Business Report

Customer Churn Prediction

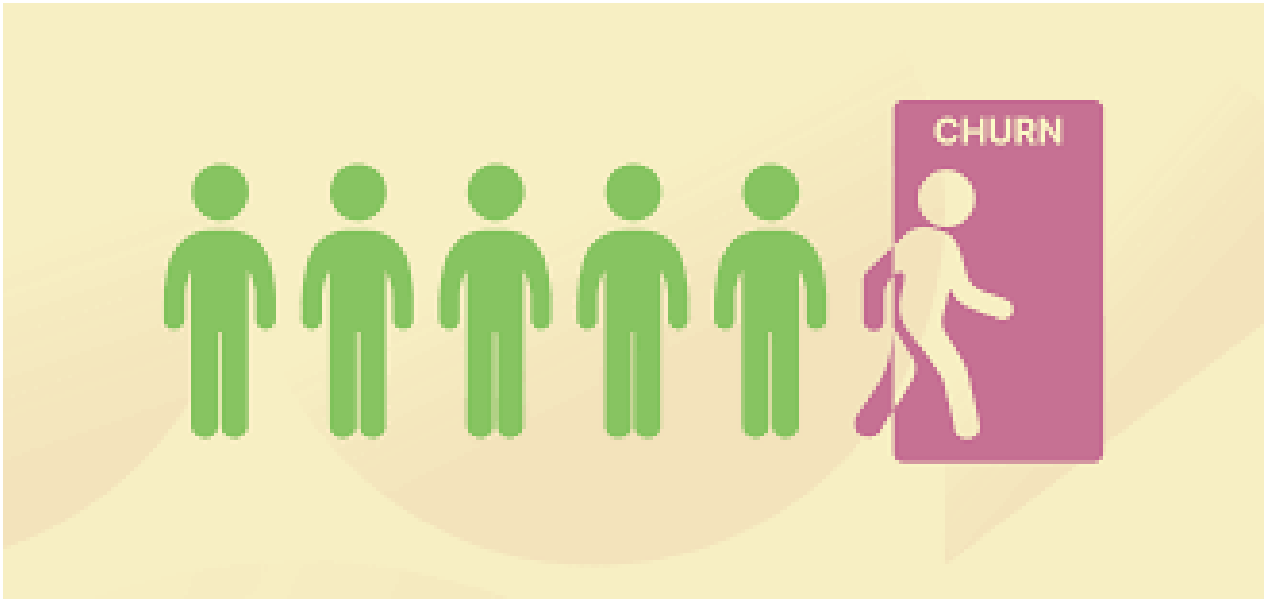

Table of Contents

Table of Figure

Problem Statement

Business Context

In the highly competitive landscape of the **E-Commerce** industry, customer retention has become a critical challenge. With multiple service providers offering similar products and services, customers have the flexibility to switch providers easily. A key concern for the company is **account churn**, which is particularly impactful because a single account may represent multiple users. Losing an account not only reduces revenue but also affects customer lifetime value and brand loyalty. Given the increasing cost of customer acquisition, it is more efficient for the company to **identify potential churners and implement targeted retention strategies** rather than focusing solely on acquiring new customers.

Objective

The goal of this project is to develop a **churn prediction model** that can identify accounts at high risk of cancellation. By leveraging historical customer data, behavioral patterns, and engagement metrics, the model will provide insights into the key factors contributing to churn. Based on the model's predictions, **a strategic retention campaign** will be designed to offer personalized incentives to at-risk customers. The recommendations will focus on **cost-effective retention strategies**, ensuring that the proposed offers maintain a balance between customer

satisfaction and the company’s profitability, aligning with the revenue assurance team's expectations.

Data Description

The dataset consists of account-level information, capturing both customer behavior and financial metrics relevant to churn prediction. Each row represents a unique account (AccountID), with multiple customers potentially linked to it. The target variable, **Churn**, indicates whether an account has churned. The dataset contains 11,260 rows and 19 columns where there are 2 integers, 5 floats and 12 objects.

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_LY	How many times all the customers of the account has contacted customer care in last 12 months

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_ly	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

Overview of the dataset

This is a sample of dataset we have loaded:

	AccountID	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	Day_Since_CC_connect	cashback	Login_device
0	20000	1	4	3.0	6.0	Debit Card	Female	3.0	3	Super	2.0	Single	9	1.0	11	1	5	159.93	Mobile
1	20001	1	0	1.0	8.0	UPI	Male	3.0	4	Regular Plus	3.0	Single	7	1.0	15	0	0	120.9	Mobile
2	20002	1	0	1.0	30.0	Debit Card	Male	2.0	4	Regular Plus	3.0	Single	6	1.0	14	0	3	NaN	Mobile
3	20003	1	0	3.0	15.0	Debit Card	Male	2.0	4	Super	5.0	Single	8	0.0	23	0	3	134.07	Mobile
4	20004	1	0	1.0	12.0	Credit Card	Male	2.0	3	Regular Plus	5.0	Single	3	0.0	11	1	3	129.6	Mobile
5	20005	1	0	1.0	22.0	Debit Card	Female	3.0	NaN	Regular Plus	5.0	Single	2	1.0	22	4	7	139.19	Computer
6	20006	1	2	3.0	11.0	Cash on Delivery	Male	2.0	3	Super	2.0	Divorced	4	0.0	14	0	0	120.86	Mobile
7	20007	1	0	1.0	6.0	Credit Card	Male	3.0	3	Regular Plus	2.0	Divorced	3	1.0	16	2	0	122.93	Mobile
8	20008	1	13	3.0	9.0	E wallet	Male	2.0	4	Regular Plus	3.0	Divorced	2	1.0	14	0	2	126.83	Mobile
9	20009	1	0	1.0	31.0	Debit Card	Male	2.0	5	Regular Plus	3.0	Single	2	0.0	12	1	1	122.93	Mobile

Figure 1 Data Preview

Exploratory Data Analysis

Univariate Analysis

Account ID

We are dropping account ID since it is a unique identifier for accounts and does not serve as a feature in our model.

Churn

Churn rate is our target variable and hence we require it to be an object type. It would be a nominal categorical variable. There are no missing values. We can see that 83.2% of the

customers have been retained and 16.8% have churned. As per our analysis this is a huge churn rate and this can be because of a highly competitive market. We need to focus to retain more customers since a loss of an account would mean a loss of a huge number of customers.

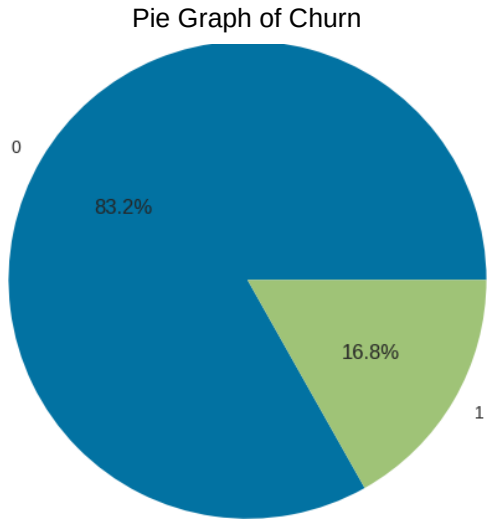

Figure 2 Univariate Analysis of Churn

Tenure

Tenure is an object data type. This is an ordinal categorical variable since there is an order to the variable where a year is less than 3 years of tenure. 14% of accounts have a tenure of 1 year followed by accounts which have less than a year and accounts to 10.9%.

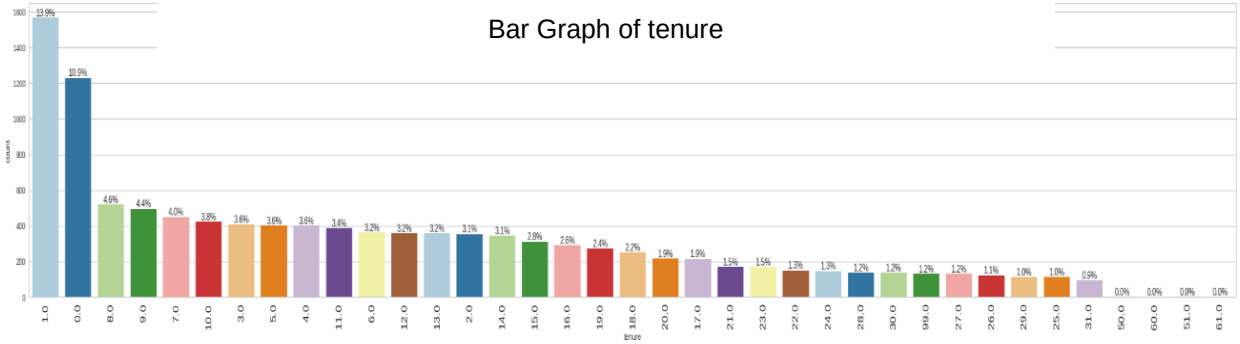

Figure 3 Univariate Analysis of Tenure

City Tier

City tier is an ordinal categorical variable since it has an order or rank. City tier was a float type and we converted it to object since it is a categorical variable. We can see that most of the customers hail from city tier 1 which is 65.5% followed by city tier accounting to 30.2%. Least share of customers hail from city tier 2 accounting to 4.3%.

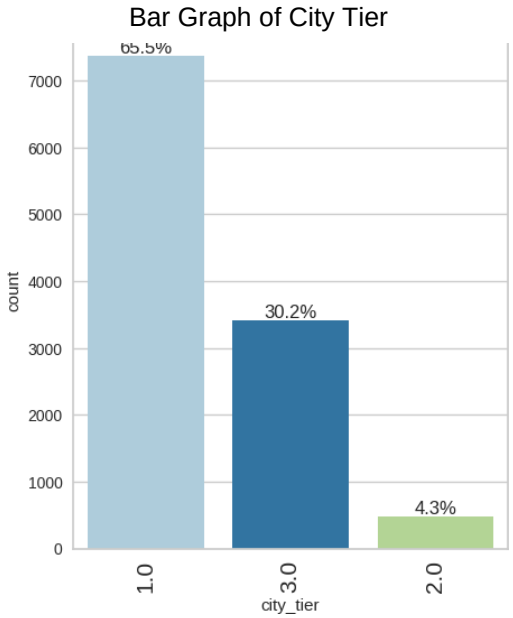

Figure 4 Univariate Analysis of City Tier

CC_Contacted_LY

This is a numeric variable. The graph is left skewed which means most of the customers contacted are between 4 to 23 days. However there are circumstances where the highest number of days were 132 days. On an average the customers contact customer care in an interval of 17 days.

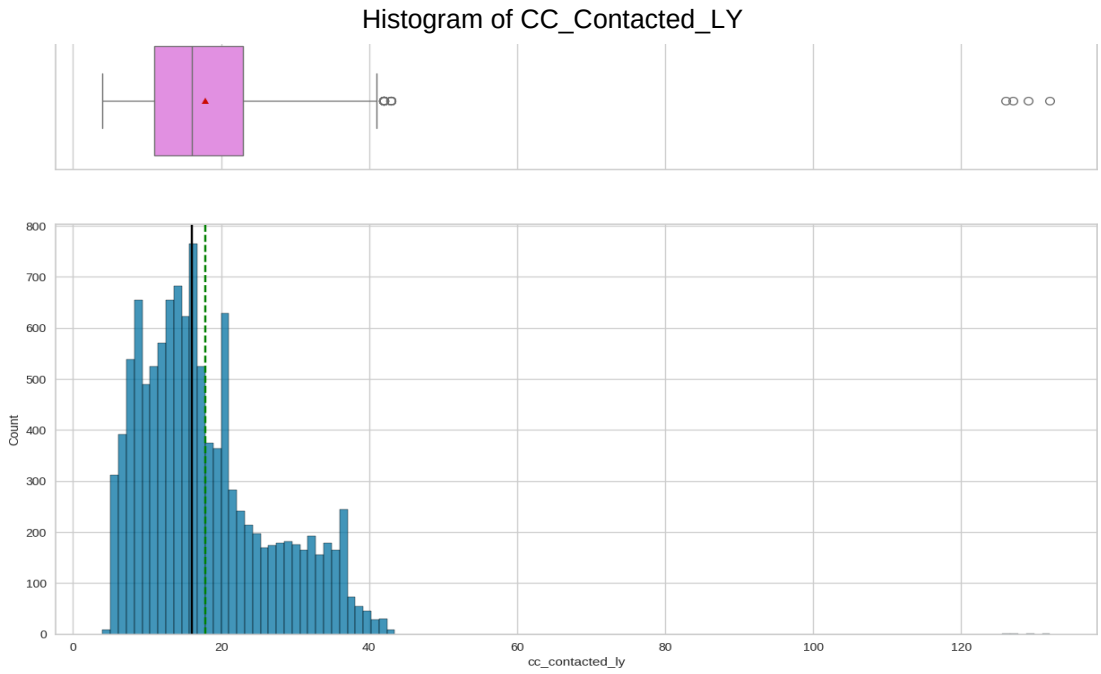

Figure 5 Univariate Analysis of CC Contacted last year

Payment

Payment method is a nominal categorical variable. The payment modes in the dataset are debit cards, credit cards, E wallet, cash on delivery and UPI. Around 42% of the customers pay through Debit cards followed by 31.2% of customers using credit cards. 8% of the customers

prefer UPI payment method which is the least.

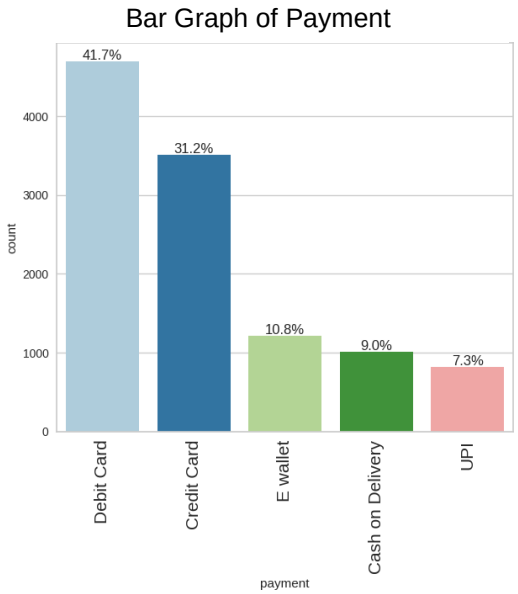

Figure 6 Univariate Analysis of payment

Gender

This is a nominal categorical variable. The data contains 60.5% of males and 39.5% of females.

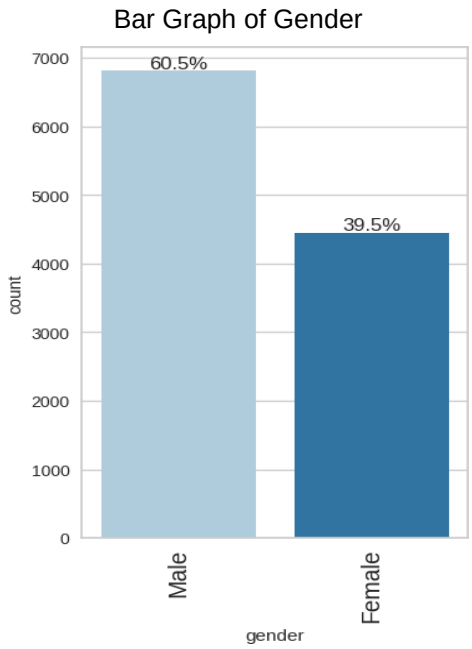

Figure 7 Univariate Analysis of gender

Service_Score

This is an ordinal categorical variable. The score is between 0 to 5. Around 50% of the customers rated the service as 3-star rating followed by 29% of customers given a rating of 2 stars and 21% provided rating of 4. Very least number of customers has rated 1 and 0 which accounts to 0.8% of the population which is a positive sign for the company.

Bar Graph of Service Score

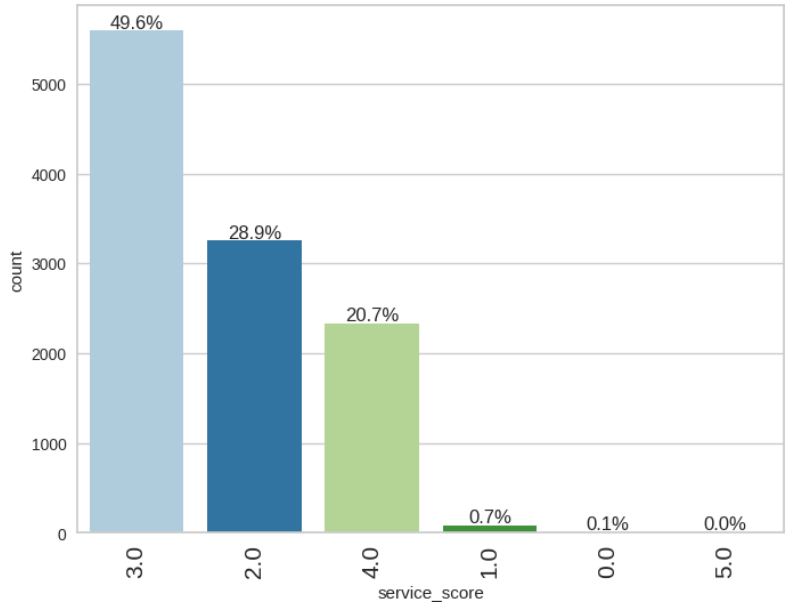

Figure 8 Univariate Analysis of service score

Account_User_Count

This is a numeric variable. Currently it is an object data type and we will convert it to integer. We can see that the highest number of customers in an account is 4. There are outliers in the dataset such as an account having 6 customers and lower outliers lying at 1 customer.

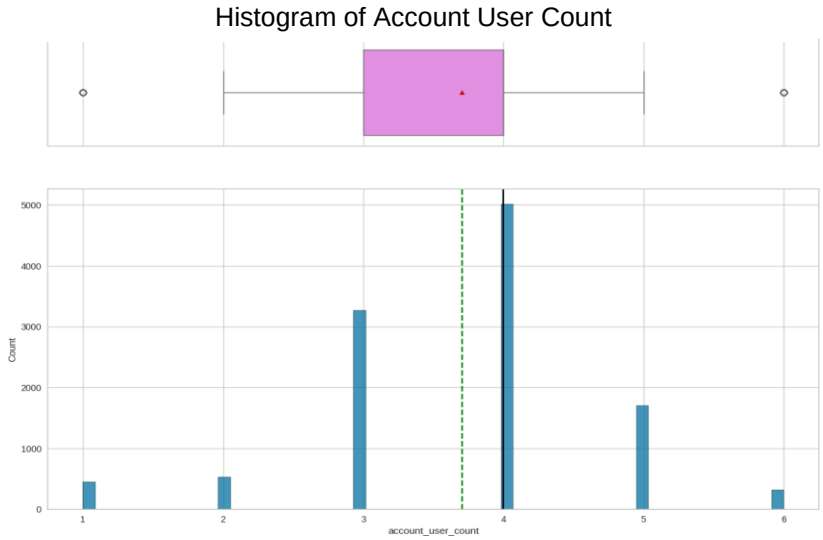

Figure 9 Univariate Analysis of account user count

Account_Segment

This is an ordinal categorical variable. The majority of the customers lie in the regular plus category followed by super which accounts to 37.5% and 36.1% respectively. Least number of customers falls under regular with 4.6% and super plus with 7.3%.

Bar Graph of Segment

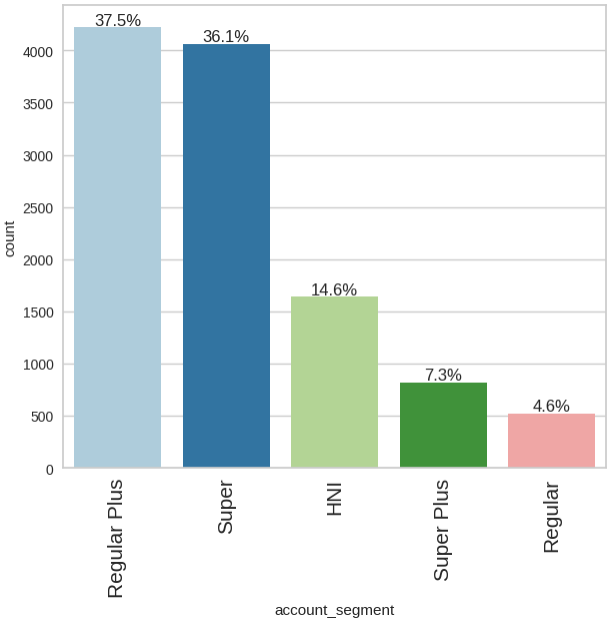

Figure 10 Univariate Analysis of account segment

CC_Agent_Score

This is an ordinal categorical variable. We will convert the data type from float to object. The highest number of scores is 3 which constitutes 30.9% of customer rating followed by 1 star rating with 20.4%. The least count is for 2 star rating which is 10.3%. The company requires to focus on the customer care service provided since the rating is very low or neutral. The company should focus on getting a 5 star rating in order to keep the customer satisfied.

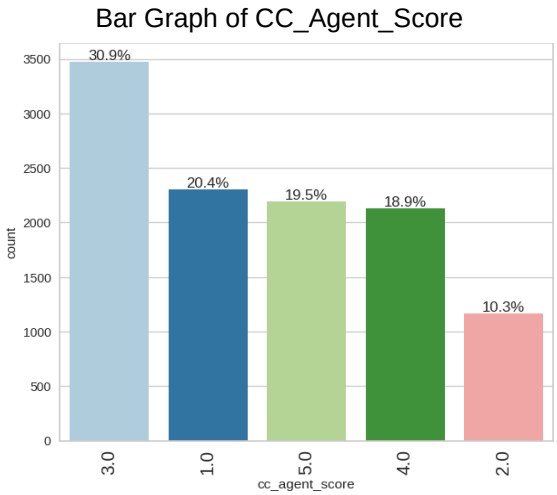

Figure 11 Univariate Analysis of cc agent score

Marital Status

This is a nominal categorical variable. Majority of the customers are married with a share of 54% followed by single customers with a share of 31.3%. The least number of customers are divorced with a share of 14.8%.

Bar Graph of Marital Status

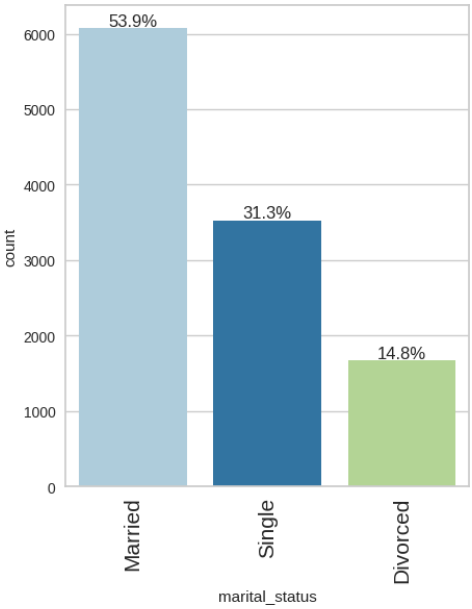

Figure 12 Univariate Analysis of marital status

Rev_Per_Month

This is a numeric variable. We will convert this data type from object to float since it's a continuous variable. We can see the outliers in the graph and this is because of the revenue being generated from different accounts and each account is diverse in nature. On an average the revenue generated is 6 whereas the minimum is 1 and maximum is 140. This column is heavily left skewed where 75% of the account lies between 1 to 7.

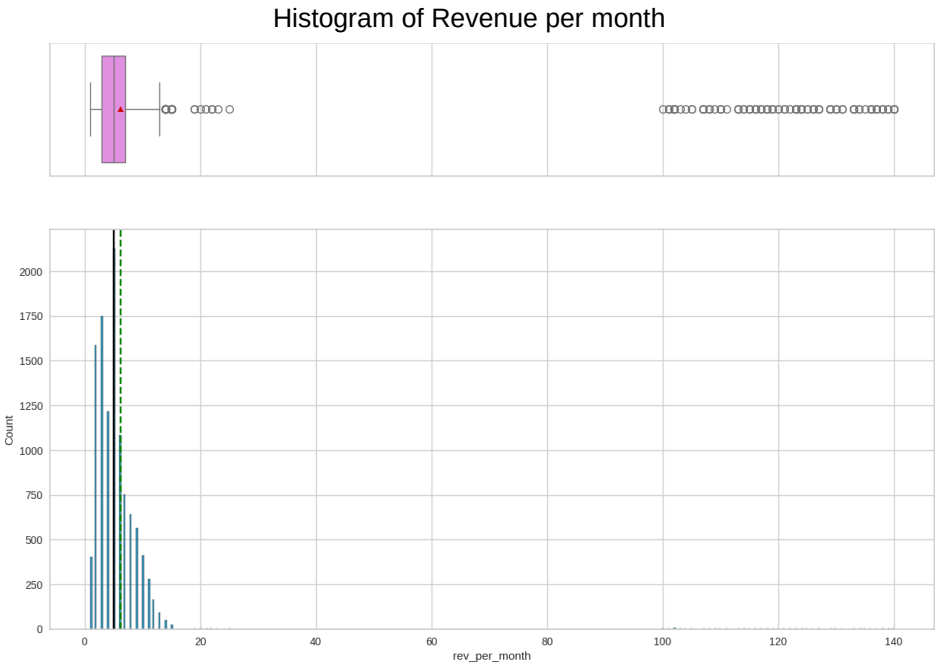

Figure 13 Univariate Analysis of revenue per month

Complain_ly

This is a categorical variable. We will convert this to an object. 72.4% of the accounts have not

raised any complaints in the last 12 months whereas 27.6% of the accounts have raised complaints. The company should focus on reducing the number of complaints in order to provide maximum customer satisfaction.

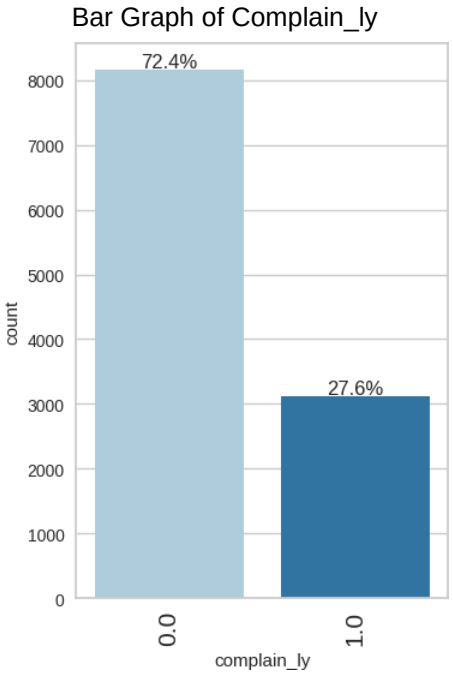

Figure 14 Univariate Analysis of complain last year

Rev_Growth_YoY

This is a numeric variable. We will convert to float data type from object type. The revenue growth on an average is 16%. The maximum growth is 28% and minimum is 4%.

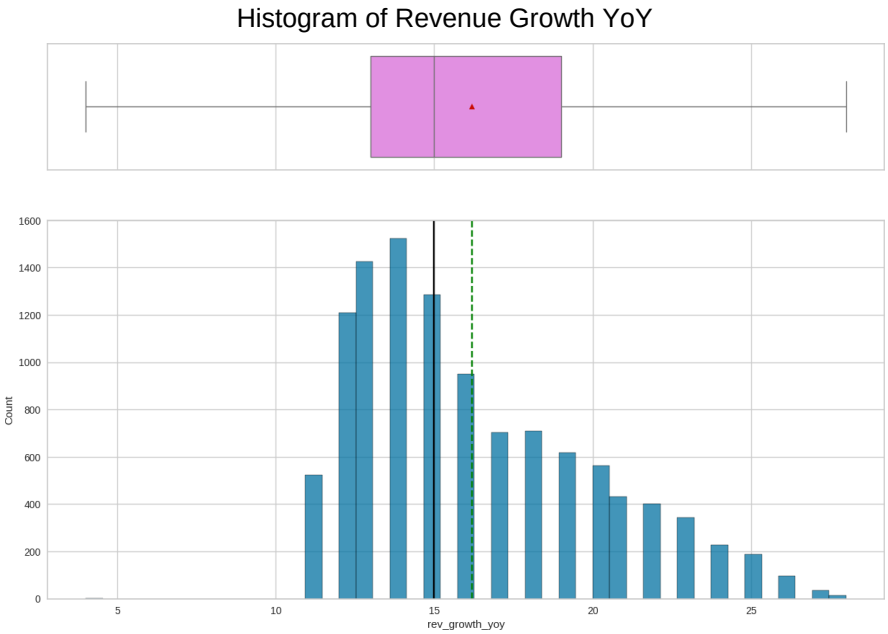

Figure 15 Univariate Analysis of revenue growth year on year

Coupon_Used_For_Payment

This is a numeric variable. From object type we will change to integer. On an average,

customers use 2 coupons for payment. The minimum coupon used is 0 and maximum is 16

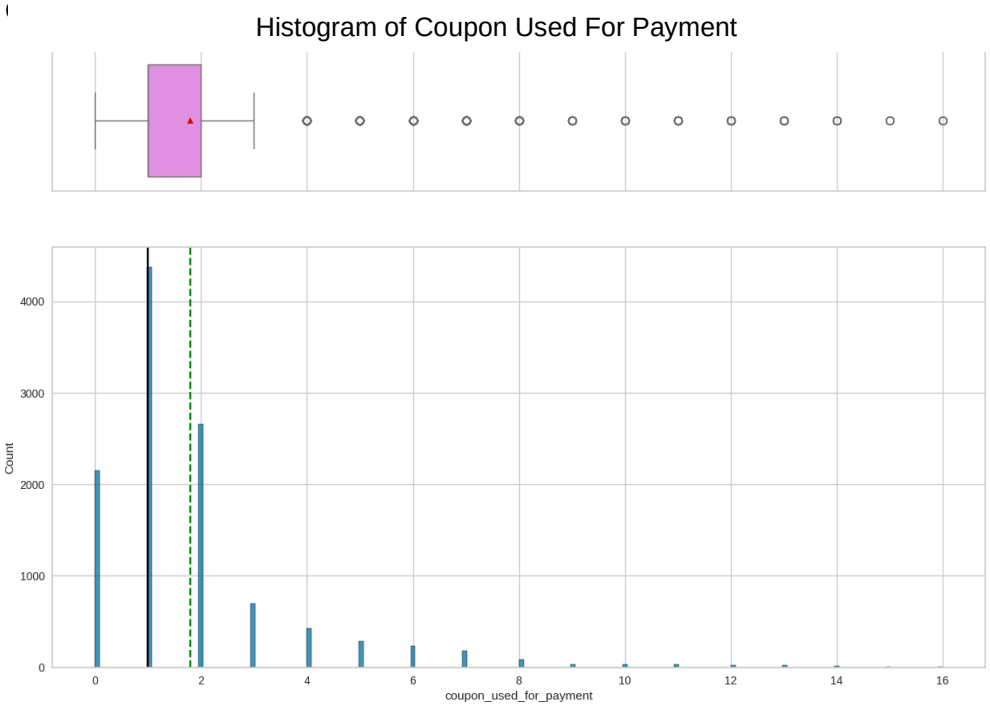

Figure 16 Univariate Analysis of coupon used for payment

Days_Since_CC_Connect

This is a numeric variable. We will convert the data type from object to integer. On an average day since the last customer care connect is 4 days. The minimum is 0 days and the maximum is 47 days.

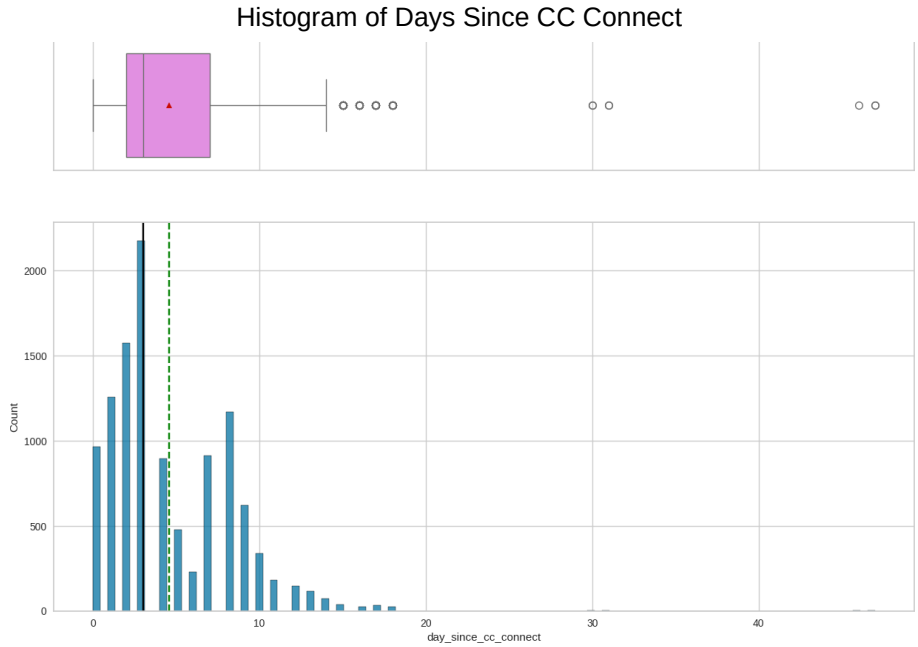

Figure 17 Univariate Analysis of days since cc connect

Cashback

This is a numeric variable and we will treat this as a float. On an average the cashback generated by an account is 194. The minimum is 0 and the maximum is 1997. The standard deviation stands at 148 which states that there is huge variability of data in a column.

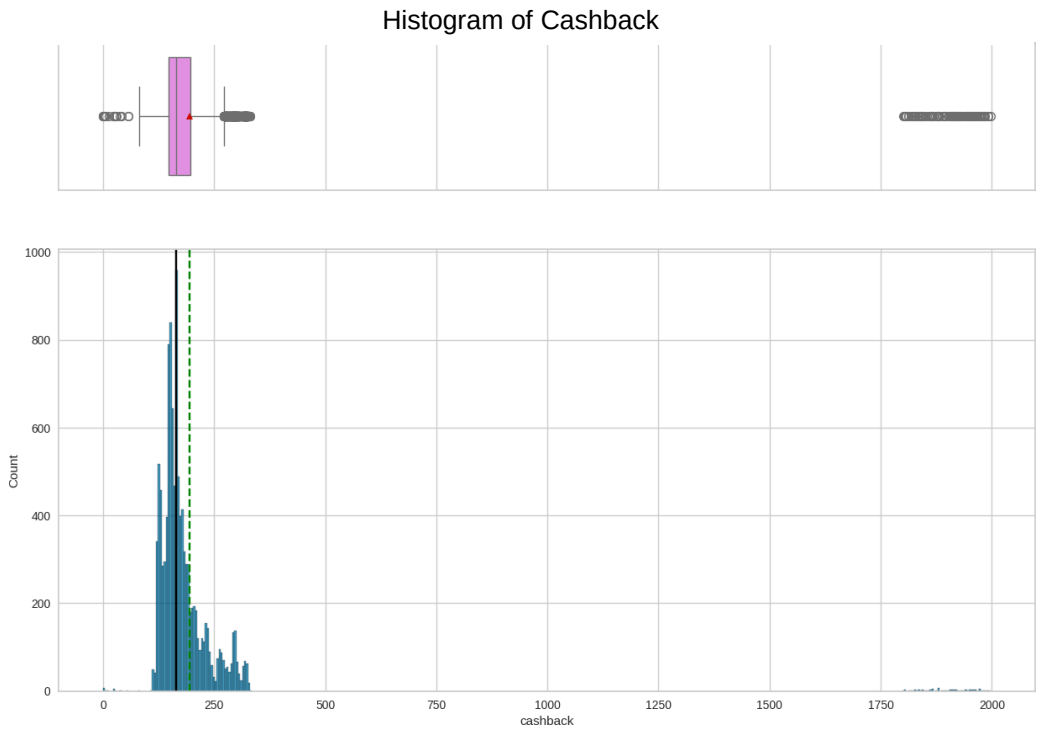

Figure 18 Univariate Analysis of cashback

Login_Device

This is a nominal categorical variable. Around 73% of the customers use mobile to login to the business platform and 27% use computer to login.

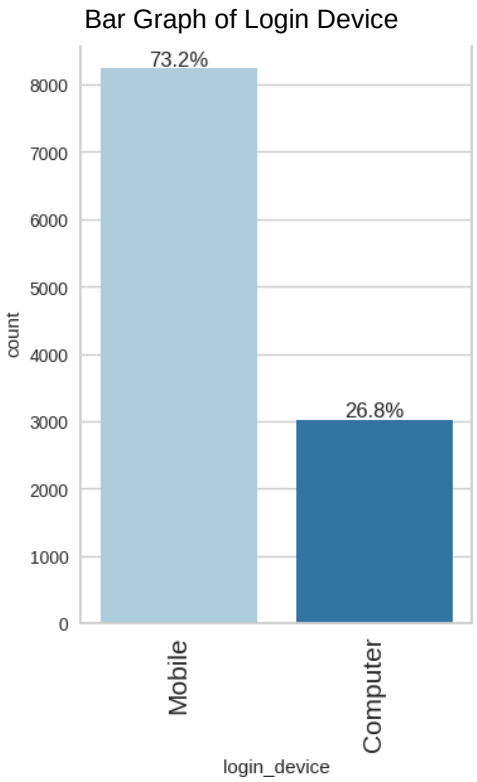

Figure 19 Univariate Analysis of login device

Bivariate Analysis

Heatmap Analysis

With heatmap we tend to check the correlation between the numeric columns.

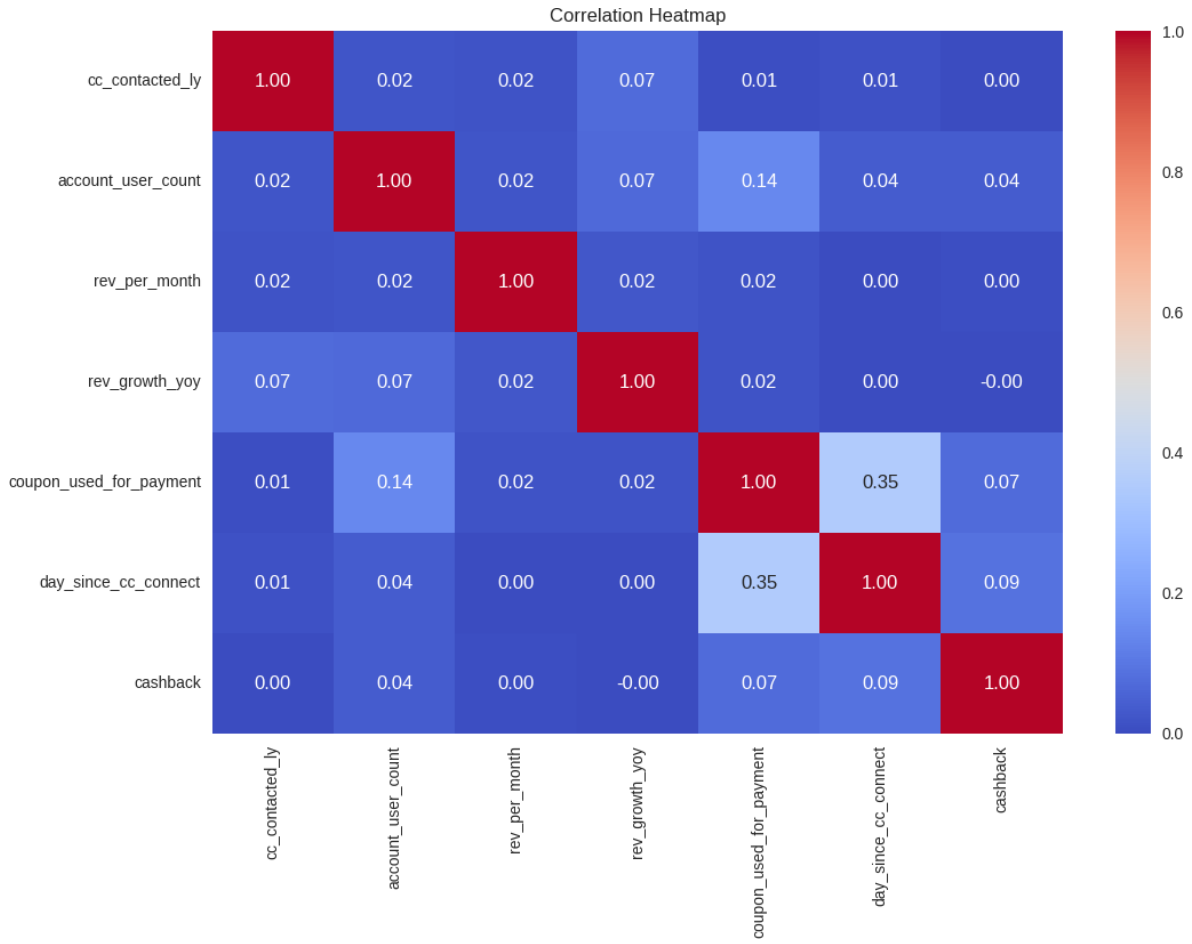

Figure 20 Heat map analysis

- There is no major correlation between variables in general.
- Coupons used for payment and days since customer care connect have a mild positive correlation of 0.35 which means an increase in one variable causes a slight increase in other variable.
- There is also a slight positive correlation between account user count and coupon used for payment which means the more the number of users in account, the more the coupons are used for payments.

Churn Rate by Tenure

We can see that the less the tenure is the greater the chance of churn happens. This is due to the competitive market and the company needs to focus on customer care.

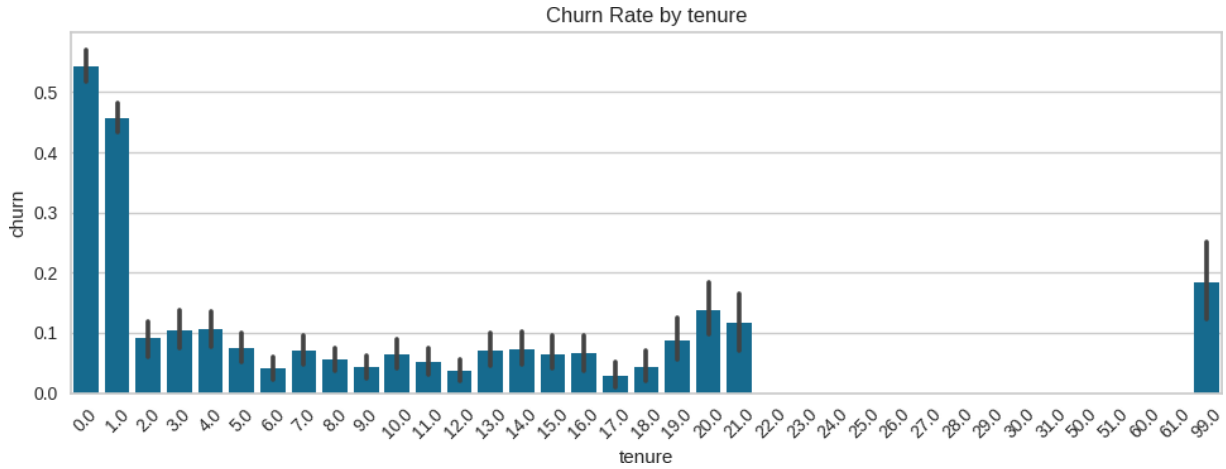

Figure 21 Churn Rate by Tenure

Churn Rate by City Tier

The churn rate is noticeably higher for customers in City Tier 3 and City Tier 2 compared to those in Tier 1. This suggests that customers located in Tier 3 and Tier 2 cities are more likely to churn than customers located in Tier 1 cities.

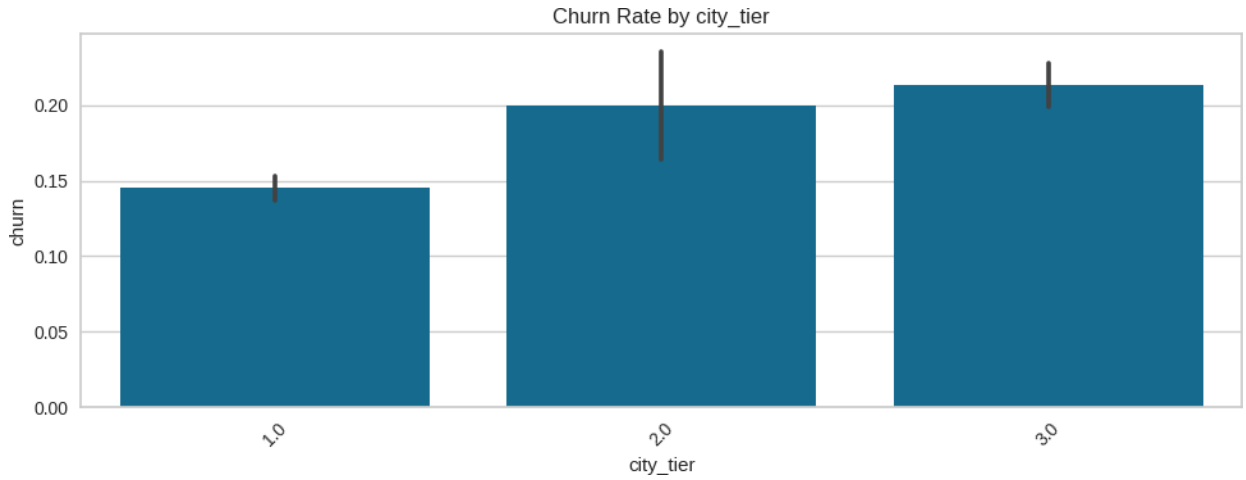

Figure 22 Churn Rate by City Tier

Churn Rate by Payment

Churn rate is higher for cash on delivery and E wallet. The churn is less for credit card and Debit card. This might be due to any offers provided by the card banks and also from a convenience perspective the customers might feel it's easier to transact since data is always saved in mobile or computer devices.

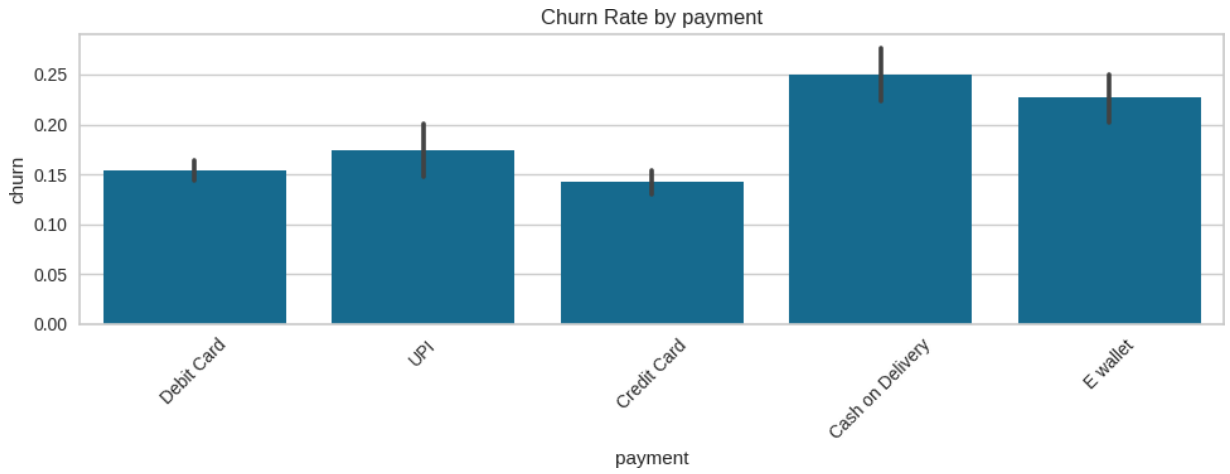

Figure 23 Churn Rate by Payment

Churn Rate by Gender

The churn rate is higher in males in comparison to females. This could be because of higher number of male populations in the dataset.

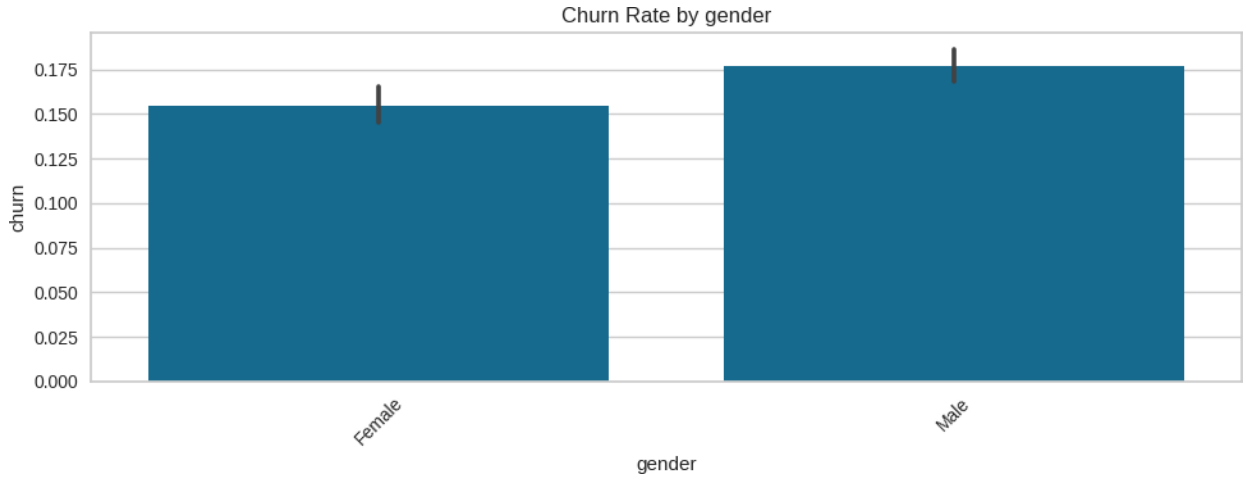

Figure 24 Churn Rate by Gender

Churn Rate by Service Score

The chances of churn rate seems to be high in 2, 3 and 4 star ratings. It seems like the customers are churning no matter the service provided.

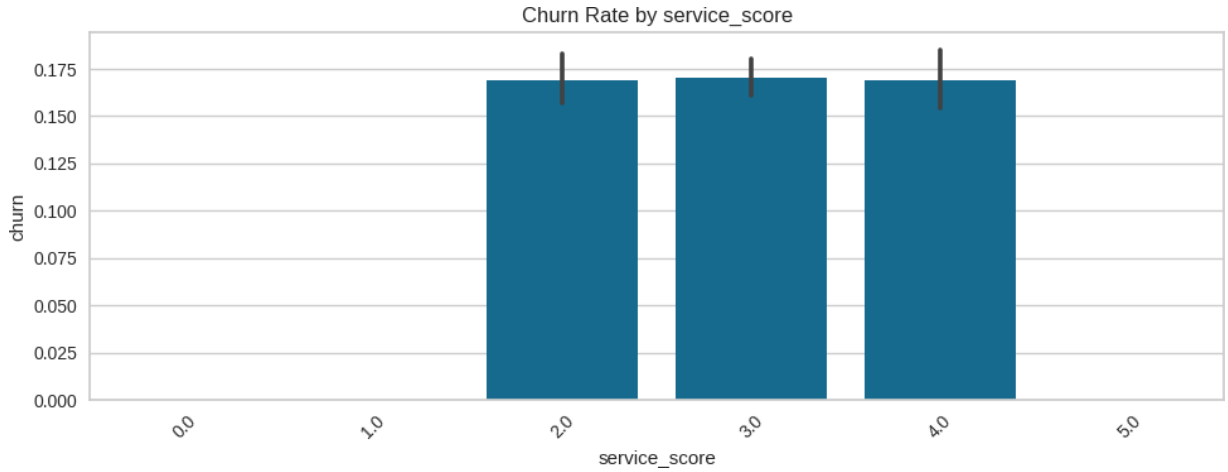

Figure 25 Churn Rate by Service Score

Churn Rate by Account Segment

Churn rate is higher in regular plus and lowest in super plus. The customers are mostly in regular plus and the company should focus on retaining them.

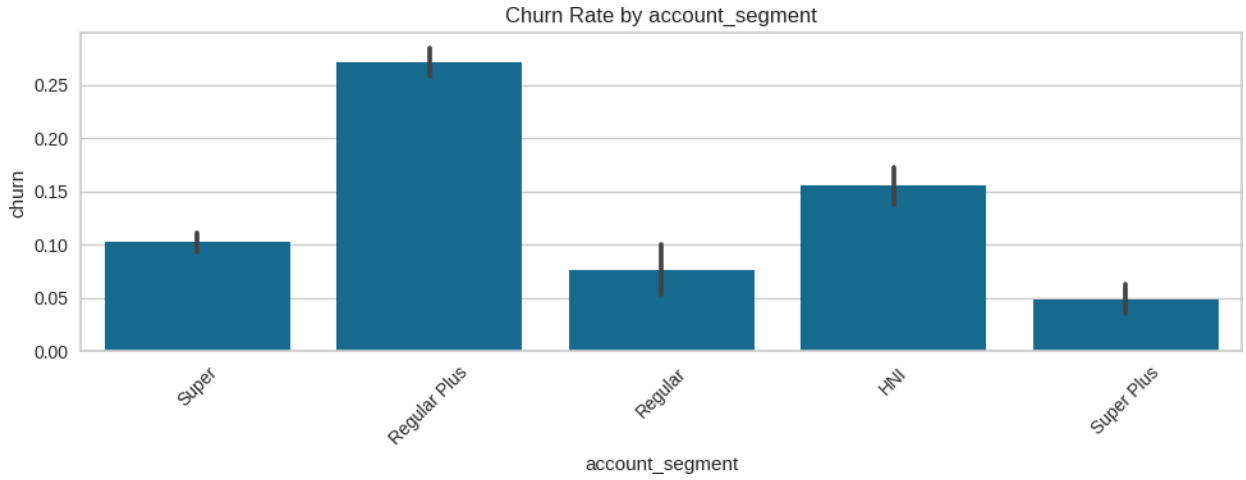

Figure 26 Churn Rate by Account Segment

Churn Rate by Customer Care Agent Score

The churn rate is higher with star rating 5 and lower in star rating 1. It seems that the customers are churning irrespective of how good the customer care agent performs.

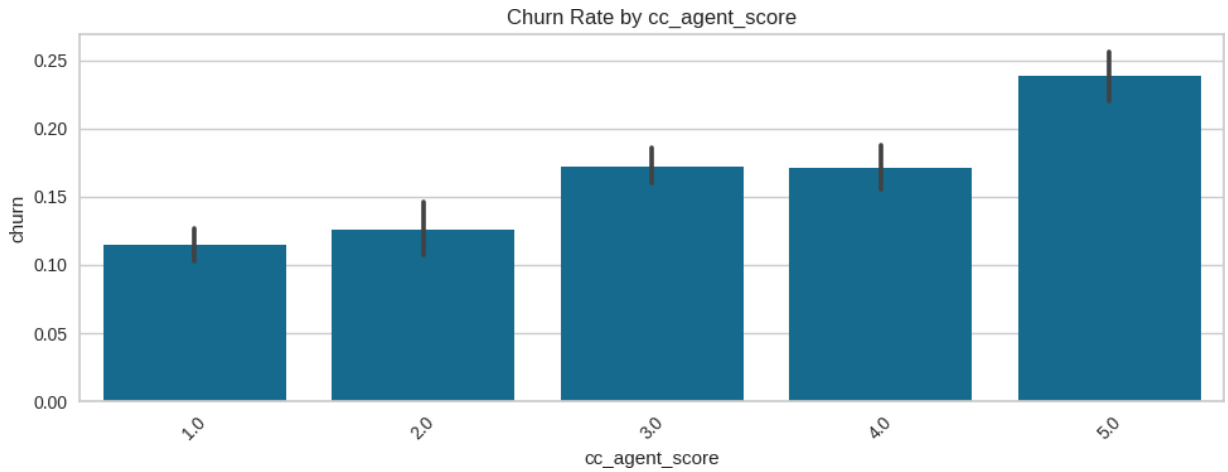

Figure 27 Churn Rate by Customer Care Agent Score

Churn Rate by Marital Status

The churn rate is higher among single customers and lowest in married customers.

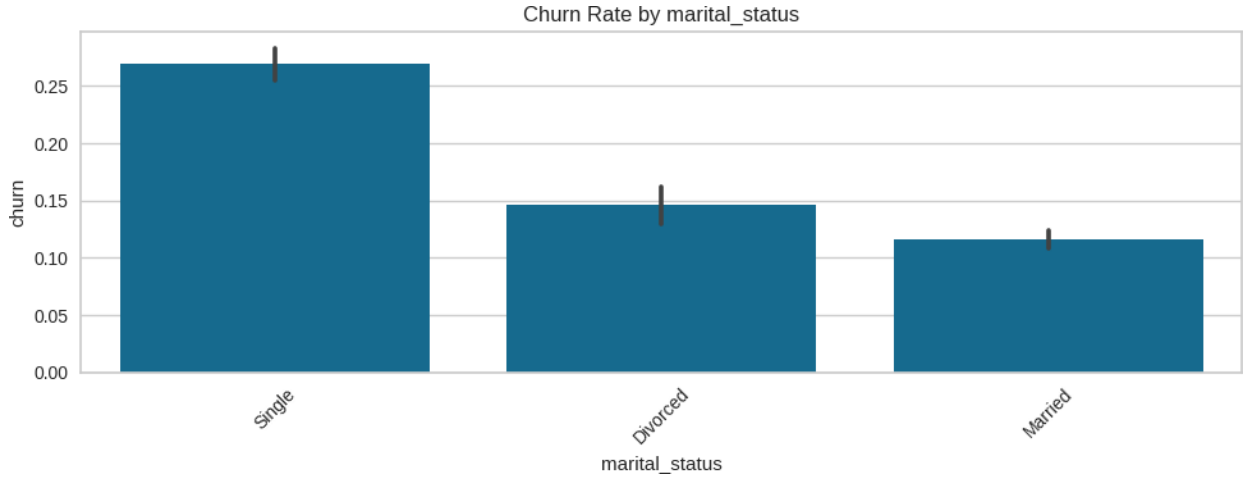

Figure 28 Churn Rate by Marital Status

Churn Rate by Complain Last Year

The churn rate is higher when the complaints are high and low when the complaints are low.

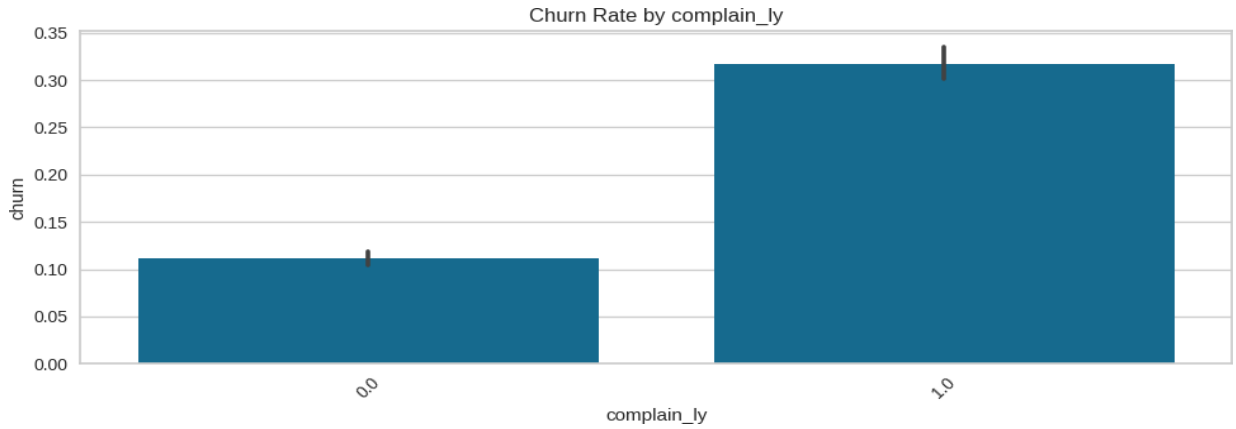

Figure 29 Churn Rate by Complain Last Year

Churn Rate by Login Device

The churn rate is higher for the customers logging in through computer in comparison to mobile devices. This might be due to better User Experience in mobile application in comparison to the computer website.

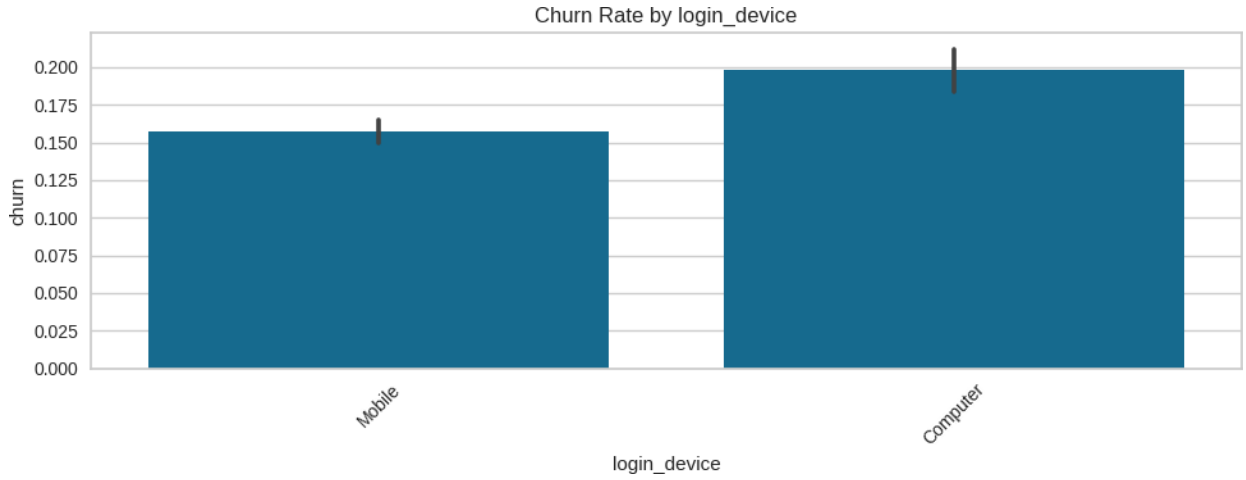

Figure 30 Churn Rate by Login Device

Treatment on dataset

Treatment of missing value and column replacement

1. Tenure

   Tenure consists of 102 missing values along with special character as '#'. We will replace the # with NaN. Subsequently, we will impute the values since we don't want to lose those many records. Tenure being a categorical variable we will impute the column with mode.

2. City tier

   City tier is a categorical variable and we impute the missing values with mode. The missing values sum up to 112 records.

3. Payment

   Payment is a categorical variable and we would want to fill it with mode. There are 109 records with missing values.

4. Gender

   This column contains male and female whereas it also has variables such as F and M. We need to replace the F and M with female and male respectively. Additionally there are 108 records with missing values and we impute the missing values with mode.

5. Service score

   There are 98 records which are missing. We will impute with mode since it is a categorical variable.

6. Account user count

   Account user count is a numeric column and there are 112 missing values. We will impute the missing values with rounded median since we are counting the number of customers it does not make sense to have decimal values. There are also special characters involved such as '@' and we will replace it with NaN and impute the missing value.

7. Account segment

   There are 97 missing records. We will impute the missing values with mode since this is a categorical variable. We have Regular + , Super +, Regular Plus and Super Plus. Hence we replace the characters with words which ultimately are converted to only Regular Plus and Super Plus.

8. Customer care agent score

   There are 116 missing records in this column. Being a categorical variable we will impute the missing values with mode.

9. Marital Status

   There are 212 missing records and we will impute the missing value with mode.

10. Rev per month

    There are 102 missing records and special characters such as + are also involved. We will replace the + character with NaN and then impute the missing value with median.

11. Complain Last Year

    This is a binary variable column. There are 357 missing records and we will impute this with mode.

12. Revenue growth year on year

    There are no missing records. However there is the presence of a special character which is $. We will replace it with NaN and impute with the median.

13. Coupon used for payment

    There are no missing records however there are presence of special characters such as $, * and #. We will replace this with NaN and then impute the value with a rounded median.

14. Day since customer care connect

    There are 357 records and '$' values in the column. We will replace the $ symbol with NaN. Since this is a numeric variable we will impute the missing values with median.

15. Cashback

    There are 471 records with missing records and '$' symbol. We will replace the $ symbol with NaN and then impute with the median.

16. Login Device

    There are 221 missing records and the presence of '&&&&' characters. We will replace it with NaN and then impute the missing with mode.

Treatment of duplicate records

There are no duplicate records in the datasets. Hence, no need to treat duplicate values.

Outlier Detection

We check the outliers in the numerical columns which are 'cc_contacted_ly', 'account_user_count', 'rev_per_month', 'rev_growth_yoy', 'coupon_used_for_payment', 'day_since_cc_connect', 'cashback'.

The treatment such as capping or flooring or dropping the rows are not necessary since the outliers are valid in this business case.

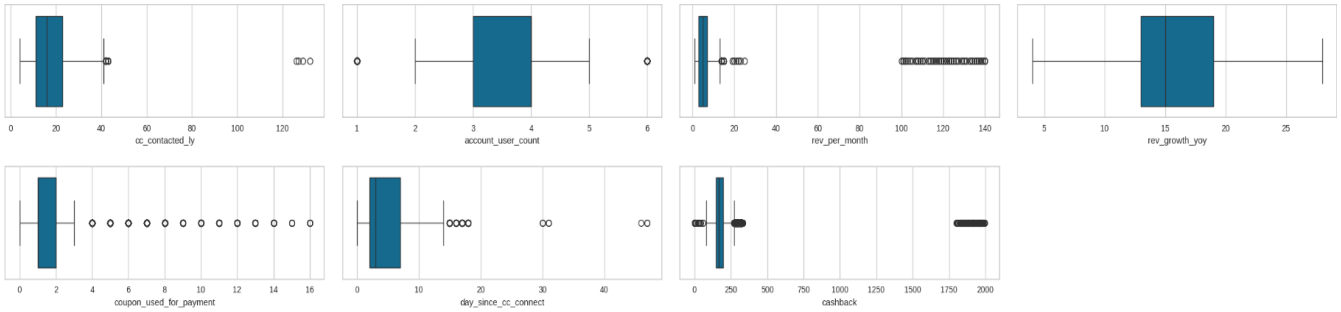

Figure 31 Outlier Detection

Conclusion on EDA

The dataset consists of account-level behavioral and financial metrics aimed at predicting churn. After a thorough univariate and bivariate analysis:

- **Customer Demographics & Behavior:** Majority of the customers are from City Tier 1, are male, and prefer mobile for logging in. Most customers have rated service and customer care between 2–4 stars. Around 42% use debit cards and 31% credit cards as preferred payment methods.

- **Churn Rate:** Approximately 16.8% of accounts have churned. Churn is higher for accounts with shorter tenure, from lower-tier cities, and those using cash on delivery or e-wallets. Accounts with single customers, low service scores, or complaints are more prone to churn.

- **Numeric Insights:** Variables like revenue, cashback, and coupon usage show high variability and outliers, though they are valid due to account diversity.

- **Correlation Analysis:** No strong correlations observed. However, coupon usage shows mild correlation with account size and days since last customer care interaction.

- **Data Cleaning & Imputation:** Special characters (like $, #, @, etc.) were handled and missing values were imputed using median or mode, depending on the variable type. No duplicate records were found.

- **Outliers:** Detected but retained, as they reflect legitimate business behavior and customer diversity.

K Means Clustering

Elbow Curve Analysis

The optimal number of clusters for K-means can often be identified by analyzing the "elbow point" in the distortion scores.

In this case the K seems to be appropriate at 4 since the distortion is less. The reduction plateaus at 4 leading to avoiding over segmentation and creating an optimal balance.

```
Number of Clusters: 1   Average Distortion: 4.068209882803089
Number of Clusters: 2   Average Distortion: 3.9385164210798767
Number of Clusters: 3   Average Distortion: 3.837706691166213
Number of Clusters: 4   Average Distortion: 3.723374286871781
Number of Clusters: 5   Average Distortion: 3.660334899749053
Number of Clusters: 6   Average Distortion: 3.596498284111274
Number of Clusters: 7   Average Distortion: 3.528038072477468
Number of Clusters: 8   Average Distortion: 3.495665631856114
Text(0.5, 1.0, 'Selecting k with the Elbow Method')
```

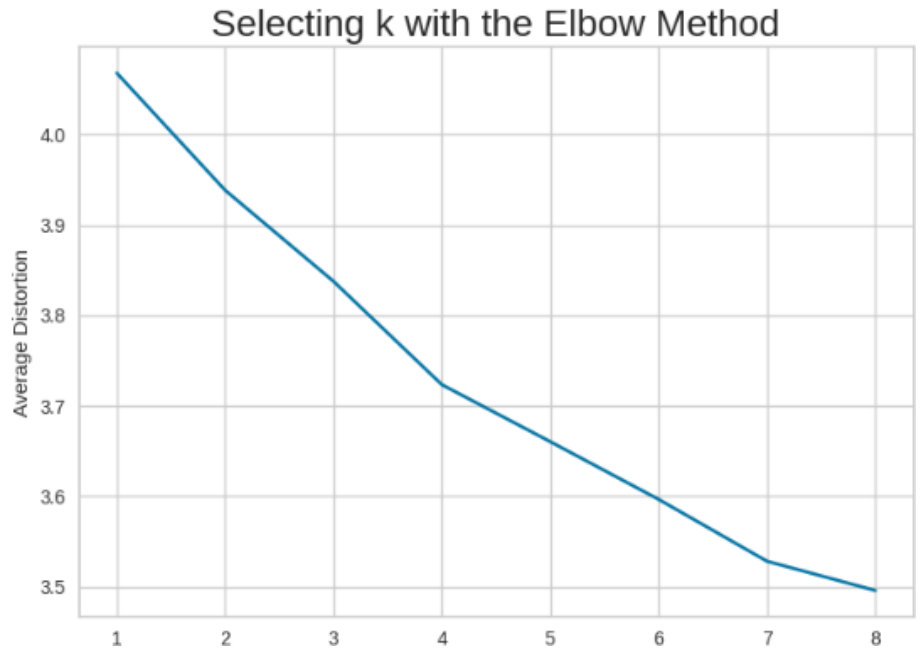

Figure 32 K value with Elbow Method

Checking Silhouette Scores

Looking at the cluster scores we can say that 2 and 3 are good since they are near to 1 and shows that there is a distinct gap between clusters.

```
For n_clusters = 2, silhouette score is 0.06802673101155947
For n_clusters = 3, silhouette score is 0.07500755001300462
For n_clusters = 4, silhouette score is 0.051892567696052064
For n_clusters = 5, silhouette score is 0.04951765660148089
For n_clusters = 6, silhouette score is 0.05677022444495208
For n_clusters = 7, silhouette score is 0.05969686924545373
For n_clusters = 8, silhouette score is 0.06444159796996325
For n_clusters = 9, silhouette score is 0.07139021086140723
[<matplotlib.lines.Line2D at 0x7a0ded643f50>]
```

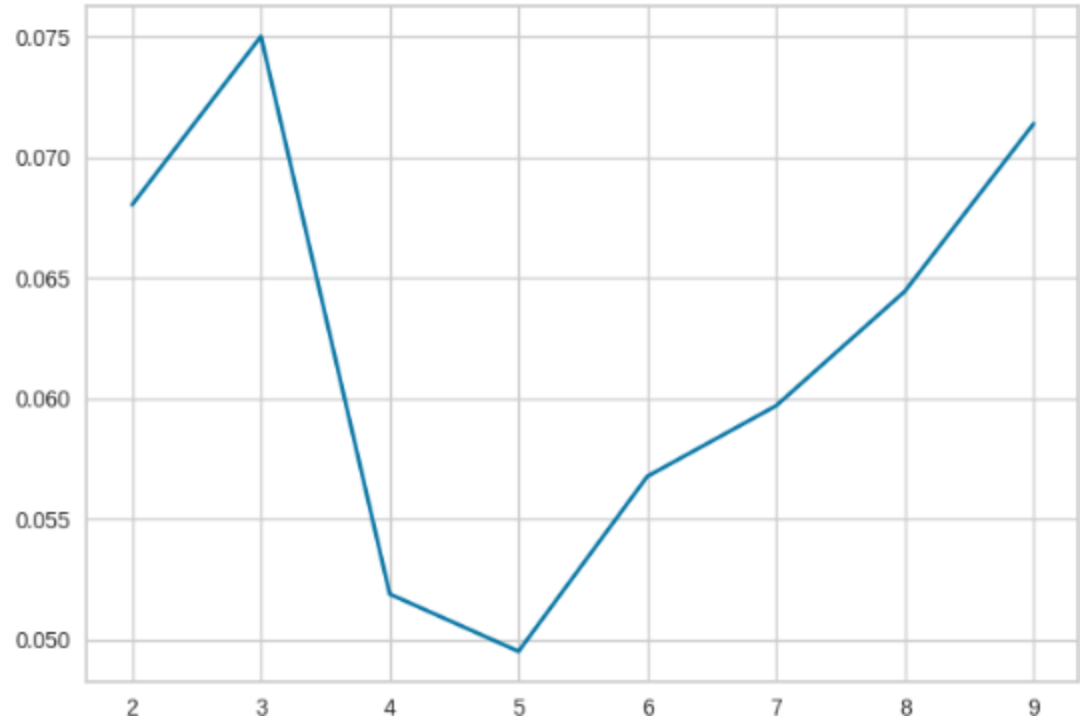

Figure 33 K value with Silhouette Score

Let us try putting 2, 3 and 4 in silhouette plots to understand the optimal value for clusters.

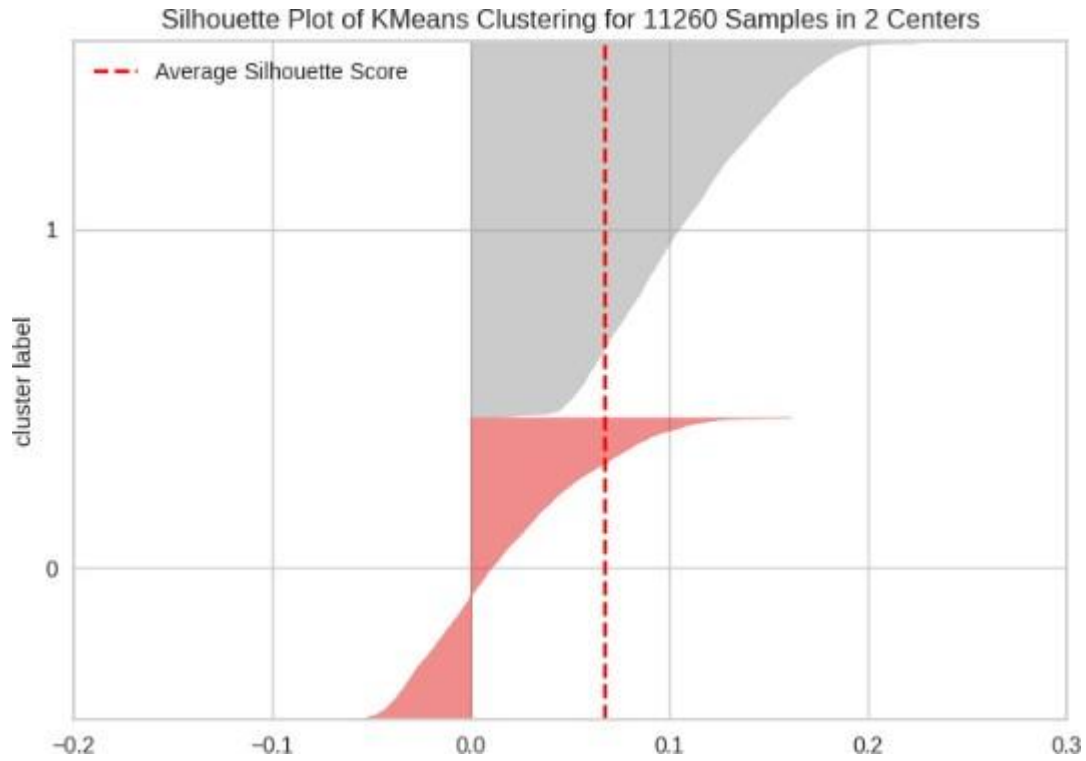

Figure 34 Silhouette Plot for 2 clusters

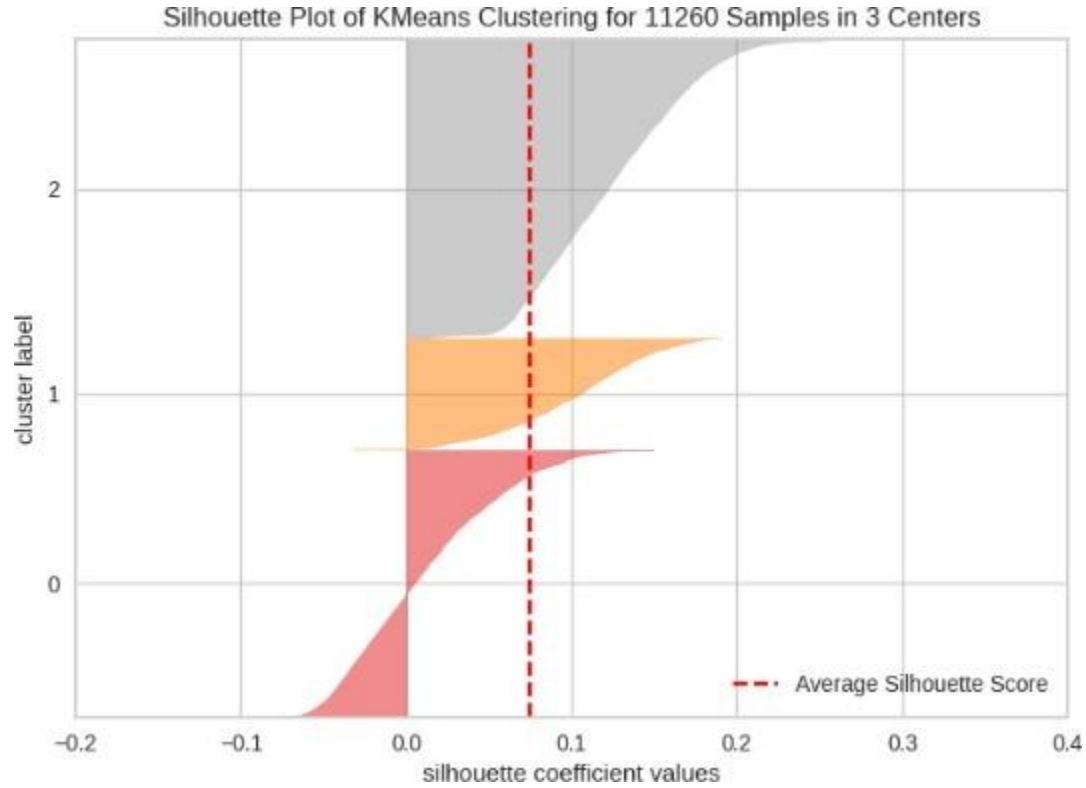

Figure 35 Silhouette Plot for 3 clusters

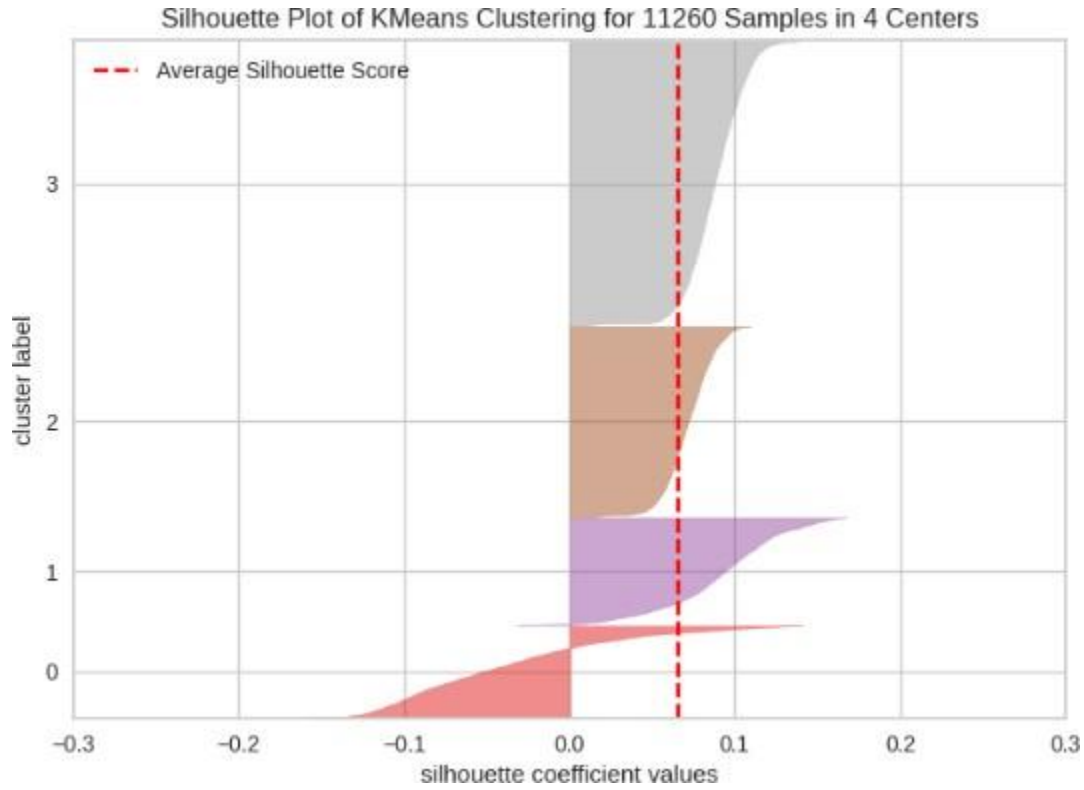

Figure 36 Silhouette Plot for 4 clusters

In the silhouette plot we need to see if the cluster crosses the average silhouette score which is the red dotted line and as we can see, all the plots cross the average silhouette score except for one.

Additionally, we need to see how wide each cluster is to determine the wide fluctuation. In this case, we can see only a plot with 3 and 4 clusters has less wide fluctuation.

Optimal Number of Cluster

The optimal number of clusters can be chosen as 3 since there is a knick at the elbow plot and also in silhouette plot, the clusters are above average silhouette score and the wide fluctuation of the clusters is less. We haven't chosen 4 because the misclassification is high when there are 4 clusters since it goes beyond -0.1 whereas the misclassification in 3 clusters stays between the range of 0 and -0.1.

Boxplot by Cluster

The boxplot shows the cluster variation with each variable. It determines where the cluster lies in each variable dataset.

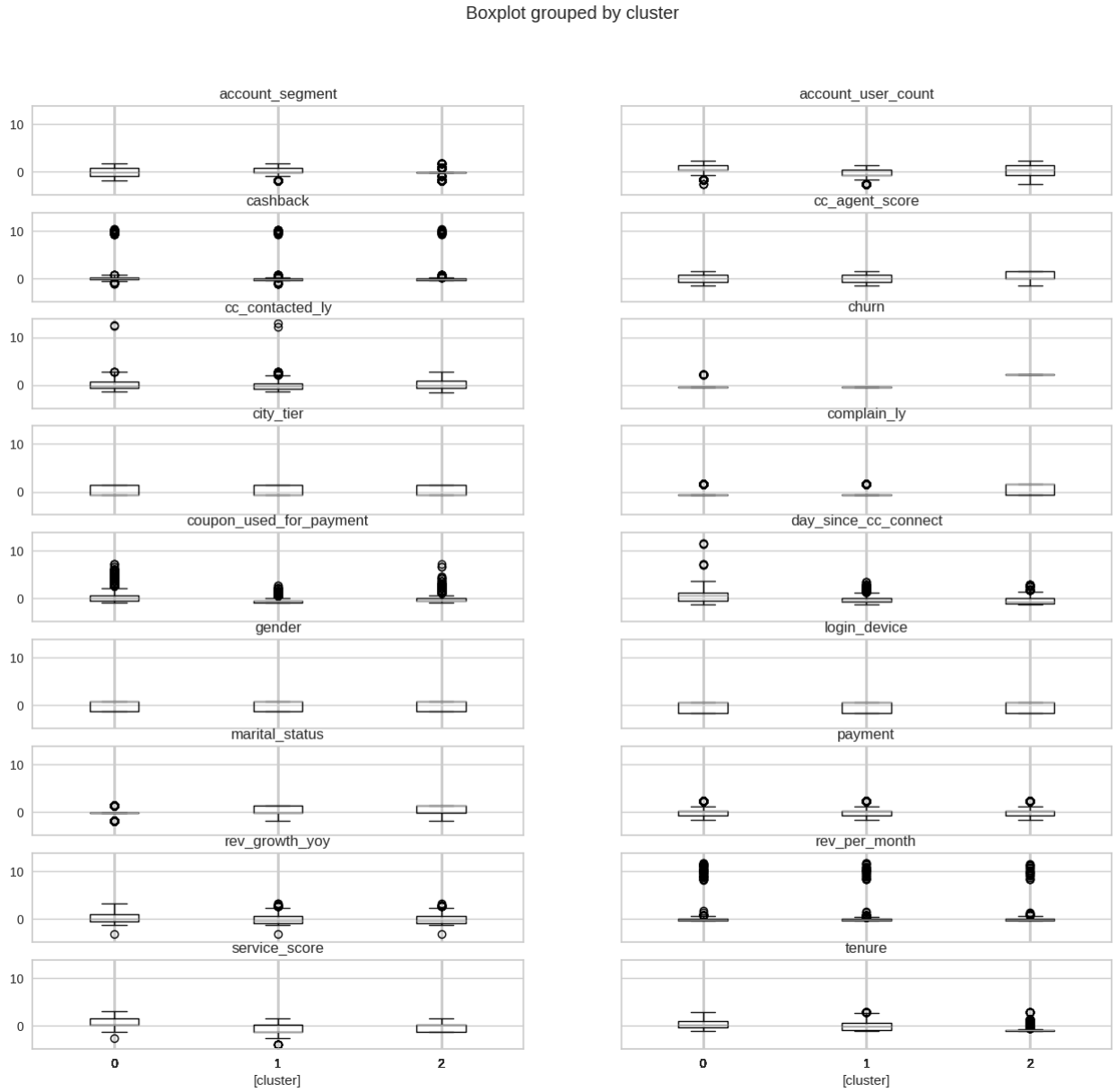

Figure 37 Boxplot by Cluster

Analysis of clusters

Cluster 0

Key traits:

- High account_user_count and longer tenure – these users likely have more stable or older relationships with the service.
- Higher service_score and cc_agent_score – generally satisfied customers with better agent interaction.
- Lower churn – they tend to stay loyal.
- Low complaint_ly and cc_contacted_ly – fewer complaints or service issues reported.
- Lower cashback and coupon_used_for_payment – possibly less price-sensitive.
- Higher payment and rev_per_month – higher revenue-generating customers.
- Moderate city_tier

Summary: Likely high-value, low-maintenance customers with strong loyalty and minimal complaints.

Cluster 1

Key traits:

- Average values across most metrics – a balanced profile without any strong highs or lows.
- Moderate account_user_count, cashback, rev_per_month, and payment – they engage decently but not heavily.
- Intermediate scores for service_score and cc_agent_score.
- Moderate coupon use and complaints.
- Slightly lower tenure compared to Cluster 0.

Summary: A middle-ground segment – not highly engaged or problematic. Could be nurtured into more profitable users with targeted offers.

Cluster 2

Key traits:

- Lowest account_user_count and tenure – newer or less engaged users
- High complaints (complain_ly), cc_contacted_ly, and churn – indicates dissatisfaction or high service needs.
- Low service_score and cc_agent_score – poor service experience.
- High coupon_used_for_payment and cashback – more price-sensitive or deal-seeking users.
- Lower payment and revenue per month – less profitable customers.
- Slightly higher day_since_cc_connect – slower to connect with customer care or sign of delayed engagement.

Summary: High-risk segment with low revenue and high complaints. Likely needs intervention, better support, or targeted retention strategies.

Model Building

Introduction

We have analysed our data and now we will move forward to create models which would help in classifying if a customer would churn or not.

In order to do that, we would split our dataset to training (70%) and testing (30%) dataset and then would train our model on training dataset and then test the accuracy and other metrics on the testing dataset.

For this business case we will be focusing on f1 score since the false positives and false negatives will cause business a potential threat and business cannot afford to to lose accounts.

Logistic Regression

Analysis

Logistic regression is a machine learning algorithm used for binary classification, predicting the probability of a binary outcome.

The summary of logistic regression is as below:

Logit Regression Results

Dep. Variable:	churn	No. Observations:	7882
Model:	Logit	Df Residuals:	7864
Method:	MLE	Df Model:	17
Date:	Sat, 19 Apr 2025	Pseudo R-squ.:	0.3119
Time:	10:53:11	Log-Likelihood:	-2457.4
converged:	True	LL-Null:	-3571.1
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.8467	0.292	-9.757	0.000	-3.419	-2.275
tenure	-0.1774	0.007	-24.252	0.000	-0.192	-0.163
city_tier	0.3174	0.040	7.977	0.000	0.239	0.395
cc_contacted_ly	0.0252	0.004	6.029	0.000	0.017	0.033
payment	-0.0720	0.036	-2.000	0.045	-0.142	-0.001
gender	0.2877	0.076	3.794	0.000	0.139	0.436
service_score	-0.0510	0.055	-0.934	0.350	-0.158	0.056
account_user_count	0.3252	0.040	8.232	0.000	0.248	0.403
account_segment	-0.3420	0.037	-9.293	0.000	-0.414	-0.270
cc_agent_score	0.2917	0.027	10.838	0.000	0.239	0.344
marital_status	0.5561	0.056	9.880	0.000	0.446	0.666
rev_per_month	0.0123	0.003	4.497	0.000	0.007	0.018
complain_ly	1.6272	0.076	21.335	0.000	1.478	1.777
rev_growth_yoy	-0.0152	0.010	-1.526	0.127	-0.035	0.004
coupon_used_for_payment	0.1297	0.022	6.017	0.000	0.087	0.172
day_since_cc_connect	-0.1168	0.013	-8.888	0.000	-0.143	-0.091
cashback	0.0002	0.000	0.936	0.349	-0.000	0.001
login_device	-0.3648	0.080	-4.583	0.000	-0.521	-0.209

Figure 1 Summary of logistic regression

When we train the data with logistic regression model we see that the confusion matrix performance metric is as below:

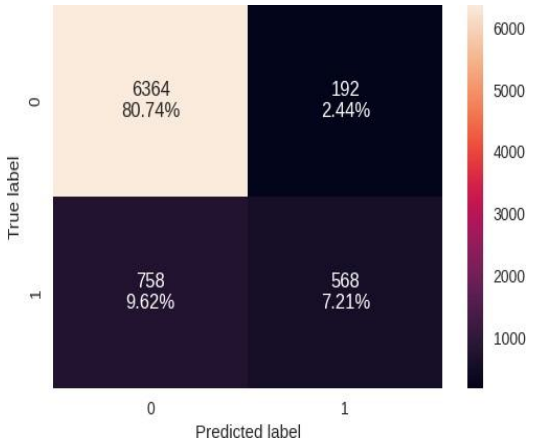

Figure 2 Confusion Matrix - Train - Logistic Regression

	Accuracy	Recall	Precision	F1
0	0.879472	0.428356	0.747368	0.544583

Figure 3 Performance Metric - train - Logistic Regression

When we test the data with logistic regression model we see that the confusion matrix performance metric is as below:

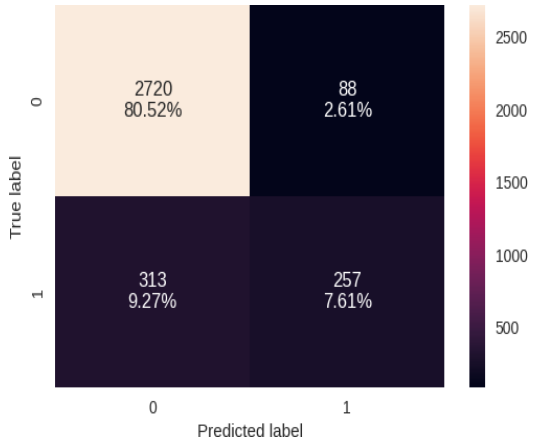

Figure 4 Confusion Matrix -Test - Logistic Regression

	Accuracy	Recall	Precision	F1
0	0.881291	0.450877	0.744928	0.561749

Figure 5 Performance Metric - Test - Logistic Regression

Inference

With similar training (87.95%) and test accuracy (88.13%), the model shows a good fit. However, the low F1 score of 56.17% indicates that it struggles to balance precision and recall effectively, limiting its practical use in classification tasks.

Naive Bayes

Analysis

Naive Bayes is a probabilistic machine learning algorithm used for classification.

When we train the data with Naive Bayes model, we see that the confusion matrix performance metric is as below:

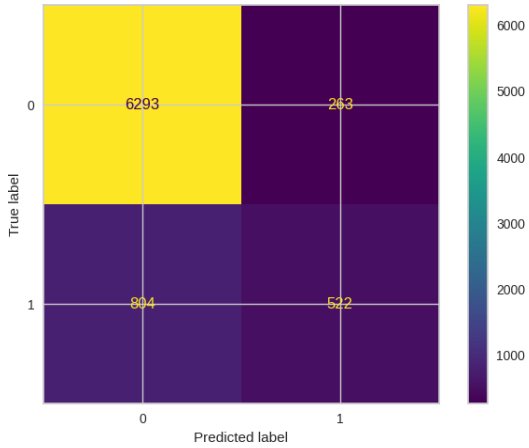

Figure 6 Confusion Matrix - Train - Naive Bayes

	Accuracy	Recall	Precision	F1
0	0.864628	0.393665	0.664968	0.494552

Figure 7 Performance Metric - Train - Naive Bayes

When we test the data with Naive Bayes model we see that the confusion matrix performance metric is as below:

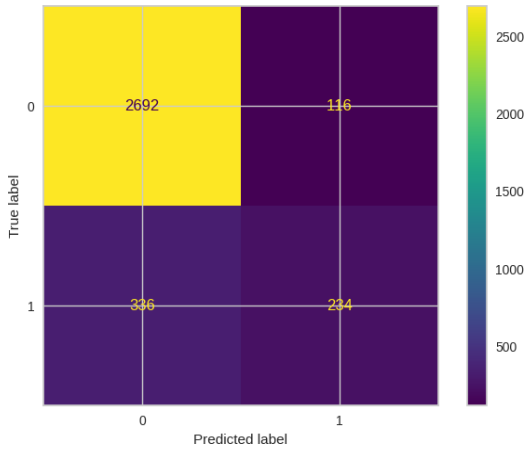

Figure 8 Confusion Matrix - Test - Naive Bayes

	Accuracy	Recall	Precision	F1
0	0.866193	0.410526	0.668571	0.508696

Figure 9 Performance Metric- Test - Naive Bayes

Inference

Naive Bayes has close training (86.46%) and test accuracy (86.61%), indicating no overfitting. Despite this, a low F1 score of 50.87% suggests weak predictive performance, likely due to its strong independence assumptions.

Decision Tree

Analysis

A decision tree is a machine learning algorithm that uses a tree-like structure to model decisions or predictions.

When we train the data with Decision Tree model we see that the confusion matrix performance metric is as below:

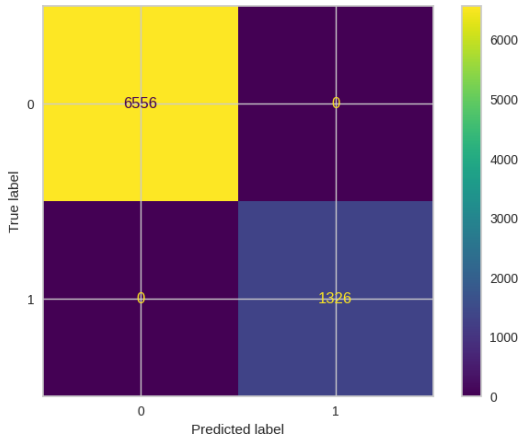

Figure 10 Confusion Matrix - Train - Decision tree

	Accuracy	Recall	Precision	F1
0	1.0	1.0	1.0	1.0

Figure 11 Performance Metric - Train - Decision tree

When we test the data with Decision Tree model we see that the confusion matrix performance metric is as below:

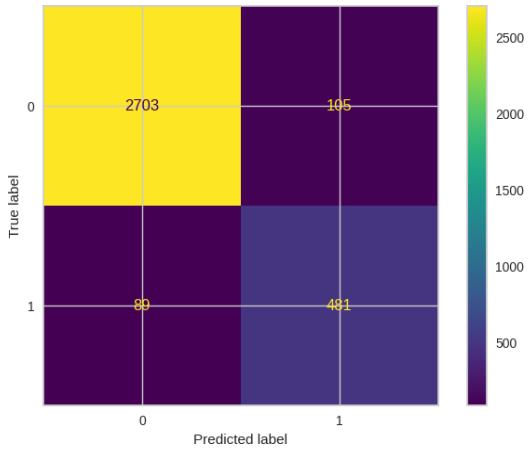

Figure 12 Confusion Matrix - Test - Decision tree

	Accuracy	Recall	Precision	F1
0	0.94257	0.84386	0.820819	0.83218

Figure 13 Performance Metric - Test - Decision tree

Inference

While training accuracy is perfect (100%), the test accuracy drops to 94.26%, showing slight overfitting. Nevertheless, the high F1 score of 83.22% confirms that it still performs well overall, capturing both precision and recall efficiently.

KNN Classifier

A K-Nearest Neighbors (KNN) classifier is a simple supervised machine learning algorithm used for classifying data points.

When we train the data with KNN Classifier model we see that the confusion matrix performance metric is as below:

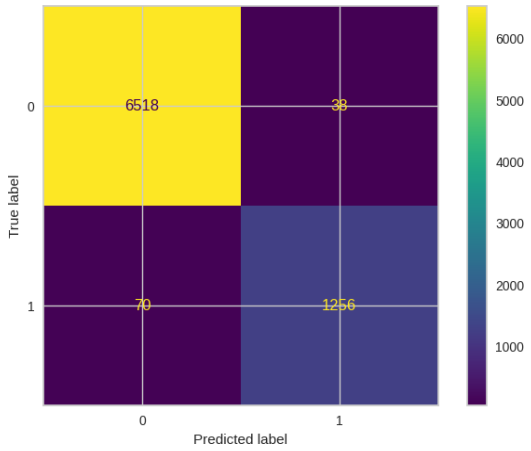

Figure 14 Confusion Matrix - Train - KNN Classifier

	Accuracy	Recall	Precision	F1
0	0.986298	0.94721	0.970634	0.958779

Figure 15 Performance Metric - Train - KNN Classifier

When we test the data with KNN Classifier model we see that the confusion matrix performance metric is as below:

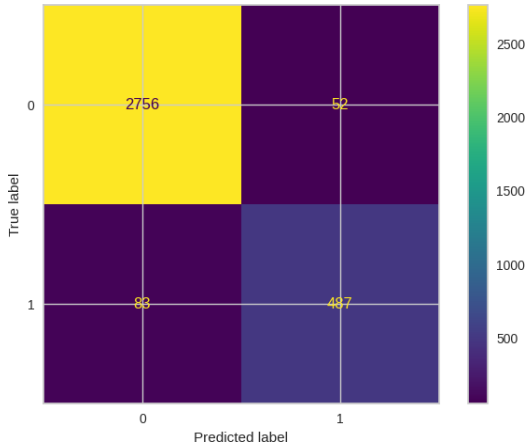

Figure 16 Confusion Matrix - Test - KNN Classifier

	Accuracy	Recall	Precision	F1
0	0.960036	0.854386	0.903525	0.878269

Figure 17 Performance Metric- Test - KNN Classifier

Inference

KNN maintains high training (98.63%) and strong test accuracy (96.00%) with a solid F1 score of 87.83%, indicating good generalization and a strong balance between bias and variance — a reliable performer.

Random Forest

Analysis

Random forest is a popular machine learning algorithm that uses an ensemble of decision trees to make predictions.

The summary of Random Forest is as below:

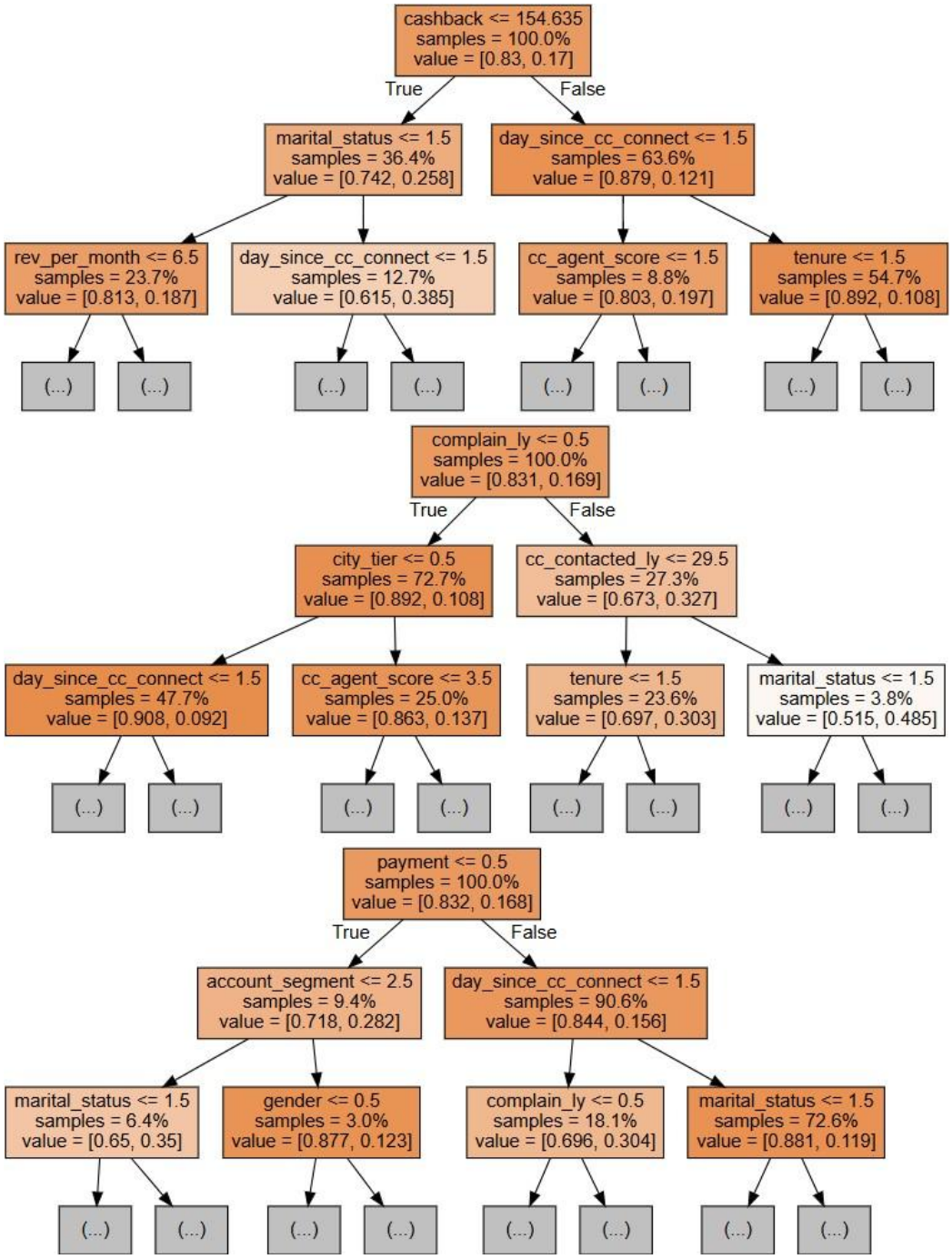

Figure 18 Summary of Random Forest

When we train the data with Random Forest model we see that the confusion matrix performance metric is as below:

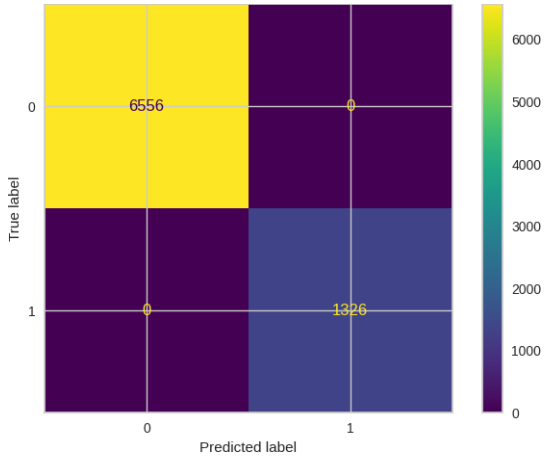

Figure 19 Confusion Matrix - Train - Random Forest

	Accuracy	Recall	Precision	F1
0	1.0	1.0	1.0	1.0

Figure 20 Performance Metric- Train - Random Forest

When we test the data with Random Forest model we see that the confusion matrix performance metric is as below:

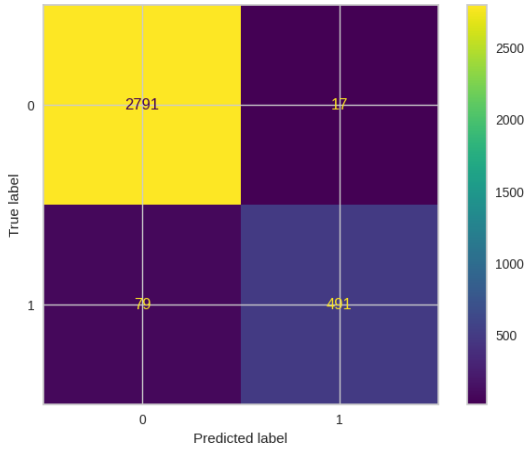

Figure 21 Confusion Matrix - Test - Random Forest

	Accuracy	Recall	Precision	F1
0	0.971581	0.861404	0.966535	0.910946

Figure 22 Performance Metric- Test - Random Forest

Inference

Random Forest shows perfect training accuracy (100%) and excellent test accuracy (97.22%). With a high F1 score of 91.28%, it captures positive cases effectively and generalizes well — a strong, well-fit model.

AdaBoost Classifier

Analysis

AdaBoost is an ensemble machine learning method that combines multiple weak classifiers (or "stumps") to create a strong classifier.

When we train the data with AdaBoost Classifier model we see that the confusion matrix performance metric is as below:

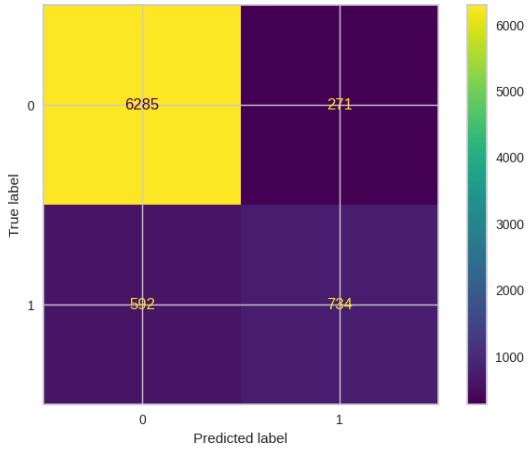

Figure 23 Confusion Matrix - Train - AdaBoost Classifier

	Accuracy	Recall	Precision	F1
0	0.89051	0.553544	0.730348	0.629773

Figure 24 Performance Metric - Train - AdaBoost Classifier

When we test the data with AdaBoost Classifier model we see that the confusion matrix performance metric is as below:

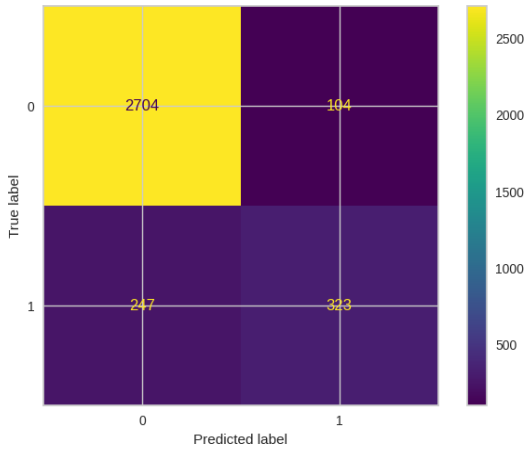

Figure 25 Confusion Matrix - Test - AdaBoost Classifier

	Accuracy	Recall	Precision	F1
0	0.896092	0.566667	0.75644	0.647944

Figure 26 Performance Metric - Test - AdaBoost Classifier

Inference

With moderately close training (89.05%) and test accuracy (89.61%), AdaBoost is well-fit. However, its F1 score of 64.79% shows only average performance in balancing recall and precision — better than basic models, but not optimal.

XGBoost Classifier

Analysis

XGBoost (eXtreme Gradient Boosting) is a powerful, open-source machine learning algorithm that implements gradient boosting, a technique for building predictive models by combining multiple weak models (like decision trees) into a strong one, particularly effective for structured or tabular data.

When we train the data with XGBoost Classifier model we see that the confusion matrix performance metric is as below:

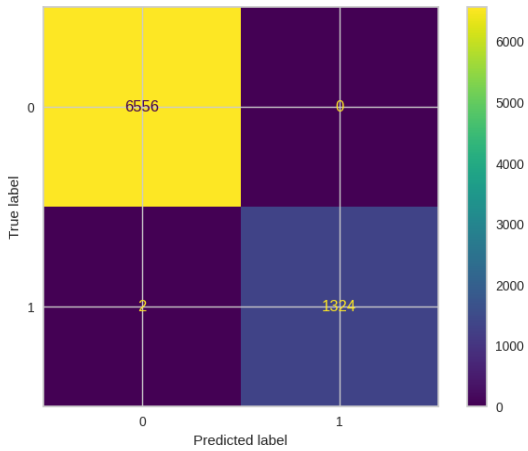

Figure 27 Confusion Matrix - Train - XGBoost Classifier

	Accuracy	Recall	Precision	F1
0	0.999746	0.998492	1.0	0.999245

Figure 28 Performance Metric - Train - XGBoost Classifier

When we test the data with XGBoost Classifier model we see that the confusion matrix performance metric is as below:

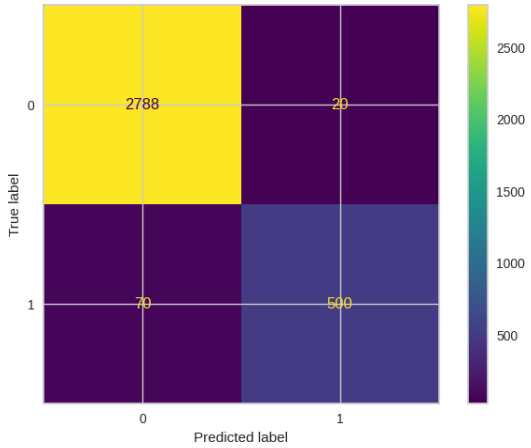

Figure 29 Confusion Matrix - Test - XGBoost Classifier

	Accuracy	Recall	Precision	F1
0	0.973357	0.877193	0.961538	0.917431

Figure 30 Performance Metric - Test - XGBoost Classifier

Inference

XGBoost combines near-perfect training accuracy (99.97%) with excellent test accuracy (97.34%) and a strong F1 score of 91.74%. It generalizes well with minimal overfitting, making it one of the best models in this comparison.

SVM Classifier

Analysis

A Support Vector Machine (SVM) classifier in machine learning is a supervised learning algorithm used for classifying data by finding the optimal hyperplane that maximizes the margin between different classes.

When we train the data with SVM Classifier model we see that the confusion matrix performance metric is as below:

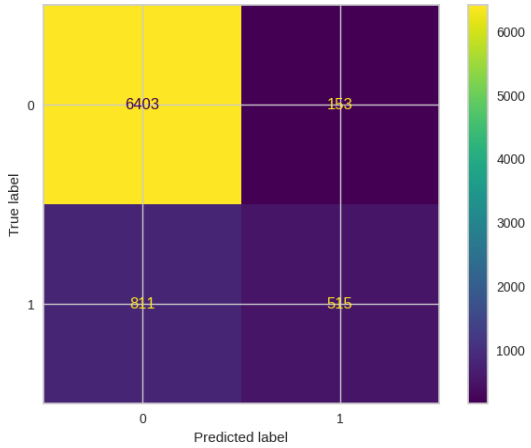

Figure 31 Confusion Matrix - Train - SVM Classifier

	Accuracy	Recall	Precision	F1
0	0.877696	0.388386	0.770958	0.51655

Figure 32 Performance Metric - Train - SVM Classifier

When we test the data with SVM Classifier model we see that the confusion matrix performance metric is as below:

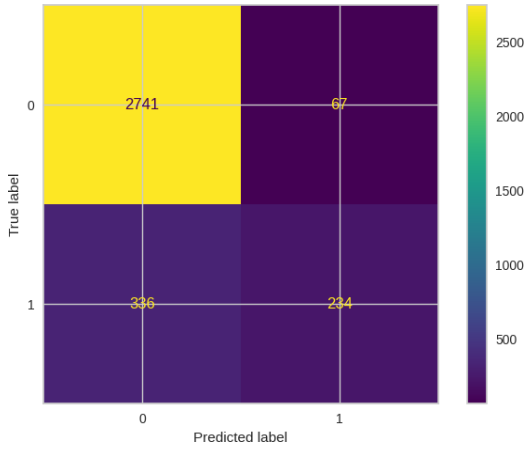

Figure 33 Confusion Matrix - Test - SVM Classifier

	Accuracy	Recall	Precision	F1
0	0.880699	0.410526	0.777409	0.537313

Figure 34 Performance Metric - Test - SVM Classifier

Inference

SVM shows a good fit with training (87.77%) and test accuracy (88.07%), but a low F1 score of 53.73% reveals weak recall performance. It may not be ideal if balanced classification is a priority.

Summary of Models

Below are the summary of the performance metrics of the models we have created:

	Model	Train_Accuracy	Test_Accuracy	Precision	Recall	F1_Score	Overfitting/Underfitting
0	Logistic Regression	0.879472	0.881291	0.744928	0.450877	0.561749	Good Fit
1	Naive Bayes	0.864628	0.866193	0.668571	0.410526	0.508696	Good Fit
2	Decision Tree	1.000000	0.942570	0.820819	0.843860	0.832180	Good Fit
3	KNN Classifier	0.986298	0.960036	0.903525	0.854386	0.878269	Good Fit
4	Random Forest	1.000000	0.971581	0.966535	0.861404	0.910946	Good Fit
5	AdaBoost Classifier	0.890510	0.896092	0.756440	0.566667	0.647944	Good Fit
6	XGBoost Classifier	0.999746	0.973357	0.961538	0.877193	0.917431	Good Fit
7	SVM Classifier	0.877696	0.880699	0.777409	0.410526	0.537313	Good Fit

Figure 35 Initial summary of models

Balanced Dataset

To ensure fair and effective model training, it is essential to address class imbalance within the dataset. Imbalanced data, where one class significantly outnumbers the other, can lead to biased models that perform poorly on the minority class. To mitigate this issue, we implemented data balancing techniques—specifically, oversampling using SMOTE (Synthetic Minority Oversampling Technique) and undersampling of the majority class. SMOTE creates synthetic examples of the minority class to enhance its representation, while undersampling reduces the number of majority class instances to match the minority class. These methods help our models learn more generalized decision boundaries and improve performance metrics, particularly recall and F1 score, which are critical when dealing with skewed data distributions.

OverSampling Model Building Method

Sampling Count

Before Oversampling, counts of label '1': 1326
Before Oversampling, counts of label '0': 6556

After Oversampling, counts of label '1': 6556
After Oversampling, counts of label '0': 6556

Model Building

We have created models with the oversampled data and below are the result of it:

	Model	Train_Accuracy	Test_Accuracy	Precision	Recall	F1_Score	Overfitting/Underfitting
0	Logistic Regression with Oversampled Data	0.800	0.768	0.408	0.835	0.548	Overfitting
1	Naive Bayes with Oversampled Data	0.779	0.751	0.383	0.779	0.514	Overfitting
2	Random Forest with Oversampled Data	1.000	0.973	0.951	0.888	0.918	Good Fit
3	XGBoost with Oversampled Data	0.999	0.964	0.923	0.861	0.891	Good Fit
4	AdaBoost with Oversampled Data	0.894	0.873	0.599	0.746	0.665	Overfitting
5	Decision Tree with Oversampled Data	1.000	0.952	0.852	0.867	0.859	Good Fit
6	KNN with Oversampled Data	0.935	0.821	0.483	0.874	0.622	Overfitting
7	SVM with Oversampled Data	0.802	0.778	0.418	0.811	0.552	Overfitting

Figure 36 Summary of models with oversampled data

Inferences

Logistic Regression with Oversampled Data

With a training accuracy of 80.0% and a lower test accuracy of 76.8%, the model exhibits overfitting. Despite a decent recall of 83.5%, the low F1 score of 54.8% shows poor precision and overall classification performance.

Naive Bayes with Oversampled Data

Naive Bayes slightly overfits with 77.9% training and 75.1% test accuracy. Although it achieves a recall of 77.9%, the F1 score of 51.4% remains weak due to low precision, limiting its effectiveness.

Random Forest with Oversampled Data

Random Forest achieves perfect training accuracy and high test accuracy (97.3%), suggesting excellent generalization. With a strong F1 score of 91.8%, it effectively balances recall and precision — a top-performing model.

XGBoost with Oversampled Data

XGBoost shows near-perfect training accuracy (99.9%) and high test accuracy (96.4%). The F1 score of 89.1% confirms strong classification ability, making this a well-balanced and effective model.

AdaBoost with Oversampled Data

AdaBoost slightly overfits, with training accuracy at 89.4% and lower test accuracy at 87.3%. Although recall is decent at 74.6%, a modest F1 score of 66.5% suggests room for improvement in precision.

Decision Tree with Oversampled Data

Decision Tree reaches perfect training accuracy but generalizes well with a test accuracy of 95.2%. A solid F1 score of 85.9% shows balanced precision and recall, making it a reliable model.

KNN with Oversampled Data

KNN shows signs of overfitting with high training accuracy (93.5%) and lower test accuracy (82.1%). While recall is strong at 87.4%, the F1 score of 62.2% reflects weaker precision.

SVM with Oversampled Data

SVM overfits slightly, with training at 80.2% and test accuracy at 77.8%. Despite a good recall (81.1%), the low F1 score of 55.2% suggests imbalanced predictions and limited overall performance.

UnderSampling Model Building Method

Sampling Count

Before Under Sampling, counts of label 'Yes': 1326
Before Under Sampling, counts of label 'No': 6556

After Under Sampling, counts of label 'Yes': 1326
After Under Sampling, counts of label 'No': 1326

Model Building

We have created models with the undersampled data and below are the result of it:

	Model	Train_Accuracy	Test_Accuracy	Precision	Recall	F1_Score	Overfitting/Underfitting
0	Logistic Regression with Undersampled Data	0.780	0.756	0.394	0.832	0.535	Overfitting
1	Naive Bayes with Undersampled Data	0.773	0.789	0.428	0.754	0.546	Overfitting
2	Random Forest with Undersampled Data	1.000	0.926	0.711	0.947	0.812	Overfitting
3	XGBoost with Undersampled Data	1.000	0.922	0.697	0.954	0.806	Overfitting
4	AdaBoost with Undersampled Data	0.822	0.847	0.531	0.814	0.643	Overfitting
5	Decision Tree with Undersampled Data	1.000	0.879	0.592	0.918	0.719	Overfitting
6	KNN with Undersampled Data	0.860	0.743	0.377	0.796	0.512	Overfitting
7	SVM with Undersampled Data	0.783	0.763	0.401	0.816	0.537	Overfitting

Figure 37 Summary of models with undersampled data

Inferences

Logistic Regression with Undersampled Data

This model shows overfitting, with training accuracy at 78.0% and test accuracy at 75.6%. Despite achieving a high recall of 83.2%, the F1 score of 53.5% is pulled down by low precision, making it a suboptimal classifier.

Naive Bayes with Undersampled Data

Naive Bayes slightly overfits with training accuracy at 77.3% and test accuracy at 78.9%. It has a balanced recall (75.4%) and precision (42.8%), resulting in a moderate F1 score of 54.6%.

Random Forest with Undersampled Data

Random Forest overfits, showing perfect training accuracy but a drop to 92.6% on test data. Still, it performs well overall with a strong F1 score of 81.2%, supported by excellent recall (94.7%).

XGBoost with Undersampled Data

XGBoost also overfits, with 100% training accuracy and 92.2% test accuracy. However, its recall (95.4%) and F1 score (80.6%) make it a robust model, although slightly less generalizable than on oversampled data.

AdaBoost with Undersampled Data

AdaBoost displays moderate overfitting. With training accuracy at 82.2% and test accuracy at 84.7%, it maintains decent recall (81.4%) and an F1 score of 64.3%, showing reasonably balanced performance.

Decision Tree with Undersampled Data

Decision Tree overfits heavily, with perfect training accuracy and 87.9% test accuracy. Although recall is high (91.8%), the lower precision causes the F1 score to drop to 71.9%, reflecting potential generalization issues.

KNN with Undersampled Data

KNN suffers from overfitting, with 86.0% training vs 74.3% test accuracy. Despite high recall (79.6%), the F1 score of 51.2% reveals poor precision, weakening the overall model effectiveness.

SVM with Undersampled Data

SVM shows overfitting as well, with a training accuracy of 78.3% and test accuracy of 76.3%. Though recall is good (81.6%), the F1 score of 53.7% is limited by low precision, making this model less ideal.

Best Model Selection

	Model	Train_Accuracy	Test_Accuracy	Precision	Recall	F1_Score	Overfitting/Underfitting
10	Random Forest with Oversampled Data	1.00	0.97	0.95	0.89	0.92	Good Fit
6	XGBoost Classifier	1.00	0.97	0.96	0.88	0.92	Good Fit
4	Random Forest	1.00	0.97	0.96	0.85	0.91	Good Fit
11	XGBoost with Oversampled Data	1.00	0.96	0.92	0.86	0.89	Good Fit
3	KNN Classifier	0.99	0.96	0.90	0.85	0.88	Good Fit
13	Decision Tree with Oversampled Data	1.00	0.95	0.85	0.87	0.86	Good Fit
2	Decision Tree	1.00	0.94	0.82	0.84	0.83	Good Fit
18	Random Forest with Undersampled Data	1.00	0.93	0.71	0.95	0.81	Overfitting
19	XGBoost with Undersampled Data	1.00	0.92	0.70	0.95	0.81	Overfitting
21	Decision Tree with Undersampled Data	1.00	0.88	0.59	0.92	0.72	Overfitting
12	AdaBoost with Oversampled Data	0.89	0.87	0.60	0.75	0.66	Overfitting
5	AdaBoost Classifier	0.89	0.90	0.76	0.57	0.65	Good Fit
20	AdaBoost with Undersampled Data	0.82	0.85	0.53	0.81	0.64	Overfitting
14	KNN with Oversampled Data	0.94	0.82	0.48	0.87	0.62	Overfitting
0	Logistic Regression	0.88	0.88	0.74	0.45	0.56	Good Fit
15	SVM with Oversampled Data	0.80	0.78	0.42	0.81	0.55	Overfitting
8	Logistic Regression with Oversampled Data	0.80	0.77	0.41	0.84	0.55	Overfitting
17	Naive Bayes with Undersampled Data	0.77	0.79	0.43	0.75	0.55	Overfitting
7	SVM Classifier	0.88	0.88	0.78	0.41	0.54	Good Fit
23	SVM with Undersampled Data	0.78	0.76	0.40	0.82	0.54	Overfitting
16	Logistic Regression with Undersampled Data	0.79	0.76	0.39	0.82	0.53	Overfitting
9	Naive Bayes with Oversampled Data	0.78	0.75	0.38	0.78	0.51	Overfitting
22	KNN with Undersampled Data	0.86	0.74	0.38	0.80	0.51	Overfitting
1	Naive Bayes	0.86	0.87	0.67	0.41	0.51	Good Fit

Figure 38 Summary of all models built

As per our analysis, we have given importance to F1 score. Hence we have sorted the models based on F1 score. The best model we have selected is Random Forest with Oversampled Data.

Below are the performance metrics:

- Train Accuracy: 100%
- Test Accuracy: 97.3%
- Precision: 95.11%
- Recall: 88.77%
- F1 Score: 91.83%
- Fit Status: Good Fit (no overfitting or underfitting signs)

Below are the justifications to select the model:

1. Highest F1 Score (0.9183)

- F1 Score balances both precision and recall, and is especially important in imbalanced classification problems.
- Precision: 0.9511 — very low false positives
- Recall: 0.8877 — very low false negatives
- F1 Score: 0.9183 — the highest among all models
- This indicates that the model is both precise and comprehensive in identifying the positive class.

2. Excellent Generalization

    - Train Accuracy: 1.000
    - Test Accuracy: 0.973
    - While a perfect training score can suggest overfitting, in this case the high test accuracy (97.3%) and the Good Fit status imply that the model generalizes well and handles the oversampled data effectively.

3. Consistently Strong Compared to Others Other top contenders like:

    - XGBoost (F1: 0.9174) — very close, but slightly lower recall
    - Random Forest without oversampling (F1: 0.9109) — good, but not handling class imbalance as well
    - XGBoost with oversampling (F1: 0.8911) — lower than the selected model

Model Improvement

GridSearchCV

We will now work on improving the model. We will use the GridSearchCV method to get the hyperparameters to see if we can further improve model performance.

Let us compare the performance metric:

Before GridSearchCV (Best Model - Random Forest with Oversampled Data):

- Accuracy: 0.973357
- Recall: 0.887719
- Precision: 0.951128
- F1 Score: 0.918330

After GridSearchCV (Tuned Random Forest with Oversampled Data):

- Accuracy: 0.972173
- Recall: 0.859649
- Precision: 0.972222
- F1 Score: 0.912477

Inferences

The GridSearchCV-tuned model maintained strong overall performance with a **very high precision (0.9722)**, slightly higher than the original. However, it comes with a **drop in recall (from 0.8877 to 0.8596)**, indicating that the model is now slightly less sensitive in identifying the positive class. As a result, the **F1 score slightly decreased from 0.9183 to 0.9125**.

This means:

- The tuned model is a bit more **conservative** in predicting positives (fewer false positives), but might **miss a few more actual positives** (slightly higher false negatives).
- The **original model (before tuning)** still holds a **slight edge in balance**, especially if **F1 score** is your primary metric.

Feature Importance

We have conducted a feature importance and below are the top 10 features:

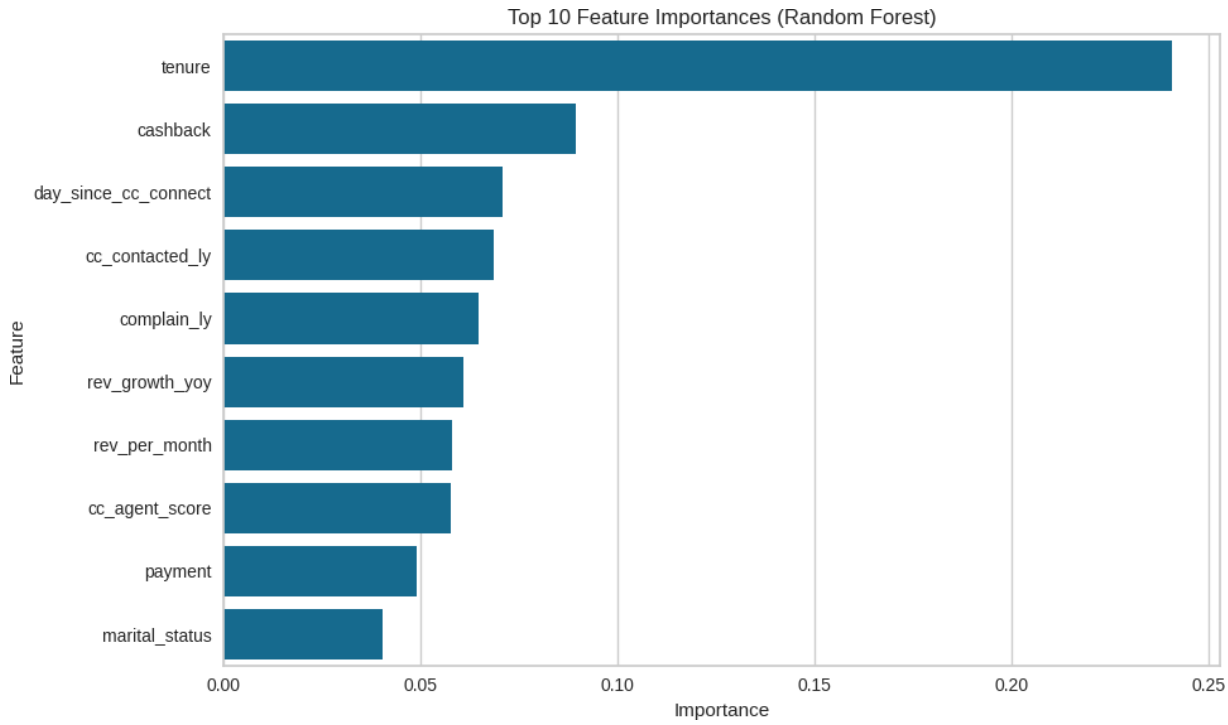

Figure 39 Top 10 features

Let us compare the model performance metric:

Before Feature Selection (Full Features – GridSearchCV Tuned Random Forest):

Accuracy	0.9722
Precision	0.9722
Recall	0.8596
F1 Score	0.9125

After Feature Selection (Top 10 Features):

Accuracy	0.9571
Precision	0.8706
Recall	0.8752
F1 Score	0.8729

Inferences

After reducing to the top 10 features, there is a slight drop in overall performance:

Accuracy dropped from 0.9722 → 0.9571

F1 Score dropped from 0.9125 → 0.8729

Precision dropped from 0.9722 → 0.8706

However, Recall slightly increased from 0.8596 → 0.8752, which means the model is now catching more true positives.

The performance trade-off is reasonable, especially given the reduced complexity:

- The model is now faster, lighter, and easier to interpret.
- Useful for real-time scenarios or deployment on resource-constrained environments.
- If interpretability, speed, or overfitting control is a concern, the top-feature model is a great alternative.

However, for maximum performance, especially in use cases where Precision is critical (like fraud or risk detection), the full-feature model still performs better.

Implications of adoption of model

High Accuracy and Balanced Performance
The model achieved a 97.3% accuracy and a high F1 Score of 91.8%, indicating it's both precise and reliable at predicting outcomes such as customer risk or churn.

Effective Handling of Class Imbalance
Oversampling helped address data imbalance, ensuring the model doesn't ignore the minority class (often the riskier or most valuable segment).

Better Customer Segmentation
Helps accurately identify different risk profiles or engagement levels among customers, enabling more tailored and effective marketing or retention strategies.

Early Warning System for High-Risk Customers
With high recall (88.8%), the model can correctly identify most of the customers at risk, allowing the business to proactively intervene before revenue loss.

Scalable & Robust for Real-World Application
Random Forest is a stable and scalable algorithm, making it suitable for deployment in production environments with minimal risk of performance issues.

Recommendations

For Cluster 0: Retain & Reward

- Launch exclusive loyalty programs (early access, personalized perks).
- Provide value-added services to enhance retention.
- Maintain service excellence — they are sensitive to experience.

For Cluster 1: Nurture & Upsell

- Run targeted campaigns to upgrade them to premium plans.
- Offer small but smart incentives (e.g., limited-time bonuses, bundled offers).
- Educate on features/benefits to increase engagement.

For Cluster 2: Rescue & Rebuild

- Trigger churn prevention workflows for accounts showing complaints or low tenure which means once we identify a flagged customer then the business has to either contact the customer personally or offer incentives for logging in and spending with the business.
- Improve onboarding experience and customer support.
- Offer limited-time retention offers (e.g., service recovery discounts).
- Monitor service/agent score feedback closely.