Capstone Project – Final Report

Submitted By: **Naveen Lokesh**

1) Introduction

Problem Statement:

The data set belongs to a leading online E-Commerce company. An online retail (E commerce) company wants to know the customers who are going to churn, so accordingly they can approach customer to offer some promos.

Need for Solving it:

Customer retention: It is one of the very important aspect for business. Once a customer goes to an e-Commerce website and order something there is a possibility that the customer would come back and buy more things (If they are happy with the experience). Customer retention helps in generating higher customer lifetime value and hence the revenue. It is good to have new customers, but existing customers bring more revenue than the new ones.

There are many ways to achieve customer retention, but the most commonly used model is- Churn Model. Churn Model helps identifying customers who are most likely to switch to different e-Commerce website. Once identified the company can take actions in order to keep its existing customers. Now the question is, how does Churn model identify these customers? The model can be used to calculate the churn rate and depending on the nature of business, different metrics can be used. Few common metrics are - • Number of customers lost • Percent of customers lost

Business/Social Opportunity:

The Churn Model provides opportunities to the businesses in many ways. Few advantages of implementing Churn Model in e-Commerce are - • Churn rate can help identifying churn customers and accordingly businesses can run retention campaigns. • It will help in the keeping the revenue flowing. • Churn Model can help business to maintain customer lifetime value. • It helps businesses to track the progress. • The inputs received from Churn Model can be very helpful for BI activities.

-PTO

2) EDA and Business Implication

Uni-variate / Bi-variate / Multi-variate analysis to understand relationship b/w variables. How your analysis is impacting the business?

→The data was collected which shows the attributes for various customers and their churn behaviour

→The data for various variables were collected below is the understanding of the variables:

Variable	Discerption
CustomerID	Unique customer ID
Churn	Churn Flag
Tenure	Tenure of customer in organization
PreferredLoginDevice	Preferred login device of customer
CityTier	City tier
WarehouseToHome	Distance in between warehouse to home of customer
PreferredPaymentMode	Preferred payment method of customer
Gender	Gender of customer
HourSpendOnApp	Number of hours spend on mobile application or website
NumberOfDeviceRegistered	Total number of deceives is registered on particular customer
PreferedOrderCat	Preferred order category of customer in last month
SatisfactionScore	Satisfactory score of customer on service
MaritalStatus	Marital status of customer
NumberOfAddress	Total number of added added on particular customer
Complain	Any complaint has been raised in last month
OrderAmountHikeFromlastYear	Percentage increases in order from last year
CouponUsed	Total number of coupon has been used in last month
OrderCount	Total number of orders has been places in last month
DaySinceLastOrder	Day Since last order by customer
CashbackAmount	Average cashback in last month

Visual inspection of data:

→Number of rows: 5630
→Number of columns: 20

→Description of data:

	CustomerID	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
count	5630.000000	5630.000000	5366.000000	5630.000000	5379.000000	5375.000000	5630.000000	5630.000000	5630.000000	5630.000000	5365.000000	5374.000000	5372.000000	5323.000000	5630.000000
mean	52815.500000	0.168384	10.189899	1.654707	15.639896	2.931535	3.688988	3.066785	4.214032	0.284902	15.707922	1.751023	3.008004	4.543491	177.223030
std	1625.385339	0.374240	8.557241	0.915389	8.531475	0.721926	1.023999	1.380194	2.583586	0.451408	3.675485	1.894621	2.939680	3.654433	49.207036
min	50001.000000	0.000000	0.000000	1.000000	5.000000	0.000000	1.000000	1.000000	1.000000	0.000000	11.000000	0.000000	1.000000	0.000000	0.000000
25%	51408.250000	0.000000	2.000000	1.000000	9.000000	2.000000	3.000000	2.000000	2.000000	0.000000	13.000000	1.000000	1.000000	2.000000	145.770000
50%	52815.500000	0.000000	9.000000	1.000000	14.000000	3.000000	4.000000	3.000000	3.000000	0.000000	15.000000	1.000000	2.000000	3.000000	163.280000
75%	54222.750000	0.000000	16.000000	3.000000	20.000000	3.000000	4.000000	4.000000	6.000000	1.000000	18.000000	2.000000	3.000000	7.000000	196.392500
max	55630.000000	1.000000	61.000000	3.000000	127.000000	5.000000	6.000000	5.000000	22.000000	1.000000	26.000000	16.000000	16.000000	46.000000	324.990000

Understanding of attributes:

→Info of the attributes in the data:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5630 entries, 0 to 5629
Data columns (total 20 columns):
CustomerID                     5630 non-null int64
Churn                          5630 non-null int64
Tenure                         5366 non-null float64
PreferredLoginDevice           5630 non-null object
CityTier                       5630 non-null int64
WarehouseToHome                5379 non-null float64
PreferredPaymentMode           5630 non-null object
Gender                         5630 non-null object
HourSpendOnApp                 5375 non-null float64
NumberOfDeviceRegistered       5630 non-null int64
PreferedOrderCat               5630 non-null object
SatisfactionScore              5630 non-null int64
MaritalStatus                  5630 non-null object
NumberOfAddress                5630 non-null int64
Complain                       5630 non-null int64
OrderAmountHikeFromlastYear    5365 non-null float64
CouponUsed                     5374 non-null float64
OrderCount                     5372 non-null float64
DaySinceLastOrder              5323 non-null float64
CashbackAmount                 5630 non-null float64
dtypes: float64(8), int64(7), object(5)
memory usage: 879.8+ KB
```

→Checking for null values in the attributes:

```
CustomerID                       0
Churn                            0
Tenure                         264
PreferredLoginDevice             0
CityTier                         0
WarehouseToHome                251
PreferredPaymentMode             0
Gender                           0
HourSpendOnApp                 255
NumberOfDeviceRegistered         0
PreferedOrderCat                 0
SatisfactionScore                0
MaritalStatus                    0
NumberOfAddress                  0
Complain                         0
OrderAmountHikeFromlastYear    265
CouponUsed                     256
OrderCount                     258
DaySinceLastOrder              307
CashbackAmount                   0
dtype: int64
```

→There are no duplicate rows in the Dataset

Univariate Analysis of data:

→Minimum values in all the attributes:

```
CustomerID                          50001
Churn                                   0
Tenure                                  0
PreferredLoginDevice             Computer
CityTier                                1
WarehouseToHome                         5
PreferredPaymentMode                   CC
Gender                             Female
HourSpendOnApp                          0
NumberOfDeviceRegistered                1
PreferedOrderCat                  Fashion
SatisfactionScore                       1
MaritalStatus                    Divorced
NumberOfAddress                         1
Complain                                0
OrderAmountHikeFromlastYear            11
CouponUsed                              0
OrderCount                              1
DaySinceLastOrder                       0
CashbackAmount                          0
dtype: object
```

→Maximum values in all the attributes:

```
CustomerID                          55630
Churn                                   1
Tenure                                 61
PreferredLoginDevice                Phone
CityTier                                3
WarehouseToHome                       127
PreferredPaymentMode                  UPI
Gender                               Male
HourSpendOnApp                          5
NumberOfDeviceRegistered                6
PreferedOrderCat                   Others
SatisfactionScore                       5
MaritalStatus                      Single
NumberOfAddress                        22
Complain                                1
OrderAmountHikeFromlastYear            26
CouponUsed                             16
OrderCount                             16
DaySinceLastOrder                      46
CashbackAmount                     324.99
dtype: object
```

→IQR of the attributes in the dataset:

```
CustomerID                     2814.5000
Churn                             0.0000
Tenure                           14.0000
CityTier                          2.0000
WarehouseToHome                  11.0000
HourSpendOnApp                    1.0000
NumberOfDeviceRegistered          1.0000
SatisfactionScore                 2.0000
NumberOfAddress                   4.0000
Complain                          1.0000
OrderAmountHikeFromlastYear       5.0000
CouponUsed                        1.0000
OrderCount                        2.0000
DaySinceLastOrder                 5.0000
CashbackAmount                   50.6225
dtype: float64
```

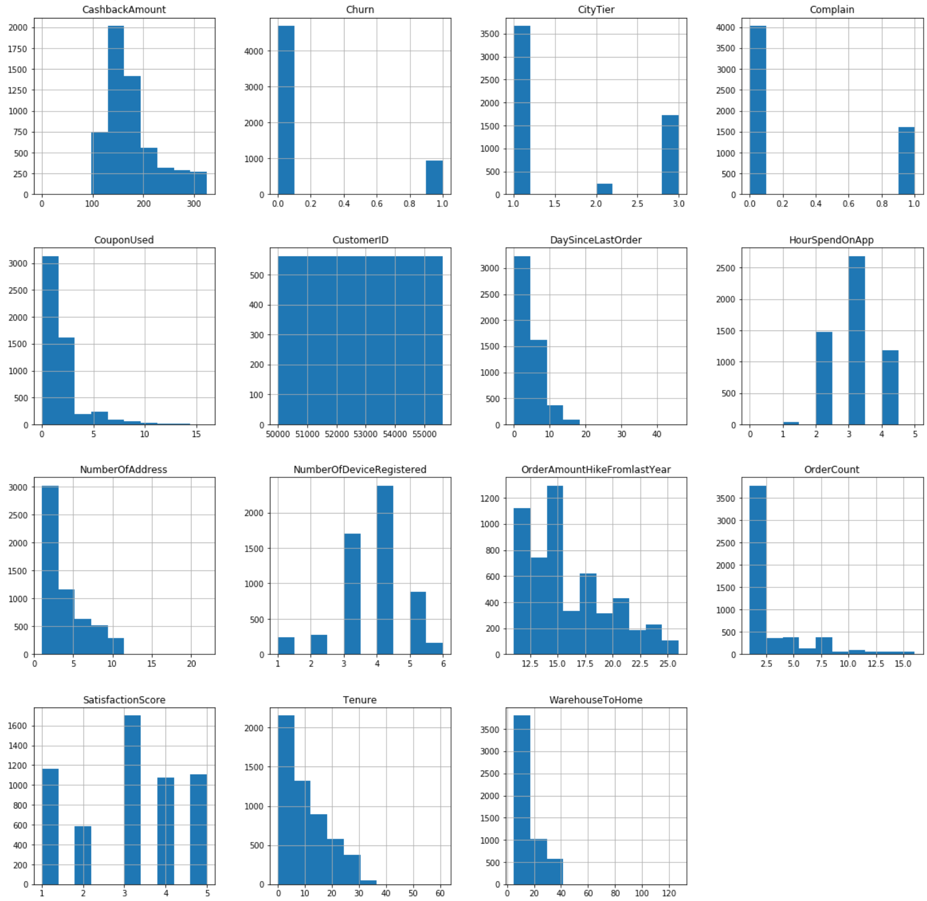
→Variance in all the attributes:

```
CustomerID                     2.641878e+06
Churn                          1.400555e-01
Tenure                         7.322637e+01
CityTier                       8.379375e-01
WarehouseToHome                7.278607e+01
HourSpendOnApp                 5.211769e-01
NumberOfDeviceRegistered       1.048573e+00
SatisfactionScore              1.904937e+00
NumberOfAddress                6.674914e+00
Complain                       2.037692e-01
OrderAmountHikeFromlastYear    1.350919e+01
CouponUsed                     3.589590e+00
OrderCount                     8.641716e+00
DaySinceLastOrder              1.335488e+01
CashbackAmount                 2.421332e+03
dtype: float64
```

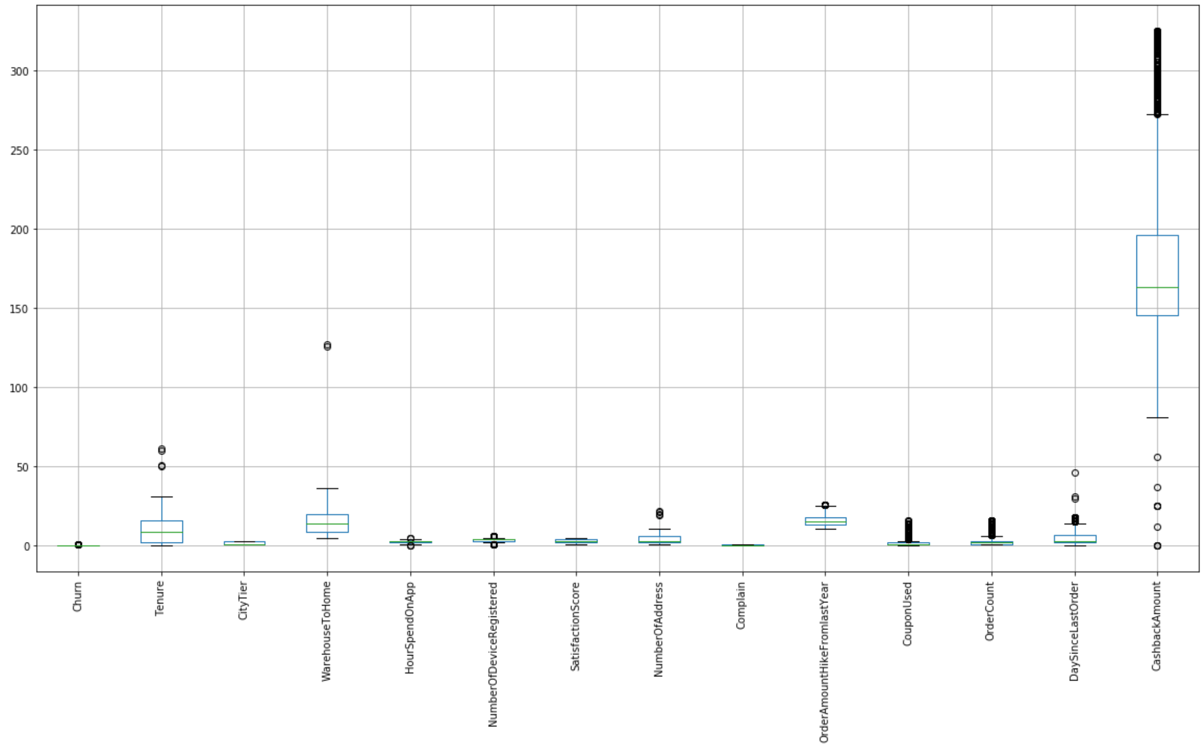
→Skewness in the dataset:

```
CustomerID                     0.000000
Churn                          1.772843
Tenure                         0.736513
CityTier                       0.735326
WarehouseToHome                1.619154
HourSpendOnApp                -0.027213
NumberOfDeviceRegistered      -0.396969
SatisfactionScore             -0.142626
NumberOfAddress                1.088639
Complain                       0.953347
OrderAmountHikeFromlastYear    0.790785
CouponUsed                     2.545653
OrderCount                     2.196414
DaySinceLastOrder              1.191000
CashbackAmount                 1.149846
dtype: float64
```

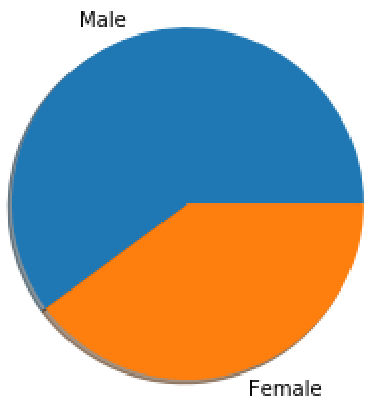
→Histogram of all the variables in the dataset:



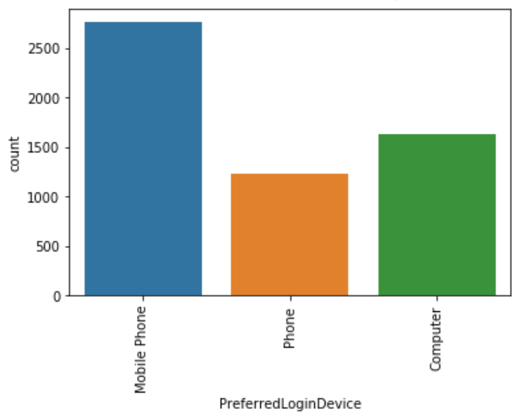
→Boxplot of the dataset to check outliers:



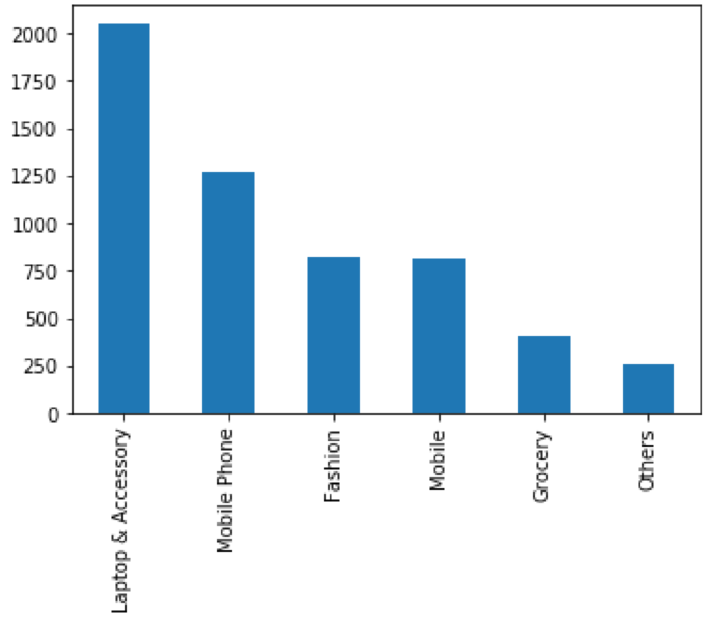
→Pie chart of the variable 'Gender':



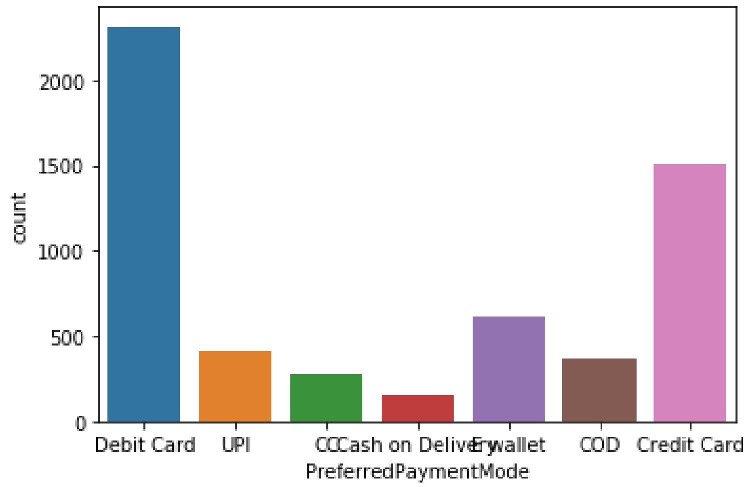
→Countplot of 'PreferredLoginDevice' variable:



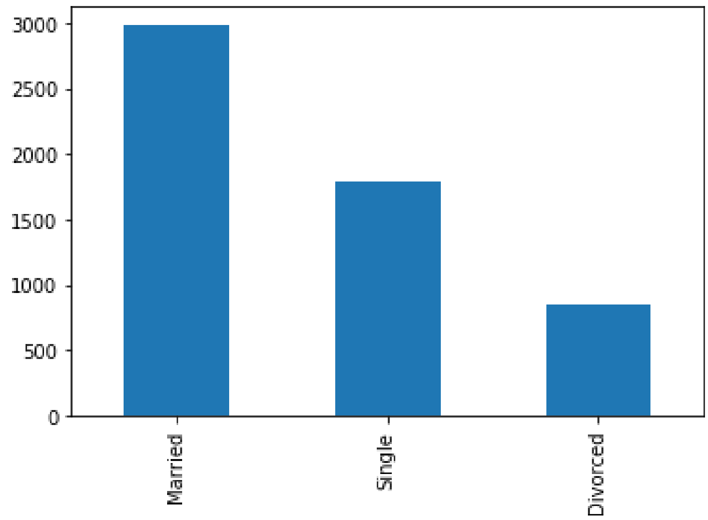
→Bar plot of 'PreferedOrderCat':



→Countplot of 'PreferredPaymentMode variable:

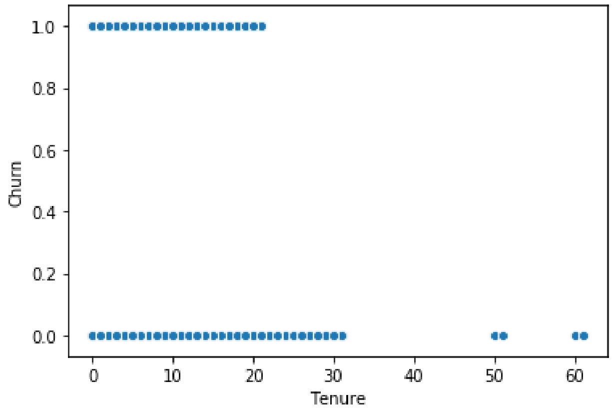


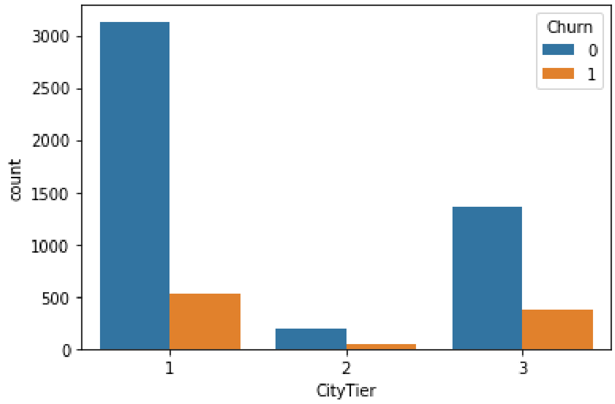
→Bar plot of MaritialStatus variable:

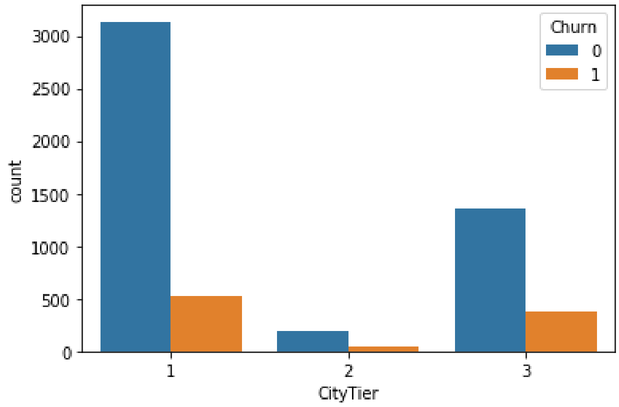


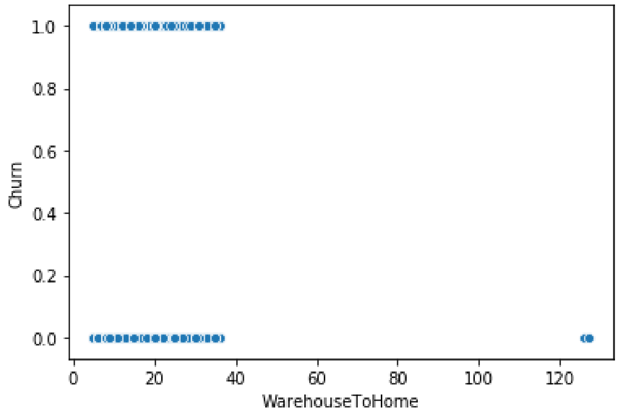
Bivariate analysis of Data:

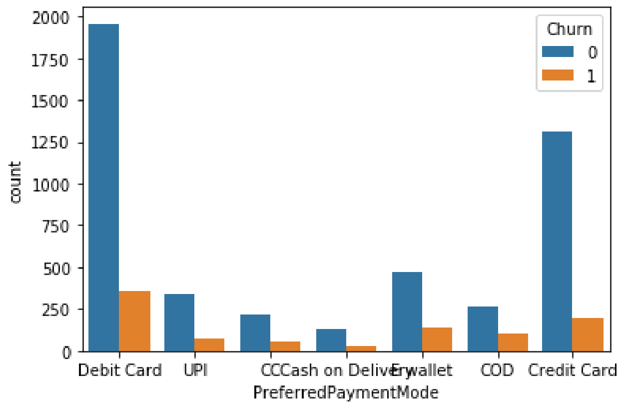
→Plot to compare the target variable 'Churn' with all other variables:

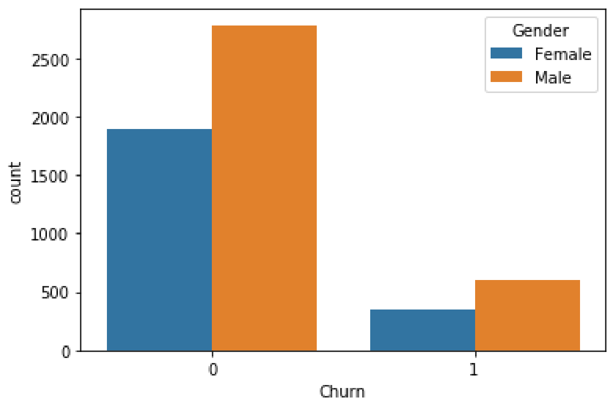


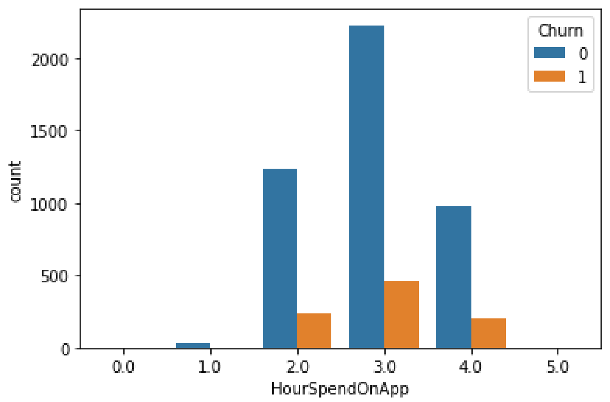


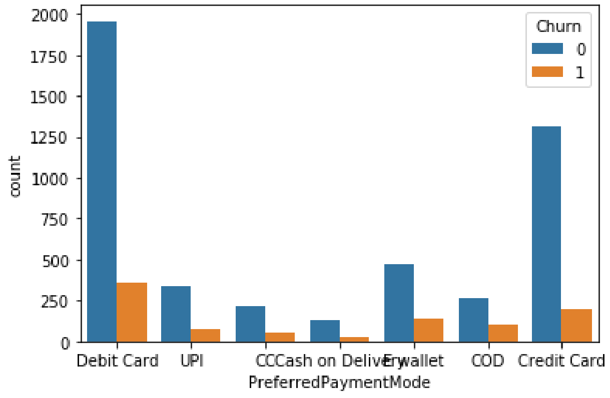


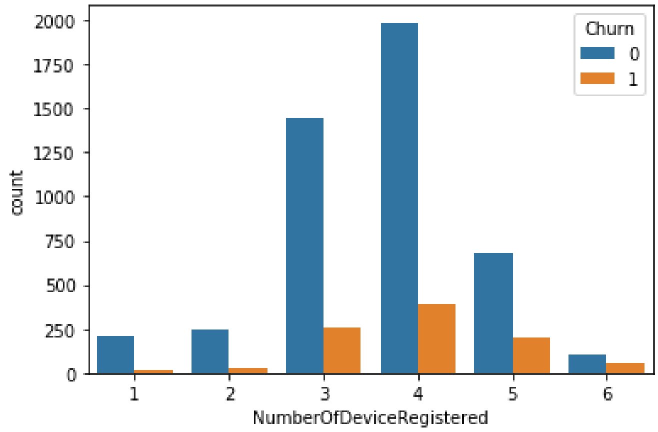


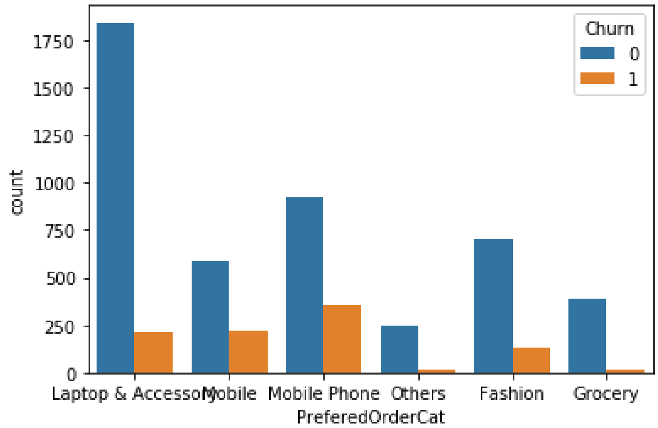

Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
Debit Card
UPI
CC
Cash on Delivery
E wallet
COD
Credit Card
PreferredPaymentMode

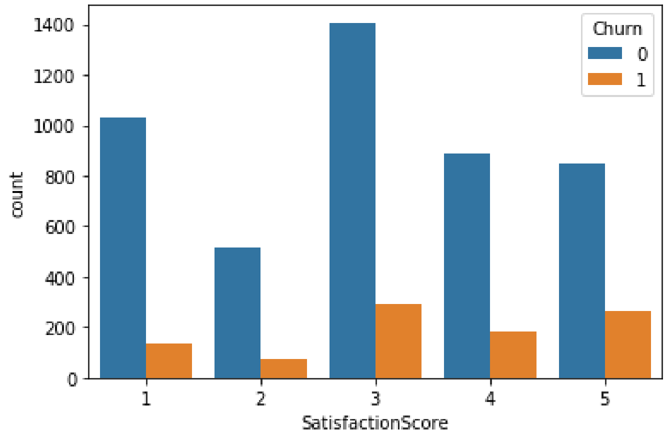

Gender
Female
Male
count
2500
2000
1500
1000
500
0
0
1
Churn

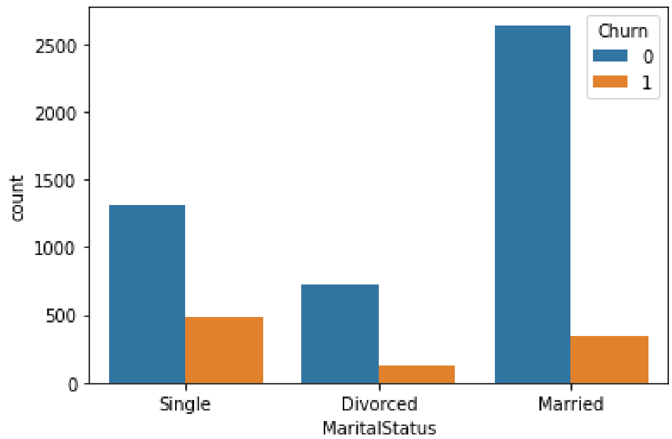

Churn
0
1
count
2000
1500
1000
500
0
0.0
1.0
2.0
3.0
4.0
5.0
HourSpendOnApp

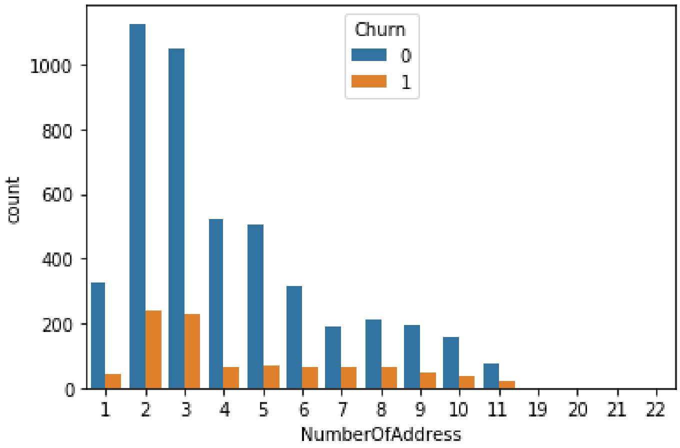

Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
Debit Card
UPI
CC
Cash on Delivery
E wallet
COD
Credit Card
PreferredPaymentMode

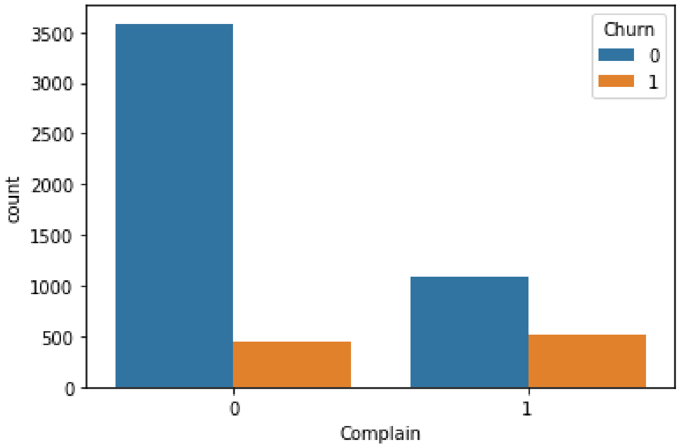

Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
1
2
3
4
5
6
NumberOfDeviceRegistered

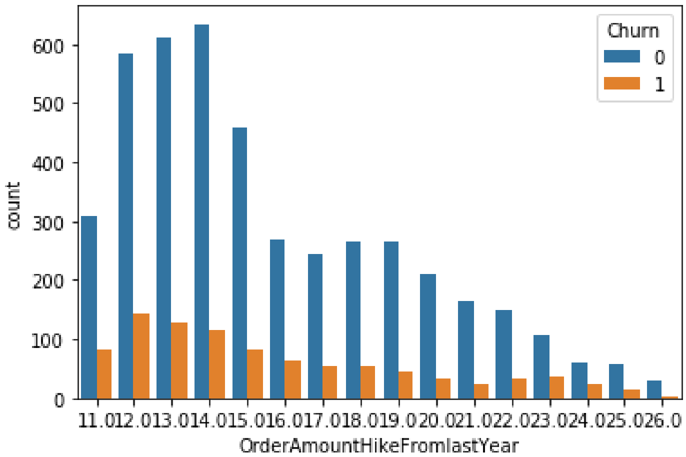

Churn
0
1
count
1750
1500
1250
1000
750
500
250
0
Laptop & Accessory
Mobile
Mobile Phone
Others
Fashion
Grocery
PreferedOrderCat

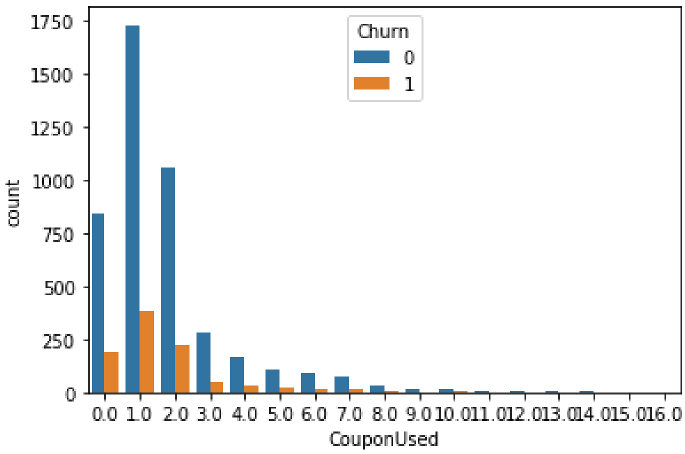

Churn
0
1
count
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
SatisfactionScore

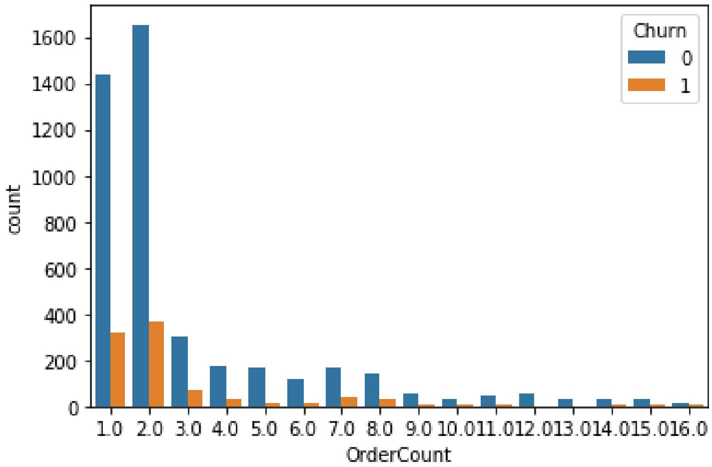

Churn
0
1
count
2500
2000
1500
1000
500
0
Single
Divorced
Married
MaritalStatus


Churn
0
1
count
0
200
400
600
800
1000
1 2 3 4 5 6 7 8 9 10 11 19 20 21 22
