CAPSTONE PROJECT FINAL REPORT

DSBA

Contents-

List of Figure-

1. Introduction - Business problem definition

1.1 Problem Statement-

An E Commerce company or DTH (you can choose either of these two domains) provider is facing a lot of competition in the current market and it has become a challenge to retain the existing customers in the current situation. Hence, the company wants to develop a model through which they can do churn prediction of the accounts and provide segmented offers to the potential churners. In this company, account churn is a major thing because 1 account can have multiple customers. hence by losing one account the company might be losing more than one customer. You have been assigned to develop a churn prediction model for this company and provide business recommendations on the campaign. Your campaign suggestion should be unique and be very clear on the campaign offer because your recommendation will go through the revenue assurance team. If they find that you are giving a lot of free (or subsidized) stuff thereby making a loss to the company; they are not going to approve your recommendation. Hence be very careful while providing campaign recommendations.

1.2 Need for the Project-

A DTH provider's biggest cost is the cost of acquisition of a new customer. A customer thus acquired, will need to be retained for quite a few years so that the initial cost of acquisition is recovered back and that particular account is profitable1. Due to this reason, customer churn directly impacts the profitability of a DTH operator. DTH providers also are in a constant pressure to increase their customer base to maintain their profitability as most of them have a fixed broadcaster/content provider fee irrespective of the number of customers in their customer base.

As customer churn directly impacts both the top-line and bottom-line revenue of the business, existing customer base needs to be protected. Providing all customers with offers to retain them would make a dent in the profitability and hence it is very important to focus only on a select set of customers who are at a higher risk of churning.

1.3 Understanding business/social opportunity-

This is a case study of a DTH company where they have customers assigned with unique account ID and a single account ID can hold many customers (like family plan) across gender and marital status, customers get flexibility in terms of mode of payment they want to opt for. Customers are again segmented across various

types of plans they opt for as per their usage which is also based on the device they use (computer or mobile) moreover they earn cashbacks on bill payment. What are the parameters that play a vital role in having customers' loyalty and making them stay? All these social responsibilities will decide the best player in the market.

Data Dictionary-

- AccountID - account unique identifier
- Churn - account churn flag (Target Variable)
- 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 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_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

2. Data Report-

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

- Data has been collected for a random 11,260 unique account ID, across gender and marital status.
- Looking at variables "CC_Contacted_L12m", "rev_per_month", "Complain_l12m", "rev_growth_yoy", "coupon_used_l12m", "Day_Since_CC_connect" and "cashback_l12m"we can conclude that the data has been collected for last 12 month.
- Data has 19 variables, 18 independent and 1 dependent or the target variable, which shows if the customer churned or not.
- The data is the combination of services customers are using along with their payment option and also basic individual details as well.
- Data is a mixture of categorical as well as continuous variables.

2.2 Visual Inspection of data(row, columns,descriptive details)-

- Dataset has 11260 rows and 19 variables.

```
<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
```

- There are **5 columns of float type, 2 columns of integer type and 12 columns of object type.**
- There are **several columns that are supposed to be read as numeric,** instead they have been read as object type for e.g., Tenure is a numeric field but has been read as object. Those columns need to be checked for

special characters and need to be treated before the column can be changed to numeric for further processing.

- There are no duplicate rows in the data set. Each account id has one unique row.
- Several columns **have null values.**
- The following table shows the number of rows containing nulls and special characters that require data cleaning. **All special characters present in the data set were treated with nulls** so that they can be imputed. Some columns such as Gender and account_segment contained multiple values to represent the same category for e.g., 'M', 'Male'. Cleaning up of those values was also performed.

Column	Values Count	% Rows With values present	Number Of Nulls	% Rows With Nulls	Data cleanup Needed?	% Rows needing data cleaning
Account ID	11260	100.00%	0	0%	None	0.00%
Churn	11260	100.00%	0	0%	None	0.00%
Tenure	11158	99.09%	102	0.91%	Yes-#	1.03%
City_Tier	11148	99.01%	112	0.99%	None	0.00%
CC_Contacted_ly	11158	99.09%	102	0.91%	None	0.00%
Payment	11151	99.03%	109	0.97%	None	0.00%
Gender	11152	99.04%	108	0.96%	Yes-M,F	5.74%
Service_Score	11162	99.13%	98	0.87%	None	0.00%
Account_user_count	11148	99.01%	112	0.99%	Yes-@	2.95%
account_segment	11163	99.14%	97	0.86%	Yes-Regular + Super +	2.74%
CC_Agent_Score	11144	98.97%	116	1.03%	None	0.00%
Marital_Status	11048	98.12%%	212	1.88%	None	0.00%
rev_per_month	11158	99.09%	102	0.91%	Yes- +	6.12%

Complain_ly	10903	96.83%	357	3.17%	None	0.00%
rev_growth_yoy	11260	100.00%	0	0.00%	Yes - $	0.03%
Coupon_used_for_ payment	11260	100.00%	0	0.00%	Yes-$,*,#	0.03%
Day_Since_CC_conne ct	10903	96.83%	357	3.17%	Yes - $	0.01%
Cashback	10789	95.82%	471	4.18%	Yes - $	0.02%
Login_device	11039	98.04%	221	1.96%	Yes-&&&	4.79%

2.3 Understanding the attributes -

This project has 18 attributes contributing towards the target variable. Let's discuss these variables one after another.

1. AccountID – This variable represents a unique ID which represents a unique customer. This is of Integer data type and there are no null values present in this.

2. Churn – This is our target variable, which represents if the customer has churned or not. This is categorical in nature and will no null values. "0" represents "NO" and "1"represents "YES".

3. Tenure – This represents the total tenure of the account since opened. This is a continuous variable with 102 null values.

4. City_Tier – These variable segregates customer into 3 parts based on city the primary customer resides. This variable is categorical in nature and has 112 null values.

5. CC_Contacted_L12m – This variable represents the number of times all the customers of the account have contacted customer care in the last 12 months. This variable is continuous in nature and has 102 null values.

6. Payment – This variable represents the preferable mve 109 null values.

7. Gender – This variable represents the gender of the pride of bill payment opted by the customer. This is categorical in nature and haimary account holder. This is categorical in nature and 108 null values.

8. Service_Score – Scores provided by the customer basis the service provided by the company. This variable is categorical in nature and has 98 null values.

9. Account_user_count – This variable gives the number of customers attached with an accountID. This is continuous in nature and has 112 null values.

10. account_segment – These variables segregate customers into different segments based on their spend and revenue generation. This is categorical in nature and has 97 null values.

11. CC_Agent_Score — Scores provided by the customer basis the service provided by the customer care representative of the company. This variable is categorical in nature and has 116 null values.

12. Marital_Status – This represents the marital status of the primary account holder. This is categorical in nature and has 212 null values.

13. rev_per_month – This represents average revenue generated per account ID in the last 12 months. This variable is continuous in nature and has 102 null values.

14. Complain_l12m – This denotes if customers have raised any complaints in the last 12 months. This is categorical in nature and has 357 null values.

15. rev_growth_yoy – This variable shows revenue growth in percentage of account for 12 months Vs 24 to 13 months. This is continuous in nature and doesn't have any null values.

16. coupon_used_l12m – This represents the number of times customer's have used discount coupons for bill payment. This is continuous in nature and doesn't have any null values.

17. Day_Since_CC_connect – This represents the number of days since customers have contacted the customer care. Higher the number of days denotes better the service. This is continuous in nature and has 357 null values.

18. cashback_l12m – This variable represents the amount of cash back earned by the customer during bill payment. This is continuous in nature and has 471 null values.

19. Login_device – This variable represents in which device the customer is availing the services if it's on phone or on computer. This is categorical in nature and has 221 null values.

The following table shows a basic statistical description of the numeric columns after data clean-up. It contains a 5-point summary of the numeric fields – minimum, maximum, 25th percentile, 50th percentile and 75th percentile. In addition, it contains the count of values present in each column, mean and standard deviation.

	count	mean	std	min	25%	50%	75%	max
Churn	11260.0	0.168384	0.374223	0.0	0.00	0.00	0.00	1.0
Tenure	11042.0	11.025086	12.879782	0.0	2.00	9.00	16.00	99.0
City_Tier	11148.0	1.653929	0.915015	1.0	1.00	1.00	3.00	3.0
CC_Contacted_LY	11158.0	17.867091	8.853269	4.0	11.00	16.00	23.00	132.0
Service_Score	11162.0	2.902526	0.725584	0.0	2.00	3.00	3.00	5.0
Account_user_count	10816.0	3.692862	1.022976	1.0	3.00	4.00	4.00	6.0
CC_Agent_Score	11144.0	3.066493	1.379772	1.0	2.00	3.00	4.00	5.0
rev_per_month	10469.0	6.362594	11.909686	1.0	3.00	5.00	7.00	140.0
Complain_ly	10903.0	0.285334	0.451594	0.0	0.00	0.00	1.00	1.0
rev_growth_yoy	11257.0	16.193391	3.757721	4.0	13.00	15.00	19.00	28.0
coupon_used_for_payment	11257.0	1.790619	1.969551	0.0	1.00	1.00	2.00	16.0
Day_Since_CC_connect	10902.0	4.633187	3.697637	0.0	2.00	3.00	8.00	47.0
cashback	10787.0	196.236370	178.660514	0.0	147.21	165.25	200.01	1997.0

- Tenure seems to have a very huge range up to 99. It could be in months in which case those would be valid values.
- The maximum limit for many of the variables seems to be very far apart from the 75th percentile for many variables such as cashback, revenue per month and customer contacted last year. There seems to be significant positive skew in these variables. A look at the boxplot and histogram will confirm the presence of outliers.

3. Exploratory Data Analysis (EDA)-

3.1 Univariate Analysis-

It has been done by observing:

- Box plots and histograms for continuous variables.
- Count plots for categorical variables.

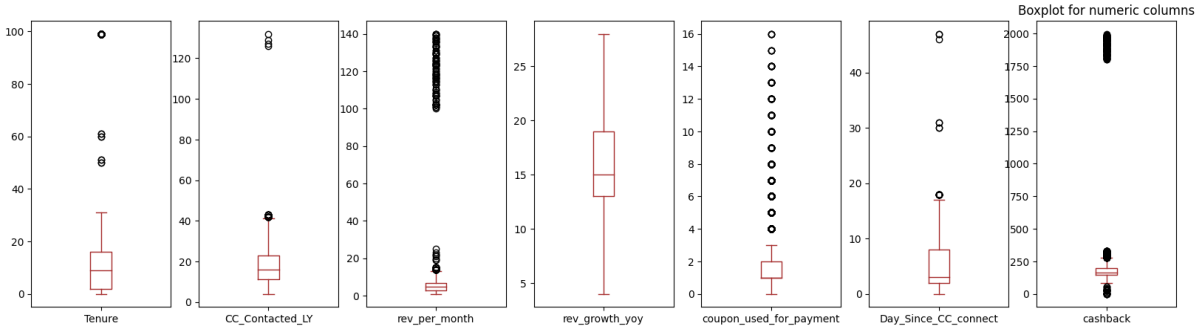
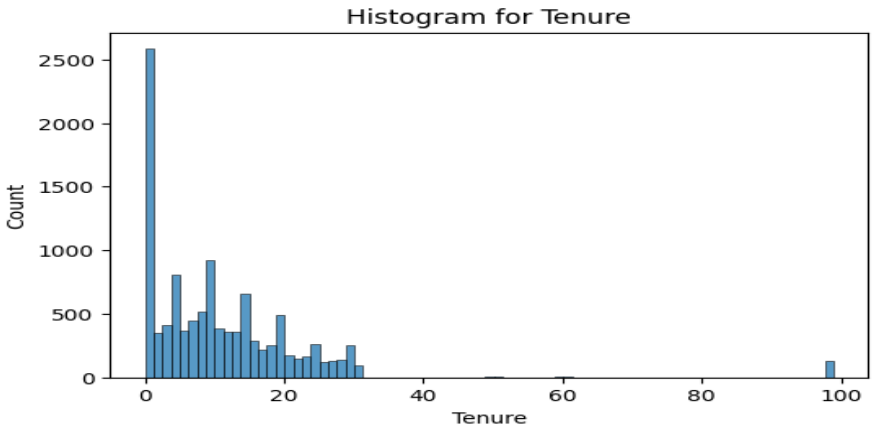
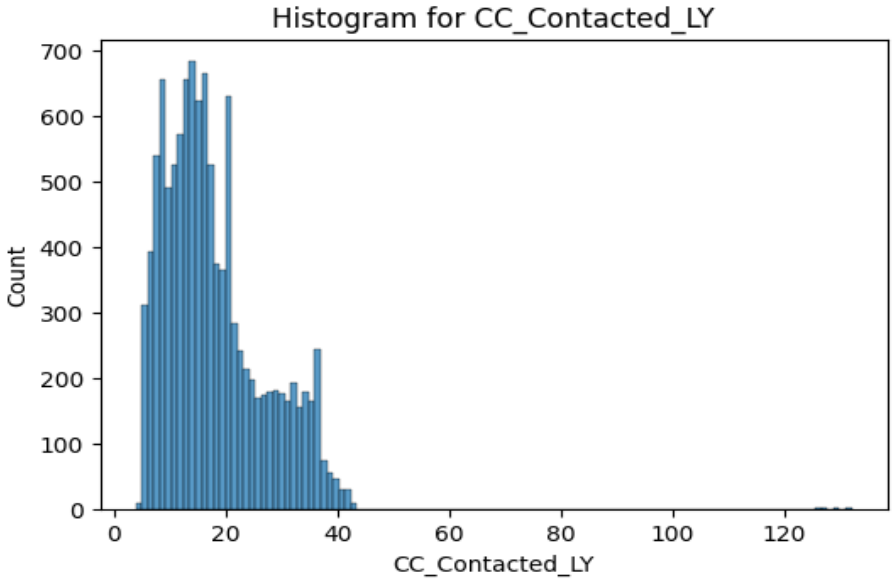
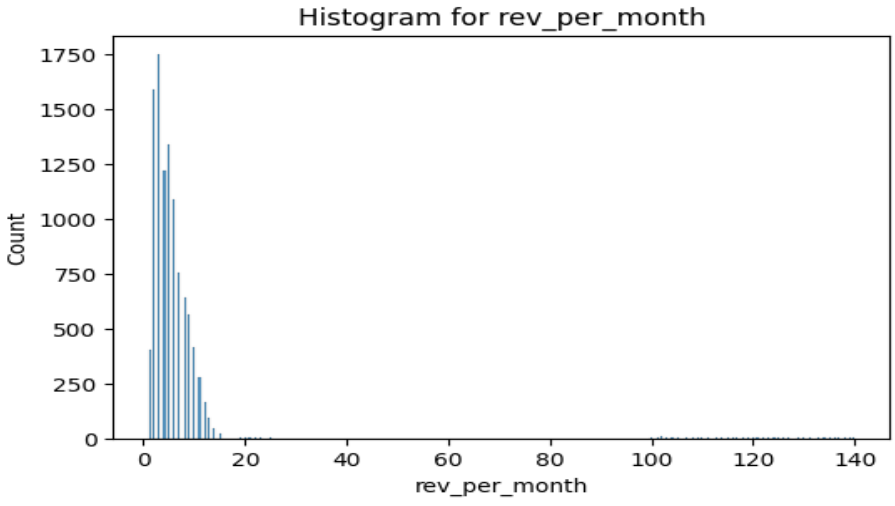


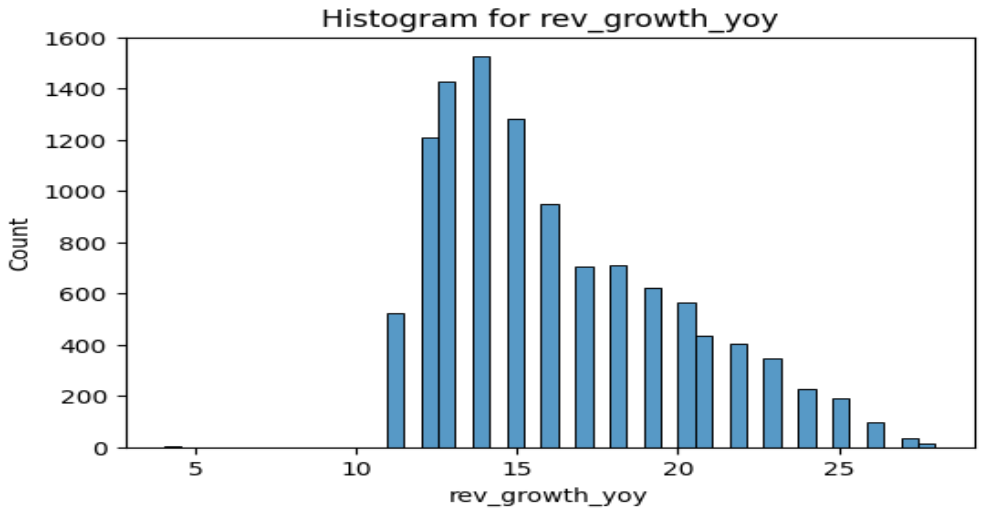
Fig 1.Boxplot for numeric variables

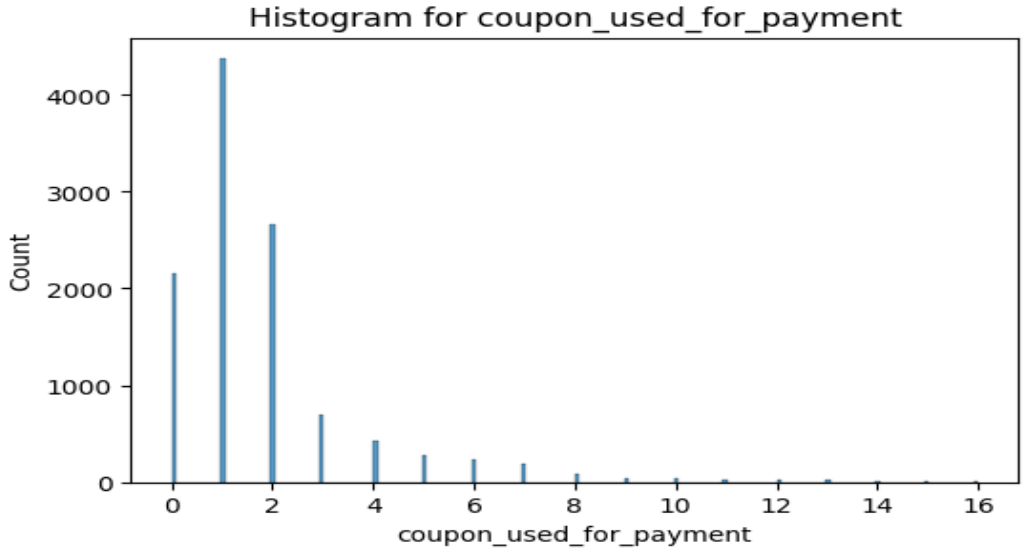
3.2 Histogram for Continuous Variables-

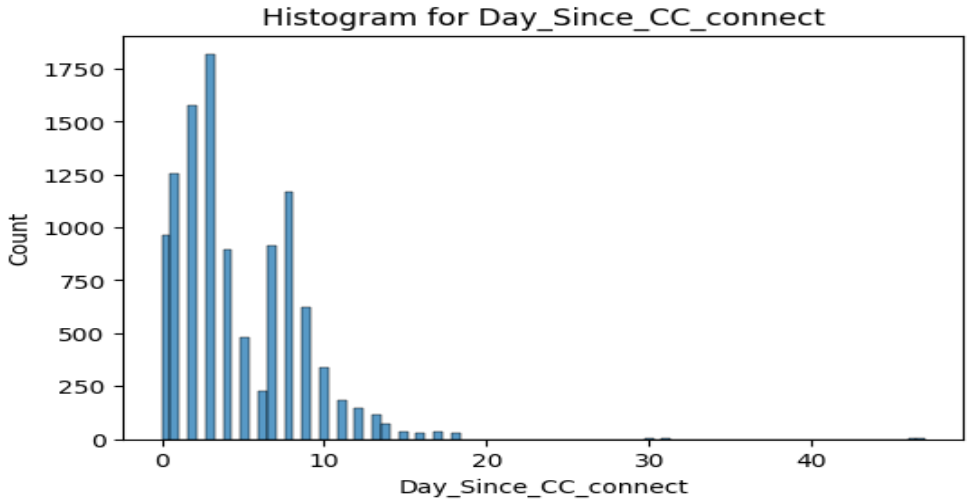

Histogram for Tenure
Count
Tenure
0
500
1000
1500
2000
2500
0
20
40
60
80
100


Histogram for CC_Contacted_LY
Count
CC_Contacted_LY
0
100
200
300
400
500
600
700
0
20
40
60
80
100
120


Histogram for rev_per_month
Count
rev_per_month
0
250
500
750
1000
1250
1500
1750
0
20
40
60
80
100
120
140


Histogram for rev_growth_yoy
Count
rev_growth_yoy
0
200
400
600
800
1000
1200
1400
1600
5
10
15
20
25


Histogram for coupon_used_for_payment
Count
coupon_used_for_payment
0
1000
2000
3000
4000
0
2
4
6
8
10
12
14
16


Histogram for Day_Since_CC_connect
Count
Day_Since_CC_connect
0
250
500
750
1000
1250
1500
1750
0
10
20
30
40

