Machine Learning Engineer Nanodegree
Capstone Project
Jonathan Ng
September 8th, 2018

I. Definition

Project Overview

Acquiring a loan can be difficult for people with little or no credit history and deceptive lenders in the industry often prey on and take advantage of this population. Giving loans to people who have a low likelihood of being able to pay it back sets those people up for failure and severely damages their economic status. In addition to giving loans to those who cannot pay them back, people who are able to pay back a loan can be rejected, resulting in a lose-lose situation for both parties.

Home Credit Group is an international consumer finance provider that focuses on responsible lending to people with little or no credit history. Home Credit Group aims to provide a successful and safe borrowing experience for this population by ensuring that clients able to repay the loan will receive one, and clients are given reasonable conditions of repayment that enable them to succeed. To better accomplish these goals, Home Credit Group has turned to Kaggle to answer the question of whether an applicant is capable of repaying a given loan based on application, demographic, and historical credit behavior data.

Lending and loan management is a huge industry today and it is essential for companies in this field to incorporate machine learning techniques into their decisions. Below is an article further detailing the context and incentive for research to this problem and a research paper about machine learning applications to this field.
https://www.techemergence.com/artificial-intelligence-applications-lending-loan-management/

Problem Statement

The problem at hand is a binary classification task. For a given loan application, the goal is to predict if the applicant will pay back the loan. A value of 0 corresponds to the applicant being able to repay the loan, and a value of 1 corresponds to the applicant not being able to repay the loan. Relevant features regarding the loan application, applicant demographic, and applicant credit history can be used to predict the binary outcome corresponding to the applicant's ability to pay off the loan. Since the labels are given in the training data, numerous supervised learning techniques can be applied to solve the problem.

However, accuracy is not a good indicator of performance. For this problem, most applicants do pay back the loan; as a result, an algorithm that predicts every applicant being capable of paying back the loan could result in a high accuracy score. For this project, the metric that will be used to measure the performance of the model is the ROC AUC scoring metric.

Evaluation Metrics

The metric that can be used to quantify the performance of the benchmark and solution model is the ROC AUC (Receiver Operating Characteristic Area Under the Curve) score. When doing binary classification, there can be 4 possible outcomes that can be formatted into the confusion

matrix. These are false positives, false negatives, true positives, and true negatives. The TPR (True Positive Rate) is the ratio of correctly classified positives to all true positive; it is also known as sensitivity or recall. The FPR (False Positive Rate) is the ratio of incorrectly classified negatives to all true negatives. The ROC AUC score combines the TPR and FPR into a single metric that scores will scrutinize models that predict a certain outcome due to a heavily one-sided distribution of target values; the result of such a model could obtain a high accuracy score, but the ROC AUC score would differ drastically and be close to 0.5, the minimum area under the ROC curve.

II. Analysis

Data Exploration

1. Application Training and Testing Sets

The application datasets contain 120 features per application and there are a total of 356,255 applications in the datasets. Many features represent similar attributes of the data and can be grouped. Below are the largest groups of data in this dataset.

- 47 columns represent normalized demographic information about where the applicant lives. Most of null values in the datasets are contained in these columns. Each column contains null values. The range is from 172863 - 248360.
    - 42 of these columns are numerical statistical information (mean, median, mode). There are 14 distinct demographic features with each one containing information about the mean, median, and mode to make up these 42 columns. Each distinct feature has the same number of null values across the three statistics and they occur for the same applicants. For example, the column COMMONAREA_AVG has the most null values in the dataset with 248360 null values and the other common area statistics COMMONAREA_MEDI and COMMONAREA_MODE will also have 248360 null values in located at the same rows.
    - The remaining 5 columns are categorical features.
- 36 columns are binary features that correspond to whether the applicant provided certain information on the application. There are no null values in these columns.
- 6 columns represent the number of enquiries to the credit bureau within different periods of time. There are 47568 null values in each column and they all occur for the same applications.
- 4 columns represent the count of observations of the applicant's social surroundings with observable 30, and 60 days past due and defaulted on 30 and 60 days past due. There are 1050 null values in each column and they all occur at the same applications.

There are a few abnormalities in this data. In the CODE_GENDER column, there are 3 unique gender types. Inspecting the value counts of this column shows that there are 4 applicants in the gender category "XNA". In the REGION_RATING_CLIENT_W_CITY column, the description says there are 3 distinct rating values, but the dataset contains 4. Below are descriptions of the rest of the columns.

Column	Description	% Null	Type	Num Unique	Sample
SK_ID_CURR	ID of loan in	0.000	int64	356255	100002
AMT_ANNUITY	Loan annuity	0.000	float64	14166	24700.5
AMT_CREDIT	Credit amount of the loan	0.000	float64	6480	406598
AMT_GOODS_PRICE	For consumer loans it is the price of the goods for which the loan is given	0.001	float64	1291	351000
AMT_INCOME_TOTAL	Income of the client	0.000	float64	2741	202500
CNT_CHILDREN	Number of children the client has	0.000	int64	16	0
CNT_FAM_MEMBERS	How many family members does client have	0.000	float64	18	1
CODE_GENDER	Gender of the client	0.000	object	3	M
DAYS_BIRTH	Client's age in days at the time of application	0.000	int64	17467	-9461
DAYS_EMPLOYED	How many days before the application the person started current employment	0.000	int64	12896	-637
DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan	0.000	int64	6224	-2120
DAYS_LAST_PHONE_CHANGE	How many days before application did client change phone	0.000	float64	3846	-1134
DAYS_REGISTRATION	How many days before the application did client change his registration	0.000	float64	15898	-3648
EXT_SOURCE_1	Normalized score from external data source	0.544	float64	134315	0.0830
EXT_SOURCE_2	Normalized score from external data source	0.002	float64	127157	0.2629
EXT_SOURCE_3	Normalized score from external data source	0.195	float64	814	0.1393
HOUR_APPR_PROCESS_START	Approximately at what hour did the client apply for the loan	0.000	int64	24	10
NAME_CONTRACT_TYPE	Identification if loan is cash or revolving	0.000	object	2	Cash loans
NAME_EDUCATION_TYPE	Level of highest education the client received	0.000	object	5	Secondary / secondary special
NAME_FAMILY_STATUS	Family status of the client	0.000	object	6	Single / not married
NAME_HOUSING_TYPE	What is the housing situation of the client (renting, living with parents, ...)	0.000	object	6	House / apartment
NAME_INCOME_TYPE	Clients income type (businessman, working, maternity leave,…)	0.000	object	8	Working
NAME_TYPE_SUITE	Who was accompanying client when he was applying for the loan	0.006	object	7	Unaccompanied
OCCUPATION_TYPE	What kind of occupation does the client have	0.314	object	18	Laborers
ORGANIZATION_TYPE	Type of organization where client works	0.000	object	58	Business Entity Type 3
OWN_CAR_AGE	Age of client's car	0.660	float64	63	NaN
REGION_POPULATION_RELATIVE	Normalized population of region where client lives	0.000	float64	82	0.018801
REGION_RATING_CLIENT	Rating of the region where client lives (1,2,3)	0.000	int64	3	2
REGION_RATING_CLIENT_W_CITY	Rating of the region where client lives with taking city into account (1,2,3)	0.000	int64	4	2
TARGET	Target	0.137	float64	2	1
WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply for the loan	0.000	object	7	WEDNESDAY

Columns that start with Days represent the number of days relative to the application. Columns with CNT represent counts and are integers. Columns that start with AMT represent amount in dollars. Columns that start with NAME are categorical.

In total there are X number of numerical features and Y number of categorical features. The dataset includes many null values. 67 out of 120 columns contain null values. And in total there are 10,556,884 null values.

2. Bureau Dataset

The Bureau dataset has 1,716,428 rows and 17 columns. There are 817,395 unique current application ids and 305811 number of unique current application ids.

Column	Description	% Null	Type	Num Unique	Sample
SK_ID_CURR	ID of loan in	0.000	int64	305811	215354
SK_BUREAU_ID	Recoded ID of previous Credit Bureau	0.000	int64	1716428	5714463
CREDIT_ACTIVE	Status of the Credit Bureau (CB) reported credits	0.000	object	4	Active
CREDIT_CURRENCY	Recoded currency of the Credit Bureau credit	0.000	object	4	Currency 1
DAYS_CREDIT	How many days before current application did client apply for Credit Bureau credit	0.000	int64	2923	-208
CREDIT_DAY_OVERDUE	Number of days past due on CB credit at the time of application for related loan in our sample	0.000	int64	942	0
DAYS_CREDIT_ENDDATE	Remaining duration of CB credit (in days) at the time of application in Home Credit	0.061	float64	14096	1075
DAYS_ENDDATE_FACT	Days since CB credit ended at the time of application in Home Credit (only for closed credit)	0.369	float64	2917	NaN
AMT_CREDIT_MAX_OVERDUE	Maximal amount overdue on the Credit Bureau credit so far (at application date of loan in our sample)	0.655	float64	68251	NaN
CNT_CREDIT_PROLONG	How many times was the Credit Bureau credit prolonged	0.000	int64	10	0
AMT_CREDIT_SUM	Current credit amount for the Credit Bureau credit	0.000	float64	236708	225000
AMT_CREDIT_SUM_DEBT	Current debt on Credit Bureau credit	0.150	float64	226537	171342
AMT_CREDIT_SUM_LIMIT	Current credit limit of credit card reported in Credit Bureau	0.345	float64	51726	NaN
AMT_CREDIT_SUM_OVERDUE	Current amount overdue on Credit Bureau credit	0.000	float64	1616	0
CREDIT_TYPE	Type of Credit Bureau credit (Car, cash,...)	0.000	object	15	Credit card
DAYS_CREDIT_UPDATE	How many days before loan application did last information about the Credit Bureau credit come	0.000	int64	2982	-20
AMT_ANNUITY	Annuity of the Credit Bureau credit	0.715	float64	40321	NaN

3. Bureau Balance

The Bureau Balance dataset has 27,299,925 rows and 3 columns. There are 817,395 number of unique Credit Bureau ids. This data shows the status of a given recorded ID of the Credit Bureau credit as the months progress. There are 8 unique categories of statuses.

Column	Description	% Null	Type	Num Unique	Sample
SK_BUREAU_ID	Recoded ID of Credit Bureau credit	0.0	int64	817395	5715448
MONTHS_BALANCE	Month of balance relative to application date	0.0	int64	97	0
STATUS	Status of Credit Bureau loan during the month	0.0	object	8	C

4. Previous Applications

The Previous Applications dataset has 1,670,214 rows and 37 columns. There are 338,857 unique current application ids.

Column	Description	% Null	Type	Num Unique	Sample
SK_ID_PREV	ID of previous application	0.000	int64	1670214	2802425
SK_ID_CURR	ID of current loan application	0.000	int64	338857	108129
NAME_CONTRACT_TYPE	Contract product type	0.000	object	4	Cash loans
AMT_ANNUITY	Annuity of previous application	0.223	float64	357959	25188.6
AMT_APPLICATION	For how much credit did client ask for	0.000	float64	93885	607500
AMT_CREDIT	Final credit amount on the previous application	0.000	float64	86803	679671
AMT_DOWN_PAYMENT	Down payment on the previous application	0.536	float64	29278	NaN
AMT_GOODS_PRICE	Goods price of good that client asked for	0.231	float64	93885	607500
WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply	0.000	object	7	THURSDAY
HOUR_APPR_PROCESS_START	Approximately at what day hour did the client apply	0.000	int64	24	11
FLAG_LAST_APPL_PER_CONTRACT	Flag if it was last application for the previous contract.	0.000	object	2	Y
NFLAG_LAST_APPL_IN_DAY	Application was the last per day of the client.	0.000	int64	2	1
RATE_DOWN_PAYMENT	Down payment rate normalized on previous credit	0.536	float64	207033	NaN
RATE_INTEREST_PRIMARY	Interest rate normalized on previous credit	0.996	float64	148	NaN
RATE_INTEREST_PRIVILEGED	Interest rate normalized on previous credit	0.996	float64	25	NaN
NAME_CASH_LOAN_PURPOSE	Purpose of the cash loan	0.000	object	25	XNA
NAME_CONTRACT_STATUS	Contract status of previous application	0.000	object	4	Approved
DAYS_DECISION	Relative to current application when was the decision about previous application made	0.000	int64	2922	-164
NAME_PAYMENT_TYPE	Payment method that client chose to pay for the previous application	0.000	object	4	XNA
CODE_REJECT_REASON	Why was the previous application rejected	0.000	object	9	XAP
NAME_TYPE_SUITE	Who accompanied client when applying for the previous application	0.491	object	7	Unaccompanied
NAME_CLIENT_TYPE	Was the client old or new	0.000	object	4	Repeater
NAME_GOODS_CATEGORY	What kind of goods did the client apply for	0.000	object	28	XNA
NAME_PORTFOLIO	Was the previous application for CASH, POS, CAR, …	0.000	object	5	Cash
NAME_PRODUCT_TYPE	Was the previous application x-sell o walk-in	0.000	object	3	X-sell
CHANNEL_TYPE	Through which channel we acquired the client	0.000	object	8	Contact center
SELLERPLACE_AREA	Selling area of seller place of the previous application	0.000	int64	2097	-1
NAME_SELLER_INDUSTRY	The industry of the seller	0.000	object	11	XNA
CNT_PAYMENT	Term of previous credit at application	0.223	float64	49	36
NAME_YIELD_GROUP	Grouped interest rate into small medium and high	0.000	object	5	Low_action
PRODUCT_COMBINATION	Detailed product combination	0.000	object	17	Cash X-sell: low
DAYS_FIRST_DRAWING	When was the first disbursement	0.403	float64	2838	365243
DAYS_FIRST_DUE	When was the first due supposed to be	0.403	float64	2892	-134
DAYS_LAST_DUE_1ST_VERSION	When was the first due of the previous application	0.403	float64	4605	916

DAYS_LAST_DUE	When was the last due date	0.403	float64	2873	365243
DAYS_TERMINATION	When was the expected termination	0.403	float64	2830	365243
NFLAG_INSURED_ON_APPROVAL	Did the client requested insurance	0.403	float64	2	1

5. Installment Payments

The Installment Payments dataset has 13,605,401 rows and 8 columns. There are 997,752 unique previous application ids and 339,587 unique current application ids.

Column	Description	% Null	Type	Num Unique	Sample
SK_ID_PREV	ID of previous credit	0.0	int64	997752	1330831.000
SK_ID_CURR	ID of loan	0.0	int64	339587	151639.000
NUM_INSTALMENT_VERSION	Version of installment calendar (0 is for credit card) of previous credit.	0.0	float64	65	0.000
NUM_INSTALMENT_NUMBER	On which installment we observe payment	0.0	int64	277	34.000
DAYS_INSTALMENT	When the installment of previous credit was supposed to be paid	0.0	float64	2922	-2156.000
DAYS_ENTRY_PAYMENT	When was the installments of previous credit paid actually	0.0	float64	3039	-2156.000
AMT_INSTALMENT	The prescribed installment amount of previous credit on this installment	0.0	float64	902539	1716.525
AMT_PAYMENT	What the client actually paid on previous credit on this installment	0.0	float64	944235	1716.525

6. Credit Card Balance

The Credit Card Balance dataset has 3,840,312 rows and 23 columns. There are 104,307 unique previous application ids and 103,558 unique current application ids. A key aspect of this dataset is that the ratio of unique values in SK_ID_PREV and SK_ID_CURR is close to 1.

Column	Description	% Null	Type	Num Unique	Sample
SK_ID_PREV	ID of previous credit	0.000	int64	104307	2582071
SK_ID_CURR	ID of loan in our sample	0.000	int64	103558	363914
MONTHS_BALANCE	Month of balance relative to application date	0.000	int64	96	-1
AMT_BALANCE	Balance during the month	0.000	float64	1347904	63975.6
AMT_CREDIT_LIMIT_ACTUAL	Credit card limit during the month of the previous credit	0.000	int64	181	45000
AMT_DRAWINGS_ATM_CURRENT	Amount drawing at ATM during the month	0.195	float64	2267	2250
AMT_DRAWINGS_CURRENT	Amount drawing during the month	0.000	float64	187005	2250
AMT_DRAWINGS_OTHER_CURRENT	Amount of other drawings during the month	0.195	float64	1832	0
AMT_DRAWINGS_POS_CURRENT	Amount drawing or buying goods during the month	0.195	float64	168748	0
AMT_INST_MIN_REGULARITY	Minimal installment for this month	0.079	float64	312266	2250
AMT_PAYMENT_CURRENT	How much did the client pay during the month	0.200	float64	163209	2250
AMT_PAYMENT_TOTAL_CURRENT	How much did the client pay during the month in total	0.000	float64	182957	2250
AMT_RECEIVABLE_PRINCIPAL	Amount receivable for principal on the previous credit	0.000	float64	1195839	60175.1
AMT_RECIVABLE	Amount receivable on the previous credit	0.000	float64	1338878	64875.6
AMT_TOTAL_RECEIVABLE	Total amount receivable on the previous credit	0.000	float64	1339008	64875.6
CNT_DRAWINGS_ATM_CURRENT	Number of drawings at ATM during this month	0.195	float64	44	1
CNT_DRAWINGS_CURRENT	Number of drawings during this month	0.000	int64	129	1
CNT_DRAWINGS_OTHER_CURRENT	Number of other drawings during this month	0.195	float64	11	0
CNT_DRAWINGS_POS_CURRENT	Number of drawings for goods during this month	0.195	float64	133	0
CNT_INSTALMENT_MATURE_CUM	Number of paid installments on the	0.079	float64	121	69

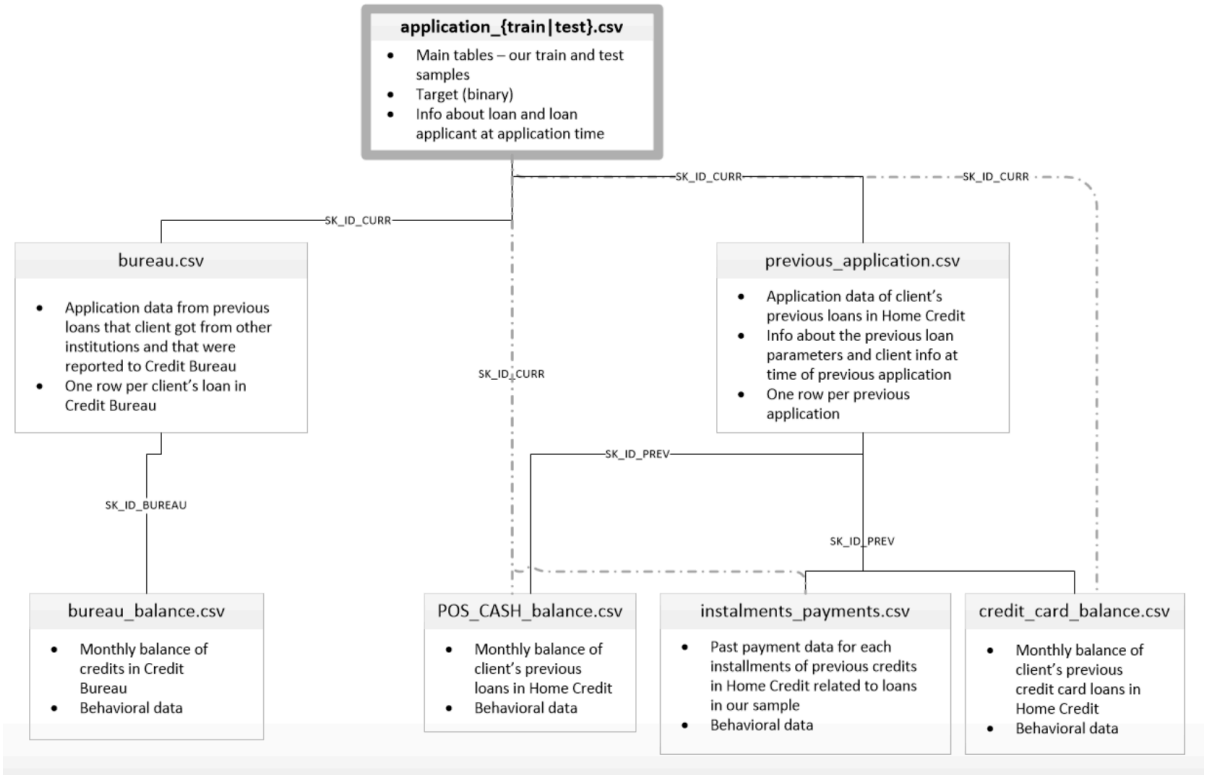
previous credit					
NAME_CONTRACT_STATUS	Contract status on the previous credit	0.000	object	7	Active
SK_DPD	DPD (Days past due) during the month	0.000	int64	917	0
SK_DPD_DEF	DPD (Days past due) during the month with tolerance (debts with low loan amounts are ignored)	0.000	int64	378	0

7. Point of Sales Cash Balance

The Point of Sales Cash Balance dataset has 10,001,358 rows and 8 columns. There are 936,325 unique previous application ids and 337,252 unique current application ids. It is important to note that the columns CNT_INSTALMENT and CNT_INSTALMENT_FUTURE are not related and present different types of information.

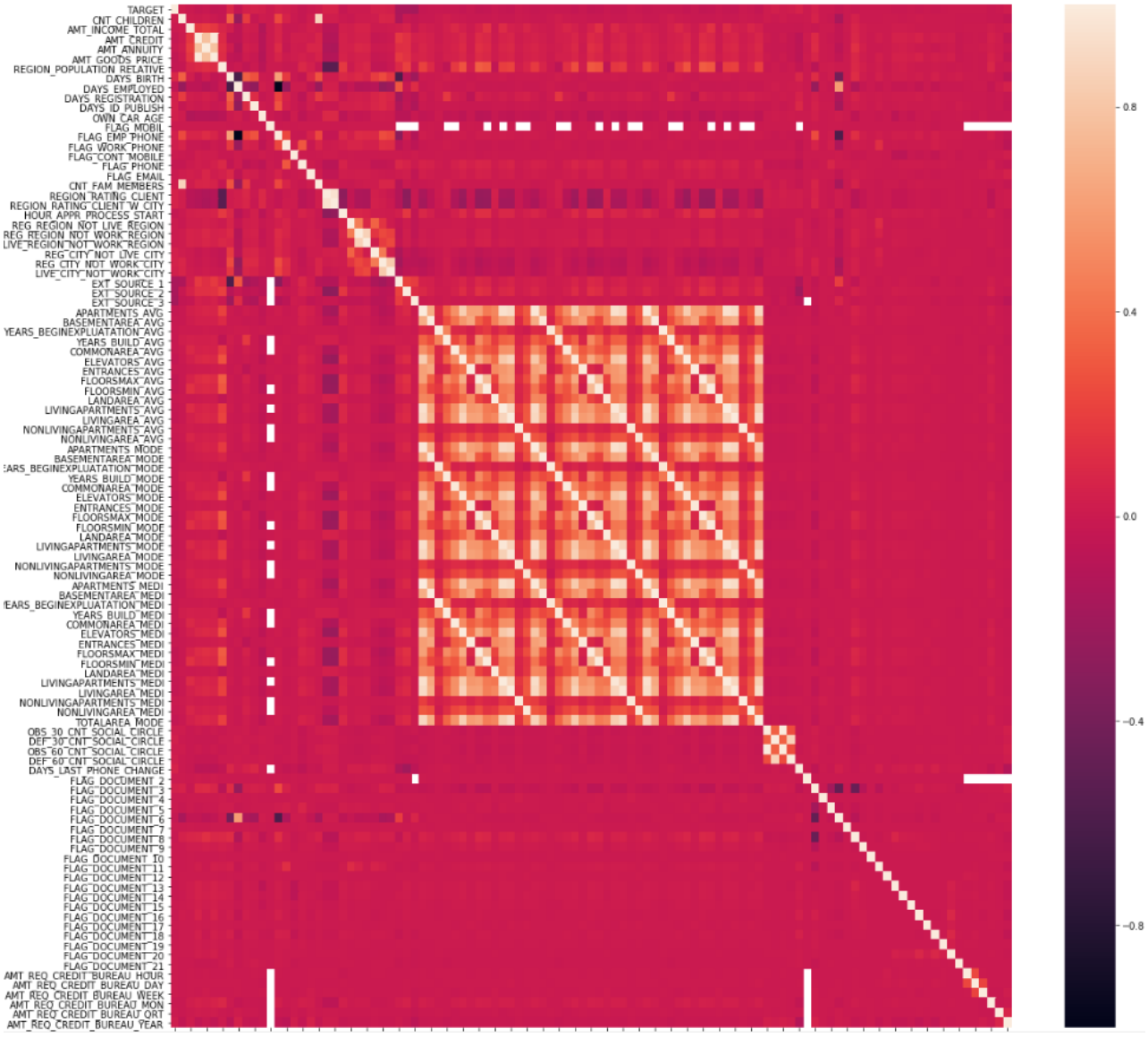
Column	Description	% Null	Type	Num Unique	Sample
SK_ID_PREV	ID of previous credit	0.000	int64	936325	1715348
SK_ID_CURR	ID of loan	0.000	int64	337252	367990
MONTHS_BALANCE	Month of balance relative to application date	0.000	int64	96	-33
CNT_INSTALMENT	Term of previous credit (can change over time)	0.003	float64	73	36
CNT_INSTALMENT_FUTURE	Installments left to pay on the previous credit	0.003	float64	79	35
NAME_CONTRACT_STATUS	Contract status during the month	0.000	object	9	Active
SK_DPD	DPD (days past due) during the month of previous credit	0.000	int64	3400	0
SK_DPD_DEF	DPD during the month with tolerance (debts with low loan amounts are ignored) of the previous credit	0.000	int64	2307	0

The interactions between all tables can be better understood and visualized in the tree below. Each dataset containing balances or payments has multiple rows for a unique id in either Credit Bureau loans or previous applications and every unique application id can have multiple rows in the Credit Bureau loans or previous applications as well.



Exploratory Visualization

From the applications training data, we can see which features correlate the most with each other as well as the target outcome. Shown below is the correlation heatmap of the applications training data.



An interesting pattern can be observed in this correlation heatmap. The demographic information about each applicant's housing/location statistics shows that the mean, median, and mode of these features highly correlate with one another.

The three features that correlate most with the target are the three EXT_SOURCE features. This makes sense as these features represent a score or rating of the applicant from an external source.

Algorithms and Techniques

The algorithm chosen to solve this problem is LightGBM. However, certain algorithms and techniques were used in the process of data preparation. The applications data is the only dataset that contains one rows for each applicant. In the other 6 datasets, multiple rows can correspond to one individual applicant. For example, the bureau_balance.csv data contains how the status of for a given credit loan from the Credit Bureau changes from month to month. The method to handle this type of data is to summarize each unique id into a row. For time series categorical features, there is an extra step of one hot encoding these features. In some cases a technique to reduce the number of categories of a feature is applied so that one hot encoding does not create extra columns that are negligible or sparse. After one hot encoding the status in the bureau balance dataset, the resulting data frame is shown below.

Three techniques are applied to the time series data in these datasets. The first technique is to aggregate certain statistics of features for each unique id. For all one hot encoded categorical features, the main statistic chosen to aggregate is the mean. After aggregating the mean in the bureau balance dataset, the new data frame consists of one row for each credit loan. The new status columns each represent the percentage of months that the status of the loan was the status indicated in the given column. For numerical features, other statistics such as the min, max, and sum are used depending on the feature and what is represents. From the month's column, we can aggregate the count which corresponds to the number of months or duration of each credit loan with the Credit Bureau.

The second technique used summarizing time series data utilizes Linear Regression. The idea is to extract the slope of a column with respect to a time variable for each unique id. In the bureau balance dataset, the slope coefficient is calculated for each status category by fitting a linear regression model to each status column with respect to the months balance column. This extracts more information about the credit loan. The idea is that in addition to seeing which statuses the credit loan consisted of during its lifetime, the transitions between statuses through time can be captured.

The third technique is weighted mean aggregation. This calculates the mean of a given feature where more recent data points have greater influence on the resulting value and older data points are given less influence.

A general technique applied to create features applied to all datasets was creating new features that represent ratios, percentages or linear combinations of other features. In addition to using techniques that create features, other techniques to remove features were implemented to reduce the dimensionality of the data. Using the correlation data frame, highly correlated features can be discovered. If two features happen to be highly correlated, one can be removed without losing much information.

Once all datasets have been processed merged into a single data frame, the final step is to optimize LightGBM and tune the hyperparameters to make the best predictions on the testing data. The method used to find the best hyperparameters for the LightGBM algorithm is Bayesian Optimization. To use this technique, the model’s results from a range of parameter values are recorded to build posterior distributions. As more parameters are explored, the algorithm can find and use better hyperparameter values.

Benchmark

As this problem originates from Kaggle, the ROC_AUC scores for user submissions is available on the leaderboard. This competition has ended recently and the highest ROC_AUC score is 0.805. There are also public kernels that users upload showing various benchmarks for models. Many of these kernels utilize gradient boosting methods and have various benchmark scores ranging mostly ranging from 0.7 to 0.79. In specific, many kernels are utilizing LightGBM or XGBoost to obtain scores above 0.7. As a benchmark model, I decided to use LightGBM to naively predict the targets in the application_train.csv file by factorizing categorical features and splitting the data into a training and testing set (85% train & 15% test). The result was a testing ROC_AUC score of 0.506. This is extremely low as a score of 0.5 is equivalent to random predictions.

III. Methodology

Data Preprocessing

The methodology for preprocessing the data consists of individually pre-processing each dataset to be indexed by a unique application id and merging these tables together. The three techniques for time series data that are applied to features in all datasets except the main applications data are the following:

- Aggregate Functions
- Linear Regression Slopes
- Weighted Means

The final step in all datasets is to drop highly correlated features. If the correlation between two features is greater than 0.95, one of the features is removed. Many highly correlated features are created through the three techniques above.

1. Bureau Balance Data

In the bureau balance dataset, the categorical status column is transformed into several one hot encoded columns for each status category. The months column does not require any transformation. For each unique credit bureau id, the mean, weighted mean, and slope with respect to months are calculated for each status column. This new data is now ready to merge with the bureau dataset on the credit bureau ids. No other pre-processing steps are taken as this dataset only contains time series data detailing the status of the previous credit from the Credit Bureau.

- Aggregate Functions: (Mean)
- Linear Regression Time Column / Weighted Means Column: DAYS_CREDIT

2. Bureau Data

There are 3 categorical columns in the bureau dataset (credit_type, credit_active, credit_currency). Since these columns contain many sparse categories, the number of categories in each of these columns is reduced before performing one hot encoding.

Categorical Column	Number of Categories	Reduced Number of Categories	Replacement
CREDIT TYPE	15	5	'OTHER'
CREDIT ACTIVE	4	2	'OTHER'
CREDIT CURRENCY	4	2	'OTHER'

The following 7 features were created from this data:

```
bur['bur_f1'] = bur['DAYS_CREDIT_ENDDATE']-bur['DAYS_ENDDATE_FACT']
bur['bur_f2'] = bur['DAYS_CREDIT']-bur['DAYS_CREDIT_ENDDATE']
bur['bur_f3'] = bur['DAYS_CREDIT']-bur['CREDIT_DAY_OVERDUE']
bur['bur_f4'] = bur['AMT_CREDIT_MAX_OVERDUE']/bur['AMT_CREDIT_SUM']
bur['bur_f5'] = bur['AMT_CREDIT_SUM_DEBT']/bur['AMT_CREDIT_SUM_LIMIT']
bur['bur_f6'] = bur['DAYS_CREDIT_UPDATE']-bur['DAYS_CREDIT']
bur['bur_f7'] = bur['AMT_CREDIT_SUM']/bur['AMT_CREDIT_SUM_LIMIT']
```

- Aggregate Functions: (Min, Max, Mean, Sum, Nunique, Count)
- Linear Regression Time Column / Weighted Means Column: DAYS_CREDIT

3. Credit Card Balance Data

There is only one categorical column in the credit card balance dataset named NAME_CONTRACT_STATUS. There are X categories for this feature, but Y% of rows in the data belong to the 'Active' category. This column is transformed into a binary categorical feature where a value of 1 means 'Active' and a value of 0 is any other original category. The cnt_percents function calculates a ratio between two features.

The following 9 features were created from this data:

```
cc['cc_f1'] = cc['AMT_CREDIT_LIMIT_ACTUAL']/cc['AMT_BALANCE']
cc['cc_f2'] = cc['AMT_DRAWINGS_ATM_CURRENT']/cc['AMT_BALANCE']
cc['cc_f3'] = cc['AMT_INST_MIN_REGULARITY'] / cc['AMT_PAYMENT_CURRENT']
cc['cc_f4'] = cc['AMT_PAYMENT_TOTAL_CURRENT'] / cc['AMT_CREDIT_LIMIT_ACTUAL']
cc['cc_f5'] = cc['AMT_RECIVABLE'] / cc['AMT_RECEIVABLE_PRINCIPAL']
cc['cc_f6'] = cc['AMT_TOTAL_RECEIVABLE'] / cc['AMT_RECEIVABLE_PRINCIPAL']
cc['cc_f7'] = hf.cnt_percents(cc,'CNT_DRAWINGS_ATM_CURRENT','CNT_DRAWINGS_CURRENT')
cc['cc_f8'] = hf.cnt_percents(cc,'CNT_DRAWINGS_OTHER_CURRENT','CNT_DRAWINGS_CURRENT')
cc['cc_f9'] = hf.cnt_percents(cc,'CNT_DRAWINGS_POS_CURRENT','CNT_DRAWINGS_CURRENT')
```

First, the data is processed to have features created for each unique previous application id:

- Aggregate Functions:

  Depending, on the feature, different aggregate functions are used.

```
cc_prev_aggs_dict = {'MONTHS_BALANCE':['count','mean']}
for i in cc.columns[3:]:
    if i.startswith('AMT'):
        cc_prev_aggs_dict[i] = ['min','max','mean','sum']
    if i.startswith('CNT'):
        cc_prev_aggs_dict[i] = ['min','max','mean','var','sum']
    if i.startswith('SK_DPD'):
        cc_prev_aggs_dict[i] = ['max','mean','var','sum']
    if i.startswith('cc_f'):
        cc_prev_aggs_dict[i] = ['min','max','mean','var','nunique']
```

- Linear Regression Time Column / Weighted Means Column: MONTHS_BALANCE

  Only aggregate functions are used to create features for each unique current application id. This is because the number of unique previous applications is extremely close to the number of current application ids.

4. Installment Payments Data

The following 5 features were created from this data. The differences are interpreted as the amount due minus the amount paid and the due date min the date paid. Binary features are also created that signal if a payment was not the full amount or the payment was late.

```
installs['PAY_DIFF'] = installs['AMT_INSTALMENT'] - installs['AMT_PAYMENT']
installs['DAYS_DIFF'] = installs['DAYS_INSTALMENT'] - installs['DAYS_ENTRY_PAYMENT']
installs['PAY_DIFF2'] = [1 if i >= 0 else 0 for i in installs['PAY_DIFF']]
installs['DAYS_DIFF2'] = [1 if i >= 0 else 0 for i in installs['DAYS_DIFF']]
installs['ins_f1'] = installs['PAY_DIFF']/installs['AMT_PAYMENT']
```

First, the data is processed to have features created for each unique previous application id:

- Aggregate Functions

```
{'NUM_INSTALMENT_VERSION':['nunique','count','var'],
'DAYS_INSTALMENT':['max','mean',],
'DAYS_ENTRY_PAYMENT':['max','mean'],
'AMT_INSTALMENT':['min','max','mean','var','sum','std'],
'AMT_PAYMENT':['min','max','mean','var','sum','std'],
'PAY_DIFF':['min','max','mean','var','sum','std'],
'DAYS_DIFF':['min','max','mean','var','sum','std'],
'PAY_DIFF2':['min','max','mean','var','sum','std'],
'DAYS_DIFF2':['min','max','mean','var','sum','std'],
'ins_f1':['min','max','mean','var','sum','std']}
```

- Linear Regression Time Column / Weighted Means Column: NUM_INSTALMENT_NUMBER

Using the same techniques, the data is then processed to create features for each unique current application id:

- Aggregate Functions: (Min, Max, Mean, Sum)
- Linear Regression Time Column / Weighted Means Column: DAYS_INSTALMENT_mean

5. Point of Sales Cash Balance Data

First, the data is processed to have features created for each unique previous application id:

- Aggregate Functions

```
if i == 'CNT_INSTALMENT':
    pos_prev_aggs_dict[i] = ['nunique','count']
if i == 'CNT_INSTALMENT_FUTURE':
    pos_prev_aggs_dict[i] = ['min','max','mean','sum']
if i.startswith('SK_DPD'):
    pos_prev_aggs_dict[i] = ['max','mean','sum','var']
else:
    pos_prev_aggs_dict[i] = ['max','mean','var']
```

- Linear Regression Time Column / Weighted Means Column: MONTHS_BALANCE

This data is then merged with the previous applications data where the new time and weighted means column be from the DAYS_CREDIT feature.

6. Previous Applications Data

There are outliers in the columns starting with DAYS. The value 365243 is replaced with np.nan. The column SELLERPLACE_AREA uses (-1) to represent missing values. These values are replaced with np.nan as well. The HOUR_APPR_PROCESS_START has standard scaling applied to it and the NFLAG_INSURED_ON_APPROVAL has its column value type changed from float to int since it is a binary category.

The following 4 features were created from the data:

```
prev['prev_f1'] = prev['AMT_ANNUITY']/prev['AMT_APPLICATION']
prev['prev_f2'] = prev['RATE_DOWN_PAYMENT']/prev['RATE_INTEREST_PRIMARY']
prev['prev_f3'] = prev['DAYS_LAST_DUE']-prev['DAYS_TERMINATION']
prev['prev_f4'] = prev['DAYS_FIRST_DUE']/prev['DAYS_FIRST_DRAWING']
```

Categorical columns have their categories reduced according to the table below:

Categorical Column	Number of Categories	Reduced Number of Categories	Replacement
NAME_CONTRACT_TYPE	4	2	'OTHER'
NAME_CASH_LOAN_PURPOSE	25	2	'OTHER'
NAME_PAYMENT_TYPE	4	2	'OTHER'
NAME_GOODS_CATEGORY	28	9	'OTHER'
NAME_PORTFOLIO	5	4	'OTHER'
CHANNEL_TYPE	8	6	'OTHER'

NAME_SELLER_INDUSTRY	11	6	'OTHER'
PRODUCT_COMBINATION	17	7	'OTHER'
NAME_CLIENT_TYPE	4	2	'OTHER'
NAME_TYPE_SUITE	7	4	'OTHER'
CODE_REJECT_REASON	9	5	'OTHER'

- Aggregate Functions:

The mean of the categorical columns is aggregated. The numerical columns' aggregate functions are the min, max, mean, and sum.

- Linear Regression Time Column / Weighted Means Column: DAYS_DECISION

7. Applications Data

The current applications dataset is the only dataset where categorical features are not transformed into numerical values. The following replacements were performed on the categorical features:

Categorical Column	Categories Replaced	Replacement
NAME_INCOME_TYPE	Maternity leave, Businessman, Student, Unemployed	Np.nan
NAME_TYPE_SUITE	Other_B,Other_A,Student,Group of people	Np.nan
NAME_FAMILY_STATUS	Unknown	Np.nan

Numerical columns CNT_FAM_MEMBERS, CNT_CHILDREN, DAYS_EMPLOYED, DAYS_ID_PUBLISH, DAYS_BIRTH, and HOUR_APPR_PROCESS_START are converted from integers to floats.

These 10 features were created from the data:

```
X_nums['APP_f1'] = X_nums['OBS_30_CNT_SOCIAL_CIRCLE'] - X_nums['DEF_30_CNT_SOCIAL_CIRCLE']
X_nums['APP_f2'] = X_nums['DEF_60_CNT_SOCIAL_CIRCLE'] - X_nums['DEF_30_CNT_SOCIAL_CIRCLE']
X_nums['APP_f3'] = X_nums['AMT_CREDIT'] / X_nums['AMT_ANNUITY']
X_nums['APP_f4'] = X_nums['AMT_INCOME_TOTAL'] / X_nums['AMT_CREDIT']
X_nums['APP_f5'] = X_nums['AMT_ANNUITY'] / X_nums['AMT_INCOME_TOTAL']
X_nums['APP_f6'] = X_nums['EXT_SOURCE_1'] + X_nums['EXT_SOURCE_2'] + X_nums['EXT_SOURCE_3']
X_nums['APP_f7'] = X_nums['DAYS_BIRTH'] / X_nums['DAYS_EMPLOYED']
X_nums['APP_f8'] = X_nums['DAYS_REGISTRATION'] / X_nums['DAYS_ID_PUBLISH']
X_nums['APP_f9'] = X_nums['DAYS_EMPLOYED'] / X_nums['OWN_CAR_AGE']
X_nums['APP_f10'] = X_nums['CNT_CHILDREN'] / X_nums['CNT_FAM_MEMBERS']
```

Categorical columns are factorized and then concatenated with numerical features. All datasets are now able to be merged as they are all indexed by unique current application ids.

Implementation

The main techniques mentioned are used multiple times in preparing the data and have associated functions in the helper_functions.py file.

1. *get_aggs_df(df, skid, agg_dict)*
    - df: Data Frame
    - skid: Name of column to group by
    - agg_dict : Dictionary of column name and desired aggregations

    Returns data frame with one unique id for each row and aggregated columns for each column specified in the agg_dict dictionary.
2. *get_slopes(df, skids, id_name, x_col, y_cols*)
    - df: Data Frame
    - skids: List of unique ids to group by
    - id_name: Name of column to group by
    - x_col: Name of column representing x-axis in Linear Regression model
    - y_cols: List of column names to calculate slopes for

    Returns data frame with one unique id for each row and linear regression slope coefficients for each column specified in y_cols.
3. get_weighted(df, id_name, count_col, cols)
    - df: Data Frame
    - skids: List of unique ids to group by
    - id_name: Name of column to group by
    - cols: List of column names to calculate weighted means for

    Returns data frame with one unique id for each row and the weighted mean value for each column specified in cols based on the order of rows.
4. *drop_high_corrs(df, threshold)*
    - df: Data Frame
    - threshold: Threshold used to find highly correlated features. If two features have a correlation > threshold, one of those features is dropped.

    Returns new data frame by finding highly correlated features that correlate more than the specified threshold and only keeping one of those features.

The hyperparameters chosen to tune the LightGBM classification model and the ranges for parameter exploration through Bayesian optimization are shown below.

Ranges of Optimized Parameters:

```
'num_leaves': (6, 63),
'feature_fraction': (0.4, 0.95),
'bagging_fraction': (0.6, 0.95),
'max_depth': (8, 15.99),
'lambda_l1': (0, 1),
'lambda_l2': (0, 1),
'min_split_gain': (0.005, 0.1),
'min_child_weight': (10, 50),
'min_data_in_leaf': (1000,8000),
'scale_pos_weight': (5,12),
'bagging_freq': (2,5)}
```

Refinement

Initially when working on this project, I tried to solve the problem with other algorithms and techniques. Before deciding on using Light GBM as the main algorithm, I used a deep learning neural network approach. This consisted on entirely different pre-processing steps and implementation. I had created a workflow that used either linear regressions, decision trees and neural networks to first predict missing values in the various datasets before making predictions of the target value for each current application. I mostly used linear regression for predicting continuous or numerical null values and I used decision trees to predict categorical missing values. I ended up achieving a submission AUC score of 0.76 on the Kaggle test data. I wanted to solve the problem with a deep learning approach as I noticed not many people in the competition could achieve a score higher than 0.75 using a neural network. I saw a significant improvement in the neural network approach from simply choosing to predict most null values in the data. Previously, I had either replaced null values with the mean of the column or zeros. This led to my increase in score using a neural network from 0.75 to 0.76. Once I felt like I couldn't increase my score from using neural networks I decided to try out LightGBM, the algorithm that many people participating in the competition were using to achieve higher scores. I ran the LightGBM algorithm on the features that I used for the neural network approach and with some simple parameter tuning, I was able to produce a 0.77 AUC score on a submission.

After seeing how the same features used in the neural network approach led to an immediate increase in score from switching the algorithm to LightGBM, I decided that my final solution would utilize the Light GBM algorithm. I had to re-think my pre-processing methodology and create new features that would be optimal to feed LightGBM. In many ways, the methodology and techniques used became simpler due to fewer steps required for data pre-processing for LightGBM. The biggest change in switching approaches was the fact that LightGBM could accept null values and there was no need to impute or predict missing values in the data. I felt more comfortable this way as creating or predicting values that were not in the original dataset felt incorrect to do for many columns.

Without the need to worry about missing values, I focused on how to create and extract better features from the data available. I eventually came up with the idea to use Linear Regression for the time series data and started to create a module with helper functions that I would use to write helpful functions that I would use on most datasets. As I got more familiar with the data, I noticed that I had to perform the same actions across multiple tables. Creating the helper functions module brought more consistency and clarity to my workflow.

After re-pre-processing all datasets to be used by LightGBM with the linear regression techniques applied, I achieved a higher submission of 0.78. At this time, I was struggling to optimize the hyperparameters of LightGBM and manually testing combinations of hyperparameters. I turned to using grid search and Bayesian optimization to tune hyperparameters. I ended up sticking with the Bayesian optimization tuning approach as it seemed to produce better results much quicker than using a grid search. After searching for and optimizing hyperparameters, I reached another submission all time high of 0.787.

IV. Results

Model Evaluation and Validation

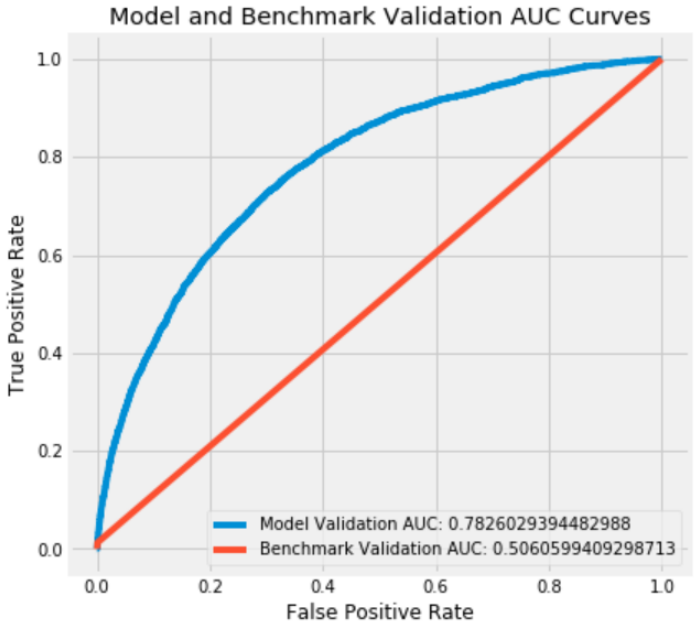
The final model contains 1252 features. A 5-fold cross validation training is performed to obtain optimal hyperparameters. The average AUC score during this hyperparameter search was 0.788. The model is robust as there is little fluctuation in score as parameters are searched. The hyperparameters for the Light GBM model are detailed below.

```
'num_leaves': 61,
'learning_rate':0.005,
'feature_fraction': 0.48661969486659096,
'bagging_fraction': 0.8503364504260739,
'max_depth': 13,
'lambda_l1': 0.11722803196057063,
'lambda_l2': 0.8893748763177247,
'min_split_gain': 0.0837960708150188,
'min_child_weight': 10.39103579011432,
'min_data_in_leaf': 2934,
'scale_pos_weight': 5.7380535943036595,
'bagging_freq': 2,
```

After finding optimal parameters, a LightGBM classifier is trained on 85% of the training data with these parameters. The remaining 15% is for the validation set. The learning rate is dropped to 0.005 and the algorithm stops boosting if scores on the validation set don't improve within 500 boosting rounds. This is data that the model has never seen before and it was able to achieve a score on the validation data of 0.782. Upon submission to the Kaggle test data, the model was able to reach a higher score of 0.787.

Justification

The model's final solution strongly outperforms the benchmark solution established. There is a significant difference in the quality of predictions between the final model and the benchmark that is quantified through the ROC AUC score. The AUC score on the validation data for the benchmark and final model are 0.506 and 0.782, respectively. We can visualize the difference in prediction quality between the benchmark and final model below.

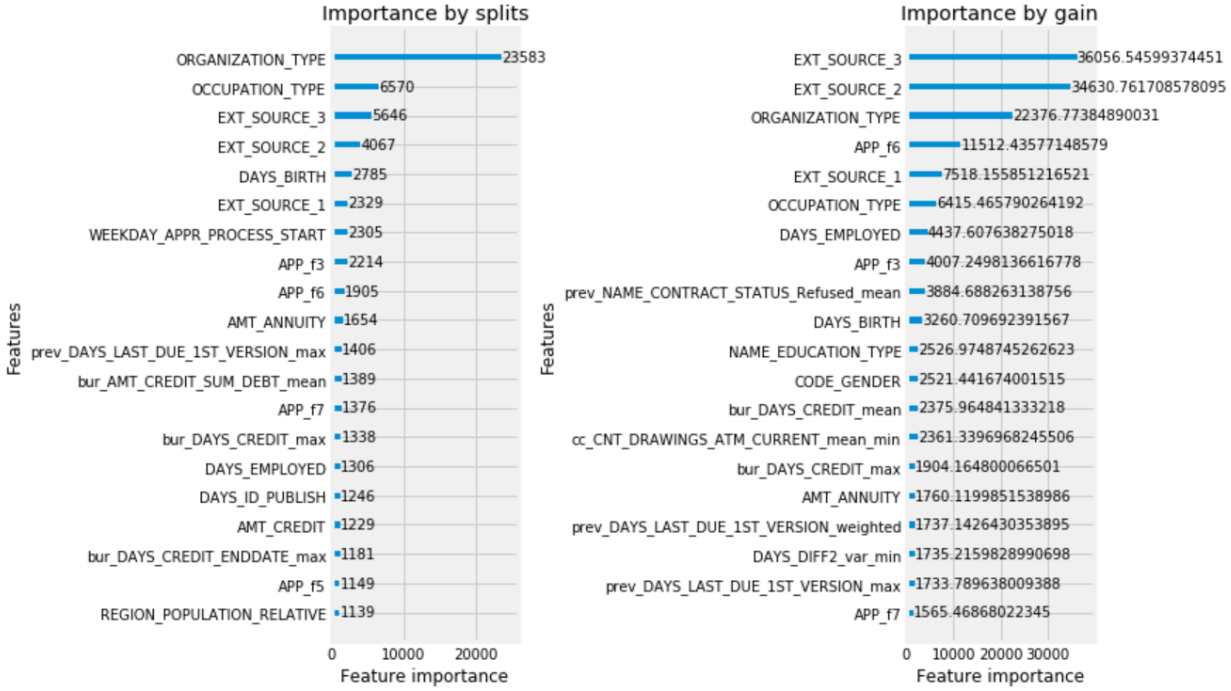


The final model performs well and is able to generalize to new data. The Kaggle submission score for using this model to predict the test data target outcomes is 0.787, which is extremely close to the training score.

V. Conclusion

Free-Form Visualization

The following visualization presents the feature importances of the best 20 features used by the final model.



The results show the EXT_SOURCE features in the applications datasets are extremely important in predicting the target outcomes. There are also other features with high importance that were created using the mentioned techniques to pre-process data.

Reflection

The solution for this project repeated uses a few key techniques to prepare data for LightGBM. All tables were processed individually and later merged to run final predictions. Most features created in this implementation consist of ratios or linear combinations of various columns that could be useful.

In each dataset other than the main applications data, linear regressions and aggregate functions are used to summarize time series data for each unique applicant. Categorical features in these datasets are converted to numerical features through one hot encoding. Each category for a given categorical column becomes a new column. If a categorical column contains many categories, there are many columns created. Reduction of sparse categories is crucial as categories with few rows pertaining to them do not give much information. I have used two ways to deal with this issue; one is to grouped categories together or entirely replace certain categories with NaN values. This works for the LightGBM algorithm because it can accept null values as input unlike many other approaches such as using neural networks. This workflow and handling of categorical time series data only allows categorical features from the main applications data to be used as categorical features by LightGBM.

Using many aggregate functions rapidly increases dimensionality. Removing highly correlated columns worked well for reducing dimensionality in the data and was often done. However, I did notice that sometimes the AUC score would drop slightly. Running LightGBM for each dataset and inspecting feature importances played a large role in reducing dimensionality and gaining insight into the features driving the algorithm.

One aspect about the project that was difficult was trying to tune LightGBM without Bayesian optimization. In the end I used Bayesian optimization because trying to manually tune parameters and using a grid search technique was tedious and took too long to run to find optimal parameters.

The final model fits my expectations for the given problem. The algorithms and techniques used are widely applicable and can be used for these types of problems and datasets.

Improvement

I believe the main improvements that would have the best results for my implementation are the methods for handling time series data. Using linear regression had a significant impact on performance, but adding new features using the weighted mean method for time series data led to an unexpectedly less significant increase in score. I believe that there is a better method to extract relevant information from various types of time series data.

The other areas of improvement include engineering better features and finding better combinations or model hyperparameters. The range of values of hyperparameters can be extended to possibly discover a new set of hyperparameters that better predict the target outcomes. However, this will greatly increase the time needed to obtain satisfactory hyperparameters.