Machine Learning: Project Report

Morgan Gant

Rachael Doyle

December 11st, 2022

**Abstract:**

The Fraud Detection dataset consists of behavioral actions customers have while using the self checkout app, using these actions the company wants to predict what customers are most likely to be committing fraud. The goal of this study is to determine what type of predictive model performs the best, or classifies the transactions correctly, using the given variables. It’s important to find a model that has a good performance because it can hurt the company if it accuses innocent customers of committing fraud. There will be several model networks tested to find the best model and parameters to train the model to correctly classify transactions.

**Table of Contents**

**Introduction 3**

**Data Description 3**

**Project Tasks 3**

**Analysis**

Initial 5

Multi-Layer Perceptron and Support Vector Machines 9

Random Forest 10

XGBoost 12

Ada Boost15

**Conclusion 19**

**References 20**

**Introduction:**

It’s important to protect people’s online lives. A few ways cybercrimes can occur is in the form of hacking: illegally sending instructions to another computer to gain access to personal information, child pornography and abuse: criminals seek minors by messaging systems in hopes of collecting child pornography, and piracy: violating copyrights and downloads (Bandakkanavar). When it comes to online fraud, data mining and statistical analysis have been significant tools in decreasing this issue. Companies can use decision trees, machine learning, cluster analysis, etc. to scan transactions to identify patterns and spot fraudulent transactions (Volkov). With online checkout becoming popular, companies need to know their customers and their behaviors as best as they can. With this data set, we can predict if a transaction will be fraud. Consider a couple important questions: What variables are significant? How should accuracy be evaluated? What machine learning model will predict the best outcomes? Are there any restrictions?

**Data:**

The Fraud Detection at Self-Checkouts in retail data set was taken from the 2019 Data Mining Cup Competition. The task was to use this historical data to create a model that will correctly classify transactions as fraudulent or not fraudulent. This data contains customer transaction information, the only thing related to the actual customer is their Trust Level (variable trustLevel).

**Row and Column Description:**

There are two separate datasets being used, train and test. Train consists of 1,879 participants (rows) transaction information variables (columns), and Test consists of these same variables but for a total of 498,121 participants. Below is a detailed table of each variable’s name, description, and value range (because all variables are integers).

|  |  |  |
| --- | --- | --- |
| ***Column Name*** | ***Description*** | ***Value Range*** |
| **trustLevel** | Customers individual trust level | {1,2,3,4,5,6}  6 is the highest |
| **totalScanTimeInSeconds** | Total time in seconds between the first and last product scanned | Positive whole number |
| **grandTotal** | Grand total of products scanned | Positive decimal number with maximum two decimal places |
| **lineItemVoids** | Number of voided scans | Positive whole number |
| **scansWithoutRegistration** | Number of attempts to activate the scanner without scanning anything | Positive whole number or 0 |
| **quanityModifications** | Number of modified quantities for one of the scanned products | Positive whole number or 0 |
| **scannedLineItemsPerSecond** | Average number of scanned products per second | Positive decimal number |
| **valuePerSecond** | Average total value of scanned products per second | Positive decimal number |
| **lineItemVoidsPerPosition** | Average number of item voids per total number of all scanned and not canceled products | Positive decimal number |
| **fraud** | Categorization of fraud (1) or not fraud (0) | {0,1} |

**Project Tasks:**

We have selected this data set to predict if transactions are fraudulent or not. Through the creation of variable selector models, predictive models, and visualizations we will address the following questions:

* What variables are significant?
  + Create a lasso variable selector to see which variables are significant when predicting on fraudulent transactions
  + Create a Random Forest Importance variable selector to compare variables to the one’s the Lasso model came up with
    - Then use these variables to then create models
* How should accuracy be evaluated?
  + Determine between mse, recall, f1-score
    - What measurement best portrays the accuracy?
* What machine learning model will predict the best outcomes?
  + Logistic regression, linear regression, lasso, adaboost, random forest

