General Electric Customer Churn Problem

GE has partnered with a cellular carrier–SmartAppCellular–that provides dedicated bandwidth and configuration services for cell phone applications. GE is beginning to experience a small amount of attrition, and based on customer feedback, it is related to the cellular service and not the application. GE Healthcare recognizes that other vendors are beginning to compete in this space and is attempting to identify ways to retain its customers.

The Customer Account Management team would like to determine if this data can be used to identify subscribers that may churn. It is important to be able to understand churn default drivers for metadata like longevity, cell usage, and other pertinent groupings which come from the analysis.

-Genesis Taylor

Verification Data Cleansing.

Import Modules

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import time
#import timeit
from matplotlib import pyplot as plt
%matplotlib inline
plt.style.use('dark_background')

#stats
from scipy import stats
from scipy.stats import chi2_contingency

#sklearn modeling and metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, log_loss
from sklearn.metrics import classification_report
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
```

```
from sklearn.model_selection  import train_test_split
from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample

#warning ignorer
import warnings
warnings.filterwarnings("ignore")

#for imbalance
from imblearn.over_sampling import SMOTE
from collections import Counter
```

Import and Explore Data Set

In [2]:
```
#import data set into datafream
df = pd.read_csv(r'Cell Data_Verification.csv')
```

In [3]:
```
#check columns and shape
print('Columns:\n',df.columns)
print('\n')
print("Dataframe Shape:", df.shape)

df.head()
```

```
Columns:
 Index(['REVENUE', 'MOU', 'RECCHRGE', 'DIRECTAS', 'OVERAGE', 'ROAM', 'CHANGEM',
       'CHANGER', 'DROPVCE', 'BLCKVCE', 'UNANSVCE', 'CUSTCARE', 'THREEWAY',
       'MOUREC', 'OUTCALLS', 'INCALLS', 'PEAKVCE', 'OPEAKVCE', 'DROPBLK',
       'CALLFWDV', 'CALLWAIT', 'CHURN', 'MONTHS', 'UNIQSUBS', 'ACTVSUBS',
       'CSA', 'PHONES', 'MODELS', 'EQPDAYS', 'CUSTOMER', 'AGE1', 'AGE2',
       'CHILDREN', 'CREDITA', 'CREDITAA', 'CREDITB', 'CREDITC', 'CREDITDE',
       'CREDITGY', 'CREDITZ', 'CREDIT_RATING', 'PRIZMRUR', 'PRIZMUB',
       'PRIZMTWN', 'Column 45', 'REFURB', 'WEBCAP', 'TRUCK', 'RV', 'OCCPROF',
       'OCCCLER', 'OCCCRFT', 'OCCSTUD', 'OCCHMKR', 'OCCRET', 'OCCSELF', 'OCC',
       'OCC_LABEL', 'OWNRENT', 'MARRYUN', 'MARRYYES', 'MARRYNO', 'MARRY',
       'MARRY_LABEL', 'MAILORD', 'MAILRES', 'MAILFLAG', 'TRAVEL', 'PCOWN',
       'CREDITCD', 'RETCALLS', 'RETACCPT', 'NEWCELLY', 'NEWCELLN', 'REFER',
       'INCMISS', 'INCOME', 'MCYCLE', 'CREDITAD', 'SETPRCM', 'SETPRC',
       'RETCALL', 'CALIBRAT', 'CHURNDEP'],
      dtype='object')


Dataframe Shape: (150, 84)
```

Out[3]:

	REVENUE	MOU	RECCHRGE	DIRECTAS	OVERAGE	ROAM	CHANGEM	CHANGER	DROPVCE	BLCKVCE	...	REFER	INCMISS	INCOME	M(
0	45.69	246.00	44.99	0.0	26.75	0.00	-109.00	-10.70	5.33	0.67	...	0	1	0	
1	40.55	291.00	29.99	0.0	3.00	3.17	155.00	6.93	2.00	0.00	...	0	0	1	
2	35.58	296.00	44.99	0.0	4.25	0.00	-15.00	1.76	9.00	1.33	...	0	1	0	
3	9.13	7.75	7.50	0.0	7.25	0.00	16.25	4.13	1.33	0.00	...	0	0	5	
4	57.99	515.50	59.99	0.0	0.00	0.00	-11.50	0.00	8.67	3.67	...	0	0	4	

5 rows × 84 columns

In [4]:
```
df.describe()
```

Out[4]:

	REVENUE	MOU	RECCHRGE	DIRECTAS	OVERAGE	ROAM	CHANGEM	CHANGER	DROPVCE	BLCKVCE	...	REFE
count	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000	149.000000	149.000000	150.000000	150.000000	...	150.00000
mean	59.297067	520.185000	45.567467	0.883667	41.760000	0.955333	-5.884228	0.069060	5.849267	4.233000	...	0.07333
std	53.091073	516.239186	23.127887	2.019887	109.776336	4.829327	218.397563	81.271003	7.359560	9.373958	...	0.26155
min	5.050000	0.000000	0.000000	0.000000	0.000000	0.000000	-814.500000	-188.890000	0.000000	0.000000	...	0.00000
25%	32.620000	208.937500	30.000000	0.000000	0.000000	0.000000	-59.000000	-4.130000	1.000000	0.082500	...	0.00000
50%	49.130000	361.250000	44.990000	0.000000	3.125000	0.000000	-10.500000	-0.450000	3.000000	0.835000	...	0.00000
75%	62.132500	628.312500	59.680000	0.740000	30.312500	0.122500	54.750000	1.140000	8.502500	4.247500	...	0.00000
max	376.390000	2617.500000	191.000000	16.090000	954.750000	52.700000	1244.750000	895.570000	36.670000	73.330000	...	1.00000

8 rows × 81 columns

In [5]:
```
#datatype count
df.dtypes.value_counts()
```

Out[5]:
```
int64      57
float64    24
object      3
dtype: int64
```

In [6]:
```python
#unique values
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df.nunique().sort_values(ascending=False))
```

```
CUSTOMER         150
REVENUE          143
MOU              143
EQPDAYS          142
CHANGEM          138
MOUREC           136
PEAKVCE          132
CHANGER          126
OPEAKVCE         124
OUTCALLS         102
UNANSVCE         100
CSA               97
OVERAGE           77
DROPBLK           64
INCALLS           57
RECCHRGE          55
DROPVCE           48
BLCKVCE           36
ROAM              34
AGE2              29
AGE1              28
CUSTCARE          26
CALLWAIT          25
DIRECTAS          23
MONTHS            17
INCOME            10
SETPRC            10
THREEWAY           9
CREDIT_RATING      7
OCC_LABEL          7
OCC                7
PHONES             5
Column 45          4
UNIQSUBS           4
MODELS             4
ACTVSUBS           3
MARRY              3
CHURN              2
CHILDREN           2
CREDITA            2
CREDITGY           2
CREDITAA           2
CREDITB            2
CREDITC            2
CREDITDE           2
```

```
PRIZMRUR          2
CREDITZ           2
RETACCPT          2
MAILORD           2
MAILRES           2
MAILFLAG          2
TRAVEL            2
PCOWN             2
CREDITCD          2
RETCALLS          2
NEWCELLY          2
CALIBRAT          2
NEWCELLN          2
REFER             2
INCMISS           2
MCYCLE            2
CREDITAD          2
SETPRCM           2
RETCALL           2
MARRY_LABEL       2
MARRYNO           2
OCCCLER           2
PRIZMTWN          2
MARRYUN           2
OWNRENT           2
OCCSELF           2
OCCRET            2
PRIZMUB           2
OCCSTUD           2
OCCCRFT           2
MARRYYES          2
OCCPROF           2
RV                2
TRUCK             2
WEBCAP            2
REFURB            2
CALLFWDV          1
OCCHMKR           1
CHURNDEP          1
dtype: int64
```

In [7]:

```python
#Check Missing/null data
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df.isnull().sum().sort_values(ascending=False))
```

```
CHURNDEP        119
AGE2              3
AGE1              3
CHANGEM           1
```

CHANGER	1
UNIQSUBS	0
ACTVSUBS	0
CSA	0
PHONES	0
MODELS	0
EQPDAYS	0
CUSTOMER	0
CHILDREN	0
CHURN	0
CREDITA	0
CREDITAA	0
CREDITB	0
CREDITC	0
CREDITDE	0
CREDITGY	0
CREDITZ	0
MONTHS	0
CALLWAIT	0
CALIBRAT	0
UNANSVCE	0
MOU	0
RECCHRGE	0
DIRECTAS	0
OVERAGE	0
ROAM	0
DROPVCE	0
BLCKVCE	0
CUSTCARE	0
CALLFWDV	0
THREEWAY	0
MOUREC	0
OUTCALLS	0
INCALLS	0
PEAKVCE	0
OPEAKVCE	0
DROPBLK	0
CREDIT_RATING	0
PRIZMRUR	0
PRIZMUB	0
MARRY_LABEL	0
MAILRES	0
MAILFLAG	0
TRAVEL	0
PCOWN	0
CREDITCD	0
RETCALLS	0
RETACCPT	0
NEWCELLY	0

```
NEWCELLN        0
REFER           0
INCMISS         0
INCOME          0
MCYCLE          0
CREDITAD        0
SETPRCM         0
SETPRC          0
RETCALL         0
MAILORD         0
MARRY           0
PRIZMTWN        0
MARRYNO         0
Column 45       0
REFURB          0
WEBCAP          0
TRUCK           0
RV              0
OCCPROF         0
OCCCLER         0
OCCCRFT         0
OCCSTUD         0
OCCHMKR         0
OCCRET          0
OCCSELF         0
OCC             0
OCC_LABEL       0
OWNRENT         0
MARRYUN         0
MARRYYES        0
REVENUE         0
dtype: int64
```

In [8]:

```
#predicited variable
df['CHURN'].value_counts(ascending=True)
```

Out[8]:

```
1     31
0    119
Name: CHURN, dtype: int64
```

Data Cleansing

In [9]:

```
#standardize all columns to lowercase for ease of use in querying
df.columns = map(str.lower, df.columns)
#verify
print('Columns:\n',df.columns)
```

```
Columns:
```

```
 Index(['revenue', 'mou', 'recchrge', 'directas', 'overage', 'roam', 'changem',
       'changer', 'dropvce', 'blckvce', 'unansvce', 'custcare', 'threeway',
       'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce', 'dropblk',
       'callfwdv', 'callwait', 'churn', 'months', 'uniqsubs', 'actvsubs',
       'csa', 'phones', 'models', 'eqpdays', 'customer', 'age1', 'age2',
       'children', 'credita', 'creditaa', 'creditb', 'creditc', 'creditde',
       'creditgy', 'creditz', 'credit_rating', 'prizmrur', 'prizmub',
       'prizmtwn', 'column 45', 'refurb', 'webcap', 'truck', 'rv', 'occprof',
       'occcler', 'occcrft', 'occstud', 'occhmkr', 'occret', 'occself', 'occ',
       'occ_label', 'ownrent', 'marryun', 'marryyes', 'marryno', 'marry',
       'marry_label', 'mailord', 'mailres', 'mailflag', 'travel', 'pcown',
       'creditcd', 'retcalls', 'retaccpt', 'newcelly', 'newcelln', 'refer',
       'incmiss', 'income', 'mcycle', 'creditad', 'setprcm', 'setprc',
       'retcall', 'calibrat', 'churndep'],
      dtype='object')
```

In [10]:

```
#mislabeled column
df.rename(columns={'column 45':'przm_num'}, inplace=True)

#verify
print('Columns:\n',df.columns)
```

```
Columns:
 Index(['revenue', 'mou', 'recchrge', 'directas', 'overage', 'roam', 'changem',
       'changer', 'dropvce', 'blckvce', 'unansvce', 'custcare', 'threeway',
       'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce', 'dropblk',
       'callfwdv', 'callwait', 'churn', 'months', 'uniqsubs', 'actvsubs',
       'csa', 'phones', 'models', 'eqpdays', 'customer', 'age1', 'age2',
       'children', 'credita', 'creditaa', 'creditb', 'creditc', 'creditde',
       'creditgy', 'creditz', 'credit_rating', 'prizmrur', 'prizmub',
       'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv', 'occprof',
       'occcler', 'occcrft', 'occstud', 'occhmkr', 'occret', 'occself', 'occ',
       'occ_label', 'ownrent', 'marryun', 'marryyes', 'marryno', 'marry',
       'marry_label', 'mailord', 'mailres', 'mailflag', 'travel', 'pcown',
       'creditcd', 'retcalls', 'retaccpt', 'newcelly', 'newcelln', 'refer',
       'incmiss', 'income', 'mcycle', 'creditad', 'setprcm', 'setprc',
       'retcall', 'calibrat', 'churndep'],
      dtype='object')
```

In [11]:

```
#predicited variable
df['churn'].value_counts(ascending=True)
```

Out[11]:

```
1     31
0    119
Name: churn, dtype: int64
```

In [12]:

```
#drop churndep because it is just a field set up for logreg
#drop calibrat bc I want to do my own separation
```

```
df = df.drop(['churndep'], axis=1)
df = df.drop(['calibrat'], axis=1)
```

Changing Data Types

In [13]:
```
'''
    Using data dictionary to fix some data types to string/objects.
    So that they won't be misrepresented in any cleaning and calculations.
    They're not actually numbers.
    This is mostly done for analysis purposes in Tableau.
    It is also done to properly handle null values.

'''

df['children'] = df['children'].apply(str)
df['churn'] = df['churn'].apply(str)
df['credit_rating'] = df['credit_rating'].apply(str)
df['credita'] = df['credita'].apply(str)
df['creditaa'] = df['creditaa'].apply(str)
df['creditad'] = df['creditad'].apply(str)
df['creditb'] = df['creditb'].apply(str)
df['creditc'] = df['creditc'].apply(str)
df['creditcd'] = df['creditcd'].apply(str)
df['creditde'] = df['creditde'].apply(str)
df['creditgy'] = df['creditgy'].apply(str)
df['creditz'] = df['creditz'].apply(str)
df['incmiss'] = df['incmiss'].apply(str)
df['income'] = df['income'].apply(str)
df['mailflag'] = df['mailflag'].apply(str)
df['mailord'] = df['mailord'].apply(str)
df['mailres'] = df['mailres'].apply(str)
df['marry'] = df['marry'].apply(str)
df['marryno'] = df['marryno'].apply(str)
df['marryun'] = df['marryun'].apply(str)
df['marryyes'] = df['marryyes'].apply(str)
df['mcycle'] = df['mcycle'].apply(str)
df['newcelln'] = df['newcelln'].apply(str)
df['newcelly'] = df['newcelly'].apply(str)
df['mailflag'] = df['mailflag'].apply(str)
df['mailord'] = df['mailord'].apply(str)
df['mailres'] = df['mailres'].apply(str)
df['marryno'] = df['marryno'].apply(str)
df['marryun'] = df['marryun'].apply(str)
df['marryyes'] = df['marryyes'].apply(str)
df['mcycle'] = df['mcycle'].apply(str)
```

```python
df['newcelln'] = df['newcelln'].apply(str)
df['newcelly'] = df['newcelly'].apply(str)
df['occ'] = df['occ'].apply(str)
df['occ_label'] = df['occ_label'].apply(str)
df['occcler'] = df['occcler'].apply(str)
df['occcrft'] = df['occcrft'].apply(str)
df['occhmkr'] = df['occhmkr'].apply(str)
df['occprof'] = df['occprof'].apply(str)
df['occret'] = df['occret'].apply(str)
df['occself'] = df['occself'].apply(str)
df['occstud'] = df['occstud'].apply(str)
df['ownrent'] = df['ownrent'].apply(str)
df['pcown'] = df['pcown'].apply(str)
df['prizmrur'] = df['prizmrur'].apply(str)
df['prizmtwn'] = df['prizmtwn'].apply(str)
df['prizmub'] = df['prizmub'].apply(str)
df['przm_num'] = df['przm_num'].apply(str)
df['refurb'] = df['refurb'].apply(str)
df['retcall'] = df['retcall'].apply(str)
df['rv'] = df['rv'].apply(str)
df['setprcm'] = df['setprcm'].apply(str)
df['travel'] = df['travel'].apply(str)
df['truck'] = df['truck'].apply(str)
df['webcap'] = df['webcap'].apply(str)
```

In [14]:

```python
#datatype count
df.dtypes.value_counts()
```

Out[14]:

```
object     48
float64    23
int64      11
dtype: int64
```

Missing Values

In [15]:

```python
#Check Missing/null data
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df.isnull().sum().sort_values(ascending=False))
```

```
age1         3
age2         3
changem      1
changer      1
retcall      0
uniqsubs     0
actvsubs     0
```

```
csa              0
phones           0
models           0
eqpdays          0
customer         0
children         0
churn            0
credita          0
creditaa         0
creditb          0
creditc          0
creditde         0
creditgy         0
months           0
callfwdv         0
callwait         0
unansvce         0
mou              0
recchrge         0
directas         0
overage          0
roam             0
dropvce          0
blckvce          0
custcare         0
setprc           0
threeway         0
mourec           0
outcalls         0
incalls          0
peakvce          0
opeakvce         0
dropblk          0
creditz          0
credit_rating    0
prizmrur         0
prizmub          0
marry_label      0
mailord          0
mailres          0
mailflag         0
travel           0
pcown            0
creditcd         0
retcalls         0
retaccpt         0
newcelly         0
newcelln         0
refer            0
```

```
incmiss         0
income          0
mcycle          0
creditad        0
setprcm         0
marry           0
marryno         0
marryyes        0
occcler         0
prizmtwn        0
przm_num        0
refurb          0
webcap          0
truck           0
rv              0
occprof         0
occcrft         0
marryun         0
occstud         0
occhmkr         0
occret          0
occself         0
occ             0
occ_label       0
ownrent         0
revenue         0
dtype: int64
```

Age1

In [16]:

```
#check values of age1

print("Age1 Values:")
print("Average Age1 w/o Zeroes: ", round(df['age1'].loc[df['age1']!=0].mean(),0))
print("Average Age1: ", round(df['age1'].mean(),0))
print("Minimum Age1 WITH Zeroes: ", df['age1'].min())
print("Minimum Age1 w/o Zeroes: ", df['age1'].loc[df['age1']!=0].min())
print("Maximum Age1: ", df['age1'].max())
print("Null values for Age1: ", pd.isnull(df['age1']).sum())

#check # 0s in age1
print("Number of Age1 Zeroes: ",(df['age1'] ==0).sum())
```

```
Age1 Values:
Average Age1 w/o Zeroes:  43.0
Average Age1:  33.0
Minimum Age1 WITH Zeroes:  0.0
Minimum Age1 w/o Zeroes:  20.0
```

```
Maximum Age1:  76.0
Null values for Age1:  3
Number of Age1 Zeroes:  35
```

In [17]:

```
''' Fill null age values to 0 to match the other ages that are missing AS 0.
    Will also create a "Missing" group for ages out of those groups later. '''

df['age1'].fillna(value=0, inplace=True)
```

In [18]:

```
#recheck values of age1

print("Age1 Values")
print("Average Age1: ", round(df['age1'].mean(),0))
print("Minimum Age1: ", df['age1'].min())
print("Maximum Age1: ", df['age1'].max())
print("Null values for Age1: ", pd.isnull(df['age1']).sum())
#check # 0s in age1
print("Number of Age1 Zeroes: ",(df['age1'] ==0).sum())
```

```
Age1 Values
Average Age1:  32.0
Minimum Age1:  0.0
Maximum Age1:  76.0
Null values for Age1:  0
Number of Age1 Zeroes:  38
```

Age2

In [19]:

```
#check values of age2

print("Age2 Values:")
print("Average Age2 w/o Zeroes: ", round(df['age2'].loc[df['age2']!=0].mean(),0))
print("Average Age2: ", round(df['age2'].mean(),0))
print("Minimum Age2 WITH Zeroes: ", df['age2'].min())
print("Minimum Age2 w/o Zeroes: ", df['age2'].loc[df['age2']!=0].min())
print("Maximum Age2: ", df['age2'].max())
print("Null values for Age2: ", pd.isnull(df['age2']).sum())

#check # 0s in age2
print("Number of Age2 Zeroes: ",(df['age2'] ==0).sum())
```

```
Age2 Values:
Average Age2 w/o Zeroes:  44.0
Average Age2:  22.0
Minimum Age2 WITH Zeroes:  0.0
Minimum Age2 w/o Zeroes:  18.0
Maximum Age2:  84.0
```

```
Null values for Age2:  3
Number of Age2 Zeroes:  72
```

In [20]:
```python
''' Fill null age values to 0 to match the other ages that are missing AS 0.
    Will also create a "Missing" group for ages out of those groups later. '''

df['age2'].fillna(value=0, inplace=True)
```

In [21]:
```python
#recheck values of age2

print("Age2 Values")
print("Average Age2: ", round(df['age2'].mean(),0))
print("Minimum Age2: ", df['age2'].min())
print("Maximum Age2: ", df['age2'].max())
print("Null values for Age2: ", pd.isnull(df['age2']).sum())
#check # 0s in age2
print("Number of Age2 Zeroes: ",(df['age2'] ==0).sum())
```

```
Age2 Values
Average Age2:  22.0
Minimum Age2:  0.0
Maximum Age2:  84.0
Null values for Age2:  0
Number of Age2 Zeroes:  75
```

Because the values for the remaining null columns can legitimately have a zero value, and are numerical and discrete, I am going to fill the rest of those with their mean. I think that it is a safe choice being that the highest null is 9/1000.

In [22]:
```python
#fill rest of nulls with their averages
df= df.fillna(df.mean())
```

In [23]:
```python
#recheck nulls
df.isnull().sum().sort_values(ascending=False)
```

Out[23]:
```
retcall      0
callwait     0
months       0
uniqsubs     0
actvsubs     0
            ..
occself      0
occ          0
occ_label    0
ownrent      0
revenue      0
Length: 82, dtype: int64
```

Outliers

```
In [24]: with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
             print(df.describe(include='all'))
```

```
           revenue          mou    recchrge    directas    overage  \
count   150.000000   150.000000  150.000000  150.000000  150.000000
unique         NaN          NaN         NaN         NaN         NaN
top            NaN          NaN         NaN         NaN         NaN
freq           NaN          NaN         NaN         NaN         NaN
mean     59.297067   520.185000   45.567467    0.883667   41.760000
std      53.091073   516.239186   23.127887    2.019887  109.776336
min       5.050000     0.000000    0.000000    0.000000    0.000000
25%      32.620000   208.937500   30.000000    0.000000    0.000000
50%      49.130000   361.250000   44.990000    0.000000    3.125000
75%      62.132500   628.312500   59.680000    0.740000   30.312500
max     376.390000  2617.500000  191.000000   16.090000  954.750000

              roam      changem     changer     dropvce     blckvce  \
count   150.000000   150.000000  150.000000  150.000000  150.000000
unique         NaN          NaN         NaN         NaN         NaN
top            NaN          NaN         NaN         NaN         NaN
freq           NaN          NaN         NaN         NaN         NaN
mean      0.955333    -5.884228    0.069060    5.849267    4.233000
std       4.829327   217.663451   80.997823    7.359560    9.373958
min       0.000000  -814.500000 -188.890000    0.000000    0.000000
25%       0.000000   -58.375000   -4.110000    1.000000    0.082500
50%       0.000000   -10.375000   -0.450000    3.000000    0.835000
75%       0.122500    53.437500    1.105000    8.502500    4.247500
max      52.700000  1244.750000  895.570000   36.670000   73.330000

          unansvce    custcare    threeway      mourec    outcalls  \
count   150.000000  150.000000  150.000000  150.000000  150.000000
unique         NaN         NaN         NaN         NaN         NaN
top            NaN         NaN         NaN         NaN         NaN
freq           NaN         NaN         NaN         NaN         NaN
mean     26.717733    1.317600    0.224000  116.226867   28.850933
std      30.862717    2.581907    0.606198  149.648203   40.973890
min       0.000000    0.000000    0.000000    0.000000    0.000000
25%       5.670000    0.000000    0.000000   13.935000    5.415000
50%      18.165000    0.000000    0.000000   63.375000   15.000000
75%      34.917500    1.330000    0.330000  148.562500   37.247500
max     182.000000   17.330000    4.330000  816.950000  338.000000

           incalls     peakvce    opeakvce     dropblk callfwdv    callwait  \
count   150.000000  150.000000  150.000000  150.000000    150.0  150.000000
unique         NaN         NaN         NaN         NaN      NaN         NaN
top            NaN         NaN         NaN         NaN      NaN         NaN
```

freq	NaN	NaN	NaN	NaN	NaN	NaN
mean	8.675533	81.731133	71.075600	10.113267	0.0	1.553267
std	18.344717	85.094147	80.103965	12.333661	0.0	3.416694
min	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	0.000000	25.002500	16.835000	2.330000	0.0	0.000000
50%	2.165000	55.000000	42.670000	6.330000	0.0	0.330000
75%	9.585000	111.752500	99.920000	12.247500	0.0	1.585000
max	136.670000	565.670000	382.670000	81.330000	0.0	26.670000

	churn	months	uniqsubs	actvsubs	csa	phones \
count	150	150.000000	150.000000	150.000000	150	150.000000
unique	2	NaN	NaN	NaN	97	NaN
top	0	NaN	NaN	NaN	NYCBRO917	NaN
freq	119	NaN	NaN	NaN	10	NaN
mean	NaN	15.886667	1.480000	1.333333	NaN	1.526667
std	NaN	3.855679	0.720924	0.551589	NaN	0.864642
min	NaN	10.000000	1.000000	1.000000	NaN	1.000000
25%	NaN	13.000000	1.000000	1.000000	NaN	1.000000
50%	NaN	15.500000	1.000000	1.000000	NaN	1.000000
75%	NaN	18.000000	2.000000	2.000000	NaN	2.000000
max	NaN	26.000000	4.000000	3.000000	NaN	5.000000

	models	eqpdays	customer	age1	age2	children \
count	150.000000	150.000000	1.500000e+02	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	2
top	NaN	NaN	NaN	NaN	NaN	0
freq	NaN	NaN	NaN	NaN	NaN	108
mean	1.346667	369.986667	1.049450e+06	32.293333	22.026667	NaN
std	0.623765	171.431008	1.078289e+04	21.902023	24.189394	NaN
min	1.000000	5.000000	1.022063e+06	0.000000	0.000000	NaN
25%	1.000000	270.750000	1.040916e+06	5.000000	0.000000	NaN
50%	1.000000	384.500000	1.052111e+06	36.000000	9.000000	NaN
75%	2.000000	486.000000	1.058888e+06	48.000000	43.500000	NaN
max	4.000000	761.000000	1.064134e+06	76.000000	84.000000	NaN

	credita	creditaa	creditb	creditc	creditde	creditgy	creditz \
count	150	150	150	150	150	150	150
unique	2	2	2	2	2	2	2
top	0	0	0	0	0	0	0
freq	146	82	115	138	129	147	143
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	credit_rating	prizmrur	prizmub	prizmtwn	przm_num	refurb	webcap	truck \

count	150	150	150	150	150	150	150	150
unique	7	2	2	2	4	2	2	2
top	2	0	0	0	0	0	1	0
freq	68	144	109	125	94	131	137	112
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	rv	occprof	occcler	occcrft	occstud	occhmkr	occret	occself	occ \
count	150	150	150	150	150	150	150	150	150
unique	2	2	2	2	2	1	2	2	7
top	0	0	0	0	0	0	0	0	0
freq	134	126	149	147	149	150	147	147	115
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	occ_label	ownrent	marryun	marryyes	marryno	marry	marry_label	mailord \
count	150	150	150	150	150	150	150	150
unique	7	2	2	2	2	3	2	2
top	NONE	0	0	0	0	3	UNKNOWN	0
freq	115	104	93	97	110	57	110	95
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	mailres	mailflag	travel	pcown	creditcd	retcalls	retaccpt \
count	150	150	150	150	150	150.000000	150.000000
unique	2	2	2	2	2	NaN	NaN
top	0	0	0	0	1	NaN	NaN
freq	94	147	141	119	103	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	0.020000	0.013333
std	NaN	NaN	NaN	NaN	NaN	0.140469	0.115082
min	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000
25%	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000
50%	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000
75%	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000

