

Telecom Churn Case Study

Analysis Approach :

- 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 become even more important than customer acquisition.
- Here we are given with 4 months of data related to customer usage. In this case study, we analyse 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.
- Churn is predicted using two approaches. Usage based churn and Revenue based churn. Usage based churn:
- Customers who have zero usage, either incoming or outgoing - in terms of calls, internet etc. over a period of time.
- This case study only considers usage based churn.
- In the Indian and the southeast Asian market, approximately 80% of revenue comes from the top 20% customers (called high-value customers). Thus, if we can reduce churn of the high-value customers, we will be able to reduce significant revenue leakage. Hence, this case study focuses on high value customers only.
- The dataset contains customer-level information for a span of four consecutive months - June, July, August and September. The months are encoded as 6, 7, 8 and 9, respectively.
- The **business objective** is to predict the churn in the last (i.e. the ninth) month using the data (features) from the first three months.
- This is a classification problem, where we need to predict whether the customers is about to churn or not. We have carried out Baseline Logistic Regression, then Logistic Regression with PCA, PCA + Random Forest, PCA + XGBoost.

Analysis Steps

Data Cleaning and EDA

1. We have started with importing Necessary packages and libraries.
2. We have loaded the dataset into a dataframe.
3. We have checked the number of columns, their data types, Null count and unique value_value_count to get some understanding about data and to check if the columns are under correct data-type.
4. Checking for duplicate records (rows) in the data. There were no duplicates.
5. Since 'mobile_number' is the unique identifier available, we have made it our index to retain the identity.
6. Have found some columns that do not follow the naming standard, we have renamed those columns to make sure all the variables follow the same naming convention.
7. Following with column renaming, we have dealt with converting the columns into their respective data types. Here, we have evaluated all the columns which are having less than or equal to 29 unique values as categorical columns and rest as continuous columns.
8. The date columns were having 'object' as their data type, we have converted to the proper datetime format.
9. Since, our analysis is focused on the HVC(High value customers), we have filtered for high value customers to carryout the further analysis. The metric of this filtering of HVC is such that all the customers whose 'Average_rech_amt' of months 6 and 7 greater than or equal to 70th percentile of the 'Average_rech_amt' are considered as High Value Customers.
10. Checked for missing values.
11. Dropped all the columns with missing values greater than 50%.
12. We have been given 4 months data. Since each months revenue and usage data is not related to other, we did month-wise drill down on missing values.

13. Some columns had similar range of missing values. So, we have looked at their related columns and checked if these might be imputed with zero.
14. We have found that 'last_date_of_the_month' had some misisng values, so this is very meaningful and we have imputed the last date based on the month.
15. We have found some columns with only one unique value, so it is of no use for the analysis, hence we have dropped those columns.
16. Once after checking all the data preparation tasks, tagged the Churn variable(which is our target variable).
17. After imputing, we have dropped churn phase columns (Columns belonging to month - 9).
18. After all the above processing, we have retained 30,011 rows and 126 columns.
19. Exploratory Data Analysis

- The telecom company has many users with negative average revenues in both phases. These users are likely to churn.
- Most customers prefer the plans of '0' category.
- The customers with lesser 'aon' are more likely to Churn when compared to the Customers with higer 'aon'.
- Revenue generated by the Customers who are about to churn is very unstable.
- The Customers whose arpu decreases in 7th month are more likely to churn when compared to ones with increase in arpu.
- The Customers with high total_og_mou in 6th month and lower total_og_mou in 7th month are more likely to churn compared to the rest.
- The Customers with decrease in rate of total_ic_mou in 7th month are more likely to churn, compared to the rest.
- Customers with stable usage of 2g volume throughout 6 and 7 months are less likely to churn.
- Customers with fall in usage of 2g volume in 7th month are more likely to Churn.
- Customers with stable usage of 3g volume throughout 6 and 7 months are less likely to churn.
- Customers with fall in consumption of 3g volume in 7th month are more likely to Churn.
- The customers with lower total_og_mou in 6th and 8th months are more likely to Churn compared to the ones with higher total_og_mou.
- The customers with lesser total_og_mou_8 and aon are more likely to churn compared to the one with higher total_og_mou_8 and aon.
- The customers with less total_ic_mou_8 are more likely to churn irrespective of aon.
- The customers with total_ic_mou_8 > 2000 are very less likely to churn.

1. Correlation analysis has been performed.
2. We have created the derived variables and then removed the variables that were used to derive new ones.
3. Outlier treatment has been performed. We have looked at the quantiles to understand the spread of Data.
4. We have capped the upper outliers to 99th percentile.
5. We have checked categorical variables and contribution of classes in those variables. The classes with less ccontribution are grouped into 'Others'.
6. Dummy Variables were created.

Pre-processing Steps

1. Train-Test Split has been performed.
2. The data has high class-imbalance with the ratio of 0.095 (class 1 : class 0).
3. SMOTE technique has been used to overcome class-imbalance.
4. Predictor columns have been standardized to mean - 0 and standard_deviatiom- 1.

Modelling

Model 1 : Logistic Regression with RFE & Manual Elimination (Interpretable Model)

Most important predictors of Churn , in order of importance and their coefficients are as follows :

- loc_ic_t2f_mou_8 -1.2736
- total_rech_num_8 -1.2033
- total_rech_num_6 0.6053
- monthly_3g_8_0 0.3994
- monthly_2g_8_0 0.3666
- std_ic_t2f_mou_8 -0.3363
- std_og_t2f_mou_8 -0.2474
- const -0.2336
- monthly_3g_7_0 -0.2099
- std_ic_t2f_mou_7 0.1532
- sachet_2g_6_0 -0.1108
- sachet_2g_7_0 -0.0987
- sachet_2g_8_0 0.0488
- sachet_3g_6_0 -0.0399

PCA: PCA : 95% of variance in the train set can be explained by first 16 principal components and 100% of variance is explained by the first 45 principal components.

Model 2 : PCA + Logistic Regression

Train Performance :

```
Accuracy : 0.627
Sensitivity / True Positive Rate / Recall : 0.918
Specificity / True Negative Rate : 0.599
Precision / Positive Predictive Value : 0.179
F1-score : 0.3
```

Test Performance :

```
Accuracy : 0.086
Sensitivity / True Positive Rate / Recall : 1.0
Specificity / True Negative Rate : 0.0
Precision / Positive Predictive Value : 0.086
F1-score : 0.158
```

Model 3 : PCA + Random Forest Classifier

Train Performance :

```
Accuracy : 0.882
Sensitivity / True Positive Rate / Recall : 0.816
Specificity / True Negative Rate :  0.888
Precision / Positive Predictive Value : 0.408
F1-score : 0.544
```

Test Performance :

```
Accuracy : 0.86
Sensitivity / True Positive Rate / Recall : 0.80
Specificity / True Negative Rate :  0.78
Precision / Positive Predictive Value : 0.37
F1-score : 0.51
```

Model 4 : PCA + XGBoost

Train Performance :

```
Accuracy : 0.873
Sensitivity / True Positive Rate / Recall : 0.887
Specificity / True Negative Rate :  0.872
Precision / Positive Predictive Value : 0.396
F1-score : 0.548
```

Test Performance :

```
Accuracy : 0.086
Sensitivity / True Positive Rate / Recall : 1.0
Specificity / True Negative Rate :  0.0
Precision / Positive Predictive Value : 0.086
F1-score : 0.158
```

Recommendations :

Following are the strongest indicators of churn

Customers who churn show lower average monthly local incoming calls from fixed line in the action period by 1.27 standard deviations , compared to users who don't churn , when all other factors are held constant. This is the strongest indicator of churn. Customers who churn show lower number of recharges done in action period by 1.20 standard deviations, when all other factors are held constant. This is the second strongest indicator of churn. Further customers who churn have done 0.6 standard deviations higher recharge than non-churn customers. This factor when coupled with above factors is a good indicator of churn. Customers who churn are more likely to be users of 'monthly 2g package-0 / monthly 3g package-0' in action period (approximately 0.3 std deviations higher than other packages), when all other factors are held constant.

Based on the above indicators the recommendations to the telecom company are :

Concentrate on users with 1.27 std deviations lower than average incoming calls from fixed line. They are most likely to churn. Concentrate on users who recharge less number of times (less than 1.2 std deviations compared to avg) in the 8th month. They are second most 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 100%.

Analysis

Data Understanding

In [6]:

```
# Importing Necessary Libraries.
import numpy as np, pandas as pd, matplotlib.pyplot as plt, seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Setting max display columns and rows.
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
```

In [7]:

```
# Reading Dataset into a DataFrame.
data=pd.read_csv('telecom_churn_data.csv')
data.head()
```

Out[7]:

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_of_mou
0	7000842753	109	0.0	0.0	0.0	6/3
1	7001865778	109	0.0	0.0	0.0	6/3
2	7001625959	109	0.0	0.0	0.0	6/3
3	7001204172	109	0.0	0.0	0.0	6/3
4	7000142493	109	0.0	0.0	0.0	6/3

In [8]:

```
# Checking information about data.
print(data.info())
def metadata_matrix(data) :
    return pd.DataFrame({
        'Datatype' : data.dtypes.astype(str),
        'Non_Null_Count': data.count(axis = 0).astype(int),
        'Null_Count': data.isnull().sum().astype(int),
        'Null_Percentage': round(data.isnull().sum()/len(data) * 100 ,
2),
        'Unique_Values_Count': data.nunique().astype(int)
    }).sort_values(by='Null_Percentage', ascending=False)

metadata_matrix(data)
```

```
<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
None
```

Out[8]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
arpu_3g_6	float64	25153	74846	74.85	
night_pck_user_6	float64	25153	74846	74.85	
total_rech_data_6	float64	25153	74846	74.85	
arpu_2g_6	float64	25153	74846	74.85	
max_rech_data_6	float64	25153	74846	74.85	
fb_user_6	float64	25153	74846	74.85	
av_rech_amt_data_6	float64	25153	74846	74.85	
date_of_last_rech_data_6	object	25153	74846	74.85	
count_rech_2g_6	float64	25153	74846	74.85	
count_rech_3g_6	float64	25153	74846	74.85	
date_of_last_rech_data_7	object	25571	74428	74.43	
total_rech_data_7	float64	25571	74428	74.43	
fb_user_7	float64	25571	74428	74.43	
max_rech_data_7	float64	25571	74428	74.43	
night_pck_user_7	float64	25571	74428	74.43	
count_rech_2g_7	float64	25571	74428	74.43	
av_rech_amt_data_7	float64	25571	74428	74.43	
arpu_2g_7	float64	25571	74428	74.43	
count_rech_3g_7	float64	25571	74428	74.43	
arpu_3g_7	float64	25571	74428	74.43	
total_rech_data_9	float64	25922	74077	74.08	
count_rech_3g_9	float64	25922	74077	74.08	
fb_user_9	float64	25922	74077	74.08	
max_rech_data_9	float64	25922	74077	74.08	
arpu_3g_9	float64	25922	74077	74.08	
date_of_last_rech_data_9	object	25922	74077	74.08	
night_pck_user_9	float64	25922	74077	74.08	
arpu_2g_9	float64	25922	74077	74.08	
count_rech_2g_9	float64	25922	74077	74.08	
av_rech_amt_data_9	float64	25922	74077	74.08	
total_rech_data_8	float64	26339	73660	73.66	
arpu_3g_8	float64	26339	73660	73.66	
fb_user_8	float64	26339	73660	73.66	
night_pck_user_8	float64	26339	73660	73.66	
av_rech_amt_data_8	float64	26339	73660	73.66	
max_rech_data_8	float64	26339	73660	73.66	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
count_rech_3g_8	float64	26339	73660	73.66	
arpu_2g_8	float64	26339	73660	73.66	
count_rech_2g_8	float64	26339	73660	73.66	
date_of_last_rech_data_8	object	26339	73660	73.66	
ic_others_9	float64	92254	7745	7.75	
std_og_mou_9	float64	92254	7745	7.75	
std_og_t2c_mou_9	float64	92254	7745	7.75	
isd_ic_mou_9	float64	92254	7745	7.75	
std_ic_mou_9	float64	92254	7745	7.75	
isd_og_mou_9	float64	92254	7745	7.75	
spl_og_mou_9	float64	92254	7745	7.75	
spl_ic_mou_9	float64	92254	7745	7.75	
og_others_9	float64	92254	7745	7.75	
loc_ic_t2t_mou_9	float64	92254	7745	7.75	
std_ic_t2o_mou_9	float64	92254	7745	7.75	
loc_ic_t2m_mou_9	float64	92254	7745	7.75	
std_ic_t2f_mou_9	float64	92254	7745	7.75	
loc_ic_t2f_mou_9	float64	92254	7745	7.75	
loc_ic_mou_9	float64	92254	7745	7.75	
std_ic_t2m_mou_9	float64	92254	7745	7.75	
std_og_t2f_mou_9	float64	92254	7745	7.75	
std_og_t2t_mou_9	float64	92254	7745	7.75	
std_ic_t2t_mou_9	float64	92254	7745	7.75	
loc_og_mou_9	float64	92254	7745	7.75	
roam_og_mou_9	float64	92254	7745	7.75	
loc_og_t2m_mou_9	float64	92254	7745	7.75	
loc_og_t2f_mou_9	float64	92254	7745	7.75	
roam_ic_mou_9	float64	92254	7745	7.75	
offnet_mou_9	float64	92254	7745	7.75	
loc_og_t2c_mou_9	float64	92254	7745	7.75	
loc_og_t2t_mou_9	float64	92254	7745	7.75	
std_og_t2m_mou_9	float64	92254	7745	7.75	
onnet_mou_9	float64	92254	7745	7.75	
onnet_mou_8	float64	94621	5378	5.38	
std_ic_t2t_mou_8	float64	94621	5378	5.38	
std_ic_mou_8	float64	94621	5378	5.38	
loc_ic_t2t_mou_8	float64	94621	5378	5.38	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
roam_og_mou_8	float64	94621	5378	5.38	
std_ic_t2m_mou_8	float64	94621	5378	5.38	
loc_ic_mou_8	float64	94621	5378	5.38	
std_ic_t2f_mou_8	float64	94621	5378	5.38	
roam_ic_mou_8	float64	94621	5378	5.38	
std_ic_t2o_mou_8	float64	94621	5378	5.38	
loc_og_t2t_mou_8	float64	94621	5378	5.38	
loc_ic_t2f_mou_8	float64	94621	5378	5.38	
offnet_mou_8	float64	94621	5378	5.38	
loc_ic_t2m_mou_8	float64	94621	5378	5.38	
loc_og_t2m_mou_8	float64	94621	5378	5.38	
isd_og_mou_8	float64	94621	5378	5.38	
ic_others_8	float64	94621	5378	5.38	
og_others_8	float64	94621	5378	5.38	
spl_ic_mou_8	float64	94621	5378	5.38	
loc_og_t2f_mou_8	float64	94621	5378	5.38	
std_og_t2m_mou_8	float64	94621	5378	5.38	
spl_og_mou_8	float64	94621	5378	5.38	
std_og_t2c_mou_8	float64	94621	5378	5.38	
isd_ic_mou_8	float64	94621	5378	5.38	
loc_og_t2c_mou_8	float64	94621	5378	5.38	
std_og_t2f_mou_8	float64	94621	5378	5.38	
std_og_t2t_mou_8	float64	94621	5378	5.38	
loc_og_mou_8	float64	94621	5378	5.38	
std_og_mou_8	float64	94621	5378	5.38	
date_of_last_rech_9	object	95239	4760	4.76	
std_ic_t2f_mou_6	float64	96062	3937	3.94	
ic_others_6	float64	96062	3937	3.94	
isd_ic_mou_6	float64	96062	3937	3.94	
std_ic_t2m_mou_6	float64	96062	3937	3.94	
std_ic_mou_6	float64	96062	3937	3.94	
spl_ic_mou_6	float64	96062	3937	3.94	
std_ic_t2o_mou_6	float64	96062	3937	3.94	
loc_ic_t2f_mou_6	float64	96062	3937	3.94	
loc_ic_t2t_mou_6	float64	96062	3937	3.94	
std_og_t2c_mou_6	float64	96062	3937	3.94	
std_og_t2f_mou_6	float64	96062	3937	3.94	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
std_og_mou_6	float64	96062	3937	3.94	
std_og_t2m_mou_6	float64	96062	3937	3.94	
isd_og_mou_6	float64	96062	3937	3.94	
std_og_t2t_mou_6	float64	96062	3937	3.94	
spl_og_mou_6	float64	96062	3937	3.94	
loc_og_mou_6	float64	96062	3937	3.94	
og_others_6	float64	96062	3937	3.94	
loc_og_t2c_mou_6	float64	96062	3937	3.94	
loc_og_t2m_mou_6	float64	96062	3937	3.94	
loc_og_t2f_mou_6	float64	96062	3937	3.94	
loc_og_t2t_mou_6	float64	96062	3937	3.94	
roam_og_mou_6	float64	96062	3937	3.94	
std_ic_t2t_mou_6	float64	96062	3937	3.94	
onnet_mou_6	float64	96062	3937	3.94	
loc_ic_mou_6	float64	96062	3937	3.94	
offnet_mou_6	float64	96062	3937	3.94	
roam_ic_mou_6	float64	96062	3937	3.94	
loc_ic_t2m_mou_6	float64	96062	3937	3.94	
loc_og_t2c_mou_7	float64	96140	3859	3.86	
roam_ic_mou_7	float64	96140	3859	3.86	
loc_og_mou_7	float64	96140	3859	3.86	
loc_og_t2t_mou_7	float64	96140	3859	3.86	
offnet_mou_7	float64	96140	3859	3.86	
loc_og_t2f_mou_7	float64	96140	3859	3.86	
std_og_t2t_mou_7	float64	96140	3859	3.86	
std_ic_t2t_mou_7	float64	96140	3859	3.86	
onnet_mou_7	float64	96140	3859	3.86	
std_og_t2m_mou_7	float64	96140	3859	3.86	
loc_og_t2m_mou_7	float64	96140	3859	3.86	
std_og_t2f_mou_7	float64	96140	3859	3.86	
roam_og_mou_7	float64	96140	3859	3.86	
std_og_t2c_mou_7	float64	96140	3859	3.86	
std_ic_t2m_mou_7	float64	96140	3859	3.86	
isd_og_mou_7	float64	96140	3859	3.86	
ic_others_7	float64	96140	3859	3.86	
loc_ic_t2f_mou_7	float64	96140	3859	3.86	
loc_ic_t2m_mou_7	float64	96140	3859	3.86	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
std_ic_mou_7	float64	96140	3859	3.86	
loc_ic_t2t_mou_7	float64	96140	3859	3.86	
std_ic_t2f_mou_7	float64	96140	3859	3.86	
loc_ic_mou_7	float64	96140	3859	3.86	
spl_ic_mou_7	float64	96140	3859	3.86	
og_others_7	float64	96140	3859	3.86	
spl_og_mou_7	float64	96140	3859	3.86	
isd_ic_mou_7	float64	96140	3859	3.86	
std_ic_t2o_mou_7	float64	96140	3859	3.86	
std_og_mou_7	float64	96140	3859	3.86	
date_of_last_rech_8	object	96377	3622	3.62	
date_of_last_rech_7	object	98232	1767	1.77	
last_date_of_month_9	object	98340	1659	1.66	
date_of_last_rech_6	object	98392	1607	1.61	
last_date_of_month_8	object	98899	1100	1.10	
loc_ic_t2o_mou	float64	98981	1018	1.02	
std_og_t2o_mou	float64	98981	1018	1.02	
loc_og_t2o_mou	float64	98981	1018	1.02	
last_date_of_month_7	object	99398	601	0.60	
sachet_3g_8	int64	99999	0	0.00	
jul_vbc_3g	float64	99999	0	0.00	
aug_vbc_3g	float64	99999	0	0.00	
aon	int64	99999	0	0.00	
jun_vbc_3g	float64	99999	0	0.00	
monthly_2g_9	int64	99999	0	0.00	
sachet_3g_6	int64	99999	0	0.00	
vol_3g_mb_9	float64	99999	0	0.00	
sachet_3g_7	int64	99999	0	0.00	
monthly_2g_8	int64	99999	0	0.00	
monthly_3g_9	int64	99999	0	0.00	
monthly_3g_8	int64	99999	0	0.00	
sachet_3g_9	int64	99999	0	0.00	
monthly_3g_7	int64	99999	0	0.00	
monthly_3g_6	int64	99999	0	0.00	
sachet_2g_9	int64	99999	0	0.00	
sachet_2g_8	int64	99999	0	0.00	
sachet_2g_7	int64	99999	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
sachet_2g_6	int64	99999	0	0.00	
monthly_2g_7	int64	99999	0	0.00	
monthly_2g_6	int64	99999	0	0.00	
mobile_number	int64	99999	0	0.00	
vol_3g_mb_8	float64	99999	0	0.00	
total_og_mou_9	float64	99999	0	0.00	
total_rech_num_7	int64	99999	0	0.00	
total_rech_num_6	int64	99999	0	0.00	
total_ic_mou_9	float64	99999	0	0.00	
total_ic_mou_8	float64	99999	0	0.00	
total_ic_mou_7	float64	99999	0	0.00	
total_ic_mou_6	float64	99999	0	0.00	
circle_id	int64	99999	0	0.00	
total_og_mou_8	float64	99999	0	0.00	
vol_3g_mb_7	float64	99999	0	0.00	
total_og_mou_7	float64	99999	0	0.00	
total_og_mou_6	float64	99999	0	0.00	
arpu_9	float64	99999	0	0.00	
arpu_8	float64	99999	0	0.00	
arpu_7	float64	99999	0	0.00	
arpu_6	float64	99999	0	0.00	
last_date_of_month_6	object	99999	0	0.00	
total_rech_num_8	int64	99999	0	0.00	
total_rech_num_9	int64	99999	0	0.00	
total_rech_amt_6	int64	99999	0	0.00	
total_rech_amt_7	int64	99999	0	0.00	
vol_3g_mb_6	float64	99999	0	0.00	
vol_2g_mb_9	float64	99999	0	0.00	
vol_2g_mb_8	float64	99999	0	0.00	
vol_2g_mb_7	float64	99999	0	0.00	
vol_2g_mb_6	float64	99999	0	0.00	
last_day_rch_amt_9	int64	99999	0	0.00	
last_day_rch_amt_8	int64	99999	0	0.00	
last_day_rch_amt_7	int64	99999	0	0.00	
last_day_rch_amt_6	int64	99999	0	0.00	
max_rech_amt_9	int64	99999	0	0.00	
max_rech_amt_8	int64	99999	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Value
max_rech_amt_7	int64	99999	0	0.00	
max_rech_amt_6	int64	99999	0	0.00	
total_rech_amt_9	int64	99999	0	0.00	
total_rech_amt_8	int64	99999	0	0.00	
sep_vbc_3g	float64	99999	0	0.00	

Data Cleaning

In [9]:

```
# Checking if there are any duplicate records.
data['mobile_number'].value_counts().sum()
```

Out[9]:

99999

- Since number of rows is same as distinct mobile numbers, there is no duplicate data

In [10]:

```
# mobile_number is a unique identifier
# Setting mobile_number as the index
data = data.set_index('mobile_number')
```

In [11]:

```
# Renaming columns
data = data.rename({'jun_vbc_3g' : 'vbc_3g_6', 'jul_vbc_3g' : 'vbc_3g_7', 'aug_v
bc_3g' : 'vbc_3g_8', 'sep_vbc_3g' : 'vbc_3g_9'}, axis=1)
```

In [12]:

```
#Converting columns into appropriate data types and extracting single value columns.  
# Columns with unique values < 29 are considered as categorical variables.  
# The number 30 is arrived at, by looking at the above metadata_matrix output.  
  
columns=data.columns  
change_to_cat=[]  
single_value_col=[]  
for column in columns:  
    unique_value_count=data[column].nunique()  
    if unique_value_count==1:  
        single_value_col.append(column)  
    if unique_value_count<=29 and unique_value_count!=0 and data[column].dtype in ['int','float']:  
        change_to_cat.append(column)  
print(' Columns to change to categorical data type : \n' ,pd.DataFrame(change_to_cat), '\n')
```

Columns to change to categorical data type :

```
0      circle_id  
1      loc_og_t2o_mou  
2      std_og_t2o_mou  
3      loc_ic_t2o_mou  
4      std_og_t2c_mou_6  
5      std_og_t2c_mou_7  
6      std_og_t2c_mou_8  
7      std_og_t2c_mou_9  
8      std_ic_t2o_mou_6  
9      std_ic_t2o_mou_7  
10     std_ic_t2o_mou_8  
11     std_ic_t2o_mou_9  
12     count_rech_3g_6  
13     count_rech_3g_7  
14     count_rech_3g_8  
15     count_rech_3g_9  
16     night_pck_user_6  
17     night_pck_user_7  
18     night_pck_user_8  
19     night_pck_user_9  
20     monthly_2g_6  
21     monthly_2g_7  
22     monthly_2g_8  
23     monthly_2g_9  
24     monthly_3g_6  
25     monthly_3g_7  
26     monthly_3g_8  
27     monthly_3g_9  
28     sachet_3g_6  
29     sachet_3g_7  
30     sachet_3g_8  
31     sachet_3g_9  
32     fb_user_6  
33     fb_user_7  
34     fb_user_8  
35     fb_user_9
```

In [13]:

```
# Converting all the above columns having <=29 unique values into categorical data type.
data[change_to_cat]=data[change_to_cat].astype('category')
```

In [14]:

```
# Converting *sachet* variables to categorical data type
sachet_columns = data.filter(regex='.*sachet.*', axis=1).columns.values
data[sachet_columns] = data[sachet_columns].astype('category')
```

In [15]:

```
#Changing datatype of date variables to datetime.
columns=data.columns
col_with_date=[]
import re
for column in columns:
    x = re.findall("^date", column)
    if x:
        col_with_date.append(column)
data[col_with_date].dtypes
```

Out[15]:

```
date_of_last_rech_6          object
date_of_last_rech_7          object
date_of_last_rech_8          object
date_of_last_rech_9          object
date_of_last_rech_data_6     object
date_of_last_rech_data_7     object
date_of_last_rech_data_8     object
date_of_last_rech_data_9     object
dtype: object
```

In [16]:

```
# Checking the date format
data[col_with_date].head()
```

Out[16]:

mobile_number	date_of_last_rech_6	date_of_last_rech_7	date_of_last_rech_8	date_of_last_rech
7000842753	6/21/2014	7/16/2014	8/8/2014	9/28/201
7001865778	6/29/2014	7/31/2014	8/28/2014	9/30/201
7001625959	6/17/2014	7/24/2014	8/14/2014	9/29/201
7001204172	6/28/2014	7/31/2014	8/31/2014	9/30/201
7000142493	6/26/2014	7/28/2014	8/9/2014	9/28/201

- Lets convert the above columns to datetime data type.

In [17]:

```
for col in col_with_date:  
    data[col]=pd.to_datetime(data[col], format="%m/%d/%Y")  
data[col_with_date].head()
```

Out[17]:

mobile_number	date_of_last_rech_6	date_of_last_rech_7	date_of_last_rech_8	date_of_last_rech
7000842753	2014-06-21	2014-07-16	2014-08-08	2014-09-2
7001865778	2014-06-29	2014-07-31	2014-08-28	2014-09-8
7001625959	2014-06-17	2014-07-24	2014-08-14	2014-09-2
7001204172	2014-06-28	2014-07-31	2014-08-31	2014-09-8
7000142493	2014-06-26	2014-07-28	2014-08-09	2014-09-2

Filtering High Value Customers

- Customers are High Values if their Average recharge amount of june and july is more than or equal to 70th percentile of Average recharge amount.

In [18]:

```
#Deriving Average recharge amount of June and July.  
data['Average_rech_amt_6n7']=(data['total_rech_amt_6']+data['total_rech_amt_7'])/2
```

In [19]:

```
#Filtering based HIGH VALUED CUSTOMERS based on (Average_rech_amt_6n7 >= 70th percentile of Average_rech_amt_6n7)  
data=data[(data['Average_rech_amt_6n7']>= data['Average_rech_amt_6n7'].quantile(0.7))]
```

Missing Values

In [20]:

```
#Checking for missing values.
missing_values = metadata_matrix(data)[['Datatype', 'Null_Percentage']].sort_values(by='Null_Percentage', ascending=False)
missing_values
```

Out[20]:

	Datatype	Null_Percentage
av_rech_amt_data_6	float64	62.02
count_rech_2g_6	float64	62.02
arpu_2g_6	float64	62.02
max_rech_data_6	float64	62.02
night_pck_user_6	category	62.02
date_of_last_rech_data_6	datetime64[ns]	62.02
total_rech_data_6	float64	62.02
arpu_3g_6	float64	62.02
fb_user_6	category	62.02
count_rech_3g_6	category	62.02
av_rech_amt_data_9	float64	61.81
count_rech_2g_9	float64	61.81
night_pck_user_9	category	61.81
arpu_3g_9	float64	61.81
arpu_2g_9	float64	61.81
fb_user_9	category	61.81
date_of_last_rech_data_9	datetime64[ns]	61.81
total_rech_data_9	float64	61.81
count_rech_3g_9	category	61.81
max_rech_data_9	float64	61.81
count_rech_2g_7	float64	61.14
count_rech_3g_7	category	61.14
arpu_2g_7	float64	61.14
arpu_3g_7	float64	61.14
av_rech_amt_data_7	float64	61.14
max_rech_data_7	float64	61.14
fb_user_7	category	61.14
total_rech_data_7	float64	61.14
date_of_last_rech_data_7	datetime64[ns]	61.14
night_pck_user_7	category	61.14
av_rech_amt_data_8	float64	60.83
count_rech_3g_8	category	60.83
total_rech_data_8	float64	60.83
arpu_3g_8	float64	60.83
max_rech_data_8	float64	60.83
date_of_last_rech_data_8	datetime64[ns]	60.83

	Datatype	Null_Percentage
arpu_2g_8	float64	60.83
fb_user_8	category	60.83
night_pck_user_8	category	60.83
count_rech_2g_8	float64	60.83
loc_og_t2t_mou_9	float64	5.68
ic_others_9	float64	5.68
isd_ic_mou_9	float64	5.68
og_others_9	float64	5.68
loc_og_t2f_mou_9	float64	5.68
roam_ic_mou_9	float64	5.68
loc_og_mou_9	float64	5.68
std_og_t2f_mou_9	float64	5.68
loc_og_t2m_mou_9	float64	5.68
std_og_t2m_mou_9	float64	5.68
loc_og_t2c_mou_9	float64	5.68
std_og_t2t_mou_9	float64	5.68
std_ic_t2o_mou_9	category	5.68
std_ic_mou_9	float64	5.68
spl_ic_mou_9	float64	5.68
std_ic_t2f_mou_9	float64	5.68
roam_og_mou_9	float64	5.68
std_ic_t2m_mou_9	float64	5.68
offnet_mou_9	float64	5.68
std_og_mou_9	float64	5.68
spl_og_mou_9	float64	5.68
loc_ic_t2t_mou_9	float64	5.68
onnet_mou_9	float64	5.68
loc_ic_t2m_mou_9	float64	5.68
loc_ic_t2f_mou_9	float64	5.68
std_og_t2c_mou_9	category	5.68
loc_ic_mou_9	float64	5.68
std_ic_t2t_mou_9	float64	5.68
isd_og_mou_9	float64	5.68
std_og_t2t_mou_8	float64	3.13
std_og_t2c_mou_8	category	3.13
std_og_t2f_mou_8	float64	3.13
std_og_mou_8	float64	3.13

	Datatype	Null_Percentage
roam_og_mou_8	float64	3.13
isd_og_mou_8	float64	3.13
loc_og_t2t_mou_8	float64	3.13
spl_ic_mou_8	float64	3.13
std_og_t2m_mou_8	float64	3.13
ic_others_8	float64	3.13
offnet_mou_8	float64	3.13
og_others_8	float64	3.13
isd_ic_mou_8	float64	3.13
roam_ic_mou_8	float64	3.13
spl_og_mou_8	float64	3.13
loc_og_t2f_mou_8	float64	3.13
std_ic_t2m_mou_8	float64	3.13
std_ic_t2f_mou_8	float64	3.13
std_ic_t2t_mou_8	float64	3.13
loc_og_t2c_mou_8	float64	3.13
loc_ic_mou_8	float64	3.13
onnet_mou_8	float64	3.13
loc_og_t2m_mou_8	float64	3.13
loc_ic_t2f_mou_8	float64	3.13
std_ic_t2o_mou_8	category	3.13
loc_og_mou_8	float64	3.13
loc_ic_t2m_mou_8	float64	3.13
std_ic_mou_8	float64	3.13
loc_ic_t2t_mou_8	float64	3.13
date_of_last_rech_9	datetime64[ns]	2.89
date_of_last_rech_8	datetime64[ns]	1.98
last_date_of_month_9	object	1.20
loc_og_mou_6	float64	1.05
std_ic_t2m_mou_6	float64	1.05
roam_og_mou_6	float64	1.05
std_ic_t2t_mou_6	float64	1.05
loc_ic_mou_6	float64	1.05
roam_ic_mou_6	float64	1.05
loc_ic_t2f_mou_6	float64	1.05
loc_ic_t2m_mou_6	float64	1.05
std_og_t2t_mou_6	float64	1.05

	Datatype	Null_Percentage
onnet_mou_6	float64	1.05
loc_ic_t2t_mou_6	float64	1.05
offnet_mou_6	float64	1.05
og_others_6	float64	1.05
loc_og_t2t_mou_6	float64	1.05
isd_og_mou_6	float64	1.05
std_og_t2m_mou_6	float64	1.05
loc_og_t2f_mou_6	float64	1.05
spl_ic_mou_6	float64	1.05
std_ic_mou_6	float64	1.05
isd_ic_mou_6	float64	1.05
loc_og_t2m_mou_6	float64	1.05
std_ic_t2o_mou_6	category	1.05
spl_og_mou_6	float64	1.05
ic_others_6	float64	1.05
std_ic_t2f_mou_6	float64	1.05
loc_og_t2c_mou_6	float64	1.05
std_og_mou_6	float64	1.05
std_og_t2f_mou_6	float64	1.05
std_og_t2c_mou_6	category	1.05
roam_ic_mou_7	float64	1.01
loc_og_t2c_mou_7	float64	1.01
loc_og_t2f_mou_7	float64	1.01
loc_og_t2m_mou_7	float64	1.01
loc_og_t2t_mou_7	float64	1.01
roam_og_mou_7	float64	1.01
std_ic_t2t_mou_7	float64	1.01
offnet_mou_7	float64	1.01
onnet_mou_7	float64	1.01
std_ic_t2f_mou_7	float64	1.01
std_ic_mou_7	float64	1.01
loc_ic_t2f_mou_7	float64	1.01
std_ic_t2m_mou_7	float64	1.01
loc_og_mou_7	float64	1.01
loc_ic_t2t_mou_7	float64	1.01
std_og_t2t_mou_7	float64	1.01
std_og_t2c_mou_7	category	1.01

	Datatype	Null_Percentage
std_og_mou_7	float64	1.01
isd_og_mou_7	float64	1.01
spl_og_mou_7	float64	1.01
og_others_7	float64	1.01
spl_ic_mou_7	float64	1.01
loc_ic_t2m_mou_7	float64	1.01
loc_ic_mou_7	float64	1.01
ic_others_7	float64	1.01
std_og_t2m_mou_7	float64	1.01
isd_ic_mou_7	float64	1.01
std_ic_t2o_mou_7	category	1.01
std_og_t2f_mou_7	float64	1.01
last_date_of_month_8	object	0.52
loc_og_t2o_mou	category	0.38
loc_ic_t2o_mou	category	0.38
date_of_last_rech_7	datetime64[ns]	0.38
std_og_t2o_mou	category	0.38
date_of_last_rech_6	datetime64[ns]	0.21
last_date_of_month_7	object	0.10
vol_3g_mb_6	float64	0.00
arpu_6	float64	0.00
total_rech_amt_8	int64	0.00
total_rech_amt_7	int64	0.00
total_rech_amt_6	int64	0.00
total_rech_num_9	int64	0.00
last_date_of_month_6	object	0.00
vol_3g_mb_8	float64	0.00
arpu_7	float64	0.00
arpu_8	float64	0.00
arpu_9	float64	0.00
total_og_mou_6	float64	0.00
total_og_mou_7	float64	0.00
vol_3g_mb_7	float64	0.00
max_rech_amt_9	int64	0.00
vol_2g_mb_9	float64	0.00
vol_2g_mb_8	float64	0.00
vol_2g_mb_7	float64	0.00

	Datatype	Null_Percentage
vol_2g_mb_6	float64	0.00
last_day_rch_amt_9	int64	0.00
last_day_rch_amt_8	int64	0.00
last_day_rch_amt_7	int64	0.00
last_day_rch_amt_6	int64	0.00
max_rech_amt_8	int64	0.00
max_rech_amt_7	int64	0.00
max_rech_amt_6	int64	0.00
total_rech_amt_9	int64	0.00
total_ic_mou_6	float64	0.00
total_og_mou_8	float64	0.00
vbc_3g_8	float64	0.00
total_ic_mou_7	float64	0.00
total_ic_mou_8	float64	0.00
sachet_3g_9	category	0.00
sachet_3g_7	category	0.00
vbc_3g_9	float64	0.00
vbc_3g_6	float64	0.00
vbc_3g_7	float64	0.00
aon	int64	0.00
sachet_3g_6	category	0.00
monthly_3g_8	category	0.00
monthly_3g_9	category	0.00
sachet_3g_8	category	0.00
monthly_3g_7	category	0.00
sachet_2g_9	category	0.00
sachet_2g_8	category	0.00
sachet_2g_7	category	0.00
sachet_2g_6	category	0.00
monthly_2g_9	category	0.00
monthly_2g_8	category	0.00
monthly_2g_7	category	0.00
monthly_2g_6	category	0.00
monthly_3g_6	category	0.00
circle_id	category	0.00
vol_3g_mb_9	float64	0.00
total_og_mou_9	float64	0.00

