In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline


import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', -1)
```

In [3]:
```
telecom_churn = pd.read_csv(r"C:\Users\Keshav\Downloads\telecom_churn_data.csv")
telecom_churn.head(10)
```

Out[3]:

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_of_month_6	last_date_of_month_7	last_date_of_mo
0	7000842753	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
1	7001865778	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
2	7001625959	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
3	7001204172	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
4	7000142493	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
5	7000286308	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
6	7001051193	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
7	7000701601	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
8	7001524846	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31
9	7001864400	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31

In [4]:
```
telecom_churn.shape
```

Out[4]:
```
(99999, 226)
```

In [5]:
```
telecom_churn.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99999 entries, 0 to 99998
Data columns (total 226 columns):
 #    Column                    Dtype  
---   ------                    -----  
 0    mobile_number             int64  
 1    circle_id                 int64  
 2    loc_og_t2o_mou            float64
 3    std_og_t2o_mou            float64
 4    loc_ic_t2o_mou            float64
 5    last_date_of_month_6      object 
 6    last_date_of_month_7      object 
 7    last_date_of_month_8      object 
 8    last_date_of_month_9      object 
 9    arpu_6                    float64
 10   arpu_7                    float64
 11   arpu_8                    float64
 12   arpu_9                    float64
 13   onnet_mou_6               float64
 14   onnet_mou_7               float64
 15   onnet_mou_8               float64
 16   onnet_mou_9               float64
 17   offnet_mou_6              float64
 18   offnet_mou_7              float64
 19   offnet_mou_8              float64
 20   offnet_mou_9              float64
 21   roam_ic_mou_6             float64
 22   roam_ic_mou_7             float64
 23   roam_ic_mou_8             float64
 24   roam_ic_mou_9             float64
 25   roam_og_mou_6             float64
 26   roam_og_mou_7             float64
 27   roam_og_mou_8             float64
 28   roam_og_mou_9             float64
 29   loc_og_t2t_mou_6          float64
 30   loc_og_t2t_mou_7          float64
 31   loc_og_t2t_mou_8          float64
 32   loc_og_t2t_mou_9          float64
 33   loc_og_t2m_mou_6          float64
 34   loc_og_t2m_mou_7          float64
 35   loc_og_t2m_mou_8          float64
 36   loc_og_t2m_mou_9          float64
 37   loc_og_t2f_mou_6          float64
```

```
38   loc_og_t2f_mou_7          float64
39   loc_og_t2f_mou_8          float64
40   loc_og_t2f_mou_9          float64
41   loc_og_t2c_mou_6          float64
42   loc_og_t2c_mou_7          float64
43   loc_og_t2c_mou_8          float64
44   loc_og_t2c_mou_9          float64
45   loc_og_mou_6              float64
46   loc_og_mou_7              float64
47   loc_og_mou_8              float64
48   loc_og_mou_9              float64
49   std_og_t2t_mou_6          float64
50   std_og_t2t_mou_7          float64
51   std_og_t2t_mou_8          float64
52   std_og_t2t_mou_9          float64
53   std_og_t2m_mou_6          float64
54   std_og_t2m_mou_7          float64
55   std_og_t2m_mou_8          float64
56   std_og_t2m_mou_9          float64
57   std_og_t2f_mou_6          float64
58   std_og_t2f_mou_7          float64
59   std_og_t2f_mou_8          float64
60   std_og_t2f_mou_9          float64
61   std_og_t2c_mou_6          float64
62   std_og_t2c_mou_7          float64
63   std_og_t2c_mou_8          float64
64   std_og_t2c_mou_9          float64
65   std_og_mou_6              float64
66   std_og_mou_7              float64
67   std_og_mou_8              float64
68   std_og_mou_9              float64
69   isd_og_mou_6              float64
70   isd_og_mou_7              float64
71   isd_og_mou_8              float64
72   isd_og_mou_9              float64
73   spl_og_mou_6              float64
74   spl_og_mou_7              float64
75   spl_og_mou_8              float64
76   spl_og_mou_9              float64
77   og_others_6               float64
78   og_others_7               float64
79   og_others_8               float64
80   og_others_9               float64
81   total_og_mou_6            float64
82   total_og_mou_7            float64
83   total_og_mou_8            float64
84   total_og_mou_9            float64
85   loc_ic_t2t_mou_6          float64
86   loc_ic_t2t_mou_7          float64
87   loc_ic_t2t_mou_8          float64
88   loc_ic_t2t_mou_9          float64
89   loc_ic_t2m_mou_6          float64
90   loc_ic_t2m_mou_7          float64
91   loc_ic_t2m_mou_8          float64
92   loc_ic_t2m_mou_9          float64
93   loc_ic_t2f_mou_6          float64
94   loc_ic_t2f_mou_7          float64
95   loc_ic_t2f_mou_8          float64
96   loc_ic_t2f_mou_9          float64
97   loc_ic_mou_6              float64
98   loc_ic_mou_7              float64
99   loc_ic_mou_8              float64
100  loc_ic_mou_9              float64
101  std_ic_t2t_mou_6          float64
102  std_ic_t2t_mou_7          float64
103  std_ic_t2t_mou_8          float64
104  std_ic_t2t_mou_9          float64
105  std_ic_t2m_mou_6          float64
106  std_ic_t2m_mou_7          float64
107  std_ic_t2m_mou_8          float64
108  std_ic_t2m_mou_9          float64
109  std_ic_t2f_mou_6          float64
110  std_ic_t2f_mou_7          float64
111  std_ic_t2f_mou_8          float64
112  std_ic_t2f_mou_9          float64
113  std_ic_t2o_mou_6          float64
114  std_ic_t2o_mou_7          float64
115  std_ic_t2o_mou_8          float64
116  std_ic_t2o_mou_9          float64
117  std_ic_mou_6              float64
118  std_ic_mou_7              float64
119  std_ic_mou_8              float64
120  std_ic_mou_9              float64
121  total_ic_mou_6            float64
122  total_ic_mou_7            float64
123  total_ic_mou_8            float64
124  total_ic_mou_9            float64
125  spl_ic_mou_6              float64
126  spl_ic_mou_7              float64
```

```
127  spl_ic_mou_8              float64
128  spl_ic_mou_9              float64
129  isd_ic_mou_6              float64
130  isd_ic_mou_7              float64
131  isd_ic_mou_8              float64
132  isd_ic_mou_9              float64
133  ic_others_6               float64
134  ic_others_7               float64
135  ic_others_8               float64
136  ic_others_9               float64
137  total_rech_num_6          int64
138  total_rech_num_7          int64
139  total_rech_num_8          int64
140  total_rech_num_9          int64
141  total_rech_amt_6          int64
142  total_rech_amt_7          int64
143  total_rech_amt_8          int64
144  total_rech_amt_9          int64
145  max_rech_amt_6            int64
146  max_rech_amt_7            int64
147  max_rech_amt_8            int64
148  max_rech_amt_9            int64
149  date_of_last_rech_6       object
150  date_of_last_rech_7       object
151  date_of_last_rech_8       object
152  date_of_last_rech_9       object
153  last_day_rch_amt_6        int64
154  last_day_rch_amt_7        int64
155  last_day_rch_amt_8        int64
156  last_day_rch_amt_9        int64
157  date_of_last_rech_data_6  object
158  date_of_last_rech_data_7  object
159  date_of_last_rech_data_8  object
160  date_of_last_rech_data_9  object
161  total_rech_data_6         float64
162  total_rech_data_7         float64
163  total_rech_data_8         float64
164  total_rech_data_9         float64
165  max_rech_data_6           float64
166  max_rech_data_7           float64
167  max_rech_data_8           float64
168  max_rech_data_9           float64
169  count_rech_2g_6           float64
170  count_rech_2g_7           float64
171  count_rech_2g_8           float64
172  count_rech_2g_9           float64
173  count_rech_3g_6           float64
174  count_rech_3g_7           float64
175  count_rech_3g_8           float64
176  count_rech_3g_9           float64
177  av_rech_amt_data_6        float64
178  av_rech_amt_data_7        float64
179  av_rech_amt_data_8        float64
180  av_rech_amt_data_9        float64
181  vol_2g_mb_6               float64
182  vol_2g_mb_7               float64
183  vol_2g_mb_8               float64
184  vol_2g_mb_9               float64
185  vol_3g_mb_6               float64
186  vol_3g_mb_7               float64
187  vol_3g_mb_8               float64
188  vol_3g_mb_9               float64
189  arpu_3g_6                 float64
190  arpu_3g_7                 float64
191  arpu_3g_8                 float64
192  arpu_3g_9                 float64
193  arpu_2g_6                 float64
194  arpu_2g_7                 float64
195  arpu_2g_8                 float64
196  arpu_2g_9                 float64
197  night_pck_user_6          float64
198  night_pck_user_7          float64
199  night_pck_user_8          float64
200  night_pck_user_9          float64
201  monthly_2g_6              int64
202  monthly_2g_7              int64
203  monthly_2g_8              int64
204  monthly_2g_9              int64
205  sachet_2g_6               int64
206  sachet_2g_7               int64
207  sachet_2g_8               int64
208  sachet_2g_9               int64
209  monthly_3g_6              int64
210  monthly_3g_7              int64
211  monthly_3g_8              int64
212  monthly_3g_9              int64
213  sachet_3g_6               int64
214  sachet_3g_7               int64
215  sachet_3g_8               int64
```

```
 216  sachet_3g_9               int64
 217  fb_user_6                 float64
 218  fb_user_7                 float64
 219  fb_user_8                 float64
 220  fb_user_9                 float64
 221  aon                       int64
 222  aug_vbc_3g                float64
 223  jul_vbc_3g                float64
 224  jun_vbc_3g                float64
 225  sep_vbc_3g                float64
dtypes: float64(179), int64(35), object(12)
memory usage: 172.4+ MB
```

In [6]:
```
telecom_churn.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
mobile_number	99999.0	7.001207e+09	695669.386290	7.000000e+09	7.000606e+09	7.001205e+09	7.001812e+09	7.002411e+09
circle_id	99999.0	1.090000e+02	0.000000	1.090000e+02	1.090000e+02	1.090000e+02	1.090000e+02	1.090000e+02
loc_og_t2o_mou	98981.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_og_t2o_mou	98981.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
loc_ic_t2o_mou	98981.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
arpu_6	99999.0	2.829874e+02	328.439770	-2.258709e+03	9.341150e+01	1.977040e+02	3.710600e+02	2.773109e+04
arpu_7	99999.0	2.785366e+02	338.156291	-2.014045e+03	8.698050e+01	1.916400e+02	3.653445e+02	3.514583e+04
arpu_8	99999.0	2.791547e+02	344.474791	-9.458080e+02	8.412600e+01	1.920800e+02	3.693705e+02	3.354362e+04
arpu_9	99999.0	2.616451e+02	341.998630	-1.899505e+03	6.268500e+01	1.768490e+02	3.534665e+02	3.880562e+04
onnet_mou_6	96062.0	1.323959e+02	297.207406	0.000000e+00	7.380000e+00	3.431000e+01	1.187400e+02	7.376710e+03
onnet_mou_7	96140.0	1.336708e+02	308.794148	0.000000e+00	6.660000e+00	3.233000e+01	1.155950e+02	8.157780e+03
onnet_mou_8	94621.0	1.330181e+02	308.951589	0.000000e+00	6.460000e+00	3.236000e+01	1.158600e+02	1.075256e+04
onnet_mou_9	92254.0	1.303023e+02	308.477668	0.000000e+00	5.330000e+00	2.984000e+01	1.121300e+02	1.042746e+04
offnet_mou_6	96062.0	1.979356e+02	316.851613	0.000000e+00	3.473000e+01	9.631000e+01	2.318600e+02	8.362360e+03
offnet_mou_7	96140.0	1.970451e+02	325.862803	0.000000e+00	3.219000e+01	9.173500e+01	2.268150e+02	9.667130e+03
offnet_mou_8	94621.0	1.965748e+02	327.170662	0.000000e+00	3.163000e+01	9.214000e+01	2.282600e+02	1.400734e+04
offnet_mou_9	92254.0	1.903372e+02	319.396092	0.000000e+00	2.713000e+01	8.729000e+01	2.205050e+02	1.031076e+04
roam_ic_mou_6	96062.0	9.950013e+00	72.825411	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.372438e+04
roam_ic_mou_7	96140.0	7.149898e+00	73.447948	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.537104e+04
roam_ic_mou_8	94621.0	7.292981e+00	68.402466	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.309536e+04
roam_ic_mou_9	92254.0	6.343841e+00	57.137537	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.464030e+03
roam_og_mou_6	96062.0	1.391134e+01	71.443196	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.775110e+03
roam_og_mou_7	96140.0	9.818732e+00	58.455762	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.812040e+03
roam_og_mou_8	94621.0	9.971890e+00	64.713221	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.337040e+03
roam_og_mou_9	92254.0	8.555519e+00	58.438186	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.428460e+03
loc_og_t2t_mou_6	96062.0	4.710076e+01	150.856393	0.000000e+00	1.660000e+00	1.191000e+01	4.096000e+01	6.431330e+03
loc_og_t2t_mou_7	96140.0	4.647301e+01	155.318705	0.000000e+00	1.630000e+00	1.161000e+01	3.991000e+01	7.400660e+03
loc_og_t2t_mou_8	94621.0	4.588781e+01	151.184830	0.000000e+00	1.600000e+00	1.173000e+01	4.011000e+01	1.075256e+04
loc_og_t2t_mou_9	92254.0	4.458445e+01	147.995390	0.000000e+00	1.360000e+00	1.126000e+01	3.928000e+01	1.038924e+04
loc_og_t2m_mou_6	96062.0	9.334209e+01	162.780544	0.000000e+00	9.880000e+00	4.103000e+01	1.103900e+02	4.729740e+03
loc_og_t2m_mou_7	96140.0	9.139713e+01	157.492308	0.000000e+00	1.002500e+01	4.043000e+01	1.075600e+02	4.557140e+03
loc_og_t2m_mou_8	94621.0	9.175513e+01	156.537048	0.000000e+00	9.810000e+00	4.036000e+01	1.090900e+02	4.961330e+03
loc_og_t2m_mou_9	92254.0	9.046319e+01	158.681454	0.000000e+00	8.810000e+00	3.912000e+01	1.068100e+02	4.429880e+03
loc_og_t2f_mou_6	96062.0	3.751013e+00	14.230438	0.000000e+00	0.000000e+00	0.000000e+00	2.080000e+00	1.466030e+03
loc_og_t2f_mou_7	96140.0	3.792985e+00	14.264986	0.000000e+00	0.000000e+00	0.000000e+00	2.090000e+00	1.196430e+03
loc_og_t2f_mou_8	94621.0	3.677991e+00	13.270996	0.000000e+00	0.000000e+00	0.000000e+00	2.040000e+00	9.284900e+02
loc_og_t2f_mou_9	92254.0	3.655123e+00	13.457549	0.000000e+00	0.000000e+00	0.000000e+00	1.940000e+00	9.274100e+02
loc_og_t2c_mou_6	96062.0	1.123056e+00	5.448946	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.428600e+02
loc_og_t2c_mou_7	96140.0	1.368500e+00	7.533445	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.162400e+02
loc_og_t2c_mou_8	94621.0	1.433821e+00	6.783335	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.020900e+02
loc_og_t2c_mou_9	92254.0	1.232726e+00	5.619021	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.398400e+02
loc_og_mou_6	96062.0	1.442012e+02	251.751489	0.000000e+00	1.711000e+01	6.511000e+01	1.682700e+02	1.064338e+04
loc_og_mou_7	96140.0	1.416705e+02	248.731086	0.000000e+00	1.748000e+01	6.368500e+01	1.643825e+02	7.674780e+03
loc_og_mou_8	94621.0	1.413282e+02	245.914311	0.000000e+00	1.711000e+01	6.373000e+01	1.661100e+02	1.103991e+04
loc_og_mou_9	92254.0	1.387100e+02	245.934517	0.000000e+00	1.556000e+01	6.184000e+01	1.622250e+02	1.109926e+04

std_og_t2t_mou_6	96062.0	7.982987e+01	252.476533	0.000000e+00	0.000000e+00	0.000000e+00	3.080750e+01	7.366580e+03
std_og_t2t_mou_7	96140.0	8.329960e+01	263.631042	0.000000e+00	0.000000e+00	0.000000e+00	3.113250e+01	8.133660e+03
std_og_t2t_mou_8	94621.0	8.328267e+01	265.486090	0.000000e+00	0.000000e+00	0.000000e+00	3.058000e+01	8.014430e+03
std_og_t2t_mou_9	92254.0	8.234292e+01	267.184991	0.000000e+00	0.000000e+00	0.000000e+00	2.823000e+01	9.382580e+03
std_og_t2m_mou_6	96062.0	8.729962e+01	255.617850	0.000000e+00	0.000000e+00	3.950000e+00	5.329000e+01	8.314760e+03
std_og_t2m_mou_7	96140.0	9.080414e+01	269.347911	0.000000e+00	0.000000e+00	3.635000e+00	5.404000e+01	9.284740e+03
std_og_t2m_mou_8	94621.0	8.983839e+01	271.757783	0.000000e+00	0.000000e+00	3.310000e+00	5.249000e+01	1.395004e+04
std_og_t2m_mou_9	92254.0	8.627662e+01	261.407396	0.000000e+00	0.000000e+00	2.500000e+00	4.856000e+01	1.022343e+04
std_og_t2f_mou_6	96062.0	1.129011e+00	7.984970	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.285600e+02
std_og_t2f_mou_7	96140.0	1.115010e+00	8.599406	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.446300e+02
std_og_t2f_mou_8	94621.0	1.067792e+00	7.905971	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.169100e+02
std_og_t2f_mou_9	92254.0	1.042362e+00	8.261770	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.084900e+02
std_og_t2c_mou_6	96062.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_og_t2c_mou_7	96140.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_og_t2c_mou_8	94621.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_og_t2c_mou_9	92254.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_og_mou_6	96062.0	1.682612e+02	389.948499	0.000000e+00	0.000000e+00	1.164000e+01	1.448375e+02	8.432990e+03
std_og_mou_7	96140.0	1.752214e+02	408.922934	0.000000e+00	0.000000e+00	1.109000e+01	1.506150e+02	1.093673e+04
std_og_mou_8	94621.0	1.741915e+02	411.633049	0.000000e+00	0.000000e+00	1.041000e+01	1.479400e+02	1.398006e+04
std_og_mou_9	92254.0	1.696645e+02	405.138658	0.000000e+00	0.000000e+00	8.410000e+00	1.421050e+02	1.149531e+04
isd_og_mou_6	96062.0	7.982775e-01	25.765248	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.900660e+03
isd_og_mou_7	96140.0	7.765721e-01	25.603052	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.490280e+03
isd_og_mou_8	94621.0	7.912471e-01	25.544471	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.681540e+03
isd_og_mou_9	92254.0	7.238921e-01	21.310751	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.244530e+03
spl_og_mou_6	96062.0	3.916811e+00	14.936449	0.000000e+00	0.000000e+00	0.000000e+00	2.430000e+00	1.023210e+03
spl_og_mou_7	96140.0	4.978279e+00	20.661570	0.000000e+00	0.000000e+00	0.000000e+00	3.710000e+00	2.372510e+03
spl_og_mou_8	94621.0	5.053769e+00	17.855111	0.000000e+00	0.000000e+00	0.000000e+00	3.990000e+00	1.390880e+03
spl_og_mou_9	92254.0	4.412767e+00	16.328227	0.000000e+00	0.000000e+00	0.000000e+00	3.230000e+00	1.635710e+03
og_others_6	96062.0	4.541571e-01	4.125911	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.008900e+02
og_others_7	96140.0	3.023539e-02	2.161717	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.701300e+02
og_others_8	94621.0	3.337198e-02	2.323464	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.949300e+02
og_others_9	92254.0	4.745572e-02	3.635466	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	7.877900e+02
total_og_mou_6	99999.0	3.051334e+02	463.419481	0.000000e+00	4.474000e+01	1.451400e+02	3.728600e+02	1.067403e+04
total_og_mou_7	99999.0	3.102312e+02	480.031178	0.000000e+00	4.301000e+01	1.415300e+02	3.785700e+02	1.136531e+04
total_og_mou_8	99999.0	3.041195e+02	478.150031	0.000000e+00	3.858000e+01	1.386100e+02	3.699000e+02	1.404306e+04
total_og_mou_9	99999.0	2.892792e+02	468.980002	0.000000e+00	2.551000e+01	1.254600e+02	3.534800e+02	1.151773e+04
loc_ic_t2t_mou_6	96062.0	4.792237e+01	140.258485	0.000000e+00	2.990000e+00	1.569000e+01	4.684000e+01	6.626930e+03
loc_ic_t2t_mou_7	96140.0	4.799052e+01	145.795055	0.000000e+00	3.230000e+00	1.574000e+01	4.581000e+01	9.324660e+03
loc_ic_t2t_mou_8	94621.0	4.721136e+01	137.239552	0.000000e+00	3.280000e+00	1.603000e+01	4.629000e+01	1.069623e+04
loc_ic_t2t_mou_9	92254.0	4.628179e+01	140.130610	0.000000e+00	3.290000e+00	1.566000e+01	4.518000e+01	1.059883e+04
loc_ic_t2m_mou_6	96062.0	1.074757e+02	171.713903	0.000000e+00	1.729000e+01	5.649000e+01	1.323875e+02	4.693860e+03
loc_ic_t2m_mou_7	96140.0	1.071205e+02	169.423620	0.000000e+00	1.859000e+01	5.708000e+01	1.309600e+02	4.455830e+03
loc_ic_t2m_mou_8	94621.0	1.084605e+02	169.723759	0.000000e+00	1.893000e+01	5.824000e+01	1.339300e+02	6.274190e+03
loc_ic_t2m_mou_9	92254.0	1.061555e+02	165.492803	0.000000e+00	1.856000e+01	5.661000e+01	1.304900e+02	5.463780e+03
loc_ic_t2f_mou_6	96062.0	1.208430e+01	40.140895	0.000000e+00	0.000000e+00	8.800000e-01	8.140000e+00	1.872340e+03
loc_ic_t2f_mou_7	96140.0	1.259970e+01	42.977442	0.000000e+00	0.000000e+00	9.300000e-01	8.282500e+00	1.983010e+03
loc_ic_t2f_mou_8	94621.0	1.175183e+01	39.125379	0.000000e+00	0.000000e+00	9.300000e-01	8.110000e+00	2.433060e+03
loc_ic_t2f_mou_9	92254.0	1.217310e+01	43.840776	0.000000e+00	0.000000e+00	9.600000e-01	8.140000e+00	4.318280e+03
loc_ic_mou_6	96062.0	1.674911e+02	254.124029	0.000000e+00	3.039000e+01	9.216000e+01	2.080750e+02	7.454630e+03
loc_ic_mou_7	96140.0	1.677195e+02	256.242707	0.000000e+00	3.246000e+01	9.255000e+01	2.058375e+02	9.669910e+03
loc_ic_mou_8	94621.0	1.674326e+02	250.025523	0.000000e+00	3.274000e+01	9.383000e+01	2.072800e+02	1.083016e+04
loc_ic_mou_9	92254.0	1.646193e+02	249.845070	0.000000e+00	3.229000e+01	9.164000e+01	2.027375e+02	1.079629e+04
std_ic_t2t_mou_6	96062.0	9.575993e+00	54.330607	0.000000e+00	0.000000e+00	0.000000e+00	4.060000e+00	5.459560e+03
std_ic_t2t_mou_7	96140.0	1.001190e+01	57.411971	0.000000e+00	0.000000e+00	0.000000e+00	4.230000e+00	5.800930e+03

