RevoBank's Paylater Initiative and Propensity Modeling

Satryo Sunu Prabowo

Colab Link

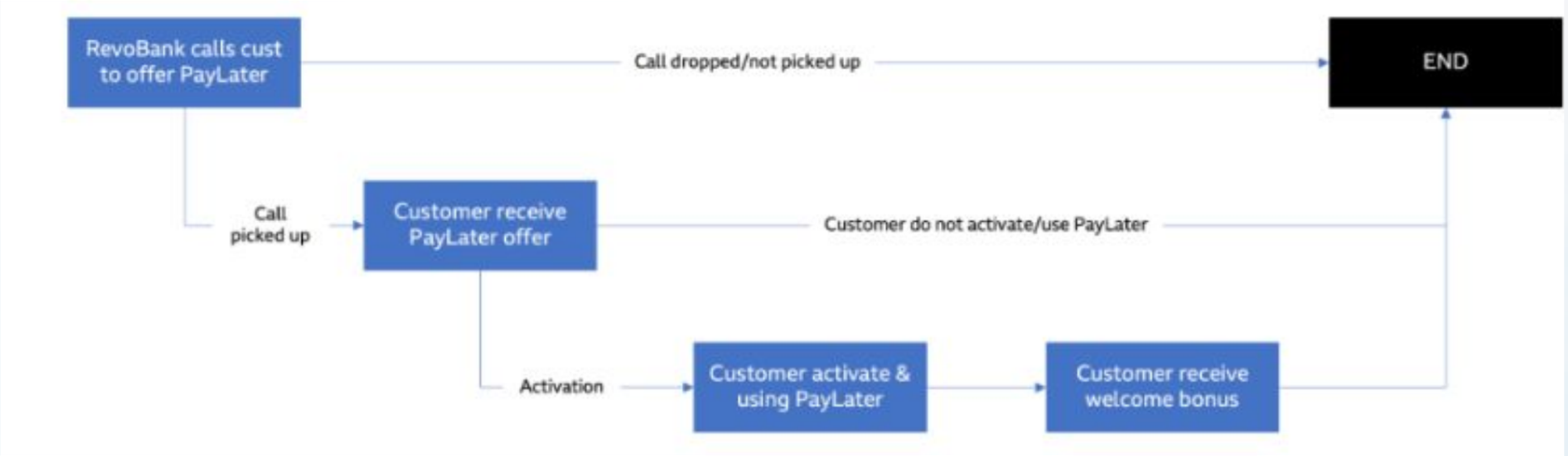
Business Background

RevoBank ran a pilot program offering an installment or PayLater feature in their bank. This pilot was successful in increasing credit card use. Now, they're looking to expand this program by reaching out to 30,000 potential customers who haven't used PayLater yet.

To target these customers effectively, they're looking to develop a model to predict who is most likely to use PayLater. This will allow them to target their outreach efforts and maximize the impact of their offer (1000 reward points for signing up and using PayLater).

Project Contact

Business Process



Propensity Model Objective

A plan has been devised to develop a propensity model. This model aims to predict which customers are most inclined to utilize the PayLater feature, facilitating targeted outreach efforts and maximizing the effectiveness of the promotion.

Defining data scope

Past: before 31 January 2023

Present: 31 January 2023

Future: February - April 2023

Data Preparation for Propensity Model

Feature Engineering

Created several new features/variables for Propensity Model analysis later on, here are 4 new variables that are created:

- **total_sales**: sum of avg_sales_L36M and cnt_sales_L36M
- **total_sales_promo**: sum of avg_sales_L36M_promo and cnt_sales_L36M_promo
- **promo_percentage**: avg_sales_promo divided by avg_sales
- **promo_response_rate**: cnt_sales_promo divide by cnt_sales

