

Credit Card Behaviour Score Prediction Using Classification and Risk-Based Techniques Report

Project Objective

To develop a machine learning pipeline that predicts whether a customer will default on their credit card payment in the next month, using historical financial and behavioral data.

1. Data Loading and Inspection

- The dataset includes customer demographic and financial features such as `age`, `LIMIT_BAL` (credit limit), `education`, `marriage`, and a series of payment (`pay_m`), bill (`bill_amt_m`), and repayment (`pay_amt_m`) records.
 - Initial inspection checked for missing values, data types, and duplicates.
 - The target variable is `next_month_default`, indicating if the customer defaulted in the following month.
-

2. Exploratory Data Analysis (EDA)

- **Class Imbalance:** The target variable is imbalanced, with significantly more non-defaulters than defaulters.
 - **Feature Distributions:** Visualizations explored the distributions of key features (e.g., `age`, `LIMIT_BAL`, `education`, `marriage`).
 - **Trends:** Payment status (`pay_m`), bill amounts, and payment amounts were analyzed for patterns.
 - **Correlation Analysis:** Heatmaps were used to identify correlations among features.
-

3. Financial Behavior Analysis

- Key behavioral indicators of default were identified:
 - Long payment delays and missed payments.
 - Sudden changes in bill or payment behavior.
 - Patterns of high credit utilization over time.
-

4. Feature Engineering

Several new features were constructed to better capture customer risk:

- `utilization_ratio` = $\text{avg}(\text{bill_amt1-6}) / \text{LIMIT_BAL}$
 - `delayed_months` = count of months with delayed payments
 - `max_delay` = maximum consecutive months of delinquency
 - `consec_delinquency` = count of consecutive delinquent months
 - Categorical features were encoded as needed.
-

5. Handling Class Imbalance

To address the class imbalance:

- **SMOTENC** was used to oversample the minority class.
 - **Ensemble learning** techniques were considered.
 - The class balance was re-examined after resampling.
-

6. Model Training and Evaluation

Multiple machine learning models were trained and compared:

- **Logistic Regression**
- **Decision Tree**
- **XGBoost**
- **LightGBM**
- **Random Forest**

Evaluation Metrics:

- Models were assessed using:
 - **AUC-ROC**
 - **Precision**
 - **Recall**
 - **F1-score**
 - **Confusion Matrix**
-

7. Threshold Tuning

- ROC and Precision-Recall curves were plotted.
 - The classification threshold was tuned to balance false positives and false negatives, with a preference for higher recall to minimize missed defaulters, aligning with typical banking risk appetites.
-

8. Model Explainability

- Feature importance can be used to visualise for tree-based models.
 - **SHAP** was used to explain individual predictions.
 - Features such as `repayment_ratio` and `delinquency_count` were highlighted as influential in predicting default.
-

9. Prediction and Output Generation

- The best-performing model was applied to the validation dataset.
 - Predictions were generated using the optimized threshold.
 - The final output was formatted as a CSV with customer IDs and predicted default status.
-

10. Summary and Key Takeaways

- **Approach:** A comprehensive machine learning pipeline was built, emphasizing robust feature engineering, handling of class imbalance, and careful model selection and evaluation.
- **Findings:** Behavioral features, especially those reflecting payment delays and credit utilization, are strong predictors of default.
- **Model Performance:** Tree-based models (XGBoost, LightGBM) provided high predictive power and interpretability, but logistic regression gave the highest recall.
- **Business Implications:** The cost of false negatives (missed defaulters) was considered more critical than false positives, guiding threshold selection.

Dataset Characteristics

- **Primary Context:** Credit default prediction dataset
 - **Key Challenge:** Highly unbalanced target variable (`next_month_default`)
 - **Total Features:** 26 columns (excluding Customer_ID)
 - **Target Variable:** `next_month_default` (binary classification)
-

Categorical Variables (9 features)

Demographics & Status:

- `marriage` : Marital status
- `sex` : Gender
- `education` : Education level

Payment History (6 features):

- `pay_0` : Payment status in current month
- `pay_2` through `pay_6` : Payment status in previous months (2-6)

Non-Categorical Variables (16 features)

Financial Capacity:

- `LIMIT_BAL` : Credit limit
- `age` : Customer age (preprocessed)

Billing Information (6 features):

- `Bill_amt1` through `Bill_amt6` : Monthly bill amounts

Payment Information (6 features):

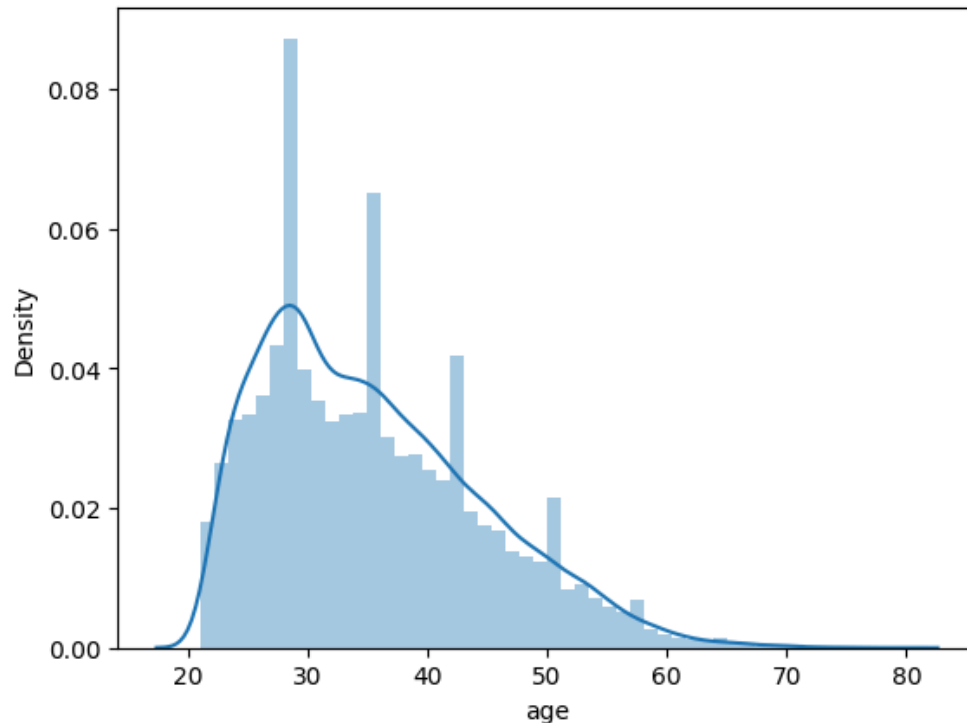
- `pay_amt1` through `pay_amt6` : Monthly payment amounts

Engineered Features (2 features):