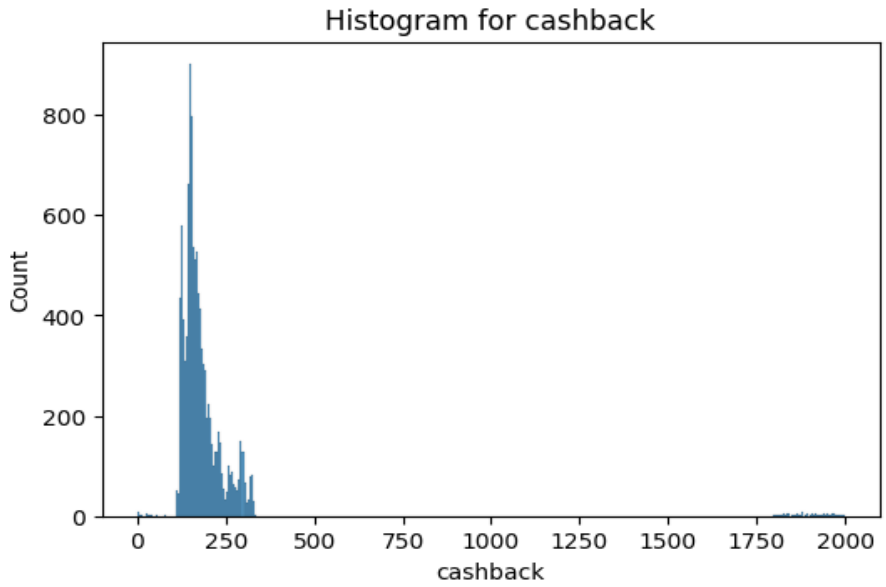


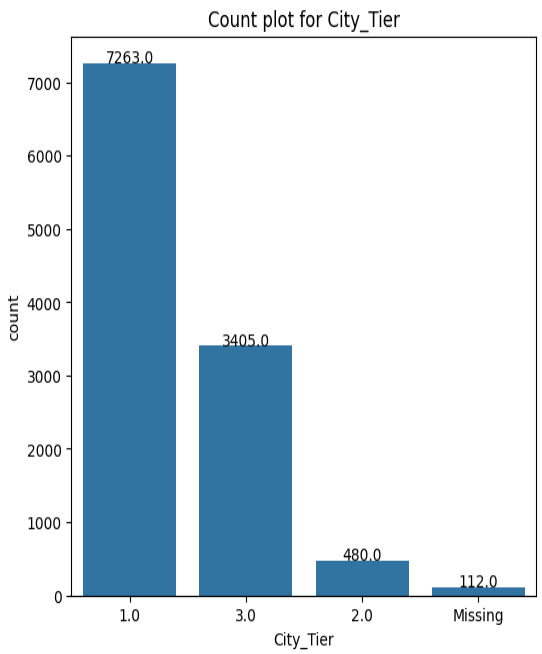
Fig.2-7 Histogram for Continuous Variables

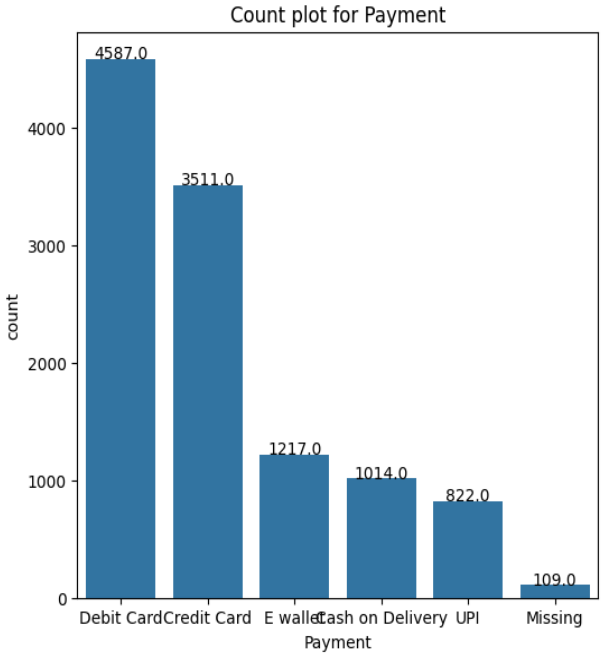
	Skewness
Tenure	3.90
CC_Contacted_LY	1.42
rev_per_month	9.09
rev_growth_yoy	0.75
coupon_used_for_payment	2.58
Day_Since_CC_connect	1.27
cashback	8.77

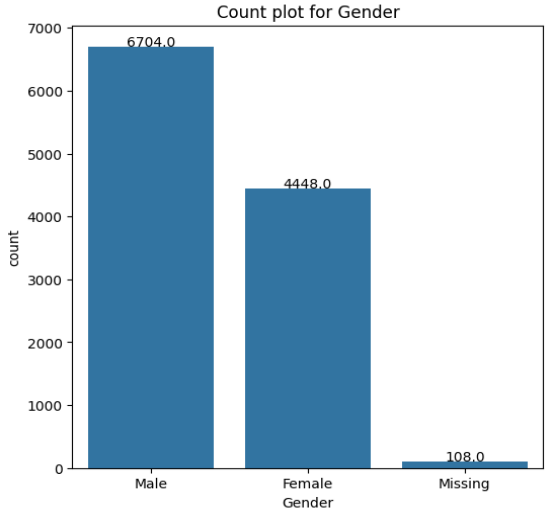
Observations-

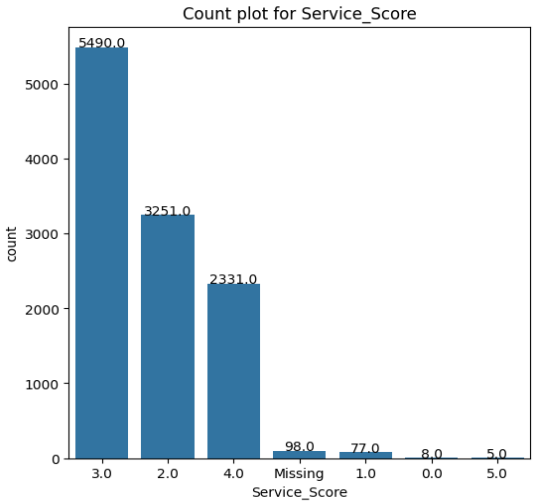
- All numeric variables with the exception of rev_growth_yoy have outliers. Some outliers for certain variables are closer to the whisker, whereas there are a group of outliers that are far beyond the whisker with no in between values.
- Coupon_used_for_payment has a very limited range 0 to 16. Hence, for the purpose of this analysis, the outliers will not be treated.
- All numeric variables with the exception of rev_growth_yoy have a high positive skew.

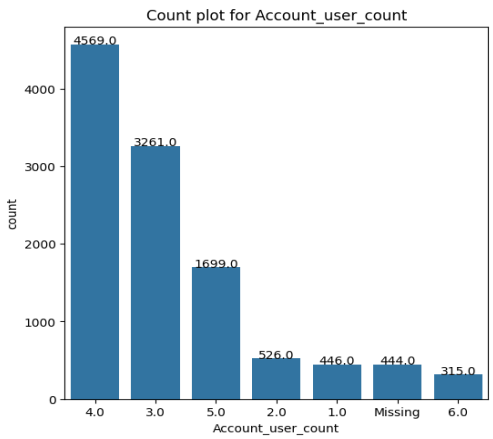
3.3 Count Plot for Categorical Variables-

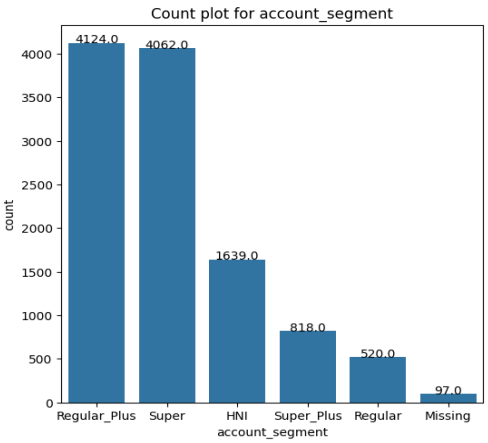

Count plot for City_Tier
count
City_Tier
1.0
3.0
2.0
Missing
7263.0
3405.0
480.0
112.0

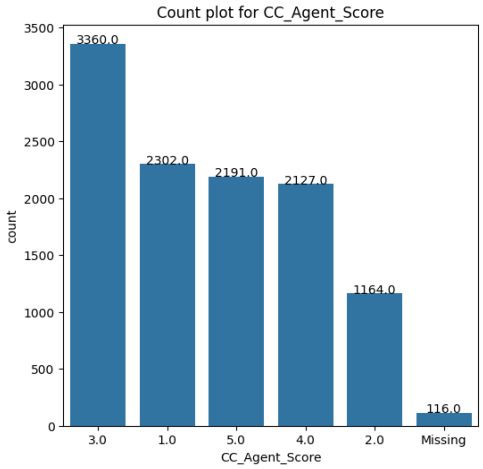

Count plot for Payment
count
Payment
Debit Card
Credit Card
E wallet
Cash on Delivery
UPI
Missing
4587.0
3511.0
1217.0
1014.0
822.0
109.0

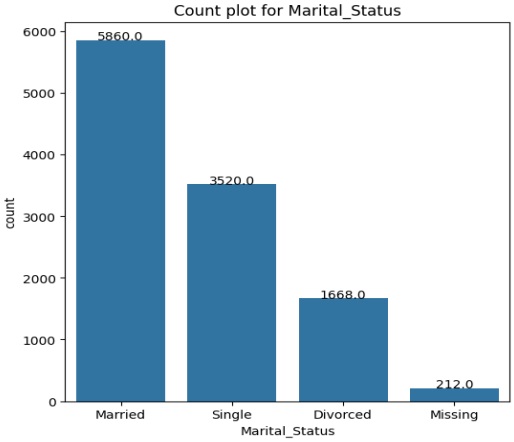

Count plot for Gender
count
Gender
Male
Female
Missing
6704.0
4448.0
108.0

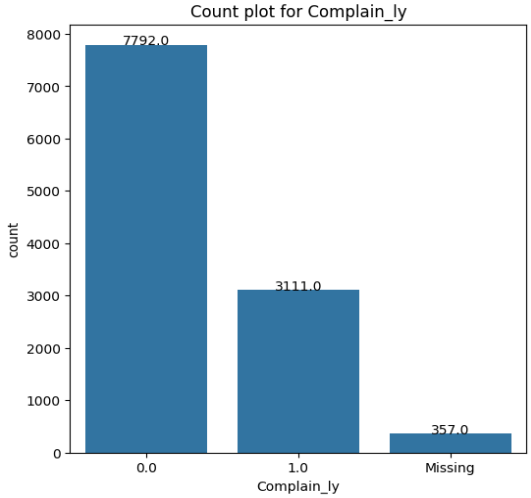

Count plot for Service_Score
count
Service_Score
3.0
2.0
4.0
Missing
1.0
0.0
5.0
5490.0
3251.0
2331.0
98.0
77.0
8.0
5.0

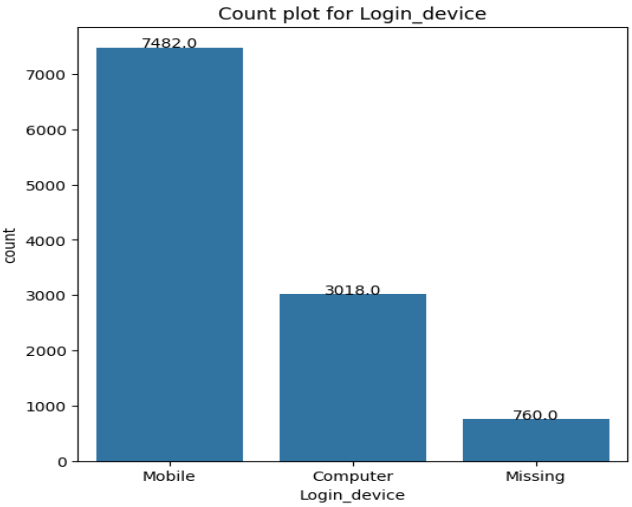

Count plot for Account_user_count
count
Account_user_count
4.0
3.0
5.0
2.0
1.0
Missing
6.0
4569.0
3261.0
1699.0
526.0
446.0
444.0
315.0


Count plot for account_segment
count
account_segment
Regular_Plus
Super
HNI
Super_Plus
Regular
Missing
4124.0
4062.0
1639.0
818.0
520.0
97.0


Count plot for CC_Agent_Score
count
CC_Agent_Score
3360.0
2302.0
2191.0
2127.0
1164.0
116.0
3.0
1.0
5.0
4.0
2.0
Missing


Count plot for Marital_Status
count
Marital_Status
5860.0
3520.0
1668.0
212.0
Married
Single
Divorced
Missing


Count plot for Complain_ly
count
Complain_ly
7792.0
3111.0
357.0
0.0
1.0
Missing


Count plot for Login_device
count
Login_device
7482.0
3018.0
760.0
Mobile
Computer
Missing

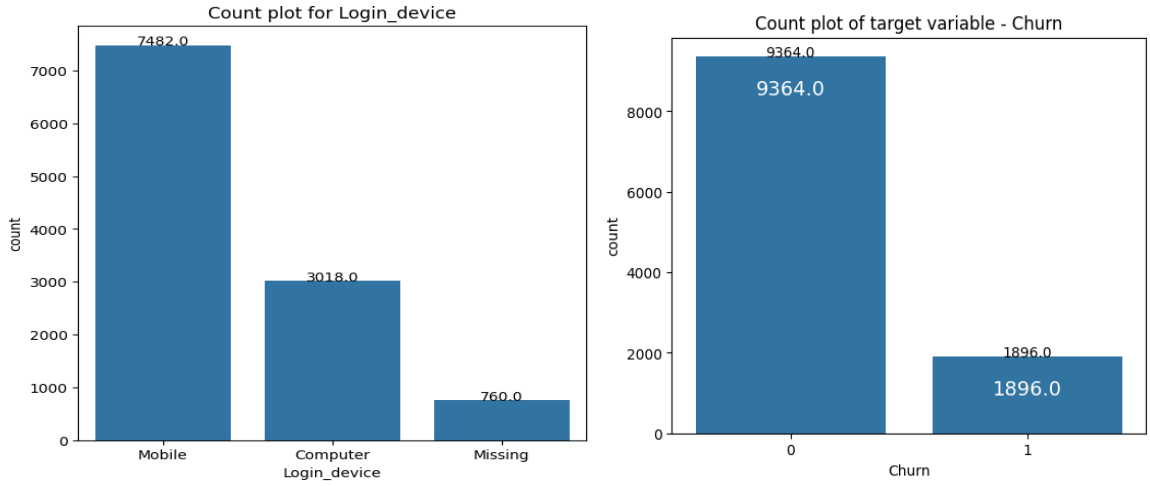


Fig 3-11 Count plot for Categorical Variable

Observations-

- This is an imbalanced dataset with target variable containing approx 16.8% churn.
- Tier 1 cities have more accounts followed by Tier 3 cities.
- Most of the accounts pay through debit card followed by credit card. UPI ranks last amongst payment methods.
- Number of male account holders outnumber females.
- Regular plus and Super are top two account segment types by number.
- Top score for both Customer service agent and Service score is 3.
- Married customers have the most accounts followed by single.
- Most accounts do not have a customer complaint filed last year.
- Most account holders use Mobile for logging and using services.

4.1 Bivariate Analysis and Multivariate Analysis-

Continuous predictor variables that show some ability to separate the target variables-

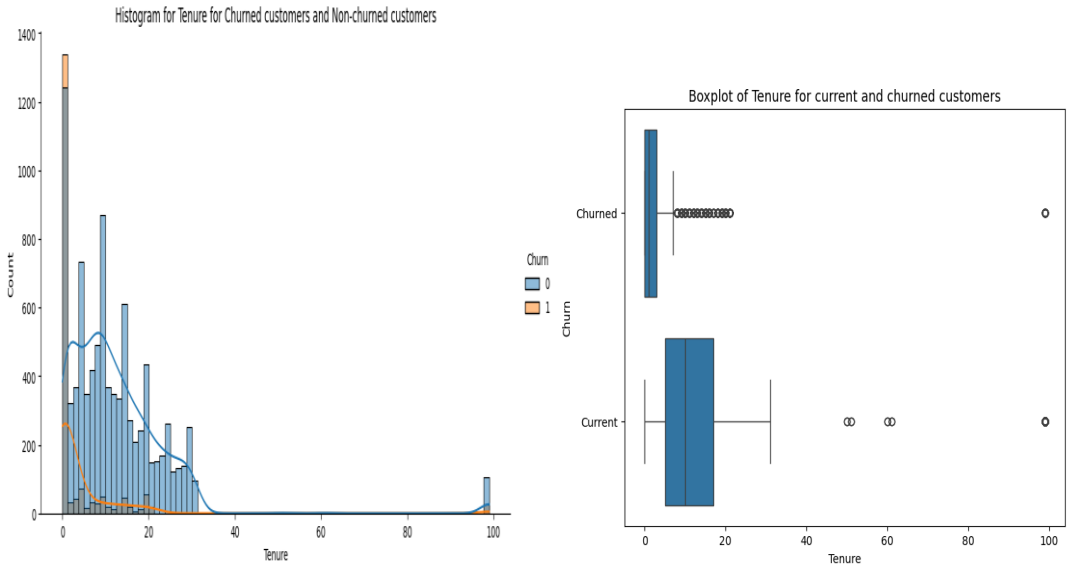


Fig 4.1 Tenure vs Churn

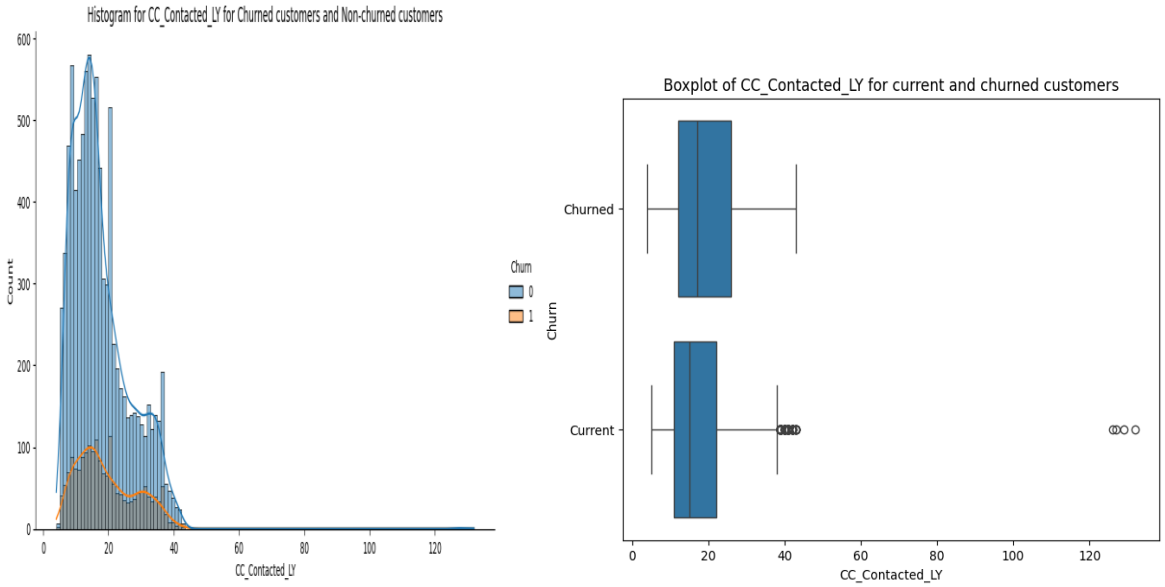


Fig 4.2 CC_Contacted_LY vs Churn

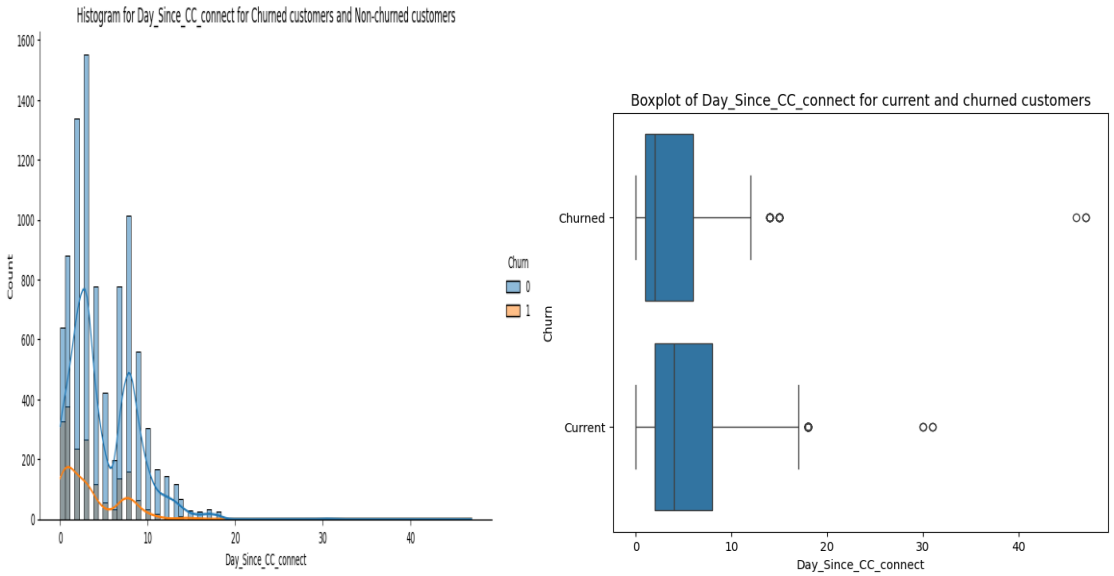


Fig 4.3 Day_Since_CC_Connect vs Churn

Continuous predictor variables that aren't able to show a clear separation between target classes-