NumberOfAddress

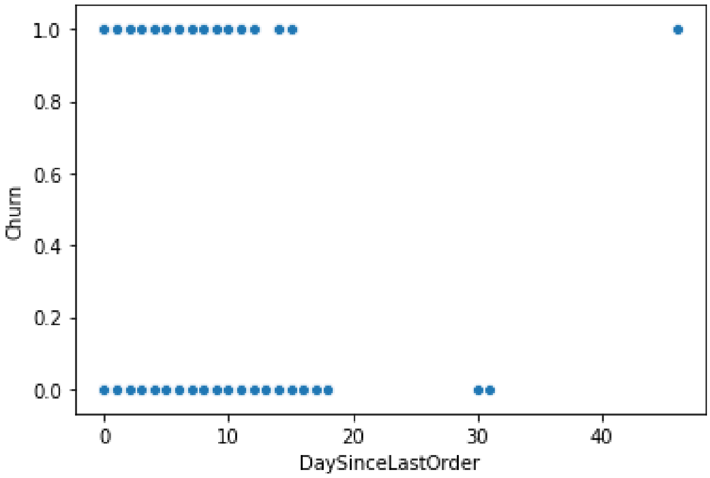

Churn
0
1
count
0
500
1000
1500
2000
2500
3000
3500
0 1
Complain


Churn
0
1
count
0
100
200
300
400
500
600
11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0 26.0
OrderAmountHikeFromlastYear

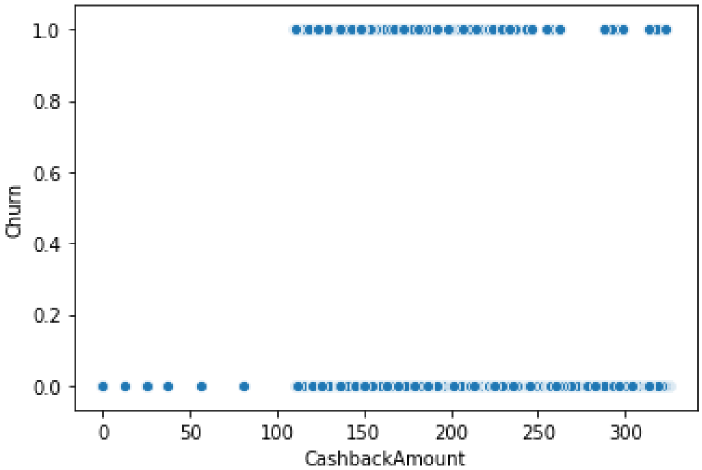

Churn
0
1
count
0
250
500
750
1000
1250
1500
1750
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0
CouponUsed



Churn
0
1
count
OrderCount

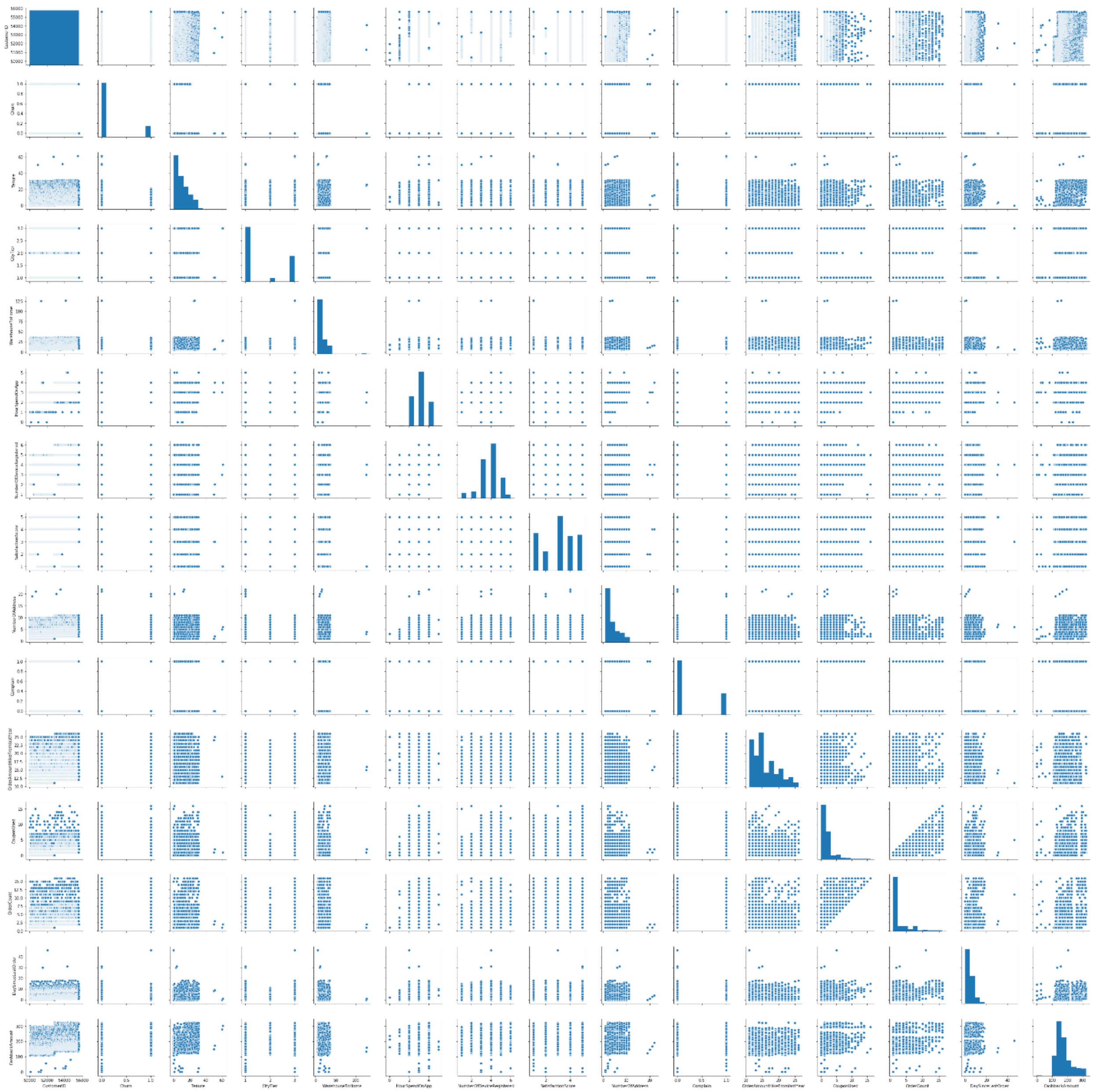


Churn
DaySinceLastOrder



Churn
CashbackAmount

→pairplot for the dataset:



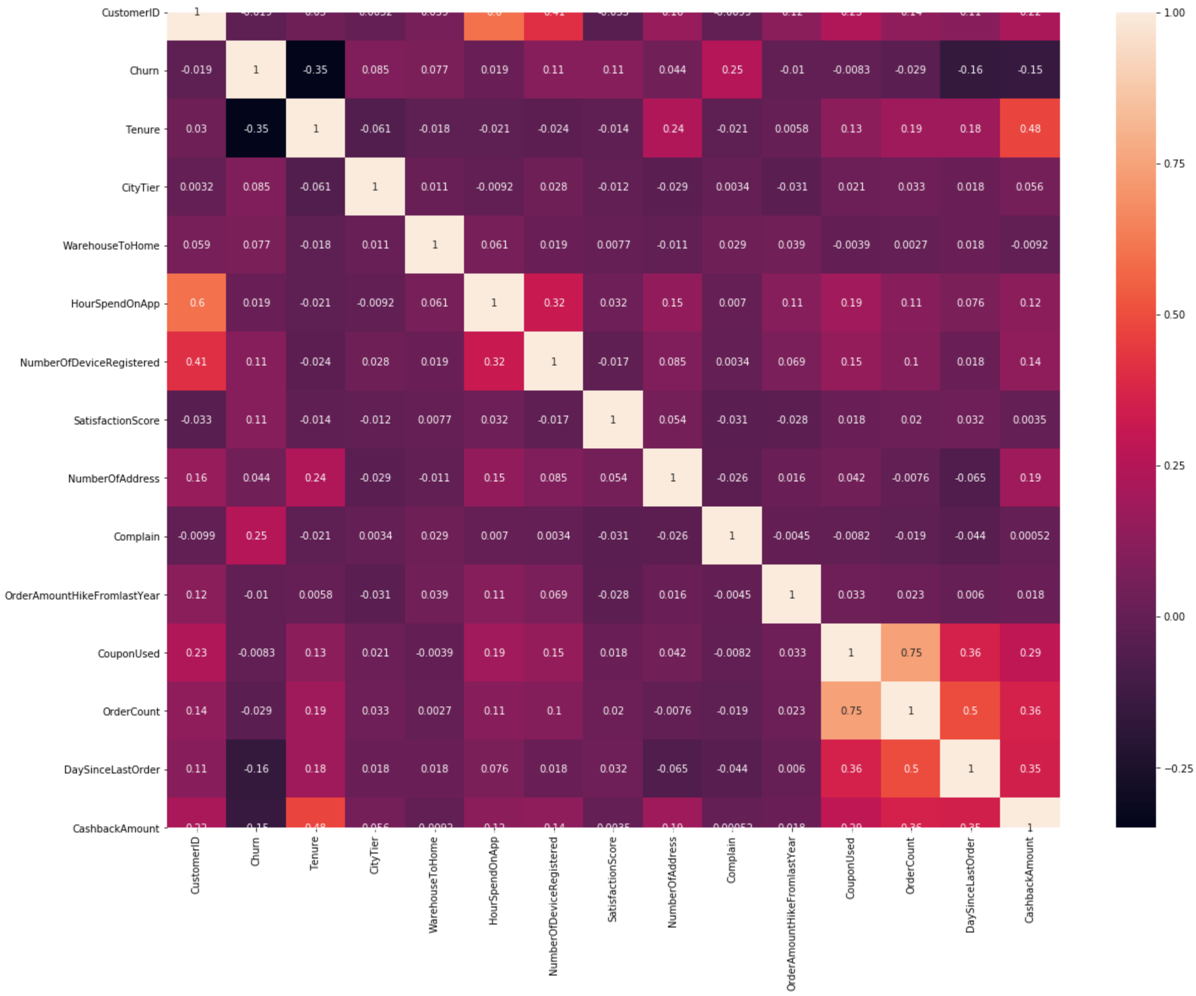
→Covariance of the dataset:

	CustomerID	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
CustomerID	2.641878e+06	-11.607746	415.767062	4.818973	816.048342	701.613022	684.227838	-74.358501	675.310446	-7.295790	700.545704	721.657476	664.341959	671.589191	17366.099122
Churn	-1.160775e+01	0.140055	-1.100586	0.029017	0.240076	0.005012	0.041364	0.054484	0.042476	0.042266	-0.014019	-0.005949	-0.031920	-0.219632	-2.838123
Tenure	4.157671e+02	-1.100586	73.226373	-0.478229	-1.338193	-0.131467	-0.210978	-0.164853	5.261460	-0.082221	0.177956	2.058108	4.680305	5.793798	198.617014
CityTier	4.818973e+00	0.029017	-0.478229	0.837938	0.083454	-0.006075	0.026184	-0.014598	-0.069625	0.001395	-0.106193	0.037293	0.089877	0.058225	2.510981
WarehouseToHome	8.160483e+02	0.240076	-1.338193	0.083454	72.786069	0.377563	0.166605	0.090792	-0.243143	0.110197	1.219237	-0.064153	0.067478	0.558954	-3.844719
HourSpendOnApp	7.016130e+02	0.005012	-0.131467	-0.006075	0.377563	0.521177	0.232839	0.031772	0.270559	0.002276	0.284254	0.262241	0.233266	0.200441	4.313659
NumberOfDeviceRegistered	6.842278e+02	0.041364	-0.210978	0.026184	0.166605	0.232839	1.048573	-0.024349	0.224866	0.001575	0.262746	0.295162	0.309143	0.068297	6.912367
SatisfactionScore	-7.435850e+01	0.054484	-0.164853	-0.014598	0.090792	0.031772	-0.024349	1.904937	0.191068	-0.019386	-0.140757	0.046926	0.080307	0.161646	0.235886
NumberOfAddress	6.753104e+02	0.042476	5.261460	-0.069625	-0.243143	0.270559	0.224866	0.191068	6.674914	-0.030788	0.147210	0.206397	-0.057346	-0.612186	23.733771
Complain	-7.295790e+00	0.042266	-0.082221	0.001395	0.110197	0.002276	0.001575	-0.019386	-0.030788	0.203769	-0.007526	-0.006992	-0.025526	-0.071899	0.011661
OrderAmountHikeFromlastYear	7.005457e+02	-0.014019	0.177956	-0.106193	1.219237	0.284254	0.262746	-0.140757	0.147210	-0.007526	13.509193	0.218439	0.232075	0.078311	2.695957
CouponUsed	7.216575e+02	-0.005949	2.058108	0.037293	-0.064153	0.262241	0.295162	0.046926	0.206397	-0.006992	0.218439	3.589590	3.876249	2.344206	24.552221
OrderCount	6.643420e+02	-0.031920	4.680305	0.089877	0.067478	0.233266	0.309143	0.080307	-0.057346	-0.025526	0.232075	3.876249	8.641716	5.065675	51.049896