Data Preparation for Propensity Model

Encoding Categorical Data

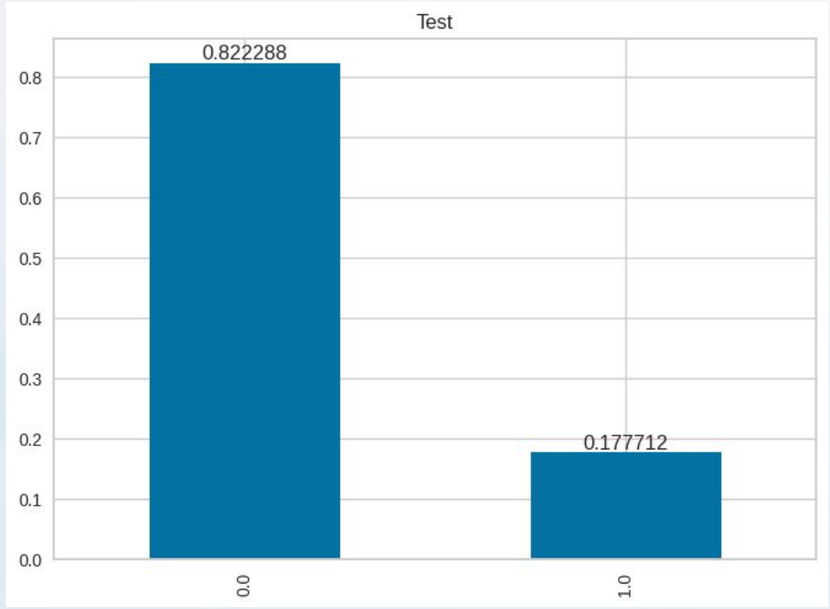
Several features need to be encoded for propensity model as this features are categorical data. Here are the lists of categorical data:

- Account Activity Level:
  - Account Activity Level X
  - Account Activity Level Y
  - Account Activity Level Z
- Customer Value Level:
  - Customer Value Level A
  - Customer Value Level B
  - Customer Value Level C
  - Customer Value Level D
  - Customer Value Level E

EDA for Training Data

Target Check Using Barchart

Data is split into 2 parts; 70% for training, 30% for testing. Let's use bar chart to check for target data in proportion to entire training and test population data.

