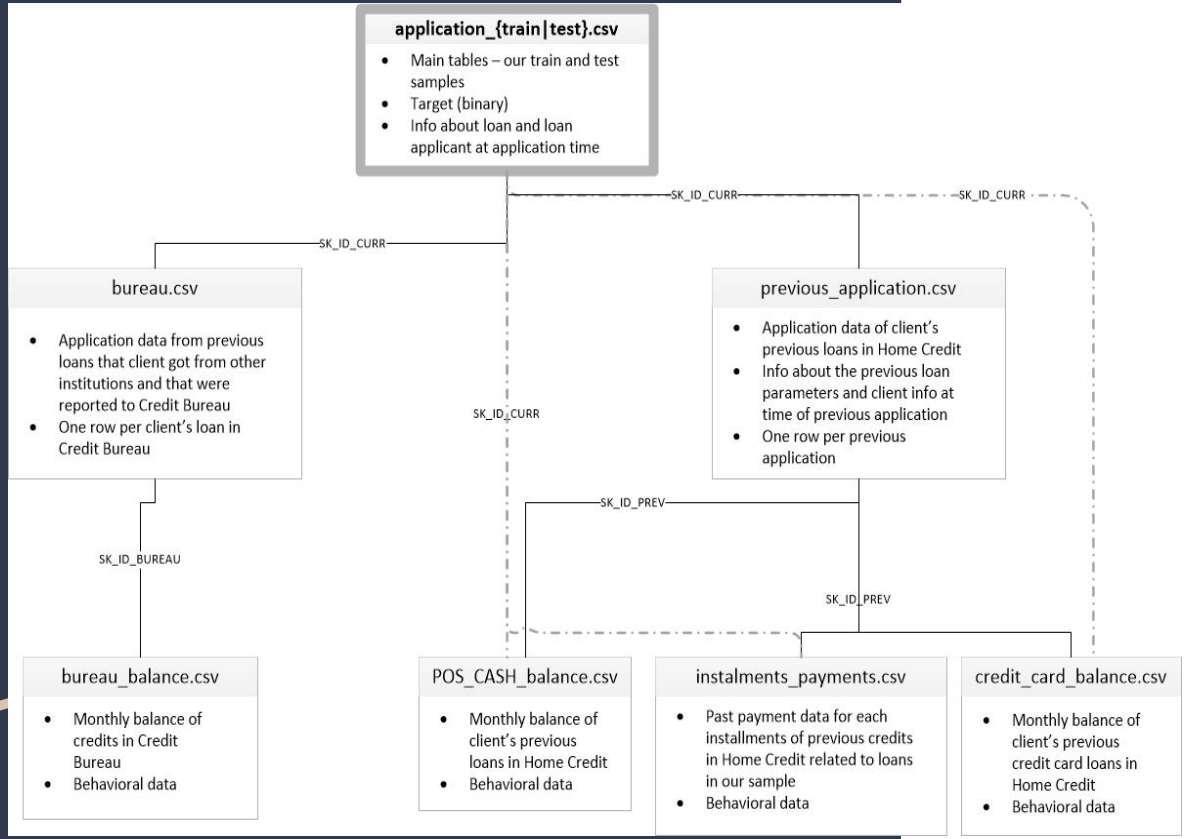
Home Credit Default Risk

Analysis and System Design

Git

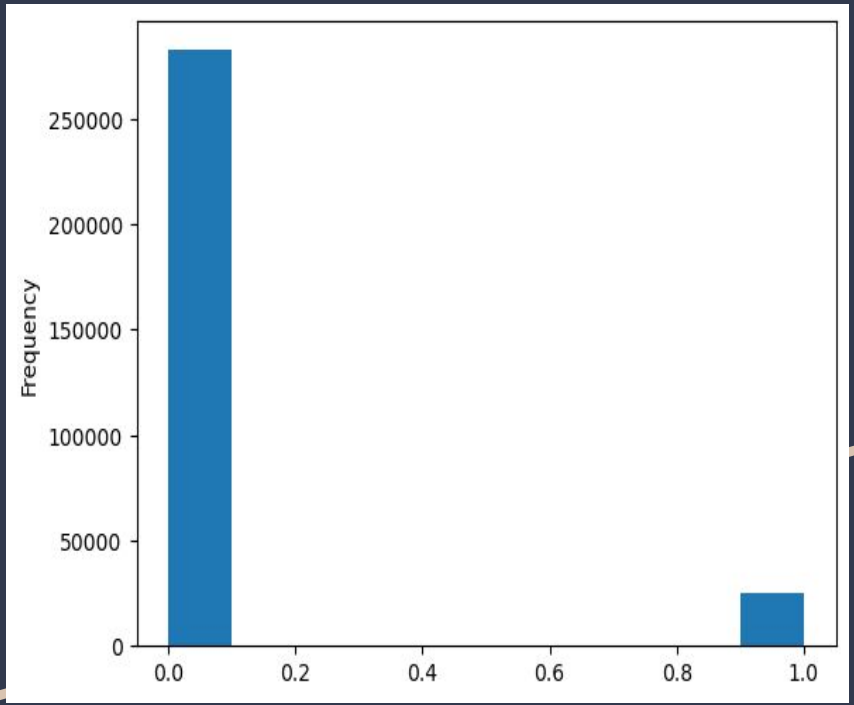
Akshay Bhosale
E: akshaysb@jobhuntmail.com
M: +16176521134

Problem Statement



The Home Credit Default Risk competition on Kaggle aims to improve the credit risk scoring models used by Home Credit, a provider of consumer finance. The objective is to predict the probability of a client defaulting on a loan, based on their application data and various related datasets. By accurately predicting default risk, Home Credit can improve their financial decision-making, offer loans to more individuals, and reduce financial risk.

Dataset Overview



application_train.csv and **application_test.csv**: These files contain the main application data for training and testing, respectively. They include demographic information, financial data, and other details about the applicants. The target variable for the training set is `TARGET`, which indicates whether an applicant defaulted on the loan. It can be observed that the data is imbalance in favour of '0' Target

bureau.csv: This file contains information about the applicant's previous credit history with other financial institutions. Each row represents a previous credit record.

bureau_balance.csv: This file provides monthly balance information for each credit record in `bureau.csv`. It helps track the applicant's repayment history over time.

credit_card_balance.csv: This dataset includes monthly balance and other credit card-related information for the applicants. It helps understand the applicant's credit card usage and payment behavior.

