Batch details	PGPDSE-FT Offline BLR April22
Team members	Rahul Patil Rajath J K Maheshreddy P Ganesh M Devan Thampi S P
Domain of Project	Retail
Proposed project title	Telecom Churn
Group Number	5
Team Leader	Rahul Patil
Mentor Name	Mr. Jayveer Nanda

Table of Contents:

PROJECT DETAILS

Overview

The telecommunication sector has become one of the main industries in developed countries. The technical progress and the increasing number of operators have raised the level of competition. Companies are working hard to survive in this competitive market depending on multiple strategies.

Customer churn is a considerable concern in service sectors with highly competitive services. On the other hand, predicting the customers who are likely to leave the company will represent potentially large additional revenue source if it is done in the early phase.

Industry Review

Introduction to domain:

Telecommunications are the means of electronic transmission of information over distances. The information may be in the form of telephone calls, data, text, images, or video. Today, telecommunications are used to organize more or less remote computer systems into telecommunications networks.

Nowadays, telecom industry faces fierce competition in satisfying its customers. The role of churn prediction system is not only restricted to accurately predict churners but also to interpret customer churn behavior.

To stay competitive, TELCOMs must continuously refine everything from customer service to plan pricing and use the power of highly targeted data analytics in helping the company secure or improve their standing in the highly competitive marketplace.

Impact in Business:

Telecommunications is an important tool for businesses. It enables companies to communicate effectively with customers and deliver high standards of customer service. Telecommunications is a key element in allowing employees to collaborate easily from wherever they are located, remote or local.

Telecommunications affects how people connect and do business on a global scale. For businesses, in particular, reliable and timely communication is the lifeblood of your company's brand reputation, productivity, and overall success.

Problem Statement:

Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenue of the companies, especially in the telecom field, companies are seeking to develop means to predict potential customer to churn. Therefore, finding factors that increase customer churn is important to take necessary actions to reduce this churn.

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 Calls Forwarded 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:

<u>Checking Shape of Data:</u>

```
# use 'shape' to check the dimension of data
df1.shape
```

```
(51047, 56)
```

DATA EXPLORATION (EDA)

Summary of Dataset:

```
df1.describe().T
```

	count	mean	std	min	25%	50%	75%	max
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	0.298838	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 features are not equal so we can say that there are missing values in the Dataset.
2. The difference Between mean and median of each variable is more, so we can say that data is not normally distributed.
3. The difference Between min and max of each variable is more, so we can say that Some of the features also contains potential outliers.

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
```

Recheck the Data type and conversions:

```
df1['HandsetPrice'] = df1['HandsetPrice'].str.replace('Unknown','500')
```

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

# change the data type of 'HandsetPrice'
df1['HandsetPrice'] = df1['HandsetPrice'].astype(int)
```

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

```
## Cleaning Categorical Columns
```

```
df1['HandsetPrice'] = df1['HandsetPrice'].replace({500:np.nan})
```

```
df1['HandsetPrice'].isnull().sum()
```

```
28990
```

```
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=5)
df1['HandsetPrice'] = imputer.fit_transform(df1[['HandsetPrice']])
```

```
df1['HandsetPrice'].isnull().sum()
```

```
0
```

Data Cleaning

Duplicate Values Check:

```
#checking for duplicate values
print(df12.duplicated().sum())
print(' ')
print(f'Dataset have {df1.duplicated().sum()} duplicate values.')
```

```
0

Dataset have 0 duplicate values.
```

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.

```
#checking missing values in the dataset
null_col=df1.columns[df1.isna().any()].to_list()
null_count=df1[null_col].isna().sum()
null_count_percent=(df1[null_col].isna().sum()/len(df1))*100
#creating a Dataframe of null count and null percent
null_data=pd.DataFrame({'Count':null_count,'Percentage':null_count_percent}).sort_values(by='Percentage',ascending=F
null_data.style.highlight_max(color = 'SkyBlue', subset = ['Count','Percentage'])
```

	Count	Percentage
AgeHH1	909	1.780712
AgeHH2	909	1.780712
PercChangeMinutes	367	0.718945
PercChangeRevenues	367	0.718945
MonthlyRevenue	156	0.305601
MonthlyMinutes	156	0.305601
TotalRecurringCharge	156	0.305601
DirectorAssistedCalls	156	0.305601
OverageMinutes	156	0.305601
RoamingCalls	156	0.305601
Handsets	1	0.001959
HandsetModels	1	0.001959
CurrentEquipmentDays	1	0.001959

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

```
#fetching the records associated with the nan values
df1[df1['AgeHH1'].isna()].head()
```

	Churn	MonthlyRevenue	MonthlyMinutes	TotalRecurringCharge	DirectorAssistedCalls	OverageMinutes	RoamingCalls	PercChangeMinutes
62	Yes	90.250000	952.000000	50.000000	0.000000	161.000000	0.000000	308.000000
87	No	122.000000	1806.000000	75.000000	15.590000	146.000000	1.000000	-156.000000
145	Yes	112.560000	1107.000000	83.000000	16.580000	0.000000	0.000000	-1107.000000
227	Yes	30.480000	47.000000	30.000000	0.500000	0.000000	0.000000	-7.000000
249	No	97.540000	482.000000	115.000000	0.000000	20.000000	0.000000	-186.000000

We shall replace the missing values with median value.

```
# obtain the mode value
df1.AgeHH1.median()
```

```
36.0
```

Interpretation: The median value of the data is '36'. We will now impute all the missing values with it.

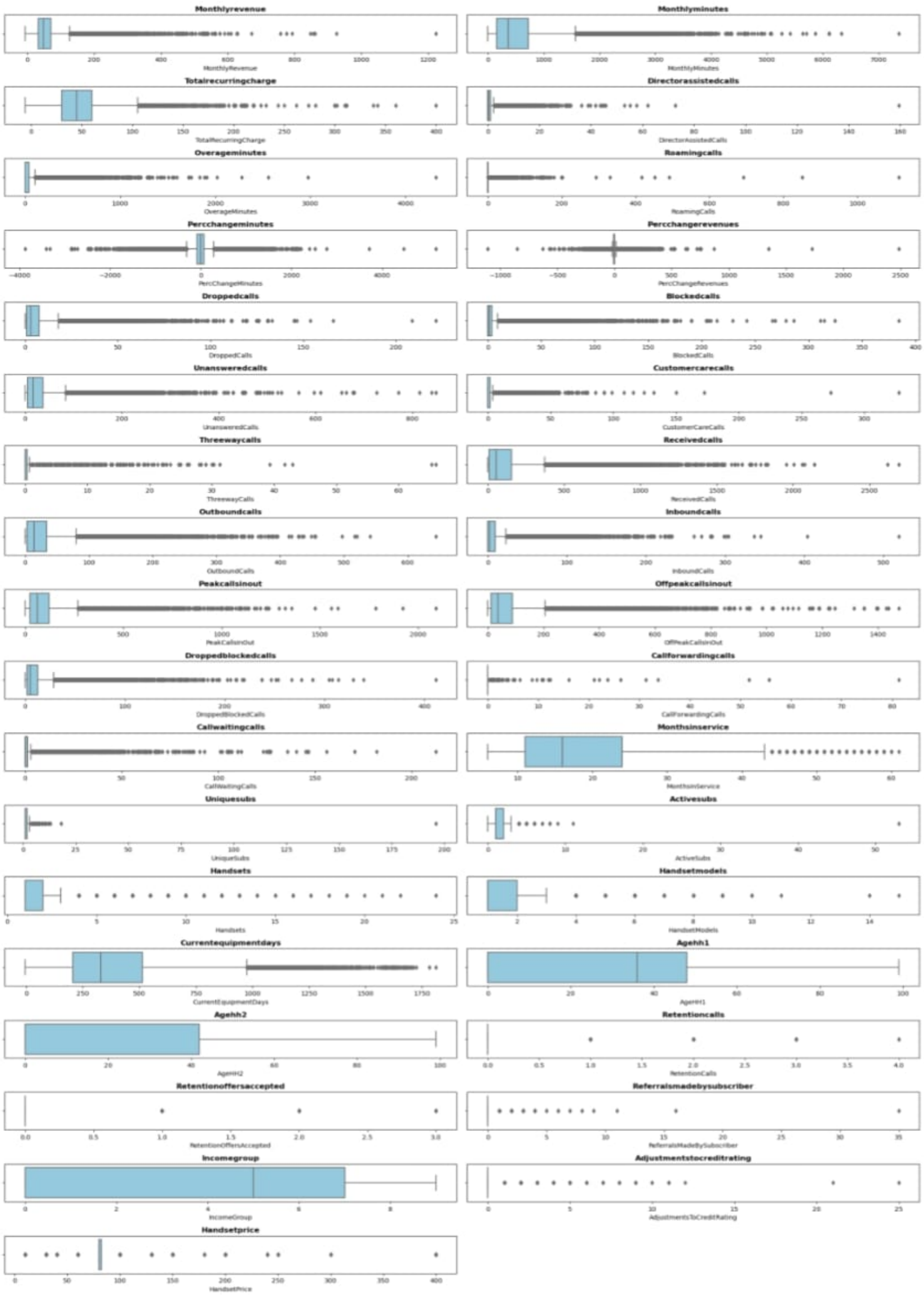
```
# replace all the missing values
df1.AgeHH1.replace(np.NaN,36 ,inplace = True)
```

The missing values have been replaced and let us do a recheck for the same.

```
df1.AgeHH1.isnull().sum()
```

```
0
```

Outlier Analysis:



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

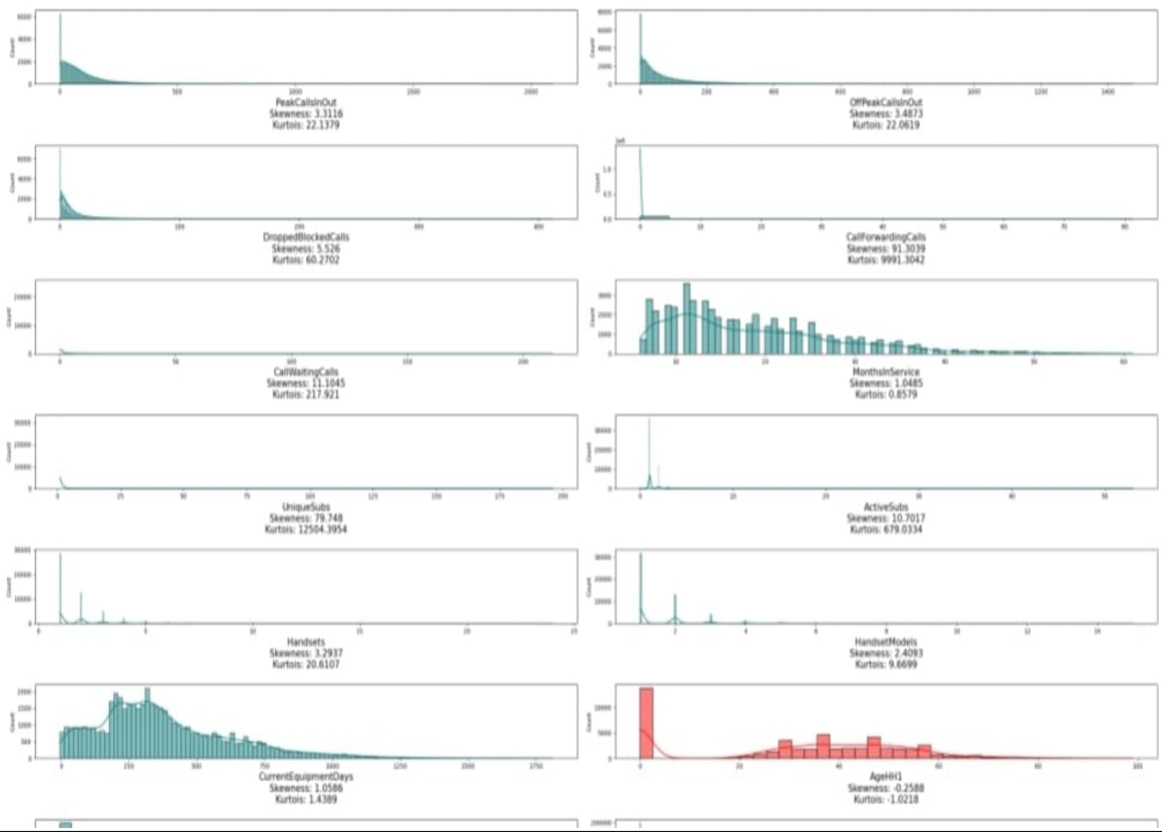
Reason for outliers exist in the data:

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.

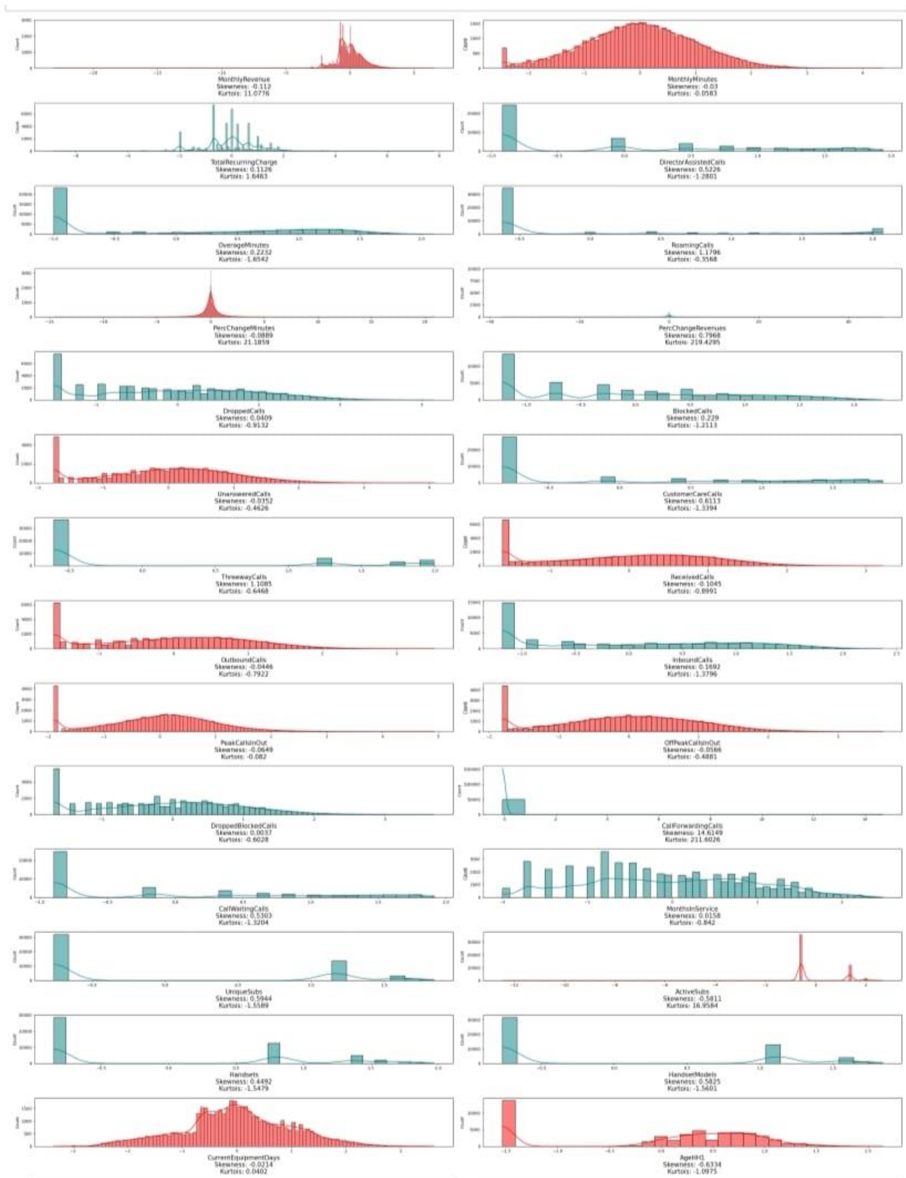
<u>Skewness Before Transformation:</u>



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

<u>Skewness After Transformation:</u>



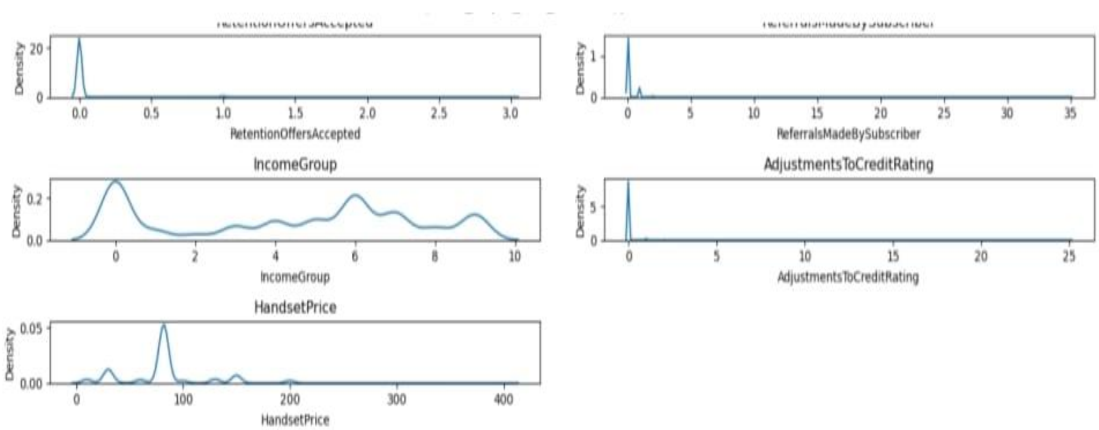
Inference: Here by visualizing dist plot we can observe that there is a reduction of skewness after Transformation.

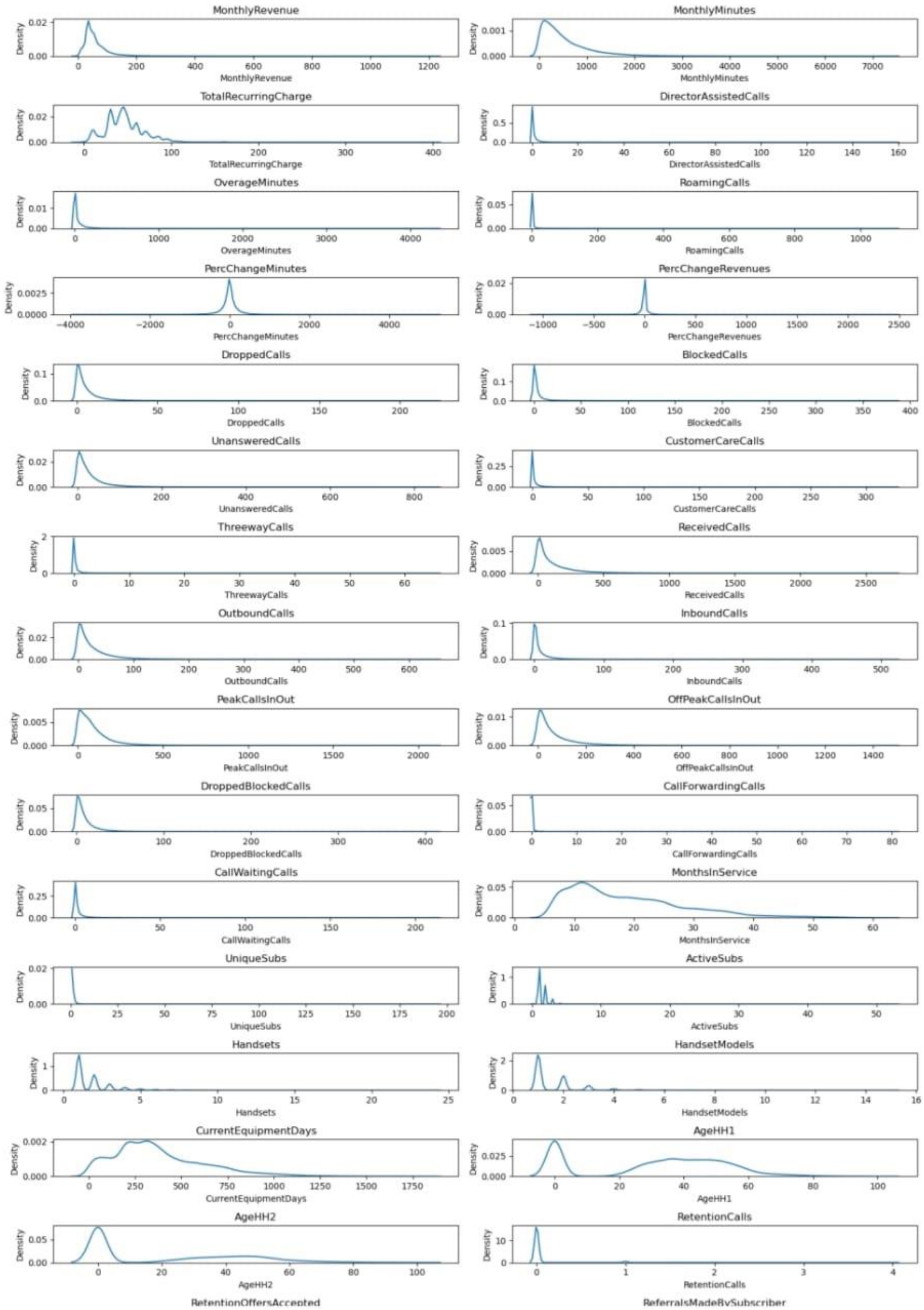
Descriptive Analysis (EDA)

Univariate Analysis:

Numerical Columns Visualization:

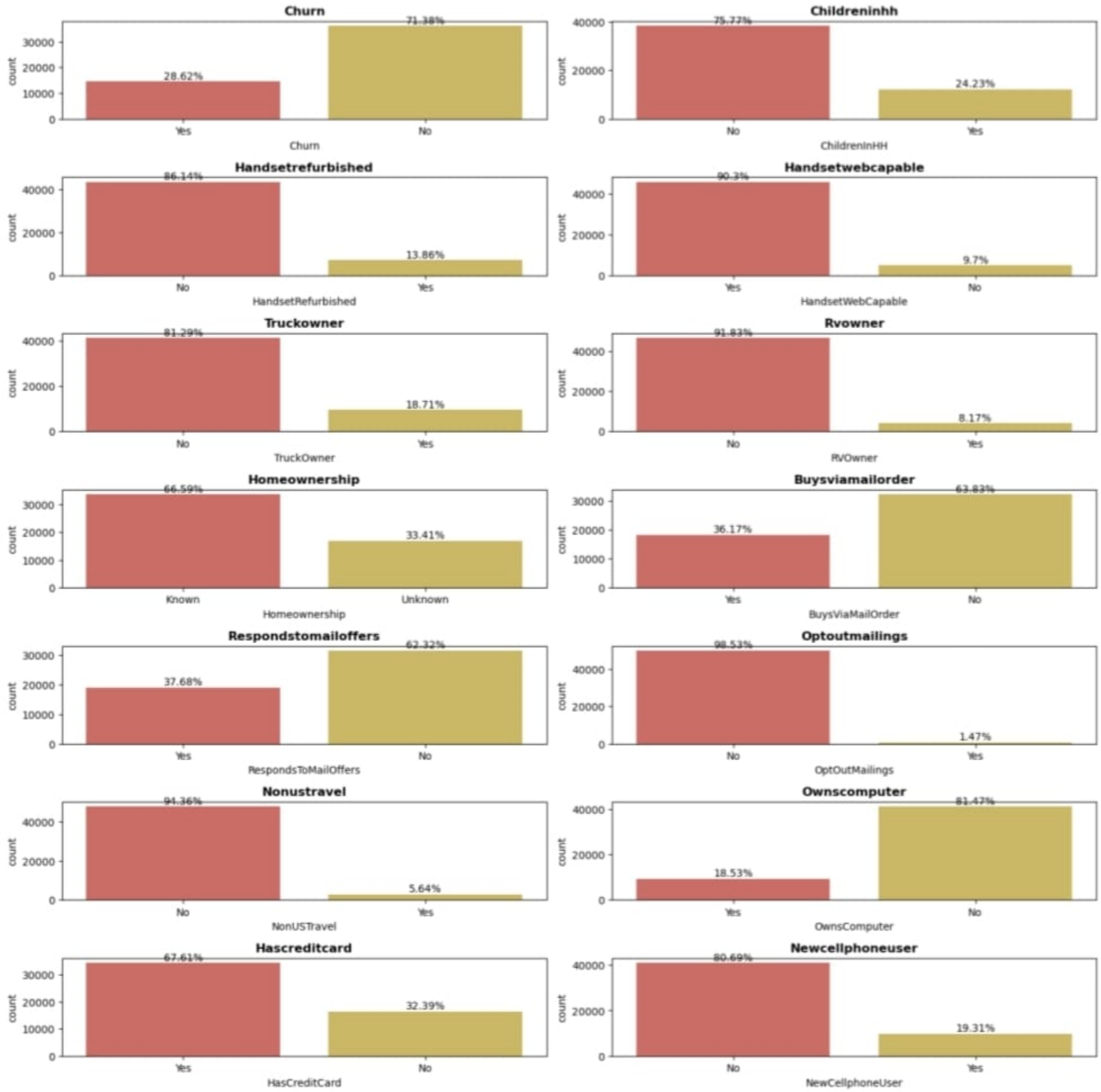
```
# Kde Plot for Numerical features

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=df1[i])
    plt.tight_layout()
    annot_percent(plot)
```

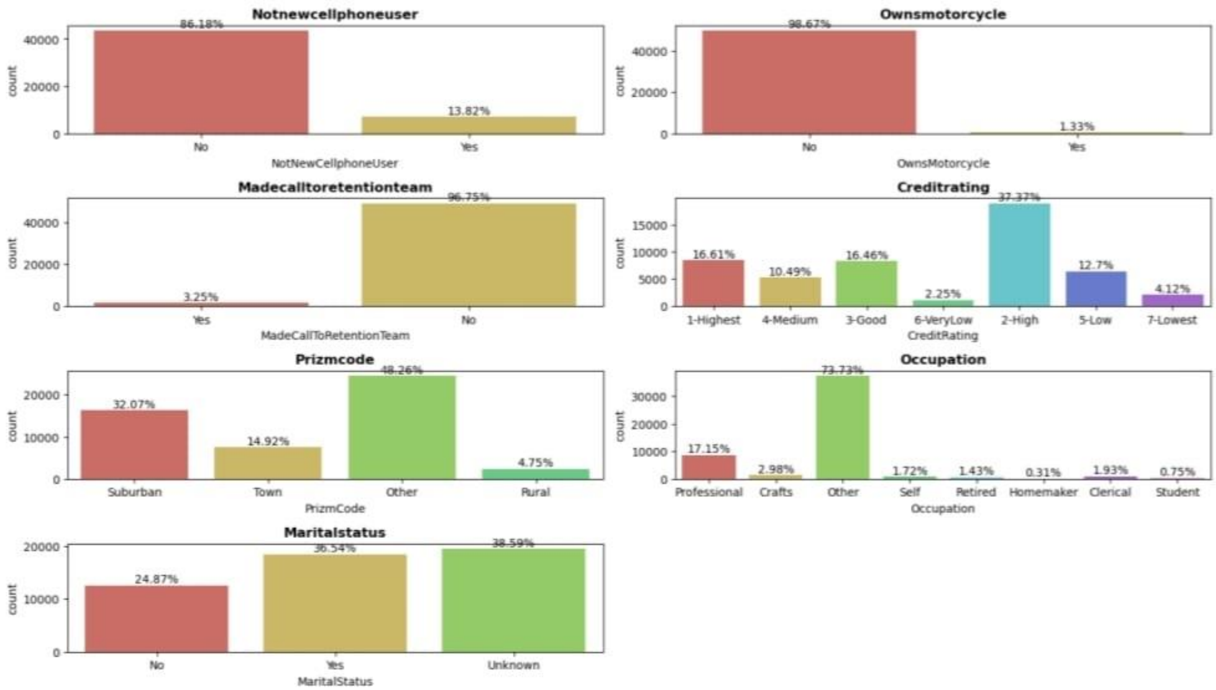



MonthlyRevenue
MonthlyMinutes
TotalRecurringCharge
DirectorAssistedCalls
OverageMinutes
RoamingCalls
PercChangeMinutes
PercChangeRevenues
DroppedCalls
BlockedCalls
UnansweredCalls
CustomerCareCalls
ThreewayCalls
ReceivedCalls
OutboundCalls
InboundCalls
PeakCallsInOut
OffPeakCallsInOut
DroppedBlockedCalls
CallForwardingCalls
CallWaitingCalls
MonthsInService
UniqueSubs
ActiveSubs
Handsets
HandsetModels
CurrentEquipmentDays
AgeHH1
AgeHH2
RetentionCalls
RetentionOffersAccepted
ReferralsMadeBySubscriber
Density

Categorical Columns Visualization:

```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(df1[i] ,palette=sns.color_palette("hls", 8))
    plt.title(f'{i.title()}',weight='bold')
    plt.tight_layout()
    annot_percent(plot)
```





Observations:

1) Churn Over 28 percent of people in the data have churned.

