Project Phase 2

Home Credit Default Risk(HCDR)
FP_GroupN_11

Luddy School of Informatics, Computing, and Engineering

Developed by-
Kumud Sharma(kumsharm@iu.edu)
Kamna Chaudhary (kamchau@iu.edu)
Bhavya Mistry(brmistry@iu.edu)
Jaydeep Patel(jp157@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 Confusion Matrix are among the ones we want to publish. In addition to Logistic Regression, Decision Tree Classifier, Random Forest, NaiveBayes Classifier, and XGBoost Classifier, we also used other baseline models.

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. We carried out 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.

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

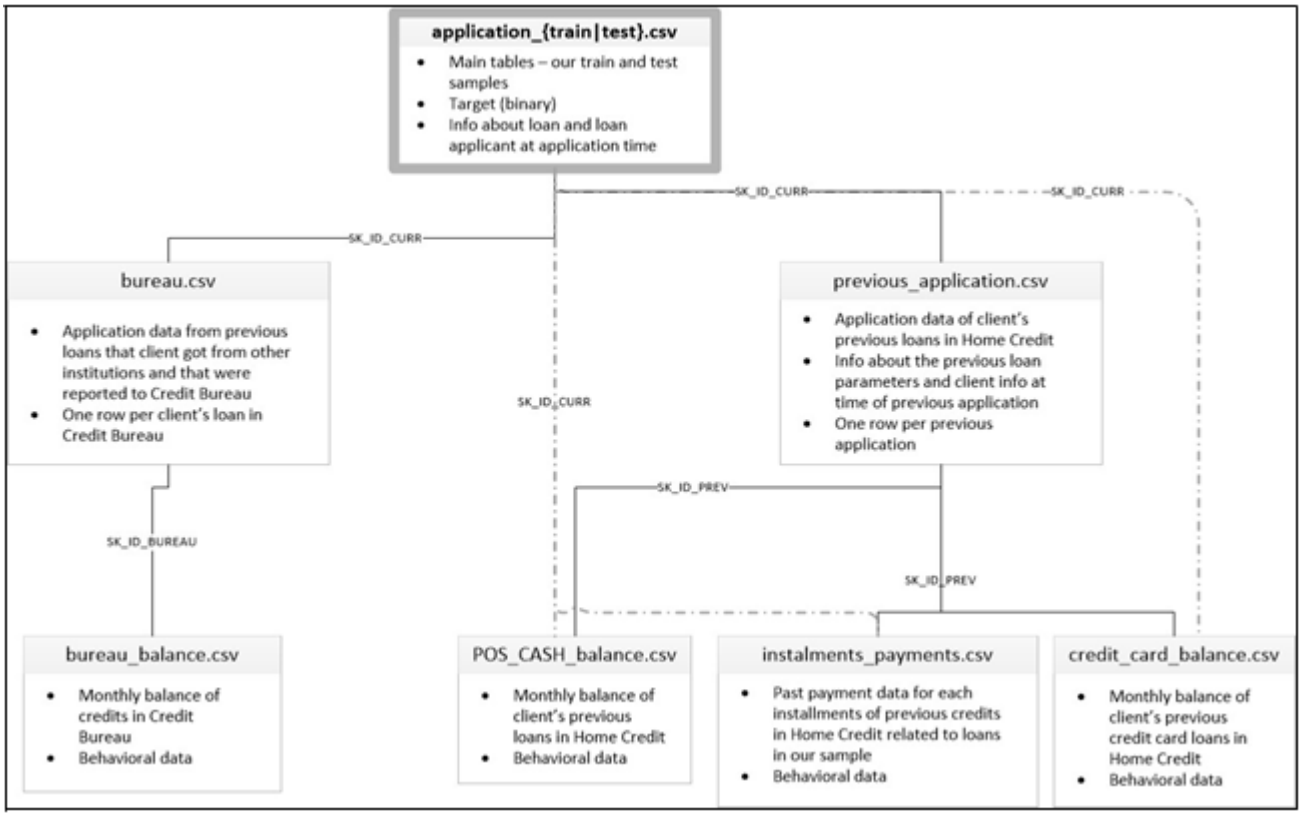
This dataset describes the individual's past credit history at Home Credit, including personal loans and consumer credit. It may be connected to the SK_ID_CURR property from the application_{train|test}.csv dataset or the SK_ID_PREV property from the previous_application.csv dataset, depending on the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



Tasks to be accomplished in Phase-2:

1.Data Visualization and Exploratory Data Analysis:
Data visualization is the graphic representation of data and information. Charts, graphs, and maps are used as visualizations to help show patterns, trends, and connections in the data. EDA, on the other hand, is the procedure of using graphical and statistical tools to assess and list the salient aspects of a dataset. Outlier detection, correlation analysis, and summary statistics are a few of the methods used in EDA. EDA is a critical step in developing a machine learning model that is effective for our problem statement and will play a significant part in verifying the correctness and quality of our data.

2. Data cleaning and preprocessing:
Several tables in the HCDR dataset have missing values, outliers, and inconsistencies. Imputation, normalization, and feature scaling will be used to address these problems in order to get the data ready for modeling.

3.Selection of Model:
In order to choose the best model for the task, a number of machine learning algorithms will be compared, taking into account performance indicators including accuracy, AUC-ROC,F-1 score. Logistic regression, decision trees, random forests, gradient boosting, and neural networks are a few examples of potential models.

4.Evaluation and Intrepretation:
To examine the generalization performance of the final model and spot any potential biases or constraints, it will be tested on a holdout set of data. To obtain understanding of the underlying connections between features and outcomes, interpretability techniques like feature importance analysis and partial dependence plots may be applied.

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.

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.

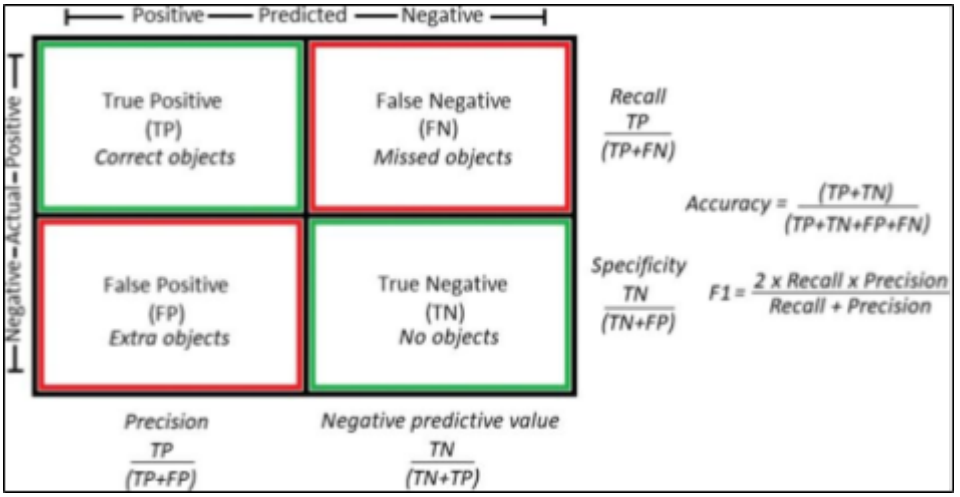
4. Neural Networks - Neural networks are a category of models that learn by varying the weights of the input data to produce a specific output. The input features are supplied into a network of linked neurons, which then manipulates the input using a number of mathematical operations. These changes are 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.



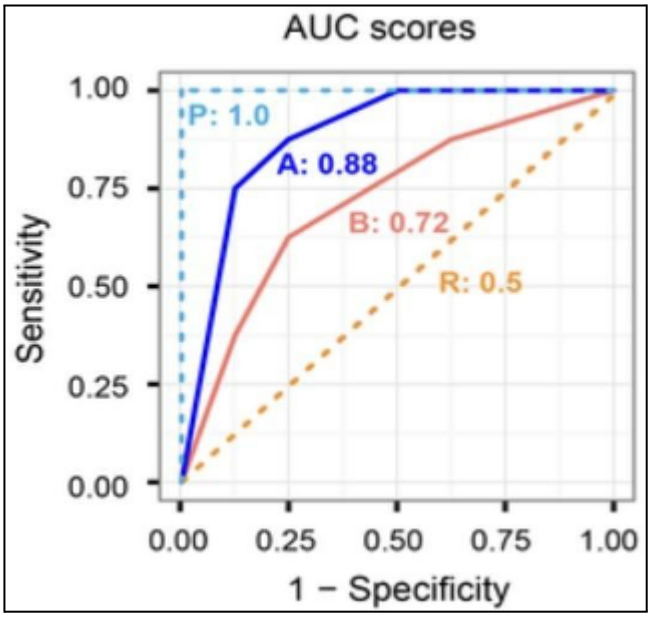
2. F-1 Score:

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.



EDA

- The Test dataset have following **numerical columns** such as:

{'SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OWN_CAR_AGE', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',

'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE', 'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21', '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'}
.

Categorical Columns:
{NAME_CONTRACT_TYPE, CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, NAME_TYPE_SUITE, NAME_INCOME_TYPE, NAME_EDUCATION_TYPE, NAME_FAMILY_STATUS, NAME_HOUSING_TYPE, OCCUPATION_TYPE, WEEKDAY_APPR_PROCESS_START, ORGANIZATION_TYPE, FONDKAPREMONT_MODE, HOUSETYPE_MODE, WALLSMATERIAL_MODE, EMERGENCYSTATE_MODE.}

Dataset size (rows columns, train, test, validation)

Application_train : Rows: 307511, Columns: 122

Application_test : Rows: 48744 , Columns: 121

Bureau : Rows: 1716428 , Columns: 17

Bureau Balance : Rows: 27299925 , Columns: 3

Credit Card Balance : Rows: 3840312 , Columns: 23

Installments Payments : Rows: 13605401 , Columns: 8

Previous Application : Rows: 1670214, Columns: 37

POS CASH Balance : Rows: 10001358 , Columns: 8

Validation: Rows: , Columns:

Summary statistics

Application Test:

```
[26]: datasets["application_test"].describe(include='all') #Look at all categorical and numerical
```

[26]:

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_G
count	48744.000000	48744	48744	48744	48744	48744.000000	4.874400e+04	4.874400e+04	48720.000000	
unique	NaN	2	2	2	2	NaN	NaN	NaN	NaN	
top	NaN	Cash loans	F	N	Y	NaN	NaN	NaN	NaN	
freq	NaN	48305	32678	32311	33658	NaN	NaN	NaN	NaN	
mean	277796.676350	NaN	NaN	NaN	NaN	0.397054	1.784318e+05	5.167404e+05	29426.240209	
std	103169.547296	NaN	NaN	NaN	NaN	0.709047	1.015226e+05	3.653970e+05	16016.368315	
min	100001.000000	NaN	NaN	NaN	NaN	0.000000	2.694150e+04	4.500000e+04	2295.000000	
25%	188557.750000	NaN	NaN	NaN	NaN	0.000000	1.125000e+05	2.606400e+05	17973.000000	
50%	277549.000000	NaN	NaN	NaN	NaN	0.000000	1.575000e+05	4.500000e+05	26199.000000	
75%	367555.500000	NaN	NaN	NaN	NaN	1.000000	2.250000e+05	6.750000e+05	37390.500000	
max	456250.000000	NaN	NaN	NaN	NaN	20.000000	4.410000e+06	2.245500e+06	180576.000000	

11 rows × 121 columns

Application Train:

```
[97]: datasets["application_train"].describe(include='all') #Look at all categorical and numerical
```

[97]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_AN
count	307511.000000	307511.000000	307511	307511	307511	307511	307511.000000	3.075110e+05	3.075110e+05	307499
unique	NaN	NaN	2	3	2	2	NaN	NaN	NaN	
top	NaN	NaN	Cash loans	F	N	Y	NaN	NaN	NaN	
freq	NaN	NaN	278232	202448	202924	213312	NaN	NaN	NaN	
mean	278180.518577	0.080729	NaN	NaN	NaN	NaN	0.417052	1.687979e+05	5.990260e+05	27108
std	102790.175348	0.272419	NaN	NaN	NaN	NaN	0.722121	2.371231e+05	4.024908e+05	14493
min	100002.000000	0.000000	NaN	NaN	NaN	NaN	0.000000	2.565000e+04	4.500000e+04	1615
25%	189145.500000	0.000000	NaN	NaN	NaN	NaN	0.000000	1.125000e+05	2.700000e+05	16524
50%	278202.000000	0.000000	NaN	NaN	NaN	NaN	0.000000	1.471500e+05	5.135310e+05	24903
75%	367142.500000	0.000000	NaN	NaN	NaN	NaN	1.000000	2.025000e+05	8.086500e+05	34596
max	456255.000000	1.000000	NaN	NaN	NaN	NaN	19.000000	1.170000e+08	4.050000e+06	258025

11 rows × 122 columns

Correlation with the target column

```
correlations = datasets["application_train"].corr()['TARGET'].sort_values()
print('Most Positive Correlations:\n', correlations.tail(10))
print('\nMost Negative Correlations:\n', correlations.head(10))
```

```
Most Positive Correlations:
 FLAG_DOCUMENT_3                0.044346
REG_CITY_NOT_LIVE_CITY          0.044395
FLAG_EMP_PHONE                  0.045982
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
DAYS_LAST_PHONE_CHANGE          0.055218
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_BIRTH                      0.078239
TARGET                          1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                 -0.178919
EXT_SOURCE_2                  -0.160472
EXT_SOURCE_1                  -0.155317
DAYS_EMPLOYED                 -0.044932
FLOORSMAX_AVG                 -0.044003
FLOORSMAX_MEDI                -0.043768
FLOORSMAX_MODE                -0.043226
AMT_GOODS_PRICE               -0.039645
REGION_POPULATION_RELATIVE    -0.037227
ELEVATORS_AVG                 -0.034199
Name: TARGET, dtype: float64
```

Observations from the above data:

1. Maximum positive correlation for the TARGET feature is 0.078239 which is observed with the DAYS_BIRTH feature.
2. Following this are the features REGION_RATING_CLIENT_W_CITY, REGION_RATING_CLIENT, DAYS_LAST_PHONE_CHANGE, DAYS_ID_PUBLISH,DAYS_ID_PUBLISH.
3. Another observation is a high value of indirect correlation between TARGET and FLOORS features,AMT_GOODS_PRICE and relative population features .

Applicant's House Wall Material Type

```
ap_train_data['WALLSMATERIAL_MODE'].value_counts()
```

```
Panel           66040
Stone, brick    64815
Block            9253
Wooden           5362
Mixed            2296
Monolithic       1779
Others           1625
Name: WALLSMATERIAL_MODE, dtype: int64
```

Applicant's House Type Part 2

```
ap_train_data['HOUSETYPE_MODE'].value_counts()
```

```
block of flats      150503
specific housing      1499
terraced house        1212
Name: HOUSETYPE_MODE, dtype: int64
```

Applicants already own Realty?

```
ap_train_data['FLAG_OWN_REALTY'].value_counts()
```

```
Y    213312
N     94199
Name: FLAG_OWN_REALTY, dtype: int64
```

For previous Application DB:

Intersection of SK_ID_CURR for Previous Application and application_train

```
#Find the intersection of two arrays.
print(f'Number of train applicants with previous applications is {len(np.intersect1d(datasets["previous_application"]["SK_ID_CURR"], datasets["application_train"]["
```

```
Number of train applicants with previous applications is 291,057
```

Intersection of SK_ID_CURR for Previous Application and application_train

```
#Find the intersection of two arrays.
print(f'Number of train applicants with previous applications is {len(np.intersect1d(datasets["previous_application"]["SK_ID_CURR"], datasets["application_test"]["S
```

```
Number of train applicants with previous applications is 47,800
```

Visual EDA:

Missing data for application train

```
percent = (datasets["application_train"].isnull().sum()/datasets["application_train"].isnull().count()*100).sort_values(ascending = False).round(2)
sum_missing = datasets["application_train"].isna().sum().sort_values(ascending = False)
missing_application_train_data  = pd.concat([percent, sum_missing], axis=1, keys=['Percent', "Train Missing Count"])
missing_application_train_data.head(20)
```

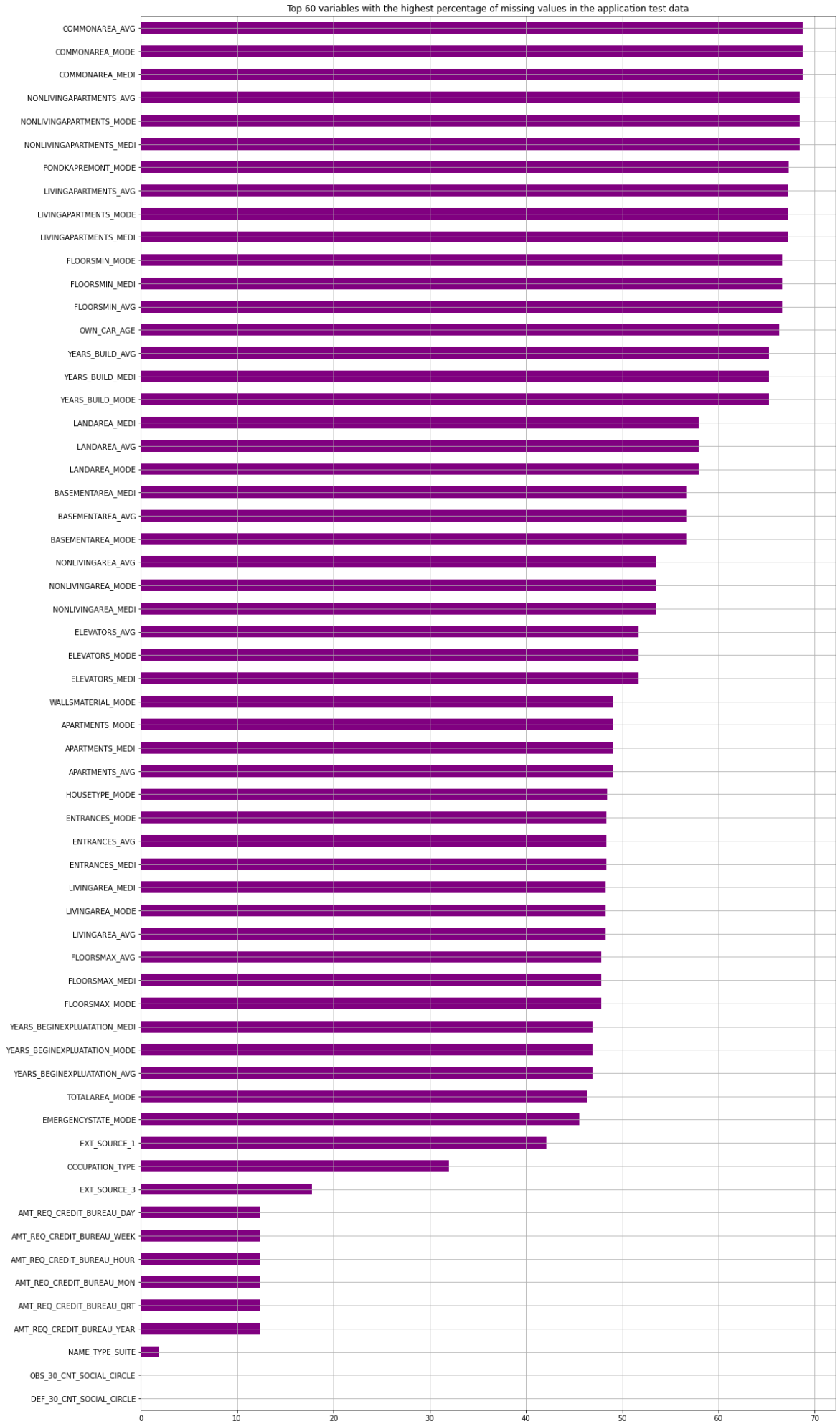
	Percent	Train Missing Count
COMMONAREA_MEDI	69.87	214865
COMMONAREA_AVG	69.87	214865
COMMONAREA_MODE	69.87	214865
NONLIVINGAPARTMENTS_MODE	69.43	213514
NONLIVINGAPARTMENTS_AVG	69.43	213514
NONLIVINGAPARTMENTS_MEDI	69.43	213514
FONDKAPREMONT_MODE	68.39	210295
LIVINGAPARTMENTS_MODE	68.35	210199
LIVINGAPARTMENTS_AVG	68.35	210199
LIVINGAPARTMENTS_MEDI	68.35	210199
FLOORSMIN_AVG	67.85	208642
FLOORSMIN_MODE	67.85	208642
FLOORSMIN_MEDI	67.85	208642
YEARS_BUILD_MEDI	66.50	204488
YEARS_BUILD_MODE	66.50	204488
YEARS_BUILD_AVG	66.50	204488
OWN_CAR_AGE	65.99	202929
LANDAREA_MEDI	59.38	182590
LANDAREA_MODE	59.38	182590
LANDAREA_AVG	59.38	182590

Missing data for application test

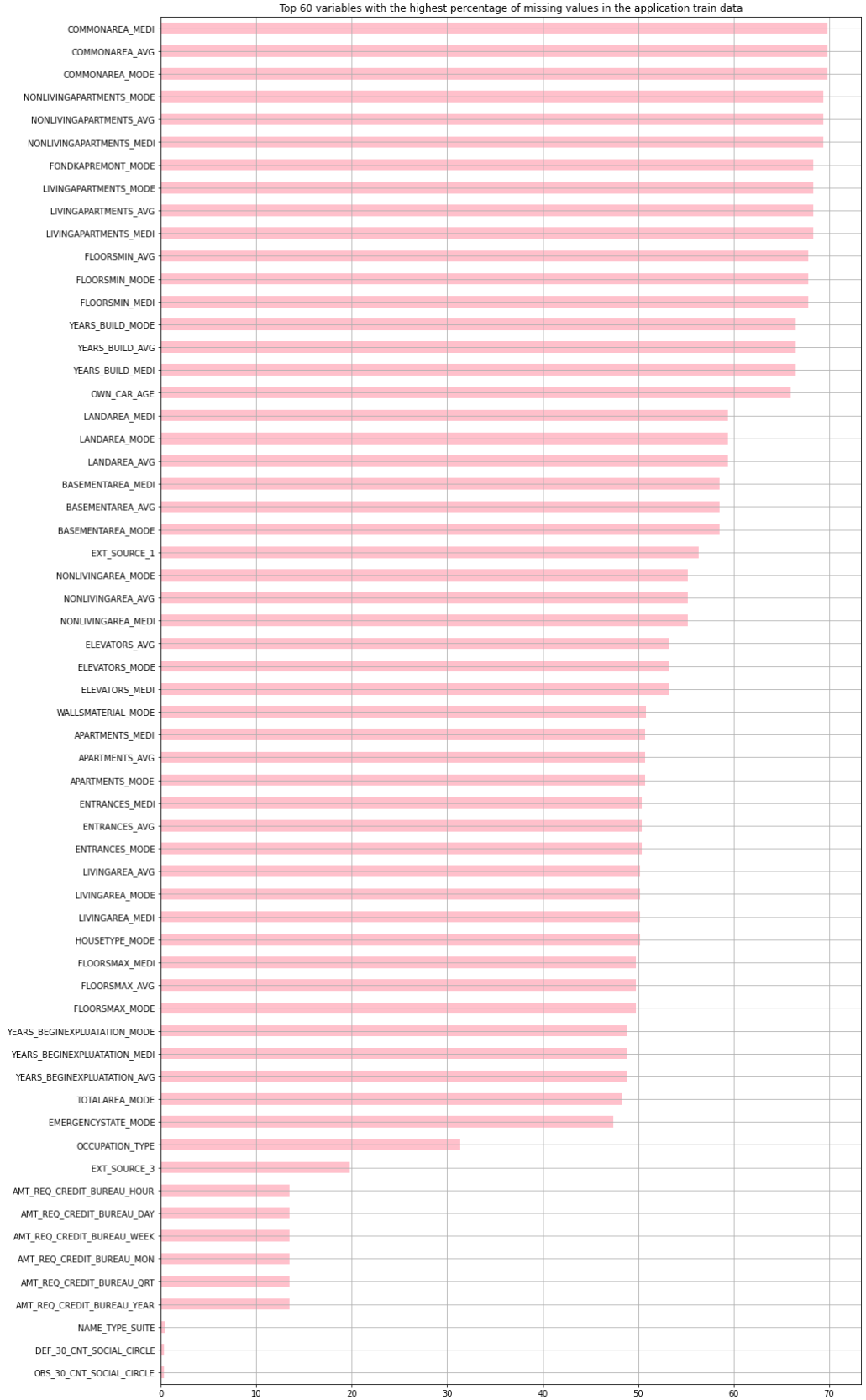
```
percent = (datasets["application_test"].isnull().sum()/datasets["application_test"].isnull().count()*100).sort_values(ascending = False).round(2)
sum_missing = datasets["application_test"].isna().sum().sort_values(ascending = False)
missing_application_test_data  = pd.concat([percent, sum_missing], axis=1, keys=['Percent', "Test Missing Count"])
missing_application_test_data.head(20)
```