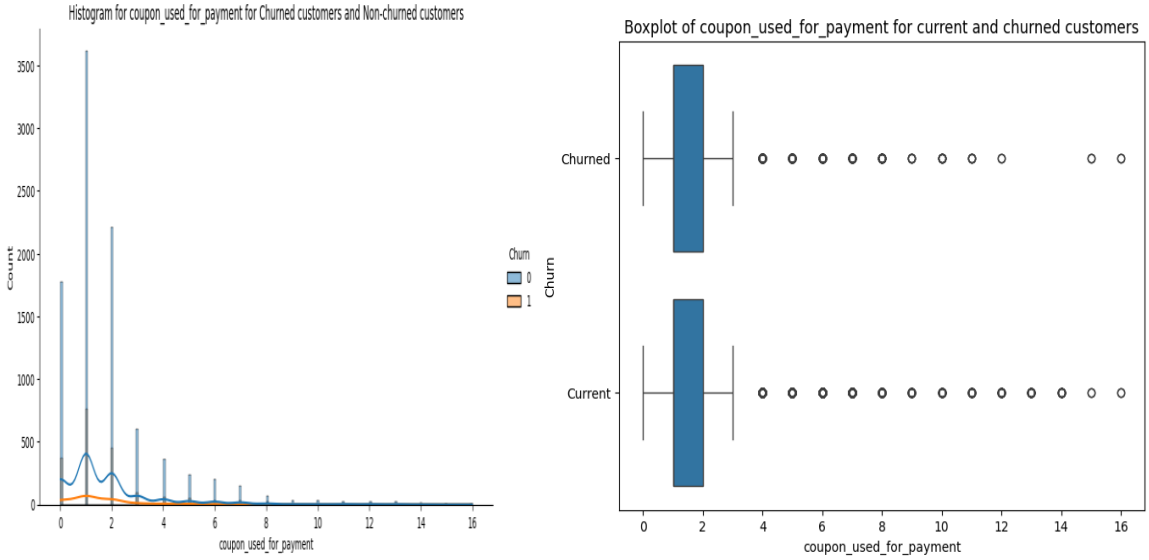


Fig 4.4 Coupon used for payment vs Churn

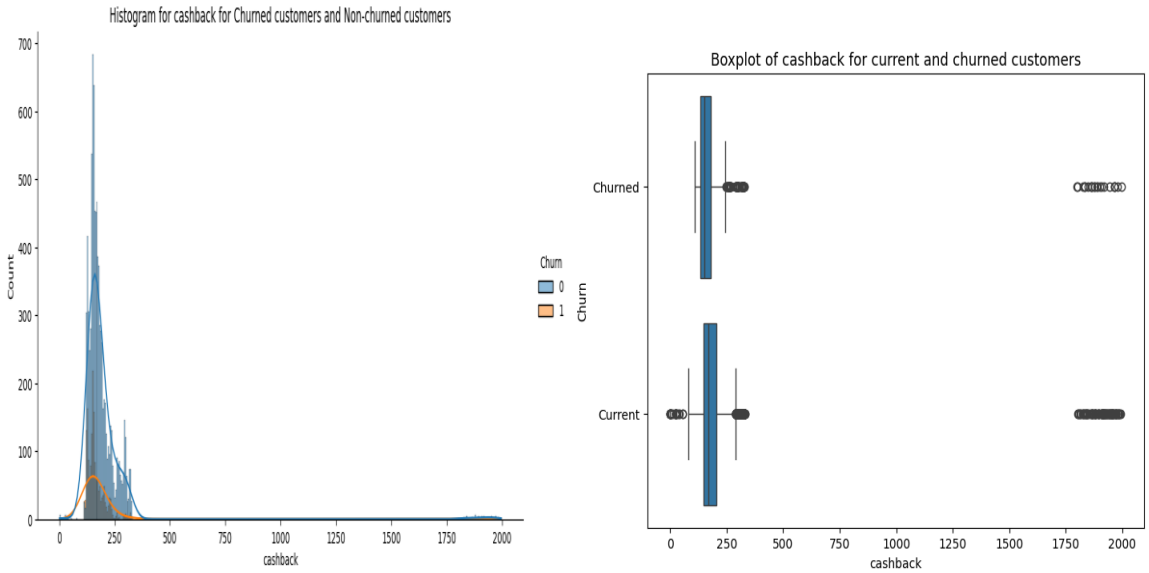


Fig 4.5 Cashback vs Churn

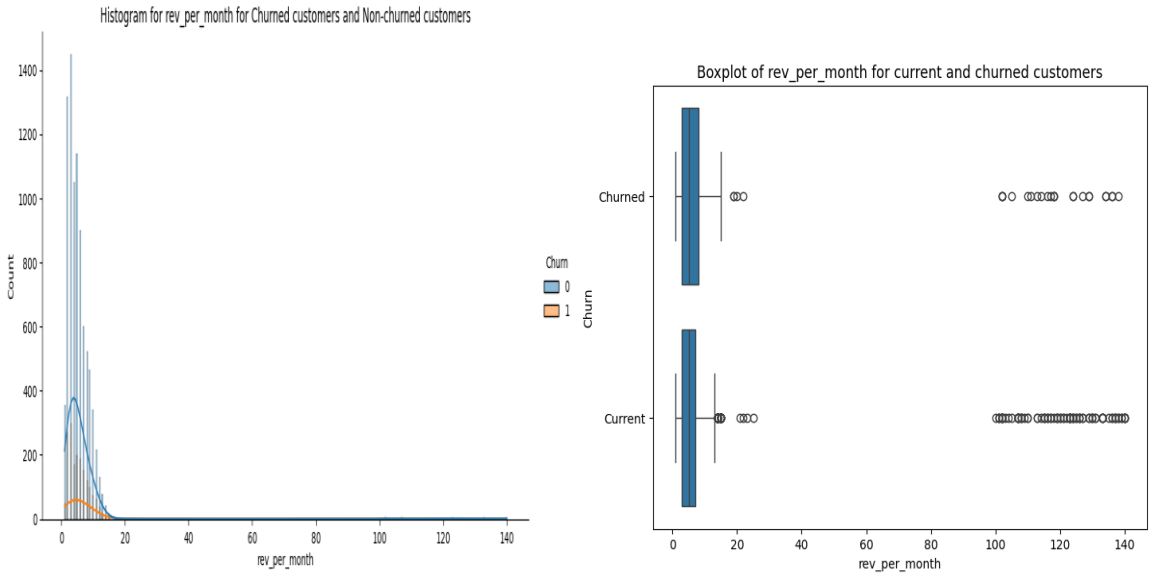


Fig 4.6 Revenue_per_month vs Churn

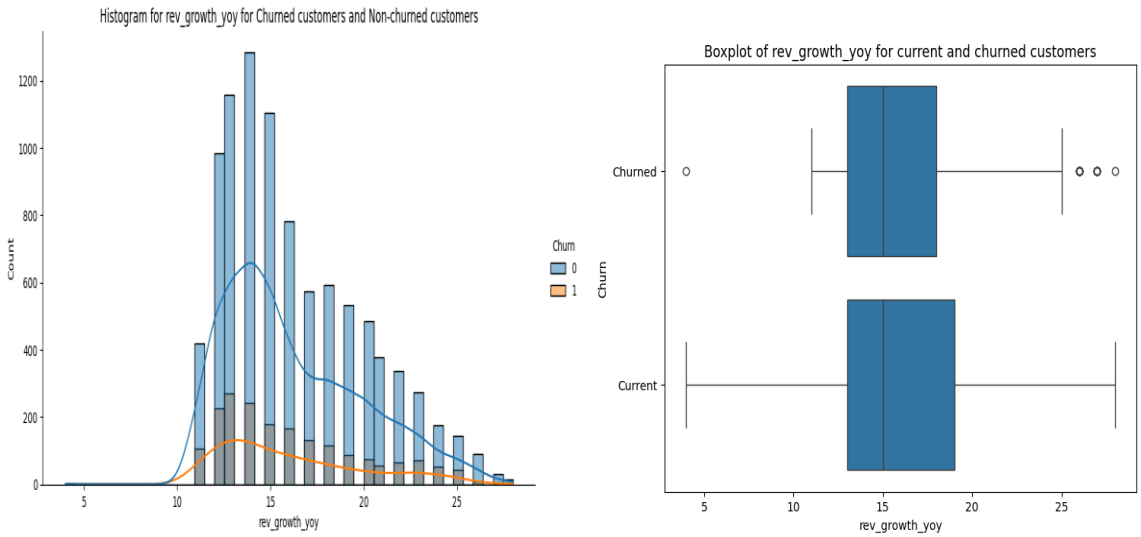


Fig 4.7 Revenue_growth_yoy vs Churn

Observations-

From the above plots we can see that variables such as Tenure, Days_since_CC_connect have some influence on the target variable. The median lines for Churned and Non-churned observations when plotted against these variables show a difference. Whereas, the medians in boxplots for variables such as coupon_used_for_payment, rev_growth_yoy do not show much difference in churned vs non-churned distributions.

4.2 Continuous Variables - ANOVA (Analysis of Variance)

H0: Means of all groups are equal

Ha: At least means of one pair of the groups is different

Results of Anova test for following variables and churn-

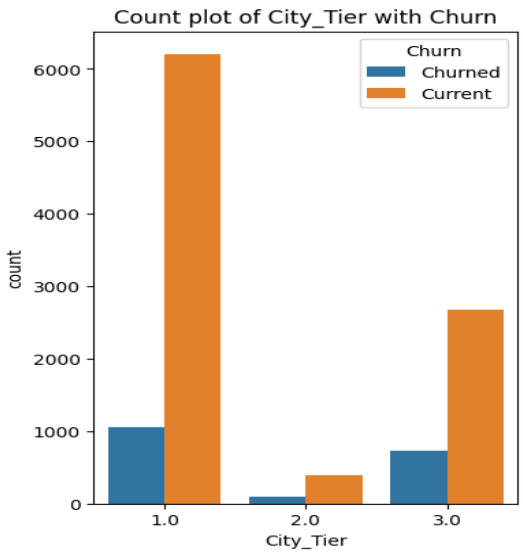
Variable	**F-Statistics**	**Probability of>F**	**Inference at significance level of 5%**
Tenure	634.6	3.3E-36	Reject null hypothesis. The means are different. Variable significant to model building.
CC_Contacted_ly	58.25	2.94E-14	Reject null hypothesis. The means are different.Variable significant to model building.
rev_per_month	5.32	0.021	Reject null hypothesis. The means are different.Variable significant to model building.

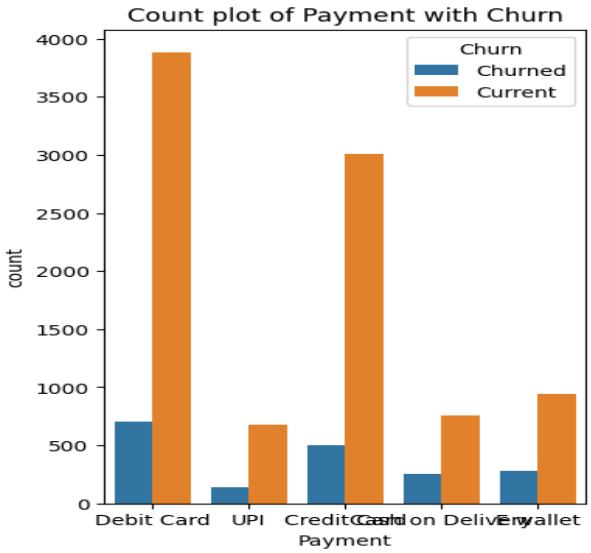
rev_growth_yoy	2.17	0.141	Cannot reject null hypothesis. The means are equal. Variable can be dropped.
Coupon_used_for_ payment	2.47	0.116	Cannot reject null hypothesis. The means are equal. Variable can be dropped.
Day_Since_CC_Connect	243.9	2.1E-54	Reject null hypothesis. The means are different.Variable significant to model building.
Cashback	11.32	0.001	Reject null hypothesis. The means are different.Variable significant to model building.

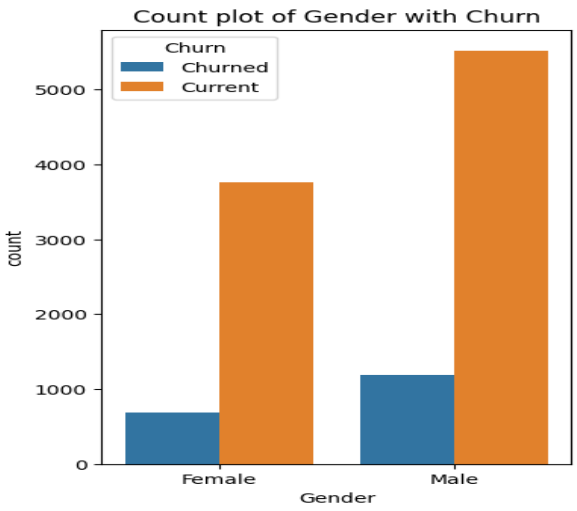
Observation-

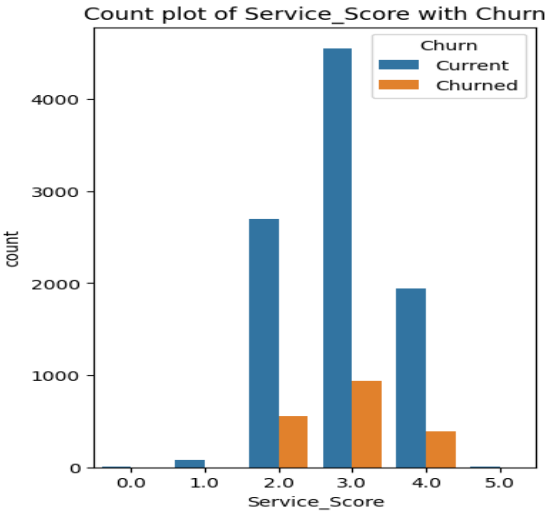
At a significance level of 0.05 (5%), the tests for the variables rev_growth_yoy and coupon_used_for_payment have given a p-value of greater than 0.05. In these two cases, Ho cannot be rejected, i.e., the means for the two groups churn=0 and churn=1 for these variables are the same. This implies that since the groups are not too different, these two variables cannot be significant predictors of the target variable.

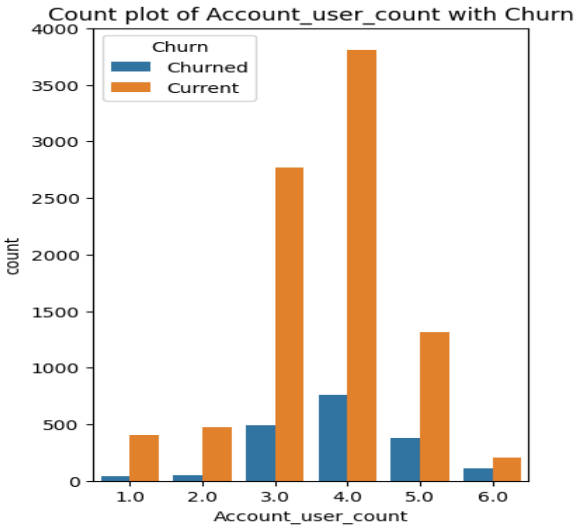
4.3 Bivariate plots for Categorical variables vs Churn-

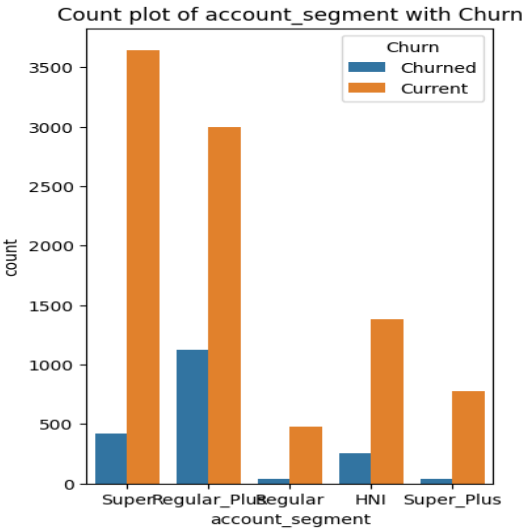


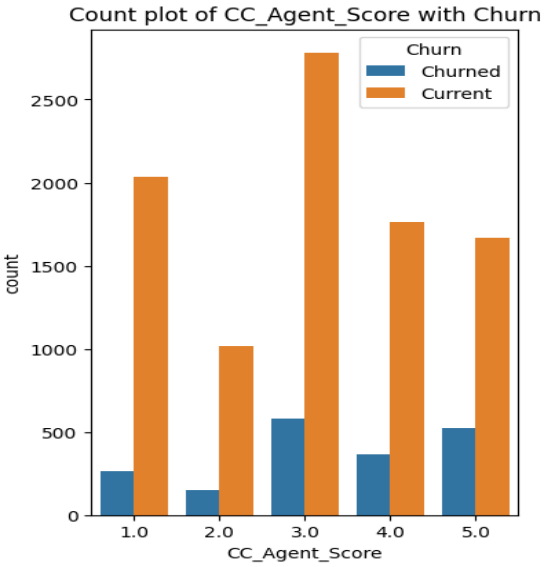


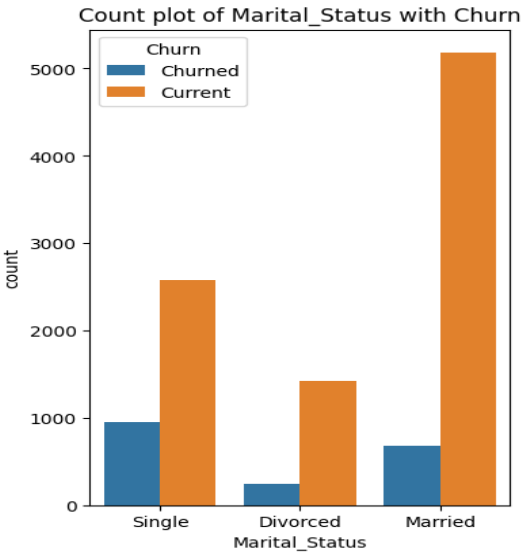

Count plot of Gender with Churn
Churn
Churned
Current
count
Female
Male
Gender


Count plot of Service_Score with Churn
Churn
Current
Churned
count
Service_Score


Count plot of Account_user_count with Churn
Churn
Churned
Current
count
Account_user_count


Count plot of account_segment with Churn
Churn
Churned
Current
count
account_segment


Count plot of CC_Agent_Score with Churn
Churn
Churned
Current
count
CC_Agent_Score


Count plot of Marital_Status with Churn
Churn
Churned
Current
count
Single
Divorced
Married
Marital_Status

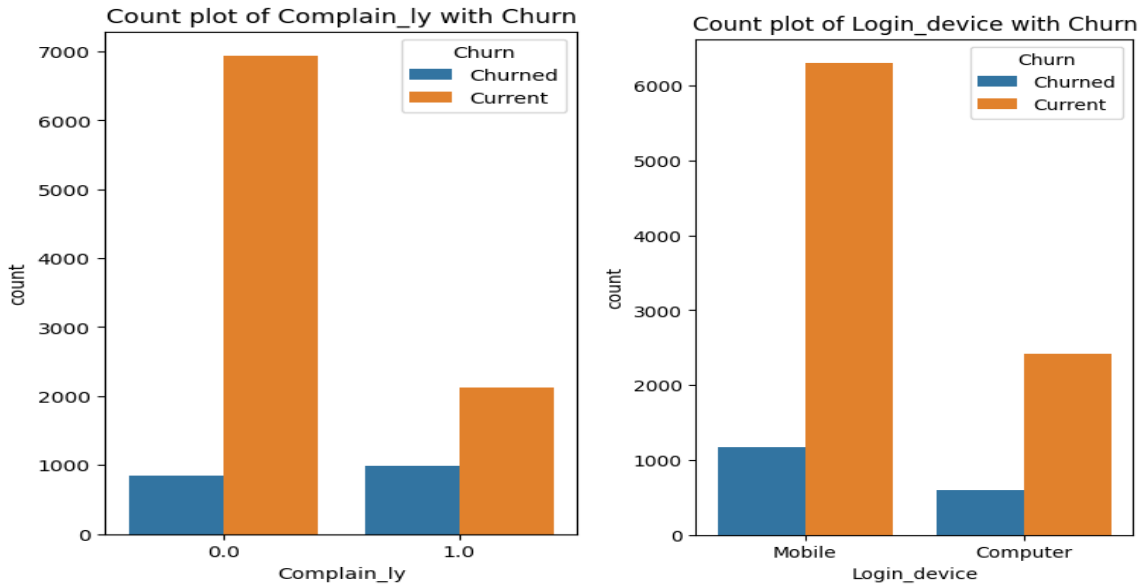


Fig 5-10 Stacked bar chart for categorical variables

Observations-

- In the above stacked bar charts, active customers are called current customers. The first bar shows distribution of the categorical predictor variable being analysed within the churned customer.
- The second bar shows distribution within active/current customers.
- From the above charts, some of the variables like city_tier, account_segment, Complain_ly, Marital_Status, CC_Agent_score seem to show a difference in distribution when churned vs current customers are considered.
- Other variables such as Gender and Service_Score have more or less similar distributions within Churned and Current/active customer bars.

4.4 Categorical variables: Chi-squared test of independence at significance level 0.05 -

Null Hypothesis: There is no relationship between two categorical variables.

Alternate Hypothesis: There is a relationship between two categorical variables.

	Variable	chi2	p-value	chi2_output
0	Gender	8.983146	2.724812e-03	Reject Ho; Dependent.
1	Service_Score	18.414690	2.469166e-03	Reject Ho; Dependent.
2	City_Tier	80.288817	3.677095e-18	Reject Ho; Dependent.
3	Payment	103.799617	1.526348e-21	Reject Ho; Dependent.
4	Account_user_count	154.959445	1.173574e-31	Reject Ho; Dependent.
5	account_segment	567.068402	2.073937e-121	Reject Ho; Dependent.
6	CC_Agent_Score	139.031565	4.549521e-29	Reject Ho; Dependent.
7	Marital_Status	379.808123	3.355165e-83	Reject Ho; Dependent.
8	Complain_ly	688.084739	1.166239e-151	Reject Ho; Dependent.

A p-value less than 0.05 (typically ≤ 0.05) is statistically significant. It indicates strong evidence against the null hypothesis, as there is less than a 5% probability the null is correct. Since the p-value returned for all the categorical variables is less than 0.05, the null hypothesis can be rejected. Hence at 5% level of significance, **it may be concluded that churn is not independent of these categorical variables.** Hence, **we will proceed to retain all these categorical predictor variables at this point.**

4.5 Pair plot for the numeric variables with hue set as target variable-

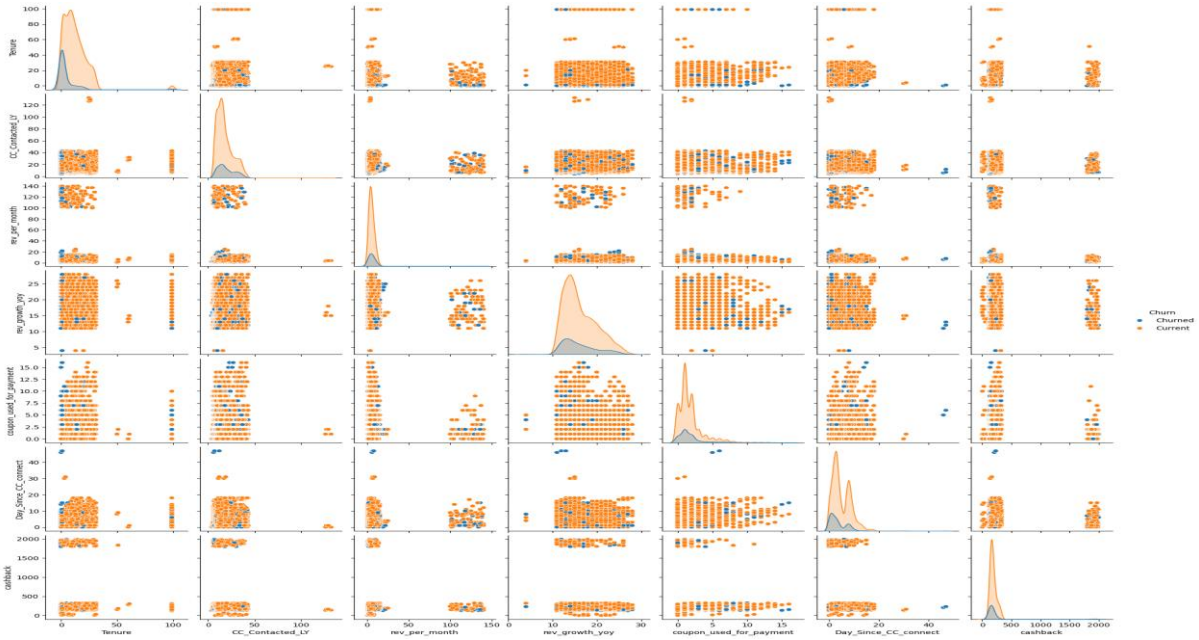


Fig 6. Pair plot for numeric predictor variables

Observations-

- Some of the variables clearly show presence of some clusters for e.g., rev_per_month and Day_Since_CC_connect.
- The diagonal kde plot for Tenure shows a slight separation with customers who churned falling on the lower side of Tenure.
- There is no linear relationship between any two continuous variables.

4.6 Correlation Heatmap-

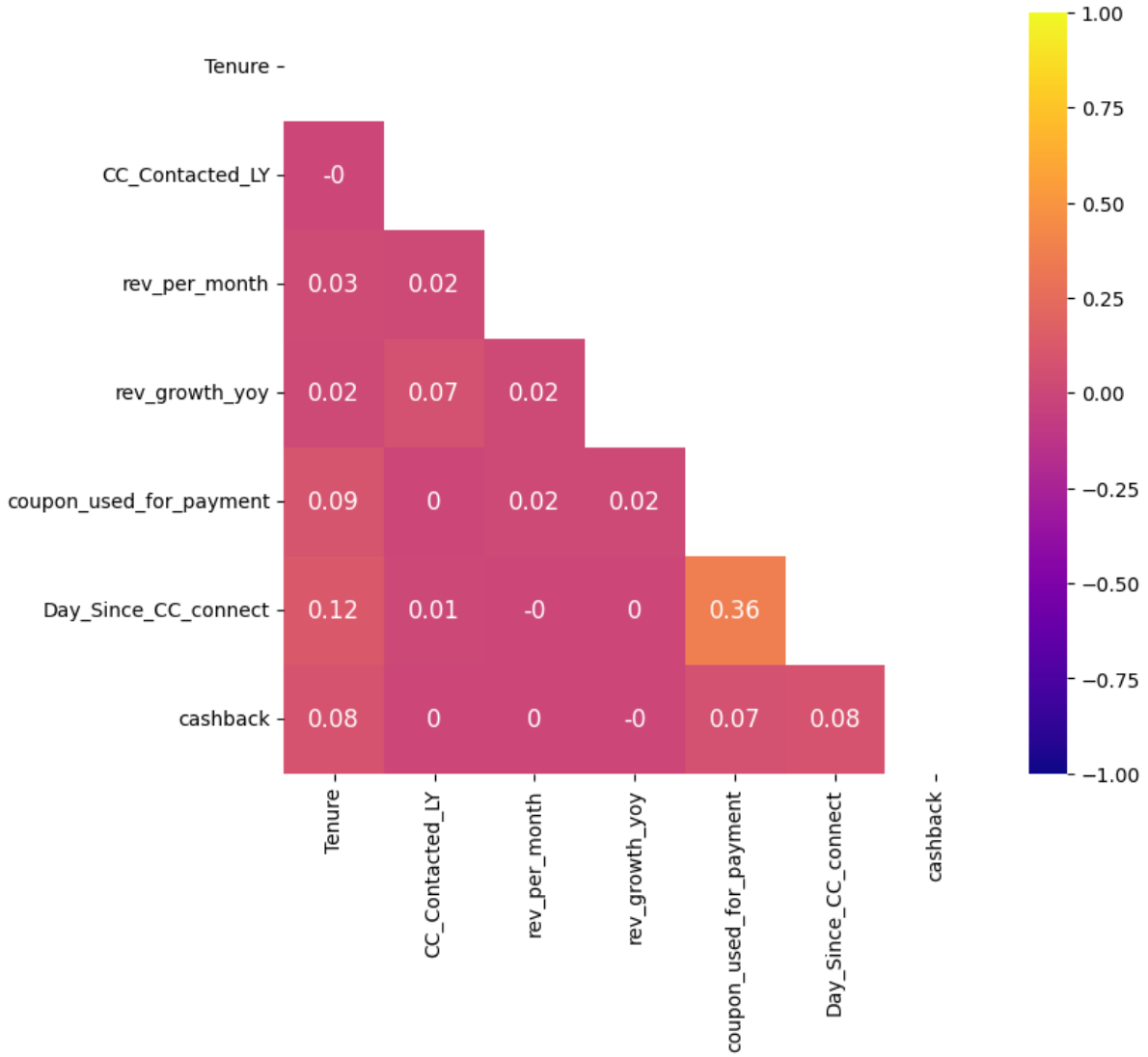


Fig 7. Correlation Heatmap

Observation- None of the numeric variables show a strong correlation. Hence there is no need to drop any variable.

5. Data cleaning and preprocessing-

5.1 Removal of unwanted variables-

- After in-depth understanding of data we conclude that removal of variables is not required at this stage of the project. We can remove the

variable “AccountID” which denotes a unique ID assigned to unique customers.

- Any variable that remains constant for all or most of the observations as this does not add any strength to prediction. As observed from the histogram and count plots/value counts (categorical), there are no variables that have constant value for all observations.
- Any predictor variable that has a very weak correlation with the target variable. As seen from the Chi square test and Anova test in the bivariate analysis.

5.2 Missing Value treatment-

Percentage of nulls-

Percentage nulls or missing values present in the predictor variables of the dataset are as follows:

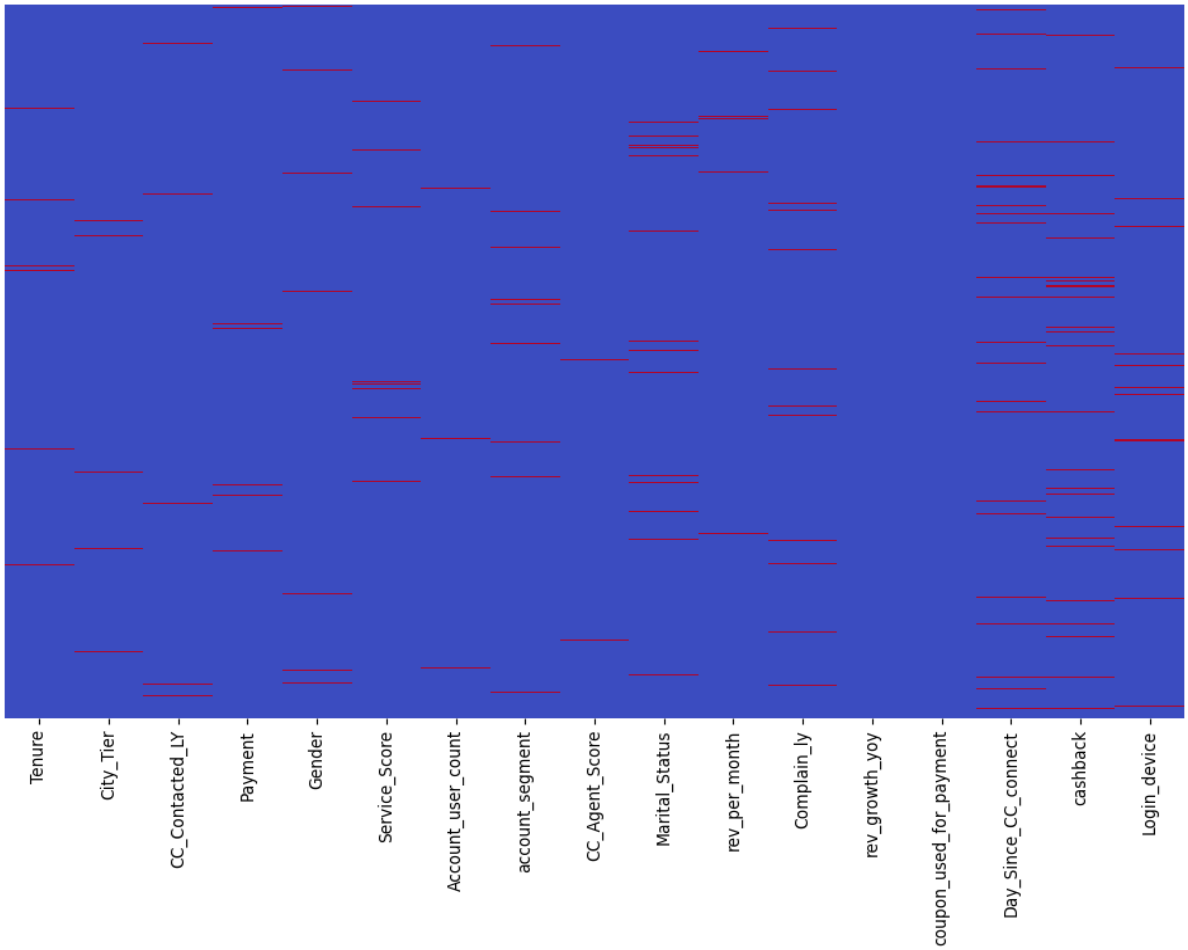


Fig 8. Visualization of nulls

```
cashback                  4.18
Day_Since_CC_connect      3.17
Complain_ly               3.17
Login_device              1.96
Marital_Status            1.88
CC_Agent_Score            1.03
Account_user_count        0.99
City_Tier                 0.99
Payment                   0.97
Gender                    0.96
rev_per_month             0.91
CC_Contacted_LY           0.91
Tenure                    0.91
Service_Score             0.87
account_segment           0.86
rev_growth_yoy            0.00
coupon_used_for_payment   0.00
dtype: float64
```

Missing Value Treatment-

Missing value treatment was done using KNN imputation, a distance-based method. The following treatments were done as they are pre-requisites for missing value treatment using KNN.

- All variables need to be numeric. Any object-type categorical variables need to be encoded suitably (label/ one-hot encoding).
- One hot encoded variables 'Payment','Gender','account_segment', 'Marital_Status','Login_device'.
- All variables need to be scaled as KNN is a distance-based algorithm. Scaling was done using Standard Scaler function from SKLearn library for the predictor variables.
- Null imputing was done using Sklearn's KNNImputer function. This algorithm imputed missing values using K-nearest neighbors.

Nulls in predictor variables after KNN Imputing-

```
Tenure                    0
City_Tier                 0
CC_Contacted_LY           0
Service_Score             0
User_Count                0
CC_Score                  0
Rev_Permonth              0
Complain_LY               0
Days_Since_CC             0
Cashback                  0
Payment_Creditcard        0
Payment_Debitcard         0
Payment_Ewallet           0
Payment_UPI               0
Gender_Male               0
ACSegment_Regular         0
ACSegment_Regularplus     0
ACSegment_Super           0
ACSegment_Superplus       0
Maritalstatus_Married     0
Maritalstatus_Single      0
Logindevice_Mobile        0
dtype: int64
```

5.3 Addition of new Variable-

Cluster code will be added to the dataset. It may be used when experimenting with model building.

5.4 Variable of transformation-

- Encoding: The variables Payment, Gender, Account_Segment, Marital_Status and Login_device are all categorical object types. They need to be converted to numeric variables. The categories in these variables do not have an order. Hence, they were one-hot encoded.
- Scaling: The dataset has been scaled as it is a pre-requisite for any distance-based algorithm like KNN imputer, K-means clustering, KNN and ANN. The scaled data can also be used by all models irrespective of whether they expect scaled input or not.
- No other transformation is expected for modeling as of now.

5.5 Outlier Treatment-

Here, outliers will be treated by capping to the lower and upper range where

- lower_range= Q1-(1.5 * IQR) and
- upper_range= Q3+(1.5 * IQR)

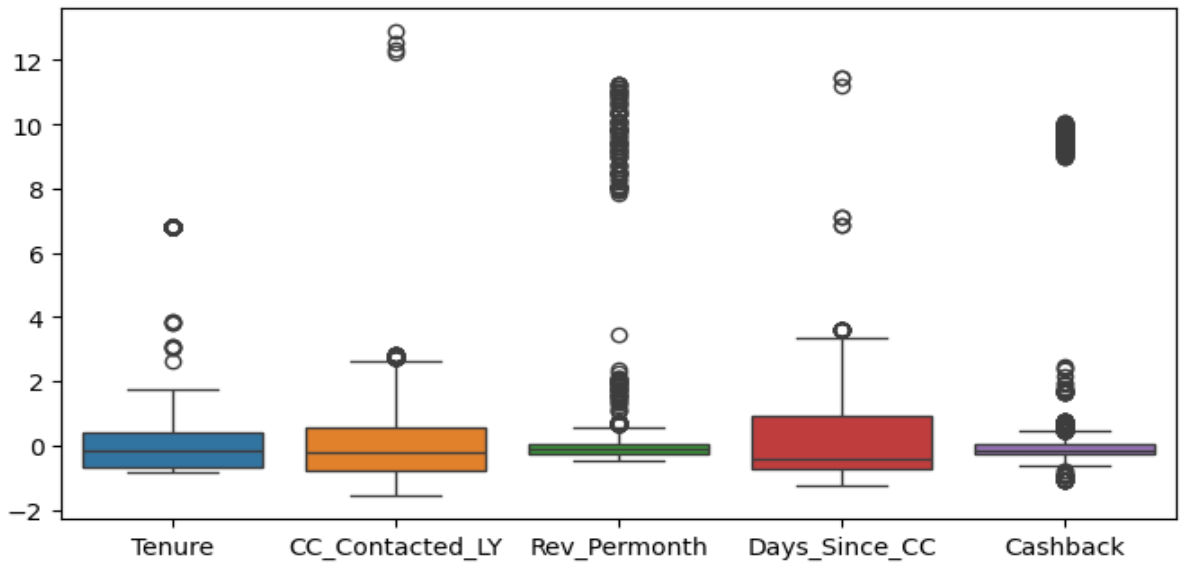


Fig 9.1 Before outlier treatment

The outliers have now been treated with lower and upper range. Plotting box plot after outlier treatment-

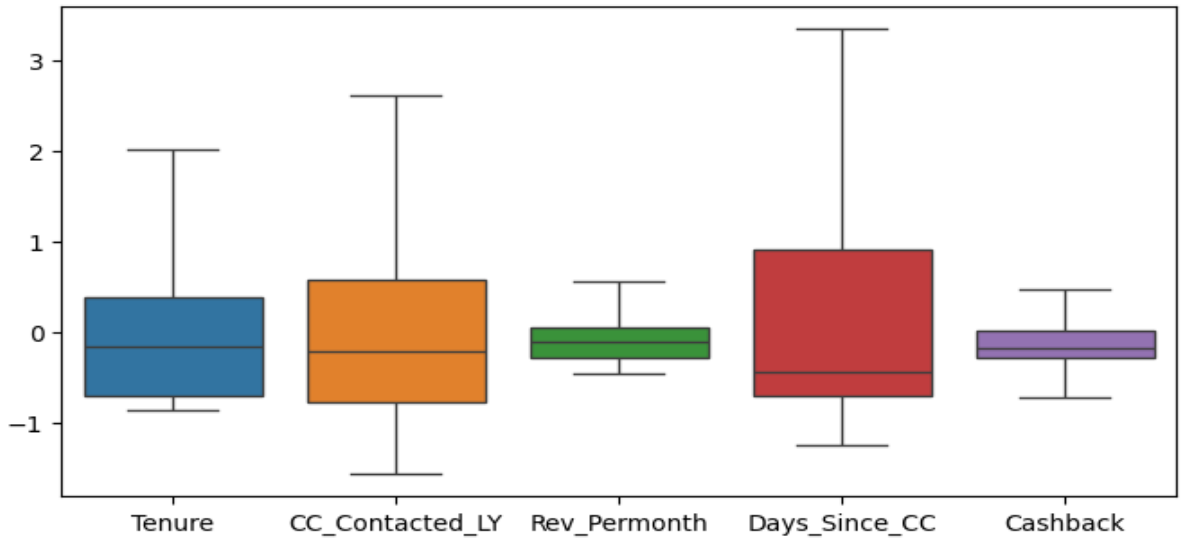


Fig 9.2 After outlier treatment

	Skewness
Tenure	0.80
CC_Contacted_LY	0.80
Rev_Permonth	0.78
Complain_LY	0.95
Days_Since_CC	0.82
Cashback	0.93

Observations-

- Some outliers for certain variables are closer to the whisker, whereas there are a group of outliers that are far beyond the whisker with no in between values.
- Two approaches to modelling were performed - one set of data with outliers treated for outlier sensitive models and another set of data with outliers not treated for outlier resistant algorithms.
- Coupon_used_for_payment has a very limited range 0 to 16. Hence, for the purpose of this analysis, the outliers will not be treated.

6. Clustering-

- Clustering is an unsupervised task and given all the features in the dataset, the clustering algorithm is allowed to group customers such that each group has similar customers and customers of different groups are dissimilar.
- For this purpose, the processed dataset (cleaned, scaled, nulls imputed, outlier treated, categorical features encoded) without the target variable was used.
- The algorithm was run for 2,3 and 4 clusters and 3 clusters seemed to have the best separation. The cluster profile was formed by grouping the observations by clusters and finding the mean for all the features.
- Although churn was not part of the features for clustering, it was added part of the cluster profile so that it is possible to appreciate how churn varies for each cluster.

kproto_3clusters	Churn	Tenure	City_Tier	CC_Contacted_LY	Complain_LY	Days_Since_CC
0	0.224396	10.669138	1.675718	30.626352	0.312352	3.846399
1	0.098399	13.556710	1.712985	14.992956	0.253772	8.880012
2	0.186470	9.535316	1.604142	13.368003	0.292514	2.212800

ACSegment_Regularplus	ACSegment_Super
0.330648	0.423092
0.135953	0.391849
0.535646	0.309122

7. Model Building-

In this business case, the need is to predict whether a given customer would churn or not. we will move towards various model building after EDA and data cleaning performed earlier followed by model tuning and assessing the performance over different metrics Accuracy, F1 Score, Recall, Precession, ROC curve, AUC score, Confusion matrix and classification report. We will choose the model which does not underfit or overfit along with the best accuracy in place.

7.1 Methodology-

- Different treatments of pre-processed data were prepared - with and without outliers, with and without SMOTE resampling, with and without scaling.
- VIF (Variance Inflation Factor) was calculated for all the predictor variables. One by one predictor variables whose VIF was more than 5 were identified. 4 variables – Cashback, Service Score, Clusters and User count had VIF greater than 5.
- It is to be noted that rev_growth_yoy and coupon_used_for_payment were dropped after Anova and Chi-square test were conducted and double checked against EDA plots. Hence none of the models have used these variables as predictors.
- Scaled data was used for distance-based algorithms such as KNN.
- Data was split into train and test sets in the ratio 70:30. 7882 records were assigned to the training dataset and 3378 records were assigned to the test dataset.
- 8 algorithms were chosen and for each algorithm-
  Base model (with default hyperparameters) was constructed and evaluation on train and test datasets performed
  ➔ Different data was used and performance measured and recorded

➔ Hyperparameters for the algorithm were tuned using SKlearn library's GridSearchCV and also manually if required.

➔ Performance was again measured for the tuned algorithm

➔ Feature importance was extracted from the model through in-built attributes for certain models. Amongst black box models, Sklearn's Permutation feature importance was used as a wrapper function on the model to obtain feature importance.

7.2 Evaluation metrics for model comparison-

The final best model was decided based on comparison of evaluation metrics for train and test data of all the models. The following criteria was used:

- The train and test performances should be comparable – i.e., no overfit or underfit. The model should be robust and reliable for use with unknown data.
- Maximum Precision score for 1s. The problem statement states that the Revenue assurance team does not want to give unnecessary freebies. If precision for 1s (churn customers) is high, then the actual churns out of the model's predicted churns would be high and hence the revenue assurance team's criteria would be satisfied.

  Precision = True positive/ (True positive + False positive)
- High F1-score (to ensure Recall is also good) for Churns. High precision should not come at the cost of recall. The model should be able to get a good part of the actual churns as predicted churns so that this prediction exercise is meaningful and the DTH provider can actually address their churn problem. Hence combination of Precision and Recall to get the F1-score is important.
- Model should be interpretable.
- Model should not be computationally expensive (like KNN).

8.Splitting Data into Train and Test Dataset: -

Following the accepted market practice, we have divided data into Train and Test dataset into a 70:30 ratio and building various models on training dataset and testing for accuracy over testing dataset.

Below is the shape of Train and Test dataset: -

```
((7882, 20), (3378, 20), (7882,), (3378,))
```

8.1 Treating im-balance nature of data-

Dataset provided is an imbalance in nature. The categorical count of our target variable "Churn" shows high variation in counts.

```
(Churn
 0    0.831642
 1    0.168358
 Name: proportion, dtype: float64,
 Churn
 0    0.831557
 1    0.168443
 Name: proportion, dtype: float64)
```

- This imbalance in the dataset can be performed using SMOTE technique will generates additional data points to balance the data.
- We need to apply SMOTE only to train the dataset not on the test dataset. divided data into train and test dataset in 70:30 ratio as an accepted market practice.

After SMOTE-

```
X_train_res (13110, 20)
y_train_res (13110,)
```

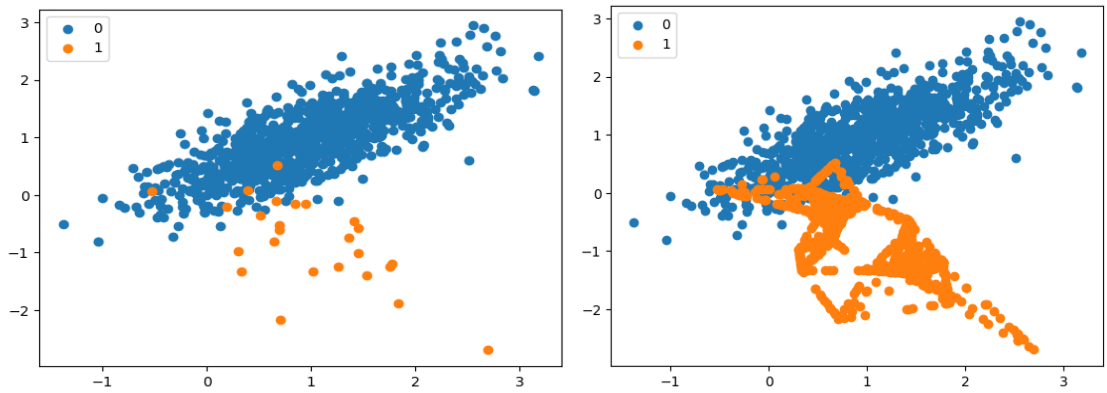


Fig.10 Before and After SMOTE

- 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.

9. Model Building and Tuning-

8 algorithms were tried for this dataset – Logistic Regression, Linear Discriminant Analysis, Support Vector Machine, Naive Bayes, K-Nearest Neighbours, Random Forest, Adaboost, Gradient Boost,Ada-Boosting. Algorithm implementations from the Sklearn package were used for all algorithms. In addition to SKlearn,

the Logistic regression algorithm was also executed using the Statsmodel package. The Appendix contains detailed information for all the 8 models including details on the base and the tuned model performance along with their interpretations.

9.1 Effort for model tuning-

Model performance improvement was accomplished using the following methods:

Around 8 different algorithms were tried across linear, non-linear and ensemble methods.

- The first model for each algorithm was the base model with the default hyperparameters. Model tuning to achieve better performance was done by tuning the hyperparameters using Grid Search CV, a function offered by Sklearn to automatically try out multiple parameter options for each hyperparameter.
- The underlying data was changed (Smote resampled/ non-resampled, outlier treated/not treated) and the effect of model performance observed for some of the models.
- Ensemble methods were used (Random Forest, Adaboost, Gradient Boost) and their hyperparameters were also tuned.

10. Building Logistic Regression Model -

10.1 Building Model with Default hyperparameters-

Post splitting data into a training and testing data set we fitted a logistic regression model into the training dataset and performed prediction on training and testing dataset using the same model. We made the first model with default hyperparameters with default solver as Ibfgs.

Below are the accuracy scores obtained from this model: -

```
Accuracy of training dataset: 0.8920324790662268
```

```
Accuracy of testing dataset: 0.8913558318531676
```

Below is the confusion matrix obtain from the model-

```
Confusion Matrix for train dataset
array([[6358,  197],
       [ 654,  673]])
```

```
Confusion Matrix for test dataset
array([[2729,   80],
       [ 287,  282]])
```

Below is the classification report obtain from this model-

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

           0       0.91      0.97      0.94      6555
           1       0.77      0.51      0.61      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.74      0.77      7882
weighted avg       0.88      0.89      0.88      7882


Classification report for test dataset
              precision    recall  f1-score   support

           0       0.90      0.97      0.94      2809
           1       0.78      0.50      0.61       569

    accuracy                           0.89      3378
   macro avg       0.84      0.73      0.77      3378
weighted avg       0.88      0.89      0.88      3378
```

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

```
AUC score and ROC curve for training dataset-
AUC: 0.884
AUC score and ROC curve for testing dataset-
AUC: 0.884
```

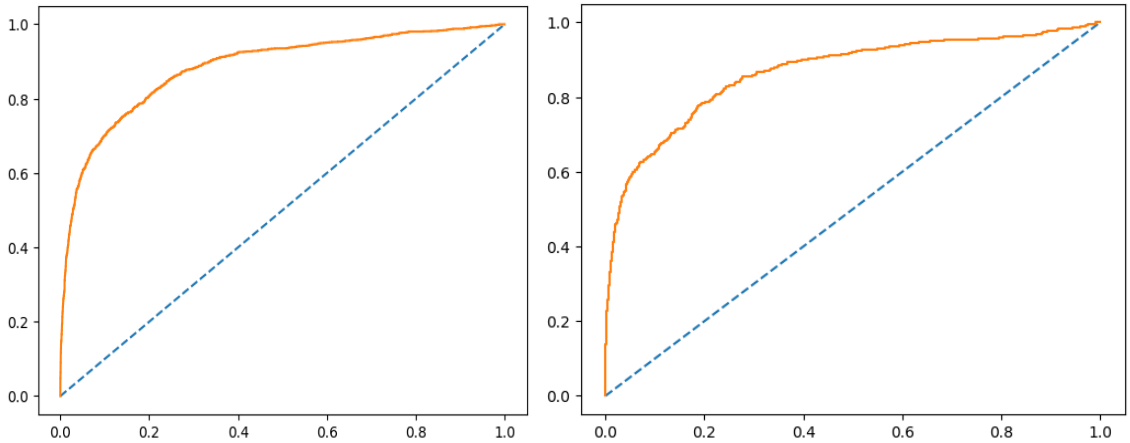


Fig.11.1 ROC curve and AUC score from Logistic Regression

Cross-validation is a process to check if the built model is correct or not. Below are the 10-fold cross validation scores: -

Cross-Validation Scores for training dataset-

```
array([0.878327  , 0.90240811, 0.88832487, 0.8857868 , 0.88451777,
       0.88705584, 0.87436548, 0.89974619, 0.89467005, 0.90482234])
```

Cross-Validation Scores for testing dataset-

```
array([0.92011834, 0.84615385, 0.87573964, 0.90828402, 0.89349112,
       0.87573964, 0.88757396, 0.8964497 , 0.88724036, 0.89317507])
```

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

10.2 Building Model using GridSearchCV and analysing the best parameters-

We use GridSearchCV to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Performed GridSearchCV with various hyperparameters like “solver”, “penalty” and “tol” and we can find that solver along with “none” penalty worked as best parameters for this dataset.

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

```
Accuracy of training dataset after gridsearchCV: 0.8903831514843948
```

```
Accuracy of testing dataset after gridsearchCV: 0.8904677323860273
```

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

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

           0       0.91      0.97      0.94      6555
           1       0.77      0.50      0.61      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.73      0.77      7882
weighted avg       0.88      0.89      0.88      7882

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.90      0.97      0.94      2809
           1       0.78      0.49      0.60       569

    accuracy                           0.89      3378
   macro avg       0.84      0.73      0.77      3378
weighted avg       0.88      0.89      0.88      3378
```

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

```
confusion matrix for train dataset
array([[6356,  199],
       [ 665,  662]])
```

```
confusion matrix for test dataset
array([[2728,   81],
       [ 289,  280]])
```

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

```
AUC score and ROC curve for training dataset
```

```
AUC: 0.884
AUC score and ROC curve for testing dataset
AUC: 0.867
```

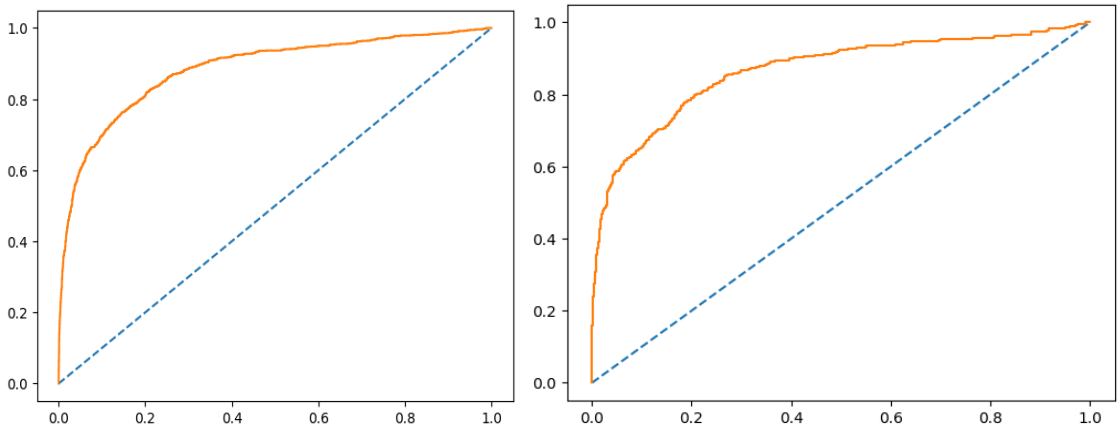


Fig.11.2 ROC Curve and AUC Score From Logistic Regression Using hyper-parameter

Below are the 10-fold cross validation scores-

```
cross validation score for training dataset
array([0.87959442, 0.89987326, 0.88705584, 0.89213198, 0.8857868 ,
       0.88832487, 0.87436548, 0.89974619, 0.89593909, 0.90482234])

cross calidation score for testing dataset
array([0.91420118, 0.84911243, 0.87573964, 0.90828402, 0.8964497 ,
       0.87573964, 0.88757396, 0.89940828, 0.884273  , 0.90207715])
```

10.3 Building Logistic regression model using SMOTE-

In our previous analysis we have seen that the data is imbalanced in nature. We can use the SMOTE technique to balance the data and then tried building a model on the balanced data to check if we can see some significant improvement in accuracy for training and testing the dataset. After building a model on a balanced dataset and checking on accuracy we can see that the performance is not that significant in terms of accuracy.

Below are the accuracy scores obtained from balanced data: -

```
Accuracy of training dataset: 0.8093058733790999

Accuracy of testing dataset: 0.7948490230905861
```

Below is the confusion matrix obtained from balanced data-

```
Confusion matrix for train dataset
array([[5237, 1318],
       [1182, 5373]])

Confusion matrix for test dataset
array([[2236,  573],
       [ 120,  449]])
```

Below is the classification report obtained from balanced data-

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

           0       0.82      0.80      0.81      6555
           1       0.80      0.82      0.81      6555

    accuracy                           0.81     13110
   macro avg       0.81      0.81      0.81     13110
weighted avg       0.81      0.81      0.81     13110

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.95      0.80      0.87      2809
           1       0.44      0.79      0.56       569

    accuracy                           0.79      3378
   macro avg       0.69      0.79      0.72      3378
weighted avg       0.86      0.79      0.82      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.889
AUC score and ROC curve for testing dataset
AUC: 0.866
```

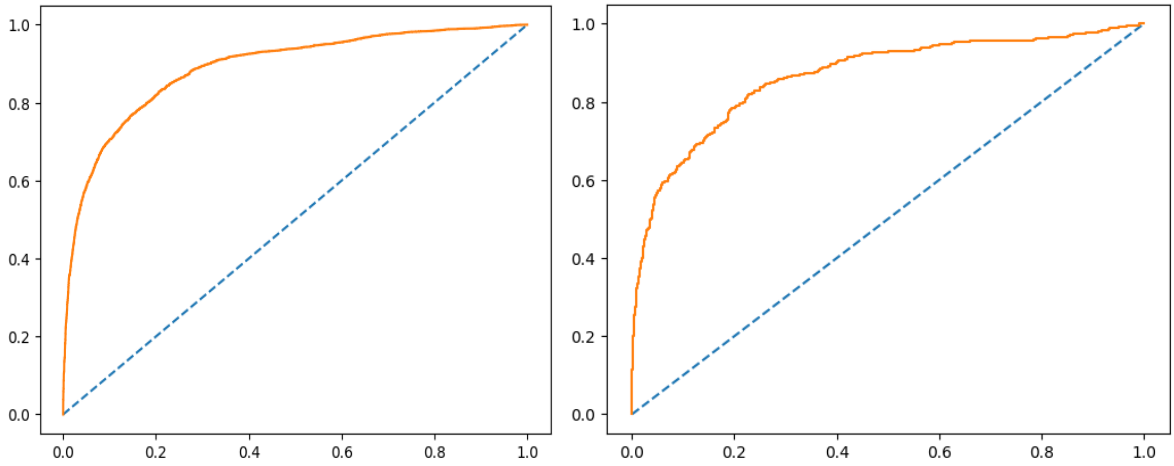


Fig 11.3 ROC Curve & AUC Scores From Logistic Regression with SMOTE

Below are the 10-fold cross validation scores-

Cross validation score for balanced training dataset-

```
array([0.80396644, 0.77955759, 0.80625477, 0.8085431 , 0.80396644,
       0.81922197, 0.81998474, 0.79328757, 0.82227307, 0.8215103 ])
```

Cross validation score for balanced test dataset-

```
array([0.92011834, 0.84615385, 0.87573964, 0.90828402, 0.89349112,
       0.87573964, 0.88757396, 0.8964497 , 0.88724036, 0.89317507])
```

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

Observations-

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using GridSearchCV is best optimized considering the best parameters obtained. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and balanced data (SMOTE). Model built on a balanced dataset using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

11. Building Linear Discriminant Analysis Model (LDA)-

11.1 Building model with default hyperparameters-

From the above split into training and testing dataset we are building Linear Discriminant Analysis (LDA) model to check if this can outperform Logistic regression model and we can choose form best for further predictions. Firstly, we are building a model using default values of LDA.

Below are the accuracy scores obtained from this model-

```
Accuracy score of training dataset: 0.8869576249682821

Accuracy score of testing dataset: 0.8848431024274719
```

Below is the confusion matrix obtained from this model-

```
Confusion matrix of training dataset
array([[6364,  191],
       [ 700,  627]])
```

```
Confusion matrix of testing dataset
array([[2731,   78],
       [ 311,  258]])
```

Below is the classification report obtained from this model-

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      6555
           1       0.77      0.47      0.58      1327

    accuracy                           0.89      7882
   macro avg       0.83      0.72      0.76      7882
weighted avg       0.88      0.89      0.88      7882


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      2809
           1       0.77      0.45      0.57       569

    accuracy                           0.88      3378
   macro avg       0.83      0.71      0.75      3378
weighted avg       0.88      0.88      0.87      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.866
AUC score and ROC curve for testing dataset
AUC: 0.866
```

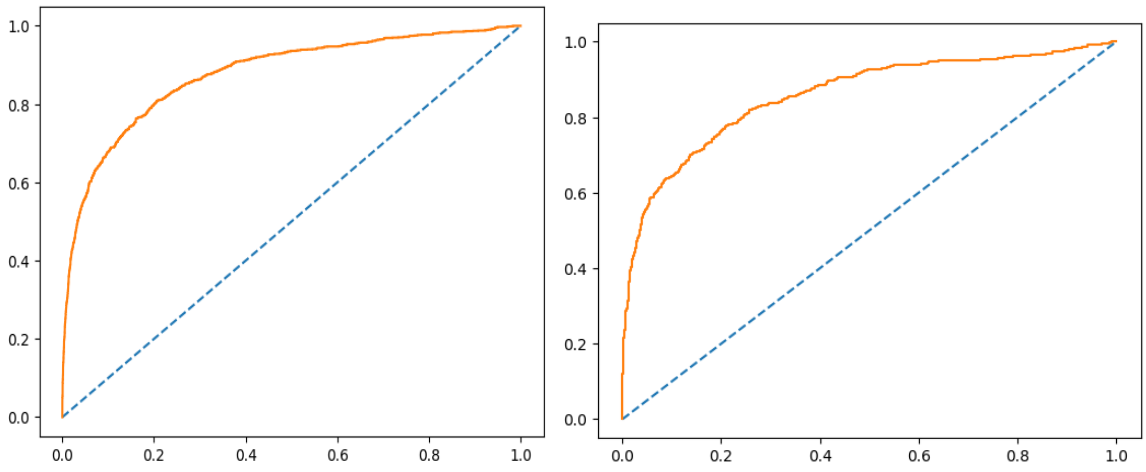


Fig. 12.1 ROC Curve and AUC Score From LDA

Below are the 10-fold cross validation scores-

```
cross validation score for training dataset
array([0.87325729, 0.89100127, 0.88071066, 0.88832487, 0.87944162,
       0.88324873, 0.87563452, 0.89593909, 0.89340102, 0.90482234])
```

```
cross validation score for testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

11.2 Building model using GridSearchCV and analysing the best parameters-

Using the GridSearchCV function we tried finding the best parameters to further tune-in the above model for better accuracy. The best model considering accuracy, precision, recall, F1, ROC curve and AUC score.

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

```
Accuracy of training dataset after gridsearchCV: 0.887338239025628

Accuracy of testing dataset after gridsearchCV: 0.8848431024274719
```

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

```
confusuon matrix for training dataset
array([[6368,  187],
       [ 701,  626]])
confusuon matrix for testing dataset
array([[2733,   76],
       [ 313,  256]])
```

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

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

           0       0.90      0.97      0.93      6555
           1       0.77      0.47      0.59      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.72      0.76      7882
weighted avg       0.88      0.89      0.88      7882


Classification report for test dataset
              precision    recall  f1-score   support

           0       0.90      0.97      0.93      2809
           1       0.77      0.45      0.57       569

    accuracy                           0.88      3378
   macro avg       0.83      0.71      0.75      3378
weighted avg       0.88      0.88      0.87      3378
```

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

```
AUC score and ROC curve for training dataset
```

```
AUC: 0.876
AUC score and ROC curve for testing dataset
AUC: 0.858
```

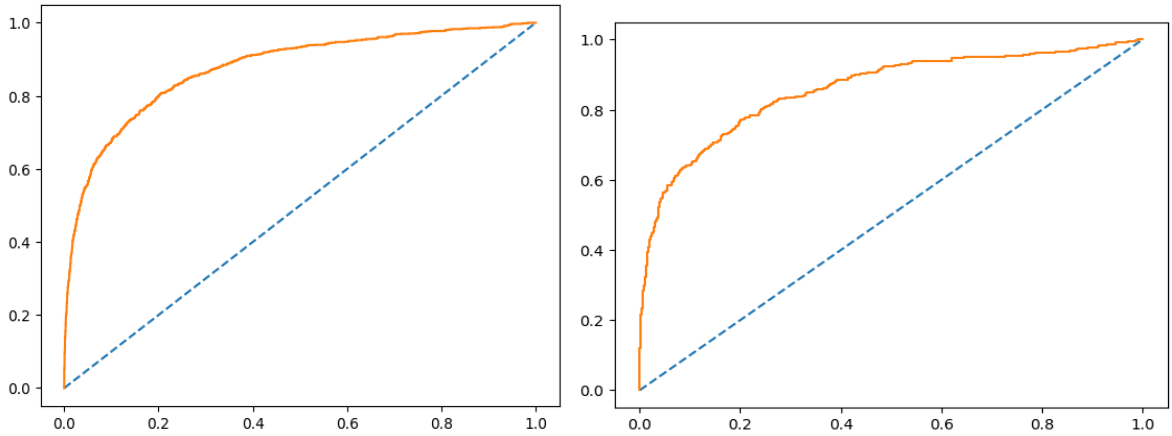


Fig.12.2 ROC Curve and AUC Score From LDA with Hypertuning

Below are the 10-fold cross validation scores-

```
cross validation scores for training dataset
array([0.87452471, 0.88846641, 0.88324873, 0.8857868 , 0.87944162,
       0.8819797 , 0.87817259, 0.89467005, 0.89340102, 0.90482234])

cross validation scores from testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

11.3 Building LDA model using SMOTE-

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

```
Accuracy of training dataset: 0.8057971014492754

Accuracy of testing dataset: 0.7835997631734755
```

Below is the confusion matrix obtained from balanced dataset-

```
confusion matrix for training dataset
array([[5139, 1416],
       [1130, 5425]])
```

```
confusion matrix for testing dataset
array([[2196,  613],
       [ 118,  451]])
```

Below is the classification report obtained from balanced dataset-

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

           0       0.82      0.78      0.80      6555
           1       0.79      0.83      0.81      6555

    accuracy                           0.81     13110
   macro avg       0.81      0.81      0.81     13110
weighted avg       0.81      0.81      0.81     13110

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.95      0.78      0.86      2809
           1       0.42      0.79      0.55       569

    accuracy                           0.78      3378
   macro avg       0.69      0.79      0.70      3378
weighted avg       0.86      0.78      0.81      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.888
AUC score and ROC curve for testing dataset
AUC: 0.864
```

