TELECOM CHURN PREDICTION CASE STUDY

BY

- MANGESH CHAUDHARI
- VIPUL SASANE

BUSINESS PROBLEM STATEMENT

- In highly competitive market, the telecommunications industry experiences an average of 15-25% annual churn rate. Given the fact that it costs 5-10 times more to acquire a new customer than to retain an existing one, **customer retention** has now become even more important than customer acquisition.
- Retaining **high profitable customers** is the number one business goal.
- To reduce customer churn, telecom companies need to **predict which customers are at high risk of churn.**

OBJECTIVE

- We will analyze customer-level data of a leading telecom firm, build **predictive models** to identify customers at high risk of churn and identify the main indicators of churn.
- The **business objective** is to predict the churn in the last (i.e. the ninth) month using the data from the first three months.

STEP BY STEP APPROACH

- Data Sourcing/ Understanding
- Data Cleaning
- Data Preparation
  - New feature creation
  - Filter High value customer
  - Tag churners
- Exploratory Data Analysis
- Model Preparation
  - Train and test data split
  - Data Normalization
  - Handling class imbalance
- Model Building
- Model Evaluation
- Final Inference

DATA SOURCING/ UNDERSTANDING

- Import necessary libraries
- Import the data

```
In [1]: ### Import libraries

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')

        ### Setting max display columns

        pd.set_option('display.max_columns', 500)
        pd.set_option('display.max_rows', 500)
```

```
In [2]: ### Reading telecom churn dataset

        data= pd.read_csv('telecom_churn_data.csv')
        data.head()
```

Out[2]:

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_of_month_6	last_date_of_month_7	last_date_of_month_8	last_date_of
0	7000842753	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	
1	7001865778	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	

BASIC SANITY CHECKS

➢ Checking the information regarding the data by

1. Shape
2. Info
3. Head
4. describe

In [3]:
```
data.describe()
```

Out[3]:

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	on
count	9.999900e+04	99999.0	98981.0	98981.0	98981.0	99999.000000	99999.000000	99999.000000	99999.000000	96062.000000	96
mean	7.001207e+09	109.0	0.0	0.0	0.0	282.987358	278.536648	279.154731	261.645069	132.395875	
std	6.956694e+05	0.0	0.0	0.0	0.0	328.439770	338.156291	344.474791	341.998630	297.207406	
min	7.000000e+09	109.0	0.0	0.0	0.0	-2258.709000	-2014.045000	-945.808000	-1899.505000	0.000000	
25%	7.000606e+09	109.0	0.0	0.0	0.0	93.411500	86.980500	84.126000	62.685000	7.380000	
50%	7.001205e+09	109.0	0.0	0.0	0.0	197.704000	191.640000	192.080000	176.849000	34.310000	
75%	7.001812e+09	109.0	0.0	0.0	0.0	371.060000	365.344500	369.370500	353.466500	118.740000	
max	7.002411e+09	109.0	0.0	0.0	0.0	27731.088000	35145.834000	33543.624000	38805.617000	7376.710000	8

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99999 entries, 0 to 99998
Columns: 226 entries, mobile_number to sep_vbc_3g
dtypes: float64(179), int64(35), object(12)
memory usage: 172.4+ MB
```

In [5]:
```
data.shape
```

Out[5]: (99999, 226)

TREATMENT OF MISSING VALUES

- Find the % missing value in the data.
- Check missing values in every column.
- Treatment to be done in missing values.
- We will not drop the missing values greater than 70%, that will leads to loss of important data.

```
In [6]: print((data.isnull().sum()/len(data) * 100).sort_values(ascending= False))
```

