



MBA Semester – IV
Research Project – Final Report

Name	Mahesh Narayan G
Project	Customer Churn
Date of Submission	01-04-2025



A study on “Customer Churn“

Research Project submitted to Jain Online (Deemed-to-be University)

In partial fulfillment of the requirements for the award of:

Master of Business Administration

Submitted by:

Mahesh Narayan G

USN:

222VMBR02617

Under the guidance of:

Hrushikesha Shastry B S
(Faculty-JAIN Online)

Jain Online (Deemed-to-be University)

Bangalore

DECLARATION

I, *Mahesh Narayan G*, hereby declare that the Research Project Report titled "*Customer Churn*" has been prepared by me under the guidance of the *Hrushikesh Shastry B S*. I declare that this Project work is towards the partial fulfillment of the University Regulations for the award of the degree of Master of Business Administration by Jain University, Bengaluru. I have undergone a project for a period of Eight Weeks. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Bangalore

Mahesh Narayan G

Date:01-04-2025

*Name of the Student
USN:222VMBR02617*

Content of Report

SL NO .	Content	Page Number
1	Introduction To Business Problem	3
2	EDA and Business Implications	4-14
3	Data Cleaning & Pre-Processing	14-23
4	Model Building & Improve Model Performance	24-31
5	Model Comparison	32
6	Model Validation	33-34
7	Final Interpretation & Recommendations	35-38
8	Appendix	38-43

Introduction of the Business Problem

Problem Statement: -

An E-commerce company is facing a lot of competition in the current market and it has become a challenge to retain the existing customers in the current competitive situation. Hence, the DTH 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 one account can have multiple customers. hence by losing one account, the company might be losing more than one customer.

We have been assigned to develop a churn prediction model for this company and provide business recommendations on the campaign. The model or campaign has to be unique and has to be sharp when offers are suggested. The offers suggested should have a win-win situation for company as well as customers so that company doesn't hit on revenue and on the other hand able to retain the customers.

Need of the study/project

This study/project is very essential for the business to plan for the future in terms of product designing, sales or in rolling out different offers for different segments of clients. The outcome of this project will give a clear understanding where the firm stands now and what capacity it holds in terms of taking risks. It will also denote what's the future perspective of the organization and how they can make it even better and can plan better for the same and can help them retain customers in a longer run.

Understanding business/social opportunity

This is a case study of an e-commerce 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 also based on the device they use (computer or mobile) moreover they earn cashbacks on bill payment.

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

We can conclude that a customer retained is a regular income for organization, a customer added is a new income for organization and a customer's loss will have a negative impact as a single account ID holds multiple number of customers i.e.; closure of one account ID means losing multiple customers.

It's a great opportunity for the company as it's a need of almost every individual of the family to have a DTH connection which in-turn also leads to increase and competition. Question arises how can a company create difference when compared to other competitors, 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.

2. EDA and Business Implication

Data Report

Dataset of problem: - Customer Churn Data 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 the 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

Data Ingestion: -

Loaded the required packages, set the work directory and load the datafile.

Data set has 11,260 observations and 19 variables (18 independent and 1 dependent or target variable).

AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	...
0	20000	1	4	3	6	1	1	3	3.0	1 ...
1	20001	1	0	1	8	2	2	3	4.0	2 ...
2	20002	1	0	1	30	1	2	2	4.0	2 ...
3	20003	1	0	3	15	1	2	2	4.0	1 ...
4	20004	1	0	1	12	3	2	2	3.0	2 ...

Table 1 – glimpse of data-frame head with top 5 rows

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

- Data has been collected for 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 individuals details as well.
- Data is a mixture of categorical as well as continuous variables.

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

- Data has 11,260 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

```

Table 2 :- Dataset Information

The shape of dataset is : (11260, 19)

Fig 1:- Shape of dataset

- Describing data: - This shows description of variation in various statistical measurements across variables which denotes that each variable is unique and different.

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

Table 3: - Describing Dataset

- Except variables “AccountID”, “Churn”, “rev_growth_yoy” and “coupon_used_for_payment” all other variables have null values present.