	Datatype	Null_Percentage
total_rech_num_8	int64	0.00
total_rech_num_7	int64	0.00
total_rech_num_6	int64	0.00
total_ic_mou_9	float64	0.00
Average_rech_amt_6n7	float64	0.00

In [21]:

```
# Columns with high missing values , > 50%
metadata = metadata_matrix(data)
condition = metadata['Null_Percentage'] > 50
high_missing_values = metadata[condition]
high_missing_values
```

Out[21]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_
av_rech_amt_data_6	float64	11397	18614	62.02	
count_rech_3g_6	category	11397	18614	62.02	
count_rech_2g_6	float64	11397	18614	62.02	
arpu_2g_6	float64	11397	18614	62.02	
max_rech_data_6	float64	11397	18614	62.02	
night_pck_user_6	category	11397	18614	62.02	
date_of_last_rech_data_6	datetime64[ns]	11397	18614	62.02	
total_rech_data_6	float64	11397	18614	62.02	
arpu_3g_6	float64	11397	18614	62.02	
fb_user_6	category	11397	18614	62.02	
max_rech_data_9	float64	11461	18550	61.81	
count_rech_3g_9	category	11461	18550	61.81	
fb_user_9	category	11461	18550	61.81	
total_rech_data_9	float64	11461	18550	61.81	
date_of_last_rech_data_9	datetime64[ns]	11461	18550	61.81	
av_rech_amt_data_9	float64	11461	18550	61.81	
arpu_2g_9	float64	11461	18550	61.81	
arpu_3g_9	float64	11461	18550	61.81	
night_pck_user_9	category	11461	18550	61.81	
count_rech_2g_9	float64	11461	18550	61.81	
fb_user_7	category	11662	18349	61.14	
date_of_last_rech_data_7	datetime64[ns]	11662	18349	61.14	
total_rech_data_7	float64	11662	18349	61.14	
night_pck_user_7	category	11662	18349	61.14	
max_rech_data_7	float64	11662	18349	61.14	
count_rech_2g_7	float64	11662	18349	61.14	
arpu_3g_7	float64	11662	18349	61.14	
av_rech_amt_data_7	float64	11662	18349	61.14	
arpu_2g_7	float64	11662	18349	61.14	
count_rech_3g_7	category	11662	18349	61.14	
night_pck_user_8	category	11754	18257	60.83	
fb_user_8	category	11754	18257	60.83	
arpu_2g_8	float64	11754	18257	60.83	
count_rech_2g_8	float64	11754	18257	60.83	
date_of_last_rech_data_8	datetime64[ns]	11754	18257	60.83	
av_rech_amt_data_8	float64	11754	18257	60.83	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_
arpu_3g_8	float64	11754	18257	60.83	
total_rech_data_8	float64	11754	18257	60.83	
count_rech_3g_8	category	11754	18257	60.83	
max_rech_data_8	float64	11754	18257	60.83	

In [22]:

```
# Dropping above columns with high missing values
high_missing_value_columns = high_missing_values.index
data.drop(columns=high_missing_value_columns, inplace=True)
```

In [23]:

```
# Looking at remaining columns with missing values
metadata_matrix(data)
```

Out[23]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
std_ic_t2o_mou_9	category	28307	1704	5.68	
spl_og_mou_9	float64	28307	1704	5.68	
isd_og_mou_9	float64	28307	1704	5.68	
roam_ic_mou_9	float64	28307	1704	5.68	
std_og_mou_9	float64	28307	1704	5.68	
roam_og_mou_9	float64	28307	1704	5.68	
std_ic_t2f_mou_9	float64	28307	1704	5.68	
std_og_t2c_mou_9	category	28307	1704	5.68	
loc_og_t2t_mou_9	float64	28307	1704	5.68	
std_og_t2f_mou_9	float64	28307	1704	5.68	
std_ic_mou_9	float64	28307	1704	5.68	
loc_og_t2m_mou_9	float64	28307	1704	5.68	
std_og_t2m_mou_9	float64	28307	1704	5.68	
loc_og_t2f_mou_9	float64	28307	1704	5.68	
std_og_t2t_mou_9	float64	28307	1704	5.68	
loc_ic_mou_9	float64	28307	1704	5.68	
loc_og_t2c_mou_9	float64	28307	1704	5.68	
offnet_mou_9	float64	28307	1704	5.68	
loc_og_mou_9	float64	28307	1704	5.68	
spl_ic_mou_9	float64	28307	1704	5.68	
std_ic_t2m_mou_9	float64	28307	1704	5.68	
loc_ic_t2f_mou_9	float64	28307	1704	5.68	
ic_others_9	float64	28307	1704	5.68	
loc_ic_t2m_mou_9	float64	28307	1704	5.68	
loc_ic_t2t_mou_9	float64	28307	1704	5.68	
std_ic_t2t_mou_9	float64	28307	1704	5.68	
isd_ic_mou_9	float64	28307	1704	5.68	
og_others_9	float64	28307	1704	5.68	
onnet_mou_9	float64	28307	1704	5.68	
std_og_mou_8	float64	29073	938	3.13	
std_og_t2m_mou_8	float64	29073	938	3.13	
og_others_8	float64	29073	938	3.13	
loc_ic_t2f_mou_8	float64	29073	938	3.13	
std_og_t2t_mou_8	float64	29073	938	3.13	
loc_og_mou_8	float64	29073	938	3.13	
std_ic_t2o_mou_8	category	29073	938	3.13	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
loc_ic_t2m_mou_8	float64	29073	938	3.13	
std_ic_t2m_mou_8	float64	29073	938	3.13	
std_ic_t2t_mou_8	float64	29073	938	3.13	
std_og_t2f_mou_8	float64	29073	938	3.13	
std_ic_t2f_mou_8	float64	29073	938	3.13	
spl_og_mou_8	float64	29073	938	3.13	
loc_ic_t2t_mou_8	float64	29073	938	3.13	
std_og_t2c_mou_8	category	29073	938	3.13	
isd_og_mou_8	float64	29073	938	3.13	
loc_ic_mou_8	float64	29073	938	3.13	
roam_ic_mou_8	float64	29073	938	3.13	
isd_ic_mou_8	float64	29073	938	3.13	
onnet_mou_8	float64	29073	938	3.13	
loc_og_t2c_mou_8	float64	29073	938	3.13	
spl_ic_mou_8	float64	29073	938	3.13	
loc_og_t2f_mou_8	float64	29073	938	3.13	
std_ic_mou_8	float64	29073	938	3.13	
roam_og_mou_8	float64	29073	938	3.13	
ic_others_8	float64	29073	938	3.13	
loc_og_t2m_mou_8	float64	29073	938	3.13	
loc_og_t2t_mou_8	float64	29073	938	3.13	
offnet_mou_8	float64	29073	938	3.13	
date_of_last_rech_9	datetime64[ns]	29145	866	2.89	
date_of_last_rech_8	datetime64[ns]	29417	594	1.98	
last_date_of_month_9	object	29651	360	1.20	
std_ic_mou_6	float64	29695	316	1.05	
offnet_mou_6	float64	29695	316	1.05	
std_ic_t2f_mou_6	float64	29695	316	1.05	
isd_ic_mou_6	float64	29695	316	1.05	
ic_others_6	float64	29695	316	1.05	
onnet_mou_6	float64	29695	316	1.05	
std_ic_t2m_mou_6	float64	29695	316	1.05	
loc_ic_t2t_mou_6	float64	29695	316	1.05	
loc_ic_t2m_mou_6	float64	29695	316	1.05	
loc_ic_t2f_mou_6	float64	29695	316	1.05	
loc_ic_mou_6	float64	29695	316	1.05	
std_ic_t2t_mou_6	float64	29695	316	1.05	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
og_others_6	float64	29695	316	1.05	
spl_og_mou_6	float64	29695	316	1.05	
roam_ic_mou_6	float64	29695	316	1.05	
spl_ic_mou_6	float64	29695	316	1.05	
std_og_t2t_mou_6	float64	29695	316	1.05	
loc_og_t2c_mou_6	float64	29695	316	1.05	
std_og_t2m_mou_6	float64	29695	316	1.05	
loc_og_t2f_mou_6	float64	29695	316	1.05	
std_og_t2f_mou_6	float64	29695	316	1.05	
loc_og_t2m_mou_6	float64	29695	316	1.05	
std_ic_t2o_mou_6	category	29695	316	1.05	
std_og_t2c_mou_6	category	29695	316	1.05	
std_og_mou_6	float64	29695	316	1.05	
loc_og_t2t_mou_6	float64	29695	316	1.05	
isd_og_mou_6	float64	29695	316	1.05	
roam_og_mou_6	float64	29695	316	1.05	
loc_og_mou_6	float64	29695	316	1.05	
isd_ic_mou_7	float64	29708	303	1.01	
std_ic_t2f_mou_7	float64	29708	303	1.01	
std_ic_t2m_mou_7	float64	29708	303	1.01	
std_ic_t2o_mou_7	category	29708	303	1.01	
ic_others_7	float64	29708	303	1.01	
spl_ic_mou_7	float64	29708	303	1.01	
std_ic_t2t_mou_7	float64	29708	303	1.01	
std_ic_mou_7	float64	29708	303	1.01	
loc_ic_t2f_mou_7	float64	29708	303	1.01	
og_others_7	float64	29708	303	1.01	
loc_ic_mou_7	float64	29708	303	1.01	
std_og_t2f_mou_7	float64	29708	303	1.01	
onnet_mou_7	float64	29708	303	1.01	
roam_ic_mou_7	float64	29708	303	1.01	
roam_og_mou_7	float64	29708	303	1.01	
loc_og_t2t_mou_7	float64	29708	303	1.01	
loc_og_t2m_mou_7	float64	29708	303	1.01	
loc_og_t2f_mou_7	float64	29708	303	1.01	
loc_og_t2c_mou_7	float64	29708	303	1.01	
loc_og_mou_7	float64	29708	303	1.01	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
std_og_t2t_mou_7	float64	29708	303	1.01	
std_og_t2m_mou_7	float64	29708	303	1.01	
offnet_mou_7	float64	29708	303	1.01	
std_og_t2c_mou_7	category	29708	303	1.01	
loc_ic_t2t_mou_7	float64	29708	303	1.01	
isd_og_mou_7	float64	29708	303	1.01	
spl_og_mou_7	float64	29708	303	1.01	
std_og_mou_7	float64	29708	303	1.01	
loc_ic_t2m_mou_7	float64	29708	303	1.01	
last_date_of_month_8	object	29854	157	0.52	
std_og_t2o_mou	category	29897	114	0.38	
loc_ic_t2o_mou	category	29897	114	0.38	
date_of_last_rech_7	datetime64[ns]	29897	114	0.38	
loc_og_t2o_mou	category	29897	114	0.38	
date_of_last_rech_6	datetime64[ns]	29949	62	0.21	
last_date_of_month_7	object	29980	31	0.10	
sachet_3g_6	category	30011	0	0.00	
monthly_2g_8	category	30011	0	0.00	
vol_2g_mb_8	float64	30011	0	0.00	
vol_2g_mb_9	float64	30011	0	0.00	
vol_2g_mb_6	float64	30011	0	0.00	
sachet_3g_9	category	30011	0	0.00	
sachet_3g_8	category	30011	0	0.00	
monthly_3g_9	category	30011	0	0.00	
vol_3g_mb_6	float64	30011	0	0.00	
vol_3g_mb_7	float64	30011	0	0.00	
vol_3g_mb_8	float64	30011	0	0.00	
vol_3g_mb_9	float64	30011	0	0.00	
monthly_2g_6	category	30011	0	0.00	
monthly_2g_7	category	30011	0	0.00	
monthly_2g_9	category	30011	0	0.00	
sachet_3g_7	category	30011	0	0.00	
sachet_2g_6	category	30011	0	0.00	
sachet_2g_7	category	30011	0	0.00	
sachet_2g_8	category	30011	0	0.00	
sachet_2g_9	category	30011	0	0.00	
vbc_3g_9	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
monthly_3g_8	category	30011	0	0.00	
monthly_3g_7	category	30011	0	0.00	
vbc_3g_6	float64	30011	0	0.00	
vbc_3g_7	float64	30011	0	0.00	
vbc_3g_8	float64	30011	0	0.00	
aon	int64	30011	0	0.00	
monthly_3g_6	category	30011	0	0.00	
vol_2g_mb_7	float64	30011	0	0.00	
circle_id	category	30011	0	0.00	
last_day_rch_amt_9	int64	30011	0	0.00	
last_day_rch_amt_8	int64	30011	0	0.00	
last_date_of_month_6	object	30011	0	0.00	
arpu_6	float64	30011	0	0.00	
arpu_7	float64	30011	0	0.00	
arpu_8	float64	30011	0	0.00	
arpu_9	float64	30011	0	0.00	
total_og_mou_6	float64	30011	0	0.00	
total_og_mou_7	float64	30011	0	0.00	
total_og_mou_8	float64	30011	0	0.00	
total_og_mou_9	float64	30011	0	0.00	
total_ic_mou_6	float64	30011	0	0.00	
total_ic_mou_7	float64	30011	0	0.00	
total_ic_mou_8	float64	30011	0	0.00	
total_ic_mou_9	float64	30011	0	0.00	
total_rech_num_6	int64	30011	0	0.00	
total_rech_num_7	int64	30011	0	0.00	
total_rech_num_8	int64	30011	0	0.00	
total_rech_num_9	int64	30011	0	0.00	
total_rech_amt_6	int64	30011	0	0.00	
total_rech_amt_7	int64	30011	0	0.00	
total_rech_amt_8	int64	30011	0	0.00	
total_rech_amt_9	int64	30011	0	0.00	
max_rech_amt_6	int64	30011	0	0.00	
max_rech_amt_7	int64	30011	0	0.00	
max_rech_amt_8	int64	30011	0	0.00	
max_rech_amt_9	int64	30011	0	0.00	
last_day_rch_amt_6	int64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
last_day_rch_amt_7	int64	30011	0	0.00	
Average_rech_amt_6n7	float64	30011	0	0.00	

- data contains information of 04 months - 6,7,8,9.
- For the purpose of missing value treatment, each month's revenue and usage data is not related to the other months.
- hence, missing value treatment could be performed month wise.

In [24]:

```
# Month 6
```

In [25]:

```
sixth_month_columns = []
for column in data.columns:
    x = re.search("6$", column)
    if x:
        sixth_month_columns.append(column)
# missing_values.loc[sixth_month_columns].sort_values(by='Null_Percentage', ascending=False)
metadata = metadata_matrix(data)
condition = metadata.index.isin(sixth_month_columns)
sixth_month_metadata = metadata[condition]
sixth_month_metadata
```

Out[25]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
std_ic_mou_6	float64	29695	316	1.05	
offnet_mou_6	float64	29695	316	1.05	
std_ic_t2f_mou_6	float64	29695	316	1.05	
isd_ic_mou_6	float64	29695	316	1.05	
ic_others_6	float64	29695	316	1.05	
onnet_mou_6	float64	29695	316	1.05	
std_ic_t2m_mou_6	float64	29695	316	1.05	
loc_ic_t2t_mou_6	float64	29695	316	1.05	
loc_ic_t2m_mou_6	float64	29695	316	1.05	
loc_ic_t2f_mou_6	float64	29695	316	1.05	
loc_ic_mou_6	float64	29695	316	1.05	
std_ic_t2t_mou_6	float64	29695	316	1.05	
og_others_6	float64	29695	316	1.05	
spl_og_mou_6	float64	29695	316	1.05	
roam_ic_mou_6	float64	29695	316	1.05	
spl_ic_mou_6	float64	29695	316	1.05	
std_og_t2t_mou_6	float64	29695	316	1.05	
loc_og_t2c_mou_6	float64	29695	316	1.05	
std_og_t2m_mou_6	float64	29695	316	1.05	
loc_og_t2f_mou_6	float64	29695	316	1.05	
std_og_t2f_mou_6	float64	29695	316	1.05	
loc_og_t2m_mou_6	float64	29695	316	1.05	
std_ic_t2o_mou_6	category	29695	316	1.05	
std_og_t2c_mou_6	category	29695	316	1.05	
std_og_mou_6	float64	29695	316	1.05	
loc_og_t2t_mou_6	float64	29695	316	1.05	
isd_og_mou_6	float64	29695	316	1.05	
roam_og_mou_6	float64	29695	316	1.05	
loc_og_mou_6	float64	29695	316	1.05	
date_of_last_rech_6	datetime64[ns]	29949	62	0.21	
sachet_3g_6	category	30011	0	0.00	
vol_2g_mb_6	float64	30011	0	0.00	
vol_3g_mb_6	float64	30011	0	0.00	
monthly_2g_6	category	30011	0	0.00	
sachet_2g_6	category	30011	0	0.00	
vbc_3g_6	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
monthly_3g_6	category	30011	0	0.00	
last_date_of_month_6	object	30011	0	0.00	
arpu_6	float64	30011	0	0.00	
total_og_mou_6	float64	30011	0	0.00	
total_ic_mou_6	float64	30011	0	0.00	
total_rech_num_6	int64	30011	0	0.00	
total_rech_amt_6	int64	30011	0	0.00	
max_rech_amt_6	int64	30011	0	0.00	
last_day_rch_amt_6	int64	30011	0	0.00	

- Note that all the columns with *_mou have exactly 3.94% rows with missing values.
- This is an indicator of a meaningful missing values.
- Further note that *_mou columns indicate minutes of usage, which are applicable only to customers using calling plans. It is probable that, the 3.94% customers not using calling plans.
- This could confirmed by looking at 'total_og_mou_6' and 'total_ic_mou_6' related columns where *_mou columns have missing values. If these columns are zero for a customer , then all _mou columns should be zero too.*

In [26]:

```
# columns with meaningful missing in 6th month
sixth_month_meaningful_missing_condition = sixth_month_metadata['Null_Percentage'] == 1.05
sixth_month_meaningful_missing_cols = sixth_month_metadata[sixth_month_meaningful_missing_condition].index.values
sixth_month_meaningful_missing_cols
```

Out[26]:

```
array(['std_ic_mou_6', 'offnet_mou_6', 'std_ic_t2f_mou_6', 'isd_ic_mou_6',
       'ic_others_6', 'onnet_mou_6', 'std_ic_t2m_mou_6',
       'loc_ic_t2t_mou_6', 'loc_ic_t2m_mou_6', 'loc_ic_t2f_mou_6',
       'loc_ic_mou_6', 'std_ic_t2t_mou_6', 'og_others_6', 'spl_og_mou_6',
       'roam_ic_mou_6', 'spl_ic_mou_6', 'std_og_t2t_mou_6',
       'loc_og_t2c_mou_6', 'std_og_t2m_mou_6', 'loc_og_t2f_mou_6',
       'std_og_t2f_mou_6', 'loc_og_t2m_mou_6', 'std_ic_t2o_mou_6',
       'std_og_t2c_mou_6', 'std_og_mou_6', 'loc_og_t2t_mou_6',
       'isd_og_mou_6', 'roam_og_mou_6', 'loc_og_mou_6'], dtype=object)
```

In [27]:

```
# Looking at all sixth month columns where rows of *_mou are null
condition = data[sixth_month_meaningful_missing_cols].isnull()
# data.loc[condition, sixth_month_columns]

# Rows is null for all the above columns
missing_rows = pd.Series([True]*data.shape[0], index = data.index)
for column in sixth_month_meaningful_missing_cols :
    missing_rows = missing_rows & data[column].isnull()

print('Total outgoing mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_og_mou_6'].unique()[0])
print('Total incoming mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_ic_mou_6'].unique()[0])
```

Total outgoing mou for each customer with missing *_mou data is 0.0
Total incoming mou for each customer with missing *_mou data is 0.0

- Hence, these could be imputed with 0

In [28]:

```
# Imputation
data[sixth_month_meaningful_missing_cols] = data[sixth_month_meaningful_missing_
cols].fillna(0)

metadata = metadata_matrix(data)

# Remaining Missing Values
metadata.iloc[metadata.index.isin(sixth_month_columns)]
```

Out[28]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
date_of_last_rech_6	datetime64[ns]	29949	62	0.21	
monthly_2g_6	category	30011	0	0.00	
vbc_3g_6	float64	30011	0	0.00	
max_rech_amt_6	int64	30011	0	0.00	
sachet_3g_6	category	30011	0	0.00	
sachet_2g_6	category	30011	0	0.00	
vol_2g_mb_6	float64	30011	0	0.00	
monthly_3g_6	category	30011	0	0.00	
vol_3g_mb_6	float64	30011	0	0.00	
last_day_rch_amt_6	int64	30011	0	0.00	
total_rech_amt_6	int64	30011	0	0.00	
loc_og_t2m_mou_6	float64	30011	0	0.00	
isd_og_mou_6	float64	30011	0	0.00	
std_og_mou_6	float64	30011	0	0.00	
std_og_t2c_mou_6	category	30011	0	0.00	
std_og_t2f_mou_6	float64	30011	0	0.00	
std_og_t2m_mou_6	float64	30011	0	0.00	
std_og_t2t_mou_6	float64	30011	0	0.00	
loc_og_mou_6	float64	30011	0	0.00	
loc_og_t2c_mou_6	float64	30011	0	0.00	
loc_og_t2f_mou_6	float64	30011	0	0.00	
loc_og_t2t_mou_6	float64	30011	0	0.00	
roam_og_mou_6	float64	30011	0	0.00	
roam_ic_mou_6	float64	30011	0	0.00	
offnet_mou_6	float64	30011	0	0.00	
onnet_mou_6	float64	30011	0	0.00	
arpu_6	float64	30011	0	0.00	
last_date_of_month_6	object	30011	0	0.00	
spl_og_mou_6	float64	30011	0	0.00	
og_others_6	float64	30011	0	0.00	
total_og_mou_6	float64	30011	0	0.00	
total_rech_num_6	int64	30011	0	0.00	
ic_others_6	float64	30011	0	0.00	
isd_ic_mou_6	float64	30011	0	0.00	
spl_ic_mou_6	float64	30011	0	0.00	
total_ic_mou_6	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
std_ic_mou_6	float64	30011	0	0.00	
std_ic_t2o_mou_6	category	30011	0	0.00	
std_ic_t2f_mou_6	float64	30011	0	0.00	
std_ic_t2m_mou_6	float64	30011	0	0.00	
std_ic_t2t_mou_6	float64	30011	0	0.00	
loc_ic_mou_6	float64	30011	0	0.00	
loc_ic_t2f_mou_6	float64	30011	0	0.00	
loc_ic_t2m_mou_6	float64	30011	0	0.00	
loc_ic_t2t_mou_6	float64	30011	0	0.00	

- Looks like there '1.61%' customers with missing date of last recharge. Let's look at 'recharge' related columns for such customers

In [29]:

```
# Looking at 'recharge' related 6th month columns for customers with missing 'date_of_last_rech_6'
condition = data['date_of_last_rech_6'].isnull()
data[condition].filter(regex='.*rech.*6$', axis=1).head()
```

Out[29]:

	total_rech_num_6	total_rech_amt_6	max_rech_amt_6	date_of_last_rech_6
mobile_number				
7001588448	0	0	0	NaT
7001223277	0	0	0	NaT
7000721536	0	0	0	NaT
7001490351	0	0	0	NaT
7000665415	0	0	0	NaT

In [30]:

```
data[condition].filter(regex='.*rech.*6$', axis=1).nunique()
```

Out[30]:

```
total_rech_num_6      1
total_rech_amt_6      1
max_rech_amt_6        1
date_of_last_rech_6    0
dtype: int64
```

- Notice, that the recharge related columns for customers with missing 'date_of_last_rech_6' has just one unique value. From the first few rows of the output, we see that this is 0.
- Hence, 'date_of_last_rech_6' is missing since there were no recharges made in this month.
- These are meaning missing values

In [31]:

```
# Check for missing values in 6th month variables
metadata = metadata_matrix(data)
metadata[metadata.index.isin(sixth_month_columns)]
```

Out[31]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
date_of_last_rech_6	datetime64[ns]	29949	62	0.21	
monthly_2g_6	category	30011	0	0.00	
vbc_3g_6	float64	30011	0	0.00	
max_rech_amt_6	int64	30011	0	0.00	
sachet_3g_6	category	30011	0	0.00	
sachet_2g_6	category	30011	0	0.00	
vol_2g_mb_6	float64	30011	0	0.00	
monthly_3g_6	category	30011	0	0.00	
vol_3g_mb_6	float64	30011	0	0.00	
last_day_rch_amt_6	int64	30011	0	0.00	
total_rech_amt_6	int64	30011	0	0.00	
loc_og_t2m_mou_6	float64	30011	0	0.00	
isd_og_mou_6	float64	30011	0	0.00	
std_og_mou_6	float64	30011	0	0.00	
std_og_t2c_mou_6	category	30011	0	0.00	
std_og_t2f_mou_6	float64	30011	0	0.00	
std_og_t2m_mou_6	float64	30011	0	0.00	
std_og_t2t_mou_6	float64	30011	0	0.00	
loc_og_mou_6	float64	30011	0	0.00	
loc_og_t2c_mou_6	float64	30011	0	0.00	
loc_og_t2f_mou_6	float64	30011	0	0.00	
loc_og_t2t_mou_6	float64	30011	0	0.00	
roam_og_mou_6	float64	30011	0	0.00	
roam_ic_mou_6	float64	30011	0	0.00	
offnet_mou_6	float64	30011	0	0.00	
onnet_mou_6	float64	30011	0	0.00	
arpu_6	float64	30011	0	0.00	
last_date_of_month_6	object	30011	0	0.00	
spl_og_mou_6	float64	30011	0	0.00	
og_others_6	float64	30011	0	0.00	
total_og_mou_6	float64	30011	0	0.00	
total_rech_num_6	int64	30011	0	0.00	
ic_others_6	float64	30011	0	0.00	
isd_ic_mou_6	float64	30011	0	0.00	
spl_ic_mou_6	float64	30011	0	0.00	
total_ic_mou_6	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
std_ic_mou_6	float64	30011	0	0.00	
std_ic_t2o_mou_6	category	30011	0	0.00	
std_ic_t2f_mou_6	float64	30011	0	0.00	
std_ic_t2m_mou_6	float64	30011	0	0.00	
std_ic_t2t_mou_6	float64	30011	0	0.00	
loc_ic_mou_6	float64	30011	0	0.00	
loc_ic_t2f_mou_6	float64	30011	0	0.00	
loc_ic_t2m_mou_6	float64	30011	0	0.00	
loc_ic_t2t_mou_6	float64	30011	0	0.00	

- No more Missing Values in 6th month columns

In [32]:

```
# Month : 7
seventh_month_columns = data.filter(regex='7$', axis=1).columns
seventh_month_columns
```

Out[32]:

```
Index(['last_date_of_month_7', 'arpu_7', 'onnet_mou_7', 'offnet_mou_7',
       'roam_ic_mou_7', 'roam_og_mou_7', 'loc_og_t2t_mou_7',
       'loc_og_t2m_mou_7', 'loc_og_t2f_mou_7', 'loc_og_t2c_mou_7',
       'loc_og_mou_7', 'std_og_t2t_mou_7', 'std_og_t2m_mou_7',
       'std_og_t2f_mou_7', 'std_og_t2c_mou_7', 'std_og_mou_7', 'isd_og_mou_7',
       'spl_og_mou_7', 'og_others_7', 'total_og_mou_7', 'loc_ic_t2t_mou_7',
       'loc_ic_t2m_mou_7', 'loc_ic_t2f_mou_7', 'loc_ic_mou_7',
       'std_ic_t2t_mou_7', 'std_ic_t2m_mou_7', 'std_ic_t2f_mou_7',
       'std_ic_t2o_mou_7', 'std_ic_mou_7', 'total_ic_mou_7', 'spl_ic_mou_7',
       'isd_ic_mou_7', 'ic_others_7', 'total_rech_num_7', 'total_rec_h_amt_7',
       'max_rech_amt_7', 'date_of_last_rech_7', 'last_day_rch_amt_7',
       'vol_2g_mb_7', 'vol_3g_mb_7', 'monthly_2g_7', 'sachet_2g_7',
       'monthly_3g_7', 'sachet_3g_7', 'vbc_3g_7', 'Average_rech_amt_6n7'],
      dtype='object')
```

In [33]:

```
seventh_month_metadata = metadata[metadata.index.isin(seventh_month_columns)]  
seventh_month_metadata
```

Out[33]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
loc_ic_t2t_mou_7	float64	29708	303	1.01	
og_others_7	float64	29708	303	1.01	
loc_ic_t2f_mou_7	float64	29708	303	1.01	
loc_ic_t2m_mou_7	float64	29708	303	1.01	
loc_ic_mou_7	float64	29708	303	1.01	
std_ic_t2t_mou_7	float64	29708	303	1.01	
std_ic_t2f_mou_7	float64	29708	303	1.01	
std_ic_t2o_mou_7	category	29708	303	1.01	
std_ic_mou_7	float64	29708	303	1.01	
spl_ic_mou_7	float64	29708	303	1.01	
isd_ic_mou_7	float64	29708	303	1.01	
ic_others_7	float64	29708	303	1.01	
std_ic_t2m_mou_7	float64	29708	303	1.01	
isd_og_mou_7	float64	29708	303	1.01	
spl_og_mou_7	float64	29708	303	1.01	
std_og_t2f_mou_7	float64	29708	303	1.01	
onnet_mou_7	float64	29708	303	1.01	
offnet_mou_7	float64	29708	303	1.01	
roam_ic_mou_7	float64	29708	303	1.01	
roam_og_mou_7	float64	29708	303	1.01	
loc_og_t2t_mou_7	float64	29708	303	1.01	
loc_og_t2f_mou_7	float64	29708	303	1.01	
loc_og_t2c_mou_7	float64	29708	303	1.01	
loc_og_mou_7	float64	29708	303	1.01	
std_og_t2t_mou_7	float64	29708	303	1.01	
std_og_t2m_mou_7	float64	29708	303	1.01	
loc_og_t2m_mou_7	float64	29708	303	1.01	
std_og_t2c_mou_7	category	29708	303	1.01	
std_og_mou_7	float64	29708	303	1.01	
date_of_last_rech_7	datetime64[ns]	29897	114	0.38	
last_date_of_month_7	object	29980	31	0.10	
vol_2g_mb_7	float64	30011	0	0.00	
max_rech_amt_7	int64	30011	0	0.00	
vbc_3g_7	float64	30011	0	0.00	
sachet_3g_7	category	30011	0	0.00	
total_rech_amt_7	int64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
monthly_2g_7	category	30011	0	0.00	
sachet_2g_7	category	30011	0	0.00	
last_day_rch_amt_7	int64	30011	0	0.00	
monthly_3g_7	category	30011	0	0.00	
vol_3g_mb_7	float64	30011	0	0.00	
total_rech_num_7	int64	30011	0	0.00	
arpu_7	float64	30011	0	0.00	
total_og_mou_7	float64	30011	0	0.00	
total_ic_mou_7	float64	30011	0	0.00	
Average_rech_amt_6n7	float64	30011	0	0.00	

- Note that all the columns with *_mou have exactly 3.86% rows with missing values.
- This is an indicator of a meaningful missing values.
- Further note that *_mou columns indicate minutes of usage, which are applicable only to customers using calling plans. It is probable that, the 3.86% customers not using calling plans.
- This could confirmed by looking at 'total_og_mou_7' and 'total_ic_mou_7' related columns where _mou columns have missing values. If these columns are zero for a customer , then all _mou columns should be zero too.

