Batch details	PGPDSE-FT Offline BLR AUG-22
Team members	Bharath N Raju Niranjan Vishal Choudhary Nikhil Kondapalli Reshma RA
Domain of Project	Retail
Proposed project title	Telecom Churn
Group Number	4
Team Leader	Vishal Choudhary
Mentor Name	Mr. Jatinder Bedi

Table of Contents:

S. No	Topic	Page No
1	Project Details	4
2	Dataset Information	6
3	Data Exploration (EDA)	10
4	All Base Models	27
5	Decision Tree	30
6	Random Forest	33

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: 57

No. of records: 51047

Target Column: churn

Redundant columns: Customer Id, NotNewCellphoneUser , ServiceArea.

DATA EXPLORATION (EDA)

Summary of Dataset:

```
1 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
```

Data Cleaning:

Duplicate value 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.
```

Treating few columns which are having inappropriate data:

1) Handsetprice variable

```
df1['HandsetPrice'].unique()
```

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

```
df1[df1['HandsetPrice']=='Unknown'].shape[0]
```

```
28981
```

```
# replacing unknown with zeros
df1['HandsetPrice']=df1['HandsetPrice'].replace(to_replace='Unknown',value= 0)
```

```
df1['HandsetPrice']=df1['HandsetPrice'].replace(to_replace=0,value= np.nan)
```

```
# calculating null value percentage for Handsetprice variable
df1['HandsetPrice'].isnull().sum()/df1.shape[0]*100
```

```
56.7742820201387
```

```
# as we can see that we have more than 56% of null values ,
# hence we are dropping the column,instead of filling 56% values which are not true.
```

```
df1.drop(columns='HandsetPrice',inplace=True)
```

```
df1.shape
```

```
(51046, 54)
```

2)

Treating credit rating variable

```
]: df1['CreditRating'].unique()
```

```
]: array(['1-Highest', '4-Medium', '3-Good', '6-VeryLow', '2-High', '5-Low',
          '7-Lowest'], dtype=object)
```

```
]: # binning the variable with high,medium and low
```

```
]: dict_rating={ '1-Highest':'High',
            '2-High':'High',
            '3-Good':'Medium',
            '4-Medium':'Medium',
            '5-Low':'Low',
            '6-VeryLow':'Low',
            '7-Lowest':'Low'
             }
```

```
]: df1['CreditRating']=df1['CreditRating'].map(dict_rating)
```

```
]: df1['CreditRating'].unique()
```

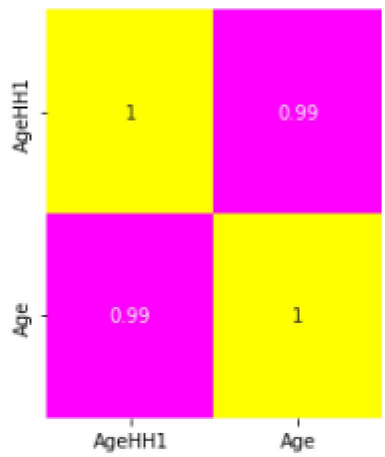
```
]: array(['High', 'Medium', 'Low'], dtype=object)
```

```
]: # now we have only 3 categories high , medium and low
```

Treating AgeHH1 and AgeHH2 variables:

```
df1[df1['AgeHH1']==0].shape
(13916, 54)
df1[df1['AgeHH2']==0].shape
(26086, 54)
```

AgeHH1	AgeHH2	Age
62.000000	0.000000	62.000000
40.000000	42.000000	41.000000
26.000000	26.000000	26.000000
30.000000	0.000000	30.000000
46.000000	54.000000	50.000000



- We are dropping AgeHH1 and AgeHH2 since they are redundant, having the new column Age.

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.

```
# calculating percentage of null values.
n/df1.shape[0]*100
```

```
MonthlyRevenue             0.305607
MonthlyMinutes             0.305607
TotalRecurringCharge       0.305607
DirectorAssistedCalls      0.305607
OverageMinutes             0.305607
RoamingCalls               0.305607
PercChangeMinutes          0.718959
PercChangeRevenues         0.718959
Age                       29.042432
dtype: float64
```

Dropping the rows of missing values in the column which is having less than 1% missing values in it.

dropping the rows which contain null values less than 1%.

```
drop_cols=['MonthlyRevenue', 'MonthlyMinutes', 'TotalRecurringCharge',
       'DirectorAssistedCalls', 'OverageMinutes', 'RoamingCalls','PercChangeMinutes','PercChangeRevenues']
```

```
df1.dropna(subset=drop_cols,inplace=True)
```

```
df1.isnull().sum()[df1.isnull().sum()!=0]
```

```
Age    14712
dtype: int64
```

```
df1=df1.reset_index(drop=True,)
```

```
df1.shape
```

```
(50679, 53)
```

Imputing Null values in age column using KNN-Imputers;

Imputing null values in age column using KNN Imputers

```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
num_scaled=ss.fit_transform(num_cols)
```

```
imputer=KNNImputer(n_neighbors=1000)
df_filled=imputer.fit_transform(num_scaled)
```

```
df_filled=pd.DataFrame(df_filled,columns=num_cols.columns)
df_filled
```

Imputing null values in MaritalStatus using KNN Classifier

```
df1['MaritalStatus'].value_counts()
```

```
Unknown     19556
Yes         18520
No          12603
Name: MaritalStatus, dtype: int64
```

```
dict_1={'Unknown':np.nan,'Yes':1,'No':0 }
```

```
mar=df1['MaritalStatus']
```

```
mar=mar.map(dict_1)
```

```
mar.isnull().sum()
```

```
19556
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)
```

```
y_pred.shape
```

```
(19556,)
```

```
index_list=df_mar[df_mar['MaritalStatus'].isnull()==True].index
```

```
df_mar['MaritalStatus'][index_list]=y_pred
```

```
df_mar.shape
```

```
(50679, 34)
```

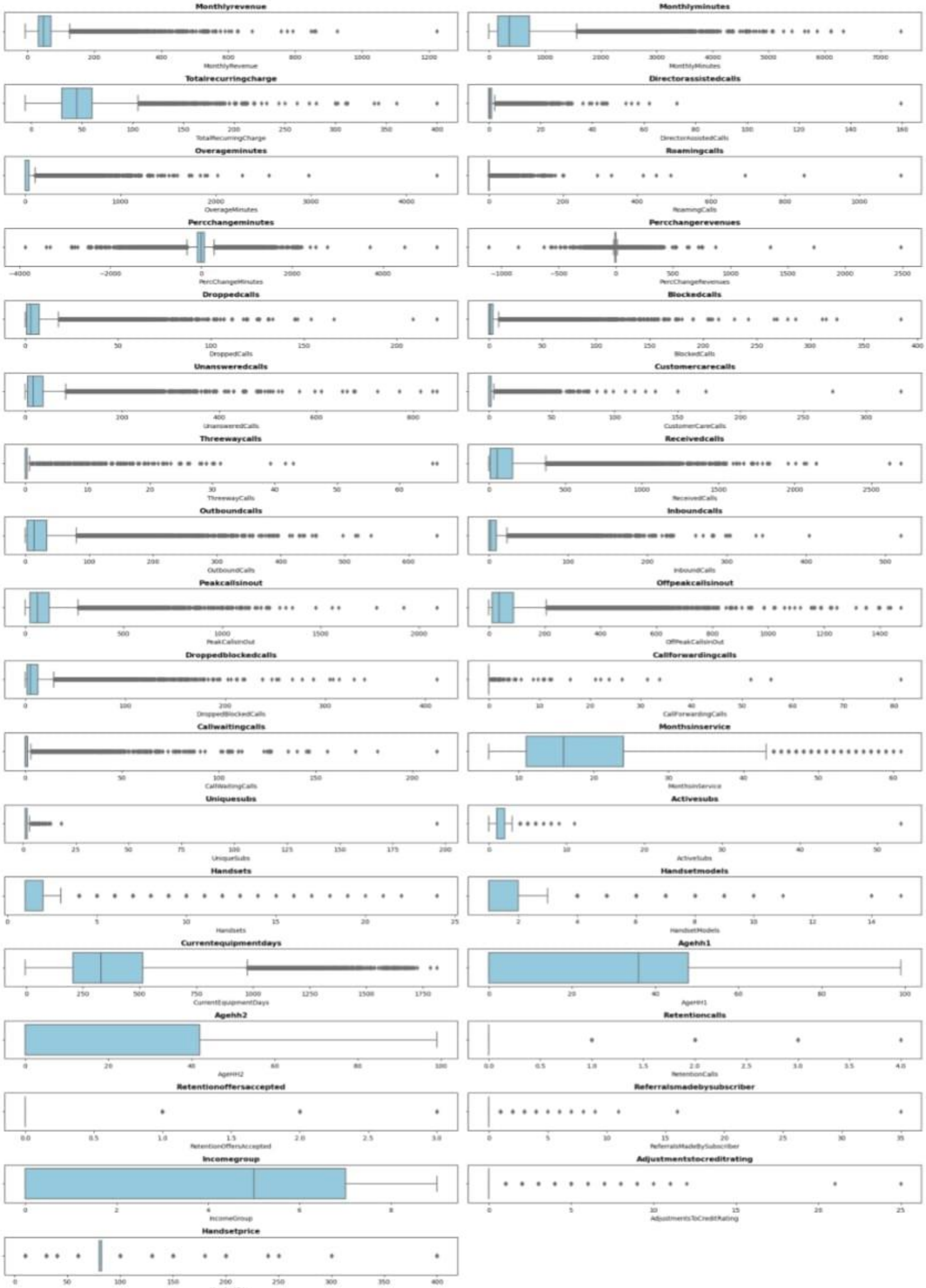
```
dict_2={1:'Yes',0:'No' }
```

```
df_mar['MaritalStatus']=df_mar['MaritalStatus'].map(dict_2)
```

```
df_mar['MaritalStatus'].value_counts()
```

```
No      25388
Yes     25291
```

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

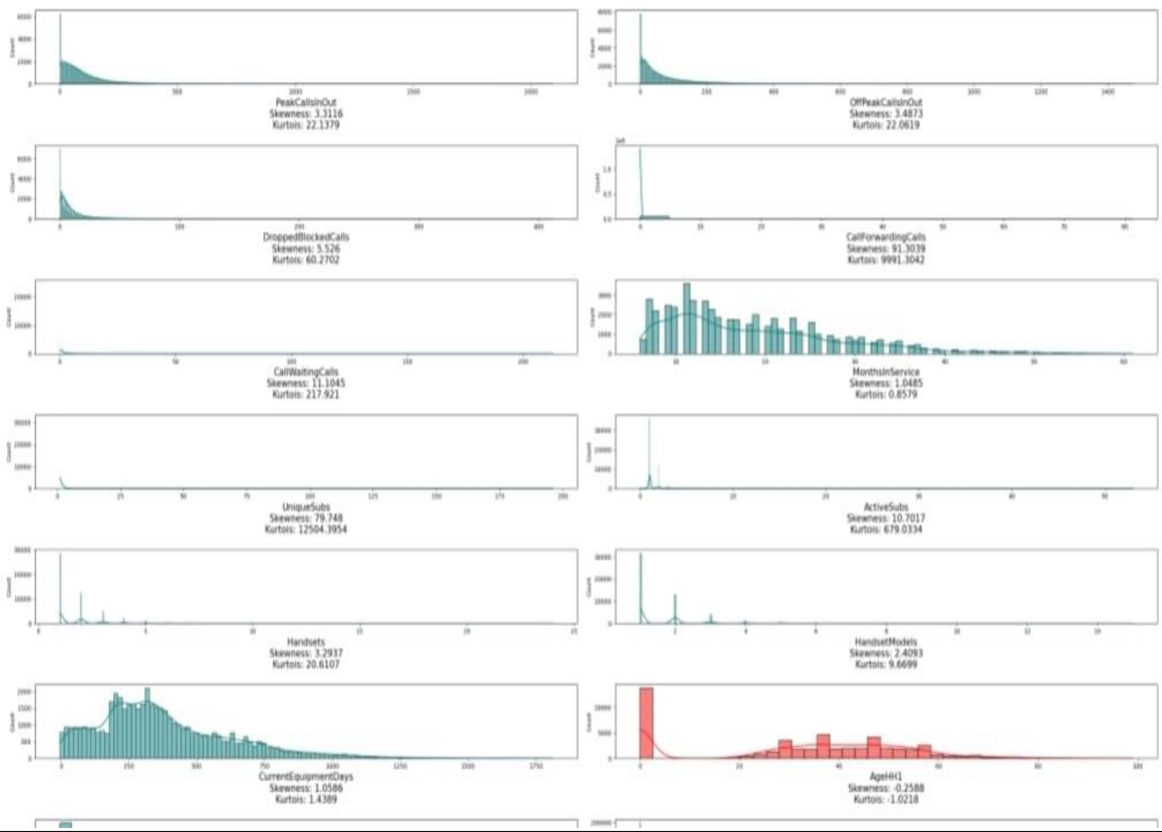
Reason for outliers exist in the data:

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:</u>



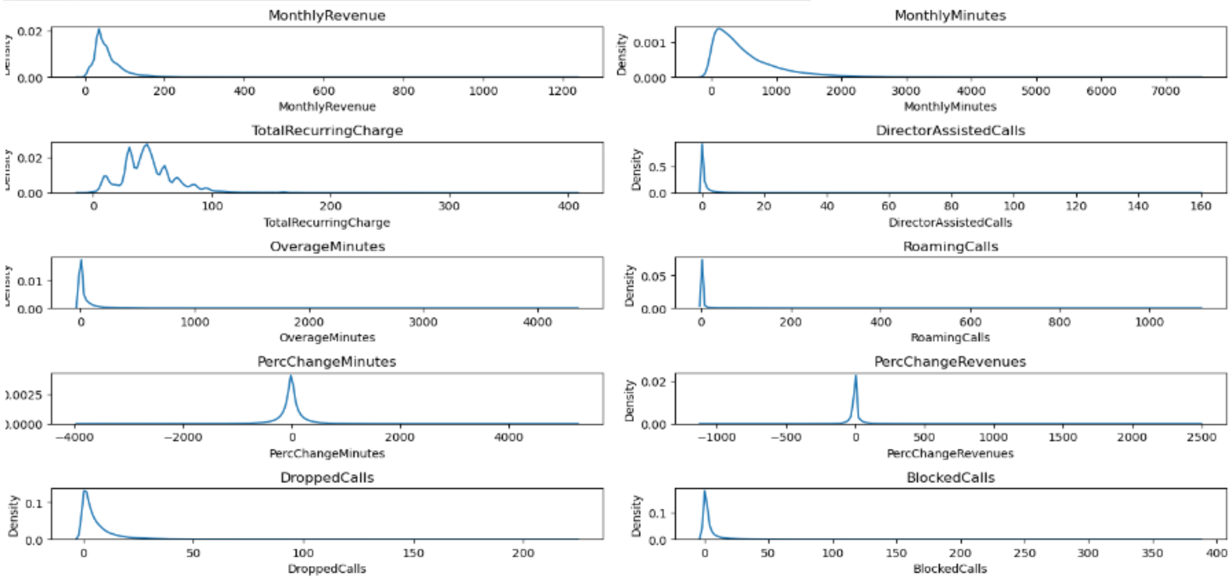
Inference: Here by visualizing dist plot we can see that the Features plotted in Teal colour are positively skewed and Features plotted in red colour are Negatively Skewed.

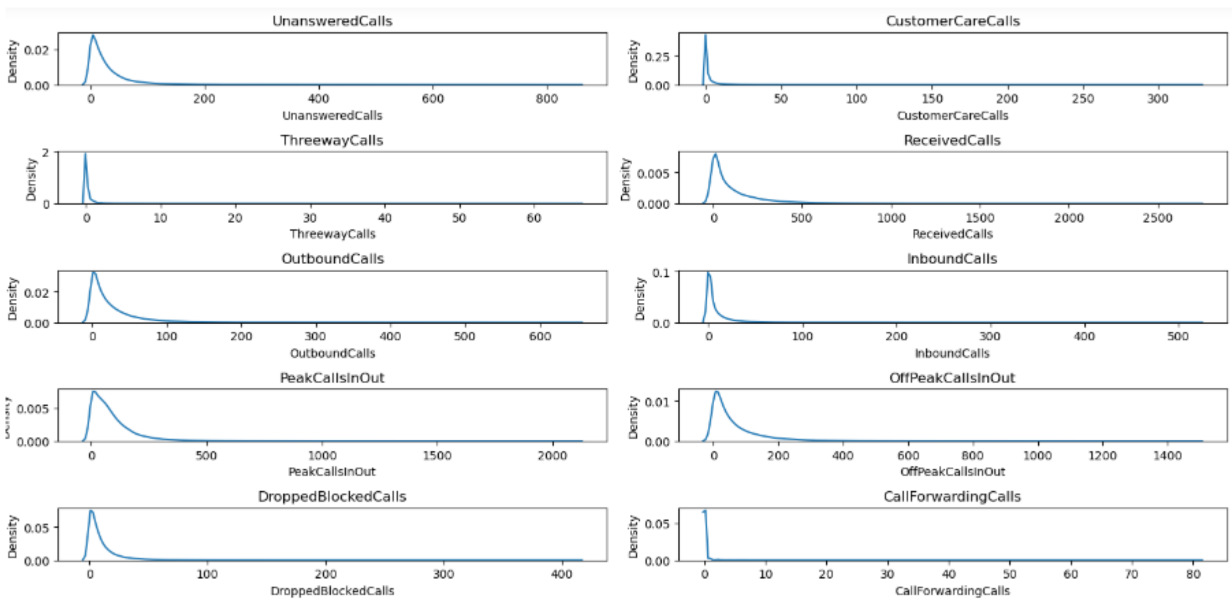
Descriptive Analysis (EDA)

Univariate Analysis:

Numerical Columns Visualization:

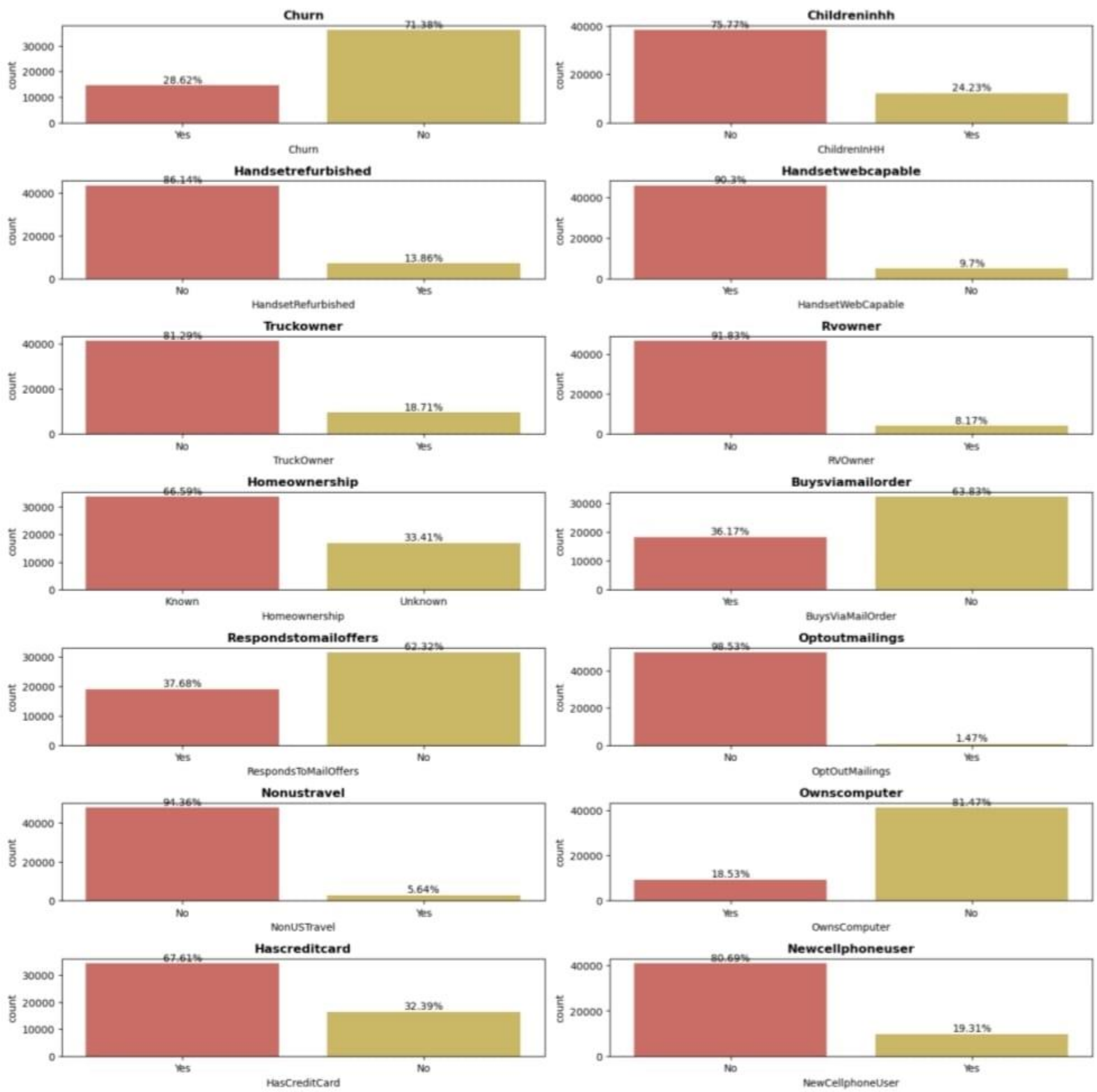
```python
# Kde Plot for Numerical features

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)
```

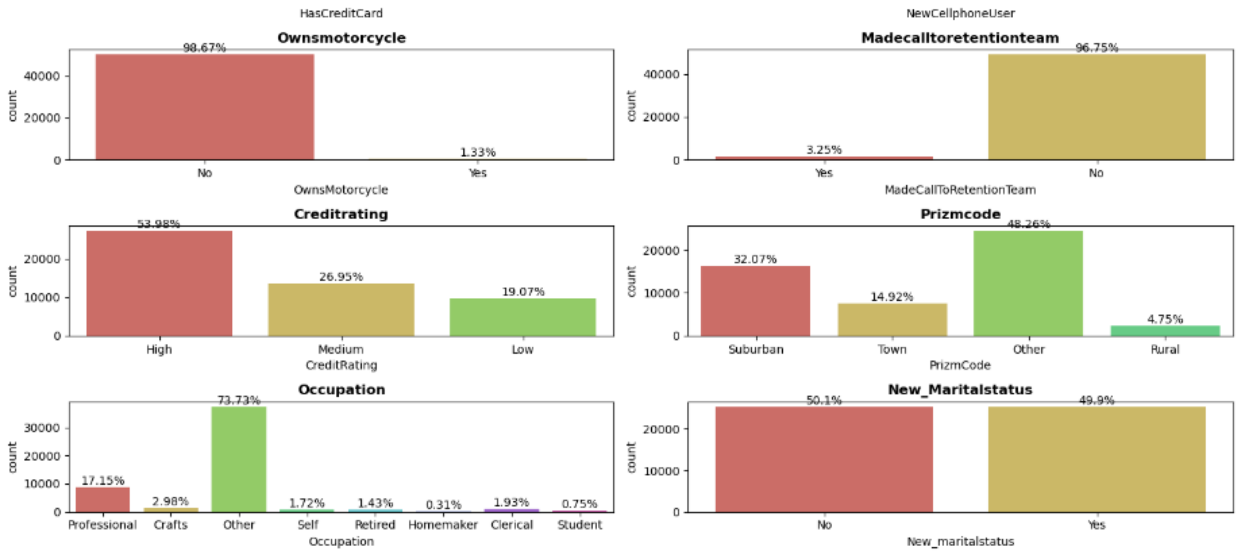




Categorical Columns Visualization:

```
#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.

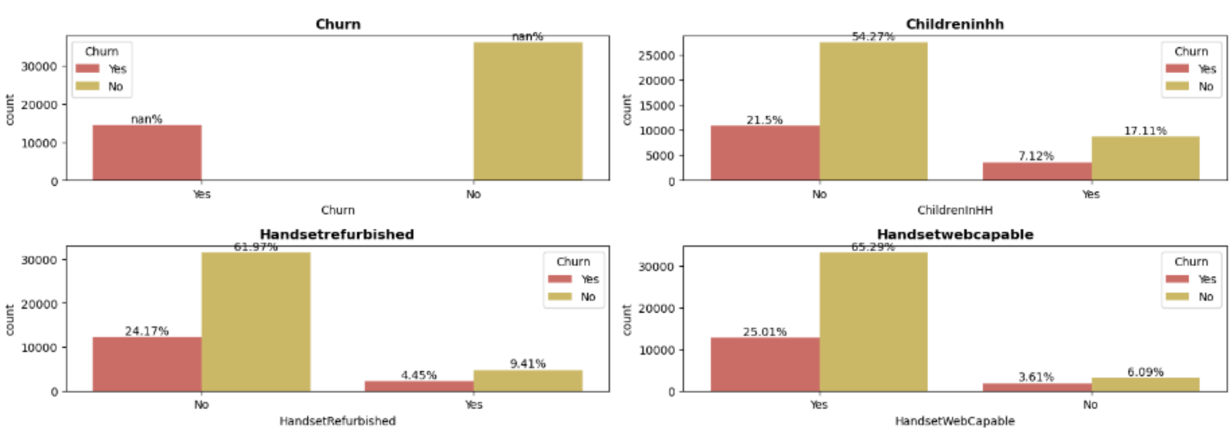
2) Handsetwebcapable More than 90 percent of the people in the data have internet support on their phone.

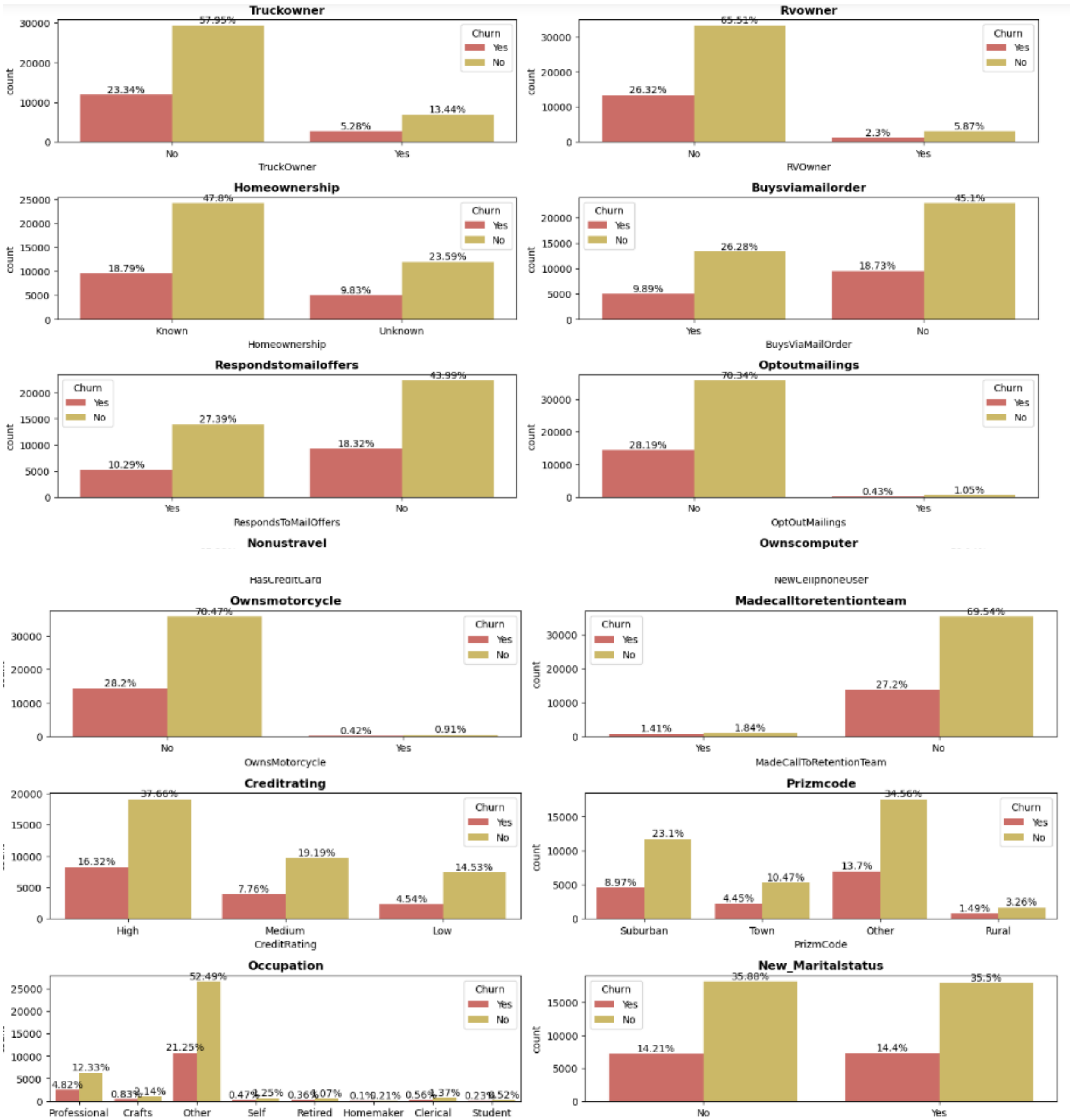
3) More than 65 percent of them don't have a credit card

4) Less than 2 percent of them own a motorcycle

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

<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

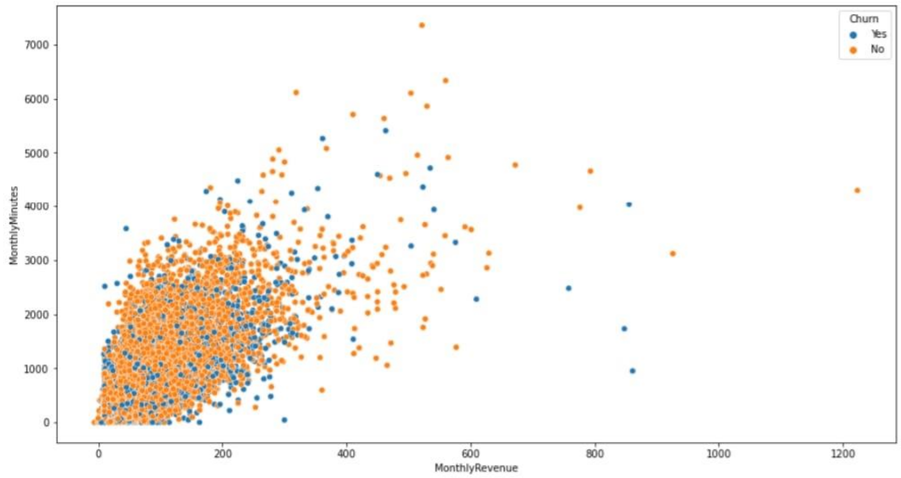
4. Marital Status of people churning is independent

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.

Statistics (Stats)

Feature	Statistical Test	P-Value	Inference
MonthlyRevenue	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
MonthlyMinutes	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
TotalRecurringCharge	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
DirectorAssistedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
OverageMinutes	kruskal wallis test	0.000009	Dependent numerical variable found after H-tes...
RoamingCalls	kruskal wallis test	0.922785	Independent numerical variable found after H-t...
PercChangeMinutes	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
PercChangeRevenues	kruskal wallis test	0.308102	Independent numerical variable found after H-t...
DroppedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
BlockedCalls	kruskal wallis test	0.000650	Dependent numerical variable found after H-tes...
UnansweredCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
CustomerCareCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
ThreewayCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
ReceivedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
OutboundCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
InboundCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
PeakCallsInOut	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
OffPeakCallsInOut	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
DroppedBlockedCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
CallForwardingCalls	kruskal wallis test	0.311887	Independent numerical variable found after H-t...
CallWaitingCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
RetentionCalls	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
RetentionOffersAccepted	kruskal wallis test	0.000000	Dependent numerical variable found after H-tes...
ReferralsMadeBySubscriber	kruskal wallis test	0.024863	Dependent numerical variable found after H-tes...
IncomeGroup	kruskal wallis test	0.026027	Dependent numerical variable found after H-tes...
AdjustmentsToCreditRating	kruskal wallis test	0.000646	Dependent numerical variable found after H-tes...
HandsetPrice	kruskal wallis test	0.242433	Independent numerical variable found after H-t...
ChildrenInHH	Chi-Square Test for Independence	0.030195	Dependent categorical variable found after Chi...
HandsetRefurbished	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
HandsetWebCapable	Chi-Square Test for Independence	0.000000	Dependent categorical variable found after Chi...
TruckOwner	Chi-Square Test for Independence	0.324832	Independent categorical variable found after C...
RVOwner	Chi-Square Test for Independence	0.500851	Independent categorical variable found after C...
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

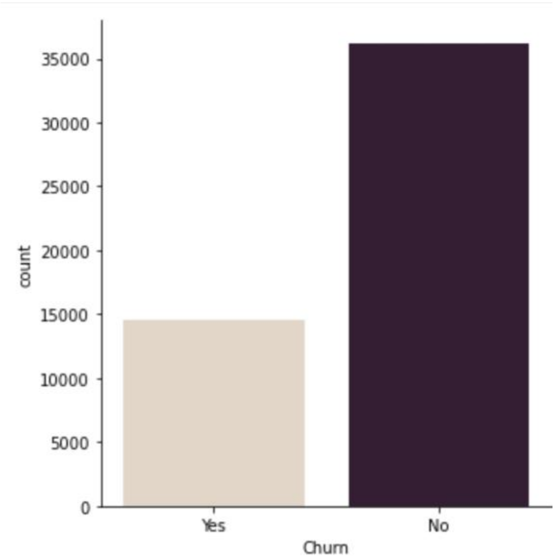
$H0$: The data is normally distributed.
$H1$: The data is not normally distributed.

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

Insignificant variables:- 13

```
insignificant columns=
      ['RoamingCalls','PercChangeRevenues','CallForwardingCalls','TruckOwner','RVOwner',
       'OptOutMailings','NonUSTravel','OwnsComputer','HasCreditCard','NewCellphoneUser',
       'OwnsMotorcycle','Occupation','New_maritalstatus']
```

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	263.089483	DroppedBlockedCalls
1	133.129373	BlockedCalls
2	91.210629	DroppedCalls
3	11.198276	MonthlyRevenue
4	6.651223	OverageMinutes
5	6.217595	MonthlyMinutes
6	5.514426	HandsetModels
7	5.181104	OffPeakCallsInOut
8	4.951826	Handsets
9	4.506253	PeakCallsInOut
10	4.261220	ReceivedCalls
11	4.122065	TotalRecurringCharge
12	3.557968	OutboundCalls
13	2.661700	MonthsInService
14	2.622790	UnansweredCalls
15	2.614363	ActiveSubs
16	2.572323	InboundCalls
17	2.531890	UniqueSubs
17	2.531890	UniqueSubs
18	2.492452	CallWaitingCalls
19	2.481722	CurrentEquipmentDays
20	2.282738	RetentionCalls
21	2.272071	RetentionOffersAccepted
22	1.632257	PercChangeMinutes
23	1.620044	PercChangeRevenues
24	1.601234	RoamingCalls
25	1.344201	CustomerCareCalls
26	1.343533	DirectorAssistedCalls
27	1.188184	ThreewayCalls
28	1.075708	IncomeGroup
29	1.071277	AdjustmentsToCreditRating
30	1.045947	Age
31	1.013083	ReferralsMadeBySubscriber
32	1.001862	CallForwardingCalls

Observation:

- The variable dropped blocked calls have high multicollinearity.

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()
```

ısistedCalls	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:                Tue, 27 Dec 2022   Pseudo R-squ.:                 0.03218
Time:                        10:23:01   Log-Likelihood:                -20484.
converged:                      False   LL-Null:                       -21165.
Covariance Type:            nonrobust   LLR p-value:                5.454e-246
```

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.

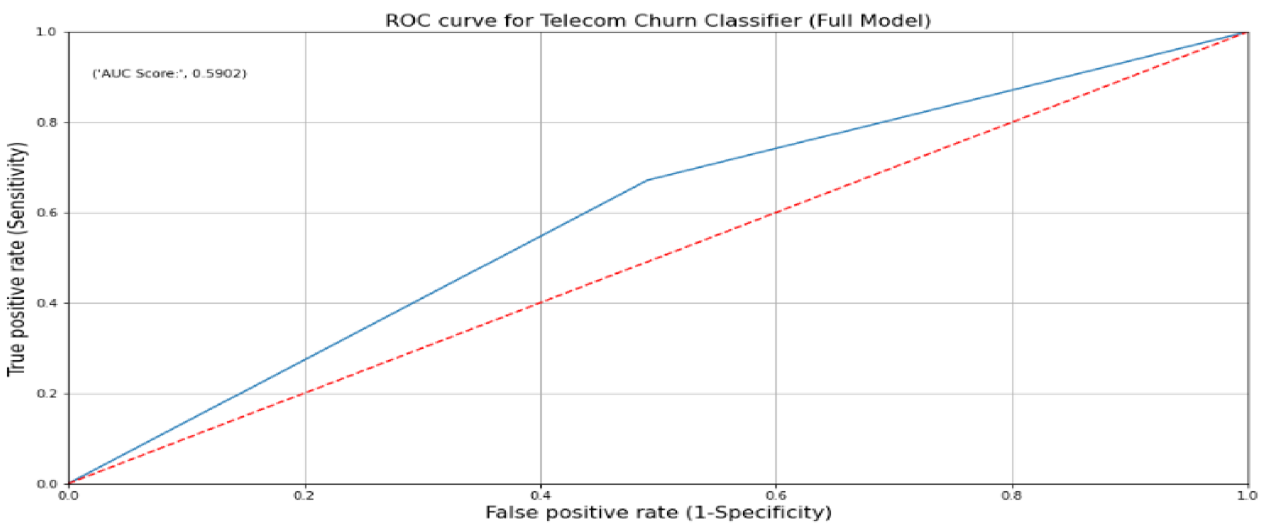
Best threshold selection report:-

	Probability Cutoff	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	0.100000	0.501590	0.292096	0.997517	0.294725	0.001859	0.451873
1	0.100000	0.501590	0.292096	0.997517	0.294725	0.001859	0.451873
2	0.100000	0.501590	0.292096	0.997517	0.294725	0.001859	0.451873
3	0.150000	0.517890	0.299186	0.978786	0.325638	0.021443	0.458287
4	0.150000	0.517890	0.299186	0.978786	0.325638	0.021443	0.458287
5	0.150000	0.517890	0.299186	0.978786	0.325638	0.021443	0.458287
6	0.200000	0.547673	0.314922	0.905890	0.398250	0.061420	0.467369
7	0.200000	0.547673	0.314922	0.905890	0.398250	0.061420	0.467369
8	0.200000	0.547673	0.314922	0.905890	0.398250	0.061420	0.467369
9	0.250000	0.582277	0.344974	0.751298	0.511773	0.122190	0.472836
10	0.260000	0.584886	0.350932	0.709546	0.532886	0.130514	0.469604
11	0.270000	0.590162	0.359961	0.671180	0.556367	0.143743	0.468605
12	0.300000	0.589415	0.382568	0.531934	0.613391	0.160396	0.445053
13	0.300000	0.589415	0.382568	0.531934	0.613391	0.160396	0.445053
14	0.300000	0.589415	0.382568	0.531934	0.613391	0.160396	0.445053
15	0.350000	0.573750	0.430589	0.323403	0.678177	0.159163	0.369377

Observation:-

- Threshold 0.27 is giving highest roc-auc score 0.59.

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 0.5902.

Report for 0.5 cutoff and best cutoff (0.27)according to Auc-score:-

```
print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.71      0.99      0.83     10773
           1       0.54      0.03      0.06      4431

    accuracy                           0.71     15204
   macro avg       0.63      0.51      0.44     15204
weighted avg       0.66      0.71      0.60     15204
```

```
print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.79      0.51      0.62     10773
           1       0.36      0.67      0.47      4431

    accuracy                           0.56     15204
   macro avg       0.58      0.59      0.54     15204
weighted avg       0.66      0.56      0.58     15204
```

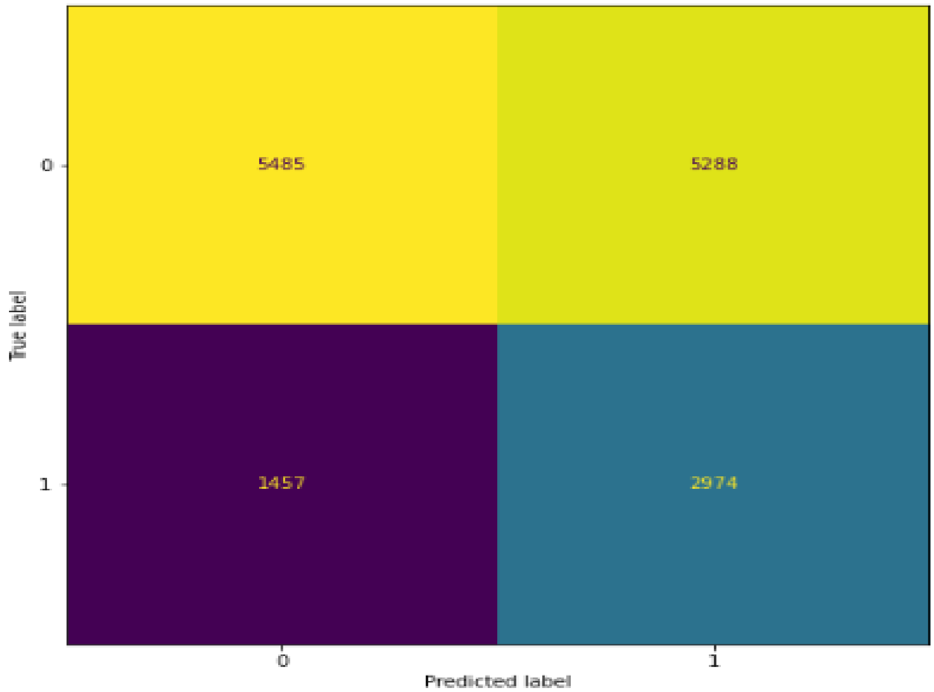
Interpretation:

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.

Confusion Matrix:-



Interpretation:

- **By the logistic regression model the maximum roc_auc score obtained by 0.27 cutoff .**
- **The accuracy for the model for the 0.27 Threshold is 0.56.**

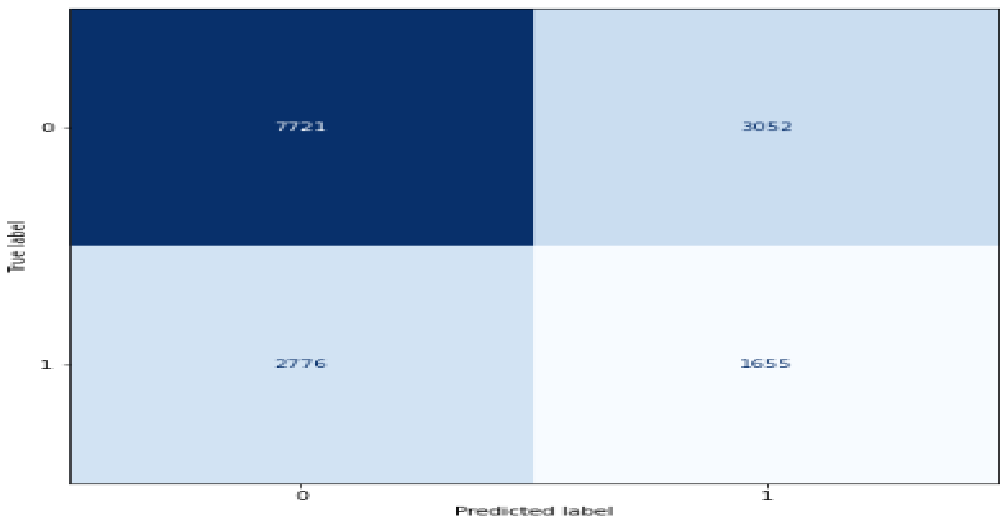
Decision Tree

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)

```
# create a dictionary with hyperparameters and its values
# pass the criteria 'entropy' and 'gini' to the parameter, 'criterion'
tuned_paramaters = [{'criterion': ['entropy', 'gini'],
                     'max_depth': range(10, 20),
                     'max_features': ["sqrt", "log2"],
                    }]
```

```
kf=KFold(n_splits=5,shuffle=True, random_state=0)
```

```
DT=DecisionTreeClassifier(random_state=0)
```

```
gr_model=GridSearchCV(estimator=DT,
    param_grid=tuned_paramaters,cv=kf)
```

```
tree_grid_model=gr_model.fit(x_train,y_train)
print('Best parameters for decision tree classifier: ', tree_grid_model.best_params_, '\n')
```

```
Best parameters for decision tree classifier:  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt'}
```

Model Performance after Tunning:

performance measures on train model

```
y_pred=dt_model.predict(x_train)
```

```
print(classification_report(y_train,y_pred))
```

	precision	recall	f1-score	support
0	0.74	0.97	0.84	25403
1	0.67	0.14	0.23	10072
accuracy			0.74	35475
macro avg	0.71	0.56	0.53	35475
weighted avg	0.72	0.74	0.67	35475

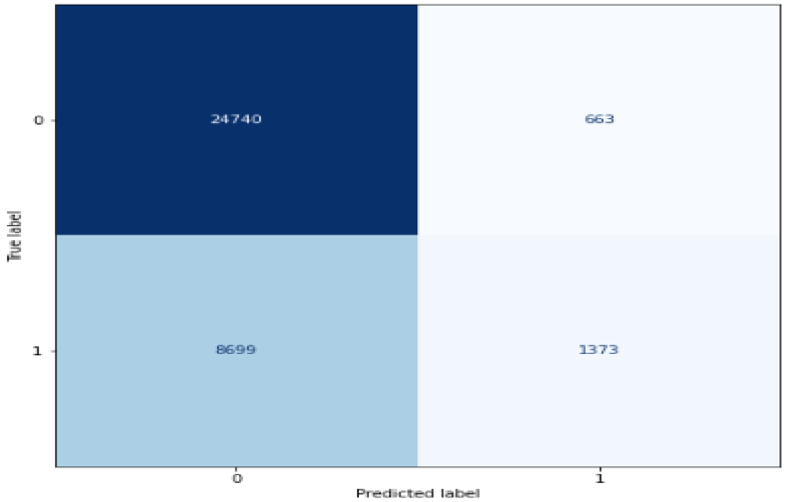
performance measures on test model

```
y_test_pred=dt_model.predict(x_test)
```

```
print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.72	0.96	0.82	10773
1	0.47	0.10	0.16	4431
accuracy			0.70	15204
macro avg	0.59	0.53	0.49	15204
weighted avg	0.65	0.70	0.63	15204

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

Random forest for classification

```
from sklearn.ensemble import RandomForestClassifier
rnd=RandomForestClassifier(random_state=0)
random_model=rnd.fit(x_train,y_train)
```

Report:

results for the train data.

```
y_pred=random_model.predict(x_train)
```

```
print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     25403
           1       1.00      1.00      1.00     10072

    accuracy                           1.00     35475
   macro avg       1.00      1.00      1.00     35475
weighted avg       1.00      1.00      1.00     35475
```

results for the test data

```
y_test_pred=random_model.predict(x_test)
```

```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.98      0.83     10773
           1       0.56      0.07      0.13      4431

    accuracy                           0.71     15204
   macro avg       0.64      0.53      0.48     15204
weighted avg       0.67      0.71      0.63     15204
```

Inferences:

- From The above model, our train accuracy is 0.1 and test accuracy is 0.71, 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)

Hyper parameters tuning by Gridsearch cv

```
]: grid={'n_estimators':range(10,100,10),'criterion':['gini','entropy'],'max_depth':range(2,25)}
```

```
]: from sklearn.model_selection import GridSearchCV,KFold
   kf=KFold(n_splits=5,shuffle=True, random_state=0)
```

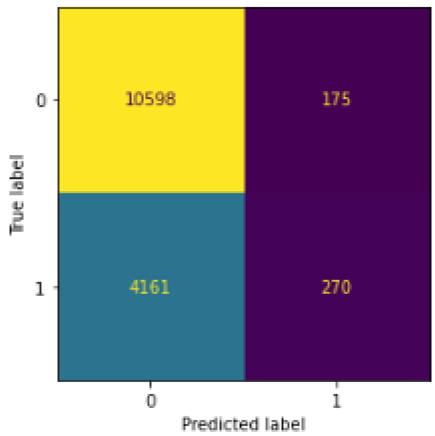
```
]: grid_model=GridSearchCV( estimator=rnd,
       param_grid=grid,
       cv=kf)
```

```
]: forest_model=grid_model.fit(x_train,y_train)
   print('Best parameters for random forest classifier: ', forest_model.best_params_, '\n')

   Best parameters for random forest classifier:  {'criterion': 'entropy', 'max_depth': 20, 'n_estimators': 70}
```

Model performance after tuning:

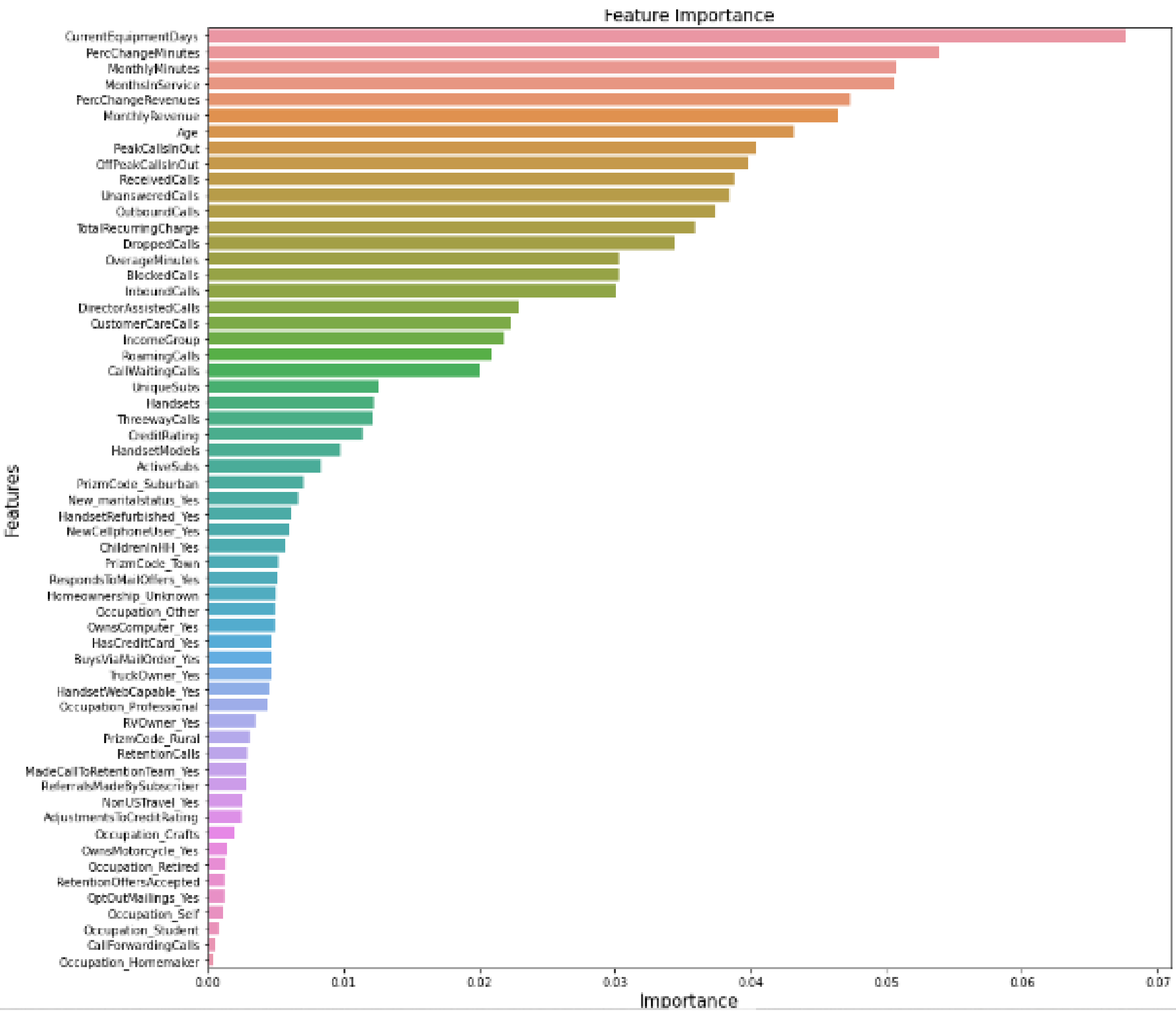
```
print(classification_report(ytest,test_predict))
              precision    recall  f1-score   support

           0       0.72      0.95      0.82     10773
           1       0.48      0.12      0.19      4431

    accuracy                           0.71     15204
   macro avg       0.60      0.53      0.51     15204
weighted avg       0.65      0.71      0.64     15204
```

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.



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.

KNN-CLASSIFIER

After Parameter Tuning:-

```
Best parameters for KNN Classifier:  {'metric': 'manhattan', 'n_neighbors': 23}

CPU times: total: 59min 21s
Wall time: 1h 8min 9s
```

```
knn_class = KNeighborsClassifier(n_neighbors = 23,metric= 'manhattan')
knn_model_1 = knn_class.fit(xtrain, ytrain)
```

Report for test:-

```
# test report
```

```
print(classification_report(ytest,test_predict))
```

```
              precision    recall  f1-score   support

           0       0.71      0.97      0.82     10773
           1       0.45      0.05      0.10      4431

    accuracy                           0.71     15204
   macro avg       0.58      0.51      0.46     15204
weighted avg       0.64      0.71      0.61     15204
```

Inference:-

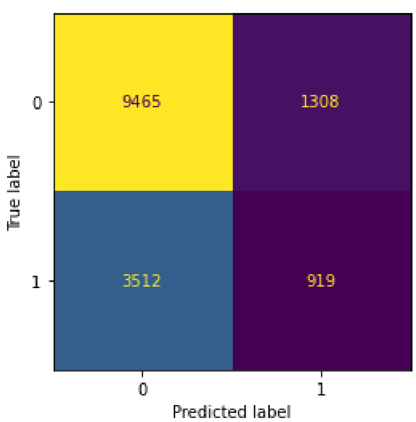
- The accuracy is 71 % which is increased compared to previous models.
- But the Recall score for Churners is reduced to 0.05.
- We try with boosting Techniques to increase the recall score.

Naive Bayes –Classifier

Train and test report:-

```
#train report

print(classification_report(ytrain,train_predict))
              precision    recall  f1-score   support

           0       0.73      0.88      0.80     25403
           1       0.39      0.19      0.26     10072

    accuracy                           0.68     35475
   macro avg       0.56      0.54      0.53     35475
weighted avg       0.64      0.68      0.65     35475

#test report

print(classification_report(ytest,test_predict))
              precision    recall  f1-score   support

           0       0.73      0.88      0.80     10773
           1       0.41      0.21      0.28      4431

    accuracy                           0.68     15204
   macro avg       0.57      0.54      0.54     15204
weighted avg       0.64      0.68      0.65     15204
```

Confusion Matrix:-



Inference:-

- Compare to previous models the Naïve Bayes is giving good recall score for churners.
- But there is slight decrease in the accuracy of the model.
- Boosting models would give good results.

greatlearning
Learning for Life

Boosting Models

1. AdaBoost classifier:-

Test report:-

```
print(classification_report(ytest,test_predict))
              precision    recall  f1-score   support

           0       0.72      0.97      0.83     10773
           1       0.57      0.09      0.16      4431

    accuracy                           0.71     15204
   macro avg       0.64      0.53      0.49     15204
weighted avg       0.68      0.71      0.63     15204
```

2. GradientBoost Classifier:-

Test report:-

```
# test report
```

```
print(classification_report(ytest,test_predict))
              precision    recall  f1-score   support

           0       0.73      0.92      0.82     10773
           1       0.49      0.19      0.27      4431

    accuracy                           0.71     15204
   macro avg       0.61      0.55      0.54     15204
weighted avg       0.66      0.71      0.66     15204
```

3. XGBoost Classifier:-

Test report:-

```
print(classification_report(ytest,test_pred))
              precision    recall  f1-score   support

           0       0.74      0.94      0.83     10773
           1       0.55      0.18      0.27      4431

    accuracy                           0.72     15204
   macro avg       0.64      0.56      0.55     15204
weighted avg       0.68      0.72      0.66     15204
```

Inference:-

- Compare to all Boosting models XGBoost model gave good Accuracy and Recall score.
- We will use Stacking Technique to increase the Recall score.

Stacking Technique

Build the stacking classifier using the Gradient Boost,Naive bayes and XGBoost as base learners (consider the hyperparameters tuned using GridSearchCV in the previous sessions).

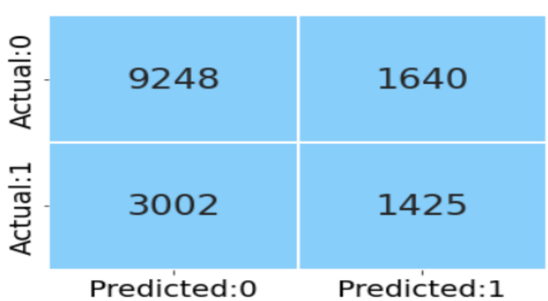
```
%%time
# consider the various algorithms as base learners
base_learners = [('Grad_model',GradientBoostingClassifier(n_estimators = 150, max_depth = 10, random_state = 10)),
                 ('xgb_model',XGBClassifier(colsample_bytree= 1, gamma= 1, learning_rate= 0.4,
                                max_depth=4, min_child_weight= 4, subsample= 1, tree_method= 'hist' )),
                 ('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())
```

Test Report:-

```
print(classification_report(ytest,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.86      0.80     10773
           1       0.47      0.31      0.37      4431

    accuracy                           0.70     15204
   macro avg       0.61      0.58      0.59     15204
weighted avg       0.67      0.70      0.68     15204
```

Confsion Matrix:-



Inference:-

- Compare to all models Stacking technique gave good accuracy of 70%.
- Recall score for churners as 0.31
- Auc_score as 0.59.
- **Compare to all models this model is best.**

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 31%
- Compare to base Decision model, the overfitting is reduced and FN errors are reduced by nearly 30%

Report Card for all models:-

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
	Threashold as 0.3								0.51	0.67	0.79	0.36	0.79	0.31	0.57
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	0.96	0.1	0.71	0.56	0.83	0.05	0.7
Random Forest	Overfit (n-estimator=70)	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	0.95	0.12	0.71	0.75	0.83	0	0.71
KNN	Only with numerical variables							0.71	0.96	0.05	0.79	0.25	0.83	0.17	0.71
Navie Bayes								0.71	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	XGBoost,Naïve Bayes,Gradient Boost							**0.72**	**0.85**	**0.31**	**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

****