std_ic_t2t_mou_8	94621.0	9.883921e+00	55.073186	0.000000e+00	0.000000e+00	0.000000e+00	4.080000e+00	4.309290e+03
std_ic_t2t_mou_9	92254.0	9.432479e+00	53.376273	0.000000e+00	0.000000e+00	0.000000e+00	3.510000e+00	3.819830e+03
std_ic_t2m_mou_6	96062.0	2.072224e+01	80.793414	0.000000e+00	0.000000e+00	2.030000e+00	1.503000e+01	5.647160e+03
std_ic_t2m_mou_7	96140.0	2.165641e+01	86.521393	0.000000e+00	0.000000e+00	2.040000e+00	1.574000e+01	6.141880e+03
std_ic_t2m_mou_8	94621.0	2.118321e+01	83.683565	0.000000e+00	0.000000e+00	2.030000e+00	1.536000e+01	5.645860e+03
std_ic_t2m_mou_9	92254.0	1.962091e+01	74.913050	0.000000e+00	0.000000e+00	1.740000e+00	1.426000e+01	5.689760e+03
std_ic_t2f_mou_6	96062.0	2.156397e+00	16.495594	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.351110e+03
std_ic_t2f_mou_7	96140.0	2.216923e+00	16.454061	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.136080e+03
std_ic_t2f_mou_8	94621.0	2.085004e+00	15.812580	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.394890e+03
std_ic_t2f_mou_9	92254.0	2.173419e+00	15.978601	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.431960e+03
std_ic_t2o_mou_6	96062.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_ic_t2o_mou_7	96140.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_ic_t2o_mou_8	94621.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_ic_t2o_mou_9	92254.0	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
std_ic_mou_6	96062.0	3.245718e+01	106.283386	0.000000e+00	0.000000e+00	5.890000e+00	2.693000e+01	5.712110e+03
std_ic_mou_7	96140.0	3.388783e+01	113.720168	0.000000e+00	0.000000e+00	5.960000e+00	2.831000e+01	6.745760e+03
std_ic_mou_8	94621.0	3.315474e+01	110.127008	0.000000e+00	1.000000e-02	5.880000e+00	2.771000e+01	5.957140e+03
std_ic_mou_9	92254.0	3.122934e+01	101.982303	0.000000e+00	0.000000e+00	5.380000e+00	2.569000e+01	5.956660e+03
total_ic_mou_6	99999.0	2.001300e+02	291.651671	0.000000e+00	3.853000e+01	1.147400e+02	2.516700e+02	7.716140e+03
total_ic_mou_7	99999.0	2.028531e+02	298.124954	0.000000e+00	4.119000e+01	1.163400e+02	2.506600e+02	9.699010e+03
total_ic_mou_8	99999.0	1.987508e+02	289.321094	0.000000e+00	3.829000e+01	1.146600e+02	2.489900e+02	1.083038e+04
total_ic_mou_9	99999.0	1.892143e+02	284.823024	0.000000e+00	3.237000e+01	1.058900e+02	2.363200e+02	1.079659e+04
spl_ic_mou_6	96062.0	6.155660e-02	0.160920	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.976000e+01
spl_ic_mou_7	96140.0	3.358477e-02	0.155725	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.133000e+01
spl_ic_mou_8	94621.0	4.036134e-02	0.146147	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.686000e+01
spl_ic_mou_9	92254.0	1.631370e-01	0.527860	0.000000e+00	0.000000e+00	0.000000e+00	6.000000e-02	6.238000e+01
isd_ic_mou_6	96062.0	7.460608e+00	59.722948	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.789410e+03
isd_ic_mou_7	96140.0	8.334936e+00	65.219829	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.289540e+03
isd_ic_mou_8	94621.0	8.442001e+00	63.813098	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.127010e+03
isd_ic_mou_9	92254.0	8.063003e+00	63.505379	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.057740e+03
ic_others_6	96062.0	8.546555e-01	11.955164	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.362940e+03
ic_others_7	96140.0	1.012960e+00	12.673099	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.495940e+03
ic_others_8	94621.0	9.708005e-01	13.284348	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.327510e+03
ic_others_9	92254.0	1.017162e+00	12.381172	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.005230e+03
total_rech_num_6	99999.0	7.558806e+00	7.078405	0.000000e+00	3.000000e+00	6.000000e+00	9.000000e+00	3.070000e+02
total_rech_num_7	99999.0	7.700367e+00	7.070422	0.000000e+00	3.000000e+00	6.000000e+00	1.000000e+01	1.380000e+02
total_rech_num_8	99999.0	7.212912e+00	7.203753	0.000000e+00	3.000000e+00	5.000000e+00	9.000000e+00	1.960000e+02
total_rech_num_9	99999.0	6.893019e+00	7.096261	0.000000e+00	3.000000e+00	5.000000e+00	9.000000e+00	1.310000e+02
total_rech_amt_6	99999.0	3.275146e+02	398.019701	0.000000e+00	1.090000e+02	2.300000e+02	4.375000e+02	3.519000e+04
total_rech_amt_7	99999.0	3.229630e+02	408.114237	0.000000e+00	1.000000e+02	2.200000e+02	4.280000e+02	4.033500e+04
total_rech_amt_8	99999.0	3.241571e+02	416.540455	0.000000e+00	9.000000e+01	2.250000e+02	4.345000e+02	4.532000e+04
total_rech_amt_9	99999.0	3.033457e+02	404.588583	0.000000e+00	5.200000e+01	2.000000e+02	4.150000e+02	3.723500e+04
max_rech_amt_6	99999.0	1.046375e+02	120.614894	0.000000e+00	3.000000e+01	1.100000e+02	1.200000e+02	4.010000e+03
max_rech_amt_7	99999.0	1.047524e+02	124.523970	0.000000e+00	3.000000e+01	1.100000e+02	1.280000e+02	4.010000e+03
max_rech_amt_8	99999.0	1.077282e+02	126.902505	0.000000e+00	3.000000e+01	9.800000e+01	1.440000e+02	4.449000e+03
max_rech_amt_9	99999.0	1.019439e+02	125.375109	0.000000e+00	2.800000e+01	6.100000e+01	1.440000e+02	3.399000e+03
last_day_rch_amt_6	99999.0	6.315625e+01	97.356649	0.000000e+00	0.000000e+00	3.000000e+01	1.100000e+02	4.010000e+03
last_day_rch_amt_7	99999.0	5.938580e+01	95.915385	0.000000e+00	0.000000e+00	3.000000e+01	1.100000e+02	4.010000e+03
last_day_rch_amt_8	99999.0	6.264172e+01	104.431816	0.000000e+00	0.000000e+00	3.000000e+01	1.300000e+02	4.449000e+03
last_day_rch_amt_9	99999.0	4.390125e+01	90.809712	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e+01	3.399000e+03
total_rech_data_6	25153.0	2.463802e+00	2.789128	1.000000e+00	1.000000e+00	1.000000e+00	3.000000e+00	6.100000e+01
total_rech_data_7	25571.0	2.666419e+00	3.031593	1.000000e+00	1.000000e+00	1.000000e+00	3.000000e+00	5.400000e+01
total_rech_data_8	26339.0	2.651999e+00	3.074987	1.000000e+00	1.000000e+00	1.000000e+00	3.000000e+00	6.000000e+01
total_rech_data_9	25922.0	2.441170e+00	2.516339	1.000000e+00	1.000000e+00	2.000000e+00	3.000000e+00	8.400000e+01
max_rech_data_6	25153.0	1.263934e+02	108.477235	1.000000e+00	2.500000e+01	1.450000e+02	1.770000e+02	1.555000e+03

max_rech_data_7	25571.0	1.267295e+02	109.765267	1.000000e+00	2.500000e+01	1.450000e+02	1.770000e+02	1.555000e+03
max_rech_data_8	26339.0	1.257173e+02	109.437851	1.000000e+00	2.500000e+01	1.450000e+02	1.790000e+02	1.555000e+03
max_rech_data_9	25922.0	1.249414e+02	111.363760	1.000000e+00	2.500000e+01	1.450000e+02	1.790000e+02	1.555000e+03
count_rech_2g_6	25153.0	1.864668e+00	2.570254	0.000000e+00	1.000000e+00	1.000000e+00	2.000000e+00	4.200000e+01
count_rech_2g_7	25571.0	2.044699e+00	2.768332	0.000000e+00	1.000000e+00	1.000000e+00	2.000000e+00	4.800000e+01
count_rech_2g_8	26339.0	2.016288e+00	2.720132	0.000000e+00	1.000000e+00	1.000000e+00	2.000000e+00	4.400000e+01
count_rech_2g_9	25922.0	1.781807e+00	2.214701	0.000000e+00	1.000000e+00	1.000000e+00	2.000000e+00	4.000000e+01
count_rech_3g_6	25153.0	5.991333e-01	1.274428	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	2.900000e+01
count_rech_3g_7	25571.0	6.217199e-01	1.394524	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	3.500000e+01
count_rech_3g_8	26339.0	6.357113e-01	1.422827	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	4.500000e+01
count_rech_3g_9	25922.0	6.593627e-01	1.411513	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	4.900000e+01
av_rech_amt_data_6	25153.0	1.926010e+02	192.646318	1.000000e+00	8.200000e+01	1.540000e+02	2.520000e+02	7.546000e+03
av_rech_amt_data_7	25571.0	2.009813e+02	196.791224	5.000000e-01	9.200000e+01	1.540000e+02	2.520000e+02	4.365000e+03
av_rech_amt_data_8	26339.0	1.975265e+02	191.301305	5.000000e-01	8.700000e+01	1.540000e+02	2.520000e+02	4.076000e+03
av_rech_amt_data_9	25922.0	1.927343e+02	188.400286	1.000000e+00	6.900000e+01	1.640000e+02	2.520000e+02	4.061000e+03
vol_2g_mb_6	99999.0	5.190496e+01	213.356445	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.028590e+04
vol_2g_mb_7	99999.0	5.122994e+01	212.302217	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	7.873550e+03
vol_2g_mb_8	99999.0	5.017015e+01	212.347892	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.111761e+04
vol_2g_mb_9	99999.0	4.471970e+01	198.653570	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.993950e+03
vol_3g_mb_6	99999.0	1.213962e+02	544.247227	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.573540e+04
vol_3g_mb_7	99999.0	1.289958e+02	541.494013	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.814412e+04
vol_3g_mb_8	99999.0	1.354107e+02	558.775335	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.003606e+04
vol_3g_mb_9	99999.0	1.360566e+02	577.394194	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.922127e+04
arpu_3g_6	25153.0	8.955506e+01	193.124653	-3.082000e+01	0.000000e+00	4.800000e-01	1.220700e+02	6.362280e+03
arpu_3g_7	25571.0	8.938412e+01	195.893924	-2.604000e+01	0.000000e+00	4.200000e-01	1.195600e+02	4.980900e+03
arpu_3g_8	26339.0	9.117385e+01	188.180936	-2.449000e+01	0.000000e+00	8.800000e-01	1.220700e+02	3.716900e+03
arpu_3g_9	25922.0	1.002641e+02	216.291992	-7.109000e+01	0.000000e+00	2.605000e+00	1.400100e+02	1.388431e+04
arpu_2g_6	25153.0	8.639800e+01	172.767523	-3.583000e+01	0.000000e+00	1.083000e+01	1.220700e+02	6.433760e+03
arpu_2g_7	25571.0	8.591445e+01	176.379871	-1.548000e+01	0.000000e+00	8.810000e+00	1.220700e+02	4.809360e+03
arpu_2g_8	26339.0	8.659948e+01	168.247852	-5.583000e+01	0.000000e+00	9.270000e+00	1.220700e+02	3.483170e+03
arpu_2g_9	25922.0	9.371203e+01	171.384224	-4.574000e+01	0.000000e+00	1.480000e+01	1.400100e+02	3.467170e+03
night_pck_user_6	25153.0	2.508647e-02	0.156391	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
night_pck_user_7	25571.0	2.303391e-02	0.150014	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
night_pck_user_8	26339.0	2.084362e-02	0.142863	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
night_pck_user_9	25922.0	1.597099e-02	0.125366	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
monthly_2g_6	99999.0	7.964080e-02	0.295058	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
monthly_2g_7	99999.0	8.322083e-02	0.304395	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e+00
monthly_2g_8	99999.0	8.100081e-02	0.299568	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e+00
monthly_2g_9	99999.0	6.878069e-02	0.278120	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
sachet_2g_6	99999.0	3.893839e-01	1.497320	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.200000e+01
sachet_2g_7	99999.0	4.396344e-01	1.636230	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.800000e+01
sachet_2g_8	99999.0	4.500745e-01	1.630263	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.400000e+01
sachet_2g_9	99999.0	3.931039e-01	1.347140	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+01
monthly_3g_6	99999.0	7.592076e-02	0.363371	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.400000e+01
monthly_3g_7	99999.0	7.858079e-02	0.387231	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.600000e+01
monthly_3g_8	99999.0	8.294083e-02	0.384947	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.600000e+01
monthly_3g_9	99999.0	8.634086e-02	0.384978	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.100000e+01
sachet_3g_6	99999.0	7.478075e-02	0.568344	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.900000e+01
sachet_3g_7	99999.0	8.040080e-02	0.628334	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.500000e+01
sachet_3g_8	99999.0	8.450085e-02	0.660234	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.100000e+01
sachet_3g_9	99999.0	8.458085e-02	0.650457	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.900000e+01
fb_user_6	25153.0	9.144038e-01	0.279772	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
fb_user_7	25571.0	9.087638e-01	0.287950	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
fb_user_8	26339.0	8.908083e-01	0.311885	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
fb_user_9	25922.0	8.609675e-01	0.345987	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00

aon	99999.0	1.219855e+03	954.733842	1.800000e+02	4.670000e+02	8.630000e+02	1.807500e+03	4.337000e+03
aug_vbc_3g	99999.0	6.817025e+01	267.580450	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.291622e+04
jul_vbc_3g	99999.0	6.683906e+01	271.201856	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.165600e+03
jun_vbc_3g	99999.0	6.002120e+01	253.938223	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.116621e+04
sep_vbc_3g	99999.0	3.299373e+00	32.408353	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.618570e+03

In [7]:
```
# Cheking percent of missing values in columns
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[7]:

	null
arpu_3g_6	74.85
night_pck_user_6	74.85
total_rech_data_6	74.85
arpu_2g_6	74.85
max_rech_data_6	74.85
fb_user_6	74.85
av_rech_amt_data_6	74.85
date_of_last_rech_data_6	74.85
count_rech_2g_6	74.85
count_rech_3g_6	74.85
date_of_last_rech_data_7	74.43
total_rech_data_7	74.43
fb_user_7	74.43
max_rech_data_7	74.43
night_pck_user_7	74.43
count_rech_2g_7	74.43
av_rech_amt_data_7	74.43
arpu_2g_7	74.43
count_rech_3g_7	74.43
arpu_3g_7	74.43
total_rech_data_9	74.08
count_rech_3g_9	74.08
fb_user_9	74.08
max_rech_data_9	74.08
arpu_3g_9	74.08
date_of_last_rech_data_9	74.08
night_pck_user_9	74.08
arpu_2g_9	74.08
count_rech_2g_9	74.08
av_rech_amt_data_9	74.08
total_rech_data_8	73.66
arpu_3g_8	73.66
fb_user_8	73.66
night_pck_user_8	73.66
av_rech_amt_data_8	73.66
max_rech_data_8	73.66
count_rech_3g_8	73.66
arpu_2g_8	73.66
count_rech_2g_8	73.66
date_of_last_rech_data_8	73.66
ic_others_9	7.75
std_og_mou_9	7.75
std_og_t2c_mou_9	7.75
isd_ic_mou_9	7.75
std_ic_mou_9	7.75

isd_og_mou_9	7.75
spl_og_mou_9	7.75
spl_ic_mou_9	7.75
og_others_9	7.75
loc_ic_t2t_mou_9	7.75
std_ic_t2o_mou_9	7.75
loc_ic_t2m_mou_9	7.75
std_ic_t2f_mou_9	7.75
loc_ic_t2f_mou_9	7.75
loc_ic_mou_9	7.75
std_ic_t2m_mou_9	7.75
std_og_t2f_mou_9	7.75
std_og_t2t_mou_9	7.75
std_ic_t2t_mou_9	7.75
loc_og_mou_9	7.75
roam_og_mou_9	7.75
loc_og_t2m_mou_9	7.75
loc_og_t2f_mou_9	7.75
roam_ic_mou_9	7.75
offnet_mou_9	7.75
loc_og_t2c_mou_9	7.75
loc_og_t2t_mou_9	7.75
std_og_t2m_mou_9	7.75
onnet_mou_9	7.75
onnet_mou_8	5.38
std_ic_t2t_mou_8	5.38
std_ic_mou_8	5.38
loc_ic_t2t_mou_8	5.38
roam_og_mou_8	5.38
std_ic_t2m_mou_8	5.38
loc_ic_mou_8	5.38
std_ic_t2f_mou_8	5.38
roam_ic_mou_8	5.38
std_ic_t2o_mou_8	5.38
loc_og_t2t_mou_8	5.38
loc_ic_t2f_mou_8	5.38
offnet_mou_8	5.38
loc_ic_t2m_mou_8	5.38
loc_og_t2m_mou_8	5.38
isd_og_mou_8	5.38
ic_others_8	5.38
og_others_8	5.38
spl_ic_mou_8	5.38
loc_og_t2f_mou_8	5.38
std_og_t2m_mou_8	5.38
spl_og_mou_8	5.38
std_og_t2c_mou_8	5.38
isd_ic_mou_8	5.38
loc_og_t2c_mou_8	5.38
std_og_t2f_mou_8	5.38
std_og_t2t_mou_8	5.38
loc_og_mou_8	5.38
std_og_mou_8	5.38
date_of_last_rech_9	4.76
std_ic_t2f_mou_6	3.94

ic_others_6	3.94
isd_ic_mou_6	3.94
std_ic_t2m_mou_6	3.94
std_ic_mou_6	3.94
spl_ic_mou_6	3.94
std_ic_t2o_mou_6	3.94
loc_ic_t2f_mou_6	3.94
loc_ic_t2t_mou_6	3.94
std_og_t2c_mou_6	3.94
std_og_t2f_mou_6	3.94
std_og_mou_6	3.94
std_og_t2m_mou_6	3.94
isd_og_mou_6	3.94
std_og_t2t_mou_6	3.94
spl_og_mou_6	3.94
loc_og_mou_6	3.94
og_others_6	3.94
loc_og_t2c_mou_6	3.94
loc_og_t2m_mou_6	3.94
loc_og_t2f_mou_6	3.94
loc_og_t2t_mou_6	3.94
roam_og_mou_6	3.94
std_ic_t2t_mou_6	3.94
onnet_mou_6	3.94
loc_ic_mou_6	3.94
offnet_mou_6	3.94
roam_ic_mou_6	3.94
loc_ic_t2m_mou_6	3.94
loc_og_t2c_mou_7	3.86
roam_ic_mou_7	3.86
loc_og_mou_7	3.86
loc_og_t2t_mou_7	3.86
offnet_mou_7	3.86
loc_og_t2f_mou_7	3.86
std_og_t2t_mou_7	3.86
std_ic_t2t_mou_7	3.86
onnet_mou_7	3.86
std_og_t2m_mou_7	3.86
loc_og_t2m_mou_7	3.86
std_og_t2f_mou_7	3.86
roam_og_mou_7	3.86
std_og_t2c_mou_7	3.86
std_ic_t2m_mou_7	3.86
isd_og_mou_7	3.86
ic_others_7	3.86
loc_ic_t2f_mou_7	3.86
loc_ic_t2m_mou_7	3.86
std_ic_mou_7	3.86
loc_ic_t2t_mou_7	3.86
std_ic_t2f_mou_7	3.86
loc_ic_mou_7	3.86
spl_ic_mou_7	3.86
og_others_7	3.86
spl_og_mou_7	3.86
isd_ic_mou_7	3.86

std_ic_t2o_mou_7	3.86
std_og_mou_7	3.86
date_of_last_rech_8	3.62
date_of_last_rech_7	1.77
last_date_of_month_9	1.66
date_of_last_rech_6	1.61
last_date_of_month_8	1.10
loc_ic_t2o_mou	1.02
std_og_t2o_mou	1.02
loc_og_t2o_mou	1.02
last_date_of_month_7	0.60
sachet_3g_8	0.00
jul_vbc_3g	0.00
aug_vbc_3g	0.00
aon	0.00
jun_vbc_3g	0.00
monthly_2g_9	0.00
sachet_3g_6	0.00
vol_3g_mb_9	0.00
sachet_3g_7	0.00
monthly_2g_8	0.00
monthly_3g_9	0.00
monthly_3g_8	0.00
sachet_3g_9	0.00
monthly_3g_7	0.00
monthly_3g_6	0.00
sachet_2g_9	0.00
sachet_2g_8	0.00
sachet_2g_7	0.00
sachet_2g_6	0.00
monthly_2g_7	0.00
monthly_2g_6	0.00
mobile_number	0.00
vol_3g_mb_8	0.00
total_og_mou_9	0.00
total_rech_num_7	0.00
total_rech_num_6	0.00
total_ic_mou_9	0.00
total_ic_mou_8	0.00
total_ic_mou_7	0.00
total_ic_mou_6	0.00
circle_id	0.00
total_og_mou_8	0.00
vol_3g_mb_7	0.00
total_og_mou_7	0.00
total_og_mou_6	0.00
arpu_9	0.00
arpu_8	0.00
arpu_7	0.00
arpu_6	0.00
last_date_of_month_6	0.00
total_rech_num_8	0.00
total_rech_num_9	0.00
total_rech_amt_6	0.00
total_rech_amt_7	0.00

vol_3g_mb_6	0.00
vol_2g_mb_9	0.00
vol_2g_mb_8	0.00
vol_2g_mb_7	0.00
vol_2g_mb_6	0.00
last_day_rch_amt_9	0.00
last_day_rch_amt_8	0.00
last_day_rch_amt_7	0.00
last_day_rch_amt_6	0.00
max_rech_amt_9	0.00
max_rech_amt_8	0.00
max_rech_amt_7	0.00
max_rech_amt_6	0.00
total_rech_amt_9	0.00
total_rech_amt_8	0.00
sep_vbc_3g	0.00

In [8]:
```
# List the columns having more than 30% missing values
col_list_missing_30 = list(telecom_churn_missing_columns.index[telecom_churn_missing_columns['null'] > 30])
```

In [9]:
```
# Delete the columns having more than 30% missing values
telecom_churn = telecom_churn.drop(col_list_missing_30, axis=1)
```

In [10]:
```
telecom_churn.shape
```

Out[10]:
```
(99999, 186)
```

In [11]:
```
# List the date columns
date_cols = [k for k in telecom_churn.columns.to_list() if 'date' in k]
print(date_cols)
```

