Credit Card Behaviour Score - Report

Team – PRIME Transformers

January 15, 2025

Abstract

This report describes the methodology and results of the Credit Card Behaviour Score prediction model developed for Bank A. The goal was to predict the probability of customers defaulting on their credit cards using historical data. The report covers the data analysis, model selection, evaluation, and insights derived from the analysis.

Contents

1 Problem Statement

1.1 Objective The goal is to develop a Behaviour Score predictive model for Bank A's Credit Card customers, designed to predict the likelihood of future defaults. This score will be integrated into the bank's risk management framework to assist with portfolio management, eligibility, limit assignment, and interest rate decisions..

1.2 Data Description The analysis uses two datasets: Development Data (96,806 samples) and Validation Data (41,792 samples). The development data includes a variety of features such as On-us attributes (credit limits), Transaction-level attributes (transaction counts and values), Bureau tradeline attributes (credit history), and Bureau enquiry attributes (recent credit inquiries), with the target variable bad flag indicating default status. The validation data mirrors the development data's features but lacks the bad flag, and will be used to validate the model.

2 Data Exploration and Preprocessing

2.1 Data Overview The data preprocessing phase involved several important steps aimed at preparing the dataset for model training. These steps included exploratory data analysis (EDA), identification of irrelevant features, handling of missing values, removal of highly correlated features, and dimensionality reduction. Below is a comprehensive breakdown of each stage.

2.2 Feature Breakdown The first step in the preprocessing pipeline was to conduct an exploratory data analysis (EDA) to understand the structure of the dataset. This was essential for identifying different types of attributes and the distribution of data.

- **On-us Attributes**: 48 features
- **Transaction Attributes**: 664 features
- **Bureau Attributes**: 452 features
- **Bureau Enquiry Attributes**: 50 features

These groups were identified based on the attributes' type and usage in the dataset.

2.3 Missing Data and Handling To ensure completeness of the data, we replaced all missing values (NaN) with zeros, as this would have no effect on model training. This step ensures that the model can process the dataset without encountering errors or issues due to missing values.

Next, we examined the dataset for columns that contained only zeros. These columns do not provide any meaningful information for model training and were thus removed.We found that exactly 50 columns were all zeroes. Those columns are listed below :

- **Bureau-related attributes:** bureau_4, bureau_16, . . . , bureau_447

- **Onus-related attribute:** onus_attribute_28
- **Bureau enquiry-related attributes:** bureau_enquiry_7, bureau_enquiry_17, . . . , bureau_enquiry_47

These columns were entirely zero-valued and were removed to reduce dimensionality and prevent them from adversely affecting the model as they would have no effect on model's performance but rather woud increase computation time

2.4 Feature Engineering In this section, we describe the feature engineering steps undertaken to enhance the dataset for model training.

- **Handling Highly Correlated Features:** We performed a correlation analysis to identify highly correlated features within each attribute group. High correlation can lead to multicollinearity, which may degrade the model's performance. By removing or combining such features, we aimed to reduce redundancy and improve model efficiency.
  - **Highly Correlated Onus Attributes:** The correlation analysis revealed no highly correlated pairs for onus attributes. Hence, no further action was required for this group.
  - **Highly Correlated Bureau Attributes:** A total of 55 highly correlated pairs were found within bureau attributes. Some examples include:
    - bureau_enquiry_10 and bureau_enquiry_20 with a correlation of 0.856362
    - bureau_enquiry_23 and bureau_enquiry_30 with a correlation of 0.855700
  - **Cluster Analysis for Bureau Attributes:** As with transaction attributes, we applied clustering for bureau attributes and replaced features within each cluster with their mean value. Example clusters include:
    - Cluster 1: bureau_440, bureau_443, Avg Correlation: 0.995590
    - Cluster 2: bureau_314, bureau_234, bureau_194, Avg Correlation: 0.959076
  - **Highly Correlated Transaction Attributes:** A total of 840 highly correlated pairs were identified among transaction attributes. Some examples include:
    - transaction_attribute_79 and transaction_attribute_81 with a correlation of 1.0
    - transaction_attribute_191 and transaction_attribute_525 with a correlation of 1.0
    - transaction_attribute_310 and transaction_attribute_651 with a correlation of 0.800860
  - **Cluster Analysis for Transaction Attributes:** We used graph-based clustering to group highly correlated features together and replaced them with their mean value within each cluster. Example clusters include:
    - Cluster 1: transaction_attribute_152, transaction_attribute_485, . . . , Avg Correlation: 0.992928

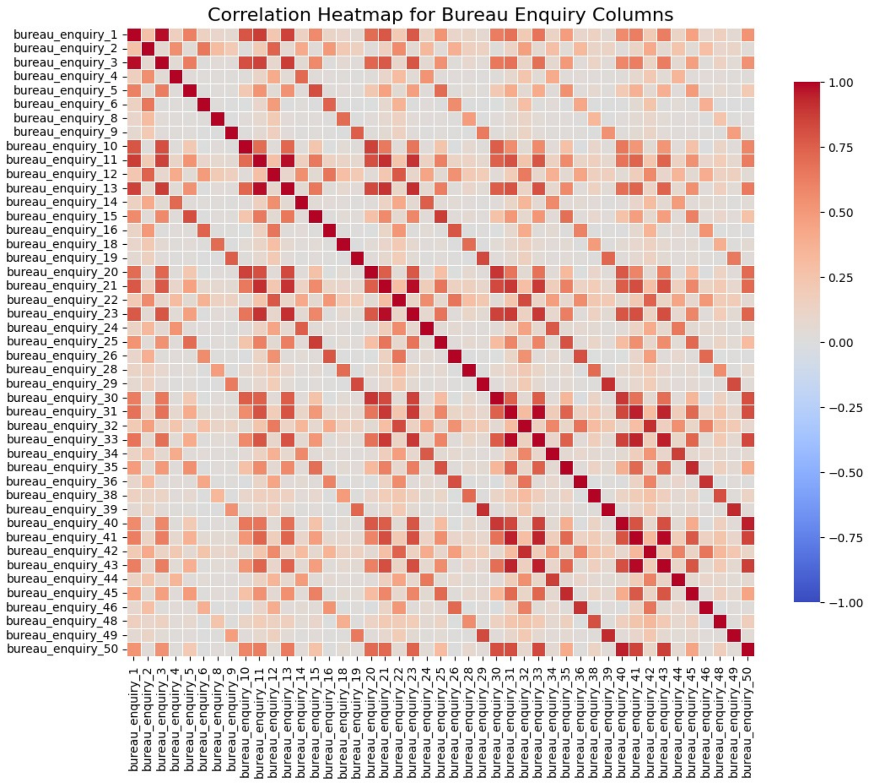

Figure 1: Correlation Matrix for Bureau Attributes

    * Cluster 2: transaction_attribute_638, transaction_attribute_556, . . . , Avg Correlation: 0.991590

- **Dimensionality Reduction:** After handling the correlation between features, we moved to reduce the total number of features in the dataset. By removing redundant features and replacing highly correlated features with their cluster mean, we effectively reduced the dimensionality of the dataset. This was done as follows:
  - **Initial Feature Count:** 1215 features
  - **Final Feature Count After Preprocessing:** 825 (We could reduce them even further to only 500 features by setting the threshfold .80 but we decided not to drop too many features and set correlation threshhold as 0.85)
  - The reduction was accomplished by:
    * Removing columns with all zeros
    * Replacing highly correlated features with their respective cluster mean values
- **Final Preprocessed Dataset:** The final dataset, now reduced in dimensionality, was cleaned and ready for model training. The preprocessing steps ensured that the dataset no longer contained irrelevant features or redundant information. It was now in an optimal state for training machine learning models, with all missing values handled and multicollinearity addressed.

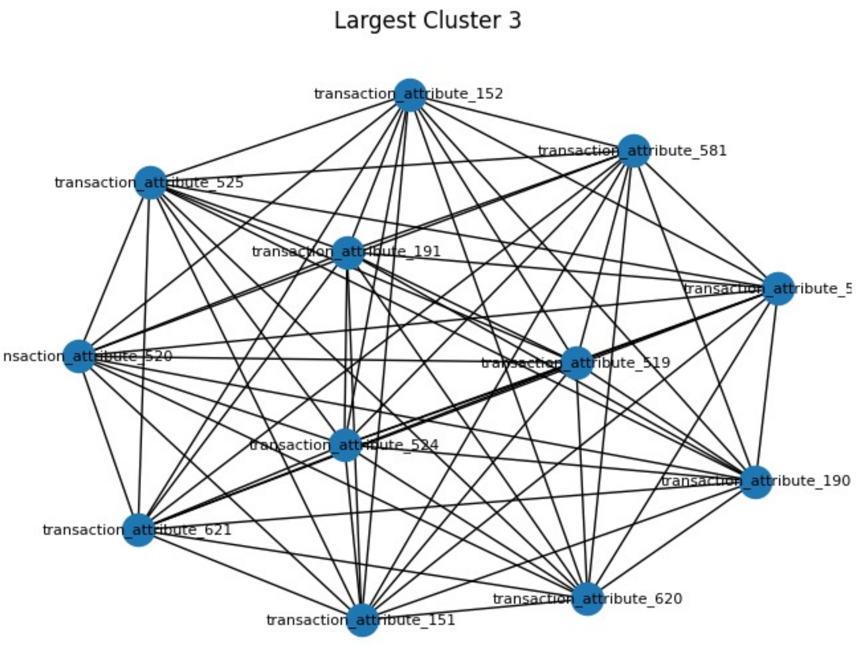
Largest Cluster 3
transaction_attribute_152
transaction_attribute_581
transaction_attribute_525
transaction_attribute_191
transaction_attribute_5
nsaction_attribute_520
transaction_attribute_519
transaction_attribute_524
transaction_attribute_190
transaction_attribute_621
transaction_attribute_620
transaction_attribute_151

Figure 2: Visualising the correlated cluster 3

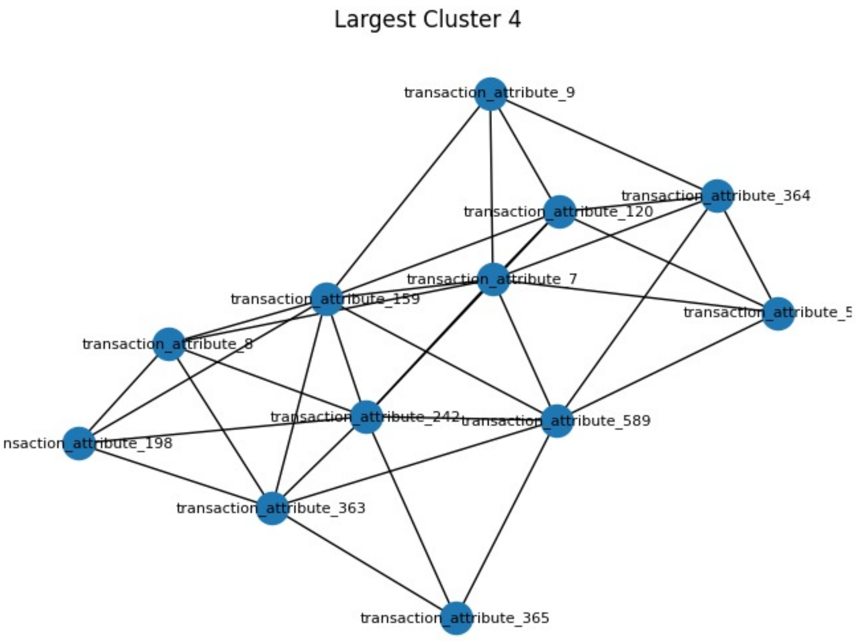
Largest Cluster 4
transaction_attribute_9
transaction_attribute_364
transaction_attribute_120
transaction_attribute_7
transaction_attribute_159
transaction_attribute_5
transaction_attribute_8
transaction_attribute_242
transaction_attribute_589
nsaction_attribute_198
transaction_attribute_363
transaction_attribute_365

Figure 3: Visualising the correlated cluster 4

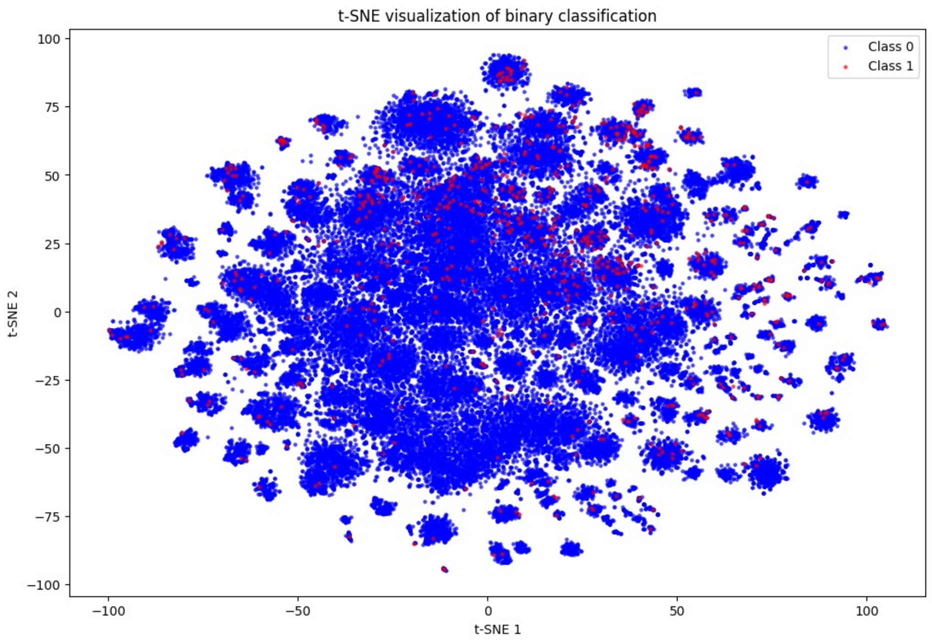

Figure 4: TSNE Plot of the Data

3 Model Selection and Approach

3.1 Choice of Model After observing results of multiple Ensemble models, We observed that different models behaved differently for our dataset, and shall be almost equally responsible for determining our predictions. ALso Neural Networks, were checked separately and trained by sampling Bad Flag 0's and 1's from data and the approach is thus discussed thoroughly in next section. This was done to solve the problem of overfitting of Neural Networks and tending more towards 0's in bad flags, being extremely high relatively compared to 1's present in dataset. From the figure 4 (TSNE Plot), we can infer the data is skewed with no clear distinction between the classes . So we definitely can use techniques like SMOTE (Synthetic Oversampling) , TOMEK or ADASYN because these techniques would just add noise due to no clear distinction between the Class 0 and Class 1 and would create problems instead of solving them. We also appliead this techniques but they just resulted in very poor F1 scores. So , we have to use either Undersampling or Oversampling to make our dataset balanced so that model doesnt show any bias towards a particular class.

3.2 Evaluation Metrics We evaluated the performance and quality of our models using key metrics such as F1 Score, AUC (Area Under the Curve), and ROC (Receiver Operating Characteristic) curves to assess their overall classification ability. In addition to these, we also calculated metrics like accuracy and precision to gain deeper insights into the model's predictive capabilities .

4 Model Implementation

So to solve the problem of High Noised and Imbalanced Dataset, having extremely high target 0's then 1's, we tried a variety of methods to solve it, including a SMOTE+ TOMEK algorithm, but could not solve our problem efficiently. So We separated our dataset of target 0's and 1's separately and divided them into some subparts each by bagging randomly at each step, and trained different Neural Networks on our Dataset, such that each model of NN gets familiarized with some of the data, does not overfit with 0's, and identify's defaulters(1's) properly. We store each model's Output separately for given Test set separately.

Now we also want our outputs from different ensemble models which are ought to perform good on our dataset. The proposed approach implements a multi-generation Random Forest training pipeline, where predictions from models in earlier generations serve as additional features for subsequent generations. Initially, balanced data groups are created to address class imbalance, and multiple classifiers (Random Forest, Gradient Boosting, Logistic Regression, LightGBM, XGBoost) are trained per generation. The process is repeated over `n_generations`, with custom configurations for test size, splits, and estimators to optimize each stage. Each generation produces models and a prediction DataFrame with probability scores, which are used as inputs for the next generation, enabling iterative refinement and improved feature representation for the classification task. This strategy enhances predictive accuracy while effectively managing class imbalance across generations.

This approach is grounded in the principle that repeatedly sampling the data a large number of times drives the probabilities to converge toward their true values, ultimately yielding accurate and reliable probability estimates.

So getting results from all our models multiple times does the required task. We combine our results for all possible probabilities using Weighted Mean Method which finally give our Required Probabilities.

5 Insights and Observations

- **Handling Missing Values:** The dataset contained many NaN values, which were replaced with zeros to maintain data integrity.

- **Feature Elimination:** 50 columns containing only zeros were removed entirely as they provided no useful information.

- **Class Imbalance:** The dataset was highly imbalanced, with approximately 1,300 target 1's and 95,000 target 0's.

- **Feature Engineering:** The dataset originally had over 1,200 features. Through correlation clustering, the feature space was reduced to around 800 columns, improving model performance and reducing noise.

- **Train-Test Split Before Sampling:** To ensure unbiased model evaluation, the data has to splitted into training and testing sets before applying any sampling techniques (e.g., undersampling or oversampling). This approach ensures that the testing set remains untouched by class imbalance handling strategies, maintaining its original

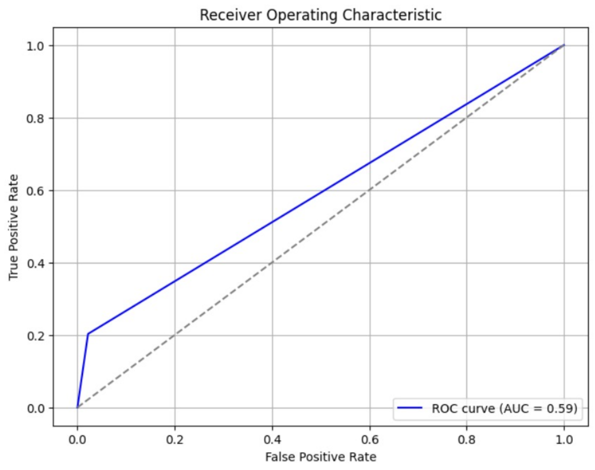

Figure 5: ROC-CURVE on TEST SET

distribution and allowing for an accurate assessment of the model's performance on unseen data. If this step is performed wrongly , then it will lead to misleading or biased metric scores .

- **High Noise and Scattering:** TSNE plots revealed that the data was noisy and highly scattered, with no clear distinction between target 0's and 1's, making synthetic sampling techniques like SMOTE, TOMEK, or ADASYN ineffective.
- **Sampling Strategy:** Due to the noise and imbalance, undersampling and oversampling techniques were adopted to balance the dataset effectively.
- **Model Performance:** Traditional machine learning algorithms, such as Random Forest, XGBoost, and LightGBM, performed poorly when used individually. However, ensemble models combining multiple classifiers yielded significantly better results.

6 Final Model Predictions and Output

6.1 Model on Validation Data Explain how the model was applied to the validation data and describe the predictions generated.

6.2 Output Present the output in the required format: account numbers and predicted probabilities for the validation set.

7 Results

Accuracy: 96.72
Precision: 0.99 for 0's, 0.12 for 1's (ON TEST SET)
F1 Score: 0.974 for 0's, 0.2 for 1's (ON TEST SET)
ROC-AUC: 0.59

8 References

1. **Wang, H., Fan, C. (2023). Handling imbalanced datasets in machine learning: An overview.** *Journal of Big Data*. SpringerOpen. Available at: `https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00684-w`

2. **Lee, J., Kim, S. (2024). Effective methods for feature reduction in high-dimensional datasets.** *Journal of Big Data*. SpringerOpen. Available at: `https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-01059-5`

3. **Kortical Team. (n.d.). Dropping features early: Why less can be more in machine learning.** Kortical Blog. Available at: `https://kortical.com/notebooks/dropping-features-early`