Credit Card Fraud Detection

Introduction and Background

In the information age, sensitive user data such as payment information has become increasingly digitized. This has created vulnerabilities in data safety. In this project, we intend to implement an automated and intelligent detection system for fraudulent transactions.

Problem Definition

The evolution of physical cash to stored information sacrifices security for speed. According to a Federal Trade Commission 2021 report, 88,354 credit card fraud instances resulted in more than $180 million in losses. By using machine learning to detect fraud, we aim to minimize these financial burdens.

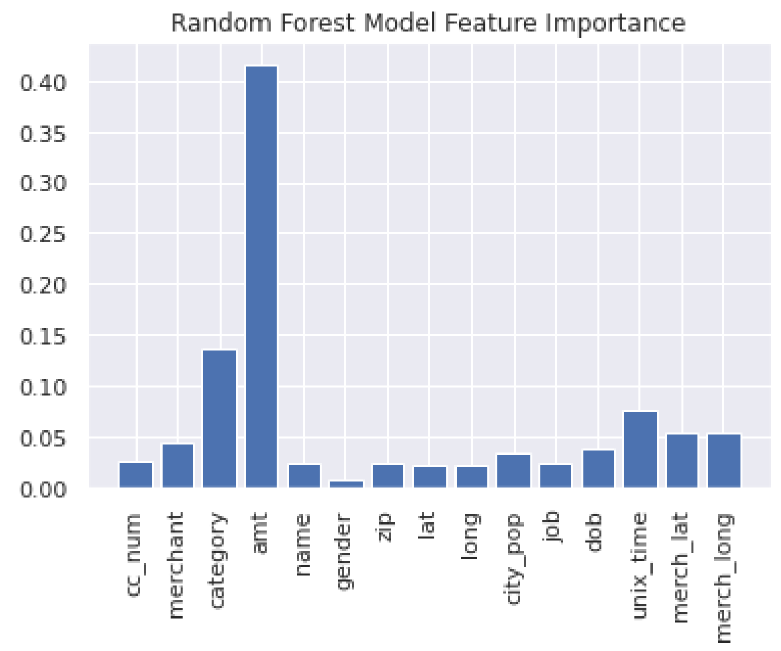
Data Collection

Our dataset is simulated consumer information by the MIT Licensed Synthetic Credit Card Transaction generator. It contains 555718 transactions, 22 features, and a single label, fraud vs. normal transactions.

Feature Selection & Data Preprocessing

In order to decrease the possibility of over-fitting in our models we limited the number of features used in our training dataset. Our first step was to do some preliminary feature reduction such as combining first_name and last_name into one feature name. Due to the surplus of qualitative features in our dataset, we first encoded all qualitative columns. Our encoding approach uses sklearn's preprocessing.LabelEncoder(), which encodes target labels with a value between 0 and n_classes - 1. This applied an integer representation of each unique value per feature.

Numerical data representation allowed us to implement a Random Forest Classifier to generate the most important features, measuring the average impurity decrease computed from all decision trees in the forest. We then determined that there were 8 relevant features that could be used based on the model below and our own predictions of what could affect the likelihood of fraud.



These are the features we decided to keep and their descriptions:

- amount - Represents the monetary value of the transaction.
- category - What kind of items were bought. We used one hot encoding to convert each category to a numerical label.
- unix_time - When the transaction was made.
- merch_lat - The latitude coordinate of the purchase.
- merch_lon - the longitude coordinate of the purchase.

We additionally retained the labels, indicating whether the transaction was fraudulent or not.

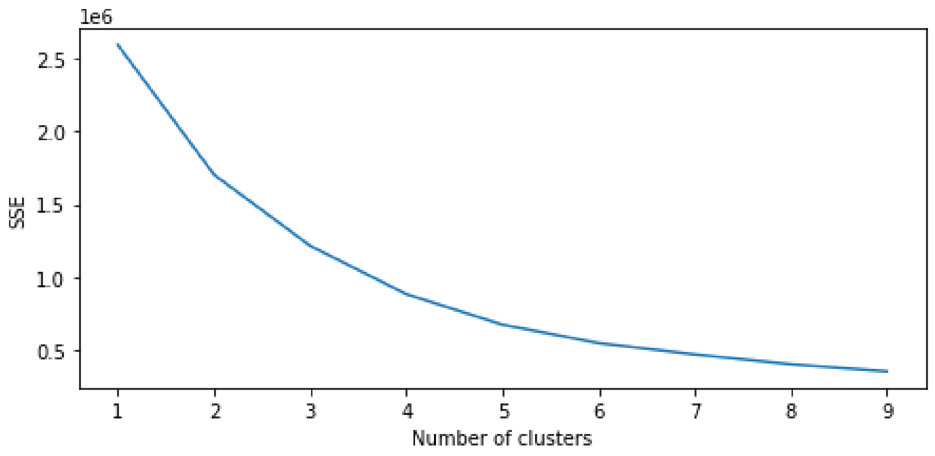
- is_fraud - True/false label that indicates whether a transaction is fraudulent.

