Capstone Project - PGP-DSBA
Customer Churn

Final Report

Isha Shukla
10 September 2024

1. Introduction to Business Problem

a. Problem Statement

The DTH provider is facing increasing competition, leading to challenges in retaining existing customers. The company needs to develop a churn prediction model to identify accounts at risk of churning. Given that one account may represent multiple customers, losing an account can significantly impact the business. The goal is to accurately predict potential churners and design targeted campaigns to retain them. These campaigns must be cost-effective, as recommendations will be reviewed by the revenue assurance team, which will reject any strategies that result in a financial loss to the company. The challenge is to create a model that maximizes customer retention while minimizing costs.

b. Need of the study/project

The study is essential due to the intense competition in the DTH market, where retaining customers is increasingly challenging. Given that losing one account can mean losing multiple customers, predicting churn is critical for targeted retention efforts. The project aims to develop a data-driven churn prediction model to identify at-risk accounts and propose cost-effective, personalized retention strategies. This understanding will empower the company to make informed decisions that will help sustain and grow its customer base while minimizing losses and ensuring profitability.

c. Understanding business/social opportunity

This case study focuses on a DTH company where customers are assigned unique account IDs, with a single account ID potentially serving multiple customers (such as in a family plan). Customers are diverse in terms of gender, marital status, and payment methods, which include a variety of options such as debit card, UPI, and cash on delivery. The company's customer base is also segmented according to the types of plans they opt for based on their usage and the devices they use (computer or mobile). Additionally, customers benefit from cashback offers on bill payments.

The success of the business largely depends on customer loyalty and the perceived value of the services provided. Offering quality service, along with value-added features, plays a crucial role in retaining customers. Running promotional and festival offers is another strategy that not only attracts new customers but also helps in retaining existing ones. The company faces a significant challenge because losing one account ID can result in losing multiple customers, which directly impacts the business's revenue and market standing.

In a market where having a DTH connection is almost a necessity for every household, competition is fierce. The company's ability to differentiate itself from competitors hinges on understanding and addressing the factors that drive customer loyalty and satisfaction.

The business opportunity in this project lies in the ability to proactively address customer churn, which is a significant challenge in the competitive DTH industry. By developing a churn prediction model, the company can identify accounts at risk of leaving and implement targeted retention strategies. This not only helps in preserving revenue by retaining existing customers but also reduces the cost associated with acquiring new customers. Furthermore, understanding the factors that contribute to churn enables the company to refine its product offerings, improve customer satisfaction, and strengthen customer loyalty. This approach positions the company to maintain a competitive edge in the market, ensuring sustained growth and profitability.

2. 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

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

- Dataset has 11260 rows and 19 columns.
- Target variable is "Churn" (1 dependent) and 18 variables are independent.
- Dataset has 11260 unique AccountID.
- "Churn" shows 1 if customer is churned and if it is not churned it shows 0.
- Dataset is a mix of categorical and continous variables.

2. In terms of time:

- **CC_Contacted_L12m, Complain_l12m, coupon_used_l12m, cashback_l12m:** These features suggest a 12-month rolling period, where data is collected on customer activities or metrics over the last 12 months.
- r**ev_per_month, rev_growth_yoy:** These are likely calculated on a monthly basis or as year-over-year comparisons.

3. **Frequency:**
   - **CC_Contacted_L12m:** Data on how many times the customer contacted customer care could be collected continuously, with a tally kept over the 12-month period.
   - **coupon_used_l12m:** The frequency of coupon usage over the past 12 months is likely recorded each time a coupon is used.
   - **cashback_l12m:** Monthly average cashback is likely calculated from individual transactions that offer cashback rewards.
   - **Day_Since_CC_connect:** The count of days since the last customer care contact could be updated daily.
4. **Methodology:**
   - **Payment, Gender, Marital_Status, account_segment**: These categorical variables might have been collected from customer account records, surveys, or registration forms.
   - **Service_Score, CC_Agent_Score:** These scores could be gathered through customer satisfaction surveys or feedback forms.
   - **Account_user_count:** This is likely tracked through the account management system, recording how many users are linked to each account.

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

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score
0	20000	1	4	3.0	6.0	Debit Card	Female	3.0	3	Super	2.0
1	20001	1	0	1.0	8.0	UPI	Male	3.0	4	Regular Plus	3.0
2	20002	1	0	1.0	30.0	Debit Card	Male	2.0	4	Regular Plus	3.0
3	20003	1	0	3.0	15.0	Debit Card	Male	2.0	4	Super	5.0
4	20004	1	0	1.0	12.0	Credit Card	Male	2.0	3	Regular Plus	5.0

First five rows of the dataset

- Dataset has 11260 rows and 19 columns.
- There are no duplicates present in the dataset.
- There are null values in the dataset.

```
(11260, 19)
```

Shape of the data

```
<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
```

Info about the dataset

- There are 5 float datatype columns, 2 integer datatype and 12 object datatypes at the starting.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
AccountID	11260.0	NaN	NaN	NaN	25629.5	3250.62635	20000.0	22814.75	25629.5	28444.25	31259.0
Churn	11260.0	NaN	NaN	NaN	0.168384	0.374223	0.0	0.0	0.0	0.0	1.0
Tenure	11158.0	38.0	1.0	1351.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
City_Tier	11148.0	NaN	NaN	NaN	1.653929	0.915015	1.0	1.0	1.0	3.0	3.0
CC_Contacted_LY	11158.0	NaN	NaN	NaN	17.867091	8.853269	4.0	11.0	16.0	23.0	132.0
Payment	11151	5	Debit Card	4587	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	11152	4	Male	6328	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Service_Score	11162.0	NaN	NaN	NaN	2.902526	0.725584	0.0	2.0	3.0	3.0	5.0
Account_user_count	11148.0	7.0	4.0	4569.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
account_segment	11163	7	Super	4062	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CC_Agent_Score	11144.0	NaN	NaN	NaN	3.066493	1.379772	1.0	2.0	3.0	4.0	5.0
Marital_Status	11048	3	Married	5860	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rev_per_month	11158.0	59.0	3.0	1746.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Complain_ly	10903.0	NaN	NaN	NaN	0.285334	0.451594	0.0	0.0	0.0	1.0	1.0
rev_growth_yoy	11260.0	20.0	14.0	1524.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
coupon_used_for_payment	11260.0	20.0	1.0	4373.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Day_Since_CC_connect	10903.0	24.0	3.0	1816.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cashback	10789.0	5693.0	155.62	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Login_device	11039	3	Mobile	7482	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Describing the dataset

- The `df.describe()` function in pandas provides a summary of statistical metrics for numerical columns in a DataFrame. It returns key statistics such as the count, mean, standard deviation, minimum, maximum, and quartile values (25%, 50%, and 75%). For categorical data, using `df.describe(include='all')` provides additional information like the number of unique values, the most frequent value, and its frequency. This function is useful for quickly understanding the distribution and spread of the data in your DataFrame.
- Unique and null entries in the dataset.

```
AccountID                11260
Churn                        2
Tenure                      38
City_Tier                    3
CC_Contacted_LY             44
Payment                      5
Gender                       4
Service_Score                6
Account_user_count           7
account_segment              7
CC_Agent_Score               5
Marital_Status               3
rev_per_month               59
Complain_ly                  2
rev_growth_yoy              20
coupon_used_for_payment     20
Day_Since_CC_connect        24
cashback                  5693
Login_device                 3
dtype: int64
```

No. of unique entries in each column

```
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
```

No. of null entries in each column

5. Exploratory Data Analysis

a. Variable transformation

The most common payment method was Debit Card, with UPI being the least used, and no immediate issues were detected. In the gender field, unexpected values like ‘M’ and ‘F’ were corrected by mapping them to ‘Male’ and ‘Female.’ Account segments such as ‘Regular+’ and ‘Super+’ were standardized (e.g., ‘Regular +’ to ‘Regular Plus’). Most customers were married, with no issues found in marital status. The questionable value ‘&&&&’ in the login device field was investigated and replaced with “Other.”

Cleaning and correcting these categories will help ensure more accurate analysis and model performance.

- There are one or more values such as **'#','@','$','+'** and **'*'** present in columns '**Tenure', 'Account_user_count', 'rev_per_month', 'rev_growth_yoy', 'coupon_used_for_payment', 'Day_Since_CC_connect' and 'cashback'**.
- Replacing these values with np.nan in order to do the EDA.

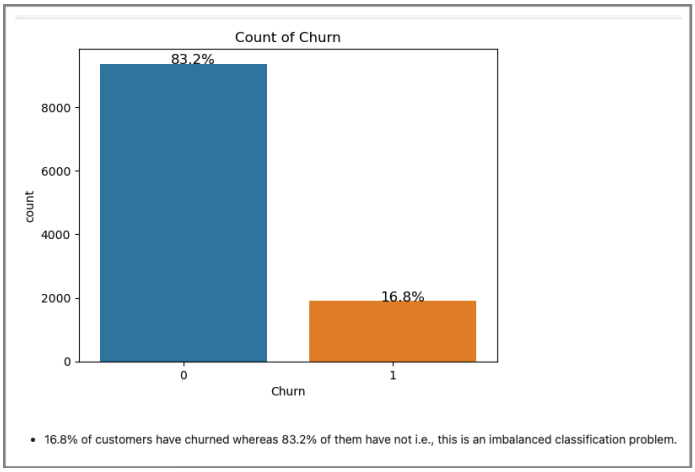
```
Unique values in Gender are :
Gender
Male      6704
Female    4448
Name: count, dtype: int64

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

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 correcting the values

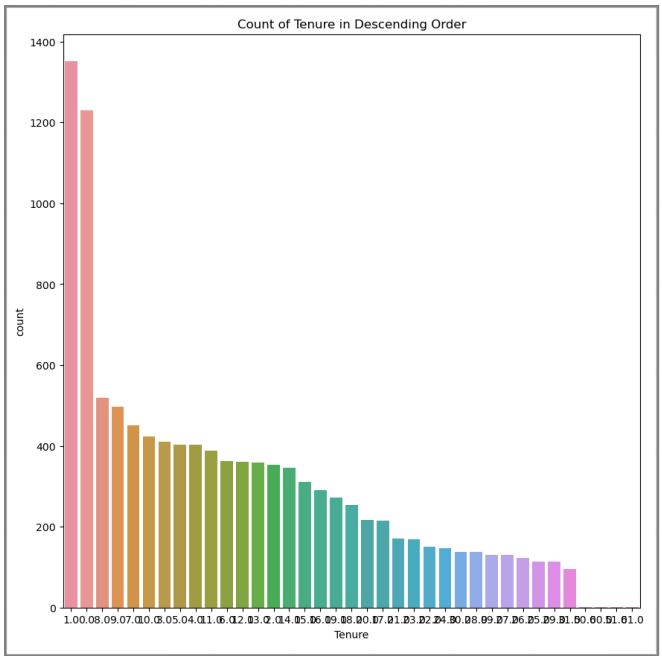
```
Unique values in Payment are :
Payment
Debit Card          4587
Credit Card         3511
E wallet            1217
Cash on Delivery    1014
UPI                  822
Name: count, dtype: int64


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

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 Marital_Status are :
Marital_Status
Married     5860
Single      3520
Divorced    1668
Name: count, dtype: int64


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

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

Unique values in the categorical columns

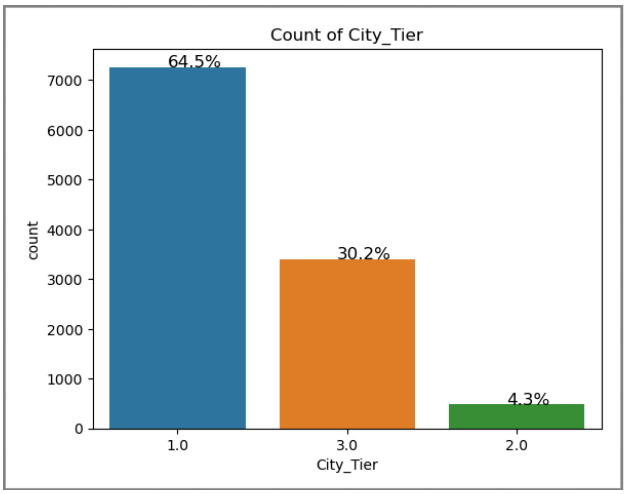
b. Univariate Analysis



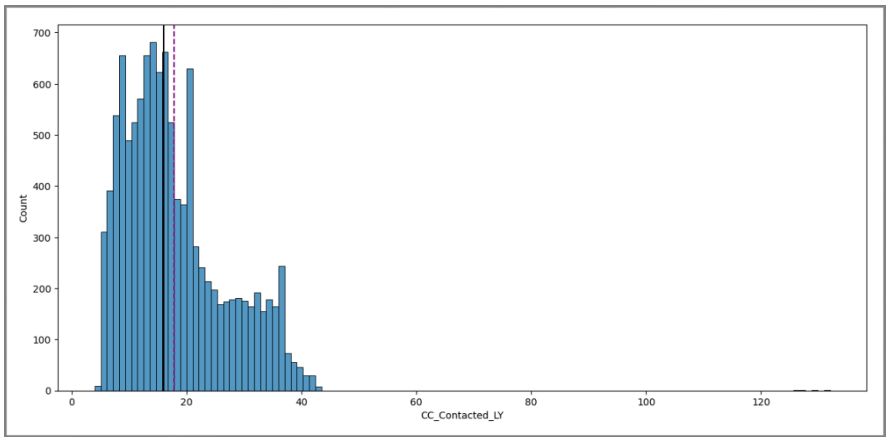
Churn



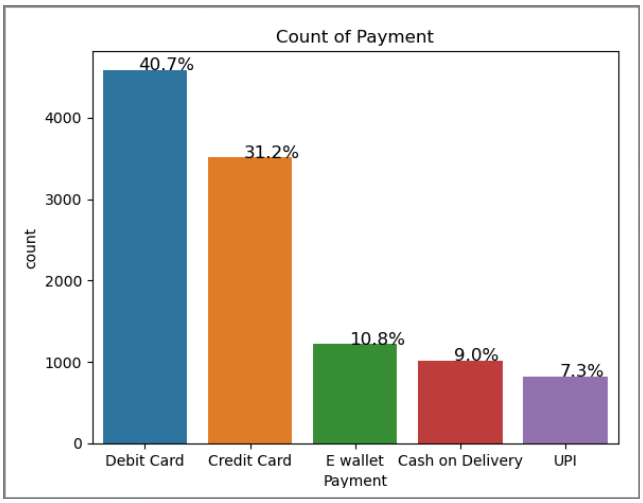
Tenure



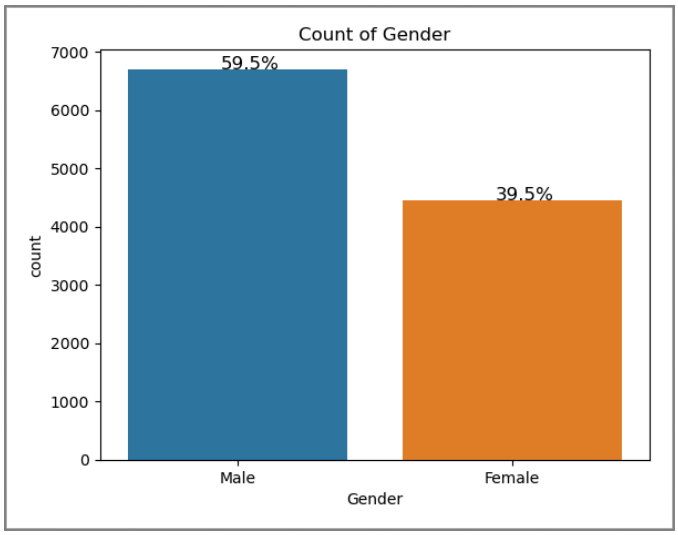
City_Tier



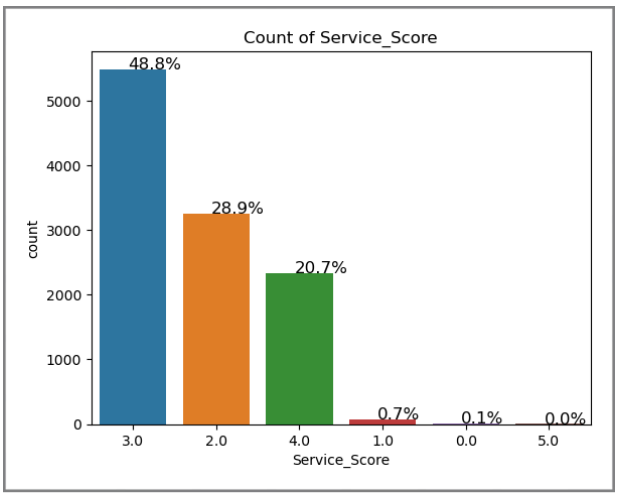
CC_Contacted_LY



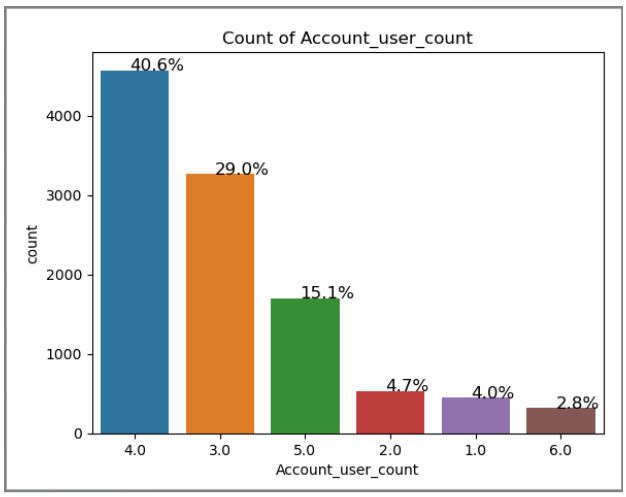
Payment



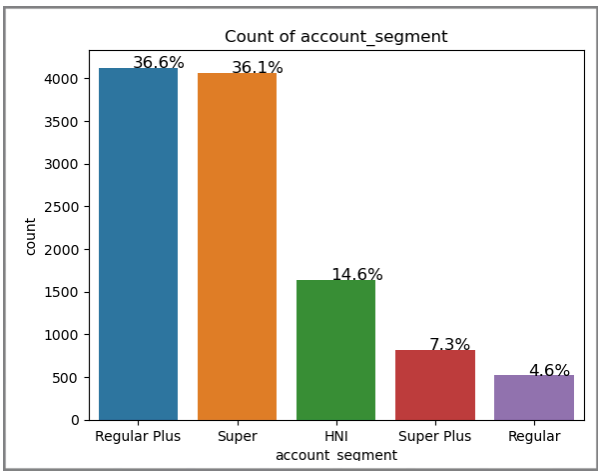
Gender



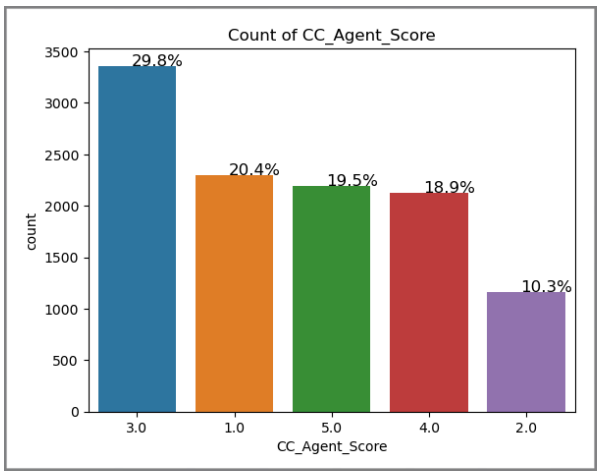
Service_Score



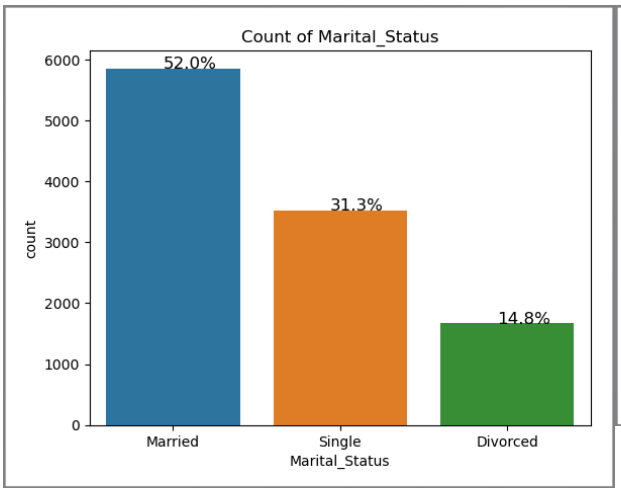
Account_user_count



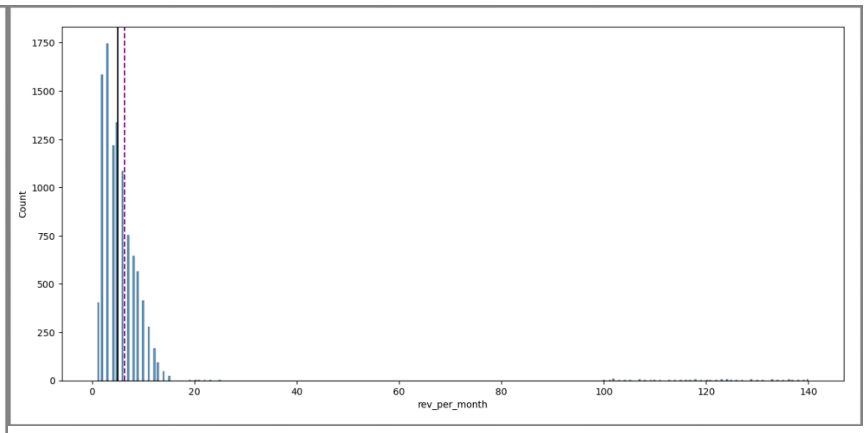
account_segment



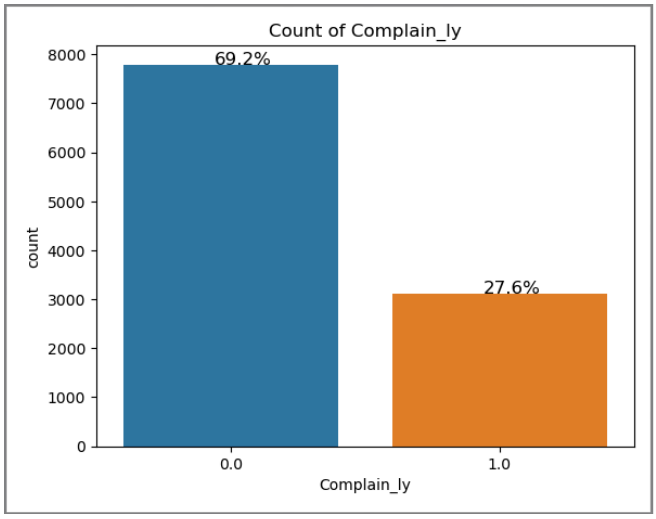
CC_Agent_Score



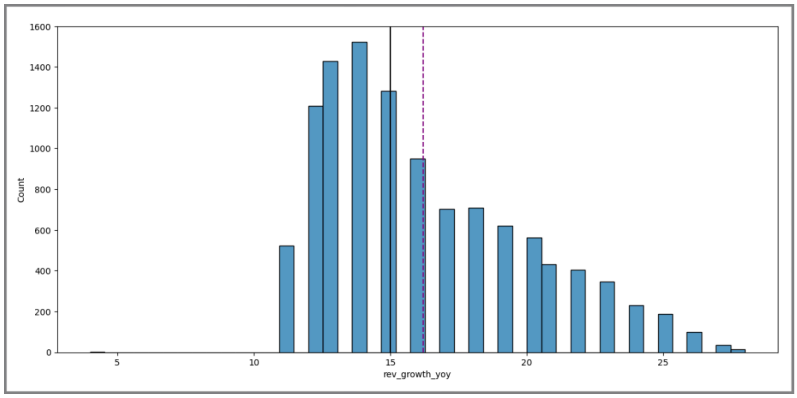
Marital_Status



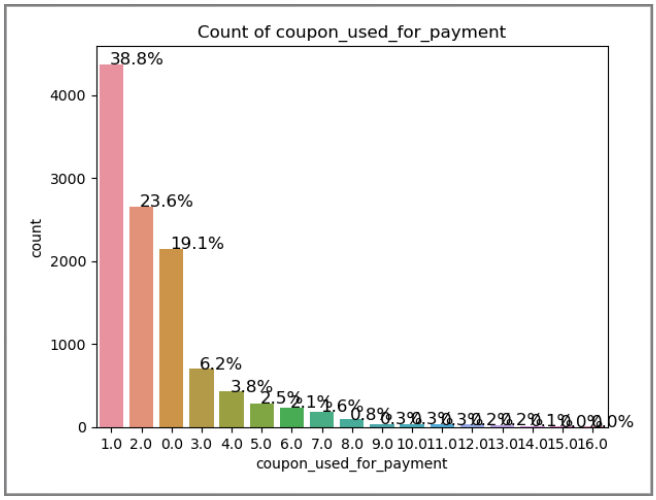
rev_per_month



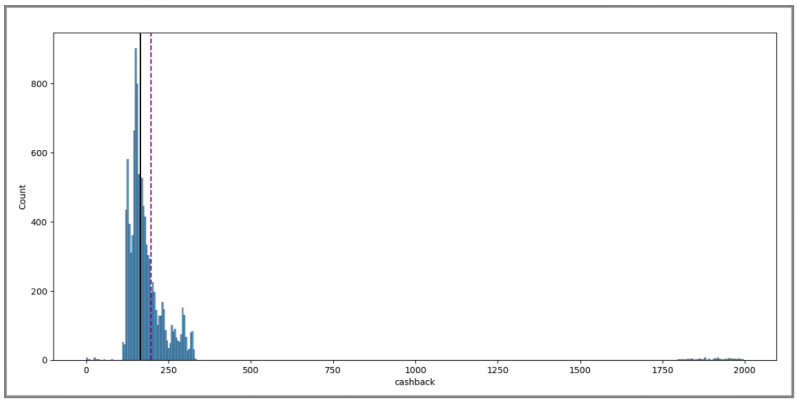
Complain_ly



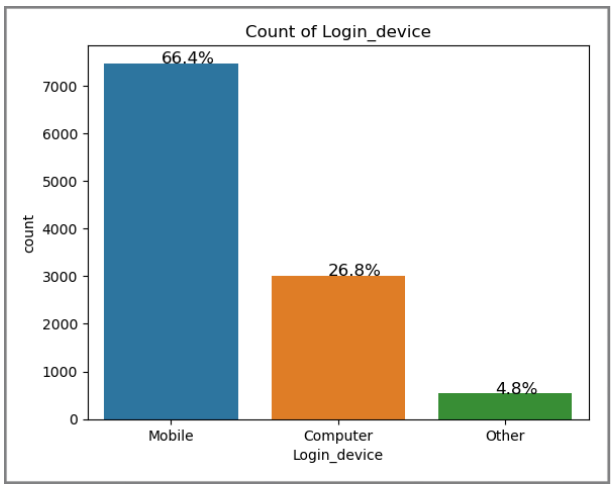
rev_growth_yoy



coupon_used_for_payment



Cashback



Login_device

Inferences from Univariate Analysis:

1. **Churn:** 16.8% of customers have churned, indicating an imbalanced classification problem.
2. **Tenure:** Highest number of customers have a tenure of less than a month.
3. **City_Tier:** Most customers are from Tier1 cities (7263).
4. **CC_Contacted_LY:** Higher contact frequency between 11 to 23 times.
5. **Payment:** 40.7% prefer Debit Card payments.

6. **Gender:** Male customers dominate at 59.5%.
7. **Service_score:** Highest service score is 3, indicating low satisfaction.
8. **Account_user_count:** Large number of accounts tagged with 4 users.
9. **Account_segment:** 'Regular Plus' and 'Super' segments have the most customers.
10. **CC_Agent_Score:** Highest Customer Agent Score is 3.
11. **Marital_Status:** 52% of primary account holders are married.
12. **rev_per_month:** Average revenue per month is around ₹6362K INR.
13. **Complain_ly:** 69.2% - No complain in last 12 months. 27.6% - Yes for complain in last 12 months.
14. **rev_growth_yoy:** On an average there is a 16% growth in revenue generated by the account in the past year compared to its previous year.
15. **coupon_used_for_payment:** Average usage is 1.8 times.
16. **Day_Since_CC_connect:** Highest reconnects within first three days.
17. **cashback:** Average cashback is near about ₹196 INR.
18. **Login_device:** Customers mostly use Mobile devices for login.

	Column	No. of outliers	Percentage of outliers
0	AccountID	0	0.000000
1	Churn	1896	16.838366
2	Tenure	139	1.234458
3	City_Tier	480	4.262877
4	CC_Contacted_LY	42	0.373002
5	Payment	0	0.000000
6	Gender	0	0.000000
7	Service_Score	90	0.799290
8	Account_user_count	1287	11.429840
9	account_segment	520	4.618117
10	CC_Agent_Score	0	0.000000
11	Marital_Status	0	0.000000
12	rev_per_month	185	1.642984
13	Complain_ly	0	0.000000
14	rev_growth_yoy	0	0.000000
15	coupon_used_for_payment	1380	12.255773
16	Day_Since_CC_connect	33	0.293073
17	cashback	879	7.806394
18	Login_device	539	4.786856

Outliers Percentage

Skewness and Kurtosis of each numerical column:

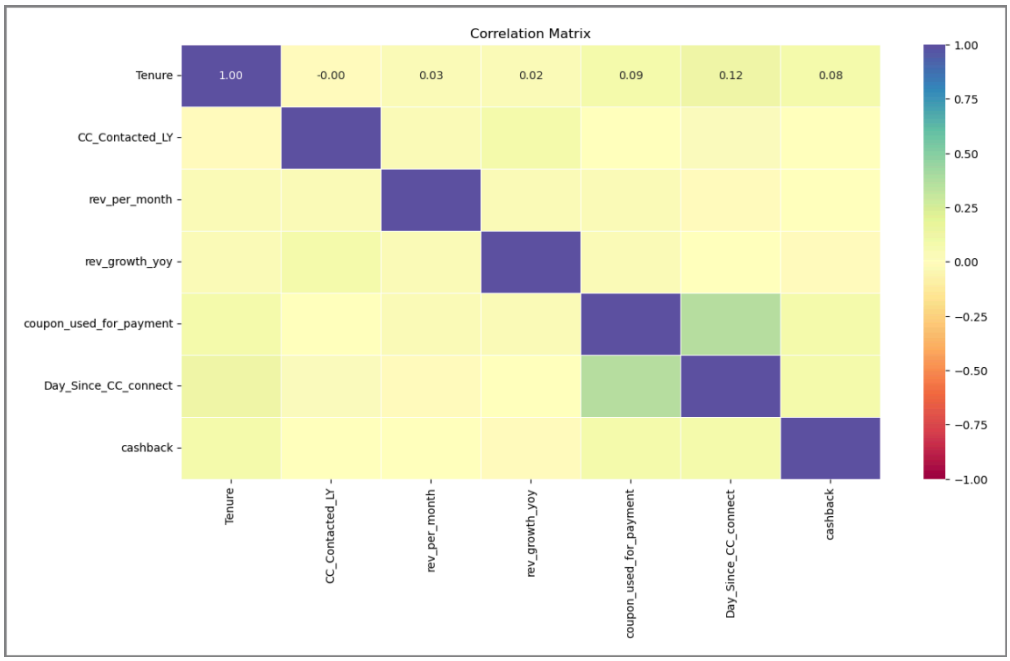
	Column	Skewness	Kurtosis
0	AccountID	0.00	-1.20
1	Churn	1.77	1.14
2	Tenure	3.90	23.37
3	CC_Contacted_LY	1.42	8.23
4	rev_per_month	9.09	86.96
5	rev_growth_yoy	0.75	-0.22
6	coupon_used_for_payment	2.58	9.10
7	Day_Since_CC_connect	1.27	5.33
8	cashback	8.77	81.11

Skewness and Kurtosis

b. Multivariate Analysis

1. **Churn:**
   - **Negative Correlation with Tenure:** Longer tenure reduces churn likelihood.
   - **Positive Correlation with Complain_ly:** Complaints increase churn likelihood.
   - **Negative Correlation with Day_Since_CC_connect:** Recent customer care contact reduces churn likelihood.
2. **Tenure:**

- **Positive Correlation with Day_Since_CC_connect:** Longer tenure



	Tenure	CC_Contacted_LY	rev_per_month	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback
Tenure	1.000000	-0.004261	0.028431	0.018824	0.089171	0.122612	0.078416
CC_Contacted_LY	-0.004261	1.000000	0.015675	0.072913	0.004969	0.012938	0.002679
rev_per_month	0.028431	0.015675	1.000000	0.024114	0.016548	-0.000923	0.002974
rev_growth_yoy	0.018824	0.072913	0.024114	1.000000	0.018341	0.002206	-0.001157
:oupon_used_for_payment	0.089171	0.004969	0.016548	0.018341	1.000000	0.361735	0.072861
Day_Since_CC_connect	0.122612	0.012938	-0.000923	0.002206	0.361735	1.000000	0.084465
cashback	0.078416	0.002679	0.002974	-0.001157	0.072861	0.084465	1.000000

means more days since last customer care contact.

- **Positive Correlation with coupon_used_for_payment:** Longer tenure slightly increases coupon usage.

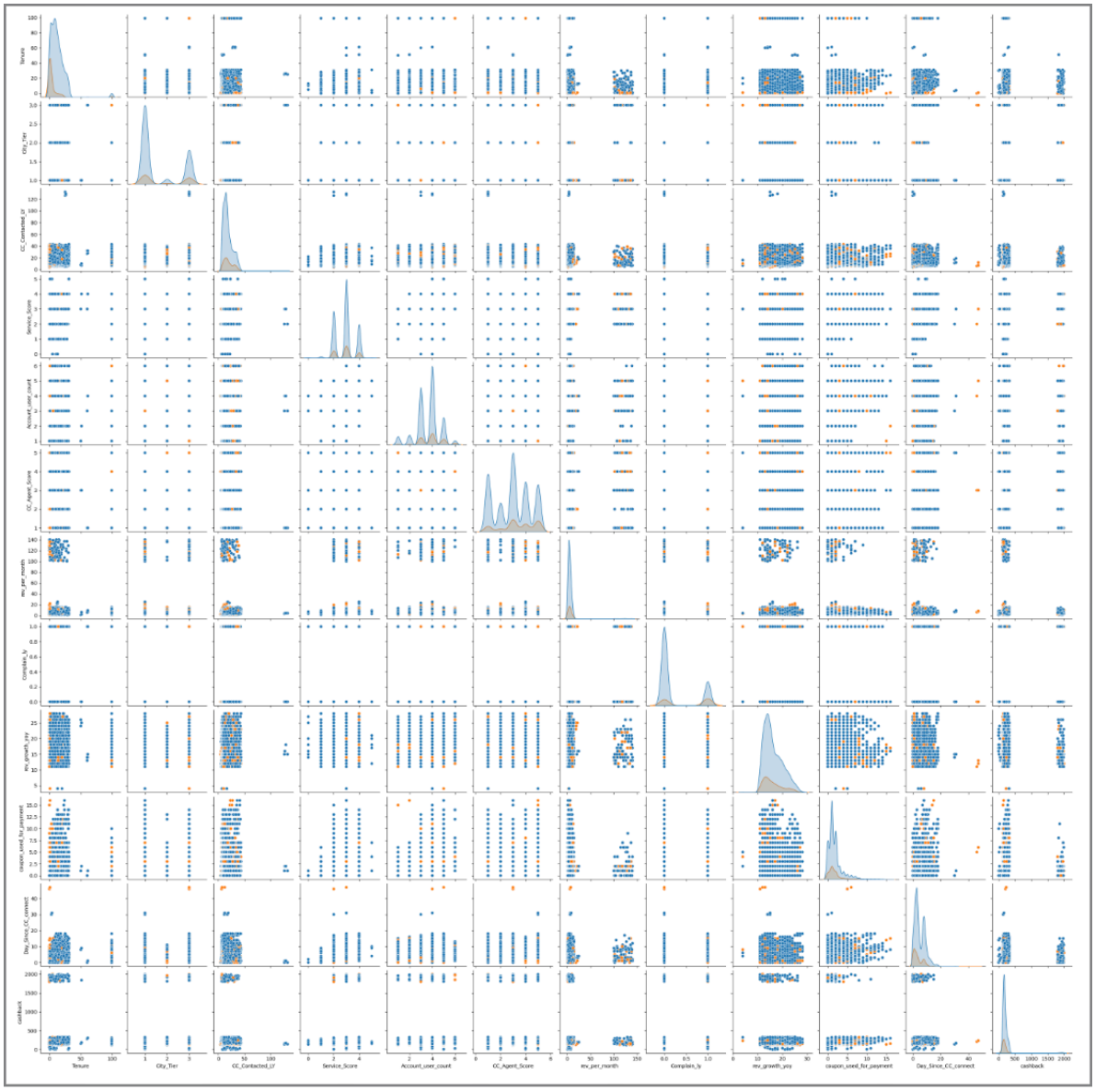
3. **City_Tier:**
   - **Low Correlation with other variables:** Minimal impact on other factors.
4. **CC_Contacted_LY:**
   - **Positive Correlation with Service_Score:** Better service scores for customers who contacted customer care.
   - **Slightly Positive Correlation with Rev_growth_yoy:** Marginal relationship between customer care contact and revenue growth.
5. **Service_Score:**
   - **Strong Positive Correlation with Account_user_count:** Higher scores for accounts with more users.
   - Positive Correlation with coupon_used_for_payment: Higher scores associated with more coupon usage.
6. **Account_user_count:**

- **Positive Correlation with coupon_used_for_payment:** More users in an account use more coupons.
- **Positive Correlation with Service_Score:** Larger accounts generally receive better service scores.

7. **CC_Agent_Score:**
   - **Low correlation with other variables:** Minimal direct influence on other factors.
8. **Rev_per_month:**
   - **Low correlation with other variables:** Relatively independent of other variables.
9. **Complain_ly:**
   - **Positive Correlation with Churn:** Complaints increase churn likelihood.
10. **Rev_growth_yoy:**
    - **Positive Correlation with Service_Score:** Better service scores slightly associated with revenue growth.
11. **coupon_used_for_payment:**
    - **Strong Positive Correlation with Day_Since_CC_connect:** Recent customer care contact increases coupon usage.
12. Day_Since_CC_connect:
    - **Positive Correlation with coupon_used_for_payment:** Recent contact associated with more coupon usage.
13. cashback:
    - **Low correlation with most variables:** Relatively independent in terms of influence on other factors.

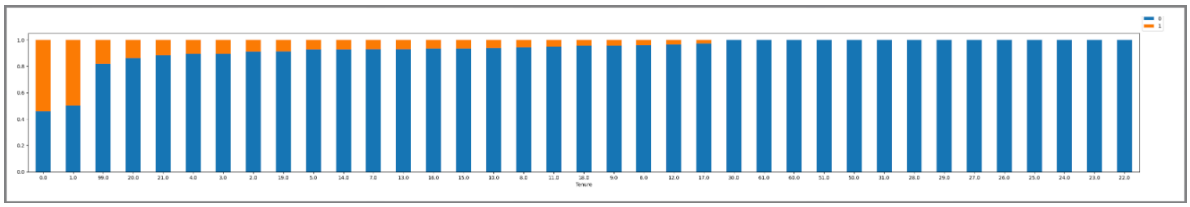
c. Pairplot

The pairplot shows that churn correlates with lower tenure and higher revenue per month. Features like revenue per month and cashback are highly skewed, with most data points at lower values and a few extreme outliers. A positive correlation exists between revenue per month, cashback, and coupon use, indicating that higher spenders receive more cashback and use more coupons. Tenure and Day_Since_CC_connect are also positively related, suggesting longer-tenure customers may have recently contacted customer care. Outliers in revenue, cashback, and tenure contribute to data skewness, which could impact model accuracy. Categorical features form distinct clusters but show some overlap, posing potential classification challenges.



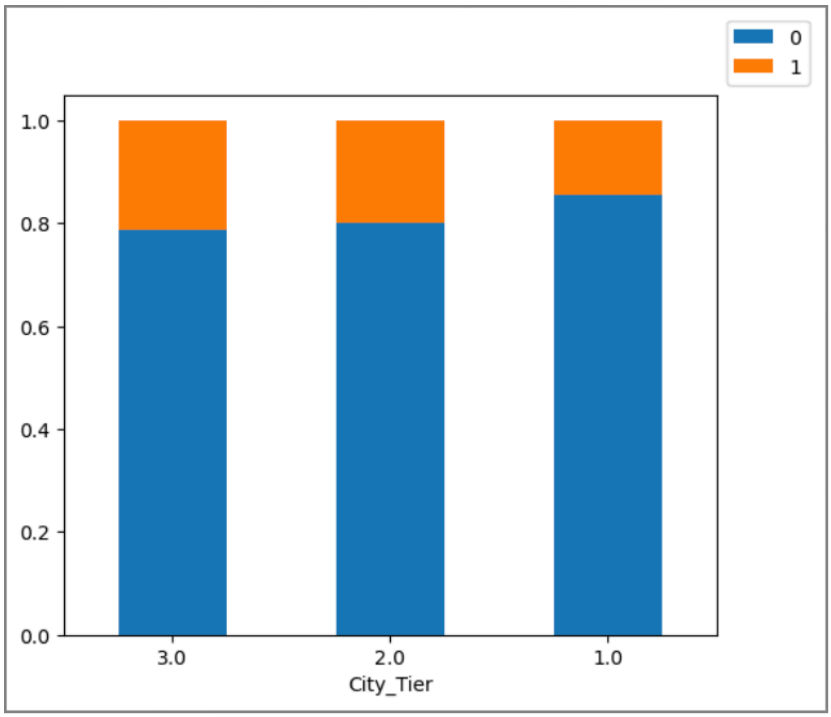
Pairplot

c) Bivariate Analysis Stacked Bar Plot: 1) Churn vs Tenure



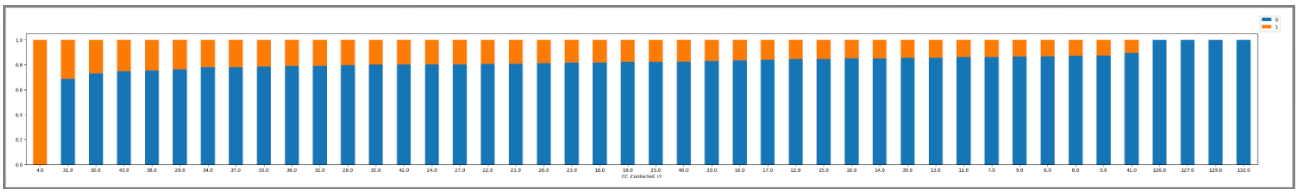
Customers with short tenures (1-3 months) have high churn rates, while churn decreases significantly for tenures above 7 months, stabilizing thereafter. Customers with over 20 months tenure show almost no churn, indicating strong loyalty. This highlights the importance of early retention efforts to reduce overall churn.

2) Churn vs City_Tier



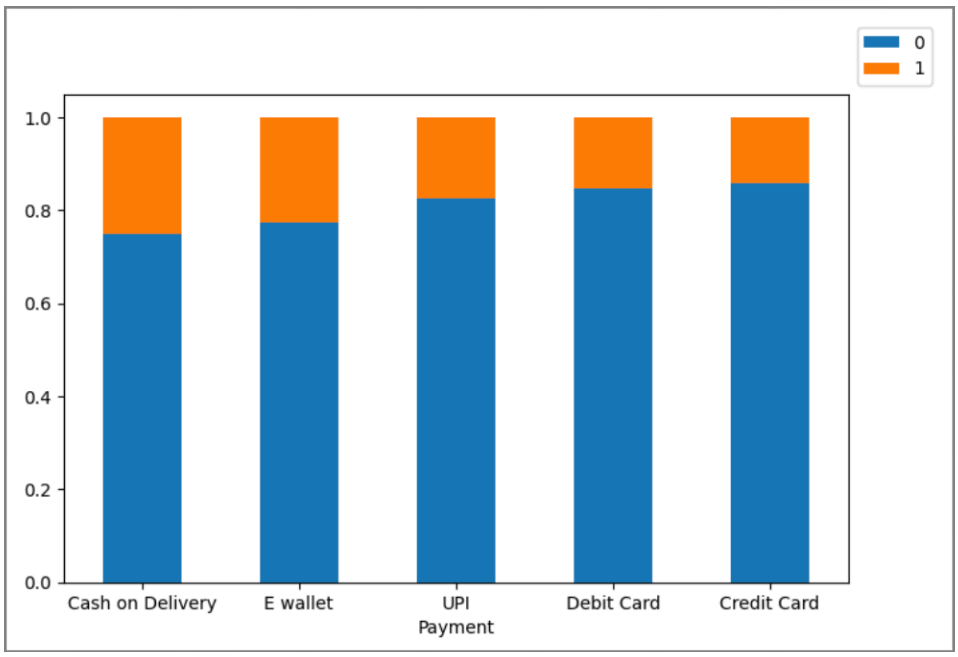
City Tier 1 has the highest retention, with most customers staying loyal (6207 out of 7263). City Tier 3 faces the highest churn risk, with 727 out of 3405 customers leaving. City Tier 2, though smaller, shows a noticeable churn rate (96 out of 480), indicating a need for targeted retention efforts. Overall, Tier 1 is the most stable, while Tier 3 poses the greatest churn challenge.

3) Churn vs CC_Contacted_LY



Churn rates generally increase with more customer care CC_Contacted_LY. Customers with 4-5 contacts show minimal churn. The data suggests that moderate customer care engagement may signal dissatisfaction leading to churn, while very frequent or rare contacts could reflect either serious issues or high satisfaction.

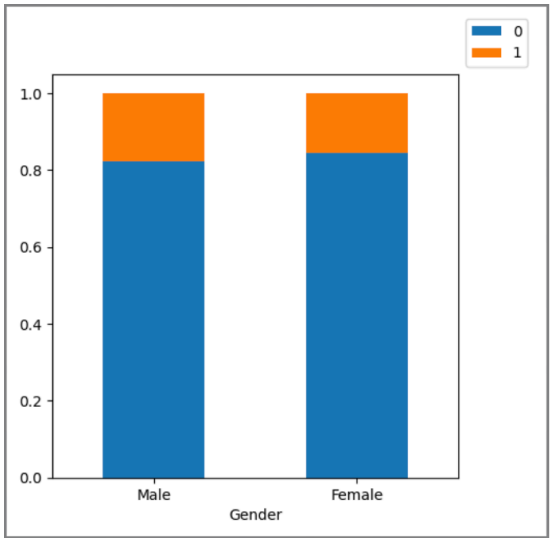
4) Churn vs Payment



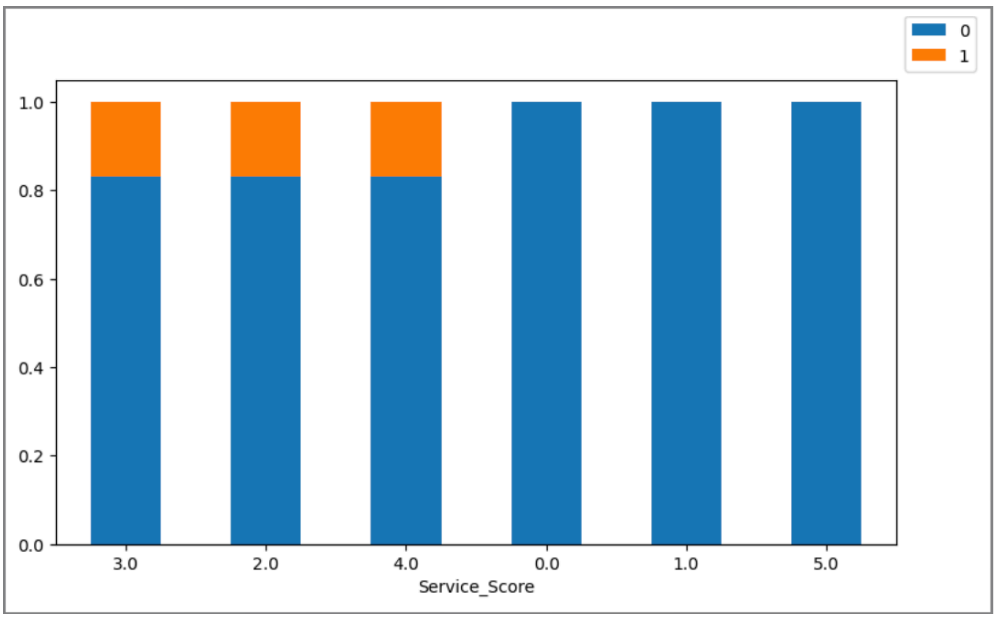
Among 11,151 customers, **Debit Card** users have the lowest churn rate at 15%, followed by **Credit Card** users at 14%. **E-wallet** and **Cash on Delivery** users show higher churn rates of 23% and 25%, respectively, while **UPI** users have a churn rate of 17%. This suggests that customers using alternative payment methods, like E-wallets and Cash on Delivery, are more likely to churn compared to those using Debit or Credit Cards.

5) Churn vs Gender

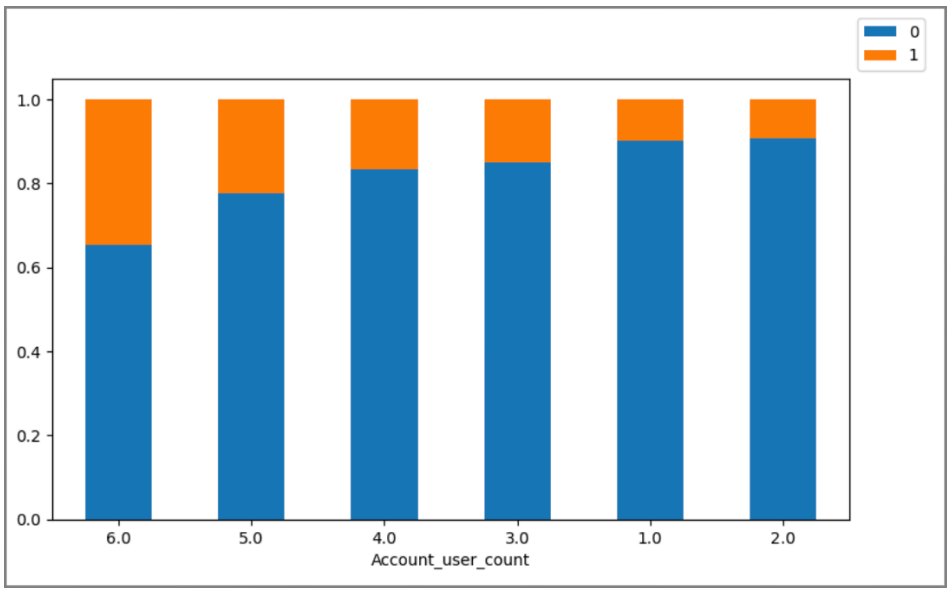
The churn rate for male customers is approximately 18% (1,185 out of 6,704), while for female customers it is about 15% (689 out of 4,448). This indicates that male customers tend to churn at a slightly higher rate than female customers, although both genders experience relatively close churn rates.



6) Churn vs Service_Score

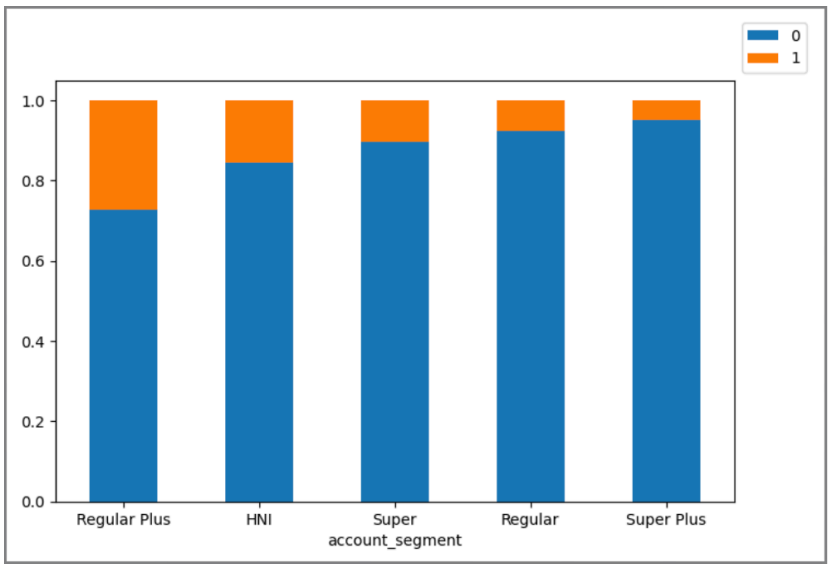


The largest group, with a Service Score of 3.0, has 5,490 customers and a churn rate of about 17% (936 out of 5,490). Similar churn rates of around 17% are observed for Service Scores of 2.0 and 4.0. For Service Scores of 0.0, 1.0, and 5.0, there are few customers and no recorded churn. Overall, churn rates are consistent across the main Service Score groups.



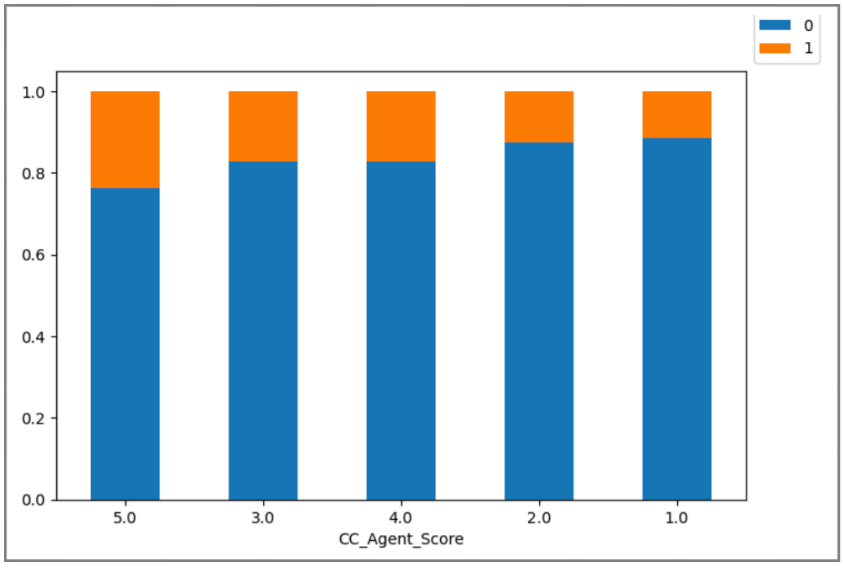
The largest group, with an Account User Count of 4.0, has 4,569 customers and a churn rate of about 17% (758 out of 4,569). The churn rate varies: 15% for Account User Count of 3.0, 22% for 5.0, and significantly higher at 35% for 6.0. Lower account counts (2.0 and 1.0) have lower churn rates. Overall, churn rates tend to increase with higher account user counts.

8) Churn vs account_segment



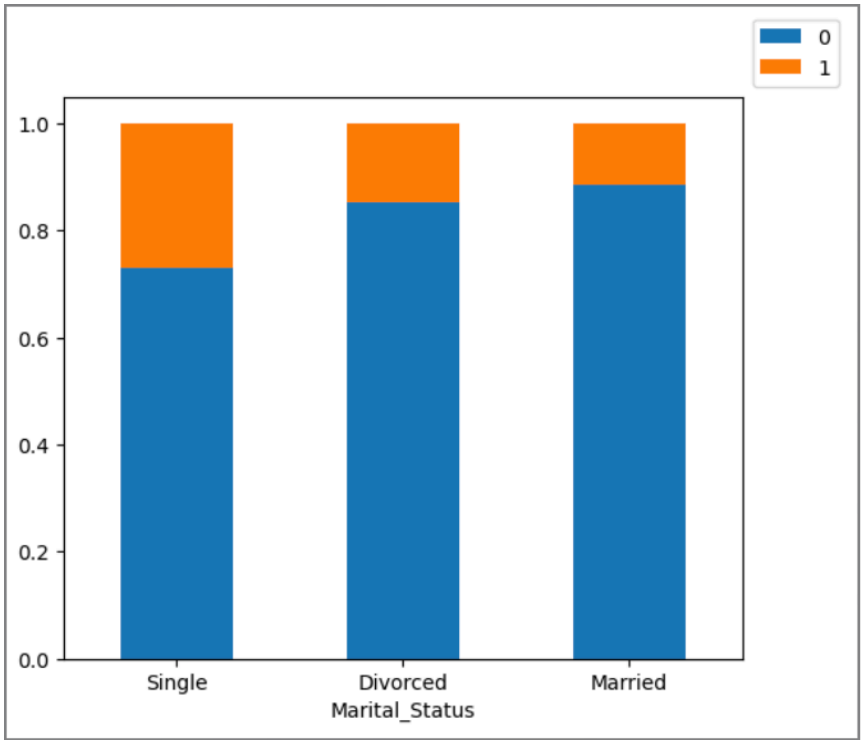
Among 11,151 customers, Debit Card users have the lowest churn rate at 15%, followed by Credit Card users at 14%. E-wallet and Cash on Delivery users show higher churn rates of 23% and 25%, respectively, while UPI users have a churn rate of 17%. This suggests that customers using alternative payment methods, like E-wallets and Cash on Delivery, are more likely to churn compared to those using Debit or Credit Cards.

9) Churn vs CC_Agent_Score



The churn data by CC_Agent_Score shows: 3.0 (17% churn, 577 out of 3,360), 5.0 (24% churn, 522 out of 2,191), 4.0 (17% churn, 364 out of 2,127), 1.0 (11% churn, 264 out of 2,302), and 2.0 (13% churn, 147 out of 1,164). Higher scores correlate with higher churn rates.

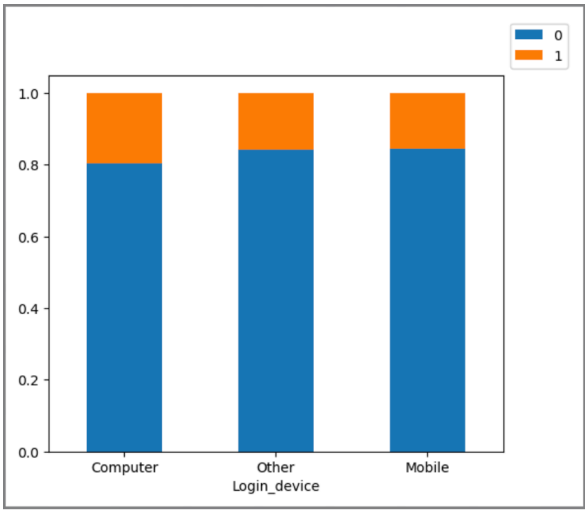
10) Churn vs Marital_Status



Single customers have a 27% churn rate, while married customers have an 11.5% churn rate. Divorced customers fall in between with a 14.6% churn rate. Single customers churn the most, while married customers are the most stable.

11) Churn vs Login_device

Most customers who churned used mobile devices (1,172 out of 1,854), followed by computer users (597 out of 3,018). The "Other" category had the lowest churn. This indicates that mobile users, despite being the largest segment, also have a significant churn rate, highlighting a potential area for targeted retention efforts.



Inferences from Univariate, Bivariate and Multivariate:

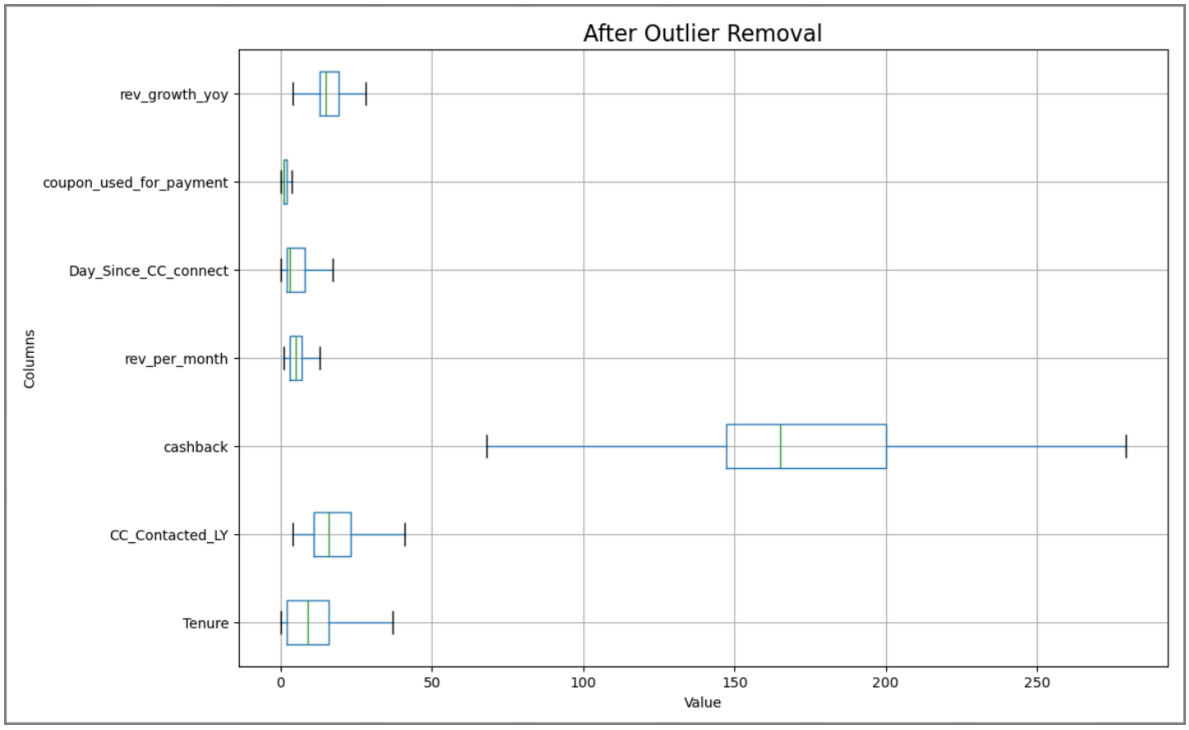
- City Tier 1 has the highest retention rate, with a majority of customers not churning.
- Male customers are more likely to churn than female customers.
- Regular Plus segment has the highest churn rate, while Super Plus segment has the lowest churn rate.
- Mobile users have a higher churn rate compared to computer users.
- Customers who have complained in the last year are more likely to churn.
- Longer-tenured customers are less likely to churn.
- Customers with more recent customer care connections are less likely to churn.
- Most customers rated services and customer care interactions as "3", indicating a neutral sentiment.
- Transaction via UPI and e-wallet is very low, suggesting a lack of adoption of these payment methods.
- Customers with marital status "single" contribute max towards churn, indicating a higher risk of churn among single customers.
- Any complaints raised in last 12 months doesn't show any impact toward churn, suggesting that complaints may not be a key driver of churn.
- Tenure and cashback are directly proportional to each other, indicating that longer-tenured customers receive more cashback.
- Computer usage is more in Tier 1 city followed by Tier 3 and Tier 2 city, highlighting differences in device usage across cities.

Removal of unwanted variables

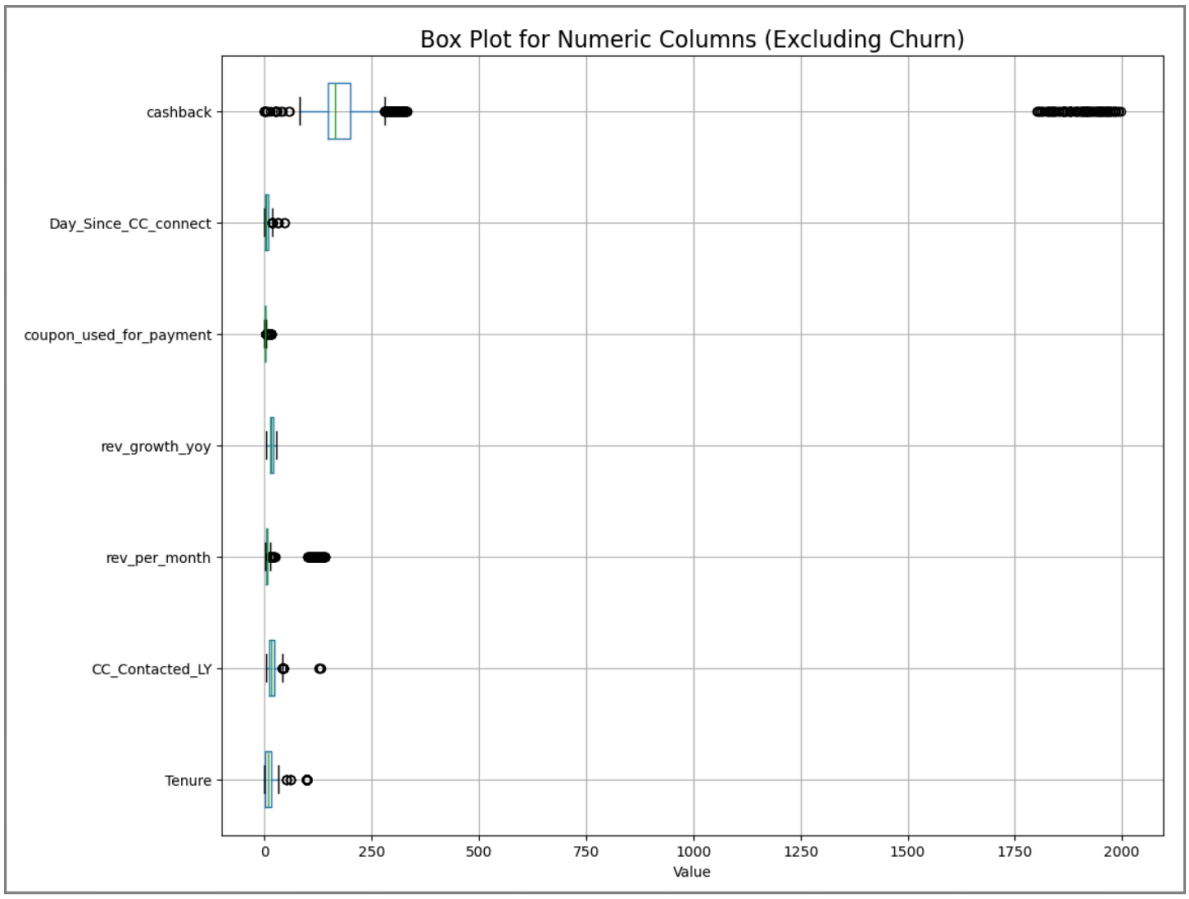
Removing the AccountID column from the dataset is appropriate since it contains unique values for each record and does not contribute to meaningful analysis or prediction. Unique identifiers like **AccountID** are primarily used to distinguish individual records but do not provide insights into patterns or relationships within the data. By excluding this column, we can streamline the dataset, reduce computational complexity, and focus on variables that have actual predictive value or informative content.

3. Data Cleaning and Pre-processing

Outlier treatment: The dataset includes both continuous and categorical variables. Since categorical variables represent different customer types, it doesn't make sense to apply outlier treatment to them. Therefore, outlier treatment will only be applied to the continuous variables.



After outlier treatment



Before outlier treatment

Missing value treatment:

To handle missing values, we will use the KNN imputer with n_neighbors=5. First, we will encode the categorical variables into numerical values to enable the imputer to calculate distances between data points effectively. This approach allows us to estimate and fill in missing values based on the similarity of data points within the dataset.

This process involves converting categorical variables into numerical representations, which is necessary for the KNN algorithm to compute distances accurately and impute missing values effectively.

Encoded Dataset:

	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_St
0	1	4.0	3.0	6.0	0.0	0.0	3.0	3.0	0.0	2.0	
1	1	0.0	1.0	8.0	1.0	1.0	3.0	4.0	1.0	3.0	
2	1	0.0	1.0	30.0	0.0	1.0	2.0	4.0	1.0	3.0	
3	1	0.0	3.0	15.0	0.0	1.0	2.0	4.0	0.0	5.0	
4	1	0.0	1.0	12.0	2.0	1.0	2.0	3.0	1.0	5.0	

Encoded Dataset

```
AccountID                0.00
Churn                    0.00
Tenure                   1.94
City_Tier                0.99
CC_Contacted_LY          0.91
Payment                  0.97
Gender                   0.96
Service_Score            0.87
Account_user_count       3.94
account_segment          0.86
CC_Agent_Score           1.03
Marital_Status           1.88
rev_per_month            7.02
Complain_ly              3.17
rev_growth_yoy           0.03
coupon_used_for_payment  0.03
Day_Since_CC_connect     3.18
cashback                 4.20
Login_device             1.96
dtype: float64
```

List of missing value in percentage

```
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
--------------------------------
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
```

After imputing, there is no null values

Variable transformation

The variables in the dataset have different scales, such as "Cashback" representing currency and "CC_Agent_Score" representing customer ratings, leading to variations in their statistical measures. To address this, scaling is necessary to normalize the data. Variable transformation is a crucial step in data preprocessing that involves modifying variables to improve the performance of a model or to meet the assumptions of statistical techniques. To scale features, the `StandardScaler` should be fitted only on the training data to learn the scaling parameters (mean and standard deviation close to 1), and then these parameters are used to transform both the training and test datasets. The correct approach is to use `sc.fit_transform(X_train)` to fit and scale the training data, and then apply `sc.transform(X_test)` to scale the test data based on the same parameters. This ensures that the test data is scaled consistently with the training data, avoiding data leakage and ensuring accurate model evaluation.

gent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
-1.544121	-1.353581	1.277646	1.565011	-1.116150	-0.439236	-1.256704	-0.231599	-0.644715
-0.811490	-0.227929	-0.769584	1.565011	-0.299957	1.371880	2.020236	0.172123	-0.644715
1.386403	0.897722	-0.087174	1.565011	-0.572021	-0.439236	-0.437469	0.031917	1.126238
-1.544121	0.897722	0.254031	-0.653270	-0.844086	1.824659	0.654845	1.227724	-0.644715
1.386403	-1.353581	0.936441	-0.653270	-0.844086	-0.439236	-0.437469	-0.567083	1.126238

Scaled test dataset

	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Stat
0	-1.148496	1.489238	-0.215234	-0.300658	-1.233817	-1.251189	-0.752048	-0.939742	-0.038437	0.8825
1	0.059570	-0.711240	-0.681723	-1.030607	0.815588	1.529487	1.406007	-0.152131	1.413271	0.8825
2	0.529374	1.489238	-0.565101	-1.030607	-1.233817	1.529487	0.326979	1.423089	-0.764291	-1.3616
3	0.976806	1.489238	1.767344	1.889190	0.815588	0.139149	1.406007	-0.939742	1.413271	0.8825
4	0.529374	-0.711240	-0.448478	-1.030607	-1.233817	0.139149	1.945521	0.635479	-0.038437	0.8825

Scaled train dataset

Also converted all the columns to float datatype.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8445 entries, 0 to 8444
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Tenure                    8445 non-null   float64
 1   City_Tier                 8445 non-null   float64
 2   CC_Contacted_LY           8445 non-null   float64
 3   Payment                   8445 non-null   float64
 4   Gender                    8445 non-null   float64
 5   Service_Score             8445 non-null   float64
 6   Account_user_count        8445 non-null   float64
 7   account_segment           8445 non-null   float64
 8   CC_Agent_Score            8445 non-null   float64
 9   Marital_Status            8445 non-null   float64
 10  rev_per_month             8445 non-null   float64
 11  Complain_ly               8445 non-null   float64
 12  rev_growth_yoy            8445 non-null   float64
 13  coupon_used_for_payment   8445 non-null   float64
 14  Day_Since_CC_connect      8445 non-null   float64
 15  cashback                  8445 non-null   float64
 16  Login_device              8445 non-null   float64
dtypes: float64(17)
memory usage: 1.1 MB
```

Addition of new variables

Adding new variables is not required for the customer churn dataset, as the existing features likely provide adequate information for predicting churn. Introducing extra variables might complicate the analysis without offering significant benefits, potentially leading to overfitting or increased complexity.

4. Model Building

Based on the visual and non-visual analysis, it appears that we are dealing with a classification problem where the target variable needs to be categorized as either "Yes" or "No." As a data analyst, we have several algorithms at our disposal to develop a model that predicts whether a given customer will churn or not. These algorithms include:

1. **Logistic Regression:** Logistic Regression is a supervised machine learning algorithm used to model the probability of a specific class or event. It is typically applied when the data is linearly separable, and the outcome is binary or dichotomous. In essence, logistic regression is commonly used for binary classification tasks.
2. **Linear Discriminant Analysis (LDA):** Linear Discriminant Analysis (LDA) is a predictive modeling algorithm designed for multi-class classification. It can also serve as a dimensionality reduction technique, projecting the training data in a way that maximally separates the examples based on their assigned classes.
3. **K-Nearest Neighbors (KNN):** K-Nearest Neighbors (KNN) is a simple, non-parametric classification algorithm that predicts the target variable for a new instance by identifying the k most similar instances (nearest neighbors) in the training dataset. The prediction is based on the majority vote of these neighbors, where the new instance is classified according to the most common class among its k nearest neighbors. This method relies on the assumption that similar data points are likely to have similar outcomes, making it effective for various classification tasks.
4. **Gaussian Naive Bayes:** Gaussian Naive Bayes is a probabilistic classification algorithm that assumes feature independence and uses Bayes' theorem. It assumes features follow a Gaussian distribution, making it suitable for continuous data and high-dimensional datasets. This method is simple and efficient, often used in various classification tasks.
5. **Random Forest:** Random forest is an algorithm that generates a 'forest' of decision trees. It then takes these many decision trees and combines them to avoid overfitting and produce more accurate predictions.
6. **Bagging - Random Forest:** Bagging (Bootstrap Aggregating) in Random Forest involves creating multiple decision trees from random subsets of

the training data and then combining their predictions to reduce variance and improve overall model accuracy.

7. **Gradient Boost:** Gradient Boosting is an ensemble learning method that combines multiple weak models (decision trees) to create a strong predictive model. It iteratively trains trees to correct the errors of previous trees, using gradients to optimize the process, resulting in improved accuracy and robustness.
8. **XGBoost:** XGBoost (Extreme Gradient Boosting) is an optimized version of gradient boosting that uses advanced techniques like regularization, parallel processing, and handling missing values to improve performance and efficiency. It is known for its speed and accuracy in handling large datasets and complex problems.
9. **Ada-Boosting:** AdaBoost (Adaptive Boosting) is an ensemble learning method that combines multiple weak models (decision trees) to create a strong predictive model. It iteratively trains trees, giving more weight to misclassified instances, to improve overall accuracy and handle complex datasets effectively.
10. **Support Vector Machine (SVM):** Support Vector Machine (SVM) is a classification algorithm that finds the best hyperplane to separate classes in the feature space, maximizing the margin between them. It can handle high-dimensional data and non-linear relationships using kernel functions, making it effective for various classification tasks.

Splitting Data into Train and Test Dataset

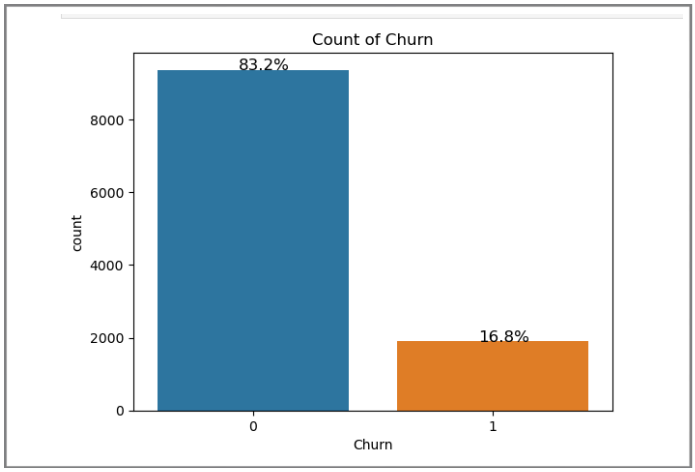
Data is split into training and testing sets in a **70:30 ratio**, adhering to standard market practice. Various models are built and trained using the training data, and their performance is evaluated by measuring their accuracy on the testing data.

Shape of Train and Test dataset

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

Dimension of Train and Test dataset

Imbalance Data



Churn

- 16.8% of customers have churned, indicating an imbalanced classification problem.

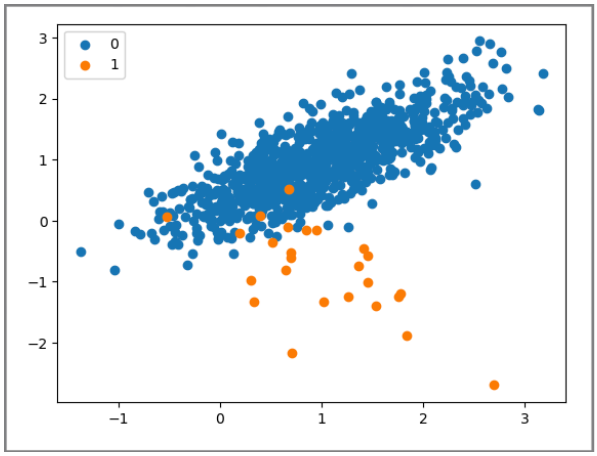
Balancing the Dataset: To address the imbalance, we use the SMOTE technique to generate additional data points, ensuring the data is balanced. SMOTE should only be applied to the training dataset, not the test dataset. The data is divided into training and test sets in a 70:30 ratio, following standard practice (though this ratio can be adjusted as needed).

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

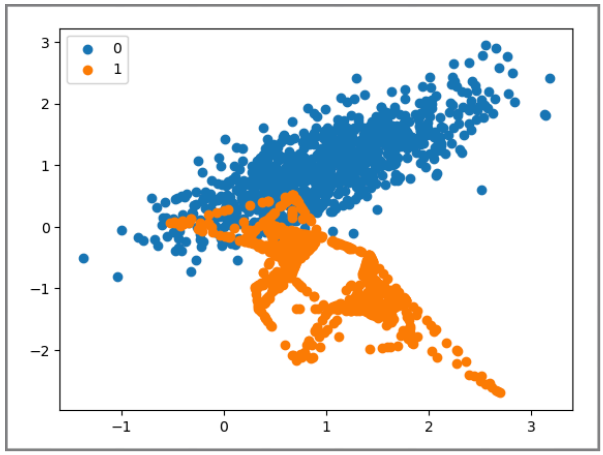
Before SMOTE

```
X_train_res (13110, 17)
y_train_res (13110,)
```

After SMOTE



Before SMOTE



After SMOTE

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

Approach is to observe model performance across various algorithms with their default parameters, then to check on model performance with tuning into different hyperparameters. Also, will observe model performance on balanced data-set to check if that out performs over imbalanced data-set.

After building various models and analysing various parameters we conclude that "**Random Forest**" with default values out performs all other models built. We have concluded this basis Accuracy, F1 score, Recall, Precision and AUC score.

i. Building Random Forest Model with default parameters

Post splitting data into training and testing data set we fitted Random Forest model into training dataset and performed prediction on training and testing dataset using the same model. We made the first model with default hyperparameters with default value with 100 decision trees (`n_estimators=100`) and a fixed random state (`random_state=42`) to ensure reproducibility of the results.

```
Training Accuracy: 1.0
Training Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      1327

    accuracy                           1.00      7882
   macro avg       1.00      1.00      1.00      7882
weighted avg       1.00      1.00      1.00      7882

Training Confusion Matrix:
[[6555    0]
 [   0 1327]]
```

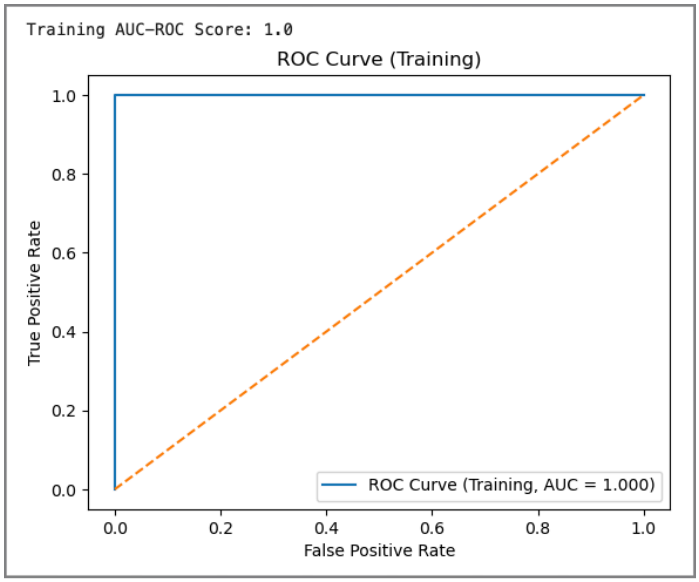
Random Forest for Train dataset performance- Accuracy, classification report and confusion matrix

```
Accuracy: 0.9641799881586738
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      2809
           1       0.95      0.83      0.89       569

    accuracy                           0.96      3378
   macro avg       0.96      0.91      0.93      3378
weighted avg       0.96      0.96      0.96      3378

Confusion Matrix:
[[2786   23]
 [  98  471]]
```

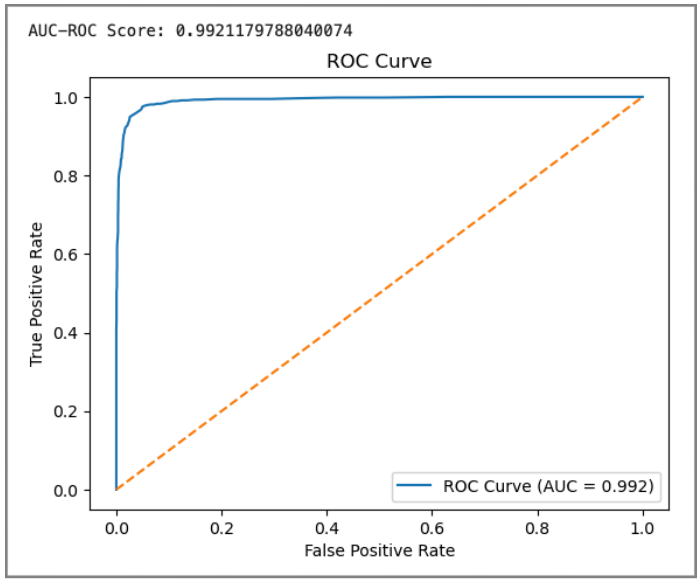
Random Forest for Test dataset performance- Accuracy, classification report and confusion matrix

AUC Score and ROC Curve

The Random Forest model performs extremely well on both the training and testing sets, with high accuracy, precision, recall, and F1-scores. The model is able to capture the underlying patterns in the data and make accurate predictions. The use of the original dataset without any preprocessing or feature engineering has not negatively impacted the model's performance.

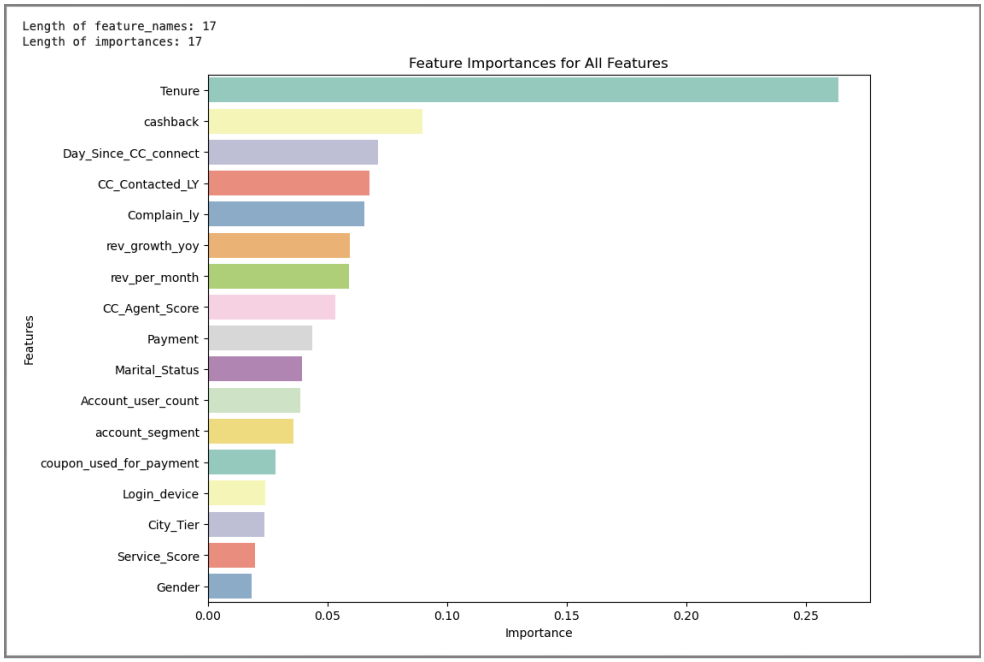


AUC Score and ROC Curve - Training dataset



AUC Score and ROC Curve - Testing dataset

Feature Importance for Random Forest



Feature Importance w.r.t. Random Forest with default parameter

The top features influencing customer churn prediction are:

- **Tenure:** The most significant feature, indicating that longer-tenured customers are less likely to churn.
- **Cashback:** Cashback offers play a crucial role in retention, as customers receiving rewards are less likely to churn.

- **Day_Since_CC_connect:** The number of days since the customer last connected with customer service is important, with more recent interactions lowering the likelihood of churn.
- **CC_Contacted_LY:** Whether the customer contacted customer care in the past year has a notable impact, suggesting that frequent engagement might indicate dissatisfaction or issues leading to churn.
- **Complain_ly:** The number of complaints lodged in the past year is another strong factor, as higher complaints can signal dissatisfaction.
- **rev_growth_yoy:** Year-over-year revenue growth is significant, meaning that customers contributing to higher revenue are less likely to churn.
- **rev_per_month**: Monthly revenue contribution is also a key indicator, with higher spending customers likely to stay longer.
- **CC_Agent_Score:** Customer ratings of call center agents affect churn, suggesting that better service scores correlate with retention.
- **Payment:** The payment method or frequency has a moderate influence, hinting at how billing experience can impact loyalty.
- **Marital_Status:** Marital status shows some effect, potentially due to lifestyle differences in service usage.
- **Account_user_count:** The number of users on the account plays a role, with larger accounts being less likely to churn.
- **account_segment:** The customer segment or tier influences retention, with certain segments being more stable than others.
- **coupon_used_for_payment:** The use of coupons for payment can indicate price sensitivity, impacting churn likelihood.
- **Login_device:** The type of device used for login has a minimal but notable influence on churn, possibly linked to customer engagement behavior.
- **City_Tier:** Has relatively low importance.
- **Service_Score:** The service score, while lower on the list, still impacts churn. Higher satisfaction scores correlate with reduced churn.
- **Gender:** Gender has the least influence, suggesting that it does not significantly affect churn predictions in this dataset.

This analysis helps in understanding which features are most influential in predicting customer churn, enabling more targeted strategies for customer retention.

Comparison: The comparison between Random Forest with default parameters and Random Forest with SMOTE shows that the SMOTE model slightly outperforms the default model in terms of accuracy (0.97 vs 0.96) and precision (0.98 vs 0.97), while maintaining similar recall and F1-score (both 0.99). Other metrics also indicate improvements with SMOTE, such as higher values for Metric 6 (0.89 vs 0.83) and Metric 7 (0.91 vs 0.89), although Metric 5 is slightly lower (0.94 vs 0.95). Overall, using SMOTE enhances the model's performance, particularly in handling class imbalance.

ii. Random Forest on the SMOTE dataset

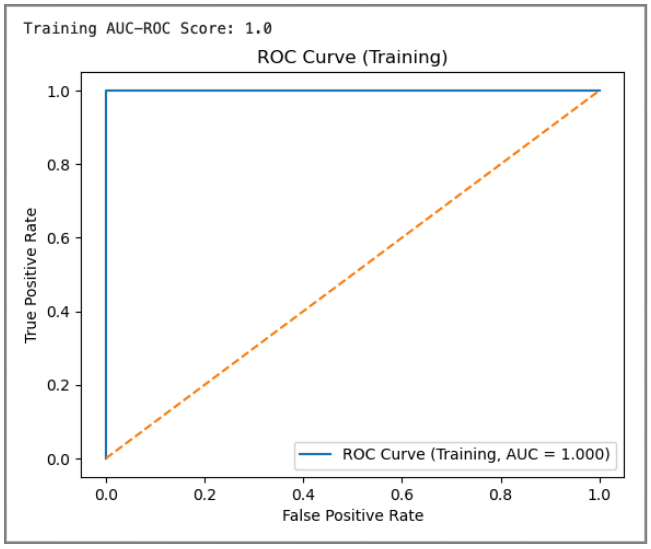
```
Training Accuracy: 1.0
Training Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      6555

    accuracy                           1.00     13110
   macro avg       1.00      1.00      1.00     13110
weighted avg       1.00      1.00      1.00     13110

Training Confusion Matrix:
[[6555    0]
 [   0 6555]]
```

Random Forest on SMOTE dataset - Train dataset performance- Accuracy, classification report and confusion matrix

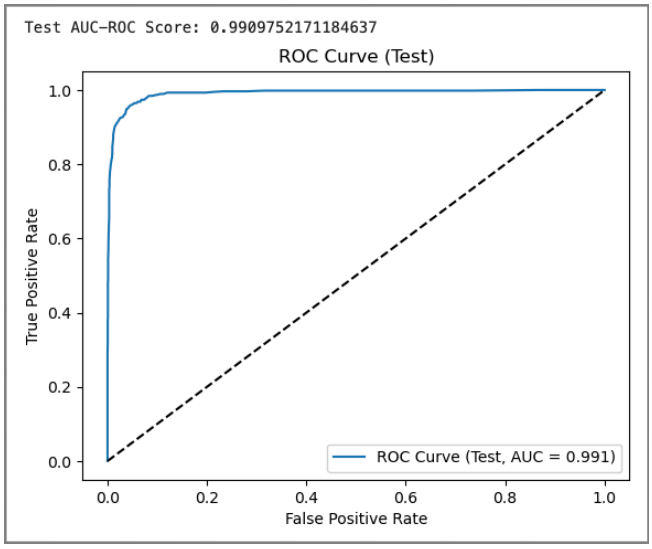
```
Test Accuracy: 0.9706927175843695
Test Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      2809
           1       0.94      0.89      0.91       569

    accuracy                           0.97      3378
   macro avg       0.96      0.94      0.95      3378
weighted avg       0.97      0.97      0.97      3378

Test Confusion Matrix:
[[2775   34]
 [  65  504]]
```

Random Forest on SMOTE dataset - Test dataset performance- Accuracy, classification report and confusion matrix

AUC Score and ROC Curve:



Random Forest on SMOTE dataset - Train dataset performance - ROC Curve



Random Forest on SMOTE dataset - Test dataset performance - ROC Curve

Random Forest on the SMOTE dataset involves applying the Random Forest algorithm to a dataset that has been balanced using the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE generates synthetic samples for the minority class, addressing class imbalance, while Random Forest builds an ensemble of decision trees to improve classification accuracy. This combination enhances model performance on imbalanced datasets, ensuring better predictions for both majority and minority classes, making it especially useful in applications like fraud detection and medical diagnoses.

iii. Bagging with Random Forest on original dataset

Bagging (Bootstrap Aggregating) is a key ensemble learning technique used in Random Forests. It helps improve the stability and accuracy of machine learning models by reducing variance and preventing overfitting.

```
Training Accuracy: 0.9934026896726719
```

```
Test Accuracy: 0.8809946714031972
```

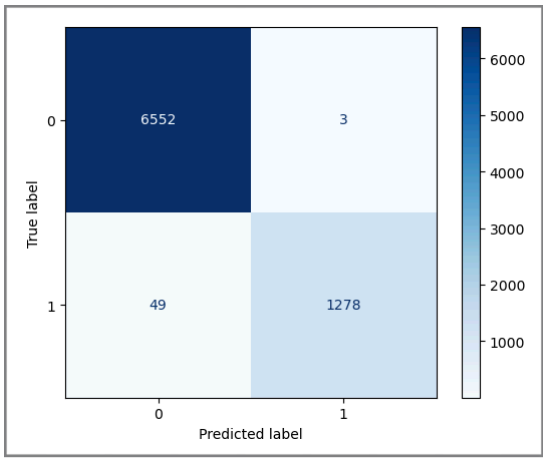
```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      6555
           1       1.00      0.96      0.98      1327

    accuracy                           0.99      7882
   macro avg       1.00      0.98      0.99      7882
weighted avg       0.99      0.99      0.99      7882
```

Classification Report - Train Dataset

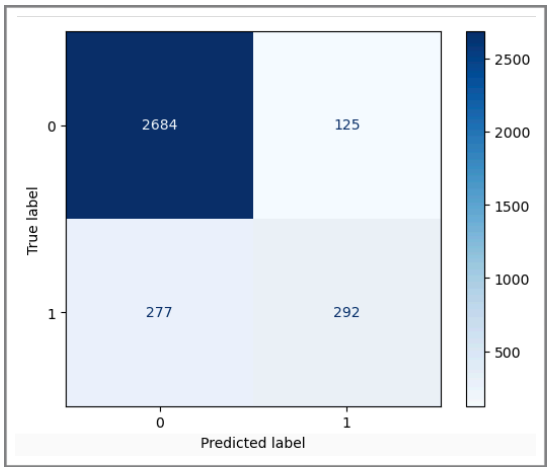
```
Test Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      2809
           1       0.70      0.51      0.59       569

    accuracy                           0.88      3378
   macro avg       0.80      0.73      0.76      3378
weighted avg       0.87      0.88      0.87      3378
```

Classification Report - Test Dataset

Confusion Matrix:

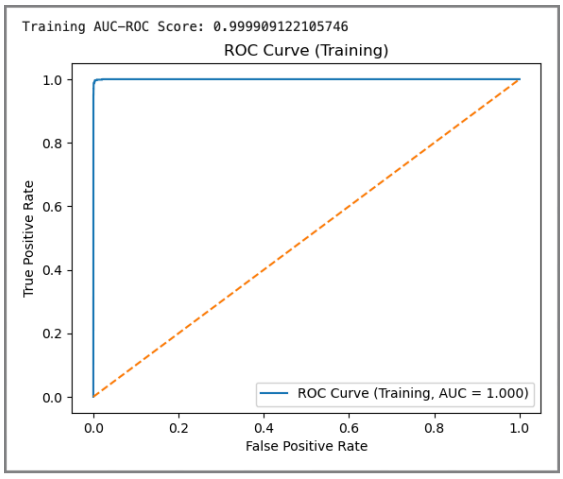


Confusion Matrix - Train Dataset-Bagging Classifier with RF

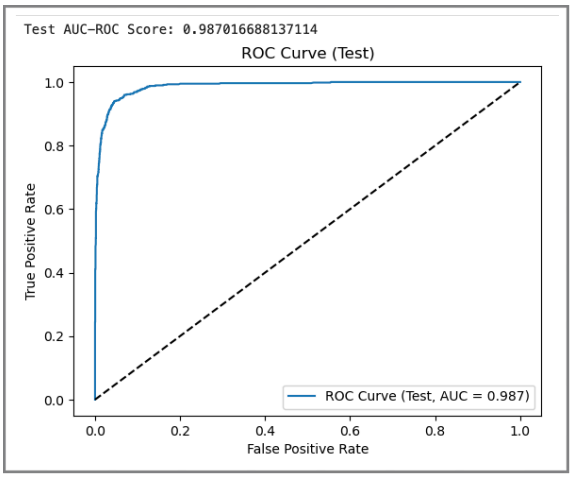


Confusion Matrix - Test Dataset-Bagging Classifier with RF

AUC Score and ROC Curve:



AUC Score and ROC Curve - Test Dataset



AUC Score and ROC Curve - Test Dataset

The Random Forest model with bagging performs well on both the training and testing sets, with high accuracy, precision, recall, and F1-scores. The model is able to capture the underlying patterns in the data and make accurate predictions. The use of bagging has helped to improve the model's performance and reduce overfitting. The original dataset without any preprocessing or feature engineering has not negatively impacted the model's performance.

iv. Bagging with Random Forest on SMOTE(balanced) dataset

Accuracy

```
Training Accuracy: 0.9977116704805492
```

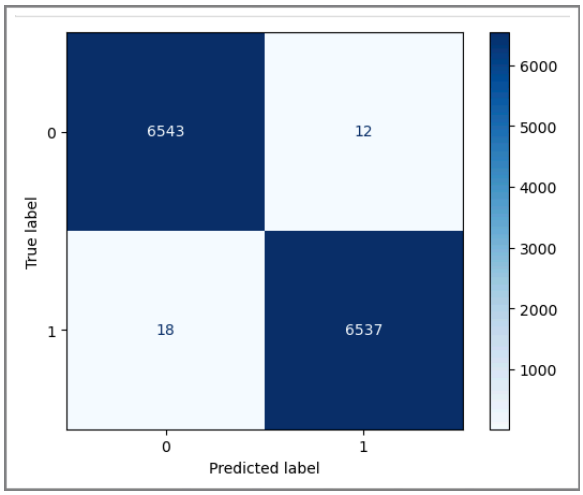
```
Test Accuracy: 0.9550029603315572
```

- **Classification Report**

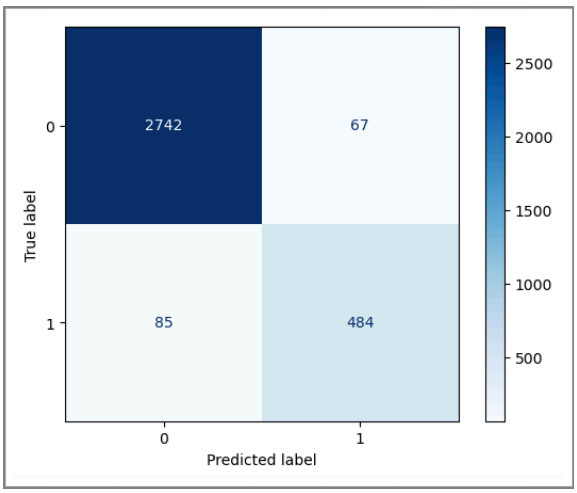
```
Training Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      6555

    accuracy                           1.00     13110
   macro avg       1.00      1.00      1.00     13110
weighted avg       1.00      1.00      1.00     13110
```

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

           0       0.97      0.98      0.97      2809
           1       0.88      0.85      0.86       569

    accuracy                           0.96      3378
   macro avg       0.92      0.91      0.92      3378
weighted avg       0.95      0.96      0.95      3378
```

- **Confusion Matrix**

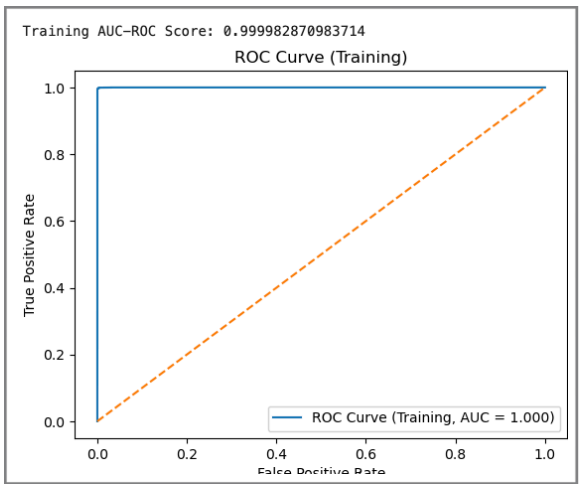


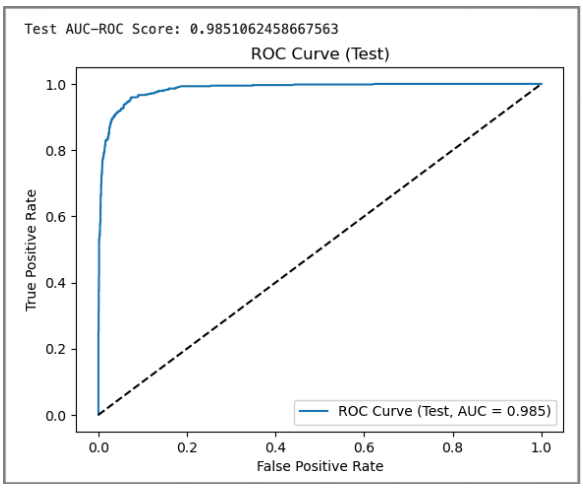
Confusion Matrix - Train Set



Confusion Matrix - Test Set

- **AUC Score and ROC Curve**





- **Cross Validation**

```
Training Cross-Validation Scores: [0.90694127 0.98321892 0.9870328  0.97787948 0.98283753]
Testing Cross-Validation Scores: [0.90828402 0.91863905 0.94674556 0.90814815 0.92296296]
```

The Random Forest model with bagging on the SMOTE dataset performs extremely well on both the training and testing sets, with high accuracy, precision, recall, and F1-scores. The use of bagging and SMOTE has helped to

improve the model's performance and reduce overfitting. The model is able to capture the underlying patterns in the data and make accurate predictions.

Comparison: The Random Forest model with bagging on the SMOTE dataset outperforms the Random Forest model with bagging on the original dataset, achieving a higher accuracy and F1-score on both the training and testing sets. The use of SMOTE has improved the model's performance on the minority class, with a higher precision and recall for class 1. The model with bagging on the SMOTE dataset has a higher accuracy of 0.96 on the testing set, compared to 0.88 for the model with bagging on the original dataset, and a higher F1-score of 0.97 for class 0 and 0.86 for class 1, compared to 0.93 for class 0 and 0.59 for class 1. Overall, the Random Forest model with bagging on the SMOTE dataset is a better performer than the Random Forest model with bagging on the original dataset.

Final Inferences from Random Forest:

In this comparison between the Random Forest model and the Random Forest model with SMOTE, the performance differences are primarily in how each model handles class 0 (the majority class) and class 1 (the minority class).

Random Forest with default parameter :

- **Training Set:** The model performs perfectly on the training data with all metrics (accuracy, precision, recall, F1-scores, and AUC score) equal to 1, which indicates that it might be overfitting.

Testing Set:

- **Class 0 (Majority Class):** The Random Forest model has a high accuracy of 96%, and it performs exceptionally well in identifying class 0. The precision is 0.97, recall is 0.99, and the F1-score is 0.98. This indicates that the model is highly accurate in predicting class 0, with very few false negatives.
- **Class 1 (Minority Class):** The performance for class 1 is lower, with a precision of 0.95, recall of 0.83, and an F1-score of 0.89. This suggests that

while the model identifies most instances of class 0 correctly, it struggles more with class 1, resulting in a higher rate of false negatives.

Random Forest with SMOTE:

- **Training Set**: Like the original model, the Random Forest with SMOTE performs perfectly on the training data, which again suggests potential overfitting.

Testing Set:

- **Class 0 (Majority Class):** The model's performance for class 0 remains strong, with slightly improved results. The precision is 0.98, recall is 0.99, and the F1-score is still 0.98. The slight increase in precision indicates that there are fewer false positives compared to the original model.
- **Class 1 (Minority Class):** The major improvement is seen here. The precision for class 1 increases to 0.94, recall improves to 0.89, and the F1-score rises to 0.91. This indicates a better balance in how the model predicts class 1 without sacrificing much performance in class 0.

Conclusion:

- The **Random Forest with SMOTE** is a better model overall, especially in handling the imbalance between class 0 and class 1. While both models perform well for class 0 (with nearly identical precision, recall, and F1-scores), the Random Forest with SMOTE provides better prediction for class 1 (minority class), reducing false negatives and improving overall balance.
- The AUC score remains constant at 0.99 for both models, reflecting excellent overall discriminatory power, but the SMOTE-applied model achieves better generalization across both classes, particularly in real-world testing data scenarios where class imbalances often occur. Therefore, **Random Forest with SMOTE** is preferable when the minority class predictions are crucial.

Overall Model Comparison

Models	Training Dataset								Testing Dataset							
	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	0.89	0.9	0.97	0.93	0.77	0.47	0.59	0.88	0.89	0.91	0.97	0.94	0.79	0.5	0.61	0.87
Logistic Regression - GridSearchCV	0.89	0.9	0.97	0.93	0.77	0.47	0.58	0.88	0.89	0.9	0.97	0.94	0.77	0.48	0.59	0.87
Logistic regression - SMOTE	0.81	0.82	0.79	0.81	0.8	0.83	0.81	0.88	0.79	0.95	0.78	0.86	0.43	0.82	0.56	0.87
Linear Discriminant Analysis Model	0.88	0.89	0.97	0.93	0.77	0.42	0.54	0.87	0.89	0.9	0.98	0.93	0.79	0.44	0.56	0.87
LDA model - GridSearchCV	0.88	0.89	0.97	0.93	0.77	0.42	0.54	0.87	0.89	0.9	0.98	0.93	0.79	0.44	0.56	0.87
LDA model - SMOTE	0.81	0.83	0.77	0.8	0.79	0.84	0.81	0.88	0.77	0.96	0.76	0.85	0.41	0.83	0.55	0.87
K-Nearest Neighbors (KNN) Model	0.93	0.94	0.98	0.96	0.86	0.68	0.76	0.83	0.88	0.91	0.96	0.93	0.7	0.51	0.59	0.73
KNN with n_neighbors = 3	0.95	0.97	0.98	0.97	0.89	0.83	0.86	0.98	0.89	0.92	0.94	0.93	0.68	0.61	0.64	0.88
KNN - GridSearchCV	1	1	1	1	1	1	1	1	0.94	0.96	0.98	0.97	0.88	0.78	0.82	0.97
KNN - SMOTE	0.94	1	0.88	0.93	0.89	1	0.94	0.99	0.82	0.97	0.81	0.88	0.48	0.86	0.62	0.91
Gaussian Naive Bayes	0.86	0.92	0.91	0.92	0.58	0.59	0.59	0.82	0.85	0.92	0.9	0.91	0.55	0.6	0.58	0.81
Gaussian Naive Bayes - SMOTE	0.74	0.77	0.68	0.72	0.71	0.8	0.76	0.83	0.69	0.94	0.67	0.78	0.32	0.78	0.46	0.81
Random Forest	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0.96**	**0.97**	**0.99**	**0.98**	**0.95**	**0.83**	**0.89**	**0.99**
Random Forest - SMOTE	1	1	1	1	1	1	1	1	0.97	0.98	0.99	0.98	0.94	0.89	0.91	0.99
Bagging - Random Forest	0.99	0.99	1	1	1	0.96	0.98	0.99	0.88	0.91	0.96	0.93	0.7	0.51	0.59	0.99
Bagging - Random Forest - SMOTE	1	1	1	1	1	1	1	0.99	0.96	0.97	0.98	0.97	0.88	0.85	0.86	0.99
Gradient Boost	0.92	0.93	0.98	0.95	0.85	0.64	0.73	0.95	0.91	0.92	0.97	0.95	0.81	0.58	0.68	0.94
Gradient Boost - SMOTE	0.93	0.92	0.94	0.93	0.94	0.92	0.93	0.98	0.9	0.94	0.94	0.94	0.7	0.73	0.71	0.93
XGBoost	0.98	0.98	1	0.99	0.98	0.91	0.94	0.99	0.95	0.96	0.98	0.97	0.9	0.77	0.83	0.98
XGBoost - SMOTE	0.98	0.97	0.99	0.98	0.99	0.97	0.98	0.99	0.94	0.96	0.97	0.96	0.85	0.78	0.82	0.97
Ada-Boost	0.89	0.9	0.97	0.94	0.78	0.46	0.58	0.89	0.89	0.9	0.97	0.94	0.77	0.48	0.59	0.9
Ada-Boost-SMOTE	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.93	0.85	0.95	0.87	0.91	0.54	0.78	0.64	0.89
SVM	0.89	0.9	0.98	0.93	0.79	0.44	0.57	0.52	0.89	0.9	0.98	0.94	0.8	0.46	0.58	0.99
SVM - GridSearchCV	0.89	0.9	0.97	0.93	0.77	0.46	0.57		0.89	0.9	0.97	0.93	0.75	0.48	0.59	
SVM - SMOTE	0.82	0.83	0.79	0.81	0.8	0.84	0.82		0.79	0.96	0.79	0.86	0.44	0.82	0.57	

Implications of the Final Model on Business :

Targeted Customer Retention Strategies: The Random Forest model, leveraging feature importance, allows businesses to focus on the most critical factors influencing customer churn. For example:

- **Tenure:** Since tenure is a top feature, businesses can implement loyalty programs and rewards for long-term customers to encourage continued engagement.
- **Revenue Growth:** Given the importance of year-over-year revenue growth, businesses can offer personalized financial incentives or premium services to high-growth customers.
- **Coupon Usage:** As coupon usage is significant, businesses can provide regular discount coupons, especially for customers who frequently use coupons, to keep them engaged.
- **Account User Count:** Since accounts with multiple users are important, businesses can offer family or group plans with discounts to retain these valuable customers.
- **Monthly Revenue:** Focusing on revenue generated per month, businesses can tailor their pricing and billing strategies to better meet the needs of high-value customers.

- Personalized Offers and Discounts:
  - **Family Floater Plans:** Offer family floater plans or group discounts to accounts with multiple users, as indicated by the importance of **Account_user_count**.
  - **E-Wallet Incentives:** Provide regular discount coupons to customers who pay through the company's e-wallet platform, leveraging the importance of **coupon_used_for_payment**.

    - **Vendor Discounts:** Offer discount vouchers from other vendors or on future bills based on minimum bill criteria, aligning with the importance of **rev_per_month**.
- **Performance Insights:**
    - The model gives businesses a clear understanding of their current customer dynamics and highlights areas for improvement. By focusing on the most important features, businesses can optimize their strategies to better retain customers.
- Data-Driven Decision Making:
    - The feature importance from the Random Forest model enables businesses to make informed decisions about where to allocate resources and how to design retention programs that are most likely to succeed.
- By leveraging these insights, businesses can develop targeted and effective strategies to improve customer retention and reduce churn.

5. Model Validation

When validating a classification model, relying solely on accuracy can be misleading. It's important to consider various other metrics such as the F1 score, recall, precision, ROC curve, AUC score, and the confusion matrix. Here's an explanation of these key metrics:

Confusion Matrix:
The confusion matrix can be tricky to understand, even for experienced users. It consists of four terms:

- **True Positive (TP):** The actual class is positive, and the prediction is also positive.
- **False Positive (FP):** The actual class is negative, but the prediction is positive.
- **True Negative (TN):** The actual class is negative, and the prediction is negative.
- **False Negative (FN):** The actual class is positive, but the prediction is negative.

Key Metrics from the Confusion Matrix:

- **Accuracy:** Measures how many predictions were correct out of all predictions.
- **Precision:** Out of all the predicted positives, how many were truly positive.
- **Recall (Sensitivity):** Out of all the actual positive cases, how many were correctly identified.
- **Specificity:** Out of all the actual negative cases, how many were correctly identified.

- **F1-Score:** The harmonic mean of precision and recall, balancing false positives and false negatives. This metric is especially useful for imbalanced datasets.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC and ROC Curve:

- **ROC Curve:** This plots the true positive rate (recall) against the false positive rate (1 - specificity).
- **AUC (Area Under the Curve):** Represents the model's ability to distinguish between classes. A higher AUC indicates better separability, with values closer to 1 indicating a model with high performance.

These metrics provide a more comprehensive understanding of model performance, especially in cases of imbalanced datasets where accuracy alone can be deceptive.

6. Final Interpretation and Recommendation

Insights from Model Building

- **Random Forest with SMOTE** is the **top-performing model**, but XGBoost and Gradient Boost are close contenders.
- **XGBoost, Random Forest, and Gradient Boosting** should be preferred for deployment as they handle class imbalance effectively while maintaining high accuracy.
- **SMOTE** is an effective technique to improve churn detection in models that initially struggle with class imbalance, though it may slightly impact overall accuracy.
- **Gradient Boosting** and **KNN** (with tuning) also perform well, especially for identifying churn cases.
- **Logistic Regression, LDA, and Naive Bayes** struggle with churn prediction, even with SMOTE, making them less ideal for churn-focused tasks.

- Using techniques like GridSearchCV improves model performance, especially for **KNN** and **SVM**, by finding the optimal hyperparameters. However, excessive tuning can lead to **overfitting**, as seen with KNN.
- **Bagging techniques, such as Random Forest**, help reduce variance and provide stable, reliable predictions. They are particularly effective in balancing overall accuracy and identifying non-churn customers (class 0). With the addition of SMOTE, Random Forest's ability to detect churn cases (class 1) improves, making it a solid choice when stability is crucial.
- **Boosting techniques, like Gradient Boosting and XGBoost**: These models perform well in identifying churn cases, delivering high accuracy and recall even without oversampling methods. XGBoost, in particular, is highly effective for imbalanced datasets, consistently delivering superior performance across all metrics.

Insights from Analysis

1. City Tier 1 has the highest retention rate, with a majority of customers not churning.
2. Male customers are more likely to churn than female customers.
3. Regular Plus segment has the highest churn rate, while Super Plus segment has the lowest churn rate.
4. Mobile users have a higher churn rate compared to computer users.
5. Customers who have complained in the last year are more likely to churn.
6. Longer-tenured customers are less likely to churn.
7. Customers with more recent customer care connections are less likely to churn.
8. Most customers rated services and customer care interactions as "3", indicating a neutral sentiment.
9. Transaction via UPI and e-wallet is very low, suggesting a lack of adoption of these payment methods.
10. Customers with marital status "single" contribute max towards churn, indicating a higher risk of churn among single customers.
11. Any complaints raised in last 12 months doesn't show any impact toward churn, suggesting that complaints may not be a key driver of churn.
12. Tenure and cashback are directly proportional to each other, indicating that longer-tenured customers receive more cashback.
13. Computer usage is more in Tier 1 city followed by Tier 3 and Tier 2 city, highlighting differences in device usage across cities.

Recommendation

- Implement a customer retention program targeting high-risk customers.
- Improve customer service and support to reduce churn.
- Provide training and incentives to customer-facing staff to improve customer service and retention.
- Offer personalized promotions and offers to high-value customers.
- Conduct regular customer surveys to gather feedback and identify areas for improvement.
- Develop a loyalty program to reward loyal customers and encourage retention.
- Analyze customer behavior and transactional data to identify early warning signs of churn.
- Offer personalized product recommendations to customers based on their purchase history and preferences.
- Improve the user experience of the company's website and mobile app to reduce friction and improve customer retention.
- Enhance customer engagement through regular communication and feedback mechanisms.
- Increase visibility in Tier-2 and Tier-3 cities to improve customer acquisition by partnering with local businesses, sponsoring local events, developing targeted marketing campaigns, and establishing a local presence through physical offices
- Introduce a referral drive for existing customers to acquire new customers.

Appendix

Importing necessary libraries

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import scipy.stats as stats
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report


from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from scipy import stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV    # Train test Split and Grid Search
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier

import statsmodels.api as SM
from sklearn import metrics

from sklearn.metrics import (
    confusion_matrix,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_curve,
    roc_auc_score
)
```

Loading the Data

```
df=pd.read_excel('/Users/ishashukla/Desktop/Data Science/Capstone Project/CC_EDTH_02_Customer+Churn_02/Customer Churn Data.xlsx',she
```

```
df
```

```
#checking for no. of unique values
df.nunique() ##  check unique entries in the data
```

```
#check for duplicate values
df.duplicated().sum()
```

```
# Fixing the data entry errors

df["Gender"] = df["Gender"].replace("F", "Female")
df["Gender"] = df["Gender"].replace("M", "Male")
df["account_segment"] = df["account_segment"].replace("Regular +", "Regular Plus")
df["account_segment"] = df["account_segment"].replace("Super +", "Super Plus")

df["Login_device"] = df["Login_device"].replace(
    "&&&&", "Other"
)  # will decide if this is to be treated as a missing value or an 'Unknown' category after EDA
```

```
# Replacing the invalid data with np.nan

df["Tenure"].replace("#", np.nan, inplace=True)
df["Account_user_count"].replace("@", np.nan, inplace=True)
df["rev_per_month"].replace("+", np.nan, inplace=True)
df["rev_growth_yoy"].replace("$", np.nan, inplace=True)
df["coupon_used_for_payment"].replace(["#", "$", "*"], np.nan, inplace=True)
df["Day_Since_CC_connect"].replace("$", np.nan, inplace=True)
df["cashback"].replace("$", np.nan, inplace=True)
```

Univariate Analysis

```
def histogram_boxplot(data, feature, figsize=(15, 10), kde=False, bins=None):
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (15,10))
    kde: whether to show the density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows=2,  # Number of rows of the subplot grid= 2
        sharex=True,  # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    )  # creating the 2 subplots
    sns.boxplot(
        data=data, x=feature, ax=ax_box2, showmeans=True, color="khaki"
    )  # boxplot will be created and a triangle will indicate the mean value of the column
    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins
    ) if bins else sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color="purple", linestyle="--"
    )  # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color="black", linestyle="-"
    )  # Add median to the histogram
```

```
def perc_on_bar(plot, feature):
    """
    plot
    feature : categorical feature
    the function won't work if a column is passed in hue parameter

    """
    total = len(feature)  # length of the column
    for p in ax.patches:
        percentage = "{:.1f}%".format(
            100 * p.get_height() / total
        )  # percentage of each class of the category
        x = p.get_x() + p.get_width() / 2 - 0.06  # width of the plot
        y = p.get_y() + p.get_height()  # height of the plot
        ax.annotate(
            percentage,
            (x, y),
            # ha="center",
            # va="center",
            size=12,
            # xytext=(0, 3),
            # textcoords="offset points",
        )  # annotate the percantage
    plt.show()  # show the plot
```

```
import pandas as pd

# Initialize the outliers_count dictionary
outliers_count = {}

# Iterate over each column in the DataFrame
for column in df.columns:
    if pd.api.types.is_numeric_dtype(df[column]):
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
        outliers_count[column] = {
            'No. of outliers': len(outliers),
            'Percentage of outliers': len(outliers) / len(df) * 100
        }

    elif pd.api.types.is_object_dtype(df[column]):
        value_counts = df[column].value_counts()
        rare_threshold = 0.05 * len(df)  # Define what you consider rare (e.g., less than 5% frequency)
        outliers = df[df[column].isin(value_counts[value_counts <= rare_threshold].index)]
        outliers_count[column] = {
            'No. of outliers': len(outliers),
            'Percentage of outliers': len(outliers) / len(df) * 100
        }

# Print the number of outliers and their percentages in each column
outliers_df = pd.DataFrame([
    {'Column': column, 'No. of outliers': details['No. of outliers'], 'Percentage of outliers': details['Percentage of outliers']}
    for column, details in outliers_count.items()
])

print(outliers_df)
```

```python
# Initialize the skewness and kurtosis dictionaries
skewness_info = {}
kurtosis_info = {}

# Iterate over each column in the DataFrame
for column in df.columns:
    if pd.api.types.is_numeric_dtype(df[column]):
        # Calculate skewness and kurtosis
        skewness = df[column].skew()
        kurtosis = df[column].kurtosis()
        skewness_info[column] = round(skewness, 2)
        kurtosis_info[column] = round(kurtosis, 2)

        # Visualize the distribution
        plt.figure(figsize=(14, 6))

        # Histogram
        plt.subplot(1, 2, 1)
        sns.histplot(df[column], kde=True)
        plt.title(f'Histogram of {column}\nSkewness: {skewness:.2f}, Kurtosis: {kurtosis:.2f}')

        # QQ plot
        plt.subplot(1, 2, 2)
        stats.probplot(df[column], dist="norm", plot=plt)
        plt.title(f'QQ plot of {column}')

        plt.tight_layout()
        plt.show()

# Print the skewness and kurtosis information
skewness_kurtosis_df = pd.DataFrame([
    {'Column': column, 'Skewness': skewness, 'Kurtosis': kurtosis}
    for column, (skewness, kurtosis) in zip(skewness_info.keys(), zip(skewness_info.values(), kurtosis_info.values()))
])

print("Skewness and Kurtosis of each numerical column:")
print(skewness_kurtosis_df)
```

```python
# Defining a method to plot stacked bar plot


def stacked_barplot(data, predictor, target):
    """
    Print the category counts and plot a stacked bar chart
    data : dataframe
    predictor : independent variable
    target : target variable

    """
    count = data[predictor].nunique()
    sorter = data[target].value_counts().index[-1]
    tab1 = pd.crosstab(data[predictor], data[target], margins=True).sort_values(
        by=sorter, ascending=False
    )
    print(tab1)
    tab = pd.crosstab(data[predictor], data[target], normalize="index").sort_values(
        by=sorter, ascending=False
    )
    tab.plot(kind="bar", stacked=True, figsize=(count + 3, 5))
    plt.legend(loc="lower left", frameon=False)
    plt.legend(loc="lower left", bbox_to_anchor=(1, 1))
    plt.xticks(rotation=0)
    plt.show()
```

```python
# Filter numeric columns (excluding 'uint8' and 'bool') and also excluding 'Churn'
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
numeric_cols = numeric_cols.drop('Churn')  # Exclude the 'Churn' column

# Construct box plot for numeric columns
plt.figure(figsize=(12, 10))
df[numeric_cols].boxplot(vert=0)
plt.title('Box Plot for Numeric Columns (Excluding Churn)', fontsize=16)
plt.xlabel('Value')
plt.show()
```

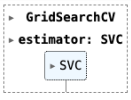
Bagging Classifier with Random Forest

```python
# Create a random forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Create a bagging classifier with 10 random forests
bag = BaggingClassifier(estimator=rf, n_estimators=10, random_state=42)
```

```python
# Train the bagging classifier on the training data
bag.fit(X_train, y_train)
```

Hypertuning SVM - grid search to tune the hyperparameters

```python
param_grid = {'kernel': ['linear', 'rbf'], 'C': np.logspace(-2, 1, 5), 'gamma': [0.2, 0.5]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```



```
# Creating a list of columns to impute missing values

cols_to_impute = [
    "Tenure",
    "City_Tier",
    "CC_Contacted_LY",
    "Payment",
    "Gender",
    "Service_Score",
    "Account_user_count",
    "account_segment",
    "CC_Agent_Score",
    "Marital_Status",
    "rev_per_month",
    "Complain_ly",
    "rev_growth_yoy",
    "coupon_used_for_payment",
    "Day_Since_CC_connect",
    "cashback",
    "Login_device",
]
```

```
# Defining the value for 'n' neighbors

imputer = KNNImputer(n_neighbors=5)
```

```
# Encoding categorical variables into numerical values to perform KNN imputation

payment = {
    "Debit Card": 0,
    "UPI": 1,
    "Credit Card": 2,
    "Cash on Delivery": 3,
    "E wallet": 4,
}
df["Payment"] = df["Payment"].map(payment)

gender = {"Female": 0, "Male": 1}
df["Gender"] = data["Gender"].map(gender)

account_segment = {
    "Super": 0,
    "Regular Plus": 1,
    "Regular": 2,
    "HNI": 3,
    "Super Plus": 4,
}
df["account_segment"] = df["account_segment"].map(account_segment)

marital_status = {"Single": 0, "Divorced": 1, "Married": 2}
df["Marital_Status"] = df["Marital_Status"].map(marital_status)

login_device = {"Mobile": 0, "Computer": 1, "Other": 2}
df["Login_device"] = df["Login_device"].map(login_device)
```

```
print("Encoded Dataset:")
df.head()
```

Splitting data into train and test data set

```
#Replace the missing values in the data using KNN Imputer
KNNimputerModel = KNNImputer(n_neighbors = 5)  ## Complete the code to select 5 neighbors for KNN Imputer

X_train = pd.DataFrame(KNNimputerModel.fit_transform(X_train), columns = X_train.columns)
X_test = pd.DataFrame(KNNimputerModel.fit_transform(X_test), columns = X_test.columns)  ## Complete the code to replace missing valu
```

Scaling the Data

```
#Scaling of features is done to bring all the features to the same scale.
sc = StandardScaler()

X_train_scaled = pd.DataFrame(sc.fit_transform(X_train), columns=X_train.columns)
X_test_scaled = pd.DataFrame(sc.fit_transform(X_test), columns=X_test.columns)  ## Complete the code to scale X_test to the same sca
```

```
X_train_scaled.head()
```

Random Forest

```
# Train a Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
# Fit the model to the training data
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Random forest model over SMOTE

```
# Train a Random Forest classifier on the balanced data
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_res, y_train_res)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```