```
['last_date_of_month_6', 'last_date_of_month_7', 'last_date_of_month_8', 'last_date_of_month_9', 'date_of_last_
rech_6', 'date_of_last_rech_7', 'date_of_last_rech_8', 'date_of_last_rech_9']
```

In [12]:
```
# Dropping date columns
telecom_churn = telecom_churn.drop(date_cols, axis=1)
```

In [13]:
```
# Drop circle_id column
telecom_churn = telecom_churn.drop('circle_id', axis=1)
```

In [14]:
```
telecom_churn.shape
```

Out[14]:
```
(99999, 177)
```

In [15]:
```
telecom_churn['avg_rech_amt_6_7'] = (telecom_churn['total_rech_amt_6'] + telecom_churn['total_rech_amt_7'])/2
```

In [16]:
```
X = telecom_churn['avg_rech_amt_6_7'].quantile(0.7)
X
```

Out[16]:
```
368.5
```

In [17]:
```
telecom_churn = telecom_churn[telecom_churn['avg_rech_amt_6_7'] >= X]
telecom_churn.head()
```

Out[17]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	c
7	7000701601	0.0	0.0	0.0	1069.180	1349.850	3171.480	500.000	57.84	54.68	
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	166.787	413.69	351.03	
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	322.732	501.76	108.39	
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	206.490	50.51	74.01	
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	257.583	1185.91	9.28	

In [18]:
```
telecom_churn.shape
```

Out[18]:
```
(30011, 178)
```

In [19]:
```
# Count the rows having more than 50% missing values
telecom_churn_missing_rows_50 =telecom_churn[(telecom_churn.isnull().sum(axis=1)) > (len(telecom_churn.columns)
telecom_churn_missing_rows_50.shape
```

Out[19]:
```
(114, 178)
```

In [20]:
```
# Deleting the rows having more than 50% missing values
telecom_churn = telecom_churn.drop(telecom_churn_missing_rows_50.index)
telecom_churn.shape
```

Out[20]:
```
(29897, 178)
```

In [21]:
```
# Checking the missing values in columns again
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[21]:

	null
loc_ic_mou_9	5.32
og_others_9	5.32
loc_og_t2t_mou_9	5.32
loc_ic_t2t_mou_9	5.32
loc_og_t2m_mou_9	5.32
loc_og_t2f_mou_9	5.32
loc_og_t2c_mou_9	5.32
std_ic_t2m_mou_9	5.32
loc_og_mou_9	5.32
std_og_t2t_mou_9	5.32
roam_og_mou_9	5.32
std_ic_t2o_mou_9	5.32
std_og_t2m_mou_9	5.32
std_og_t2f_mou_9	5.32
spl_og_mou_9	5.32
std_og_t2c_mou_9	5.32
std_og_mou_9	5.32
isd_og_mou_9	5.32
std_ic_t2t_mou_9	5.32
std_ic_mou_9	5.32
onnet_mou_9	5.32
spl_ic_mou_9	5.32
ic_others_9	5.32
isd_ic_mou_9	5.32
loc_ic_t2f_mou_9	5.32
offnet_mou_9	5.32
loc_ic_t2m_mou_9	5.32
std_ic_t2f_mou_9	5.32
roam_ic_mou_9	5.32
loc_og_t2t_mou_8	2.76
og_others_8	2.76
roam_ic_mou_8	2.76
std_og_t2m_mou_8	2.76
isd_ic_mou_8	2.76
std_ic_t2o_mou_8	2.76
std_og_t2f_mou_8	2.76
spl_og_mou_8	2.76
std_og_t2t_mou_8	2.76
std_og_t2c_mou_8	2.76
loc_ic_mou_8	2.76
std_ic_t2f_mou_8	2.76
std_og_mou_8	2.76
ic_others_8	2.76
roam_og_mou_8	2.76
loc_ic_t2t_mou_8	2.76

loc_ic_t2f_mou_8	2.76
onnet_mou_8	2.76
loc_og_t2c_mou_8	2.76
loc_og_t2m_mou_8	2.76
isd_og_mou_8	2.76
loc_og_t2f_mou_8	2.76
offnet_mou_8	2.76
std_ic_mou_8	2.76
loc_ic_t2m_mou_8	2.76
std_ic_t2m_mou_8	2.76
std_ic_t2t_mou_8	2.76
loc_og_mou_8	2.76
spl_ic_mou_8	2.76
loc_ic_t2f_mou_6	0.68
og_others_6	0.68
std_ic_t2f_mou_6	0.68
loc_ic_t2t_mou_6	0.68
spl_og_mou_6	0.68
std_ic_t2m_mou_6	0.68
loc_ic_t2m_mou_6	0.68
loc_ic_mou_6	0.68
std_og_t2m_mou_6	0.68
isd_og_mou_6	0.68
loc_og_t2c_mou_6	0.68
ic_others_6	0.68
onnet_mou_6	0.68
offnet_mou_6	0.68
isd_ic_mou_6	0.68
roam_ic_mou_6	0.68
roam_og_mou_6	0.68
loc_og_t2t_mou_6	0.68
loc_og_t2m_mou_6	0.68
spl_ic_mou_6	0.68
std_ic_t2o_mou_6	0.68
loc_og_t2f_mou_6	0.68
loc_og_mou_6	0.68
std_og_t2t_mou_6	0.68
std_ic_mou_6	0.68
std_ic_t2t_mou_6	0.68
std_og_t2f_mou_6	0.68
std_og_t2c_mou_6	0.68
std_og_mou_6	0.68
std_ic_t2m_mou_7	0.63
std_ic_mou_7	0.63
spl_ic_mou_7	0.63
std_ic_t2f_mou_7	0.63
isd_ic_mou_7	0.63
std_ic_t2o_mou_7	0.63
std_ic_t2t_mou_7	0.63
ic_others_7	0.63
loc_ic_t2f_mou_7	0.63
og_others_7	0.63
loc_ic_mou_7	0.63
std_og_t2f_mou_7	0.63
onnet_mou_7	0.63

offnet_mou_7	0.63
roam_ic_mou_7	0.63
roam_og_mou_7	0.63
loc_og_t2m_mou_7	0.63
loc_og_t2f_mou_7	0.63
loc_og_t2c_mou_7	0.63
loc_og_mou_7	0.63
std_og_t2t_mou_7	0.63
std_og_t2m_mou_7	0.63
loc_og_t2t_mou_7	0.63
std_og_t2c_mou_7	0.63
isd_og_mou_7	0.63
spl_og_mou_7	0.63
std_og_mou_7	0.63
loc_ic_t2m_mou_7	0.63
loc_ic_t2t_mou_7	0.63
monthly_2g_6	0.00
sachet_2g_6	0.00
monthly_2g_9	0.00
monthly_2g_8	0.00
monthly_2g_7	0.00
vol_3g_mb_9	0.00
sep_vbc_3g	0.00
vol_3g_mb_8	0.00
vol_3g_mb_7	0.00
vol_3g_mb_6	0.00
vol_2g_mb_9	0.00
vol_2g_mb_8	0.00
vol_2g_mb_7	0.00
sachet_2g_7	0.00
aug_vbc_3g	0.00
jun_vbc_3g	0.00
jul_vbc_3g	0.00
aon	0.00
sachet_3g_9	0.00
monthly_3g_9	0.00
sachet_3g_6	0.00
monthly_3g_8	0.00
sachet_3g_7	0.00
sachet_2g_8	0.00
sachet_2g_9	0.00
monthly_3g_6	0.00
monthly_3g_7	0.00
sachet_3g_8	0.00
last_day_rch_amt_9	0.00
vol_2g_mb_6	0.00
mobile_number	0.00
last_day_rch_amt_8	0.00
total_ic_mou_8	0.00
std_og_t2o_mou	0.00
loc_ic_t2o_mou	0.00
arpu_6	0.00
arpu_7	0.00
arpu_8	0.00
arpu_9	0.00

total_og_mou_6	0.00
total_og_mou_7	0.00
total_og_mou_8	0.00
total_og_mou_9	0.00
loc_og_t2o_mou	0.00
total_ic_mou_6	0.00
total_ic_mou_7	0.00
total_ic_mou_9	0.00
last_day_rch_amt_7	0.00
total_rech_num_6	0.00
total_rech_num_7	0.00
total_rech_num_8	0.00
total_rech_num_9	0.00
total_rech_amt_6	0.00
total_rech_amt_7	0.00
total_rech_amt_8	0.00
total_rech_amt_9	0.00
max_rech_amt_6	0.00
max_rech_amt_7	0.00
max_rech_amt_8	0.00
max_rech_amt_9	0.00
last_day_rch_amt_6	0.00
avg_rech_amt_6_7	0.00

In [22]:
```
# Listing the columns of MOU Sep(9)
print(((telecom_churn_missing_columns[telecom_churn_missing_columns['null'] == 5.32]).index).to_list())
```

```
['loc_ic_mou_9', 'og_others_9', 'loc_og_t2t_mou_9', 'loc_ic_t2t_mou_9', 'loc_og_t2m_mou_9', 'loc_og_t2f_mou_9',
'loc_og_t2c_mou_9', 'std_ic_t2m_mou_9', 'loc_og_mou_9', 'std_og_t2t_mou_9', 'roam_og_mou_9', 'std_ic_t2o_mou_9'
, 'std_og_t2m_mou_9', 'std_og_t2f_mou_9', 'spl_og_mou_9', 'std_og_t2c_mou_9', 'std_og_mou_9', 'isd_og_mou_9', '
std_ic_t2t_mou_9', 'std_ic_mou_9', 'onnet_mou_9', 'spl_ic_mou_9', 'ic_others_9', 'isd_ic_mou_9', 'loc_ic_t2f_mo
u_9', 'offnet_mou_9', 'loc_ic_t2m_mou_9', 'std_ic_t2f_mou_9', 'roam_ic_mou_9']
```

In [23]:
```
# Creating a dataframe with the condition, in which MOU for Sep(9) are null
telecom_churn_null_mou_9 = telecom_churn[(telecom_churn['loc_og_t2m_mou_9'].isnull()) & (telecom_churn['loc_ic_
  (telecom_churn['loc_og_t2t_mou_9'].isnull()) & (telecom_churn['std_ic_t2t_mou_9'].isnull()) & (telecom_churn[
  (telecom_churn['loc_og_t2c_mou_9'].isnull()) & (telecom_churn['loc_og_mou_9'].isnull()) & (telecom_churn['std
  (telecom_churn['loc_ic_t2m_mou_9'].isnull()) & (telecom_churn['std_og_t2m_mou_9'].isnull()) & (telecom_churn[
  (telecom_churn['std_og_t2c_mou_9'].isnull()) & (telecom_churn['og_others_9'].isnull()) & (telecom_churn['std_
  (telecom_churn['std_ic_t2f_mou_9'].isnull()) & (telecom_churn['isd_og_mou_9'].isnull()) & (telecom_churn['std
  (telecom_churn['isd_ic_mou_9'].isnull()) & (telecom_churn['ic_others_9'].isnull()) & (telecom_churn['std_ic_t
  (telecom_churn['spl_ic_mou_9'].isnull())]

telecom_churn_null_mou_9.head()
```

Out[23]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7
7	7000701601	0.0	0.0	0.0	1069.180	1349.850	3171.480	500.0	57.84	54.68
97	7000589828	0.0	0.0	0.0	374.863	294.023	183.043	0.0	433.59	415.66
111	7001300706	0.0	0.0	0.0	596.301	146.073	0.000	0.0	55.19	3.26
143	7000106299	0.0	0.0	0.0	695.609	39.981	0.000	0.0	1325.91	28.61
188	7000340381	0.0	0.0	0.0	734.641	183.668	0.000	0.0	4.38	0.98

In [24]:
```
telecom_churn_null_mou_9.shape
```

Out[24]:
```
(1590, 178)
```

In [25]:
```
# Deleting the records for which MOU for Sep(9) are null
telecom_churn = telecom_churn.drop(telecom_churn_null_mou_9.index)
```

In [26]:
```
# Again Cheking percent of missing values in columns
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[26]:

	null
isd_og_mou_8	0.55
roam_ic_mou_8	0.55

loc_og_mou_8	0.55
std_ic_t2o_mou_8	0.55
roam_og_mou_8	0.55
loc_ic_t2f_mou_8	0.55
loc_og_t2t_mou_8	0.55
std_ic_t2f_mou_8	0.55
std_og_t2m_mou_8	0.55
loc_og_t2m_mou_8	0.55
std_og_t2t_mou_8	0.55
std_ic_t2m_mou_8	0.55
loc_og_t2f_mou_8	0.55
spl_og_mou_8	0.55
loc_ic_mou_8	0.55
loc_og_t2c_mou_8	0.55
std_ic_t2t_mou_8	0.55
loc_ic_t2m_mou_8	0.55
std_og_t2f_mou_8	0.55
spl_ic_mou_8	0.55
std_ic_mou_8	0.55
offnet_mou_8	0.55
ic_others_8	0.55
og_others_8	0.55
loc_ic_t2t_mou_8	0.55
onnet_mou_8	0.55
isd_ic_mou_8	0.55
std_og_t2c_mou_8	0.55
std_og_mou_8	0.55
isd_og_mou_6	0.50
spl_og_mou_6	0.50
std_og_t2t_mou_6	0.50
loc_ic_t2t_mou_6	0.50
std_og_t2c_mou_6	0.50
std_og_t2m_mou_6	0.50
loc_ic_mou_6	0.50
std_og_mou_6	0.50
loc_ic_t2m_mou_6	0.50
std_og_t2f_mou_6	0.50
loc_ic_t2f_mou_6	0.50
std_ic_t2t_mou_6	0.50
og_others_6	0.50
loc_og_t2t_mou_6	0.50
ic_others_6	0.50
isd_ic_mou_6	0.50
offnet_mou_6	0.50
spl_ic_mou_6	0.50
roam_ic_mou_6	0.50
loc_og_mou_6	0.50
std_ic_mou_6	0.50
roam_og_mou_6	0.50
onnet_mou_6	0.50
std_ic_t2o_mou_6	0.50
loc_og_t2m_mou_6	0.50
std_ic_t2f_mou_6	0.50
loc_og_t2f_mou_6	0.50
loc_og_t2c_mou_6	0.50

std_ic_t2m_mou_6	0.50
loc_ic_t2t_mou_7	0.23
og_others_7	0.23
isd_ic_mou_7	0.23
ic_others_7	0.23
std_ic_t2t_mou_7	0.23
std_ic_mou_7	0.23
spl_ic_mou_7	0.23
loc_ic_t2m_mou_7	0.23
std_ic_t2o_mou_7	0.23
std_ic_t2f_mou_7	0.23
loc_ic_mou_7	0.23
std_ic_t2m_mou_7	0.23
loc_ic_t2f_mou_7	0.23
loc_og_t2t_mou_7	0.23
isd_og_mou_7	0.23
offnet_mou_7	0.23
std_og_t2c_mou_7	0.23
loc_og_mou_7	0.23
onnet_mou_7	0.23
std_og_mou_7	0.23
std_og_t2m_mou_7	0.23
roam_ic_mou_7	0.23
std_og_t2f_mou_7	0.23
loc_og_t2c_mou_7	0.23
spl_og_mou_7	0.23
loc_og_t2f_mou_7	0.23
roam_og_mou_7	0.23
std_og_t2t_mou_7	0.23
loc_og_t2m_mou_7	0.23
sachet_3g_7	0.00
sachet_3g_6	0.00
max_rech_amt_7	0.00
monthly_3g_9	0.00
max_rech_amt_6	0.00
vol_3g_mb_9	0.00
total_rech_amt_9	0.00
total_rech_amt_8	0.00
max_rech_amt_8	0.00
total_rech_amt_6	0.00
sachet_3g_8	0.00
total_rech_num_9	0.00
sachet_3g_9	0.00
aon	0.00
aug_vbc_3g	0.00
total_rech_num_8	0.00
jul_vbc_3g	0.00
jun_vbc_3g	0.00
sep_vbc_3g	0.00
total_rech_amt_7	0.00
monthly_3g_6	0.00
monthly_3g_8	0.00
monthly_3g_7	0.00
monthly_2g_6	0.00
total_rech_num_6	0.00

vol_3g_mb_8	0.00
vol_3g_mb_7	0.00
vol_3g_mb_6	0.00
vol_2g_mb_9	0.00
vol_2g_mb_8	0.00
monthly_2g_7	0.00
vol_2g_mb_7	0.00
monthly_2g_8	0.00
monthly_2g_9	0.00
vol_2g_mb_6	0.00
last_day_rch_amt_9	0.00
sachet_2g_6	0.00
last_day_rch_amt_8	0.00
sachet_2g_7	0.00
last_day_rch_amt_7	0.00
sachet_2g_8	0.00
sachet_2g_9	0.00
last_day_rch_amt_6	0.00
max_rech_amt_9	0.00
total_rech_num_7	0.00
mobile_number	0.00
ic_others_9	0.00
isd_og_mou_9	0.00
std_og_t2c_mou_9	0.00
std_og_t2f_mou_9	0.00
std_og_t2m_mou_9	0.00
std_og_t2t_mou_9	0.00
loc_og_mou_9	0.00
loc_og_t2c_mou_9	0.00
loc_og_t2f_mou_9	0.00
loc_og_t2m_mou_9	0.00
loc_og_t2t_mou_9	0.00
roam_og_mou_9	0.00
roam_ic_mou_9	0.00
offnet_mou_9	0.00
onnet_mou_9	0.00
arpu_9	0.00
arpu_8	0.00
arpu_7	0.00
arpu_6	0.00
loc_ic_t2o_mou	0.00
std_og_t2o_mou	0.00
std_og_mou_9	0.00
spl_og_mou_9	0.00
isd_ic_mou_9	0.00
og_others_9	0.00
spl_ic_mou_9	0.00
total_ic_mou_9	0.00
total_ic_mou_8	0.00
total_ic_mou_7	0.00
total_ic_mou_6	0.00
std_ic_mou_9	0.00
std_ic_t2o_mou_9	0.00
std_ic_t2f_mou_9	0.00
std_ic_t2m_mou_9	0.00

std_ic_t2t_mou_9	0.00
loc_ic_mou_9	0.00
loc_ic_t2f_mou_9	0.00
loc_og_t2o_mou	0.00
loc_ic_t2m_mou_9	0.00
loc_ic_t2t_mou_9	0.00
total_og_mou_9	0.00
total_og_mou_8	0.00
total_og_mou_7	0.00
total_og_mou_6	0.00
avg_rech_amt_6_7	0.00

In [27]:
```
# Listing the columns of MOU Aug(8)
print(((telecom_churn_missing_columns[telecom_churn_missing_columns['null'] == 0.55]).index).to_list())
```

