UNIVERSITY OF HYDERABAD

Project titled

“Predicting the capability of each applicant in repaying a loan”

Thesis submitted in fulfillment of “Diploma in AI and ML” by

Saurabha Daa

Enrl. No.: 40AIML529-21/1

Email: saurabhadaa@gmail.com

CONTENTS

1.0.0.0. Literature survey & Data Acquisition

1.1.0.0. Problem Description

1.1.1.0. The problem at hand is an interesting and real-life business problem which is faced daily by financial lenders. The problem is about automating the loan approval process which in turn will impact the KPIs (Key Performance Index) of profitable lending teams.

A lending company - Home Credit - wishes to expand its lending business and provide loans to people with insufficient or non-existent credit histories. For taking the decision of whether to give loan to an individual or not, machine learning model(s) shall be trained and deployed to measure a clients' repayment abilities. For training these models 2.68 GB of data is provided in form of various csv files. These are a variety of in-house data and data collected from other sources.

1.1.2.0. The importance and impact of this problem are as under:

1.1.2.1. Profitability of lending company - If the process of decision making is automated it will positively affect KPIs like velocity, pull through, cost-to-close [a] which in turn enhance the profits of the lending company. It is quite evident that the time saved by automating this process will result in fast and efficient working of lending companies which translate to increase in profit. The trained model (for automating the process of loan approval) will be evaluated against various metrics (to be discussed subsequently). This evaluation is necessary so that the credit worthy people are not deprived from loans and the credit unworthy are not given loans.

1.1.2.2. Social and financial inclusion of people with insufficient or non-existent credit histories - The presence of organised and trustworthy lenders for the unbanked population is important else this population will be exploited by the unorganised and untrustworthy lenders. Loans from organised lenders will also ensure that the credit worthiness of this population is documented which in turn makes it easy for the same and other lenders to give loans to this populsation.

1.2.0.0. Dataset

1.2.1.0. Data (2.68 GB) sourced from kaggle [b] is in the form of csv files. There are 8 csv files containing data tables and 1 csv file containing the columns description of the other 8 files. To understand the data set it is important to go through all the tables represented by these csv files with the understanding of each column from HomeCredit_columns_description.csv. The data set is huge in size and needs considerable time to be spent to correlate various tables. The data files can be broadly classified into three categories which are listed below with brief description of each data table:

1.2.1.1. Train and test data - These are data provided for training and evaluating the model:
application_train.csv - This data table consists of 122 columns. Each row indicates an entry corresponding to 1 loan application. The column titled SK_ID_CURR is the

main identifier for each entry. Column titled TARGET has two values - 1 for clients with payment difficulties or default and 0 for clients without payment difficulties or no default. There are several other columns e.g., columns indicating whether the client has a car or not, gender of client, loan annuity, income of client, age of client, the day on which application was made, detail of client's home etc. We have a lot of features/parameters.

application_test.csv - The columns of this data table are the same as columns of application_train.csv except for the absence of the TARGET column. This data set shall be used to evaluate the model trained using application_train.csv.

1.2.1.2. Credit data from other sources - It consists of data tables which contain information of applicants collected from sources other than Home Credit and which were reported to Credit Bureau (Credit Bureau is a centralised agency which documents the data of all financial lenders). Following are the data sets in this category:

bureau.csv - This consists of data of the client's previous credits with each row indicating 1 credit. Thus there can be multiple rows pertaining to 1 client. The identifier for a credit is given in SK_BUREAU_ID and it is related to the client identified by SK_ID_CURR column.

bureau_balance.csv - This data table consists of monthly data against the credits specified in bureau.csv. The identifier for credit is SK_BUREAU_ID.

1.2.1.3. Credit data from client's previous application with Home Credit - It consists of data tables which contain information of applicants who are already availing loans from Home Credit. Following are the data tables in this category:

previous_application.csv - This data table consists of data of a client's previous loan application with Home Credit. Each row indicates a previous loan by a client where a previous loan is identified by SK_ID_PREV column and client is identified by SK_ID_CURR column.

POS_CASH_balance.csv - The rows in this data table indicate each month of previous cash loan with Home Credit.

credit_card_balance.csv - The rows of this data table capture the details of each month of each previous credit card the client has with Home Credit.

installments_payment.csv - This data table consists of 1 row of every payment that was made and one row for every missed payment pertaining to previous credits of clients from Home Credit.

1.2.2.0. Challenges with this data set - The data set is huge and hence requires considerable time to be spent for finding the correlation among different data tables. At the same time the pre-processing will involve a lot of effort for cleaning involving deduplication, missing value imputation, removal of data with values which are out of bounds etc. While so many tables provide opportunity for feature engineering, it also comes with challenges like creating and trying different features and finding which are really impacting the result.

1.2.3.0. The data is furnished in the form of csv files. Pandas and Numpy shall be used to read the data and do various processing to bring the data in the form which can be fed in the machine learning model. Libraries like Matplotlib, Seaborn etc. shall be used for visualisation. The choice of library shall be dynamic in nature and will depend upon the need as perceived during different phases of this project.

1.2.4.0. Data acquisition from other sources can only be possible if the lending companies are ready to share their data. It is also possible for someone to have access to other company's data if they are working for that company and have necessary authorization or are working with the regulatory board. Such data is sensitive for any lender and not publicly available and hence not possible to get easily.

1.3.0.0. Key metric (KPI) to optimize

There are certain business metrics to evaluate for financial lenders. Some examples of these are velocity, pull through, cost-to-close [a]. These are financial lending domain specific metrics and can be understood in easy language through references made in this document. However, the metrics used for the machine learning task at hand shall be measurements like accuracy, deductions from confusion matrix (Precision, Recall, F1 score etc.), AUC (area under curve) in ROC (receiver operating characteristic) curve etc.

Data is expected to be highly imbalanced as for any profitable lending business the default rate has to be very low. In such a situation, accuracy will not be helpful as accuracy can be high just by using a simple yes or no model in case of imbalanced data. Hence, metrics related to confusion matrix and AUC in ROC curve will be useful in this case.

In this case the requirement is that the genuine clients should get a loan and expected defaulters should not get a loan. The cost of false positive and false negative both are high, especially false positive (where loan is approved for lenders who will default). Similar situations arise where the cost of false negatives is very high e.g., medical; and in case of detection of fraudulent transactions. In all these situations the metrics discussed above are used for evaluating the model.

An implementation of these metrics from scratch for a toy data set is annexed with this document. The implementations also contain comparison of results with those obtained from standard libraries. Moreover, explanations of codes are also given in the form of comments. Complete code is split into logical sections. Implementation of these metrics through standard libraries shall be done in relevant phases of this project.

1.4.0.0. Real world challenges and constraints

1.4.1.0. The various challenges and constraints are discussed as under:

1.4.1.1. Size and structure of data - 2.68 GB of data with 7 data tables for training, 1 data table for evaluation and hundreds of columns is a lot to process. While handling is

not difficult with so many optimised libraries, data preprocessing/cleaning becomes a challenge with this size of data. As there are a lot of columns spread across 8 data tables, a lot of correlation has to be understood across different tables. This needs considerable time and effort.

1.4.1.2. Imbalance in data - The data is expected to be imbalanced. So accuracy cannot be a reliable metric for evaluation of a model. This calls for use of other metrics for evaluation e.g., confusion matrix, AUC in ROC curve.

1.4.1.3. Domain knowledge - This is financial domain data. An understanding of the basic concepts and sometimes applied concepts is required for a machine learning engineer for feature engineering. The lending business involves understanding various ratios [c] which are key metrics while deciding credit-worthiness of a person (i.e., whether a person shall be given a loan or not).

1.4.1.4. Data cleaning - A cursory view of the tabular data reveals that a lot of data is missing. Missing value imputation shall be done in this situation. However, the missing value imputation methodology will vary from column to column. It may happen that some columns have to be dropped as there is a very high percentage of missing value. Data needs to be checked for values which are out of bound and such data need to be removed or replaced. Data shall be checked for deduplication and removal of duplicate rows. Categorical data shall be converted into numerical data. For this one hot encoding shall be used. All these shall be done along with EDA (exploratory data analysis).

1.4.2.0. While implementing machine learning, more than 1 model shall be tested and the best shall be deployed. This model is expected to give low false positives and low false negatives.

1.5.0.0. <u>Approach to solve similar problems:</u>

1.5.1.0. After reading blogs and literature available on the internet [d], it is found that the approach to solve such problems involves EDA and data cleansing as first steps. EDA helps us get insight into data and gives an idea of the various types of data cleansing to be employed. After the data is cleaned, more than 1 model is tried. The most common models that are applied are logistic regression, tree based models and neural networks.

1.5.2.0. My approach towards solving this problem will include the following. The details shall be documented during the relevant phases. The approach and methodology may evolve over time as we get more insight during different phases of this project:

1.5.2.1. Studying the data and finding correlation among different data tables is the first step. During this step various domain specific insights shall also be collected which will help in feature engineering if required. This step is the phase 1 (current phase) of the project thesis.

1.5.2.2. EDA and data cleansing shall be done simultaneously. During EDA, various insights of data shall be obtained. This will give a bird's eye view of the correlation that exists among various parameters.

1.5.2.3. Machine learning models shall be evaluated. As it is a classification problem, the first choice is logistic regression. Tree based models and neural networks will also be tried. Based on the performance of these models, a final model will be finalised for deployment.

1.5.3.0. During the course of the project, this document's previous phases may be updated owing to learning and evolution of ideas & approach.

1.6.0.0. References:

[a] https://himaxwell.com/resources/blog/5-kpis-profitable-lending-teams-measure/, https://smeloan.sg/blog/5-commercial-loan-ratios/

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

[c] https://corporatefinanceinstitute.com/resources/knowledge/finance/lending-ratios/, https://www.investopedia.com/terms/c/credit-worthiness.asp

[d] https://towardsdatascience.com/machine-learning-predicting-bank-loan-defaults-d48bffb9aee2, https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012042/pdf

2.0.0.0. EDA and Feature Extraction

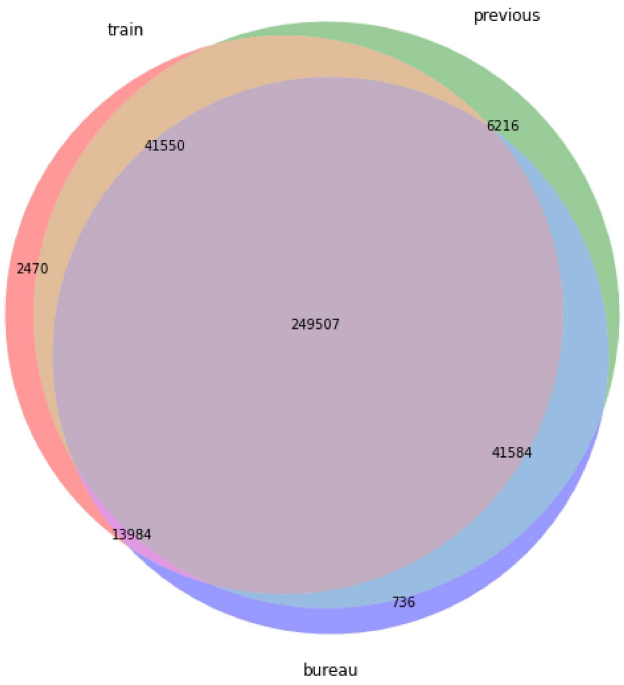
The ipynb file corresponding to this phase is divided into sections. This phase 2 documentation has to be read along with the ipynb file for correlation of different terminologies and outputs. Moreover, only a few visualisations are reproduced here just to give a glimpse. Detailed visualisations can be found in the ipynb file. The description given here follows the same section wise approach as the ipynb file.