- `AVG_Bill_amt` : Average billing amount
- `PAY_TO_BILL_ratio` : Payment-to-bill ratio

EDA and financial findings

The only feature that contains null values is age. the density plot of `age` looks as follow:



Data Distribution Analysis

Initial Findings

- **Distribution Shape:** The age variable exhibits a right-skewed distribution, deviating significantly from normality
- **Implication:** Traditional mean imputation would be inappropriate due to the skewed nature, as the mean would be pulled toward higher values by the right tail

Statistical Rationale

Right-skewed distributions are common in age data, particularly in datasets that may include:

- Younger populations with fewer elderly participants
- Age-restricted samples (e.g., working-age populations)
- Datasets with natural lower bounds (age cannot be negative)

Outlier Assessment

Methodology

- **Detection Method:** Box plot analysis using interquartile range (IQR)
- **Threshold Identified:** Ages above 60 years classified as outliers

- **Outlier Prevalence:** 0.8595% of the dataset

Analysis

The low percentage of outliers (< 1%) suggests:

- The dataset likely focuses on a younger demographic
- Outliers represent legitimate but uncommon age values
- Removal may not be necessary given the small proportion

Log Transformation Approach

Technical Implementation

1. **Transformation:** Applied natural logarithm to age values
2. **Objective:** Reduce right skewness to approximate normal distribution
3. **Validation:** Distribution plot of $\log(\text{age})$ showed improved normality

Mathematical Foundation

Log transformation is effective for right-skewed data because:

- It compresses larger values more than smaller ones
- Converts multiplicative relationships to additive ones
- Often results in approximately normal distributions

Imputation Strategy

Value Calculation

- **Log-scale Mean:** $\exp(\text{mean}(\log(\text{age}))) = 34.33$
- **Log-scale Median:** $\exp(\text{median}(\log(\text{age}))) = 34.00$
- **Selected Value:** 34 (rounded median for integer consistency)

Rationale for Median Selection

1. **Robustness:** Less sensitive to outliers than the mean
2. **Proximity:** Close agreement between transformed mean (34.33) and median (34.00) indicates successful normalization
3. **Data Type Consistency:** Integer value maintains the natural discrete nature of age data

Methodology Strengths

Advantages

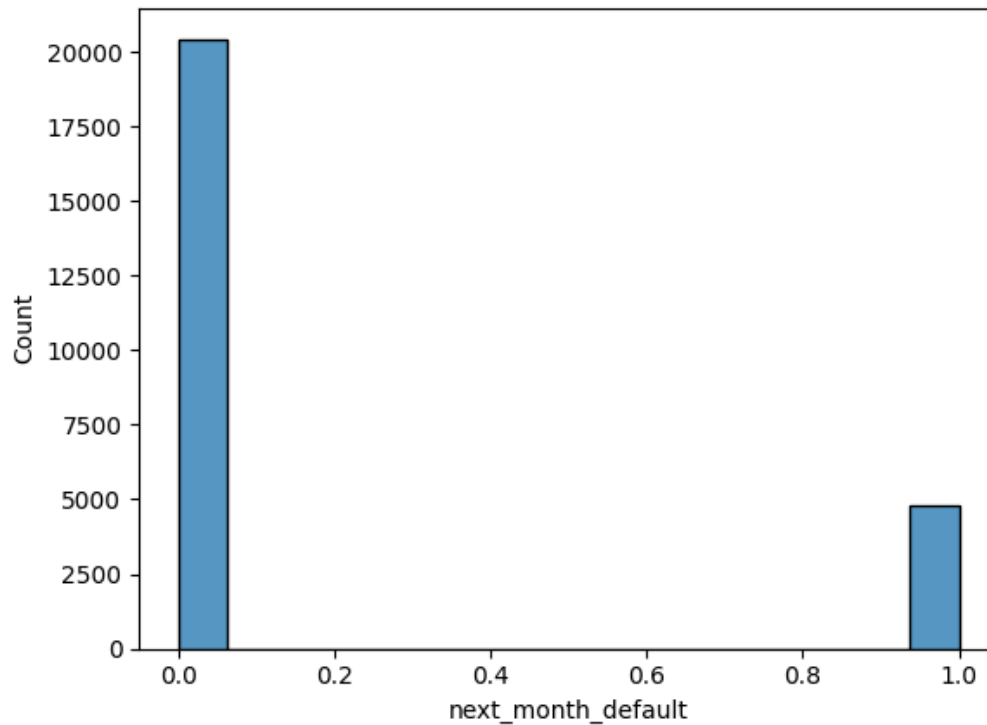
- **Theoretically Sound:** Addresses skewness before imputation
- **Data-Driven:** Uses actual distribution characteristics to guide decisions
- **Conservative Approach:** Median imputation reduces bias introduction
- **Maintains Data Integrity:** Preserves integer nature of age variable

Best Practices Demonstrated

- Distribution analysis before imputation
- Appropriate transformation selection
- Validation of transformation effectiveness
- Consideration of data type requirements

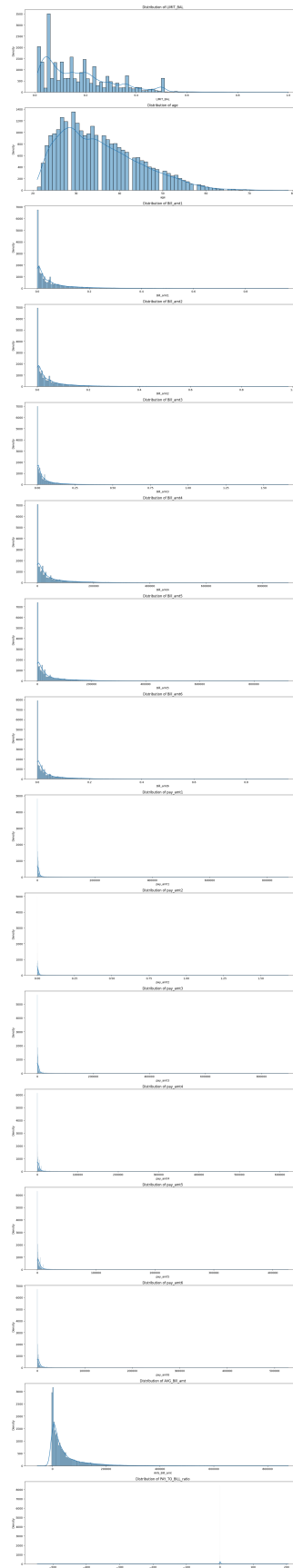
Class Imbalance Implications

Dataset is highly unbalanced. It contains approximately 20,000 non default samples and 5000 default samples.



Non-Categorical variables

None of the non-categorical variable follow *normal distribution*



Creating a New Feature: Credit Utilization Ratio

To better understand a customer's credit behavior, we calculate a new feature called `utilization_ratio`.

