Home Credit Default Risk (HCDR)

Group 16
Aravind Sheru
Sai Charan Chintala
Seongbo Sim
Yun Joo An

Abstract

HomeCredit offers unsecured lending based on past credit history, repayment patterns and alternate data of the users using Machine Learning Modelling. Credit history is a measure explaining the credibility of a user generated using parameters like average/min/max balance maintained by the user, Bureau scores reported, salary etc and repayment patterns. As a part of this project, we use the datasets provided by kaggle to perform exploratory data analysis, build machine learning pipelines and evaluate the models across several evaluation metrics for a model to be deployed. In phase 2, we provide feature engineering, hyperparameter tuning, and modeling pipelines. We experimented with a regression of baseline inputs, selected features for Logistic regression, Decision Making Tree, Lasso, and Ridge Regressions. The baseline pipeline has the highest test accuracy with 92, followed by Logistic regression with 91.98, then Decision Making tree, and finally Lasso and Ridge being the least accurate.

Data and Task Description

1. Data source

We are planning to use the existing datasets provided by Kaggle.
Source: https://www.kaggle.com/c/home-credit-default-risk/data

POS_CASH_balance.csv
This dataset gives information about previous credits information such as contract status, number of installments left to pay, DPD(days past due), etc… of the current application

bureau.csv
This dataset gives information about type of credit, debt, limit, overdue, maximum overdue, annuity, remaining days for previous credit, etc…

bureau_balance.csv
This dataset gives information about Status of Credit Bureau loan during the month, Month of balance relative to application date, Recoded ID of Credit Bureau credit

credit_card_balance.csv
This dataset gives information about financial transactions aggregated values such as amount received, drawings, number of transactions of previous credit, installments, etc…

installments_payments.csv
This dataset gives information about payments, installments supposed to be paid and their details.

previous_application.csv
This dataset contains information about previous application details of an applicant.

2. **Train dataset in application.csv**

- Shape: (307511, 122)
- First five rows and seven columns look like:

Table 1. Train dataset

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
100002	1	Cash loans	M	N	Y	0
100003	0	Cash loans	F	N	N	0
100004	0	Revolving loans	M	Y	Y	0
100006	0	Cash loans	F	N	Y	0
100007	0	Cash loans	M	N	Y	0

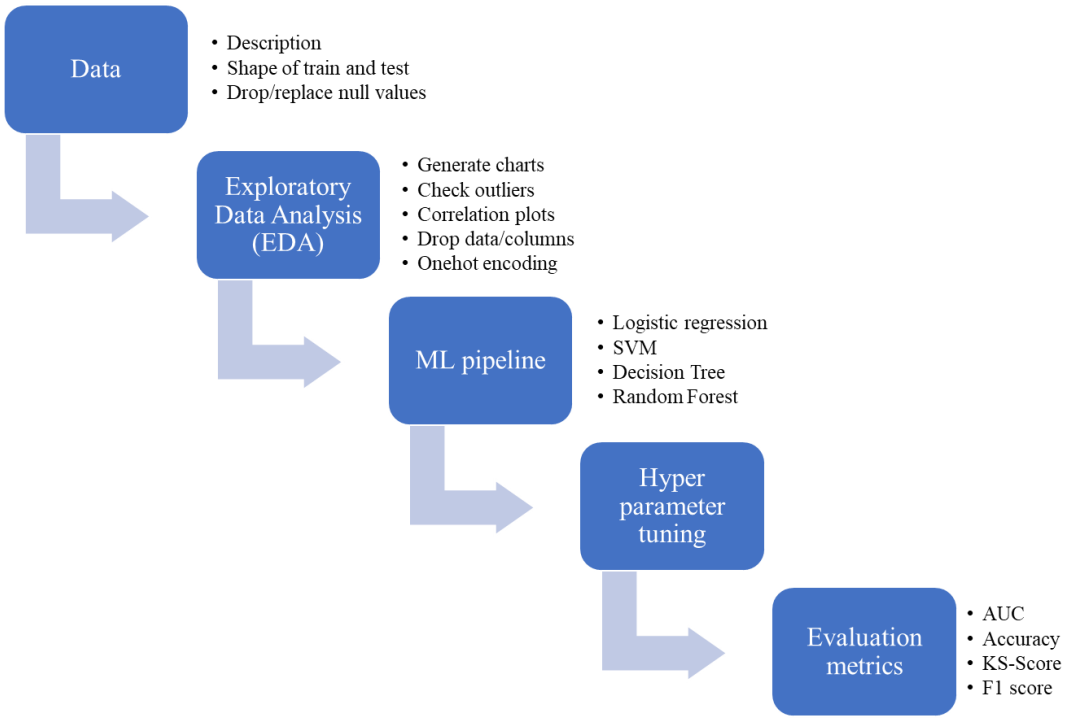
- Wed discarded features with null values more than 30%. We present null percentage of ten features in table 2 as an example.