2.1.0.0. Common commands: In this section google drive is mounted for accessing data files, important packages are installed and relevant libraries are imported. Three custom functions - one for dataframe optimisation, other for plotting group lot and last for plotting pie chart - are also defined.

2.2.0.0. Data set level analysis: This is high level analysis performed on the csv files provided as input. This analysis helps in getting a general overview of data and deciding the kind of operations to be performed during data preparation.

2.2.1.0. Number of data points are calculated for application_train, application_test and their overlap & exclusion with datpoints in bureau and previous_application.

2.2.2.0. It is observed that 14.31% of applications from application_train are not available in bureau. 5.35% of applications from application_train are not available in previous_application. Overall 81% of applications from application_train are available either in bureau or in previous_application. The overlaps are indicated through a venn diagram in the ipynb file corresponding to this stage of the project.An example of actual venn diagram is as under:



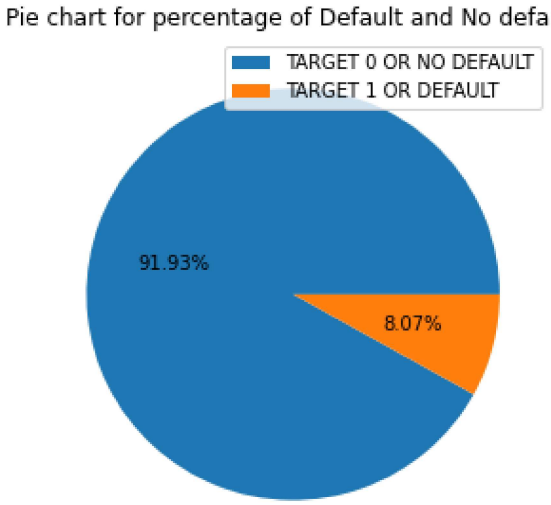
2.2.3.0. It is observed that 13.18% of applications from application_test are not available in bureau. 1.94% of applications from application_test are not available in previous_application. Overall 85.31% of applications from application_test are available either in bureau or in previous_application. The overlaps are indicated through a venn diagram in the ipynb file corresponding to this stage of the project. It is decided that features/columns from bureau and previous_application shall be added to application_train and application_test as a part of feature engineering to get more features.

2.2.4.0. It is observed that 67 columns of application_train and 64 columns of application_test have null values. In some of the columns more than 50% of the values are missing.

2.2.5.0. All the columns shall be retained and missing values shall be tackled using imputation.

2.3.0.0. Univariate and multivariate analysis: With the domain specific knowledge collected during 1st phase of this project, univariate and multivariate analysis are performed on select features. During multivariate analysis the common feature is TARGET value as the ultimate business objective is to reduce default. These univariate and multivariate analysis will give us an insight in the percentage of default among various categories/groups. However, the decision cannot be based only on these analyses alone. Overall decision shall be taken based on final model output and its interpretation. Group plot and pie charts are plotted in the ipynb file. A few of them are reproduced here.

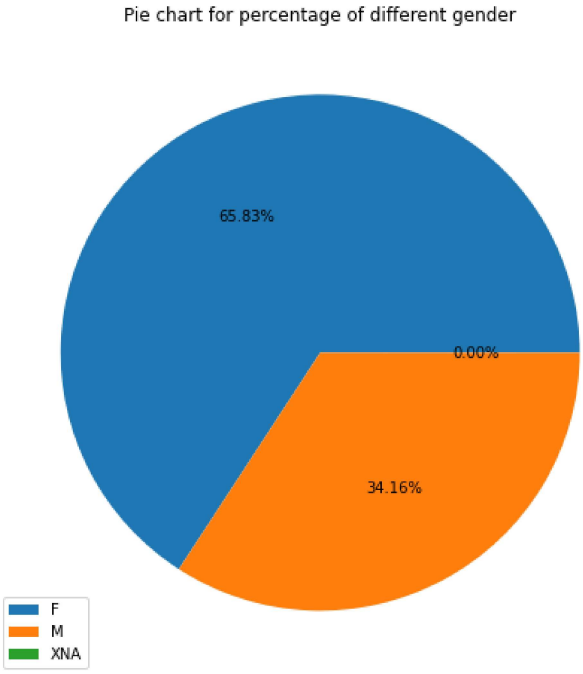
2.3.1.0. An analysis of default indicates that 8.07% of applicants in application_train are defaulters. From a business perspective, this percentage needs to be reduced. From a machine learning perspective, the dataset is imbalanced.



2.3.2.0. An analysis of family status of applicants and default indicates that maximum applicants are married and percentage of default is lower among married people. From a business point of view, married people should be further targeted for loans.
Default among widows is also very low however, the percentage of widow applicants is low. This group can be targeted and further analysed till there is a substantial population in this group.
Civil marriage and Single / not married categories are where focus should be to reduce the percentage of defaulters. The loan process may put an extra check for these groups.

2.3.3.0. An analysis of the type of loans and default indicates that the percentage of cash loans is much higher than revolving loans. The percentage of default is lower in case of revolving loans. Thrust should be laid on disbursing revolving loans, which generally have variable rates of interest.

2.3.4.0. An analysis based on gender and default indicates that the number of male applicants is double the number of female applicants. This result also has a social connotation apart from business impact. It can be concluded that more number of male are earning compared to females and a gender disparity exists. Policy makers will have the responsibility to remove this disparity or evaluate their steps taken earlier.



As the percentage of female defaulters is lesser than male defaulters, this calls for promotion of loan among females.
From this preliminary analysis, it seems that gender is an important parameter. However, the decision to retain or remove this parameter shall be taken after mathematical analysis of data in the feature selection section.

2.3.5.0. For conducting analysis based on income, the incomes data is divided among bins of 10 percentile each. Group plot for number and percentage of defaulters for each bin is plotted. There is not much significant disparity among different income groups except for 135000.0-147150.0 where loan applicants are very low.

Loan applicants are more in lower income groups compared to higher income groups. Loan defaulters are also leeser in higher income groups compared to lower income groups.

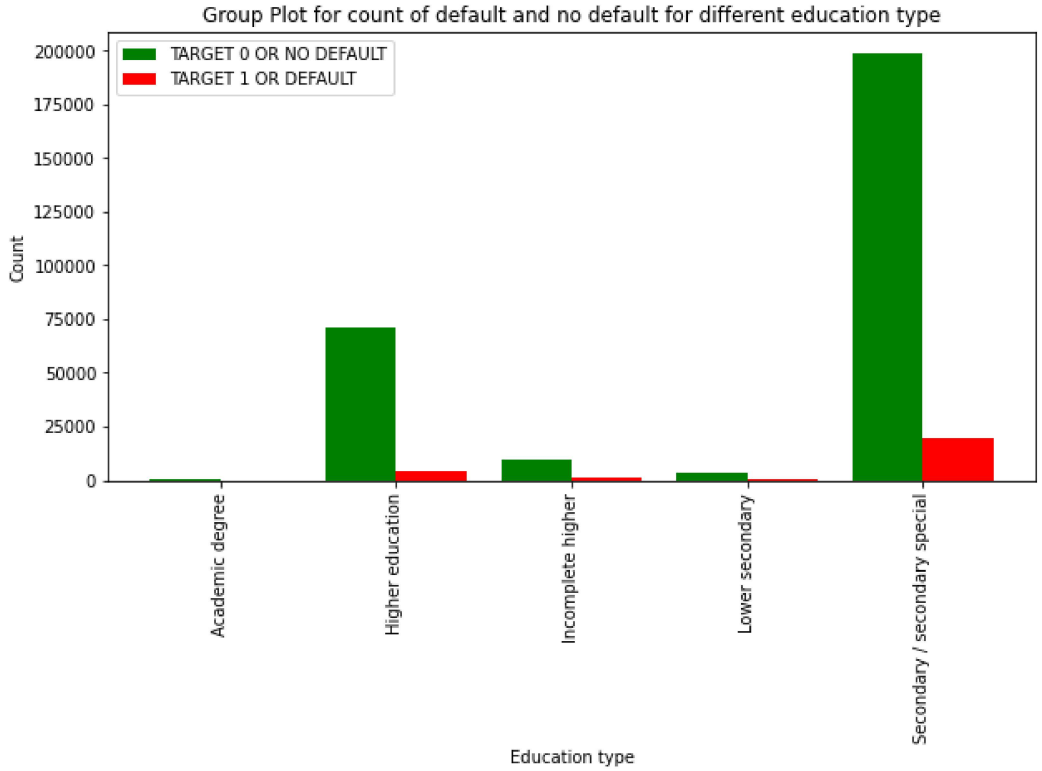
Two particular income groups with higher percentage of applicants are 112500.0-135000.0 and 180000.0-225000.0. This data also indicates a few inordinately high income applicants. These data points may be outliers which will be dealt separately in the outlier detection and removal section.

2.3.6.0. An analysis based on income type indicates huge disparity among the groups with different income types.

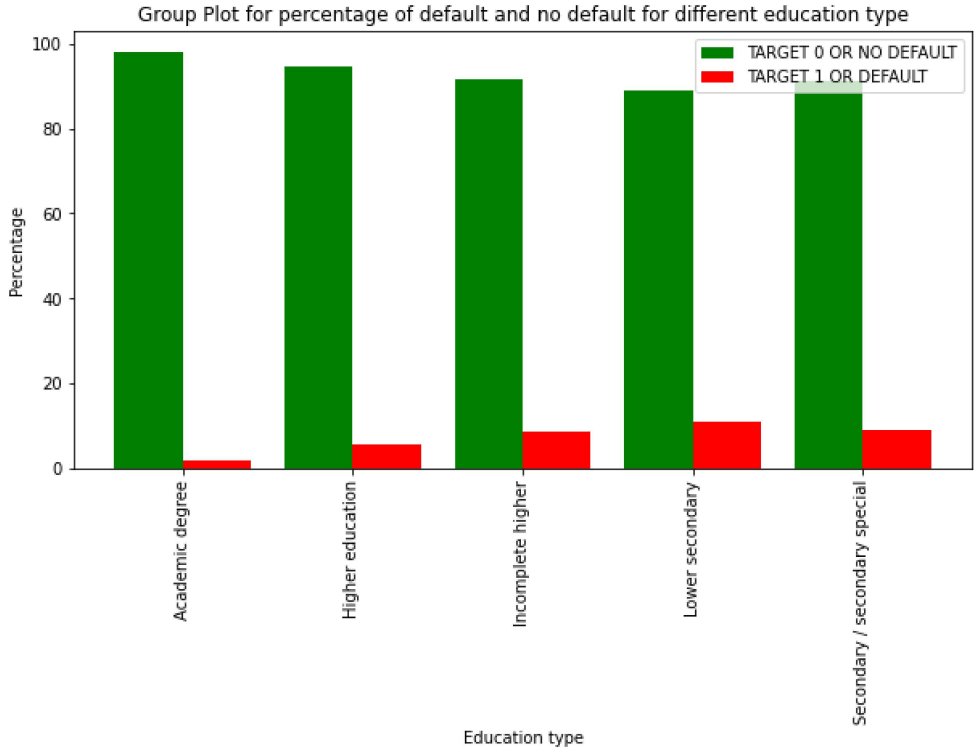
The default rate is very low among pensioners and government servants. They become a target group for promotion of loans and further evaluation of results. Groups with lower numbers of applicants can be targeted for loans and results shall be evaluated after some time.

From this preliminary analysis, it seems that income type is an important parameter. However, the decision to retain or remove this parameter shall be taken after mathematical analysis of data in the feature selection section.

2.3.7.0. An analysis based on education type indicates huge disparity among the groups with different education types.



It is observed that as the level of education increases, the percentage of default decreases. Education is in general correlated to income. Hence, this observation suggests that groups with higher levels of education should be targeted for loans.
From this preliminary analysis, it seems that education is an important parameter. However, the decision to retain or remove this parameter shall be taken after mathematical analysis of data in the feature selection section.



2.3.8.0. 31.35% data is missing in case of occupation type. Among the available data, labourers constitute the highest percentage of defaulters.
Groups with low percentage of default e.g., Accountants, Core staff, HR staff, High skill tech staff, IT staff etc., should be targeted for disbursing loans.
Accountants have the lowest percentage of default and also low percentage of application. They form an important target group.
From this preliminary analysis, it seems that occupation type is an important parameter. However, the decision to retain or remove this parameter shall be taken after mathematical analysis of data in the feature selection section.

2.3.9.0. An analysis based on day of the week indicates, most loan applications are done on weekdays. Number of loan applications decreases on Saturday and becomes very less on Sunday.
This data is very useful for staff management as more staff is required on weekdays compared to weekends.

2.4.0.0. Feature Engineering: Two types of feature engineering are performed. They are performed on both application_train and application_test.

2.4.1.0. Three new ratios are added. These ratios are added based on the domain knowledge acquired during the 1st phase of this project. These ratios are considered by financial institutions for taking decisions pertaining to loans. These ratios help financial institutions to decide whether to give loans or not and how much loan to sanction. These ratios measure the loan repayment capability of applicants. They are:

2.4.1.1. Debt-to-Income Ratio - This is the ratio of loan annuity (AMT_ANNUITY) and income (AMT_INCOME_TOTAL) of the applicants.

2.4.1.2. Loan-to-Value Ratio - This is the ratio of loan amount (AMT_CREDIT) and price of the goods for which loan is given (AMT_GOODS_PRICE) to the applicants.

2.4.1.3. Loan-to-Income Ratio - This is the ratio of loan amount (AMT_CREDIT) and income (AMT_INCOME_TOTAL) of the applicants.

2.4.2.0. Features are merged from bureau and previous_application: Based on the dataset level analysis, it is decided to add features from bureau and previous_application to application_train and application_test. The columns of bureau which are common in application_train/application_test are renamed by appending _BUREAU to common column names of bureau. The columns of previous_application which are common in application_train/application_test are renamed by appending _PREVIOUS_APPLICATION to common column names of previous_application.

Left merge is performed for merging bureau and previos_application data with application_train and application_test. Merge is performed on column named SK_ID_CURR. SK_ID_BUREAU and SK_ID_PREV columns are dropped.

While merging numerical data, the mean of all the data corresponding to each unique SK_ID_CURR in each column is calculated and merged with application_train and application_test against corresponding SK_ID_CURR.

While merging categorical data, the categorical data is 1st one hot encoded and then the median of all the data corresponding to each unique SK_ID_CURR in each column is calculated and merged with application_train and application_test against corresponding SK_ID_CURR.

A track of columns originally having numerical values and categorical values is kept for further use. A target data is maintained which consists of values from the TARGET column of application_train.

At the end of this stage we find that there are 322 columns in application_train_final (prepared by merging bureau and previous_application data with application_train) and 321 columns in application_test_final (prepared by merging bureau and previous_application data with application_test). Application_train_final has an additional column because of the presence of the TARGET column.

Note: Only bureau and previous_application are merged with application_train and application_test mainly because of limited resources (RAM limitations encountered in Google Colaboratory). This also reduces complexity. However, with no dearth of resources, it is preferred to merge other data tables also. Finally, the most important features can be selected during feature selection.

2.5.0.0. Data Preparation: One hot encoding, imputation and standard scaling: In this section, data is prepared for mathematical operations.

2.5.1.0. One hot encoding: Before performing one hot encoding on remaining categorical columns, application_train_final and application_test_final are vertically concatenated. After one hot encoding is performed, slicing of data is done to obtain application_train_final_ohe and application_train_final_ohe. application_train_final_ohe is one hot encoded data of application_train merged with bureau and previous application. application_test_final_ohe is one hot encoded data of application_test merged with bureau and previous application.

2.5.2.0. Imputation: Median imputation is performed on numerical columns. These numerical columns are those which were numerical in the original datasets (application_train, application_test, bureau and previous).

2.5.3.0. Standard scaling: Standard scaling is performed on numerical columns. These numerical columns are those which were numerical in the original datasets (application_train, application_test, bureau and previous). Standard scaling is not performed on the columns resulting from one hot encoding.

2.5.4.0. X_train_final, X_validate_final and X_test_final are obtained as prepared data after performing one hot encoding, imputation and standard scaling. These are the data split from application_train. application_test_final_ohe_combined is obtained as prepared data corresponding to application_test. y_train, y_validate and y_test are target values corresponding to X_train_final, X_validate_final and X_test_final respectively. application_test_final_ohe_combined is also obtained which is prepared data corresponding to appliucation_test merged with bureau and previous_application.

2.5.5.0. All these data are saved as csv files with the same name for further use. These data are saved to create a restore point. Any further action can be performed by reading data from these csv files. This is also important as RAM issues are encountered in Google Colaboratory.

2.6.0.0. Outlier detection and removal

2.6.1.0. When a box plot for data under AMT_INCOME_TOTAL of X_train_final is plotted, it is observed that the box is not even visible and some data points are extending well beyond the whisker limits of the box plot. This indicates the presence of outliers and calls for outlier detection and subsequent removal.

2.6.2.0. Outlier detection is performed using Local Outlier Factor (LOF) based outlier detection module of pyod library. Pyod needs to be installed and updated in Google Colaboratory. Contamination is set at 0.05 which indicates that 5% of the total data points shall be detected as outliers. Total count of outliers and inliers is stored in variables named outliers and inliers respectively. A new dataframe named X_train_final_outlier is created and data from X_train_final is copied to it. A new column named outlier is added to X_train_final_outlier which indicates whether a data point is outlier or not. Target values are also horizontally concatenated with X_train_final_outlier and outlier removal is done based on the values in the column named outlier. X_train_final_outlier_removed and y_train_outlier_removed are the datasets obtained after removing outlier points.

2.6.3.0. Box plot is again plotted for data under AMT_INCOME_TOTAL of X_train_final_outlier_removed. The plot has improved significantly.

2.6.4.0. It is observed that the percentage of defaulters has not changed significantly after outlier removal. Percentage does not look significant but the actual number considering the volume of applicants may be significant. It can even become more significant if the loan amount involved is very high. Hence, it is always suggested to take the opinion of domain experts.

2.7.0.0. Feature selection

2.7.1.0. After the complete preparation of data, 444 columns/features are observed. This is a lot of features to deal with. It is quite possible that many of these features will not contribute towards model performance but increase the complexity. Hence, feature selection is performed.

2.7.2.0. XGBoost based and Gradient Boosting based feature selection are performed. Top 20 features with their feature importance are plotted in bar graph for both selection criteria.

2.7.3.0. Common features from top 225 features based on both selection methodology are selected. Thus 176 features are selected. It is observed that the ratios created during feature engineering figure among selected features.

2.7.4.0. Heat map is generated for 20 selected numerical features. It is observed that only a few features are strongly correlated to other/others. From this perspective also feature selection is correctly done.

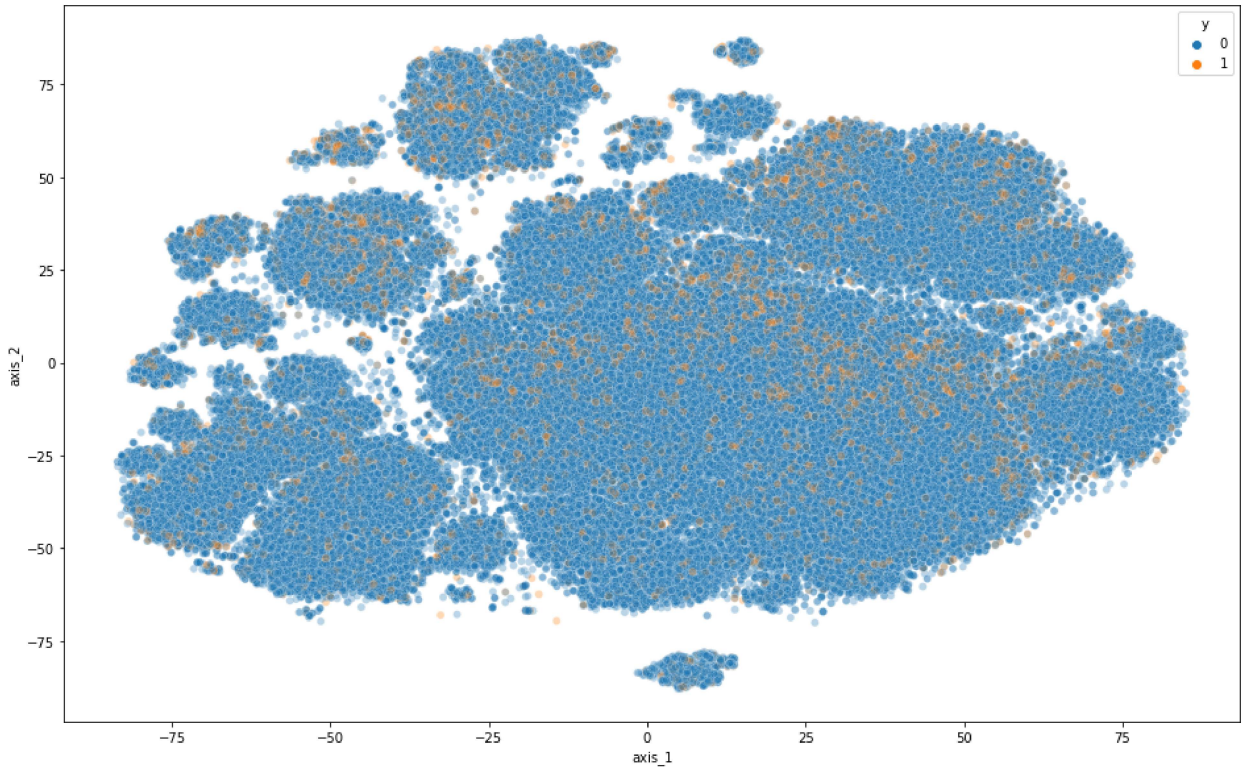
2.7.5.0. Based on these features selected, X_train_final_feature_selected, X_validate_final_feature_selected, X_test_final_feature_selected are defined each with 176 selected features. y_train_final_feature_selected, y_validate_final_feature_selected and y_test_final_feature_selected are corresponding datasets with target values. application_test_final_feature_selected is also defined with selected features.

2.7.6.0. All these datasets are saved as csv files with the same name. This is done to create a restore point so that further actions can be done by importing data from these csv files.

2.8.0.0. Perform TSNE

2.8.1.0. TSNE is performed with some values of perplexity and iterations. A typical render with 30 as perplexity and 1000 as iteration value takes 1 hour and 30 minutes on Google Colaboratory. This gives an indication of the scale of data we are dealing with.

2.8.2.0. Two plots - 1 with perplexity as 30 & 1000 iterations and other with perplexity as 50 and 1000 iterations - are retained in ipynb file. TSNE scatter plot with perplexity as 50 and 1000 iterations is reproduced here. The two axes are 2 dimensions representing all the features.



Some clusters are obtained but the 2 target values are mixed. Separability shall be further evaluated after model training. Different models shall be tried.

2.9.0.0. Conclusion

2.9.1.0. All the relevant files can be accessed through the following link: https://drive.google.com/drive/folders/1evFZRwFWh4zkR9CiT46llB9PlaXFLfLA?usp=sharing

2.9.2.0. This is a very important and lengthy phase as a lot of hits and trials need to be done. The datasets required to be fed to the machine learning model get prepared at this stage.

2.9.3.0. Useful mathematical and business insights are obtained in this phase. Visualisations obtained at this stage may be directly useful to managers.

2.9.4.0. Limitation of resources is a constraint at this stage as is clearly evinced by RAM limitation of Google Colaboratory. At this stage the importance of computing power is also understood.

2.9.5.0. It is important that while dealing with any machine learning project, presence or guidance of domain experts is highly valuable. Domain knowledge is especially required for feature engineering. It is also helpful in deciding whether to retain a column/parameter/feature with a lot of missing values. Domain knowledge is useful in deciding whether to do mean imputation or mean imputation.

2.9.6.0. In the section named 'Common commands' in the ipynb file, a function named df_size_optimizer() is used. This is very effective in optimising the size of dataframe obtained by importing data from csv files. The due credits for this function is given in the comment before the start of the function.

3.0.0.0. Modeling and Error Analysis

In phase 2 EDA was performed and data preparation was done. In this phase the data prepared in phase 2 shall be used for training machine learning models. Various machine learning models will be trained and results will be analyzed to find the best model. Best model will be further tuned to improve the result. Lastly, error analysis will be performed.

The ipynb file corresponding to this phase is divided into sections. This phase 3 documentation has to be read along with the ipynb file for correlation of different terminologies and outputs. The description given here follows the same section wise approach as the ipynb file. All the relevant files can be accessed through the following link: https://drive.google.com/drive/folders/1evFZRwFWh4zkR9CiT46llB9PlaXFLfLA?usp=sharing

3.1.0.0. Common commands: The following actions are performed in this section:

3.1.1.0. Google Drive is mounted for accessing data files.

3.1.2.0. Relevant packages are installed.

3.1.3.0. Relevant libraries are imported. It may happen that some packages and libraries are not used. However, they appear because they were used in the process of development of the final ipynb file.

3.1.4.0. One custom function for dataframe optimisation is defined. Data sets created in the previous phase are imported and their shapes are printed.

3.1.5.0. Data sets (both train and test) are quite unbalanced. So, upsampling is performed on both data sets with all features and dataset with selected features. This will be used for model training and to conclude any advantage is drawn from upsampling or not.

3.2.0.0. Model comparison:

3.2.1.0. Model comparison is performed using a package named Pycaret. Pycaret is an easy to use package giving options to create a pipeline for data preprocessing, train & compare model and deploy model. There are other features too such as PCA, hyperparameter tuning etc.

3.2.2.0. Three models namely logistic regression, random forest and light GBM are compared for four different conditions:

- Model comparison with selected features and without upsampling
- Model comparison with selected features and with upsampling
- Model comparison with all features and without upsampling
- Model comparison with all features and with upsampling.

Thus we are training 12 models.

3.2.3.0. Column names for the datasets are modified to suit the acceptance of pycaret. As preprocessed train and test data are already available, the same data sets are used in the setup of data.

3.2.4.0. Based on the results of comparison, the following are the observations:

- Upsampling does not give good results. Hence, this strategy will be dropped henceforth.
- Light GBM gives best results with feature selected data without upsampling. This model will be further tuned for best results.
- Feature selected data shall be used for model tuning and further training.

3.3.0.0. Tuning the best model:

3.3.1.0. Based on the observations of the previous section, lightgbm is tuned for feature selected data without upsampling. Tuning is performed with two conditions - Tuning while optimizing for accuracy, and Tuning while optimizing for AUC. Both conditions give the same accuracy and AUC for test data.

3.3.2.0. As the results for 2 conditions are the same, the model with AUC optimized is selected for further tasks. We refer to this model as best_model_auc.

3.3.3.0. It is observed that the accuracy of this tuned model is best among all the models trained so far. However, the AUC is slightly lower than the untuned lightgbm for data with features selected and without upsampling (saved in section 2.1 of phase 3 .ipynb file as best_model_feature_selected). A final comparison is done between tuned and untuned lightgbm models based upon the confusion matrix in the next section.

3.4.0.0. Comparison of confusion matrix and final selection of model:

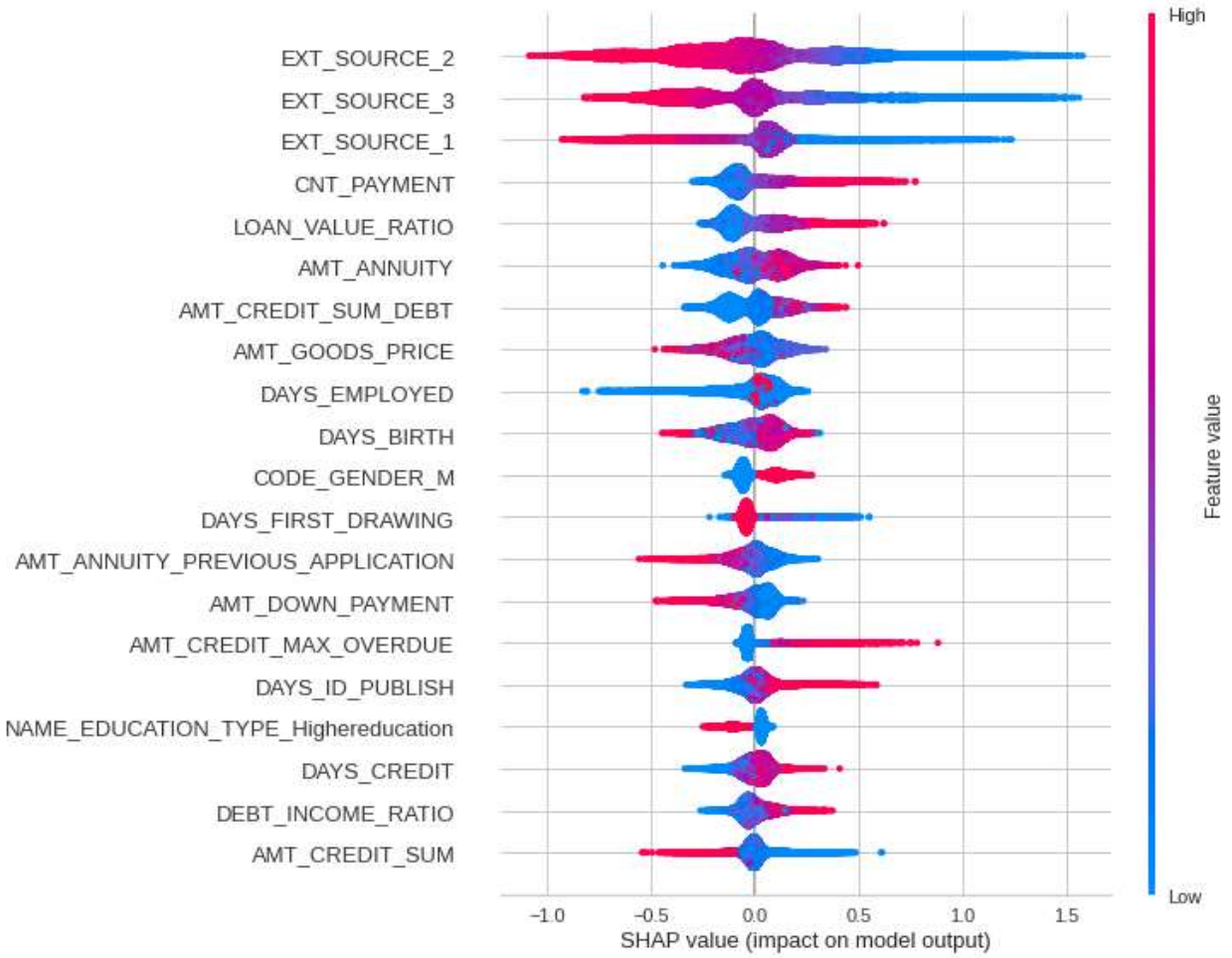
3.4.1.0. In this section confusion matrices are drawn for results obtained on test data using two models - untuned lightgbm and tuned lightgbm.

3.4.2.0. It is observed that the number of applicants who should not be given loan but are predicted as eligible for loan (i.e., the count of the 3rd quadrant) is lower in the tuned model. Hence, tuned ligbm is selected as the best model based on which error analysis will be performed in the next section. Here after this tuned lightgbm model will be named **best_model**.

3.5.0.0. Error Analysis

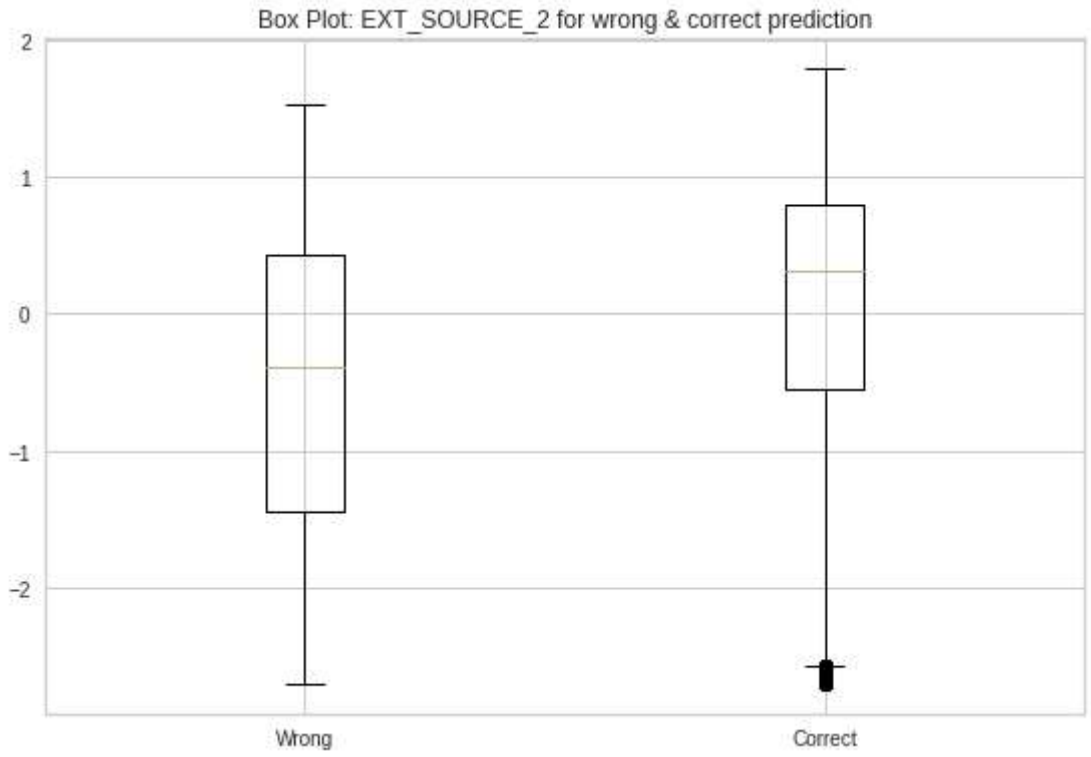
3.5.1.0. At the time of execution of this section of code, the saved tuned lightgbm model was throwing an error. Hence, lightgbm was again trained with tuned parameter values. Thus the model trained earlier and the one being trained currently will have the same weights and same results. Predictions are made on test data and the predicted values are saved for further use.

3.5.2.0. SHAP (from pycaret model interpretation) is used for getting the top 20 features. Top 3 features are EXT_SOUCE_2, EXT_SORCE_3 and EXT_SOURCE_1. SHAP interpretation from the ipynb file is reproduced here.

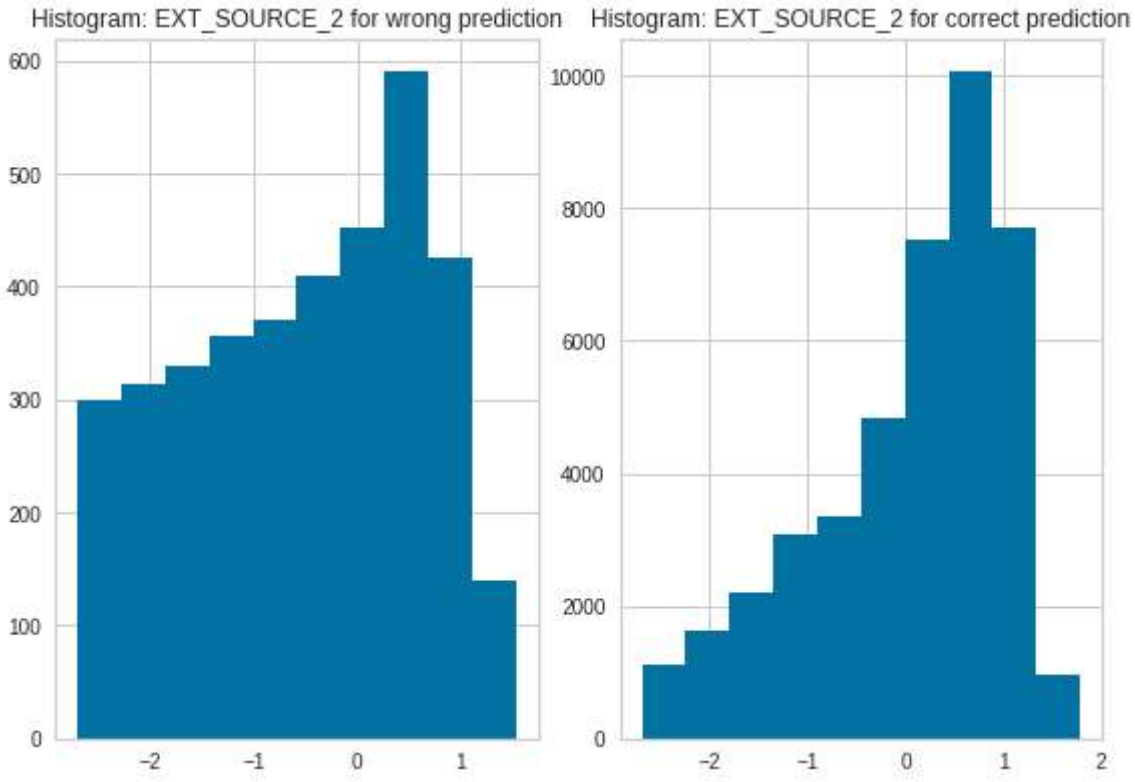


AUC is also plotted for best_model just for visualization. Lime is used for getting top 3 features for 10 wrongly predicted data points. Lime is also used for getting top 3 features for 10 correctly predicted data points. Thus it is observed that EXT_SOUCE_2, EXT_SORCE_3 and EXT_SOURCE_1 are the top 3 features for both wrongly predicted and correctly predicted data points.

Box plots and histograms are made for top 3 features for wrongly and correctly predicted points. Sample box plots and histograms from the ipynb file are reproduced here.



Sample Box plot



Sample Histogram

3.5.3.0. Observations:

- Top 3 features influencing wrongly and correctly predicted points are the same. They are EXT_SOUCE_2, EXT_SORCE_3 and EXT_SOURCE_1.
- Spread of data points for these 3 features for wrongly predicted data points is lower than correctly predicted data points.
- Statistically data is found in the same scale (Standard scaling was performed on data during phase 2) as observed in the box plot.
- From the histograms, it is observed that outliers are more pronounced for correctly predicted data points.
- It can be concluded that features are not having an impact on prediction as top 3 features for wrongly and correctly predicted data points are the same.
- Plots for the top features don't give any conclusive interpretation for predictions.
- It calls for introduction of more features through advanced feature engineering. Advanced feature engineering will be performed in the next section.

4.0.0.0. Advanced Modeling and Feature Engineering

In phase 3 the best model was found and further tuned. Error analysis was also done. Only classical non deep-learning models were used - Logistic regression, Random forest and Lightgbm. In this phase, PCA shall be performed on test data and further analysis of correctly and wrongly predicted points will be done. New features shall be added to the original dataset and models will be trained on new dataset. Deep learning models shall also be trained. Results from the new models shall be analyzed.

The ipynb file corresponding to this phase is divided into sections. This phase 4 documentation has to be read along with the ipynb file for correlation of different terminologies and outputs. The description given here follows the same section wise approach as the ipynb file. All the relevant files can be accessed through the following link: https://drive.google.com/drive/folders/1evFZRwFWh4zkR9CiT46llB9PlaXFLfLA?usp=sharing

4.1.0.0. Common commands: The following actions are performed in this section:

4.1.1.0. Google Drive is mounted for accessing data files.

4.1.2.0. Relevant packages are installed.

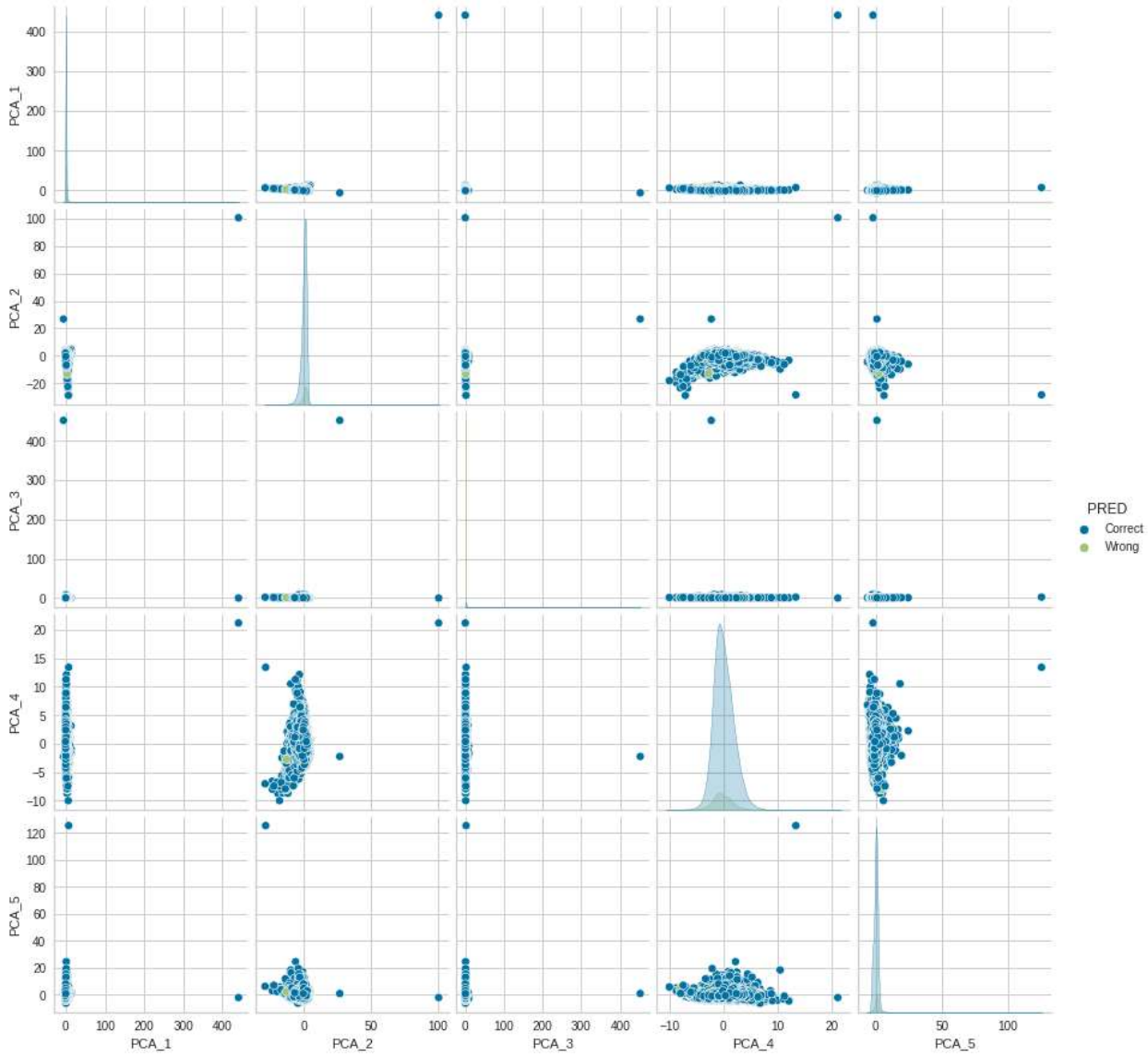
4.1.3.0. Relevant libraries are imported. It may happen that some packages and libraries are not used. However, they appear because they were used in the process of development of the final ipynb file.

4.1.4.0. One custom function for dataframe optimisation is defined. Data sets created in the previous phase are imported and their shapes are printed.

4.2.0.0. PCA:

4.2.1.0. Data Preparation - We use the test dataset with predictions saved in phase 3. Some columns are dropped from the dataframe as they cannot be used to perform PCA.

4.2.2.0. PCA is performed with 5 components i.e., PCA outcome has 5 features. A data frame is formed with these 5 features. Target, Label and Score columns are added to this dataframe. Two new columns - Pred and Type - are added. Pred column indicates whether the data point is correctly predicted or wrongly predicted. Type column indicates prediction with correct and wrong label e.g., Correct_0 means the prediction is 0 and it is correctly predicted. We make pairplots from the 5 PCA features. Pair plots from the ipynb file is reproduced here.



From the pair plots (which are scatter plots considering 2 features at a time), it is observed that the spread of wrongly predicted data is lower compared to spread for correctly predicted data.

4.2.3.0. We also perform PCA with 2 features and make a scatterplot. From the scatter plot it is observed that the spread of wrongly predicted data is lower compared to spread for correctly predicted data.

4.3.0.0. Train model with CONFIDENCE as output

4.3.1.0. Light GBM with tuned parameters (from Phase 3) is trained on feature selected data (X_train_feature_selected). [Please note that the best model was saved in phase 3. However, using the saved model was throwing an error. So, the model had to be trained again with the already available tuned parameter values.] Pycaret is used for this training as was done in Phase 3. For this data setup is done and column names

of input data are changed to suit the acceptability of Pycaret. This trained model is named best_model. Prediction is made on test data and thus we obtain a dataframe with test data and TARGET, Label and Score columns. A new column is added which indicates the prediction and confidence. For example, Correct_High indicates that the datapoint is correctly predicted with high confidence. Similarly data points are labeled as Correct_Low, Wrong_High and Wrong_Low. This new column is added based on the Score column. The score column indicates the probability of predicted Label. A cut off point is selected as defining low or high confidence. This cut off value is 0.75. If the score is less than or equal to 0.75, the confidence is low otherwise high. Thus a correctly predicted point with Score more than 0.75 is labeled Correct_High. Similarly other points are classified. This new test data frame is named predict_test.

Prediction is also made on train data and confidence column is added. This new data frame is named predict_train.

4.3.2.0. New data frames are defined as train and test data. These are X_predict_train and X_predict_test respectively. These are obtained by dropping TARGET, Label and Score columns from predict_train and predict_test respectively. Data set up is done for Pycaret. Here, the target (i.e., y values or dependent variable) is the CONFIDENCE column. Thus we are going to train our model with the same data (as was used in the previous step to train the best_model) with the difference that now the target values are those in the CONFIDENCE column.

4.3.3.0. Lightgbm is trained and the model is called M2. Predictions are made on train and test data. As there are 4 different classes in the target column (CONFIDENCE column), 4 different score columns are generated corresponding to each class. New data frames are created by adding score columns to the original data set (i.e., X_train_feature_selected_with_target and X_test_feature_selected_with_target). New data sets are named X_train_data3 and X_test_data3.

4.4.0.0. <u>LDA and new feature</u>

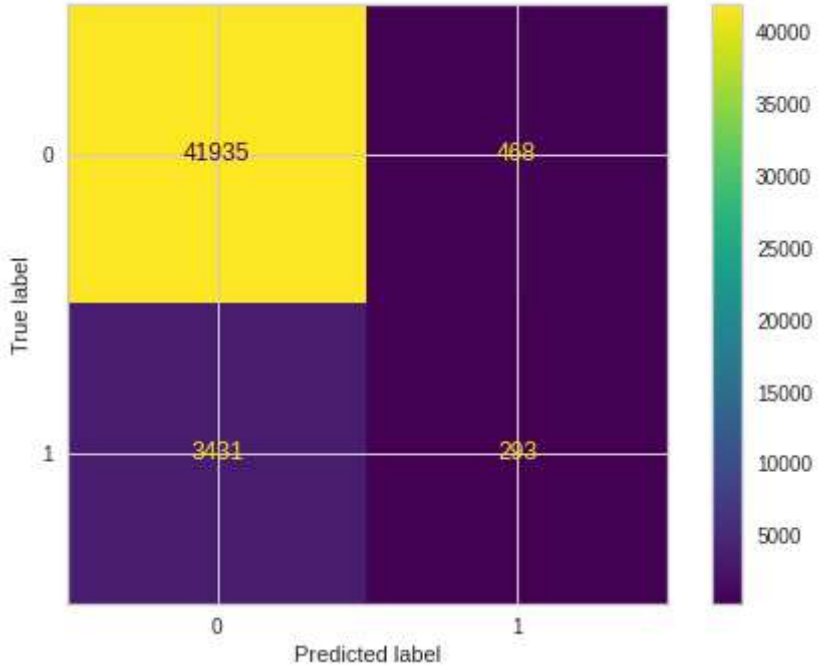
4.4.1.0. LDA is performed on X_train_data3 and X_test_data3. Based on LDA a new feature column named LDA is added to both train and test data (i.e., X_train_data3 and X_test_data3).

4.5.0.0. <u>Train model with new features</u>

4.5.1.0. LGBM: Lightgbm is trained on the dataset with added features (X_train_data3) and further tuned. This tuned model is called lgbm_added_features. Predictions are made on test data (X_test_data3) and corresponding accuracy, AUC and confusion matrix are obtained. Following are the observations when compared to best_model (which is the best tuned model on original train data i.e., X_train_feature_selected_with_target. best_model can be referred to in section 3.1 of this phase):

- Overfitting is observed with substantial difference in accuracy & AUC values between predictions on train data and test data.
- Although AUC has increased compared to best_model, accuracy has decreased.

Based on the above observations, it is concluded that lgbm_added_features is not better than best_model. Confusion matrix from the ipynb file is reproduced here as a sample.



4.5.2.0. Stacked Model: Stacked model is trained using pycaret - Logistic regression, random forest & lightgbm are used as estimators; and lightgbm is used as predictor. This model is called stacker. Predictions are made on test data (X_test_data3) and corresponding accuracy, AUC and confusion matrix are obtained.Following are the observations when compared to best_model:

- Overfitting is observed with substantial difference in accuracy & AUC values between predictions on train data and test data.
- Accuracy and AUC have decreased compared to best_model.

Based on the above observations, it is concluded that stacker is not better than best_model.

4.5.3.0. Neural Network: Neural network is trained. This model is called NN_model. Following are the observations when compared to best_model:

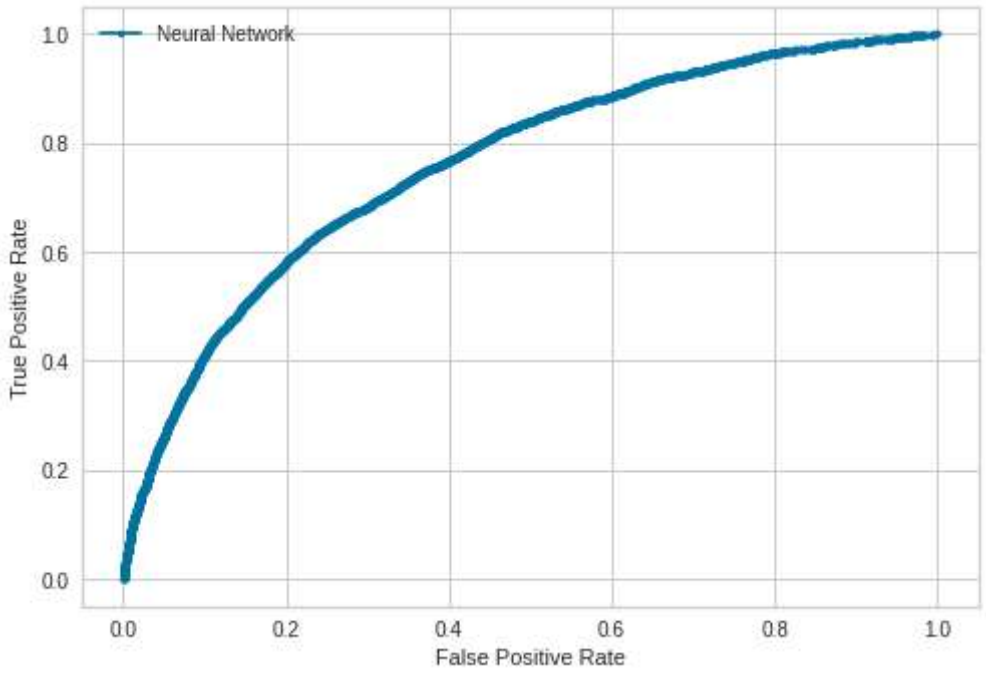
- Overfitting is observed although not major.
- Accuracy and AUC have decreased compared to best_model.

Based on the above observations, it is concluded that NN_model is not better than best_model.

4.5.4.0. Neural Network for original data: As a trial, Neural Network is trained on original data (data without additional features). This is done here as the Neural network was not tried for original data. Following are the observations when compared to best_model:

- Overfitting is not observed.
- Accuracy and AUC have decreased only slightly compared to best_model.
- The number of people who should have been rejected for loan but are predicted as eligible for loan is huge.

Based on the above observations, it is concluded that model_feature_selected is not better than best_model. AUC plot from ipynb file is reproduced here as a sample.



4.6.0.0. <u>Conclusion</u>

4.6.1.0. After trying so many models with so many different conditions (across phase 3 and phase 4), it is observed that tuned lightgbm is the best performer. Even deep learning models fare poorly as compared to tuned lightgbm. Also best results are obtained on feature selected data.

4.6.2.0. Many permutations and combinations are applied to reach this conclusion. However, more approaches can be tried which may or may not give better results. Such approaches can be tried with availability of time and resources.

4.6.3.0. This phase concludes the model training step. Further the trained model shall be deployed in the coming phase.

5.0.0.0. Deployment & Productionization

In this phase machine learning models shall be deployed for productionisation. The models trained in phase 3 and phase 4 shall not be used for deployment. Model shall be trained from scratch to maintain a coherence for pipeline creation. Further the relevant ipynb file shall be a single documentation of data preprocessing, model training and deployment. Till now trained models were tested on a small portion of data (obtained through train test split) from processed application_train. Now, predictions will be made on application_test.

The ipynb file corresponding to this phase is divided into sections. This phase 5 documentation has to be read along with the ipynb file for correlation of different terminologies and outputs. The description given here follows the same section wise approach as the ipynb file. All the relevant files can be accessed through the following link: https://drive.google.com/drive/folders/1evFZRwFWh4zkR9CiT46llB9PlaXFLfLA?usp=sharing

5.1.0.0. Common commands: The following actions are performed in this section:

5.1.1.0. Google Drive is mounted for accessing data files.

5.1.2.0. Relevant packages are installed.

5.1.3.0. Relevant libraries are imported. It may happen that some packages and libraries are not used. However, they appear because they were used in the process of development of the final ipynb file.

5.1.4.0. One custom function for dataframe optimisation is defined. Data sets created in the previous phase are imported and their shapes are printed.

5.2.0.0. Feature Engineering and data merger: Following actions are performed in this section:

5.2.1.0. Three new columns - DEBT_INCOME_RATIO, LOAN_VALUE_RATIO, LOAN_INCOME_RATIO - are added to application_train data. DEBT_INCOME_RATIO is the ratio of loan annuity (AMT_ANNUITY) and income (AMT_INCOME_TOTAL) of the applicants. LOAN_VALUE_RATIO is the ratio of loan amount (AMT_CREDIT) and price of the goods for which loan is given (AMT_GOODS_PRICE) to the applicants. LOAN_INCOME_RATIO is the ratio of loan amount (AMT_CREDIT) and income (AMT_INCOME_TOTAL) of the applicants.

5.2.2.0. Data from bureau and previous_application which bear the same SK_ID_CURR as those in application_train are merged to application_train.

5.2.3.0. SK_ID_CURR is dropped from the data obtained after merger. The data thus obtained is named as train_data and is saved for future use.

5.3.0.0. Training and Pipeline using Pycaret:

5.3.1.0. Lightgbm is trained and tuned for prediction using 100%, 50% and 25% data (train_data) obtained in the previous sub-section. Of these, training using 100% data and 25% data is retained on the ipynb notebook. For getting 50% and 25% data from train_data, train test split is used with stratification. Sklearn is used for this purpose. Trained model is saved for predictions.

5.3.2.0. Pipe line is defined using saved data, model and column names. A custom function named inference is defined which carries out all the transformations required in the pipeline for getting predictions. Predictions are made on application_test.

5.3.3.0. Conclusion:

- Pycaret is a very convenient tool for data pre-processing and model training. A one line code for data set-up does all the pre-processing. Further one line code can prepare a model and another line can tune it.
- Model created by Pycaret is huge in size. Model created using 100% train_data was 173MB in size. Predictions can be made in Google colaboratory using saved models. However, deployment on Heroku throws an error. So models were trained using 50% and 25% data thinking that error in deployment was due to size of the model. Error persisted with models prepared using 50% and 25% data.
- It was decided to switch to Sklearn and try to deploy a model created by using Sklearn. This is discussed in the next sub-section.

5.4.0.0. <u>Training and Pipeline using Sklearn:</u>

5.4.1.0. Sklearn is used for data preprocessing and training. Median imputation and standard scaling are performed on numerical features. Constant imputation and one hot encoding are performed on categorical data. For dealing with unseen categorical feature value(s), handle_unknown is set to ‘ignore’ for one hot encoding. This ignores the unseen category values and does not throw an error. The imputation, scaling and one hot encoding strategy are saved for use in the pipeline.

5.4.2.0. Outlier detection is performed using Local Outlier Factor (LOF) based outlier detection module of pyod library. Pyod needs to be installed and updated in Google Colaboratory which is performed in the common commands section. Contamination is set at 0.05 which indicates that 5% of the total data points shall be detected as outliers. Outliers are removed from the data.

5.4.3.0. Feature selection is done using GradientBoostingClassifier of Sklearn. Top 175 features are selected. The 175 selected features/columns are saved for future use.

5.4.4.0. GradientBoostingClassifier model is trained on feature selected data. Model is saved for performing predictions.

5.4.5.0. Pipe line is defined using saved data, model and column names. A custom function named inference is defined which carries out all the transformations required in the pipeline for getting predictions. Predictions are made on application_test.

5.4.6.0. Conclusion:

- Number of lines of code increases significantly when compared to Pycaret.
- Models created are significantly smaller in size compared to models created by Pycaret.
- Deployment on Heroku did not throw any error.

5.5.0.0. Deployment:

The source code for deployment can be accessed from https://github.com/Saurabha-Daa/test. These codes are not a part of ipynb notebook.

5.5.1.0. Deployment is performed using Heroku.

5.5.2.0. Github is used as a data repository and connected to Heroku for deployment.

5.5.3.0. Deployed web page can be accessed by the following url - https://deployment-0.herokuapp.com/.

5.5.4.0. The journey to final deployment saw failures and learnings as a result. Few major highlights in the road to successful deployment are mentioned below:

- Github puts a limitation on the size of files that can be uploaded. Through the web interface, files of size up to 25 MB can be uploaded. Using Git desktop, files upto 100 MB can be uploaded normally. For files more than 100 MB in size, Git LFS has to be used.
- Although Pycaret is user friendly, the model size created by it is huge and creates problems while hosting on Heroku and AWS. Currently support for Pycaret is also not that great.
- Sklearn is quite stable and did not create any problem in deployment. Number of lines of code is more for Sklearn compared to Pycaret.
- FastApi was tried for deployment. However, with some initial failures (more due to Pycaret), I switched to Streamlit. Streamlit is very simple to use and finds seamless integration with python. While FastApi or Flask requires some prior knowledge of web development, streamlit has no such requirement. Streamlit is most suited for machine learning engineers as deployment can be shown without getting into the complexities of web development. Although I have prior experience of working with web development, I decided to do the deployment without using html to show the power of Streamlit and to keep the code as simple as possible.
- Three cloud hosting platforms were tried. A comparison of these platforms can be found on the internet. However, a comparison from my perspective (a first time user) is given as under:

HEROKU	AZURE	AWS
Needs connection with Github.	Needs connection with Github.	Does not need connection with Github.
Easiest and simplest to set up.	Comparatively difficult and complex to set up.	Comparatively difficult and complex to set up.
Requires git.	Requires git.	Requires ssh and ftp.
Does not require a credit card.	Requires credit card.	Requires credit card.

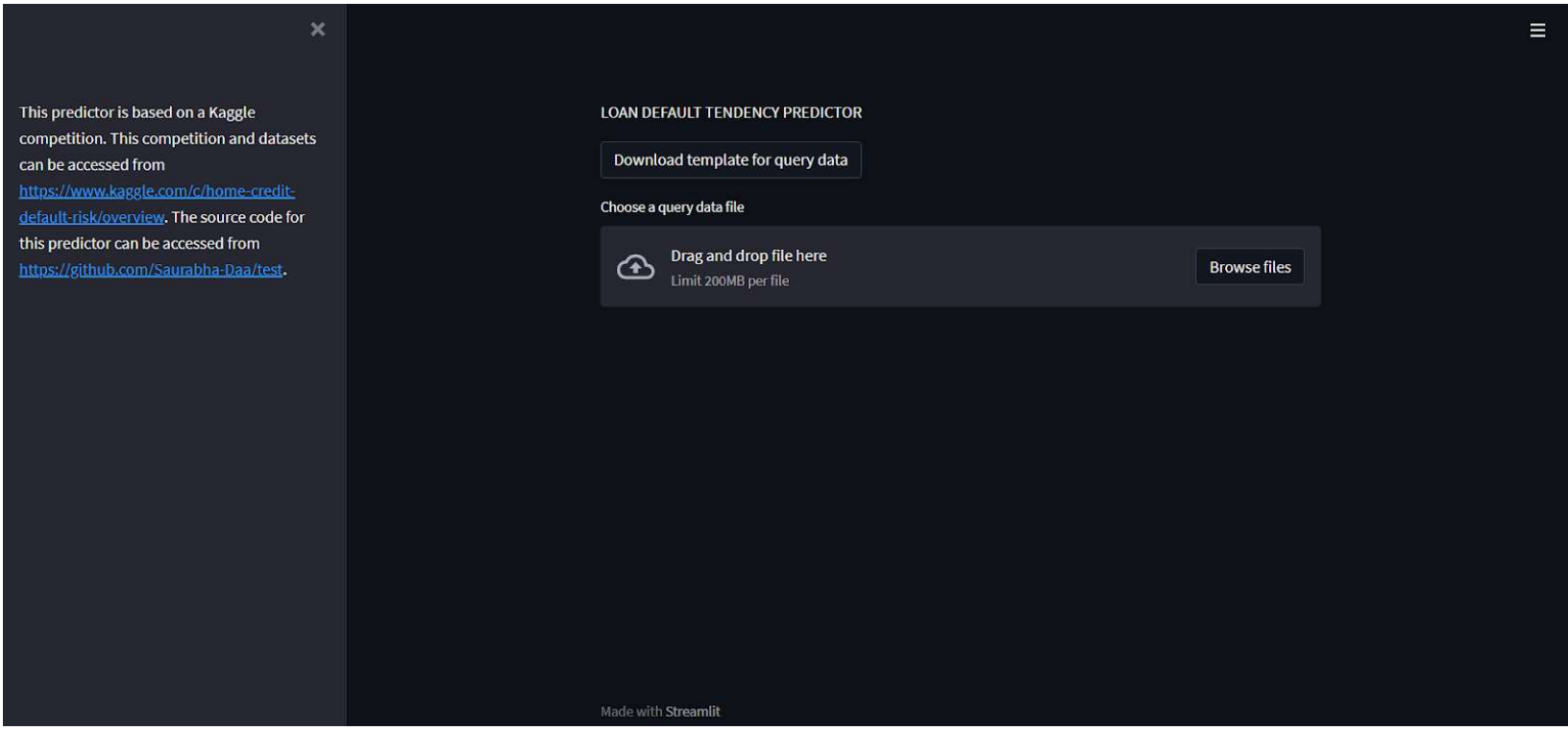
- Final deployment is done on Heroku.
- Even after deciding to use Streamlit and Heroku, I faced roadblocks while deployment. The code which was giving results on Google Colaboratory failed during deployment. Deployment was successful but I was not able to get the predictions. After reading the Heroku deployment logs, it was found that during prediction RAM consumption was exceeding allotment. Deployment code was optimized to reduce memory consumption and thus desired outcomes were achieved. Now, the successful deployment is able to handle 25 MB of application_test for predictions.

5.5.5.0. Let's talk about deployed web page.

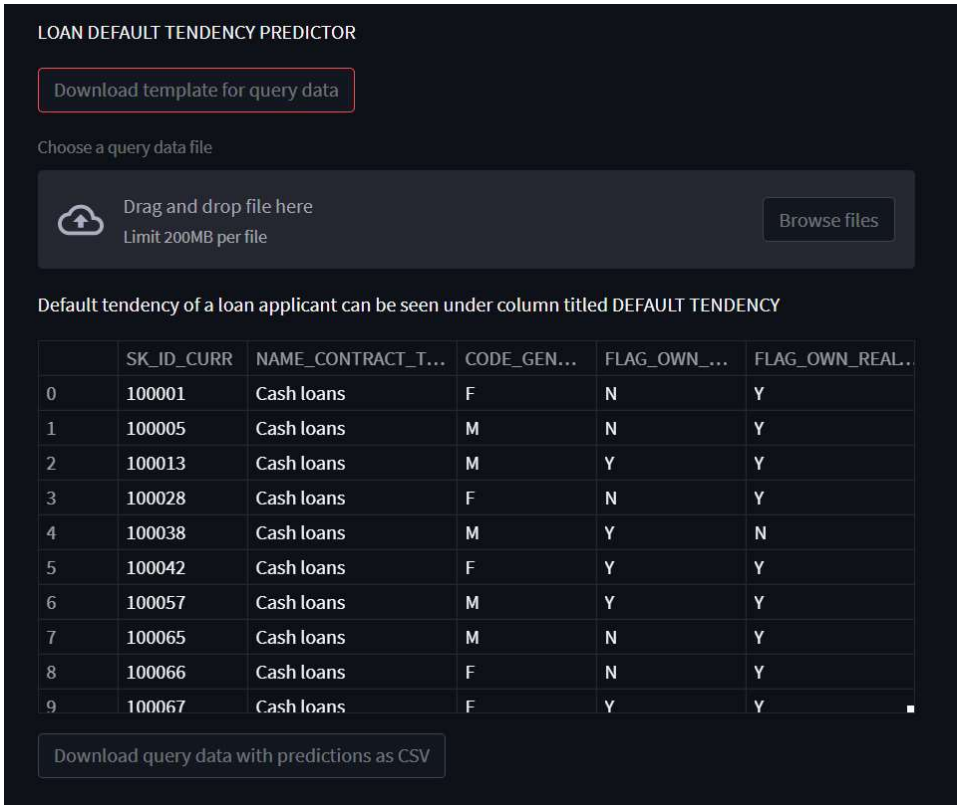
I have kept it as simple as possible and did not use any html code. A sidebar gives a crisp explanation of the task source (Kaggle competition) and source code (hosted on GitHub). Links to access the Kaggle competition and source code are also given. This sidebar can be hidden.

On the main page, an option to download the template for the query is also given. Next an upload option is provided to upload a csv file of query points. Once a query csv is uploaded, the result is shown on screen with all the columns from the query file and an additional column showing default tendency (This additional column is titled DEFAULT TENDENCY). Also an option to download a csv file with query points and predictions is available. Internally during prediction, following error handling are done:

- A check is placed in the code which checks whether the columns in the uploaded csv match the columns specified in the template.
- For dealing with unseen category value(s), handle_unknown is set to 'ignore'. This ignores the unseen category values and does not throw an error.



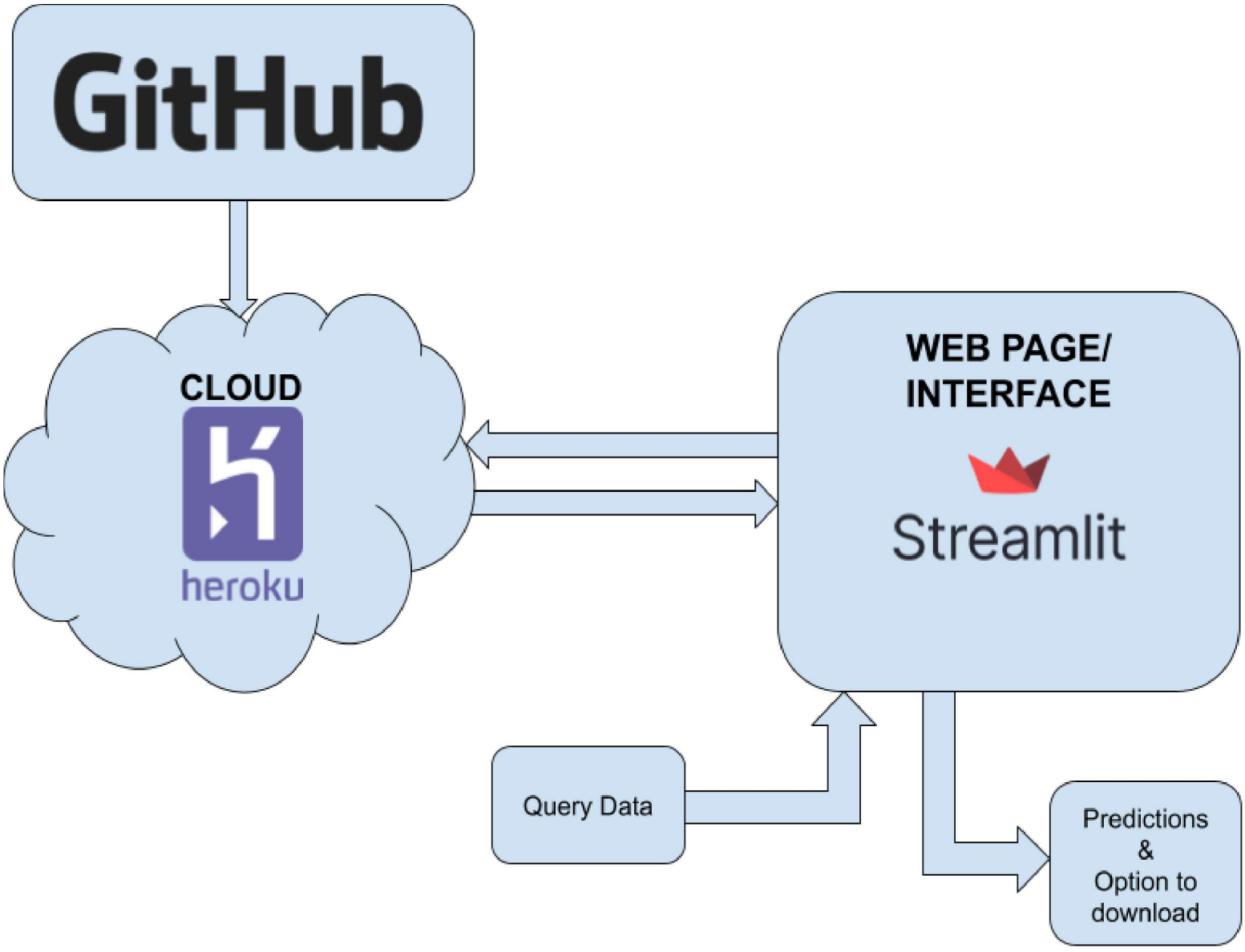
Page layout as it is rendered



	SK_ID_CURR	NAME_CONTRACT_T...	CODE_GEN...	FLAG_OWN_...	FLAG_OWN_REAL..
0	100001	Cash loans	F	N	Y
1	100005	Cash loans	M	N	Y
2	100013	Cash loans	M	Y	Y
3	100028	Cash loans	F	N	Y
4	100038	Cash loans	M	Y	N
5	100042	Cash loans	F	Y	Y
6	100057	Cash loans	M	Y	Y
7	100065	Cash loans	M	N	Y
8	100066	Cash loans	F	N	Y
9	100067	Cash loans	F	Y	Y

Predictions on web page

5.5.6.0. System Architecture: System architecture is as shown in the block diagram given below. GitHub acts as a repository for code. Heroku is the cloud platform for deployment. The data flow between Heroku and GitHub is one directional. Web page is created using the Streamlit API. Data flow between Web page and Cloud is two directional. Query data is passed to the web page to get predictions and option to download predictions.



5.5.8.0. The current deployment has been checked by inputting 48744 data points at a time. The deployment performed as intended and predictions were made within 30 seconds. Thus the deployment is highly scalable.

The quantum of data that we used for testing is not expected frequently, most of the time not even in a day. Also the input is not expected to be real time. Input will be used for predictions in batches. So, latency and throughput are not a major requirement.

5.5.9.0. Following are the limitations and scope of improvement:

- The limitations of the system is mainly due to the memory quota given by Heroku. The memory quota is 500 MB. So, in the due course of successful deployment issues were faced due to memory quota limitation. The issues were resolved by code optimization. Thus, this is not a code issue rather it is an issue pertaining to memory limitation.
- Errors pertaining to use of Pycaret can be resolved and the system can be tested again.
- Deployment can be done on AWS & Azure and latency can be tested.

6.0.0.0. Resources

6.1.0.0. Important links

6.1.1.0. The project is based on a Kaggle competition which can be accessed from https://www.kaggle.com/c/home-credit-default-risk/data.

6.1.2.0. The app deployed to predict default tendency of a loan applicant can be accessed from https://deployment-0.herokuapp.com/.

6.1.3.0. The codes and relevant csv and pickle files for the deployed app can be accessed from https://github.com/Saurabha-Daa/test. The README.md file provides instructions for app deployment. It is reproduced here.

- Description:
  This app is created to predict default tendency of a loan applicant. This app is based on a Kaggle competition which can be viewed at https://www.kaggle.com/c/home-credit-default-risk/data.
- How to run the code:
  Run app.py using streamlit. Please note that app1.py also exists. However, app1.py does not need to be run.
- How to use the rendered page:
  Template to enter query data can be downloaded from the rendered web page.
  Data in the given format can be uploaded using the file uploader option on the page.
  Once the file gets uploaded, predictions are displayed on the web page. Also a download button appears which can be clicked to download the csv file which consists of query data set with an additional 'DEFAULT TENDENCY' column.
  If the uploaded file is not in the format of the given template, an error is displayed which says 'Query columns do not match the columns of required format as given in template. Please upload query data in the given format'.
  Sidebar contains links to Github page (for codes) and Kaggle competition page (for task description and data).

6.1.4.0. The ipynb files with their pdf copies and consolidated project report can be downloaded from https://github.com/Saurabha-Daa/uoh-aiml-project-resources.

6.2.0.0. Libraries used

6.2.1.0. Major libraries used in this project are:

- Google.colab
- Pandas
- Numpy
- Pyod
- Matplotlib
- Sklearn
- Pycaret

- Keras
- Lime
- Shap
- Re
- Collections
- Imblearn
- Pickle