greatlearning
Learning for Life

Great Learning
POWER AHEAD

Batch details	PGPDSE-FT Offline BLR Oct22
Team members	Venkata Sai Pavan Teja Angina Shivaprasad G Akhil Anilkumar Arun Sv Sahil Kumar Meher
Domain of Project	Retail
Proposed project title	Telecom Churn
Group Number	9
Team Leader	Sahil Kumar Meher
Mentor Name	Ms. Pranita Mahajan

greatlearning
Learning for Life

Table of Contents:

PROJECT DETAILS

Overview

In developed countries, the telecommunication industry has become a major player, with a large number of operators competing fiercely due to technological advancements. With this competition, companies are striving to survive and thrive by implementing various strategies. One of the major concerns in this industry is customer churn, which is the rate at which customers switch to other service providers. Predicting which customers are likely to leave early can potentially generate significant additional revenue for the company.

Industry Review

Introduction to domain:

Telecommunications involves the transmission of information electronically across long distances, including voice, data, text, images, and video. Telecommunications networks connect computer systems over remote locations, allowing for efficient communication.

In the present day, the telecommunications industry faces intense competition in satisfying its customers. Predicting customer churn has become crucial not only for accurately identifying potential churners, but also for understanding customer behavior.

To remain competitive, telecom companies must constantly improve their services, including customer support and pricing plans, and leverage the power of data analytics to gain a competitive edge in the market. By utilizing targeted data analytics, telecom companies can better understand their customers' behavior and preferences, helping them to maintain or enhance their position in the highly competitive marketplace.

Impact in Business:

Businesses rely heavily on telecommunications as a critical tool for effective communication with their customers and delivering top-quality customer service. Telecommunications facilitates seamless collaboration among employees, regardless of their location, whether they work remotely or locally.

The impact of telecommunications extends beyond individual businesses, as it influences how people connect and conduct business on a global scale. In particular, reliable and timely communication is crucial to maintaining a company's brand reputation, productivity, and overall success. By leveraging the power of telecommunications, businesses can improve their operational efficiency, streamline their processes, and enhance customer satisfaction, ultimately leading to greater success and growth.

Problem Statement:

One of the biggest challenges that large companies face is customer churn, which has a direct impact on their revenue, particularly in the telecommunications industry. As a result, companies are constantly seeking ways to predict potential churn and prevent it from happening. Identifying the factors that contribute to customer churn is crucial for taking necessary measures to reduce it.

By analyzing customer behavior and preferences, companies can gain insights into the factors that contribute to churn, such as poor service quality, pricing, or lack of customer support. Armed with this information, companies can take proactive steps to address these issues and implement measures to retain their customers. In this way, understanding the factors that increase customer churn is essential for any company that wants to remain competitive and profitable in the long term.

Dataset Information:

Target Variable:

FEATURE	DATA TYPE	DESCRIPTION
CHURN	Object	Detecting which customers are likely to leave a service or to cancel a subscription to a service

Features Understanding:

Feature	DATA TYPE	Description
Customer ID	Integer	Primary key of the record.
Churn	Object	Detecting which customers are likely to leave a service or to cancel a subscription to a service
Monthly Revenue	Float	Revenue of each Customer
Monthly Minutes	Float	Number of Minutes call spoken by Customer
Total Recurring Charge	Float	The Charges for the Service
Director Assisted Calls	Float	When we call an operator to request a telephone number
Overage Minutes	Float	Count of Call used over duration to particular post-paid cell phone plan
Roaming Calls	Float	The ability to get access to the Internet when away from home at the price of a local call or at a charge considerably less than the regular long-distance charges.
Three-way Calls	Float	A way of adding a third party to your conversation without the assistance of a telephone operator.

Dropped Calls	Float	Count of Phone calls gets disconnected somehow from the cellular network.
Blocked Calls	Float	Count of Telephone call that is unable to connect to an intended recipient.
Un-answered Calls	Float	Count of Calling that an individual perceives but is not currently pursuing.

Received Calls	Float	Number of calls received by the customer.
Out bound Calls	Float	Call initiated by the call centre agent to customer on behalf of client to know the target customer behaviour and needs.
Inbound Calls	Float	In inbound calls, call-centre or customer-care receives call from customer with issues and questions.
Peak Calls in Out	Float	Amount of time period with fewer calls than are handled in a busy period.
Call Forwarding Calls	Float	Count of CallsForwarded by user.
Dropped Blocked Calls	Float	Number of VM messages customer currently has on the server.
Call Waiting Calls	Float	Duration of call-in waiting period
Months In Service	Integer	Number of months customer using service.
Unique Subs	Integer	subscription of different networks
Active Subs	Integer	subscription of the networks that are active or in usage.
Service Area	Object	Network service area
Handsets	Integer	Count of Handset with user
Handset Models	Float	Count of Handsets are used to Contact one to one.

Feature name	Data Type	Description
Age HH1	Float	User aged below 45
Age HH2	Float	User aged above 45
Children in HH	Integer	Whether there are Children in House hold
Handset Refurbished	Object	Are the handsets refurbished or not
Handset Web Capable	Object	Are the handsets capable of internet connectivity
Truck Owner	Object	Is the user a Truck Owner
RV Owner	Object	Is the user an RV owner
Home Ownership	Object	Is the house the user is staying, his own
Buys Visa Mail Order	Object	Does the user buy Visa Mail order
Responds to Mail Offers	Object	Does the user respond to Mail offers
Opt-out Mailings	Object	Did he opt out of the mail offers sent to him
Non-US-Travel	Object	Does the user travel to other countries
Owns-Computer	Object	Does he have a computer or not
Has-Credit Card	Object	Does he have a credit card or not
Retention Calls	Integer	No of Retention Calls
Retention Offers Accepted	Integer	Customers accepting retaining the retaining offers given by the company.
New Cell phone User	Object	Number of customers buying new cell phone.
Not New cell phone User	Object	Number of customers uses existing cell phone
Referrals Made by Subscriber	Integer	Referrals made by the existing customer to the other customer.
Income Group	Integer	The column talks about the customer saying to which category the customer belongs to.
Adjustments To Credit Rating	Integer	Rating Scale

Handset Price	Object	Its amount paid by the customer for his cell phone.
Made call to retention team	Object	User call to Retention in same company
Credit Rating	Object	Credit card user rating (out of 7)
PrimzCode	object	Grouping of regions according to users
Occupation	Object	Occupation of User
Marital status	Object	Marital Status Indicated by Yes/No/Unknown

Dataset Information:

Data is taken from Kaggle (Telecom(churn))

No. of features: 56

No. of records: 51047

Target Column: churn

Redundant columns: Service area, Customer Id.

Understanding the Data:

Checking Shape of Data:

```
df.shape
```

```
(51047, 58)
```

```
# As per the data we have 51047 observations and 58 variables are there
```

DATA EXPLORATION (EDA)

Summary of Dataset:

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
CustomerID	51047.000000	3201956.877818	116905.561666	3000002.000000	3100632.000000	3201534.000000	3305376.000000	3399994.000000
MonthlyRevenue	50891.000000	58.834492	44.507336	-6.170000	33.610000	48.460000	71.065000	1223.380000
MonthlyMinutes	50891.000000	525.653416	529.871063	0.000000	158.000000	366.000000	723.000000	7359.000000
TotalRecurringCharge	50891.000000	46.830088	23.848871	-11.000000	30.000000	45.000000	60.000000	400.000000
DirectorAssistedCalls	50891.000000	0.895229	2.228546	0.000000	0.000000	0.250000	0.990000	159.390000
OverageMinutes	50891.000000	40.027785	96.588076	0.000000	0.000000	3.000000	41.000000	4321.000000
RoamingCalls	50891.000000	1.236244	9.818294	0.000000	0.000000	0.000000	0.300000	1112.400000
PercChangeMinutes	50680.000000	-11.547908	257.514772	-3875.000000	-83.000000	-5.000000	66.000000	5192.000000
PercChangeRevenues	50680.000000	-1.191985	39.574915	-1107.700000	-7.100000	-0.300000	1.600000	2483.500000
DroppedCalls	51047.000000	6.011489	9.043955	0.000000	0.700000	3.000000	7.700000	221.700000
BlockedCalls	51047.000000	4.085672	10.946905	0.000000	0.000000	1.000000	3.700000	384.300000
UnansweredCalls	51047.000000	28.288981	38.876194	0.000000	5.300000	16.300000	36.300000	848.700000
CustomerCareCalls	51047.000000	1.868999	5.096138	0.000000	0.000000	0.000000	1.700000	327.300000
ThreewayCalls	51047.000000	0.298638	1.168277	0.000000	0.000000	0.000000	0.300000	66.000000
ReceivedCalls	51047.000000	114.800121	166.485896	0.000000	8.300000	52.800000	153.500000	2692.400000
OutboundCalls	51047.000000	25.377715	35.209147	0.000000	3.300000	13.700000	34.000000	644.300000
InboundCalls	51047.000000	8.178104	16.665878	0.000000	0.000000	2.000000	9.300000	519.300000
PeakCallsInOut	51047.000000	90.549515	104.947470	0.000000	23.000000	62.000000	121.300000	2090.700000
OffPeakCallsInOut	51047.000000	67.650790	92.752699	0.000000	11.000000	35.700000	88.700000	1474.700000
DroppedBlockedCalls	51047.000000	10.158003	15.555284	0.000000	1.700000	5.300000	12.300000	411.700000
CallForwardingCalls	51047.000000	0.012277	0.594168	0.000000	0.000000	0.000000	0.000000	81.300000
CallWaitingCalls	51047.000000	1.840504	5.585129	0.000000	0.000000	0.300000	1.300000	212.700000
MonthsInService	51047.000000	18.756264	9.800138	6.000000	11.000000	16.000000	24.000000	61.000000
UniqueSubs	51047.000000	1.532157	1.223384	1.000000	1.000000	1.000000	2.000000	196.000000
ActiveSubs	51047.000000	1.354340	0.675477	0.000000	1.000000	1.000000	2.000000	53.000000
Handsets	51046.000000	1.805646	1.331173	1.000000	1.000000	1.000000	2.000000	24.000000
HandsetModels	51046.000000	1.558751	0.905932	1.000000	1.000000	1.000000	2.000000	15.000000
CurrentEquipmentDays	51046.000000	380.545841	253.801982	-5.000000	205.000000	329.000000	515.000000	1812.000000
AgeHH1	50138.000000	31.338127	22.094635	0.000000	0.000000	36.000000	48.000000	99.000000
AgeHH2	50138.000000	21.144142	23.931368	0.000000	0.000000	0.000000	42.000000	99.000000
RetentionCalls	51047.000000	0.037201	0.206483	0.000000	0.000000	0.000000	0.000000	4.000000
RetentionOffersAccepted	51047.000000	0.018277	0.142458	0.000000	0.000000	0.000000	0.000000	3.000000
ReferralsMadeBySubscriber	51047.000000	0.052070	0.307592	0.000000	0.000000	0.000000	0.000000	35.000000
IncomeGroup	51047.000000	4.324524	3.138236	0.000000	0.000000	5.000000	7.000000	9.000000
AdjustmentsToCreditRating	51047.000000	0.053911	0.383147	0.000000	0.000000	0.000000	0.000000	25.000000

Interpretation:

1. Count of all the columns are different.So we can assume that their is nullvalues
2. We can see difference in both mean and median.So the data is not normally distributed
3. Also there is presence of outliers because the difference between min and max is more

Check the Data Type:

Check the data type of each variable. If the data type is not as per the data definition, change the data type.

```
Churn                       object
MonthlyRevenue             float64
MonthlyMinutes             float64
TotalRecurringCharge       float64
DirectorAssistedCalls      float64
OverageMinutes             float64
RoamingCalls               float64
PercChangeMinutes          float64
PercChangeRevenues         float64
DroppedCalls               float64
BlockedCalls               float64
UnansweredCalls            float64
CustomerCareCalls          float64
ThreewayCalls              float64
ReceivedCalls              float64
OutboundCalls              float64
InboundCalls               float64
PeakCallsInOut             float64
OffPeakCallsInOut          float64
DroppedBlockedCalls        float64
CallForwardingCalls        float64
CallWaitingCalls           float64
MonthsInService              int64
UniqueSubs                   int64
ActiveSubs                   int64
Handsets                   float64
HandsetModels              float64
CurrentEquipmentDays       float64
AgeHH1                     float64
AgeHH2                     float64
ChildrenInHH                object
HandsetRefurbished          object
HandsetWebCapable           object
TruckOwner                  object
RVOwner                     object
Homeownership               object
BuysViaMailOrder            object
RespondsToMailOffers        object
OptOutMailings              object
NonUSTravel                 object
OwnsComputer                object
HasCreditCard               object
RetentionCalls               int64
RetentionOffersAccepted      int64
NewCellphoneUser            object
NotNewCellphoneUser         object
ReferralsMadeBySubscriber    int64
IncomeGroup                  int64
OwnsMotorcycle              object
AdjustmentsToCreditRating    int64
HandsetPrice                object
MadeCallToRetentionTeam     object
CreditRating                object
PrizmCode                   object
Occupation                  object
MaritalStatus               object
dtype: object
```

Drop unnecessary columns

```
df.drop(['ServiceArea','CustomerID'], axis=1, inplace=True)
```

```
df.shape
```

```
(51047, 56)
```

Data Cleaning

Missing Values Treatment:

Missing values plays a prominent role in the dataset. Generally, we can drop the columns or rows depending the percentage of missing values. We can also replace the missing values with optimum values. In order to perform such operations, we will first look into the overall missing values in each column using the below python code.

```
dtyp = pd.DataFrame(df.dtypes).rename(index={0:'column Type'})
Total=df.isnull().sum().sort_values(ascending=True)
Percent = (df.isnull().sum()*100/df.isnull().count()).sort_values(ascending=True)
missing_data = pd.concat([dtyp,Total, Percent], axis = 1, keys = ['Data type','Total', 'Percentage of Missing Values'])
missing_data
```

	Data type	Total	Percentage of Missing Values
	0	0	1
Churn	object	0	0.000000
MonthlyRevenue	float64	156	0.305601
MonthlyMinutes	float64	156	0.305601
TotalRecurringCharge	float64	156	0.305601
DirectorAssistedCalls	float64	156	0.305601
OverageMinutes	float64	156	0.305601
RoamingCalls	float64	156	0.305601
PercChangeMinutes	float64	367	0.718945
PercChangeRevenues	float64	367	0.718945
DroppedCalls	float64	0	0.000000
BlockedCalls	float64	0	0.000000
UnansweredCalls	float64	0	0.000000
CustomerCareCalls	float64	0	0.000000
ThreewayCalls	float64	0	0.000000
ReceivedCalls	float64	0	0.000000
OutboundCalls	float64	0	0.000000
InboundCalls	float64	0	0.000000
PeakCallsInOut	float64	0	0.000000
OffPeakCallsInOut	float64	0	0.000000
DroppedBlockedCalls	float64	0	0.000000
CallForwardingCalls	float64	0	0.000000
CallWaitingCalls	float64	0	0.000000
MonthsInService	int64	0	0.000000
UniqueSubs	int64	0	0.000000
ActiveSubs	int64	0	0.000000
Handsets	float64	1	0.001959
HandsetModels	float64	1	0.001959
CurrentEquipmentDays	float64	1	0.001959
AgeHH1	float64	909	1.780712
AgeHH2	float64	909	1.780712
ChildrenInHH	object	0	0.000000
HandsetRefurbished	object	0	0.000000

We shall replace the missing values with median value.

```
# select the columns to replace missing values
cols_to_replace = ['AgeHH1', 'AgeHH2', 'PercChangeMinutes', 'PercChangeRevenues', 'MonthlyMinutes',
                   'TotalRecurringCharge', 'DirectorAssistedCalls', 'OverageMinutes', 'RoamingCalls',
                   'MonthlyRevenue', 'HandsetModels', 'Handsets', 'CurrentEquipmentDays']

# replace missing values with the median of each column
for col in cols_to_replace:
    median_val = df[col].median()
    df[col] = df[col].fillna(median_val)
```

Let us now consider each variable separately for missing value treatment.

```
# As we can see their is no null values
```

```
df['HandsetPrice'].unique() # Checking unique values
```

```
array(['30', 'Unknown', '10', '80', '150', '300', '40', '200', '100',
       '130', '60', '400', '240', '250', '180', '500'], dtype=object)
```

```
df['HandsetPrice'] = df['HandsetPrice'].replace('Unknown',np.nan)
```

```
df['HandsetPrice'].unique()
```

```
array(['30', nan, '10', '80', '150', '300', '40', '200', '100', '130',
       '60', '400', '240', '250', '180', '500'], dtype=object)
```

```
median_price = df['HandsetPrice'].median()
median_price
```

```
60.0
```

```
df['HandsetPrice'] = df['HandsetPrice'].fillna(median_price)
```

```
#Here we replace the word unknown from the 'HandsetPrice' column by using median. Because their is outliers, so we didn't
#use the mean
```

```
# convert numerical variables to categorical (object)
# use astype() to change the data type

# change the data type of 'HandsetPrice'
df['HandsetPrice'] = df['HandsetPrice'].astype(int)
```

Interpretation: Now, all the variables have the correct data type

Duplicate Values Check:

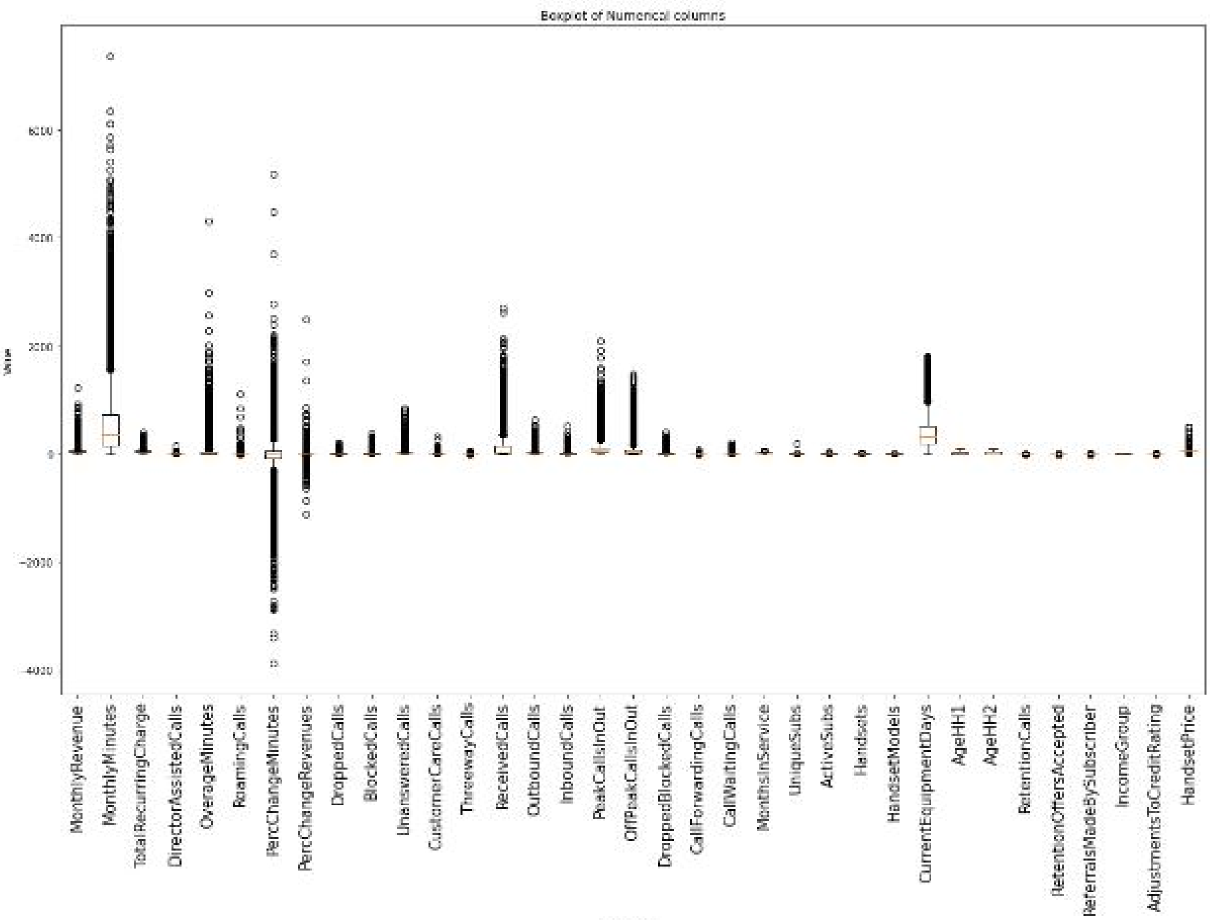
```
#checking for duplicate values
print(df1.duplicated().sum())
print(' ')
print(f'Dataset have {df1.duplicated().sum()} duplicate values.')
```

```
0

Dataset have 0 duplicate values.
```

Outlier Analysis:

```
num_vars = ['MonthlyRevenue', 'MonthlyMinutes', 'TotalRecurringCharge', 'DirectorAssistedCalls', 'OverageMinutes',
            'RoamingCalls', 'PercChangeMinutes', 'PercChangeRevenues', 'DroppedCalls', 'BlockedCalls', 'UnansweredCalls',
            'CustomerCareCalls', 'ThreewayCalls', 'ReceivedCalls', 'OutboundCalls', 'InboundCalls', 'PeakCallsInOut',
            'OffPeakCallsInOut', 'DroppedBlockedCalls', 'CallForwardingCalls', 'CallWaitingCalls', 'MonthsInService',
            'UniqueSubs', 'ActiveSubs', 'Handsets', 'HandsetModels', 'CurrentEquipmentDays', 'AgeHH1', 'AgeHH2',
            'RetentionCalls', 'RetentionOffersAccepted', 'ReferralsMadeBySubscriber', 'IncomeGroup',
            'AdjustmentsToCreditRating', 'HandsetPrice']
df_num = df[num_vars]
plt.boxplot(df_num.values)
plt.xticks(range(1,len(num_vars)+1),num_vars,rotation = 'vertical', fontsize = 15)
plt.title('Boxplot of Numerical columns')
plt.xlabel('Variable')
plt.ylabel('Value')
plt.show()
```



Inference: By Visualizing above boxplot, we can see that all the Features have potential outliers and some features there are extreme values as well.

Outliers: Outliers is an observation which deviates so much from the other observations, that it become suspicious that it was generated by different mechanism or simply by error

Extreme Values: Extreme Values is an observation with value at the boundaries of the domain

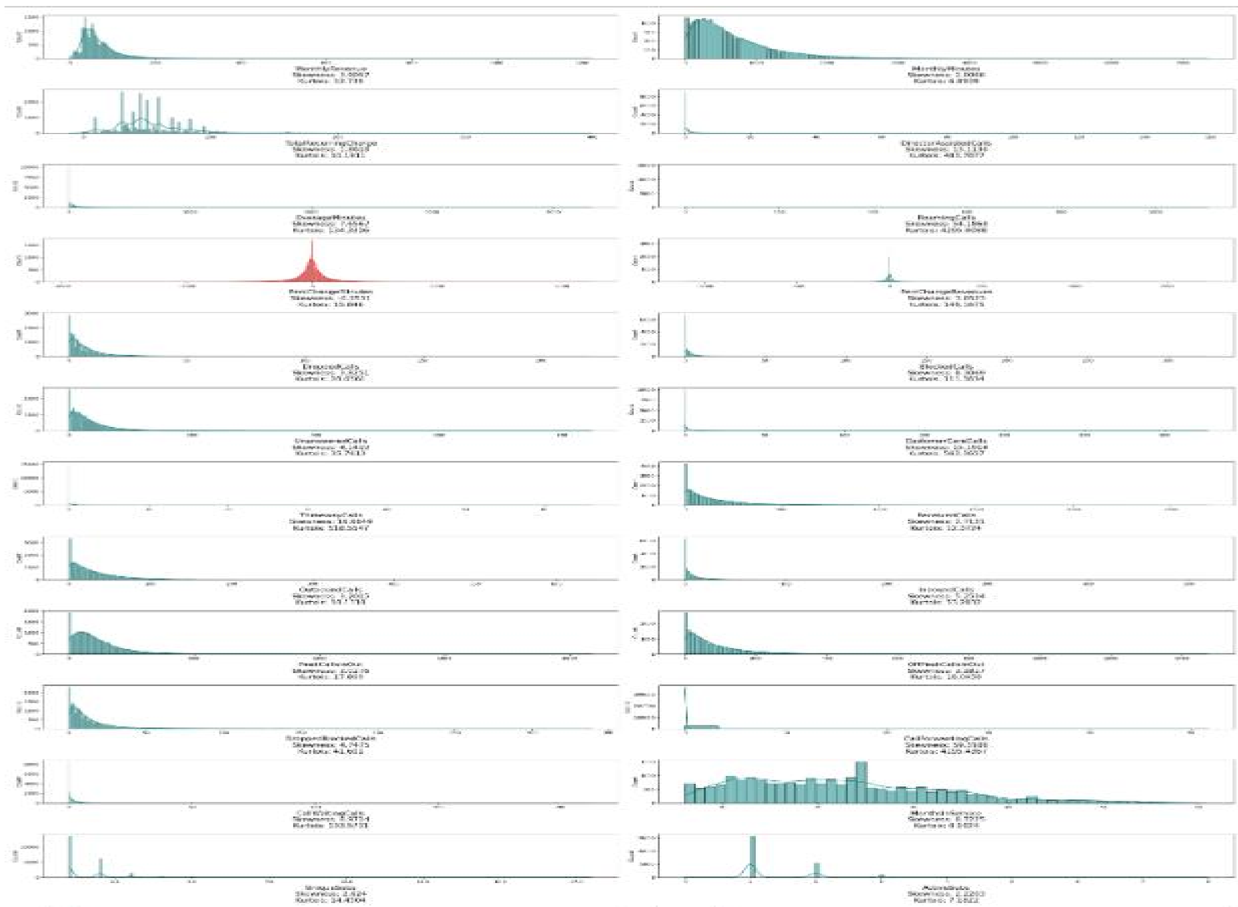
Reason for outliers exist in the data:

1. Variability in the Data
2. An experimental measurement errors

Impact of outliers on Dataset:

1. It causes various problem during statistical analysis.
2. It effects the mean and standard deviation.

<u>Skewness Before Transformation:</u>



Inference: Here by visualizing dist plot we can see that the Features plotted in Teal colour are positively skewed and Features plotted in red colour are Negatively Skewed.

--: To reduce the impact of skewness we can use various transformation techniques here we are using box cox transformation

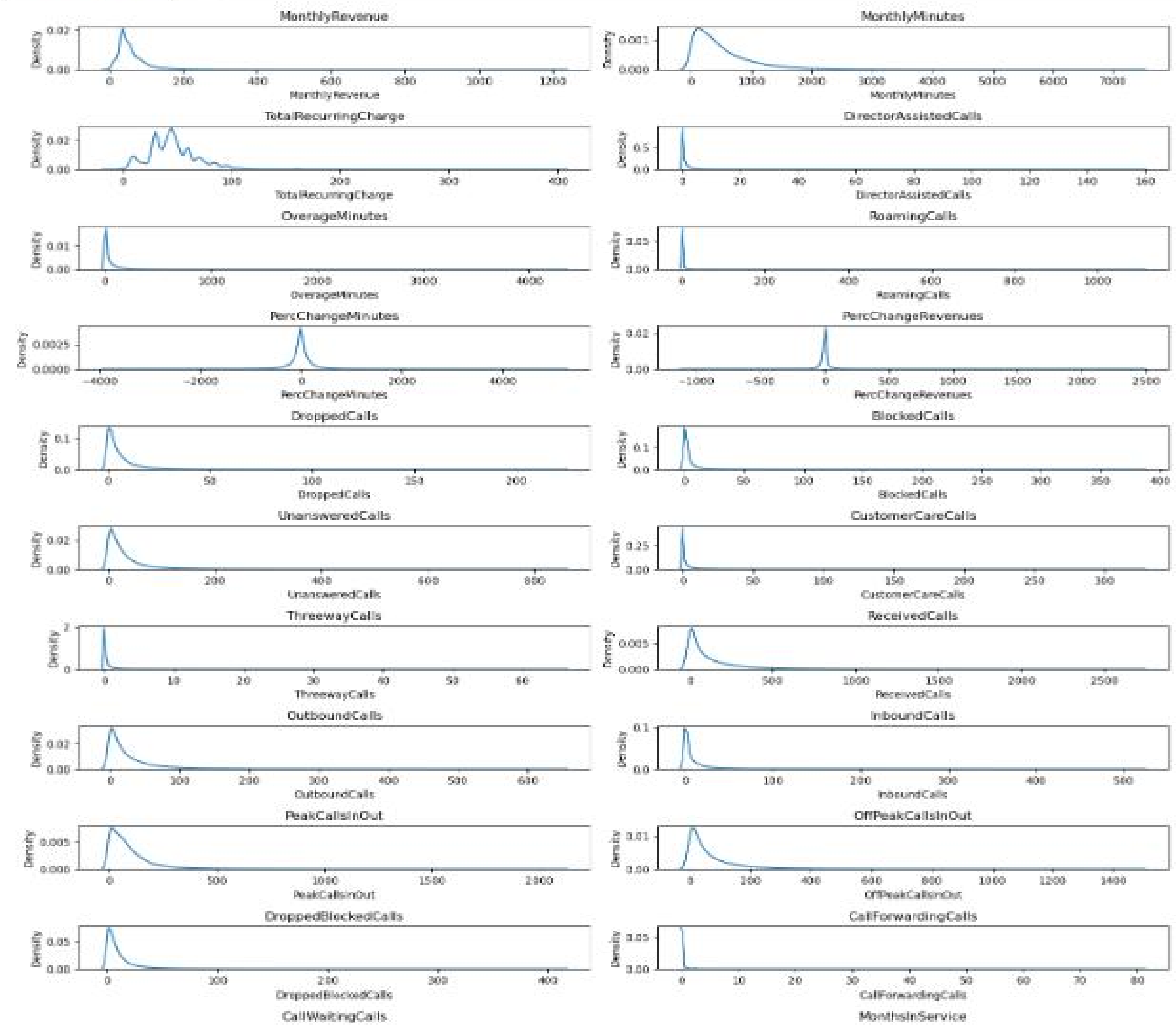
Descriptive Analysis (EDA)

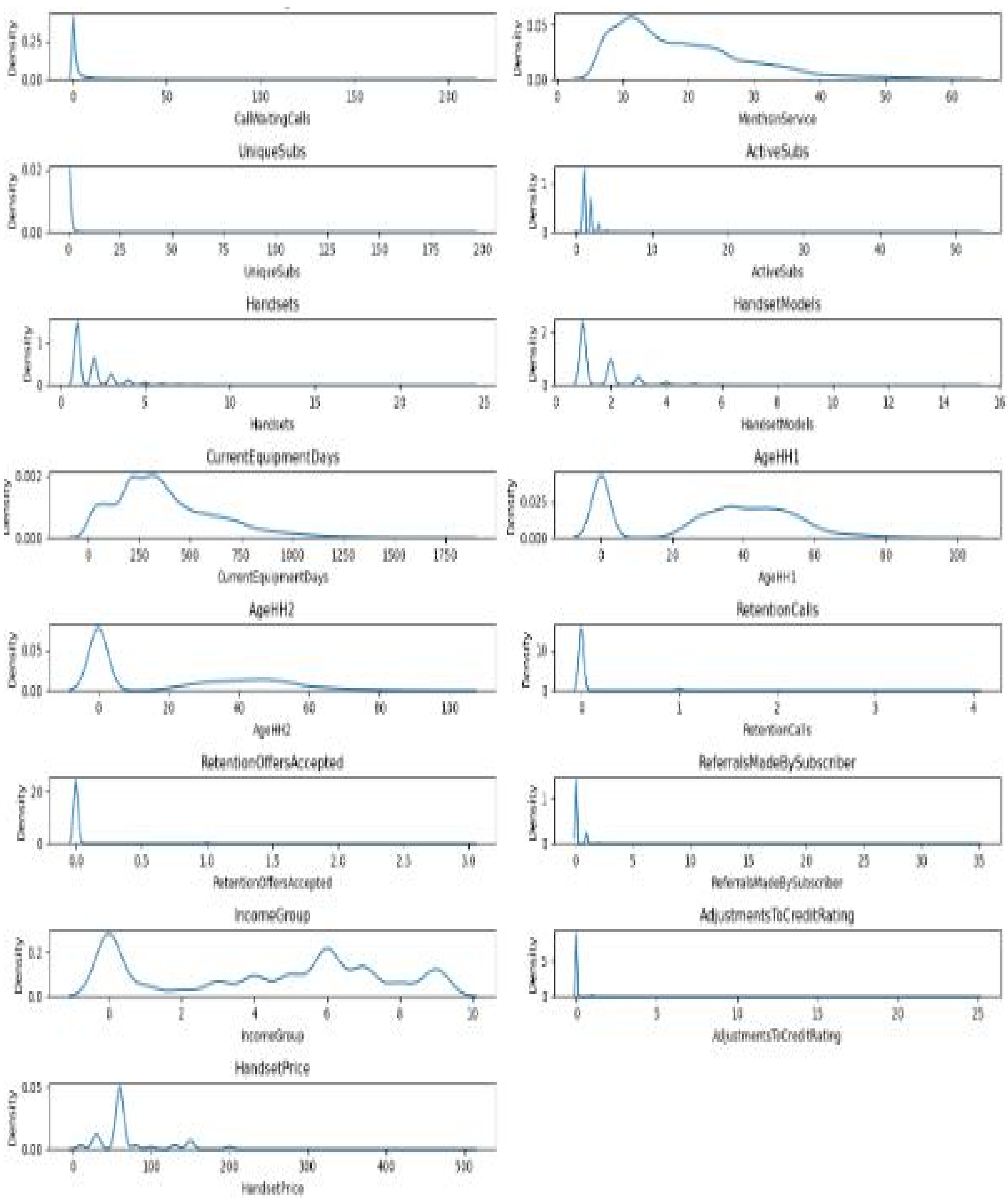
Univariate Analysis:

Numerical Columns Visualization:

```
# Kde Plot for Numerical features

plt.figure(figsize=(15,25),dpi=100)
n=1
for i in df_num:
    plot=plt.subplot(18,2,n)
    n+=1
    plt.title(i)
    sns.kdeplot(data=df[i])
    plt.tight_layout()
    annot_percent(plot)
```