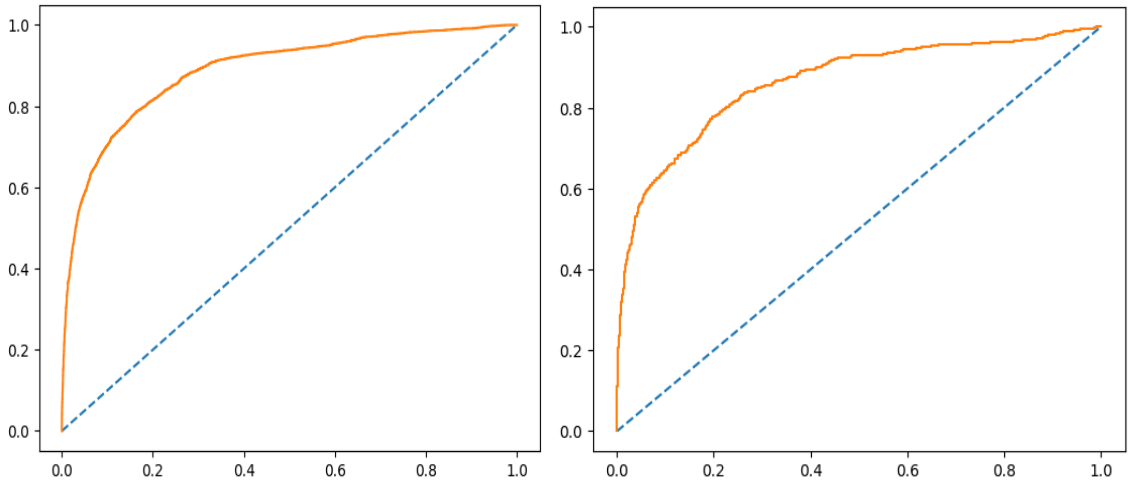


Fig.12.3 ROC Curve and AUC Score From LDA with SMOTE

Below are the 10-fold cross validation scores-

```
cross validation scores for training dataset
array([0.80015256, 0.78108314, 0.80320366, 0.80930587, 0.80472921,
       0.81388253, 0.81617086, 0.78947368, 0.81617086, 0.82227307])

cross validation scores for testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

Inference from LDA Model-

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using the parameters from GridSearchCV is best optimized. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and models built using GridSearchCV. Model built on a balanced data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

12. Building KNN Model-

12.1 Building model with default hyperparameters-

Post splitting data into training and testing data set we fitted KNN 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 of n_neighbour as “5”.

Below are the accuracy scores obtained from this model-

```
Accracy of training dataset: 0.9545800558233951

accuracy for testing dataset 0.9177027827116637
```

Below are the confusion matrices obtained from this model-

```
confusion matrix of training dataset  confusion matrix of training dataset
[[6432  123]                          [[6432  123]
 [ 235 1092]]                          [ 235 1092]]
```

Below are the classification Report obtained from this model-

```
classificatoin report of training dataset
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      6555
           1       0.90      0.82      0.86      1327

    accuracy                           0.95      7882
   macro avg       0.93      0.90      0.92      7882
weighted avg       0.95      0.95      0.95      7882
```

```
classsification report for testing dataset
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      2809
           1       0.80      0.68      0.73       569

    accuracy                           0.92      3378
   macro avg       0.87      0.82      0.84      3378
weighted avg       0.91      0.92      0.91      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.986
AUC score and ROC curve for testing dataset
AUC: 0.941
```

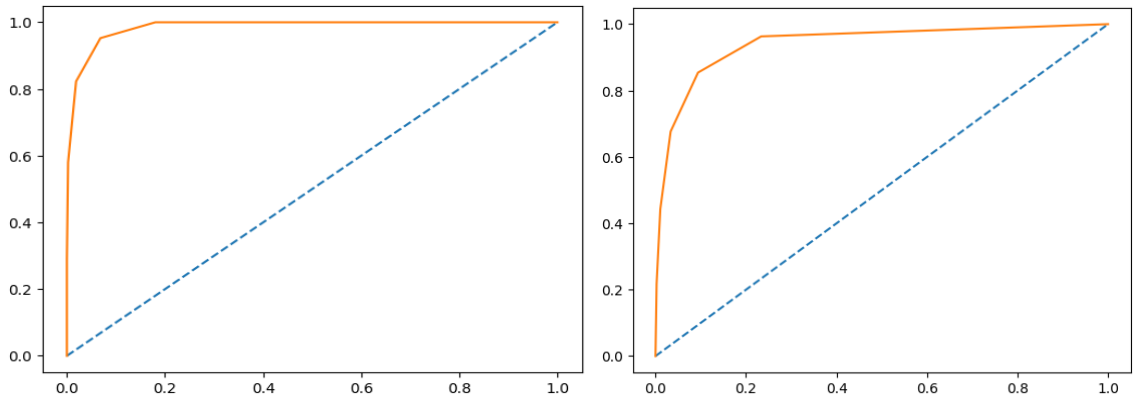


Fig.13.1 ROC Curve and AUC Scores From KNN

Observation-Train dataset has overfit because of a smaller number of neighbours selection. Let us try to observe the f1-score and accuracy for different values of K (neighbours) again using the above hyperparameters**.**

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.89607098, 0.92141952, 0.91116751, 0.90989848, 0.91116751,
       0.90609137, 0.90228426, 0.92639594, 0.91243655, 0.91370558])

cross validation scores for test dataset
array([0.88757396, 0.85798817, 0.85798817, 0.88461538, 0.89349112,
       0.87278107, 0.88461538, 0.88461538, 0.83976261, 0.87240356])
```

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

12.2Tuning KNN with varying values of K-

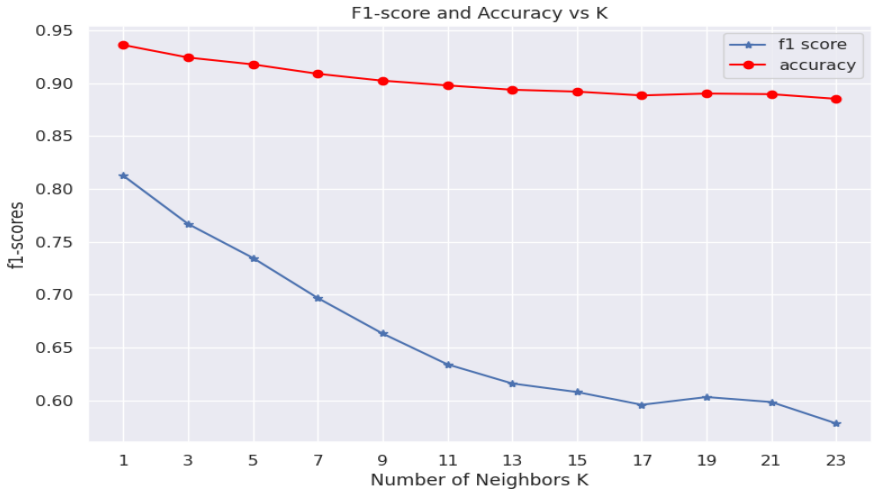


Fig.13.2 f1-score and Accuracy vs K

Observation-

- 5 seems to be the optimum value but for that, train dataset has fully grown.
- KNN_model2 has the best grid with best neighbours. Even though the training data performance has fully grown, test data is not far behind and has a difference of 7%. Let's use cross validation on full data to see if the f1-score of 93% holds true. For 5-fold cross validation on full data, the following are the F1-scores.

12.3 Find the right value of n_neighbor-

It's very important to have the right value of n_neighbors to fetch the best accuracy from the model. We can decide on the best value for n_neighbors based on MSE (mean squared error) scores. The value with least score of MSE indicated least error and will fetch the best optimized n_neoghbors value.

Below are the MSE scores-

```
[0.06394316163410307,
 0.07578448786264058,
 0.08229721728833628,
 0.09117821195973952,
 0.09769094138543521,
 0.10213143872113672,
 0.10627590290112487,
 0.10805210183540559,
 0.1116044997039668,
 0.1098283007696862]
```

Below is the graphical version of of MSE scores across numerous values of n_neighbors-

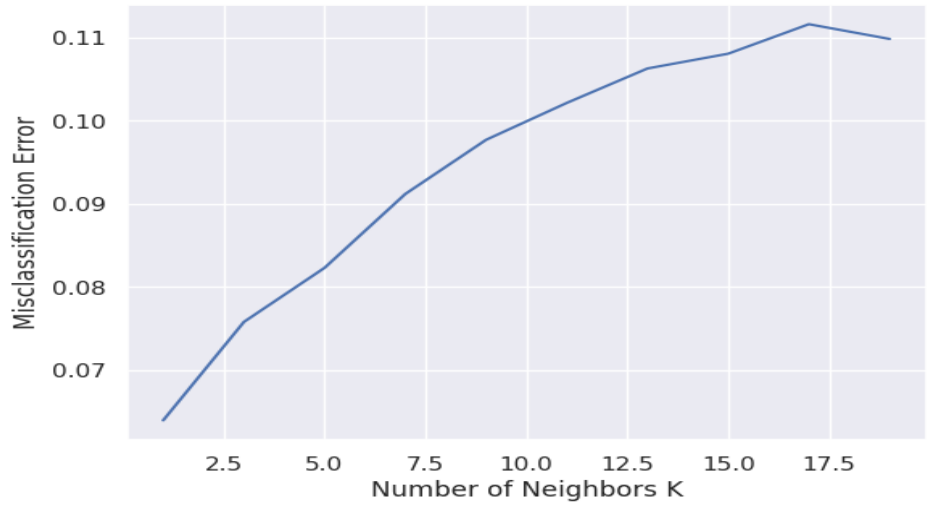


Fig.13.3 Graphical Version Of MSE Score

12.4 Building model using GridSearchCV and getting the best hyperparameters-

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

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

```
Accuracy of training dataset after gridsearchCV: 1.0

Accuracy of testing dataset after gridsearchCV: 0.9547069271758437
```

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

```
confusuon matrix for training dataset
array([[6555,    0],
       [   0, 1327]])
confusuon matrix for testing dataset
array([[2757,   52],
       [ 101,  468]])
```

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

```
Classification report for train dataset
              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

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      2809
           1       0.90      0.82      0.86       569

    accuracy                           0.95      3378
   macro avg       0.93      0.90      0.92      3378
weighted avg       0.95      0.95      0.95      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.979
```

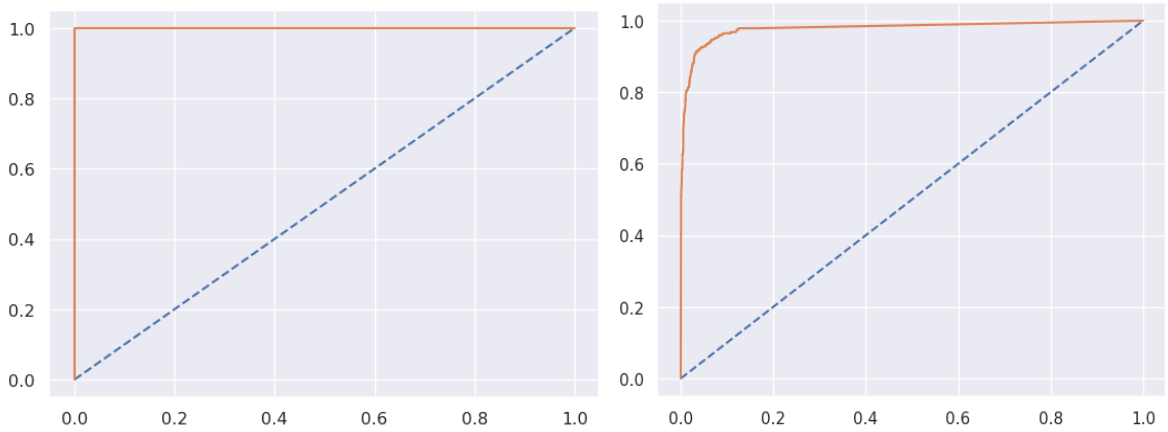


Fig.13.4 ROC Curve and AUC Score From KNN with Hyperparameter Tuning

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.93789607, 0.95437262, 0.95939086, 0.95431472, 0.95812183,
       0.96573604, 0.94162437, 0.96700508, 0.95304569, 0.95939086])

cross validation scores for test dataset
array([0.90828402, 0.8816568 , 0.89940828, 0.90236686, 0.9260355 ,
       0.91420118, 0.93195266, 0.91715976, 0.90207715, 0.9347181 ])
```

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

12.5 Building KNN model over balanced dataset using SMOTE-

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

```
Accuracy of training dataset: 0.8931350114416476

Accuracy of testing dataset: 0.8001776198934281
```

Below are the confusion matrices obtained from balanced dataset-

```
confusion matrix for training dataset
array([[5230, 1325],
       [  76, 6479]])
confusion matrix for testing dataset
array([[2188,  621],
       [  54,  515]])
```

Below are the classification reports obtained from balanced dataset-

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

           0       0.99      0.80      0.88      6555
           1       0.83      0.99      0.90      6555

    accuracy                           0.89     13110
   macro avg       0.91      0.89      0.89     13110
weighted avg       0.91      0.89      0.89     13110

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.98      0.78      0.87      2809
           1       0.45      0.91      0.60       569

    accuracy                           0.80      3378
   macro avg       0.71      0.84      0.74      3378
weighted avg       0.89      0.80      0.82      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.984
AUC score and ROC curve for testing dataset
AUC: 0.928
```

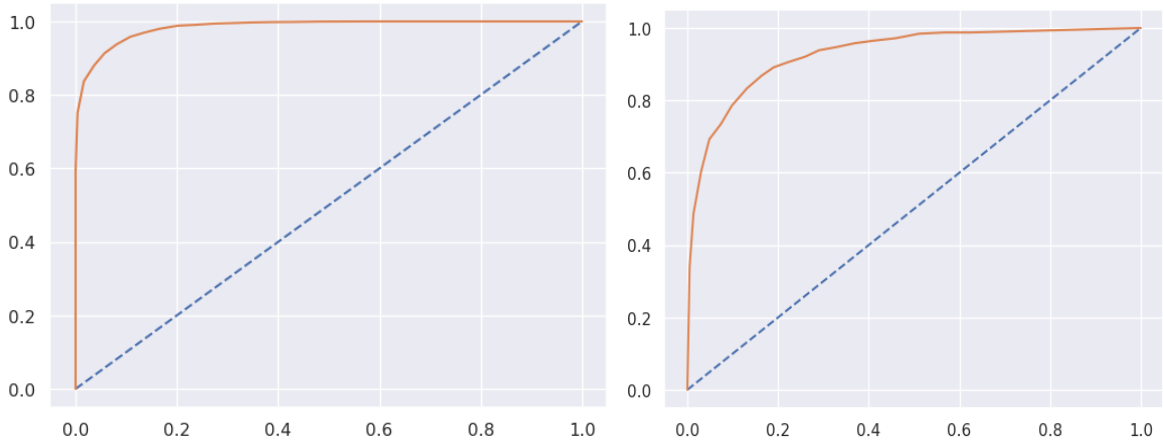


Fig.13.5 ROC Curve and AUC Scores From KNN with SMOTE

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.86575133, 0.85583524, 0.90541571, 0.86956522, 0.8733791 ,
       0.8863463 , 0.87795576, 0.88024409, 0.86880244, 0.89016018])

cross validation scores for test dataset
array([0.88757396, 0.84319527, 0.84023669, 0.86390533, 0.87869822,
       0.87573964, 0.87573964, 0.87573964, 0.84866469, 0.88724036])
```

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

Inference-

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using grid search CV is the best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with default values of KNN and also with model built on balanced dataset. Model built on a balanced data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

13. Building Gaussian Naive Bayes-

13.1 Building model with default hyperparameters-

Post splitting data into training and testing we are now ready to build model using Naive Bayes algorithm. Bayes' algorithm is based on Bayes theorem of conditional probability. Consider that all events are independent of each other and then we find the probability of an event happening under the condition that one of the events has already occurred.

Below are accuracy scores using this model-

```
Accracy of training dataset: 0.7854605430093885

Accracy of testing dataset: 0.7818235642391947
```

Below are confusion matrices and classification reports using this model-

```
Confusion matrix of train dataset
[[5240 1315]
 [ 376  951]]
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      6555
           1       0.42      0.72      0.53      1327

    accuracy                           0.79      7882
   macro avg       0.68      0.76      0.70      7882
weighted avg       0.85      0.79      0.81      7882

Confusion matrix of test dataset
[[2234  575]
 [ 162  407]]
Classification report of test dataset
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      2809
           1       0.41      0.72      0.52       569

    accuracy                           0.78      3378
   macro avg       0.67      0.76      0.69      3378
weighted avg       0.85      0.78      0.80      3378
```

Below are AUC scores and ROC curves using this model-

```
AUC score and ROC curve for training dataset
AUC: 0.827
AUC score and ROC curve for testing dataset
AUC: 0.810
```

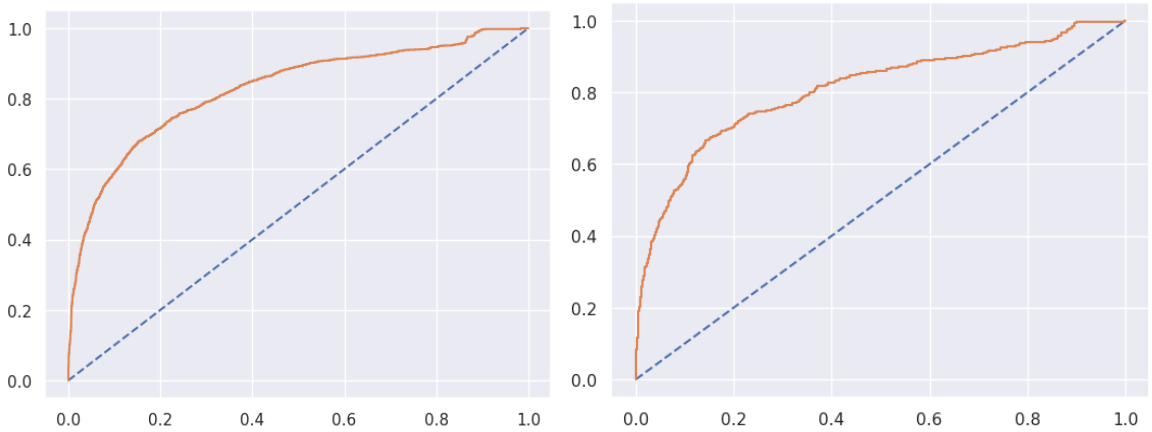


Fig.14.1 ROC Curve and AUC Scores From Naïve Bayes

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

```
cross validation scores for train dataset
array([0.76299113, 0.7896071 , 0.76395939, 0.79187817, 0.7677665 ,
       0.79187817, 0.80203046, 0.77918782, 0.79187817, 0.80329949])
cross validation scores for test dataset
array([0.80473373, 0.73668639, 0.72781065, 0.81065089, 0.76331361,
       0.78402367, 0.76627219, 0.80769231, 0.71513353, 0.78635015])
```

13.2 Building Gaussian Naive Bayes over balanced data using SMOTE-

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

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

```
Accracy of training dataset: 0.736231884057971

Accracy of testing dataset: 0.6397276494967437
```

Below are the confusion matrices and classification reports obtained using Naive Bayes algorithm over balanced dataset-

```
Confusion matrix of train dataset
[[3970 2585]
 [ 873 5682]]
              precision    recall  f1-score   support

           0       0.82      0.61      0.70      6555
           1       0.69      0.87      0.77      6555

    accuracy                           0.74     13110
   macro avg       0.75      0.74      0.73     13110
weighted avg       0.75      0.74      0.73     13110

Confusion matrix of test dataset
[[1683 1126]
 [  91  478]]
Classification report of test dataset
              precision    recall  f1-score   support

           0       0.95      0.60      0.73      2809
           1       0.30      0.84      0.44       569

    accuracy                           0.64      3378
   macro avg       0.62      0.72      0.59      3378
weighted avg       0.84      0.64      0.68      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.837
```

```
AUC score and ROC curve for testing dataset
AUC: 0.809
```

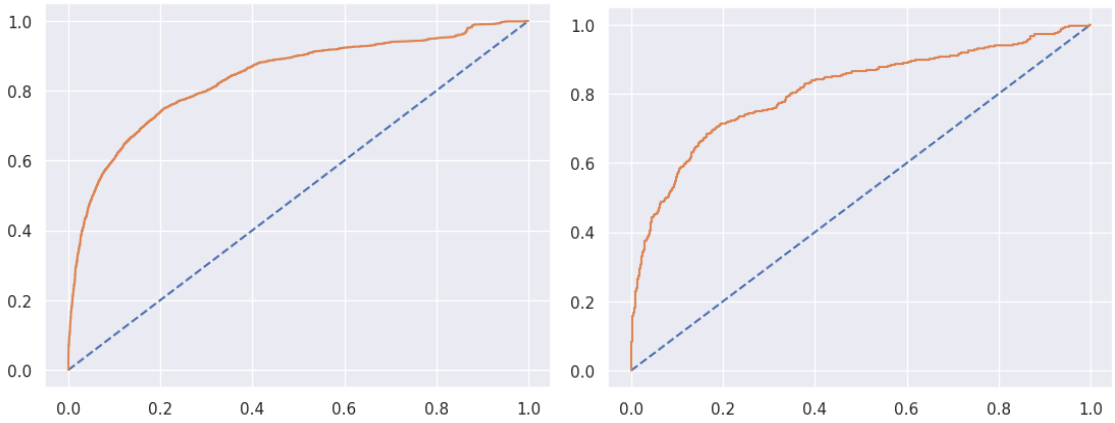


Fig.14.2 ROC Curve and AUC Scores From Naïve Bayes with SMOTE

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

```
cross validation scores for train dataset
array([0.72158658, 0.72234935, 0.74065599, 0.74141876, 0.72158658,
       0.74065599, 0.75133486, 0.71700992, 0.74599542, 0.75591152])

cross validation scores for test dataset
array([0.80473373, 0.73668639, 0.72781065, 0.81065089, 0.76331361,
       0.78402367, 0.76627219, 0.80769231, 0.71513353, 0.78635015])
```

Inference from Naive Bayes Model-

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using original imbalanced data is the best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with models built on a balanced dataset. Model built on a balanced data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to training dataset.

14. Ensemble Model-Random Forest-

Random forest is an ensemble machine learning algorithm that uses bootstrapping to reduce variance in the underlying decision trees. It also selects only a subset of features for each node split decision. Since it is an ensemble of trees with varying features for each node, each tree is different from another. Random forest is resistant to outliers and does not require the data to be scaled.

14.1Random Forest base model with default hyperparameters-

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

```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[6555    0]
 [   0 1327]]
classification report for training dataset
              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

accuracy score for testing dataset: 0.9742451154529307
confusion matrix for testing dataset
[[2794   15]
 [  72  497]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      2809
           1       0.97      0.87      0.92       569

    accuracy                           0.97      3378
   macro avg       0.97      0.93      0.95      3378
weighted avg       0.97      0.97      0.97      3378
```

Below are ROC Curve and AUC Score of train and test dataset-

```
AUC score and ROC curve for training dataset
AUC: 0.825
AUC score and ROC curve for testing dataset
AUC: 0.809
```

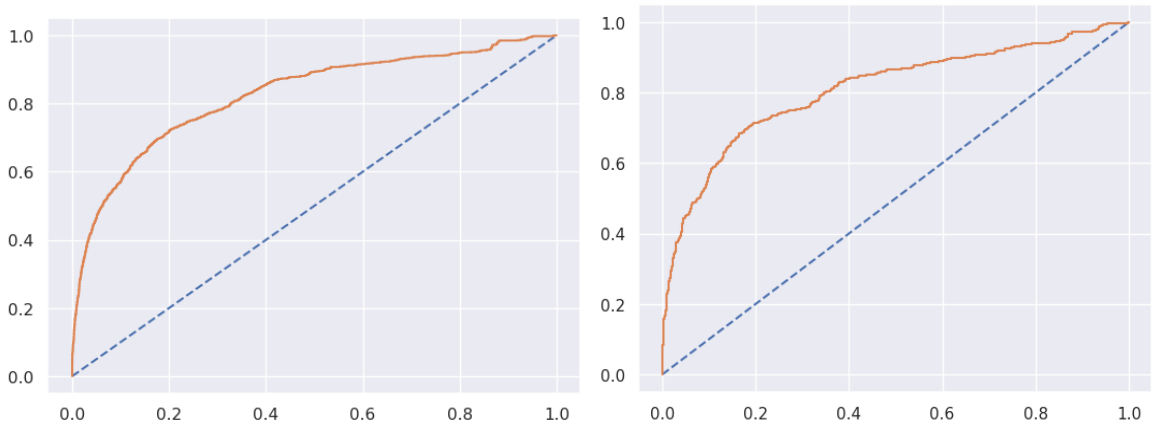


Fig.15.1 ROC Curve and AUC Scores From Random Forest

14.2Building random forest using GridSearchCV and getting the best hyperparameters-

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

```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[6555    0]
 [   0 6555]]
classification report for training dataset
              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
```

```
accuracy score for testing dataset: 0.9748371817643576
confusion matrix for testing dataset
[[2791   18]
 [  67  502]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2809
           1       0.97      0.88      0.92       569

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

14.3 Feature Importance-

This model has overfitted both in the default base model as well as the hyper parameter tuned model. The tuned model in fact performed worse than the base model. Hence in the final comparison, this cannot get selected as the best model.