```

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

```

Table 4: - Showing Null Values in Dataset

- Data has “NIL” duplicate observations.
- With the above understanding of data renaming of any of the variable is not required.

We can move towards the EDA part where we will understand the data a little better along with treating bad data, null values and outliers.

Exploratory data analysis

Univariate analysis (distribution and spread for every continuous attribute, distribution of data in categories for categorical ones)

Univariate Analysis: -

- The variable shows outlier in data, which needs to be treated in further steps.

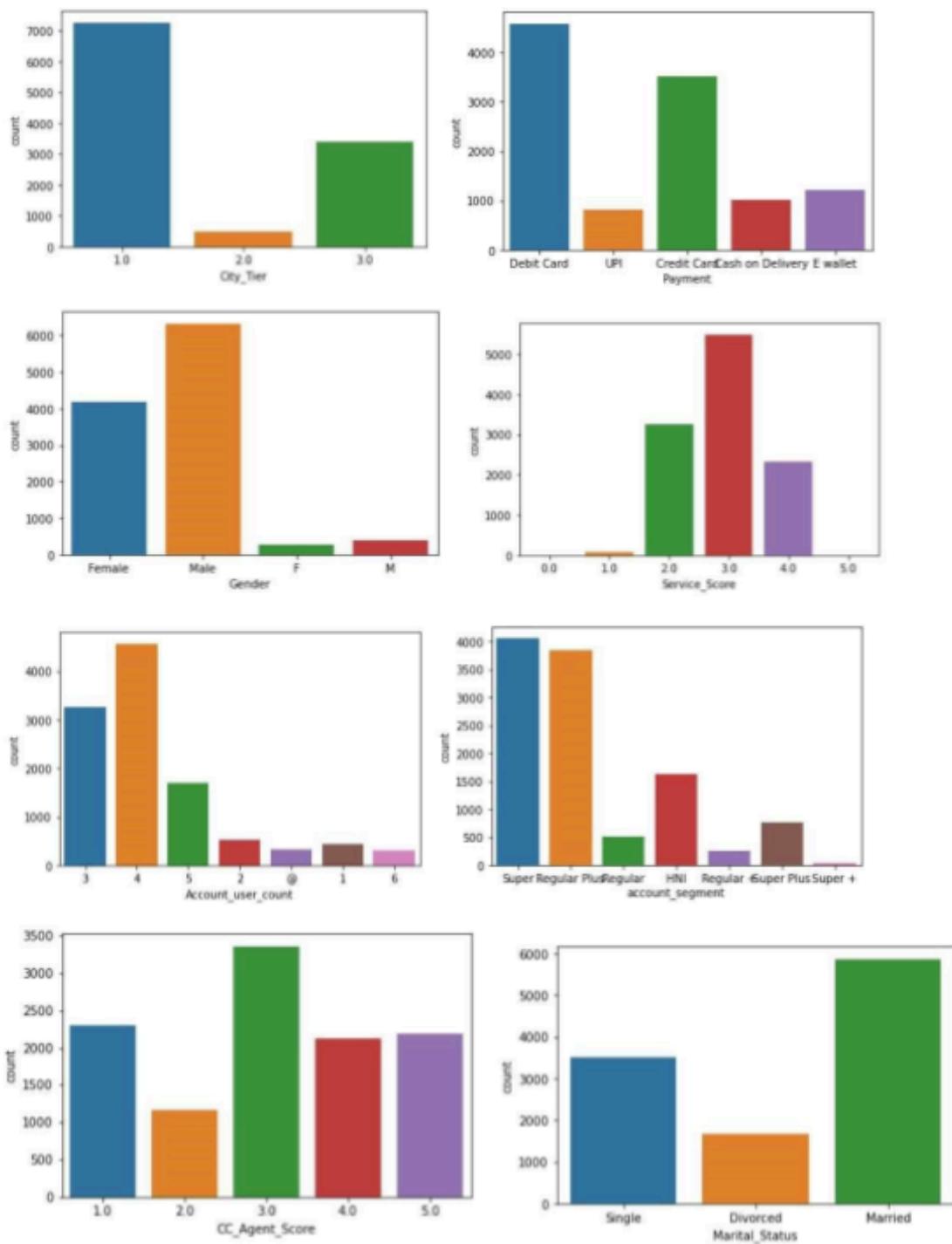
	outlier %
AccountID	0.00
Account_user_count	0.00
CC_Agent_Score	0.00
CC_Contacted_LY	0.37
Churn	16.84
City_Tier	0.00
Complain_ly	0.00
Day_Since_CC_connect	0.00
Gender	0.00
Login_device	0.00
Marital_Status	0.00
Payment	0.00
Service_Score	0.12
Tenure	0.00
account_segment	0.00
cashback	0.00
coupon_used_for_payment	0.00
rev_growth_yoy	0.00
rev_per_month	0.00

Table 5: - Showing Outliers in data

- None of the variables shows normal distribution and are skewed in nature.

	Kurtosis	Skewness
AccountID	-1.20	0.00
Churn	1.14	1.77
City_Tier	-1.40	0.74
CC_Contacted_LY	8.23	1.42
Service_Score	-0.67	0.00
CC_Agent_Score	-1.12	-0.14
Complain_ly	-1.10	0.95

Table 6:- Showing skewness and kurtosis



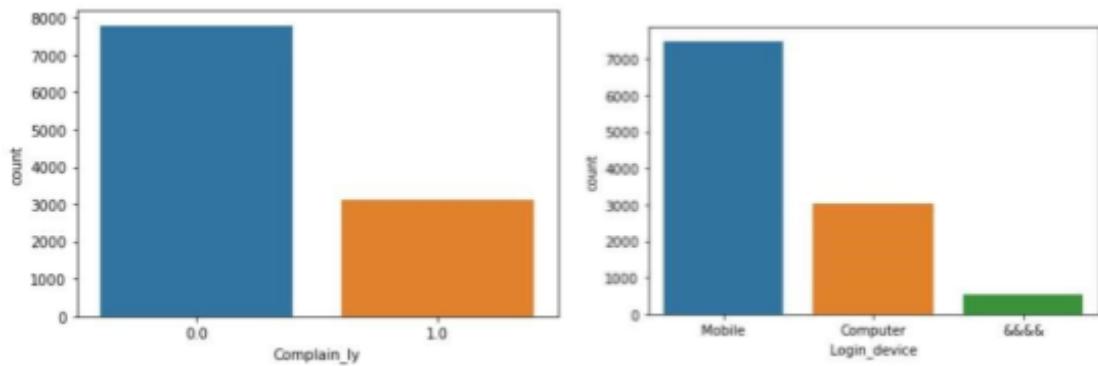
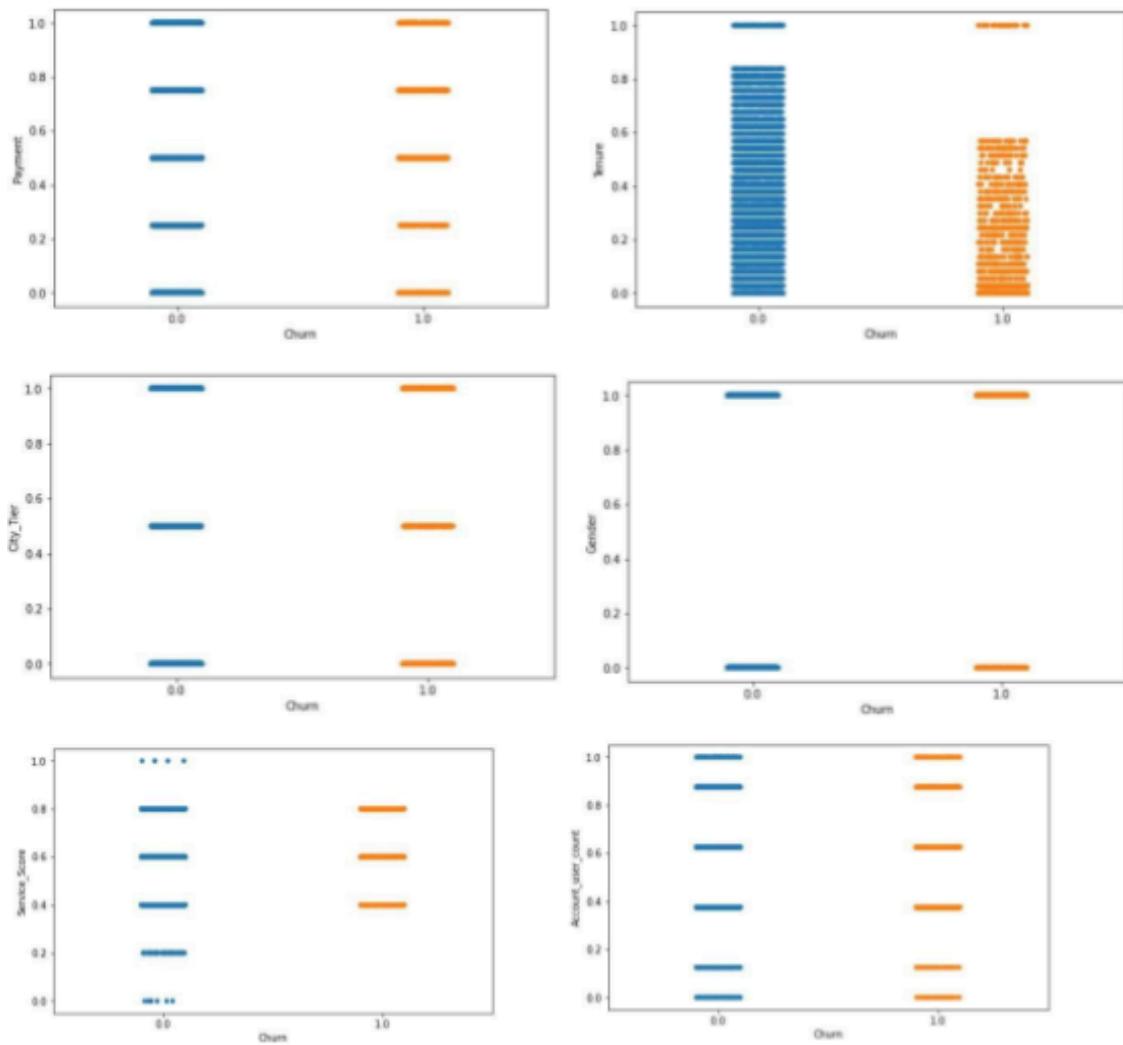


Fig 2: - Count plot of categorical variables



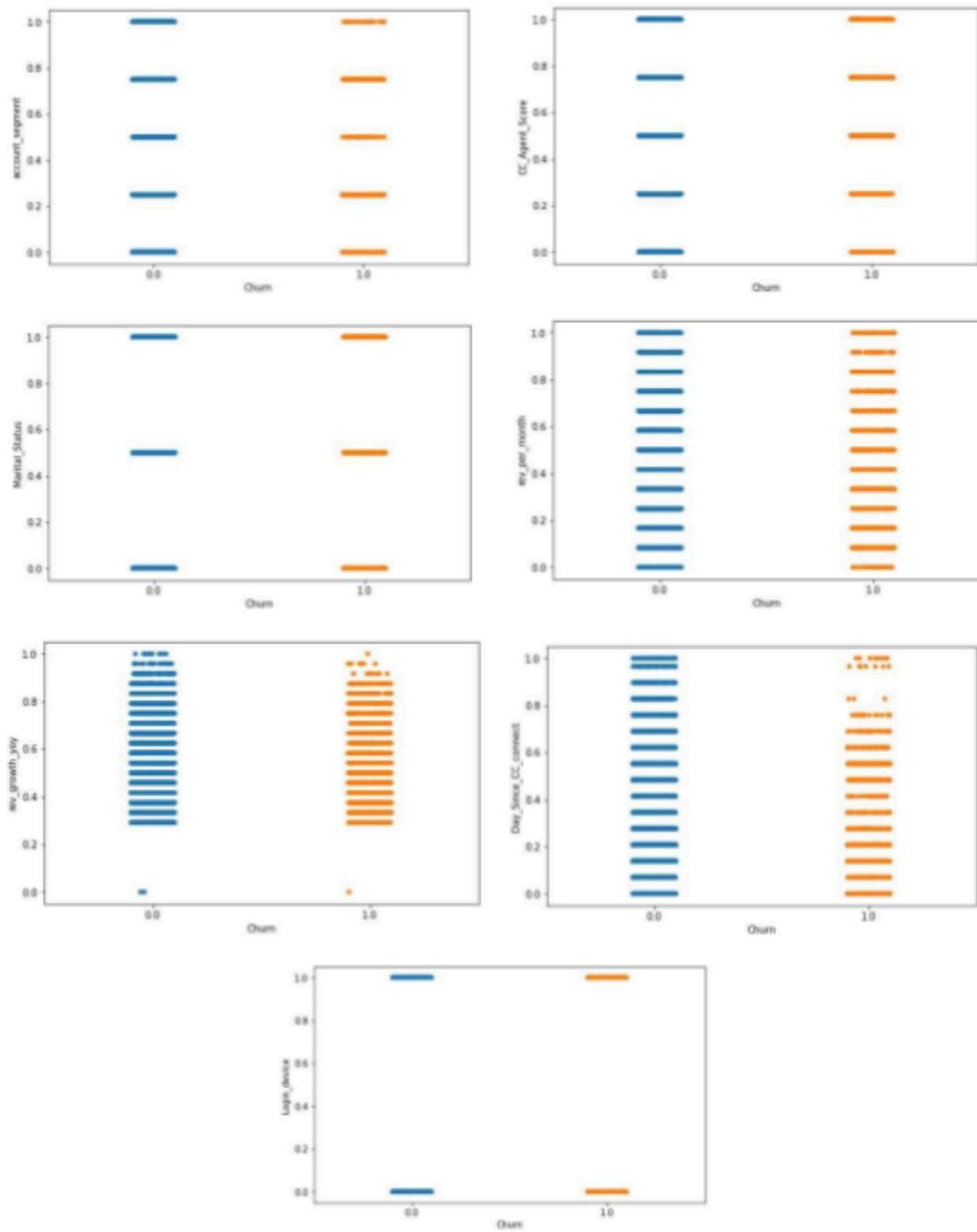


Fig 3: - Strip plot of across variables

Inferences from count plot: -

Maximum customers are from city tire type “1”, which indicates the high number of population density in this city type.

Maximum number of customers prefer debit and credit cards as their preferred mode of payment.

The ratio of male customers are higher when compared to females.

Average service score given by a customer for the service provided is around “3” which shows the area of improvements.

