must, at the end of the day, be decided by the company itself, though we recommend placing the most emphasis on the Recall score, as missing a fraudulent transaction may be devastating to the victim.

A primary drawback of our study is in terms of the dataset used. With such an unbalanced divide between positive and negative classifications, the models may not have been trained optimally. Oversampling and undersampling could be utilized in the future in an attempt to balance this dataset, or new data can be collected that may be less unbalanced. It should be noted that in the real world fraudulent credit card transactions are indeed this rare, so it may be hard to find datasets with more even splits.

There are also more models that could be tried. For example, there are countless more possible combinations of FNN models still to test. Another strategy to employ next could be bagging boosting. Potentially using 5 of the 6 models discussed in this paper (dropping Logistic Regression to make an odd number) to vote would be a starting point.

I. References

[1] Dornadula, Vaishnavi Nath, and S Geetha. "Credit Card Fraud Detection Using Machine Learning Algorithms." *Procedia Computer Science*, vol. 165, 2019, pp. 631–641., doi:10.1016/j.procs.2020.01.057.

[2]Sun, Luke. "Credit Card Fraud Detection." *Medium*, Towards Data Science, 16 Oct. 2020, towardsdatascience.com/credit-card-fraud-detection-9bc8db79b956.

[3]ULB, Machine Learning Group -. "Credit Card Fraud Detection Data Set." *Kaggle*, 23 Mar. 2018, www.kaggle.com/mlg-ulb/creditcardfraud.

Statement of Responsibilities

Of the six models we developed and optimized, each group member was responsible for two. Yiran did the Linear and KNN models, Roya did the FNN and the Logistic Regression models, and Maxwell did the Random Forest and the SVM models. Roya cleaned the data, did descriptive statistics, visuals, plots, and data exploration, and initialized the format with which to report performance of each model. Yiran compiled each of our code files into a final document, organized it, made sure everything was clean, in the same format, well commented, and ran without errors, and then ran the document for a final training and output of our models in Jupyter. Maxwell wrote the report and added the visuals we would be using from the Jupyter file. We then collaboratively added comments to the report draft and finalized the project.