University of Hyderabad

Predicting the capability of each applicant in repaying a loan

Thesis submitted in fulfilment of Diploma in AIML

by

Narasimha N. Shenoy

nanorohan@gmail.com

Contents

Chapter 1

INTRODUCTION

1.1.0.0. **Motivation**

1.1.1.0. Over the course of several years of evolution, humankind has devised various tools and systems to ensure their continued survival and to enhance quality of life. One such system that was conceived, widely adopted and has stood the test of time is the concept of money. Money helps people achieve a better quality of education, larger chance of business success, access to medical facilities and higher work output. In anyone's life, a situation may come when all of sudden you require cash. In such times of crisis, not everyone can readily arrange for funds from friends, relatives or such means.

1.1.2.0. To address this need, the concept of loan originated. Loans are an important means to tide over difficult times, aim for upward mobility and in the development of individuals and industries alike. With such profound socio-economic considerations, judicious accessibility to loans as well as mutual benefit to lenders as well as recipients are aspects that need to be ensured with a high degree of integrity. Majorly, the onus for ensuring an objective and mutually beneficial lending process lies with the lender. Any mechanism to aid the lender in this process will help them sustain and become or stay profitable and also enable greater disbursal of loans to applicants deemed eligible.

1.2.0.0. **Problem At Hand**

1.2.1.0. Given a loan application of a potential or existing client at Home Credit, the ML model needs to predict whether the client will be able to repay the loan or not. Access to past data for a sample of Home Credit applicants aid in the building of this prediction model.

1.3.0.0. **What is the problem all about?**

1.3.1.0. Increasing population coupled with modernization and consequently, human’s quest for lifestyle enhancements have resulted in a huge demand for credit or loans. There is intense competition among traditional banks and numerous credit-lending start-ups to grab their share of this business and attract people by providing different attractive schemes. Start-ups especially, are targeting and catering to unbanked population or first-time credit seekers who have insufficient or non-existent credit histories due to which they are at disadvantage with traditional financial institutions. This unprecedented accessibility in credit availability, market competition and consumption has led to an increase in losses resulting from bad loans. Instead of making money from loan interest, lenders are suffering a huge capital loss. In order to prevent the loss, it is very important to have a system in place which will accurately predict the loan defaulters even before approving the loan. This is especially important for institutions with lending as their primary source of business; such as Home Credit, in order to sustain and grow in the market.

1.4.0.0. **Why is this an important problem to solve?**

1.4.1.0. Post-pandemic world has disrupted many aspects of life including financial requirements. Many are resorting to loans to ensure basic subsistence. Some struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, such a population is

often taken advantage of by unscrupulous lenders. Secondly, lending institutions need to ensure very low credit delinquency rates to stay profitable and provide loans to worthy applicants. An objective model helps the lending agencies disburse prudent loans. This system helps them process and disburse loans faster, increase profit and client base as well as possibly protect genuine debtors from predatory lenders.

1.5.0.0. **Business/Real-world impact of solving this problem.**

1.5.1.0. Lender's context - Data-driven, objective decision regarding credit-delinquency ensures a lending process which is swift, has a rational basis and safeguards the agencies' interests while maximising profits. Correlating the metrics of model evaluation to the KPIs of the organisation helps in understanding their standing in the market, potential client base and formulation of future strategies. Going beyond mere binary classification, a model predicting a loan amount threshold which can be offered to the applicant ensuring low default score can help lending-only agencies, especially new companies to gain new clients and grants them the opportunity for greater engagement and interpersonal interactions, a base for future business.

1.5.2.0. Debtor's context - A model that does not rely extensively on past credit histories allows for financial inclusion of people with insufficient or non-existent credit histories. Such groups, which are vulnerable to exploitative lending practices, can be catered to by the organised sectors. Loans from organised lenders will in turn improve credit worthiness of this population helping them gain loans in the future, empowering them, financially and socially.

Chapter 2

FAMILIARISING WITH THE HOME CREDIT DATASET

2.1.0.0. **Understanding the dataset & its quirks**

2.1.1.0. The dataset is sourced from Home credit Default Risk Kaggle competition.

2.1.1.1. The dataset totalling to approximately 2.68GB is in csv format split over 9 files.

2.1.1.2. For ease of understanding, the entire dataset can be grouped under three major heads.

2.1.2.0. *The main dataset*

2.1.2.1. This is the primary data for training the model and testing its performance. It has two files, '*application_train.csv*' and '*application_test.csv*'.

2.1.2.2. The training set contains 307,511 observations of 122 variables and provides static data for all applicants of Home Credit services. The target variable resides in this dataset and indicates whether clients have had difficulties in meeting payment with two values - 1 for clients who defaulted and 0 for those that did not default. We may consider this as the default dataset. Each observation is a loan application and includes the target value, demographic variables and some other information.

2.1.2.3. The test set contains 48745 entries with 121 variables as it being the test data, target values column shall not be present.

2.1.3.0. *'Applicants having existing history with Home Credit' dataset*

2.1.3.1. This data is pertaining to applicants who are already existing clients of Home Credit

2.1.3.2. The file '*previous_application.csv'* contains records of all previous loan parameters and client information for Home Credit loans of clients who have loans in the sample.

2.1.3.3. The file '*POS_CASH_balance.csv'* details monthly balance snapshots of previous PoS (point of sales) and cash loans that the applicant has had with Home Credit.

2.1.3.4. The *file 'installments_payments.csv'* has repayment history for the previously disbursed loans in Home Credit related to the loans in the sample.

2.1.3.5. The file '*credit_card_balance.csv'* contains monthly balance records of previous credit card loans that the applicant has had with Home Credit.

2.1.4.0. *'Applicants having no history with Home Credit' dataset*

2.1.4.1. This data is pertaining to applicants who have no prior history with Home Credit.

2.1.4.2. The file '*bureau.csv'* contains data pertaining to previous loans a client had secured from other financial institutions, the details of which were reported to the Credit Bureau (for clients who have a loan in the sample).

2.1.4.3. The file '*bureau_balance.csv'* details the monthly balances of the client's previous credits reported to the Credit Bureau.

2.1.5.0. The Home Credit dataset is fairly large sized [~2.6GB] with 121 columns or features in the primary (train) dataset to tinker with. Moreover, the secondary data provides scope for creation of new features, feature interactions and combinations to help the model.

2.1.6.0. As is the case with any profitable lending agency, the Home Credit training dataset has a very small number [approximately 8% of total samples] of credit defaulters. This is the case of an imbalanced dataset which needs to be addressed by either sampling techniques or using appropriate models and metrics.

2.1.7.0. Standard pre-processing and data cleaning procedures like missing value imputation, removal of duplicate entries, detection and handling of glaring anomalies/outliers are especially important in this dataset in order to have relevant features as inputs and produce interpretable results.

2.2.0.0. **Listing approaches towards handling the datasets**

2.2.1.0. As the dataset is primarily a csv file, using Pandas is preferred owing to its capabilities in handling tabular data such as this dataset; and phenomenal documentation and support in case of possible bugs. In the case of encountering memory problems or sluggishness, Vaex or Dask may be used as an alternative.

2.2.2.0. For EDA and visualizations, Seaborn shall be used for mapping the data on the informative and interactive plots and deriving visual insights.

2.2.3.0. Sklearn shall be used for building machine learning and statistical models such as clustering, classification, regression, etc. Depending on the outcome of EDA and feature engineering, more specific libraries may be tried out.

2.2.4.0. For working out initial strategies regarding metrics, features, popular and powerful libraries such as Numpy and SciPy shall be used.

2.2.5.0. More specific libraries may be used depending on the outcome of EDA and feature-specific cases and the same shall be highlighted at the appropriate stages.

2.2.6.0. *Possibility of Dataset augmentation*

2.2.6.1. The Home Credit dataset from Kaggle competition is used in accordance with Kaggle's Data Access and Use policy. The dataset is created with Home Credit's existing credit delinquency model parameters in context.

2.2.6.2. Open-source data from other sources may not be used directly as both the datasets may have been created with different philosophies and hence, different parameters. However, after feature engineering there is a possibility of using other datasets with similar features, employing suitable imputation or encodings.

Chapter 3

PERFORMANCE METRICS FOR OUR MODEL

3.1.0.0. **Need for a performance metric**

3.1.1.0. The 'learning' of any machine learning model is basically on a constructive feedback principle. We build a model, get feedback from metrics, make improvements and continue until we achieve a desirable value of the chosen metric.

3.1.2.0. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model outcomes

3.1.3.0. *Prerequisite for choosing a performance metric - Balance between Domain knowledge and Machine Learning expertise*

3.1.3.1. Sound knowledge of the KPIs in the credit sector as well as the correlation between the ML model metrics and the KPIs is essential in order to not merely use the model as a binary segregator, but as a quantifiable means to monitor the organisation's health. Moreover, the ML model should be interpretable and be able to achieve the intent.

3.1.3.2. As humans are filling the loan application form fields and there is possibility for error or falsification, a domain knowledge of financial industry and understanding of the demographics helps in addressing this aspect. Also, depending on the interactions with a potential client seeking a substantial sum, knowledge and experience in the financial domain might motivate investing time and efforts to convert the application which might not be fully implementable in a binary model.

3.1.3.3. ML expertise can help understand and tackle the quirks related to the problem at hand such as imbalanced dataset, missing values or additional data needed. Suitability of models with context to accuracy vs. interpretability also requires knowledge of ML. It always helps to have an understanding of the inner logic of the model and its parameters today to assess whether the same is valid tomorrow.

3.1.3.4. Finding the right balance between business utility, interpretability and fidelity, robustness of ML model is an important constraint which needs to be resolved before dwelling further.

3.2.0.0. **The prominent KPIs in the credit lending sector**

3.2.1.0. There are some common KPIs frequently followed in lending agencies to assess their healthiness, processes and growth and formulation of business strategy. A brief regarding these is as follows:

3.2.1.1. Pull Through Rate - This KPI measures process efficiency by dividing total funded loans by the number of applications submitted during a defined period.

3.2.1.2. Abandoned loan rate - This KPI measures the percentage of loan applications that are abandoned by a borrower after they have been approved by the lender.

3.2.1.3. Application approval rate - This KPI is calculated by dividing the number of approved applications by the amount of submitted applications.

3.2.1.4. Customer Acquisition Cost - This key financial measurement is the ratio of a borrower's lifetime value to a borrower's acquisition cost. This KPI is used by lenders to help determine how much of its resources can be profitably spent on a particular customer.

The costs include but aren't limited to research, marketing and advertising. Ideally, the customer acquisition cost should be greater than one since a borrower isn't profitable if the cost to acquire is greater than the profit they will bring to a lender.

3.2.2.0. These business KPIs can be correlated with the model evaluation metrics to have a quantifiable, rational, data-based evaluation criteria. To elaborate, KPIs like abandoned loan rate as well as application approval rate can be directly arrived at by the input data for the model and the predictions. Other KPIs such as customer acquisition cost perhaps may need additional data to make sense.

3.3.0.0. **Setting the context for deciding metrics appropriate for case at hand**

3.3.1.0. The Home Credit dataset has the target equal to 0 for clients who repay the loan on time and target equal to 1 for those that default. So, this is a two state or binary classification problem.

3.3.2.0. The data is quite imbalanced because there is a high number of clients who repay the loan compared to clients who default.

3.3.3.0. While translating the model prediction to business outcome for Home Credit, there are two cases which result in a situation of loss.

3.3.3.1. Case 1 - The model has predicted the client will repay the loan but actually he has defaulted. This is critical as it results in loss of capital equivalent to defaulted credit to Home Credit.

3.3.3.2. Case 2 - The model has predicted the client will default but he can actually repay the loan back. Here, Home Credit faces the loss of a potential client and potential loss in return interest or lost business opportunity cost. Apart from this, there exists the fact that a deserving client is not getting a loan on account of the model prediction.

3.3.4.0. As the primary intent of using the model is to protect the interests of the credit lending agency, case 1 shall be a focal point in deciding the performance metrics.

3.3.5.0. Listing and assessing the possible Metrics appropriate for the context

3.3.5.1. Since the problem at hand is a binary classification problem, the following metrics are insightful -

★ Accuracy ★ Precision ★ Recall ★ F1 Score ★ ROC-AUC score

3.3.6.0. Evaluation of the suitability of these metrics to the Home Credit model is as below:

3.3.6.1. Accuracy is the most intuitive performance metric and it is simply the ratio of correctly predicted observation to the total observations. However, considering the imbalanced dataset, even a dumb model predicting every client as non-defaulter can bag an accuracy score of ~0.92 considering the training sample distribution, which is quite erroneous. Unless the imbalance is resolved, accuracy is a poor metric to use.

3.3.6.2. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations, by the model. Its interpretation is fairly simple and intuitive. In context to the Home Credit dataset, this translates to ratio of correctly identified defaulters to the sum total of correctly and incorrectly identified defaulters, by the model. Precision is a good metric when the cost of a false positive is high. This metric shall address the

case 2 scenario. As case 2 is not the dominant loss case, this may not be the primary metric.

3.3.6.3. Recall is the ratio of correctly predicted positive observations to the total positive observations, by the model. Its interpretation is fairly simple and intuitive. In context to the Home Credit dataset, this translates to the ratio of correctly identified defaulters to the actual defaulters, by the model. Recall is a good metric when the cost of a false negative is high. This metric shall address the case 1 scenario. As case 1 is the dominant loss case, this may be the primary metric.

3.3.6.4. F1 Score is basically the geometric mean of Precision and Recall. This score takes both false positives and false negatives into account. Though F1 Score is usually more useful than accuracy, especially for imbalanced distributions, intuitively it is not as easy to interpret and understand as accuracy. Though this metric addresses both the loss cases, to translate into business impact it has to be viewed along with the constituent Precision and Recall scores. This can be considered as a secondary metric.

3.3.6.5. ROC-AUC Score is basically the area under the curve for a plot of True Positive Rate [TPR] or Recall on Y-axis vs False Positive Rate [FPR] on X-axis. An excellent model scores closer to 1 implying it has a good measure of separability. A poor model scores near 0 which describes that it has the worst measure of separability. In fact, it means it is reciprocating the result and predicting 0s as 1s and 1s as 0s. When an AUC is 0.5, it means the model has no class separation capacity present whatsoever and is basically a random model. AUC score covers both loss cases and the graph has very good interpretability. Hence ROC-AUC Score can be the primary performance metric. Also, the original Kaggle Home Credit competition had this as the evaluation criterion.

3.3.6.6. PR Curve is basically the plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the ROC curve. Similar to the AUC of ROC curves, AUC-PR is typically in the range [0.5,1]. If a classifier obtains an AUC-PR smaller than 0.5, the labels should be inverted. But importantly, the Precision-Recall Plot is more informative than the ROC Plot when evaluating Binary Classifiers on Imbalanced Datasets, especially when one cares more about positive than negative class. The Home Credit fits the criteria as the dataset is highly imbalanced and as explained in the cases, predicting the defaulter (positive case) is the priority. Hence, the PRC graph shall also be plotted.

3.3.7.0. Cases where the above-mentioned Metrics shine

3.3.7.1. When the dataset is almost balanced, Accuracy is the one of the most widely used metrics owing to its high interpretability. A classic example of the balanced dataset is the Iris dataset.

3.3.7.2. Precision is best used when the cost of false positives is high as in the case of weather prediction for launching satellites. If the model predicts that it is a good day, but it is actually a bad day to launch the satellite (false positive) then the satellites may be destroyed and the cost of damages will be in the billions.

3.3.7.3. Recall is best used when the cost of false negatives is high as in the case of cancer detection. A false positive can be detected by further specialised tests but a false negative can be lethal by preventing timely diagnosis and thus, treatment.

3.4.0.0. **<u>Common approaches frequented for similar problems</u>**

3.4.1.0. Upon reading blog posts, resource forums and research papers pertaining to modelling ML solutions to similar problems, the logic can generally be categorised in any of the three broad strategies -

3.4.1.1. Extensive feature engineering, creating new features, interactions and using these inputs on fairly standard classification models such as LR, DT and its variants and ensemble models; with relatively lesser emphasis on model hyperparameter optimization.

3.4.1.2. Standard or minimal feature engineering and using many standard models with intensive hyperparameter tuning, usage of neural networks. Here, the majority of thought work is focused on optimising the model parameters.

3.4.1.3. A very few works have actually done both, extensive feature engineering as well as trying out a variety of models, each with hyperparameter tuning. Some have tried out DL models with varying results.

3.4.1.4. All of these approaches are in a way relevant to the problem at hand. Initial strategy shall be to understand the data with EDA and also try out the standard classification models. Eventually, after dwelling further, model-specific approaches can be formulated.

Chapter 4

EXPLORATORY DATA ANALYSIS

4.1.0.0. **EDA - What is it about?**

4.1.1.0. Exploratory Data Analysis [EDA] is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the actual modelling task.

4.1.2.0. EDA is primarily making sense of data at hand, before getting them dirty with it.

4.1.3.0. Occam's razor, which summarizes that of two competing theories, the simpler explanation of an entity is to be preferred; forms the basis for feature analysis.

4.1.4.0. Feature engineering refers to a process of selecting and transforming variables when creating a predictive model using machine learning or statistical modelling. The process involves a combination of data analysis, applying rules of thumb and domain-based knowledge.

4.1.5.0. *Objectives of performing EDA and Feature Analysis on the Home Credit dataset*

4.1.5.1. Understanding the given data sets individually and interactions among them.

4.1.5.2. Gaining insights regarding the features by variable-level visualizations and their relations with the 'Target' outcome.

4.1.5.3. Listing missing values and devising a strategy for filling these out logically.

4.1.5.4. Identifying outliers and rationally addressing them.

4.2.0.0. **EDA on the dataset using Colab notebook**

4.2.1.0. The following are the major phases of the EDA as carried out in the Colab notebook -

4.2.2.0. Section 1.0.0 - A brief summary of the project, dataset and intent of this notebook.

4.2.3.0. Section 2.0.0 - Contains the necessary groundwork for carrying out the EDA.

- Section 2.1.0 comprises the code for installing dependencies for some of the specialised libraries used in this notebook.
- Section 2.2.0 has the list of all the libraries loaded for the purpose of EDA in this notebook along with a context for each library.
- Section 2.3.0 lists the custom functions defined for the utilities which shall be frequently used or a feature which is important for the purpose of EDA.
- Section 2.4.0 loads the available datasets for EDA and further analysis.

4.2.4.0. Section 3.0.0 - Comprises of dataset-level analysis

- Section 3.1.0 presents a comprehensive summary for each dataset loaded. This helps form an action plan regarding EDA and downstream feature-engineering strategies.
- Section 3.2.0 gives an elaborate quantification of interactions between the main datasets for getting context of applicants in the sample population.
- Section 3.3.0 summarizes the key insights of the dataset-level summary.

4.2.5.0. Section 4.0.0 - consists of feature-level univariate and multivariate analysis.

- Section 4.1.0 contains visualizations for features which 'appear' to be important for defaulter prediction based on domain knowledge or common sense. Each visualization is followed by a summary for the same and some observations.
- Section 4.2.0 has visualizations for interactions among the features or bi-variate feature plots. These help in viewing the relation between the features involved coupled with the TARGET variable and derive key insights.
- Section 4.3.0 has correlation heatmaps among the features of each dataset along with the TARGET variable.
- Purpose for doing this is to get a fair idea of the degree of correlation among features which will give some basis for selecting features for modelling or assessing features selected in case of model-based feature selection [done in section 9.0.0].
- Each heatmap set is followed by insights derived from the same.

4.2.6.0. Section 5.0.0 - consists of feature engineering.
- Section 5.1.0 has some extracted features as well as the description for the same. Creation of these features is based on acquired domain knowledge and literature review.
- Section 5.2.0 comprises the process of merging the main datasets. Rationale for merging the datasets is explained as is the process followed in the same.
- Section 5.3.0 deals with encoding the categorical features and imputation of missing values. Strategy for the same is described in brief.
- Section 5.4.0 involves splitting the processed dataset into train, validation and test dataset.
- Based on future modeling insights, the validation set may be merged into the train dataset itself, opting for k-fold CV.

4.2.7.0. Section 6.0.0 - is a checkpoint where datasets processed thus far are saved or 'pickled' for further analysis, saving on RAM consumption.
- This is done as running the notebook on Colab free edition hits the RAM limit, even after using optimised datasets with context to size.

4.2.8.0. Section 7.0.0 - loads the saved files from section 6.0.0 for further processing.
- This is done in order to save on RAM consumption.

4.2.9.0. Section 8.0.0 - deals with removal of outliers.
- Presence of outliers was determined during EDA and the same shall be addressed in this section.
- Section 8.1.0 involves usage of the pyOD library to detect and remove the outliers.
- Section 8.2.0 involves verifying whether the outlier removal was effective or not.

4.2.10.0. Section 9.0.0 - comprises feature selection.
- As there are many features in the processed dataset and it is quite likely that some may very negligibly - or not at all, contribute towards defaulting tendency prediction, a sort of selection of useful features needs to be done.
- In this notebook, presently 2 models for feature selection are run and output of one of them is selected for further engineering.
- In future phases, more complex combinations of feature selection strategies shall be applied based on modeling and outcomes.
- Section 9.1.0 is about Extra tree Classifier from SKLearn for feature selection.

- Output of this is considered for further processing in the notebook.
- Section 9.2.0 is about Random Forest Regressor, again from SKLearn, for feature selection.
- Almost all the features thought of as important towards Defaulter prediction during EDA do figure in the top selected features, as do the created or extracted features as listed in section 5.1.0
- Section 9.3.0 has a correlation matrix heatmap for the top 25 features, just as a sanity check and for visualization.

4.2.11.0. Section 10.0.0 is about High-level data visualization.
- Section 10.1.0 and Section 10.2.0 deal with saving and loading the 'pickled' datasets, due to RAM constraints on Colab.
- Section 10.3.0 is about the actual tSNE visualization for getting a 'feel' of the data separability.
- Owing to the significant compute times for tSNE, only 2 combinations of parameters are evaluated and visualized.

4.2.12.0. Section 11.0.0 concludes this phase by summarizing key takeaways of the EDA and Feature Analysis.
- Section 11.1.0 lists the general highlights of the EDA phase.
- Section 11.2.0 summarizes the context-specific quirks and insights.

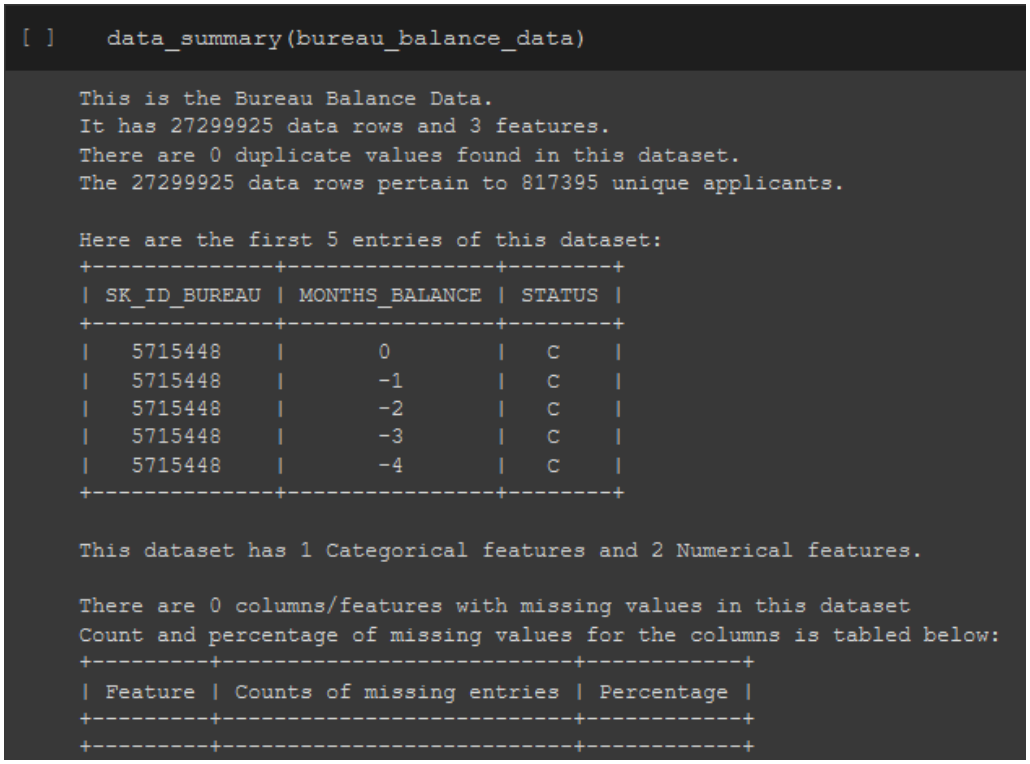
4.3.0.0. **Summary of EDA & Feature Analysis on the Home Credit dataset**

4.3.1.0. Dataset level analysis

4.3.1.1. Objective of the dataset-level analysis is understanding each dataset in context of types of data/features it contains, total data points and their uniqueness, proportion of missing values and finally, interaction among the main datasets.

4.3.1.2. All the datasets are loaded as Pandas DataFrame and a high-level summary is tabulated for each.

4.3.1.3. A screengrab of the summary tabulation for one of the datasets [bureau balance dataset] is shown for representation.



4.3.1.4. The summary for each dataset includes listing total and unique number of entries, nature of the features, tabulation of features with missing values and a dataslice for visualizing the dataset.

4.3.2.0. <u>Highlights of interactions among the main datasets is summarized below:</u>

4.3.2.1. Of the 307511 unique applicants in application_train dataset, 263491 [85.69% of the total applicants] have at least one recorded instance of past non-Home Credit history in the Bureau dataset. Alternately, of the 305811 unique applicants in Bureau dataset, 42320 [13.84% of the total unique entries] are exclusive to the Bureau and have no presence in the application_train dataset.

4.3.2.2. Of the 48744 unique applicants in application_test dataset, 42320 [86.82% of the total applicants] have at least one recorded instance of past non-Home Credit history in the Bureau dataset. Alternately, of the 305811 unique applicants in Bureau dataset, 263491 [86.16% of the total unique entries] are exclusive to the Bureau and have no presence in the application_test dataset.

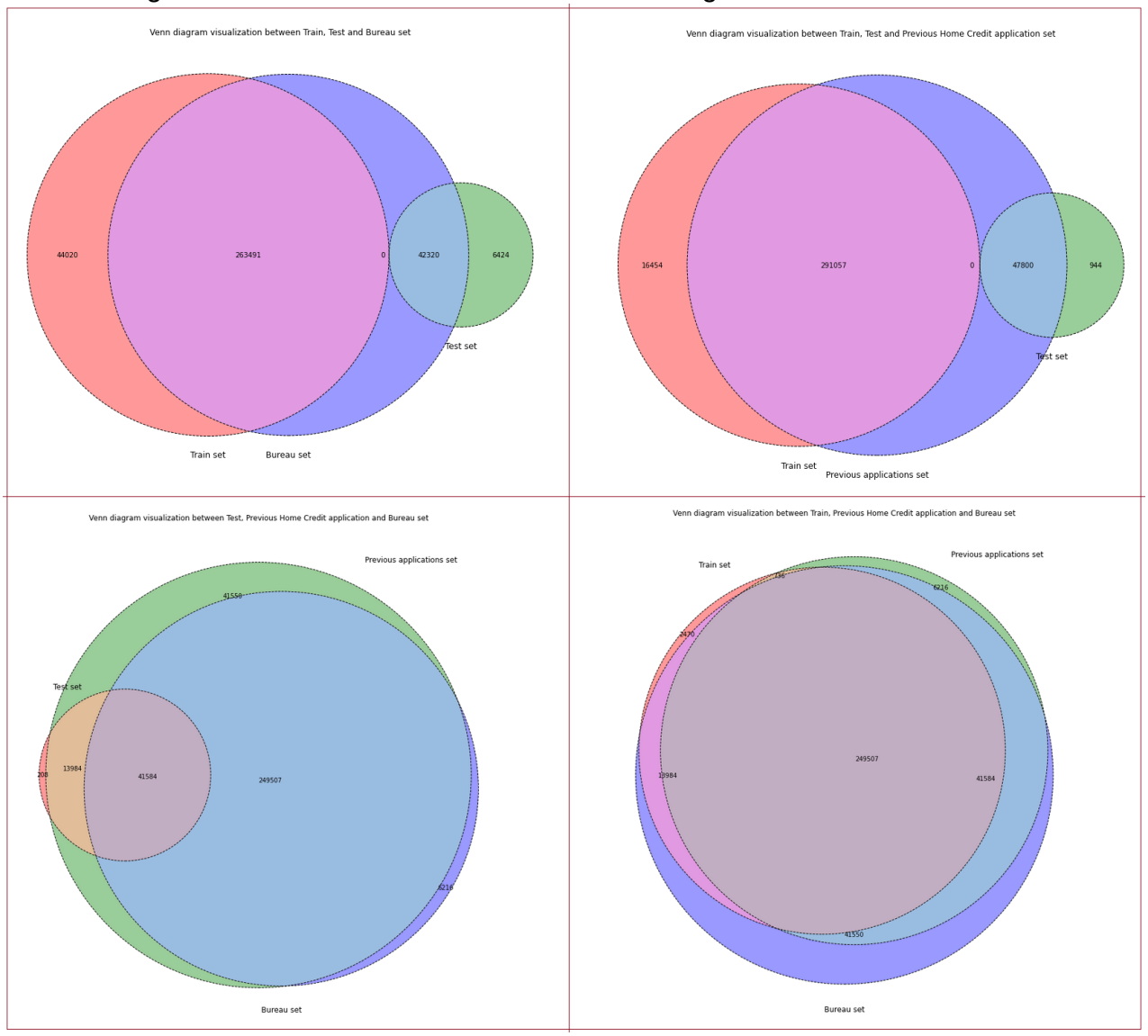
4.3.2.3. Of the 307511 unique applicants in application_train dataset, 291057 [94.65% of the total applicants] have at least one recorded instance of past Home Credit history in the previous applications dataset. Alternately, of the 338857 unique applicants in the previous applications dataset, 47800 [14.11% of the total unique entries] are exclusive to the previous applications and have no presence in the application_train dataset.

4.3.2.4. Of the 48744 unique applicants in application_test dataset, 47800 [98.06% of the total applicants] have at least one recorded instance of past Home Credit history in the previous applications dataset. Alternately, of the 338857 unique applicants in the previous applications dataset, 291057 [85.89% of the total unique entries] are exclusive to the previous applications and have no presence in the application_test dataset.

4.3.2.5. Of the 305811 unique applicants in bureau dataset, 291091 [95.19% of the total applicants] have at least one recorded instance of past Home Credit history in the previous applications dataset. Alternately, of the 338857 unique applicants in previous applications dataset, 47766 [14.1% of the total unique entries] are exclusive to the Home Credit previous applications and have no credit history with other agencies recorded in the bureau dataset.

4.3.2.6. Of the 305811 unique applicants in bureau dataset, 291091 [95.19% of the total applicants] have at least one recorded instance of past Home Credit history in the previous applications dataset. Alternately, of the 338857 unique applicants in previous applications dataset, 47766 [14.1% of the total unique entries] are exclusive to the Home Credit previous applications and have no credit history with other agencies recorded in the bureau dataset.

4.3.2.7. Venn Diagram visualization for the interactions among the main datasets



4.3.3.0. <u>Key insights from dataset-level EDA and analysis</u>

4.3.3.1. Around 86% of the training sample applicants are not first time credit seekers and have some credit history with lending agencies apart from Home Credit as recorded in the Bureau dataset.

4.3.3.2. Around 95% of the training sample applicants already have some credit history with Home Credit recorded in the previous applications dataset. Such a high number of repeat applicants might be indicative of customer's preference to Home Credit's lending processes and products over competition.

4.3.3.3. Barely 1% of the training sample applicants are first-time credit seekers from Home Credit with no recorded financial history in any agency.

4.3.3.4. As there is a very high number of applicants having previous loan records in the Bureau or Previous Home Credit database, these shall be used along with the Application Train dataset for modeling purposes.

4.3.3.5. There are many features/columns with missing values across the datasets with some having over 60% data missing. A suitable imputation strategy shall be employed unless it is evidenced by further analysis that dropping these features is a better strategy.

4.3.4.0. Feature-level univariate & multivariate analysis

4.3.4.1. Objective of the feature-level analysis is understanding each feature in context to its distribution, relation with the output or key variable, the values it takes, and possible anomalies.

4.3.4.2. To this effect, grouped bar charts, pie charts, box plots and histograms are plotted as per suitability with the type of feature under analysis.

4.3.4.3. As the whole objective of this exercise is predicting a potential defaulter, all the features shall mostly be plotted with the ‘Target’ variable as criterion.

4.3.4.4. Secondly, as there are 122 features in the Application Train dataset alone, visualizing each and every feature and deriving meaningful insights can be pretty time-consuming.

4.3.4.5. Hence, based on literature reviews and consequent domain knowledge coupled with practical intuition, features which ‘may have’ significant bearing on the defaulter prediction shall be visualized.

4.3.4.6. Commencing the EDA with the distribution of defaulters in the Application Train dataset [feature - ‘TARGET’], it is observed that the application_train dataset is heavily imbalanced, as expected for a healthy lending company. This fact shall govern the major decisions such as model evaluation metrics.

4.3.4.7. Visualizing the gender-wise distribution [feature - CODE_GENDER], it can be observed that women secured a greater number of loans as compared to men, almost twice as much.

4.3.4.8. Moreover, the credit default rate is slightly lower for women than for men.

4.3.4.9. These demographic insights can help Home Credit formulate focused products and campaigns catering to females as well as introspect the disparity in genders of applicants.

4.3.4.10. There are 4 entries where Gender='XNA'. Defaulting tendency for this category is 0. Since this is not providing much information to be retained as a separate representative category, these entries may be dropped eventually unless significant insights prove contrary.

4.3.4.11. Plotting the graphs for type of loans availed [feature - NAME_CONTRACT_TYPE], it is evident that a vast majority of the applicant sample population have availed cash loans over revolving loans.

4.3.4.12. The number of defaulters for revolving loan type is a tad little lower than cash loans and may be explored by Home Credit for in-depth assessment. With context to defaulter prediction considering sample the rates are not too different and hence, loan type does not highlight a quirk.

4.3.4.13. The gender-wise split is also expected, given the ratio of female-to-male applicants.

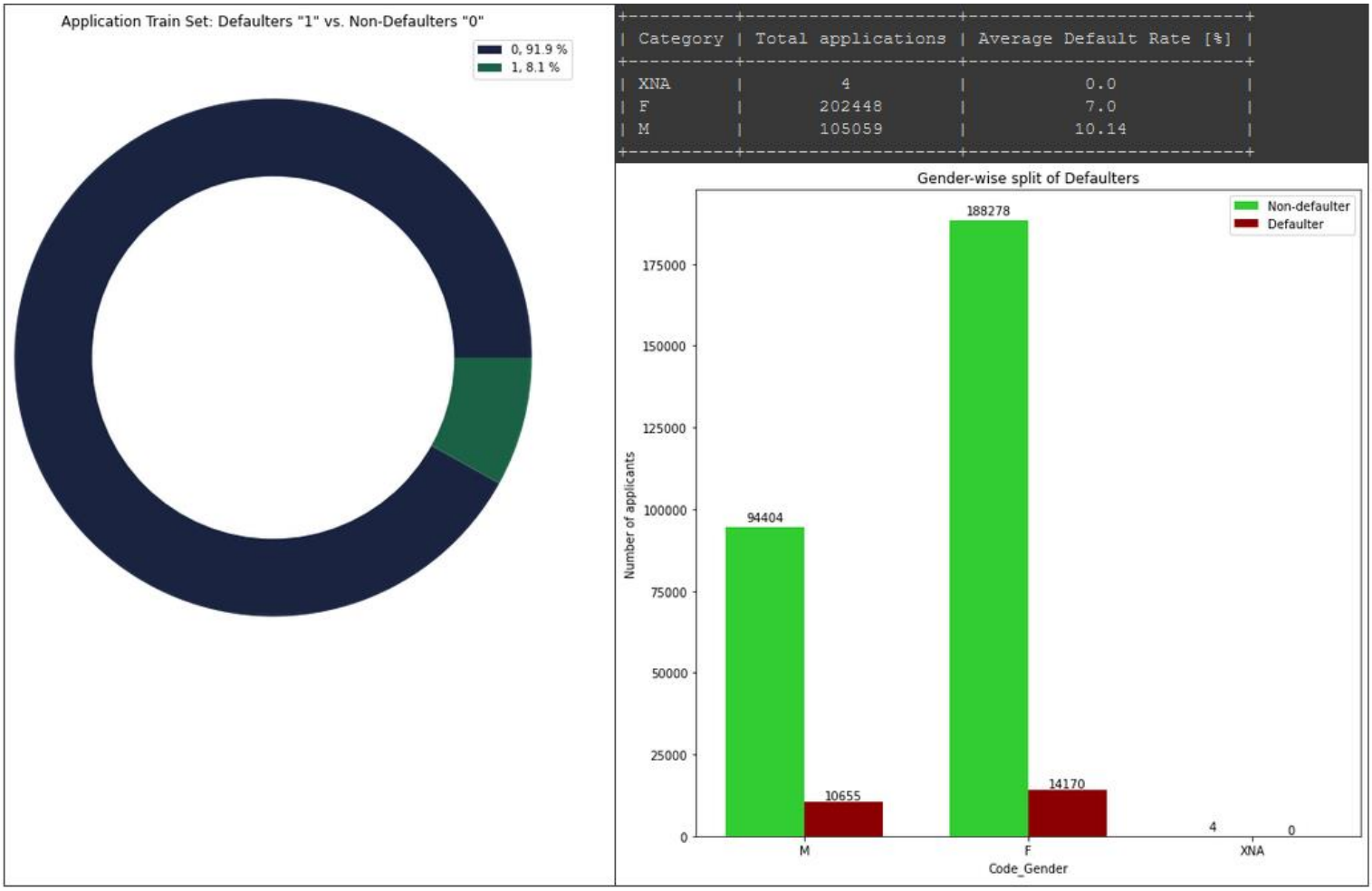


Image - Sample visualizations for context

4.3.4.14. From the 6 familial infographics [features - CNT_CHILDREN & NAME_FAMILY_STATUS], following are the insights -

4.3.4.15. A majority of applicants are married as well as having no offspring, indicative of a young demographic. However, this does not translate to a pattern for defaulter rate.

4.3.4.16. There is a significant variability in defaulter rate among other classes [ex. - High number of offspring or unknown marital status]. However, the data points are far too few to make any sort of meaningful inference or generalization.

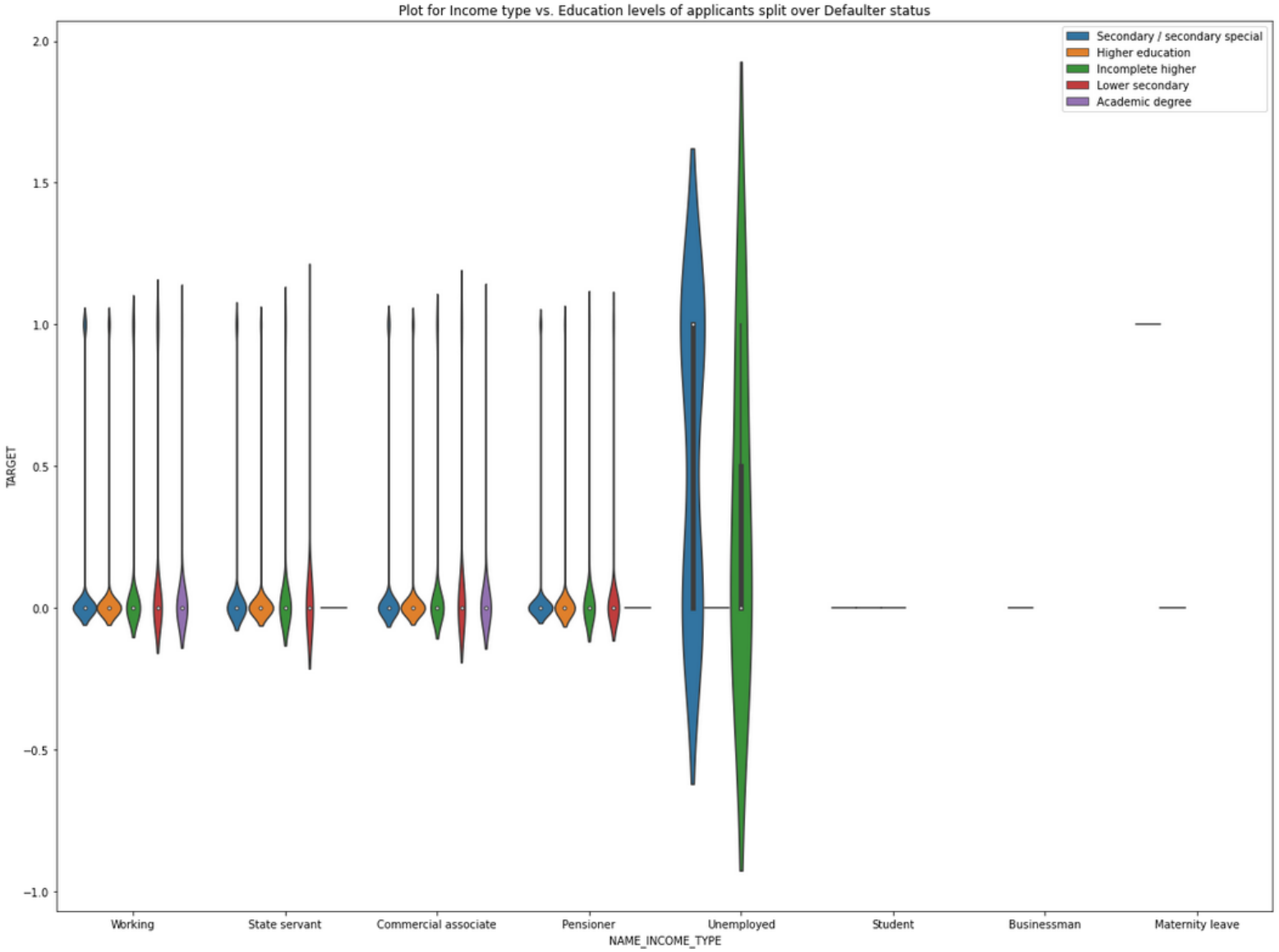
4.3.4.17. Relatively greater number of applicants do not own a car than those that do [feature - FLAG_OWN_CAR].

4.3.4.18. However, the defaulter rate is almost the same for both the cases and does not indicate a unique pattern.

4.3.4.19. A majority of the applicants are owning flats or some form of real estate [feature - FLAG_OWN_REALTY]. This can give an insight into the primary client-base patronising Home Credit.
4.3.4.20. However, there is no defaulter-wise aberration or pattern observed with context to realty ownership.
4.3.4.21. The statistics regarding car and realty ownership can provide Home Credit some insights regarding general wealth levels of their client base though as a feature for defaulter prediction, these statistics may not be too profound.

4.3.4.22. A majority of applicants have not provided their occupation type in the application [approx. 31.3%] [feature - OCCUPATION_TYPE].
4.3.4.23. Low-skill labourers, Drivers & Waiters have a relatively greater defaulter rate than other occupations.
4.3.4.24. Relatively high-skill applicants such as High-skill tech staff, HR staff, Core staff, Accountants and IT staff have a relatively lower defaulter rate than other occupations. Though this can be attributable to a very limited sample population, considering otherwise, this occupation demographic may be offered special incentives to avail products by Home Credit, depending on Home Credit's business goals and values [values is emphasised on as - the primary goal of Home Credit for predicting defaulters is to ensure that first time credit seekers as well as marginalised borrowers are given equal opportunities and occupation-wise promotion may be contrasting to their guiding spirit.]
4.3.4.25. These insights may help in understanding the borrowing patterns and possibly wilful defaulters in conjunction with income data.
4.3.4.26. Since there is a chance of this feature being significant to defaulter prediction, filling in the missing values is an important consideration.
4.3.4.27. Also, it would serve Home Credit well to record this data for future clients with due diligence as it 'may' affect their loan approval status.

4.3.4.28. Majority of applicants have attained secondary education [feature - NAME_EDUCATION_TYPE].
4.3.4.29. Cursory glance at defaulter-rate-per-education level suggests an inverse relation. This is more of a social insight.
4.3.4.30. To elaborate, defaulter rate is high among applicants with secondary education and this can be attributed to the vast majority in the sample population.
4.3.4.31. Among the other levels, as mentioned earlier, defaulter rate lowers substantially with increasing education.
4.3.4.32.
4.3.4.33. 'Working' income category applicants avail the greatest number of loans whereas Commercial Associates, Pensioners and State Servants take considerably lesser number of loans [feature - NAME_INCOME_TYPE].
4.3.4.34. Unemployed applicants and those on maternity leave have a very high default rate whereas Students & Businessmen have no defaults. However, considering the available data points' extremely limited representation, there can be no generalization possible.
4.3.4.35. A bivariate violin plot gives a visual insight regarding correlation between education, income source & defaulter status.



4.3.5.0. <u>Other key insights in a nutshell</u>

4.3.5.1. The External Source Normalized scores show different natures for the different defaulter states and may prove to be an important feature.

4.3.5.2. The graph for loan amount split over defaulter status is almost similar for both the defaulter classes, which suggests that defaulter tendency is independent of loan amount.

4.3.5.3. Loan amount and Annuity are directly proportional to each other which is logical. If the loan amount is high, the annuity amount for the same will also be high. However, the defaulters are split almost uniformly over the entire space which makes logistic regression'esque binary classification almost useless.

4.3.5.4. It is observed that the Default tendency for those who do provide work phone numbers is more than those who do not. This can be attributed to the fact that the wilful defaulters might be providing their work phone numbers so that they do not get disturbed on their personal mobile phone.

4.3.5.5. The bivariate graph for employment in years vs. loan amount sanctioned indicates weird values for days/years of employment. Hence this is also a case for outlier detection. Upon plotting with sanitised values, one can see defaulters are somewhat concentrated towards the lower left side indicating lower employment as well as lower loan amounts.

4.4.0.0. **Feature Engineering and Transformations**

4.4.1.0. Based on the domain-specific literature reviews and the features available, a few indicators of financial health or default tendency can be created.

4.4.2.0. The following are the additional features created -

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

4.4.2.2. ***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.

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

4.4.3.0. As was evidenced in the Venn diagram visualization, Bureau and Previous Application datasets also have valuable records worth investigation and same are merged with training dataset for further analysis and modeling.

4.4.4.0. However, an alternative approach may be explored eventually as Home Credit also aspires to cater to first time credit-seekers or marginalised populace and the model should reflect this thought process. Bureau and previous application data shall be virtually non-existent for such applicants.

4.4.5.0. ***Filling-in missing values and transformation***

4.4.5.1. Categorical features are ‘one-hot encoded’ using Pandas’ get dummies operator with the methods for handling NaN as a category.

4.4.5.2. Missing values in numerical features are filled using the median in order to mitigate the effects of outlier values.

4.4.6.0. ***Outlier detection and handling***

4.4.6.1. While performing the feature analysis on AMT_INCOME_TOTAL, the histogram was heavily distorted.

4.4.6.2. Generating the boxenplot, it is observed that there are some extreme income levels which are skewing the distribution.

4.4.6.3. Investigating further, there is a female applicant with a very high income level who is also a defaulter. Analysing dataset further, it is observed that loan amount is almost lying in the mid-levels which 'might' be indicative of an error in recording income levels rather than a wilful defaulter.

4.4.6.4. Considering this logic, there is a case for outlier removal.

4.4.6.5. Outlier detection is performed using the Cluster-Based Local Outlier Factor (CBLOF) scheme of the outlier detection module of pyOD library.

4.4.6.6. After specifying the parameters and carrying out the outlier removal, the dataset is checked for its split with context to class [TARGET] imbalance and it is found to be almost unchanged which is a good thing.

4.4.6.7. Moreover, plotting the boxenplot on cleansed data shows the effectiveness of the removal process as the data is much more legible as seen by the shape.

4.4.7.0. ***Feature selection***

4.4.7.1. After the processing of data up to this point, 444 features are present in the train dataset. As many of the features may not contribute at all towards outcome prediction or even to

a varying degree, it serves one well to weed out those superfluous features as is the main idea of Occam's Razor.

4.4.7.2. Currently, 2 standard feature selection models in SKLearn library are used and top 25 features are displayed for visualization.

4.4.7.3. High points of this visualization of the important features are -

4.4.7.4. The engineered features created are figuring in the top 25 features.

4.4.7.5. Many of the features thought as important during EDA do figure in the list.

4.4.8.0. ***High-dimensional data visualization***

4.4.8.1. From bivariate analysis, it is already observed that the data is not linearly separable and hence, PCA may not provide additional insights. Towards high-dimensional visualization of the processed data, t-SNE which considers non-linear relations, is carried out as it gives one a sense or intuition of how the data is arranged in a high-dimensional space.

4.4.8.2. Visualizing the output, there is no immediate separation between defaulters and non-defaulters. Basically, it implies that both are a part of a similar class with overall similar properties.

4.4.8.3. It can be inferred that linear models may not work well and hence, other ML models capable of handling complex non-linear relationships shall be employed.

4.5.0.0. **Key Takeaways of the EDA & Feature Analysis phase**

4.5.1.0. *General Highlights*

4.5.1.1. This is a very time-intensive phase involving tinkering with a myriad of features and its combinations, and requires participation from varied sources such as programmers to code effective visualizations and domain experts in order to know what to visualize.

4.5.1.2. Outcomes of this phase are very visually rich and are most useful to convey data to 'non-technical' populace.

4.5.1.3. Insights obtained in the EDA phase may be very valuable as they show patterns not usually discernible; helping translate them into tangible business outcomes.

4.5.2.0. *Specific Highlights to Home Credit dataset-context EDA performed on Google Colab [Free edition]*

4.5.2.1. The dataset is pretty big and needs some form of size optimization as well as storing of intermediate outputs in order to be performed on Colab's free boxes, owing to RAM usage.

4.5.2.2. There are too many features to visualize owing to time constraint and summarization and hence only a few are actually visualized in the notebook and among them, those with significant insights are listed in this report.

4.5.2.3. Owing to the various processes involved, section and subsection headings used in this report are fairly consistent with the ones used in the Colab notebook for ease of understanding and correlation.

4.5.2.4. There is a very high scope for further feature engineering based on preliminary model outputs and feature selection methods employed.

4.5.2.5. Hence, data processing done in this phase such as the features created, imputation strategies followed may be revisited based on outcome of future phases and modeling.

4.5.2.6. Recalling the objectives of this phase listed in 1.4.0.0. above, the work carried out so far accomplishes the intended objectives and is helpful in the formulation of model selection and other activities for upcoming phases.

4.5.2.7. Regarding the t-SNE visualization, which is pretty time-consuming on the Colab box, combinations of perplexity and iterations were tried out with help of Saurabha Daa, my diploma batchmate doing this same project, and the results are not all that different.

4.5.2.8. Regarding some visualizations such as correlations, advanced statistics such as the Phi-K statistic is used for categorical as well as mixed features. A complex correlation visualization may be constructed in the upcoming Advanced Modeling and Feature Engineering phase.

Chapter 5

MODELING AND EXPLAINABILITY

5.1.0.0. **Modeling & Explainability - Summary of approach**

5.1.1.0. Primarily, the processed datasets are used, which are inherently imbalanced.

5.1.2.0. Hence, as an alternative, balancing of the dataset by up sampling of minority class using SMOTE is done and models' predictions on both, the original imbalanced as well as balanced datasets are compared with context to the performance metric.

5.1.3.0. To further add to the breadth of modeling analysis, the original data with and without feature selection are being considered for this modeling phase.

5.1.4.0. Simply put, the following are the initial datasets on which modeling shall be carried out -

5.1.4.1. All features and with the class imbalance

5.1.4.2. All features and perfectly balanced classes

5.1.4.3. Selected features with the class imbalance

5.1.4.4. Selected features and perfectly balanced classes

5.1.5.0. PyCaret library is used extensively for this phase and its compare models module is used on all the four datasets listed above.

5.1.6.0. Compare models simultaneously runs five (and subsequently four) models on each of the datasets.

5.1.7.0. Further modeling and approach are based on the outcomes of the model comparisons.

5.1.8.0. Selected best model is fine-tuned by hyperparameter optimization and subsequently, model Explainability using LIME & SHAP is considered to give us insights for carrying out error analysis.

5.2.0.0. **Modeling & Comparisons**

5.2.1.0. Foundational groundwork

5.2.1.1. All the dependencies for libraries used in this phase are installed. Three new libraries which are installed are LIME and SHAP for error analysis and PyCaret as the overall pipeline for the modeling and comparison phase.

5.2.1.2. Following this, the datasets as listed above, are imported and a custom dataframe size reducer, created and elaborated upon in EDA phase; is re-used here for optimizing the Colab box RAM usage.

5.2.1.3. Importantly, the datasets are balanced as described in the summary, in order to check if removal of imbalance leads to any performance gain.

5.2.2.0. Modeling and comparisons with PyCaret framework

5.2.2.1. Rather than defining a baseline model, a 'baseline dataset' approach is followed wherein all the models (initially five) are predicting on the baseline dataset.

5.2.2.2. The dataset with selected features which is a reduced set is considered as the baseline dataset with the simplifying assumption that the features which were left out are also somewhat independent.

5.2.2.3. This assumption holds validity as during the feature selection, the number of features to be retained was specified manually. Hence, theoretically one can consider the dataset with selected features as the base dataset.

5.2.2.4. Base dataset with imbalance as well as balanced version is fed to the PyCaret pipeline initialised with 5 models - * Logistic Regression, * Ada Boost Classifier, * Naive Bayes Classifier, * Random Forest Classifier, * light Gradient Boosting Machine.

5.2.3.0. With this pipeline, two baseline decisions are taken -

5.2.3.1. For the five models' pipeline, the dataset among balanced and imbalanced ones yielding the better performance metric results is considered for further modeling phases.

5.2.3.2. The model with lowest score on metric is dropped from further evaluation and another of the remaining four (except the top performing) is randomly replaced with another model to have a semblance of variance.

5.2.4.0. Accordingly, the PyCaret pipeline models on the baseline dataset and the outcomes are as follows -

5.2.4.1. Naive Bayes classifier has the lowest accuracy and the AuC evaluates to zero (perhaps due to implementational issues). Regardless, as NB is an oversimplified model, it can be considered the bottom and is dropped.

5.2.4.2. The AuC score for the 'best model' (LightGBM) is substantially higher on the imbalanced dataset than on the balanced one. Thus, it may be posited that balancing the dataset has no real impact on model performance. However, this aspect shall be rechecked with the dataset containing "all" the features again.

5.2.5.0. The PyCaret pipeline, now with four models - * Logistic Regression, * Ada Boost Classifier, * Decision Trees Classifier, * light Gradient Boosting Machine is now fed the dataset with all the features and the outcomes are as follows -

5.2.5.1. The AuC score for the 'best model' (LightGBM) is still higher on the imbalanced dataset than on the balanced one and hence, it can be considered that balancing the dataset is not yielding any significant performance gain in model performance.

5.2.5.2. Importantly, the best model metrics are higher on the dataset with selected features in comparison with the dataset with all the features, implying that the dataset with the selected features is better suited to the task of predicting defaulters.

5.2.6.0. Tuning the best model and cross-checking with Confusion Matrix

5.2.6.1. The LightGBM model turned out to be the best performing one across the overwhelming majority of the various permutations of the input data variants.

5.2.6.2. This model is now hyperparameter-tuned twice - one with emphasis on accuracy maximisation and another on AuC maximisation; for assessing impact of both on predictions.

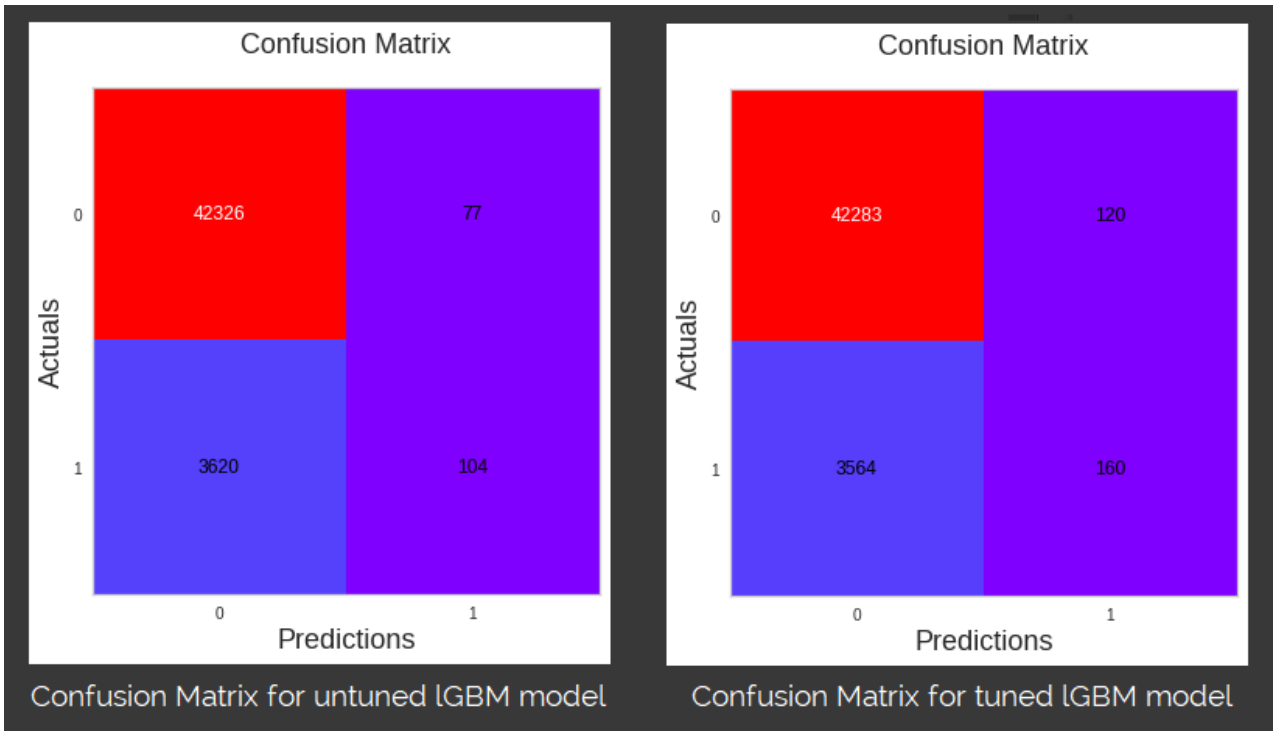
5.2.6.3. The dataset on which these tuned models predict is the imbalanced one with selected features, as mentioned earlier.

5.2.6.4. The AuC score for the tuned lGBM models is, surprisingly, the same for the AuC-focussed tuning as well as accuracy-focussed tuning.

5.2.7.0. Now, as a sanity-check as well as for reiterating the actual purpose of the modeling exercise; which is highest chance of detection of probable defaulters, the confusion matrix is plotted for the following scenarios -

5.2.7.1. Best, untuned model on the imbalanced dataset with selected features
5.2.7.2. Best tuned model on the imbalanced dataset with selected features

5.2.8.0. The Confusion Matrix for the two scenarios are as follows -



5.2.8.1. The numbers which are particularly significant are the correctly identified as well as wrongly predicted (predicted as non-defaulters) defaulters.
5.2.8.2. It is to be noted that as a business, it is pertinent to maximise the prediction of a probable defaulter.
5.2.8.3. The tuned model works better than the untuned model on these parameters, which makes it the best model for the task at hand.

5.3.0.0. **Model Interpretation & Error Analysis**
5.3.1.0. Model Interpretability using SHAP & LIME
5.3.1.1. In order to perform the error analysis, a dataset of the correctly predicted as well as incorrectly predicted datapoints is created.
5.3.1.2. SHAP is run on the best model and the features - EXT_SOURCE_1, EXT_SOURCE_2 and EXT_SOURCE_3 have the biggest impact towards a class prediction.
5.3.1.3. An important point to note is the fact that the engineered features also do figure in the SHAP explanation, which reaffirm their utility towards modeling the predictions.
5.3.1.4. LIME explanation is carried out on a single random incorrectly predicted datapoint for visualizing the features contributing features towards this error. The results are quite different to the SHAP values.
5.3.1.5. Thus, it is decided to run LIME Explainability on 15 data points from correctly predicted as well as incorrectly predicted sets.
5.3.1.6. Top three of the most recurring features attributable towards the prediction, are identified for further error analysis.

5.3.2.0. Error Analysis based on LIME explanation

5.3.2.1. On carrying out LIME Explainability for 15 random data points each in correctly predicted as well as incorrectly predicted datasets, the following are observed -

5.3.2.2. The features - EXT_SOURCE_3, EXT_SOURCE_2 and AMT_CREDIT_SUM_DEBT are the top three features influencing the incorrectly predicted points' classification.

5.3.2.3. Also, the features - EXT_SOURCE_3, EXT_SOURCE_2 and EXT_SOURCE_1 are the top three features impacting the correctly predicted points' classification.

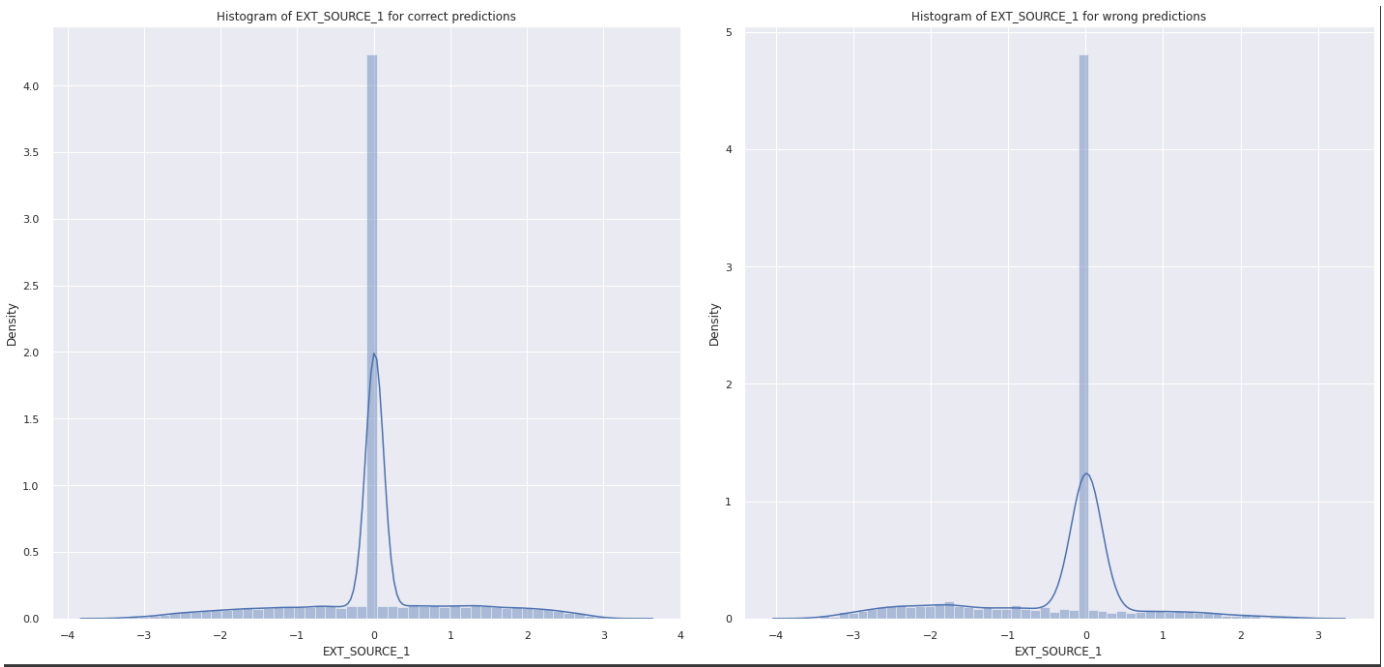
5.3.2.4. In order to observe any inherent separability or an anomaly in the top features regarding the correctly as well as incorrectly predicted points, box-plots and histograms are plotted for each of the features for both the datasets.

5.3.3.0. The following are the key insights of the graphical analysis -

5.3.3.1. Upon reviewing the boxplots for the 4 features, there is no fundamentally different or anomalous behaviour between the statistics pertaining to the data points of the 2 sets.

5.3.3.2. Considering the imbalanced dataset used and the high accuracy of the final best model, the extreme skew between the boxplots is understandable.

5.3.3.3. Upon observing the histograms for the features, one can notice the similarity regarding peaks as well as the variability distribution among the correctly predicted as well as the misclassified data points.



Histogram & Distplot for the EXT_SOURCE_1 feature for correctly predicted as well as misclassified points - Similar peak locations and spread of data

5.3.3.4. There is no clear distinguishing attribute of the feature to help one identify a misclassification.

5.3.3.5. This is expected as the features contributing towards misclassification are the same as those resulting in a classification, as evidenced from the LIME explanation.

5.3.3.6. This leads to the consideration for creation of additional features which might help in reducing the errors or misclassifications.

5.4.0.0. **Summary of this phase**

5.4.1.0. In this phase, the strategy of balancing out the minority class was tried out using SMOTE and it is evidenced that the balancing did not result in any gain in model prediction performance. Hence, the strategy is dropped.

5.4.2.0. The dataset with all features was compared with the set with selected features over a series of models and it was observed that the dataset with the limited, selected features is a better input for modeling prediction of a defaulter. This insight is important towards advanced feature engineering in the next phase.

5.4.3.0. SHAP analysis reiterated the importance of feature engineering as the features engineered in previous phase figure in the top features influencing the model's prediction.

Chapter 6

ADVANCED MODELING AND FEATURE ENGINEERING

6.1.0.0. **Summary of approach**

6.1.1.0. Primarily, the processed datasets are used, which are inherently imbalanced. This dataset was found to be most optimal from the previous phase.

6.1.2.0. The previous phase of error analysis highlighted the need for advanced feature engineering which shall be done here.

6.2.0.0. **Creation of advanced features**

6.2.1.0. ***Principal Component Analysis [PCA]***

6.2.1.1. PCA is performed with 5 components i.e., PCA outcome has 5 features. A data frame is formed with these 5 features and the same is visualized.

6.2.1.2. Subsequently, PAC with 2 components is carried out and these features are added to the set.

6.2.1.3. Two new columns - Pred and pred_case is created. Pred column indicates whether the data point is correctly predicted or not; pred_case indicates prediction with correct and wrong label e.g., Correct_0 means the prediction is 0 and it is correctly predicted. We make pair plots from the 5 PCA features as well as the 2 features.

6.2.1.4. 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.

6.2.1.5. 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.

6.2.2.0. ***Training model with confidence as the output***

6.2.2.1. Light GBM with tuned parameters (from Phase 3) is trained on the dataset with selected features. PyCaret is used for this training as was done in Phase 3.

6.2.2.2. 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 labelled as Correct_Low, Wrong_High and Wrong_Low.

6.2.2.3. This new column is added based on the Score column.

6.2.2.4. The score column indicates the probability of predicted Label. A cut off point is selected as defining low or high confidence.

6.2.2.5. 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 labelled Correct_High.

6.2.2.6. Similarly other points are classified. This new test data frame is named predict_test.

6.2.2.7. Prediction is also made on train data and confidence columns are added. This new data frame is named train_predict.

6.2.3.0. ***LDA as a new feature***

6.2.3.1. Subsequently, LDA is performed on the resultant dataset and based on LDA, a new feature column named LDA is added to both train and test data.

6.3.0.0. **Impact of the advanced features on the models**

6.3.1.0. *LightGBM model*

6.3.1.1. LightGBM is trained on the dataset with added features and further tuned.

6.3.1.2. Predictions are made on test data and corresponding accuracy; AUC and confusion matrix are obtained.

6.3.2.0. Following are the observations when compared to best_model -

6.3.2.1. Overfitting is noticed with substantial difference in accuracy & AuC values between predictions on train data and test data.

6.3.2.2. Although AuC has increased compared to best_model, accuracy has decreased.

6.3.2.3. Based on the above observations, it is concluded that the new lGBM model is not better than best_model.

6.3.3.0. *Stacking based model*

6.3.3.1. Stacked model is trained using PyCaret, consisting of - AdaBoost, DT and LightGBM as the estimators.

6.3.3.2. Predictions are made on test data and corresponding accuracy, AuC and confusion matrix are obtained.

6.3.4.0. Following are the observations when compared to best_model -

6.3.4.1. Overfitting is observed with substantial difference in accuracy & AuC values between predictions on train data and test data.

6.3.4.2. Accuracy and AuC are lower in comparison to best_model.

6.3.4.3. Based on the above observations, it is concluded that stacking-based model is not better than best_model.

6.3.5.0. *Neural Network based model*

6.3.5.1. Neural network model is created using Keras.

6.3.6.0. Following are the observations when compared to best_model:

6.3.6.1. Overfitting is observed although not major.

6.3.6.2. Accuracy and AuC have decreased compared to best_model.

6.3.6.3. Based on the above observations, it is concluded that a NN based model is not better than the best_model.

6.3.7.0. For the sake of sanity check and to check efficacy of NN on the data at hand, Neural Network is trained on original data (dataset with selected features).

6.3.8.0. Following are the observations when compared to best_model:

6.3.8.1. Overfitting is not observed.

6.3.8.2. Accuracy and AuC have decreased only slightly compared to best_model.

6.3.8.3. The number of people who should have been rejected for loan but are predicted as eligible for loan is substantial.

6.3.8.4. Based on the above observations, it is concluded that the lGBM model is the best case scenario.

6.4.0.0. **Summary of this phase**

6.4.1.0. Upon evaluating the myriad of models with varying datasets and feature permutations, it is evidenced that tuned LightGBM is the best performer.

6.4.2.0. Even deep learning models fared relatively poorly in comparison to the tuned LightGBM model.

6.4.3.0. This phase concludes the model training step. Further the finalised best model shall be deployed in the next phase.

Chapter 7

MODEL DEPLOYMENT

7.1.0.0. **Summary of deployment process**

7.1.1.0. The LightGBM model is the best model and PyCaret library was used for the whole process from data ingestion to the prediction stage on Colab notebook.

7.1.2.0. For the deployment phase, the entire data pipeline and model were recreated using sklearn.

7.1.3.0. All the necessary codes, input data and auxiliary files were pushed to a repository on GitHub.

7.1.4.0. Streamlit is used owing to its simplicity and intuitive UI.

7.1.5.0. The app is hosted on Heroku free tier service and deployed.

7.2.0.0. **Initiating the deployment process**

7.2.1.0. The PyCaret pipeline & its quirks

7.2.1.1. The best model & pipeline developed in the previous phase was used for deployment using FastAPI on Heroku.

7.2.1.2. The PyCaret model is substantially large (approximately 180mb). Though I could load the pickled model in Colab & get the predictions on defaulting tendency for the entire test dataset, deployment on Heroku threw an error. The error also persisted upon using Streamlit.

7.2.1.3. A possible cause might be the PyCaret pipeline or the usage of GitLFS while pushing the large files to Git repo.

7.2.1.4. To debug considering GitLFS as a compatibility issue, to reduce the file sizes, the model was trained using 50% & 25% of the original training data. However, there was no improvement.

7.2.1.5. Thus, presuming that PyCaret has compatibility issues in deployment, considering the time constraint, I decided to recreate the entire model and data pipeline in sklearn.

7.2.2.0. Recreating the model & data pipeline in sklearn

7.2.2.1. The ipython notebook has the entire modeling & data pipeline created in the sklearn library. The pickled model and pipelines were relatively very light-sized which eliminated the need to use LFS to push these files to the repo.

7.2.2.2. A thing worth noting is the amount of coding involved in using sklearn compared to PyCaret.

7.2.2.3. Importantly, the deployment on Heroku was successful emphasising that PyCaret or the LFS had compatibility issues.

7.2.3.0. The detailed process of deployment

7.2.3.1. Before listing the steps involved in this iterative process, it is important to list the various deployment strategies executed as this has bearing on the whole cycle.

7.2.3.2. The following deployment options were carried out -

***** FastAPI + Heroku *FastAPI + AWS *Streamlit + Heroku *Streamlit + Azure

*Streamlit + AWS

7.2.3.3. Initially, the model in the PyCaret framework was deployed using FastAPI on Heroku. For this, all the code files and data sets along with the auxiliary files need to be pushed to the GitHub repository.

7.2.3.4. Owing to the large file size of the datasets as well as the PyCaret's model and pipeline size, pushing to Git through CLI or desktop was not possible and Git LFS [Large File Storage] was used to push these large files to the repo. Size restrictions for GitHub file upload & CLI/Desktop are 25mb & 100mb respectively whereas the files were around 300mb.

7.2.3.5. Usage of the LFS and PyCaret apparently has compatibility issues with either Streamlit or Heroku itself as the deployed app crashed with H10 error.

[A screengrab of the copied log for H10 error highlighted]

7.2.3.6. Searching on the internet for the error in our environment context highlighted compatibility errors between LFS and possibly PyCaret.

7.2.3.7. Owing to the time constraint, a detailed debugging was skipped in favour of trying out options to reduce dependency on LFS & PyCaret.

7.2.3.8. Thus, in order to create a light-weight model as well as the total pipeline, sklearn was used to build the whole framework from scratch.

7.2.3.9. The resulting files [pickled files] were very lightweight which eliminated the need for LFS.

7.2.3.10. By implementing the dataflow pipeline in sklearn, PyCaret library was dropped too.

7.2.3.11. The whole pipeline and model creation using sklearn is coded on this Colab notebook and the outputs are pickled for usage.

7.3.0.0. **<u>The actual deployment cycles</u>**

7.3.1.0. Having fixed the data pipeline and the model, the first deployment combination was FastAPI + Heroku.

7.3.1.1. FastAPI uses uvicorn owing to its ASGI implementation making it pretty fast.

7.3.1.2. This was very easy to set up. The UI is by OpenAPI (previously Swagger) which is pretty barebones.

7.3.1.3. Using Jinja, making a more customised UI may have been possible but time constraint compelled me to try out other options.

7.3.1.4. FastAPI is very easy to work with and for my case, the documentation needed is fairly elaborate. I deployed a few toy example models and they were extremely fast and behaving as intended. However, the UI (OpenAPI/Swagger) was too plain for me and I could not use Jinja effectively to customise the UI which made me try out Streamlit.

7.3.2.0. While using Streamlit, I tried out Azure, originally with the PyCaret pipeline. Setting up Azure though fairly easy, takes a lot of time for setting up the box as well as after connecting to Git repo.

7.3.2.1. The PyCaret model on Azure was not successfully deployed throwing up memory exceed error. The overall time to rebuild made me check the AWS platform.

7.3.3.0. FastAPI + AWS / Streamlit + AWS was tried out initially with the PyCaret model and deployment failed due to the RAM consumption. The sklearn model was deployed using Streamlit on AWS and it was successfully deployed.

7.3.3.1. Biggest advantage of AWS is the fact that large files can be easily pushed to the remote box using FTP programs like WinSCP.

7.3.3.2. AWS has elaborate documentation and online resources which make setting up ec2 instances extremely easy for people with coding experience.

7.3.3.3. Deployment on AWS though well documented, especially on external platforms, is relatively involved. I needed two additional software - Putty/PuttyGen & WinSCP for SSHing into and doing file transfers with the box respectively.

7.3.3.4. Code's RAM usage needs to be quite optimised as a deployment that worked on Heroku failed on AWS due to RAM consumption. Thus, I kept the AWS implementation on backburners and considered Heroku as primary.

7.3.4.0. **A summary of the pros & cons of each option I used while deploying my model is given below** -

Note - This table is compiled based on my experience while deploying the PyCaret &/or sklearn model and my experience of coding.

Sl. no.	Platform/ Software	Pros	Cons
1	FastAPI	☑ Easy to set up and get running ☑ Actually fast [ASGI implementation]	✕ Limited examples especially for Frontend/ UI customization
2	Streamlit	☑ Easy to set up and get running ☑ Good community support for quick front-end customization	None
3	AWS	☑ Elaborate documentation and resources/examples ☑ FTP to transfer large files avoids LFS dependence	✕ Free-tier box provides relatively limited compute resources, need very optimised code for using box fully ✕ Non-coding beginners can be intimidated by steps in getting the box running ✕ Additional software for SSH & FTP required for deployment ✕ Process of keys/ private keys and SSH may be little intimidating
4	Azure	☑ Beginner-friendly, especially windows users ☑ Elaborate session & error logs ☑ No additional software needed to deploy app	✕ Fairly time-consuming process for initial set up as well as during deployment ✕ Lot of tools like PowerBI integrated in dashboard, usage of which has a steep learning curve for beginners
5	Heroku	☑ Beginner-friendly & easy to set up and get running, no server configuration ☑ No additional software needed to deploy app ☑ Good logs and version control	✕ Procfile content need fair knowledge of CS ✕ Dependence on GitHub [& LFS for large files] sometimes throws errors

7.4.0.0. **<u>The finalised deployment platform</u>**

7.4.1.0. After trying out the mentioned combinations of platforms and services, I opted for Streamlit+Heroku as the primary method of deployment for the following reasons -

7.4.2.0. Streamlit allowed me to customise the app UI much better than FastAPI to the extent I could.

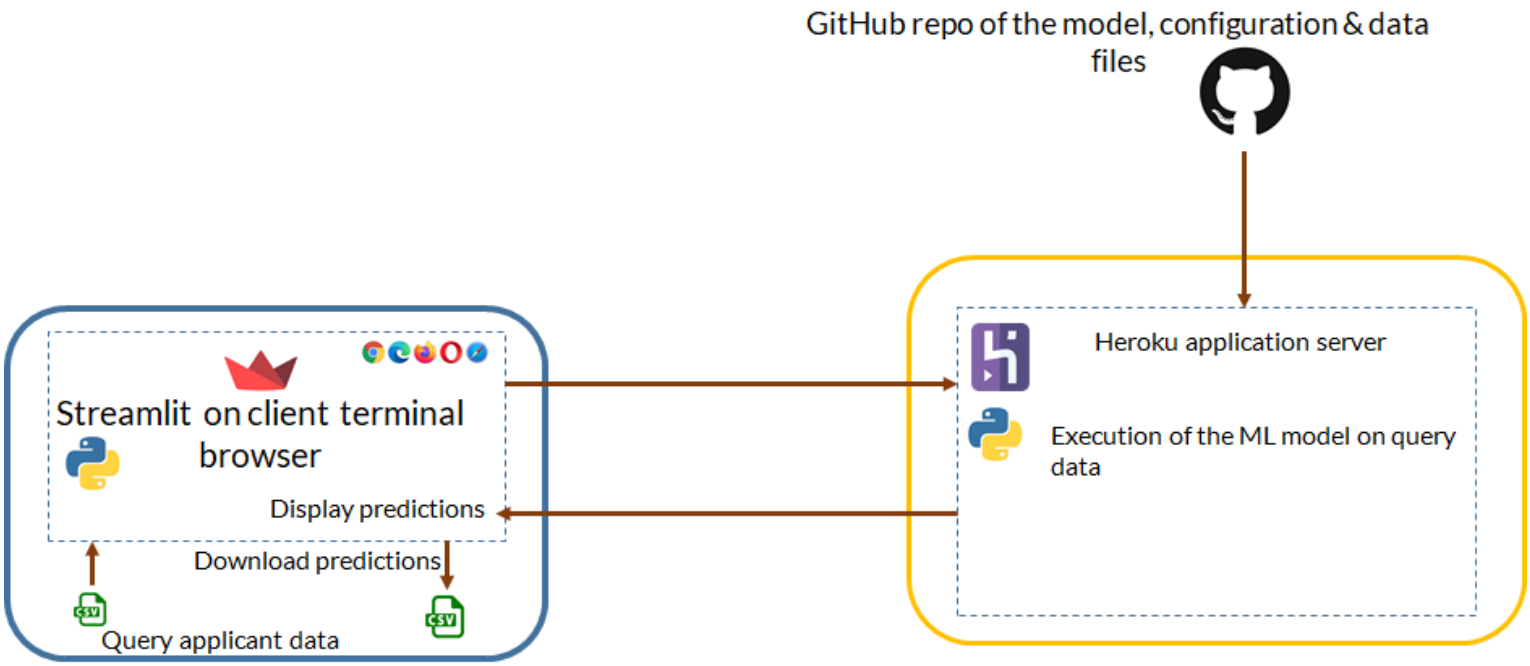
7.4.3.0. Heroku required the least amount of time for iterative deployment and the entire repository being on GitHub, I could modify, build & redeploy from anywhere.

7.4.4.0. <u>Basic architecture of the app system</u>

7.4.4.1. The user interacts with the app via any browser on their local PC running the Streamlit client by uploading the query csv file containing the applicant data.

7.4.4.2. The model hosted on the remote Heroku box computes the predictions and sends back the results which are displayed on the user's browser as well as can be downloaded.

7.4.4.3. Upon linking the GitHub repo with the Heroku site for the first time, the files are pulled into the Heroku box.



Simplified view of the model system architecture [img source - self]

7.5.0.0. **<u>Highlights of the deployed app</u>**

7.5.1.0. ***App engagement***

7.5.2.0. The app accepts the home credit applicant details in a CSV file as is in the test dataset.

7.5.3.0. A downloadable template is provided for the user to enter data into.

7.5.4.0. Individual form fields are not provided owing to the large number of fields which will result in an unpleasant UX.

7.5.5.0. The output of the model predictions is displayed on the screen as an interactable dataframe as well as a downloadable CSV file appended to the original query set.

7.5.6.0. Importantly, following error handling methods are implemented -

7.5.7.0. The uploaded csv is checked for correctness w.r.t. the actual feature names required in template and in case of mismatch, displays a message stating the same.

7.5.8.0. When a new, unseen categorical variable is encountered in the query data, handling is done by ignoring it which is implemented by setting ***'handle_unknown'*** parameter to ***'ignore'***. This ignores the unseen category values and proceeds ahead.

7.5.9.0. ***Scalability, Throughput, Latency and real-world case***

7.5.10.0. The app was fed the Home Credit raw test dataset consisting of around 50k applicant records with a file size of approximately 26mb.

7.5.11.0. The app, after upload [depending on internet connectivity took around 5 -30 seconds] does the entire data processing and predicts the defaulting tendency for the applicants in less than 40 seconds.

7.5.12.0. Considering the real-world scenario, the latency is not a strict requirement and is acceptable.

7.5.13.0. With context to throughput, as the app can be run frequently per day or even per application, the throughput volumes are not a limiting case.

7.5.14.0. ***Visible limitations and scope for improvement/innovation***

7.5.15.0. Going through the logic of current execution, the query data is converted to a Pandas dataframe and compared with existing Bureau & Previous Home Credit data (also a Pandas dataframe). This implementation might be checked with SQL db for optimised and scalable performance which 'might' improve the system latency or reduce memory requirement.

7.5.16.0. Going ahead, the debugging of the issue with PyCaret may also be done so that the low code tool can also be used.

7.6.0.0. **Summary**

7.6.1.0. The model and data pipeline using sklearn was successfully deployed on Heroku using the Streamlit framework.

7.6.2.0. Latency requirements being non-stringent, the time taken to predict the defaulting tendency for the test dataset is acceptable.

7.6.3.0. As the prediction system can be run frequently, even per applicant as requirement is not real-time, the system throughput is not a cause for concern as the app hosted on the free tier Heroku box handled predictions for around 50K data points in one batch.

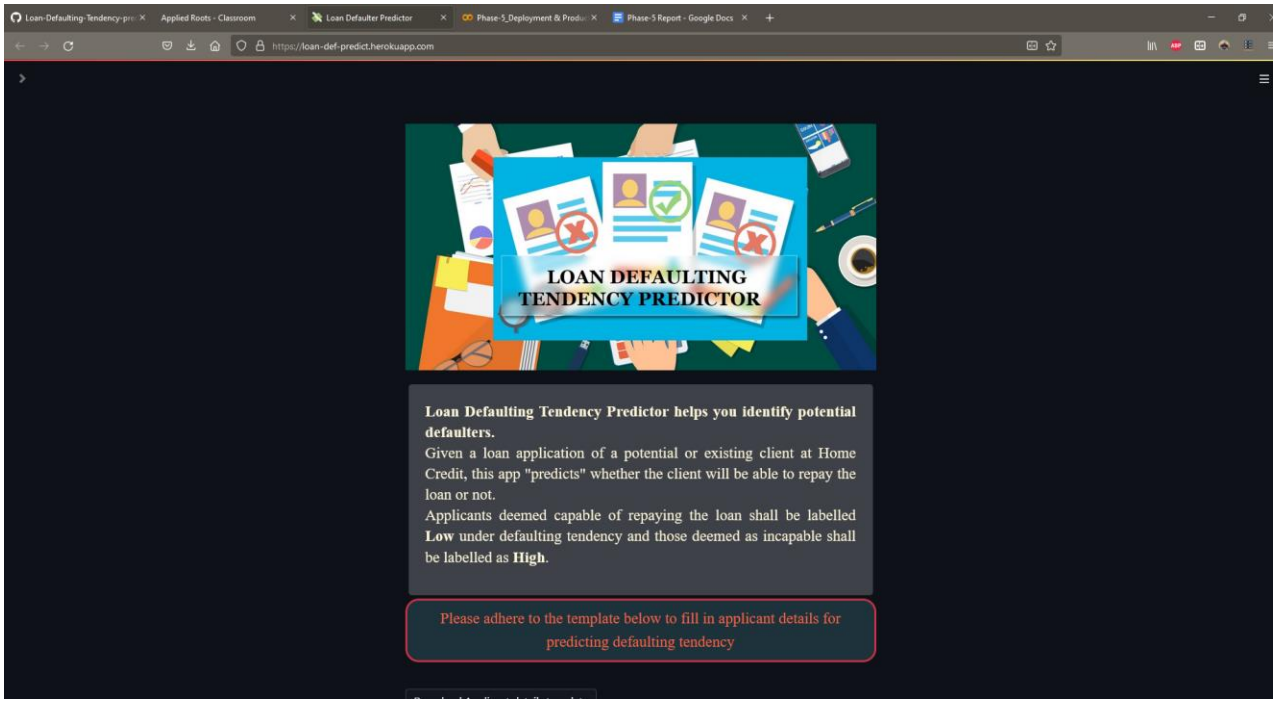
Chapter 8

CONCLUSION AND SUMMARIZATION

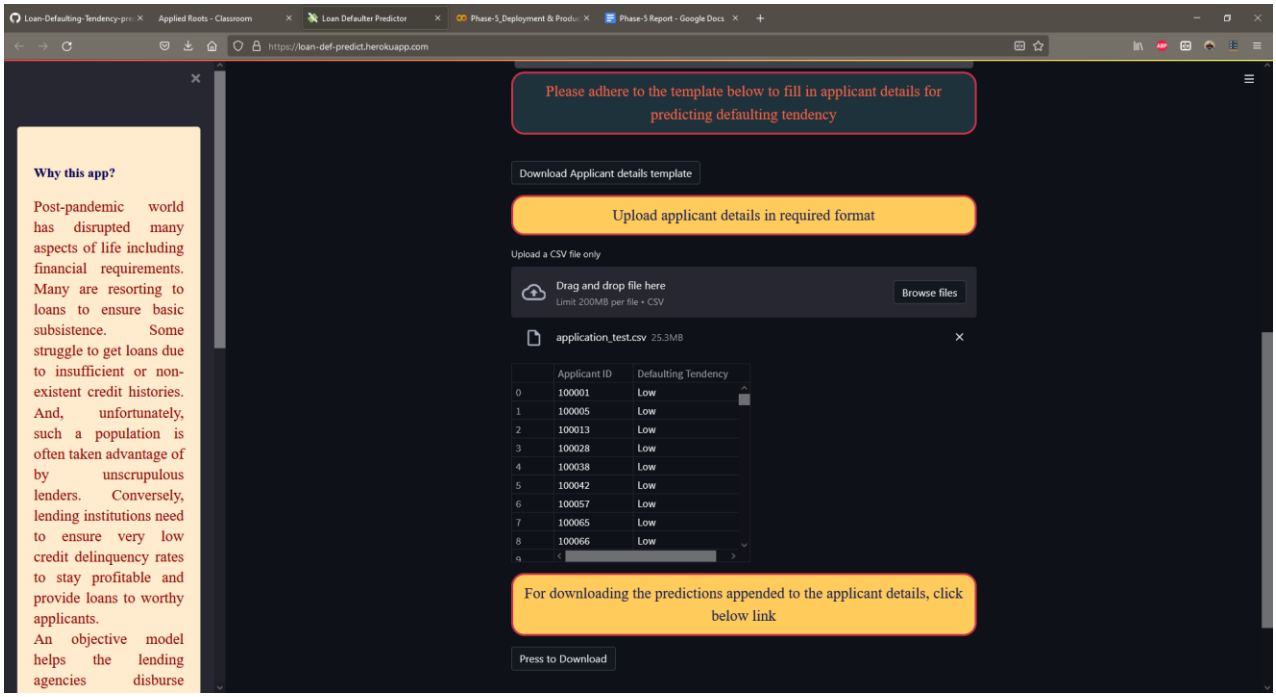
8.1.0.0. The end-to-end project on “Predicting the capability of each applicant in repaying a loan” was successfully carried out using the Kaggle Home Credit Default Risk dataset.

8.1.0.0. Domain knowledge from the literature review phase was used to create features which were quite useful in the defaulting tendency predictions.

8.2.0.0. App interface



App home screen interface screengrab



App interface with prediction & sidebar for introduction - screengrab

8.3.0.0. Important information, resources & links

8.3.1.1. Libraries used

✅ Python Libraries used

Name	What its used for
Streamlit	*An open source app framework for building beautiful data and ML apps*
PyCaret	*An open-source, low-code machine learning library in Python that automates machine learning workflows.*
PyOD	*A comprehensive and scalable Python toolkit for detecting outlying objects in multivariate data.*
Phi-K	*A new and practical correlation coefficient based on several refinements to Pearson's hypothesis test of independence of two variables.*
MLxtend	*A Python library of useful tools for the day-to-day data science tasks.*
Keras	*An open-source software library that provides a Python interface for artificial neural networks.*
SHAP	*A game theoretic approach to explain the output of any machine learning model implemented in Python.*
LIME	*A Python library about explaining what machine learning classifiers (or models) are doing, supporting explaining individual predictions*
sklearn	*A Python library containing an ensemble of simple and efficient tools for predictive data analysis*
seaborn	*A Python data visualization library providing a high-level interface for drawing attractive and informative statistical graphics.*
matplotlib	*A comprehensive library for creating static, animated and interactive visualizations in Python.*
Pandas	*A Python library for data manipulaion and analysis*
Numpy	*A Python library for numerical computation*

8.3.1.2. The GitHub repository for all the Colab notebooks along with a clear README for navigating through the same is here.
8.3.1.3. The GitHub repository for the deployed app with complete documentation & clear README for using the app is here.
8.3.1.4. The dataset used for training the model is the Kaggle's Home Credit Default Risk dataset from here.
8.3.1.5. Finally, the app is deployed through Streamlit framework and hosted on Heroku here.