	Percent	Test Missing Count
COMMONAREA_AVG	68.72	33495
COMMONAREA_MODE	68.72	33495
COMMONAREA_MEDI	68.72	33495
NONLIVINGAPARTMENTS_AVG	68.41	33347
NONLIVINGAPARTMENTS_MODE	68.41	33347
NONLIVINGAPARTMENTS_MEDI	68.41	33347
FONDKAPREMONT_MODE	67.28	32797
LIVINGAPARTMENTS_AVG	67.25	32780
LIVINGAPARTMENTS_MODE	67.25	32780
LIVINGAPARTMENTS_MEDI	67.25	32780
FLOORSMIN_MEDI	66.61	32466
FLOORSMIN_AVG	66.61	32466
FLOORSMIN_MODE	66.61	32466
OWN_CAR_AGE	66.29	32312
YEARS_BUILD_AVG	65.28	31818
YEARS_BUILD_MEDI	65.28	31818
YEARS_BUILD_MODE	65.28	31818
LANDAREA_MEDI	57.96	28254
LANDAREA_AVG	57.96	28254
LANDAREA_MODE	57.96	28254

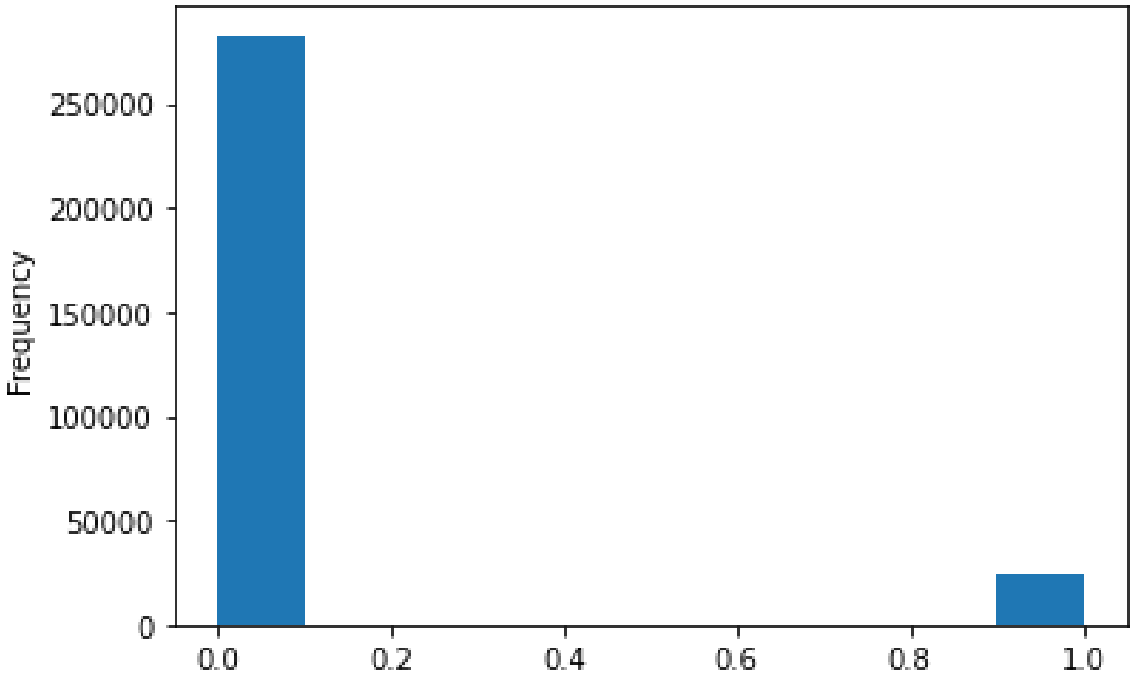
Top 60 variables with the highest percentage of missing values in the application test data

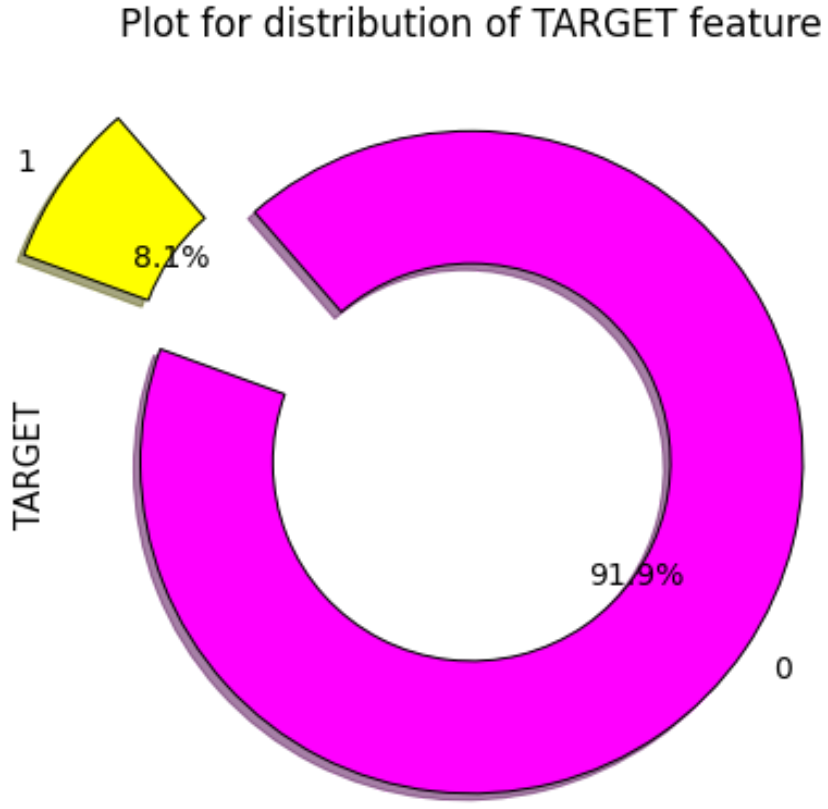

Top 60 variables with the highest percentage of missing values in the application test data
COMMONAREA_AVG
COMMONAREA_MODE
COMMONAREA_MEDI
NONLIVINGAPARTMENTS_AVG
NONLIVINGAPARTMENTS_MODE
NONLIVINGAPARTMENTS_MEDI
FONDKAPREMONT_MODE
LIVINGAPARTMENTS_AVG
LIVINGAPARTMENTS_MODE
LIVINGAPARTMENTS_MEDI
FLOORSMIN_MODE
FLOORSMIN_MEDI
FLOORSMIN_AVG
OWN_CAR_AGE
YEARS_BUILD_AVG
YEARS_BUILD_MEDI
YEARS_BUILD_MODE
LANDAREA_MEDI
LANDAREA_AVG
LANDAREA_MODE
BASEMENTAREA_MEDI
BASEMENTAREA_AVG
BASEMENTAREA_MODE
NONLIVINGAREA_AVG
NONLIVINGAREA_MODE
NONLIVINGAREA_MEDI
ELEVATORS_AVG
ELEVATORS_MODE
ELEVATORS_MEDI
WALLSMATERIAL_MODE
APARTMENTS_MODE
APARTMENTS_MEDI
APARTMENTS_AVG
HOUSETYPE_MODE
ENTRANCES_MODE
ENTRANCES_AVG
ENTRANCES_MEDI
LIVINGAREA_MEDI
LIVINGAREA_MODE
LIVINGAREA_AVG
FLOORSMAX_AVG
FLOORSMAX_MEDI
FLOORSMAX_MODE
YEARS_BEGINEXPLUATATION_MEDI
YEARS_BEGINEXPLUATATION_MODE
YEARS_BEGINEXPLUATATION_AVG
TOTALAREA_MODE
EMERGENCYSTATE_MODE
EXT_SOURCE_1
OCCUPATION_TYPE
EXT_SOURCE_3
AMT_REQ_CREDIT_BUREAU_DAY
AMT_REQ_CREDIT_BUREAU_WEEK
AMT_REQ_CREDIT_BUREAU_HOUR
AMT_REQ_CREDIT_BUREAU_MON
AMT_REQ_CREDIT_BUREAU_QRT
AMT_REQ_CREDIT_BUREAU_YEAR
NAME_TYPE_SUITE
OBS_30_CNT_SOCIAL_CIRCLE
DEF_30_CNT_SOCIAL_CIRCLE
0
10
20
30
40
50
60
70

Top 60 variables with the highest percentage of missing values in the application train data

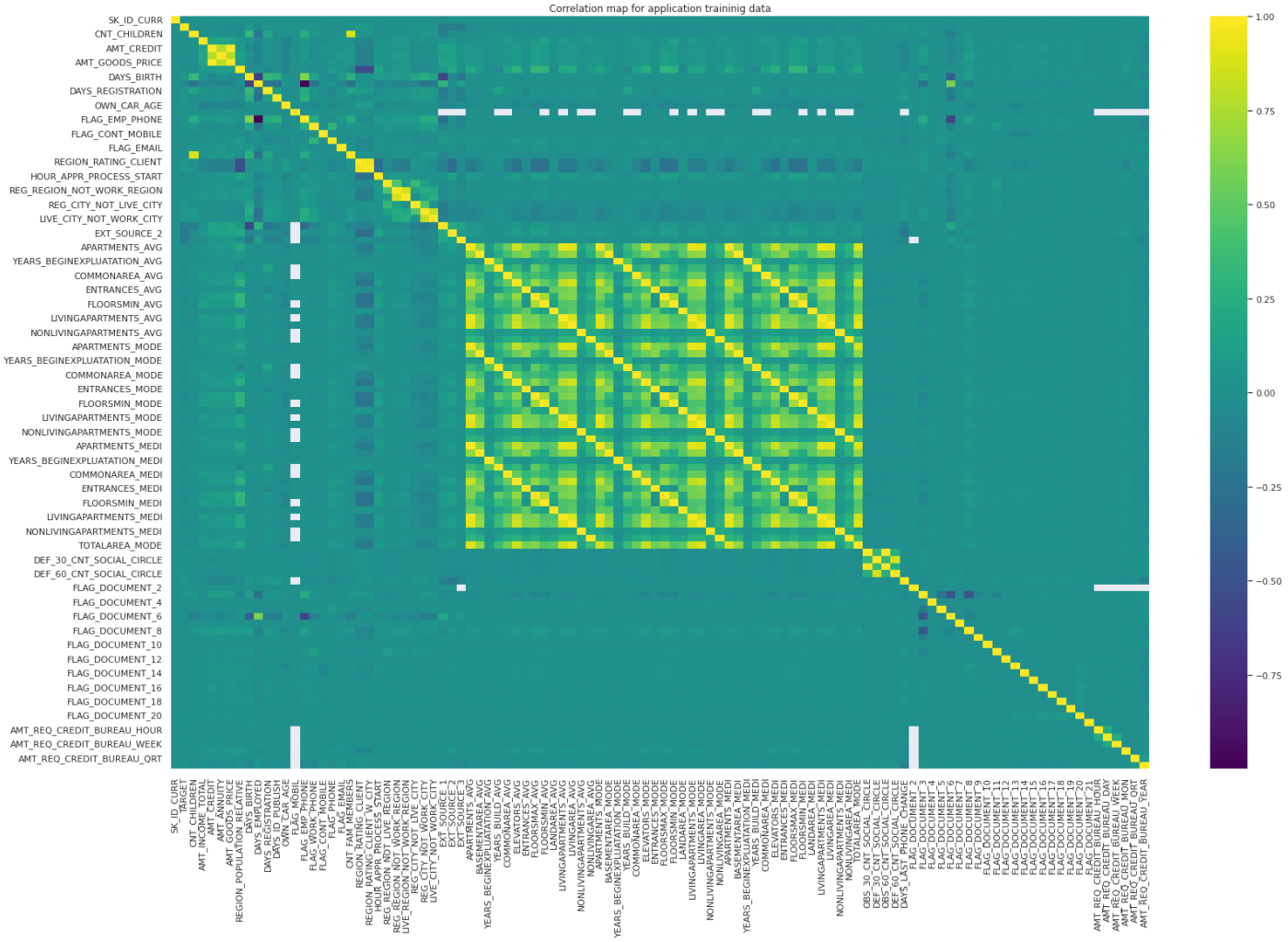


Distribution of the target column



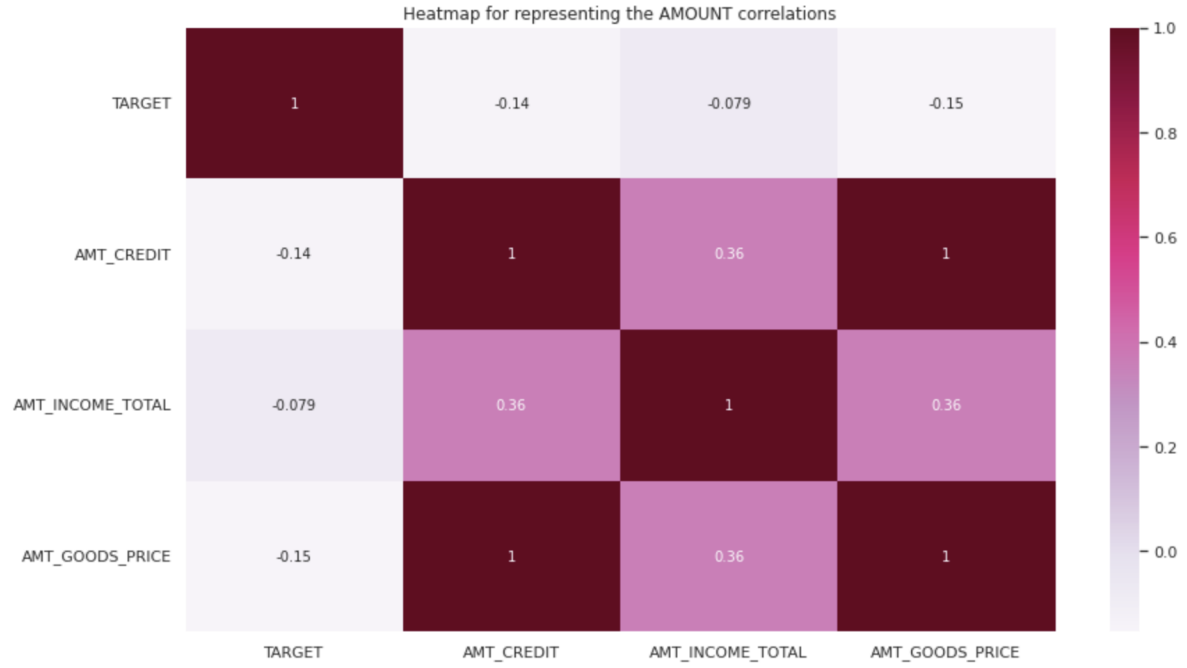


Correlation with the Application Train Data columns



Observations from the above plot: -The heatmap is a bit ambiguous to read as we have 122 columns to compare from.

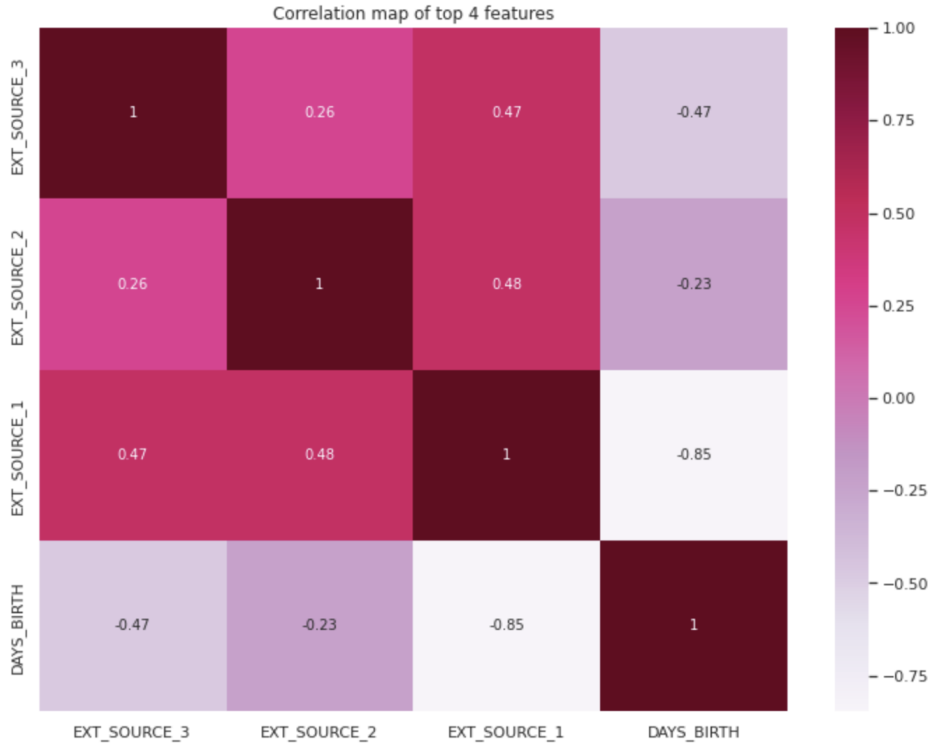
Correlation Map for target column with the Amount Columns



Observations from the above data: -

The above plot is the correlation between the target column and the amount columns, with the plot we can clearly depict from the plot that target column has very low correlation with the amount columns.

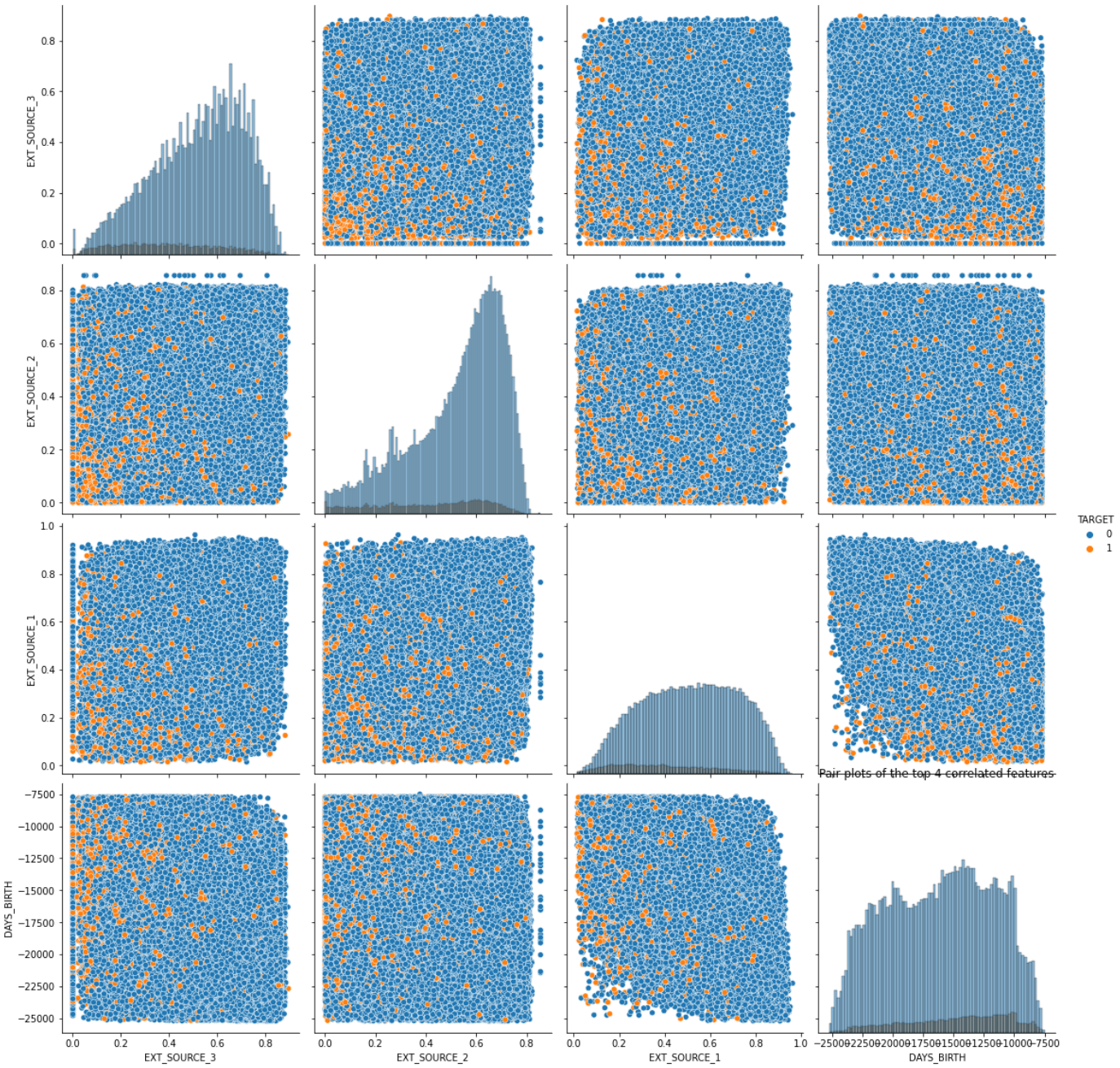
Correlation Map for top 4 features:



Observation based on above

- The heatmap demonstrates how extrinsic factors have a secondary impact on the TARGET characteristic.
- But, it is also clear from the correlations between them that multicollinearity exists.

Pair plots of the top 4 correlated features

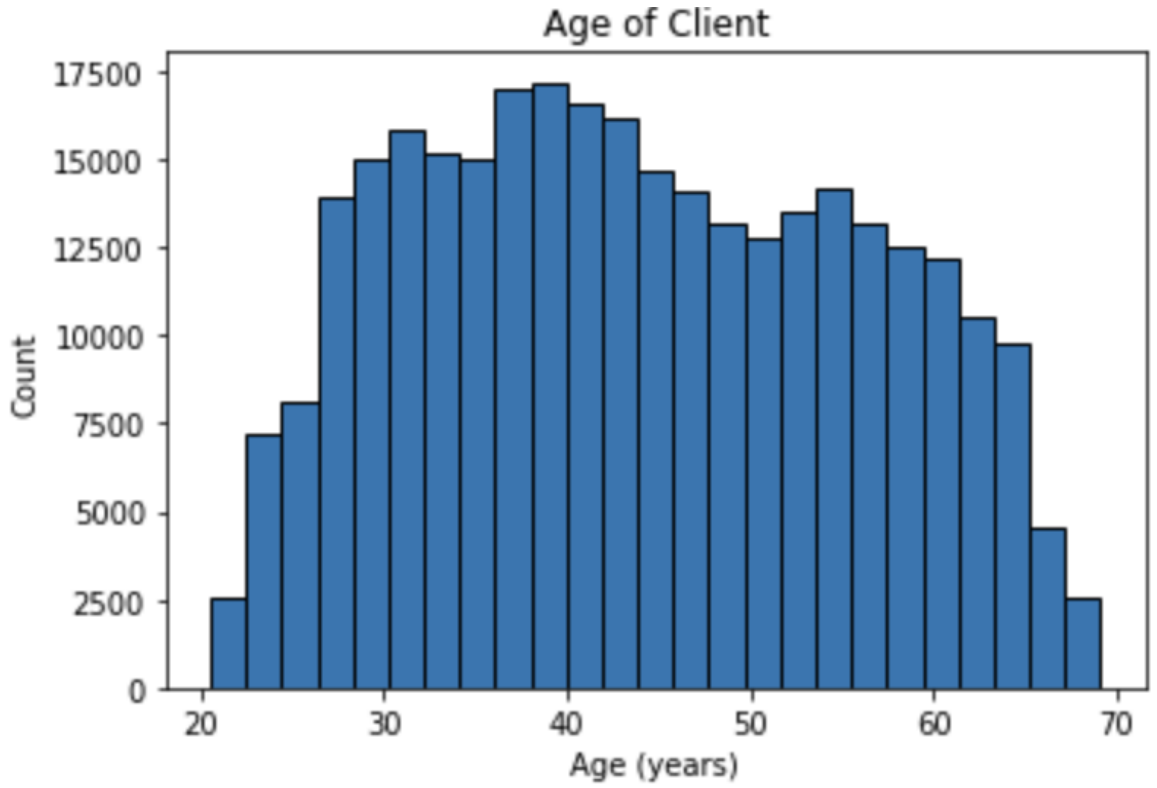


Observation based on above

- The pair plot matrix demonstrates how extrinsic factors have a secondary impact on the TARGET characteristic, which is used \

- But, it is also clear from the correlations between them that multicollinearity exists.