Formula:

Let:

- `AVG_BILL_AMTi` = Average bill amount over the past months for customer i
- `LIMIT_BALi` = Credit limit for customer i




Then,

$$\text{utilization_ratio}_i = \text{AVG_BILL_AMT}_i / \text{LIMIT_BAL}_i$$

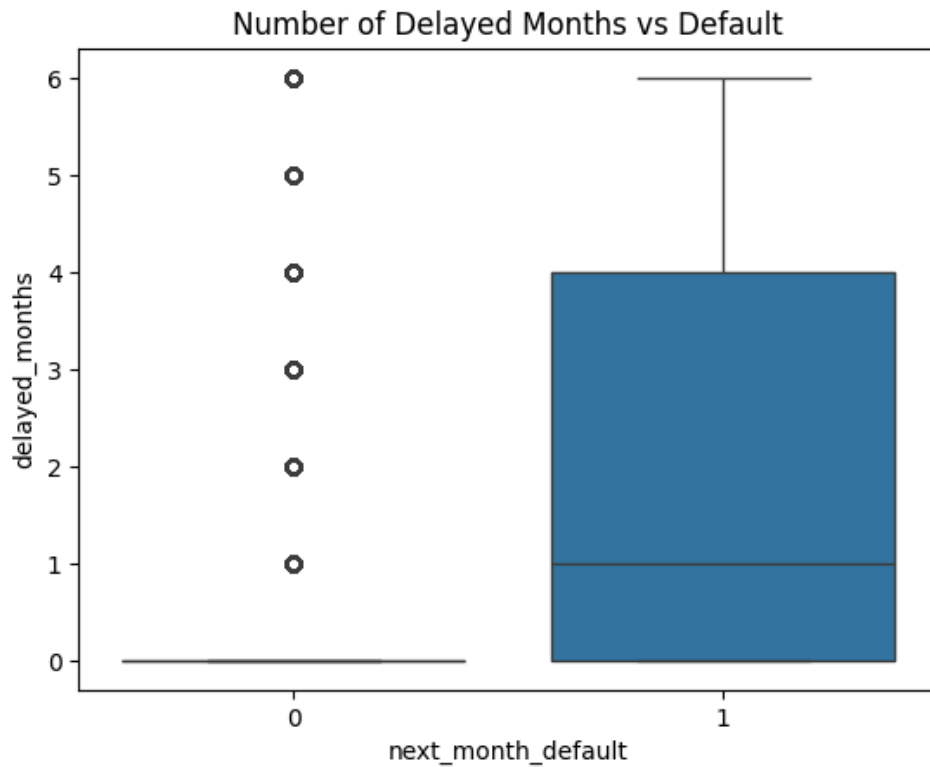
mean utilization ratio of the default is slightly more.

Feature Engineering: Payment Delay Metrics

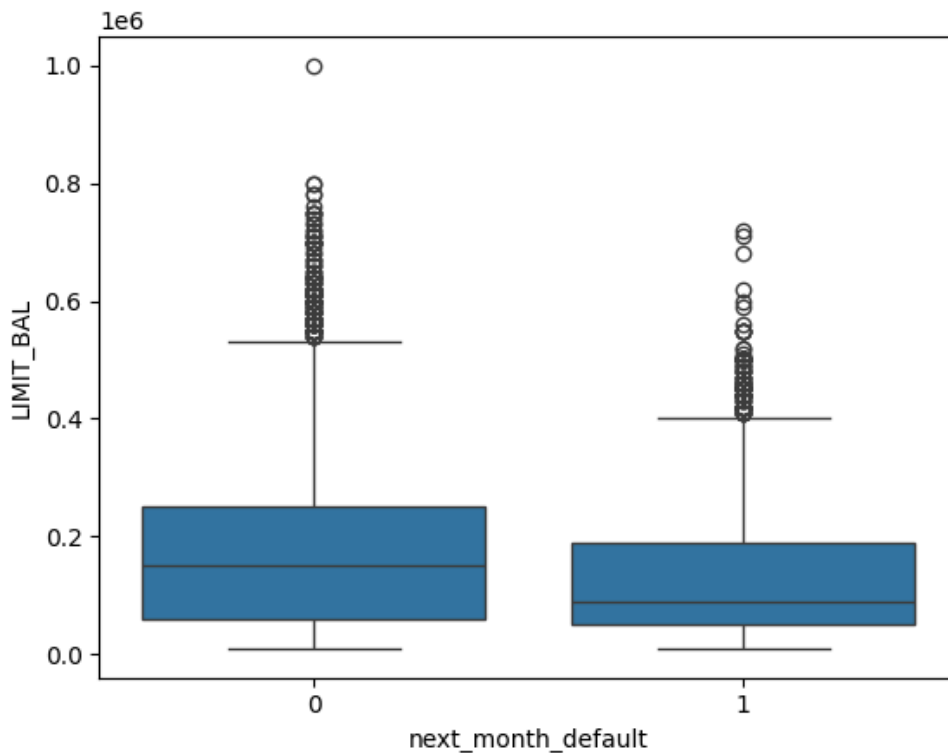
The following derived features are calculated to capture customer payment behavior:

-  `delayed_months`
→ Total number of months in which the **bill payment was delayed** (i.e., `pay_m > 0`)
-  `max_delay`
→ The **maximum number of months** a customer delayed bill payment across all months
-  `consec_delinquency`
→ The number of months where the **payment delay exceeded 3 months** (i.e., `pay_m > 3`)

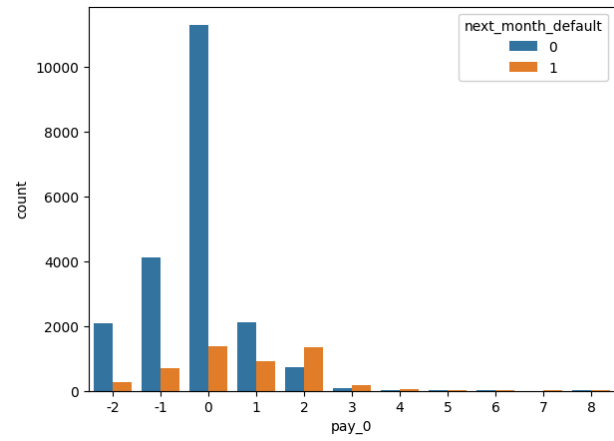
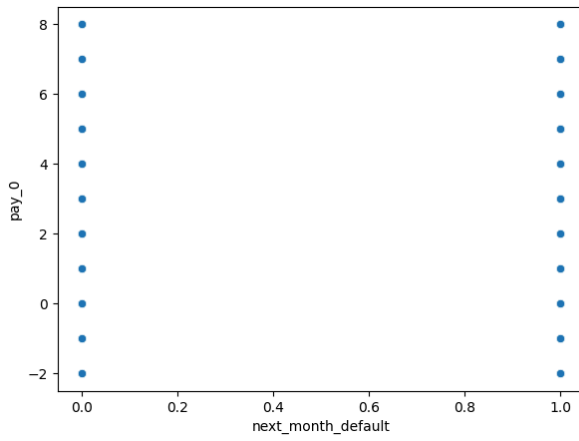
✓ These features help quantify credit risk and payment discipline over time.

