esade

# AI2 Assignment 1

## Lending Club Loans Business Case

**Darren Darius Tan**
**MIBA 24/25**

Do Good. Do Better.

# 01 – Data Preparation

# 02 – Model Creation (Closed Loans)

# 03 – Model Interpretation and Implementation (Closed Loans)

# 04 – Model Application to Open Loans

# 01

# Data Preparation

**For Parts 01, we refer only**
**to AI2_Assignment1_data_cleaning.ipynb exclusively**

# 1.0: Understanding the Data Columns

- There are a total of 142 columns, and 2,029,952 rows, and can be categorized below.
- Based on domain knowledge, here are some of the important columns that we would use later are noted below:

Loan Identification

Loan Basic Information

1. loan_amnt
2. term
3. int_rate
4. Grade
5. loan_status

Borrower Information

Joint Applicant Information

Credit History

Detailed Credit Metrics

Payment Information

1. out_prncp
2. total_rec_prncp
3. total_rec_int
4. Grade
5. loan_status

Hardship Information

Note to self that some columns here (by nomenclature) introduces data leakage

# 1.1: Loading and Initial Inspection

## Goals due to massive dataset:

1. After cleaning the data, I aim to load store the data in 2 separate parquet files for df_closed and df_open, so as to reduce time re-running the python notebook from the beginning.

2. Hence, data cleaning in this notebook (from df to df_clean) shall only remove columns, and will not remove any rows that would disrupt the indexing of rows (which I would join columns to dataframes later).

3. After removing columns, missing values that remain in df_closed and df_opened shall only be imputed.

## Issues Log from Columns (Cumulative)

| S/N | Variables Affected | Description of Issue | Planned Fix |
|---|---|---|---|
| 1 | ‘next_pymnt_d’<br>‘verification_status_joint’<br>‘sec_app_earliest_cr_line’ | Columns 48, 58, 117 have mixed type | Check for specific dtypes and convert to what make sense |
| 2 | ‘int_rate’ | Representation of percentage: e.g. 7.97% | Need to divide by 100, remove % |
| 3 | ‘loan_status’ | Need to feature engineer into ‘1’ and ‘0’ for binary classifications | To filter Good and Bad loans based on Closed and Open Loans respectively, into 2 separate dataframes |

# 1.2: Check Missing and Duplicate Values

| | count | pct |
|---|---|---|
| hardship_loan_status | 1956776 | 96.395 |
| hardship_reason | 1956678 | 96.390 |
| hardship_status | 1956677 | 96.390 |
| hardship_dpd | 1956675 | 96.390 |
| hardship_length | 1956674 | 96.390 |
| hardship_type | 1956674 | 96.390 |
| hardship_end_date | 1956674 | 96.390 |
| hardship_start_date | 1956674 | 96.390 |
| deferral_term | 1956674 | 96.390 |
| payment_plan_start_date | 1956674 | 96.390 |
| orig_projected_additional_accrued_interest | 1938298 | 95.485 |
| hardship_payoff_balance_amount | 1935474 | 95.346 |
| hardship_last_payment_amount | 1935474 | 95.346 |
| hardship_amount | 1935474 | 95.346 |
| sec_app_revol_util | 1923768 | 94.769 |
| revol_bal_joint | 1921932 | 94.679 |
| sec_app_open_act_il | 1921931 | 94.679 |
| sec_app_earliest_cr_line | 1921931 | 94.679 |
| sec_app_open_acc | 1921931 | 94.679 |
| sec_app_chargeoff_within_12_mths | 1921931 | 94.679 |
| sec_app_inq_last_6mths | 1921931 | 94.679 |
| sec_app_fico_range_high | 1921931 | 94.679 |
| sec_app_num_rev_accts | 1921931 | 94.679 |
| sec_app_fico_range_low | 1921931 | 94.679 |
| sec_app_collections_12_mths_ex_med | 1921931 | 94.679 |
| sec_app_mort_acc | 1921931 | 94.679 |

```
# number of columns with missing values
missing_df[missing_df['count'] > 0].shape[0]
✓ 0.0s
69
```

```
# Check there is no duplicated rows
df.duplicated().sum()
✓ 15.2s
0
```

```
len(missing_to_drop)
✓ 0.0s
35
```

**Observations:**

1. Identified 69 columns with missing values and their percentage of missing values (pct).
2. Confirmed that there are no duplicated rows (and 'id').
3. 35 columns (in the list: missing_to_drop) also have more than 50% missing values.

**Cleaning Plan:**

1. Remove the 35 columns with more than 50% missing values
2. For the remaining columns with missing values, we keep it in the dataframe for further imputation in later steps.

esade

# 1.3: Inspect Numerical and Categorical

## Issues Log from Columns (Cumulative)

# numerical variables: 108
# categorical variables: 34

| S/N | Variables Affected | Description of Issue | Planned Fix / Remarks | Order of Fix in Code |
|---|---|---|---|---|
| 1 | ‘next_pymnt_d’<br>‘verification_status_joint’<br>‘sec_app_earliest_cr_line’ | Columns 48, 58, 117 have mixed type | Check for specific dtypes and convert to what make sense | |
| 2 | ‘int_rate’ | Representation of percentage: e.g. 7.97% | Need to divide by 100, remove % | |
| 3 | ‘loan_status’ | Need to feature engineer into ‘1’ and ‘0’ for binary classifications | To filter Good and Bad loans based on Closed and Open Loans respectively, into 2 separate dataframes | |
| 4 | ‘int_rate’<br>‘revol_util’ | Not object but float variable | Convert from object to float | |
| 5 | url,<br>desc,<br>Sub_grade,<br>Emp_title,<br>Title,<br>Zip_code,<br>Addr_state | Removal due to lack of semantic meaning, high dimensionality, already have a parent variable, or is a ‘zero-variance’ column | See data_cleaning.ipynb for more information | |
| 6 | 1. issue_d<br>2. earliest_cr_line<br>3. last_pymnt_d<br>4. next_pymnt_d (> 50% missing values, remove anyways)<br>5. last_credit_pull_d<br>6. sec_app_earliest_cr_line (> 50% missing values, remove anyways)<br>7. hardship_start_date (> 50% missing values, remove anyways)<br>8. hardship_end_date (> 50% missing values, remove anyways)<br>9. payment_plan_start_date ((> 50% missing values, remove anyways) | Datetime object requires feature engineering.<br>We also notice its in mmm-YYYY format | Convert from object to datetime | |

# 1.4: Identifying Highly Correlated Variables

**Next, we set a threshold to display highly correlated variables of > 85%. If too highly correlated, it would mean**

1. They might be representing the same meaning in domain knowledge
2. Even if variables represent different meaning, the issue of multi-collinearity exist. This then creates redundancy in the model.

**Goal of the correlation calculations:**

To remove duplicated variables with correlations in one dimension (based on domain understanding representation) that is above our stated threshold 85%:

1. Categorize them manually through an experimental process. This means that I will let the dataset decide amongst a correlation pair which variable to drop, and eventually, I will know that the number of categories.
2. In the end by the end of the process, I know that there are 12 variables which remain, in which any variable could arbitrarily represent these 12 different meanings.
3. From here, to decipher what these 12 variables would represent, I then look through each variable and its pair (some of which are repeated), and then I put them in the same category.
4. After putting these highly correlated variables in the same category, the category would have its own aggregate semantic meaning based on how similar the names of the variable are in the category. I then give this category a new name below.

**Note:**

This approach I undertake below sounds like a precursor to Principal Component Analysis. However as of this assignment, we technically have not learnt PCA yet. And hence, I take a logical approach to determine the removal of highly correlated features.

# 1.4: Identifying Highly Correlated Variables

| | Column1 | Column2 | Correlation |
|---|---|---|---|
| 21 | hardship_length | deferral_term | 1.000000 |
| 20 | sec_app_fico_range_high | sec_app_fico_range_low | 1.000000 |
| 6 | fico_range_high | fico_range_low | 1.000000 |
| 7 | out_prncp_inv | out_prncp | 0.999999 |
| 0 | funded_amnt | loan_amnt | 0.999999 |
| 8 | total_pymnt_inv | total_pymnt | 0.999995 |
| 2 | funded_amnt_inv | funded_amnt | 0.999995 |
| 1 | funded_amnt_inv | loan_amnt | 0.999994 |
| 15 | num_sats | open_acc | 0.998953 |
| 11 | collection_recovery_fee | recoveries | 0.991855 |
| 14 | num_rev_tl_bal_gt_0 | num_actv_rev_tl | 0.982798 |
| 16 | tot_hi_cred_lim | tot_cur_bal | 0.975068 |
| 9 | total_rec_prncp | total_pymnt | 0.959616 |
| 10 | total_rec_prncp | total_pymnt_inv | 0.959606 |
| 18 | total_il_high_credit_limit | total_bal_il | 0.952186 |
| 4 | installment | funded_amnt | 0.945134 |
| 3 | installment | loan_amnt | 0.945133 |
| 5 | installment | funded_amnt_inv | 0.945054 |
| 17 | total_bal_ex_mort | total_bal_il | 0.900038 |
| 13 | mths_since_recent_revol_delinq | mths_since_recent_bc_dlq | 0.890643 |
| 19 | total_il_high_credit_limit | total_bal_ex_mort | 0.878709 |
| 12 | mths_since_recent_revol_delinq | mths_since_last_delinq | 0.862655 |

| | |
|---|---|
| **Category 1: "Length of Time borrowers can take a pause from paying back"** | 1. hardship_length [SELECTED]<br>2. deferral_term |
| **Category 2: "Co-borrower's Credit Score rating bounds"** | 1. sec_app_fico_range_high [SELECTED]<br>2. sec_app_fico_range_low |
| **Category 3: "Main borrower's Credit Score rating bounds"** | 1. fico_range_high [SELECTED]<br>2. fico range_low |
| **Category 4: "Remaining Outstanding Principal Amount"** | 1. out_prncp [SELECTED]<br>2. out_prncp_inv |
| **Category 5: "Loan amount borrowed"** | 1. funded_amnt<br>2. loan_amnt [SELECTED]<br>3. funded_amnt_inv<br>4. installment |
| **Category 6: "Total Payment received to date for the loan** | 1. total_pymnt_inv<br>2. total_pymnt<br>3. total_rec_prncp [SELECTED] |
| **Category 7: "Number of Accounts"** | 1. num_sats [SELECTED]<br>2. open_acc |
| **Category 8: "Recovery Fees"** | 1. collection_recovery_fee [SELECTED]<br>2. recoveries |
| **Category 9: "Number of Total Revolving Credit Accounts"** | 1. num_rev_tl_bal_gt_0 [SELECTED]<br>2. num_actv_rev_tl |
| **Category 10: "Total Current balance of Credit Accounts"** | 1. tot_hi_cred_lim [SELECTED]<br>2. tot_cur_bal |
| **Category 11: "Total Balance of Installments"** | 1. total_il_high_credit_limit [SELECTED]<br>2. total_bal_il<br>3. total_bal_ex_mort |
| **Category 12: "Number of months since last delinquent defaulting"** | 1. mths_since_recent_revol_delinq [SELECTED]<br>2. mths_since_recent_bc_dlq<br>3. mths_since_last_delinq |

esade

# 1.5: Removal of Columns

## Issues Log from Columns (Cumulative)

| S/N | Variables Affected | Description of Issue | Planned Fix / Remarks | Order of Fix in Code |
|---|---|---|---|---|
| **8** | Variables in **RED** from the previous slide | Highly correlated variables in Section 1.4 | Remove them, keeping only the selected 12 variables highlighted in yellow as it represents a domain meaning | **#1** |
| **9** | url,<br>desc,<br>sub_grade,<br>emp_title,<br>title,<br>zip_code,<br>addr_state<br>application_type,<br>pymnt_plan | Irrelevant categorical variables | Remove all of them | **#2** |
| **10** | id,<br>Unnamed: 0,<br>policy_code | Irrelevant numerical variables | Remove all of them | **#3** |
| **11** | 'hardship_loan_status', 'hardship_reason', 'hardship_status', 'hardship_dpd', 'hardship_length', 'hardship_type', 'hardship_end_date', 'hardship_start_date', 'deferral_term', 'payment_plan_start_date', 'orig_projected_additional_accrued_interest', 'hardship_payoff_balance_amount', 'hardship_last_payment_amount', 'hardship_amount',<br><br>'sec_app_revol_util', 'revol_bal_joint', 'sec_app_open_act_il', **'sec_app_earliest_cr_line**', 'sec_app_open_acc', 'sec_app_chargeoff_within_12_mths', 'sec_app_inq_last_6mths', 'sec_app_fico_range_high', 'sec_app_num_rev_accts', 'sec_app_fico_range_low', 'sec_app_collections_12_mths_ex_med', 'sec_app_mort_acc', **'verification_status_joint**', 'dti_joint', 'annual_inc_joint', 'mths_since_last_record', **'next_pymnt_d',** 'mths_since_recent_bc_dlq', 'mths_since_last_major_derog', 'mths_since_recent_revol_delinq', 'mths_since_last_delinq' | Columns with more than 50% missing values | Remove all of them | **#4** |
| **12** | 'term' | Not object but float variable | Convert from object to float,<br>But we want it to be a binary variable first during the model training | **# Last** |

esade

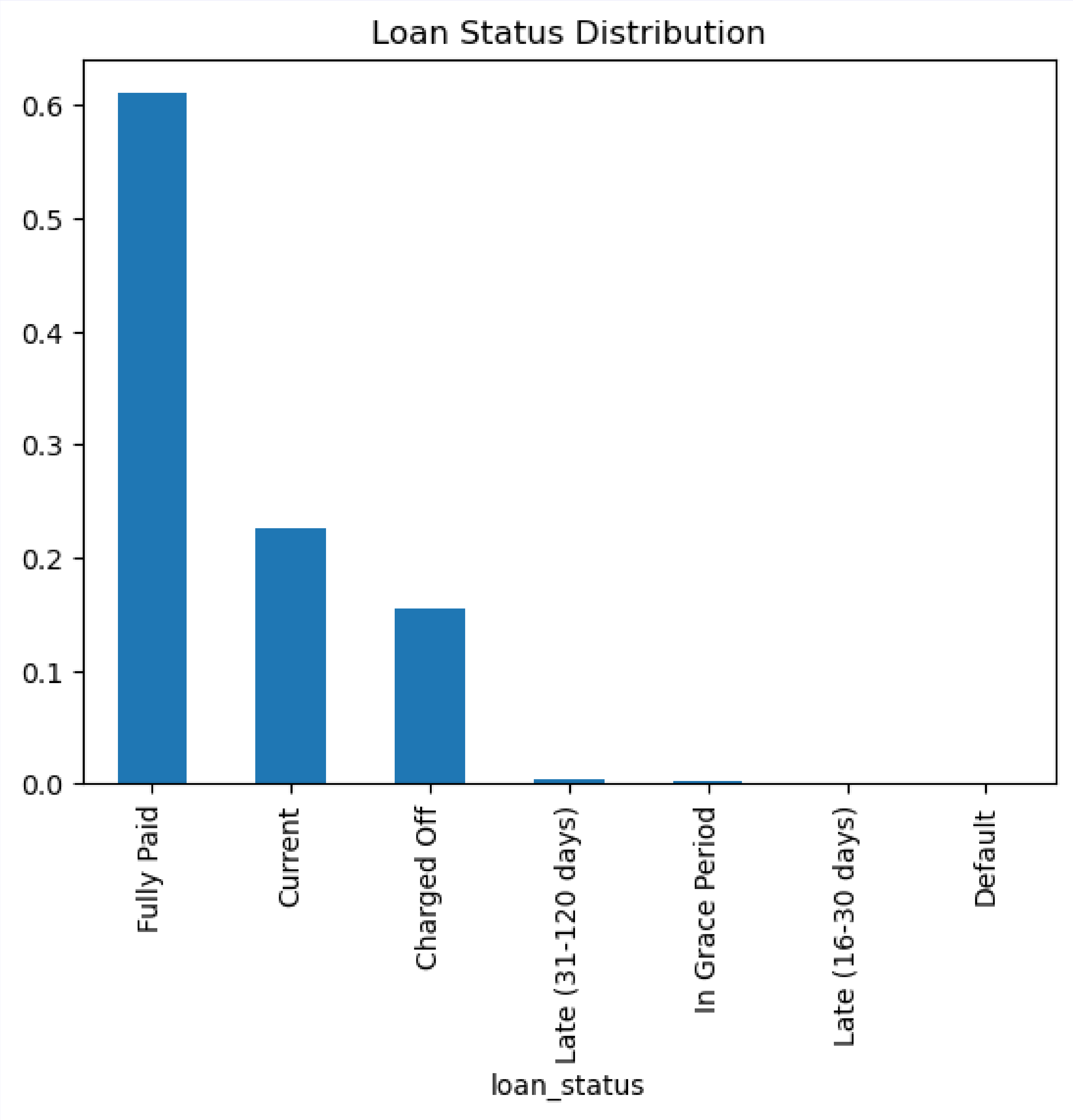


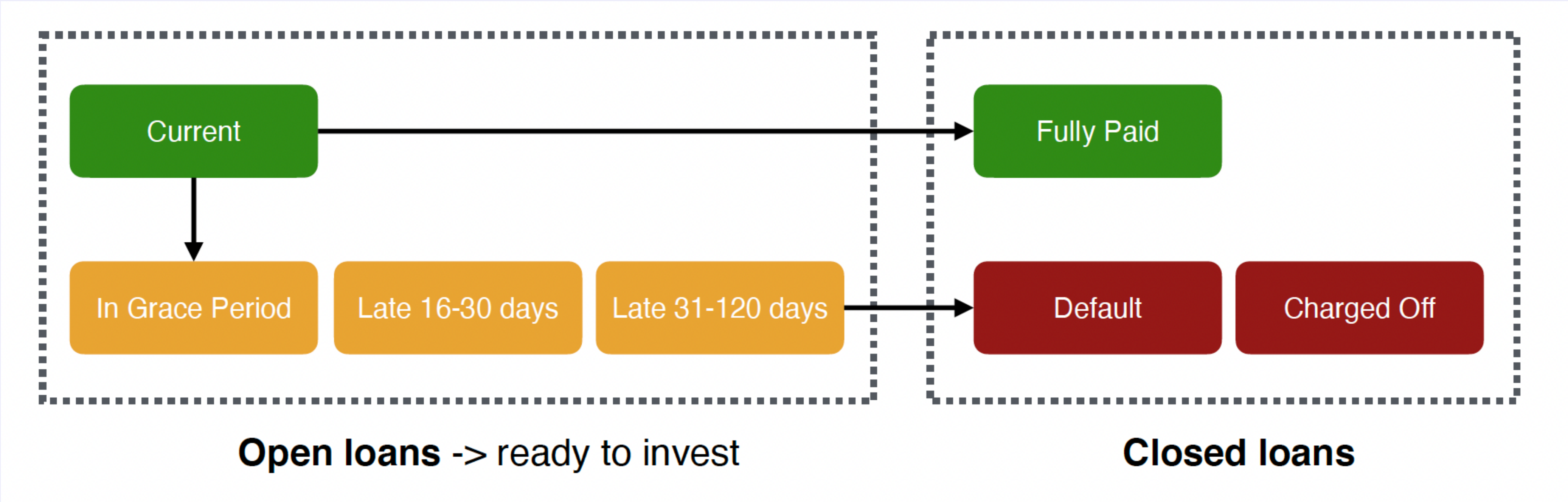
# 1.6: Conversion of Column Types

## Issues Log from Columns (Cumulative)

| S/N | Variables Affected | Description of Issue | Planned Fix / Remarks | Order of Fix in Code |
|---|---|---|---|---|
| 1 | 'next_pymnt_d'<br>'verification_status_joint'<br>'sec_app_earliest_cr_line' | Columns 48, 58, 117 have mixed type | Check for specific dtypes and convert to what make sense | **Included in Step #4** |
| 2 | 'int_rate' | Representation of percentage: e.g. 7.97% | Need to divide by 100, remove % | **#5** |
| 3 | 'loan_status' | Need to feature engineer into '1' and '0' for binary classifications | To filter Good and Bad loans based on Closed and Open Loans respectively, into 2 separate dataframes | **#7<br>(See Section 1.7)** |
| 4 | 'int_rate'<br>'revol_util' | Not object but float variable | Convert from object to float | **# 6** |
| 5 | url,<br>desc,<br>Sub_grade,<br>Emp_title,<br>Title,<br>Zip_code,<br>Addr_state | Removal due to lack of semantic meaning, high dimensionality, already have a parent variable, or is a 'zero-variance' column | See data_cleaning.ipynb for more information | **Included in Step #2** |
| 6 | 1. issue_d<br>2. earliest_cr_line<br>3. last_pymnt_d<br>4. next_pymnt_d (> 50% missing values, removed)<br>5. last_credit_pull_d<br>6. sec_app_earliest_cr_line (> 50% missing values, removed)<br>7. hardship_start_date (> 50% missing values, removed)<br>8. hardship_end_date (> 50% missing values, removed)<br>9. payment_plan_start_date ((> 50% missing values, removed) | Datetime object requires feature engineering.<br>We also notice its in mmm-YYYY format | Convert from object to datetime | **# 6** |

