1.0.0.0. **INTRODUCTION**

1.1.0.0. Setting the context

1.1.1.0. This report is to be read in conjunction with the Google Colab notebook for Modeling & Error Analysis on the Home Credit loan defaulter prediction problem hosted on Kaggle here.

1.1.2.0. An understanding of the complete problem context and high level summary of the datasets used can be sought from here.

1.1.3.0. The datasets used in this phase are the ones already processed based on the insights and feature engineering in earlier EDA phase, which can be referred to, here.

1.2.0.0. A quick refresher about Home Credit's motivation for predicting potential defaulters

1.2.1.0. Though there are a lot of people seeking loans from banks and lending institutions, only a few of them get approved. This is primarily because of insufficient or non-existent credit histories of the applicant. Such a population is taken advantage of by untrustworthy lenders. In order to make sure that these applicants have a positive loan taking experience, Home Credit uses Data Analytics to predict the applicants' loan repayment abilities, trying to ensure that the clients capable of loan repayment do not have their applications rejected.

1.3.0.0. Modeling & Error Analysis - Summary of approach

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

1.3.2.0. Hence, as an alternative, balancing of the dataset by upsampling 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.

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

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

1.3.4.1. All features and with the class imbalance

1.3.4.2. All features and perfectly balanced classes

1.3.4.3. Selected features with the class imbalance

1.3.4.4. Selected features and perfectly balanced classes

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

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

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

1.3.8.0. Selected best model is fine-tuned by hyperparameter optimization and subsequently, error analysis is carried out.

2.0.0.0. **MODELING AND COMPARISONS**

2.1.0.0. Foundational groundwork

2.1.1.0. 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.

2.1.2.0. 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.
2.1.3.0. Importantly, the datasets are balanced as described in the summary, in order to check if removal of imbalance leads to any performance gain.

2.2.0.0. <u>Modeling and comparisons with PyCaret framework</u>
2.2.1.0. Rather than defining a baseline model, a 'baseline dataset' approach is followed wherein all the models (initially five) are predicting on the baseline dataset.
2.2.2.0. 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.
2.2.3.0. 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.
2.2.4.0. Base dataset with imbalance as well as balanced version are fed to the PyCaret pipeline initialised with 5 models - * Logistic Regression, * Ada Boost Classifier, * Naive Bayes Classifier, * Random Forest Classifier, * light Gradient Boosting Machine.

2.2.5.0. With this pipeline, two baseline decisions are taken -
2.2.5.1. For the five models pipeline, the dataset among balanced and imbalanced ones yielding the better performance metric results is considered for further modeling phases.
2.2.5.2. 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.

2.2.6.0. Accordingly, the PyCaret pipeline models on the baseline dataset and the outcomes are as follows -
2.2.6.1. 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.
2.2.6.2. 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.

2.2.7.0. 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 -
2.2.7.1. 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.
2.2.7.2. 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.

2.3.0.0. <u>Tuning the best model and cross-checking with Confusion Matrix</u>

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

2.3.2.0. 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.

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

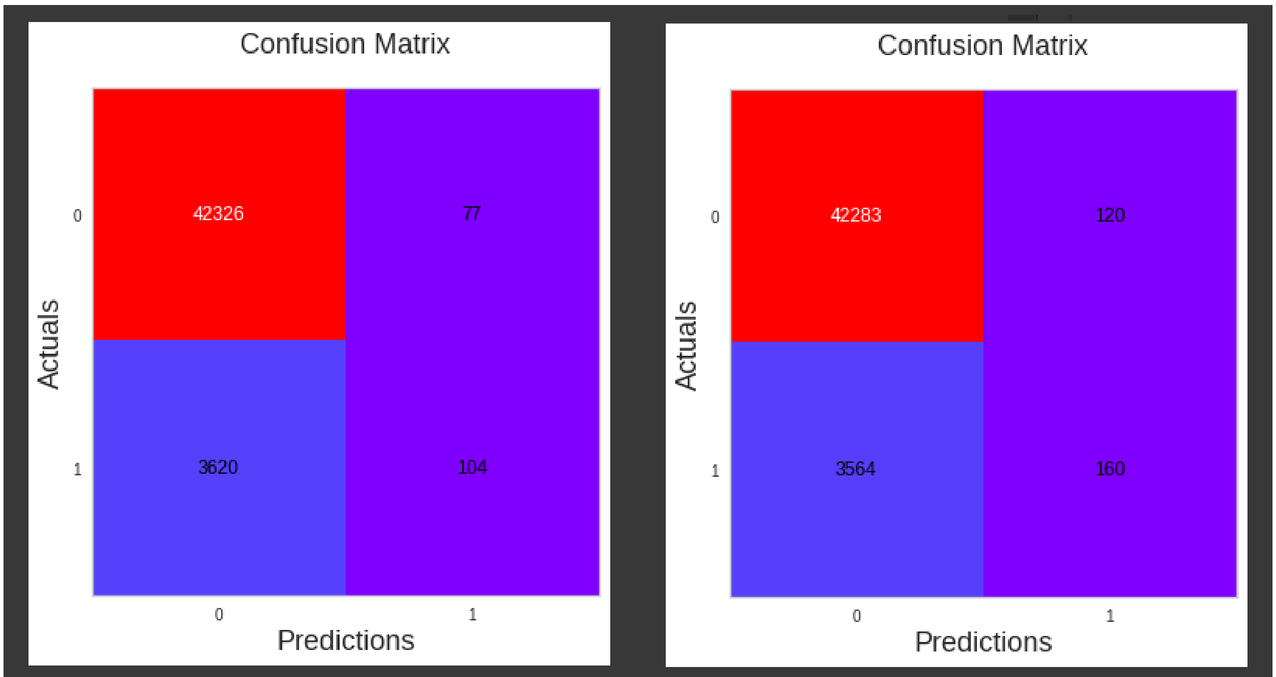
2.3.4.0. The AuC score for the tuned lGBM models is, surprisingly, the same for the AuC-focussed tuning as well as accuracy-focussed tuning.

2.3.5.0. 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 -

2.3.5.1. Best, untuned model on the imbalanced dataset with selected features

2.3.5.2. Best tuned model on the imbalanced dataset with selected features

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



Confusion Matrix for untuned lGBM model Confusion Matrix for tuned lGBM model

2.3.6.1. The numbers which are particularly significant are the correctly identified as well as wrongly predicted (predicted as non-defaulters) defaulters.

2.3.6.2. It is to be noted that as a business, it is pertinent to maximise the prediction of a probable defaulter.

2.3.6.3. The tuned model works better than the untuned model on these parameters, which makes it the best model for the task at hand.

3.0.0.0. **INTERPRETABILITY AND ERROR ANALYSIS**

3.1.0.0. <u>Model Interpretability using SHAP & LIME</u>

3.1.1.0. In order to perform the error analysis, a dataset of the correctly predicted as well as incorrectly predicted datapoints is created.

3.1.2.0. 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.

3.1.3.0. 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.

3.1.4.0. 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.

3.1.5.0. Thus, it is decided to run LIME explainability on 15 data points from correctly predicted as well as incorrectly predicted sets.

3.1.6.0. Top three of the most recurring features attributable towards the prediction, are identified for further error analysis.

3.2.0.0. <u>Error Analysis based on LIME explanation</u>

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

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

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

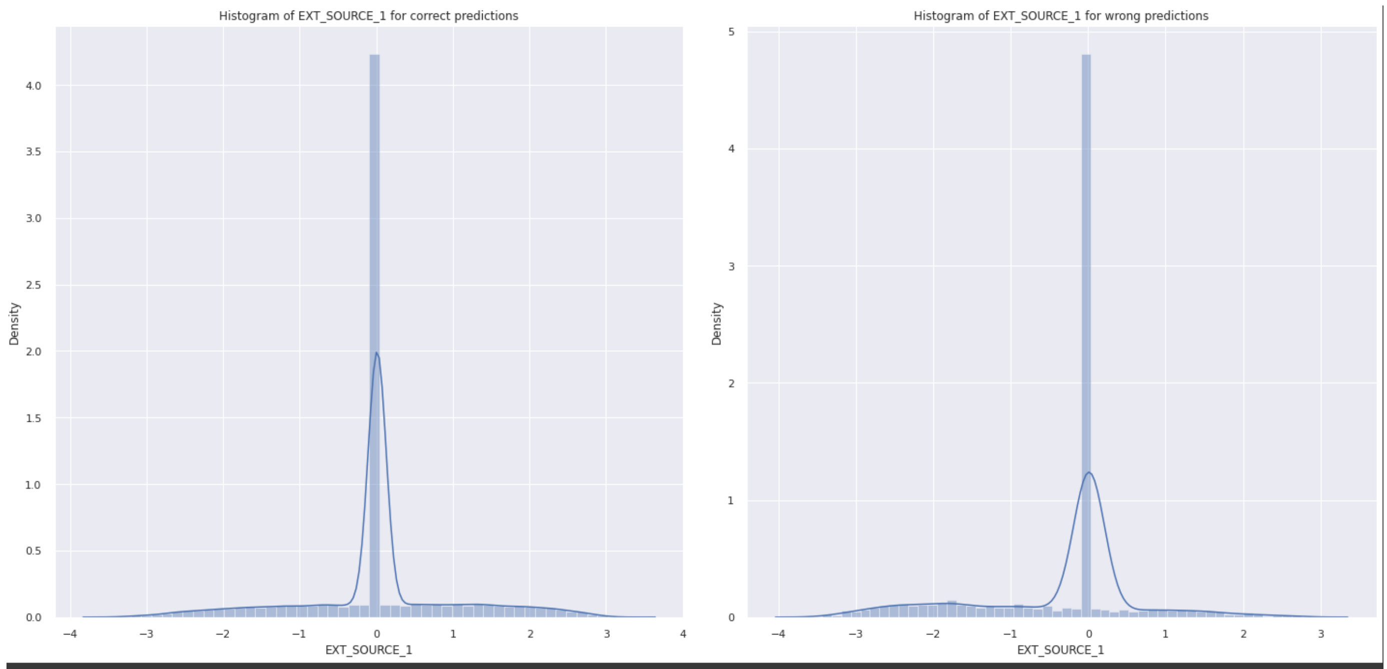
3.2.4.0. In order to observe any inherent separability or an anomaly in the top features regarding the correctly as well as incorrectly predicted points, box-plots and histograms are plotted for each of the features for both the datasets.

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

3.2.5.1. 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.

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

3.2.5.3. 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

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

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

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

4.0.0.0. **CONCLUSION**

4.1.0.0. 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.

4.2.0.0. 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.

4.3.0.0. 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.