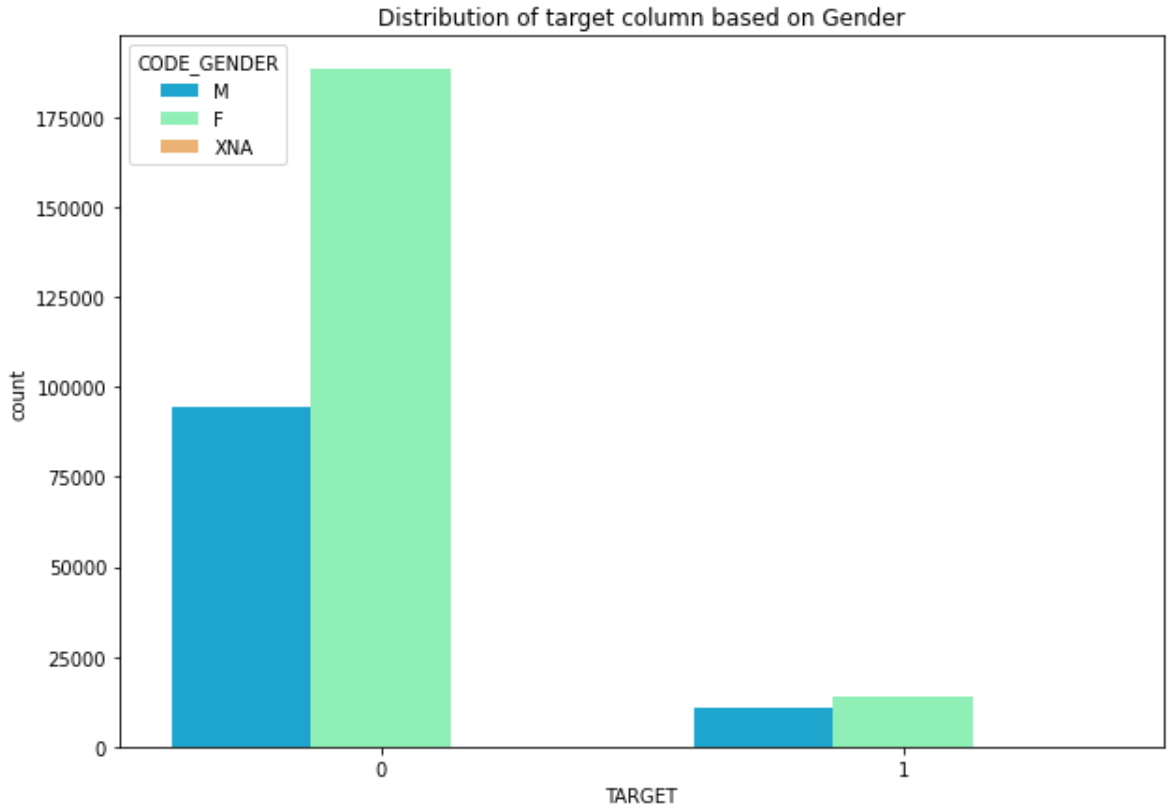
Distribution for Applicants Age



Observations from the above plot

- The DAYS_BIRTH feature, which contains negative values, is used to calculate age. This is inconsistent and needs to be fixed.
- When we plot age as a function of years, we observe a pretty normal distribution, which is encouraging in a challenging dataset where the DAYS BIRTH feature is substantially linked with the TARGET feature.

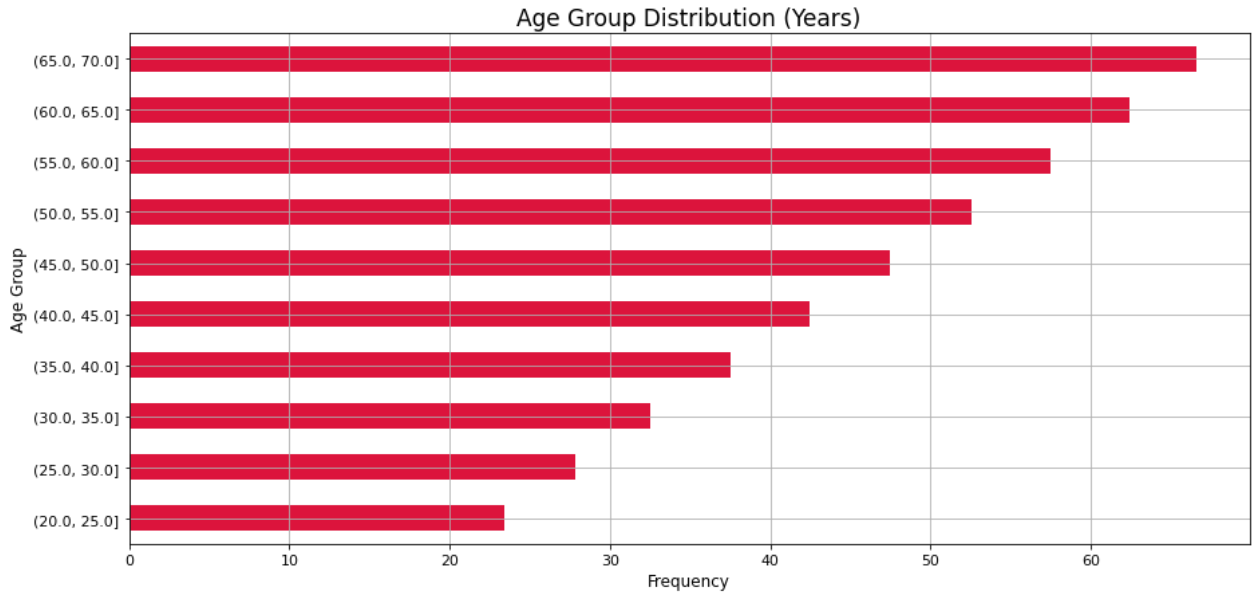
Distribution for Target Column based on Gender



Observation from the above plot

The purpose of this code is to group the 'YEARS_BIRTH' column into different age groups and analyze the distribution of 'TARGET' variable in each age group.

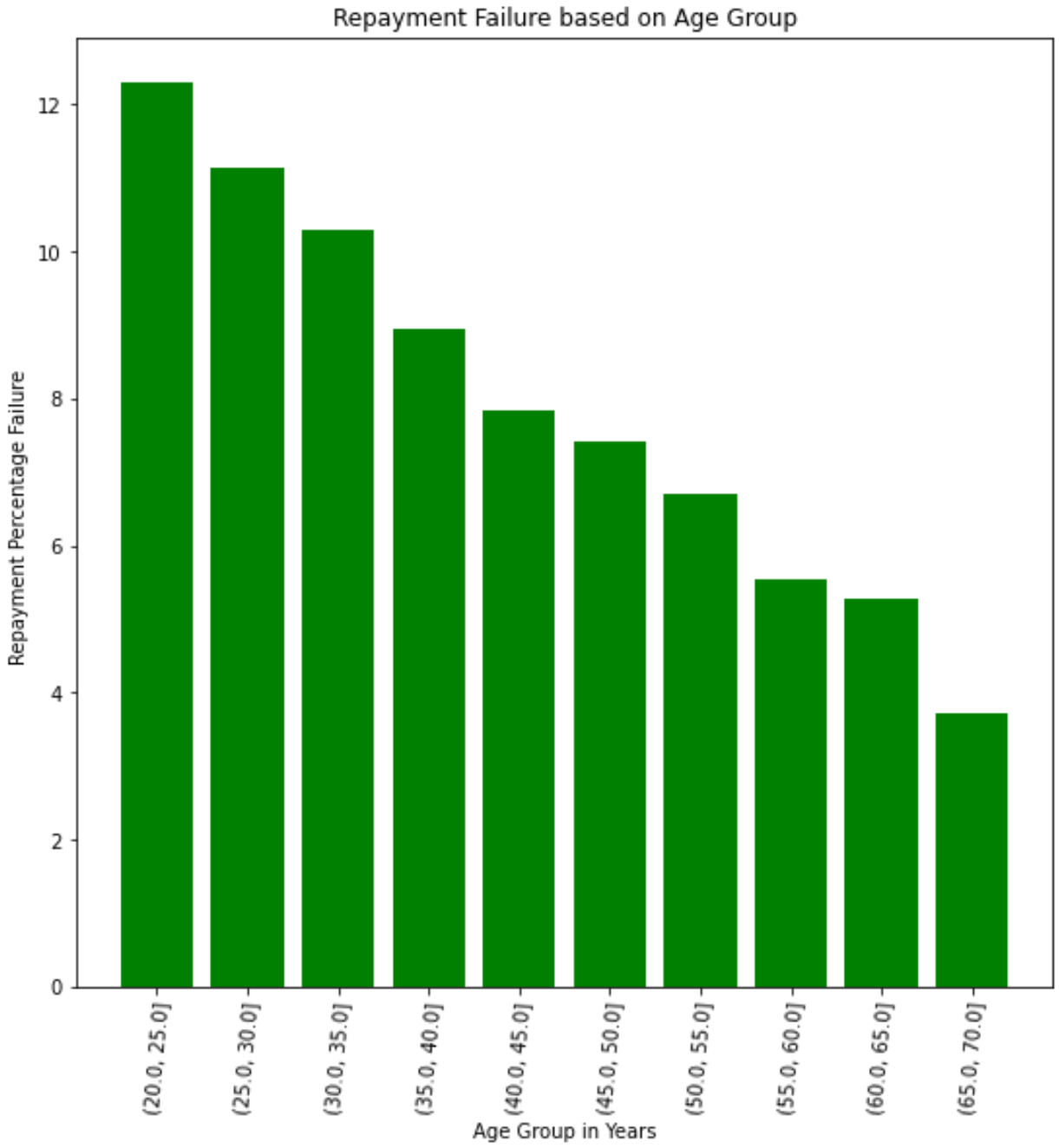
Age Group Distribution (Years)



Observation from the above plot-

Following binning, we see that older persons have a propensity to take out more loans than younger people.

Repayment Failure based on Age Group

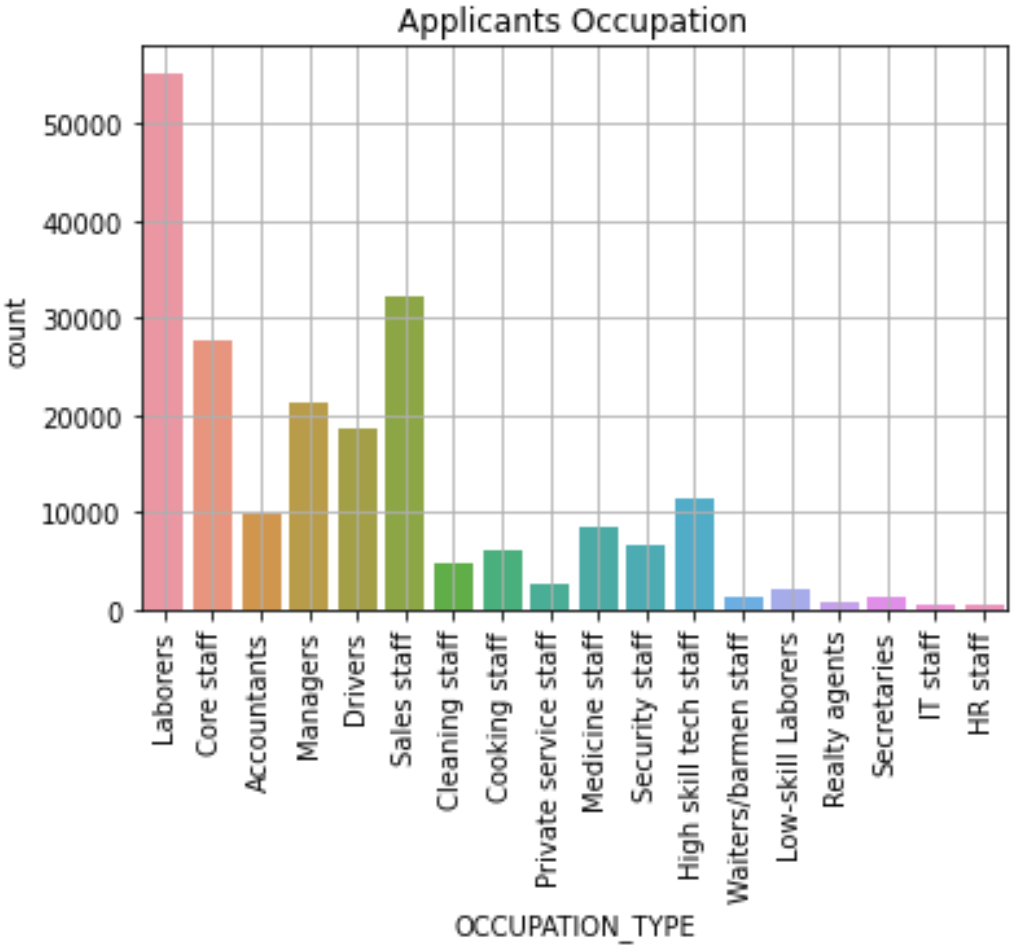


Observations from the above plot: The Binning method is utilized to provide precise information regarding age.

Now, it can generally be deduced that the age range of 20 to 25 years is where the most majority of people who are essentially unable to essentially repay the loan on time for the most part reside. Contrary to popular opinion, a decline in the number of failures is typically observed as age

increases.Younger applicants are actually more likely to default on the loan, showing that it can now be generally deduced that the majority of borrowers who normally are unable to return the loan on time type of fall into the age range of 20–25 years, which is actually quite substantial.Younger applicants are generally more likely to default on a loan, so banks should generally be advised against doing so. This proves that, contrary to popular belief, the age group of 20 to 25 years is actually where the majority of borrowers fail to repay loans in a timely manner.

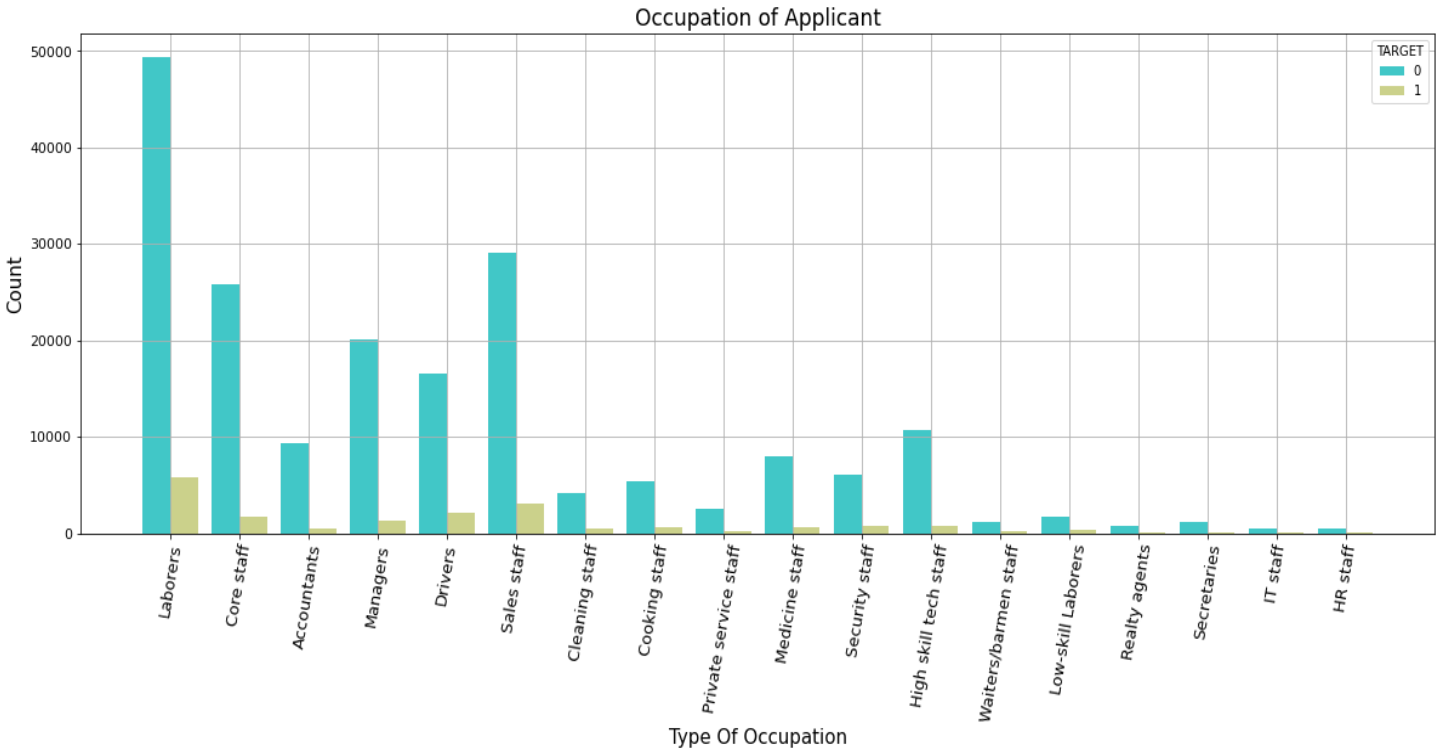
Applicants occupations



Observation based on plot:

Contrary to popular opinion, the majority of the candidates' occupations are studied here. It is generally accepted that laborers kind of tend to kind of take the loan most fundamentally followed by sales staff, demonstrating how this is generally accepted that laborers kind of tend to kind of take the loan most fundamentally followed by sales staff in a subtle way.

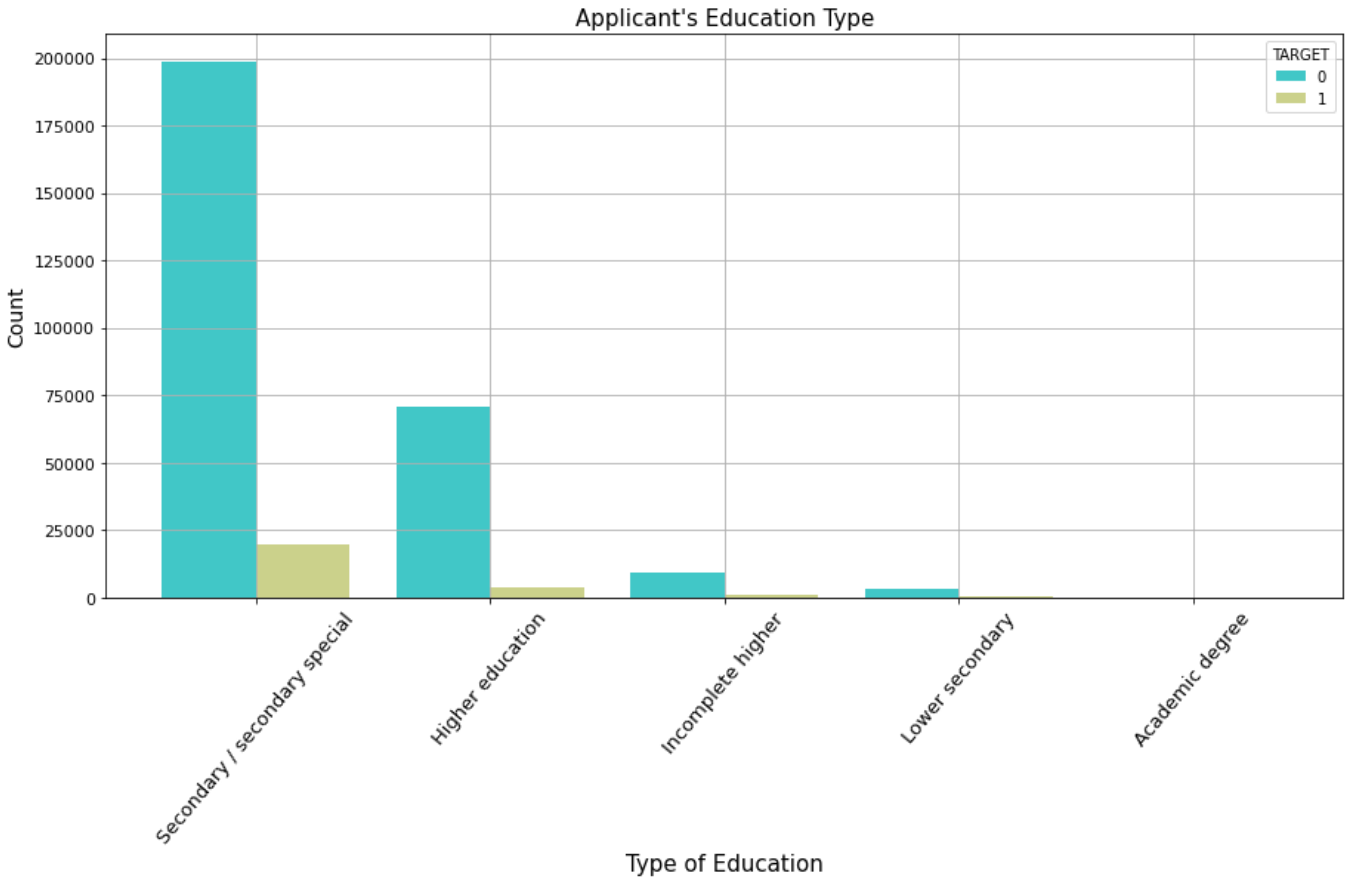
Occupation of Applicant:



Observation of the above plot

- There are 18 distinct occupations represented among the borrowers, with laborers, salespeople, core employees, managers, and drivers at the top.
- There is no clear pattern in the occupation classes whose borrowers effectively repay their loans.

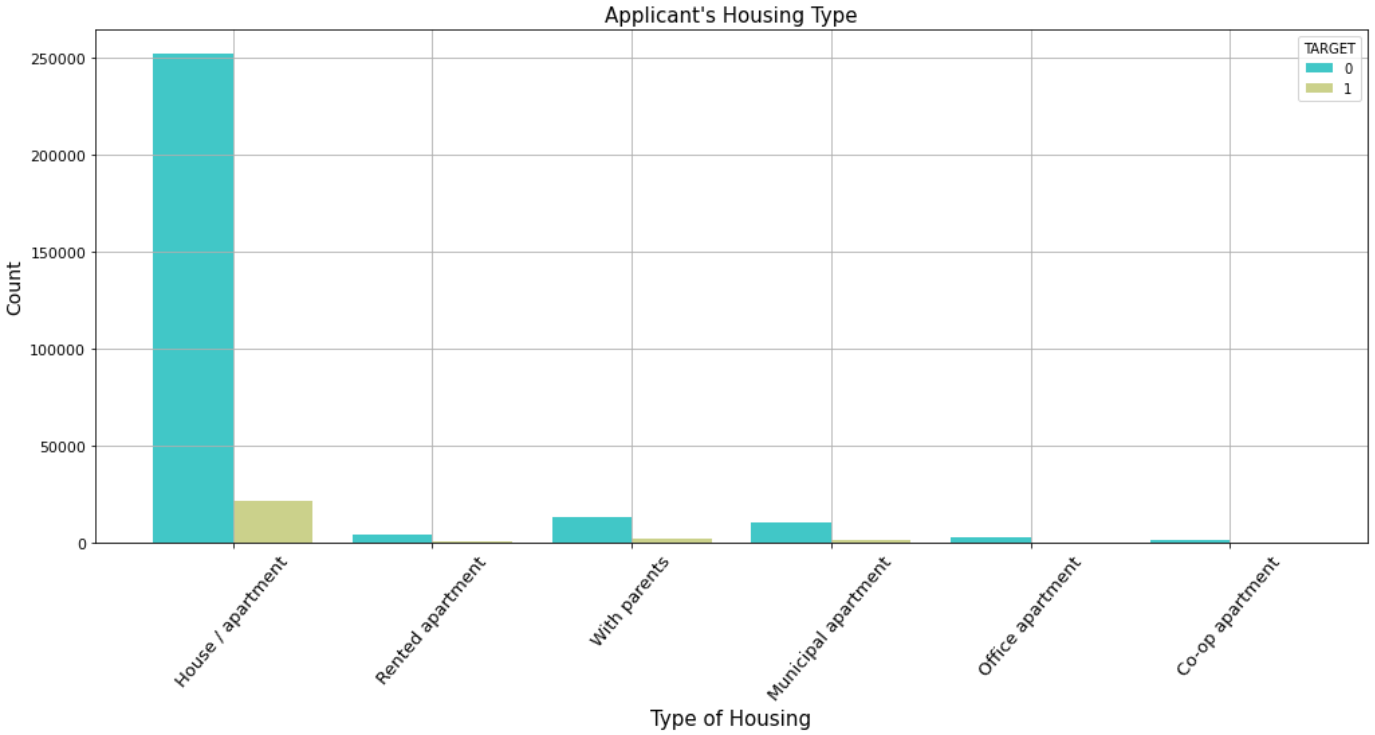
Applicant's Education Type



Observation from the plot

- The majority of candidates have the greatest levels of secondary and higher education, with academic degrees being the lowest.

