Identifying At-Risk Customers: A Predictive Analysis of Credit Card Attrition

1. Introduction

A. Overview

Customer attrition is still a major problem for companies in a variety of sectors, and its effects must be lessened with efficient tactics. Churn, also known as customer attrition, is the reduction in the number of customers who stop using a business's goods or services. This problem is widespread in sectors like financial services, e-commerce, and telecommunications that depend on subscription-based business models, where steady income streams are critical to the long-term viability of enterprises. Since keeping current customers is much less expensive than acquiring new ones, client retention is an essential part of successful business plans. As a result, anticipating and resolving churn has emerged as a crucial area of emphasis for customer relationship management and company performance.

The banking sector is no exception to these challenges. The growth of digital-only banks and technological breakthroughs have caused a dramatic change in the sector during the past 20 years. Customers today anticipate more customization, smooth digital integration, and measurable value from the services they use, which has significantly changed their expectations due to these changes. Customer churn is especially problematic in the banking industry because of its extensive effects. High churn rates affect the bank's capacity to establish and preserve enduring connections with clients in addition to causing immediate revenue losses.

One financial product where client attrition might have serious repercussions is credit card services. Credit card customers are often highly profitable for banks, contributing to revenue through transaction fees, interest payments, and auxiliary services. In addition to an immediate drop in revenue, banks often see a decline in customer lifetime value (CLV), a crucial indicator for long-term financial planning, when these clients depart.

Banks are increasingly using data-driven methods to comprehend and predict client loss in response to these difficulties. To address the unique requirements and preferences of at-risk clients, these actions could take the form of tailored offers, better customer support, or strengthened loyalty programs. These goals may now be accomplished with the help of predictive analytics and machine learning models, which provide banks the ability to use massive datasets to find patterns and trends that guide retention tactics.

This study analyzes the BankChurners dataset to forecast credit card user attrition through the use of machine learning techniques. Numerous client factors, such as demographic information, account history, and transactional activities, are included in the collection. Through the analysis of this data, the study seeks to pinpoint the main causes of customer attrition and create prediction models that can precisely identify consumers who are at danger. Banks will be able to better manage resources and prioritize retention initiatives with the aid of these models.

This study aims to advance the general knowledge of attrition in the financial services industry by investigating the reasons and predictors of customer churn. It is anticipated that the results would emphasize the significance of contextual elements like customer tenure and demographic traits, as well as behavioral indicators like transaction frequency and credit utilization. The study lays the groundwork for focused interventions that tackle the root causes of attrition by identifying key churn drivers. Additionally, this analysis's application of machine learning techniques highlights the need of sophisticated analytics in addressing challenging business issues. The ability to anticipate and stop client attrition will continue to be a crucial factor in determining success as banks navigate an increasingly dynamic and competitive environment.

B. Literature review

In the case of credit card consumers, recent research has improved the accuracy of churn predictions by utilizing machine learning approaches. Al-Najjar et al. (2022) used five machine learning models—the Bayesian Network, C5 Decision Tree, CHAID Tree, CR Tree, and Neural Networks—as well as feature selection approaches to create a prediction model for credit card customer attrition. The study showed that combining multi-categorical factors enhanced model performance and underlined the significance of early prediction for retention measures. These results are consistent with the emphasis on feature selection and sophisticated machine learning for churn prediction in this research.

Similarly, in order to determine when and why customers are likely to stop using their services, Siddiqui et al. (2023) created churn prediction models, concentrating on credit card users. Three models were suggested by the study: Model 1 included all variables, Model 2 distinguished between continuous and categorical features, and Model 3 used feature selection methods to rank the most important aspects. The study highlighted important characteristics of churners and determined the best machine learning techniques for churn prediction. In order to improve feature selection and churn prediction accuracy, the study also compared its models to other methods already in use. By applying several sampling techniques and building a customer churn prediction model with an enhanced XGBoost algorithm, Peng et al. (2023) addressed the problem of data imbalance in churn prediction. Interpretability analysis was also used in their study to offer practical advice to financial institutions looking to reduce customer churn.

All of these experiments demonstrate how well machine learning models predict credit card client attrition. As the main indications of churn, they emphasize the crucial relevance of behavioral measures like transaction frequency and credit utilization. Furthermore, improving the forecast accuracy and interpretability of these models depends critically on the use of feature selection strategies and the management of data imbalances.

Financial institutions can proactively identify clients who are at risk of leaving and put targeted retention plans into place by combining these cutting-edge analytical techniques. This contributes to the institution's long-term success by strengthening client relationships and reducing the possibility of revenue losses.

2. Dataset

A. Description and Exploratory Analysis

Key information about customer demographics, behavior, and account-related traits is revealed by the dataset analysis, which is crucial for comprehending attrition trends. With a mean age of almost 46 years, the age distribution shows that the majority of clients are between the ages of 40 and 60, indicating that the bank predominantly caters to middle-aged people. The gender breakdown indicates that female clients make up a slight majority. The "Graduate" and "High School" categories have the highest concentration of education levels. Also, according to income, the largest group makes less than $40,000 per year, while data on marital status shows that about half of the clients are married.

Characteristics pertaining to behavior and accounts are essential to comprehending churn. Lower transaction volumes and counts increase the likelihood of customer attrition, underscoring the significance of engagement in retention. Customer who regularly use their accounts are less likely to depart, according to behavioral variables like total transaction count and total transaction amount, which have substantial negative associations with attrition. The average credit limit of attrited consumers is typically lower, which may restrict their financial options (Table 1).

Higher attrition rates are also linked to shorter tenures at the bank, as indicated by "Months on Book," as well. Short-term bank customers are less likely to stick around, which emphasizes the value of establishing lasting relationships. Additionally, another important metric linked to attrition is a lower "Total Relationship Count," which indicates the number of accounts a customer has with the bank. This highlights that clients are more likely to depart if they have fewer accounts or are less involved with the bank. Engagement is further shown as a critical predictor of attrition by declines in transactional activity, especially across consecutive quarters.

Furthermore, compared to current customers, attrited clients have higher average utilization percentages, which may be a sign of financial distress or discontent with their credit limitations. Low engagement is a key predictor of attrition, as seen by the clear patterns in the transactional behavior of attrited consumers. These clients are more likely to have low transaction counts, usually less than 20 transactions year, and low transaction quantities, with annual expenditure frequently dipping below $2,000. Such little activity raises the risk of churn by indicating a decreased dependence on or level of satisfaction with the bank's services. Customers that have smaller open credit amounts or high utilization ratios are also more likely to experience attrition. These trends can be a sign of financial strain or discontent with the available credit limits, which would encourage clients to look for other financial institutions.

Additional indications of disengagement among attrited clients can be found in changes in activity levels, especially between successive quarters. A notable decline in transaction counts and amounts from one quarter to the next (e.g., Q4 to Q1) underscores the role of diminishing engagement as a strong signal of potential churn. Predictive modeling is significantly hampered by the dataset's class imbalance. With only 16.1% of customers classified as "Attrited Customers" and 83.9% of customers classified as "Existing Customers," the dataset is significantly biased in favor of the majority class.

Mean	Attrited Customers	Existing Customers
Customer Age	46.66	46.26
Credit Limit	$8,136	$8,726
Transaction Amount	$3,095	$4,654
Transaction Count	44.93	68.67
Utilization Ratio	16.2%	29.6%

Table 1: Comparison of Key Metrics Between Attrited and Existing Customers

B. Preprocessing

The dataset was prepared for analysis in several ways. First, features that were not essential to the prediction task were eliminated, such as the unique customer identity CLIENTNUM. To ensure interoperability with machine learning algorithms, binary features for each category were created by transforming categorical variables—such as gender, marital status, and income category—using one-hot encoding. To standardize ranges and keep features with bigger scales from taking over the models, MinMaxScaler was used to normalize numerical variables like Total Transaction Amount and Average Utilization Ratio. To lessen their impact on model performance, outliers found during EDA were also capped using IQR criteria.

In order to improve the dataset, feature engineering was essential. Total Transaction Amount was divided by Months on Book to create a new variable called Transactions Per Month. Deeper insights into engagement patterns were made possible by this, which offered a normalized measure of client activity. The combination of these preprocessing steps ensured a robust and reliable dataset for modeling.

3. Predictive Task and Evaluation

The objective of the prediction task was to use the available features to categorize consumers as either existing or attrited. To give a thorough evaluation of model performance, suitable evaluation criteria were chosen in light of the dataset's notable class imbalance. To make sure that retention efforts are concentrated on truly at-risk consumers, precision was utilized to quantify the percentage of accurately identified attrited customers among all anticipated attrited customers. The model's capacity to identify at-risk instances was demonstrated by using recall to assess the percentage of correctly identified attrited consumers among all real attrited clients.

A balanced statistic to assess the model's overall performance was provided by the F1-score, which is a harmonic mean of precision and recall. Furthermore, the confusion matrix offers a thorough examination of true positives, false positives, true negatives, and false negatives, enabling error analysis and enhancing the model's dependability.

4. Model Development and Comparison

The development and comparison of models began with baseline approaches and advanced to more sophisticated techniques, including ensemble methods and feature selection, to improve performance. Each model's strengths and limitations were carefully assessed to achieve optimal predictions.

A. Baseline Models

Logistic Regression was used as the baseline model due to its simplicity and interpretability. This model provided a good starting point for comparison, as it is computationally efficient and easy to implement. However, its linear nature limited its ability to capture complex, non-linear relationships in the data. Logistic Regression achieved a precision of 0.78, recall of 0.48 for predicting attrition, and an F1-score of 0.69, demonstrating modest performance but highlighting the need for more advanced models.

K-Nearest Neighbors (KNN) offered slightly improved results compared to Logistic Regression in terms of precision of 0.75, but recall dropped to 0.4, and F1-score to 0.52, still leaving room for better solutions.