Our feature selection was entirely dependent on the generated feature importance from the Random Forest Classifier. We only retained features with an importance of over .05, thus limiting the dimensionality of our data, decreasing runtime, but also restricting the nuance of a prediction. We were also wary of the vast increase in importance for the amount of the transaction. This stark deviation leads us to believe that determining fraud will be heavily dependent on the amount of the transaction.

Methods

Unsupervised Learning:

K-Means



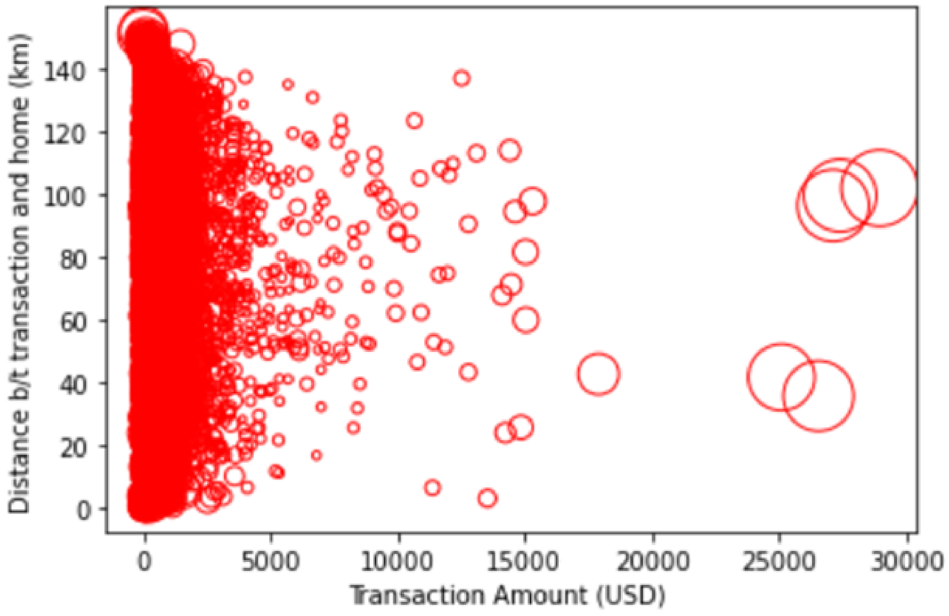
To find the best k-value for our clustering model we implemented the elbow method, shown in the graph above. Based on the graph we found that the ideal number of clusters for our dataset was at a k-value of 4. This was determined based on the decreasing slope of the graph at k = 4, indicating that the error between our predicted and actual values decreased.

The model algorithm was run using the amount value of each transaction along with the distance between the credit card holder's location and the coordinates of the purchase. This distance was calculated using the lat, lon, merch_lat, and merch_lon features.

Local Outlier Factoring (LOF)

For our second unsupervised learning model we implemented the Local Outlier Factor (LOF) algorithm with the associated scikit code template (scikit, 2022). Similar to our K-means algorithm, we clustered the data points based on the transaction amount and distance between the transaction location and home address which resulted in a large number of points clustered at the far left end of the plot with greater density and fewer points on the right end of the plot with greater distances between them. Then with the LOF algorithm we fitted the model for outlier detection. For the number of neighbors considered we used n_neighbors = 20. This number is used as the generic value given that we do not have enough information regarding our data set to determine a minimum and maximum value for the number of data points we should expect our clusters to contain. According to outside reports using the LOF algorithm an n_neighbors value of 20 is generally successful. For our expected contamination percentage, we used the true value of fraudulent cases in our data set as a percentage based on the actual number of cases used in our training data. This value came to be 0.0135%. The contamination value is the proportion of outliers we set for the data set and is used to create a threshold for the outlier scores of the data points. We decided on using the true percentage of fraud cases from our training data as the contamination value because we believed that would create a more accurate prediction model.

```
LocalOutlierFactor(n_neighbors = 20, contamination = percent_fraud)
```



Supervised Learning:

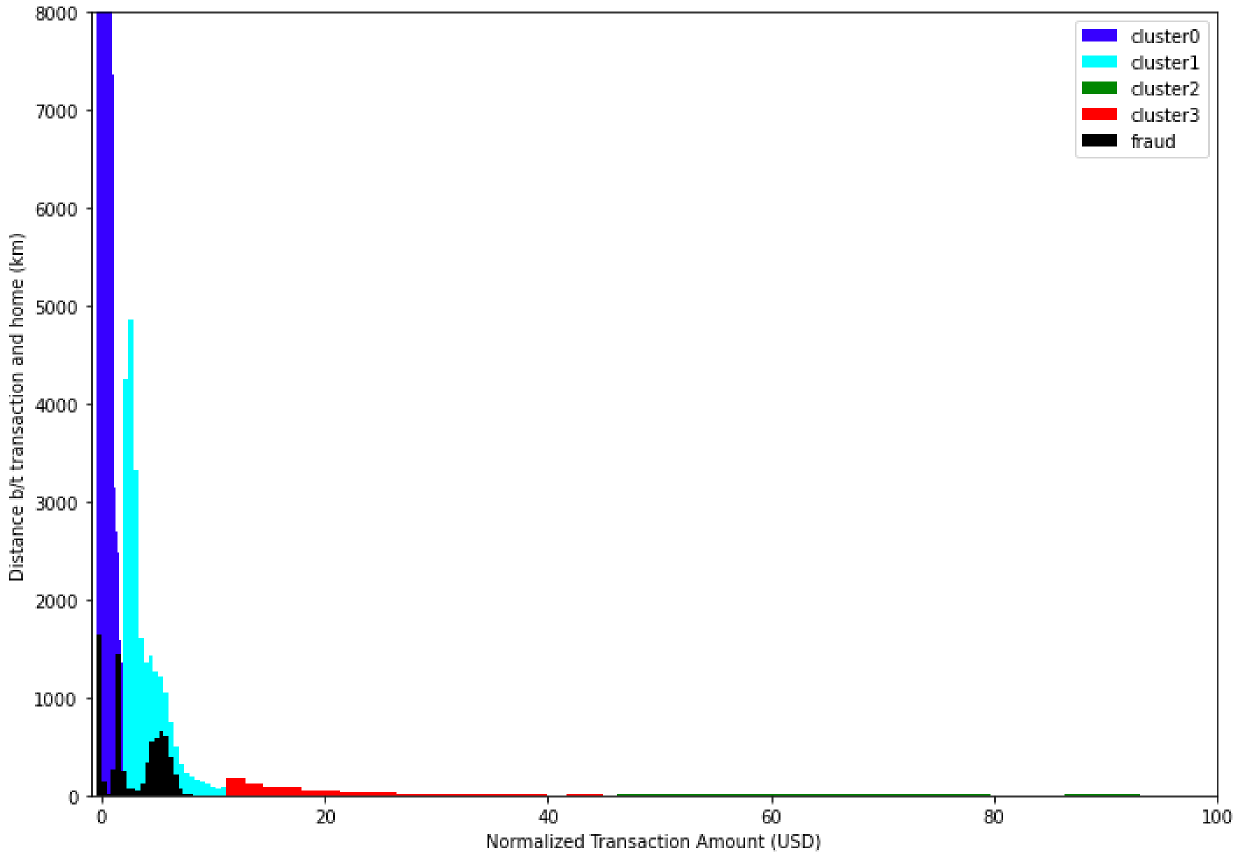
Artificial Neural Networks (ANN)

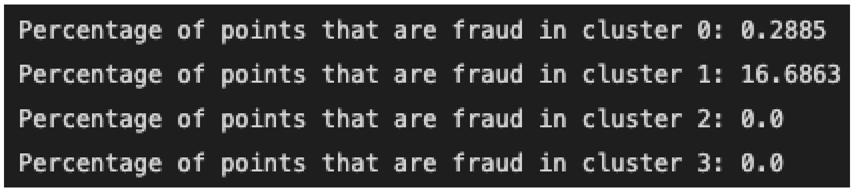
We will construct an ANN model in order to view any related patterns between fraudulent credit card transactions. Based on this we can create a predicted value range of transaction amounts for cases of fraud and use this as a basis for detection. An ANN model is well suited for this problem given how it learns using previously established patterns, and we will use the backpropagation algorithm to improve the neural network (Mittal et al., 2019).

Results and Discussion

Unsupervised Learning:

K-Means





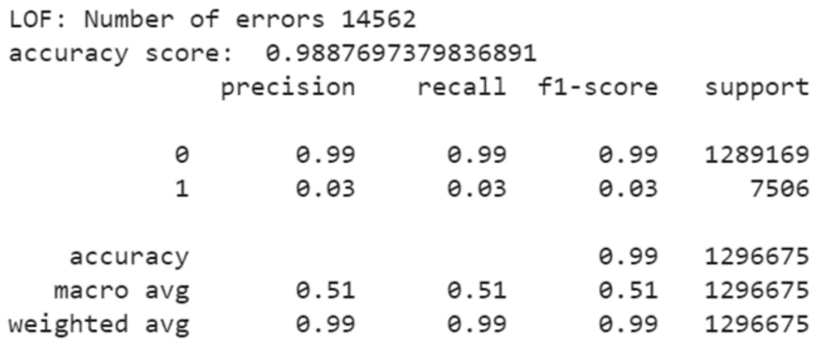
```
Percentage of points that are fraud in cluster 0: 0.2885
Percentage of points that are fraud in cluster 1: 16.6863
Percentage of points that are fraud in cluster 2: 0.0
Percentage of points that are fraud in cluster 3: 0.0
```

The above figure is the end model using the k-means with a k-value of 4. Note that some of cluster 0's points go off the graph in the y-direction, exceeding 50,000 km. Each cluster differs based on distance and amount values. Cluster 0 consists of low transaction amounts and high distance between the transaction and where the person lives. Cluster 1 consists of low transaction amounts with medium distance between transaction and home. Cluster 2 is made up of medium amount values with a much smaller distance between transaction and home, Cluster 3 is similar in that it has smaller distance values but consists of all high amount value transactions.

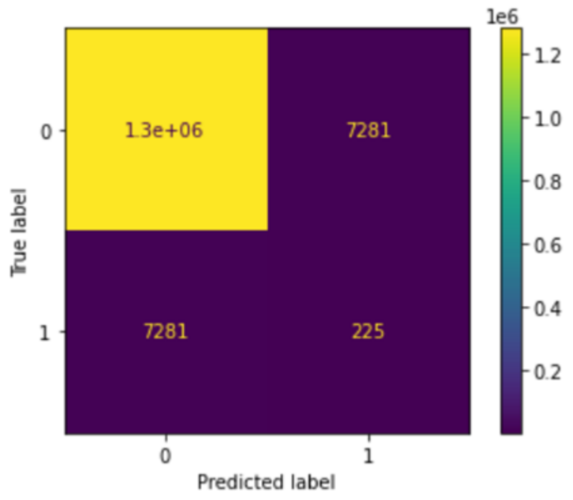
Fraud cases are marked in black on the figure above. All fraudulent transactions fall in either cluster 0 or 1, suggesting that fraud is less likely in the higher transaction amount clusters of 2 and 3. A staggering 16.7 % of points in cluster 1 are fraud, which is much higher than the next closest in .28%. Overall, our K-Means model did a reasonable job in classifying what types of transactions may require closer attention and which are most likely not fraud as clusters had differences in fraud percentage, but it cannot rule with certainty whether a transaction is fraud or not based on cluster id alone. Even with the stark differences between clusters, the cluster that had the most fraud still only has a 16.7% fraud rate.

LOF

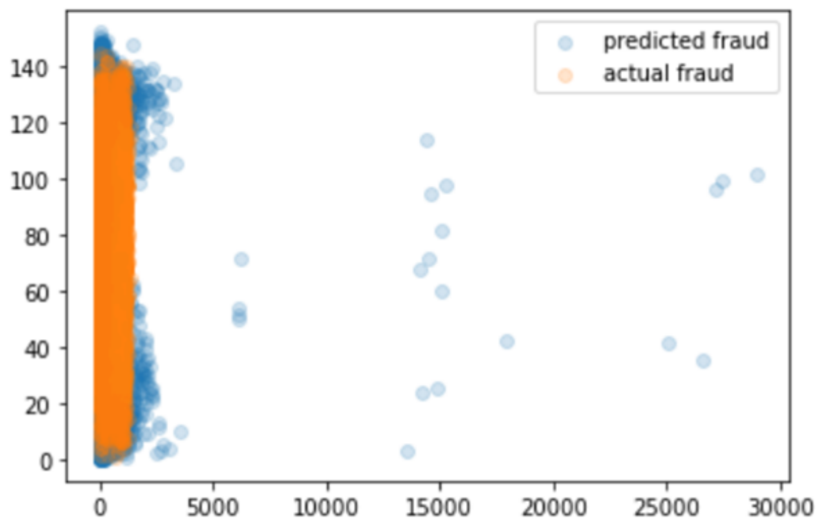
Our LOF algorithm yielded an accuracy of approximately 98.9% and only 14562 error predictions out of the 555718 transactions in our training data. This was a remarkably high accuracy for this learning model but given the setup of our training data and the function of the LOF algorithm this was still within our expectations. The contamination value we used in our model was very small given that we used the actual percentage of fraudulent cases from our data set. As a result, the prediction model is much more likely to lean towards predicting that a transaction is not fraud then that it is. Therefore, since nearly 99% of our dataset consisted of not fraudulent transactions the model was more likely to produce a correct prediction for each transaction. However what is more telling is the breakdown of the prediction metrics for our LOF model as seen in the two figures down below.


```
LOF: Number of errors 14562
accuracy score:  0.9887697379836891
              precision    recall  f1-score   support

           0       0.99      0.99      0.99   1289169
           1       0.03      0.03      0.03      7506

    accuracy                           0.99   1296675
   macro avg       0.51      0.51      0.51   1296675
weighted avg       0.99      0.99      0.99   1296675
```

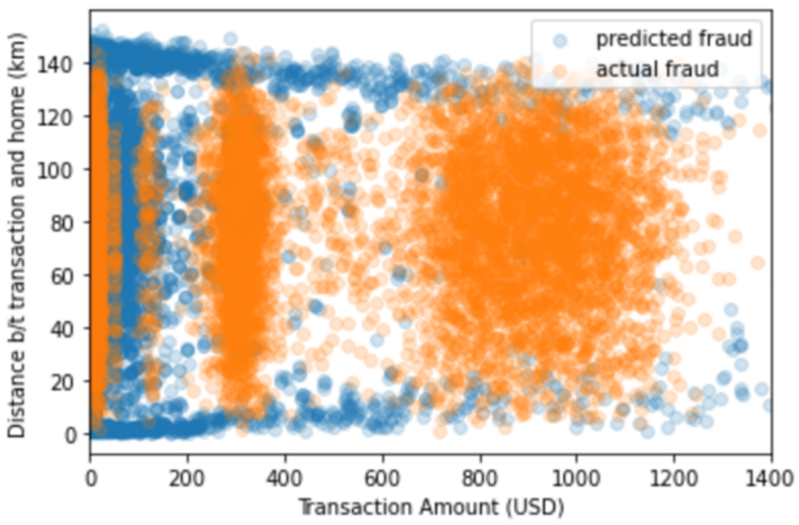
For not fraudulent cases, the model is able to predict with a precision of 0.99. This falls within the expectations of our model given the value of our contamination value. However, when attempting to predict fraud transactions the precision of our LOF model falls to 0.03. That means that the model is very unreliable when trying to determine cases of fraud.



As also supported by the table above, only 225 of the fraud labeled transactions were correctly identified from the model in comparison to the 7506 fraud transactions actually present in the dataset. In other words, although our model has a high accuracy rate overall, it can be said that it is in fact a very poor model for determining cases of fraud. When trying to understand why this may be we further examined the distribution of predicted versus actual cases of fraud from the LOF model in the plots below.



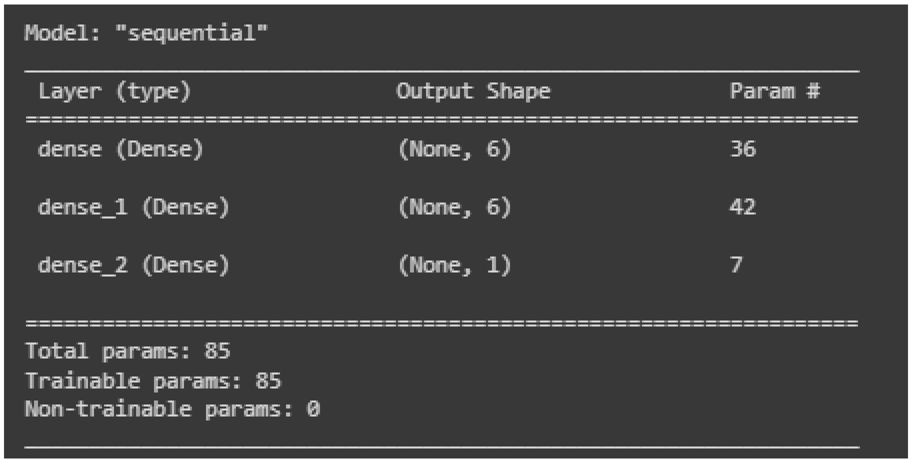
This first model was difficult to analyze given the majority of data points being clustered on one end of the plot. Therefore, to better visualize our results we focused on the far left portion of the data plots as seen in the revised plot below.



The above plot demonstrates that many of the predicted fraud cases are determined to be in the upper and lower regions of the y-axis and spread out across the x-axis. This falls in line with our first assumptions about the model. We had believed that outliers would more likely be cases of fraud and assumed that the LOF model would be accurate in predicting such results. What the above model shows instead, is that actual cases of fraud are more clustered within the middle of the plot, indicating that fraud transactions are more similar to actual transactions in appearance than outliers.

ANN

In implementing the Supervised Learning Artificial Neural Network, we built two different models. The first model we analyzed was composed of three Dense Layers, displayed in the summary image, Figure X. The first dense layer, our input layer, contains 6 hidden neurons in the second hidden layer, a ‘relu’ activation function, and input dimensions of 5. We selected this initial input dimension as we have five features within our model. Additionally, we selected the ‘relu’ activation function, a nonlinear function which takes the maximum of its input and 0, in order to avoid any vanishing gradient issues. We added an additional Dense layer with an output of 6 hidden neurons in the next hidden layer and a ‘relu’ activation function. Finally, our output layer consisted of a single output dimension and a ‘linear’ activation function. Selection of a ‘linear’ activation function was meant to serve as a comparison point for the next implementation of ANN we will discuss.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 6)                 36