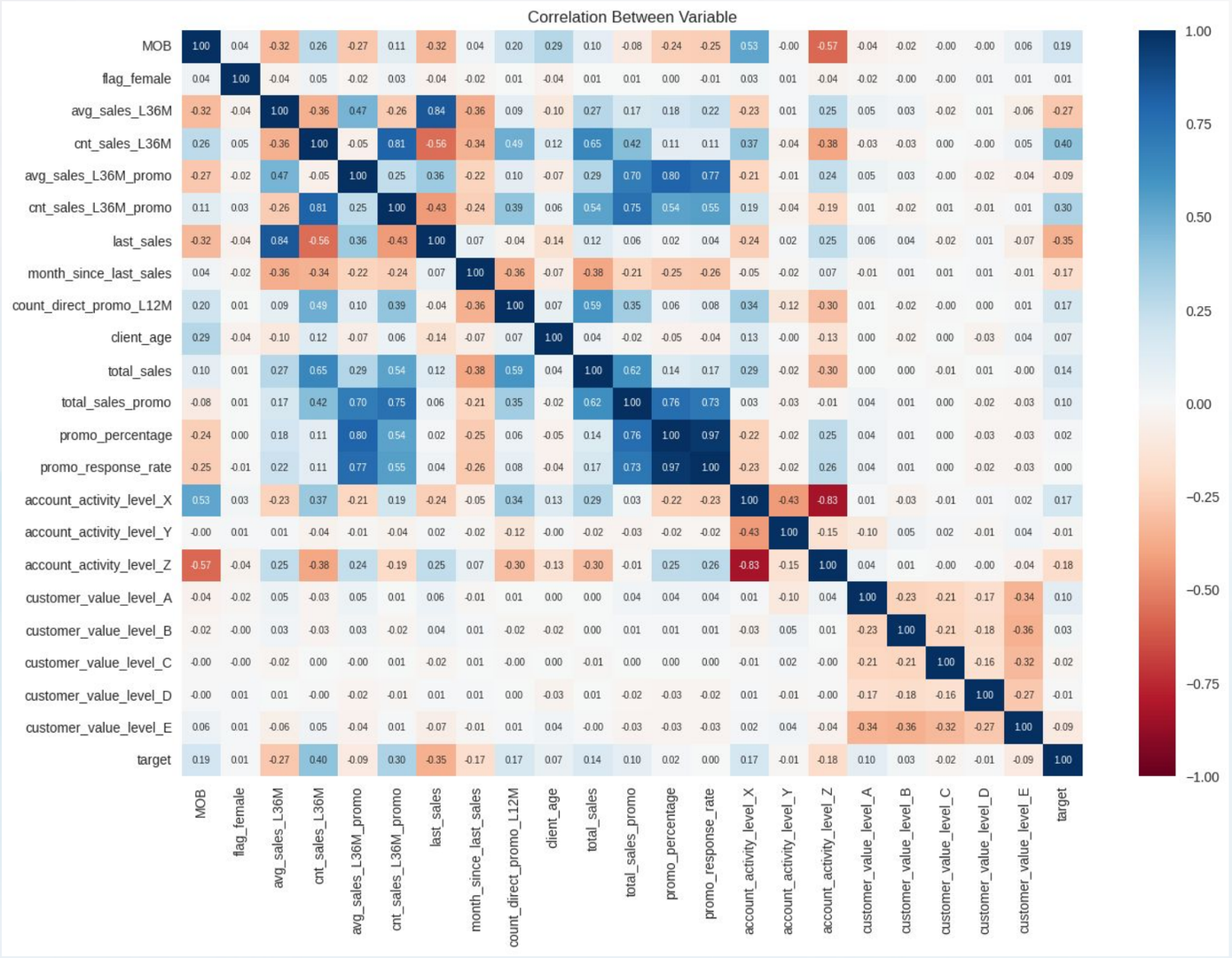


Both training and test data are **Class Imbalance**. In this case, the target data (1.0) occupies around **17%** of the dataset. If the training data is heavily skewed towards one class, the model might become biased towards predicting the majority class (0.0). This means the model might miss a significant portion of potential PayLater users (1.0).