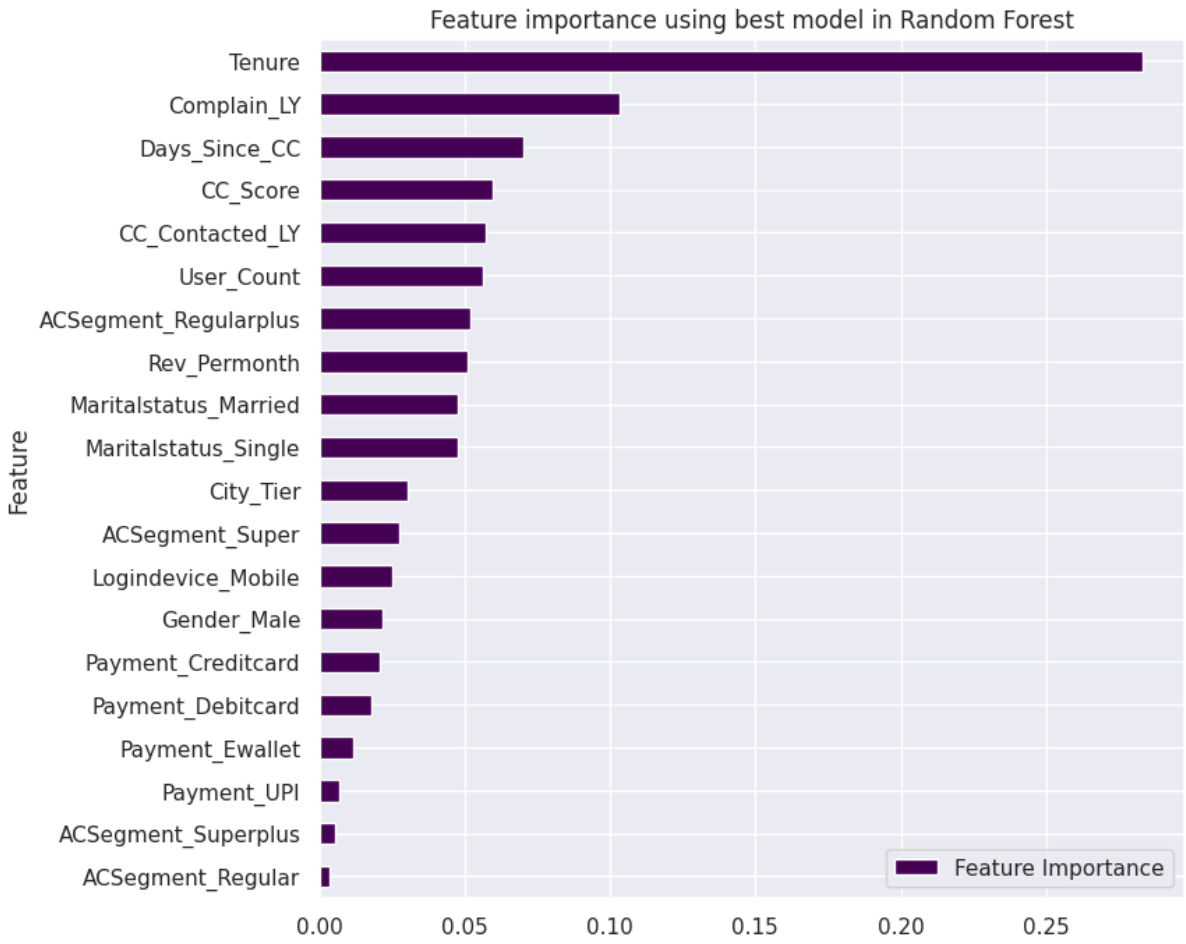


Fig.15.2 Feature importance from the best Random Forest model

This model has picked Tenure to be the most important feature followed by Customer care contacted previous year and Days since customer care last contact.

- From EDA, we can observe that for lower tenures especially within the first year, the churn is higher. Hence, once a customer has been acquired, the first year is very important to keep the customer satisfied.
- The next important parameter to predict customer churn per this model is the number of times customer care was contacted by the customer. Per EDA, the median and third quartile of number of times customer care was contacted previous year is higher for churned customers compared to active/current customers.
- Churned customers had contacted customer care more recently before churning than active customers. Per EDA, median days since last customer connect is higher for active customers. Churned customers had contacted customer care recently before churning.

15.1 Bagging on original dataset-

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

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

```
accuracy score or training dataset: 0.9954326313118498
confusion report for training dataset
[[6552    3]
 [  33 1294]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      6555
           1       1.00      0.98      0.99      1327

    accuracy                           1.00      7882
   macro avg       1.00      0.99      0.99      7882
weighted avg       1.00      1.00      1.00      7882

Accuracy score for testing datatset: 0.9641799881586738
confusuion matrix for testing dataset
[[2792   17]
 [ 104  465]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      2809
           1       0.96      0.82      0.88       569

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

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.990
```

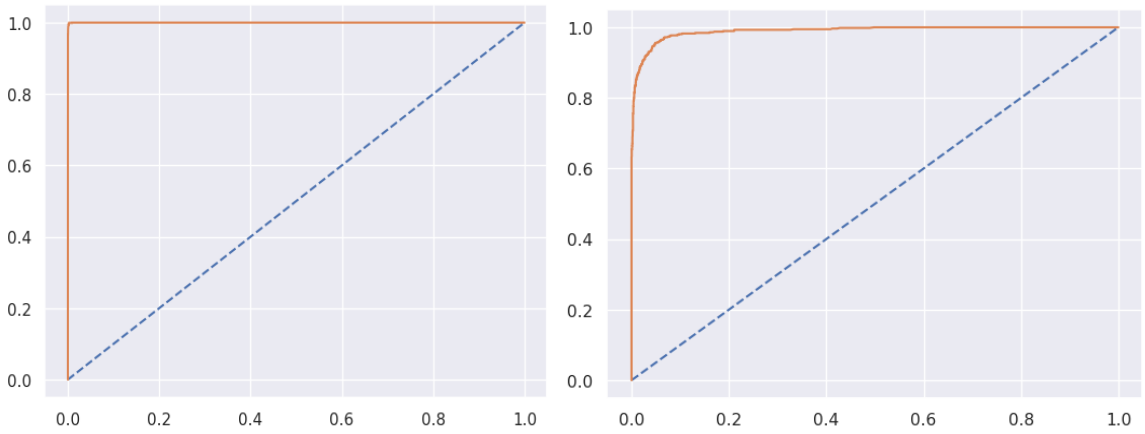


Fig.16.1 ROC Curve and AUC Score Bagging On Original Dataset

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

```
array([0.95690748, 0.97338403, 0.96954315, 0.95939086, 0.96192893,
       0.97208122, 0.94923858, 0.9606599 , 0.95939086, 0.95812183])
array([0.94674556, 0.91420118, 0.91715976, 0.9408284 , 0.92307692,
       0.93786982, 0.9260355 , 0.93195266, 0.91691395, 0.94065282])
```

15.2 Bagging on Balanced dataset-

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

```
accuracy score or training dataset: 0.998093058733791
confusion report for training dataset
[[6547    8]
 [  17 6538]]
classification report for training dataset
              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

Accuracy score for testing datatset: 0.9656601539372409
confusuion matrix for testing dataset
[[2771   38]
 [  78  491]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      2809
           1       0.93      0.86      0.89       569

    accuracy                           0.97      3378
   macro avg       0.95      0.92      0.94      3378
weighted avg       0.97      0.97      0.97      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.988
```

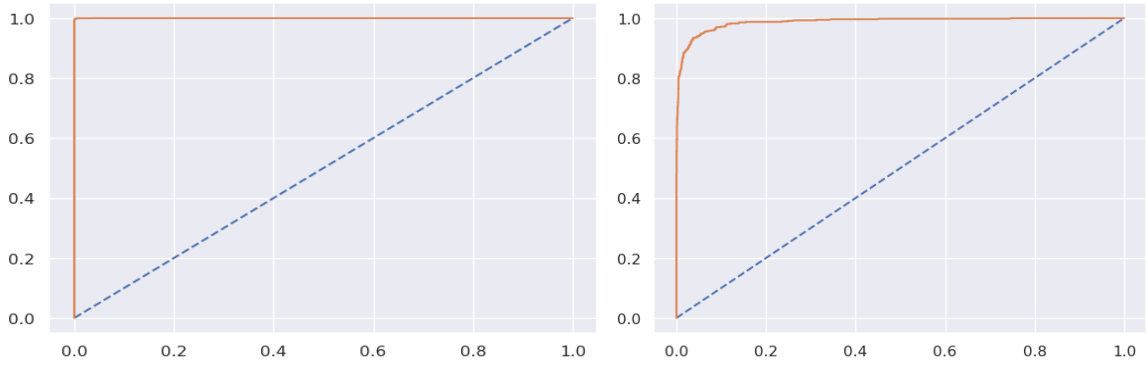


Fig.16.2 ROC Curve and AUC Scores from Bagging On Balanced Dataset

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

cross validation scores for test dataset-

```
array([0.93897788, 0.94050343, 0.99237223, 0.99313501, 0.98855835,
       0.98855835, 0.98932113, 0.99160946, 0.99237223, 0.99237223])
```

cross validation scores for test dataset-

```
array([0.94674556, 0.91420118, 0.91715976, 0.9408284 , 0.92307692,
       0.93786982, 0.9260355 , 0.93195266, 0.91691395, 0.94065282])
```

16. Ensemble Method AdaBoost-

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

16.1 Building Ada-Boost Model on original dataset-

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

```
Accuracy for training dataset: 0.8987566607460036
confusion matrix for training dataset
[[6276  279]
 [ 519  808]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      6555
           1       0.74      0.61      0.67      1327

    accuracy                           0.90      7882
   macro avg       0.83      0.78      0.80      7882
weighted avg       0.89      0.90      0.89      7882

accuracy score for testing dataset: 0.8996447602131439
confusion matrix for testing dataset
[[2698  111]
 [ 228  341]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      2809
           1       0.75      0.60      0.67       569

    accuracy                           0.90      3378
   macro avg       0.84      0.78      0.80      3378
weighted avg       0.89      0.90      0.89      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.915
AUC score and ROC curve for testing dataset
AUC: 0.903
```

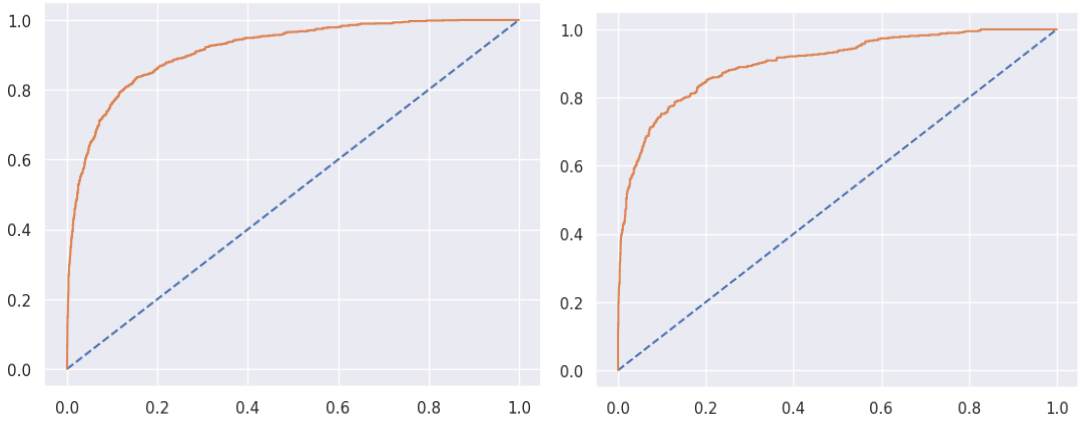


Fig. 17.1 ROC curve and AUC score from Ada-Boosting on Original Dataset

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

Cross validation scores for test dataset-

```
cross validation scores for train dataset
array([0.86311787, 0.90367554, 0.90736041, 0.91243655, 0.89340102,
       0.89086294, 0.88832487, 0.89847716, 0.89720812, 0.90482234])
```

Cross validation scores for training dataset-

```
cross validation scores for test dataset
array([0.9408284 , 0.85502959, 0.86390533, 0.9112426 , 0.90236686,
       0.88757396, 0.90532544, 0.90828402, 0.884273  , 0.90207715])
```

16.2 Building Ada-Boost Model on balanced dataset-

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

```
Accuracy for training dataset: 0.9082379862700228
confusion matrix for training dataset
[[5961  594]
 [ 609 5946]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      6555
           1       0.91      0.91      0.91      6555

    accuracy                           0.91     13110
   macro avg       0.91      0.91      0.91     13110
weighted avg       0.91      0.91      0.91     13110
```

```
accuracy score for testing dataset: 0.8783303730017762
confusion matrix for testing dataset
[[2552  257]
 [ 154  415]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      2809
           1       0.62      0.73      0.67       569

    accuracy                           0.88      3378
   macro avg       0.78      0.82      0.80      3378
weighted avg       0.89      0.88      0.88      3378
```

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

AUC score and ROC curve for training dataset

AUC: 0.967

```
AUC score and ROC curve for testing dataset
AUC: 0.901
```

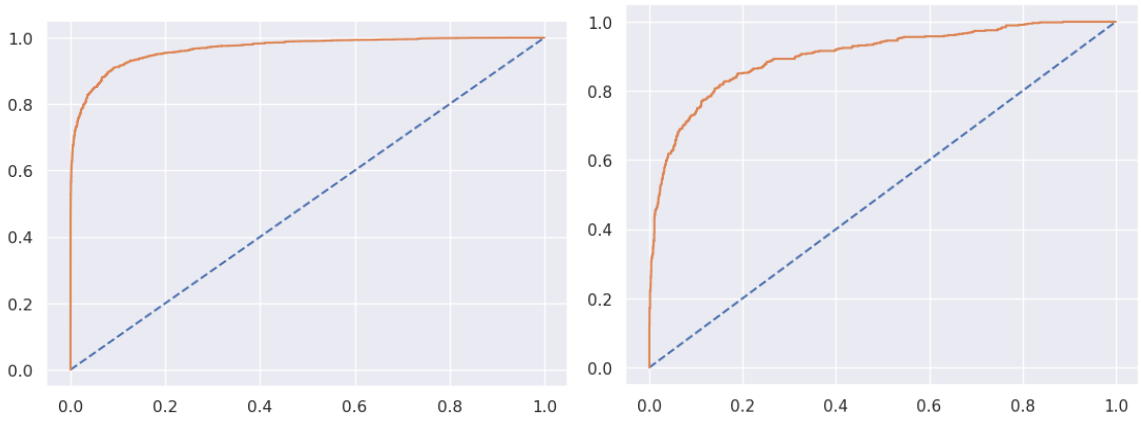


Fig.17.2 ROC Curve and AUC Score from Ada-Boosting on Balanced Dataset

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

Cross validation scores for train dataset-

```
cross validation scores for train dataset
array([0.80549199, 0.81083143, 0.92372235, 0.93058734, 0.92677346,
       0.91914569, 0.91685736, 0.91990847, 0.91914569, 0.92143402])
```

Cross validation scores for test dataset-

```
cross validation scores for test dataset
array([0.9408284 , 0.85502959, 0.86390533, 0.9112426 , 0.90236686,
       0.88757396, 0.90532544, 0.90828402, 0.884273  , 0.90207715])
```

The performance of this model when compared with other models is low. But it has not shown any overfitting or underfitting.

Feature importance from Adaboost-

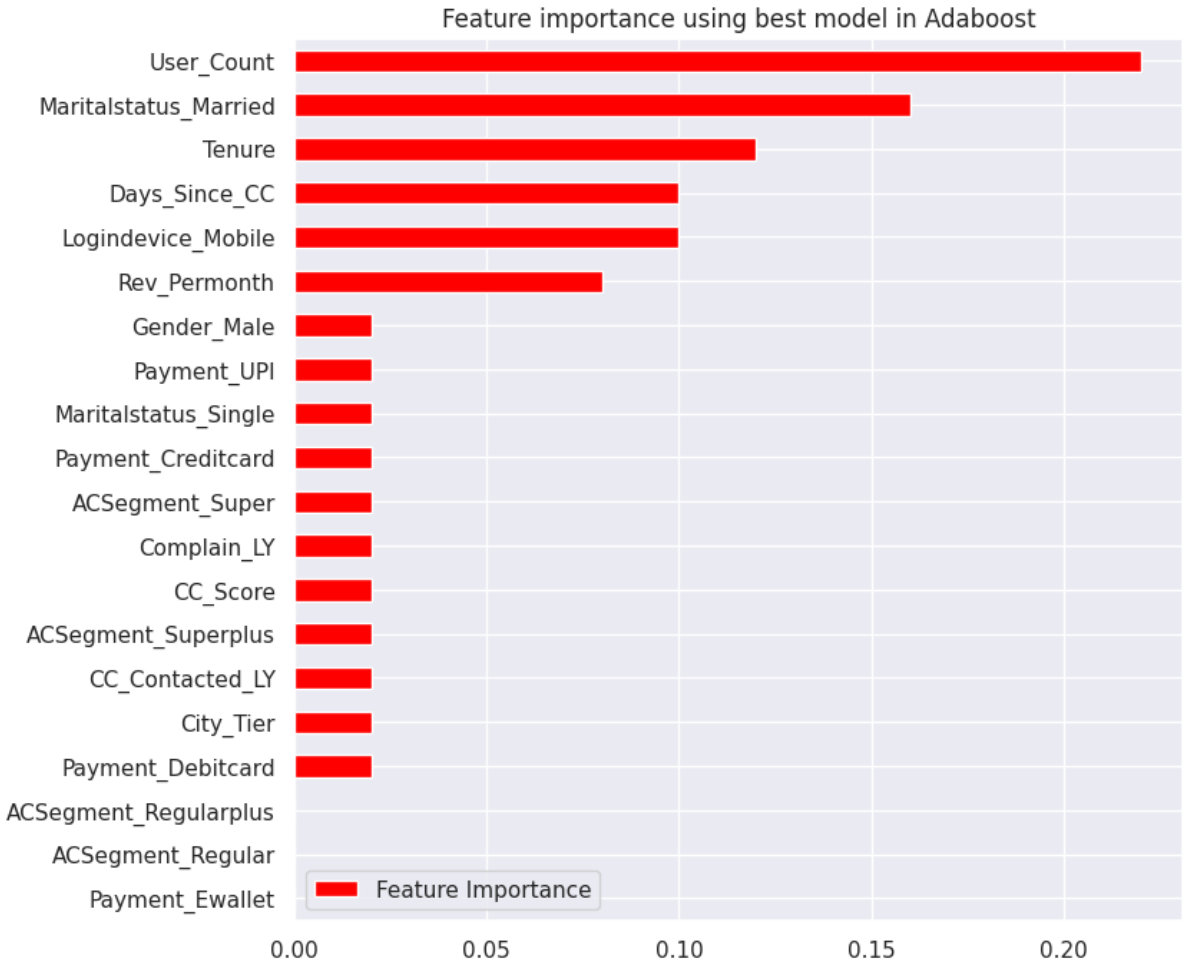


Fig. 17.3 Feature importance from best Adaboost model

Observations-

- Churned customers had contacted customer care more recently before churning than active customers. Per EDA, median days since last customer connect is higher for active customers. Churned customers had contacted customer care recently before churning.
- The next important parameter to predict customer churn per this model is the number of times customer care was contacted by the customer. Per EDA, the median and third quartile of number of times customer care was contacted previous year is higher for churned customers compared to active/current customers.