```
['isd_og_mou_8', 'roam_ic_mou_8', 'loc_og_mou_8', 'std_ic_t2o_mou_8', 'roam_og_mou_8', 'loc_ic_t2f_mou_8', 'loc
_og_t2t_mou_8', 'std_ic_t2f_mou_8', 'std_og_t2m_mou_8', 'loc_og_t2m_mou_8', 'std_og_t2t_mou_8', 'std_ic_t2m_mou
_8', 'loc_og_t2f_mou_8', 'spl_og_mou_8', 'loc_ic_mou_8', 'loc_og_t2c_mou_8', 'std_ic_t2t_mou_8', 'loc_ic_t2m_mo
u_8', 'std_og_t2f_mou_8', 'spl_ic_mou_8', 'std_ic_mou_8', 'offnet_mou_8', 'ic_others_8', 'og_others_8', 'loc_ic
_t2t_mou_8', 'onnet_mou_8', 'isd_ic_mou_8', 'std_og_t2c_mou_8', 'std_og_mou_8']
```

In [28]:
```
# Creating a dataframe with the condition, in which MOU for Aug(8) are null
telecom_churn_null_mou_8 = telecom_churn[(telecom_churn['loc_og_t2m_mou_8'].isnull()) & (telecom_churn['loc_ic_
  (telecom_churn['loc_og_t2t_mou_8'].isnull()) & (telecom_churn['std_ic_t2t_mou_8'].isnull()) & (telecom_churn[
  (telecom_churn['loc_og_t2c_mou_8'].isnull()) & (telecom_churn['loc_og_mou_8'].isnull()) & (telecom_churn['std
  (telecom_churn['loc_ic_t2m_mou_8'].isnull()) & (telecom_churn['std_og_t2m_mou_8'].isnull()) & (telecom_churn[
  (telecom_churn['std_og_t2c_mou_8'].isnull()) & (telecom_churn['og_others_8'].isnull()) & (telecom_churn['std_
  (telecom_churn['std_ic_t2f_mou_8'].isnull()) & (telecom_churn['isd_og_mou_8'].isnull()) & (telecom_churn['std
  (telecom_churn['isd_ic_mou_8'].isnull()) & (telecom_churn['ic_others_8'].isnull()) & (telecom_churn['std_ic_t
  (telecom_churn['spl_ic_mou_8'].isnull())]

telecom_churn_null_mou_8.head()
```

Out[28]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	on
375	7002252754	0.0	0.0	0.0	580.477	111.878	0.0	378.881	249.43	39.64	
578	7000248548	0.0	0.0	0.0	569.612	237.289	0.0	4.440	718.01	212.73	
788	7000636808	0.0	0.0	0.0	532.742	546.756	0.0	269.274	1173.39	891.83	
1802	7000516213	0.0	0.0	0.0	810.455	0.000	0.0	0.000	91.33	NaN	
4837	7002192662	0.0	0.0	0.0	649.150	149.572	0.0	0.250	1354.24	85.13	

In [29]:
```
# Deleting the records for which MOU for Aug(8) are null
telecom_churn = telecom_churn.drop(telecom_churn_null_mou_8.index)
```

In [30]:
```
# Again cheking percent of missing values in columns
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[30]:

	null
roam_ic_mou_6	0.44
spl_og_mou_6	0.44
og_others_6	0.44
loc_ic_t2t_mou_6	0.44
loc_og_t2m_mou_6	0.44
loc_og_t2c_mou_6	0.44
loc_ic_t2m_mou_6	0.44
isd_og_mou_6	0.44
loc_og_t2t_mou_6	0.44
std_og_t2m_mou_6	0.44
loc_ic_t2f_mou_6	0.44
ic_others_6	0.44
roam_og_mou_6	0.44
loc_ic_mou_6	0.44
std_og_mou_6	0.44
loc_og_t2f_mou_6	0.44
isd_ic_mou_6	0.44

std_ic_t2t_mou_6	0.44
std_ic_mou_6	0.44
std_og_t2t_mou_6	0.44
std_ic_t2o_mou_6	0.44
std_og_t2f_mou_6	0.44
std_ic_t2f_mou_6	0.44
spl_ic_mou_6	0.44
onnet_mou_6	0.44
std_og_t2c_mou_6	0.44
std_ic_t2m_mou_6	0.44
offnet_mou_6	0.44
loc_og_mou_6	0.44
std_og_t2f_mou_7	0.16
isd_og_mou_7	0.16
std_og_mou_7	0.16
loc_ic_t2f_mou_7	0.16
ic_others_7	0.16
spl_og_mou_7	0.16
og_others_7	0.16
loc_ic_t2t_mou_7	0.16
loc_ic_t2m_mou_7	0.16
loc_ic_mou_7	0.16
isd_ic_mou_7	0.16
std_ic_t2t_mou_7	0.16
std_ic_t2m_mou_7	0.16
spl_ic_mou_7	0.16
std_ic_t2f_mou_7	0.16
std_ic_t2o_mou_7	0.16
std_og_t2m_mou_7	0.16
std_og_t2c_mou_7	0.16
loc_og_t2m_mou_7	0.16
loc_og_mou_7	0.16
onnet_mou_7	0.16
offnet_mou_7	0.16
roam_ic_mou_7	0.16
roam_og_mou_7	0.16
loc_og_t2t_mou_7	0.16
loc_og_t2f_mou_7	0.16
loc_og_t2c_mou_7	0.16
std_ic_mou_7	0.16
std_og_t2t_mou_7	0.16
std_ic_mou_9	0.00
spl_ic_mou_8	0.00
monthly_3g_6	0.00
vol_3g_mb_7	0.00
isd_ic_mou_9	0.00
isd_ic_mou_8	0.00
monthly_3g_7	0.00
monthly_3g_8	0.00
vol_2g_mb_7	0.00
monthly_3g_9	0.00
sachet_3g_6	0.00
vol_2g_mb_8	0.00
spl_ic_mou_9	0.00
sachet_3g_7	0.00

ic_others_8	0.00
sachet_3g_8	0.00
sachet_3g_9	0.00
aon	0.00
aug_vbc_3g	0.00
total_ic_mou_9	0.00
total_ic_mou_8	0.00
jul_vbc_3g	0.00
jun_vbc_3g	0.00
total_ic_mou_7	0.00
total_ic_mou_6	0.00
sep_vbc_3g	0.00
sachet_2g_9	0.00
vol_3g_mb_8	0.00
last_day_rch_amt_8	0.00
sachet_2g_8	0.00
last_day_rch_amt_7	0.00
vol_3g_mb_9	0.00
monthly_2g_6	0.00
last_day_rch_amt_6	0.00
max_rech_amt_9	0.00
last_day_rch_amt_9	0.00
monthly_2g_7	0.00
max_rech_amt_8	0.00
max_rech_amt_7	0.00
monthly_2g_8	0.00
vol_2g_mb_6	0.00
max_rech_amt_6	0.00
total_rech_amt_9	0.00
monthly_2g_9	0.00
sachet_2g_6	0.00
total_rech_amt_8	0.00
total_rech_amt_7	0.00
total_rech_amt_6	0.00
total_rech_num_9	0.00
total_rech_num_8	0.00
total_rech_num_7	0.00
vol_3g_mb_6	0.00
sachet_2g_7	0.00
total_rech_num_6	0.00
ic_others_9	0.00
vol_2g_mb_9	0.00
mobile_number	0.00
std_ic_mou_8	0.00
loc_og_t2t_mou_8	0.00
std_og_t2m_mou_9	0.00
std_og_t2m_mou_8	0.00
std_og_t2t_mou_9	0.00
std_og_t2t_mou_8	0.00
loc_og_mou_9	0.00
loc_og_mou_8	0.00
loc_og_t2c_mou_9	0.00
loc_og_t2c_mou_8	0.00
loc_og_t2f_mou_9	0.00

loc_og_t2f_mou_8	0.00
loc_og_t2m_mou_9	0.00
loc_og_t2m_mou_8	0.00
loc_og_t2t_mou_9	0.00
roam_og_mou_9	0.00
std_og_t2f_mou_9	0.00
roam_og_mou_8	0.00
roam_ic_mou_9	0.00
roam_ic_mou_8	0.00
offnet_mou_9	0.00
offnet_mou_8	0.00
onnet_mou_9	0.00
onnet_mou_8	0.00
arpu_9	0.00
arpu_8	0.00
arpu_7	0.00
arpu_6	0.00
loc_ic_t2o_mou	0.00
std_og_t2o_mou	0.00
std_og_t2f_mou_8	0.00
std_og_t2c_mou_8	0.00
std_ic_t2o_mou_9	0.00
loc_ic_t2m_mou_8	0.00
std_ic_t2o_mou_8	0.00
std_ic_t2f_mou_9	0.00
std_ic_t2f_mou_8	0.00
std_ic_t2m_mou_9	0.00
std_ic_t2m_mou_8	0.00
std_ic_t2t_mou_9	0.00
std_ic_t2t_mou_8	0.00
loc_ic_mou_9	0.00
loc_ic_mou_8	0.00
loc_ic_t2f_mou_9	0.00
loc_ic_t2f_mou_8	0.00
loc_og_t2o_mou	0.00
loc_ic_t2m_mou_9	0.00
loc_ic_t2t_mou_9	0.00
std_og_t2c_mou_9	0.00
loc_ic_t2t_mou_8	0.00
total_og_mou_9	0.00
total_og_mou_8	0.00
total_og_mou_7	0.00
total_og_mou_6	0.00
og_others_9	0.00
og_others_8	0.00
spl_og_mou_9	0.00
spl_og_mou_8	0.00
isd_og_mou_9	0.00
isd_og_mou_8	0.00
std_og_mou_9	0.00
std_og_mou_8	0.00
avg_rech_amt_6_7	0.00

In [31]:
```
# Listing the columns of MOU Jun(6)
print(((telecom_churn_missing_columns[telecom_churn_missing_columns['null'] == 0.44]).index).to_list())
```

```
['roam_ic_mou_6', 'spl_og_mou_6', 'og_others_6', 'loc_ic_t2t_mou_6', 'loc_og_t2m_mou_6', 'loc_og_t2c_mou_6', 'l
oc_ic_t2m_mou_6', 'isd_og_mou_6', 'loc_og_t2t_mou_6', 'std_og_t2m_mou_6', 'loc_ic_t2f_mou_6', 'ic_others_6', 'r
oam_og_mou_6', 'loc_ic_mou_6', 'std_og_mou_6', 'loc_og_t2f_mou_6', 'isd_ic_mou_6', 'std_ic_t2t_mou_6', 'std_ic_
mou_6', 'std_og_t2t_mou_6', 'std_ic_t2o_mou_6', 'std_og_t2f_mou_6', 'std_ic_t2f_mou_6', 'spl_ic_mou_6', 'onnet_
mou_6', 'std_og_t2c_mou_6', 'std_ic_t2m_mou_6', 'offnet_mou_6', 'loc_og_mou_6']
```

In [32]:
```
# Creating a dataframe with the condition, in which MOU for Jun(6) are null
telecom_churn_null_mou_6 = telecom_churn[(telecom_churn['loc_og_t2m_mou_6'].isnull()) & (telecom_churn['loc_ic_
  (telecom_churn['loc_og_t2t_mou_6'].isnull()) & (telecom_churn['std_ic_t2t_mou_6'].isnull()) & (telecom_churn[
  (telecom_churn['loc_og_t2c_mou_6'].isnull()) & (telecom_churn['loc_og_mou_6'].isnull()) & (telecom_churn['std
  (telecom_churn['loc_ic_t2m_mou_6'].isnull()) & (telecom_churn['std_og_t2m_mou_6'].isnull()) & (telecom_churn[
  (telecom_churn['std_og_t2c_mou_6'].isnull()) & (telecom_churn['og_others_6'].isnull()) & (telecom_churn['std_
  (telecom_churn['std_ic_t2f_mou_6'].isnull()) & (telecom_churn['isd_og_mou_6'].isnull()) & (telecom_churn['std
  (telecom_churn['isd_ic_mou_6'].isnull()) & (telecom_churn['ic_others_6'].isnull()) & (telecom_churn['std_ic_t
  (telecom_churn['spl_ic_mou_6'].isnull())]

telecom_churn_null_mou_6.head()
```

Out[32]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	or
77	7001328263	0.0	0.0	0.0	30.000	82.378	674.950	158.710	NaN	34.23	
364	7002168045	0.0	0.0	0.0	0.000	792.112	989.368	923.040	NaN	433.49	
423	7000635248	0.0	0.0	0.0	213.802	304.194	149.710	329.643	NaN	0.00	
934	7002152278	0.0	0.0	0.0	48.000	764.152	500.030	194.400	NaN	14.24	
1187	7000486275	0.0	0.0	0.0	0.000	757.170	995.719	0.000	NaN	1366.71	

In [33]:
```
# Deleting the records for which MOU for Jun(6) are null
telecom_churn = telecom_churn.drop(telecom_churn_null_mou_6.index)
```

In [34]:
```
# Again cheking percent of missing values in columns
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[34]:

	null
loc_ic_t2f_mou_7	0.12
isd_ic_mou_7	0.12
loc_og_t2f_mou_7	0.12
loc_og_t2c_mou_7	0.12
loc_og_mou_7	0.12
std_og_t2t_mou_7	0.12
std_og_t2f_mou_7	0.12
std_og_t2c_mou_7	0.12
std_og_mou_7	0.12
ic_others_7	0.12
isd_og_mou_7	0.12
spl_og_mou_7	0.12
loc_og_t2t_mou_7	0.12
og_others_7	0.12
spl_ic_mou_7	0.12
loc_ic_t2t_mou_7	0.12
std_ic_mou_7	0.12
loc_ic_t2m_mou_7	0.12
std_ic_t2o_mou_7	0.12
std_ic_t2f_mou_7	0.12
loc_ic_mou_7	0.12
std_ic_t2t_mou_7	0.12
loc_og_t2m_mou_7	0.12
std_og_t2m_mou_7	0.12
std_ic_t2m_mou_7	0.12
roam_ic_mou_7	0.12
onnet_mou_7	0.12
roam_og_mou_7	0.12
offnet_mou_7	0.12
isd_ic_mou_8	0.00
monthly_3g_9	0.00

sachet_3g_6	0.00
isd_ic_mou_6	0.00
spl_ic_mou_9	0.00
std_ic_mou_9	0.00
isd_ic_mou_9	0.00
spl_ic_mou_8	0.00
sachet_3g_7	0.00
spl_ic_mou_6	0.00
total_ic_mou_9	0.00
total_ic_mou_8	0.00
total_ic_mou_7	0.00
total_ic_mou_6	0.00
sachet_3g_8	0.00
monthly_3g_8	0.00
std_ic_mou_8	0.00
sep_vbc_3g	0.00
std_ic_t2m_mou_8	0.00
std_ic_t2m_mou_9	0.00
std_ic_t2f_mou_6	0.00
jun_vbc_3g	0.00
jul_vbc_3g	0.00
std_ic_t2f_mou_8	0.00
std_ic_t2f_mou_9	0.00
std_ic_t2o_mou_6	0.00
aug_vbc_3g	0.00
std_ic_t2o_mou_8	0.00
std_ic_t2o_mou_9	0.00
aon	0.00
std_ic_mou_6	0.00
sachet_3g_9	0.00
ic_others_6	0.00
monthly_2g_8	0.00
ic_others_8	0.00
monthly_2g_9	0.00
sachet_2g_8	0.00
last_day_rch_amt_8	0.00
last_day_rch_amt_9	0.00
vol_2g_mb_6	0.00
sachet_2g_7	0.00
vol_2g_mb_7	0.00
vol_2g_mb_8	0.00
vol_2g_mb_9	0.00
vol_3g_mb_6	0.00
vol_3g_mb_7	0.00
vol_3g_mb_8	0.00
sachet_2g_6	0.00
vol_3g_mb_9	0.00
monthly_2g_6	0.00
monthly_2g_7	0.00
last_day_rch_amt_7	0.00
last_day_rch_amt_6	0.00
max_rech_amt_9	0.00
total_rech_amt_6	0.00
ic_others_9	0.00
total_rech_num_6	0.00

total_rech_num_7	0.00
monthly_3g_6	0.00
total_rech_num_8	0.00
total_rech_num_9	0.00
total_rech_amt_7	0.00
sachet_2g_9	0.00
total_rech_amt_8	0.00
total_rech_amt_9	0.00
std_ic_t2m_mou_6	0.00
max_rech_amt_6	0.00
max_rech_amt_7	0.00
max_rech_amt_8	0.00
monthly_3g_7	0.00
mobile_number	0.00
std_ic_t2t_mou_9	0.00
loc_og_t2c_mou_6	0.00
loc_og_t2t_mou_9	0.00
loc_og_t2m_mou_6	0.00
loc_og_t2m_mou_8	0.00
loc_og_t2m_mou_9	0.00
loc_og_t2f_mou_6	0.00
loc_og_t2f_mou_8	0.00
loc_og_t2f_mou_9	0.00
loc_og_t2c_mou_8	0.00
loc_og_t2t_mou_6	0.00
loc_og_t2c_mou_9	0.00
loc_og_mou_6	0.00
loc_og_mou_8	0.00
loc_og_mou_9	0.00
std_og_t2t_mou_6	0.00
std_og_t2t_mou_8	0.00
std_og_t2t_mou_9	0.00
loc_og_t2t_mou_8	0.00
roam_og_mou_9	0.00
std_ic_t2t_mou_8	0.00
onnet_mou_8	0.00
std_og_t2o_mou	0.00
loc_ic_t2o_mou	0.00
arpu_6	0.00
arpu_7	0.00
arpu_8	0.00
arpu_9	0.00
onnet_mou_6	0.00
onnet_mou_9	0.00
roam_og_mou_8	0.00
offnet_mou_6	0.00
offnet_mou_8	0.00
offnet_mou_9	0.00
roam_ic_mou_6	0.00
roam_ic_mou_8	0.00
roam_ic_mou_9	0.00
roam_og_mou_6	0.00
std_og_t2m_mou_6	0.00
std_og_t2m_mou_8	0.00

std_og_t2m_mou_9	0.00
loc_ic_t2m_mou_9	0.00
total_og_mou_8	0.00
total_og_mou_9	0.00
loc_ic_t2t_mou_6	0.00
loc_ic_t2t_mou_8	0.00
loc_ic_t2t_mou_9	0.00
loc_ic_t2m_mou_6	0.00
loc_ic_t2m_mou_8	0.00
loc_ic_t2f_mou_6	0.00
std_og_t2f_mou_6	0.00
loc_og_t2o_mou	0.00
loc_ic_t2f_mou_8	0.00
loc_ic_t2f_mou_9	0.00
loc_ic_mou_6	0.00
loc_ic_mou_8	0.00
loc_ic_mou_9	0.00
std_ic_t2t_mou_6	0.00
total_og_mou_7	0.00
total_og_mou_6	0.00
og_others_9	0.00
og_others_8	0.00
std_og_t2f_mou_8	0.00
std_og_t2f_mou_9	0.00
std_og_t2c_mou_6	0.00
std_og_t2c_mou_8	0.00
std_og_t2c_mou_9	0.00
std_og_mou_6	0.00
std_og_mou_8	0.00
std_og_mou_9	0.00
isd_og_mou_6	0.00
isd_og_mou_8	0.00
isd_og_mou_9	0.00
spl_og_mou_6	0.00
spl_og_mou_8	0.00
spl_og_mou_9	0.00
og_others_6	0.00
avg_rech_amt_6_7	0.00

In [35]:
```
# Listing the columns of MOU Jul(7)
print(((telecom_churn_missing_columns[telecom_churn_missing_columns['null'] == 0.12]).index).to_list())
```

