Home Credit Default Risk

Introduction

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

Home Credit, a global lending platform, strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data, including telecom and transactional information, to predict their clients' repayment abilities. **Thus given a set of client specific features, the aim of this ML project is to ascertain whether a given customer *(if approved) would default on the home loan or not.***

Dataset

The data is spread across 7 different CSV files, which are linked with each other through common set of keys, as shown in the diagram below.

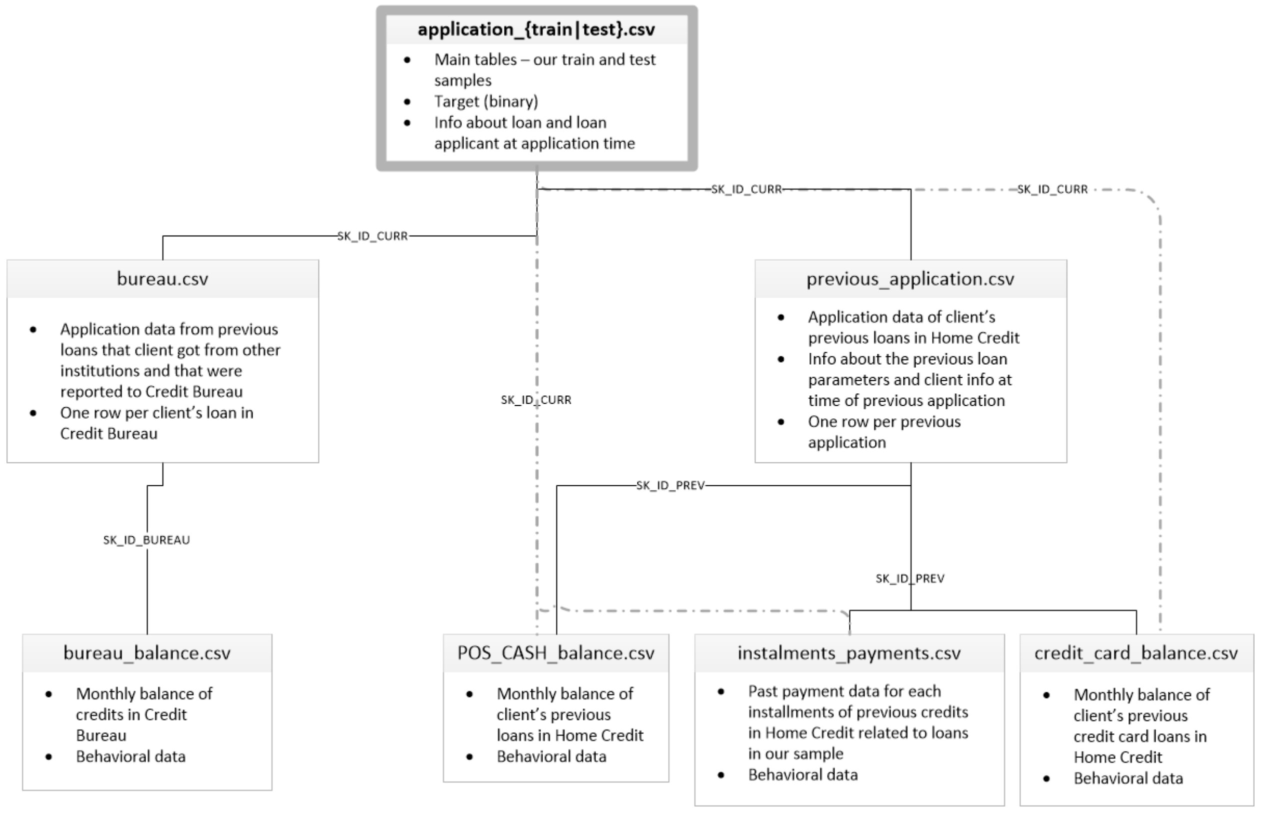

application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_PREV
SK_ID_BUREAU
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

DIAGRAM_1 : Graphic Representation of the entire dataset, with key fields connecting various tables

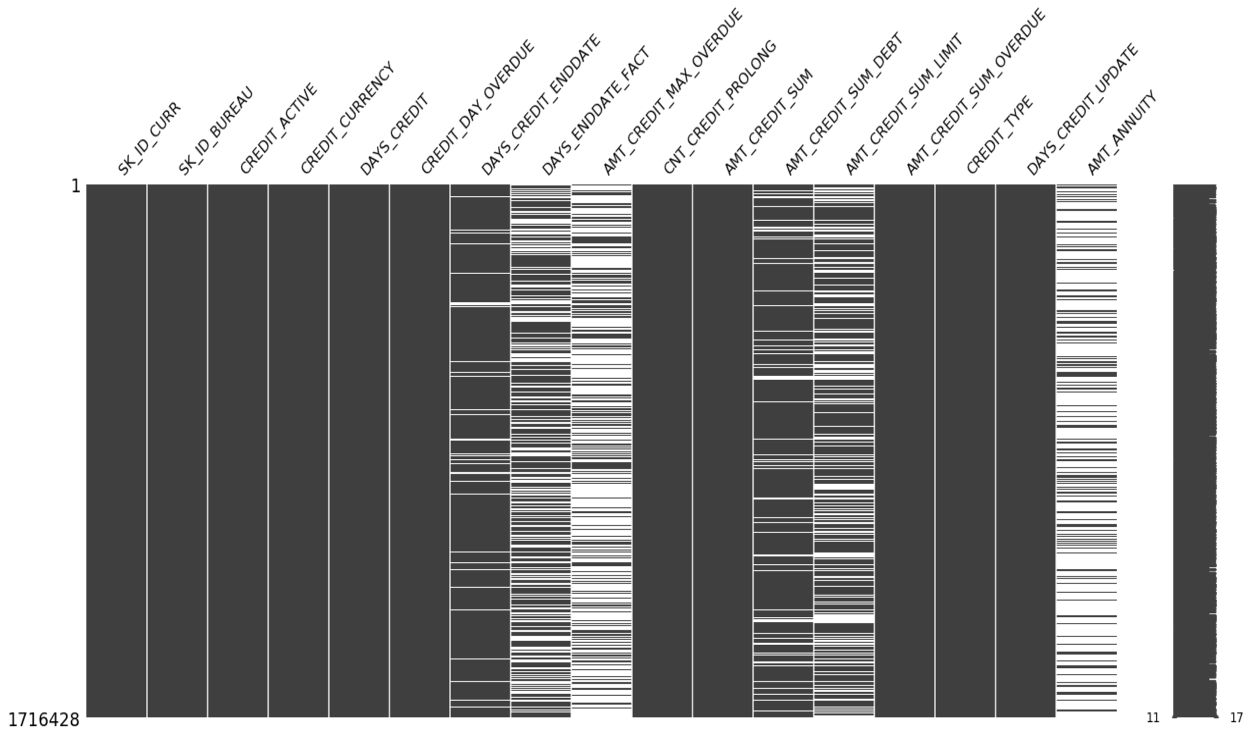
Data Wrangling,Cleaning & Preliminary EDA

Analyzing various files & Combining them with each other

1) bureau.csv: This file contains the following info about the clients

a) All clients' previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample)
b) For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.

This file originally had 17 columns with lot many missing values as shown in the diagram below.



From the above diagram, we can clearly make out that the data is not missing at random from the **bureau** dataset. All the columns which had more than 30% missing values were deleted, which decreased the no. of columns from 17 to 13 for **bureau** dataset.

2) **bureau_balance.csv:** This file contains the following info about the clients.

a) Monthly balances of previous credits in Credit Bureau.
b) This table has one row for each month of history of every previous credit reported to Credit Bureau.

This file had 3 columns with no missing values. Hence all the columns were retained.

3) **POS_CASH_balance.csv:** This file contains the following info about the clients.

a) Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.
b) This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample.

This file had 8 columns, with very few missing values, so all the 8 columns were retained.

