Home Credit Default Risk

Group 13 (Phase - 3)




William Cutchin

wcutchin@iu.edu

Krisha Mehta

krimeht@iu.edu

Kunal Mehra

kumehra@iu.edu

Kalyani Malokar

kmalokar@iu.edu

Outline:

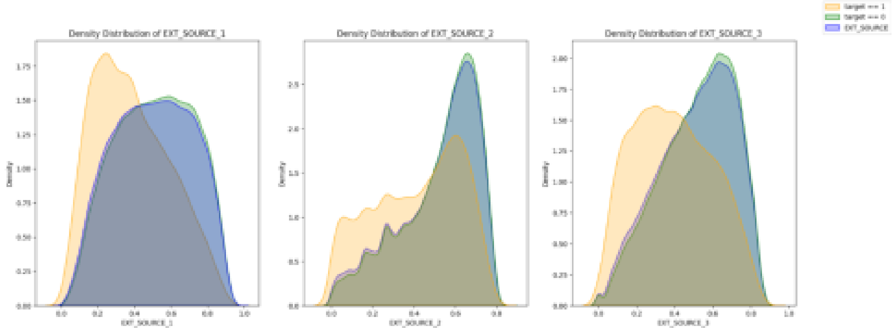
- Project Description
- Summary Visual EDA
- ML Pipeline
- Modeling Pipeline Description
- Feature Engineering
- Hyperparameter Tuning
- Results and discussion of results
- Conclusion and next steps

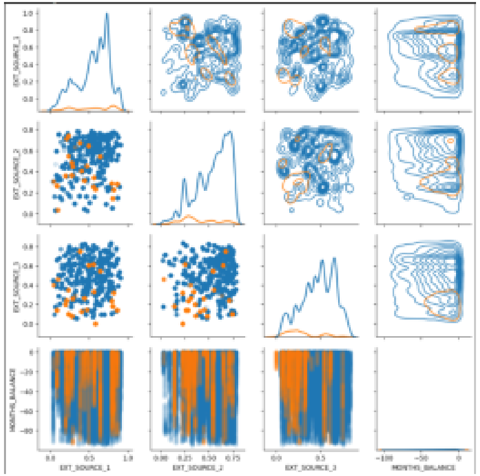
Project Description

- Home Credit Group looks to use alternative data that will help them decide if they are able to lend to these individuals and to predict their client's repayment abilities.
- Here we are using various machine learning and statistical models like Logistic Regression and Random Forest Classifier and Decision Tree to get these predictions.
- Our goal in this phase is to perform feature engineering and hyperparameter tuning on the selected features and train model using best parameters and setting.
- We aim at cleaning and preprocessing techniques on the data, perform feature Engineering, and hyperparameter tuning as we build pipeline after examining the accuracies of explored pipelines.

Summary Visual EDA

- The use of Kernel Distribution Estimation was extremely helpful when examining separation of target distributions
- Heatmaps were essential in understanding correlations and feature selection





Feature Engineering

- Process of feature Engineering
  - Take the secondary table
  - Build new features for the data
  - Run Correlation Analysis on the new table including the new features
  - Select the features with a correlation above the specified threshold
  - Merge this new highly correlated table to the main train set
- We chose this method since correlation to the target seems to be an accurate predictor of data significance

Feature Engineering: Feature Creation

```
[ ] # Create Features for the Bureau and Bureau_balance
    #--------------------------------------------------

    ## term of credit granted to the individual with the loan
    bur_merge['BUR_END_DAY_RATIO'] = bur_merge['DAYS_CREDIT_ENDDATE'] / bur_merge['DAYS_CREDIT']
    bur_merge['BUR_END_DAY_RATIO'].replace([np.inf, -np.inf], np.nan, inplace=True)
    bur_merge['BUR_END_DAY_RATIO'] = bur_merge['BUR_END_DAY_RATIO'].fillna(bur_merge['BUR_END_DAY_RATIO'].mean())

    ## amount repaid per year
    bur_merge['BUR_DEBT_ANNUITY_RATIO'] = bur_merge['AMT_CREDIT_SUM_DEBT'] / bur_merge['AMT_ANNUITY']
    bur_merge['BUR_DEBT_ANNUITY_RATIO'].replace([np.inf, -np.inf], np.nan, inplace=True)
    bur_merge['BUR_DEBT_ANNUITY_RATIO'] = bur_merge['BUR_DEBT_ANNUITY_RATIO'].fillna(bur_merge['BUR_DEBT_ANNUITY_RATIO'].mean())

    # debt to limit ratio - responsibility with credit
    bur_merge['BUR_DEBT_LIMIT_RATIO'] = bur_merge['AMT_CREDIT_SUM_DEBT'] / bur_merge['AMT_CREDIT_SUM_LIMIT']
    bur_merge['BUR_DEBT_LIMIT_RATIO'].replace([np.inf, -np.inf], np.nan, inplace=True)
    bur_merge['BUR_DEBT_LIMIT_RATIO'] = bur_merge['BUR_DEBT_LIMIT_RATIO'].fillna(bur_merge['BUR_DEBT_LIMIT_RATIO'].mean())

    # proportion of the borrower's income that is dedicated to repaying the loan.
    bur_merge['BUR_CREDIT_ANNUITY_RATIO'] = bur_merge['AMT_CREDIT_SUM'] / bur_merge['AMT_ANNUITY']
    bur_merge['BUR_CREDIT_ANNUITY_RATIO'].replace([np.inf, -np.inf], np.nan, inplace=True)
    bur_merge['BUR_CREDIT_ANNUITY_RATIO'] = bur_merge['BUR_CREDIT_ANNUITY_RATIO'].fillna(bur_merge['BUR_CREDIT_ANNUITY_RATIO'].mean())

    # total debt for each loan reported in the bureau data.
    bur_merge['BUR_CREDIT_DEBT_RATIO'] = bur_merge['AMT_CREDIT_SUM'] / bur_merge['AMT_CREDIT_SUM_DEBT']
    bur_merge['BUR_CREDIT_DEBT_RATIO'].replace([np.inf, -np.inf], np.nan, inplace=True)
    bur_merge['BUR_CREDIT_DEBT_RATIO'] = bur_merge['BUR_CREDIT_DEBT_RATIO'].fillna(bur_merge['BUR_CREDIT_DEBT_RATIO'].mean())

    # difference between credit record date and update
    bur_merge['BUR_DAY_UPDATE_DIFF'] = bur_merge['DAYS_CREDIT'] - bur_merge['DAYS_CREDIT_UPDATE']
```

Hyperparameter Tuning

The models we have selected are Logistic Regression, Random Forest and Decision Tree. For them the hyperparameters we chose are as follows:

Logistic Regression - Selected the C parameter to control the penalty strength, the penalty parameter to determine the type of loss to use and solver was kept as saga.

```
params = {'lr__C':[0.01, 0.1, 1.0, 10.0],
          'lr__penalty': ['l1','l2'],
          'lr__solver': ['saga']}
```

Random Forest - Selected max_depth parameter to control the number of levels allowed in each decision tree, max_features parameter for selecting number of features considered at every step and n_estimators parameter to select number of trees.

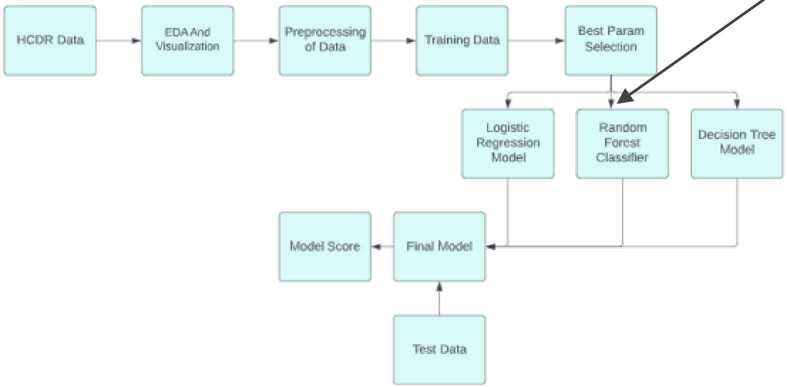
```
params = {
          'rf__max_depth': [5, 10, 15, 20, 50],
          'rf__max_features': ['log2', 'sqrt'],
          'rf__n_estimators' : [1, 10, 50, 100]}
```