Most of the customers are into the “Super+” segment and the least number of customers are into the “Regular” segment.

Most of the customers availing services are “Married”.

Most of the customers prefer “Mobile” as the device to avail services.

Bi-variate Analysis: -

- Pair plot across all categorical data and its impact towards the target variable.

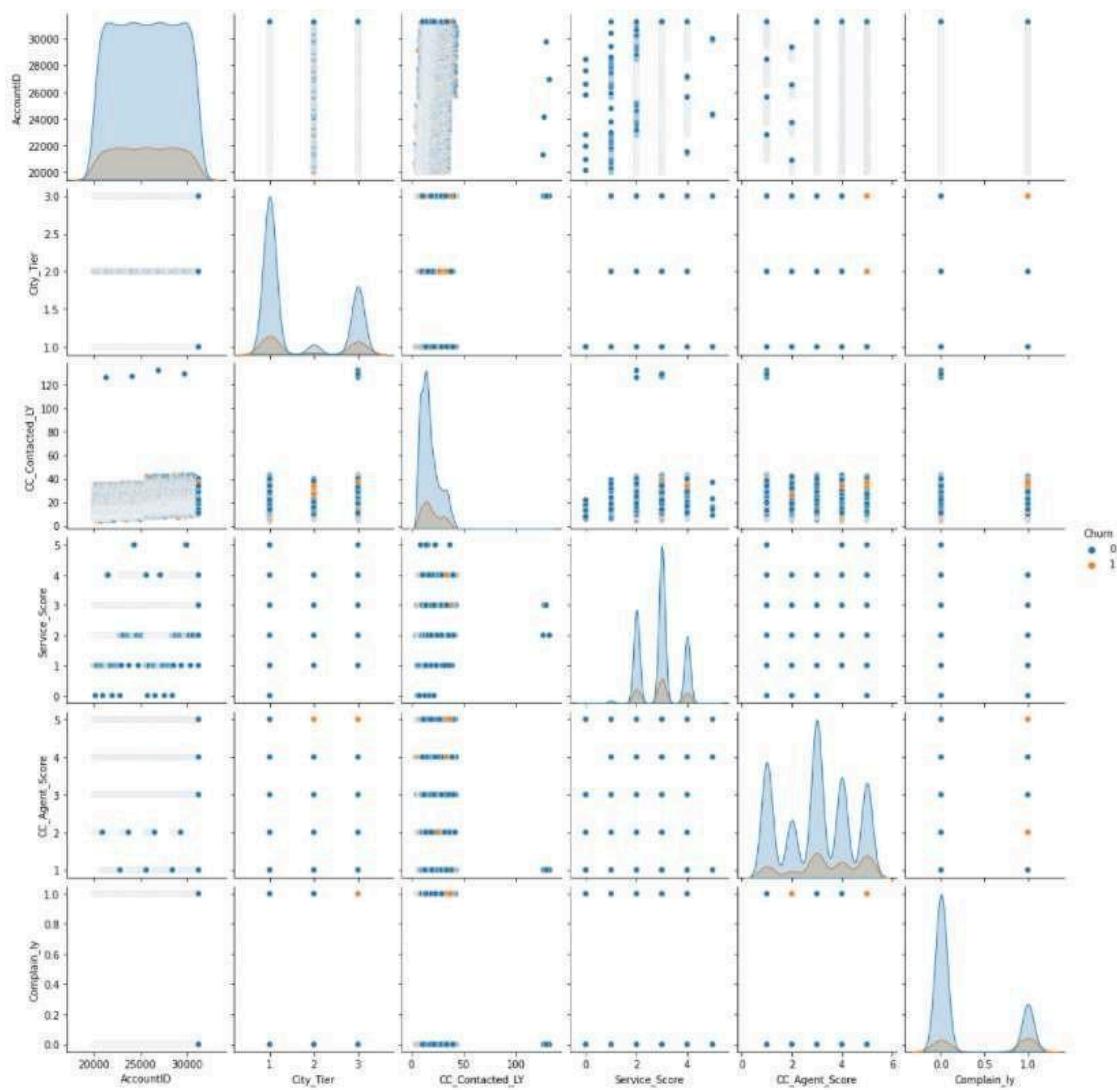
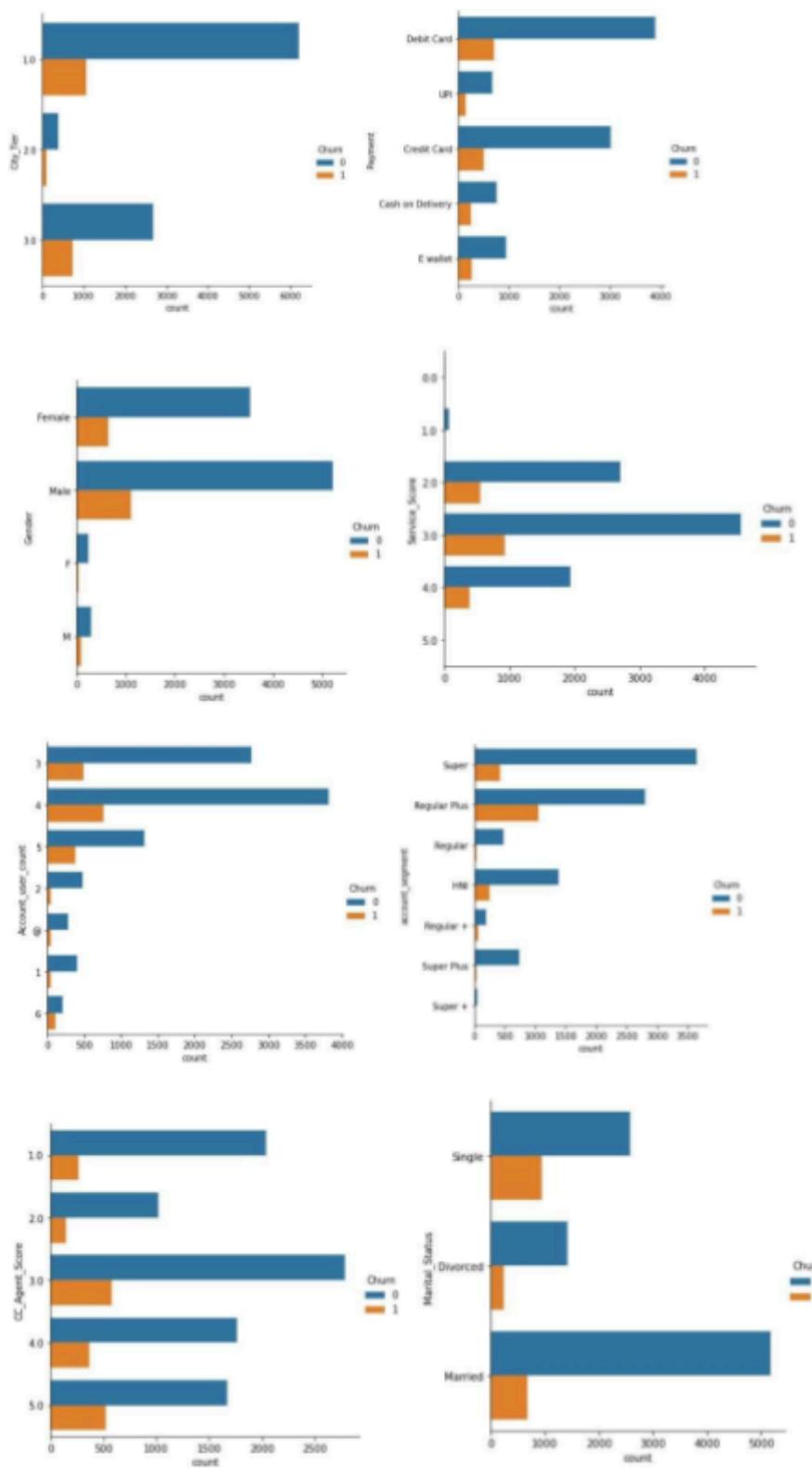


fig 4: - pair plot across categorical variables

The pair-plot shown above indicates that the independent variable are week or poor predictors of the target variable as we the density of independent variable overlaps with the density of target variable.



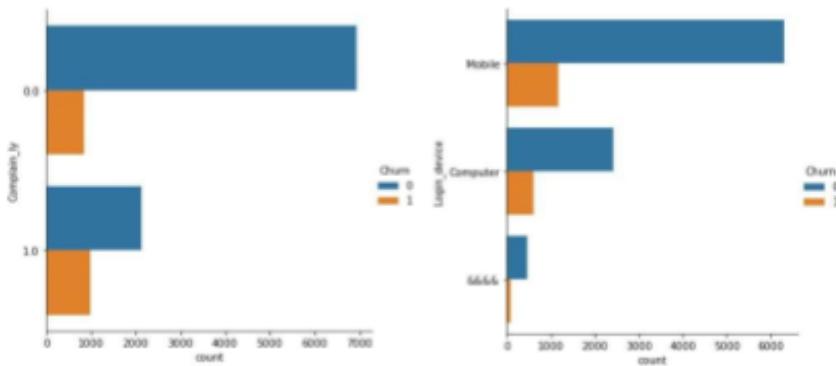


Fig 5: - Contribution of categorical variable towards churn

City_tier “1” has shown maximum churning when compared with “2” and “3”.

Customers with preferred modes of payment as “debit card” and “credit card” are more prone to churn.

Customers with gender as “Male” are showing more Churn ratio as compared to females.

Customers into the “Regular Plus” segment showing more churn.

Single customers tend to churn when compared with divorced and married.

Customers using the service over mobile shows more churn.

Correlation among variable:-

We have performed correlation between variables after treating bad data and missing values. We have also converted into integer data types to check on correlation as data type as categorical wont show in the pictures below.

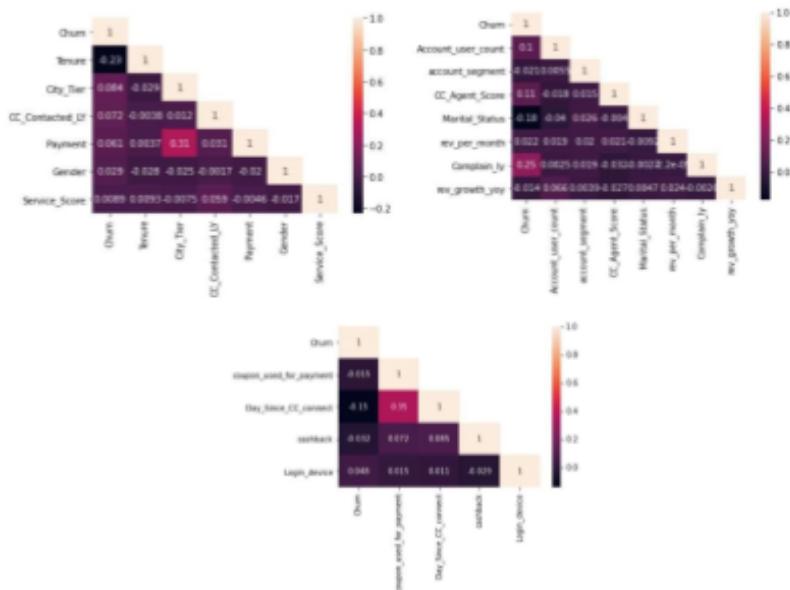


Fig 6: - Correlation among variables

Inferences from correlation: -

Variable “Tenure” shows high co-relation with Churn.

Variable “Marital Status” shows high co-relation with churn.

Variable “complain_ly” shows high- correlation with churn.

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. However, removing them will lead to 8 duplicate rows. Rest all the variables look important looking at the univariate and bi-variate analysis.

3. Data Cleaning and Preprocessing

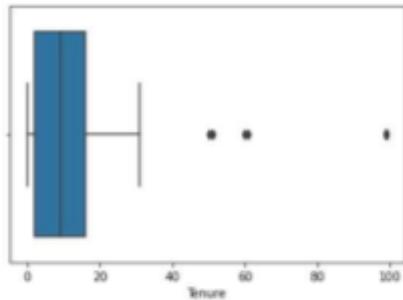
Outlier treatment:

This dataset is the mix of continuous as well as categorical variables. It doesn’t make any sense if we perform outlier treatment on categorical variables as each category denotes a type of customer. So, we are performing outlier treatment only for variables continuous in nature.

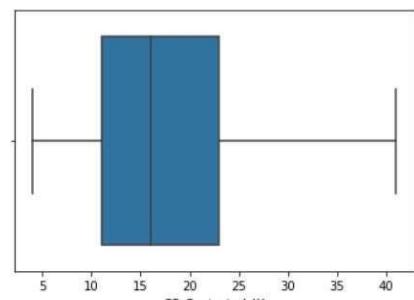
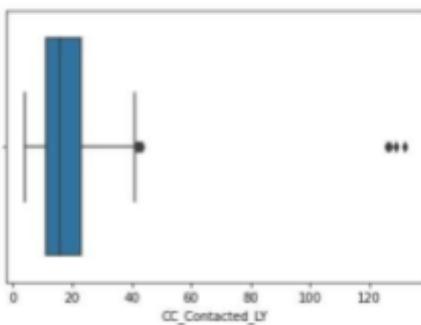
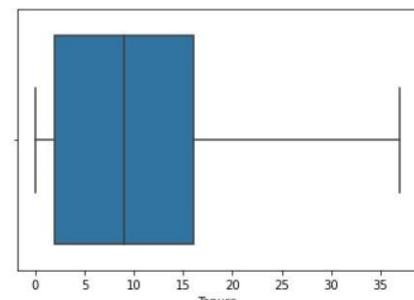
- We have 8 continuous variables in the dataset namely, “Tenure”, “CC_Contacted_LY”, “Account_user_count”, “cashback”, “rev_per_month”, “Day_Since_CC_connect”, “coupon_used_for_payment” and “rev_growth_yoy”.

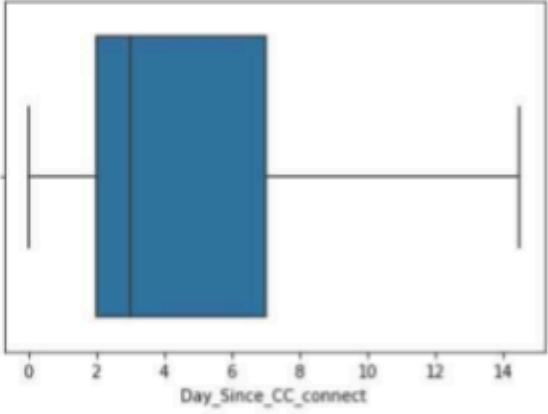
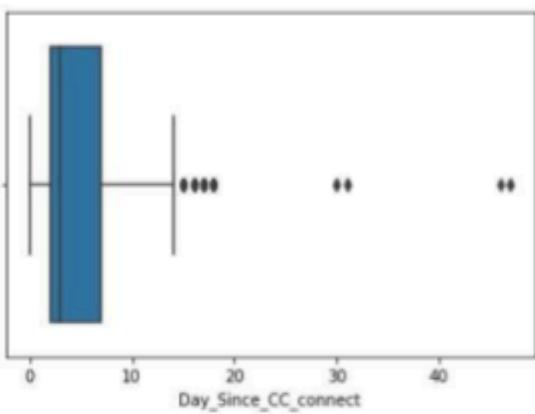
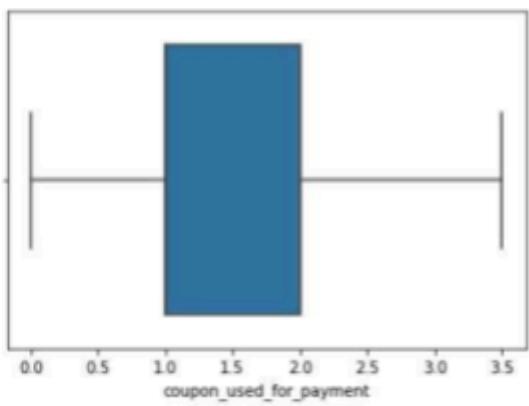
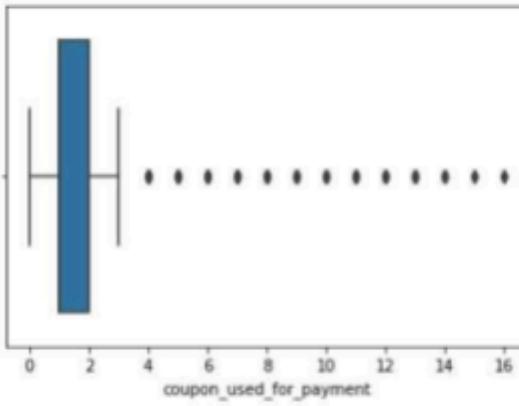
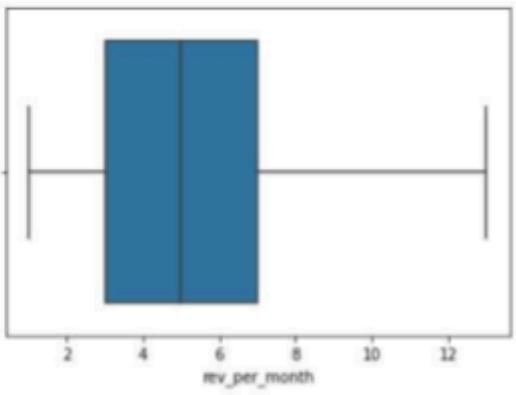
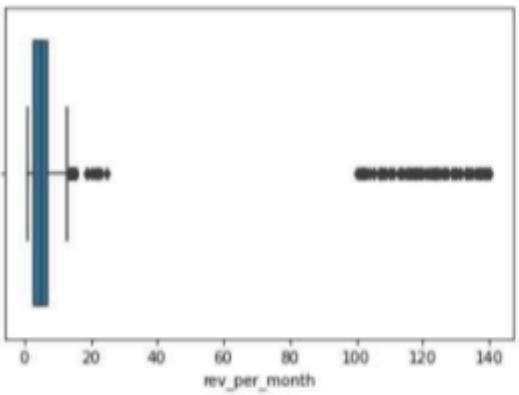
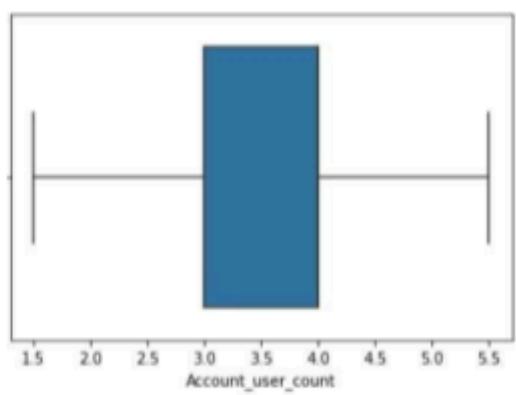
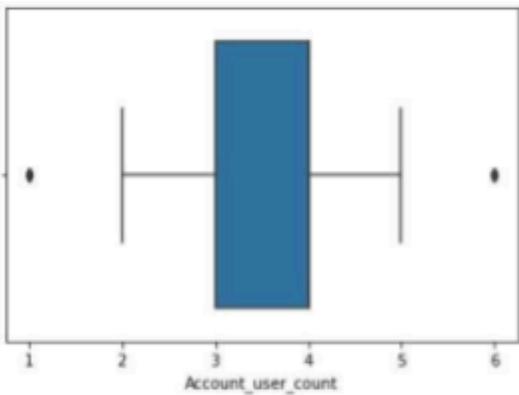
- We have used upper limit and lower limit to remove outliers. Below is the pictorial representation of variables before and after outlier treatment.

Before



After





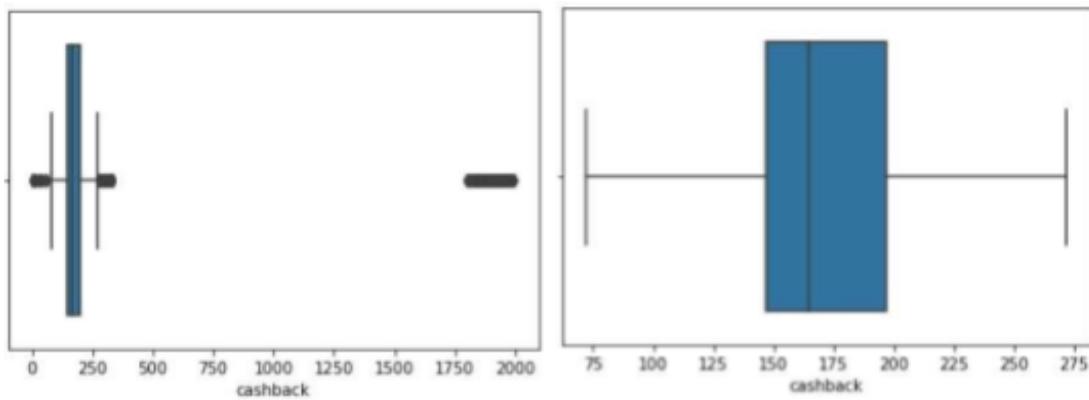


Fig 7: - Before and after outlier treatment

Missing Value treatment and variable transformation: -

- Out of 19 variables we have data anomalies present in 17 variables and null values in 15 variables.
- Using “Median” to impute null values where the variable is continuous in nature because Median is less prone to outliers when compared with mean.
- Using “Mode: to impute null values where variables are categorical in nature.
- We have treated null values variable by variable as each and every variable is unique in its nature.

Treating Variable “Tenure”

- We look at the unique observations in the variable and see that we have “#” and “nan” present in the data. Where “#” is an anomaly and “nan” represents null value.

```
array([4, 0, 2, 13, 11, '#', 9, 99, 19, 20, 14, 8, 26, 18, 5, 30, 7, 1,
       23, 3, 29, 6, 28, 24, 25, 16, 10, 15, 22, nan, 27, 12, 21, 17, 50,
       60, 31, 51, 61], dtype=object)
```

Fig 8: - before treatment

- Replacing “#” with “nan” and further we replace “nan” with the calculated median of the variable and now we don’t see any presence of bad data and null values.
- Converted data type to integer, because IDE has recognized it as object data type due presence of bad data.

```
<IntegerArray>
[ 4,  0,  2, 13, 11,  9, 99, 19, 20, 14,  8, 26, 18,  5, 30,  7,  1, 23,  3,
 29,  6, 28, 24, 25, 16, 10, 15, 22, 27, 12, 21, 17, 50, 60, 31, 51, 61]
Length: 37, dtype: Int64
```

Fig 9: - after treatment

Treating Variable “City_Tier”

- We look at the unique observations in the variable and see presence of null value as shown below.

```
array([ 3.,  1., nan,  2.])
```

Fig 10: - before treatment

- We are replacing “nan” with the calculated mode of the variable and now we don’t see any presence of null values.
- Converted data type to integer, because IDE has recognized it as object data type due presence of bad data.

```
array([3., 1., 2.])
```

Fig 11: - after treatment

Treating Variable “CC_Contacted_LY”

- We look at the unique observations in the variable and see the presence of null value as shown below.

```
array([ 6.,  8., 30., 15., 12., 22., 11.,  9., 31., 18., 13.,
       20., 29., 28., 26., 14., 10., 25., 27., 17., 23., 33.,
       19., 35., 24., 16., 32., 21., nan, 34., 5., 4., 126.,
       7., 36., 127., 42., 38., 37., 39., 40., 41., 132., 43.,
      129.])
```

Fig 12: - before treatment

- We are replacing “nan” with calculated Median of the variable and now we don’t see any presence of null values.
- Converted data type to integer, because IDE has recognized it as object data type due presence of bad data.

```
array([6, 8, 30, 15, 12, 22, 11, 9, 31, 18, 13, 20, 29, 28, 26, 14, 10,  
      25, 27, 17, 23, 33, 19, 35, 24, 16, 32, 21, 34, 5, 4, 41.0, 7, 36,  
      38, 37, 39, 40], dtype=object)
```

Fig 13: - after treatment

Treating Variable “Payment”

- We look at the unique observations in the variable and see the presence of null value as shown below.

```
array(['Debit Card', 'UPI', 'Credit Card', 'Cash on Delivery', 'E wallet',  
      nan], dtype=object)
```

Fig 14: - before treatment

- we are replacing “nan” with calculated Mode of the variable and now we don’t see any presence of null values.
- Also performed label encoding for the observations. Where 1 = Debit card, 2 = UPI, 3 = credit card, 4 = cash on delivery and 5 = e-wallet. Then converting them to integer data type as it will be used for further model building.

```
array(['1', '2', '3', '4', '5'])
```

Fig 15: - after treatment

Treating Variable “Gender”

- We look at the unique observations in the variable and see presence of null value and multiple abbreviations of the same observations as shown below.

```
array(['Female', 'Male', 'F', nan, 'M']).
```

Fig 16: - before treatment

- we are replacing “nan” with calculated Mode of the variable and now we don’t see any presence of null values.
- Also performed label encoding for the observations. Where 1 = Female card and 2 = Male. Then converting them to integer data type as it will be used for further model building.

```
array(['1', '2'],
```

Fig 17: - after treatment

Treating Variable “Service_Score”

- We look at the unique observations in the variable and see presence of null value as shown below.

```
array([ 3.,  2.,  1., nan,  0.,  4.,  5.])
```

Fig 18: - before treatment

- we are replacing “nan” with calculated Mode of the variable and now we don’t see any presence of null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([3., 2., 1., 0., 4., 5.])
```

Fig 19: - after treatment

Treating Variable “Account_user_count”

- We look at the unique observations in the variable and see the presence of null value as well “@” as bad data, shown below.

```
array([3, 4, nan, 5, 2, '@', 1, 6])
```

Fig 20: - before treatment

- Replacing “@” with “nan” and further we replace “nan” with the calculated median of the variable and now we don’t see any presence of bad data and null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([3., 4., 5., 2., 1., 6.])
```

Fig 21: - after treatment

Treating Variable “account_segment”

- We look at the unique observations in the variable and see presence of null value as well different denotations for the same type of observations, shown below.

```
array(['Super', 'Regular Plus', 'Regular', 'HNI', 'Regular +', nan,
       'Super Plus', 'Super +'], dtype=object)
```

Fig 22: - before treatment

- Replacing “nan” with calculated Mode of the variable and also labelled different account segments, where 1 = Super, 2 = Regular Plus, 3 = Regular, 4 = HNI and 5 = Super Plus and now we don’t see any presence of bad data and null values.
- Then converting them to integer data type as it will be used for further model building.

```
array(['1', '2', '3', '4', '5'])
```

Fig 23: - after treatment

Treating Variable “CC_Agent_Score”

- We look at the unique observations in the variable and see the presence of null value as shown below.

```
array([ 2.,  3.,  5.,  4., nan,  1.])
```

Fig 24: - before treatment

- Replacing “nan” with calculated Mode of the variable and now we don’t see any presence of bad data and null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([2., 3., 5., 4., 1.])
```

Fig 25: - after treatment

Treating Variable “Marital_Status”

- We look at the unique observations in the variable and see the presence of null value as shown below.

```
array(['Single', 'Divorced', 'Married', nan],
```

Fig 26: - before treatment

- Replacing “nan” with calculated Mode of the variable and also labelled the observations.

Where in 1 = Single, 2 = Divorced and 3 = Married and now we don't see any presence of bad data and null values.

- Then converting them to integer data type as it will be used for further model building.

```
array([1., 2., 3.])
```

Fig 27: - after treatment

Treating Variable “rev_per_month”

- We look at the unique observations in the variable and see presence of null value as well as presence of “+” which denoted bad data. shown below.

```
array([9, 7, 6, 8, 3, 2, 4, 10, 1, 5, '+', 130, nan, 19, 139, 102, 120,
       138, 127, 123, 124, 116, 21, 126, 134, 113, 114, 108, 140, 133,
       129, 107, 118, 11, 105, 20, 119, 121, 137, 110, 22, 101, 136, 125,
       14, 13, 12, 115, 23, 122, 117, 131, 104, 15, 25, 135, 111, 109,
       100, 103], dtype=object)
```

Fig 28: - before treatment

- Replacing “+” with “nan” and further we replace “nan” with the calculated median of the variable and now we don't see any presence of bad data and null values.

- Then converting them to integer data type as it will be used for further model building.

```
array([ 9.,  7.,  6.,  8.,  3.,  2.,  4., 10.,  1.,  5., 130.,
       19., 139., 102., 120., 138., 127., 123., 124., 116., 21., 126.,
       134., 113., 114., 108., 140., 133., 129., 107., 118., 11., 105.,
       20., 119., 121., 137., 110., 22., 101., 136., 125., 14., 13.,
       12., 115., 23., 122., 117., 131., 104., 15., 25., 135., 111.,
       109., 100., 103.])
```

Fig 29: - after treatment

Treating Variable “Complain_ly”

- We look at the unique observations in the variable and see the presence of null value as shown below.

```
array([ 1.,  0., nan])
```

Fig 30: - before treatment

- Replacing “nan” with calculated Mode of the variable and now we don’t see any presence of null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([1., 0.])
```

Fig 31: - after treatment

Treating Variable “rev_growth_yoy”

- We look at the unique observations in the variable and see the presence of “\$” which denotes bad data. shown below.

```
array([11, 15, 14, 23, 22, 16, 12, 13, 17, 18, 24, 19, 20, 21, 25, 26,  
'$', 4, 27, 28], dtype=object)
```

Fig 32: - before treatment

- Replacing “\$” with “nan” and further we replace “nan” with calculated median of the variable and now we don’t see any presence of bad data and null values. Then converting them to integer data type as it will be used for further model building.

```
array([11., 15., 14., 23., 22., 16., 12., 13., 17., 18., 24., 19., 20.,  
21., 25., 26., 4., 27., 28.])
```

Fig

33: - after treatment

Treating Variable “coupon_used_for_payment”

- We look at the unique observations in the variable and see presence of “\$”, “*” and “#” which denoted bad data. shown below.

```
array([1, 0, 4, 2, 9, 6, 11, 7, 12, 10, 5, 3, 13, 15, 8, '#', '$', 14,  
'*', 16], dtype=object)
```

Fig 34: - before treatment

- Replacing “\$”, “*” and “#” with “nan” and further we replace “nan” with the calculated median of the variable and now we don’t see any presence of bad data and null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([ 1.,  0.,  4.,  2.,  9.,  6., 11.,  7., 12., 10.,  5.,  3., 13.,  
15.,  8., 14., 16.])
```

Fig 35: - after treatment

Treating Variable “Day_Since_CC_connect”

- We look at the unique observations in the variable and see the presence of “\$” which denotes bad data and also the presence of null values. shown below.

```
array([5, 0, 3, 7, 2, 1, 8, 6, 4, 15, nan, 11, 10, 9, 13, 12, 17, 16, 14,  
      30, '$', 46, 18, 31, 47], dtype=object)
```

Fig 36: - before treatment

- Replacing “\$” with “nan” and further we replace “nan” with the calculated median of the variable and now we don’t see any presence of bad data and null values.

- Then converting them to integer data type as it will be used for further model building.

```
array([ 5.,  0.,  3.,  7.,  2.,  1.,  8.,  6.,  4., 15., 11., 10.,  9.,  
      13., 12., 17., 16., 14., 30., 46., 18., 31., 47.])
```

Fig 37: - after treatment

Treating Variable “cashback”

- We look at the unique observations in the variable and see the presence of “\$” which denotes bad data and also the presence of null values. shown below.

```
array([159.93, 120.9, nan, ..., 227.36, '$', 91, 191.42])
```

Fig 38: - before treatment

Replacing “\$” with “nan” and further we replace “nan” with the calculated median of the variable and now we don’t see any presence of bad data and null values.