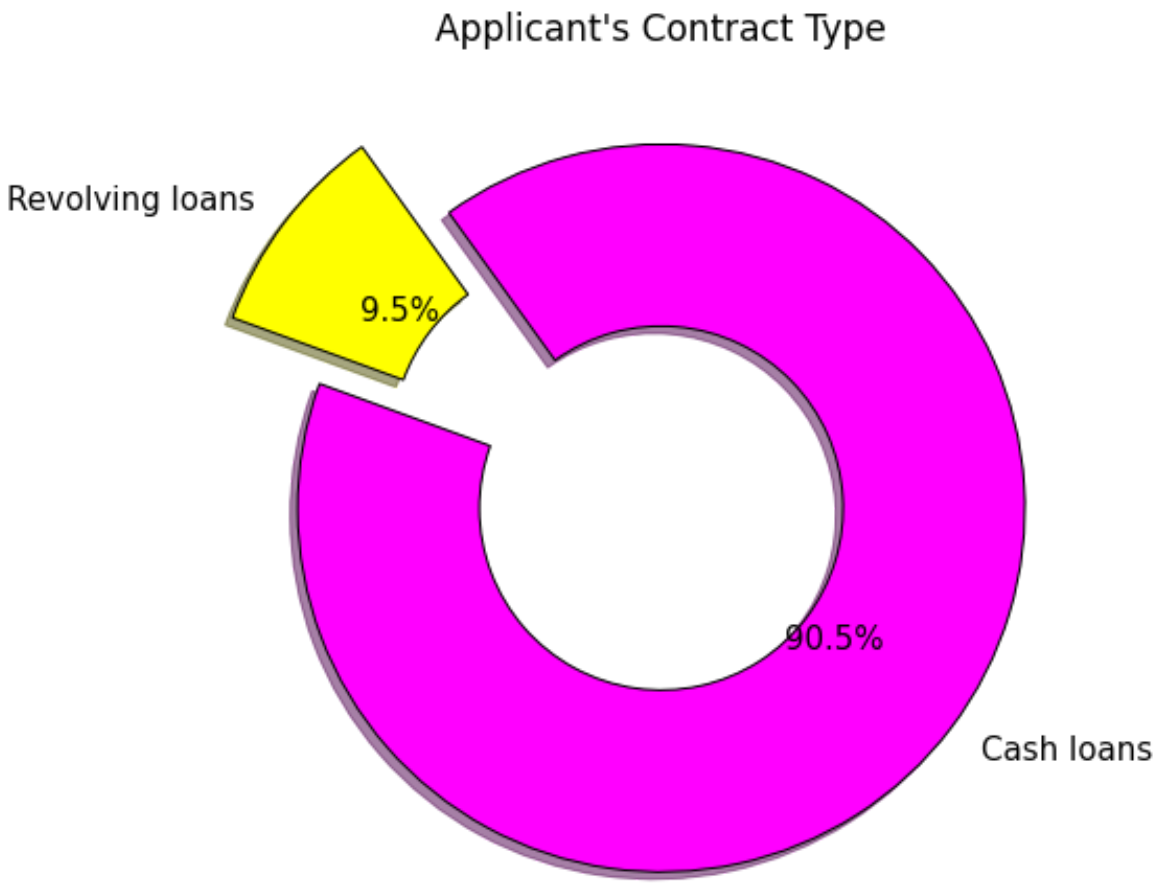
Applicant Housing Type versus TARGET



Observation from the plot

- The applicant now lives in houses or apartments the majority of the time, followed by municipal apartments and living with parents while looking for a new place for themselves.

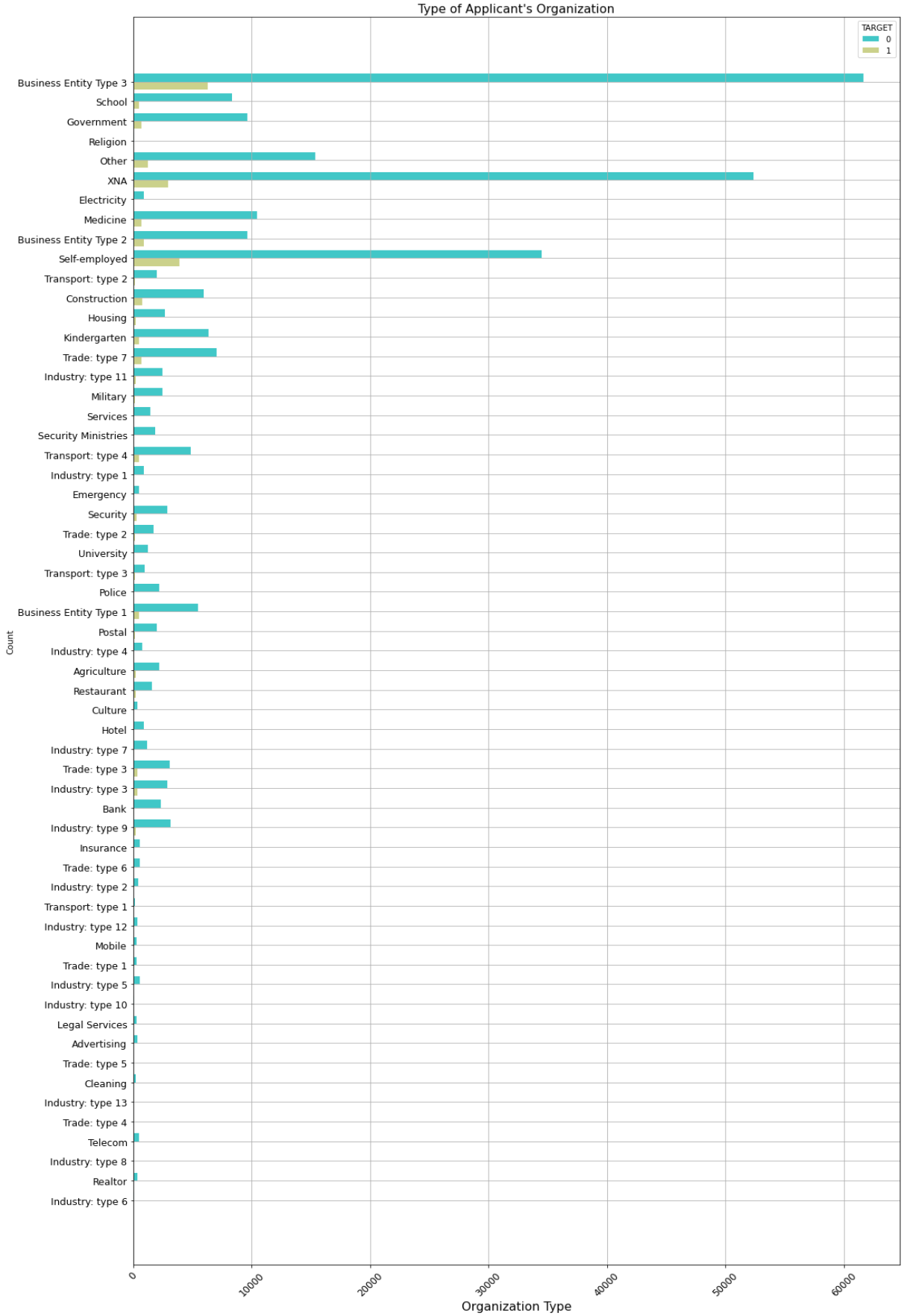
Applicant Contract Type



Observation from the plot

- Cash loans makeup 90.5% of all loan contracts, while revolving loans, which can be repaid and refinanced repeatedly, account for the remaining 9.5%.

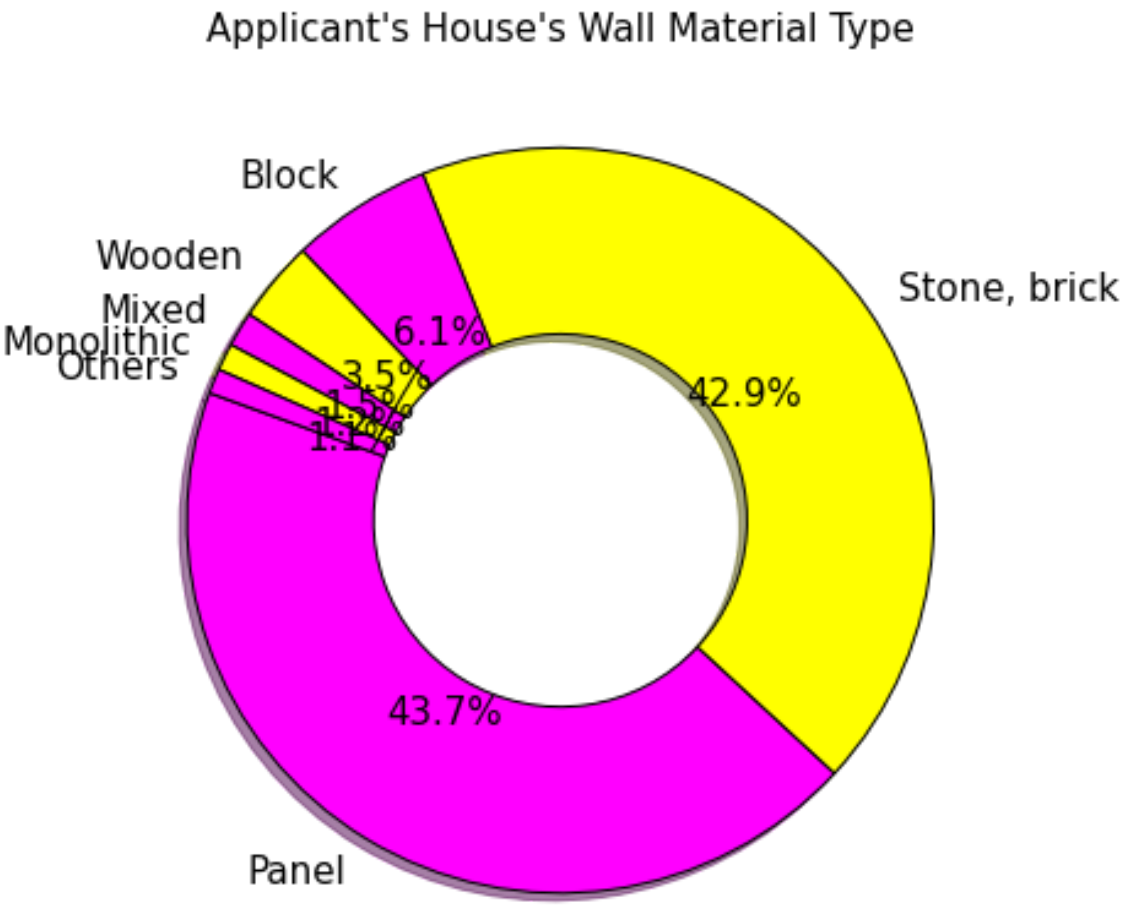
Applicant Organization Type



Observation from the plot

- The application pool includes a variety of organization kinds, with the majority coming from Type 3 commercial entities and self-employment.

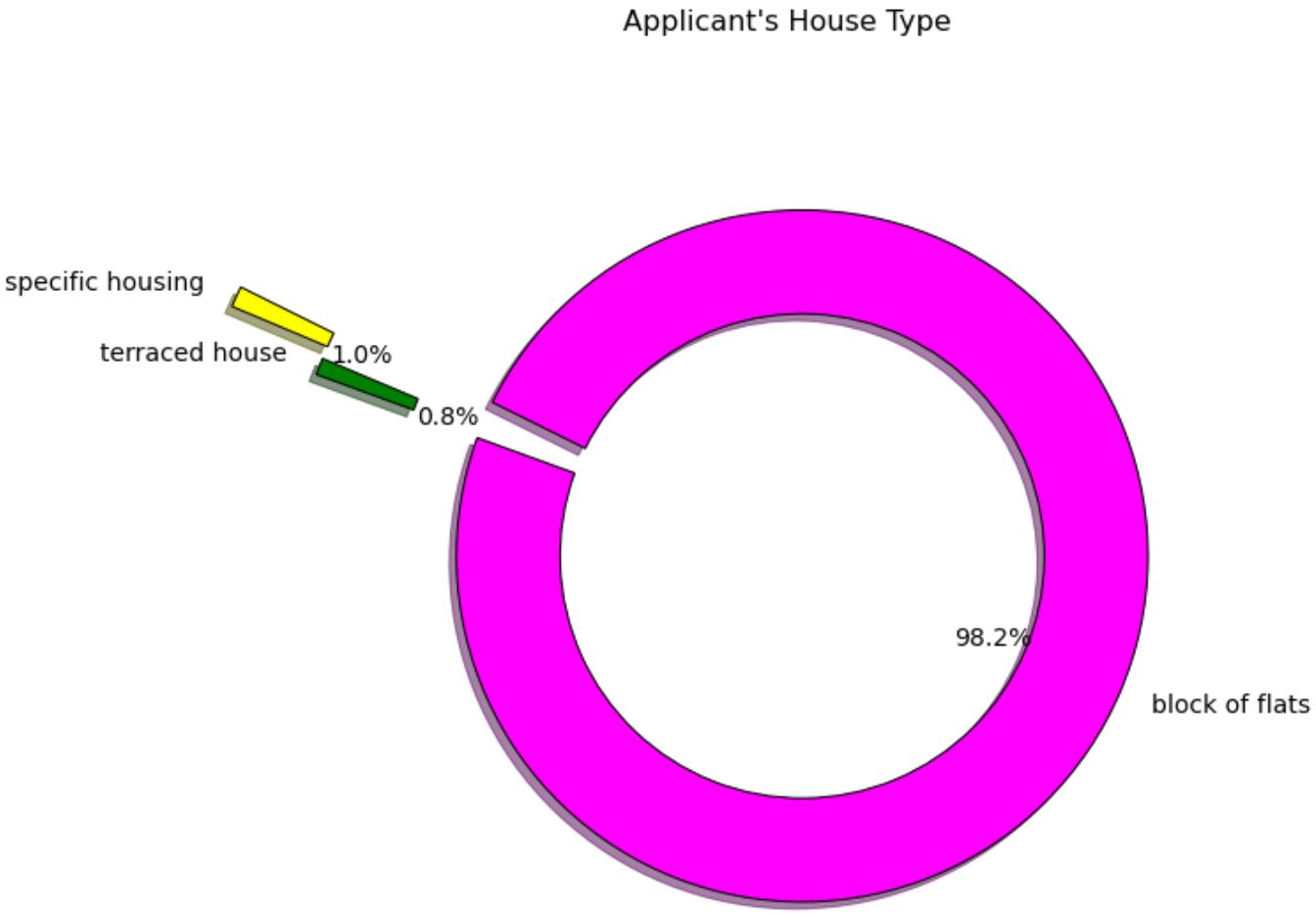
Applicant's House Wall Material Type



Observation from the plot

- The most common wall materials in applicant's homes are panels, stones, and bricks, followed by cement blocks, wood, or a combination of the aforementioned.

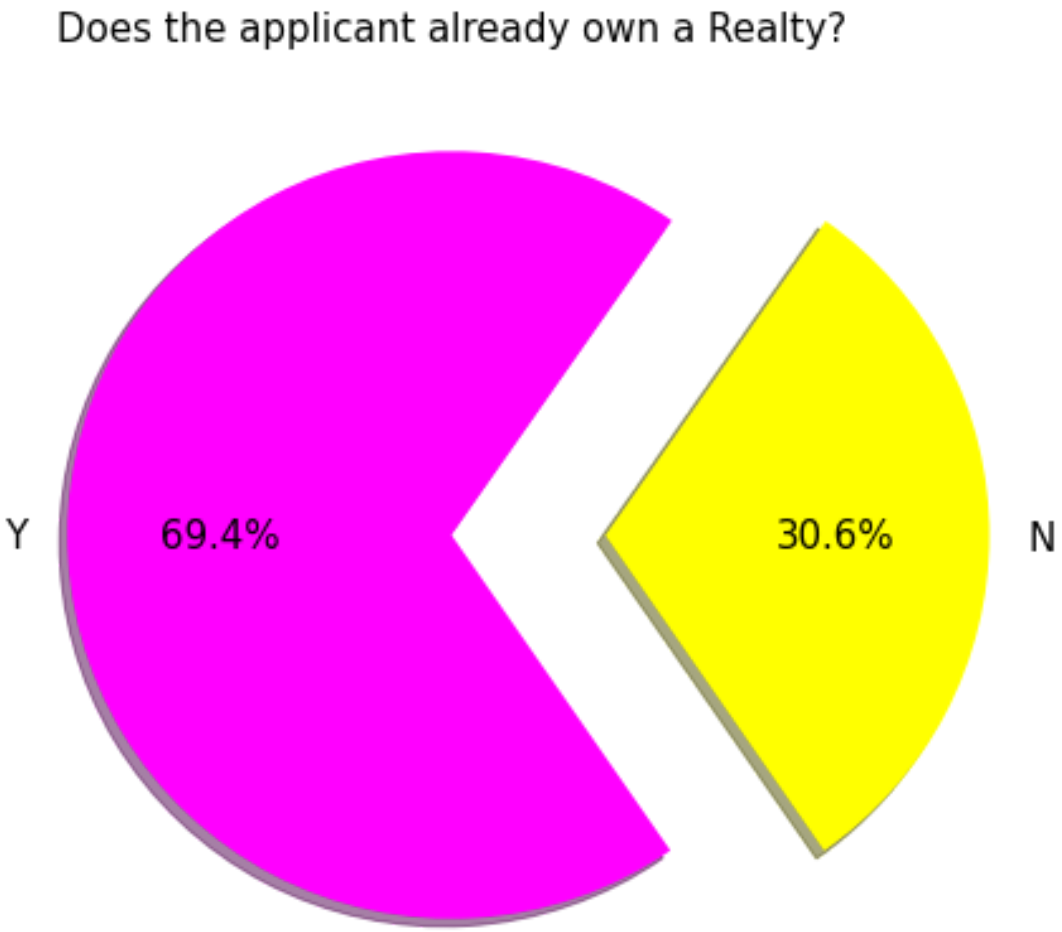
Applicant's House Type Part 2



Observation from the plot

- Applicants mostly reside in flats (more than 98%) while the remaining either live in terraced or other specific house types.

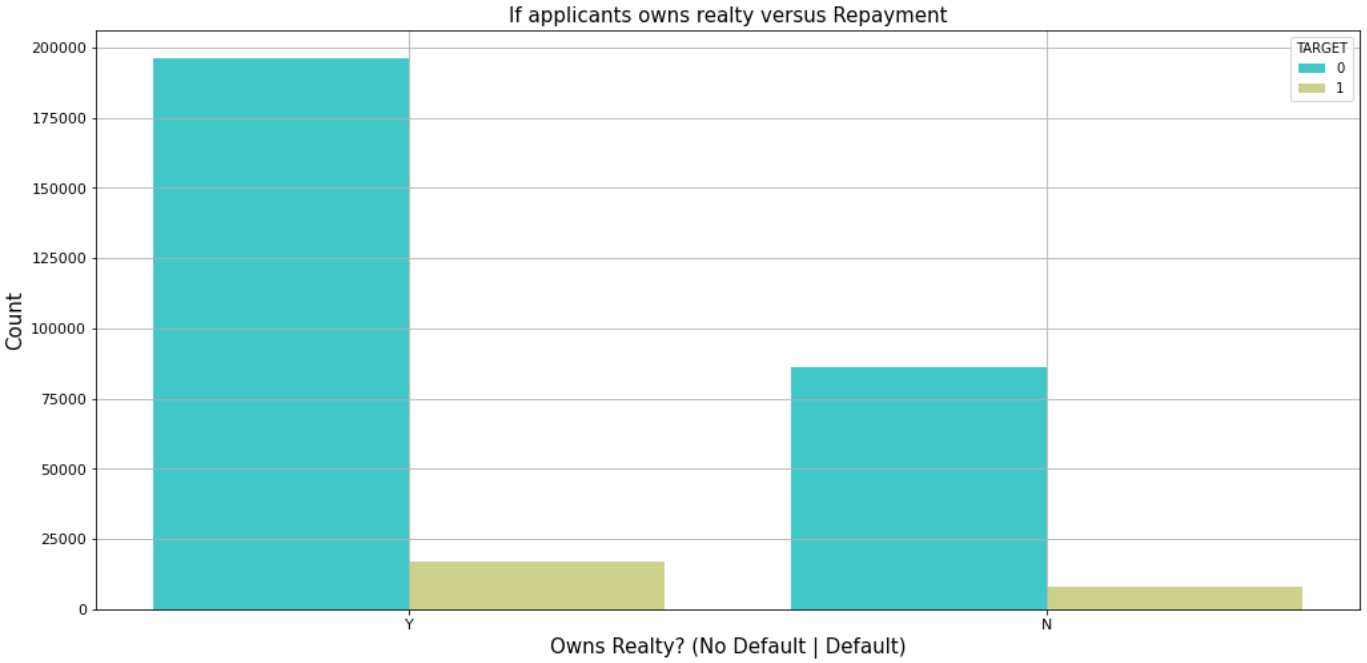
Applicant already own Realty?



Observation from the plot

- 69.4% of applicants have a home of their own. Check out the repayment's distribution.

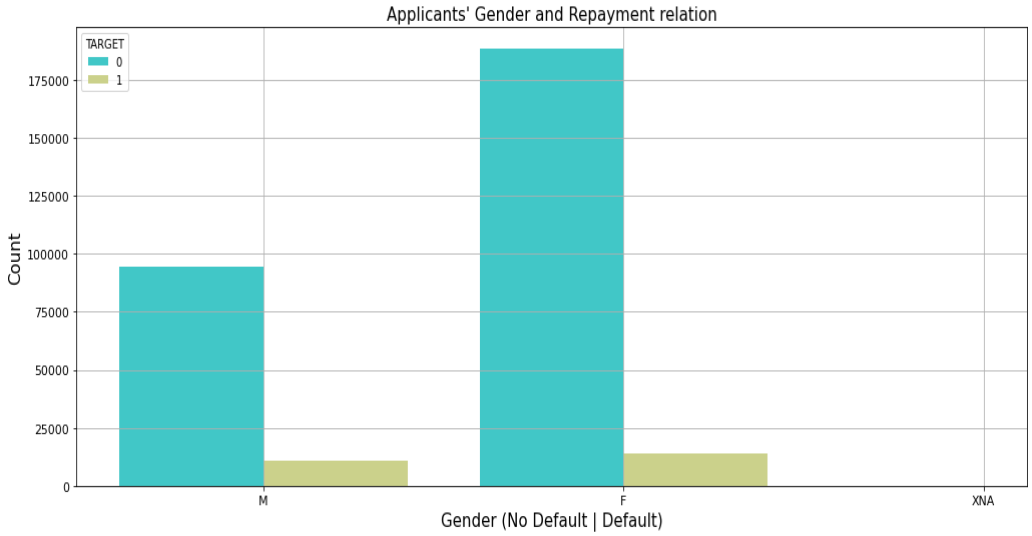
If applicants owns realty versus Repayment



Observation from the above plot:

-The majority of applicants in either class are not defaulters. Less than 25000 applicants have real estate and are behind on their payments.