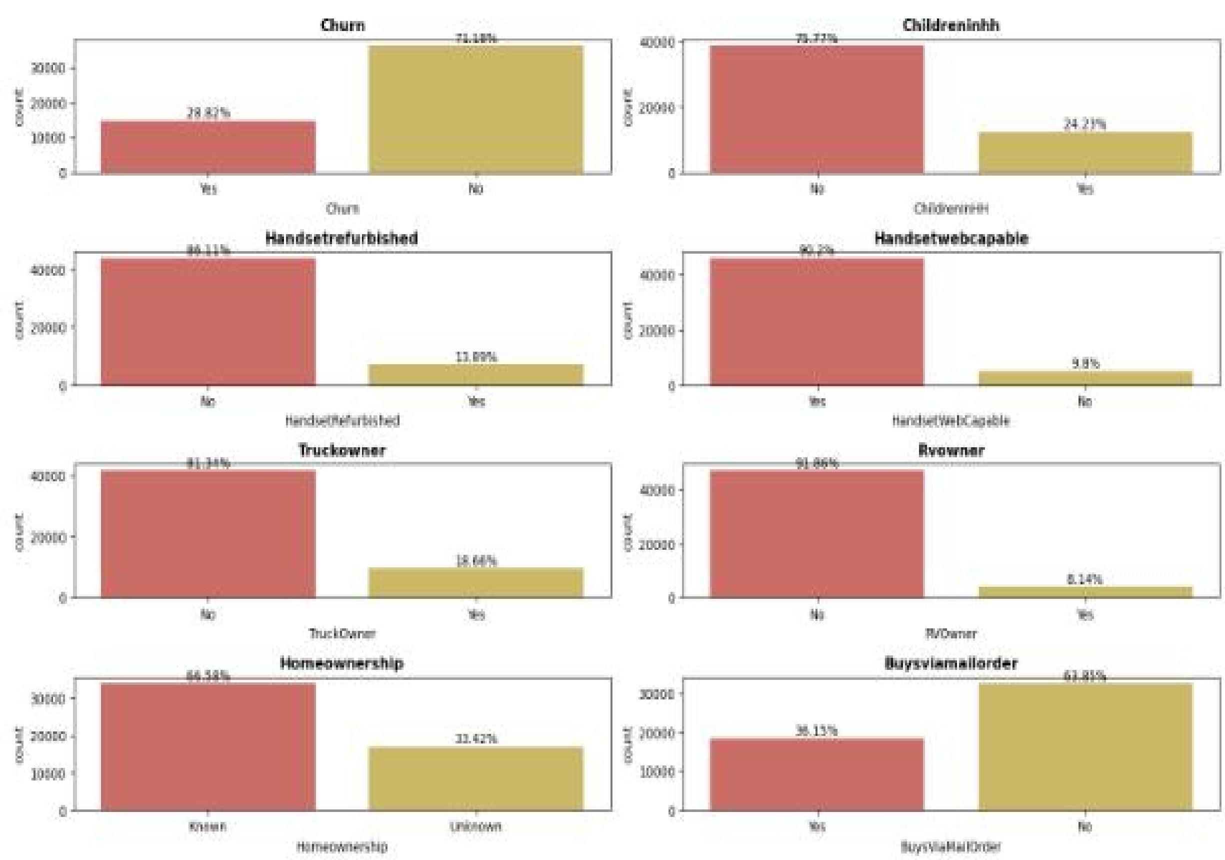

CallWaitingCalls
MonthsInService
UniqueSubs
ActiveSubs
Handsets
HandsetModels
CurrentEquipmentDays
AgeHH1
AgeHH2
RetentionCalls
RetentionOffersAccepted
ReferralsMadeBySubscriber
IncomeGroup
AdjustmentsToCreditRating
HandsetPrice
Density

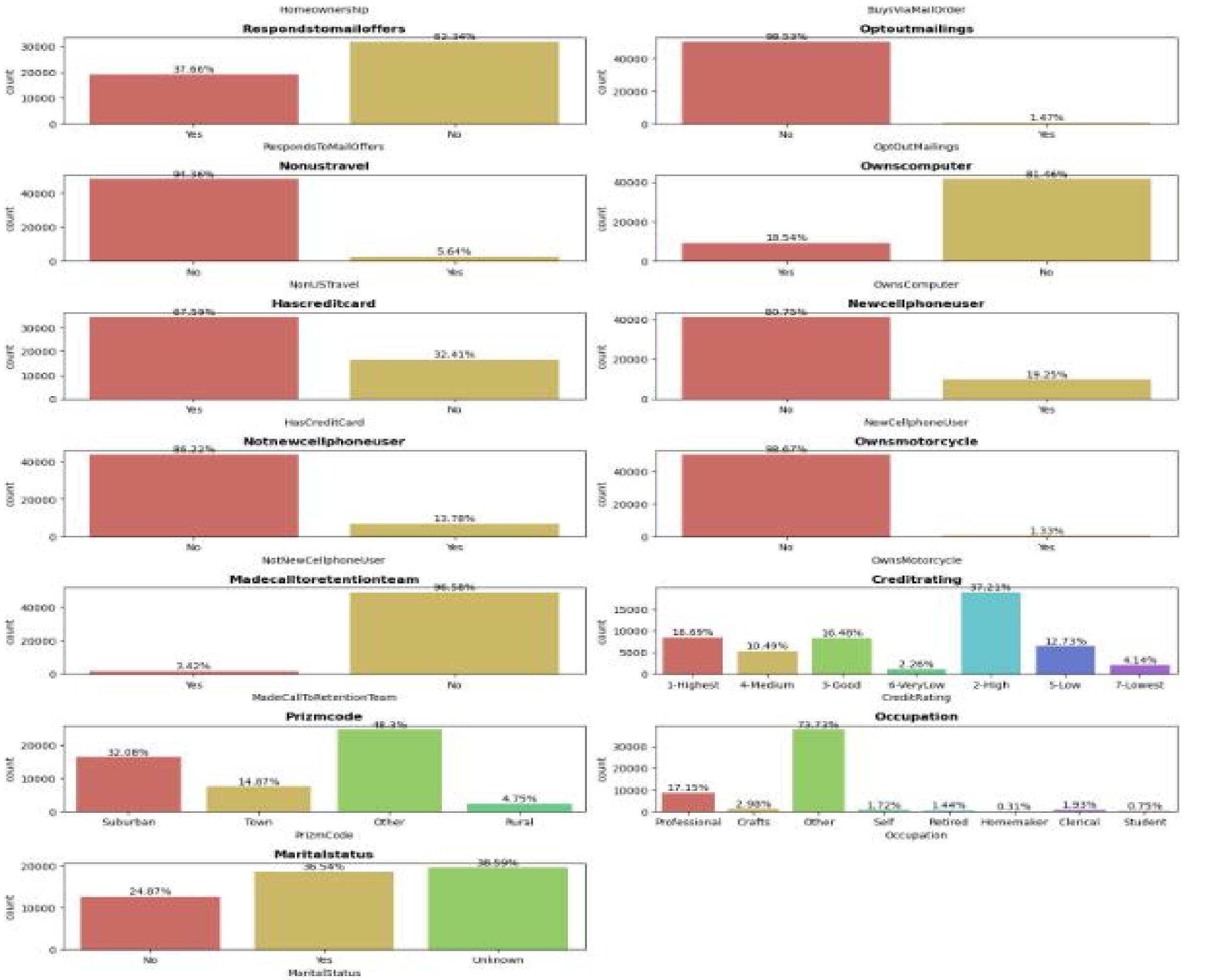
Categorical Columns Visualization:

```python
def annot_percent(axes):
    for p in plot.patches:
        total = sum(p.get_height() for p in plot.patches)/100
        percent = round((p.get_height()/total),2)
        x = p.get_x() + p.get_width()/2
        y = p.get_height()
        plot.annotate(f'{percent}%', (x, y), ha='center', va='bottom')
```

```python
#plotting countplot for some categorical variable

plt.figure(figsize=(15,25),dpi=100)
n=1
for i in df_cat:
    plot=plt.subplot(12,2,n)
    n+=1
    sns.countplot(df[i] ,palette=sns.color_palette("hls", 8))
    plt.title(f'{i.title()}',weight='bold')
    plt.tight_layout()
    annot_percent(plot)
```





Obeservations

1) Churn Over 27 percent of people in the data have churned.

2) Handseter capable More than 90 percent of the people in the data have internet support on their phone.

3) More than 65 percent of them don't have a credit card

4) Less than 2 percent of them own a motorcycle

5) More than half of the people's handset price is unknown

6) Over 70 percent of the data has occupations other than the ones mentioned.

7) Martial status of 60 percent of the data is known out of which, 26 percent are not married. The rest are unknown.