- Then converting them to integer data type as it will be used for further model building.

```
array([159., 120., 165., 134., 129., 139., 122., 126., 272., 153., 133.,  
      196., 157., 160., 149., 161., 203., 116., 206., 142., 172., 123.,  
      189., 143., 208., 127., 194., 125., 124., 186., 130., 150., 111.,  
      204., 131., 144., 195., 237., 267., 135., 152., 162., 168., 138.,  
      166., 176., 121., 148., 193., 184., 199., 224., 235., 188., 221.,  
      72., 179., 187., 132., 260., 137., 236., 164., 200., 209., 169.,  
      268., 155., 140., 234., 218., 219., 156., 163., 145., 154., 147.,  
      158., 114., 180., 136., 112., 220., 270., 175., 146., 174., 215.,  
      171., 182., 259., 225., 167., 128., 266., 141., 243., 183., 265.,  
      117., 241., 202., 190., 198., 232., 261., 118., 205., 254., 177.,  
      110., 211., 248., 217., 178., 151., 216., 271., 263., 207., 238.,  
      242., 197., 231., 239., 227., 233., 173., 119., 170., 185., 240.,  
      247., 192., 113., 264., 115., 212., 201., 252., 229., 181., 257.,  
      210., 269., 228., 214., 244., 253., 262., 191., 249., 213., 245.,  
      250., 223., 230., 222., 256., 258., 246., 226., 81., 251.])
```

Fig 39: - after treatment

Treating Variable “Login_device”

- We look at the unique observations in the variable and see the presence of “&&&&” which denotes bad data and also the presence of null values. shown below.

```
array(['Mobile', 'Computer', '&&&&', nan])
```

Fig 40: - before treatment

- Replacing “&&&&” with “nan” and further we replace “nan” with calculated Mode of the variable. Also, labelling the observations where 1= Mobile and 2 = Computer and now we don’t see any presence of bad data and null values.
- Then converting them to integer data type as it will be used for further model building.

```
array([1, 2])
```

Fig 41: - after treatment

Count of null values before and after treatment

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

Fig 42: - Before and after null value treatment

We see NIL null values across variables which indicate that the data is now cleaned and we can move further for data transformation if required.

Variable transformation:

- We see that the different variables have different dimensions. Like variable “Cashback” denotes currency whereas “CC_Agent_Score” denotes rating provided by the customers. Due to which they differ in their statistical rating as well.
- Scaling would be required for this data set which in turn will normalize the date and standard deviation will be close to “0”.
- Using MinMax scalar to perform normalization of data.

Standard Deviation Before and After Normalization: -

	Before	After
	standard deviation of variables	
AccountId	3250.626350	Churn 0.374223
Churn	0.374223	Tenure 0.240241
City_Tier	0.915015	city_Tier 0.456381
CC_Contacted_LY	8.853269	CC_Contacted_LY 0.231463
Service_Score	0.725584	Payment 0.344845
CC_Agent_Score	1.379772	Gender 0.488878
Complain_ly	0.451594	Service_Score 0.144495
		Account_user_count 0.231069
		account_segment 0.316751
		CC_Agent_Score 0.343166
		Marital_Status 0.447373
		rev_per_month 0.239968
		Complain_ly 0.447181
		rev_growth_yoy 0.156553
		coupon_used_for_payment 0.314928
		Day_Since_CC_connect 0.240931
		cashback 0.218847
		Login_device 0.442952

Fig 43: - Before and after Normalization

- We see that the standard deviation of variables are now close to “0”.
- Also converted variables to int data type which will help in further model building process.

Addition of new variables:

At the current stage we don't see to create any new variable as such. May be required at further stages of model building and can be created accordingly.

4. Model Building

From the above visual and non-visual analysis, we can conclude that it's a case of classification model, where the target variable needs to be classified into "Yes" or "No".

As a data analyst we have below algorithms to build the desired mechanism to predict if a given customer will churn or not: -

- Logistic Regression -- Logistic Regression is a "Supervised machine learning" algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable and the outcome is binary or dichotomous in nature. That means Logistic regression is usually used for Binary classification problems.
- Linear Discriminant Analysis (LDA) -- Linear Discriminant Analysis, or LDA for short, is a predictive modelling algorithm for multi-class classification. It can also be used as a dimensionality reduction technique, providing a projection of a training dataset that best separates the examples by their assigned class.
- KNN -- KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
- Naïve Bayes -- Naive Bayes is a kind of classifier which uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that a given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class.
- Bagging (Random Forest) -- Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once.
- Ada- Boosting -- The process of boosting involves improving the power of a machine learning program by adding more complex or capable algorithms. This process can reduce both bias and variance in machine learning, which helps to create more effective results.

- Gradient Boosting -- Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero
- Support Vector Machine (SVM) -- SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

Splitting Data into Train and Test Dataset: -

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

Below is the shape of Train and Test dataset: -

```
x_train (7882, 17)
x_test (3378, 17)
y_train (7882,)
y_test (3378,)
```

Fig 44: - Shape of training and test dataset

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. We have the count of “0” as 9364 and count of “1” as 1896.

AccountID	
Churn	
0	9364
1	1896

Table 7: - Imbalanced dataset

- 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 (can be changed later as instructed).

Before SMOTE

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

After SMOTE

```
X_train_res (13112, 17)
y_train_res (13112,)
```

Table 8: - Before and

after SMOTE

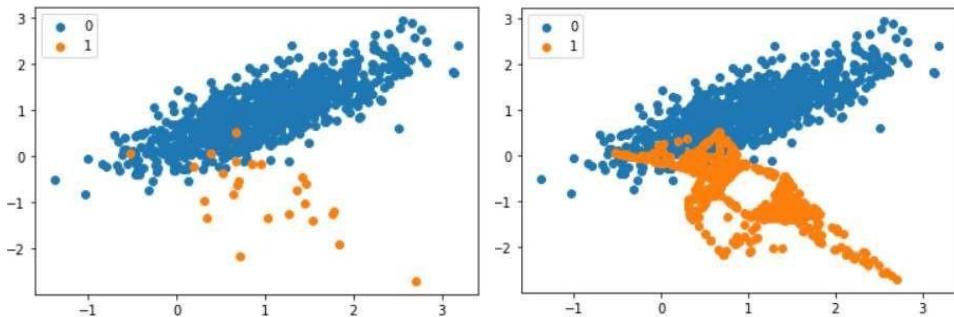


Fig 45: - Before and after SMOTE

- 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 outperforms over imbalanced data-set.

After building various models and analysing various parameters we conclude that “KNN” with default values outperforms all other models built. We have concluded this based on Accuracy, F1 score, Recall, Precision and AUC score.

BUILDING KNN MODEL: -

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: -

Accuracy of training dataset: 0.8572697284953058

accuracy for testing dataset 0.8404381290704559

Fig 46: - Accuracy Scores From KNN

Below are the confusion matrices obtained from this model: -

confusion matrix of training dataset [[6312 244] [881 445]]	confusion matrix for testing dataset [[2679 129] [410 160]]
--	--

Fig 47: - Confusion Matrix From KNN

Below are the classification Report obtained from this model: -

classification report of training dataset	classification report for testing dataset
precision recall f1-score support	precision recall f1-score support
0.0 0.88 0.96 0.92 6556	0.0 0.87 0.95 0.91 2808
1.0 0.65 0.34 0.44 1326	1.0 0.55 0.28 0.37 570
accuracy	accuracy
macro avg	macro avg
weighted avg	weighted avg

Fig 48: - Classification Report From KNN

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

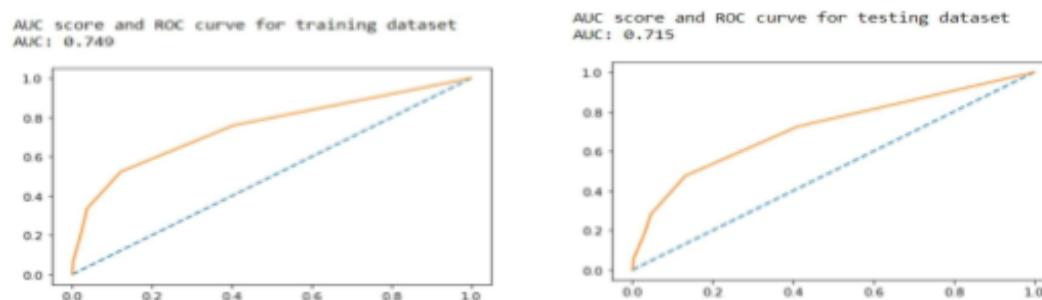


Fig 49: - ROC Curve and AUC Scores From KNN

Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset  
array([0.83523447, 0.82129278, 0.84898477, 0.84771574, 0.84390863,  
      0.8464467 , 0.84010152, 0.79568528, 0.84137056, 0.83883249])  
  
cross validation scores for test dataset  
array([0.83136095, 0.83136095, 0.83727811, 0.84023669, 0.82248521,  
      0.82544379, 0.82544379, 0.82544379, 0.81305638, 0.80118694])
```

Fig 50: - Cross Validation Scores From KNN

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

Effort to improve model performance.

Find the right value of n_neighbor: -

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

Below are the MSE scores: -

```
[0.24304322084073415,  
 0.17021906453522795,  
 0.1595618709295441,  
 0.16489046773238603,  
 0.16015393724097093,  
 0.16193013617525165,  
 0.16341030195381878,  
 0.16252220248667848,  
 0.16370633510953225,  
 0.16281823564239195]
```

Fig 51: - MSE Scores

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

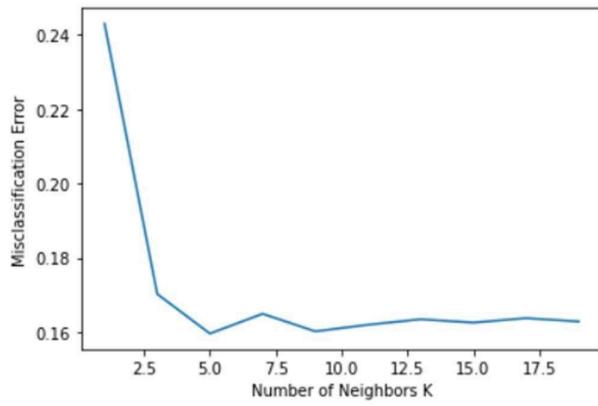


Fig 52: - Graphical Version Of MSE Score

From the above plotted graph we can see the n_neighbors with value “5” gives the least MSE score. With which we can proceed and build KNN model with n_neighbor value as “5” which is also the default n_neighbor. Hence, different model building with the correct number of n_neighbor is not required as it's the same as the default value of n_neighbor.

```
confusion matrix for training dataset      confusion matrix for testing dataset
array([[6342,  214],
       [ 903,  423]], dtype=int64)        array([[2694,  114],
                                                 [ 421,  149]], dtype=int64)
```

```
confusion matrix for training dataset      confusion matrix for testing dataset
array([[4370, 2186],
       [1566, 4990]], dtype=int64)        array([[1851,  957],
                                                 [ 168,  402]], dtype=int64)
```

Below are the classification reports obtained from balanced dataset: -

Classification report for train dataset				Classification report for test dataset					
	precision	recall	fi-score	support		precision	recall	fi-score	support
0.0	0.74	0.67	0.70	6556	0.0	0.92	0.66	0.77	2808
1.0	0.70	0.76	0.73	6556	1.0	0.30	0.71	0.42	570
accuracy			0.71	13112	accuracy			0.67	3378
macro avg	0.72	0.71	0.71	13112	macro avg	0.61	0.68	0.59	3378
weighted avg	0.72	0.71	0.71	13112	weighted avg	0.81	0.67	0.71	3378

Inference/Conclusion from KNN Model: -

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also infer 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 models 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.

Inferences on final 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 default 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.
- We also understand that the accuracy and other measuring parameters of a model can be improved by trying various other combinations of hyper-parameter. Model building is an iterative process. Model performance both on training and testing dataset can be improved.

Similar approaches like above have been used to create other models and below is the model comparison across parameters. However, we have showcased the result of KNN as we have concluded that it outperforms all other models.

Overall Model Building Comparison across parameters: -

	Training Dataset (70%)							Test Dataset (30%)								
	Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score	Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score
Logistic Regression	83.91	0.85	0.99	0.91	0.62	0.11	0.19	0.75	83.98	0.85	0.98	0.91	0.62	0.13	0.21	0.75
Logistic Regression - CV	83.92	0.85	0.99	0.91	0.62	0.12	0.2	0.75	83.98	0.85	0.98	0.91	0.62	0.13	0.21	0.752
Logistic Regression - SM	68.21	0.69	0.67	0.68	0.68	0.7	0.69	0.751	67.67	0.92	0.67	0.77	0.3	0.71	0.43	0.748
LDA	84.21	0.85	0.98	0.91	0.61	0.17	0.27	0.748	83.62	0.85	0.98	0.91	0.55	0.15	0.24	0.748
LDA - CV	84.16	0.85	0.98	0.91	0.6	0.18	0.28	0.748	83.54	0.85	0.97	0.91	0.54	0.16	0.25	0.747
LDA - SM	68.66	0.69	0.67	0.68	0.68	0.7	0.69	0.752	67.85	0.92	0.67	0.78	0.31	0.72	0.43	0.748
KNN	85.72	0.88	0.96	0.92	0.65	0.34	0.44	0.748	84.04	0.87	0.95	0.91	0.55	0.28	0.37	0.715
KNN - 5	Same As Default							Same As Default								
KNN - CV	85.82	0.88	0.97	0.92	0.66	0.32	0.43	0.757	84.16	0.86	0.96	0.91	0.57	0.26	0.36	0.72
KNN - SM	71.38	0.74	0.67	0.7	0.7	0.76	0.73	0.781	66.69	0.92	0.66	0.77	0.3	0.71	0.42	0.738
Naive Bayes	28.06	0.93	0.15	0.25	0.18	0.94	0.31	0.715	28.98	0.96	0.15	0.26	0.19	0.96	0.31	0.721
Naïve Bayes - SM	55.6	0.77	0.16	0.26	0.53	0.95	0.68	0.723	29.75	0.94	0.17	0.28	0.19	0.95	0.31	0.708
Bagging	86.24	0.87	0.98	0.92	0.73	0.29	0.41	0.833	84.28	0.86	0.97	0.91	0.59	0.22	0.32	0.794
Bagging - SM	74.5	0.75	0.74	0.74	0.74	0.75	0.75		71.81	0.92	0.72	0.81	0.34	0.69	0.45	0.785
Ada- Boosting	83.88	0.85	0.98	0.91	0.61	0.12	0.2	0.75	83.95	0.85	0.98	0.91	0.61	0.13	0.21	0.752
Ada- Boosting - SM	67.85	0.68	0.68	0.68	0.68	0.68	0.68	0.751	68.56	0.92	0.68	0.78	0.31	0.7	0.43	0.747
Gradient Boosting	84.77	0.85	0.98	0.91	0.69	0.17	0.27	0.782	84.13	0.85	0.98	0.91	0.61	0.16	0.25	0.774
Gradient Boosting - SM	71.77	0.71	0.74	0.72	0.73	0.7	0.71	0.785	71.78	0.92	0.73	0.81	0.33	0.67	0.44	0.769
SVM	85.29	0.86	0.99	0.92	0.74	0.19	0.31	0.75	84.31	0.85	0.98	0.91	0.64	0.16	0.26	0.754
SVM - CV	86.33	0.87	0.98	0.92	0.75	0.28	0.41	0.753	84.33	0.86	0.97	0.91	0.6	0.22	0.32	0.751
SVM - SM	74.09	0.74	0.75	0.74	0.74	0.74	0.74	0.753	71.99	0.92	0.73	0.81	0.34	0.68	0.45	0.75

Table 9: Comparison Across Various Models

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 balanced dataset with model name as prefix.
- KNN-5: - Indicates KNN model built with N_neighbors as “5”.

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 in minimum bill criteria.
- This model gives businesses an idea where they stand currently and what best they can do to improve on the same.

5. Model Validation

When it comes to model validation of a classification problem statement we cannot just rely on accuracy, we need to look at various other parameters like F1 score, Recall, precision, ROC curve and AUC score, along with a confusion matrix. The details of these parameters are described below: -

		Actual class	
		Positive	Negative
Predicted class	Positive	TP: True Positive	FP: False Positive (Type I Error)
	Negative	FN: False Negative (Type II Error)	TN: True Negative

Fig :-63 – A look of confusion matrix

Confusion Matrix:

Confusion Matrix usually causes a lot of confusion even in those who are using them regularly. Terms used in defining a confusion matrix are TP, TN, FP, and FN.

True Positive (TP): - The actual is positive in real and at the same time the prediction was classified correctly.

False Positive (FP): - The actual was actually negative but was falsely classified as positive.

True Negative: - The actuals were actually negative and were also classified as negative which is the right thing to do.

False Negative: - The actuals were actually positive but were falsely classified as negative.

Various Components Of Classification Report: -

Accuracy: This term tells us how many right classifications were made out of all the classifications.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$$

Precision: Out of all that were marked as positive, how many are actually truly positive.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Recall or Sensitivity: Out of all the actual real positive cases, how many were identified as positive.

$$\text{Recall} = \text{TP} / (\text{TN} + \text{FN})$$

Specificity: Out of all the real negative cases, how many were identified as negative.

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$$

F1-Score: As we saw above, sometimes we need to give weightage to FP and sometimes to FN. F1 score is a weighted average of Precision and Recall, which means there is equal importance given to FP and FN. This is a very useful metric compared to “Accuracy”. The problem with using accuracy is that if we have a highly imbalanced dataset for training (for example, a training dataset with 95% positive class and 5% negative class), the model will end up learning how to predict the positive class properly and will not learn how to identify the negative class. But the model will still have very high accuracy in the test dataset too as it will know how to identify the positives really well.

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Area Under Curve (AUC) and ROC Curve: AUC or Area Under Curve is used in conjunction with ROC Curve which is Receiver Operating Characteristics Curve. AUC is the area under the ROC Curve. So, let's first understand the ROC Curve.

A ROC Curve is drawn by plotting TPR or True Positive Rate or Recall or Sensitivity (which we saw above) in the y-axis against FPR or False Positive Rate in the x-axis.
 $\text{FPR} = 1 - \text{Specificity}$ (which we saw above).

$$\begin{aligned}\text{TPR} &= \text{TP} / (\text{TP} + \text{FN}) \\ \text{FPR} &= 1 - \text{TN} / (\text{TN} + \text{FP}) = \text{FP} / (\text{TN} + \text{FP})\end{aligned}$$

When we want to select the best model, we want a model that is closest to the perfect model. In other words, a model with AUC close to 1. When we say a model has a high AUC score, it means the model's ability to separate the classes is very high (high separability). This is a very important metric that should be checked while selecting a

6. Final interpretation / recommendation

Insights From Analysis: -

- Businesses have visibility in tier-1 cities.
- 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 default 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.
- We also understand that the accuracy and other measuring parameters of a model can be improved by trying various other combinations of hyper-parameters.

Model building is an iterative process. Model performance both on training and testing dataset can be improved

Recommendations: -

Four Stages of Churn Management



- Introduce pre-defined customer segmentation and according to customers' needs and usage.
- Acquiring customers through different strategies



- Delighting customers in order to increase customers loyalty as compared to competitors.



- Preventing Customers from attrition through analyzing various churn signals and triggers



- Focus on saving customers from leaving the firm through several campaigns.

Other Recommendations: -

- Business can introduce referral drive for existing customers to acquire new customers.

- Businesses can be in joint with other lifestyle vendors to provide vouchers to the new as well existing loyal customers.
- Business can internally bifurcate its customers based on spending patterns into deal seeker, tariff optimizer etc. and can have different acquisition strategy for each set of customers.
- Offering free cloud storage to loyal customers.
- Customized email response to priority customers basis segmentation for better customer interaction.
- Specialized team of customer service for Top notch customers to avoid waiting time and better customer experience and interaction.
- Understanding customers' profiles and sending small token of gift on special days.
- Thanking customers with hand written notes on invoices will create a good will factor.
- Follow-up on customers issues and taking regular feedback on the same.
- Conducting satisfaction surveys to understand change in customers behavior.
- Business needs to make sure that all complaints and queries raised are resolved on time.
- Businesses can promote using their own e-wallet as payment option by giving a certain discount over the bill.
- Businesses need to come up with subsidized offers for customers who are single as they show a high trend to churn.
- Business needs to introduce an all-in-one family plan with extra services, it will make accessibility easier for customers.
- Business needs to increase in visibility in Tier-2 cities for better customer acquisition.

- Businesses can promote payment via standing instruction in a bank account or UPI which can be hassle free and safe for customers.

Customer Segment Approach: -

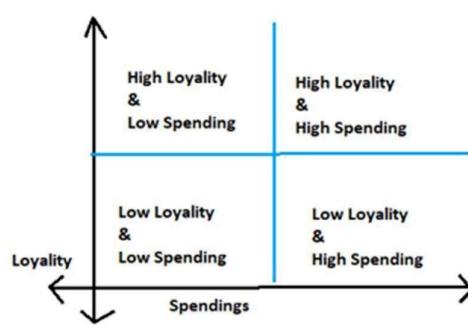


Fig 64: Graphical representation of customers division basis spending and loyalty

- Customers can divide into 4 sets as shown in figure.
- And at times business needs to take the harsh decision of letting go of the customers with “low on loyalty and Low on spending”.
- Customers under the set of “High Loyalty & High Spending” can be retained by delighting them with various offers.
- Customers under “High Loyalty & Low Spending” can be offered with a bundled family floater plan to increase their spending.
- The 4th quadrant can be the major area of observation for business where if customers are “low on loyalty but high on spending’s”, they can be retained by increasing the service level index and with proper follow-up on running offers and subscriptions.

Appendix

Codes Snapshot: -

```

# importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

# Loading data
churn = pd.read_excel(r'C:\Users\abhay\Downloads\Customer Churn Data.xlsx', sheet_name = 'Data for DSBA')

# checking info of data
churn.info()

# checking shape of dataset
print("The shape of dataset is :{}".format(churn.shape))

# checking for duplicate values
print("Number of duplicate rows:",churn.duplicated().sum())

# checking for skewness
churn.hist(figsize=(20,15));

# ktest check kurtosis and skewness of data
print("kurtosis and skewness of data is as below")
pd.DataFrame(data = [churn.kurtosis(), churn.skew()], index=['Kurtosis','Skewness']).T.round(2)

# plotting sns plot
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# pair plot to check on data distribution and co-linearity
sns.pairplot(churn, hue = 'Churn', diag_kind='kde')
plt.show()

```

```
#checking if the data is balanced or not
churn.groupby(["Churn"]).count()

sns.countplot(churn["City_Tier"]);

sns.countplot(churn["Payment"]);

sns.countplot(churn["Gender"]);

sns.countplot(churn["Service_Score"]);

sns.countplot(churn["Account_user_count"]);

sns.countplot(churn["account_segment"]);

sns.countplot(churn["CC_Agent_Score"]);

sns.countplot(churn["Marital_Status"]);

sns.countplot(churn["Complain_ly"]);

sns.countplot(churn["Login_device"]);

sns.catplot(y="City_Tier", hue="Churn", kind="count", data=churn)

sns.catplot(y="Payment", hue="Churn", kind="count", data=churn)

sns.catplot(y="Gender", hue="Churn", kind="count", data=churn)

sns.catplot(y="Service_Score", hue="Churn", kind="count", data=churn)

sns.catplot(y="Account_user_count", hue="Churn", kind="count", data=churn)

sns.catplot(y="account_segment", hue="Churn", kind="count", data=churn)

sns.catplot(y="CC_Agent_Score", hue="Churn", kind="count", data=churn)

sns.catplot(y="Marital_Status", hue="Churn", kind="count", data=churn)

sns.catplot(y="Complain_ly", hue="Churn", kind="count", data=churn)
```

```
sns.catplot(y="Login_device", hue="Churn", kind="count", data=churn)
```

```
# plotting heatmap of correlation
cor = churn.corr()
mask = np.array(cor)
mask[np.tril_indices_from(mask)] = False
fig,ax=plt.subplots()
fig.set_size_inches(10,10)
sns.heatmap(cor, mask=mask, vmax=1, square=True, annot=True)
plt.show()
```

```
churn['account_segment'] = churn['account_segment'].replace('Super','1')
churn['account_segment'] = churn['account_segment'].replace('Regular Plus','2')
churn['account_segment'] = churn['account_segment'].replace('Regular +','2')
churn['account_segment'] = churn['account_segment'].replace('Regular','3')
churn['account_segment'] = churn['account_segment'].replace('HNI','4')
churn['account_segment'] = churn['account_segment'].replace('Super Plus','5')
churn['account_segment'] = churn['account_segment'].replace('Super +','5')
```

```
# Lets check the percentage of outlier in each column
Q1 = churn.quantile(0.25)
Q3 = churn.quantile(0.75)
IQR = Q3 - Q1
pd.DataFrame(((churn < (Q1 - 1.5 * IQR)) | (churn > (Q3 + 1.5 * IQR))).sum()/churn.shape[0]*100,
             columns = ['outlier %'], index = None).round(2)
```

```
# lets check kurtosis and skewness of data
print("kurtosis and skewness of data is as below")
pd.DataFrame(data = [churn.kurtosis(), churn.skew()], index=['Kurtosis','Skewness']).T.round(2)
```

```
# Lets check the percentage of outlier in each column
Q1 = churn.quantile(0.25)
Q3 = churn.quantile(0.75)
IQR = Q3 - Q1
pd.DataFrame(((churn < (Q1 - 1.5 * IQR)) | (churn > (Q3 + 1.5 * IQR))).sum()/churn.shape[0]*100,
             columns = ['outlier %'], index = None).round(2)
```

```
from sklearn.preprocessing import MinMaxScaler

churn['scaled_Churn'] = MinMaxScaler().fit_transform(churn[['Churn']])
churn['scaled_Tenure'] = MinMaxScaler().fit_transform(churn[['Tenure']])
churn['scaled_City_Tier'] = MinMaxScaler().fit_transform(churn[['City_Tier']])
churn['scaled_CC_Contacted_LY'] = MinMaxScaler().fit_transform(churn[['CC_Contacted_LY']])
churn['scaled_Payment'] = MinMaxScaler().fit_transform(churn[['Payment']])
churn['scaled_Gender'] = MinMaxScaler().fit_transform(churn[['Gender']])
churn['scaled_Service_Score'] = MinMaxScaler().fit_transform(churn[['Service_Score']])
churn['scaled_Account_user_count'] = MinMaxScaler().fit_transform(churn[['Account_user_count']])
churn['scaled_account_segment'] = MinMaxScaler().fit_transform(churn[['account_segment']])
churn['scaled_CC_Agent_Score'] = MinMaxScaler().fit_transform(churn[['CC_Agent_Score']])
churn['scaled_Marital_Status'] = MinMaxScaler().fit_transform(churn[['Marital_Status']])
churn['scaled_rev_per_month'] = MinMaxScaler().fit_transform(churn[['rev_per_month']])
churn['scaled_Complain_ly'] = MinMaxScaler().fit_transform(churn[['Complain_ly']])
churn['scaled_rev_growth_yoy'] = MinMaxScaler().fit_transform(churn[['rev_growth_yoy']])
churn['scaled_coupon_used_for_payment'] = MinMaxScaler().fit_transform(churn[['coupon_used_for_payment']])
churn['scaled_Day_Since_CC_connect'] = MinMaxScaler().fit_transform(churn[['Day_Since_CC_connect']])
churn['scaled_cashback'] = MinMaxScaler().fit_transform(churn[['cashback']])
churn['scaled_Login_device'] = MinMaxScaler().fit_transform(churn[['Login_device']])
```

```
churn_scaled = pd.DataFrame({
    'Churn': churn['Scaled_Churn'] ,
    'Tenure': churn['Scaled_Tenure'] ,
    'City_Tier': churn['Scaled_City_Tier'] ,
    'CC_Contacted_LY': churn['Scaled_CC_Contacted_LY'] ,
    'Payment': churn['Scaled_Payment'] ,
    'Gender': churn['Scaled_Gender'] ,
    'Service_Score': churn['Scaled_Service_Score'] ,
    'Account_user_count': churn['Scaled_Account_user_count'] ,
    'account_segment': churn['Scaled_account_segment'] ,
    'CC_Agent_Score': churn['Scaled_CC_Agent_Score'] ,
    'Marital_Status': churn['Scaled_Marital_Status'] ,
    'rev_per_month': churn['Scaled_rev_per_month'] ,
    'Complain_ly': churn['Scaled_Complain_ly'] ,
    'rev_growth_yoy': churn['Scaled_rev_growth_yoy'] ,
    'coupon_used_for_payment': churn['Scaled_coupon_used_for_payment'] ,
    'Day_Since_CC_connect': churn['Scaled_Day_Since_CC_connect'] ,
    'cashback': churn['Scaled_cashback'] ,
    'Login_device': churn['Scaled_Login_device'] })
churn_scaled
```

```
# plotting sns plot
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# pair plot to check on data distribution and co-linearity
sns.pairplot(churn_scaled, hue = 'Churn', diag_kind='kde')
plt.show()
```

```
from numpy import where
import matplotlib.pyplot as plt

from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
```

```
X, y = make_classification(n_samples=1000, n_features=2, n_redundant=0, n_clusters_per_class=1, weights=[0.975], flip_y=0,
counter=Counter(y)
counter
```



```
from collections import Counter
for label, _ in counter.items():
    row_ix = where(y == label)[0]
    plt.scatter(X[row_ix, 0], X[row_ix, 1], label=str(label))
plt.legend()
plt.show()
```

Hyperparameters

```
# 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')
```

```
# 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')
```

```
# getting the ideal number of value of "N"
# empty list that will hold accuracy scores
ac_scores = []

# perform accuracy metrics for values from 1,3,5....19
for k in range(1,20,2):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    # evaluate test accuracy
    scores = knn.score(X_test, y_test)
    ac_scores.append(scores)

# changing to misclassification error
MCE = [1 - x for x in ac_scores]
MCE
```

```
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')
```

```
from sklearn.ensemble import BaggingClassifier
Bagging=BaggingClassifier(base_estimator=rf,random_state=1)
Bagging.fit(X_train, y_train)
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
adb = AdaBoostClassifier(random_state=1)
adb.fit(X_train,y_train)
```

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier(random_state=1)
gb.fit(X_train, y_train)
```

```
from sklearn import svm
SVM = svm.SVC()
SVM.fit(X_train, y_train)
```