**Initial Exploratory Analysis:**

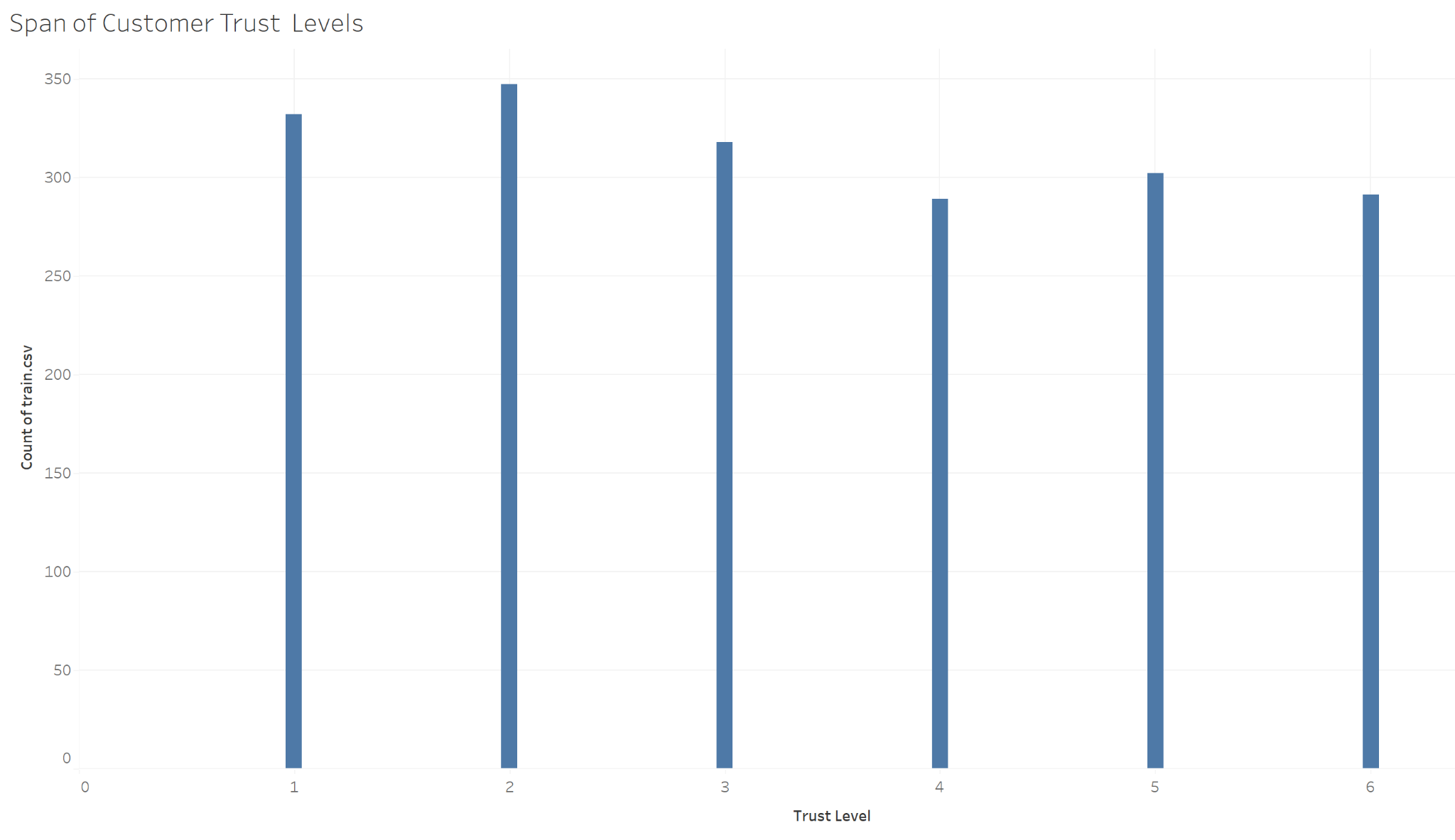
*Figure 1*

Figure 1: It’s important to know your customer as well as their behaviors. With this being the only customer related variable, it’s important to see how well the customers in this training study are trusted. The range is from 1 to 6, with 6 being the best score. It wasn’t explained how the trust level was computed. From the graph, we can see there are more lower trusted customers than higher trusted customers but not by very many. The difference between each group is less than 100.

