Google Collab: Bank Churn Prediction

Table of Contents

1.0 Introduction

1.1 Project Background

In the credit card business, customer loyalty is the main problem that banks must deal with. A fair number of customers stop using their credit cards for various reasons, and this situation, which is usually called customer churn, can result in a shrinkage of profits. Since acquiring new clients usually costs more than maintaining existing ones, banks nowadays allocate more resources towards determining the reasons behind customer departure and retaining their clients (HubSpot, 2023).

With the help of data science, it has now become possible to study client activities and identify the following patterns: warning signs of churn. By analyzing factors like age, income level, card usage, and overall credit limit, we stand in a position to roughly figure which customers are on the verge of losing their credit cards. Such knowledge enables banks to be proactive and take early action, such as providing special promotions or expanding the range of services offered, which meet customer needs better.

Our project will analyze a dataset with information about credit card customers and their activities. The core of this research is to examine the data and identify key factors that drive customer attrition, followed by development of a model that can predict it. Through such a model, banks might not only avert loss of clientele but also assure themselves of providing more personalized services that ensure increased gratification and loyalty of their clients.

1.2 Project Goal

The goal of the project is to develop a predictive model that will help to predict which factors cause credit card customers to churn. Using data science techniques and models using python will help to determine possible correlations that could help identify a potential pattern. The model will provide evidence for the bank to evaluate which type of customers could possibly leave and allow the bank to take action with the insights we provide in our report by supplying them with the pattern and report on the analysis around what the bank may need to change.

- To identify and select a relevant number of features that are indicative of churn patterns - including both financial behaviours (i.e. dollar amounts related to transactions) and non-financial behaviours (i.e. demographic information)
- To analyze the dataset and uncover which variables have the strongest influence on whether a customer will churn.
- To build and test several machine learning models that can accurately predict churn, aiming for a performance accuracy of at least 80%.
- To interpret the results of our model in a meaningful way, highlighting patterns and relationships, and to suggest practical strategies that banks can apply to retain at-risk customers.

Therefore, data mining techniques are being used to enhance our understanding of customer attrition, improve churn prediction, support early intervention strategies, and help banks personalise retention efforts for at-risk customers.

2.0 **Overview of the Dataset**

2.1 **Source and description of the dataset**

The dataset used in this project is taken from Kaggle uploaded by Sakshi Goyals, which included customer banking information like demographic details, credit card usage patterns, financial behaviours and predictive analytics related to customer attrition. These include factors such as customer age, income category, total transaction amount, credit limit, and card category. The purpose of this dataset is to perform predictive analysis to predict the likelihood of customer attrition of a bank based on various factors. The dataset consists of 10 000 rows and 23 columns, with one target variables. The explanation of each column are as follows:

Column name	Description
CLIENTNUM	A unique identifier that is assigned to each customer, which is numerical data.
Attrition Flag	Indicates whether the customer has churned or is still active, which is categorical data.
Customer Age	The age of the customer, which is a numerical data.
Gender	The gender of the customer, which is a categorical data.
Dependent Count	The number of dependents that the customer has, which is a numerical data.
Education Level	The highest level of education attained by the customer, which is a categorical data.
Marital Status	Indicate whether the customer is married, single or divorced, which is a categorical data.
Income Category	The income category of customer, which is a categorical data
Card Category	The type of credit card that the customer has, which is a categorical data.
Months on Book	The duration of the customer's relationship with the bank, which is a numerical data.
Contacts Count 12 Months	The number of the contacts the customer has had with the bank in the last 12 months, which is a numerical data.
Credit Limit	The credit limit assigned to the customer, which is a numerical data.
Total Revolving Balance	The total revolving balance on the customer's credit card, which is a numerical data.
Average Open to Buy	The average amount available to the customer to spend, which is a numerical data.
Total Amount Change Q4 to Q1	The change in transaction amount form the fourth quarter to the first quarter, which is a numerical data.
Total Transaction Amount	The total amount of transactions made by the customer, which is a numerical data.
Total Transaction Credit	The total count of transactions made by the customer, which is a numerical data.
Total Credit Change Q4 to Q1	The change in transaction count from the fourth quarter to the first quarter, which is a numerical data.
Avg Utilization Ratio	The average utilization ratio of the credit card (calculated as Total Balance divided by Credit Limit), which is a numerical data.

```
DataFrame Shape: (10127, 21)

First 5 rows:
    CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0  768805383  Existing Customer            45      M                3
1  818770008  Existing Customer            49      F                5
2  713982108  Existing Customer            51      M                3
3  769911858  Existing Customer            40      F                4
4  709106358  Existing Customer            40      M                3

  Education_Level Marital_Status Income_Category Card_Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
3     High School        Unknown  Less than $40K          Blue
4      Uneducated        Married     $60K - $80K          Blue

   Months_on_book  ...  Months_Inactive_12_mon  Contacts_Count_12_mon  \
0              39  ...                       1                      3
1              44  ...                       1                      2
2              36  ...                       1                      0
3              34  ...                       4                      1
4              21  ...                       1                      0

   Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  Total_Amt_Chng_Q4_Q1  \
0       12691.0                  777          11914.0                 1.335
1        8256.0                  864           7392.0                 1.541
2        3418.0                    0           3418.0                 2.594
3        3313.0                 2517            796.0                 1.405
4        4716.0                    0           4716.0                 2.175

   Total_Trans_Amt  Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
0             1144              42                1.625                  0.061
1             1291              33                3.714                  0.105
2             1887              20                2.333                  0.000
3             1171              20                2.333                  0.760
4              816              28                2.500                  0.000
```

```
Last 5 rows:
       CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
10122  772366833  Existing Customer            50      M                2
10123  710638233  Attrited Customer            41      M                2
10124  716506083  Attrited Customer            44      F                1
10125  717406983  Attrited Customer            30      M                2
10126  714337233  Attrited Customer            43      F                2

      Education_Level Marital_Status Income_Category Card_Category  \
10122        Graduate         Single     $40K - $60K          Blue
10123         Unknown       Divorced     $40K - $60K          Blue
10124     High School        Married  Less than $40K          Blue
10125        Graduate        Unknown     $40K - $60K          Blue
10126        Graduate        Married  Less than $40K        Silver

       Months_on_book  ...  Months_Inactive_12_mon  Contacts_Count_12_mon  \
10122              40  ...                       2                      3
10123              25  ...                       2                      3
10124              36  ...                       3                      4
10125              36  ...                       3                      3
10126              25  ...                       2                      4

       Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  \
10122        4003.0                 1851           2152.0
10123        4277.0                 2186           2091.0
10124        5409.0                    0           5409.0
10125        5281.0                    0           5281.0
10126       10388.0                 1961           8427.0

       Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
10122                 0.703            15476             117
10123                 0.804             8764              69
10124                 0.819            10291              60
10125                 0.535             8395              62
10126                 0.703            10294              61



       Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
10122                0.857                 0.462
10123                0.683                 0.511
10124                0.818                 0.000
10125                0.722                 0.000
10126                0.649                 0.189

[5 rows x 21 columns]
```

Fig. 2.1.0 Code output of dataset overview

2.2 High-level statistics and Exploratory Data Analysis

High-level statistics offer a good first step in exploring the dataset. It offers a summary of the key features and distributions of the dataset, providing a rudimentary understanding of the data before scrutinizing it more in depth. It does this by summarizing the structure of the data, recognizing trends, and identifying anomalies before starting modelling (Hayes, 2024). In the case of the bank churner dataset we made use of high-level statistics to summarize both numerical and categorical data, including investigating customer behaviour, and checking for possible relationships between churn and the other columns.

```
# ----------------------------------------------------
# 1.0 DATA OVERVIEW
# ----------------------------------------------------

# Drop last few columns that are irrelevant (based on the uploader statement on Kaggle)
df = df.iloc[:, :-2]

# Initial Data Overview
print("DataFrame Shape:", df.shape)
print("\nFirst 5 rows:\n", df.head())
print("\nLast 5 rows:\n", df.tail())
print("\nData Types and Missing Values:\n")
df.info()

print("Statistical summary of features:")
print(df.describe())
```

Fig. 2.2.1. Code snippet high-level statistical summary of features.

```
Statistical summary of features:
          CLIENTNUM  Customer_Age  Dependent_count  Months_on_book  \
count  1.012700e+04  10127.000000     10127.000000    10127.000000
mean   7.391776e+08     46.325960         2.346203       35.928409
std    3.690378e+07      8.016814         1.298908        7.986416
min    7.080821e+08     26.000000         0.000000       13.000000
25%    7.130368e+08     41.000000         1.000000       31.000000
50%    7.179264e+08     46.000000         2.000000       36.000000
75%    7.731435e+08     52.000000         3.000000       40.000000
max    8.283431e+08     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
```

Fig. 2.2.2a Code output of high-level statistical summary of features.

```
       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
```

Fig. 2.2.b Code output of high-level statistical summary of features.

To do this, four statistical techniques and visualisations were applied: histograms, attrition flag counts, boxplots, and correlation heatmaps. Collectively, this high-level summary statistics are a total first look at the dataset. They will then be the basis for a more detailed analysis by identifying important patterns, possible relationships, and underlying data quality issues. This first chance to understand this data ensures that any predictive modeling chosen later (for example, churn prediction) is based on data that is clearly understood and prepared.

```python
# ------------------------------------
# 4.0 Exploratory Data Analysis (EDA)
# ------------------------------------

# 4.1 Distribution of all features

plt.figure(figsize=(20, 35))
for i, col in enumerate(df.columns):
    plt.subplot((len(df.columns) + 2) // 3, 3, i + 1)
    sns.histplot(df[col], kde=False, bins=30)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.suptitle('Distribution of All Features', fontsize=22, y=1.02)
plt.show()
```

Fig. 2.2.0. Code snippet to create histograms for the distribution of all columns.