17. Gradient Boosting-

Gradient boost is an ensemble machine learning algorithm that trains underlying models in a gradual, additive and sequential manner.

17.1 Building Gradient Boosting Model on original dataset-

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

```
accuracy for training dataset: 0.9222278609489977
confusion matrix for training dataset
[[6378  177]
 [ 436  891]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      6555
           1       0.83      0.67      0.74      1327

    accuracy                           0.92      7882
   macro avg       0.89      0.82      0.85      7882
weighted avg       0.92      0.92      0.92      7882

accuracy score for testing dataset: 0.911190053285968
confusuon matrix for testing dataset
[[2730   79]
 [ 221  348]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2809
           1       0.81      0.61      0.70       569

    accuracy                           0.91      3378
   macro avg       0.87      0.79      0.82      3378
weighted avg       0.91      0.91      0.91      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.948
AUC score and ROC curve for testing dataset
AUC: 0.927
```

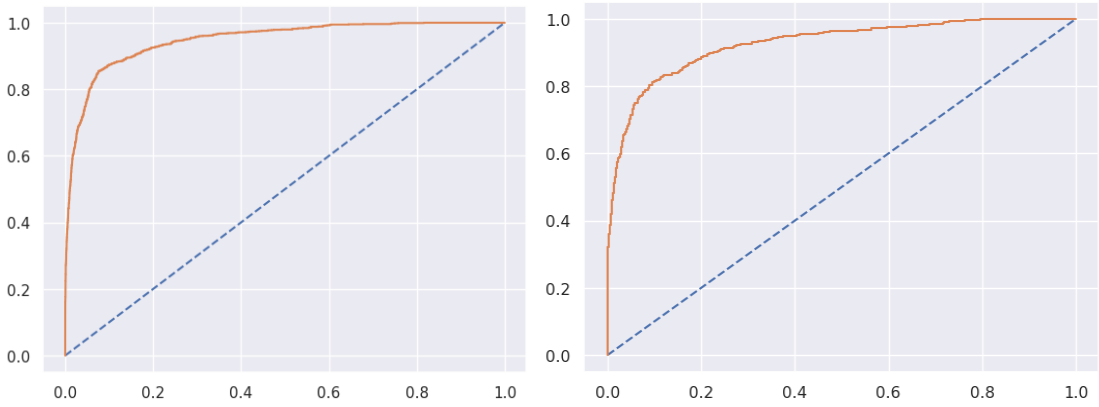


Fig.18.1 ROC Curve and AUC Score from Gradient-Boosting on Original Dataset

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

Cross validation scores for train dataset-

```
cross validation scores for train dataset
array([0.89607098, 0.91508238, 0.92385787, 0.91624365, 0.90736041,
       0.91497462, 0.89593909, 0.92005076, 0.90736041, 0.92005076])
```

Cross validation scores for test dataset-

```
cross validation scores for test dataset
array([0.9260355 , 0.89053254, 0.88757396, 0.91420118, 0.89053254,
       0.90236686, 0.90828402, 0.9112426 , 0.88130564, 0.91988131])
```

17.2 Gradient Boost best model with tuned hyperparameters-

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

```
accuracy for training dataset: 0.9399694889397406
confusion matrix for training dataset
[[6203  352]
 [ 435 6120]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      6555
           1       0.95      0.93      0.94      6555

    accuracy                           0.94     13110
   macro avg       0.94      0.94      0.94     13110
weighted avg       0.94      0.94      0.94     13110

accuracy score for testing dataset: 0.9073416222616933
confusuon matrix for testing dataset
[[2657  152]
 [ 161  408]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      2809
           1       0.73      0.72      0.72       569

    accuracy                           0.91      3378
   macro avg       0.84      0.83      0.83      3378
weighted avg       0.91      0.91      0.91      3378
```

The model is robust (no overfit or underfit) but the recall is quite poor in both train and test data. Tuning was done to see if performance can be improved on the model.

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

```
AUC score and ROC curve for training dataset
AUC: 0.984
AUC score and ROC curve for testing dataset
```

```
AUC: 0.919
```

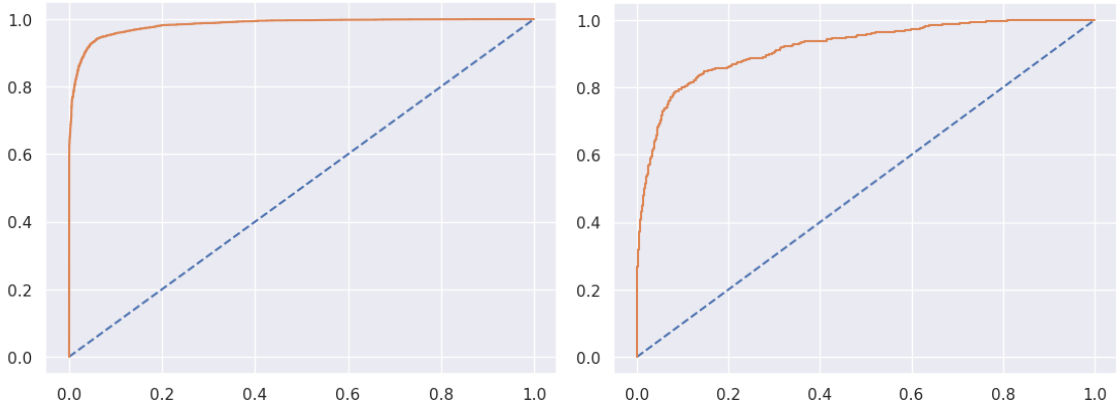


Fig.18.2 ROC Curve and AUC Score from Gradient-Boosting

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

Cross validation scores for train dataset-

```
cross validation scores for train dataset
array([0.83371472, 0.83142639, 0.95194508, 0.95728452, 0.96033562,
       0.94965675, 0.94736842, 0.95881007, 0.95499619, 0.96338673])
```

Cross validation scores for test dataset-

```
cross validation scores for test dataset
array([0.9260355 , 0.89053254, 0.88757396, 0.91420118, 0.89053254,
       0.90236686, 0.90828402, 0.9112426 , 0.88130564, 0.91988131])
```

Feature Importance for Gradient-Boosting-

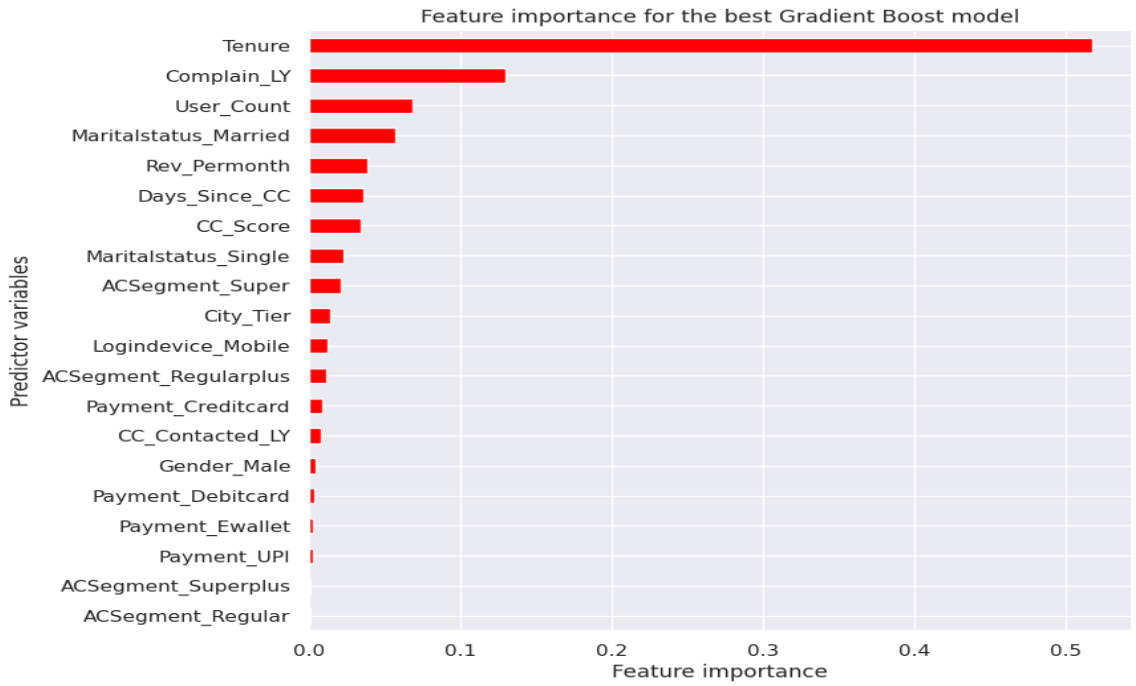


Fig.18.3 Feature Importance for the best Gradient Boost Model

18. Model Validation-

The given data was split into a train and test dataset in the ratio 70:30. The test dataset was held out and kept for validation purposes only.

- All models were trained only on the train dataset. The trained model was used to predict the train dataset target variable. The performance metrics such as Accuracy, F1-score, Precision, Recall, confusion matrix, ROC curve and AUC was observed and recorded on the train dataset.
- The trained model was then used to predict the target variable on the test dataset. All the above said performance metrics were observed and recorded for the test data performance as well.
- If train data seemed to be giving all 1's and the difference between train and test performances are not off by more than 10%, a validation of test scores was done by doing 5-fold and 10-fold cross validation on the entire dataset. The scores on this cross validation were compared to test dataset scores to ensure that model had not overfitted on train dataset.

18.2 Criteria for the best performing model-

Precision, Recall & F1-score for 1s: This is the case of class imbalance as the dataset has 16.8% churns. In this case study 'Precision' and 'Recall' of class 1 or the minority class is most important. Combining these 2 metrics, F1-score for class 1 is also used in the comparison.

The following table shows performance of the best model from each algorithm built so far.

Secondary criteria-

Accuracy: This is a classification problem and the dataset has class imbalance. That is, the proportion of churn and non-churn customers are not equal. With imbalanced classes, it's easy to get a high accuracy without actually making useful predictions.

AUROC: In addition to the above metrics, the Area under curve of ROC curve is also used to evaluate model performance. An ROC curve (or receiver operating characteristic curve) is a plot that summarizes the performance of a binary classification model on the positive class.

18.3 Overall Model Building Comparison across parameters-

Training Dataset (70%)	Test Dataset(30%)

	Accuracy	Precision	Recall	F1	AUC	Accuracy	Precision	Recall	F1	AUC
LR	0.89	0.77	0.51	0.61	0.88	0.89	0.78	0.50	0.61	0.88
LR-CV	0.89	0.77	0.50	0.61	0.88	0.89	0.78	0.49	0.60	0.86
LR-SM	0.80	0.80	0.82	0.81	0.88	0.79	0.44	0.79	0.56	0.86
LDA	0.88	0.77	0.47	0.56	0.86	0.88	0.77	0.45	0.57	0.86
LDA-CV	0.88	0.77	0.47	0.59	0.87	0.88	0.77	0.45	0.59	0.85
LDA-SM	0.80	0.79	0.83	0.81	0.88	0.78	0.42	0.79	0.55	0.86
KNN	0.95	0.90	0.82	0.86	0.98	0.91	0.80	0.68	0.73	0.94
KNN-CV	1.0	1.0	1.0	1.0	1.0	0.95	0.90	0.82	0.86	0.97
KNN-SM	0.89	0.83	0.99	0.90	0.98	0.80	0.45	0.91	0.60	0.92
RF	1.0	1.0	0.98	1.0	0.82	0.97	0.97	0.87	0.92	0.80
RF - SM	1.0	1.0	0.97	1.0	0.84	0.97	0.97	0.88	0.92	0.82
NB	0.78	0.42	0.72	0.53	0.82	0.78	0.41	0.72	0.52	0.81
NB- SM	0.73	0.69	0.87	0.77	0.83	0.63	0.30	0.84	0.44	0.80
Bagging	0.99	1.0	0.98	0.99	1.0	0.96	0.96	0.88	0.92	0.99
Bagging - CV	0.99	1.0	1.0	1.0	1.0	0.96	0.93	0.86	0.89	0.98
Ada-Boost	0.89	0.74	0.61	0.67	0.91	0.89	0.75	0.60	0.67	0.990
Ada-Boost-CV	0.90	0.91	0.91	0.91	0.96	0.87	0.62	0.73	0.67	0.90
GB	0.92	0.83	0.67	0.74	0.94	0.91	0.81	0.61	0.70	0.92
GB - CV	0.93	0.95	0.93	0.94	0.98	0.90	0.73	0.72	0.72	0.91

Indicators/symbols for above tabular data-

- CV - indicates scores for models built on best params obtained from GridSearchCV with model name as prefix.
- SM - indicates scores for models built on a balanced dataset with model name as prefix.

19. Why KNN is the best Model?

- The **hyperparameter tuned KNN model has given the best performance in terms of test data precision, recall and f1-score which are the primary evaluation metrics**. The Accuracy and AUC are also highest amongst all the models. As the model looked like it overfit on the train dataset, a further 5-fold and 10-fold cross-validation was done on complete dataset in which the F1-score on all folds was either comparable or greater than test data f1-score.
- The difference between train data metrics and test data metrics is within 10%.

19.1 Interpretation of the most optimum model-

- From the above tabular representation of all the scores for training and testing dataset across various models we can conclude that **the KNN model with GridSearchCV of hyper-parameters is best optimized for the given dataset.** (highlighted in BOLD)
- There is marginal difference in accuracy for Logistic regression and LDA, but comparatively LDA had a little better performance than logistic regression.
- Model with bagging and boosting is also well optimized but the difference in accuracy for training and testing dataset is little on the higher side as compared to KNN.
- Other model's namely Naïve Bayes, LDA and RF worked well on training dataset but the accuracy came down when performed over testing dataset. Which indicates overfitting of data in that model.

19.2 Implication of final model on Business-

- Using the model built above, businesses can plan various strategies to make customers stick with them.
- They can roll out different Offers and discounts as family floaters.
- They can give regular discount coupons if paid by their e-wallet platform.

- Discount vouchers of other vendors or on the next bill can be provided based on minimum bill criteria.
- This model gives businesses an idea where they stand currently and what best they can do to improve on the same.

20. Final Interpretation/Recommendation-

Insights form Analysis-

- Business have visibility **in tier-1 city.**
- Most customers **rated "3" for the services provided** by the business.
- Most customers **rated "3" for the interactions** they have with customer care representatives.
- Transactions via **UPI and e-wallet are very low.**
- Maximum churn is from the account segment **"Regular+"**.
- Customers with marital status who are **"single" contribute max towards churn.**
- Any complaints raised in the **last 12 months doesn't show any impact toward churn.**
- Tenure and cashback are directly proportional to each other.
- Computer usage is more in tier **1 city followed by tier 3 and tier 2 city.**

Insights from Model Building-

- From the above tabular representation of all the scores for training and testing dataset across various models we can conclude that the **KNN model with Grid-search CV values of hyper-parameters is best optimized** for the given dataset. (highlighted in BOLD)
- There is marginal difference in accuracy for Logistic regression and LDA, but comparatively LDA had a little better performance than logistic regression.
- Model with bagging and boosting is also well optimized but the difference in accuracy for training and testing dataset is little on the higher side as compared to KNN.
- Other model's namely Naïve Bayes, LDA and SVM worked well on training dataset but the accuracy came down when performed over testing dataset. Which indicates overfitting of data in that model.
- All models built on balances dataset showed overfitting.

Recommendations-

- Business can introduce referral drive for existing customers to acquire new customers.
- Business needs to increase in visibility in Tier-2 city for better customer acquisition.
- Business can promote payment via standing instruction in a bank account or UPI which can be hassle free and safe for customers.
- Conducting satisfaction survey to understand change in customers behaviour.
- Business needs to make sure that all complaints and queries raised are resolved on time.
- Thanking customers with hand written notes on invoices will create a good will factor.

Appendix-

Codes Snapshot:-

```
#Import all necessary modules
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
```

```
df.info()
```

```
# creating ROC curve and getting AUC score for train data set
# predict probabilities
probs = best_grid.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
print("AUC score and ROC curve for training dataset")
auc = roc_auc_score(y_train, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
train_fpr, train_tpr, train_thresholds = roc_curve(y_train, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(train_fpr, train_tpr);
```

```
param_grid = {
    'n_neighbors': [5,7,9],
    'weights' : ['uniform','distance'],
    'metric' : ['euclidean', 'manhattan'],
    'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']

}

grid_search = GridSearchCV(estimator = knn, param_grid = param_grid, cv = 10, n_jobs=-1,scoring='f1')
```

```
# empty list that will hold accuracy scores
ac_f1scores = []
ac_scores = []

# perform accuracy metrics for values from 1,3,5....25
for k in range(1,25,2):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    knn_ytest_pred = knn.predict(X_test)
    # evaluate test f1_score, accuracy
    f1scores = f1_score(y_test, knn_ytest_pred)
    scores = knn.score(X_test, y_test)
    ac_f1scores.append(f1scores)
    ac_scores.append(scores)
```

```
# creating ROC curve and getting AUC score for test data set
# predict probabilities
probs = best_grid.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
print("AUC score and ROC curve for testing dataset")
auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr);
```

```
# creating dataframe for GridSearchCV
from sklearn.model_selection import GridSearchCV
param_grid = {
    'solver': ['svd', 'lsqr', 'eigen'],
    'tol' : [0.0001,0.0002,0.0003],
    'shrinkage' : ['auto', 'float', 'None'],

}

grid_search = GridSearchCV(estimator = lda, param_grid = param_grid, cv = 10, n_jobs=-1,scoring='f1')
```

```
# fitting model into training dataset
grid_search.fit(X_train, y_train)
```

```
# predict probabilities
probs = best_grid.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
print("AUC score and ROC curve for testing dataset")
auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr);
```

```
# getting classification report for train data set
print("Classification report for train dataset")
print(classification_report(y_train, ytrain_predict_lgcv))
```

```
# getting best estimators
best_grid = grid_search.best_estimator_
```

```
# predicting training and testing dataset
ytrain_predict_lgcv = best_grid.predict(X_train)
ytest_predict_lgcv = best_grid.predict(X_test)
```

```
#Accuracy - Training Data
print("Accuracy of training dataset after gridsearchCV:",best_grid.score(X_train, y_train))
```

```
# loading GridSearchCV and creating dataframe for parameters
from sklearn.model_selection import GridSearchCV
param_grid = {
    'solver': ['lbfgs','newton-cg', 'liblinear', 'sag', 'saga'],
    'penalty': ['l1','l2','none'],
    'tol':[0.0001,0.00001]
}

grid_search = GridSearchCV(estimator = lg, param_grid = param_grid, cv = 10, n_jobs=-1,scoring='f1')
```

```
CC_with_outlier['Clusters'] = clusters3
CC_with_outlier.to_csv("CC_with_outlier.csv", index=False)
```

```
df_total['Clusters'] = clusters3
df_total.to_csv('CC_without_outlier.csv', index=False)
```

```
from google.colab import files
files.download("CC_without_outlier.csv")
```

```
Outlier_treat_vars = ['Tenure','CC_Contacted_LY','Rev_Permonth','Days_Since_CC','Cashback']
```

```
plt.figure(figsize=(8,4))
sns.boxplot(data = X[Outlier_treat_vars]);
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
```

```
for i in num_cols:
    plt.figure(figsize=(6,8))
    sns.boxplot(data = df, y=i, x='Churn')
    plot_title = "Box plot of "+i+" with Churn"
    plt.title(plot_title)
    plt.show();
```

```
df1 = pd.DataFrame()
for i in cat_cols:
    plt.figure(figsize=(6,6))

    df1[i] = df_predictor[i].fillna("Missing")
    order = df1[i].value_counts(ascending=False).index
    ax = sns.countplot(x=df1[i],order=order)

    for p in ax.patches:
        height = p.get_height()
        ax.text(p.get_x()+p.get_width()/2., height+1, height, ha="center")

    plot_title = 'Count plot for ' + i
    plt.title(plot_title)
    plt.show();
```

```
plt.figure(figsize=(6,5))
ax = sns.countplot(x='Churn', data=df)

## To display on top of the bar
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2., height+1, height, ha="center")

## To display inside the bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=14)
plt.title("Count plot of target variable - Churn");
```

```
plt.figure(figsize=(10,6))
df_predictor[num_cols].boxplot()
plt.xticks(rotation=90)
plt.title("Box plot for all numeric continuous variables");
```

```
## Making a separate list for numeric and categorical columns
num_cols = ['Tenure', 'CC_Contacted_LY', 'rev_per_month','rev_growth_yoy','coupon_used_for_payment','Day_Since_CC_connect','cashback']
cat_cols = ['City_Tier','Payment','Gender','Service_Score','Account_user_count','account_segment','CC_Agent_Score','Marital_Status','Complain_ly','Login_device'
```

```
## Get upper bound and lower bound
Q1 = df_predictor[num_cols].quantile(0.25)
Q3 = df_predictor[num_cols].quantile(0.75)
IQR = Q3 - Q1
UL = Q3 + 1.5*IQR
LL = Q1 - 1.5*IQR
```


```
## Get upper bound and lower bound
Q1 = df_predictor[num_cols].quantile(0.25)
Q3 = df_predictor[num_cols].quantile(0.75)
IQR = Q3 - Q1
UL = Q3 + 1.5*IQR
LL = Q1 - 1.5*IQR
```

```
df_predictor = df.drop(['AccountID','Churn'],axis=1)

print("\nTotal number of nulls in all predictor fields ")
print(df_predictor.isnull().sum().sum())

print("\nTotal % nulls in all predictor fields ")
print(round(((df_predictor.isnull().sum().sum()/ df_predictor.size) * 100 ),2))

## Percentage nulls by column in predictor dataset
print("\n\nProportion of nulls by column - multiply by 100 for percentage")
print(df_predictor.isnull().sum().sort_values(ascending = False)/df_predictor.index.size)

## Rows that have more than 4 null fields
print("\n\nRows that have more than 4 null fields")
df_predictor_temp =df_predictor[df_predictor.isnull().sum(axis = 1) >= 4]
df_predictor_temp
```

```
[ ]  df.isnull().sum()
```

```
]  df.duplicated().sum()
```

……END OF REPORT……