Variable Removal using Correlation

We want to remove some variables due to:

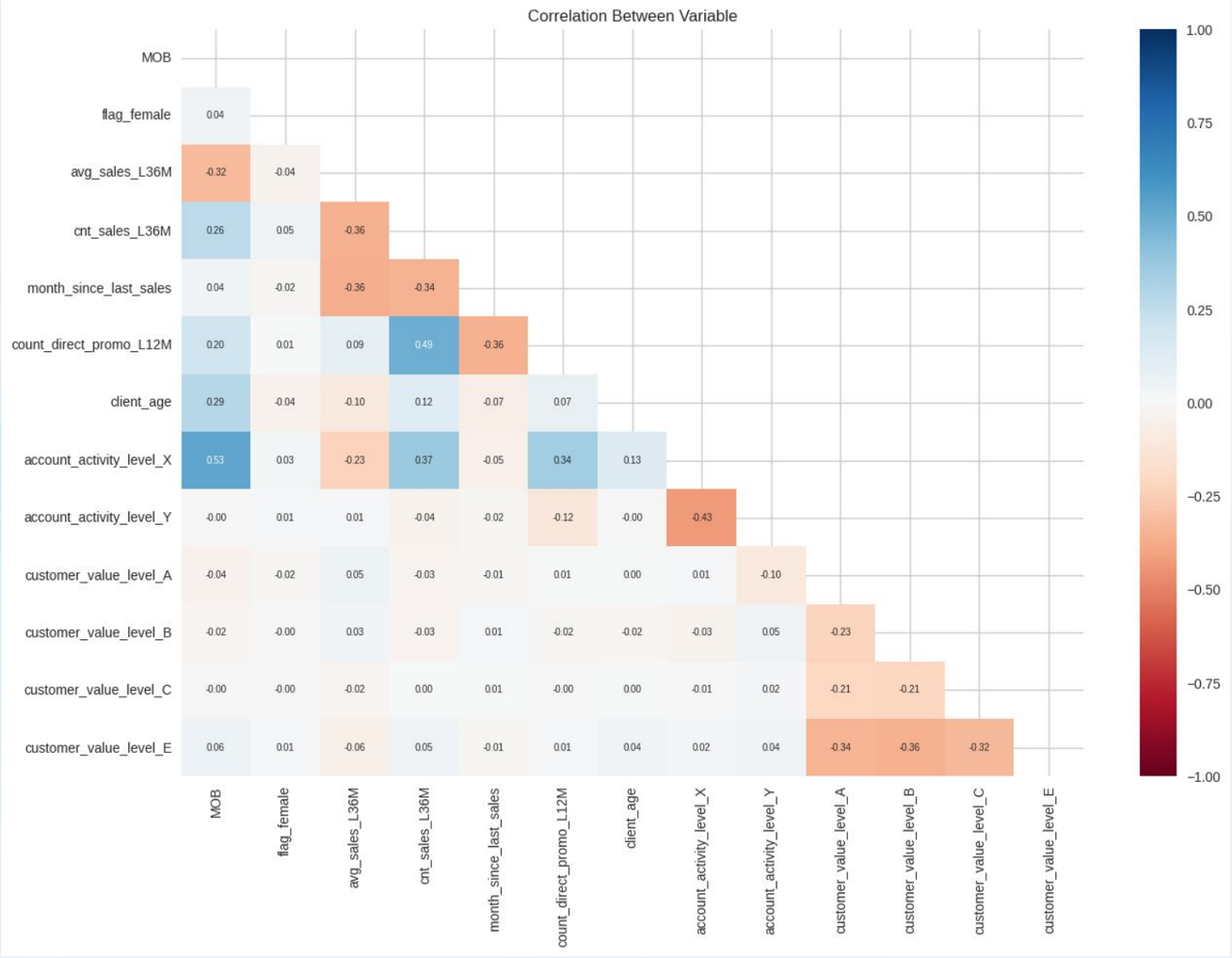
High correlation (>0.5) to avoid violating multicollinearity assumption, and **low correlation with target (<0.2)** to remove variables that has no significance to target.



Variable Removal using Correlation

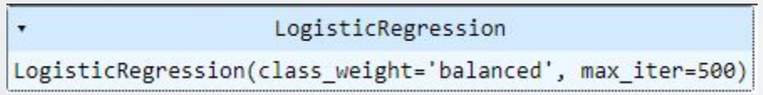
These are variables after removal

Please note that one variable “account_activity_level_X” still has correlation value of **0.53** to MOB. This will not affect the propensity model prediction significantly so we will leave it there.



Propensity Model Training & Testing

Propensity Model Training



Logistic Regression Model

Using Logistic Regression Model imported from library scikit-learn.

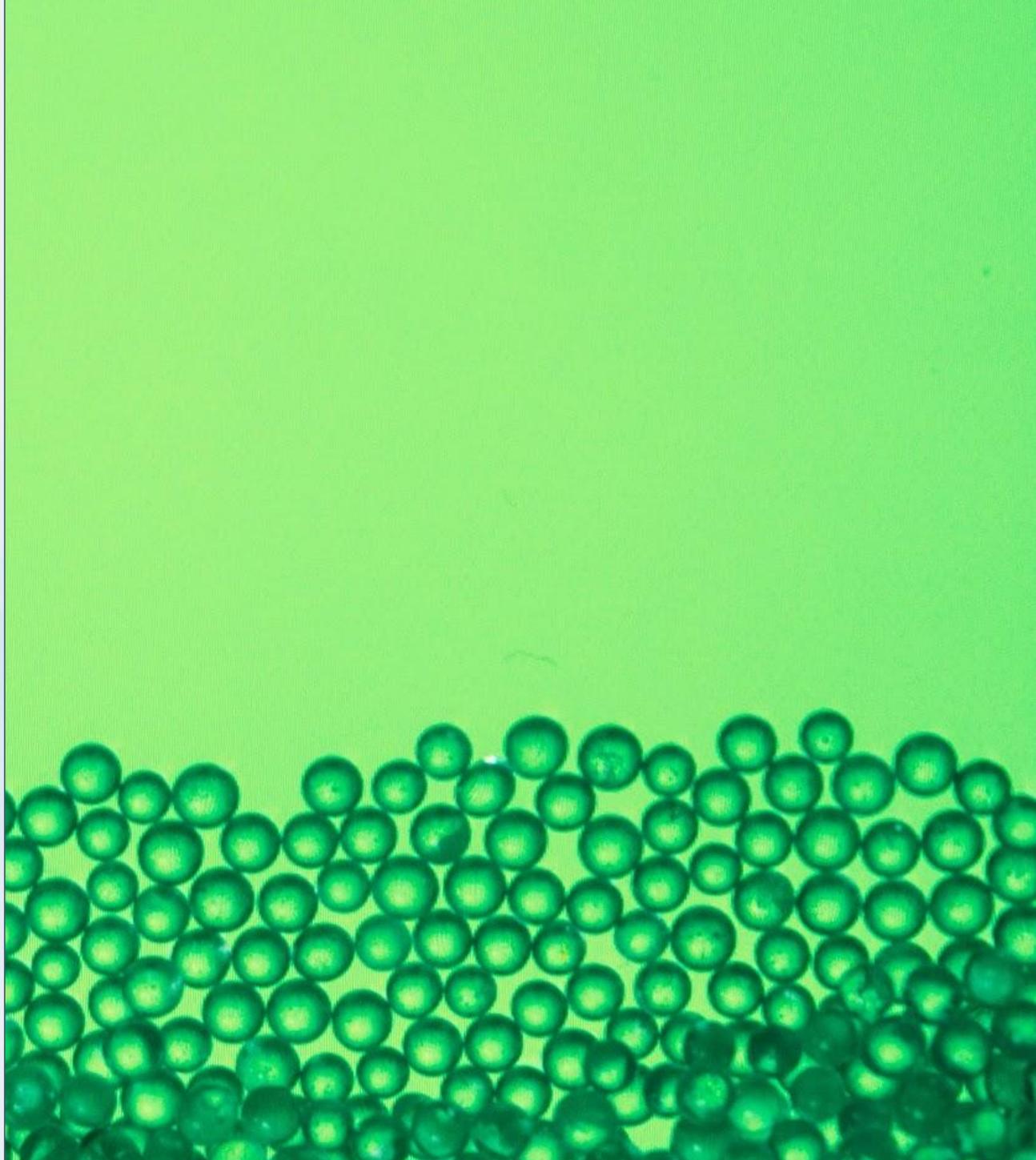
Class Weight

This parameter is set to **'balanced'**, which means that the logistic regression model will **automatically adjust the weights of the classes** inversely proportional to their frequencies. This is useful when **dealing with imbalanced datasets**, where one class has significantly fewer samples than the other. The model will give more weight to the minority class, helping it to be better represented in the training process.

Max Iteration

Default number is 100, this parameter sets the maximum number of iterations for the solver to converge. It is set to **500**, meaning that the optimization algorithm will iterate at most 500 times to find the optimal parameters.

Propensity Model Evaluation



Evaluation - Accuracy

```
[ ]  # Accuracy dr prediksi model dengan data training
     model.score(x_training_model, y_training)

     0.7729328749773837

[ ]  # Accuracy dr prediksi model dengan data test
     model.score(x_test_model, y_test)

     0.7676234698184888
```

Overall Performance:

Both training and test set accuracies are around **77%**. The training and test set accuracies are relatively close, indicating that the model's performance on unseen data is comparable to its performance on the training data. This suggests that the model is not severely overfitting.

Training Set Accuracy (0.7729):

The model correctly predicts around **77.29%** of the outcomes in the training data. A high training set accuracy indicates that the model has learned the patterns in the training data reasonably well.

Test Set Accuracy (0.7676):

The model correctly predicts around **76.76%** of the outcomes in the test data. It's important to note that the test set accuracy is slightly lower than the training set accuracy, which is a common scenario. This suggests that there may be some degree of **overfitting**, where the model performs well on the training data but struggles to generalize to new data.

While the **accuracies** are **reasonably high**, they don't provide the full picture of model performance. It's essential to consider other metrics, such as **precision**, **recall**, and **F1-score**, especially given the class imbalance mentioned earlier.
Further analysis, such as examining the model's **confusion matrix** or conducting cross-validation, can provide deeper insights into its performance and potential areas for improvement.

Evaluation - Confusion Matrix

True Negatives (Top-left cell): The value 7,075 represents the number of instances that were correctly predicted as negative (non-takers in this case).

False Negatives (Bottom-left cell): The value 425 represents the number of instances that were incorrectly predicted as negative (non-takers), when they were actually positives (takers).