```
arpu_3g_6                   74.846748
night_pck_user_6            74.846748
total_rech_data_6           74.846748
arpu_2g_6                   74.846748
max_rech_data_6             74.846748
fb_user_6                   74.846748
av_rech_amt_data_6          74.846748
date_of_last_rech_data_6    74.846748
count_rech_2g_6             74.846748
count_rech_3g_6             74.846748
date_of_last_rech_data_7    74.428744
total_rech_data_7           74.428744
fb_user_7                   74.428744
max_rech_data_7             74.428744
night_pck_user_7            74.428744
count_rech_2g_7             74.428744
av_rech_amt_data_7          74.428744
arpu_2g_7                   74.428744
count_rech_3g_7             74.428744
arpu_3g_7                   74.428744
total_rech_data_9           74.077741
count_rech_3g_9             74.077741
fb_user_9                   74.077741
max_rech_data_9             74.077741
arpu_3g_9                   74.077741
date_of_last_rech_data_9    74.077741
night_pck_user_9            74.077741
arpu_2g_9                   74.077741
count_rech_2g_9             74.077741
av_rech_amt_data_9          74.077741
```

TREATMENT OF MISSING VALUES

- Missing values for the column total_rech_data and date_of_last_rech_data has same missing values.
- So, we can conclude that no recharge was done and we will impute missing values as 0.

In [17]:
```python
# The logic here would be to check if columns - 'total_rech_data_6' and 'date_of_last_rech_data_6' both
# have null values at the same index. If yes, then that would mean there was no data recharge done for that month
# and we can safely impute the 'total_rech_data_6' value with 0.

total_rech_data_6_index = data['total_rech_data_6'].isnull()
date_of_last_rech_data_6_index = data['date_of_last_rech_data_6'].isnull()

if total_rech_data_6_index.equals(date_of_last_rech_data_6_index):
    print('The indexes for NULL values for month 6 are equal')
```

```
The indexes for NULL values for month 6 are equal
```

So we see that the two indexes object are equal and we can safely conclude that no data recharge was done for that month and the 'total_rech_data_6' missing values can be imputed with 0. Also as the total data recharge for the month is 0, we can impute 0 for 'av_rech_amt_data_6' column as well.

In [18]:
```python
data['total_rech_data_6'].fillna(0, inplace=True)
data['av_rech_amt_data_6'].fillna(0, inplace=True)
```

We will follow the same logic for 'total_rech_data_7', av_rech_amt_data_7, 'total_rech_data_8' & 'av_rech_amt_data_8' columns as well.

DATA PREPARATION

1. NEW FEATURE CREATION
    - we can derive new feature for the respective months called total_data_rech_amt_ which equals

total_rech_data_ * av_rech_amt_data_

1. New feature creation

Now we have values for 'total_rech_data_' and 'av_rech_amt_data_' (for months 6, 7, 8 & 9). Using these 2 values we can derive new feature for the respective months called total_data_rech_amt_ which equals total_rech_data_ * av_rech_amt_data_

```
data['total_data_rech_amt_6'] = data['total_rech_data_6'] * data['av_rech_amt_data_6']
data['total_data_rech_amt_7'] = data['total_rech_data_7'] * data['av_rech_amt_data_7']
data['total_data_rech_amt_8'] = data['total_rech_data_8'] * data['av_rech_amt_data_8']
data['total_data_rech_amt_9'] = data['total_rech_data_9'] * data['av_rech_amt_data_9']
```

DATA PREPARATION

2. Filtering High Value Customers
   - we can define High value customer as those who have recharged with an amount more than or equal to 70th percentile of the average recharge amount.

2. Filtering High value customers

```
avg_recharge_amount_month_6_7 = data[['total_data_rech_amt_6','total_data_rech_amt_7','total_rech_amt_6',
                                      'total_rech_amt_7']].mean(axis = 1)

amount_70th_percentile = np.percentile(avg_recharge_amount_month_6_7, 70)

print("70th percentile of the average recharge amount in the first two months is - ", amount_70th_percentile)
```

```
70th percentile of the average recharge amount in the first two months is -  239.0
```

