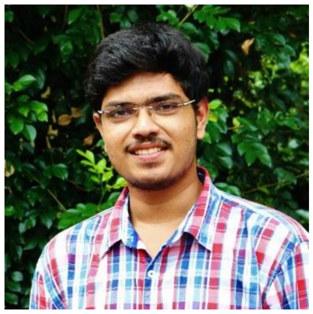
1. Abstract:

Home Credit Default Risk is a project where we determine the credit worthiness of people that have applied for the loans. In previous phases, we had completed basic EDA, Feature Engineering and ran the baseline model for logistic regression and the hyperparameter tuning for XGBoost model.
In this Phase, we have significantly improved our project. We have updated the EDA, implemented robust Feature engineering for all dataset files, and did experimental analysis for hyper-parameter tuning for Logistic Regression, XGBoost and Random Forest Models. We conducted experiments using both original imbalanced data as well as resampled data. After comparison we found out that the XGBoost model with parameters: { learning_rate: 0.1, max_depth=5, min_child_weight=3 } was the best model, using the model performance criteria of Test AUC Score (0.7427).
For the deep learning Pytorch model, we used a feed-forward MLP with two hidden layers of 128 and 64 neurons each. We used a sigmoid activation function and SGD optimizer with cross entropy as loss function. The model achieved the test accuracy of 40%. The best Kaggle submission that we obtained was from XGBoost model with the Private Score 0.64788 and Public Score of 0.65231.

Team Members:

Team Information

Bhushan Patil	**Vaibhav Vishwanath**	**Gavin Henry Lewis**	**Prathamesh Deshmukh**
bpatil@iu.edu	**vavish@iu.edu**	**gavlewis@iu.edu**	**pdeshmukh@iu.edu**
			

2. Project Description

Part A: Data Description

Background Home Credit Group

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

Home Credit Group

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Background on the dataset

Home Credit is a non-banking financial institution, founded in 1997 in the Czech Republic.

The company operates in 14 countries (including United States, Russia, Kazahstan, Belarus, China, India) and focuses on lending primarily to people with little or no credit history which will either not obtain loans or became victims of untrustworthly lenders.

Home Credit group has over 29 million customers, total assests of 21 billions Euro, over 160 millions loans, with the majority in Asia and and almost half of them in China (as of 19-05-2018).

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data files overview

There are 7 different sources of data:

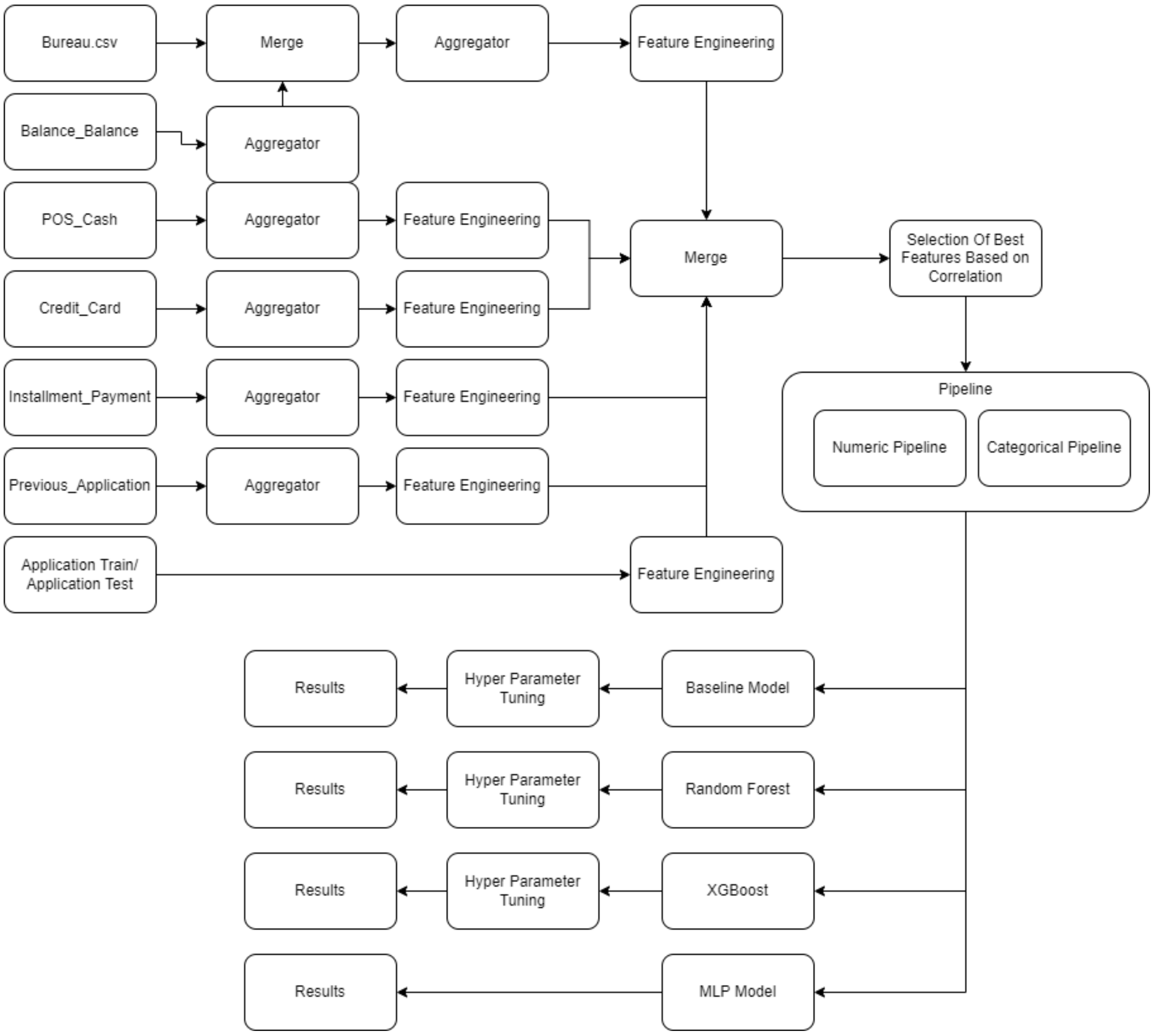
- **application_train/application_test:** the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating **0: the loan was repaid** or **1: the loan was not repaid**. The target variable defines if the client had payment difficulties meaning he/she had late payment more than X days on at least one of the first Y installments of the loan. Such case is marked as 1 while other all other cases as 0.

- **bureau:** data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- **bureau_balance:** monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- **previous_application:** previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- **POS_CASH_BALANCE:** monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.
- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.
- **installments_payment:** payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

Part B: Tasks to be tackled during this phase

1. Robust Feature Engineering on all data files
2. Exploring different models with hyperparameter tuning and selection of best model based on performance metrics such as AUC and F1.
3. Optimization of input data for efficient memory usage and avoiding kernel crashes.
4. Implementation of MLP model and tensorboard for visualization of model training.
5. Kaggle submission for both the best model and the MLP model.

Workflow Diagram For entire Project



Neural networks

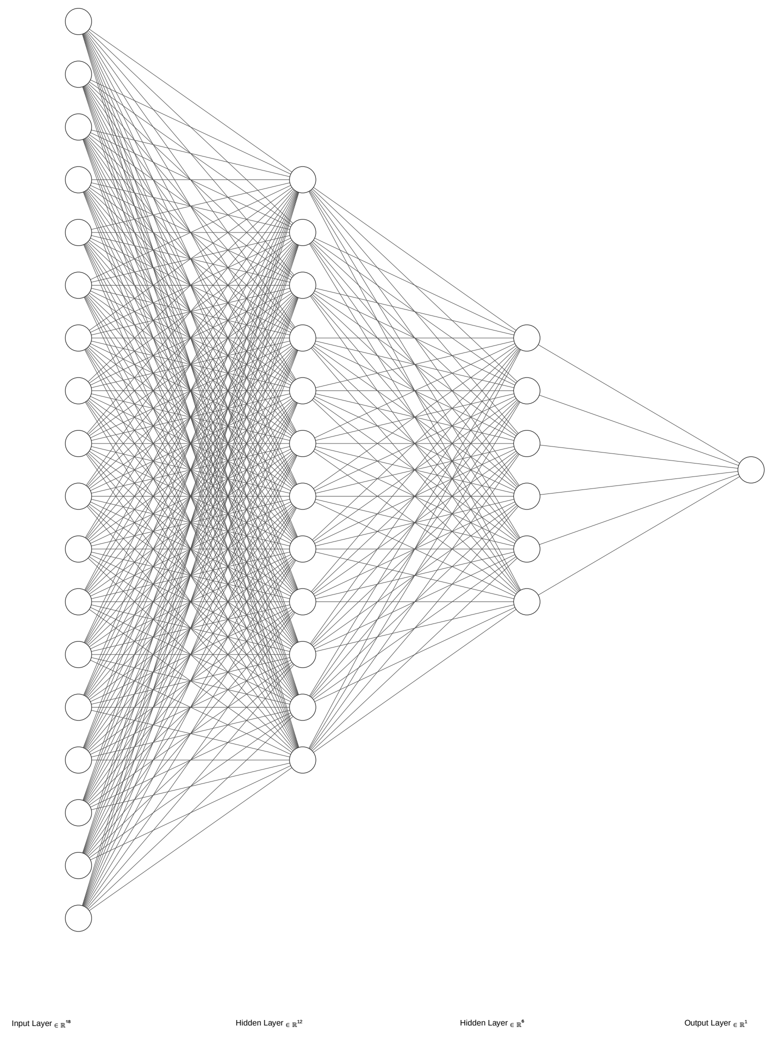

Input Layer $\in \mathbb{R}^{18}$
Hidden Layer $\in \mathbb{R}^{12}$
Hidden Layer $\in \mathbb{R}^{6}$
Output Layer $\in \mathbb{R}^{1}$

Fig: Sample visualization of Neural Network Architecture Designed

Architecture Of Neural Network:

The neural network has 4 layers:

Layer 1: 188 neurons (Input Layer)

Layer 2: 128 neurons	Activation function: Sigmoid
Layer 3: 64 neurons	Activation function: Sigmoid

Layer 4: 1 neuron (Output Layer)   Activation function: Sigmoid

Hyper Parameters:

Learning Rate= 0.01

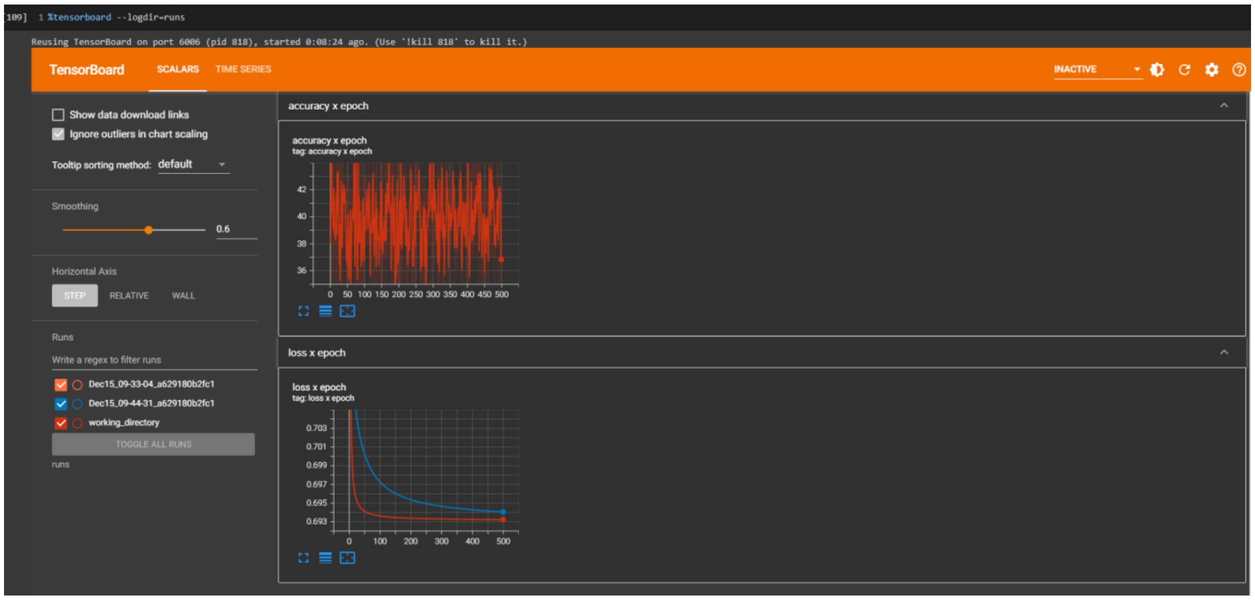
Batch Size =400

Optimizer= SGD

Loss Function= Cross Entropy Loss (BCEwithLogitsLoss())

We trained the model for 500 epochs and the training loss converged to 0.69 starting with 0.74. The accuracy was fluctuating in the range of 30-50. We achieved the test accuracy of 40% on the trained model. Since the model was run only for resampled balanced data with only 50,000 training samples. This is why were are getting lower accuracies, and found out that the traditional algorithms were performing better.

Tensor board Dashboard for training visualization:



```
[109]  %tensorboard --logdir-runs
Reusing TensorBoard on port 6006 (pid 818), started 0:08:24 ago. (Use '!kill 818' to kill it.)
```

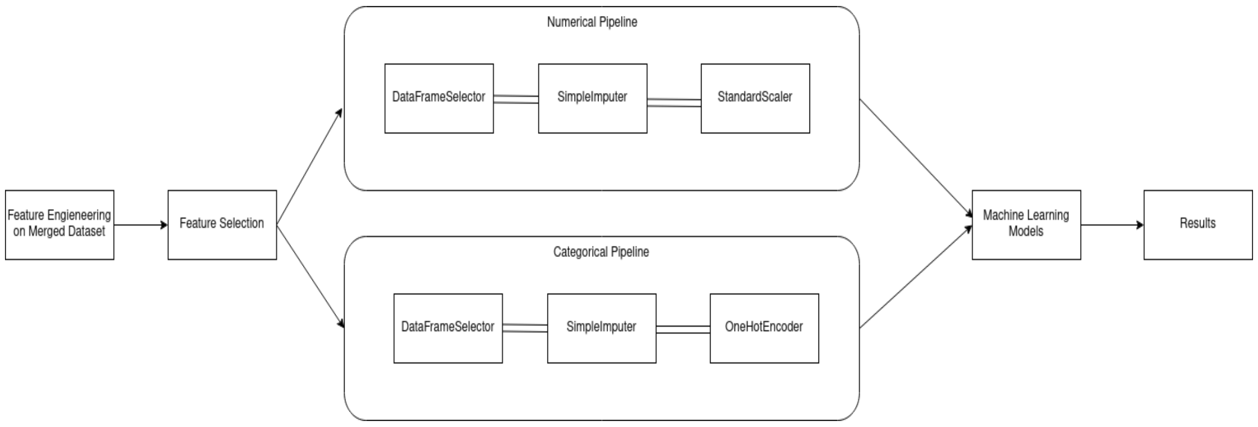
[ ] Show data download links
[✓] Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing: 0.6
Horizontal Axis: [✓ STEP] RELATIVE WALL
Runs
Write a regex to filter runs
[✓] Dec15_09-33-04_a629180b2fc1
[✓] Dec15_09-44-31_a629180b2fc1
[✓] working_directory

Data Leakage Control (HCDR):

We performed feature engineering on secondary datasets and merged with application train and application test. For feature selection from application train, we selected top 50 numeric features based on the correlation with target variable. All the numeric features as well as the engineered features were part of the final dataset.

We used pipelines to avoid the data leakage during preprocessing of numeric and categorical features. All the models were built with data pipeline + estimator as a single pipeline for Cross Validation purposes.

Modelling Pipelines:



Families of Input Features:

Inputs used for the modelling primarily consists of following:

Engineered Features:

Total count of engineered features is 24. This is the set of input features derived from data files other than application_train. The top 5 most correlated features are:

1. cc_LIMIT_USE: total used credit as a percentage of total limit
2. cc_DRAWING_LIMIT_RATIO: Ratio of total drawings of the month with credit limit amount
3. ins_INSTALMENT_PAYMENT_RATIO: ratio of actual installment payment and prescribed installment amount
4. ins_LATE_PAYMENT_RATIO : Installment payment ratio of late payments
5. bur_UPDATE_DIFF: Difference between the credit end date and last update date

Numeric Features:

We have considered top 50 correlated numeric features. These are the set of numeric features from application_train as well as aggregated features from secondary datasets. Top 5 highly correlated features are:

1. EXT_SOURCE_3
2. EXT_SOURCE_2
3. cc_CNT_DRAWINGS_ATM_CURRENT_sum
4. cc_CNT_DRAWINGS_ATM_CURRENT_mean
5. cc_CNT_DRAWINGS_ATM_CURRENT

Categorical Features:

We considered the categorical features from the application train dataset, the total count of categorical features is 16.

Hyperparameter Settings:

Logistic Regression

```
Cslist = [10,100]
regulars = ['l1','l2']
solvers=['liblinear']
paramsgrid = {'linear__C':Cslist,
        'linear__penalty': regulars,
        'linear__solver': solvers}
```

XGBoost

```
params={
 "xgb__learning_rate"    : [0.05, 0.10] ,
 "xgb__max_depth"        : [ 3,5],
 "xgb__min_child_weight" : [ 1, 3, 5]
}
```

Random Forest:

```
param_grid = {
    'rmf__bootstrap': [True],
    'rmf__max_depth': [10, 20],
    'rmf__max_features': [2, 3],
    'rmf__n_estimators': [100, 200]
}
```

MLP:

Learning Rate= 0.01

Batch Size =400

Optimizer= SGD

Loss Function= Cross Entropy Loss (BCEwithLogitsLoss())

Loss Functions:

L1 Regularization

$\sum_{i=0}^{N}(y_i-\sum_{j=0}^{M}x_{ij}W_j)^2 + \lambda\sum_{j=0}^{M}|W_j|$

L2 Regularization
$\sum_{i=0}^{N}(y_i-\sum_{j=0}^{M}x_{ij}W_j)^2 + \lambda\sum_{j=0}^{M}W_j^2$

Experiments Conducted:

1.**Baseline Experiment:**

Baseline model is a logistics regression model with no hyperparameter tuning to establish a baseline. Baseline model performed with the accuracy of 91.93%,

exp_name	Train Acc	Test Acc	Train AUC	Test AUC
Baseline Model	0.9193	0.9194	0.7418	0.7432

2.Logistic Regression model with GridsearchCV:

We created a modelling pipeline for preprocessing + logistic regressor, this was then passed on to a GridSearchCV with a set of hyperparameters and CV=5. Test score was calculated on the best parameters.

Best parameters:

The best parameters decided by the GridSearchCV for Cross Validation= 5 are

{'linear__C':100, 'linear_Penalty': 'l2', 'linear__solver': 'liblinear'}

Model score for different evaluation criterion

exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
Logreg crossvalidation best	0.9192	0.9193	0.7421	0.7431	0.8828	0.8828

Model did not perform particularly well with unbalanced label(Target=1)

3.XGBoost model with GridsearchCV:

XGboost model was created using the same data processing pipeline with XGBClassifier as a estimator. Model was run for the set of hyperparameters and then the test score was calculated using the best parameters given by the GridsearchCV

Hyperparamete

"xgb__learning_rate" : [0.05, 0.10] ,

"xgb__max_depth" : [3,5],

"xgb__min_child_weight" : [1, 3, 5]

Best Parameters:

{'xgb__learning_rate': 0.1, 'xgb__max_depth': 5, 'xgb__min_child_weight': 3}

exp_name	**Train Acc**	**Test Acc**	**Train AUC**	**Test AUC**	**Train F1**	**Test F1**
XGBoost crossvalidation best	0.92	0.9199	0.7652	0.7466	0.8833	0.883

This model too performed poorly on the unbalanced label(Target=1) but the results bested the Logistics regressor with the test set F1 score of 0.883

4.Random Forest model with GridsearchCV:

Random forest model was created using the same data processing pipeline with RandomForestClassifier as an estimator. Model was run for the set of hyperparameters and then the test score was calculated using the best parameters given by the GridsearchCV

Hyperparamete:

```
param_grid = {
    'rmf__bootstrap': [True],
    'rmf__max_depth': [10, 20],
    'rmf__max_features': [2, 3],
    'rmf__n_estimators': [100, 200]
}
```

Best parameters:

{'rmf__bootstrap': True,

'rmf__max_depth': 20,

'rmf__max_features': 3,

'rmf__n_estimators': 100}

exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
Random Forest crossvalidation best	0.9223	0.9193	0.9065	0.7105	0.8877	0.8806

Random forest regressor performed well on the training set but failed to show the similar performance on the test set, which indicates a slight overfitting.

Data sampling for balancing the target variable:

We performed data sampling using sklearn's resample module with the ratio of 2:3 for '1' and '0' target value. We re-ran all the models on best parameters on the new dataset and recalculated the score on evaluation criteria for test set.

Final Model Tuned:

Based on the AUC and F1 score on the test dataset, our final tuned model was the XGBoost model with following hyperparameters:

{'xgb__learning_rate': 0.1, 'xgb__max_depth': 5, 'xgb__min_child_weight': 3}

Results and Discussion:

Following table summarizes the model performances before and after data balancing:

exp_name	Train Acc	Test Acc	Train AUC	Test AUC	Train F1	Test F1
Logreg crossvalidation best	0.919	0.919	0.742	0.743	0.883	0.883
XGBoost crossvalidation best	0.920	0.920	0.765	0.747	0.883	0.883
Random Forest crossvalidation best	0.922	0.919	0.907	0.711	0.888	0.881
Logreg balanced best	0.795	0.797	0.743	0.743	0.833	0.834
XGB balanced best	0.792	0.790	0.762	0.744	0.831	0.829
RF balanced best	0.835	0.798	0.888	0.728	0.865	0.834

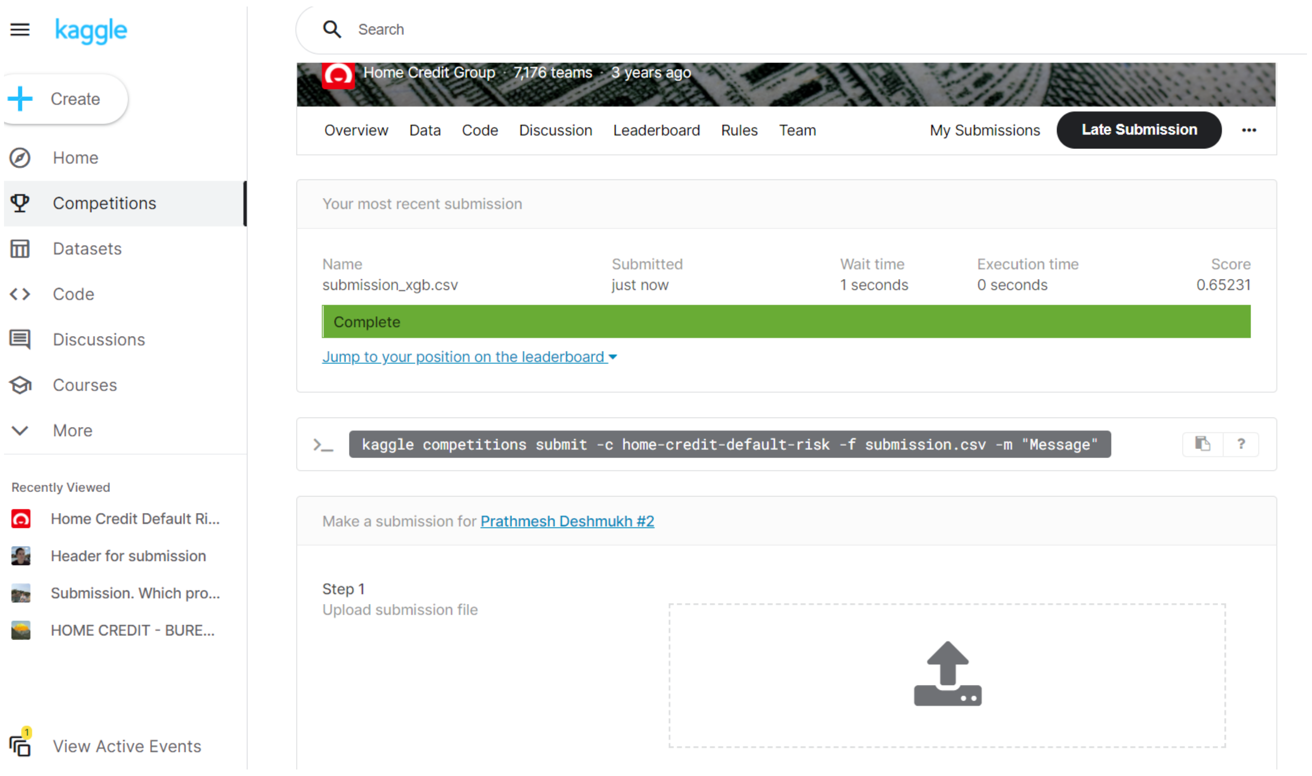
Loss based on models trained on balanced data:

exp_name	Train loss	Test loss
Logreg Log-loss	7.065	7.002
Randomforest Log-loss	5.713	6.982
XGB Log-loss	7.198	7.258

- In this phase, we have improved on the Phase 2 submission by improving our EDA and implementing additional feature engineering on all the secondary datasets.
- We also performed hyperparameter tuning on different models.
- We improved on the model evaluation criteria as accuracy was giving a false representation of the goodness of fit. We have used F1 score and AUC as our primary model evaluators
- We also implemented the Pytorch Deep Learning Model with 2 hidden layers.
- We also did Kaggle Submissions for the Random Forest, XGBoost and the Neural Network model.

Public Kaggle Scores

XGBoost	0.65
Random Forest	0.64
MLP	0.50



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission_xgb.csv	just now	1 seconds	0 seconds	0.65231

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c home-credit-default-risk -f submission.csv -m "Message"
```

Make a submission for Prathmesh Deshmukh #2

Step 1
Upload submission file

Conclusion:

The aim of the project is to determine individuals who are capable of repaying the loans. Our Machine Learning model is able to predict whether an individual should be given a loan or not on the basis of the applicant’s previous applications, credit bureau history, payment installments and other primary features such as sources of income, number of family members, dependents, etc.

All the machine learning models trained with imbalanced data performed poorly for target value 1. Model was retrained using resampled data and predictions for Target value 1 significantly improved as evident from the confusion matrices.

The Deep Learning Model which was created did not perform on par with the traditional machine learning models as we trained the model with a smaller subset of data (resampled).

Our best performing model was the XGBoost Model with Test AUC score of 74.7% and F1 score of 88.3%.The worst performing model was the MLP model.The best performing model had a Kaggle score of 65%.