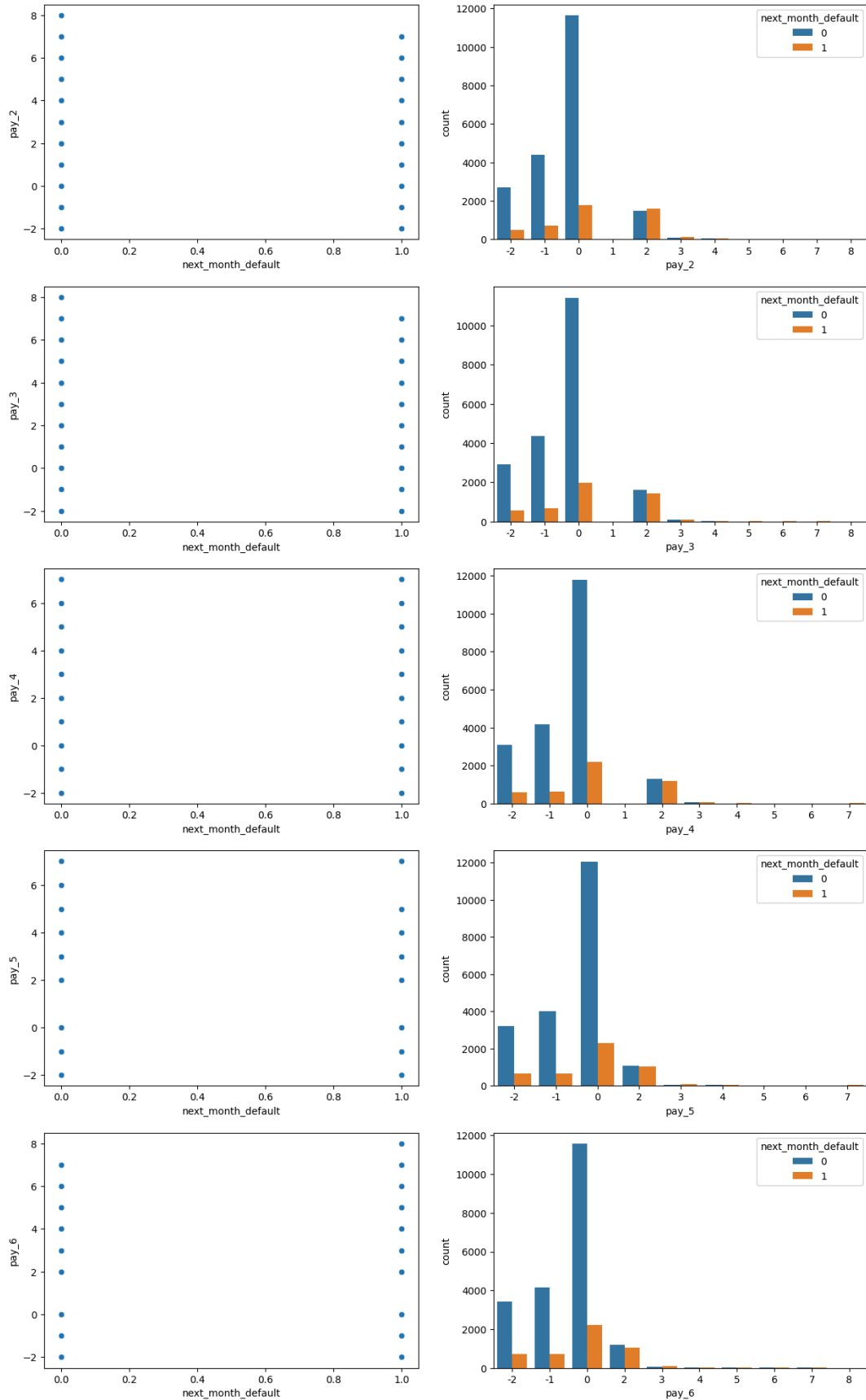


mean delayed months of defaults is 1 on the other hand mean delayed month of non default is 0 however some have greater values of delayed months.



limit_bal mean is greater for non default than default





Distribution of Unique `pay_m` Values and Their Counts

This visualization presents:

- ◆ The **unique `pay_m` values** for both classes of `next_month_default`
- ◆ A **count plot** (on the right) showing the frequency of each `pay_m` value

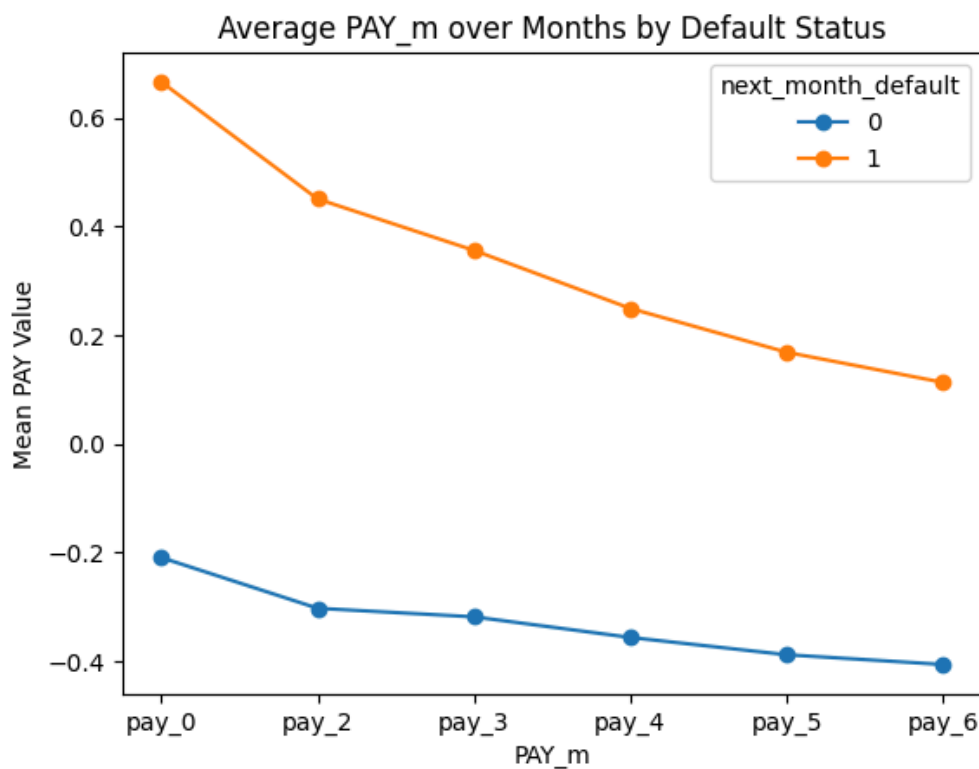
Observation

- Values **greater than 2** have **negligible counts**, indicating that such delays are rare in the dataset.
- To simplify the distribution and reduce sparsity, we will **merge all `pay_m` values ≥ 2** into a single category.