2) Handsetwebcapable 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

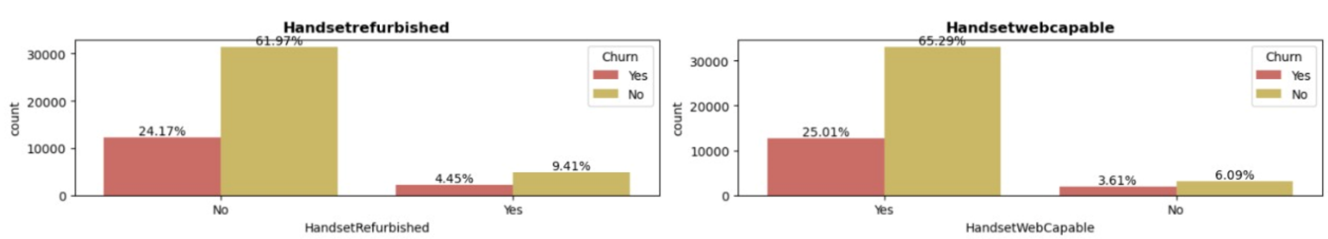
4) Less than 2 percent of them own a motorcycle

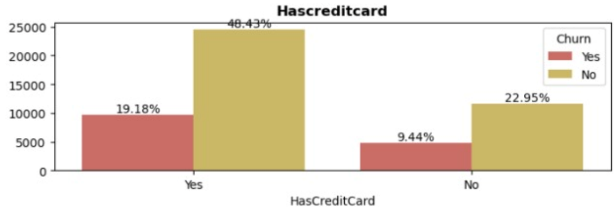
5) More than half of the people's handset price is unknown

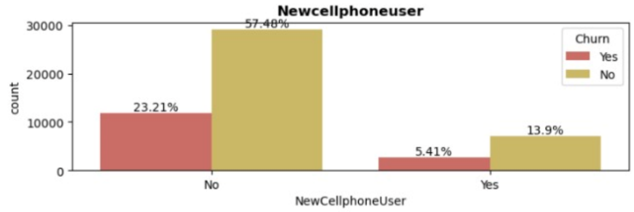
6) Over 70 percent of the data has occupations other than the ones mentioned.

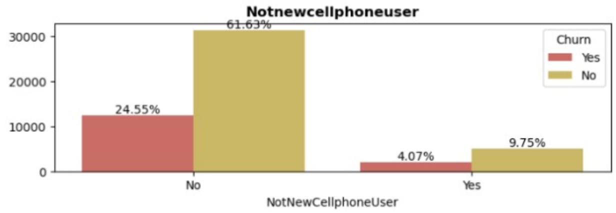
7) Martial status of 60 percent of the data is known out of which, 25 percent are not married. The rest are unknown.

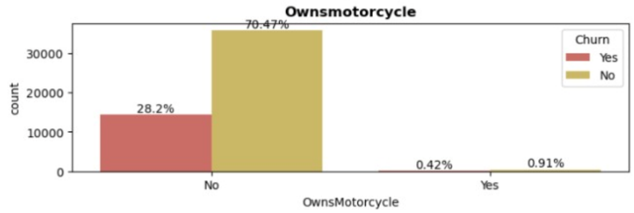
<u>Bivariate Analysis:</u>

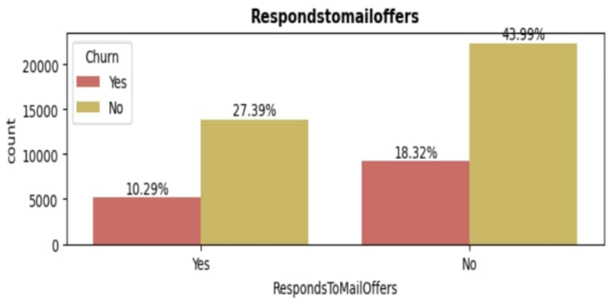


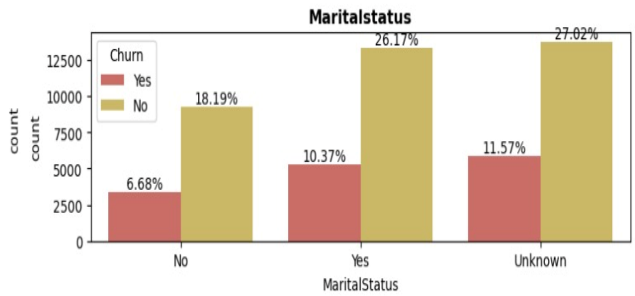


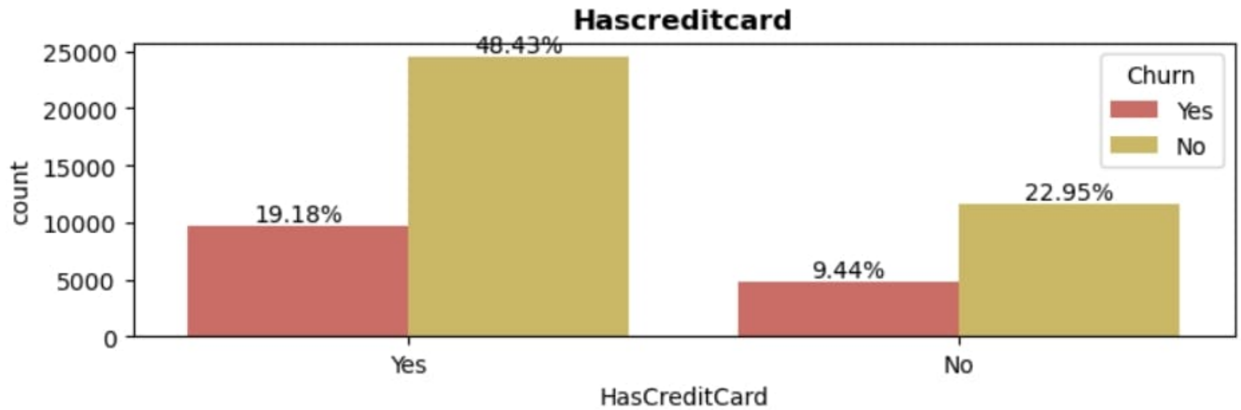












observation:

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

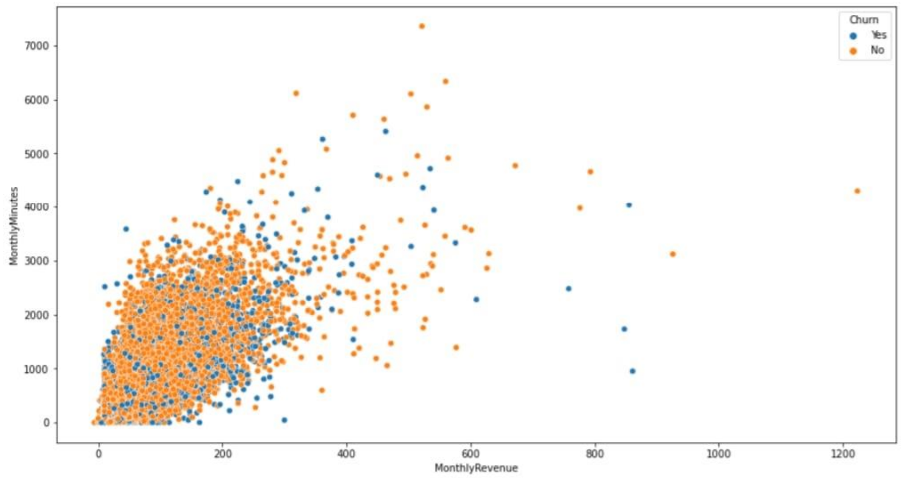
4. Marital Status of people churning is independent

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

Multivariate Analysis:

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

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

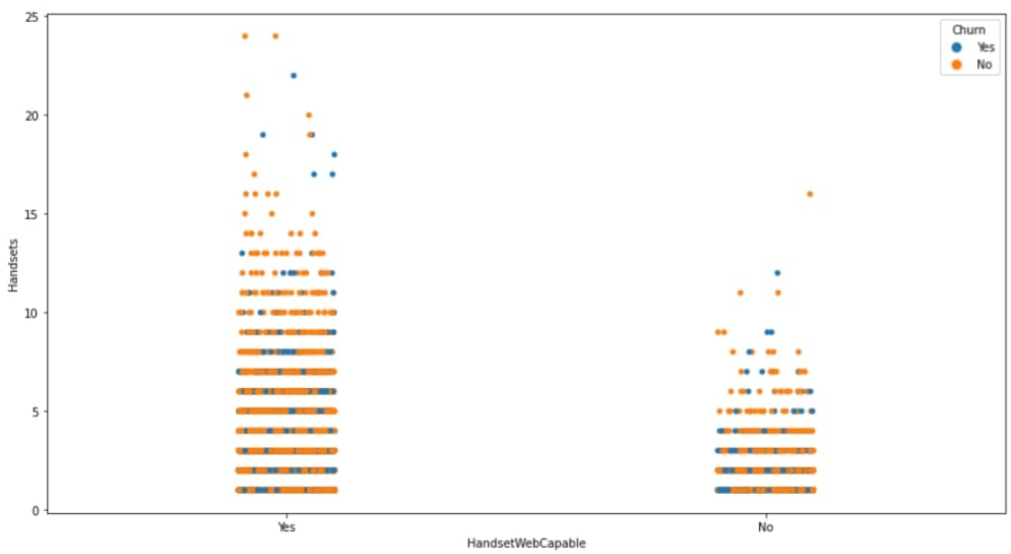


Observation:

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

```
1 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.

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).

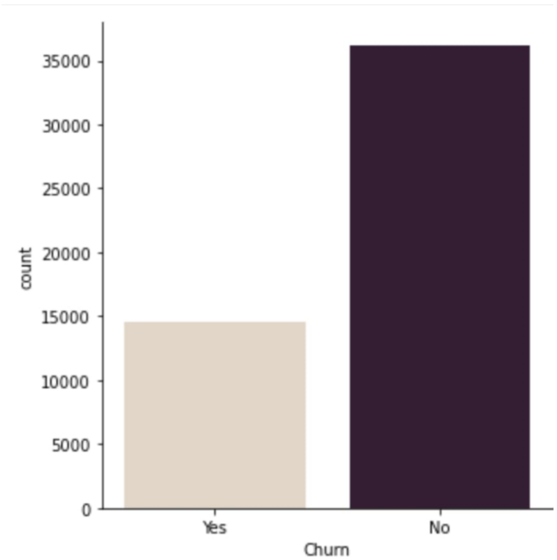
We have used Jarque-bera test to check the normality of data

$H0$: 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:



Here we can see that our target variable is too imbalanced, and to treat that we are going to use oversampling techniques like smote.

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	-0.8249
-0.912074	-0.995504	-0.621914	0.018886	0.088432	-1.200278	-1.216280	-1.013639	-0.8249
-0.912074	-0.995504	-0.621914	0.026624	0.088432	-1.515686	-1.216280	-1.760480	-0.8249
1.170112	-0.995504	-0.621914	0.655061	0.284291	2.228625	1.290204	1.349529	1.5019
-0.912074	-0.995504	-0.621914	0.034384	0.083251	-1.515686	-1.216280	-1.760480	-0.8249

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−($Log−Likelihood$/LL−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.

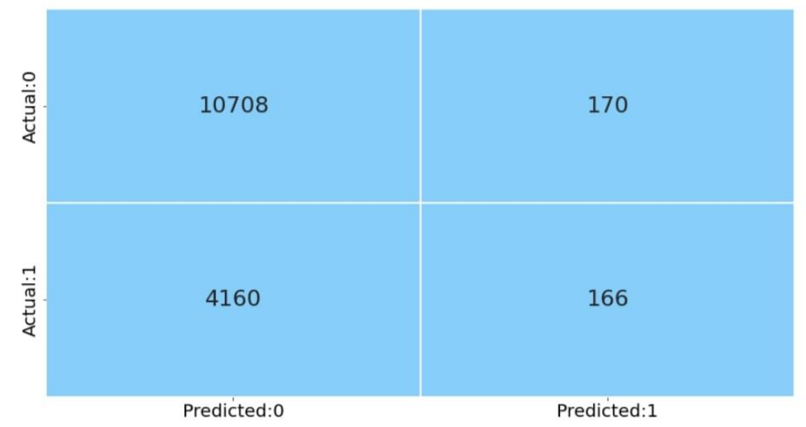
Cox & Snell R-squared: The convergence of the logistic model can be determined by the R-squared value. It is one of the types of Pseudo R-square.

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).

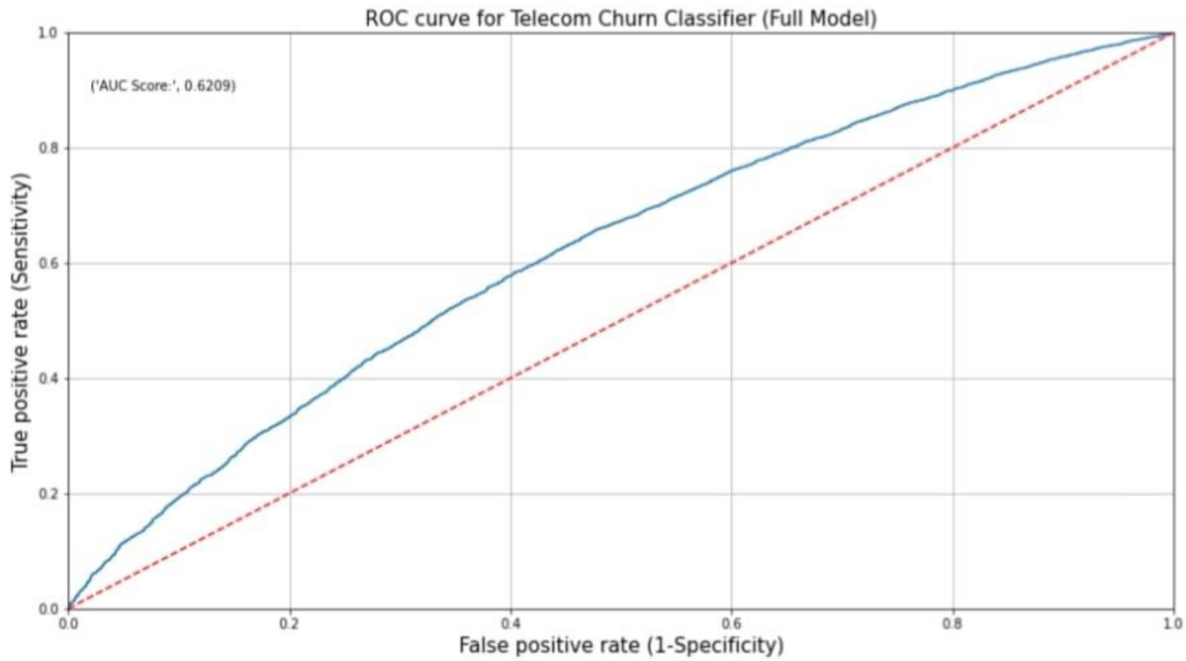
The AIC (Akaike Information Criterion) value:
It is a relative measure of model evaluation. It gives a trade-off between model accuracy and model complexity.

AIC: 41172.911

Confusion Matrix:



ROC Curve:



Inference:

- The red dotted line represents the ROC curve of a pure random classifier; a good classifier stays as far away from that line as possible (towards top-left corner).
- From the above plot, we can see that our classifier (logistic regression) is away from the dotted line; with the AUC score 0.6209.

Classification Report:

	precision	recall	f1-score	support
0	0.72	0.98	0.83	10878
1	0.49	0.04	0.07	4326
accuracy			0.72	15204
macro avg	0.61	0.51	0.45	15204
weighted avg	0.66	0.72	0.62	15204

Interpretation:

From the above output, we can infer that the recall of the positive class is known as sensitivity and the recall of the negative class is specificity.

support is the number of observations in the corresponding class.

The macro average in the output is obtained by averaging the unweighted mean per label and the weighted average is given by averaging the support-weighted mean per label.

Score Card for Logistic Regression:

	Probability Cutoff	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	0.100000	0.620859	0.285081	0.995608	0.288345	0.001535	0.443244
1	0.200000	0.620859	0.308895	0.899908	0.398645	0.062941	0.459921
2	0.300000	0.620859	0.371621	0.536986	0.609905	0.155104	0.439255
3	0.400000	0.620859	0.438914	0.179380	0.701263	0.107294	0.254677
4	0.500000	0.620859	0.494048	0.038373	0.715207	0.031492	0.071214
5	0.600000	0.620859	0.625000	0.008091	0.716390	0.008766	0.015974
6	0.700000	0.620859	0.500000	0.000693	0.715470	0.000597	0.001385
7	0.800000	0.620859	0.000000	0.000000	0.715470	0.000000	0.000000
8	0.900000	0.620859	0.000000	0.000000	0.715470	0.000000	0.000000

Interpretation: The above data frame shows that, the model cutoff probability 0.6, returns the highest AUC score, f1-score, kappa score and accuracy.

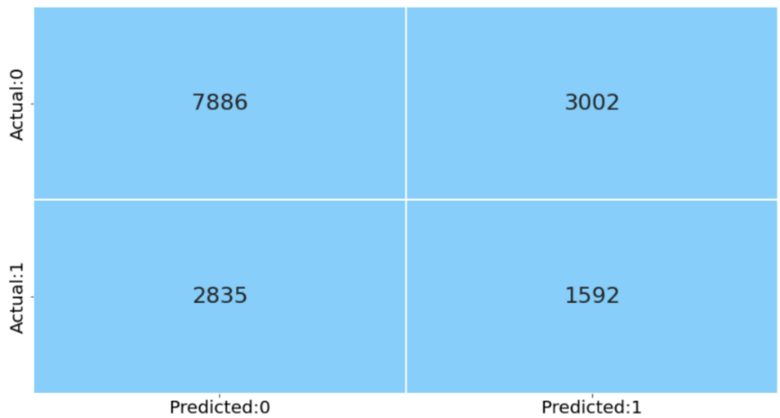
Decision Tree

Build a full decision tree model on a train dataset using 'gini'.

```
# instantiate the 'DecisionTreeClassifier' object using 'gini' criterion
# pass the 'random_state' to obtain the same samples for each time you run the code
decision_tree_classification = DecisionTreeClassifier(criterion = 'gini', random_state = 10)

# fit the model using fit() on train data
decision_tree = decision_tree_classification.fit(x_train, y_train)
```

Model Performance: -

1.Confusion Matrix



2.Report: -

Calculate performance measures on the train set.

```
# compute the performance measures on train data
# call the function 'get_train_report'
# pass the decision tree to the function
train_report = get_train_report(decision_tree)

# print the performance measures
print(train_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     25448
           1       1.00      1.00      1.00     10284

    accuracy                           1.00     35732
   macro avg       1.00      1.00      1.00     35732
weighted avg       1.00      1.00      1.00     35732
```

Calculate performance measures on the test set.

```
# compute the performance measures on test data
# call the function 'get_test_report'
# pass the decision tree to the function
test_report = get_test_report(decision_tree)

# print the performance measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.74      0.72      0.73     10888
           1       0.35      0.36      0.35      4427

    accuracy                           0.62     15315
   macro avg       0.54      0.54      0.54     15315
weighted avg       0.62      0.62      0.62     15315
```

Inference: -

- From The above model, our train accuracy is 1 and test accuracy is 0.62, result is Overfitting
- As the model is over fitted, our false Negative and false Positive is inaccurate
- In the next step we tuned the Hyperparameters and rebuild the model.

Tune the Hyperparameters using GridSearchCV (Decision Tree)

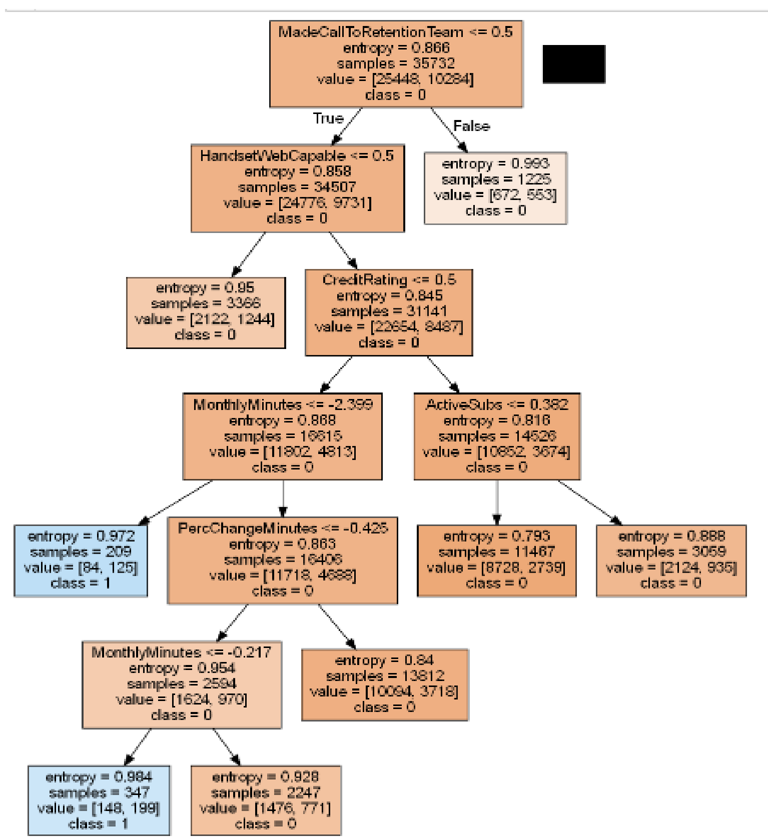
```
# get the best parameters
print('Best parameters for decision tree classifier: ', tree_grid_model.best_params_, '\n')
```

```
Best parameters for decision tree classifier:  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'log2', 'max_leaf_node
s': 8, 'min_samples_leaf': 1, 'min_samples_split': 2}

CPU times: total: 3min 3s
Wall time: 3min 3s
```

```
tree_grid_model.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 6,
 'max_features': 'log2',
 'max_leaf_nodes': 8,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

Hyper Tuned Decision Tree:



Model Performance after Tunning:

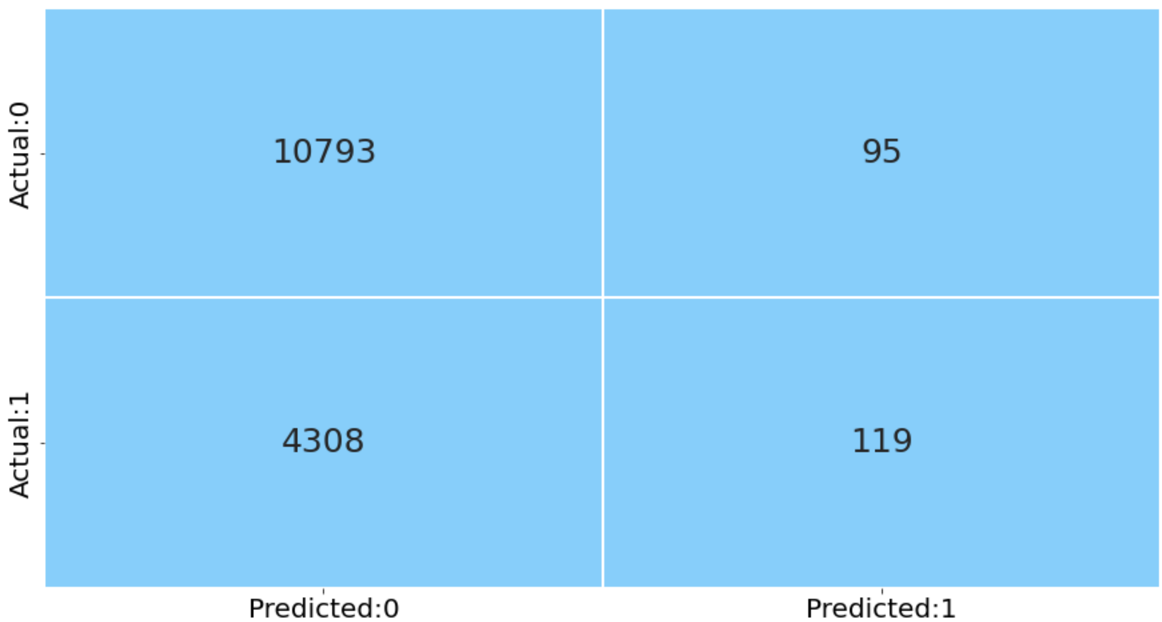
Classification Report for train set:

	precision	recall	f1-score	support
0	0.72	0.99	0.83	25448
1	0.58	0.03	0.06	10284
accuracy			0.71	35732
macro avg	0.65	0.51	0.45	35732
weighted avg	0.68	0.71	0.61	35732

Classification Report for test set:

	precision	recall	f1-score	support
0	0.71	0.99	0.83	10888
1	0.56	0.03	0.05	4427
accuracy			0.71	15315
macro avg	0.64	0.51	0.44	15315
weighted avg	0.67	0.71	0.61	15315

Confusion Matrix:



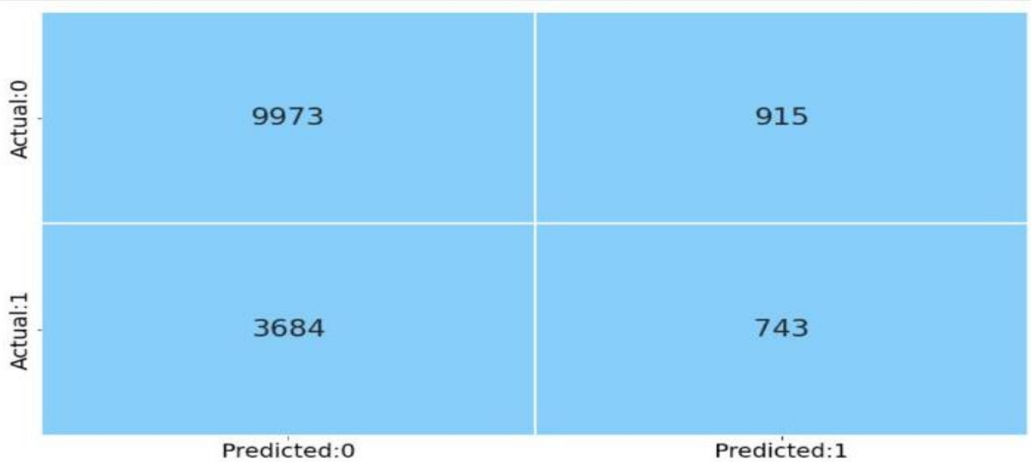
Inference: -

- The train and test Accuracy are comparable, which shows the reduction in overfitting.
- In this case the false negative and false positive values can be trusted and the FN value are quite High, but as our focus is on reduction of False negative values
- Typically, Random Forest classifier is more accurate than a single decision tree, we rebuild the model using the same to reduce the FN and increase the accuracy.

Random forest for classification

```
# instantiate the 'RandomForestClassifier'
# pass the required number of trees in the random forest to the parameter, 'n_estimators'
# pass the 'random_state' to obtain the same samples for each time you run the code
rf_classification = RandomForestClassifier(n_estimators = 15, random_state = 10)

# use fit() to fit the model on the train set
rf_model = rf_classification.fit(x_train, y_train)
```

Confusion matrix:



Report:

Calculate performance measures on the train set.

```
# compute the performance measures on train data
# call the function 'get_train_report'
# pass the random forest model to the function
train_report = get_train_report(rf_model)

# print the performace measures
print(train_report)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     25448
           1       1.00      0.98      0.99     10284

    accuracy                           0.99     35732
   macro avg       1.00      0.99      0.99     35732
weighted avg       0.99      0.99      0.99     35732
```

Calculate performance measures on the test set.

```
# compute the performance measures on test data
# call the function 'get_test_report'
# pass the random forest model to the function
test_report = get_test_report(rf_model)

# print the performace measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.73      0.92      0.81     10888
           1       0.45      0.17      0.24      4427

    accuracy                           0.70     15315
   macro avg       0.59      0.54      0.53     15315
weighted avg       0.65      0.70      0.65     15315
```

Inferences:

- From The above model, our train accuracy is 0.99 and test accuracy is 0.70, result is Overfitting
- As the model is over fitted, our false Negative and false Positive is inaccurate
- In the next step we tuned the Hyperparameters and rebuild the model.

Tuned the Hyperparameters using GridSearchCV (Random Forest)

```
# get the best parameters
print('Best parameters for random forest classifier: ', rf_grid_model.best_params_, '\n')
```

```
Best parameters for random forest classifier:  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'max_leaf_nodes': 11, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 50}

CPU times: total: 2h 58min 37s
Wall time: 2h 59min 54s
```

'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'max_leaf_nodes': 11, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 50

Model performance after tuning:

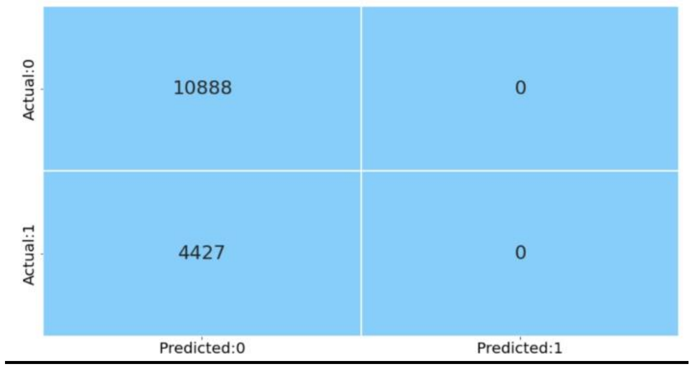
```
# print the performance measures for test set for the model with best parameters
print('Classification Report for test set:\n', get_test_report(rf_model))
```

```
Classification Report for test set:
               precision    recall  f1-score   support

           0       0.71      1.00      0.83     10888
           1       0.00      0.00      0.00      4427

    accuracy                           0.71     15315
   macro avg       0.36      0.50      0.42     15315
weighted avg       0.51      0.71      0.59     15315
```

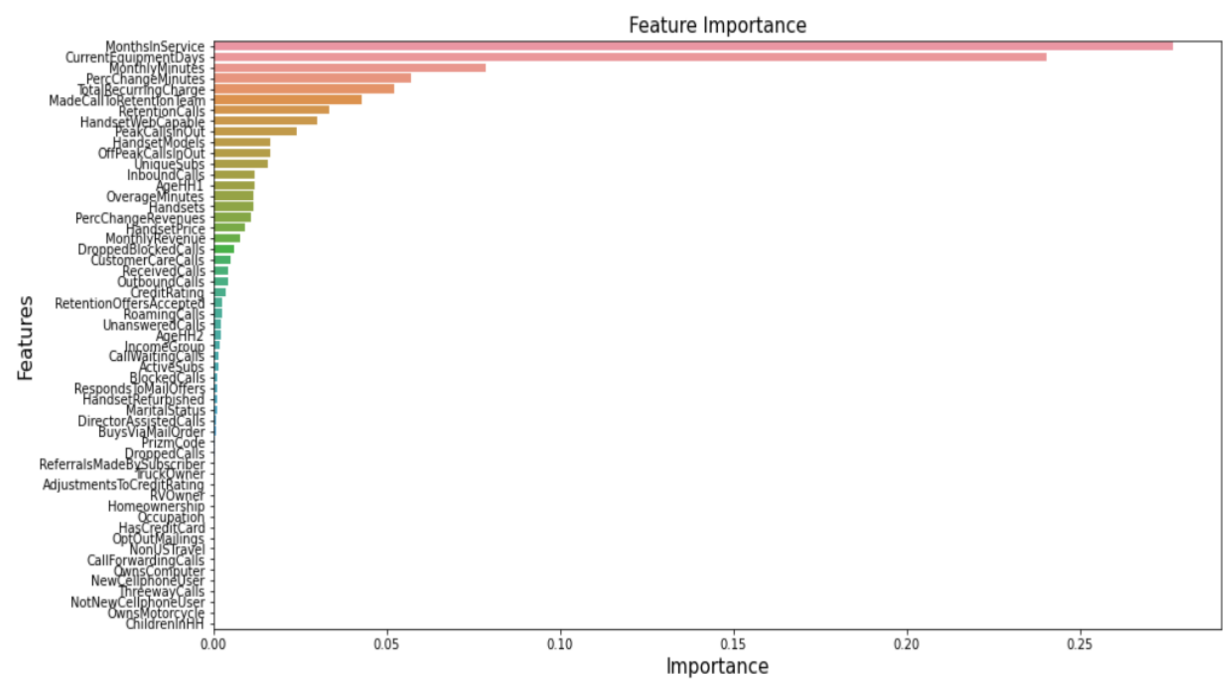
```
# print the performance measures for train set for the model with best parameters
print('Classification Report for train set:\n', get_train_report(rf_model))
```

```
Classification Report for train set:
               precision    recall  f1-score   support

           0       0.71      1.00      0.83     25448
           1       1.00      0.00      0.00     10284

    accuracy                           0.71     35732
   macro avg       0.86      0.50      0.42     35732
weighted avg       0.80      0.71      0.59     35732
```

Confusion matrix:



Feature importance:



The method feature-importance returns the value corresponding to each feature which is defined as the ratio of total decrease in Gini impurity across every tree in the forest where the feature is used to the total count of trees in the forest. This is also called as, Gini importance.

Tune the Hyperparameters using GridSearchCV (Random Forest) -2

```
# get the best parameters
print('Best parameters for random forest classifier: ', rf_grid_model.best_params_, '\n')
```

```
Best parameters for random forest classifier:  {'criterion': 'gini', 'max_depth': 9, 'max_features': 'sqrt', 'max_leaf_nodes': 13, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 54}

CPU times: total: 9h 38min 53s
Wall time: 9h 47min 31s
```


```
Best parameters for random forest classifier:  {'criterion': 'gini', 'max_depth': 9, 'max_features': 'sqrt',
'max_leaf_nodes': 13, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 54}

CPU times: total: 9h 38min 53s
Wall time: 9h 47min 31s
```


Model performance after tuning:

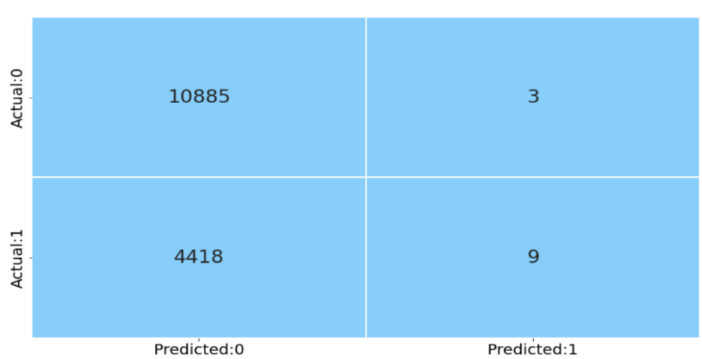
```
# print the performance measures for test set for the model with best parameters
print('Classification Report for test set:\n', get_test_report(rf_model))
```

```
Classification Report for test set:
               precision    recall  f1-score   support

           0       0.71      1.00      0.83     10888
           1       0.75      0.00      0.00      4427

    accuracy                           0.71     15315
   macro avg       0.73      0.50      0.42     15315
weighted avg       0.72      0.71      0.59     15315
```

```
# print the performance measures for train set for the model with best parameters
print('Classification Report for train set:\n', get_train_report(rf_model))
```

```
Classification Report for train set:
               precision    recall  f1-score   support

           0       0.71      1.00      0.83     25448
           1       1.00      0.00      0.00     10284

    accuracy                           0.71     35732
   macro avg       0.86      0.50      0.42     35732
weighted avg       0.80      0.71      0.59     35732
```

Confusion matrix:



Inference:

The train and test Accuracy are comparable, which shows the reduction in overfitting.

- In this case the false negative and false positive values can be trusted and the FN value are quite High, but as our focus is on reduction of False negative values
- Typically, Random Forest classifier is more accurate than a single decision tree, we rebuild the model using the same to reduce the FN and increase the accuracy.

Boosting

Boosting Methods:

The Ensemble technique considers multiple models for predicting the results. Bagging and Boosting are two of the types of ensembles. The bagging methods construct the multiple models in parallel; whereas, the boosting methods construct the models sequentially.

Earlier, we have studied one of the bagging (bootstrap aggregating) technique i.e., Random Forest.
The boosting method fits multiple weak classifiers to create a strong classifier. In this method, the model tries to correct the errors in the previous model. In this section, we learn some of the boosting methods such as AdaBoost, Gradient Boosting and Boost.

1 Gradient Boosting:

This method optimizes the differentiable loss function by building the number of weak learners (decision trees) sequentially. It considers the residuals from the previous model and fits the next model to the residuals. The algorithm uses a gradient descent method to minimize the error.

```
print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.74      0.93      0.82     10888
           1       0.51      0.18      0.27      4427

    accuracy                           0.71     15315
   macro avg       0.62      0.56      0.55     15315
weighted avg       0.67      0.71      0.66     15315
```

```
y_pred=gboost_model.predict(X_test)
gboost_model.score(X_train,y_train)
```

```
0.9811373558714878
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[10111,   777],
       [ 3616,   811]], dtype=int64)
```

Inference:

- According to the train-test report, train value shows 98% and test value shows 71% from this we can say the model is overfitted.

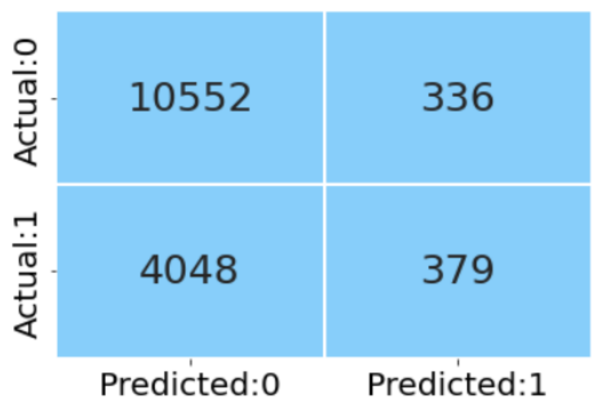
2 AdaBoost:

Let us build the AdaBoost classifier with decision trees. The model creates several stumps (decision tree with only a single decision node and two leaf nodes) on the train set and predicts the class based on these weak learners (stumps). For the first model, it assigns equal weights to each sample. It assigns the higher weight for the wrongly predicted samples and lower weight for the correctly predicted samples. This method continues till all the observations are correctly classified or the predefined number of stumps is created.

```
# instantiate the 'AdaBoostClassifier'
# n_estimators: number of estimators at which boosting is terminated
# pass the 'random_state' to obtain the same results for each code implementation
ada_model = AdaBoostClassifier(n_estimators = 40, random_state = 10)

# fit the model using fit() on train data
ada_model.fit(X_train, y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=40, random_state=10)
```

<u>Confusion matrix:</u>



<u>Model Performance:</u>

```
print(test_report)

              precision    recall  f1-score   support

           0       0.72      0.97      0.83     10888
           1       0.53      0.09      0.15      4427

    accuracy                           0.71     15315
   macro avg       0.63      0.53      0.49     15315
weighted avg       0.67      0.71      0.63     15315
```

```
In [44]: y_pred=ada_model.predict(X_test)
         ada_model.score(X_train,y_train)

Out[44]: 0.7178999216388671
```

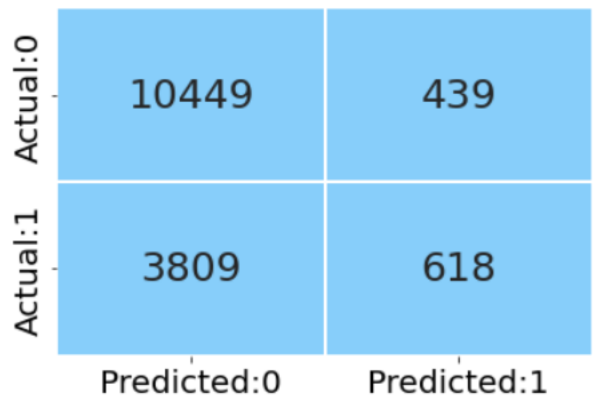
Inference:

- According to Adaboost report, the train-test report shows comparable values with each other.
- The model fit is good, but still need to improve the model by reducing the FN.

3. XGBoost:

XGBoost (extreme gradient boost) is an alternative form of gradient boosting method. This method generally considers the initial prediction as 0.5 and build the decision tree to predict the residuals. It considers the regularization parameter to avoid overfitting.

Model Performance:




```
xgb_model.score(X_train,y_train)
```

```
0.7478730549647374
```

```
xgb_model.score(X_test,y_test)
```

```
0.7226248775710088
```

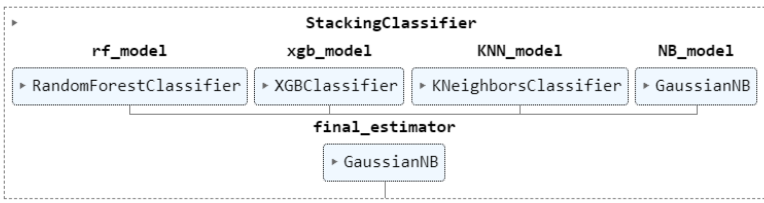
Inference:

- According to XGBoost model report, we can observe that train-test values are good.
- Compared to previous models, the errors have been reduced.
- Compare to all the boosting methods this is best.

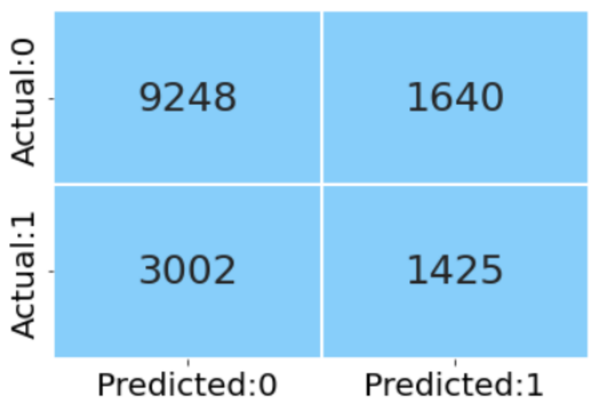
Stack Generalization:

```
%%time
# consider the various algorithms as base learners
base_learners = [('rf_model', RandomForestClassifier(criterion = 'entropy', max_depth = 10, max_features = 'sqrt',
                                                      max_leaf_nodes = 8, min_samples_leaf = 5, min_samples_split = 2,
                                                      n_estimators = 50, random_state = 10)),
                 ('xgb_model',XGBClassifier(colsample_bytree= 1, gamma= 1, learning_rate= 0.2,
                                    max_depth=4, min_child_weight= 4, subsample= 1, tree_method= 'hist' )),
                 ('KNN_model', KNeighborsClassifier(n_neighbors = 17, metric = 'euclidean')),
                 ('NB_model', GaussianNB())]

# initialize stacking classifier
# pass the base learners to the parameter, 'estimators'
# pass the Naive Bayes model as the 'final_estimator'/ meta model
stack_model = StackingClassifier(estimators = base_learners, final_estimator = GaussianNB())

# fit the model on train dataset
stack_model.fit(X_train, y_train)
```

```
CPU times: total: 1min 3s
Wall time: 14.5 s
```



Confusion Matrix:



Model Performance:

	precision	recall	f1-score	support
0	0.75	0.85	0.80	10888
1	0.46	0.32	0.38	4427
accuracy			0.70	15315
macro avg	0.61	0.59	0.59	15315
weighted avg	0.67	0.70	0.68	15315

```
# train report
stack_model.score(X_train,y_train)
```

```
0.7287585357662599
```

greatlearning
Learning for Life

Inference:

- According to stack Generalization report, we can observe that train test values are good.
- The model is good fit, type II errors are reduced by 807
- **<u>Compared to all the models built this model is the best model.</u>**

Limitations:

- The data which we have is highly imbalanced this might lead to inaccurate predictions.
- To enhance the data quality and to reduce errors we have transformed the data using power transformer, getting Business insights out of this would be difficult.
- To proceed with Feature Engineering, we need to have domain knowledge

Conclusion:

- At first, we dealt with the null value imputation and then we proceeded with Exploratory data analysis to analyse the univariant and bivariant features to understand why the customers are churning.
- As the data was not normal, we use non parametrical statistical test **Kruskal Wallis test**
- This test is used to check features are dependent or independent to Target variables.
- We have built various classification algorithms and final outcomes are as follows
- Compare to base logistic model, the overfitting is reduced and FN errors are reduced by nearly 32%
- Comparatively the recall value has been boosted from 4% to 32%
- Compare to base Decision model, the overfitting is reduced and FN errors are reduced by nearly 30%

ALGORITHMS	Remark	Train Set							Test Set						
		Recall		Precision		F1 Score		Accuracy	Recall		Precision		F1 Score		Accuracy
		0	1	0	1	0	1		0	1	0	1	0	1	
Logistic Regresion	Threashold as 0.5								**0.98**	**0.04**	**0.72**	**0.49**	**0.83**	**0.07**	**0.71**
Logistic Regresion	Using SMOTE with Threashold =0.6								0.86	0.25	0.74	0.42	0.79	0.31	0.68
Decision Tree	Overfit	1	1	1	1	1	1	1	0.72	0.36	0.74	0.35	0.73	0.35	0.71
Decision Tree	After Hyper tunning	0.99	0.03	0.72	0.58	0.83	0.06	0.71	0.99	0.03	0.71	0.56	0.83	0.05	0.71
Random Forest	Overfit (n-estimator=15)	1	0.98	0.99	1	1	0.99	0.99	0.92	0.17	0.7	0.45	0.81	0.24	0.7
Random Forest	After Hyper tunning	1	0	0.71	0	0.83	0	0.71	1	0	0.71	0.75	0.83	0	0.71
KNN	Only with numerical variables							0.72	0.87	0.22	0.73	0.39	0.79	0.28	0.68
XG Boost	max_depth = 10, gamma = 1							0.99	0.89	0.24	0.74	0.48	0.81	0.32	0.7
Stack Model	RandomForest,XGBoost,KNN_model, Naïve Bayes							0.72	0.85	0.32	0.75	0.46	0.8	0.38	0.7

Reference:

1.Customer Churn Analysis
Brief Overview of Customer Churn Analysis and Prediction with Decision Tree Classifier.
Retrieved from https://towardsdatascience.com/customer-churn-analysis-4f77cc70b3bd

2. Customer churn analysis: Churn determinants and mediation effects of partial defection in the Korean mobile telecommunications service industry (2006).
Retrieved from http://people.stern.nyu.edu/shan2/customerchurn.pdf

3. Kiran Dahiya, Surbhi Bhatia. Customer Churn Analysis in Telecom Industry.
Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7359318

4.Teemu Mutanen. Customer churn analysis – a case study.
Retrieved from
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.7169&rep=rep1&type=pdf

5.Churn Analysis: 3-Step Guide to Analysing Customer Churn Dominique Jackson (March 31, 2020).
Retrieved from https://baremetrics.com/blog/churn-analysis

6.Customer Churn Analysis: A Comprehensive Guide Amit Phaujdar on Churn Analysis, Marketing Analytics (March 15th, 2021).
Retrieved from https://hevodata.com/learn/understanding-customer-churn-analysis/

7.Understanding Random Forest How the Algorithm Works and Why it Is So Effective.
Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

8.Logistic Regression. Retrieved from https://www.sciencedirect.com/topics/computer-science/logistic-regression

9.Decision Tree Algorithm, explained. Retrieved from
https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

10.Prashant Gupta : Decision Trees in Machine Learning(May 18, 2017).
Retrieved from https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

11.Logistic regression.
Retrieved from https://www.ibm.com/topics/logistic-regression

12.Gradient Boosting from scratch.
Retrieved from https://blog.mlreview.com/gradient-boosting-from-scratch-1e317ae4587d

****