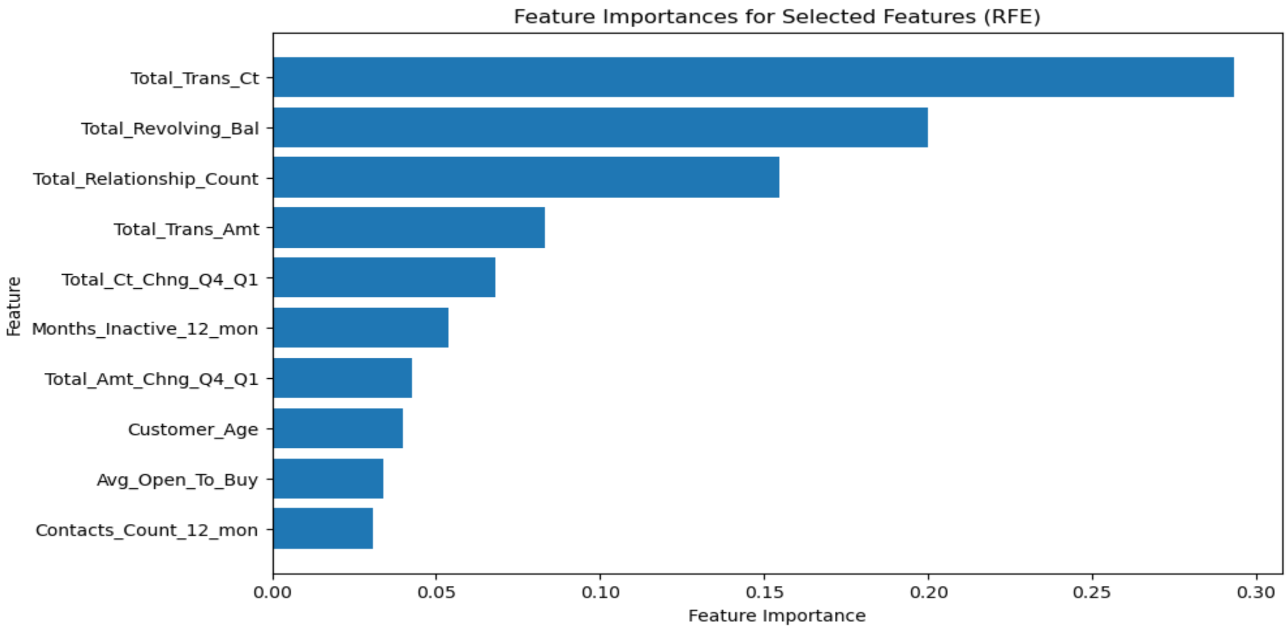


Fig 1.1 : Feature Importances for Selected Features Identified by Recursive Feature Elimination (RFE)

B. Advanced Models

Advanced machine learning models addressed the shortcomings of the baseline methods, and the performance significantly improved. With a precision of 0.80, recall of 0.81, and F1-score of 0.80, a Decision Tree model outperformed the baseline models and provided improved interpretability.

Random Forest (with SMOTE) tackled the class imbalance in the dataset by balancing the minority class using SMOTE before training. Random Forest achieved a precision of 0.87, recall of 0.90, and an F1-score of 0.89, demonstrating overall predictive power. Gradient Boosting further improved performance by reducing overfitting and capturing non-linear relationships more effectively than Random Forest. By focusing on errors from previous iterations, Gradient Boosting refined predictions achieving a precision of 0.93, a recall of 0.85, and an F1-score of 0.89 for the minority class, showcasing its ability to capture complex patterns in the data. LightGBM (LGBM), optimized for speed and scalability, offered an efficient alternative to Gradient Boosting. LGBM achieved a precision of 0.91, recall of 0.91, and an F1-score of 0.91, combining computational efficiency with strong predictive performance.

XGBoost emerged as the best-performing standalone model. We incorporated regularization techniques to prevent overfitting and underwent hyperparameter tuning through grid search to optimize its performance. XGBoost achieved a precision of 0.92, a recall of 0.90, and an F1-score of 0.91, excelling across all evaluation metrics due to its robust handling of imbalanced data and complex relationships.

To streamline the modeling process and improve interpretability, Recursive Feature Elimination (RFE) was applied to XGBoost. This feature selection technique identified the most impactful predictors, with Total Transaction Count, Total Revolving Balance and Total Relationship Count emerging as the top features. By focusing on these critical variables, the model retained its strong predictive capabilities while reducing complexity. XGBoost with RFE achieved a precision of 0.92, recall of 0.91, and an F1-score of 0.91, balancing efficiency with performance.

The most important characteristics affecting customer attrition are highlighted by key findings from the investigation. Higher transaction counts were linked to a lower likelihood of turnover, making Total_Trans_Ct a powerful predictor of active involvement. While Avg_Utilization_Ratio demonstrated prudent credit usage and had a favorable correlation with retention, Total_Relationship_Count measured customer loyalty and retention. The importance of behavioral measures in churn prediction was highlighted by the fact that they routinely surpassed demographic indicators in predictive power, such as transaction and relationship count. These results were

Model	Precision (False/True)	Recall (False/True)	F1 Score (False/True)	Comments
Logistic Regression	0.78 / 0.93	0.62 / 0.97	0.69 / 0.95	Baseline model; struggled with recall for the False class.
KNN	0.73 / 0.89	0.40 / 0.97	0.52 / 0.93	Poor recall for the False class; not suitable for imbalanced datasets.
Decision Tree	0.80 / 0.96	0.81 / 0.96	0.80 / 0.96	Better balance than Logistic Regression and KNN.
Random Forest (SMOTE)	0.87 / 0.98	0.90 / 0.97	0.89 / 0.98	Handled class imbalance well using SMOTE.
Gradient Boost	0.93 / 0.97	0.85 / 0.99	0.89 / 0.98	Competitive performance but slightly below LightGBM and XGBoost models.
Light GBM	0.91 / 0.98	0.91 / 0.98	0.91 / 0.98	Excellent balance across metrics; tied with XGBoost in performance.
XG Boost	0.92 / 0.98	0.90 / 0.99	0.91 / 0.98	Robust performance and balanced metrics; highly interpretable.
XG Boost (RFE)	0.92 / 0.98	0.91 / 0.98	0.91 / 0.98	Best model; combines feature interpretability and top-tier performance.

Table 2 : Model Comparison Table: Precision, Recall, F1 Score, and Performance Analysis

confirmed by the use of Recursive Feature Elimination (RFE), which recognized these features as crucial and made it possible to create a simplified model with fewer variables while retaining high prediction accuracy.

5. Results and Conclusion

The best model for forecasting customer attrition, according to the data, is XGBoost with Recursive Feature Elimination (RFE). For the minority class (attrited consumers), the model's precision and recall were 0.92 and 0.91, respectively, indicating that it can effectively identify at-risk clients while reducing false positives. The model's dependability is further highlighted by the F1-score of 0.91, which shows a good balance between precision and recall. Furthermore, with precision, recall, and F1-scores all at 0.98, the model demonstrated outstanding performance for the majority class (current customers), guaranteeing no trade-off between the two classes. The model's resilience and consistency in managing unbalanced datasets are highlighted by its overall accuracy of 0.97.

References

1. AL-Najjar D, Al-Rousan N, AL-Najjar H. Machine Learning to Develop Credit Card Customer Churn Prediction. Journal of Theoretical and Applied Electronic Commerce Research. 2022; 17(4):1529-1542. https://doi.org/10.3390/jtaer17040077

2. Siddiqui, N., Haque, M.A., Khan, S.M.S. et al. Different ML-based strategies for customer churn prediction in banking sector. J. of Data, Inf. and Manag. 6, 217–234 (2024). https://doi.org/10.1007/s42488-024-00126-z

3. Peng, H., Zhang, L., & Liu, Q. (2023). Handling data imbalance in customer churn prediction for credit card users: An XGBoost approach. PLOS ONE, 18(8), e0289724. https://doi.org/10.1371/journal.pone.0289724

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'BankChurners.csv'
data = pd.read_csv(file_path)

data_info = data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 23 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   CLIENTNUM
10127 non-null  int64
 1   Attrition_Flag
10127 non-null  object
 2   Customer_Age
10127 non-null  int64
 3   Gender
10127 non-null  object
 4   Dependent_count
10127 non-null  int64
 5   Education_Level
10127 non-null  object
 6   Marital_Status
10127 non-null  object
 7   Income_Category
10127 non-null  object
 8   Card_Category
10127 non-null  object
 9   Months_on_book
10127 non-null  int64
 10  Total_Relationship_Count
10127 non-null  int64
 11  Months_Inactive_12_mon
10127 non-null  int64
 12  Contacts_Count_12_mon
10127 non-null  int64
 13  Credit_Limit
10127 non-null  float64
 14  Total_Revolving_Bal
```

```
10127 non-null  int64
 15  Avg_Open_To_Buy
10127 non-null  float64
 16  Total_Amt_Chng_Q4_Q1
10127 non-null  float64
 17  Total_Trans_Amt
10127 non-null  int64
 18  Total_Trans_Ct
10127 non-null  int64
 19  Total_Ct_Chng_Q4_Q1
10127 non-null  float64
 20  Avg_Utilization_Ratio
10127 non-null  float64
 21
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_
mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1  10127
non-null  float64
 22
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_
mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2  10127
non-null  float64
dtypes: float64(7), int64(10), object(6)
memory usage: 1.8+ MB
```

```
data.drop(['CLIENTNUM',

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12
_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1',

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12
_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2'],
        axis=1,inplace=True)
```