```
max        NaN       NaN    NaN   NaN      NaN    1.000000    1.000000

       newcelly newcelln       refer incmiss income mcycle creditad setprcm  \
count       150      150  150.000000     150    150    150      150     150
unique        2        2         NaN       2     10      2        2       2
top           0        0         NaN       0      0      0        0       1
freq        127      141         NaN     118     32    148      149      97
mean        NaN      NaN    0.073333     NaN    NaN    NaN      NaN     NaN
std         NaN      NaN    0.261556     NaN    NaN    NaN      NaN     NaN
min         NaN      NaN    0.000000     NaN    NaN    NaN      NaN     NaN
25%         NaN      NaN    0.000000     NaN    NaN    NaN      NaN     NaN
50%         NaN      NaN    0.000000     NaN    NaN    NaN      NaN     NaN
75%         NaN      NaN    0.000000     NaN    NaN    NaN      NaN     NaN
max         NaN      NaN    1.000000     NaN    NaN    NaN      NaN     NaN

           setprc retcall
count   150.00000     150
unique        NaN       2
top           NaN       0
freq          NaN     147
mean     32.52980     NaN
std      56.89825     NaN
min       0.00000     NaN
25%       0.00000     NaN
50%       0.00000     NaN
75%      37.49000     NaN
max     199.99000     NaN
```

In [25]:

```
#creating a backup dataframe before removing outliers using IQR
df3 = df
```

In [26]:

```
#outlier detection
Q1 = df.quantile(0.05)
Q3 = df.quantile(0.95)
IQR = Q3 - Q1

#list of the outliers
dfiqr =((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum().sort_values(ascending=False)
```

In [27]:

```
#new dataframe with outliers removed
df=df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [28]:

```
#new dataframe shape
df.shape
```

Out[28]:

```
(134, 82)
```

In [29]:
```
#predicited variable
df['churn'].value_counts(ascending=True)
```

Out[29]:
```
1     25
0    109
Name: churn, dtype: int64
```

In [30]:
```
#with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    #print(np.median(df['churn']))
```

In [31]:
```
df['age1'].value_counts(ascending=True)
```

Out[31]:
```
22.0     1
76.0     1
64.0     1
68.0     1
70.0     1
20.0     1
66.0     1
38.0     2
60.0     2
50.0     2
24.0     2
72.0     2
34.0     3
44.0     4
40.0     4
54.0     4
26.0     4
48.0     4
56.0     5
58.0     5
30.0     6
36.0     6
28.0     6
52.0     7
46.0     8
42.0     8
32.0     9
0.0     34
Name: age1, dtype: int64
```

In [32]:
```
#Group the ages into groups
binsage=[0,17, 25, 35, 45, 55, 65, 99]
labelsage=['Missing','18-24','25-34','35-44','45-54', '55-64', '65+']
df['age_group'] = pd.cut(df['age1'], binsage, labels=labelsage, include_lowest=True)
```

In [33]:
```python
#Age
print("Distinct values for age:\n", set(df['age_group']))
```

```
Distinct values for age:
 {'65+', '35-44', '18-24', '25-34', 'Missing', '45-54', '55-64'}
```

In [34]:
```python
df['age_group'].value_counts(ascending=True)
```

Out[34]:
```
18-24       4
65+         6
55-64      13
35-44      24
45-54      25
25-34      28
Missing    34
Name: age_group, dtype: int64
```

In [35]:
```python
#Group the ages into groups
df['age_group2'] = pd.cut(df['age2'], binsage, labels=labelsage, include_lowest=True)

#Age
print("Distinct values for age:\n", set(df['age_group2']))
df['age_group2'].value_counts(ascending=True)
```

```
Distinct values for age:
 {'65+', '35-44', '18-24', '25-34', 'Missing', '45-54', '55-64'}
```

Out[35]:
```
65+         4
55-64       6
18-24       7
25-34      12
35-44      19
45-54      19
Missing    67
Name: age_group2, dtype: int64
```

In [36]:
```python
print("roam Values")
print("Average roam: ", df['roam'].mean())
print("Minimum roam: ", df['roam'].min())
print("Maximum roam: ", df['roam'].max())
print("Null values: ", pd.isnull(df['roam']).sum())
print("Roam Value Counts:", df['roam'].value_counts(ascending=True))
#check # 0s in Roam
print("Number of Roam Zeroes: ",(df['roam'] ==0).sum())
```

```
roam Values
Average roam:  0.3184328358208955
Minimum roam:  0.0
```

```
Maximum roam:  6.73
Null values:  0
Roam Value Counts: 0.48      1
0.71      1
1.85      1
4.12      1
0.32      1
0.15      1
0.13      1
0.45      1
1.51      1
4.91      1
3.17      1
0.26      1
6.73      1
1.84      1
2.38      1
0.44      1
4.00      1
1.75      1
4.66      1
0.19      1
0.64      2
0.20      2
0.10      3
0.16      4
0.00    103
Name: roam, dtype: int64
Number of Roam Zeroes:  103
```

In [37]:

```
#create groups for roaming
binsroam=[0,0.000000000001,1,2,3,4,5,6,7,8,9,10,11]
labelsroam=['Not_Roaming','1','2','3','4','5','6','7','8','9','10','over 10']
df['roaming_range'] = pd.cut(df['roam'], bins=binsroam, labels=labelsroam, include_lowest=True)
```

In [38]:

```
print("Distinct values for roam range:\n", set(df['roaming_range']))
print("Value counts for roaming range", df['roaming_range'].value_counts(ascending=True))
```

```
Distinct values for roam range:
 {'2', '4', '5', '7', '1', '3', 'Not_Roaming'}
Value counts for roaming range 6                0
8                0
9                0
10               0
over 10          0
3                1
7                1
4                2
```

```
5                  3
2                  4
1                 20
Not_Roaming      103
Name: roaming_range, dtype: int64
```

In [39]:
```
print("setprc Values")
print("Average setprc: ", df['setprc'].mean())
print("Minimum setprc: ", df['setprc'].min())
print("Maximum setprc: ", df['setprc'].max())
print("Null values: ", pd.isnull(df['setprc']).sum())
#check # 0s in age1
print("Number of SetPrc Zeroes: ",(df['setprc'] ==0).sum())
```

```
setprc Values
Average setprc:  26.862462686567135
Minimum setprc:  0.0
Maximum setprc:  199.99
Null values:  0
Number of SetPrc Zeroes:  91
```

In [40]:
```
print("Distinct values for setprc:\n", set(df['setprc']))
df['setprc'].value_counts(ascending=True)
```

```
Distinct values for setprc:
 {0.0, 129.99, 99.99, 39.99, 199.99, 9.99, 79.99, 149.99, 59.99, 29.99}
```

Out[40]:
```
129.99     1
39.99      2
199.99     2
9.99       4
99.99      4
59.99      5
79.99      6
29.99      9
149.99    10
0.00      91
Name: setprc, dtype: int64
```

In [41]:
```
df['age_group'] = df['age_group'].astype(str)
df['age_group2'] = df['age_group2'].astype(str)
df['roaming_range'] = df['roaming_range'].astype(str)

df.dtypes
```

Out[41]:
```
revenue            float64
mou                float64
recchrge           float64
directas           float64
```

```
overage          float64
                   ...
setprc           float64
retcall           object
age_group         object
age_group2        object
roaming_range     object
Length: 85, dtype: object
```

In [42]:
```
#did they churn
df['churn_status'] = df.churn.replace(to_replace=[0,1], value=['no','yes'])
```

In [43]:
```
df.dtypes
```

Out[43]:
```
revenue          float64
mou              float64
recchrge         float64
directas         float64
overage          float64
                   ...
retcall           object
age_group         object
age_group2        object
roaming_range     object
churn_status      object
Length: 86, dtype: object
```

In [44]:
```
#df.to_csv(r'celldata1_to_visualize.csv', index=False)
```

In [45]:
```
print("Object Columns:\n",list(df.select_dtypes(['object'])))
```

```
Object Columns:
 ['churn', 'csa', 'children', 'credita', 'creditaa', 'creditb', 'creditc', 'creditde', 'creditgy', 'creditz', 'credit_rat
ing', 'prizmrur', 'prizmub', 'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv', 'occprof', 'occcler', 'occcrft',
'occstud', 'occhmkr', 'occret', 'occself', 'occ', 'occ_label', 'ownrent', 'marryun', 'marryyes', 'marryno', 'marry', 'mar
ry_label', 'mailord', 'mailres', 'mailflag', 'travel', 'pcown', 'creditcd', 'newcelly', 'newcelln', 'incmiss', 'income',
'mcycle', 'creditad', 'setprcm', 'retcall', 'age_group', 'age_group2', 'roaming_range', 'churn_status']
```

In [46]:
```
print("Float Columns:\n",list(df.select_dtypes(['float64'])))
```

```
Float Columns:
 ['revenue', 'mou', 'recchrge', 'directas', 'overage', 'roam', 'changem', 'changer', 'dropvce', 'blckvce', 'unansvce', 'c
ustcare', 'threeway', 'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce', 'dropblk', 'callwait', 'age1', 'age2', 'set
prc']
```

In [47]:
```
print("Int Columns:\n",list(df.select_dtypes(['int64'])))
```

```
Int Columns:
```

```
['callfwdv', 'months', 'uniqsubs', 'actvsubs', 'phones', 'models', 'eqpdays', 'customer', 'retcalls', 'retaccpt', 'refe
r']
```

In [48]:

```python
#separate the data into object vs nonobjects
notif=df.select_dtypes(exclude=['int','float','int64'])
intfldtypes = df.select_dtypes(include=['int','float','int64'])
print(df.shape)
print(notif.shape)
print(intfldtypes.shape)
```

```
(134, 86)
(134, 52)
(134, 34)
```