Mapping Strategy

- All values ≥ 2 will be grouped and represented as:

`pay_m = 2` → Indicates payment is delayed by 2 or more months



Average `pay_m` Value for Default vs Non-Default Customers

This plot shows the **average** `pay_m` values for customers who defaulted and those who did not.

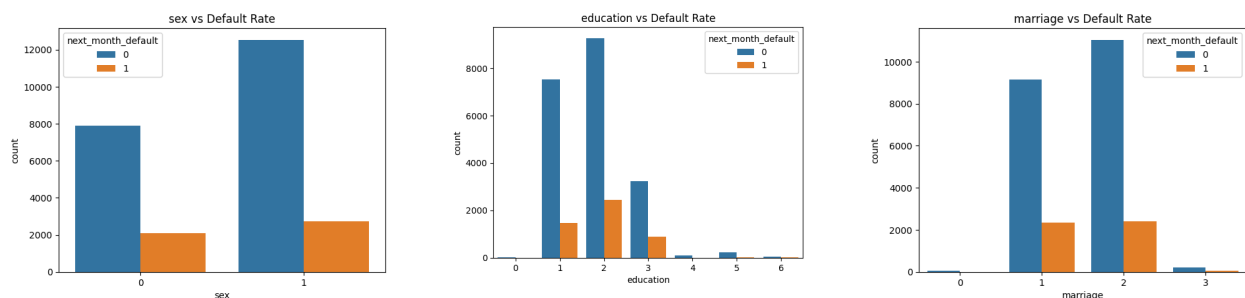
- **Defaults:**

The mean `pay_m` values are **positive**, indicating that these customers **consistently failed to pay their bills in full**, leaving some amount due each month.

- **Non-Defaults:**

The average `pay_m` values are **less than zero**, suggesting that most non-defaulting customers **paid their bills on time** or even **in advance**.

✓ A lower or negative `pay_m` value generally reflects good repayment behavior, while a higher value indicates delinquency.



these lower counts could be due to dataset imbalance

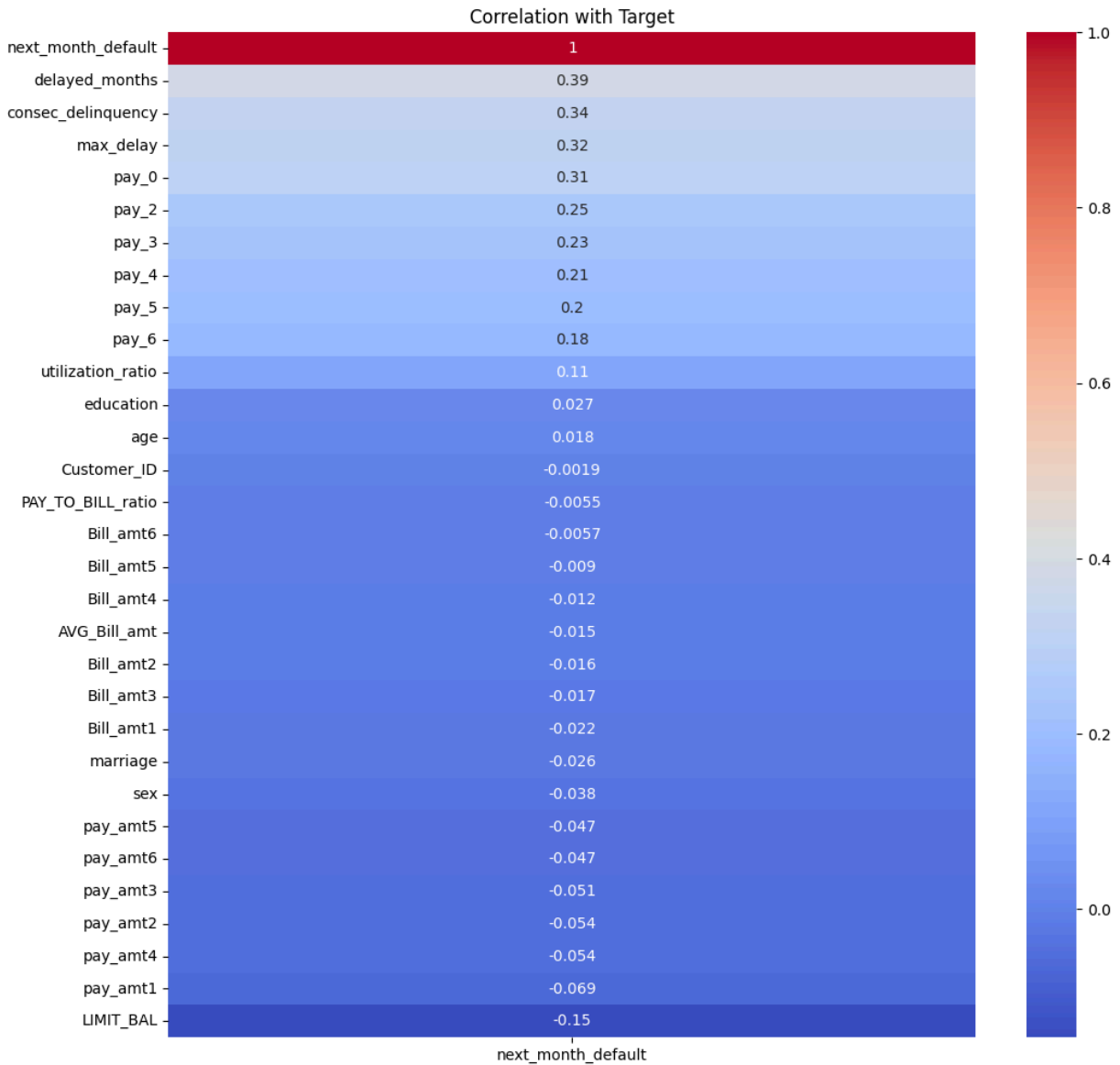
Financial Insights

Customers who underpay repeatedly despite small bills are riskier.

Customers with high utilization and long payment delay streaks are high risk.

Older customers might have different payment behavior than younger ones.