```
missing_values = data.isnull().sum()
missing_values.sum()
```

```
0
```

```
numerical_summary = data.describe()
numerical_summary
```

```
       Customer_Age  Dependent_count  Months_on_book  \
count  10127.000000     10127.000000    10127.000000
mean      46.325960         2.346203       35.928409
std        8.016814         1.298908        7.986416
min       26.000000         0.000000       13.000000
25%       41.000000         1.000000       31.000000
50%       46.000000         2.000000       36.000000
75%       52.000000         3.000000       40.000000
max       73.000000         5.000000       56.000000
```

```
       Total_Relationship_Count  Months_Inactive_12_mon  \
count              10127.000000            10127.000000
mean                   3.812580                2.341167
std                    1.554408                1.010622
min                    1.000000                0.000000
25%                    3.000000                2.000000
50%                    4.000000                2.000000
75%                    5.000000                3.000000
max                    6.000000                6.000000

       Contacts_Count_12_mon  Credit_Limit  Total_Revolving_Bal  \
count           10127.000000  10127.000000         10127.000000
mean                2.455317   8631.953698          1162.814061
std                 1.106225   9088.776650           814.987335
min                 0.000000   1438.300000             0.000000
25%                 2.000000   2555.000000           359.000000
50%                 2.000000   4549.000000          1276.000000
75%                 3.000000  11067.500000          1784.000000
max                 6.000000  34516.000000          2517.000000

       Avg_Open_To_Buy  Total_Amt_Chng_Q4_Q1  Total_Trans_Amt
Total_Trans_Ct  \
count     10127.000000          10127.000000     10127.000000
10127.000000
mean       7469.139637              0.759941      4404.086304
64.858695
std        9090.685324              0.219207      3397.129254
23.472570
min           3.000000              0.000000       510.000000
10.000000
25%        1324.500000              0.631000      2155.500000
45.000000
50%        3474.000000              0.736000      3899.000000
67.000000
75%        9859.000000              0.859000      4741.000000
81.000000
max       34516.000000              3.397000     18484.000000
139.000000

       Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
count         10127.000000           10127.000000
mean              0.712222               0.274894
std               0.238086               0.275691
min               0.000000               0.000000
25%               0.582000               0.023000
50%               0.702000               0.176000
75%               0.818000               0.503000
max               3.714000               0.999000
```

`Numerical Summary`

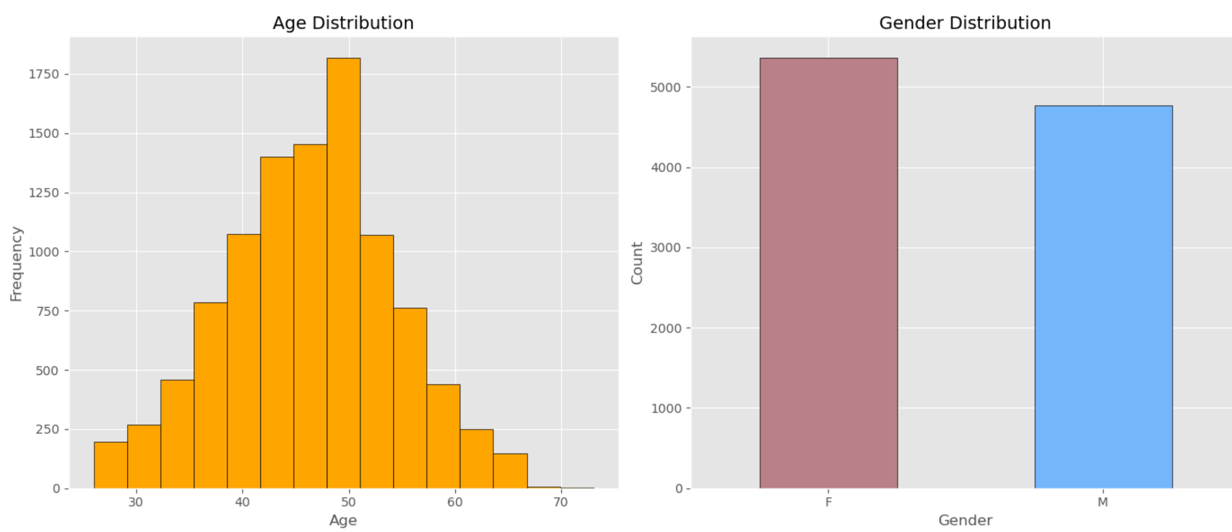
a. Wide Ranges

- Features like Credit_Limit and Total_Trans_Amt have large maximum values relative to the mean, suggesting the presence of outliers.

b. High Variability

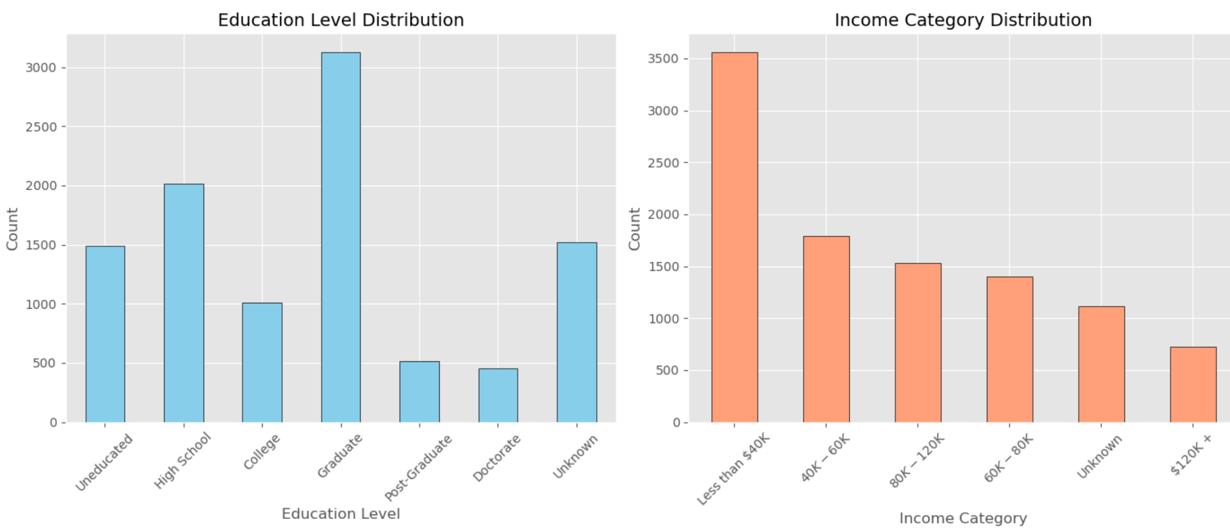
- Columns like Total_Amt_Chng_Q4_Q1 (mean = 0.76, max = 3.39) and Total_Ct_Chng_Q4_Q1 (mean = 0.71, max = 3.71) indicate some extreme customer behaviors.

```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Bar Plot 1: Age Distribution
data['Customer_Age'].hist(bins=15, ax=axes[0], color='orange',
edgecolor='black')
axes[0].set_title('Age Distribution', fontsize=14)
axes[0].set_xlabel('Age', fontsize=12)
axes[0].set_ylabel('Frequency', fontsize=12)

# Bar Plot 2: Gender Distribution
data['Gender'].value_counts().plot(kind='bar', ax=axes[1],
color=['#ba8087', '#74b6fc'], edgecolor='black')
axes[1].set_title('Gender Distribution', fontsize=14)
axes[1].set_xlabel('Gender', fontsize=12)
axes[1].set_ylabel('Count', fontsize=12)
axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation=0)

