```python
import pandas
import numpy
import datetime
import math

import warnings
import missingno as msno

pandas.set_option('display.max_columns', 500)
warnings.simplefilter('ignore')

import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap

from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from sklearn.metrics import precision_recall_curve
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn import svm

import statsmodels.api as sm
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.linear_model import BayesianRidge
from sklearn.impute import KNNImputer

from imblearn.over_sampling import SMOTE
from collections import Counter
```

Setting Up

Visualization Functions

```python
def ratio_plot_data(data, var, target, nBins):
    plot_data = data.loc[data[var].isnull()==False]
```

```
    m=min(data[var])
    M=max(data[var])

    bins=list(numpy.linspace(m,M,nBins+1,endpoint=True))

    plot_data[var+'_bins'] = pandas.cut(plot_data[var],bins=bins)
    plot_data = plot_data.groupby(var+'_bins').agg({target:
['sum','count']})
    plot_data.columns = ['issue_count', 'total_count']
    plot_data['issue%'] = plot_data['issue_count'] /
plot_data['total_count']
    plot_data = plot_data.reset_index()
    plot_data[var+'_bins'] = plot_data[var+'_bins'].apply(lambda
x:str(round(x.left,1))+'-'+str(round(x.right,1)))
    return plot_data

def ratio_plots_num(data, var_list, target, nBins, c_palette):
    var_list_2 = []
    for var in var_list:
        try:
            plot_data = ratio_plot_data(data, var, target, nBins)
            var_list_2.append(var)
        except:
            continue
    cols = 3
    rows = math.ceil(len(var_list_2)/3)
    plt.subplots(rows, cols, figsize=(20, rows * 7))
    index = 1
    for var in var_list_2:
        plt.subplot(rows, cols, index)
        try:
            plot_data = ratio_plot_data(data, var, target, nBins)
        except:
            continue
        sns.barplot(plot_data[var+'_bins'].astype('str'),
plot_data['issue%'], palette=c_palette)
        plt.xticks(rotation=45)
        plt.title(var)
        plt.xlabel(var)
        plt.ylabel("Converted %")
        index=index+1
    plt.tight_layout()
    plt.show()
```

Utility Functions

```
def format_date(x):
    if str(x) == 'nan':
        return x
    else:
        return datetime.datetime.strptime(x,"%m/%d/%Y")
```

```
def imputation(data, column, value):
    data.loc[data[column].isnull()==True,column] = value
    return data


def missing_value_percentage(data):
    missing_value_summary =
pandas.DataFrame(data.isnull().sum()*100/data.shape[0])
    missing_value_summary.columns = ['Invalid Data %']
    missing_value_summary =
missing_value_summary.loc[missing_value_summary['Invalid Data
%']>0].sort_values('Invalid Data %', ascending=False)
    return missing_value_summary

def create_outlier_df(data, var_list):

outlier_df=pandas.DataFrame(columns=['ColumnName','OutlierCount','Outl
ier%'])
    for var in var_list:
        try:
            Q1 = data[var].quantile(0.25)
            Q3 = data[var].quantile(0.75)
            IQR = Q3 - Q1
            outlier_count=((data[var] < (Q1 - (1.5 * IQR))) |
(data[var] > (Q3 + (1.5 * IQR)))).sum()
            new_row = {
                        'ColumnName':var,
'OutlierCount':outlier_count,
                        'Outlier
%':round(outlier_count*100/data[var].shape[0],2)
                        }
            outlier_df=outlier_df.append(new_row,ignore_index=True)
        except TypeError:
            print('Error with column '+var)
    return outlier_df

def cap_outlier(data, var_list):
    for col in var_list:
        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)
        IQR = Q3 - Q1
        data[col][data[col] <= (Q1 - 1.5 * IQR)] = (Q1 - 1.5 * IQR)
        data[col][data[col] >= (Q3 + 1.5 * IQR)] = (Q3 + 1.5 * IQR)
    return data

def vif_ranks(data, features, row_count):
    vif = pandas.DataFrame()
    vif['Features'] = data[features].columns
```

```
    vif['VIF'] = [variance_inflation_factor(X_train[features].values, 
i) for i in range(X_train[features].shape[1])]
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by = "VIF", ascending = False)
    return vif.head(row_count)

def create_correlation_df(data,var_list,threshold):
    listi=[]
    listj=[]
    data=data[var_list]
    resultDf=pandas.DataFrame(columns=['Feature 1','Feature 
2','Correlation Value'])
    corrDf=data.corr()
    for i in corrDf.columns:
        for j in corrDf.columns:
            if i==j:
                break
            if (corrDf.loc[i,j] >=threshold) and (str(i)!=str(j)):

                new_row = {
                        'Feature 1':str(i), 'Feature 2':str(j),
                        'Correlation Value':round(corrDf.loc[i,j],2)
                        }
                resultDf=resultDf.append(new_row,ignore_index=True)
    return resultDf
```

Performance Metric Functions

```
def predict_summarize(predicted, actual, threshold, plot_roc_=False):
    y_pred_final = pandas.DataFrame({'Converted':actual, 
'Converted_Probability':predicted})
    y_pred_final['CustID'] = range(len(predicted))
    y_pred_final['predicted'] = 
y_pred_final['Converted_Probability'].map(lambda x: 1 if x > threshold 
else 0)

    # Confusion matrix
    confusion = metrics.confusion_matrix(y_pred_final['Converted'], 
y_pred_final['predicted'] )

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

    sensitivity = TP / float(TP+FN)
    specificity = TN / float(TN+FP)
    false_positive_rate = FP/ float(TN+FP)
    precision = TP / float(TP + FP)
    recall = TP / float(TP + FN)
```

```
    confusion = pandas.DataFrame(confusion)
    confusion.columns = ['predicted_no','predicted_yes']
    confusion['ind'] = ['actual_no','actual_yes']
    confusion = confusion.set_index('ind')

    print('Accuracy =
',metrics.accuracy_score(y_pred_final['Converted'],
y_pred_final.predicted))
    print('Sensitivity = ',sensitivity)
    print('Specificity = ',specificity)
    print('False Positive Rate = ',false_positive_rate)
    print('\nPrecision = ',precision)
    print('Recall = ',recall)

    if plot_roc_:
        plot_roc(y_pred_final['Converted'],
y_pred_final['Converted_Probability'])


    return confusion

def plot_roc( actual, probs ):
    print('Plotting')
    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

def plot_precision_recall_curve(predicted, actual):
    y_pred_final = pandas.DataFrame({'Converted':actual,
'Converted_Probability':predicted})
    y_pred_final['CustID'] = range(len(actual))

    p, r, thresholds =
precision_recall_curve(y_pred_final['Converted'],
y_pred_final['Converted_Probability'])
    plt.plot(thresholds, p[:-1], "g-",label='precision')
```

```
    plt.plot(thresholds, r[:-1], "r-",label='recall')
    plt.xlabel('Thresholds')
    plt.title('Precision Recall Curve')
    plt.legend()
    plt.show()

def plot_feature_importance(features, parameters):
    feature_importance = pandas.DataFrame(list(zip(features,
parameters)))
    feature_importance.columns = ['column', 'value']
    feature_importance['abs_value'] = abs(feature_importance['value'])
    feature_importance =
feature_importance.sort_values('abs_value',ascending=False)
    feature_importance =
feature_importance.loc[feature_importance['abs_value']>0]
    plt.figure(figsize=(10,10))

sns.barplot(feature_importance['value'],feature_importance['column'],p
alette='husl')
    plt.title('Feature Importances')
```

Data Preparation

```
data = pandas.read_csv('..//Data//telecom_churn_data.csv')

data.shape

(99999, 226)

data.head()

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

  last_date_of_month_6 last_date_of_month_7 last_date_of_month_8  \
0            6/30/2014            7/31/2014            8/31/2014
1            6/30/2014            7/31/2014            8/31/2014
2            6/30/2014            7/31/2014            8/31/2014
3            6/30/2014            7/31/2014            8/31/2014
4            6/30/2014            7/31/2014            8/31/2014
```

```
  last_date_of_month_9   arpu_6   arpu_7   arpu_8   arpu_9
onnet_mou_6  \
0            9/30/2014  197.385  214.816  213.803   21.100
NaN
1            9/30/2014   34.047  355.074  268.321   86.285
24.11
2            9/30/2014  167.690  189.058  210.226  290.714
11.54
3            9/30/2014  221.338  251.102  508.054  389.500
99.91
4            9/30/2014  261.636  309.876  238.174  163.426
50.31

   onnet_mou_7  onnet_mou_8  onnet_mou_9  offnet_mou_6
offnet_mou_7  \
0          NaN         0.00          NaN           NaN           NaN

1        78.68         7.68        18.34         15.74         99.84

2        55.24        37.26        74.81        143.33        220.59

3        54.39       310.98       241.71        123.31        109.01

4       149.44        83.89        58.78         76.96         91.88


   offnet_mou_8  offnet_mou_9  roam_ic_mou_6  roam_ic_mou_7
roam_ic_mou_8  \
0          0.00           NaN            NaN            NaN
0.00
1        304.76         53.76            0.0           0.00
0.00
2        208.36        118.91            0.0           0.00
0.00
3         71.68        113.54            0.0          54.86
44.38
4        124.26         45.81            0.0           0.00
0.00

   roam_ic_mou_9  roam_og_mou_6  roam_og_mou_7  roam_og_mou_8
roam_og_mou_9  \
0            NaN            NaN            NaN           0.00
NaN
1           0.00            0.0           0.00           0.00
0.00
2          38.49            0.0           0.00           0.00
70.94
3           0.00            0.0          28.09          39.04
```

```
0.00
4           0.00            0.0           0.00           0.00
0.00

   loc_og_t2t_mou_6  loc_og_t2t_mou_7  loc_og_t2t_mou_8
loc_og_t2t_mou_9  \
0               NaN               NaN              0.00
NaN
1             23.88             74.56              7.68
18.34
2              7.19             28.74             13.58
14.39
3             73.68             34.81             10.61
15.49
4             50.31            149.44             83.89
58.78

   loc_og_t2m_mou_6  loc_og_t2m_mou_7  loc_og_t2m_mou_8
loc_og_t2m_mou_9  \
0               NaN               NaN              0.00
NaN
1             11.51             75.94            291.86
53.76
2             29.34             16.86             38.46
28.16
3            107.43             83.21             22.46
65.46
4             67.64             91.88            124.26
37.89

   loc_og_t2f_mou_6  loc_og_t2f_mou_7  loc_og_t2f_mou_8
loc_og_t2f_mou_9  \
0               NaN               NaN              0.00
NaN
1              0.00              0.00              0.00
0.00
2             24.11             21.79             15.61
22.24
3              1.91              0.65              4.91
2.06
4              0.00              0.00              0.00
1.93

   loc_og_t2c_mou_6  loc_og_t2c_mou_7  loc_og_t2c_mou_8
loc_og_t2c_mou_9  \
0               NaN               NaN              0.00
NaN
1               0.0              2.91              0.00
0.00
2               0.0            135.54             45.76
```

```
0.48
3              0.0              0.00             0.00
0.00
4              0.0              0.00             0.00
0.00

   loc_og_mou_6  loc_og_mou_7  loc_og_mou_8  loc_og_mou_9
std_og_t2t_mou_6  \
0           NaN           NaN          0.00           NaN
NaN
1         35.39        150.51        299.54         72.11
0.23
2         60.66         67.41         67.66         64.81
4.34
3        183.03        118.68         37.99         83.03
26.23
4        117.96        241.33        208.16         98.61
0.00

   std_og_t2t_mou_7  std_og_t2t_mou_8  std_og_t2t_mou_9
std_og_t2m_mou_6  \
0               NaN              0.00               NaN
NaN
1              4.11              0.00              0.00
0.00
2             26.49             22.58              8.76
41.81
3             14.89            289.58            226.21
2.99
4              0.00              0.00              0.00
9.31

   std_og_t2m_mou_7  std_og_t2m_mou_8  std_og_t2m_mou_9
std_og_t2f_mou_6  \
0               NaN              0.00               NaN
NaN
1              0.46              0.13              0.00
0.00
2             67.41             75.53              9.28
1.48
3              1.73              6.53              9.99
0.00
4              0.00              0.00              0.00
0.00

   std_og_t2f_mou_7  std_og_t2f_mou_8  std_og_t2f_mou_9
std_og_t2c_mou_6  \
0               NaN              0.00               NaN
NaN
1              0.00              0.00               0.0
```

```
0.0
2             14.76              22.83               0.0
0.0
3              0.00               0.00               0.0
0.0
4              0.00               0.00               0.0
0.0

   std_og_t2c_mou_7  std_og_t2c_mou_8  std_og_t2c_mou_9  std_og_mou_6
\
0                NaN               0.0               NaN           NaN

1                0.0               0.0               0.0          0.23

2                0.0               0.0               0.0         47.64

3                0.0               0.0               0.0         29.23

4                0.0               0.0               0.0          9.31


   std_og_mou_7  std_og_mou_8  std_og_mou_9  isd_og_mou_6
isd_og_mou_7  \
0           NaN          0.00           NaN           NaN
NaN
1          4.58          0.13          0.00           0.0
0.0
2        108.68        120.94         18.04           0.0
0.0
3         16.63        296.11        236.21           0.0
0.0
4          0.00          0.00          0.00           0.0
0.0

   isd_og_mou_8  isd_og_mou_9  spl_og_mou_6  spl_og_mou_7
spl_og_mou_8  \
0           0.0           NaN           NaN           NaN
0.00
1           0.0           0.0          4.68         23.43
12.76
2           0.0           0.0         46.56        236.84
96.84
3           0.0           0.0         10.96          0.00
18.09
4           0.0           0.0          0.00          0.00
0.00

   spl_og_mou_9  og_others_6  og_others_7  og_others_8  og_others_9  \
0           NaN          NaN          NaN          0.0          NaN
```

```
1         0.00         0.00          0.0          0.0          0.0
2        42.08         0.45          0.0          0.0          0.0
3        43.29         0.00          0.0          0.0          0.0
4         5.98         0.00          0.0          0.0          0.0

   total_og_mou_6  total_og_mou_7  total_og_mou_8  total_og_mou_9  \
0            0.00            0.00            0.00            0.00
1           40.31          178.53          312.44           72.11
2          155.33          412.94          285.46          124.94
3          223.23          135.31          352.21          362.54
4          127.28          241.33          208.16          104.59

   loc_ic_t2t_mou_6  loc_ic_t2t_mou_7  loc_ic_t2t_mou_8
loc_ic_t2t_mou_9  \
0               NaN               NaN              0.16
NaN
1              1.61             29.91             29.23
116.09
2            115.69             71.11             67.46
148.23
3             62.08             19.98              8.04
41.73
4            105.68             88.49            233.81
154.56

   loc_ic_t2m_mou_6  loc_ic_t2m_mou_7  loc_ic_t2m_mou_8
loc_ic_t2m_mou_9  \
0               NaN               NaN              4.13
NaN
1             17.48             65.38            375.58
56.93
2             14.38             15.44             38.89
38.98
3            113.96             64.51             20.28
52.86
4            106.84            109.54            104.13
48.24

   loc_ic_t2f_mou_6  loc_ic_t2f_mou_7  loc_ic_t2f_mou_8
loc_ic_t2f_mou_9  \
0               NaN               NaN              1.15
NaN
1              0.00              8.93              3.61
0.00
2             99.48            122.29             49.63
158.19
3             57.43             27.09             19.84
65.59
4              1.50              0.00              0.00
0.00
```

```
   loc_ic_mou_6  loc_ic_mou_7  loc_ic_mou_8  loc_ic_mou_9
std_ic_t2t_mou_6  \
0           NaN           NaN          5.44           NaN
NaN
1         19.09        104.23        408.43        173.03
0.00
2        229.56        208.86        155.99        345.41
72.41
3        233.48        111.59         48.18        160.19
43.48
4        214.03        198.04        337.94        202.81
0.00

   std_ic_t2t_mou_7  std_ic_t2t_mou_8  std_ic_t2t_mou_9
std_ic_t2m_mou_6  \
0               NaN              0.00               NaN
NaN
1              0.00              2.35              0.00
5.90
2             71.29             28.69             49.44
45.18
3             66.44              0.00            129.84
1.33
4              0.00              0.86              2.31
1.93

   std_ic_t2m_mou_7  std_ic_t2m_mou_8  std_ic_t2m_mou_9
std_ic_t2f_mou_6  \
0               NaN              0.00               NaN
NaN
1              0.00             12.49             15.01
0.00
2            177.01            167.09            118.18
21.73
3             38.56              4.94             13.98
1.18
4              0.25              0.00              0.00
0.00

   std_ic_t2f_mou_7  std_ic_t2f_mou_8  std_ic_t2f_mou_9
std_ic_t2o_mou_6  \
0               NaN              0.00               NaN
NaN
1              0.00              0.00              0.00
0.0
2             58.34             43.23              3.86
0.0
3              0.00              0.00              0.00
0.0
```

```
4               0.00                0.00                0.00
0.0

   std_ic_t2o_mou_7  std_ic_t2o_mou_8  std_ic_t2o_mou_9  std_ic_mou_6
\
0                NaN               0.0               NaN           NaN

1                0.0               0.0               0.0          5.90

2                0.0               0.0               0.0        139.33

3                0.0               0.0               0.0         45.99

4                0.0               0.0               0.0          1.93


   std_ic_mou_7  std_ic_mou_8  std_ic_mou_9  total_ic_mou_6
total_ic_mou_7  \
0           NaN          0.00           NaN            0.00
0.00
1          0.00         14.84         15.01           26.83
104.23
2        306.66        239.03        171.49          370.04
519.53
3        105.01          4.94        143.83          280.08
216.61
4          0.25          0.86          2.31          216.44
198.29

   total_ic_mou_8  total_ic_mou_9  spl_ic_mou_6  spl_ic_mou_7
spl_ic_mou_8  \
0            5.44            0.00           NaN           NaN
0.0
1          423.28          188.04          0.00           0.0
0.0
2          395.03          517.74          0.21           0.0
0.0
3           53.13          305.38          0.59           0.0
0.0
4          338.81          205.31          0.00           0.0
0.0

   spl_ic_mou_9  isd_ic_mou_6  isd_ic_mou_7  isd_ic_mou_8
isd_ic_mou_9  \
0           NaN           NaN           NaN           0.0
NaN
1          0.00          1.83          0.00           0.0
0.00
2          0.45          0.00          0.85           0.0
```

```
0.01
3          0.55          0.00          0.00           0.0
0.00
4          0.18          0.00          0.00           0.0
0.00

   ic_others_6  ic_others_7  ic_others_8  ic_others_9
total_rech_num_6  \
0          NaN          NaN          0.0          NaN
4
1         0.00         0.00          0.0         0.00
4
2         0.93         3.14          0.0         0.36
5
3         0.00         0.00          0.0         0.80
10
4         0.48         0.00          0.0         0.00
5

   total_rech_num_7  total_rech_num_8  total_rech_num_9
total_rech_amt_6  \
0                 3                 2                 6
362
1                 9                11                 5
74
2                 4                 2                 7
168
3                11                18                14
230
4                 6                 3                 4
196

   total_rech_amt_7  total_rech_amt_8  total_rech_amt_9
max_rech_amt_6  \
0               252               252                 0
252
1               384               283               121
44
2               315               116               358
86
3               310               601               410
60
4               350               287               200
56

   max_rech_amt_7  max_rech_amt_8  max_rech_amt_9 date_of_last_rech_6
\
0             252             252               0           6/21/2014

1             154              65              50           6/29/2014
```

```
2            200             86             100          6/17/2014

3             50             50              50          6/28/2014

4            110            110              50          6/26/2014


  date_of_last_rech_7 date_of_last_rech_8 date_of_last_rech_9  \
0           7/16/2014            8/8/2014           9/28/2014
1           7/31/2014           8/28/2014           9/30/2014
2           7/24/2014           8/14/2014           9/29/2014
3           7/31/2014           8/31/2014           9/30/2014
4           7/28/2014            8/9/2014           9/28/2014

   last_day_rch_amt_6  last_day_rch_amt_7  last_day_rch_amt_8  \
0                 252                 252                 252
1                  44                  23                  30
2                   0                 200                  86
3                  30                  50                  50
4                  50                 110                 110

   last_day_rch_amt_9 date_of_last_rech_data_6
date_of_last_rech_data_7  \
0                   0                6/21/2014
7/16/2014
1                   0                      NaN
7/25/2014
2                   0                      NaN
NaN
3                  30                      NaN
NaN
4                  50                 6/4/2014
NaN

  date_of_last_rech_data_8 date_of_last_rech_data_9  total_rech_data_6
\
0                 8/8/2014                      NaN                1.0

1                8/10/2014                      NaN                NaN

2                      NaN                9/17/2014                NaN

3                      NaN                      NaN                NaN

4                      NaN                      NaN                1.0


   total_rech_data_7  total_rech_data_8  total_rech_data_9
```

```
max_rech_data_6  \
0                1.0               1.0               NaN
252.0
1                1.0               2.0               NaN
NaN
2                NaN               NaN               1.0
NaN
3                NaN               NaN               NaN
NaN
4                NaN               NaN               NaN
56.0

   max_rech_data_7  max_rech_data_8  max_rech_data_9  count_rech_2g_6
\
0            252.0            252.0              NaN              0.0

1            154.0             25.0              NaN              NaN

2              NaN              NaN             46.0              NaN

3              NaN              NaN              NaN              NaN

4              NaN              NaN              NaN              1.0


   count_rech_2g_7  count_rech_2g_8  count_rech_2g_9  count_rech_3g_6
\
0              0.0              0.0              NaN              1.0

1              1.0              2.0              NaN              NaN

2              NaN              NaN              1.0              NaN

3              NaN              NaN              NaN              NaN

4              NaN              NaN              NaN              0.0


   count_rech_3g_7  count_rech_3g_8  count_rech_3g_9
av_rech_amt_data_6  \
0              1.0              1.0              NaN
252.0
1              0.0              0.0              NaN
NaN
2              NaN              NaN              0.0
NaN
3              NaN              NaN              NaN
NaN
4              NaN              NaN              NaN
```

```
56.0

   av_rech_amt_data_7  av_rech_amt_data_8  av_rech_amt_data_9
vol_2g_mb_6  \
0               252.0               252.0                 NaN
30.13
1               154.0                50.0                 NaN
0.00
2                 NaN                 NaN                46.0
0.00
3                 NaN                 NaN                 NaN
0.00
4                 NaN                 NaN                 NaN
0.00

   vol_2g_mb_7  vol_2g_mb_8  vol_2g_mb_9  vol_3g_mb_6  vol_3g_mb_7  \
0         1.32         5.75          0.0        83.57       150.76
1       108.07       365.47          0.0         0.00         0.00
2         0.00         0.00          0.0         0.00         0.00
3         0.00         0.00          0.0         0.00         0.00
4         0.00         0.00          0.0         0.00         0.00

   vol_3g_mb_8  vol_3g_mb_9  arpu_3g_6  arpu_3g_7  arpu_3g_8
arpu_3g_9  \
0       109.61         0.00     212.17     212.17     212.17
NaN
1         0.00         0.00        NaN       0.00       0.00
NaN
2         0.00         8.42        NaN        NaN        NaN
2.84
3         0.00         0.00        NaN        NaN        NaN
NaN
4         0.00         0.00       0.00        NaN        NaN
NaN

   arpu_2g_6  arpu_2g_7  arpu_2g_8  arpu_2g_9  night_pck_user_6  \
0     212.17     212.17     212.17        NaN               0.0
1        NaN      28.61       7.60        NaN               NaN
2        NaN        NaN        NaN        0.0               NaN
3        NaN        NaN        NaN        NaN               NaN
4       0.00        NaN        NaN        NaN               0.0

   night_pck_user_7  night_pck_user_8  night_pck_user_9  monthly_2g_6
\
0               0.0               0.0               NaN             0

1               0.0               0.0               NaN             0

2               NaN               NaN               0.0             0
```

```
3               NaN               NaN               NaN              0

4               NaN               NaN               NaN              0


   monthly_2g_7  monthly_2g_8  monthly_2g_9  sachet_2g_6  sachet_2g_7
\
0             0             0             0            0            0

1             1             0             0            0            0

2             0             0             0            0            0

3             0             0             0            0            0

4             0             0             0            1            0


   sachet_2g_8  sachet_2g_9  monthly_3g_6  monthly_3g_7  monthly_3g_8
\
0            0            0             1             1             1

1            2            0             0             0             0

2            0            1             0             0             0

3            0            0             0             0             0

4            0            0             0             0             0


   monthly_3g_9  sachet_3g_6  sachet_3g_7  sachet_3g_8  sachet_3g_9  \
0             0            0            0            0            0
1             0            0            0            0            0
2             0            0            0            0            0
3             0            0            0            0            0
4             0            0            0            0            0

   fb_user_6  fb_user_7  fb_user_8  fb_user_9   aon  aug_vbc_3g
jul_vbc_3g  \
0        1.0        1.0        1.0        NaN   968        30.4
0.0
1        NaN        1.0        1.0        NaN  1006         0.0
0.0
2        NaN        NaN        NaN        1.0  1103         0.0
0.0
3        NaN        NaN        NaN        NaN  2491         0.0
```

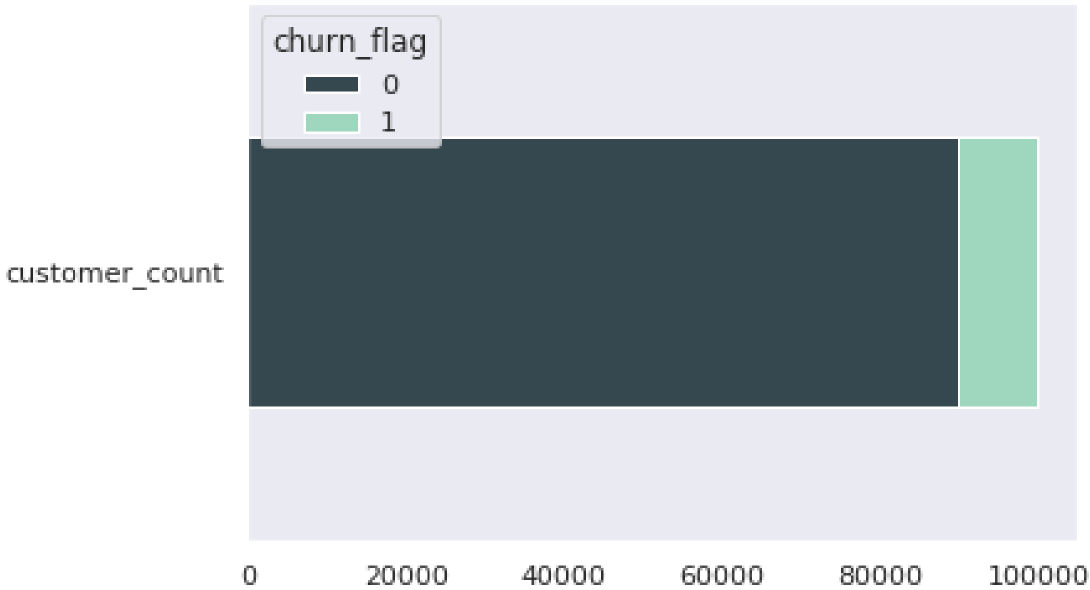
```
0.0
4        0.0        NaN        NaN        NaN  1526         0.0
0.0

   jun_vbc_3g  sep_vbc_3g
0      101.20        3.58
1        0.00        0.00
2        4.17        0.00
3        0.00        0.00
4        0.00        0.00
```

```
data.drop(['last_date_of_month_6', 'last_date_of_month_7',
           'last_date_of_month_8', 'last_date_of_month_9',], axis=1,
inplace=True)

data['date_of_last_rech_data_6'] =
data['date_of_last_rech_data_6'].apply(lambda x:format_date(x))
data['date_of_last_rech_data_7'] =
data['date_of_last_rech_data_7'].apply(lambda x:format_date(x))
data['date_of_last_rech_data_8'] =
data['date_of_last_rech_data_8'].apply(lambda x:format_date(x))
data['date_of_last_rech_data_9'] =
data['date_of_last_rech_data_9'].apply(lambda x:format_date(x))

data['date_of_last_rech_6'] = data['date_of_last_rech_6'].apply(lambda
x:format_date(x))
data['date_of_last_rech_7'] = data['date_of_last_rech_7'].apply(lambda
x:format_date(x))
data['date_of_last_rech_8'] = data['date_of_last_rech_8'].apply(lambda
x:format_date(x))
data['date_of_last_rech_9'] = data['date_of_last_rech_9'].apply(lambda
x:format_date(x))
```

Tag Churners (Target Variable)

```
condition = (data['total_ic_mou_9'] == 0) & (data['total_og_mou_9'] ==
0) &\
            (data['vol_2g_mb_9'] == 0) & (data['vol_3g_mb_9'] == 0)

data['churn_flag'] = 0
data.loc[condition, 'churn_flag'] = 1

churn_summary = pandas.DataFrame(data.groupby('churn_flag')
['mobile_number'].count())
churn_summary.columns = ['customer_count']
sns.set()
churn_summary.T.plot(kind='barh', stacked=True,

colormap=ListedColormap(sns.color_palette("GnBu_d", 10)), grid=False)
plt.show()
churn_summary['customer(%)'] = round((churn_summary['customer_count']
```

```
/ churn_summary['customer_count'].sum()) * 100, 0)
display(churn_summary)
```



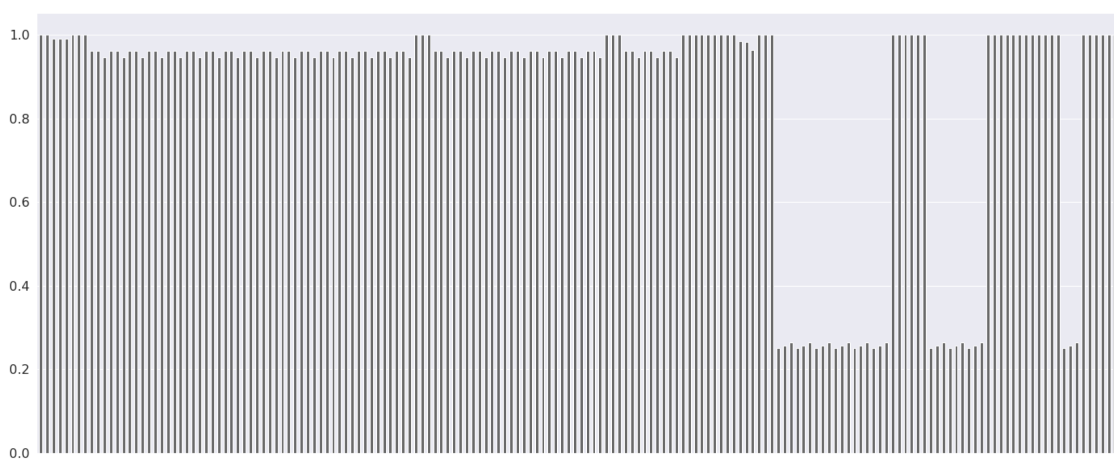

```
            customer_count  customer(%)
churn_flag
0                    89808         90.0
1                    10191         10.0
```

Eliminate Churn Month Data To Avoid Data Leakage

```
churn_month_columns = [x for x in data.columns if x[-1]=='9']
data = data.drop(churn_month_columns, axis=1)
```

Missing Value Checks

```
msno.bar(data)
plt.show()
```



```
missing_value_columns = missing_value_percentage(data)
print(len(missing_value_columns), '')
```

```
123
```

```
for column in missing_value_columns.loc[missing_value_columns['Invalid
Data %']>0].index.to_list():
    if 'mou' in column:
        print('Imputing ',column)
        data[column] = data[column].fillna(0)
    elif 'count' in column:
        print('Imputing ',column)
        data[column] = data[column].fillna(0)
```

```
Imputing  count_rech_2g_6
Imputing  count_rech_3g_6
Imputing  count_rech_2g_7
Imputing  count_rech_3g_7
Imputing  count_rech_2g_8
Imputing  count_rech_3g_8
Imputing  isd_og_mou_8
Imputing  isd_ic_mou_8
Imputing  spl_ic_mou_8
Imputing  spl_og_mou_8
Imputing  std_ic_mou_8
Imputing  loc_ic_mou_8
Imputing  std_ic_t2t_mou_8
Imputing  loc_ic_t2t_mou_8
Imputing  std_ic_t2f_mou_8
Imputing  loc_ic_t2m_mou_8
Imputing  std_ic_t2m_mou_8
Imputing  loc_ic_t2f_mou_8
Imputing  std_ic_t2o_mou_8
Imputing  std_og_mou_8
Imputing  std_og_t2t_mou_8
Imputing  std_og_t2m_mou_8
Imputing  loc_og_t2t_mou_8
Imputing  loc_og_t2f_mou_8
Imputing  loc_og_t2c_mou_8
Imputing  roam_og_mou_8
Imputing  loc_og_mou_8
Imputing  roam_ic_mou_8
Imputing  loc_og_t2m_mou_8
Imputing  offnet_mou_8
Imputing  std_og_t2f_mou_8
Imputing  std_og_t2c_mou_8
Imputing  onnet_mou_8
Imputing  spl_ic_mou_6
Imputing  roam_ic_mou_6
Imputing  loc_og_t2m_mou_6
Imputing  std_ic_mou_6
```

```
Imputing  roam_og_mou_6
Imputing  offnet_mou_6
Imputing  std_ic_t2o_mou_6
Imputing  isd_ic_mou_6
Imputing  loc_og_t2t_mou_6
Imputing  onnet_mou_6
Imputing  std_ic_t2f_mou_6
Imputing  isd_og_mou_6
Imputing  std_ic_t2t_mou_6
Imputing  std_ic_t2m_mou_6
Imputing  loc_og_t2f_mou_6
Imputing  loc_og_t2c_mou_6
Imputing  loc_ic_mou_6
Imputing  loc_og_mou_6
Imputing  loc_ic_t2f_mou_6
Imputing  std_og_t2t_mou_6
Imputing  loc_ic_t2m_mou_6
Imputing  std_og_t2m_mou_6
Imputing  loc_ic_t2t_mou_6
Imputing  std_og_t2f_mou_6
Imputing  std_og_t2c_mou_6
Imputing  spl_og_mou_6
Imputing  std_og_mou_6
Imputing  loc_ic_t2f_mou_7
Imputing  spl_og_mou_7
Imputing  onnet_mou_7
Imputing  offnet_mou_7
Imputing  roam_ic_mou_7
Imputing  roam_og_mou_7
Imputing  loc_og_t2t_mou_7
Imputing  loc_og_t2m_mou_7
Imputing  loc_og_t2f_mou_7
Imputing  loc_og_t2c_mou_7
Imputing  loc_og_mou_7
Imputing  std_og_t2t_mou_7
Imputing  std_og_t2m_mou_7
Imputing  std_og_t2f_mou_7
Imputing  std_og_t2c_mou_7
Imputing  std_og_mou_7
Imputing  isd_og_mou_7
Imputing  std_ic_t2f_mou_7
Imputing  loc_ic_t2t_mou_7
Imputing  loc_ic_t2m_mou_7
Imputing  loc_ic_mou_7
Imputing  std_ic_t2t_mou_7
Imputing  std_ic_t2m_mou_7
Imputing  std_ic_t2o_mou_7
Imputing  std_ic_mou_7
Imputing  spl_ic_mou_7
Imputing  isd_ic_mou_7
```

```
Imputing  std_og_t2o_mou
Imputing  loc_ic_t2o_mou
Imputing  loc_og_t2o_mou
```

```
missing_value_columns = missing_value_percentage(data)
```

Observation:

Drop columns with more than 30% missing values

```
drop_columns_list =
missing_value_columns.loc[missing_value_columns['Invalid Data
%']>30].index
data = data.drop(drop_columns_list, axis=1)
print(len(drop_columns_list), ' columns have been dropped.')
```

```
24  columns have been dropped.
```

```
drop_columns_list
```

```
Index(['arpu_2g_6', 'av_rech_amt_data_6', 'fb_user_6', 'arpu_3g_6',
       'night_pck_user_6', 'max_rech_data_6',
'date_of_last_rech_data_6',
       'total_rech_data_6', 'max_rech_data_7', 'arpu_3g_7',
       'total_rech_data_7', 'date_of_last_rech_data_7', 'arpu_2g_7',
       'night_pck_user_7', 'fb_user_7', 'av_rech_amt_data_7',
       'av_rech_amt_data_8', 'arpu_3g_8', 'arpu_2g_8',
'night_pck_user_8',
       'fb_user_8', 'max_rech_data_8', 'total_rech_data_8',
       'date_of_last_rech_data_8'],
      dtype='object')
```

Observation:

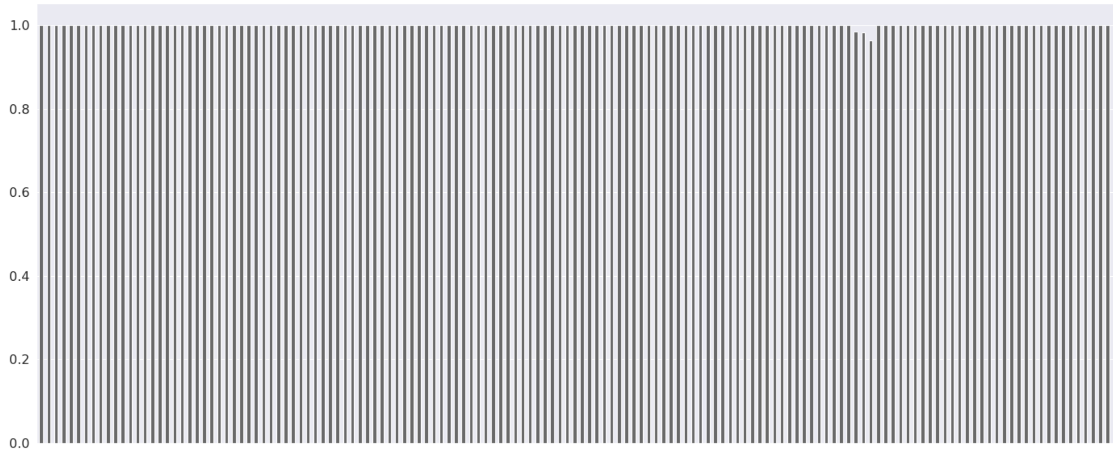
For the remaining columns, with less than 60% of the data missing, we can use iterative imputer if the columns are not like dates, which we can treat later after deriving metrics out of them.

```
missing_value_columns =
missing_value_columns.loc[missing_value_columns['Invalid Data %']<=30]
missing_value_columns =
pandas.DataFrame(data[missing_value_columns.index].dtypes)
missing_value_columns =
missing_value_columns.loc[missing_value_columns[0]!
='datetime64[ns]',:].index

print(len(missing_value_columns), 'columns have to be imputed.')
```

```
6 columns have to be imputed.
```

```
imputer = imputer = IterativeImputer(BayesianRidge())
data[missing_value_columns] =
imputer.fit_transform(data[missing_value_columns])
```

```
msno.bar(data)
plt.show()
```



Save the data and reuse to avoid re running of imputation

Filter: High Valued Customers

```
percentile_70 = ((data['total_rech_amt_6'] +
data['total_rech_amt_7'])/2).quantile(0.70)
print(percentile_70, 'is the 70th percentile of the average recharge
amount.')
```

```
368.5 is the 70th percentile of the average recharge amount.
```

```
data = data.loc[((data['total_rech_amt_6'] +
data['total_rech_amt_7'])/2) >= percentile_70,:]
print(data.shape[0],' customers are high valued customers and are
hence retained in the data.')
```

```
30011  customers are high valued customers and are hence retained in
the data.
```

Feature Creation

As defined, there are mainly 3 phases: good, action & churn. While we have only one month data for each of the action & churn phases, we have 2 months data for the good phase. We can create features by taking the average of the good phases information. This will help us narrow down on the features to optimum level

```
data.head()
```

```
    mobile_number  circle_id  loc_og_t2o_mou  std_og_t2o_mou
loc_ic_t2o_mou  \
7      7000701601        109             0.0             0.0
```

```
0.0
8      7001524846        109            0.0            0.0
0.0
13     7002191713        109            0.0            0.0
0.0
16     7000875565        109            0.0            0.0
0.0
17     7000187447        109            0.0            0.0
0.0

      arpu_6    arpu_7    arpu_8  onnet_mou_6  onnet_mou_7
onnet_mou_8  \
7   1069.180  1349.850  3171.480        57.84        54.68
52.29
8    378.721   492.223   137.362       413.69       351.03
35.08
13   492.846   205.671   593.260       501.76       108.39
534.24
16   430.975   299.869   187.894        50.51        74.01
70.61
17   690.008    18.980    25.499      1185.91         9.28
7.79

    offnet_mou_6  offnet_mou_7  offnet_mou_8  roam_ic_mou_6
roam_ic_mou_7  \
7         453.43        567.16        325.91          16.23
33.49
8          94.66         80.63        136.48           0.00
0.00
13        413.31        119.28        482.46          23.53
144.24
16        296.29        229.74        162.76           0.00
2.83
17         61.64          0.00          5.54           0.00
4.76

    roam_ic_mou_8  roam_og_mou_6  roam_og_mou_7  roam_og_mou_8  \
7           31.64          23.74          12.59          38.06
8            0.00           0.00           0.00           0.00
13          72.11           7.98          35.26           1.44
16           0.00           0.00          17.74           0.00
17           4.81           0.00           8.46          13.34

    loc_og_t2t_mou_6  loc_og_t2t_mou_7  loc_og_t2t_mou_8
loc_og_t2m_mou_6  \
7              51.39             31.38             40.28
308.63
8             297.13            217.59             12.49
80.96
13             49.63              6.19             36.01
```

```
151.13
16              42.61              65.16              67.38
273.29
17              38.99               0.00               0.00
58.54

    loc_og_t2m_mou_7  loc_og_t2m_mou_8  loc_og_t2f_mou_6
loc_og_t2f_mou_7  \
7              447.38             162.28              62.13
55.14
8               70.58              50.54               0.00
0.00
13              47.28             294.46               4.54
0.00
16             145.99             128.28               0.00
4.48
17               0.00               0.00               0.00
0.00

    loc_og_t2f_mou_8  loc_og_t2c_mou_6  loc_og_t2c_mou_7
loc_og_t2c_mou_8  \
7               53.23                0.0                0.0
0.00
8                0.00                0.0                0.0
7.15
13              23.51                0.0                0.0
0.49
16              10.26                0.0                0.0
0.00
17               0.00                0.0                0.0
0.00

    loc_og_mou_6  loc_og_mou_7  loc_og_mou_8  std_og_t2t_mou_6  \
7         422.16        533.91        255.79              4.30
8         378.09        288.18         63.04            116.56
13        205.31         53.48        353.99            446.41
16        315.91        215.64        205.93              7.89
17         97.54          0.00          0.00           1146.91

    std_og_t2t_mou_7  std_og_t2t_mou_8  std_og_t2m_mou_6
std_og_t2m_mou_7  \
7               23.29              12.01              49.89
31.76
8              133.43              22.58              13.69
10.04
13              85.98             498.23             255.36
52.94
16               2.58               3.23              22.99
64.51
17               0.81               0.00               1.55
```

```
0.00

    std_og_t2m_mou_8  std_og_t2f_mou_6  std_og_t2f_mou_7
std_og_t2f_mou_8  \
7              49.14              6.66             20.08
16.68
8              75.69              0.00              0.00
0.00
13            156.94              0.00              0.00
0.00
16             18.29              0.00              0.00
0.00
17              0.00              0.00              0.00
0.00

    std_og_t2c_mou_6  std_og_t2c_mou_7  std_og_t2c_mou_8  std_og_mou_6
\
7                0.0               0.0               0.0         60.86

8                0.0               0.0               0.0        130.26

13               0.0               0.0               0.0        701.78

16               0.0               0.0               0.0         30.89

17               0.0               0.0               0.0       1148.46


    std_og_mou_7  std_og_mou_8  isd_og_mou_6  isd_og_mou_7
isd_og_mou_8  \
7          75.14         77.84           0.0          0.18
10.01
8         143.48         98.28           0.0          0.00
0.00
13        138.93        655.18           0.0          0.00
1.29
16         67.09         21.53           0.0          0.00
0.00
17          0.81          0.00           0.0          0.00
0.00

    spl_og_mou_6  spl_og_mou_7  spl_og_mou_8  og_others_6  og_others_7
\
7           4.50          0.00          6.50         0.00          0.0

8           0.00          0.00         10.23         0.00          0.0

13          0.00          0.00          4.78         0.00          0.0
```

```
16          0.00          3.26          5.91         0.00          0.0

17          2.58          0.00          0.00         0.93          0.0


    og_others_8  total_og_mou_6  total_og_mou_7  total_og_mou_8  \
7           0.0          487.53          609.24          350.16
8           0.0          508.36          431.66          171.56
13          0.0          907.09          192.41         1015.26
16          0.0          346.81          286.01          233.38
17          0.0         1249.53            0.81            0.00

    loc_ic_t2t_mou_6  loc_ic_t2t_mou_7  loc_ic_t2t_mou_8
loc_ic_t2m_mou_6  \
7              58.14             32.26             27.31
217.56
8              23.84              9.84              0.31
57.58
13             67.88              7.58             52.58
142.88
16             41.33             71.44             28.89
226.81
17             34.54              0.00              0.00
47.41

    loc_ic_t2m_mou_7  loc_ic_t2m_mou_8  loc_ic_t2f_mou_6
loc_ic_t2f_mou_7  \
7             221.49            121.19            152.16
101.46
8              13.98             15.48              0.00
0.00
13             18.53            195.18              4.81
0.00
16            149.69            150.16              8.71
8.68
17              2.31              0.00              0.00
0.00

    loc_ic_t2f_mou_8  loc_ic_mou_6  loc_ic_mou_7  loc_ic_mou_8  \
7              39.53        427.88        355.23        188.04
8               0.00         81.43         23.83         15.79
13              7.49        215.58         26.11        255.26
16             32.71        276.86        229.83        211.78
17              0.00         81.96          2.31          0.00

    std_ic_t2t_mou_6  std_ic_t2t_mou_7  std_ic_t2t_mou_8
std_ic_t2m_mou_6  \
7              36.89             11.83             30.39
91.44
```

```
8               0.00             0.58             0.10
22.43
13            115.68            38.29           154.58
308.13
16             68.79            78.64             6.33
18.68
17              8.63             0.00             0.00
1.28

    std_ic_t2m_mou_7  std_ic_t2m_mou_8  std_ic_t2f_mou_6
std_ic_t2f_mou_7  \
7             126.99            141.33            52.19
34.24
8               4.08              0.65             0.00
0.00
13             29.79            317.91             0.00
0.00
16             73.08             73.93             0.51
0.00
17              0.00              0.00             0.00
0.00

    std_ic_t2f_mou_8  std_ic_t2o_mou_6  std_ic_t2o_mou_7
std_ic_t2o_mou_8  \
7              22.21               0.0               0.0
0.0
8               0.00               0.0               0.0
0.0
13              1.91               0.0               0.0
0.0
16              2.18               0.0               0.0
0.0
17              0.00               0.0               0.0
0.0

    std_ic_mou_6  std_ic_mou_7  std_ic_mou_8  total_ic_mou_6
total_ic_mou_7  \
7         180.54        173.08        193.94          626.46
558.04
8          22.43          4.66          0.75          103.86
28.49
13        423.81         68.09        474.41          968.61
172.58
16         87.99        151.73         82.44          364.86
381.56
17          9.91          0.00          0.00           91.88
2.31

    total_ic_mou_8  spl_ic_mou_6  spl_ic_mou_7  spl_ic_mou_8
isd_ic_mou_6  \
```

```
7           428.74          0.21           0.0           0.0
2.06
8            16.54          0.00           0.0           0.0
0.00
13         1144.53          0.45           0.0           0.0
245.28
16          294.46          0.00           0.0           0.0
0.00
17            0.00          0.00           0.0           0.0
0.00

    isd_ic_mou_7  isd_ic_mou_8  ic_others_6  ic_others_7  ic_others_8
\
7          14.53         31.59        15.74        15.19        15.14

8           0.00          0.00         0.00         0.00         0.00

13         62.11        393.39        83.48        16.24        21.44

16          0.00          0.23         0.00         0.00         0.00

17          0.00          0.00         0.00         0.00         0.00


    total_rech_num_6  total_rech_num_7  total_rech_num_8
total_rech_amt_6  \
7                  5                 5                 7
1580
8                 19                21                14
437
13                 6                 4                11
507
16                10                 6                 2
570
17                19                 2                 4
816

    total_rech_amt_7  total_rech_amt_8  max_rech_amt_6  max_rech_amt_7
\
7                790              3638            1580             790

8                601               120              90             154

13               253               717             110             110

16               348               160             110             110

17                 0                30             110               0
```

```
    max_rech_amt_8 date_of_last_rech_6 date_of_last_rech_7  \
7             1580          2014-06-27          2014-07-25   
8               30          2014-06-25          2014-07-31   
13             130          2014-06-20          2014-07-22   
16             130          2014-06-30          2014-07-31   
17              30          2014-06-30          2014-07-30   

   date_of_last_rech_8  last_day_rch_amt_6  last_day_rch_amt_7  \
7           2014-08-26                   0                   0   
8           2014-08-30                  50                   0   
13          2014-08-30                 110                  50   
16          2014-08-14                 100                 100   
17          2014-08-25                  30                   0   

    last_day_rch_amt_8  count_rech_2g_6  count_rech_2g_7  
count_rech_2g_8  \
7                  779              0.0              0.0
0.0   
8                   10              0.0              2.0
3.0   
13                   0              0.0              0.0
3.0   
16                 130              0.0              0.0
0.0   
17                   0              0.0              0.0
0.0   

    count_rech_3g_6  count_rech_3g_7  count_rech_3g_8  vol_2g_mb_6  \
7               0.0              0.0              0.0          0.0   
8               0.0              0.0              0.0          0.0   
13              0.0              0.0              0.0          0.0   
16              0.0              0.0              0.0          0.0   
17              0.0              0.0              0.0          0.0   

    vol_2g_mb_7  vol_2g_mb_8  vol_3g_mb_6  vol_3g_mb_7  vol_3g_mb_8  \
7           0.0         0.00          0.0         0.00         0.00   
8         356.0         0.03          0.0       750.95        11.94   
13          0.0         0.02          0.0         0.00         0.00   
16          0.0         0.00          0.0         0.00         0.00   
17          0.0         0.00          0.0         0.00         0.00   

    monthly_2g_6  monthly_2g_7  monthly_2g_8  sachet_2g_6  sachet_2g_7  
\
7              0             0             0            0            0   

8              0             1             0            0            1   

13             0             0             0            0            0   
```

```
16              0             0             0            0            0

17              0             0             0            0            0

    sachet_2g_8  monthly_3g_6  monthly_3g_7  monthly_3g_8  sachet_3g_6  
\
7             0             0             0             0            0   

8             3             0             0             0            0   

13            3             0             0             0            0   

16            0             0             0             0            0   

17            0             0             0             0            0   

    sachet_3g_7  sachet_3g_8   aon  aug_vbc_3g  jul_vbc_3g  jun_vbc_3g  
\
7             0            0   802       57.74       19.38       18.74   

8             0            0   315       21.03      910.65      122.16   

13            0            0  2607        0.00        0.00        0.00   

16            0            0   511        0.00        2.45       21.89   

17            0            0   667        0.00        0.00        0.00   

    sep_vbc_3g  churn_flag  
7          0.0           1  
8          0.0           0  
13         0.0           0  
16         0.0           0  
17         0.0           0  
```

Feature:

Combine local and std calls to customer care

```
data['total_og_t2c_mou_6'] = data['loc_og_t2c_mou_6'] + 
data['std_og_t2c_mou_6']
data['total_og_t2c_mou_7'] = data['loc_og_t2c_mou_7'] + 
data['std_og_t2c_mou_7']
data['total_og_t2c_mou_8'] = data['loc_og_t2c_mou_8'] + 
data['std_og_t2c_mou_8']
```

```
data = data.drop(['loc_og_t2c_mou_6',
 'loc_og_t2c_mou_7',
 'loc_og_t2c_mou_8',
 'std_og_t2c_mou_6',
 'std_og_t2c_mou_7',
 'std_og_t2c_mou_8'], axis=1)
```

Feature:

Combine all local & std metrics into one

```
loc_metrics = [x[4:] for x in data.columns if 'loc' in x]
std_metrics = [x[4:] for x in data.columns if 'std' in x]
common_metrics = list(set(loc_metrics) & set(std_metrics))
for metric in common_metrics:
    data['l_s_' + metric] = (data['loc_' + metric] + data['std_' +
metric])/2
    data = data.drop(['loc_' + metric, 'std_' + metric], axis=1)
```

Feature:

Average of Month 6 & 7 - All Metrics

```
for x in data.columns:
    if 'date' in str(x):
        data.drop(columns=x,inplace=True)

for x in data.columns:
        if str(x[-1])=='6':
            colName=str(x[0:len(x)-1])+'good'
            data[colName]=(data[x]+data[x[0:len(x)-1]+'7'])/2

for x in data.columns:
        if ((str(x[-1])=='6') or (str(x[-1])=='7')) and '67' not in
str(x) :
            data.drop(columns=x,inplace=True)
```

Feature:

Percentage change of metrics towards the action phase

```
for x in data.columns:
    if str(x[-1])=='8':
        metric = str(x[0:len(x)-2])
        good_phase_metric = metric + '_good'
        action_phase_metric = metric + '_8'
        data[metric + '_perc_change'] = (data[good_phase_metric] -
data[action_phase_metric])/data[good_phase_metric]
        data.loc[data[good_phase_metric]==0,metric + '_perc_change'] =
0
```

Note:

We can drop all columns with only one unique value

```
for column in data.columns:
    if data[column].nunique() == 1:
        data = data.drop(column, axis=1)
```

Outliers Analysis

```
data.describe(percentiles=(0.5,0.75,0.99,1),exclude=['object'])
```

```
       mobile_number         arpu_8   onnet_mou_8  offnet_mou_8  
roam_ic_mou_8  \
count   3.001100e+04  30011.000000  30011.000000  30011.000000  
30011.00000   
mean    7.001223e+09    534.857433    267.600412    375.021691  
13.25627   
std     6.846405e+05    492.259586    466.560947    477.489377  
74.55207   
min     7.000000e+09   -945.808000      0.000000      0.000000  
0.00000   
50%     7.001232e+09    452.091000     99.440000    240.940000  
0.00000   
75%     7.001814e+09    671.150000    297.735000    482.610000  
0.00000   
99%     7.002387e+09   1987.934400   2188.504000   2211.642000  
288.48500   
100%    7.002411e+09  33543.624000  10752.560000  14007.340000  
4169.81000   
max     7.002411e+09  33543.624000  10752.560000  14007.340000  
4169.81000   

       roam_og_mou_8  isd_og_mou_8  spl_og_mou_8   og_others_8  \
count   30011.000000  30011.000000  30011.000000  30011.000000   
mean       21.469272      2.029314      6.885193      0.058360   
std       106.244774     44.794926     22.893414      3.320865   
min         0.000000      0.000000      0.000000     -1.985314   
50%         0.000000      0.000000      0.490000      0.000000   
75%         0.000000      0.000000      6.380000      0.000000   
99%       432.743000     31.240000     74.112000      0.052535   
100%     5337.040000   5681.540000   1390.880000    394.930000   
max      5337.040000   5681.540000   1390.880000    394.930000   

       total_og_mou_8  total_ic_mou_8  spl_ic_mou_8  isd_ic_mou_8  \
count    30011.000000    30011.000000  30011.000000  30011.000000   
mean       623.774684      295.426531      0.027660     11.700835   
std        685.983313      360.343153      0.116574     74.928607   
min          0.000000        0.000000      0.000000      0.000000   
50%        435.330000      193.440000      0.000000      0.000000   
75%        833.100000      380.410000      0.000000      0.000000   
```

```
99%       3251.785000     1746.224000      0.610000    249.888000
100%     14043.060000     5990.710000      6.230000   4100.380000
max      14043.060000     5990.710000      6.230000   4100.380000

        ic_others_8  total_rech_num_8  total_rech_amt_8
max_rech_amt_8  \
count  30011.000000      30011.000000      30011.000000
30011.000000
mean       1.273249         10.225317        613.638799
162.869348
std       12.926832          9.478572        601.821630
172.605809
min        0.000000          0.000000          0.000000
0.000000
50%        0.000000          8.000000        520.000000
130.000000
75%        0.130000         13.000000        790.000000
198.000000
99%       21.728000         46.000000       2341.900000
951.000000
100%    1209.860000        196.000000      45320.000000
4449.000000
max     1209.860000        196.000000      45320.000000
4449.000000

       last_day_rch_amt_8  count_rech_2g_8  count_rech_3g_8
vol_2g_mb_8  \
count        30011.000000     30011.000000     30011.000000
30011.000000
mean            95.653294         0.721669         0.313618
69.209105
std            145.260363         1.870910         1.161561
268.494284
min              0.000000         0.000000         0.000000
0.000000
50%             50.000000         0.000000         0.000000
0.000000
75%            130.000000         1.000000         0.000000
9.620000
99%            619.000000         9.000000         5.000000
1256.619000
100%          4449.000000        44.000000        45.000000
11117.610000
max           4449.000000        44.000000        45.000000
11117.610000

        vol_3g_mb_8  monthly_2g_8   sachet_2g_8  monthly_3g_8
sachet_3g_8  \
count  30011.000000  30011.000000  30011.000000  30011.000000
30011.000000
```

```
mean     269.864111      0.114058      0.607611      0.173203
0.140415
std      859.299266      0.357272      1.844444      0.582932
0.974727
min        0.000000      0.000000      0.000000      0.000000
0.000000
50%        0.000000      0.000000      0.000000      0.000000
0.000000
75%        0.000000      0.000000      0.000000      0.000000
0.000000
99%     3790.385000      2.000000      9.000000      3.000000
3.000000
100%   30036.060000      5.000000     44.000000     16.000000
41.000000
max    30036.060000      5.000000     44.000000     16.000000
41.000000

                aon    aug_vbc_3g    jul_vbc_3g    jun_vbc_3g
sep_vbc_3g  \
count  30011.000000  30011.000000  30011.000000  30011.000000
30011.000000
mean    1264.064776    129.439626    135.127102    121.360548
6.562685
std      975.263117    390.478591    408.024394    389.726031
48.638658
min      180.000000      0.000000      0.000000      0.000000
0.000000
50%      914.000000      0.000000      0.000000      0.000000
0.000000
75%     1924.000000      1.600000      1.990000      0.000000
0.000000
99%     3651.000000   1822.115000   1941.598000   1866.386000
173.662000
100%    4321.000000  12916.220000   9165.600000  11166.210000
2618.570000
max     4321.000000  12916.220000   9165.600000  11166.210000
2618.570000

         churn_flag  total_og_t2c_mou_8  l_s_ic_mou_8
l_s_og_t2t_mou_8  \
count  30011.000000        30011.000000  30011.000000
30011.000000
mean       0.086402            1.712739    141.226284
129.668201
std        0.280961            7.397562    173.216137
231.027026
min        0.000000            0.000000      0.000000
0.000000
50%        0.000000            0.000000     92.605000
45.685000
```

```
75%        0.000000          0.050000    182.307500
141.740000
99%        1.000000         28.871000    833.123500
1083.609500
100%       1.000000        351.830000   2995.350000
5376.280000
max        1.000000        351.830000   2995.350000
5376.280000

       l_s_og_mou_8  l_s_ic_t2f_mou_8  l_s_ic_t2t_mou_8
l_s_og_t2f_mou_8  \
count  30011.000000      30011.000000      30011.000000
30011.000000
mean     307.396382          8.693718         40.494264
4.142021
std      340.942184         24.711588         86.278911
12.255472
min        0.000000          0.000000          0.000000
0.000000
50%      213.160000          1.290000         18.625000
0.205000
75%      410.797500          7.235000         44.515000
3.040000
99%     1609.752500        106.416500        363.049000
54.647000
100%    7020.785000        794.265000       2196.490000
464.245000
max     7020.785000        794.265000       2196.490000
464.245000

       l_s_ic_t2m_mou_8  l_s_og_t2m_mou_8     arpu_good
onnet_mou_good  \
count      30011.000000      30011.000000  30011.000000
30011.000000
mean          92.031641        173.579995    588.209915
300.188833
std          123.793712        233.606622    409.006147
436.981100
min            0.000000          0.000000   -749.783000
0.000000
50%           57.535000        105.885000    485.602500
136.845000
75%          117.485000        222.877500    674.492000
380.202500
99%          569.370000       1069.472500   1867.815750
2053.946500
100%        2880.225000       7001.500000  31438.461000
7331.060000
max         2880.225000       7001.500000  31438.461000
7331.060000
```

```
       offnet_mou_good  roam_ic_mou_good  roam_og_mou_good  
isd_og_mou_good  \
count     30011.000000      30011.000000      30011.000000  
30011.000000   
mean        420.928874         15.467439         25.678826  
2.235913   
std         440.662739         67.369363         94.737246  
44.785169   
min           0.000000          0.000000          0.000000  
0.000000   
50%         299.095000          0.000000          0.000000  
0.000000   
75%         532.092500          3.055000          7.270000  
0.000000   
99%        2145.356000        280.745500        444.284000  
42.724000   
100%       8314.795000       3060.600000       2410.835000  
5695.470000   
max        8314.795000       3060.600000       2410.835000  
5695.470000   

       spl_og_mou_good  og_others_good  total_og_mou_good  
total_ic_mou_good  \
count     30011.000000    30011.000000       30011.000000  
30011.000000   
mean          6.670925        0.372831         697.911136  
311.193772   
std          18.350992        1.846397         610.373239  
345.411427   
min           0.000000       -0.117216           0.000000  
0.000000   
50%           1.730000        0.000000         547.010000  
211.185000   
75%           7.032500        0.000000         896.840000  
395.232500   
99%          65.081500        4.794500        2956.728500  
1686.116000   
100%       1144.500000      185.065000        9347.210000  
6266.145000   
max        1144.500000      185.065000        9347.210000  
6266.145000   

       spl_ic_mou_good  isd_ic_mou_good  ic_others_good  
total_rech_num_good  \
count     30011.000000     30011.000000    30011.000000  
30011.000000   
mean          0.042399        11.758360        1.342866  
12.017394   
std           0.152451        67.199626       13.400827  
```

```
8.729543
min           0.000000         0.000000        0.000000
0.500000
50%           0.000000         0.000000        0.000000
9.500000
75%           0.000000         0.802500        0.270000
15.000000
99%           0.415000       243.307000       20.327500
44.500000
100%         16.610000      3811.385000     1420.040000
155.500000
max          16.610000      3811.385000     1420.040000
155.500000

       total_rech_amt_good  max_rech_amt_good
last_day_rch_amt_good  \
count         30011.000000       30011.000000           30011.000000

mean            696.664356         173.537553             104.886392

std             488.782088         153.504272             115.077568

min             368.500000           9.000000               0.000000

50%             568.500000         124.000000              80.000000

75%             795.500000         200.000000             124.000000

99%            2216.300000         799.500000             550.000000

100%          37762.500000        3299.000000            3100.000000

max           37762.500000        3299.000000            3100.000000


       count_rech_2g_good  count_rech_3g_good  vol_2g_mb_good
vol_3g_mb_good  \
count        30011.000000        30011.000000    30011.000000
30011.000000
mean             0.671887            0.323581       78.515195
268.243209
std              1.695099            1.033207      254.201180
794.962391
min              0.000000            0.000000        0.000000
0.000000
50%              0.000000            0.000000        0.000000
0.000000
75%              0.500000            0.000000       29.330000
99.697500
```

```
99%              8.500000           4.000000     1220.188500
3347.470000
100%            38.500000          27.000000     7939.075000
36667.845000
max             38.500000          27.000000     7939.075000
36667.845000

       monthly_2g_good  sachet_2g_good  monthly_3g_good
sachet_3g_good  \
count     30011.000000    30011.000000     30011.000000
30011.000000
mean          0.128103        0.543784         0.179518
0.144064
std           0.336706        1.668069         0.540083
0.852098
min           0.000000        0.000000         0.000000
0.000000
50%           0.000000        0.000000         0.000000
0.000000
75%           0.000000        0.000000         0.000000
0.000000
99%           1.500000        8.000000         2.500000
3.000000
100%          4.500000       38.000000        11.500000
27.000000
max           4.500000       38.000000        11.500000
27.000000

       total_og_t2c_mou_good  l_s_ic_mou_good  l_s_og_t2f_mou_good  \
count           30011.000000     30011.000000         30011.000000
mean                1.714694       149.025898             4.530401
std                 6.703410       166.803750            13.061159
min                 0.000000         0.000000             0.000000
50%                 0.000000       101.257500             0.590000
75%                 1.040000       189.495000             3.655000
99%                22.605000       810.235000            55.344750
100%              420.575000      3113.980000           750.432500
max               420.575000      3113.980000           750.432500

       l_s_og_mou_good  l_s_ic_t2t_mou_good  l_s_og_t2m_mou_good  \
count     30011.000000         30011.000000         30011.000000
mean        344.312472            43.045869           194.765197
std         303.671453            86.213315           215.632185
min           0.000000             0.000000             0.000000
50%         269.455000            21.805000           132.955000
75%         444.155000            47.056250           248.416250
99%        1464.662000           355.991250          1052.473750
100%       4514.047500          2919.277500          3998.060000
max        4514.047500          2919.277500          3998.060000
```

	l_s_ic_t2m_mou_good	l_s_ic_t2f_mou_good	l_s_og_t2t_mou_good \
count	30011.000000	30011.000000	30011.000000
mean	96.479788	9.493277	145.010266
std	116.788251	25.037982	216.380664
min	0.000000	0.000000	0.000000
50%	63.235000	1.862500	63.690000
75%	120.778750	8.338750	180.698750
99%	564.908000	110.928250	1023.323000
100%	2263.127500	685.240000	3665.525000
max	2263.127500	685.240000	3665.525000

	arpu_perc_change	onnet_mou_perc_change	offnet_mou_perc_change \
count	30011.000000	30011.000000	30011.000000
mean	0.054280	-0.318394	-0.037691
std	1.247809	14.556860	1.963606
min	-159.521376	-2415.000000	-180.866667
50%	0.092445	0.131863	0.112235
75%	0.360677	0.533966	0.448030
99%	1.000000	1.000000	1.000000
100%	32.029066	1.000000	1.000000
max	32.029066	1.000000	1.000000

	roam_ic_mou_perc_change	roam_og_mou_perc_change \
count	30011.000000	30011.000000
mean	-0.560839	-0.560960
std	16.420730	16.048688
min	-1692.860465	-1608.541667
50%	0.000000	0.000000

```
75%                    0.000000                 0.106725
99%                    1.000000                 1.000000
100%                   1.000000                 1.000000
max                    1.000000                 1.000000

       isd_og_mou_perc_change  spl_og_mou_perc_change
og_others_perc_change  \
count            30011.000000            30011.000000
30011.000000
mean                -0.317038               -3.587897
0.241487
std                 22.130220               89.624125
0.533021
min              -3301.000000            -9241.000000             -
28.343173
50%                  0.000000                0.000000
0.000000
75%                  0.000000                0.973571
0.000000
99%                  1.000000                1.000000
1.000000
100%                 1.000000                1.000000
3.851003
max                  1.000000                1.000000
3.851003

       total_og_mou_perc_change  total_ic_mou_perc_change  \
count              30011.000000              30011.000000
mean                  -0.071427                 -0.126384
std                    6.303876                  3.631704
min                -1022.153846               -421.807692
50%                    0.091907                  0.057532
75%                    0.418111                  0.361538
99%                    1.000000                  1.000000
100%                   1.000000                  1.000000
max                    1.000000                  1.000000

       spl_ic_mou_perc_change  isd_ic_mou_perc_change
ic_others_perc_change  \
count            30011.000000             30011.00000
30011.000000
mean                 0.136893                -2.03782              -
1.107639
std                  0.793283               128.71216
19.513004
min                -31.000000            -20467.00000            -
1798.333333
50%                  0.000000                 0.00000
0.000000
75%                  0.000000                 0.00000
```

```
0.303026
99%                1.000000                 1.00000
1.000000
100%               1.000000                 1.00000
1.000000
max                1.000000                 1.00000
1.000000

       total_rech_num_perc_change  total_rech_amt_perc_change  \
count                30011.000000                30011.000000
mean                     0.132477                    0.101014
std                      0.489753                    0.561578
min                     -7.250000                  -11.937143
50%                      0.185185                    0.119871
75%                      0.454545                    0.426385
99%                      1.000000                    1.000000
100%                     1.000000                    1.000000
max                      1.000000                    1.000000

       max_rech_amt_perc_change  last_day_rch_amt_perc_change  \
count              30011.000000                  30011.000000
mean                  -0.007710                     -0.120641
std                    0.579296                      2.195628
min                  -10.976048                   -187.514286
50%                    0.000000                      0.000000
75%                    0.272727                      0.800000
99%                    1.000000                      1.000000
100%                   1.000000                      1.000000
max                    1.000000                      1.000000

       count_rech_2g_perc_change  count_rech_3g_perc_change  \
count               30011.000000               30011.000000
mean                   -0.011431                   0.038128
std                     0.936077                   0.568915
min                   -23.000000                 -21.000000
50%                     0.000000                   0.000000
75%                     0.000000                   0.000000
99%                     1.000000                   1.000000
100%                    1.000000                   1.000000
max                     1.000000                   1.000000

       vol_2g_mb_perc_change  vol_3g_mb_perc_change
monthly_2g_perc_change  \
count           30011.000000           30011.000000
30011.000000
mean              -12.768452              -4.022263
0.044191
std               674.767276             281.026005
0.357355
min            -67949.000000          -41197.000000               -
```

```
7.000000
50%                0.000000               0.000000
0.000000
75%                0.151052               0.000000
0.000000
99%                1.000000               1.000000
1.000000
100%               1.000000               1.000000
1.000000
max                1.000000               1.000000
1.000000

       sachet_2g_perc_change  monthly_3g_perc_change
sachet_3g_perc_change  \
count           30011.000000            30011.000000
30011.000000
mean               -0.004292                0.029569
0.036178
std                 0.854959                0.421215
0.470230
min               -21.000000              -13.000000             -
25.000000
50%                 0.000000                0.000000
0.000000
75%                 0.000000                0.000000
0.000000
99%                 1.000000                1.000000
1.000000
100%                1.000000                1.000000
1.000000
max                 1.000000                1.000000
1.000000

       total_og_t2c_mou_perc_change  l_s_ic_mou_perc_change  \
count                  30011.000000            30011.000000
mean                      -1.609834               -0.180846
std                       60.379931               11.332222
min                    -8361.000000            -1859.666667
50%                        0.000000                0.060770
75%                        0.869018                0.365278
99%                        1.000000                1.000000
100%                       1.000000                1.000000
max                        1.000000                1.000000

       l_s_og_t2t_mou_perc_change  l_s_og_mou_perc_change  \
count                30011.000000            30011.000000
mean                    -0.375677               -0.082759
std                     14.707390                6.347731
min                  -2415.000000            -1022.153846
50%                      0.124609                0.091683
```

```
75%                          0.544132                0.420734
99%                          1.000000                1.000000
100%                         1.000000                1.000000
max                          1.000000                1.000000

       l_s_ic_t2f_mou_perc_change  l_s_ic_t2t_mou_perc_change  \
count                30011.000000                30011.000000
mean                    -1.081201                   -0.552800
std                     22.376735                   19.118517
min                  -2648.666667                -2585.000000
50%                      0.035122                    0.106480
75%                      0.759021                    0.524674
99%                      1.000000                    1.000000
100%                     1.000000                    1.000000
max                      1.000000                    1.000000

       l_s_og_t2f_mou_perc_change  l_s_ic_t2m_mou_perc_change  \
count                30011.000000                30011.000000
mean                    -0.696418                   -0.242943
std                     10.282067                    8.598364
min                   -691.000000                -1244.333333
50%                      0.000000                    0.062092
75%                      0.777778                    0.400177
99%                      1.000000                    1.000000
100%                     1.000000                    1.000000
max                      1.000000                    1.000000

       l_s_og_t2m_mou_perc_change
count                30011.000000
mean                    -0.190344
std                      7.438990
min                   -770.000000
50%                      0.108718
75%                      0.472157
99%                      1.000000
100%                     1.000000
max                      1.000000
```

```
rem_col=[
        'mobile_number','circle_id','date_of_last_rech_6',
        'date_of_last_rech_7','date_of_last_rech_8','churn_flag'
        ]
outlier_col_check=[x for x in data.columns if x not in rem_col]

outlier_df=create_outlier_df(data,outlier_col_check)
display(outlier_df.sort_values('Outlier%'))
print("Number of columns with more than 5% outliers
",len(outlier_df[outlier_df['Outlier%']>5]))
print("Number of columns with more than 10% outliers
",len(outlier_df[outlier_df['Outlier%']>10]))
```

```
print("Number of columns with more than 20% outliers
",len(outlier_df[outlier_df['Outlier%']>20]))
```

```
                  ColumnName OutlierCount  Outlier%
25                       aon           28      0.09
87  total_rech_amt_perc_change        795      2.65
86  total_rech_num_perc_change        908      3.03
73            arpu_perc_change       1004      3.35
7                   og_others_8       1118      3.73
..                         ...          ...       ...
85       ic_others_perc_change       9074     30.24
77     roam_og_mou_perc_change       9276     30.91
84      isd_ic_mou_perc_change       9552     31.83
76     roam_ic_mou_perc_change       9656     32.17
92       vol_2g_mb_perc_change      10191     33.96

[107 rows x 3 columns]

Number of columns with more than 5% outliers  98
Number of columns with more than 10% outliers  54
Number of columns with more than 20% outliers  25
```

- As high number of columns are having outliers, removing them will reduce the data significantly (due to union of outlier cell rows), should be avoided

- Transforming the outlier with the mean will also not be favourable as columns exists with high percentage of outliers, that would have skewed the mean

- Capping the outliers seems the best option here to avoid incorrect deductions

```
#Capping outliers to the IQR limit
data=cap_outlier(data,outlier_col_check)
#Create dataframe of outlier count to verify
outlier_df=create_outlier_df(data,outlier_col_check)
```

```
data.shape
```

```
(30011, 109)
```

```
outlier_df['Outlier%'].value_counts()
```

```
0.0    107
Name: Outlier%, dtype: int64
```

Dropping columns:

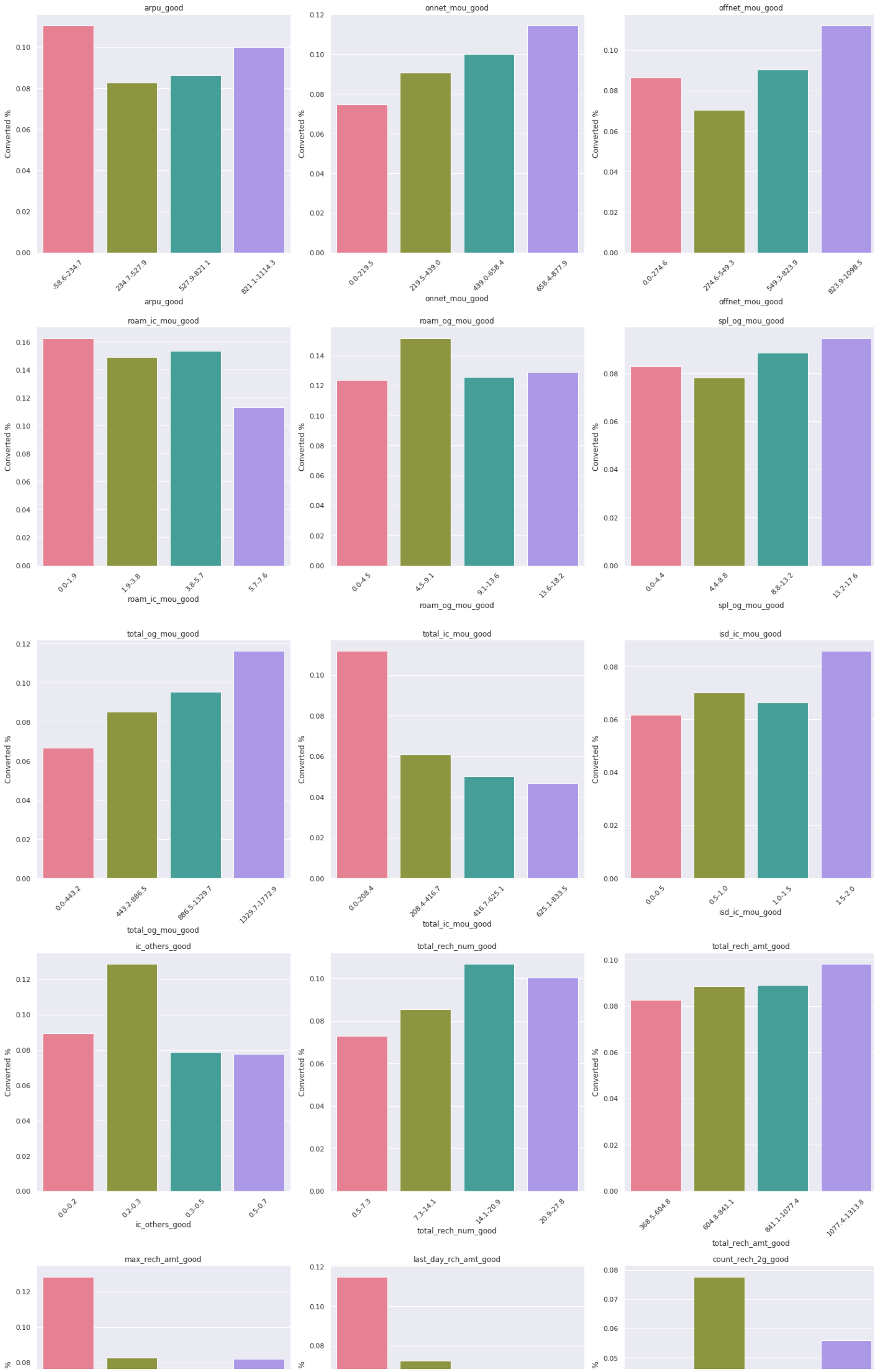
```
mobile_number: because it is identifier
```

```
data.drop(columns=['mobile_number'],inplace=True)
```

EDA

EDA Can be performed on good phase, action phase & 8th month data to remove variables which clearly do not seem to impact the churn ratio.

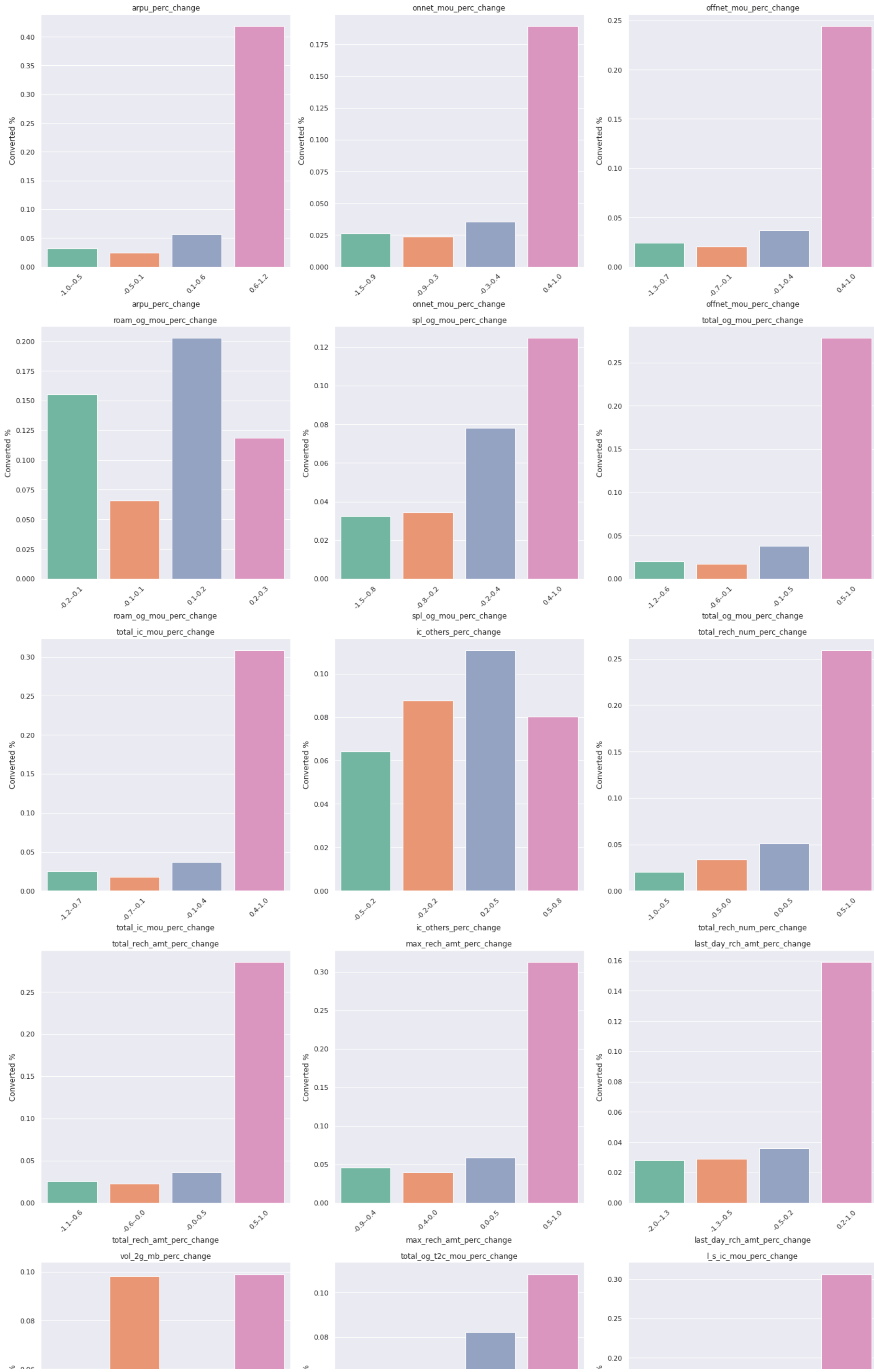
```
ratio_plots_num(data, [x for x in data.columns if '_good' in x],
'churn_flag', 4, 'husl')
```


arpu_good
Converted %
-58.6-234.7
234.7-527.9
527.9-821.1
821.1-1114.3
arpu_good
onnet_mou_good
Converted %
0.0-219.5
219.5-439.0
439.0-658.4
658.4-877.9
onnet_mou_good
offnet_mou_good
Converted %
0.0-274.6
274.6-549.3
549.3-823.9
823.9-1098.5
offnet_mou_good
roam_ic_mou_good
Converted %
0.0-1.9
1.9-3.8
3.8-5.7
5.7-7.6
roam_ic_mou_good
roam_og_mou_good
Converted %
0.0-4.5
4.5-9.1
9.1-13.6
13.6-18.2
roam_og_mou_good
spl_og_mou_good
Converted %
0.0-4.4
4.4-8.8
8.8-13.2
13.2-17.6
spl_og_mou_good
total_og_mou_good
Converted %
0.0-443.2
443.2-886.5
886.5-1329.7
1329.7-1772.9
total_og_mou_good
total_ic_mou_good
Converted %
0.0-208.4
208.4-416.7
416.7-625.1
625.1-833.5
total_ic_mou_good
isd_ic_mou_good
Converted %
0.0-0.5
0.5-1.0
1.0-1.5
1.5-2.0
isd_ic_mou_good
ic_others_good
Converted %
0.0-0.2
0.2-0.3
0.3-0.5
0.5-0.7
ic_others_good
total_rech_num_good
Converted %
0.5-7.3
7.3-14.1
14.1-20.9
20.9-27.8
total_rech_num_good
total_rech_amt_good
Converted %
368.5-604.8
604.8-841.1
841.1-1077.4
1077.4-1313.8
total_rech_amt_good
max_rech_amt_good
last_day_rch_amt_good
count_rech_2g_good

Observation:

Some of the good phase columns do not depict any difference in the churn ratios and we can hence drop them.

```
insignificant_columns = 
['roam_og_mou_good','spl_og_mou_good','total_rech_amt_good','vol_2g_mb_good',
                        'total_og_t2c_mou_good']

data = data.drop(insignificant_columns, axis=1)

ratio_plots_num(data, [x for x in data.columns if '_perc_change' in x], 'churn_flag', 4, 'Set2')
```

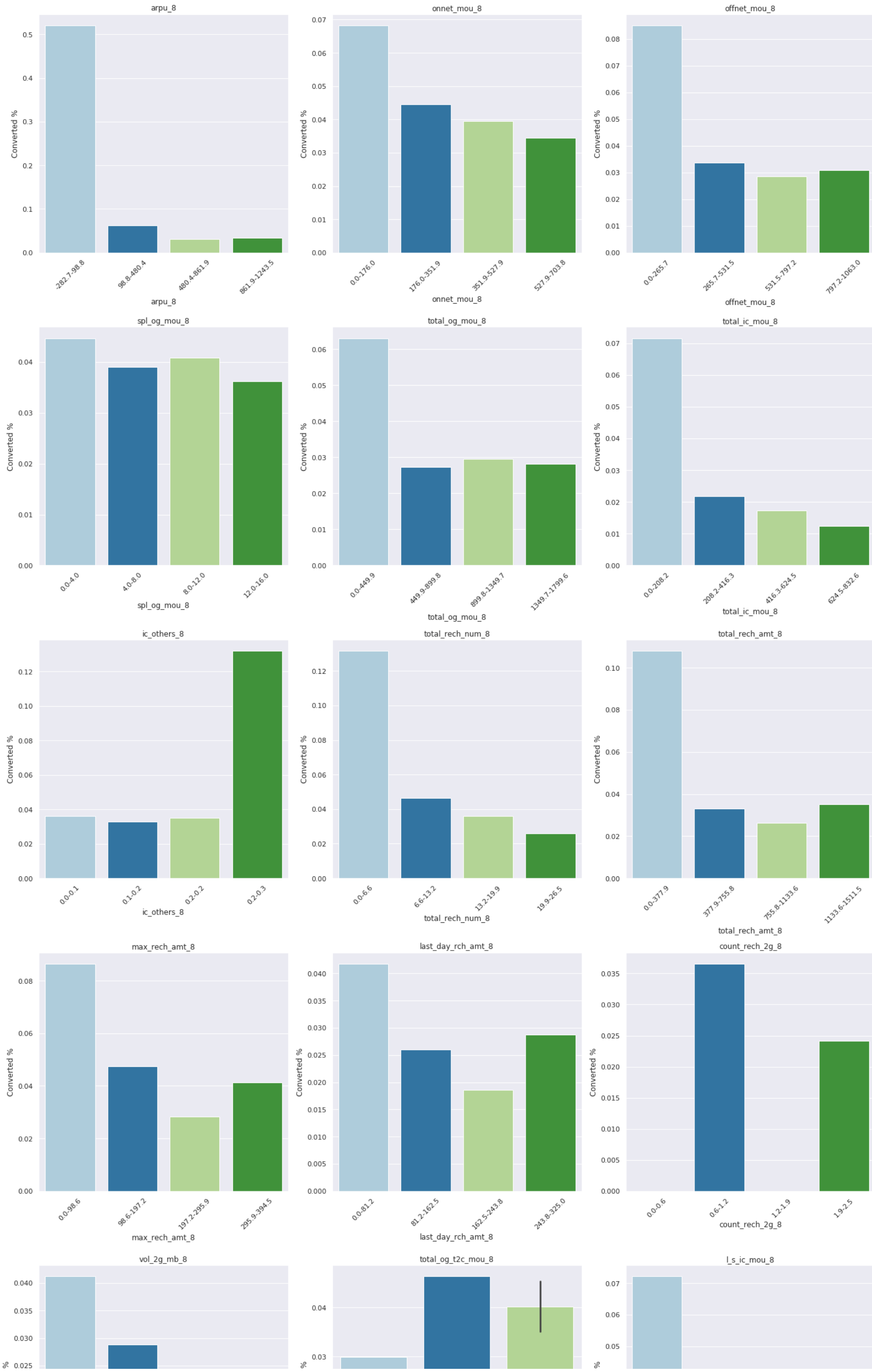


arpu_perc_change
Converted %
-1.0--0.5
-0.5-0.1
0.1-0.6
0.6-1.2
arpu_perc_change
onnet_mou_perc_change
Converted %
-1.5--0.9
-0.9--0.3
-0.3-0.4
0.4-1.0
onnet_mou_perc_change
offnet_mou_perc_change
Converted %
-1.3--0.7
-0.7--0.1
-0.1-0.4
0.4-1.0
offnet_mou_perc_change
roam_og_mou_perc_change
Converted %
-0.2--0.1
-0.1-0.1
0.1-0.2
0.2-0.3
roam_og_mou_perc_change
spl_og_mou_perc_change
Converted %
-1.5--0.8
-0.8--0.2
-0.2-0.4
0.4-1.0
spl_og_mou_perc_change
total_og_mou_perc_change
Converted %
-1.2--0.6
-0.6--0.1
-0.1-0.5
0.5-1.0
total_og_mou_perc_change
total_ic_mou_perc_change
Converted %
-1.2--0.7
-0.7--0.1
-0.1-0.4
0.4-1.0
total_ic_mou_perc_change
ic_others_perc_change
Converted %
-0.5--0.2
-0.2-0.2
0.2-0.5
0.5-0.8
ic_others_perc_change
total_rech_num_perc_change
Converted %
-1.0--0.5
-0.5-0.0
0.0-0.5
0.5-1.0
total_rech_num_perc_change
total_rech_amt_perc_change
Converted %
-1.1--0.6
-0.6--0.0
-0.0-0.5
0.5-1.0
total_rech_amt_perc_change
max_rech_amt_perc_change
Converted %
-0.9--0.4
-0.4-0.0
0.0-0.5
0.5-1.0
max_rech_amt_perc_change
last_day_rch_amt_perc_change
Converted %
-2.0--1.3
-1.3--0.5
-0.5-0.2
0.2-1.0
last_day_rch_amt_perc_change
vol_2g_mb_perc_change
total_og_t2c_mou_perc_change
l_s_ic_mou_perc_change

Observation:

All of the perc_change columns seem to demarcate the churn ratio well, so we do not have to drop any of them.

```
ratio_plots_num(data, [x for x in data.columns if '_8' in x],
'churn_flag', 4, 'Paired')
```



arpu_8
Converted %
-282.7-98.8
98.8-480.4
480.4-861.9
861.9-1243.5
arpu_8
onnet_mou_8
Converted %
0.0-176.0
176.0-351.9
351.9-527.9
527.9-703.8
onnet_mou_8
offnet_mou_8
Converted %
0.0-265.7
265.7-531.5
531.5-797.2
797.2-1063.0
offnet_mou_8
spl_og_mou_8
Converted %
0.0-4.0
4.0-8.0
8.0-12.0
12.0-16.0
spl_og_mou_8
total_og_mou_8
Converted %
0.0-449.9
449.9-899.8
899.8-1349.7
1349.7-1799.6
total_og_mou_8
total_ic_mou_8
Converted %
0.0-208.2
208.2-416.3
416.3-624.5
624.5-832.6
total_ic_mou_8
ic_others_8
Converted %
0.0-0.1
0.1-0.2
0.2-0.2
0.2-0.3
ic_others_8
total_rech_num_8
Converted %
0.0-6.6
6.6-13.2
13.2-19.9
19.9-26.5
total_rech_num_8
total_rech_amt_8
Converted %
0.0-377.9
377.9-755.8
755.8-1133.6
1133.6-1511.5
total_rech_amt_8
max_rech_amt_8
Converted %
0.0-98.6
98.6-197.2
197.2-295.9
295.9-394.5
max_rech_amt_8
last_day_rch_amt_8
Converted %
0.0-81.2
81.2-162.5
162.5-243.8
243.8-325.0
last_day_rch_amt_8
count_rech_2g_8
Converted %
0.0-0.6
0.6-1.2
1.2-1.9
1.9-2.5
count_rech_2g_8
vol_2g_mb_8
total_og_t2c_mou_8
l_s_ic_mou_8

Observation:

One of the action phase columns do not depict any difference in the churn ratios and we can hence drop it.

```
insignificant_columns = ['spl_og_mou_8']

data = data.drop(insignificant_columns, axis=1)
```

Correlation

To remove interdependent features and avoid redundancy to increase model efficiency. As the number of cokumns is high , creating a heatmap will be tricy to analyse same goes with the corr() Dataframe, hence creating a function that takes input - the dataframe , the column list, threshold. Returns a dataframe with unique column pairs and their corr value (>= threshold)

```
corr_col=list(data.columns)

# Finding the correlation of features > 0.9
corr_df=create_correlation_df(data,corr_col,0.9)
corr_df.sort_values(by=('Correlation
Value'),ascending=False,inplace=True)
corr_df
```

```
                     Feature 1                 Feature 2  Correlation
Value
3                 l_s_og_mou_8            total_og_mou_8
1.00
7              l_s_og_mou_good         total_og_mou_good
1.00
2             l_s_og_t2t_mou_8               onnet_mou_8
0.99
9          l_s_og_t2t_mou_good            onnet_mou_good
0.99
12      l_s_og_mou_perc_change  total_og_mou_perc_change
0.99
1                 l_s_ic_mou_8            total_ic_mou_8
0.98
5             l_s_og_t2m_mou_8              offnet_mou_8
0.98
6              l_s_ic_mou_good         total_ic_mou_good
0.98
8          l_s_og_t2m_mou_good           offnet_mou_good
0.98
10      l_s_ic_mou_perc_change  total_ic_mou_perc_change
0.98
11  l_s_og_t2t_mou_perc_change     onnet_mou_perc_change
0.95
```

```
0            total_rech_amt_8                  arpu_8
0.94
13  l_s_og_t2m_mou_perc_change    offnet_mou_perc_change
0.93
4             l_s_ic_t2m_mou_8              l_s_ic_mou_8
0.91
```

The above Dataframe gives us the Features highly correlated to each other. We can remove some of the highly correlated variables and retain the rest, which can be removed after VIF checks.

```
correlated_column_list = ['l_s_og_mou_8', 'l_s_og_mou_good',
'l_s_og_t2t_mou_8',
                           'l_s_og_t2t_mou_good',
'l_s_og_mou_perc_change',
                           'l_s_og_t2m_mou_8', 'l_s_ic_mou_8',
'l_s_og_t2m_mou_good',
                           'l_s_ic_mou_good',
'l_s_ic_mou_perc_change',
                           'l_s_og_t2t_mou_perc_change',
'total_rech_amt_8',
                           'l_s_og_t2m_mou_perc_change',
'l_s_ic_mou_8']
```

```
data.drop(columns=correlated_column_list,inplace=True)
```

```
data.shape
```

```
(30011, 89)
```

Class Imbalance

```
print('Churn flag 1 count: ',len(data[data['churn_flag']==1]))
print('Churn flag 0 count: ',len(data[data['churn_flag']==0]))
ClassImbRatio=len(data[data['churn_flag']==0])/len(data[data['churn_fl
ag']==1])

print('Class imbalance ratio: ',round(ClassImbRatio,2))
```

```
Churn flag 1 count:  2593
Churn flag 0 count:  27418
Class imbalance ratio:  10.57
```

The class imbalance ratio is high ~10.6, due to this the predictions will be biased towards the majority class. This will result in a lower accuracy model when compared to a model trained on a balanced data set.

We can use either over sampling or under sampling for the data:

- Undersampling will reduce the major class data points to match the minority class data points count.This will result is reduction of overall data which is not recommended.

- Oversampling will match the minority class data points count to match the majority class data points count (Recommended)

We would be using SMOTE from imblearn.over_sampling to do the same, on the training data set only.

```
data.shape
```

```
(30011, 89)
```

```
#Splitting the data into X and y (target variable)
data_temp=data.copy()
y=data_temp.pop('churn_flag')
X=data_temp

# Splitting into train and test
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,rando
m_state=100)

# Implementing SMOTE
smote=SMOTE(sampling_strategy=0.6, random_state=42, k_neighbors=3)

X_train_smote,y_train_smote=smote.fit_sample(X_train,y_train)

print('Before SMOTE',Counter(y_train))
print('After SMOTE',Counter(y_train_smote))
```

```
Before SMOTE Counter({0: 19184, 1: 1823})
After SMOTE Counter({0: 19184, 1: 11510})
```

Now we have the class balanced for the training data

Feature Scaling

```
X_train_smote_original = X_train_smote.copy()
X_test_original = X_test.copy()

scaler = preprocessing.StandardScaler()
X_train_smote[X_train_smote.columns] =
scaler.fit_transform(X_train_smote)
X_test[X_train_smote.columns] =
scaler.transform(X_test[X_train_smote.columns])
```

Eliminate Insignificant Variables Using VIF

Though we will be using PCA to extract the features for building the model, the objective still remains to be able to extract the important features impacting the churn. We will thus work towards this by eliminating the redundant variables before application of PCA.

```
features_set_1 = X_train_smote.columns

vif_ranks(X_train_smote, features_set_1, 10)
```

```
                Features    VIF
40    total_ic_mou_good  53.19
8        total_ic_mou_8  52.05
33            arpu_good  50.26
0                arpu_8  47.41
39    total_og_mou_good  44.47
56  l_s_ic_t2m_mou_good  36.59
32     l_s_ic_t2m_mou_8  36.58
7        total_og_mou_8  34.01
44  total_rech_num_good  27.99
13        max_rech_amt_8  27.62
```

Note:

RFE can be used to eliminate variables with high VIF

```
lr = LogisticRegression()

rfe = RFE(lr, 25)
rfe = rfe.fit(X_train_smote[features_set_1], y_train_smote)

features_set_2 = list(data_temp.columns[rfe.support_])

X_train_smote = X_train_smote[features_set_2]
X_test = X_test[features_set_2]

X_train_smote_original = X_train_smote_original[features_set_2]
X_test_original = X_test_original[features_set_2]

features_set_2
```

```
['arpu_8',
 'total_og_mou_8',
 'ic_others_8',
 'total_rech_num_8',
 'max_rech_amt_8',
 'last_day_rch_amt_8',
 'vol_2g_mb_8',
 'aon',
 'l_s_ic_t2f_mou_8',
 'l_s_ic_t2t_mou_8',
 'l_s_og_t2f_mou_8',
 'l_s_ic_t2m_mou_8',
 'arpu_good',
 'roam_ic_mou_good',
 'total_ic_mou_good',
 'last_day_rch_amt_good',
 'count_rech_2g_good',
 'l_s_ic_t2m_mou_good',
 'arpu_perc_change',
 'total_og_mou_perc_change',
```

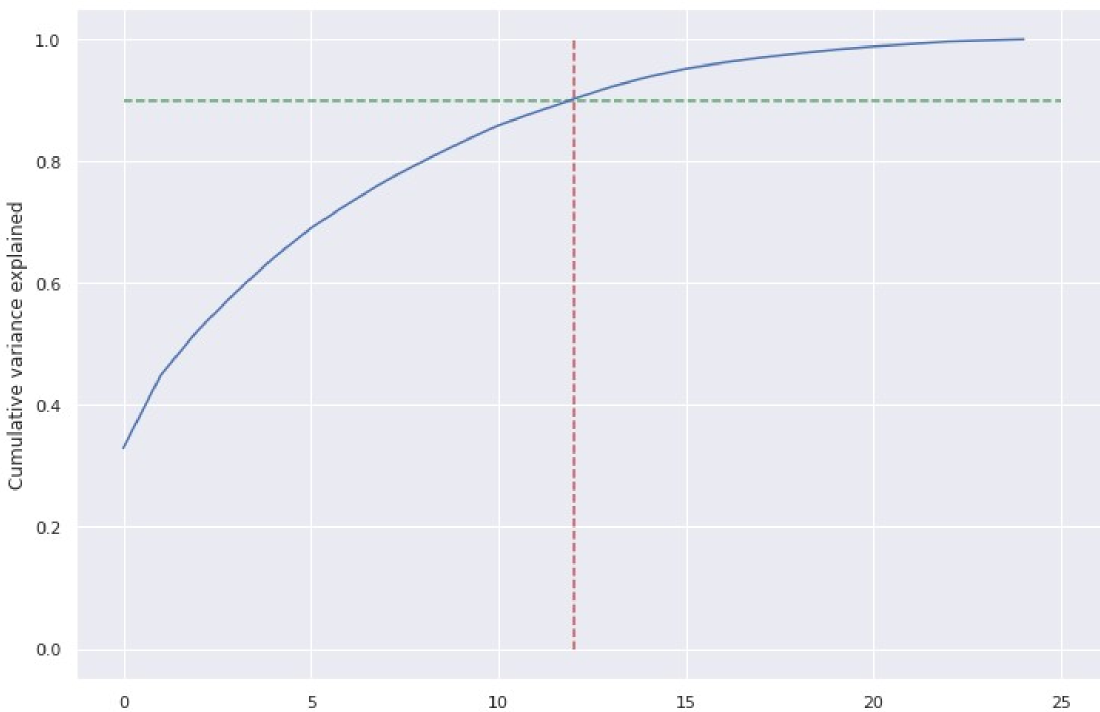
```
 'total_ic_mou_perc_change',
 'total_rech_amt_perc_change',
 'max_rech_amt_perc_change',
 'last_day_rch_amt_perc_change',
 'l_s_ic_t2m_mou_perc_change']
```

PCA

```
pca_iteration = PCA(random_state=0)
pca_iteration.fit(X_train_smote)
```

```
PCA(copy=True, iterated_power='auto', n_components=None,
random_state=0,
    svd_solver='auto', tol=0.0, whiten=False)
```

```
var_cumu = numpy.cumsum(pca_iteration.explained_variance_ratio_)
fig = plt.figure(figsize=[12,8])
plt.vlines(x=12, ymax=1, ymin=0, colors="r", linestyles="--")
plt.hlines(y=0.90, xmax=25, xmin=0, colors="g", linestyles="--")
plt.plot(var_cumu)
plt.ylabel("Cumulative variance explained")
plt.show()
```



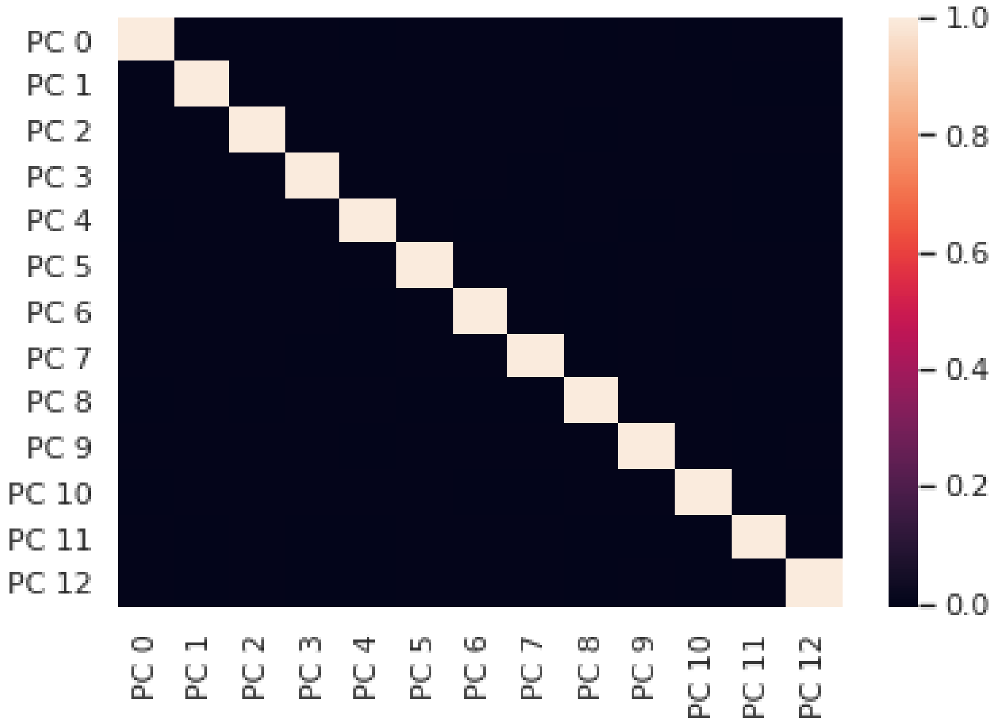

```
pca_iteration_2 = IncrementalPCA(n_components=13)
X_train_smote_transformed =
pca_iteration_2.fit_transform(X_train_smote)
X_test_transformed = pca_iteration_2.transform(X_test)
```

```
X_train_smote_transformed =
pandas.DataFrame(X_train_smote_transformed)
```

```
X_train_smote_transformed.columns = ['PC '+str(x) for x in
X_train_smote_transformed]

X_test_transformed = pandas.DataFrame(X_test_transformed)
X_test_transformed.columns = ['PC '+str(x) for x in
X_test_transformed]
```

```
sns.heatmap(X_train_smote_transformed.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93595c8dd0>
```



Observation:

PCA has been used to select only the top features contributing to the 95% of the variation & we can verify through the correlation plot that there absolutely exists no correlation amongst the principal components.

Model Building - Features: PCA

Logistic Regression

Iteration 1

```
pc_features_set_1 = X_train_smote_transformed.columns

features = pc_features_set_1
X_train_sm = sm.add_constant(X_train_smote_transformed[features])
```

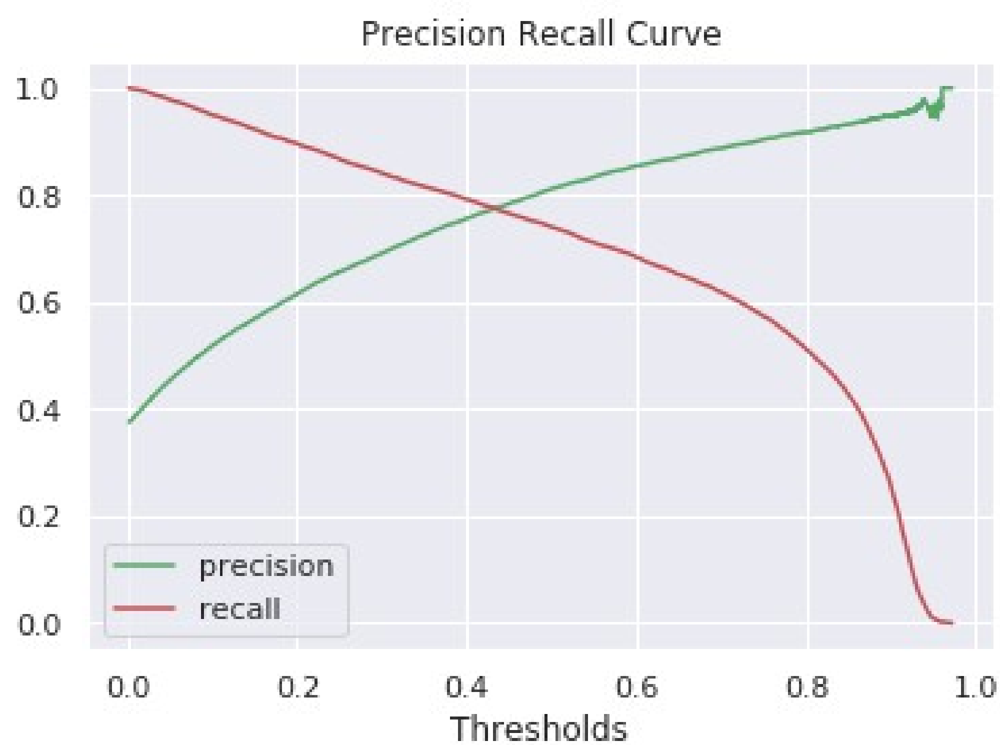
```
X_test_sm = sm.add_constant(X_test_transformed[features])
logm2 = sm.GLM(y_train_smote, X_train_sm, family =
sm.families.Binomial())
model = logm2.fit()
model.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results

======================================================================
========
Dep. Variable:            churn_flag   No. Observations:
30694
Model:                           GLM   Df Residuals:
30680
Model Family:               Binomial   Df Model:
13
Link Function:                 logit   Scale:
1.0000
Method:                         IRLS   Log-Likelihood:
-11987.
Date:             Thu, 21 May 2020   Deviance:
23973.
Time:                       22:32:09   Pearson chi2:
3.73e+04
No. Iterations:                    6

Covariance Type:           nonrobust

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
const         -0.9209      0.018    -50.532      0.000      -0.957
-0.885
PC 0           0.7125      0.008     91.004      0.000       0.697
0.728
PC 1          -0.0434      0.011     -3.922      0.000      -0.065
-0.022
PC 2           0.0497      0.013      3.876      0.000       0.025
0.075
PC 3           0.4972      0.014     34.843      0.000       0.469
0.525
PC 4          -0.0808      0.015     -5.419      0.000      -0.110
-0.052
PC 5           0.0372      0.016      2.395      0.017       0.007
0.068
```

```
PC 6            -0.0107      0.017      -0.647      0.518      -0.043
0.022
PC 7             0.0115      0.017       0.672      0.501      -0.022
0.045
PC 8             0.0364      0.020       1.825      0.068      -0.003
0.075
PC 9            -0.2280      0.020     -11.474      0.000      -0.267
-0.189
PC 10           -0.3098      0.021     -14.723      0.000      -0.351
-0.269
PC 11            0.1160      0.025       4.720      0.000       0.068
0.164
PC 12           -0.0094      0.025      -0.379      0.705      -0.058
0.039
======================================================================
========
"""
```

Tuning The Probability Theshold

```
plot_precision_recall_curve(model.predict(X_train_sm), y_train_smote)
```

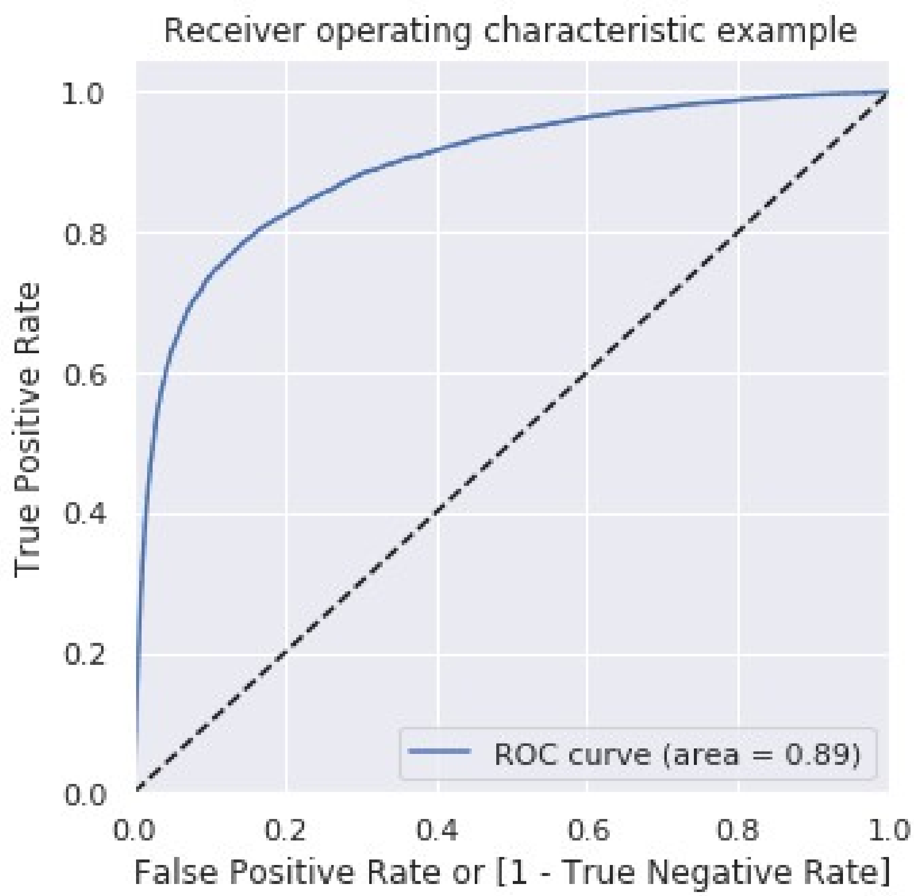


Observation:

From this, it appears that the optimal threshold is 0.45

```
predict_summarize(model.predict(X_train_sm), y_train_smote, 0.45,
True)
```

```
Accuracy =  0.8331921548185313
Sensitivity =  0.7662033014769766
Specificity =  0.873384070058382
False Positive Rate =  0.12661592994161802

Precision =  0.7840504978662873
Recall =  0.7662033014769766
Plotting
```

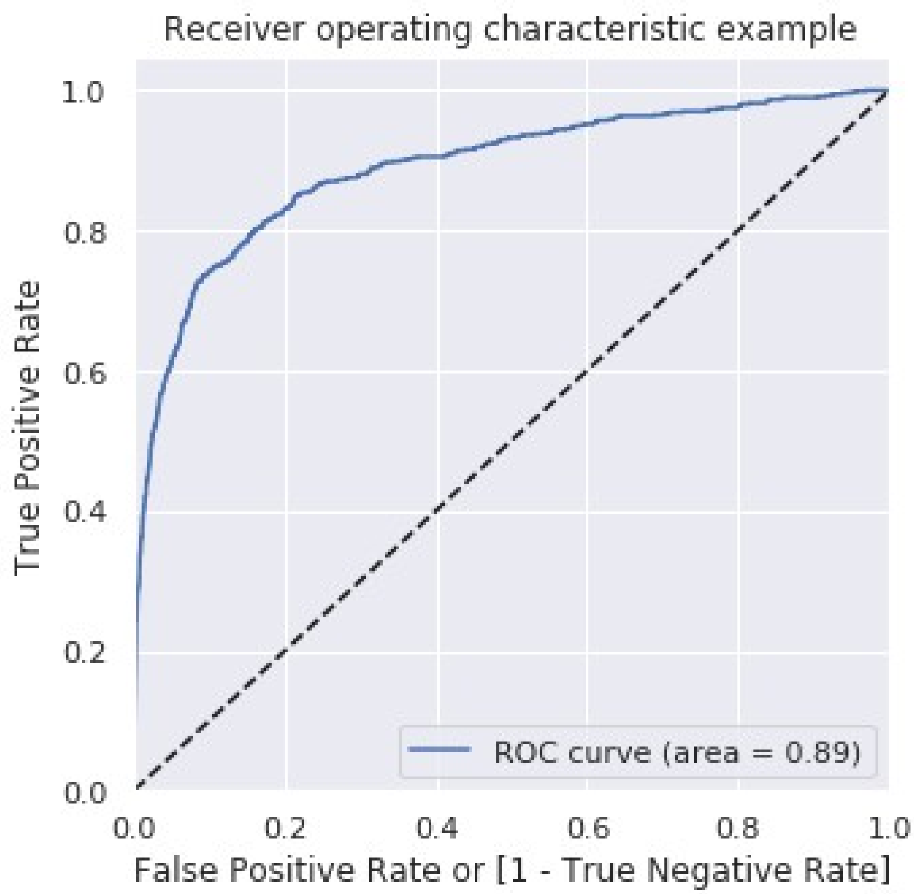



```
            predicted_no  predicted_yes
ind
actual_no          16755           2429
actual_yes          2691           8819
```

```
predict_summarize(model.predict(X_test_sm), y_test.values, 0.45, True)
```

```
Accuracy =  0.8668369613505109
Sensitivity =  0.7558441558441559
Specificity =  0.8772164197230994
False Positive Rate =  0.12278358027690066

Precision =  0.3653483992467043
```

```
Recall =  0.7558441558441559
Plotting
```




```
            predicted_no  predicted_yes
ind
actual_no           7223           1011
actual_yes           188            582
```

Random Forest

Apart from creating features, we will have to further tune the model by arriving at the best hyper parameters either by trial & erro or by grid search.

Choosing multiple values of parameters to avoid over/under fitting of the RF Model. Using 3 folds CV we have come accross the best suited param set for RF model :

'bootstrap': True,To reduce latency

'criterion': gini & entropy are the 2 possible criterion, and considering use cases where there is a class imbalance, entropy proves to be a better fit.

'max_depth': Having higher number of tress will make the model learn the data to a level of overfitting. And will perform poorly in real time scenario. Hence restricting to [5,10,15,20,25]

'min_impurity_decrease': This parameter ensures that the tree does grow more without a substantial increase in the purity.

'min_samples_split': Less min_samples_split would have caused the tree to increase and over fit the data.

'n_estimators': To avoid over fitting, more the estimators, better is the model, but unnecessary increase in this also slows down the model build process with no improvement in the accuracy.

'oob_score': Out of bag scoring will ensure that the model works well even on unseen data.

```
features = [x for x in pc_features_set_1 if x not in ['PC 6','PC 7','PC 8','PC 12']]
X_train_forest = X_train_smote_transformed[features]
X_test_forest = X_test_transformed[features]

n_estimators = [500,800,1000]
max_features = ['auto','sqrt']
criterion = ["gini", "entropy"]
max_depth = [5,10,15,20,25]
min_samples_split = [50, 100]
min_impurity_decrease = [0.1, 0.2]
param_grid = {'n_estimators': n_estimators,
                'max_features': max_features,
                'max_depth': max_depth,
                'min_samples_split': min_samples_split,
                'min_impurity_decrease':min_impurity_decrease,
                'criterion':criterion,
                'bootstrap':[True],
                'oob_score':[True]}

grid_search = GridSearchCV(estimator = RandomForestClassifier(),
                           param_grid = param_grid,
                        cv = 3, n_jobs = 8, verbose = 2)
grid_search.fit(X_train_forest, y_train_smote)
grid_search
```

```
Fitting 3 folds for each of 240 candidates, totalling 720 fits

[Parallel(n_jobs=8)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  25 tasks      | elapsed:   49.6s
[Parallel(n_jobs=8)]: Done 146 tasks      | elapsed:  4.6min
[Parallel(n_jobs=8)]: Done 349 tasks      | elapsed: 11.2min
[Parallel(n_jobs=8)]: Done 632 tasks      | elapsed: 27.7min
[Parallel(n_jobs=8)]: Done 720 out of 720 | elapsed: 33.1min finished
```

```
GridSearchCV(cv=3, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True,
ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini',
max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,

min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,

min_weight_fraction_leaf=0.0,
                                              n_estimators=100,
n_jobs=None,...
                                              warm_start=False),
             iid='deprecated', n_jobs=8,
             param_grid={'bootstrap': [True], 'criterion': ['gini',
'entropy'],
                         'max_depth': [5, 10, 15, 20, 25],
                         'max_features': ['auto', 'sqrt'],
                         'min_impurity_decrease': [0.1, 0.2],
                         'min_samples_split': [50, 100],
                         'n_estimators': [500, 800, 1000],
                         'oob_score': [True]},
             pre_dispatch='2*n_jobs', refit=True,
return_train_score=False,
             scoring=None, verbose=2)
```

```
grid_search.best_params_
```

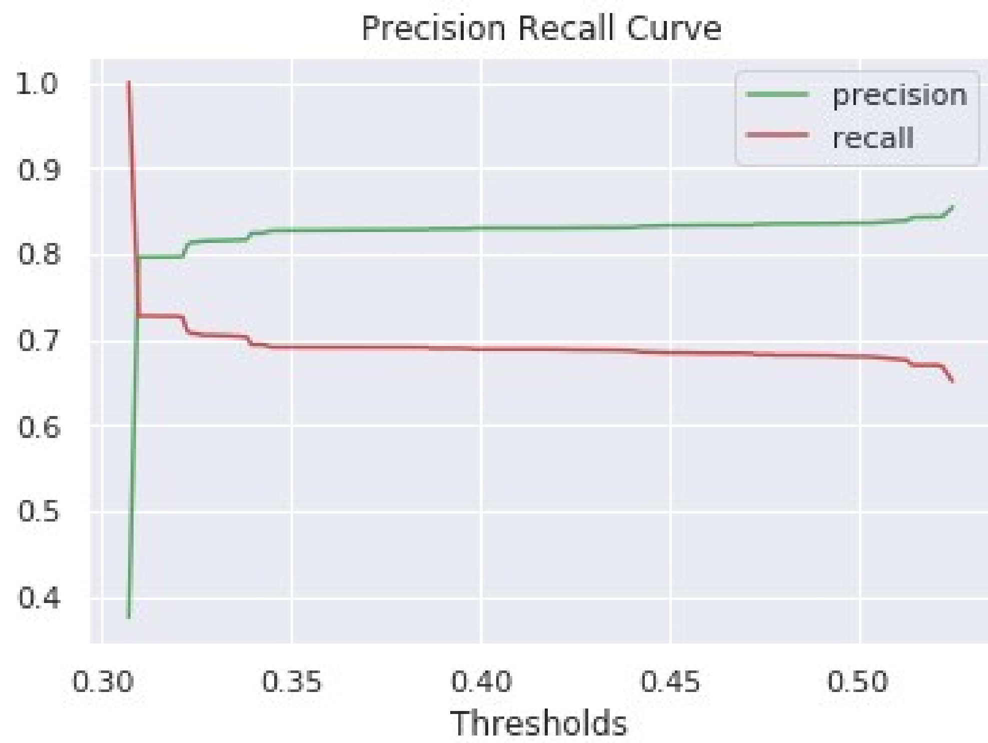
```
{'bootstrap': True,
 'criterion': 'entropy',
 'max_depth': 15,
 'max_features': 'sqrt',
 'min_impurity_decrease': 0.2,
 'min_samples_split': 50,
 'n_estimators': 500,
 'oob_score': True}
```

```
model_rf = RandomForestClassifier(bootstrap=True,
                                  criterion = 'entropy',
                                  max_depth=15,
                                  max_features='sqrt',
                                  min_samples_split=50,
                                  n_estimators=500,
                                  min_impurity_decrease=0.2,
                                  random_state = 42,
```

```
                                  oob_score=True
                                 )
model_rf.fit(X_train_forest, y_train_smote)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight=None,
                       criterion='entropy', max_depth=15,
max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.2,
min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=50,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=True, random_state=42,
verbose=0,
                       warm_start=False)
```

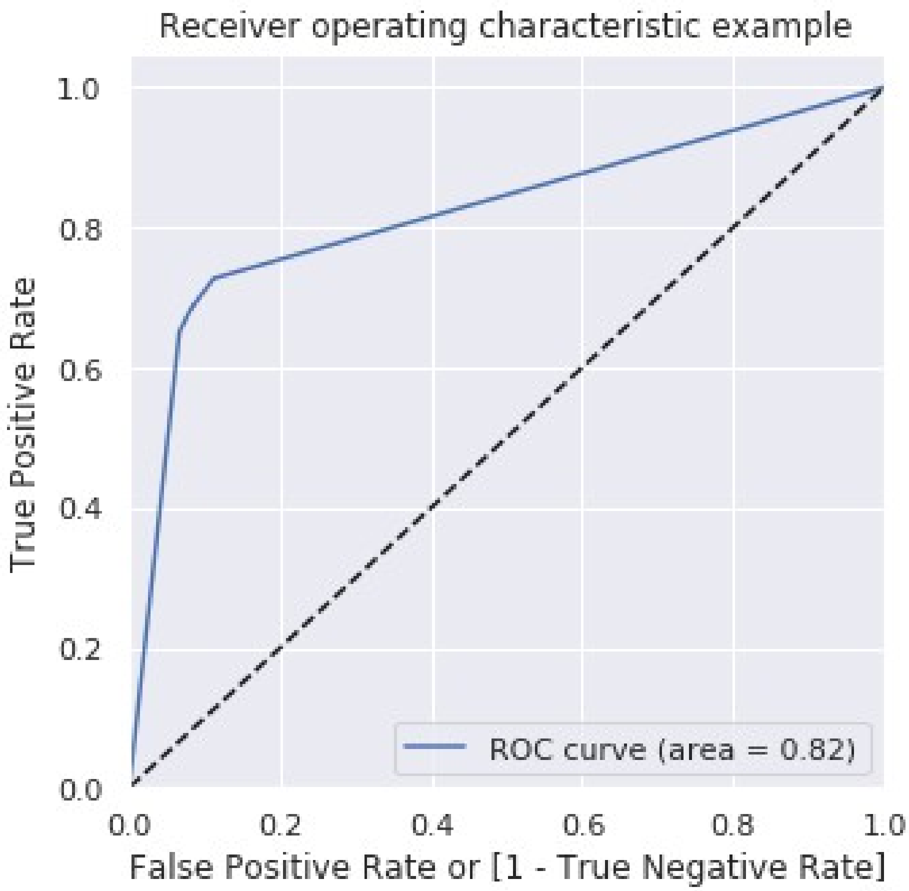
```
plot_precision_recall_curve([x[1] for x in
model_rf.predict_proba(X_train_forest)], y_train_smote.values)
```



From the above graph we see that the recall and precision is not dipping with some variation in the threshhold. The model is robust around the threshold value and has very well bifurcated the 0s and 1s.

```
predict_summarize([x[1] for x in
model_rf.predict_proba(X_train_forest)],
                  y_train_smote, 0.32, True)
```

```
Accuracy =  0.8277187723985143
Sensitivity =  0.7259774109470026
Specificity =  0.8887614678899083
False Positive Rate =  0.11123853211009174

Precision =  0.7965681601525262
Recall =  0.7259774109470026
Plotting
```

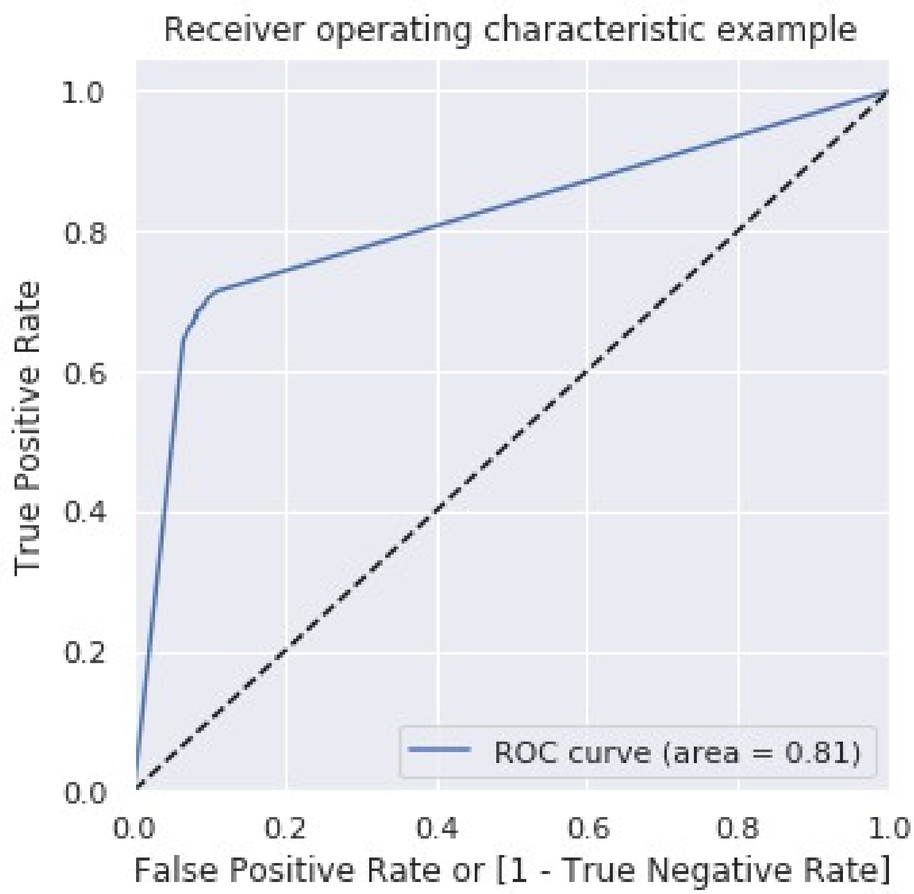



```
            predicted_no  predicted_yes
ind
actual_no          17050           2134
actual_yes          3154           8356
```

```
predict_summarize([x[1] for x in
model_rf.predict_proba(X_test_forest)],
                  y_test, 0.32, True)
```

```
Accuracy =  0.8745002221235006
Sensitivity =  0.7142857142857143
Specificity =  0.8894826329851834
False Positive Rate =  0.11051736701481661
```

```
Precision =  0.3767123287671233
Recall =  0.7142857142857143
Plotting
```




```
            predicted_no  predicted_yes
ind
actual_no           7324            910
actual_yes           220            550
```

Recall, which tells us the correctly predicted relevant results is ~70% with accuracy of ~87%.

SVM

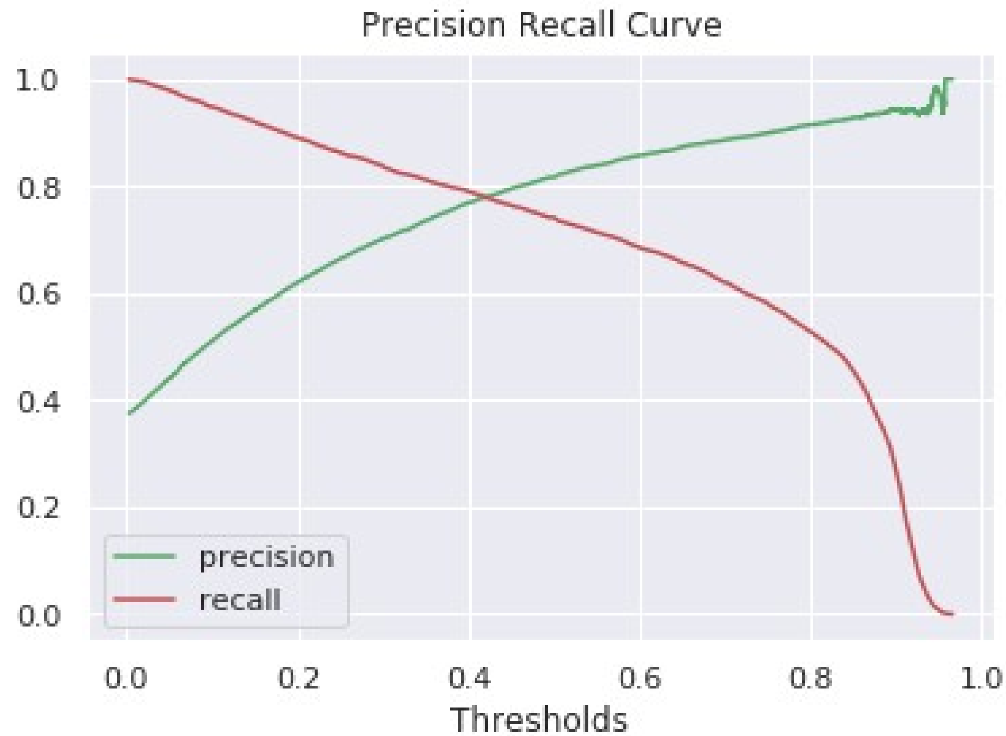
We can attain the best hyper parameters using the grid search for SVM as well

```
features = pc_features_set_1
X_train_svm = X_train_smote_transformed[features]
X_test_svm = X_test_transformed[features]
```

'gamma' and 'C' : multiple ranges to find the best combination

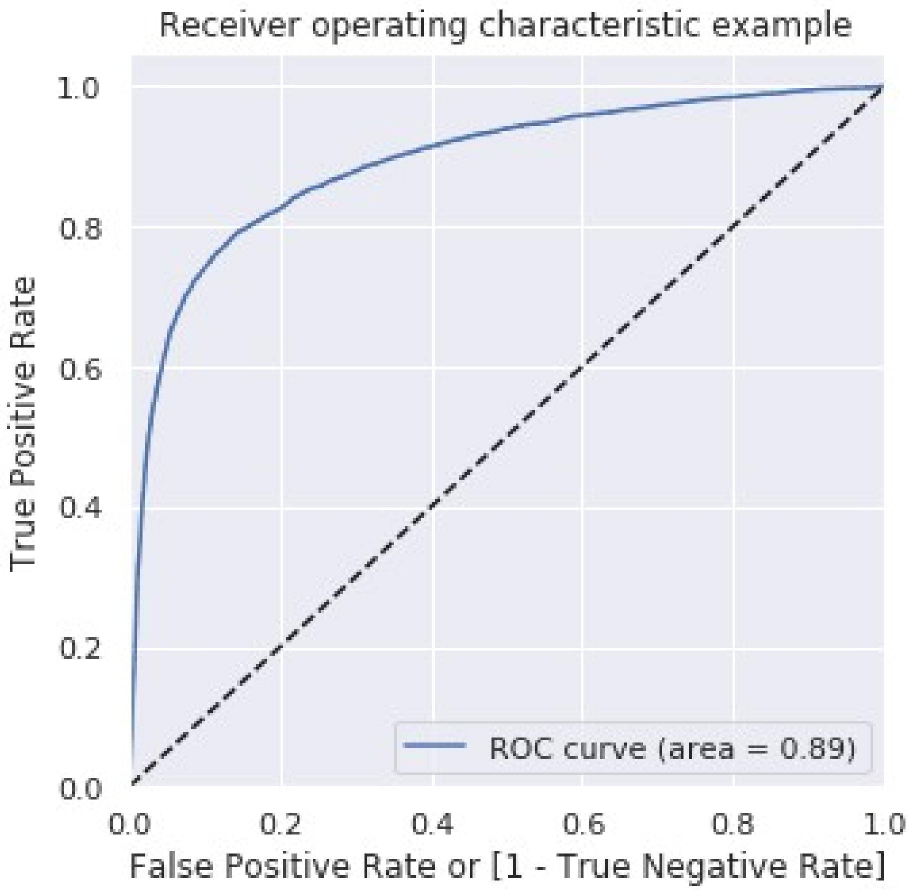
```
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
```

```
                 'kernel': ['rbf']}

svm_grid_search = GridSearchCV(svm.SVC(), param_grid, refit = True,
verbose = 3, n_jobs=6)

svm_grid_search.fit(X_train, y_train)

svm_grid_search.best_estimator_.get_params()

{'C': 10,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 0.0001,
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}

clf = svm.SVC(C=10, break_ties=False, cache_size=200,
class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.0001,
kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
clf.fit(X_train_svm, y_train_smote)

SVC(C=10, break_ties=False, cache_size=200, class_weight=None,
coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.0001,
kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True,
tol=0.001,
    verbose=False)

plot_precision_recall_curve([x[1] for x in
clf.predict_proba(X_train_svm)], y_train_smote)
```




```
predict_summarize([x[1] for x in clf.predict_proba(X_train_svm)],
y_train_smote, 0.4, True)
```

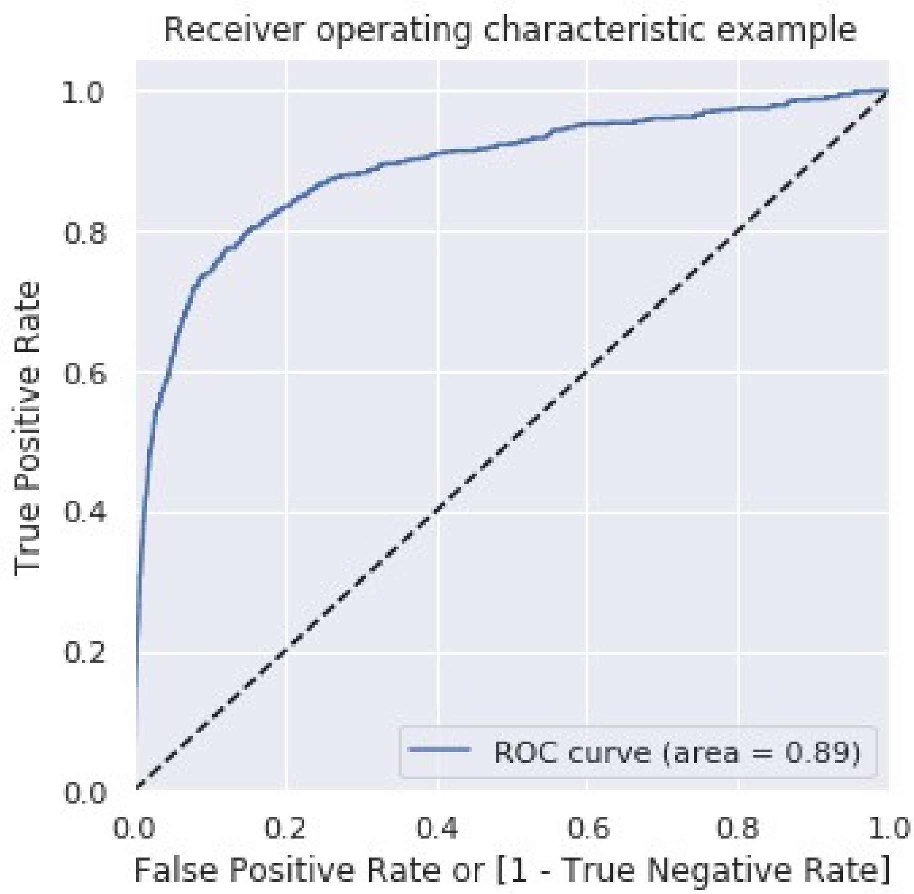
```
Accuracy =  0.8321821854434092
Sensitivity =  0.7900086880973067
Specificity =  0.8574854045037531
False Positive Rate =  0.14251459549624687

Precision =  0.7688340238437473
Recall =  0.7900086880973067
Plotting
```




```
            predicted_no  predicted_yes
ind
actual_no          16450           2734
actual_yes          2417           9093
```

```
predict_summarize([x[1] for x in clf.predict_proba(X_test_svm)],
y_test, 0.4, True)
```

```
Accuracy =  0.8530653043091959
Sensitivity =  0.7831168831168831
Specificity =  0.8596065095943648
False Positive Rate =  0.14039349040563517

Precision =  0.3428084138715179
Recall =  0.7831168831168831
Plotting
```




```
             predicted_no  predicted_yes
ind
actual_no            7078           1156
actual_yes            167            603
```

Observation:

In this problem, the objective is to identify the churners correctly as compared to identification of non churners. We can thuse look for an optimal Sensitivity value. The SVM model gives the best sensitivity at 78%, accuracy of approximately 85 in both train & test set.Although, in this case, precision is low as some non churners have been identified as churners in the test set. But this can be compensated for the higher sensitivity/recall which is of higher priority. Hence, the model which can be used to identify churners best is *SVM*

Model Building - Features: Original Variables

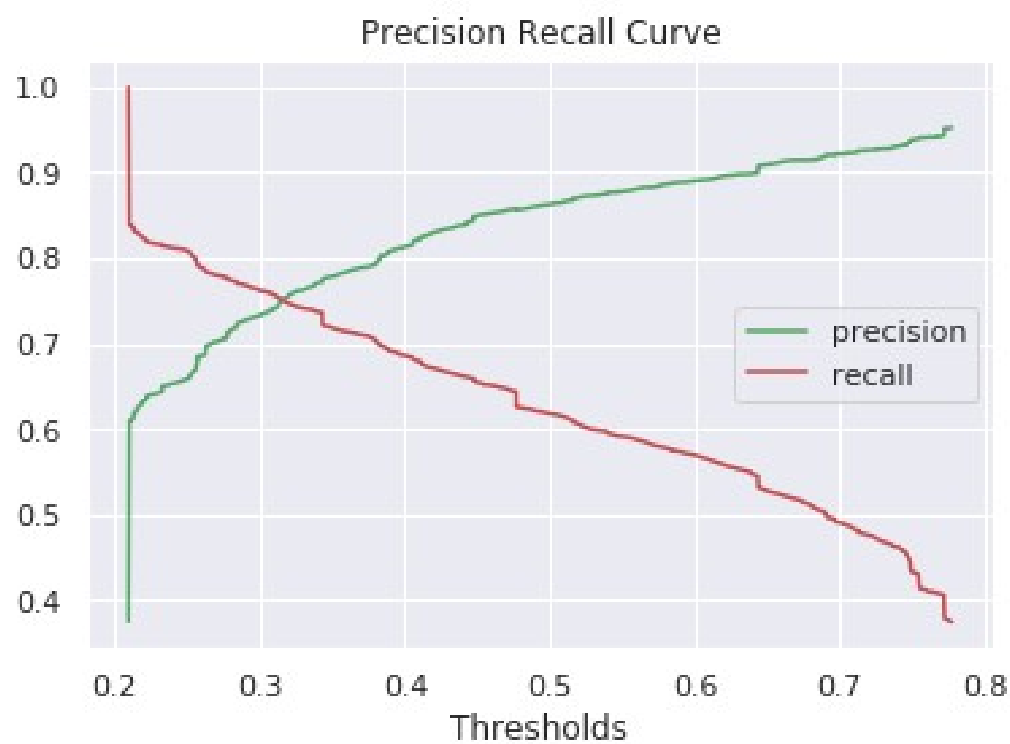
RF

The objective of this is to come up with variables and their impact on the churn for the business to be able to take better decisions.

```
model_rf_2 = RandomForestClassifier(bootstrap=True,
                                    criterion = 'entropy',
                                    max_depth=10,
                                    max_features='auto',
                                    min_samples_split=100,
                                    n_estimators=1000,
                                    min_impurity_decrease=0.2,
                                    random_state = 42)
model_rf_2.fit(X_train_smote_original, y_train_smote)
```

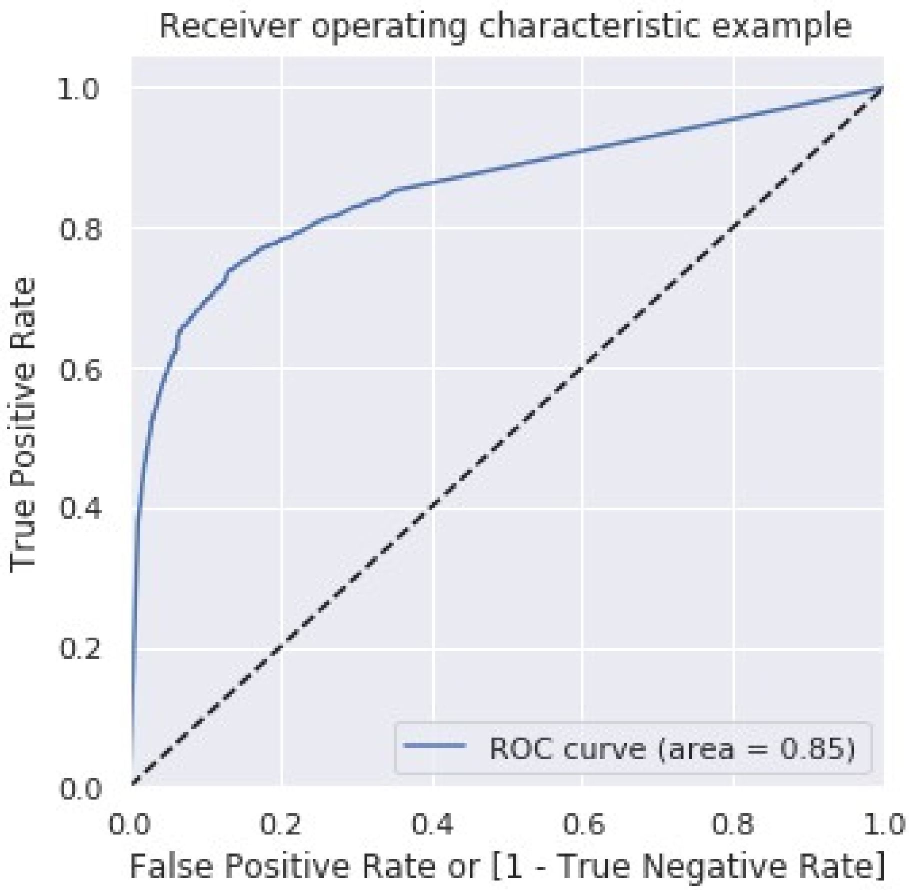
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight=None,
                       criterion='entropy', max_depth=10,
max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.2,
min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=100,
                       min_weight_fraction_leaf=0.0,
n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=42,
verbose=0,
                       warm_start=False)
```

```
plot_precision_recall_curve([x[1] for x in
model_rf_2.predict_proba(X_train_smote_original)], y_train_smote)
```



```
predict_summarize([x[1] for x in
model_rf_2.predict_proba(X_train_smote_original)], y_train_smote,
0.32, True)
```

```
Accuracy =  0.8149801264090701
Sensitivity =  0.7466550825369244
Specificity =  0.8559737281067556
False Positive Rate =  0.14402627189324438

Precision =  0.756713920929823
Recall =  0.7466550825369244
Plotting
```

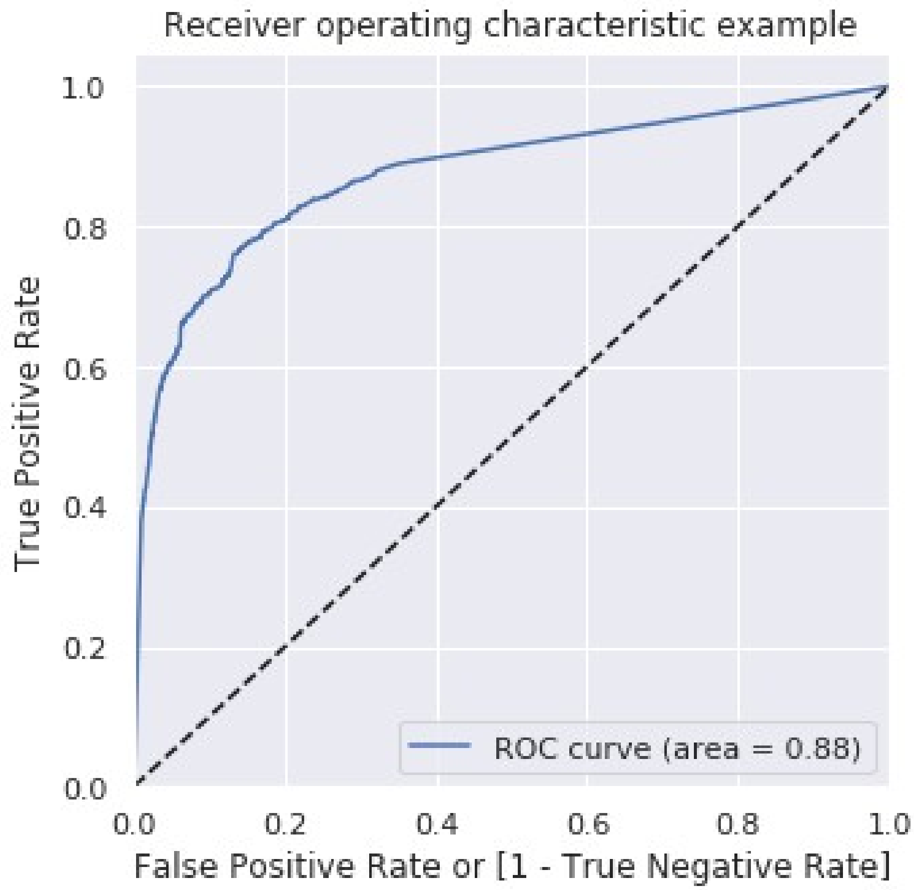



```
            predicted_no  predicted_yes
ind
actual_no          16421           2763
actual_yes          2916           8594
```

```
predict_summarize([x[1] for x in
model_rf_2.predict_proba(X_test_original)], y_test, 0.32, True)
```

```
Accuracy =  0.8471790315415371
Sensitivity =  0.7714285714285715
```

```
Specificity =  0.8542628127277143
False Positive Rate =  0.14573718727228566

Precision =  0.3311036789297659
Recall =  0.7714285714285715
Plotting
```

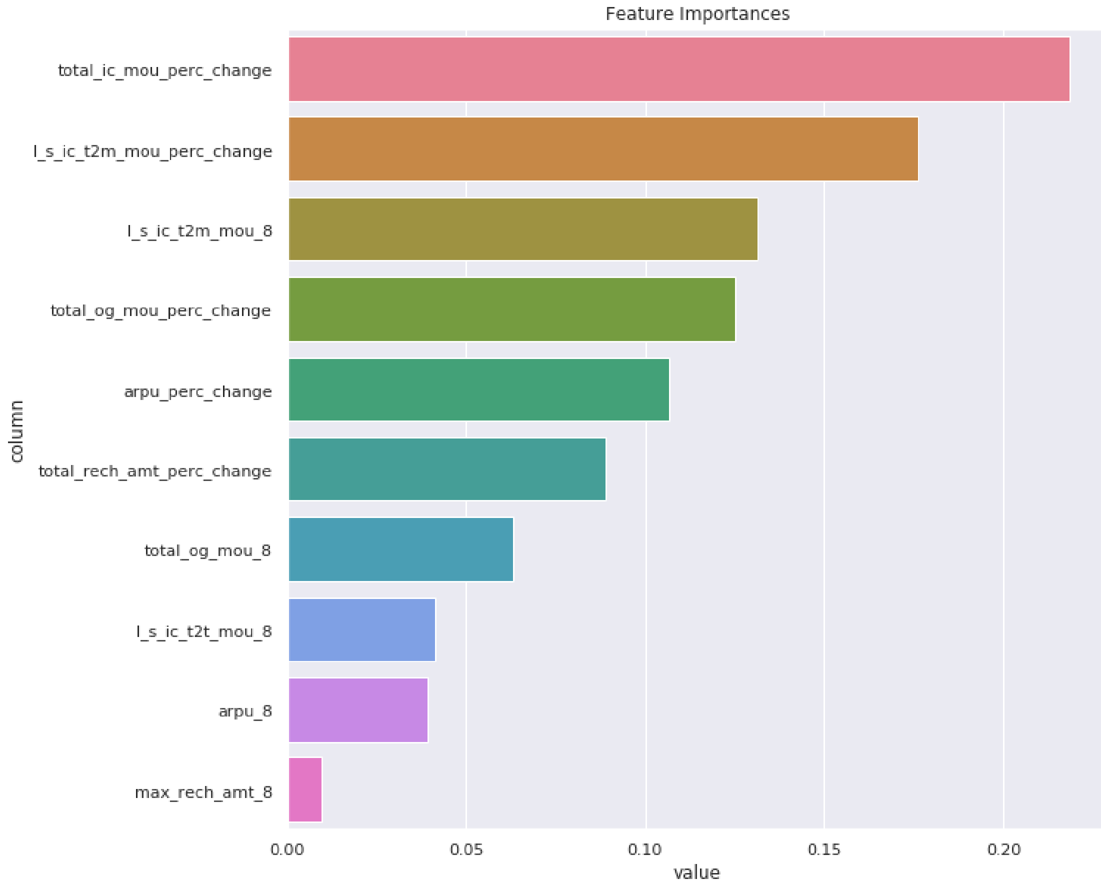



```
            predicted_no  predicted_yes
ind
actual_no           7034           1200
actual_yes           176            594
```

Observation

The above model built using the origial non-scaled, variables also gives a good sensitivity of 77 on the test data and we can thus use the feature importance from the above model to obtain important observations.

```
plot_feature_importance(X_train_smote_original.columns,
model_rf_2.feature_importances_)
```



Important Features - Visualization

```
important_parameters =
pandas.DataFrame(zip(X_train_smote_original.columns,
model_rf_2.feature_importances_))
important_parameters.columns = ['parameter','importance']
important_parameters =
important_parameters.loc[important_parameters['importance']>0]
important_parameters.sort_values('importance', ascending=False)
```

	parameter	importance
20	total_ic_mou_perc_change	0.218478
24	l_s_ic_t2m_mou_perc_change	0.176087
11	l_s_ic_t2m_mou_8	0.131522
19	total_og_mou_perc_change	0.125000
18	arpu_perc_change	0.106522
21	total_rech_amt_perc_change	0.089130
1	total_og_mou_8	0.063043
9	l_s_ic_t2t_mou_8	0.041304
0	arpu_8	0.039130
4	max_rech_amt_8	0.009783

```
variables = important_parameters['parameter'].values
ratio_plots_num(data, variables, 'churn_flag', 5, 'husl')
```


arpu_8
Converted %
-282.7-22.5
22.5-327.8
327.8-633.0
633.0-938.2
938.2-1243.5
total_og_mou_8
0.0-359.9
359.9-719.8
719.8-1079.7
1079.7-1439.7
1439.7-1799.6
max_rech_amt_8
0.0-78.9
78.9-157.8
157.8-236.7
236.7-315.6
315.6-394.5
l_s_ic_t2t_mou_8
0.0-20.6
20.6-41.1
41.1-61.7
61.7-82.2
82.2-102.8
l_s_ic_t2m_mou_8
0.0-52.2
52.2-104.3
104.3-156.5
156.5-208.7
208.7-260.8
arpu_perc_change
-1.0--0.6
-0.6--0.1
-0.1-0.3
0.3-0.7
0.7-1.2
total_og_mou_perc_change
-1.2--0.7
-0.7--0.3
-0.3-0.1
0.1-0.6
0.6-1.0
total_ic_mou_perc_change
-1.2--0.8
-0.8--0.3
-0.3-0.1
0.1-0.6
0.6-1.0
total_rech_amt_perc_change
-1.1--0.7
-0.7--0.2
-0.2-0.2
0.2-0.6
0.6-1.0
l_s_ic_t2m_mou_perc_change
-1.4--0.9
-0.9--0.4
-0.4-0.0
0.0-0.5
0.5-1.0

Conclusion:

We can see few of the above parameters impact clearly on the churn ratio.

- arpu_perc_change > 0.7
- total_og_mou_perc_change > 0.6
- total_ic_mou_perc_change > 0.6
- total_rech_amt_perc_change > 0.6
- l_s_ic_t2m_nou_perc_change > 0.5

All the above changes from the good phase to the action phase are clear indicators of churn through which the operator can identify & mitigate the risk.

The below are indicators from the action phase which can further be used to confirm the same.

- arpu_8 < 22
- total_og_mou_8 < 360
- max_rech_amt_8 < 79
- l_s_ic_t2t_mou_8 < 21
- l_s_ic_t2m_mou_8 < 52

The team must ideally use the above symptoms to identify possible churners and appropriately handle their retention strategy.