DaySinceLastOrder	6.715892e+02	-0.219632	5.793798	0.058225	0.558954	0.200441	0.068297	0.161646	-0.612186	-0.071899	0.078311	2.344206	5.065675	13.354882	62.705781
CashbackAmount	1.736610e+04	-2.838123	198.617014	2.510981	-3.844719	4.313659	6.912367	0.235886	23.733771	0.011661	2.695957	24.552221	51.049896	62.705781	2421.332409

→Correlation of the dataset:

	CustomerID	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
CustomerID	1.000000	-0.019083	0.029952	0.003239	0.058909	0.598417	0.411098	-0.033146	0.160814	-0.009944	0.117243	0.234302	0.139008	0.113243	0.217129
Churn	-0.019083	1.000000	-0.349408	0.084703	0.076630	0.018675	0.107939	0.105481	0.043931	0.250188	-0.010058	-0.008264	-0.028697	-0.160757	-0.154118
Tenure	0.029952	-0.349408	1.000000	-0.060688	-0.018218	-0.021226	-0.023983	-0.013903	0.237666	-0.021268	0.005825	0.129035	0.186403	0.184552	0.476380
CityTier	0.003239	0.084703	-0.060688	1.000000	0.010624	-0.009150	0.027934	-0.011554	-0.029440	0.003375	-0.031408	0.021456	0.033388	0.017525	0.055746
WarehouseToHome	0.058909	0.076630	-0.018218	0.010624	1.000000	0.060990	0.019071	0.007722	-0.011020	0.028696	0.038795	-0.003935	0.002681	0.017829	-0.009200
HourSpendOnApp	0.598417	0.018675	-0.021226	-0.009150	0.060990	1.000000	0.316800	0.031858	0.145126	0.006976	0.106843	0.191528	0.109575	0.075716	0.121490
NumberOfDeviceRegistered	0.411098	0.107939	-0.023983	0.027934	0.019071	0.316800	1.000000	-0.017228	0.084997	0.003407	0.069475	0.151685	0.103464	0.018208	0.137183
SatisfactionScore	-0.033146	0.105481	-0.013903	-0.011554	0.007722	0.031858	-0.017228	1.000000	0.053583	-0.031115	-0.027730	0.017936	0.019764	0.032082	0.003473
NumberOfAddress	0.160814	0.043931	0.237666	-0.029440	-0.011020	0.145126	0.084997	0.053583	1.000000	-0.026399	0.015533	0.042120	-0.007609	-0.064847	0.186688
Complain	-0.009944	0.250188	-0.021268	0.003375	0.028696	0.006976	0.003407	-0.031115	-0.026399	1.000000	-0.004529	-0.008174	-0.019307	-0.043546	0.000525
OrderAmountHikeFromlastYear	0.117243	-0.010058	0.005825	-0.031408	0.038795	0.106843	0.069475	-0.027730	0.015533	-0.004529	1.000000	0.033201	0.023101	0.006003	0.017869
CouponUsed	0.234302	-0.008264	0.129035	0.021456	-0.003935	0.191528	0.151685	0.017936	0.042120	-0.008174	0.033201	1.000000	0.745245	0.358930	0.286728
OrderCount	0.139008	-0.028697	0.186403	0.033388	0.002681	0.109575	0.103464	0.019764	-0.007609	-0.019307	0.023101	0.745245	1.000000	0.497928	0.360984
DaySinceLastOrder	0.113243	-0.160757	0.184552	0.017525	0.017829	0.075716	0.018208	0.032082	-0.064847	-0.043546	0.006003	0.358930	0.497928	1.000000	0.347172
CashbackAmount	0.217129	-0.154118	0.476380	0.055746	-0.009200	0.121490	0.137183	0.003473	0.186688	0.000525	0.017869	0.286728	0.360984	0.347172	1.000000

→Heatmap of the correlation between the variables:



Checking the churn rate in city tier, Gender and PreferredPaymentMode:

City Tier:

```
CityTier  Churn
1         0        0.854883
          1        0.145117
2         0        0.801653
          1        0.198347
3         0        0.786295
          1        0.213705
Name: Churn, dtype: float64
```

Gender:

```
Gender  Churn
Female  0        0.845058
        1        0.154942
Male    0        0.822695
        1        0.177305
Name: Churn, dtype: float64
```

Preferred Payment Mode:

```
PreferredPaymentMode  Churn
Cash on Delivery      0        0.750973
                      1        0.249027
Credit Card           0        0.857948
                      1        0.142052
Debit Card            0        0.846154
                      1        0.153846
E wallet              0        0.771987
                      1        0.228013
UPI                   0        0.826087
                      1        0.173913
Name: Churn, dtype: float64
```

How your analysis is impacting the business?

- A customer churning out doesn't mean he/she stopped buying products online . They might have just moved to a different e-commerce company.
- This will directly impact the revenue of the business.
- The market share of other companies will increase, and the market share of o ur company will decrease.
- Saving the churns will help us keep the revenue flowing and giving better cust omer experience by collecting feedbacks.

Both visual and non-visual understanding of the data.

- There are 16.8% of the total customers who are churning.
- Male and Female customers show similar behavior in churning
- 21.3% of total customers present in Tier 3 cities are churning which is greater than the overall churn percentage

3. Data Cleaning and Pre-processing

→Approach used for identifying and treating missing values and outlier treatment (and why)

Missing Value treatment:

→Checking for missing values:

```
Churn                           0
Tenure                        264
PreferredLoginDevice            0
CityTier                        0
WarehouseToHome               251
PreferredPaymentMode            0
Gender                          0
HourSpendOnApp                255
NumberOfDeviceRegistered        0
PreferedOrderCat                0
SatisfactionScore               0
MaritalStatus                   0
NumberOfAddress                 0
Complain                        0
OrderAmountHikeFromlastYear   265
CouponUsed                    256
OrderCount                    258
DaySinceLastOrder             307
CashbackAmount                  0
dtype: int64
```

→ The 0s in Tenure variable was changed to NaN values as the 0 Tenure doesn't make any sense

→checking the description of the dataset after the above change:

	Churn	Tenure	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	NumberOfAddress	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
count	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000	5630.000000
mean	0.168384	11.083304	15.566785	2.934636	3.688988	4.214032	15.674600	1.716874	2.961812	4.459325	177.223030
std	0.374240	7.721916	8.345961	0.705528	1.023999	2.583586	3.591058	1.857640	2.879248	3.570626	49.207036
min	0.000000	1.000000	5.000000	0.000000	1.000000	1.000000	11.000000	0.000000	1.000000	0.000000	0.000000
25%	0.000000	5.000000	9.000000	2.000000	3.000000	2.000000	13.000000	1.000000	1.000000	2.000000	145.770000
50%	0.000000	10.000000	14.000000	3.000000	4.000000	3.000000	15.000000	1.000000	2.000000	3.000000	163.280000
75%	0.000000	15.000000	20.000000	3.000000	4.000000	6.000000	18.000000	2.000000	3.000000	7.000000	196.392500
max	1.000000	61.000000	127.000000	5.000000	6.000000	22.000000	26.000000	16.000000	16.000000	46.000000	324.990000

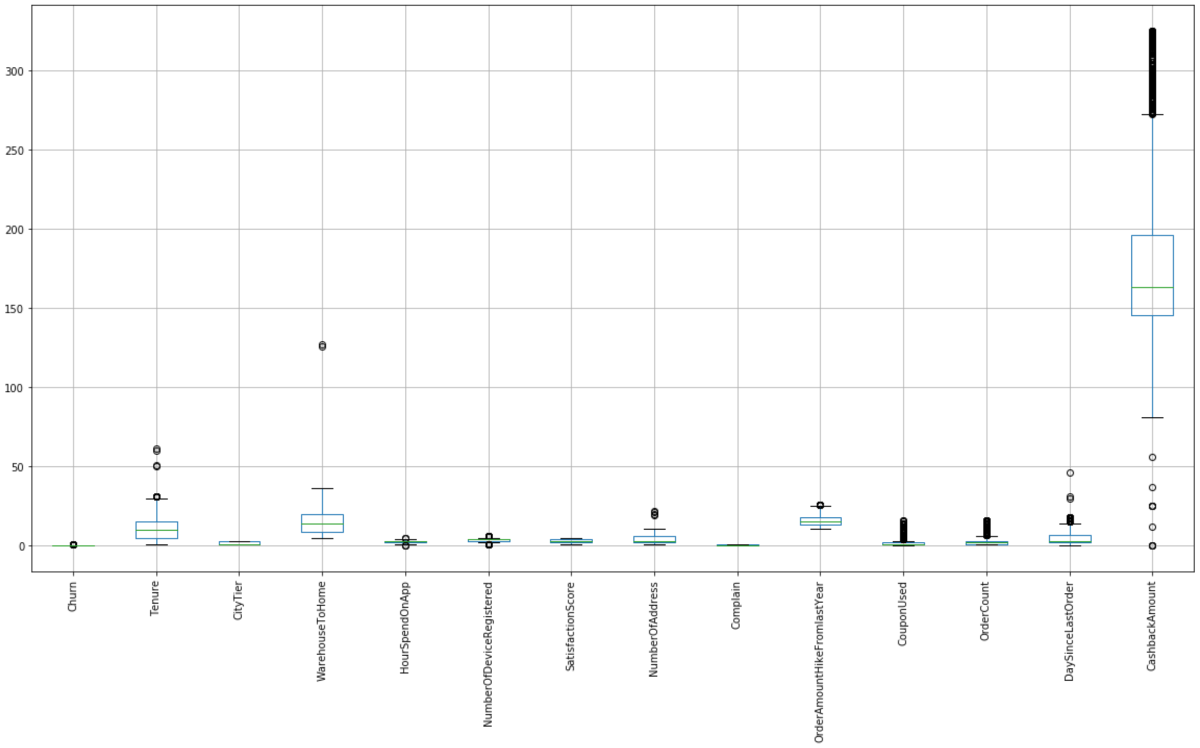
→The missing values were replaced with the median of the variable using the fillna() function on the dataset.

→Checking for missing values after imputation:

```
Churn                          0
Tenure                         0
PreferredLoginDevice           0
CityTier                       0
WarehouseToHome                0
PreferredPaymentMode           0
Gender                         0
HourSpendOnApp                 0
NumberOfDeviceRegistered       0
PreferedOrderCat               0
SatisfactionScore              0
MaritalStatus                  0
NumberOfAddress                0
Complain                       0
OrderAmountHikeFromlastYear    0
CouponUsed                     0
OrderCount                     0
DaySinceLastOrder              0
CashbackAmount                 0
dtype: int64
```

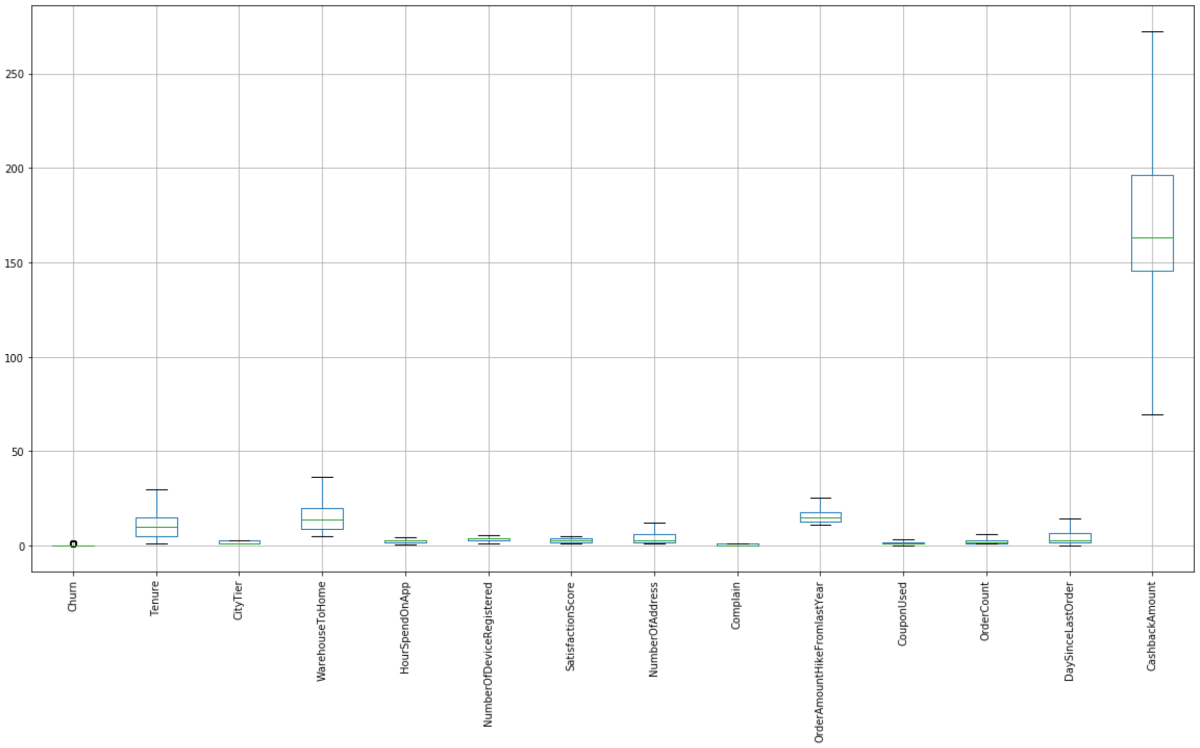
Outlier treatment:

→Boxplot of the dataset to understand the outliers present in the dataset:



→ The outliers were treated with the IQR range for the interger variables

→Boxplot of the dataset after the missing value treatment:



Need for variable transformation (if any)

→Checked for data types which are in int and are supposed to be Categorical
→Changed the data type from int to object for variables CityTier, SatisfactionScore and Complain

→Checking the info after changing the datatypes of above-mentioned variables:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5630 entries, 0 to 5629
Data columns (total 19 columns):
Churn                          5630 non-null int64
Tenure                         5630 non-null float64
PreferredLoginDevice           5630 non-null object
CityTier                       5630 non-null object
WarehouseToHome                5630 non-null float64
PreferredPaymentMode           5630 non-null object
Gender                         5630 non-null object
HourSpendOnApp                 5630 non-null float64
NumberOfDeviceRegistered       5630 non-null float64
PreferedOrderCat               5630 non-null object
SatisfactionScore              5630 non-null object
MaritalStatus                  5630 non-null object
NumberOfAddress                5630 non-null float64
Complain                       5630 non-null object
OrderAmountHikeFromlastYear    5630 non-null float64
CouponUsed                     5630 non-null float64
OrderCount                     5630 non-null float64
DaySinceLastOrder              5630 non-null float64
CashbackAmount                 5630 non-null float64
dtypes: float64(10), int64(1), object(8)
memory usage: 835.8+ KB
```

→Replacing 'Phone' with 'Mobile Phone' in PreferredLoginDevice variable as both mean the same:

Value counts before change:

```
Mobile Phone     2765
Computer         1634
Phone            1231
Name: PreferredLoginDevice, dtype: int64
```

Values counts after change:

```
Mobile Phone     3996
Computer         1634
Name: PreferredLoginDevice, dtype: int64
```

→Replacing 'CC' with 'Credit Card' in PreferredPaymentMode variable as both mean the same and Replacing 'COD' with 'COD' in Cash on Delivery variable as both mean the same:

Value counts before change:

```
Debit Card          2314
Credit Card         1501
E wallet             614
UPI                  414
COD                  365
CC                   273
Cash on Delivery     149
Name: PreferredPaymentMode, dtype: int64
```

Value counts after the change:

```
Debit Card          2314
Credit Card         1774
E wallet             614
Cash on Delivery     514
UPI                  414
Name: PreferredPaymentMode, dtype: int64
```

→Replacing 'Mobile' with 'Mobile Phone' in PreferedOrderCat variable as both mean the same:

Value counts before the change:

```
Laptop & Accessory    2050
Mobile Phone          1271
Fashion                826
Mobile                 809
Grocery                410
Others                 264
Name: PreferedOrderCat, dtype: int64
```

Value counts after the change:

```
Mobile Phone          2080
Laptop & Accessory    2050
Fashion                826
Grocery                410
Others                 264
Name: PreferedOrderCat, dtype: int64
```

Variables removed or added and why (if any)

→The variable CustomerID was removed using the drop function on the dataframe.
→The new shape of the data after removing the variable is `(5630, 19)`

4. Model Building

→Spliting the data set into independent variables and dependent variables by using drop and pop function on the dataset. Splitting the dataset into training and testing data by using the train_test_split function from sklearn.model_selection package. kept the test size as 30%

→Used GridsearchCV from sklearn.model_selection package to find the best fitting model by checking the best_params_ and best_estimator_.

