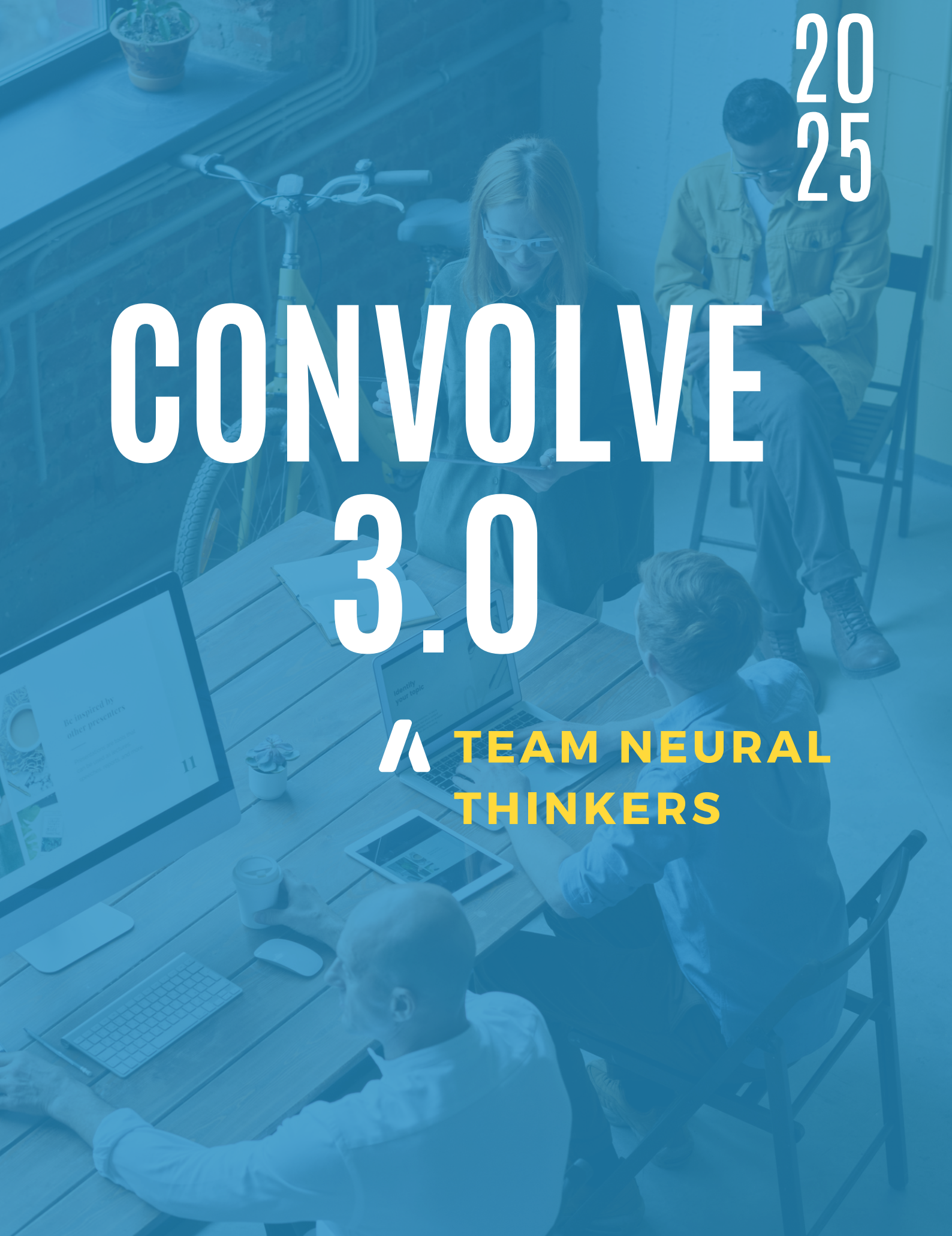


2025

CONVOLVE 3.0

TEAM NEURAL THINKERS

IDFC FIRST Bank

Table of Contents

Introduction

Bank Defaulter Risk Prediction Model Based on Account Scoring

The Behaviour Score is a sophisticated analytical tool designed to provide a comprehensive understanding of each customer's financial behavior. By leveraging this predictive model, the Bank can gain deep insights into customer profiles, enabling it to proactively identify potential risks and take preventive measures to mitigate them. This approach allows the Bank to dynamically adjust strategies such as credit limit modifications, personalized financial advice, and restructuring payment plans to better align with individual customer needs.

Beyond risk mitigation, the Behaviour Score serves as a cornerstone for enhancing customer relationships. By enabling more informed and data-driven decision-making, the model empowers the Bank to reward financially reliable customers with benefits such as lower interest rates or exclusive offers. These incentives foster trust and loyalty among customers while reinforcing a positive engagement experience. Furthermore, the Behaviour Score facilitates effective customer segmentation, ensuring that resources are allocated efficiently and profitably, optimizing the balance between risk management and business growth.

The implementation of the Behaviour Score represents a forward-thinking strategy in risk management and customer engagement. It embodies the Bank's commitment to innovation, allowing it to adapt swiftly to evolving market demands and customer behaviors. Through its ability to anticipate and address potential issues, the Behaviour Score aligns seamlessly with the Bank’s mission to deliver dynamic, responsive, and customer-centric services. By embracing this model, the Bank positions itself not only as a prudent financial institution but also as a trusted partner in its customers’ financial journeys, ensuring sustainable profitability and long-term customer satisfaction.

Dataset Description

Dataset Overview

Here’s an improved format for presenting the information in a report:
Dataset Overview

- Development Data (Dev_Data):
  - Number of Rows: 96,806
  - Number of Columns: 1,216
- Validation Data (Validation_Data):
  - Number of Rows: 41,792
  - Number of Columns: 1,215

Feature Groups

Dataset Features and Objective

The dataset comprises four primary types of features:

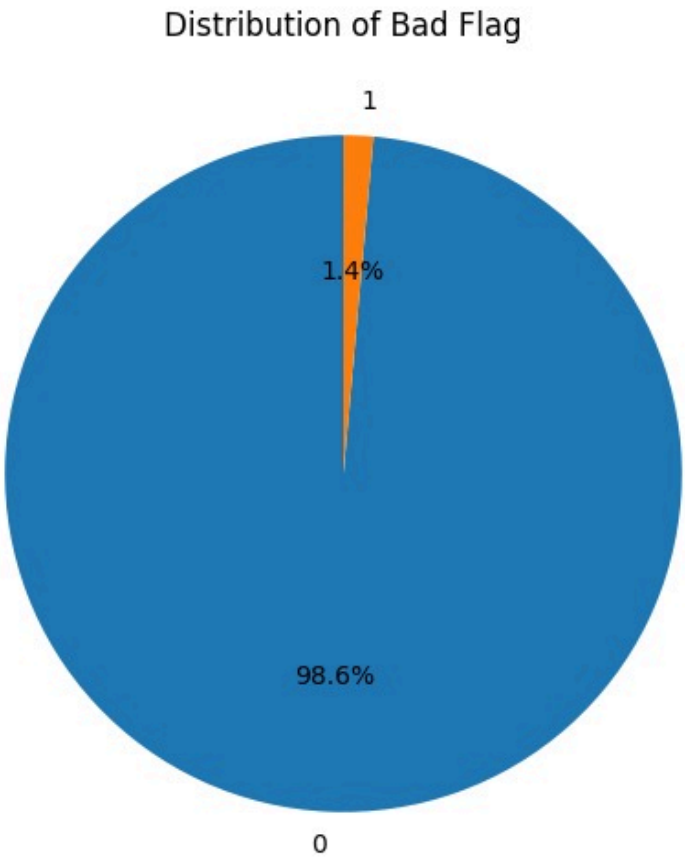
1. Onus Attributes
   - Attributes related to credit limits and customer obligations.
2. Transaction-Level Attributes
   - Details of transaction counts and values categorized by merchant type.
3. Bureau Tradeline-Level Attributes
   - Information about product holdings, repayment history, and delinquencies.
4. Bureau Inquiry-Level Attributes
   - Records of personal loan inquiries made in the past three months.

The development dataset includes all these features along with a BAD_FLAG, which indicates whether a customer has defaulted. This flag serves as the target variable for model training. The validation dataset, on the other hand, contains the same features but excludes the BAD_FLAG.

Objective

The primary goal is to leverage the features in the dataset to develop a predictive model that estimates the probability of default for the customers in the validation dataset. This enables effective risk assessment and informed decision-making.
Our dataset presents two significant challenges that require attention. Firstly, it contains numerous null values, which need to be handled effectively to ensure data integrity and reliability in our analysis. Secondly, the dataset exhibits class imbalance, with a predominance of the 0 class, which could bias the model's performance. Addressing these issues is crucial for building a robust and fair predictive model.

Challenges

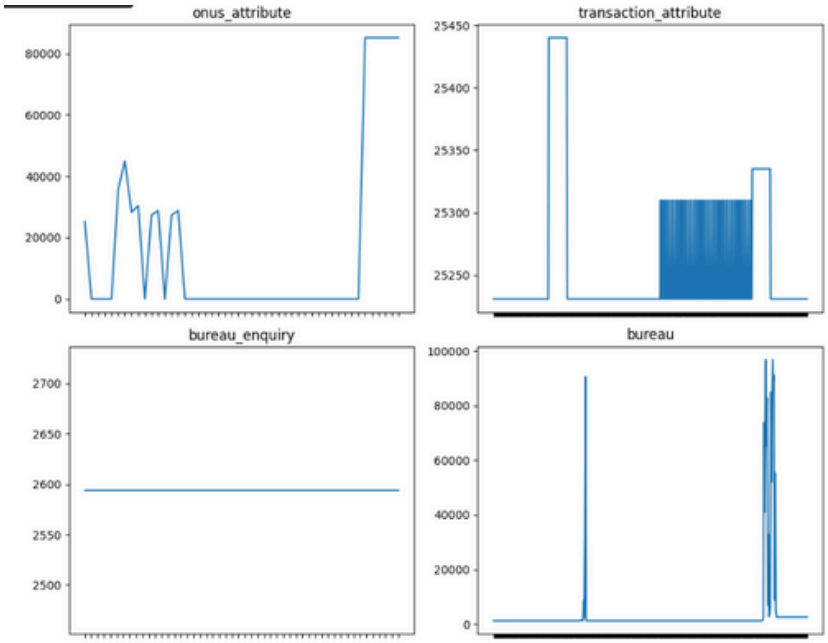


We encountered a dataset that was raw and sparse, containing a significant number of NaN values and many irrelevant features based on feature importance. Additionally, there was considerable class imbalance, with the majority of labels being 0 for the bad_flag. To address this, we first managed the NaN values by reducing columns and imputing them with the median and mode. Next, we tackled the class imbalance by assigning more weight to the label 1 during model training. We also addressed this issue by randomly selecting five equal samples of the data, training models on each, and then combining those models to achieve improved predictions.

Methodology

Handling Null Values

Visualization of NaN Values Across All Four Feature Types



Removing columns :

- To clean our data effectively, we started by eliminating columns with a high proportion of null values. Specifically, we applied a threshold of 30%, removing any columns where more than 30% of the values were missing. This step was critical in improving the quality of our dataset. Following this, we discarded columns with only a single unique value, as such columns offer no variability or useful information for the analysis.

```
threshold = len(df) * 0.3
columns_to_drop = df.columns[df.isnull().sum() > threshold]
```

- To clean our data, we began by eliminating columns that contained only a single unique value. These columns were removed because they offer no variability and do not contribute meaningful information to the analysis. A feature with only one unique value would result in no distinction between observations, making it ineffective for predictive modeling. By removing these uninformative columns, we ensured that the dataset contained only features with sufficient variation, which are necessary for building robust models.

```
filtered_df_nunique = df_nunique[df_nunique == 1]
columns_with_one_unique_value = filtered_df_nunique.index
```

Identifying Categorical Columns:

```
def get_categorical_cols_df(df, range_threshold):
    categorical_data = []
    for col in df.columns:
        nunique = df[col].nunique()
        max_val = df[col].max()

        if nunique == max_val + 1 and nunique <= range_threshold:
            categorical_data.append([col, nunique])  # Append column name and nunique

    # Create DataFrame
    categorical_df = pd.DataFrame(categorical_data, columns=['Categorical Column', 'Nunique'])

    return categorical_df
```

In this process, we identified categorical columns using the following logic: if the number of unique values (nunique) of a column is equal to the maximum number of columns plus one, we considered it categorical. To further refine the selection, we set a threshold of 25 unique values to avoid mistakenly categorizing columns like "bureau_1" as categorical. This approach serves as a general rule, though there could be edge cases where it may not apply perfectly. Nevertheless, it has proven effective in most instances for distinguishing between categorical and continuous variables.

Here, we have the no. of columns corresponding to nunique , get through this logic.

```
categorical_df['Nunique'].value_counts()
```

Nunique	count
2	227
7	10
4	7
3	5
9	4
5	3
6	3
16	2
8	2
13	2
14	1
22	1
20	1
12	1
10	1
15	1

Imputing Nan Values :

To handle the missing values, we first categorized the feature columns into two distinct groups: binary and continuous. For the continuous features, we imputed the missing values using the median of each respective column, ensuring that the central tendency of the data was preserved. For the binary features, we filled the missing values with the mode (most frequent value) of each column, maintaining the consistency of the categorical nature of these features. This approach helped to retain the integrity of the dataset while addressing the issue of missing data effectively.

```
for column in df.columns:
    if column in categorical_columns:
        df[column] = df[column].fillna(df[column].mode()[0])
    else:
        df[column] = df[column].fillna(df[column].median())
```

Handling Imbalance

Manual Class Weight Assignment:
Assigned class weights manually to penalize the model more for misclassifying the minority class. This encourages the model to focus on correctly predicting the minority class, which balances the impact of the imbalanced data.

Dataset Splitting and Combination:

- Divided the dataset into 5 parts:
- Class 0: 96,000 instances
- Class 1: 1,500 instances
- Split the class 0 instances into 5 IID parts.
- For each of the 5 parts, combined the split class 0 with class 1 to create a balanced dataset for training.
- This strategy generated 5 distinct datasets, each used for training, ensuring a more varied and balanced representation of both classes.

DIMENSIONALITY REDUCTION

We experimented with several dimensionality reduction techniques, including the Chi-Square test, ANOVA, and Principal Component Analysis (PCA), to reduce the number of features and improve model efficiency. However, these methods led to a decrease in performance metrics, particularly Precision-Recall AUC (PR AUC). The reduction in performance suggested that these dimensionality reduction techniques diminished the model's ability to effectively classify the minority class. In particular, PCA, which projects data into a lower-dimensional space, appeared to obscure key patterns that were critical for identifying the minority class, resulting in a less discriminative model. As a result, we found that dimensionality reduction negatively impacted the model's ability to capture the nuances of the imbalanced data.

Model Selection

LightGBM:

LightGBM (Light Gradient Boosting Machine) is an open source, distributed, high-performance implementation of gradient boosting developed by Microsoft. LightGBM is designed to be faster and more efficient than traditional gradient boosting methods. It excels in handling large datasets with millions of instances and features while using less memory. LightGBM is highly optimized for speed, making it one of the most preferred libraries for large-scale machine learning tasks.
One of the key innovations of LightGBM is its use of leaf-wise growth instead of the traditional level-wise growth, which leads to deeper trees and more accurate models. Additionally, LightGBM supports histogram-based learning, which allows it to efficiently handle continuous features and large-scale datasets. It also supports categorical features natively, reducing the need for preprocessing.
LightGBM is widely used in business applications such as click-through rate prediction, fraud detection, and recommendation systems due to its high scalability and speed. It also boasts a strong reputation for handling imbalanced datasets and delivering state-of-the-art performance in Kaggle competitions.

XGBoost:

XGBoost (Extreme Gradient Boosting) is an open-source machine learning algorithm developed by Tianqi Chen. It is an optimized and efficient implementation of gradient boosting that is designed for speed and performance. XGBoost can handle both regression and classification tasks and is known for its scalability, speed, and high accuracy. It works well with both structured and unstructured data and is particularly effective in handling large datasets with many features.
XGBoost leverages techniques such as tree boosting, regularization, and parallelization, which help the model achieve higher performance compared to traditional gradient boosting. By implementing early stopping and pruning, it also ensures the model does not overfit, making it well-suited for large-scale datasets with complex patterns.
It has proven to be highly effective in a variety of applications such as classification, regression, ranking, and user-defined prediction tasks, with notable success in areas like finance, healthcare, and e-commerce.

Catboost:

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.
As discussed, the library works well with multiple Categories of data, such as
audio, text, image including historical data. "Boost" comes from gradient boosting machine learning algorithm as this library is based on gradient boosting library. Gradient boosting is a powerful machine learning algorithm that is widely applied to multiple types of business challenges like fraud detection, recommendation items, forecasting and it performs well also.
It can also return very good result with relatively less data, unlike DL models that
need to learn from a massive amount of data.

Evaluation Metrics

To evaluate the performance of the proposed approach on the imbalanced dataset, the following metrics were utilized:

Precision-Recall AUC (PR AUC)

1. The primary metric used to assess the model's effectiveness was the Precision-Recall Area Under the Curve (PR AUC). This metric is particularly well-suited for imbalanced datasets because it emphasizes the model's performance on the minority class (label 1). PR AUC focuses on two critical aspects:
2. Precision: The proportion of correctly predicted positive cases (label 1) out of all cases predicted as positive. High precision indicates fewer false positives.
3. Recall: The proportion of correctly predicted positive cases out of all actual positive cases. High recall ensures fewer false negatives.
4. The PR AUC combines these two measures across different threshold levels, providing a robust evaluation of the model's ability to identify the minority class effectively. This metric is ideal in scenarios were identifying the minority class correctly is more important than overall accuracy.

Baseline PR AUC = Number of Positive Samples / Total Samples

The baseline PR AUC, calculated as the ratio of positive samples to total samples, provides a simple benchmark for evaluating model performance. In this case, a baseline PR AUC of 0.014 suggests that a naive classifier that always predicts the positive class would achieve this level of performance. This baseline serves as a crucial reference point, as any model with a PR AUC lower than this baseline indicates that it performs worse than a simple random guess.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 Score (Class 1)

- As a secondary metric, the F1 Score for the minority class (label 1) was calculated. The F1 score is the harmonic mean of precision and recall, offering a single metric that balances the trade-off between these two. A high F1 score indicates that the model performs well in predicting the minority class without overproducing false positives or false negatives.
- The F1 score is particularly useful in determining how well the model balances precision and recall in real-world scenarios where both metrics are crucial for reliable predictions.
- Given the extreme imbalance in the dataset (0: 95,434; 1: 1,372), traditional metrics like accuracy are not reliable, as they can be dominated by the majority class. Instead, PR AUC and F1 Score provide a more focused assessment of the model's performance on the minority class, ensuring that the approach effectively addresses the imbalance and provides meaningful predictions.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

MODEL TRAINING

LightGBM (Class_weight)

To address class imbalance and optimize PR AUC, we train a LightGBM model with class weights. This technique assigns higher importance to minority classes during training, preventing the model from being biased towards the majority class. By focusing on the PR AUC metric, we prioritize models that effectively identify positive instances while maintaining high precision, crucial for imbalanced datasets.

```
model = lgb.LGBMClassifier(class_weight={0: 1, 1: 15}, random_state=42)
model.fit(X_train, y_train)
```

The model exhibits a high ROC AUC (0.8177), indicating good overall class separation, which is a sign of the model's ability to distinguish between the classes. However, the PR AUC (0.0740) is significantly lower, suggesting difficulties in maintaining high precision while achieving high recall, particularly due to the class imbalance. Compared to the baseline PR AUC of 0.014, the model shows an improvement, though it still struggles to effectively identify and classify the minority class (class 1). This discrepancy likely arises from the model prioritizing the majority class, leading to numerous false positives for the minority class. While ROC AUC reflects overall class separation, PR AUC reveals that the model still faces challenges in accurately predicting the minority class, as evidenced by the substantial gap between the ROC and PR AUC scores.

```
--- LightGBM ---
ROC AUC: 0.8177
PR AUC: 0.0740
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     38174
           1       0.13      0.11      0.12       549

    accuracy                           0.98     38723
   macro avg       0.56      0.55      0.55     38723
weighted avg       0.98      0.98      0.98     38723
```

Comparing LightGBM XGBoost CatBoost

When comparing the performance of LightGBM, XGBoost, and CatBoost on this imbalanced dataset, LightGBM stands out as the best performer in terms of both PR AUC (0.0740) and F1 score (for class 1, 0.12). While all three models achieve high ROC AUC scores, indicating good overall class separation, the PR AUC values reveal that LightGBM handles the minority class (class 1) better than the other models. XGBoost and CatBoost show notably lower PR AUC values (0.0392 and 0.0529, respectively), and their F1 scores for the minority class are also quite low (0.08 and 0.08). These results suggest that LightGBM is more effective at balancing precision and recall for the minority class, making it the preferred model for handling the challenges posed by class imbalance in this case. Despite its lower PR AUC compared to the ROC AUC, LightGBM is the best choice for improving performance on the minority class, as evidenced by its higher PR AUC and F1 score in this comparison.

```
--- LightGBM ---
ROC AUC: 0.8177
PR AUC: 0.0740
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     38174
           1       0.13      0.11      0.12       549

    accuracy                           0.98     38723
   macro avg       0.56      0.55      0.55     38723
weighted avg       0.98      0.98      0.98     38723

--------------------
--- XGBoost ---
ROC AUC: 0.7311
PR AUC: 0.0392
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     38174
           1       0.06      0.10      0.08       549

    accuracy                           0.96     38723
   macro avg       0.52      0.54      0.53     38723
weighted avg       0.97      0.96      0.97     38723

--------------------
--- CatBoost ---
ROC AUC: 0.7695
PR AUC: 0.0529
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     38174
           1       0.09      0.07      0.08       549

    accuracy                           0.98     38723
   macro avg       0.54      0.53      0.54     38723
weighted avg       0.97      0.98      0.98     38723
```

Ensemble of LightGBM XGBoost CatBoost

The ensemble model, while maintaining a competitive ROC AUC of 0.8024, shows a slight decrease in PR AUC (0.0582) compared to the single LightGBM model. This suggests that while the ensemble approach might have improved overall class separation, it may have slightly compromised the model's ability to maintain high precision while achieving high recall, particularly for the minority class. This is further evidenced by the classification report, where the precision and recall for the minority class (class 1) remain relatively low, indicating that the ensemble model still struggles to accurately identify and classify instances of the minority class.

```
[LightGBM] [Info] Start training from score -1.534351
Ensemble ROC AUC: 0.8024
Ensemble PR AUC: 0.0582

Ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     38174
           1       0.11      0.07      0.08       549

    accuracy                           0.98     38723
   macro avg       0.55      0.53      0.54     38723
weighted avg       0.97      0.98      0.98     38723
```

Grouped Resampling (5 LightGBM Ensemble)

The Grouped Resampling (5 LightGBM Ensemble) technique addresses class imbalance by splitting the majority class (class 0) into five independent and identically distributed (IID) subsets, each of which is combined with the full minority class (class 1). This approach ensures that the model is exposed to more balanced data during training, allowing it to better learn the characteristics of the minority class. To further improve the robustness of the model, an ensemble of five LightGBM classifiers is used, each trained on one of these subsets, with predictions averaged for the final outcome. This technique helps improve the model's performance on the minority class without losing the information from the majority class. Additionally, StratifiedKFold is utilized to ensure that each fold maintains the same class distribution, improving model validation and helping reduce any bias that could arise from random sampling. By combining this approach with ensemble learning, the model is able to better generalize to unseen data, especially in imbalanced datasets, resulting in improved performance metrics such as PR AUC and F1 score for the minority class.

The ensemble of 5 LightGBM models with Grouped Resampling demonstrates superior performance compared to the single LightGBM model. While the ensemble maintains a high ROC AUC of 0.8285, it notably improves the PR AUC to 0.0851, a significant enhancement over the single model's PR AUC and even surpassing the baseline PR AUC. This improvement suggests that the ensemble effectively addresses the class imbalance issue, leading to better precision and recall for the minority class. The classification report further supports this, showing improved recall for the minority class (class 1) compared to the single model, indicating a more accurate identification of positive instances. This suggests that the Grouped Resampling technique effectively leverages the strengths of multiple LightGBM models, leading to a more robust and accurate prediction model, particularly for the challenging minority class in this imbalanced dataset.

```
Ensemble ROC AUC: 0.8285
Ensemble PR AUC: 0.0851

Ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     28630
           1       0.12      0.19      0.15       412

    accuracy                           0.97     29042
   macro avg       0.56      0.58      0.57     29042
weighted avg       0.98      0.97      0.97     29042
```

Model Limitations

Despite the use of advanced techniques like Grouped Resampling (5 LightGBM Ensemble) and StratifiedKFold, the model still exhibits a low F1 score for the minority class (1), highlighting its inability to accurately classify the minority class. The low F1 score indicates that while the model may achieve high precision for the majority class (0), it struggles to balance both precision and recall for the minority class. This could be due to several factors, such as the imbalance in the dataset, where the vast majority of instances belong to the dominant class, leading the model to prioritize it over the minority class. As a result, the model tends to produce false negatives (misclassifying actual minority instances as the majority class), which lowers recall, and thus, the F1 score.

VALIDATION DATA CLEANING & RESULTS

DATA CLEANING

To begin with, we clean our data by removing the columns that were eliminated in our development dataset. Next, we address the missing values: we fill the categorical features' NaN values with the mode and the continuous features' NaN values with the median.

PREDICTIONS

We predict the likelihood of default for our validation data by first applying a pretrained model to the development data, followed by using the model to forecast defaults on the validation data.

Behaviour Score Interpretation

1. The Behavior Score is a creditworthiness metric calculated using the formula:
2. Behavior Score = (1 - Predicted Probability of Default) × 100.
3. It ranges from 0 to 100, where a higher score indicates lower default risk and better customer behavior.
4. It ranges from 0 to 100, where a higher score indicates lower default risk and better.
5. 100: Best behavior, minimal default risk.
6. 0: Worst behavior, high default risk.

	account_number	behavior_score
0	100001	85.129777
1	100002	98.746376
2	100003	98.980152
3	100004	98.766968
4	100005	84.123598

Conclusion & Future Scope

Conclusion

Techniques to calculate the behavior score for credit card customers. The model demonstrated robust performance, as reflected by a high AUC (Area Under the Curve) score, indicating its strong ability to distinguish between defaulting and non-defaulting customers. However, the F1 score was relatively low, pointing to challenges in achieving a balance between precision and recall, likely due to the inherent class imbalance in the dataset. Overall, the project met its objective of providing a reliable risk assessment framework, enabling better portfolio management for the bank.

In conclusion, we developed a Behavior Score as a creditworthiness metric to assess the risk of default. This score, calculated as **Behavior Score = (1 - Predicted Probability of Default) × 100**, ranges from 0 to 100. A higher score indicates lower default risk and better customer behavior, while a lower score reflects higher default risk and poorer behavior. The score provides a clear and quantifiable measure, where 100 represents the best behavior with minimal default risk, and 0 represents the worst behavior with a high likelihood of default.

Future Scope

The developed framework holds significant potential for broader applications and enhancements. It can be extended to incorporate additional data sources, such as real-time transaction data or customer demographics, to improve its predictive accuracy and adaptability.Moreover, the framework can be adapted for other financial products, such as loans or mortgages, enabling a comprehensive risk assessment across multiple portfolios. Regular updates and re-calibration using new data will ensure the framework remains robust in dynamic market conditions. Lastly, integrating the framework into a real-time decision-making pipeline could enable proactive and personalized interventions, ensuring better customer management and portfolio profitability.