Fig. 2.2.1 Output of the code snippet in Fig. 2.2.0.

The 21 histograms provided several important findings. The 'Attrition Flag' column had a great deal of imbalance with a much larger number of existing customers compared to churned customers. Customer demographic distributions also had lots of data to share. It appeared that most customers were aged 40-50, there was a bit of imbalance in the gender distribution, education level was dominated by graduates, marital status was mostly married or single, income appeared to be concentrated in the lower-middle, and when looking at Banking Usage Distribution, it appeared that the majority of bank customers held 'Blue' cards. When looking into dependents, the majority of customers are represented as having 0-3 dependents. When looking at Behavioural attributes, Months On Book was concentrated in 30-40 months of AMD, while it appears that more customers had low score counts in 'Months Inactive 12 Mon' and 'Contacts Count 12 Mon' attributes which suggest that more customers are interacting with their bank accounts regularly.Financial metrics such as ‘Credit Limit’, ‘Total Revolving Balance’, ‘Avg Open To Buy’, and ‘Total Transaction Amount’ are all heavily skewed to the left, indicating that there are much more customers with lower values than those with higher ones. The ‘Total Transaction Count’ column has 2 peaks, one relatively center and one slightly skewed to the left. This implies there are distinct customer segments based on transaction frequency.

```
# 4.2 Count of Attrition Flag by each Categorical Feature

for col in df.columns:
    if col != 'Attrition Flag' and df[col].dtype in ['object', 'category']:
        plt.figure(figsize=(10, 6))

        # Create a count plot with hue='Attrition_Flag'
        sns.countplot(
            x=col,
            hue='Attrition Flag',  # Splits bars by attrition status
            data=df,
            palette='viridis'
        )

        plt.title(f'Attrition Flag Count by {col}', fontsize=14)
        plt.xlabel(col, fontsize=12)
        plt.ylabel('Count', fontsize=12)
        plt.xticks(rotation=45)
        plt.legend(title='Attrition Status')
        plt.tight_layout()
        plt.show()
```

Fig. 2.2.2 Code snippet to create a bar chart counting attrition flags within each column.

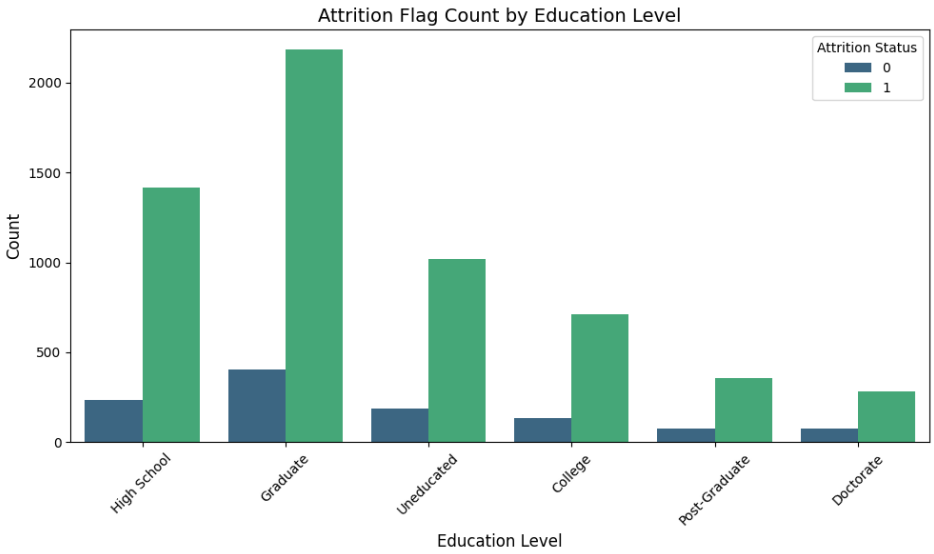

Fig. 2.2.3 Output of the code snippet in Fig. 2.2.2.

This bar chart showing attrition count by education level revealed that the highest number of churners comes from the 'Graduate', 'High School', and 'Uneducated' categories. The 'Post-Graduate' or 'Doctorate' levels of education, however both had lower attrition counts, suggesting that education may play a role in bank churning and customer retention.

```
# 4.3 Boxplot to Identify Outliers
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

n_cols = 3  # Number of columns per row
n_rows = (len(numeric_cols) + n_cols - 1) // n_cols

plt.figure(figsize=(15, 5 * n_rows))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.boxplot(y=df[col], color='lightblue')
    plt.title(col)
plt.tight_layout()
plt.show()
```

Fig. 2.2.4 Code snippet to create boxplots to show distribution of all numerical columns.

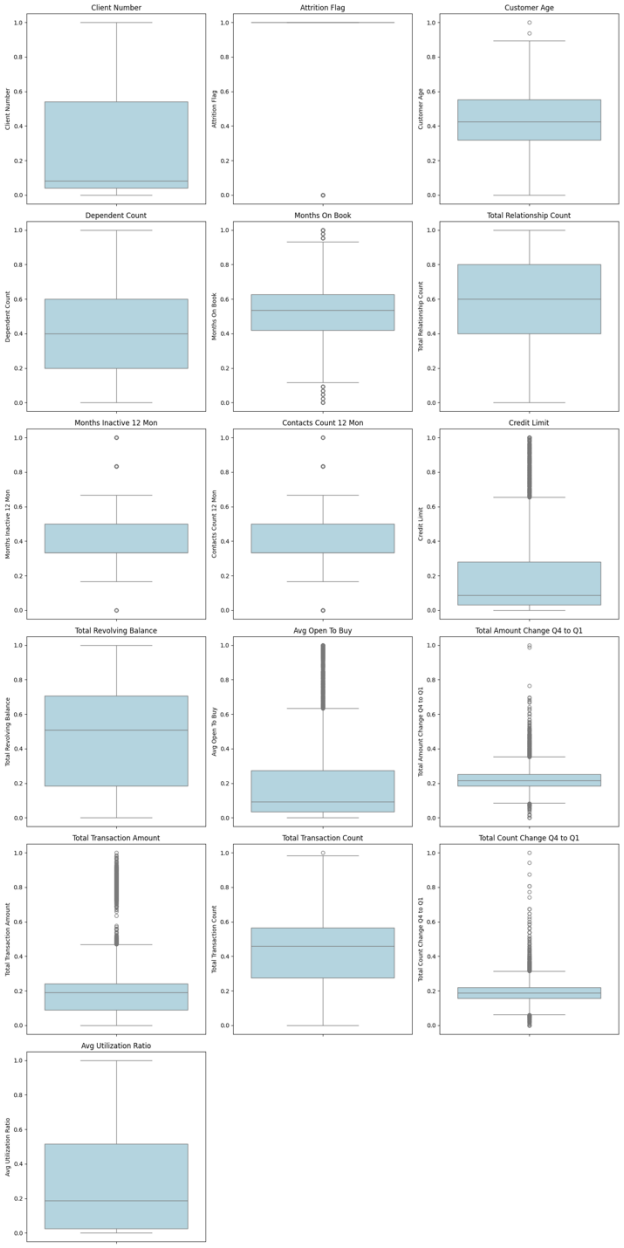

Fig. 2.2.5 Output of the code snippet in Fig. 2.2.4.

The boxplots were used to examine the distribution of each numerical feature and identify any outliers. Numerous variables such as ‘Credit Limit’, ‘Total Transaction Amount’, ‘Avg Open To Buy’, and ‘Total Count Change Q4 to Q1’ all displayed a large number of outliers, indicating long-tailed distributions, or perhaps the presence of a low number of customers with extreme values. Other features such as ‘Customer Age’, ‘Dependent Count’, and ‘Months on Book’ showcased more close and symmetric distributions.

```python
# 4.4 Correlation Heatmap

# Create a temporary copy of the DataFrame for encoding
df_temp = df.copy()

# Data Encoding (Data Transformation)
# Encoding all features in dataset
for col in df_temp.columns:
    if df_temp[col].dtype == 'object':  # If column is string type
        le = LabelEncoder()
        df_temp[col] = le.fit_transform(df_temp[col])

# Generate and plot the correlation heatmap
plt.figure(figsize=(18, 16))
corr = df_temp.corr()
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt='.2f', square=True)
plt.title('Correlation Heatmap of All Features (Encoded Copy)', fontsize=16)
plt.show()
```

Fig. 2.2.6 Code snippet to create a correlation heat map of all columns.