Dataset

installments_payments.csv: This file contains payment history for previous loans the applicant had with Home Credit. It includes details about the payment amounts and whether they were made on time.

POS_CASH_balance.csv: This dataset provides monthly balance snapshots for point-of-sale and cash loans. It includes information about the status and amount of the loans.

previous_application.csv: This file contains information about the applicant's previous loan applications with Home Credit. It includes details about the loan type, amount, and whether it was approved or rejected.

Each dataset plays a crucial role in building a comprehensive view of the applicant's financial behavior and creditworthiness. Combining and analyzing these datasets can help in developing a robust model to predict the likelihood of default.

Application Train | Test

1. Size
    a. application_train - rows: 307511 columns: 122
    b. application_test - rows: 48744 columns: 121
2. Gender Anomaly: **XNA**
3. Income Outlier at $117M
4. Employed Anomaly: 1000 years
5. Replace 365243 with Nan across the dataset for all DAY features
6. Group and Bin Ages to make Year Bins
7. Get ratio and weights for all 3 EXT rating sources along with their statistics
8. Convert DAY Income Annuity and Payment Rate features to percentages
9. Compute previous loans from bureau dataset and attach it to this dataframe as a feature
10. Perform Standard One Hot Encoding
11. Carefully read through column importance and dropped columns with least significance after creating new features
12. Get ratios of:
    a. Income Features
    b. Credit Features
    c. Time Features