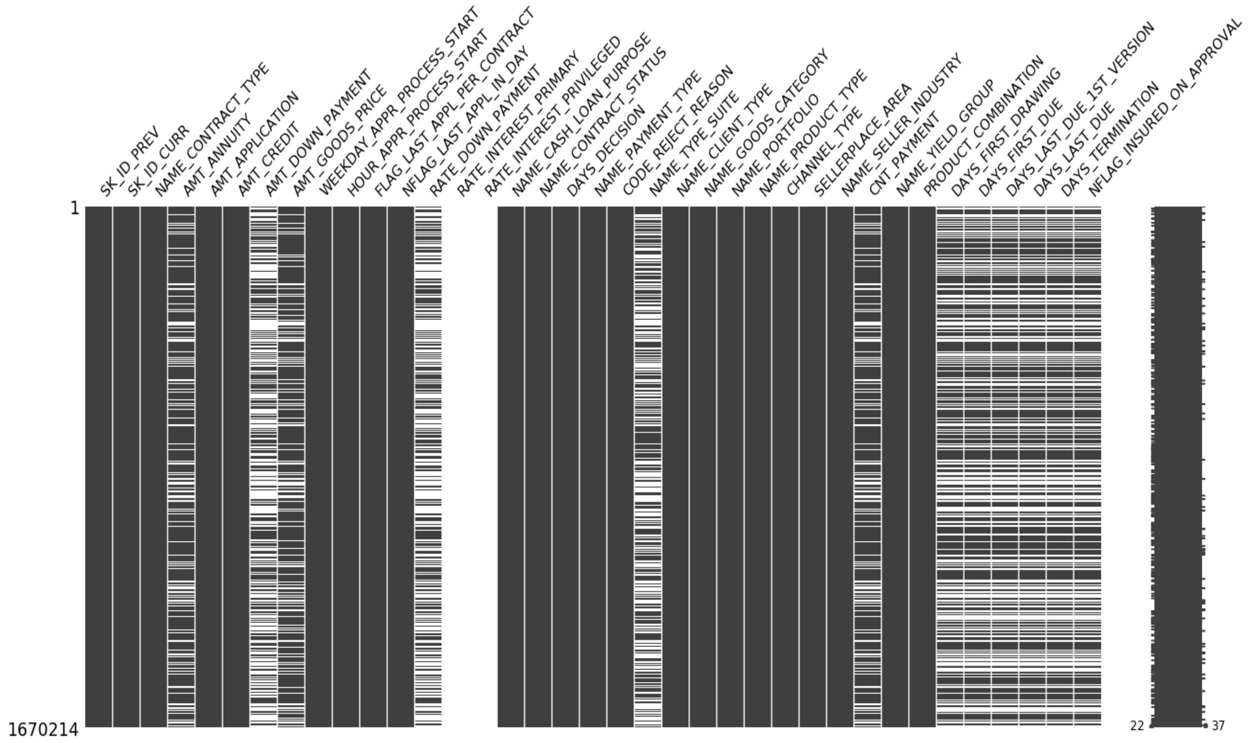
4) **credit_card_balance.csv:** This file contains the following info about the clients.

a) Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.
b) This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample.

This file had 23 columns, but since no column had more than 30% missing values, all the 23 columns were retained.

5) **previous_application.csv:** This file contains the following info about the clients.

a) All previous applications for Home Credit loans of clients who have loans in our sample.
b) There is one row for each previous application related to loans in our data sample.



From the above diagram, we can clearly make out that the data is not missing at random from **previous_application** dataset. All the columns which had more than 30% missing values were deleted, which decreased the no. of columns from 37 to 26 for **previous_application** dataset.

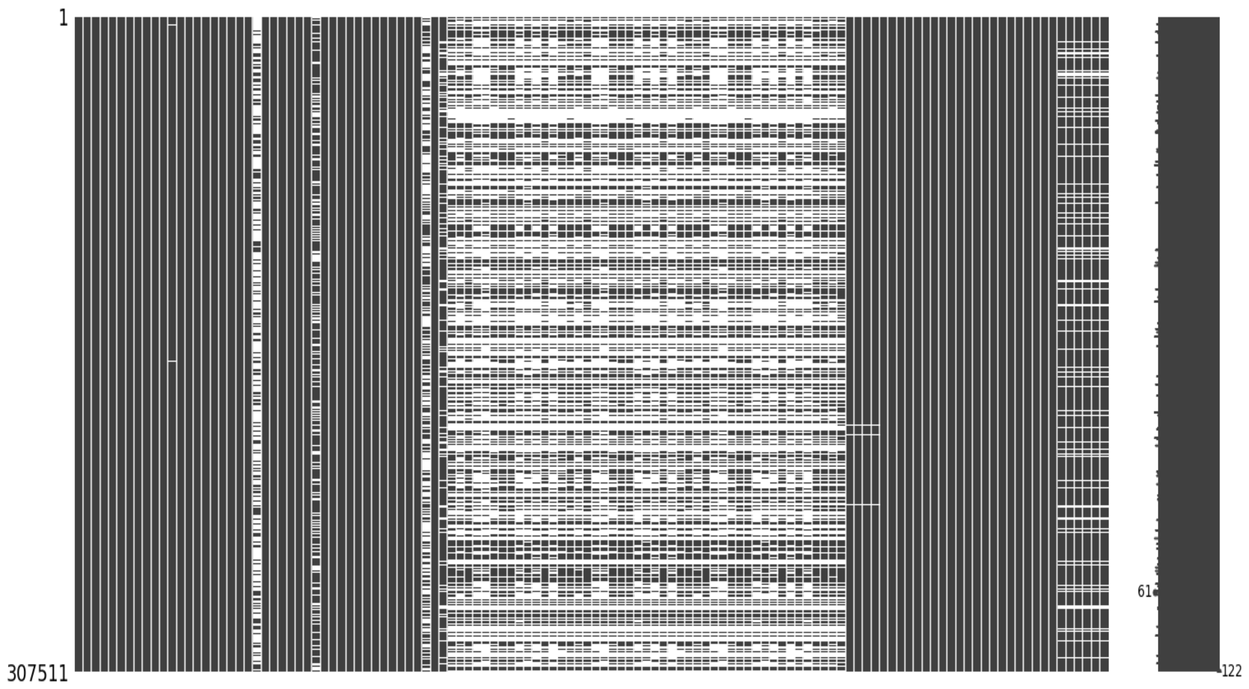
6) **installments_payments.csv:** This file contains the following info about the clients.

a) Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample.
b) There is a) one row for every payment that was made plus b) one row each for missed payment.
c) One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

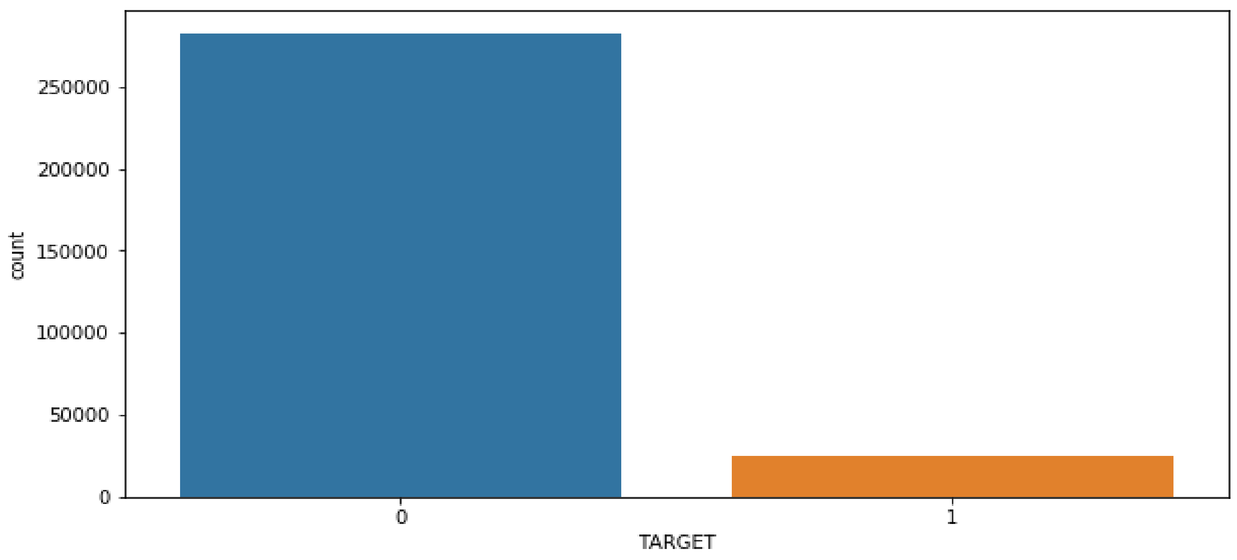
This file had 8 columns, with very few missing values, so all the 8 columns were retained.

7) **application.csv:** This file contains the following info about the clients.

a) This is the master file & contains static data for all applications. One row represents one loan in our data sample.



From the above diagram, we can clearly make out that the data is not missing at random from the main **application** dataset. All the columns which had more than 30% missing values were deleted, which decreased the no. of columns from 122 to 72 for **application** dataset.



From the above count plot of class distribution in the **application** dataset, we can see that the dataset was highly imbalanced .There were many more people who repaid home loan on time (**class 0**) than there were who didn't (**class 1**). **Hence roc_auc score, and not accuracy, would be the desired metric of choice for this entire dataset.**

<u>Merging Various Dataframes</u>

As can ascertained from the previous discussions above, the whole dataset was very dirty and needed to be cleaned before any ML models could be run on them and any predictions made. Thus the first step towards attaining that goal was to combine all the 7 dataframes with each other on the common keys, as shown in the diagram 1 above (e.g the tables bureau & bureau_balance were combined on the common key SK_ID_BUREAU). In order to achieve the above objective, less frequent categories of various categorical columns were dropped, numerical columns were averaged, the column names were altered to reflect the original dataframe to which the column belonged to in the merged dataframes etc. Exact merging details can be found in the files 8_Merging_DataFrames.ipynb, 9_Merging_DataFrames.ipynb and 10_Fully_merged_data_original.ipynb respectively.

Feature Space Cleaning & EDA Contd.

After merging various auxiliary datasets with the main application dataset, the dataset was split into the training & test set. The final merged training & test sets were still very dirty, so the following steps were taken to clean them up, followed by EDA.

a) Both training & test sets still had columns with >30% missing values (corresponding to key values of column SK_ID_CURR, which were present in application table, but were absent in auxiliary tables). All of these columns were eventually dropped from both datasets.
b) The column names were standardized to have all uppercase letters, with '_' in between words.
c) The units of all the days' columns were converted from the days to years with the column names appropriately changed.
d) The entries corresponding to the recently created year columns were all –ve, which were converted to corresponding +ve values.
e) There were some nonsense –ve entries in certain columns such as YEARS_EMPLOYED & BU_YEARS_CREDIT_ENDDATE as shown in Fig_1 & Fig_2 below, which were substituted with NaN values.

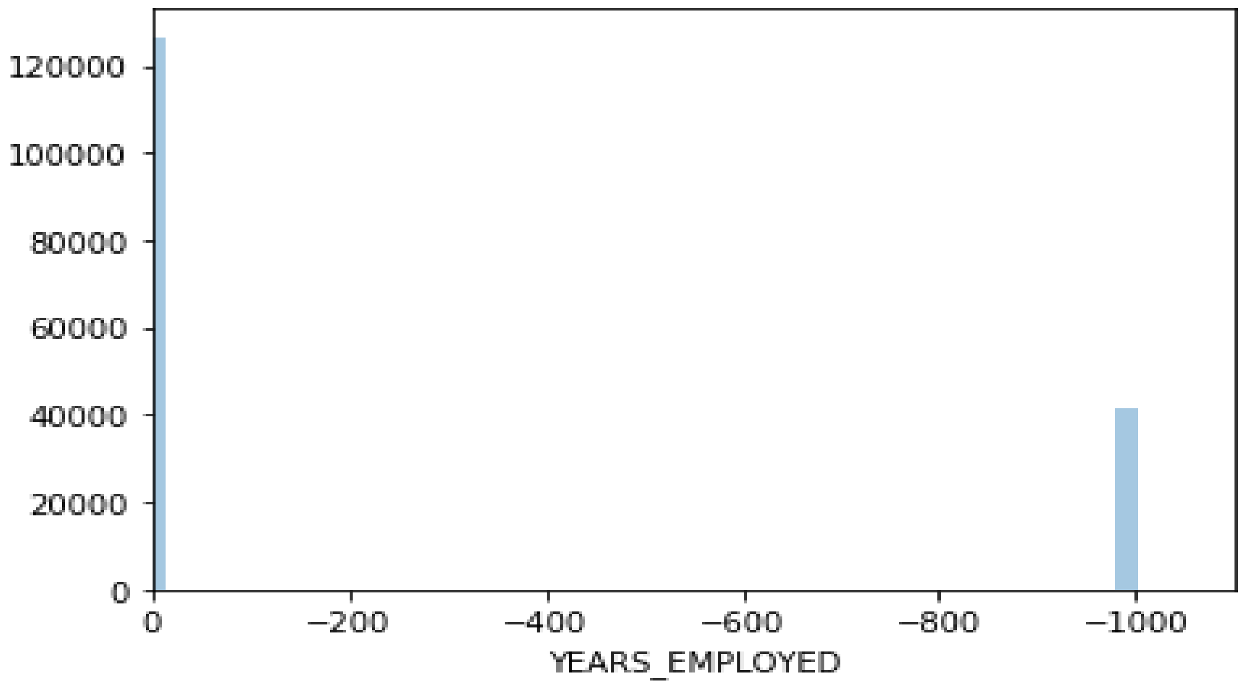


Fig. 1 : Histogram of Years_Employed Column

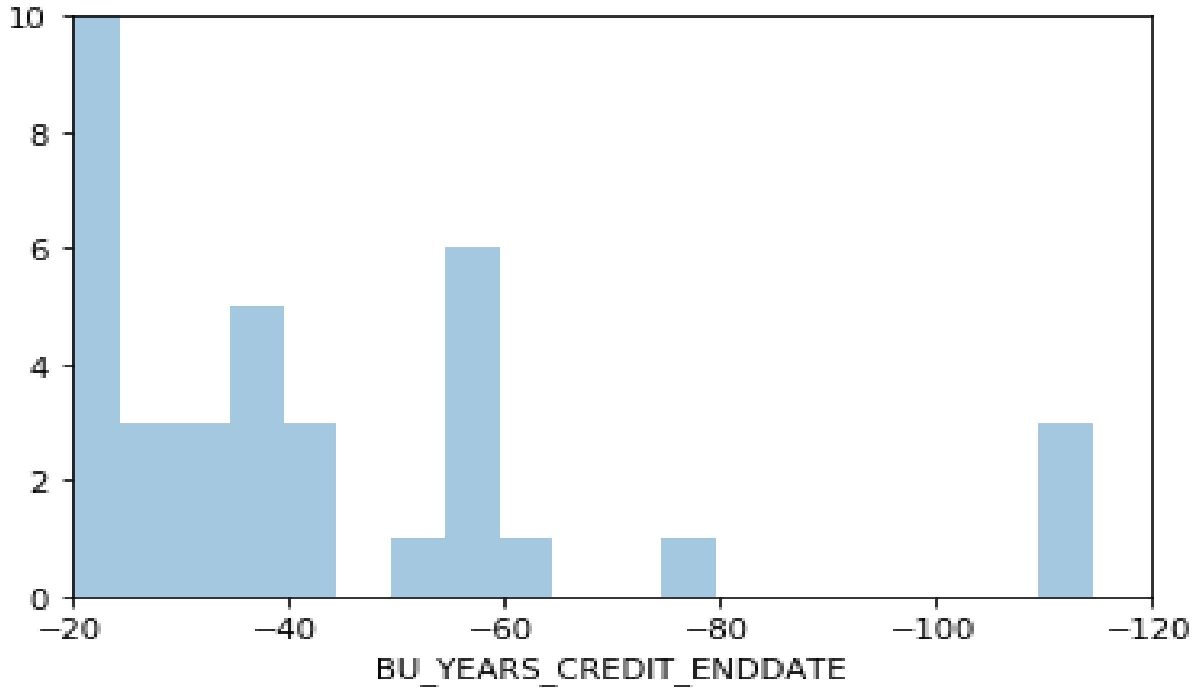


Fig. 2 : Histogram of BU_YEARS_CREDIT_ENDDATE Column

f) The Fig_3 below shows heatmap of correlation coefficients between various numerical columns. It confirms that many columns were significantly correlated with each other, thus linear models such as logistics regression would not perform well on this dataset.

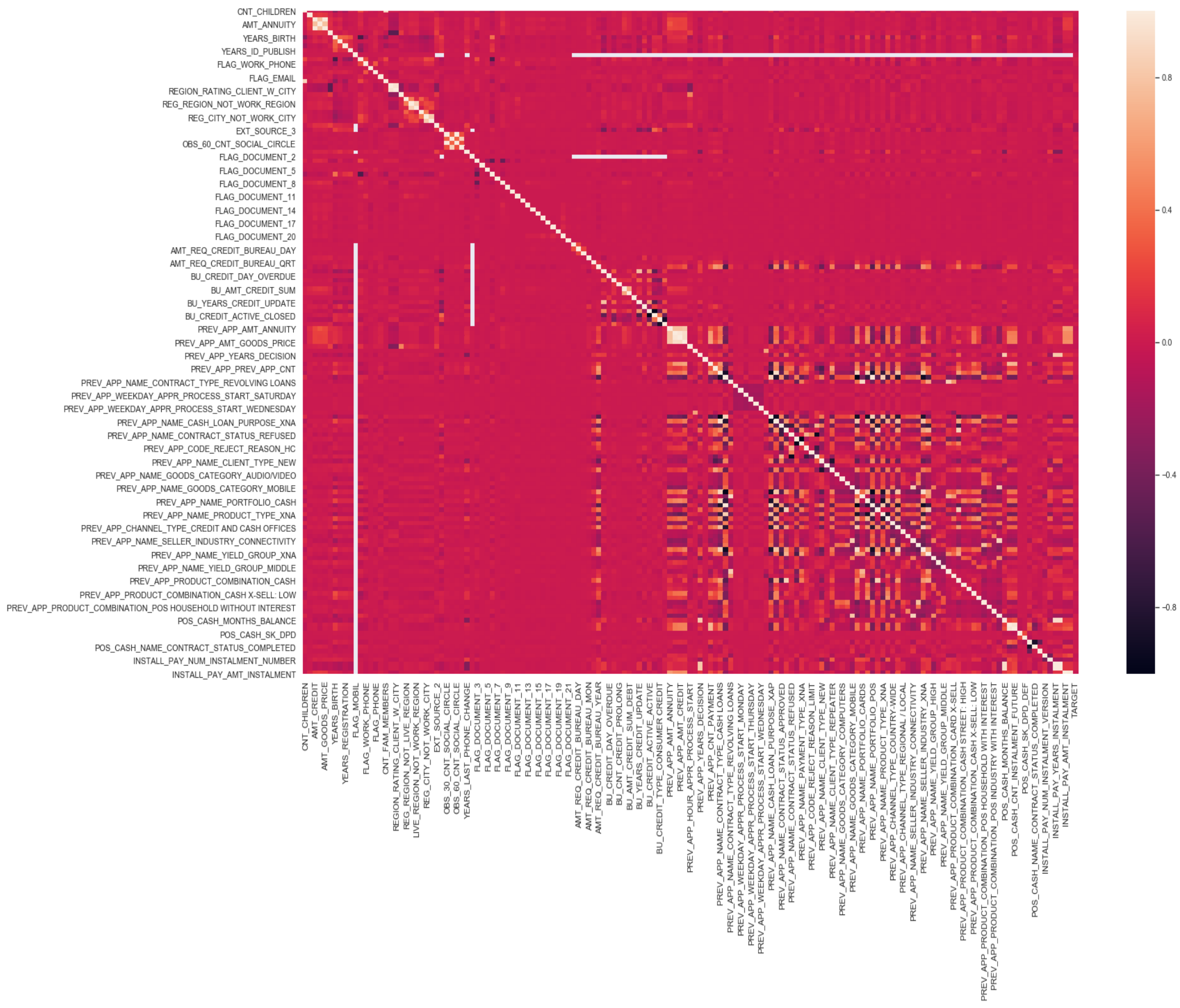


Fig 3: Heatmap of correlation coefficients between various numerical columns.

