Data Capstone Project 2

Eric Huynh

April 2 Cohort

Problem/Overview:

Financial institutions (e.g. banks) refuse a countless number of loan/credit applications to people in need each day, but they have a cause for doing so. These institutions put themselves at monetary risk by lending credit to others, especially with no guarantee of the money's return. As a result, financial institutions are meticulous and stringent on the qualifications required to request large loans. This means that they will only approve credit applications to clients that have a strong record for repayment, as well as proven financial means to repay the loans.

Although this strategy minimizes the risk of a loan default, it prevents the newer clients from receiving the loans they need. This is especially true since the only way to obtain good credit is to consistently spend smaller amounts of credit over a longer period. Those individuals uninformed about credit are unable to build their credit early, resulting in difficulty applying for larger loans at a later age. As a result, some of these individuals are forced to become patrons for unreliable lenders.

To address that, some financial institutions have taken the risk and strived to expand the financial inclusion to those with insufficient or non-existing credit histories. These institutions utilize a variety of alternate data to gauge and measure their clients' risk profile.

My objective for this project will be to experiment with a sample of credit data provided by Home Credit Group, an international financial institution focused on lending to the population with little to no credit history. I will analyze the data and generate a predictive model that can reliably classify each client as high-risk or low-risk for credit loans.

Potential Clients:

The potential clients for this project include two groups: financial institutions and loan applicants. All financial institutions can utilize this data to expand their loans applicant pool. If the model can reliably predict high-risk and low-risk loan applicants via alternate data, then the risk of defaults can be minimized. With a larger approved applicant pool, financial institutions will earn larger returns. Additionally, when a client has a positive loan experience, they are more likely to return as a reoccurring applicant. As a result, this would yield higher returns both short-term and long-term for the financial institutions.

Loan applicants, especially applicants with near non-existent credit history, can utilize this information to see and understand alternative attributes that can help them develop towards becoming a low-risk applicant despite short credit histories. However, this information is applicable across as larger spectrum, essentially to anyone who is applying for a loan they may not currently qualify for under normal financial institutions' requirements. This may help the applicant reliably acquire the money they need.

Data Wrangling:

This dataset was provided by the Home Credit Group, an international financial institution focused on providing for underserved groups of the population with insufficient credit histories, to a currently ongoing Kaggle Competition. The data consists of one training dataset supplemented with six other datasets of varying information. No exact time-frame is provided, but there are records for up to 8 years of prior credit history relative to the time of application data. Overall, the model will be trained with over three-hundred thousand unique applicants, and any associating information. This data contains over a total of three hundred attributes, consisting of a heterogeneous mixture of both categorical and quantitative information.

The Kaggle Dataset was already cleanly organized with specified merging columns between each of the seven datasets. However, a complete merge would result in over three hundred attributes in one table. As a result, the first step was to perform some feature selection.

Feature Selection:

With seven datasets and hundreds of features, I chose to go through each dataset one at a time, starting with the training data. I performed the same routine of feature selection, feature engineering, and data cleaning to each of the datasets. However, I chose to leave out two provided datasets, bureau_balance.csv and POS_CASH_balance.csv, because I felt they did not provide any additional details.

With my limited domain knowledge, I could not accurately select features with importance to finance data. However, with my limited computational power, I could not try every single variable to calculate feature importance. As a result, I performed some manual feature selection based on an informational .csv file containing information regarding every feature. My focus was to remove columns that provided redundant information, then select specific features that made practical sense in relaying information of the clients' characters. To supplement this, I used RandomForestClassifier and seaborn to plot the feature importance for the training data. From the plot, I decided to add an additional two variables that I previously left out.

Additionally, I added columns that I would use in feature engineering. From these select columns, I performed some feature engineering and added it to the columns of interest. For each of the supplementary datasets, I grouped the data by their unique client IDs so that each client could be represented by a single row. I also reduced the credit history to the last 12 months (if available) to reduce the weight of credit history when developing my model.

Data Cleaning:

Next, I began to address the null values in each of the datasets, filling them with values I felt were appropriate (e.g. 0, 1, -1, etc.) based on their numerical representations. Additionally, I sought out odd values that did not make contextual sense (e.g. amount of days since employment

= 365243) and attempted to understand them and replace them with their appropriate representations. For the extreme outliers, I chose to scale them towards the second largest extreme value instead of outright removing them because, with the provided data, I could not determine if the value was a mistake. I removed a couple of columns from the datasets as I found them to be redundant with other columns.

Within the training dataset, I added two domain features inspired by Aguiar's script for the same Kaggle competition. PAYMENT_RATE was defined by AMT_ANNUITY divided by AMT_CREDIT, and ANN_INCOME_PERC was defined as AMT_ANNUITY divided by AMT_INCOME_TOTAL.

Once I cleaned every dataset individually, I began to merge the supplementary datasets to the training dataset with a left merge to retain all the unique client IDs in the training data (in this case, I don't need information about the clients not in the training data). With every merge, I needed to fill in the null values in each column for clients that had no prior history in that database. One surprising merge was with the credit card balance history, where there were more than 220,000 clients without history. The null values were filled accordingly to represent missing data. The completely merged data had a total of 44 heterogenous columns.

After ensuring that the merged training data was completely cleaned, I applied the same data wrangling, feature selection, data cleaning, feature engineering, and data merging for the test data.

Exploratory Data Analysis and Data Story:

The data analysis started with the correlations of the numerical columns in the fully merged training data. To prevent multicollinearity, I used a heatmap to find and visualize variables that were highly correlated (Figure 1). In total, there were 6 notable positive correlations (> 0.35) and 5 notable negative correlations (< -0.35).

From the 12 noted correlations, I only acted on three sets: (AMT_CREDIT, AMT_ANNUITY, AMT_GOODS_PRICE) ,(DAYS_DECISION, DAYS_TERMINATION) and (CNT_CHILDREN and CNT_FAM_MEMBERS). The first set is represented by the 3x3 white block on the top left corner of the Figure 1, whereas the second send is the 2x2 white block towards the bottom right. The bottom left white block represented the last correlation. I chose to remove AMT_GOODS_PRICE (total cost of goods loan is being applied for) because it had the higher correlation to the other two variables, DAYS_DECISION (date when last application was approved) because I felt it offered less information regarding the client, and CNT_CHILDREN because it was a factor of CNT_FAM_MEMBERS.

However, to better understand the correlations, I plotted a couple of the correlations on regression plots (Figure 2). None of the plots showed any real correlation, and the distribution appeared random.

Next, I attempted to utilize a series of boxplots to visually search for statistical differences between target and non-target groups (Figure 3). For the most part, the boxplots were obscure and difficult to draw any conclusive insights from. As a result, I chose to rely on inferential statistics to test for statistical differences between the target and non-targets groups in respect to each numerical variable.

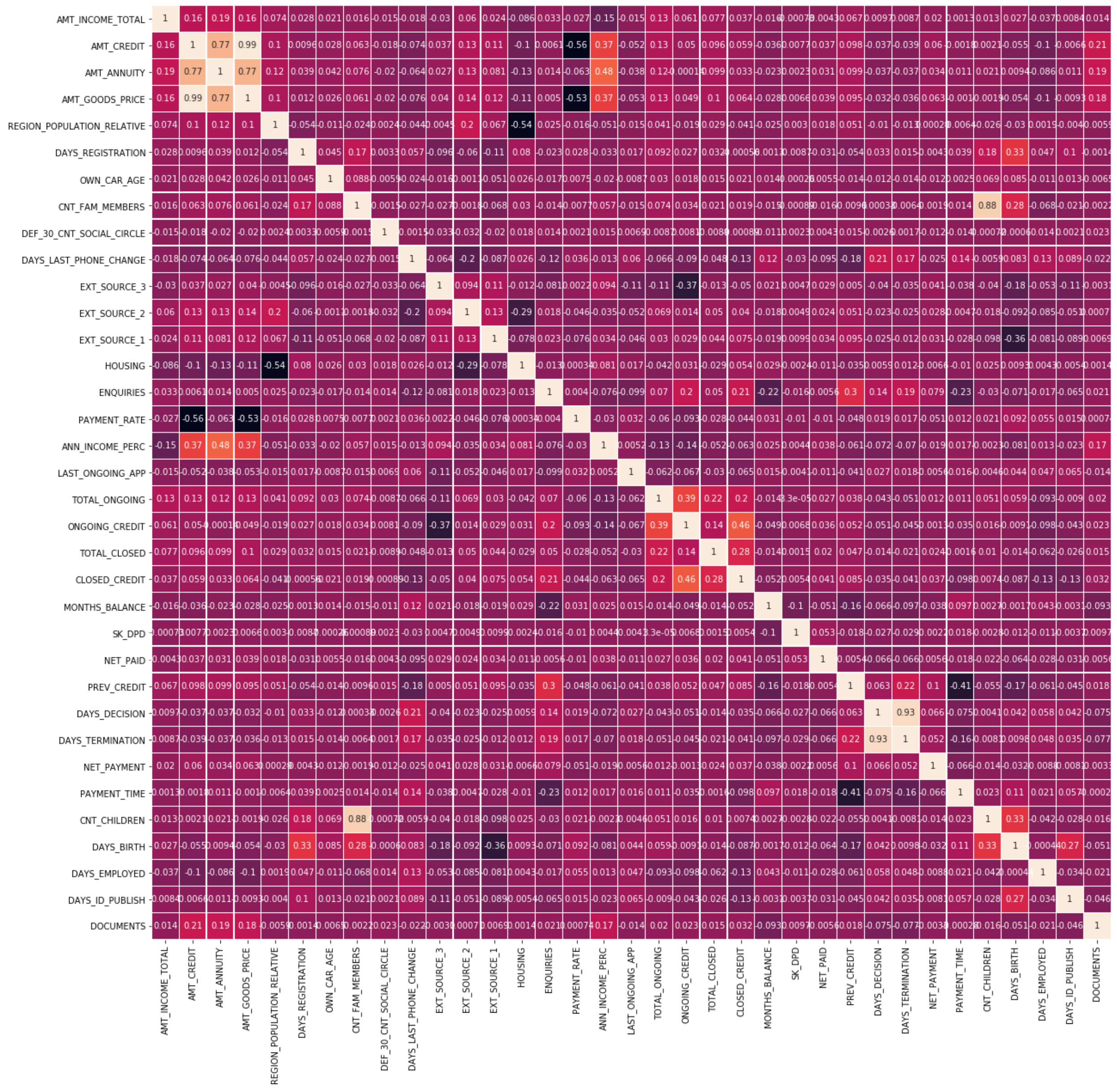

Figure 1. Seaborn heatmap of the correlations of the numerical columns in the training data. Only correlations with a magnitude greater than 0.35 were noted.

