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 Advanced Modeling and Feature Engineering 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.

1.1.4.0. The model referred to as the best model and the dataset with selected features is most suitable towards maximising the model performance; facts evidenced in the previous phase which can be read 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. Summary of approach

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

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

2.0.0.0. **ADVANCED FEATURE CREATION**

2.1.0.0. Principal Component Analysis [PCA]

2.1.1.0. 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.

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

2.1.3.0. Two new columns - Pred and pred_case are 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 pairplots from the 5 PCA features as well as the 2 features.

2.1.4.0. 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.

2.1.5.0. 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.

2.2.0.0. Training model with confidence as the output

2.2.1.0. Light GBM with tuned parameters (from Phase 3) is trained on dataset with selected features. Pycaret is used for this training as was done in Phase 3.
2.2.2.0. 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.
2.2.3.0. This new column is added based on the Score column.
2.2.4.0. The score column indicates the probability of predicted Label. A cut off point is selected as defining low or high confidence.
2.2.5.0. 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.
2.2.6.0. Similarly other points are classified. This new test data frame is named predict_test.
2.2.7.0. Prediction is also made on train data and confidence columns are added. This new data frame is named train_predict.

2.3.0.0. LDA as a new feature
2.3.1.0. 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.

3.0.0.0. TRAINING MODELS WITH THE ADDITIONAL FEATURES

3.1.0.0. lGBM model
3.1.1.0. Lightgbm is trained on the dataset with added features and further tuned.
3.1.2.0. Predictions are made on test data and corresponding accuracy, AUC and confusion matrix are obtained.

3.1.3.0. Following are the observations when compared to best_model -
3.1.3.1. Overfitting is noticed with substantial difference in accuracy & AuC values between predictions on train data and test data.
3.1.3.2. Although AuC has increased compared to best_model, accuracy has decreased.
3.1.3.3. Based on the above observations, it is concluded that the new lGBM model is not better than best_model.

3.2.0.0. Stacking based model
3.2.1.0. Stacked model is trained using PyCaret, consisting of - Ada Boost, DT and lightGBM as the estimators.
3.2.2.0. Predictions are made on test data and corresponding accuracy, AuC and confusion matrix are obtained.

3.2.3.0. Following are the observations when compared to best_model -
3.2.3.1. Overfitting is observed with substantial difference in accuracy & AuC values between predictions on train data and test data.
3.2.3.2. Accuracy and AuC are lower in comparison to best_model.
3.2.3.3. Based on the above observations, it is concluded that stacking-based model is not better than best_model.

3.3.0.0. Neural Network based model

3.3.1.0. Neural network model is created using Keras.

3.3.2.0. Following are the observations when compared to best_model:
3.3.2.1. Overfitting is observed although not major.
3.3.2.2. Accuracy and AuC have decreased compared to best_model.
3.3.2.3. Based on the above observations, it is concluded that NN based model is not better than the best_model.

3.3.3.0. 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).

3.3.4.0. Following are the observations when compared to best_model:
3.3.4.1. Overfitting is not observed.
3.3.4.2. Accuracy and AuC have decreased only slightly compared to best_model.
3.3.4.3. The number of people who should have been rejected for loan but are predicted as eligible for loan is substantial.
3.3.4.4. Based on the above observations, it is concluded that the lGBM model is the best case scenario.

4.0.0.0. CONCLUSION

4.1.1.0. Upon evaluating the myriad of models with varying datasets and feature permutations, it is evidenced that tuned lightGBM is the best performer.
4.1.2.0. Even deep learning models fared relatively poorly in comparison to the tuned lightGBM model.
4.1.3.0. This phase concludes the model training step. Further the finalised best model shall be deployed in the next phase.