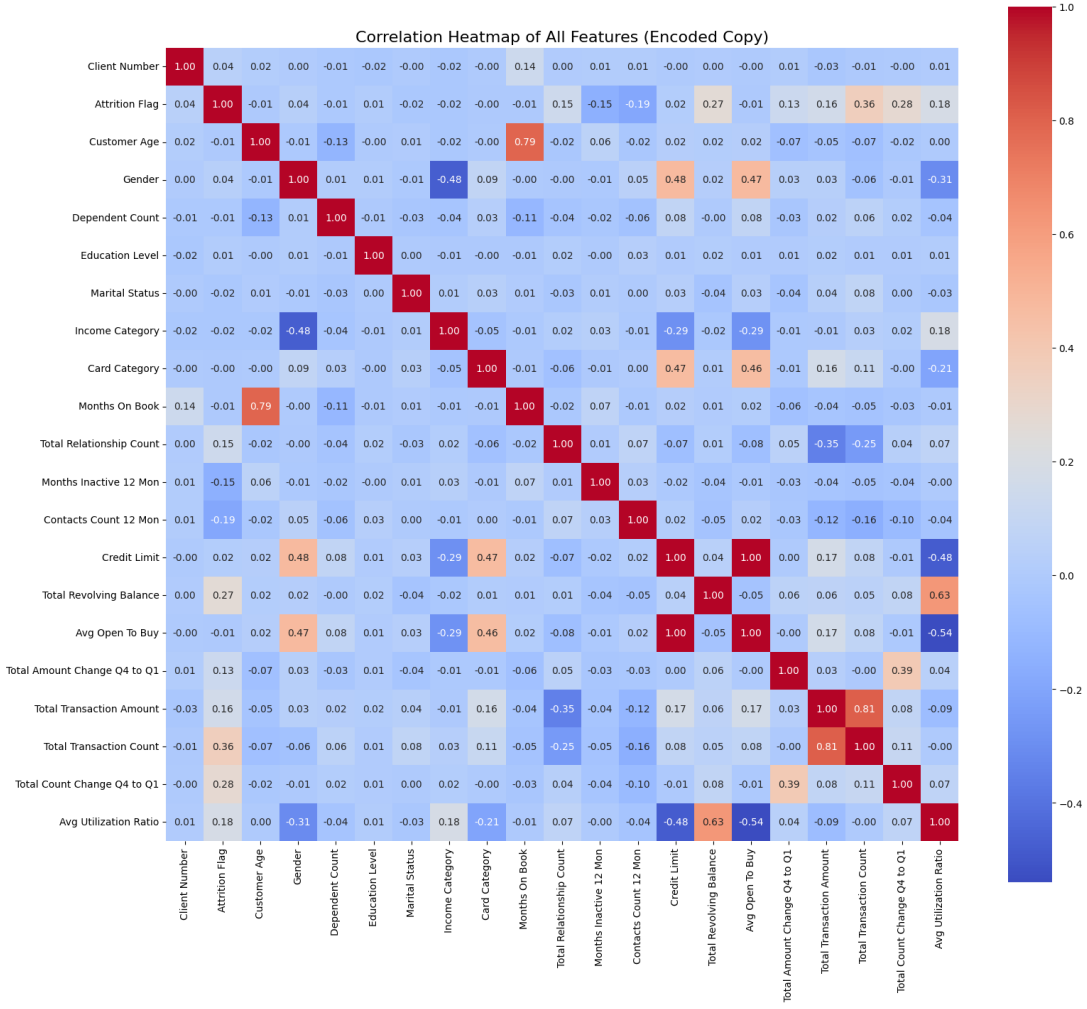

Fig. 2.2.7 Output of the code snippet in Fig. 2.2.6.

This correlation heatmap was used to examine how much the numerical attributes are related to one another. This visualisation revealed numerous strong positive relationships, such as that between ‘Total Transaction Count’ and ‘Total Transaction Amount’ with a correlation of 0.81, and to a slightly lesser degree, ‘Credit Limit’ and ‘Avg Open To Buy with a correlation of 0.63. These two relationships are those that can be thought of intuitively, as customers making more transactions naturally have a higher total spending, and customers with higher credit limits tend to have more credit available. Another relationship that can fall under this ‘intuitive’ category is the one between ‘Months On Book’ and ‘Customer Age’. This had a strong positive correlation, indicating the older a customer is, the longer they will have their bank account open. However, the churn indication column ‘Attrition Flag’ showed only weak correlations with most of the numerical attributes, suggesting that bank churning is not driven by any single feature, and is likely affected by a combination of all of them.

3.0 Pre-processing Techniques

3.1 Dataset issues

Based on the selected dataset, a number of issues have been identified that have the potential to interfere with the data efficiency, accuracy, or clarity. Among those issues are column names, missing values not easily detected through code, outliers and categorical variables. We ensure that the dataset can be used efficiently for future analysis, modelling and decision making the most accurate predictive model by addressing the issues. Data set issues:

1. **Fixing column names**
   When we reviewed the dataset, we discovered that some column names were inconsistent, illegible, and difficult to understand. For example, some column names have underscores, such as 'Dependant_Count,' while others have abbreviated words, such as 'Total_Revolving_Bal.' This may not be easily recognizable without documentation and/or context.

2. **Hidden missing values**
   We also notice that after carefully reviewing the dataset, some missing values are written as "unknown" rather than NaN or left blank. This is an issue as Python would not make "unknown" as missing because it is still a valid string, which can be very problematic during predictive modelling where the accuracy and results are influenced by this.

3. **Extreme outliers in “Credit Limit”, “Average Open to Buy”, “Total Amount Change Q4 to Q1”, “Total Transaction Amount”, and “Total Count Change Q4 to Q1” columns.**
   Boxplots revealed that several columns contain extreme outliers. These outliers lie far outside the typical rang and can negatively affect the accuracy when predictive modelling is carried out skewing the results and even inflating error metrics.

4. **Categorical Data Encoding**
   The dataset includes categorical datamust be converted into numerical data. In order to ensure that categorical data can be used effectively in machine learning models, this encoding method is required.

3.2 Handling missing values

```
# Checking for unknown values
display(df.head())
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue

5 rows × 21 columns

```
# ------------------------------------------------
# 2.0 DATA CLEANING
# ------------------------------------------------

# 2.1 Handling Missing Values

# Dataset has no 'missing values' but the initial data overview indicates some features have 'unknown' such as for Marital Status
# After looking through the raw dataset, there is  a total of 3,380 'Unknown'. These are not recognised as missing values as Pandas considers them as valid str

# Convert all unknown in the dataset to NaN values
df.replace('Unknown', np.nan, inplace=True)

# Checking if values were successfully converted to NaN values
display(df.head())
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	...	Months_I
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue	39	...	
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44	...	
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue	36	...	
3	769911858	Existing Customer	40	F	4	High School	NaN	Less than $40K	Blue	34	...	
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue	21	...	

```
# drop NaN values
df.dropna(inplace=True)

# Checking if 'unknown' values
display(df.head())
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	...	Mo
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue	39	...	
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44	...	
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue	36	...	
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue	21	...	
5	713061558	Existing Customer	44	M	2	Graduate	Married	$40K - $60K	Blue	36	...	

5 rows × 21 columns

Fig. 3.2.0 Code snippet and code output of handling missing values in the dataset

No missing values were found in the early data overview. However, when further inspection was done on the dataset it showed that "unknown" was used in columns containing categorical data as a missing value. Because "unknown" is a valid string, functions like isnull() or dropna() would not be able to identify or drop strings as they are not NaN values. Hence, we can replace cells with the string "unknown" with NaN and then properly drop the rows using dropna().

3.3 Handling outliers

```
# ------------------------------------
# 4.0 Data Pre-Processing Techniques
# ------------------------------------

# 4.1 Handling Outliers

def cap_outliers(series, lower_quantile=0.01, upper_quantile=0.99):
    lower = series.quantile(lower_quantile)
    upper = series.quantile(upper_quantile)
    return series.clip(lower, upper)

# Apply to relevant columns
cols_with_extreme_outliers = [
    'Credit Limit', 'Avg Open To Buy', 'Total Amount Change Q4 to Q1',
    'Total Transaction Amount', 'Total Count Change Q4 to Q1'
]

for col in cols_with_extreme_outliers:
    df[col] = cap_outliers(df[col])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7081 entries, 0 to 10126
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Client Number             7081 non-null   float64
 1   Attrition Flag            7081 non-null   int64
 2   Customer Age              7081 non-null   float64
 3   Gender                    7081 non-null   int64
 4   Dependent Count           7081 non-null   float64
 5   Education Level           7081 non-null   int64
 6   Marital Status            7081 non-null   int64
 7   Income Category           7081 non-null   int64
 8   Card Category             7081 non-null   int64
 9   Months On Book            7081 non-null   float64
 10  Total Relationship Count  7081 non-null   float64
 11  Months Inactive 12 Mon    7081 non-null   float64
 12  Contacts Count 12 Mon     7081 non-null   float64
 13  Credit Limit              7081 non-null   float64
 14  Total Revolving Balance   7081 non-null   float64
 15  Avg Open To Buy           7081 non-null   float64
 16  Total Amount Change Q4 to Q1  7081 non-null  float64
 17  Total Transaction Amount  7081 non-null   float64
 18  Total Transaction Count   7081 non-null   float64
 19  Total Count Change Q4 to Q1   7081 non-null  float64
 20  Avg Utilization Ratio     7081 non-null   float64
dtypes: float64(15), int64(6)
memory usage: 1.2 MB
```

Fig. 3.3.0 Code snippet and code output of outliers in the dataset

To handle extreme outliers, we applied a capping method using the 1st and 99th percentiles. This was done on columns like Credit Limit, Avg Open To Buy, Total Amount Change Q4 to Q1, Total Transaction Amount, and Total Count Change Q4 to Q1 to lessen the impact of unusually high or low values (that could distort the model resulting in bias and ultimately inaccurate predictions).

3.4 Renaming columns

```
# 2.2 Renaming the Columns
# Removing the _ in the column names and renaming short form of certain word to improve readability and to avoid misunderstanding

# Cleaning column names (remove underscores, title case)
df.columns = [col.replace('_', ' ').title() for col in df.columns]

# Renaming specific columns for clarity
rename_map = {
    "Clientnum": "Client Number",
    "Total Revolving Bal": "Total Revolving Balance",
    "Total Trans Ct": "Total Transaction Count",
    "Total Trans Amt": "Total Transaction Amount",
    "Total Amt Chng Q4 Q1": "Total Amount Change Q4 to Q1",
    "Total Ct Chng Q4 Q1": "Total Count Change Q4 to Q1"
}
df.rename(columns=rename_map, inplace=True)

# Display updated column names
df.columns.tolist()
```

```
['Client Number',
 'Attrition Flag',
 'Customer Age',
 'Gender',
 'Dependent Count',
 'Education Level',
 'Marital Status',
 'Income Category',
 'Card Category',
 'Months On Book',
 'Total Relationship Count',
 'Months Inactive 12 Mon',
 'Contacts Count 12 Mon',
 'Credit Limit',
 'Total Revolving Balance',
 'Avg Open To Buy',
 'Total Amount Change Q4 to Q1',
 'Total Transaction Amount',
 'Total Transaction Count',
 'Total Count Change Q4 to Q1',
 'Avg Utilization Ratio']
```

Fig. 3.4.0 Code snippet and code output of renaming columns for this dataset

Column names were renamed as some were abbreviated and all contained '_'. This improves the readability and avoid misunderstanding.

3.5 Data transformation

```
# ------------------------------------------------------
# 3.0 DATA TRANSFORMATION
# ------------------------------------------------------

# Encoding target categorical variable
le = LabelEncoder()
df['Attrition Flag'] = le.fit_transform(df['Attrition Flag'])  # 'Existing Customer' = 0, 'Attrited Customer' = 1

# Scaling numeric features
scaler = MinMaxScaler()
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns.drop('Attrition Flag')
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

print("\n--- Data After Transformation ---")
print(df.head())
```

```
--- Data After Transformation ---
   Client Number  Attrition Flag  Customer Age Gender  Dependent Count  \
0       0.505115               1      0.404255      M              0.6
1       0.920736               1      0.489362      F              1.0
2       0.049078               1      0.531915      M              0.6
4       0.008520               1      0.297872      M              0.6
5       0.041421               1      0.382979      M              0.4

  Education Level Marital Status Income Category Card Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
4      Uneducated        Married     $60K - $80K          Blue
5        Graduate        Married     $40K - $60K          Blue

   Months On Book  ...  Months Inactive 12 Mon  Contacts Count 12 Mon  \
0        0.604651  ...                0.166667               0.500000
1        0.720930  ...                0.166667               0.333333
2        0.534884  ...                0.166667               0.000000
4        0.186047  ...                0.166667               0.000000
5        0.534884  ...                0.166667               0.333333

   Credit Limit  Total Revolving Balance  Avg Open To Buy  \
0      0.340190                 0.308701         0.345116
1      0.206112                 0.343266         0.214093
2      0.059850                 0.000000         0.098948
4      0.099091                 0.000000         0.136557
5      0.077747                 0.495431         0.079970

   Total Amount Change Q4 to Q1  Total Transaction Amount  \
0                      0.392994                  0.036260
1                      0.453636                  0.044667
2                      0.763615                  0.078753
4                      0.640271                  0.017501
5                      0.405063                  0.033057

   Total Transaction Count  Total Count Change Q4 to Q1  Avg Utilization Ratio
0                 0.258065                     0.437534               0.061061
1                 0.185484                     1.000000               0.105105
2                 0.080645                     0.628164               0.000000
4                 0.145161                     0.673129               0.000000
5                 0.112903                     0.227787               0.311311

[5 rows x 21 columns]
```

Fig. 3.5.0 Code snippet and code output of data transformation for this dataset

To prepare the data better for predictive modelling, LabelEncoder was used to encode the values in the "Attrition Flag" column in which Attrited customers were encoded by 0 and the existing customers by 1. This gave us a defined target variable for binary classification. Next, we applied MinMax scaling on all numerical columns (excluding the target) to prepare and make them within a numerical range of 0 to 1. This was important to keep features with larger scales from overwhelming that those with smaller ranges, thus helping the model to learn faster and fairer for all contributing variables.

3.6 Feature selection

```
# 2.4 Feature Engineering
# Feature Selection

# Through the the heat correlation map as well as research done of this topic, columns including 'Client Number', 'Education Level', 'Maritial Status',
# Average Open to Buy will also be dropped as it shows little correlation and has a perfect correlation with Credit Limit

# Step 1: Manual Feature Dropping based on correlation and multicollinearity
columns_to_drop = [
    'Client Number',        # Identifier
    'Education Level',      # Low correlation
    'Marital Status',       # Low correlation
    'Card Category',        # Low correlation
    'Dependent Count',      # Low correlation
    'Avg Open To Buy'       # Perfect correlation with Credit Limit
]

# Drop only columns in df
columns_to_drop = [col for col in columns_to_drop if col in df.columns]

print(f"Shape before dropping columns: {df.shape}")
df.drop(columns=columns_to_drop, axis=1, inplace=True)
print(f"Shape after dropping columns: {df.shape}")

# Selecting top 5 features using SelectKBest
X = df.drop('Attrition Flag', axis=1)
y = df['Attrition Flag']

k = 5
skb = SelectKBest(score_func=f_classif, k=k)
X_new = skb.fit_transform(X, y)
```

```
# Get names of selected top 5 features
mask = skb.get_support()
top_k_features = X.columns[mask].tolist()

print(f"\nTop {k} features selected by SelectKBest:")
print(top_k_features)
```

```
Shape before dropping columns: (7081, 21)
Shape after dropping columns: (7081, 15)

Top 5 features selected by SelectKBest:
['Contacts Count 12 Mon', 'Total Revolving Balance', 'Total Transaction Count', 'Total Count Change Q4 to Q1', 'Avg Utilization Ratio']
```

Fig. 3.6.0 Code snippet and code output of feature selection for this dataset

For feature selection, we dropped columns like “Client Number”, “Education Level”, “Marital Status”, “Card Category”, “Dependent Count”, and “Avg Open To Buy” due to low correlation with attrition and low correlation with other features. We then used SelectKBest to double-check the top 5 features most related to the target. However, we didn’t rely only on these five, as our EDA showed that no single feature dominates, attrition is likely influenced by a combination of factors, so we kept other relevant features for modeling.

4.0 Techniques, Model Testing and Comparison

4.1 Resampling & Imbalance handling

The dataset used for churn prediction is imbalanced. It can be observed that there are significantly fewer attritted customers compared to existing ones. This might cause the prediction result of the model to be biased. The model tends to predict the majority class and fails to identify instances of churn accurately. In order to address this issue, we used two techniques, which are SMOTE (Synthetic Minority Over-sampling Technique) and class weighting, to handle the imbalance.

First, SMOTE is a technique that involves generating synthetic samples of the minority class. It enables the model to learn the decision boundaries better and reduce bias. For our project, SMOTE was used in the training pipeline to avoid data leakage and oversampling was performed only on training folds (Brownlee, 2021). Next, class weighting is applied for Random Forest and Logistic Regression models. It involves assigning a bigger penalty to misclassifying minority class samples during training. By doing this, the sensitivity to attrition cases can be improved without changing data distribution.

Model Selection

Three classification algorithms were chosen for modeling: Logistic Regression (LR), Random Forest (RF), and XGBoost. Logistic Regression was selected as it is considered simple, fast, and interpretable (Kleimann, 2020). Meanwhile, the RF can process non-linear relationships and is more accurate. It manages overfitting with multiple decision trees. Finally, we selected XGBoost due to its high accuracy. It can handle complex patterns as well as class imbalance once tuned adequately (Fatima, *et al.*, 2023). In our project, we can compare these three classification models' performances and choose the one most suitable to predict credit card churning.

4.1.1 Model Validation

- **Validation strategy: Train-Test Split vs Stratified K-Fold Cross-Validation**

At first, a typical train-test split was used. However, class balance is not always preserved in a randomly selected test set, which can lead to biased performance estimates and decreased reliability. To avoid this, we utilized Stratified K-Fold Cross-Validation, which splits the dataset into *k* equal-sized folds that retain the ratio of class labels within each fold (Olamendy, 2024). This ensures that during training and validation, both majority and minority classes are uniformly represented throughout all folds. Every instance in the dataset may be utilized for both training and validation as each fold serves as a test set once and the model is trained on the remaining $k-1$ folds. This method improves generalisability and it can detect overfitting because the model is verified on several data subsets. By averaging performance over all folds, the model can be more reliable.

4.2 Models Testing

To compare the various versions of each model using different techniques to handle class imbalance, we tested 3 variations of each classifier, where no balancing technique are used, SMOTE oversampling and class weight balancing (indicated by class_weight='balanced' or scale_pos_weight for XGBoost). All of which are implemented through imblearn.Pipeline to make sure data leakage does not happen while maintaining clean preprocessing for the code.

Within the pipeline, hyperparameter tuning was performed through GridSearchCV with stratified 5-fold cross-validation. This makes it possible to discover settings that maximize model performance by methodically exploring important parameters (such as the number of trees, maximum depth for RF, and regularization strength for LR). After which, we applied the data to ROC Curve plots and confusion matrices to get a better understanding of the differences between each model version on decisions thresholds and false positives or negative rates.

```
# Model 2: Without SMOTE but with class_weight='balanced'
pipe_weighted = SkPipeline([
    ('model', RandomForestClassifier(random_state=42, class_weight='balanced'))
])
grid_weighted = GridSearchCV(pipe_weighted, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_weighted.fit(X_train, y_train)
y_pred_weighted = grid_weighted.predict(X_test)
print("\n-----Without SMOTE, Class_weight='balanced'-----")
print("Best parameters:", grid_weighted.best_params_)
print("Best CV score:", grid_weighted.best_score_)
print("\nClassification Report:\n", classification_report(y_test, y_pred_weighted))
```

Fig. 4.2.0.1 Example of class_weight='balanced', imblearn.Pipeline and GridSearchCV being used.

After comparison between each model, we check it with Stratified K Fold and cross validation (done separately from GridSearchCV) with 5 folds. This ensures that the class distribution is consistent across the splits. Then, by using accuracy, macro F1 score, F1-score and class specific recall, we compare the performance of each model with a focus on class 0, which is the recall for churners, our main objective.

4.2.1 Feature Importance

To interpret the models, comparisons must be made. To do so, extraction of feature importance for each model needs to be done. To look for the most influential features of each model, there are specific techniques that are used that differ from model to model.

For Random Forest, the feature importance plot was used which measures the total reduction in Gini impurity across trees when paired with individual features.

For Logistic Regression, we take the learned coefficient and associate it with its number. If it is positive, it means they are more likely to churn; negative means less likely. The value determines how influential it is regardless of the sign.

For XGBoost, Plot importance was used. There are two options when using the plot importance function to get feature importance, gain and weight. We are using gain in for this situation due to weight being susceptible to fallacies as it measures how many times features are used in all splits across trees, whereas gain measures the average improvement in accuracy when features are used in a split with a slight increase in computational power required. As a result, giving a more accurate result of how much each feature contributes to the model's prediction overall, and which ones the model relies on more.

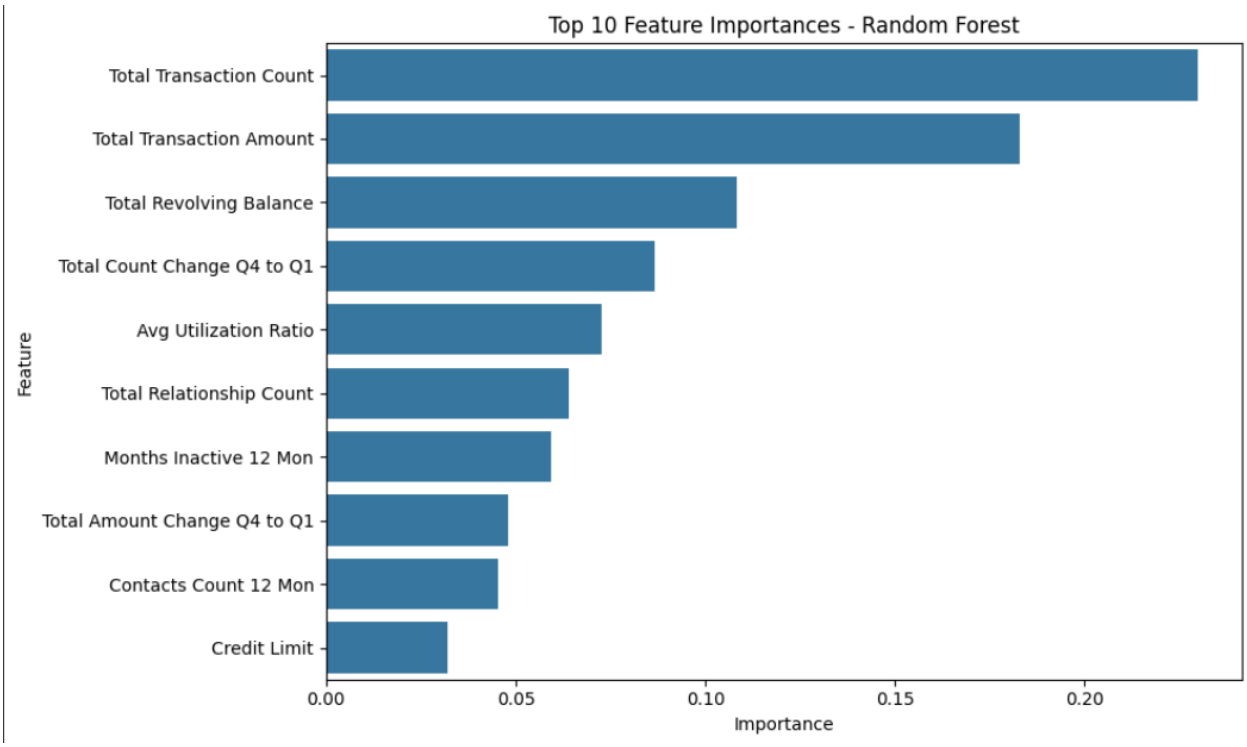

Fig 4.X.1.1 Top 10 Feature Importance Graph (Random Forest)

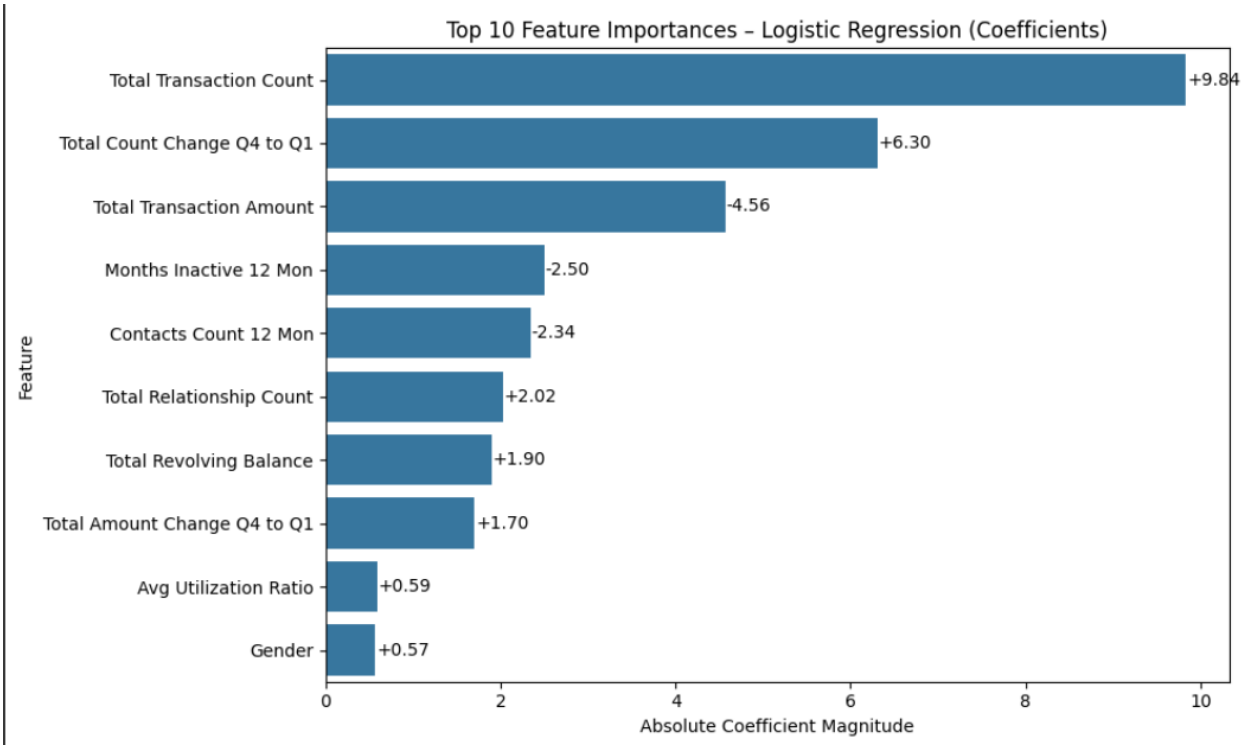

Fig 4.2.1.1 Top 10 Feature Importance Graph (Logistic Regression)

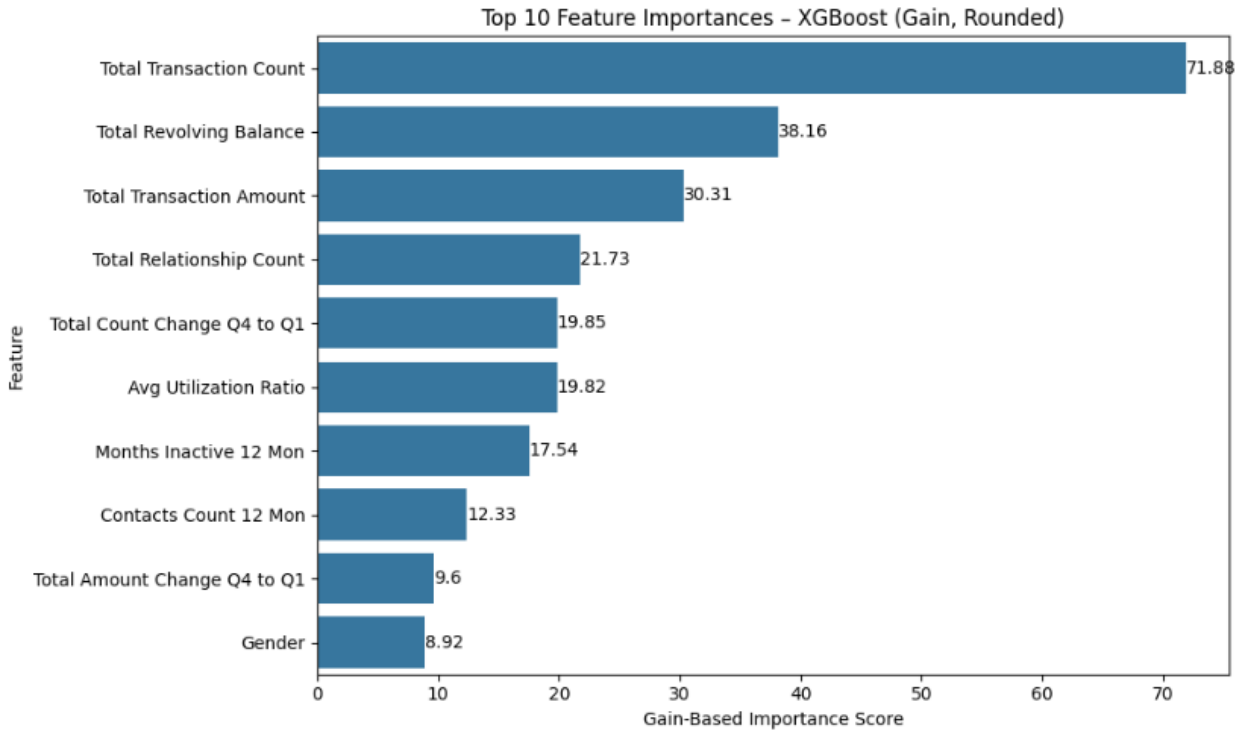

Fig 4.2.1.2 Top 10 Feature Importance Graph (XGBoost)

Through the runs we did with our models (XGBoost and Random Forest) and according to the key features that consistently ranked the highest are: Total Transaction Count, ranked 1# across all models, Total Transaction Amount and Total Revolving Balance, ranking top 3 for all models.

Other features that show up frequently are Utilization Ratio, Count Change from Q4 to Q1 and Months Inactive in the last 12 Months, all of which shows behavioral patterns that can be related to declining usage as a customer for the bank, therefore leading to churning. With that said, these features indirectly connect to reduced financial interactions, which is our top 3 features of importance.

Overall, it can be presumed that churning happens when reduced transactional activity and engagement from the customer are present. Therefore, serving as an indicator for the bank to look out for.

4.2.2 Receiver Operating Characteristic Curve (ROC Curve)

We did ROC curve; however, it shows minimal differences between each variation of the model across the 3 model that we have tested, we have included the graphs for proof:

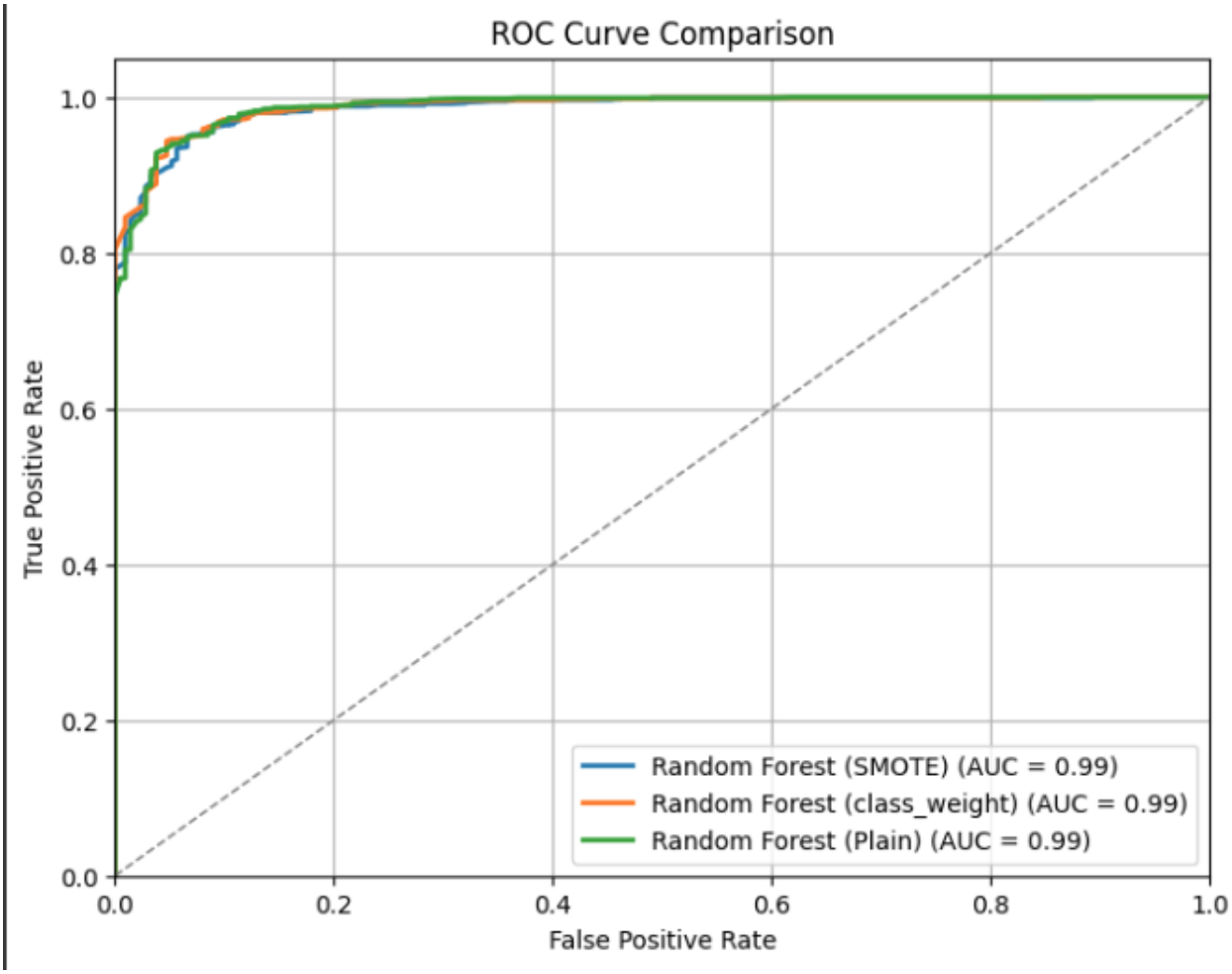

Fig. 4.2.2.1 ROC curve of Random Forest Classifier

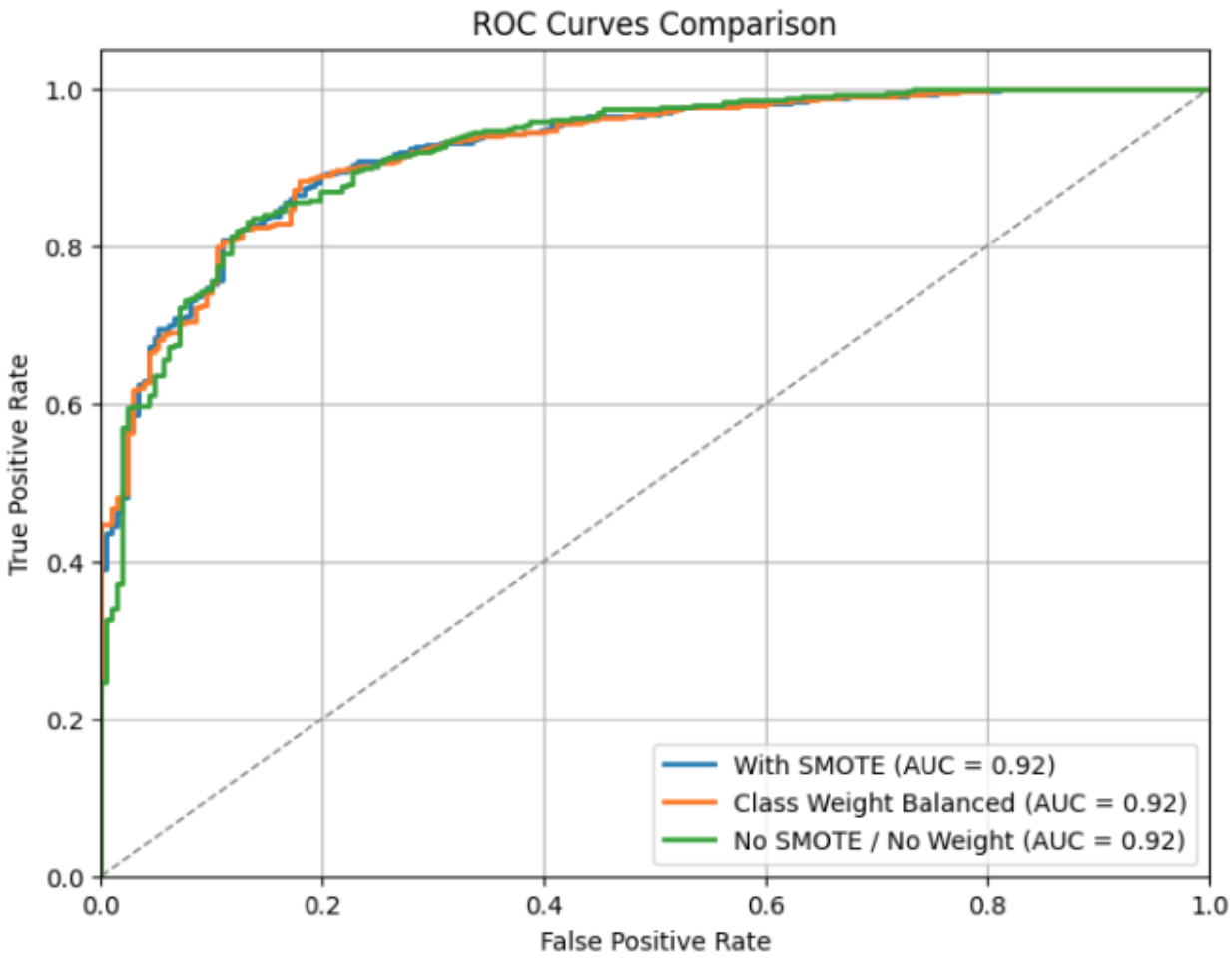

Fig. 4.2.2.2 ROC curve of Logistic Regression

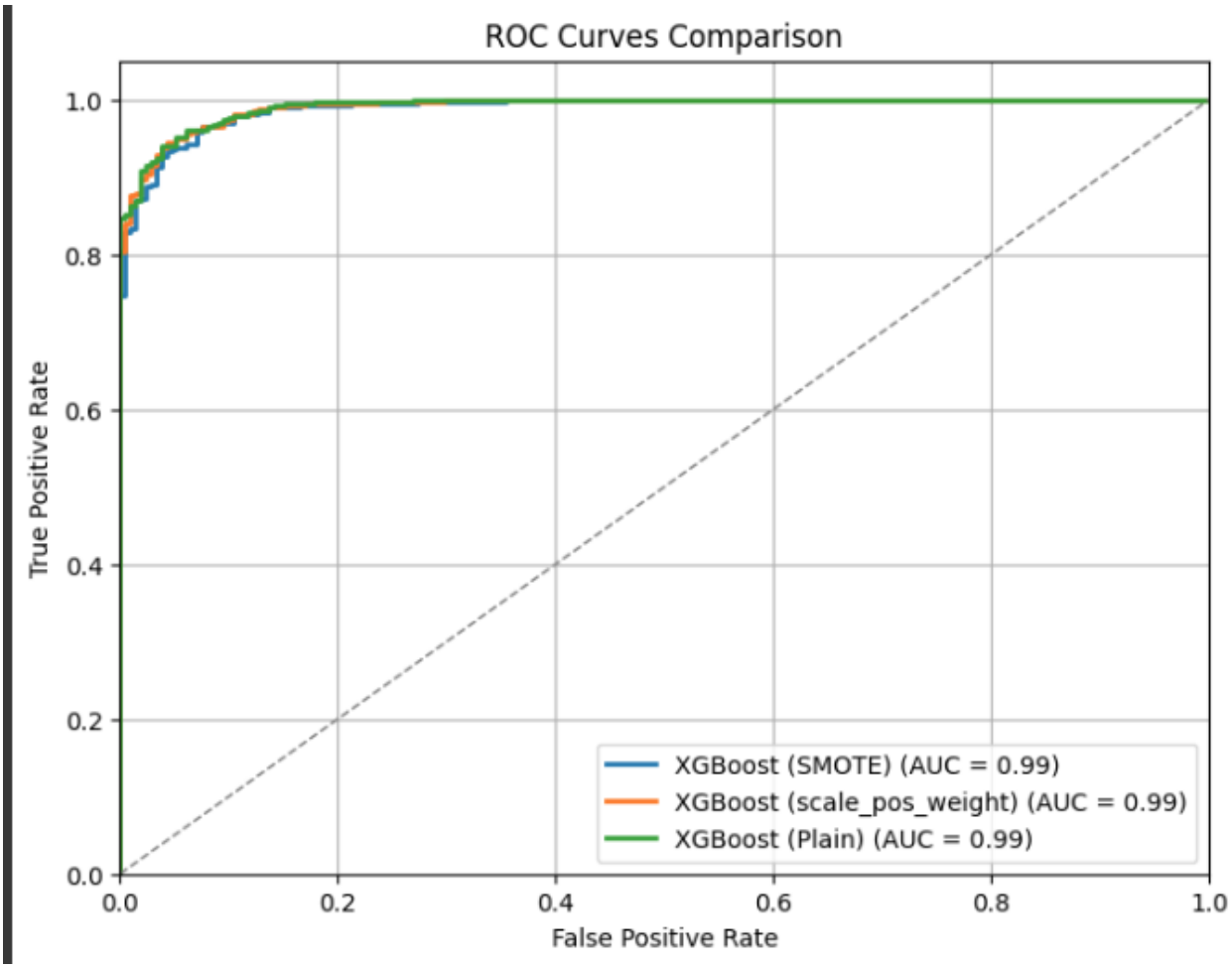

Fig. 4.2.2.3 ROC curve of XGBoost

Although there are no significant differences from the variations of the same model, XGBoost and Random Forest perform better than Logistic Regression with a difference of 0.07 in AUC.

4.2.3 Confusion Matrices

With the yy Matrices, we will be able to see how each model performs much more easily, as we are looking for the least false positive (avoid flagging loyal customers) and False Negative (able to catch more churners).

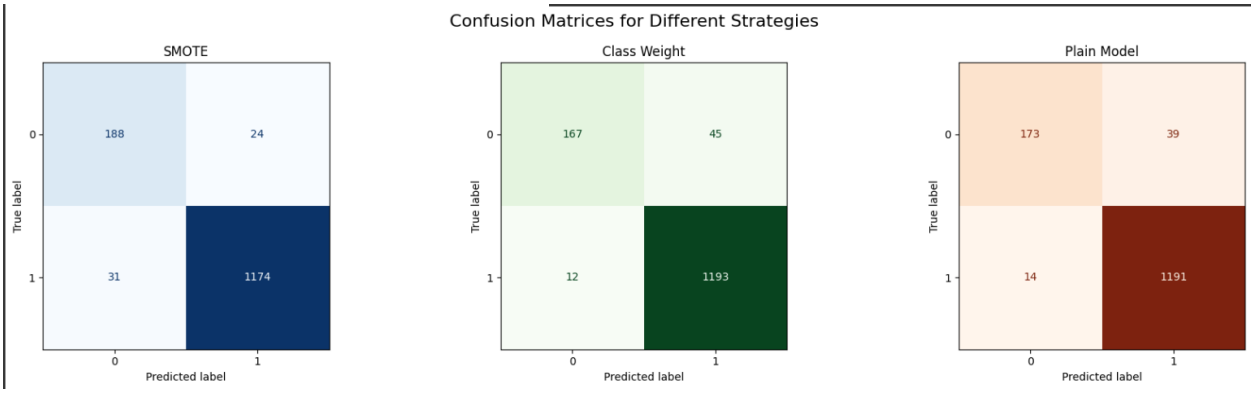

Fig. 4.2.3.1 Confusion Matrices of Random Forest Classifier

For Random Forest, The SMOTE model was able to catch 188 Churners as the most effective model, but misclassify 31 customers as churners, this shows that SMOTE is more sensitive, but at the cost of flagging non problem customers.

Opposingly, the Class Weighed Model was able to avoid the most false positives at 12 but missed 45 churners, Plain can identify 173 churners and not flag as many loyal customers at 14 false positives, same as the previous model. Overall, the best model here is SMOTE, as banks prioritize a higher churner detection rate over lower false positives rate.

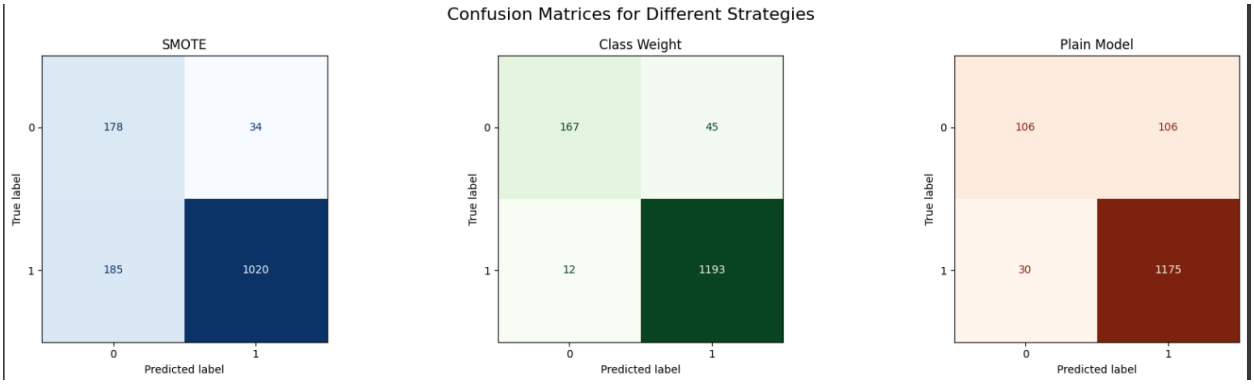

Fig. 4.2.3.2 Confusion Matrices of Logistic Regression

For Logistic Regression, The SMOTE model here was able to identify the most amount of churners with 178 churners, but the cost was 185 loyal customers being flagged, which is not acceptable when more loyal customers are flagged than actual churners.

On the other hand, the class weighted model was able to identify 167 churners with 12 false positive, which is the same statistic as the previous class weight model.

Lastly, the plain model performed the worst in terms of identifying churners, at 106 true negative and 106 false negative. This is bad for banks as it may allow many real churners to pass undetected by the system.

Overall, Class Weight was the best choice here for the most balanced performance, keeping churner detection high and false positives low.

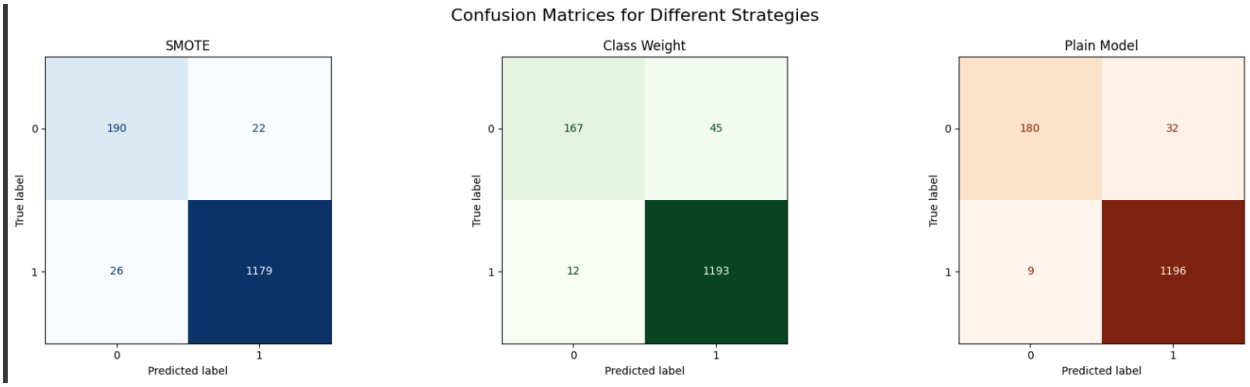

Fig. 4.2.3.3 Confusion Matrices of XGBoost

For XGBoost, the SMOTE model performed the best again with 190 churners identified, of the 212 churners, and only misclassifying 22 churners, which is the highest recall for class 0 out of every other model and versions of this model. It does have 26 false positives which means some customers were falsely classified.

For Class weight, it performed the same as the previous model, however, it performed worst out of the 3 versions of XGBoost.

For plain, it is still able to identify 180 churners while only 9 false positives, meaning it has the highest overall accuracy, which can lead to better generalization without needing more imbalancing handling.

Overall, Plain has the highest accuracy across all models and versions, but XGBoost with SMOTE still can identify more churners, which banks may prefer over having false positives as identifying more churners and retaining them will result in retaining more revenue than just falsely flagging a loyal customer.

4.2.4 Results from Stratified K Fold with Cross Validation

4.2.4.1 Metric Selection

Traditional criteria like total accuracy were deemed insufficient due to the class imbalance. A model that consistently predicts the majority class ("Existing Customer") would be highly accurate, but it would be useless for detecting churners, which are the focus of a commercial setting.

Thus, we concentrated on the following important metrics:

- **Recall (Sensitivity):** The percentage of real churners who were accurately recognized is measured by recall (sensitivity). To reduce false negatives, customers who are likely to depart but are overlooked by the model, high recall is essential.
- **F1-Score:** The F1-Score is a representation of accuracy and recall harmonic mean. When the cost of false positives and false negatives fluctuates, this statistic offers a fair assessment of the model's performance.
- **Recall for Class 0 (Non-Churn):** Although the primary objective is to detect churners, false labeling devoted clients may also result in needless retention initiatives. Thus, to guarantee balanced performance, recall for the majority class was also monitored.

These metrics gave us a thorough framework for review, allowing us to identify models that not only performed well overall but also successfully identified attrition situations.

Results derived from using Stratified K Fold with cross validation to confirm the accuracy, recall and F1 score are consistent and not lucky splits and confirm that it is not overfitted.

The Classification Report in the code will show similar results (0.001 difference without rounding), therefore, to reduce redundancy, it will not be included in this report.

4.2.4.2 Random Forest Classifier

Metric	RF - Plain	RF - Class Weight	RF - SMOTE
Test Accuracy	0.960174	0.956361	0.957350
Test F1 Macro	0.920983	0.911535	0.920326
Test Recall (Class 0)	0.813994	0.778071	**0.875999**

Test Recall (Class 1)	0.987433	0.989611	0.972519
Test F1 (Class 0)	0.865335	0.848565	0.866015

Fig 4.2.4.1.1 Random Forest Classifier Results from Stratified K Fold with Cross Validation

Result shows it is not overfitting data, and with the use of smote increase of 7.6% recall for class 0 gain over plain. slight trade off for class 1 recall from 0.99 to 0.97. Plain perform better in Class 1, making it a better choice for overall accuracy.

In summary among the three Random Forest configurations tested, the version trained with SMOTE achieved the highest recall and F1-score for the minority class of class 0 (churners), making it the most suitable choice for this churn prediction task, where identifying at-risk customers is a primary goal.

4.2.4.3 Logistic Regression

Metric	LogReg - Plain	LogReg - Class Weight	LogReg - SMOTE
Test Accuracy	0.898884	0.834203	0.840417
Test F1 Macro	0.784496	0.752415	0.758202
Test Recall (Class 0)	0.541757	**0.824829**	0.818543
Test Recall (Class 1)	**0.965480**	0.835956	0.844502
Test F1 (Class 0)	0.627509	0.610132	0.617224

Fig 4.2.4.2.1 Logistic Regression Results from Stratified K Fold with Cross Validation

From the result, we can see that Class Weighting has a significantly higher recall at 52% for class 0 over plain but does worst with 13% (from 0.965 to 0.836) lower recall for class 1.

Result shows SMOTE does not provide any significant improvement, where class 0 prediction quality (f1) is low even with SMOTE where it is on par with Class Weighting, Random Forest performs better overall compared to Logistic Regression except for plain Random Forest's class 0 recall at 0.814. To catch the greatest number of churners, class weight serves as the best option for logistic regressions.

4.2.4.4 XGBoost

Metric	XGB - Plain	XGB - Weighted	XGB - SMOTE
Test Accuracy	**0.968789**	0.965541	0.966812
Test F1 Macro	**0.939876**	0.936334	0.937323
Test Recall (Class 0)	0.875098	**0.916422**	0.893952
Test Recall (Class 1)	**0.986260**	0.974699	0.980396
Test F1 (Class 0)	**0.898184**	0.893213	0.894332

Fig 4.2.4.3.1 XGBoost Results from Stratified K Fold with Cross Validation

Plain XGB is most accurate in general and F1 score, however, only by <0.5%, SMOTE has virtually no difference when compared to Weighted, but does perform slightly better than Plain at class 0 recall (2.2% increase) but suffer slightly in performance in every other category, it is also

worst at class 0 recall than Weighted (2.5% decrease). Weighted XGB is the best option here due to it having a 4.7% increase in recall over plain XGB.

In terms of ranking the models of their ability to recall Class 0 (Churners) Weighted XGBoost rank first at 0.916, followed by Random Forest with SMOTE at 0.876 then lastly Logistic Regression Class Weighted at 0.825.

5.0 Conclusion

The goal of this project was to identify customer churn behavioral indicators, test their predictive validity, and develop machine learning models that achieve a minimum of 80% accuracy in identifying customers at risk of churn.

According to the results of our predictive modelling, the models showed that there lie distinct behavioral patterns when variables are combined. This aligns with our initial dataset overview that demonstrated that not one feature had a strong correlation with churn alone. For all of the models, the common significant features were Total Revolving Balance, Total Transaction Amount, and Total Transaction Count. These features (along with Utilisation Ratio, Change in Transactions Count (Q4 to Q1), Months Inactive in the Last 12 Months, and Number Contacts with Bank), consistently demonstrated a pattern indicating that less engagement and lower financial activity are likely to predict churn. For example, customers who are younger, have low transaction frequency, and have not contacted the bank recently are more likely to attrit a profile that banks should monitor closely.

We tested three models: Logistic Regression, Random Forest, and XGBoost, under different class imbalance handling methods. Although XGBoost with SMOTE performed well in confusion matrices across many of the algorithms by capturing the most churners overall, it lacked generalizability when using Stratified K-Fold Cross Validation for repeated data splits. XGBoost with class weighting was the most balanced and reliable model that consistently achieved strong recall for churners while retaining consistent accuracy, which was above 80% and this and counts of false positives were still low across all of the data splits we performed. Overall we confidently recommend class weighted XGBoost as the model we would most rely upon in practice because of the results and relative stability demonstrated in our analysis.

To support banks in translating these findings into practice, we suggest the following strategies:

Early Warning Systems: Set up dashboards for real-time monitoring that watch the key churn indicators, such as decreases in transactions, increases in inactivity, or decreases in revolving balances. Set alerts and thresholds to catch potential at-risk profiles early (Dalbah, Ali, Al-Naymat, 2022).

Campaigns for Segmented Retention: Carry out retention campaigns based on behavioural segments, especially for inactive but once valuable clients the re-engagement could be better received through loyalty rewards or financial check-ins (Tariq, 2020). Alternatively, for younger customers who's not as engaged, the banks could re-engaged them through educational content such as a refresher on their account or tutorials on digital banking.

Proactive Personal Engagement: Banks should implement automated but personalised nudges (via SMS, email or engagement app notifications) for customer re-engagement before the churn behavior is fully carried out, as by the point it would be too late. Re-engagement nudges can trigger specific messages to customers that haven't engaged after two months of inactivity or are showing decreased spending activity from their historical baseline levels (Ashrafuzzaman, et al., 2025).

In conclusion, this project not only achieved its primary goals of identifying important churn indicators and attaining high forecast accuracy, but it also offered useful information that banks can utilise to create customer-focused, data-driven retention strategies.

7.0 References

Ashrafuzzaman, M., Parveen, R., Sumiya, M. A., Rahman, A. (2025) AI-Powered Personalization In Digital Banking: A Review Of Customer Behavior Analytics And Engagement. *American Journal of Interdisciplinary Studies* [online]. 6(1), pp. 40-71. [Accessed 2 July 2025].

Brownlee, J. (2021) SMOTE for Imbalanced Classification with Python, *Machine Learning Mastery*, 17 March [online]. Available from: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ [Accessed 25 June 2025].

Dalbah, L. M., Ali, S., Al-Naymat, G. (2022) An Interactive Dashboard for Predicting Bank Customer Attrition. I*nternational Conference on Emerging Trends in Computing and Engineering Applications (ETCEA)* [online]. [Accessed 2 July 2025].

Hayes, A. (2024) Descriptive Statistics: Definition, Overview, Types, and Examples, *Investopedia*, 27 June [online]. Available from: https://www.investopedia.com/terms/d/descriptive_statistics.asp [Accessed 2 July 2025].

HubSpot (2023). Customer acquisition vs. retention: Why retention is more important than ever, *HubSpot* [online]. Available from: https://blog.hubspot.com/service/customer-acquisition-vs-retention [Accessed 5 May 2025].

Olamendy, J. C. (2024) A Comprehensive Guide to Stratified K-Fold Cross-validation for Unbalanced Data, *Medium*, 2 April [online]. Available from: https://medium.com/@juanc.olamendy/a-comprehensive-guide-to-stratified-k-fold-cross-validation-for-unbalanced-data-014691060f17 [Accessed 29 June 2025].

Tariq, H. (2020) How To Segment And Retain Your Customers, *Forbes*, 31 March [online]. Available from: https://www.forbes.com/councils/forbescommunicationscouncil/2020/03/31/how-to-segment-and-retain-your-customers/ [Accessed 2 July 2025].

Fatima, S., Hussain, A., Amir, S., Ahmed, S. H. (2023) XGBoost and Random Forest Algorithms: An in Depth Analysis. *Pakistan Journal of Scientific Research* [online]. 3(1), pp. 26-31. [Accessed 25 June 2025].

Kleimann, S. G. A. (2020) Logistic Regression for Binary Classification, *Toward Data Science*, 25 September [online]. Available from: https://towardsdatascience.com/logistic-regression-for-binary-classification-56a2402e62e6/ [Accessed 27 June 2025].