# 1.7: Dissecting ‘Loan Status’



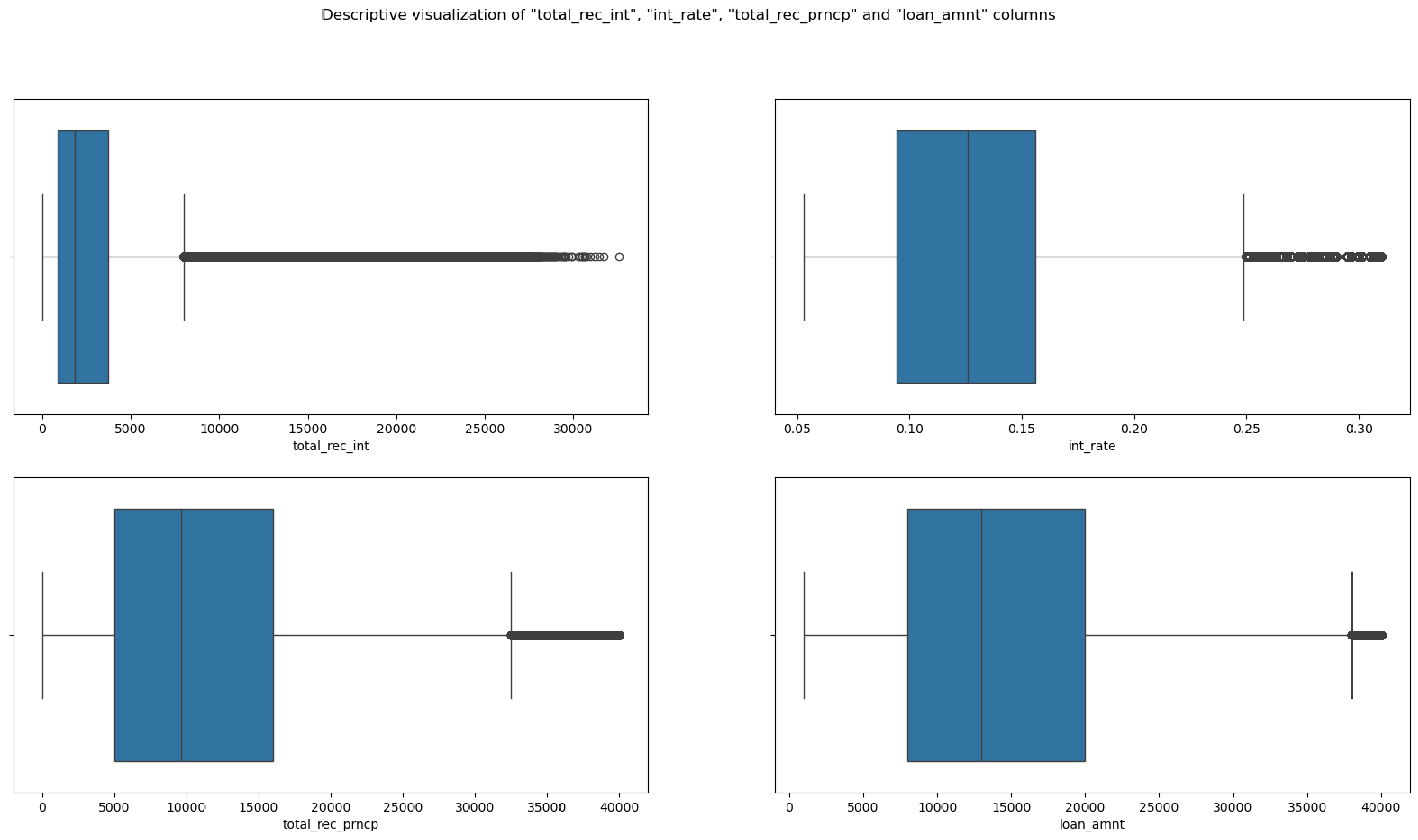

**Step 1: Filter by the values in ‘loan_status’ based on the dotted box, to form a dataframe for Closed Loans and Open Loans respectively.**

1. Open loans: df_open → Stored to parquet file
2. Closed loans: df_closed → Stored to parquet file

**Step 2: Feature engineer a ‘loan_default’ column for binary classification from ‘loan_status’ variable, and removing ‘loan_status’ at the end:**

1. In open loans, ‘Current’ (good outcomes) is class 0 and ‘In Grace Period’ , ‘Late 16-30 days’ and ‘Late 31-120 days’ (bad outcomes) are class 1.
2. In closed loans ‘Fully Paid’ (good outcomes) is class 0 and ‘Default’ and ‘Charged Off’ (bad outcomes) are class 1.

# 1.8: Visualizing Key Variables for Case




**Understanding distribution for eventual mapping of confusion matrix values to business function:**

1. `total_rec_int` will be used in closed loans, for FP (missed revenue).
2. `loan_amnt` will be used in open and closed loans to map FP FN and TN values.
3. `total_rec_prncp` will be used closed loans, for FN (unexpected loss)
4. `int_rate` will be used in both open loans to calculate good loans (with `term`) corresponding to FP and TN values.

# 02

# Model Creation – Closed Loans

**For Parts 02 and 03, we refer only to AI2_Assignment1_closed_loans.ipynb exclusively**

# 2.1: Removing Features causing Data Leakage

Dataframe: df_closed

| | |
|---|---|
| **Category 1: Payment Information Statuses** | 1. \`out_prncp\` → Remaining principal balance (not available at loan origination)<br>2. \`total_rec_prncp\` → Total principal received (post-loan payments)<br>3. \`total_rec_int\` → Total interest received (depends on borrower's repayment)<br>4. \`total_rec_late_fee\` → Late fees received (reveals delinquency behavior)<br>5. \`collection_recovery_fee\` → Collection fees applied after default<br>6. \`last_pymnt_amnt\` → Amount of last payment (only known after loan approval) |
| **Category 2: Credit Information** | 1. \`last_fico_range_high\` → Updated FICO score (should use initial FICO)<br>2. \`last_fico_range_low\` → Updated FICO score (should use initial FICO)<br>3. \`last_credit_pull_d_months\` → Last credit pull by lender (happens after loan approval)<br>4. \`last_pymnt_d_months\` → Last payment date (only available after loan approval) |
| **Category 3: Hardship and Settlement Information** | Will only know after a borrower defaults:<br>1. \`hardship_flag\` → Indicates if a hardship plan is active<br>2. \`debt_settlement_flag\` → Shows if the borrower settled the loan for less than owed |

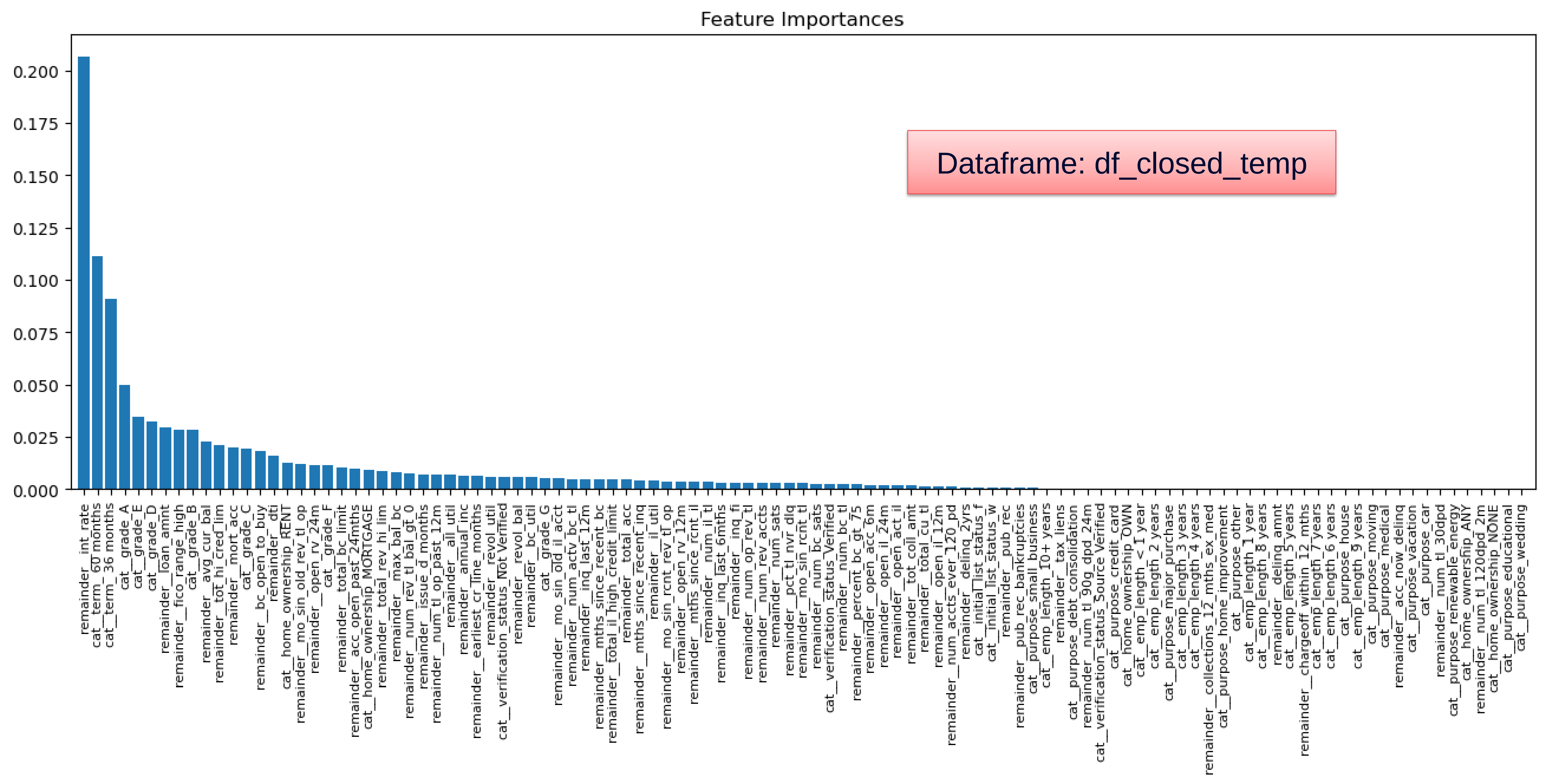
- The following columns will be removed permanently from df_closed, as these features provide information that would only be made known after a borrower repays or defaults.
- If these columns are not removed, it is suspected that they would cause data leakage as predictors to the model, translating into very accurate predictions (when it is not true)
- Hence in here, we remove the above columns to give **71 columns**, before we proceed to feature selection – to rank the importance of our existing features.

# 2.2: Feature Selection

1. The Goal is to the importance of our remaining features, to understand at face value the most important features (and their available values) and perform feature selection for potential predictors to the model that we can build.

2. To do so, all rows with missing values are dropped. As a temporary dataframe df_closed_temp, we end up with 636,000 rows (out of 1.55 million), which is still sizeable to do predictions with 83 columns.

3. After one hot encoding of categorical variables, a Random Forest Classifier is fitted into the model with an accuracy score of 0.80.



# 2.3: Feature Selection for df_closed_new

| | Feature | Importance |
|---|---|---|
| 45 | remainder__int_rate | 0.206717 |
| 1 | cat__term_ 60 months | 0.111130 |
| 0 | cat__term_ 36 months | 0.091222 |
| 2 | cat__grade_A | 0.049929 |
| 6 | cat__grade_E | 0.034572 |
| 5 | cat__grade_D | 0.032231 |
| 44 | remainder__loan_amnt | 0.029542 |
| 49 | remainder__fico_range_high | 0.028596 |
| 3 | cat__grade_B | 0.028564 |
| 73 | remainder__avg_cur_bal | 0.022939 |
| 102 | remainder__tot_hi_cred_lim | 0.021190 |
| 82 | remainder__mort_acc | 0.020130 |
| 4 | cat__grade_C | 0.019163 |
| 74 | remainder__bc_open_to_buy | 0.018412 |
| 47 | remainder__dti | 0.016163 |
| 24 | cat__home_ownership_RENT | 0.012527 |
| 79 | remainder__mo_sin_old_rev_tl_op | 0.012074 |
| 65 | remainder__open_rv_24m | 0.011680 |
| 7 | cat__grade_F | 0.011647 |
| 103 | remainder__total_bc_limit | 0.010562 |
| 72 | remainder__acc_open_past_24mths | 0.009660 |

Mean Feature Importance Score: 0.009345794392523367

| | Feature | Importance | Above Threshold |
|---|---|---|---|
| 45 | remainder__int_rate | 0.206717 | True |
| 1 | cat__term_ 60 months | 0.111130 | True |
| 0 | cat__term_ 36 months | 0.091222 | True |
| 2 | cat__grade_A | 0.049929 | True |
| 6 | cat__grade_E | 0.034572 | True |
| 5 | cat__grade_D | 0.032231 | True |
| 44 | remainder__loan_amnt | 0.029542 | True |
| 49 | remainder__fico_range_high | 0.028596 | True |
| 3 | cat__grade_B | 0.028564 | True |
| 73 | remainder__avg_cur_bal | 0.022939 | True |
| 102 | remainder__tot_hi_cred_lim | 0.021190 | True |
| 82 | remainder__mort_acc | 0.020130 | True |
| 4 | cat__grade_C | 0.019163 | True |
| 74 | remainder__bc_open_to_buy | 0.018412 | True |
| 47 | remainder__dti | 0.016163 | True |
| 24 | cat__home_ownership_RENT | 0.012527 | True |
| 79 | remainder__mo_sin_old_rev_tl_op | 0.012074 | True |
| 65 | remainder__open_rv_24m | 0.011680 | True |
| 7 | cat__grade_F | 0.011647 | True |
| 103 | remainder__total_bc_limit | 0.010562 | True |
| 72 | remainder__acc_open_past_24mths | 0.009660 | True |
| 21 | cat__home_ownership_MORTGAGE | 0.009280 | False |
| 68 | remainder__total_rev_hi_lim | 0.008453 | False |
| 66 | remainder__max_bal_bc | 0.008065 | False |
| 92 | remainder__num_rev_tl_bal_gt_0 | 0.007315 | False |

Dataframe: df_closed_temp

1. Using the automated method from from sklearn.feature_selection (import SelectFromModel), we obtain 21 columns to keep.
2. To confirm and understand how the automated selection of the 21 columns work, the mean feature importance of the 71 (+ including their additional one-hot encoded) columns is calculated.
3. In the right image, the columns above the Mean Feature Importance Score returns ‘True’ in Above Threshold; and there are 21 of these columns.