plt.tight_layout()
plt.show()
```

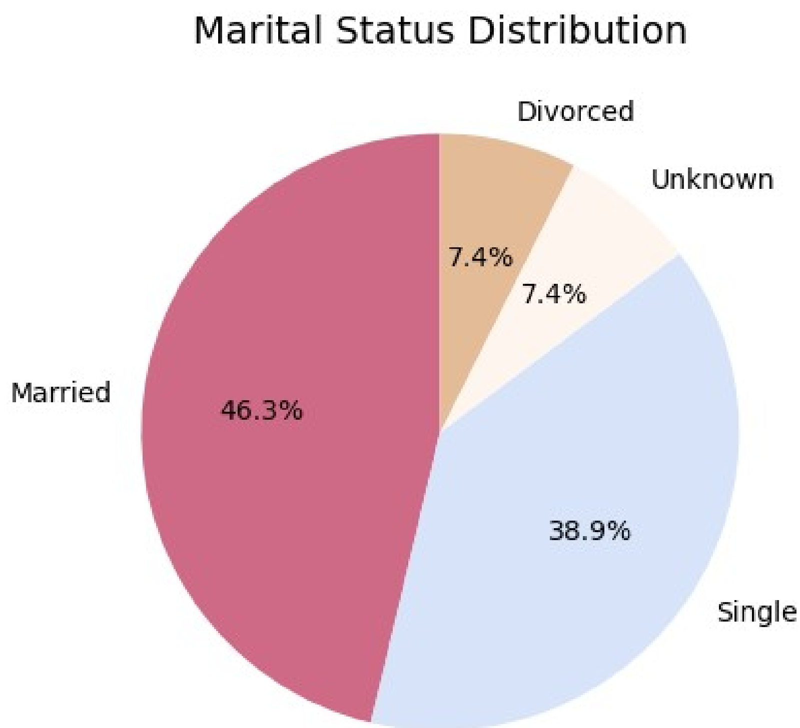



```
education_order = ['Uneducated', 'High School', 'College', 'Graduate',
'Post-Graduate', 'Doctorate', 'Unknown']
```

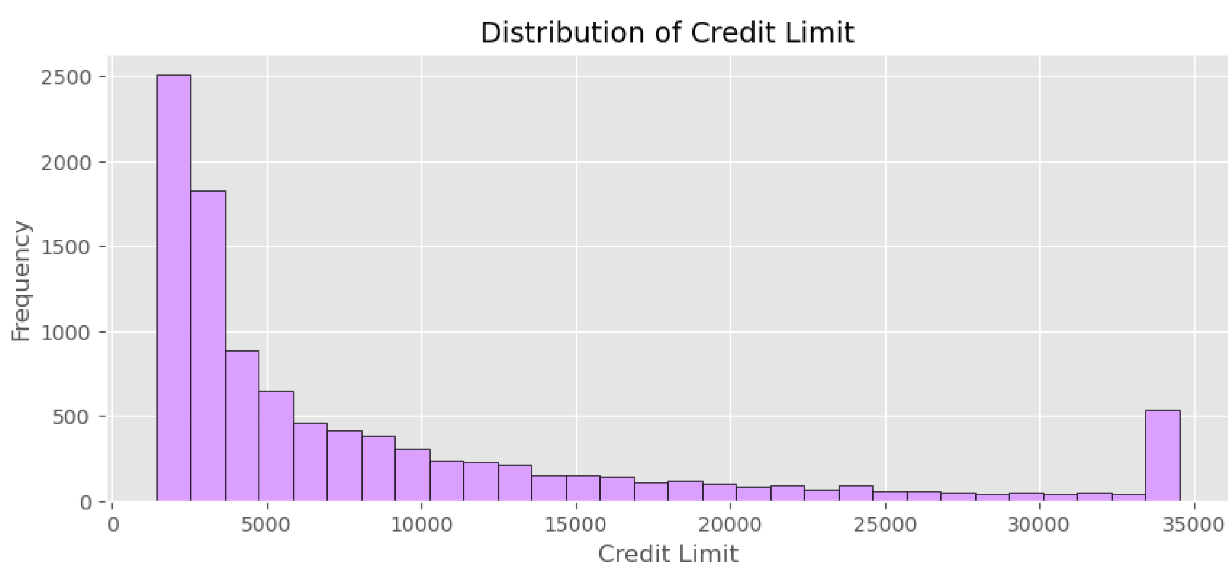
```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Bar Plot 1: Education Level Distribution (Original Order)
data['Education_Level'].value_counts()
[education_order].plot(kind='bar', ax=axes[0], color='#87CEEB',
edgecolor='black')
axes[0].set_title('Education Level Distribution', fontsize=14)
axes[0].set_xlabel('Education Level', fontsize=12)
axes[0].set_ylabel('Count', fontsize=12)
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=45,
fontsize=10)

# Bar Plot 2: Income Category Distribution
data['Income_Category'].value_counts().plot(kind='bar', ax=axes[1],
color='#FFA07A', edgecolor='black')
axes[1].set_title('Income Category Distribution', fontsize=14)
axes[1].set_xlabel('Income Category', fontsize=12)
axes[1].set_ylabel('Count', fontsize=12)
axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation=45,
fontsize=10)

plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(5, 5))
colors = ['#CE6A85','#D6E3F8', '#FEF5EF', '#E4BB97']
data['Marital_Status'].value_counts().plot(kind='pie', autopct='%1.1f%
%', colors=colors, startangle=90)
plt.title('Marital Status Distribution', fontsize=14)
plt.ylabel('')
plt.show()
```



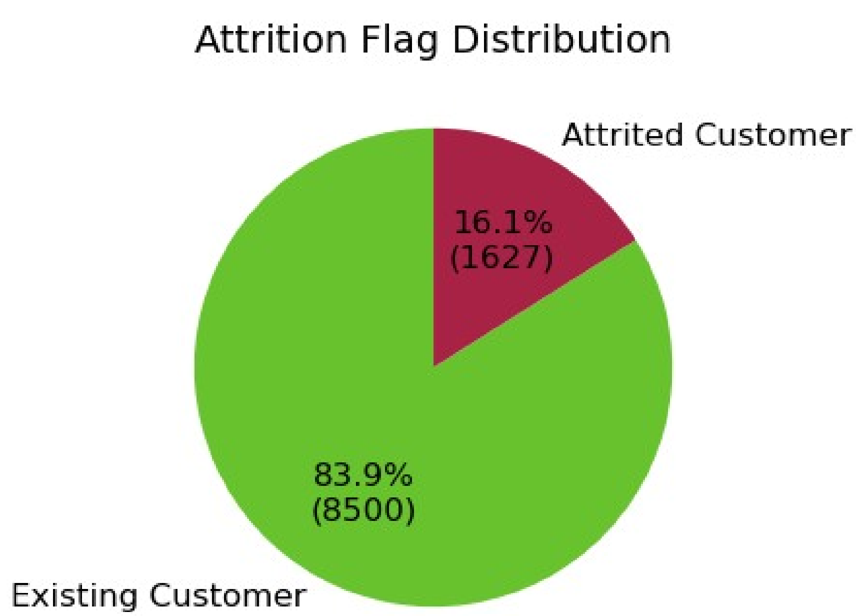

```
plt.figure(figsize=(10, 4))
data['Credit_Limit'].hist(bins=30, color='#db9fff', edgecolor='black')
plt.title('Distribution of Credit Limit', fontsize=14)
plt.xlabel('Credit Limit', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.show()
```



`Summary`

The dataset represents a middle-aged demographic with slightly more females than males, predominantly graduates, and customers in lower-to-middle income brackets. Most customers are married or single, and credit limits tend to be on the lower end.

```
# Pie Chart for Attrition Flag Distribution with numbers displayed
plt.figure(figsize=(5, 4))
attrition_counts = data['Attrition_Flag'].value_counts()
colors = ['#68c22d', '#a82246']

# Plotting the pie chart
plt.pie(attrition_counts, labels=attrition_counts.index,
autopct=lambda p: f'{p:.1f}%\n({int(p * attrition_counts.sum() /
100)})',
        startangle=90, colors=colors, textprops={'fontsize': 12})

plt.title('Attrition Flag Distribution', fontsize=14)
plt.show()
```



1. Proportion of Existing vs. Attrited Customers:
- The chart shows that 83.9% (8,500) of the customers are existing customers, while 16.1% (1,627) are attrited customers.
- This indicates a significant class imbalance in the dataset, with the majority of customers being retained.
1. Focus for Analysis:
- The attrited customers represent a potential loss of revenue, and understanding their reasons for leaving could help mitigate future attrition.

- Efforts should focus on profiling and understanding the behaviors and characteristics of the attrited customers to identify:
- Key factors driving attrition (e.g., low credit limits, high utilization ratios, low transaction counts). Potential strategies for early intervention and retention.

```
numeric_columns = data.select_dtypes(include=['number']).columns

grouped_means = data.groupby('Attrition_Flag')[numeric_columns].mean()
grouped_means = pd.DataFrame(grouped_means)

from IPython.display import display
display(grouped_means.round(3))
```

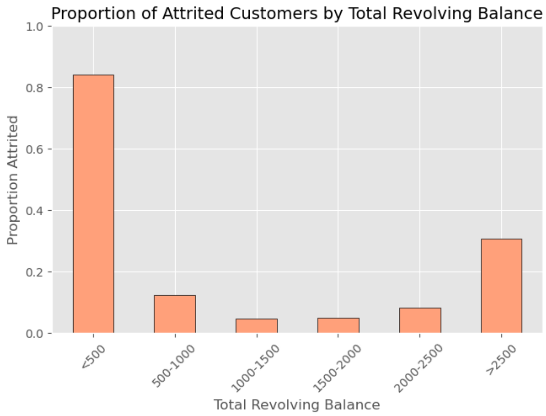
```
                   Customer_Age  Dependent_count  Months_on_book  \
Attrition_Flag
Attrited Customer        46.659            2.403          36.178
Existing Customer        46.262            2.335          35.881

                   Total_Relationship_Count  Months_Inactive_12_mon  \
Attrition_Flag
Attrited Customer                     3.280                   2.693
Existing Customer                     3.915                   2.274

                   Contacts_Count_12_mon  Credit_Limit
Total_Revolving_Bal  \
Attrition_Flag

Attrited Customer                  2.972      8136.039
672.823
Existing Customer                  2.356      8726.878
1256.604

                   Avg_Open_To_Buy  Total_Amt_Chng_Q4_Q1
Total_Trans_Amt  \
Attrition_Flag

Attrited Customer         7463.216                 0.694
3095.026
Existing Customer         7470.273                 0.773
4654.656

                   Total_Trans_Ct  Total_Ct_Chng_Q4_Q1
Avg_Utilization_Ratio
Attrition_Flag

Attrited Customer          44.934                0.554
0.162
Existing Customer          68.673                0.742
0.296
```

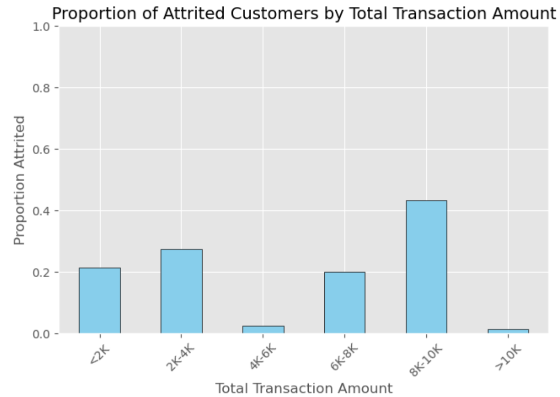
```
def plot_attrition_proportion_for_numeric(column, bins, labels, title, color):
    data[f'{column}_Group'] = pd.cut(data[column], bins=bins, labels=labels)
    attrition = data.groupby([f'{column}_Group', 'Attrition_Flag']).size().unstack(fill_value=0)
    # Calculate proportions
    attrition_proportion = attrition.div(attrition.sum(axis=1), axis=0)

    # Plot the proportion of attrited customers
    attrition_proportion['Attrited Customer'].plot(kind='bar', color=color, edgecolor='black')
    plt.title(f'Proportion of Attrited Customers by {title}', fontsize=14)
    plt.xlabel(title, fontsize=12)
    plt.ylabel('Proportion Attrited', fontsize=12)
    plt.ylim(0, 1)
    plt.xticks(rotation=45, fontsize=10)

# Setting up the plots
fig, axes = plt.subplots(3, 2, figsize=(18, 18))
plt.subplots_adjust(hspace=0.4, wspace=0.4)

# Total_Revolving_Bal
plt.sca(axes[0, 0])
plot_attrition_proportion_for_numeric('Total_Revolving_Bal',
                                      bins=[0, 500, 1000, 1500, 2000, 2500, 3000],
                                      labels=['<500', '500-1000', '1000-1500', '1500-2000', '2000-2500', '>2500'],
                                      title='Total Revolving Balance',

                                      color='#FFA07A')

# Total_Trans_Amt
plt.sca(axes[0, 1])
plot_attrition_proportion_for_numeric('Total_Trans_Amt',
                                      bins=[0, 2000, 4000, 6000, 8000, 10000, 20000],
                                      labels=['<2K', '2K-4K', '4K-6K', '6K-8K', '8K-10K', '>10K'],
                                      title='Total Transaction Amount',
                                      color='#87CEEB')

# Total_Trans_Ct
plt.sca(axes[1, 0])
plot_attrition_proportion_for_numeric('Total_Trans_Ct',
                                      bins=[0, 20, 40, 60, 80, 100,
```

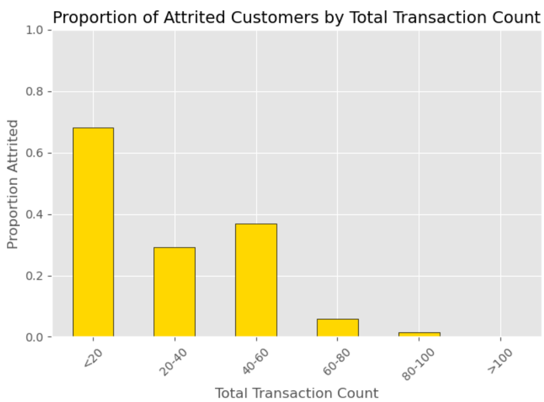
```
140],
                                      labels=['<20', '20-40', '40-60',
'60-80', '80-100', '>100'],
                                      title='Total Transaction Count',

                                      color='#FFD700')

# Total_Ct_Chng_Q4_Q1
plt.sca(axes[1, 1])
plot_attrition_proportion_for_numeric('Total_Ct_Chng_Q4_Q1',
                                      bins=[0, 0.5, 1.0, 1.5, 2.0,
2.5, 3.5],
                                      labels=['<0.5', '0.5-1.0', '1.0-
1.5', '1.5-2.0', '2.0-2.5', '>2.5'],
                                      title='Total Count Change Q4-
Q1',
                                      color='#32CD32')

# Avg_Utilization_Ratio
plt.sca(axes[2, 0])
plot_attrition_proportion_for_numeric('Avg_Utilization_Ratio',
                                      bins=[0, 0.2, 0.4, 0.6, 0.8,
1.0],
                                      labels=['0-0.2', '0.2-0.4',
'0.4-0.6', '0.6-0.8', '0.8-1.0'],
                                      title='Average Utilization
Ratio',
                                      color='#FF6347')

# Hiding the last empty subplot
axes[2, 1].axis('off')

plt.show()
```

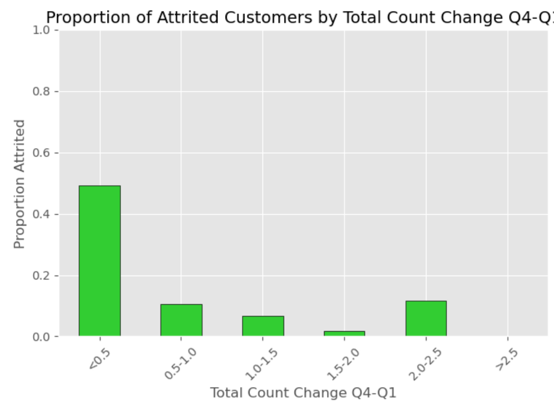
```
/tmp/ipykernel_4745/4239478096.py:7: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  attrition = data.groupby([f'{column}_Group',
'Attrition_Flag']).size().unstack(fill_value=0)
/tmp/ipykernel_4745/4239478096.py:7: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  attrition = data.groupby([f'{column}_Group',
'Attrition_Flag']).size().unstack(fill_value=0)
/tmp/ipykernel_4745/4239478096.py:7: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

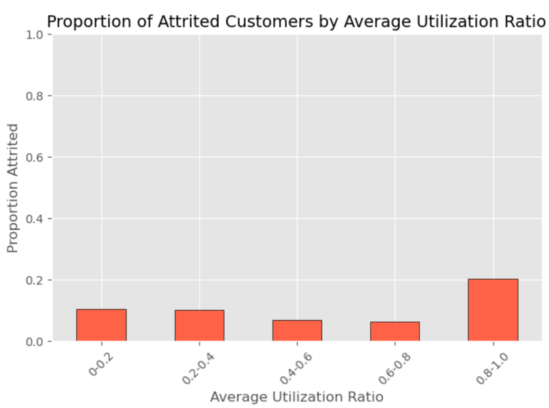
```
  attrition = data.groupby([f'{column}_Group',
'Attrition_Flag']).size().unstack(fill_value=0)
/tmp/ipykernel_4745/4239478096.py:7: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  attrition = data.groupby([f'{column}_Group',
'Attrition_Flag']).size().unstack(fill_value=0)
/tmp/ipykernel_4745/4239478096.py:7: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  attrition = data.groupby([f'{column}_Group',
'Attrition_Flag']).size().unstack(fill_value=0)
```











`Proportion Summary`

- Low Engagement Indicators: Low transaction amounts, fewer transactions, and declines in transaction activity strongly correlate with attrition.
- High Utilization Risk: Customers with high utilization ratios are at risk of attrition, potentially due to financial strain.
- Retention Focus: Customers with low revolving balances or low transaction activity should be flagged for retention strategies.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

file_path = '/home/jovyan/Web_Recommender/BankChurners.csv'
data = pd.read_csv(file_path)

columns_to_drop = [
    "CLIENTNUM",

"Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12
_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1",

"Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12
_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2"
]
data_cleaned = data.drop(columns=columns_to_drop, errors='ignore')

data_cleaned.columns = data_cleaned.columns.str.strip()

categorical_columns = [
    "Attrition_Flag", "Gender", "Education_Level",
    "Marital_Status", "Income_Category", "Card_Category"
]
data_encoded = pd.get_dummies(data_cleaned,
columns=categorical_columns, drop_first=True)

numerical_columns = [
    "Customer_Age", "Dependent_count", "Months_on_book",
"Total_Relationship_Count",
    "Months_Inactive_12_mon", "Contacts_Count_12_mon", "Credit_Limit",

    "Total_Revolving_Bal", "Avg_Open_To_Buy", "Total_Amt_Chng_Q4_Q1",
    "Total_Trans_Amt", "Total_Trans_Ct", "Total_Ct_Chng_Q4_Q1",
"Avg_Utilization_Ratio"
]
scaler = MinMaxScaler()
data_encoded[numerical_columns] =
scaler.fit_transform(data_encoded[numerical_columns])

for col in numerical_columns:
    Q1 = data_encoded[col].quantile(0.25)
    Q3 = data_encoded[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data_encoded[col] = data_encoded[col].clip(lower=lower_bound,
upper=upper_bound)

data_encoded["Transactions_Per_Month"] = (
    data_encoded["Total_Trans_Amt"] / (data_encoded["Months_on_book"]
```

```
+ 1)
)

output_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'
data_encoded.to_csv(output_path, index=False)

output_path
```

```
'/home/jovyan/Web_Recommender/Cleaned_dataset.csv'
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.preprocessing import StandardScaler

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Change this to
your target column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

logistic_model = LogisticRegression(random_state=42, max_iter=1000)

logistic_model.fit(X_train_scaled, y_train)

y_pred = logistic_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

coefficients = pd.DataFrame({
    'Feature': features.columns,
    'Coefficient': logistic_model.coef_[0]
```

```
}).sort_values(by='Coefficient', ascending=False)

print("\nLogistic Regression Coefficients:")
print(coefficients)

Classification Report:
              precision    recall  f1-score   support

       False       0.78      0.62      0.69       325
        True       0.93      0.97      0.95      1701

    accuracy                           0.91      2026
   macro avg       0.85      0.79      0.82      2026
weighted avg       0.91      0.91      0.91      2026

Accuracy Score: 0.9106614017769002
Confusion Matrix:
[[ 201  124]
 [  57 1644]]

Logistic Regression Coefficients:
                           Feature  Coefficient
11                  Total_Trans_Ct     3.512005
12             Total_Ct_Chng_Q4_Q1     0.730180
7              Total_Revolving_Bal     0.708653
3         Total_Relationship_Count     0.601870
14                        Gender_M     0.554394
27  Income_Category_Less than $40K     0.502909
24     Income_Category_$40K - $60K     0.427671
28         Income_Category_Unknown     0.351166
21          Marital_Status_Married     0.291811
25     Income_Category_$60K - $80K     0.223433
8                  Avg_Open_To_Buy     0.183869
26    Income_Category_$80K - $120K     0.182795
9             Total_Amt_Chng_Q4_Q1     0.141366
0                     Customer_Age     0.095678
13           Avg_Utilization_Ratio     0.086499
23          Marital_Status_Unknown     0.046666
6                     Credit_Limit     0.035938
22           Marital_Status_Single     0.019190
16        Education_Level_Graduate    -0.007410
17     Education_Level_High School    -0.032070
30          Card_Category_Platinum    -0.034264
29              Card_Category_Gold    -0.034658
2                   Months_on_book    -0.037354
19      Education_Level_Uneducated    -0.047129
20         Education_Level_Unknown    -0.053187
18   Education_Level_Post-Graduate    -0.061149
31            Card_Category_Silver    -0.076589
15       Education_Level_Doctorate    -0.093723
```

```
1                  Dependent_count    -0.155334
32          Transactions_Per_Month    -0.398914
5            Contacts_Count_12_mon    -0.580440
4           Months_Inactive_12_mon    -0.599302
10                 Total_Trans_Amt    -1.790278
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.preprocessing import StandardScaler

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Change this to
your target column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn_model = KNeighborsClassifier(n_neighbors=5)  # You can experiment
with the number of neighbors (n_neighbors)

knn_model.fit(X_train_scaled, y_train)

y_pred = knn_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.73      0.40      0.52       325
        True       0.89      0.97      0.93      1701
```

```
    accuracy                           0.88      2026
   macro avg       0.81      0.69      0.73      2026
weighted avg       0.87      0.88      0.87      2026

Accuracy Score: 0.8800592300098716
Confusion Matrix:
[[ 131  194]
 [  49 1652]]
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.tree import export_text

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Corrected target
column name
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

dt_model = DecisionTreeClassifier(random_state=42, max_depth=5)  #
Adjust max_depth for complexity

dt_model.fit(X_train, y_train)

y_pred = dt_model.predict(X_test)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

tree_rules = export_text(dt_model,
feature_names=list(features.columns))
print("\nDecision Tree Rules:")
print(tree_rules)
```

```
Classification Report:
              precision    recall  f1-score   support
```

```
       False       0.80      0.81      0.80       325
        True       0.96      0.96      0.96      1701

    accuracy                           0.94      2026
   macro avg       0.88      0.89      0.88      2026
weighted avg       0.94      0.94      0.94      2026

Accuracy Score: 0.9363277393879565
Confusion Matrix:
[[ 264   61]
 [  68 1633]]

Decision Tree Rules:
|--- Total_Trans_Ct <= 0.34
|   |--- Total_Revolving_Bal <= 0.24
|   |   |--- Total_Ct_Chng_Q4_Q1 <= 0.18
|   |   |   |--- Months_Inactive_12_mon <= 0.25
|   |   |   |   |--- Total_Trans_Amt <= 0.09
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Trans_Amt >  0.09
|   |   |   |   |   |--- class: False
|   |   |   |--- Months_Inactive_12_mon >  0.25
|   |   |   |   |--- Total_Trans_Amt <= 0.07
|   |   |   |   |   |--- class: False
|   |   |   |   |--- Total_Trans_Amt >  0.07
|   |   |   |   |   |--- class: False
|   |   |--- Total_Ct_Chng_Q4_Q1 >  0.18
|   |   |   |--- Total_Relationship_Count <= 0.30
|   |   |   |   |--- Card_Category_Gold <= 0.50
|   |   |   |   |   |--- class: False
|   |   |   |   |--- Card_Category_Gold >  0.50
|   |   |   |   |   |--- class: True
|   |   |   |--- Total_Relationship_Count >  0.30
|   |   |   |   |--- Total_Trans_Amt <= 0.08
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Trans_Amt >  0.08
|   |   |   |   |   |--- class: False
|   |--- Total_Revolving_Bal >  0.24
|   |   |--- Total_Relationship_Count <= 0.30
|   |   |   |--- Total_Ct_Chng_Q4_Q1 <= 0.26
|   |   |   |   |--- Contacts_Count_12_mon <= 0.13
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Contacts_Count_12_mon >  0.13
|   |   |   |   |   |--- class: False
|   |   |   |--- Total_Ct_Chng_Q4_Q1 >  0.26
|   |   |   |   |--- Total_Trans_Amt <= 0.09
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Trans_Amt >  0.09
|   |   |   |   |   |--- class: False
```

```
|   |   |--- Total_Relationship_Count >  0.30
|   |   |   |--- Total_Trans_Amt <= 0.09
|   |   |   |   |--- Transactions_Per_Month <= 0.02
|   |   |   |   |   |--- class: False
|   |   |   |   |--- Transactions_Per_Month >  0.02
|   |   |   |   |   |--- class: True
|   |   |   |--- Total_Trans_Amt >  0.09
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 <= 0.17
|   |   |   |   |   |--- class: False
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 >  0.17
|   |   |   |   |   |--- class: True
|--- Total_Trans_Ct >  0.34
|   |--- Total_Trans_Amt <= 0.27
|   |   |--- Total_Trans_Ct <= 0.37
|   |   |   |--- Total_Relationship_Count <= 0.30
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 <= 0.24
|   |   |   |   |   |--- class: False
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 >  0.24
|   |   |   |   |   |--- class: False
|   |   |   |--- Total_Relationship_Count >  0.30
|   |   |   |   |--- Total_Trans_Amt <= 0.23
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Trans_Amt >  0.23
|   |   |   |   |   |--- class: False
|   |   |--- Total_Trans_Ct >  0.37
|   |   |   |--- Total_Trans_Ct <= 0.41
|   |   |   |   |--- Total_Revolving_Bal <= 0.16
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Revolving_Bal >  0.16
|   |   |   |   |   |--- class: True
|   |   |   |--- Total_Trans_Ct >  0.41
|   |   |   |   |--- Card_Category_Platinum <= 0.50
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Card_Category_Platinum >  0.50
|   |   |   |   |   |--- class: False
|   |--- Total_Trans_Amt >  0.27
|   |   |--- Total_Trans_Ct <= 0.53
|   |   |   |--- Total_Revolving_Bal <= 0.34
|   |   |   |   |--- Credit_Limit <= 0.04
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Credit_Limit >  0.04
|   |   |   |   |   |--- class: False
|   |   |   |--- Total_Revolving_Bal >  0.34
|   |   |   |   |--- Total_Amt_Chng_Q4_Q1 <= 0.26
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Amt_Chng_Q4_Q1 >  0.26
|   |   |   |   |   |--- class: False
|   |   |--- Total_Trans_Ct >  0.53
|   |   |   |--- Total_Trans_Ct <= 0.60
```

```
|   |   |   |   |--- Total_Trans_Amt <= 0.44
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Trans_Amt >  0.44
|   |   |   |   |   |--- class: False
|   |   |   |--- Total_Trans_Ct >  0.60
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 <= 0.25
|   |   |   |   |   |--- class: True
|   |   |   |   |--- Total_Ct_Chng_Q4_Q1 >  0.25
|   |   |   |   |   |--- class: True
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update this with the correct path to your preprocessed dataset
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Change this to
your target column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)

rf_classifier.fit(X_train, y_train)

y_pred = rf_classifier.predict(X_test)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

feature_importances = pd.DataFrame({
    'Feature': features.columns,
    'Importance': rf_classifier.feature_importances_
}).sort_values(by='Importance', ascending=False)
```

```python
print("\nFeature Importances:")
print(feature_importances)
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.93      0.82      0.87       325
        True       0.97      0.99      0.98      1701

    accuracy                           0.96      2026
   macro avg       0.95      0.90      0.92      2026
weighted avg       0.96      0.96      0.96      2026

Accuracy Score: 0.9610069101678184
Confusion Matrix:
[[ 265   60]
 [  19 1682]]

Feature Importances:
                           Feature  Importance
11                  Total_Trans_Ct    0.142320
10                 Total_Trans_Amt    0.128612
32          Transactions_Per_Month    0.122163
12             Total_Ct_Chng_Q4_Q1    0.106575
7              Total_Revolving_Bal    0.102195
9             Total_Amt_Chng_Q4_Q1    0.056982
13           Avg_Utilization_Ratio    0.056403
3         Total_Relationship_Count    0.056097
0                     Customer_Age    0.033364
6                     Credit_Limit    0.029344
8                  Avg_Open_To_Buy    0.027856
2                   Months_on_book    0.024362
4           Months_Inactive_12_mon    0.022898
5            Contacts_Count_12_mon    0.021790
1                  Dependent_count    0.013752
14                        Gender_M    0.008632
21          Marital_Status_Married    0.005864
22           Marital_Status_Single    0.004623
27  Income_Category_Less than $40K    0.003727
16        Education_Level_Graduate    0.003555
17     Education_Level_High School    0.003200
20         Education_Level_Unknown    0.003151
25     Income_Category_$60K - $80K    0.002827
19      Education_Level_Uneducated    0.002805
26    Income_Category_$80K - $120K    0.002779
24     Income_Category_$40K - $60K    0.002635
23          Marital_Status_Unknown    0.002301
28         Income_Category_Unknown    0.002103
18   Education_Level_Post-Graduate    0.002081
15       Education_Level_Doctorate    0.001929
```

```
31            Card_Category_Silver    0.001666
29              Card_Category_Gold    0.001164
30          Card_Category_Platinum    0.000245
```

```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in
/opt/conda/lib/python3.11/site-packages (0.12.4)
Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/lib/python3.11/site-packages (from imbalanced-learn)
(1.26.4)
Requirement already satisfied: scipy>=1.5.0 in
/opt/conda/lib/python3.11/site-packages (from imbalanced-learn)
(1.12.0)
Requirement already satisfied: scikit-learn>=1.0.2 in
/opt/conda/lib/python3.11/site-packages (from imbalanced-learn)
(1.5.1)
Requirement already satisfied: joblib>=1.1.1 in
/opt/conda/lib/python3.11/site-packages (from imbalanced-learn)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.11/site-packages (from imbalanced-learn)
(3.5.0)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Update this with
your target column name
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

smote = SMOTE(random_state=42)
```

```python
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled,
y_train)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train_smote, y_train_smote)

y_pred = rf_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

feature_importances = pd.DataFrame({
    'Feature': features.columns,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.87      0.90      0.89       325
        True       0.98      0.97      0.98      1701

    accuracy                           0.96      2026
   macro avg       0.93      0.94      0.93      2026
weighted avg       0.96      0.96      0.96      2026

Accuracy Score: 0.9629812438302073
Confusion Matrix:
[[ 293   32]
 [  43 1658]]

Feature Importances:
                       Feature  Importance
11              Total_Trans_Ct    0.182145
10             Total_Trans_Amt    0.143929
32      Transactions_Per_Month    0.128141
7          Total_Revolving_Bal    0.085220
12         Total_Ct_Chng_Q4_Q1    0.075869
3     Total_Relationship_Count    0.054354
4       Months_Inactive_12_mon    0.050511
9          Total_Amt_Chng_Q4_Q1    0.045538
13       Avg_Utilization_Ratio    0.044740
```

```
5            Contacts_Count_12_mon    0.042764
0                     Customer_Age    0.024315
6                     Credit_Limit    0.019517
2                   Months_on_book    0.018512
8                  Avg_Open_To_Buy    0.018491
1                  Dependent_count    0.017042
14                        Gender_M    0.010819
21          Marital_Status_Married    0.005848
22           Marital_Status_Single    0.005107
16        Education_Level_Graduate    0.003113
27  Income_Category_Less than $40K    0.002932
26    Income_Category_$80K - $120K    0.002914
19      Education_Level_Uneducated    0.002333
17     Education_Level_High School    0.002300
24     Income_Category_$40K - $60K    0.002207
25     Income_Category_$60K - $80K    0.001956
20         Education_Level_Unknown    0.001893
18   Education_Level_Post-Graduate    0.001628
28         Income_Category_Unknown    0.001492
23          Marital_Status_Unknown    0.001309
31            Card_Category_Silver    0.001116
15       Education_Level_Doctorate    0.001011
29              Card_Category_Gold    0.000805
30          Card_Category_Platinum    0.000128
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.preprocessing import StandardScaler

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Update this with
your target column name
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

gb_model = GradientBoostingClassifier(n_estimators=100,
```

```python
learning_rate=0.1, max_depth=3, random_state=42)

gb_model.fit(X_train_scaled, y_train)

y_pred = gb_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

feature_importances = pd.DataFrame({
    'Feature': features.columns,
    'Importance': gb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.93      0.85      0.89       325
        True       0.97      0.99      0.98      1701

    accuracy                           0.97      2026
   macro avg       0.95      0.92      0.93      2026
weighted avg       0.96      0.97      0.96      2026

Accuracy Score: 0.9654491609081934
Confusion Matrix:
[[ 276   49]
 [  21 1680]]

Feature Importances:
                           Feature  Importance
11                  Total_Trans_Ct    0.333336
10                 Total_Trans_Amt    0.186573
7              Total_Revolving_Bal    0.166748
12             Total_Ct_Chng_Q4_Q1    0.118200
3         Total_Relationship_Count    0.094148
9             Total_Amt_Chng_Q4_Q1    0.033666
32          Transactions_Per_Month    0.018806
4           Months_Inactive_12_mon    0.015172
0                     Customer_Age    0.014258
5            Contacts_Count_12_mon    0.007577
8                  Avg_Open_To_Buy    0.002917
2                   Months_on_book    0.001867
```

```
14                        Gender_M    0.001674
21          Marital_Status_Married    0.001650
1                  Dependent_count    0.001225
13           Avg_Utilization_Ratio    0.000766
30          Card_Category_Platinum    0.000631
29              Card_Category_Gold    0.000475
6                     Credit_Limit    0.000161
22           Marital_Status_Single    0.000095
16        Education_Level_Graduate    0.000054
20         Education_Level_Unknown    0.000002
19      Education_Level_Uneducated    0.000000
15       Education_Level_Doctorate    0.000000
24     Income_Category_$40K - $60K    0.000000
25     Income_Category_$60K - $80K    0.000000
26    Income_Category_$80K - $120K    0.000000
27  Income_Category_Less than $40K    0.000000
28         Income_Category_Unknown    0.000000
18   Education_Level_Post-Graduate    0.000000
17     Education_Level_High School    0.000000
31            Card_Category_Silver    0.000000
23          Marital_Status_Unknown    0.000000
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.preprocessing import StandardScaler
from lightgbm import LGBMClassifier

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Correct target
column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

lgbm_model = LGBMClassifier(
    objective='binary',
    boosting_type='gbdt',
    learning_rate=0.1,
```

```python
    n_estimators=100,
    num_leaves=31,
    random_state=42
)

lgbm_model.fit(
    X_train_scaled,
    y_train,
    eval_set=[(X_test_scaled, y_test)],
    eval_metric='logloss'
)

y_pred = lgbm_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

feature_importances = pd.DataFrame({
    'Feature': features.columns,
    'Importance': lgbm_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

```
[LightGBM] [Info] Number of positive: 6799, number of negative: 1302
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.005755 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2322
[LightGBM] [Info] Number of data points in the train set: 8101, number
of used features: 32
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.839279 ->
initscore=1.652874
[LightGBM] [Info] Start training from score 1.652874
Classification Report:
              precision    recall  f1-score   support

       False       0.91      0.91      0.91       325
        True       0.98      0.98      0.98      1701

    accuracy                           0.97      2026
   macro avg       0.95      0.95      0.95      2026
weighted avg       0.97      0.97      0.97      2026
```

```
Accuracy Score: 0.9718657453109576
Confusion Matrix:
[[ 297   28]
 [  29 1672]]

Feature Importances:
                           Feature  Importance
10                 Total_Trans_Amt         543
11                  Total_Trans_Ct         362
9              Total_Amt_Chng_Q4_Q1         350
12              Total_Ct_Chng_Q4_Q1         238
32            Transactions_Per_Month         205
0                     Customer_Age         179
7               Total_Revolving_Bal         172
3          Total_Relationship_Count         153
6                     Credit_Limit         135
5             Contacts_Count_12_mon         110
2                   Months_on_book         107
8                  Avg_Open_To_Buy         103
4            Months_Inactive_12_mon          92
13            Avg_Utilization_Ratio          66
21           Marital_Status_Married          40
1                  Dependent_count          39
14                        Gender_M          25
25      Income_Category_$60K - $80K          16
18   Education_Level_Post-Graduate          10
22            Marital_Status_Single           9
24      Income_Category_$40K - $60K           8
16         Education_Level_Graduate           7
20          Education_Level_Unknown           7
19       Education_Level_Uneducated           6
15        Education_Level_Doctorate           4
27  Income_Category_Less than $40K           4
23           Marital_Status_Unknown           3
26     Income_Category_$80K - $120K           3
17      Education_Level_High School           2
28          Income_Category_Unknown           2
29               Card_Category_Gold           0
30           Card_Category_Platinum           0
31             Card_Category_Silver           0

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
```

```
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Correct target column
features = data.drop(columns=[target_column])  # Remove the target column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

xgb_model = XGBClassifier(
    objective='binary:logistic',
    learning_rate=0.1,
    n_estimators=100,
    max_depth=5,
    random_state=42,
    use_label_encoder=False,  # To avoid a warning for newer versions of XGBoost
    eval_metric='logloss'  # Specify evaluation metric
)

xgb_model.fit(X_train_scaled, y_train)

y_pred = xgb_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

feature_importances = pd.DataFrame({
    'Feature': features.columns,
    'Importance': xgb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.92      0.90      0.91       325
```

```
        True       0.98      0.99      0.98      1701

    accuracy                           0.97      2026
   macro avg       0.95      0.94      0.95      2026
weighted avg       0.97      0.97      0.97      2026

Accuracy Score: 0.9713721618953604
Confusion Matrix:
[[ 292   33]
 [  25 1676]]

Feature Importances:
                           Feature  Importance
11                    Total_Trans_Ct    0.241002
7                Total_Revolving_Bal    0.151338
3           Total_Relationship_Count    0.103535
10                   Total_Trans_Amt    0.059479
12               Total_Ct_Chng_Q4_Q1    0.049363
4             Months_Inactive_12_mon    0.043459
8                     Avg_Open_To_Buy    0.033538
9               Total_Amt_Chng_Q4_Q1    0.030702
0                       Customer_Age    0.029823
32            Transactions_Per_Month    0.027283
14                          Gender_M    0.027039
5              Contacts_Count_12_mon    0.022109
6                       Credit_Limit    0.018385
21            Marital_Status_Married    0.016800
2                     Months_on_book    0.014726
1                    Dependent_count    0.013676
23            Marital_Status_Unknown    0.012103
13             Avg_Utilization_Ratio    0.011663
15         Education_Level_Doctorate    0.011065
28           Income_Category_Unknown    0.010864
16          Education_Level_Graduate    0.010569
25      Income_Category_$60K - $80K    0.010164
18    Education_Level_Post-Graduate    0.009461
19       Education_Level_Uneducated    0.007283
27  Income_Category_Less than $40K    0.006160
22             Marital_Status_Single    0.005295
17      Education_Level_High School    0.005217
24      Income_Category_$40K - $60K    0.005143
26     Income_Category_$80K - $120K    0.004528
29               Card_Category_Gold    0.004250
20           Education_Level_Unknown    0.003978
30           Card_Category_Platinum    0.000000
31             Card_Category_Silver    0.000000


import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Correct target
column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

xgb_model = XGBClassifier(
    objective='binary:logistic',
    learning_rate=0.1,
    n_estimators=100,
    max_depth=5,
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss'
)

rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=5,
    random_state=42
)

lr_model = LogisticRegression(
    random_state=42,
    max_iter=1000
)

ensemble_model = VotingClassifier(
    estimators=[
        ('xgb', xgb_model),
        ('rf', rf_model),
        ('lr', lr_model)
```

```
    ],
    voting='hard'  # Use 'soft' for probability-based voting
)

ensemble_model.fit(X_train_scaled, y_train)

y_pred = ensemble_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.92      0.71      0.80       325
        True       0.95      0.99      0.97      1701

    accuracy                           0.94      2026
   macro avg       0.93      0.85      0.88      2026
weighted avg       0.94      0.94      0.94      2026

Accuracy Score: 0.9432379072063178
Confusion Matrix:
[[ 230   95]
 [  20 1681]]
```

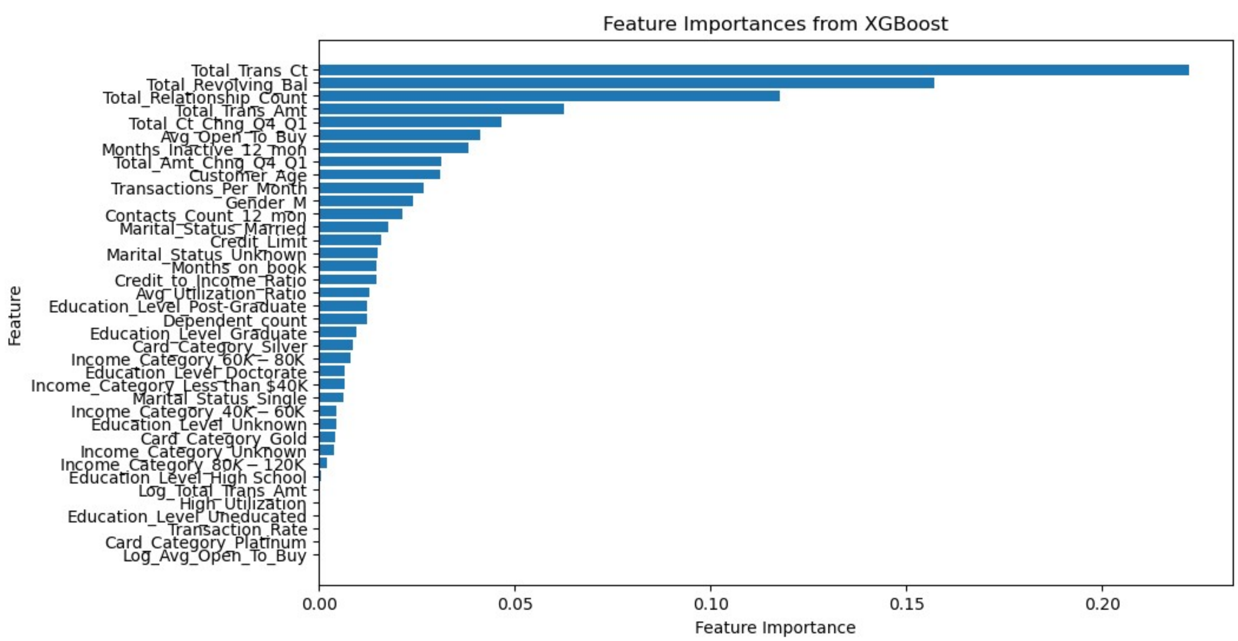
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from xgboost import XGBClassifier
import numpy as np

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Correct target
column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

features['Credit_to_Income_Ratio'] = features['Credit_Limit'] /
(features['Avg_Open_To_Buy'] + 1)
```

```
features['Transaction_Rate'] = features['Total_Trans_Amt'] /
(features['Months_on_book'] + 1)

features['High_Utilization'] = (features['Avg_Utilization_Ratio'] >
0.5).astype(int)

for col in ['Total_Trans_Amt', 'Avg_Open_To_Buy']:
    features[f'Log_{col}'] = np.log1p(features[col])

categorical_cols = features.select_dtypes(include=['object']).columns
if len(categorical_cols) > 0:
    encoder = OneHotEncoder(sparse=False, drop='first')
    encoded_cols = encoder.fit_transform(features[categorical_cols])
    encoded_features = pd.DataFrame(encoded_cols,
columns=encoder.get_feature_names_out(categorical_cols))
    features = pd.concat([features.drop(columns=categorical_cols),
encoded_features], axis=1)

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

xgb_model = XGBClassifier(
    objective='binary:logistic',
    learning_rate=0.1,
    n_estimators=100,
    max_depth=5,
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss'
)

xgb_model.fit(X_train_scaled, y_train)

y_pred = xgb_model.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

import matplotlib.pyplot as plt

feature_importances = pd.DataFrame({
```

```python
    'Feature': features.columns,
    'Importance': xgb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)

plt.figure(figsize=(10, 6))
plt.barh(feature_importances['Feature'],
feature_importances['Importance'])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances from XGBoost')
plt.gca().invert_yaxis()
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.91      0.90      0.91       325
        True       0.98      0.98      0.98      1701

    accuracy                           0.97      2026
   macro avg       0.95      0.94      0.94      2026
weighted avg       0.97      0.97      0.97      2026

Accuracy Score: 0.9703849950641659
Confusion Matrix:
[[ 293   32]
 [  28 1673]]

Feature Importances:
                       Feature  Importance
11              Total_Trans_Ct    0.222327
7          Total_Revolving_Bal    0.157008
3     Total_Relationship_Count    0.117635
10             Total_Trans_Amt    0.062698
12         Total_Ct_Chng_Q4_Q1    0.046681
8              Avg_Open_To_Buy    0.041149
4       Months_Inactive_12_mon    0.038210
9         Total_Amt_Chng_Q4_Q1    0.031182
0                 Customer_Age    0.031005
32      Transactions_Per_Month    0.026838
14                    Gender_M    0.024096
5        Contacts_Count_12_mon    0.021386
21      Marital_Status_Married    0.017549
6                 Credit_Limit    0.015906
23      Marital_Status_Unknown    0.014859
2               Months_on_book    0.014714
33      Credit_to_Income_Ratio    0.014592
```

```
13           Avg_Utilization_Ratio    0.012785
18   Education_Level_Post-Graduate    0.012311
1                  Dependent_count    0.012289
16        Education_Level_Graduate    0.009521
31            Card_Category_Silver    0.008518
25     Income_Category_$60K - $80K    0.007960
15       Education_Level_Doctorate    0.006525
27  Income_Category_Less than $40K    0.006424
22           Marital_Status_Single    0.006138
24     Income_Category_$40K - $60K    0.004566
20         Education_Level_Unknown    0.004320
29              Card_Category_Gold    0.004271
28         Income_Category_Unknown    0.003815
26    Income_Category_$80K - $120K    0.002059
17     Education_Level_High School    0.000665
36             Log_Total_Trans_Amt    0.000000
35                High_Utilization    0.000000
19      Education_Level_Uneducated    0.000000
34                Transaction_Rate    0.000000
30          Card_Category_Platinum    0.000000
37             Log_Avg_Open_To_Buy    0.000000
```




```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFE
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
```

```python
from xgboost import XGBClassifier

file_path = '/home/jovyan/Web_Recommender/Cleaned_dataset.csv'  #
Update with your dataset path
data = pd.read_csv(file_path)

target_column = 'Attrition_Flag_Existing Customer'  # Correct target
column
features = data.drop(columns=[target_column])  # Remove the target
column from features
target = data[target_column]

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42, stratify=target)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

xgb_model = XGBClassifier(
    objective='binary:logistic',
    learning_rate=0.1,
    n_estimators=100,
    max_depth=5,
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss'
)

rfe = RFE(estimator=xgb_model, n_features_to_select=10, step=1)  #
Select top 10 features
rfe.fit(X_train_scaled, y_train)

selected_features = features.columns[rfe.support_]
print("Selected Features:")
print(selected_features)

X_train_rfe = rfe.transform(X_train_scaled)
X_test_rfe = rfe.transform(X_test_scaled)

xgb_model.fit(X_train_rfe, y_train)

y_pred = xgb_model.predict(X_test_rfe)

print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

print("Confusion Matrix:")
```

```python
print(confusion_matrix(y_test, y_pred))

import matplotlib.pyplot as plt

feature_importances = pd.DataFrame({
    'Feature': selected_features,
    'Importance': xgb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances for Selected Features:")
print(feature_importances)

plt.figure(figsize=(10, 6))
plt.barh(feature_importances['Feature'],
feature_importances['Importance'])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances for Selected Features (RFE)')
plt.gca().invert_yaxis()
plt.show()
```

```
Selected Features:
Index(['Customer_Age', 'Total_Relationship_Count',
'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Total_Revolving_Bal',
'Avg_Open_To_Buy',
       'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt', 'Total_Trans_Ct',
       'Total_Ct_Chng_Q4_Q1'],
      dtype='object')
Classification Report:
              precision    recall  f1-score   support

       False       0.92      0.91      0.91       325
        True       0.98      0.98      0.98      1701

    accuracy                           0.97      2026
   macro avg       0.95      0.95      0.95      2026
weighted avg       0.97      0.97      0.97      2026

Accuracy Score: 0.9718657453109576
Confusion Matrix:
[[ 295   30]
 [  27 1674]]

Feature Importances for Selected Features:
                    Feature  Importance
8            Total_Trans_Ct    0.293107
4       Total_Revolving_Bal    0.199824
1  Total_Relationship_Count    0.154497
7           Total_Trans_Amt    0.083097
```

```
9       Total_Ct_Chng_Q4_Q1    0.068072
2    Months_Inactive_12_mon    0.053755
6      Total_Amt_Chng_Q4_Q1    0.042817
0              Customer_Age    0.039844
5           Avg_Open_To_Buy    0.034047
3     Contacts_Count_12_mon    0.030942
```