```
['loc_ic_t2f_mou_7', 'isd_ic_mou_7', 'loc_og_t2f_mou_7', 'loc_og_t2c_mou_7', 'loc_og_mou_7', 'std_og_t2t_mou_7'
, 'std_og_t2f_mou_7', 'std_og_t2c_mou_7', 'std_og_mou_7', 'ic_others_7', 'isd_og_mou_7', 'spl_og_mou_7', 'loc_o
g_t2t_mou_7', 'og_others_7', 'spl_ic_mou_7', 'loc_ic_t2t_mou_7', 'std_ic_mou_7', 'loc_ic_t2m_mou_7', 'std_ic_t2
o_mou_7', 'std_ic_t2f_mou_7', 'loc_ic_mou_7', 'std_ic_t2t_mou_7', 'loc_og_t2m_mou_7', 'std_og_t2m_mou_7', 'std_
ic_t2m_mou_7', 'roam_ic_mou_7', 'onnet_mou_7', 'roam_og_mou_7', 'offnet_mou_7']
```

In [36]:
```
# Creating a dataframe with the condition, in which MOU for Jul(7) are null
telecom_churn_null_mou_7 = telecom_churn[(telecom_churn['loc_og_t2m_mou_7'].isnull()) & (telecom_churn['loc_ic_
  (telecom_churn['loc_og_t2t_mou_7'].isnull()) & (telecom_churn['std_ic_t2t_mou_7'].isnull()) & (telecom_churn[
  (telecom_churn['loc_og_t2c_mou_7'].isnull()) & (telecom_churn['loc_og_mou_7'].isnull()) & (telecom_churn['std
  (telecom_churn['loc_ic_t2m_mou_7'].isnull()) & (telecom_churn['std_og_t2m_mou_7'].isnull()) & (telecom_churn[
  (telecom_churn['std_og_t2c_mou_7'].isnull()) & (telecom_churn['og_others_7'].isnull()) & (telecom_churn['std_
  (telecom_churn['std_ic_t2f_mou_7'].isnull()) & (telecom_churn['isd_og_mou_7'].isnull()) & (telecom_churn['std
  (telecom_churn['isd_ic_mou_7'].isnull()) & (telecom_churn['ic_others_7'].isnull()) & (telecom_churn['std_ic_t
  (telecom_churn['spl_ic_mou_7'].isnull())]

telecom_churn_null_mou_7.head()
```

Out[36]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7
5616	7001238202	0.0	0.0	0.0	760.815	531.088	992.818	1144.676	324.91	NaN
9451	7001477649	0.0	0.0	0.0	1129.566	0.000	128.252	802.648	11.89	NaN
9955	7001658068	0.0	0.0	0.0	925.028	189.000	789.761	445.707	46.39	NaN
10724	7001391499	0.0	0.0	0.0	894.818	85.000	207.040	363.314	117.21	NaN
12107	7000131738	0.0	0.0	0.0	1803.475	0.000	0.600	25.243	1742.61	NaN

In [37]:
```
# Deleting the records for which MOU for Jul(7) are null
telecom_churn = telecom_churn.drop(telecom_churn_null_mou_7.index)
```

In [38]:
```
# Again cheking percent of missing values in columns
telecom_churn_missing_columns = (round(((telecom_churn.isnull().sum()/len(telecom_churn.index))*100),2).to_fram
telecom_churn_missing_columns
```

Out[38]:

	null
mobile_number	0.0
total_rech_num_7	0.0
std_ic_mou_7	0.0
std_ic_mou_8	0.0
std_ic_mou_9	0.0
total_ic_mou_6	0.0
total_ic_mou_7	0.0
total_ic_mou_8	0.0
total_ic_mou_9	0.0
spl_ic_mou_6	0.0
spl_ic_mou_7	0.0
spl_ic_mou_8	0.0
spl_ic_mou_9	0.0
isd_ic_mou_6	0.0
isd_ic_mou_7	0.0
isd_ic_mou_8	0.0
isd_ic_mou_9	0.0
ic_others_6	0.0
ic_others_7	0.0
ic_others_8	0.0
ic_others_9	0.0
std_ic_mou_6	0.0
std_ic_t2o_mou_9	0.0
std_ic_t2o_mou_8	0.0
std_ic_t2t_mou_9	0.0
loc_ic_t2f_mou_9	0.0
loc_ic_mou_6	0.0
loc_ic_mou_7	0.0
loc_ic_mou_8	0.0
loc_ic_mou_9	0.0
std_ic_t2t_mou_6	0.0
std_ic_t2t_mou_7	0.0
std_ic_t2t_mou_8	0.0
std_ic_t2m_mou_6	0.0
std_ic_t2o_mou_7	0.0
std_ic_t2m_mou_7	0.0
std_ic_t2m_mou_8	0.0
std_ic_t2m_mou_9	0.0
std_ic_t2f_mou_6	0.0
std_ic_t2f_mou_7	0.0
std_ic_t2f_mou_8	0.0
std_ic_t2f_mou_9	0.0

std_ic_t2o_mou_6	0.0
total_rech_num_6	0.0
total_rech_num_8	0.0
loc_og_t2o_mou	0.0
total_rech_num_9	0.0
monthly_2g_8	0.0
monthly_2g_9	0.0
sachet_2g_6	0.0
sachet_2g_7	0.0
sachet_2g_8	0.0
sachet_2g_9	0.0
monthly_3g_6	0.0
monthly_3g_7	0.0
monthly_3g_8	0.0
monthly_3g_9	0.0
sachet_3g_6	0.0
sachet_3g_7	0.0
sachet_3g_8	0.0
sachet_3g_9	0.0
aon	0.0
aug_vbc_3g	0.0
jul_vbc_3g	0.0
jun_vbc_3g	0.0
sep_vbc_3g	0.0
monthly_2g_7	0.0
monthly_2g_6	0.0
vol_3g_mb_9	0.0
last_day_rch_amt_6	0.0
total_rech_amt_6	0.0
total_rech_amt_7	0.0
total_rech_amt_8	0.0
total_rech_amt_9	0.0
max_rech_amt_6	0.0
max_rech_amt_7	0.0
max_rech_amt_8	0.0
max_rech_amt_9	0.0
last_day_rch_amt_7	0.0
vol_3g_mb_8	0.0
last_day_rch_amt_8	0.0
last_day_rch_amt_9	0.0
vol_2g_mb_6	0.0
vol_2g_mb_7	0.0
vol_2g_mb_8	0.0
vol_2g_mb_9	0.0
vol_3g_mb_6	0.0
vol_3g_mb_7	0.0
loc_ic_t2f_mou_8	0.0
loc_ic_t2f_mou_7	0.0
loc_ic_t2f_mou_6	0.0
loc_ic_t2m_mou_9	0.0
loc_og_t2t_mou_6	0.0
loc_og_t2t_mou_7	0.0
loc_og_t2t_mou_8	0.0
loc_og_t2t_mou_9	0.0
loc_og_t2m_mou_6	0.0

loc_og_t2m_mou_7	0.0
loc_og_t2m_mou_8	0.0
loc_og_t2m_mou_9	0.0
loc_og_t2f_mou_6	0.0
loc_og_t2f_mou_7	0.0
loc_og_t2f_mou_8	0.0
loc_og_t2f_mou_9	0.0
loc_og_t2c_mou_6	0.0
loc_og_t2c_mou_7	0.0
loc_og_t2c_mou_8	0.0
loc_og_t2c_mou_9	0.0
loc_og_mou_6	0.0
loc_og_mou_7	0.0
loc_og_mou_8	0.0
roam_og_mou_9	0.0
roam_og_mou_8	0.0
roam_og_mou_7	0.0
onnet_mou_8	0.0
std_og_t2o_mou	0.0
loc_ic_t2o_mou	0.0
arpu_6	0.0
arpu_7	0.0
arpu_8	0.0
arpu_9	0.0
onnet_mou_6	0.0
onnet_mou_7	0.0
onnet_mou_9	0.0
roam_og_mou_6	0.0
offnet_mou_6	0.0
offnet_mou_7	0.0
offnet_mou_8	0.0
offnet_mou_9	0.0
roam_ic_mou_6	0.0
roam_ic_mou_7	0.0
roam_ic_mou_8	0.0
roam_ic_mou_9	0.0
loc_og_mou_9	0.0
std_og_t2t_mou_6	0.0
std_og_t2t_mou_7	0.0
total_og_mou_7	0.0
spl_og_mou_7	0.0
spl_og_mou_8	0.0
spl_og_mou_9	0.0
og_others_6	0.0
og_others_7	0.0
og_others_8	0.0
og_others_9	0.0
total_og_mou_6	0.0
total_og_mou_8	0.0
isd_og_mou_9	0.0
total_og_mou_9	0.0
loc_ic_t2t_mou_6	0.0
loc_ic_t2t_mou_7	0.0
loc_ic_t2t_mou_8	0.0

loc_ic_t2t_mou_9	0.0
loc_ic_t2m_mou_6	0.0
loc_ic_t2m_mou_7	0.0
loc_ic_t2m_mou_8	0.0
spl_og_mou_6	0.0
isd_og_mou_8	0.0
std_og_t2t_mou_8	0.0
std_og_t2f_mou_9	0.0
std_og_t2t_mou_9	0.0
std_og_t2m_mou_6	0.0
std_og_t2m_mou_7	0.0
std_og_t2m_mou_8	0.0
std_og_t2m_mou_9	0.0
std_og_t2f_mou_6	0.0
std_og_t2f_mou_7	0.0
std_og_t2f_mou_8	0.0
std_og_t2c_mou_6	0.0
isd_og_mou_7	0.0
std_og_t2c_mou_7	0.0
std_og_t2c_mou_8	0.0
std_og_t2c_mou_9	0.0
std_og_mou_6	0.0
std_og_mou_7	0.0
std_og_mou_8	0.0
std_og_mou_9	0.0
isd_og_mou_6	0.0
avg_rech_amt_6_7	0.0

In [39]:
```
telecom_churn.shape
```

Out[39]:
```
(27991, 178)
```

In [40]:
```
telecom_churn['churn'] = np.where((telecom_churn['total_ic_mou_9']==0) & (telecom_churn['total_og_mou_9']==0) &
```

In [41]:
```
telecom_churn.head()
```

Out[41]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	onn
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	166.787	413.69	351.03	
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	322.732	501.76	108.39	
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	206.490	50.51	74.01	
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	257.583	1185.91	9.28	
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	578.596	102.41	132.11	

In [42]:
```
# List the columns for churn month(9)
col_9 = [col for col in telecom_churn.columns.to_list() if '_9' in col]
print(col_9)
```

```
['arpu_9', 'onnet_mou_9', 'offnet_mou_9', 'roam_ic_mou_9', 'roam_og_mou_9', 'loc_og_t2t_mou_9', 'loc_og_t2m_mou
_9', 'loc_og_t2f_mou_9', 'loc_og_t2c_mou_9', 'loc_og_mou_9', 'std_og_t2t_mou_9', 'std_og_t2m_mou_9', 'std_og_t2
f_mou_9', 'std_og_t2c_mou_9', 'std_og_mou_9', 'isd_og_mou_9', 'spl_og_mou_9', 'og_others_9', 'total_og_mou_9',
'loc_ic_t2t_mou_9', 'loc_ic_t2m_mou_9', 'loc_ic_t2f_mou_9', 'loc_ic_mou_9', 'std_ic_t2t_mou_9', 'std_ic_t2m_mou
_9', 'std_ic_t2f_mou_9', 'std_ic_t2o_mou_9', 'std_ic_mou_9', 'total_ic_mou_9', 'spl_ic_mou_9', 'isd_ic_mou_9',
'ic_others_9', 'total_rech_num_9', 'total_rech_amt_9', 'max_rech_amt_9', 'last_day_rch_amt_9', 'vol_2g_mb_9', '
vol_3g_mb_9', 'monthly_2g_9', 'sachet_2g_9', 'monthly_3g_9', 'sachet_3g_9']
```

In [43]:
```
# Deleting the churn month columns
telecom_churn = telecom_churn.drop(col_9, axis=1)
```

In [44]:
```
round(100*(telecom_churn['churn'].mean()),2)
```

Out[44]:
```
3.39
```

In [45]:
```
telecom_churn['mobile_number'] = telecom_churn['mobile_number'].astype(object)
telecom_churn['churn'] = telecom_churn['churn'].astype(object)
```

In [46]:
```
telecom_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27991 entries, 8 to 99997
Columns: 137 entries, mobile_number to churn
dtypes: float64(110), int64(25), object(2)
memory usage: 29.5+ MB
```

In [47]:
```
# List only the numeric columns
numeric_cols = telecom_churn.select_dtypes(exclude=['object']).columns
print(numeric_cols)
```

```
Index(['loc_og_t2o_mou', 'std_og_t2o_mou', 'loc_ic_t2o_mou', 'arpu_6',
       'arpu_7', 'arpu_8', 'onnet_mou_6', 'onnet_mou_7', 'onnet_mou_8',
       'offnet_mou_6',
       ...
       'monthly_3g_8', 'sachet_3g_6', 'sachet_3g_7', 'sachet_3g_8', 'aon',
       'aug_vbc_3g', 'jul_vbc_3g', 'jun_vbc_3g', 'sep_vbc_3g',
       'avg_rech_amt_6_7'],
      dtype='object', length=135)
```

In [48]:
```
# Removing outliers below 10th and above 90th percentile
for col in numeric_cols:
    q1 = telecom_churn[col].quantile(0.10)
    q3 = telecom_churn[col].quantile(0.90)
    iqr = q3-q1
    range_low  = q1-1.5*iqr
    range_high = q3+1.5*iqr
    # Assigning the filtered dataset into data
    data_frame = telecom_churn.loc[(telecom_churn[col] > range_low) & (telecom_churn[col] < range_high)]

data_frame.shape
```

Out[48]:
```
(27705, 137)
```

In [49]:
```
# List the columns of total mou, rech_num and rech_amt
[total for total in data_frame.columns.to_list() if 'total' in total]
```

Out[49]:
```
['total_og_mou_6',
 'total_og_mou_7',
 'total_og_mou_8',
 'total_ic_mou_6',
 'total_ic_mou_7',
 'total_ic_mou_8',
 'total_rech_num_6',
 'total_rech_num_7',
 'total_rech_num_8',
 'total_rech_amt_6',
 'total_rech_amt_7',
 'total_rech_amt_8']
```

In [50]:
```
# Total mou at good phase incoming and outgoing
data_frame['total_mou_good'] = (data_frame['total_og_mou_6'] + data_frame['total_ic_mou_6'])
```

In [51]:
```
# Avg. mou at action phase
# We are taking average because there are two months(7 and 8) in action phase
data_frame['avg_mou_action'] = (data_frame['total_og_mou_7'] + data_frame['total_og_mou_8'] + data_frame['total
```

In [52]:
```
# Difference avg_mou_good and avg_mou_action
data_frame['diff_mou'] = data_frame['avg_mou_action'] - data_frame['total_mou_good']
```

In [53]:
```
# Checking whether the mou has decreased in action phase
data_frame['decrease_mou_action'] = np.where((data_frame['diff_mou'] < 0), 1, 0)
```

In [54]:
```
data_frame.head()
```

Out[54]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	413.69	351.03	35.08
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	501.76	108.39	534.24
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	50.51	74.01	70.61
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	1185.91	9.28	7.79
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	102.41	132.11	85.14

In [55]:
```
# Avg rech number at action phase
data_frame['avg_rech_num_action'] = (data_frame['total_rech_num_7'] + data_frame['total_rech_num_8'])/2
```

In [56]:
```
# Difference total_rech_num_6 and avg_rech_action
data_frame['diff_rech_num'] = data_frame['avg_rech_num_action'] - data_frame['total_rech_num_6']
```

In [57]:
```
# Checking if rech_num has decreased in action phase
data_frame['decrease_rech_num_action'] = np.where((data_frame['diff_rech_num'] < 0), 1, 0)
```

In [58]:
```
data_frame.head()
```

Out[58]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	413.69	351.03	35.08
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	501.76	108.39	534.24
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	50.51	74.01	70.61
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	1185.91	9.28	7.79
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	102.41	132.11	85.14

In [59]:
```
# Avg rech_amt in action phase
data_frame['avg_rech_amt_action'] = (data_frame['total_rech_amt_7'] + data_frame['total_rech_amt_8'])/2
```

In [60]:
```
# Difference of action phase rech amt and good phase rech amt
data_frame['diff_rech_amt'] = data_frame['avg_rech_amt_action'] - data_frame['total_rech_amt_6']
```

In [61]:
```
# Checking if rech_amt has decreased in action phase
data_frame['decrease_rech_amt_action'] = np.where((data_frame['diff_rech_amt'] < 0), 1, 0)
```

In [62]:
```
data_frame.head()
```

Out[62]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	413.69	351.03	35.08
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	501.76	108.39	534.24
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	50.51	74.01	70.61
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	1185.91	9.28	7.79
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	102.41	132.11	85.14

In [63]:
```
# ARUP in action phase
data_frame['avg_arpu_action'] = (data_frame['arpu_7'] + data_frame['arpu_8'])/2
```

In [64]:
```
# Difference of good and action phase ARPU
data_frame['diff_arpu'] = data_frame['avg_arpu_action'] - data_frame['arpu_6']
```

In [65]:
```
# Checking whether the arpu has decreased on the action month
data_frame['decrease_arpu_action'] = np.where(data_frame['diff_arpu'] < 0, 1, 0)
```

In [66]:
```
data_frame.head()
```

Out[66]:

	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	413.69	351.03	35.08
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	501.76	108.39	534.24
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	50.51	74.01	70.61
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	1185.91	9.28	7.79
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	102.41	132.11	85.14

In [67]:
```
# VBC in action phase
data_frame['avg_vbc_3g_action'] = (data_frame['jul_vbc_3g'] + data_frame['aug_vbc_3g'])/2
```

In [68]:
```
# Difference of good and action phase VBC
data_frame['diff_vbc'] = data_frame['avg_vbc_3g_action'] - data_frame['jun_vbc_3g']
```

In [69]:
```
# Checking whether the VBC has decreased on the action month
data_frame['decrease_vbc_action'] = np.where(data_frame['diff_vbc'] < 0 , 1, 0)
```

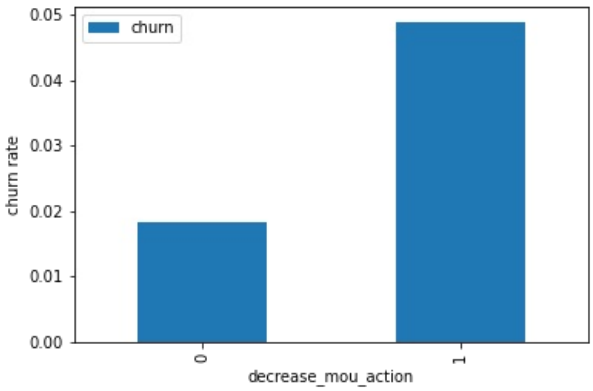
In [70]:
```
data_frame.head (10)
```

Out[70]:

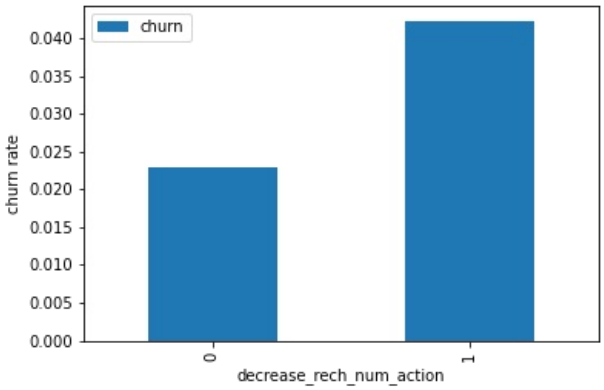
	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	413.69	351.03	35.
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	501.76	108.39	534.
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	50.51	74.01	70.
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	1185.91	9.28	7.
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	102.41	132.11	85.
24	7001125315	0.0	0.0	0.0	422.050	359.730	354.793	124.19	55.19	141.
33	7000149764	0.0	0.0	0.0	977.020	2362.833	409.230	0.00	0.00	0.
38	7000815202	0.0	0.0	0.0	363.987	486.558	393.909	248.99	619.96	666.
41	7000721289	0.0	0.0	0.0	482.832	425.764	229.769	86.39	118.88	80.
48	7000294396	0.0	0.0	0.0	1873.271	575.927	179.218	2061.69	881.43	156.

In [71]:
```
# Converting churn column to int in order to do aggfunc in the pivot table
data_frame['churn'] = data_frame['churn'].astype('int64')
```

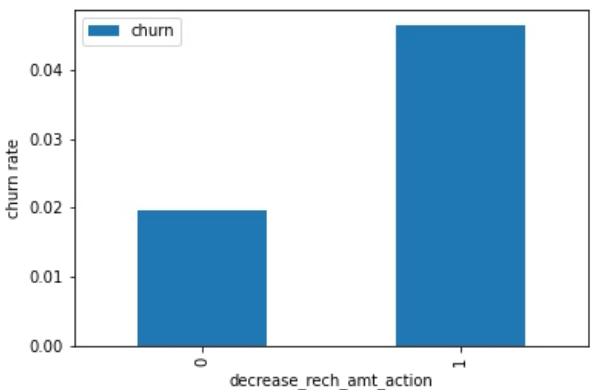
In [72]:
```
data_frame.pivot_table(values='churn', index='decrease_mou_action', aggfunc='mean').plot.bar()
plt.ylabel('churn rate')
plt.show()
```



In [73]:
```
data_frame.pivot_table(values='churn', index='decrease_rech_num_action', aggfunc='mean').plot.bar()
plt.ylabel('churn rate')
plt.show()
```

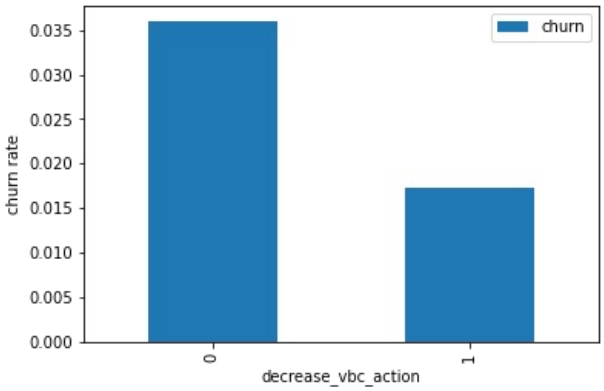


In [74]:
```
data_frame.pivot_table(values='churn', index='decrease_rech_amt_action', aggfunc='mean').plot.bar()
plt.ylabel('churn rate')
plt.show()
```



In [75]:
```
data_frame.pivot_table(values='churn', index='decrease_vbc_action', aggfunc='mean').plot.bar()
```

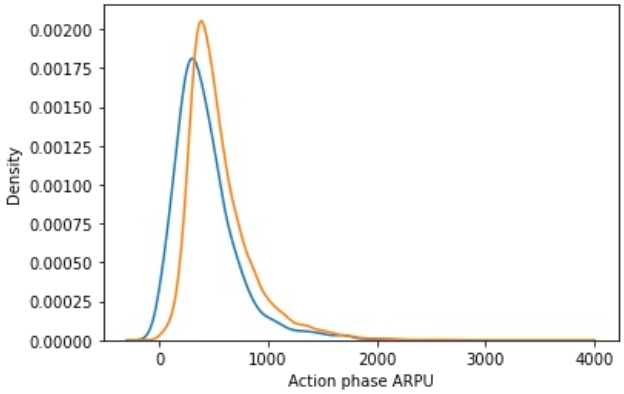
```
plt.ylabel('churn rate')
plt.show()
```



In [76]:
```
# Creating churn dataframe
data_frame_churn = data_frame[data_frame['churn'] == 1]
# Creating not churn dataframe
data_frame_non_churn = data_frame[data_frame['churn'] == 0]
```

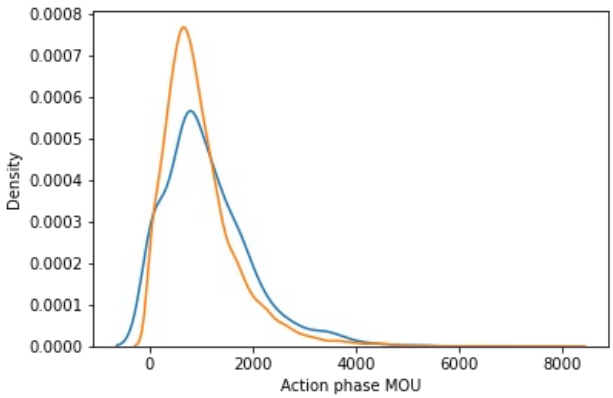
In [77]:
```
# Distribution plot
ax = sns.distplot(data_frame_churn['avg_arpu_action'],label='churn',hist=False)
ax = sns.distplot(data_frame_non_churn['avg_arpu_action'],label='not churn',hist=False)
ax.set(xlabel='Action phase ARPU')
```

Out[77]:
```
[Text(0.5, 0, 'Action phase ARPU')]
```

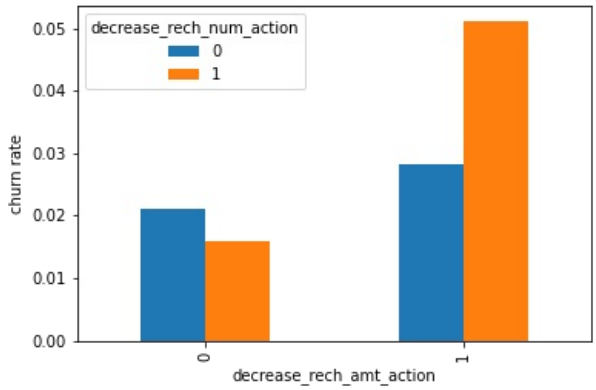


In [78]:
```
# Distribution plot
ax = sns.distplot(data_frame_churn['total_mou_good'],label='churn',hist=False)
ax = sns.distplot(data_frame_non_churn['total_mou_good'],label='non churn',hist=False)
ax.set(xlabel='Action phase MOU')
```

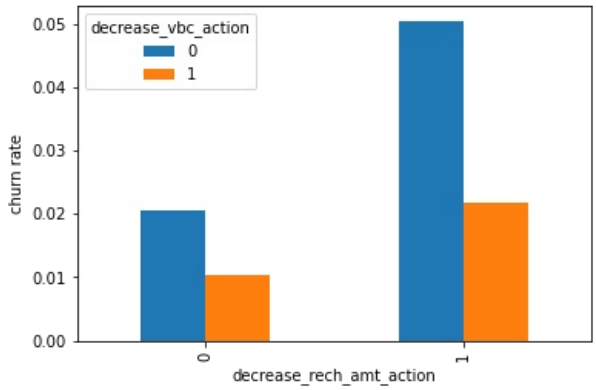
Out[78]:
```
[Text(0.5, 0, 'Action phase MOU')]
```



In [79]:
```
data_frame.pivot_table(values='churn', index='decrease_rech_amt_action', columns='decrease_rech_num_action', ag
plt.ylabel('churn rate')
plt.show()
```



In [80]:
```
data_frame.pivot_table(values='churn', index='decrease_rech_amt_action', columns='decrease_vbc_action', aggfunc
plt.ylabel('churn rate')
plt.show()
```



In [81]:
```
data_frame = data_frame.drop(['total_mou_good','avg_mou_action','diff_mou','avg_rech_num_action','diff_rech_num
                'diff_rech_amt','avg_arpu_action','diff_arpu','avg_vbc_3g_action','diff_vbc','avg_rech_amt_6_7
```

In [82]:
```
# Import library
from sklearn.model_selection import train_test_split
```

In [83]:
```
# Putting feature variables into X
X = data_frame.drop(['mobile_number','churn'], axis=1)
```

In [84]:
```
# Putting target variable to y
y = data_frame['churn']
```

In [85]:
```
# Splitting data into train and test set 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=100)
```

In [86]:
```
# Imporing SMOTE
!pip install imblearn
from imblearn.over_sampling import SMOTE
```