Gender wise repayment analysis



Observations from the above plot:

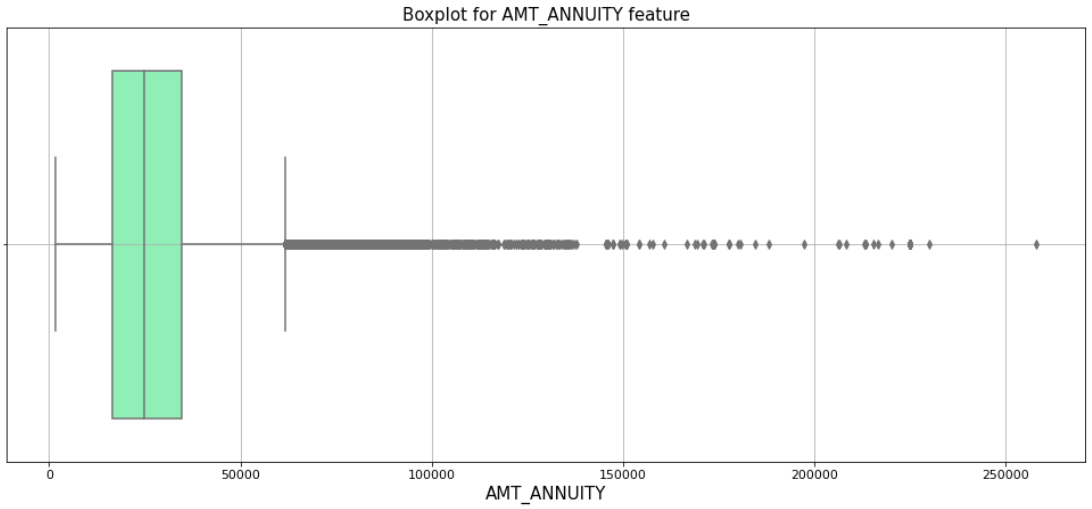
- Female applicants make up the majority, and the majority of them have no history of defaults.
- In the case of male candidates, it is clear that a disproportionately higher percentage of applicants default.

AMT_CREDIT feature distribution Plot

Observations from the plot:

- The feature is right-skewed, as is evident.
- Sizing could aid in implementing this functionality effectively.

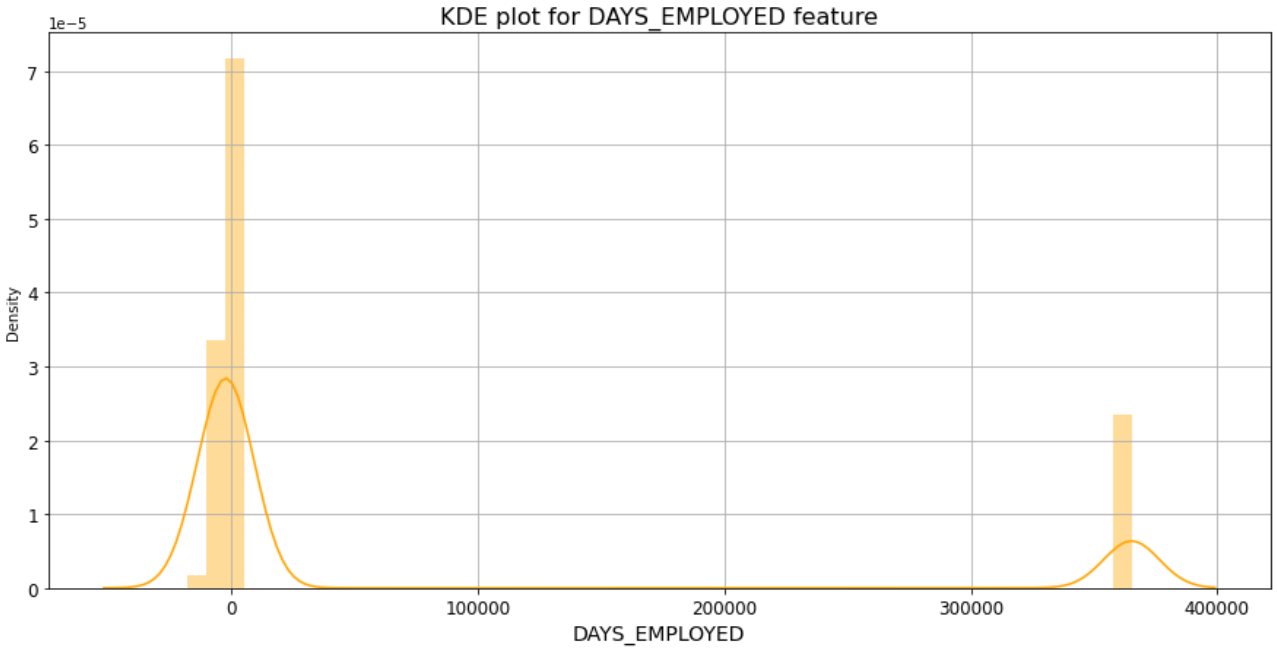
AMT_ANNUITY feature distribution Plot



Observations from the plot

- A right-skewed feature with numerous outliers is once again visible.
- We can't get rid of these outliers because doing so could mean losing crucial data.

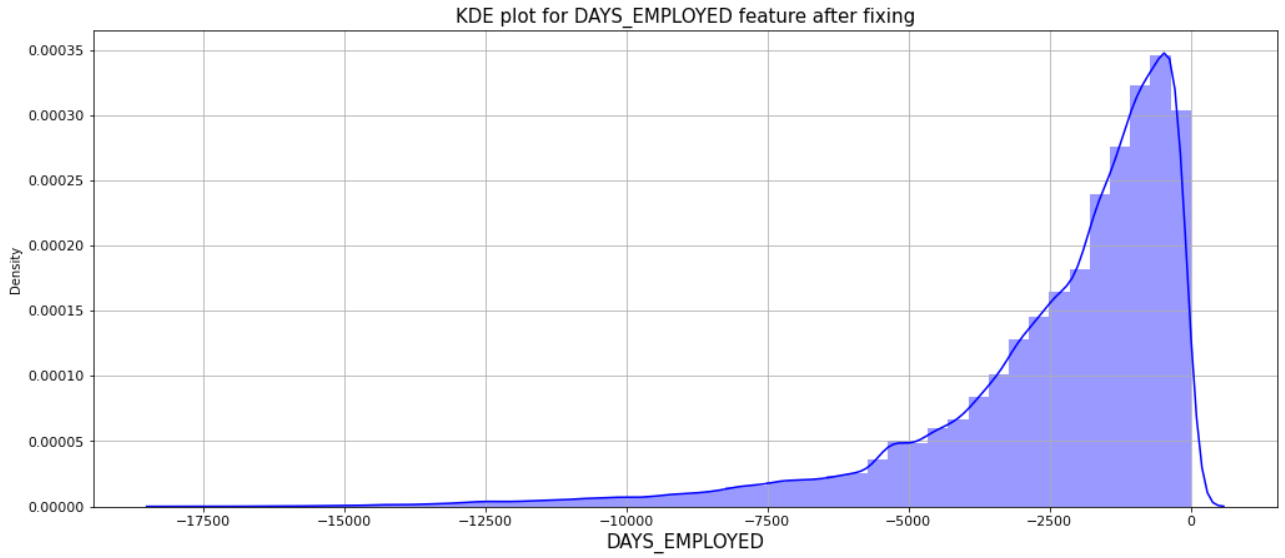
DAYS_EMPLOYED feature distribution Plot



Observations from the plot

- The negative days values for this feature are the same as those for DAYS_BIRTH.
- But, we notice a strange anomaly in this situation: the maximum number of days that can be worked is 365243, or a thousand years.
- We won't analyze the anomaly, we'll just ignore it and go ahead and examine the feature distribution once more.

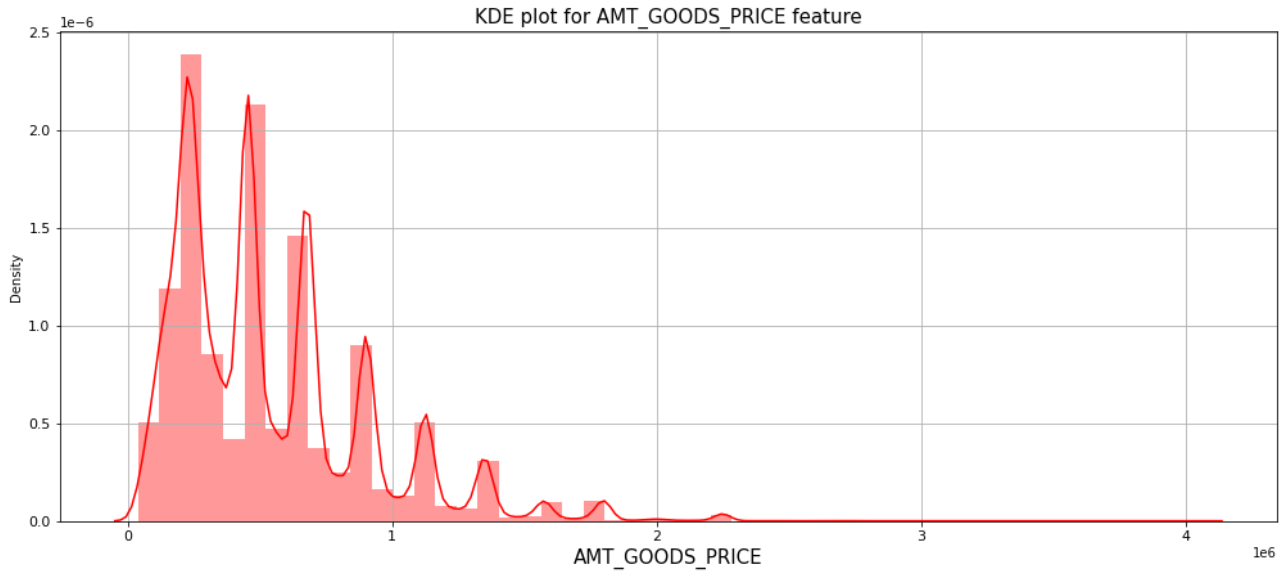
Distribution of DAYS_EMPLOYED feature after removing the inconsistent value



Observations from the plot

- In this graphic, we see left-skewed data that would become right-skewed if the days were flipped to the positive side

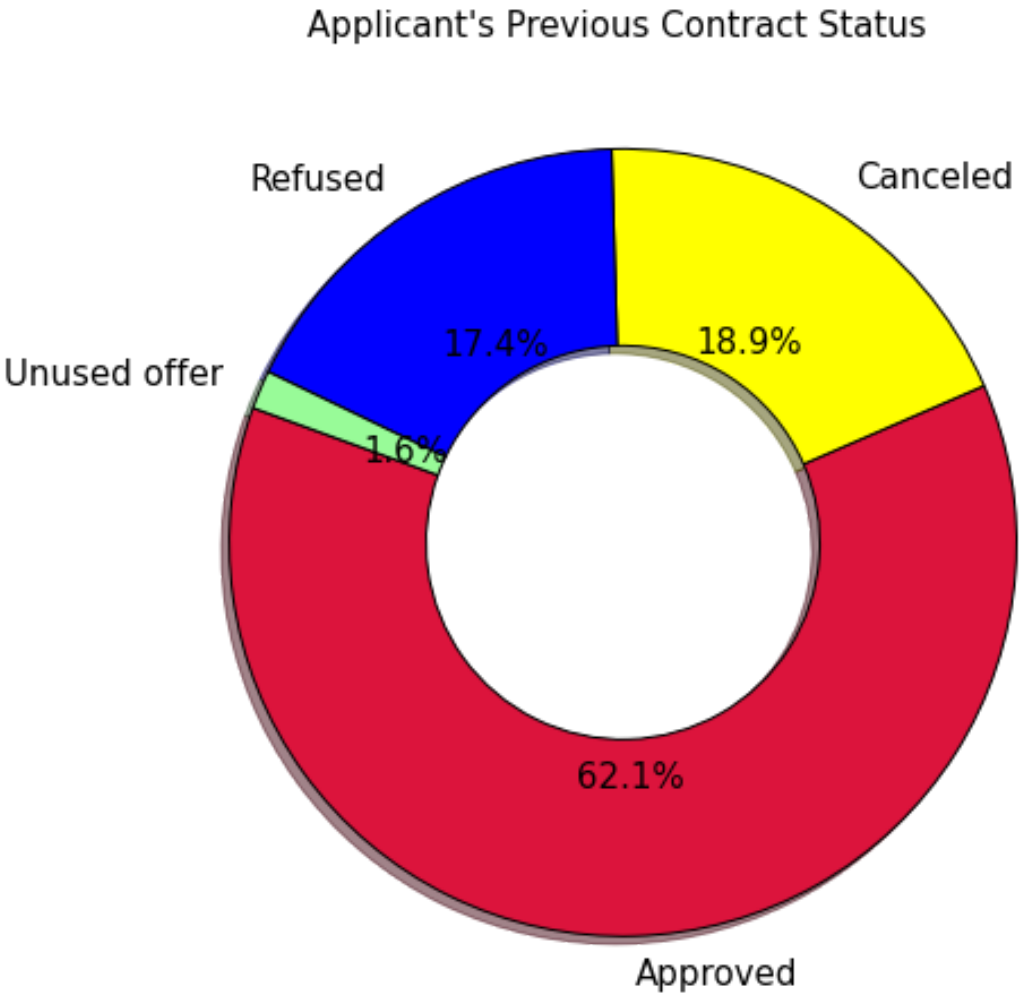
Distribution of AMT_GOODS_PRICE



Observations from the plot

- We see yet another multi-modal skewed distribution.
- Binning might make better use of this functionality.

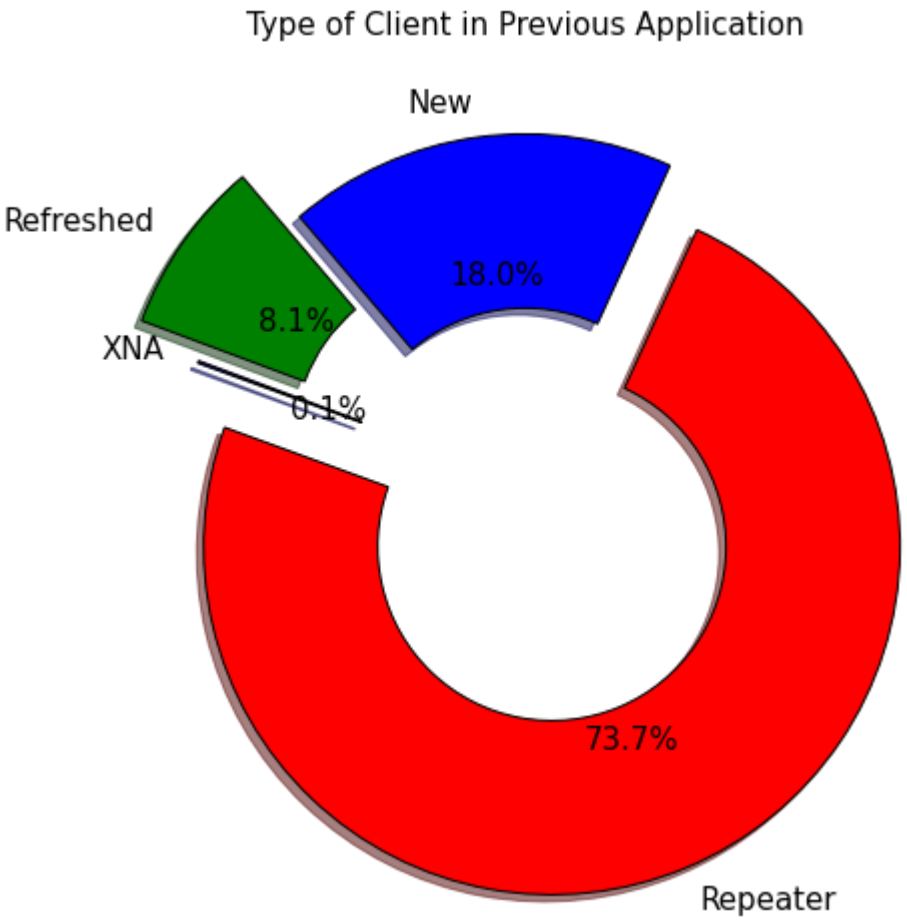
Applicant's Previous Contract Status from Prev Application DB:



Observation from above

- The majority of the applicants had their contracts approved in prior applications.
- 36% of candidates had their contracts refused or terminated, and the remaining 1.6% never used their contracts.

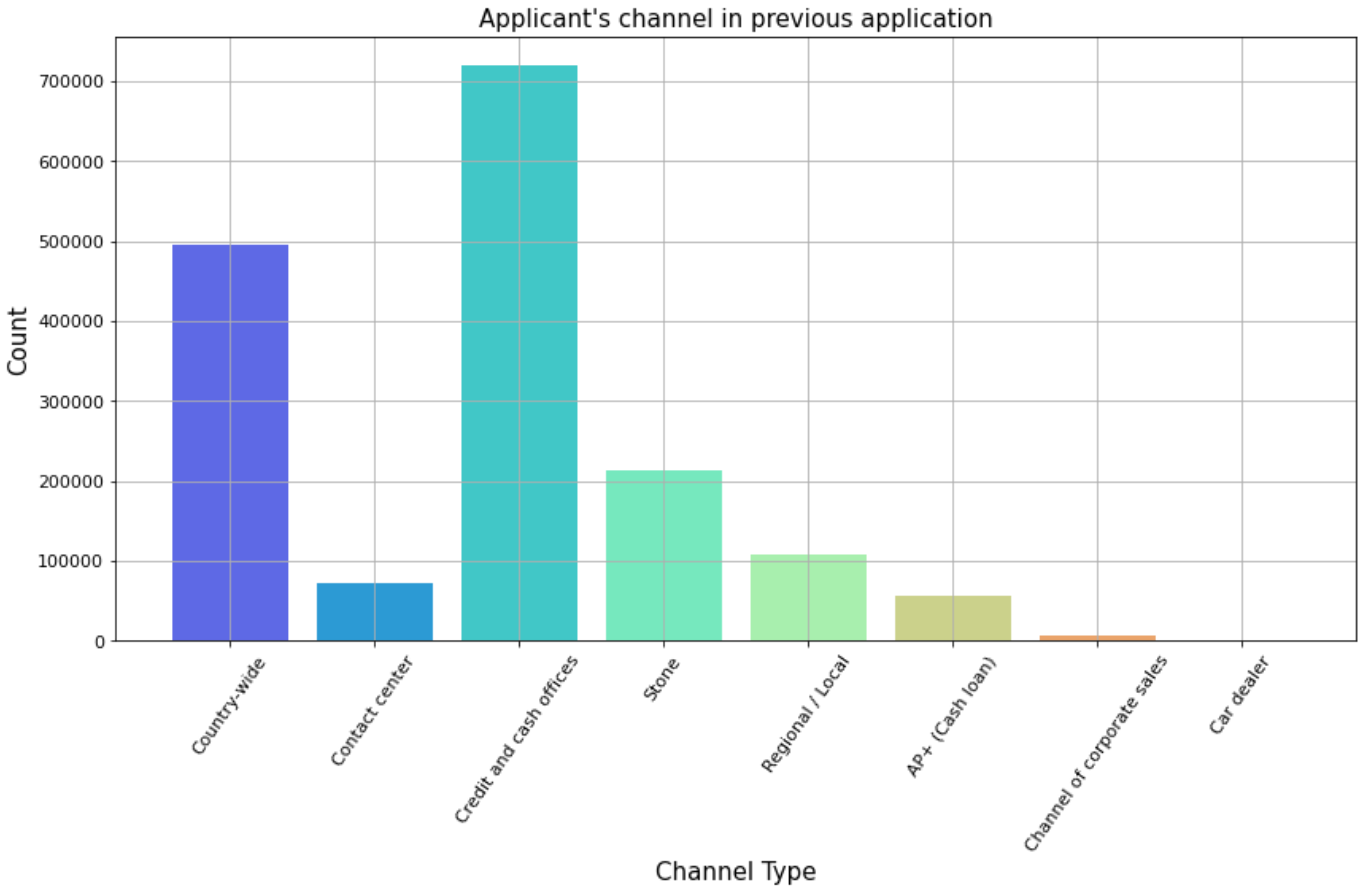
Type of Client in Previous Application



Observation from above

- The majority of applicants are returning candidates, followed by fresh and new candidates.

Applicant's channel in previous application



Observation from above

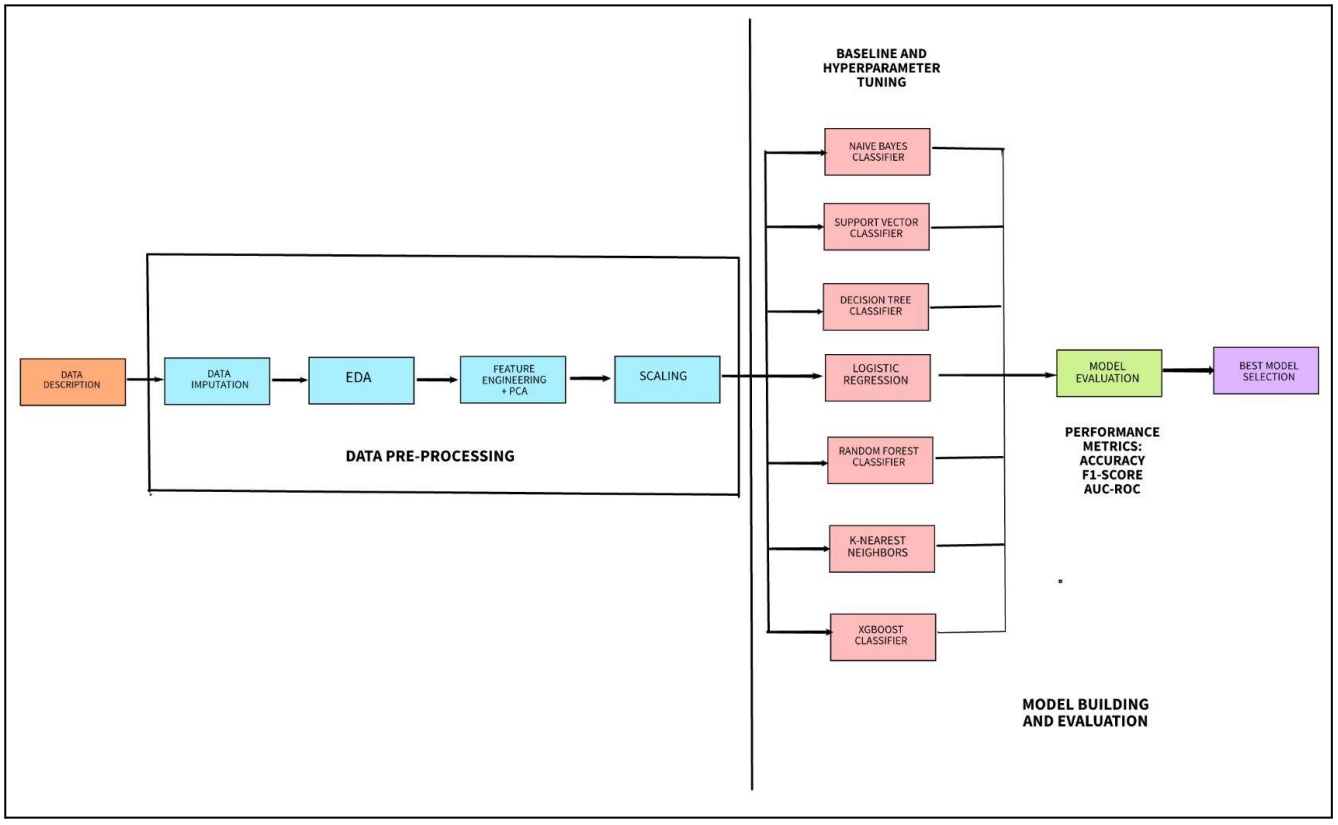
- The majority of prior applicants were referred by credit and cash offices, and vehicle dealers were the least common.

Modeling Pipelines

- We have designed two distinct pipelines, one for numerical features and the other for categorical features. We can conduct scaling on numerical features and fill in missing values by building different pipelines.
- Both of these pipelines use an imputer to fill in the missing data, with the numerical pipeline using the median and the categorical pipeline using the most frequent value.
- To enhance the efficiency of our models, the categorical pipeline uses One Hot Encoding to convert categorical variables into integers.
- The numerical pipeline uses the Standard Scaler as a scaling tool.

```
num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='median')),
        ('std_scaler', StandardScaler()),
    ])
# Identify the categorical features we wish to consider.
cat_attribs = ['FLAG_OWN_REALTY','FLAG_OWN_CAR','NAME_CONTRACT_TYPE', 'FLAG_OWN_CAR']

# Notice handle_unknown="ignore" in OHE which ignore values from the validation/test that
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('imputer', SimpleImputer(strategy='most_frequent')),
        # ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])

data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])

selected_features = numerical_attris + categorical_attris
tot_features = f"{len(selected_features)}:   Num:{len(numerical_attris)},    Cat:{len(categorical_attris)}"
tot_features
```

```
'67:   Num:63,    Cat:4'
```



Following that, the two pipelines are integrated into a single "data prep pipeline" that is utilized for modeling and experiments.

Baseline Model

To get a baseline, we will use some of the features after being preprocessed through the pipeline. The baseline model is a logistic regression model

$$\underset{\theta}{\operatorname{argmin}}\,[\mathrm{CXE}] = \underset{\theta}{\operatorname{argmin}}\left[-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right)+(1-y^{(i)})log\left(1-\hat{p}^{(i)}\right)\right]\right]$$

```
print('The Accuracy Score for Train Dataset:', np.round(accuracy_score(y_train, model.predict(X_train)), 3))
print('The F1 Score for Train Dataset:', np.round(f1_score(y_train, model.predict(X_train)), 3))
con_train = confusion_matrix(y_train, model.predict(X_train))
con_val = confusion_matrix(y_valid, model.predict(X_valid))
con_test = confusion_matrix(y_test, model.predict(X_test))
plt.figure(figsize=(7,6))
print('The Confusion Matrix for Training Set')
sns.heatmap(con_train, annot=True, fmt='g',cmap='viridis')
plt.show()
plt.figure(figsize=(7,6))
print('The Confusion Matrix for Validation Set')
sns.heatmap(con_val, annot=True, fmt='g',cmap='viridis')
plt.show()
plt.figure(figsize=(7,6))
print('The Confusion Matrix for Test Set')
sns.heatmap(con_test, annot=True, fmt='g',cmap='viridis')
plt.show()
plt.figure(figsize=(9,6))
print('The AUC-ROC for Train Set')
plot_roc_curve(model, X_train, y_train);
plt.show()
plt.figure(figsize=(9,6))
print('The AUC-ROC for Valid Set')
plot_roc_curve(model, X_valid, y_valid);
plt.show()
plt.figure(figsize=(10,8))
print('AUC-ROC for Test Set')
plot_roc_curve(model, X_test, y_test);
plt.show()
```

```
The Accuracy Score for Train Dataset: 0.919
The F1 Score for Train Dataset: 0.008
```

Feature Engineering and Transformers

1.

a) For both the test and train datasets, categorical features were subjected to one-hot encoding (OHE). In addition to making our models' testing process simpler, this will help to increase their accuracy.

[150]:
```
# one-hot encoding of features
ap_train_data = pd.get_dummies(ap_train_data)
ap_test_data = pd.get_dummies(ap_test_data)

print('Training Features shape: ', ap_train_data.shape)
print('Testing Features shape: ', ap_test_data.shape)
```

```
Training Features shape:  (307511, 243)
Testing Features shape:  (48744, 239)
```

b) Our pipeline for processing categorical characteristics also makes advantage of OHE.

```
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('imputer', SimpleImputer(strategy='most_frequent')),
        # ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])
```

2. Using the PolynomialFeatures class, we performed some feature engineering to obtain 3-degree features on the test and train datasets. This will enhance the functionality of our models and be more beneficial while modifying the hyperparameters.

[158]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.impute import SimpleImputer

# Make a new dataframe for polynomial features
poly_features_df = ap_train_data[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH', 'TARGET']]
poly_features_test_df = ap_test_data[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH']]

# imputer for handling missing values
# from sklearn.preprocessing import Imputer
imputer = SimpleImputer(strategy = 'median')

pn_target = poly_features_df['TARGET']

poly_features_df = poly_features_df.drop(columns = ['TARGET'])

# Need to impute missing values
poly_features_df = imputer.fit_transform(poly_features_df)
poly_features_test_df = imputer.transform(poly_features_test_df)

# Create the polynomial object with specified degree
pn_transformer = PolynomialFeatures(degree = 3)
```

[159]:
```
# Train the polynomial features
pn_transformer.fit(poly_features_df)

# Transform the features
poly_features_df = pn_transformer.transform(poly_features_df)
poly_features_test_df = pn_transformer.transform(poly_features_test_df)
print('Shape of polynomial features: ', poly_features_df.shape)
```

```
Shape of polynomial features:  (307511, 35)
```

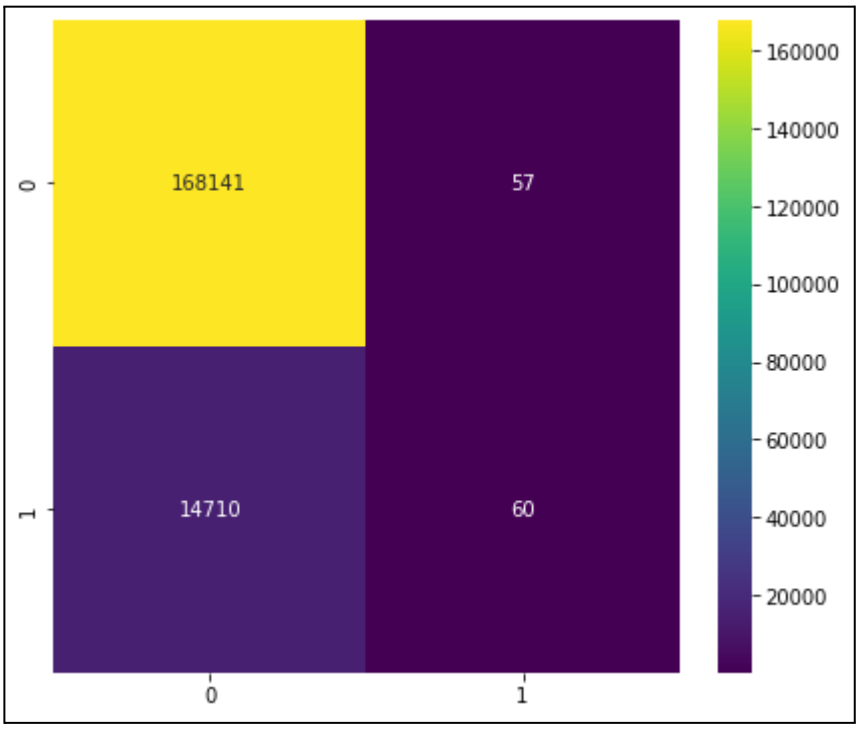
3. Categorical features from the test and train datasets have also undergone label encoding.

[149]:
```
# Label Encoding
# Create a label encoder object
le = LabelEncoder()
le_count = 0

# Iterating through the columns
for col in ap_train_data:
    if ap_train_data[col].dtype == 'object':
        # If two or fewer unique categories
        if len(list(ap_train_data[col].unique())) <= 2:
            # Train on the training data
            le.fit(ap_train_data[col])
            # Transforming both training and testing data
            ap_train_data[col] = le.transform(ap_train_data[col])
            ap_test_data[col] = le.transform(ap_test_data[col])

            # Tracking of how many columns were label encoded
            le_count += 1

print('%d columns were label encoded.' % le_count)
```

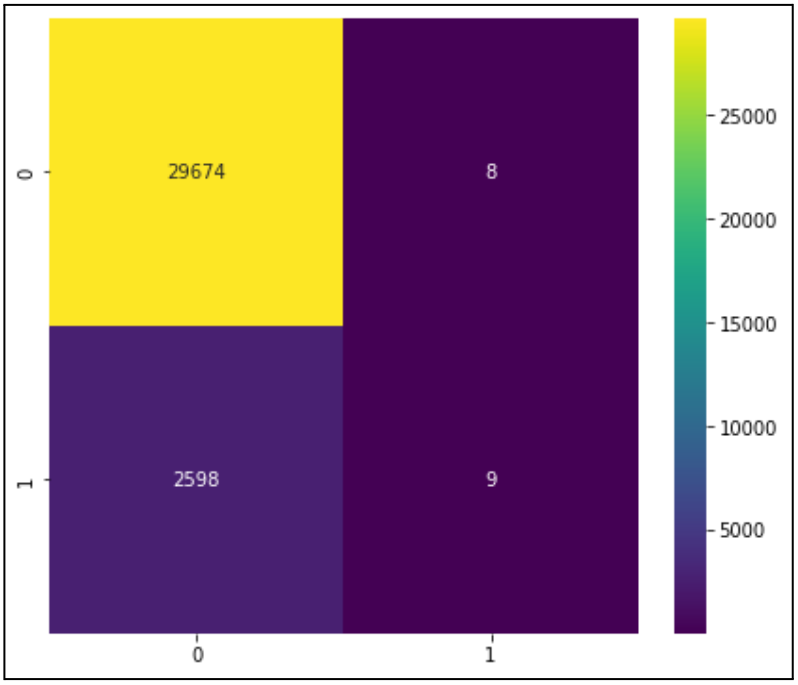
```
3 columns were label encoded.
```

Baseline Logistic Regression

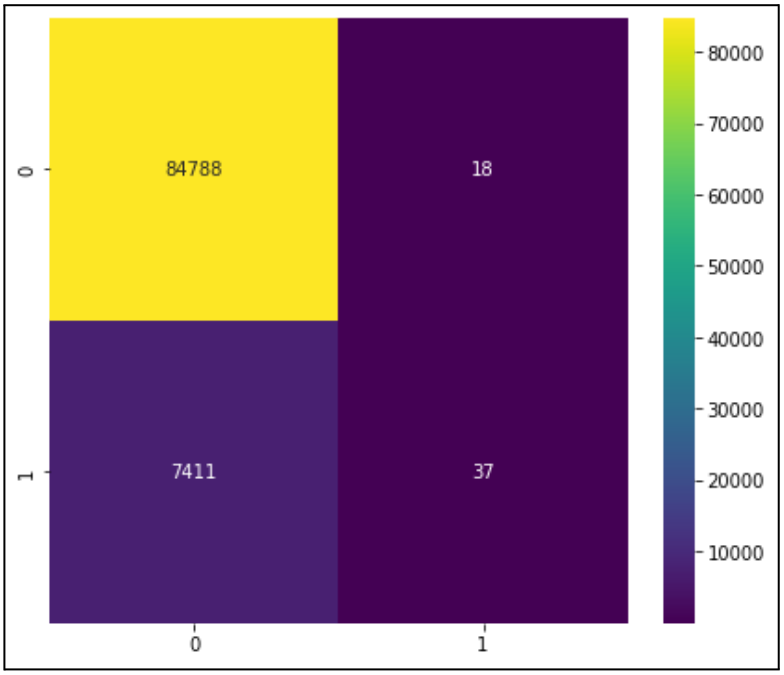
Training Set Confusion Matrix



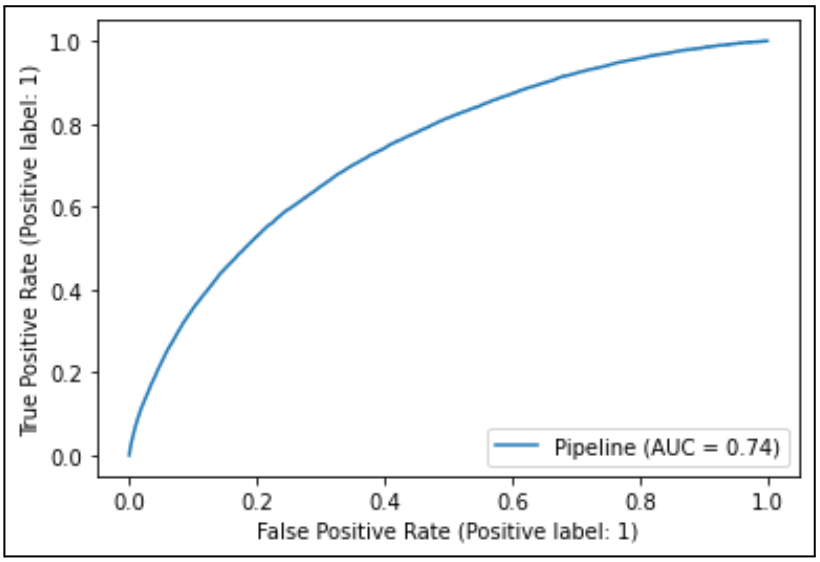
Validation Set Confusion matrix



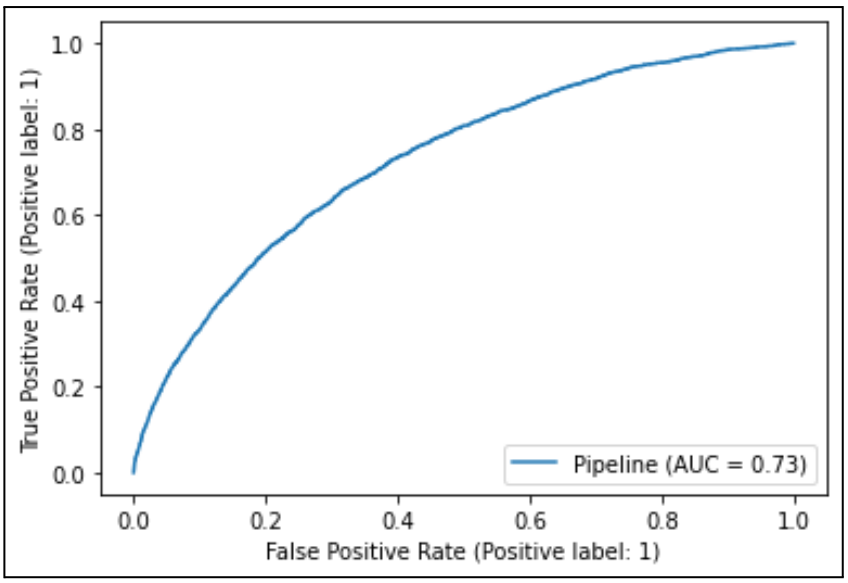
Test Set Confusion Matrix



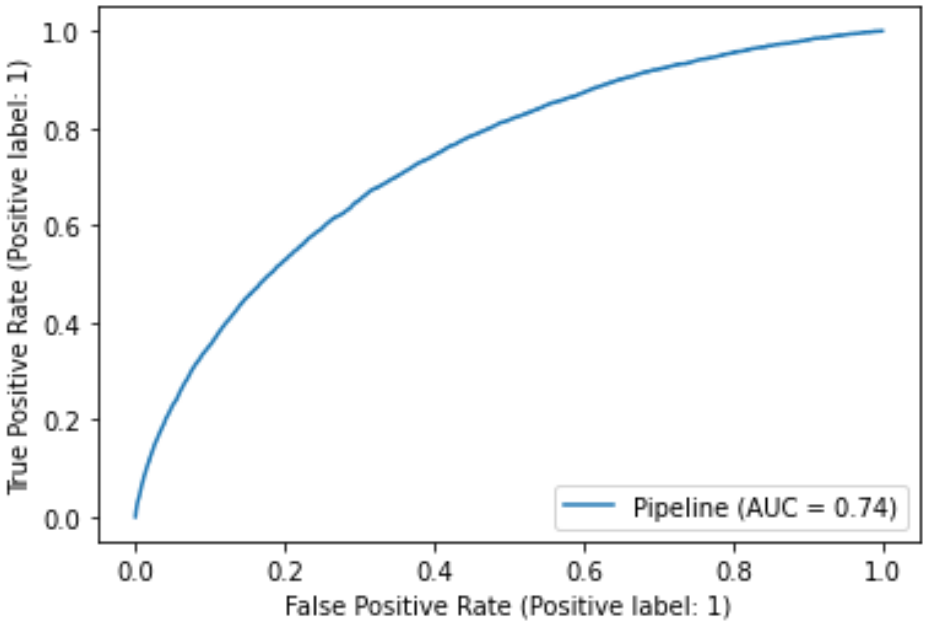
AUC-ROC for Train Set



AUC-ROC for Valid Set

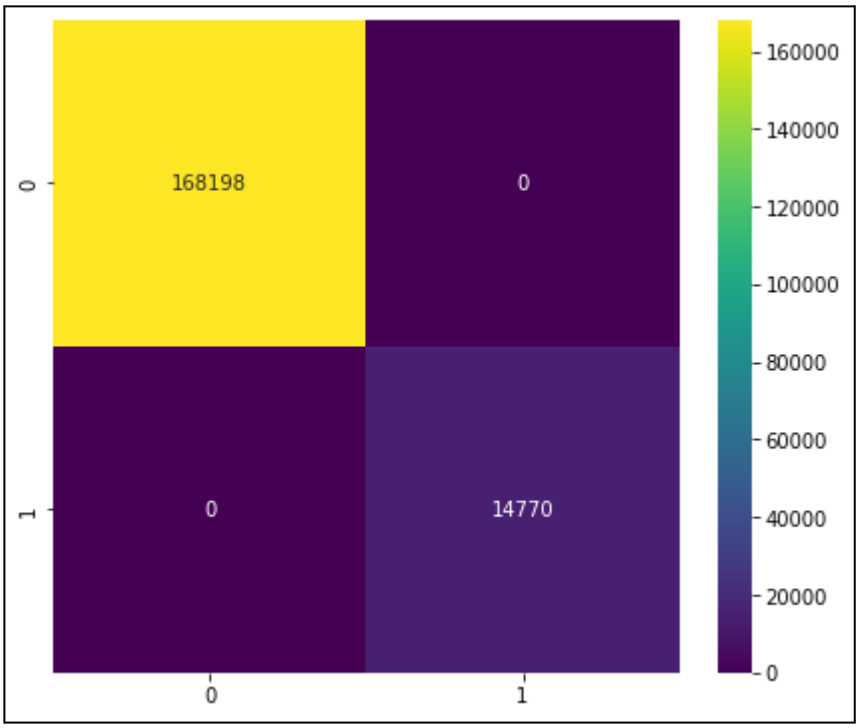


AUC-ROC for Test Set

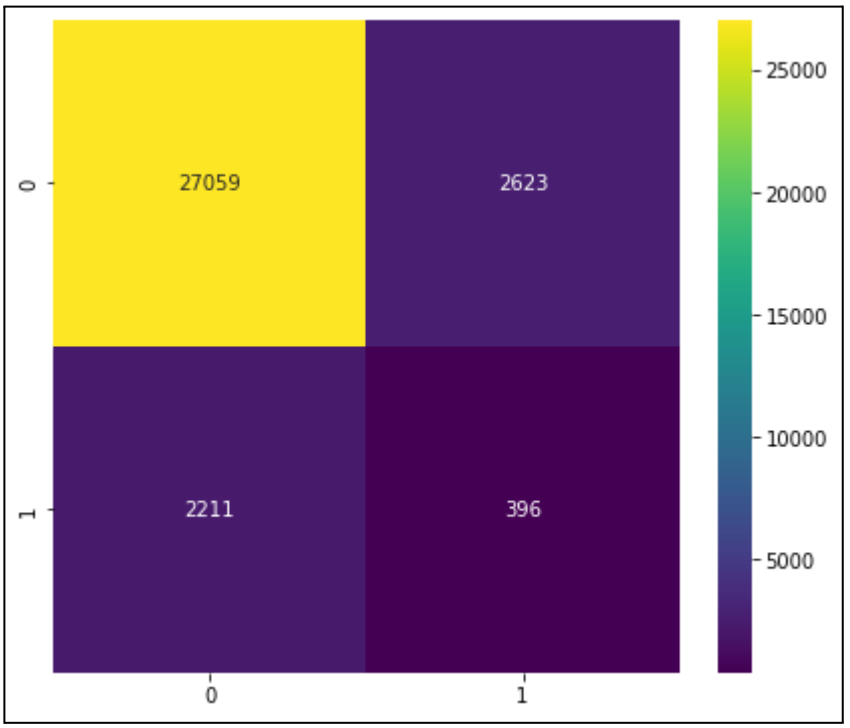


Baseline DecisionTreeClassifier

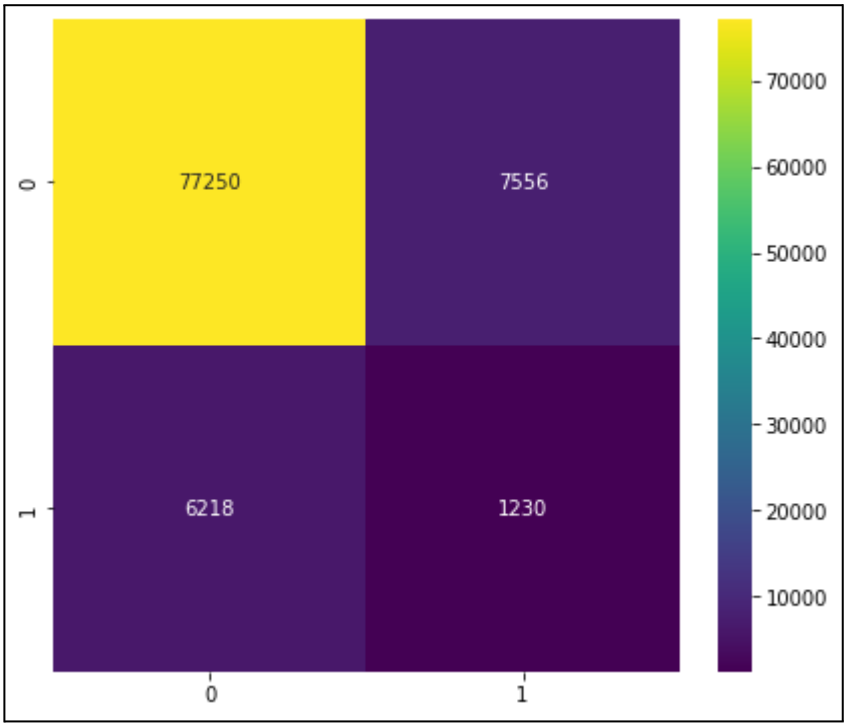
Training Set Confusion Matrix



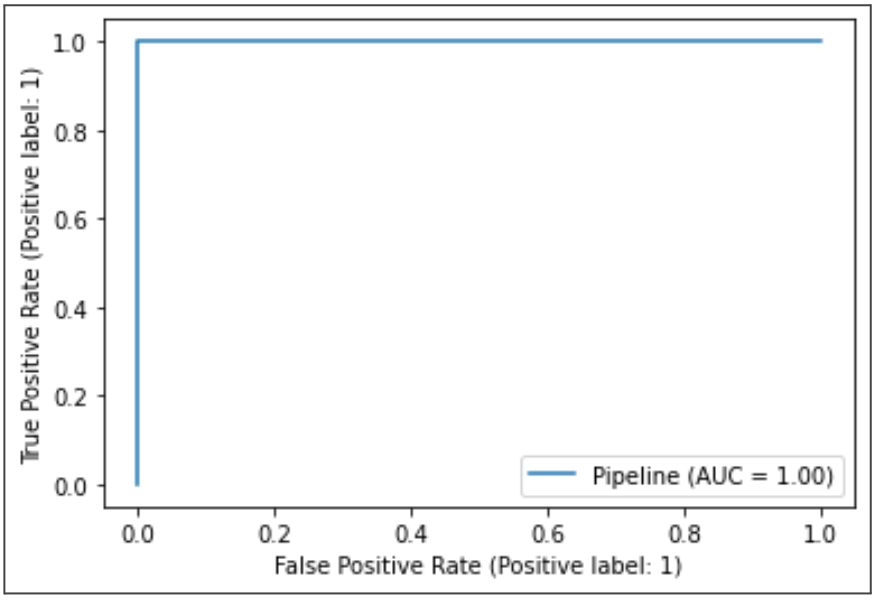
Validation Set Confusion matrix



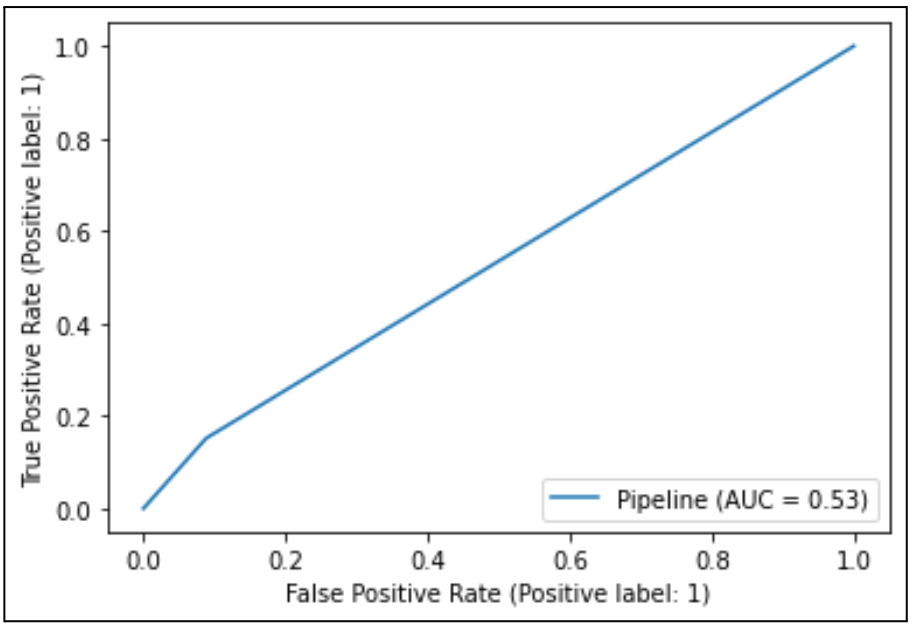
Test Set Confusion Matrix



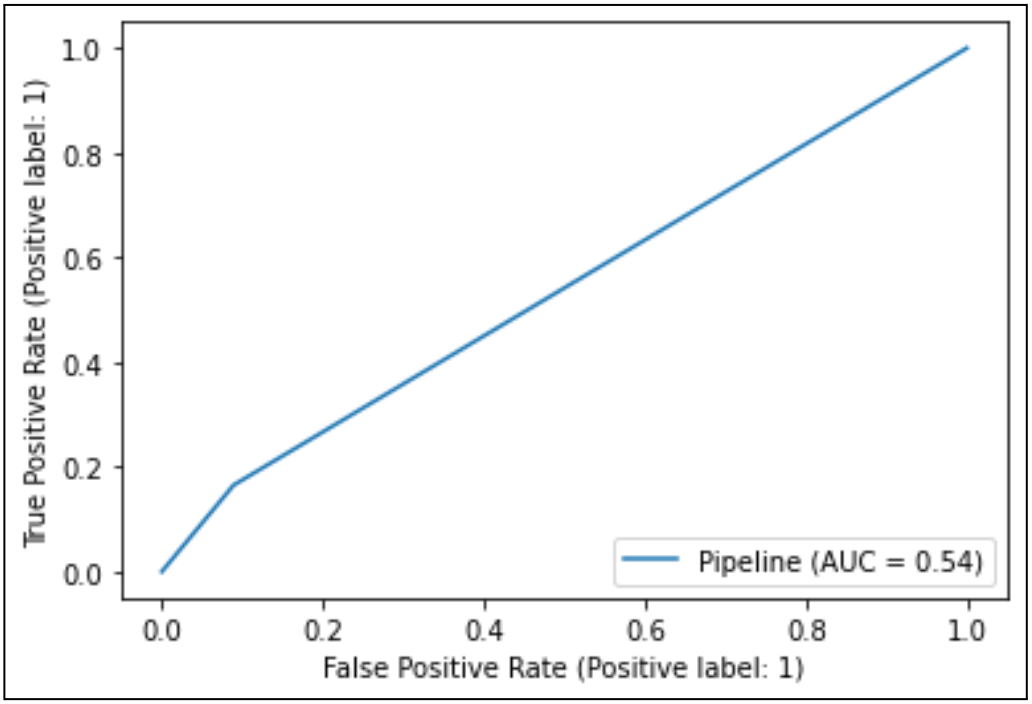
AUC-ROC for Train Set



AUC-ROC for Valid Set

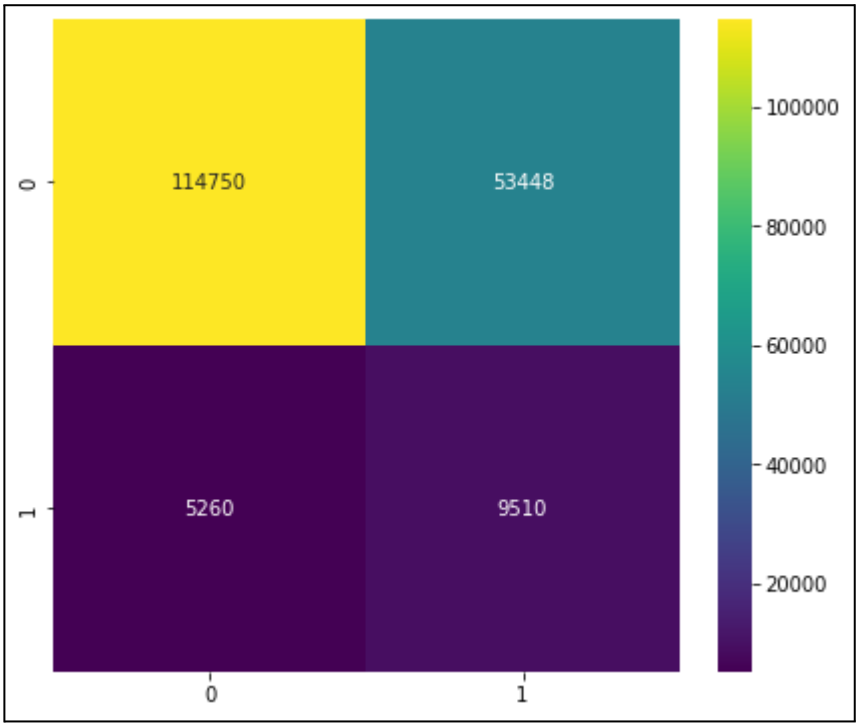


AUC-ROC for Test Set

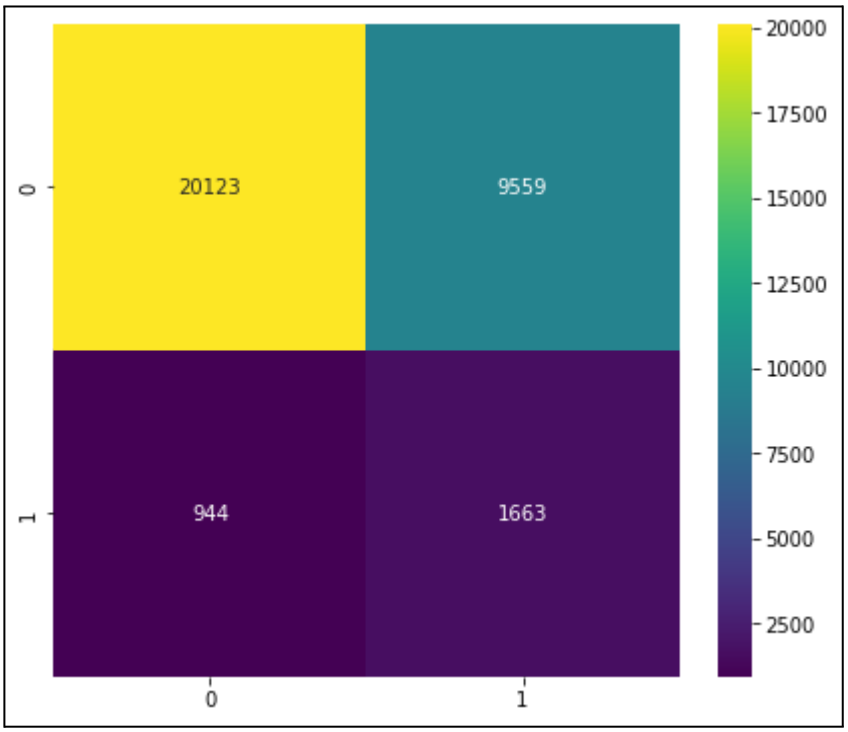


Baseline GaussianNB

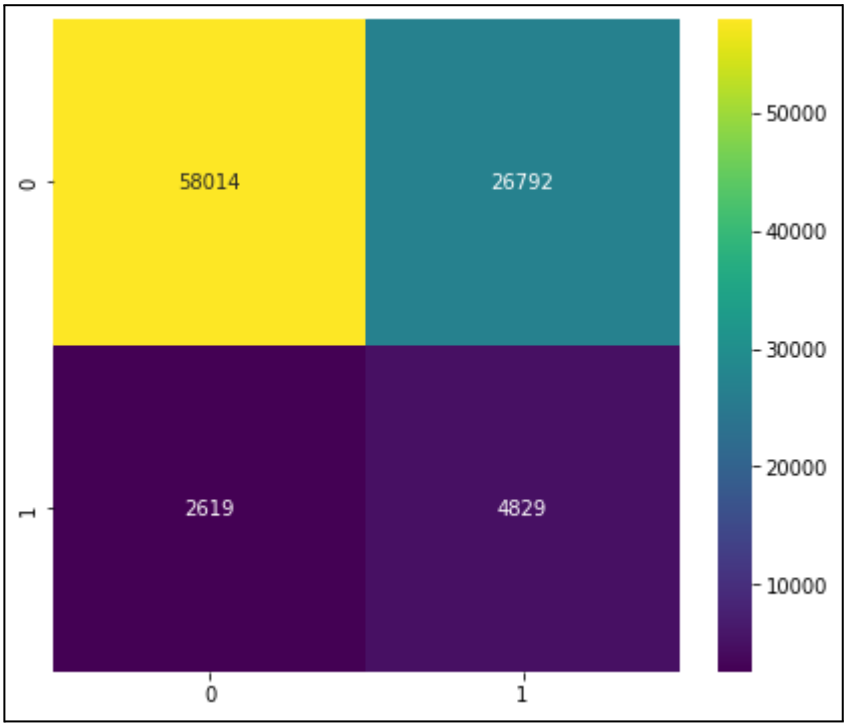
Training Set Confusion Matrix



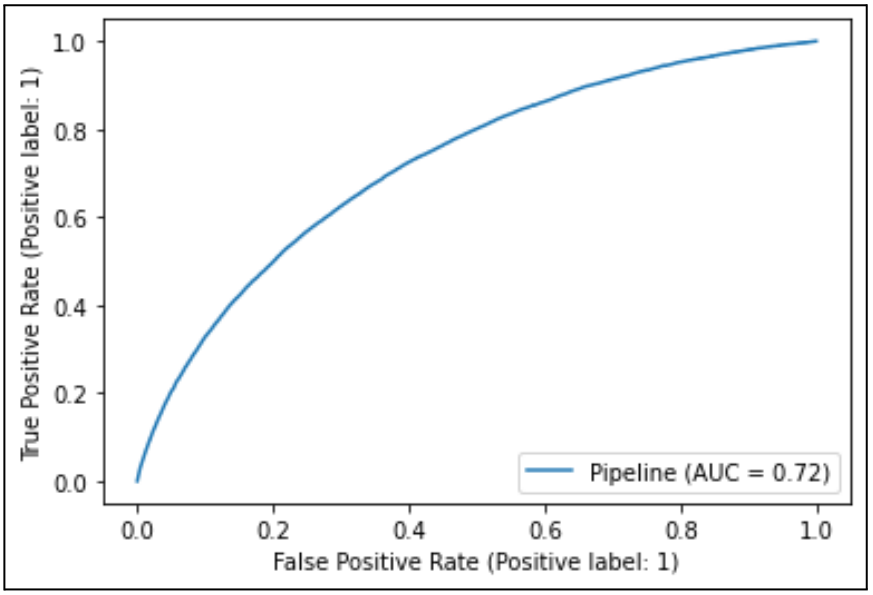
Validation Set Confusion matrix



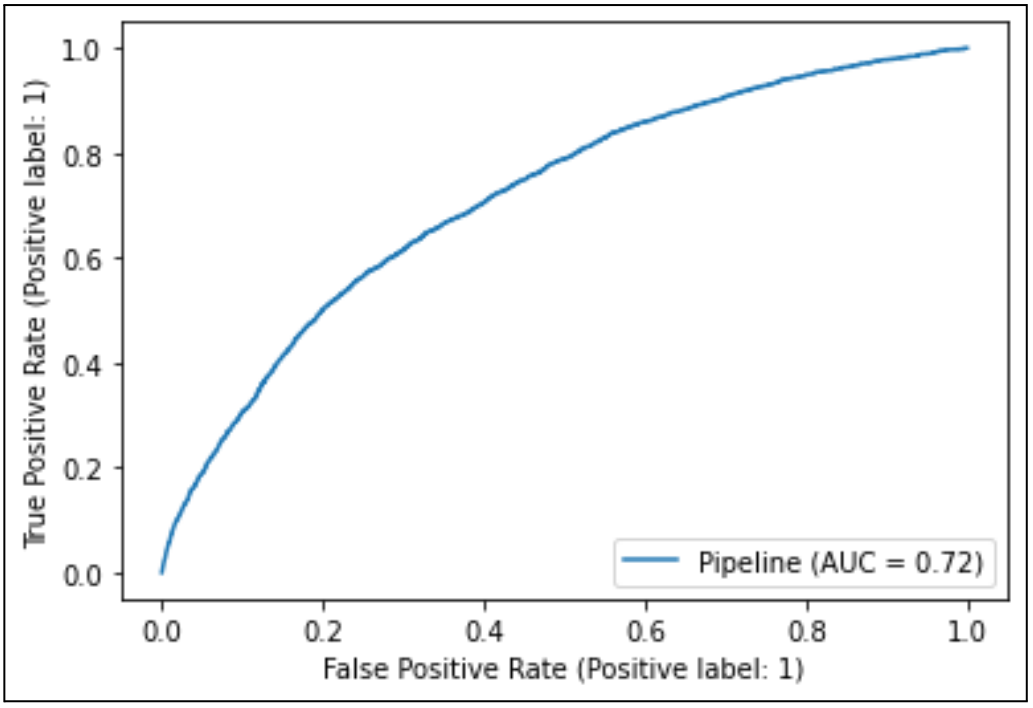
Test Set Confusion Matrix



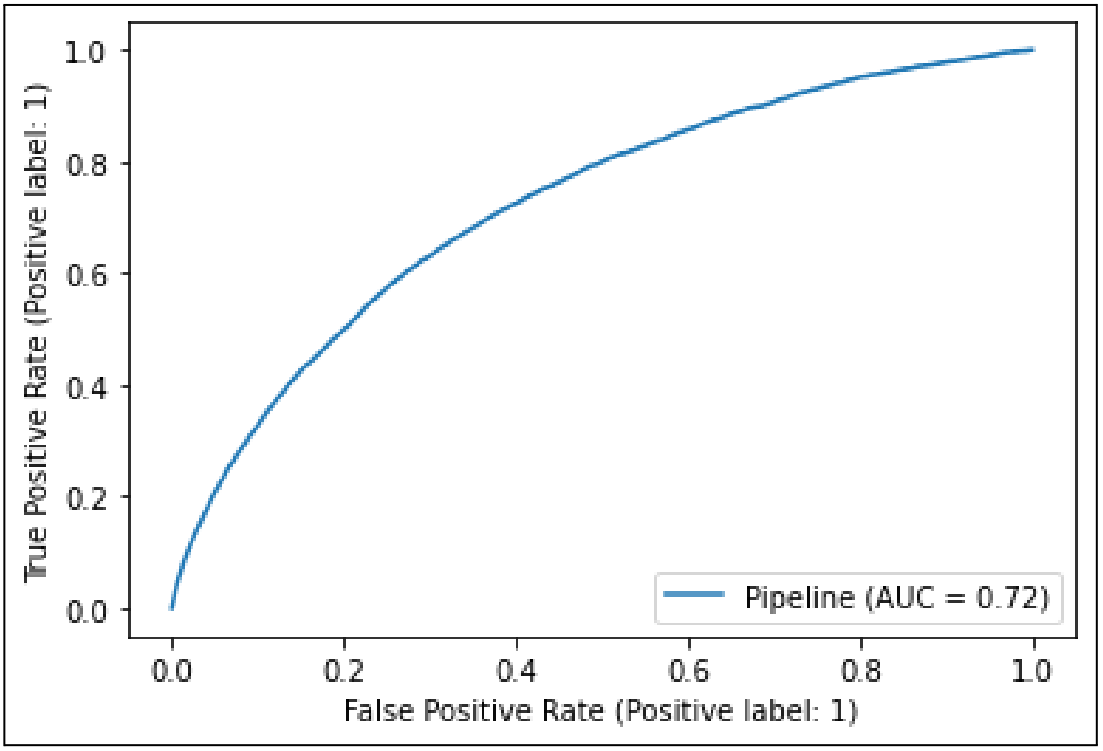
AUC-ROC for Train Set



AUC-ROC for Valid Set

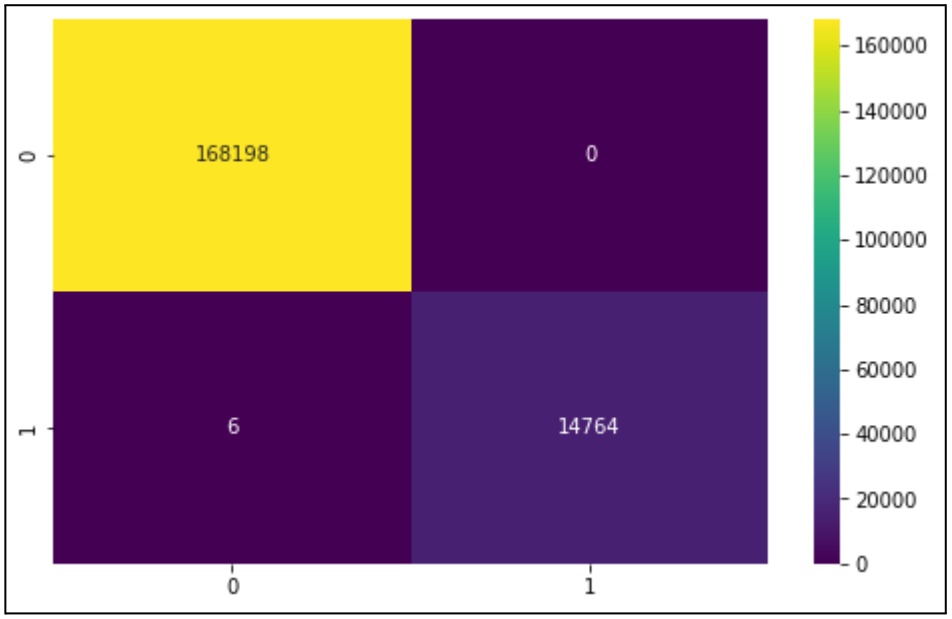


AUC-ROC for Test Set

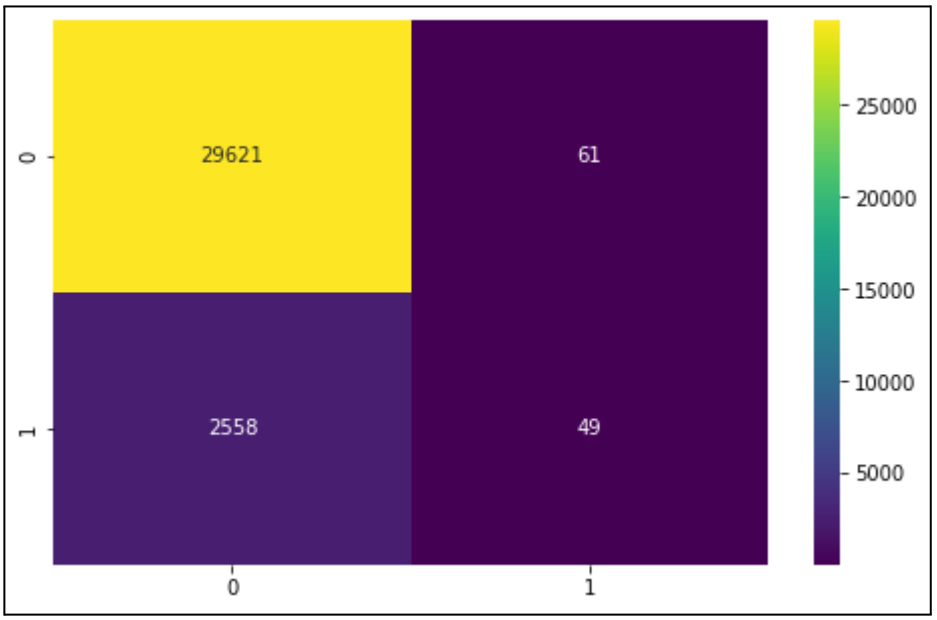


Baseline RandomForestClassifier

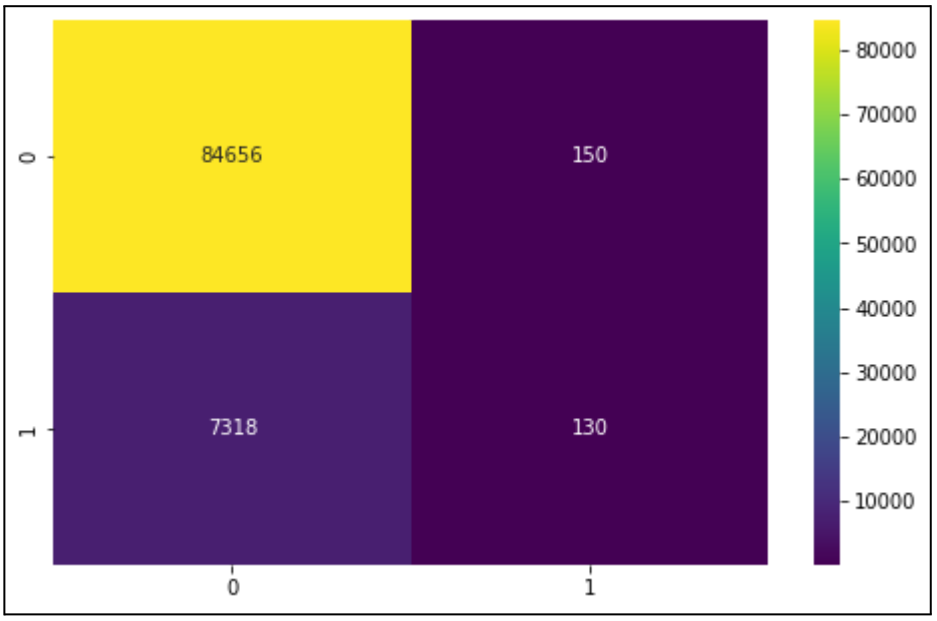
Training Set Confusion Matrix



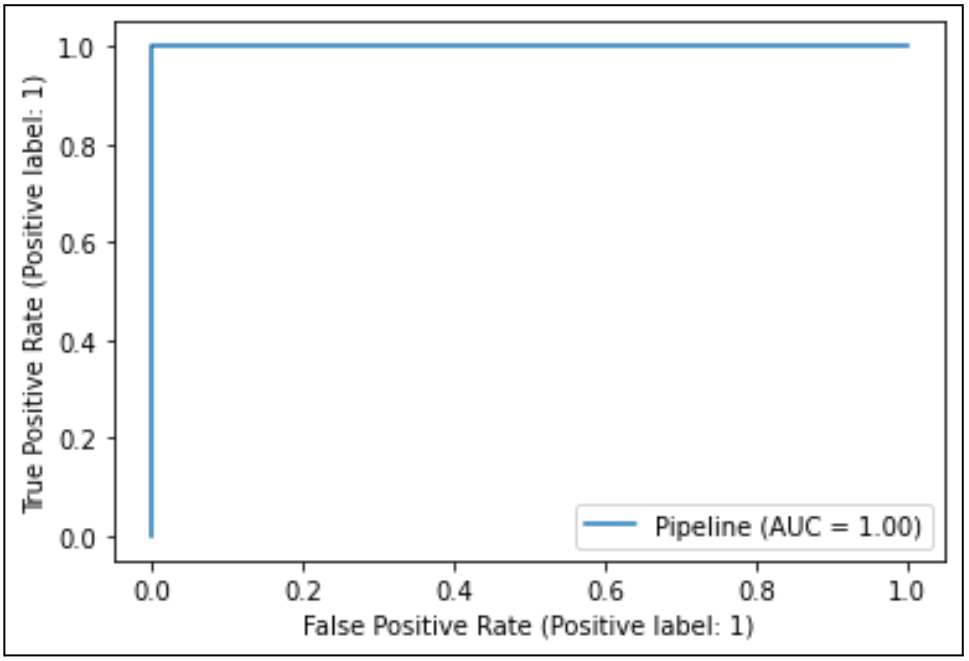
Validation Set Confusion matrix



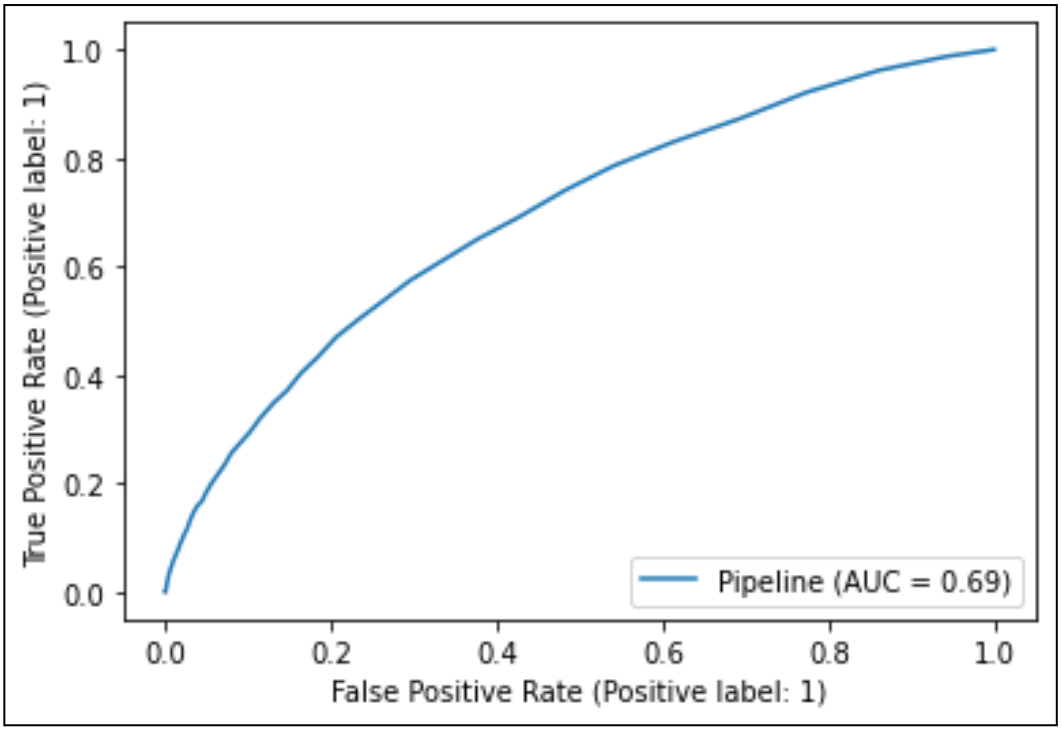
Test Set Confusion Matrix



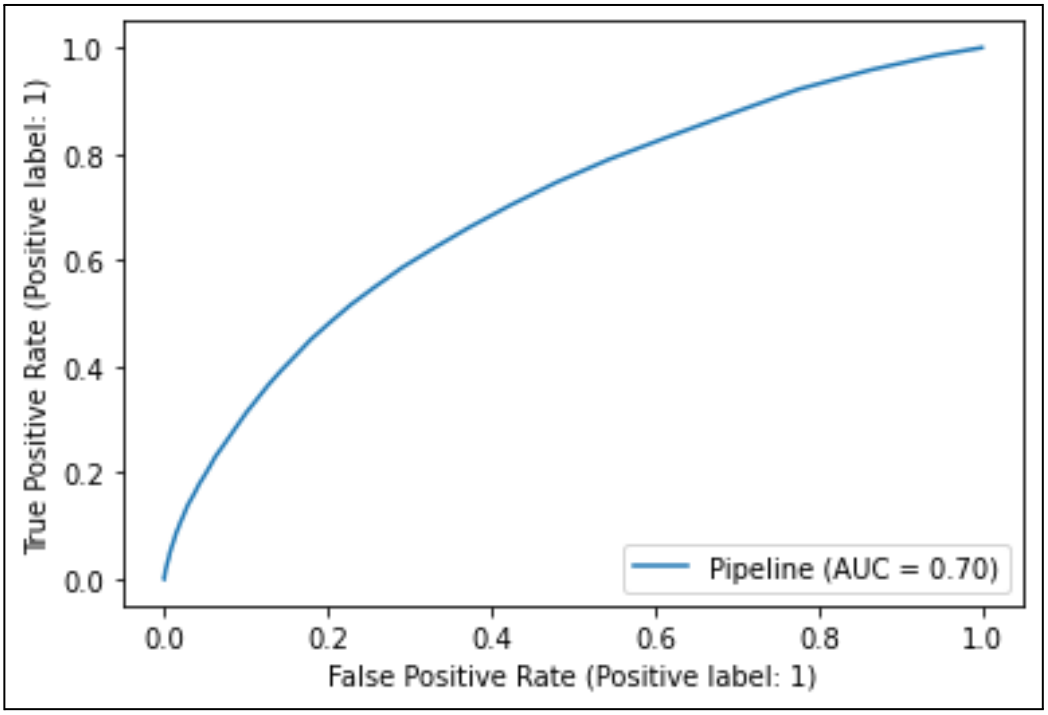
AUC-ROC for Train Set



AUC-ROC for Valid Set

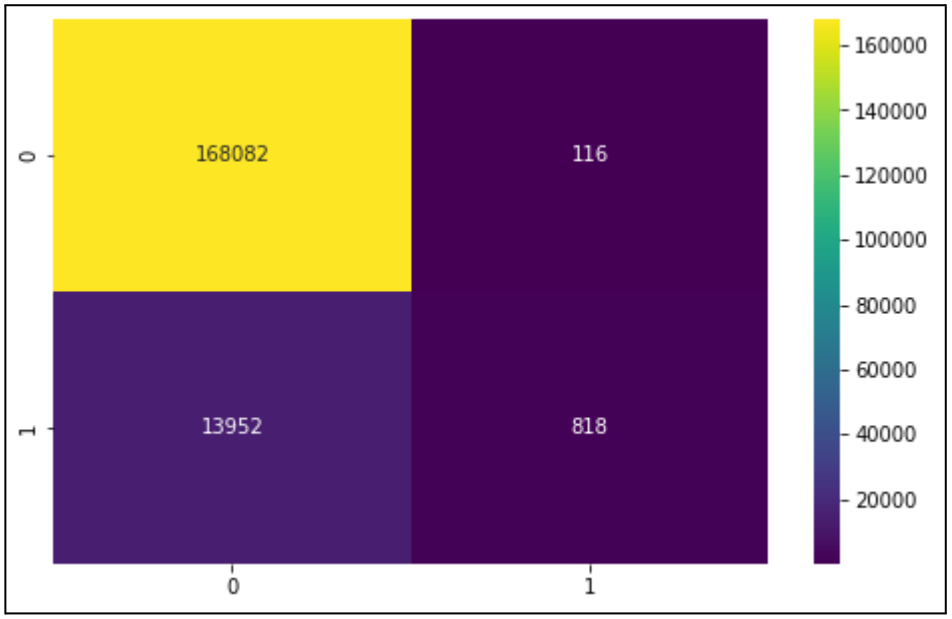


AUC-ROC for Test Set

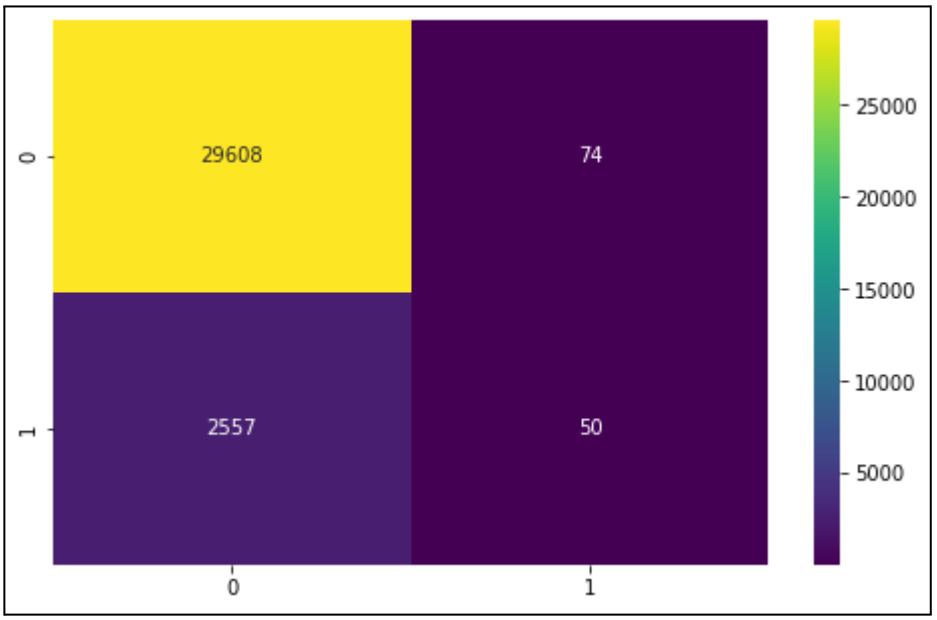


Baseline XGBClassifier

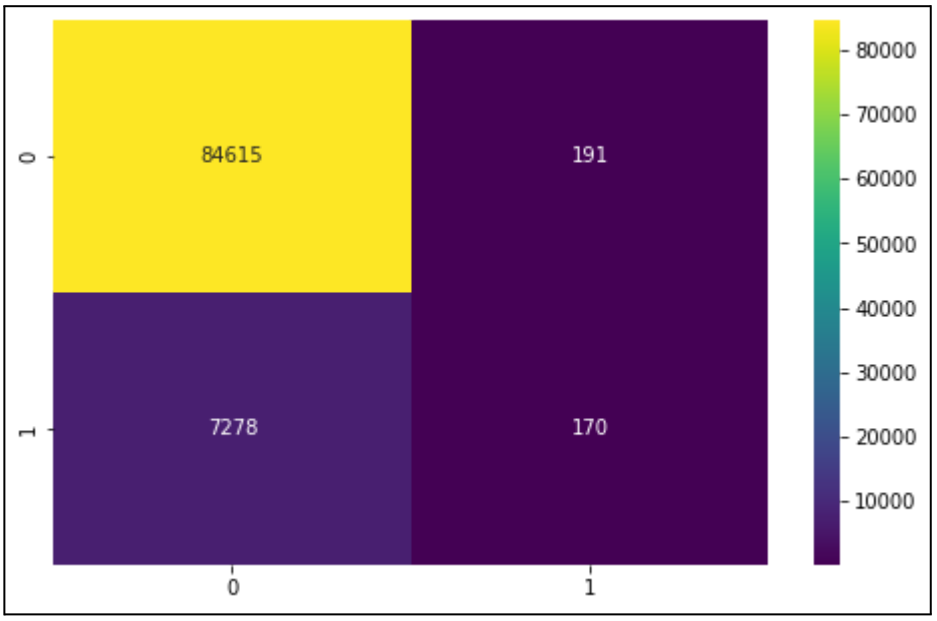
Training Set Confusion Matrix



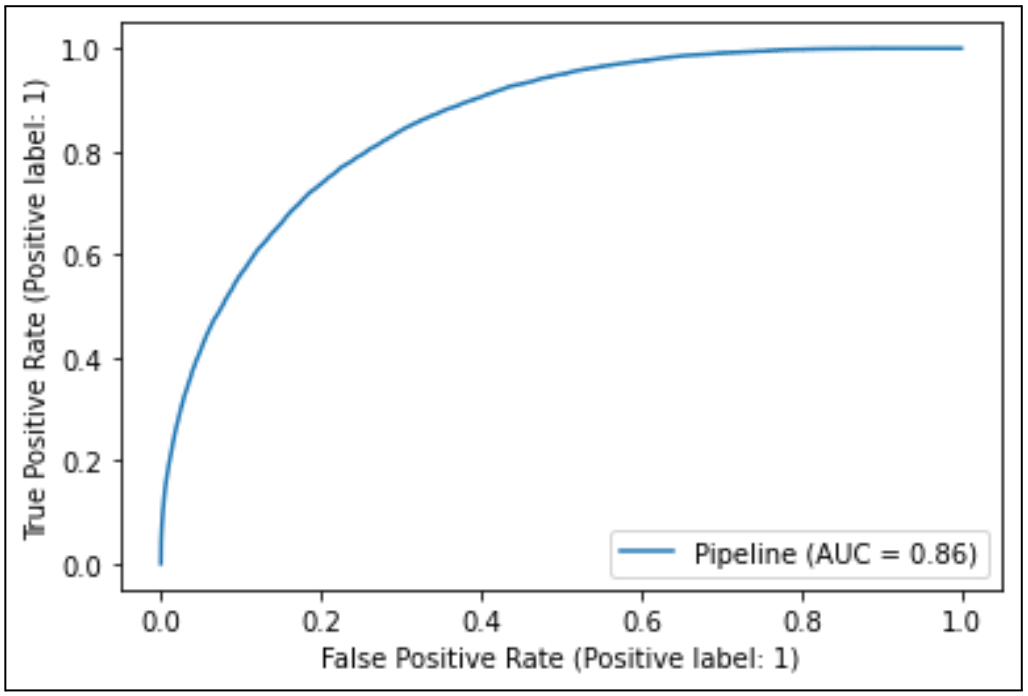
Validation Set Confusion matrix



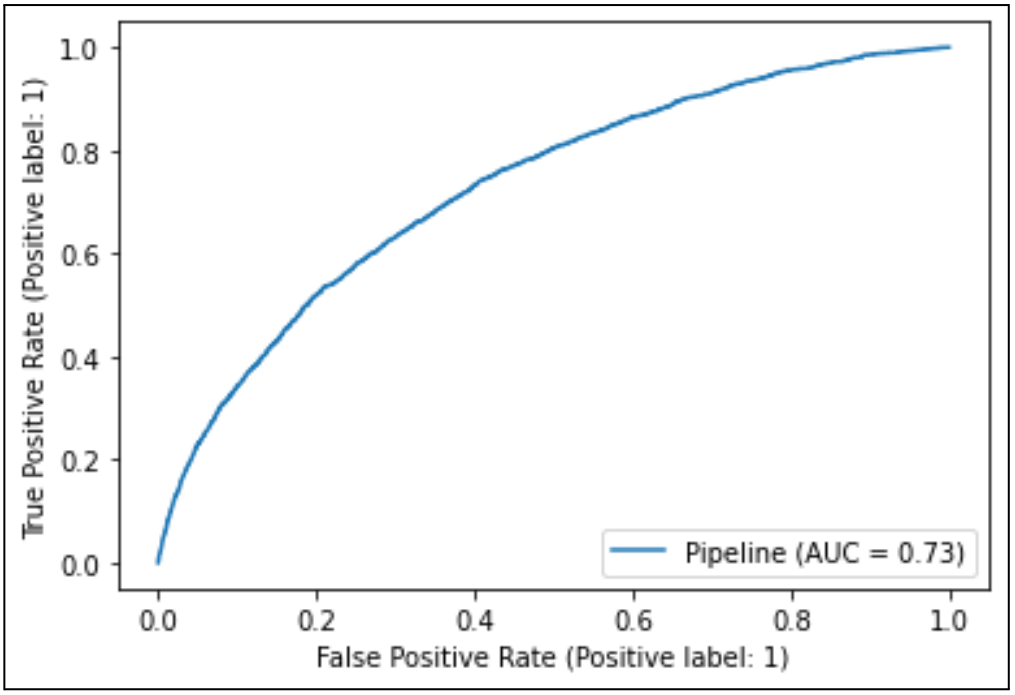
Test Set Confusion Matrix



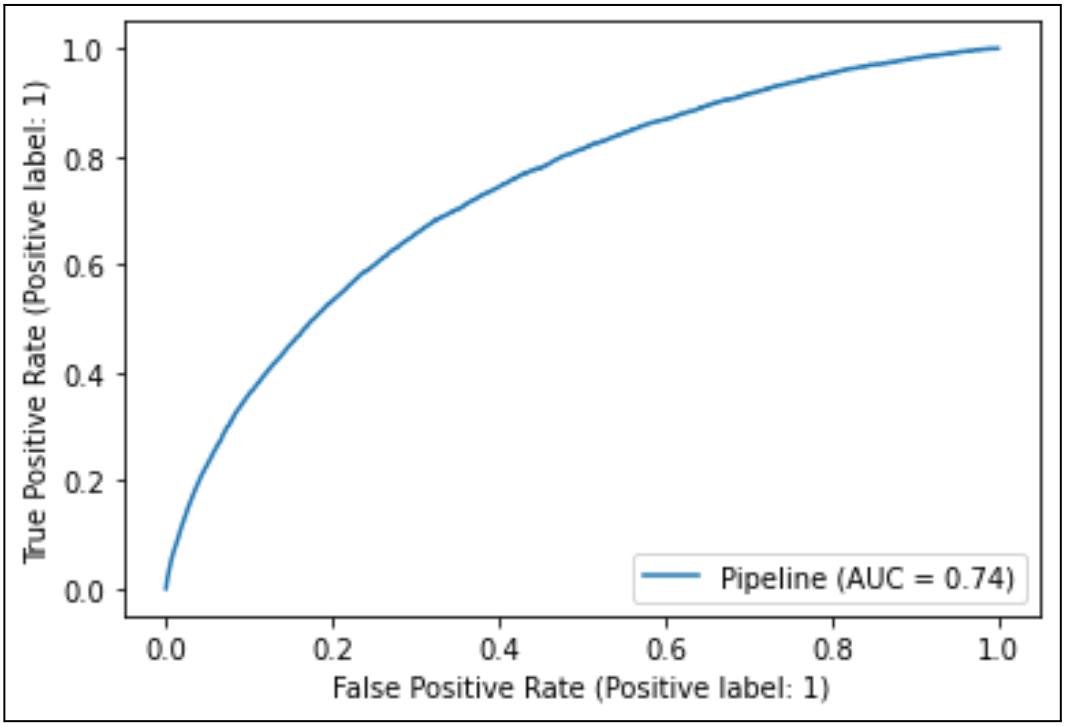
AUC-ROC for Train Set



AUC-ROC for Valid Set



AUC-ROC for Test Set



Results and discussion of results

In [252…]
```
expLog
```

Out[252]:

	exp_name	Model_name	Train Acc	Valid Acc	Test Acc	Train F1	Valid F1	Test F1	Train AUC	Valid AUC	Test AUC	Fit Time
0	Baseline_67_features	Baseline XGBClassifier	0.9193	0.9193	0.9195	0.0081	0.0069	0.0099	0.7377	0.7300	0.7385	49.3522
1	Baseline_67_features	Baseline XGBClassifier	0.9193	0.9193	0.9195	0.0081	0.0069	0.0099	0.7377	0.7300	0.7385	49.3522
2	Baseline_67_features	Baseline LogisticRegression	0.9193	0.9194	0.9195	0.0083	0.0076	0.0093	0.7378	0.7299	0.7386	5.8745
3	Baseline_67_features	Baseline LogisticRegression	0.9193	0.9193	0.9195	0.0081	0.0069	0.0099	0.7377	0.7300	0.7385	6.3473
4	Baseline_67_features	Baseline DecisionTreeClassifier	1.0000	0.8503	0.8507	1.0000	0.1408	0.1515	1.0000	0.5318	0.5380	22.4071
5	Baseline_67_features	Baseline GaussianNB	0.6791	0.6747	0.6812	0.2447	0.2405	0.2472	0.7229	0.7156	0.7225	1.8760