In [49]:

```python
#label encode objects
obj_le= notif.apply(LabelEncoder().fit_transform)
#re-add with non-objects
df_ml= pd.concat([obj_le,intfldtypes], axis=1, sort=False)
df_ml=df_ml.drop(['churn_status'], axis=1)
#check shape
print(df_ml.shape)
```

```
(134, 85)
```

In [50]:

```python
#check correlation
#corr
print("Pearson's Correlation:")

with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df_ml.corr(method='pearson')['churn'].sort_values(ascending=False))
#corr[np.argsort(corr,axis=0)[::-1]]
```

```
Pearson's Correlation:
churn           1.000000
customer        0.424075
setprcm         0.206099
occret          0.186500
occstud         0.181058
creditz         0.174195
age_group       0.152528
occ_label       0.136923
creditc         0.135928
roam            0.135112
overage         0.131354
marry           0.121273
age_group2      0.099291
marryun         0.098048
occ             0.085445
```

credit_rating	0.078044
mailflag	0.057013
ownrent	0.056152
creditaa	0.040641
incmiss	0.036569
mailord	0.034136
mailres	0.026603
prizmub	0.020502
webcap	0.016021
children	0.004157
marryyes	0.001789
refurb	0.000882
changer	-0.007114
revenue	-0.009746
age2	-0.009921
pcown	-0.010549
prizmrur	-0.011060
income	-0.013418
przm_num	-0.022186
csa	-0.025300
outcalls	-0.036248
eqpdays	-0.037584
newcelln	-0.039823
creditad	-0.041527
mcycle	-0.041527
occcler	-0.041527
age1	-0.043118
threeway	-0.043320
directas	-0.044013
creditde	-0.048370
occprof	-0.048370
travel	-0.051973
mourec	-0.055399
callwait	-0.057724
custcare	-0.058329
occself	-0.058950
peakvce	-0.060489
changem	-0.068242
dropvce	-0.071463
credita	-0.072474
occcrft	-0.072474
creditgy	-0.072474
unansvce	-0.073098
blckvce	-0.073169
refer	-0.073432
creditcd	-0.089371
newcelly	-0.093076
dropblk	-0.098908
rv	-0.100952

```
recchrge        -0.102965
truck           -0.103159
uniqsubs        -0.105178
incalls         -0.105403
marryno         -0.110323
marry_label     -0.110323
roaming_range   -0.116318
prizmtwn        -0.116394
actvsubs        -0.118400
opeakvce        -0.120346
mou             -0.132489
creditb         -0.138096
setprc          -0.154269
phones          -0.179582
models          -0.183799
months          -0.355607
occhmkr               NaN
retcall               NaN
callfwdv              NaN
retcalls              NaN
retaccpt              NaN
Name: churn, dtype: float64
```

In [51]:

```
#check correlation
#corr
print("Spearman's Correlation:")
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df_ml.corr(method='spearman')['churn'].sort_values(ascending=False))
#corr[np.argsort(corr,axis=0)[::-1]]
```

```
Spearman's Correlation:
churn            1.000000
customer         0.437543
setprcm          0.206099
occret           0.186500
occstud          0.181058
creditz          0.174195
roam             0.143785
age_group        0.137135
creditc          0.135928
marry            0.120290
marryun          0.098048
overage          0.094028
age_group2       0.088486
occ_label        0.083749
mailflag         0.057013
ownrent          0.056152
creditaa         0.040641
threeway         0.040026
```

credit_rating	0.037234
incmiss	0.036569
mailord	0.034136
mailres	0.026603
occ	0.022535
blckvce	0.021562
prizmub	0.020502
webcap	0.016021
children	0.004157
marryyes	0.001789
refurb	0.000882
callwait	0.000800
income	-0.010271
pcown	-0.010549
prizmrur	-0.011060
age2	-0.018006
csa	-0.019567
przm_num	-0.019634
custcare	-0.028044
peakvce	-0.031203
age1	-0.032729
changem	-0.038877
newcelln	-0.039823
creditad	-0.041527
mcycle	-0.041527
occcler	-0.041527
revenue	-0.042839
directas	-0.048260
creditde	-0.048370
occprof	-0.048370
travel	-0.051973
unansvce	-0.057708
occself	-0.058950
outcalls	-0.064901
dropblk	-0.065169
creditgy	-0.072474
credita	-0.072474
occcrft	-0.072474
refer	-0.073432
mourec	-0.081513
eqpdays	-0.082210
recchrge	-0.083486
creditcd	-0.089371
newcelly	-0.093076
uniqsubs	-0.097675
dropvce	-0.100705
rv	-0.100952
truck	-0.103159
actvsubs	-0.109141

```
marryno         -0.110323
marry_label     -0.110323
changer         -0.114963
prizmtwn        -0.116394
mou             -0.120592
roaming_range   -0.129095
creditb         -0.138096
opeakvce        -0.146609
models          -0.193441
setprc          -0.196133
phones          -0.202638
incalls         -0.216802
months          -0.369625
occhmkr               NaN
retcall               NaN
callfwdv              NaN
retcalls              NaN
retaccpt              NaN
Name: churn, dtype: float64
```

In [52]:

```python
print("Kendall's Correlation")
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(df_ml.corr(method ='kendall')['churn'].sort_values(ascending=False))
```

```
Kendall's Correlation
churn            1.000000
customer         0.358583
setprcm          0.206099
occret           0.186500
occstud          0.181058
creditz          0.174195
roam             0.136084
creditc          0.135928
age_group        0.121771
marry            0.113600
marryun          0.098048
overage          0.081933
occ_label        0.081316
age_group2       0.080690
mailflag         0.057013
ownrent          0.056152
creditaa         0.040641
threeway         0.038890
incmiss          0.036569
mailord          0.034136
credit_rating    0.034074
mailres          0.026603
occ              0.021880
prizmub          0.020502
```

blckvce	0.018531
webcap	0.016021
children	0.004157
marryyes	0.001789
refurb	0.000882
callwait	0.000721
income	-0.008932
pcown	-0.010549
prizmrur	-0.011060
age2	-0.016027
csa	-0.016113
przm_num	-0.018843
custcare	-0.025333
peakvce	-0.025616
age1	-0.027825
changem	-0.031884
revenue	-0.035127
newcelln	-0.039823
mcycle	-0.041527
creditad	-0.041527
occcler	-0.041527
directas	-0.043799
unansvce	-0.047513
occprof	-0.048370
creditde	-0.048370
travel	-0.051973
outcalls	-0.053491
dropblk	-0.054024
occself	-0.058950
mourec	-0.067112
eqpdays	-0.067404
recchrge	-0.070351
credita	-0.072474
creditgy	-0.072474
occcrft	-0.072474
refer	-0.073432
dropvce	-0.084068
creditcd	-0.089371
newcelly	-0.093076
uniqsubs	-0.094535
changer	-0.094715
mou	-0.098873
rv	-0.100952
truck	-0.103159
actvsubs	-0.107462
marryno	-0.110323
marry_label	-0.110323
prizmtwn	-0.116394
opeakvce	-0.120427

```
roaming_range   -0.125094
creditb         -0.138096
setprc          -0.183382
incalls         -0.184641
models          -0.190311
phones          -0.195723
months          -0.313618
occhmkr               NaN
retcall               NaN
callfwdv              NaN
retcalls              NaN
retaccpt              NaN
Name: churn, dtype: float64
```

In [53]:

```python
df_ml.columns
```

Out[53]:

```
Index(['churn', 'csa', 'children', 'credita', 'creditaa', 'creditb', 'creditc',
       'creditde', 'creditgy', 'creditz', 'credit_rating', 'prizmrur',
       'prizmub', 'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv',
       'occprof', 'occcler', 'occcrft', 'occstud', 'occhmkr', 'occret',
       'occself', 'occ', 'occ_label', 'ownrent', 'marryun', 'marryyes',
       'marryno', 'marry', 'marry_label', 'mailord', 'mailres', 'mailflag',
       'travel', 'pcown', 'creditcd', 'newcelly', 'newcelln', 'incmiss',
       'income', 'mcycle', 'creditad', 'setprcm', 'retcall', 'age_group',
       'age_group2', 'roaming_range', 'revenue', 'mou', 'recchrge', 'directas',
       'overage', 'roam', 'changem', 'changer', 'dropvce', 'blckvce',
       'unansvce', 'custcare', 'threeway', 'mourec', 'outcalls', 'incalls',
       'peakvce', 'opeakvce', 'dropblk', 'callfwdv', 'callwait', 'months',
       'uniqsubs', 'actvsubs', 'phones', 'models', 'eqpdays', 'customer',
       'age1', 'age2', 'retcalls', 'retaccpt', 'refer', 'setprc'],
      dtype='object')
```