# 2.3: Feature Selection for df_closed_new

```
# Step 1: Create a dictionary to map encoded feature names to original names
mapping_dict = {
    'remainder__int_rate': 'int_rate',                              # 1
    'cat__term_ 60 months': 'term',                                 # 2
    'cat__term_ 36 months': 'term',
    'cat__grade_A': 'grade',                                        # 3
    'remainder__loan_amnt': 'loan_amnt',                            # 4
    'cat__grade_E': 'grade',
    'cat__grade_D': 'grade',
    'cat__grade_B': 'grade',
    'remainder__mort_acc': 'mort_acc',                              # 5
    'remainder__tot_hi_cred_lim': 'tot_hi_cred_lim',                # 6
    'remainder__fico_range_high': 'fico_range_high',                # 7
    'remainder__avg_cur_bal': 'avg_cur_bal',                        # 8
    'cat__grade_C': 'grade',
    'remainder__bc_open_to_buy': 'bc_open_to_buy',                  # 9
    'remainder__dti': 'dti',                                        # 10
    'cat__grade_F': 'grade',
    'remainder__open_rv_24m': 'open_rv_24m',                        # 11
    'remainder__mo_sin_old_rev_tl_op': 'mo_sin_old_rev_tl_op',      # 12
    'cat__home_ownership_RENT': 'home_ownership',                   # 13
    'remainder__acc_open_past_24mths': 'acc_open_past_24mths',      # 14
    'remainder__total_bc_limit': 'total_bc_limit'                   # 15
}
```

```
Contribution of Top 21 Features: 0.7986512377109438
```

Dataframe: df_closed_temp

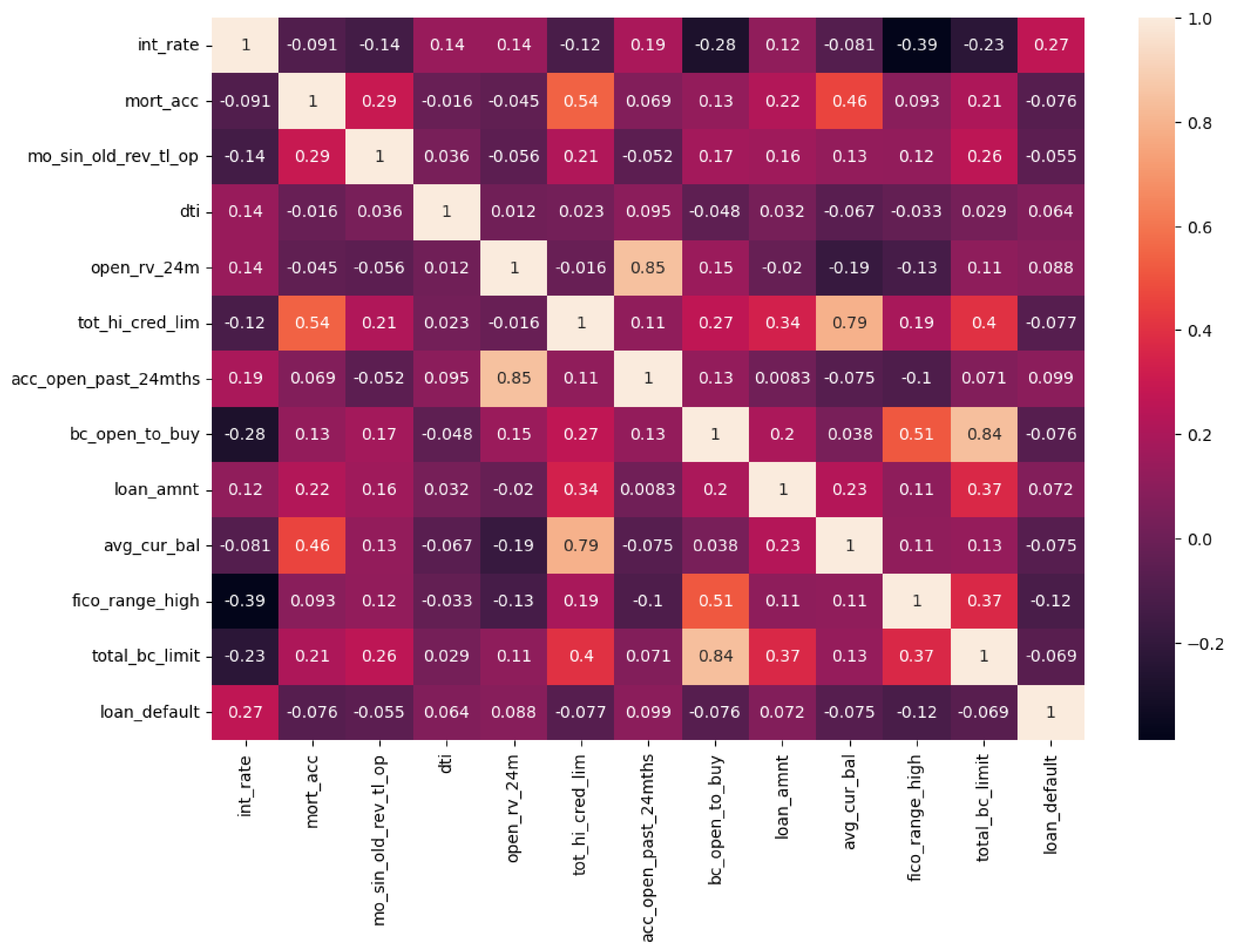
1. However, because of sk_learn preprocessor, the original features' names have additional suffixes for categorical variables indicating their values.
2. All 21 columns are inspected manually to view their original column name. They are then reassigned a number, where a dictionary is created to map these feature names back to their original names.
3. In the end, 15 original remaining features are wholly/partially identified. We then store the selected features list into a pickle file (to use in the Python Notebook for open loans later).

esade



# 2.4: Model Training with Focused Features

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1553106 entries, 0 to 1553105
Data columns (total 15 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   int_rate              1553106 non-null  float64
 1   mort_acc              1553106 non-null  float64
 2   mo_sin_old_rev_tl_op  1553106 non-null  float64
 3   dti                   1552205 non-null  float64
 4   open_rv_24m           934630 non-null   float64
 5   tot_hi_cred_lim       1553106 non-null  float64
 6   grade                 1553106 non-null  object
 7   home_ownership        1553106 non-null  object
 8   acc_open_past_24mths  1553106 non-null  float64
 9   bc_open_to_buy        1535397 non-null  float64
 10  loan_amnt             1553106 non-null  float64
 11  avg_cur_bal           1553074 non-null  float64
 12  term                  1553106 non-null  object
 13  fico_range_high       1553106 non-null  float64
 14  total_bc_limit        1553106 non-null  float64
dtypes: float64(12), object(3)
memory usage: 177.7+ MB
```



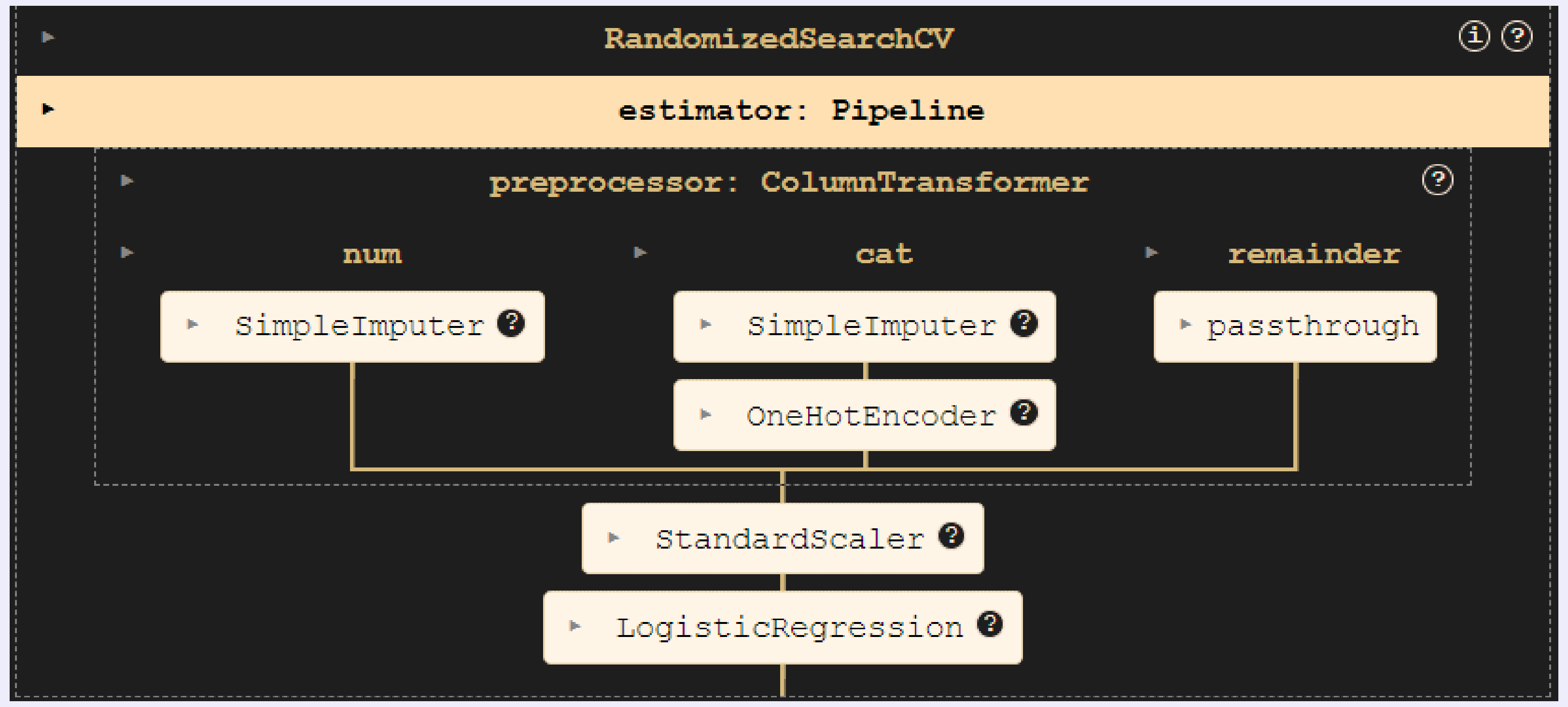

Dataframe: df_closed_new

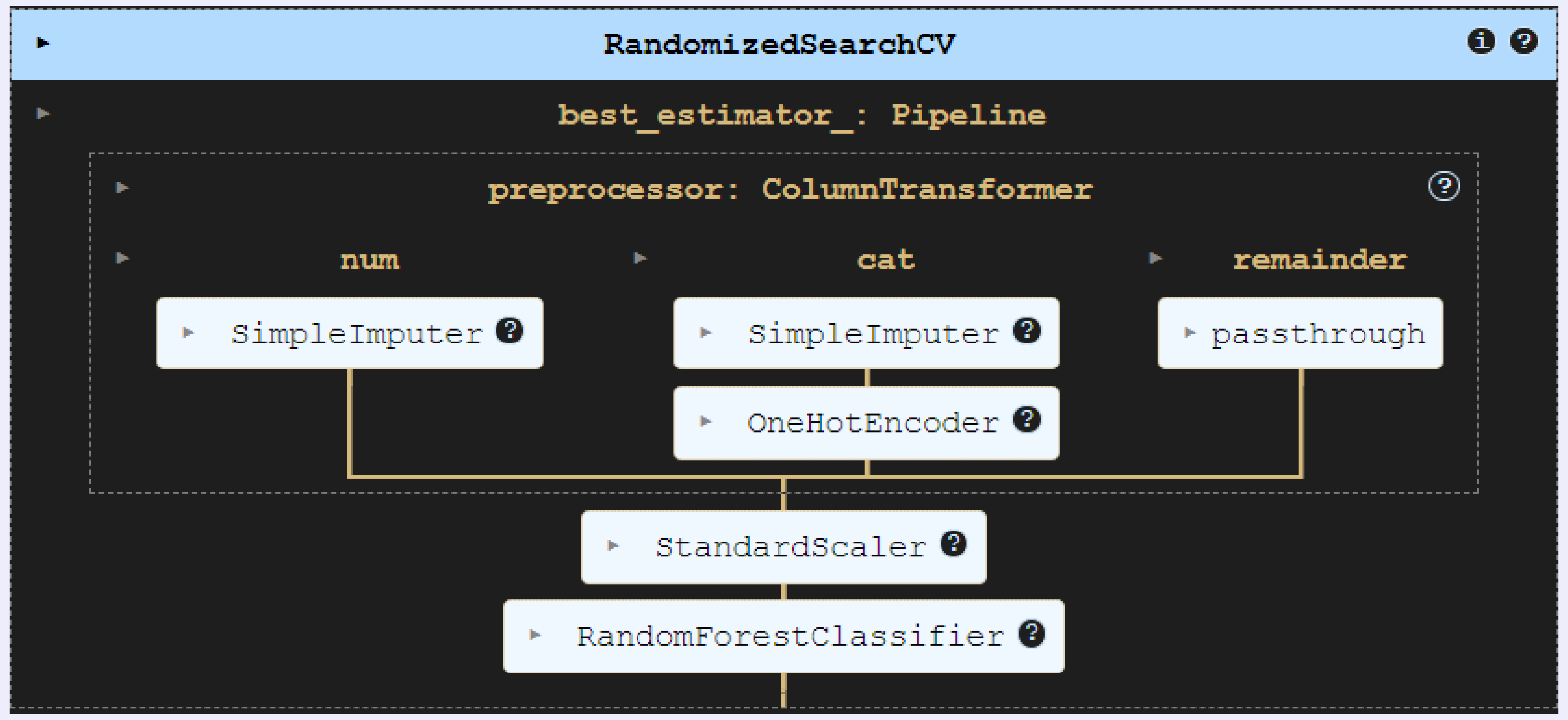
```
y.value_counts(normalize=True)
✓ 0.0s

loan_default
0    0.797814
1    0.202186
Name: proportion, dtype: float64
```

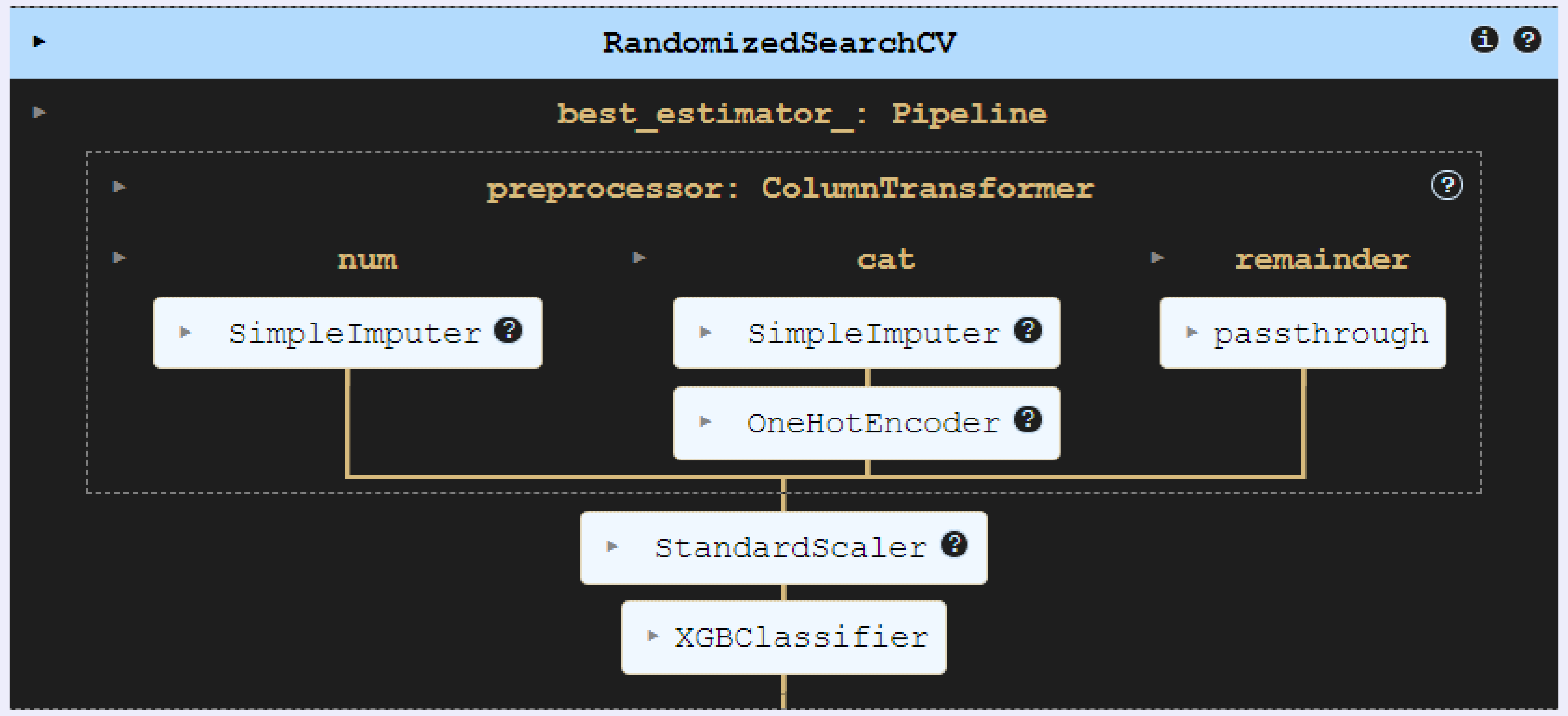
```
# check y counts
y.value_counts()
✓ 0.0s

loan_default
0    1239089
1     314017
Name: count, dtype: int64
```

1. After selecting the 15 original features, the df_closed_new is finalized for our eventual model training and selection, with the correlation matrix confirming there are no highly correlated features within feature matrix ‘X’
2. Upon inspection of ‘X’, ‘tot_hi_cred_lim’ and ‘bc_open_to_buy’ has missing values, which will be imputed with median inside the preprocessor in the next step.
3. For target ‘y’, there are also about 20% of defaulters and 80% repayers (amongst the 1.55 million recorded instances in this df)

# 2.5: Model Selection & Hyperparameter Tuning



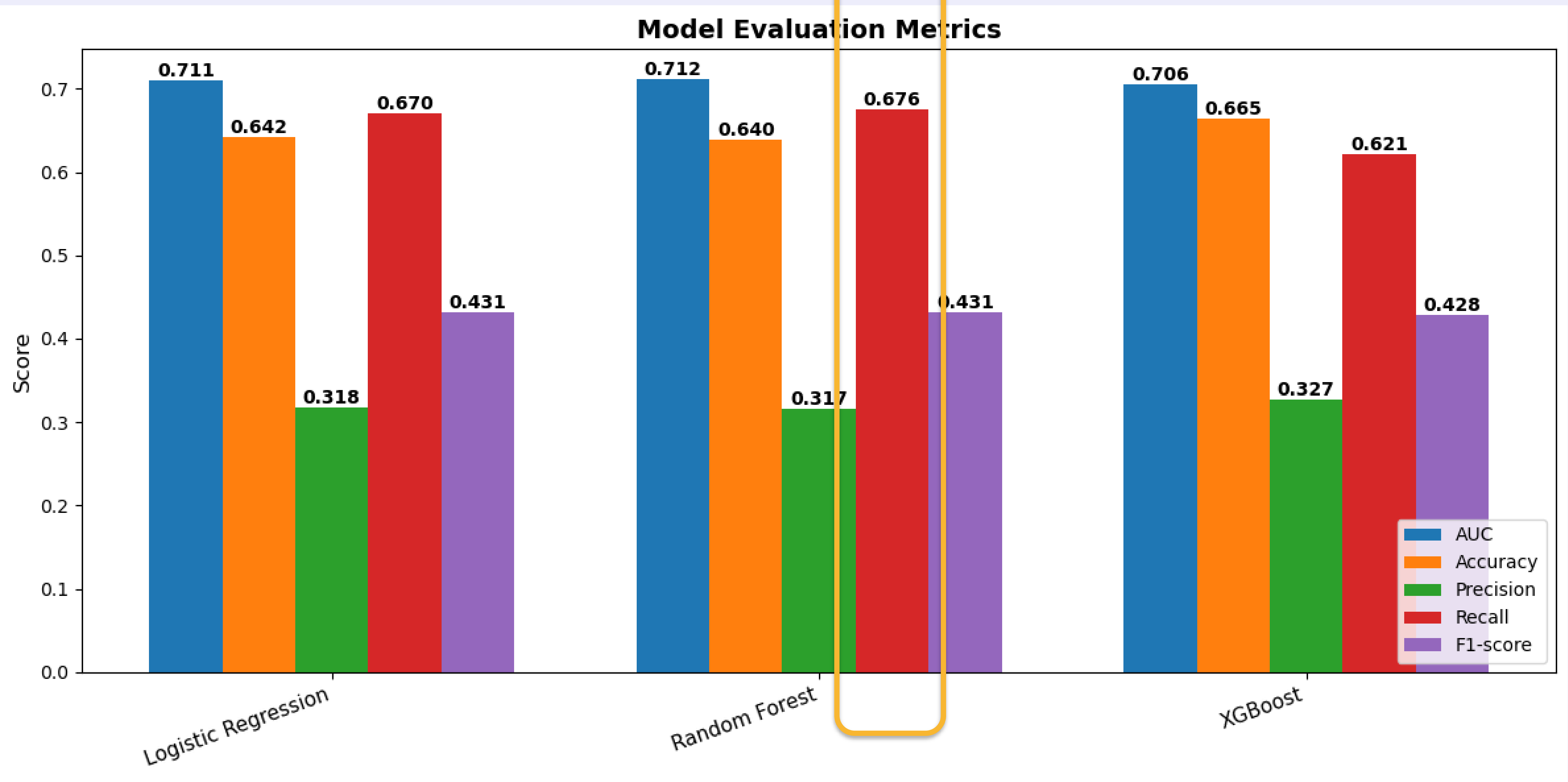




1. 3 models (Logistic Regression, Random Forest Classifier, XGboost) were chosen with Randomized Search CV after:
   a) Fitting in the preprocessor:
      i. imputes missing values by Simple Imputer with the median
      ii. one-hot-encodes the categorical values to numerical values
   b) Performing Standard Scaling

2. Rebalancing techniques to address class imbalance here was also ignored because:
   a) it distorts the natural class distribution of the data, which could misrepresent the actual risk exposure in terms of default rates and business implications
   b) our final business objective is not just classification accuracy, but also mapping the confusion matrix values financial return, we needed to maintain the real-world class balance
   c) The minority class (defaulters = 20%) were also not too few such that we needed to upsample or generate synthetic data like SMOTE just to ensure representation (for eventual model to be sensitive to detect the minority class)

3. To allow model to focus more on the minority class without altering the dataset:
   a) stratified cross-validation (StratifiedKFold) was used to ensure balanced class representation during model training
   b) scale_pos_weight = (# of non-defaults) / (# of defaults) parameter in XGBoost → to preserve the true loan distribution

# 2.5: Model Selection & Hyperparameter Tuning




1. Best Metric for Selecting the Best Model: **<u>Highest Recall</u>**
   a) Optimizing Recall reduces FN, ensuring we identify more risky borrowers and avoid issuing loans to them.
   b) False Negatives (FN) lead to financial losses. If we fail to detect a defaulting borrower, we issue a loan that will not be repaid, leading to direct monetary loss.
2. Best Model for Recall: Random-Forest (See results below)
3. Best Hyperparameters: (See results below)

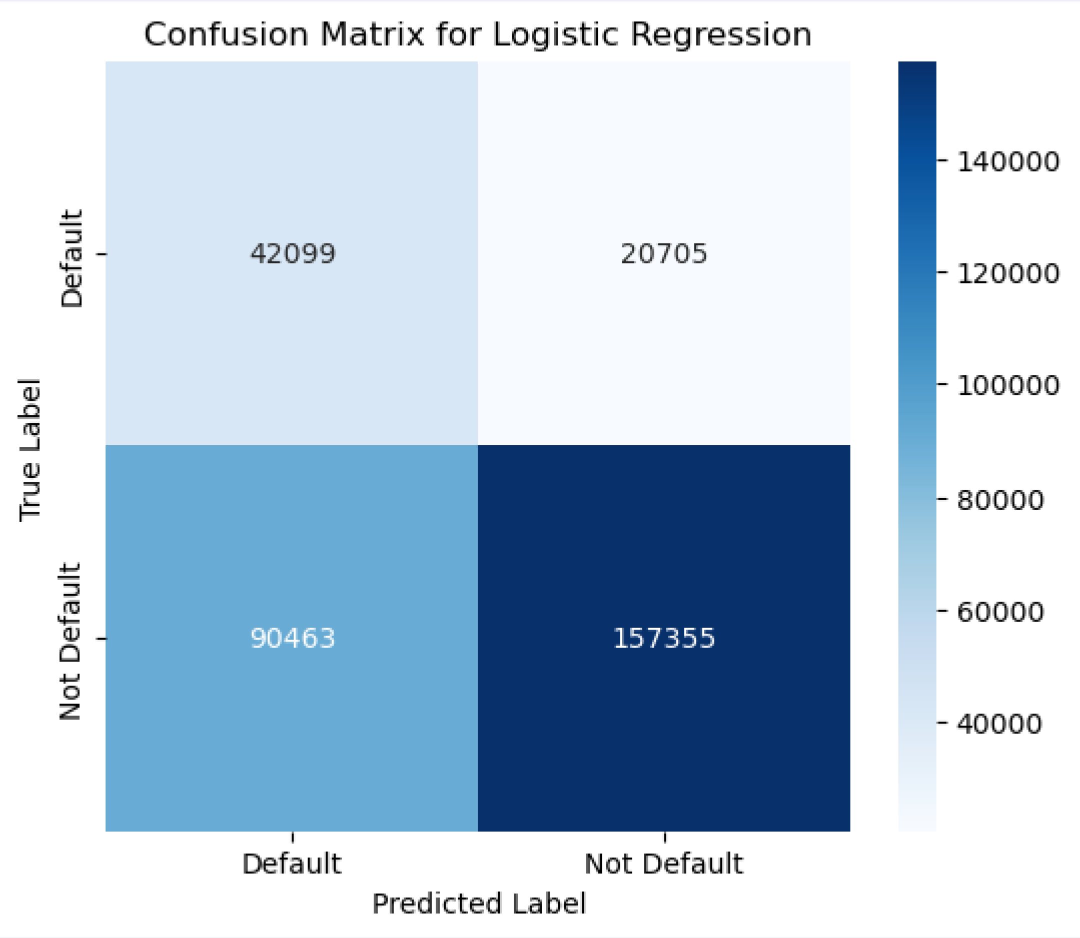
```
Best Model: Random Forest
AUC: 0.7120811955890558
Accuracy: 0.6397776075100926
Precision: 0.31676508574030804
Recall: 0.6756098337685498
F1-score: 0.43130797184315517
Predictions: [0 0 0 ... 0 0 1]
Probabilities: [0.37808455 0.22598701 0.32639066 ... 0.3229197  0.44996031 0.62679789]
```

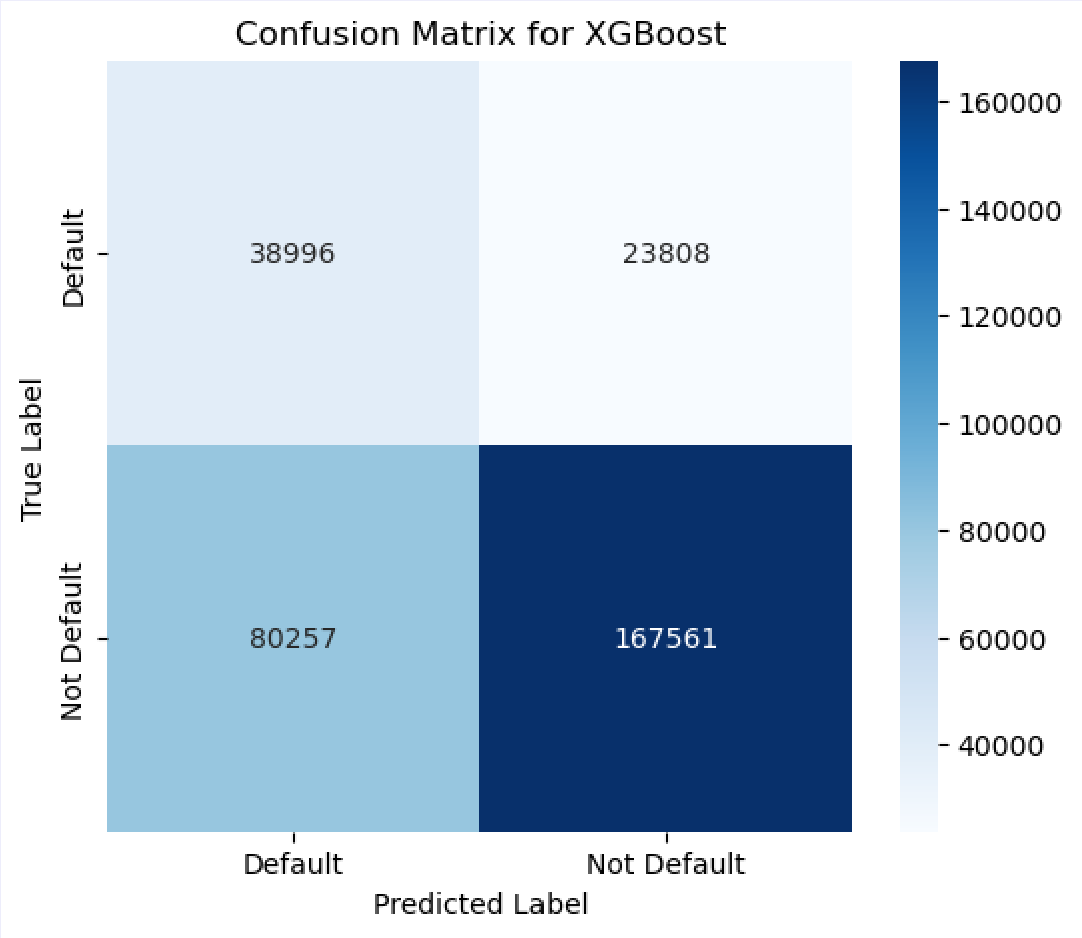
```
# Random Forest best hyperparameters
best_model.best_params_
```

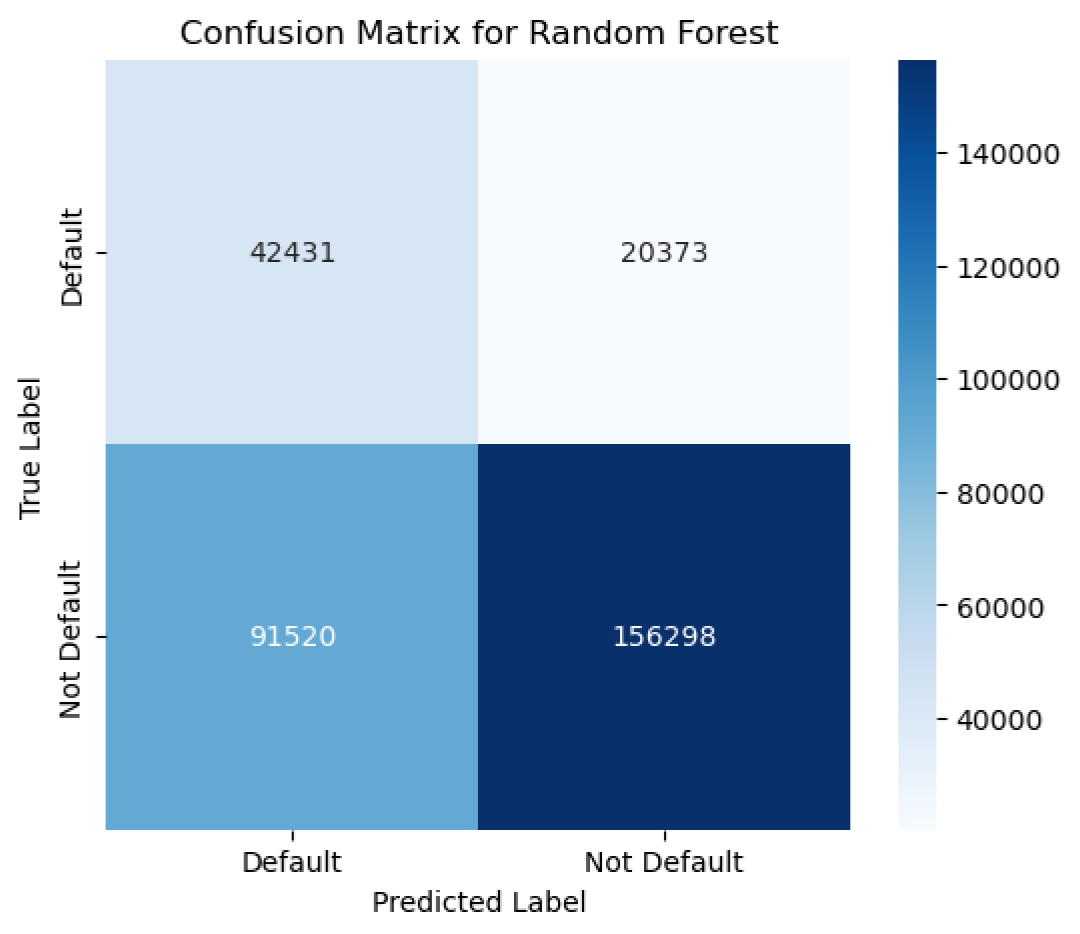
✓ 0.0s

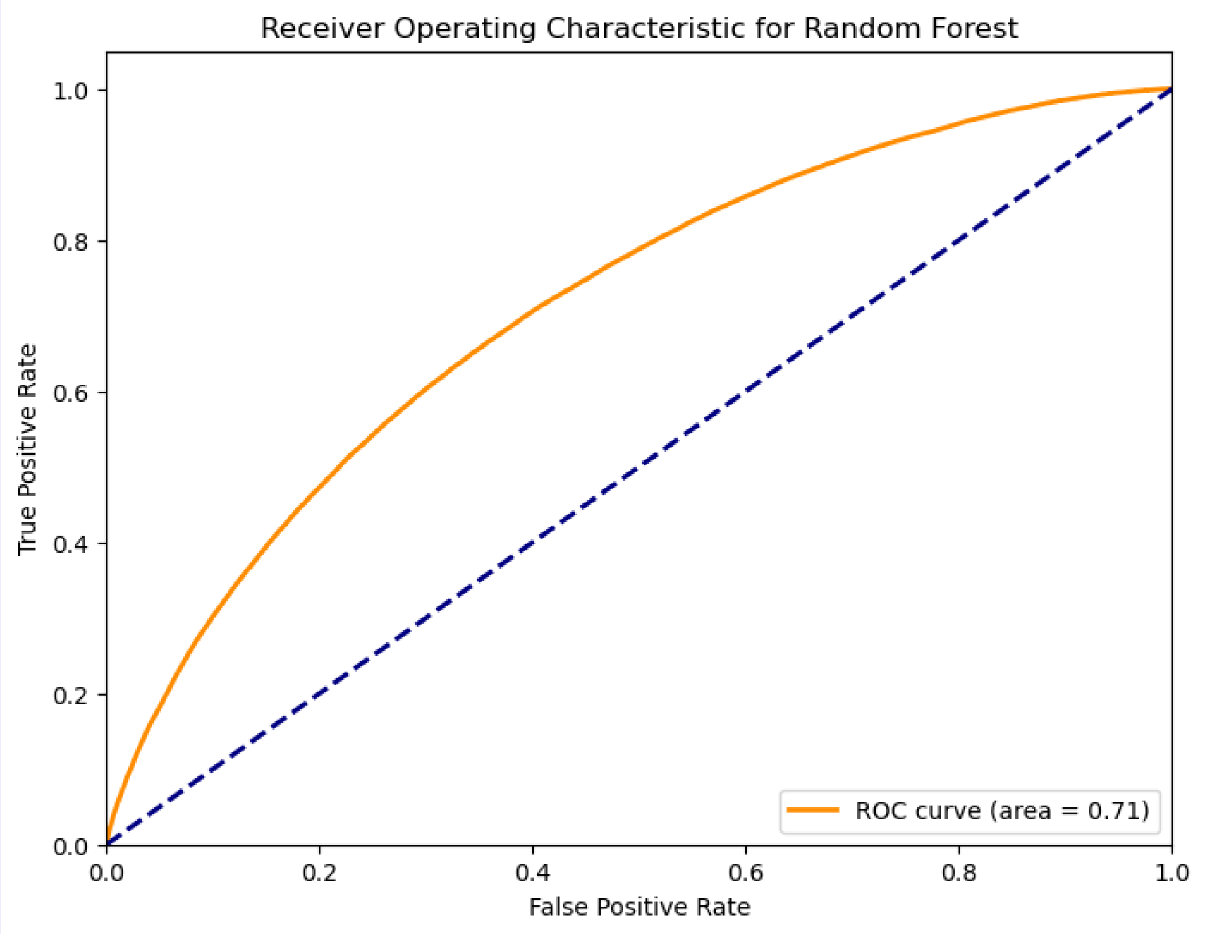
```
{'classifier__n_estimators': 100,
 'classifier__max_depth': 10,
 'classifier': RandomForestClassifier(class_weight='balanced', random_state=42)}
```

# 2.6: Collection of Model Metrics




1. Confusion Matrix of the 3 models for the lowest FN, at the standard threshold level = 0.5:
   a) Logistic Regression: 20,705
   b) XGBoost: 23,808
   c) Random Forest: 20,373
   - **We also note that the values are non-deterministic as hyperparameter tuning changes the # FNs per run.**

2. Other Metrics:
   a) Reviewing Accuracy (previous slide): XGBoost has the highest accuracy, however as of now FN is prioritized first before further threshold tuning.
   b) Additionally, we have the AUROC curve with AUC = 0.712 which is highest for Random Forest

**Best model before tuning: Random Forest**

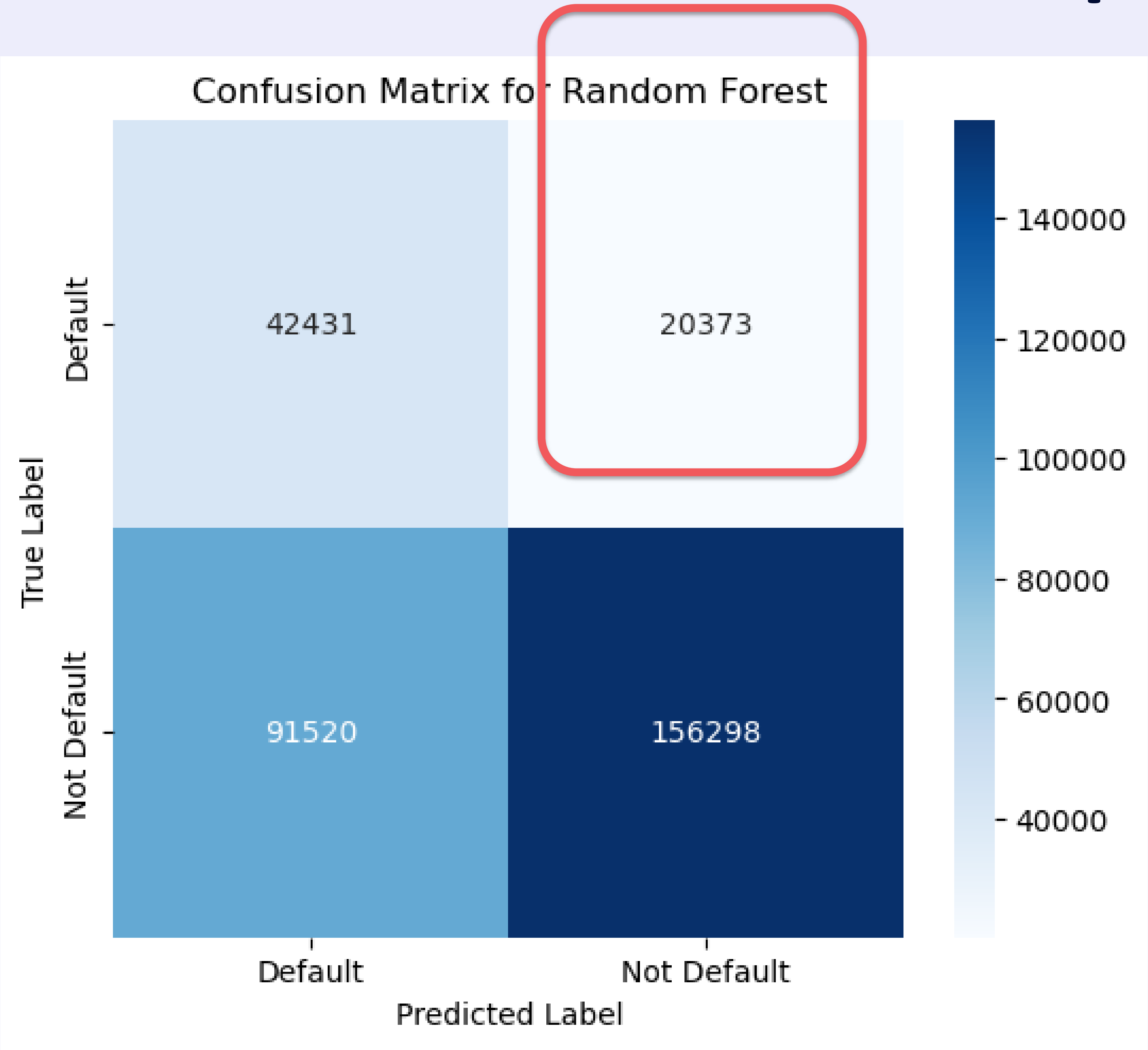
# 03

# Model Interpretation and Implementation onto Closed Loans

**For Parts 02 and 03, we refer only to AI2_Assignment1_closed_loans.ipynb exclusively**

esade

# 3.1: Business Impact for Minimized FNs




1. With the best model as Random Forest with minimized FN (at this run there are 20373 instances) , the function 'impact_test_set' aims to experiment by using the trade-off between FP-FN to model the best threshold and business impact.
2. The Goal is then to maximize the FN-FP return, given that FN is minimum in random forest.
3. However, the issue arise when Logistic Regression returns a higher FN-FP return than Random Forest. Intuitively, this signifies that all confusion matrices values should be mapped for cost-sensitive threshold tuning.
4. Additionally, threshold tuning was conducted only with FP and FN; and it was possible to just make FN have 0 loss, thus threshold == 0.

| | Missed Revenue (FP) | Default Loss (FN) | Risk-Adjusted Return (RAR) |
|---|---|---|---|
| Logistic Regression | 3.442442e+08 | 1.681735e+08 | 1.760706e+08 |
| Random Forest | 3.383267e+08 | 1.739737e+08 | 1.643530e+08 |
| XGBoost | 2.976963e+08 | 2.061516e+08 | 9.154463e+07 |

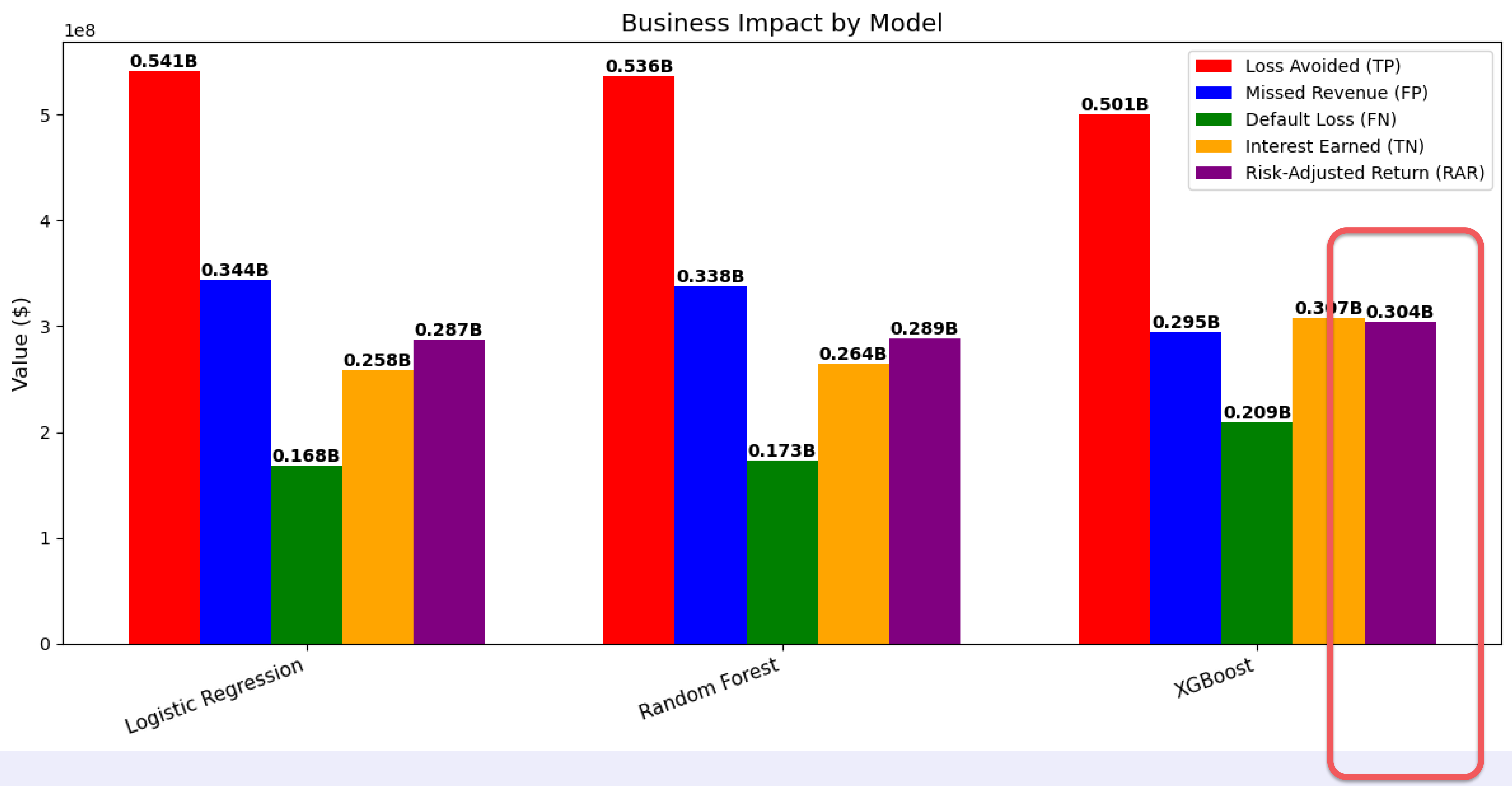
--- THRESHOLD-TUNED RESULTS ---

| | Model | Optimized Threshold | TP | FP | FN | TN | Missed Revenue (FP) | Default Loss (FN) | Risk-Adjusted Return (RAR) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Logistic Regression | 0.0 | 62804 | 247818 | 0 | 0 | 6.022936e+08 | 0.0 | 6.022936e+08 |
| 1 | Random Forest | 0.0 | 62804 | 247818 | 0 | 0 | 6.022936e+08 | 0.0 | 6.022936e+08 |
| 2 | XGBoost | 0.0 | 62804 | 247818 | 0 | 0 | 6.022936e+08 | 0.0 | 6.022936e+08 |

**Best Threshold for a FP- FN function will be 0, with 0 FNs -> Not possible**

# 3.2: Mapping Full Matrix Impact onto Closed Loans

1. Instead of looking at minimizing FN, we can map the actual FN, FP, TN, TPs to actual gain and loss functions from the feature columns in the dataset. In doing so, the goal would then be to maximize the Risk-Adjusted Returns (RAR) from the contribution of the 4-confusion matrix values (illustrated in bottom right of slide)
2. The best model would be the one that maximises the RAR → As of now XGBoost with $0.304 Billion




Columns Extracted to Map Confusion Matrix Values

**1. TP: Loss Avoided**

- ('Loan Amount' – 'Total Received Principal') x #TP

**2. FP: Missed Interest**

- 'Total Received Interest' x #FP

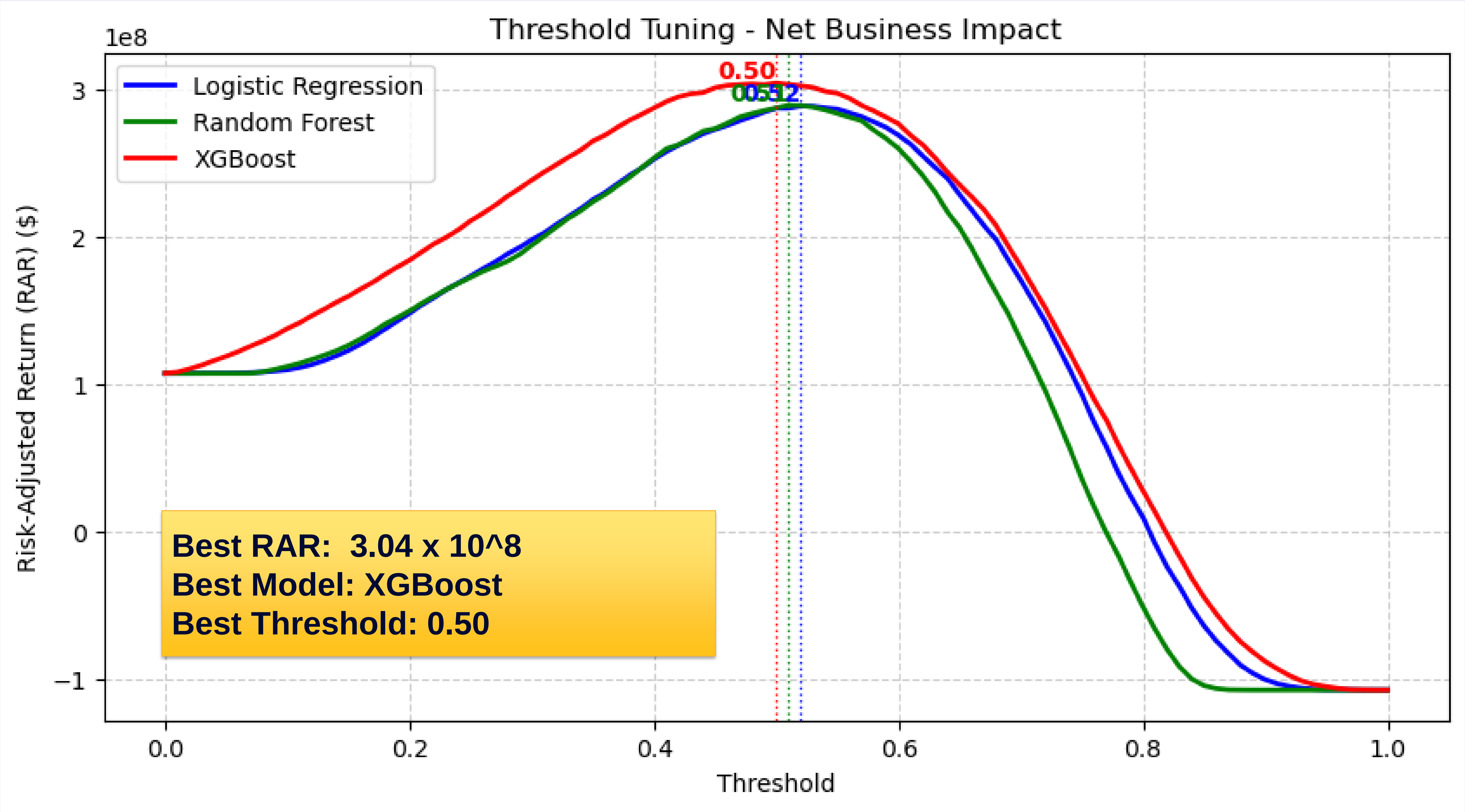
**3. FN: Unrecovered Principal**

- 'Loan Amount' – 'Total Received Principal' x #FN

**4. TN: Already Earned Interest**

- 'Total Received Interest' x #TN

# 3.3: Threshold Tuning for Cost-Sensitivity




| | Model | Optimized Threshold | TP | FP | FN | TN | Loss Avoided (TP) | Missed Revenue (FP) | Default Loss (FN) | Interest Earned (TN) | Risk-Adjusted Return (RAR) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Logistic Regression | 0.52 | 39620 | 80939 | 23184 | 166879 | 5.189541e+08 | 3.210839e+08 | 1.906081e+08 | 2.812097e+08 | 2.884719e+08 |
| 1 | Random Forest | 0.51 | 40857 | 85123 | 21947 | 162695 | 5.241114e+08 | 3.250475e+08 | 1.854508e+08 | 2.772461e+08 | 2.908592e+08 |
| 2 | XGBoost | 0.50 | 39112 | 80356 | 23692 | 167462 | 5.005306e+08 | 2.948363e+08 | 2.090316e+08 | 3.074573e+08 | 3.041202e+08 |

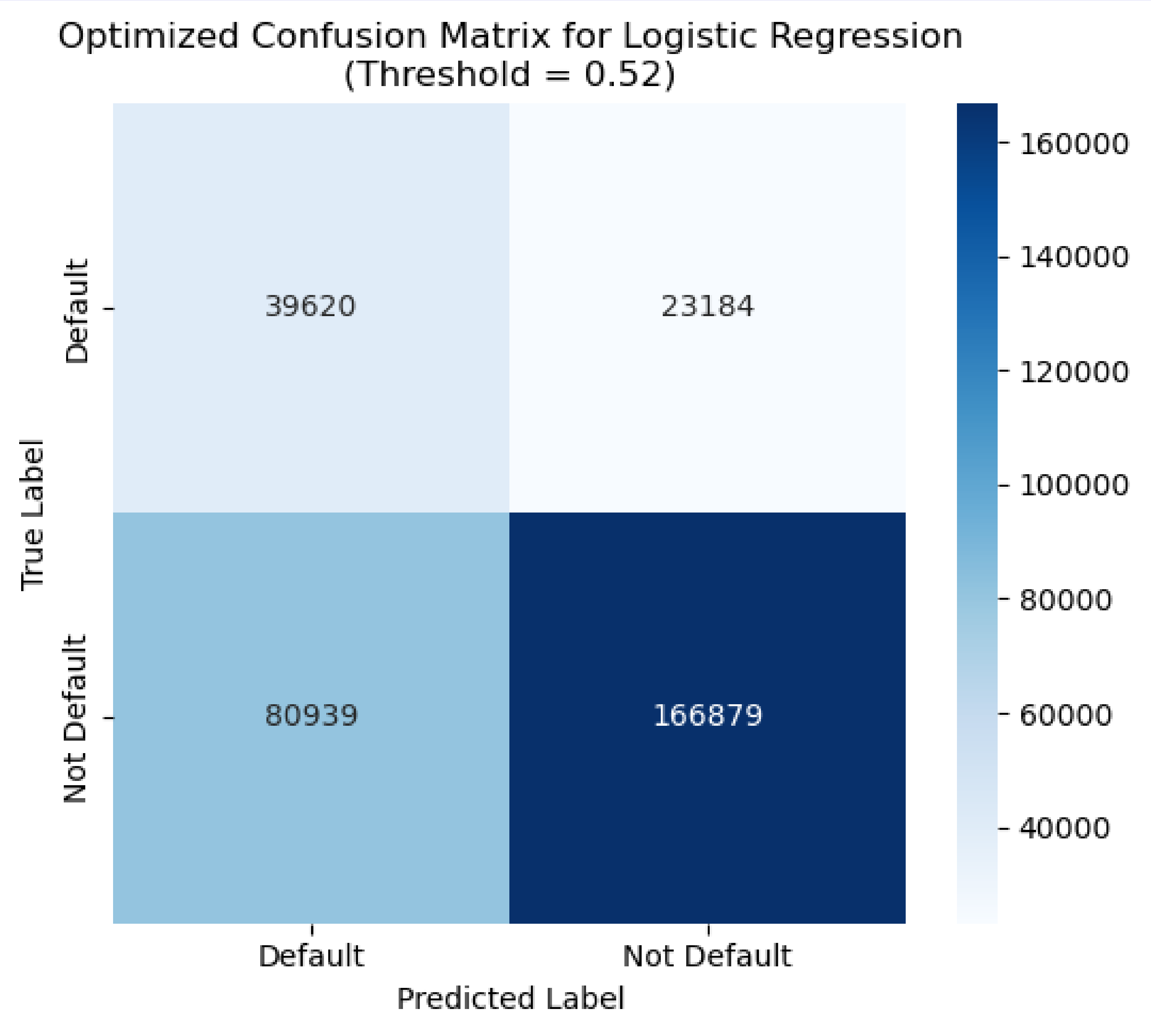
Before threshold tuning:

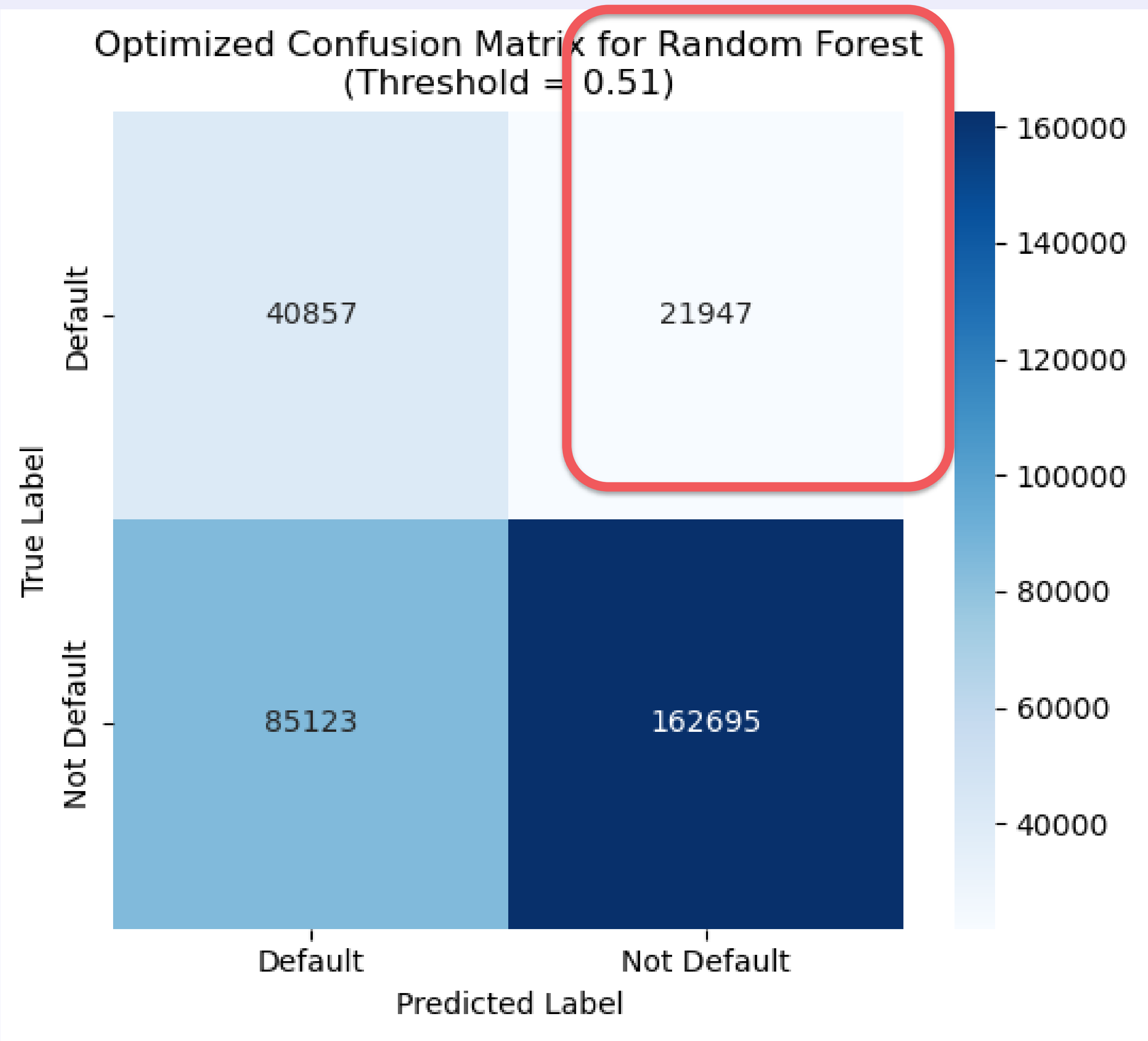
1. Predicted and actual values (confusion matrix) was based purely on classification performance metrics (e.g., accuracy, AUC, recall, etc.)
2. Does not consider financial impact (applying the values to appropriate columns, shown in the previous slide)

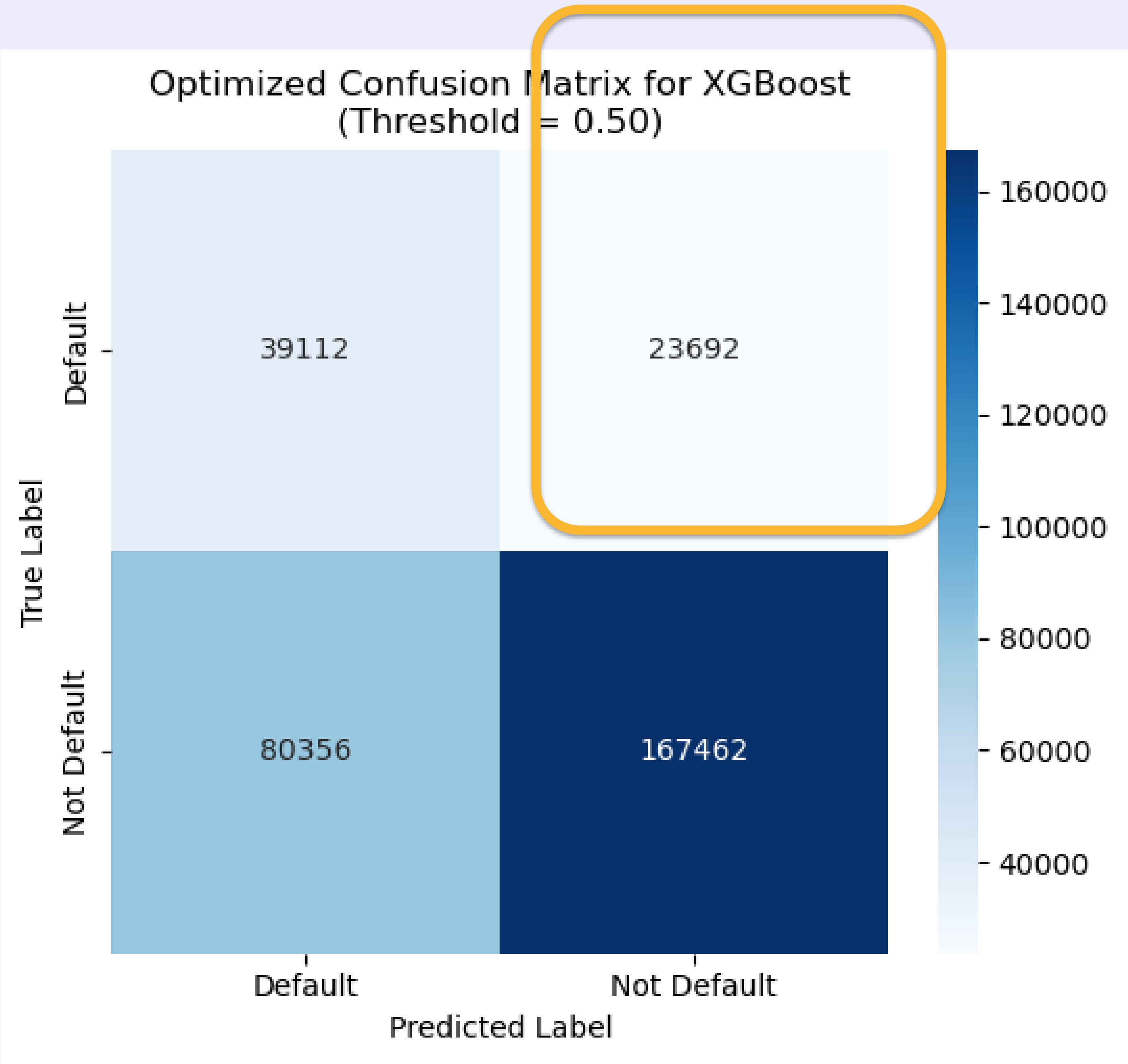
After threshold tuning:

1. Decision boundary was adjusted to optimize financial returns (RAR), which led to fewer False Positives (FP) but more False Negatives (FN) as a trade-off.
2. This shift happens because the model is now prioritizing business impact rather than pure classification performance.

# 3.3: Threshold Tuning for Cost-Sensitivity

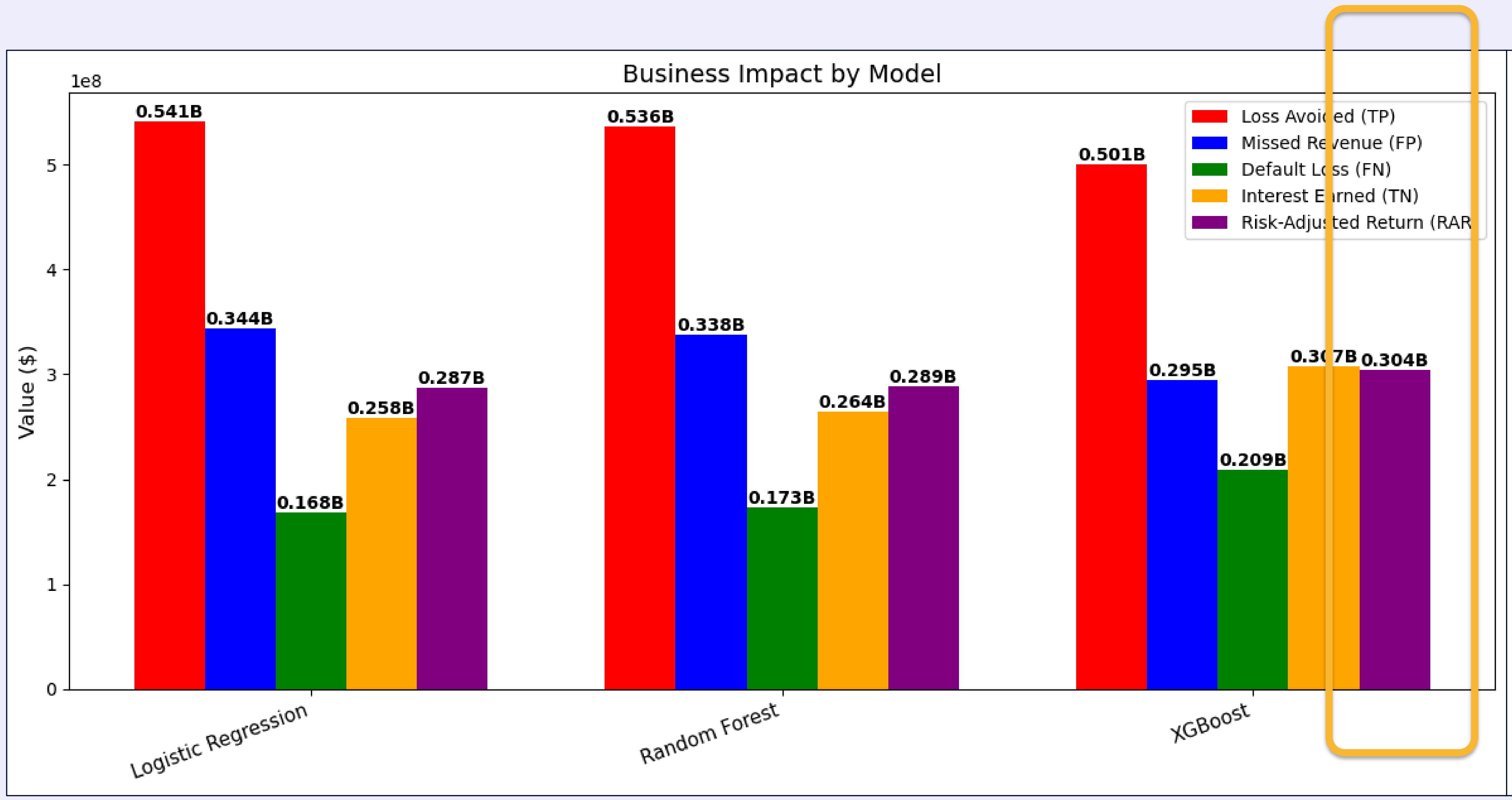


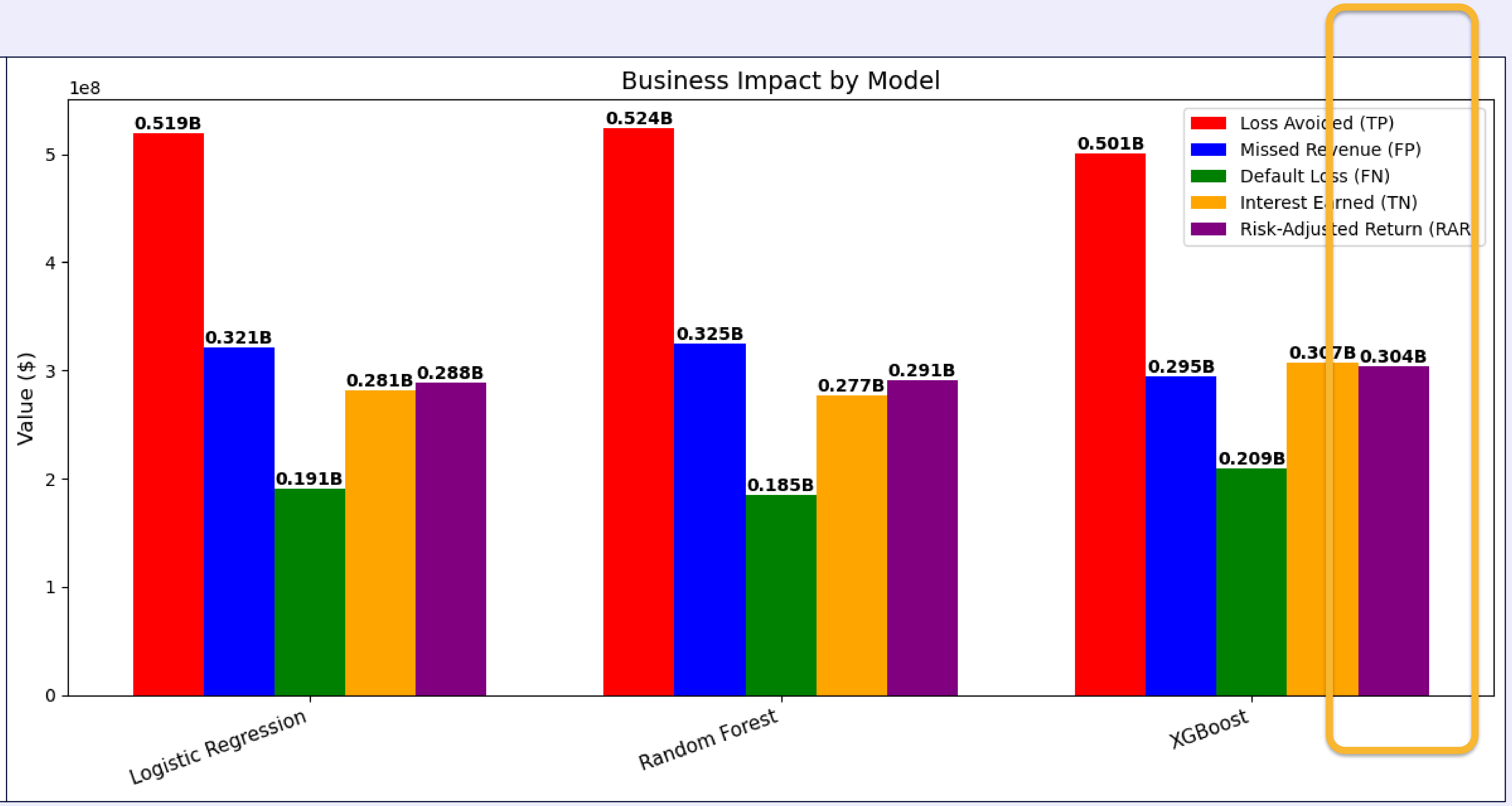




1. In performing threshold tuning, it is seen also that the number of FN classified will increase in Maximising RAR than when in trying to minimize FN. An example would be comparing the number of FNs for Random Forest in Section 3.3 and Section 3.1
2. Even though XGBoost FN (Default Loss) is slightly higher, it is compensated by lower FP (Missed Revenue) and higher TN (Interest Earned).

# 3.4: Conclusion After Threshold Tuning




- Best model XGBoost has the highest Risk-Adjusted Return (RAR) = 3.041202e+08, reflecting domain cost function.
- Threshold at == 0.5 remains the same for XGBoost before and after tuning, thereby giving the same values (unlike for other models where their RAR improves.
- Since RAR aggregates all costs and revenues, selecting model with highest RAR is the most profit-optimized for closed loans.

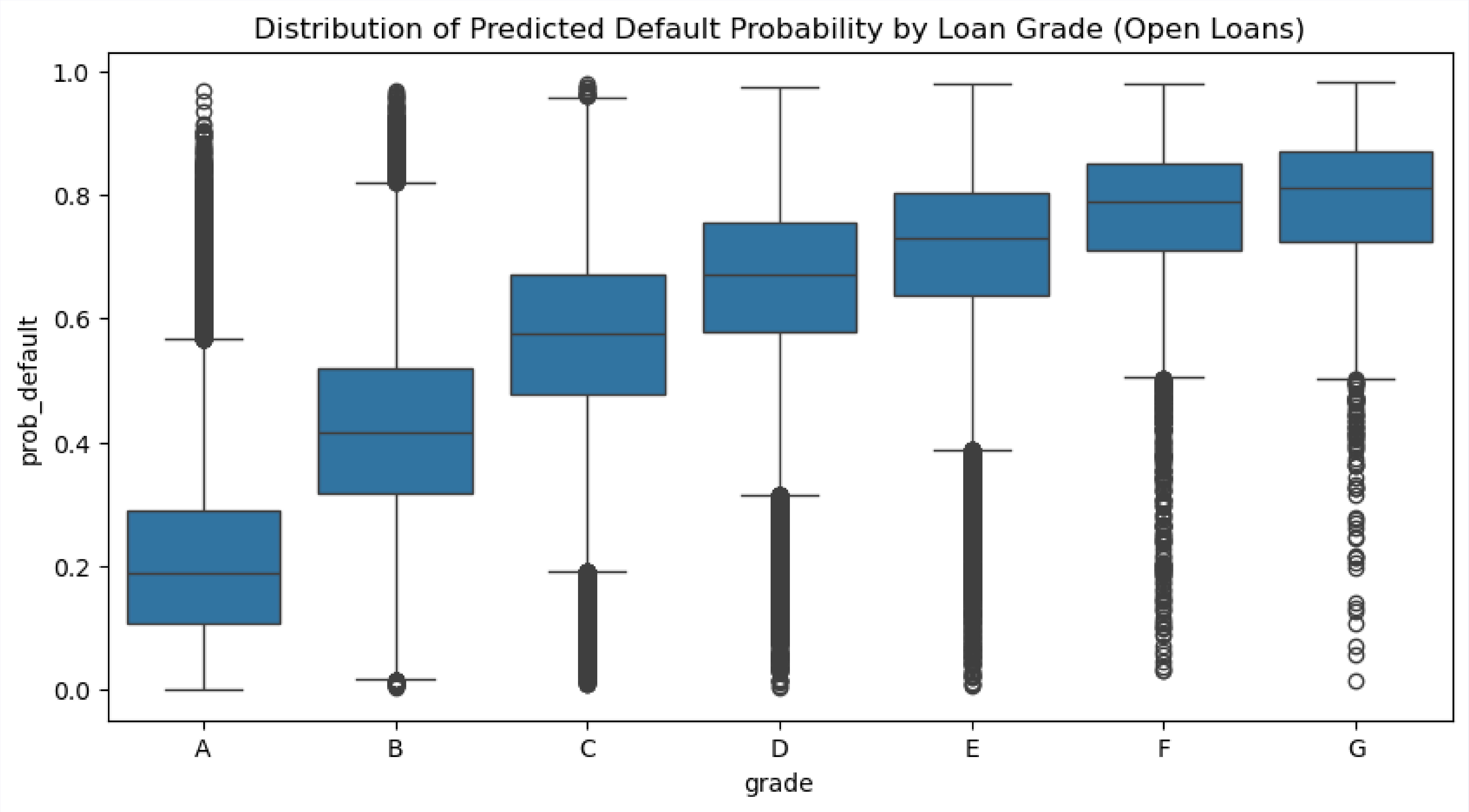
# 04

# Model Implementation Onto Open Loans

**For Parts 04, we refer only**
**to AI2_Assignment1_open_loans.ipynb exclusively**

# 4.1: Initializing Dataframe for Open Loans

1. Creating df_open_new dataframe with the following features:
   a) Selected features used in closed loans model training
   b) Merged 'Loan Default' and its binary values encoded for open loans context
   c) 'Prob_Default' → Predicted probability of default in open loans
   d) 'Pred_Default' → Convert value in 'Prob_Default' to 0/1 binary class
   e) Convert 'term' from categorical to numerical variable for math operations later

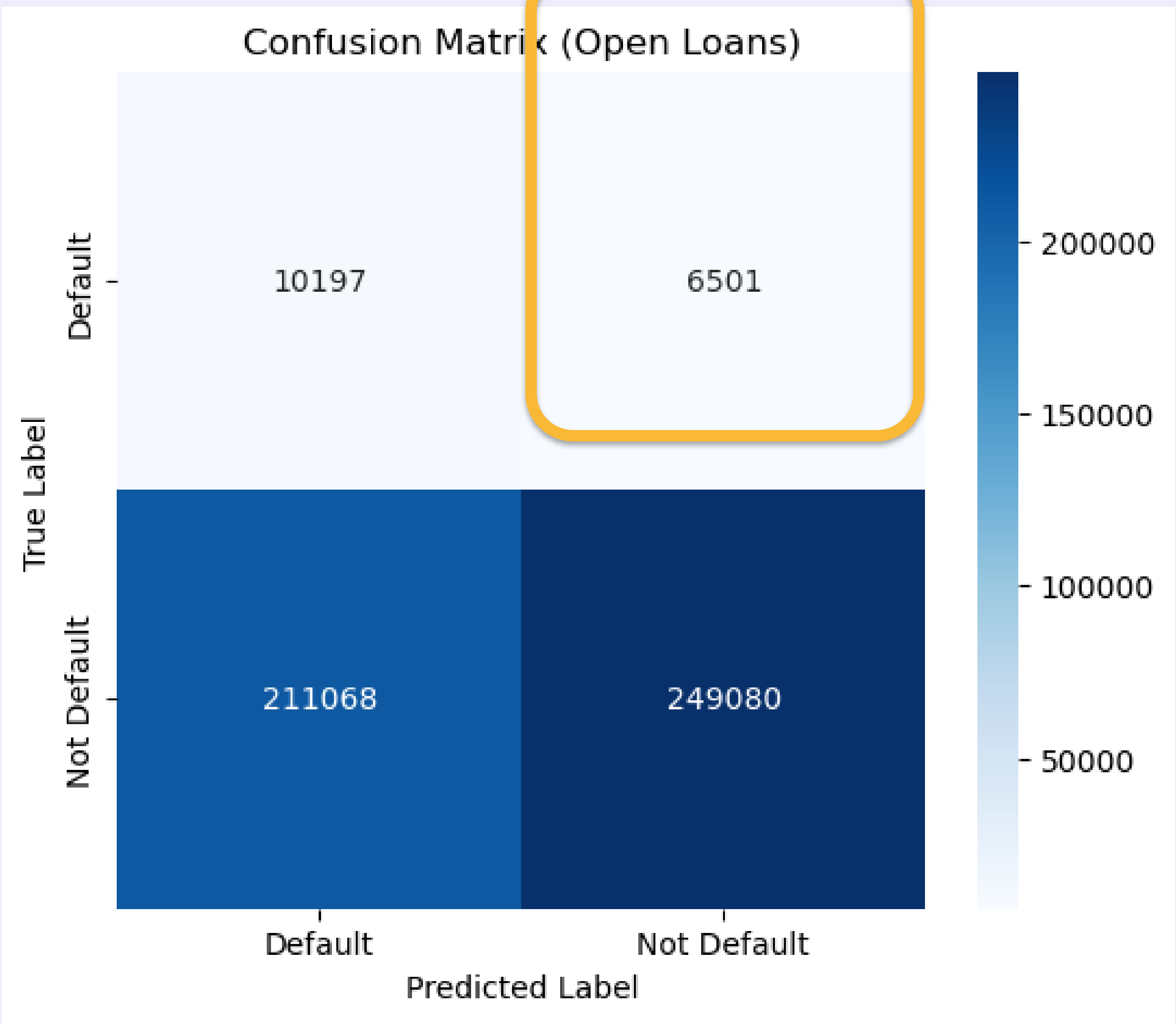


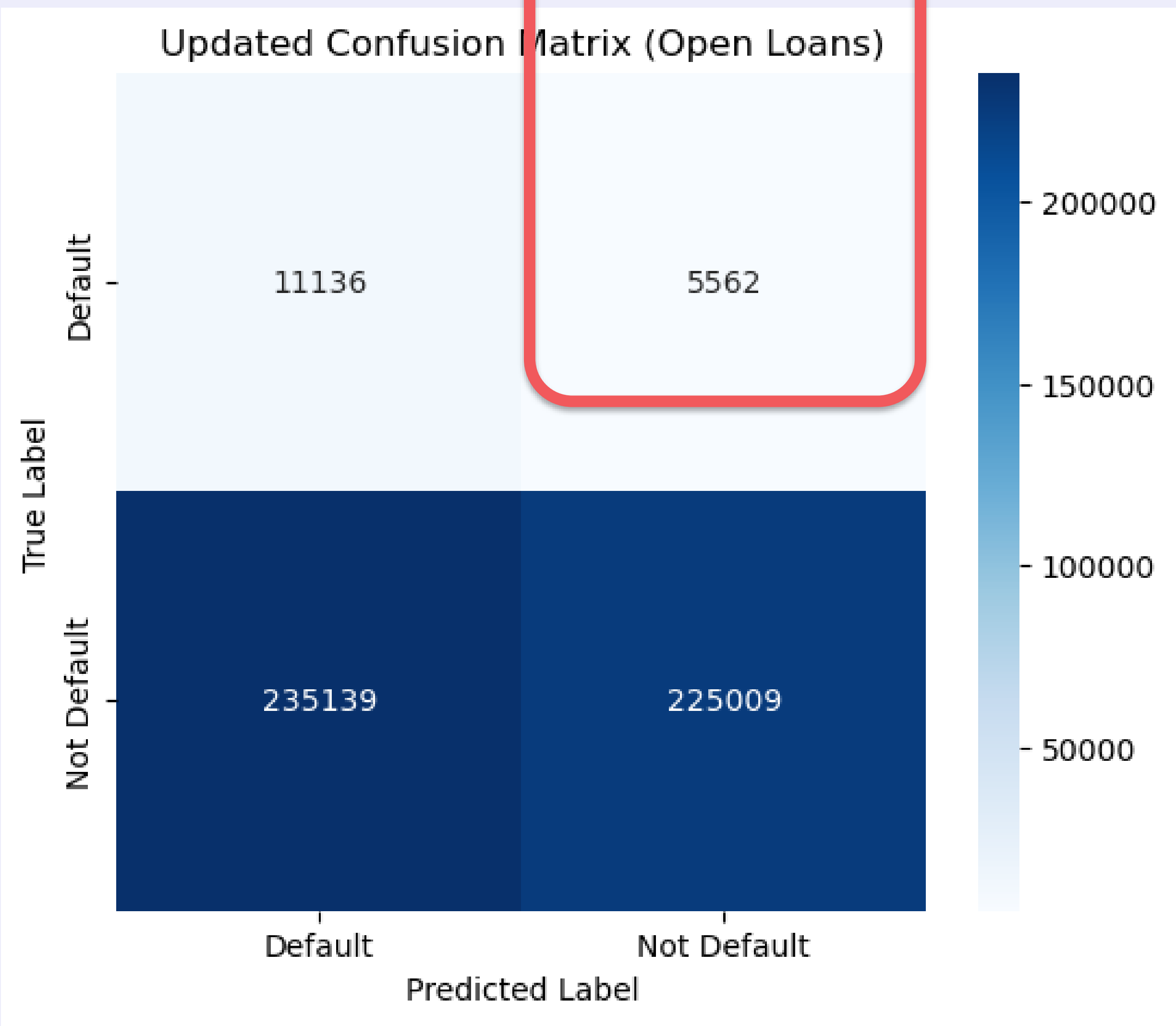

2. Distribution of Prob_Default based on Loan Grade (on the right):

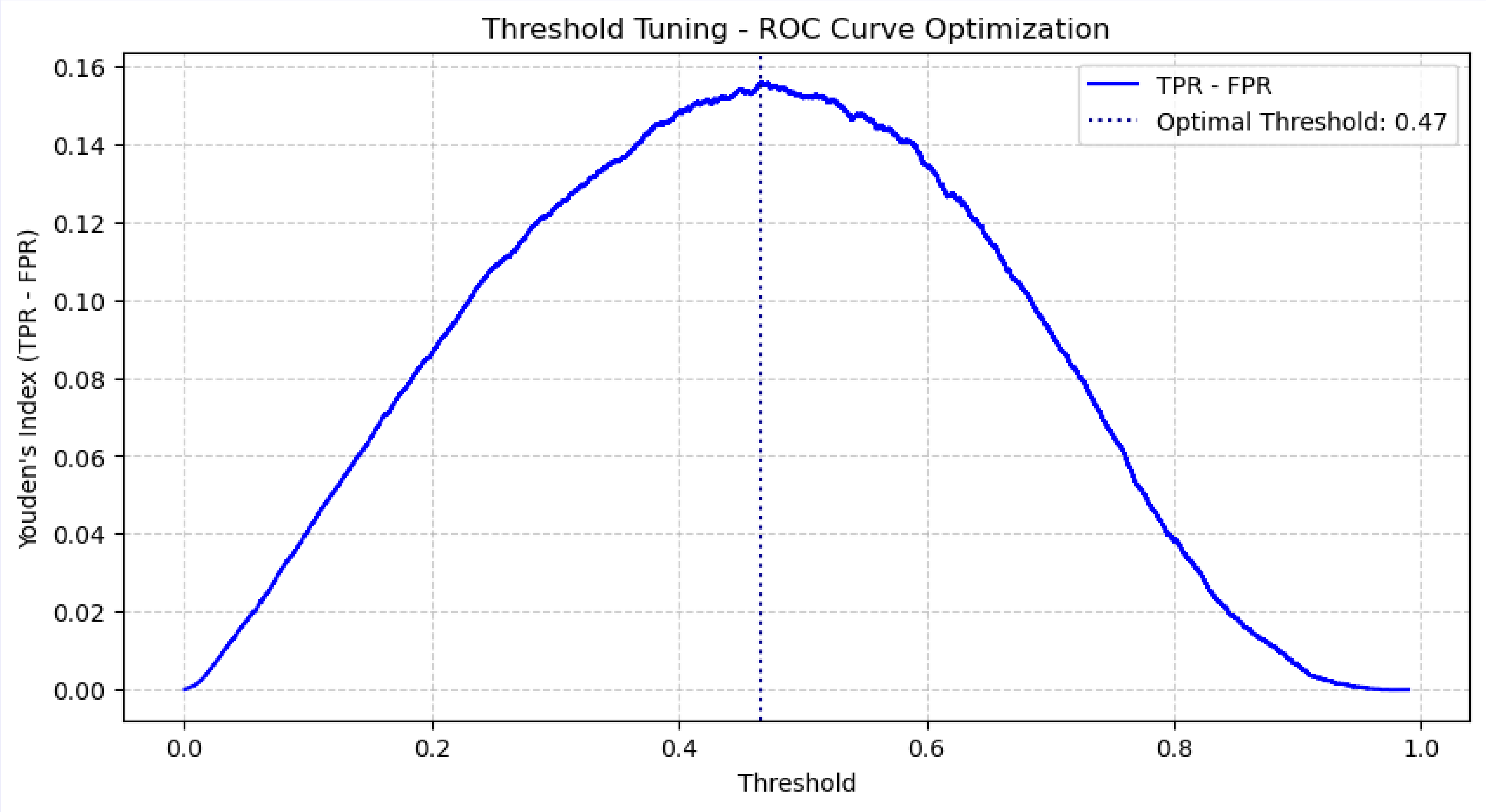
```
loan_default
0    0.964982
1    0.035018
Name: proportion, dtype: float64
```

```
pred_default
0    0.53412
1    0.46588
Name: proportion, dtype: float64
```

# 4.2: Recalibrating Threshold and Estimating Impact




1. First Confusion Matrix Using Best Model (XGboost) and Threshold (t==0.5) from Closed Loans:
   a) **Number of FN: 6501**
   b) Predicted defaults (Class 1): 0.464
   c) However, in Open Loans, number of loan defaulters (Class 1) is much less: 0.035

2. Second Confusion Matrix (After Calibrating threshold)
   a) Calculated Optimal Threshold for Open loans == 0.47
   b) **Number of FN: 5562**
   c) Estimated RAR: $2.839 billion

| | Missed Revenue (FP) | Default Loss (FN) | Expected Interest Earned (TN) | Risk-Adjusted Return (RAR) |
|---|---|---|---|---|
| 0 | 2.926920e+09 | 87449300.0 | 1.274154e+09 | 2.839471e+09 |

esade

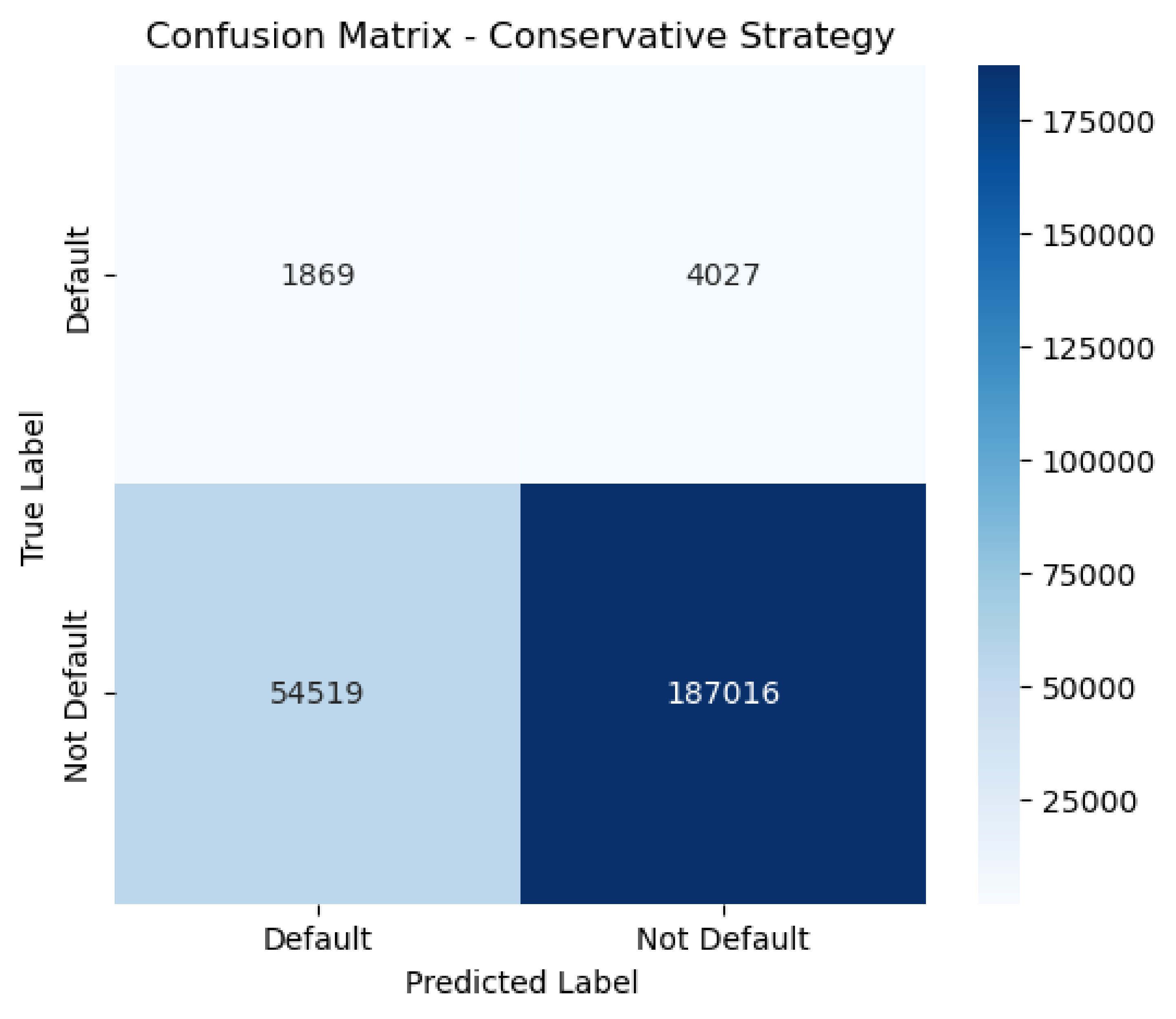
# 4.3: Business Impact for the 3 Strategies

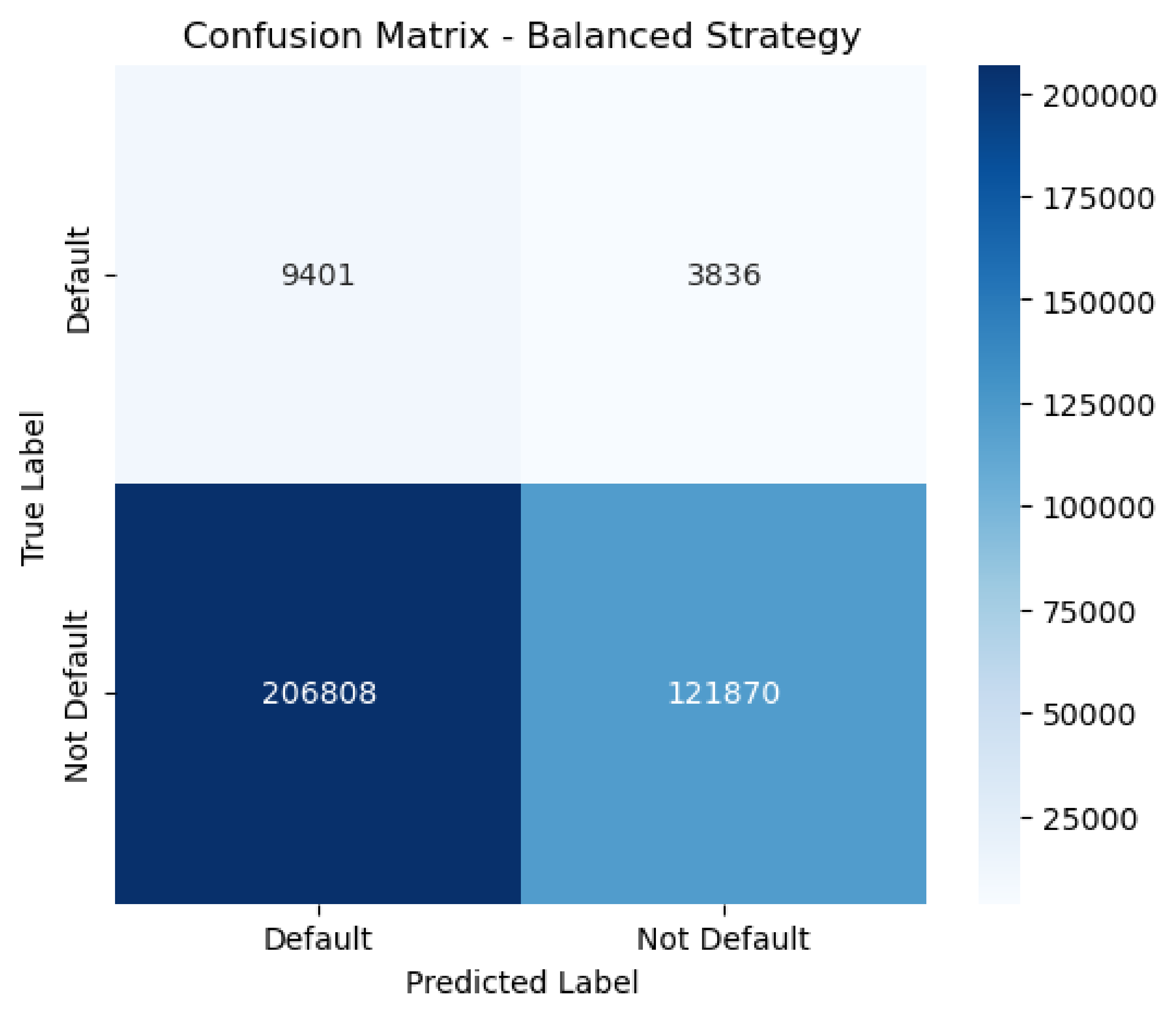
Goal: To get the best thresholds and respective returns for each strategy

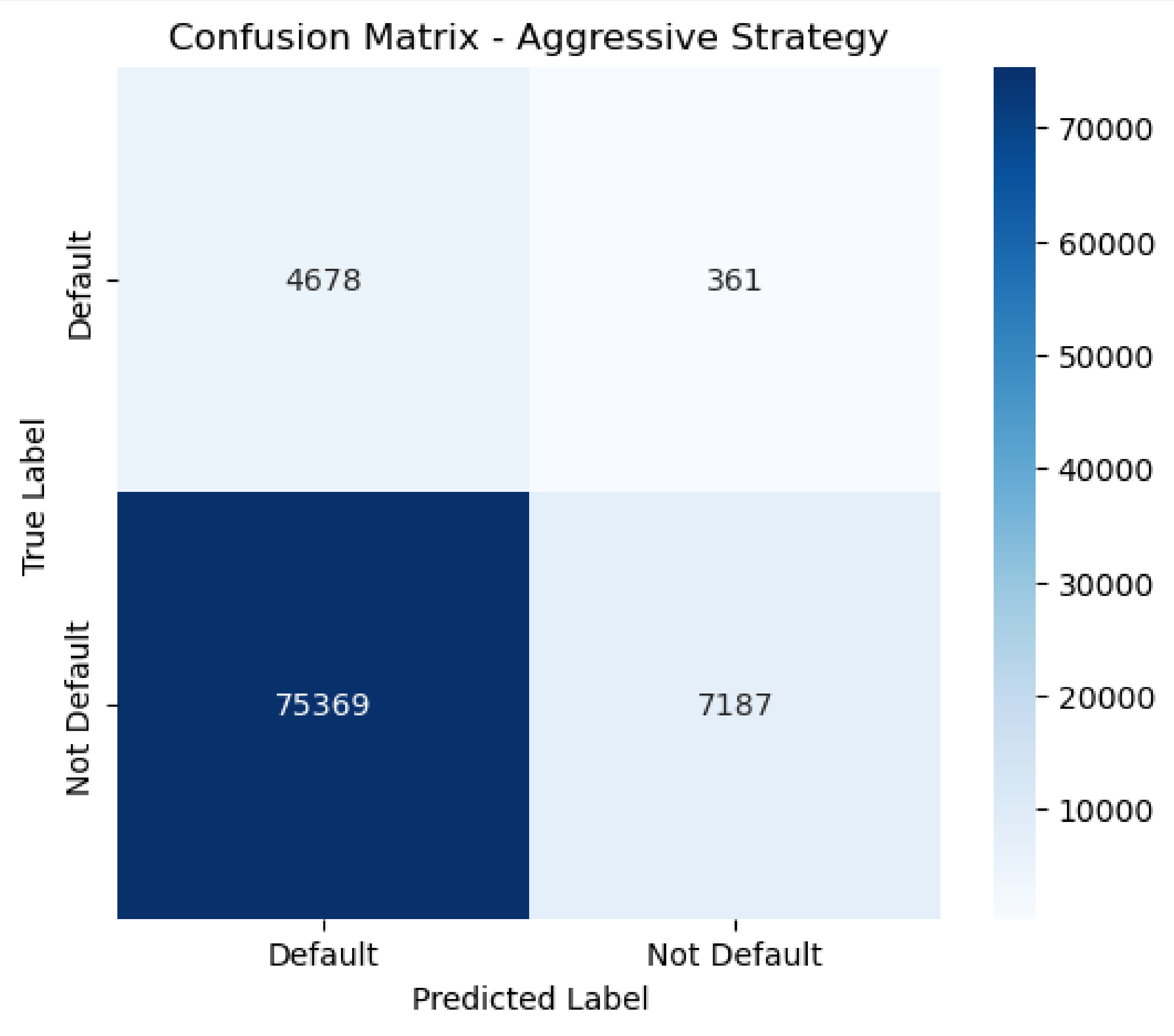
Current state: Model == XGboost; Threshold == 0.466

**Data Manipulation Strategy:**

1. Define the strategy dictionary
2. Get 3 different confusion matrices
3. Threshold Tune for Optimal threshold and Visualization
4. Compute 3 different business impact and compare



esade



# 4.3: Business Impact for the 3 Strategies

- Objective: Threshold tuning is performed to obtain maximum RAR
- For each strategy, the maxima point of TP-FP is determined and then finding the corresponding x-value.
- Mapping of Confusion Matrix Values:

Columns Extracted to Map Confusion Matrix Values

**1. TP: Not considered** (as we do not want to invest in correctly predicted bad loans, if they are still open)

**2. FP: Missed Interest**

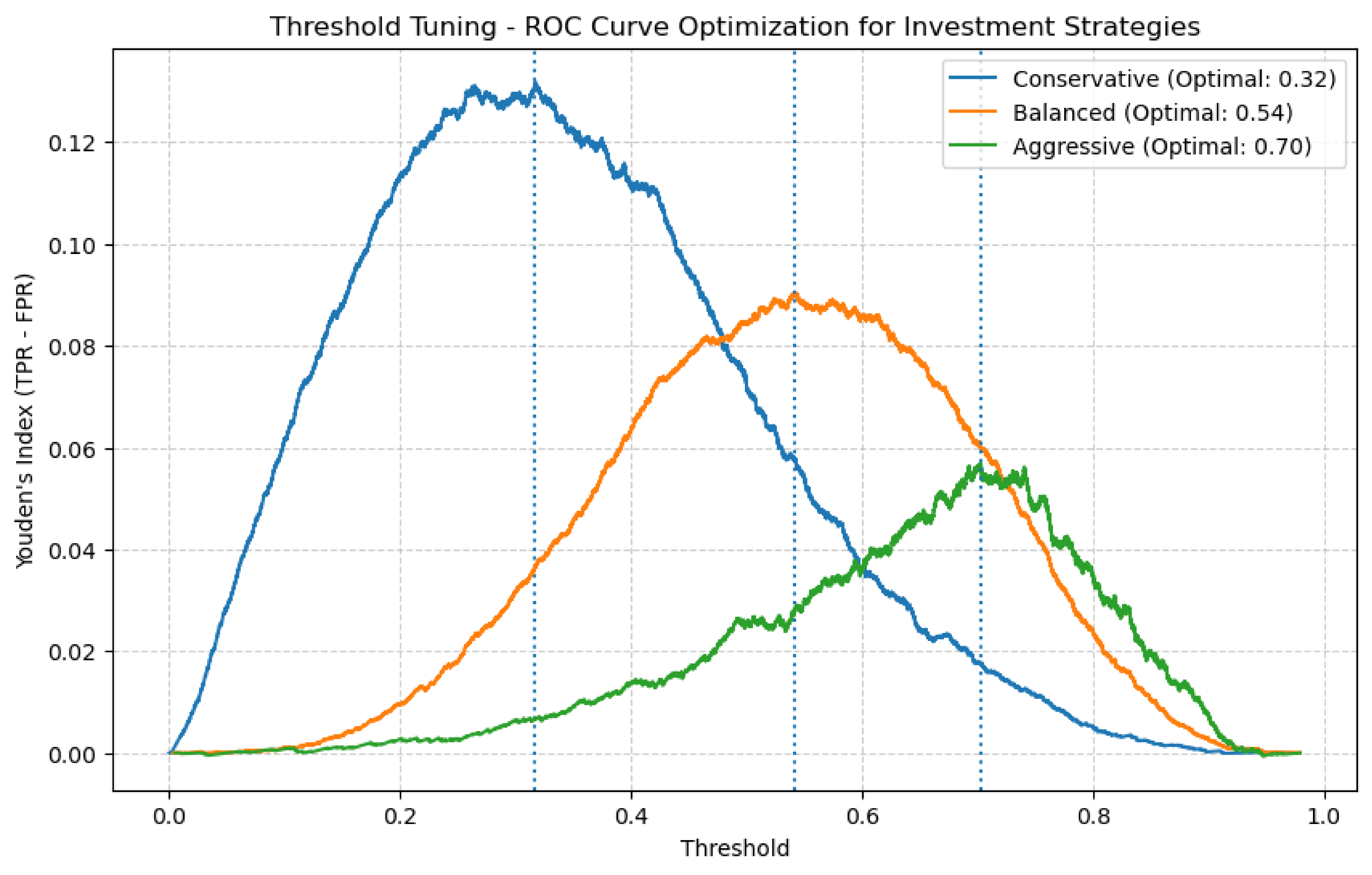
- ('Loan Amount' x 'Interest Rate' x 'Term') x #FP

**3. FN: Principal Loss**

- 'Loan Amount' x #FN

**4. TN: Interest from Actual Good Loans**

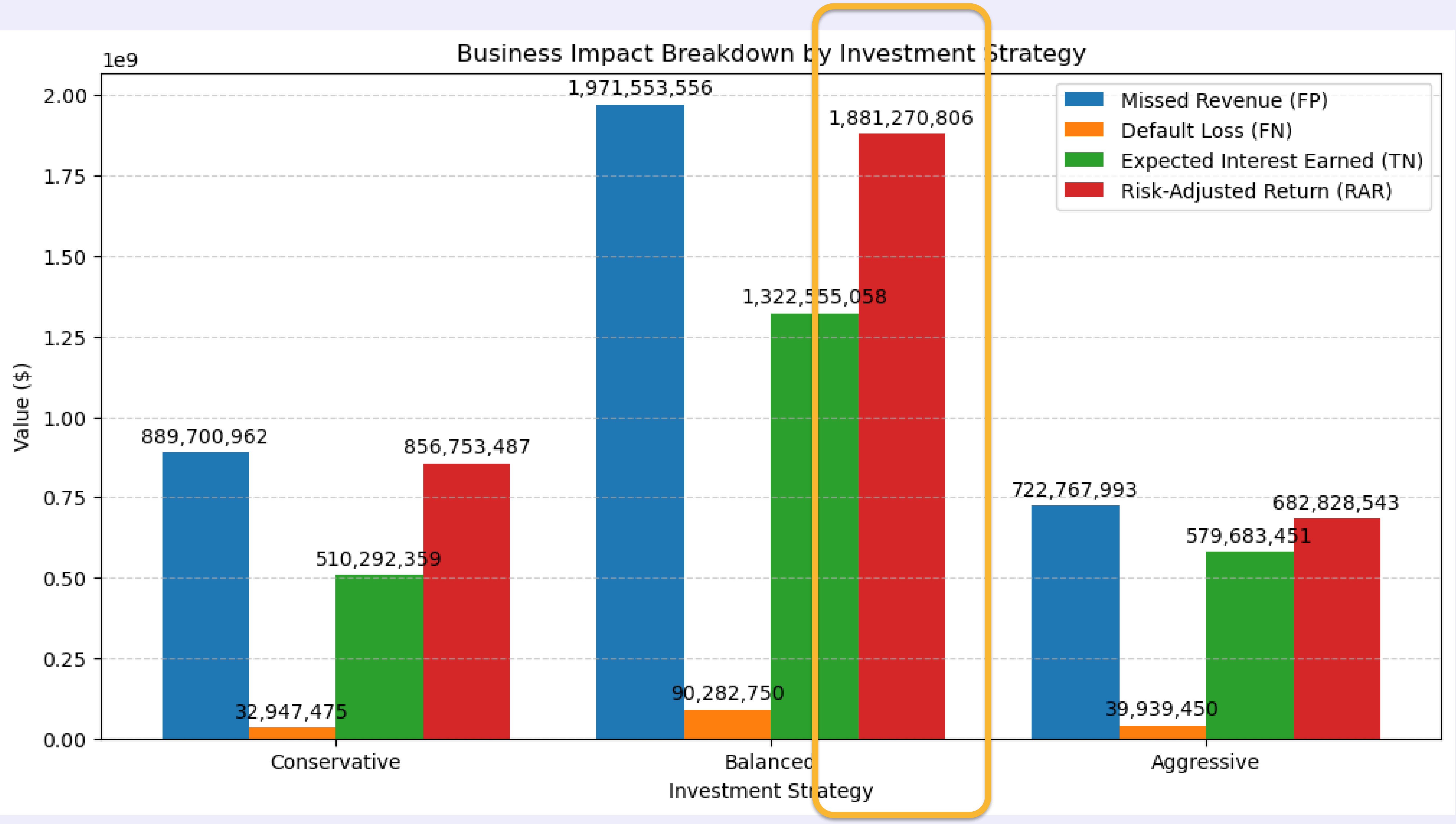
- ('Loan Amount' x 'Interest Rate' x 'Term') x #TN



esade



# 4.4: Overall Conclusion




| | Missed Revenue (FP) | Default Loss (FN) | Expected Interest Earned (TN) | Risk-Adjusted Return (RAR) |
|---|---|---|---|---|
| Conservative | 8.897010e+08 | 32947475.0 | 5.102924e+08 | 8.567535e+08 |
| Balanced | 1.971554e+09 | 90282750.0 | 1.322555e+09 | 1.881271e+09 |
| Aggressive | 7.227680e+08 | 39939450.0 | 5.796835e+08 | 6.828285e+08 |

**By applying our model from closed loans, we estimated business impact metrics, revealing:**

1. Missed revenue of $2.92B due to rejected profitable loans
2. Default loss of $87.4M
3. Expected interest earned of $1.27B
4. This leads to risk-adjusted return (RAR) of $2.83B.

**Investment Strategies and Risk-Return Insights**
**After segmenting loans into Conservative, Balanced, and Aggressive strategies, we found:**

1. Conservative (Grades A & B): Had the lowest default loss ($28.1M) but limited RAR ($0.93B).
2. Balanced (Grades B, C, & D): Optimized both risk and return, achieving the highest RAR of $1.67B.
3. Aggressive (Grades D, E, & F): Maximized interest earned but suffered a high default loss ($39.8M), reducing RAR to $0.69B.

**Key Takeaways:**

1. Threshold Optimization: Tuning the decision threshold to 0.47 maximized risk-adjusted return, balancing missed revenue and defaults.
2. Optimal Strategy: Balanced (if only 1 choice) With the highest RAR, it offers the best risk-return tradeoff.
3. Dynamic Adjustments: Future refinements in threshold tuning can further optimize profitability based on economic conditions.

esade

Do Good. Do Better.