Project description:

This project is the first part of credit card fraud detection, we will be using this 160,000 entry data set with 200 predictors to find a correlation in the data to create a linear model that will accurately predict if credit card fraud has occurred in future instances.

Data visualization:

We used various techniques like violin plots and box and whisker plots to take a look at our different predictors within our data set.

A diagram of a diagram

Description automatically generatedA graph of a function

Description automatically generated

After analyzing the Y or Response column we found that the data had a binary value count of:

0 = 141959

1 = 18040.

0 signified no fraud while 1 signified fraud occurred.

Data preprocessing and splitting:

To preprocess the Data we did a few thing for example, we filled in missing values or ‘NAN’ with a code so we wouldn’t have any holes in our data set.

Another thing we did was creating a X scaled data set so all the data could be scaled to a mean of 0 and STD of 1 to even the differentiation between the different predictors.

After doing the steps above we split the data by creating a code which will split the data into a 30/70 split, meaning 30% is test data and 70% is the training data. Also used a random state and shuffle to prevent order bias on the predictors and ensure the state of data is equal throughout all.

Description of model:

We trained a logistic regression model for this data set.

After research we came to a conclusion to try out the resampling method SMOTE. We used this to try balance out our dataset and better representation of the minority group.

To gauge our prediction accuracy we will be using AUC score to determine it which our score was 0.86 which is in the good range.

A blue line with a black line

Description automatically generated with medium confidence

Confusion matrix:

[[33703 8995]

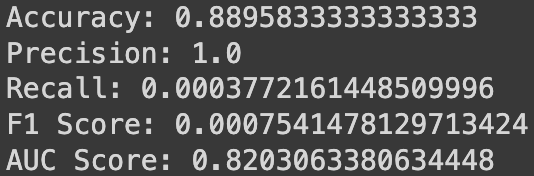
[ 1223 4079]]

I think the following model produced the following results because it is using all the predictors to find a trend to be able to predict if credit card fraud is occurring or not. To improve upon this project we need to figure out how to strengthen the data correlation into the model.

Part 2:

# Random Forest:

The first model we selected for the second deliverable is a random forest model. A random forest model is a tree-based model that creates multiple trees and then combines them to find an answer. The reason we chose this model was because it is a very popular model used in this industry and we were curious how it could improve the performance of our predictive model.



Our Random Forest model produced the performance metrics above. As one can see our random forest model produced an Accuracy of 0.8896 with our testing set, which is really good as it represents a pretty accurate model. This Model also produced an impressive precision of 1.0 as it did not produce any False positive (Confusion matrix will be displayed below). Our F1 score is 0.0007 which is really low so that means our Recall for this model was also very poor. This could be a bad sign on moving forward with this model. Our AUC score was 0.82 which is in the good range (0.8 – 0.9), this shows our model had good discrimination capability which means it does a good job choosing between positive and negative. Below is the visual representation for the ROC curve and Confusion Matrix that represents the numbers above.

A graph with numbers and a blue square

Description automatically generated

A graph of a curve

Description automatically generated

# Gradient Boosting Machine (GBM):

The second model we chose was GBM also known as Gradient Boosting Machine, the reason we chose this model was to compare the random forest model to a similar tree-based model to see which will perform better in the scenario. This model ended being the one we chose to test against our data set as it seemed to be a better fit. We will go over the results below.

A number on a black background

Description automatically generated

As we can see above the Accuracy in this model is 0.907 which means this model is more accurate than random forest. The precision is 0.7 which is not bad but obviously 1.0 is better from random forest. This model still wins in this aspect because the F1 score is 0.393 which represent between precision and recall this model is more balance with the data. The AUC score is 0.855 which is higher than random forest which means this model is even better at discriminating between a positive and a negative. Overall this model had better performance than random forest and that is why we chose it to test it on our test data set. The results will be in the CSV file. Below I will show the Confusion matrix and ROC curve that will visualize the data shown above.

A blue and white graph

Description automatically generatedA graph of a curve

Description automatically generated

# Naïve Bayes:

For deliverable 3 we decided to use a Naïve Bayes model for the reason of its computational efficiency to see how it compares to our tree-based model. We applied smote to balance out our imbalance data set to help the model train the positive and negative results well. The results from Naïve Bayes are the one below:

A black background with white numbers

Description automatically generated

The Accuracy for this model is pretty good sitting at 0.85 but the rest of the scores are a little concerning. The model had a F1 score of 0.12 resulting from a precision of 0.2 and a recall of 0.09. Both of these scores are relatively low meaning the model did not demonstrate a good job with the positive class. The AUC score was also very average being a 0.60 which represents the model did not distinguish classes that well either. According to the results given I would not use this model over Random Forest or GBM. Both of the tree models were a lot more accurate so the computation saved with Naïve Bayes is not worth it in this example. The Confusion matrix and AUC graph will be shown below for visual demonstration.

A graph of a curve

Description automatically generated

A blue and white graph

Description automatically generated