BUSINESS REPORT

Prepared by: - Durgesh Bhargava

PGP-DSBA (PGP DSBA.O.JULY24.A)

CONTENTS

List of Figures

LIST OF TABLES

PROJECT NOTE - I

Introduction of the business problem

Problem Statement: -

Shop4you, an e-commerce company, is facing intense competition from larger marketplaces and niche platforms. One of its key challenges is retaining existing customer accounts, many of which serve as shared family or business accounts. Account-level churn significantly affects overall customer volume and sales. The objective of this project is to build a data-driven churn prediction model that identifies high-risk accounts and enables the company to deploy personalized, cost-effective retention campaigns.

Need of the study/project

Churn directly impacts revenue and profitability. Given that a single Shop4you account may include multiple active customers, losing one account could mean a multi-fold loss in user base, order volume, and brand loyalty. As acquiring new customers is significantly more expensive than retaining existing ones, this study is essential to improving customer retention through smarter, targeted marketing. Additionally, Shop4you requires a balance between preventing churn and avoiding excessive promotional spending that could hurt profit margins

UNDERSTANDING BUSINESS/SOCIAL OPPORTUNITY

This a case study of an e-commerce company where in they have customers assigned with unique account ID and a single account ID can hold many customers (like family membership) across gender and marital status, customers get flexibility in terms of mode of payment they want to opt for. Customers are again segmented across various types of plans they opt for as per their usage which also based on the device they use (computer or mobile) moreover they ear cashbacks on bill payment.

The overall business runs in customers loyalty and stickiness which in-turn comes from providing quality and value-added services. Also, running various promotional and festivals offers may help organization in getting new customers and also retaining the old one.

This project provides SHOP4YOU with a valuable business opportunity:

- **Customer Rentition** : The model will help identify churn-prone accounts early, enabling to take preventive measures.
- **Market Campaign**: By segmenting customers based on churn risk and behavior, the marketing team can design targeted offers.
- **Long-Term Growth**: Retaining loyal customers leads to increased lifetime value, word-of-mouth promotion, and better forecasting.
- **Social Value**: By offering meaningful, non-intrusive incentives, Shop4you can enhance customer satisfaction and trust without resorting to unsustainable discounts.

Data Report

Variable	Description
AccountID	account unique identifier
Churn	account churn flag (Target)
Tenure	Tenure of account
City_Tier	Tier of primary customer's city
CC_Contacted_L12m	How many times all the customers of the account has contacted customer care in last 12months
Payment	Preferred Payment mode of the customers in the account
Gender	Gender of the primary customer of the account
Service_Score	Satisfaction score given by customers of the account on service provided by company
Account_user_count	Number of customers tagged with this account
account_segment	Account segmentation on the basis of spend
CC_Agent_Score	Satisfaction score given by customers of the account on customer care service provided by company
Marital_Status	Marital status of the primary customer of the account
rev_per_month	Monthly average revenue generated by account in last 12 months
Complain_l12m	Any complaints has been raised by account in last 12 months
rev_growth_yoy	revenue growth percentage of the account (last 12 months vs last 24 to 13 month)
coupon_used_l12m	How many times customers have used coupons to do the payment in last 12 months
Day_Since_CC_connect	Number of days since no customers in the account has contacted the customer care
cashback_l12m	Monthly average cashback generated by account in last 12 months
Login_device	Preferred login device of the customers in the account

Data Ingestion:

Imported the necessary packages, configured the working directory, and loaded the dataset. The dataset comprises 11,260 observations and 19 variables, including 18 independent features and 1 dependent (target) variable.

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Status	rev_
0	20000	1	4	3.00	6.00	Debit Card	Female	3.00	3	Super	2.00	Single	
1	20001	1	0	1.00	8.00	UPI	Male	3.00	4	Regular Plus	3.00	Single	
2	20002	1	0	1.00	30.00	Debit Card	Male	2.00	4	Regular Plus	3.00	Single	
3	20003	1	0	3.00	15.00	Debit Card	Male	2.00	4	Super	5.00	Single	
4	20004	1	0	1.00	12.00	Credit Card	Male	2.00	3	Regular Plus	5.00	Single	

Table 1 – Preview of DataFrame (Top 5 Rows)

Understanding how data was collected in terms of time, frequency and methodology

The dataset consists of information collected from 11,260 unique account IDs, spanning various genders and marital statuses.

- Based on variables such as “CC_Contacted_L12m”, “rev_per_month”, “Complain_l12m”, “rev_growth_yoy”, “coupon_used_l12m”, “Day_Since_CC_connect”, and “cashback_l12m”, it is evident that the data captures customer behavior over the past 12 months.
- The dataset includes a total of 19 variables, comprising 18 independent features and 1 dependent (target) variable that indicates whether a customer has churned.
- It integrates customer service usage details, payment preferences, and basic demographic information.
- The dataset contains a mix of categorical and continuous variables.

Visual inspection of data (rows, columns, descriptive details)

The dataset comprises 11,260 records and 19 variables.

```
Rows: 11260, Columns: 19
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11260 entries, 0 to 11259
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   AccountID                11260 non-null  int64
 1   Churn                    11260 non-null  int64
 2   Tenure                   11158 non-null  object
 3   City_Tier                11148 non-null  float64
 4   CC_Contacted_LY          11158 non-null  float64
 5   Payment                  11151 non-null  object
 6   Gender                   11152 non-null  object
 7   Service_Score            11162 non-null  float64
 8   Account_user_count       11148 non-null  object
 9   account_segment          11163 non-null  object
 10  CC_Agent_Score           11144 non-null  float64
 11  Marital_Status           11048 non-null  object
 12  rev_per_month            11158 non-null  object
 13  Complain_ly              10903 non-null  float64
 14  rev_growth_yoy           11260 non-null  object
 15  coupon_used_for_payment  11260 non-null  object
 16  Day_Since_CC_connect     10903 non-null  object
 17  cashback                 10789 non-null  object
 18  Login_device             11039 non-null  object
dtypes: float64(5), int64(2), object(12)
memory usage: 1.6+ MB
```

Table 2: Dataset Information (Summary of DataFrame)

Describing Data:
This section presents the statistical summary of the variables, highlighting the variation in key measurements and confirming the uniqueness and diversity of each variable.

	AccountID	Churn	City_Tier	CC_Contacted_LY	Service_Score	CC_Agent_Score	Complain_ly
count	11260.000	11260.000	11148.000	11158.000	11162.000	11144.000	10903.000
mean	25629.500	0.168	1.654	17.867	2.903	3.066	0.285
std	3250.626	0.374	0.915	8.853	0.726	1.380	0.452
min	20000.000	0.000	1.000	4.000	0.000	1.000	0.000
25%	22814.750	0.000	1.000	11.000	2.000	2.000	0.000
50%	25629.500	0.000	1.000	16.000	3.000	3.000	0.000
75%	28444.250	0.000	3.000	23.000	3.000	4.000	1.000
max	31259.000	1.000	3.000	132.000	5.000	5.000	1.000

Table 3: Descriptive Statistics of the Dataset

- All variables except **"AccountID"**, **"Churn"**, **"rev_growth_yoy"**, and **"coupon_used_for_payment"** contain some missing (null) values.

```
AccountID                0
Churn                    0
Tenure                 102
City_Tier              112
CC_Contacted_LY        102
Payment                109
Gender                 108
Service_Score           98
Account_user_count     112
account_segment         97
CC_Agent_Score         116
Marital_Status         212
rev_per_month          102
Complain_ly            357
rev_growth_yoy           0
coupon_used_for_payment  0
Day_Since_CC_connect   357
cashback               471
Login_device           221
dtype: int64
```

Table 4: - Showing Null Values in Dataset

- The dataset contains **no duplicate observations**.

Understanding of Attributes (Variable Information and Renaming)

This project includes **18 attributes** contributing to the target variable. Below is a detailed description of each variable:

1. **AccountID** – A unique integer identifier for each customer. There are no missing values in this variable.
2. **Churn** – The target categorical variable indicating whether a customer has churned. It has no missing values, where "0" means **No** and "1" means **Yes**.
3. **Tenure** – Continuous variable representing the total duration the account has been active. It contains 102 missing values.
4. **City_Tier** – Categorical variable that classifies customers into three tiers based on their primary city of residence. It has 112 missing values.

5. **CC_Contacted_L12m** – Continuous variable denoting the number of times customers contacted customer care in the last 12 months. There are 102 missing values.
6. **Payment** – Categorical variable representing the preferred mode of bill payment. It has 109 missing values.
7. **Gender** – Categorical variable indicating the gender of the primary account holder. There are 108 missing values.
8. **Service_Score** – Categorical scores provided by customers evaluating the service quality. This variable has 98 missing values.
9. **Account_user_count** – Continuous variable indicating the number of customers linked to an AccountID. It contains 112 missing values.
10. **account_segment** – Categorical variable segmenting customers based on spending and revenue generation. It has 97 missing values.
11. **CC_Agent_Score** – Categorical scores reflecting customer ratings for the customer care representative. There are 116 missing values.
12. **Marital_Status** – Categorical variable indicating the marital status of the primary account holder, with 212 missing values.
13. **rev_per_month** – Continuous variable representing average revenue per account in the last 12 months. It contains 102 missing values.
14. **Complain_l12m** – Categorical variable denoting whether the customer raised complaints in the last 12 months. There are 357 missing values.
15. **rev_growth_yoy** – Continuous variable showing the year-over-year revenue growth percentage (12 months vs 24 to 13 months). No missing values are present.
16. **coupon_used_l12m** – Continuous variable indicating the number of times customers used discount coupons for bill payment in the last 12 months. No missing values.
17. **Day_Since_CC_connect** – Continuous variable representing the number of days since the last customer care contact. Higher values indicate better service. It has 357 missing values.
18. **cashback_l12m** – Continuous variable showing the cashback amount earned by the customer during bill payments. There are 471 missing values.
19. **Login_device** – Categorical variable indicating the device used by the customer to access services (phone or computer). It contains 221 missing values.

- Based on this review, **no variable renaming is necessary**.
- With this understanding of the dataset, we will proceed to the Exploratory Data Analysis (EDA) phase to better explore the data, handle missing values, and treat outliers.

Data Cleaning and Preparation Steps Before EDA

Special Character Treatment:

During initial data cleaning, certain columns were found to contain invalid special characters such as #, $, @, +, and *. These characters can cause issues in numerical calculations and data analysis. To address this, all occurrences of these special characters were identified and replaced with missing values (null or NaN). This ensures that the dataset remains clean and consistent for subsequent analysis steps like null value treatment and modelling.

Null Value Treatment:

AccountID	0
Churn	0
Tenure	218
City_Tier	112
CC_Contacted_LY	102
Payment	109
Gender	108
Service_Score	98
Account_user_count	444
account_segment	97
CC_Agent_Score	116
Marital_Status	212
rev_per_month	791
Complain_ly	357
rev_growth_yoy	3
coupon_used_for_payment	3
Day_Since_CC_connect	358
cashback	473
Login_device	221
dtype: int64	

Churn	0
Tenure	0
City_Tier	0
CC_Contacted_LY	0
Payment	0
Gender	0
Service_Score	0
Account_user_count	0
account_segment	0
CC_Agent_Score	0
Marital_Status	0
rev_per_month	0
Complain_ly	0
rev_growth_yoy	0
coupon_used_for_payment	0
Day_Since_CC_connect	0
cashback	0
Login_device	0
dtype: int64	

Table 5. Missing Values (Before and after)

To handle these missing values effectively, K-Nearest Neighbours (KNN) imputation was employed. This technique estimates the missing values based on the similarity of other observations in the dataset, using five nearest neighbours as reference points.

Categorical variables were first encoded to numeric labels to allow the imputation algorithm to process them correctly. After imputation, the encoded categorical variables were transformed back to their original categories.

As a result of this treatment, all missing values across the dataset were successfully imputed, leaving no null values in any of the variables. This ensures the dataset is complete and ready for further analysis, such as exploratory data analysis and modelling.

Data Standardization:

During the initial data review, it was observed that some categorical variables contained inconsistent representations for the same category. For example, the Gender column had values like "M" and "Male" used interchangeably for the same gender, and similar inconsistencies were found in the account_segment and Login_device variables.

To ensure consistency and improve data quality, these variations were standardized by replacing different representations with a single uniform label. For instance:

- "M" was standardized to "Male" and "F" to "Female".
- Variants like "Regular +" and "Super +" were standardized to "Regular Plus" and "Super Plus", respectively.
- Anomalous entries such as "&&&&" in the Login_device column were recoded to "Other".

This standardization step is crucial for reliable analysis, ensuring that categories are correctly grouped and interpreted.

```
Unique values in Gender are :
Gender
Male       6328
Female     4178
M           376
F           270
Name: count, dtype: int64

----------------------------------------

Unique values in account_segment are :
account_segment
Super            4062
Regular Plus     3862
HNI              1639
Super Plus        771
Regular           520
Regular +         262
Super +            47
Name: count, dtype: int64

----------------------------------------

Unique values in Login_device are :
Login_device
Mobile       7482
Computer     3018
&&&&          539
Name: count, dtype: int64
```

Before standardization

```
Unique values in Gender are :
Gender
Male       6704
Female     4448
Name: count, dtype: int64

----------------------------------------

Unique values in account_segment are :
account_segment
Regular Plus    4124
Super           4062
HNI             1639
Super Plus       818
Regular          520
Name: count, dtype: int64
----------------------------------------

Unique values in account_segment are :
Login_device
Mobile      7482
Computer    3018
Other        539
Name: count, dtype: int64
```

After standardization

Fig 1. Before and after standardization in dataset columns

Removal of unwanted variables

The '**AccountID**' column was removed from the dataset prior to performing exploratory data analysis, as it serves only as a unique identifier and does not provide any meaningful insights for analysis.

Exploratory Data Analysis

Before starting EDA, we performed an initial check for outliers, as they can affect analysis and visualizations. The actual treatment of outliers will be done post-EDA, once we better understand the data and its context.

	outlier %
Churn	16.840
Tenure	1.240
City_Tier	0.000
CC_Contacted_LY	0.370
Service_Score	0.120
Account_user_count	6.760
CC_Agent_Score	0.000
rev_per_month	1.960
Complain_ly	0.000
rev_growth_yoy	0.000
coupon_used_for_payment	12.260
Day_Since_CC_connect	1.150
cashback	8.180
Total_CC_Score	0.500
Avg_Coupon_Per_Month	9.130
Is_Long_Term_Customer	0.000
High_Cashback_User	24.990

Table 6. Outliers in dataset

None of the variables follow a normal distribution; most are skewed in nature

Kurtosis and skewness of dataset are as below:

	Kurtosis	Skewness
Churn	1.140	1.770
Tenure	23.610	3.910
City_Tier	-1.390	0.740
CC_Contacted_LY	8.280	1.430
Service_Score	-0.670	0.000
Account_user_count	0.630	-0.390
CC_Agent_Score	-1.110	-0.140
rev_per_month	91.210	9.260
Complain_ly	-1.050	0.980
rev_growth_yoy	-0.220	0.750
coupon_used_for_payment	9.100	2.580
Day_Since_CC_connect	5.470	1.280
cashback	83.370	8.860

Table 7. Kurtosis md Skewness in dataset

1. Univariate Analysis

- **Churn**

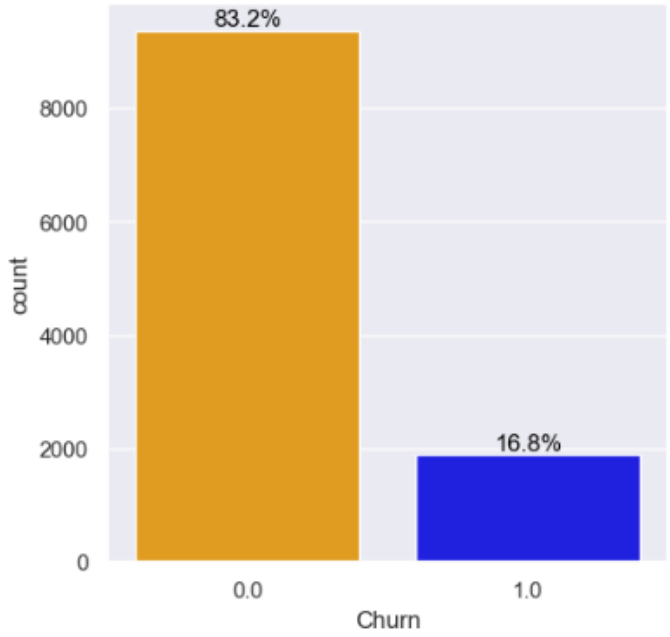


Fig 2. Bar plot of Churn segment

This chart reveals a significant class imbalance in your churn data, with 83.2% non-churners and only 16.8% churners, highlighting the critical need for techniques like SMOTE to prevent model bias.

- **Tenure**

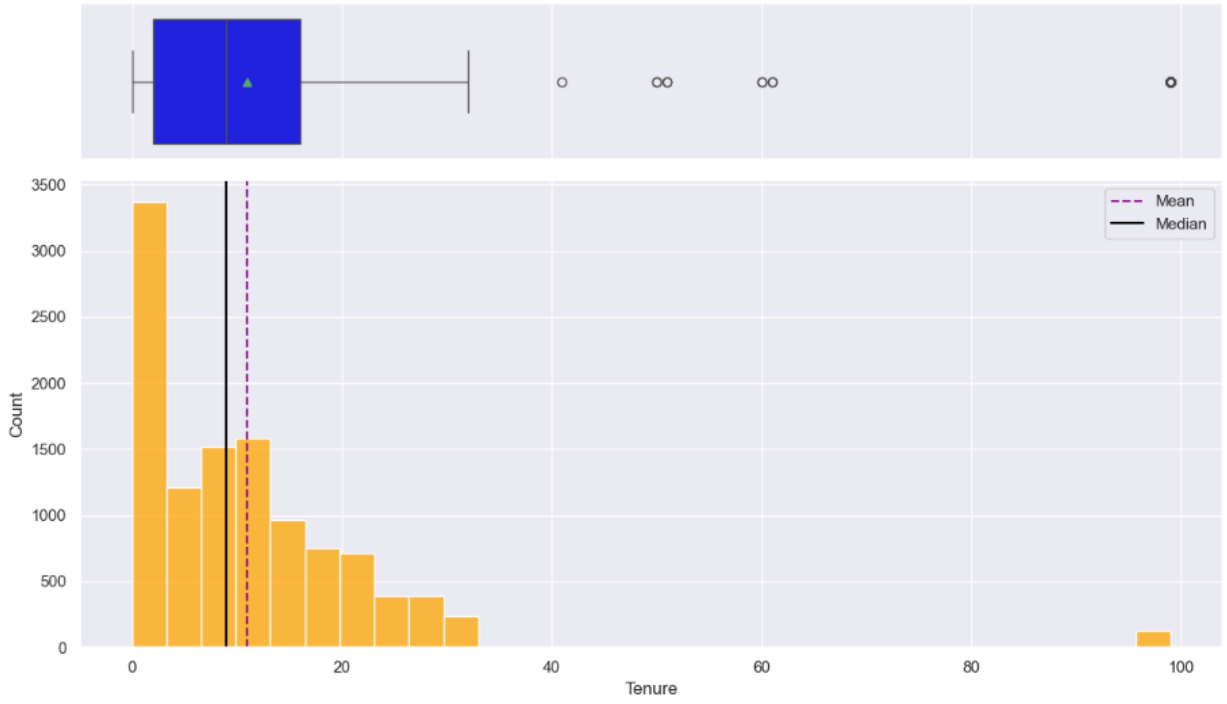


Fig 3. Hist-boxplot of Tenure

The 'Tenure' distribution shows the highest number of customers with less than a month's tenure, indicating a large new customer base, alongside several outliers clustered around 50, 65, and 100 months.

- **City Tier**

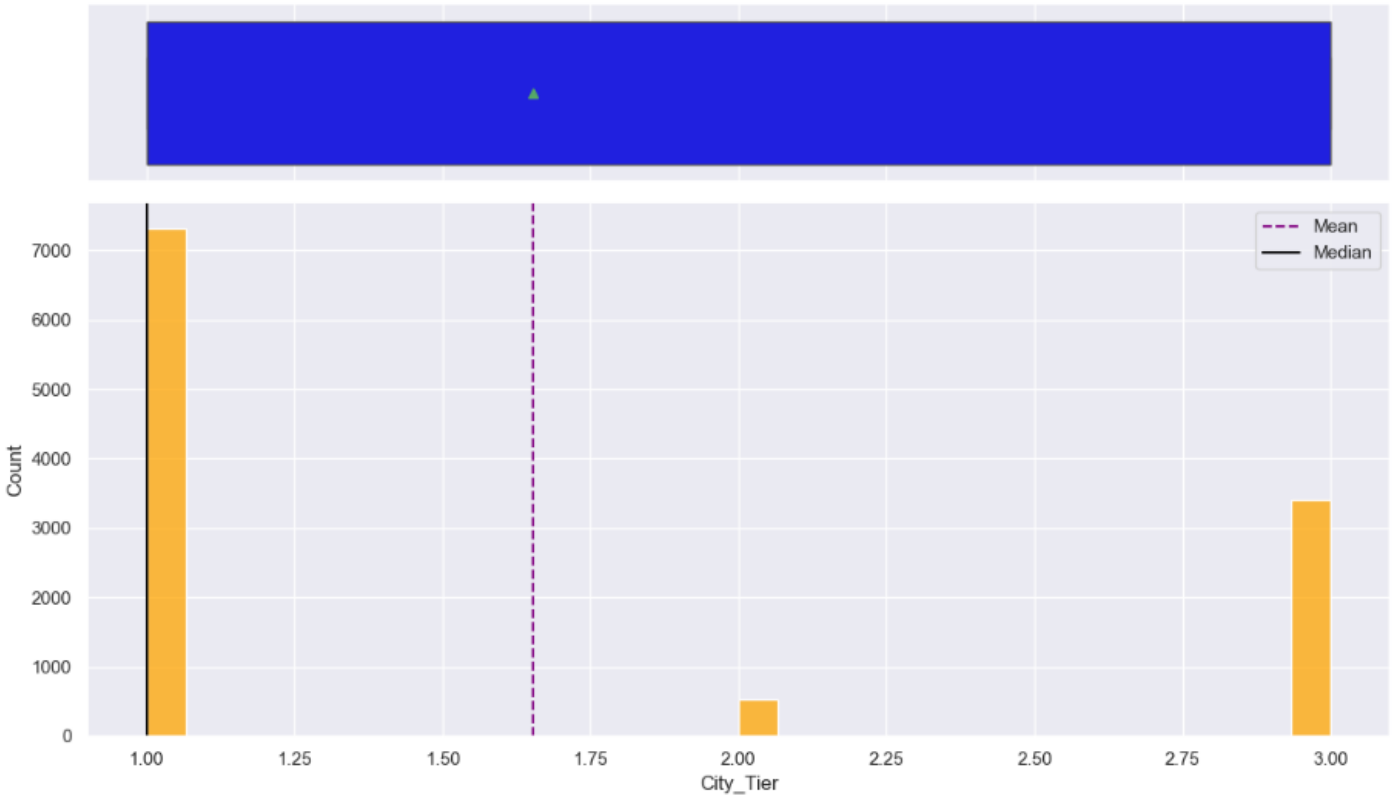


Fig 4. Hist-boxplot of City Tier

The 'City_Tier' distribution reveals that most customers originate from Tier 1 cities, followed by Tier 3, with a significantly smaller representation from Tier 2; this feature is effectively categorical.

- **Customer Care contact in the past year**

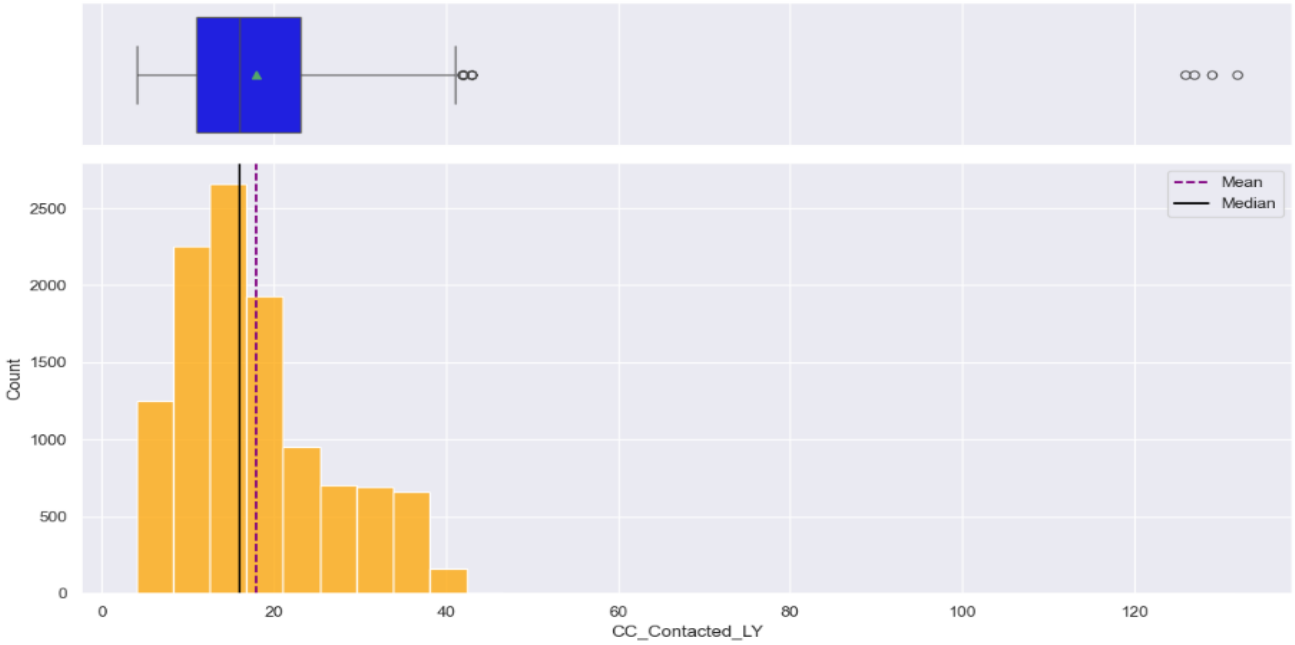


Fig 5..Hist-boxplot of Customer contact per year

The distribution indicates that most customers contacted customer care between 10 to 25 times. However, there are noticeable outliers around 45 contacts and some extreme cases exceeding 125 contacts.

- **Payment**

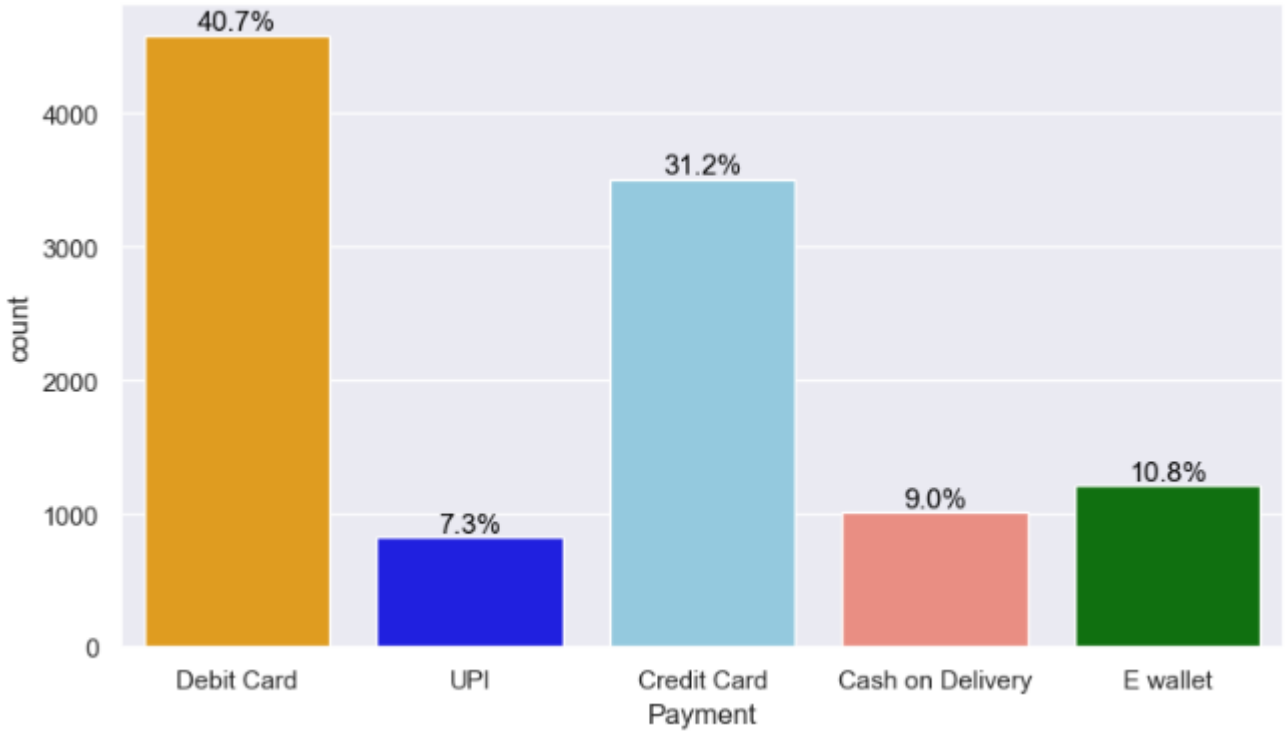


Fig 6. Bar plot of Payment method

Around 40.7% of customers prefer using Debit Cards for payments, followed by 31.2% who opt for Credit Cards. UPI (Unified Payment Interface) is the least preferred mode, likely due to its relatively recent introduction in the market.

- **Service Score**

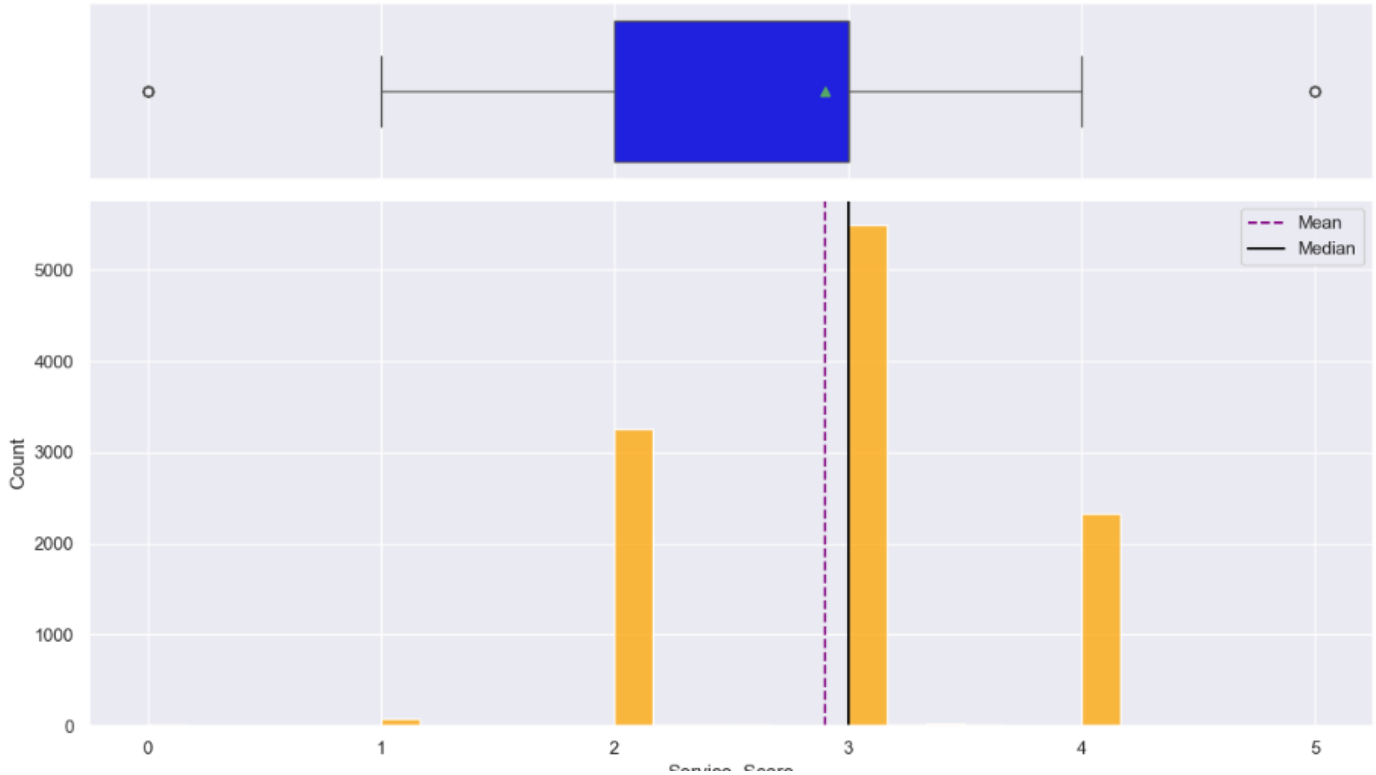


Fig 7. Hist-boxplot of Service score

The average service satisfaction score given by customers is 3, with a few outliers at 0 and 5. This indicates that overall customer satisfaction is moderate, and the presence of low scores highlights potential service issues that warrant further investigation.

- **Account User Count**

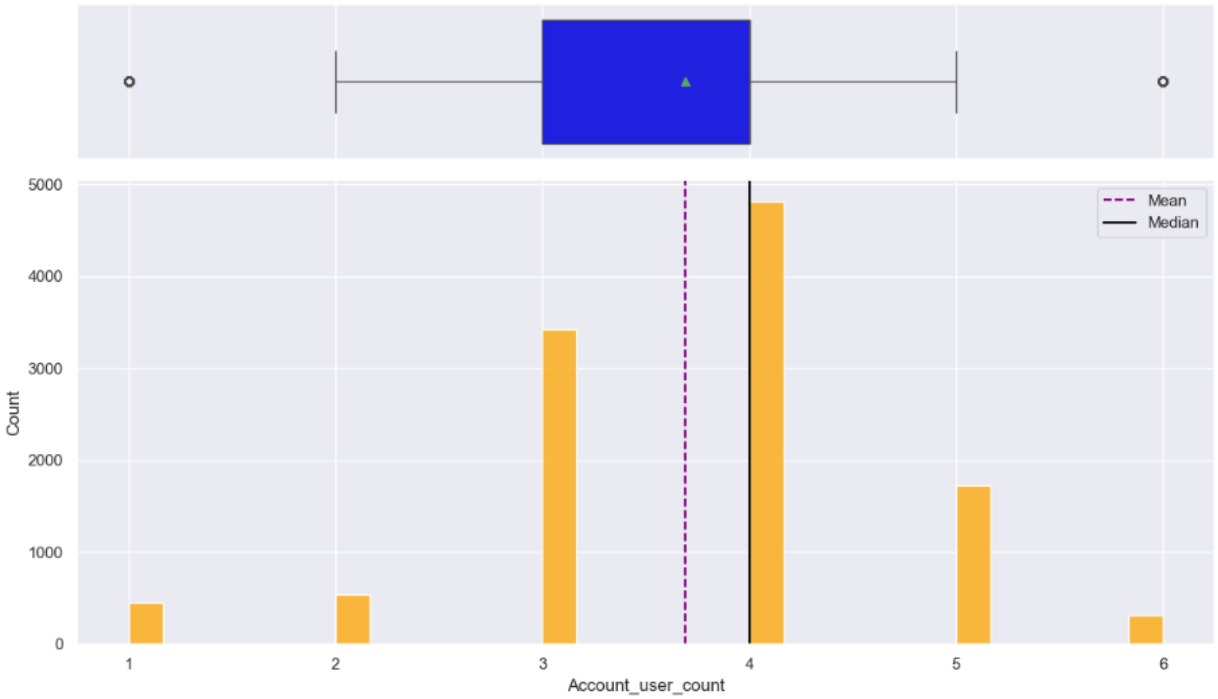


Fig 8. Hist-boxplot of number user per account

A significant number of accounts are associated with 4 users, followed by those with 3 and 5 users per account. Outliers are observed at 1 and 6 users. Given this distribution, Account_user_count may also be considered as a categorical variable for analysis.

- **Account Segment**

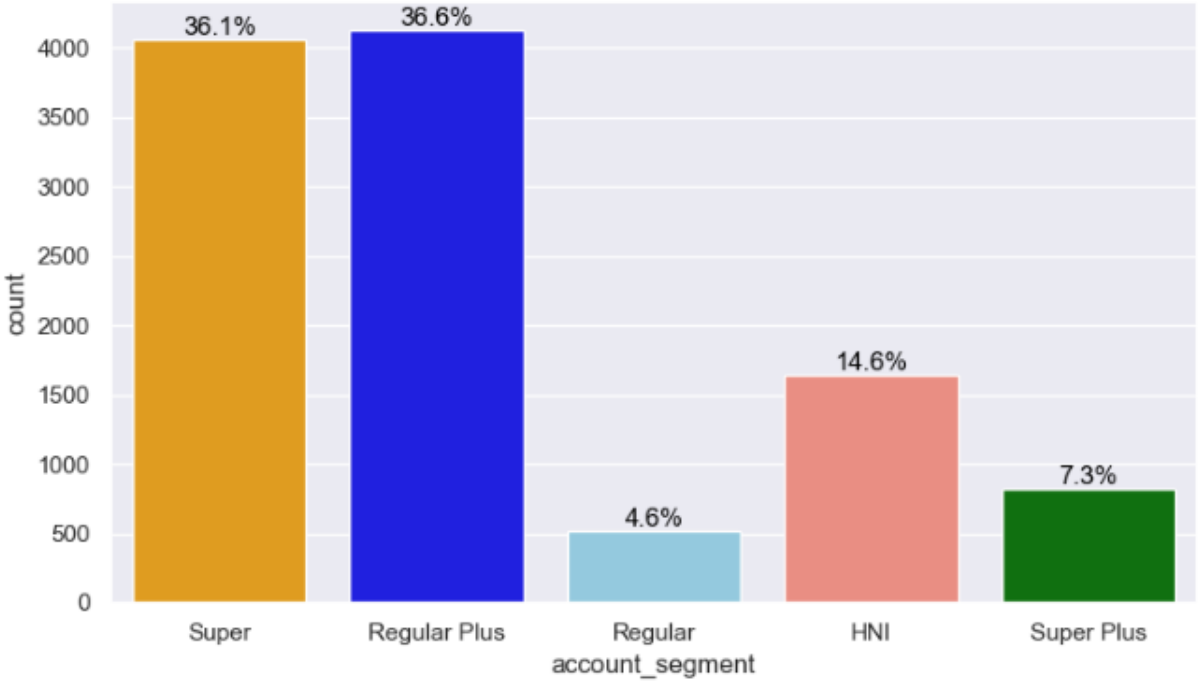


Fig 9. Bar plot of number Account segment type

The 'Regular Plus' and 'Super' segments together account for the largest share of customers, each contributing over 36%. In contrast, the 'Regular' segment has the smallest customer base at under 5%, while the High Net Income (HNI) segment represents approximately 14.6% of the total customers.

- **Customer Care Agent Score**

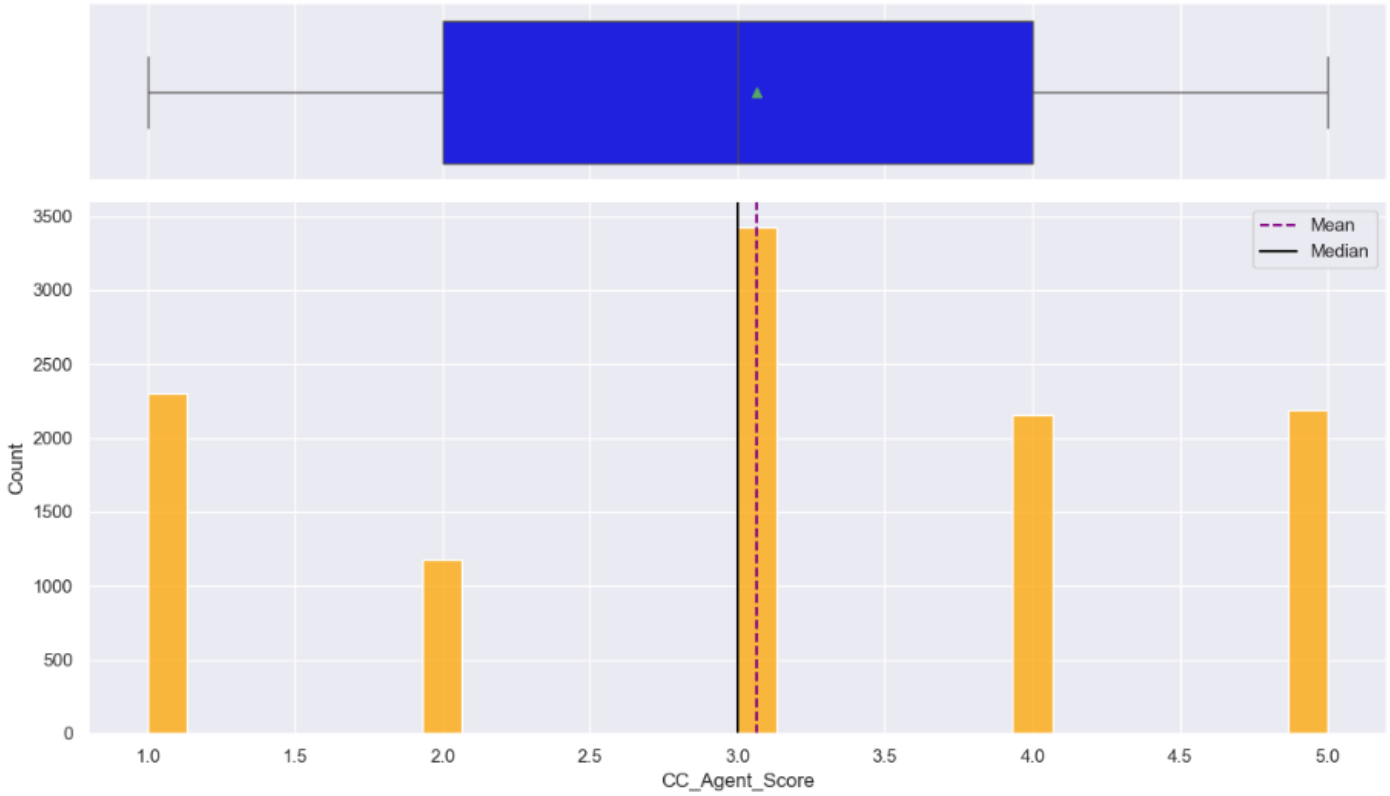


Fig 10. Hist-boxplot of Customer care agent score

On average, customers have rated the customer care service with a score of 3. Interestingly, a similar number of customers have given both high scores (4 and 5) and low scores (1), indicating a polarized perception. This suggests that the quality of customer service tends to be inconsistent, with nearly equal chances of being perceived as excellent or poor.

- **Marital Status**

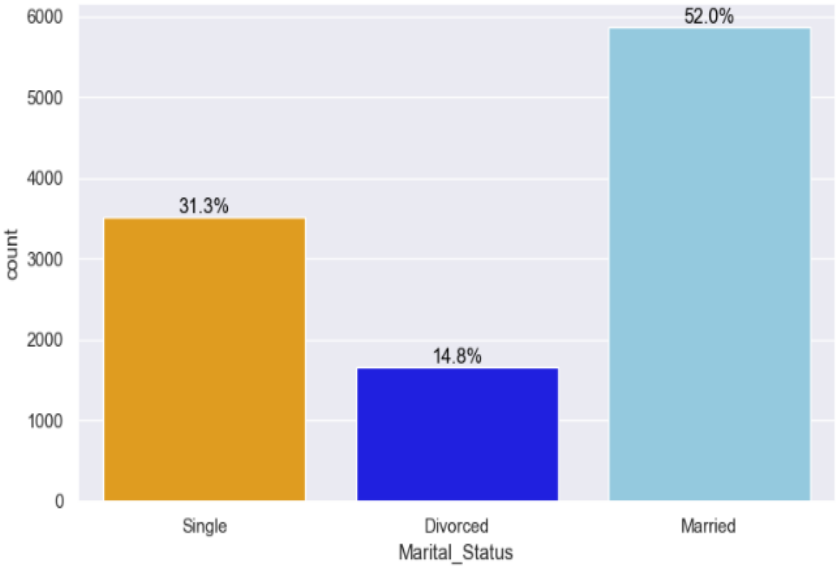


Fig 11. Bar plot of Martial Status

Around 52% of primary account holders are married, approximately 31% are single, and about 15% are divorced.

- **Average Revenue per Month**

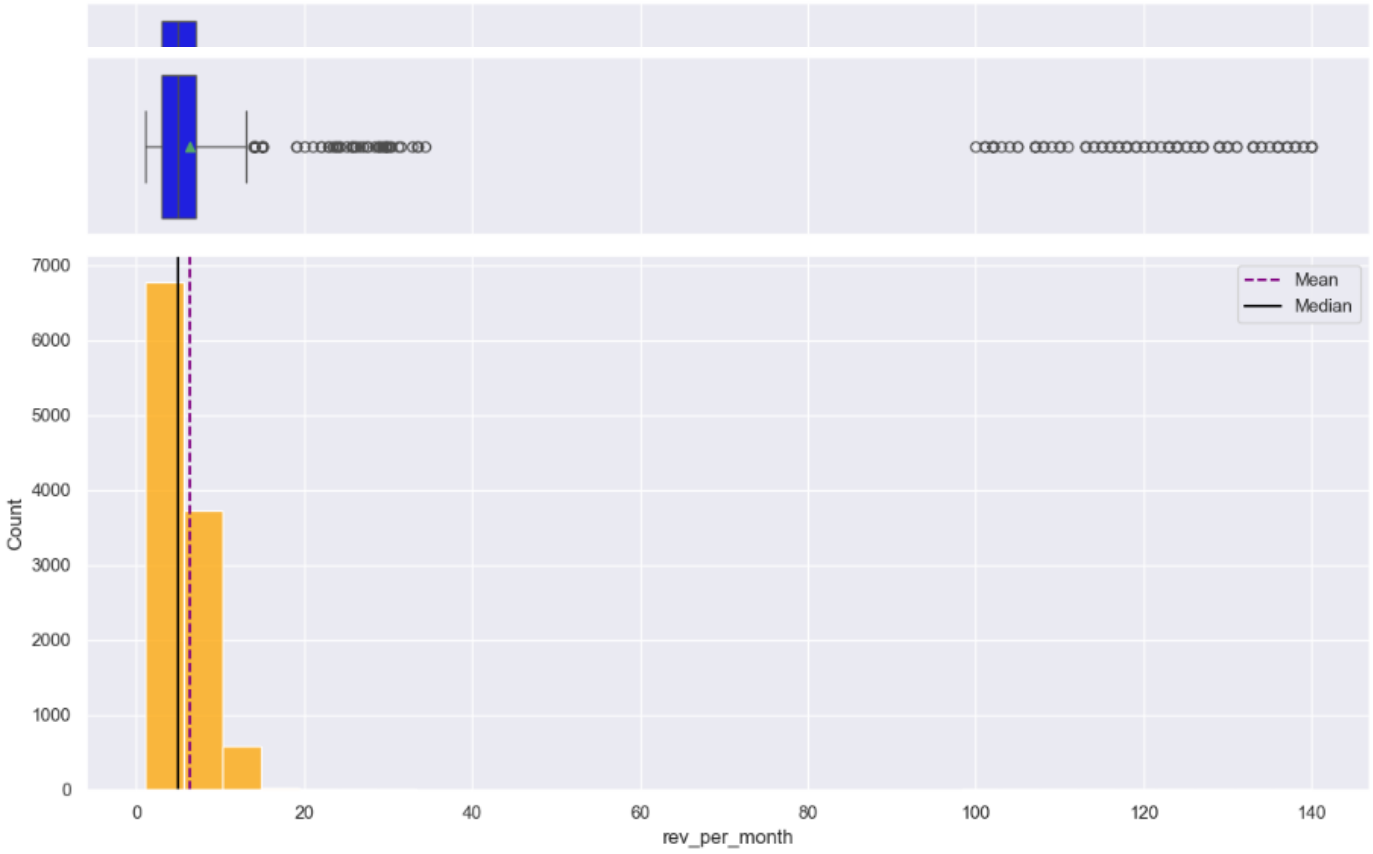


Fig 12. Hist-Boxplot of Average Revenue per Month

The currency is represented in thousands of INR. The interquartile range (IQR) for the average monthly revenue per account falls below ₹10K. However, the distribution is highly skewed due to outliers exceeding ₹100K.

- **Complaints raised in the past year**

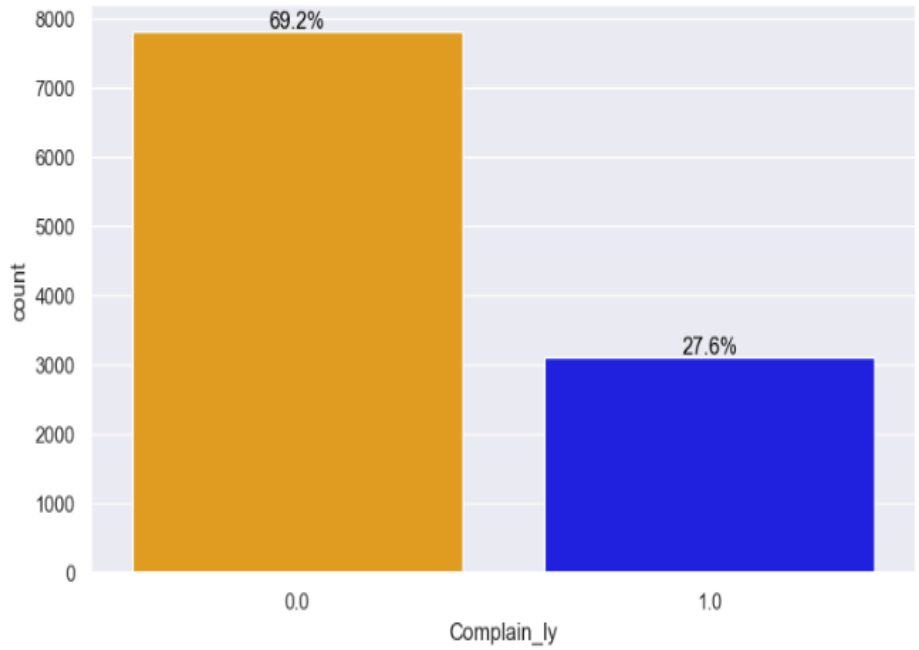


Fig 13. Bar plot of Complaint raised in the past year

This is binary data, where 0 indicates 'No' and 1 indicates 'Yes'. Complaints were raised approximately 28% of the time in the past year, while 69% of the time no complaints were recorded. Identifying recurring or potential complaints can be valuable in predicting customer churn.

- **Revenue Growth Percentage**

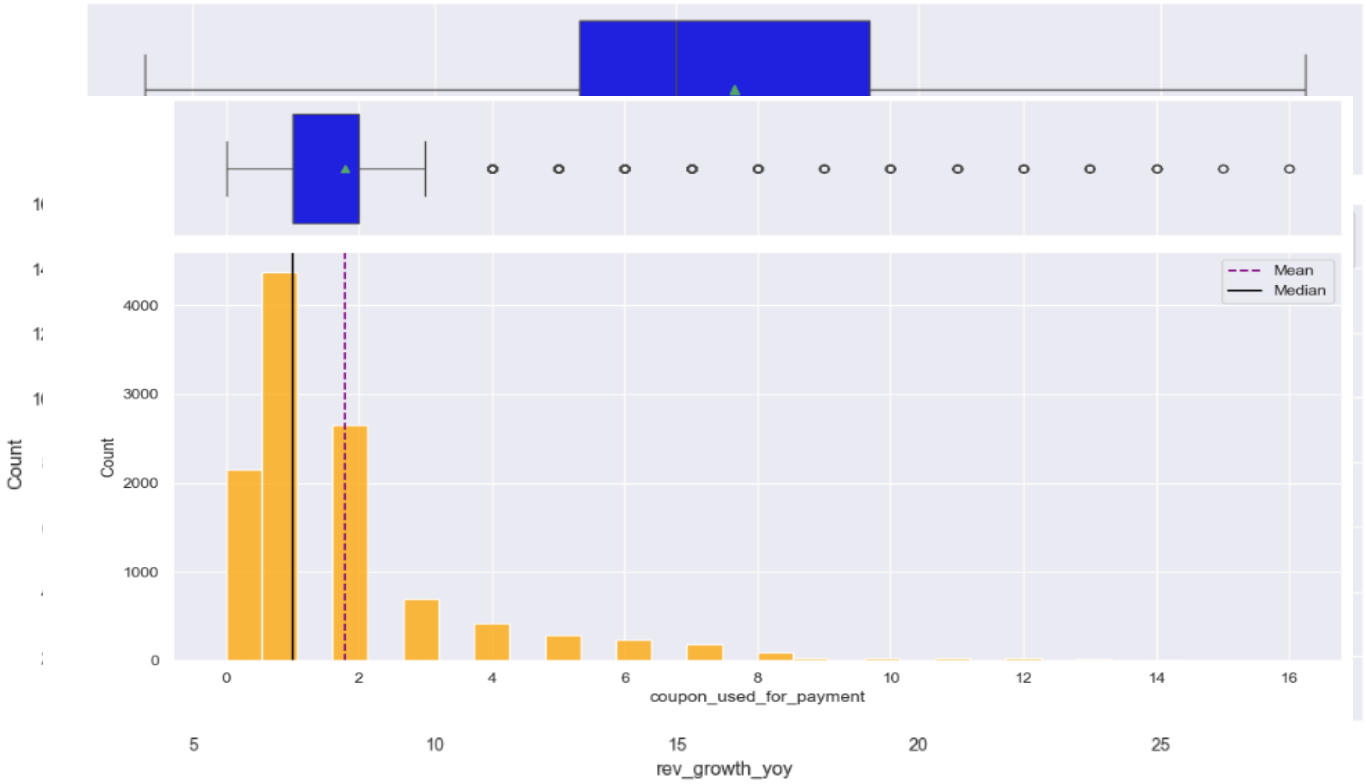


Fig 14. Hist-Boxplot of revenue growth percentage

On average, accounts have shown a ~16% revenue growth over the past year compared to the previous year. While there's room for improvement, this is a positive indicator for the management. The growth rate ranges from approximately 4% to 29%, with an interquartile range (IQR) between ~12% and ~18%.

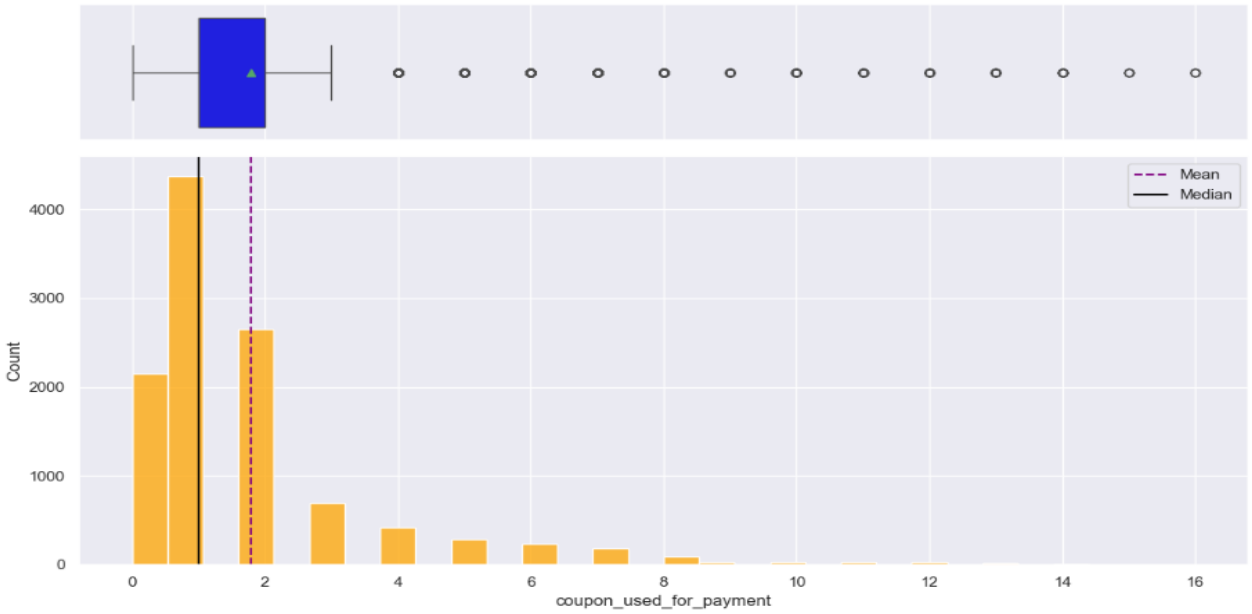
- **Coupon Used for Payment**

Fig 15. Hist-Boxplot of Coupon Used for Payment

On average, coupons were used about 2 times for payments in the past year. The most common usage was 2 times, followed by no usage at all. Outliers are observed in the range of 4 to 16 uses.

- **Days since Customer Care was contacted**

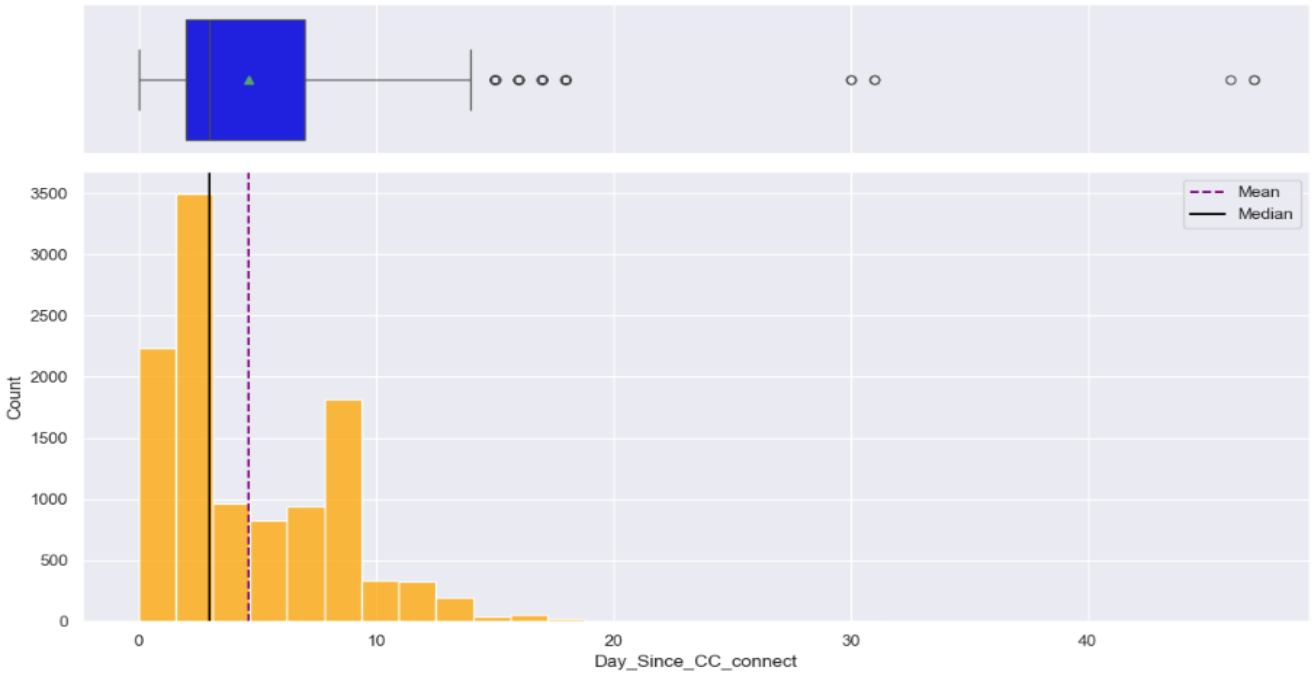
Fig 16. Hist-Boxplot of Coupon Used for Days since Customer Care was contacted



Most customers reconnect within the first three days after reaching out, indicating an opportunity to enhance customer service by identifying and addressing recurring issues. On average, it takes about 5 days for a customer to reconnect, with a noticeable pattern repeating around 5 to 6 days. Outliers are observed around 20, 30, and 50 days.

- **Cashback**

  On average, the cashback earned by accounts in the past year is around 200 INR. However, outliers exist with cashback values ranging between 1800 and 2000 INR.



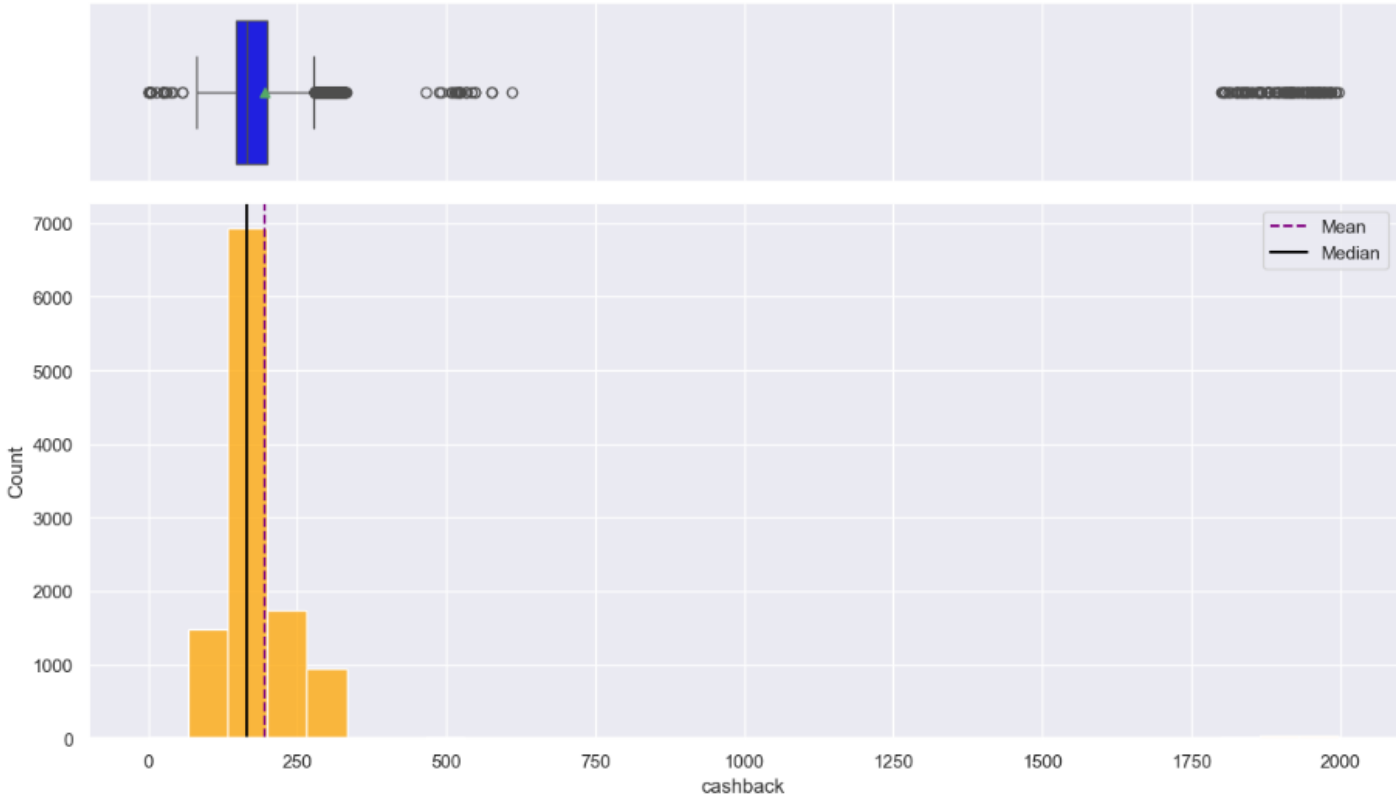


Fig.17 Hist-Boxplot of cashback given

- **Login Device**

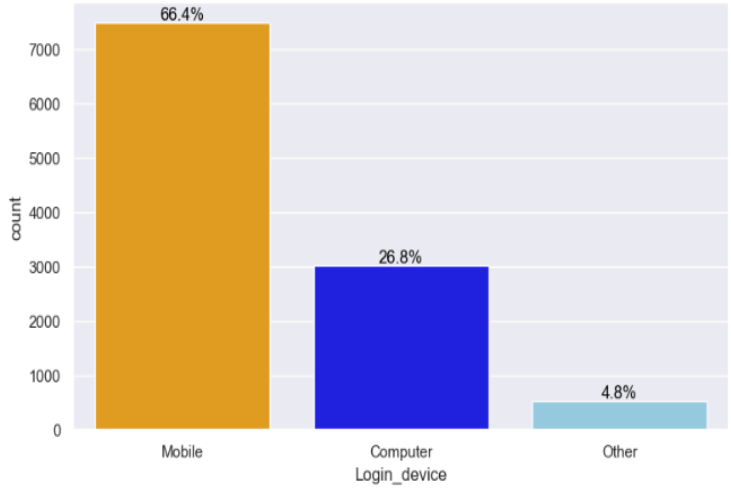


Fig.18 Bar plot of cashback given

Mobile is the most preferred login device, used by approximately 67% of customers, followed by Computers at around 29%. The 'Other' category accounts for about 5%, which may include devices such as Smart TVs.

2. Bivariate Analysis

- **Co-relation Matrix**

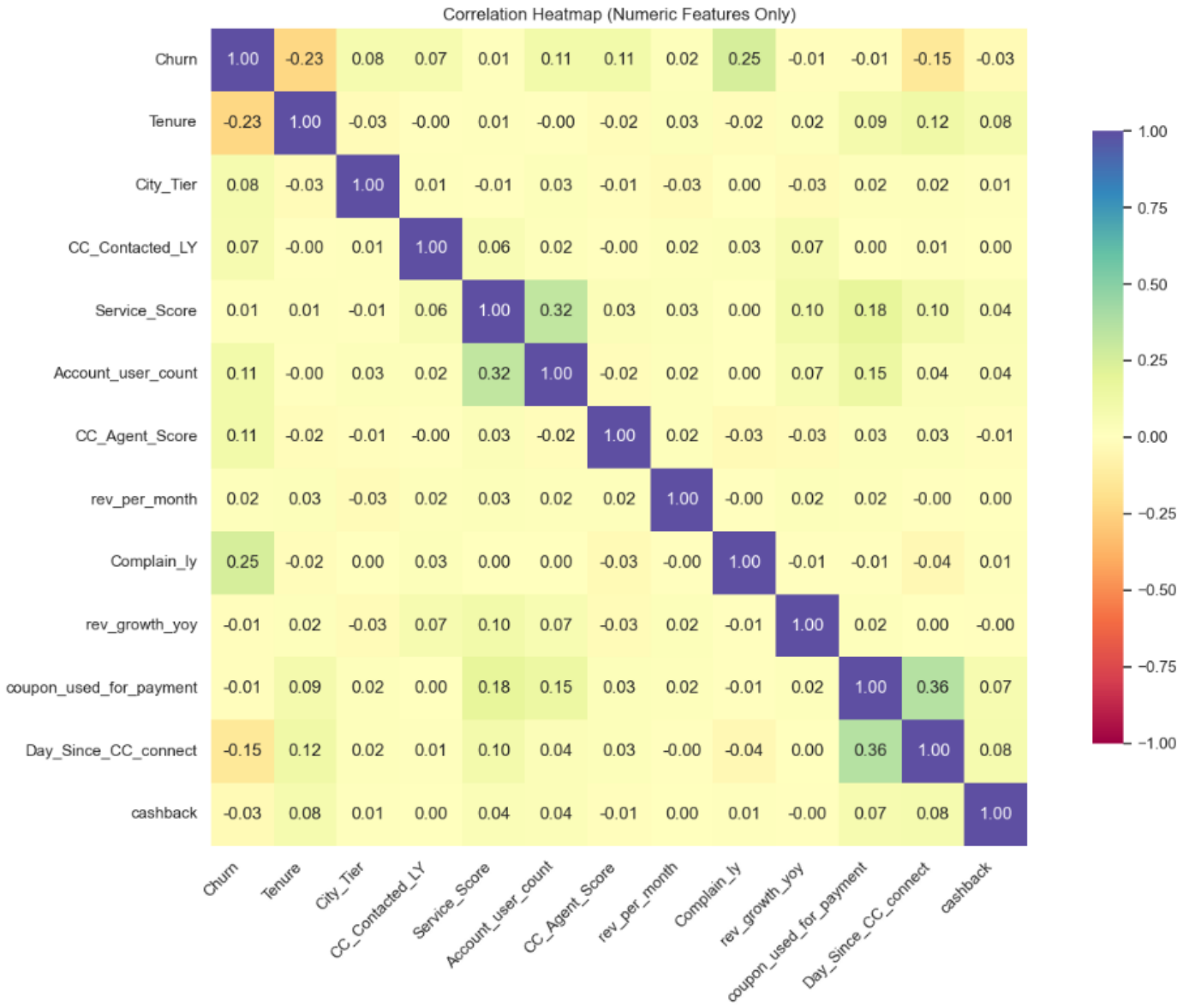


Fig 19. Co-relation matrix

- Churn has a moderate positive correlation with Complain_ly, Account_user_count, and CC_Agent_Score, and a moderate negative correlation with Tenure and Day_Since_CC_connect.
- Service_Score shows a moderate positive correlation with Account_user_count and coupon_used_for_payment, and a weak positive correlation with rev_growth_yoy and Day_Since_CC_connect.
- Account_user_count is moderately positively correlated with Service_Score, coupon_used_for_payment, and Churn.
- rev_growth_yoy has a weak positive correlation with Service_Score.
- coupon_used_for_payment shows moderate positive correlation with Day_Since_CC_connect, Service_Score, and Account_user_count.
- City_Tier, CC_Contacted_LY, rev_per_month, rev_growth_yoy, and cashback have weak positive or negative correlations with other variables.
- Overall, no strong correlations (positive or negative) are observed among the variables.

- **Pair plot: Visualizing Bivariate Relationships Between Key Features**

Fig.20 Pair plot between key features

- Customers with lower tenure in the company have a higher likelihood of churning.
- City Tier 1 and 3 customers exhibit relatively higher churn rates compared to others.
- Customers who have had fewer interactions with customer care in the past year tend to churn more.
- Customers who provided service satisfaction scores of 2, 3, or 4 are more prone to churn.
- Accounts with 3, 4, or 5 users show a greater tendency to churn.
- Agent satisfaction scores of 3 and above are commonly observed among churned customers.
- Customers associated with lower average monthly revenue and lower revenue growth (YoY) are more likely to churn.
- There is no clear churn pattern based on the number of complaints received in the past year.
- Infrequent usage of coupons for payment correlates with a higher probability of churn.
- Customers receiving lower cashback and those frequently contacting customer care have a higher likelihood of churning.

- **Target vs Tenure**

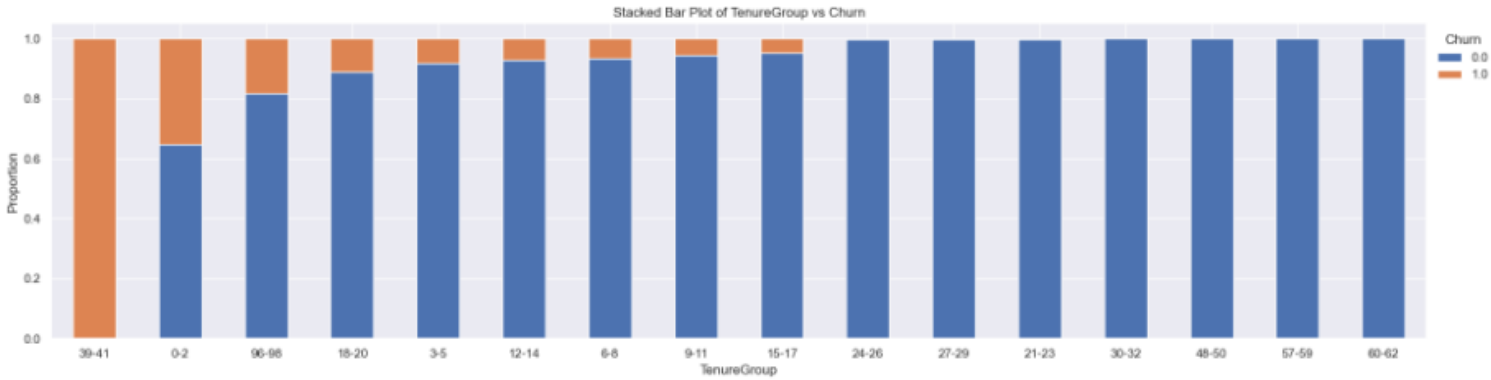


Fig.21 Stack bar-plot Target vs Tenure

- Customers with shorter tenure, particularly those with 1 month or less, have a higher likelihood of churning.
- Therefore, targeted efforts should focus on retaining these new customers.
- Notably, there is no churn observed among customers with tenure between 22 and 61 months.

- **Target vs City_Tier**

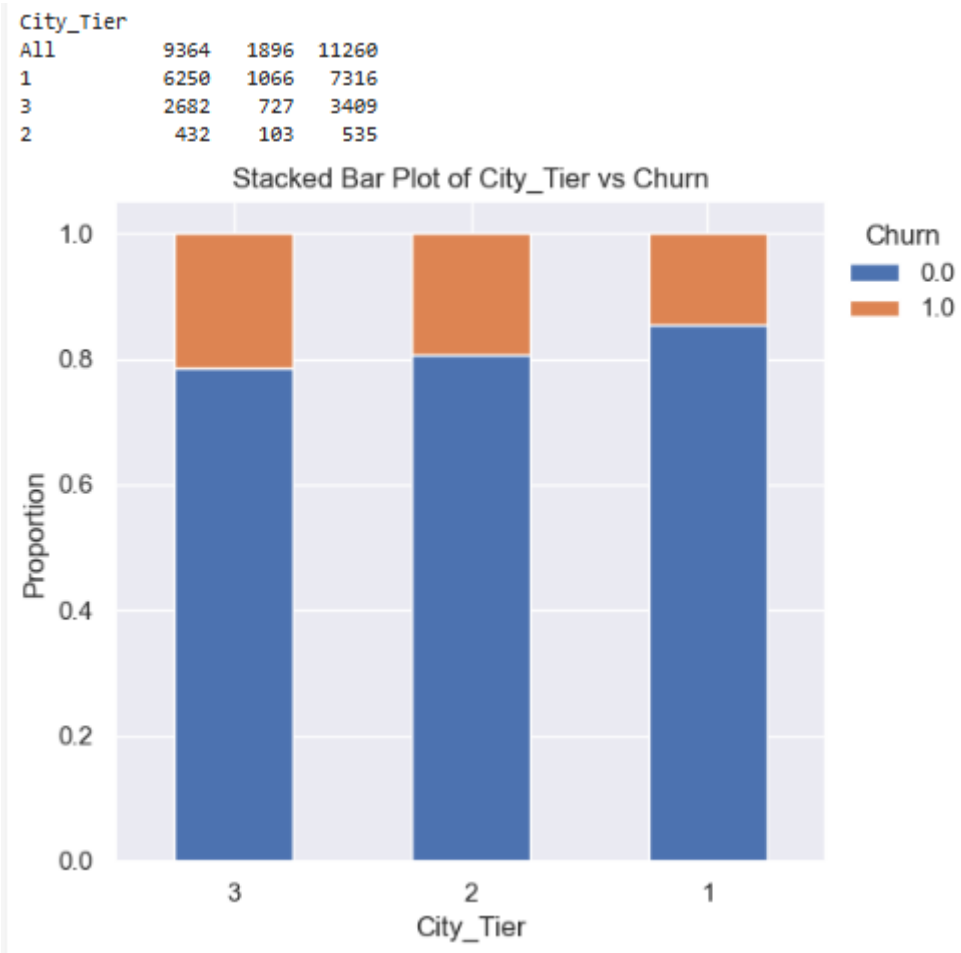


Fig.22 Stack bar-plot Target vs City_Tier

- Customers from Tier 3 cities show the highest churn rate, followed by those from Tier 2 and Tier 1 cities.
- Despite being fewer in number, Tier 2 customers exhibit a relatively high churn rate.

- **Target vs Customer Care Contacted (Past year)**

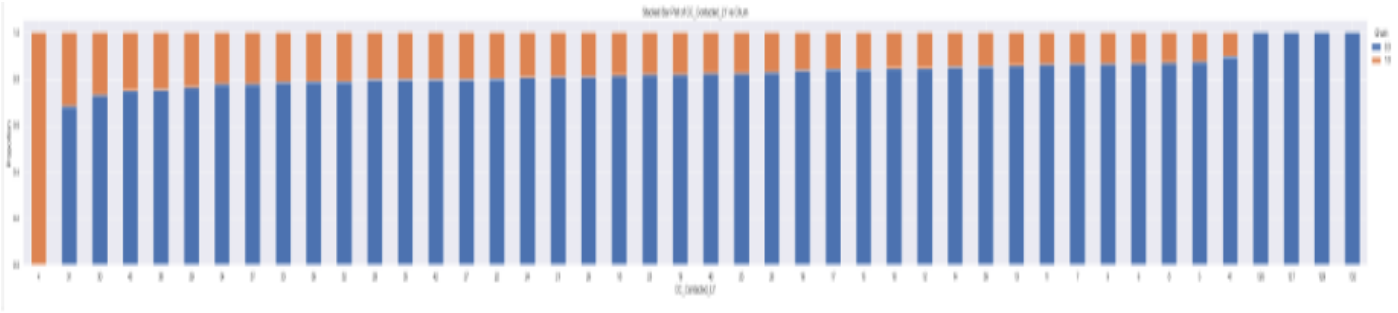


Fig.23 Stack bar-plot Target vs Customer Care Contacted

- Only one customer has contacted customer care either exactly 4 times or more than 100 times in the past year and has churned. Since this represents a single instance, no concrete conclusion can be drawn from it.
- The record showing more than 100 customer care contacts requires further investigation to determine if it is a valid case or a potential data entry error.
- Overall, there is an observable trend suggesting that a higher number of customer care contacts is associated with an increased likelihood of customer attrition.

- **Target vs Payment**

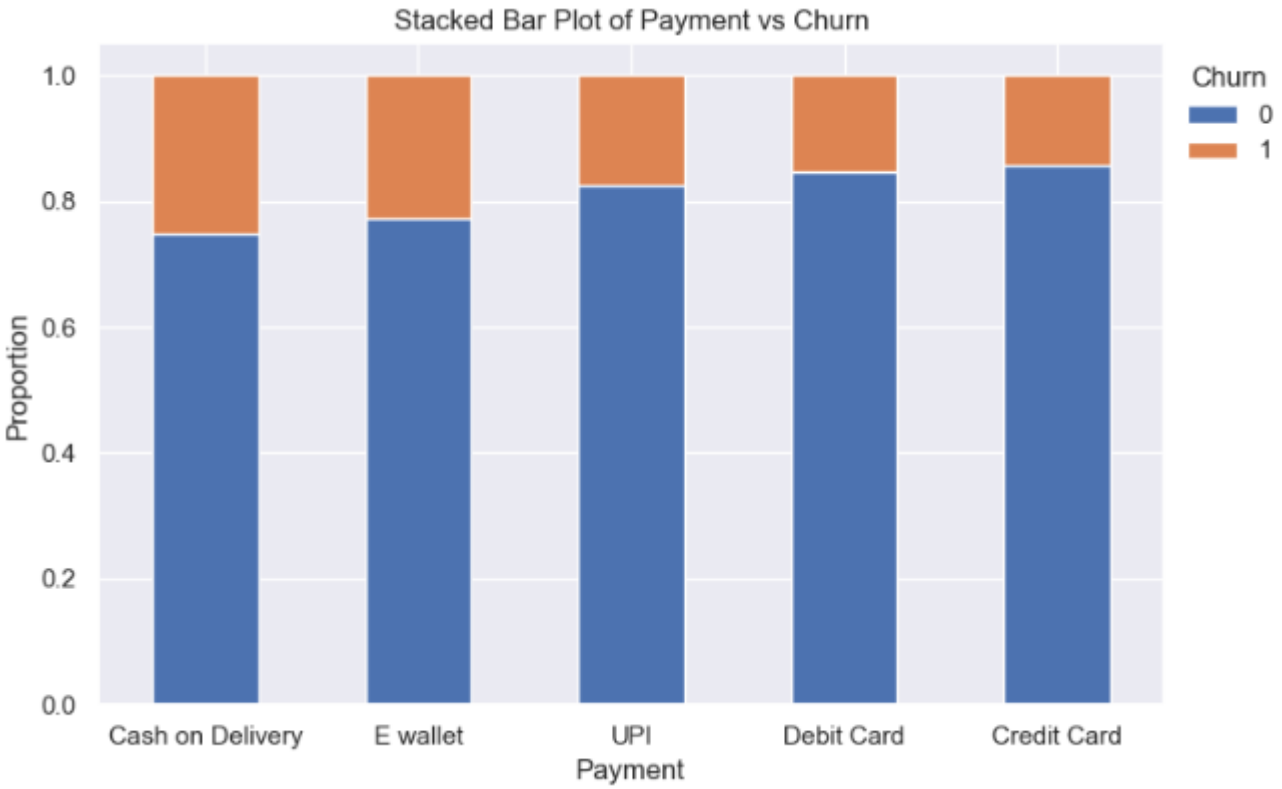


Fig.24 Stack bar-plot Target vs Payment method

- Customers using Cash on Delivery and E-wallet as their payment methods tend to churn more.
- Payment methods like UPI, Debit Card, and Credit Card exhibit a similar churn pattern.

- **Target vs Gender**

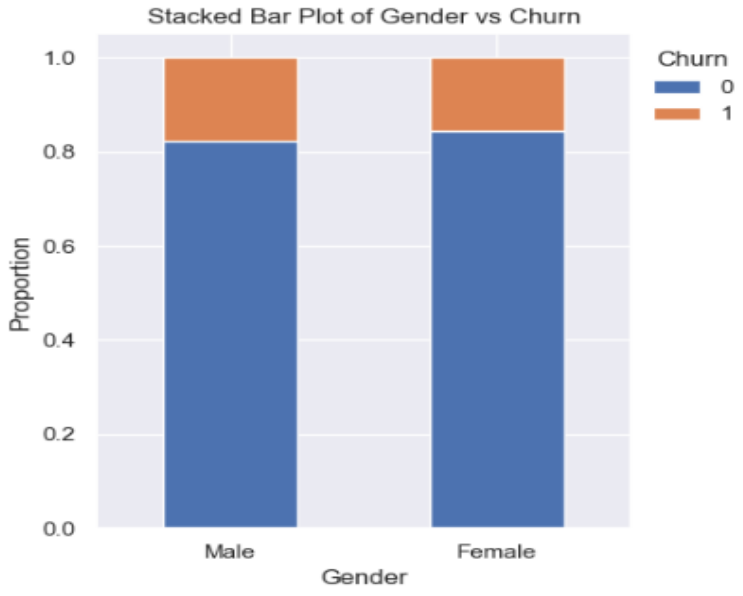


Fig.25 Stack bar-plot Target vs Gender

- Male customers exhibit a slightly higher attrition rate, which may be due to the larger proportion of male customers compared to female customers.

- **Target vs Service Score**

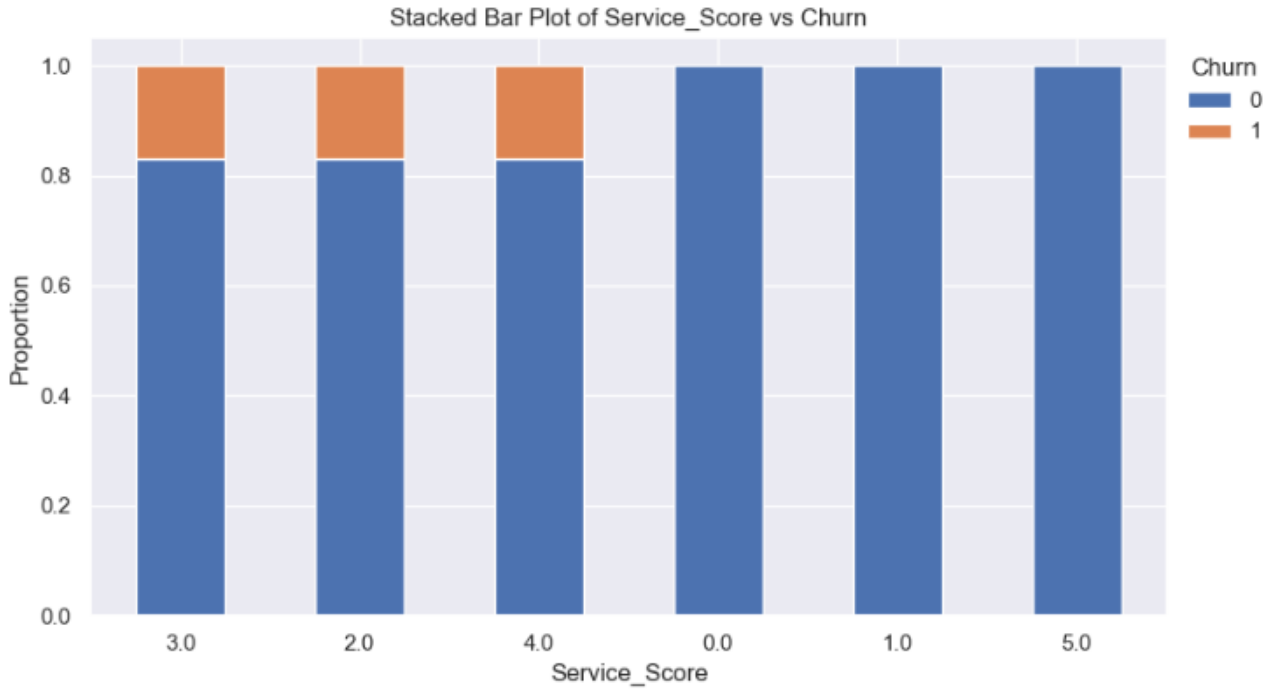


Fig.26 Stack bar-plot Target vs service score

As observed, customers who rated the satisfaction score as 2, 3, or 4 have a higher attrition rate. Interestingly, those who gave very low scores of 1 or below tend to remain with the service.

- **Target vs Account user count**

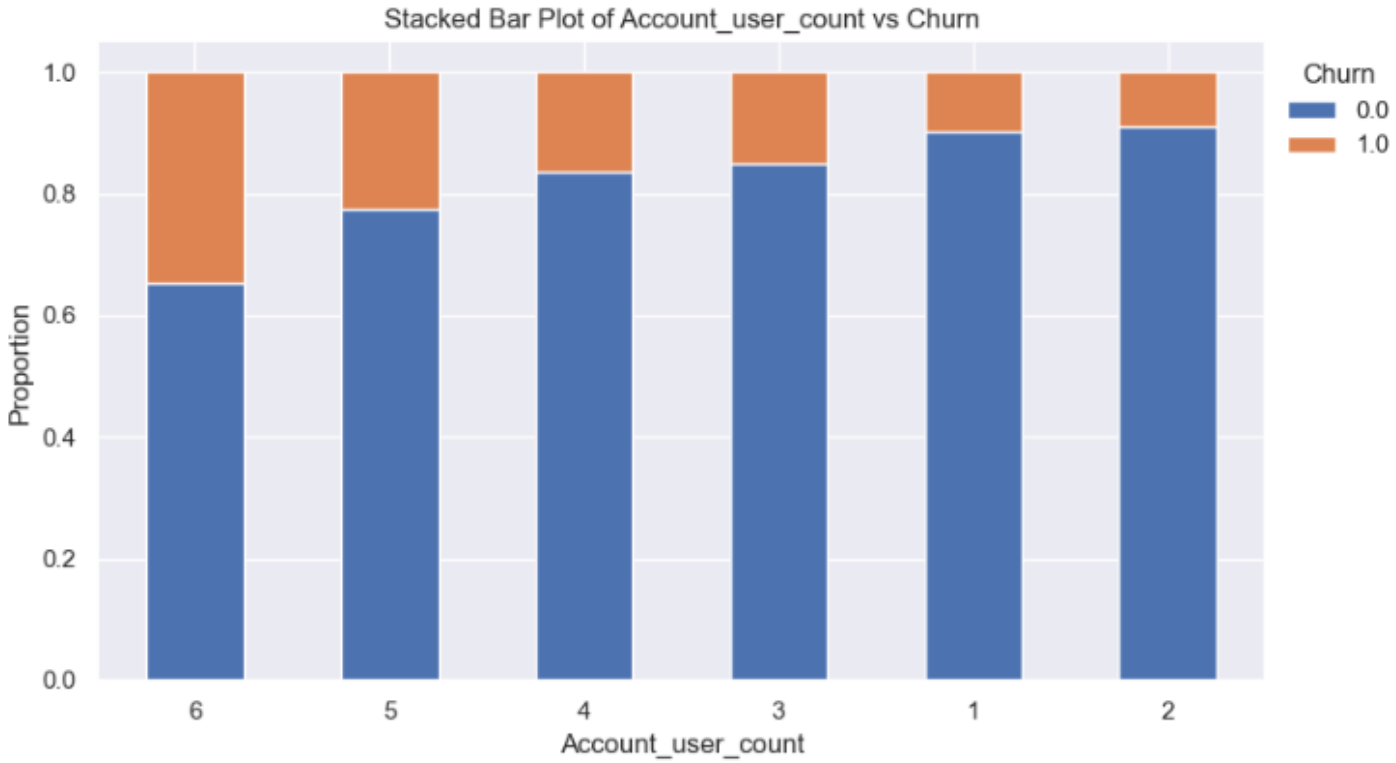


Fig.27 Stack bar-plot Target vs Account user count

The greater the number of customers tagged to an account, the higher the likelihood of customer churn.

8. Target vs Account segment

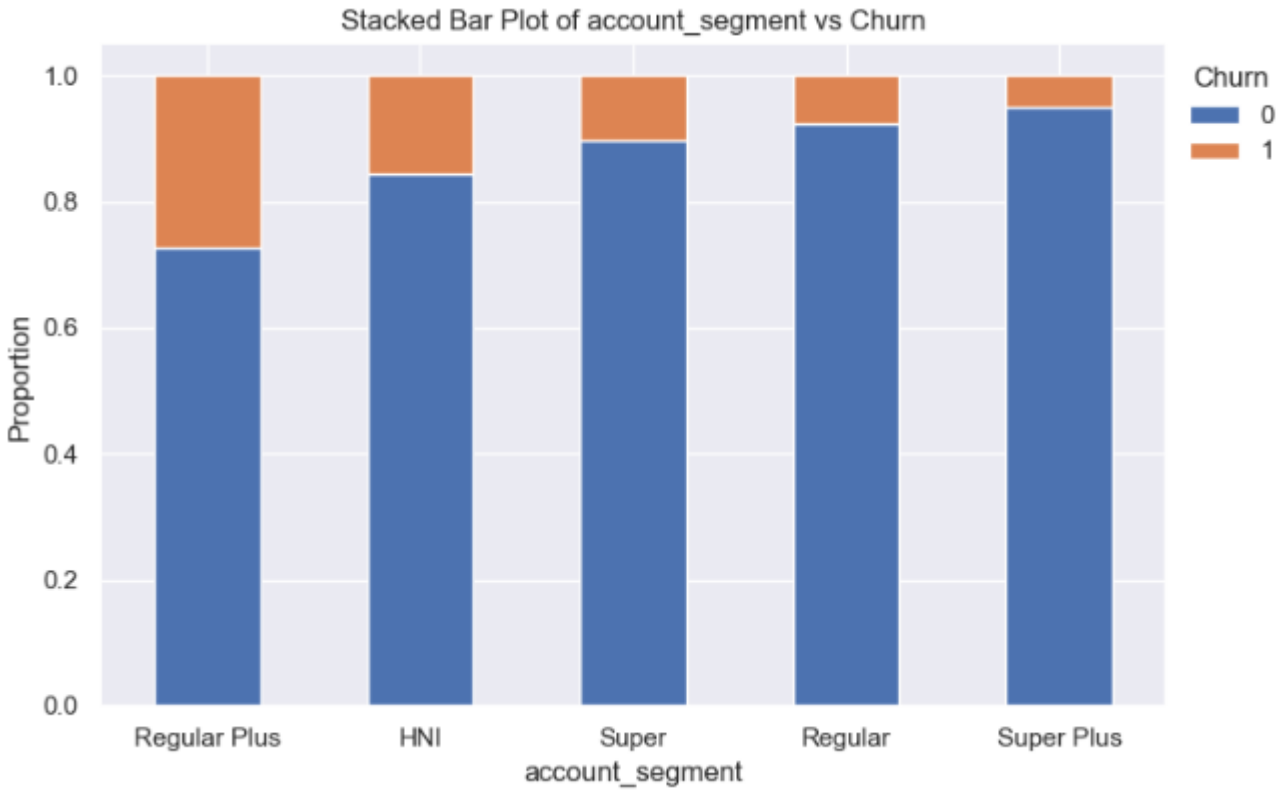


Fig.28 Stack bar-plot Target vs Account segment type

- Regular Plus show higher churn rate followed by HNI and Super segments

Target vs Customer Care Agent Score

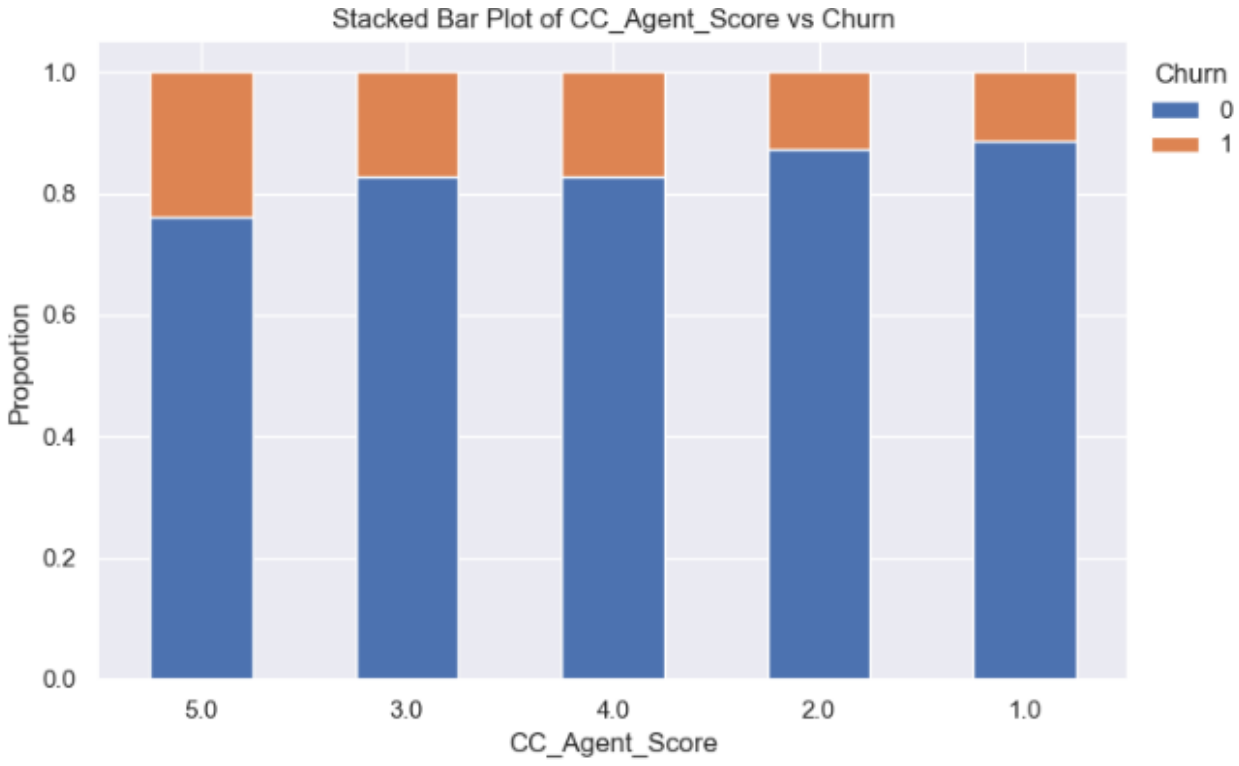


Fig.29 Stack bar-plot Target vs Customer Care Agent Score

Customers who rated the customer care agent with a score of 3 or higher exhibit a higher churn rate.

10. Target vs Marital Status

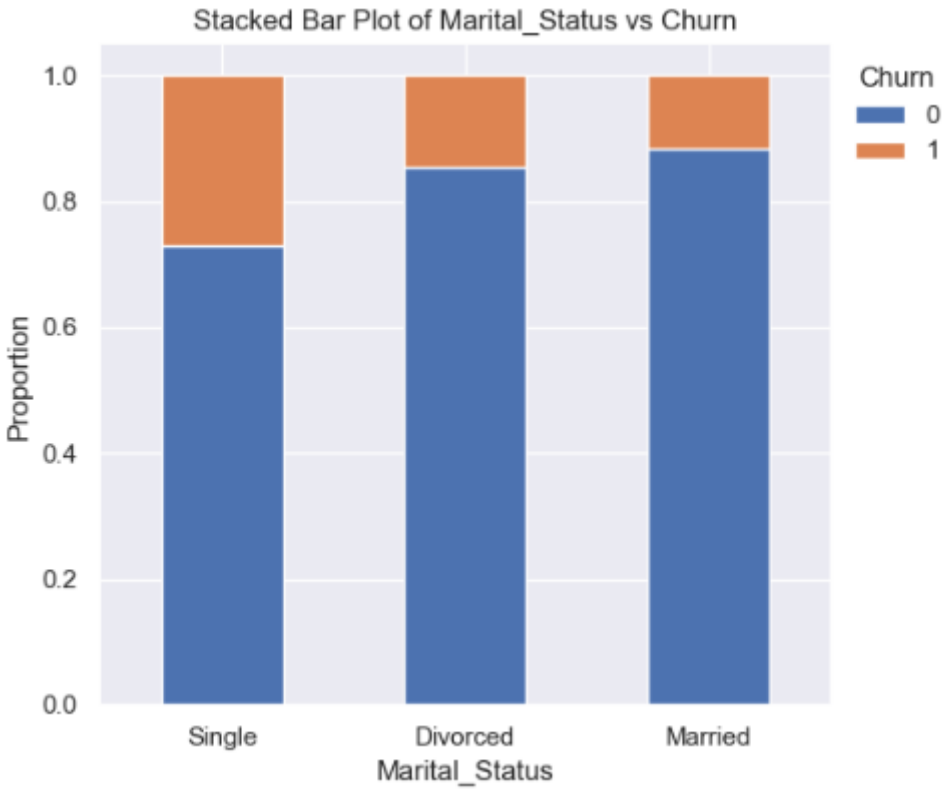


Fig.30 Stack bar-plot Target vs Marital Status

Single customers have the highest churn rate, followed by Divorced and then Married customers.

11. Target vs Average Revenue per Month

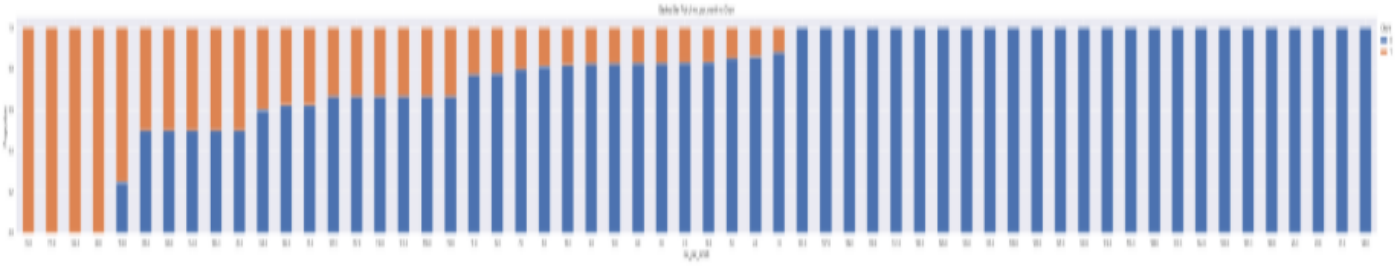


Fig.31 Stack bar-plot Target vs Average Revenue

Accounts with an average monthly income between 3,000 and 9,000 INR have exhibited a comparatively higher attrition rate**.**

12. Target vs Complain in the past year

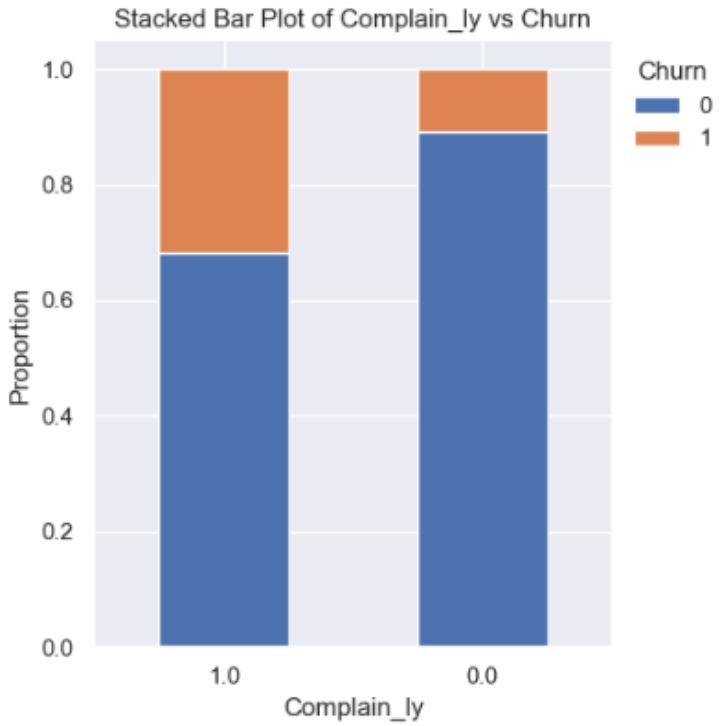


Fig.32 Stack bar-plot Target vs complain

Customers who raised complaints in the past year have shown higher churn rates. It is crucial for the service and customer care teams to respond promptly and follow up to ensure issues are fully resolved, helping to improve customer retention.

13. Target vs revenue growth percentage

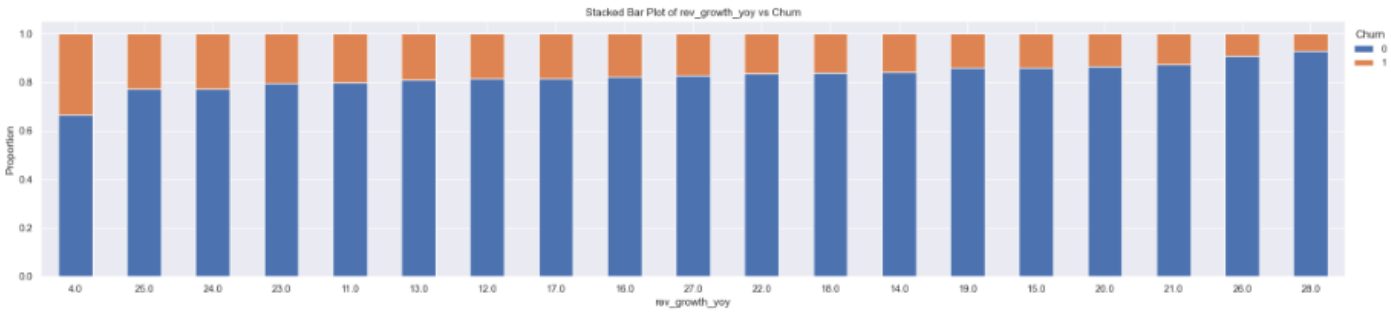


Fig.33 Stack bar-plot Target vs revenue growth percentage

- On an average, accounts with lower revenue growth percentage in the past year compared to the previous year have quit the services

14. Target vs coupon_used_for_payment

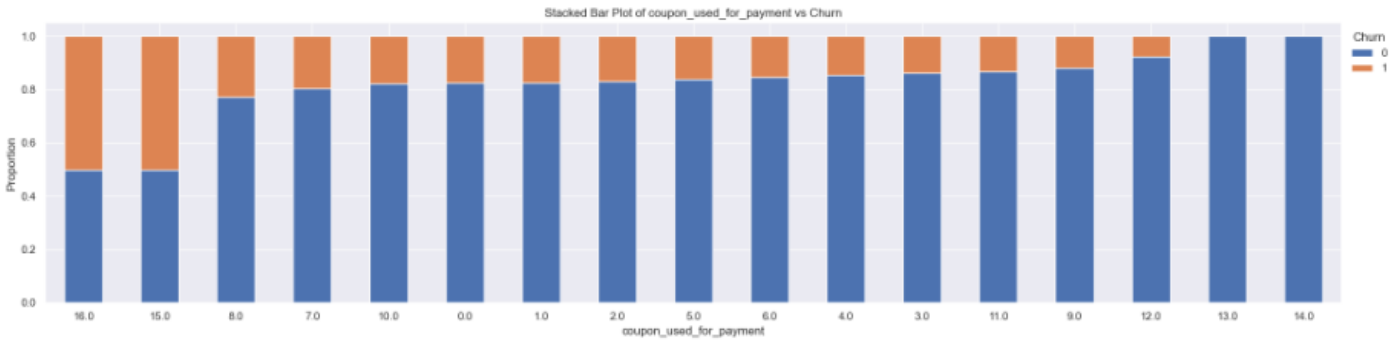


Fig.34 Stack bar-plot Target vs coupon_used

Only two customers used coupons the highest number of times (15 or 16) for payment, and both have churned. On average, customers who used coupons two times or less tend to have a higher attrition rate.

15. Target vs Day_Since_CC_connect

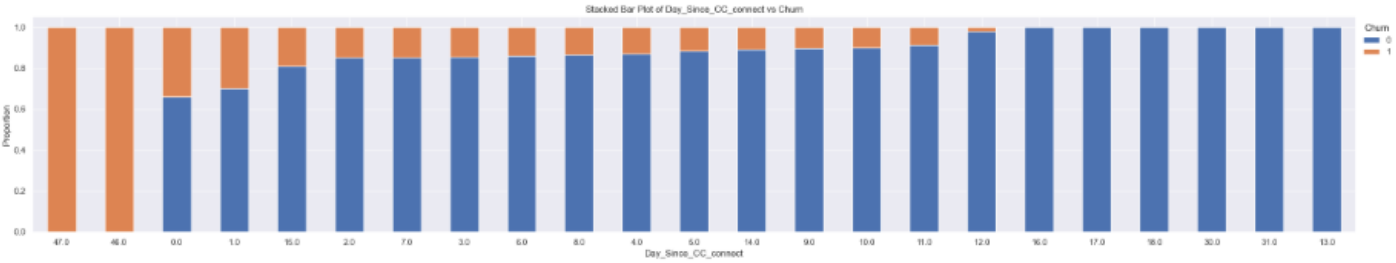


Fig.35 Stack bar-plot Target vs Day since_CC_connect

On average, customers who frequently contacted customer care show a higher attrition rate. Only about two customers contacted customer care after approximately six weeks and have churned.

16. Target vs cashback

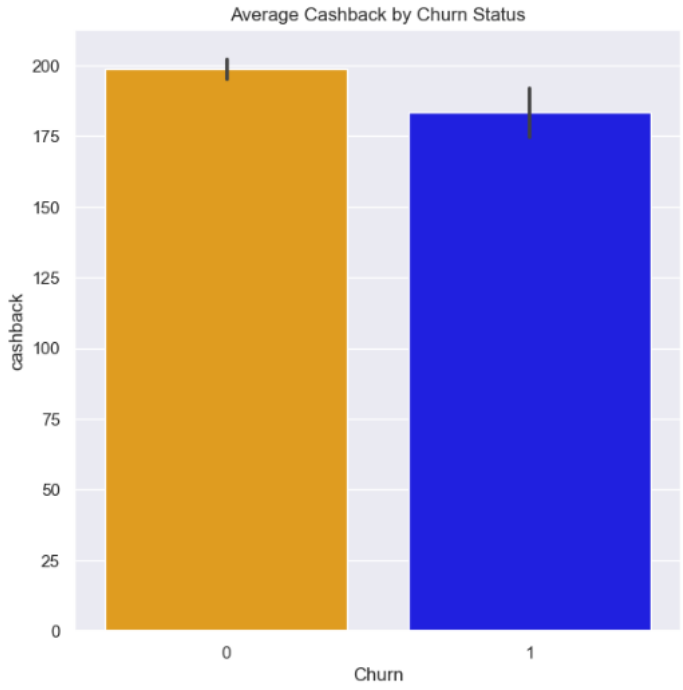


Fig.36 Bar-plot Target vs Cashback

In general, customers receiving very low cashback tend to have a higher likelihood of churn

17. Target vs Login Device

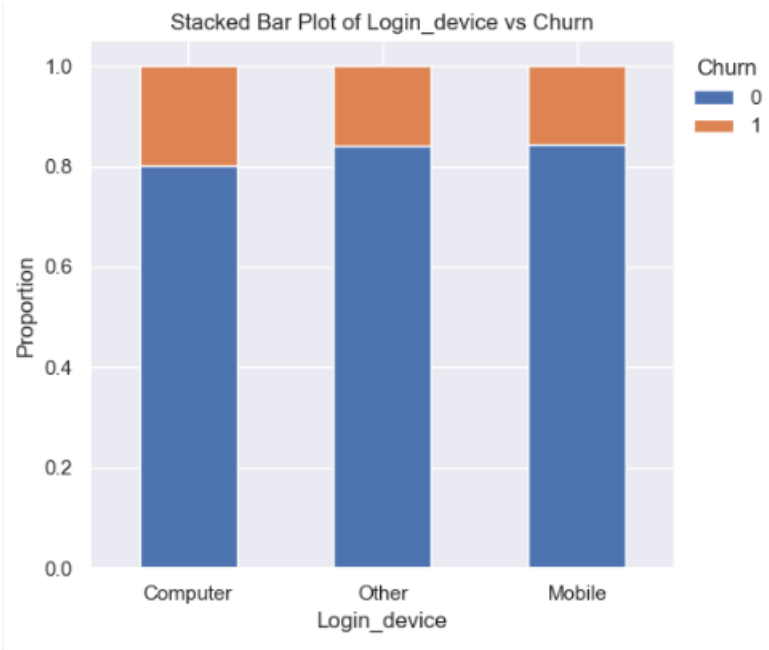
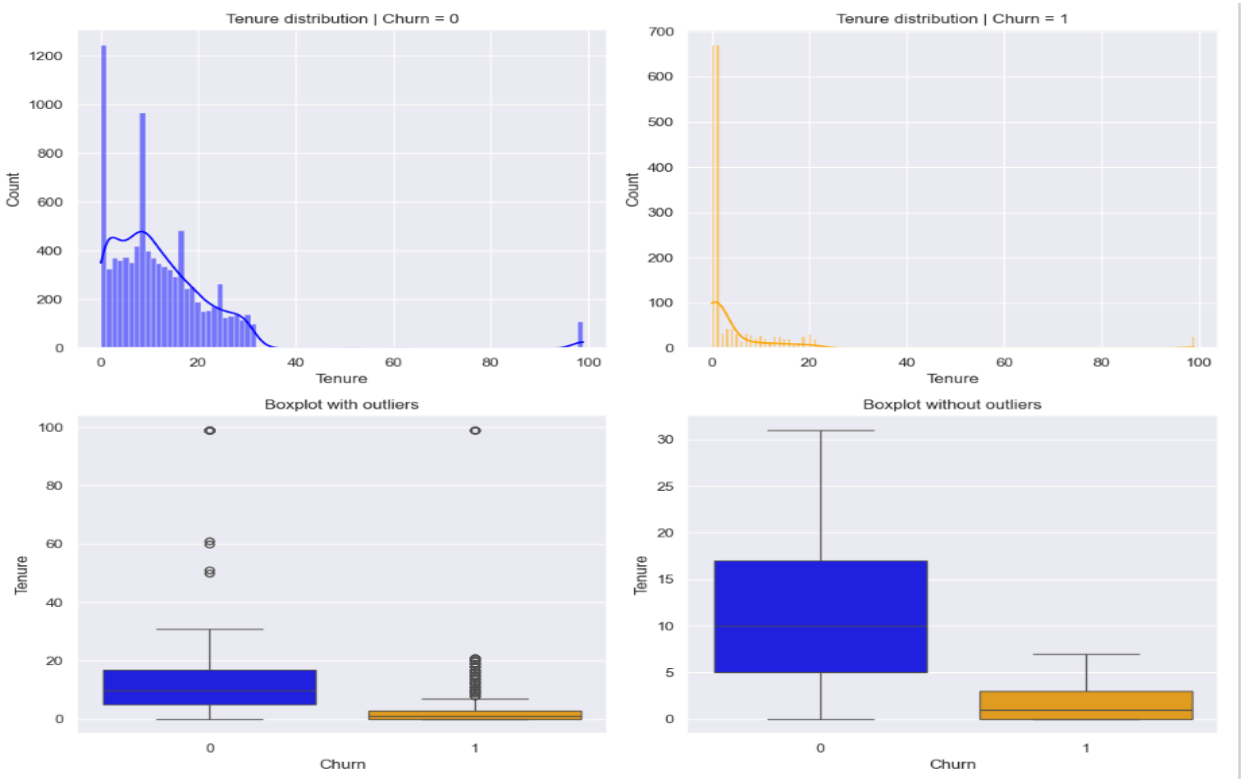


Fig.37 Stack bar-plot Target vs Login Device

Customers logging in via Computers exhibit a relatively higher attrition rate compared to other devices.

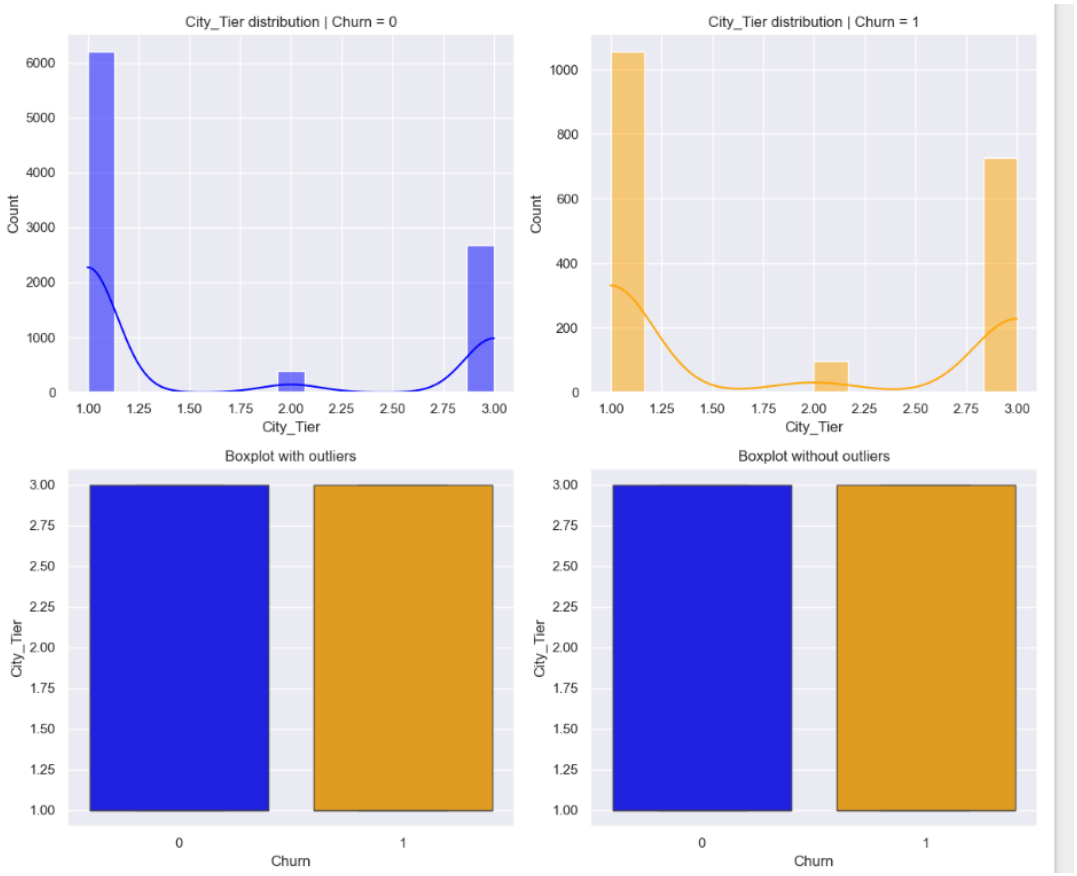
Distribution of Target w.r.t Independent variables

1. Churn vs Tenure



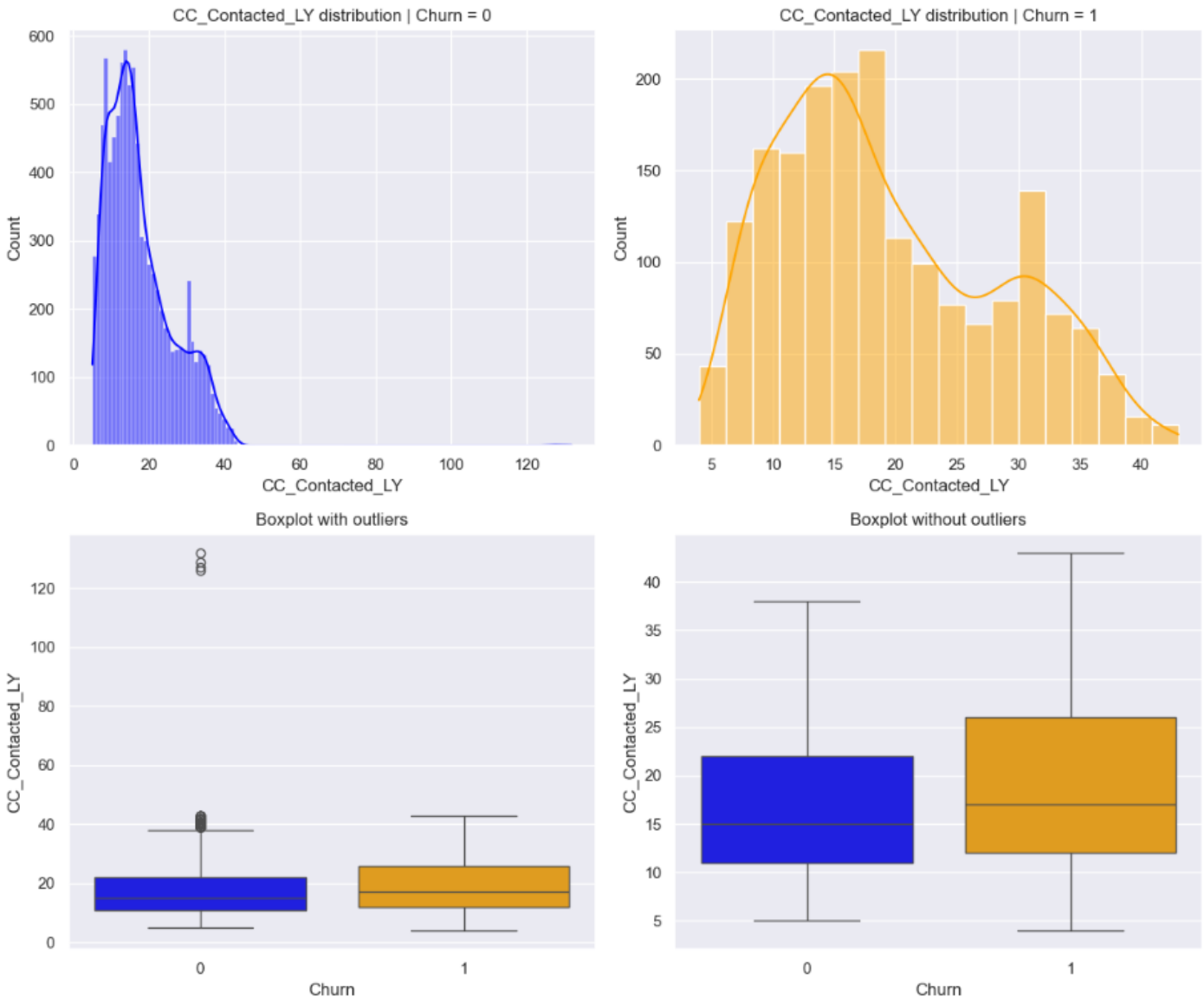
Customers with shorter tenure, typically new customers, have shown higher attrition rates.

2. **Churn vs City_Tier**



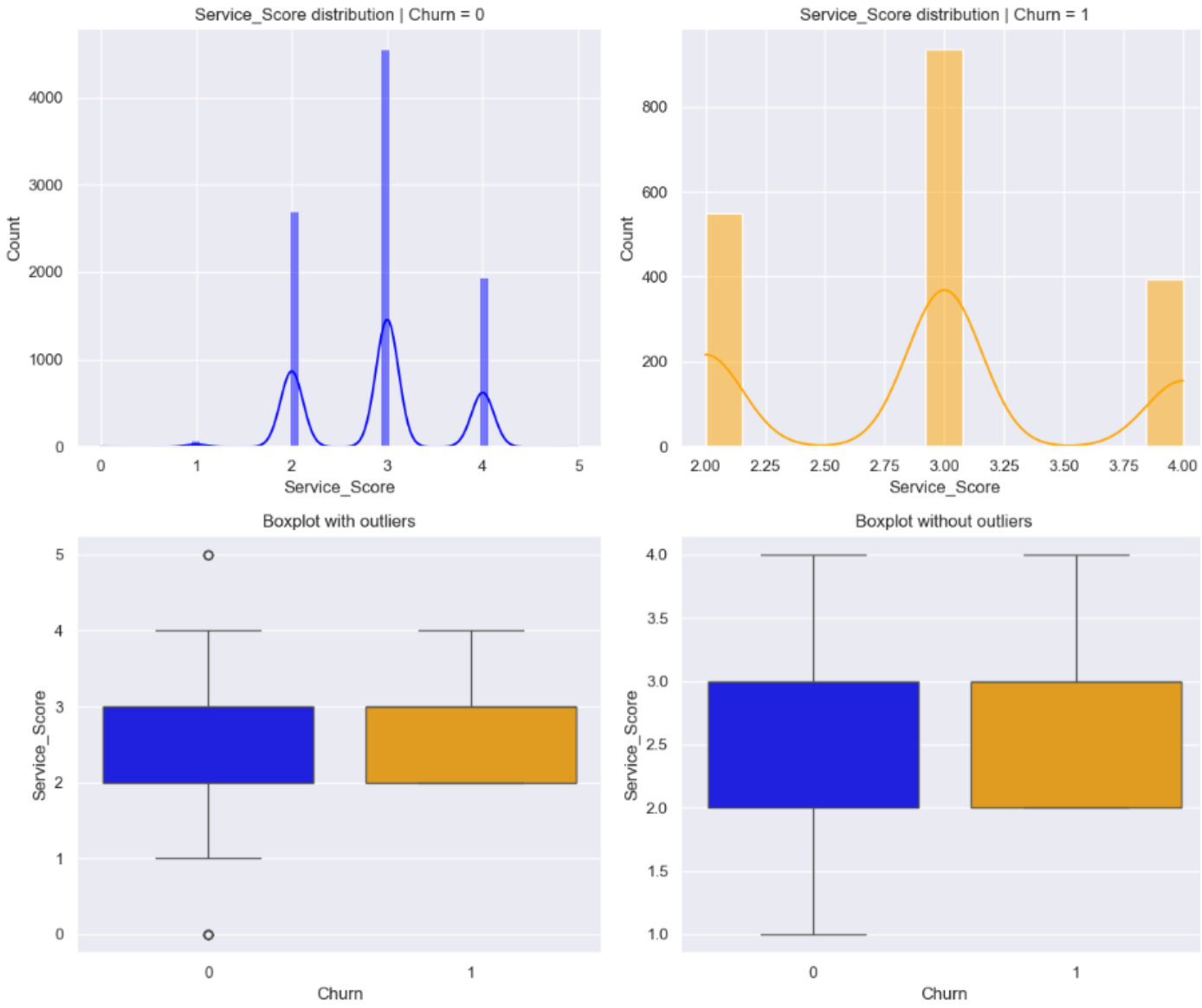
The distribution of customers across city tiers is similar for both attrited and non-attrited groups.

3. Churn vs CC_Contacted_LY



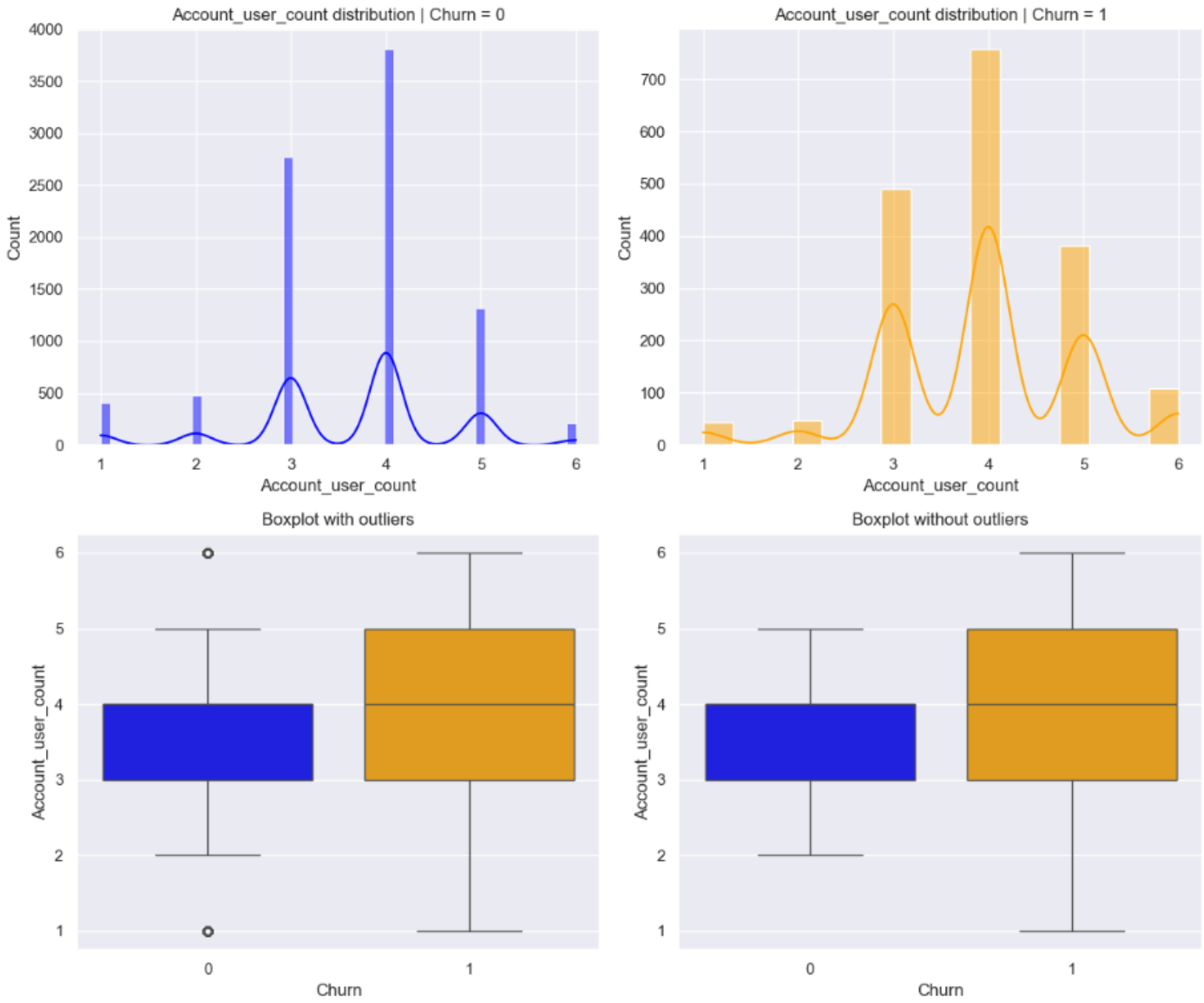
- If the customer care is contacted anywhere between 12 to 27 times a year, higher the chances for the customer to attrite

4. Churn vs Service_Score



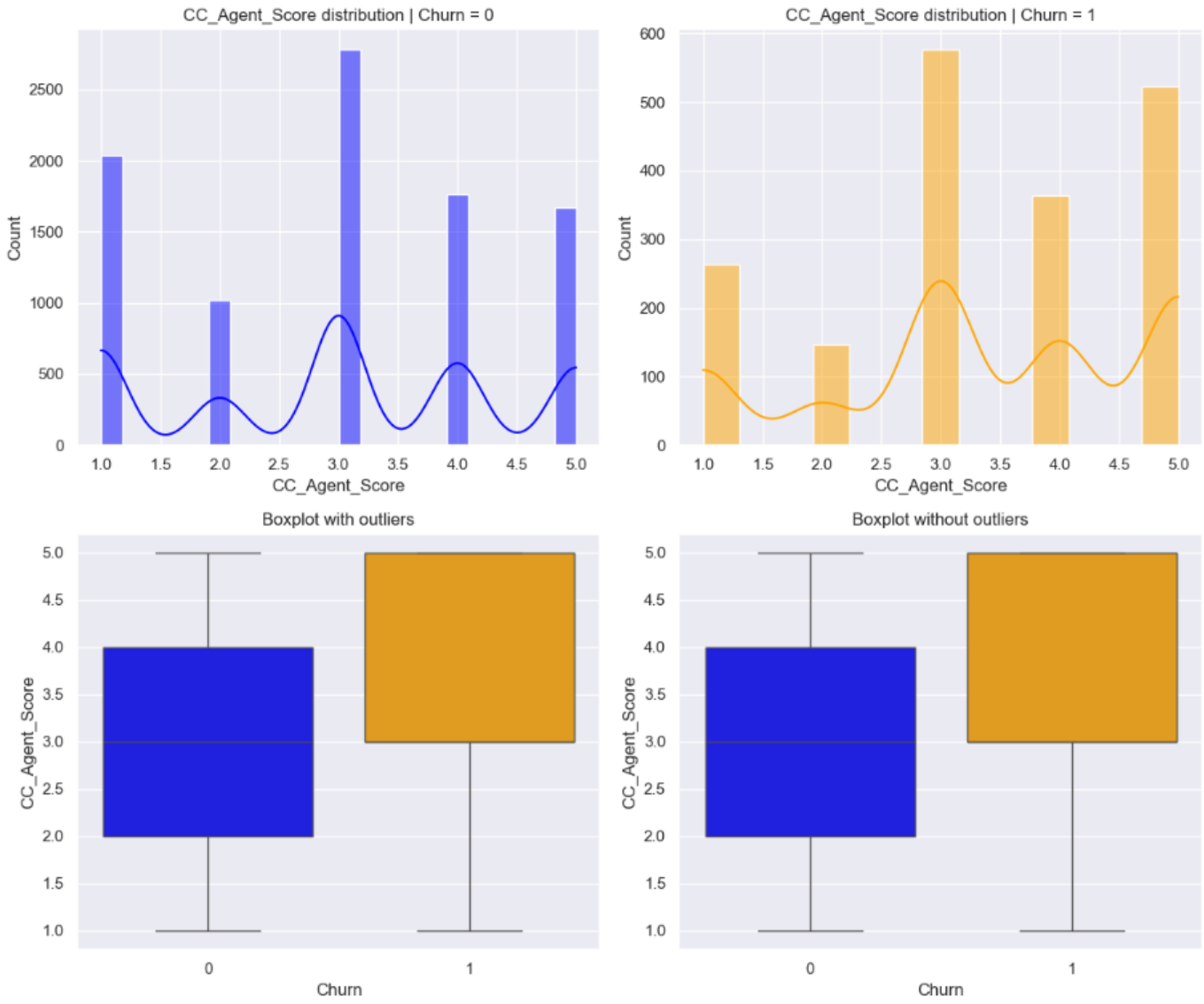
Customers who gave a service score of 2,3 or 4 are the ones who attrited

5. Churn vs number of user per account



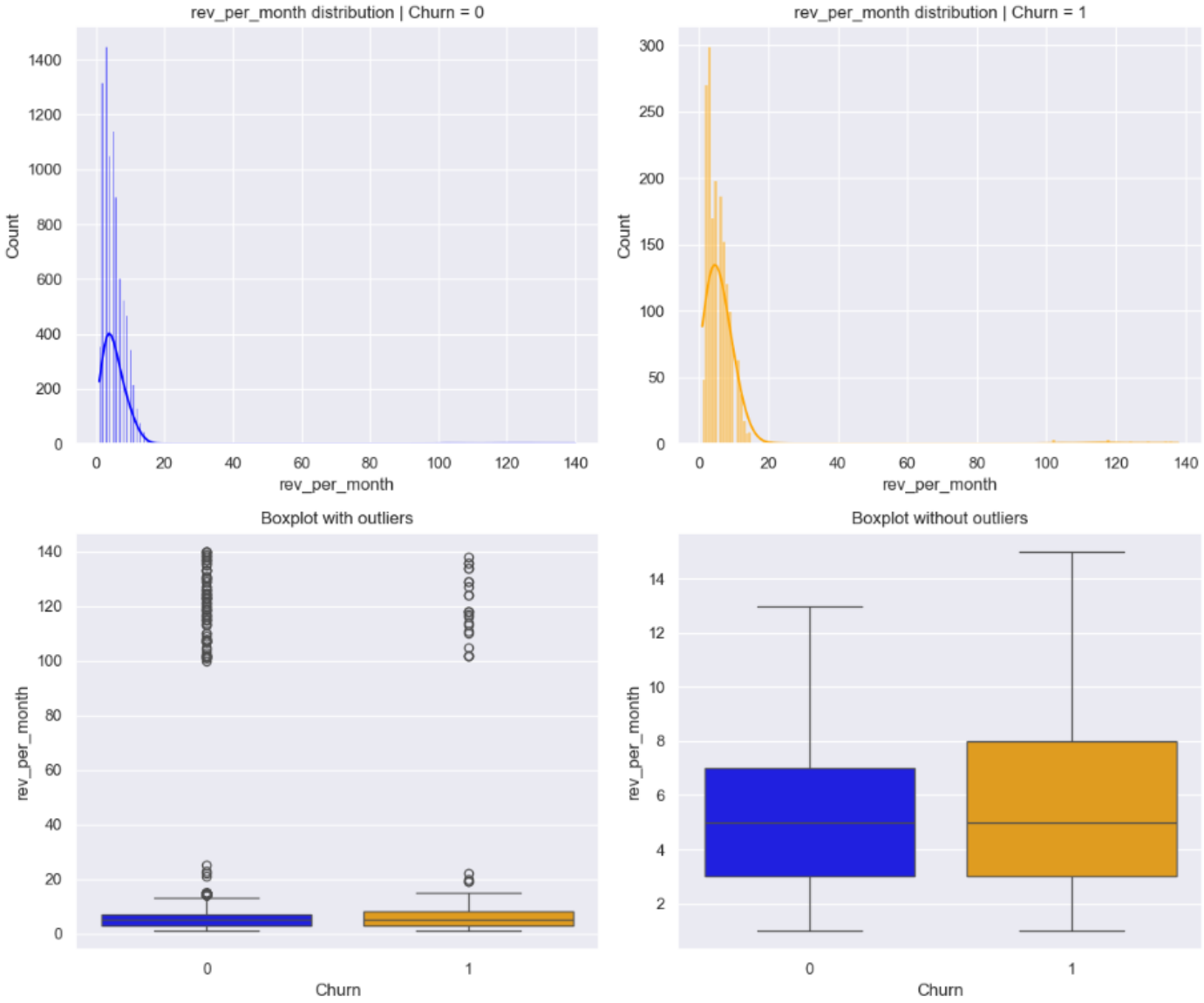
- Attrition is higher among customers with 3 or more users tagged with the account

6. Churn vs Customer care agent score



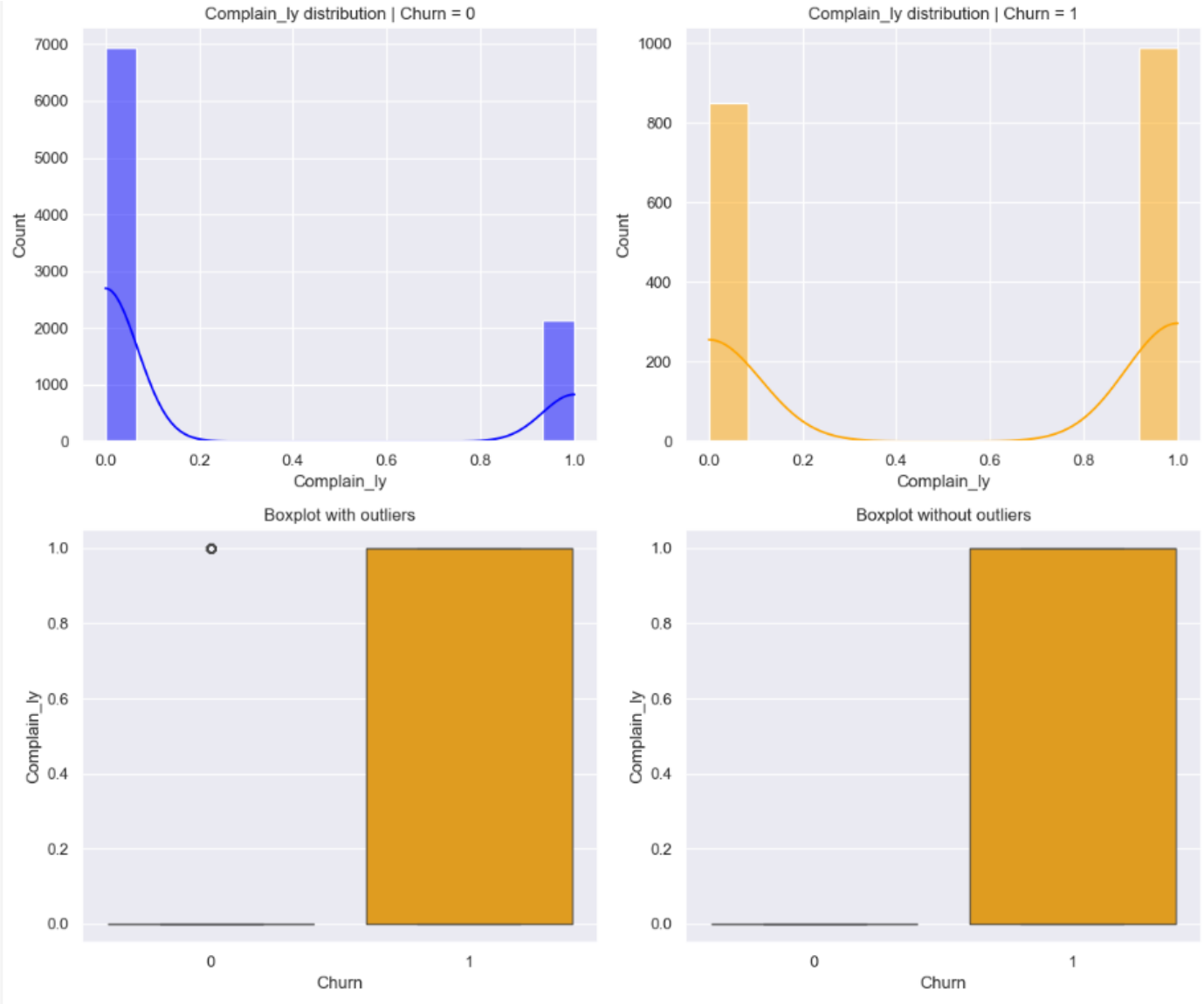
Attrition is higher among customers who have given an agent score of 3 or more

7. Churn vs revenue_per_month



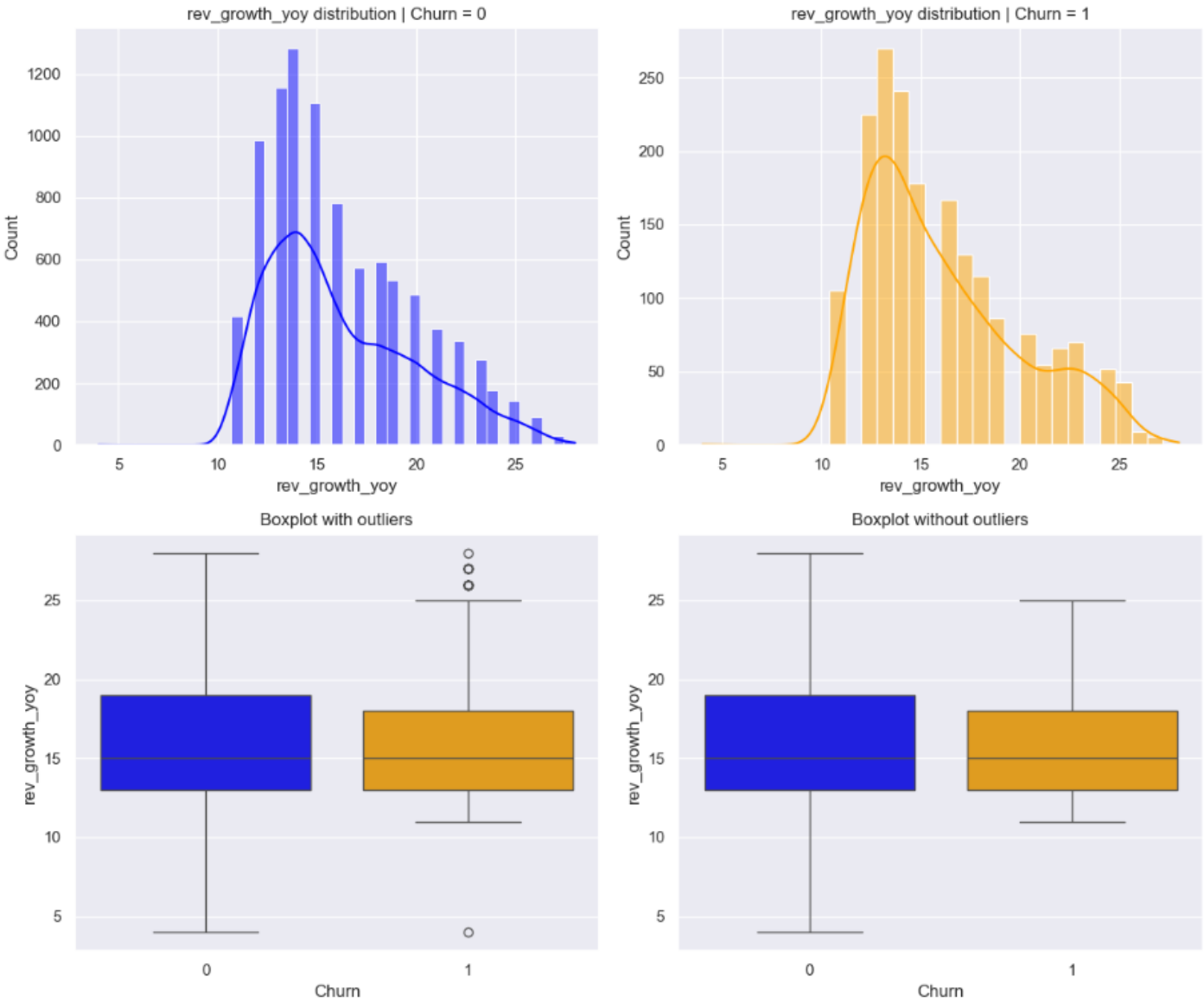
- Customers with accounts generating lower average revenue per month tend to churn more

8. Churn vs Complain in last past year



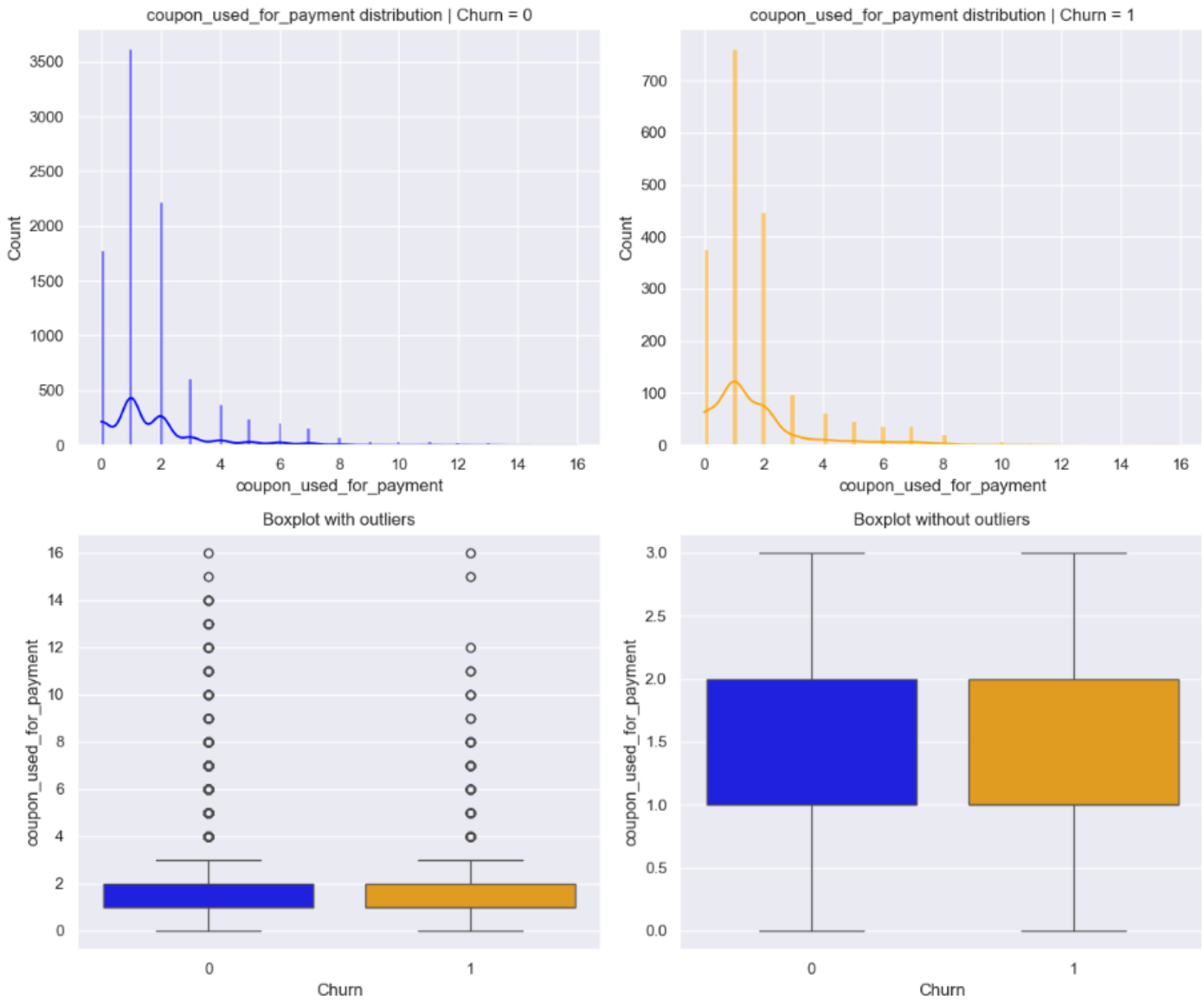
- Higher are the chances for the customers who raise raised a complain to attrite in comparison to those who did not

9. Churn vs revenue growth percentage



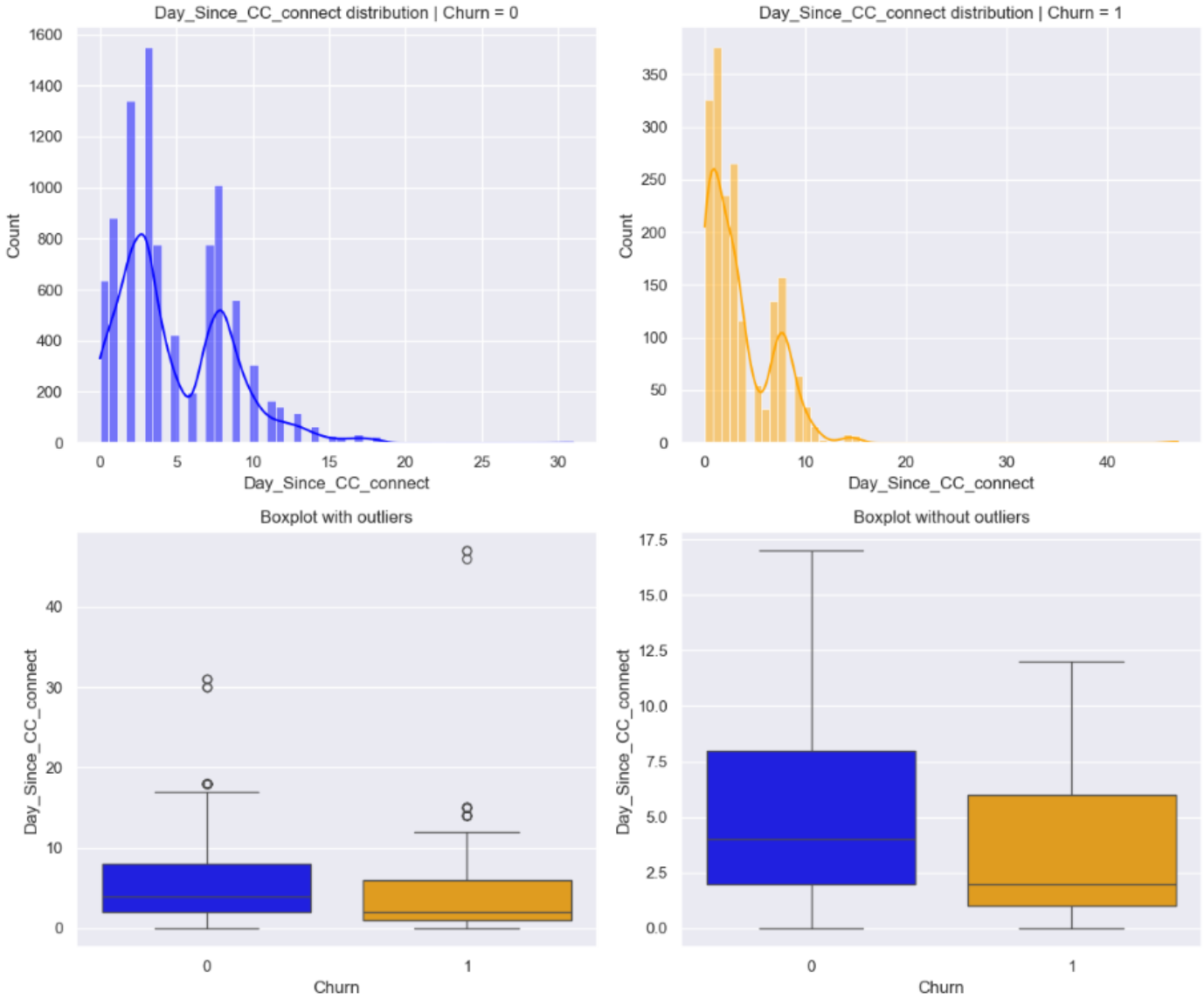
- Lower the revenue growth percentage in the past year in comparison to the previous year, higher the chances for the customer to churn

10. Churn vs coupon used for payment



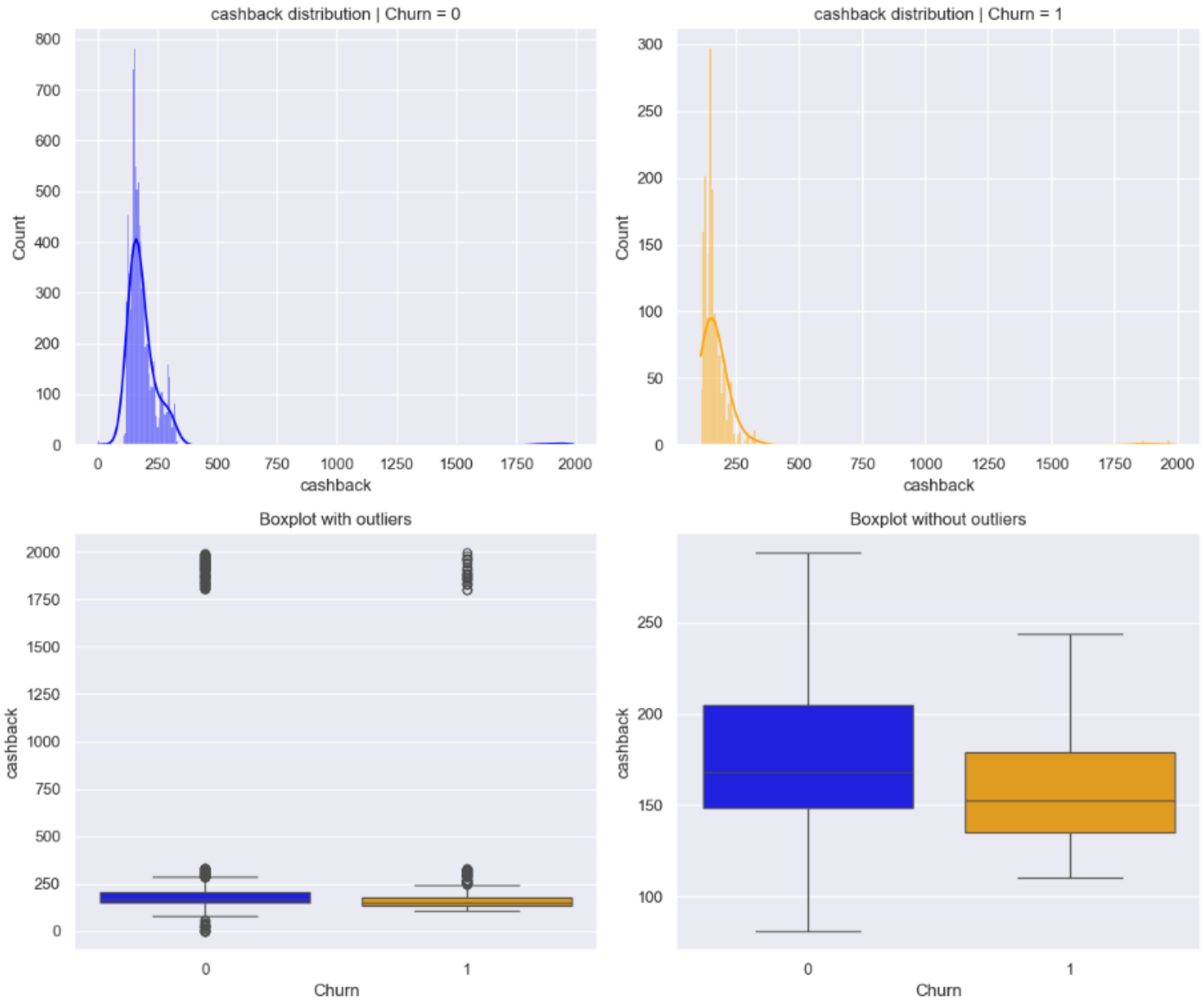
- There's not much difference in the distribution between attrited and non-attrited customer w.r.t the number of coupons used for payment

11. Churn vs Day Since Customer care connect



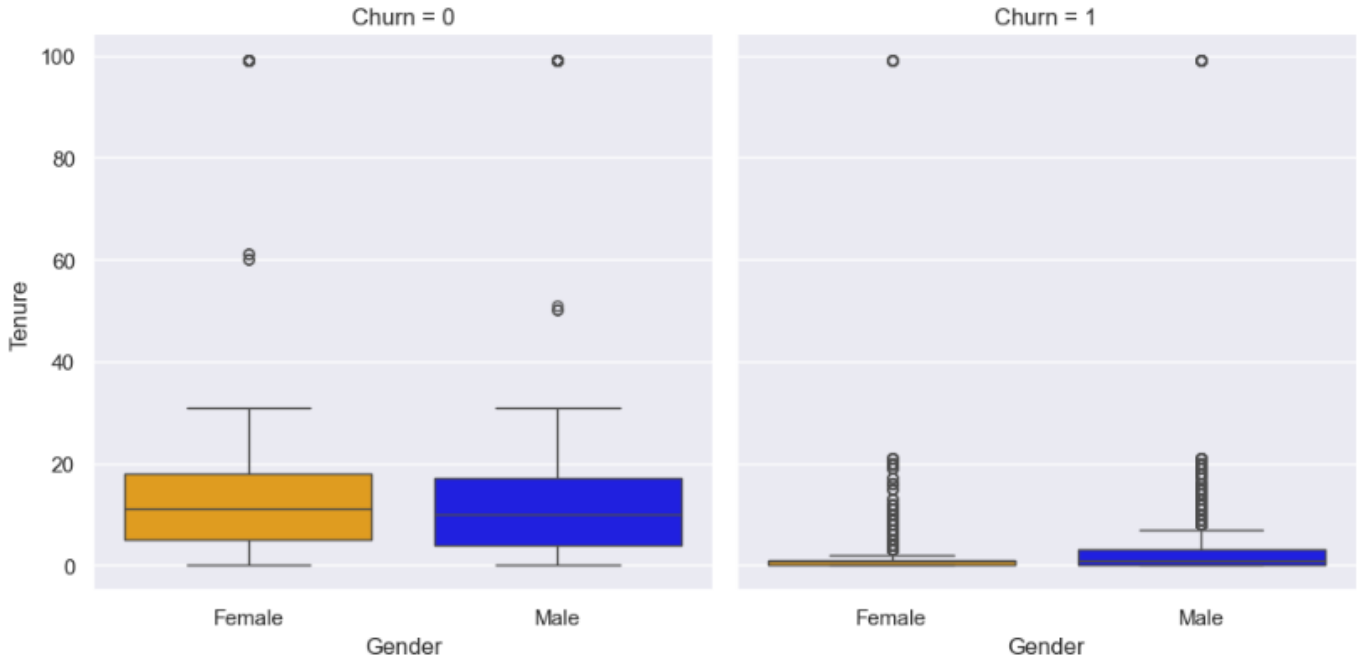
- Users who reached out to the customer care within a week are at a higher risk to churn

12. Churn vs cashback



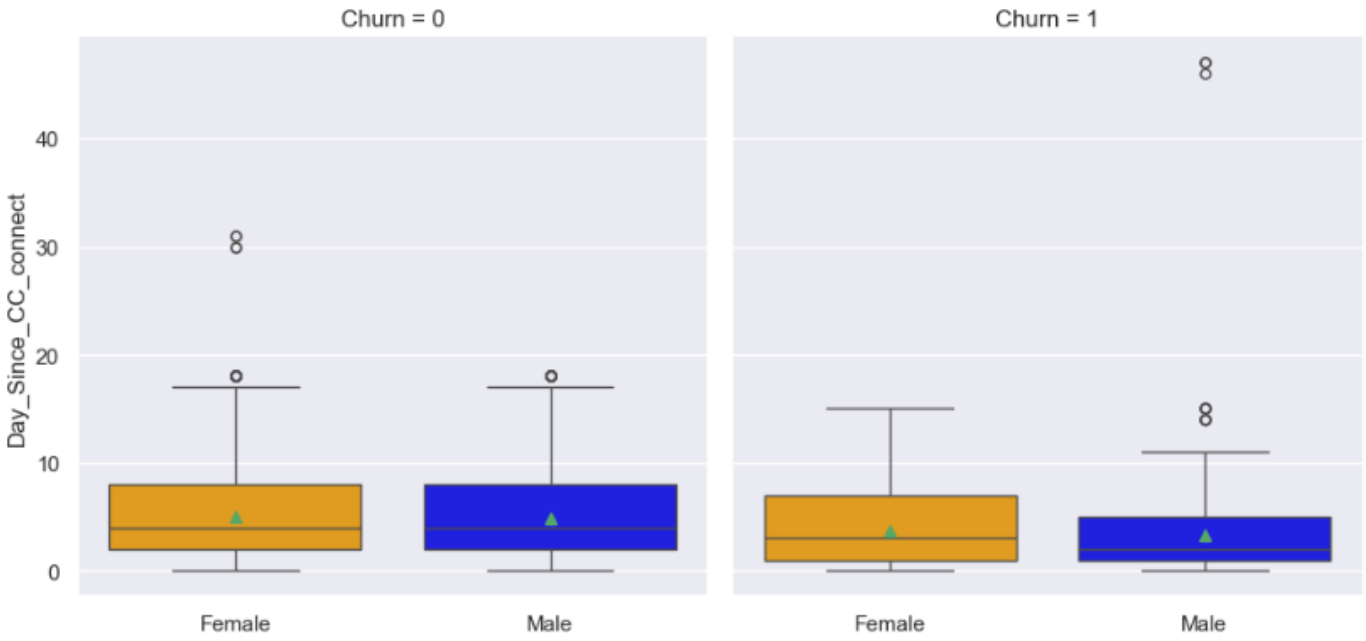
Attrition is higher among customers with monthly average cashback under 200INR

13. Churn vs Tenure vs Gender



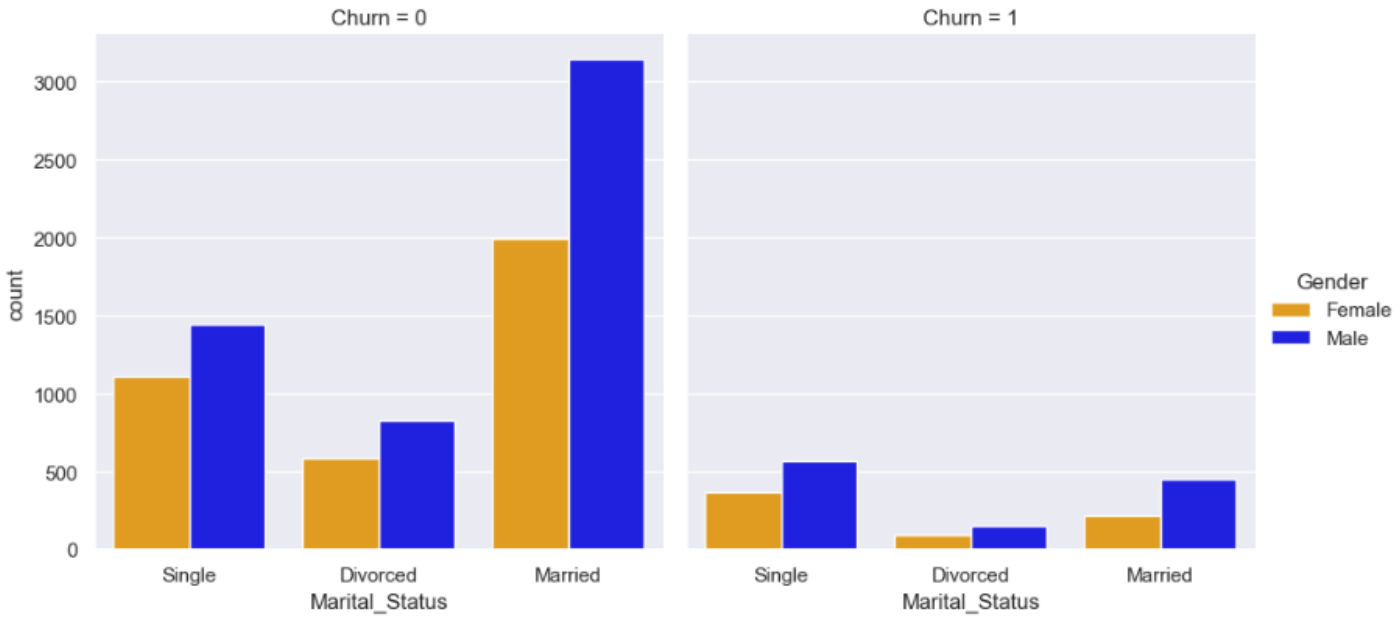
For lower Tenure, the churn rate is higher with Male customers in comparison to Female customers

14. Churn vs Day_Since_CC_connect vs Gender



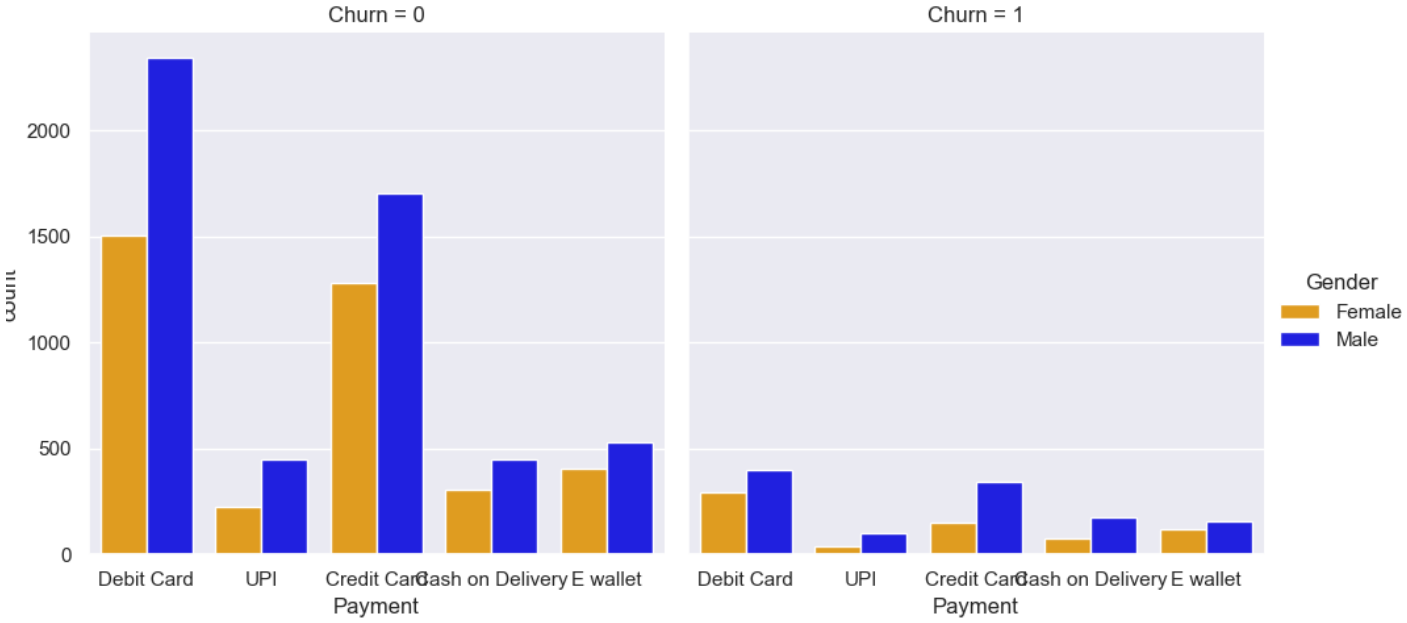
Male customers who call much more frequently than the Female customers churn more

15. Churn vs Marital_Status vs Gender



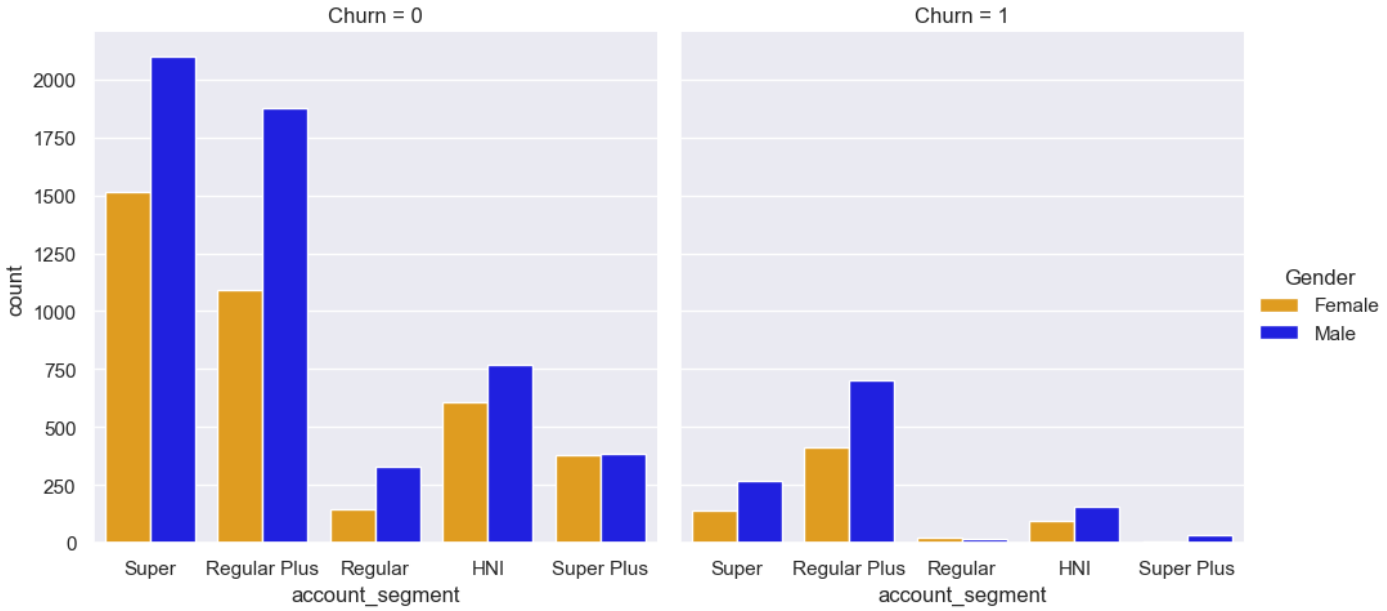
Churn rate is higher among Male customers irrespective of their Marital_Status in comparison to Female customers

16. Churn vs Payment vs Gender



Male customers have churned more than Female customers across different payment methods

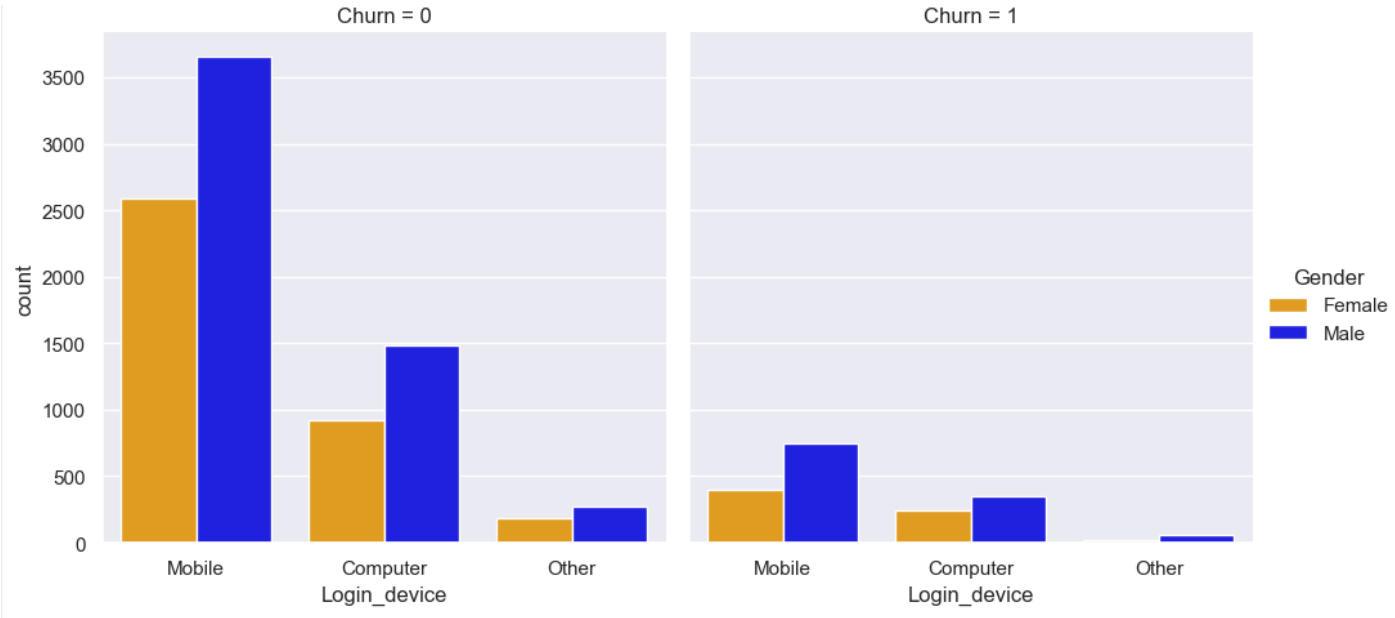
17. Churn vs account_segment vs Gender



For account segment 'Regular' Female customers churn more than Male customers

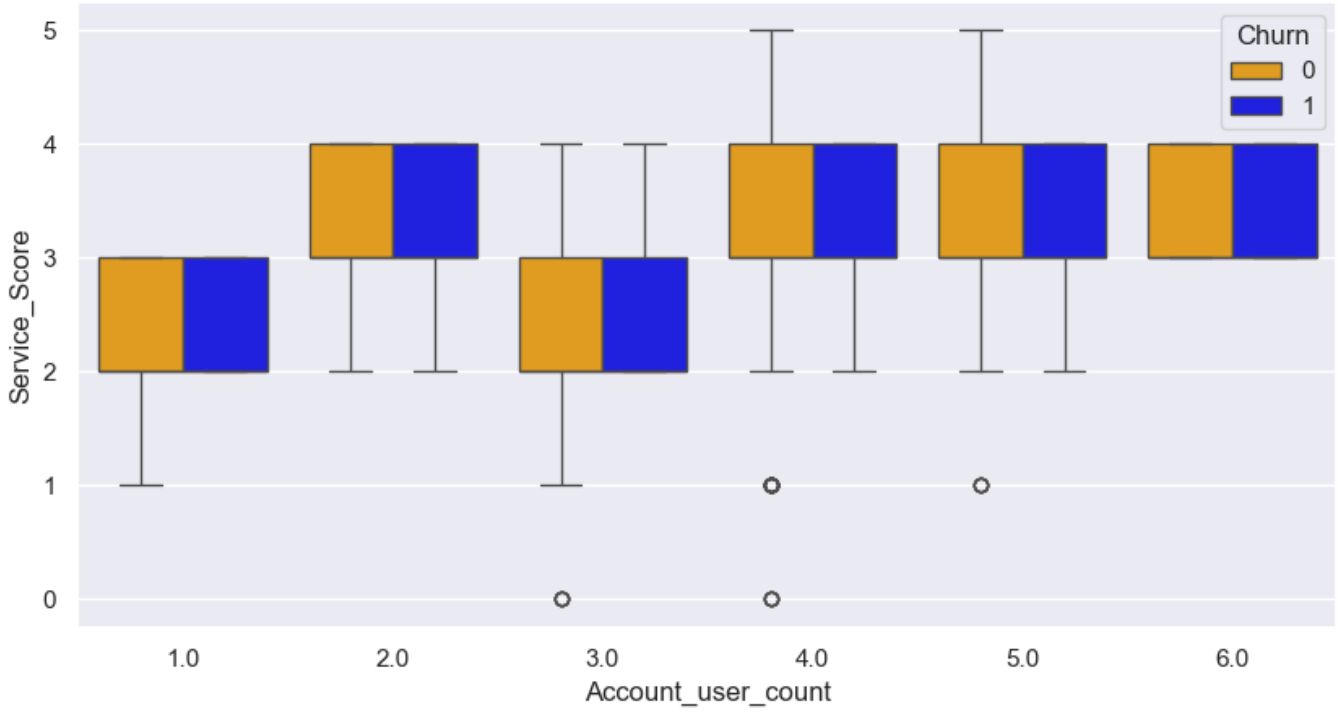
For all other account segments Male customers show higher attrition rate

18. Churn vs Login_device vs Gender



Male customers churn more than Female customers irrespective of the preferred login device

19. Churn vs Account_user_count vs Service_Score



The churn pattern across Account_user_count varies with Service_Score given by the customer

Key Insights from Exploratory Data Analysis (EDA)

The EDA reveals several factors contributing to customer churn, ranging from account characteristics and behavior patterns to demographic and engagement variables. Below is a consolidated summary of insights:

1. Tenure and Churn

- Customers with shorter tenure (less than ~5 months), i.e., newer customers, exhibit significantly higher attrition rates.
- Among these, male customers show a higher churn rate compared to females.

2. City Tier

- The distribution of churn is fairly consistent across city tiers, with no major differences observed between attrited and non-attrited groups.
- However, a slightly higher churn is seen in Tier 1 cities, followed by Tier 3.

3. Customer Service Interaction

- Customers who contacted customer care between **12 to 27 times** in the past year are more likely to churn.
- Those who connected with customer care **within a week** also show a higher risk of attrition.
- The churn risk is elevated for customers who gave **service satisfaction scores of 2, 3, or 4**, and those who gave **agent scores of 3 or more**.
- Male customers tend to contact customer care more frequently and also show a higher churn rate compared to females.

4. Complaints and Feedback

- Customers who raised a complaint in the past year have a higher probability of churning.
- Improvement in customer service and feedback mechanisms is crucial for reducing attrition.

5. Revenue and Cashback

- Customers generating **lower average monthly revenue** tend to churn more.
- A **low revenue growth percentage** compared to the previous year is also associated with higher churn.
- Customers receiving **less than ₹200 as average monthly cashback** are more likely to leave.

6. Account Usage and Segmentation

- Churn is higher among **Regular Plus** account holders.
- Accounts with **3 or more users** are more prone to attrition.
- In the 'Regular' account segment, **female customers** churn more, whereas in other segments, **male customers** have higher attrition rates.

7. Demographics

- Churn is significantly higher among **male and single customers**, regardless of marital status.
- Males show higher churn across different payment modes, login devices, and service interaction categories.

8. Payment Behavior

- Customers who used **coupons less than 2 times** are slightly more likely to churn, although the overall distribution is similar between both churned and non-churned groups.
- Encouraging hassle-free payment methods such as **standing instructions or UPI** could enhance retention.

9. Multi-Variable Interactions

- The impact of **Account_User_Count** on churn varies depending on the **Service_Score** provided.
- Churn patterns become more evident when combining demographic variables (like gender) with behavioral variables (like service interaction and payment mode).

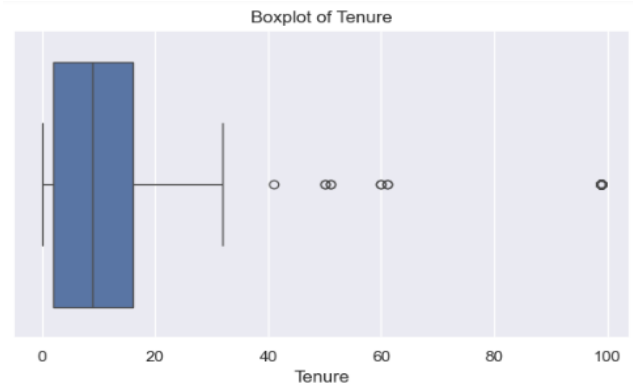
Outlier treatment: -

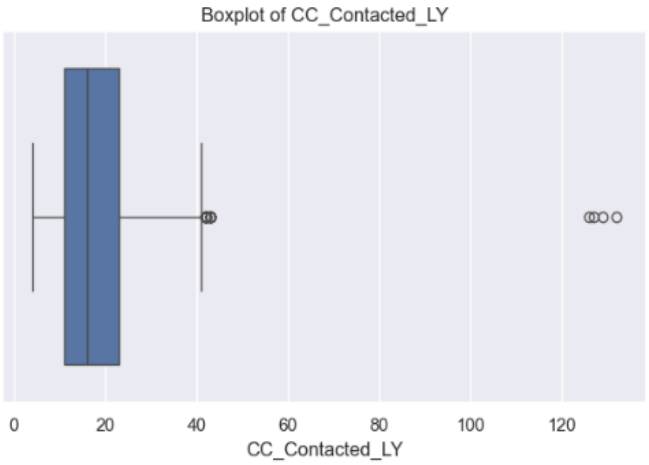
The dataset comprises both continuous and categorical variables. Since each category in a categorical variable represents a distinct customer type, applying outlier treatment to such variables is not meaningful. Therefore, outlier treatment is performed only on continuous variables.

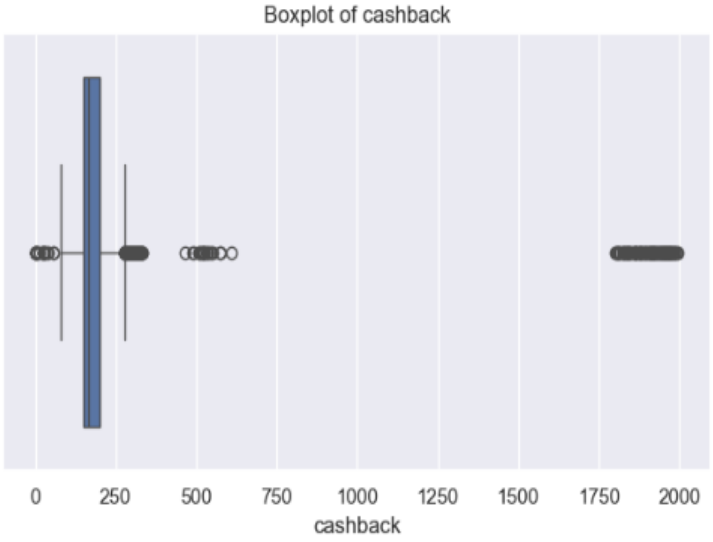
- Box plots were utilized to detect the presence of outliers in the continuous variables.

- Points falling outside the upper and lower bounds (whiskers) of the box plot indicate potential outliers.
- The continuous variables in this dataset include Tenure, CC_Contacted_LY, Account_user_count, cashback, rev_per_month, Day_Since_CC_connect, coupon_used_for_payment, and rev_growth_yoy.
- Outliers were removed using calculated upper and lower thresholds.
- Below is a visual comparison of these variables before and after the outlier treatment.

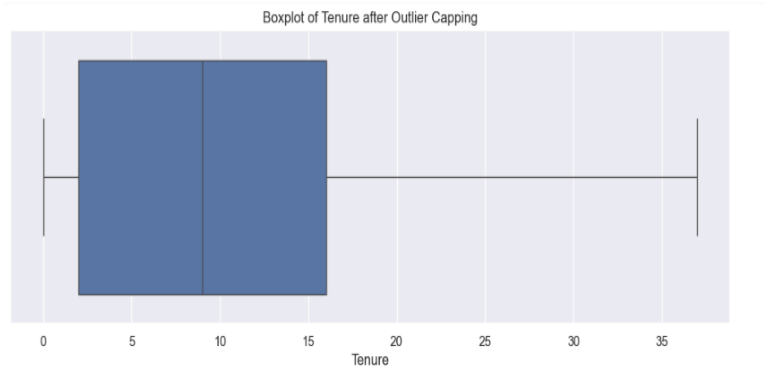
Before

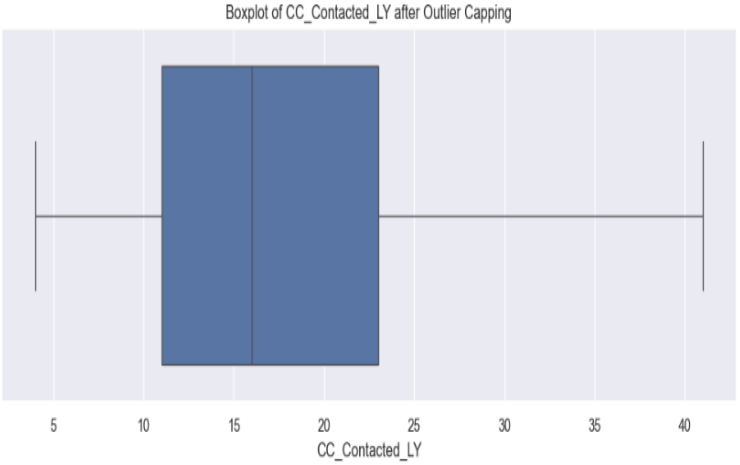


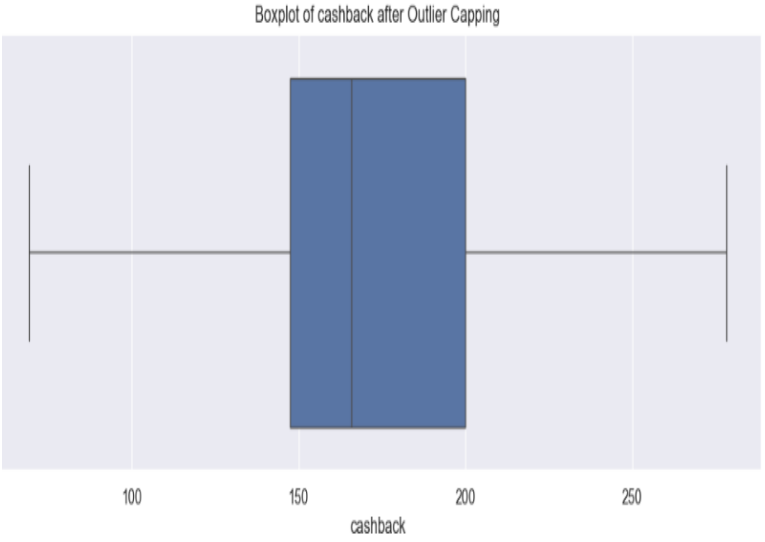


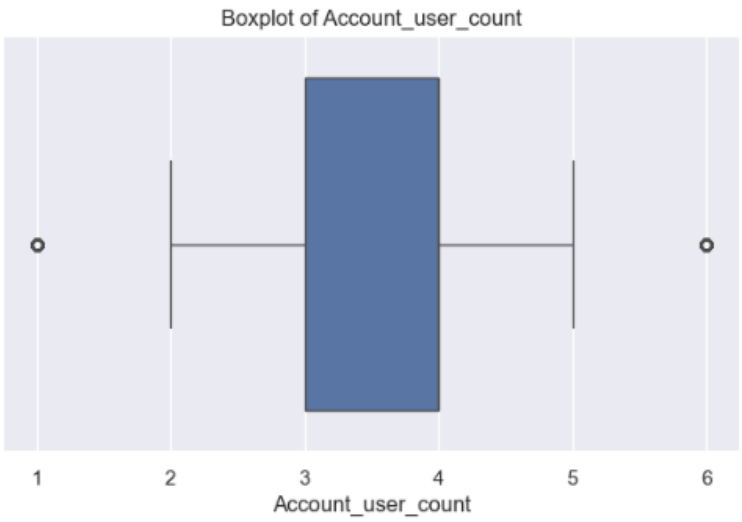


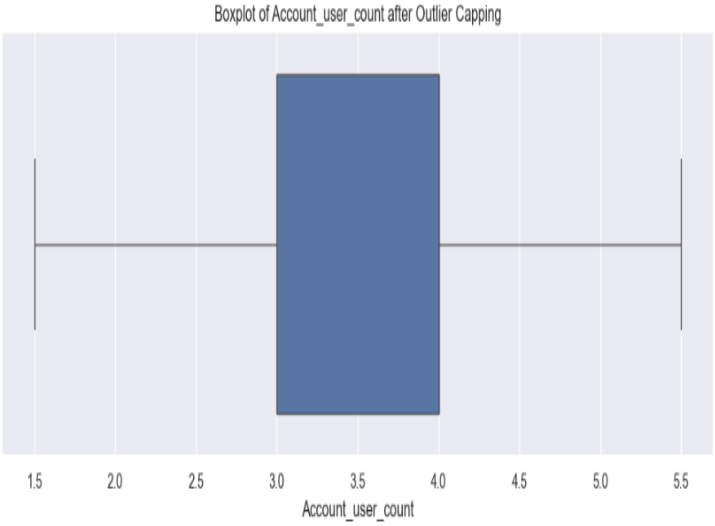
After

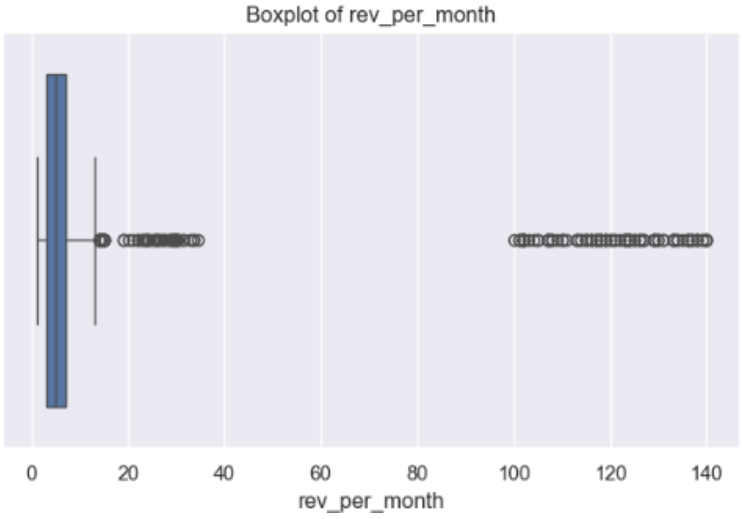


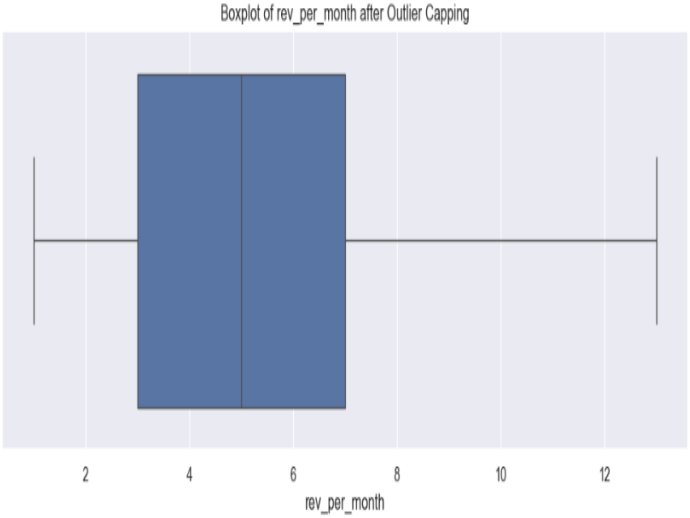




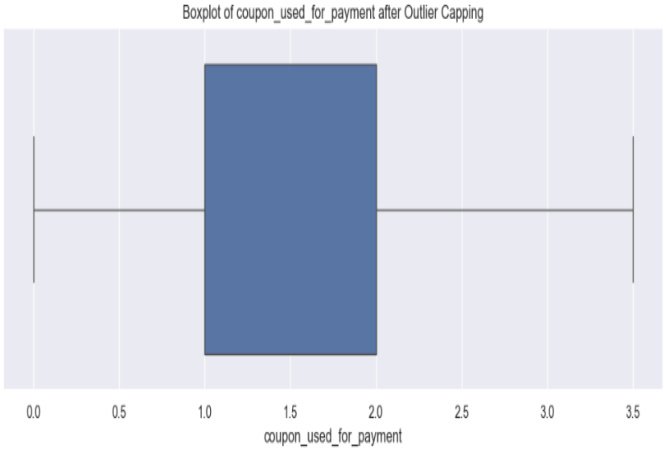

Boxplot of Account_user_count
1
2
3
4
5
6
Account_user_count

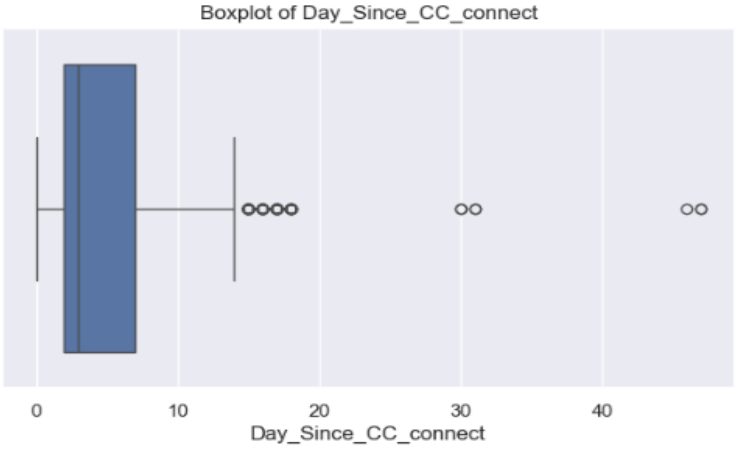

Boxplot of Account_user_count after Outlier Capping
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
Account_user_count

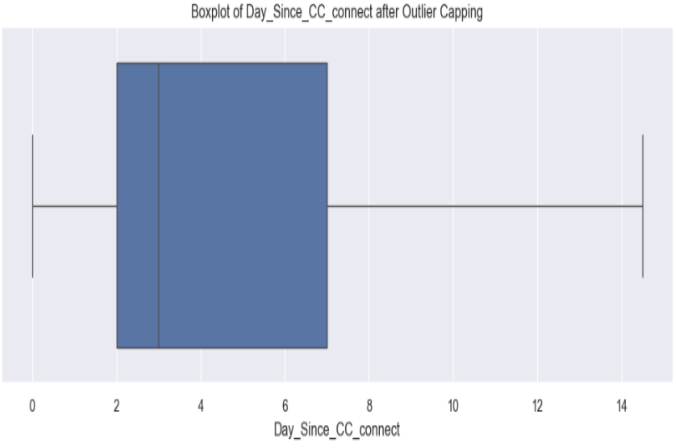

Boxplot of rev_per_month
0
20
40
60
80
100
120
140
rev_per_month


Boxplot of rev_per_month after Outlier Capping
2
4
6
8
10
12
rev_per_month


Boxplot of coupon_used_for_payment
0
2
4
6
8
10
12
14
16
coupon_used_for_payment


Boxplot of coupon_used_for_payment after Outlier Capping
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
coupon_used_for_payment


Boxplot of Day_Since_CC_connect
0
10
20
30
40
Day_Since_CC_connect


Boxplot of Day_Since_CC_connect after Outlier Capping
0
2
4
6
8
10
12
14
Day_Since_CC_connect

Variable Transformation:

- **The variables in the dataset have different units and scales. For example, the variable "Cashback" represents a currency amount, while "CC_Agent_Score" indicates a customer rating. This leads to differences in their statistical values.**
- **Therefore, scaling is necessary to normalize the dataset, which will bring the standard deviation of the variables closer to zero.**
- **To achieve this, Min-Max scaling is applied to normalize the data.**

Before

Churn	0.374
Tenure	12.758
City_Tier	0.913
CC_Contacted_LY	8.815
Payment	1.008
Gender	0.489
Service_Score	0.722
Account_user_count	1.004
account_segment	1.119
CC_Agent_Score	1.373
Marital_Status	0.659
rev_per_month	11.489
Complain_ly	0.447
rev_growth_yoy	3.757
coupon_used_for_payment	1.969
Day_Since_CC_connect	3.650
cashback	174.978
Login_device	0.517
Cluster	1.039
dtype: float64	

After

Churn	0.374
Tenure	0.129
City_Tier	0.456
CC_Contacted_LY	0.069
Payment	0.252
Gender	0.489
Service_Score	0.144
Account_user_count	0.201
account_segment	0.280
CC_Agent_Score	0.343
Marital_Status	0.329
rev_per_month	0.083
Complain_ly	0.447
rev_growth_yoy	0.157
coupon_used_for_payment	0.123
Day_Since_CC_connect	0.078
cashback	0.088
Login_device	0.259
dtype: float64	

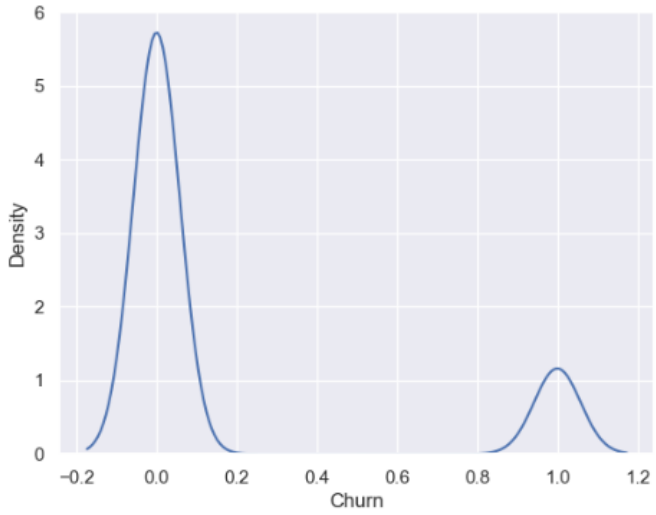
- The standard deviations of the variables are now close to zero.
- Additionally, the variables have been converted to integer data types to facilitate the subsequent model-building process.

Addition of New Variables:

At this stage, there is no need to create new variables. However, new variables may be introduced later during the model-building process as required.

Business insights from EDA

Is the data unbalanced? If so, what can be done? Please explain in the context of the business

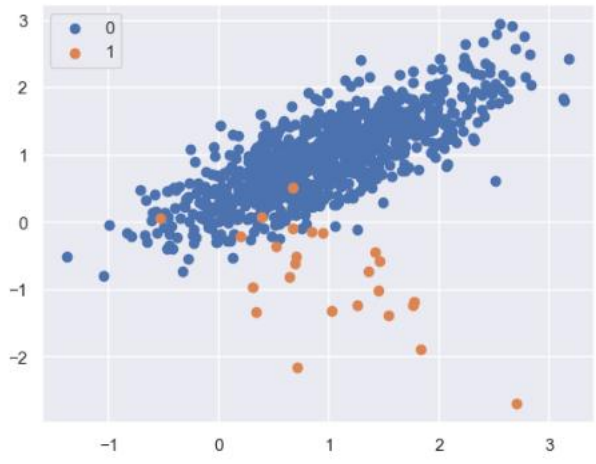



```
Churn
0    9364
1    1896
Name: count, dtype: int64
```

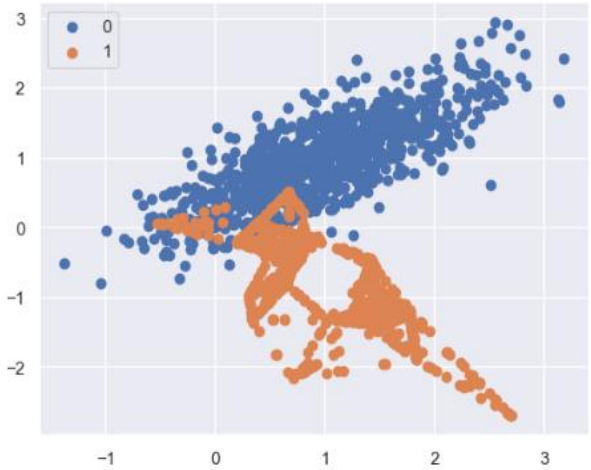
The dataset is imbalanced in nature. The target variable “Churn” shows a significant disparity in class distribution, with 9364 instances of class '0' (non-churn) and 1896 instances of class '1' (churn).

- This imbalance can be addressed using the SMOTE (Synthetic Minority Over-sampling Technique), which generates synthetic data points for the minority class to balance the dataset.
- SMOTE should be applied only to the training dataset, not the test dataset, to avoid data leakage.
- The dataset was split into training and testing sets in a 70:30 ratio, a common industry practice (this ratio can be adjusted if required).

Before



After



- The increase in density of the orange dots indicates the increase in data points.

Any business insights using clustering

Clustering Analysis

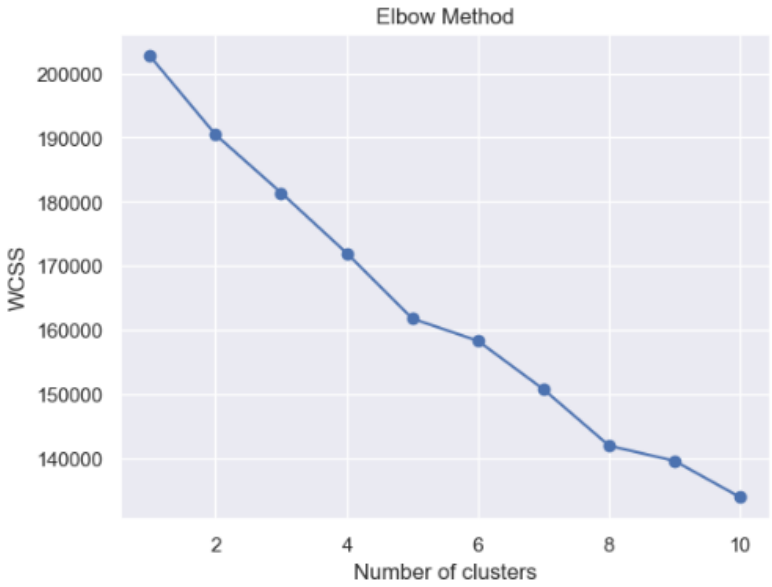
To better understand customer segments and their relationship with churn, clustering analysis was performed on the dataset.

Data Preparation:

Categorical variables were first encoded into numeric values to enable clustering algorithms to

process them effectively. Following encoding, all features were standardized using z-score normalization to ensure all variables contributed equally without bias due to scale differences.

Determining the Number of Clusters:



The K-Means clustering algorithm was applied with a range of cluster counts from 1 to 10. The optimal number of clusters was determined using the Elbow Method, which analyzes the within-cluster sum of squares (WCSS). The elbow plot suggested 5 clusters as the ideal choice, balancing cluster compactness and model simplicity.

Clustering Results and Insights:

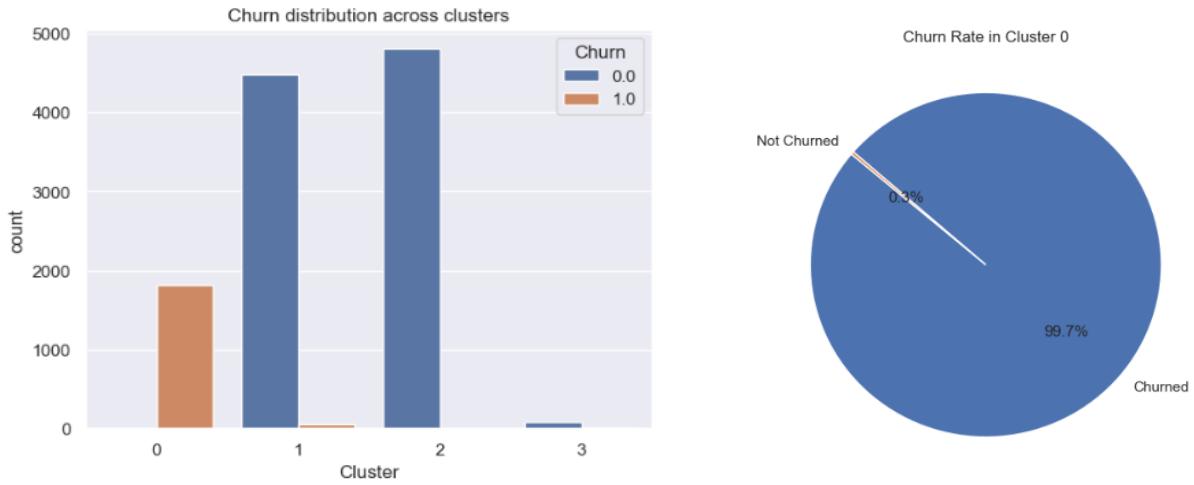


Fig 3. Clustering results

The dataset was segmented into 5 clusters. Analysis of churn distribution within these clusters revealed significant variation. Notably, Cluster 0 showed an exceptionally high churn rate, with approximately 99.97% of customers marked as churned. This indicates that customers in this cluster share characteristics strongly associated with churn and could be prioritized for retention efforts.

This cluster-based segmentation provides valuable insights to tailor targeted marketing and customer retention strategies more effectively.

Additional Business Insights:

- The dataset shows a healthy variation, capturing a mix of services availed, customer ratings, and customer profile information.
- The business should focus on increasing visibility in Tier 2 cities to expand its customer base.
- Promoting payment options like standing instructions through bank accounts or UPI can offer customers a more convenient and secure experience.
- There is room for improvement in service scores, indicating unexplored areas in customer satisfaction. Conducting customer surveys could help better understand their expectations.
- Training customer care executives to enhance their interaction quality may lead to improved customer feedback and satisfaction scores.
- The business can design personalized plans for customers, not just based on their spending patterns, but also considering their tenure with the company.
- Introducing customized plans for married customers—such as family floater packages—could add value and boost customer retention.

End of Project Note – 1

PROJECT NOTE – II

Model Building

In this phase of the capstone project, we proceed with building various models after completing the Exploratory Data Analysis (EDA) and data cleaning steps. This will be followed by model tuning and evaluation using performance metrics such as Accuracy, F1 Score, Recall, Precision, ROC Curve, AUC Score, Confusion Matrix, and Classification Report. Our goal is to select a model that neither underfits nor overfits, while also delivering the highest possible accuracy.

Data Splitting: Train and Test Sets

As per standard industry practice, the dataset has been split into training and testing sets in a 70:30 ratio. The models are trained on the training dataset and evaluated for their accuracy and performance on the test dataset.

Below is the shape of Train and Test dataset: -

```
X_train (7882, 17)
X_test (3378, 17)
y_train (7882,)
y_test (3378,)
```

Fig 38. Shape of training and test data

Building Logistic Regression Model on Dataset: -

Building the Model with Default Hyperparameters:

After splitting the data into training and testing sets, we fitted a Logistic Regression model to the training dataset and made predictions on both the training and testing data. The initial model was built using default hyperparameters, with the solver set to 'lbfgs' by default.

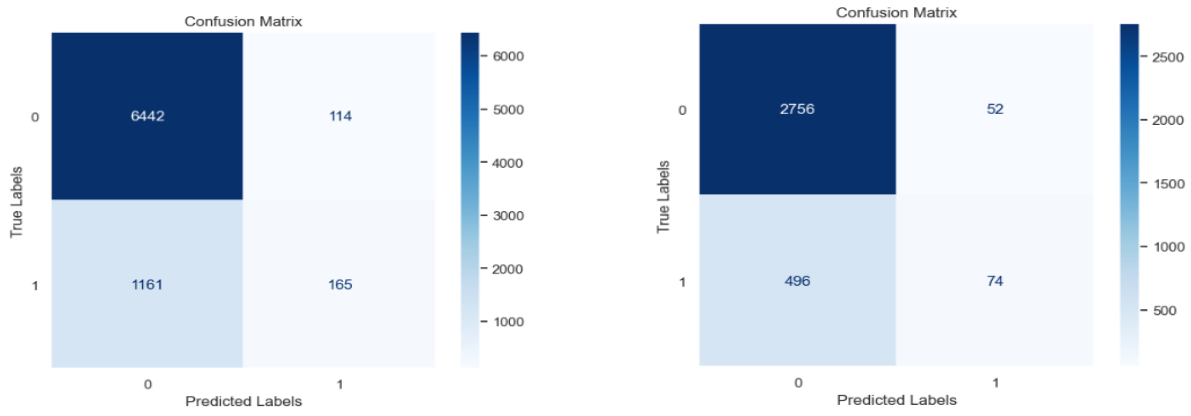
The accuracy scores obtained from this baseline model are as follows:

```
Accuracy of training dataset: 0.8382390256280132

Accuracy of testing dataset: 0.8377738306690349
```

Fig 39. Accuracy From Logistic Regression

Below is the confusion matrix obtained from this model: -



Train **Test**

Fig 40: - Confusion Matrix From Logistic Regression

Below is the classification report obtained from this model: -

Classification Report (Train):

	precision	recall	f1-score	support
0.0	0.847	0.983	0.910	6556.000
1.0	0.591	0.124	0.206	1326.000
accuracy	0.838	0.838	0.838	0.838
macro avg	0.719	0.554	0.558	7882.000
weighted avg	0.804	0.838	0.791	7882.000

Classification Report (Test):

	precision	recall	f1-score	support
0.0	0.847	0.981	0.910	2808.000
1.0	0.587	0.130	0.213	570.000
accuracy	0.838	0.838	0.838	0.838
macro avg	0.717	0.556	0.561	3378.000
weighted avg	0.804	0.838	0.792	3378.000

Fig 41: - Clasification Report From Logistic Reression

Below is the AUC score and ROC curve obtained from this model: -

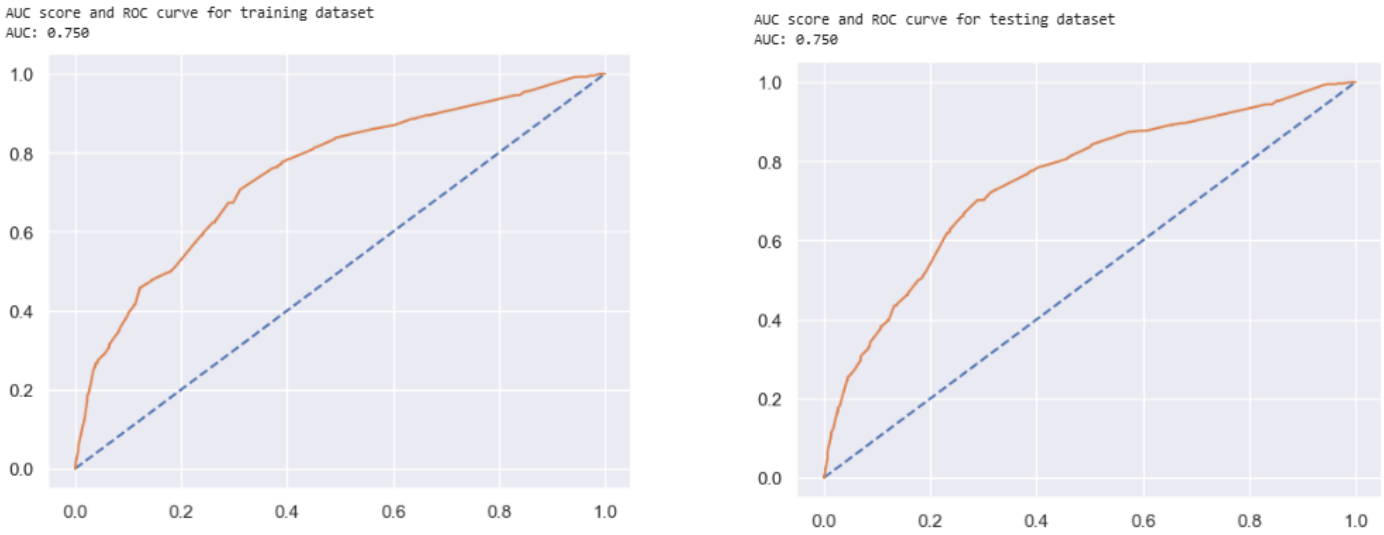


Fig 42: - ROC Curve & AUC Score From Logistic Regression

To validate the reliability of the model, we performed 10-fold cross-validation using the Logistic Regression model with default settings. Cross-validation helps assess the model's generalizability and ensures it is not overfitting or underfitting the data. The cross-validation scores are presented below:

```
cross validation scroes for traning dataset
array([0.83269962, 0.8365019 , 0.84771574, 0.84390863, 0.84137056,
       0.83756345, 0.84010152, 0.83121827, 0.83883249, 0.83629442])

cross calidation scores for testing dataset
array([0.82248521, 0.84023669, 0.84023669, 0.82840237, 0.84911243,
       0.84615385, 0.83727811, 0.83727811, 0.83976261, 0.84272997])
```

Fig 43: - Cross Validation Scores from Logistic Regression

we can observe that the cross validations scores are almost same for all the folds. Which

indicates that the model built is correct.

Building model using GridSearchCV and analysing the best parameters: -

To identify the optimal set of hyperparameters that minimizes the loss function and improves model performance, we utilized GridSearchCV. Various hyperparameters such as "solver," "penalty," and

"tol" were tested. The best combination for this dataset was found to be the "lbfgs" solver with "none" penalty. However, it was observed that the improvement in accuracy between the training and testing datasets was marginal and not significantly impactful.

Below are the accuracy scores obtained from this model using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8382390256280132

Accuracy of testing dataset after gridsearchCV: 0.8377738306690349
```

Fig 44: - Accuracy From Logistic Regression Using hyper-parameter

Below is the classification report obtained from this model using GridSearchCV: -

```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.59      0.12      0.21      1326

    accuracy                           0.84      7882
   macro avg       0.72      0.55      0.56      7882
weighted avg       0.80      0.84      0.79      7882

Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.59      0.13      0.21       570

    accuracy                           0.84      3378
   macro avg       0.72      0.56      0.56      3378
weighted avg       0.80      0.84      0.79      3378
```

Fig 45: - Classification Report from Logistic Regression Using hyper-parameter

Below is the confusion matrix obtained from this model using GridSearchCV: -

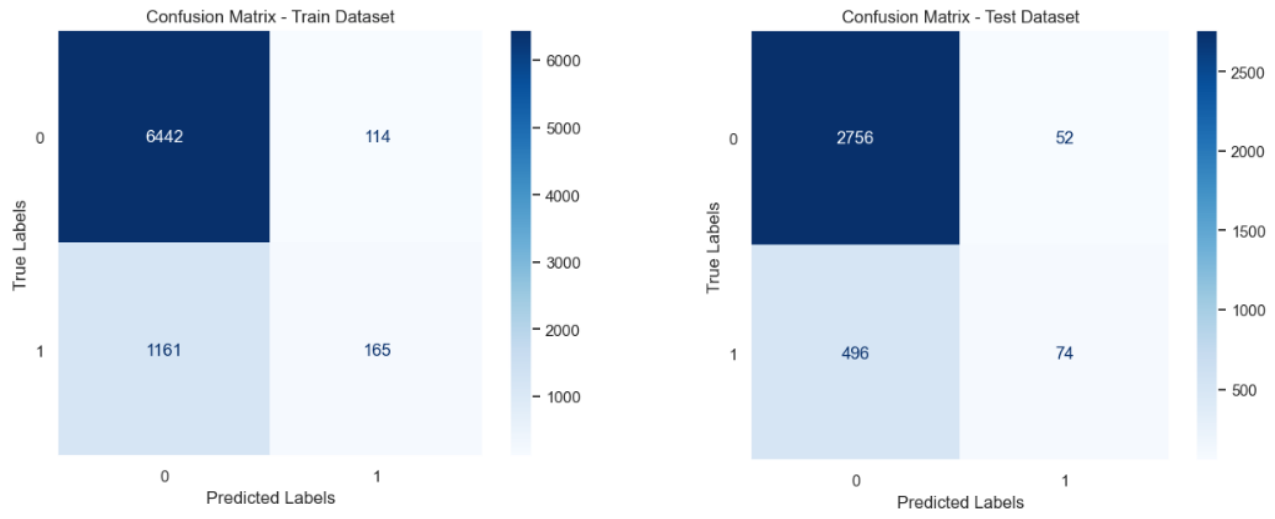


Fig 46: - Confusion Matrix from Logistic Regression Using hyper-parameter

Below is the AUC score and ROC curve obtained from this model using GridSearchCV: -

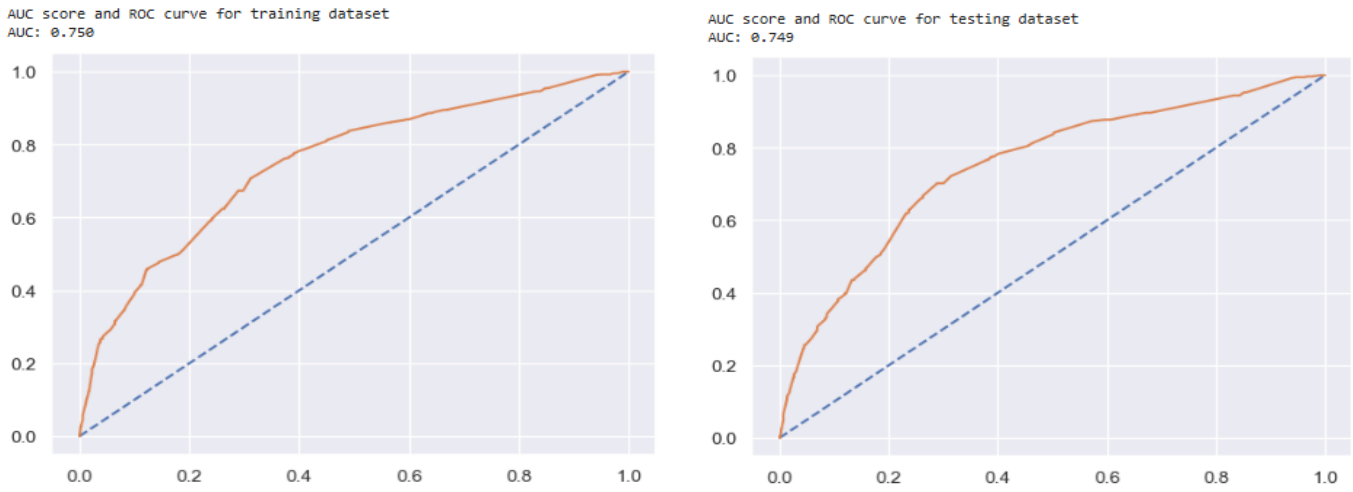


Fig 47: - ROC Curve and AUC Score from Logistic Regression Using hyper-parameter

Below are the 10-fold cross validation scores: -

```
cross validation score for training dataset
array([0.83269962, 0.8365019 , 0.84771574, 0.84390863, 0.84137056,
       0.83756345, 0.84010152, 0.83121827, 0.83883249, 0.83629442])

cross calidation score for testing dataset
array([0.82248521, 0.84023669, 0.84023669, 0.82840237, 0.84911243,
       0.84615385, 0.83727811, 0.83727811, 0.83976261, 0.84272997])
```

Fig 48: - Cross Validation Scores from Logistic Regression Using hyper-parameter

Building model using SMOTE: -

As observed in our earlier analysis, the dataset is imbalanced. To address this, we applied the SMOTE (Synthetic Minority Over-sampling Technique) method to balance the class distribution. We then rebuilt the model using the balanced dataset to evaluate whether this leads to any notable improvement in performance. However, upon evaluating the model's accuracy on both training and testing datasets, we found that the improvement was not significant.

Below are the accuracy scores obtained from balanced data: -

```
Accuracy of training dataset: 0.6908175716900549

Accuracy of testing dataset: 0.6524570751924216
```

Fig 49: - Accuracy Score from Logistic Regression with SMOTE

Below is the confusion matrix obtained from balanced data: -

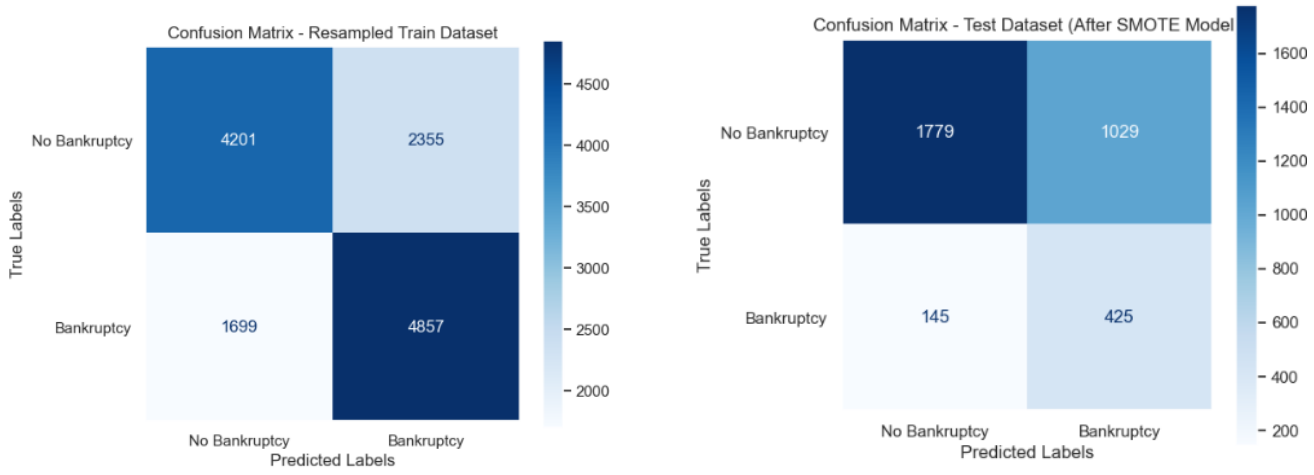


Fig 50: - Confusion Matrix from Logistic Regression with SMOTE

Below is the classification report obtained from balanced data: -

Classification report for train dataset

	precision	recall	f1-score	support
0.0	0.71	0.64	0.67	6556
1.0	0.67	0.74	0.71	6556
accuracy			0.69	13112
macro avg	0.69	0.69	0.69	13112
weighted avg	0.69	0.69	0.69	13112

Classification report for test dataset

	precision	recall	f1-score	support
0.0	0.92	0.63	0.75	2808
1.0	0.29	0.75	0.42	570
accuracy			0.65	3378
macro avg	0.61	0.69	0.59	3378
weighted avg	0.82	0.65	0.70	3378

Fig 51: - Classification Report from Logistic Regression with SMOTE

Below are the AUC scores and ROC curve obtained from balanced data: -

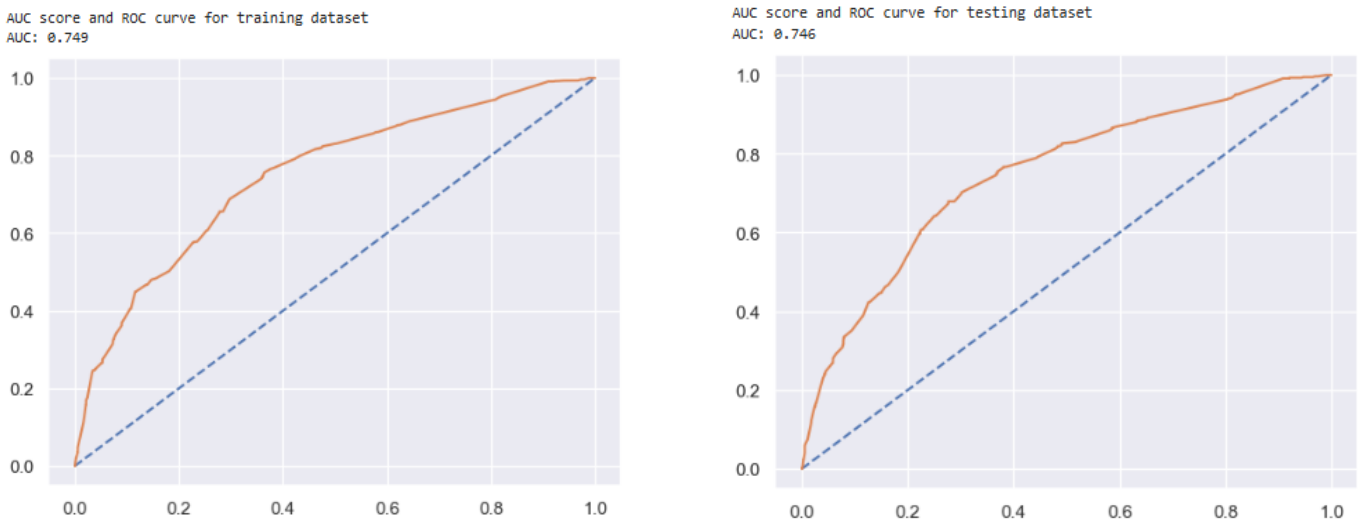


Fig 52: - ROC Curve & AUC Scores From Logistic Regression with SMOTE

Below are the 10-fold cross validation scores: -

```
cross validation score for balanced training dataset
array([0.67911585, 0.68368902, 0.6819222 , 0.70022883, 0.68726163,
       0.71777269, 0.67658276, 0.68421053, 0.69870328, 0.70022883])

cross validation score for testing dataset
array([0.82248521, 0.84023669, 0.84023669, 0.82840237, 0.84911243,
       0.84615385, 0.83727811, 0.83727811, 0.83976261, 0.84272997])
```

Fig 53: - Cross Validation Scores From Logistic Regression with SMOTE

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from Logistic Regression Model:
Based on the analysis, it can be concluded that the model is neither overfitting nor underfitting. The Logistic Regression model tuned using GridSearchCV is the most optimized version in terms of hyperparameter selection. However, the overall performance—measured by accuracy, recall, precision, F1 score, ROC curve, and AUC—did not show substantial improvement over the default model or the one trained on SMOTE-balanced data.
While the SMOTE-based model performed well on the training data, its accuracy dropped significantly on the testing data, indicating potential overfitting when applied to the balanced dataset.

Building Linear Discriminant Analysis Model (LDA)

Building model with default hyperparameters: -

Using the previously split training and testing datasets, we now build a Linear Discriminant Analysis (LDA) model to evaluate whether it can outperform the Logistic Regression model and be considered a better choice for future predictions. Initially, the LDA model is constructed using its default parameters, with the solver set to “svd” and shrinkage set to “none.”

Below are the accuracy scores obtained from this model: -

```
Accuracy score of training dataset: 0.8439482364882009

Accuracy score of testing dataset: 0.8377738306690349
```

Fig 54: - Accuracy From LDA

Below is the confusion matrix obtained from this model: -

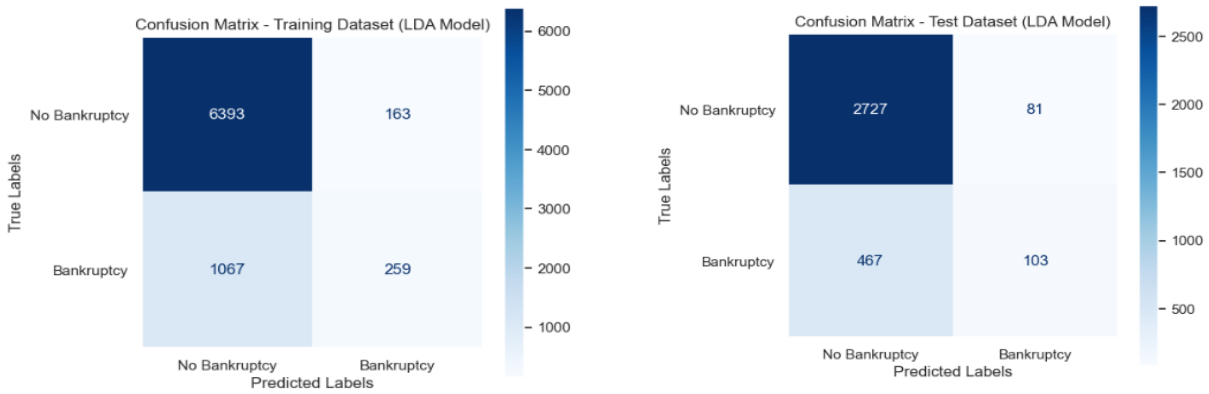


Fig 55: - Confusion Matrix From LDA

Below is the classification report obtained from this model: -

Classification Report of the training data:

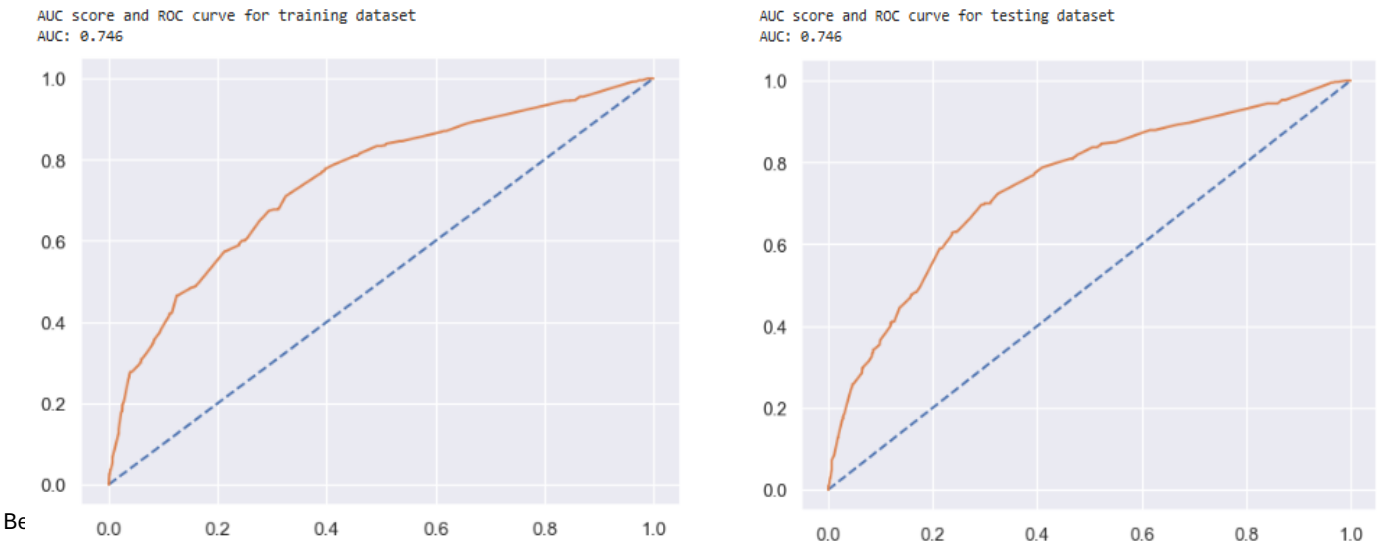
	precision	recall	f1-score	support
0.0	0.86	0.98	0.91	6556
1.0	0.61	0.20	0.30	1326
accuracy			0.84	7882
macro avg	0.74	0.59	0.60	7882
weighted avg	0.82	0.84	0.81	7882

Classification Report of the test data:

	precision	recall	f1-score	support
0.0	0.85	0.97	0.91	2808
1.0	0.56	0.18	0.27	570
accuracy			0.84	3378
macro avg	0.71	0.58	0.59	3378
weighted avg	0.80	0.84	0.80	3378

Fig 56: - Classification Report From LDA

Below are the AUC scores and ROC curves obtained from this model: -



Be

Fig 57. ROC Curve and AUC Score From LDA

Below are the 10-fold cross validation scores: -

```
cross validation score for training dataset
array([0.83269962, 0.84157161, 0.85279188, 0.84771574, 0.84010152,
       0.84390863, 0.84137056, 0.84137056, 0.84771574, 0.84263959])

cross validation score for testing dataset
array([0.81656805, 0.82840237, 0.83727811, 0.82840237, 0.85207101,
       0.83727811, 0.83727811, 0.85207101, 0.83976261, 0.83086053])
```

Fig 58: - Cross Validation Score From LDA

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using GridSearchCV and analysing the best parameters: -

Using GridSearchCV function we tried finding the best parameters to further tune-in the above model for better accuracy and we have find that shrinkage as "auto", solver as "lsqr" and tol value of "0.001" gives the best model considering accuracy, precision,recall, F1, ROC curve and AUC score.

Below are the accuracy scores obtained from model built using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8454706927175843

Accuracy of testing dataset after gridsearchCV: 0.8362936648904677
```

Fig 59: - Accuracy Score From LDA with Hypertuning

Below is the confusion matrix obtained from model built using GridSearchCV: -

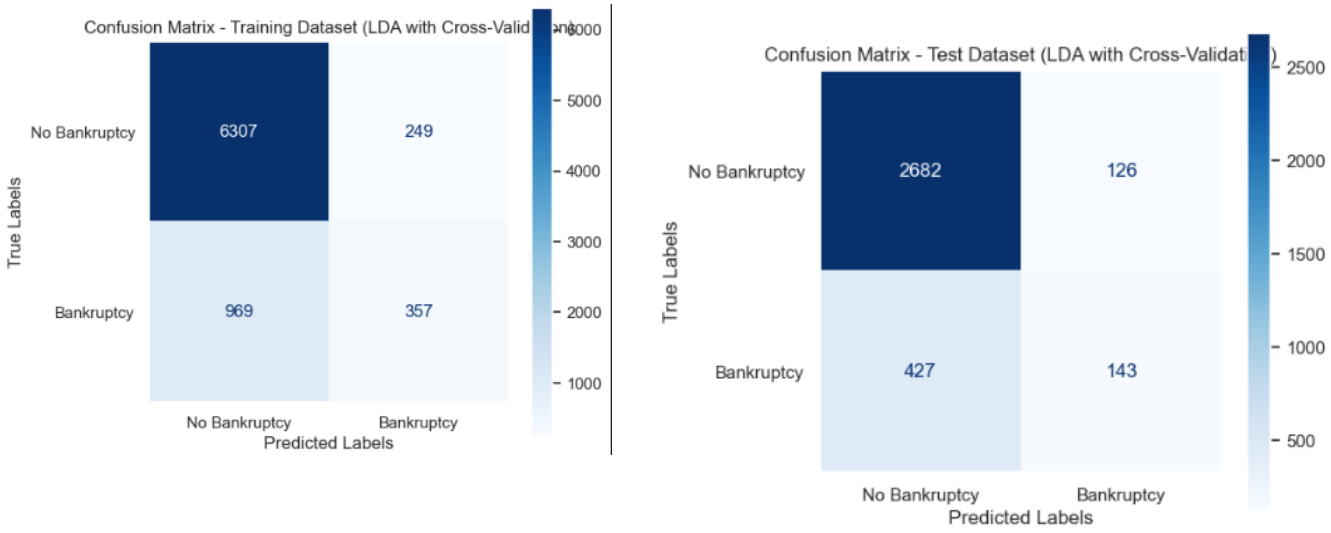


Fig 68: - Confusion Matrix From LDA with Hypertuning

Below are the classification reports obtained from model built using GridSearchCV: -

Classification report for train dataset

	precision	recall	f1-score	support
0.0	0.87	0.96	0.91	6556
1.0	0.59	0.27	0.37	1326
accuracy			0.85	7882
macro avg	0.73	0.62	0.64	7882
weighted avg	0.82	0.85	0.82	7882

Classification report for test dataset

	precision	recall	f1-score	support
0.0	0.86	0.96	0.91	2808
1.0	0.53	0.25	0.34	570
accuracy			0.84	3378
macro avg	0.70	0.60	0.62	3378
weighted avg	0.81	0.84	0.81	3378

Fig 69: - Classification Report from LDA with Hypertuning

Below are the ROC curve and AUC scores obtained from model built using GridSearchCV: -

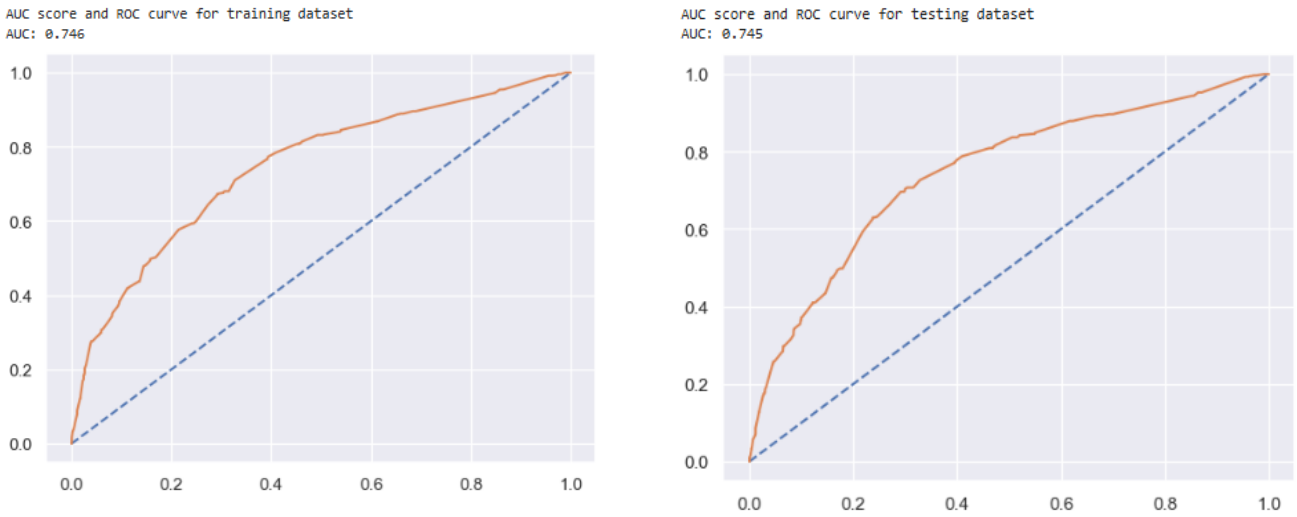


Fig 70: - ROC Curve and AUC Score From LDA with Hypertuning

Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset
array([0.83396705, 0.83776933, 0.85786802, 0.85152284, 0.83756345,
       0.85152284, 0.84517766, 0.84137056, 0.85152284, 0.84517766])
 cross validation scores from testing dataset
 array([0.81656805, 0.82840237, 0.83727811, 0.82840237, 0.85207101,
        0.83727811, 0.83727811, 0.85207101, 0.83976261, 0.83086053])
```

Fig 71: - Cross Validartion Scores From LDA with Hypertuning

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building LDA model using SMOTE: -

From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied SMOTE technique to oversample the data and to obtain a balanced dataset.

Below are the accuracy scores obtained from balanced dataset: -

```
Accuracy of training dataset: 0.6957748627211714

Accuracy of testing dataset: 0.6992303137951451
```

Fig 72: - Accuracy From LDA with SMOTE

Below is the confusion matrix obtained from balanced dataset: -

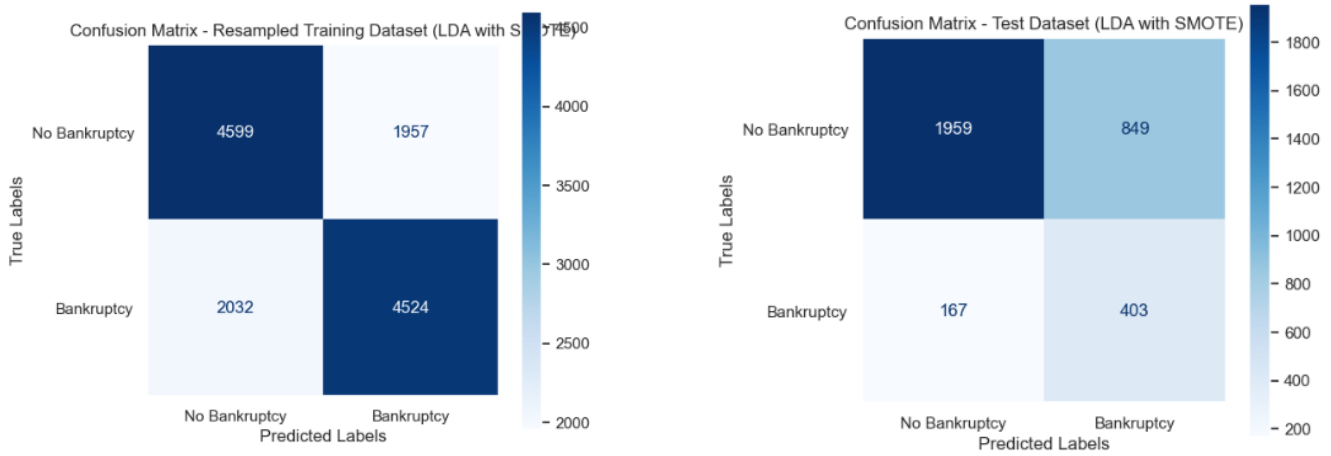


Fig 73: - Confusion Matrix from LDA with SMOTE

Below is the classification report obtained from balanced dataset: -

Classification report for train dataset

	precision	recall	f1-score	support
0.0	0.69	0.70	0.70	6556
1.0	0.70	0.69	0.69	6556
accuracy			0.70	13112
macro avg	0.70	0.70	0.70	13112
weighted avg	0.70	0.70	0.70	13112

Classification report for test dataset

	precision	recall	f1-score	support
0.0	0.92	0.70	0.79	2808
1.0	0.32	0.71	0.44	570
accuracy			0.70	3378
macro avg	0.62	0.70	0.62	3378
weighted avg	0.82	0.70	0.73	3378

Fig 74: - Classification Report From LDA with SMOTE

Below are the ROC curve and AUC scores obtained from balanced dataset: -

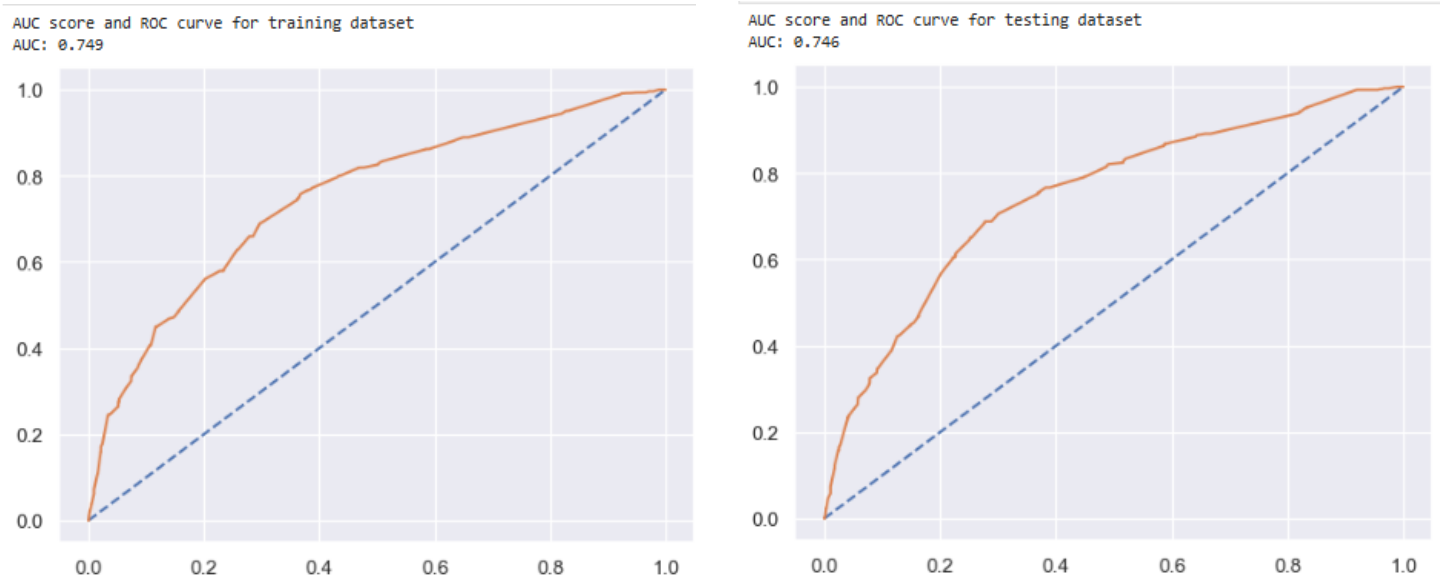


Fig 75: - ROC Curve and AUC Score from LDA with SMOTE

Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset
array([0.66615854, 0.6882622 , 0.67353166, 0.70938215, 0.67963387,
       0.71853547, 0.67963387, 0.69336384, 0.6979405 , 0.69565217])

 cross validation scores for testing dataset
]: array([0.81656805, 0.82840237, 0.83727811, 0.82840237, 0.85207101,
       0.83727811, 0.83727811, 0.85207101, 0.83976261, 0.83086053])
```

Fig 76: - Cross Validation Scores from LDA with SMOTE

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from LDA Model:

The analysis indicates that the LDA model does not suffer from overfitting or underfitting. While the model tuned using GridSearchCV is optimized in terms of parameter selection, the overall performance—measured by accuracy, recall, precision, F1 score, ROC curve, and AUC—is not significantly better compared to the default model or the one built using GridSearchCV. Additionally, the LDA model trained on SMOTE-balanced data performs well on the training dataset, but shows a noticeable drop in accuracy on the test dataset, suggesting overfitting when applied to the balanced data.

BUILDING KNN MODEL: -

Building model with default hyperparameters: -

After splitting the data into training and testing sets, we trained a K-Nearest Neighbors (KNN) model on the training dataset and made predictions on both the training and testing data. The initial model was built using default hyperparameters, with the number of neighbors (n_neighbors) set to 5 by default.

Below are the accuracy scores obtained from this model: -

```
Accracy of training dataset: 0.8360822126363867

accuracy for testing dataset 0.8312611012433393
```

Fig 59: - Accuracy Scores From KNN

Below are the confusion matrices obtained from this model: -

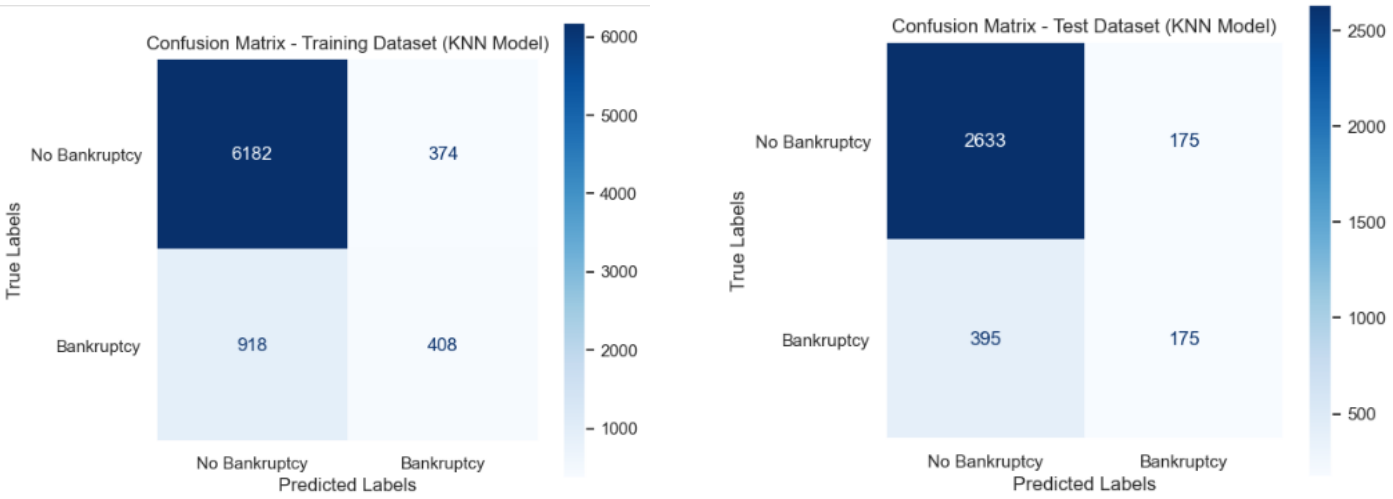


Fig 60: - Confusion Matrix From KNN

Below are the classification Report obtained from this model: -

classificatoin report of training dataset

	precision	recall	f1-score	support
0.0	0.87	0.94	0.91	6556
1.0	0.52	0.31	0.39	1326
accuracy			0.84	7882
macro avg	0.70	0.63	0.65	7882
weighted avg	0.81	0.84	0.82	7882

classsification report for testing dataset

	precision	recall	f1-score	support
0.0	0.87	0.94	0.90	2808
1.0	0.50	0.31	0.38	570
accuracy			0.83	3378
macro avg	0.68	0.62	0.64	3378
weighted avg	0.81	0.83	0.81	3378

Fig 61: - Classification Report From KNN

Below are the AUC scores and ROC curves obtained from this model: -

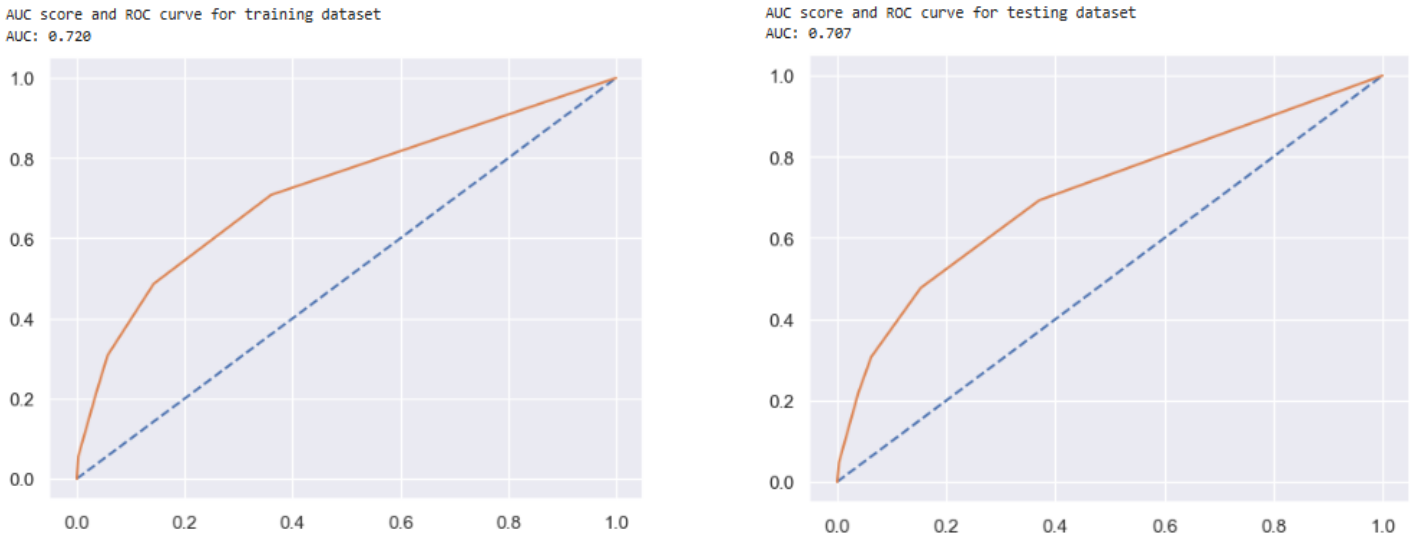


Fig 62: - ROC Curve and AUC Scores From KNN

Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset
array([0.82509506, 0.82636248, 0.8286802 , 0.8248731 , 0.81979695,
       0.82994924, 0.84771574, 0.83629442, 0.84390863, 0.82741117])
cross validation scores for test dataset
array([0.81952663, 0.83727811, 0.82248521, 0.79585799, 0.80473373,
       0.83431953, 0.80473373, 0.79881657, 0.74777448, 0.84866469])
```

Fig 63: - Cross Validation Scores From KNN

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Find the right value of n_neighbor: -

Choosing the appropriate value for n_neighbors is crucial to achieving optimal accuracy in the KNN model. To determine the best value, we evaluate the model using Mean Squared Error (MSE) scores. The value of n_neighbors that yields the lowest MSE indicates the least prediction error and is considered the most optimal for the model.

Below are the MSE scores: -

```
[0.29840142095914746,
 0.17229129662522202,
 0.1687388987566607,
 0.16370633510953225,
 0.16400236826524572,
 0.16548253404381286,
 0.1642984014209592,
 0.15571343990526942,
 0.16341030195381878,
 0.16459443457667255]
```

Fig 64: - MSE Scores

Below is the graphical version of of MSE scores acorss numerous values of n_neighbors.

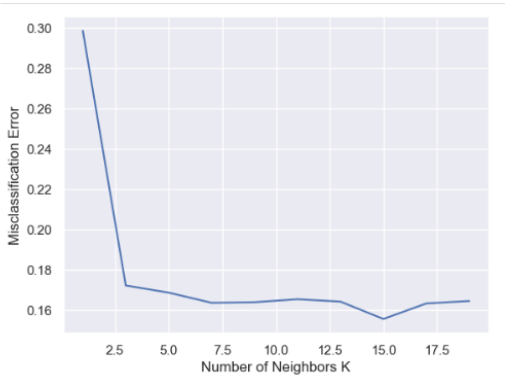


Fig 64: - Graphical Version of MSE Score

From the above plotted graph, we can see the n_neighbors with value "15" gives the least MSE score. With which we can proceed and build KNN model with n_neighbor value as "15".

Below are the accuracy scores obtained from this model: -

```
Accuracy of training dataset: 0.8553666582085765

Accuracy of training dataset: 0.8496151568975725
```

Fig 65: - Accuracy Scores from KNN (n=15)

Below are the confusion matrices obtained from this model: -

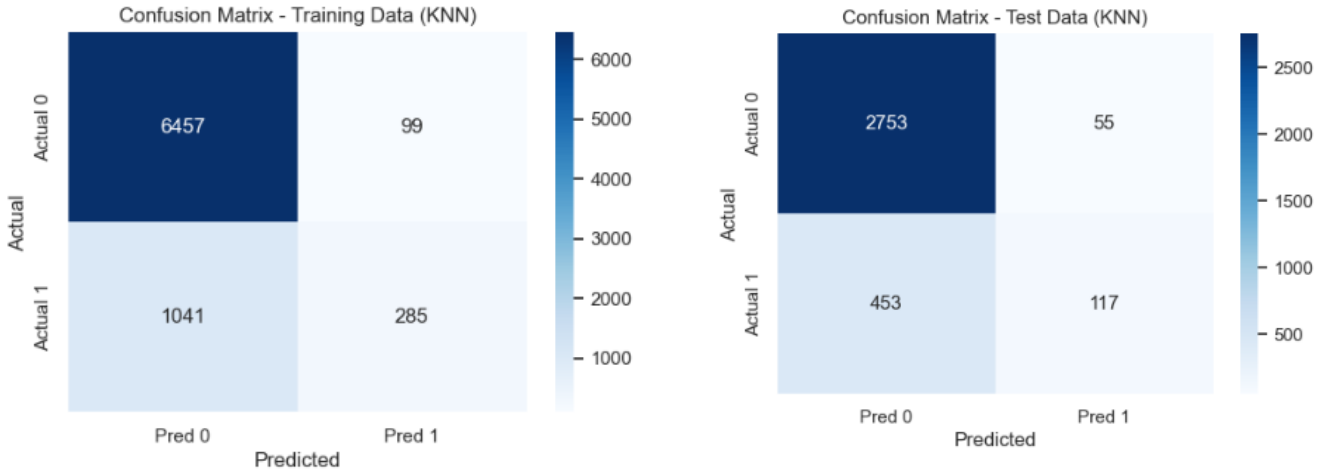


Fig 66: - Confusion Matrix from KNN (n=15)

Below are the classification Report obtained from this model: -

classification report for training dataset

	precision	recall	f1-score	support
0.0	0.86	0.98	0.92	6556
1.0	0.74	0.21	0.33	1326
accuracy			0.86	7882
macro avg	0.80	0.60	0.63	7882
weighted avg	0.84	0.86	0.82	7882

classification report for testing dataset

	precision	recall	f1-score	support
0.0	0.86	0.98	0.92	2808
1.0	0.68	0.21	0.32	570
accuracy			0.85	3378
macro avg	0.77	0.59	0.62	3378
weighted avg	0.83	0.85	0.81	3378

Fig 67: - Classification Report from KNN (n=15)

Below are the AUC scores and ROC curves obtained from this model: -

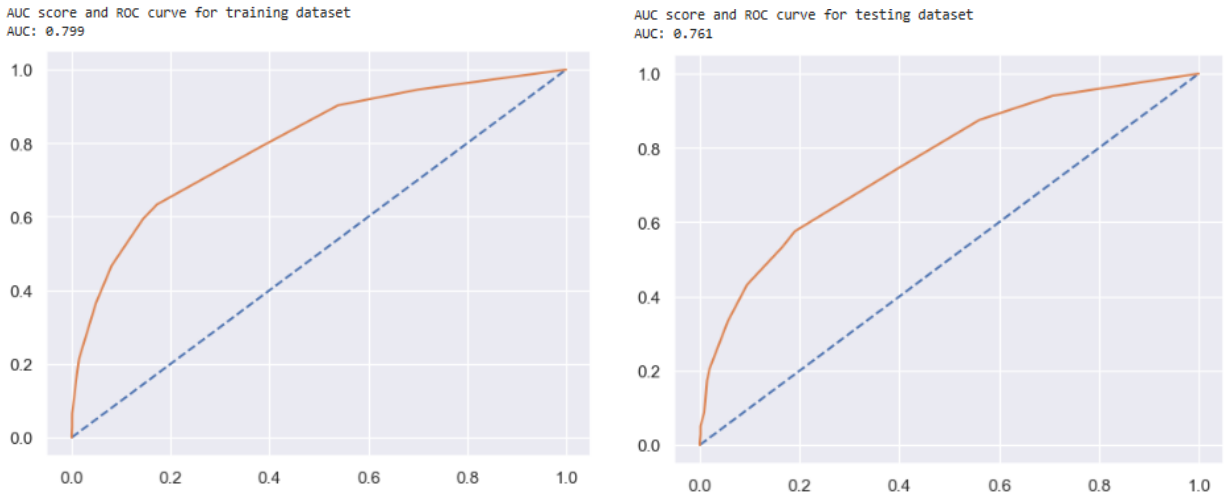


Fig 68: - ROC Curve and AUC Scores From KNN (n=15)

Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset
array([0.83776933, 0.84664132, 0.84771574, 0.85913706, 0.8464467 ,
       0.85279188, 0.84263959, 0.84263959, 0.85406091, 0.85025381])
cross validation scores for testing dataset
array([0.85207101, 0.83136095, 0.82248521, 0.82248521, 0.85207101,
       0.83431953, 0.83727811, 0.82544379, 0.83976261, 0.83976261])
```

Fig 69: - Cross Validation Scores From KNN(n=15)

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using GridSearchCV and getting the best hyperparameters: -

After building the model with its n_neighbors as shown above, we will try and find the best hyper parameters to check if we can outperform the accuracy achieved by the model built with default values of hyperparameter. From GridSearchCV we found that the best parameters are “ball-tree” as algorithm, “Manhattan” as metrics, “5” as n_neighbors and “distance” as weights.

Below are the accuracy scores obtained from this model using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8572697284953058

Accuracy of testing dataset after gridsearchCV: 0.84251036116045
```

Fig 70: - Accuracy From KNN with Hyperparameter Tuning

Below are the confusion matrices obtained from this model using GridSearchCV: -

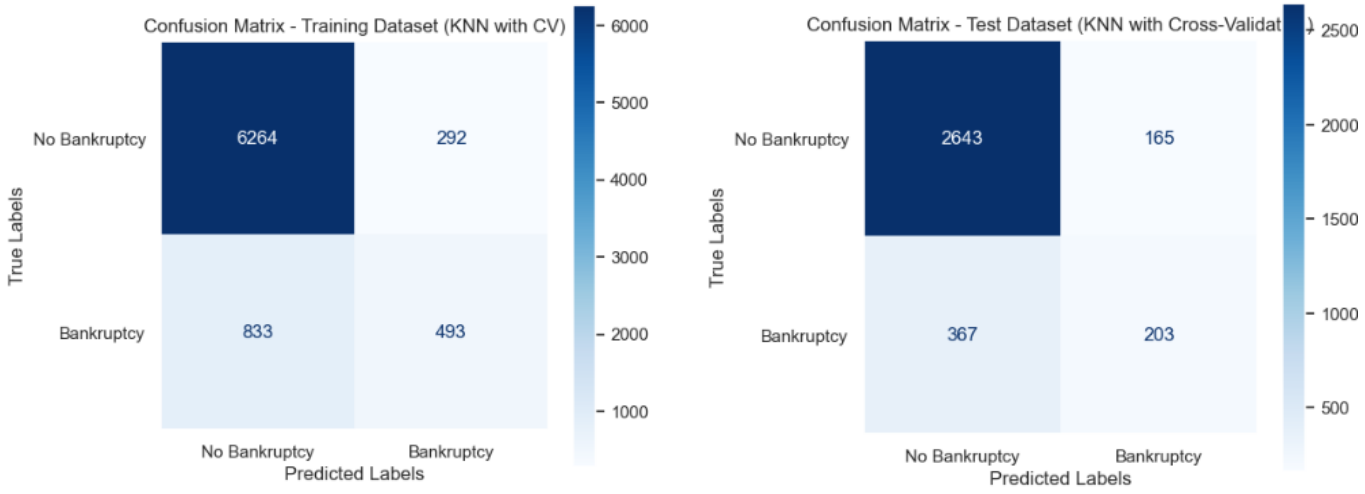


Fig 71: - Confusion Matrix from KNN with Hyperparameter Tuning

Below is the classification report obtained from this model using GridSearchCV: -

Classification report for train dataset

	precision	recall	f1-score	support
0.0	0.88	0.96	0.92	6556
1.0	0.63	0.37	0.47	1326
accuracy			0.86	7882
macro avg	0.76	0.66	0.69	7882
weighted avg	0.84	0.86	0.84	7882

Classification report for test dataset

	precision	recall	f1-score	support
0.0	0.88	0.94	0.91	2808
1.0	0.55	0.36	0.43	570
accuracy			0.84	3378
macro avg	0.71	0.65	0.67	3378
weighted avg	0.82	0.84	0.83	3378

Fig 72: - Classification Report from KNN with Hyperparameter Tuning

Below are the AUC scores and ROC curves obtained from this model using GridSearchCV: -

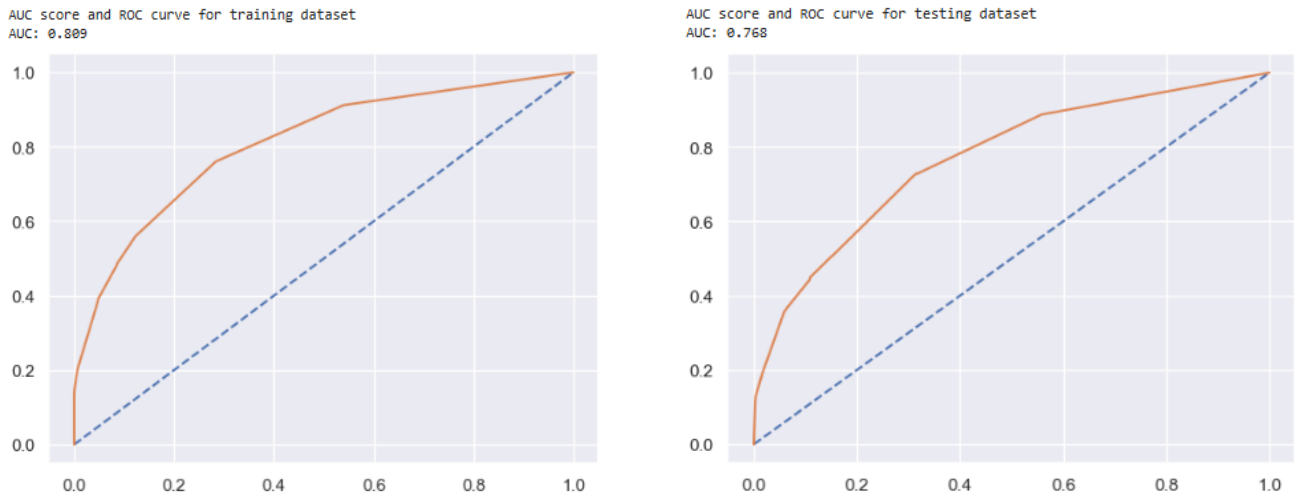


Fig 73: - ROC Curve and AUC Score from KNN with Hyperparameter Tuning

Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset
array([0.83523447, 0.85171103, 0.84771574, 0.85279188, 0.84137056,
       0.85406091, 0.8464467 , 0.83629442, 0.86294416, 0.84010152])

cross validation scores for test dataset
array([0.85798817, 0.84615385, 0.81952663, 0.83136095, 0.84615385,
       0.84615385, 0.84319527, 0.83136095, 0.85459941, 0.85756677])
```

Fig 74: Cross Validation Scores From KNN with Hyperparamter Tuning

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using SMOTE: -

The descriptive analysis indicates that the original dataset is imbalanced. To address this, we applied the SMOTE (Synthetic Minority Over-sampling Technique) method to oversample the minority class and create a balanced dataset. This approach allows us to assess whether model performance improves when trained on balanced data.

Below are the accuracy scores obtained from balanced dataset: -

```
Accuracy of training dataset: 0.7454240390482001

Accuracy of testing dataset: 0.7249851983422143
```

Fig 75: Accuracy Score from KNN with SMOTE

Below are the confusion matrices obtained from balanced dataset: -

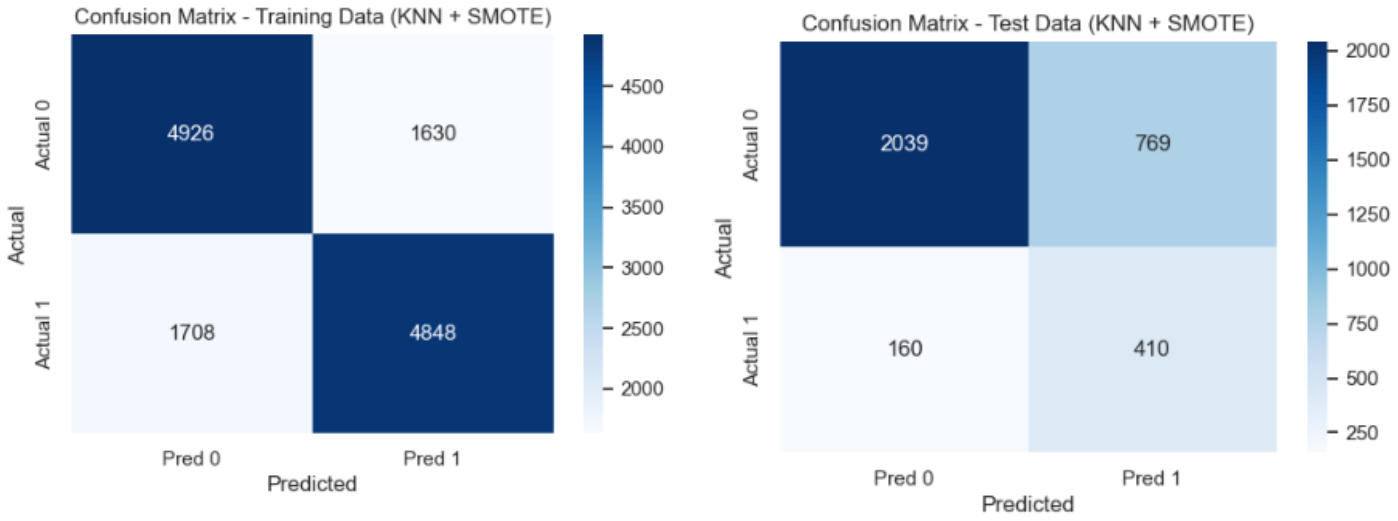


Fig 76: Confusion Matrix from KNN with SMOTE

Below are the classification reports obtained from balanced dataset: -

Classification report for train dataset

	precision	recall	f1-score	support
0.0	0.74	0.75	0.75	6556
1.0	0.75	0.74	0.74	6556
accuracy			0.75	13112
macro avg	0.75	0.75	0.75	13112
weighted avg	0.75	0.75	0.75	13112

Classification report for test dataset

	precision	recall	f1-score	support
0.0	0.93	0.73	0.81	2808
1.0	0.35	0.72	0.47	570
accuracy			0.72	3378
macro avg	0.64	0.72	0.64	3378
weighted avg	0.83	0.72	0.76	3378

Fig 77: Classification Reports from KNN with SMOTE

Below are the AUC scores and ROC curves obtained from balanced dataset: -

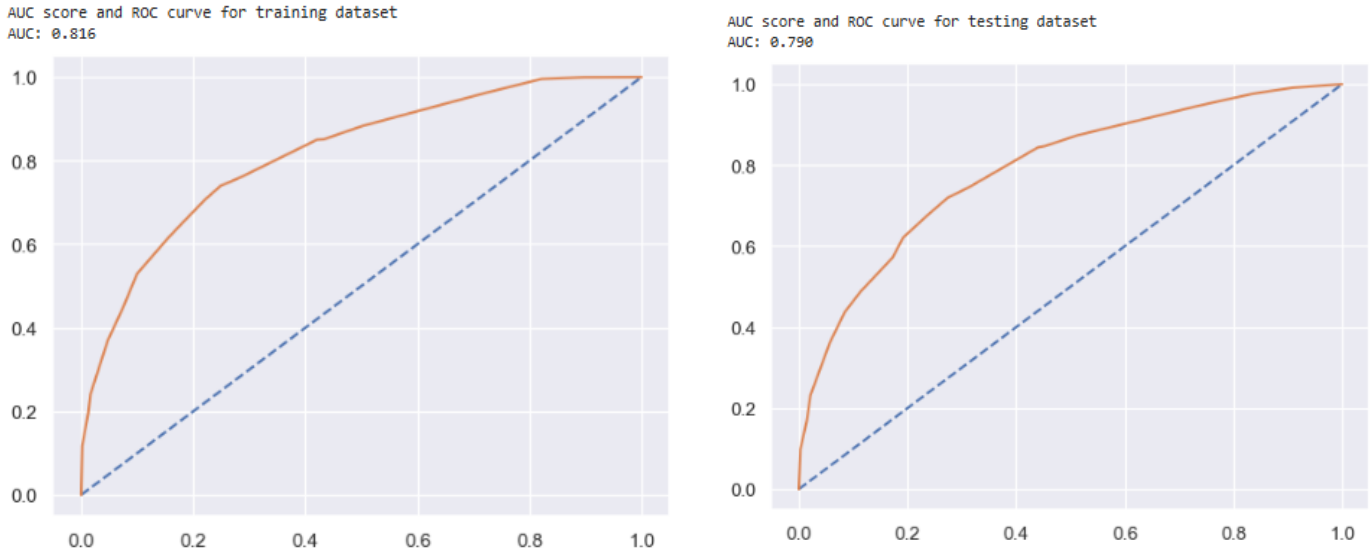


Fig 78: ROC Curve and AUC Scores From KNN with SMOTE

Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset
array([0.7027439 , 0.72942073, 0.72311213, 0.72921434, 0.72540046,
       0.74141876, 0.70480549, 0.71167048, 0.72692601, 0.71777269])

cross validation scores for test dataset
array([0.85502959, 0.82840237, 0.82248521, 0.83136095, 0.84615385,
       0.83431953, 0.82840237, 0.83136095, 0.8189911 , 0.83086053])
```

Fig 79: Cross Validation Scores From KNN with SMOTE

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from KNN Model:

The analysis suggests that the KNN model is neither overfitting nor underfitting. The model tuned using GridSearchCV is found to be the most optimized for prediction. However, notable variations are observed in performance metrics such as accuracy, F1 score, recall, precision, ROC curve, and AUC when compared with both the default KNN model and the model trained on SMOTE-balanced data.

While the SMOTE-based model performs well on the training dataset, a significant drop in accuracy is observed on the test dataset, indicating potential overfitting when using oversampled data.

Building Gaussian Naive Bayes

After splitting the dataset into training and testing sets, we proceed to build a model using the Naïve Bayes algorithm. This algorithm is based on Bayes' Theorem of conditional probability and operates under the assumption that all features are independent of each other. It calculates the probability of an event occurring, given that another event has already occurred, making it a simple yet powerful classification technique.

Below are accuracy scores using this model: -

```
Accuracy of training dataset: 0.28431870083735095

Accuracy of testing dataset: 0.2921847246891652
```

Fig 80: Training Scores from Naïve Bayes with SMOTE

Below are confusion matrices using this model: -

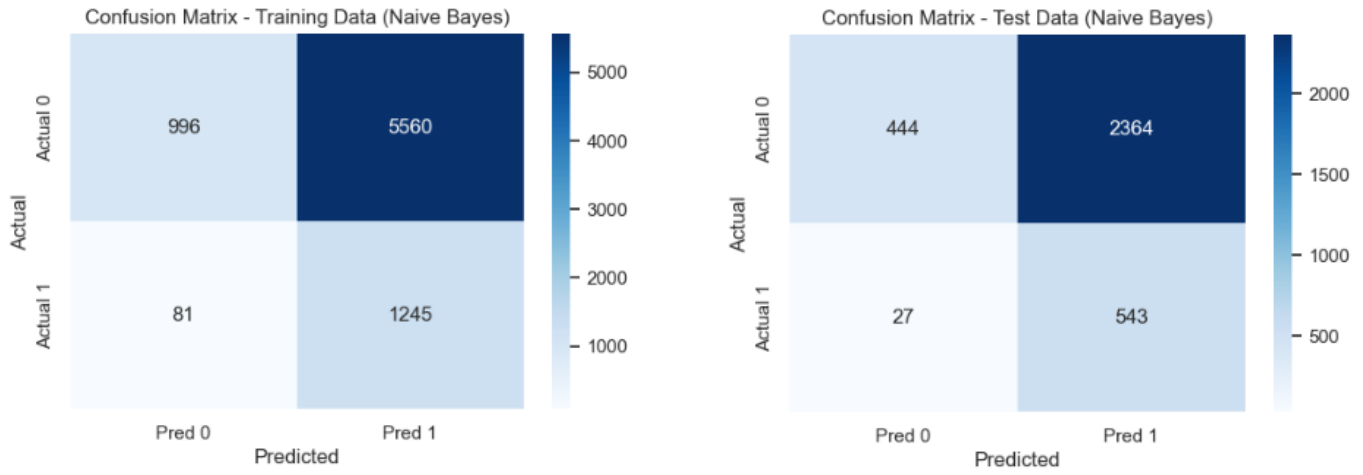


Fig 81: Confusion Matrix from Naïve Bayes with SMOTE

Below are classification reports using this model: -

Classification report of training dataset:

	precision	recall	f1-score	support
0.0	0.92	0.15	0.26	6556
1.0	0.18	0.94	0.31	1326
accuracy			0.28	7882
macro avg	0.55	0.55	0.28	7882
weighted avg	0.80	0.28	0.27	7882

Classification report of testing dataset:

	precision	recall	f1-score	support
0.0	0.94	0.16	0.27	2808
1.0	0.19	0.95	0.31	570
accuracy			0.29	3378
macro avg	0.56	0.56	0.29	3378
weighted avg	0.82	0.29	0.28	3378

Fig 82: Classification Report from Naïve Bayes with SMOTE

Below are AUC scores and ROC curves using this model: -

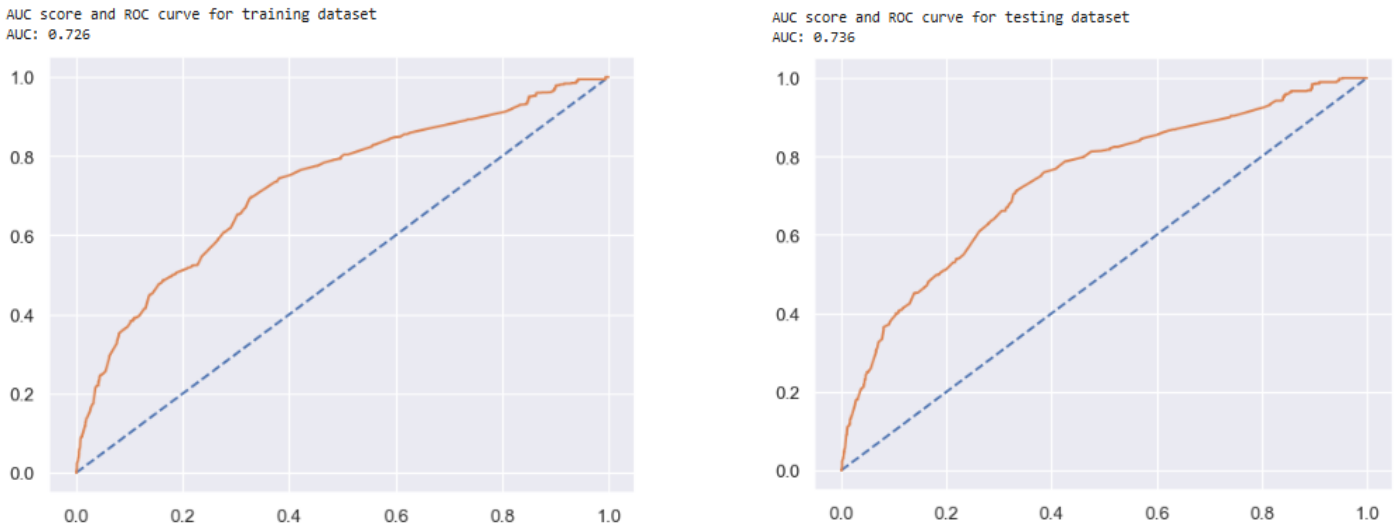


Fig 83: ROC Curve and AUC Scores from Naïve Bayes with SMOTE

Below are 10-fold cross validation scores using this model: -

```
cross validation scores for train dataset
array([0.23320659, 0.21926489, 0.23857868, 0.22969543, 0.23096447,
       0.23984772, 0.22335025, 0.23730964, 0.16878173, 0.25126904])

cross validation scores for test dataset
array([0.23372781, 0.23668639, 0.22781065, 0.26035503, 0.24852071,
       0.23076923, 0.24852071, 0.22781065, 0.23738872, 0.23442136])
```

Fig 84: Cross Validation Scores from Naïve Bayes with SMOTE

Building Gaussian Naive Bayes over balanced data using SMOTE

After building naïve bayes model using original imbalanced data. Now, we can try and build the same model using balanced data to check if it outperforms in terms of accuracy and other measurement factors.

Below are the accuracy scores obtained using Naïve Bayes algorithm over balanced dataset: -

```
Accuracy of training dataset: 0.5617754728492984

Accuracy of testing dataset: 0.3063943161634103
```

Fig 85: Accuracy Scores from Naïve Bayes with SMOTE

Below are the confusion obtained using Naïve Bayes algorithm over balanced dataset: -

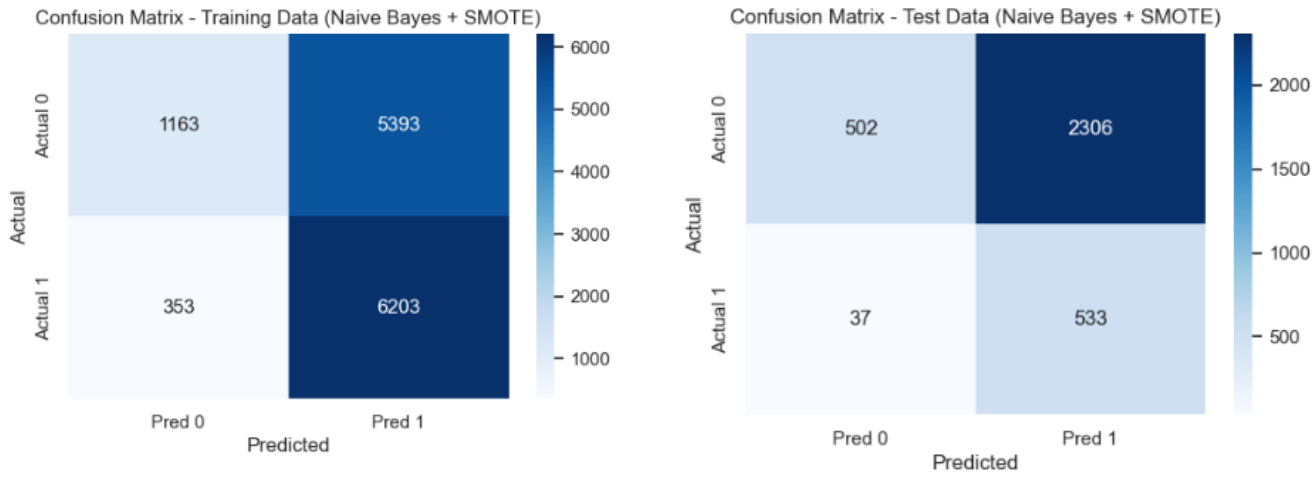


Fig 86: Confusion matrix from Naïve Bayes with SMOTE

Below is classification reports obtained using Naïve Bayes algorithm over balanced dataset: -

Classification Report for Training Dataset:

	precision	recall	f1-score	support
0.0	0.77	0.18	0.29	6556
1.0	0.53	0.95	0.68	6556
accuracy			0.56	13112
macro avg	0.65	0.56	0.49	13112
weighted avg	0.65	0.56	0.49	13112

Classification report of test dataset:

	precision	recall	f1-score	support
0.0	0.93	0.18	0.30	2808
1.0	0.19	0.94	0.31	570
accuracy			0.31	3378
macro avg	0.56	0.56	0.31	3378
weighted avg	0.81	0.31	0.30	3378

Fig 87: Classification Report from Naïve Bayes with SMOTE

Below are the AUC scores and ROC curves obtained using Naïve Bayes algorithm over balanced dataset: -

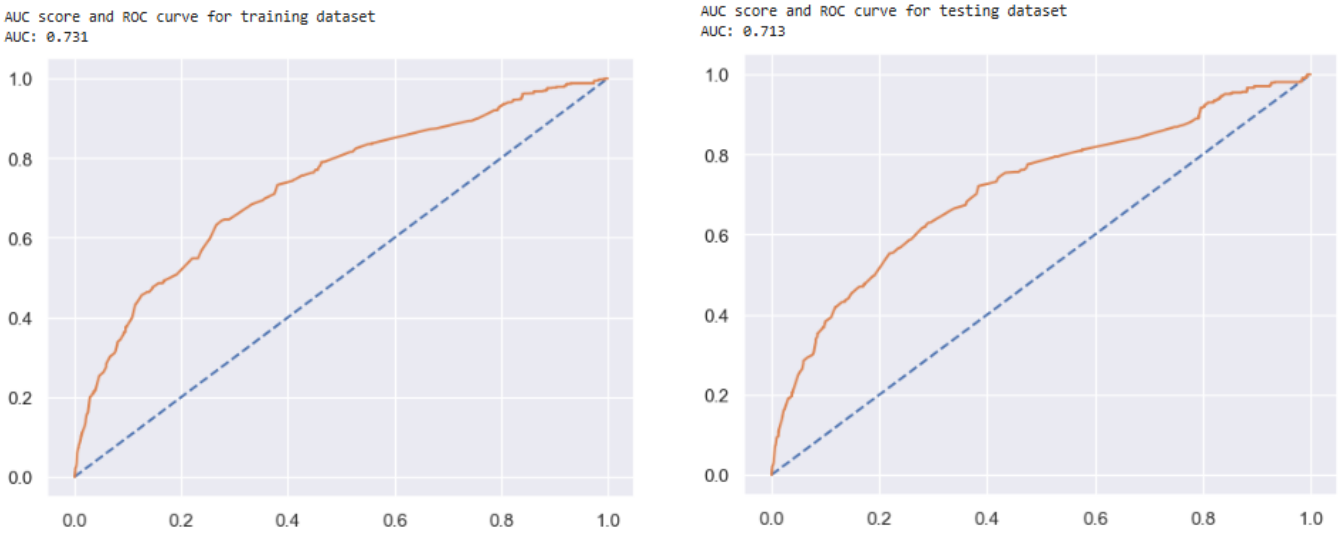


Fig 88: ROC Curve and AUC Scores from Naïve Bayes with SMOTE

Below are the 10-fold cross validation scores obtained using Naïve Bayes algorithm over balanced dataset: -

```
cross validation scores for train dataset
array([0.54954268, 0.52515244, 0.57742182, 0.5553013 , 0.56521739,
       0.5713196 , 0.55758963, 0.55606407, 0.56064073, 0.57818459])

cross validation scores for test dataset
array([0.29289941, 0.29289941, 0.28402367, 0.31656805, 0.28994083,
       0.26627219, 0.30769231, 0.27810651, 0.30860534, 0.27002967])
```

Fig 89: Cross Validation Scores from Naïve Bayes with SMOTE

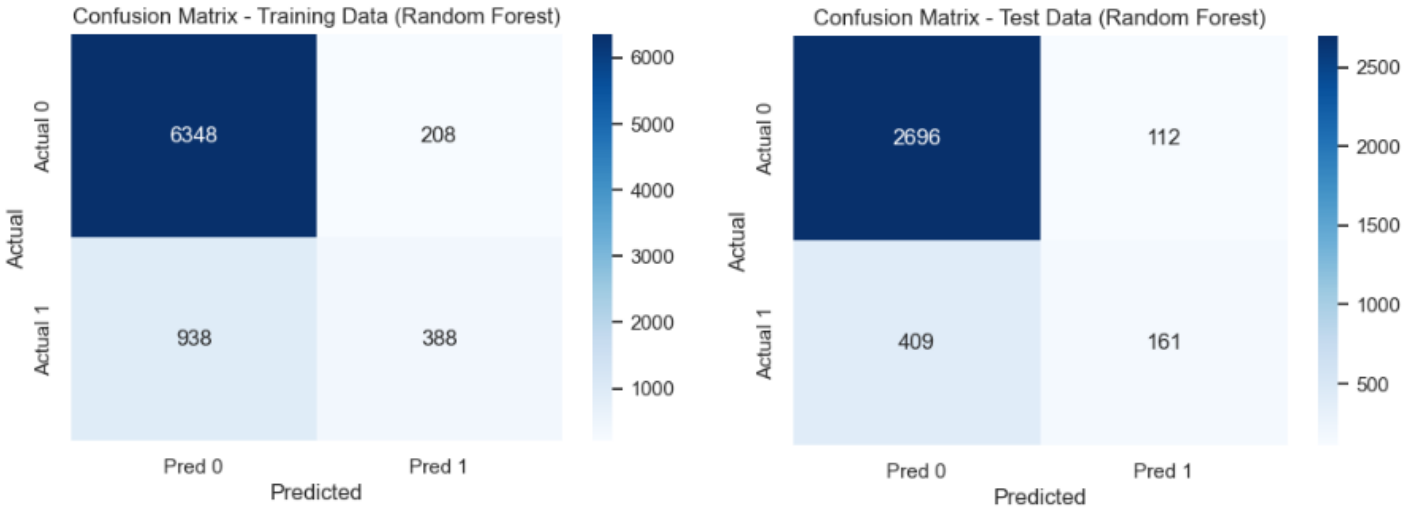
Inference/Conclusion from Naïve Bayes Model:
The analysis indicates that the Naïve Bayes model does not exhibit signs of overfitting or underfitting. Among the variations tested, the model trained on the original imbalanced dataset demonstrates the most optimized performance for prediction.
However, notable differences are observed in metrics such as accuracy, F1 score, recall, precision, ROC curve, and AUC when compared to the model trained on SMOTE-balanced data. While the SMOTE-based model performs well on the training data, its accuracy drops considerably on the testing data, suggesting overfitting due to oversampling.

Building Random Forest model: -

Firstly, let's build and random forest model for original dataset and balanced dataset and check the performance for the same.

Random Forest Built in original dataset: -

Below are the accuracy scores, confusion matrix and classification report for training and testing dataset: -



Classification report for training dataset:

	precision	recall	f1-score	support
0.0	0.87	0.97	0.92	6556
1.0	0.65	0.29	0.40	1326
accuracy			0.85	7882
macro avg	0.76	0.63	0.66	7882
weighted avg	0.83	0.85	0.83	7882

Classification report for testing dataset:

	precision	recall	f1-score	support
0.0	0.87	0.96	0.91	2808
1.0	0.59	0.28	0.38	570
accuracy			0.85	3378
macro avg	0.73	0.62	0.65	3378
weighted avg	0.82	0.85	0.82	3378

Fig 90: Accuracy Parameters from Random Forrest

Below are ROC Curve and AUC Score of train and test data set

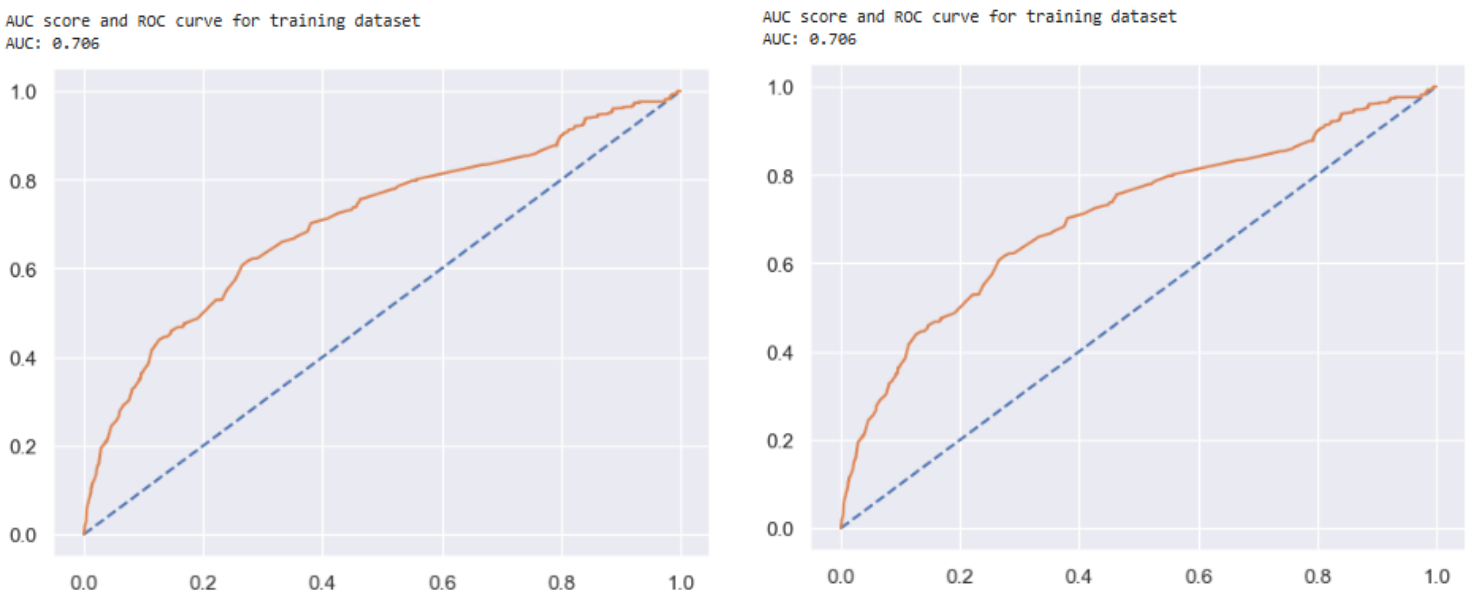


Fig 91: ROC Curve and AUC Scores from Random Forrest

Random Forest Built in balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training and Testing dataset: -

```
accuracy score for training dataset: 0.725594874923734
confusion matrix for training dataset
[[5074 1482]
 [2116 4440]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.71      0.77      0.74      6556
         1.0       0.75      0.68      0.71      6556

    accuracy                           0.73     13112
   macro avg       0.73      0.73      0.72     13112
weighted avg       0.73      0.73      0.72     13112
```

```
accuracy score for testing dataset: 0.7448194197750148
confusion matrix for testing dataset
[[2144  664]
 [ 198  372]]
classification report for testing dataste
              precision    recall  f1-score   support

         0.0       0.92      0.76      0.83      2808
         1.0       0.36      0.65      0.46       570

    accuracy                           0.74      3378
   macro avg       0.64      0.71      0.65      3378
weighted avg       0.82      0.74      0.77      3378
```

Fig 92: Accuracy Parameters From Random Forrest with SMOTE

Bagging on original dataset: -

Let's build bagging model using random forest and check if it can perform better than general random forest model.

Below are the accuracy scores, confusion matrix and classification report for training and Testing dataset:

```
accuracy score or training dataset: 0.8650088809946714
confusion report for training dataset
[[6350  206]
 [ 858  468]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.88      0.97      0.92      6556
         1.0       0.69      0.35      0.47      1326

    accuracy                           0.87      7882
   macro avg       0.79      0.66      0.70      7882
weighted avg       0.85      0.87      0.85      7882
```

```
Accuracy score for testing datatset: 0.8496151568975725
confusuion matrix for testing dataset
[[2678  130]
 [ 378  192]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.88      0.95      0.91      2808
         1.0       0.60      0.34      0.43       570

    accuracy                           0.85      3378
   macro avg       0.74      0.65      0.67      3378
weighted avg       0.83      0.85      0.83      3378
```

Fig 93: Bagging On Original Dataset

Below is the AUC score and ROC curve for training and testing dataset: -

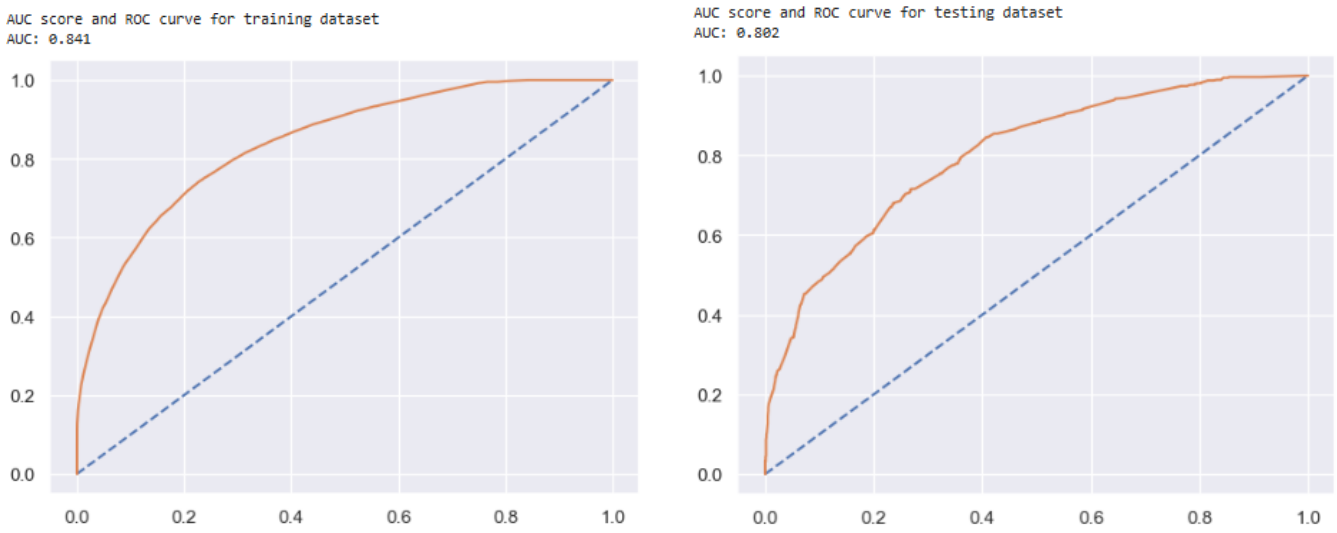


Fig 94: ROC Curve and AUC Score Bagging on Original Dataset

Bagging on Balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training and Testing dataset: -

```
accuracy score or training dataset: 0.7255186089078707
confusion report for training dataset
[[5067 1489]
 [2110 4446]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.71      0.77      0.74      6556
         1.0       0.75      0.68      0.71      6556

    accuracy                           0.73     13112
   macro avg       0.73      0.73      0.72     13112
weighted avg       0.73      0.73      0.72     13112
```

```
Accuracy score for testing datatset: 0.7448194197750148
confusuion matrix for testing dataset
[[2143  665]
 [ 197  373]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.76      0.83      2808
         1.0       0.36      0.65      0.46       570

    accuracy                           0.74      3378
   macro avg       0.64      0.71      0.65      3378
weighted avg       0.82      0.74      0.77      3378
```

Fig 95: Accuracy Parameters from Bagging on Balanced Dataset

Below is the AUC score and ROC curve for training and testing dataset: -

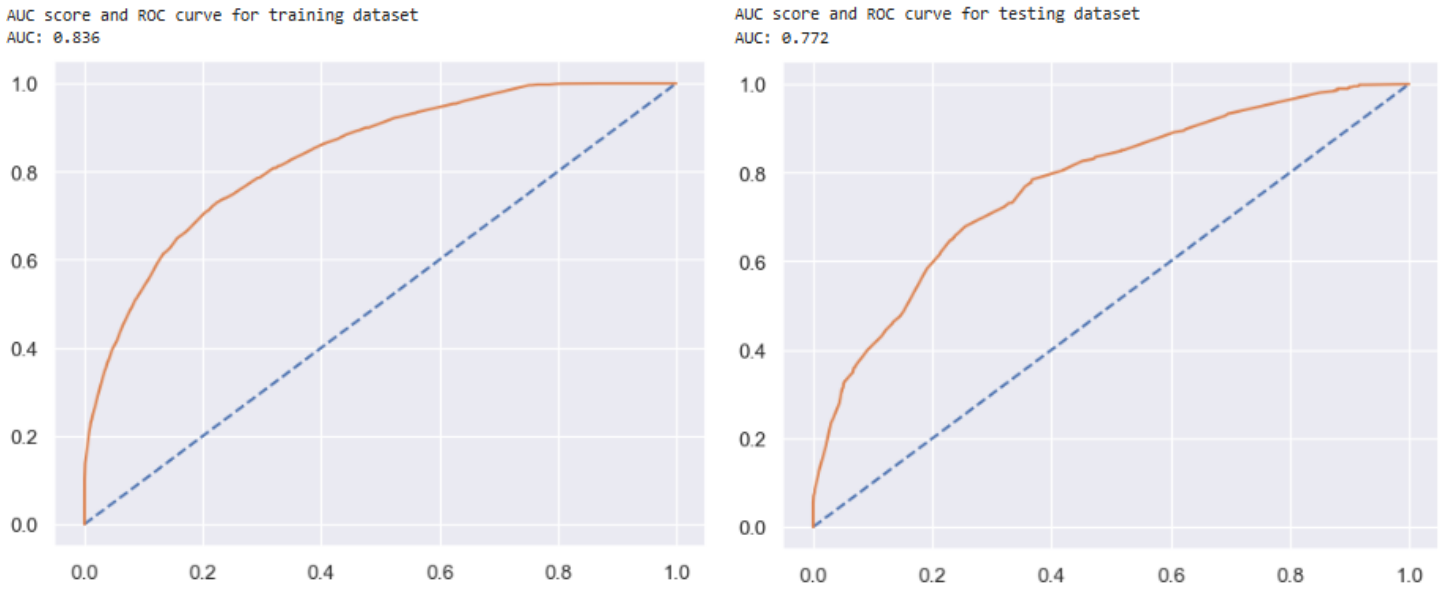


Fig 96: ROC Curve and AUC Scores from Bagging On Balanced Dataset

Building Ada-Boost Model on original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
Accuracy for training dataset: 0.840268967267191
confusion matrix for training dataset
[[6440  116]
 [1143  183]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.61      0.14      0.23      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.56      0.57      7882
weighted avg       0.81      0.84      0.80      7882
```

```
accuracy score for testing dataset: 0.8413262285375962
confusion matrix for testing dataset
[[2755   53]
 [ 483   87]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.62      0.15      0.25       570

    accuracy                           0.84      3378
   macro avg       0.74      0.57      0.58      3378
weighted avg       0.81      0.84      0.80      3378
```

Fig 97: Accuracy Parameters from Ada-Boosting on Original Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -

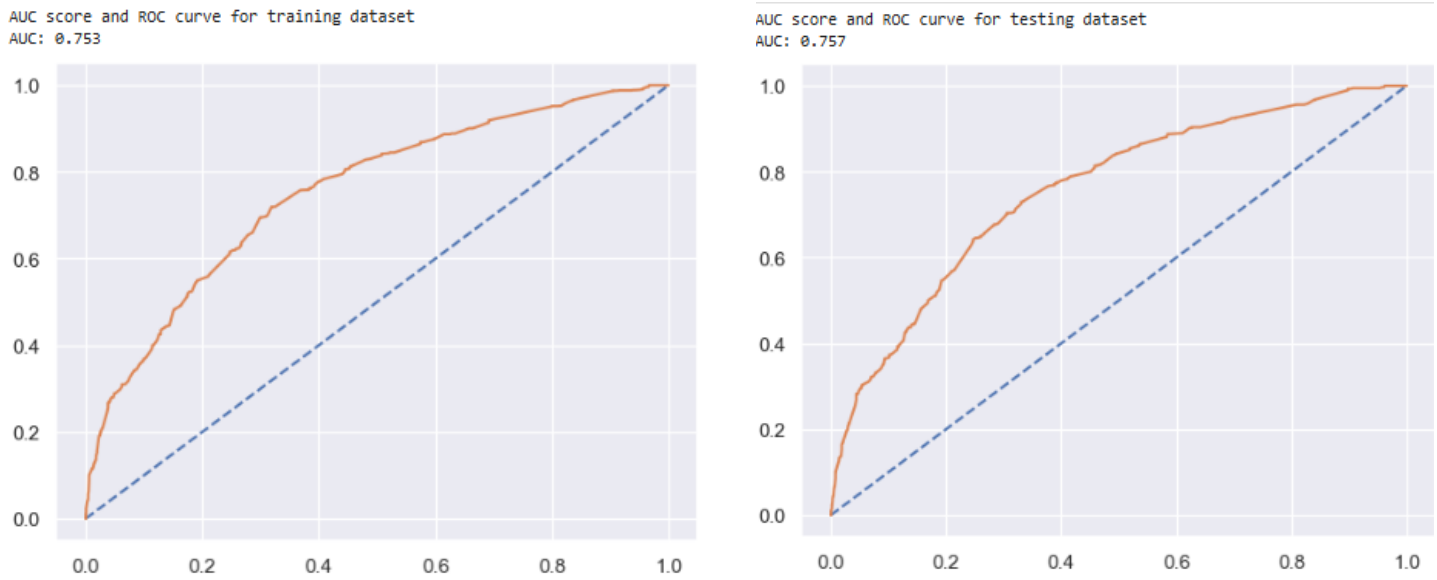


Fig 98: ROC curve and AUC score from Ada-Boosting on Original Dataset

Building Ada-Boost Model on balanced dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
Accuracy for training dataset: 0.6978340451494814
confusion matrix for training dataset
[[4280 2276]
 [1686 4870]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.72      0.65      0.68      6556
         1.0       0.68      0.74      0.71      6556

    accuracy                           0.70     13112
   macro avg       0.70      0.70      0.70     13112
weighted avg       0.70      0.70      0.70     13112
```

```
accuracy score for testing dataset: 0.6571936056838366
confusion matrix for testing dataset
[[1802 1006]
 [ 152  418]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.64      0.76      2808
         1.0       0.29      0.73      0.42       570

    accuracy                           0.66      3378
   macro avg       0.61      0.69      0.59      3378
weighted avg       0.82      0.66      0.70      3378
```

Fig 99: Accuracy Parameter from Ada-Boosting on Balanced Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -

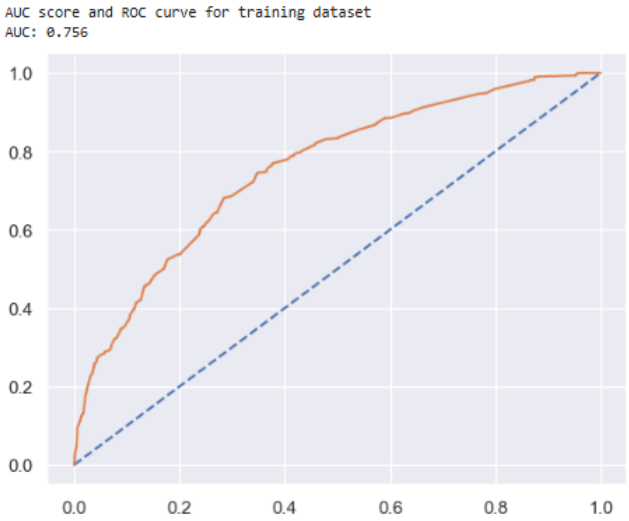


Fig 100: ROC Curve and AUC Score from Ada-Boosting on Balanced Dataset

Building Gradient Boosting Model on original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
accuracy for training dataset: 0.8506724181679777
confusion matrix for training dataset
[[6450  106]
 [1071  255]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.86      0.98      0.92      6556
         1.0       0.71      0.19      0.30      1326

    accuracy                           0.85      7882
   macro avg       0.78      0.59      0.61      7882
weighted avg       0.83      0.85      0.81      7882
```

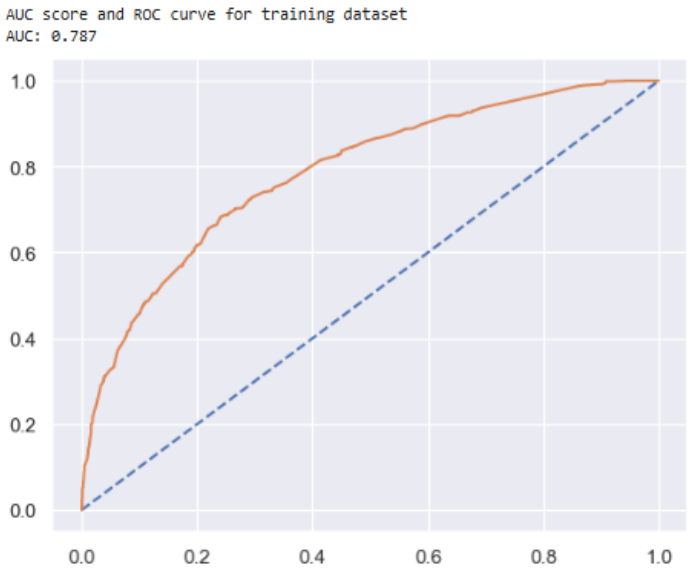
```
accuracy score for testing dataset: 0.8463587921847247
confusuon matrix for testing dataset
[[2745   63]
 [ 456  114]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.86      0.98      0.91      2808
         1.0       0.64      0.20      0.31       570

    accuracy                           0.85      3378
   macro avg       0.75      0.59      0.61      3378
weighted avg       0.82      0.85      0.81      3378
```

Fig 101: Accuracy Parameter from Gradient-Boosting on Original Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -



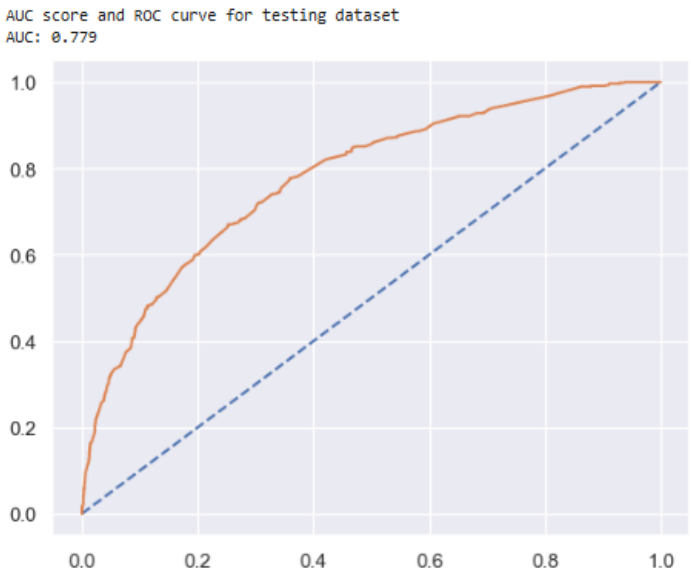


Fig 102: ROC Curve and AUC Score from Gradient-Boosting on Original Dataset

Building Gradient Boosting Model on balanced dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
accuracy for training dataset: 0.7210189139719341
confusion matrix for training dataset
[[4871 1685]
 [1973 4583]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.71      0.74      0.73      6556
         1.0       0.73      0.70      0.71      6556

    accuracy                           0.72     13112
   macro avg       0.72      0.72      0.72     13112
weighted avg       0.72      0.72      0.72     13112
```

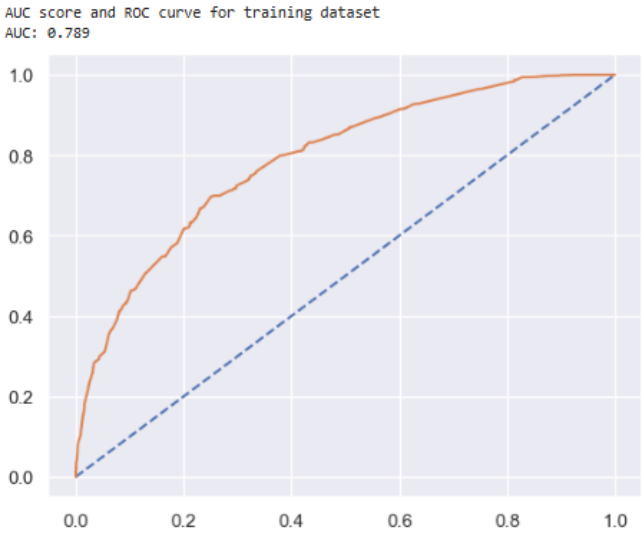
```
accuracy score for testing dataset: 0.7175843694493783
confusuon matrix for testing dataset
[[2037  771]
 [ 183  387]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.73      0.81      2808
         1.0       0.33      0.68      0.45       570

    accuracy                           0.72     3378
   macro avg       0.63      0.70      0.63     3378
weighted avg       0.82      0.72      0.75     3378
```

Fig 103: Accuracy Parameter from Gradient-Boosting on Balanced Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -



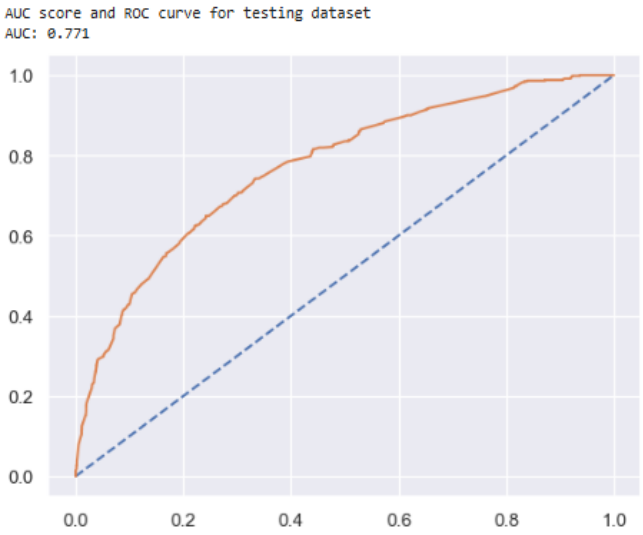


Fig 104: ROC Curve and AUC Score from Gradient-Boosting on Balanced Dataset

Inferences from Bagging and Boosting Models:

- From both models, it is evident that some level of overfitting still exists.
- When trained on the SMOTE-balanced dataset, the models demonstrate strong performance on the training data. However, a noticeable drop in accuracy is observed on the testing data, indicating reduced generalization capability.

Building Support Vector Machine (SVM) Model on Original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
accuracy for training dataset: 0.8570159857904085
confusion matrix for training dataset
[[6399  157]
 [ 970  356]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6556
         1.0       0.69      0.27      0.39      1326

    accuracy                           0.86      7882
   macro avg       0.78      0.62      0.65      7882
weighted avg       0.84      0.86      0.83      7882
```

```
accuracy score for testing dataset: 0.8481349911190054
confusuon matrix for testing dataset
[[2710   98]
 [ 415  155]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.87      0.97      0.91      2808
         1.0       0.61      0.27      0.38       570

    accuracy                           0.85      3378
   macro avg       0.74      0.62      0.65      3378
weighted avg       0.82      0.85      0.82      3378
```

Fig 105: Accuracy parameter Scores from SVM on Original Dataset

Building Support Vector Machine (SVM) Model on Original dataset Using Hyper-Parameter: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
Accuracy of training dataset after gridsearchCV: 0.8652626236995686

Accuracy of testing dataset after gridsearchCV: 0.85198342214328
```

```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6556
         1.0       0.74      0.31      0.44      1326

    accuracy                           0.87      7882
   macro avg       0.81      0.64      0.68      7882
weighted avg       0.85      0.87      0.84      7882
```

```
confusuon matrix for training dataset
array([[6410,  146],
       [ 916,  410]], dtype=int64)
```

```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.87      0.97      0.92      2808
         1.0       0.63      0.29      0.40       570

    accuracy                           0.85      3378
   macro avg       0.75      0.63      0.66      3378
weighted avg       0.83      0.85      0.83      3378
```

```
confusuon matrix for testing dataset
array([[2712,   96],
       [ 404,  166]], dtype=int64)
```

Fig 107: Accuracy parameter Scores from SVM on Original Dataset Using Hyper-Parameter

Building Support Vector Machine (SVM) Model on Balanced Dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

```
accuracy for training dataset: 0.7524405125076266
confusion matrix for training dataset
[[5112 1444]
 [1802 4754]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.74      0.78      0.76      6556
         1.0       0.77      0.73      0.75      6556

    accuracy                           0.75     13112
   macro avg       0.75      0.75      0.75     13112
weighted avg       0.75      0.75      0.75     13112
```

```
accuracy score for testing dataset: 0.7362344582593251
confusuon matrix for testing dataset
[[2108  700]
 [ 191  379]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.75      0.83      2808
         1.0       0.35      0.66      0.46       570

    accuracy                           0.74      3378
   macro avg       0.63      0.71      0.64      3378
weighted avg       0.82      0.74      0.76      3378
```

Fig 108: Accuracy parameter Scores from SVM on Balanced Dataset

Inference from SVM Model:

The analysis suggests that the SVM model does not exhibit overfitting or underfitting. Among the variations tested, the model trained on the SMOTE-balanced dataset appears to be the most optimized for prediction. Compared to the model built on the original imbalanced data, significant improvements are observed in accuracy, F1 score, recall, precision, ROC curve, and AUC scores. The SVM model trained on balanced data using SMOTE performs well on both the training and testing datasets, indicating better generalization and robustness.

Overall Model Building Comparison across parameters: -

MODELS	Training Dataset - 70%								Test Dataset - 70%							
	Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score	Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score
Logistic Regression	0.83	0.84	0.98	0.91	0.59	0.12	0.2	0.75	0.83	0.84	0.98	0.91	0.58	0.13	0.21	0.75
Logistic Regression - CV	0.83	0.85	0.98	0.91	0.59	0.12	0.21	0.75	0.83	0.85	0.98	0.91	0.59	0.13	0.21	0.74
Logistic Regression - SM	0.69	0.71	0.64	0.67	0.67	0.74	0.71	0.74	0.65	0.92	0.63	0.75	0.29	0.75	0.42	0.74
LDA	0.84	0.86	0.98	0.91	0.61	0.2	0.3	0.74	0.83	0.85	0.97	0.91	0.56	0.18	0.27	0.74
LDA - CV	0.84	0.87	0.96	0.91	0.59	0.27	0.37	0.74	0.83	0.86	0.96	0.91	0.53	0.25	0.34	0.74
LDA - SM	0.69	0.69	0.7	0.7	0.7	0.69	0.69	0.74	0.69	0.92	0.7	0.79	0.32	0.71	0.44	0.74
KNN	0.83	0.87	0.94	0.91	0.52	0.31	0.39	0.72	0.83	0.87	0.94	0.9	0.5	0.31	0.38	0.7
KNN - 15	0.85	0.86	0.98	0.92	0.74	0.21	0.33	0.79	0.84	0.86	0.98	0.92	0.68	0.21	0.32	0.76
KNN - CV	0.85	0.88	0.96	0.92	0.63	0.37	0.47	0.81	0.84	0.88	0.94	0.91	0.55	0.36	0.43	0.76
KNN - SM	0.74	0.74	0.75	0.75	0.75	0.74	0.74	0.81	0.72	0.93	0.73	0.81	0.35	0.72	0.47	0.79
Naive Bayes	0.28	0.92	0.15	0.26	0.18	0.94	0.31	0.72	0.29	0.94	0.16	0.27	0.19	0.95	0.31	0.73
Naive Bayes - SM	0.56	0.77	0.18	0.29	0.53	0.95	0.68	0.73	0.3	0.93	0.18	0.3	0.19	0.94	0.31	0.71
Random Forest	0.85	0.87	0.97	0.92	0.65	0.29	0.4	0.7	0.7	0.87	0.96	0.91	0.59	0.28	0.38	0.7
Random Forest - SM	0.72	0.71	0.77	0.74	0.75	0.68	0.71	0.7	0.74	0.92	0.76	0.83	0.36	0.65	0.46	0.7
Bagging	0.86	0.88	0.97	0.92	0.69	0.35	0.47	0.84	0.84	0.88	0.95	0.91	0.6	0.34	0.43	0.8
Bagging - SM	0.72	0.71	0.77	0.74	0.75	0.68	0.71	0.83	0.74	0.92	0.76	0.83	0.36	0.65	0.46	0.77
Ada - Boosting	0.84	0.85	0.98	0.91	0.61	0.14	0.23	0.75	0.84	0.85	0.98	0.91	0.62	0.15	0.25	0.75
Ada - Boosting - SM	0.69	0.72	0.65	0.68	0.68	0.74	0.71	0.75	0.65	0.92	0.64	0.76	0.29	0.73	0.42	0.75
Gradient Boosting	0.85	0.86	0.98	0.92	0.71	0.19	0.3	0.78	0.84	0.86	0.98	0.91	0.64	0.2	0.31	0.77
Gradient Boosting - SM	0.72	0.71	0.74	0.73	0.73	0.7	0.71	0.78	0.71	0.92	0.73	0.81	0.33	0.68	0.45	0.77
SVM	0.85	0.87	0.98	0.92	0.69	0.27	0.39	0.75	0.84	0.87	0.97	0.91	0.61	0.27	0.38	0.74
SVM - CV	0.86	0.87	0.98	0.92	0.74	0.31	0.44	0.75	0.85	0.87	0.97	0.92	0.63	0.29	0.4	0.75
SVM - SM	0.75	0.74	0.78	0.76	0.77	0.73	0.75	0.75	0.73	0.92	0.75	0.83	0.35	0.66	0.46	0.75

Table 9: Comparison Across Various Models

Indicators/Symbols Used in the Tabular Data:

- **CV**: Refers to the scores of the model built using the best hyperparameters obtained through GridSearchCV. The model name is used as a prefix (e.g., LR-CV for Logistic Regression with GridSearchCV).
- **SM**: Represents the scores of the model trained on a SMOTE-balanced dataset. The model name precedes the indicator (e.g., NB-SM for Naïve Bayes with SMOTE).
- **KNN-15**: Indicates the K-Nearest Neighbors model built with n_neighbors set to 15.

Inferences on Final Model Selection:

- **KNN models**, particularly KNN with cross-validation (KNN - CV) and KNN with 15 neighbors, show **strong performance**, with training accuracy around 0.85 and test accuracy near 0.84, and relatively balanced precision and recall scores compared to other models. The AUC scores for KNN variants (up to 0.79 on training and 0.76 on test) are among the highest, indicating good discrimination ability.
- **Logistic Regression and LDA models** perform quite similarly with an accuracy around 0.83-0.84 on training and testing sets. LDA tends to have slightly better recall and F1 scores for the positive class

(label 1) compared to Logistic Regression, suggesting it may be better at identifying the minority class in this imbalanced dataset.

- **Bagging classifier** consistently achieves one of the highest AUC scores (up to 0.84 training, 0.80 testing) and balanced precision/recall, making it one of the best performing ensemble methods here.
- Models trained with **SMOTE (Synthetic Minority Oversampling Technique)** show higher recall for the positive class but generally lower overall accuracy and precision for the negative class, which is expected since SMOTE tries to improve minority class detection at some cost to majority class precision.
- **Naive Bayes** performs poorly overall with very low accuracy and F1 scores, likely due to its strong assumptions not matching the data well.
- **Random Forest and Gradient Boosting models** also perform well, but slightly below Bagging and tuned KNN in this case.
- **SVM with cross-validation** shows stable performance with balanced precision and recall, AUC around 0.75-0.76.

Overall, the Bagging classifier and KNN (especially tuned with CV) stand out as best-performing models, balancing accuracy, recall, and AUC metrics effectively for this classification task. Logistic Regression and LDA remain strong baseline models, with slight advantages for LDA in minority class detection.

Implications of the Final Model on Business :

- The model, especially Bagging and tuned KNN, provides **reliable predictions of customers at risk of churn**, enabling the business to **focus retention efforts where they matter most**.
- Since recall for the minority class (churners) is moderate (around 30-40% for best models without SMOTE), the business should **use these predictions as a key input but combine with other signals or manual checks** to minimize false negatives.
- The business can design **targeted marketing campaigns** such as family floater plans or exclusive discounts to the predicted high-risk customers to reduce churn probability.
- Incentives like **regular discount coupons or e-wallet payment rewards** can be optimized by focusing on segments identified by the model as more likely to churn or respond to offers.
- With models trained using SMOTE showing improved recall but reduced precision and accuracy, the business should **balance between aggressive retention offers (higher recall) and cost-effective marketing (higher precision)** to avoid excessive spend on false positives.
- This predictive model acts as a **data-driven baseline**, helping the business understand its current customer retention landscape and measure improvements from future strategies.