Underpayment or payment stagnation despite increasing bills is a red flag.



this shows the correlation values of the features with the target variable

checking for unique values and their counts


```
df.delayed_months.unique()
```

```
array([3, 0, 1, 5, 2, 6, 4])
```

```
df.consec_delinquency.unique()
```

```
array([1, 0])
```

```
df.max_delay.unique()
```

```
array([ 2,  0, -2,  1, -1,  8,  3,  4,  7,  5,  6])
```

```
print(f'unique values in marriage{df.marriage.unique()}')
```

```
unique values in marriage[2 1 3 0]
```

```
print(f'unique values in sex{df.sex.unique()}')
```

```
unique values in sex[0 1]
```

```
print(f'unique values in education{df.education.unique()}')
```


```
unique values in education[2 1 3 4 5 6 0]
```

combining some feature values

```
[ ] df['marriage'] = df['marriage'].apply(lambda x: 3 if x == 0 else x)
```

```
[ ] df['education'] = df['education'].apply(lambda x: 4 if x in [0,5,6] else x)
```

```
[ ] for col in pay_cols:  
    df[col] = df[col].apply(lambda x: 2 if x in [3,4,5,6,7,8] else x)
```



 **Meaning of pay_m = 2**

When the value of pay_m is 2, it indicates that the **borrower delayed payment by at least 2 months**.

Using **SMOTENC** to Handle Dataset Imbalance

SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous features) is an advanced variant of the **SMOTE** algorithm. It is specifically designed to handle **imbalanced**

classification problems where the dataset contains both:

-  Categorical (nominal) features
-  Numerical (continuous) features

⚠️ Problem with Regular SMOTE

Standard SMOTE creates synthetic samples by interpolating between existing minority class samples. This works well for **continuous features**, but is **inappropriate for categorical data**.

❌ Why?

Interpolation between categorical values is meaningless.

Example:

Color = Red (encoded as 0)

Color = Blue (encoded as 1)

Interpolated value = 0.5 → ❌ Invalid (0.5 ≠ a real category)

Compared to other techniques:

Technique	Categorical Support	Synthetic Data	Risk of Overfitting	Data Loss
Random OverSampling	✅ Yes	❌ No	⚠️ High	❌ No
Random UnderSampling	✅ Yes	❌ No	✅ Low	⚠️ Yes
SMOTE	❌ No	✅ Yes	✅ Low	❌ No
ADASYN	❌ No	✅ Yes	✅ Low	❌ No
SMOTENC (used)	✅ Yes	✅ Yes	✅ Low	❌ No

Model Comparison Report: Classification Performance (F2-Focused)

Performance Summary

Metric	XGBoost	Decision Tree	Random Forest	LightGBM	Logistic Regression
F2 Score	0.4188	0.4187	0.4629	0.4742	0.5562
Recall (Class 1)	0.3992	0.4543	0.4511	0.4615	0.6019
Precision (Class 1)	0.5210	0.3187	0.5167	0.5324	0.4267

AUC-ROC	0.7509	0.6129	0.7653	0.7675	0.7602
Accuracy	0.82	0.71	0.82	0.82	0.7701

Key Insights

Best Model by F2 Score (Recall-weighted)

- **Logistic Regression (F2 = 0.556):**
 - Best performance on **F2 Score**, emphasizing **recall**.
 - Ideal for imbalanced classification tasks.
 - Highest recall (0.60), but lower precision (0.43).

Balanced Model

- **LightGBM:**
 - Very balanced performance across all metrics.
 - Strong **Precision** (0.532) and **ROC-AUC** (0.767).
 - Good choice for general use when you want strong F2, precision, and recall.

Weakest Model




- **Decision Tree:**
 - Lowest AUC and Precision.
 - Underperforms across most metrics.
 - Should be avoided unless interpretability is the top priority.

Class-Wise Behavior from Confusion Matrices






Model	True Neg (TN)	False Pos (FP)	False Neg (FN)	True Pos (TP)
XGBoost	3735	353	578	384
Decision Tree	3154	934	525	437
Random Forest	3682	406	528	434
LightGBM	3698	390	518	444
Logistic Reg.	3310	778	383	579

- **Logistic Regression** has the **lowest false negatives (383)** — excellent for **recall-focused** applications.
- **LightGBM** achieves a good balance between precision and recall.

Recommendations

-  If your **goal is recall** (e.g., fraud detection, disease prediction), use **Logistic Regression**.
-  For **balanced performance**, choose **LightGBM**.
-  Avoid **Decision Trees** due to poor performance across all metrics.

Final Verdict

Rank	Model	Reason
	Logistic Regression	Best F2 Score and Recall; strong ROC-AUC; good for imbalanced data
	LightGBM	Balanced performance across metrics; high precision and AUC
	Random Forest	Decent overall but slightly worse recall
	XGBoost	High precision but lower recall reduces F2
	Decision Tree	Weakest overall; poor AUC and precision

Ensemble Methods Without SMOTE: Performance & Reasoning

Objective

We tested ensemble models on **imbalanced data without SMOTE** to assess their performance in identifying credit card defaulters. Since detecting defaulters (positive class) is more important than identifying non-defaulters, **Recall and F2 Score** are prioritized.

Balanced Random Forest

Balanced Random Forest is designed to handle class imbalance by undersampling the majority class during each bootstrap iteration.

Metrics:

- **F2 Score:** 0.5369
- **Recall:** 0.5582
- **Precision:** 0.4657
- **AUC-ROC:** 0.7812



Classification Report:

Class	Precision	Recall	F1-score	Support
0	0.89	0.85	0.87	4088
1	0.47	0.56	0.51	962

- **Confusion Matrix:**

[[3472, 616],

[425, 537]]



Reasoning:

- The model shows **strong recall for defaulters (class 1)** at 0.56, meaning it **correctly captures over half of the actual defaulters**.
- Although precision is lower (0.46), this trade-off is acceptable in financial risk contexts.
- **Balanced Random Forest is effective without SMOTE** because it handles imbalance internally via resampling.



Regular Random Forest (Without SMOTE)

This model is trained **without any resampling or class weighting**, on the original imbalanced data.



Metrics:

- **F2 Score:** 0.3313
- **Recall:** 0.2942
- **Precision:** 0.6690
- **AUC-ROC:** 0.7700



Classification Report:

Class	Precision	Recall	F1-score	Support
0	0.85	0.97	0.91	4088
1	0.67	0.29	0.41	962

- **Confusion Matrix:**

[[3948, 140],

[679, 283]]






Reasoning:

- The model is **heavily biased toward the majority class (non-defaulters)**.
- While it achieves high precision (0.67), the **recall is very low (0.29)**, meaning **most defaulters are missed**.
- This makes it **unreliable for identifying risky customers**, despite the good overall accuracy and AUC-ROC.

Conclusion

Model	F2 Score	Recall (Class 1)	Precision (Class 1)	AUC-ROC
Balanced RandomForest	0.537	0.558	0.466	0.781
Regular RandomForest	0.331	0.294	0.669	0.770

-  **Balanced Random Forest** outperforms standard Random Forest in scenarios where **identifying defaulters is crucial**.
-  In default prediction tasks, **recall and F2 Score** are more meaningful than accuracy or AUC, as they ensure fewer **false negatives**.
-  **Conclusion:** Balanced approaches or SMOTE-based methods are **essential when working with imbalanced financial datasets**.

LightGBM Hyperparameter Tuning with SMOTE: Evaluation & Insights

Grid Search Setup

We used `GridSearchCV` to find optimal hyperparameters for the **LightGBM classifier** using **SMOTE-resampled data** to address class imbalance.

Parameters and Evaluation:

- **Search space** included:
 - `n_estimators`: [100, 200]
 - `learning_rate`: [0.01, 0.05, 0.1]
 - `num_leaves`: [15, 31, 62]
 - `max_depth`: [-1, 5, 10]
- **Evaluation Metric:** F2 Score (focuses on recall)
- **Class Weight:** `balanced`

- **Cross-validation:** 3-fold

Best Parameters Found

```
{  
  'learning_rate': 0.05,  
  'max_depth': 10,  
  'n_estimators': 100,  
  'num_leaves': 62  
}
```

Best F2 Score (on SMOTE-resampled CV)

F2 Score (CV on SMOTE-resampled data): 0.7896

Final Evaluation on Original Validation Set

After retraining the model using the best hyperparameters on the full **SMOTE-resampled training set**, we evaluated its performance on the **original (non-SMOTE) validation set**.

Performance Metrics

- **F2 Score:** 0.4801
- **Recall:** 0.4678
- **Precision:** 0.5364
- **AUC-ROC:** 0.7671

Classification Report

Class	Precision	Recall	F1-score	Support
0	0.88	0.90	0.89	4088
1	0.54	0.47	0.50	962

Confusion Matrix

```
[[3699, 389],  
 [ 512, 450]]
```

Interpretation & Reasoning

Although the cross-validated F2 score on SMOTE-resampled data was high (~0.79), the actual performance on the original imbalanced validation set dropped significantly (**F2 \approx 0.48**).

This discrepancy can be explained as follows:

- **SMOTE** synthetically balances the dataset, improving performance during training and cross-validation.
- However, in the **real-world imbalanced distribution**, the model finds it harder to detect the minority class (defaulters), resulting in lower recall and F2 score.

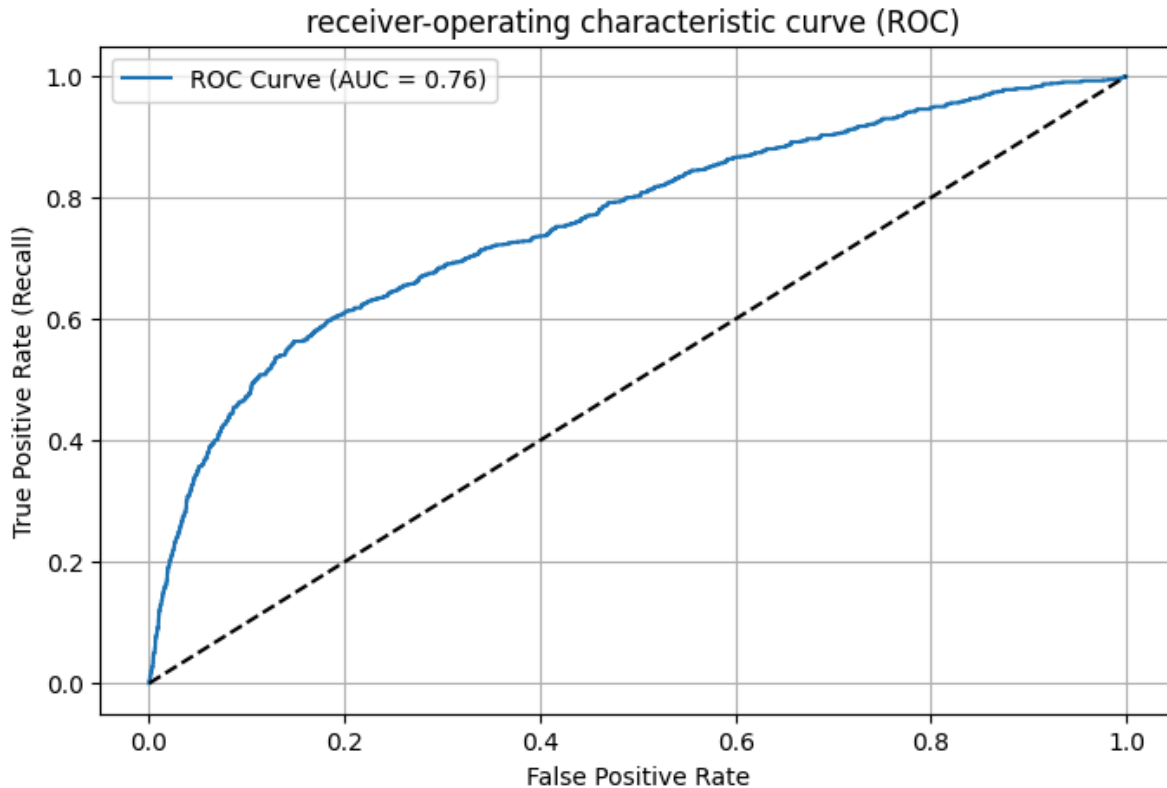
Comparison with Logistic Regression

Model	F2 Score	Recall	Precision	AUC-ROC
Logistic Regression	0.5562	0.6019	0.4267	0.7602
Tuned LightGBM	0.4801	0.4678	0.5364	0.7671

Despite being a simpler model, **Logistic Regression** outperforms the **tuned LightGBM** in **recall and F2 score**—the most important metrics in this credit default detection task.

Conclusion

- **Recall is critical** in credit default problems because missing a defaulter (false negative) is far more costly than a false positive.
- **F2 Score**, which places more emphasis on recall, is the right choice for evaluation in this context.
- While **SMOTE + hyperparameter tuning** helped during model training, the LightGBM model still **underperforms Logistic Regression** on actual (imbalanced) data.
- This highlights the **importance of evaluating models on the true distribution**, not just on balanced or synthetic datasets.



Threshold Tuning to Maximize F2 Score

Objective

In credit default prediction, **recall** is more critical than precision — we want to catch as many defaulters (positive class) as possible, even if it means a few false alarms.

To do this, we tuned the decision threshold of our classifier to **maximize the F2 score**, which gives more weight to recall.