```
# Filtering the high values customers

data = data[avg_recharge_amount_month_6_7 >= amount_70th_percentile]
```

```
data.shape
```

```
(30001, 229)
```

DATA PREPARATION

3. Tag Churners
   - we tag churned customers (churn=1, else 0) based on the fourth month those who have not made calls (Incoming and Outgoing).

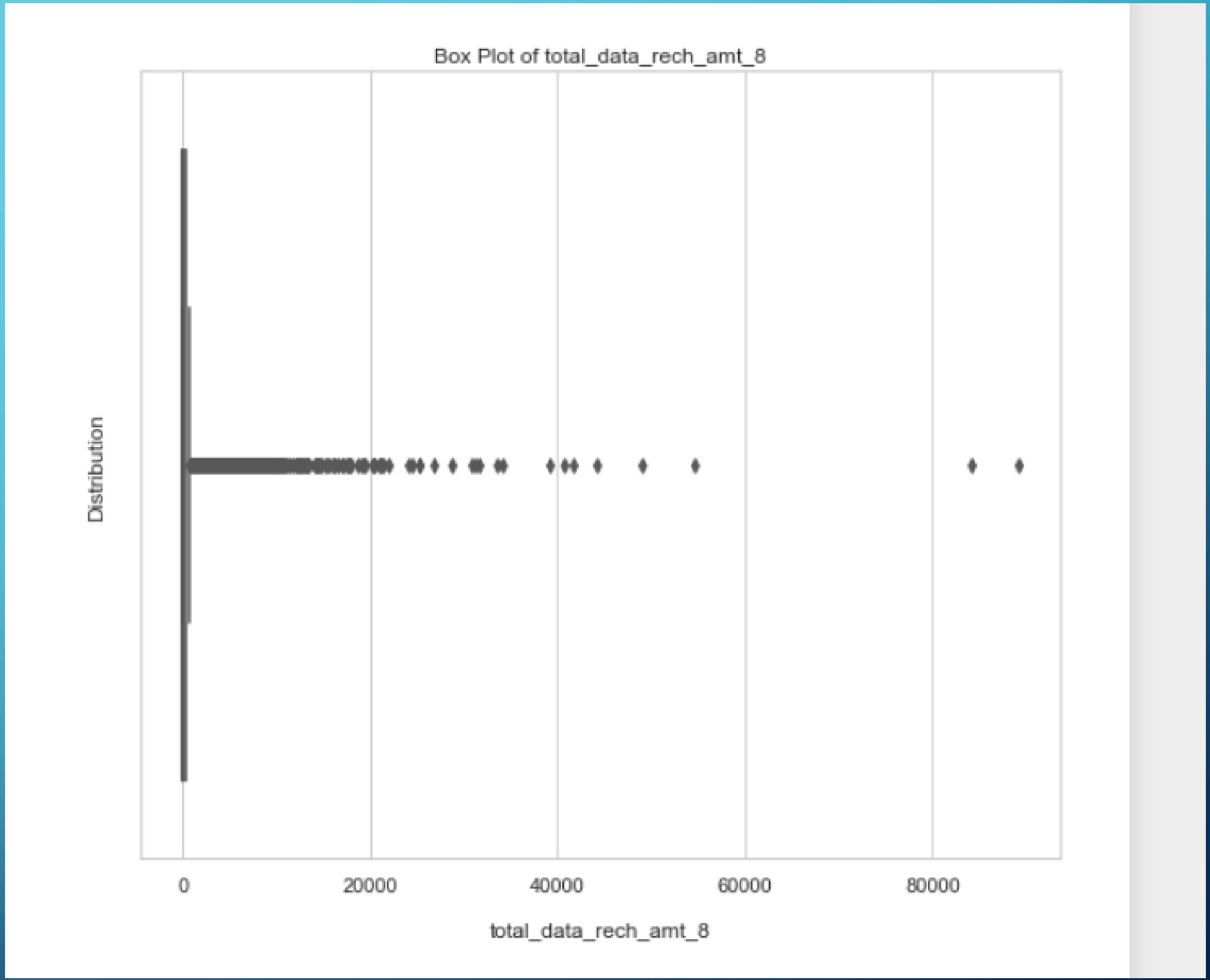
```
In [31]: data['churn'] = data.apply(lambda x: 1 if((x.total_ic_mou_9 == 0) and
                                                   (x.total_og_mou_9 == 0) and
                                                   (x.vol_2g_mb_9 == 0) and
                                                   (x.vol_3g_mb_9 == 0)) else 0, axis=1)
```