In [34]:

```
# columns with meaningful missing in 7th month
seventh_month_meaningful_missing_condition = seventh_month_metadata['Null_Percentage'] == 1.01
seventh_month_meaningful_missing_cols = seventh_month_metadata[seventh_month_meaningful_missing_condition].index.values
seventh_month_meaningful_missing_cols
```

Out[34]:

```
array(['loc_ic_t2t_mou_7', 'og_others_7', 'loc_ic_t2f_mou_7',
       'loc_ic_t2m_mou_7', 'loc_ic_mou_7', 'std_ic_t2t_mou_7',
       'std_ic_t2f_mou_7', 'std_ic_t2o_mou_7', 'std_ic_mou_7',
       'spl_ic_mou_7', 'isd_ic_mou_7', 'ic_others_7', 'std_ic_t2m_mo
u_7',
       'isd_og_mou_7', 'spl_og_mou_7', 'std_og_t2f_mou_7', 'onnet_mo
u_7',
       'offnet_mou_7', 'roam_ic_mou_7', 'roam_og_mou_7',
       'loc_og_t2t_mou_7', 'loc_og_t2f_mou_7', 'loc_og_t2c_mou_7',
       'loc_og_mou_7', 'std_og_t2t_mou_7', 'std_og_t2m_mou_7',
       'loc_og_t2m_mou_7', 'std_og_t2c_mou_7', 'std_og_mou_7'],
      dtype=object)
```

In [35]:

```
# Looking at all 7th month columns where rows of *_mou are null
condition = data[seventh_month_meaningful_missing_cols].isnull()

# Rows is null for all the above columns
missing_rows = pd.Series([True]*data.shape[0], index = data.index)
for column in seventh_month_meaningful_missing_cols :
    missing_rows = missing_rows & data[column].isnull()

print('Total outgoing mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_og_mou_7'].unique()[0])
print('Total incoming mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_ic_mou_7'].unique()[0])
```

```
Total outgoing mou for each customer with missing *_mou data is  0.0
Total incoming mou for each customer with missing *_mou data is  0.0
```

- Hence, these could be imputed with 0

In [36]:

```
# Imputation
data[seventh_month_meaningful_missing_cols] = data[seventh_month_meaningful_missing_cols].fillna(0)

metadata = metadata_matrix(data)

# Remaining Missing Values
metadata.iloc[metadata.index.isin(seventh_month_columns)]
```

Out[36]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
date_of_last_rech_7	datetime64[ns]	29897	114	0.38	
last_date_of_month_7	object	29980	31	0.10	
total_rech_num_7	int64	30011	0	0.00	
ic_others_7	float64	30011	0	0.00	
isd_ic_mou_7	float64	30011	0	0.00	
spl_ic_mou_7	float64	30011	0	0.00	
total_rech_amt_7	int64	30011	0	0.00	
sachet_2g_7	category	30011	0	0.00	
monthly_3g_7	category	30011	0	0.00	
sachet_3g_7	category	30011	0	0.00	
vbc_3g_7	float64	30011	0	0.00	
max_rech_amt_7	int64	30011	0	0.00	
last_day_rch_amt_7	int64	30011	0	0.00	
vol_2g_mb_7	float64	30011	0	0.00	
monthly_2g_7	category	30011	0	0.00	
vol_3g_mb_7	float64	30011	0	0.00	
loc_ic_t2f_mou_7	float64	30011	0	0.00	
total_ic_mou_7	float64	30011	0	0.00	
loc_og_t2t_mou_7	float64	30011	0	0.00	
std_og_t2m_mou_7	float64	30011	0	0.00	
std_og_t2t_mou_7	float64	30011	0	0.00	
loc_og_mou_7	float64	30011	0	0.00	
loc_og_t2c_mou_7	float64	30011	0	0.00	
loc_og_t2f_mou_7	float64	30011	0	0.00	
loc_og_t2m_mou_7	float64	30011	0	0.00	
roam_og_mou_7	float64	30011	0	0.00	
roam_ic_mou_7	float64	30011	0	0.00	
offnet_mou_7	float64	30011	0	0.00	
onnet_mou_7	float64	30011	0	0.00	
arpu_7	float64	30011	0	0.00	
std_og_t2f_mou_7	float64	30011	0	0.00	
std_og_t2c_mou_7	category	30011	0	0.00	
loc_ic_t2m_mou_7	float64	30011	0	0.00	
std_ic_mou_7	float64	30011	0	0.00	
std_ic_t2o_mou_7	category	30011	0	0.00	
std_ic_t2f_mou_7	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
std_ic_t2m_mou_7	float64	30011	0	0.00	
std_ic_t2t_mou_7	float64	30011	0	0.00	
loc_ic_mou_7	float64	30011	0	0.00	
loc_ic_t2t_mou_7	float64	30011	0	0.00	
total_og_mou_7	float64	30011	0	0.00	
og_others_7	float64	30011	0	0.00	
spl_og_mou_7	float64	30011	0	0.00	
isd_og_mou_7	float64	30011	0	0.00	
std_og_mou_7	float64	30011	0	0.00	
Average_rech_amt_6n7	float64	30011	0	0.00	

- Looks like there '1.77%' customers with missing date of last recharge. Let's look at 'recharge' related columns for such customers

In [37]:

```
# Looking at 'recharge' related 7th month columns for customers with missing 'date_of_last_rech_7'
condition = data['date_of_last_rech_7'].isnull()
data[condition].filter(regex='.*rech.*7$', axis=1).head()
```

Out[37]:

mobile_number	total_rech_num_7	total_rech_amt_7	max_rech_amt_7	date_of_last_rech_7	Ave
7000369789	0	0	0	NaT	
7001967148	0	0	0	NaT	
7000066601	0	0	0	NaT	
7001189556	0	0	0	NaT	
7002024450	0	0	0	NaT	

In [38]:

```
data[condition].filter(regex='.*rech.*7$', axis=1).nunique()
```

Out[38]:

```
total_rech_num_7      1
total_rech_amt_7      1
max_rech_amt_7        1
date_of_last_rech_7    0
Average_rech_amt_6n7  90
dtype: int64
```

- Notice, that the recharge related columns for customers with missing 'date_of_last_rech_7' has just one unique value. From the first few rows of the output, we see that this is 0.
- Hence, 'date_of_last_rech_7' is missing since there were no recharges made in this month.
- These are meaning missing values

In [39]:

```
# Month : 8
```

In [40]:

```
eighth_month_columns = data.filter(regex="8$", axis=1).columns
metadata = metadata_matrix(data)
condition = metadata.index.isin(eighth_month_columns)
eighth_month_metadata = metadata[condition]
eighth_month_metadata
```

Out[40]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
std_og_t2c_mou_8	category	29073	938	3.13	
std_og_mou_8	float64	29073	938	3.13	
isd_og_mou_8	float64	29073	938	3.13	
loc_ic_mou_8	float64	29073	938	3.13	
std_og_t2m_mou_8	float64	29073	938	3.13	
loc_ic_t2m_mou_8	float64	29073	938	3.13	
loc_og_mou_8	float64	29073	938	3.13	
std_og_t2t_mou_8	float64	29073	938	3.13	
std_og_t2f_mou_8	float64	29073	938	3.13	
loc_ic_t2f_mou_8	float64	29073	938	3.13	
loc_og_t2c_mou_8	float64	29073	938	3.13	
ic_others_8	float64	29073	938	3.13	
loc_og_t2m_mou_8	float64	29073	938	3.13	
spl_og_mou_8	float64	29073	938	3.13	
roam_ic_mou_8	float64	29073	938	3.13	
std_ic_mou_8	float64	29073	938	3.13	
spl_ic_mou_8	float64	29073	938	3.13	
std_ic_t2o_mou_8	category	29073	938	3.13	
onnet_mou_8	float64	29073	938	3.13	
loc_og_t2f_mou_8	float64	29073	938	3.13	
offnet_mou_8	float64	29073	938	3.13	
std_ic_t2f_mou_8	float64	29073	938	3.13	
og_others_8	float64	29073	938	3.13	
loc_ic_t2t_mou_8	float64	29073	938	3.13	
std_ic_t2m_mou_8	float64	29073	938	3.13	
std_ic_t2t_mou_8	float64	29073	938	3.13	
roam_og_mou_8	float64	29073	938	3.13	
isd_ic_mou_8	float64	29073	938	3.13	
loc_og_t2t_mou_8	float64	29073	938	3.13	
date_of_last_rech_8	datetime64[ns]	29417	594	1.98	
last_date_of_month_8	object	29854	157	0.52	
total_rech_num_8	int64	30011	0	0.00	
total_rech_amt_8	int64	30011	0	0.00	
last_day_rch_amt_8	int64	30011	0	0.00	
sachet_2g_8	category	30011	0	0.00	
monthly_3g_8	category	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
sachet_3g_8	category	30011	0	0.00	
vbc_3g_8	float64	30011	0	0.00	
monthly_2g_8	category	30011	0	0.00	
max_rech_amt_8	int64	30011	0	0.00	
total_ic_mou_8	float64	30011	0	0.00	
vol_2g_mb_8	float64	30011	0	0.00	
vol_3g_mb_8	float64	30011	0	0.00	
arpu_8	float64	30011	0	0.00	
total_og_mou_8	float64	30011	0	0.00	

In [41]:

```
# columns with meaningful missing in 8th month
eighth_month_meaningful_missing_condition = eighth_month_metadata['Null_Percentage'] == 3.13
eighth_month_meaningful_missing_cols = eighth_month_metadata[eighth_month_meaningful_missing_condition].index.values
eighth_month_meaningful_missing_cols
```

Out[41]:

```
array(['std_og_t2c_mou_8', 'std_og_mou_8', 'isd_og_mou_8', 'loc_ic_mou_8',
       'std_og_t2m_mou_8', 'loc_ic_t2m_mou_8', 'loc_og_mou_8',
       'std_og_t2t_mou_8', 'std_og_t2f_mou_8', 'loc_ic_t2f_mou_8',
       'loc_og_t2c_mou_8', 'ic_others_8', 'loc_og_t2m_mou_8',
       'spl_og_mou_8', 'roam_ic_mou_8', 'std_ic_mou_8', 'spl_ic_mou_8',
       'std_ic_t2o_mou_8', 'onnet_mou_8', 'loc_og_t2f_mou_8',
       'offnet_mou_8', 'std_ic_t2f_mou_8', 'og_others_8',
       'loc_ic_t2t_mou_8', 'std_ic_t2m_mou_8', 'std_ic_t2t_mou_8',
       'roam_og_mou_8', 'isd_ic_mou_8', 'loc_og_t2t_mou_8'], dtype=object)
```

In [42]:

```
# Looking at all 8th month columns where rows of *_mou are null
condition = data[eighth_month_meaningful_missing_cols].isnull()

# Rows is null for all the above columns
missing_rows = pd.Series([True]*data.shape[0], index = data.index)
for column in eighth_month_meaningful_missing_cols :
    missing_rows = missing_rows & data[column].isnull()

print('Total outgoing mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_og_mou_8'].unique()[0])
print('Total incoming mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_ic_mou_8'].unique()[0])
```

Total outgoing mou for each customer with missing *_mou data is 0.0
Total incoming mou for each customer with missing *_mou data is 0.0

In [43]:

```
# Imputation
data[eighth_month_meaningful_missing_cols] = data[eighth_month_meaningful_missing_cols].fillna(0)

metadata = metadata_matrix(data)

# Remaining Missing Values
metadata.iloc[metadata.index.isin(eighth_month_columns)]
```

Out[43]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
date_of_last_rech_8	datetime64[ns]	29417	594	1.98	
last_date_of_month_8	object	29854	157	0.52	
spl_ic_mou_8	float64	30011	0	0.00	
total_rech_num_8	int64	30011	0	0.00	
std_ic_t2f_mou_8	float64	30011	0	0.00	
ic_others_8	float64	30011	0	0.00	
std_ic_t2o_mou_8	category	30011	0	0.00	
std_ic_mou_8	float64	30011	0	0.00	
total_ic_mou_8	float64	30011	0	0.00	
isd_ic_mou_8	float64	30011	0	0.00	
sachet_2g_8	category	30011	0	0.00	
monthly_3g_8	category	30011	0	0.00	
sachet_3g_8	category	30011	0	0.00	
vbc_3g_8	float64	30011	0	0.00	
monthly_2g_8	category	30011	0	0.00	
total_rech_amt_8	int64	30011	0	0.00	
max_rech_amt_8	int64	30011	0	0.00	
last_day_rch_amt_8	int64	30011	0	0.00	
vol_2g_mb_8	float64	30011	0	0.00	
vol_3g_mb_8	float64	30011	0	0.00	
std_ic_t2m_mou_8	float64	30011	0	0.00	
loc_og_t2m_mou_8	float64	30011	0	0.00	
loc_og_t2f_mou_8	float64	30011	0	0.00	
loc_og_t2c_mou_8	float64	30011	0	0.00	
loc_og_mou_8	float64	30011	0	0.00	
std_og_t2t_mou_8	float64	30011	0	0.00	
loc_og_t2t_mou_8	float64	30011	0	0.00	
onnet_mou_8	float64	30011	0	0.00	
arpu_8	float64	30011	0	0.00	
roam_og_mou_8	float64	30011	0	0.00	
offnet_mou_8	float64	30011	0	0.00	
roam_ic_mou_8	float64	30011	0	0.00	
std_og_t2m_mou_8	float64	30011	0	0.00	
loc_ic_t2t_mou_8	float64	30011	0	0.00	
loc_ic_t2m_mou_8	float64	30011	0	0.00	
loc_ic_t2f_mou_8	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
loc_ic_mou_8	float64	30011	0	0.00	
std_ic_t2t_mou_8	float64	30011	0	0.00	
total_og_mou_8	float64	30011	0	0.00	
og_others_8	float64	30011	0	0.00	
std_og_t2f_mou_8	float64	30011	0	0.00	
std_og_t2c_mou_8	category	30011	0	0.00	
std_og_mou_8	float64	30011	0	0.00	
isd_og_mou_8	float64	30011	0	0.00	
spl_og_mou_8	float64	30011	0	0.00	

In [44]:

```
# Looking at 'recharge' related 8th month columns for customers with missing 'date_of_last_rech_8'
condition = data['date_of_last_rech_8'].isnull()
data[condition].filter(regex='.*rech.*8$', axis=1).head()
```

Out[44]:

	total_rech_num_8	total_rech_amt_8	max_rech_amt_8	date_of_last_rech_8
mobile_number				
7000340381	0	0	0	NaT
7000608224	0	0	0	NaT
7000369789	0	0	0	NaT
7000248548	0	0	0	NaT
7001967063	0	0	0	NaT

In [45]:

```
data[condition].filter(regex='.*rech.*8$', axis=1).nunique()
```

Out[45]:

```
total_rech_num_8      1
total_rech_amt_8      1
max_rech_amt_8        1
date_of_last_rech_8    0
dtype: int64
```

In [46]:

```
# Month : 9
```

In [47]:

```
ninth_month_columns = data.filter(regex="9$", axis=1).columns
metadata = metadata_matrix(data)
condition = metadata.index.isin(ninth_month_columns)
ninth_month_metadata = metadata[condition]
ninth_month_metadata
```

Out[47]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
std_og_t2c_mou_9	category	28307	1704	5.68	
spl_ic_mou_9	float64	28307	1704	5.68	
loc_og_t2m_mou_9	float64	28307	1704	5.68	
og_others_9	float64	28307	1704	5.68	
loc_og_t2c_mou_9	float64	28307	1704	5.68	
isd_ic_mou_9	float64	28307	1704	5.68	
loc_og_t2t_mou_9	float64	28307	1704	5.68	
spl_og_mou_9	float64	28307	1704	5.68	
loc_ic_t2t_mou_9	float64	28307	1704	5.68	
loc_og_mou_9	float64	28307	1704	5.68	
roam_og_mou_9	float64	28307	1704	5.68	
std_ic_mou_9	float64	28307	1704	5.68	
loc_ic_t2m_mou_9	float64	28307	1704	5.68	
roam_ic_mou_9	float64	28307	1704	5.68	
std_og_t2t_mou_9	float64	28307	1704	5.68	
offnet_mou_9	float64	28307	1704	5.68	
loc_ic_t2f_mou_9	float64	28307	1704	5.68	
std_ic_t2f_mou_9	float64	28307	1704	5.68	
isd_og_mou_9	float64	28307	1704	5.68	
std_og_mou_9	float64	28307	1704	5.68	
std_og_t2f_mou_9	float64	28307	1704	5.68	
ic_others_9	float64	28307	1704	5.68	
std_ic_t2t_mou_9	float64	28307	1704	5.68	
std_ic_t2o_mou_9	category	28307	1704	5.68	
loc_og_t2f_mou_9	float64	28307	1704	5.68	
std_og_t2m_mou_9	float64	28307	1704	5.68	
loc_ic_mou_9	float64	28307	1704	5.68	
std_ic_t2m_mou_9	float64	28307	1704	5.68	
onnet_mou_9	float64	28307	1704	5.68	
date_of_last_rech_9	datetime64[ns]	29145	866	2.89	
last_date_of_month_9	object	29651	360	1.20	
total_rech_num_9	int64	30011	0	0.00	
total_ic_mou_9	float64	30011	0	0.00	
monthly_3g_9	category	30011	0	0.00	
monthly_2g_9	category	30011	0	0.00	
sachet_2g_9	category	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
sachet_3g_9	category	30011	0	0.00	
vbc_3g_9	float64	30011	0	0.00	
vol_3g_mb_9	float64	30011	0	0.00	
total_rech_amt_9	int64	30011	0	0.00	
max_rech_amt_9	int64	30011	0	0.00	
last_day_rch_amt_9	int64	30011	0	0.00	
vol_2g_mb_9	float64	30011	0	0.00	
arpu_9	float64	30011	0	0.00	
total_og_mou_9	float64	30011	0	0.00	

In [48]:

```
# columns with meaningful missing in 9th month
ninth_month_meaningful_missing_condition = ninth_month_metadata['Null_Percentage'] == 5.68
ninth_month_meaningful_missing_cols = ninth_month_metadata[ninth_month_meaningful_missing_condition].index.values
ninth_month_meaningful_missing_cols
```

Out[48]:

```
array(['std_og_t2c_mou_9', 'spl_ic_mou_9', 'loc_og_t2m_mou_9',
       'og_others_9', 'loc_og_t2c_mou_9', 'isd_ic_mou_9',
       'loc_og_t2t_mou_9', 'spl_og_mou_9', 'loc_ic_t2t_mou_9',
       'loc_og_mou_9', 'roam_og_mou_9', 'std_ic_mou_9',
       'loc_ic_t2m_mou_9', 'roam_ic_mou_9', 'std_og_t2t_mou_9',
       'offnet_mou_9', 'loc_ic_t2f_mou_9', 'std_ic_t2f_mou_9',
       'isd_og_mou_9', 'std_og_mou_9', 'std_og_t2f_mou_9', 'ic_others_9',
       'std_ic_t2t_mou_9', 'std_ic_t2o_mou_9', 'loc_og_t2f_mou_9',
       'std_og_t2m_mou_9', 'loc_ic_mou_9', 'std_ic_t2m_mou_9',
       'onnet_mou_9'], dtype=object)
```

In [49]:

```
# Looking at all 9th month columns where rows of *_mou are null
condition = data[ninth_month_meaningful_missing_cols].isnull()

# Rows is null for all the above columns
missing_rows = pd.Series([True]*data.shape[0], index = data.index)
for column in ninth_month_meaningful_missing_cols :
    missing_rows = missing_rows & data[column].isnull()

print('Total outgoing mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_og_mou_9'].unique()[0])
print('Total incoming mou for each customer with missing *_mou data is ', data.loc[missing_rows,'total_ic_mou_9'].unique()[0])
```

```
Total outgoing mou for each customer with missing *_mou data is 0.0
Total incoming mou for each customer with missing *_mou data is 0.0
```

In [50]:

```
# Imputation
data[ninth_month_meaningful_missing_cols] = data[ninth_month_meaningful_missing_
cols].fillna(0)

metadata = metadata_matrix(data)

# Remaining Missing Values
metadata.iloc[metadata.index.isin(ninth_month_columns)]
```

Out[50]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
date_of_last_rech_9	datetime64[ns]	29145	866	2.89	
last_date_of_month_9	object	29651	360	1.20	
spl_ic_mou_9	float64	30011	0	0.00	
total_ic_mou_9	float64	30011	0	0.00	
std_ic_mou_9	float64	30011	0	0.00	
isd_ic_mou_9	float64	30011	0	0.00	
ic_others_9	float64	30011	0	0.00	
loc_ic_mou_9	float64	30011	0	0.00	
std_ic_t2t_mou_9	float64	30011	0	0.00	
std_ic_t2m_mou_9	float64	30011	0	0.00	
std_ic_t2f_mou_9	float64	30011	0	0.00	
std_ic_t2o_mou_9	category	30011	0	0.00	
total_rech_amt_9	int64	30011	0	0.00	
total_rech_num_9	int64	30011	0	0.00	
monthly_3g_9	category	30011	0	0.00	
monthly_2g_9	category	30011	0	0.00	
sachet_2g_9	category	30011	0	0.00	
sachet_3g_9	category	30011	0	0.00	
vbc_3g_9	float64	30011	0	0.00	
max_rech_amt_9	int64	30011	0	0.00	
vol_3g_mb_9	float64	30011	0	0.00	
last_day_rch_amt_9	int64	30011	0	0.00	
vol_2g_mb_9	float64	30011	0	0.00	
loc_ic_t2f_mou_9	float64	30011	0	0.00	
loc_og_t2t_mou_9	float64	30011	0	0.00	
loc_og_t2m_mou_9	float64	30011	0	0.00	
loc_og_t2f_mou_9	float64	30011	0	0.00	
loc_og_t2c_mou_9	float64	30011	0	0.00	
loc_og_mou_9	float64	30011	0	0.00	
roam_og_mou_9	float64	30011	0	0.00	
onnet_mou_9	float64	30011	0	0.00	
arpu_9	float64	30011	0	0.00	
offnet_mou_9	float64	30011	0	0.00	
roam_ic_mou_9	float64	30011	0	0.00	
std_og_t2t_mou_9	float64	30011	0	0.00	
spl_og_mou_9	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Val
og_others_9	float64	30011	0	0.00	
total_og_mou_9	float64	30011	0	0.00	
loc_ic_t2t_mou_9	float64	30011	0	0.00	
loc_ic_t2m_mou_9	float64	30011	0	0.00	
isd_og_mou_9	float64	30011	0	0.00	
std_og_t2m_mou_9	float64	30011	0	0.00	
std_og_t2f_mou_9	float64	30011	0	0.00	
std_og_t2c_mou_9	category	30011	0	0.00	
std_og_mou_9	float64	30011	0	0.00	

In [51]:

```
# Looking at 'recharge' related 9th month columns for customers with missing 'date_of_last_rech_9'
condition = data['date_of_last_rech_9'].isnull()
data[condition].filter(regex='.*rech.*9$', axis=1).head()
```

Out[51]:

	total_rech_num_9	total_rech_amt_9	max_rech_amt_9	date_of_last_rech_9
mobile_number				
7000340381	0	0	0	NaT
7000854899	0	0	0	NaT
7000369789	0	0	0	NaT
7001967063	0	0	0	NaT
7000066601	0	0	0	NaT

In [52]:

```
data[condition].filter(regex='.*rech.*9$', axis=1).nunique()
```

Out[52]:

```
total_rech_num_9      1
total_rech_amt_9      1
max_rech_amt_9        1
date_of_last_rech_9    0
dtype: int64
```

In [53]:

```
# Imputing "last_date_of_month_*
```

In [54]:

```
print('Missing Value Percentage in last_date_of_month columns : \n', 100*data.filter(regex='last_date_of_month_.*', axis=1).isnull().sum() / data.shape[0], '\n')
print('The unique values in last_date_of_month_6 : ', data['last_date_of_month_6'].unique())
print('The unique values in last_date_of_month_7 : ', data['last_date_of_month_7'].unique())
print('The unique values in last_date_of_month_8 : ', data['last_date_of_month_8'].unique())
print('The unique values in last_date_of_month_9 : ', data['last_date_of_month_9'].unique())
```

Missing Value Percentage in last_date_of_month columns :

```
last_date_of_month_6      0.000000
last_date_of_month_7      0.103295
last_date_of_month_8      0.523142
last_date_of_month_9      1.199560
dtype: float64
```

```
The unique values in last_date_of_month_6 :  ['6/30/2014']
The unique values in last_date_of_month_7 :  ['7/31/2014' nan]
The unique values in last_date_of_month_8 :  ['8/31/2014' nan]
The unique values in last_date_of_month_9 :  ['9/30/2014' nan]
```

- Last date of month is the last calender date of a particular month, it is independent of the churn data.
- Lets impute these missing values using mode.

In [55]:

```
# Imputing last_date_of_month_* values
data['last_date_of_month_7'] = data['last_date_of_month_7'].fillna(data['last_date_of_month_7'].mode()[0])
data['last_date_of_month_8'] = data['last_date_of_month_8'].fillna(data['last_date_of_month_8'].mode()[0])
data['last_date_of_month_9'] = data['last_date_of_month_9'].fillna(data['last_date_of_month_9'].mode()[0])
```

In [56]:

```
data['last_date_of_month_7'].unique()
```

Out[56]:

```
array(['7/31/2014'], dtype=object)
```

In [57]:

```
metadata = metadata_matrix(data)
metadata
```

Out[57]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
date_of_last_rech_9	datetime64[ns]	29145	866	2.89	
date_of_last_rech_8	datetime64[ns]	29417	594	1.98	
loc_og_t2o_mou	category	29897	114	0.38	
date_of_last_rech_7	datetime64[ns]	29897	114	0.38	
std_og_t2o_mou	category	29897	114	0.38	
loc_ic_t2o_mou	category	29897	114	0.38	
date_of_last_rech_6	datetime64[ns]	29949	62	0.21	
isd_ic_mou_6	float64	30011	0	0.00	
total_ic_mou_6	float64	30011	0	0.00	
total_ic_mou_7	float64	30011	0	0.00	
total_ic_mou_8	float64	30011	0	0.00	
total_ic_mou_9	float64	30011	0	0.00	
spl_ic_mou_6	float64	30011	0	0.00	
spl_ic_mou_7	float64	30011	0	0.00	
spl_ic_mou_8	float64	30011	0	0.00	
spl_ic_mou_9	float64	30011	0	0.00	
total_rech_num_6	int64	30011	0	0.00	
ic_others_9	float64	30011	0	0.00	
std_ic_mou_8	float64	30011	0	0.00	
isd_ic_mou_7	float64	30011	0	0.00	
isd_ic_mou_8	float64	30011	0	0.00	
isd_ic_mou_9	float64	30011	0	0.00	
ic_others_6	float64	30011	0	0.00	
ic_others_7	float64	30011	0	0.00	
ic_others_8	float64	30011	0	0.00	
std_ic_mou_9	float64	30011	0	0.00	
std_ic_mou_7	float64	30011	0	0.00	
total_rech_num_8	int64	30011	0	0.00	
std_ic_t2m_mou_7	float64	30011	0	0.00	
loc_ic_mou_6	float64	30011	0	0.00	
loc_ic_mou_7	float64	30011	0	0.00	
loc_ic_mou_8	float64	30011	0	0.00	
loc_ic_mou_9	float64	30011	0	0.00	
std_ic_t2t_mou_6	float64	30011	0	0.00	
std_ic_t2t_mou_7	float64	30011	0	0.00	
std_ic_t2t_mou_8	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
std_ic_t2t_mou_9	float64	30011	0	0.00	
std_ic_t2m_mou_6	float64	30011	0	0.00	
std_ic_t2m_mou_8	float64	30011	0	0.00	
std_ic_mou_6	float64	30011	0	0.00	
std_ic_t2m_mou_9	float64	30011	0	0.00	
std_ic_t2f_mou_6	float64	30011	0	0.00	
std_ic_t2f_mou_7	float64	30011	0	0.00	
std_ic_t2f_mou_8	float64	30011	0	0.00	
std_ic_t2f_mou_9	float64	30011	0	0.00	
std_ic_t2o_mou_6	category	30011	0	0.00	
std_ic_t2o_mou_7	category	30011	0	0.00	
std_ic_t2o_mou_8	category	30011	0	0.00	
std_ic_t2o_mou_9	category	30011	0	0.00	
total_rech_num_7	int64	30011	0	0.00	
circle_id	category	30011	0	0.00	
total_rech_num_9	int64	30011	0	0.00	
monthly_3g_9	category	30011	0	0.00	
monthly_2g_9	category	30011	0	0.00	
sachet_2g_6	category	30011	0	0.00	
sachet_2g_7	category	30011	0	0.00	
sachet_2g_8	category	30011	0	0.00	
sachet_2g_9	category	30011	0	0.00	
monthly_3g_6	category	30011	0	0.00	
monthly_3g_7	category	30011	0	0.00	
monthly_3g_8	category	30011	0	0.00	
sachet_3g_6	category	30011	0	0.00	
monthly_2g_7	category	30011	0	0.00	
sachet_3g_7	category	30011	0	0.00	
sachet_3g_8	category	30011	0	0.00	
sachet_3g_9	category	30011	0	0.00	
aon	int64	30011	0	0.00	
vbc_3g_8	float64	30011	0	0.00	
vbc_3g_7	float64	30011	0	0.00	
vbc_3g_6	float64	30011	0	0.00	
vbc_3g_9	float64	30011	0	0.00	
monthly_2g_8	category	30011	0	0.00	
monthly_2g_6	category	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
total_rech_amt_6	int64	30011	0	0.00	
last_day_rch_amt_7	int64	30011	0	0.00	
loc_ic_t2f_mou_9	float64	30011	0	0.00	
total_rech_amt_8	int64	30011	0	0.00	
total_rech_amt_9	int64	30011	0	0.00	
max_rech_amt_6	int64	30011	0	0.00	
max_rech_amt_7	int64	30011	0	0.00	
max_rech_amt_8	int64	30011	0	0.00	
max_rech_amt_9	int64	30011	0	0.00	
last_day_rch_amt_6	int64	30011	0	0.00	
last_day_rch_amt_8	int64	30011	0	0.00	
vol_3g_mb_9	float64	30011	0	0.00	
last_day_rch_amt_9	int64	30011	0	0.00	
vol_2g_mb_6	float64	30011	0	0.00	
vol_2g_mb_7	float64	30011	0	0.00	
vol_2g_mb_8	float64	30011	0	0.00	
vol_2g_mb_9	float64	30011	0	0.00	
vol_3g_mb_6	float64	30011	0	0.00	
vol_3g_mb_7	float64	30011	0	0.00	
vol_3g_mb_8	float64	30011	0	0.00	
total_rech_amt_7	int64	30011	0	0.00	
loc_ic_t2f_mou_7	float64	30011	0	0.00	
loc_ic_t2f_mou_8	float64	30011	0	0.00	
roam_og_mou_7	float64	30011	0	0.00	
roam_og_mou_9	float64	30011	0	0.00	
loc_og_t2t_mou_6	float64	30011	0	0.00	
loc_og_t2t_mou_7	float64	30011	0	0.00	
loc_og_t2t_mou_8	float64	30011	0	0.00	
loc_og_t2t_mou_9	float64	30011	0	0.00	
loc_og_t2m_mou_6	float64	30011	0	0.00	
loc_og_t2m_mou_7	float64	30011	0	0.00	
loc_og_t2m_mou_8	float64	30011	0	0.00	
loc_og_t2m_mou_9	float64	30011	0	0.00	
loc_og_t2f_mou_6	float64	30011	0	0.00	
loc_og_t2f_mou_7	float64	30011	0	0.00	
loc_og_t2f_mou_8	float64	30011	0	0.00	
loc_og_t2f_mou_9	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
loc_og_t2c_mou_6	float64	30011	0	0.00	
loc_og_t2c_mou_7	float64	30011	0	0.00	
loc_og_t2c_mou_8	float64	30011	0	0.00	
loc_og_t2c_mou_9	float64	30011	0	0.00	
loc_og_mou_6	float64	30011	0	0.00	
loc_og_mou_7	float64	30011	0	0.00	
roam_og_mou_8	float64	30011	0	0.00	
roam_og_mou_6	float64	30011	0	0.00	
loc_og_mou_9	float64	30011	0	0.00	
roam_ic_mou_9	float64	30011	0	0.00	
last_date_of_month_6	object	30011	0	0.00	
last_date_of_month_7	object	30011	0	0.00	
last_date_of_month_8	object	30011	0	0.00	
last_date_of_month_9	object	30011	0	0.00	
arpu_6	float64	30011	0	0.00	
arpu_7	float64	30011	0	0.00	
arpu_8	float64	30011	0	0.00	
arpu_9	float64	30011	0	0.00	
onnet_mou_6	float64	30011	0	0.00	
onnet_mou_7	float64	30011	0	0.00	
onnet_mou_8	float64	30011	0	0.00	
onnet_mou_9	float64	30011	0	0.00	
offnet_mou_6	float64	30011	0	0.00	
offnet_mou_7	float64	30011	0	0.00	
offnet_mou_8	float64	30011	0	0.00	
offnet_mou_9	float64	30011	0	0.00	
roam_ic_mou_6	float64	30011	0	0.00	
roam_ic_mou_7	float64	30011	0	0.00	
roam_ic_mou_8	float64	30011	0	0.00	
loc_og_mou_8	float64	30011	0	0.00	
std_og_t2t_mou_6	float64	30011	0	0.00	
loc_ic_t2f_mou_6	float64	30011	0	0.00	
isd_og_mou_9	float64	30011	0	0.00	
spl_og_mou_7	float64	30011	0	0.00	
spl_og_mou_8	float64	30011	0	0.00	
spl_og_mou_9	float64	30011	0	0.00	
og_others_6	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
og_others_7	float64	30011	0	0.00	
og_others_8	float64	30011	0	0.00	
og_others_9	float64	30011	0	0.00	
total_og_mou_6	float64	30011	0	0.00	
total_og_mou_7	float64	30011	0	0.00	
total_og_mou_8	float64	30011	0	0.00	
total_og_mou_9	float64	30011	0	0.00	
loc_ic_t2t_mou_6	float64	30011	0	0.00	
loc_ic_t2t_mou_7	float64	30011	0	0.00	
loc_ic_t2t_mou_8	float64	30011	0	0.00	
loc_ic_t2t_mou_9	float64	30011	0	0.00	
loc_ic_t2m_mou_6	float64	30011	0	0.00	
loc_ic_t2m_mou_7	float64	30011	0	0.00	
loc_ic_t2m_mou_8	float64	30011	0	0.00	
loc_ic_t2m_mou_9	float64	30011	0	0.00	
spl_og_mou_6	float64	30011	0	0.00	
isd_og_mou_8	float64	30011	0	0.00	
std_og_t2t_mou_7	float64	30011	0	0.00	
isd_og_mou_7	float64	30011	0	0.00	
std_og_t2t_mou_8	float64	30011	0	0.00	
std_og_t2t_mou_9	float64	30011	0	0.00	
std_og_t2m_mou_6	float64	30011	0	0.00	
std_og_t2m_mou_7	float64	30011	0	0.00	
std_og_t2m_mou_8	float64	30011	0	0.00	
std_og_t2m_mou_9	float64	30011	0	0.00	
std_og_t2f_mou_6	float64	30011	0	0.00	
std_og_t2f_mou_7	float64	30011	0	0.00	
std_og_t2f_mou_8	float64	30011	0	0.00	
std_og_t2f_mou_9	float64	30011	0	0.00	
std_og_t2c_mou_6	category	30011	0	0.00	
std_og_t2c_mou_7	category	30011	0	0.00	
std_og_t2c_mou_8	category	30011	0	0.00	
std_og_t2c_mou_9	category	30011	0	0.00	
std_og_mou_6	float64	30011	0	0.00	
std_og_mou_7	float64	30011	0	0.00	
std_og_mou_8	float64	30011	0	0.00	
std_og_mou_9	float64	30011	0	0.00	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Va
isd_og_mou_6	float64	30011	0	0.00	
Average_rech_amt_6n7	float64	30011	0	0.00	

In [58]:

```
print(data[data['date_of_last_rech_6'].isnull()][['date_of_last_rech_6','total_rech_amt_6','total_rech_num_6']].nunique())
print(data[data['date_of_last_rech_7'].isnull()][['date_of_last_rech_7','total_rech_amt_7','total_rech_num_7']].nunique())
print(data[data['date_of_last_rech_8'].isnull()][['date_of_last_rech_8','total_rech_amt_8','total_rech_num_8']].nunique())
print(data[data['date_of_last_rech_9'].isnull()][['date_of_last_rech_9','total_rech_amt_9','total_rech_num_9']].nunique())
```

```
date_of_last_rech_6      0
total_rech_amt_6        1
total_rech_num_6        1
dtype: int64
date_of_last_rech_7      0
total_rech_amt_7        1
total_rech_num_7        1
dtype: int64
date_of_last_rech_8      0
total_rech_amt_8        1
total_rech_num_8        1
dtype: int64
date_of_last_rech_9      0
total_rech_amt_9        1
total_rech_num_9        1
dtype: int64
```

In [59]:

```
print("\n",data[data['date_of_last_rech_6'].isnull()][['total_rech_amt_6','total_rech_num_6']].head())
print("\n",data[data['date_of_last_rech_7'].isnull()][['total_rech_amt_7','total_rech_num_7']].head())
print("\n",data[data['date_of_last_rech_8'].isnull()][['total_rech_amt_8','total_rech_num_8']].head())
print("\n",data[data['date_of_last_rech_9'].isnull()][['total_rech_amt_9','total_rech_num_9']].head())

mobile_number      total_rech_amt_6  total_rech_num_6
7001588448          0              0
7001223277          0              0
7000721536          0              0
7001490351          0              0
7000665415          0              0

mobile_number      total_rech_amt_7  total_rech_num_7
7000369789          0              0
7001967148          0              0
7000066601          0              0
7001189556          0              0
7002024450          0              0

mobile_number      total_rech_amt_8  total_rech_num_8
7000340381          0              0
7000608224          0              0
7000369789          0              0
7000248548          0              0
7001967063          0              0

mobile_number      total_rech_amt_9  total_rech_num_9
7000340381          0              0
7000854899          0              0
7000369789          0              0
7001967063          0              0
7000066601          0              0
```

- The columns 'date_of_last_rech' for june,july and August does not have any value because there are no recharges done by the user during those months.

Dropping columns with one unique value.