→The shape of the train and test datasets:

```
x_train (3941, 18)
x_test (1689, 18)
y_train (3941,)
y_test (1689,)
```

→Decision Tree:

Best Parameter after using GridSearchCV:

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 50, 'm
in_samples_split': 150, 'random_state': 0}
```

Feature importances:

	Imp
Tenure	0.407454
Complain	0.242203
CashbackAmount	0.155259
DaySinceLastOrder	0.076986
NumberOfAddress	0.067982
MaritalStatus	0.050117
WarehouseToHome	0.000000
PreferredPaymentMode	0.000000
Gender	0.000000
HourSpendOnApp	0.000000
NumberOfDeviceRegistered	0.000000
PreferedOrderCat	0.000000
PreferredLoginDevice	0.000000
CityTier	0.000000
OrderAmountHikeFromlastYear	0.000000
CouponUsed	0.000000
OrderCount	0.000000
SatisfactionScore	0.000000

→Random Forest Classifier:

Best Parameters using GridSearchCV:

```
{'max_depth': 6,
 'max_features': 12,
 'min_samples_leaf': 50,
 'min_samples_split': 150,
 'n_estimators': 301,
'random_state': 0}
```

Feature importances:

	Imp
Tenure	0.439793
Complain	0.184214
CashbackAmount	0.125465
DaySinceLastOrder	0.073799
NumberOfAddress	0.050385
MaritalStatus	0.034296
PreferedOrderCat	0.031590
CityTier	0.013849
SatisfactionScore	0.013074
WarehouseToHome	0.012421
NumberOfDeviceRegistered	0.006574
PreferredPaymentMode	0.004042
OrderAmountHikeFromlastYear	0.003380
OrderCount	0.002209
CouponUsed	0.001770
PreferredLoginDevice	0.001615
Gender	0.000911
HourSpendOnApp	0.000613

→Artificial Neural Networks:

Best Parameters using GridSearchCV:

```
{'hidden_layer_sizes': 200,
'max_iter': 10000,
'random_state': 0,
'solver': 'adam',
'tol': 0.01}
```

→Logistic Regression:

Used LogisticRegression() algorithm from sklearn.linear_model package.

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→Linear Discriminant Analysis:

Used LinearDiscriminantAnalysis() algorithm from sklearn.discriminant_analysis package.

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→KNN:

Used KNeighborsClassifier() algorithm from sklearn.neighbors package.

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→Naïve Bayes Model:

Used GaussianNB() algorithm from sklearn.naive_bayes package.

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→Support Vector Machine Model:

Used SVC() algorithm from sklearn.svm package.

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→Ada Boost:

Used AdaBoostClassifier () algorithm from sklearn.ensemble package.

Used n_estimators=100 and random_state=1

Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

→XGBoost:

Used xgb() algorithm from xgboost package.

User learning rate of 0.01 and random state of 1.

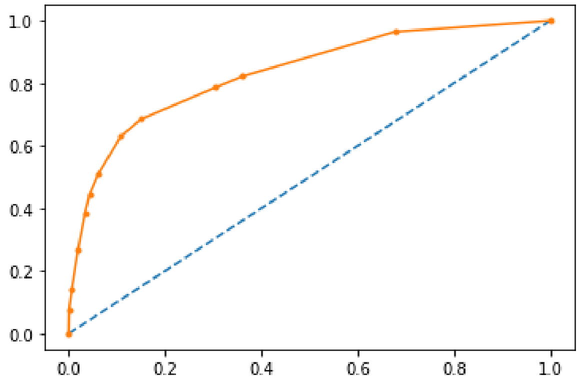
Did model fitting by passing dependent and target variables from training dataset by using .fit() function.

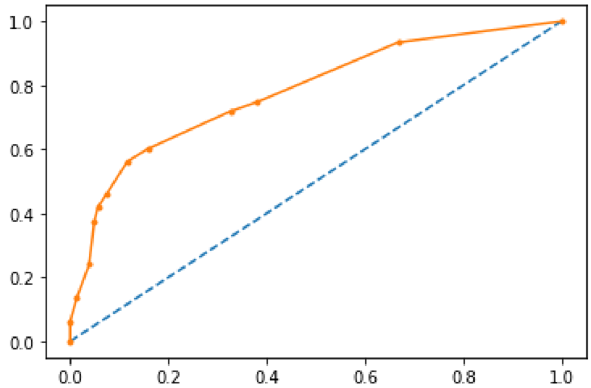
Did predictions by using .predict() function on the model, by passing the training and testing set of dependent variables.

Model Evaluation:

→Decision Tree:

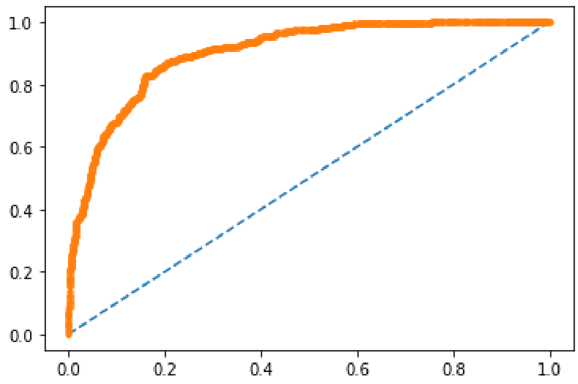
```
0.8690687642730272
0.8584961515689757
[[3125  142]
 [ 374  300]]
[[1335   80]
 [ 159  115]]
              precision    recall  f1-score   support

           0       0.89      0.96      0.92      3267
           1       0.68      0.45      0.54       674

    accuracy                           0.87      3941
   macro avg       0.79      0.70      0.73      3941
weighted avg       0.86      0.87      0.86      3941

              precision    recall  f1-score   support

           0       0.89      0.94      0.92      1415
           1       0.59      0.42      0.49       274

    accuracy                           0.86      1689
   macro avg       0.74      0.68      0.70      1689
weighted avg       0.84      0.86      0.85      1689

0.8332311515478497
0.7794085785767714
```

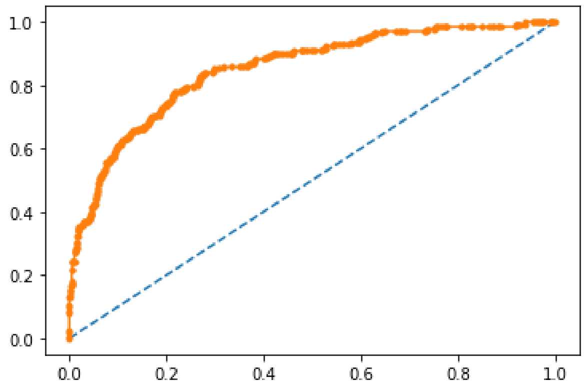




→Random Forest Classifier:

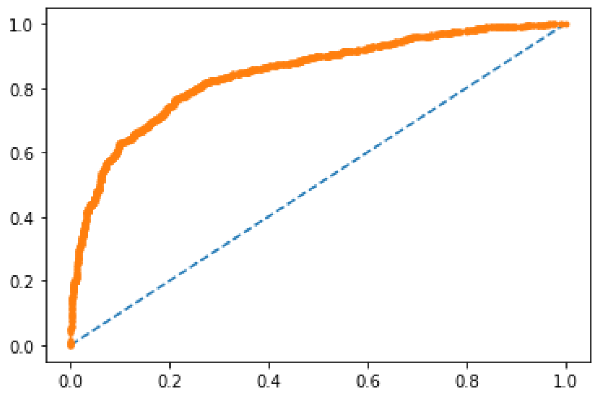
```
0.8741436183709719
0.8650088809946714
[[3164  103]
 [ 393  281]]
[[1349   66]
 [ 162  112]]
              precision    recall  f1-score   support

           0       0.89      0.97      0.93      3267
           1       0.73      0.42      0.53       674

    accuracy                           0.87      3941
   macro avg       0.81      0.69      0.73      3941
weighted avg       0.86      0.87      0.86      3941

              precision    recall  f1-score   support

           0       0.89      0.95      0.92      1415
           1       0.63      0.41      0.50       274

    accuracy                           0.87      1689
   macro avg       0.76      0.68      0.71      1689
weighted avg       0.85      0.87      0.85      1689

0.904313570013597
0.8515681824043745
```

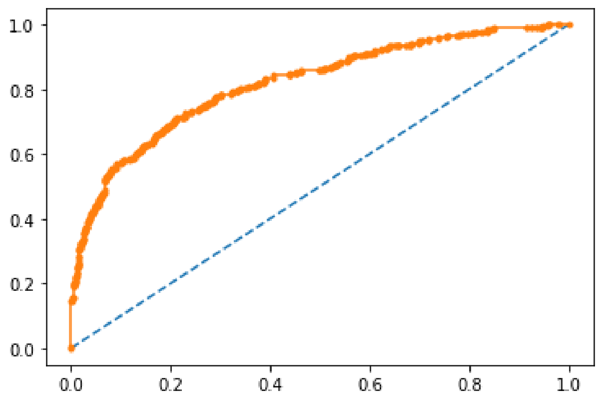




→Artificial Neural Networks:

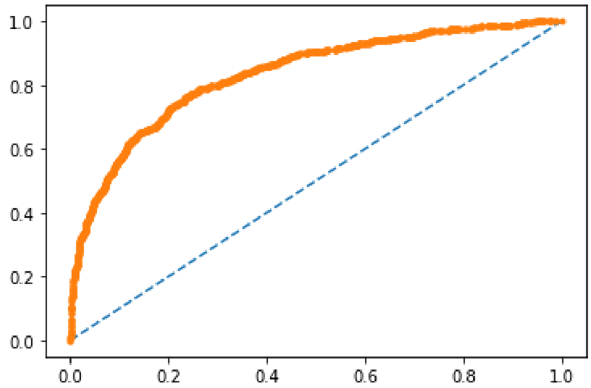
```
0.8660238518142603
0.8715216104203671
[[3107  160]
 [ 368  306]]
[[1362   53]
 [ 164  110]]
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      3267
           1       0.66      0.45      0.54       674

    accuracy                           0.87      3941
   macro avg       0.78      0.70      0.73      3941
weighted avg       0.85      0.87      0.86      3941

              precision    recall  f1-score   support

           0       0.89      0.96      0.93      1415
           1       0.67      0.40      0.50       274

    accuracy                           0.87      1689
   macro avg       0.78      0.68      0.71      1689
weighted avg       0.86      0.87      0.86      1689

0.8448103914788565
0.8196641819916948
```

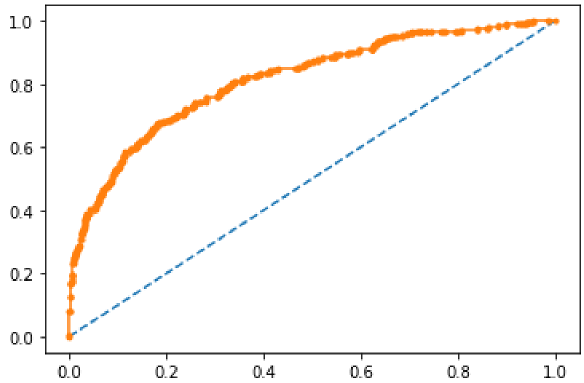




→Logistic Regression:

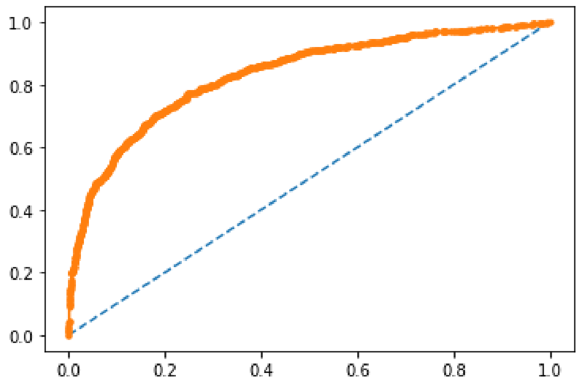
```
0.8619639685359046
0.8673771462403789
[[3168   99]
 [ 445  229]]
[[1387   28]
 [ 196   78]]
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      3267
           1       0.70      0.34      0.46       674

    accuracy                           0.86      3941
   macro avg       0.79      0.65      0.69      3941
weighted avg       0.85      0.86      0.84      3941

              precision    recall  f1-score   support

           0       0.88      0.98      0.93      1415
           1       0.74      0.28      0.41       274

    accuracy                           0.87      1689
   macro avg       0.81      0.63      0.67      1689
weighted avg       0.85      0.87      0.84      1689

0.834344705938987
0.8154161615640556
```

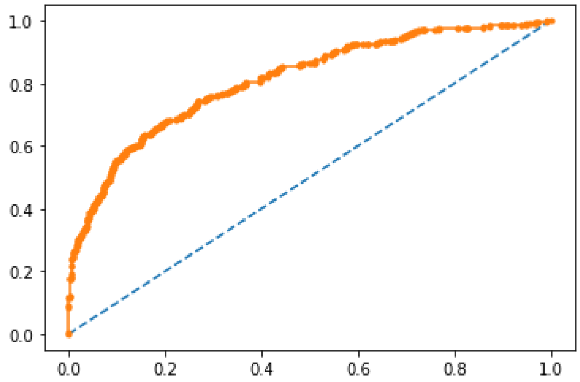




→Linear Discriminant Analysis:

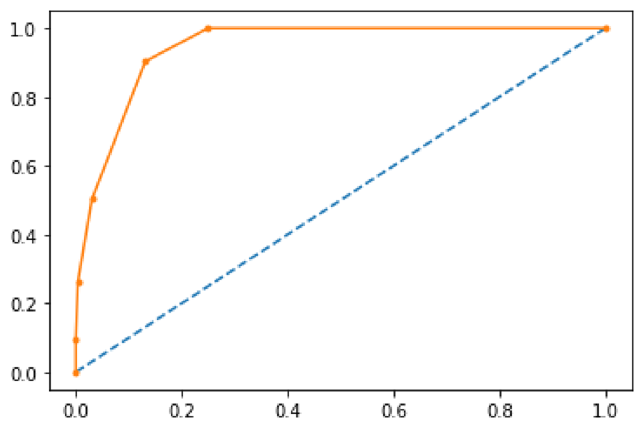
```
0.8655163664044658
0.8691533451746596
[[3165  102]
 [ 428  246]]
[[1385   30]
 [ 191   83]]
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      3267
           1       0.71      0.36      0.48       674

    accuracy                           0.87      3941
   macro avg       0.79      0.67      0.70      3941
weighted avg       0.85      0.87      0.85      3941

              precision    recall  f1-score   support

           0       0.88      0.98      0.93      1415
           1       0.73      0.30      0.43       274

    accuracy                           0.87      1689
   macro avg       0.81      0.64      0.68      1689
weighted avg       0.86      0.87      0.85      1689