*Figure 2*

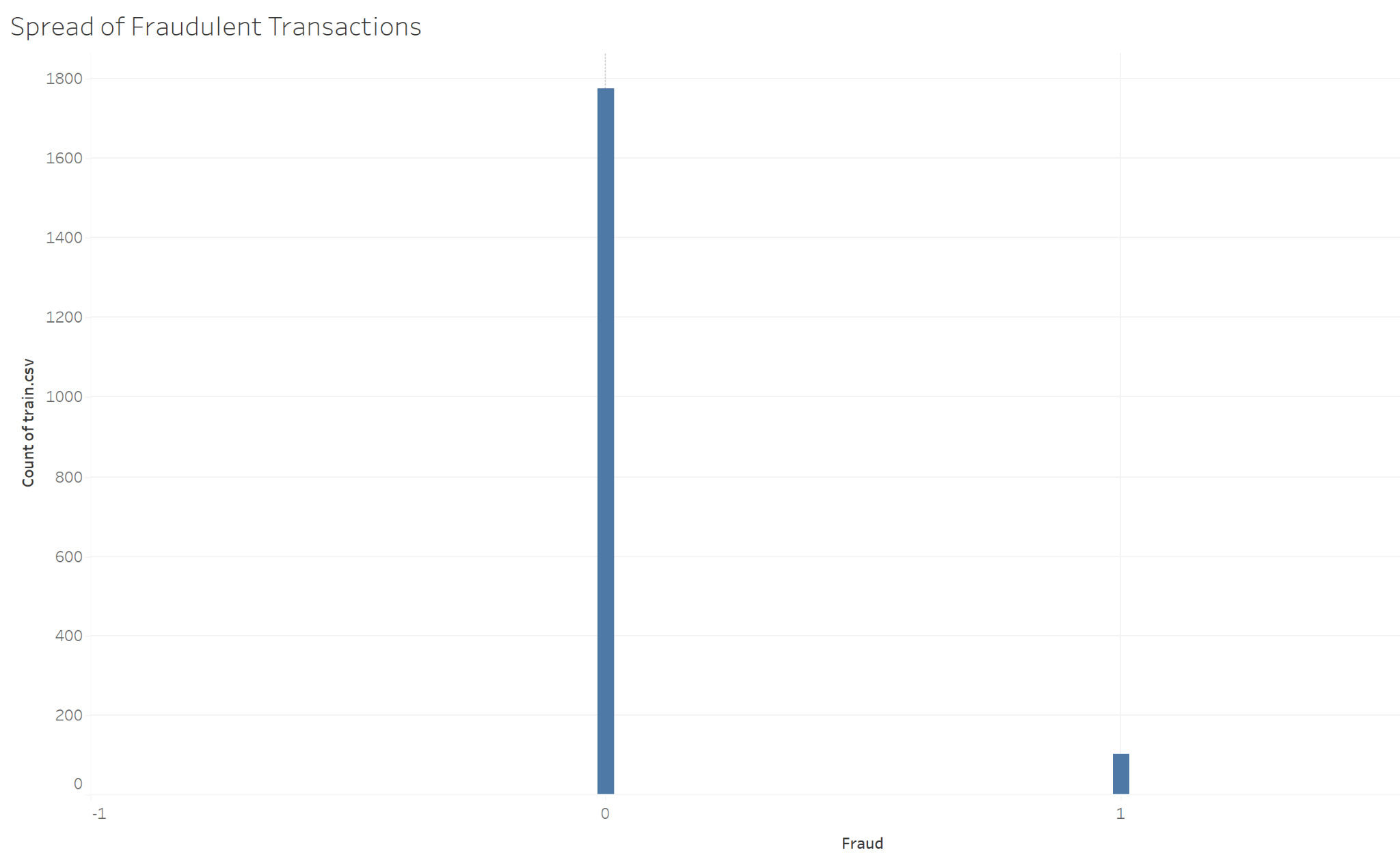


Figure 2: This chart shows the amount of fraudulent transactions total. It is being shown using a 0-1 scale, 0 being not a fraudulent transaction, 1 being a fraudulent transaction. We can see that this number is low compared to the amount of data that we have. In total we have 1,879 transactions, and we have a little over 100 fraudulent charges.

