

Capstone #2: Final Report

[GitHub Project Link](#)

Introduction	2
Problem Statement	2
Data	2
Descriptions	2
Overview	4
Missing Values	4
Outlier Treatment	4
Feature Engineering	5
Combine Datasets	5
Exploratory Data Analysis (EDA)	6
Target Distribution	6
Children	6
Gender	7
Total Income	7
Finalizing the Data	8
Encode the Data	8
Balance the Data	9
Split the Data	9
Variable Selection	9
Information Value Analysis	9
Variance Inflation Factor (VIF)	9
Machine Learning	10
Preprocess the Data	10
Instantiate the Models	10
Hyperparameter Space	10
Train and Score Models	10
Model Metrics	11
Top Model Performers	12
Summary	13
Next Steps	13

Introduction

Problem Statement

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

[Home Credit](#) strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that the loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data

Descriptions

There are eight data files listed and described below. Notice that the application dataset has been previously split up into train and test data.

application_{train|test}.csv

- This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET). Static data for all applications. One row represents one loan in our data sample.

bureau.csv

- All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample). For every loan in our sample, there are as many rows as the number of credits the client had in Credit Bureau before the application date.

bureau_balance.csv

- Monthly balances of previous credits in Credit Bureau. This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has

(#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.

POS_CASH_balance.csv

- Monthly balance snapshots of previous POS (point of sale) and cash loans that the applicant had with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.

credit_card_balance.csv

- Monthly balance snapshots of previous credit cards that the applicant has with Home Credit. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.

previous_application.csv

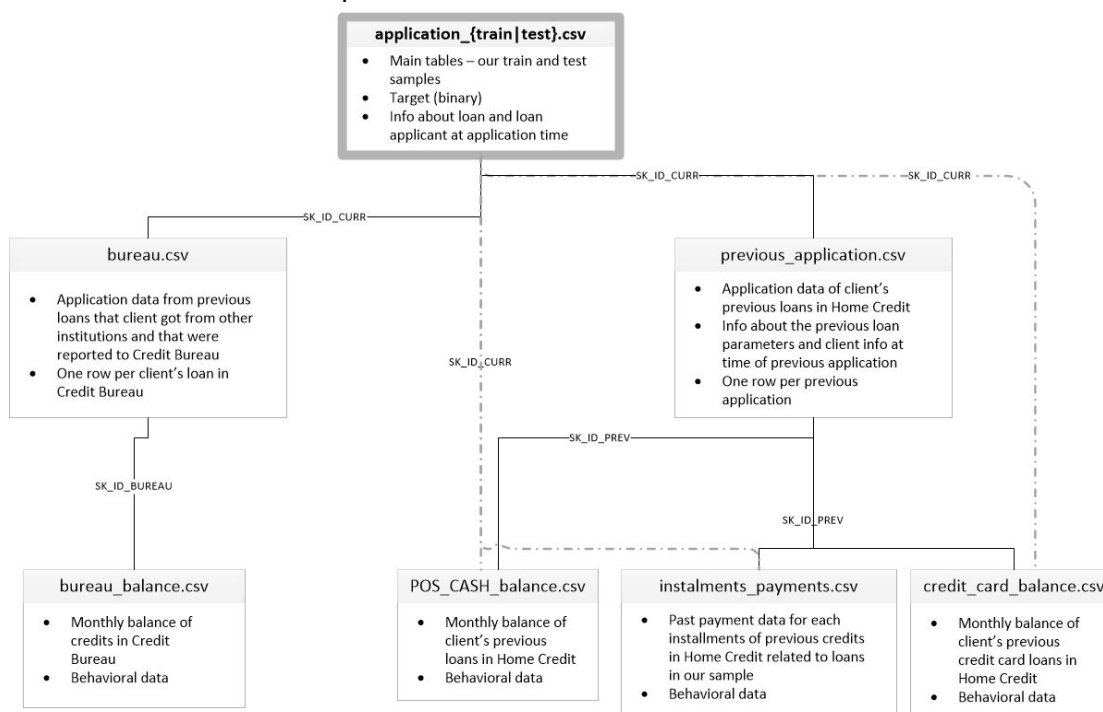
- All previous applications for Home Credit loans of clients who have loans in our sample. There is one row for each previous application related to loans in our data sample.

instalments_payments.csv

- Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample. There is a) one row for every payment that was made plus b) one row each for missed payment. One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

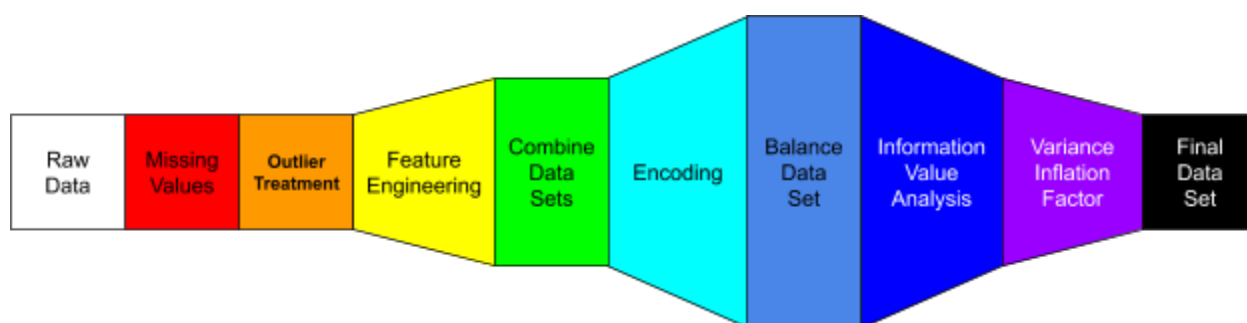
HomeCredit_columns_description.csv

- This file contains descriptions for the columns in the various data files.



Overview

Many steps are used in the process of wrangling, cleaning and preparing the data before modeling it. Feature Engineering and Encoding add new columns to the data while Information Value Analysis and Variance Inflation Factor reduce the number of columns. The below diagram is a graphical representation of number of columns of data throughout the project.



Missing Values

It is important to handle missing values in the dataset because when variables are selected through IV and VIF, there cannot be any missing values. For this reason, each type of value will be handled in a different manner.

For a first attempt, all rows that contain a NaN value were removed. If the number of rows removed was small relative to the total number of rows, this may be a good way to handle the missing values. But in this case, a lot of rows have missing values, so this method removes ~90% of the total rows. In this project, each column will be handled independently.

- **Categorical** - NaN replaced with 'unknown' string category
- **Numerical**
 - AVG - NaN replaced with mean value of column
 - MEDI - NaN replaced with median value of column
 - MODE - NaN replaced with mode value of column

Outlier Treatment

Outliers are defined as values that are greater than 3 standard deviations from the mean in either the positive or negative direction. This is applied per each column of the DataFrame and only to numerical column types. The outlier treatment shouldn't be applied to flag/boolean type columns; columns with values 0 and 1. This will dramatically skew the data in those columns.

Feature Engineering

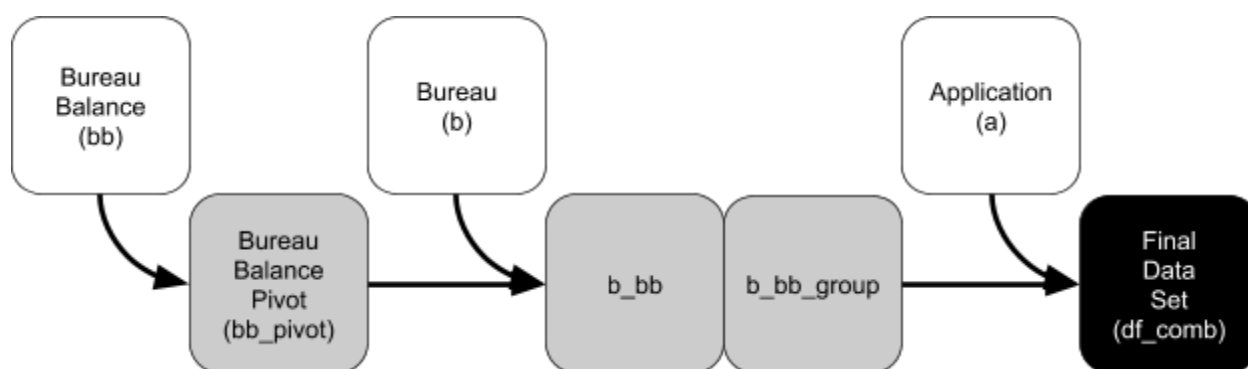
Feature Engineering is a technique used to expand upon the original data provided for the project. It may combine information from two or more columns by adding them together or counting how many times an item occurs, etc. This second level information may be helpful in creating a more accurate model down the line.

- $\text{FLAG_CHILDREN} = 1$ if $\text{CNT_Children} > 0$; else 0
- $\text{FLAG_INCOME} = 1$ if $\text{AMT_INCOME_TOTAL} > 0$; else 0
- $\text{FLAG_INCOME_6figs} = 1$ if $\text{AMT_INCOME_TOTAL} \geq 100000$; else 0
- $\text{CNT_DOC} = \text{SUM}(\text{FLAG_DOC})$

Combine Datasets

Three datasets were combined to create the final dataset. The first step was to pivot the bureau balance dataset. The new table has `SK_ID_BUREAU` as the index, the status months as the columns, and the statuses as the values in the table. This meant that each row has a unique `SK_ID_BUREAU`. The next step was to merge this table with the bureau dataset. When merged, this new table had duplicates of indexes.

Many `SK_ID_CURR`'s had multiple rows for each `SK_ID_BUREAU`. Before merging `b_bb` with the application dataset, the `SK_ID_CURR`'s have to be reduced to a single unique row. The `CREDIT_ACTIVE` column was transposed into multiple columns, counting each type of credit, along with the total. For the rest of the numerical columns, each column was averaged per `SK_ID_CURR`. At this point, the `b_bb_group` dataset has unique indexes for each `SK_ID_CURR`. The final step is to merge the application data with the `b_bb_group` dataset and create 'df_comb'.

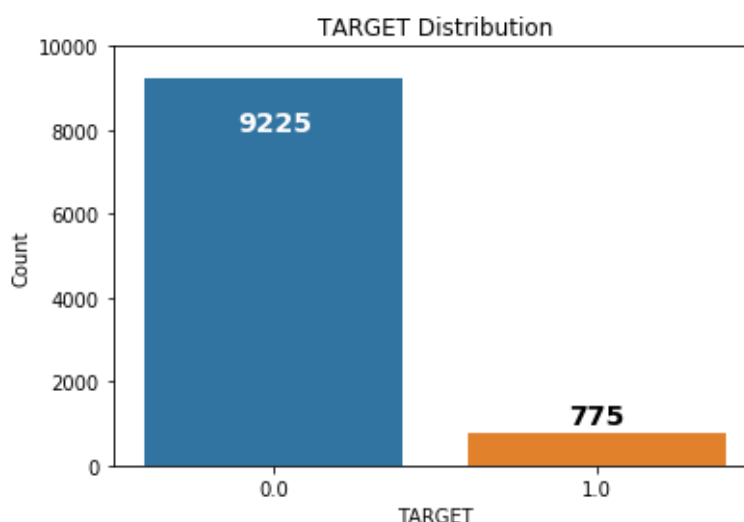


Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is the process of manually evaluating the data in the tables. This could be a distribution of a column or a combination of categorical data.

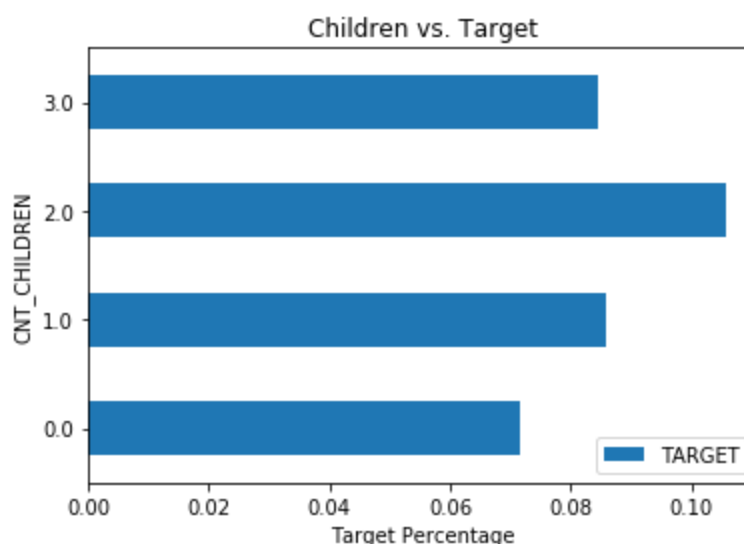
Target Distribution

The target data column is a 1 or 0: 1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample, 0 - all other cases. This shows that the dataset is unbalanced, roughly 92% to 8%.



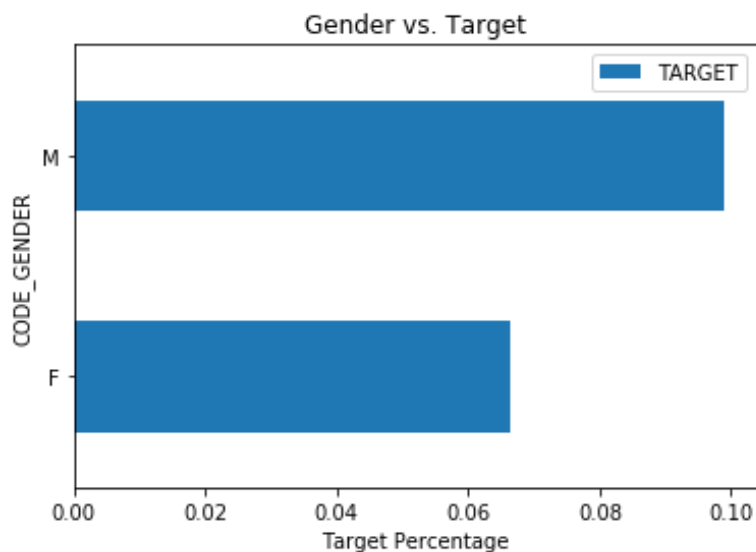
Children

This plot compares the number of children an applicant has with the percentage of 1s and 0s. Applicants without children are the least likely to have payment difficulties. Applicants with 2 children are most likely, ~50% more likely.



Gender

This plot compares the gender of an applicant to the percentage of 1s and 0s. Men are ~50% more likely to have payment issues when compared to women. This is quite a significant difference.



Total Income

This plot shows the distribution of total income per applicant per each target category, 1 or 0. Men with high income tend to have less payment issues. Also, men tend to have more income per category, 1 or 0.




Finalizing the Data


Encode the Data

Encoding is a process in which a single column is expanded into new columns to simplify the dataframe. There are two types of encoding: Label Encoding and One-Hot-Encoding.

- **LABEL ENCODING:** Assign each unique category in a categorical variable with an integer. No new columns are created. Use for columns with only 2 categories or one where categories have order to them: sizes - small, medium, large.
- **ONE-HOT ENCODING:** Creates a new column for each unique category in a categorical variable. Each observation receives a 1 in the column for its corresponding category and a 0 in all other new columns. Use for columns with 3+ categories and categorical data where order doesn't matter: colors - red, blue, green.

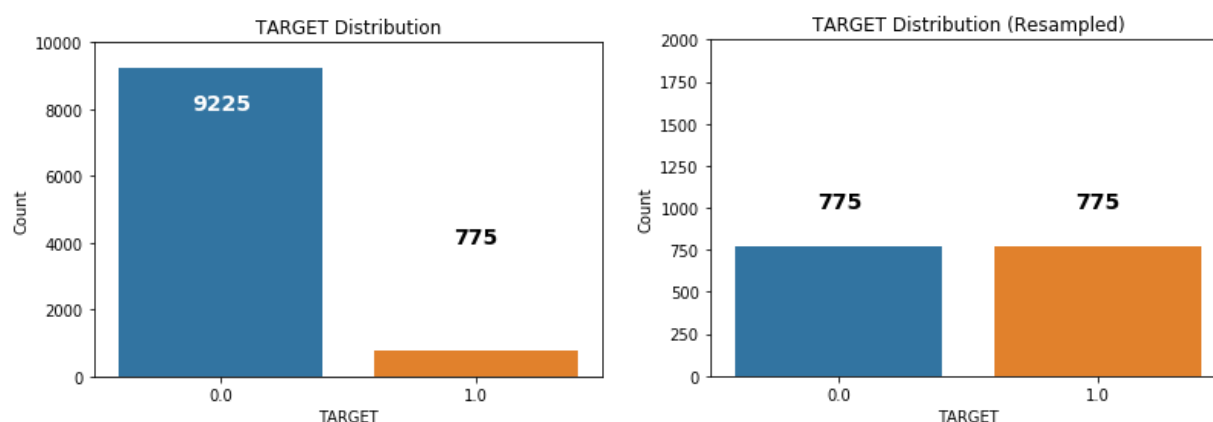
In this project, three categorical columns were Label Encoded: CODE_GENDER, NAME_CONTRACT_TYPE, and CREDIT_CURRENCY. For the rest of the categorical data, it was split off into a separate dataframe, one-hot-encoded, and then merged back into the original dataset. The final dataset has 366 columns of data.

Male		0
Female		1
Female		1
Male		0

Color		Blue	Green	Red
Blue		1	0	0
Green		0	1	0
Red		0	0	1
Blue		1	0	0

Balance the Data

Because the data is unbalanced, undersampling was used to balance the dataset. This balanced the data from 92-8 to 50-50 by reducing the rows with target values of 1 to the same as number of rows with target values of 0. The left chart shows the unbalanced dataset, the right chart shows the balanced dataset.



Split the Data

The data then needed to be split into train and test. This is done using the TRAIN_TEST column that was implemented earlier in the code. Then the data was split into X and Y, X being the independent columns and Y being the dependent column, TARGET.

Variable Selection

Information Value Analysis

Information Value Analysis (IV) is a data exploration technique that helps determine which data columns in a dataset have predictive power or influence on the value of a specified dependent variable. After this technique, 53 columns remain in the dataset.

Variance Inflation Factor (VIF)

Variance Inflation Factor (VIF) is the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone. It quantifies the severity of multicollinearity in an ordinary least squares regression analysis. After this technique is applied, 41 columns remain.

Machine Learning

Preprocess the Data

Two pipelines were created to handle the different types of data. One for handling numerical data and one for handling categorical data. Then these were combined into a Column Transformer.

Instantiate the Models

Each model was instantiated using the different libraries available. Some needed parameters set to specific values when created.

Hyperparameter Space

Each model has its unique parameters. The hyperparameter space is where a list of values are given for each parameter so that the model can be attempted with different parameter combinations. The more parameters attempted, the longer the process takes.

Train and Score Models

A function was created to do a multitude of things:

1. Create the CV Object
2. Fit the Model on the Training Data
3. Predict the Output with the Test Data
4. Score the Model with the Test Data

This information is stored in the Metrics table. Additional information is presented as well, along with a plot of the ROC Curve of True Positive Rate vs. False Positive Rate. Finally the models are run through the pipeline and train_score function. A switches dictionary was created to allow the user to turn specific models ON or OFF when testing new parts of the code.

Model Metrics

A metrics table was created to store the metrics of each of the different models. This table includes information like accuracy, precision, recall, area under the curve (auc), and time to train. Below is a table with all the results of the different models related scores utilizing the same dataset. The AUC is one of the best ways to compare models. Notice that the best model is the Logistic Regression, with an AUC score of 67, followed by XGBoost and the Random Forest.

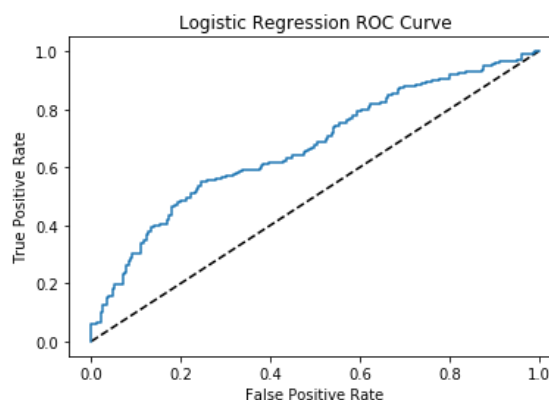
Metrics	Dummy Model	Logistic Regression	K Nearest Neighbors	Decision Tree	Random Forest	Naive Bayes	XGBoost
accuracy	49.200	61.500	55.500	58.300	59.100	54.800	60.600
precision	0.000	62.900	57.000	57.600	62.100	54.600	63.100
recall	0.000	58.900	50.000	67.400	50.000	64.800	54.200
auc	50.000	67.000	58.500	60.700	64.300	60.700	65.300
time to train	0.008	6.619	5.088	0.798	5.373	0.009	72.646

Top Model Performers

The top three models utilize certain parameter sets, as shown below:

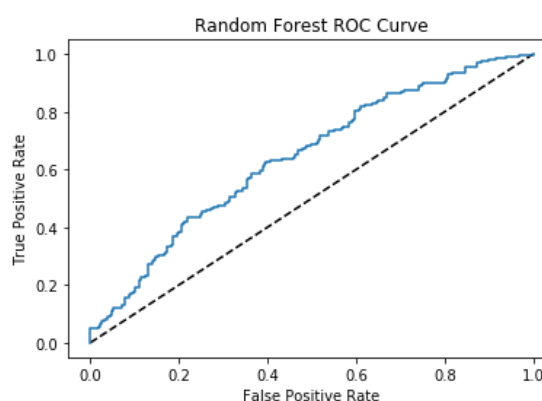
3.7.1) Logistic Regression Model

```
1 lr_cv.best_params_  
{'classifier__C': 1.0000000000000001e-05,  
 'classifier__penalty': 'l2',  
 'preprocessor__num__imputer__strategy': 'mean'}
```



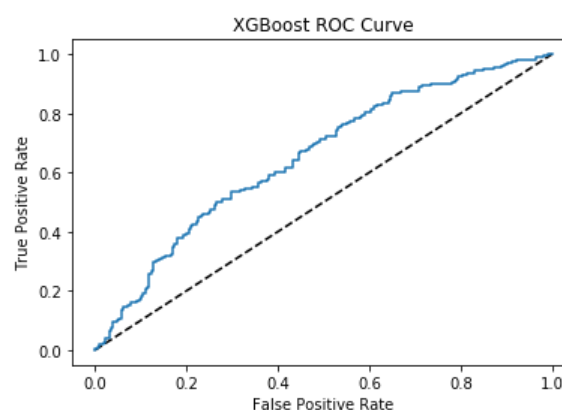
3.7.2) Random Forest Model

```
1 rf_cv.best_params_  
{'classifier__criterion': 'gini',  
 'classifier__max_depth': 20,  
 'classifier__max_features': 1,  
 'classifier__min_samples_leaf': 8,  
 'classifier__min_samples_split': 7,  
 'classifier__n_estimators': 100,  
 'preprocessor__num__imputer__strategy': 'median'}
```



3.7.3) XGBoost Model

```
1 xgb_cv.best_params_  
{'classifier__n_estimators': 568,  
 'classifier__max_depth': 803,  
 'classifier__learning_rate': 0.056}
```



Summary

There is a lot of financial data for each applicant. It can be used to better serve new applicants who are looking to expand their financial footprint in a responsible fashion. This alternative data can be taken advantage of to empower these new clients. The data sets were wrangled and combined to create a final dataset. Seven different models were tested on the final data set. Each one utilized a range of parameters to find the best version of each. According to the Area Under the Curve (AUC), the Logistic Regression had the highest score, 67 out of 100.

Next Steps

Additional Datasets

- In this project, three out of the seven datasets were utilized. This showed the value in combining multiple datasets into one before modeling the information. In a future project, the remaining datasets could be included in the final dataset to further increase the accuracy of the models.

Feature Engineering

- Another step that could be taken would be to increase the number of Feature Engineered columns. This could be done with further insight into the background of the raw data, how it was collected, and how it's applied currently in the field.

TPOT

- TPOT is a Python library that automates the entire Machine Learning pipeline and returns the best performing Machine Learning model. This library could be used to increase the scope and the speed of the ML process, and eventually could return a better final model.
-