6	Baseline_67_features	Baseline RandomForestClassifier	1.0000	0.9189	0.9190	0.9998	0.0361	0.0336	1.0000	0.6903	0.6980	136.4937
7	Baseline_67_features	Baseline XGBClassifier	0.9231	0.9185	0.9190	0.1042	0.0366	0.0435	0.8586	0.7290	0.7381	51.3018

Our results showed that Logistic Regression and XGBoost had the highest training accuracy, achieving an accuracy of 91%. The test accuracy for Decision Tree Classifier and GaussianNB were found to be 53.8% and 72.25% respectively. After submitting our predictions to Kaggle, we obtained AUC private and public scores of 71.54% and 71.49% respectively.

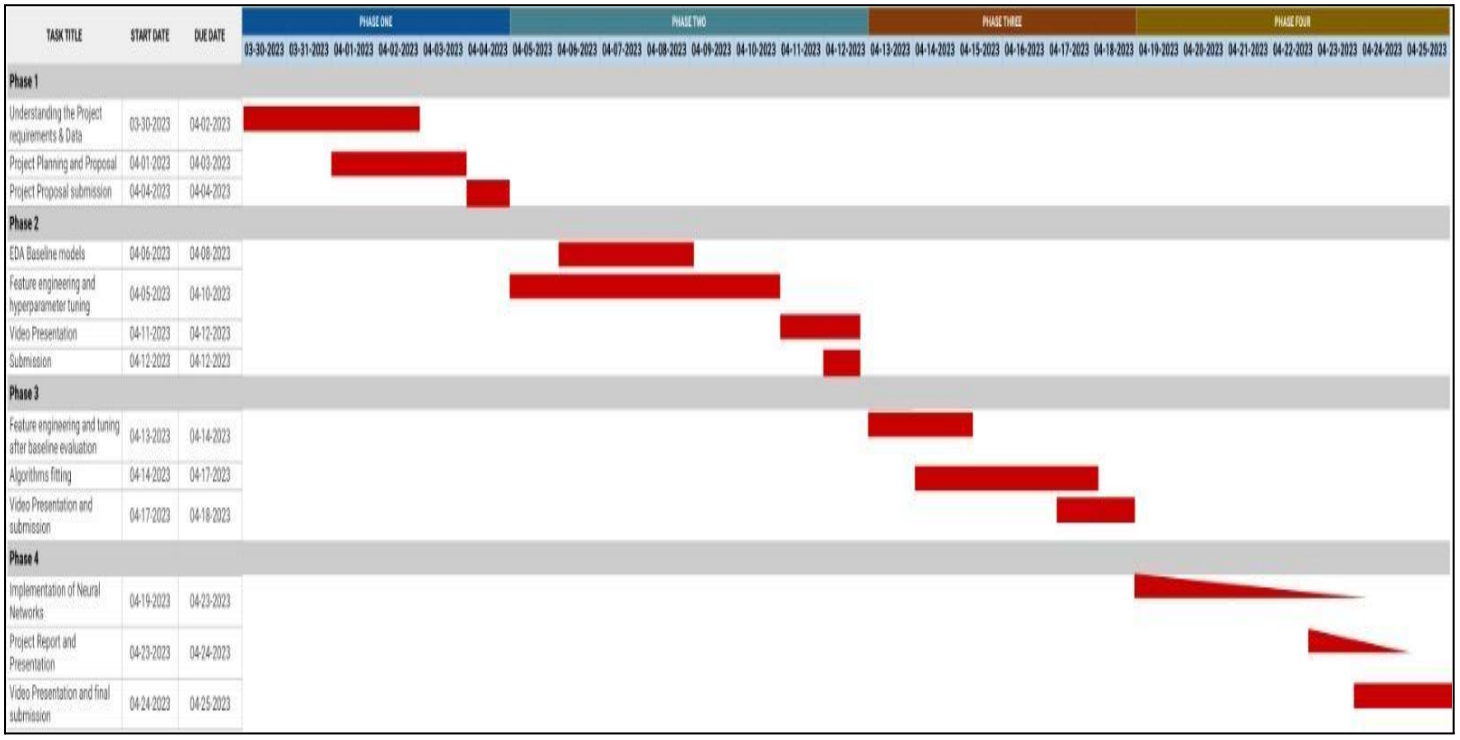
Further tests revealed that XGBoost and Logistic Regression also had the highest accuracy in the test set, both achieving an accuracy of 73.85%. These results suggest that Logistic Regression and XGBoost are promising models for predicting borrower's repayment capacity using the "Home Credit Default Risk" dataset.

And the Other models, Random ForestClassifier and The Decision Tree Classifier had low accuracy of 53.8% and 69.8% which is comparatively low, and may be improved if selected 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 XGBoost 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:



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	**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/ NN	Will Provide additional features to be added to training data and implementing neural networks.
	Kamna Chaudhary	Hyperparameter tuning/NN	Tuning model to find optimal parameters.
	Bhavya Mistry	Modeling pipelines	Visualization of modeling pipelines
	Kumud Sharma	Model evaluation and discussing results	Analysing and comparing the results
Phase 4	Kumud Sharma	Fina model evualtion	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:

The focus of our project is to develop a machine learning model that predicts a borrower's repayment capacity using the "Home Credit Default Risk" dataset. This is important because financial institutions need to assess whether a borrower has the ability to repay a loan. By conducting basic Exploratory Data Analysis on application, demographic, and credit behavior history data, and developing a baseline pipeline that handles missing, anomalous, and unusual data, we can create an accurate model that assesses risk and categorizes clients as Defaulters.

Conducted basic Exploratory Data Analysis on the "Home Credit Default Risk" dataset. Developed a baseline pipeline that handles missing, anomalous, and unusual data. Employed several baseline models, including Logistic Regression, Decision Tree Classifier, Random Forest, Naive Bayes Classifier, and XGBoost Classifier

When submitted to Kaggle, the baseline Ridge Logistic Regression model generated private scores of 71.54% and roughly 71.49% for public.Constructed and examined baseline models - Random Forest Logistic Regression.In the future, we plan to further refine our model by incorporating more advanced machine learning techniques and exploring additional parameters that can improve its accuracy. Overall, our project demonstrates the potential of machine learning to improve lending practices and reduce the risk of loan defaults.