*Figure 3*

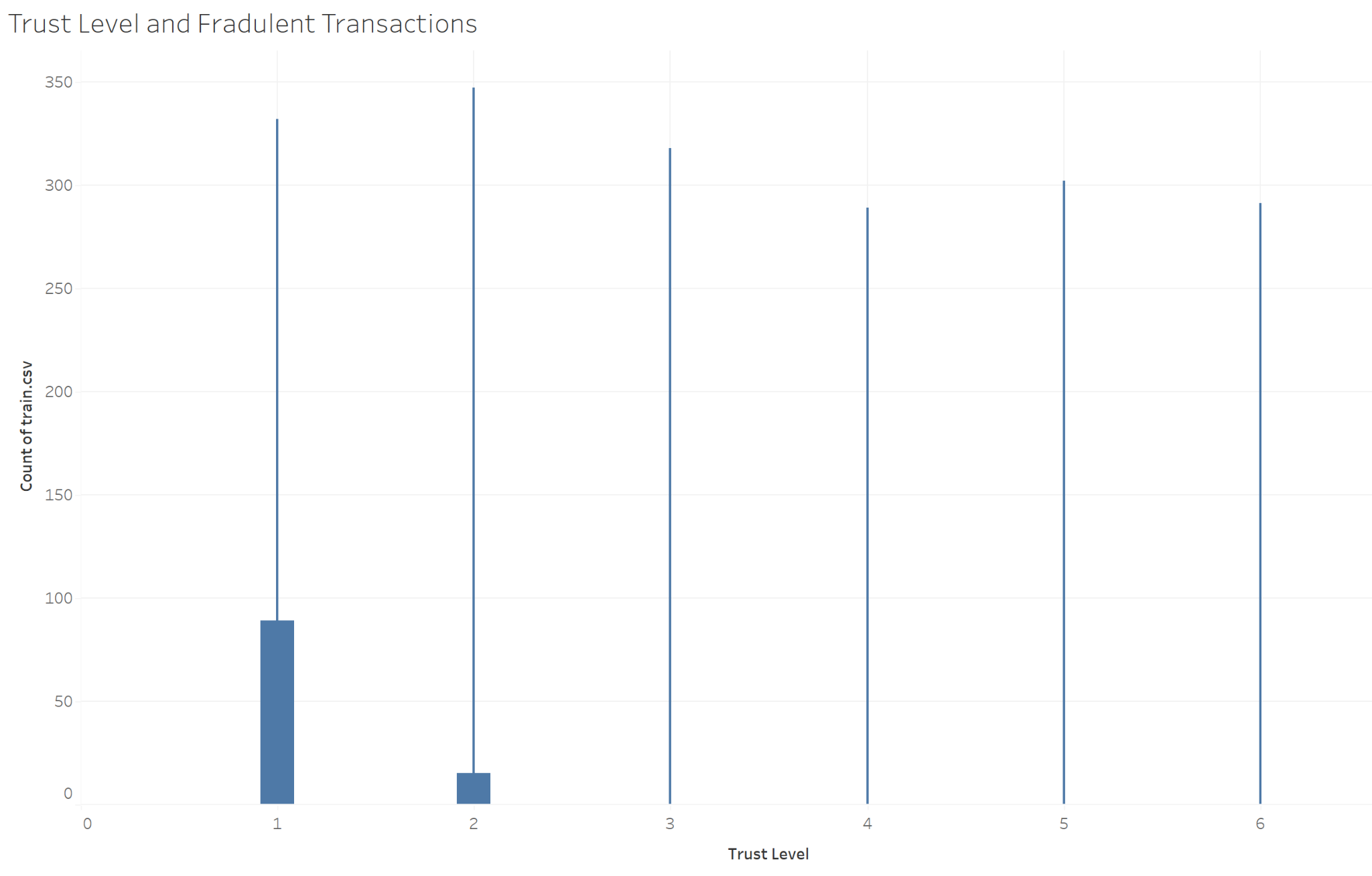
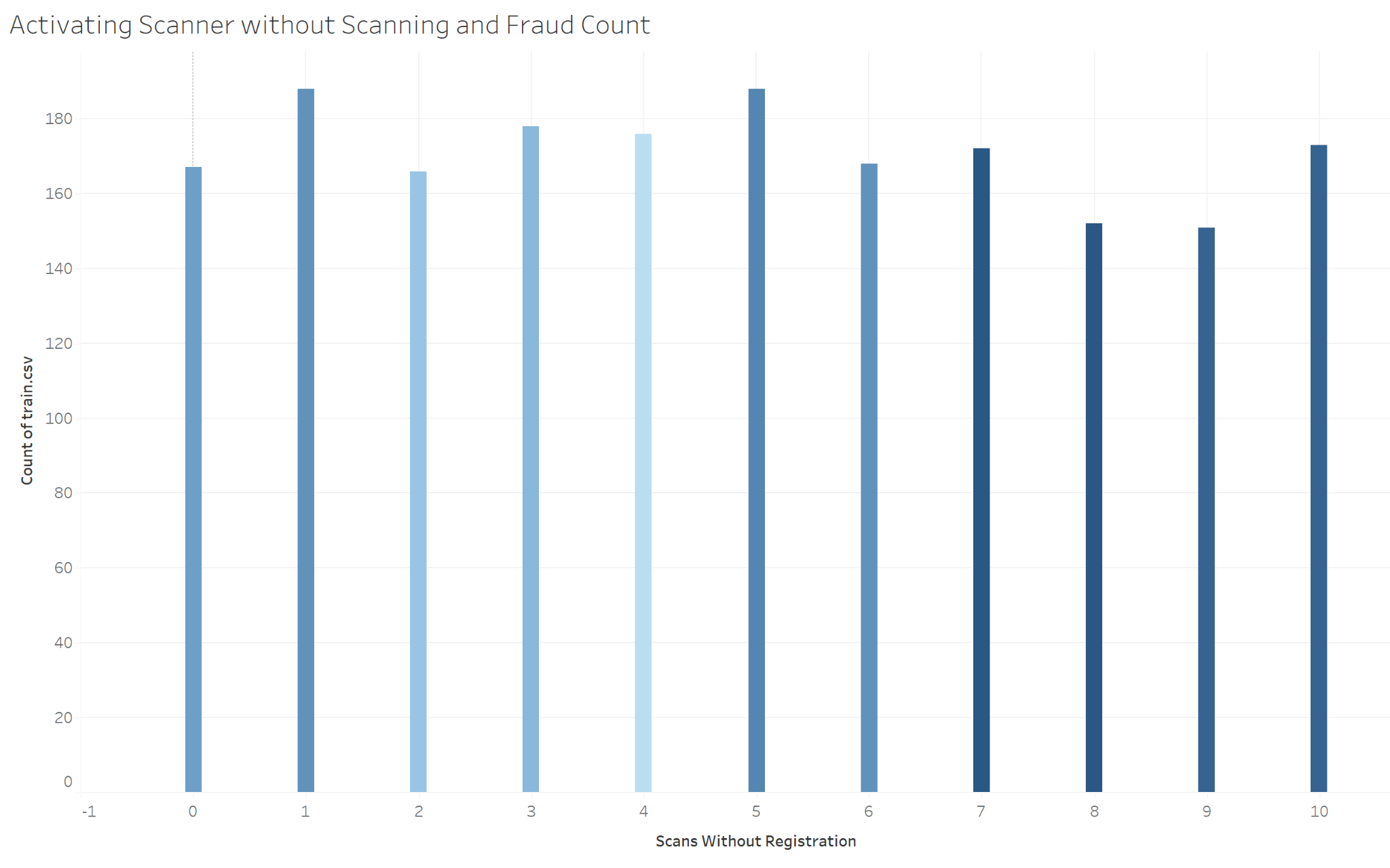


Figure 3: This chart represents the range of trust level, 0-6, and how many fraudulent transactions are reported for each trust level. We can see that the only trust levels with fraudulent transactions are 0 and 1, which are the lowest levels of trust, and the most being at trust level 0.

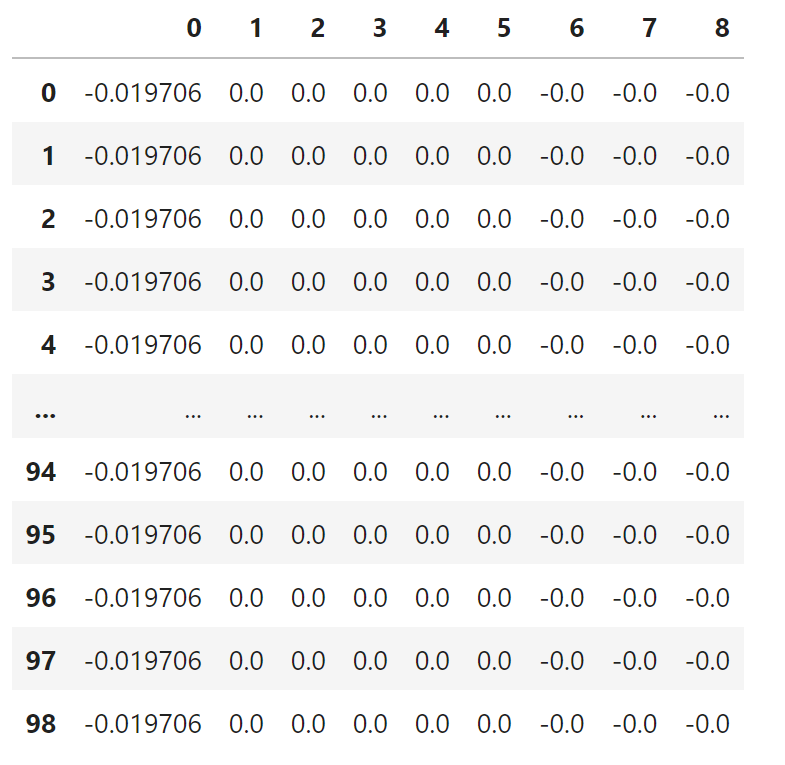
*Figure 4*

Figure 4: Darker blue represents more fraudulent transactions, where lighter blue represents less fraudulent transactions. This graph shows that the more times a scanner is activated without anything being shown, it has a higher likelihood of being fraudulent. There is some darker shading between 0 and 1 times the scanner was activated but the most dramatic effect is between activations 5 to 10 times. This gives the evidence that when moving forward with modeling, it could be an important variable in model training.

Each variable could be compared to Trust Level and each other. It would be more efficient and accurate to run two tests that are made to pick out important variables. First is Lasso. Lasso uses an L1 penalty which means it will limit the size of the coefficients. When the tuning parameter is large, it can lead to some coefficients becoming zero and then be taken out of the model. The steps taken to create these results:

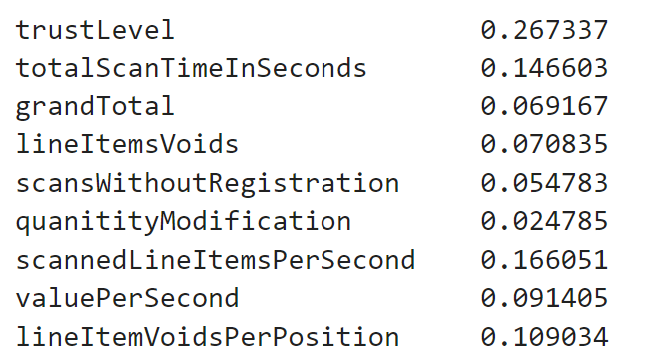
1. Import Data
2. Make sure there are no null values and if there are, drop them
3. Define the x and y variables
4. Split the train data set in 80/20 → 80% to train and 20% to test
5. Define a list to store appended lasso coefficients in
6. Estimate the lambda
7. Extract the Lambda
8. Build the Lasso Model
9. Convert list to A data frame

Below are the results:

****

As you can see, the lambda was large enough to shrink the majority of the variables down to zero. At first this could be very deceiving because it looks like all but one of the variables is insignificant. One thing to keep in mind though, is that Lasso assumes your data has a linear association. So, your results will be inaccurate if there isn’t a linear association in the data. The next test to run is the Random Forest Importance feature selector!

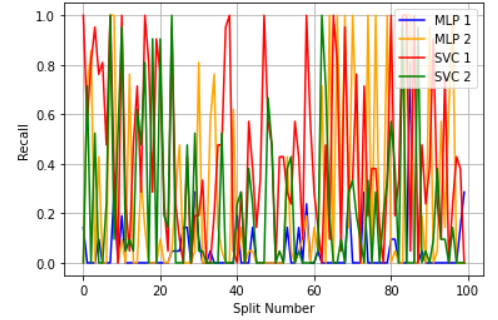
This model is used to check and determine what variables in the data are deemed important. This importance is shown through higher values. We are going to be using the top five important variables which are: trustLevel, scannedLineItemsPerSecond, totalScanTimeInSeconds, lineItemVoidsPerPosition, valuePerSecond. Below are the results of the Importance Selector:

****

The Lasso and Random Forest results might not always match and that’s okay because of the assumption lasso has on the data. We are going to move forward with the results of the Random Forest test. We will build our models using the variables: trustLevel, scannedLineItemsPerSecond, totalScanTimeInSeconds, lineItemVoidsPerPosition, valuePerSecond.

From here we will use our train data set to build models such as Multi-Layer Perceptron, Vector Support Machines, AdaBoost, Random Forest, etc. to see which model reports the best accuracy measurements. While doing this we will have to adjust tuning parameters on models to see how the model should be set up to begin with, then we can compare the models to each other.

**Multi-Layer Perceptron and Support Vector Machines**



|  |  |
| --- | --- |
| **Model** | **Recall Score** |
| MLP 1 | .04523809523809524 |
| MLP 2 | 0.24904761904761905 |
| SVC 1 | 0.43 |
| SVC 2 | 0.21904761904761905 |

Looking at the graph above, there were two types of models tested, each having two models with different parameters. The Multi-Layer perceptron models had one model with one single hidden layer with 4 neurons (hyperbolic tangent as the activation function) and softmax as the activation function for the output, the stochastic descent gradient as the method to estimate the weights (optimizer = ’sgd’) and metrics = [’accuracy’], epochs = 100 and batch size = 500 to build the model. The second MLP model had one single hidden layer with 4 neurons (ReLU as the activation function) and softmax as the activation function for the output, the stochastic descent gradient as the method to estimate the weights (optimizer = ’sgd’) and metrics = [’accuracy’], epochs = 100 and batch size = 500 to build the model.

Also featured in the graph is the Support Vector Machine models, one model used ‘rbf’ as the kernel and the other used ‘poly’.

By looking at the graph, you can see that all four models will have low average recalls because of the consistent fluctuation of recall values. This taken into consideration with our future model recall scores, we decided that these models are insignificant to our overall study on fraud prediction.

**Random Forest Results**

With random forest testing having many variations of parameters to possibly test we put them all into a data frame, seen below, to get the different combinations we would want to see. In total there were 12 different combinations between 100 trees, 300 trees, 500 trees, and 1000 trees, along with depths of three, five, and seven. Using this data frame a loop was created to run each model 100 times, with varying splits each time.

|  |  |
| --- | --- |
| **Original Data-frame** | **Result Data-frame** |
|  | **Recall-**  **Accuracy-** |

All of the models with a depth of three had on average a recall score around .89, then the models with depths of five had on average values of .91 for recall, and models with depths of seven had recall values between .89 and .92. Next we have to take into consideration accuracy scores. Looking at the model that has the best recall, which was a model with 1000 trees and a depth of 5, this model has an accuracy of .87, which is close to the average of all models seen above. To determine what the accuracy should be, it would depend on the business standards, but for this study we will continue with model 10, a Random Forest model with 1000 trees and a depth of 5 as the best model for this testing group.