 dense_1 (Dense)             (None, 6)                 42

 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 85
Trainable params: 85
Non-trainable params: 0
_________________________________________________________________
```

Figure X

We ran our sequential model with our training data, consisting of 80% of our total data. Implementing a batch_size of 10 and 5 epochs, we generated a resoundingly consistent and high accuracy per each epoch. This training accuracy was concerning, as there is an unequivocal correlation between high training accuracy and overfitting. Figure Y displays the accuracy per each epoch, resting at a steady .99.

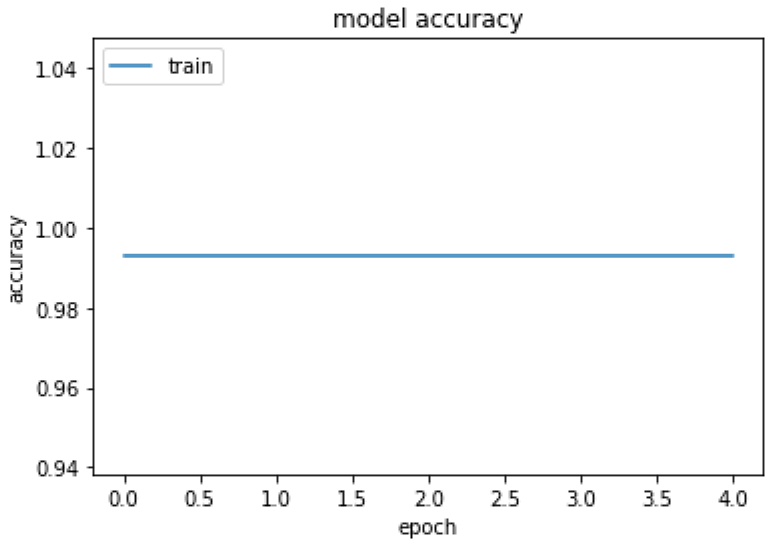
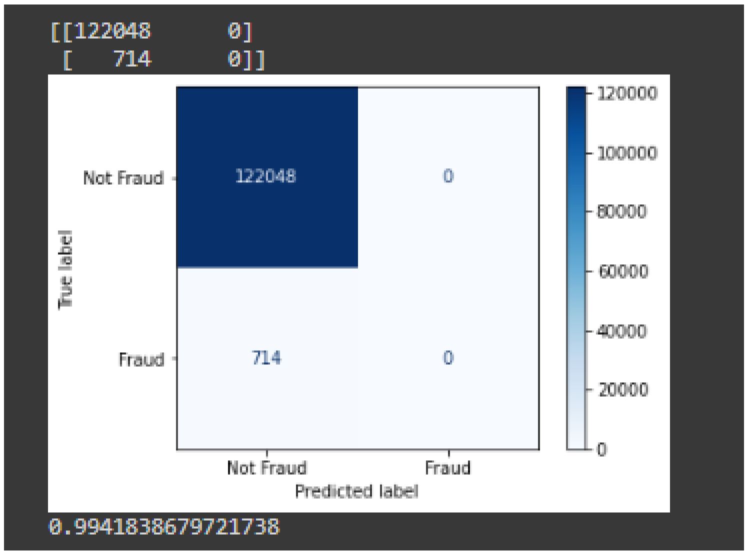


Figure Y

The final step was to test our trained model. We used our classifier to predict our test data, assigning a 1 to the result (fraud) if the prediction value was greater than .5 and a 0 otherwise. This resulted in a shocking confusion matrix. Our model resulted in 99.4% test accuracy, generating 714 or .05% false negatives, 0% false positives, and overall not correctly identifying a single fraudulent transaction. Thus, the precision percentage is 0%, indicating there were zero total true or false positives. Additionally, the recall percentage is 0%, as there were zero total true positives or false negatives.

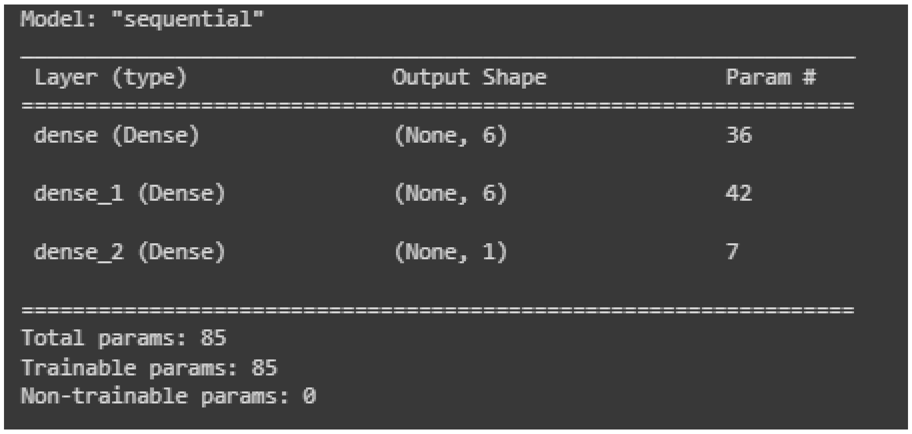


```
[[122048      0]
 [   714      0]]
```

```
0.9941838679721738
```

Figure Z

With these results, we decided to adjust our ANN model by changing layer types, the output layer activation functions, and the number of batches/epoches we compiled with. Within the second model, we implemented three layers. The input layer was Dense with an output of 6 hidden neurons and an activation function, ‘relu’. A secondary layer was added, again Dense, with a relu activation function. Finally, the output layer consists of a single output neuron and a sigmoid activation here. From the first model, we switched this model’s activation function to be non-linear. The full model is displayed below, Figure A.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 6)                 36

 dense_1 (Dense)             (None, 6)                 42

 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 85
Trainable params: 85
Non-trainable params: 0
```

Figure A

With this model, we fit our training data with a batch size of 15 and 10 epochs, resulting in the following (Figure B) accuracy graph. Each epoch generated values ranging from .9942 to .9963, a resounding high accuracy and highly similar to our first model.

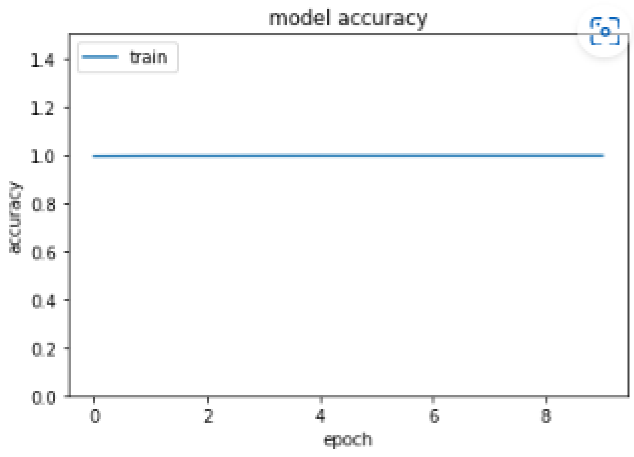
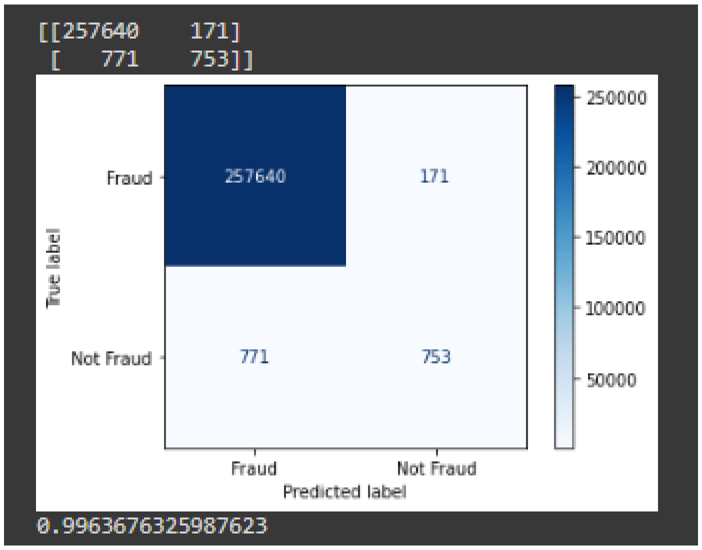


Figure B

However, we noticed more variation with precision and recall statistics. Figure C displays the confusion matrix generated with this model. The overall accuracy determined was 99.6%, generating 771 false negatives, 171 false positives, and overall correctly identifying 753 fraudulent transactions. Thus, the precision percentage is 81.5%, indicative of true or false positives. Additionally, the recall percentage is 49.4%. Both of these are an improvement from the previous model.



```
[[257640    171]
 [   771    753]]
```

```
0.9963676325987623
```

Figure C

Conclusion

We created three different models in order to detect instances of fraudulent credit card transactions: two unsupervised and one supervised. In the case of both unsupervised models, the outcomes were not sufficient in terms of being effective prediction models. In the k-means clusters, while the clusters did differ in fraud percentage, there was not a significantly high chance of fraud in any given cluster. The LOF model did not perform better though that is not immediately obvious based on its overall accuracy. Despite being accurate in predicting cases of 'not fraud', the model had a very low precision when trying to determine fraud transactions. In the first ANN model, the results were similar to the first two models with the algorithm not correctly identifying any of the fraudulent transactions. The second version of the ANN model fared much better however it still did not perform as well as needed to accurately detect fraud.

Through the outcome of these three models we've concluded that the reason none of them performed well was due to issues with our dataset. In both the training data and larger dataset, the number of transactions that were not fraud far outweighed the number of transactions that were fraud. We believe that this discrepancy is what caused the prediction models to be so skewed in their precision and accuracy when trying to detect fraud. For future research purposes when confronting similar problems it would be best to use a more varied dataset with a closer to even ratio of fraudulent a 'not fraud' labels. Were we able to start this project again, our models would have benefitted from using real-life values instead of the generated transactions we had.

References

Mittal, S., & Tyagi, S. (2019, January). Performance evaluation of machine learning algorithms for credit card fraud detection. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 320-324). IEEE.

Outlier detection with local outlier factor (LOF). scikit. (n.d.). Retrieved October 6, 2022, from https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html#:~:text=The%20Local%20Outlier%20Factor%20(LOF,lower%20density%20than%20their%20neighbors.

Proposed Timeline:

Our Gantt chart can be found here

Contribution Table:

Team Member	**Contribution**
Samantha Burger	Model coding, Results evaluation and analysis, Final Report
Olivia Lawson	Model coding, Results evaluation and analysis, Final Report
Munim Riddhi	Model coding, Final Report
Rob Schleusner	Model coding, Final Report

Samuel Wysocki	Model coding, Results evaluation and analysis, Final Report