0.8334836540933116
0.8120528229862526
```

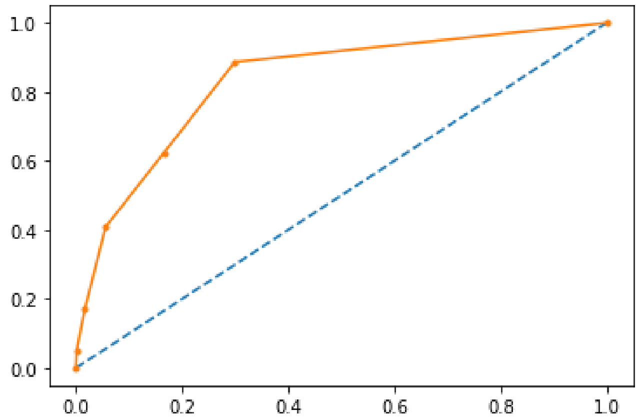




→K-Nearest Neighbours:

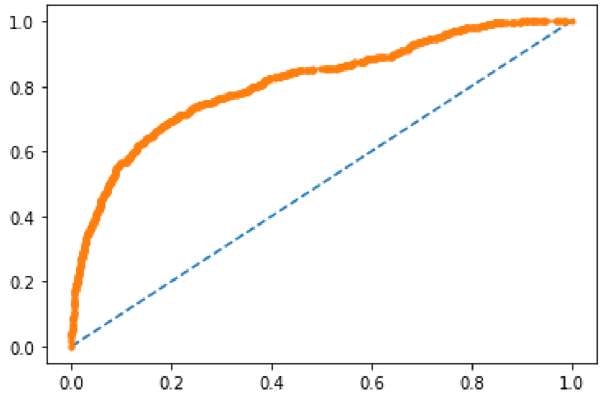
```
0.8896219233697031
0.857312018946122
[[3167  100]
 [ 335  339]]
[[1336   79]
 [ 162  112]]
              precision    recall  f1-score   support

           0       0.90      0.97      0.94      3267
           1       0.77      0.50      0.61       674

    accuracy                           0.89      3941
   macro avg       0.84      0.74      0.77      3941
weighted avg       0.88      0.89      0.88      3941
              precision    recall  f1-score   support

           0       0.89      0.94      0.92      1415
           1       0.59      0.41      0.48       274

    accuracy                           0.86      1689
   macro avg       0.74      0.68      0.70      1689
weighted avg       0.84      0.86      0.85      1689

0.9446964928486374
0.8316215728250497
```

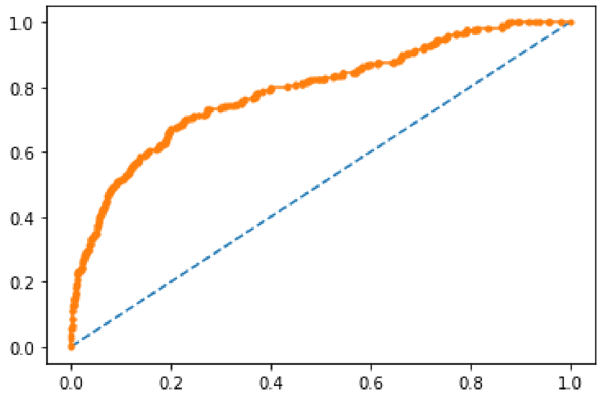




→Naïve Bayes:

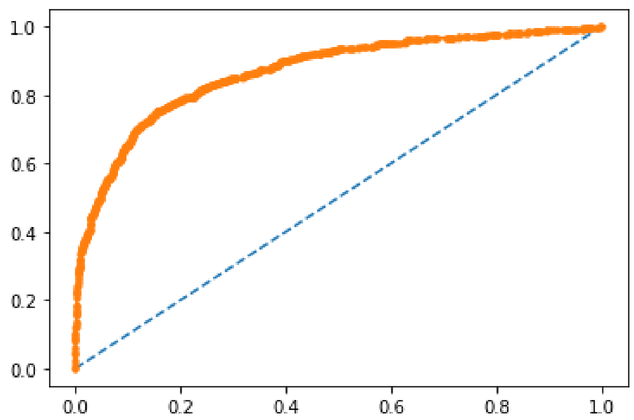
```
0.8520680030449125
0.8525754884547069
[[3021  246]
 [ 337  337]]
[[1319   96]
 [ 153  121]]
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      3267
           1       0.58      0.50      0.54       674

    accuracy                           0.85      3941
   macro avg       0.74      0.71      0.72      3941
weighted avg       0.84      0.85      0.85      3941

              precision    recall  f1-score   support

           0       0.90      0.93      0.91      1415
           1       0.56      0.44      0.49       274

    accuracy                           0.85      1689
   macro avg       0.73      0.69      0.70      1689
weighted avg       0.84      0.85      0.85      1689

0.8106880331050819
0.7890794666116426
```

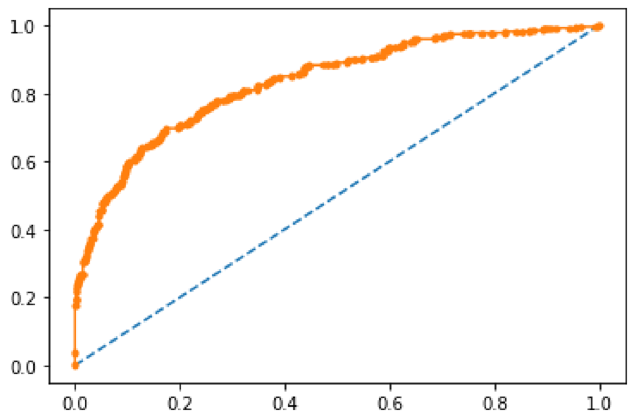




→Support Vector Machine:

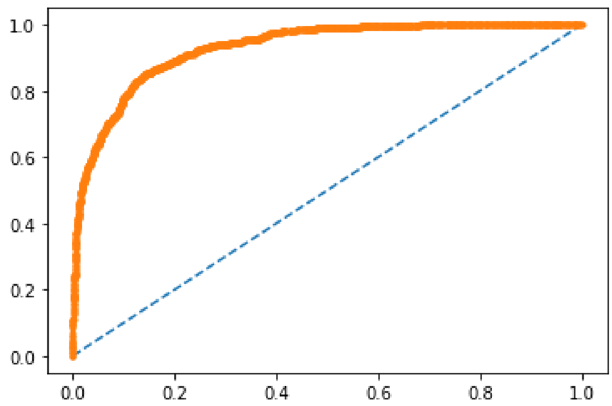
```
0.8289774168992642
0.8377738306690349
[[3267    0]
 [ 674    0]]
[[1415    0]
 [ 274    0]]
              precision    recall  f1-score   support

           0       0.83      1.00      0.91      3267
           1       0.00      0.00      0.00       674

    accuracy                           0.83      3941
   macro avg       0.41      0.50      0.45      3941
weighted avg       0.69      0.83      0.75      3941

              precision    recall  f1-score   support

           0       0.84      1.00      0.91      1415
           1       0.00      0.00      0.00       274

    accuracy                           0.84      1689
   macro avg       0.42      0.50      0.46      1689
weighted avg       0.70      0.84      0.76      1689
```

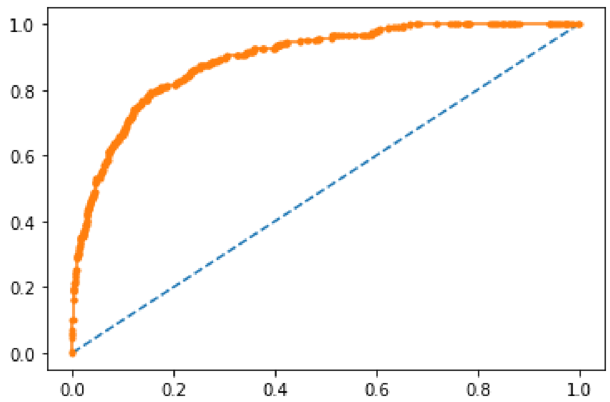




→Metrics for ADA Boost:

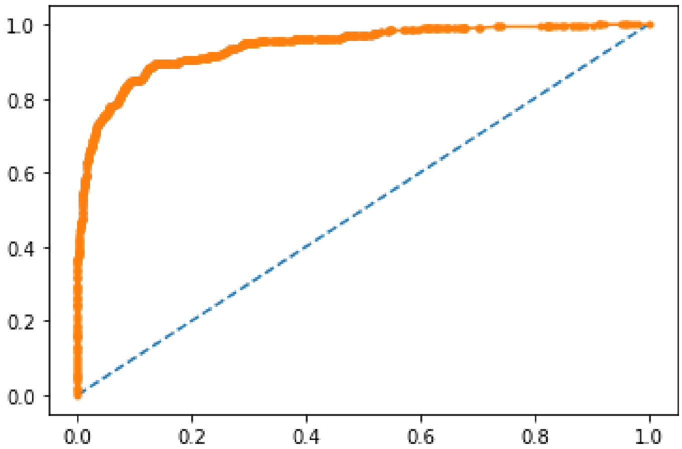
```
0.8995178888606953
0.881586737714624
[[3155  112]
 [ 284  390]]
[[1362   53]
 [ 147  127]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      3267
           1       0.78      0.58      0.66       674

    accuracy                           0.90      3941
   macro avg       0.85      0.77      0.80      3941
weighted avg       0.89      0.90      0.89      3941

              precision    recall  f1-score   support

           0       0.90      0.96      0.93      1415
           1       0.71      0.46      0.56       274

    accuracy                           0.88      1689
   macro avg       0.80      0.71      0.75      1689
weighted avg       0.87      0.88      0.87      1689

0.9305477216186685
0.8944778313688064
```

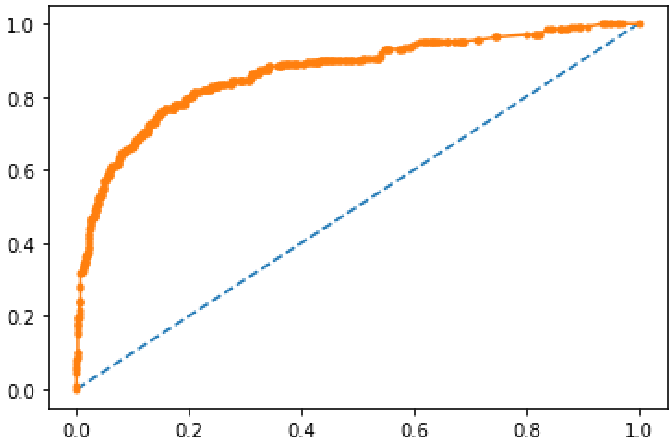




→XGB Metrics:

```
0.9182948490230906
0.886915334517466
[[3219   48]
 [ 274  400]]
[[1368   47]
 [ 144  130]]
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      3267
           1       0.89      0.59      0.71       674

    accuracy                           0.92      3941
   macro avg       0.91      0.79      0.83      3941
weighted avg       0.92      0.92      0.91      3941

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      1415
           1       0.73      0.47      0.58       274

    accuracy                           0.89      1689
   macro avg       0.82      0.72      0.76      1689
weighted avg       0.88      0.89      0.88      1689

0.9416355806968163
0.8680895514688813
```





Interpretation of the models:

→Considering that we are working on a customer churn problem, recall will be one of the most important factors for us to decide on which model is giving the best results.

Checking on the Recall and Accuracy of all the models:

→Tree algorithms:

→Decision Tree Classifier has a better recall when compared with Random Forest Classifier.
→Random Forest Classifier has a slightly better accuracy when compared with Decision Tree Classifier.

In artificial Neural Network, we see:

→The algorithm has similar performance compared to the other tree algorithms.

→The accuracy of test data is slightly better than train data

→In Logistic Regression:

→The algorithm has least recall value when compared with all of the other algorithms.

→The accuracy score seems to be same when compared with above algorithms.

→Linear Discriminant Analysis:

→The metrics show that this algorithm has one of the least recall when compared with all other algorithms.

→But the accuracy score looks similar to the model built on Logistic Regression.

→K-Nearest Neighbour:

→This model has one of the highest recall value when compared with other algorithms.

→However, the accuracy score looks similar as most of the other algorithms.

→Naïve Bayes:

→The model has the highest recall value when compared with other algorithms.

→However, the accuracy score looks similar as most of the other algorithms.

→Support Vector Machine

→The recall value is the least when compared with other algorithms.

→The accuracy score also is very low when compared with other algorithms.

→From the above comparison, we can see that Naïve Bayes model is best performing for our data followed by K-Nearest Neighbours model.

→Boosting algorithms:

ADA Boost and XG Boost have better model performance considering the metrics and compared to other models above.

XG Boost seem to been performing slightly better compared to ADA Boost both in terms of accuracy and recall factor.

Effort to improve model performance.

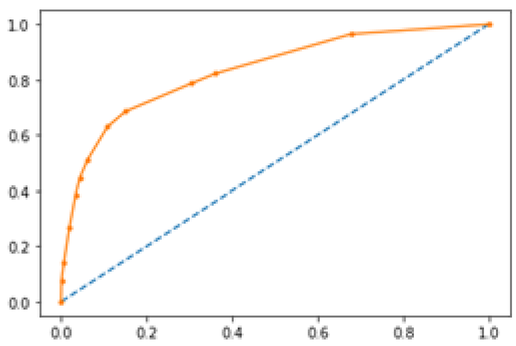
→Few of the model tuning measures are scaling and using SMOTE for the imbalanced data.

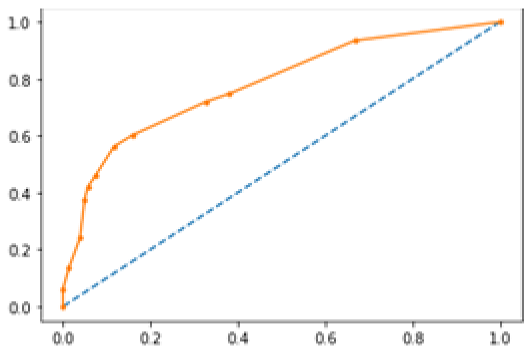
→Both of these techniques were used and few of the models were built on the modified datasets.

→For Scaled data:

→Metrics for Decision tree after scaling:

```
0.8690687642730272
0.8584961515689757
[[3125  142]
 [ 374  300]]
[[1335   80]
 [ 159  115]]
              precision    recall  f1-score   support

           0       0.89      0.96      0.92      3267
           1       0.68      0.45      0.54       674

    accuracy                           0.87      3941
   macro avg       0.79      0.70      0.73      3941
weighted avg       0.86      0.87      0.86      3941

              precision    recall  f1-score   support

           0       0.89      0.94      0.92      1415
           1       0.59      0.42      0.49       274

    accuracy                           0.86      1689
   macro avg       0.74      0.68      0.70      1689
weighted avg       0.84      0.86      0.85      1689

0.8332311515478497
0.7794085785767714
```

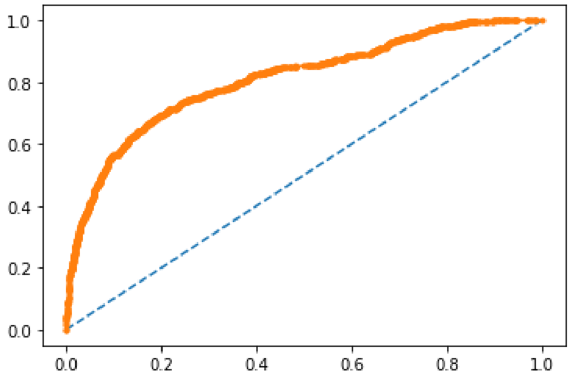


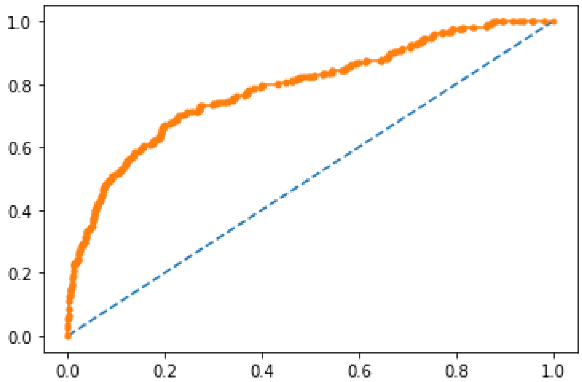


→On scaling, the results did not change much. The metrics are almost same as the model built on non-scaled data.

→Metrics for Naïve Bayes after scaling:

```
0.8520680030449125
0.8525754884547069
[[3021  246]
 [ 337  337]]
[[1319   96]
 [ 153  121]]
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      3267
           1       0.58      0.50      0.54       674

    accuracy                           0.85      3941
   macro avg       0.74      0.71      0.72      3941
weighted avg       0.84      0.85      0.85      3941

              precision    recall  f1-score   support

           0       0.90      0.93      0.91      1415
           1       0.56      0.44      0.49       274

    accuracy                           0.85      1689
   macro avg       0.73      0.69      0.70      1689
weighted avg       0.84      0.85      0.85      1689

0.8106880331050819
0.7890794666116426
```

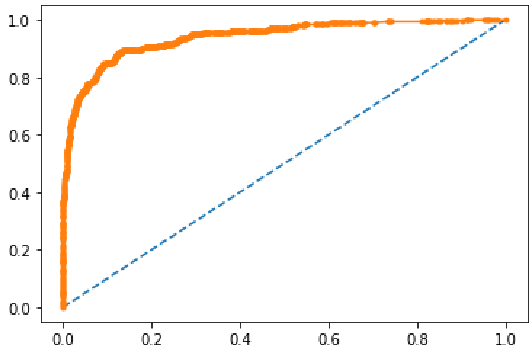


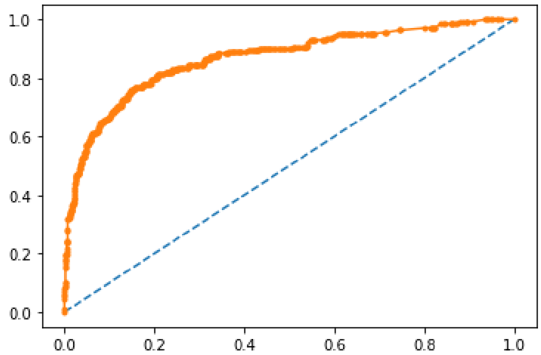


→On scaling, the results did not change much. The metrics are almost same as the model built on non-scaled data.

→Metrics for XG Boost after scaling:

```
0.9182948490230906
0.886915334517466
[[3219   48]
 [ 274  400]]
[[1368   47]
 [ 144  130]]
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      3267
           1       0.89      0.59      0.71       674

    accuracy                           0.92      3941
   macro avg       0.91      0.79      0.83      3941
weighted avg       0.92      0.92      0.91      3941

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      1415
           1       0.73      0.47      0.58       274

    accuracy                           0.89      1689
   macro avg       0.82      0.72      0.76      1689
weighted avg       0.88      0.89      0.88      1689

0.9416355806968163
0.8680895514688813
```





→On scaling, the results did not change much. The metrics are almost same as the model built on non-scaled data.

→Using SMOTE to balance the scaled data:

SMOTE function was used from imblearn.over_sampling library, with the random state of 2 to balance the data.

```
Before OverSampling, counts of label '1': 674
Before OverSampling, counts of label '0': 3267

After OverSampling, the shape of train_X: (6534, 18)
After OverSampling, the shape of train_y: (6534,)

After OverSampling, counts of label '1': 3267
After OverSampling, counts of label '0': 3267
```

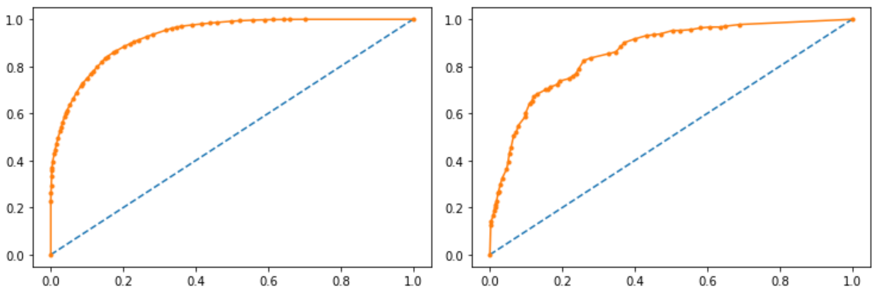
→Decision tree:

The best estimator using GridSearchCV came out to be:

{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 150, 'random_state': 0}

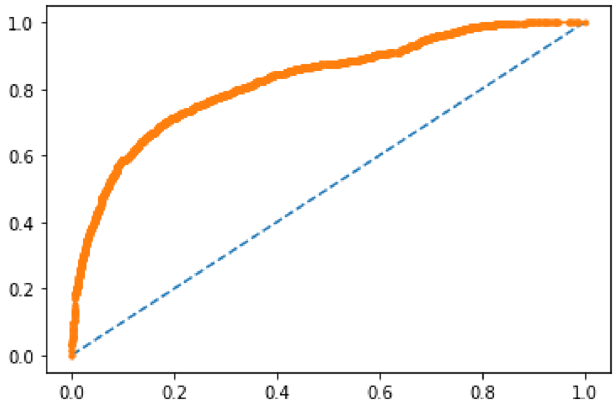
Metrics:

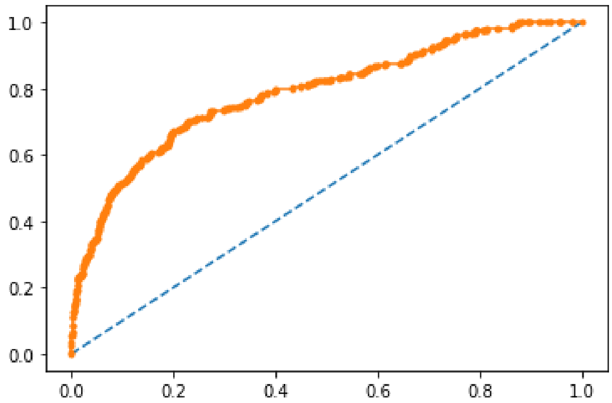
```
0.8431282522191613
0.8146832445233866
[[2759  508]
 [ 517 2750]]
[[1181  234]
 [  79  195]]
              precision    recall  f1-score   support

           0       0.84      0.84      0.84      3267
           1       0.84      0.84      0.84      3267

    accuracy                           0.84      6534
   macro avg       0.84      0.84      0.84      6534
weighted avg       0.84      0.84      0.84      6534
              precision    recall  f1-score   support

           0       0.94      0.83      0.88      1415
           1       0.45      0.71      0.55       274

    accuracy                           0.81      1689
   macro avg       0.70      0.77      0.72      1689
weighted avg       0.86      0.81      0.83      1689

0.9290056701359816
0.8585876041371128
```



→Naïve Bayes metrics after applying SMOTE on the scaled dataset:

```
0.7246709519436793
0.8525754884547069
[[3021  246]
 [1553 1714]]
[[1319   96]
 [ 153  121]]
              precision    recall  f1-score   support

           0       0.66      0.92      0.77      3267
           1       0.87      0.52      0.66      3267

    accuracy                           0.72      6534
   macro avg       0.77      0.72      0.71      6534
weighted avg       0.77      0.72      0.71      6534

              precision    recall  f1-score   support

           0       0.90      0.93      0.91      1415
           1       0.56      0.44      0.49       274

    accuracy                           0.85      1689
   macro avg       0.73      0.69      0.70      1689
weighted avg       0.84      0.85      0.85      1689

0.8263067738538702
0.7890794666116426
```

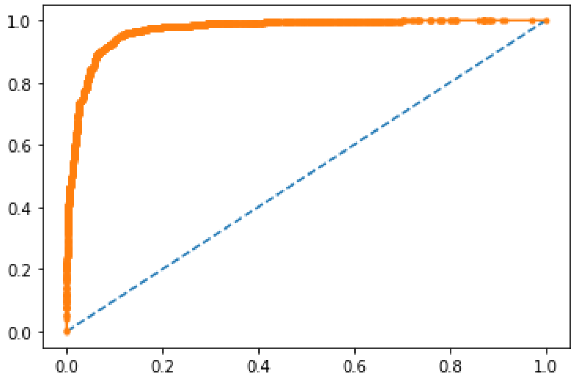


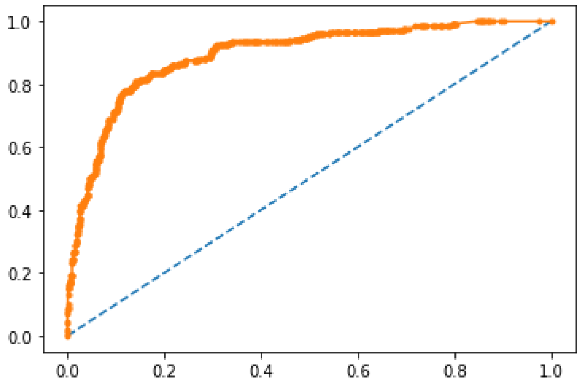


→Applying SMOTE technique on the scaled dataset helped in better results as we can see from the above metrics.

→Metrics for XG Boost model using balanced data:

```
0.9152127333945516
0.8679692125518058
[[2971  296]
 [ 258 3009]]
[[1269  146]
 [  77  197]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.91      3267
           1       0.91      0.92      0.92      3267

    accuracy                           0.92      6534
   macro avg       0.92      0.92      0.92      6534
weighted avg       0.92      0.92      0.92      6534

              precision    recall  f1-score   support

           0       0.94      0.90      0.92      1415
           1       0.57      0.72      0.64       274

    accuracy                           0.87      1689
   macro avg       0.76      0.81      0.78      1689
weighted avg       0.88      0.87      0.87      1689

0.9705129318619593
0.8939142658172344
```





→Applying SMOTE technique on the scaled dataset helped in better results as we can see from the above metrics.

5. Model validation:

How was the model validated ? Just accuracy, or anything else too ?

→Accuracy of the model will not completely tell us the model performance.

→Recall meaning – What percentage of churners are captured correctly by the model.

→In our project, the recall factor is important since it is customer churn analysis and we do not want to miss out on the potential churners

→Hence, a combination of Accuracy and recall factor is chosen for model validation

-PTO

6. Final interpretation / recommendation

Interpretation:

The analysis was made after applying multiple algorithms on normal data, scaled data and after applying SMOTE technique to balance the data after scaling.

The performance metrics of original data and scaled data seem to be similar.

The performance metrics for models built on the balanced data show way better results when compared with the other models.

From the above analysis, we can see that the model built on XG Boost looks to be performing the best for us.

The best results are seen when the model is built using the scaled and balanced data.

XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting.

This model will be one of the best to use for our analysis of customer churn for this business due the metrics such as Recall and Accuracy being the best when compared with others. This is also better considering the below:

We can consider XG Boost model for our further analysis as this model shows the best metrics during the model evaluation.

The XG Boost model is advantageous as well. It does non-greedy tree pruning, built-in cross-validation and handling the missing values.

Recommendations:

The Company can consider starting to collect the feedbacks to understand what makes customers unhappy which can cause their churning

The company can consider promotional offers in the area where churning rate is seen to be high

Few ways to stop churning:

Giving cashbacks and free promotional cash to lure them to buy from our company.

Start a payback card and actively communicating the offers and discounts they can avail.