In [54]:

```python
class ChiSquare:
    def __init__(self, dataframe):
        self.df_ml = dataframe
        self.p = None #P-Value
        self.chi2 = None #Chi Test Statistic
        self.dof = None

        self.df_mlObserved = None
        self.df_mlExpected = None

    def _print_chisquare_result(self, colX, alpha):
        result = ""
        if self.p<alpha:
            result="\n~~~~The column {0} is IMPORTANT for Prediction.~~~~\n".format(colX)
        else:
```

```python
            result="The column {0} is NOT an important predictor.".format(colX)

        print(result)

    def TestIndependence(self,colX,colY, alpha=0.10):
        X = self.df_ml[colX].astype(str)
        Y = self.df_ml[colY].astype(str)

        self.df_mlObserved = pd.crosstab(Y,X)
        chi2, p, dof, expected = stats.chi2_contingency(self.df_mlObserved.values)
        self.p = p
        self.chi2 = chi2
        self.dof = dof

        self.df_mlExpected = pd.DataFrame(expected, columns=self.df_mlObserved.columns,
                                           index = self.df_mlObserved.index)

        self._print_chisquare_result(colX,alpha)

#Initialize ChiSquare Class
cT = ChiSquare(df_ml)

#Feature Selection
testColumns = ['age_group2','csa', 'occ_label', 'marry_label', 'age_group', 'roaming_range',
               'revenue', 'mou', 'recchrge', 'directas', 'overage', 'roam', 'changem',
               'changer', 'dropvce', 'blckvce', 'unansvce', 'custcare', 'threeway',
               'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce', 'dropblk',
               'callfwdv', 'callwait', 'months', 'uniqsubs', 'actvsubs',
               'phones', 'models', 'eqpdays', 'customer', 'age1','age2', 'children',
               'credita', 'creditaa', 'creditb', 'creditc', 'creditde', 'creditgy',
               'creditz', 'credit_rating', 'prizmrur', 'prizmub', 'prizmtwn',
               'przm_num', 'refurb', 'webcap', 'truck', 'rv', 'occprof', 'occcler',
               'occcrft', 'occstud', 'occhmkr', 'occret', 'occself', 'occ', 'ownrent',
               'marryun', 'marryyes', 'marryno', 'marry', 'mailord', 'mailres',
               'mailflag', 'travel', 'pcown', 'creditcd', 'retcalls', 'retaccpt',
               'newcelly', 'newcelln', 'refer', 'incmiss', 'income', 'mcycle',
               'creditad', 'setprcm', 'setprc', 'retcall']
for var in testColumns:
    cT.TestIndependence(colX=var,colY="churn" )
```

```
The column age_group2 is NOT an important predictor.

~~~~The column csa is IMPORTANT for Prediction.~~~~

The column occ_label is NOT an important predictor.
The column marry_label is NOT an important predictor.
```

```
The column age_group is NOT an important predictor.

~~~~The column roaming_range is IMPORTANT for Prediction.~~~~

The column revenue is NOT an important predictor.
The column mou is NOT an important predictor.
The column recchrge is NOT an important predictor.
The column directas is NOT an important predictor.
The column overage is NOT an important predictor.

~~~~The column roam is IMPORTANT for Prediction.~~~~

The column changem is NOT an important predictor.
The column changer is NOT an important predictor.
The column dropvce is NOT an important predictor.
The column blckvce is NOT an important predictor.
The column unansvce is NOT an important predictor.
The column custcare is NOT an important predictor.
The column threeway is NOT an important predictor.
The column mourec is NOT an important predictor.
The column outcalls is NOT an important predictor.
The column incalls is NOT an important predictor.
The column peakvce is NOT an important predictor.
The column opeakvce is NOT an important predictor.
The column dropblk is NOT an important predictor.
The column callfwdv is NOT an important predictor.
The column callwait is NOT an important predictor.

~~~~The column months is IMPORTANT for Prediction.~~~~

The column uniqsubs is NOT an important predictor.
The column actvsubs is NOT an important predictor.
The column phones is NOT an important predictor.
The column models is NOT an important predictor.
The column eqpdays is NOT an important predictor.
The column customer is NOT an important predictor.
The column age1 is NOT an important predictor.
The column age2 is NOT an important predictor.
The column children is NOT an important predictor.
The column credita is NOT an important predictor.
The column creditaa is NOT an important predictor.
The column creditb is NOT an important predictor.
The column creditc is NOT an important predictor.
The column creditde is NOT an important predictor.
The column creditgy is NOT an important predictor.
The column creditz is NOT an important predictor.
The column credit_rating is NOT an important predictor.
The column prizmrur is NOT an important predictor.
The column prizmub is NOT an important predictor.
```

```
The column prizmtwn is NOT an important predictor.
The column przm_num is NOT an important predictor.
The column refurb is NOT an important predictor.
The column webcap is NOT an important predictor.
The column truck is NOT an important predictor.
The column rv is NOT an important predictor.
The column occprof is NOT an important predictor.
The column occcler is NOT an important predictor.
The column occcrft is NOT an important predictor.
The column occstud is NOT an important predictor.
The column occhmkr is NOT an important predictor.
The column occret is NOT an important predictor.
The column occself is NOT an important predictor.
The column occ is NOT an important predictor.
The column ownrent is NOT an important predictor.
The column marryun is NOT an important predictor.
The column marryyes is NOT an important predictor.
The column marryno is NOT an important predictor.
The column marry is NOT an important predictor.
The column mailord is NOT an important predictor.
The column mailres is NOT an important predictor.
The column mailflag is NOT an important predictor.
The column travel is NOT an important predictor.
The column pcown is NOT an important predictor.
The column creditcd is NOT an important predictor.
The column retcalls is NOT an important predictor.
The column retaccpt is NOT an important predictor.
The column newcelly is NOT an important predictor.
The column newcelln is NOT an important predictor.
The column refer is NOT an important predictor.
The column incmiss is NOT an important predictor.
The column income is NOT an important predictor.
The column mcycle is NOT an important predictor.
The column creditad is NOT an important predictor.

~~~~The column setprcm is IMPORTANT for Prediction.~~~~

The column setprc is NOT an important predictor.
The column retcall is NOT an important predictor.
```

In [55]:
```
df = df_ml
```

In [56]:
```
#new columns
"""
We will be making new columns out of the important columns from the Chi-Squared test above.

The important columns are as follows:
                            months, creditgy, przm_num, refurb, webcap, mailord, and travel.
```

```python
Some will be columns that I think would match well with the important column
and others will be a combination of important columns.

"""

#months
df['months_mou'] = df['months'].astype(str) + '_' + df['mou'].astype(str)
df['months_creditgy'] = df['months'].astype(str) + '_' + df['creditgy'].astype(str)
df['months_przm_num'] = df['months'].astype(str) + '_' + df['przm_num'].astype(str)
df['months_refurb'] = df['months'].astype(str) + '_' + df['refurb'].astype(str)
df['months_webcap'] = df['months'].astype(str) + '_' + df['webcap'].astype(str)
df['months_mailord'] = df['months'].astype(str) + '_' + df['mailord'].astype(str)
df['months_travel'] = df['months'].astype(str) + '_' + df['travel'].astype(str)
df['months_models'] = df['months'].astype(str) + '_' + df['models'].astype(str)
df['months_agegroup'] = df['months'].astype(str) + '_' + df['age_group'].astype(str)
df['months_agegroup2'] = df['months'].astype(str) + '_' + df['age_group2'].astype(str)

#creditgy
df['creditgy_przm_num'] = df['creditgy'].astype(str) + '_' + df['przm_num'].astype(str)
df['creditgy_refurb'] = df['creditgy'].astype(str) + '_' + df['refurb'].astype(str)
df['creditgy_webcap'] = df['creditgy'].astype(str) + '_' + df['webcap'].astype(str)
df['creditgy_mailord'] = df['creditgy'].astype(str) + '_' + df['mailord'].astype(str)
df['creditgy_travel'] = df['creditgy'].astype(str) + '_' + df['travel'].astype(str)
df['creditgy_income'] = df['creditgy'].astype(str) + '_' + df['income'].astype(str)
df['creditgy_agegroup'] = df['creditgy'].astype(str) + '_' + df['age_group'].astype(str)
df['creditgy_agegroup2'] = df['creditgy'].astype(str) + '_' + df['age_group2'].astype(str)
df['creditgy_occ'] = df['creditgy'].astype(str) + '_' + df['occ'].astype(str)

#przm_num
df['przm_num_refurb'] = df['przm_num'].astype(str) + '_' + df['refurb'].astype(str)
df['przm_num_webcap'] = df['przm_num'].astype(str) + '_' + df['webcap'].astype(str)
df['przm_num_mailord'] = df['przm_num'].astype(str) + '_' + df['mailord'].astype(str)
df['przm_num_travel'] = df['przm_num'].astype(str) + '_' + df['travel'].astype(str)
df['przm_num_dropblk'] = df['przm_num'].astype(str) + '_' + df['dropblk'].astype(str)
df['przm_num_dropvce'] = df['przm_num'].astype(str) + '_' + df['dropvce'].astype(str)
df['przm_num_roam_range'] = df['przm_num'].astype(str) + '_' + df['roaming_range'].astype(str)


#refurb
df['refurb_webcap'] = df['refurb'].astype(str) + '_' + df['webcap'].astype(str)
df['refurb_mailord'] = df['refurb'].astype(str) + '_' + df['mailord'].astype(str)
df['refurb_travel'] = df['refurb'].astype(str) + '_' + df['travel'].astype(str)
df['refurb_models'] = df['refurb'].astype(str) + '_' + df['models'].astype(str)
df['refurb_dropblk'] = df['refurb'].astype(str) + '_' + df['dropblk'].astype(str)
df['refurb_dropvce'] = df['refurb'].astype(str) + '_' + df['dropvce'].astype(str)
```

```python
df['refurb_custcare'] = df['refurb'].astype(str) + '_' + df['custcare'].astype(str)
df['refurb_retcalls'] = df['refurb'].astype(str) + '_' + df['retcalls'].astype(str)
df['refurb_retcall'] = df['refurb'].astype(str) + '_' + df['retcall'].astype(str)

#webcap
df['webcap_mailord'] = df['webcap'].astype(str) + '_' + df['mailord'].astype(str)
df['webcap_travel'] = df['webcap'].astype(str) + '_' + df['travel'].astype(str)
df['webcap_agegroup'] = df['webcap'].astype(str) + '_' + df['age_group'].astype(str)
df['webcap_agegroup2'] = df['webcap'].astype(str) + '_' + df['age_group2'].astype(str)
df['webcap_income'] = df['webcap'].astype(str) + '_' + df['income'].astype(str)
df['webcap_setprc'] = df['webcap'].astype(str) + '_' + df['setprc'].astype(str)
df['webcap_retcall'] = df['webcap'].astype(str) + '_' + df['retcall'].astype(str)

#mailord
df['mailord_travel'] = df['mailord'].astype(str) + '_' + df['travel'].astype(str)
df['mailord_mailres'] = df['mailord'].astype(str) + '_' + df['mailres'].astype(str)
df['mailord_mailflag'] = df['mailord'].astype(str) + '_' + df['mailflag'].astype(str)
df['mailord_agegroup'] = df['mailord'].astype(str) + '_' + df['age_group'].astype(str)
df['mailord_agegroup2'] = df['mailord'].astype(str) + '_' + df['age_group2'].astype(str)

#travel
df['travel_roaming_range'] =df['travel'].astype(str) + '_' + df['roaming_range'].astype(str)
df['travel_income'] =df['travel'].astype(str) + '_' + df['income'].astype(str)
df['travel_occ'] =df['travel'].astype(str) + '_' + df['occ'].astype(str)
df['travel_marry'] =df['travel'].astype(str) + '_' + df['marry'].astype(str)
```

In [57]:

```python
#re_separate the data into object vs nonobjects
notif2=df.select_dtypes(exclude=['int','float','int64'])
intfldtypes2 = df.select_dtypes(include=['int','float','int64'])
print(df.shape)
print(notif2.shape)
print(intfldtypes2.shape)
```

```
(134, 136)
(134, 51)
(134, 85)
```

In [58]:

```python
#label encode objects
obj_le2= notif2.apply(LabelEncoder().fit_transform)
#re-add with non-objects
df_ml2= pd.concat([obj_le2,intfldtypes2], axis=1, sort=False)
#df_ml2=df_ml2.drop(['churn_status'], axis=1)
#check shape
print(df_ml2.shape)
```

```
(134, 136)
```

In [59]:
```python
pd.options.display.max_columns = None
pd.options.display.max_rows = None
print(df_ml2.columns.tolist())
```

```
['months_mou', 'months_creditgy', 'months_przm_num', 'months_refurb', 'months_webcap', 'months_mailord', 'months_travel', 'months_models', 'months_agegroup', 'months_agegroup2', 'creditgy_przm_num', 'creditgy_refurb', 'creditgy_webcap', 'creditgy_mailord', 'creditgy_travel', 'creditgy_income', 'creditgy_agegroup', 'creditgy_agegroup2', 'creditgy_occ', 'przm_num_refurb', 'przm_num_webcap', 'przm_num_mailord', 'przm_num_travel', 'przm_num_dropblk', 'przm_num_dropvce', 'przm_num_roam_range', 'refurb_webcap', 'refurb_mailord', 'refurb_travel', 'refurb_models', 'refurb_dropblk', 'refurb_dropvce', 'refurb_custcare', 'refurb_retcalls', 'refurb_retcall', 'webcap_mailord', 'webcap_travel', 'webcap_agegroup', 'webcap_agegroup2', 'webcap_income', 'webcap_setprc', 'webcap_retcall', 'mailord_travel', 'mailord_mailres', 'mailord_mailflag', 'mailord_agegroup', 'mailord_agegroup2', 'travel_roaming_range', 'travel_income', 'travel_occ', 'travel_marry', 'churn', 'csa', 'children', 'credita', 'creditaa', 'creditb', 'creditc', 'creditde', 'creditgy', 'creditz', 'credit_rating', 'prizmrur', 'prizmub', 'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv', 'occprof', 'occcler', 'occcrft', 'occstud', 'occhmkr', 'occret', 'occself', 'occ', 'occ_label', 'ownrent', 'marryun', 'marryyes', 'marryno', 'marry', 'marry_label', 'mailord', 'mailres', 'mailflag', 'travel', 'pcown', 'creditcd', 'newcelly', 'newcelln', 'incmiss', 'income', 'mcycle', 'creditad', 'setprcm', 'retcall', 'age_group', 'age_group2', 'roaming_range', 'revenue', 'mou', 'recchrge', 'directas', 'overage', 'roam', 'changem', 'changer', 'dropvce', 'blckvce', 'unansvce', 'custcare', 'threeway', 'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce', 'dropblk', 'callfwdv', 'callwait', 'months', 'uniqsubs', 'actvsubs', 'phones', 'models', 'eqpdays', 'customer', 'age1', 'age2', 'retcalls', 'retaccpt', 'refer', 'setprc']
```

In [60]:
```python
class ChiSquare:
    def __init__(self, dataframe):
        self.df_ml2 = dataframe
        self.p = None #P-Value
        self.chi2 = None #Chi Test Statistic
        self.dof = None

        self.df_ml2Observed = None
        self.df_ml2Expected = None

    def _print_chisquare_result(self, colX, alpha):
        result = ""
        if self.p<alpha:
            result="\n~~~~The column {0} is IMPORTANT for Prediction.~~~~\n".format(colX)
        else:
            result="The column {0} is NOT an important predictor.".format(colX)

        print(result)

    def TestIndependence(self,colX,colY, alpha=0.10):
        X = self.df_ml2[colX].astype(str)
        Y = self.df_ml2[colY].astype(str)

        self.df_ml2Observed = pd.crosstab(Y,X)
        chi2, p, dof, expected = stats.chi2_contingency(self.df_ml2Observed.values)
        self.p = p
```

```python
        self.chi2 = chi2
        self.dof = dof

        self.df_ml2Expected = pd.DataFrame(expected, columns=self.df_ml2Observed.columns,
                                           index = self.df_ml2Observed.index)

        self._print_chisquare_result(colX,alpha)

#Initialize ChiSquare Class
cT = ChiSquare(df_ml2)

#Feature Selection
testColumns = ['months_mou', 'months_creditgy', 'months_przm_num', 'months_refurb', 'months_webcap',
               'months_mailord', 'months_travel', 'months_models', 'months_agegroup',
               'months_agegroup2', 'creditgy_przm_num', 'creditgy_refurb', 'creditgy_webcap',
               'creditgy_mailord', 'creditgy_travel', 'creditgy_income', 'creditgy_agegroup',
               'creditgy_agegroup2', 'creditgy_occ', 'przm_num_refurb', 'przm_num_webcap',
               'przm_num_mailord', 'przm_num_travel', 'przm_num_dropblk', 'przm_num_dropvce',
               'przm_num_roam_range', 'refurb_webcap', 'refurb_mailord', 'refurb_travel',
               'refurb_models', 'refurb_dropblk', 'refurb_dropvce', 'refurb_custcare',
               'refurb_retcalls', 'refurb_retcall', 'webcap_mailord', 'webcap_travel',
               'webcap_agegroup', 'webcap_agegroup2', 'webcap_income', 'webcap_setprc',
               'webcap_retcall', 'mailord_travel', 'mailord_mailres', 'mailord_mailflag',
               'mailord_agegroup', 'mailord_agegroup2', 'travel_roaming_range', 'travel_income',
               'travel_occ', 'travel_marry', 'csa', 'children', 'credita', 'creditaa', 'creditb',
               'creditc', 'creditde', 'creditgy', 'creditz', 'credit_rating', 'prizmrur', 'prizmub',
               'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv', 'occprof', 'occcler',
               'occcrft', 'occstud', 'occhmkr', 'occret', 'occself', 'occ', 'occ_label', 'ownrent',
               'marryun', 'marryyes', 'marryno', 'marry', 'marry_label', 'mailord', 'mailres',
               'mailflag', 'travel', 'pcown', 'creditcd', 'newcelly', 'newcelln', 'incmiss',
               'income', 'mcycle', 'creditad', 'setprcm', 'retcall', 'age_group',
               'age_group2', 'roaming_range', 'revenue', 'mou', 'recchrge', 'directas',
               'overage', 'roam', 'changem', 'changer', 'dropvce', 'blckvce', 'unansvce',
               'custcare', 'threeway', 'mourec', 'outcalls', 'incalls', 'peakvce', 'opeakvce',
               'dropblk', 'callfwdv', 'callwait', 'months', 'uniqsubs', 'actvsubs', 'phones',
               'models', 'eqpdays', 'customer', 'age1', 'age2', 'retcalls', 'retaccpt', 'refer',
               'setprc']
for var in testColumns:
    cT.TestIndependence(colX=var,colY="churn" )
```

```
The column months_mou is NOT an important predictor.
The column months_creditgy is NOT an important predictor.
The column months_przm_num is NOT an important predictor.
The column months_refurb is NOT an important predictor.
The column months_webcap is NOT an important predictor.
The column months_mailord is NOT an important predictor.
```

```
The column months_travel is NOT an important predictor.
The column months_models is NOT an important predictor.
The column months_agegroup is NOT an important predictor.
The column months_agegroup2 is NOT an important predictor.
The column creditgy_przm_num is NOT an important predictor.
The column creditgy_refurb is NOT an important predictor.
The column creditgy_webcap is NOT an important predictor.
The column creditgy_mailord is NOT an important predictor.
The column creditgy_travel is NOT an important predictor.
The column creditgy_income is NOT an important predictor.
The column creditgy_agegroup is NOT an important predictor.
The column creditgy_agegroup2 is NOT an important predictor.
The column creditgy_occ is NOT an important predictor.
The column przm_num_refurb is NOT an important predictor.
The column przm_num_webcap is NOT an important predictor.
The column przm_num_mailord is NOT an important predictor.
The column przm_num_travel is NOT an important predictor.
The column przm_num_dropblk is NOT an important predictor.
The column przm_num_dropvce is NOT an important predictor.

~~~~The column przm_num_roam_range is IMPORTANT for Prediction.~~~~

The column refurb_webcap is NOT an important predictor.
The column refurb_mailord is NOT an important predictor.
The column refurb_travel is NOT an important predictor.
The column refurb_models is NOT an important predictor.
The column refurb_dropblk is NOT an important predictor.
The column refurb_dropvce is NOT an important predictor.
The column refurb_custcare is NOT an important predictor.
The column refurb_retcalls is NOT an important predictor.
The column refurb_retcall is NOT an important predictor.
The column webcap_mailord is NOT an important predictor.
The column webcap_travel is NOT an important predictor.
The column webcap_agegroup is NOT an important predictor.
The column webcap_agegroup2 is NOT an important predictor.
The column webcap_income is NOT an important predictor.
The column webcap_setprc is NOT an important predictor.
The column webcap_retcall is NOT an important predictor.
The column mailord_travel is NOT an important predictor.
The column mailord_mailres is NOT an important predictor.
The column mailord_mailflag is NOT an important predictor.
The column mailord_agegroup is NOT an important predictor.
The column mailord_agegroup2 is NOT an important predictor.

~~~~The column travel_roaming_range is IMPORTANT for Prediction.~~~~

The column travel_income is NOT an important predictor.
The column travel_occ is NOT an important predictor.
The column travel_marry is NOT an important predictor.
```

```
~~~~The column csa is IMPORTANT for Prediction.~~~~

The column children is NOT an important predictor.
The column credita is NOT an important predictor.
The column creditaa is NOT an important predictor.
The column creditb is NOT an important predictor.
The column creditc is NOT an important predictor.
The column creditde is NOT an important predictor.
The column creditgy is NOT an important predictor.
The column creditz is NOT an important predictor.
The column credit_rating is NOT an important predictor.
The column prizmrur is NOT an important predictor.
The column prizmub is NOT an important predictor.
The column prizmtwn is NOT an important predictor.
The column przm_num is NOT an important predictor.
The column refurb is NOT an important predictor.
The column webcap is NOT an important predictor.
The column truck is NOT an important predictor.
The column rv is NOT an important predictor.
The column occprof is NOT an important predictor.
The column occcler is NOT an important predictor.
The column occcrft is NOT an important predictor.
The column occstud is NOT an important predictor.
The column occhmkr is NOT an important predictor.
The column occret is NOT an important predictor.
The column occself is NOT an important predictor.
The column occ is NOT an important predictor.
The column occ_label is NOT an important predictor.
The column ownrent is NOT an important predictor.
The column marryun is NOT an important predictor.
The column marryyes is NOT an important predictor.
The column marryno is NOT an important predictor.
The column marry is NOT an important predictor.
The column marry_label is NOT an important predictor.
The column mailord is NOT an important predictor.
The column mailres is NOT an important predictor.
The column mailflag is NOT an important predictor.
The column travel is NOT an important predictor.
The column pcown is NOT an important predictor.
The column creditcd is NOT an important predictor.
The column newcelly is NOT an important predictor.
The column newcelln is NOT an important predictor.
The column incmiss is NOT an important predictor.
The column income is NOT an important predictor.
The column mcycle is NOT an important predictor.
The column creditad is NOT an important predictor.

~~~~The column setprcm is IMPORTANT for Prediction.~~~~
```

```
The column retcall is NOT an important predictor.
The column age_group is NOT an important predictor.
The column age_group2 is NOT an important predictor.

~~~~The column roaming_range is IMPORTANT for Prediction.~~~~

The column revenue is NOT an important predictor.
The column mou is NOT an important predictor.
The column recchrge is NOT an important predictor.
The column directas is NOT an important predictor.
The column overage is NOT an important predictor.

~~~~The column roam is IMPORTANT for Prediction.~~~~

The column changem is NOT an important predictor.
The column changer is NOT an important predictor.
The column dropvce is NOT an important predictor.
The column blckvce is NOT an important predictor.
The column unansvce is NOT an important predictor.
The column custcare is NOT an important predictor.
The column threeway is NOT an important predictor.
The column mourec is NOT an important predictor.
The column outcalls is NOT an important predictor.
The column incalls is NOT an important predictor.
The column peakvce is NOT an important predictor.
The column opeakvce is NOT an important predictor.
The column dropblk is NOT an important predictor.
The column callfwdv is NOT an important predictor.
The column callwait is NOT an important predictor.

~~~~The column months is IMPORTANT for Prediction.~~~~

The column uniqsubs is NOT an important predictor.
The column actvsubs is NOT an important predictor.
The column phones is NOT an important predictor.
The column models is NOT an important predictor.
The column eqpdays is NOT an important predictor.
The column customer is NOT an important predictor.
The column age1 is NOT an important predictor.
The column age2 is NOT an important predictor.
The column retcalls is NOT an important predictor.
The column retaccpt is NOT an important predictor.
The column refer is NOT an important predictor.
The column setprc is NOT an important predictor.
```

In [61]:

```
df1 = df[['age_group', 'months_creditgy', 'months_przm_num', 'months_refurb', 'months_webcap',
          'months_models', 'creditgy_przm_num', 'creditgy_refurb', 'creditgy_webcap',
          'creditgy_mailord', 'creditgy_travel', 'przm_num_refurb', 'przm_num_webcap',
```

```python
            'przm_num_mailord', 'refurb_webcap', 'refurb_mailord', 'refurb_travel',
            'refurb_models', 'refurb_retcalls', 'refurb_retcall', 'webcap_mailord',
            'webcap_travel', 'webcap_setprc', 'webcap_retcall', 'mailord_mailres',
            'creditgy', 'przm_num', 'refurb', 'webcap', 'mailord', 'travel',
            'months','churn']]
```

In [62]:

```python
#df1.to_csv(r'celldata_to_visualize.csv', index=False)
```

In [63]:

```python
#separate dtypes
notif=df.select_dtypes(exclude=['int','float','int64'])
intfldtypes = df.select_dtypes(include=['int','float','int64'])
print('Objects',notif.columns)
print("\nNonObjects",intfldtypes.columns)

#checking to make sure all are accounted for
print(df.shape)
print(notif.shape)
print(intfldtypes.shape)
```

```
Objects Index(['months_mou', 'months_creditgy', 'months_przm_num', 'months_refurb',
       'months_webcap', 'months_mailord', 'months_travel', 'months_models',
       'months_agegroup', 'months_agegroup2', 'creditgy_przm_num',
       'creditgy_refurb', 'creditgy_webcap', 'creditgy_mailord',
       'creditgy_travel', 'creditgy_income', 'creditgy_agegroup',
       'creditgy_agegroup2', 'creditgy_occ', 'przm_num_refurb',
       'przm_num_webcap', 'przm_num_mailord', 'przm_num_travel',
       'przm_num_dropblk', 'przm_num_dropvce', 'przm_num_roam_range',
       'refurb_webcap', 'refurb_mailord', 'refurb_travel', 'refurb_models',
       'refurb_dropblk', 'refurb_dropvce', 'refurb_custcare',
       'refurb_retcalls', 'refurb_retcall', 'webcap_mailord', 'webcap_travel',
       'webcap_agegroup', 'webcap_agegroup2', 'webcap_income', 'webcap_setprc',
       'webcap_retcall', 'mailord_travel', 'mailord_mailres',
       'mailord_mailflag', 'mailord_agegroup', 'mailord_agegroup2',
       'travel_roaming_range', 'travel_income', 'travel_occ', 'travel_marry'],
      dtype='object')

NonObjects Index(['churn', 'csa', 'children', 'credita', 'creditaa', 'creditb', 'creditc',
       'creditde', 'creditgy', 'creditz', 'credit_rating', 'prizmrur',
       'prizmub', 'prizmtwn', 'przm_num', 'refurb', 'webcap', 'truck', 'rv',
       'occprof', 'occcler', 'occcrft', 'occstud', 'occhmkr', 'occret',
       'occself', 'occ', 'occ_label', 'ownrent', 'marryun', 'marryyes',
       'marryno', 'marry', 'marry_label', 'mailord', 'mailres', 'mailflag',
       'travel', 'pcown', 'creditcd', 'newcelly', 'newcelln', 'incmiss',
       'income', 'mcycle', 'creditad', 'setprcm', 'retcall', 'age_group',
       'age_group2', 'roaming_range', 'revenue', 'mou', 'recchrge', 'directas',
       'overage', 'roam', 'changem', 'changer', 'dropvce', 'blckvce',
       'unansvce', 'custcare', 'threeway', 'mourec', 'outcalls', 'incalls',
```

```
       'peakvce', 'opeakvce', 'dropblk', 'callfwdv', 'callwait', 'months',
       'uniqsubs', 'actvsubs', 'phones', 'models', 'eqpdays', 'customer',
       'age1', 'age2', 'retcalls', 'retaccpt', 'refer', 'setprc'],
      dtype='object')
(134, 136)
(134, 51)
(134, 85)
```

In [64]:

```
#label encode objects
obj_le= notif.apply(LabelEncoder().fit_transform)
#re-add with non-objects
df_pred= pd.concat([obj_le,intfldtypes], axis=1, sort=False)
#check shape
print(df_pred.shape)
```

```
(134, 136)
```

In [65]:

```
df_pred.churn.value_counts(ascending=True)
```

Out[65]:

```
1     25
0    109
Name: churn, dtype: int64
```

In [66]:

```
df_pred.to_csv(r'verificationdataset.csv',index=False)
```