Best Threshold Found

- **Threshold:** 0.356
- **F2 Score:** 0.586

Performance at Threshold = 0.356

- **Precision:** 0.264
- **Recall:** 0.845
- **F1 Score:** 0.402

- **F2 Score:** 0.586

Classification Report

Class	Precision	Recall	F1-score	Support
0	0.92	0.44	0.60	4088
1	0.26	0.85	0.40	962

- **Accuracy:** 0.52
- **Macro Avg F1:** 0.50
- **Weighted Avg F1:** 0.56

Confusion Matrix

[[1816, 2272],
[149, 813]]

Interpretation & Reasoning

- **High Recall (0.845):** The model catches a large portion of actual defaulters (positive class).
- **Low Precision (0.264):** Many of the predicted defaulters are actually non-defaulters.
- **High False Positives:** 2,272 non-defaulters were misclassified.
- **Low False Negatives:** Only 149 defaulters were missed — this is a good outcome in this domain.

Why F2 Score Improved?

- **F2 score** gives **more importance to recall** than precision.
- By lowering the threshold to 0.35, the model becomes more "aggressive" in predicting the positive class (default).
- This results in **more true positives** (higher recall), which helps improve the F2 score.
- However, the **precision drops**, because many negative class instances are incorrectly labeled as defaulters.
- Since F2 still considers precision (though less than recall), the low precision limits how high the F2 can go — but overall, this trade-off is acceptable for catching defaulters.

Summary

Metric	Value
Best Threshold	0.35
F2 Score	0.5847
Recall (Defaulters Caught)	0.845
Precision	0.264
False Negatives	149
False Positives	2272



Final Insight

In credit default detection, **minimizing false negatives** is more important than keeping false positives low.

Thus, **threshold tuning is an effective strategy** for maximizing F2 and catching more defaulters — even at the cost of lower precision.



Why Recall and F2 Score Matter in Credit Card Default Prediction



Problem Context

The objective of this classification task is to **predict credit card defaulters**—customers who are likely to miss payments in the future. Identifying these individuals is crucial for minimizing financial risk to the bank or credit institution.



Why High Recall is Critical

- **False Negatives (FN)** — i.e., **failing to identify a defaulter** — are far more costly than false positives.
- Letting a potential defaulter pass as a "safe" customer may result in **loan disbursement to a high-risk individual**, leading to **financial loss**.
- Hence, our model must focus on **catching as many actual defaulters as possible**, even at the cost of occasionally flagging non-defaulters.



This is exactly what Recall measures:

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

It quantifies the percentage of actual defaulters that were correctly identified by the model.

Why Use F2 Score Instead of F1

- The **F1 Score** balances **precision** and **recall** equally.
- However, in credit default prediction, **recall is more important than precision**.
 - It's acceptable to have a few false alarms (non-defaulters flagged as defaulters) as long as we **don't miss actual defaulters**.
- The **F2 Score** gives more weight to **recall** than to precision:

$$F_{\beta} = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

| where $\beta = 2$, so Recall gets 4× more importance than Precision.

✓ Therefore, **F2 Score is a better evaluation metric** for this problem.

Conclusion

To **reduce financial risk**, our model must be **recall-oriented**, ensuring that **true defaulters are rarely missed**. That's why:

- We **optimize for F2 Score**
 - We **prioritize models with higher recall**
 - Models like **Logistic Regression** and **LightGBM**, which offer **higher recall and F2**, are preferred.
 - Still in order to get better accuracy one should use **LightGBM** but accuracy is not a requirement here.
-

Logistic Regression Feature Importance Analysis

Objective

We interpret the coefficients of a **Logistic Regression** model trained to predict **credit card default**.

This analysis helps understand **how each feature influences** the likelihood of a customer defaulting on payment.

Top Features by Coefficient & Odds Ratio

Feature	Coefficient	Odds Ratio
delayed_months	0.857	2.356
Bill_amt2	0.407	1.502
pay_0	0.308	1.361
Bill_amt6	0.119	1.126
Bill_amt3	0.105	1.111
utilization_ratio	0.095	1.100
max_delay	0.080	1.083
age	0.071	1.073
consec_delinquency	0.031	1.031
PAY_TO_BILL_ratio	0.017	1.017
sex	0.008	1.008
pay_2	-0.011	0.989
AVG_Bill_amt	-0.025	0.976
pay_amt3	-0.040	0.961
pay_amt5	-0.040	0.960
pay_amt6	-0.058	0.944
pay_amt4	-0.062	0.939
Bill_amt4	-0.063	0.939
Bill_amt5	-0.090	0.914
pay_4	-0.099	0.906
pay_6	-0.099	0.905
marriage	-0.108	0.898
education	-0.120	0.887
pay_3	-0.127	0.881
pay_5	-0.130	0.878
LIMIT_BAL	-0.198	0.820
pay_amt1	-0.273	0.761
pay_amt2	-0.282	0.755
Bill_amt1	-0.338	0.713



Interpretability: Key Insights

✓ Positive Coefficients / Odds Ratio > 1

These features **increase the likelihood of default**:

- **delayed_months** (**OR = 2.36**): Customers with more delayed months are significantly more likely to default.
- **pay_0** (**OR = 1.36**): Delay in the most recent payment is a strong signal of risk.
- **High bill amounts** (**Bill_amt2** , **Bill_amt6** , etc.): Suggest ongoing debt load.

✗ Negative Coefficients / Odds Ratio < 1

These features **decrease the odds of default**:

- **LIMIT_BAL** (**OR = 0.82**): Higher credit limit is associated with lower risk — likely due to financial stability.
- **pay_amt1** , **pay_amt2** : Higher past repayments reduce default risk.
- **education** , **marriage** : Demographic features may correlate with financial behavior.

🎯 Why Interpretability Matters

- Logistic Regression is **inherently interpretable**, allowing us to **trust and audit** predictions — crucial in **finance and risk modeling**.
- Each coefficient indicates **log-odds impact** on the default probability.
- **Odds Ratio > 1** → Higher risk of default **Odds Ratio < 1** → Lower risk of default

This helps:

- **Risk teams** understand customer profiles.
- **Regulators** trust the system's decisions.
- **Business** strategize on credit limit adjustments or targeted outreach.

📌 Model Selection Context

Since our evaluation metric is **F2 Score**, we prioritize **recall** (catching as many defaulters as possible):

F2 gives more weight to recall than precision, ideal in scenarios like:

- Credit risk assessment
- Fraud detection
- Medical diagnosis

False negatives (missed defaulters) are **costly** — so we prefer models that **maximize recall** while maintaining interpretability.

✓ Final Thought

While more complex models (like Random Forest or LGBM) may offer slightly higher raw performance, **Logistic Regression** remains a **strong baseline** due to:

- Solid recall performance (especially with threshold tuning)
- Transparent feature importance
- Actionable insights for