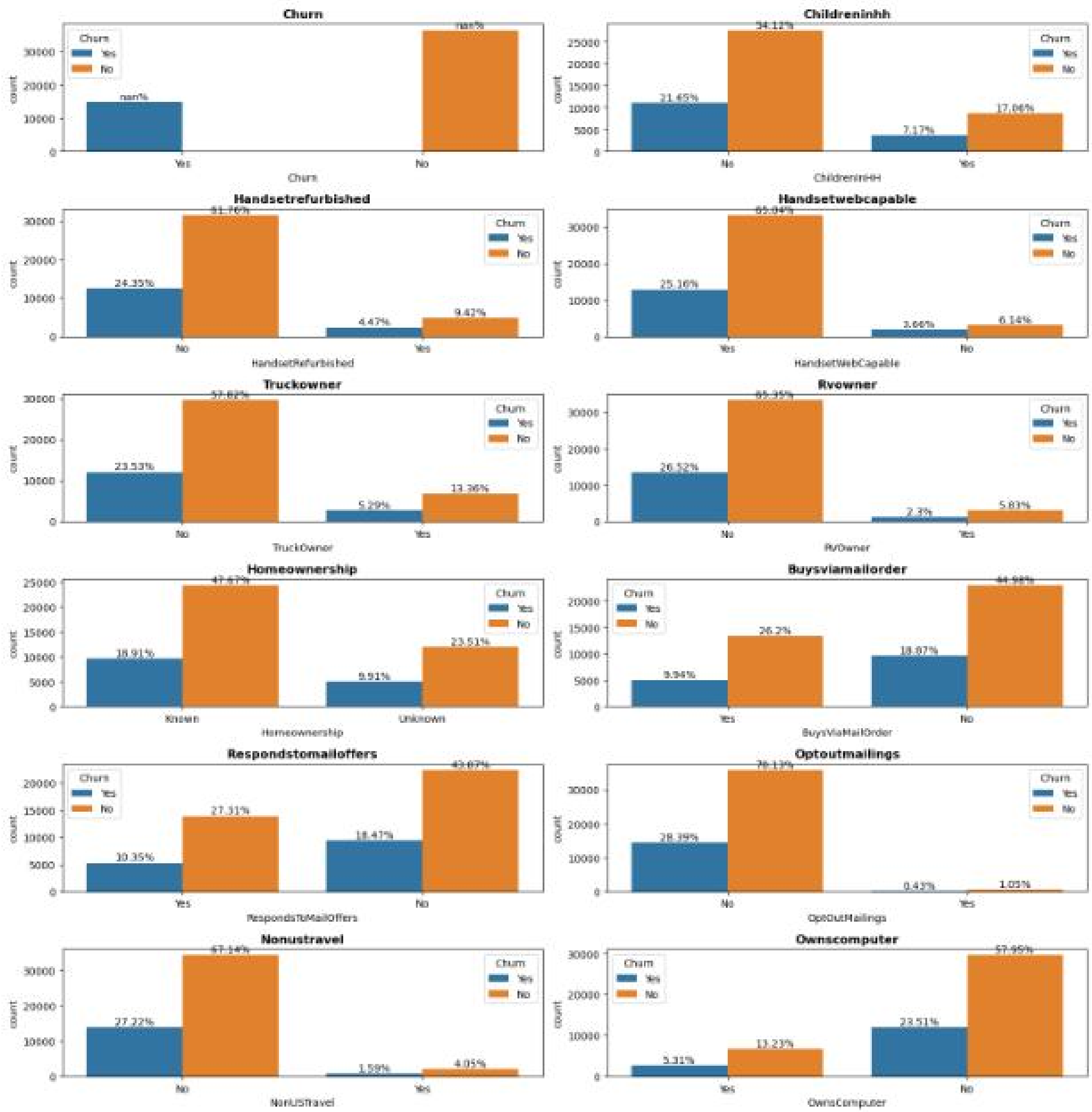
6) Over 70 percent of the data has occupations other than the ones mentioned.

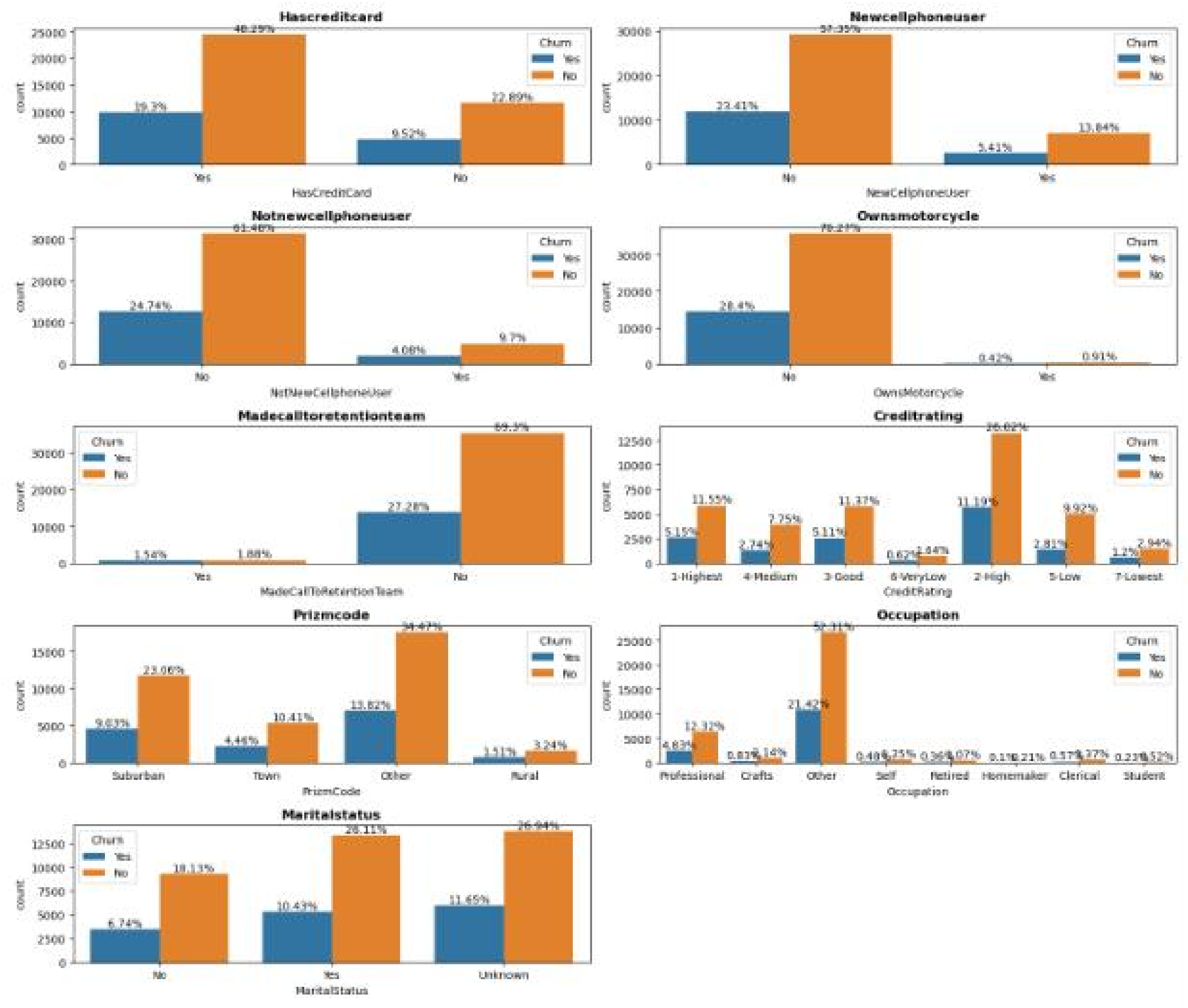
7) Martial status of 60 percent of the data is known out of which, 25 percent are not married. The rest are unknown.

Bivariate Analysis:

Bivariate Analysis on Categorical - Categorical

```
plt.figure(figsize=(15,30),dpi=100)
n=1
for i in df_cat:
    plot=plt.subplot(12,2,n)
    n+=1
    sns.countplot(df[i],hue= df['Churn'])
    plt.title(f'{i.title()}',weight='bold')
    plt.tight_layout()
    annot_percent(plot)
```





observation:

1. In Handset web capability over 25% of people who have churned has more than 90% of Internet capability on their phone.