**XGBoost Results**

The result data frame has numbered columns that match the data frame with the list of trees. The results data frame shows many statistics but what we are focused on is the average recall and accuracy that was computed for each model (row-mean). XGBoost had 36 models that were tested. All the same combinations of models from Random Forest were used, except this type of modeling also takes into consideration learning rates, which are 0.001, 0.01. and 0.1 that were tested with those 12 previous models, thus giving us our 36 models. Using this data frame, a loop was created to run each model 100 times, with varying splits each time.

|  |  |
| --- | --- |
| **Original Data Frame** | **Results Data Frame** |
|  | **Recall-**        **Accuracy-** |

A data frame with all the trees being tested are above, for XGBoost there were 36 models run. For XGBoost, there are many models with the same recall score of 1. However, seeing these results we decided to take accuracy into account, and all the models with Recall values of 1, had extremely low accuracy scores, in the range of 5.5e-2. This means that we would not choose any of these to predict fraud, because accuracy is so low. We then look for the model with the next highest recall scores, .93, which is seen with model 10, 300 trees a depth of three and a learning rate of .01. This model also has a higher accuracy score of .89, therefore this is the best XGBoost model for predicting fraud.

**AdaBoost Results**

AdaBoost also had 36 models that were tested. Like XGBoost, this type of modeling takes into consideration learning rates. The same model combinations from XGBoost were tested with this type of model. Using this data frame, a loop was created to run each model 100 times, with varying splits each time.

|  |  |
| --- | --- |
| **Original Data Frame** | **Results Data Frame** |
|  | **Recall-**          **Accuracy-** |

For the AdaBoost models, there were 35 models run. All 35 models had recall scores within similar ranges, and the accuracy scores were also similar. However, the model that stood out as best was model 30 because it had the slightly higher recall value of .591, and when looking at the accuracy score paired with this model we see a higher score as well, of .9185. Therefore, even though the models are similar overall, the best AdaBoost model for predicting fraud is model 30, which consists of 1000 trees, a depth of 5 and a learning rate of .1.

**Concluding Results:**

Taking the best models from each of the modeling types we have narrowed our choices down to a Random Forest model with 1000 trees and a depth of 7 for a recall score of .92 and an accuracy score of .87. An XGBoost model with 300 trees, a depth of 3, and a learning rate of .01 for a recall score of .93 and an accuracy score of .89. Along with an AdaBoost model with 1000 trees, a depth of 5, and a learning rate of .1 for a recall score of .591 and an accuracy score of .9185.

After comparing these three models, the conclusion is that the XGBoost model with parameters of having 300 trees, a depth of 3, and a learning rate of .01 is the best model overall for detecting fraud. As seen earlier this was not the highest recall score, we saw out of all the XGBoost models, however taking into consideration the accuracy scores this became our best choice of XGBoost models, and the overall models.

To improve our results, we could alter more parameters, execute more iterations, and even add more types of models. Being that there are so many different types of models, with different parameter combinations, there could be an infinite number of models possible, leading us to the conclusion that we are never certain what model is perfect in each scenario.

Throughout this process we learned the best way to write our models and perform them, through trial and error. One mistake we originally made was running the models on different splits causing the issue that the models could not be compared because they were not using the same data splits. This is because splits are completely randomized each iteration the models are not being trained the same way, creating inconsistent results. After these trials and errors, we were able to clean up our code and run the models correctly, giving us organized results to present.

**Works Cited**

Bandakkanavar, Ravi. “Causes of CyberCrime and Preventitive Measures”.  *Krazytech,* 18 June

2022,<https://krazytech.com/technical-papers/cyber-crime>

Volkov, Michael. “Fraud Detection: New Technologies and Analytics (Part II of III)”. *Volkov*

*Law,*11 December 2018, <https://blog.volkovlaw.com/2018/12/fraud-detection-new-technologies-and-analytics-part-ii-of-iii/#:~:text=Data%20mining%20and%20statistical%20analysis,patterns%20and%20detect%20fraudulent%20transactions>.