Table 2. Chart on nullity/ missing values by column.

	col_name	null_count	count_%
0	NAME_CONTRACT_TYPE	0	0.000000
1	CODE_GENDER	0	0.000000
2	FLAG_OWN_CAR	0	0.000000
3	FLAG_OWN_REALTY	0	0.000000
4	CNT_CHILDREN	0	0.000000
...	...	...	...
115	AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631
116	AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631
117	AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631
118	AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631
119	AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631

3. Diagram of workflow

Figure 1. Diagram of workflow



Feature Engineering

	col_name	null_count	count_%	col_type
7	AMT_ANNUITY	12	0.003902	float64
8	AMT_GOODS_PRICE	278	0.090403	float64
27	CNT_FAM_MEMBERS	2	0.000650	float64
40	EXT_SOURCE_2	660	0.214626	float64
41	EXT_SOURCE_3	60965	19.825307	float64
89	OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
90	DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
91	OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
92	DEF_60_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
93	DAYS_LAST_PHONE_CHANGE	1	0.000325	float64
114	AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.501631	float64
115	AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631	float64
116	AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631	float64
117	AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631	float64
118	AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631	float64
119	AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631	float64

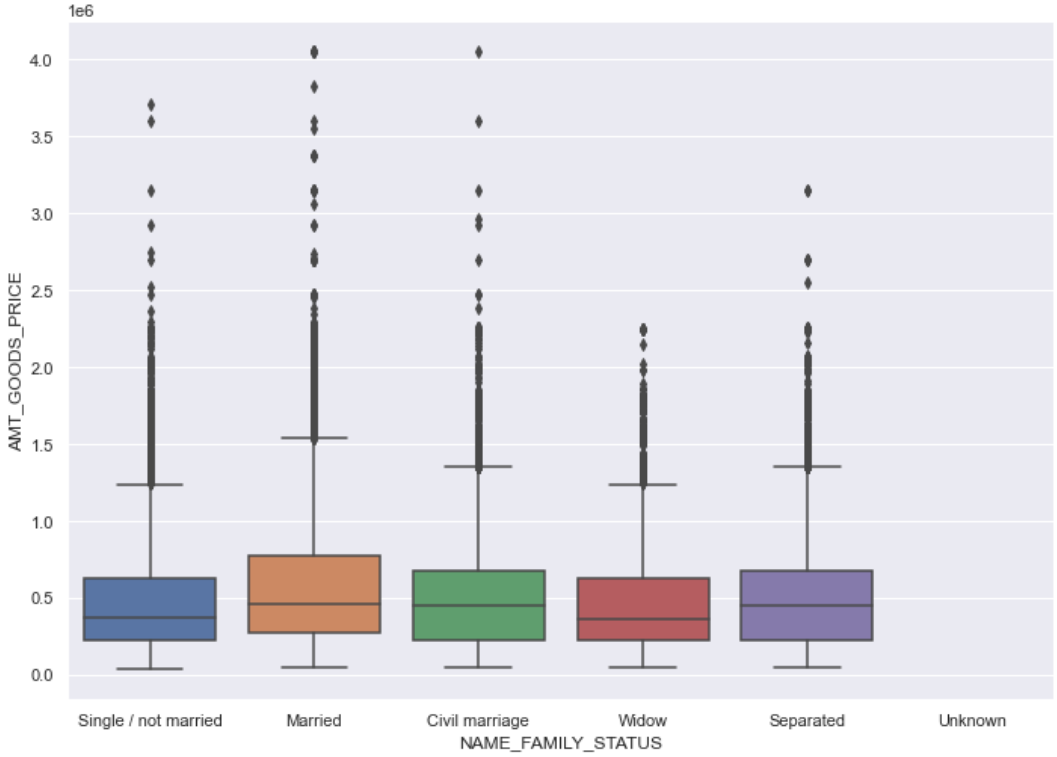
Step 1: - We discarded columns which are having more than 30% of Null Values
The above table shows the count of NA values of remaining columns and their percentages

