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:

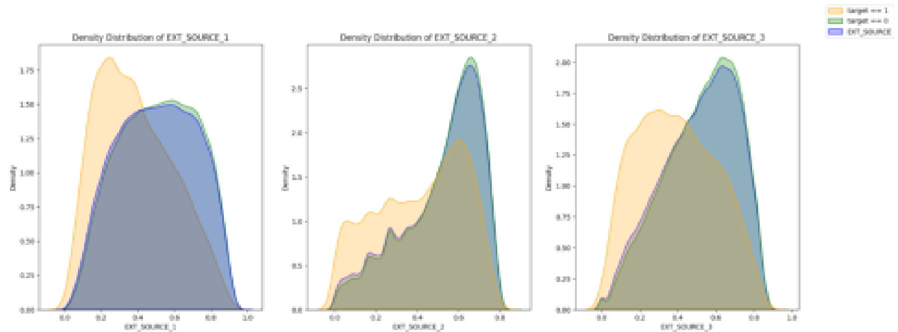
- Project Description
- Summary Visual EDA
- Feature Engineering and Top Features
- ML Pipeline
- Overview of Modeling
- Results and discussion of results
- Conclusion and next steps

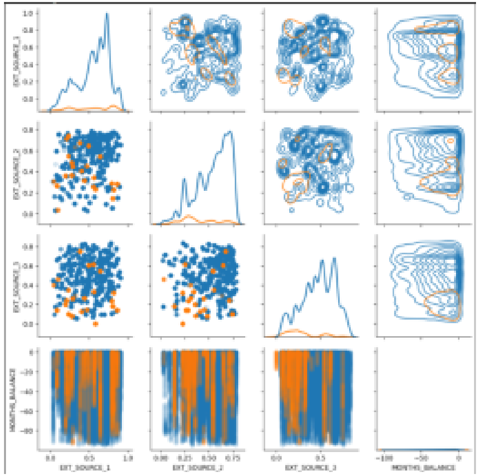
Project Description

- 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 use a Neural Network Model using Pytorch for loan default classification.
- Use Tensor board to visualize the results of training and modeling

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





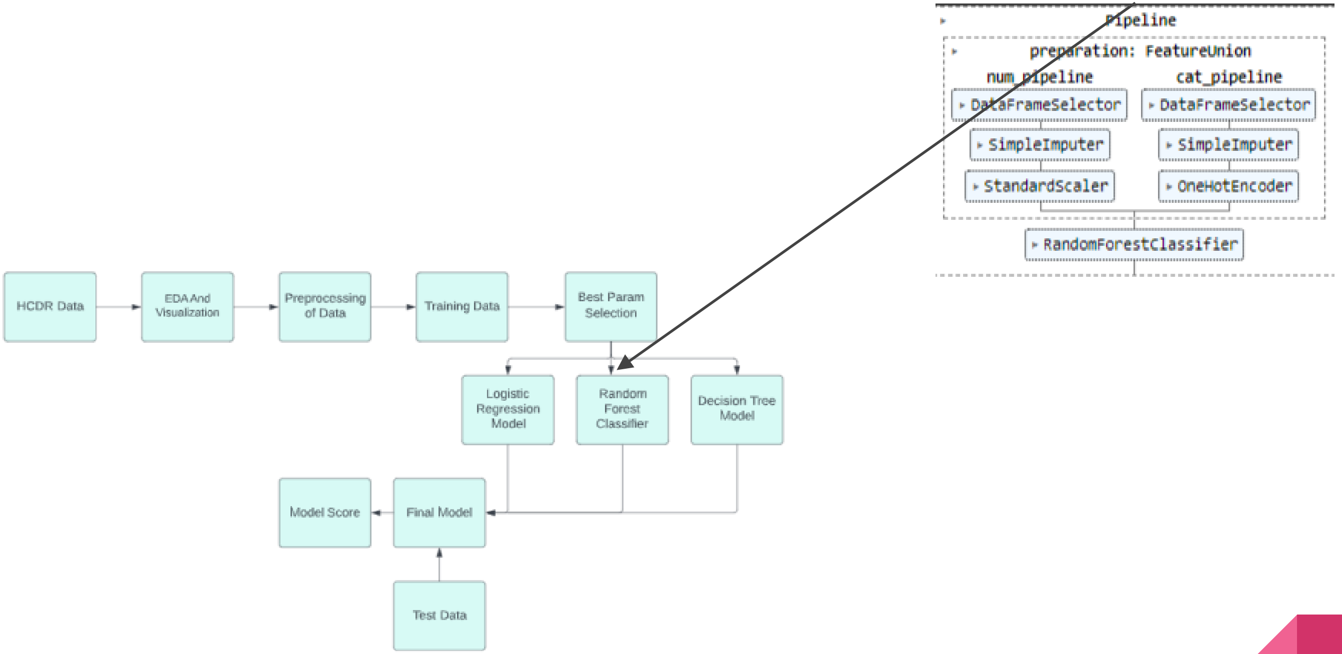
Feature Engineering

- 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

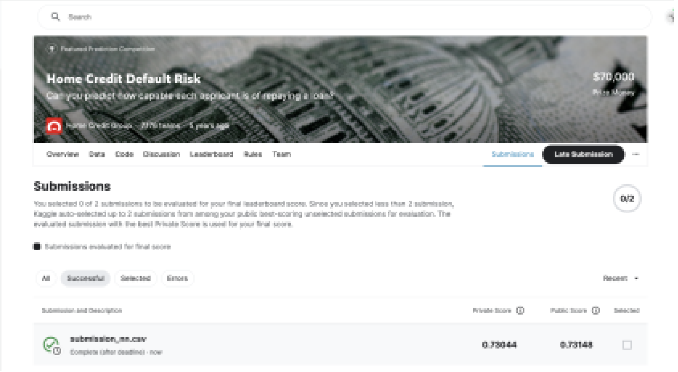
```
# Create Features for the Bureau and Bureau_balance
#-------------------------------------------------

## 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']
```

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
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

	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

	Experiment	Train BCE Loss	Train ROC AUC Score	Test BCE Loss	Test ROC AUC Score
0	Single Layer Neural Network Model 1	0.255943	0.793480	0.30699	0.735541
1	Single Layer Neural Network Model 2	0.259213	0.787107	0.30215	0.738270

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
  - Implemented Neural Network Model using Pytorch for loan default classification.
- Planned:
  - We will try to improve the accuracy and AUC score of model using Deep learning.
- Problems:
  - Final training on the larger dataset with additional features.

Conclusion and next steps

- In this last phase we implemented a NN model using Pytorch for loan default classification.
- Deep Learning models require huge amount of data to train itself and thus on a longer run Deep Learning models would work best for HCDR classification as compared to usual supervised models.
- The future scope for this project can include using embeddings in deep learning models or using some advanced classification models like lightGBM/other boosting models that can produce better results
- 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.