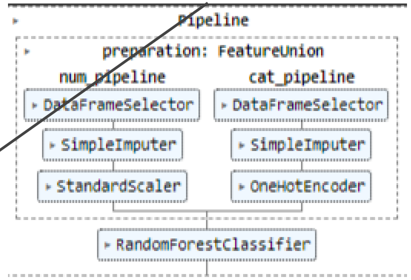
Decision Tree - Selected criterion parameter to measure the quality of the split, max_depth parameter to select the maximum depth of the tree and min_samples_leaf to select the minimum number of samples required to be at a leaf node.

```
params = {'dt__criterion':['gini', 'entropy'],
          'dt__max_depth': [5, 10, 15, 20, 50],
          'dt__min_samples_leaf' : [1,2,3,4,5]}
```

```
{'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}

Grid scores on development set:

0.633 (+/-0.004) for {'rf__max_depth': 5, 'rf__max_features': 'log2', 'rf__n_estimators': 1}
0.712 (+/-0.006) for {'rf__max_depth': 5, 'rf__max_features': 'log2', 'rf__n_estimators': 10}
0.716 (+/-0.002) for {'rf__max_depth': 5, 'rf__max_features': 'log2', 'rf__n_estimators': 50}
0.716 (+/-0.004) for {'rf__max_depth': 5, 'rf__max_features': 'log2', 'rf__n_estimators': 100}
0.644 (+/-0.005) for {'rf__max_depth': 5, 'rf__max_features': 'sqrt', 'rf__n_estimators': 1}
0.715 (+/-0.008) for {'rf__max_depth': 5, 'rf__max_features': 'sqrt', 'rf__n_estimators': 10}
0.720 (+/-0.004) for {'rf__max_depth': 5, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}
0.720 (+/-0.004) for {'rf__max_depth': 5, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}
0.648 (+/-0.008) for {'rf__max_depth': 10, 'rf__max_features': 'log2', 'rf__n_estimators': 1}
0.720 (+/-0.004) for {'rf__max_depth': 10, 'rf__max_features': 'log2', 'rf__n_estimators': 10}
0.729 (+/-0.003) for {'rf__max_depth': 10, 'rf__max_features': 'log2', 'rf__n_estimators': 50}
0.731 (+/-0.003) for {'rf__max_depth': 10, 'rf__max_features': 'log2', 'rf__n_estimators': 100}
0.672 (+/-0.011) for {'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 1}
0.723 (+/-0.001) for {'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 10}
0.733 (+/-0.002) for {'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}
0.734 (+/-0.003) for {'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}
0.610 (+/-0.015) for {'rf__max_depth': 15, 'rf__max_features': 'log2', 'rf__n_estimators': 1}
0.701 (+/-0.003) for {'rf__max_depth': 15, 'rf__max_features': 'log2', 'rf__n_estimators': 10}
0.725 (+/-0.003) for {'rf__max_depth': 15, 'rf__max_features': 'log2', 'rf__n_estimators': 50}
0.729 (+/-0.003) for {'rf__max_depth': 15, 'rf__max_features': 'log2', 'rf__n_estimators': 100}
0.615 (+/-0.007) for {'rf__max_depth': 15, 'rf__max_features': 'sqrt', 'rf__n_estimators': 1}
0.703 (+/-0.003) for {'rf__max_depth': 15, 'rf__max_features': 'sqrt', 'rf__n_estimators': 10}
0.726 (+/-0.002) for {'rf__max_depth': 15, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}
0.730 (+/-0.001) for {'rf__max_depth': 15, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}
0.560 (+/-0.011) for {'rf__max_depth': 20, 'rf__max_features': 'log2', 'rf__n_estimators': 1}
0.672 (+/-0.008) for {'rf__max_depth': 20, 'rf__max_features': 'log2', 'rf__n_estimators': 10}
0.714 (+/-0.006) for {'rf__max_depth': 20, 'rf__max_features': 'log2', 'rf__n_estimators': 50}
0.721 (+/-0.003) for {'rf__max_depth': 20, 'rf__max_features': 'log2', 'rf__n_estimators': 100}
0.566 (+/-0.018) for {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__n_estimators': 1}
0.674 (+/-0.006) for {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__n_estimators': 10}
0.714 (+/-0.003) for {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}
0.722 (+/-0.004) for {'rf__max_depth': 20, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}
0.532 (+/-0.002) for {'rf__max_depth': 50, 'rf__max_features': 'log2', 'rf__n_estimators': 1}
0.634 (+/-0.003) for {'rf__max_depth': 50, 'rf__max_features': 'log2', 'rf__n_estimators': 10}
0.696 (+/-0.003) for {'rf__max_depth': 50, 'rf__max_features': 'log2', 'rf__n_estimators': 50}
0.709 (+/-0.002) for {'rf__max_depth': 50, 'rf__max_features': 'log2', 'rf__n_estimators': 100}
0.534 (+/-0.003) for {'rf__max_depth': 50, 'rf__max_features': 'sqrt', 'rf__n_estimators': 1}
0.638 (+/-0.001) for {'rf__max_depth': 50, 'rf__max_features': 'sqrt', 'rf__n_estimators': 10}
0.699 (+/-0.004) for {'rf__max_depth': 50, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}
0.711 (+/-0.004) for {'rf__max_depth': 50, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}

Best roc_auc score: 0.734
Best parameters set:
        rf__max_depth: 10
        rf__max_features: 'sqrt'
        rf__n_estimators: 100
Top 2 GridSearch results: (roc_auc, hyperparam Combo)
 ({'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 100}, 0.7336096338317161)
 ({'rf__max_depth': 10, 'rf__max_features': 'sqrt', 'rf__n_estimators': 50}, 0.7328864644575827)
```

ML Pipeline

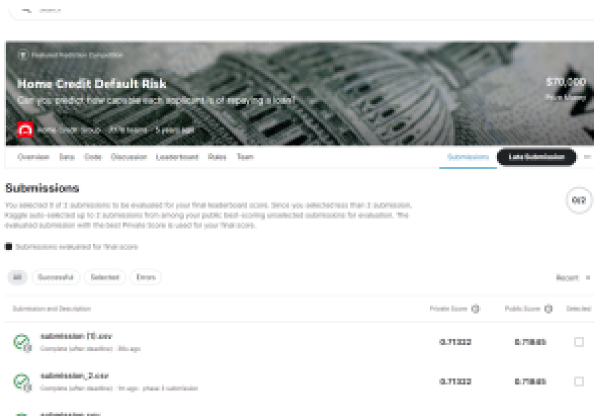

HCDR Data
EDA And Visualization
Preprocessing of Data
Training Data
Best Param Selection
Logistic Regression Model
Random Forest Classifier
Decision Tree Model
Model Score
Final Model
Test Data


Pipeline
preparation: FeatureUnion
num_pipeline
cat_pipeline
DataFrameSelector
DataFrameSelector
SimpleImputer
SimpleImputer
StandardScaler
OneHotEncoder
RandomForestClassifier

Results and discussion of results

- We can see that as we trained our data on the newly created features the AUC dropped, mainly because of a reduction of features
- However we have found that Random Forest Classifier Performs the best on this set

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
4	Best_Param_Decision_Tree	0.9200	0.9164	0.9194	0.7106	0.7005	0.7012
1	Best_Param_Logistic_Reg	0.9200	0.9164	0.9195	0.7290	0.7314	0.7295
2	Best_Param_Random_Forest	0.9202	0.9164	0.9194	0.8013	0.7371	0.7334
3	Best_Param_Decision_Tree	0.9200	0.9164	0.9194	0.7106	0.7005	0.7012



	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
0	Baseline_Logistic_Regression	0.9199	0.9165	0.9193	0.7534	0.7542	0.7511
1	Baseline_Logistic_Regression	0.9201	0.9167	0.9195	0.7556	0.7569	0.7521
2	Baseline_Logistic_Regression	0.9201	0.9167	0.9194	0.7556	0.7569	0.7521
3	FE_Baseline_Logistic_Regression	0.9200	0.9165	0.9195	0.7293	0.7318	0.7296

4Ps: Past, Present, Planned, Problems

- Past: Context of the Project
  - The HCDR Project aims to predict if customers of Home Credit are able to repay a loan without a recorded credit score or agreed upon metric.
- Present: What we have done
  - Done Feature analysis, engineering, selection based on correlation
  - Hyperparameter tuning with gridsearch over our algorithms
- Planned:
  - Dig deeper into the RandomForestClassifier Algorithm and use neural networks in our final submission
- Problems:
  - Final training on the larger dataset with additional features.

Conclusion and next steps

- In this phase we merged the datasets and extracted all possible features.
- We preprocessed the data and found the most relevant features to the Target variable and prediction.
- We then created our own features based on the most correlated features which contributed to the prediction.
- We featured the data performing OHE and applied imputing methods to fix the data before feeding it to the model.
- In our next phase, we will be implementing a deep learning model using PyTorch for classification and regression and also build a multi-headed load default system.