In [60]:

```
metadata=metadata_matrix(data)
singular_value_cols=metadata[metadata['Unique_Values_Count']==1].index.values
#data.loc[metadata_matrix(data)['Unique_Values_Count']==1].index
```

In [61]:

```
#Dropping singular value columns.  
data.drop(columns=singular_value_cols,inplace=True)
```

In [62]:

```
# Dropping date columns
# since they are not usage related columns and can't be used for modelling
date_columns = data.filter(regex='^date.*').columns
data.drop(columns=date_columns, inplace=True)
metadata_matrix(data)
```

Out[62]:

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Values
arpv_6	float64	30011	0	0.0	
total_ic_mou_6	float64	30011	0	0.0	
total_ic_mou_8	float64	30011	0	0.0	
total_ic_mou_9	float64	30011	0	0.0	
spl_ic_mou_6	float64	30011	0	0.0	
spl_ic_mou_7	float64	30011	0	0.0	
spl_ic_mou_8	float64	30011	0	0.0	
spl_ic_mou_9	float64	30011	0	0.0	
isd_ic_mou_6	float64	30011	0	0.0	
isd_ic_mou_7	float64	30011	0	0.0	
isd_ic_mou_8	float64	30011	0	0.0	
isd_ic_mou_9	float64	30011	0	0.0	
ic_others_6	float64	30011	0	0.0	
ic_others_7	float64	30011	0	0.0	
ic_others_8	float64	30011	0	0.0	
ic_others_9	float64	30011	0	0.0	
total_rech_num_6	int64	30011	0	0.0	
total_rech_num_7	int64	30011	0	0.0	
total_rech_num_8	int64	30011	0	0.0	
total_ic_mou_7	float64	30011	0	0.0	
std_ic_mou_9	float64	30011	0	0.0	
total_rech_amt_6	int64	30011	0	0.0	
std_ic_mou_8	float64	30011	0	0.0	
loc_ic_mou_7	float64	30011	0	0.0	
loc_ic_mou_8	float64	30011	0	0.0	
loc_ic_mou_9	float64	30011	0	0.0	
std_ic_t2t_mou_6	float64	30011	0	0.0	
std_ic_t2t_mou_7	float64	30011	0	0.0	
std_ic_t2t_mou_8	float64	30011	0	0.0	
std_ic_t2t_mou_9	float64	30011	0	0.0	
std_ic_t2m_mou_6	float64	30011	0	0.0	
std_ic_t2m_mou_7	float64	30011	0	0.0	
std_ic_t2m_mou_8	float64	30011	0	0.0	
std_ic_t2m_mou_9	float64	30011	0	0.0	
std_ic_t2f_mou_6	float64	30011	0	0.0	
std_ic_t2f_mou_7	float64	30011	0	0.0	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Values
std_ic_t2f_mou_8	float64	30011	0	0.0	
std_ic_t2f_mou_9	float64	30011	0	0.0	
std_ic_mou_6	float64	30011	0	0.0	
std_ic_mou_7	float64	30011	0	0.0	
total_rech_num_9	int64	30011	0	0.0	
total_rech_amt_7	int64	30011	0	0.0	
arpu_7	float64	30011	0	0.0	
monthly_2g_8	category	30011	0	0.0	
sachet_2g_6	category	30011	0	0.0	
sachet_2g_7	category	30011	0	0.0	
sachet_2g_8	category	30011	0	0.0	
sachet_2g_9	category	30011	0	0.0	
monthly_3g_6	category	30011	0	0.0	
monthly_3g_7	category	30011	0	0.0	
monthly_3g_8	category	30011	0	0.0	
monthly_3g_9	category	30011	0	0.0	
sachet_3g_6	category	30011	0	0.0	
sachet_3g_7	category	30011	0	0.0	
sachet_3g_8	category	30011	0	0.0	
sachet_3g_9	category	30011	0	0.0	
aon	int64	30011	0	0.0	
vbc_3g_8	float64	30011	0	0.0	
vbc_3g_7	float64	30011	0	0.0	
vbc_3g_6	float64	30011	0	0.0	
vbc_3g_9	float64	30011	0	0.0	
monthly_2g_9	category	30011	0	0.0	
monthly_2g_7	category	30011	0	0.0	
total_rech_amt_8	int64	30011	0	0.0	
monthly_2g_6	category	30011	0	0.0	
total_rech_amt_9	int64	30011	0	0.0	
max_rech_amt_6	int64	30011	0	0.0	
max_rech_amt_7	int64	30011	0	0.0	
max_rech_amt_8	int64	30011	0	0.0	
max_rech_amt_9	int64	30011	0	0.0	
last_day_rch_amt_6	int64	30011	0	0.0	
last_day_rch_amt_7	int64	30011	0	0.0	
last_day_rch_amt_8	int64	30011	0	0.0	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Values
last_day_rch_amt_9	int64	30011	0	0.0	
vol_2g_mb_6	float64	30011	0	0.0	
vol_2g_mb_7	float64	30011	0	0.0	
vol_2g_mb_8	float64	30011	0	0.0	
vol_2g_mb_9	float64	30011	0	0.0	
vol_3g_mb_6	float64	30011	0	0.0	
vol_3g_mb_7	float64	30011	0	0.0	
vol_3g_mb_8	float64	30011	0	0.0	
vol_3g_mb_9	float64	30011	0	0.0	
loc_ic_mou_6	float64	30011	0	0.0	
loc_ic_t2f_mou_9	float64	30011	0	0.0	
loc_ic_t2f_mou_8	float64	30011	0	0.0	
loc_og_t2t_mou_7	float64	30011	0	0.0	
loc_og_t2t_mou_9	float64	30011	0	0.0	
loc_og_t2m_mou_6	float64	30011	0	0.0	
loc_og_t2m_mou_7	float64	30011	0	0.0	
loc_og_t2m_mou_8	float64	30011	0	0.0	
loc_og_t2m_mou_9	float64	30011	0	0.0	
loc_og_t2f_mou_6	float64	30011	0	0.0	
loc_og_t2f_mou_7	float64	30011	0	0.0	
loc_og_t2f_mou_8	float64	30011	0	0.0	
loc_og_t2f_mou_9	float64	30011	0	0.0	
loc_og_t2c_mou_6	float64	30011	0	0.0	
loc_og_t2c_mou_7	float64	30011	0	0.0	
loc_og_t2c_mou_8	float64	30011	0	0.0	
loc_og_t2c_mou_9	float64	30011	0	0.0	
loc_og_mou_6	float64	30011	0	0.0	
loc_og_mou_7	float64	30011	0	0.0	
loc_og_mou_8	float64	30011	0	0.0	
loc_og_mou_9	float64	30011	0	0.0	
loc_og_t2t_mou_8	float64	30011	0	0.0	
loc_og_t2t_mou_6	float64	30011	0	0.0	
loc_ic_t2f_mou_7	float64	30011	0	0.0	
roam_og_mou_9	float64	30011	0	0.0	
arpu_8	float64	30011	0	0.0	
arpu_9	float64	30011	0	0.0	
onnet_mou_6	float64	30011	0	0.0	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Values
onnet_mou_7	float64	30011	0	0.0	
onnet_mou_8	float64	30011	0	0.0	
onnet_mou_9	float64	30011	0	0.0	
offnet_mou_6	float64	30011	0	0.0	
offnet_mou_7	float64	30011	0	0.0	
offnet_mou_8	float64	30011	0	0.0	
offnet_mou_9	float64	30011	0	0.0	
roam_ic_mou_6	float64	30011	0	0.0	
roam_ic_mou_7	float64	30011	0	0.0	
roam_ic_mou_8	float64	30011	0	0.0	
roam_ic_mou_9	float64	30011	0	0.0	
roam_og_mou_6	float64	30011	0	0.0	
roam_og_mou_7	float64	30011	0	0.0	
roam_og_mou_8	float64	30011	0	0.0	
std_og_t2t_mou_6	float64	30011	0	0.0	
std_og_t2t_mou_7	float64	30011	0	0.0	
std_og_t2t_mou_8	float64	30011	0	0.0	
std_og_t2t_mou_9	float64	30011	0	0.0	
og_others_6	float64	30011	0	0.0	
og_others_7	float64	30011	0	0.0	
og_others_8	float64	30011	0	0.0	
og_others_9	float64	30011	0	0.0	
total_og_mou_6	float64	30011	0	0.0	
total_og_mou_7	float64	30011	0	0.0	
total_og_mou_8	float64	30011	0	0.0	
total_og_mou_9	float64	30011	0	0.0	
loc_ic_t2t_mou_6	float64	30011	0	0.0	
loc_ic_t2t_mou_7	float64	30011	0	0.0	
loc_ic_t2t_mou_8	float64	30011	0	0.0	
loc_ic_t2t_mou_9	float64	30011	0	0.0	
loc_ic_t2m_mou_6	float64	30011	0	0.0	
loc_ic_t2m_mou_7	float64	30011	0	0.0	
loc_ic_t2m_mou_8	float64	30011	0	0.0	
loc_ic_t2m_mou_9	float64	30011	0	0.0	
loc_ic_t2f_mou_6	float64	30011	0	0.0	
spl_og_mou_9	float64	30011	0	0.0	
spl_og_mou_8	float64	30011	0	0.0	

	Datatype	Non_Null_Count	Null_Count	Null_Percentage	Unique_Values
spl_og_mou_7	float64	30011	0	0.0	
std_og_t2f_mou_9	float64	30011	0	0.0	
std_og_t2m_mou_6	float64	30011	0	0.0	
std_og_t2m_mou_7	float64	30011	0	0.0	
std_og_t2m_mou_8	float64	30011	0	0.0	
std_og_t2m_mou_9	float64	30011	0	0.0	
std_og_t2f_mou_6	float64	30011	0	0.0	
std_og_t2f_mou_7	float64	30011	0	0.0	
std_og_t2f_mou_8	float64	30011	0	0.0	
std_og_mou_6	float64	30011	0	0.0	
spl_og_mou_6	float64	30011	0	0.0	
std_og_mou_7	float64	30011	0	0.0	
std_og_mou_8	float64	30011	0	0.0	
std_og_mou_9	float64	30011	0	0.0	
isd_og_mou_6	float64	30011	0	0.0	
isd_og_mou_7	float64	30011	0	0.0	
isd_og_mou_8	float64	30011	0	0.0	
isd_og_mou_9	float64	30011	0	0.0	
Average_rech_amt_6n7	float64	30011	0	0.0	

Tagging Churn (TARGET variable)

In [63]:

```
data['Churn'] = 0
churned_customers = data.query('total_og_mou_9 == 0 & total_ic_mou_9 == 0 & vol_2g_mb_9 == 0 & vol_3g_mb_9 == 0').index
data.loc[churned_customers, 'Churn']=1
data['Churn'] = data['Churn'].astype('category')
```

In [64]:

```
# Churn proportions
data['Churn'].value_counts(normalize=True).to_frame()
```

Out[64]:

Churn

0	0.913598
1	0.086402

Dropping Churn Phase Columns

In [65]:

```
churn_phase_columns = data.filter(regex='9$').columns
data.drop(columns=churn_phase_columns, inplace=True)
print('Retained Columns')
data.columns.to_frame(index=False)
```

Retained Columns

Out[65]:

	0
0	arpu_6
1	arpu_7
2	arpu_8
3	onnet_mou_6
4	onnet_mou_7
5	onnet_mou_8
6	offnet_mou_6
7	offnet_mou_7
8	offnet_mou_8
9	roam_ic_mou_6
10	roam_ic_mou_7
11	roam_ic_mou_8
12	roam_og_mou_6
13	roam_og_mou_7
14	roam_og_mou_8
15	loc_og_t2t_mou_6
16	loc_og_t2t_mou_7
17	loc_og_t2t_mou_8
18	loc_og_t2m_mou_6
19	loc_og_t2m_mou_7
20	loc_og_t2m_mou_8
21	loc_og_t2f_mou_6
22	loc_og_t2f_mou_7
23	loc_og_t2f_mou_8
24	loc_og_t2c_mou_6
25	loc_og_t2c_mou_7
26	loc_og_t2c_mou_8
27	loc_og_mou_6
28	loc_og_mou_7
29	loc_og_mou_8
30	std_og_t2t_mou_6
31	std_og_t2t_mou_7
32	std_og_t2t_mou_8
33	std_og_t2m_mou_6
34	std_og_t2m_mou_7
35	std_og_t2m_mou_8

0

36 std_og_t2f_mou_6
37 std_og_t2f_mou_7
38 std_og_t2f_mou_8
39 std_og_mou_6
40 std_og_mou_7
41 std_og_mou_8
42 isd_og_mou_6
43 isd_og_mou_7
44 isd_og_mou_8
45 spl_og_mou_6
46 spl_og_mou_7
47 spl_og_mou_8
48 og_others_6
49 og_others_7
50 og_others_8
51 total_og_mou_6
52 total_og_mou_7
53 total_og_mou_8
54 loc_ic_t2t_mou_6
55 loc_ic_t2t_mou_7
56 loc_ic_t2t_mou_8
57 loc_ic_t2m_mou_6
58 loc_ic_t2m_mou_7
59 loc_ic_t2m_mou_8
60 loc_ic_t2f_mou_6
61 loc_ic_t2f_mou_7
62 loc_ic_t2f_mou_8
63 loc_ic_mou_6
64 loc_ic_mou_7
65 loc_ic_mou_8
66 std_ic_t2t_mou_6
67 std_ic_t2t_mou_7
68 std_ic_t2t_mou_8
69 std_ic_t2m_mou_6
70 std_ic_t2m_mou_7
71 std_ic_t2m_mou_8
72 std_ic_t2f_mou_6

0

73 std_ic_t2f_mou_7
74 std_ic_t2f_mou_8
75 std_ic_mou_6
76 std_ic_mou_7
77 std_ic_mou_8
78 total_ic_mou_6
79 total_ic_mou_7
80 total_ic_mou_8
81 spl_ic_mou_6
82 spl_ic_mou_7
83 spl_ic_mou_8
84 isd_ic_mou_6
85 isd_ic_mou_7
86 isd_ic_mou_8
87 ic_others_6
88 ic_others_7
89 ic_others_8
90 total_rech_num_6
91 total_rech_num_7
92 total_rech_num_8
93 total_rech_amt_6
94 total_rech_amt_7
95 total_rech_amt_8
96 max_rech_amt_6
97 max_rech_amt_7
98 max_rech_amt_8
99 last_day_rch_amt_6
100 last_day_rch_amt_7
101 last_day_rch_amt_8
102 vol_2g_mb_6
103 vol_2g_mb_7
104 vol_2g_mb_8
105 vol_3g_mb_6
106 vol_3g_mb_7
107 vol_3g_mb_8
108 monthly_2g_6
109 monthly_2g_7

```
0
110 monthly_2g_8
111 sachet_2g_6
112 sachet_2g_7
113 sachet_2g_8
114 monthly_3g_6
115 monthly_3g_7
116 monthly_3g_8
117 sachet_3g_6
118 sachet_3g_7
119 sachet_3g_8
120 aon
121 vbc_3g_8
122 vbc_3g_7
123 vbc_3g_6
124 Average_rech_amt_6n7
125 Churn
```

In [66]:

```
print('retained no of rows', data.shape[0])
print('retain no of columns', data.shape[1])
```

```
retained no of rows 30011
retain no of columns 126
```

Exploratory Data Analysis

Summary Statistics

In [67]:

```
data.describe()
```

Out[67]:

	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8	churn
count	30011.000000	30011.000000	30011.000000	30011.000000	30011.000000	30011.000000	30011.000000
mean	587.284404	589.135427	534.857433	296.034461	304.343206	267.600412	0.300000
std	442.722413	462.897814	492.259586	460.775592	481.780488	466.560947	0.000000
min	-2258.709000	-2014.045000	-945.808000	0.000000	0.000000	0.000000	0.000000
25%	364.161000	365.004500	289.609500	41.110000	40.950000	27.010000	0.000000
50%	495.682000	493.561000	452.091000	125.830000	125.460000	99.440000	0.000000
75%	703.922000	700.788000	671.150000	353.310000	359.925000	297.735000	0.000000
max	27731.088000	35145.834000	33543.624000	7376.710000	8157.780000	10752.560000	1.000000

- The telecom company has many users with negative average revenues in both phases. These users are likely to churn

In [68]:

```
categorical_columns = data.dtypes[data.dtypes == 'category'].index.values
print('Mode : ')
data[categorical_columns].mode().T
```

Mode :

Out[68] :

	0
monthly_2g_6	0
monthly_2g_7	0
monthly_2g_8	0
sachet_2g_6	0
sachet_2g_7	0
sachet_2g_8	0
monthly_3g_6	0
monthly_3g_7	0
monthly_3g_8	0
sachet_3g_6	0
sachet_3g_7	0
sachet_3g_8	0
Churn	0

- Most customers prefer the plans of '0' category

Univariate Analysis

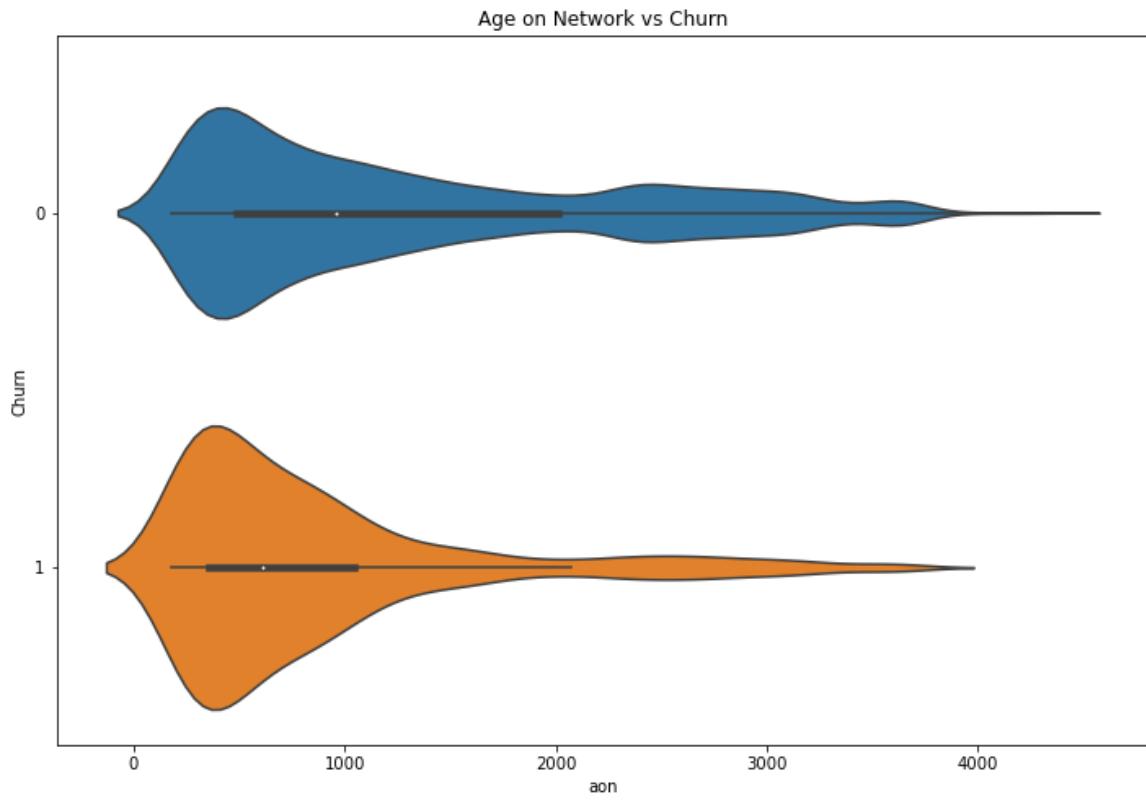
In [69]:

```
churned_customers = data[data['Churn'] == 1]
non_churned_customers = data[data['Churn'] == 0]
```

Age on Network

In [70]:

```
plt.figure(figsize=(12,8))
sns.violinplot(x='aon', y='Churn', data=data)
plt.title('Age on Network vs Churn')
plt.show()
```



- The customers with lesser 'aon' are more likely to Churn when compared to the Customers with higer 'aon'

In [71]:

```
# function for numerical variable univariate analysis
from tabulate import tabulate
def num_univariate_analysis(column_names,scale='linear') :
    # boxplot for column vs target

    fig = plt.figure(figsize=(16,8))
    ax1 = fig.add_subplot(1,3,1)
    sns.violinplot(x='Churn', y = column_names[0], data = data, ax=ax1)
    title = ''.join(column_names[0]) + ' vs Churn'
    ax1.set(title=title)
    if scale == 'log' :
        plt.yscale('log')
        ax1.set(ylabel= column_names[0] + '(Log Scale)')

    ax2 = fig.add_subplot(1,3,2)
    sns.violinplot(x='Churn', y = column_names[1], data = data, ax=ax2)
    title = ''.join(column_names[1]) + ' vs Churn'
    ax2.set(title=title)
    if scale == 'log' :
        plt.yscale('log')
        ax2.set(ylabel= column_names[1] + '(Log Scale)')

    ax3 = fig.add_subplot(1,3,3)
    sns.violinplot(x='Churn', y = column_names[2], data = data, ax=ax3)
    title = ''.join(column_names[2]) + ' vs Churn'
    ax3.set(title=title)
    if scale == 'log' :
        plt.yscale('log')
        ax3.set(ylabel= column_names[2] + '(Log Scale)')

# summary statistic

print('Customers who churned (Churn : 1)')
print(churned_customers[column_names].describe())

print('\nCustomers who did not churn (Churn : 0)')
print(non_churned_customers[column_names].describe(), '\n')
```

In [72]:

```
# function for categorical variable univariate analysis
!pip install sidetable
import sidetable
def cat_univariate_analysis(column_names, figsize=(16,4)) :

    # column vs target count plot
    fig = plt.figure(figsize=figsize)

    ax1 = fig.add_subplot(1,3,1)
    sns.countplot(x=column_names[0],hue='Churn',data=data, ax=ax1)
    title = column_names[0] + ' vs No of Churned Customers'
    ax1.set(title= title)
    ax1.legend(loc='upper right')

    ax2 = fig.add_subplot(1,3,2)
    sns.countplot(x=column_names[1],hue='Churn',data=data, ax=ax2)
    title = column_names[1] + ' vs No of Churned Customers'
    ax2.set(title= title)
    ax2.legend(loc='upper right')

    ax3 = fig.add_subplot(1,3,3)
    sns.countplot(x=column_names[2],hue='Churn',data=data, ax=ax3)
    title = column_names[2] + ' vs No of Churned Customers'
    ax3.set(title= title)
    ax3.legend(loc='upper right')

    # Percentages
    print('Customers who churned (Churn : 1)')
    print(tabulate(pd.DataFrame(churned_customers.stb.freq([column_names[0]])), headers='keys', tablefmt='psql'), '\n')
    print(tabulate(pd.DataFrame(churned_customers.stb.freq([column_names[1]])), headers='keys', tablefmt='psql'), '\n')
    print(tabulate(pd.DataFrame(churned_customers.stb.freq([column_names[2]])), headers='keys', tablefmt='psql'), '\n')

    print('\nCustomers who did not churn (Churn : 0)')
    print(tabulate(pd.DataFrame(non_churned_customers.stb.freq([column_names[0]])), headers='keys', tablefmt='psql'), '\n')
    print(tabulate(pd.DataFrame(non_churned_customers.stb.freq([column_names[1]])), headers='keys', tablefmt='psql'), '\n')
    print(tabulate(pd.DataFrame(non_churned_customers.stb.freq([column_names[2]])), headers='keys', tablefmt='psql'), '\n')
```

```
Requirement already satisfied: sidetable in /Users/UMAER/Documents/Documentation/anaconda3/lib/python3.8/site-packages (0.7.0)
Requirement already satisfied: pandas>=1.0 in /Users/UMAER/Documents/DataScience/anaconda3/lib/python3.8/site-packages (from sidetable) (1.0.5)
Requirement already satisfied: pytz>=2017.2 in /Users/UMAER/Documents/DataScience/anaconda3/lib/python3.8/site-packages (from pandas>=1.0->sidetable) (2020.1)
Requirement already satisfied: numpy>=1.13.3 in /Users/UMAER/Documents/DataScience/anaconda3/lib/python3.8/site-packages (from pandas>=1.0->sidetable) (1.18.5)
Requirement already satisfied: python-dateutil>=2.6.1 in /Users/UMAER/Documents/DataScience/anaconda3/lib/python3.8/site-packages (from pandas>=1.0->sidetable) (2.8.1)
Requirement already satisfied: six>=1.5 in /Users/UMAER/Documents/DataScience/anaconda3/lib/python3.8/site-packages (from python-dateutil>=2.6.1->pandas>=1.0->sidetable) (1.15.0)
```

arpu_6, arpu_7 , arpu_8

In [73]:

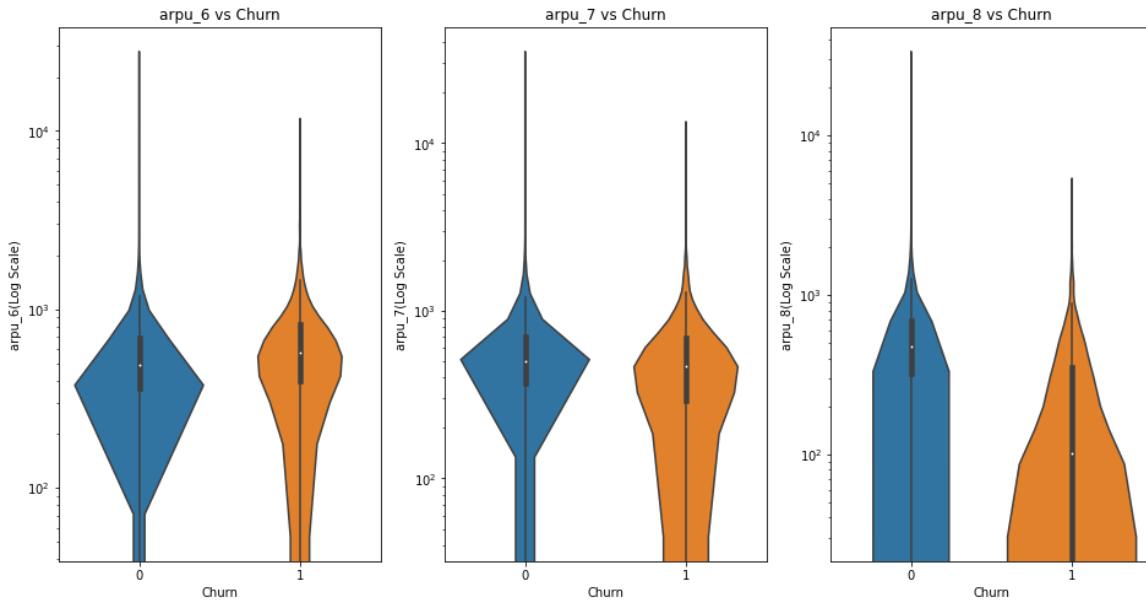
```
columns = ['arpu_6', 'arpu_7', 'arpu_8']
num_univariate_analysis(columns, 'log')
```

Customers who churned (Churn : 1)

	arpu_6	arpu_7	arpu_8
count	2593.000000	2593.000000	2593.000000
mean	678.716970	550.511946	243.063343
std	551.792864	517.241221	378.843531
min	-209.465000	-158.963000	-37.887000
25%	396.507000	289.641000	0.000000
50%	573.396000	464.674000	101.894000
75%	819.460000	691.588000	351.028000
max	11505.508000	13224.119000	5228.826000

Customers who did not churn (Churn : 0)

	arpu_6	arpu_7	arpu_8
count	27418.000000	27418.000000	27418.000000
mean	578.637360	592.788162	562.453248
std	429.988265	457.265996	492.802655
min	-2258.709000	-2014.045000	-945.808000
25%	362.218000	369.610500	319.118500
50%	489.324000	496.182500	471.024000
75%	690.891750	701.418000	690.921000
max	27731.088000	35145.834000	33543.624000



- We can understand from the above plots that revenue generated by the Customers who are about to churn is very unstable.
- The Customers whose arpu decreases in 7th month are more likely to churn when compared to ones with increase in arpu.

total_og_mou_6, total_og_mou_7, total_og_mou_8

In [74]:

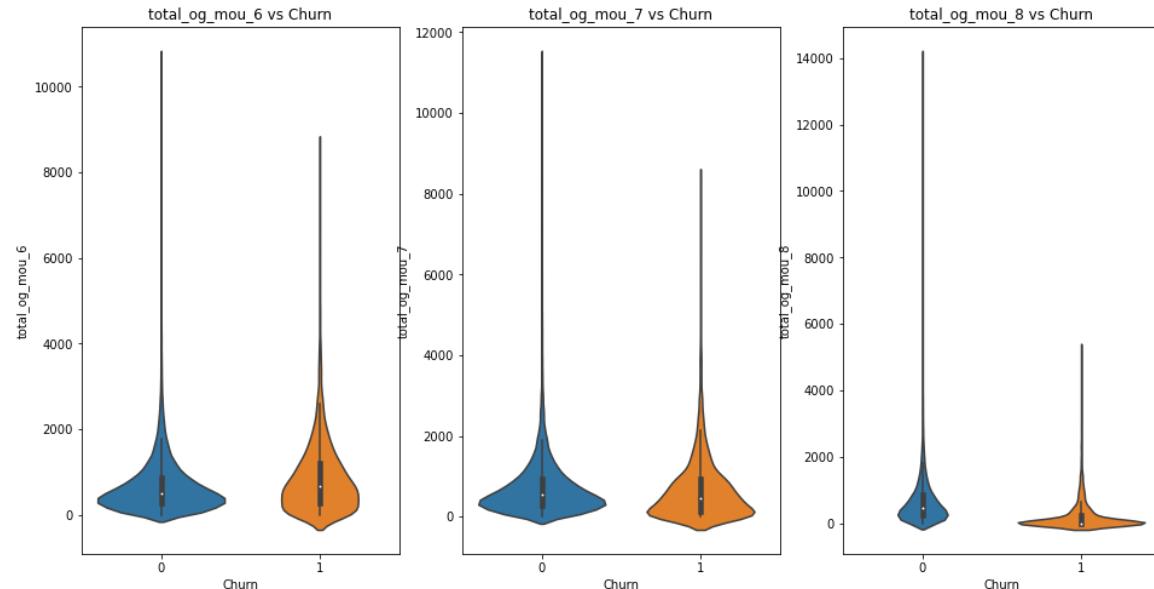
```
columns = ['total_og_mou_6', 'total_og_mou_7', 'total_og_mou_8']
num_univariate_analysis(columns)
```

Customers who churned (Churn : 1)

	total_og_mou_6	total_og_mou_7	total_og_mou_8
count	2593.000000	2593.000000	2593.000000
mean	867.961342	677.868909	225.083741
std	852.697688	786.961399	471.672718
min	0.000000	0.000000	0.000000
25%	277.880000	110.090000	0.000000
50%	658.360000	466.910000	0.000000
75%	1209.040000	926.760000	255.810000
max	8488.360000	8285.640000	5206.210000

Customers who did not churn (Churn : 0)

	total_og_mou_6	total_og_mou_7	total_og_mou_8
count	27418.000000	27418.000000	27418.000000
mean	669.554896	712.080684	661.480046
std	636.531612	674.580516	691.079113
min	0.000000	0.000000	0.000000
25%	265.682500	284.500000	227.970000
50%	500.410000	529.935000	470.475000
75%	872.070000	931.197500	866.045000
max	10674.030000	11365.310000	14043.060000



- The Customers with high total_og_mou in 6th month and lower total_og_mou in 7th month are more likely to churn compared to the rest.

'total_ic_mou_6', 'total_ic_mou_7', 'total_ic_mou_8'

In [75]:

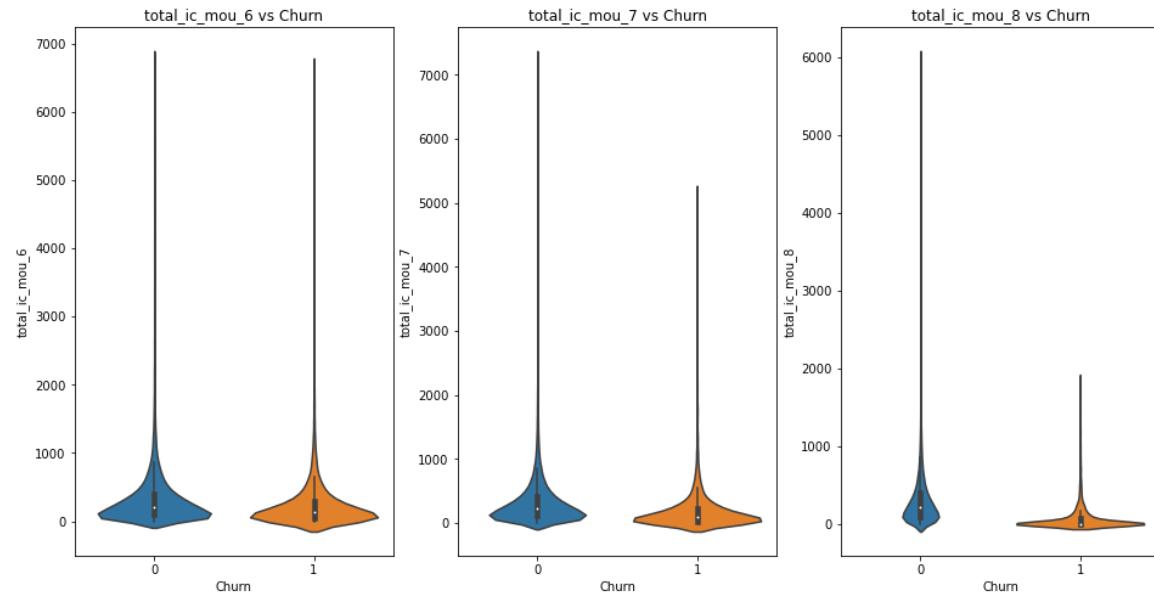
```
columns = ['total_ic_mou_6', 'total_ic_mou_7', 'total_ic_mou_8']
num_univariate_analysis(columns)
```

Customers who churned (Churn : 1)

	total_ic_mou_6	total_ic_mou_7	total_ic_mou_8
count	2593.000000	2593.000000	2593.000000
mean	241.954404	193.341076	68.807042
std	360.836586	318.183813	154.450340
min	0.000000	0.000000	0.000000
25%	49.460000	27.890000	0.000000
50%	137.330000	99.980000	0.000000
75%	289.510000	235.740000	70.290000
max	6633.180000	5137.560000	1859.280000

Customers who did not churn (Churn : 0)

	total_ic_mou_6	total_ic_mou_7	total_ic_mou_8
count	27418.000000	27418.000000	27418.000000
mean	313.712052	326.369333	316.858595
std	360.580253	372.112086	366.818717
min	0.000000	0.000000	0.000000
25%	94.460000	107.802500	98.265000
50%	212.160000	222.290000	212.360000
75%	401.602500	410.182500	402.270000
max	6798.640000	7279.080000	5990.710000



- The Customers with decrease in rate of total_ic_mou in 7th month are more likely to churn, compared to the rest.

vol_2g_mb_6, vol_2g_mb_7, vol_2g_mb_8

In [76]:

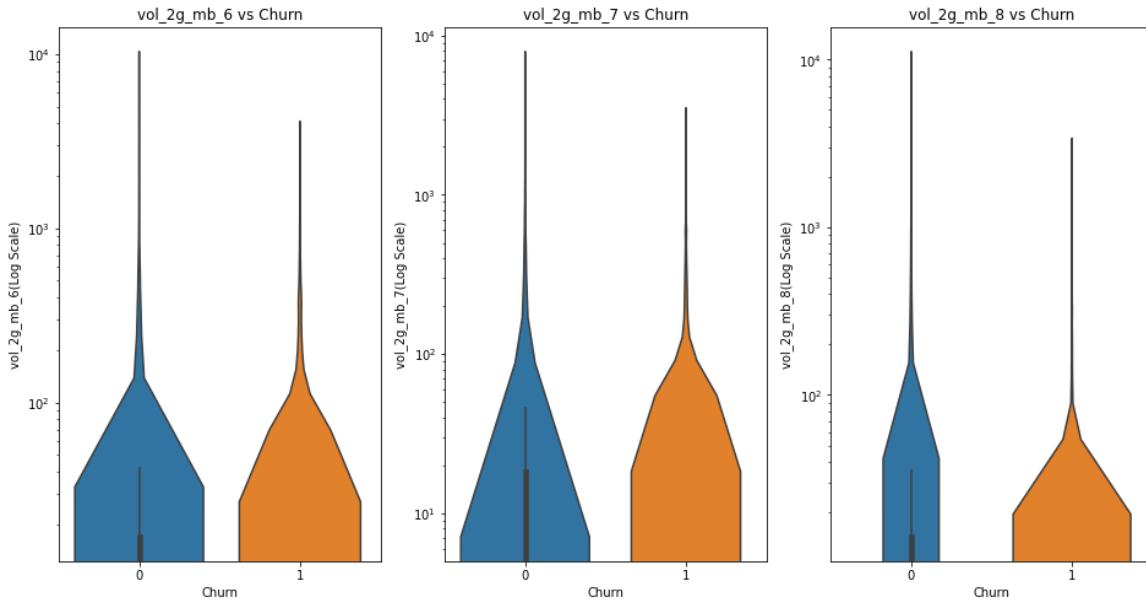
```
columns = ['vol_2g_mb_6', 'vol_2g_mb_7', 'vol_2g_mb_8']
num_univariate_analysis(columns, 'log')
```

Customers who churned (Churn : 1)

	vol_2g_mb_6	vol_2g_mb_7	vol_2g_mb_8
count	2593.000000	2593.000000	2593.000000
mean	60.775588	49.054393	15.283185
std	243.084276	219.485813	120.975111
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	4017.160000	3430.730000	3349.190000

Customers who did not churn (Churn : 0)

	vol_2g_mb_6	vol_2g_mb_7	vol_2g_mb_8
count	27418.000000	27418.000000	27418.000000
mean	80.569210	80.925060	74.309036
std	280.420463	285.265125	277.889339
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	16.937500	18.267500	14.245000
max	10285.900000	7873.550000	11117.610000



- Customers with stable usage of 2g volumes throughout 6 and 7 months are less likely to churn.
- Customers with fall in consumption of 2g volumes in 7th month are more likely to Churn.

vol_3g_mb_6, vol_3g_mb_7, vol_3g_mb_8, monthly_3g_6

In [77]:

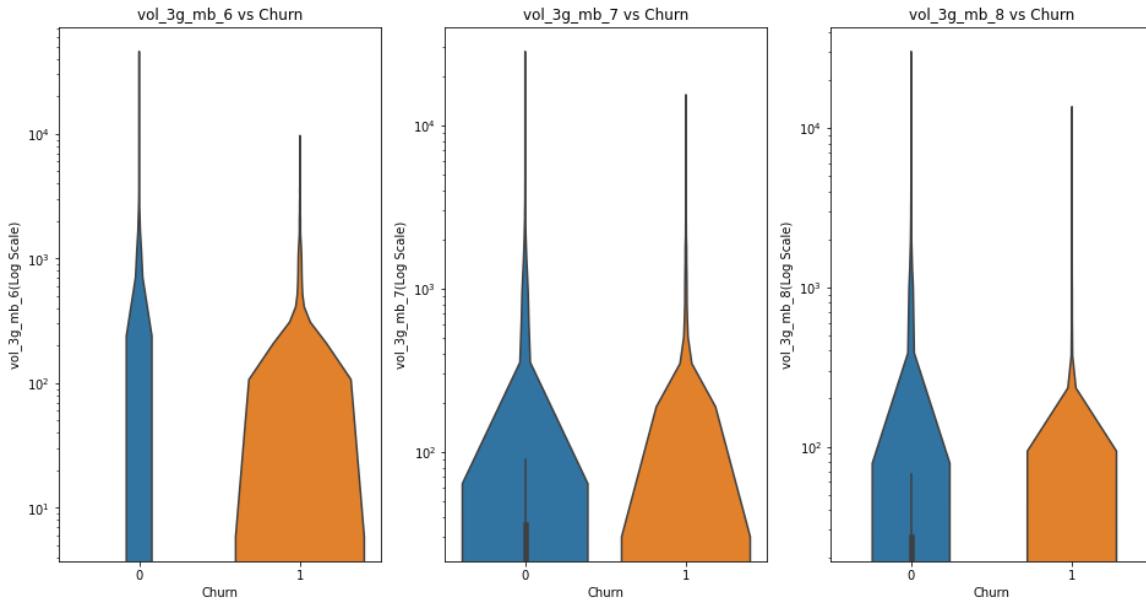
```
columns = ['vol_3g_mb_6', 'vol_3g_mb_7', 'vol_3g_mb_8', 'monthly_3g_6']
num_univariate_analysis(columns, 'log')
```

Customers who churned (Churn : 1)

	vol_3g_mb_6	vol_3g_mb_7	vol_3g_mb_8
count	2593.000000	2593.000000	2593.000000
mean	188.395461	157.714254	56.776880
std	715.327843	690.773561	446.532769
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	9400.120000	15115.510000	13440.720000

Customers who did not churn (Churn : 0)

	vol_3g_mb_6	vol_3g_mb_7	vol_3g_mb_8
count	27418.000000	27418.000000	27418.000000
mean	265.012522	289.478375	290.016390
std	878.846885	868.808831	885.821105
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	35.855000	27.120000
max	45735.400000	28144.120000	30036.060000