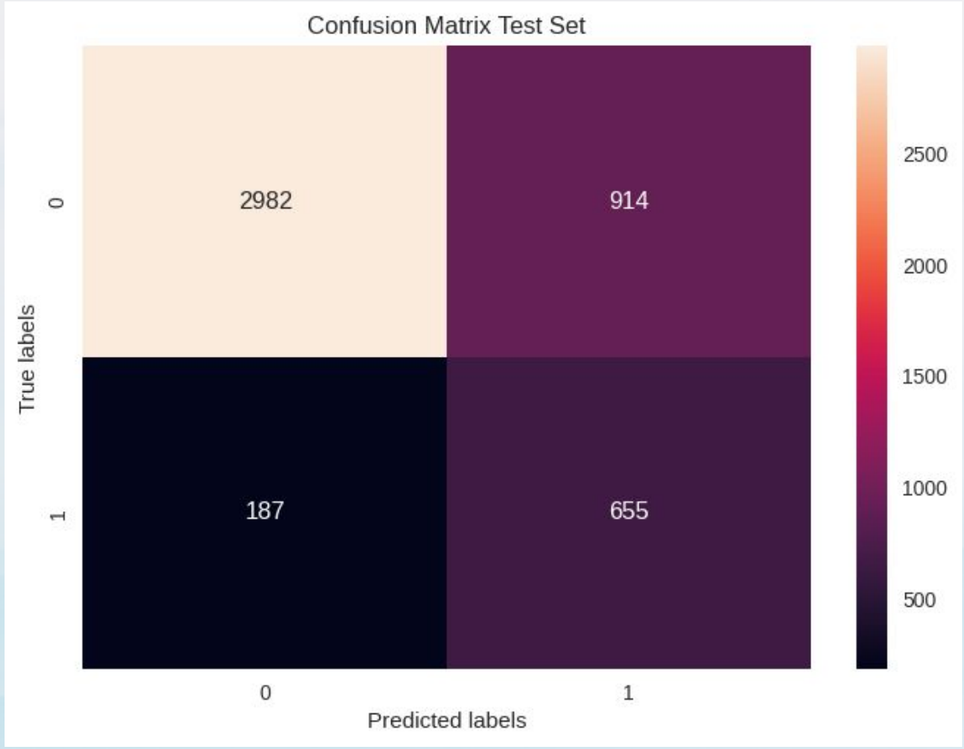
False Positives (Top-right cell): The value 2,085 represents the number of instances that were incorrectly predicted as positive (takers), when they were actually negative (non-takers).

True Positives (Bottom-right cell): The value 1,469 represents the number of instances that were correctly predicted as positive (takers)

Overall Performance:

The confusion matrix indicates that the model **performs well** in predicting the **majority class (non-takers)** but **struggles** with the **minority class (takers)**, likely due to the **class imbalance** in the training data. Addressing this issue and improving the model's performance on the minority class could lead to better overall performance.

Evaluation - Confusion Matrix



Overall, the confusion matrices reveal that while the model performs better in identifying the minority class (takers) on the test set, there is still room for improvement, particularly in reducing the false positive rate and addressing potential overfitting issues on the training data.

Similar to the training set, the test set also exhibits a **class imbalance**, with a significantly higher number of negative instances (non-takers) compared to positive instances (takers).

While the model still struggles with the minority class (takers) on the test set, the performance appears to be **worse** compared to the training set, with a higher false negative rate (914 vs. 2,085) and a lower true positive rate (655 vs. 1,469) relative to the dataset sizes.

The differences in the confusion matrix values between the training and test sets indicate that the **model's performance may vary** on unseen data. However, the overall pattern suggests that the model generalizes **reasonably well** to the test set, with improvements in some areas (e.g., lower false positive rate) and potential overfitting in others (e.g., lower true positive rate relative to dataset size).

Evaluation - Classification Report

	precision	recall	f1-score	support
0.0	0.94	0.77	0.85	9160
1.0	0.41	0.78	0.54	1894
accuracy			0.77	11054
macro avg	0.68	0.77	0.69	11054
weighted avg	0.85	0.77	0.80	11054

Training

	precision	recall	f1-score	support
0.0	0.94	0.77	0.84	3896
1.0	0.42	0.78	0.54	842
accuracy			0.77	4738
macro avg	0.68	0.77	0.69	4738
weighted avg	0.85	0.77	0.79	4738

Test

Classification report of both training and test data are quite similar, with accuracy **77%.**

Insight from the test data:

For non-takers (0.0):

- Precision is 0.94, which means that when the model predicts an instance as a non-taker, it is **correct 94% of the time.**
- Recall is 0.77, indicating that the model correctly **identifies 77% of the actual non-taker** instances.
- F1-score is 0.84, which is a **balanced measure of precision and recall** for this class.