```
In [32]: data['churn'].head()
```

```
Out[32]: mobile_number
         7000842753    1
         7000701601    1
         7001524846    0
         7002124215    0
         7000887461    0
         Name: churn, dtype: int64
```

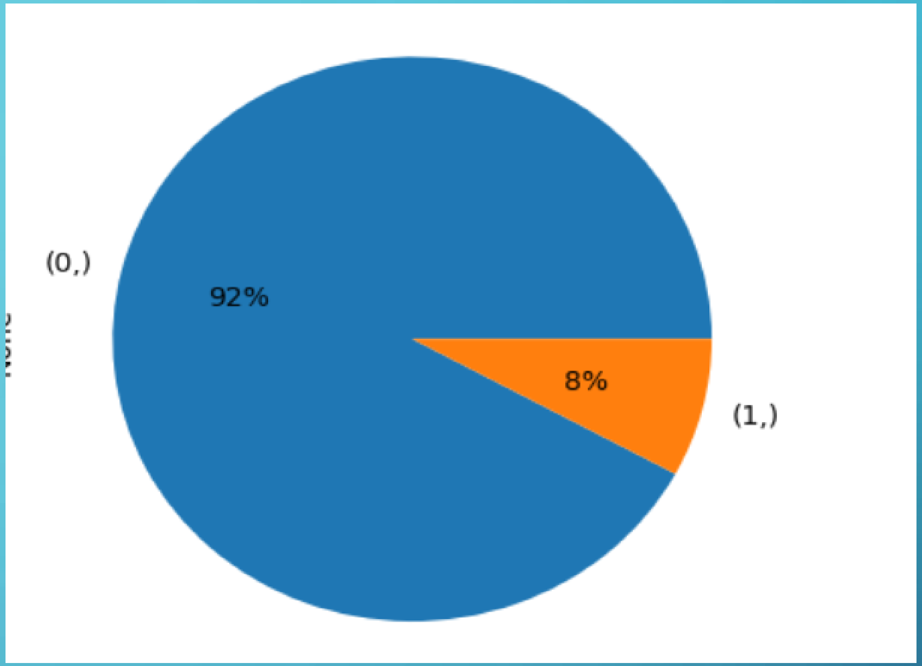
OUTLIERS IDENTIFICATION AND HANDLING

- There are Outliers present in the data.
- We will removing them by normalizing.



CHECKING OF DATA IMBALANCE

- There is data imbalance present in data which is handled by SMOTE.



Class Imbalance

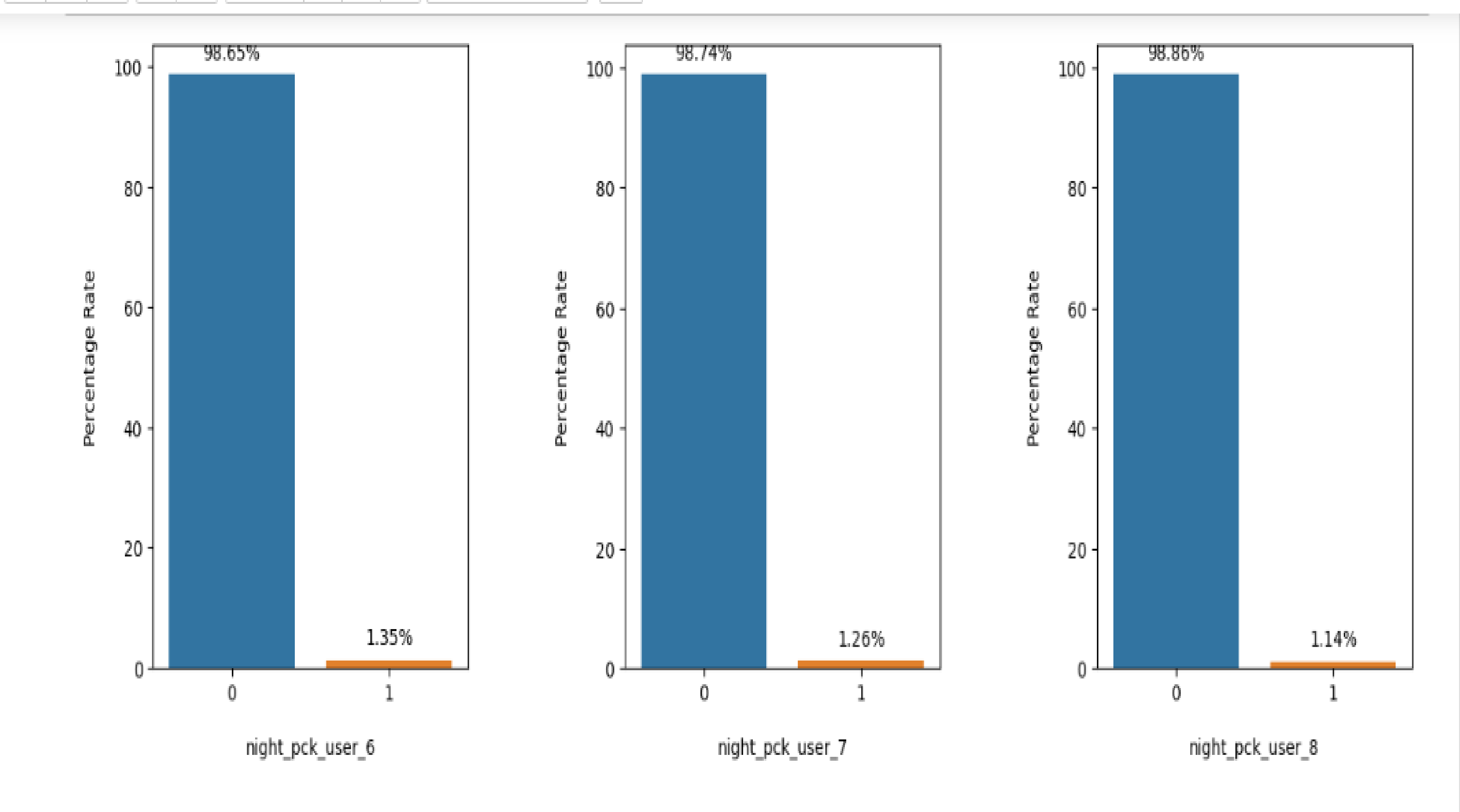
```
In [83]: y.value_counts(normalize=True).to_frame()
Out[83]:
```

	churn
0	0.920735
1	0.079265

ANALYSIS PART OF DATA

UNIVARIATE ANALYSIS

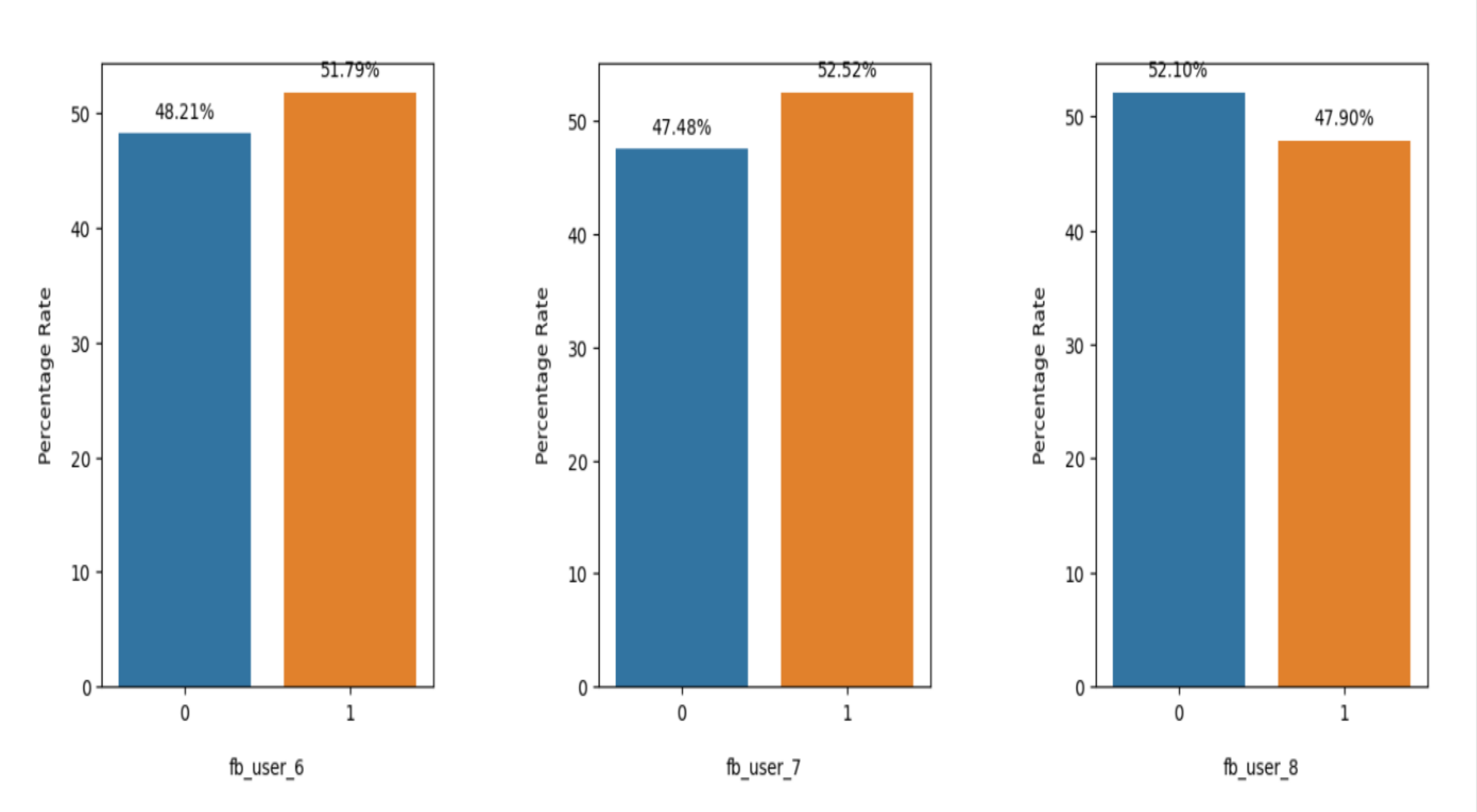