- Customers with stable usage of 3g volumes throughout 6 and 7 months are less likely to churn.
- Customers with fall in consumption of 3g volumes in 7th month are more likely to Churn.

monthly_2g_6, monthly_2g_7, monthly_2g_8

In [78]:

```
columns = ['monthly_2g_6', 'monthly_2g_7', 'monthly_2g_8']
cat_univariate_analysis(columns)
```

Customers who churned (Churn : 1)

monthly_2g_6	count	percent	cumulative_count
cumulative_percent			
0	0	2454	2454
94.6394	1	126	2580
99.4987	2	11	2591
99.9229	4	2	2593
100			

monthly_2g_7	count	percent	cumulative_count
cumulative_percent			
0	0	2477	2477
95.5264	1	104	2581
99.5372	2	12	2593
100			

monthly_2g_8	count	percent	cumulative_count
cumulative_percent			
0	0	2555	2555
98.5345	1	37	2592
99.9614	2	1	2593
100			

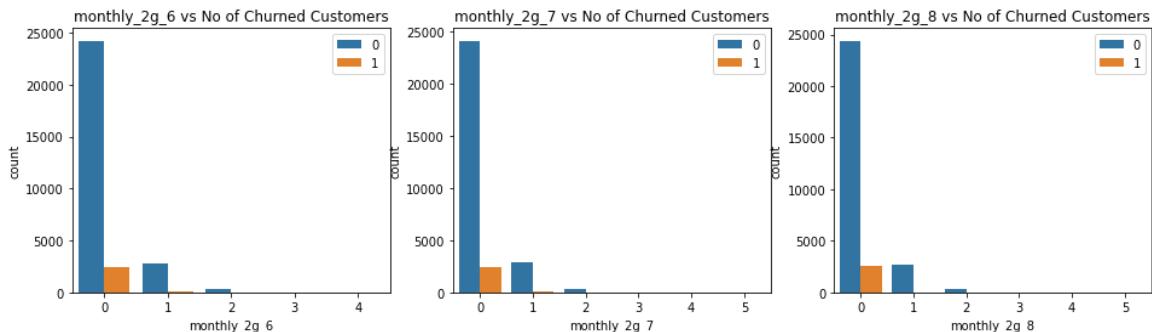
Customers who did not churn (Churn : 0)

monthly_2g_6	count	percent	cumulative_count
cumulative_percent			
0	0	24228	24228
88.3653	1	2825	27053
98.6688	2	334	27387

99.8869					
3	3	26	0.0948282		27413
99.9818					
4	4	5	0.0182362		27418
100					

+-----+-----+-----+-----+					
monthly_2g_7 count percent cumulative_count					
cumulative_percent					
-----+-----+-----+-----+					
0 0 24079 87.8219					24079
87.8219					
1 1 2909 10.6098					26988
98.4317					
2 2 394 1.43701					27382
99.8687					
3 3 29 0.10577					27411
99.9745					
4 4 5 0.0182362					27416
99.9927					
5 5 2 0.00729448					27418
100					

+-----+-----+-----+-----+					
monthly_2g_8 count percent cumulative_count					
cumulative_percent					
-----+-----+-----+-----+					
0 0 24383 88.9306					24383
88.9306					
1 1 2724 9.93508					27107
98.8657					
2 2 282 1.02852					27389
99.8942					
3 3 22 0.0802393					27411
99.9745					
4 4 5 0.0182362					27416
99.9927					
5 5 2 0.00729448					27418
100					



monthly_3g_6, monthly_3g_7, monthly_3g_8

In [79]:

```
columns = ['monthly_3g_6', 'monthly_3g_7', 'monthly_3g_8']
cat_univariate_analysis(columns)
```

Customers who churned (Churn : 1)

monthly_3g_6	count	percent	cumulative_count
cumulative_percent			
0	0	90.7057	2352
90.7057			
1	170	6.55611	2522
97.2619			
2	49	1.8897	2571
99.1516			
3	13	0.50135	2584
99.6529			
4	4	0.154261	2588
99.8072			
5	4	0.154261	2592
99.9614			
6	1	0.0385654	2593
100			

monthly_3g_7	count	percent	cumulative_count
cumulative_percent			
0	0	92.5183	2399
92.5183			
1	136	5.24489	2535
97.7632			
2	48	1.85114	2583
99.6143			
3	9	0.347088	2592
99.9614			
4	1	0.0385654	2593
100			

monthly_3g_8	count	percent	cumulative_count
cumulative_percent			
0	0	97.339	2524
97.339			
1	56	2.15966	2580
99.4987			
2	8	0.308523	2588
99.8072			
3	4	0.154261	2592
99.9614			
4	1	0.0385654	2593
100			

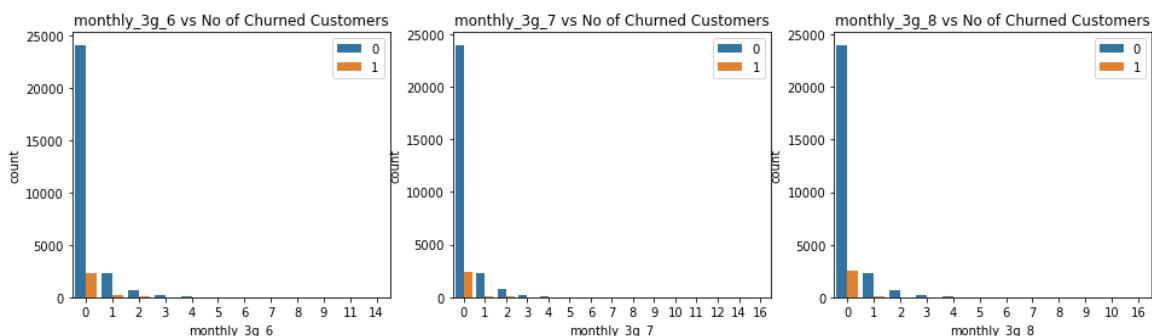
Customers who did not churn (Churn : 0)

cumulative_percent	monthly_3g_6	count	percent	cumulative_count
87.8255	0	24080	87.8255	24080
96.4731	1	2371	8.6476	26451
98.8365	2	648	2.36341	27099
99.5441	3	194	0.707564	27293
99.7994	4	70	0.255307	27363
99.9015	5	28	0.102123	27391
99.938	6	10	0.0364724	27401
99.9708	7	9	0.0328252	27410
99.9818	8	3	0.0109417	27413
99.9891	9	2	0.00729448	27415
99.9964	10	2	0.00729448	27417
100	11	1	0.00364724	27418

cumulative_percent	monthly_3g_7	count	percent	cumulative_count
87.3951	0	23962	87.3951	23962
95.8932	1	2330	8.49807	26292
98.7162	2	774	2.82296	27066
99.4383	3	198	0.722153	27264
99.6863	4	68	0.248012	27332
99.8249	5	38	0.138595	27370
99.9088	6	23	0.0838865	27393
99.9453	7	10	0.0364724	27403
99.9635	8	5	0.0182362	27408
99.9912	9	4	0.014589	27412

99.9781					
10	11	2	0.00729448		27414
99.9854					
11	16	1	0.00364724		27415
99.9891					
12	14	1	0.00364724		27416
99.9927					
13	12	1	0.00364724		27417
99.9964					
14	10	1	0.00364724		27418
100					

monthly_3g_8	count	percent	cumulative_count
cumulative_percent			
0	0	24002	24002
87.541			
1	1	2347	26349
96.1011			
2	2	728	27077
98.7563			
3	3	193	27270
99.4602			
4	4	86	27356
99.7739			
5	5	30	27386
99.8833			
6	6	14	27400
99.9343			
7	7	9	27409
99.9672			
8	9	3	27412
99.9781			
9	8	3	27415
99.9891			
10	10	2	0.00729448
99.9964			
11	16	1	0.00364724
100			



sachet_3g_6, sachet_3g_7, sachet_3g_8

In [1]:

```
columns = ['sachet_3g_6', 'sachet_3g_7', 'sachet_3g_8']
print(data[columns].dtypes)
cat_univariate_analysis(columns)
```

```
-----
NameError                                 Traceback (most recent call
1 last)
<ipython-input-1-43696c0d750b> in <module>
    1 columns = ['sachet_3g_6', 'sachet_3g_7', 'sachet_3g_8']
----> 2 print(data[columns].dtypes)
    3 cat_univariate_analysis(columns)

NameError: name 'data' is not defined
```

aug_vbc_3g, jul_vbc_3g, jun_vbc_3g

In [81]:

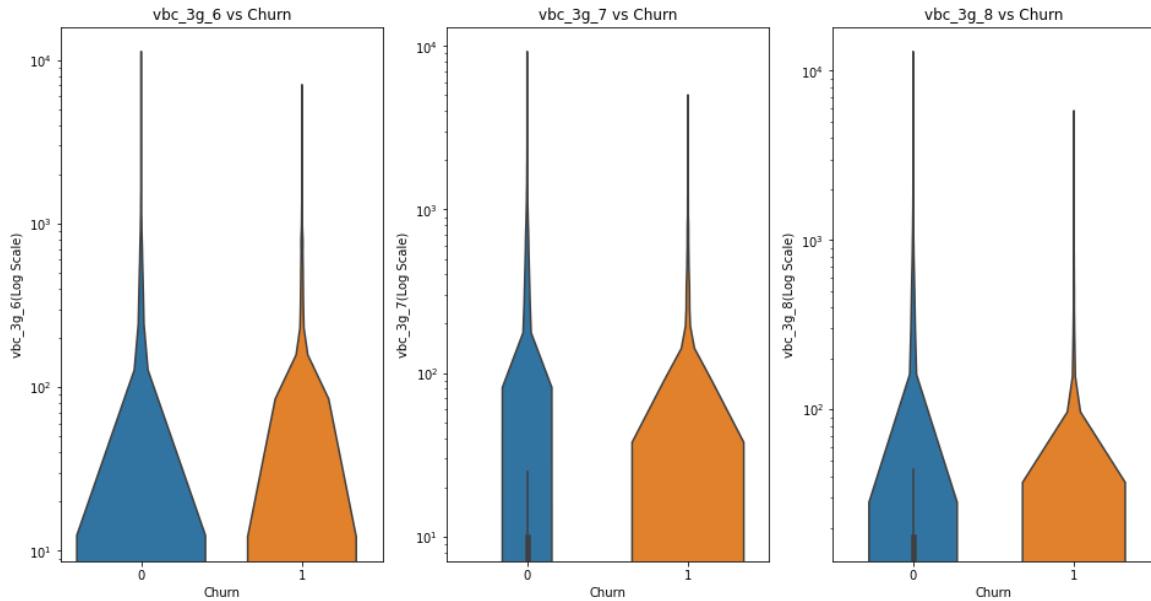
```
columns = [ 'vbc_3g_6', 'vbc_3g_7', 'vbc_3g_8' ]
num_univariate_analysis(columns, 'log')
```

Customers who churned (Churn : 1)

	vbc_3g_6	vbc_3g_7	vbc_3g_8
count	2593.000000	2593.000000	2593.000000
mean	81.564601	71.143880	32.610659
std	320.898511	284.882601	197.998246
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	6931.810000	4908.270000	5738.740000

Customers who did not churn (Churn : 0)

	vbc_3g_6	vbc_3g_7	vbc_3g_8
count	27418.000000	27418.000000	27418.000000
mean	125.124167	141.178182	138.597023
std	395.413666	417.292310	402.761779
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	9.940000	17.675000
max	11166.210000	9165.600000	12916.220000



Bivariate Analysis

In [96]:

```
data.head()
```

Out[96]:

mobile_number	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8	offne
7000701601	1069.180	1349.850	3171.480	57.84	54.68	52.29	
7001524846	378.721	492.223	137.362	413.69	351.03	35.08	
7002191713	492.846	205.671	593.260	501.76	108.39	534.24	
7000875565	430.975	299.869	187.894	50.51	74.01	70.61	
7000187447	690.008	18.980	25.499	1185.91	9.28	7.79	

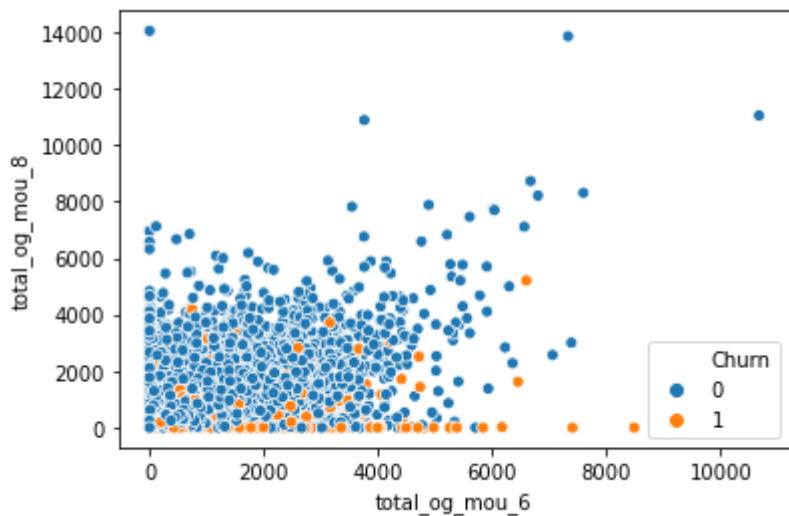
'total_og_mou_6' vs 'total_og_mou_8' with respect to Churn.

In [123]:

```
sns.scatterplot(x=data['total_og_mou_6'], y=data['total_og_mou_8'], hue=data['Churn'])
```

Out[123]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc15cc7190>
```



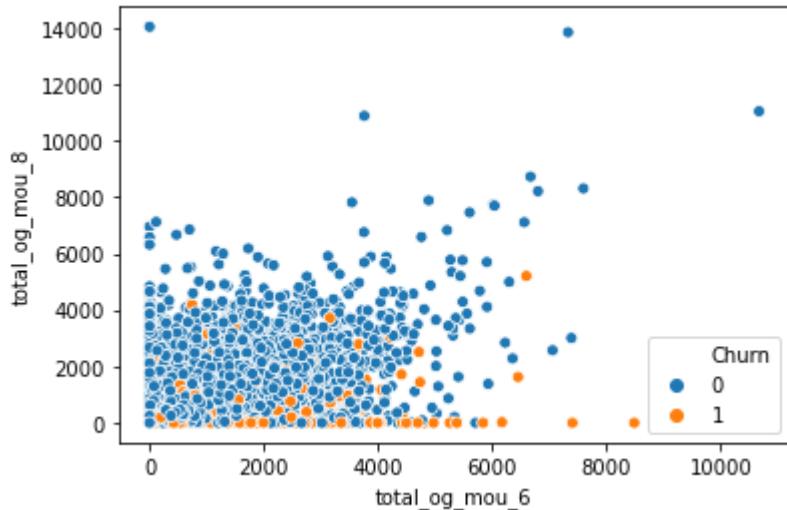
'total_og_mou_7' vs 'total_og_mou_8' with respect to Churn.

In [122]:

```
sns.scatterplot(x=data['total_og_mou_6'],y=data['total_og_mou_8'],hue=data['Churn'])
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fffc1a6f59a0>
```



- The customers with lower total_og_mou in 6th and 8th months are more likely to Churn compared to the ones with higher total_og_mou.

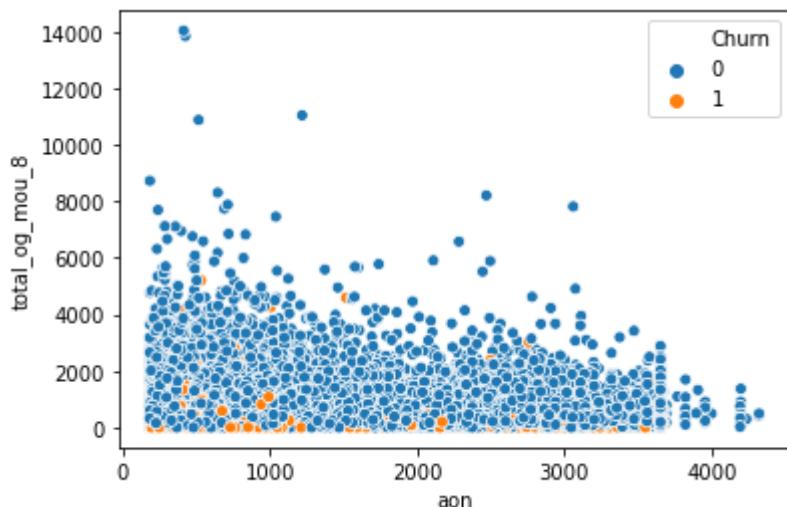
'aon' vs 'total_og_mou_8' with respect to Churn.

In [119]:

```
sns.scatterplot(x=data['aon'],y=data['total_og_mou_8'],hue=data['Churn'])
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fffc128bd790>
```



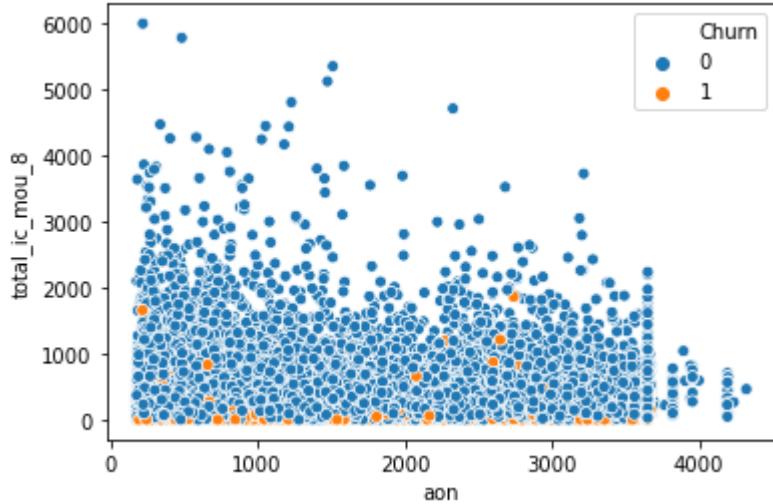
- The customers with lesser total_ic_mou_8 and aon are more likely to churn compared to the one with higher total_ic_mou_8 and aon.

In [120]:

```
sns.scatterplot(x=data['aon'],y=data['total_ic_mou_8'],hue=data['Churn'])
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc197fbdc0>
```



- The customers with less total_ic_mou_8 are more likely to churn irrespective of aon.
- The customers with total_ic_mou_8 > 2000 are very less likely to churn.

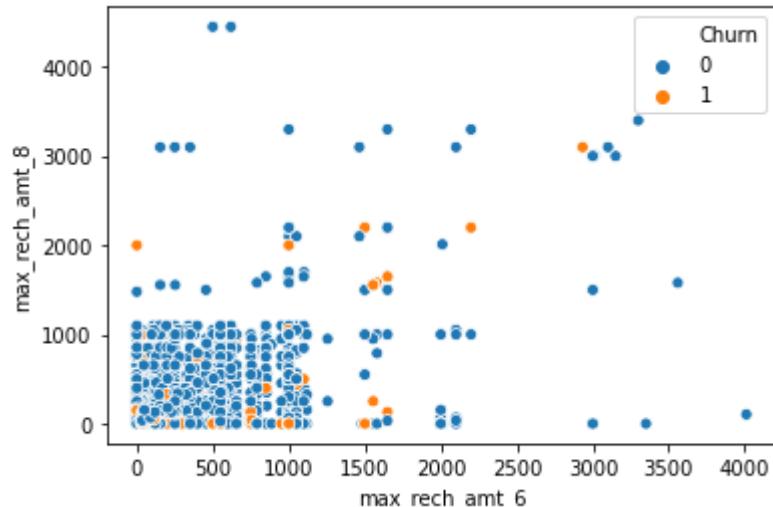
'max_rech_amt_6' vs 'max_rech_amt_8' with respect to 'Churn'.

In [124]:

```
sns.scatterplot(x=data['max_rech_amt_6'],y=data['max_rech_amt_8'],hue=data['Churn'])
```

Out[124]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc1b2ad970>
```



Correlation Analysis

In [186]:

```
# function to correlate variables
def correlation(dataframe) :

    columnsForAnalysis = set(dataframe.columns.values) - {'Churn'}
    cor0=dataframe[columnsForAnalysis].corr()
    type(cor0)
    cor0.where(np.triu(np.ones(cor0.shape),k=1).astype(np.bool))
    cor0=cor0.unstack().reset_index()
    cor0.columns=[ 'VAR1' , 'VAR2' , 'CORR' ]
    cor0.dropna(subset=[ 'CORR' ], inplace=True)
    cor0.CORR=round(cor0[ 'CORR' ],2)
    cor0.CORR=cor0.CORR.abs()
    cor0.sort_values(by=[ 'CORR' ],ascending=False)
    cor0=cor0[~(cor0[ 'VAR1' ]==cor0[ 'VAR2' ])]

    # removing duplicate correlations
    cor0[ 'pair' ] = cor0[['VAR1', 'VAR2']].apply(lambda x: '{}-{}'.format(*sorted((x[0], x[1])))), axis=1)

    cor0 = cor0.drop_duplicates(subset=[ 'pair' ], keep='first')
    cor0 = cor0[['VAR1', 'VAR2','CORR']]
    return pd.DataFrame(cor0.sort_values(by=[ 'CORR' ],ascending=False))
```

In [187]:

```
# Correlations for Churn : 0 - non churn customers
# Absolute values are reported
pd.set_option('precision', 2)
cor_0 = correlation(non_churned_customers)

# filtering for correlations >= 40%
condition = cor_0['CORR'] > 0.4
cor_0 = cor_0[condition]
cor_0.style.background_gradient(cmap='GnBu').hide_index()
```

Out[187]:

VAR1	VAR2	CORR
isd_og_mou_8	isd_og_mou_7	0.96
isd_og_mou_8	isd_og_mou_6	0.95
arpu_8	total_rech_amt_8	0.95
isd_og_mou_7	isd_og_mou_6	0.95
arpu_6	total_rech_amt_6	0.94
total_rech_amt_7	arpu_7	0.94
Average_rech_amt_6n7	arpu_7	0.91
total_rech_amt_7	Average_rech_amt_6n7	0.91
total_ic_mou_6	loc_ic_mou_6	0.90
Average_rech_amt_6n7	total_rech_amt_6	0.90
arpu_6	Average_rech_amt_6n7	0.89
loc_ic_mou_8	total_ic_mou_8	0.89
loc_ic_mou_7	total_ic_mou_7	0.88
loc_ic_mou_7	loc_ic_mou_8	0.85
std_og_t2t_mou_8	onnet_mou_8	0.85
loc_ic_mou_8	loc_ic_t2m_mou_8	0.85
loc_ic_t2m_mou_6	loc_ic_mou_6	0.85
std_og_t2m_mou_8	offnet_mou_8	0.85
std_og_t2t_mou_7	onnet_mou_7	0.84
std_og_mou_8	total_og_mou_8	0.84
loc_og_mou_8	loc_og_mou_7	0.84
std_ic_mou_8	std_ic_t2m_mou_8	0.84
std_og_t2t_mou_6	onnet_mou_6	0.84
offnet_mou_7	std_og_t2m_mou_7	0.84
total_og_mou_7	std_og_mou_7	0.83
loc_ic_mou_7	loc_ic_mou_6	0.83
total_ic_mou_7	total_ic_mou_8	0.83
loc_og_t2t_mou_8	loc_og_t2t_mou_7	0.83
loc_ic_mou_7	loc_ic_t2m_mou_7	0.83
loc_ic_t2m_mou_7	loc_ic_t2m_mou_8	0.82
loc_og_t2f_mou_8	loc_og_t2f_mou_7	0.82
loc_og_t2m_mou_8	loc_og_t2m_mou_7	0.82
onnet_mou_7	onnet_mou_8	0.82
std_ic_t2m_mou_6	std_ic_mou_6	0.82
std_og_t2t_mou_7	std_og_t2t_mou_8	0.82
std_ic_t2m_mou_7	std_ic_mou_7	0.81
loc_ic_t2t_mou_6	loc_ic_t2t_mou_7	0.81

VAR1	VAR2	CORR
std_og_mou_8	std_og_mou_7	0.81
offnet_mou_6	std_og_t2m_mou_6	0.81
total_ic_mou_6	total_ic_mou_7	0.81
loc_ic_t2t_mou_8	loc_ic_t2t_mou_7	0.81
total_og_mou_6	std_og_mou_6	0.80
loc_og_mou_6	loc_og_mou_7	0.80
loc_ic_t2m_mou_6	loc_ic_t2m_mou_7	0.80
loc_og_t2t_mou_6	loc_og_t2t_mou_7	0.80
std_og_t2m_mou_8	std_og_t2m_mou_7	0.79
loc_og_t2f_mou_7	loc_og_t2f_mou_6	0.79
loc_ic_t2f_mou_7	loc_ic_t2f_mou_8	0.79
loc_og_mou_6	loc_og_t2m_mou_6	0.79
total_rech_num_8	total_rech_num_7	0.78
loc_og_t2m_mou_7	loc_og_t2m_mou_6	0.78
arpu_8	Average_rech_amt_6n7	0.78
offnet_mou_7	offnet_mou_8	0.78
loc_og_t2t_mou_8	loc_og_mou_8	0.77
arpu_7	total_rech_amt_8	0.77
arpu_8	arpu_7	0.77
arpu_8	total_rech_amt_7	0.77
loc_og_t2m_mou_8	loc_og_mou_8	0.77
total_og_mou_7	total_og_mou_8	0.77
std_og_t2f_mou_7	std_og_t2f_mou_8	0.77
loc_og_mou_7	loc_og_t2t_mou_7	0.77
total_ic_mou_8	loc_ic_t2m_mou_8	0.76
std_ic_mou_8	std_ic_mou_7	0.76
loc_ic_t2m_mou_6	total_ic_mou_6	0.76
isd_ic_mou_7	isd_ic_mou_6	0.75
isd_ic_mou_8	isd_ic_mou_7	0.75
loc_og_mou_6	loc_og_t2t_mou_6	0.75
std_ic_mou_6	std_ic_mou_7	0.75
loc_ic_mou_7	total_ic_mou_8	0.75
loc_og_t2m_mou_7	loc_og_mou_7	0.75
loc_ic_mou_8	loc_ic_mou_6	0.75
total_ic_mou_7	loc_ic_mou_8	0.75
Average_rech_amt_6n7	total_rech_amt_8	0.75
loc_ic_t2f_mou_6	loc_ic_t2f_mou_7	0.75
vol_3g_mb_8	vol_3g_mb_7	0.75
std_og_t2m_mou_6	std_og_t2m_mou_7	0.75

VAR1	VAR2	CORR
std_og_mou_8	std_og_t2m_mou_8	0.75
std_og_t2m_mou_6	std_og_mou_6	0.74
std_og_mou_8	std_og_t2t_mou_8	0.74
std_ic_t2f_mou_7	std_ic_t2f_mou_6	0.74
std_og_t2m_mou_7	std_og_mou_7	0.74
loc_ic_mou_7	total_ic_mou_6	0.74
loc_ic_t2m_mou_7	total_ic_mou_7	0.74
std_og_mou_6	std_og_t2t_mou_6	0.74
std_ic_t2t_mou_6	std_ic_t2t_mou_7	0.74
std_ic_t2t_mou_8	std_ic_t2t_mou_7	0.73
loc_og_t2f_mou_8	loc_og_t2f_mou_6	0.73
std_og_t2t_mou_7	std_og_t2t_mou_6	0.73
std_ic_t2m_mou_7	std_ic_t2m_mou_8	0.73
onnet_mou_7	onnet_mou_6	0.73
total_rech_amt_7	total_rech_amt_8	0.73
loc_og_mou_6	loc_og_mou_8	0.73
total_ic_mou_6	total_ic_mou_8	0.73
std_og_t2t_mou_7	std_og_mou_7	0.73
std_og_mou_6	std_og_mou_7	0.73
total_ic_mou_7	loc_ic_mou_6	0.72
std_ic_t2f_mou_7	std_ic_t2f_mou_8	0.72
offnet_mou_6	offnet_mou_7	0.72
loc_ic_t2m_mou_6	loc_ic_t2m_mou_8	0.72
loc_ic_t2m_mou_7	loc_ic_mou_8	0.72
total_og_mou_8	offnet_mou_8	0.72
std_ic_t2m_mou_7	std_ic_t2m_mou_6	0.72
loc_og_t2t_mou_8	loc_og_t2t_mou_6	0.72
total_og_mou_8	onnet_mou_8	0.71
vbc_3g_8	vbc_3g_7	0.71
vol_3g_mb_6	vol_3g_mb_7	0.71
std_og_t2f_mou_6	std_og_t2f_mou_7	0.71
total_og_mou_7	onnet_mou_7	0.71
ic_others_8	ic_others_7	0.71
total_og_mou_6	offnet_mou_6	0.70
total_rech_amt_6	arpu_7	0.70
total_og_mou_7	offnet_mou_7	0.70
std_ic_t2t_mou_7	std_ic_mou_7	0.70
arpu_6	arpu_7	0.70
total_og_mou_6	onnet_mou_6	0.70

VAR1	VAR2	CORR
loc_ic_t2t_mou_6	loc_ic_t2t_mou_8	0.70
loc_ic_mou_7	loc_ic_t2m_mou_8	0.70
last_day_rch_amt_8	max_rech_amt_8	0.69
std_og_t2t_mou_7	onnet_mou_8	0.69
vbc_3g_7	vbc_3g_6	0.69
loc_og_t2m_mou_8	loc_og_t2m_mou_6	0.69
loc_ic_t2m_mou_7	loc_ic_mou_6	0.69
total_rech_num_6	total_rech_num_7	0.69
vol_2g_mb_7	vol_2g_mb_8	0.69
std_og_t2t_mou_8	onnet_mou_7	0.69
loc_ic_mou_7	loc_ic_t2m_mou_6	0.68
arpu_6	total_rech_amt_7	0.68
loc_ic_mou_7	loc_ic_t2t_mou_7	0.68
total_ic_mou_6	loc_ic_mou_8	0.68
std_ic_t2f_mou_8	std_ic_t2f_mou_6	0.67
ic_others_7	ic_others_6	0.67
loc_ic_t2t_mou_6	loc_ic_mou_6	0.67
loc_ic_t2t_mou_8	loc_ic_mou_8	0.67
vol_2g_mb_7	vol_2g_mb_6	0.67
loc_ic_t2f_mou_6	loc_ic_t2f_mou_8	0.67
total_og_mou_6	total_og_mou_7	0.67
vol_3g_mb_8	vol_3g_mb_6	0.67
std_ic_t2t_mou_6	std_ic_mou_6	0.67
total_ic_mou_8	loc_ic_mou_6	0.66
std_ic_t2t_mou_8	std_ic_mou_8	0.66
total_og_mou_7	std_og_mou_8	0.66
std_og_t2m_mou_8	offnet_mou_7	0.66
std_ic_mou_8	std_ic_mou_6	0.66
vbc_3g_7	vol_3g_mb_7	0.65
max_rech_amt_6	last_day_rch_amt_6	0.65
std_og_t2m_mou_7	offnet_mou_8	0.65
std_og_t2f_mou_6	std_og_t2f_mou_8	0.65
loc_og_t2t_mou_8	loc_og_mou_7	0.65
arpu_6	total_rech_amt_8	0.64
arpu_6	arpu_8	0.64
total_og_mou_8	std_og_mou_7	0.64
std_og_t2m_mou_8	total_og_mou_8	0.64
loc_ic_t2m_mou_6	loc_ic_mou_8	0.64
roam_og_mou_6	roam_ic_mou_6	0.64

VAR1	VAR2	CORR
std_ic_t2m_mou_7	std_ic_mou_8	0.64
loc_og_mou_8	loc_og_t2t_mou_7	0.64
loc_ic_t2m_mou_7	total_ic_mou_8	0.64
total_rech_amt_7	total_rech_amt_6	0.64
total_ic_mou_7	loc_ic_t2m_mou_8	0.63
vol_3g_mb_6	vbc_3g_6	0.63
vbc_3g_8	vol_3g_mb_8	0.63
total_ic_mou_6	loc_ic_t2m_mou_7	0.63
roam_ic_mou_7	roam_og_mou_7	0.63
onnet_mou_8	onnet_mou_6	0.63
loc_og_t2m_mou_8	loc_og_mou_7	0.63
arpu_8	total_rech_amt_6	0.63
std_og_t2t_mou_8	std_og_t2t_mou_6	0.63
loc_ic_t2m_mou_8	loc_ic_mou_6	0.63
loc_og_t2t_mou_6	loc_og_mou_7	0.63
total_og_mou_7	std_og_t2m_mou_7	0.63
ic_others_8	ic_others_6	0.63
std_ic_t2m_mou_7	std_ic_mou_6	0.63
loc_og_mou_8	loc_og_t2m_mou_7	0.63
std_ic_t2m_mou_6	std_ic_t2m_mou_8	0.63
total_rech_amt_6	total_rech_amt_8	0.63
isd_ic_mou_8	isd_ic_mou_6	0.62
std_og_t2t_mou_8	std_og_mou_7	0.62
total_og_mou_8	std_og_t2t_mou_8	0.61
loc_og_t2m_mou_6	loc_og_mou_7	0.61
onnet_mou_7	std_og_t2t_mou_6	0.61
vbc_3g_8	vbc_3g_6	0.61
loc_og_mou_6	loc_og_t2m_mou_7	0.61
std_og_mou_8	onnet_mou_8	0.61
std_ic_t2m_mou_8	std_ic_mou_7	0.61
last_day_rch_amt_7	max_rech_amt_7	0.61
std_og_t2m_mou_6	offnet_mou_7	0.61
std_og_mou_8	std_og_mou_6	0.61
max_rech_amt_6	max_rech_amt_8	0.60
std_og_mou_8	std_og_t2m_mou_7	0.60
total_rech_num_8	total_rech_num_6	0.60
total_ic_mou_7	loc_ic_t2t_mou_7	0.60
loc_ic_t2m_mou_6	total_ic_mou_7	0.60
roam_og_mou_8	roam_ic_mou_8	0.60

VAR1	VAR2	CORR
std_og_t2t_mou_7	onnet_mou_6	0.60
total_og_mou_7	std_og_t2t_mou_7	0.60
std_og_mou_8	std_og_t2t_mou_7	0.60
loc_og_mou_6	loc_og_t2t_mou_7	0.60
total_og_mou_6	std_og_t2m_mou_6	0.60
std_og_mou_8	offnet_mou_8	0.60
std_og_t2m_mou_8	std_og_t2m_mou_6	0.60
std_og_mou_6	onnet_mou_6	0.59
loc_ic_t2t_mou_8	total_ic_mou_8	0.59
onnet_mou_7	std_og_mou_7	0.59
total_ic_mou_6	loc_ic_t2t_mou_6	0.59
std_og_t2m_mou_8	std_og_mou_7	0.59
offnet_mou_6	offnet_mou_8	0.59
std_ic_t2t_mou_6	std_ic_t2t_mou_8	0.59
loc_ic_mou_7	loc_ic_t2t_mou_8	0.59
total_og_mou_7	onnet_mou_8	0.58
std_ic_t2m_mou_6	std_ic_mou_7	0.58
total_og_mou_6	std_og_t2t_mou_6	0.58
offnet_mou_6	std_og_t2m_mou_7	0.58
roam_og_mou_8	roam_og_mou_7	0.58
total_ic_mou_6	loc_ic_t2m_mou_8	0.58
spl_og_mou_7	spl_og_mou_8	0.57
total_og_mou_7	std_og_mou_6	0.57
offnet_mou_7	std_og_mou_7	0.57
loc_ic_mou_7	loc_ic_t2t_mou_6	0.57
loc_og_t2t_mou_8	loc_og_mou_6	0.57
std_og_t2m_mou_6	std_og_mou_7	0.56
max_rech_amt_7	max_rech_amt_8	0.56
loc_ic_t2m_mou_6	total_ic_mou_8	0.56
spl_og_mou_7	spl_og_mou_6	0.56
roam_ic_mou_7	roam_ic_mou_8	0.56
std_ic_t2m_mou_8	std_ic_mou_6	0.56
loc_og_mou_8	loc_og_t2t_mou_6	0.56
loc_ic_mou_8	loc_ic_t2t_mou_7	0.56
loc_og_mou_8	loc_og_t2m_mou_6	0.56
total_og_mou_8	onnet_mou_7	0.56
total_og_mou_6	total_og_mou_8	0.55
loc_og_t2m_mou_8	loc_og_mou_6	0.55
std_ic_t2t_mou_6	std_ic_mou_7	0.55

VAR1	VAR2	CORR
loc_ic_t2t_mou_7	loc_ic_mou_6	0.55
total_og_mou_7	offnet_mou_8	0.54
std_og_t2t_mou_7	std_og_mou_6	0.54
std_ic_mou_8	std_ic_t2m_mou_6	0.54
offnet_mou_6	std_og_mou_6	0.54
std_og_mou_6	std_og_t2m_mou_7	0.54
total_og_mou_8	offnet_mou_7	0.54
spl_og_mou_7	loc_og_t2c_mou_7	0.53
std_og_t2t_mou_6	std_og_mou_7	0.53
total_og_mou_6	std_og_mou_7	0.53
std_og_t2t_mou_6	onnet_mou_8	0.53
std_ic_t2t_mou_8	std_ic_mou_7	0.53
loc_og_t2c_mou_8	loc_og_t2c_mou_7	0.53
Average_rech_amt_6n7	isd_og_mou_7	0.53
std_og_t2t_mou_8	onnet_mou_6	0.52
vol_2g_mb_8	vol_2g_mb_6	0.52
isd_og_mou_7	arpu_7	0.52
loc_ic_t2t_mou_8	loc_ic_mou_6	0.51
arpu_8	total_og_mou_8	0.51
total_ic_mou_7	loc_ic_t2t_mou_8	0.51
vol_3g_mb_7	vbc_3g_6	0.51
vbc_3g_8	vol_3g_mb_7	0.51
total_og_mou_6	arpu_6	0.51
total_rech_amt_7	isd_og_mou_7	0.50
roam_og_mou_7	roam_og_mou_6	0.50
onnet_mou_8	std_og_mou_7	0.50
Average_rech_amt_6n7	isd_og_mou_6	0.50
loc_ic_t2t_mou_6	total_ic_mou_7	0.50
loc_ic_t2t_mou_6	loc_ic_mou_8	0.50
std_ic_t2t_mou_7	std_ic_mou_6	0.50
max_rech_amt_6	max_rech_amt_7	0.50
isd_og_mou_8	Average_rech_amt_6n7	0.50
std_ic_mou_8	std_ic_t2t_mou_7	0.50
loc_ic_t2m_mou_6	loc_og_t2m_mou_6	0.50
total_og_mou_6	total_rech_amt_6	0.49
loc_ic_t2t_mou_7	total_ic_mou_8	0.49
total_og_mou_7	std_og_t2m_mou_8	0.49
isd_og_mou_8	total_rech_amt_8	0.49
total_og_mou_7	std_og_t2t_mou_8	0.49

VAR1	VAR2	CORR
total_og_mou_8	total_rech_amt_8	0.49
loc_og_t2m_mou_7	loc_ic_t2m_mou_7	0.49
total_ic_mou_6	loc_ic_t2t_mou_7	0.49
vbc_3g_7	vol_3g_mb_8	0.49
std_og_mou_8	onnet_mou_7	0.49
loc_og_t2m_mou_8	loc_ic_t2m_mou_8	0.49
total_og_mou_7	onnet_mou_6	0.48
total_og_mou_7	offnet_mou_6	0.48
std_og_t2m_mou_6	offnet_mou_8	0.48
total_og_mou_7	arpu_7	0.48
max_rech_amt_8	total_rech_amt_8	0.48
isd_og_mou_8	arpu_7	0.48
total_og_mou_8	std_og_t2m_mou_7	0.48
arpu_6	isd_og_mou_6	0.48
total_og_mou_6	onnet_mou_7	0.48
isd_og_mou_7	total_rech_amt_8	0.48
isd_og_mou_8	arpu_8	0.48
loc_og_t2c_mou_6	spl_og_mou_6	0.48
offnet_mou_6	std_og_t2m_mou_8	0.47
vol_3g_mb_8	vbc_3g_6	0.47
total_rech_amt_6	isd_og_mou_6	0.47
onnet_mou_7	loc_og_t2t_mou_7	0.47
std_og_t2t_mou_8	std_og_mou_6	0.47
arpu_6	isd_og_mou_7	0.47
total_og_mou_6	offnet_mou_7	0.47
offnet_mou_6	loc_og_t2m_mou_6	0.47
std_og_t2t_mou_7	total_og_mou_8	0.47
loc_og_t2t_mou_6	onnet_mou_6	0.47
arpu_8	offnet_mou_8	0.47
roam_ic_mou_7	roam_ic_mou_6	0.46
arpu_7	isd_og_mou_6	0.46
offnet_mou_8	total_rech_amt_8	0.46
total_og_mou_7	total_rech_amt_7	0.46
spl_og_mou_8	loc_og_t2c_mou_8	0.46
isd_og_mou_8	arpu_6	0.46
isd_og_mou_7	total_rech_amt_6	0.46
std_og_mou_8	std_og_t2m_mou_6	0.46
arpu_8	isd_og_mou_7	0.46
std_ic_mou_8	total_ic_mou_8	0.46

VAR1	VAR2	CORR
total_rech_amt_7	max_rech_amt_7	0.46
vbc_3g_7	vol_3g_mb_6	0.46
total_og_mou_8	std_og_mou_6	0.46
loc_og_t2t_mou_8	onnet_mou_8	0.46
arpu_6	offnet_mou_6	0.46
isd_og_mou_8	total_rech_amt_7	0.46
offnet_mou_6	total_rech_amt_6	0.45
total_ic_mou_6	loc_ic_t2t_mou_8	0.45
total_ic_mou_7	std_ic_mou_7	0.45
std_og_mou_8	offnet_mou_7	0.45
isd_og_mou_8	total_rech_amt_6	0.45
total_og_mou_7	std_og_t2m_mou_6	0.45
std_og_mou_8	std_og_t2t_mou_6	0.45
loc_og_mou_6	loc_ic_mou_6	0.45
total_ic_mou_6	std_ic_mou_6	0.44
loc_og_t2m_mou_8	loc_ic_mou_8	0.44
loc_og_t2m_mou_8	offnet_mou_8	0.44
total_og_mou_6	loc_og_mou_6	0.44
total_og_mou_6	std_og_mou_8	0.44
isd_og_mou_6	total_rech_amt_8	0.44
std_og_mou_7	offnet_mou_8	0.44
std_og_t2m_mou_8	std_og_mou_6	0.44
loc_og_t2m_mou_6	loc_ic_mou_6	0.44
std_ic_t2t_mou_6	std_ic_mou_8	0.44
loc_og_mou_8	loc_ic_mou_8	0.44
offnet_mou_7	arpu_7	0.44
arpu_8	isd_og_mou_6	0.43
loc_og_t2m_mou_7	loc_ic_t2m_mou_8	0.43
max_rech_amt_6	total_rech_amt_6	0.43
loc_og_t2m_mou_8	loc_ic_t2m_mou_7	0.43
total_rech_amt_7	isd_og_mou_6	0.43
vbc_3g_8	vol_3g_mb_6	0.43
total_rech_amt_7	offnet_mou_7	0.43
loc_ic_t2t_mou_6	total_ic_mou_8	0.43
onnet_mou_7	std_og_mou_6	0.43
loc_og_mou_8	total_og_mou_8	0.42
loc_ic_t2m_mou_7	loc_og_t2m_mou_6	0.42
total_og_mou_6	Average_rech_amt_6n7	0.42
loc_ic_mou_7	loc_og_mou_7	0.42

VAR1	VAR2	CORR
loc_ic_mou_7	loc_og_t2m_mou_7	0.42
total_og_mou_7	Average_rech_amt_6n7	0.42
last_day_rch_amt_6	max_rech_amt_8	0.42
std_ic_t2t_mou_8	std_ic_mou_6	0.42
loc_ic_t2m_mou_6	loc_og_t2m_mou_7	0.42
loc_og_t2m_mou_7	offnet_mou_7	0.42
total_og_mou_6	onnet_mou_8	0.42
spl_og_mou_8	spl_og_mou_6	0.41
last_day_rch_amt_8	max_rech_amt_7	0.41
offnet_mou_6	Average_rech_amt_6n7	0.41
max_rech_amt_6	last_day_rch_amt_8	0.41
loc_ic_t2m_mou_6	loc_og_mou_6	0.41
total_og_mou_8	onnet_mou_6	0.41

In [188]:

```
# Correlations for Churn : 1 - churned customers
# Absolute values are reported
pd.set_option('precision', 2)
cor_1 = correlation(churned_customers)

# filtering for correlations >= 40%
condition = cor_1['CORR'] > 0.4
cor_1 = cor_1[condition]
cor_1.style.background_gradient(cmap='GnBu').hide_index()
```

Out[188]:

VAR1	VAR2	CORR
og_others_7	og_others_8	1.00
arpu_8	total_rech_amt_8	0.96
arpu_6	total_rech_amt_6	0.95
std_og_mou_8	total_og_mou_8	0.95
total_rech_amt_7	arpu_7	0.95
std_og_t2t_mou_7	onnet_mou_7	0.95
total_og_mou_7	std_og_mou_7	0.94
og_others_8	loc_og_t2f_mou_6	0.93
std_og_t2t_mou_8	onnet_mou_8	0.93
loc_og_t2f_mou_7	loc_og_t2f_mou_6	0.93
og_others_7	loc_og_t2f_mou_6	0.93
total_og_mou_6	std_og_mou_6	0.92
offnet_mou_6	std_og_t2m_mou_6	0.92
offnet_mou_7	std_og_t2m_mou_7	0.92
std_og_t2t_mou_6	onnet_mou_6	0.92
std_ic_mou_8	std_ic_t2m_mou_8	0.92
loc_og_t2f_mou_7	og_others_8	0.91
loc_og_t2f_mou_7	og_others_7	0.91
loc_ic_mou_8	loc_ic_t2m_mou_8	0.90
loc_ic_t2m_mou_6	loc_ic_mou_6	0.90
loc_ic_mou_8	total_ic_mou_8	0.89
loc_og_t2m_mou_8	loc_og_mou_8	0.88
total_ic_mou_6	loc_ic_mou_6	0.87
std_og_t2m_mou_8	offnet_mou_8	0.87
loc_ic_mou_7	total_ic_mou_7	0.86
loc_ic_mou_7	loc_ic_t2m_mou_7	0.84
loc_og_t2m_mou_7	loc_og_mou_7	0.84
std_ic_t2m_mou_7	std_ic_mou_7	0.82
total_ic_mou_8	loc_ic_t2m_mou_8	0.81
std_og_mou_8	std_og_t2t_mou_8	0.79
std_ic_t2t_mou_6	std_ic_t2t_mou_7	0.78
Average_rech_amt_6n7	arpu_7	0.77
loc_og_mou_6	loc_og_t2m_mou_6	0.77
loc_ic_t2m_mou_6	total_ic_mou_6	0.77
std_ic_t2m_mou_6	std_ic_mou_6	0.77
total_rech_amt_7	Average_rech_amt_6n7	0.76
Average_rech_amt_6n7	total_rech_amt_6	0.76

VAR1	VAR2	CORR
loc_og_mou_6	loc_og_t2t_mou_6	0.75
total_og_mou_8	std_og_t2t_mou_8	0.75
std_og_t2m_mou_7	std_og_mou_7	0.74
std_og_mou_8	onnet_mou_8	0.74
total_og_mou_8	onnet_mou_8	0.74
arp_6	Average_rech_amt_6n7	0.73
loc_og_t2t_mou_8	loc_og_t2t_mou_7	0.73
loc_ic_t2t_mou_8	loc_ic_mou_8	0.73
loc_ic_t2t_mou_6	loc_ic_mou_6	0.72
max_rech_amt_6	last_day_rch_amt_6	0.72
std_og_t2m_mou_6	std_og_mou_6	0.72
std_ic_t2t_mou_6	std_ic_mou_6	0.72
roam_ic_mou_7	roam_ic_mou_8	0.72
total_og_mou_7	offnet_mou_7	0.72
loc_ic_t2m_mou_7	total_ic_mou_7	0.72
std_og_mou_8	std_og_t2m_mou_8	0.71
total_og_mou_8	offnet_mou_8	0.70
last_day_rch_amt_8	max_rech_amt_8	0.70
loc_og_mou_7	loc_og_t2t_mou_7	0.69
std_og_t2t_mou_7	std_og_mou_7	0.69
total_og_mou_7	std_og_t2m_mou_7	0.69
loc_ic_mou_7	loc_ic_t2t_mou_7	0.69
loc_og_t2t_mou_8	loc_og_mou_8	0.68
total_og_mou_6	offnet_mou_6	0.68
std_og_mou_6	std_og_t2t_mou_6	0.68
std_og_t2m_mou_8	total_og_mou_8	0.68
max_rech_amt_8	total_rech_amt_8	0.68
spl_og_mou_7	loc_og_t2c_mou_7	0.68
loc_ic_t2f_mou_6	loc_ic_t2f_mou_7	0.67
vol_3g_mb_8	vol_3g_mb_7	0.67
std_og_t2t_mou_7	std_og_t2t_mou_6	0.67
total_og_mou_6	std_og_t2m_mou_6	0.67
offnet_mou_7	std_og_mou_7	0.66
total_og_mou_7	onnet_mou_7	0.66
onnet_mou_7	std_og_mou_7	0.65
loc_og_mou_8	loc_ic_mou_8	0.65
std_ic_t2t_mou_7	std_ic_mou_7	0.65
loc_og_t2m_mou_8	loc_ic_t2m_mou_8	0.65
roam_og_mou_8	roam_og_mou_7	0.65

VAR1	VAR2	CORR
total_og_mou_6	onnet_mou_6	0.65
total_og_mou_7	std_og_t2t_mou_7	0.64
loc_ic_t2t_mou_8	total_ic_mou_8	0.64
onnet_mou_7	onnet_mou_6	0.64
loc_og_mou_8	loc_ic_t2m_mou_8	0.64
onnet_mou_7	std_og_t2t_mou_6	0.63
loc_og_mou_8	loc_og_mou_7	0.63
total_ic_mou_6	loc_ic_t2t_mou_6	0.63
offnet_mou_6	std_og_mou_6	0.63
roam_og_mou_6	roam_ic_mou_6	0.63
std_og_mou_8	offnet_mou_8	0.63
roam_ic_mou_7	roam_ic_mou_6	0.62
arpu_8	max_rech_amt_8	0.62
std_og_t2m_mou_6	std_og_t2m_mou_7	0.62
total_og_mou_6	std_og_t2t_mou_6	0.62
vol_3g_mb_6	vbc_3g_6	0.62
onnet_mou_7	onnet_mou_8	0.62
loc_ic_t2f_mou_7	loc_ic_t2f_mou_8	0.62
loc_og_mou_8	total_ic_mou_8	0.61
vbc_3g_8	vbc_3g_7	0.61
loc_og_t2m_mou_7	loc_og_t2m_mou_6	0.61
std_og_t2t_mou_7	onnet_mou_6	0.61
roam_og_mou_7	roam_og_mou_6	0.61
std_og_t2t_mou_7	std_og_t2t_mou_8	0.61
std_og_mou_6	onnet_mou_6	0.61
std_og_t2t_mou_8	onnet_mou_7	0.60
std_ic_mou_6	std_ic_mou_7	0.60
loc_ic_t2m_mou_6	loc_ic_t2m_mou_7	0.60
arpu_8	total_og_mou_8	0.60
std_og_t2f_mou_7	std_og_t2f_mou_8	0.60
isd_og_mou_8	isd_og_mou_7	0.60
loc_og_mou_6	loc_og_mou_7	0.59
loc_og_t2m_mou_8	loc_og_t2m_mou_7	0.59
last_day_rch_amt_7	max_rech_amt_7	0.59
arpu_8	offnet_mou_8	0.59
std_og_mou_8	std_og_mou_7	0.58
total_og_mou_8	total_rech_amt_8	0.58
loc_og_t2m_mou_7	loc_ic_t2m_mou_7	0.58
loc_og_t2m_mou_8	loc_ic_mou_8	0.58

VAR1	VAR2	CORR
loc_ic_mou_7	loc_ic_mou_8	0.58
std_og_t2m_mou_8	std_og_t2m_mou_7	0.58
total_ic_mou_7	loc_ic_t2t_mou_7	0.58
offnet_mou_8	total_rech_amt_8	0.58
std_og_mou_6	std_og_mou_7	0.58
spl_og_mou_8	loc_og_t2c_mou_8	0.57
loc_ic_mou_7	loc_ic_mou_6	0.57
isd_ic_mou_7	isd_ic_mou_6	0.57
offnet_mou_6	offnet_mou_7	0.57
offnet_mou_7	offnet_mou_8	0.57
vol_3g_mb_6	vol_3g_mb_7	0.57
isd_og_mou_7	arpu_7	0.57
loc_ic_t2m_mou_7	loc_ic_t2m_mou_8	0.57
total_rech_num_8	total_og_mou_8	0.57
loc_og_t2t_mou_6	loc_og_t2t_mou_7	0.56
std_og_t2t_mou_7	onnet_mou_8	0.56
total_og_mou_7	total_og_mou_8	0.56
vbc_3g_7	vol_3g_mb_7	0.56
total_rech_amt_7	isd_og_mou_7	0.56
std_ic_mou_8	total_ic_mou_8	0.56
loc_ic_t2t_mou_8	loc_ic_t2t_mou_7	0.56
loc_og_t2c_mou_6	spl_og_mou_6	0.56
std_og_t2m_mou_6	offnet_mou_7	0.55
ic_others_7	ic_others_6	0.55
total_og_mou_7	std_og_mou_8	0.55
total_ic_mou_6	total_ic_mou_7	0.55
total_rech_num_8	total_rech_amt_8	0.55
arpu_8	total_rech_num_8	0.54
loc_ic_t2m_mou_7	loc_ic_mou_6	0.54
std_ic_t2t_mou_8	std_ic_mou_8	0.54
loc_og_t2c_mou_6	loc_og_t2c_mou_7	0.54
std_og_t2m_mou_8	offnet_mou_7	0.54
total_rech_num_8	total_rech_num_7	0.54
total_ic_mou_7	std_ic_mou_7	0.54
std_ic_t2t_mou_7	std_ic_mou_6	0.54
loc_ic_mou_7	loc_ic_t2m_mou_6	0.54
offnet_mou_6	std_og_t2m_mou_7	0.54
std_og_mou_8	total_rech_num_8	0.54
loc_og_t2m_mou_8	total_ic_mou_8	0.54

VAR1	VAR2	CORR
total_ic_mou_7	total_ic_mou_8	0.54
std_ic_t2t_mou_8	std_ic_t2t_mou_7	0.54
total_ic_mou_6	std_ic_mou_6	0.53
vol_2g_mb_7	vol_2g_mb_6	0.53
vbc_3g_7	vbc_3g_6	0.53
arpu_6	isd_og_mou_6	0.53
total_og_mou_8	std_og_mou_7	0.52
std_ic_mou_8	std_ic_mou_7	0.52
total_rech_amt_6	isd_og_mou_6	0.52
loc_og_mou_8	loc_og_t2m_mou_7	0.52
arpu_8	std_og_mou_8	0.51
loc_og_t2m_mou_8	loc_og_mou_7	0.51
loc_ic_t2m_mou_7	loc_og_mou_7	0.51
loc_og_t2m_mou_6	loc_og_mou_7	0.51
arpu_8	total_rech_amt_7	0.51
total_og_mou_7	arpu_7	0.51
total_og_mou_6	total_og_mou_7	0.51
roam_ic_mou_7	roam_og_mou_7	0.51
loc_ic_t2m_mou_7	loc_ic_mou_8	0.51
total_og_mou_7	std_og_mou_6	0.51
loc_ic_t2m_mou_6	loc_og_t2m_mou_6	0.50
std_ic_t2m_mou_7	std_ic_t2m_mou_8	0.50
std_ic_t2m_mou_7	std_ic_t2m_mou_6	0.50
total_og_mou_6	std_og_mou_7	0.50
total_rech_amt_7	max_rech_amt_7	0.50
std_og_t2m_mou_7	offnet_mou_8	0.50
arpu_8	onnet_mou_8	0.50
onnet_mou_8	total_rech_amt_8	0.50
loc_ic_mou_7	loc_og_mou_7	0.50
total_ic_mou_7	loc_ic_mou_8	0.50
std_og_mou_8	total_rech_amt_8	0.50
loc_ic_mou_7	loc_ic_t2m_mou_8	0.50
last_day_rch_amt_8	total_rech_amt_8	0.50
std_og_t2f_mou_7	loc_og_t2f_mou_7	0.50
loc_og_t2t_mou_8	loc_og_mou_7	0.50
arpu_8	arpu_7	0.50
loc_ic_mou_7	total_ic_mou_6	0.49
std_ic_t2t_mou_6	std_ic_t2t_mou_8	0.49
loc_ic_mou_7	loc_og_t2m_mou_7	0.49

VAR1	VAR2	CORR
loc_ic_mou_7	total_ic_mou_8	0.49
vol_2g_mb_7	vol_2g_mb_8	0.49
vbc_3g_8	vol_3g_mb_8	0.49
loc_ic_mou_8	loc_ic_t2f_mou_8	0.49
arpu_7	total_rech_amt_8	0.48
std_og_t2f_mou_7	og_others_8	0.48
loc_og_t2t_mou_8	loc_ic_t2t_mou_8	0.48
vol_3g_mb_8	vol_3g_mb_6	0.48
std_ic_t2t_mou_6	std_ic_mou_7	0.48
isd_og_mou_7	isd_ic_mou_7	0.48
isd_ic_mou_8	isd_ic_mou_7	0.48
std_ic_t2m_mou_8	total_ic_mou_8	0.48
total_og_mou_7	total_rech_amt_7	0.48
std_og_t2f_mou_7	og_others_7	0.48
std_og_t2f_mou_7	loc_og_t2f_mou_6	0.48
loc_og_mou_8	total_og_mou_8	0.47
total_rech_num_8	onnet_mou_8	0.47
total_ic_mou_6	loc_ic_t2m_mou_7	0.47
loc_ic_mou_7	loc_ic_t2t_mou_8	0.47
total_og_mou_6	arpu_6	0.47
std_og_mou_8	std_og_t2t_mou_7	0.46
Average_rech_amt_6n7	isd_og_mou_7	0.46
spl_og_mou_7	spl_og_mou_8	0.46
loc_ic_t2f_mou_6	loc_ic_t2f_mou_8	0.46
roam_og_mou_8	last_day_rch_amt_8	0.46
total_og_mou_7	total_rech_num_7	0.46
total_og_mou_6	total_rech_amt_6	0.46
total_ic_mou_7	loc_ic_mou_6	0.46
std_ic_t2m_mou_7	std_ic_mou_8	0.46
loc_og_mou_8	loc_og_t2t_mou_7	0.46
max_rech_amt_6	total_rech_amt_6	0.46
arpu_8	roam_og_mou_8	0.45
loc_og_t2m_mou_6	loc_ic_mou_6	0.45
total_rech_num_8	std_og_t2t_mou_8	0.45
loc_og_mou_6	loc_og_t2t_mou_7	0.45
std_ic_t2m_mou_8	std_ic_mou_7	0.45
std_ic_t2m_mou_7	total_ic_mou_7	0.45
total_rech_num_6	total_rech_num_7	0.45
offnet_mou_7	arpu_7	0.45

VAR1	VAR2	CORR
loc_og_mou_6	loc_ic_mou_6	0.45
std_ic_t2f_mou_7	std_ic_t2f_mou_6	0.45
max_rech_amt_7	max_rech_amt_8	0.45
total_rech_amt_7	total_rech_amt_8	0.45
loc_og_mou_6	loc_og_t2m_mou_7	0.45
std_og_t2m_mou_8	std_og_mou_7	0.44
arpu_8	Average_rech_amt_6n7	0.44
total_ic_mou_7	loc_og_mou_7	0.44
std_og_mou_8	onnet_mou_7	0.44
loc_og_t2c_mou_8	loc_og_t2c_mou_7	0.44
roam_og_mou_8	roam_ic_mou_8	0.44
loc_ic_t2m_mou_6	total_ic_mou_7	0.44
roam_og_mou_8	total_rech_amt_8	0.44
arpu_8	last_day_rch_amt_8	0.44
ic_others_6	isd_ic_mou_6	0.44
loc_og_t2m_mou_8	offnet_mou_8	0.44
loc_ic_t2m_mou_7	total_ic_mou_8	0.43
std_og_t2t_mou_8	std_og_mou_7	0.43
loc_og_mou_8	offnet_mou_8	0.43
total_og_mou_6	total_rech_num_6	0.43
total_og_mou_8	onnet_mou_7	0.43
std_ic_t2m_mou_6	total_ic_mou_6	0.43
total_rech_num_8	offnet_mou_8	0.43
spl_og_mou_7	spl_og_mou_6	0.43
total_ic_mou_8	loc_ic_t2f_mou_8	0.43
std_ic_t2f_mou_7	std_og_t2f_mou_7	0.43
loc_og_t2t_mou_8	loc_ic_mou_8	0.42
loc_ic_t2m_mou_6	loc_og_mou_6	0.42
loc_ic_mou_7	loc_ic_t2f_mou_7	0.42
total_ic_mou_7	loc_ic_t2m_mou_8	0.42
max_rech_amt_6	max_rech_amt_8	0.42
loc_og_mou_8	loc_ic_t2t_mou_8	0.42
std_og_t2t_mou_7	std_og_mou_6	0.42
std_ic_t2m_mou_6	std_ic_mou_7	0.42
arpu_8	total_ic_mou_8	0.42
loc_og_t2m_mou_8	loc_ic_t2m_mou_7	0.42
last_day_rch_amt_7	max_rech_amt_8	0.42
arpu_6	offnet_mou_6	0.42
total_rech_amt_7	offnet_mou_7	0.42

VAR1	VAR2	CORR
loc_og_t2m_mou_7	total_ic_mou_7	0.42
total_og_mou_7	std_og_t2m_mou_8	0.42
Average_rech_amt_6n7	total_rech_amt_8	0.42
std_og_mou_7	arpu_7	0.41
std_og_mou_6	std_og_t2m_mou_7	0.41
total_rech_num_7	std_og_mou_7	0.41
total_og_mou_7	offnet_mou_8	0.41
spl_ic_mou_8	spl_ic_mou_6	0.41
std_og_t2t_mou_6	std_og_mou_7	0.41
offnet_mou_6	total_rech_amt_6	0.41
loc_og_t2t_mou_8	loc_og_t2t_mou_6	0.41
std_og_t2t_mou_7	total_og_mou_8	0.41
total_og_mou_8	total_ic_mou_8	0.41
std_og_t2m_mou_6	std_og_mou_7	0.41

Data Preparation

Derived Variables

In [189]:

```
# Derived variables to measure change in usage

# Usage
data['delta_vol_2g'] = data['vol_2g_mb_8'] - data['vol_2g_mb_6'].add(data['vol_2g_mb_7']).div(2)
data['delta_vol_3g'] = data['vol_3g_mb_8'] - data['vol_3g_mb_6'].add(data['vol_3g_mb_7']).div(2)
data['delta_total_og_mou'] = data['total_og_mou_8'] - data['total_og_mou_6'].add(data['total_og_mou_7']).div(2)
data['delta_total_ic_mou'] = data['total_ic_mou_8'] - data['total_ic_mou_6'].add(data['total_ic_mou_7']).div(2)
data['delta_vbc_3g'] = data['vbc_3g_8'] - data['vbc_3g_6'].add(data['vbc_3g_7']).div(2)

# Revenue
data['delta_arpu'] = data['arpu_8'] - data['arpu_6'].add(data['arpu_7']).div(2)
data['delta_total_rech_amt'] = data['total_rech_amt_8'] - data['total_rech_amt_6'].add(data['total_rech_amt_7']).div(2)
```

In [190]:

```
# Removing variables used for derivation :  
data.drop(columns=[  
    'vol_2g_mb_8', 'vol_2g_mb_6', 'vol_2g_mb_7',  
    'vol_3g_mb_8', 'vol_3g_mb_6', 'vol_3g_mb_7',  
    'total_og_mou_8', 'total_og_mou_6', 'total_og_mou_7',  
    'total_ic_mou_8', 'total_ic_mou_6', 'total_ic_mou_7',  
    'vbc_3g_8', 'vbc_3g_6', 'vbc_3g_7',  
    'arpu_8', 'arpu_6', 'arpu_7',  
    'total_rech_amt_8', 'total_rech_amt_6', 'total_rech_amt_7'  
], inplace=True)
```

Outlier Treatment

In [191]:

```
# Looking at quantiles from 0.90 to 1.  
data.quantile(np.arange(0.9, 1.01, 0.01)).style.bar()
```

Out[191]:

	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou_6	offnet_mou_7	o
0.9	794.98	824.38	723.61	915.58	935.69	
0.91	848.97	878.35	783.49	966.74	984.02	
0.92	909.05	941.99	848.96	1031.39	1038.09	
0.93	990.48	1016.15	920.96	1094.77	1103.93	
0.9400000000000001	1066.85	1097.12	1007.56	1168.09	1186.36	
0.9500000000000001	1153.97	1208.17	1115.66	1271.47	1286.28	
0.9600000000000001	1282.78	1344.04	1256.34	1406.07	1407.78	
0.9700000000000001	1444.23	1497.25	1441.53	1578.82	1585.02	
0.9800000000000001	1694.68	1772.62	1700.24	1837.93	1838.39	
0.9900000000000001	2166.37	2220.37	2188.50	2326.29	2410.10	
1.0	7376.71	8157.78	10752.56	8362.36	9667.13	

In [192]:

```
# Looking at percentage change in quantiles from 0.90 to 1.  
data.quantile(np.arange(0.9,1.01,0.01)).pct_change().mul(100).style.bar()
```

Out[192]:

	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou_6	offnet_mou_7	o
0.9	nan	nan	nan	nan	nan	nan
0.91	6.79	6.55	8.27	5.59	5.17	
0.92	7.08	7.25	8.36	6.69	5.49	
0.93	8.96	7.87	8.48	6.14	6.34	
0.9400000000000001	7.71	7.97	9.40	6.70	7.47	
0.9500000000000001	8.17	10.12	10.73	8.85	8.42	
0.9600000000000001	11.16	11.25	12.61	10.59	9.45	
0.9700000000000001	12.59	11.40	14.74	12.29	12.59	
0.9800000000000001	17.34	18.39	17.95	16.41	15.99	
0.9900000000000001	27.83	25.26	28.72	26.57	31.10	
1.0	240.51	267.41	391.32	259.47	301.11	

In [193]:

```
# Columns with outliers
pct_change_99_1 = data.quantile(np.arange(0.9,1.01,0.01)).pct_change().mul(100).iloc[-1]
outlier_condition = pct_change_99_1 > 100
columns_with_outliers = pct_change_99_1[outlier_condition].index.values
print('Columns with outliers :\n', columns_with_outliers)
```

Columns with outliers :

```
['onnet_mou_6' 'onnet_mou_7' 'onnet_mou_8' 'offnet_mou_6' 'offnet_mou_7'
 'offnet_mou_8' 'roam_ic_mou_6' 'roam_ic_mou_7' 'roam_ic_mou_8'
 'roam_og_mou_6' 'roam_og_mou_7' 'roam_og_mou_8' 'loc_og_t2t_mou_6'
 'loc_og_t2t_mou_7' 'loc_og_t2t_mou_8' 'loc_og_t2m_mou_6'
 'loc_og_t2m_mou_7' 'loc_og_t2m_mou_8' 'loc_og_t2f_mou_6'
 'loc_og_t2f_mou_7' 'loc_og_t2f_mou_8' 'loc_og_t2c_mou_6'
 'loc_og_t2c_mou_7' 'loc_og_t2c_mou_8' 'loc_og_mou_6' 'loc_og_mou_7'
 'loc_og_mou_8' 'std_og_t2t_mou_6' 'std_og_t2t_mou_7' 'std_og_t2t_mou_8'
 'std_og_t2m_mou_6' 'std_og_t2m_mou_7' 'std_og_t2m_mou_8'
 'std_og_t2f_mou_6' 'std_og_t2f_mou_7' 'std_og_t2f_mou_8' 'std_og_mou_6'
 'std_og_mou_7' 'std_og_mou_8' 'isd_og_mou_6' 'isd_og_mou_7'
 'isd_og_mou_8' 'spl_og_mou_6' 'spl_og_mou_7' 'spl_og_mou_8' 'og_others_6'
 'og_others_7' 'og_others_8' 'loc_ic_t2t_mou_6' 'loc_ic_t2t_mou_7'
 'loc_ic_t2t_mou_8' 'loc_ic_t2m_mou_6' 'loc_ic_t2m_mou_7'
 'loc_ic_t2m_mou_8' 'loc_ic_t2f_mou_6' 'loc_ic_t2f_mou_7'
 'loc_ic_t2f_mou_8' 'loc_ic_mou_6' 'loc_ic_mou_7' 'loc_ic_mou_8'
 'std_ic_t2t_mou_6' 'std_ic_t2t_mou_7' 'std_ic_t2t_mou_8'
 'std_ic_t2m_mou_6' 'std_ic_t2m_mou_7' 'std_ic_t2m_mou_8'
 'std_ic_t2f_mou_6' 'std_ic_t2f_mou_7' 'std_ic_t2f_mou_8' 'std_ic_mou_6'
 'std_ic_mou_7' 'std_ic_mou_8' 'spl_ic_mou_6' 'spl_ic_mou_7'
 'spl_ic_mou_8' 'isd_ic_mou_6' 'isd_ic_mou_7' 'isd_ic_mou_8' 'ic_others_6'
 'ic_others_7' 'ic_others_8' 'total_rech_num_6' 'total_rech_num_7'
 'total_rech_num_8' 'max_rech_amt_6' 'max_rech_amt_7' 'max_rech_amt_8'
 'last_day_rch_amt_6' 'last_day_rch_amt_7' 'last_day_rch_amt_8'
 'Average_rech_amt_6n7' 'delta_vol_2g' 'delta_vol_3g' 'delta_total_og_mou'
 'delta_total_ic_mou' 'delta_vbc_3g' 'delta_arpu' 'delta_total_rech_amt']
```

In [194]:

```
# capping outliers to 99th percentile values
outlier_treatment = pd.DataFrame(columns=['Column', 'Outlier Threshold', 'Outliers replaced'])
for col in columns_with_outliers :
    outlier_threshold = data[col].quantile(0.99)
    condition = data[col] > outlier_threshold
    outlier_treatment = outlier_treatment.append({'Column' : col , 'Outlier Threshold' : outlier_threshold, 'Outliers replaced' : data.loc[condition,col].shape[0] }, ignore_index=True)
    data.loc[condition, col] = outlier_threshold
outlier_treatment
```

Out[194]:

	Column	Outlier Threshold	Outliers replaced
0	onnet_mou_6	2166.37	301
1	onnet_mou_7	2220.37	301
2	onnet_mou_8	2188.50	301
3	offnet_mou_6	2326.29	301
4	offnet_mou_7	2410.10	301
5	offnet_mou_8	2211.64	301
6	roam_ic_mou_6	349.35	301
7	roam_ic_mou_7	292.54	301
8	roam_ic_mou_8	288.49	301
9	roam_og_mou_6	543.71	301
10	roam_og_mou_7	448.13	301
11	roam_og_mou_8	432.74	301
12	loc_og_t2t_mou_6	1076.24	301
13	loc_og_t2t_mou_7	1059.88	301
14	loc_og_t2t_mou_8	956.50	301
15	loc_og_t2m_mou_6	1147.05	301
16	loc_og_t2m_mou_7	1112.66	301
17	loc_og_t2m_mou_8	1092.59	301
18	loc_og_t2f_mou_6	90.88	301
19	loc_og_t2f_mou_7	91.06	301
20	loc_og_t2f_mou_8	86.68	300
21	loc_og_t2c_mou_6	24.86	301
22	loc_og_t2c_mou_7	28.24	301
23	loc_og_t2c_mou_8	28.87	301
24	loc_og_mou_6	1806.94	301
25	loc_og_mou_7	1761.43	301
26	loc_og_mou_8	1689.07	301
27	std_og_t2t_mou_6	1885.20	301
28	std_og_t2t_mou_7	1919.19	301
29	std_og_t2t_mou_8	1938.13	301
30	std_og_t2m_mou_6	1955.61	301
31	std_og_t2m_mou_7	2112.66	301
32	std_og_t2m_mou_8	1905.81	301
33	std_og_t2f_mou_6	44.39	301
34	std_og_t2f_mou_7	43.89	301
35	std_og_t2f_mou_8	38.88	301

	Column	Outlier Threshold	Outliers replaced
36	std_og_mou_6	2744.49	301
37	std_og_mou_7	2874.65	301
38	std_og_mou_8	2800.87	301
39	isd_og_mou_6	41.25	301
40	isd_og_mou_7	40.43	301
41	isd_og_mou_8	31.24	300
42	spl_og_mou_6	71.36	301
43	spl_og_mou_7	79.87	301
44	spl_og_mou_8	74.11	301
45	og_others_6	9.31	301
46	og_others_7	0.00	164
47	og_others_8	0.00	180
48	loc_ic_t2t_mou_6	625.35	301
49	loc_ic_t2t_mou_7	648.79	301
50	loc_ic_t2t_mou_8	621.67	301
51	loc_ic_t2m_mou_6	1026.44	301
52	loc_ic_t2m_mou_7	1009.29	301
53	loc_ic_t2m_mou_8	976.09	301
54	loc_ic_t2f_mou_6	197.17	301
55	loc_ic_t2f_mou_7	205.25	301
56	loc_ic_t2f_mou_8	185.62	301
57	loc_ic_mou_6	1484.99	301
58	loc_ic_mou_7	1515.87	301
59	loc_ic_mou_8	1459.55	301
60	std_ic_t2t_mou_6	215.64	301
61	std_ic_t2t_mou_7	231.15	301
62	std_ic_t2t_mou_8	215.20	301
63	std_ic_t2m_mou_6	393.73	301
64	std_ic_t2m_mou_7	408.58	301
65	std_ic_t2m_mou_8	372.61	301
66	std_ic_t2f_mou_6	53.39	301
67	std_ic_t2f_mou_7	56.59	300
68	std_ic_t2f_mou_8	49.41	301
69	std_ic_mou_6	577.89	301
70	std_ic_mou_7	616.89	301
71	std_ic_mou_8	563.89	301
72	spl_ic_mou_6	0.68	278