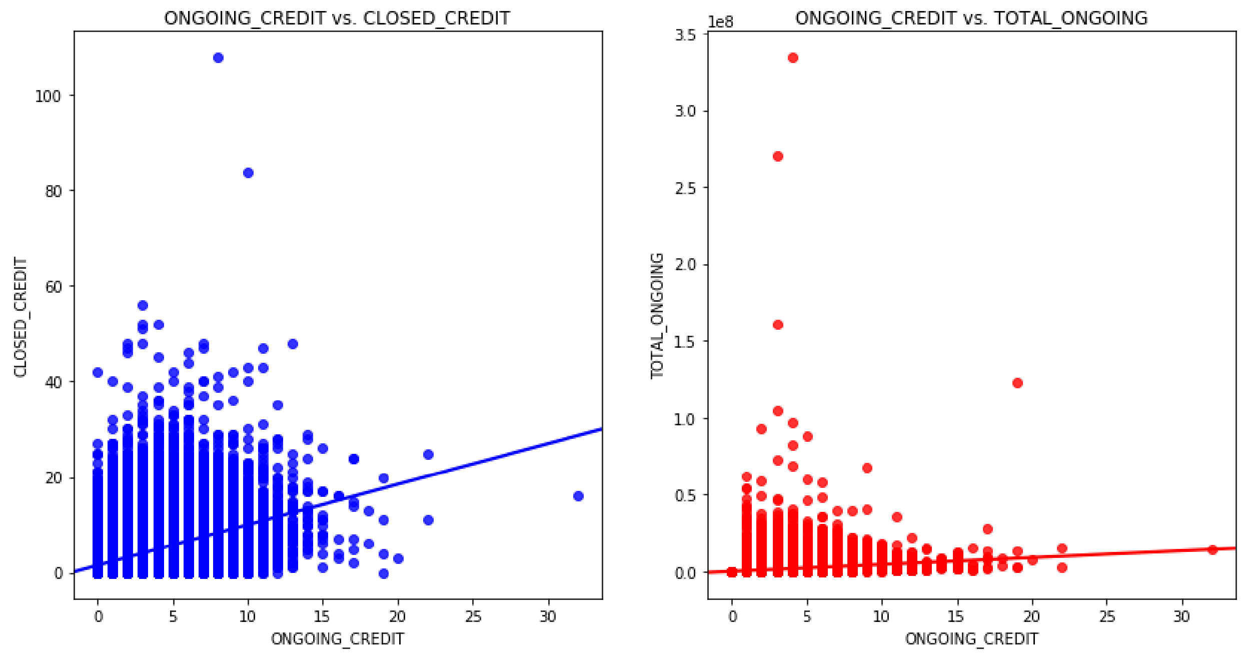

Figure 2. Seaborn regression plot of 2 positive correlations between numerical columns.

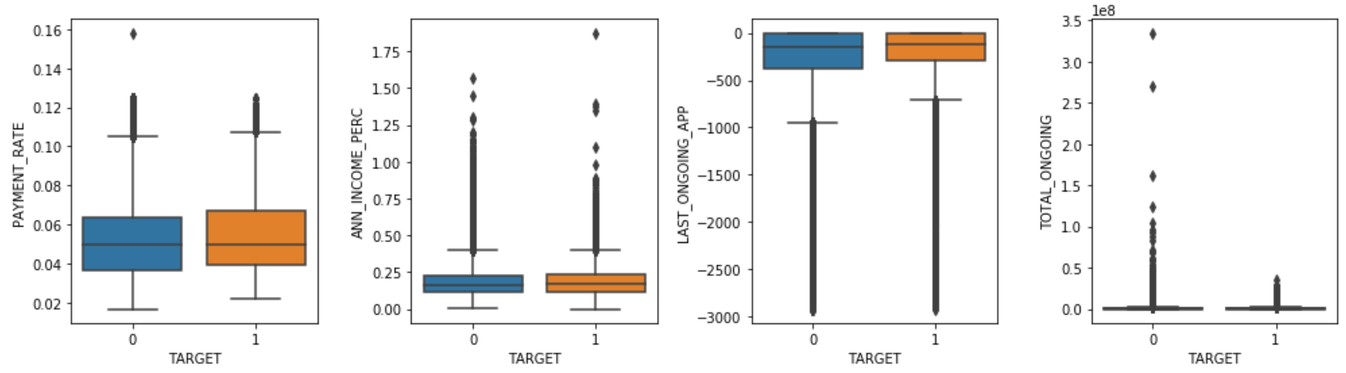

Figure 3. Boxplots of a couple numerical columns. The plots were obscure and difficult to draw insights from.

Inferential Statistics:

Using a p-value of 0.05, I attempted to prove that there were statistical differences in each variable between the target and non-target groups. I applied the inferential statistics to both the numerical and binary categorical data.

For the numerical columns, there was strong support from both the two-sided t-test and frequentist bootstrap approach that there were no statistically significant differences between the two groups for OWN_CAR_AGE (0.205), ENQUIRIES (0.1), SK_DPD (0.62), and DAYS_TERMINATION (0.69). Additionally, AMT_INCOME_TOTAL was also shown to have no statistical differences by the frequentist approach. However, a closer observation of the 95% confidence intervals showed that there was a small range for the total income for the non-target data, whereas the target data had a large spread that encompassed the smaller range (Figure 4).

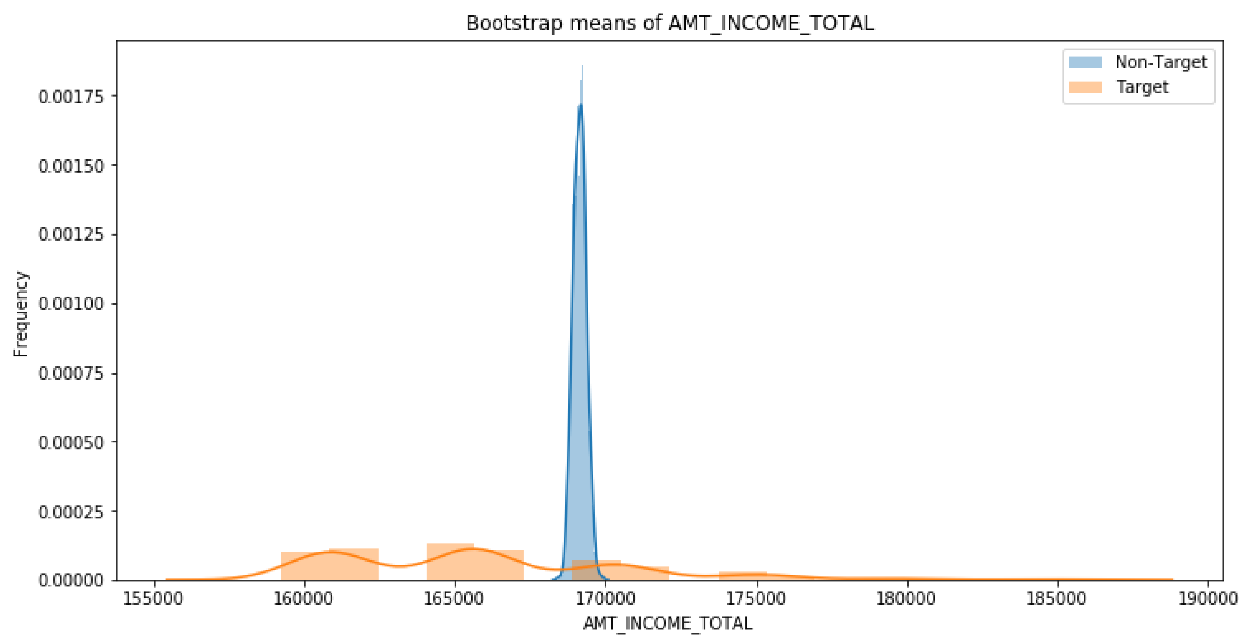

Figure 4. Distribution plot for bootstrap samples of Non-Target and Target groups in respect to AMT_INCOME_TOTAL. There is a normally distributed, single peak for the non-target group, whereas the target group is rightly skewed and multi-peaked.

Besides the 4 noted variables and AMT_INCOME_TOTAL, the remaining numerical variables all had non-overlapping 95% confidence intervals. To explore this more closely, I plotted the bootstrap means of the NET_PAYMENT (amount client pays relative to demanded monthly instalment payment) data as it had the most extreme non-overlaps (Figure 5).

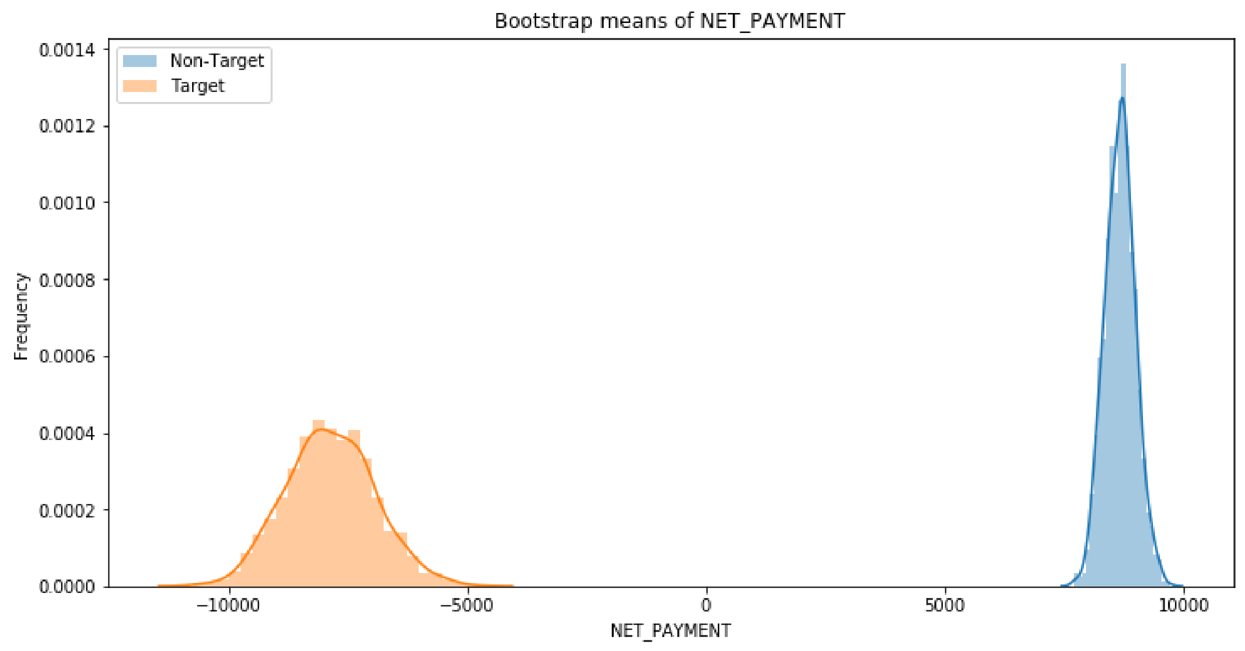

Figure 5. Distribution plot of the bootstrap samples of Non-Target and Target groups in respect to NET_PAYMENT. Both groups are normally distributed at their extremes.

Amongst the numerical variables, there were 3 variables that stood out as they had the strongest magnitude in correlation. EXT_SOURCE_[1, 2, and 3] all were negatively correlated with the target variable. According to the provided informational csv, these three variables represent a normalized score from external data sources. To better understand them, I chose to plot their distributions side-by-side (Figure 6). According to the visualizations, the target group had more lower scores than the non-target group.

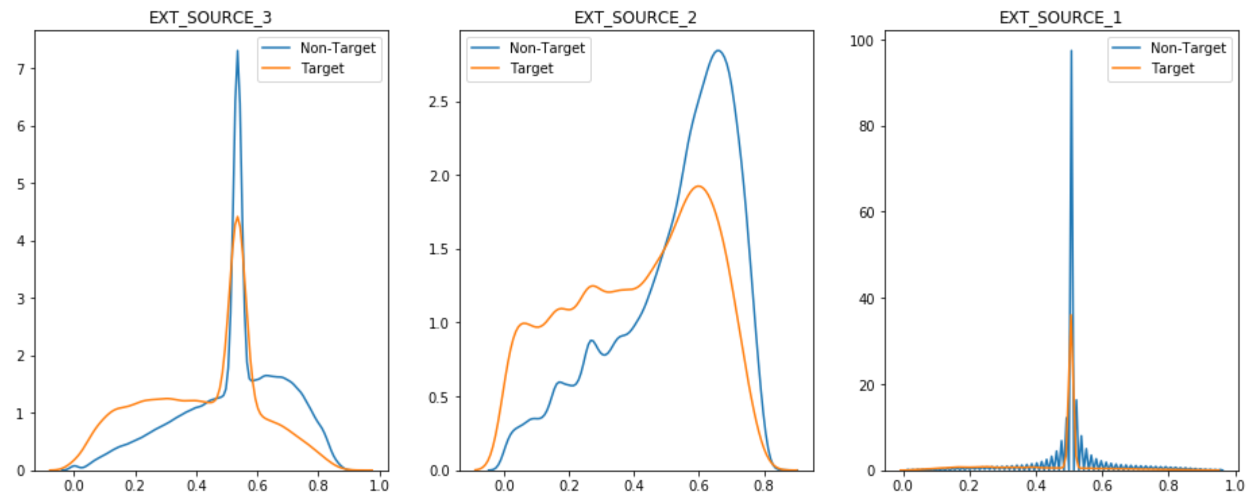

Figure 6. KDE Plots of the EXT_SOURCE variables. All datasets besides EXT_SOURCE 1 are noticeably left skewed. Additionally, the target group had more lower scores.

For the binary categorical columns, I first used LabelEncoder from sklearn to convert the text to integers. I also applied the two-sided t-test and frequentist bootstrap approach to check for statistically significant differences. Both tests support statistically significant differences for all binary categorical variables besides VALID_MOBILE (if a mobile number was provided and contactable). To take a closer look at the other variables, I plotted the bootstrap samples of NAME_TYPE_SUITE (Figure 7).

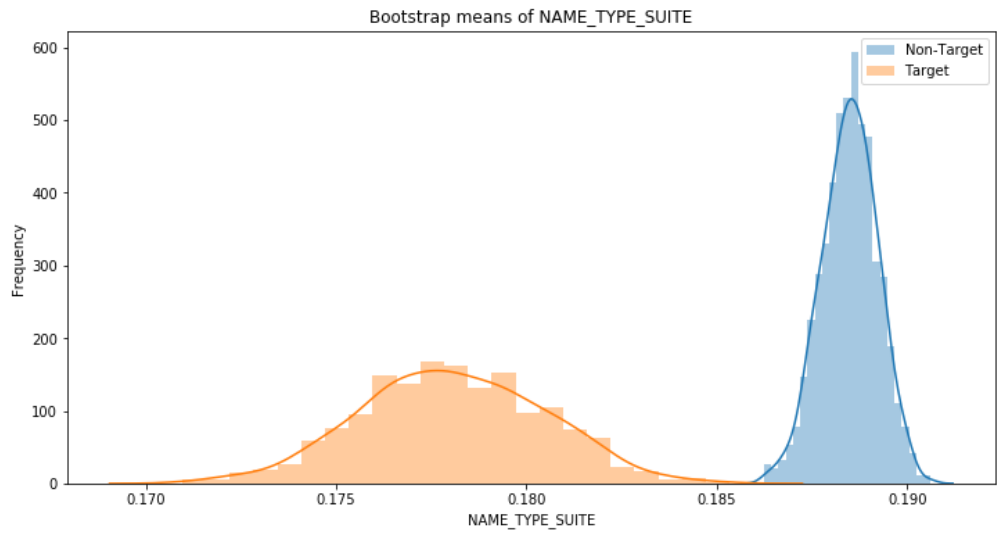

Figure 7. Distribution plot of bootstrap samples for Non-Target and Target groups in respect to NAME_TYPE_SUITE.

However, I acknowledge that these tests can be biased due to the unbalanced nature of the dataset (# non-targets >> # targets). For all the columns that failed to show statistical differences, I chose to temporarily keep them and train my model with and without them.

Building the Model:

With the removal of two numerical columns, by the time I reached the model, I was left with 43 columns. Additionally, I decided to remove the columns that were previously shown to have show no statistically significant differences between the two groups, leaving me with 23 numerical columns and 7 categorical columns (excluding the TARGET and SK_ID_CURR columns).

Preprocessing:

With the remainder of the data, I performed an 80/20 train test split with no stratify parameter. I utilized FeatureUnion and FunctionTransformer to create a pipeline step that could apply StandardScaler and OneHotEncoder to my numerical and categorical columns respectively. With this pipeline step, I could fit my data to multiple different machine learning models without having to complete the same preprocessing steps.

Before applying any models, I wanted to figure out the feature importance for the numerical columns. Using a Random Forest model, I fitted my numerical data and plotted the feature importance (Figure 8).

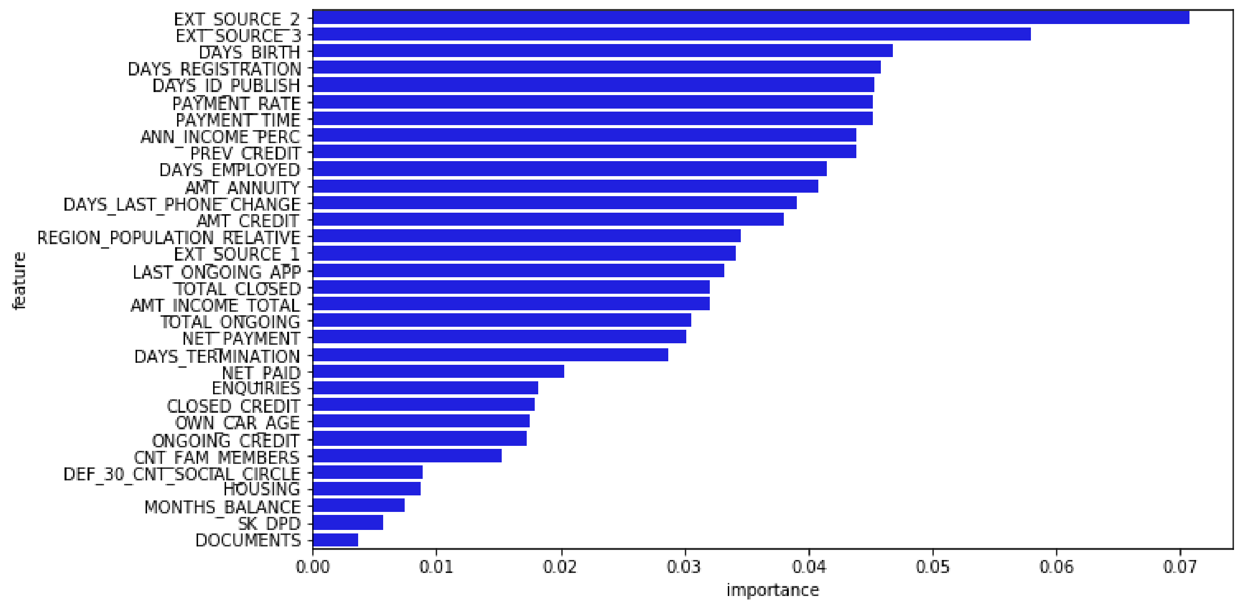

Figure 8. Horizontal barplot of feature importance for the numerical columns in the training data.

Algorithm and Variable Experimentation:

Thus far, I have attempted five different machine learning algorithms: Logistic Regression, Decision Trees, Random Forests, LinearSVC, and XGBClassifier . Of the five, Decision Tree had to lowest performance early on and was no longer tested on.

With the LogisticRegression model, it appears that the model overestimates the number of non-targets. (Figure 9). The single DecisionTreeClassifier model overestimates the number of targets (Figure 10). The RandomForestClassifer model overestimates the number of non-targets, but also displays an irregular multi-peaked distribution (Figure 11). The LinearSVC model's predictions are normally distributed, with the majority around the middle of the spectrum (Figure 12). Lastly, the XGBClassifier model also overestimated the number of non-targets (Figure 13).

Some variable variations I have attempted are with and without the previously tagged variables, with and without the inspired domain features and additional numerical variables (added due to score in feature importance analysis). My range of scores are from 0.524 to 0.761. Prior to any hyperparameter tuning, my max score was .693 with the LogisticRegression model. To this point, the tuned XGBClassifier model with the tagged variables removed, with the inspired domain features and additional numerical variables present, has shown the highest performance at 0.761.

Additionally, I did attempt to utilize AdaBoost, particularly with the LogisticRegression, but it caused the performance to decline. As a result, I discontinued its application in my other models (Figure 14).

Future Direction:

Provided time, I would focus on developing a more in-depth domain knowledge. My model only utilizes two domain features which were inspired by another Kaggle competitor. These two features caused my score to increase by 0.7, which is the largest increase thus far. The second largest was with hyper parameter tuning that increased by score by slightly less than 0.7, but it was also very time-consuming.

Next, I'd attempt to calculate the feature importance for all the variables (300+) prior to any manual feature selection I did. However, to do so, I would need more computational power to fit the RandomForest model.

Additionally, I would like to perform some additional data transformation to mitigate the imbalanced dataset (8% target, 92% non-target). Some approaches for this include adding bootstrap samples to the total dataset to increase the # of target variables.

Model Figures:

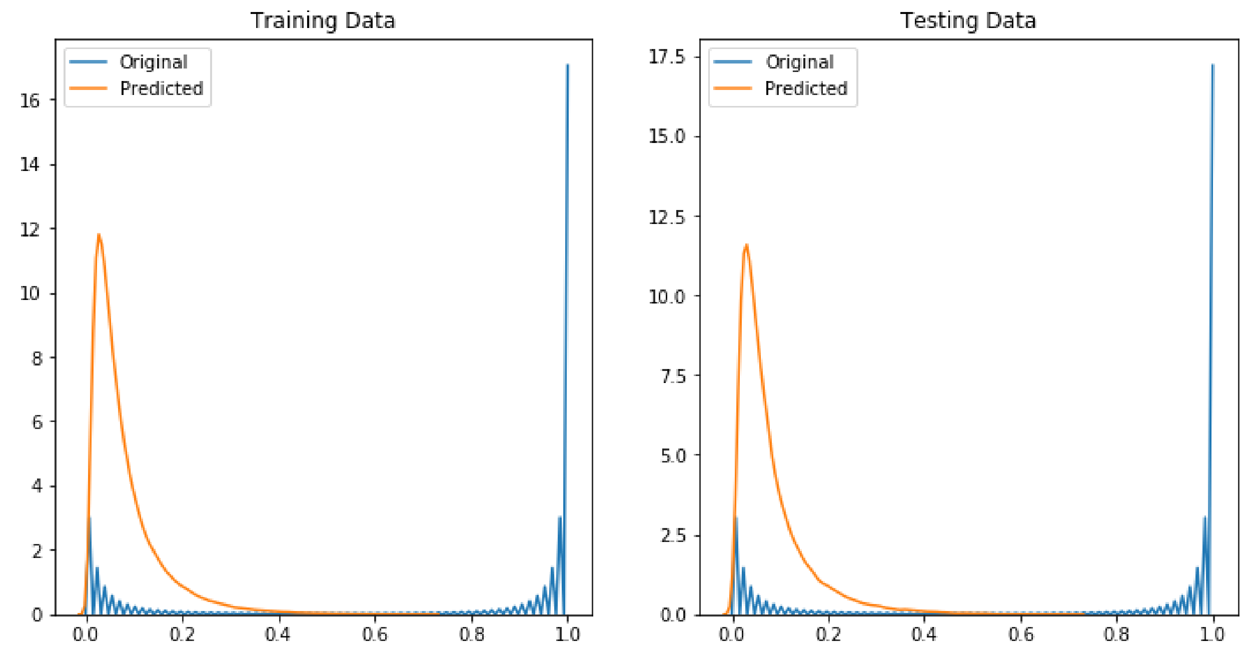

Figure 9. KDE plot of the training and test data with the LogisticRegression model.

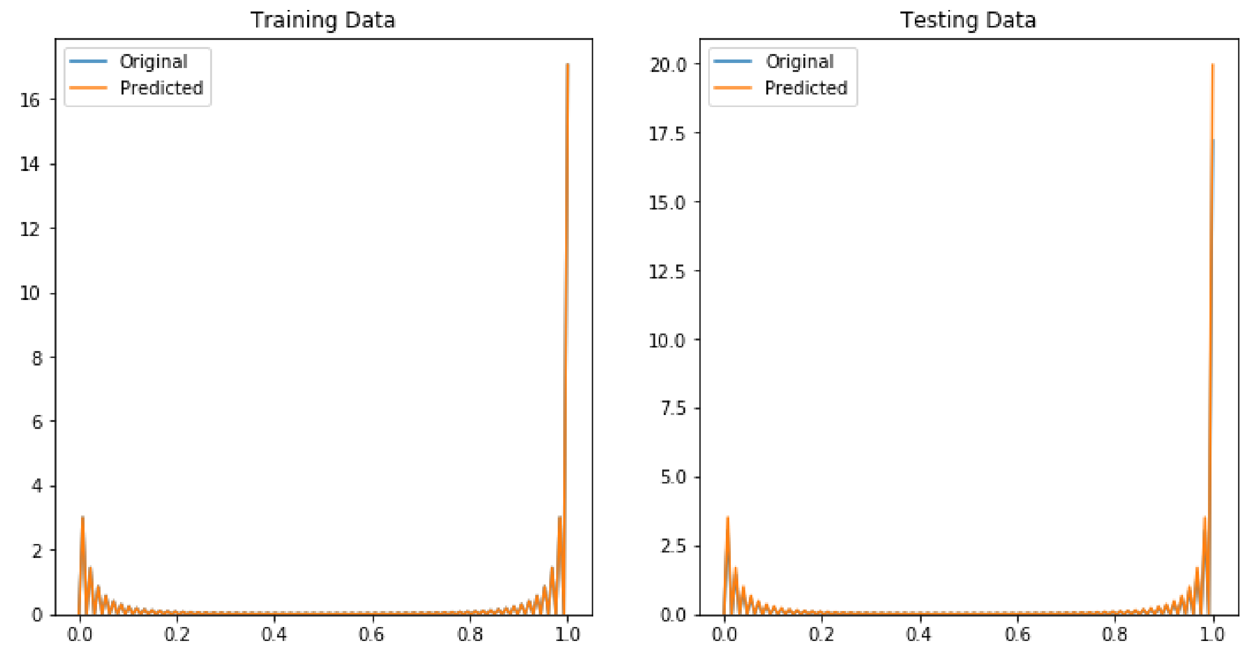

Figure 10. KDE plot of the training and test data with the DecisionTreeClassifier model.

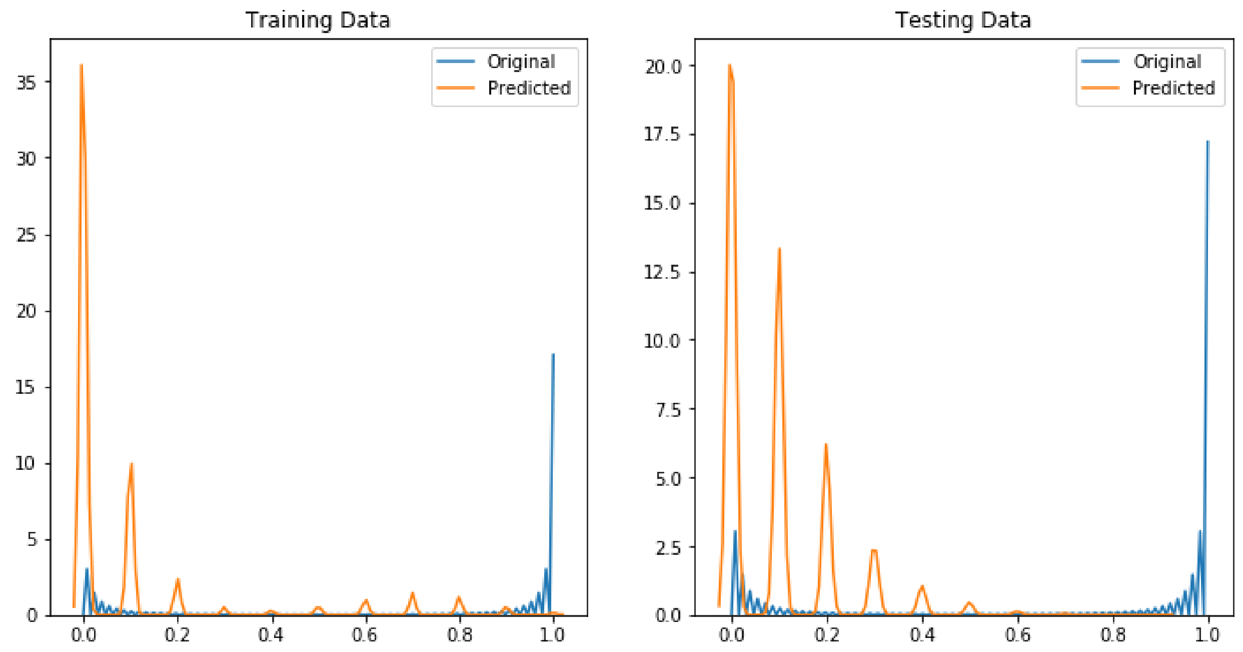

Figure 11. KDE plot of the training and test data with the RandomForestClassifier model.

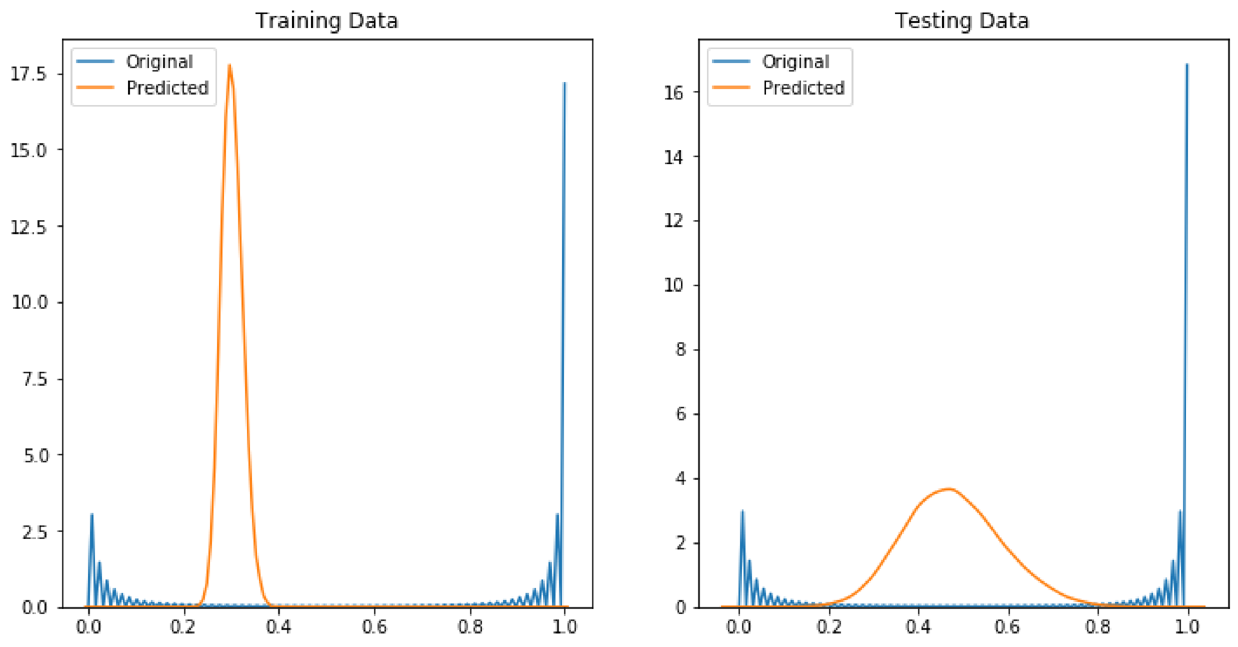

Figure 12. KDE plot of the training and test data with the LinearSVC model.

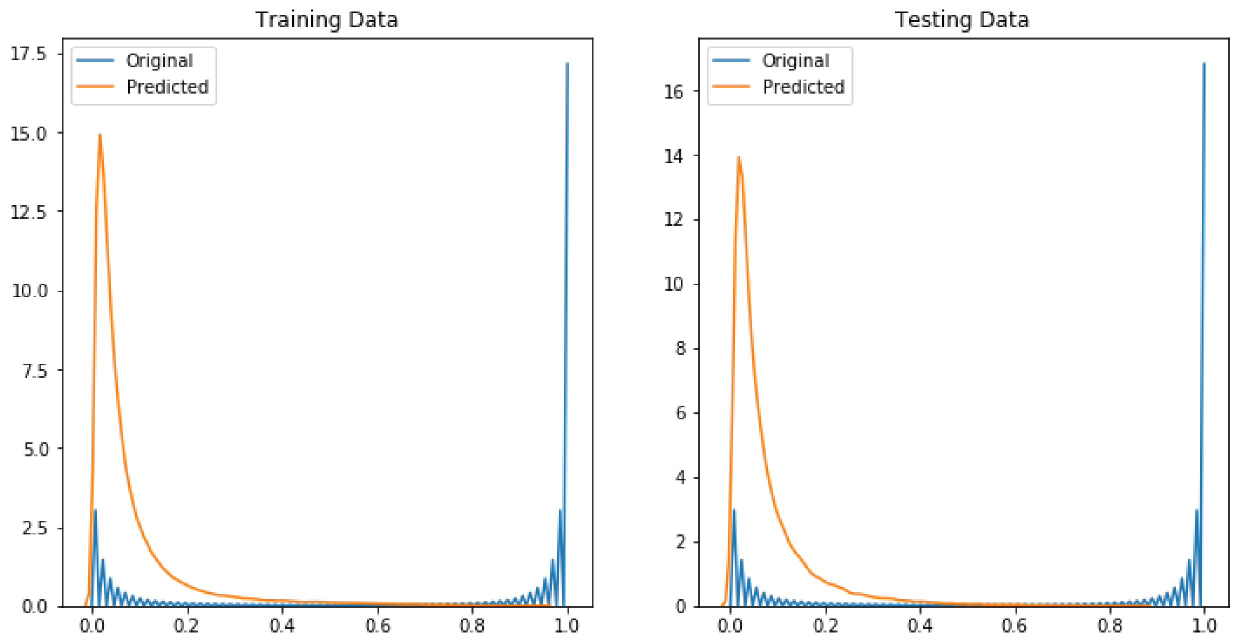

Figure 13. KDE plot of training and test data with XGBClassifier

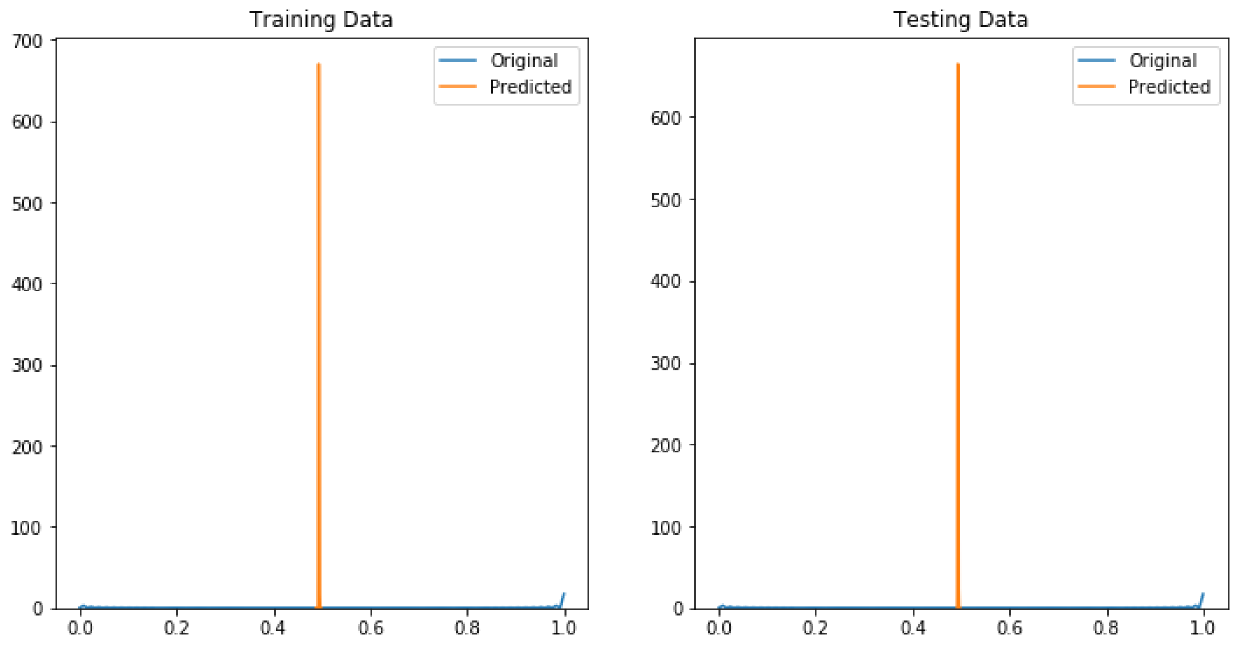

Figure 14. KDE plot of the training and test data with AdaBoost applied to LogisticRegression.