```
Requirement already satisfied: imblearn in c:\users\keshav\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\keshav\anaconda3\lib\site-packages (from imblearn)
(0.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\keshav\anaconda3\lib\site-packages (from imbalanced-le
arn->imblearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\keshav\anaconda3\lib\site-packages (from imbala
nced-learn->imblearn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\keshav\anaconda3\lib\site-packages (from imbalanced-le
arn->imblearn) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in c:\users\keshav\anaconda3\lib\site-packages (from imbalanced-lea
rn->imblearn) (1.7.3)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\keshav\anaconda3\lib\site-packages (from imbalan
ced-learn->imblearn) (1.0.2)
```

In [87]:
```
# Instantiate SMOTE
sm = SMOTE(random_state=27)
```

In [88]:
```
# Fittign SMOTE to the train set
X_train, y_train = sm.fit_resample(X_train, y_train)
```

In [89]:
```
# Standardization method
from sklearn.preprocessing import StandardScaler
```

In [90]:
```
# Instantiate the Scaler
scaler = StandardScaler()
```

In [91]:
```
# List of the numeric columns
cols_scale = X_train.columns.to_list()
```

```
# Removing the derived binary columns
cols_scale.remove('decrease_mou_action')
cols_scale.remove('decrease_rech_num_action')
cols_scale.remove('decrease_rech_amt_action')
cols_scale.remove('decrease_arpu_action')
cols_scale.remove('decrease_vbc_action')
```

In [92]:
```
# Fit the data into scaler and transform
X_train[cols_scale] = scaler.fit_transform(X_train[cols_scale])
```

In [93]:
```
X_train.head()
```

Out[93]:

	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_mou
0	0.0	0.0	0.0	0.140777	-0.522792	-0.276289	0.106540	-0.662084	-0.465777	-0.2112
1	0.0	0.0	0.0	-1.427243	4.428047	3.254270	-0.658491	-0.236590	-0.004450	-0.7760
2	0.0	0.0	0.0	-0.222751	0.543206	0.809117	-0.601239	-0.599206	-0.331043	-0.3633
3	0.0	0.0	0.0	-0.911173	0.842273	0.731302	-0.702232	-0.650471	-0.458464	-0.7897
4	0.0	0.0	0.0	0.271356	0.247684	1.256421	-0.356392	-0.180394	0.114727	0.8992

In [94]:
```
# Transform the test set
X_test[cols_scale] = scaler.transform(X_test[cols_scale])
X_test.head()
```

Out[94]:

	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	onnet_mou_6	onnet_mou_7	onnet_mou_8	offnet_r
5704	0.0	0.0	0.0	0.244310	-0.268832	1.005890	-0.725286	-0.690223	-0.476634	0.4
64892	0.0	0.0	0.0	0.048359	-0.779609	-0.157969	-0.734066	-0.698072	-0.502219	-0.3
39613	0.0	0.0	0.0	0.545470	0.184388	1.403349	-0.537110	-0.521615	-0.206890	0.6
93118	0.0	0.0	0.0	0.641508	0.816632	-0.211023	-0.058843	0.029897	-0.155872	-0.1
81235	0.0	0.0	0.0	3.878627	0.911619	2.745295	4.117829	1.452446	2.809582	-0.0

In [95]:
```
#Import PCA
from sklearn.decomposition import PCA
```

In [96]:
```
# Instantiate PCA
pca = PCA(random_state=42)
```

In [97]:
```
# Fit train set on PCA
pca.fit(X_train)
```

Out[97]:
```
PCA(random_state=42)
```

In [98]:
```
# Principal components
pca.components_
```

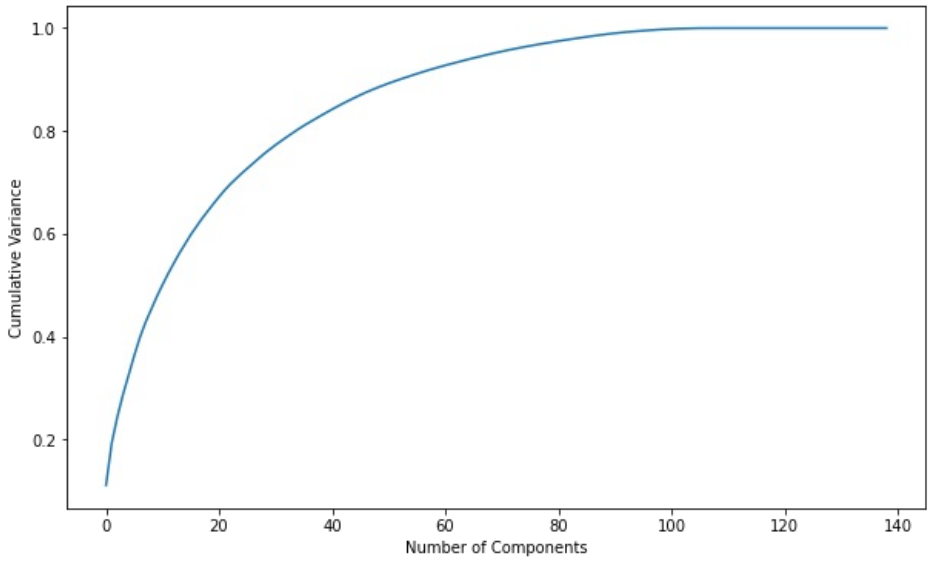
Out[98]:
```
array([[-7.47443877e-20, -5.55111512e-17,  3.33066907e-16, ...,
        -2.59893257e-02, -2.57807140e-02,  1.40722138e-02],
       [-1.61727231e-19, -5.55111512e-17,  5.55111512e-16, ...,
        -1.16748539e-02, -9.94435513e-03, -1.42905153e-02],
       [ 1.92623359e-19, -2.77555756e-17,  3.05311332e-16, ...,
        -4.14768206e-02, -4.24208604e-02,  2.47523684e-02],
       ...,
       [-0.00000000e+00,  6.97658776e-02, -7.60233928e-02, ...,
        -2.14238349e-16,  2.60208521e-17,  5.46437895e-17],
       [ 0.00000000e+00, -4.57218055e-02, -1.25278596e-01, ...,
         6.41847686e-17,  1.38777878e-17, -1.02348685e-16],
       [ 9.99999893e-01,  3.60018160e-04,  2.09437765e-04, ...,
         1.88464831e-19,  4.97208340e-19, -1.92242598e-17]])
```

In [99]:
```
# Cumuliative varinace of the PCs
variance_cumu = np.cumsum(pca.explained_variance_ratio_)
print(variance_cumu)
```

```
[0.11133461 0.19283097 0.2440926  0.28752955 0.32615518 0.36367182
 0.39890175 0.42837946 0.45375469 0.47847974 0.50116085 0.52292175
 0.54326865 0.56289425 0.5810293  0.59863107 0.6144831  0.63007181
 0.64450464 0.65852914 0.67213545 0.68515654 0.69687462 0.70752035
 0.71775148 0.72749314 0.73710332 0.74667617 0.75607246 0.76459763
 0.77304509 0.78098    0.78865022 0.79621975 0.80354734 0.81053736
 0.81707319 0.82350577 0.82986303 0.83618425 0.84224822 0.84817968
 0.85405437 0.85960826 0.86490524 0.87013009 0.87525965 0.87994438
 0.88441128 0.88868207 0.8927598  0.89674958 0.90049645 0.90419989
 0.90781635 0.91134659 0.91483337 0.91821245 0.92142454 0.92456967
 0.92763871 0.93057638 0.93350131 0.93631784 0.93911753 0.94186119
 0.94456629 0.94721564 0.94978601 0.95227185 0.95470619 0.95700032
 0.95926845 0.96147738 0.96352752 0.96554115 0.96746831 0.96937606
 0.97123625 0.9730694  0.97487072 0.97658393 0.97824346 0.97989006
 0.98148902 0.98304748 0.98459115 0.98612531 0.98763521 0.98886954
 0.9900688  0.99121915 0.99230894 0.99326729 0.99412586 0.99496343
 0.99577255 0.99655479 0.99719781 0.99778212 0.99833344 0.99881761
 0.99913004 0.99943115 0.9996942  0.99985431 0.99994779 0.99998118
 0.9999984  0.99999963 0.99999989 1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.        ]
```

In [100…
```
# Plotting scree plot
fig = plt.figure(figsize = (10,6))
plt.plot(variance_cumu)
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Variance')
```

Out[100]:
```
Text(0, 0.5, 'Cumulative Variance')
```



In [101…
```
# Importing incremental PCA
from sklearn.decomposition import IncrementalPCA
```

In [102…
```
# Instantiate PCA with 60 components
pca_final = IncrementalPCA(n_components=60)
```

In [103…
```
# Fit and transform the X_train
X_train_pca = pca_final.fit_transform(X_train)
```

In [104…
```
X_test_pca = pca_final.transform(X_test)
```

In [105…
```
# Importing scikit logistic regression module
from sklearn.linear_model import LogisticRegression
```

In [106…
```
# Impoting metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

In [107…
```
# Importing libraries for cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

In [108…
```
# Creating KFold object with 5 splits
folds = KFold(n_splits=5, shuffle=True, random_state=4)

# Specify params
params = {"C": [0.01, 0.1, 1, 10, 100, 1000]}
```

```
# Specifing score as recall as we are more focused on acheiving the higher sensitivity than the accuracy
model_cv = GridSearchCV(estimator = LogisticRegression(),
                        param_grid = params,
                        scoring= 'recall',
                        cv = folds,
                        verbose = 1,
                        return_train_score=True)

# Fit the model
model_cv.fit(X_train_pca, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

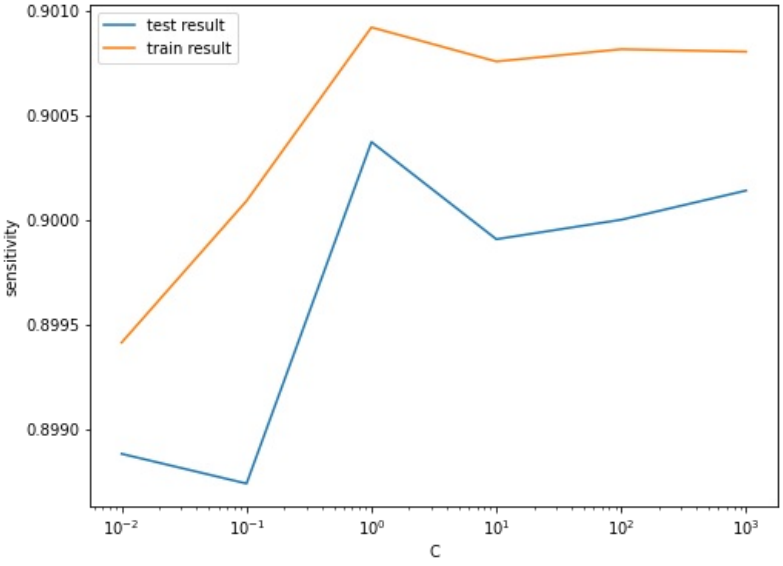
Out[108]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=4, shuffle=True),
             estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1, 10, 100, 1000]},
             return_train_score=True, scoring='recall', verbose=1)
```

In [109…
```
# results of grid search CV
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results
```

Out[109]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	params	split0_test_score	split1_test_score	split2_test_score	sp
0	0.234571	0.013652	0.005606	0.000802	0.01	{'C': 0.01}	0.902202	0.899860	0.897442	
1	0.357583	0.012023	0.004633	0.001192	0.1	{'C': 0.1}	0.901492	0.899627	0.898140	
2	0.435472	0.007125	0.004903	0.000921	1	{'C': 1}	0.903386	0.900327	0.898837	
3	0.434911	0.002301	0.005599	0.000489	10	{'C': 10}	0.903149	0.900093	0.899302	
4	0.438139	0.012606	0.005626	0.000493	100	{'C': 100}	0.903149	0.900327	0.899070	
5	0.424412	0.004572	0.005349	0.000840	1000	{'C': 1000}	0.902913	0.901027	0.899302	

In [110…
```
# plot of C versus train and validation scores

plt.figure(figsize=(8, 6))
plt.plot(cv_results['param_C'], cv_results['mean_test_score'])
plt.plot(cv_results['param_C'], cv_results['mean_train_score'])
plt.xlabel('C')
plt.ylabel('sensitivity')
plt.legend(['test result', 'train result'], loc='upper left')
plt.xscale('log')
```



In [111…
```
# Best score with best C
best_score = model_cv.best_score_
best_C = model_cv.best_params_['C']

print(" The highest test sensitivity is {0} at C = {1}".format(best_score, best_C))
```

```
 The highest test sensitivity is 0.9003727758066977 at C = 1
```

In [112…
```
# Instantiate the model with best C
logistic_pca = LogisticRegression(C=best_C)
```

In [113…
```
# Fit the model on the train set
log_pca_model = logistic_pca.fit(X_train_pca, y_train)
```

In [114…
```
# Predictions on the train set
y_train_pred = log_pca_model.predict(X_train_pca)
```

In [115…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_train, y_train_pred)
print(confusion)
```

```
[[17873  3552]
 [ 2110 19315]]
```

In [116…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [117…
```
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.8678646441073512
Sensitivity:- 0.9015169194865811
Specificity:- 0.8342123687281213
```

In [118…
```
# Prediction on the test set
y_test_pred = log_pca_model.predict(X_test_pca)
```

In [119…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_test, y_test_pred)
print(confusion)
```

```
[[4439  909]
 [  37  156]]
```

In [120…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [121…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.8292726944594838
Sensitivity:- 0.8082901554404145
Specificity:- 0.8300299177262528
```

In [122…
```
# Importing decision tree classifier
from sklearn.tree import DecisionTreeClassifier
```

In [123…
```
# Create the parameter grid
param_grid = {
    'max_depth': range(5, 15, 5),
    'min_samples_leaf': range(50, 150, 50),
    'min_samples_split': range(50, 150, 50),
}


# Instantiate the grid search model
dtree = DecisionTreeClassifier()

grid_search = GridSearchCV(estimator = dtree,
                           param_grid = param_grid,
                           scoring= 'recall',
                           cv = 5,
                           verbose = 1)

# Fit the grid search to the data
grid_search.fit(X_train_pca,y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
```

Out[123]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': range(5, 15, 5),
                         'min_samples_leaf': range(50, 150, 50),
                         'min_samples_split': range(50, 150, 50)},
             scoring='recall', verbose=1)
```

In [124…
```
# cv results
cv_results = pd.DataFrame(grid_search.cv_results_)
cv_results
```

Out[124]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_min_samples_leaf	param_min_samples_split	
0	1.682542	0.039775	0.005602	0.001199	5	50	50	'mir 'min
1	1.567686	0.030802	0.006070	0.001327	5	50	100	'mir 'min
2	1.195377	0.114730	0.004600	0.001200	5	100	50	'mir 'min
3	1.078858	0.040336	0.003800	0.000400	5	100	100	'mir 'min
4	1.831122	0.059986	0.004254	0.000387	10	50	50	{' 'mir 'min
5	1.807553	0.088931	0.003801	0.000749	10	50	100	{' 'mir 'min
6	1.519720	0.019945	0.003473	0.000450	10	100	50	{' 'mir 'min
7	1.480777	0.018721	0.003899	0.000487	10	100	100	{' 'mir 'min

In [125…
```
# Printing the optimal sensitivity score and hyperparameters
print("Best sensitivity:-", grid_search.best_score_)
print(grid_search.best_estimator_)
```

```
Best sensitivity:- 0.9072112018669779
DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=50)
```

In [126…
```
# Model with optimal hyperparameters
dt_pca_model = DecisionTreeClassifier(criterion = "gini",
                                      random_state = 100,
                                      max_depth=10,
                                      min_samples_leaf=50,
                                      min_samples_split=50)

dt_pca_model.fit(X_train_pca, y_train)
```

Out[126]:
```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=50,
                       random_state=100)
```

In [127…
```
# Predictions on the train set
y_train_pred = dt_pca_model.predict(X_train_pca)
```

In [128…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_train, y_train_pred)
print(confusion)
```

```
[[18561  2864]
 [ 1409 20016]]
```

In [129…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [130…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.9002800466744457
Sensitivity:- 0.9342357059509918
Specificity:- 0.8663243873978996
```

In [131…]
```
# Prediction on the test set
y_test_pred = dt_pca_model.predict(X_test_pca)
```

In [132…]
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_test, y_test_pred)
print(confusion)
```

```
[[4506  842]
 [  62  131]]
```

In [133…]
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [134…]
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.8368525536906696
Sensitivity:- 0.6787564766839378
Specificity:- 0.8425579655946148
```

In [135…]
```
# Importing random forest classifier
from sklearn.ensemble import RandomForestClassifier
```

In [136…]
```
param_grid = {
    'max_depth': range(5,10,5),
    'min_samples_leaf': range(50, 150, 50),
    'min_samples_split': range(50, 150, 50),
    'n_estimators': [100,200,300],
    'max_features': [10, 20]
}
# Create a based model
rf = RandomForestClassifier()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf,
                           param_grid = param_grid,
                           cv = 3,
                           n_jobs = -1,
                           verbose = 1,
                           return_train_score=True)

# Fit the model
grid_search.fit(X_train_pca, y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
```

Out[136]:
```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': range(5, 10, 5), 'max_features': [10, 20],
                         'min_samples_leaf': range(50, 150, 50),
                         'min_samples_split': range(50, 150, 50),
                         'n_estimators': [100, 200, 300]},
             return_train_score=True, verbose=1)
```

In [137…]
```
# printing the optimal accuracy score and hyperparameters
print('We can get accuracy of',grid_search.best_score_,'using',grid_search.best_params_)
```

```
We can get accuracy of 0.8444807248372057 using {'max_depth': 5, 'max_features': 20, 'min_samples_leaf': 50, 'm
in_samples_split': 100, 'n_estimators': 100}
```

In [138…]
```
# model with the best hyperparameters

rfc_model = RandomForestClassifier(bootstrap=True,
                           max_depth=5,
                           min_samples_leaf=50,
                           min_samples_split=100,
                           max_features=20,
                           n_estimators=300)
```

In [139…]
```
# Fit the model
rfc_model.fit(X_train_pca, y_train)
```

Out[139]:
```
RandomForestClassifier(max_depth=5, max_features=20, min_samples_leaf=50,
                       min_samples_split=100, n_estimators=300)
```

In [140…]
```
# Predictions on the train set
y_train_pred = rfc_model.predict(X_train_pca)
```

In [141…]
```
# Confusion matrix
```

```
confusion = metrics.confusion_matrix(y_train, y_train_pred)
print(confusion)
```

```
[[17331  4094]
 [ 2429 18996]]
```

In [142…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [143…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.8477712952158694
Sensitivity:- 0.8866277712952159
Specificity:- 0.8089148191365227
```

In [144…
```
# Prediction on the test set
y_test_pred = rfc_model.predict(X_test_pca)
```

In [145…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_test, y_test_pred)
print(confusion)
```

```
[[4284 1064]
 [  47  146]]
```

In [146…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [147…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.7994946760512542
Sensitivity:- 0.7564766839378239
Specificity:- 0.8010471204188482
```

In [148…
```
##### Importing stats model
import statsmodels.api as sm
```

In [149…
```
# Instantiate the model
# Adding the constant to X_train
log_no_pca = sm.GLM(y_train,(sm.add_constant(X_train)), family=sm.families.Binomial())
```

In [150…
```
# Fit the model
log_no_pca = log_no_pca.fit().summary()
```

In [151…
```
# Summary
log_no_pca
```

Out[151]:

Generalized Linear Model Regression Results

Dep. Variable:	churn	**No. Observations:**	42850
Model:	GLM	**Df Residuals:**	42719
Model Family:	Binomial	**Df Model:**	130
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	nan
Date:	Tue, 06 Jun 2023	**Deviance:**	23407.
Time:	12:25:40	**Pearson chi2:**	4.03e+05
No. Iterations:	100	**Pseudo R-squ. (CS):**	nan
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-57.0829	4420.158	-0.013	0.990	-8720.434	8606.268
loc_og_t2o_mou	1.591e-07	2.16e-05	0.007	0.994	-4.22e-05	4.26e-05
std_og_t2o_mou	1.678e-06	0.000	0.016	0.987	-0.000	0.000

loc_ic_t2o_mou	4.339e-07	6.32e-05	0.007	0.995	-0.000	0.000
arpu_6	-0.0558	0.082	-0.684	0.494	-0.216	0.104
arpu_7	0.1168	0.087	1.345	0.178	-0.053	0.287
arpu_8	0.0773	0.111	0.699	0.484	-0.139	0.294
onnet_mou_6	15.8127	3.589	4.406	0.000	8.778	22.847
onnet_mou_7	-4.4517	1.809	-2.461	0.014	-7.997	-0.907
onnet_mou_8	2.8340	1.826	1.552	0.121	-0.746	6.414
offnet_mou_6	15.2708	3.376	4.524	0.000	8.655	21.887
offnet_mou_7	-1.8379	1.713	-1.073	0.283	-5.196	1.520
offnet_mou_8	-0.2395	1.883	-0.127	0.899	-3.931	3.452
roam_ic_mou_6	0.1620	0.037	4.396	0.000	0.090	0.234
roam_ic_mou_7	-0.0189	0.053	-0.358	0.721	-0.123	0.085
roam_ic_mou_8	0.2281	0.046	4.947	0.000	0.138	0.318
roam_og_mou_6	-5.2242	1.136	-4.598	0.000	-7.451	-2.997
roam_og_mou_7	0.8860	0.472	1.877	0.061	-0.039	1.811
roam_og_mou_8	-0.0130	0.531	-0.024	0.980	-1.054	1.028
loc_og_t2t_mou_6	-3147.2299	659.693	-4.771	0.000	-4440.205	-1854.255
loc_og_t2t_mou_7	-1562.3857	682.983	-2.288	0.022	-2901.009	-223.763
loc_og_t2t_mou_8	5555.8956	630.274	8.815	0.000	4320.582	6791.210
loc_og_t2m_mou_6	-3184.9758	667.487	-4.772	0.000	-4493.226	-1876.726
loc_og_t2m_mou_7	-1474.9139	644.122	-2.290	0.022	-2737.371	-212.457
loc_og_t2m_mou_8	5929.9029	672.526	8.817	0.000	4611.776	7248.030
loc_og_t2f_mou_6	-271.7827	56.996	-4.768	0.000	-383.493	-160.072
loc_og_t2f_mou_7	-130.3352	56.943	-2.289	0.022	-241.942	-18.728
loc_og_t2f_mou_8	490.9303	55.705	8.813	0.000	381.750	600.111
loc_og_t2c_mou_6	0.0405	0.022	1.851	0.064	-0.002	0.083
loc_og_t2c_mou_7	0.0122	0.022	0.565	0.572	-0.030	0.054
loc_og_t2c_mou_8	0.0625	0.022	2.786	0.005	0.019	0.106
loc_og_mou_6	3396.5816	1275.486	2.663	0.008	896.675	5896.488
loc_og_mou_7	6133.1427	1336.616	4.589	0.000	3513.424	8752.861
loc_og_mou_8	-323.2059	1357.040	-0.238	0.812	-2982.955	2336.543
std_og_t2t_mou_6	-1.265e+04	1876.176	-6.741	0.000	-1.63e+04	-8970.639
std_og_t2t_mou_7	-9354.2832	1829.610	-5.113	0.000	-1.29e+04	-5768.314
std_og_t2t_mou_8	6286.0278	1516.109	4.146	0.000	3314.508	9257.547
std_og_t2m_mou_6	-1.173e+04	1740.506	-6.742	0.000	-1.51e+04	-8322.660
std_og_t2m_mou_7	-9126.9015	1784.776	-5.114	0.000	-1.26e+04	-5628.805
std_og_t2m_mou_8	6405.4545	1544.260	4.148	0.000	3378.761	9432.148
std_og_t2f_mou_6	-246.8160	36.570	-6.749	0.000	-318.492	-175.140
std_og_t2f_mou_7	-206.5358	40.427	-5.109	0.000	-285.771	-127.301
std_og_t2f_mou_8	152.9663	36.905	4.145	0.000	80.634	225.298
std_og_t2c_mou_6	2.399e-06	0.000	0.014	0.989	-0.000	0.000
std_og_t2c_mou_7	-3.173e-06	0.000	-0.012	0.990	-0.001	0.001
std_og_t2c_mou_8	1.591e-06	0.000	0.011	0.991	-0.000	0.000
std_og_mou_6	1.362e+04	2978.210	4.572	0.000	7778.206	1.95e+04
std_og_mou_7	2.129e+04	3114.740	6.835	0.000	1.52e+04	2.74e+04
std_og_mou_8	7184.1720	2777.899	2.586	0.010	1739.590	1.26e+04
isd_og_mou_6	-55.3623	29.786	-1.859	0.063	-113.742	3.017
isd_og_mou_7	102.8116	27.901	3.685	0.000	48.126	157.497
isd_og_mou_8	320.8748	34.278	9.361	0.000	253.691	388.058
spl_og_mou_6	-89.2634	47.942	-1.862	0.063	-183.229	4.702
spl_og_mou_7	250.7939	67.996	3.688	0.000	117.524	384.063
spl_og_mou_8	523.6974	55.823	9.381	0.000	414.285	633.109
og_others_6	-10.8362	5.838	-1.856	0.063	-22.278	0.606
og_others_7	17.1839	4.923	3.490	0.000	7.534	26.834
og_others_8	-4098.1424	3.22e+05	-0.013	0.990	-6.35e+05	6.27e+05

total_og_mou_6	3658.4549	1978.237	1.849	0.064	-218.819	7535.729
total_og_mou_7	-8542.8713	2316.725	-3.687	0.000	-1.31e+04	-4002.175
total_og_mou_8	-1.896e+04	2020.310	-9.383	0.000	-2.29e+04	-1.5e+04
loc_ic_t2t_mou_6	-433.0090	403.509	-1.073	0.283	-1223.872	357.854
loc_ic_t2t_mou_7	1954.8242	442.997	4.413	0.000	1086.566	2823.082
loc_ic_t2t_mou_8	6488.0231	419.562	15.464	0.000	5665.696	7310.350
loc_ic_t2m_mou_6	-607.6670	566.360	-1.073	0.283	-1717.712	502.378
loc_ic_t2m_mou_7	2632.2800	596.425	4.413	0.000	1463.308	3801.252
loc_ic_t2m_mou_8	9349.4089	604.611	15.464	0.000	8164.393	1.05e+04
loc_ic_t2f_mou_6	-120.0635	111.808	-1.074	0.283	-339.202	99.075
loc_ic_t2f_mou_7	570.1459	129.238	4.412	0.000	316.845	823.447
loc_ic_t2f_mou_8	1776.4687	114.900	15.461	0.000	1551.269	2001.669
loc_ic_mou_6	-1357.1927	1060.719	-1.280	0.201	-3436.164	721.779
loc_ic_mou_7	-2572.7900	1120.911	-2.295	0.022	-4769.735	-375.845
loc_ic_mou_8	3340.2537	1141.024	2.927	0.003	1103.887	5576.620
std_ic_t2t_mou_6	-2053.2195	318.022	-6.456	0.000	-2676.531	-1429.908
std_ic_t2t_mou_7	-330.7973	318.776	-1.038	0.299	-955.587	293.992
std_ic_t2t_mou_8	-605.4093	228.037	-2.655	0.008	-1052.354	-158.465
std_ic_t2m_mou_6	-2123.4928	328.903	-6.456	0.000	-2768.132	-1478.854
std_ic_t2m_mou_7	-339.4794	326.926	-1.038	0.299	-980.243	301.284
std_ic_t2m_mou_8	-927.0661	349.453	-2.653	0.008	-1611.981	-242.151
std_ic_t2f_mou_6	-365.6859	56.656	-6.455	0.000	-476.729	-254.643
std_ic_t2f_mou_7	-63.4813	61.339	-1.035	0.301	-183.703	56.740
std_ic_t2f_mou_8	-152.3225	57.144	-2.666	0.008	-264.324	-40.321
std_ic_t2o_mou_6	7.293e-08	5.09e-06	0.014	0.989	-9.91e-06	1.01e-05
std_ic_t2o_mou_7	2.362e-06	0.000	0.014	0.989	-0.000	0.000
std_ic_t2o_mou_8	-3.428e-07	5.02e-05	-0.007	0.995	-9.87e-05	9.8e-05
std_ic_mou_6	2088.4473	604.652	3.454	0.001	903.351	3273.544
std_ic_mou_7	1144.7438	614.337	1.863	0.062	-59.334	2348.822
std_ic_mou_8	8467.8254	571.709	14.811	0.000	7347.296	9588.355
total_ic_mou_6	2634.0832	946.207	2.784	0.005	779.551	4488.616
total_ic_mou_7	-1488.6791	1012.209	-1.471	0.141	-3472.573	495.215
total_ic_mou_8	-1.987e+04	1039.295	-19.117	0.000	-2.19e+04	-1.78e+04
spl_ic_mou_6	-1.4036	0.564	-2.487	0.013	-2.510	-0.297
spl_ic_mou_7	0.5556	0.520	1.069	0.285	-0.463	1.574
spl_ic_mou_8	5.2198	0.296	17.638	0.000	4.640	5.800
isd_ic_mou_6	-445.2453	159.896	-2.785	0.005	-758.636	-131.855
isd_ic_mou_7	265.3716	180.378	1.471	0.141	-88.164	618.907
isd_ic_mou_8	3515.6679	183.907	19.117	0.000	3155.217	3876.119
ic_others_6	-74.5944	26.745	-2.789	0.005	-127.014	-22.175
ic_others_7	41.0742	27.944	1.470	0.142	-13.696	95.844
ic_others_8	551.3440	28.890	19.084	0.000	494.720	607.968
total_rech_num_6	0.0210	0.035	0.597	0.550	-0.048	0.090
total_rech_num_7	0.0736	0.040	1.825	0.068	-0.005	0.153
total_rech_num_8	-0.6481	0.041	-15.800	0.000	-0.728	-0.568
total_rech_amt_6	0.6504	0.083	7.865	0.000	0.488	0.813
total_rech_amt_7	-0.2252	0.081	-2.782	0.005	-0.384	-0.067
total_rech_amt_8	0.2918	0.115	2.539	0.011	0.067	0.517
max_rech_amt_6	-0.2253	0.037	-6.094	0.000	-0.298	-0.153
max_rech_amt_7	-0.0479	0.036	-1.327	0.185	-0.119	0.023
max_rech_amt_8	0.1497	0.043	3.449	0.001	0.065	0.235
last_day_rch_amt_6	-0.1838	0.029	-6.303	0.000	-0.241	-0.127
last_day_rch_amt_7	0.0128	0.029	0.439	0.661	-0.044	0.070
last_day_rch_amt_8	-0.5247	0.033	-15.910	0.000	-0.589	-0.460

vol_2g_mb_6	0.1408	0.030	4.646	0.000	0.081	0.200
vol_2g_mb_7	0.0267	0.033	0.821	0.412	-0.037	0.091
vol_2g_mb_8	0.0729	0.035	2.087	0.037	0.004	0.141
vol_3g_mb_6	0.3796	0.050	7.571	0.000	0.281	0.478
vol_3g_mb_7	0.3873	0.057	6.812	0.000	0.276	0.499
vol_3g_mb_8	-0.1923	0.069	-2.789	0.005	-0.328	-0.057
monthly_2g_6	-0.6080	0.046	-13.355	0.000	-0.697	-0.519
monthly_2g_7	-0.4153	0.042	-9.856	0.000	-0.498	-0.333
monthly_2g_8	-0.6412	0.059	-10.794	0.000	-0.758	-0.525
sachet_2g_6	-0.0265	0.031	-0.847	0.397	-0.088	0.035
sachet_2g_7	-0.2139	0.033	-6.397	0.000	-0.279	-0.148
sachet_2g_8	-0.2184	0.032	-6.830	0.000	-0.281	-0.156
monthly_3g_6	-0.3294	0.049	-6.736	0.000	-0.425	-0.234
monthly_3g_7	-0.6040	0.054	-11.239	0.000	-0.709	-0.499
monthly_3g_8	-0.8796	0.080	-10.999	0.000	-1.036	-0.723
sachet_3g_6	-0.0372	0.033	-1.110	0.267	-0.103	0.028
sachet_3g_7	-0.0945	0.043	-2.187	0.029	-0.179	-0.010
sachet_3g_8	-0.1217	0.051	-2.402	0.016	-0.221	-0.022
aon	-0.1686	0.022	-7.775	0.000	-0.211	-0.126
aug_vbc_3g	-0.1401	0.057	-2.474	0.013	-0.251	-0.029
jul_vbc_3g	-0.0177	0.047	-0.377	0.706	-0.110	0.074
jun_vbc_3g	0.2170	0.050	4.328	0.000	0.119	0.315
sep_vbc_3g	-1.0879	0.110	-9.911	0.000	-1.303	-0.873
decrease_mou_action	-0.4893	0.053	-9.238	0.000	-0.593	-0.385
decrease_rech_num_action	-1.0335	0.048	-21.558	0.000	-1.127	-0.940
decrease_rech_amt_action	-0.3131	0.065	-4.800	0.000	-0.441	-0.185
decrease_arpu_action	-0.1824	0.067	-2.728	0.006	-0.313	-0.051
decrease_vbc_action	-1.6643	0.132	-12.578	0.000	-1.924	-1.405

In [152…
```
# Importing logistic regression from sklearn
from sklearn.linear_model import LogisticRegression
# Intantiate the logistic regression
logreg = LogisticRegression()
```

In [156…
```
# Importing RFE
from sklearn.feature_selection import RFE

# Intantiate RFE with 15 columns
rfe = RFE(estimator=LogisticRegression(), n_features_to_select=15)

# Fit the rfe model with train set
rfe = rfe.fit(X_train, y_train)
```

In [157…
```
# RFE selected columns
rfe_cols = X_train.columns[rfe.support_]
print(rfe_cols)
```

```
Index(['offnet_mou_7', 'offnet_mou_8', 'roam_og_mou_8', 'std_og_t2m_mou_8',
       'isd_og_mou_8', 'og_others_7', 'og_others_8', 'loc_ic_t2f_mou_8',
       'loc_ic_mou_8', 'std_ic_t2f_mou_8', 'ic_others_8', 'monthly_2g_8',
       'monthly_3g_8', 'sep_vbc_3g', 'decrease_vbc_action'],
      dtype='object')
```

In [158…
```
# Adding constant to X_train
X_train_sm_1 = sm.add_constant(X_train[rfe_cols])

#Instantiate the model
log_no_pca_1 = sm.GLM(y_train, X_train_sm_1, family=sm.families.Binomial())

# Fit the model
log_no_pca_1 = log_no_pca_1.fit()

log_no_pca_1.summary()
```

Out[158]:

Generalized Linear Model Regression Results

Dep. Variable:	churn	**No. Observations:**	42850
Model:	GLM	**Df Residuals:**	42834
Model Family:	Binomial	**Df Model:**	15
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	nan
Date:	Tue, 06 Jun 2023	**Deviance:**	30600.
Time:	12:32:38	**Pearson chi2:**	9.71e+06
No. Iterations:	41	**Pseudo R-squ. (CS):**	nan
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-53.2620	4234.336	-0.013	0.990	-8352.408	8245.884
offnet_mou_7	0.6909	0.026	26.690	0.000	0.640	0.742
offnet_mou_8	-3.4031	0.109	-31.286	0.000	-3.616	-3.190
roam_og_mou_8	1.2625	0.032	39.012	0.000	1.199	1.326
std_og_t2m_mou_8	2.3636	0.095	24.817	0.000	2.177	2.550
isd_og_mou_8	-0.9043	0.176	-5.126	0.000	-1.250	-0.559
og_others_7	-1.2372	0.774	-1.598	0.110	-2.755	0.280
og_others_8	-3789.7350	3.08e+05	-0.012	0.990	-6.08e+05	6e+05
loc_ic_t2f_mou_8	-0.6001	0.070	-8.534	0.000	-0.738	-0.462
loc_ic_mou_8	-2.2360	0.067	-33.229	0.000	-2.368	-2.104
std_ic_t2f_mou_8	-0.6939	0.074	-9.403	0.000	-0.838	-0.549
ic_others_8	-1.5404	0.132	-11.657	0.000	-1.799	-1.281
monthly_2g_8	-0.8868	0.044	-20.062	0.000	-0.973	-0.800
monthly_3g_8	-0.9602	0.044	-21.947	0.000	-1.046	-0.874
sep_vbc_3g	-1.0260	0.096	-10.644	0.000	-1.215	-0.837
decrease_vbc_action	-1.1397	0.073	-15.578	0.000	-1.283	-0.996

In [159…

```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [160…

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train[rfe_cols].columns
vif['VIF'] = [variance_inflation_factor(X_train[rfe_cols].values, i) for i in range(X_train[rfe_cols].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[160]:

	Features	VIF
1	offnet_mou_8	7.29
3	std_og_t2m_mou_8	6.26
0	offnet_mou_7	1.89
8	loc_ic_mou_8	1.67
7	loc_ic_t2f_mou_8	1.21
2	roam_og_mou_8	1.16
14	decrease_vbc_action	1.08
12	monthly_3g_8	1.07
6	og_others_8	1.05
11	monthly_2g_8	1.05
5	og_others_7	1.04
9	std_ic_t2f_mou_8	1.02
10	ic_others_8	1.02
13	sep_vbc_3g	1.02
4	isd_og_mou_8	1.01

In [161…