- Analyze night pack user.



INSIGHT-
99% of customers are not using night packs.

UNIVARIATE ANALYSIS

- Analyze fb user column.

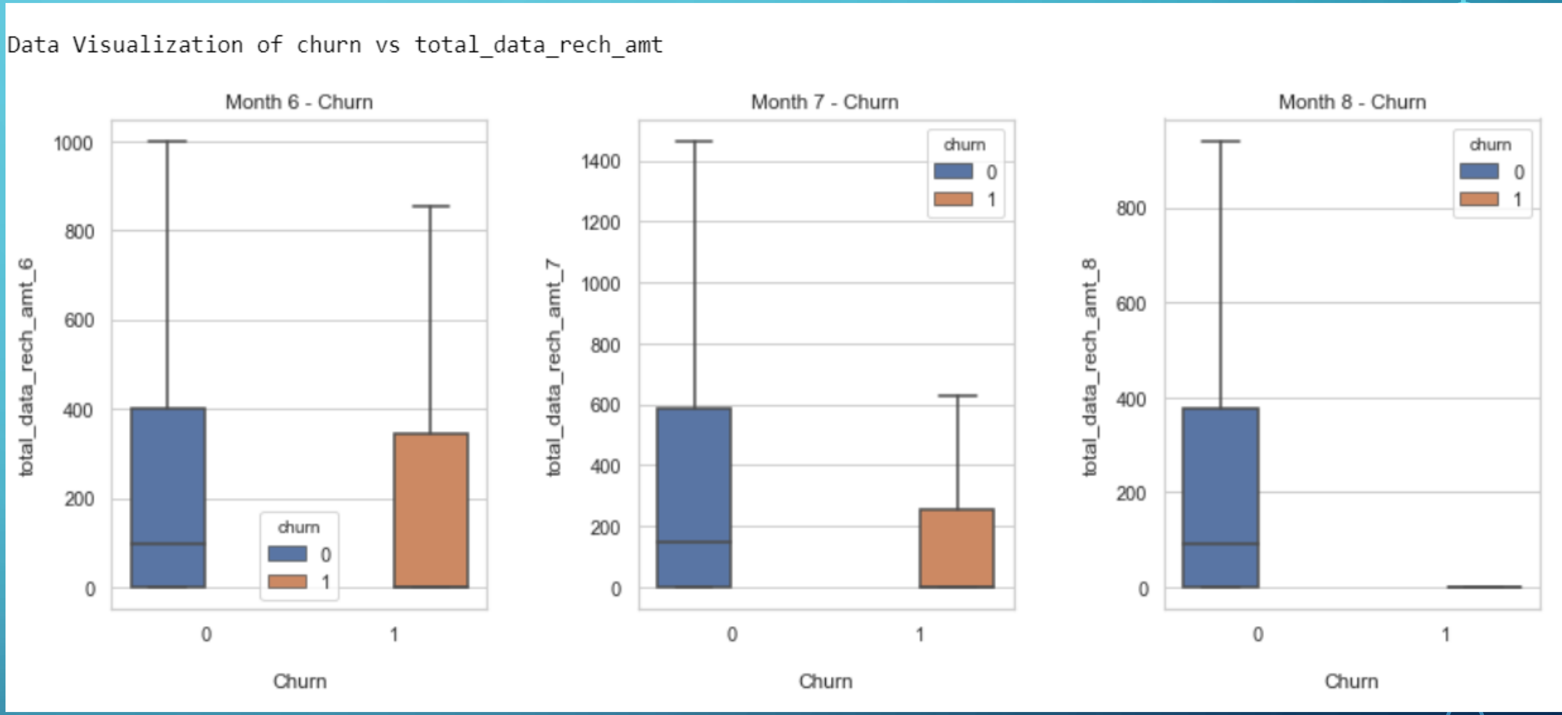


INSIGHT-

The percentage of churn and not churn are same in 3 months.

BIVARIATE ANALYSIS

- Analyze the churn vs total data recharge amount.

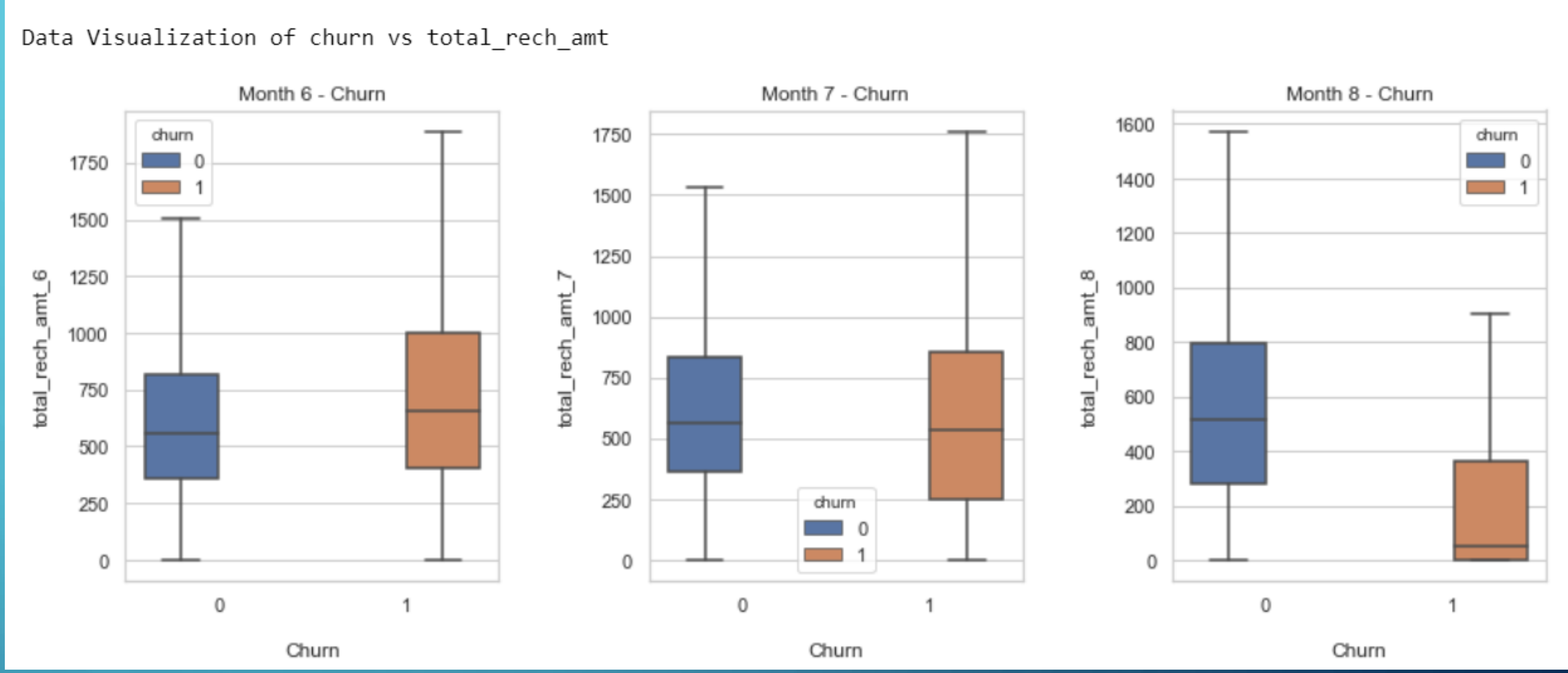


INSIGHT-

There is significant drop observed in 8th month.

BIVARIATE ANALYSIS

- Analyze the Churn and Total recharge amount.



INSIGHT-

- ➢ The churn probability increased with increase in total recharge amount in month 6 and month 7.

BUILDING MULTIPLE MODELS

1. Logistic Regression with RFE.
2. PCA+ Logistic Regression
3. PCA+ Random forest classifier

LOGISTIC REGRESSION WITH RFE

```
Train Performance:

Accuracy : 0.804
Sensitivity / True Positive Rate / Recall : 0.802
Specificity / True Negative Rate :  0.806
Precision / Positive Predictive Value : 0.805
F1-score : 0.803


Test Performance :

Accuracy : 0.801
Sensitivity / True Positive Rate / Recall : 0.65
Specificity / True Negative Rate :  0.814
Precision / Positive Predictive Value : 0.242
F1-score : 0.353
```

```
ROC AUC score for Train :  0.877

ROC AUC score for Test :  0.801
```

PCA+ LOGISTIC REGRESSION

```
Train Performance :

Accuracy : 0.561
Sensitivity / True Positive Rate / Recall : 0.93
Specificity / True Negative Rate :  0.529
Precision / Positive Predictive Value : 0.142
F1-score : 0.246

Test Performance :

Accuracy : 0.634
Sensitivity / True Positive Rate / Recall : 0.669
Specificity / True Negative Rate :  0.631
Precision / Positive Predictive Value : 0.141
F1-score : 0.233
```

PCA + RANDOM FOREST CLASSIFIER

```
Train Performance :

Accuracy : 0.876
Sensitivity / True Positive Rate / Recall : 0.81
Specificity / True Negative Rate :  0.881
Precision / Positive Predictive Value : 0.364
F1-score : 0.502

Test Performance :

Accuracy : 0.836
Sensitivity / True Positive Rate / Recall : 0.568
Specificity / True Negative Rate :  0.861
Precision / Positive Predictive Value : 0.271
F1-score : 0.367
```

5]:
```
## out of bag error
pca_rf_best_fit.oob_score_
```

5]:
```
0.8597911477294501
```

MOST IMPORTANT PREDICTORS OF CHURN

```
Most Important Predictors of churn , in the order of importance are :

]: const                    1.9137
   fb_user_8               -1.2250
   monthly_2g_7            -1.1677
   sachet_2g_8_0           -0.8884
   count_rech_3g_7_0.0     -0.7465
   monthly_2g_6            -0.6385
   total_rech_data_8       -0.2816
   sachet_2g_7_0           -0.2508
   total_rech_num_8        -0.2053
   count_rech_2g_6          0.1390
   total_rech_num_7         0.0835
   Name: coef, dtype: float64
```

RECOMMENDATIONS-

- The customer who used the 2G network in 7th month likely to churn.
- The customer who avail for Facebook and other social sites in 8th month are likely to churn.
- Models with high sensitivity are the best for predicting churn. Use the PCA + Logistic Regression model to predict churn. It has an ROC score of 0.87, test sensitivity of 67%.

THANK YOU