**Predicting Credit Card Fraud**

**Milestone: Project Report**

**Group 23**

Vincenzo Coppola

Prachi Sanghvi

(413)-441-6859

(857)-269-4369

coppola.v@northeastern.edu

sanghvi.pr@northeastern.edu

**Percentage of Effort Contributed by Vincenzo Coppola: 50 percent**

**Percentage of Effort Contributed by Prachi Sanghvi: 50 percent**

**Signature: *Vincenzo Coppola***

**Signature: Prachi Sanghvi**

**Submission Date: 05/01/2022**

[List of figures 2](#_Toc102311947)

[List of Tables 3](#_Toc102311948)

[Problem Setting 4](#_Toc102311949)

[Problem Definition 4](#_Toc102311950)

[Data Sources 5](#_Toc102311951)

[Data Description 6](#_Toc102311952)

[Data Mining and Exploration 7](#_Toc102311953)

[Data Preprocessing 7](#_Toc102311954)

[Data Validation 8](#_Toc102311955)

[Data Wrangling 9](#_Toc102311956)

[Dimensionality Reduction 11](#_Toc102311957)

[Overly Unique Attributes 11](#_Toc102311958)

[Redundancy Removal 12](#_Toc102311959)

[PCA 13](#_Toc102311960)

[Data Mining Models / Methods 16](#_Toc102311961)

[KNN 18](#_Toc102311962)

[Logistic Regression 19](#_Toc102311963)

[Decision Tree 20](#_Toc102311964)

[NB 21](#_Toc102311965)

[Model Performance Evaluation and Selection 21](#_Toc102311966)

[Model Optimization 23](#_Toc102311967)

[Final Feature Removal 23](#_Toc102311968)

[Cost-Complexity Analysis 25](#_Toc102311969)

[Performance Evaluation 27](#_Toc102311970)

[Project Results 29](#_Toc102311971)

[Impact of Project Outcomes 30](#_Toc102311972)

[Sources / Acknowledgements 31](#_Toc102311973)

# List of figures

[Figure 1- Dataset with no useful information provided 5](#_Toc102311424)

[Figure 2- SPARKOV Simulated Dataset utilized for project 5](#_Toc102311425)

[Figure 3 Data / Library Import for Project 8](#_Toc102311426)

[Figure 4- Null Verification Using isnull() 8](#_Toc102311427)

[Figure 5- Difference in Class Distribution 9](#_Toc102311428)

[Figure 6- Creation of unix\_diff 10](#_Toc102311429)

[Figure 7- Categorical Data Encoding 10](#_Toc102311430)

[Figure 8- Data Distribution Visualizations 11](#_Toc102311431)

[Figure 9- Unique Ratios per Attribute 12](#_Toc102311432)

[Figure 10- Correlation Redundancy Removal 13](#_Toc102311433)

[Figure 11- Standardization of Features for PCA 14](#_Toc102311434)

[Figure 12- Eigenvalues for every PC and Eigenvectors for PC1 14](#_Toc102311435)

[Figure 13- Plot of PC 1 vs PC 2 Results 15](#_Toc102311436)

[Figure 14- Information Percentage for each PC 15](#_Toc102311437)

[Figure 15- Diagram of Test/Train Split with Sampling 16](#_Toc102311438)

[Figure 16- Code for Test/Train Split with Sampling and verification 17](#_Toc102311439)

[Figure 17- Null Accuracy for Project 17](#_Toc102311440)

[Figure 18- KNN n\_neighbors Selection 19](#_Toc102311441)

[Figure 19- Logistic Regression 20](#_Toc102311442)

[Figure 20- Decision Tree 20](#_Toc102311443)

[Figure 21 - Naive Bayes 21](#_Toc102311444)

[Figure 22- ROC Curve for All Models 22](#_Toc102311445)

[Figure 23- Cross Validation Code and AMT Results 24](#_Toc102311446)

[Figure 24- Validation Model After Feature Reduction 24](#_Toc102311447)

[Figure 25- CCP vs Accuracy analysis 25](#_Toc102311448)

[Figure 26- ccp\_alphas’ Results 26](#_Toc102311449)

[Figure 27- Original Tree (Left) vs Optimized Tree (Right) 27](#_Toc102311450)

[Figure 28- Test Set Preparation 28](#_Toc102311451)

[Figure 29- Final Model Execution and Results 28](#_Toc102311452)

[Figure 30- Final Model ROC Curve 29](#_Toc102311453)

# List of Tables

[Table 1- Dataset Description 5](#_Toc102309928)

[Table 2- Model Results 23](#_Toc102309929)

[Table 3- Final Model Results with cpp\_alpha 26](#_Toc102309930)

# Problem Setting

Over [80% of Americans](https://www.bankrate.com/finance/credit-cards/credit-card-ownership-usage-statistics/#:~:text=Key%20credit%20card%20statistics,most%20recent%20data%20from%20Experian.) have at least one credit card and on average that number exceeds 3.84 cards per person. With the advancement of modern technology and now an unprecedented global pandemic, credit card fraud has been one of the fastest growing types of fraud and is thought to be one of the most difficult to prevent. Credit card fraud has the ability to affect each and every one of us, but we can do things to mitigate these issues such as securely store our cards and shop in person. Financial institutions however take responsibility for the exposure of records in order to mitigate person by person concerns. In the last decade big name companies such as Facebook, Yahoo, and Capital One have been a part of some of the biggest exposures including upwards of 4 billion exposed records. These types of outbreaks can cost financial institutions billions of dollars and it is not in their hands whether their customers do practice safe shopping. It is an ever-growing issue to prevent fraud and loss, but it is one where the greatest potential of mitigation comes from Data Mining.

# Problem Definition

After all is said and done, the challenge being addressed is to automatically discern fraudulent credit card transactions. Data Mining can help do this through a type of supervised machine learning called classification. This can be done through a variety of different classification models, but any good model put in place should be able to accurately and efficiently detect fraud. Therefore, to address the concerns that come with fraudulent purchases using accurate models, the goals are defined as follows:

1. Accurately classify fraud from existing data
2. Discover which features are important to do so and minimize the number needed
3. Implement several models to determine the optimal model
4. Optimize that model and accurately predict fraud on a fresh set of data

# Data Sources

A problem immediately arose in the project when searching for data to address the problem. The available data for credit card transactions has its drawbacks as much of the financial information involved can be considered confidential. There is a trust between customers and their card suppliers to keep their personal information stored in their database system safely and securely. For this reason, access to this type of data is hard to acquire and any public datasets that could be found were run through some type of confidentiality cloak such as PCA. This strips the actual information and creates them into components instead of features. Although it would still be possible to perform prediction / classification accurately, this wasn’t the ideal path we wanted to take the project. We wanted to perform the classification and see exactly what information was required to do so. After days of searching, we found a dataset of simulated credit card transactions which was developed for the SPARKOV project. Over a year of python, shell and batch code development went into this dataset simulation and it helps to solve the issue of confidential credit data. Below in Figure 1 the problematic data can be seen showing the lack of information available for the project and Figure 2 shows the utilized dataset.

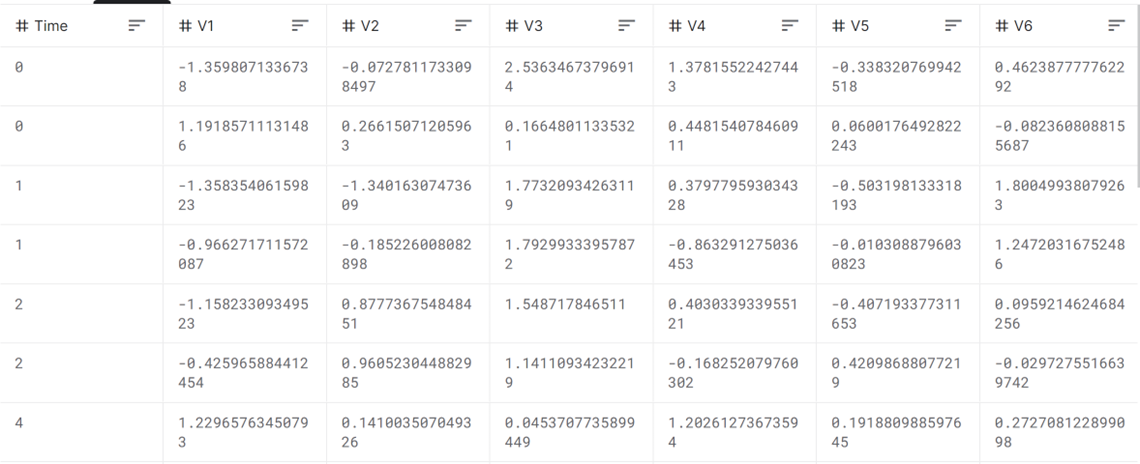


Figure 1- Dataset with no useful information provided

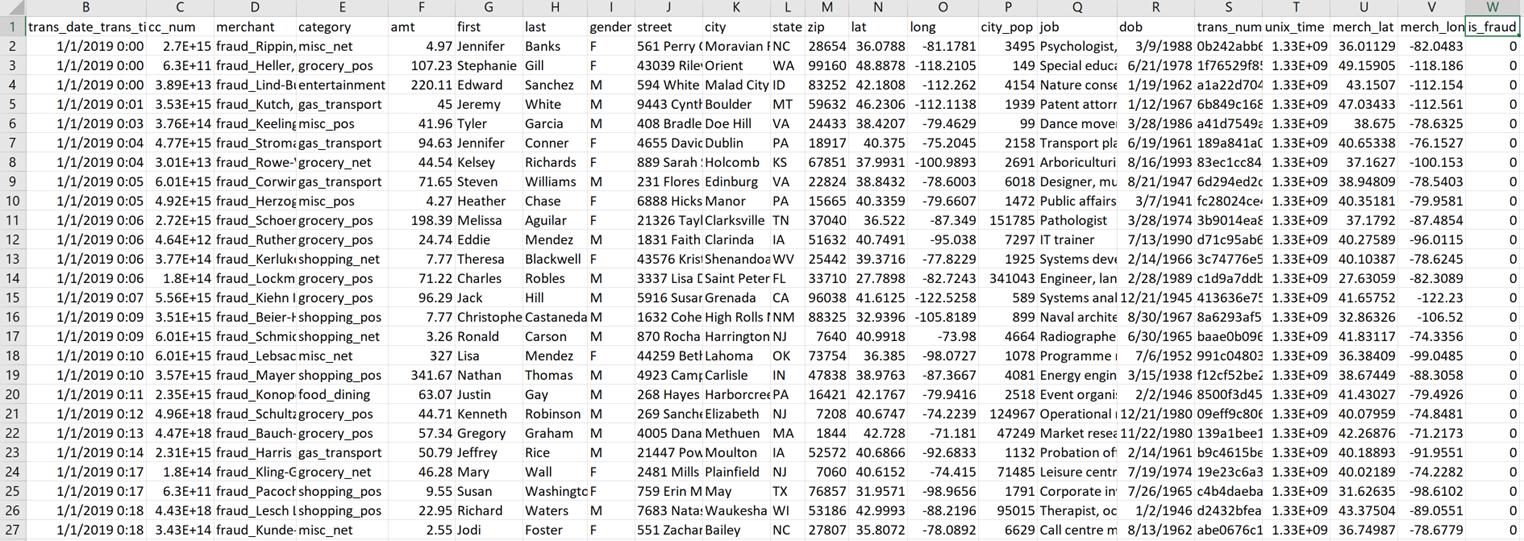


Figure 2- SPARKOV Simulated Dataset utilized for project

The datasets can be found at the following sites:

Unused Dataset

* <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>​
* <https://data.world/raghu543/credit-card-fraud-data>

Utilized Dataset

* <https://www.kaggle.com/datasets/kartik2112/fraud-detection>​
* <https://github.com/namebrandon/Sparkov_Data_Generation>​

# Data Description

The dataset selected contains information about the transactions carried out so some of the details include Transaction (date | time | number), amount, credit card number, name (first | last), address (street | city | state | latitude | longitude | zip), date of birth, job title, merchant (name | longitude | latitude) and other features totaling to 22 attributes. The records are each classified in the 23rd column as is\_fraud, where a real purchase is denoted by a 0 and fraud is denoted by a 1. The SPARKOV simulation was separated into two datasets, one titled FraudTrain.csv and the other FraudTest.csv. For all data mining including initial processing, visualizing and modeling, FraudTrain was used and FraudTrain was only used at the final stage for performance evaluation. The two sets are defined below in Table 1 as follows:

Table 1- Dataset Description

|  |  |  |
| --- | --- | --- |
| Dataset Name | **Number of rows** | **Number of columns** |
| FraudTrain.csv | 1048574 | 23 |
| FraudTest.csv | 555719 | 23 |

# 

# Data Mining and Exploration

Data Mining is the heart of what makes this project happen and data exploration is just as much of a crucial step in that process. Before beginning any real data manipulation in the data mining process, one must be familiar with the data’s attributes in order to prepare the best mining approach catered to the data. It also allows for a better understanding of what makes up the data, making it easier to explore, uncover trends and transition into prediction. With well explored and understood data, things like prediction become much easier and more efficient. In this project as in many projects, the exploration and mining processes were not mutually exclusive or linear. Rather, the processes were connected and very cyclical in nature as we found oftentimes data would be explored and then modeled revealing imperfections in the original process. As the processes were completed and repeated several times, the data exploration and Mining in this project included several steps which we group into what we consider is Data Preprocessing.

## Data Preprocessing

Data Preprocessing is required to understand and best prepare the data for prediction. We can preprocess data to reduce data size and remove dirty, incomplete and inconsistent data from the dataset as well as visualize the good data to better understand trends. Data Preprocessing mainly involves the following steps

* Data Validation
* Data Wrangling
* Data Visualization
* Dimensionality Reduction

In order to complete this we utilized mainly Jupyter Notebook as well as Google Colab and started with importing the data as well as python libraries including NumPy, pandas, matplotlib and sklearn to name a few. Figure 3 below shows the data/library import.



Figure 3 Data / Library Import for Project

### Data Validation

In Data Validation we first checked for missing values in the dataset. This was done in two ways: one using isnull() search from pandas and one by checking the type of each attribute seeing which were non-Null. Since the database we used was simulated, it allowed for us to have a clean dataset from import and there were 0 Nulls found. Below in Figure 4, the first check can be seen as verification.





Figure 4- Null Verification Using isnull()

The next step of data validation was to determine the distribution of classes as we were not sure of the data’s Fraudulent to real transaction count. This was done through finding the sum of each class and dividing by the total number of records as a ration. The code can be seen below in Figure 5, as well as the corresponding bar graph showing the significant difference in results (99.43% non-fraud to 0.57% Fraudulent).





Figure 5- Difference in Class Distribution

### Data Wrangling

Now rest assured that the data is clean and having a better idea of what it is composed of, we set out to discover key qualities from the dataset that would ideally aid us in determining the relations between the columns in this process. In order to do this, we wanted to visualize the distribution of the data for each class and see where large discrepancies lie. Before doing this, we had a hypothesis based on our personal credit card knowledge. One of the columns in our dataset is trans\_date\_trans\_time which gives the date and time of each transaction. Knowing this information, we believed that fraudulent transactions potentially happen back-to-back to back for the same card number. If one card was used twice in 3 seconds for example, we believed that may be a red flag as that is unrealistically fast. Therefore, we wanted to create a custom column which we called unix\_diff. It was made by splitting the date/time column into two separate columns named trans\_date and trans\_time. We then have grouped the data according to the user CC number and found the difference in time between each transaction of the users imputing any NA’s (first transaction for every card) with 0. The code for this column can be seen below in Figure 6.



Figure 6- Creation of unix\_diff

With all columns needed we realized that many of the attributes were text and categorical which we later found would cause complexities in things like dimensionality reduction and model implementation. Therefore, we decided to encode the categorical data which is a method of representing categories as numeric. Using sklearn’s labelencoder() we encoded the data as shown in Figure 7 below.

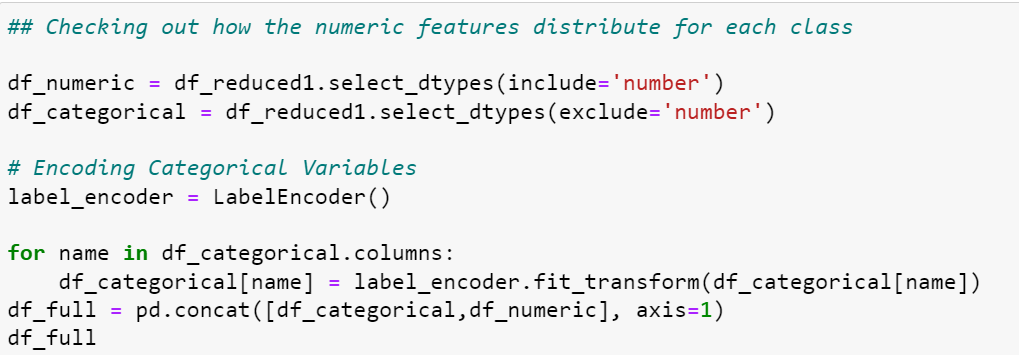
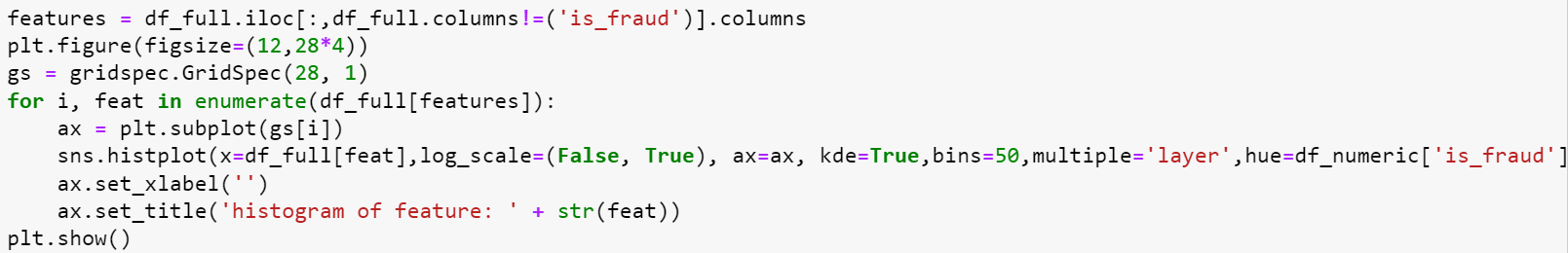
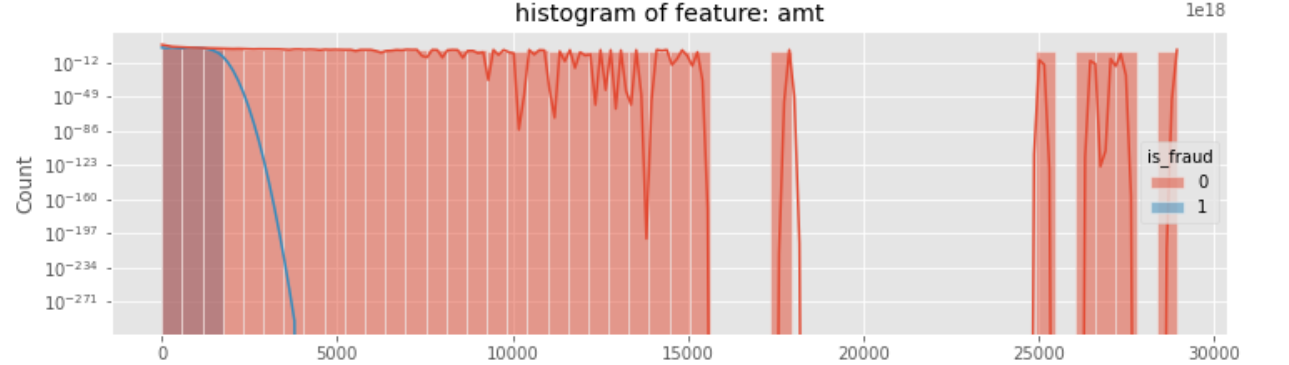


Figure 7- Categorical Data Encoding

With the data prepared, we then used histograms to visualize the relationship between attributes and the target column (is\_fraud). The results showed some very encouraging distributions including categories such as amt (transaction amount) and unix\_diff (previously mentioned custom column). Those two are shown below as examples along with the code in Figure 8.





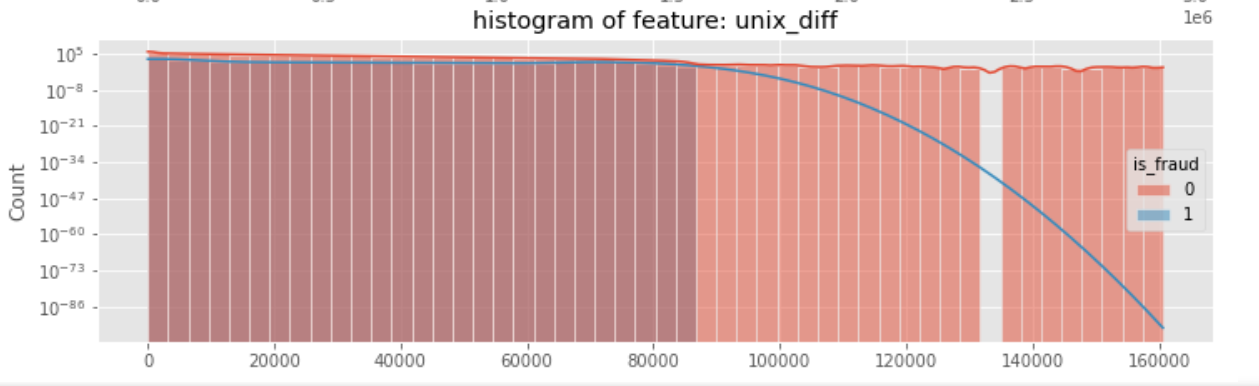


Figure 8- Data Distribution Visualizations

### Dimensionality Reduction

Dimensionality reduction helps us in removing multicollinearity between the variables. It enables us to remove redundant information and other types of un-useful attributes, as well as potentially combine column information based on variance. In this project, we implemented several ways of reducing dimensions including first removing attributes with overly unique-dense ratios, removing one of two overly correlated attributes and performing Principal Component Analysis (PCA).

#### Overly Unique Attributes

Prior to even implementing some of the previous wrangling such as the encoding and visualizations, we wanted to remove some attributes which we knew for certain would not be useful. Therefore, we set out to remove overly unique attributes. To put it in perspective, if there are 100 columns and 90 of those columns have a different value then the column is 90% unique and there wouldn’t be any trends to predict upon. In this project we decided to iterate through each column and find the ratio of unique values, removing any above 75% unique. The code and results are below in Figure 9 showing that the index column, transaction number / date & time and others are overly unique. This makes sense as these attributes are things that pertain to only one or a couple records and therefore will greatly differ from record to record.

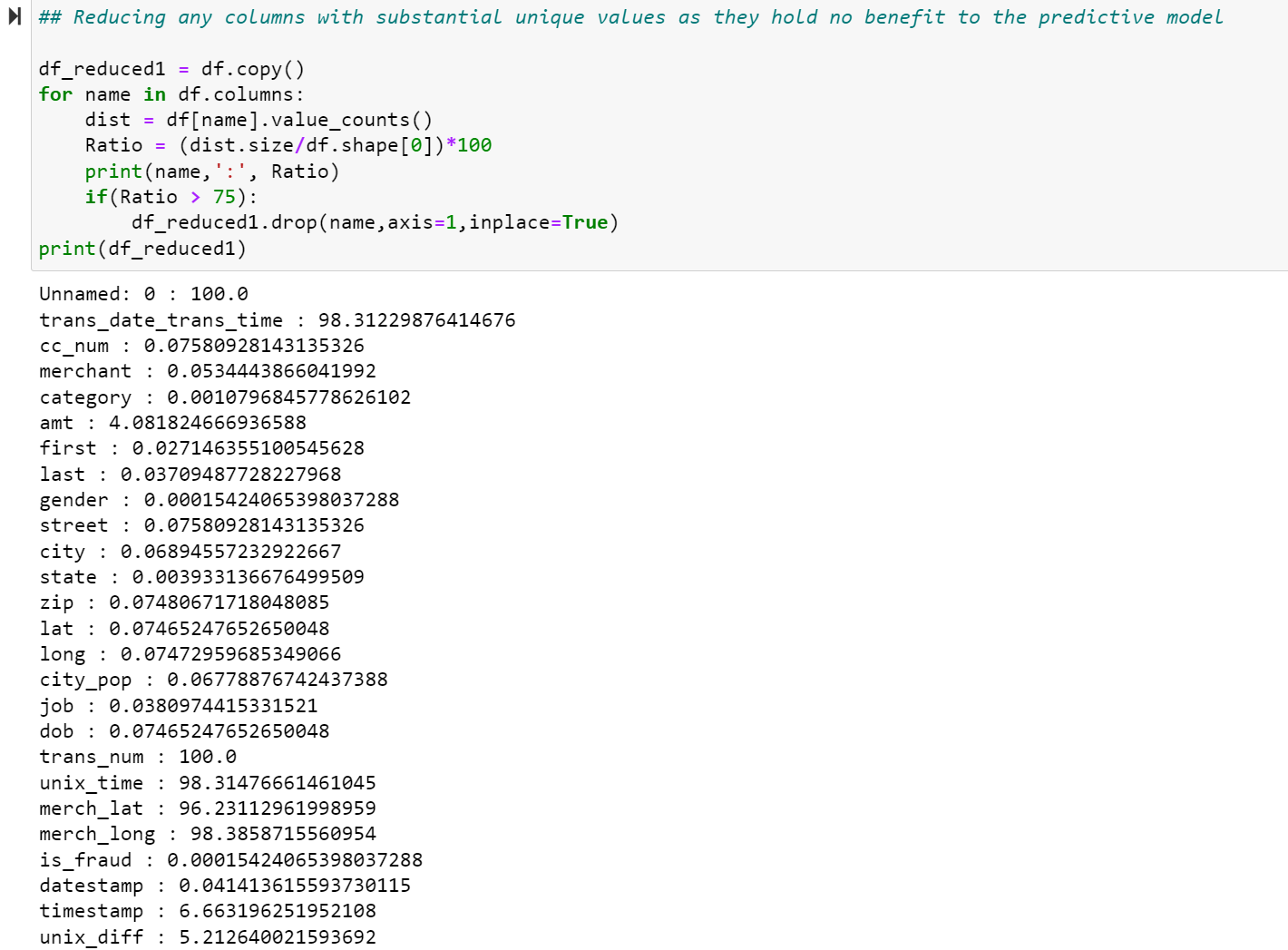
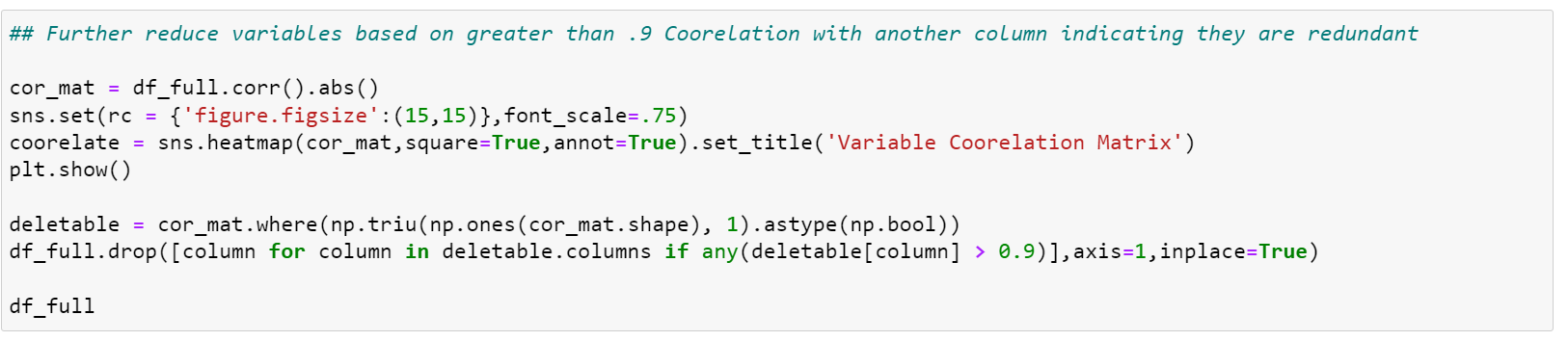


Figure 9- Unique Ratios per Attribute

#### Redundancy Removal

Another method of reducing the attribute quantity is to see which columns may be highly correlated. This dictates that the columns may be redundant and therefore the prediction would only require one of them. We found the correlations between columns and removed one of two columns where the correlation was found to be above 0.90 determined to be the redundant level. The code and results can be seen below in Figure 10 showing that long (longitude) and zip (zip code) had 0.91 correlation and therefore long was removed.



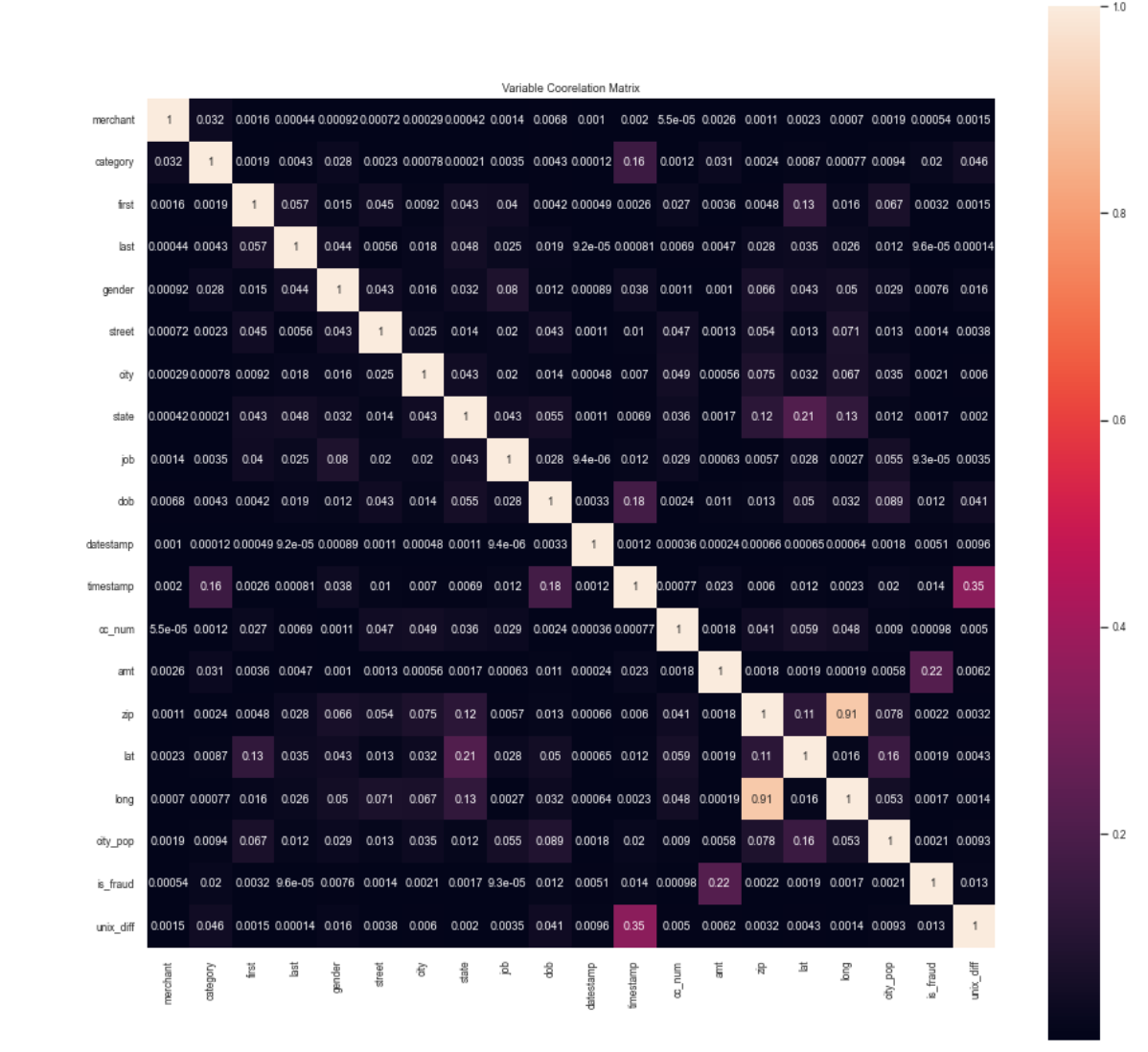


Figure 10- Correlation Redundancy Removal

#### PCA

PCA is a dimensionality reduction method that reduces the attribute count of large data sets by turning a large number of variables into a smaller number called principal components (PC) which still contain the majority of the data information. PCA takes the original features and reduces them into components which hold orthogonality in their eigenvectors. The Steps taken to complete PCA are described below:

1. Standardize the dataset to ensure that everything is on the same scale and larger domain attributes do not take over smaller values. This is shown in Figure 11 below.

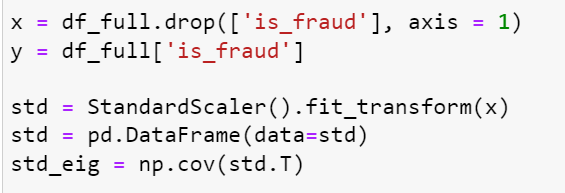


Figure 11- Standardization of Features for PCA

1. Calculate the eigenvalues and eigenvectors which help reveal orthogonality. This is shown in Figure 12 below with an example of eigenvectors (PC1).

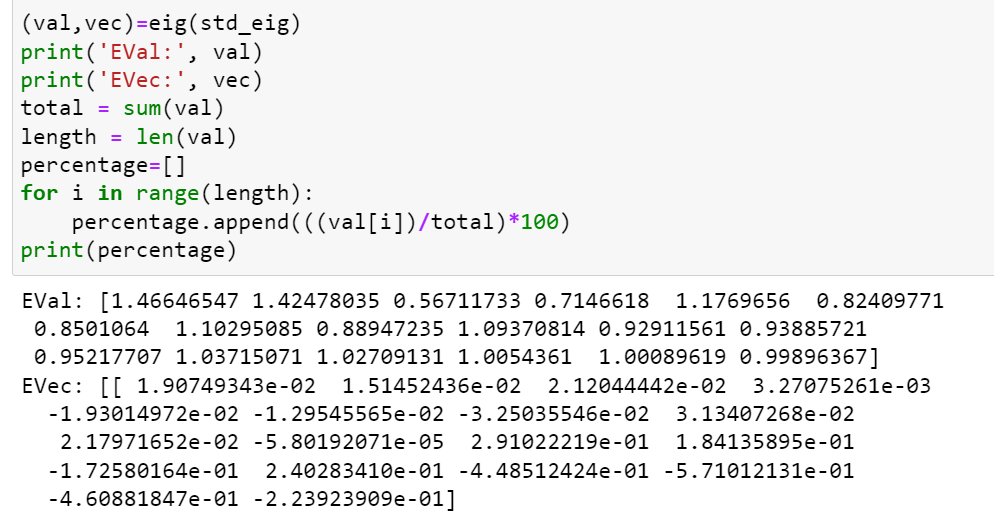
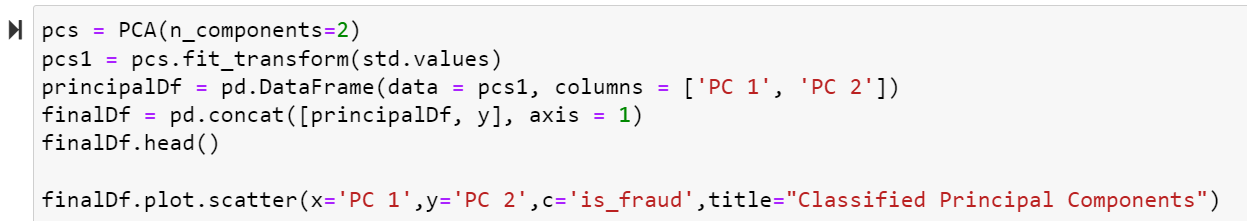


Figure 12- Eigenvalues for every PC and Eigenvectors for PC1

1. Visualize top two Principal Components graphically. This is shown in Figure 13 below.



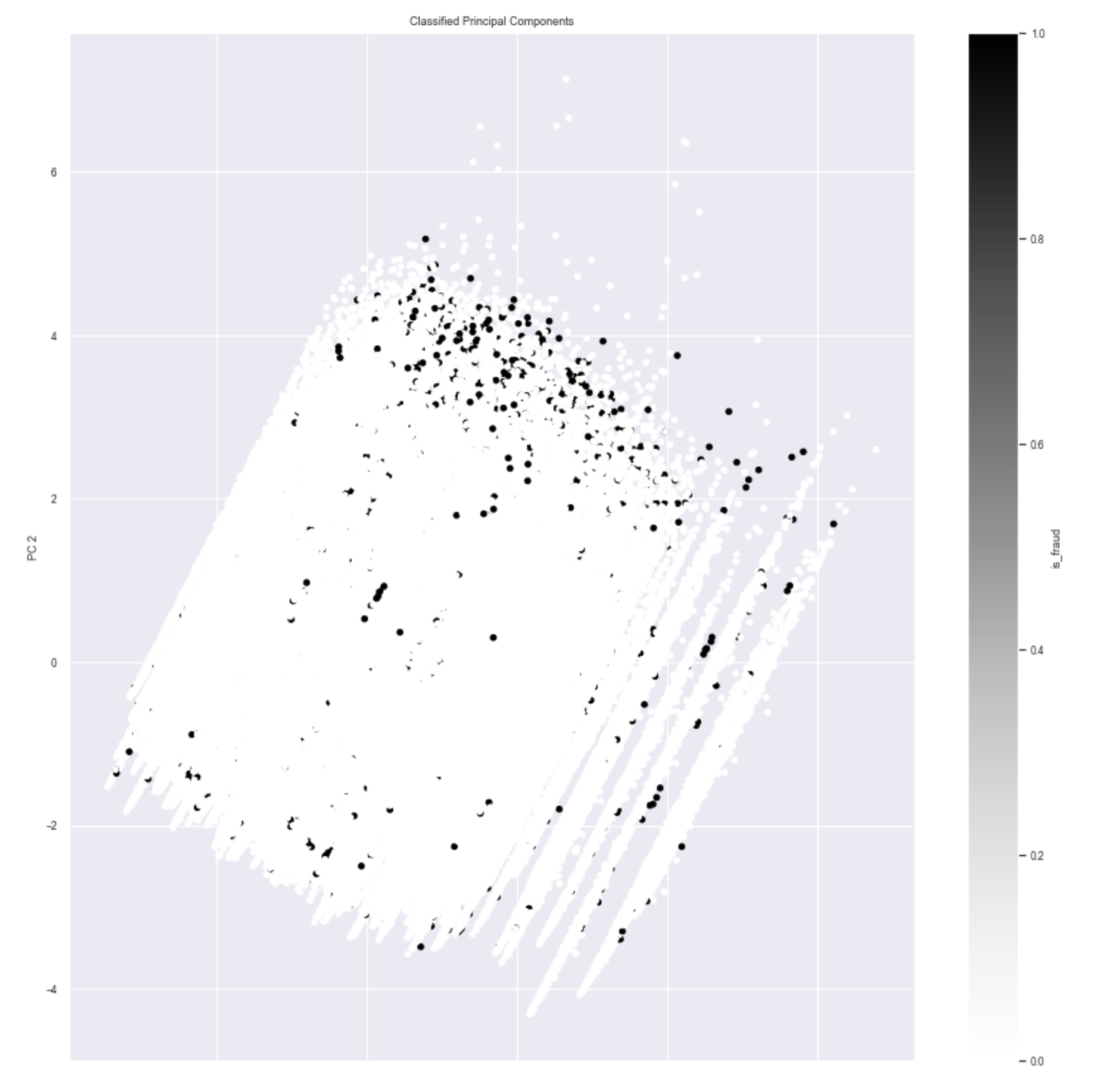


Figure 13- Plot of PC 1 vs PC 2 Results

As can be seen, PCA was not useful for this dataset and project. The two first Principal components show almost no differences in class. Digging further into why this is the case, we can look at the information retained which is derived from the eigenvectors. As can be seen in Figure 14 below, every component holds less than 10% of the information including PC 1 and 2 which hold 8.15% and 7.92% respectively which combine for only ~16% of the data information. This process shows that PCA provides almost no assistance for this project as nearly all of the features are already orthogonal in nature. Still PCA is a very useful method of reducing dimensions and it was something that was very useful to try and get results for.

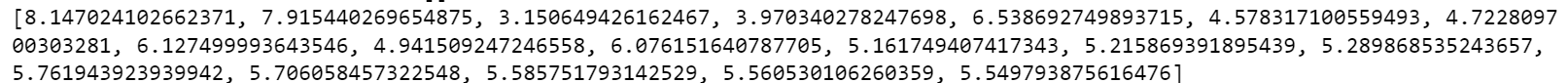


Figure 14- Information Percentage for each PC

# Data Mining Models / Methods

With all exploration and wrangling complete, the project could finally advance to the final stages and address its actual purpose, classifying fraud. Before any classification can be done, a dataset must be split into a training set which is typically 80% of the records (used for fitting/training the model by allowing the model to be exposed to records and matched classes) as well as a validation set with the other 20% of records (used for validating the model and assessing accuracy metrics). The problem we immediately ran into was that our data as mentioned several times had 99.43% non-fraud and only 0.53 percent fraudulent records and therefore was extremely biased. We ran the models with the 80/20 TestTrainSplit but failed to get good validation sensitivity due to the training on biased data. Therefore, we found that we needed to sample the data and train on a balanced dataset. In short, as we had 7506 fraud cases, we took half of them (3753) and put them into the training set with the same number of fraud cases. We then took the other 3753 and put them in the validation set with the number X = 644585 that would retain the original data ratio of 99.43% real. The diagram in Figure 15 and the code in Figure 16 show the split design and execution respectively.

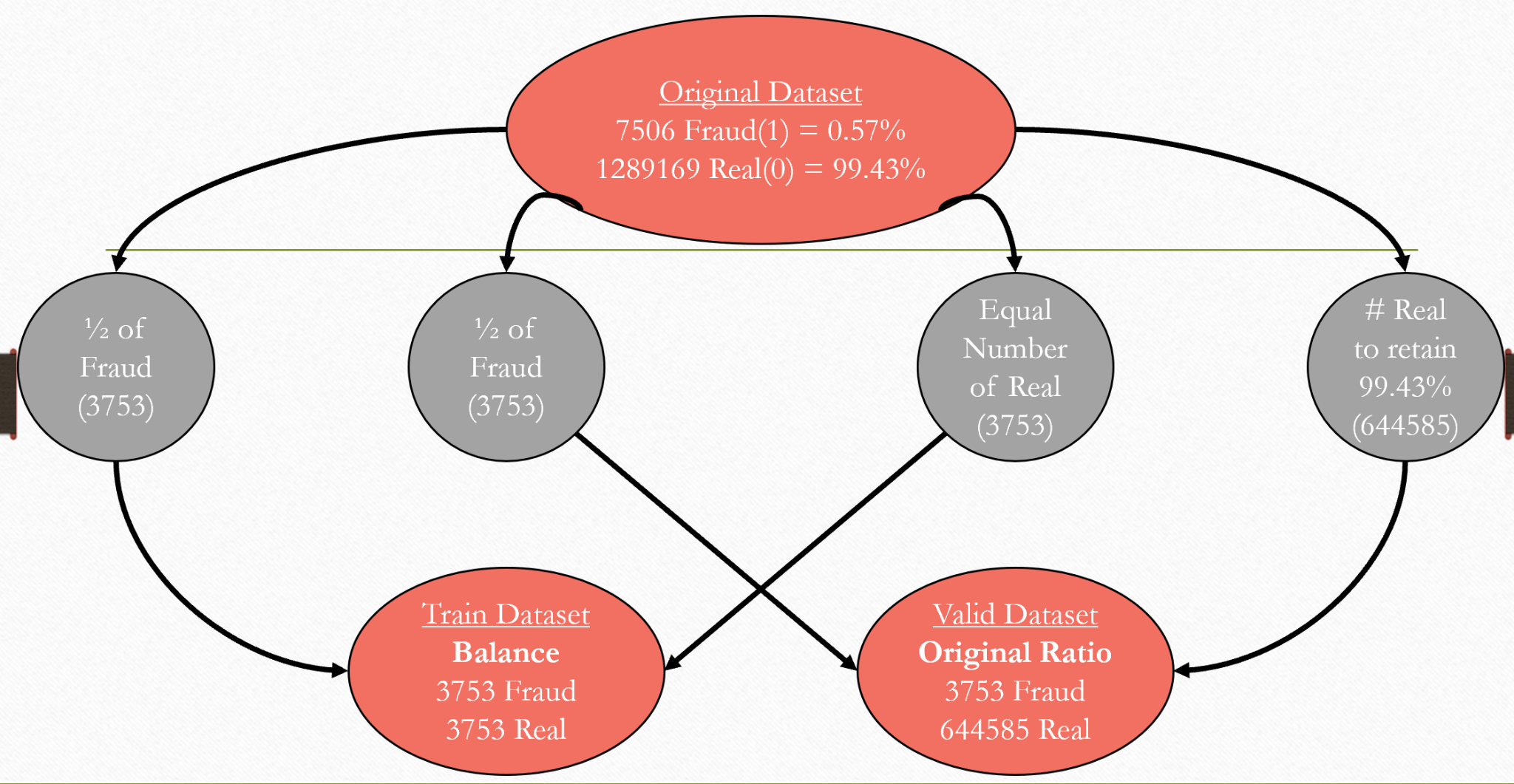


Figure 15- Diagram of Test/Train Split with Sampling

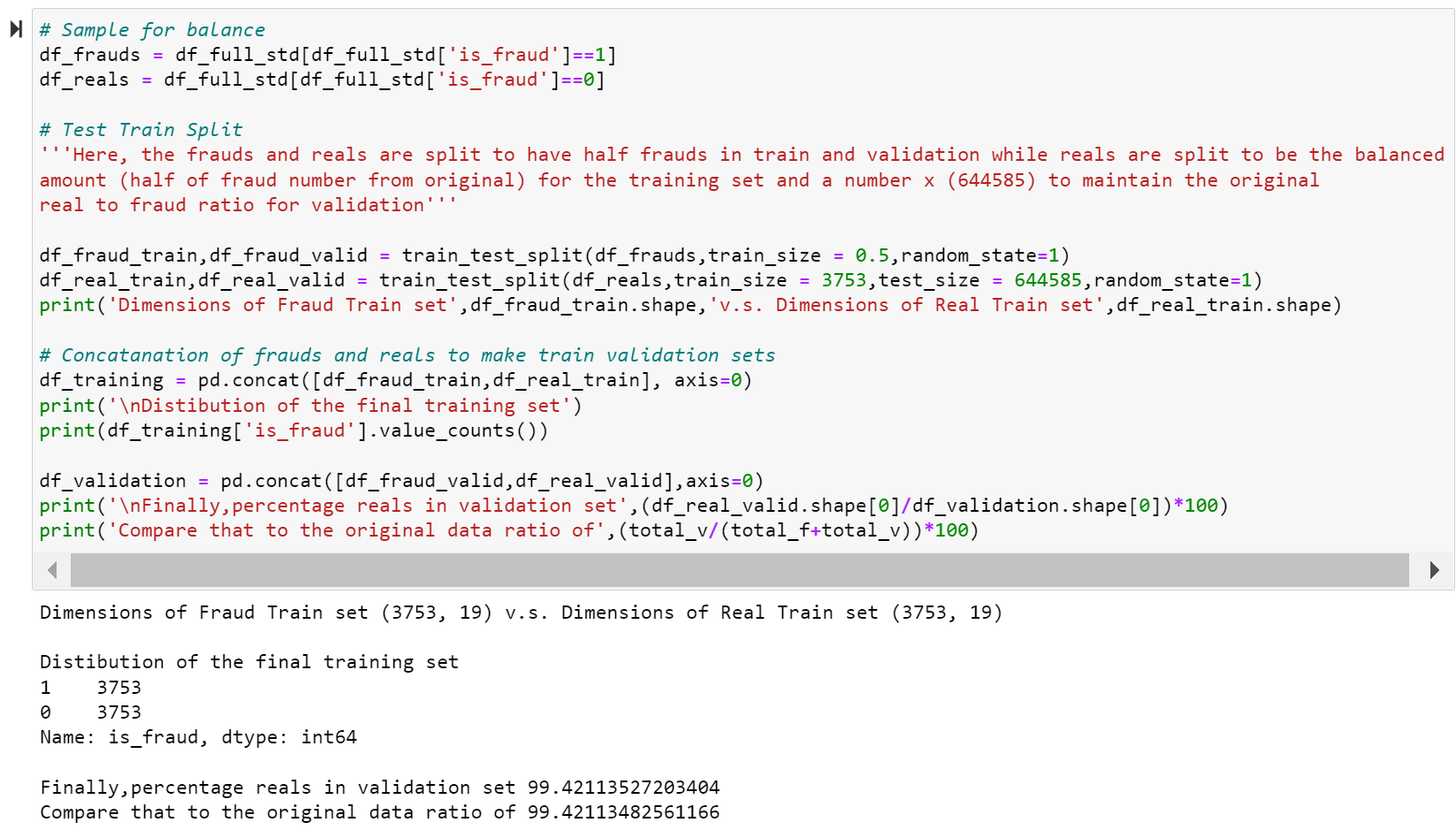


Figure 16- Code for Test/Train Split with Sampling and verification

With this completed, we were ready to address the aim for predicting Credit Card Fraud which is to find the optimal model to use. However, there are many metrics which can be looked at when talking about optimality including Accuracy, Root Mean Squared Error, Specificity, F1-Score and many more. When approaching how to best measure the model's effectiveness, we first wanted to get a sense of what the model had as a benchmark accuracy. To do so we found something called the Null Accuracy which is the accuracy under the assumption that all records are predicted to one superior class and in our project’s case that was the non-fraud class. As it has 99.43% non-fraudulent records, under this assumption the accuracy would be an astonishing 99.43% if all records were predicted 0. This fact shown coded in Figure 17 below, illustrates why accuracy is not a useful metric in this project because although the accuracy is extremely high, the “Null” model would catch 0 Fraud and prove useless.

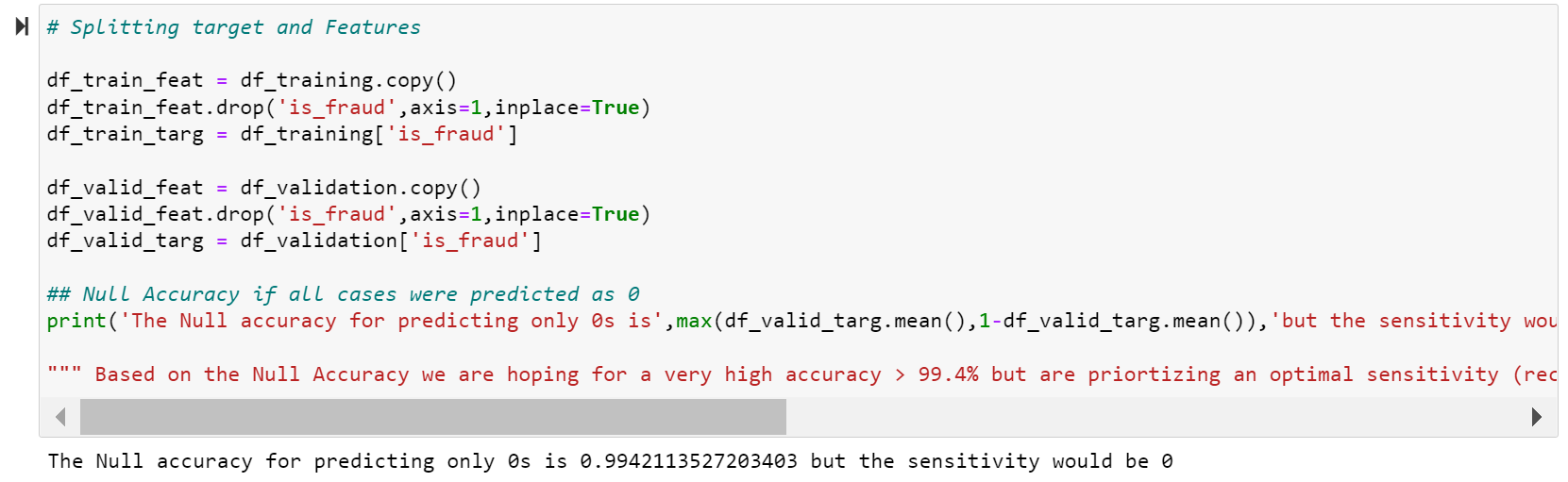


Figure 17- Null Accuracy for Project

Therefore, through further analysis and thought, we declared that the important feature to be prioritized for our models is Sensitivity, which is the rate at which the model catches fraudulent transaction cases (1s). In the finance industry, it is much better to call real transactions fake and have the buyer call to confirm their purchase, than to miss a fraudulent transaction and allow someone to essentially get away with theft thus costing an organization potentially billions of dollars. With all of this understood we were ready to test models and from research and best practices we thought to try the following four methods as they are well established classification models in Machine Learning:

1. Logistic Regression
2. K Nearest Neighbors (KNN)
3. Naive Bayes (NB)
4. Decision Tree

When applying the models, each model was trained with the train set and validated with the validation set. Many metrics were tested but in the end the confusion matrix and classification report were printed for each model and used for assessment. The implementation of each model went as follows:

## KNN

First, KNN was modeled. KNN has a drawback which involves the process of needing to select the n\_neighbors value. In order to do this the confusion matrix and classification report was calculated for n\_neighbors = 1-10 and it was determined that results were optimal when n\_neighbors was equal to 1. Figure 18 below shows the KNN n\_neighbors selection. Another drawback that was revealed was time inefficiency in KNN as this process of n\_neighbors selection took over an hour.

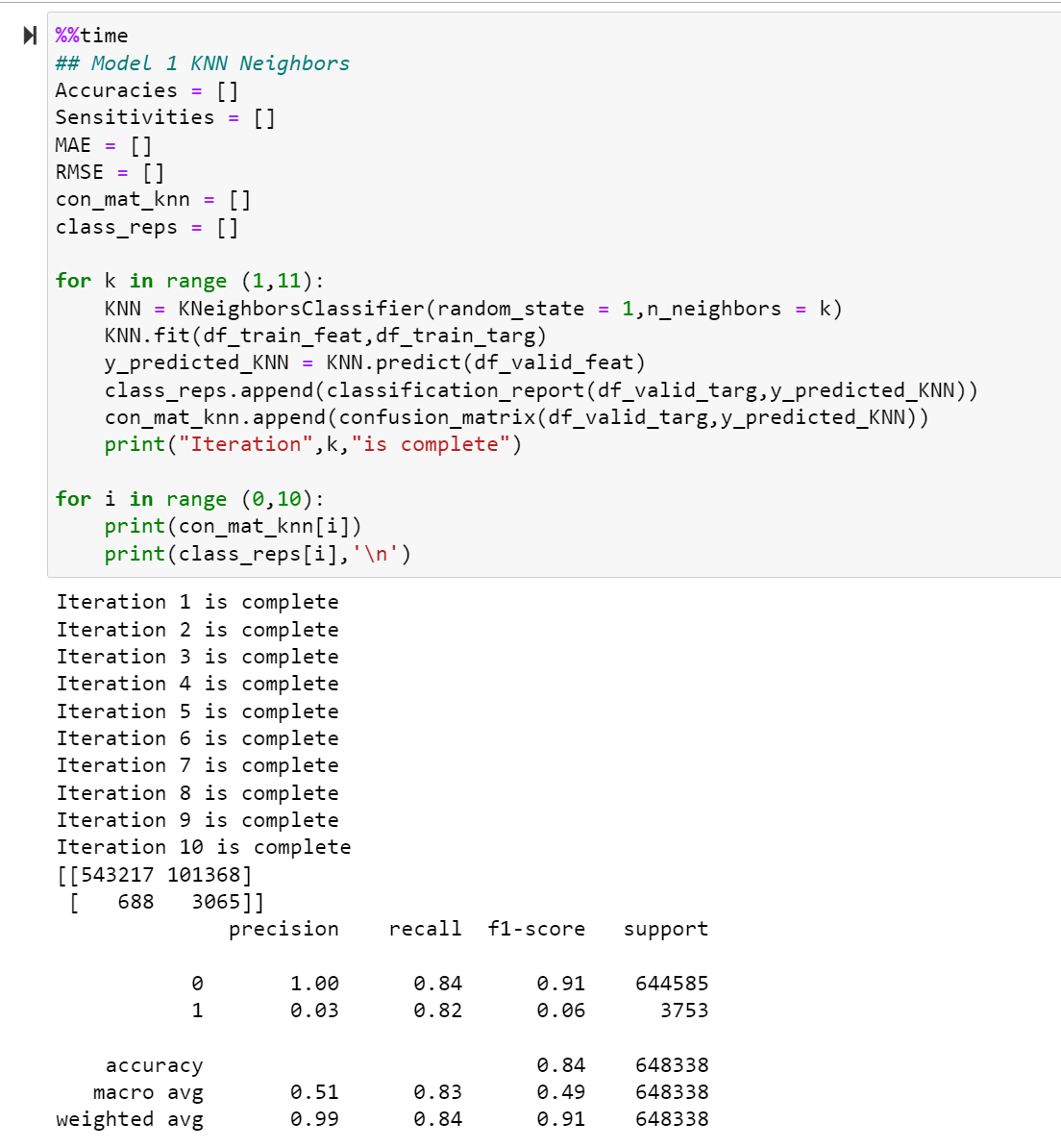


Figure 18- KNN n\_neighbors Selection

Following this, the model was finalized using n\_neighbor=1 and the results were as seen above in Figure showing 84% Accuracy and 82% Sensitivity which is better fraud detection than the Null accuracy. Although some of the other values of n\_neighbors such as 2 showed accuracy numbers in the 90s, sensitivity was sacrificed which is the reasoning behind choosing 1. Overall KNN was successful in improving sensitivity from the Null hypothesis however each iteration took on average over 3 minutes and didn't have as good results as some other models.

## Logistic Regression

After KNN, Logistic Regression was implemented. This model worked well in improving sensitivity as well but not as good as KNN. In what it lacks in sensitivity results it makes up for in stronger time efficiency and improved accuracy. The results for logistic regression were encouraging however not much better if at all than KNN and this is thought to be due to the linear limitations of the regression. The results can be seen below in Figure 19.

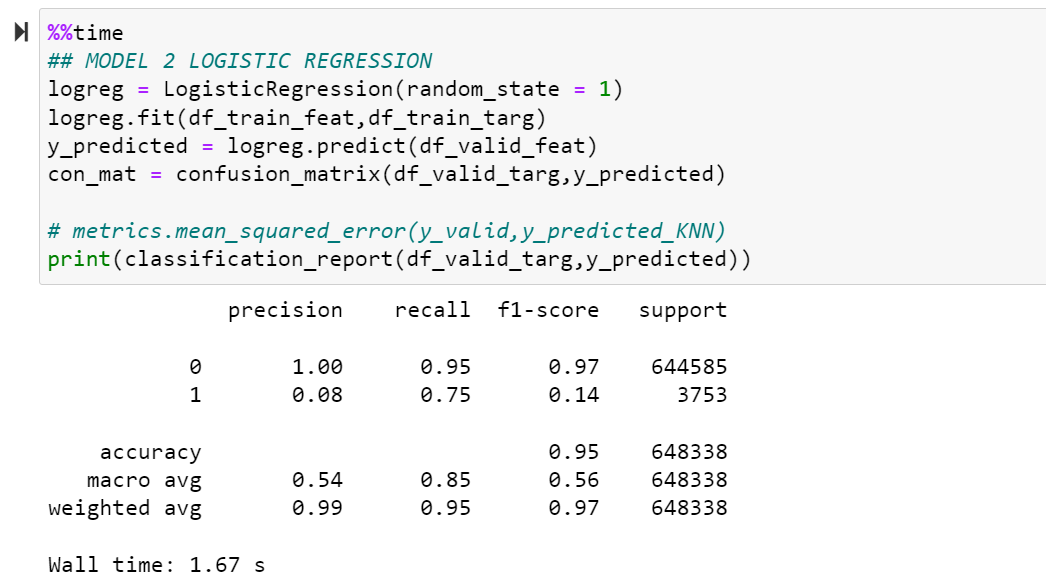


Figure 19- Logistic Regression

## Decision Tree

After KNN and Logistic Regression, the next model understood and therefore implemented was Decision Tree Classifier. The classifier was run in similar timing to the logistic regression but had exponentially better results as will be shown in Figure 20 below. The results showed 94% and 95% Sensitivity and Accuracy respectively. This was the first sign of 90+% Sensitivity for the project was the most encouraging to see in the model implementation stage. The only drawback for the Decision Tree was the overwhelming sense that the tree overfitted which is highly possible for Decision Trees.



Figure 20- Decision Tree

## NB

Naive Bayes was the final model to be implemented. The results for this model were discouraging in comparison to Decision Tree as it achieved barely adequate results similar to the Logistic Regression as aforementioned. The results can be seen below in Figure 21.

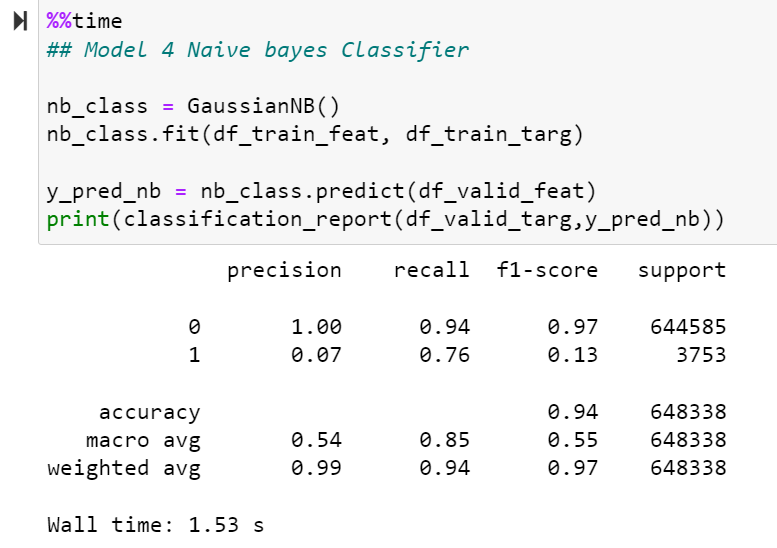
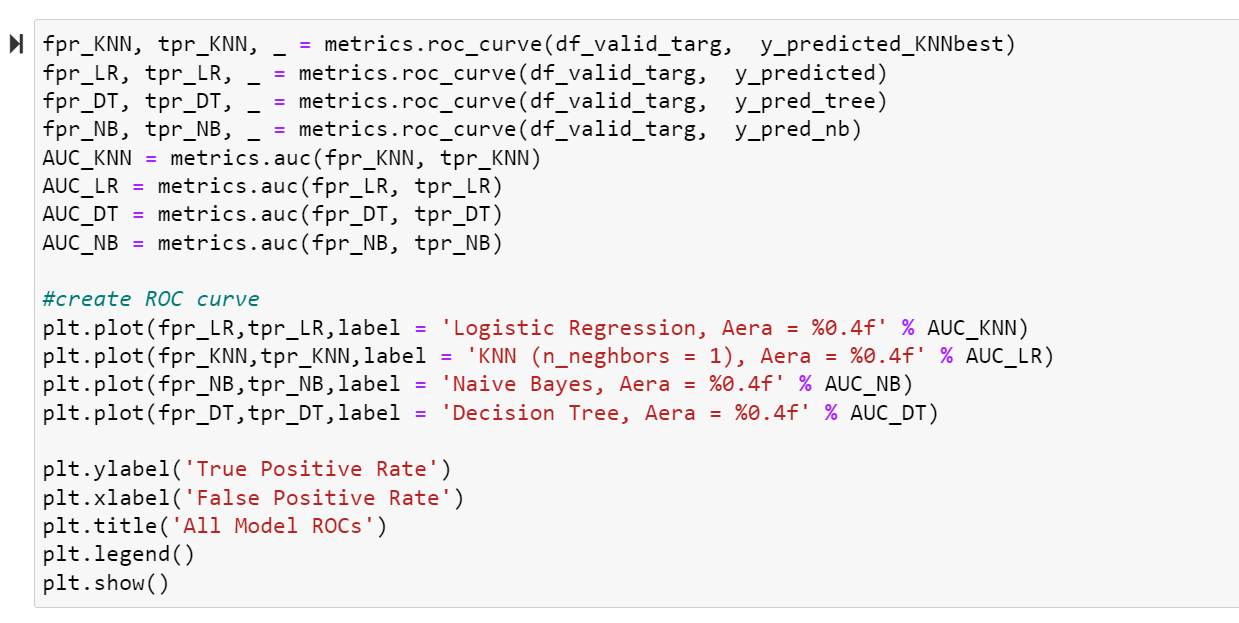


Figure 21 - Naive Bayes

## Model Performance Evaluation and Selection

Although Sensitivity was a very important attribute, we knew that it would be hard to base a selection of one or two factors when others might be sacrificed and therefore, we decided to validate the decision with something all-inclusive. Since a prediction’s effectiveness can vary based on different discrimination cutoffs selected, we wanted to see the Receiver operating characteristic (ROC) curve for each model to find the best Area Under the Curve (AUC) and get an overall picture. This was done through the following code and output in Figure 22 below.



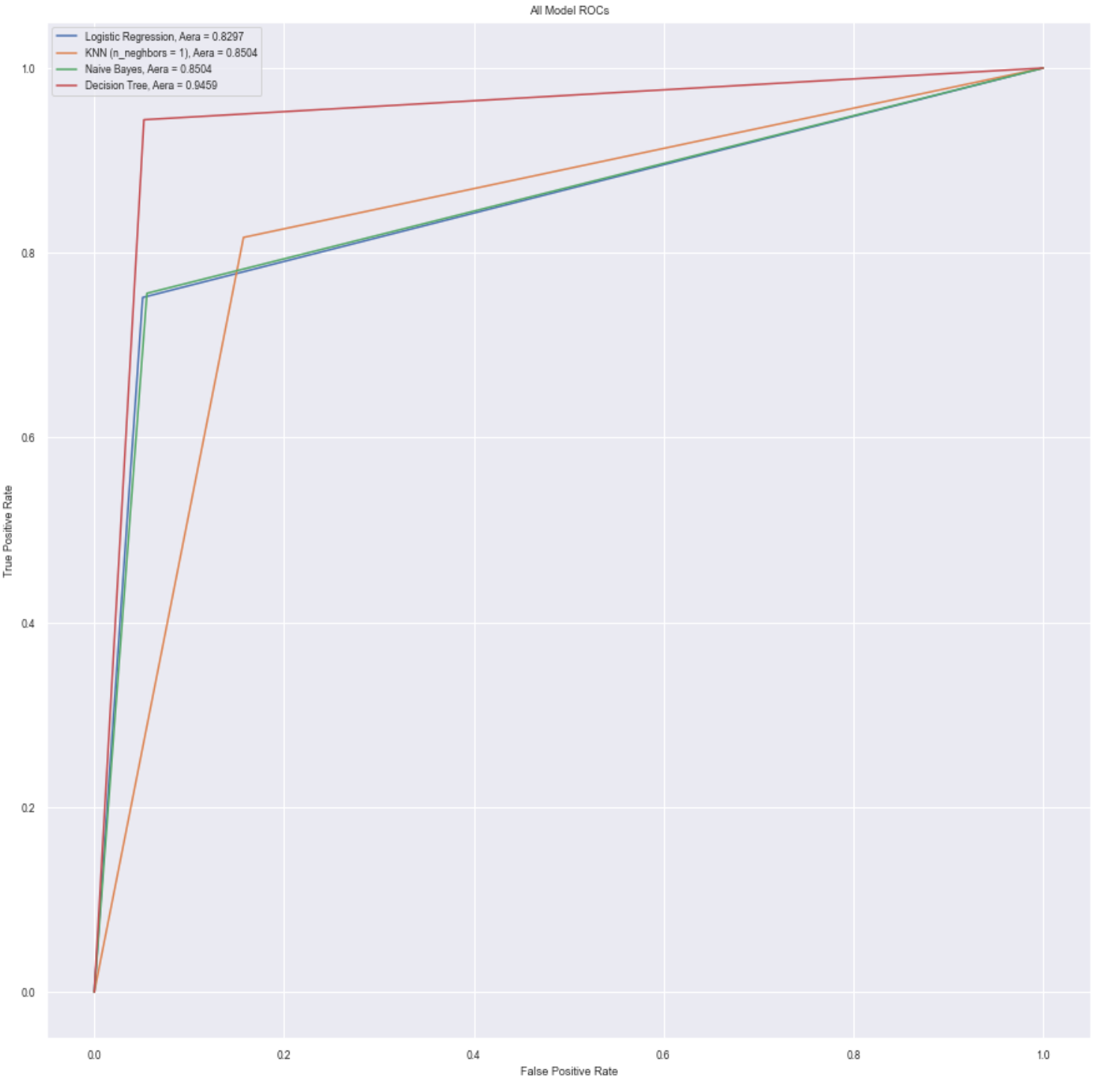


Figure 22- ROC Curve for All Models

Clearly, no curve is perfect but as can be seen from the results above, the decision tree is far and away the best overall curve with 0.9459 AUC as a baseline. Overall, the decision was made to move forward with this model as the selected model, but all model results would likely be considered good results as they can be seen below in Table 2.

Table 2- Model Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Sensitivity | AUC | Avg. Time | Selection |
| KNN | 84 | 82 | 0.8454 | ~ 3 min |  |
| Logistic Regression | 95 | 75 | 0.8297 | ~ 1.75 s |  |
| Decision Tree | 95 | 94 | 0.9459 | ~ 1.5 s |  |
| NB | 94 | 76 | 0.8454 | ~ 1.5 s |  |

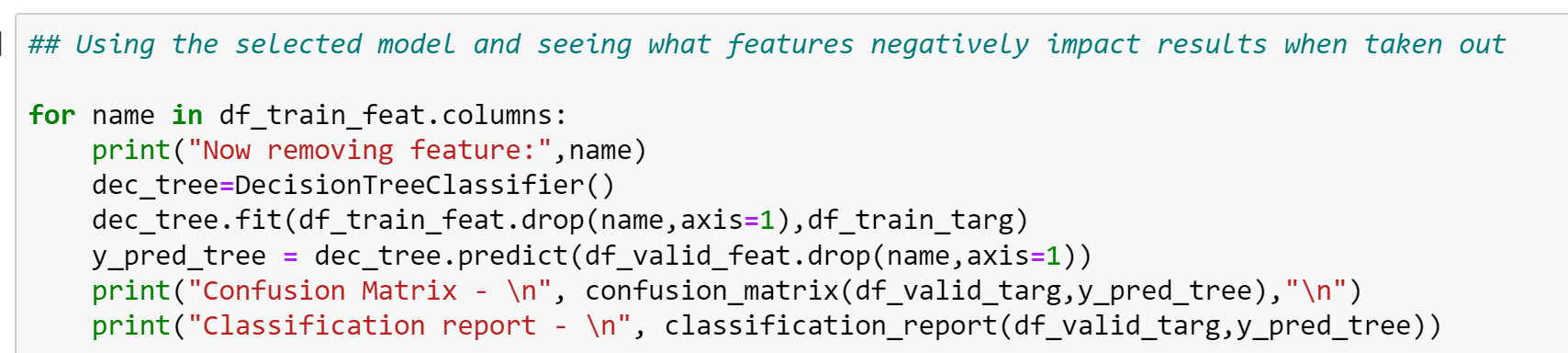
## Model Optimization

Based on the model testing, described above, the Decision Tree Classifier offered the best accuracy and sensitivity at face value. The real test was going to be finding how to maximize these metrics without sacrificing efficiency or other metrics in the process. In order to do a few approaches were taken including the following:

* Final Feature Removal
* Cost Complexity Analysis

### Final Feature Removal

First, as was mentioned in the goals of the project, we wanted to minimize the number of features required to predict because a successful model for us would take basic information such as amount and time and make a classification on that, rather than on hundreds of pieces of information. While it might show that the devised model would be capable of potentially even better predictions with additional information this really was done as it would show maximum efficiency. Therefore, we iterated through the features and removed them one at a time observing the model metrics. Then, any features which had no negative drawbacks on the results were considered redundant for the model and removed. Any features which significantly changed the results and hindered the metrics were considered crucial and kept. In Figure 23 below, the code can be seen along with an example of the removal of the transaction amount, a crucial feature.



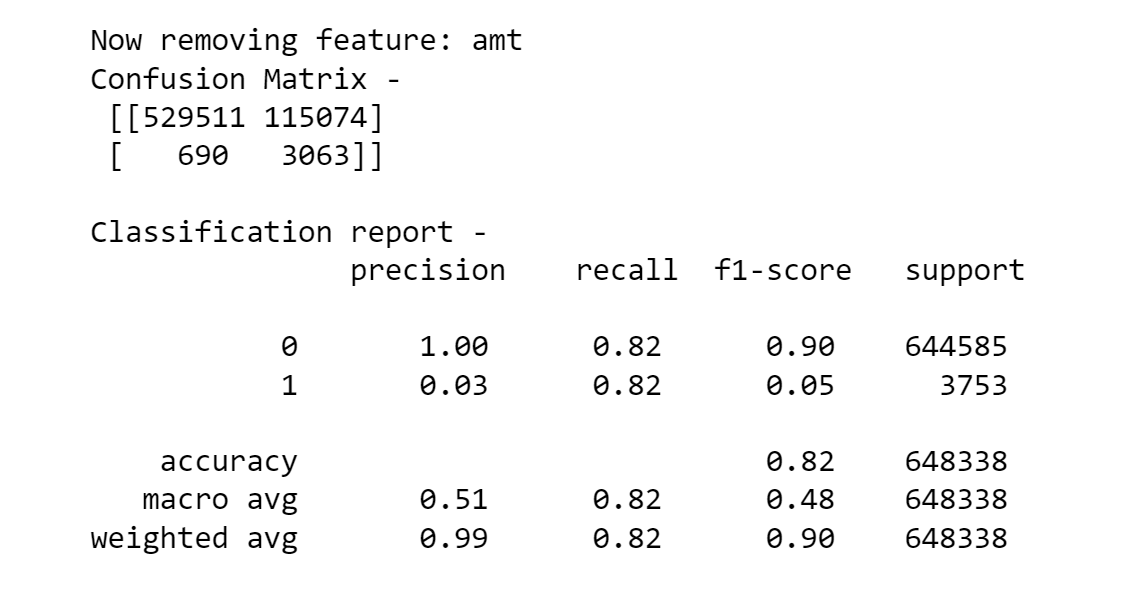


Figure 23- Cross Validation Code and AMT Results

The model was found to be able to make great predictions as seen before with as few as 5 features: category, street, dob, amt, city\_pop but the feature which was added in pre-processing (unix\_diff) did have small impacts during reduction and so was kept as well to observe as it is an important feature tied to time. In the end, leaving one time-based feature whether it was timestamp or unix\_diff (time between purchases) improved the accuracy (timestamp) or sensitivity (unix\_diff). As sensitivity is priority the second choice was made, and the results are as shown below in Figure 24.

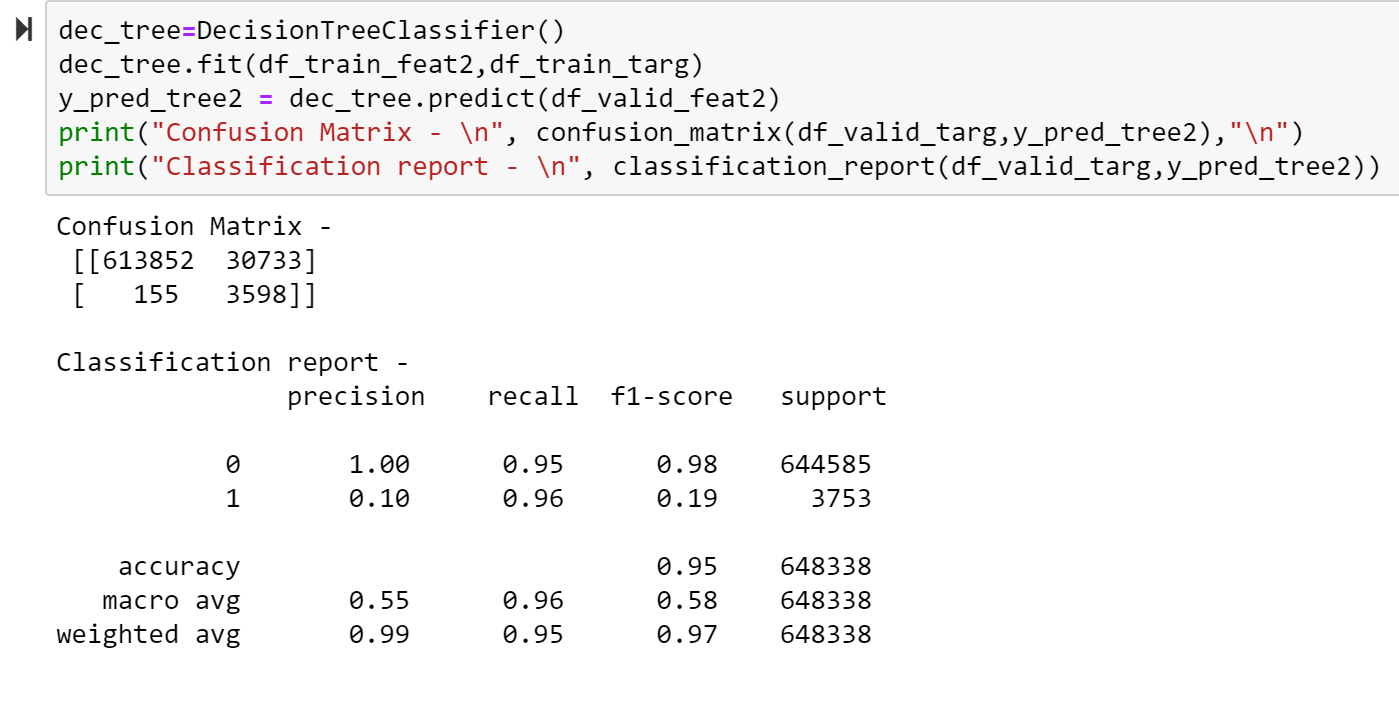


Figure 24- Validation Model After Feature Reduction

After this stage notice the improved metrics all around the board including 96% sensitivity and 95% accuracy with others such as f1-score also improving (although because of the tilted distribution of classes f1-score and precision were known to be bad moving into the evaluation).

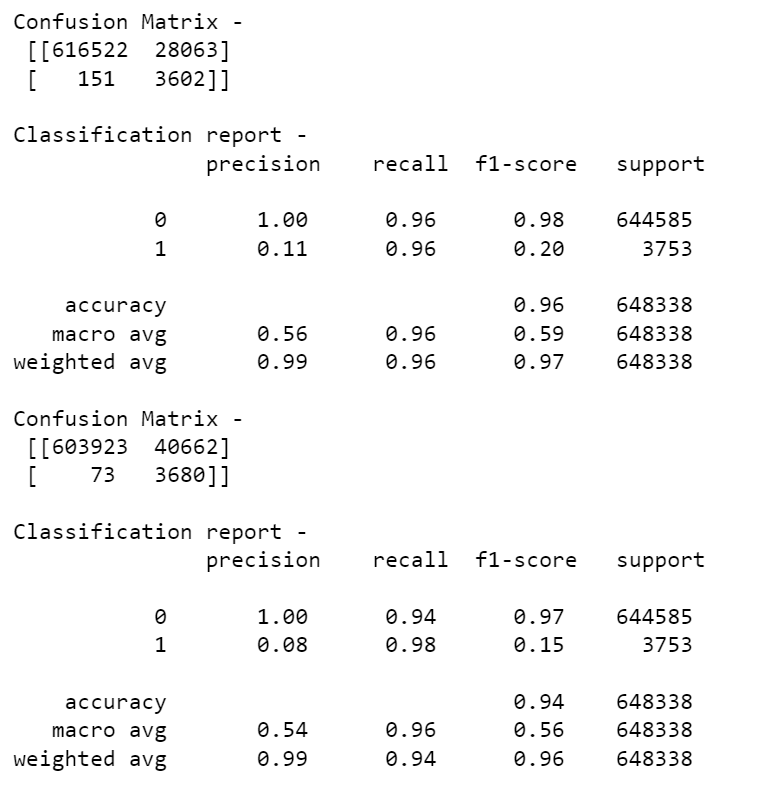
### Cost-Complexity Analysis

Decision Tree Classifiers are notorious for being potentially overfit which was a major concern looking at these results. So, in order to mitigate this, we set out to reduce the tree size based on cost-complexity. This was done through varying the ccp\_alpha input of the classifier and finding the validation and training accuracy and sensitivity. Typically, this is done with accuracy to show where training accuracy far out does validation accuracy indicating signs of overfitting. The code and results of the ccp\_alpha iterations are shown below in Figure 25.



Figure 25- CCP vs Accuracy analysis

After analysis, it was determined that as ccp grew outside the range of this plot the results diverged far away as in the very beginning. Therefore, the plot was trimmed to show just this range of important values but note that the diverged metrics indicate overfitting since the train accuracy is much higher than validation. We selected 2 ccp values based on this plot. We selected .0004 as that is where the overfitting subsides, and accuracy is highest for validation. We also selected .0010 as this is where sensitivity is highest. In figure 26 below see the results of both as well as the ROC curve which proved again to assist in finding the best unbiased option.



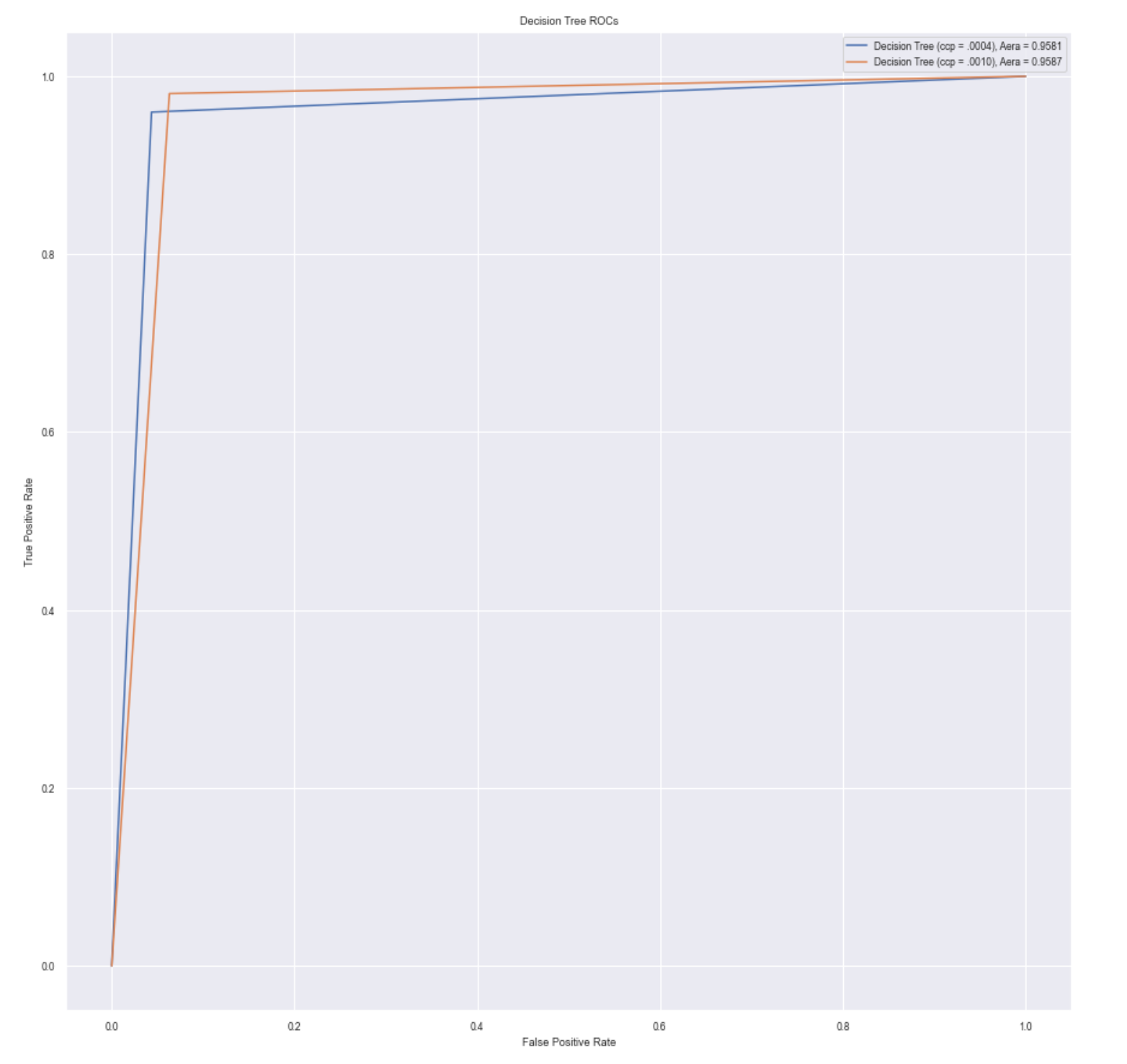


Figure 26- ccp\_alphas’ Results

As can be seen, the two are exceptionally close but when comparing the results, we considered everything including the main points seen in Table 3 below.

Table 3- Final Model Results with cpp\_alpha

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ccp\_alpha value | Accuracy | Sensitivity | AUC | Selection |
| .0004 | 96 | 96 | 0.9581 |  |
| .0010 | 94 | 98 | **0.9587** |  |

# Performance Evaluation

After selecting the Decision Tree Model and optimizing it to use 6 features and ccp\_alpha value of .0010, we wanted to evaluate all our decisions. First, we looked at the difference in tree sizes to see if we did cut down cost complexity. Below in Figure 27, the tree before optimization can be seen on the left and the post-optimization tree can be seen on the right.

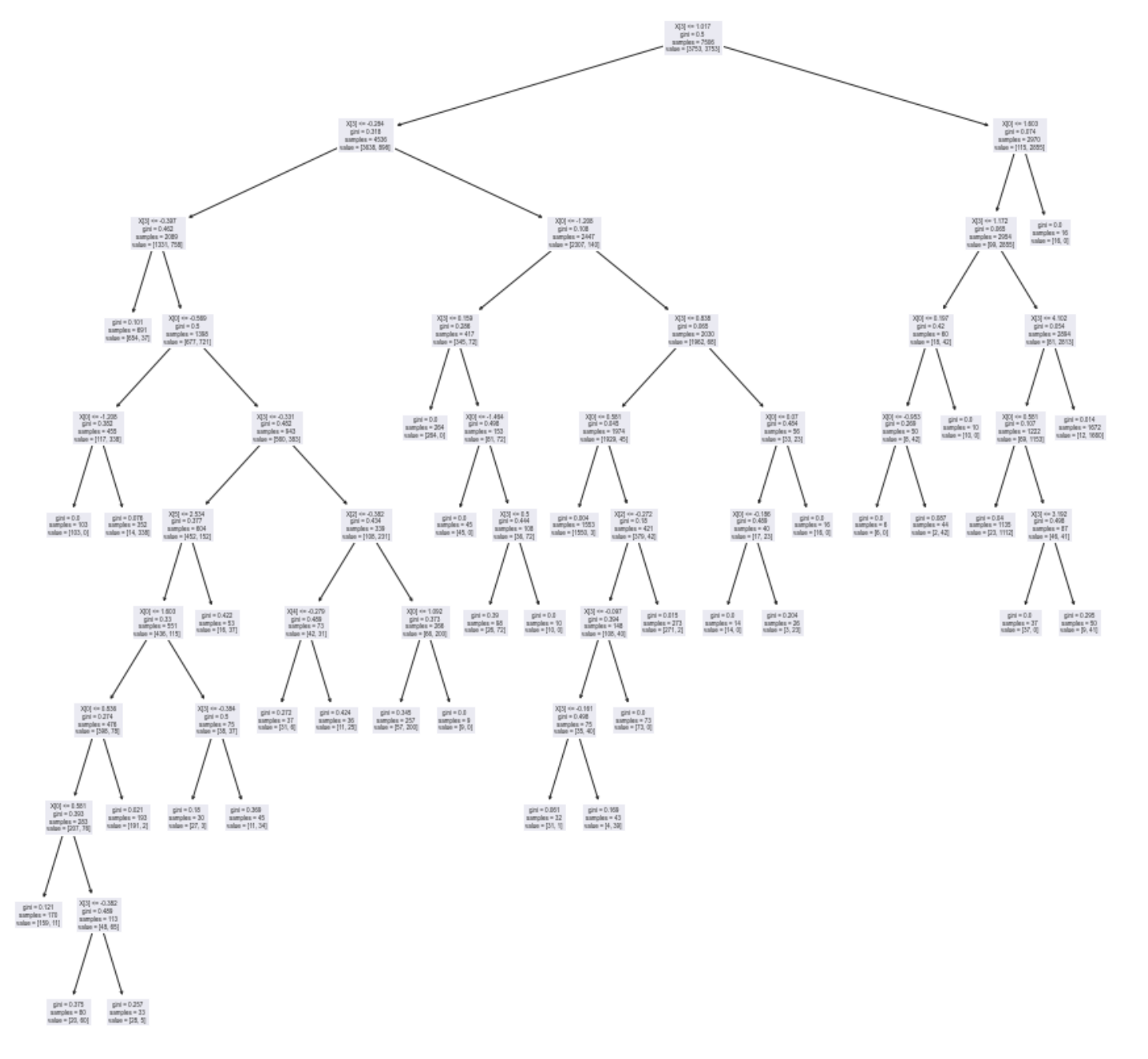
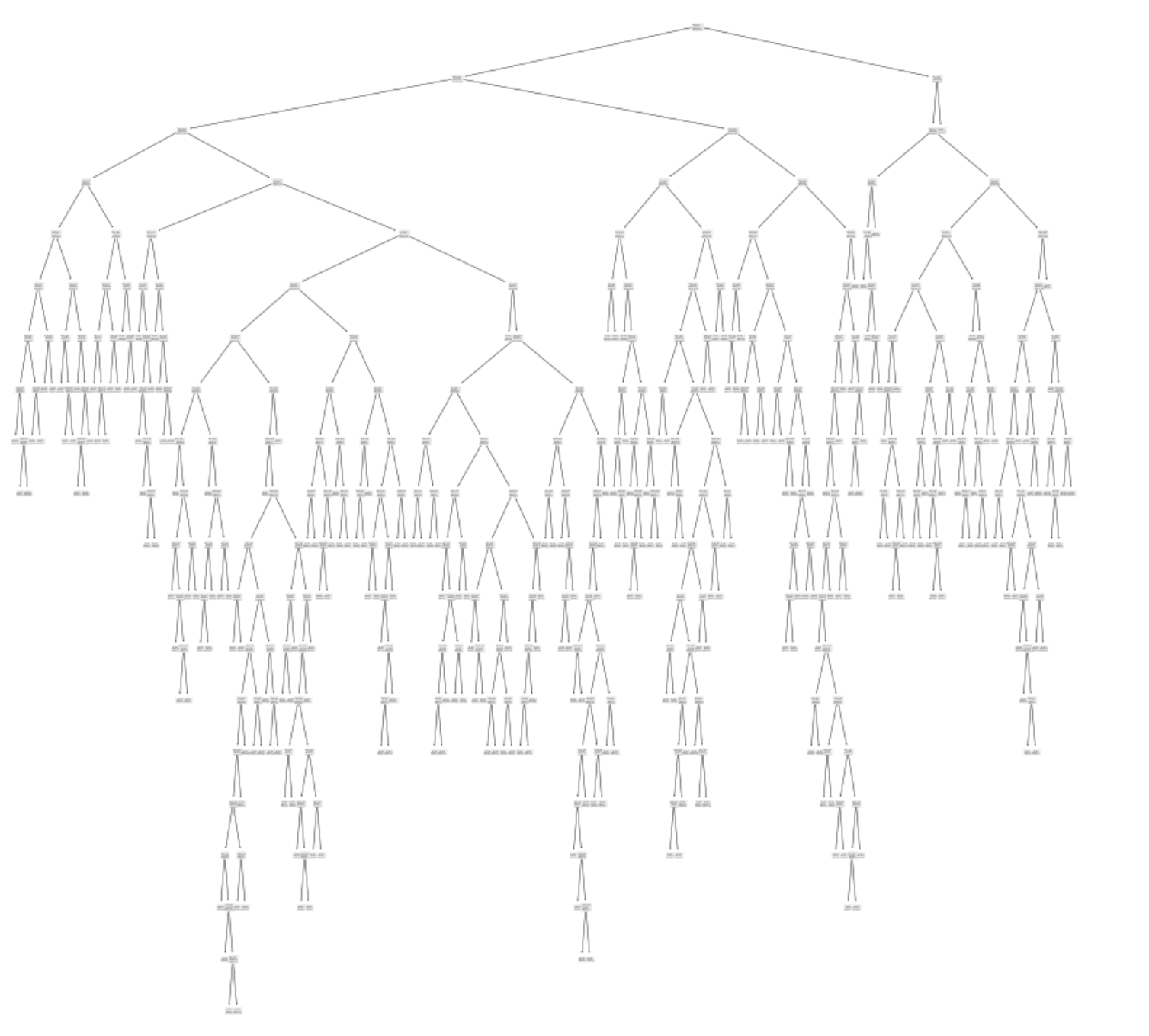
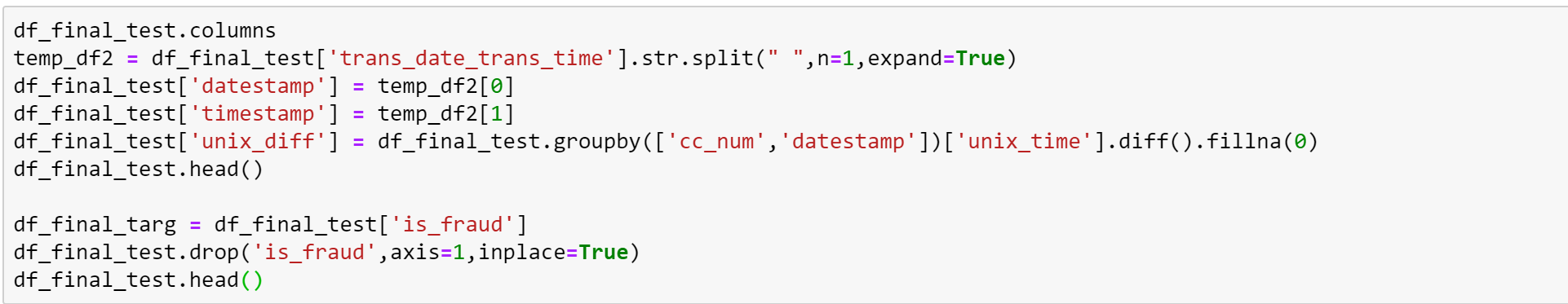


Figure 27- Original Tree (Left) vs Optimized Tree (Right)

Although both trees are shown from a very high level, it becomes much clearer that the optimization process was effective in reducing the complexity of the tree and thus the overfitting potential it had.

Moving forward, we believed we had our finalized model in hand. We therefore wanted to confirm this on a whole new level with a whole new dataset. This is where FraudTest.csv came into play. The dataset described in a previous section was imported and pre-processed including feature reduction down to those 6 features as well as encoded/standardized to match the data required by the model. This process can be seen coded below in Figure 28.





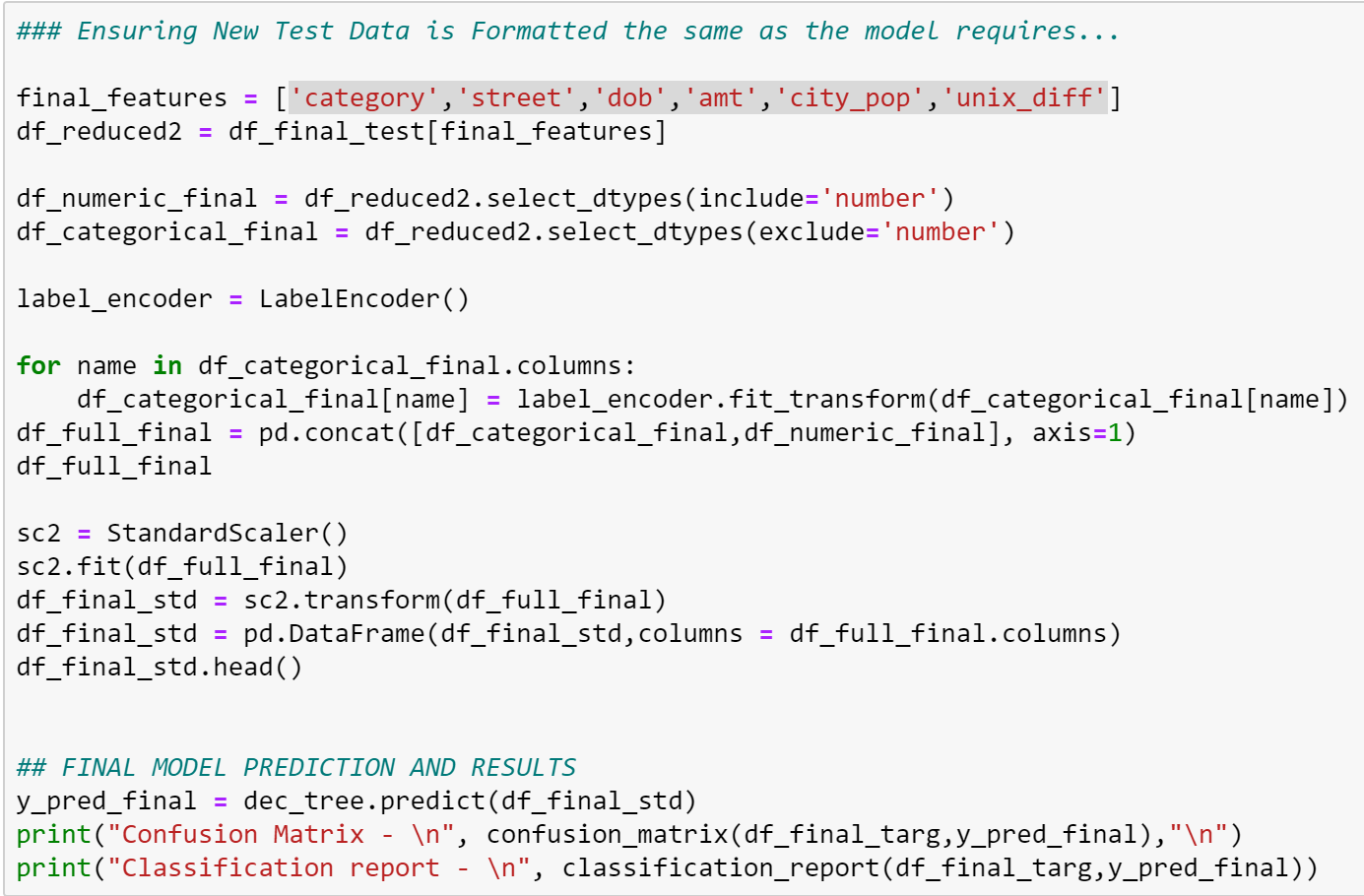
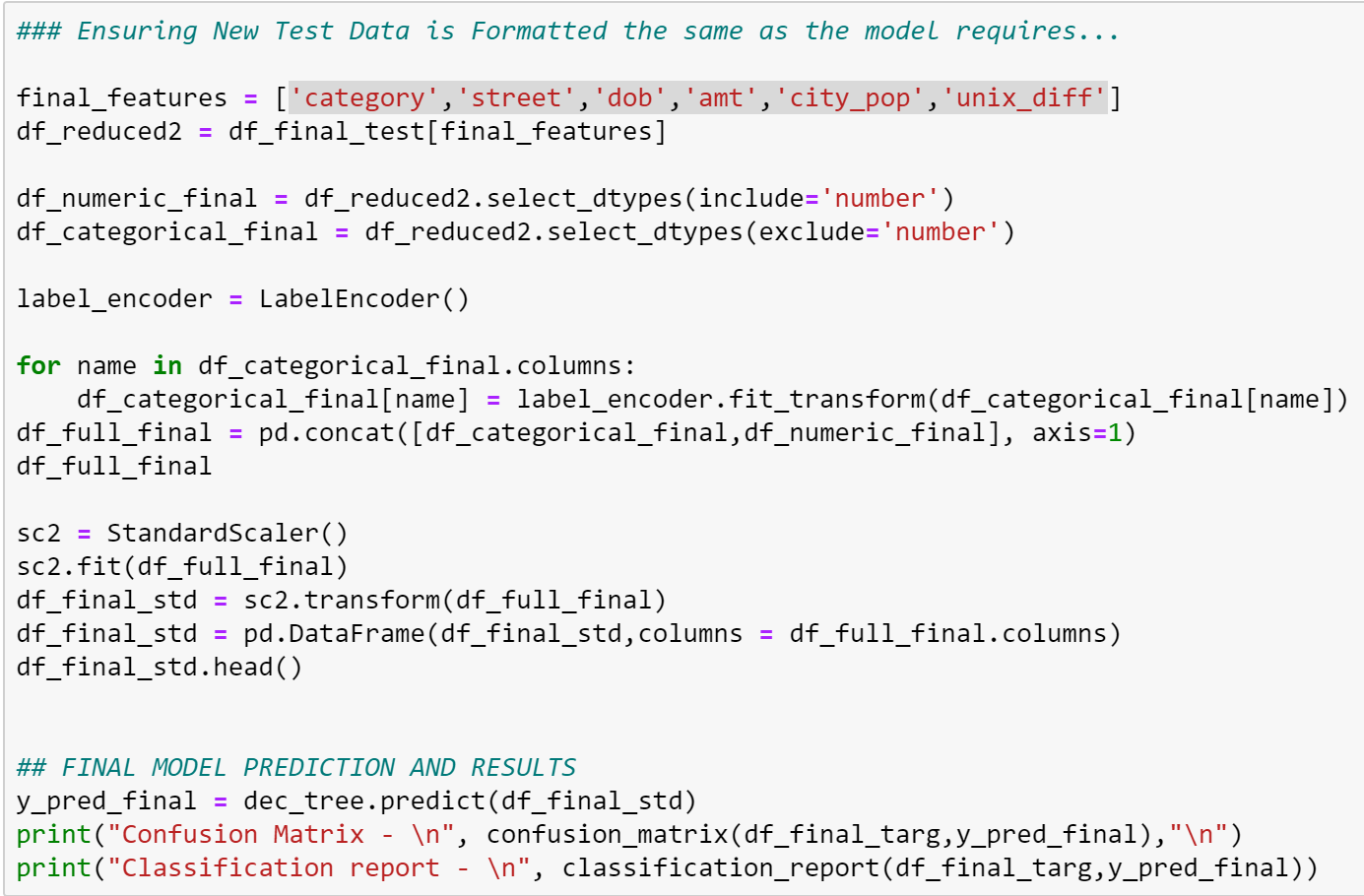


Figure 28- Test Set Preparation

It was then time to finally run the model on the brand-new dataset. This can be seen below in Figure 29 coded and the final results.



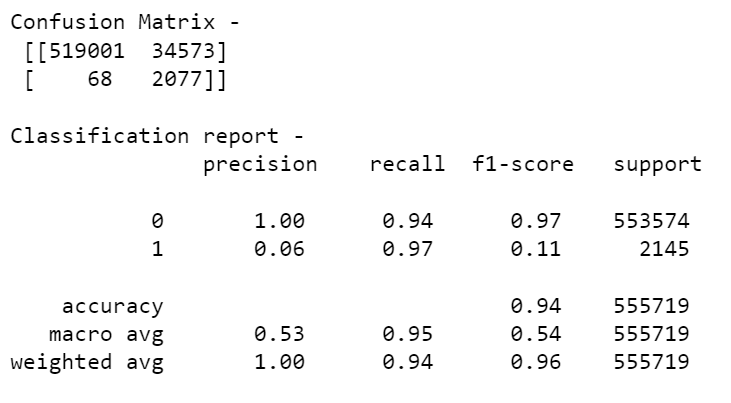


Figure 29- Final Model Execution and Results

# Project Results

In conclusion, the model showed an exceptional 97% sensitivity catching all but 68 cases of fraud and an overall 94% accuracy. This was better than we had hoped for and although we were worried by a plethora of external factors including overfitting and cut off selection, we were able to cross validate the work and minimize the project to require a minimal 6 pieces of information to correctly predict 94%+ cases of fraud. Before considering the project complete, we wanted to visualize one final ROC curve to get a sense of our success. As can be seen below in Figure 30, the results are exceptionally close to the ideal ROC curve with an AUC of 0.9529.

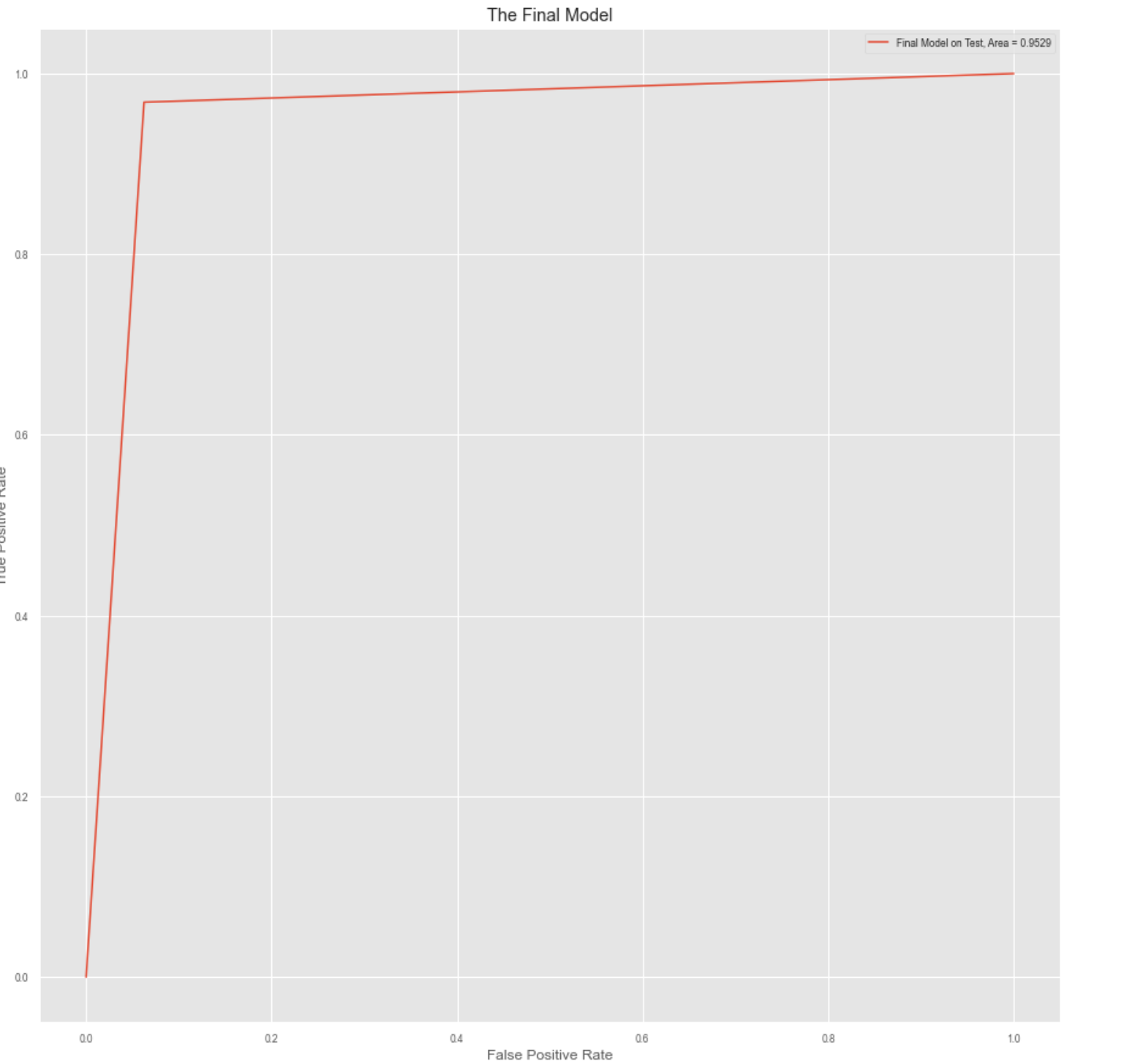


Figure 30- Final Model ROC Curve

Overall, the results were better than anticipated and it can be seen that a great deal of analysis and thought went into each and every decision made. 4 models were tested including KNN, Logistic Regression, Naive Bayes and Decision Tree Classifier. The models were tested repeatedly with various pre-processing and wrangling techniques used starting from multiple failed PCA attempts and restricting data to numeric-only to finishing with label encoding and correlation/unique feature reduction. Sensitivity was prioritized in the end, but the ROC curves and AUC were compared to ensure that the choices were optimized fairly. In the end, Cost Complexity Analysis was used to ensure that the model was not overfit, and complexity was minimized, while the features were reduced down to a mere 6 required for high-level results. All of these methods and results illustrate how each and every goal of the project was accomplished.

# Impact of Project Outcomes

After the conclusion of the project, we observed how impactful our model was. If implemented in a financial institution it would be capable of taking an incredibly small amount of credit card transaction information and determining whether that transaction was fraudulent or real an astonishing 94%+ of the time. The big impact with this model is that it catches 97%+ of the fraudulent cases which proves that almost all fraudulent cases would be caught. Some of the big-name companies mentioned earlier such as Facebook and Yahoo have been a part of the many many companies who have had billions of records exposed in recent years and they undoubtedly could benefit from a model of this nature. They likely have models in place similar to this and much more advanced but to even compare with results is an astonishing feat for what has been learned to this point.

Moving forward, it would be beneficial to work with this data and try other models including Support vector machines (SVM), RandomForest, and Neural Networks which are higher level prediction models. It would also be extremely beneficial to work and minimize False Positives as our model had poor f1 and precision scores. We know this is due to the immense quantity of real transactions, but it would still be an impactful piece of optimization if possible.

Overall, there is no perfect model and we know that. There will always be a tradeoff whether that be accuracy and precision or in the case of this credit card fraud detection project, the ability to maximally catch fraud vs miss fraud but more accurately allow real purchases. As mentioned before, for this model we would rather potentially inconvenience some customers asking them to confirm a purchase vs losing billions of dollars on missed fraud. In the end, no model is perfect but for what we set out to achieve as a goal, this model is an exceptional example of using Data Mining in Credit Card Fraud Detection.

# Sources / Acknowledgements

A few sources that proved useful in this project, whether that be for data, statistics or implementation knowledge, are linked below.

* <https://www.kaggle.com/datasets/kartik2112/fraud-detection>​
* <https://github.com/namebrandon/Sparkov_Data_Generation>​
* <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>​
* <https://data.world/raghu543/credit-card-fraud-data>
* <https://www.fool.com/the-ascent/research/identity-theft-credit-card-fraud-statistics/>
* <https://www.youtube.com/dataschool>