```
# Removing og_others_8 column
log_cols = rfe_cols.to_list()
log_cols.remove('og_others_8')
print(log_cols)
```

```
['offnet_mou_7', 'offnet_mou_8', 'roam_og_mou_8', 'std_og_t2m_mou_8', 'isd_og_mou_8', 'og_others_7', 'loc_ic_t2
f_mou_8', 'loc_ic_mou_8', 'std_ic_t2f_mou_8', 'ic_others_8', 'monthly_2g_8', 'monthly_3g_8', 'sep_vbc_3g', 'dec
rease_vbc_action']
```

In [162…

```
# Adding constant to X_train
X_train_sm_2 = sm.add_constant(X_train[log_cols])

#Instantiate the model
log_no_pca_2 = sm.GLM(y_train, X_train_sm_2, family=sm.families.Binomial())

# Fit the model
log_no_pca_2 = log_no_pca_2.fit()

log_no_pca_2.summary()
```

Out[162]:

Generalized Linear Model Regression Results

Dep. Variable:	churn	**No. Observations:**	42850
Model:	GLM	**Df Residuals:**	42835
Model Family:	Binomial	**Df Model:**	14
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-15336.
Date:	Tue, 06 Jun 2023	**Deviance:**	30673.
Time:	12:32:56	**Pearson chi2:**	9.53e+06
No. Iterations:	11	**Pseudo R-squ. (CS):**	0.4885
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.2302	0.032	-38.112	0.000	-1.293	-1.167
offnet_mou_7	0.6896	0.026	26.682	0.000	0.639	0.740
offnet_mou_8	-3.4040	0.109	-31.340	0.000	-3.617	-3.191
roam_og_mou_8	1.2614	0.032	39.124	0.000	1.198	1.325
std_og_t2m_mou_8	2.3636	0.095	24.852	0.000	2.177	2.550
isd_og_mou_8	-0.9962	0.188	-5.302	0.000	-1.365	-0.628
og_others_7	-1.7695	0.733	-2.414	0.016	-3.206	-0.333
loc_ic_t2f_mou_8	-0.5992	0.070	-8.525	0.000	-0.737	-0.461
loc_ic_mou_8	-2.2351	0.067	-33.248	0.000	-2.367	-2.103
std_ic_t2f_mou_8	-0.7069	0.074	-9.542	0.000	-0.852	-0.562
ic_others_8	-1.5352	0.132	-11.625	0.000	-1.794	-1.276
monthly_2g_8	-0.8922	0.045	-20.044	0.000	-0.979	-0.805
monthly_3g_8	-0.9583	0.044	-21.924	0.000	-1.044	-0.873
sep_vbc_3g	-1.0234	0.096	-10.625	0.000	-1.212	-0.835
decrease_vbc_action	-1.1416	0.073	-15.619	0.000	-1.285	-0.998

In [163…

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train[log_cols].columns
vif['VIF'] = [variance_inflation_factor(X_train[log_cols].values, i) for i in range(X_train[log_cols].shape[1])
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[163]:

	Features	VIF
1	offnet_mou_8	7.29
3	std_og_t2m_mou_8	6.26
0	offnet_mou_7	1.89
7	loc_ic_mou_8	1.67
6	loc_ic_t2f_mou_8	1.21
2	roam_og_mou_8	1.16
13	decrease_vbc_action	1.08
11	monthly_3g_8	1.07
10	monthly_2g_8	1.05
8	std_ic_t2f_mou_8	1.02
12	sep_vbc_3g	1.02
4	isd_og_mou_8	1.01
9	ic_others_8	1.01
5	og_others_7	1.00

In [164…

```
# Removing offnet_mou_8 column
log_cols.remove('offnet_mou_8')
```

In [165…

```
# Adding constant to X_train
X_train_sm_3 = sm.add_constant(X_train[log_cols])

#Instantiate the model
log_no_pca_3 = sm.GLM(y_train, X_train_sm_3, family=sm.families.Binomial())

# Fit the model
log_no_pca_3 = log_no_pca_3.fit()

log_no_pca_3.summary()
```

Out[165]:

Generalized Linear Model Regression Results

Dep. Variable:	churn	**No. Observations:**	42850
Model:	GLM	**Df Residuals:**	42836
Model Family:	Binomial	**Df Model:**	13
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-16077.
Date:	Tue, 06 Jun 2023	**Deviance:**	32154.
Time:	12:33:00	**Pearson chi2:**	5.11e+06
No. Iterations:	11	**Pseudo R-squ. (CS):**	0.4705
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.3404	0.033	-40.993	0.000	-1.404	-1.276
offnet_mou_7	0.4371	0.022	19.769	0.000	0.394	0.480
roam_og_mou_8	0.6955	0.025	27.977	0.000	0.647	0.744
std_og_t2m_mou_8	-0.4470	0.021	-20.827	0.000	-0.489	-0.405
isd_og_mou_8	-1.2494	0.201	-6.209	0.000	-1.644	-0.855
og_others_7	-2.2296	0.767	-2.908	0.004	-3.733	-0.727
loc_ic_t2f_mou_8	-0.5239	0.072	-7.231	0.000	-0.666	-0.382
loc_ic_mou_8	-3.7047	0.058	-64.141	0.000	-3.818	-3.591
std_ic_t2f_mou_8	-0.8533	0.077	-11.044	0.000	-1.005	-0.702
ic_others_8	-1.5551	0.129	-12.061	0.000	-1.808	-1.302
monthly_2g_8	-0.9187	0.045	-20.377	0.000	-1.007	-0.830
monthly_3g_8	-1.0632	0.048	-22.353	0.000	-1.156	-0.970
sep_vbc_3g	-1.0149	0.090	-11.301	0.000	-1.191	-0.839
decrease_vbc_action	-1.1764	0.072	-16.291	0.000	-1.318	-1.035

In [166…

```
vif = pd.DataFrame()
vif['Features'] = X_train[log_cols].columns
vif['VIF'] = [variance_inflation_factor(X_train[log_cols].values, i) for i in range(X_train[log_cols].shape[1])
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[166]:

	Features	VIF
0	offnet_mou_7	1.71
2	std_og_t2m_mou_8	1.70
6	loc_ic_mou_8	1.28
5	loc_ic_t2f_mou_8	1.21
12	decrease_vbc_action	1.08
10	monthly_3g_8	1.07
1	roam_og_mou_8	1.05
9	monthly_2g_8	1.05
7	std_ic_t2f_mou_8	1.02
11	sep_vbc_3g	1.02
8	ic_others_8	1.01
3	isd_og_mou_8	1.00
4	og_others_7	1.00

In [167…
```
# Getting the predicted value on the train set
y_train_pred_no_pca = log_no_pca_3.predict(X_train_sm_3)
y_train_pred_no_pca.head()
```

Out[167]:
```
0    3.138083e-01
1    1.525542e-01
2    4.431012e-02
3    5.293679e-03
4    1.739629e-19
dtype: float64
```

In [168…
```
y_train_pred_final = pd.DataFrame({'churn':y_train.values, 'churn_prob':y_train_pred_no_pca.values})

#Assigning Customer ID for each record for better readblity
#CustID is the index of each record.
y_train_pred_final['CustID'] = y_train_pred_final.index

y_train_pred_final.head()
```

Out[168]:

	churn	churn_prob	CustID
0	0	3.138083e-01	0
1	0	1.525542e-01	1
2	0	4.431012e-02	2
3	0	5.293679e-03	3
4	0	1.739629e-19	4

In [169…
```
# Creating columns for different probablity cutoffs
prob_cutoff = [float(p/10) for p in range(10)]

for i in prob_cutoff:
    y_train_pred_final[i] = y_train_pred_final['churn_prob'].map(lambda x : 1 if x > i else 0)

y_train_pred_final.head()
```

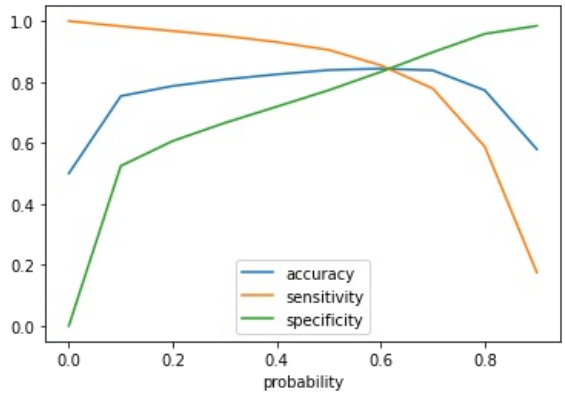
Out[169]:

	churn	churn_prob	CustID	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	3.138083e-01	0	1	1	1	1	0	0	0	0	0	0
1	0	1.525542e-01	1	1	1	0	0	0	0	0	0	0	0
2	0	4.431012e-02	2	1	0	0	0	0	0	0	0	0	0
3	0	5.293679e-03	3	1	0	0	0	0	0	0	0	0	0
4	0	1.739629e-19	4	1	0	0	0	0	0	0	0	0	0

In [170…
```
# Creating a dataframe
cutoff_df = pd.DataFrame(columns=['probability', 'accuracy', 'sensitivity', 'specificity'])

for i in prob_cutoff:
    cm1 = metrics.confusion_matrix(y_train_pred_final['churn'], y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

```
     probability  accuracy  sensitivity  specificity
0.0  0.0          0.500000  1.000000     0.000000
0.1  0.1          0.753862  0.982917     0.524807
0.2  0.2          0.786954  0.967375     0.606534
0.3  0.3          0.808471  0.951039     0.665904
0.4  0.4          0.825134  0.931015     0.719253
0.5  0.5          0.839230  0.905298     0.773162
0.6  0.6          0.844037  0.854796     0.833279
0.7  0.7          0.838180  0.778576     0.897783
0.8  0.8          0.772789  0.587585     0.957993
0.9  0.9          0.579533  0.174842     0.984224
```

In [171…
```
# Plotting accuracy, sensitivity and specificity for different probabilities.
cutoff_df.plot('probability', ['accuracy','sensitivity','specificity'])
plt.show()
```



In [172…
```
# Creating a column with name "predicted", which is the predicted value for 0.5 cutoff
y_train_pred_final['predicted'] = y_train_pred_final['churn_prob'].map(lambda x: 1 if x > 0.5 else 0)
y_train_pred_final.head()
```

Out[172]:

	churn	churn_prob	CustID	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	predicted
0	0	3.138083e-01	0	1	1	1	1	0	0	0	0	0	0	0
1	0	1.525542e-01	1	1	1	0	0	0	0	0	0	0	0	0
2	0	4.431012e-02	2	1	0	0	0	0	0	0	0	0	0	0
3	0	5.293679e-03	3	1	0	0	0	0	0	0	0	0	0	0
4	0	1.739629e-19	4	1	0	0	0	0	0	0	0	0	0	0

In [173…
```
# Confusion metrics
confusion = metrics.confusion_matrix(y_train_pred_final['churn'], y_train_pred_final['predicted'])
print(confusion)
```

```
[[16565  4860]
 [ 2029 19396]]
```

In [174…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

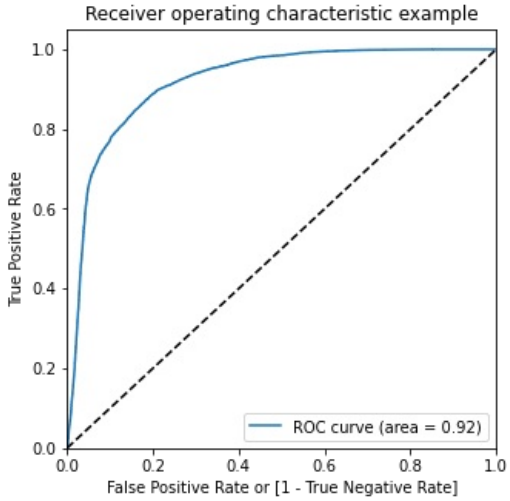
In [175…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train_pred_final['churn'], y_train_pred_final['predicted']))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.8392298716452742
Sensitivity:- 0.9052975495915986
Specificity:- 0.7731621936989498
```

In [176…
```
# ROC Curve function

def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
```

```
    plt.show()

    return None
```

In [177…
```
draw_roc(y_train_pred_final['churn'], y_train_pred_final['churn_prob'])
```



In [178…
```
# Taking a copy of the test set
X_test_log = X_test.copy()
```

In [179…
```
# Taking only the columns, which are selected in the train set after removing insignificant and multicollinear
X_test_log = X_test_log[log_cols]
```

In [180…
```
# Adding constant on the test set
X_test_sm = sm.add_constant(X_test_log)
```

In [181…
```
# Predict on the test set
y_test_pred = log_no_pca_3.predict(X_test_sm)
```

In [182…
```
y_test_pred.head()
```

Out[182]:
```
5704     0.016571
64892    0.000682
39613    0.429515
93118    0.012149
81235    0.039178
dtype: float64
```

In [183…
```
# Converting y_test_pred to a dataframe because y_test_pred is an array
y_pred_1 = pd.DataFrame(y_test_pred)
y_pred_1.head()
```

Out[183]:

	0
5704	0.016571
64892	0.000682
39613	0.429515
93118	0.012149
81235	0.039178

In [184…
```
# Convetting y_test to a dataframe
y_test_df = pd.DataFrame(y_test)
y_test_df.head()
```

Out[184]:

	churn
5704	0
64892	0
39613	0
93118	0
81235	0

In [185…
```
# Putting index to Customer ID
y_test_df['CustID'] = y_test_df.index
```

In [186…
```
# Removing index form the both dataframes for merging them side by side
y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

In [187…
```
# Appending y_pred_1 and y_test_df
y_test_pred_final = pd.concat([y_test_df, y_pred_1], axis=1)
```

In [188…
```
y_test_pred_final.head()
```

Out[188]:

	churn	CustID	0
0	0	5704	0.016571
1	0	64892	0.000682
2	0	39613	0.429515
3	0	93118	0.012149
4	0	81235	0.039178

In [189…
```
# Renaming the '0' column as churn probablity
y_test_pred_final = y_test_pred_final.rename(columns={0:'churn_prob'})
```

In [204…
```
y_test_pred_final.head()
```

Out[204]:

	churn	CustID	churn_prob	test_predicted
0	0	5704	0.016571	0
1	0	64892	0.000682	0
2	0	39613	0.429515	0
3	0	93118	0.012149	0
4	0	81235	0.039178	0

In [205…
```
# In the test set using probablity cutoff 0.5, what we got in the train set
y_test_pred_final['test_predicted'] = y_test_pred_final['churn_prob'].map(lambda x: 1 if x > 0.5 else 0)
```

In [206…
```
y_test_pred_final.head()
```

Out[206]:

	churn	CustID	churn_prob	test_predicted
0	0	5704	0.016571	0
1	0	64892	0.000682	0
2	0	39613	0.429515	0
3	0	93118	0.012149	0
4	0	81235	0.039178	0

In [207…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_test_pred_final['churn'], y_test_pred_final['test_predicted'])
print(confusion)
```

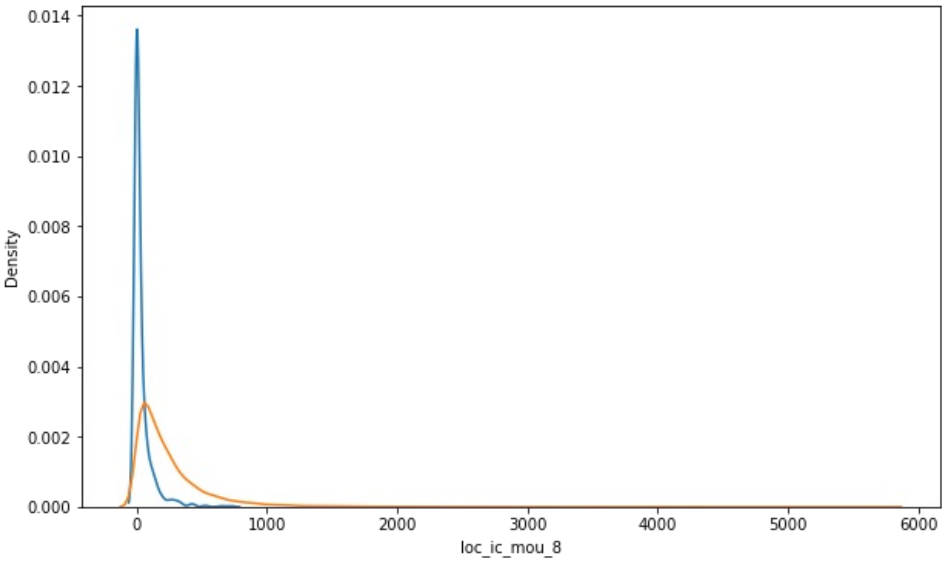
```
[[4089 1259]
 [  34  159]]
```

In [208…
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

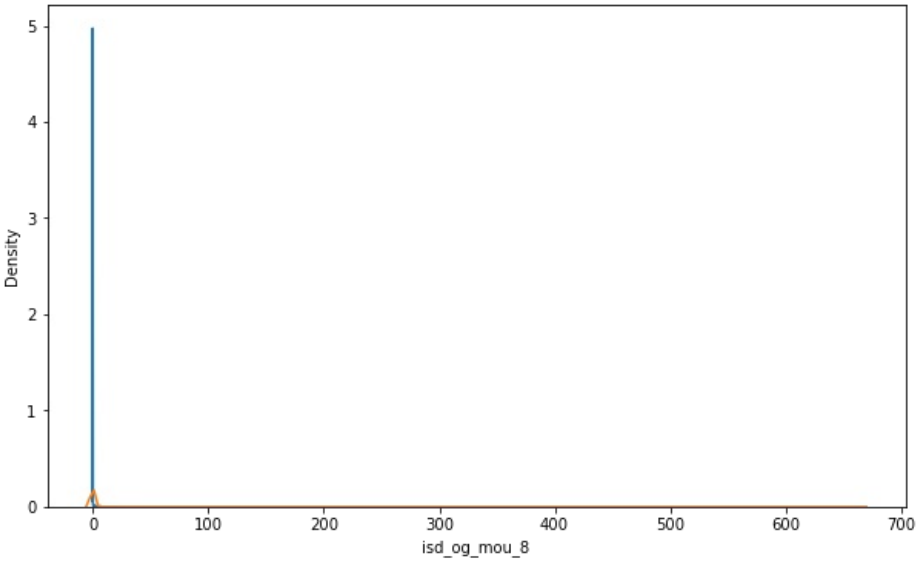
In [210…
```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test_pred_final['churn'], y_test_pred_final['test_predicted']))

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))

# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Accuracy:- 0.7666486193827828
Sensitivity:- 0.8238341968911918
Specificity:- 0.7645848915482424
```

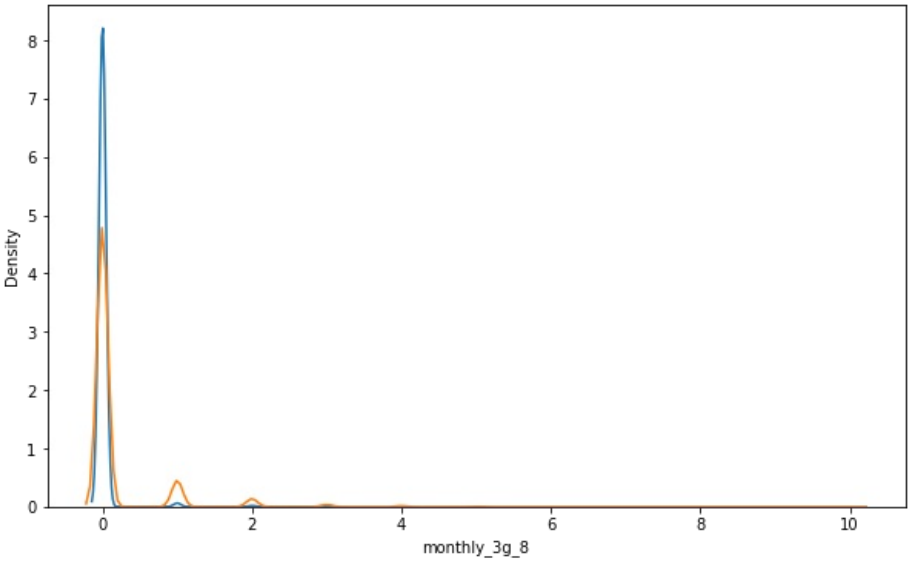
In [212…
```
# Plotting loc_ic_mou_8 predictor for churn and not churn customers
fig = plt.figure(figsize=(10,6))
sns.distplot(data_frame_churn['loc_ic_mou_8'],label='churn',hist=False)
sns.distplot(data_frame_non_churn['loc_ic_mou_8'],label='not churn',hist=False)
plt.show()
```



In [213…
```
# Plotting isd_og_mou_8 predictor for churn and not churn customers
fig = plt.figure(figsize=(10,6))
sns.distplot(data_frame_churn['isd_og_mou_8'],label='churn',hist=False)
sns.distplot(data_frame_non_churn['isd_og_mou_8'],label='not churn',hist=False)
plt.show()
```



In [214…
```
# Plotting monthly_3g_8 predictor for churn and not churn customers
fig = plt.figure(figsize=(10,6))
sns.distplot(data_frame_churn['monthly_3g_8'],label='churn',hist=False)
sns.distplot(data_frame_non_churn['monthly_3g_8'],label='not churn',hist=False)
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: