# Santander Transaction Prediction

Group 18: Yichen Zuo, Tian Pei, Xiaodu Huang

*Statistical Data Mining - Final Project*

---

**Abstract**

The purpose of this report is to provide a classification model for Santander bank in order to determine which customer will make a transaction using an untagged dataset of 200 variables and 200,000 training observations. This report will provide a summary of the dataset provided to train the model and preliminary data exploration of the potential model features. The report will also outline methods attempted to reduce the dimensionality of the dataset. The report will then go over the best resulting model (lightGMB) and other models attempted to classify which customer will make a transaction. Finally, the report will provide details on the best model selected and highlight difficulties encountered in the project and suggestions for improvements going forward.

---

## 1. Background and Data

### *1.1. Description of Variables*

The purpose of this report is to predict whether if a customer will make a transcation in the Santander bank data. The report will provide an overview of the variables and exploratory data analyis. We were provided a training dataset with 202 variables 200 of which are features we can train our models on. The training dataset had 200,000 observations that we need to use to predict the target variable for the 200,000 observations in the test dataset. The variables were provided to us with no descriptive variable names and no additional explanation in the form of a data dictionary. As a result, we did not have a way to logically infer which are variables of special importance (in case this could be things like the age of the account, the net worth of the customer, and other demographic features). In addition, it was hard for us to add additional features from the existing list of variables as we were unsure what the relationship of the different variables are and whether if summary variables such as PCA vectors were already included in the 200 provided to us.

We observed that the the dataset is highly unbalanced where only 10% of the training data is tagged as 1 in the variable **target** which is an indicator flagging whether if a customer made a transaction. This is an important feature to keep in mind as we may need to oversample the customers that made a transaction to create a better model. A bar chart showing the unbalanced distribution in the training data can be seen below:

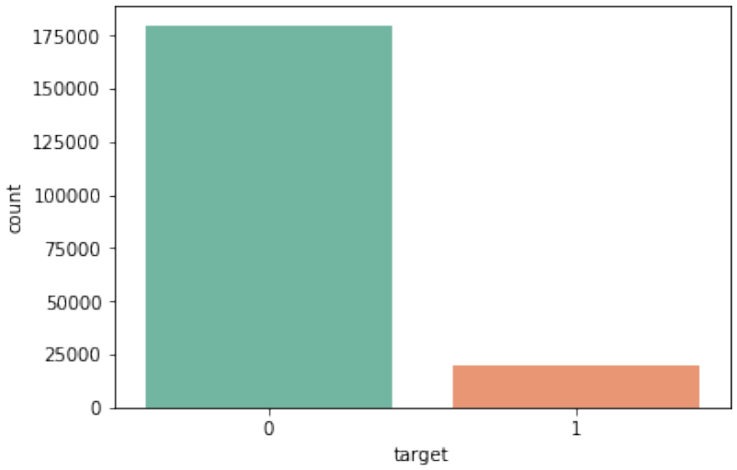


Figure 1: Distribution of the target variable

There are several ways to mitigate Class Imbalance Problem:

Cost Function Based Approaches: One false negative is worse than one false positive, we will count that one false negative as, e.g., 100 false negatives instead.

Sampling Based Approaches:

1.Oversampling, by adding more of the minority class so it has more effect on the machine learning algorithm
Weakness: overfitting
Methods: RandomOverSampler SMOTE

2.Undersampling, by removing some of the majority class so it has less effect on the machine learning algorithm
Weakness: may lose useful information
Methods: RandomUnderSampler TomekLinks ClusterCentroids

3.Hybrid, a mix of oversampling and undersampling
Weakness: trade-off
Methods: SMOTETomek

We will address some of the methods listed later in the report.

We also plotted the distribution of all 200 variables to see if there are any obvious instances where the distribution of customers who made a transaction visually differed from customers who did not make a transaction. The full list of distribution chart can be found in the appendices, below is a few variables we flagged as potentially important ones for differentiation and selected to feature in the report and also the summary statistics of columns and rows by the target feature:

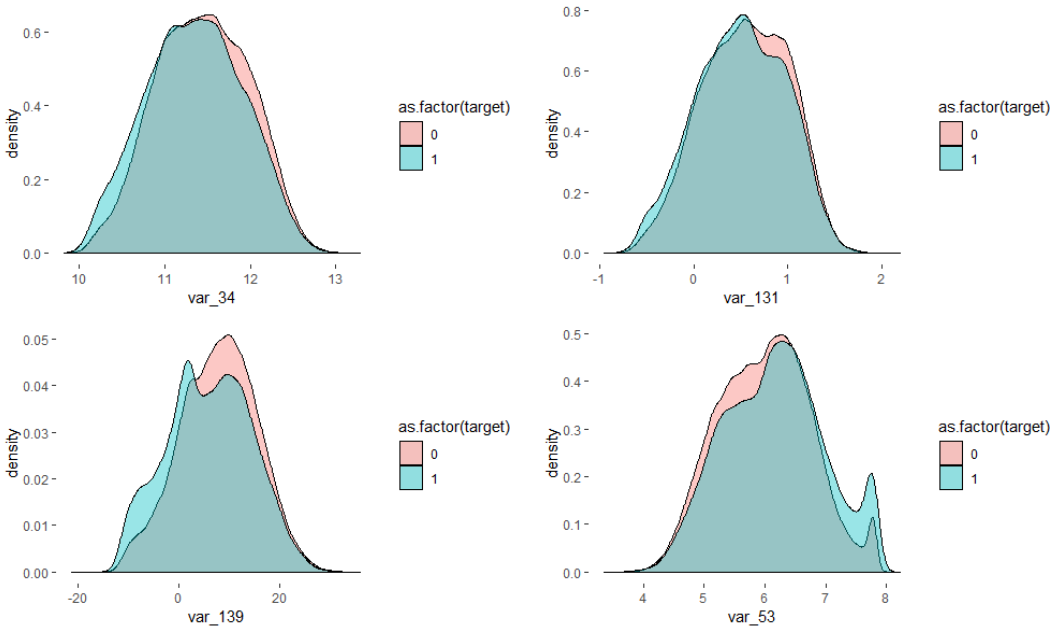


Figure 2: Distribution of Selected Variables by Target

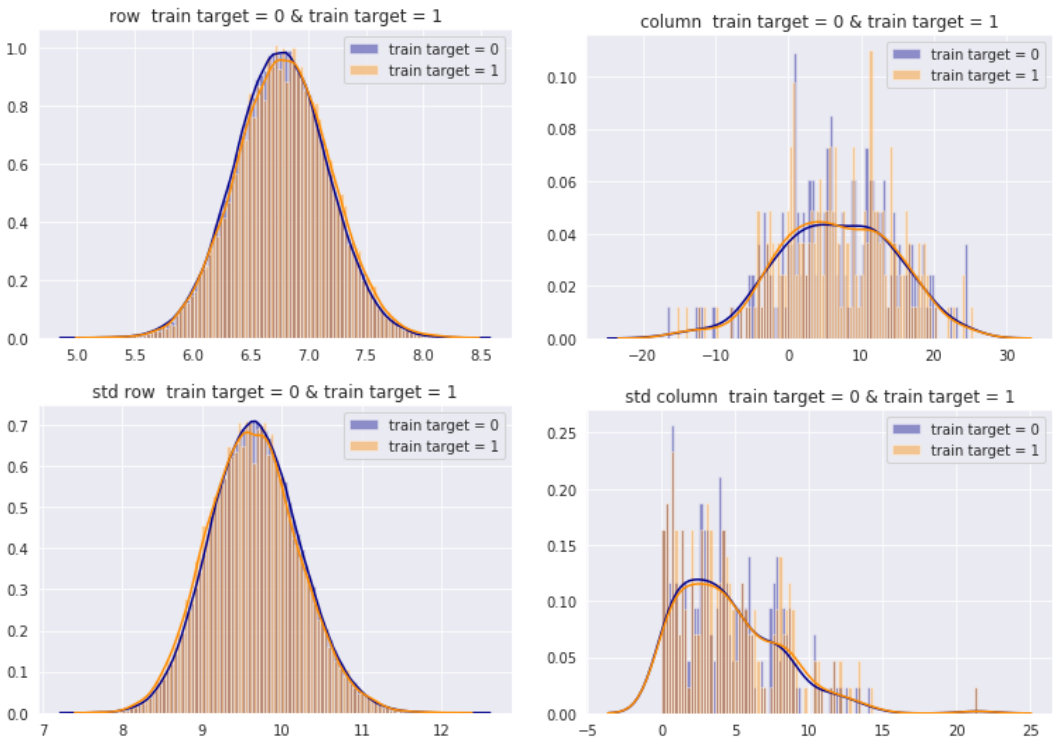


Figure 3: Distribution of Summary Statistics by Target

## 2. Creating Models

### *2.1. Dimension reduction*

As we have 200 potential features, we explored options for dimensionality reduction. A simple approach we first attempted was principle component analysis (PCA). Unfortunately, ths did not yield an easily implementable result as there was no clear "kink" and all the different components explained around the same amount of variance in the data. As no data dictionary was provided, this could very possibly be the result of inclusions of principle components as one of the 200 features. In short, PCA analysis was not a successful way to select features. See a plot of this below:

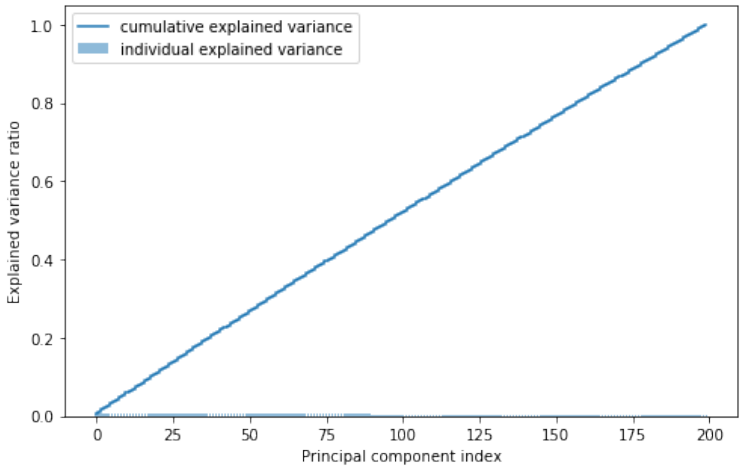


Figure 4: PCA Analysis

We also tried to use Lasso regression as a way of reducing features. This method yielded better results than PCA analysis and after training the lambda parameter we were able to identify 22 variables to exclude including the following variables: 185, 183, 160, 161, 158, 136, 124, 126, 117, 103, 100, 98, 96, 46,41, 38,39, 30, 27, 17, 10, and 7.

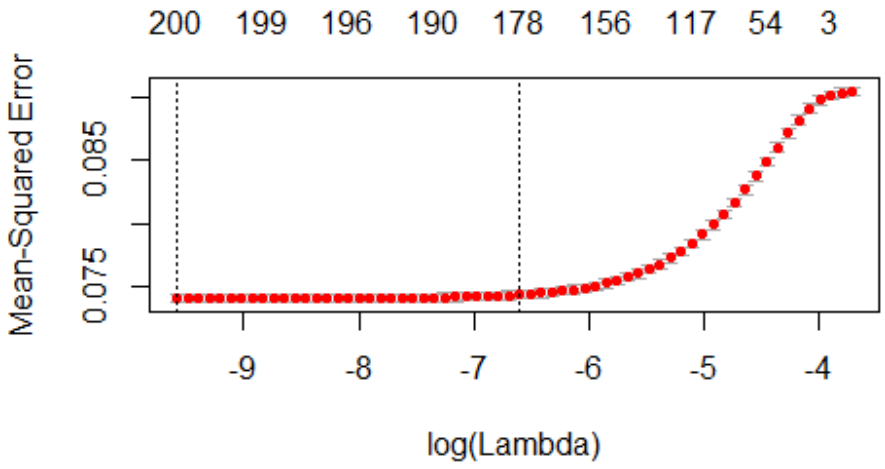


Figure 5: PCA Analysis

*2.2. Best Model*

Our best model was created using boosting - specifically **lightGBM**. We achieved an AUC score of 0.896 using this method after tuning.

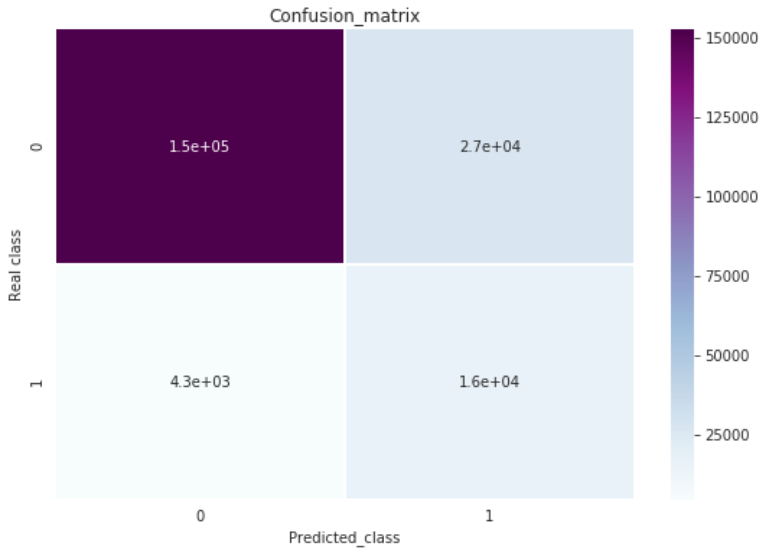


Figure 6: Confusion Matrix from lightGBM model

We also used **lightGBM** as a way to do feature selection, while there is no dramatic kink below, we can see that the feature importance does drop around the green section of the plot below:

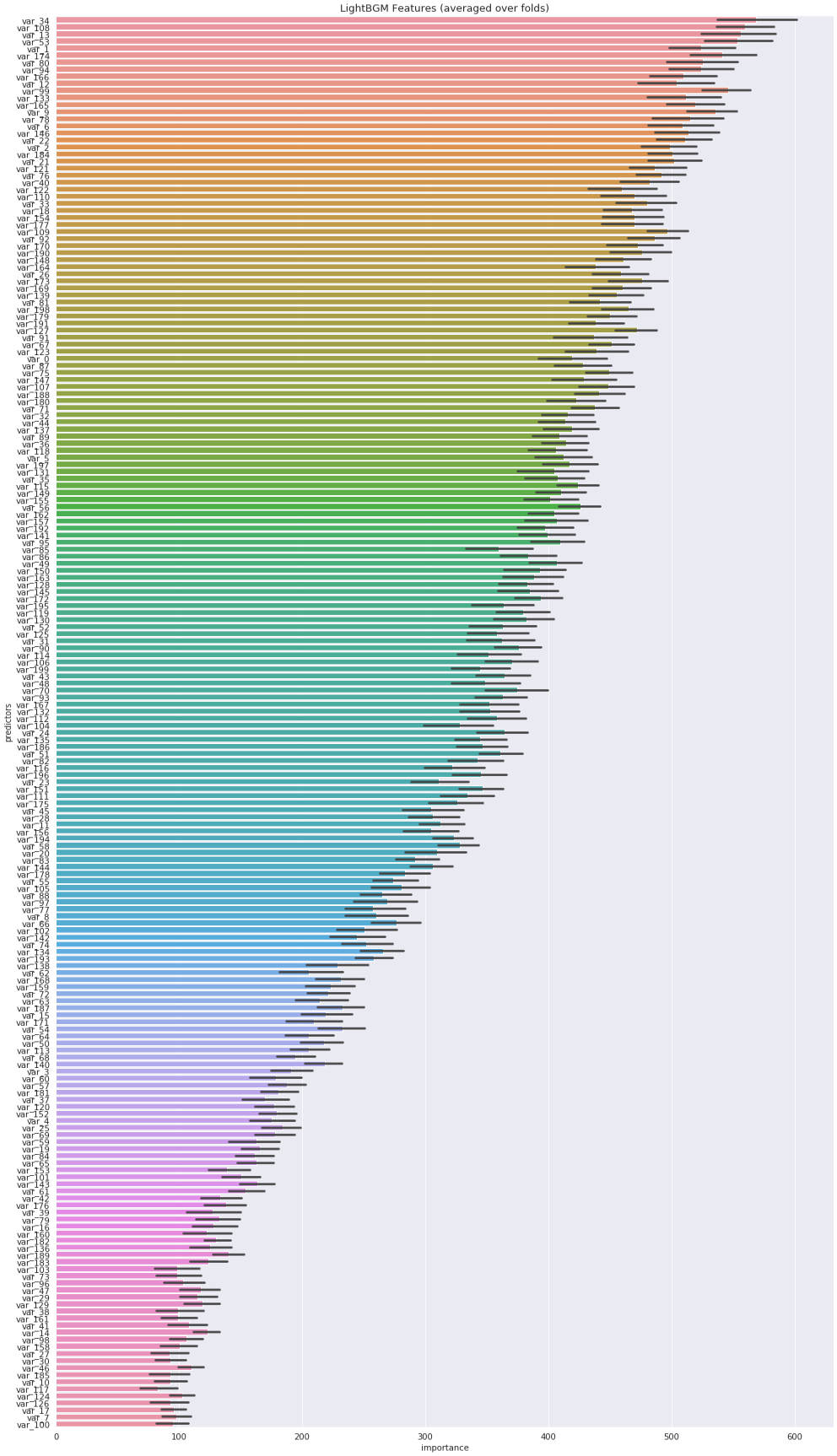


Figure 7: LightBGM Features (averaged over folds)

*2.3. Other Models Explored*

Our other successful models were created also using boosting techniques. With **xgBoost**, we achieved a AUC score of 0.894 on Kaggle which was right behind our best model. We tuned the parameters of this model to improve the score from 0.87 to 0.894.

With gradient boosting, we achieved a AUC score of 0.895 on Kaggle after tuning the paramters - tuning the parameters helped us to improve the AUC score from the low 0.80's to almost 0.9. Below is our feature selection results from using gradient boosting:

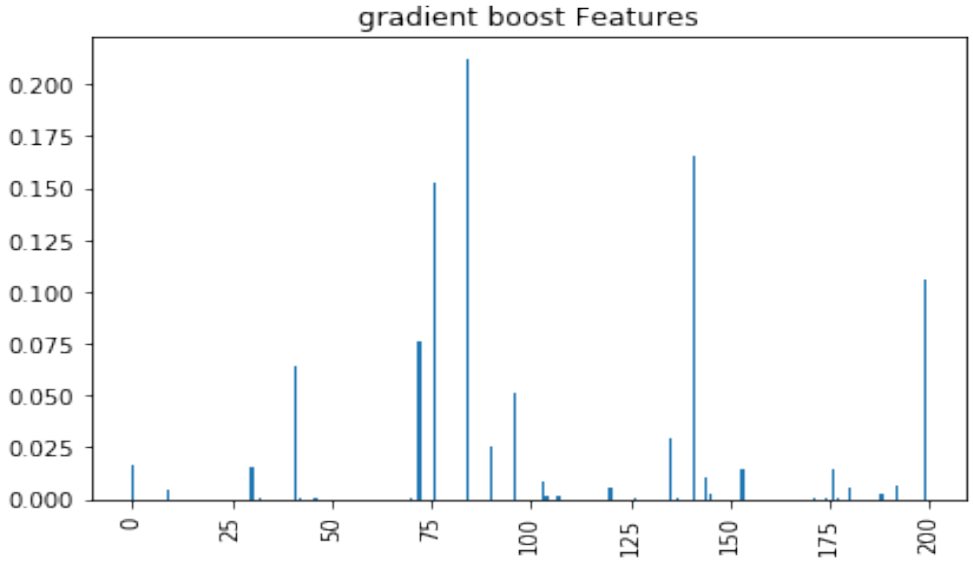


Figure 8: Key Features from Gradient Boost

| | predictot | importance |
|---|---|---|
| **0** | 0 | 0.016861 |
| **30** | 30 | 0.015679 |
| **41** | 41 | 0.064624 |
| **72** | 72 | 0.075694 |
| **76** | 76 | 0.152214 |
| **84** | 84 | 0.212018 |
| **90** | 90 | 0.025066 |
| **96** | 96 | 0.051507 |
| **103** | 103 | 0.008187 |
| **120** | 120 | 0.005964 |
| **135** | 135 | 0.029404 |
| **141** | 141 | 0.165352 |
| **144** | 144 | 0.010113 |
| **153** | 153 | 0.014563 |
| **176** | 176 | 0.015013 |
| **180** | 180 | 0.005254 |
| **192** | 192 | 0.006859 |
| **199** | 199 | 0.106048 |

Figure 9: Key Features from Gradient Boost

SVM classification uses a series of support vectors to classify the data. We tuned the SVM paramters on the 1500 observations from teh training dataset as the dataset has so many features. We attempted to use Support Vector Machine (SVM) to classifiy the data, however this procedure was extremely resource intensive and yielded mediocre AUC scores on Kaggle. The model yielded We chose not to pursue this model after inital testing on a smaller subsample of the data. The SVM model yielded very high false positive results:

| Predicted | False | True | __all__ |
|---|---|---|---|
| Actual | | | |
| False | 1234 | 134 | 1368 |
| True | 0 | 132 | 132 |
| __all__ | 1234 | 266 | 1500 |

Figure 10: SVM Confusion Matrix

To kick off our model building, we also begun with logistic regression the get a sense of the data's features and complexities and to have a baseline upon which we built more complicated models. We noted that a logistic regression with all possible features yielded a result where most features were statistically significant at the 5% level. After splitting the training data into 70% training and 30% validating subsets, we noted that using logistica regression yielded the following confusion matrix below. The model yielded alarmingly high false negative rate at 72% depsite a high accuracy ratio of over 90% on the subsetted validation dataset:

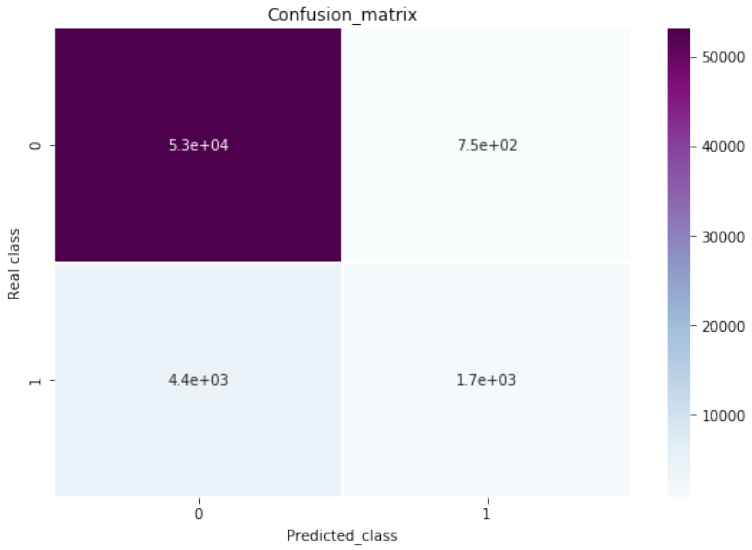


Figure 11: Logistic Confusion Matrix

We also tried linear discriminant analysis to classify the data. We further divided the training data to half training and half validation. Using the basic LDA classification we achieved an AUC of over 0.9 on the validation subsample and an AUC score of 0.86 on Kaggle.