2. Less than 6% of people who own new phone have churned.

3. Data shows that people who have Credit Cards are more likely to Churn

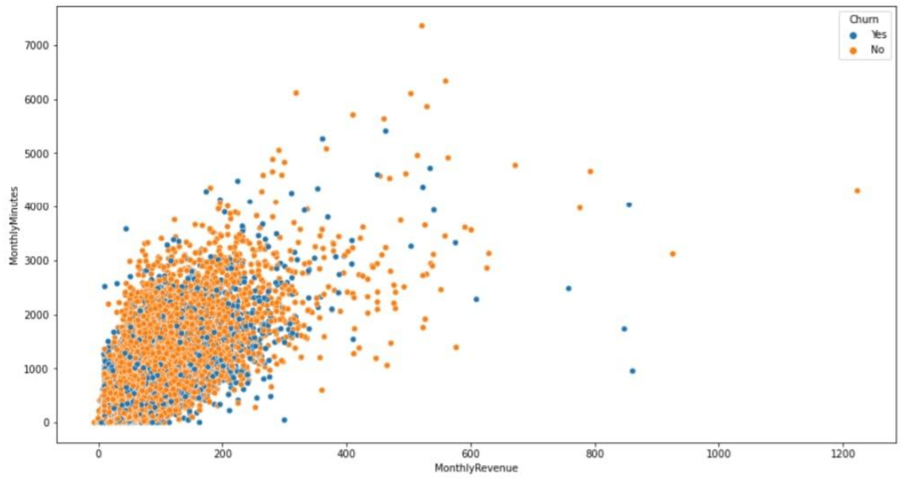
4. Marital Status of people churning is independent

5. People who have responded mail offer are less likely to churn

Multivariate Analysis:

```
sns.scatterplot(x="MonthlyRevenue", y="MonthlyMinutes",hue='Churn', data=df1)
```

<AxesSubplot:xlabel='MonthlyRevenue', ylabel='MonthlyMinutes'>

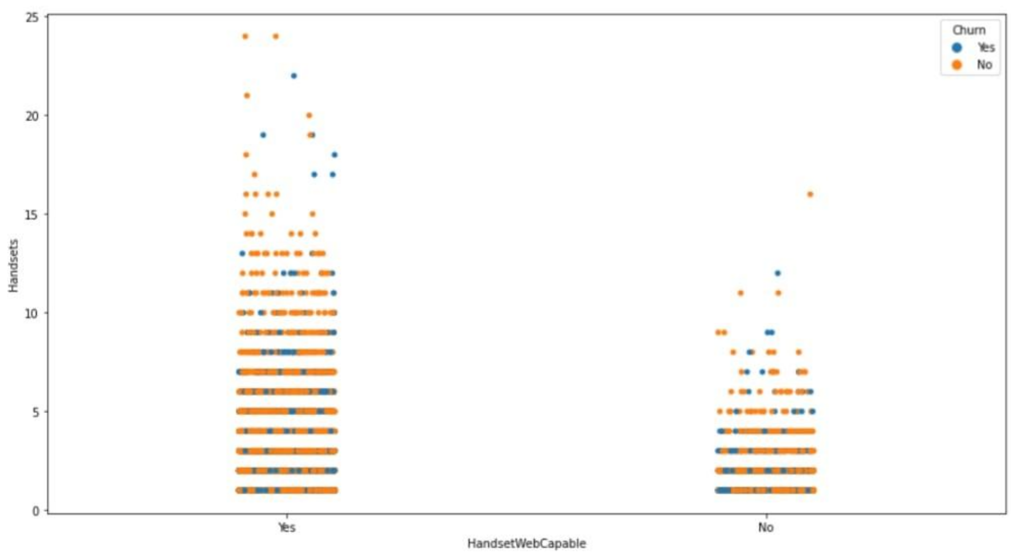


Observation:

According to plot, as Monthly Revenue Increases, then number of Monthly Minutes increases,but we could not draw any conclusion on churn.

```
sns.stripplot(y=df1["Handsets"],x='HandsetWebCapable',hue='Churn', data=df1,orient='v')
```

<AxesSubplot:xlabel='HandsetWebCapable', ylabel='Handsets'>



Observation:

As the number of handset Increases, with this certain percentage peoples are more likely to churn.

greatlearning
Learning for Life

Statistics (Stats)

	Feature	Statistical Test	P-Value	Inference
0	MonthlyRevenue	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
1	MonthlyMinutes	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
2	TotalRecurringCharge	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
3	DirectorAssistedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
4	OverageMinutes	kruskal wallis test	0.000009	Dependent numerical variable found after H-tes...
5	RoamingCalls	kruskal wallis test	0.922785	Independent numerical variable found after H-t...
6	PercChangeMinutes	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
7	PercChangeRevenues	kruskal wallis test	0.308102	Independent numerical variable found after H-t...
8	DroppedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
9	BlockedCalls	kruskal wallis test	0.000650	Dependent numerical variable found after H-tes...
10	UnansweredCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
11	CustomerCareCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
12	ThreewayCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
13	ReceivedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
14	OutboundCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
15	InboundCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
16	PeakCallsInOut	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
17	OffPeakCallsInOut	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
18	DroppedBlockedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
19	CallForwardingCalls	kruskal wallis test	0.311887	Independent numerical variable found after H-t...
20	CallWaitingCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
21	MonthsInService	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
22	UniqueSubs	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
23	ActiveSubs	kruskal wallis test	0.000003	Dependent numerical variable found after H-tes...
24	Handsets	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
25	HandsetModels	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
26	CurrentEquipmentDays	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
27	AgeHH1	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
28	AgeHH2	kruskal wallis test	0.000383	Dependent numerical variable found after H-tes...
29	RetentionCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
30	RetentionOffersAccepted	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
31	ReferralsMadeBySubscriber	kruskal wallis test	0.024863	Dependent numerical variable found after H-tes...
32	IncomeGroup	kruskal wallis test	0.026027	Dependent numerical variable found after H-tes...
33	AdjustmentsToCreditRating	kruskal wallis test	0.000646	Dependent numerical variable found after H-tes...
34	HandsetPrice	kruskal wallis test	0.242433	Independent numerical variable found after H-t...
35	ChildrenInHH	Chi-Square Test for Independence	0.030195	Dependent categorical variable found after Chi...
36	HandsetRefurbished	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
37	HandsetWebCapable	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
38	TruckOwner	Chi-Square Test for Independence	0.324832	Independent categorical variable found after C...
39	RVOwner	Chi-Square Test for Independence	0.500851	Independent categorical variable found after C...
40	Homeownership	Chi-Square Test for Independence	0.004931	Dependent categorical variable found after Chi...

	Feature	Statistical Test	P-Value	Inference
41	BuysViaMailOrder	Chi-Square Test for Independence	0.000002	Dependent categorical variable found after Chi...
42	RespondsToMailOffers	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
43	OptOutMailings	Chi-Square Test for Independence	0.837419	Independent categorical variable found after C...
44	NonUSTravel	Chi-Square Test for Independence	0.562279	Independent categorical variable found after C...
45	OwnsComputer	Chi-Square Test for Independence	0.810924	Independent categorical variable found after C...
46	HasCreditCard	Chi-Square Test for Independence	0.071275	Independent categorical variable found after C...
47	NewCellphoneUser	Chi-Square Test for Independence	0.141394	Independent categorical variable found after C...
48	NotNewCellphoneUser	Chi-Square Test for Independence	0.106749	Independent categorical variable found after C...
49	OwnsMotorcycle	Chi-Square Test for Independence	0.089071	Independent categorical variable found after C...
50	MadeCallToRetentionTeam	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
51	CreditRating	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
52	PrizmCode	Chi-Square Test for Independence	0.000295	Dependent categorical variable found after Chi...
53	Occupation	Chi-Square Test for Independence	0.253384	Independent categorical variable found after C...
54	MaritalStatus	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...

We have used Chi-Square Test for Independence to test whether the categorical variables are independent or not.

$H0$: The variables are independent.

$H1$: The variables are not independent (i.e., variables are dependent).

We have used Jarque-bera test to check the normality of data

$H0$: The data is normally distributed.

$H1$: The data is not normally distributed.

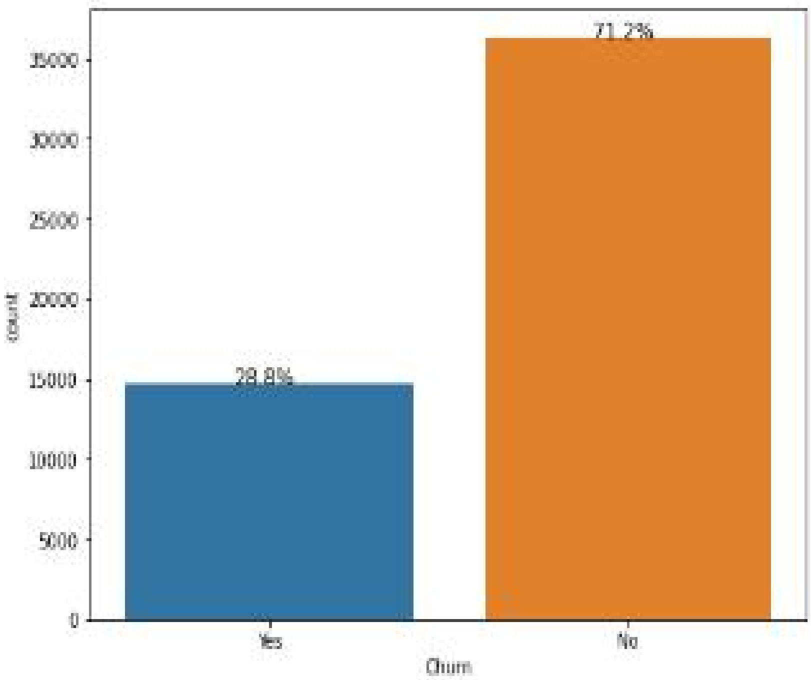
We found that data is not normal therefore we use Kruskal Wallis test to check its dependency on the target variable

Class Imbalance and its Treatment:

```
# Create count plot
plt.figure(figsize=(8,6))
sns.countplot(x='Churn', data=df)

# Add percentage text to each bar
total = len(df['Churn'])
for p in plt.gca().patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width() / 2 - 0.1
    y = p.get_y() + p.get_height()
    plt.gca().annotate(percentage, (x, y), size=12)

# Show plot
plt.show()
```



Here we can see that our target variable is slightly imbalanced, and we are not going to use oversampling techniques like smote.

greatlearning
Learning for Life

Check of Multicollinearity:

	VIF_Factor	Features
0	375.668313	DroppedBlockedCalls
1	151.715464	BlockedCalls
2	131.663762	DroppedCalls
3	30.821246	MonthlyRevenue
4	20.151614	HandsetModels
5	18.863594	TotalRecurringCharge
6	14.101876	Handsets
7	13.164124	MonthsInService
8	12.337600	MonthlyMinutes
9	11.787734	ActiveSubs
10	7.843933	OffPeakCallsInOut
11	7.803766	PeakCallsInOut
12	7.791593	OverageMinutes
13	7.023598	CurrentEquipmentDays
14	6.931251	AgeHH1
15	6.433891	UniqueSubs
16	6.288748	ReceivedCalls
17	5.418122	OutboundCalls
18	4.915661	IncomeGroup
19	4.809451	HandsetPrice
20	4.015448	UnansweredCalls
21	3.214218	AgeHH2
22	3.188755	InboundCalls
23	2.729470	CallWaitingCalls
24	2.350160	RetentionCalls
25	2.308168	RetentionOffersAccepted
26	1.635248	PercChangeMinutes
27	1.626432	RoamingCalls
28	1.621602	PercChangeRevenues
29	1.558542	DirectorAssistedCalls
30	1.517198	CustomerCareCalls
31	1.265618	ThreewayCalls
32	1.089253	AdjustmentsToCreditRating
33	1.041547	ReferralsMadeBySubscriber
34	1.002325	CallForwardingCalls

Transformation:

Transformation is a process that can be used to change the scale of the original data to get more accurate results.We used Power transformation, as we can see that there is large number of outliers present so we use Yeo-Johnson transformation technique to reduce the outliers and make the data more normally distributed.

```
from sklearn.preprocessing import PowerTransformer
PT=PowerTransformer()
for i in num_f.columns:
    num_f[i]=PT.fit_transform(num_f[[i]])
```

```
num_f.head()
```

sistedCalls	OverageMinutes	RoamingCalls	PercChangeMinutes	PercChangeRevenues	DroppedCalls	BlockedCalls	UnansweredCalls	CustomerCareCa
-0.044197	-0.995504	-0.621914	-0.566549	-0.450622	-0.889336	-0.306586	-0.568253	-0.824
-0.912074	-0.995504	-0.621914	0.018886	0.088432	-1.200278	-1.216280	-1.013639	-0.824
-0.912074	-0.995504	-0.621914	0.026624	0.088432	-1.515686	-1.216280	-1.760480	-0.824
1.170112	-0.995504	-0.621914	0.655061	0.284291	2.228625	1.290204	1.349529	1.501
-0.912074	-0.995504	-0.621914	0.034384	0.083251	-1.515686	-1.216280	-1.760480	-0.824

Logistic Regression (Base Model)

Build a full logistic model on a training dataset.

```
# build the model on train data (x_train and y_train)
# use fit() to fit the logistic regression model
logreg = sm.Logit(y_train,x_train).fit()

# print the summary of the model
print(logreg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                35475
Model:                          Logit   Df Residuals:                    35415
Method:                           MLE   Df Model:                           59
Date:                Thu, 11 Aug 2022   Pseudo R-squ.:                 0.03456
Time:                        16:19:23   Log-Likelihood:                -20526.
converged:                      False   LL-Null:                       -21261.
Covariance Type:            nonrobust   LLR p-value:                2.191e-268
------------------------------------------------------------------------------
```

Interpretation: The Pseudo R-squ. obtained from the above model summary is the value of McFadden's R-squared. This value can be obtained from the formula:

McFadden's R-squared = $1-(Log-Likelihood/\text{LL}-\text{Null})$

Where,
Log-Likelihood: It is the maximum value of the log-likelihood function
LL-Null: It is the maximum value of the log-likelihood function for the model containing only the intercept

1. The LLR p-value is less than 0.05, implies that the model is significant.

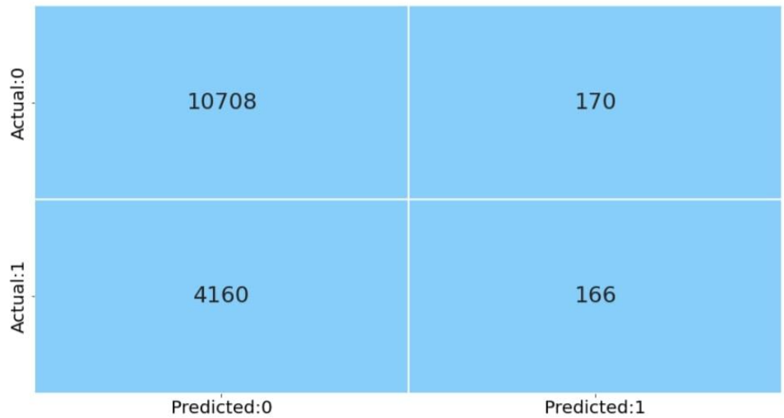
Cox & Snell R-squared: The convergence of the logistic model can be determined by the R-squared value. It is one of the types of Pseudo R-square.

2. The maximum of Cox & Snell R-squared is always less than 1. By above model Cox & Snell R-squared is less than 1 i.e. (0.03456).

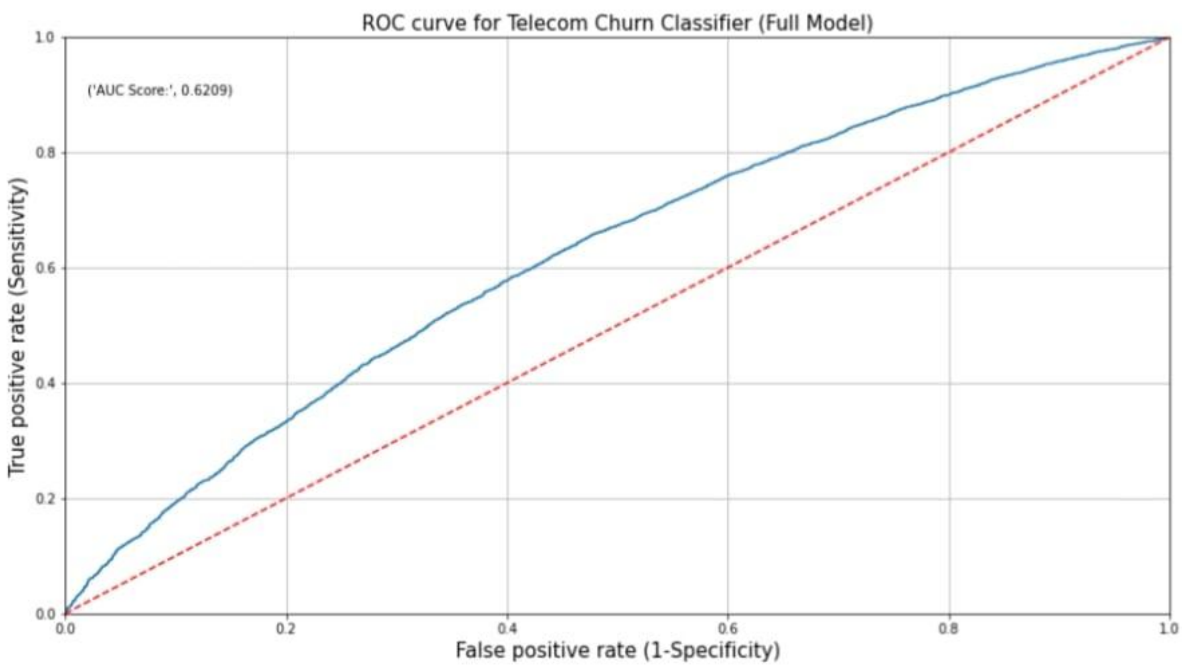
The AIC (Akaike Information Criterion) value:
It is a relative measure of model evaluation. It gives a trade-off between model accuracy and model complexity.

AIC: 41172.911

Confusion Matrix:



ROC Curve:



Inference:

- Interpretation: An AUC score of 0.6251 indicates that the model can correctly identify a higher proportion of positive instances than negative instances, but not by much. Therefore, the model may be biased towards the majority class, which in this case is the negative class labeled as 0. This could be due to an imbalance in the dataset, where there are more instances of the negative class than the positive class, causing the model to favor predicting the negative class more frequently.