For takers (1.0):

- Precision is 0.42, which means that when the model predicts an instance as a taker, it is **correct only 42%** of the time.
- Recall is 0.78, indicating that the model **correctly identifies 78% of the actual taker** instances.
- F1-score is 0.54, which is **relatively low due to the low precision** for this class.

Evaluation - Classification Report

	precision	recall	f1-score	support
0.0	0.94	0.77	0.85	9160
1.0	0.41	0.78	0.54	1894
accuracy			0.77	11054
macro avg	0.68	0.77	0.69	11054
weighted avg	0.85	0.77	0.80	11054

Training

	precision	recall	f1-score	support
0.0	0.94	0.77	0.84	3896
1.0	0.42	0.78	0.54	842
accuracy			0.77	4738
macro avg	0.68	0.77	0.69	4738
weighted avg	0.85	0.77	0.79	4738

Test

Comparison with the training data:

- The precision, recall, and F1-score values for the non-takers class (0.0) are very similar between the training and test data, indicating consistent performance for this class.
- For the takers class (1.0), the precision value is slightly higher in the test data (0.42) compared to the training data (0.41), but the recall value remains the same (0.78).
- The F1-score for takers is lower in the test data (0.54) compared to the training data (0.78), likely due to the lower precision in the test data.

Overall, the model seems to perform consistently well in identifying non-takers (0.0) in both the training and test data. However, it struggles with accurately predicting takers (1.0), as evident from the **lower precision values** in both datasets.

This discrepancy in performance between the two classes could be due to several factors, such as class imbalance (fewer taker instances).

Evaluation - ROC-AUC

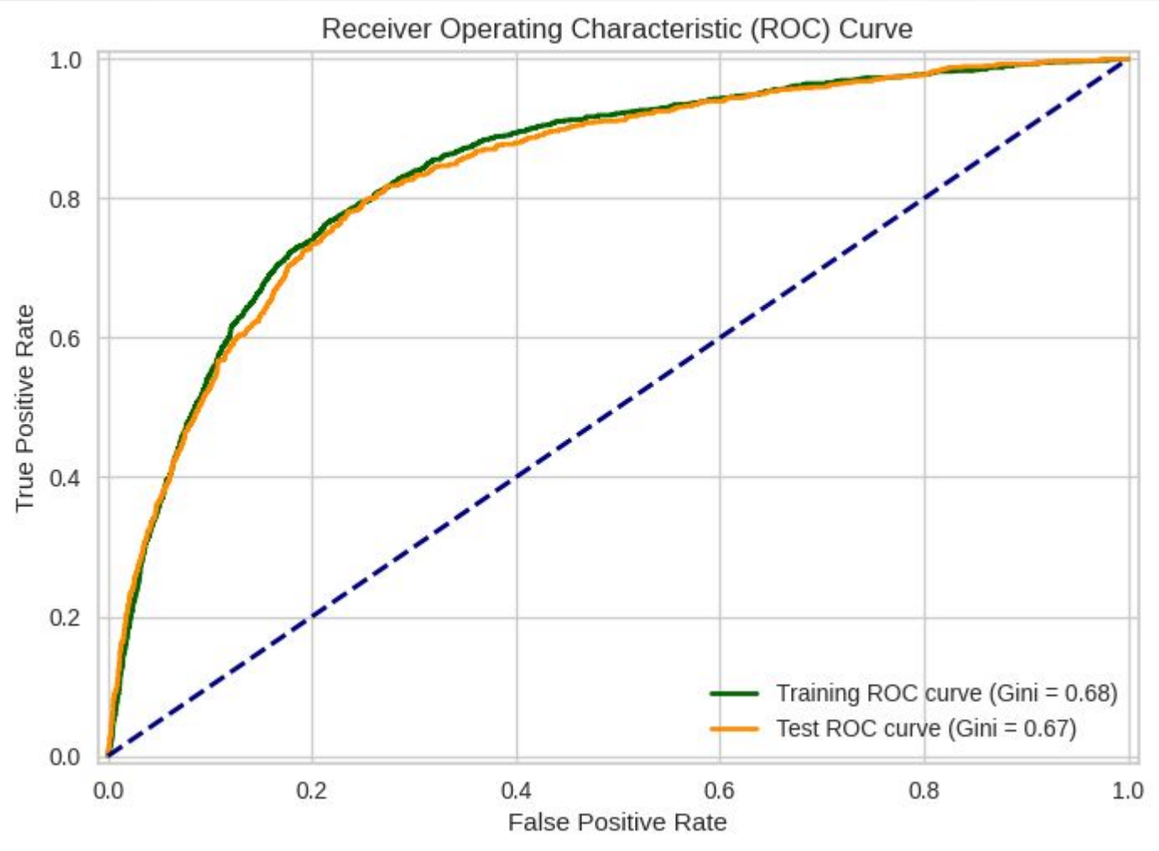
Overall Performance:

The AUC score for both training (0.84) and test data (0.83) suggests acceptable performance. An AUC of 1 represents a perfect classifier, while 0.5 represents a random guess. In general, a score **above 0.8 is considered good.**

Insights from Gini Coefficient:

The Gini coefficient values (0.68 for training and 0.67 for test) are **lower** than the corresponding AUC scores.

A rule of thumb suggests that a Gini coefficient above 0.5 indicates a good separation between classes. In this case, the values are around **0.67**, which means the model **can somewhat distinguish** between PayLater users and non-PayLater users, but it's not perfect.



Evaluation - Decile Performance

binning	actual_#non_takers	actual_#takers	total_obs	prob_takers	%non_takers	%takers	cumm%_non_takers	cumm%_takers
(0.0008518, 0.074678]	1087	19	1106	0.0172	0.1187	0.0100	0.1187	0.0100
(0.074678, 0.12016]	1074	31	1105	0.0281	0.1172	0.0164	0.2359	0.0264
(0.12016, 0.17471]	1068	37	1105	0.0335	0.1166	0.0195	0.3525	0.0459
(0.17471, 0.23882]	1054	52	1106	0.0470	0.1151	0.0275	0.4676	0.0734
(0.23882, 0.32155]	1056	49	1105	0.0443	0.1153	0.0259	0.5829	0.0993
(0.32155, 0.41285]	996	109	1105	0.0986	0.1087	0.0576	0.6916	0.1568
(0.41285, 0.52479]	930	176	1106	0.1591	0.1015	0.0929	0.7931	0.2497
(0.52479, 0.65875]	817	288	1105	0.2606	0.0892	0.1521	0.8823	0.4018
(0.65875, 0.80528]	634	471	1105	0.4262	0.0692	0.2487	0.9515	0.6505
(0.80528, 0.99772]	444	662	1106	0.5986	0.0485	0.3495	1.0000	1.0000

Insight:

The **'prob_takers'** column indicates the probability of an individual being classified as a 'taker' by the model within each bin. This **probability increases as the propensity score increases**, indicating a **higher likelihood** of being a 'taker' for individuals with higher propensity scores.

Analyzing the decile performance reveals that the **top 3 deciles** (highest predicted probability based on the "prob_takers" column), where the model assigns a **probability greater than 0.5**, **capture 30% of the total customer base**. This suggests that by focusing our outreach efforts on these top deciles (customers with a predicted probability > 0.5 of using PayLater), we can target a substantial portion of potential PayLater users (30%) with a higher degree of accuracy.

Conclusion & Further Recommendations

These findings suggest that the propensity model is **reasonably effective** in identifying the most promising paylater customers, especially in the **higher probability bins**.

Consider **cost-benefit analysis** to understand the optimal balance between the number of customers reached and the potential conversion rate for PayLater activation.

Implement techniques like oversampling, undersampling, SMOTE, or cost-sensitive learning during model training to **address the class imbalance** and **improve the F1-score** for PayLater users.

By setting an appropriate probability threshold, the model can be used to target customers who are more likely to take the paylater option. For instance, if the business **prioritizes precision** (minimizing false positives), a **higher threshold** could be set to target only the customers in the top probability bins, where the proportion of actual takers is highest.

Alternatively, if the business **prioritizes recall** (maximizing true positives) and is willing to tolerate a higher false positive rate, a **lower probability threshold** could be used to capture a larger segment of potential paylater customers.

Thank You!

Satryo Sunu Prabowo