	Column	Outlier Threshold	Outliers replaced
73	spl_ic_mou_7	0.51	295
74	spl_ic_mou_8	0.61	293
75	isd_ic_mou_6	239.60	301
76	isd_ic_mou_7	240.13	301
77	isd_ic_mou_8	249.89	301
78	ic_others_6	20.71	301
79	ic_others_7	25.26	301
80	ic_others_8	21.53	300
81	total_rech_num_6	48.00	283
82	total_rech_num_7	48.00	283
83	total_rech_num_8	46.00	287
84	max_rech_amt_6	1000.00	169
85	max_rech_amt_7	1000.00	204
86	max_rech_amt_8	951.00	289
87	last_day_rch_amt_6	655.00	284
88	last_day_rch_amt_7	655.00	300
89	last_day_rch_amt_8	619.00	283
90	Average_rech_amt_6n7	2216.30	301
91	delta_vol_2g	654.31	301
92	delta_vol_3g	1878.12	301
93	delta_total_og_mou	1465.10	301
94	delta_total_ic_mou	619.69	301
95	delta_vbc_3g	929.64	301
96	delta_arpu	864.34	301
97	delta_total_rech_amt	1036.40	301

In [195]:

```
categorical = data.dtypes == 'category'  
categorical_vars = data.columns[categorical].to_list()  
ind_categorical_vars = set(categorical_vars) - {'Churn'} #independent categorical  
l variables  
ind_categorical_vars
```

Out[195]:

```
{'monthly_2g_6',  
'monthly_2g_7',  
'monthly_2g_8',  
'monthly_3g_6',  
'monthly_3g_7',  
'monthly_3g_8',  
'sachet_2g_6',  
'sachet_2g_7',  
'sachet_2g_8',  
'sachet_3g_6',  
'sachet_3g_7',  
'sachet_3g_8'}
```

Grouping Categories with less Contribution

In [196]:

```
# Finding & Grouping categories with less than 1% contribution in each column in
# to "Others"
for col in ind_categorical_vars :
    category_counts = 100*data[col].value_counts(normalize=True)
    print('\n',tabulate(pd.DataFrame(category_counts), headers='keys', tablefmt
='psql'), '\n')
    low_count_categories = category_counts[category_counts <= 1].index.to_list()
    print(f'Replaced {low_count_categories} in {col} with category : Others')
    data[col].replace(low_count_categories, 'Others', inplace=True)
```

	sachet_3g_6
0	93.4091
1	4.35507
2	1.04295
3	0.396521
4	0.219919
5	0.123288
6	0.089967
7	0.0866349
8	0.0499817
9	0.0499817
10	0.0366532
11	0.0266569
15	0.0166606
12	0.0133284
19	0.0133284
13	0.00999633
14	0.00999633
18	0.00999633
23	0.00999633
16	0.00666422
22	0.00666422
29	0.00666422
28	0.00333211
17	0.00333211
21	0.00333211

Replaced [3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 12, 19, 13, 14, 18, 23, 1
6, 22, 29, 28, 17, 21] in sachet_3g_6 with category : Others

	monthly_2g_7
0	88.4876
1	10.0397
2	1.35284
3	0.0966312
4	0.0166606
5	0.00666422

Replaced [3, 4, 5] in monthly_2g_7 with category : Others

	monthly_2g_8
0	89.7604
1	9.19996
2	0.942988
3	0.0733065
4	0.0166606
5	0.00666422

Replaced [2, 3, 4, 5] in monthly_2g_8 with category : Others

	sachet_3g_8

0	94.2388
1	3.52537
2	0.839692
3	0.429842
4	0.243244
5	0.219919
6	0.0866349
7	0.0766386
8	0.0733065
9	0.0399853
12	0.0366532
13	0.0333211
10	0.0333211
11	0.0199927
14	0.0199927
15	0.0166606
16	0.00999633
17	0.00666422
18	0.00666422
20	0.00666422
21	0.00666422
23	0.00666422
38	0.00333211
19	0.00333211
25	0.00333211
27	0.00333211
29	0.00333211
30	0.00333211
41	0.00333211

Replaced [2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 10, 11, 14, 15, 16, 17, 1
 8, 20, 21, 23, 38, 19, 25, 27, 29, 30, 41] in sachet_3g_8 with categ
 ory : Others

	monthly_3g_7
0	87.8378
1	8.21699
2	2.739
3	0.689747
4	0.226584
5	0.129952
6	0.0766386
7	0.0333211
8	0.0166606
9	0.0133284
11	0.00666422
16	0.00333211
14	0.00333211
12	0.00333211
10	0.00333211

Replaced [3, 4, 5, 6, 7, 8, 9, 11, 16, 14, 12, 10] in monthly_3g_7 w
 ith category : Others

	sachet_2g_6

0	82.5631
1	7.87378
2	3.3621
3	2.0126
4	1.32951
5	0.703076
6	0.509813
7	0.356536
8	0.286562
9	0.239912
10	0.17327
12	0.146613
11	0.0999633
13	0.0566459
14	0.0533138
15	0.0433175
17	0.0366532
18	0.029989
19	0.029989
16	0.0233248
22	0.0133284
20	0.00999633
21	0.00999633
24	0.00999633
25	0.00999633
39	0.00333211
27	0.00333211
30	0.00333211
32	0.00333211
34	0.00333211
28	0
42	0

Replaced [5, 6, 7, 8, 9, 10, 12, 11, 13, 14, 15, 17, 18, 19, 16, 22, 20, 21, 24, 25, 39, 27, 30, 32, 34, 28, 42] in sachet_2g_6 with category : Others

	monthly_2g_6
0	88.9074
1	9.83306
2	1.14958
3	0.0866349
4	0.0233248

Replaced [3, 4] in monthly_2g_6 with category : Others

	sachet_2g_7
0	81.8033
1	7.24068
2	3.34877
3	1.96595
4	1.50945
5	1.20622
6	0.843024

7	0.543134
8	0.403185
10	0.239912
9	0.219919
11	0.159941
12	0.0966312
14	0.0799707
13	0.0666422
15	0.0499817
16	0.0366532
18	0.0333211
17	0.029989
20	0.0266569
19	0.0233248
21	0.00999633
26	0.00999633
27	0.00999633
22	0.00666422
23	0.00666422
30	0.00666422
42	0.00333211
24	0.00333211
25	0.00333211
29	0.00333211
32	0.00333211
35	0.00333211
48	0.00333211
28	0

Replaced [6, 7, 8, 10, 9, 11, 12, 14, 13, 15, 16, 18, 17, 20, 19, 21, 26, 27, 22, 23, 30, 42, 24, 25, 29, 32, 35, 48, 28] in sachet_2g_7 with category : Others

sachet_3g_7	
0	93.4757
1	4.10849
2	1.03962
3	0.383193
4	0.239912
5	0.219919
6	0.139949
7	0.059978
9	0.0533138
8	0.0466496
11	0.0433175
10	0.0333211
12	0.0333211
15	0.0166606
14	0.0166606
13	0.0133284
18	0.0133284
19	0.00999633
20	0.00999633
22	0.00999633
17	0.00666422
21	0.00666422
24	0.00666422
33	0.00333211

16	0.00333211
31	0.00333211
35	0.00333211

Replaced [3, 4, 5, 6, 7, 9, 8, 11, 10, 12, 15, 14, 13, 18, 19, 20, 2
2, 17, 21, 24, 33, 16, 31, 35] in sachet_3g_7 with category : Others

	monthly_3g_8
0	88.3876
1	8.00706
2	2.45243
3	0.656426
4	0.289894
5	0.0999633
6	0.0466496
7	0.029989
9	0.00999633
8	0.00999633
10	0.00666422
16	0.00333211

Replaced [3, 4, 5, 6, 7, 9, 8, 10, 16] in monthly_3g_8 with category
: Others

	monthly_3g_6
0	88.0744
1	8.4669
2	2.32248
3	0.689747
4	0.246576
5	0.106628
6	0.0366532
7	0.029989
8	0.00999633
11	0.00666422
9	0.00666422
14	0.00333211

Replaced [3, 4, 5, 6, 7, 8, 11, 9, 14] in monthly_3g_6 with category
: Others

	sachet_2g_8
0	79.7274
1	8.87008
2	3.25881
3	2.19253
4	1.81267
5	1.44947
6	0.88301
7	0.459831
8	0.313218
9	0.249908

```

| 10 |      0.169938 |
| 11 |      0.123288 |
| 12 |      0.113292 |
| 14 |      0.0766386 |
| 15 |      0.0566459 |
| 13 |      0.0499817 |
| 16 |      0.0433175 |
| 18 |      0.0266569 |
| 17 |      0.0233248 |
| 19 |      0.0233248 |
| 20 |      0.0133284 |
| 34 |      0.00666422 |
| 29 |      0.00666422 |
| 27 |      0.00666422 |
| 24 |      0.00666422 |
| 22 |      0.00666422 |
| 21 |      0.00666422 |
| 23 |      0.00333211 |
| 25 |      0.00333211 |
| 26 |      0.00333211 |
| 31 |      0.00333211 |
| 32 |      0.00333211 |
| 33 |      0.00333211 |
| 44 |      0.00333211 |
+----+-----+

```

Replaced [6, 7, 8, 9, 10, 11, 12, 14, 15, 13, 16, 18, 17, 19, 20, 3
4, 29, 27, 24, 22, 21, 23, 25, 26, 31, 32, 33, 44] in sachet_2g_8 with category : Others

Creating Dummy Variables

In [197]:

```
dummy_vars = pd.get_dummies(data[ind_categorical_vars], drop_first=False, prefix=ind_categorical_vars, prefix_sep='_')
dummy_vars.head()
```

Out[197]:

mobile_number	sachet_3g_6_0	sachet_3g_6_1	sachet_3g_6_2	sachet_3g_6_Others	monthly_2g
7000701601	1	0	0	0	
7001524846	1	0	0	0	
7002191713	1	0	0	0	
7000875565	1	0	0	0	
7000187447	1	0	0	0	

In []:

In [198]:

```
reference_cols = dummy_vars.filter(regex='.*Others$').columns.to_list() # Using
category 'Others' in each column as reference.
dummy_vars.drop(columns=reference_cols, inplace=True)
reference_cols
```

Out[198]:

```
['sachet_3g_6_Others',
 'monthly_2g_7_Others',
 'monthly_2g_8_Others',
 'sachet_3g_8_Others',
 'monthly_3g_7_Others',
 'sachet_2g_6_Others',
 'monthly_2g_6_Others',
 'sachet_2g_7_Others',
 'sachet_3g_7_Others',
 'monthly_3g_8_Others',
 'monthly_3g_6_Others',
 'sachet_2g_8_Others']
```

In [199]:

```
# concatenating dummy variables with original 'data'
data.drop(columns=ind_categorical_vars, inplace=True) # dropping original categorical columns
data = pd.concat([data, dummy_vars], axis=1)
data.head()
```

Out[199]:

mobile_number	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou_6	offnet_mou_7	offnet
7000701601	57.84	54.68	52.29	453.43	567.16	
7001524846	413.69	351.03	35.08	94.66	80.63	
7002191713	501.76	108.39	534.24	413.31	119.28	
7000875565	50.51	74.01	70.61	296.29	229.74	
7000187447	1185.91	9.28	7.79	61.64	0.00	

In [200]:

```
dummy_cols = dummy_vars.columns.to_list()
data[dummy_cols] = data[dummy_cols].astype('category')
```

In [201]:

```
data.shape
```

Out[201]:

```
(30011, 142)
```

----joint

This following section contains

- Test Train Split
- Class Imbalance
- Standardization
- Modelling
 - Model 1 : Logistic Regression with RFE & Manual Elimination (Interpretable Model)
 - Model 2 : PCA + Logistic Regression
 - Model 3 : PCA + Random Forest Classifier
 - Model 4 : PCA + XGBoost

Train-Test Split

In [3]:

```
y = data.pop('Churn') # Predicted / Target Variable  
X = data # Predictor variables
```

In [4]:

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_  
state=42)
```

Class Imbalance

In [5]:

```
y.value_counts(normalize=True).to_frame()
```

Out[5]:

Churn
0 0.913598
1 0.086402

In [6]:

```
# Ratio of classes  
class_0 = y[y == 0].count()  
class_1 = y[y == 1].count()  
  
print(f'Class Imbalance Ratio : {round(class_1/class_0,3)}')
```

Class Imbalance Ratio : 0.095

- To account for class imbalance, Synthetic Minority Class Oversampling Technique (SMOTE) could be used.

Using SMOTE

In [7]:

```
#!pip install imblearn
from imblearn.over_sampling import SMOTE
smt = SMOTE(random_state=42, k_neighbors=5)

# Resampling Train set to account for class imbalance

X_train_resampled, y_train_resampled= smt.fit_resample(X_train, y_train)
X_train_resampled.head()
```

Out[7]:

	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou_6	offnet_mou_7	offnet_mou_8	roam
0	53.01	52.64	37.48	316.01	195.74	68.36	
1	91.39	216.14	150.58	504.19	301.98	434.41	
2	11.96	14.13	0.40	1.51	0.00	0.00	
3	532.66	537.31	738.21	49.03	71.64	39.43	
4	122.68	105.51	149.33	302.23	211.44	264.11	

Standardizing Columns

In [8]:

```
# columns with numerical data
condition1 = data.dtypes == 'int'
condition2 = data.dtypes == 'float'
numerical_vars = data.columns[condition1 | condition2].to_list()
```

In [9]:

```
# Standard scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Fit and transform train set
X_train_resampled[numerical_vars] = scaler.fit_transform(X_train_resampled[numerical_vars])

# Transform test set
X_test[numerical_vars] = scaler.transform(X_test[numerical_vars])
```

In [10]:

```
# summary statistics of standardized variables
round(X_train_resampled.describe(),2)
```

Out[10]:

	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou_6	offnet_mou_7	offnet_mou_8	1
count	38374.00	38374.00	38374.00	38374.00	38374.00	38374.00	38374.00
mean	-0.00	-0.00	0.00	0.00	-0.00	-0.00	-0.00
std	1.00	1.00	1.00	1.00	1.00	1.00	1.00
min	-0.73	-0.68	-0.53	-0.94	-0.89	-0.70	-0.70
25%	-0.63	-0.60	-0.52	-0.66	-0.65	-0.66	-0.66
50%	-0.42	-0.41	-0.40	-0.33	-0.33	-0.36	-0.36
75%	0.20	0.15	0.01	0.27	0.26	0.23	0.23
max	4.09	4.46	5.67	4.02	4.45	5.24	5.24

Modelling

Model 1 : Interpretable Model : Logistic Regression

Baseline Logistic Regression Model

In [11]:

```
from sklearn.linear_model import LogisticRegression

baseline_model = LogisticRegression(random_state=100, class_weight='balanced') # `weight of class` balancing technique used
baseline_model = baseline_model.fit(X_train, y_train)

y_train_pred = baseline_model.predict_proba(X_train)[:,1]
y_test_pred = baseline_model.predict_proba(X_test)[:,1]
```

In [12]:

```
y_train_pred = pd.Series(y_train_pred, index = X_train.index, ) # converting test
and train to a series to preserve index
y_test_pred = pd.Series(y_test_pred, index = X_test.index)
```

Baseline Performance

In [13]:

```
# Function for Baseline Performance Metrics
import math
def model_metrics(matrix) :
    TN = matrix[0][0]
    TP = matrix[1][1]
    FP = matrix[0][1]
    FN = matrix[1][0]
    accuracy = round((TP + TN)/float(TP+TN+FP+FN),3)
    print('Accuracy : ',accuracy )
    sensitivity = round(TP/float(FN + TP),3)
    print('Sensitivity / True Positive Rate / Recall : ', sensitivity)
    specificity = round(TN/float(TN + FP),3)
    print('Specificity / True Negative Rate : ', specificity)
    precision = round(TP/float(TP + FP),3)
    print('Precision / Positive Predictive Value : ', precision)
    print('F1-score : ', round(2*precision*sensitivity/(precision + sensitivity),
3))
```

In [14]:

```
# Prediction at threshold of 0.5
classification_threshold = 0.5

y_train_pred_classified = y_train_pred.map(lambda x : 1 if x > classification_threshold else 0)
y_test_pred_classified = y_test_pred.map(lambda x : 1 if x > classification_threshold else 0)
```

In [15]:

```
from sklearn.metrics import confusion_matrix
train_matrix = confusion_matrix(y_train, y_train_pred_classified)
print('Confusion Matrix for train:\n', train_matrix)
test_matrix = confusion_matrix(y_test, y_test_pred_classified)
print('\nConfusion Matrix for test: \n', test_matrix)
```

Confusion Matrix for train:

```
[[16001  3186]
 [ 326  1494]]
```

Confusion Matrix for test:

```
[[6090 2141]
 [ 149  624]]
```

In [16]:

```
# Baseline Model Performance :  
  
print('Train Performance : \n')  
model_metrics(train_matrix)  
  
print('\n\nTest Performance : \n')  
model_metrics(test_matrix)
```

Train Performance :

```
Accuracy : 0.833  
Sensitivity / True Positive Rate / Recall : 0.821  
Specificity / True Negative Rate : 0.834  
Precision / Positive Predictive Value : 0.319  
F1-score : 0.459
```

Test Performance :

```
Accuracy : 0.746  
Sensitivity / True Positive Rate / Recall : 0.807  
Specificity / True Negative Rate : 0.74  
Precision / Positive Predictive Value : 0.226  
F1-score : 0.353
```

Baseline Performance - Finding Optimum Probability Cutoff

In [17]:

```
# Specificity / Sensitivity Tradeoff

# Classification at probability thresholds between 0 and 1
y_train_pred_thres = pd.DataFrame(index=X_train.index)
thresholds = [float(x)/10 for x in range(10)]

def thresholder(x, thresh) :
    if x > thresh :
        return 1
    else :
        return 0

for i in thresholds:
    y_train_pred_thres[i]= y_train_pred.map(lambda x : thresholder(x,i))
y_train_pred_thres.head()
```

Out[17]:

0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

mobile number

In [18]:

```
# # sensitivity, specificity, accuracy for each threshold
metrics_df = pd.DataFrame(columns=['sensitivity', 'specificity', 'accuracy'])

# Function for calculation of metrics for each threshold
def model_metrics_thres(matrix) :
    TN = matrix[0][0]
    TP = matrix[1][1]
    FP = matrix[0][1]
    FN = matrix[1][0]
    accuracy = round((TP + TN)/float(TP+TN+FP+FN),3)
    sensitivity = round(TP/float(FN + TP),3)
    specificity = round(TN/float(TN + FP),3)
    return sensitivity,specificity,accuracy

# generating a data frame for metrics for each threshold
for thres,column in zip(thresholds,y_train_pred_thres.columns.to_list()):
    confusion = confusion_matrix(y_train, y_train_pred_thres.loc[:,column])
    sensitivity,specificity,accuracy = model_metrics_thres(confusion)

    metrics_df = metrics_df.append({
        'sensitivity' :sensitivity,
        'specificity' : specificity,
        'accuracy' : accuracy
    }, ignore_index = True)

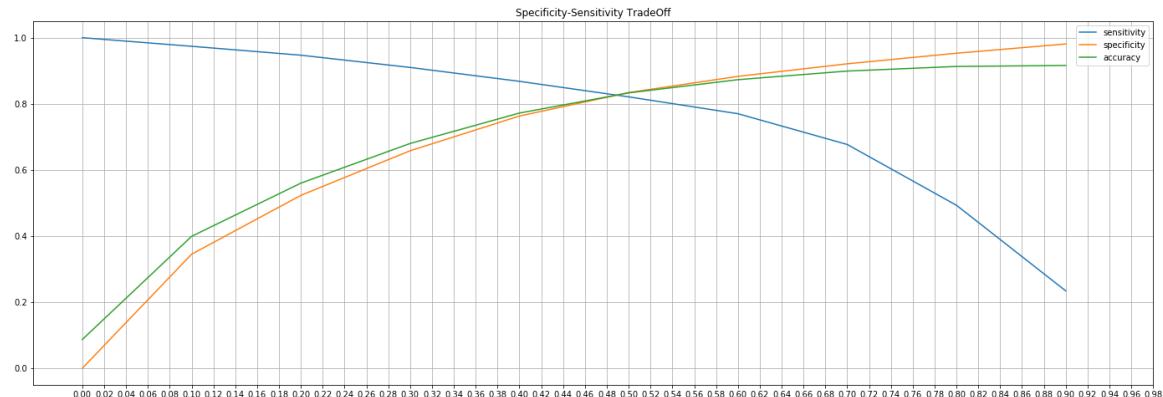
metrics_df.index = thresholds
metrics_df
```

Out[18]:

	sensitivity	specificity	accuracy
0.0	1.000	0.000	0.087
0.1	0.974	0.345	0.399
0.2	0.947	0.523	0.560
0.3	0.910	0.658	0.680
0.4	0.868	0.763	0.772
0.5	0.821	0.834	0.833
0.6	0.770	0.883	0.873
0.7	0.677	0.921	0.899
0.8	0.493	0.953	0.913
0.9	0.234	0.981	0.916

In [19]:

```
metrics_df.plot(kind='line', figsize=(24,8), grid=True, xticks=np.arange(0,1,0.02),
                title='Specificity-Sensitivity TradeOff');
```



Baseline Performance at Optimum Cutoff

In [20]:

```
optimum_cutoff = 0.49
y_train_pred_final = y_train_pred.map(lambda x : 1 if x > optimum_cutoff else 0)
y_test_pred_final = y_test_pred.map(lambda x : 1 if x > optimum_cutoff else 0)

train_matrix = confusion_matrix(y_train, y_train_pred_final)
print('Confusion Matrix for train:\n', train_matrix)
test_matrix = confusion_matrix(y_test, y_test_pred_final)
print('\nConfusion Matrix for test: \n', test_matrix)
```

Confusion Matrix for train:

```
[[15888  3299]
 [ 318   1502]]
```

Confusion Matrix for test:

```
[[1329 6902]
 [ 16  757]]
```

In [21]:

```
print('Train Performance: \n')
model_metrics(train_matrix)

print('\n\nTest Performance : \n')
model_metrics(test_matrix)
```

Train Performance:

```
Accuracy : 0.828
Sensitivity / True Positive Rate / Recall : 0.825
Specificity / True Negative Rate :  0.828
Precision / Positive Predictive Value : 0.313
F1-score : 0.454
```

Test Performance :

```
Accuracy : 0.232
Sensitivity / True Positive Rate / Recall : 0.979
Specificity / True Negative Rate :  0.161
Precision / Positive Predictive Value : 0.099
F1-score : 0.18
```

In [22]:

```
# ROC_AUC score
from sklearn.metrics import roc_auc_score
print('ROC AUC score for Train : ',round(roc_auc_score(y_train, y_train_pred), 3), '\n' )
print('ROC AUC score for Test : ',round(roc_auc_score(y_test, y_test_pred),3) )
```

ROC AUC score for Train : 0.891

ROC AUC score for Test : 0.838

Feature Selection using RFE

In [23]:

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=100 , class_weight='balanced')
rfe = RFE(lr, 15)
results = rfe.fit(X_train,y_train)
results.support_
```

Out[23]:

```
array([False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, True,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           False, False, True, False, False, False, False, False, False,  
e,           False, False, False, False, True, True, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           True, False, True, False, False, False, False, False, False,  
e,           False, False, False, False, False, False, False, False, False,  
e,           True, False, False, True, False, False, True, False, False,  
e,           False, True, False, False, True, False, False, False, False,  
e,           True, False, False, True, False, False, False, False, False,  
e,           False, False, False, True, False, False, False, False, False,  
e,           True, False, False, False, False, False, False, False, False])
```

In [24]:

Out[24]:

	Column	Rank	Support
99	sachet_3g_6_0	1	True
120	sachet_2g_7_0	1	True
102	monthly_2g_7_0	1	True
135	sachet_2g_8_0	1	True
81	total_rech_num_6	1	True
129	monthly_3g_8_0	1	True
105	monthly_2g_8_0	1	True
83	total_rech_num_8	1	True
117	monthly_2g_6_0	1	True
68	std_ic_t2f_mou_8	1	True
67	std_ic_t2f_mou_7	1	True
112	sachet_2g_6_0	1	True
109	monthly_3g_7_0	1	True
56	loc_ic_t2f_mou_8	1	True
35	std_og_t2f_mou_8	1	True
40	isd_og_mou_7	2	False
53	loc_ic_t2m_mou_8	3	False
19	loc_og_t2f_mou_7	4	False
62	std_ic_t2t_mou_8	5	False
61	std_ic_t2t_mou_7	6	False
107	sachet_3g_8_0	7	False
41	isd_og_mou_8	8	False
89	last_day_rch_amt_8	9	False
11	roam_og_mou_8	10	False
132	monthly_3g_6_0	11	False
39	isd_og_mou_6	12	False
79	ic_others_7	13	False
50	loc_ic_t2t_mou_8	14	False
7	roam_ic_mou_7	15	False
58	loc_ic_mou_7	16	False
71	std_ic_mou_8	17	False
75	isd_ic_mou_6	18	False
33	std_og_t2f_mou_6	19	False
38	std_og_mou_8	20	False
66	std_ic_t2f_mou_6	21	False
29	std_og_t2t_mou_8	22	False

		Column	Rank	Support
32	std_og_t2m_mou_8	23	False	
78	ic_others_6	24	False	
44	spl_og_mou_8	25	False	
97	delta_arpu	26	False	
85	max_rech_amt_7	27	False	
70	std_ic_mou_7	28	False	
64	std_ic_t2m_mou_7	29	False	
30	std_og_t2m_mou_6	30	False	
42	spl_og_mou_6	31	False	
27	std_og_t2t_mou_6	32	False	
18	loc_og_t2f_mou_6	33	False	
60	std_ic_t2t_mou_6	34	False	
36	std_og_mou_6	35	False	
51	loc_ic_t2m_mou_6	36	False	
15	loc_og_t2m_mou_6	37	False	
94	delta_total_og_mou	38	False	
69	std_ic_mou_6	39	False	
65	std_ic_t2m_mou_8	40	False	
2	onnet_mou_8	41	False	
55	loc_ic_t2f_mou_7	42	False	
28	std_og_t2t_mou_7	43	False	
13	loc_og_t2t_mou_7	44	False	
1	onnet_mou_7	45	False	
9	roam_og_mou_6	46	False	
21	loc_og_t2c_mou_6	47	False	
14	loc_og_t2t_mou_8	48	False	
84	max_rech_amt_6	49	False	
26	loc_og_mou_8	50	False	
8	roam_ic_mou_8	51	False	
10	roam_og_mou_7	52	False	
48	loc_ic_t2t_mou_6	53	False	
57	loc_ic_mou_6	54	False	
6	roam_ic_mou_6	55	False	
106	monthly_2g_8_1	56	False	
87	last_day_rch_amt_6	57	False	
49	loc_ic_t2t_mou_7	58	False	
98	delta_total_rech_amt	59	False	

		Column	Rank	Support
88		last_day_rch_amt_7	60	False
34		std_og_t2f_mou_7	61	False
126		sachet_3g_7_0	62	False
23		loc_og_t2c_mou_8	63	False
103		monthly_2g_7_1	64	False
118		monthly_2g_6_1	65	False
92		delta_vol_2g	66	False
16		loc_og_t2m_mou_7	67	False
4		offnet_mou_7	68	False
43		spl_og_mou_7	69	False
130		monthly_3g_8_1	70	False
20		loc_og_t2f_mou_8	71	False
17		loc_og_t2m_mou_8	72	False
63		std_ic_t2m_mou_6	73	False
93		delta_vol_3g	74	False
76		isd_ic_mou_7	75	False
24		loc_og_mou_6	76	False
12		loc_og_t2t_mou_6	77	False
54		loc_ic_t2f_mou_6	78	False
0		onnet_mou_6	79	False
3		offnet_mou_6	80	False
77		isd_ic_mou_8	81	False
5		offnet_mou_8	82	False
22		loc_og_t2c_mou_7	83	False
95		delta_total_ic_mou	84	False
52		loc_ic_t2m_mou_7	85	False
59		loc_ic_mou_8	86	False
90		aon	87	False
74		spl_ic_mou_8	88	False
136		sachet_2g_8_1	89	False
121		sachet_2g_7_1	90	False
113		sachet_2g_6_1	91	False
108		sachet_3g_8_1	92	False
80		ic_others_8	93	False
137		sachet_2g_8_2	94	False
138		sachet_2g_8_3	95	False
114		sachet_2g_6_2	96	False

	Column	Rank	Support
123	sachet_2g_7_3	97	False
133	monthly_3g_6_1	98	False
125	sachet_2g_7_5	99	False
131	monthly_3g_8_2	100	False
119	monthly_2g_6_2	101	False
25	loc_og_mou_7	102	False
104	monthly_2g_7_2	103	False
110	monthly_3g_7_1	104	False
100	sachet_3g_6_1	105	False
139	sachet_2g_8_4	106	False
134	monthly_3g_6_2	107	False
111	monthly_3g_7_2	108	False
37	std_og_mou_7	109	False
31	std_og_t2m_mou_7	110	False
140	sachet_2g_8_5	111	False
101	sachet_3g_6_2	112	False
72	spl_ic_mou_6	113	False
86	max_rech_amt_8	114	False
73	spl_ic_mou_7	115	False
96	delta_vbc_3g	116	False
82	total_rech_num_7	117	False
115	sachet_2g_6_3	118	False
124	sachet_2g_7_4	119	False
127	sachet_3g_7_1	120	False
91	Average_rech_amt_6n7	121	False
45	og_others_6	122	False
116	sachet_2g_6_4	123	False
128	sachet_3g_7_2	124	False
122	sachet_2g_7_2	125	False
47	og_others_8	126	False
46	og_others_7	127	False

In [25]:

```
# RFE Selected columns
rfe_selected_columns = rfe_support.loc[rfe_support['Rank'] == 1, 'Column'].tolist()
rfe_selected_columns
```

Out[25]:

```
['sachet_3g_6_0',
 'sachet_2g_7_0',
 'monthly_2g_7_0',
 'sachet_2g_8_0',
 'total_rech_num_6',
 'monthly_3g_8_0',
 'monthly_2g_8_0',
 'total_rech_num_8',
 'monthly_2g_6_0',
 'std_ic_t2f_mou_8',
 'std_ic_t2f_mou_7',
 'sachet_2g_6_0',
 'monthly_3g_7_0',
 'loc_ic_t2f_mou_8',
 'std_og_t2f_mou_8']
```

Logistic Regression with RFE Selected Columns

Model I

In [26]:

```
# Logistic Regression Model with RFE columns
import statsmodels.api as sm

# Note that the SMOTE resampled Train set is used with statsmodels.api.GLM since
# it doesnot support class_weight
logr = sm.GLM(y_train_resampled,(sm.add_constant(x_train_resampled[rfe_selected_
columns])), family = sm.families.Binomial())
logr_fit = logr.fit()
logr_fit.summary()
```

Out[26]:

Generalized Linear Model Regression Results

Dep. Variable:	Churn	No. Observations:	38374			
Model:	GLM	Df Residuals:	38358			
Model Family:	Binomial	Df Model:	15			
Link Function:	logit	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-19485.			
Date:	Mon, 30 Nov 2020	Deviance:	38969.			
Time:	21:57:09	Pearson chi2:	2.80e+05			
No. Iterations:	7					
Covariance Type:	nonrobust					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.2334	0.015	-15.657	0.000	-0.263	-0.204
sachet_3g_6_0	-0.0396	0.014	-2.886	0.004	-0.066	-0.013
sachet_2g_7_0	-0.0980	0.016	-6.201	0.000	-0.129	-0.067
monthly_2g_7_0	0.0096	0.016	0.594	0.552	-0.022	0.041
sachet_2g_8_0	0.0489	0.015	3.359	0.001	0.020	0.077
total_rech_num_6	0.6047	0.017	35.547	0.000	0.571	0.638
monthly_3g_8_0	0.3993	0.017	23.439	0.000	0.366	0.433
monthly_2g_8_0	0.3697	0.018	21.100	0.000	0.335	0.404
total_rech_num_8	-1.2013	0.019	-62.378	0.000	-1.239	-1.164
monthly_2g_6_0	-0.0194	0.015	-1.262	0.207	-0.050	0.011
std_ic_t2f_mou_8	-0.3364	0.026	-12.792	0.000	-0.388	-0.285
std_ic_t2f_mou_7	0.1535	0.019	8.148	0.000	0.117	0.190
sachet_2g_6_0	-0.1117	0.016	-6.847	0.000	-0.144	-0.080
monthly_3g_7_0	-0.2094	0.017	-12.602	0.000	-0.242	-0.177
loc_ic_t2f_mou_8	-1.2743	0.038	-33.599	0.000	-1.349	-1.200
std_og_t2f_mou_8	-0.2476	0.021	-11.621	0.000	-0.289	-0.206

Logistic Regression with Manual Feature Elimination

In [27]:

```
# Using P-value and vif for manual feature elimination

from statsmodels.stats.outliers_influence import variance_inflation_factor
def vif(X_train_resampled, logr_fit, selected_columns) :
    vif = pd.DataFrame()
    vif[ 'Features' ] = rfe_selected_columns
    vif[ 'VIF' ] = [variance_inflation_factor(X_train_resampled[selected_columns].values, i) for i in range(X_train_resampled[selected_columns].shape[1])]
    vif[ 'VIF' ] = round(vif[ 'VIF' ], 2)
    vif = vif.set_index( 'Features' )
    vif[ 'P-value' ] = round(logr_fit.pvalues,4)
    vif = vif.sort_values(by = [ "VIF", 'P-value' ], ascending = [ False, False ])
    return vif

vif(X_train_resampled, logr_fit, rfe_selected_columns)
```

Out[27]:

Features	VIF	P-value
std_ic_t2f_mou_8	1.66	0.0000
sachet_2g_6_0	1.64	0.0000
sachet_2g_7_0	1.57	0.0000
std_ic_t2f_mou_7	1.56	0.0000
monthly_2g_7_0	1.54	0.5524
monthly_3g_7_0	1.54	0.0000
monthly_3g_8_0	1.52	0.0000
monthly_2g_8_0	1.43	0.0000
monthly_2g_6_0	1.38	0.2069
sachet_2g_8_0	1.36	0.0008
total_rech_num_6	1.27	0.0000
total_rech_num_8	1.25	0.0000
std_og_t2f_mou_8	1.20	0.0000
sachet_3g_6_0	1.12	0.0039
loc_ic_t2f_mou_8	1.09	0.0000

- 'monthly_2g_7_0' has the very p-value. Hence, this feature could be eliminated

In [28]:

```
selected_columns = rfe_selected_columns  
selected_columns.remove('monthly_2g_7_0')  
selected_columns
```

Out[28]:

```
['sachet_3g_6_0',  
 'sachet_2g_7_0',  
 'sachet_2g_8_0',  
 'total_rech_num_6',  
 'monthly_3g_8_0',  
 'monthly_2g_8_0',  
 'total_rech_num_8',  
 'monthly_2g_6_0',  
 'std_ic_t2f_mou_8',  
 'std_ic_t2f_mou_7',  
 'sachet_2g_6_0',  
 'monthly_3g_7_0',  
 'loc_ic_t2f_mou_8',  
 'std_og_t2f_mou_8']
```

Model II

In [29]:

```
logr2 = sm.GLM(y_train_resampled,(sm.add_constant(X_train_resampled[selected_columns])), family = sm.families.Binomial())
logr2_fit = logr2.fit()
logr2_fit.summary()
```

Out[29]:

Generalized Linear Model Regression Results

Dep. Variable:	Churn	No. Observations:	38374			
Model:	GLM	Df Residuals:	38359			
Model Family:	Binomial	Df Model:	14			
Link Function:	logit	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-19485.			
Date:	Mon, 30 Nov 2020	Deviance:	38970.			
Time:	21:57:09	Pearson chi2:	2.80e+05			
No. Iterations:	7					
Covariance Type:	nonrobust					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.2335	0.015	-15.662	0.000	-0.263	-0.204
sachet_3g_6_0	-0.0395	0.014	-2.881	0.004	-0.066	-0.013
sachet_2g_7_0	-0.0982	0.016	-6.217	0.000	-0.129	-0.067
sachet_2g_8_0	0.0491	0.015	3.372	0.001	0.021	0.078
total_rech_num_6	0.6049	0.017	35.566	0.000	0.572	0.638
monthly_3g_8_0	0.4000	0.017	23.521	0.000	0.367	0.433
monthly_2g_8_0	0.3733	0.016	22.696	0.000	0.341	0.406
total_rech_num_8	-1.2012	0.019	-62.375	0.000	-1.239	-1.163
monthly_2g_6_0	-0.0163	0.014	-1.128	0.259	-0.045	0.012
std_ic_t2f_mou_8	-0.3361	0.026	-12.784	0.000	-0.388	-0.285
std_ic_t2f_mou_7	0.1532	0.019	8.136	0.000	0.116	0.190
sachet_2g_6_0	-0.1111	0.016	-6.823	0.000	-0.143	-0.079
monthly_3g_7_0	-0.2098	0.017	-12.633	0.000	-0.242	-0.177
loc_ic_t2f_mou_8	-1.2749	0.038	-33.622	0.000	-1.349	-1.201
std_og_t2f_mou_8	-0.2476	0.021	-11.620	0.000	-0.289	-0.206

In [30]:

```
# vif and p-values
vif(X_train_resampled, logr2_fit, selected_columns)
```

Out[30]:

Features	VIF	P-value
std_ic_t2f_mou_8	1.66	0.0000
sachet_2g_6_0	1.63	0.0000
sachet_2g_7_0	1.57	0.0000
std_ic_t2f_mou_7	1.56	0.0000
monthly_3g_7_0	1.54	0.0000
monthly_3g_8_0	1.52	0.0000
sachet_2g_8_0	1.36	0.0007
total_rech_num_6	1.27	0.0000
total_rech_num_8	1.25	0.0000
monthly_2g_8_0	1.23	0.0000
monthly_2g_6_0	1.21	0.2595
std_og_t2f_mou_8	1.20	0.0000
sachet_3g_6_0	1.12	0.0040
loc_ic_t2f_mou_8	1.09	0.0000

- 'monthly_2g_6_0' has very high p-value. Hence, this feature could be eliminated

In [31]:

```
selected_columns.remove('monthly_2g_6_0')
selected_columns
```

Out[31]:

```
['sachet_3g_6_0',
 'sachet_2g_7_0',
 'sachet_2g_8_0',
 'total_rech_num_6',
 'monthly_3g_8_0',
 'monthly_2g_8_0',
 'total_rech_num_8',
 'std_ic_t2f_mou_8',
 'std_ic_t2f_mou_7',
 'sachet_2g_6_0',
 'monthly_3g_7_0',
 'loc_ic_t2f_mou_8',
 'std_og_t2f_mou_8']
```

Model III

In [32]:

```
logr3 = sm.GLM(y_train_resampled,(sm.add_constant(X_train_resampled[selected_columns])), family = sm.families.Binomial())
logr3_fit = logr3.fit()
logr3_fit.summary()
```

Out[32]:

Generalized Linear Model Regression Results

Dep. Variable:	Churn	No. Observations:	38374			
Model:	GLM	Df Residuals:	38360			
Model Family:	Binomial	Df Model:	13			
Link Function:	logit	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-19486.			
Date:	Mon, 30 Nov 2020	Deviance:	38971.			
Time:	21:57:10	Pearson chi2:	2.79e+05			
No. Iterations:	7					
Covariance Type:	nonrobust					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.2336	0.015	-15.667	0.000	-0.263	-0.204
sachet_3g_6_0	-0.0399	0.014	-2.916	0.004	-0.067	-0.013
sachet_2g_7_0	-0.0987	0.016	-6.249	0.000	-0.130	-0.068
sachet_2g_8_0	0.0488	0.015	3.354	0.001	0.020	0.077
total_rech_num_6	0.6053	0.017	35.581	0.000	0.572	0.639
monthly_3g_8_0	0.3994	0.017	23.494	0.000	0.366	0.433
monthly_2g_8_0	0.3666	0.015	23.953	0.000	0.337	0.397
total_rech_num_8	-1.2033	0.019	-62.720	0.000	-1.241	-1.166
std_ic_t2f_mou_8	-0.3363	0.026	-12.788	0.000	-0.388	-0.285
std_ic_t2f_mou_7	0.1532	0.019	8.137	0.000	0.116	0.190
sachet_2g_6_0	-0.1108	0.016	-6.810	0.000	-0.143	-0.079
monthly_3g_7_0	-0.2099	0.017	-12.640	0.000	-0.242	-0.177
loc_ic_t2f_mou_8	-1.2736	0.038	-33.621	0.000	-1.348	-1.199
std_og_t2f_mou_8	-0.2474	0.021	-11.617	0.000	-0.289	-0.206

In [33]:

```
# vif and p-values
vif(X_train_resampled, logr3_fit, selected_columns)
```

Out[33]:

Features	VIF	P-value
std_ic_t2f_mou_8	1.66	0.0000
sachet_2g_6_0	1.63	0.0000
sachet_2g_7_0	1.57	0.0000
std_ic_t2f_mou_7	1.56	0.0000
monthly_3g_7_0	1.54	0.0000
monthly_3g_8_0	1.52	0.0000
sachet_2g_8_0	1.36	0.0008
total_rech_num_6	1.27	0.0000
total_rech_num_8	1.24	0.0000
std_og_t2f_mou_8	1.20	0.0000
sachet_3g_6_0	1.12	0.0035
loc_ic_t2f_mou_8	1.09	0.0000
monthly_2g_8_0	1.03	0.0000

- All features have low p-values(<0.05) and VIF (<5)
- This model could be used as the interpretable logistic regression model.

Final Logistic Regression Model with RFE and Manual Elimination

In [34]:

```
logr3_fit.summary()
```

Out[34]:

Generalized Linear Model Regression Results

Dep. Variable:	Churn	No. Observations:	38374			
Model:	GLM	Df Residuals:	38360			
Model Family:	Binomial	Df Model:	13			
Link Function:	logit	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-19486.			
Date:	Mon, 30 Nov 2020	Deviance:	38971.			
Time:	21:57:10	Pearson chi2:	2.79e+05			
No. Iterations:	7					
Covariance Type:	nonrobust					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.2336	0.015	-15.667	0.000	-0.263	-0.204
sachet_3g_6_0	-0.0399	0.014	-2.916	0.004	-0.067	-0.013
sachet_2g_7_0	-0.0987	0.016	-6.249	0.000	-0.130	-0.068
sachet_2g_8_0	0.0488	0.015	3.354	0.001	0.020	0.077
total_rech_num_6	0.6053	0.017	35.581	0.000	0.572	0.639
monthly_3g_8_0	0.3994	0.017	23.494	0.000	0.366	0.433
monthly_2g_8_0	0.3666	0.015	23.953	0.000	0.337	0.397
total_rech_num_8	-1.2033	0.019	-62.720	0.000	-1.241	-1.166
std_ic_t2f_mou_8	-0.3363	0.026	-12.788	0.000	-0.388	-0.285
std_ic_t2f_mou_7	0.1532	0.019	8.137	0.000	0.116	0.190
sachet_2g_6_0	-0.1108	0.016	-6.810	0.000	-0.143	-0.079
monthly_3g_7_0	-0.2099	0.017	-12.640	0.000	-0.242	-0.177
loc_ic_t2f_mou_8	-1.2736	0.038	-33.621	0.000	-1.348	-1.199
std_og_t2f_mou_8	-0.2474	0.021	-11.617	0.000	-0.289	-0.206

In [35]:

```
selected_columns
```

Out[35]:

```
['sachet_3g_6_0',
 'sachet_2g_7_0',
 'sachet_2g_8_0',
 'total_rech_num_6',
 'monthly_3g_8_0',
 'monthly_2g_8_0',
 'total_rech_num_8',
 'std_ic_t2f_mou_8',
 'std_ic_t2f_mou_7',
 'sachet_2g_6_0',
 'monthly_3g_7_0',
 'loc_ic_t2f_mou_8',
 'std_og_t2f_mou_8']
```

In [36]:

```
# Prediction
y_train_pred_lr = logr3_fit.predict(sm.add_constant(X_train_resampled[selected_columns]))
y_train_pred_lr.head()
```

Out[36]:

```
0    0.118916
1    0.343873
2    0.381230
3    0.015277
4    0.001595
dtype: float64
```

In [37]:

```
y_test_pred_lr = logr3_fit.predict(sm.add_constant(X_test[selected_columns]))
y_test_pred_lr.head()
```

Out[37]:

```
mobile_number
7002242818    0.013556
7000517161    0.903162
7002162382    0.247123
7002152271    0.330787
7002058655    0.056105
dtype: float64
```

Performance

Finding Optimum Probability Cutoff

In [38]:

```
# Specificity / Sensitivity Tradeoff

# Classification at probability thresholds between 0 and 1
y_train_pred_thres = pd.DataFrame(index=X_train_resampled.index)
thresholds = [float(x)/10 for x in range(10)]

def thresholder(x, thresh) :
    if x > thresh :
        return 1
    else :
        return 0

for i in thresholds:
    y_train_pred_thres[i]= y_train_pred_lr.map(lambda x : thresholder(x,i))
y_train_pred_thres.head()
```

Out[38]:

In [39]:

```
# DataFrame for Performance metrics at each threshold

logr_metrics_df = pd.DataFrame(columns=['sensitivity', 'specificity', 'accuracy'])

for thres, column in zip(thresholds, y_train_pred_thres.columns.to_list()):
    confusion = confusion_matrix(y_train_resampled, y_train_pred_thres.loc[:, column])
    sensitivity, specificity, accuracy = model_metrics_thres(confusion)
    logr_metrics_df = logr_metrics_df.append({
        'sensitivity' : sensitivity,
        'specificity' : specificity,
        'accuracy' : accuracy
    }, ignore_index = True)

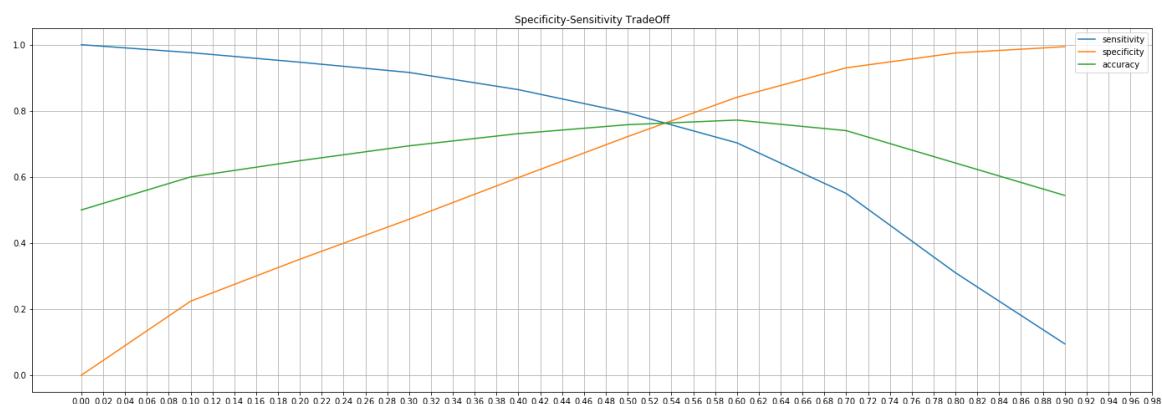
logr_metrics_df.index = thresholds
logr_metrics_df
```

Out[39]:

	sensitivity	specificity	accuracy
0.0	1.000	0.000	0.500
0.1	0.976	0.224	0.600
0.2	0.947	0.351	0.649
0.3	0.916	0.472	0.694
0.4	0.864	0.598	0.731
0.5	0.794	0.722	0.758
0.6	0.703	0.841	0.772
0.7	0.550	0.930	0.740
0.8	0.310	0.975	0.642
0.9	0.095	0.994	0.544

In [40]:

```
logr_metrics_df.plot(kind='line', figsize=(24,8), grid=True, xticks=np.arange(0, 1, 0.02),
                      title='Specificity-Sensitivity TradeOff');
```



- The optimum probability cutoff for Logistic regression model is 0.53

In [41]:

```
optimum_cutoff = 0.53
y_train_pred_lr_final = y_train_pred_lr.map(lambda x : 1 if x > optimum_cutoff else 0)
y_test_pred_lr_final = y_test_pred_lr.map(lambda x : 1 if x > optimum_cutoff else 0)

train_matrix = confusion_matrix(y_train_resampled, y_train_pred_lr_final)
print('Confusion Matrix for train:\n', train_matrix)
test_matrix = confusion_matrix(y_test, y_test_pred_lr_final)
print('\nConfusion Matrix for test: \n', test_matrix)
```

Confusion Matrix for train:

```
[[14531  4656]
 [ 4411 14776]]
```

Confusion Matrix for test:

```
[[6313 1918]
 [ 191  582]]
```

In [42]:

```
print('Train Performance: \n')
model_metrics(train_matrix)

print('\n\nTest Performance : \n')
model_metrics(test_matrix)
```

Train Performance:

```
Accuracy : 0.764
Sensitivity / True Positive Rate / Recall : 0.77
Specificity / True Negative Rate : 0.757
Precision / Positive Predictive Value : 0.76
F1-score : 0.765
```

Test Performance :

```
Accuracy : 0.766
Sensitivity / True Positive Rate / Recall : 0.753
Specificity / True Negative Rate : 0.767
Precision / Positive Predictive Value : 0.233
F1-score : 0.356
```

In [43]:

```
# ROC_AUC score
print('ROC AUC score for Train : ',round(roc_auc_score(y_train_resampled, y_train_pred_lr),3), '\n')
print('ROC AUC score for Test : ',round(roc_auc_score(y_test, y_test_pred_lr),3))
```

ROC AUC score for Train : 0.843

ROC AUC score for Test : 0.828

Model 1 : Logistic Regression (Interpretable Model Summary)

In [44]:

```
lr_summary_html = logr3_fit.summary().tables[1].as_html()
lr_results = pd.read_html(lr_summary_html, header=0, index_col=0)[0]
coef_column = lr_results.columns[0]
print('Most important predictors of Churn , in order of importance and their coefficients are as follows : \n')
lr_results.sort_values(by=coef_column, key=lambda x: abs(x), ascending=False)[ 'coef' ]
```

Most important predictors of Churn , in order of importance and their coefficients are as follows :

Out[44]:

```
loc_ic_t2f_mou_8      -1.2736
total_rech_num_8      -1.2033
total_rech_num_6       0.6053
monthly_3g_8_0         0.3994
monthly_2g_8_0         0.3666
std_ic_t2f_mou_8      -0.3363
std_og_t2f_mou_8      -0.2474
const                  -0.2336
monthly_3g_7_0         -0.2099
std_ic_t2f_mou_7       0.1532
sachet_2g_6_0          -0.1108
sachet_2g_7_0          -0.0987
sachet_2g_8_0          0.0488
sachet_3g_6_0          -0.0399
Name: coef, dtype: float64
```

- The above model could be used as the interpretable model for predicting telecom churn.

PCA

In [45]:

```
from sklearn.decomposition import PCA
pca = PCA(random_state = 42)
pca.fit(X_train) # note that pca is fit on original train set instead of resampled train set.
pca.components_
```

Out[45]:

```
array([[ 1.64887430e-01,  1.93987506e-01,  1.67239205e-01, ...,
       1.43967238e-06, -1.55704675e-06, -1.88892194e-06],
       [ 6.48591961e-02,  9.55966684e-02,  1.20775174e-01, ...,
       -2.12841595e-06, -1.47944145e-06, -3.90881587e-07],
       [ 2.38415388e-01,  2.73645507e-01,  2.38436263e-01, ...,
      -1.25598531e-06, -4.37900299e-07,  6.19889336e-07],
       ...,
       [ 1.68015588e-06,  1.93600851e-06, -1.82065762e-06, ...,
       4.25473944e-03,  2.56738368e-03,  3.51118176e-03],
       [ 0.00000000e+00, -1.11533905e-16,  1.57807487e-16, ...,
       1.73764144e-15,  6.22907679e-16,  1.45339158e-16],
       [ 0.00000000e+00,  4.98537742e-16, -6.02718139e-16, ...,
      1.27514583e-15,  1.25772226e-15,  3.41773342e-16]])
```

In [46]:

```
pca.explained_variance_ratio_
```

Out[46]:

```
array([2.72067612e-01, 1.62438240e-01, 1.20827535e-01, 1.06070063e-0
1,
      9.11349433e-02, 4.77504400e-02, 2.63978655e-02, 2.56843982e-0
2,
      1.91789343e-02, 1.68045932e-02, 1.55523468e-02, 1.31676589e-0
2,
      1.04552128e-02, 7.72970448e-03, 7.22746863e-03, 6.14494838e-0
3,
      5.62073089e-03, 5.44579273e-03, 4.59009989e-03, 4.38488162e-0
3,
      3.46703626e-03, 3.27941490e-03, 2.78099200e-03, 2.13444270e-0
3,
      2.07542043e-03, 1.89794720e-03, 1.41383936e-03, 1.30240760e-0
3,
      1.15369576e-03, 1.05262500e-03, 9.64293417e-04, 9.16686049e-0
4,
      8.84067044e-04, 7.62966236e-04, 6.61794767e-04, 5.69667265e-0
4,
      5.12585166e-04, 5.04441248e-04, 4.82396680e-04, 4.46889495e-0
4,
      4.36441254e-04, 4.10389488e-04, 3.51844810e-04, 3.12626195e-0
4,
      2.51673027e-04, 2.34723896e-04, 1.96950034e-04, 1.71296745e-0
4,
      1.59882693e-04, 1.48330353e-04, 1.45919483e-04, 1.08583729e-0
4,
      1.04038518e-04, 8.90621848e-05, 8.53009223e-05, 7.60704088e-0
5,
      7.57150133e-05, 6.16615717e-05, 6.07777411e-05, 5.70517541e-0
5,
      5.36161089e-05, 5.28495367e-05, 5.14887086e-05, 4.73768570e-0
5,
      4.71283394e-05, 4.11523975e-05, 4.10392906e-05, 2.86090257e-0
5,
      2.19793282e-05, 1.58203581e-05, 1.50969788e-05, 1.42865579e-0
5,
      1.34537530e-05, 1.33026062e-05, 1.10239870e-05, 8.27539516e-0
6,
      7.55845974e-06, 6.45372276e-06, 6.22570067e-06, 3.42288900e-0
6,
      3.20804681e-06, 3.09270863e-06, 2.86608967e-06, 2.44898003e-0
6,
      2.08230568e-06, 1.85144734e-06, 1.64714248e-06, 1.45630245e-0
6,
      1.35265729e-06, 1.05472047e-06, 9.89133015e-07, 8.65864423e-0
7,
      7.45065121e-07, 3.66727807e-07, 6.49277820e-08, 6.13357428e-0
8,
      4.35995018e-08, 2.28152900e-08, 2.00441141e-08, 1.84235145e-0
8,
      1.66102335e-08, 1.47870989e-08, 1.23390691e-08, 1.12094165e-0
8,
      1.09702422e-08, 9.51924270e-09, 8.61596309e-09, 7.38051070e-0
9,
      7.15370081e-09, 6.29095319e-09, 5.00739371e-09, 4.68791660e-0
9,
      4.23376173e-09, 4.04558169e-09, 3.75847771e-09, 3.71213838e-0
9,
      3.32806929e-09, 3.23527525e-09, 3.12734302e-09, 2.82062311e-0
9,
      2.72602311e-09, 2.66103741e-09, 2.46562734e-09, 2.20243536e-0
```

```

9,
  2.15044476e-09, 1.59498492e-09, 1.47087974e-09, 1.06159357e-0
9,
  9.33938436e-10, 8.10080735e-10, 8.04656028e-10, 6.12994365e-1
0,
  4.82074297e-10, 4.02577318e-10, 3.58059984e-10, 3.28374076e-1
0,
  3.03687605e-10, 7.12091816e-11, 6.13978255e-11, 1.04375208e-3
3,
  1.04375208e-33])

```

Scree Plot

In [47]:

```

var_cum = np.cumsum(pca.explained_variance_ratio_)
plt.figure(figsize=(20,8))
sns.set_style('darkgrid')
sns.lineplot(np.arange(1,len(var_cum) + 1), var_cum)
plt.xticks(np.arange(0,140,5))
plt.axhline(0.95,color='r')
plt.axhline(1.0,color='r')
plt.axvline(15,color='b')
plt.axvline(45,color='b')
plt.text(10,0.96,'0.95')

plt.title('Scree Plot of Telecom Churn Train Set');

```



- From the above scree plot, it is clear that 95% of variance in the train set can be explained by first 16 principal components and 100% of variance is explained by the first 45 principal components.

In [48]:

```
# Perform PCA using the first 45 components
pca_final = PCA(n_components=45, random_state=42)
transformed_data = pca_final.fit_transform(X_train)
X_train_pca = pd.DataFrame(transformed_data, columns=["PC_" + str(x) for x in range(1,46)], index = X_train.index)
data_train_pca = pd.concat([X_train_pca, y_train], axis=1)

data_train_pca.head()
```

Out[48]:

mobile_number	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6
7000166926	-907.572208	-342.923676	13.094442	58.813506	-95.616159	-1050.5352
7001343085	573.898045	-902.385767	-424.839214	-331.153508	-148.987005	-36.9557
7001863283	-1538.198366	514.032564	846.865497	57.032319	-1126.228705	-84.2095
7002275981	486.830772	-224.929803	1130.460535	-496.189015	6.009139	81.1068
7001086221	-1420.949314	794.071749	99.221352	155.118564	145.349456	784.72358

In [49]:

```
## Plotting principal components
sns.pairplot(data=data_train_pca, x_vars=["PC_1"], y_vars=["PC_2"], hue = "Churn", size=8);
```



Model 2 : PCA + Logistic Regression Model

In [50]:

```
# X,y Split
y_train_pca = data_train_pca.pop('Churn')
X_train_pca = data_train_pca

# Transforming test set with pca ( 45 components)
X_test_pca = pca_final.transform(X_test)

# Logistic Regression
lr_pca = LogisticRegression(random_state=100, class_weight='balanced')
lr_pca.fit(X_train_pca,y_train_pca )
```

Out[50]:

```
LogisticRegression(class_weight='balanced', random_state=100)
```

In [51]:

```
# y_train predictions
y_train_pred_lr_pca = lr_pca.predict(X_train_pca)
y_train_pred_lr_pca[:5]
```

Out[51]:

```
array([1, 0, 0, 0, 0])
```

In [52]:

```
# Test Prediction
X_test_pca = pca_final.transform(X_test)
y_test_pred_lr_pca = lr_pca.predict(X_test_pca)
y_test_pred_lr_pca[:5]
```

Out[52]:

```
array([1, 1, 1, 1, 1])
```

Baseline Performance

In [53]:

```
train_matrix = confusion_matrix(y_train, y_train_pred_lr_pca)
test_matrix = confusion_matrix(y_test, y_test_pred_lr_pca)

print('Train Performance :\n')
model_metrics(train_matrix)

print('\nTest Performance :\n')
model_metrics(test_matrix)
```

Train Performance :

```
Accuracy : 0.645
Sensitivity / True Positive Rate / Recall : 0.905
Specificity / True Negative Rate : 0.62
Precision / Positive Predictive Value : 0.184
F1-score : 0.306
```

Test Performance :

```
Accuracy : 0.086
Sensitivity / True Positive Rate / Recall : 1.0
Specificity / True Negative Rate : 0.0
Precision / Positive Predictive Value : 0.086
F1-score : 0.158
```

Hyperparameter Tuning

In [54]:

```
# Creating a Logistic regression model using pca transformed train set
from sklearn.pipeline import Pipeline
lr_pca = LogisticRegression(random_state=100, class_weight='balanced')
```

In [55]:

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV, StratifiedKFold
params = {
    'penalty' : ['l1', 'l2', 'none'],
    'C' : [0, 1, 2, 3, 4, 5, 10, 50]
}
folds = StratifiedKFold(n_splits=4, shuffle=True, random_state=100)

search = GridSearchCV(cv=folds, estimator = lr_pca, param_grid=params, scoring='roc_auc', verbose=True, n_jobs=-1)
search.fit(X_train_pca, y_train_pca)
```

Fitting 4 folds for each of 24 candidates, totalling 96 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 42 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 96 out of 96 | elapsed:    6.9s finished
```

Out[55]:

```
GridSearchCV(cv=StratifiedKFold(n_splits=4, random_state=100, shuffle=True),
             estimator=LogisticRegression(class_weight='balanced',
                                           random_state=100),
             n_jobs=-1,
             param_grid={'C': [0, 1, 2, 3, 4, 5, 10, 50],
                         'penalty': ['l1', 'l2', 'none']},
             scoring='roc_auc', verbose=True)
```

In [56]:

```
# Optimum Hyperparameters
print('Best ROC-AUC score :', search.best_score_)
print('Best Parameters :', search.best_params_)
```

```
Best ROC-AUC score : 0.8763924253372933
Best Parameters : {'C': 0, 'penalty': 'none'}
```

In [57]:

```
# Modelling using the best LR-PCA estimator
lr_pca_best = search.best_estimator_
lr_pca_best_fit = lr_pca_best.fit(X_train_pca, y_train_pca)

# Prediction on Train set
y_train_pred_lr_pca_best = lr_pca_best_fit.predict(X_train_pca)
y_train_pred_lr_pca_best[:5]
```

Out[57]:

```
array([1, 1, 0, 0, 0])
```

In [58]:

```
# Prediction on test set
y_test_pred_lr_pca_best = lr_pca_best_fit.predict(X_test_pca)
y_test_pred_lr_pca_best[:5]
```

Out[58]:

```
array([1, 1, 1, 1, 1])
```

In [59]:

```
## Model Performance after Hyper Parameter Tuning

train_matrix = confusion_matrix(y_train, y_train_pred_lr_pca_best)
test_matrix = confusion_matrix(y_test, y_test_pred_lr_pca_best)

print('Train Performance :\n')
model_metrics(train_matrix)

print('\nTest Performance :\n')
model_metrics(test_matrix)
```

Train Performance :

```
Accuracy : 0.627
Sensitivity / True Positive Rate / Recall : 0.918
Specificity / True Negative Rate : 0.599
Precision / Positive Predictive Value : 0.179
F1-score : 0.3
```

Test Performance :

```
Accuracy : 0.086
Sensitivity / True Positive Rate / Recall : 1.0
Specificity / True Negative Rate : 0.0
Precision / Positive Predictive Value : 0.086
F1-score : 0.158
```

Model 3 : PCA + Random Forest

In [60]:

```
from sklearn.ensemble import RandomForestClassifier

# creating a random forest classifier using pca output

pca_rf = RandomForestClassifier(random_state=42, class_weight= {0 : class_1/(class_0 + class_1) , 1 : class_0/(class_0 + class_1) } , oob_score=True, n_jobs=-1,
verbose=1)
pca_rf
```

Out[60]:

```
RandomForestClassifier(class_weight={0: 0.08640165272733331,
                                         1: 0.9135983472726666},
                           n_jobs=-1, oob_score=True, random_state=42, v
erbose=1)
```

In [68]:

```
# Hyper parameter Tuning
params = {
    'n_estimators' : [30,40,50,100],
    'max_depth' : [3,4,5,6,7],
    'min_samples_leaf' : [15,20,25,30]
}
folds = StratifiedKFold(n_splits=4, shuffle=True, random_state=42)
pca_rf_model_search = GridSearchCV(estimator=pca_rf, param_grid=params,
                                    cv=folds, scoring='roc_auc', verbose=True, n_
jobs=-1 )

pca_rf_model_search.fit(X_train_pca, y_train)
```

Fitting 4 folds for each of 80 candidates, totalling 320 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   23.2s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done 320 out of 320 | elapsed:  5.5min finish
ed
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurre
nt workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:     1.2s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:     2.6s finish
ed
```

Out[68]:

```
GridSearchCV(cv=StratifiedKFold(n_splits=4, random_state=42, shuffle
=True),
             estimator=RandomForestClassifier(class_weight={0: 0.086
40165272733331,
                                                1: 0.913
5983472726666},
                                                n_jobs=-1, oob_score=T
rue,
                                                random_state=42, verbo
se=1),
             n_jobs=-1,
             param_grid={'max_depth': [3, 4, 5, 6, 7],
                         'min_samples_leaf': [15, 20, 25, 30],
                         'n_estimators': [30, 40, 50, 100]},
             scoring='roc_auc', verbose=True)
```

In [69]:

```
# Optimum Hyperparameters
print('Best ROC-AUC score :', pca_rf_model_search.best_score_)
print('Best Parameters :', pca_rf_model_search.best_params_)
```

```
Best ROC-AUC score : 0.8861621751601011
Best Parameters : {'max_depth': 7, 'min_samples_leaf': 20, 'n_estima
tors': 100}
```

In [70]:

```
# Modelling using the best PCA-RandomForest Estimator
pca_rf_best = pca_rf_model_search.best_estimator_
pca_rf_best_fit = pca_rf_best.fit(X_train_pca, y_train)

# Prediction on Train set
y_train_pred_pca_rf_best = pca_rf_best_fit.predict(X_train_pca)
y_train_pred_pca_rf_best[:5]
```

[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 42 tasks | elapsed: 1.1s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 2.7s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done 42 tasks | elapsed: 0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed: 0.1s finished

Out[70]:

```
array([0, 0, 0, 0, 0])
```

In [71]:

```
# Prediction on test set
y_test_pred_pca_rf_best = pca_rf_best_fit.predict(X_test_pca)
y_test_pred_pca_rf_best[:5]
```

[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done 42 tasks | elapsed: 0.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed: 0.1s finished

Out[71]:

```
array([0, 0, 0, 0, 0])
```

In [72]:

```
## PCA - RandomForest Model Performance - Hyper Parameter Tuned

train_matrix = confusion_matrix(y_train, y_train_pred_pca_rf_best)
test_matrix = confusion_matrix(y_test, y_test_pred_pca_rf_best)

print('Train Performance :\n')
model_metrics(train_matrix)

print('\nTest Performance :\n')
model_metrics(test_matrix)
```

Train Performance :

```
Accuracy : 0.882
Sensitivity / True Positive Rate / Recall : 0.816
Specificity / True Negative Rate : 0.888
Precision / Positive Predictive Value : 0.408
F1-score : 0.544
```

Test Performance :

```
Accuracy : 0.86
Sensitivity / True Positive Rate / Recall : 0.80
Specificity / True Negative Rate : 0.78
Precision / Positive Predictive Value : 0.37
F1-score : 0.51
```

In [67]:

```
## out of bag error
pca_rf_best_fit.oob_score_
```

Out[67]:

```
0.8625220164707003
```

Model 4 : PCA + XGBoost

In [74]:

```
import xgboost as xgb
pca_xgb = xgb.XGBClassifier(random_state=42, scale_pos_weight= class_0/class_1 ,
                             tree_method='hist',
                             objective='binary:logistic',
                             )    # scale_pos_weight takes care of class imbalance
pca_xgb.fit(X_train_pca, y_train)
```

Out[74]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=10.573852680293097,
              subsample=1, tree_method='hist', validate_parameters=1,
              verbosity=None)
```

In [75]:

```
print('Baseline Train AUC Score')
roc_auc_score(y_train, pca_xgb.predict_proba(X_train_pca)[:, 1])
```

Baseline Train AUC Score

Out[75]:

0.9999996277241286

In [76]:

```
print('Baseline Test AUC Score')
roc_auc_score(y_test, pca_xgb.predict_proba(X_test_pca)[:, 1])
```

Baseline Test AUC Score

Out[76]:

0.46093390352284136

In [77]:

```
## Hyper parameter Tuning
parameters = {
    'learning_rate': [0.1, 0.2, 0.3],
    'gamma' : [10,20,50],
    'max_depth': [2,3,4],
    'min_child_weight': [25,50],
    'n_estimators': [150,200,500]}
pca_xgb_search = GridSearchCV(estimator=pca_xgb , param_grid=parameters,scoring
='roc_auc', cv=folds, n_jobs=-1, verbose=1)
pca_xgb_search.fit(X_train_pca, y_train)
```

Fitting 4 folds for each of 162 candidates, totalling 648 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   28.3s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  4.8min
[Parallel(n_jobs=-1)]: Done 648 out of 648 | elapsed:  8.0min finished
```

Out[77]:

```
GridSearchCV(cv=StratifiedKFold(n_splits=4, random_state=42, shuffle
=True),
            estimator=XGBClassifier(base_score=0.5, booster='gbtre
e',
                                    colsample_bylevel=1, colsample_
bynode=1,
                                    colsample_bytree=1, gamma=0, gp
u_id=-1,
                                    importance_type='gain',
                                    interaction_constraints='',
                                    learning_rate=0.300000012,
                                    max_delta_step=0, max_depth=6,
                                    min_child_weight=1, missing=na
n,
                                    monotone_...
                                    n_estimators=100, n_jobs=0,
                                    num_parallel_tree=1, random_st
te=42,
                                    reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=10.57385268029
3097,
                                    subsample=1, tree_method='his
t',
                                    validate_parameters=1, verbosit
y=None),
            n_jobs=-1,
            param_grid={'gamma': [10, 20, 50],
                        'learning_rate': [0.1, 0.2, 0.3],
                        'max_depth': [2, 3, 4], 'min_child_weight':
[25, 50],
                        'n_estimators': [150, 200, 500]},
            scoring='roc_auc', verbose=1)
```

In [78]:

```
# Optimum Hyperparameters
print('Best ROC-AUC score :', pca_xgb_search.best_score_)
print('Best Parameters :', pca_xgb_search.best_params_)
```

```
Best ROC-AUC score : 0.8955777259491308
Best Parameters : {'gamma': 10, 'learning_rate': 0.1, 'max_depth': 2, 'min_child_weight': 50, 'n_estimators': 500}
```

In [79]:

```
# Modelling using the best PCA-XGBoost Estimator
pca_xgb_best = pca_xgb_search.best_estimator_
pca_xgb_best_fit = pca_xgb_best.fit(X_train_pca, y_train)

# Prediction on Train set
y_train_pred_pca_xgb_best = pca_xgb_best_fit.predict(X_train_pca)
y_train_pred_pca_xgb_best[:5]
```

Out[79]:

```
array([0, 0, 0, 0, 0])
```

In [84]:

```
X_train_pca.head()
```

Out[84]:

mobile_number	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6
7000166926	-907.572208	-342.923676	13.094442	58.813506	-95.616159	-1050.5352
7001343085	573.898045	-902.385767	-424.839214	-331.153508	-148.987005	-36.9557
7001863283	-1538.198366	514.032564	846.865497	57.032319	-1126.228705	-84.2095
7002275981	486.830772	-224.929803	1130.460535	-496.189015	6.009139	81.1068
7001086221	-1420.949314	794.071749	99.221352	155.118564	145.349456	784.72358

In [85]:

```
# Prediction on test set
X_test_pca = pca_final.transform(X_test)
X_test_pca = pd.DataFrame(X_test_pca, index=X_test.index, columns = X_train_pca.columns)
y_test_pred_pca_xgb_best = pca_xgb_best_fit.predict(X_test_pca)
y_test_pred_pca_xgb_best[:5]
```

Out[85]:

```
array([1, 1, 1, 1, 1])
```

In [86]:

```
## PCA - XGBOOST [Hyper parameter tuned] Model Performance

train_matrix = confusion_matrix(y_train, y_train_pred_pca_xgb_best)
test_matrix = confusion_matrix(y_test, y_test_pred_pca_xgb_best)

print('Train Performance :\n')
model_metrics(train_matrix)

print('\nTest Performance :\n')
model_metrics(test_matrix)
```

Train Performance :

```
Accuracy : 0.873
Sensitivity / True Positive Rate / Recall : 0.887
Specificity / True Negative Rate : 0.872
Precision / Positive Predictive Value : 0.396
F1-score : 0.548
```

Test Performance :

```
Accuracy : 0.086
Sensitivity / True Positive Rate / Recall : 1.0
Specificity / True Negative Rate : 0.0
Precision / Positive Predictive Value : 0.086
F1-score : 0.158
```

In [87]:

```
## PCA - XGBOOST [Hyper parameter tuned] Model Performance
print('Train AUC Score')
print(roc_auc_score(y_train, pca_xgb_best.predict_proba(X_train_pca)[:, 1]))
print('Test AUC Score')
print(roc_auc_score(y_test, pca_xgb_best.predict_proba(X_test_pca)[:, 1]))
```

```
Train AUC Score
0.9442462043611259
Test AUC Score
0.6353301334697982
```

Recommendations

In [88]:

```
print('Most Important Predictors of churn , in the order of importance are : ')
lr_results.sort_values(by=coef_column, key=lambda x: abs(x), ascending=False)[ 'coefs' ]
```

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

Out[88]:

```
loc_ic_t2f_mou_8      -1.2736
total_rech_num_8      -1.2033
total_rech_num_6       0.6053
monthly_3g_8_0         0.3994
monthly_2g_8_0         0.3666
std_ic_t2f_mou_8      -0.3363
std_og_t2f_mou_8      -0.2474
const                  -0.2336
monthly_3g_7_0         -0.2099
std_ic_t2f_mou_7       0.1532
sachet_2g_6_0          -0.1108
sachet_2g_7_0          -0.0987
sachet_2g_8_0          0.0488
sachet_3g_6_0          -0.0399
Name: coef, dtype: float64
```

From the above, the following are the strongest indicators of churn

- Customers who churn show lower average monthly local incoming calls from fixed line in the action period by 1.27 standard deviations , compared to users who don't churn , when all other factors are held constant. This is the strongest indicator of churn.
- Customers who churn show lower number of recharges done in action period by 1.20 standard deviations, when all other factors are held constant. This is the second strongest indicator of churn.
- Further customers who churn have done 0.6 standard deviations higher recharge than non-churn customers. This factor when coupled with above factors is a good indicator of churn.
- Customers who churn are more likely to be users of 'monthly 2g package-0 / monthly 3g package-0' in action period (approximately 0.3 std deviations higher than other packages), when all other factors are held constant.

Based on the above indicators the recommendations to the telecom company are :

- Concentrate on users with 1.27 std deviations lower than average incoming calls from fixed line. They are most likely to churn.
- Concentrate on users who recharge less number of times (less than 1.2 std deviations compared to avg) in the 8th month. They are second most 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 100%

In []:

In []: