Project Phase 3

Home Credit Default Risk(HCDR)
FP_GroupN_11

Luddy School of Informatics, Computing, and Engineering

Developed by-

Bhavya Mistry(brmistry@iu.edu)
Kumud Sharma(kumsharm@iu.edu)
Jaydeep Patel(jp157@iu.edu)
Kamna Chaudhary (kamchau@iu.edu)

Abstract

Using the "Home Credit Default Risk" dataset, we attempted to develop a machine learning model that forecasts a borrower's repayment capacity. To do this, we used application, demographic, and credit behavior history data to perform basic exploratory data analysis, and we created a baseline pipeline that also dealt with missing, aberrant, and uncommon data.

The metrics Accuracy, F1 Score, and ROC score, Confusion Matrix are among the ones we want to publish. In addition to Logistic Regression(L1-LibLinear), Logistic Regression(L1,L2(SGD)) Decision Tree Classifier, Random Forest, NaiveBayes Classifier, XGBoost Classifier , AdaBoost classifier and Gradient Boosting Classifier, we also used other baseline models.

The major objective of this phase is to carry out additional feature engineering, hyperparameter tuning, and feature selection relevant to the prediction of the target risk variable. Given 335 columns distinguished as 166 as numerical attributes and 169 as categorical attributes, we first had trouble combining data from several databases. We had to use a subset of the data (maximum = 10,000 rows) for training rather than the complete set, due to time-exceeding and resource overloading. We increased the number of columns as compared to previous Phase and out of 166 we took top 100 numerical correlated features and took top 10 correlated categorical features and dropped some columns which seems to no use such as: [email, number, etc.]. In the phase before, we used PolynomialFeatures Class to engineer certain features to obtain 3-degree features. The previous_application.csv file and bureau.csv file were used to build aggregated features in this phase based on the already-existing numerical and category data.
We have developed distinct pipelines for both categorical and numerical features ,we use the imputer to fill in the missing data, StandardScaler and Onehot Encoder to normalize the data. For Hyper-parameter Tuning we used following classifiers: Logistic Regression(L1-LibLinear), Logistic Regression(L1,L2(SGD)), Random Forest, XGBoost Classifier , AdaBoost classifier and Gradient Boosting Classifier. We performed the experimentation by using both GridsearchCV and RandomizedSearchCV out of both GridSearchCV provided better AUC-ROC score of 72.38% with AdaBoostClassifier (learning rate=0.1) and 'classifier__algorithm': 'SAMME.R'. Upon Kaggle submissions, AdaBoostClassifier was the model that performed the best, with an AUC public score of 71.6% and a private score of 70.4%.
Logistic Regression and XGBoost, our basic model, had the highest training accuracy of 91%. DecisionTreeClassifier, GaussianNB test accuracy were found to be 53.8%, 72.25%. After submitting to Kaggle, we received an AUC private score of 71.54% and a public score of 71.49%. Our tests revealed that XGBoost and Logistic Regression had the highest accuracy, both at 73.85%.

Introduction:

Using the "Home Credit Default Risk" dataset, the goal of this study was to create a machine learning model that forecasts a borrower's capacity to repay a loan. In the last phase we completed exploratory data analysis and built a foundational pipeline to manage missing, abnormal, and unusual data. We tested with a number of baseline models, including Logistic Regression, Decision Tree Classifier, Random Forest, NaiveBayes Classifier, and XGBoost Classifier.
In this phase, tasks were to perform additional feature engineering, hyperparameter tuning, feature selection, analysis of feature importance, and perform ensemble methods and non-ensemble methods.

Data Description:

The task to hand is to build a machine learning model that can predict if a client will repay the loan or not based on the credit history and many other features. The primary data for the task is stored in the csv files: "application_train.csv" and "application_test.csv". The training data contains 121 feature columns, which have categorical and numerical variables. The target column indicates whether the client repaid the loan or not and is present in training dataset.

Other than the primary dataset, there are six subsidiary datasets that contain credit history information for each client. These datasets include "bureau.csv," which contains any credit history for the customer prior to the application date, and "bureau_balance.csv," which includes data for the prior credits stated in the bureau dataset for each month in history. These datasets interact with one another and the application dataset to provide a complete picture of each client's credit history.

To build the machine learning model, we will first need to perform exploratory data analysis (EDA) to understand the relationships between the features and the target variable. We can use various diagrams, such as box plot, scatter plots, heatmaps, and histograms, to visualize the data and identify any patterns or correlations. Once we have a good understanding of the data, we can start pre-processing it by handling missing values, encoding categorical variables, and scaling numerical features.

Next, we will select a suitable machine learning algorithm. We can use cross-validation to evaluate the model's performance and fine-tune its hyperparameters to optimize its accuracy.

Finally, we can use the trained model to make predictions on the test dataset and submit them to the competition platform to see how well our model performs compared to other participants. Overall, this is a challenging task that requires a thorough understanding of the data and careful selection and optimization of machine learning algorithms.

1. application_{train|test}.csv

This dataset is the primary dataset and serves as the base for connecting with other datasets. It contains information on the loan applicants' personal characteristics and uses the SK_ID_CURR property as the primary key.

2. previous _application.csv

This dataset contains details regarding prior Home Credit loans made by the applicant. It includes characteristics such as loan status, down payment, and loan type. The SK_ID_CURR property from the application_{train|test}.csv dataset serves as the primary key for connecting with this dataset.

3. instalments_payments.csv

This dataset contains information on loan repayment from the past. It may be connected to the SK_ID_CURR property from the application_{train|test}.csv dataset or the SK_ID_PREV property from the previous_application.csv dataset, depending on the context.

4. credit_card_balance.csv

This dataset contains information on Home Credit credit card transactions. It may be connected to the SK_ID_CURR property from the application_{train|test}.csv dataset or the SK_ID_PREV property from the previous_application.csv dataset, depending on the context.

5. POS_CASH_balance.csv

This dataset describes the individual's past credit history at Home Credit, including personal loans and consumer credit. It may be connected to the SK_ID_CURR property from the application_{train|test}.csv dataset or the SK_ID_PREV property from the previous_application.csv dataset, depending on the context.

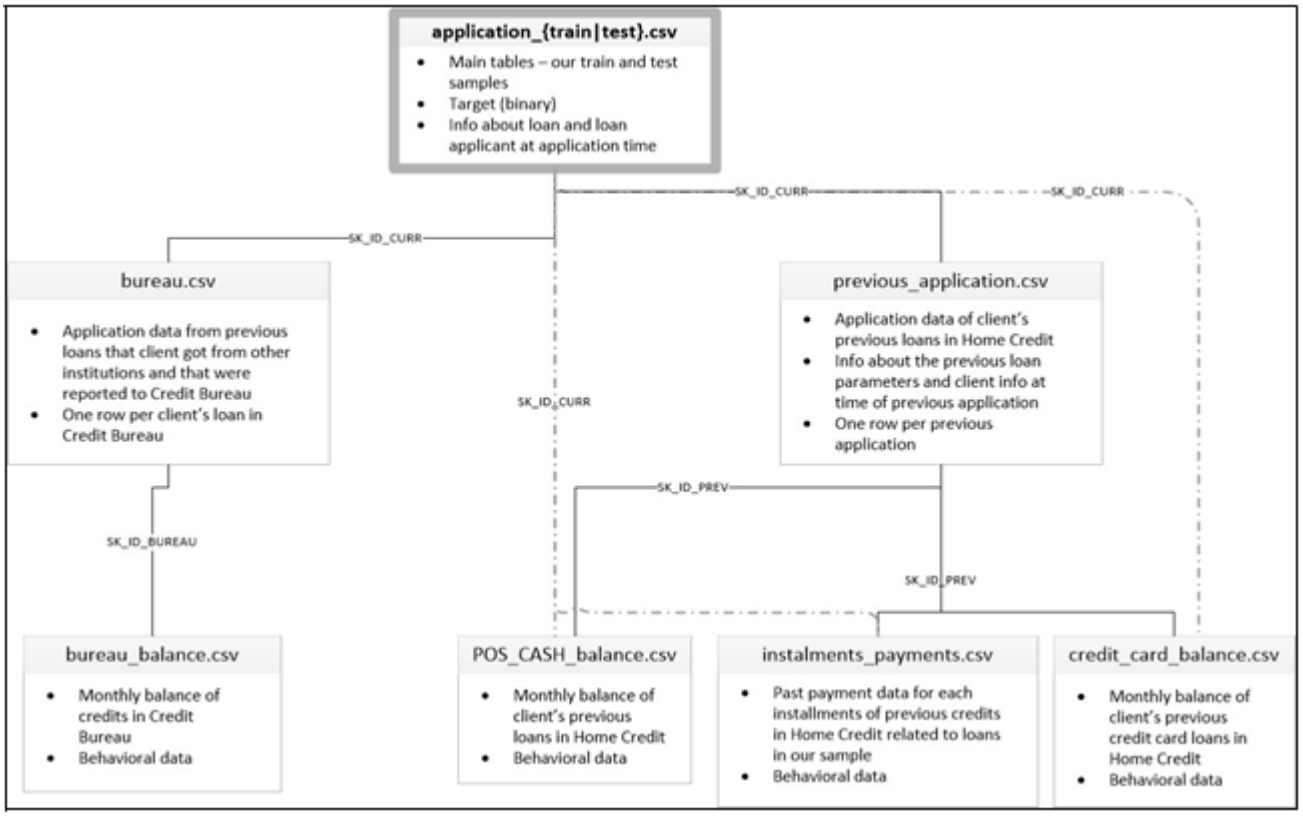
6. bureau.csv

This dataset contains details about a person's prior credit history at other financial institutions that were reported to a credit agency. It may be connected to the SK_ID_CURR property from the

application_{train|test}.csv dataset or the SK_ID_BUREAU property from the bureau.csv dataset, depending on the context.

7. Previous_application.csv

This dataset includes the credit bureau's monthly credit balance. It may be connected to the SK_ID_CURR property from the application_{train|test}.csv dataset or the SK_ID_BUREAU property from the bureau.csv dataset, depending on the context.



Experiments:

Models:

We are aiming to anticipate the client's ability to repay the loan, as explained in the abstract and data description module, and this ability is decided by a single binary output variable called TARGET (0|1). Under supervised learning, this is a binary classification issue because the goal has only two discrete possibilities that could occur. We will compare various categorization models using the chosen metrics to find the model that best fits the situation.
We will use Logistic Regression (with Lasso, Ridge, and No Regularization) as our basic model because this is a binary classification problem. Based on the Logistic Regression model, we will

evaluate the effectiveness of different models. The likelihood of various classes or clusters occurring in the dataset will be predicted using a Naive Bayes classifier. We will use the Decision Tree Classifier model by developing rules based on the application train.csv dataset, notably on the income and credit columns such as amt income total, amt credit, etc. We will also investigate Support Vector Machines (SVM), Random Forest Classifier, K-Nearest Neighbors, and XGBoost Classifier. Later on, we'll concentrate on dimensionality reduction and hyperparameter tweaking, setting the stage for a thrilling algorithmic race.
A brief introduction of the models is as below:

1. Logistic Regression - This statistical model type is employed to investigate the relationship between a binary or categorical dependent variable and one or more independent variables. To simulate the possibility that the dependent variable will, a logistic function is used. Any input is transformed by the logistic function into a value between 0 and 1, which represents the likelihood of the binary outcome. Log Loss is a tool used in logistic regression.This is only used in for numerical features.

2.**Decision Tree Classification** - Using the features of the input data, a tree-like model of decisions and possible outcomes is constructed in this model. In order to partition the data according to a specific criterion, such as Gini Impurity or Information Gain, the approach starts with the entire dataset and selects the best feature. Depending on the chosen characteristic, the data are then separated into subsets, and the operation is then repeated repeatedly for each subset until all of the data in each subset belong to the same class, or a stopping condition is satisfied.

3.**Random Forest Classification** - This model constructs a number of decision trees by randomly selecting features for each split and using subsets of the input data. Since each decision tree is trained using a bootstrapped sample of the data, it only sees a portion of the data. The outcome of the algorithm is determined by the majority vote of the decision trees.

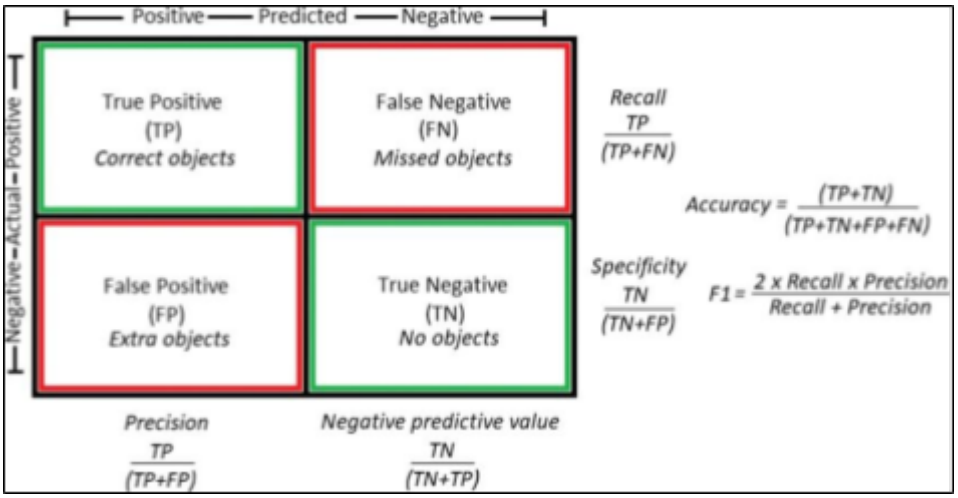
4. Neural Networks - Neural networks are a category of models that learn by varying the weights of the input data to produce a specific output. The input features are supplied into a network of linked neurons, which then manipulates the input using a number of mathematical operations. These changes are controlled by a set of weights that are developed during training. The weights are then modified using a loss function, such as cross-entropy loss, when the output is compared to the actual goal output.

$$BinaryCrossEntropy = H_p(q) = -\frac{1}{N}\sum_{i=1}^{n} y_i . log(p(y_i)) + (1 - y_i) . log(1 - p(y_i))$$

Metrics:

1. **Confusion Matrix:**

A confusion matrix is used to evaluate the effectiveness of a classification model, and the resulting matrix reveals the degree of classification accuracy and potential prediction mistakes for each record. It is used to gauge AUC-ROC curve, recall, accuracy, and precision.



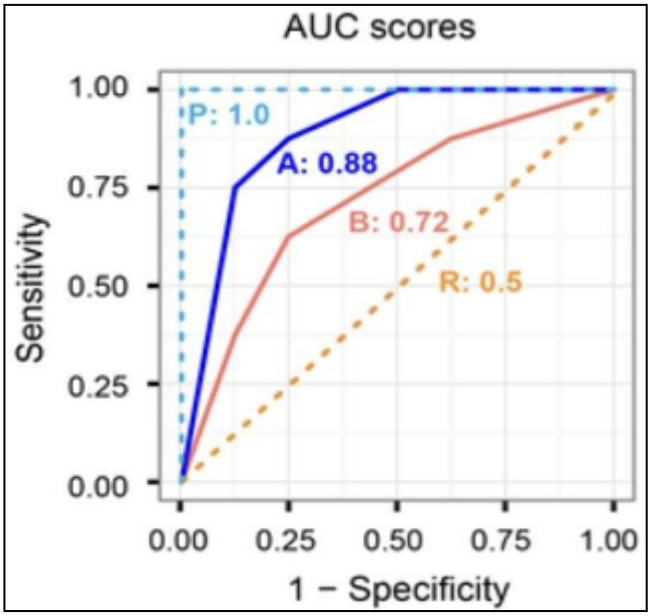
2. F-1 Score:

It will be used to compare the effectiveness of two classifiers based on their precision and recall values. It will be crucial in deciding which model fits the issue more effectively.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

3.AUC-ROC Curve:

When the actual outcome is positive, the ROC curve will assist us in calculating the likelihood of correctly predicting the positive class. Its performance is distilled into a single value using the AUC curve.



Pipelines

- We performed creation of two separate pipelines for both numerical and categorical features respectively.

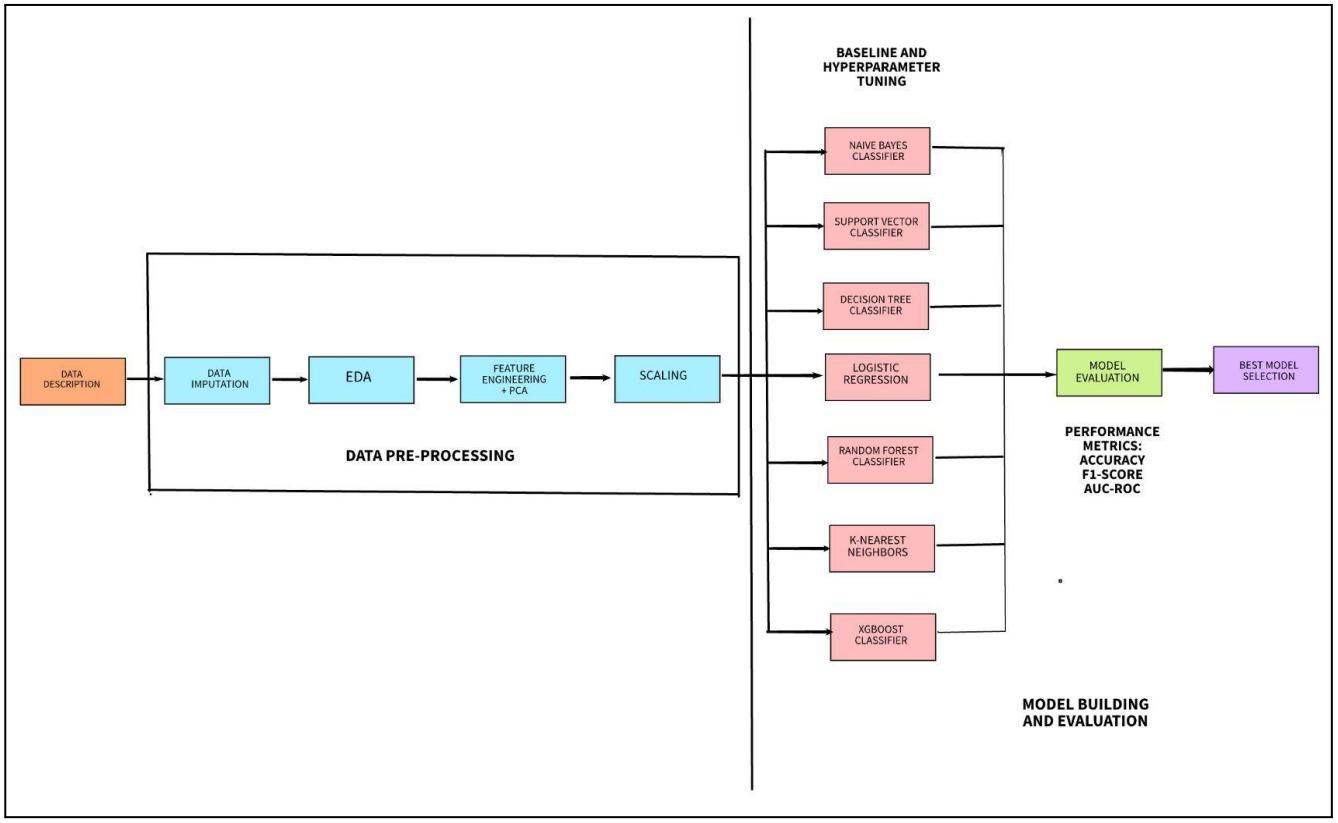
- Creation of separate pipelines helps us to fill in missing values and perform scaling on numerical features:- ('DAYS_BIRTH_x','DAYS_BIRTH_y','DAYS_EMPLOYED','DAYS_REGISTRATION','OWN_CAR_AGE').

- We are using SimpleImputer for both pipelines to fill in the missing values with the median being used, for the numerical pipeline and the most frequent value being used,for the categorical pipeline.

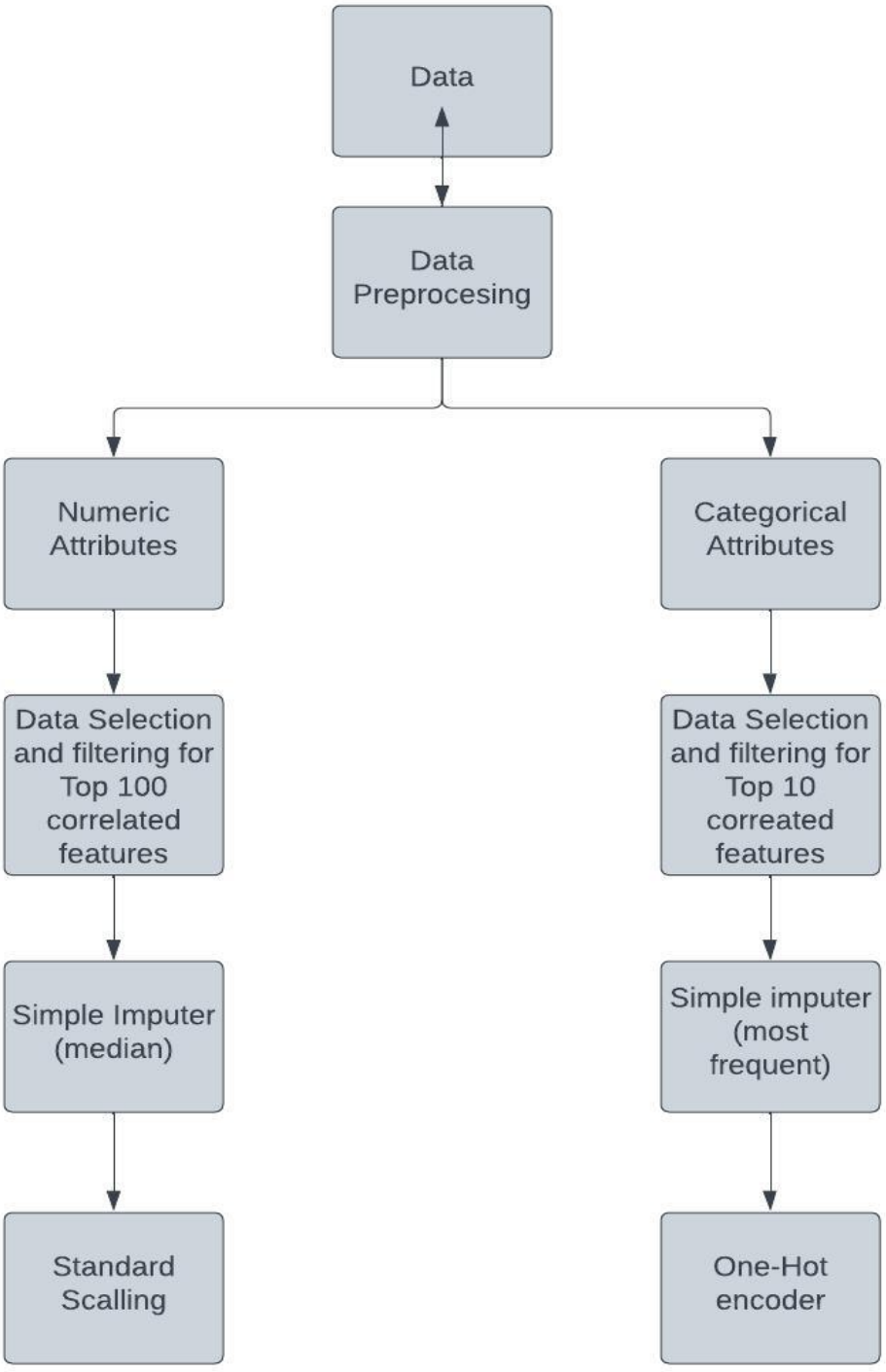
- In the categorical pipeline we used One Hot Encoding, to convert categorical values : ('FLAG_OWN_CAR','FLAG_OWN_REALTY','FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHONE','FLAG_CONT_MOBILE', 'FLAG_PHONE') into integers for improving the performance of our models.

- Standard Scaler is used in the numerical pipeline for scaling purposes. The two pipelines are then combined into a single ‘ data_prep_pipeline ’ .We also tried using ‘MinMaxScaler’ but we were getting ‘column mismatch’ error.

- We then performed hyperparamter tuning of the models.



DATA LINEAGE



The diagram shows the flow of data lineage for a machine learning model that involves both numerical and categorical attributes.

Firstly, the data is preprocessed, which includes cleaning, transforming, and normalizing data to make it more suitable for analysis. Once the data is preprocessed, the numerical attributes are selected and filtered to identify the top correlated features with the target feature. From these selected features, the 100 most highly correlated attributes are chosen to be used in the model.
After selecting the numerical attributes, a simple imputer of median is used to fill in any missing data points. The data is then standardized using standard scaling, which transforms the data to have a mean of 0 and a standard deviation of 1.

Next, the categorical attributes are selected and filtered to identify the top 10 correlated features with the target feature. A simple imputer of the most frequent value is used to fill in any missing data points. Finally, one-hot encoding is applied to the categorical data, which converts the categorical attributes into binary values to be used in the machine learning model.

Overall, this data preprocessing flow aims to identify the most important numerical and categorical attributes and remove any unnecessary ones, impute any missing data points, and transform the data to be more suitable for analysis in a machine learning model.

Feature Engineering

- The application_train.csv and application_test.csv datasets were the only ones used in the prior phase's feature engineering, which introduced 3rd-degree polynomial features. To train and test the baseline models, we used a portion of the fresh data.

- In this stage, we prepare features from additional secondary tables that the HCDR dataset offers us. In this article, the previous_application.csv and bureau.csv tables have received the most of our attention.

- This phase will be divided into the following three sections:
  Feature Engineering in the previous_application.csv table.
  Feature Engineering in the brueau.csv table.
  Combining the two secondary tables with the application_train.csv and application_test.csv tables.

For previous_application.csv table:

- We created three types of aggregated features (min, max, mean) for the following columns: 'AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE', 'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY', 'RATE_INTEREST_PRIVILEGED', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE', 'CNT_PAYMENT', 'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION'

- For AMT_APPLICATION , we created a range_AMT_APPLICATION that takes the range from the maximum and minimum of the said feature.

- The newly created aggregated table was then combined with the application_train.csv and application_test.csv tables using a straightforward merge function based on the SK_ID_CURR identifier.

For bureau.csv table:

- This table contains the information for 263,491 applicants, as can be seen.

- For all of these applicants, we first developed a previous_loan_count feature based on the SK_ID_BUREAU feature.

- Then, by executing a groupby on the SK_ID_CURR feature, we produced five categories of aggregated features (count, mean, max, min, and sum) for the remaining columns:

- We executed a merging of the aggregated bureau table with the application_train.csv and application_test.csv tables based on the SK_ID_CURR identification after suitably changing the fields.

- After these processes, the training dataset (merged application tr number of trees, the maximumain dataset) has 335 columns, while the testing dataset (merged application test.csv) has 334 columns.

Hyperparameter Tuning

- We employed an indirect approach to efficiently carry out a hyperparameter tuning grid search and random search and used cross-validation to 5, to divide five equally sized folds, and the model will be trained and evaluated five times, with each fold used once for evaluation and the other four folds used for training.
- Finding which classifier performed best and with which hyperparameters was the aim of the hyperparameter tuning.
- We evaluated the following classifiers: Gradient Boosting Classifier, Decision Tree Classifier, Random Forest Classifier, Logistic Regression Classifier, and XG Boost Classifier, AdaBoost Classifier.
- We used ensemble classifiers such as: Gradient Boosting Classifier, Random Forest Classifier, XG Boost Classifier, AdaBoost Classifier.
- Non-ensemble classifiers such as: Decision Tree Classifier, Logistic Regression Classifiers.
- For each of these models, we also tested a large number of hyperparameters for both grid search and random search and compared the results, where grid-search were slightly better then RandomSearch for AUC-ROC score.

Logistic Regression: We put the penalty type and penalty coefficient to the test, we used : [L1 regularization, L2 regularization, and elasticnet is a combination of both.]. We ran the Logistic Regression for different solver's : 'liblinear' and 'saga'. and set the regularization strength to [1, 10].

Random Forest: The number of trees, the maximum depth of each tree, and the bare minimum number of samples that could be used for each leaf node were all put to the test.

We evaluated the maximum depth and the bare minimum sample sizes for leaf nodes in decision trees.

Decision Tree Classifier: We experimented with maximum depth of the decision tree, minimum number of samples required to be at a leaf node.

Gradient Boosting Classifier: We experimented with the number of boosting stages, the minimum samples for leaf nodes, the maximum depth, and the subsample parameter, which determines how many data would be used to fit each tree.

AdaBoost Classifier: We tried different algorithms like: 'SAMME' and 'SAMME.R', and set learning_rates as [0.1, 0.5, and 1] and number of estimators we tried [50, 100, and 200].

XGBoost classifier: We tested the depth, the subsample, the 'colsample_bytree' parameter that controls the proportion of columns used to build each tree, the gamma parameter that determines how much loss reduction is required to partition further down the tree, the alpha and lambda regularization parameters, and the 'colsample_bytree' parameter.

We used a different strategy because some of these classifiers require a long training period.

We used 5000 training samples to test every classifier with every setting. As a result, we discovered that AdaBoost was the first in terms of effectiveness. We picked the best-performing AdaBoost settings to compare to the other classifiers and their parameters because AdaBoost appeared to have the most impact on the speed of the grid search. We utilized 10,000 examples this time. Once more, the AdaBoost classifier showed to be the most effective.

We got AUC-ROC score 72.354 % by random search for the model RandomForestClassifier(min_samples_leaf=10, n_estimators=200).
We got AUC-ROC score 72.38 % by GridsearchCV for the model AdaBoostClassifier(learning_rate=0.1), 'classifier__algorithm': 'SAMME.R')

<u>Discussion</u>

- We had to make a number of significant judgments during this stage of the project based on the availability of numerous models and a sizable dataset.
- With a primary focus on the recently prepared features, but making sure that our past data allows us to tune the models in a reasonable amount of time, we developed two distinct

pipelines for hyperparameter tuning, based on data from the previous phase and the present phase.

- We took chunks of data (10000 rows was the most before we found the models did not converge to a solution) to gauge how long it takes for the models to train because we are aware that we will be plagued by the high dimensionality of the data.
- Since accuracy cannot be a meaningful metric to evaluate performance until we oversample (unlikely due to complexity and huge number of data) or undersample (more likely) the dataset in accordance with the TARGET feature, imbalance in the dataset appears to be a worry as well at this time.
- Since random forest is an ensemble model that uses different subsets of the data, we intend to create the feature importances attribute since it will provide the best estimate of which features should be taken into account for our final run of the pipeline.

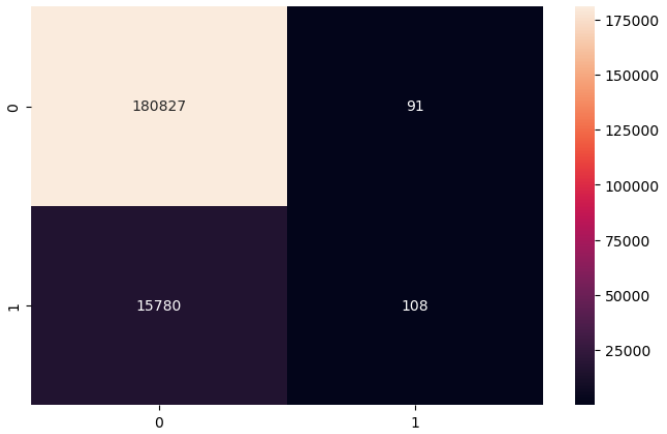
Discussion of results:

As per our results, when we fitted the models for the entire dataset and all the number of rows, and 104 columns (100 numerical, 4 categorical), as per the above expLog we can conclude and compare the AUC-ROC scores that GradientBoost Classifier was the best model, and according to the rankings we have printed the model names in descending order for Test AUC.

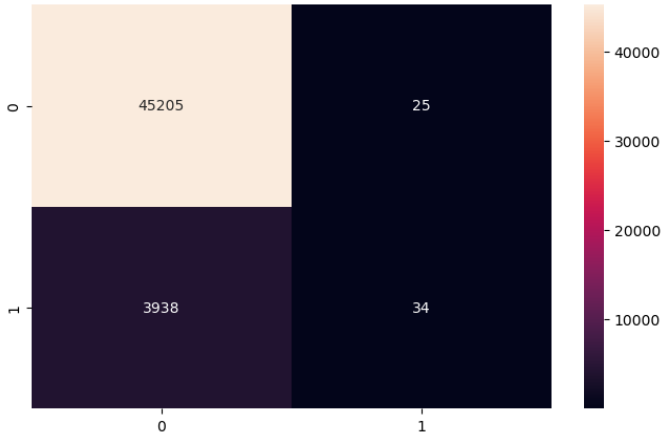
Logistic Regression [liblinear]:

Confusion Matrix:

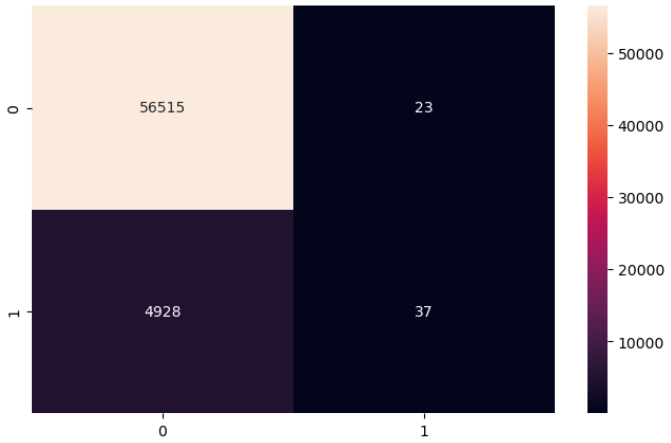
Train Set:



Validation Set:

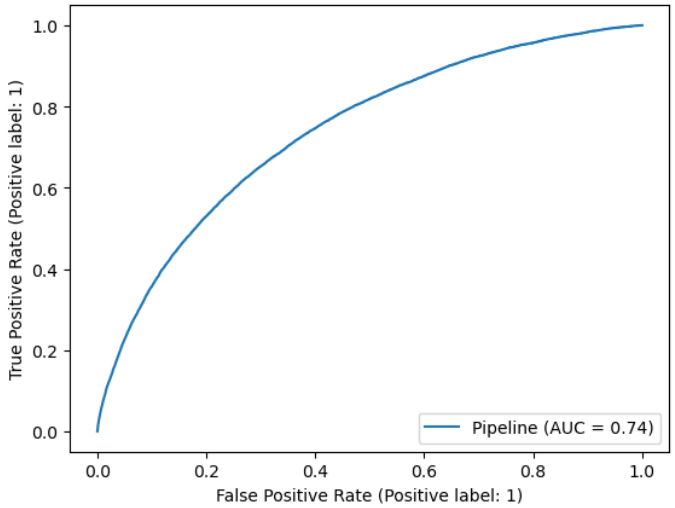


Test Set:

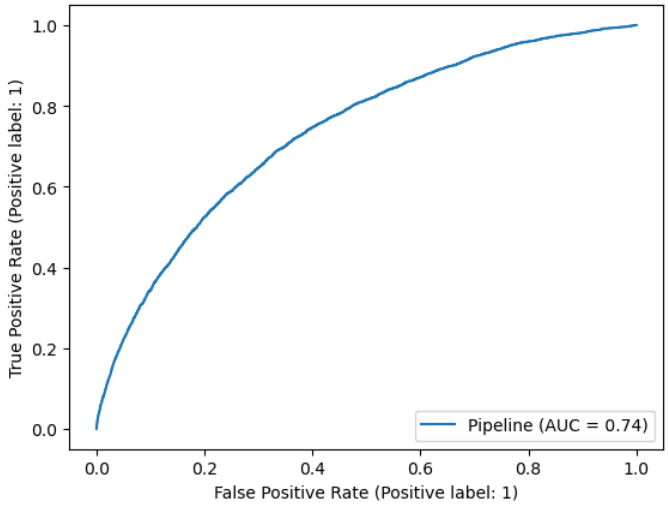


AUC-ROC Curve:

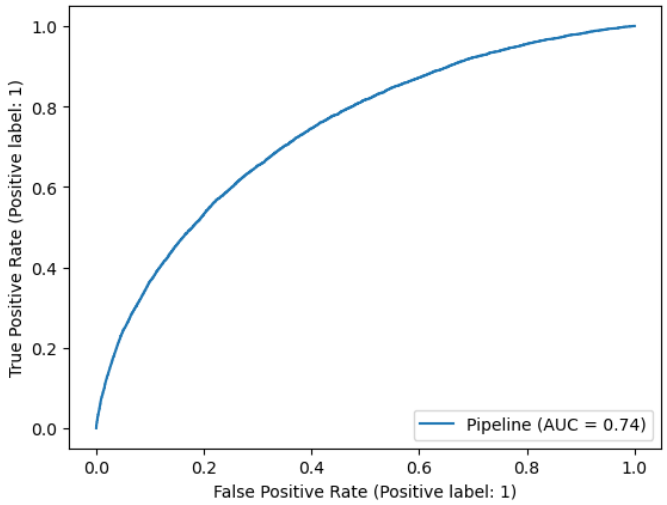
Train Set:



Validation Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.74)
0.0
0.2
0.4
0.6
0.8
1.0

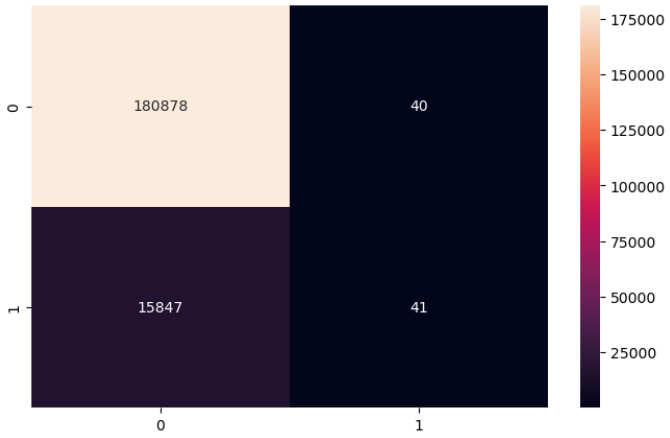
Test Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.74)
0.0
0.2
0.4
0.6
0.8
1.0

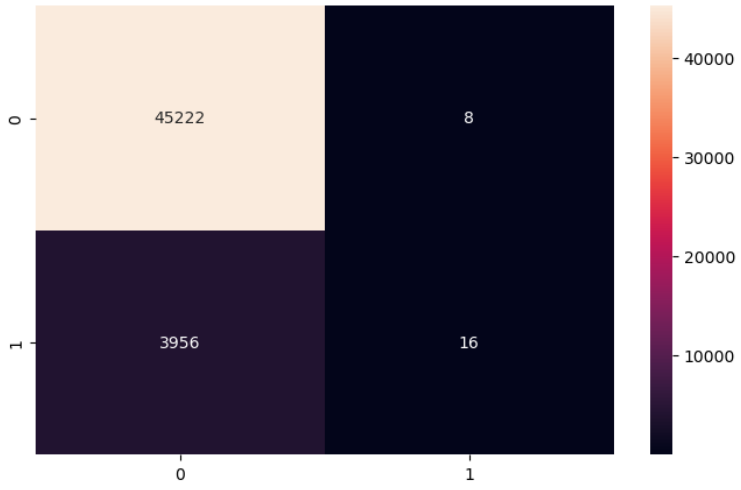
Logistic Regression [saga]:

Confusion Matrix:

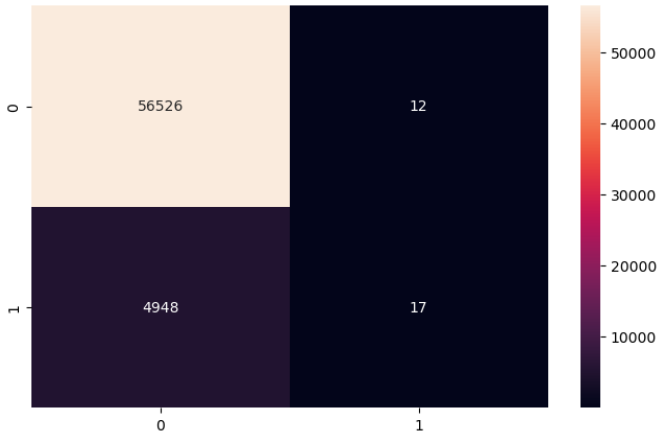
Train Set:



Validation Set:

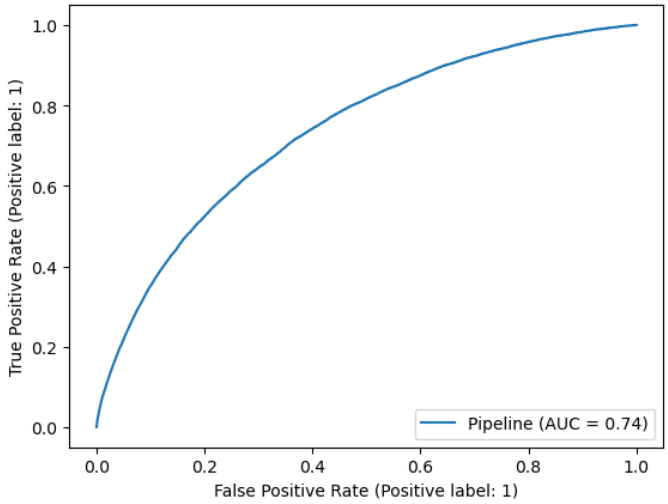


Test Set:

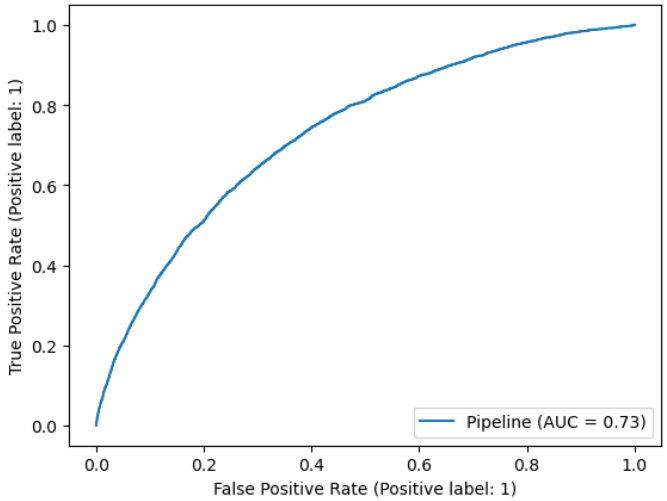


AUC-ROC Curve:

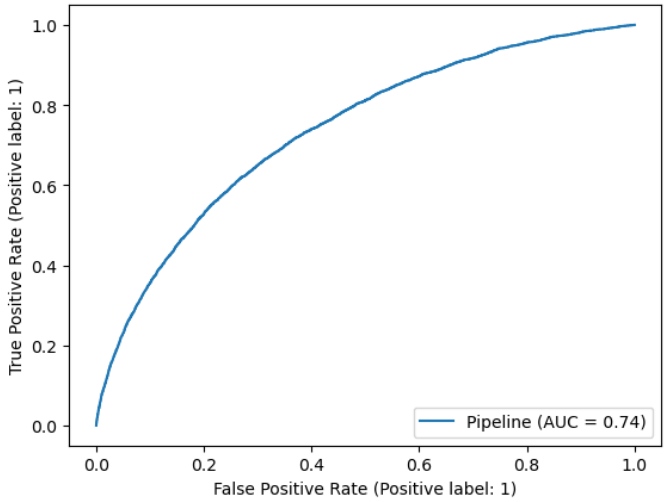
Train Set:



Validation Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.73)
0.0
0.2
0.4
0.6
0.8
1.0

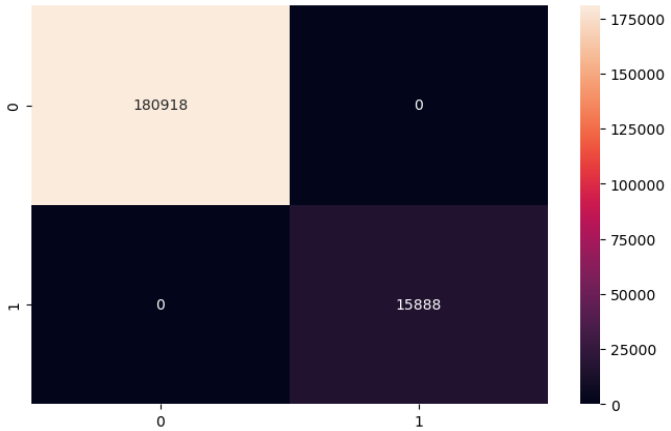
Test Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.74)
0.0
0.2
0.4
0.6
0.8
1.0

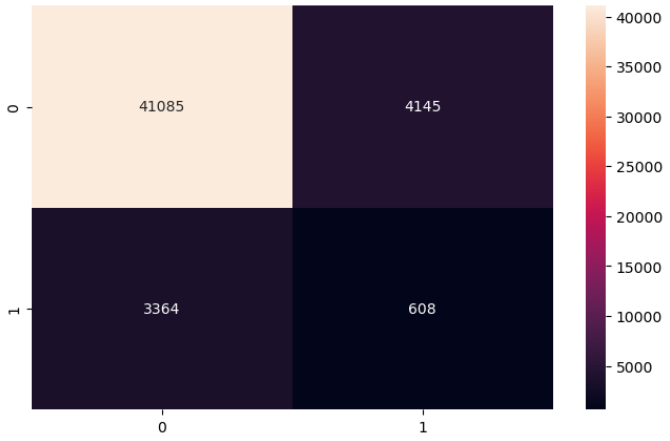
DecisionTreeClassifier:

Confusion Matrix:

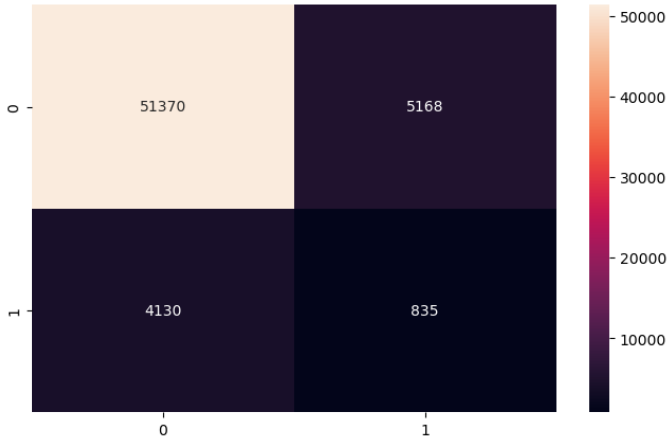
Train Set:



Validation Set:

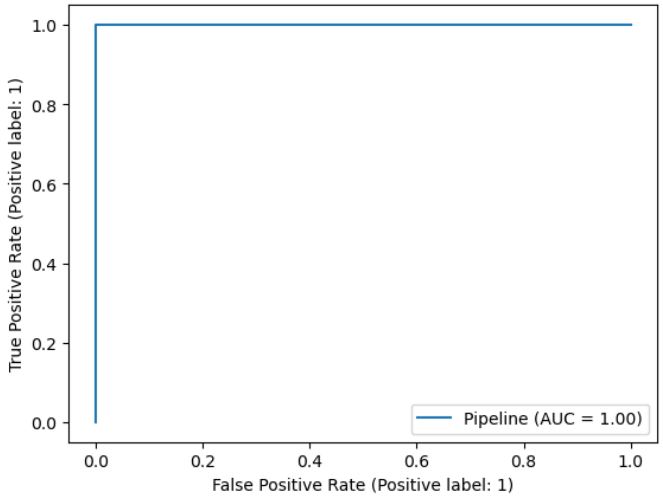


Test Set:

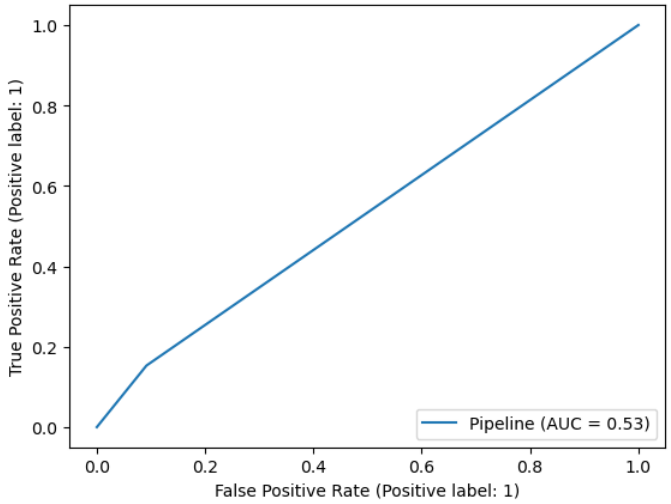


AUC-ROC Curve:

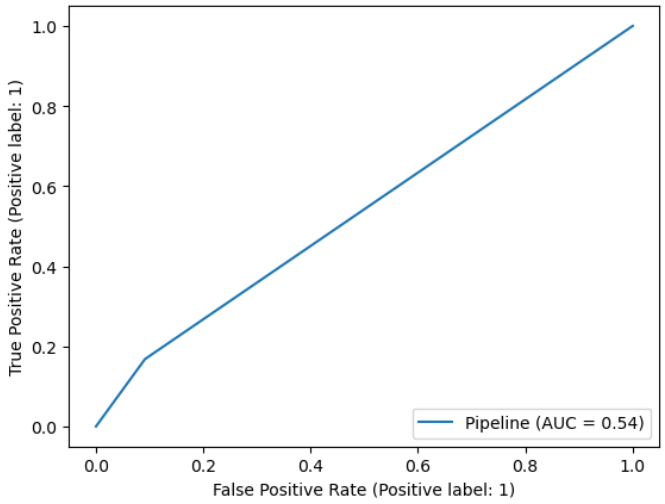
Train Set:



Validation Set:



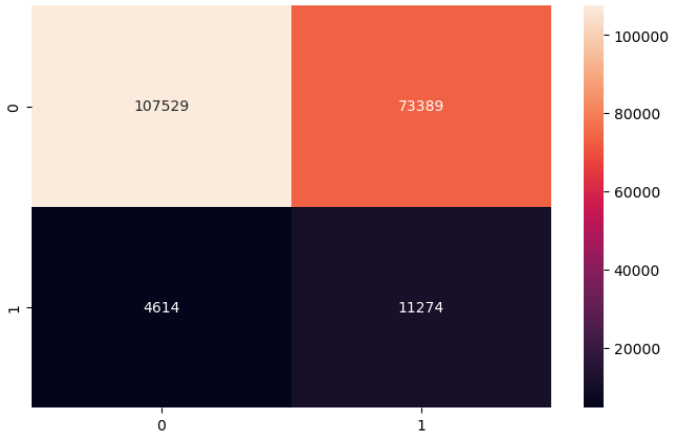
Test Set:



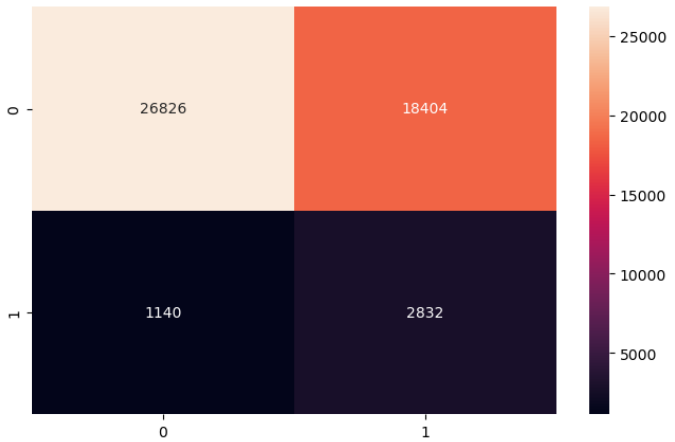
Gaussian Naive Bayes:

Confusion Matrix:

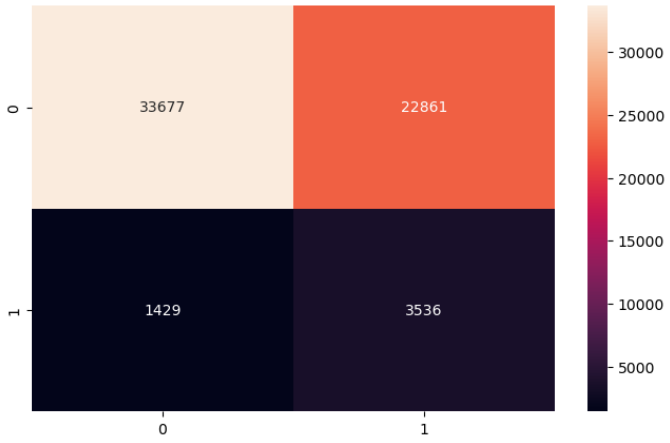
Train Set:



Validation Set:

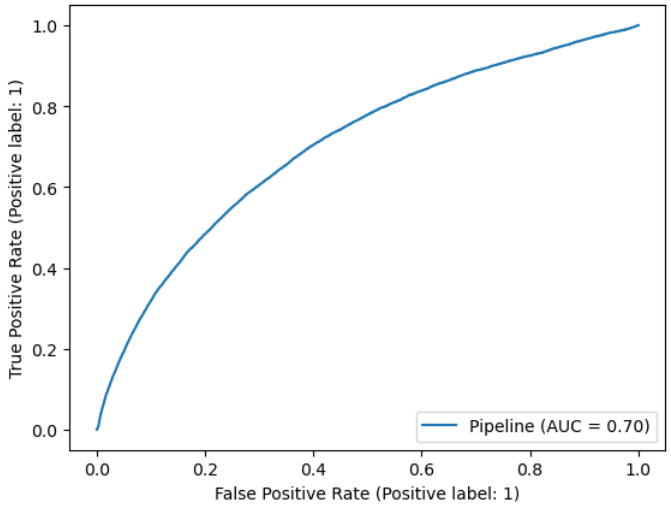


Test Set:

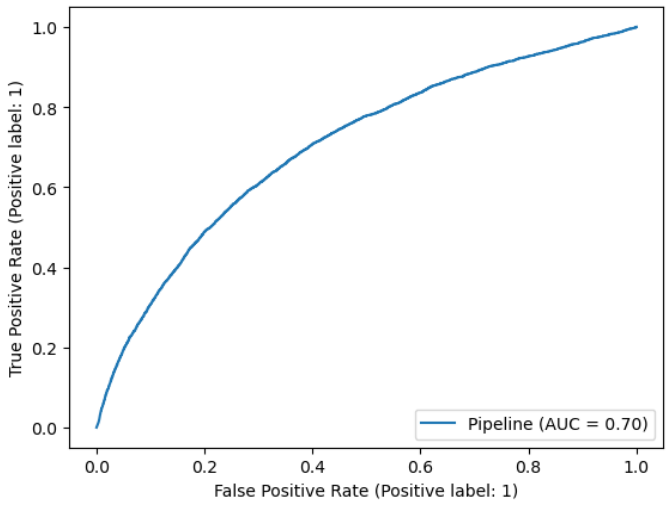


AUC-ROC Curve:

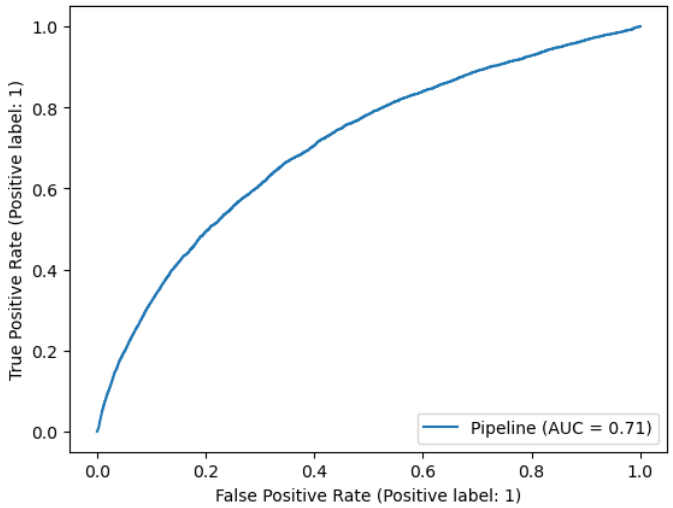
Train Set:



Validation Set:



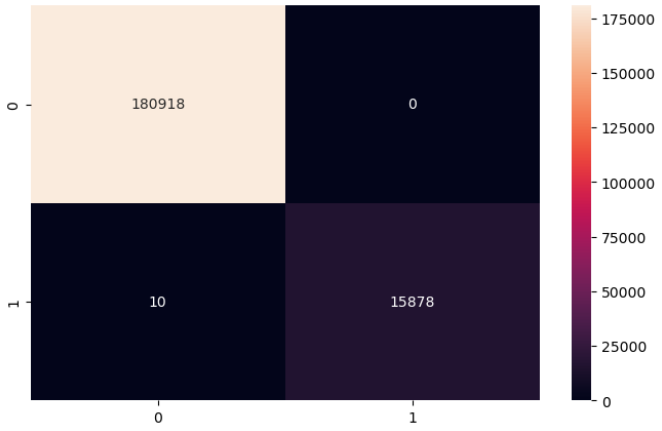
Test Set:


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.71)

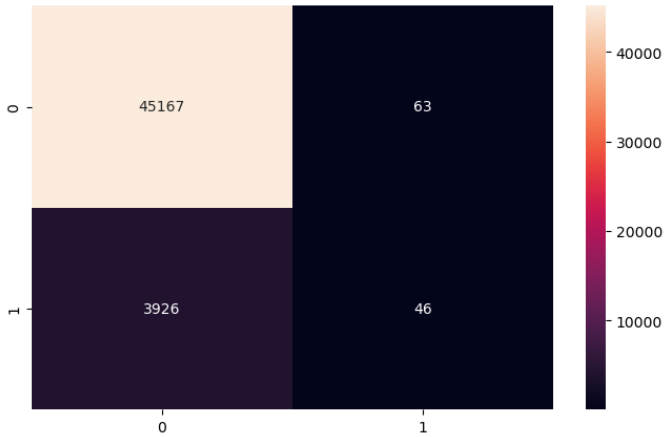
RandomForestClassifier:

Confusion Matrix:

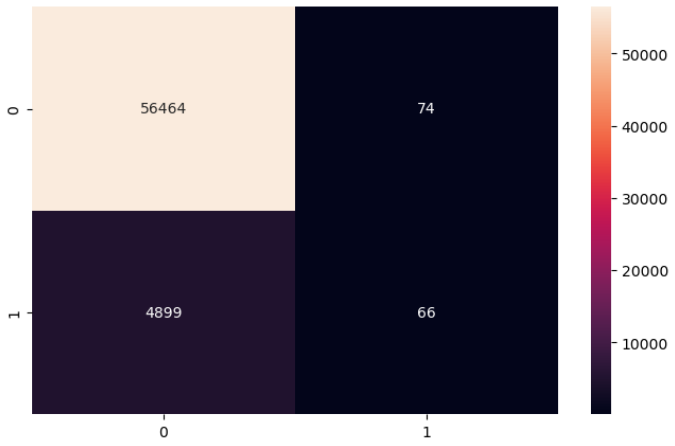
Train Set:



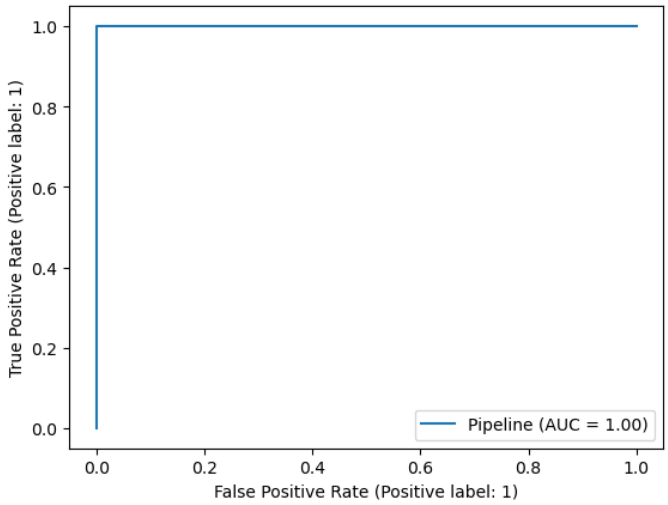
Validation Set:



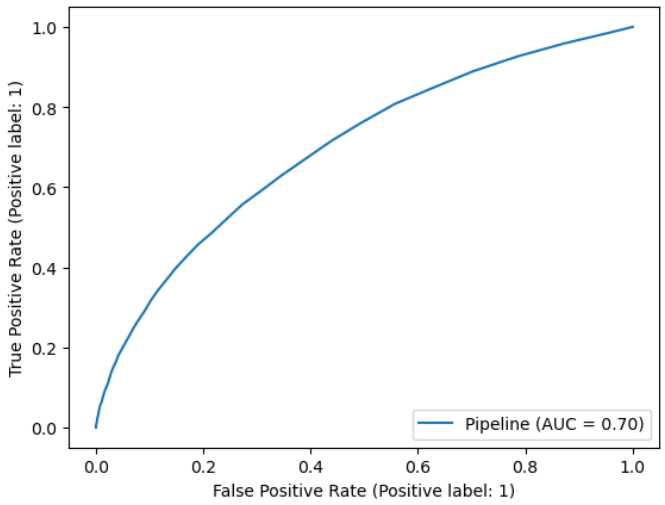
Test Set:



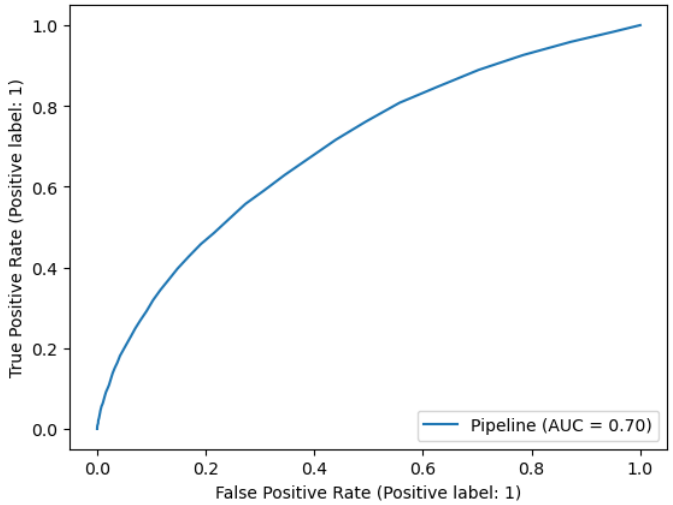
AUC-ROC Curve:



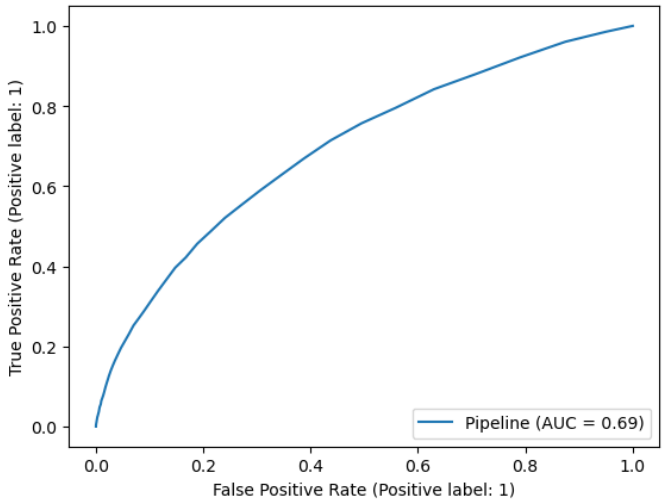
Train Set:



Validation Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.70)
0.0
0.2
0.4
0.6
0.8
1.0

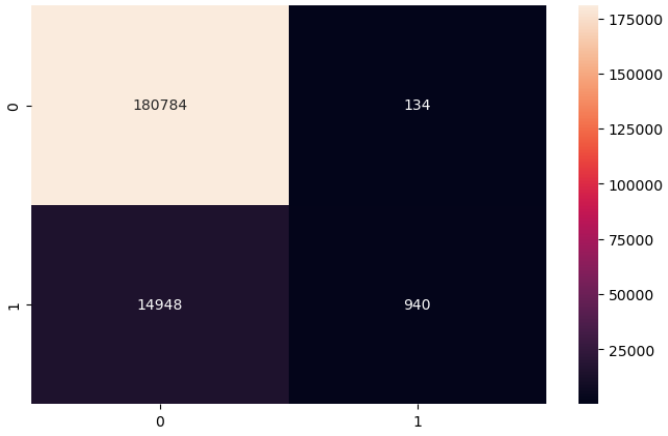
Test Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.69)
0.0
0.2
0.4
0.6
0.8
1.0

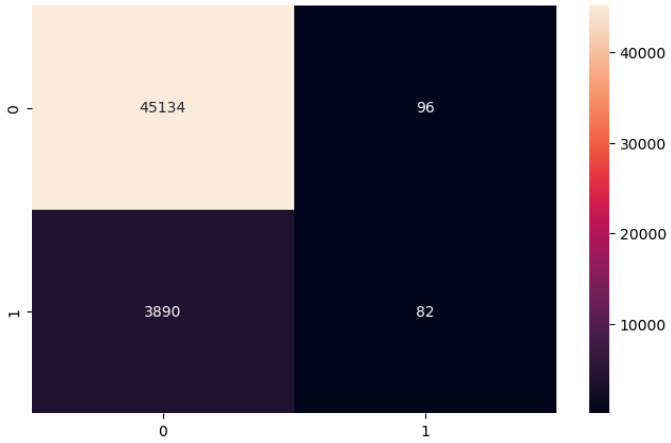
XGBClassifier:

Confusion Matrix:

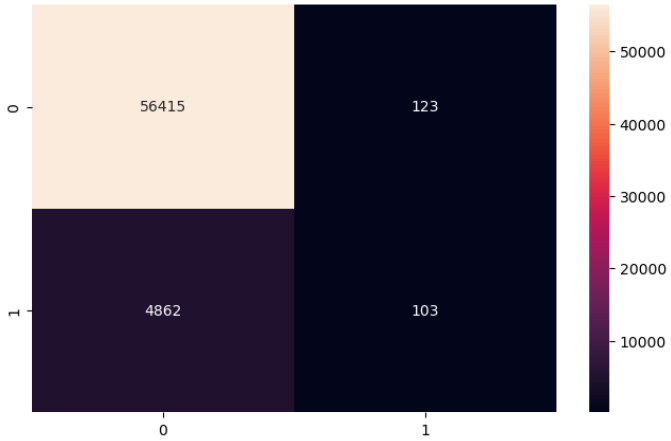
Train Set:



Validation Set:

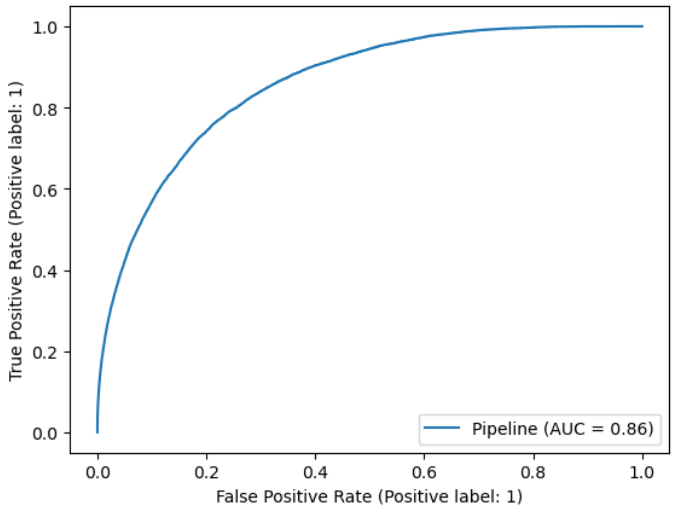


Test Set:

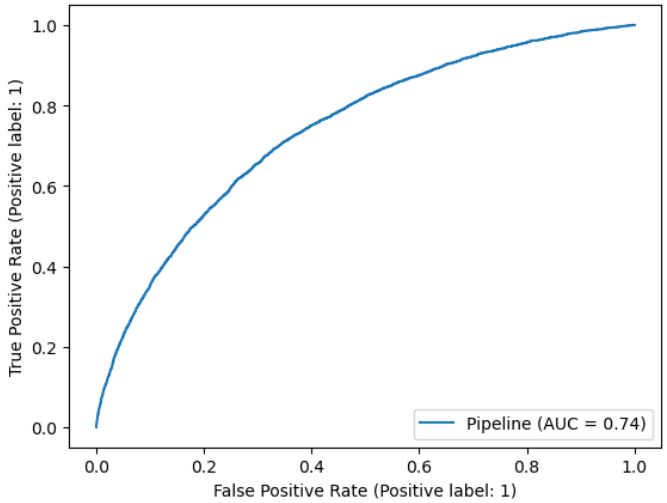


AUC-ROC Curve:

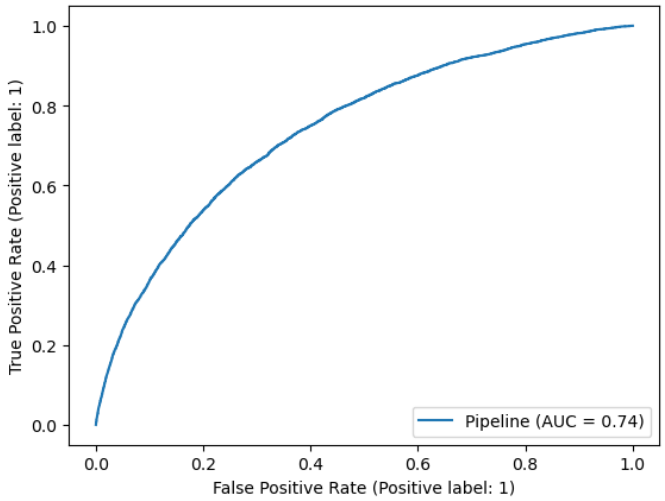
Train Set:



Validation Set:



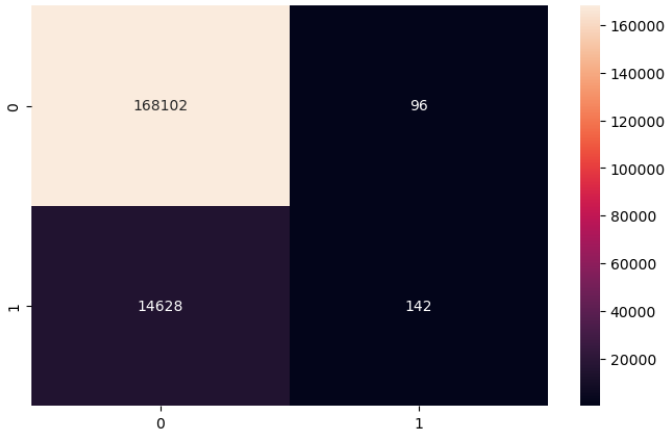
Test Set:


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.74)

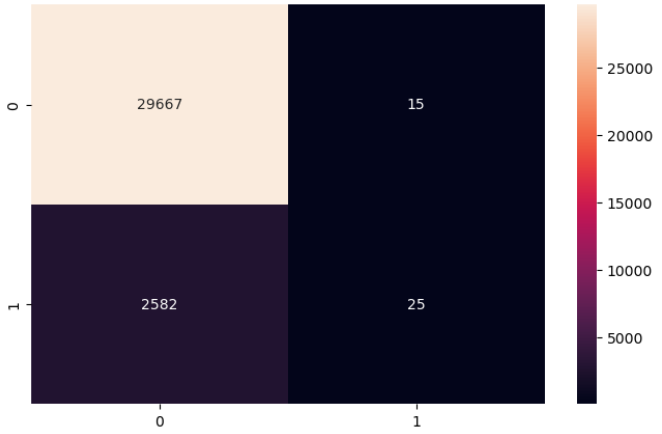
AdaBoostClassifier:

Confusion Matrix:

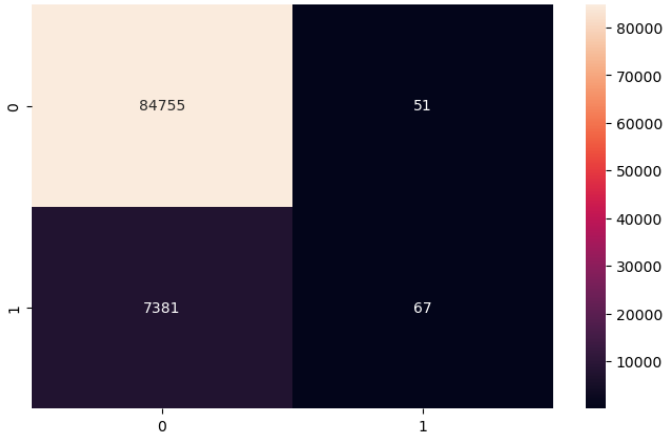
Train Set:



Validation Set:

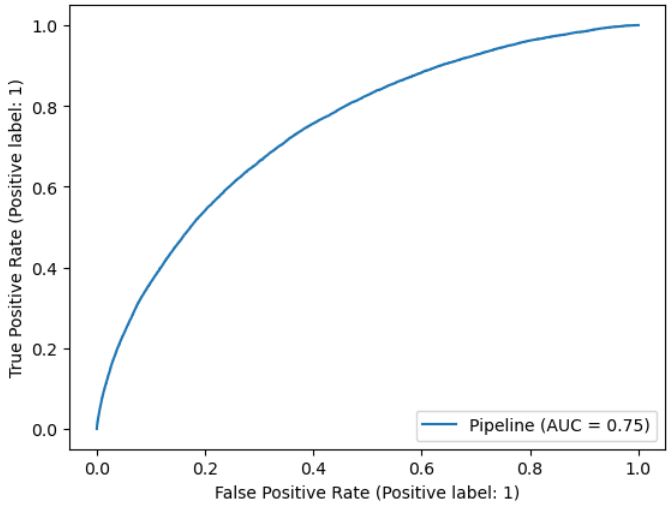


Test Set:

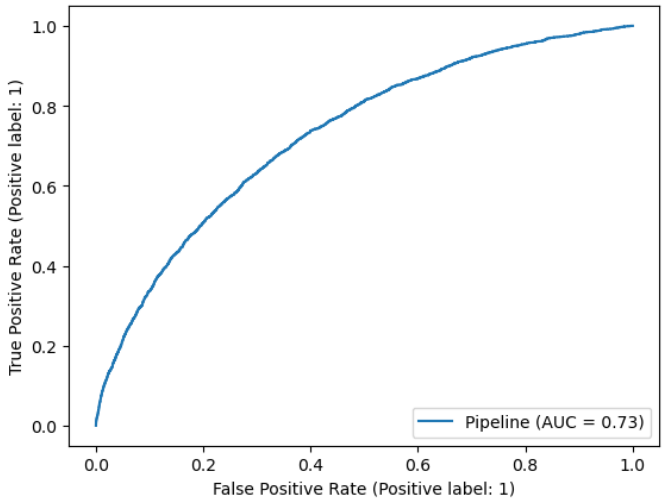


AUC-ROC Curve:

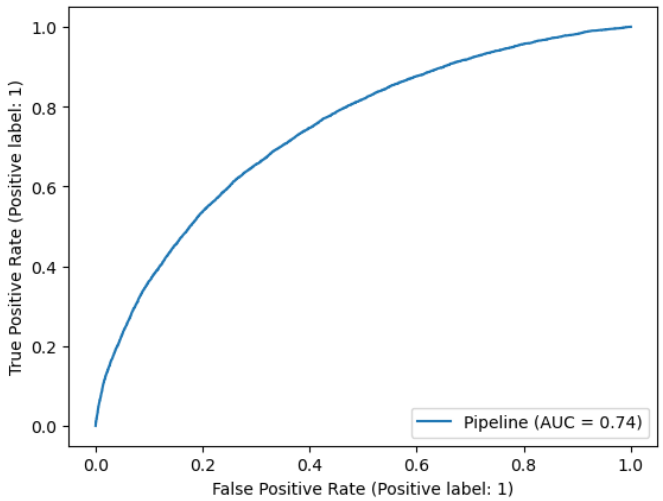
Train Set:



Validation Set:



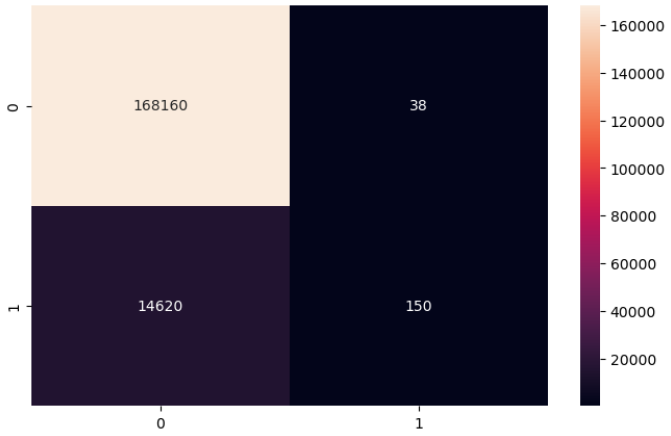
Test Set:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.74)
0.0
0.2
0.4
0.6
0.8
1.0

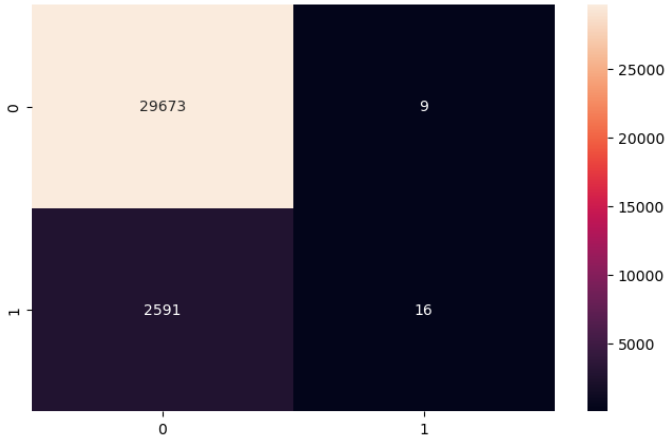
GradientBoostingClassifier:

Confusion Matrix:

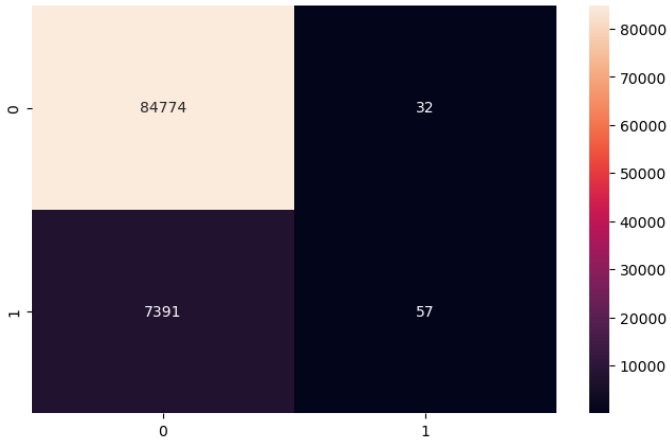
Train Set:



Validation Set:

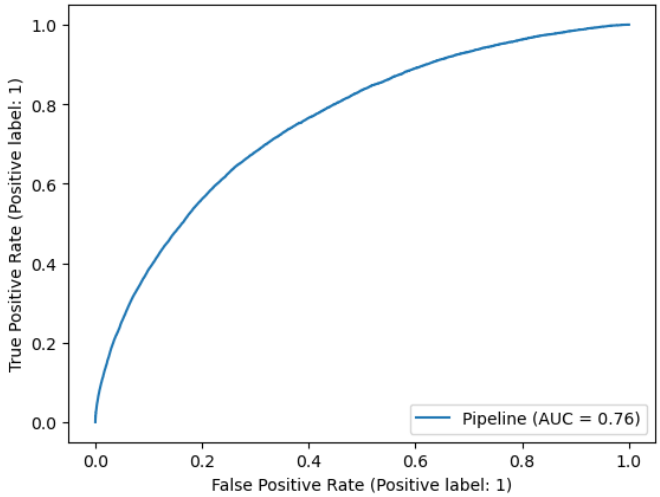


Test Set:

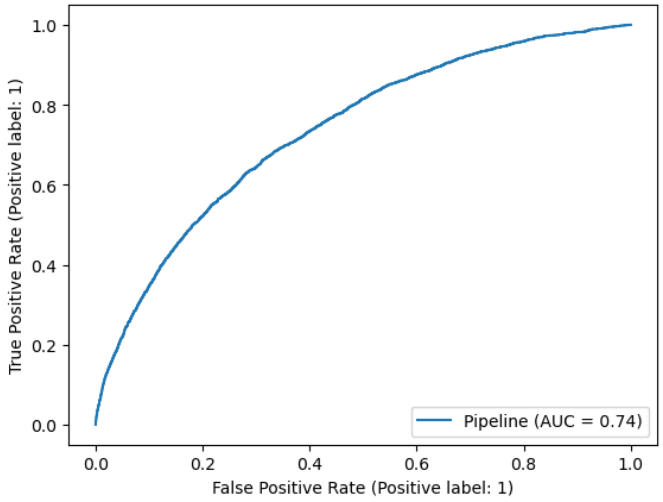


AUC-ROC Curve:

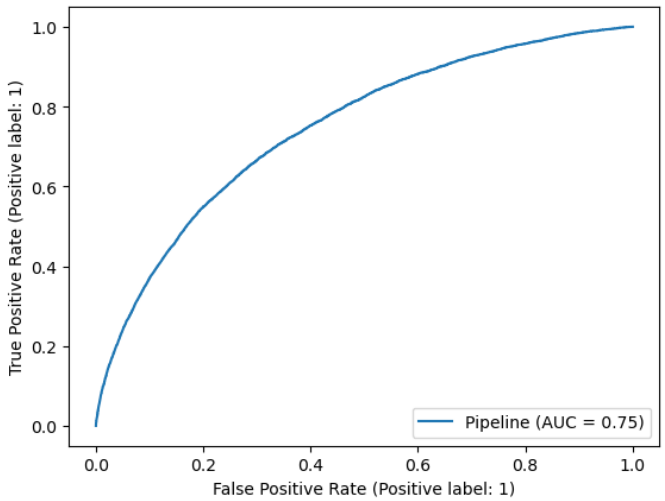
Train Set:



Validation Set:



Test Set:



	exp_name	Model name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1	Valid F1	Test F1	Fit Time
0	Baseline_104_features	Baseline LogisticRegression	0.9194	0.9195	0.9195	0.7400	0.7362	0.7397	0.0134	0.0169	0.0147	36.0503
1	Baseline_104_features	Baseline LogisticRegression	0.9193	0.9194	0.9194	0.7369	0.7330	0.7367	0.0051	0.0080	0.0068	29.1212
2	Baseline_104_features	Baseline DecisionTreeClassifier	1.0000	0.8491	0.8490	1.0000	0.5358	0.5351	1.0000	0.1478	0.1467	38.1139
3	Baseline_104_features	Baseline GaussianNB	0.6037	0.6028	0.6051	0.7043	0.7037	0.7077	0.2242	0.2247	0.2255	4.8185
4	Baseline_104_features	Baseline AdaBoostClassifier	0.9194	0.9194	0.9194	0.7452	0.7392	0.7431	0.0147	0.0149	0.0155	82.8952
5	Baseline_104_features	Baseline GradientBoostingClassifier	0.9199	0.9195	0.9194	0.7541	0.7438	0.7460	0.0223	0.0188	0.0171	413.8077
6	Baseline_104_features	Baseline RandomForestClassifier	0.9999	0.9193	0.9190	1.0000	0.6987	0.6934	0.9996	0.0260	0.0224	183.4761
7	Baseline_104_features	Baseline XGBClassifier	0.9234	0.9190	0.9189	0.8587	0.7399	0.7420	0.1108	0.0395	0.0397	64.2782
8	Baseline_104_features	Baseline AdaBoostClassifier	0.9194	0.9194	0.9194	0.7452	0.7392	0.7431	0.0147	0.0149	0.0155	77.9032
9	Baseline_104_features	Baseline GradientBoostingClassifier	0.9199	0.9195	0.9194	0.7541	0.7438	0.7460	0.0223	0.0188	0.0171	407.1805

```
AUC scores sorted in descending order:
5    Baseline GradientBoostingClassifier
9    Baseline GradientBoostingClassifier
4            Baseline AdaBoostClassifier
8            Baseline AdaBoostClassifier
7                 Baseline XGBClassifier
0            Baseline LogisticRegression
1            Baseline LogisticRegression
3                    Baseline GaussianNB
6        Baseline RandomForestClassifier
2        Baseline DecisionTreeClassifier
Name: Model name, dtype: object          AUC: 5    0.7460
9    0.7460
4    0.7431
8    0.7431
7    0.7420
0    0.7397
1    0.7367
3    0.7077
6    0.6934
2    0.5351
Name: Test  AUC, dtype: float64
```

For Test-Accuracy:

```
Test Accuracy scores sorted in descending order:
0            Baseline LogisticRegression
1            Baseline LogisticRegression
4            Baseline AdaBoostClassifier
5    Baseline GradientBoostingClassifier
8            Baseline AdaBoostClassifier
9    Baseline GradientBoostingClassifier
6        Baseline RandomForestClassifier
7                 Baseline XGBClassifier
2        Baseline DecisionTreeClassifier
3                    Baseline GaussianNB
Name: Model name, dtype: object
```

Whereas for Testing Accuracy, Logistic Regression and AdaBoost had the highest testing accuracy, and worst for Decision Tree Classifier and Baseline GaussianNB, whereas if we compare on the basis of F-1 score then

```
Test F1 scores sorted in descending order:
3                    Baseline GaussianNB
2         Baseline DecisionTreeClassifier
5                 Baseline XGBClassifier
4         Baseline RandomForestClassifier
7    Baseline GradientBoostingClassifier
6             Baseline AdaBoostClassifier
0             Baseline LogisticRegression
1             Baseline LogisticRegression
Name: Model name, dtype: object
```

On the basis of Test F1 score, Baseline GaussianNB and Baseline Decision Tree Classifier was better, but which is totally contradictory to the above results that's why we consider, since accuracy cannot be a meaningful metric to evaluate performance until we oversample (unlikely due to complexity and huge number of data) or undersample (more likely) the dataset in accordance with the TARGET feature, imbalance in the dataset appears to be a worry as well at this time.

Our results showed that DecisionTreeClassifier model is experiencing overfitting, In this case, the model is performing perfectly on the training set (Train-Accuracy: 1.0000), but its performance drops significantly on the validation and test sets (Valid-Accuracy: 0.8491 and Test-Accuracy: 0.8490, respectively). It appears that the RandomForestClassifier model is slightly overfitting. In this case, the model is performing very well on the training set (Train-Accuracy: 0.9999), but its performance drops slightly on the validation and test sets (Valid-Accuracy: 0.9193 and Test-Accuracy: 0.9190, respectively).

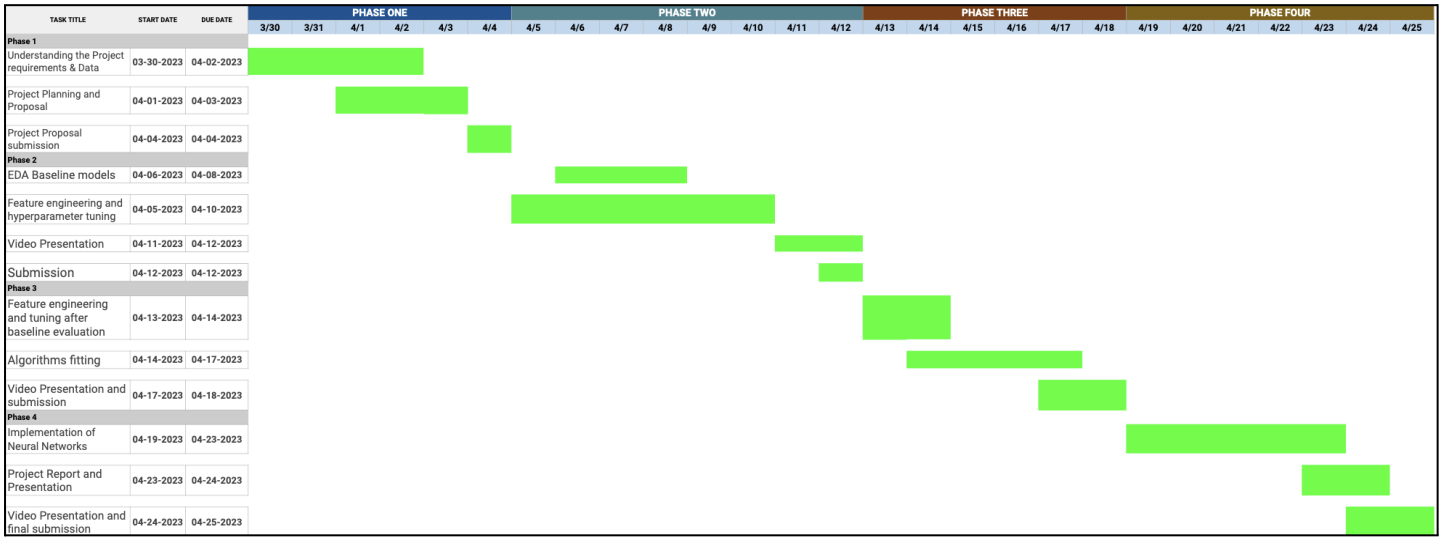
Further tests revealed that XGBoost and Logistic Regression also had the highest Accuracy in the test set. These results suggest that Logistic Regression and AdaBoost are promising models for predicting borrower's repayment capacity using the "Home Credit Default Risk" dataset.

And the Other models, Random ForestClassifier and The Decision Tree Classifier had low accuracy which is comparatively low, and may be improved if selected in some other columns.

Overall, our experiment demonstrated the effectiveness of machine learning models in predicting borrower's repayment capacity and highlighted the potential of Logistic Regression and AdaBoost or we can also use GradientBoost as suitable models for this task. However, further research and fine-tuning of the models may be necessary to improve their performance and applicability in real-world scenarios.

Timeline

The proposed timeline for the Home Credit Default Risk Project:



TASK TITLE	START DATE	DUE DATE
Phase 1		
Understanding the Project requirements & Data	03-30-2023	04-02-2023
Project Planning and Proposal	04-01-2023	04-03-2023
Project Proposal submission	04-04-2023	04-04-2023
Phase 2		
EDA Baseline models	04-06-2023	04-08-2023
Feature engineering and hyperparameter tuning	04-05-2023	04-10-2023
Video Presentation	04-11-2023	04-12-2023
Submission	04-12-2023	04-12-2023
Phase 3		
Feature engineering and tuning after baseline evaluation	04-13-2023	04-14-2023
Algorithms fitting	04-14-2023	04-17-2023
Video Presentation and submission	04-17-2023	04-18-2023
Phase 4		
Implementation of Neural Networks	04-19-2023	04-23-2023
Project Report and Presentation	04-23-2023	04-24-2023
Video Presentation and final submission	04-24-2023	04-25-2023

Phase Leader Plan

Final Project Phase	Phase Leader	Phase Plan
Phase 0	Team	Team Formation
Phase 1: Project Proposal	Kumud Sharma	There will be a project proposal created outlining all of the project's components. Each team member is given a task to do. The dataset is investigated, and potential machine learning techniques that could be used in the project are described. A base pipeline for the project is chosen, and appropriate evaluation metrics are determined.
Phase 2: EDA + Baseline	Bhavya Mistry	With the chosen machine learning methods, this phase focuses on baseline model training and evaluation as well as exploratory data analysis, data imputation, and baseline model evaluation. The EDA and baseline performances of several models will be used to derive conclusions, and then judgments will be taken regarding hyperparameter tuning and feature selection.

Phase 3: Feature Engineering + Hyperparameter Tuning	Kamna Chaudhary	Here, we'll concentrate on the feature of choosing the most suitable features based on several strategies, including correlation, developing new features, and dimensionality reduction, as well as our comprehension of the data from earlier phases. To help determine the ideal set of parameters for each machine learning model, we will also begin experimenting with the different parameters of the models. This will pave the stage for an exciting race between the various algorithms.
Phase 4: Final Submission	Jaydeep Patel	Deep neural networks will be used in this step, and we will compare their performance to the existing fine-tuned models from our previous phase. We will choose the best model and submit it in our final submission based on our numerous performance indicators.

Credit Assignment Plan (Phase 3):

Team Member	Task
Jaydeep Patel	The PolynomialFeatures class was utilized to generate three-degree features, which improved the efficiency of our data pipeline.
Kumud Sharma	GridsearchCV was used to tuning the hyperparameters for a number of classifiers, including the Gradient Boosting, Decision Tree, Random Forest, and XGBoost classifiers as well as Logistic Regression. A combination of aggregated and polynomial features was used to assess the performance of newly engineered data, and the best model was then chosen.
Kamna Chaudhary	The procedure of performing feature engineering using aggregated features from previous_application and bureau data. Additionally, the train and test datasets for polynomial features were combined and modified to fit with the aforementioned aggregated tables.
Bhavya Mistry	GridsearchCV was used for Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier,,, and XGBoost Classifier during the process of optimizing

	hyperparameters. The performance of the data with respect to the polynomial features from the initial phase was then assessed in order to choose the best model.

Credit Assignment Plan (All phases):

Phase	Member Name	Task Name	Description
Phase 1	Kumud Sharma Bhavya Mistry Kamna Chaudhary Jaydeep Patel	Understanding the Data and problemset	Understanding the Dataset given and problemset provided.
	Kumud Sharma	Assigning tasks and phase planning	Setting the timeline for the project and discussing with teammates the tasks to be done.
	Jaydeep Patel	Abstract, Data Description	Writing a detailed abstract and data description including all the tables and columns.
	Kamna Chaudhary	ML pipeline	Describing the flow of the project with the ML models we will be using.
	Bhavya Mistry	ML algorithms and metrices	Defing the different metrics to see how well the model is performing.
Phase 2	Kumud Sharma	EDA and data visualization	Performed data visualisation and analysed datasest.
	Jaydeep Patel	Feature engineering and model evaluation	Transformed data to remove or add irrelevant or relevant and performed model evaluation to see how well model is performing.
	Kamna Chaudhary	Creating pipeline and describing hyperparameters	Created a pipeline for different features like numeric and categorical and collected hyperparameterd for model.

	Bhavya Mistry	Training model, discussing results	Trained the model using different ML algorithms with the pipeline we created. Compared the accuracy and result of different baseline models we created.
Phase 3	Jaydeep Patel	Feature Engineering using Polynomial Feature class to get 3-degree features.	When there is a nonlinear relationship between the input characteristics and the output variable, polynomial features can be helpful. So we will capture more intricate interactions between the features and the output variable by creating polynomial features.
	Kamna Chaudhary	Feature Engineering using aggregated features	Aggregated features are created by applying mathematical functions such as mean, median, max, min, to groups of related features, that help to capture higher-level information and patterns in the data.
	Bhavya Mistry	Hyperparameter tuning with a combination of polynomial features	Hyperparameter Tuning performed using GridsearchCV on Logistic Regression, Decision TreeClassifier, Random Forest Classifier, Gradient Boosting Classifier, and XGBoost Classifier and best model selection on data from previous phase with polynomial features.
	Kumud Sharma	Hyperparameter tuning with a combination of polynomial and aggregated features	Hyperparameter Tuning performed using GridsearchCV on Logistic Regression, Decision TreeClassifier, Random Forest Classifier, Gradient Boosting Classifier, and XGBoost Classifier.Also selecting best model based on newly feature engineered data with a combination of

			aggregated and polynomial features.
Phase 4	Kumud Sharma	Final model evaluation	Analysing and comparing the results achieved on our final model
	Jaydeep Patel	Final project presentation	Preparing final presentation for our project
	Kamna Chaudhary	Final project report	Preparing final report for our project
	Bhavya Mistry	Hyperparameter tuning and baseline models revised	Conducting hyperparameter tuning and baseline models if needed.

Bibliography:

https://www.kaggle.com/c/home-credit-default-risk/data

https://seaborn.pydata.org/

https://scikit-learn.org

https://pandas.pydata.org/

https://numpy.org/

https://www.kaggle.com/code/flavioblondeau/home-credit-default-risk-notebook

Conclusion:

In this project, we aim to predict the likelihood of loan repayment for a specific client. In the previous phase, we conducted exploratory data analysis, performed feature engineering, and created baseline models. Among the models, XGBoost showed the best performance.

In the current phase, we merged the datasets and extracted significant features for the target variable prediction. We faced some challenges during the data merging due to the large number of columns

(330). To address this, we extracted Polynomial Features and aggregated features based on existing numerical and categorical features. We also conducted Hyperparameter tuning, and the AdaBoostClassifier showed the best AUC score of 0.746. We used ensemble methods such as RandomForest, AdaBoosting, Gradient Boosting, and XGBoost, and non-ensemble methods such as Decision Trees and Logistic Regression. We submitted the Grid Search CV for AdaBoost, which showed the best AUC public score of 71.6% and a private score of 70.4%.

Moving forward, we plan to undersample the data to make the dataset balanced, introduce dimensionality reduction using PCA, and develop a Multilayer Perceptron to achieve higher accuracy in predicting the target variable. Finally, we will compare all models based on their performance metrics and select the best model for the project.