Bureau and Bureau Balance

1. SK_ID_BUREAU and SK_ID_CURR help us group this dataset
2. Credit duration and credit/account end date difference
3. Credit to debt ratio and difference
4. Credit Day Overdue
5. Perform Aggregations on Bureau Balance and Merge with Bureau
6. Status of Credit Bureau Loan during Month
7. Bureau and bureau_balance numeric features and categorical features aggregation
8. Bureau: Active credits - using only numerical aggregations
9. Bureau: Closed credits - using only numerical aggregations
10. One Hot Encoding with Nan

Previous Applications

1. Use SK_ID_CURR for groupby
2. Days 365.243 values -> nan
3. Add feature: value ask / value received percentage
4. Ratio and Difference with Amount of application, Credit Amount, and Down Payment Amount
5. Simplified Interest Ratio on Previous Application
6. Numerical Features:
    a. Previous Approved Applications
    b. Previous Refused Applications

POS CASH Balance

1. Use SK_ID_CURR and SK_ID_PREV for grouping
2. Flag Features for months with late payment
3. Aggregate numerical features for simple stats
4. Aggregate categorical features for mean
5. Count pos cash accounts
6. Percentage of previous loans completed and completed before initial term
7. Number of remaining installments (future installments) and percentage from total
8. Group by SK_ID_CURR and merge
9. Percentage of late payments for the 3 most recent applications
10. Last 3 most recent applications

Installment Payments

1. Group with SKI_ID_CURR
2. Group payments and get Payment difference
3. Percentage and difference paid in each installment
4. Days past due and days before due
5. Flag features for late payment added
6. Flag late payments that have a significant amount
7. Count installments account
8. Perform aggregations on numeric data

Credit Card Balance

1. Use SK_ID_CURR and SK_ID_PREV
2. Amount used from limit
3. Current payment vs Min payment
4. Days Past Overdue or Late Payment flags
5. How much money withdrawn of the limit
6. General aggregations and grouping
7. Count credit card lines for all SK_ID_CURR
8. Last month balance of each credit card application

Model Training for Feature Importances

- Merge the individual datasets into one, using left join on SK_ID_CURR
- Drop all features that have more than 75% missing values
- Drop columns that have single unique value
- This helps us reduce unnecessary noise from model training process

1. LightGBM with 3-Fold Cross Validation

- Before making the final submission, train and fit a LightGBM on the merged dataframe cross-validated with few folds
- Keep a list of columns that have feature importance score below a threshold
- Randomly pick hyperparameter to observe changes in drop features list
- Drop the columns from the compiled data frame to achieve a shorter
- Change threshold to make main model training features less

```
# Create the model
model = LGBMClassifier(n_estimators=2226, objective = 'binary',
                       class_weight = 'balanced', learning_rate = 0.01,
                       reg_alpha = 0.02, reg_lambda = 0.9,
                       subsample = 0.8324, n_jobs = -1, random_state = 500)

# Train the model
model.fit(train_features, train_labels, eval_metric = 'auc',
          eval_set = [(valid_features, valid_labels), (train_features, train_labels)],
          eval_names = ['valid', 'train'])
```

2. LightGBM with K-Fold CV

```
for n_fold, (train_idx, valid_idx) in enumerate(folds.split(train_df[feats], train_df['TARGET'])):
    train_x, train_y = train_df[feats].iloc[train_idx], train_df['TARGET'].iloc[train_idx]
    valid_x, valid_y = train_df[feats].iloc[valid_idx], train_df['TARGET'].iloc[valid_idx]
    clf = LGBMClassifier(nthread=-1,
                         #device_type='gpu',
                         n_estimators=5000,
                         learning_rate=0.01,
                         max_depth=11,
                         num_leaves=58,
                         colsample_bytree=0.613,
                         subsample=0.708,
                         max_bin=407,
                         reg_alpha=3.564,
                         reg_lambda=4.930,
                         min_child_weight=6,
                         min_child_samples=165,
                         #keep_training_booster=True,
                         silent=-1,
                         verbose=-1,)

    clf.fit(train_x, train_y, eval_set=[(train_x, train_y), (valid_x, valid_y)], eval_metric='auc')
```

Perform post processing on the entire dataframe.

Drop features with less importance score achieved on previously trained LightGBM

Use risk groupanizer function to calculate the risk around individual SK_ID_CURR with respect Target and add features

Using parameters best suited for modeling on this problem statement, referring Olivier's solution for domain expertise

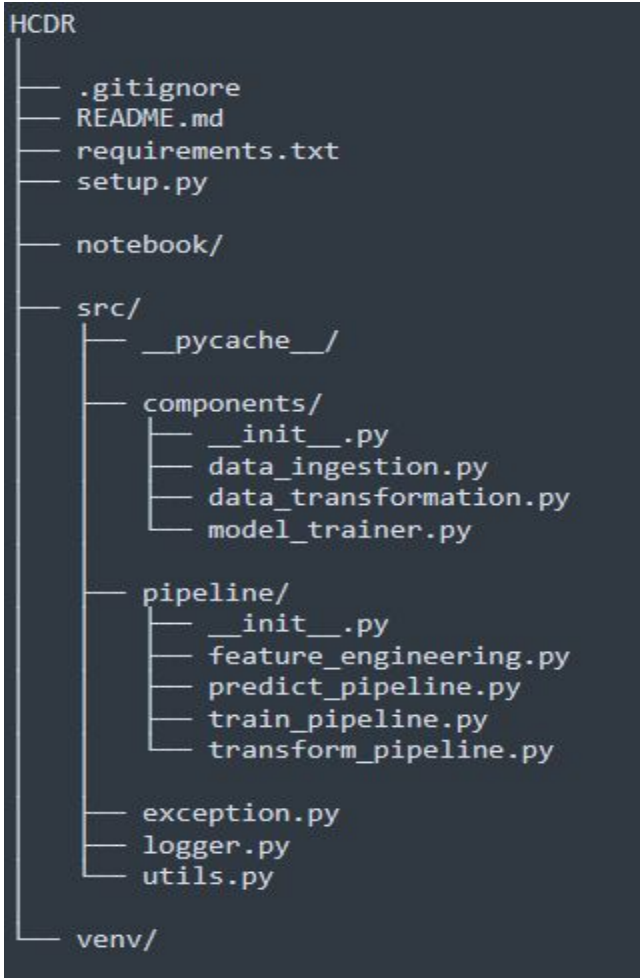
Reinitiate modeling on shorter and important features only (as per our threshold) data, while also using the initially acquired submission set to floor or ceil values and train LightGBM again

System Design

The System design shows basic pipeline structure of the ML system for modeling Home Credit default Risk's problem statement

The pipeline ensures smooth functioning of important activities

While the component structures help in bridging gaps in between pipeline activities.



Further Development

- We can use containers with Docker for easier deployment
- Further with containerized app, we can deploy this to AWS ECR
- Further domain expertise and feature engineering can help increase ROC-AUC score
- Using PCA for dimensionality reduction, although does not explains which feature is given importance
- Use automated feature engineering and automated model engineering tools

Submissions - ROC AUC SCORE

submission.csv Complete (after deadline) · 1h ago	0.78578	0.78485
submission.csv Complete (after deadline) · 6h ago	0.78549	0.78459
submissionxgb.csv Complete (after deadline) · 3d ago	0.75213	0.75733
submission2.csv Complete (after deadline) · 3d ago	0.50000	0.50000
submission2.csv Complete (after deadline) · 3d ago	0.76312	0.76746
submission1.csv Complete (after deadline) · 3d ago	0.78249	0.78561