g) Furthermore all the numerical columns with outliers had their missing values replaced with corresponding columns' median values and the ones without outliers had their NaNs replaced with corresponding mean values.

h) Finally categorical columns were cleaned by reducing the no. of categories in various columns by clubbing them as 'other' category(the categories which didn't occur frequently) & removing the corresponding dummy variable columns(once the dummy variables were created). Further NaN values in various categorical columns automatically disappeared once dummy variables were created.

Finally, after all the columns (numerical & categorical) were cleaned, both training as well as test datasets, were saved for the nest step of feature engineering/selection.

Feature Engineering & Feature Selection

Although right feature engineering is possible with the domain knowledge, three new features which were better suited for our analysis could be readily computed. These three features which were ratios are defined as below:

a) CREDIT_INCOME_RATIO = AMT_CREDIT/AMT_INCOME_TOTAL

b) ANNUITY_INCOME_RATIO = AMT_ANNUITY/AMT_INCOME_TOTAL

c) PAYOVER_TIME_YEARS = AMT_GOODS_PRICE/AMT_ANNUITY

Once these above mentioned new features columns were included in both training & test datasets, the original 4 features from which these new features were derived, were removed from both training & test datasets.

After feature engineering, we were still left with 204 features in both training & test datasets, which were simply too many features. Our dataset would have suffered from **curse of dimensionality,** if we had run any ML models on it & the predictions won't have been reliable. Let's see if we can reduce the dimensionality of the dataset, by removing some noisy features by using **Mutual Information (MI)** between the features (X) and the response variable (y).MI is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. Further, it takes both linear as well as nonlinear dependencies between two RVs into account unlike correlation, which is a measure of only linear dependency.

Since, Sklearn uses KNN to estimate MI; I first normalized all the features in the training set to be in the range [0, 1]. Further after computing MIs between individual features & response variable (Fig_4), I selected all the features whose corresponding MI values > .001 (an arbitrary threshold chosen, keeping in mind the range of MI values). This brought down the dimensionality of feature space from 204 to 103, thereby reducing the dimensionality in the process.

Furthermore, after feature selection, in both training & test datasets, all the numerical features were standardized & the categorical variables were represented by their dummy equivalents.

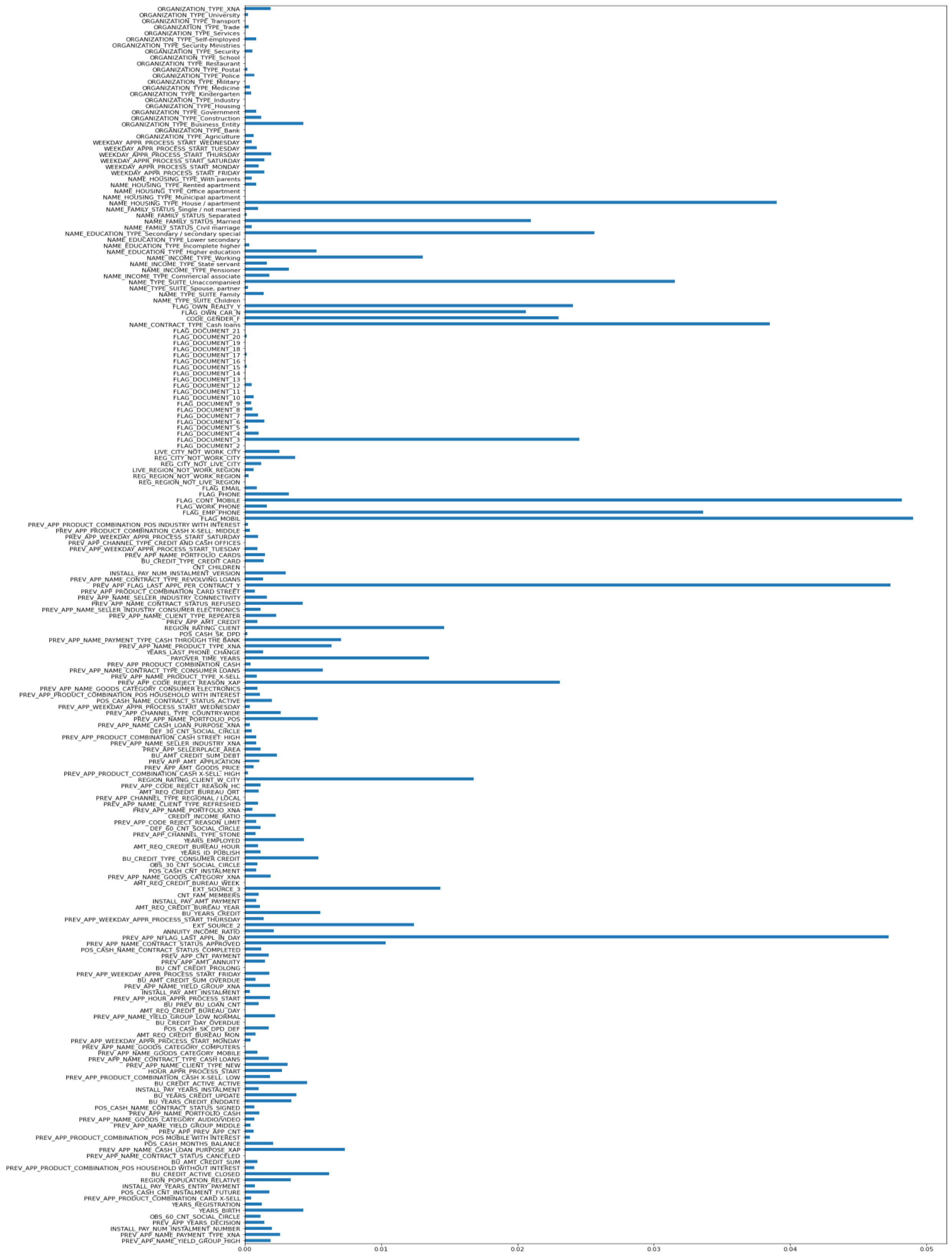


Fig 4: Mutual Information between Features & Response Variable (y)

Baseline Model: Dummy Classifier with default parameters

Dummy classifier is a classifier that makes predictions using simple rules & is useful as a simple baseline to compare with other (real) classifiers.

The dummy classifier with strategy='most_frequent', was fit reduced feature (reduced using mutual information) training set and the following observations were made.

a) The reduced feature training set accuracy, using dummy classifier, was 0.9192700090620163

b) The reduced feature test set accuracy, using dummy classifier, was 0.9192746949712531

c) The reduced feature training set roc_auc score, using dummy classifier, was 0.5

d) The reduced feature test set roc_auc score, using dummy classifier, was 0.5

e) The test set accuracy was 91.93%, which was achieved by labeling all the test set instances (y) equal to 0, the % of class 0 (majority class) in the whole dataset. **Thus, the default dummy classifier was certainly under-fitting the training set & hence we needed more powerful classifiers.**

f) Though the test set accuracy was 91.93%, the corresponding roc_auc score was measly .5, equal to that of random guessing, underscoring the fact that roc_auc score is always a better metric than accuracy, for an imbalanced dataset such as this one.

Model_1: Logistics Regression with tuned Hyper-parameters using Optuna.

Logistic Regression classifier with tuned hyper-parameters (using model hyperparameter tuning library, Optuna) was fit on reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Logistic Regression model was 0.7542487254350991

b) The roc_auc score for the reduced feature test set using the tuned Logistic Regression model was 0.7536336223586382

c) The roc_auc test set score for the tuned Logistic model was much higher than that of the baseline model, which was expected, as the Logistic Regression is more complex model than the simple Dummy Classifier (which underfitted the dataset).

d) The training & test set roc_auc scores were almost equal to each other, indicating that Logistic Regression well fitted the dataset with no indications of overfitting.

Defining Reward_Risk (R_R) Ratio for a Family of Machine Learning Models:

R_R Ratio = *Mean of K Fold CV Score Metric of Training Data / Std_Dev of K Fold CV Score Metric of Training Data*

The R_R ratio may be helpful in choosing among models having approximately same computational complexity or from the same family.

e) R_R Ratio for the tuned Logistic Regression model using roc_auc metric was 144.80149383967927

Model_2: Linear Discriminant Analysis Classifiers

2.1) Vanilla LDA classifier with SVD solver

Vanilla LDA classifier was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the Vanilla lda classifier, with SVD solver, was 0.752784029146089

b) The roc_auc score for the reduced feature test set using the Vanilla lda classifier, with SVD solver, was 0.7533073224907925

c) The R_R ratio for the Vanilla lda classifier using roc_auc metric was 143.64202545415918

2.2) LDA classifier with Eigen Solver with tuned Hyper-parameters using Optuna

LDA classifier, with Eigen Solver, & tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned LDA Classifier with Eigen solver was 0.7527559360951943

b) The roc_auc score for the reduced feature test set using the tuned LDA Classifier with Eigen solver was 0.7533710812916229

c) The R_R ratio for the tuned LDA classifier with Eigen solver using roc_auc metric was 142.3418843832703

2.3) LDA classifier with Eigen Solver & shrinkage utilizing LW Lemma

LDA classifier, with Eigen Solver & shrinkage utilizing LW lemma was fit on the reduced feature training set and the following observations were made.

a) The roc_auc_score for the Reduced feature training set using the LDA Classifier with Ledoit-Wolf shrinkage was 0.7528107805845184
b) The roc_auc_score for the Reduced feature test set using the LDA Classifier with Ledoit-Wolf shrinkage was 0.7532314840830818
c) The reward risk ratio for the lda Classifier utilizing LW lemma shrinkage using roc_auc metric was 144.39223895284022

Final Observations for various LDA Classifiers

a) All the LDA variants fitted the dataset pretty well, with no signs of overfitting & test set roc_auc score were well above that of the baseline model.

b) The LDA model with highest test set roc_auc score corresponded to LDA with eigen solver & tuned shrinkage. This model performed worse than the tuned Logistic Regression Model on the test set. This was expected as the underlying feature space is not multivariate normal and doesn't have the same covariance matrix for both the classes, which are the underlying assumptions of the LDA models. Further the R_R ratio of the tuned Logistic Regression model was greater that of the LDA classifier using LW lemma Shrinkage (which had best R_R ratio among all LDA variants.)

c) As discussed in point b, for this dataset, the Logistic Regression model completely dominated all the 3 LDA variants in all respects & until this point was the model of choice.

Model_3: Quadratic Discriminant Analysis Classifier (QDA) with tuned Hyper-parameters using Optuna

QDA classifier with tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned QDA classifier was 0.7258946495984695

b) The roc_auc score for the reduced feature test set using the tuned QDA classifier was 0.7230740649171299

c) The R_R Ratio for the best QDA classifier using reduced feature set was 118.95544640930348

d) The tuned QDA model had the worst test set roc_auc score as well as R_R ratio of all the fitted models till now. This was expected as the underlying Feature space was not multivariate normal, which is an assumption of the QDA model. Any departure from normality affects the QDA more than the LDA, which was also observed here.

e) Thus, QDA was all but ruled out for this dataset.

<u>Model_4: Light-Gbm classifiers with Tuned Hyperparameters using Optuna.</u>

<u>4.1) Light GBM Classifier_1</u>

Light Gbm Classifier_1 with tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Light Gbm classifier_1 was 0.9516343112915224

b) The roc_auc score for the reduced feature test set using the tuned Light Gbm classifier_1 was 0.7725786275246422

c) The R_R Ratio for the tuned Light Gbm classifier_1, trained on reduced feature training set, using roc_auc score was 166.50499771635376

d) Of all the models fitted till now, Light Gbm classifier_1 had the best test set roc_auc score as well as the R_R ratio.

e) But clearly lgb classifier_1 over-fitted the dataset, as there was a large difference between the training set and test set roc_auc score. This overfitting is typical of most boosting classifier

<u>4.2) Light GBM Classifier_2: A More regularized version of the Light GBM_1</u>

Light Gbm Classifier_2 with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Light Gbm classifier_2 was 0.8443935385323283

b) The roc_auc score for the reduced feature test set using the tuned Light Gbm classifier_2 was 0.7721968636105587

c) The R_R Ratio for the tuned Light Gbm classifier_2, trained on reduced feature training set, using roc_auc score was 167.37634851965672

d) With more regularization, Light Gbm classifier_2 was able to substantially reduce overfitting as compared to that of Light Gbm classifier_1, with the test set roc_auc scores being almost equal for both of them.

e) Further R_R ratio for the Light Gbm classifier_2 was marginally higher than that of Light Gbm classifier_1.

f) **Thus Light Gbm classifier_2 beat the Light Gbm classifier_1 hands down and had the best test set roc_auc as well as R_R ratio of all the models tested till now for this dataset & now had become new model of choice for this dataset.**

<u>Model_5: XG-Boost Classifier with Tuned Hyperparameters using Optuna.</u>

XG-Boost Classifier with tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned XgBoost classifier was 0.8401392081413007

b) The roc_auc score for the reduced feature test set using the tuned XgBoost classifier was 0.7691327155992314

c) The R_R Ratio for the tuned XgBoost classifier, trained on reduced feature training set, using roc_auc score was 154.84871626030798

d) The XgBoost classifier's test set roc_auc score & R_R ratios were slightly less than those of Light Gbm classifier_2. May be more extensive hyperparameter search, might have resulted in better score for XgBoost classifier.

e) The XgBoost classifier was clearly overfitting the dataset, which was evident after looking at the difference between training set & test set roc_auc scores.

f) Tuning XgBoost requires more computational resources, which may be better done on cloud than on PC & since we were already getting good performance using Light Gbm classifier, we didn’t further tune XgBoost classifier for this dataset.

g) Taking everything into consideration, such as overfitting, test set roc_auc score, R_R ratios & computational costs, the best tree based boosting classifier until now was Light Gbm classifier_2.

Model_6: Random Forest Classifiers with different Hyperparameters using RandomSearch CV

Random Forest Classifiers with various hyperparameters using RandomSearch CV were fit on the reduced feature training set and the following results were obtained for 5 fold training & test sets.

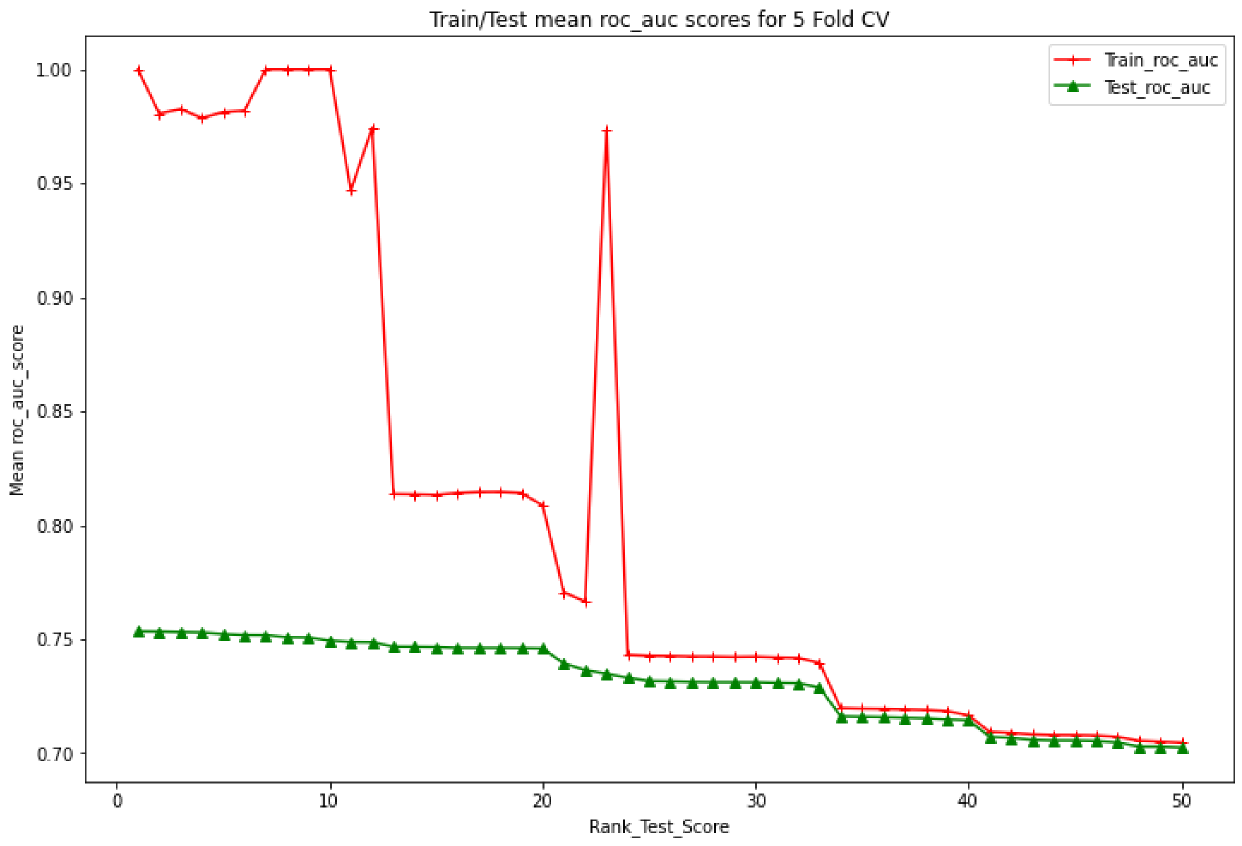


Fig_5: Train/Test 5 fold mean roc_auc score

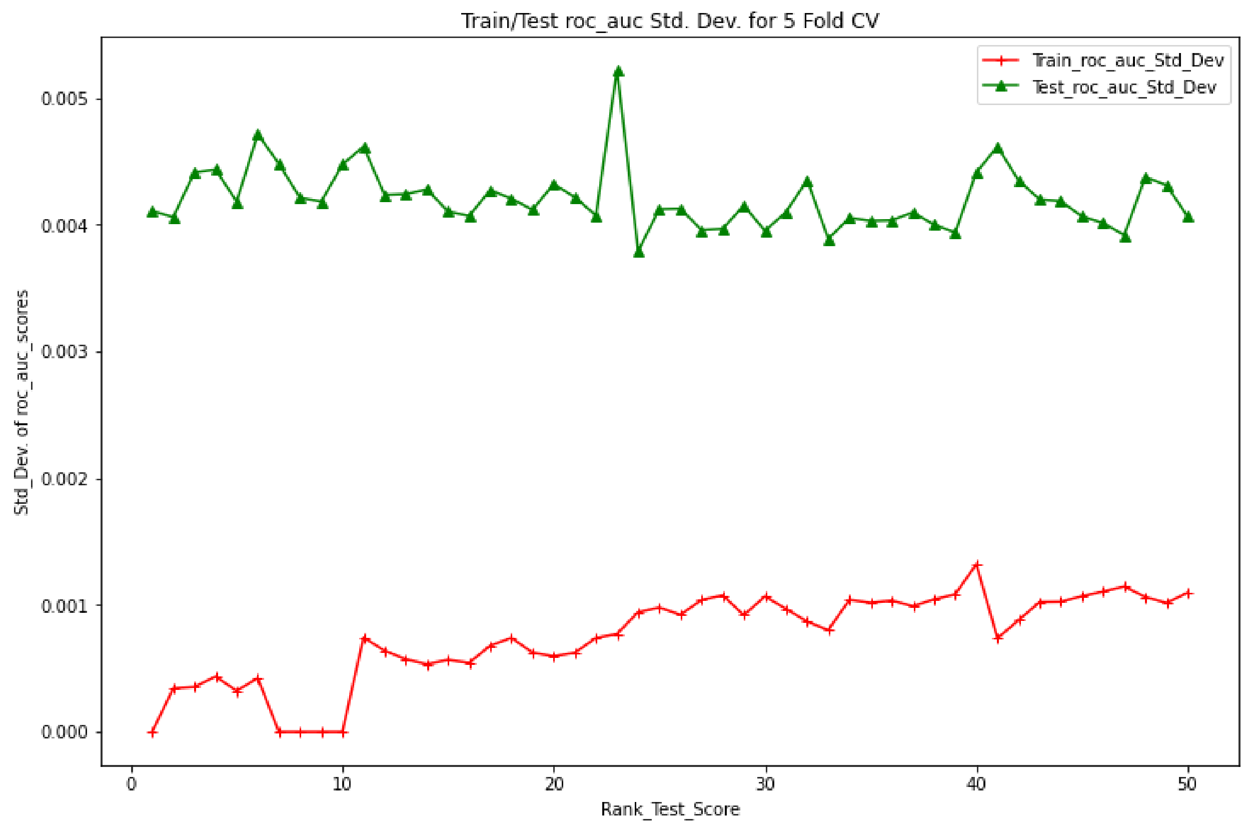


Fig_6: Train/Test 5 fold mean roc_auc Std. Dev

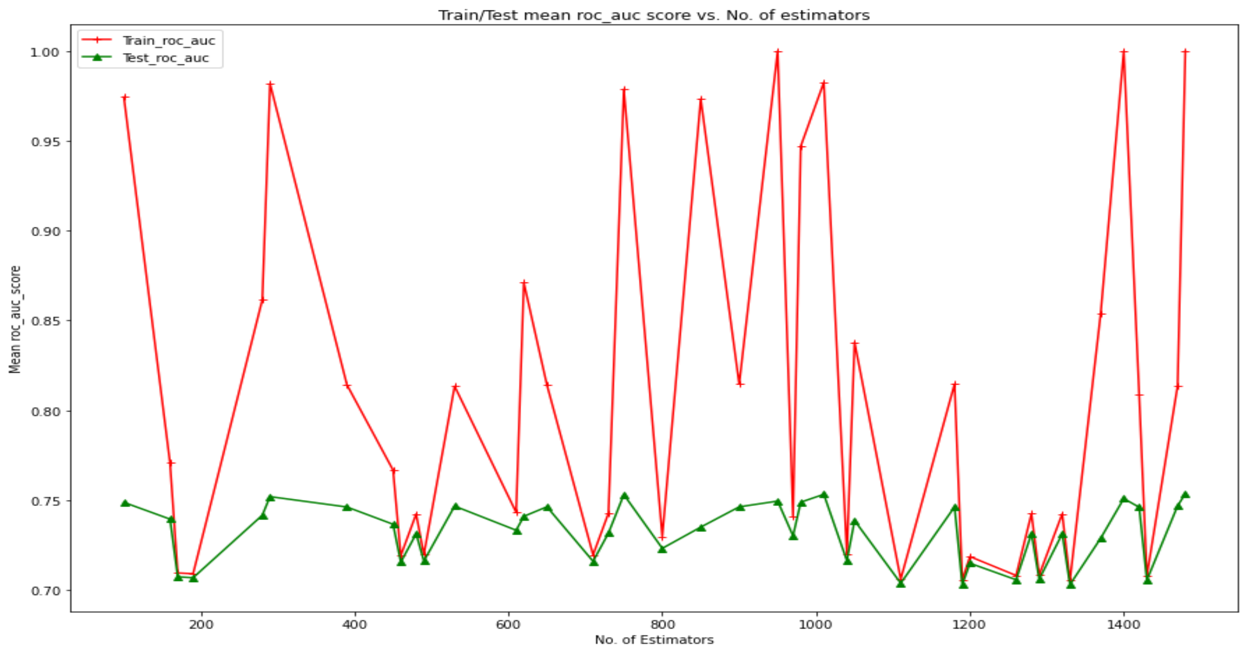


Fig_7: Train/Test 5 fold mean roc_auc vs. No. of Estimators

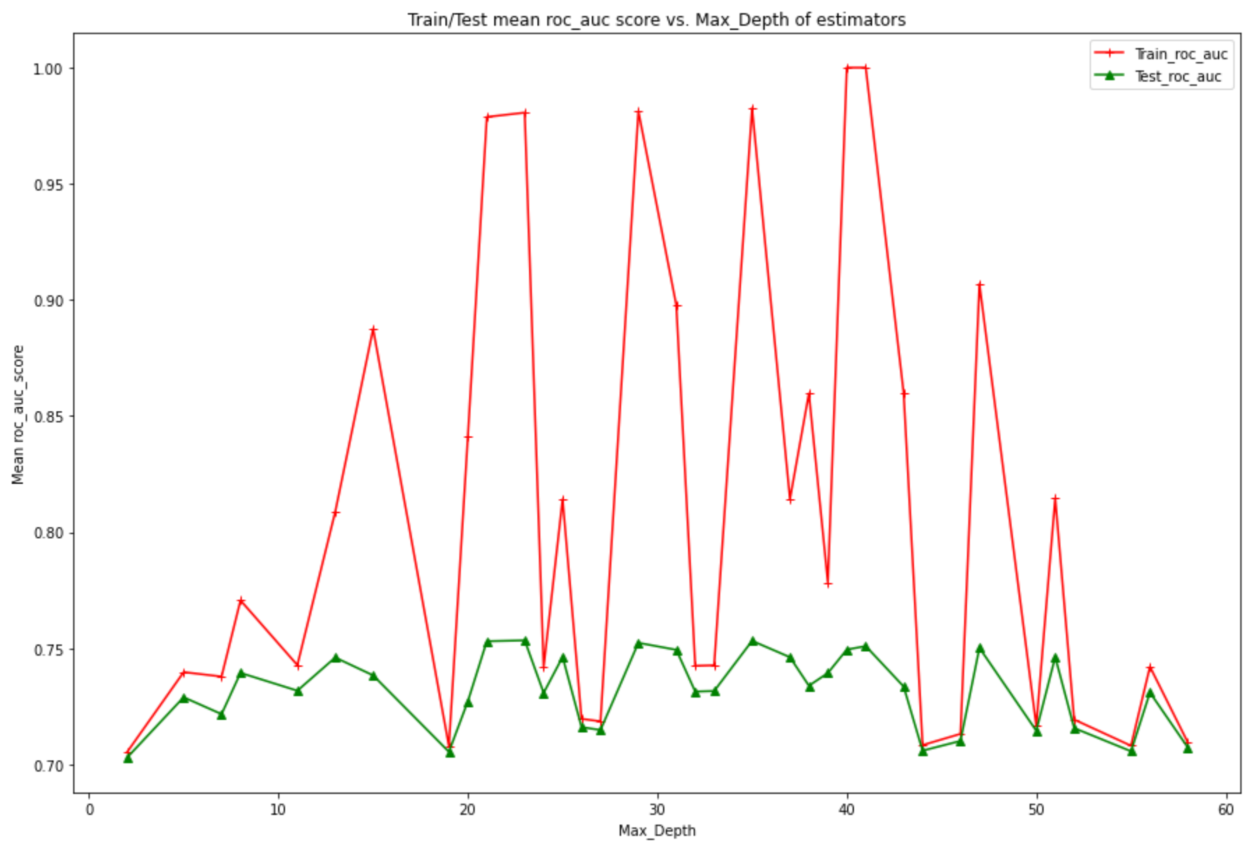


Fig_8: Train/Test 5 fold mean roc_auc vs. Max_Depth of Estimators

The following observations can be made from the above graphs:

1. The roc_auc_score for the reduced feature training set using the best Random Forest Classifier, Random Forest Classifier_1, was 1.0
2. The roc_auc_score for the reduced feature test set using the best Random Forest Classifier , Random Forest Classifier_1, was 0.7592636813381282
3. The training & test set roc_auc score for the Random Forest Classifier_1 were miles apart. Hence, this best model couldn’t be used for forecasting in light of severe overfitting.
4. The aforementioned overfitting is also clearly evident in the previous graphs; with 5 fold mean roc_auc scores for training folds being consistently greater than those of test folds & mean roc_auc Std. Devs. of training folds being consistently lesser than those of corresponding test folds.

Model_7: Random Forest Classifier with Tuned Hyperparameters using Optuna.

Random Forest Classifier_2, with more regularization & Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using (more regularized) tuned Random Forest classifier_2 was 0.8019391919388983

b) The roc_auc score for the reduced feature test set using (more regularized) tuned Random Forest classifier_2 was 0.7516519945423706

c) The R_R Ratio for the tuned (more regularized) Random Forest classifier_2, trained on reduced feature training set, using roc_auc score was 174.68657322808752

d) Thus, out of all the Random Forest Classifiers tested, Random Forest Classifier_2 gave the best results.

Model_8: Extra Trees Classifiers with Tuned Hyperparameters using Optuna.

8.1) Extra Trees Classifier

Extra trees Classifier with tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Extra Trees Classifier was 1

b) The roc_auc score for the reduced feature test set using the tuned Extra Trees Classifier was 0.7545603594015475

c) The R_R Ratio for the tuned Extra Trees classifier, trained on reduced feature training set, using roc_auc score was 200.0351593064627

d) The Extra Trees Classifier was severely overfitting , as can be viewed from the above results. However, typical of an Extra Trees Classifier, its R_R ratio was the highest we had seen so far of any classifier.

8.2) Extra Trees Classifier_1: A More regularized version of the above Extra Trees Classifier

Extra Trees Classifier_1 with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Extra Trees Classifier_1 was 0.8048057417779043

b) The roc_auc score for the reduced feature test set using the tuned Extra Trees Classifier_1 was 0.7463489011999241

c) The R_R Ratio for the tuned Extra Trees classifier_1, trained on reduced feature training set, using roc_auc score was 193.8728094477271

d) As observed above, the Extra Trees classifier_1 had substantially reduced the overfitting, but the test set roc_auc score as well as the R_R ratio also took a beating.

e) The Extra Trees Classifier_1 made use of much larger no. of estimators to achieve scores comparable to the best Random Forest Classifier, viz. Random forest classifier_2, resulting in a more computationally expensive model.

f) **Taking everything into consideration, viz. overfitting, test set roc_auc score, R_R ratios & computational costs, the best tree based bagging classifier was Random Forest classifier_2.**

Model_9: Voting Classifier with Default Weights & Soft Voting

Voting Classifier (with default weights & Soft Voting) having component classifiers namely, best tuned Logistic Regression, Random Forest, Light Gbm & LDA classifiers, was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using Voting Classifier with default weights & soft voting was 0.8046460997742609

b) The roc_auc score for the reduced feature test set using Voting Classifier with default weights & soft voting was 0.7664216167852927

c) The R_R Ratio for the Voting Classifier with default weights & soft voting, trained on reduced feature training set, using roc_auc score was 161.03129442118123

d) The test set roc_auc score for the default voting classifier was more than that of component Logistic Regression, LDA & Random Forest models, but less than that of component Light Gbm. Tuning the weights of voting classifier did result in better performance.

e) Furthermore, the R_R ratio for the default voting classifier model was more than those of component Logistic Regression & LDA models, but less than those of component Random forest & Light Gbm models**. So clearly Light GBM bests default Voting classifier hands down.**

Model_10: Voting Classifier with Tuned weights using Optuna

Voting Classifier (with tuned weights & Soft Voting) having component classifiers namely, best tuned Logistic Regression, Random Forest, Light Gbm & LDA classifiers was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using Voting Classifier with tuned weights & soft voting was 0.8376093835249532

b) The roc_auc score for the reduced feature test set using Voting Classifier with tuned weights & soft voting was 0.7721661743865859

c) The R_R Ratio for the Voting Classifier with tuned weights & soft voting, trained on reduced feature training set, using roc_auc score was 165.12186332349236

d) The tuned voting classifier overfitted the dataset even slightly more than the default voting classifier.

e) The test set roc_auc score of the tuned voting classifier was again more than that of all component models (by a good margin), but for that of the component Light GBM and also exceeded that of un-tuned default version.

f) Further, the R_R ratio for the tuned voting classifier model was more than that of the default version, but still less than that of the component Random Forest classifier.

g) **Taking everything into consideration, the tuned Voting Classifier, even with slightly more overfitting, dominated default voting classifier.**

Model_11: Keras Dense Model with Equal Neurons/layer, SELU Activation & Tuned Learning Rate.

Neural Net Classifier with Equal no. of Neurons/layer, SELU activation and tuned LR rate with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Dense Neural Network (with equal no. of Neurons/layer) classifier was 0.7735250280260082

b) The roc_auc score for the reduced feature test set using the tuned Dense Neural Network (with equal no. of Neurons/layer) classifier was 0.757445351657831

c) Though the tuned SELU dense neural classifier had test set roc_auc score of 0.757445351657831, with almost no overfitting, certain optima trials ended with roc_auc score of 0.5, corresponding to random guessing. It may be that dataset is allergic to NADAM optimizer or SELU activation function. Hence we discarded this model for this dataset.

Model_11.1 : Keras Dense Model with Equal Neurons/layer, SELU Activation, RMS Prop Optimizer & Tuned Learning Rate

Neural Net Classifier with Equal no. of Neurons/layer, SELU activation, RMS Prop Optimizer and tuned LR rate with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) Similar to the previous model, certain optima trials again ended with roc_auc score of 0.5. Thus, SELU activation was not the right activation function for this dataset. Hence we didn't use any neural net with SELU activation on this dataset.

Model_12: Keras Dense Model with Equal Neurons/layer, ELU Activation, RMS Prop Optimizer & Tuned Learning Rate

Neural Net Classifier with Equal no. of Neurons/layer, ELU activation, RMS Prop Optimizer and tuned LR rate, with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Dense Neural Network (with ELU activation & RMS Prop optimizer) classifier was 0.7455322275775218

b) The roc_auc score for the reduced feature test set using the tuned Dense Neural Network (with ELU activation & RMS Prop optimizer) classifier was 0.7453847568202309

c) The R_R Ratio for the Dense Neural Network (with ELU activation & RMS Prop optimizer), trained on reduced feature training set, using roc_auc score was 202.64567650164165

d) This Dense Neural Network (with ELU activation & RMS Optimizer) classifier fitted the Training set very well, with no overfitting at all. Though it had the highest R_R ratio of all the models fitted till now, its test set roc_auc score was the one of the smallest of the lot.

e) Though, ELU activation function clearly fitted the dataset well, the next model with NADAM optimizer resulted in the better results.

Model_13: Keras Dense Model with Equal Neurons/layer, ELU Activation, NADAM optimizer & Tuned Learning Rate

Neural Net Classifier with Equal no. of Neurons/layer, ELU activation, NADAM Optimizer and tuned LR rate, with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Dense Neural Network (with ELU activation & NADAM optimizer) classifier was 0.7727873525704507

b) The roc_auc score for the reduced feature test set using the tuned Dense Neural Network (with ELU activation & NADAM optimizer) classifier was 0.7627502861257128

c) The R_R Ratio for the Dense Neural Network (with ELU activation & NADAM optimizer), trained on reduced feature training set, using roc_auc score was 149.5339076085092

d) The Dense Neural Classifier (with ELU activation & NADAM Optimizer) beautifully fitted the Training set, with no signs of overfitting. But though its test set roc_auc score was better than that of the previous dense neural net (with ELU activation & RMS Optimizer), its R_R ratio was much less than that of the latter.

e) The Dense Neural classifier, discussed next, with Leaky RELU activation produced better R_R ratio with almost equal test set roc_auc score as the current model.

Model_14: Keras Dense Model with Equal Neurons/layer, Leaky RELU Activation, Nadam optimizer & Tuned Learning Rate

Neural Net Classifier with Equal no. of Neurons/layer, Leaky RELU activation(α=.2), NADAM Optimizer and tuned LR rate, with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Dense Neural Network (with Leaky RELU activation & NADAM optimizer) classifier was 0.7683335420626638

b) The roc_auc score for the reduced feature test set using the tuned Dense Neural Network (with Leaky RELU activation & NADAM optimizer) classifier was 0.7622685941311187

c) The R_R Ratio for the Dense Neural Network (Leaky RELU activation & NADAM optimizer), trained on reduced feature training set, using roc_auc score was 160.313391453082

d) Unfortunately, a big disadvantage of the above classifier was enormous no. of parameters that needed to be tuned, which would add to the computational complexity. In-fact, of all the neural nets tested thus far, this model used most no. of layers as well as neurons/layer.

e) The next Neural net with RELU activation remedied the above mentioned disadvantage.

Model_15: Keras Dense Model with Equal Neurons/layer, RELU Activation, Nadam optimizer & Tuned Learning Rate.

Neural Net Classifier with Equal no. of Neurons/layer, RELU activation, NADAM Optimizer and tuned LR rate, with Optuna tuned hyper-parameters was fit on the reduced feature training set and the following observations were made.

a) The roc_auc score for the reduced feature training set using the tuned Dense Neural Network (with RELU activation & NADAM optimizer) classifier was 0.7705065237291805

b) The roc_auc score for the reduced feature test set using the tuned Dense Neural Network (with RELU activation & NADAM optimizer) classifier was 0.7625576360042436

c) The R_R Ratio for the Dense Neural Network (RELU activation & NADAM optimizer) classifier, trained on reduced feature training set, using roc_auc score was 176.57074673639582

d) The Dense Neural (with RELU activation & NADAM Optimizer) classifier fitted the training set very well, with no signs of overfitting. Also, it used one of the fewest no. of parameters (among all the Neural nets considered here) that needed to be tuned.

e) One big advantage of this model was that both its test set roc_auc score and R_R ratio were better than the corresponding values of the previous dense neural net (one with Leaky RELU activation & NADAM Optimizer), which was a significant improvement. **In-fact, this above classifier had one of the highest test set roc_auc score as well as the R_R ratio of all the afore-tested Neural Nets.**

f) **Taking everything into consideration, Dense Neural Classifier (with RELU activation & NADAM Optimizer), dominated all other Neural Nets & thus was clearly the winner in the neural net category for this dataset.**

<u>Model_16: Weighted Aggregating Classifier</u>

This classifier computes the weighted aggregation of the predicted probabilities of constituent models (tuned Voting classifier & Neural net with equal neurons/layer, RELU activation & NADAM optimizer here). Weighted Aggregating Classifier with Optuna tuned weights was fit on the reduced feature training set and the following observations were made.

a) The training set roc_auc score of the weighted aggregating classifier using the Optuna tuned weights was 0.768794222802716

b) The test set roc_auc score of the weighted aggregating classifier using the Optuna tuned weights was 0.7726740445628326

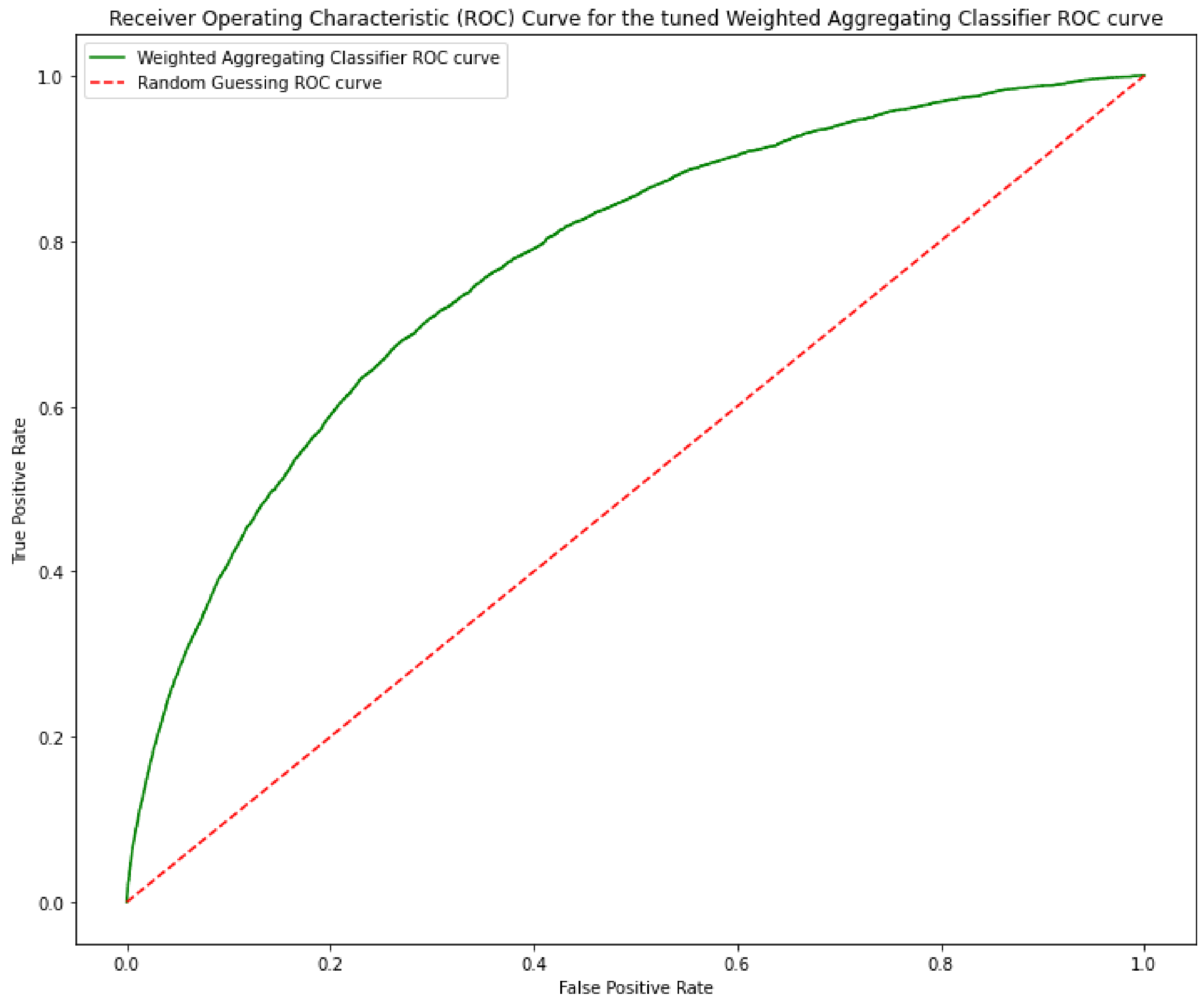
c) The R_R Ratio for the tuned weighted aggregating Classifier using roc_auc metric was 168.00930403535745

d) The Weighted Aggregating Classifier fitted the training set very well, with no signs of overfitting. Also its test set roc_auc score was the highest of any classifier tested till now on this dataset. Further Its R_R ratio was also on the higher side of the spectrum. **Thus this classifier was one of the best classifiers (among those tested) for this dataset.**

<u>Conclusion</u>

Considering all the aspects such as overfitting, test set roc_auc score, R_R ratio etc., Weighted Aggregating Classifier was the best choices among the fitted models. Hence we would use it for making forecasts on future observations and for calculating the optimal threshold probability for this dataset (*Note: Optimal Prob. threshold is one where diff between tpr & fpr is max.)*

Let’s calculate the optimal probability threshold value from the following roc curve that would give us the best TPR or Recall viz a viz FPR on the test or unseen data



From the above roc plot, we determined (fpr, tpr) point, with coordinates (0.2696824767942042, 0.6788591685465678) corresponding to the optimal probability threshold (0.5039920234281092). **Thus, in order to maximize recall viz. a viz. FPR, any test observation with conditional probability (outputted by Weighted Aggregating Classifier) > 0.5039920234281092, should be classified as belonging to class 1.**

Suggestions For Further Improvements

Although, we employed some feature engineering, feature selection & trained various models, some more feature engineering (which would have been possible if we had more metadata about the dataset) & training better ML models on this dataset could surely have produced better R_R ratios as well as test set roc_auc scores. But this would require more business domain knowledge as well as computational resources especially when dealing with high computational cost models such as SVMs, Neural Nets etc. **Thus as a next step, I would recommend making use of cloud computational resources which can enable one to experiment with many different classifiers and their varied architectures and hence produce better results. Further with fast computational cloud resources, instead of dropping the whole column having > 30% missing values (as was done here); we can impute missing values by MICE (Multiple Imputations by Chained Equations).**