Step 2: - AMT_REQ_CREDIT_BUREAU_HOUR, AMT_REQ_CREDIT_BUREAU_DAY, AMT_REQ_CREDIT_BUREAU_WEEK, AMT_REQ_CREDIT_BUREAU_MON, AMT_REQ_CREDIT_BUREAU_QRT, AMT_REQ_CREDIT_BUREAU_YEAR gives the number of enquiries done. As the number is not available, we can assume no enquiries are made. So, we replace NA values with 0

Step 3: - OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE, OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE gives us number of immediate connections who have a loan in Home Credit. Since we don't have data, we can assume there are no immediate connections. So, we replace NA values with 0.

Step 4: - CNT_FAM_MEMBERS NA values are filled with median

Step 5: - AMT_GOODS_PRICE values are depending on NAME_FAMILY_STATUS categories. So we replaced NA values with medians w.r.t NAME_FAMILY_STATUS. We can see in below figure that AMT_GOODS_PRICE is depends on NAME_FAMILY_STATUS.



Step 6: - Dropped DAYS_LAST_PHONE_CHANGE column because there is only 1 row.

Step 7: - To replace EXT_SOURCE_2 NA values, we found top 5 variables which are highly correlated. REGION_RATING_CLIENT is highly correlated with EXT_SOURCE_2. As REGION_RATING_CLIENT is categorical, we fill NA values with median based on categories.

Step 8: - To replace EXT_SOURCE_3 NA values, we found top 5 variables which are highly correlated. DAYS_BIRTH is highly correlated with EXT_SOURCE_2. As DAYS_BIRTH is numerical, we fill NA values using Linear Regression.

Provide additional features added to training data

- We trained and tested with selected columns.
    - AMT_CREDIT_TO_ANNUITY_RATIO
    - Tot_EXTERNAL_SOURCE
    - Salary_to_credit
    - Annuity_to_salary_ratio.
- Additionally we tried several other features like segregated bins based on AMT_ANNUITY, AMT_SALARY based on percentile, but these features were not of help for us either in increasing accuracy or AUC.
- We also tried using one-hot encoding instead of label encoding and performed modelling across Baseline and checked the results, but we found no improvement across AUC or accuracy.

Impact that these new features added to the model

- Then, we ran Logistic regression with these selected features and compared it with the baseline logistic regression model, as presented in the modeling session.
- The addition of these features to the training data have made an impact as presented in Table 3.

Table 3. Baseline Logistic regression before and after

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...

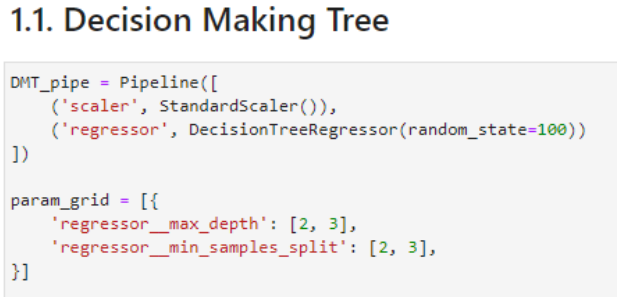
Why we chose the method and approach

- Filling null values across categorical and continuous variables should be handled separately. It won't be feasible to fill all categorical variables with most_repeated or least repeated values and fill with either 0's or median values across continuous variables.
- Ratios across income/credit requested/credit to be paid per year would be a good measure to judge an individual's credibility and repayment ability. So we considered adding above

Hyperparameter Tuning

- After Feature Engineering, we tuned our model to find the optimal parameters. As we will further discuss, we will use Decision Making Tree in addition to baseline and selected Logistic Regression.
- The hyperparameter tuning for grid search was conducted for decision Making Tree, Lasso Regression, Ridge Regression, and Logistic Regression.

Figure 2. Hyperparameter Tuning



- For a Decision Tree model, we used different maximum depths and the number of sample split.

1.2. Lasso Regression

```
lasso_pipeline = Pipeline([
                    ('scaler',StandardScaler()),
                    ('model',Lasso())
])


lasso_pipeline = GridSearchCV(lasso_pipeline,
                     {'model__alpha':[0.0001,0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10]},
                     cv = 5, scoring="neg_mean_squared_error",verbose=3
                     )
```

- For Lasso Regression, we used different alpha parameters and controlled the weighting of the penalty to the loss function.

1.3. Ridge Regression

```
ridge_pipeline = Pipeline([
                    ('scaler',StandardScaler()),
                    ('model',Ridge())
])


ridge_pipeline = GridSearchCV(ridge_pipeline,
                     {'model__alpha':[0.0001,0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10]},
                     cv = 5, scoring="neg_mean_squared_error",verbose=3
                     )
```

- For Ridge Regression, we used different alpha parameters and controlled the weighting of the penalty to the loss function.

1.4. Logistic Regression

```
logistic = LogisticRegression(max_iter=10000, tol=0.1)
clf_pipe = Pipeline(steps=[("logistic", logistic)])
param_grid = {
    "logistic__C": np.logspace(-4, 4, 10)
}
```

- For Logistic Regression, we used different C parameters and controlled the penalty strength.

Results for Modeling Pipeline

1. Logistic Regression

- In table 3, results for baseline and selected logistic regressions have been presented.
- Baseline logistic regression and selected logistic regression have similar test accuracies and AUC. They have similar cross fold train accuracy as well.
- Since baseline logistic regression already has fairly high test accuracy i.e. 92 and decent AUC 0.504, there seems no significant advantage for selecting features.

2. Decision Making Tree

Table 4. Decision Tree

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...

- In table 4, results for the decision tree have been presented.
- Decision tree has low test accuracy i.e. 7.56% and high AUC 0.738.
- Compared to baseline regressions, the decision tree lost test accuracy but gained AUC.
- This loss of test accuracy may be due to the short dept of the decision tree, compared to the number of variables we have used.

3. Lasso Regression

Table 5. Lasso Regression

:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection

- In table 5, results for the lasso regression have been presented.
- Test accuracy has decreased to -6.89%, whereas AUC increased to 0.755.
- Although AUC has increased, since test accuracy has significantly fallen, we should not use Lasso regression.
- We believe that these negative results could be due to score functions in the lasso regression blocks.

4. Ridge Regression

Table 6. Ridge Regression

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection
4	Ridge Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.756776	30.4069	0.0134	Ridge Regression for feature selection

- In table 6, results for ridge regression have been presented.
- Test accuracy has decreased to -6.89%, whereas AUC increased to 0.756.
- Although AUC has increased, since test accuracy has significantly fallen, we should not use Ridge regression.
- We believe that these negative results could be due to score functions in the ridge regression blocks.

5. Logistic Regression GridSearch

Table 7. Logistic Regression GridSearch

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection
4	Ridge Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.756776	30.4069	0.0134	Ridge Regression for feature selection
5	Gridsearch LogReg with 79 inputs	91.91%	91.98%	91.96%	0.500000	55.1705	0.0168	LogReg GridSearch with selected features

- Results for Logistic Regression GridSearch have been presented.
- Test accuracy is as high as 91.98%, but AUC is slightly lower than the baseline regression.
- AUC for logistic regression gridsearch is 0.5, whereas that of baseline is 0.504.
- Since test accuracy and AUC are both higher for baseline regression, we should choose baseline logistic regression instead of gridsearch logistic regression.

Conclusion

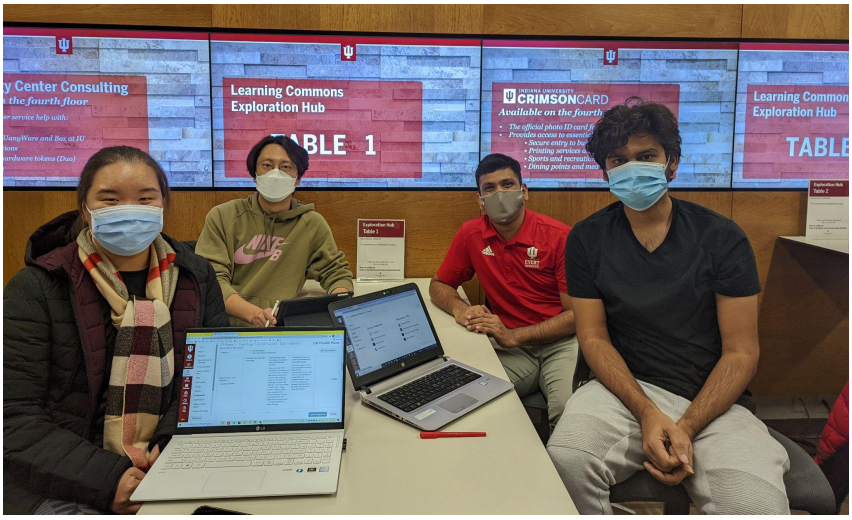
The objective of the HCDR project is to predict the repayment ability of the financially under-served population. This project is important because well-established predictions are necessary to both the loaner and borrower. Real-time Homecredit is able to display loan offers to its customers with the maximum amount and APR using their ML pipelines where fetching data from the data providers via APIs, performing EDA and fitting it to the model to generate scores occurs in microseconds of time. Hence Risk analysis becomes very critical in this regard where NPA(Non-Performing Asset) expected is less than 5% in order to run a profitable business.

Credit history is a measure explaining the credibility of a user generated using parameters like average/min/max balance maintained by the user, Bureau scores reported, salary etc and repayment patterns could be analysed using the timely defaults/repayments made by the user in the past. Alternate data includes other parameters like geographic data, social media data, calling/SMS data etc. As part of this project we would be using the datasets provided by kaggle to perform exploratory data analysis, build machine learning pipelines and evaluate the models across several evaluation metrics for a model to be deployed.

In phase 2, we estimated several models including both classification and regression models. We did feature selection, data imputation, and hyperparameter tuning. First, we did feature selection and imputation. We filled in the missing values of selected features. Then, we decided to add relevant features based on our prior knowledge. Next, we tuned the hyperparameters with the help of GridSearchCV. To find the best model, we trained and evaluated several models like Logistic Regression, Decision Tree Model, Lasso Regression and Ridge Regression. In phase 2, classification models cannot win the baseline model. Among regression models, the ridge regression model shows the best performance.

In phase 3, we plan to implement a deep learning model and build additional models in PyTorch. The problems that we are currently facing are that with the feature selection and imputation step, we cannot improve the test accuracy or AUC of the baseline model. Unlike regression models, we cannot develop a classification model better than the baseline. So, to address these issues, in phase 3, we plan to build a multilayer model in PyTorch for loan default classification. As a stretch goal, we will develop and implement a new multitask loss function in Pytorch. These will be submitted to Kaggle and we will report our scores.

Team photo



Individual Profile

Aravind Reddy Sheru
Email: asheru@iu.edu

Sai Charan Chintala
Email: sachin@iu.edu

Seongbo Sim
Email: simseo@iu.edu

Yun Joo An
Email: yunjooan@iu.edu