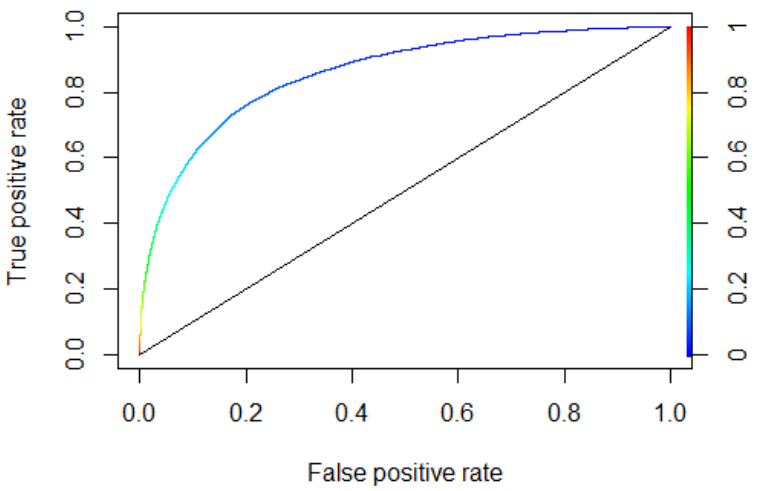


Figure 12: LDA ROC Curve

*2.4. Unsupervised Learning*

To implement an unsupervised learning method besides PCA, we tried K-means clustering. K-means performed poorly on the training dataset as the accuracy of clustering on the training data was only 26%. The ROC curve can be seen below. The poor results of the K-mean clustering just further shows us that the data is extremely noisy and the simple minimization of the Euclidean distance is not enough to capture the complexity of the dataset and underperforms in the case of high dimensional datasets.

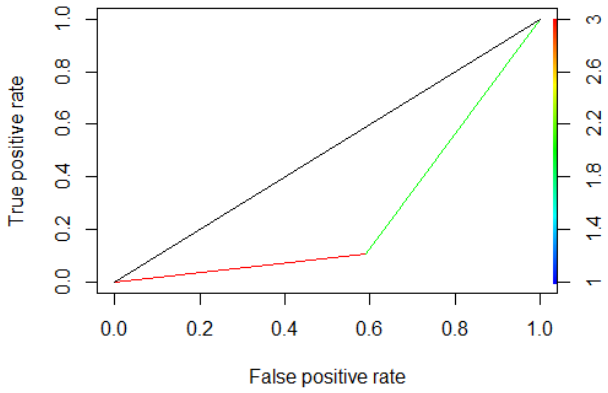


Figure 13: K-Means ROC Curve

## 3. Discussion

### *3.1. Overview of Best Model - lightGBM*

Light GBM uses a decision tree algorithm framework to perform classification tasks and it differs from other boosting algorithm by spltting the tree leaf-wise than depth-wise. Boosting helps model performance as our model builds sequential trees where each subsequent tree reduces errors of the previous trees and learns from the errors of the previous trees to minimize the loss function and iteratively pushes the model to its best performance given the parameters we specified. It is also less resource intensive than other boosting algorithmns in terms of memory usage especially in comparison to xgBoost. Our best model used a 10-fold cross validation. We selected a 0.01 learning rate for the final model and 13 leaves which determines the number of child nodes in a tree. As the data is unbalanced, we set the the unbalanced model parameter to TRUE. We set the bagging frequency to 3 and the bagging fraction to 0.5. We did not set a limit to the maximum depth of the tree.

### *3.2. Hurdles Encountered*

The biggest difficulty my group faced was dimension reduction. As we did not have any information on the dataset provided, we were not able to do a "common sense" interpretation of the variables we have on hand to determine important features and also to determine if there are sensible variable transformations we can implement such as taking the log of networth and etc. The number of variables also caused computational difficulties as techiniques such as SVM became very computationally burdensome. With more resources, we can improve our model by doing parallel processing to implement some of these methods. We would have also liked to use neural networks but we also found that to be too burdensome computationally.

As this project happened at the end of the semester, we had to work remotely to complete the tasks as one of us has already gone back home to China and one of us has to work full time and had a tight work deadline that led to a 80 hour work week right before the project deadline. We did our best trying to split up work and coordinate.

### *3.3. Ideas for the Future*

We would have liked to do some in depth variable exploration to infer what the variable is supposed to be. For instance, the winning teams on Kaggle discussed how they removed "fakes" from the training dataset. The winning team also created a lot of extra features that categorically counted the number of times certain value appear in the variables and focused on observations that are more "unique" compared to the massess. As we spent a lot of our time on dimension reduction, we did not attempt anything similar. We would have explored some of the bimodally distributed variables in the dataset and performed variable transformations to heighten the effects of those distributions. While the winning team also used lightGBM like our best model, they spend a lot more time on exploring the features in the dataset despite the lack of information provided by Santander.

## 4. Appendices

Please see the two separately submitted Python Jupyter notebooks and the separately submitted R codes.