Telecommunication Project - Resume

February 8, 2024

Analyzed the dataset of an European Telecom Company.

This Telecom's Churn Dataset, consists of cleaned customer activity data (features), along with a churn label specifying whether a customer cancelled the subscription or not.

Analyzed the data to discover key factors responsible for customer churn and come up with ways/recommendations to ensure customer retention.

1 Business Understanding Of A Telecom Industry Customer Churn:

Customer churn is a big problem in any industry and one of the most important concerns for the Telecom industry.

The effect on the revenues of the companies, because of this customer churns is huge, especially in the telecom field, that's why these companies are seeking to develop a predictive potential customer churn.

In this highly competitive market, the telecommunications industry experiences an average of 15-25% annual churn rate, and it costs 5-10 times more to acquire a new customer than to retain an existing one, that's why customer retention has now become even more important than customer acquisition.

Therefore, finding those factors that increase customer churn is important to take necessary actions to reduce this churn.

The main goal of this project is to develop an understanding of the cause of customer churn which assists telecom operators to predict customers who are most likely subject to churn, and what to do to retain the most valuable customer.

1.0.1 I will find how I can maximize the profit by retaining customer, and, how I can reduce the churn rate by identifying the issues.

```
[2]: # Importing the required libraries

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[3]: # importing the dataset

data = pd.read_csv("Telecom Dataset.csv")
```

---

2 Exploring the Dataset

```
[3]: data                    # To have a look at our dataset ; it will show Top5 &␣
       ↪Bottom5 rows, and about 20 columns
```

```
[3]:      state  account length  area code phone number international plan  \
     0       KS             128        415     382-4657                 no
     1       OH             107        415     371-7191                 no
     2       NJ             137        415     358-1921                 no
     3       OH              84        408     375-9999                yes
     4       OK              75        415     330-6626                yes
     …      …              …        …          …                  …
     3328    AZ             192        415     414-4276                 no
     3329    WV              68        415     370-3271                 no
     3330    RI              28        510     328-8230                 no
     3331    CT             184        510     364-6381                yes
     3332    TN              74        415     400-4344                 no

          voice mail plan  number vmail messages  total day minutes  \
     0                yes                     25              265.1
     1                yes                     26              161.6
     2                 no                      0              243.4
     3                 no                      0              299.4
     4                 no                      0              166.7
     …              …                    …                 …
     3328             yes                     36              156.2
     3329              no                      0              231.1
     3330              no                      0              180.8
     3331              no                      0              213.8
     3332             yes                     25              234.4

           total day calls  total day charge  …  total eve calls  \
     0                 110             45.07  …               99
     1                 123             27.47  …              103
     2                 114             41.38  …              110
```

```
3                  71            50.90  …             88
4                 113            28.34  …            122
…                 …               …  …             …
3328               77            26.55  …            126
3329               57            39.29  …             55
3330              109            30.74  …             58
3331              105            36.35  …             84
3332              113            39.85  …             82

      total eve charge  total night minutes  total night calls  \
0                16.78                244.7                 91
1                16.62                254.4                103
2                10.30                162.6                104
3                 5.26                196.9                 89
4                12.61                186.9                121
…                  …                   …                 …
3328             18.32                279.1                 83
3329             13.04                191.3                123
3330             24.55                191.9                 91
3331             13.57                139.2                137
3332             22.60                241.4                 77

      total night charge  total intl minutes  total intl calls  \
0                  11.01                10.0                 3
1                  11.45                13.7                 3
2                   7.32                12.2                 5
3                   8.86                 6.6                 7
4                   8.41                10.1                 3
…                    …                  …                …
3328               12.56                 9.9                 6
3329                8.61                 9.6                 4
3330                8.64                14.1                 6
3331                6.26                 5.0                10
3332               10.86                13.7                 4

      total intl charge  customer service calls  churn
0                  2.70                       1  False
1                  3.70                       1  False
2                  3.29                       0  False
3                  1.78                       2  False
4                  2.73                       3  False
…                  …                        …    …
3328               2.67                       2  False
3329               2.59                       3  False
3330               3.81                       2  False
3331               1.35                       2  False
3332               3.70                       0  False
```

```
[3333 rows x 21 columns]
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3333 non-null   object
 1   account length          3333 non-null   int64
 2   area code               3333 non-null   int64
 3   phone number            3333 non-null   object
 4   international plan      3333 non-null   object
 5   voice mail plan         3333 non-null   object
 6   number vmail messages   3333 non-null   int64
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64
 20  churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

[7]:
```
data.shape                  # To show the number of Rows & Columns in the DataFrame
```

[7]:
```
(3333, 21)
```

[8]:
```
data.nunique()              # To show the number of unique values in each column
```

[8]:
```
state                       51
account length             212
area code                    3
phone number              3333
international plan           2
voice mail plan              2
number vmail messages       46
total day minutes         1667
total day calls            119
total day charge          1667
total eve minutes         1611
total eve calls            123
total eve charge          1440
total night minutes       1591
total night calls          120
total night charge         933
total intl minutes         162
total intl calls            21
total intl charge          162
customer service calls      10
churn                        2
dtype: int64
```

[9]:
```
data.columns              # To show all the column names of the DataFrame
```

[9]:
```
Index(['state', 'account length', 'area code', 'phone number',
       'international plan', 'voice mail plan', 'number vmail messages',
       'total day minutes', 'total day calls', 'total day charge',
       'total eve minutes', 'total eve calls', 'total eve charge',
       'total night minutes', 'total night calls', 'total night charge',
       'total intl minutes', 'total intl calls', 'total intl charge',
       'customer service calls', 'churn'],
      dtype='object')
```

[10]:
```
data.dtypes                # To show the data-type of each column
```

[10]:
```
state                      object
account length              int64
area code                   int64
phone number               object
international plan         object
voice mail plan            object
number vmail messages       int64
total day minutes         float64
total day calls             int64
total day charge          float64
total eve minutes         float64
total eve calls             int64
total eve charge          float64
total night minutes       float64
total night calls           int64
total night charge        float64
total intl minutes        float64
total intl calls            int64
total intl charge         float64
```

```
customer service calls      int64
churn                        bool
dtype: object
```

[5]:
```
data.describe()
```

[5]:
```
       account length    area code  number vmail messages  total day minutes  \
count     3333.000000  3333.000000            3333.000000        3333.000000
mean       101.064806   437.182418               8.099010         179.775098
std         39.822106    42.371290              13.688365          54.467389
min          1.000000   408.000000               0.000000           0.000000
25%         74.000000   408.000000               0.000000         143.700000
50%        101.000000   415.000000               0.000000         179.400000
75%        127.000000   510.000000              20.000000         216.400000
max        243.000000   510.000000              51.000000         350.800000

       total day calls  total day charge  total eve minutes  total eve calls  \
count      3333.000000       3333.000000        3333.000000      3333.000000
mean        100.435644         30.562307         200.980348       100.114311
std          20.069084          9.259435          50.713844        19.922625
min           0.000000          0.000000           0.000000         0.000000
25%          87.000000         24.430000         166.600000        87.000000
50%         101.000000         30.500000         201.400000       100.000000
75%         114.000000         36.790000         235.300000       114.000000
max         165.000000         59.640000         363.700000       170.000000

       total eve charge  total night minutes  total night calls  \
count       3333.000000          3333.000000        3333.000000
mean          17.083540           200.872037         100.107711
std            4.310668            50.573847          19.568609
min            0.000000            23.200000          33.000000
25%           14.160000           167.000000          87.000000
50%           17.120000           201.200000         100.000000
75%           20.000000           235.300000         113.000000
max           30.910000           395.000000         175.000000

       total night charge  total intl minutes  total intl calls  \
count         3333.000000         3333.000000       3333.000000
mean             9.039325           10.237294          4.479448
std              2.275873            2.791840          2.461214
min              1.040000            0.000000          0.000000
25%              7.520000            8.500000          3.000000
50%              9.050000           10.300000          4.000000
75%             10.590000           12.100000          6.000000
max             17.770000           20.000000         20.000000

       total intl charge  customer service calls
```

```
count        3333.000000          3333.000000
mean            2.764581             1.562856
std             0.753773             1.315491
min             0.000000             0.000000
25%             2.300000             1.000000
50%             2.780000             1.000000
75%             3.270000             2.000000
max             5.400000             9.000000
```

[12]:
```
data.describe(include='object')       # It gives the summary of all categorical␣
 ↪columns
# It shows the count of non-null and unique values in each column, Top value␣
 ↪with its occurance in each column
```

[12]:
```
       state phone number international plan voice mail plan
count   3333         3333               3333            3333
unique    51         3333                  2               2
top       WV     341-9443                 no              no
freq     106            1               3010            2411
```

3 Checking Missing and Duplicate Values

[13]:
```
data.head(2)               # To show the Top2 records of the DataFrame
```

[13]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3                2.7
1                13.7                 3                3.7
```

```
   customer service calls  churn
0                       1  False
1                       1  False

[2 rows x 21 columns]
```

isna( )

```
[14]: data.isna().sum()          # To show the count of missing (null) values in␣
       ↪each column
```

```
[14]: state                     0
      account length            0
      area code                 0
      phone number              0
      international plan        0
      voice mail plan           0
      number vmail messages     0
      total day minutes         0
      total day calls           0
      total day charge          0
      total eve minutes         0
      total eve calls           0
      total eve charge          0
      total night minutes       0
      total night calls         0
      total night charge        0
      total intl minutes        0
      total intl calls          0
      total intl charge         0
      customer service calls    0
      churn                     0
      dtype: int64
```

isnull( )

```
[15]: data.isnull().sum()          # Alternatively, To show the count of missing␣
       ↪(null) values in each column
```

```
[15]: state                     0
      account length            0
      area code                 0
      phone number              0
      international plan        0
      voice mail plan           0
      number vmail messages     0
      total day minutes         0
      total day calls           0
      total day charge          0
      total eve minutes         0
      total eve calls           0
      total eve charge          0
      total night minutes       0
      total night calls         0
      total night charge        0
      total intl minutes        0
      total intl calls          0
      total intl charge         0
      customer service calls    0
      churn                     0
      dtype: int64
```

notnull( )

```
[16]: data.notnull().sum()          # To show the count of non-null values in each␣
        ↪column
```

```
[16]: state                     3333
      account length            3333
      area code                 3333
      phone number              3333
      international plan        3333
      voice mail plan           3333
      number vmail messages     3333
      total day minutes         3333
      total day calls           3333
      total day charge          3333
      total eve minutes         3333
      total eve calls           3333
      total eve charge          3333
      total night minutes       3333
      total night calls         3333
      total night charge        3333
      total intl minutes        3333
      total intl calls          3333
      total intl charge         3333
      customer service calls    3333
      churn                     3333
      dtype: int64
```

notna( )

```
[18]: data[data.duplicated()]          # To show the duplicate records present in the␣
        ↪DataFrame
```

```
                                   # In this dataframe, no duplicate records␣
 ↪present
```

[18]:
```
Empty DataFrame
Columns: [state, account length, area code, phone number, international plan,
voice mail plan, number vmail messages, total day minutes, total day calls,
total day charge, total eve minutes, total eve calls, total eve charge, total
night minutes, total night calls, total night charge, total intl minutes, total
intl calls, total intl charge, customer service calls, churn]
Index: []

[0 rows x 21 columns]
```

[19]:
```
data = data.drop_duplicates()      # To drop all the duplicate records from the␣
 ↪DataFrame, and saving the updated DF
```

[20]:
```
data                               # To have a look at the DataFrame
```

[20]:
```
     state  account length  area code phone number international plan  \
0       KS             128        415     382-4657                 no
1       OH             107        415     371-7191                 no
2       NJ             137        415     358-1921                 no
3       OH              84        408     375-9999                yes
4       OK              75        415     330-6626                yes
…     …              …        …          …                  …
3328    AZ             192        415     414-4276                 no
3329    WV              68        415     370-3271                 no
3330    RI              28        510     328-8230                 no
3331    CT             184        510     364-6381                yes
3332    TN              74        415     400-4344                 no

     voice mail plan  number vmail messages  total day minutes  \
0                yes                     25              265.1
1                yes                     26              161.6
2                 no                      0              243.4
3                 no                      0              299.4
4                 no                      0              166.7
…                …                    …                  …
3328             yes                     36              156.2
3329              no                      0              231.1
3330              no                      0              180.8
3331              no                      0              213.8
3332             yes                     25              234.4

      total day calls  total day charge  …  total eve calls  \
0                 110             45.07  …               99
1                 123             27.47  …              103
```

```
2                114            41.38  …             110
3                 71            50.90  …              88
4                113            28.34  …             122
…                …              …  …              …
3328              77            26.55  …             126
3329              57            39.29  …              55
3330             109            30.74  …              58
3331             105            36.35  …              84
3332             113            39.85  …              82

      total eve charge  total night minutes  total night calls  \
0                16.78                244.7                 91
1                16.62                254.4                103
2                10.30                162.6                104
3                 5.26                196.9                 89
4                12.61                186.9                121
…                 …                 …                …
3328             18.32                279.1                 83
3329             13.04                191.3                123
3330             24.55                191.9                 91
3331             13.57                139.2                137
3332             22.60                241.4                 77

      total night charge  total intl minutes  total intl calls  \
0                  11.01                10.0                 3
1                  11.45                13.7                 3
2                   7.32                12.2                 5
3                   8.86                 6.6                 7
4                   8.41                10.1                 3
…                   …                 …                …
3328               12.56                 9.9                 6
3329                8.61                 9.6                 4
3330                8.64                14.1                 6
3331                6.26                 5.0                10
3332               10.86                13.7                 4

      total intl charge  customer service calls  churn
0                  2.70                       1  False
1                  3.70                       1  False
2                  3.29                       0  False
3                  1.78                       2  False
4                  2.73                       3  False
…                  …                     …    …
3328               2.67                       2  False
3329               2.59                       3  False
3330               3.81                       2  False
3331               1.35                       2  False
```

```
3332               3.70                       0  False

[3333 rows x 21 columns]
```

4   1. Analyzing the 'Churn' Variable

```
[21]: data.head()                         # To show Top5 records of the DataFrame
```

```
[21]:   state  account length  area code phone number international plan  \
     0    KS             128        415     382-4657                 no
     1    OH             107        415     371-7191                 no
     2    NJ             137        415     358-1921                 no
     3    OH              84        408     375-9999                yes
     4    OK              75        415     330-6626                yes

       voice mail plan  number vmail messages  total day minutes  total day calls  \
     0             yes                     25              265.1              110
     1             yes                     26              161.6              123
     2              no                      0              243.4              114
     3              no                      0              299.4               71
     4              no                      0              166.7              113

        total day charge  …  total eve calls  total eve charge  \
     0             45.07  …               99             16.78
     1             27.47  …              103             16.62
     2             41.38  …              110             10.30
     3             50.90  …               88              5.26
     4             28.34  …              122             12.61

        total night minutes  total night calls  total night charge  \
     0                244.7                 91               11.01
     1                254.4                103               11.45
     2                162.6                104                7.32
     3                196.9                 89                8.86
     4                186.9                121                8.41

        total intl minutes  total intl calls  total intl charge  \
     0                10.0                 3               2.70
     1                13.7                 3               3.70
     2                12.2                 5               3.29
     3                 6.6                 7               1.78
     4                10.1                 3               2.73
```

```
   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

```
[6]: data['churn'].unique()              # To show the unique values present in the␣
     ↪column 'churn'
```

```
[6]: array([False,  True])
```

```
[23]: A = data['churn'].value_counts()     # To show the count of the unique values␣
      ↪of the column 'churn' ...
      print(A)                             # ... and saving the result in variable␣
      ↪'A' and printing variable 'A'
```

```
False    2850
True      483
Name: churn, dtype: int64
```

```
[24]: str(483/3333*100) + " %"             # Just to check the percentage of True␣
      ↪(churned customers) from the dataset ...
                                           # ... and showing the result in string␣
      ↪format, after adding % sign at the end
```

```
[24]: '14.491449144914492 %'
```

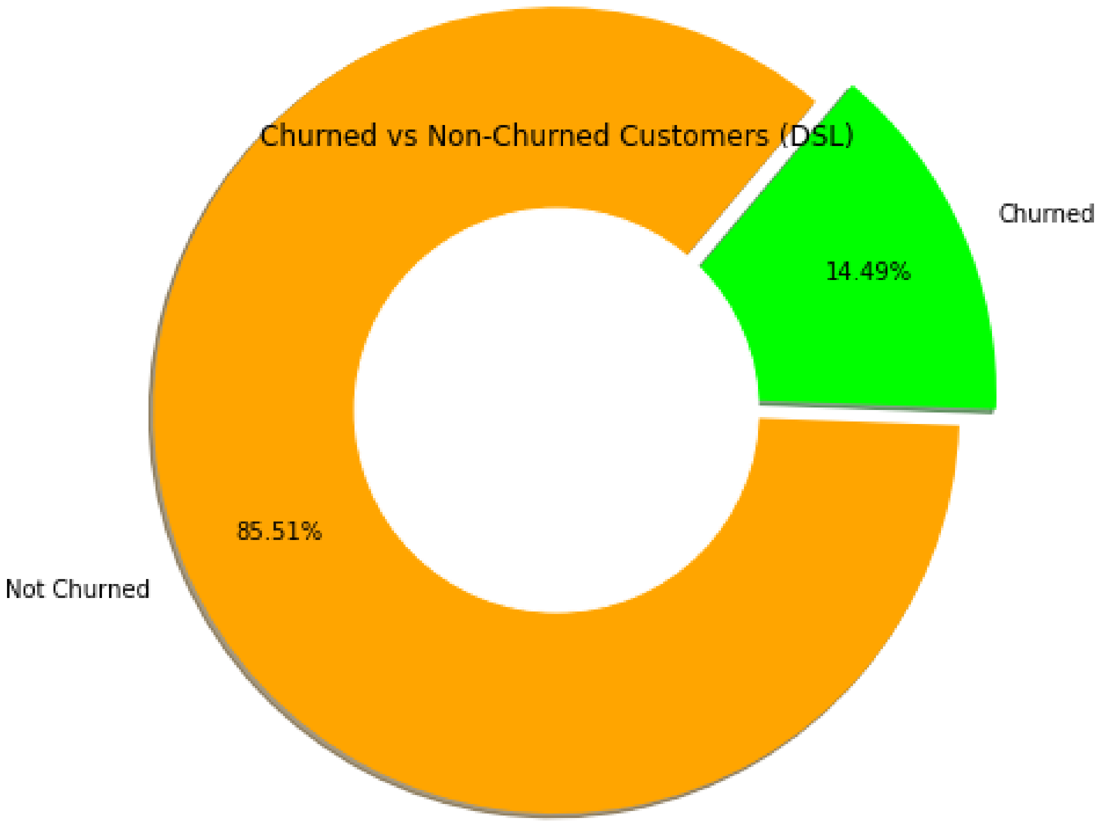
```
[25]: type(A)                              # To check the type of variable 'A' ; it's␣
      ↪a Series
```

```
[25]: pandas.core.series.Series
```

4.0.1 Donut Chart

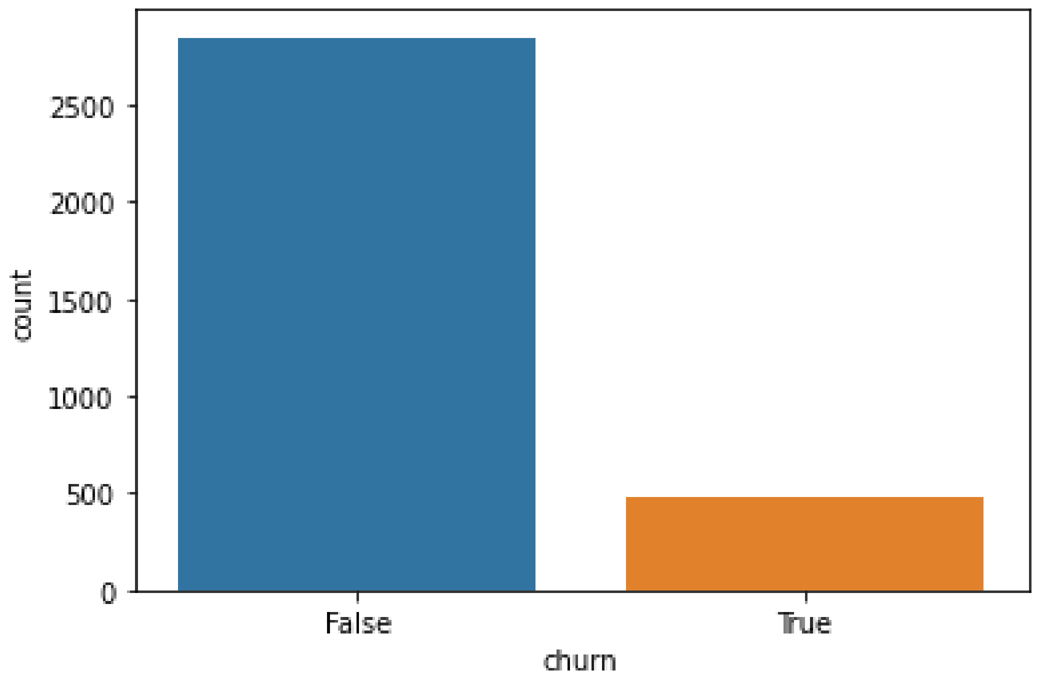
```
[26]: # Creating a Donut Chart for Churn vs Non-Churn Customers

      plt.pie(A, labels=['Not Churned', 'Churned'], colors=['orange','lime'],␣
      ↪startangle=50, shadow=True, radius=2,
              explode=(0,0.2), autopct='%1.2f%%', pctdistance=0.75);

      circle = plt.Circle((0,0), 1, color='white')
      c = plt.gcf()

      c.gca().add_artist(circle)
```

```
plt.title("Churned vs Non-Churned Customers (DSL)")
plt.show()

# Donut chart is a modified Pie chart, with an area from center cut out
#1 First, drawing a pie plot, using Matplotlib library
# The data is taken from the variable 'A' .... labels are given as 'Churned' &␣
 ↪'Not Churned' ...
# ... color is given to each label 'orange' & 'lime' ... startangle means the␣
 ↪angle for slicing, set as 50 ...
# ... shadow is True means it will drop some shadow of the chart ... radius of␣
 ↪the circle is set as 2 ...
# ... explode is used to cut the slice out of the figure ...
# ... autopct is used to show the % on the chart upto required decimal points ..
 ↪.
# ... pctdistance is given for distance of % from the center
#2 Second, using plt.Circle...we will create a circle and save it in the␣
 ↪variable name 'circle' ...
# ... putting (0,0) by default ... 1 is radius of circle ... and color of␣
 ↪circle is white
# plt.gcf() is used to get the current figure ... and we are saving it in␣
 ↪variable 'c'
#3 Third, we will add the 'circle' at the center of pie chart ... using gca().
 ↪add_artist()
# We have given the title to the chart using plt.title()
# plt.show() - To show the chart
```



4.0.2 Countplot

[27]:
```
# Countplot for Churn vs Non-Churn Customers

sns.countplot(data['churn']);

# Drawing a countplot for the column 'churn' from the datset using seaborn␣
↪library
```



Outcome - After analyzing the Churn column, we notice that almost 15% customers have churned. Now, we will analyze other columns to find out the possible reasons for this churn.

---

5 2. Analyzing the 'State' Variable

[28]:
```
data.head(2)                    # To show Top 2 records of the dataset
```

[28]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
```

```
   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3                2.7
1                13.7                 3                3.7

   customer service calls  churn
0                       1  False
1                       1  False

[2 rows x 21 columns]
```

```
[29]: data.state.nunique()          # To show the total number of unique values␣
       ↪present in column 'state'
```

```
[29]: 51
```

unique( )

```
[30]: data.state.unique()           # To show all the unique values of the column␣
       ↪'state'
                                    # it shows the output in the form of 1-D array
```

```
[30]: array(['KS', 'OH', 'NJ', 'OK', 'AL', 'MA', 'MO', 'LA', 'WV', 'IN', 'RI',
             'IA', 'MT', 'NY', 'ID', 'VT', 'VA', 'TX', 'FL', 'CO', 'AZ', 'SC',
             'NE', 'WY', 'HI', 'IL', 'NH', 'GA', 'AK', 'MD', 'AR', 'WI', 'OR',
             'MI', 'DE', 'UT', 'CA', 'MN', 'SD', 'NC', 'WA', 'NM', 'NV', 'DC',
             'KY', 'ME', 'MS', 'TN', 'PA', 'CT', 'ND'], dtype=object)
```

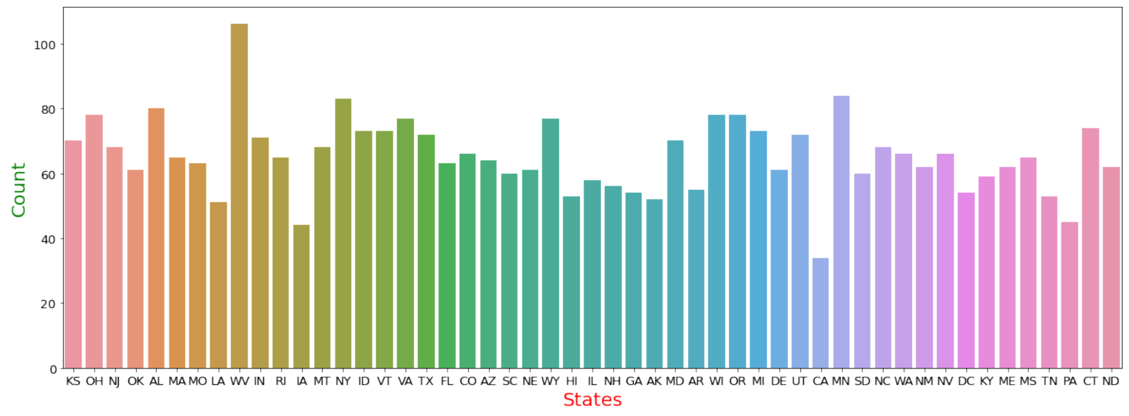
```
[7]: data.state.value_counts()      # To show the occurance/count of all unique␣
      ↪values of the column 'state'

     # By default , it shows result in descending order
```

```
[7]: state
     WV    106
     MN     84
     NY     83
     AL     80
     WI     78
     OH     78
     OR     78
     WY     77
     VA     77
     CT     74
     MI     73
```

```
ID     73
VT     73
TX     72
UT     72
IN     71
MD     70
KS     70
NC     68
NJ     68
MT     68
CO     66
NV     66
WA     66
RI     65
MA     65
MS     65
AZ     64
FL     63
MO     63
NM     62
ME     62
ND     62
NE     61
OK     61
DE     61
SC     60
SD     60
KY     59
IL     58
NH     56
AR     55
GA     54
DC     54
HI     53
TN     53
AK     52
LA     51
PA     45
IA     44
CA     34
Name: count, dtype: int64
```

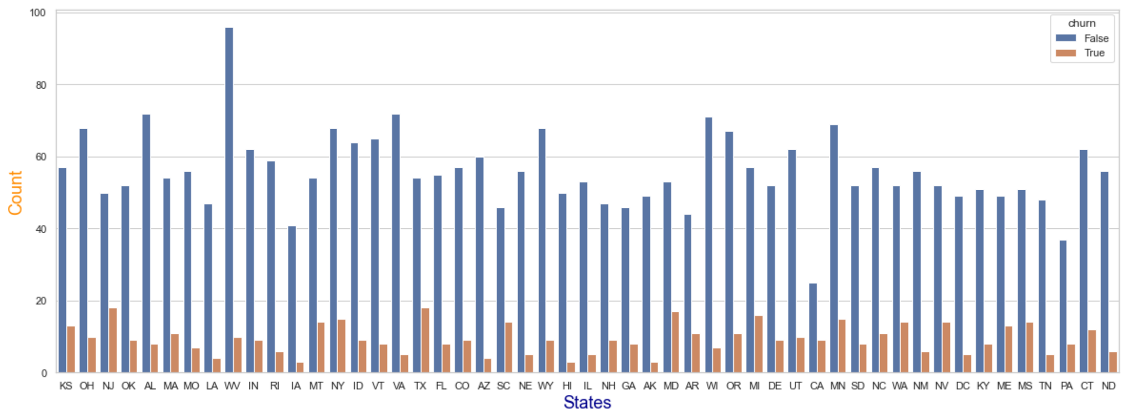
5.0.1 Countplot

[32]:
```
plt.figure(figsize=(20,7))        # Setting the size of the figure as 20 x 7,␣
 ↪using matplotlib library

sns.countplot(data['state'])     # Drawing a countplot for the column 'state'␣
 ↪from the datset using seaborn library
plt.xticks(fontsize=13)          # Setting the fontsize of states on x-axis
plt.yticks(fontsize=13)          # Setting the fontsize of numbers on y-axis
plt.xlabel('States', fontsize=20, color='Red')       # Setting the label on␣
 ↪x-axis as States , its fontsize , its color
plt.ylabel('Count', fontsize=20, color='Green')      # Setting the label on␣
 ↪y-axis as Count , its fontsize , its color
plt.show()                                           # To display the figure

# The x-axis showing the names of the states ... and y-axis showing their␣
 ↪counts (occurance) ...
# ... means how many times a state is present in the column
```



[33]:
```
# Showing Churn/Not Churn State-wise

sns.set(style="whitegrid")           # Using seaborn library, setting the␣
 ↪background of the figure as "whitegrid"

plt.figure(figsize=(20,7))           # Setting the size of the figure as 20 x␣
 ↪7, using matplotlib library

sns.countplot(data.state, hue = data.churn)

plt.xlabel('States', fontsize=18, color='DarkBlue')   # Setting the Label on␣
 ↪x-axis as States, its fontsize & color
```

```
plt.ylabel('Count', fontsize=18, color='DarkOrange'); # Setting the Label on␣
↪y-axis as Count , its fontsize & color

# Drawing a countplot for the column 'state' from the datset using seaborn␣
↪library ...
# ... selecting the Churn column for hue
# 'hue = data.churn' represents that we want to use column "churn" for color␣
↪encoding
# i.e. color the bars for Churn and Not-Churn differently
# Blue color representing False (Not Churn) and Orange color representing True␣
↪(Churn) for each state separately
```



[34]:
```
data.head(2)                    # To show Top2 records of the dataset
```

[34]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3                2.7
```

```
1                13.7                 3                3.7

   customer service calls  churn
0                       1  False
1                       1  False

[2 rows x 21 columns]
```

[35]:
```
a = data.groupby("state")['churn'].mean()*100

a.sort_values()

# Here, we will consider two columns while using groupby()
# Used groupby on ("state") column ... and showing the mean of values of␣
 ↪['churn'] column wrt to each state ...
# ... and multiplying the result by 100
# By default, it will consider 'True' values from ['churn'] column to calculate␣
 ↪mean ...
# ... bcz the data-type of column ['churn'] is boolean type
```

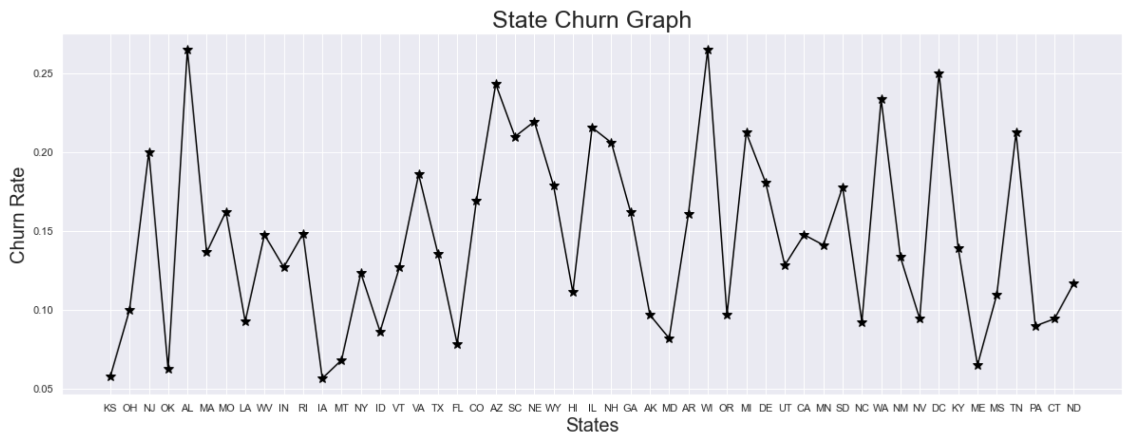
[35]:
```
state
HI     5.660377
AK     5.769231
AZ     6.250000
VA     6.493506
IA     6.818182
LA     7.843137
NE     8.196721
IL     8.620690
WI     8.974359
RI     9.230769
DC     9.259259
TN     9.433962
WV     9.433962
NM     9.677419
ND     9.677419
AL    10.000000
VT    10.958904
MO    11.111111
WY    11.688312
ID    12.328767
IN    12.676056
FL    12.698413
OH    12.820513
SD    13.333333
KY    13.559322
CO    13.636364
```

```
UT    13.888889
OR    14.102564
OK    14.754098
DE    14.754098
GA    14.814815
NH    16.071429
NC    16.176471
CT    16.216216
MA    16.923077
PA    17.777778
MN    17.857143
NY    18.072289
KS    18.571429
AR    20.000000
MT    20.588235
ME    20.967742
WA    21.212121
NV    21.212121
MS    21.538462
MI    21.917808
SC    23.333333
MD    24.285714
TX    25.000000
NJ    26.470588
CA    26.470588
Name: churn, dtype: float64
```

5.0.2 Line Chart

[36]:
```
# Drawing a Line Chart showing State-wise Churn rate

X = data.state.unique()      # Considering all unique values of 'state' column,␣
 ↪and saving the output in variable X
Y = data.groupby("state")['churn'].mean()    # Using groupby on 'state' column,␣
 ↪and calculating the mean of churn ...
                                             # ... values against each state,␣
 ↪and saving the output in variable Y
sns.set(style="darkgrid")                    # Using seaborn library, setting the␣
 ↪background of the figure as "whitegrid"

plt.rcParams['figure.figsize'] = (20,7)     # Using rcParams from matplotlib␣
 ↪library ...
                                            # ... setting the size of the figure␣
 ↪as 20 x 7
plt.rcParams['lines.linestyle'] = '-'       # Using rcParams from matplotlib␣
 ↪library ...
```

```
                                                    # ... setting the style of the line␣
↪as ' - '

plt.plot(X, Y, color='black', marker = '*', markersize='10')     # Plotting a␣
↪Line chart with X and Y ...
# ... States showing on x-axis & Churn Rate showing on y-axis ... color of line␣
↪set as 'black' ...
# ... marker set as '*' ... markersize set as '10'

plt.title("State Churn Graph", fontsize=25)  # Using matplotlib, setting the␣
↪Title of figure, with fontsize of 25
plt.xlabel("States", fontsize=20)            # Using matplotlib, setting the␣
↪Label of x-axis, with fontsize of 20
plt.ylabel("Churn Rate", fontsize=20);       # Using matplotlib, setting the␣
↪Label of y-axis, with fontsize of 20

# rcParams - run command parameters
# matplotlib.rcParams contains some properties in matplotlibrc file. We can use␣
↪it to control the defaults ...
# ... of almost every property in Matplotlib: figure size and DPI, line width,␣
↪color and style, axes, axis ...
# ... and grid properties, text and font properties and so on. In order to use␣
↪matplotlib.
```



[37]:
```
#x = data.state
#x = data.groupby("state")['churn'].mean().sort_values(ascending=False).head(5);
#y = data.groupby("state")['churn'].mean().sort_values(ascending=False);

#plt.bar(x,y)

data.groupby("state")['churn'].mean().sort_values(ascending=False).head(5)*100
```

```
[37]: state
      NJ    26.470588
      CA    26.470588
      TX    25.000000
      MD    24.285714
      SC    23.333333
      Name: churn, dtype: float64
```

```
[38]: data.head()              # To show Top5 records of the dataset
```

```
[38]:   state  account length  area code phone number international plan  \
      0    KS             128        415     382-4657                 no
      1    OH             107        415     371-7191                 no
      2    NJ             137        415     358-1921                 no
      3    OH              84        408     375-9999                yes
      4    OK              75        415     330-6626                yes

        voice mail plan  number vmail messages  total day minutes  total day calls  \
      0             yes                     25              265.1              110
      1             yes                     26              161.6              123
      2              no                      0              243.4              114
      3              no                      0              299.4               71
      4              no                      0              166.7              113

         total day charge  …  total eve calls  total eve charge  \
      0             45.07  …               99             16.78
      1             27.47  …              103             16.62
      2             41.38  …              110             10.30
      3             50.90  …               88              5.26
      4             28.34  …              122             12.61

         total night minutes  total night calls  total night charge  \
      0                244.7                 91               11.01
      1                254.4                103               11.45
      2                162.6                104                7.32
      3                196.9                 89                8.86
      4                186.9                121                8.41

         total intl minutes  total intl calls  total intl charge  \
      0                10.0                 3               2.70
      1                13.7                 3               3.70
      2                12.2                 5               3.29
      3                 6.6                 7               1.78
      4                10.1                 3               2.73

         customer service calls  churn
      0                       1  False
```

```
1                        1  False
2                        0  False
3                        2  False
4                        3  False

[5 rows x 21 columns]
```

[39]:

```
# Calculatiog State vs Churn Percentage

c = pd.crosstab(data.state, data.churn ,margins=True)
                                 # Using crosstab function from pandas library␣
 ↪to compute a simple cross-tabulation...
                                 # ...of two columns i.e., 'state' & 'churn'␣
 ↪and saving the output in varibale 'c'
                                 # ... margins=True means - to show the sum of␣
 ↪both values in a new column 'All'
c['Percentage_Churn'] = c[True]/(c['All']) * 100       # Creating a new column␣
 ↪'Percantage_Churn', which shows the...
                                                       # ...percentage of churn␣
 ↪customers in each state
print(c.head())                                        # Printing top 5 records␣
 ↪of the result
print(type(c))                                         # Printing type of 'c' ;␣
 ↪it's is a DataFrame
```

```
churn  False  True  All  Percentage_Churn
state
AK        49     3   52          5.769231
AL        72     8   80         10.000000
AR        44    11   55         20.000000
AZ        60     4   64          6.250000
CA        25     9   34         26.470588
<class 'pandas.core.frame.DataFrame'>
```

[40]:

```
c.info()              # To show the basic information of the DataFrame 'c'
#It shows indexes, columns counts, each column name with its non-null values␣
 ↪count & data-type and, memory of DataFrame
# Now, in this dataframe 'c', we have 4 columns␣
 ↪'False,True,All,Percentage_Churn' each having 52 non-null values
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 52 entries, AK to All
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   False              52 non-null     int64
 1   True               52 non-null     int64
 2   All                52 non-null     int64
 3   Percentage_Churn  52 non-null     float64
dtypes: float64(1), int64(3)
memory usage: 2.0+ KB
```

[41]:
```
c.sort_values(by='Percentage_Churn', ascending = False).head()
# sorting the records of the dataframe 'c' wrt the column 'Percentage_Churn' in␣
 ↪descending order...
#...and showing top 5 records using head function
```

[41]:
```
churn  False  True  All  Percentage_Churn
state
CA        25     9   34         26.470588
NJ        50    18   68         26.470588
TX        54    18   72         25.000000
MD        53    17   70         24.285714
SC        46    14   60         23.333333
```

6 3. Analyzing the 'Area Code' Column

[8]:
```
data.head()
```

[8]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61
```

```
   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[43]:
```
data['area code'].nunique()          # To show the total number of unique␣
 ↪values present in the column 'area code'
```

[43]:
```
3
```

[44]:
```
data['area code'].unique()           # To show all the unique values of the␣
 ↪column 'area code'...
                                     # it shows the output in the form of 1-D␣
 ↪array
```

[44]:
```
array([415, 408, 510], dtype=int64)
```

[45]:
```
data['area code'].value_counts()     # To show the occurance/count of all␣
 ↪unique values of the column 'area code'

# By default , it shows result in descending order
# To show the results in ascending order, we can write data.state.
 ↪value_counts(ascending=True)
```

[45]:
```
415    1655
510     840
408     838
Name: area code, dtype: int64
```

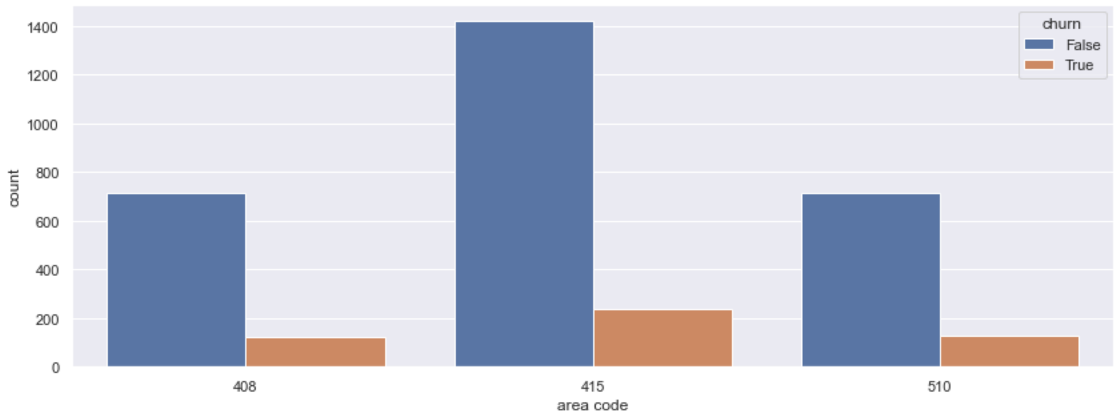
[46]:
```
data.groupby('area code')['churn'].mean() * 100

# Here, we will consider two columns while using groupby()
# Used groupby on 'area_code' column ... and showing the mean of values of␣
 ↪'churn' column wrt to each area code...
# ...and multiplying the result by 100
```

[46]:
```
area code
408    14.558473
415    14.259819
510    14.880952
Name: churn, dtype: float64
```

[47]:
```
# Calculatiog Area Code vs Churn Percentage

d = pd.crosstab(data['area code'], data['churn'], margins= True)
                # Using crosstab function from pandas library to compute a␣
 ↪simple...
                #...cross-tabulation of two columns i.e., 'area code' & 'churn'␣
 ↪and saving the output in varibale 'd'
                # margins=True means - to show sum of both values in a new␣
 ↪column 'All'
d['Churn Percentage'] = d[True]/(d['All']) * 100                    # Creating␣
 ↪a new column 'Percantage_Churn'...
                                                    #...which shows the␣
 ↪percentage of churn customers in each area code
d                                                   # showing result of 'd'
```

[47]:
```
churn      False  True   All  Churn Percentage
area code
408          716   122   838         14.558473
415         1419   236  1655         14.259819
510          715   125   840         14.880952
All         2850   483  3333         14.491449
```

[48]:
```
# Drawing a Bar Graph to show Area Code wise Churn

plt.figure(figsize=(14,5))
# Setting the size of the figure as 14x5, using matplotlib library

sns.countplot(x='area code', hue='churn', data=data);
# Drawing a countplot for the column 'area code' from the dataset using seaborn␣
 ↪library...
# ...selecting the Churn column for hue
# 'hue=churn' represents that we want to use column 'churn' for color encoding
# i.e. color the bars for Churn and Not-Churn differently
# Blue color representing False (Not Churn) and Orange color representing True
↪(Churn) for each area code separately
```



7 4. Analyzing the 'Account Length' Column

[9]:
```
data.head()
```

[9]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61
```

```
   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01   
1                254.4                103               11.45   
2                162.6                104                7.32   
3                196.9                 89                8.86   
4                186.9                121                8.41   

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70   
1                13.7                 3               3.70   
2                12.2                 5               3.29   
3                 6.6                 7               1.78   
4                10.1                 3               2.73   

   customer service calls  churn  
0                       1  False  
1                       1  False  
2                       0  False  
3                       2  False  
4                       3  False  

[5 rows x 21 columns]
```

[50]:
```
data['account length'].nunique()   # To show the total number of unique values␣
 ↪present in the column 'account length'
```

[50]:
```
212
```

[11]:
```
# Creating two different dataframes - one for 'churned customers' & second for␣
 ↪'non-churned customers'

churn_data = data[data['churn'] == True]
#creating a new dataframe "churn_data", in which we are considering all the␣
 ↪records where churn is 'True'

not_churn_data = data[data['churn'] == False]
#creating a new dataframe "not_churn_data", in which we are considering all the␣
 ↪records where churn is 'False'
```

[52]:
```
churn_data.head(2)               # To show Top2 records of the dataframe
```

[52]:
```
   state  account length  area code phone number international plan  \
10    IN              65        415     329-6603                 no   
15    NY             161        415     351-7269                 no   

   voice mail plan  number vmail messages  total day minutes  total day calls  \
10               no                      0            129.1              137
15               no                      0            332.9               67

    total day charge  …  total eve calls  total eve charge  \
10             21.95  …               83             19.42
15             56.59  …               97             27.01

    total night minutes  total night calls  total night charge  \
10                208.8                111                9.40
15                160.6                128                7.23

    total intl minutes  total intl calls  total intl charge  \
10                12.7                 6               3.43
15                 5.4                 9               1.46

    customer service calls  churn
10                       4   True
15                       4   True

[2 rows x 21 columns]
```

[53]:
```
not_churn_data.head(2)            # To show Top2 records of the dataframe
```

[53]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3                2.7
1                13.7                 3                3.7

   customer service calls  churn
0                       1  False
1                       1  False
```

```
[2 rows x 21 columns]
```

[12]:
```
churn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 483 entries, 10 to 3323
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   483 non-null    object
 1   account length          483 non-null    int64
 2   area code               483 non-null    int64
 3   phone number            483 non-null    object
 4   international plan      483 non-null    object
 5   voice mail plan         483 non-null    object
 6   number vmail messages   483 non-null    int64
 7   total day minutes       483 non-null    float64
 8   total day calls         483 non-null    int64
 9   total day charge        483 non-null    float64
 10  total eve minutes       483 non-null    float64
 11  total eve calls         483 non-null    int64
 12  total eve charge        483 non-null    float64
 13  total night minutes     483 non-null    float64
 14  total night calls       483 non-null    int64
 15  total night charge      483 non-null    float64
 16  total intl minutes      483 non-null    float64
 17  total intl calls        483 non-null    int64
 18  total intl charge       483 non-null    float64
 19  customer service calls  483 non-null    int64
 20  churn                   483 non-null    bool
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 79.7+ KB
```

[13]:
```
not_churn_data.info()
```

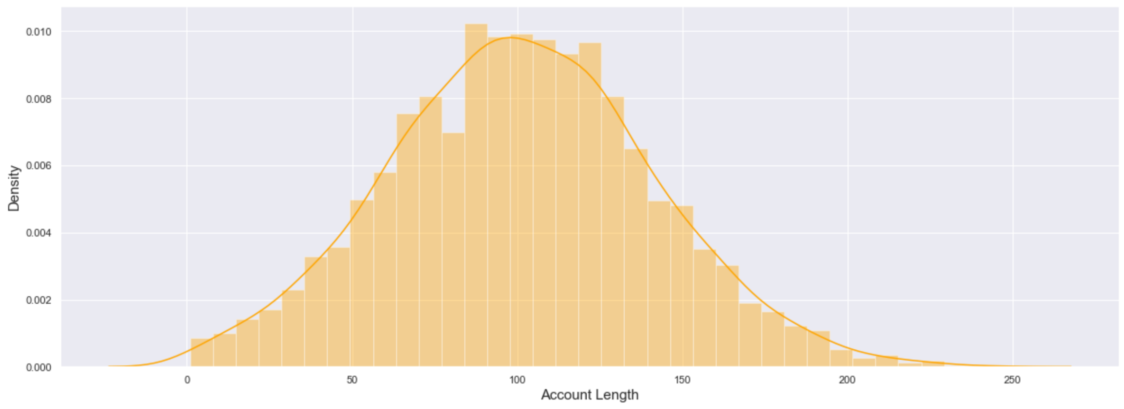
```
<class 'pandas.core.frame.DataFrame'>
Index: 2850 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   2850 non-null   object
 1   account length          2850 non-null   int64
 2   area code               2850 non-null   int64
 3   phone number            2850 non-null   object
 4   international plan      2850 non-null   object
 5   voice mail plan         2850 non-null   object
 6   number vmail messages   2850 non-null   int64
 7   total day minutes       2850 non-null   float64
 8   total day calls         2850 non-null   int64
 9   total day charge        2850 non-null   float64
 10  total eve minutes       2850 non-null   float64
 11  total eve calls         2850 non-null   int64
 12  total eve charge        2850 non-null   float64
 13  total night minutes     2850 non-null   float64
 14  total night calls       2850 non-null   int64
 15  total night charge      2850 non-null   float64
 16  total intl minutes      2850 non-null   float64
 17  total intl calls        2850 non-null   int64
 18  total intl charge       2850 non-null   float64
 19  customer service calls  2850 non-null   int64
 20  churn                   2850 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 470.4+ KB
```

[14]:
```python
data.shape
```

[14]:
```
(3333, 21)
```

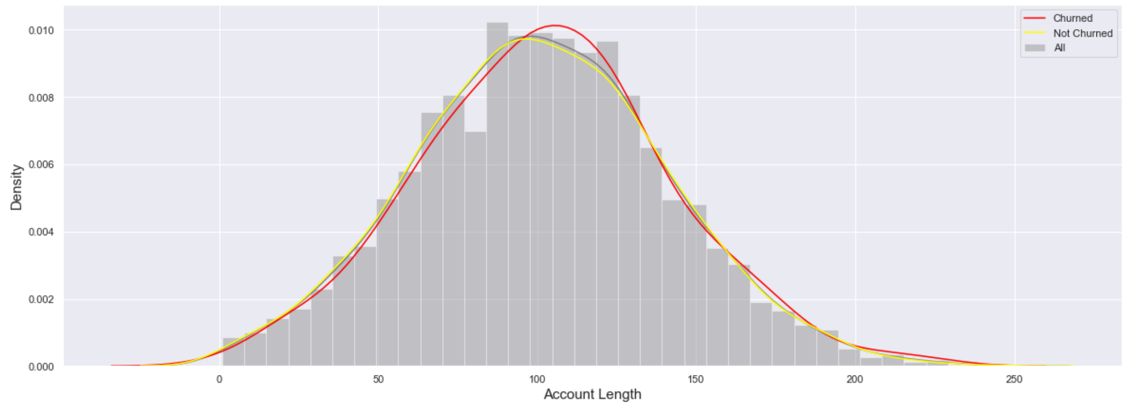
7.0.1 Distribution Plot

[57]:
```python
# Creating a Distribution Plot for Account Length column

sns.distplot(data['account length'], color='orange')
plt.xlabel("Account Length", fontsize=15)
plt.ylabel("Density", fontsize=15)
plt.show()

# The distplot shows a histogram with a line on it. It represents the␣
 ↪distribution of a variable against the...
#...density distribution.
# Here, using Seaborn library we are plotting a distribution plot for the␣
 ↪column 'account length'...
# ...with 'orange' color.
# Using matplotlib, setting the label of x-axis as "Account Length", with␣
 ↪fontsize of 15
# Using matplotlib, setting the label of x-axis as "Density", with fontsize of␣
 ↪15
# plt.show() - To show the plot
```



[58]:
```
# Comparing Account Length of 'churned' and 'not churned' customers, using␣
↪distplot

sns.distplot(data['account length'], color='grey', label='All')
# Using Seaborn library, we are plotting a distribution plot for the column␣
↪'account length' from original dataframe...
# ...with 'grey' color ... and setting the label as 'All'

sns.distplot(churn_data['account length'], hist=False, color='red',␣
↪label='Churned')
# Using Seaborn library, we are plotting a distribution plot for the column␣
↪'account length' from 'churn_data'...
#...dataframe with 'red' color ... and setting the label as 'Churned'. Here,␣
↪hist=False is given because...
#...we don't want to show a histogram for this, we want to show a line only

sns.distplot(not_churn_data['account length'], hist=False, color='yellow',␣
↪label='Not Churned')
# Using Seaborn library, we are plotting a distribution plot for the column␣
↪'account length' from 'non_churn_data'...
#...dataframe with 'yellow' color ... and setting the label as 'Not Churned'.␣
↪Here, hist=False is given because...
#...we don't want to show a histogram for this, we want to show a line only

plt.xlabel("Account Length", fontsize=15)
# Using matplotlib, setting the label of x-axis as "Account Length", with␣
↪fontsize of 15

plt.ylabel("Density", fontsize=15)
# Using matplotlib, setting the label of y-axis as "Density", with fontsize of␣
↪15
```

```
plt.rcParams['figure.figsize'] = (15,7);
# setting the size of the figure as 15x7 using matplotlib's rcParams

plt.show()        # To show the plot
```



8 5. Analyzing the 'International Plan' Column

[15]:
```
data.head()
```

[15]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
```

```
3             50.90  …                88              5.26
4             28.34  …               122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[60]:
```
data['international plan'].unique()      # To show all the unique values of the␣
 ↪column 'international plan'
                                         # Two unique values present - 'no' and␣
 ↪'yes'
```

[60]:
```
array(['no', 'yes'], dtype=object)
```

[61]:
```
data['international plan'].value_counts()

# To show the occurance/count of all unique values of the column 'international␣
 ↪plan'
# By default , it shows result in descending order
# To show the results in ascending order, we can write data.state.
 ↪value_counts(ascending=True)
```

[61]:
```
no     3010
yes     323
Name: international plan, dtype: int64
```

[18]:
```
# Creating a new dataframe 'yes_int_plan',...
```

```
#...containing the records of those Churned customers who were having␣
↪international plan

yes_int_plan = data[(data['international plan'] == 'yes') &␣
↪(data['churn']==True)]
yes_int_plan

# Here, we have used filtering with '&' operator, to select records satisfying␣
↪two conditions -
# 1. 'international plan' == 'yes'
# 2. 'churn' == 'True'

# it shows there are total 137 records satisfying these two conditions
```

[18]:

```
     state  account length  area code phone number international plan  \
41      MD             135        408     383-6029                yes
115     ME              36        510     363-1069                yes
144     VT             117        408     390-2390                yes
198     ME             131        510     353-7292                yes
214     FL              70        510     366-6345                yes
…      …               …         …          …                  …
3246    NC              77        408     334-6129                yes
3255    RI             138        510     411-6823                yes
3291    MI             119        510     335-7324                yes
3304    IL              71        510     330-7137                yes
3320    GA             122        510     411-5677                yes

     voice mail plan  number vmail messages  total day minutes  \
41               yes                     41              173.1
115              yes                     42              196.8
144               no                      0              167.1
198              yes                     26              292.9
214               no                      0              226.7
…              …                     …                 …
3246             yes                     44              103.2
3255              no                      0              286.2
3291             yes                     22              172.1
3304              no                      0              186.1
3320              no                      0              140.0

      total day calls  total day charge  …  total eve calls  \
41                 85             29.43  …              107
115                89             33.46  …              122
144                86             28.41  …               87
198               101             49.79  …               97
214                98             38.54  …              115
…                …               …  …              …
```

```
3246              117             17.54  …             86
3255               61             48.65  …             60
3291              119             29.26  …            133
3304              114             31.64  …            140
3320              101             23.80  …             77

      total eve charge  total night minutes  total night calls  \
41               17.33                122.2                 78
115              21.67                138.3                126
144              15.09                249.4                132
198              16.97                255.3                127
214              19.39                 73.2                 93
…                  …                  …                …
3246             20.09                203.5                101
3255             15.91                146.2                114
3291             19.01                150.0                 94
3304             16.88                206.5                 80
3320             16.69                120.1                133

      total night charge  total intl minutes  total intl calls  \
41                  5.50                14.6                15
115                 6.22                20.0                 6
144                11.22                14.1                 7
198                11.49                13.8                 7
214                 3.29                17.6                 4
…                    …                  …               …
3246                9.16                11.9                 2
3255                6.58                11.0                 4
3291                6.75                13.9                20
3304                9.29                13.8                 5
3320                5.40                 9.7                 4

      total intl charge  customer service calls  churn
41                 3.94                       0   True
115                5.40                       0   True
144                3.81                       2   True
198                3.73                       4   True
214                4.75                       2   True
…                   …                      …    …
3246               3.21                       0   True
3255               2.97                       2   True
3291               3.75                       1   True
3304               3.73                       4   True
3320               2.62                       4   True

[137 rows x 21 columns]
```

[19]:
```
# Creating a new dataframe 'no_int_plan',...
#...containing the records of those Churned customers who don't have␣
↪international plan

no_int_plan = data[(data['international plan'] == 'no') & (data['churn']==True)]
no_int_plan

# Here, we have used filtering with '&' operator, to select records satisfying␣
↪two conditions -
# 1. 'international plan' == 'no'
# 2. 'churn' == 'True'

# it shows there are total 346 records satisfying these two conditions
```

[19]:
```
     state  account length  area code phone number international plan  \
10      IN              65        415     329-6603                 no
15      NY             161        415     351-7269                 no
21      CO              77        408     393-7984                 no
33      AZ              12        408     360-1596                 no
48      ID             119        415     398-1294                 no
…      …               …         …           …                 …
3280    AR              76        408     345-3614                 no
3287    KS             170        415     404-5840                 no
3301    CA              84        415     417-1488                 no
3322    MD              62        408     409-1856                 no
3323    IN             117        415     362-5899                 no

     voice mail plan  number vmail messages  total day minutes  \
10                no                      0              129.1
15                no                      0              332.9
21                no                      0               62.4
33                no                      0              249.6
48                no                      0              159.1
…                …                      …                …
3280              no                      0              107.3
3287             yes                     42              199.5
3301              no                      0              280.0
3322              no                      0              321.1
3323              no                      0              118.4

      total day calls  total day charge  …  total eve calls  \
10                137             21.95  …               83
15                 67             56.59  …               97
21                 89             10.61  …              121
33                118             42.43  …              119
48                114             27.05  …              117
…                …               …  …              …
```

```
3280              140             18.24  …             133
3287              119             33.92  …              90
3301              113             47.60  …              90
3322              105             54.59  …             122
3323              126             20.13  …              97

      total eve charge  total night minutes  total night calls  \
10               19.42                208.8                111
15               27.01                160.6                128
21               14.44                209.6                 64
33               21.45                280.2                 90
48               19.66                143.2                 91
…                …                  …                 …
3280             20.25                271.8                116
3287             11.48                184.6                 49
3301             17.19                156.8                103
3322             22.57                180.5                 72
3323             21.19                227.0                 56

      total night charge  total intl minutes  total intl calls  \
10                  9.40                12.7                 6
15                  7.23                 5.4                 9
21                  9.43                 5.7                 6
33                 12.61                11.8                 3
48                  6.44                 8.8                 3
…                  …                  …                …
3280               12.23                10.0                 3
3287                8.31                10.9                 3
3301                7.06                10.4                 4
3322                8.12                11.5                 2
3323               10.22                13.6                 3

      total intl charge  customer service calls  churn
10                 3.43                       4   True
15                 1.46                       4   True
21                 1.54                       5   True
33                 3.19                       1   True
48                 2.38                       5   True
…                 …                    …    …
3280               2.70                       4   True
3287               2.94                       4   True
3301               2.81                       0   True
3322               3.11                       4   True
3323               3.67                       5   True

[346 rows x 21 columns]
```

[64]:
```
# we noticed that 323 customers were using international plan, and 137␣
 ↪customers churned out of them
# means a churn percentage of 42.4

137/323*100
```

[64]:
```
42.414860681114554
```

[65]:
```
# we noticed that 3010 customers were not using international plan, and 346␣
 ↪customers churned out of them
# means a churn percentage of 11.4

346/3010*100
```

[65]:
```
11.495016611295682
```

crosstab( )

[21]:
```
int_plan_data = pd.crosstab(data['international plan'], data['churn'],␣
 ↪margins=True)
            # Using crosstab function from pandas library to compute a simple␣
 ↪cross-tabulation...
            #...of two columns i.e., 'international plan' & 'churn' and saving␣
 ↪the output in varibale 'int_plan_data'
            # ... margins=True means - to show the sum of both values in a new␣
 ↪column 'All'
int_plan_data
```

[21]:
```
churn               False  True   All
international plan
no                   2664   346  3010
yes                   186   137   323
All                  2850   483  3333
```

[22]:
```
#creating a new column 'churn percent'- to show the churn percentage of the␣
 ↪customers who were using international plan

int_plan_data['churn percent'] = int_plan_data[True]/int_plan_data['All']*100
int_plan_data
```

[22]:
```
churn               False  True   All  churn percent
international plan
no                   2664   346  3010      11.495017
yes                   186   137   323      42.414861
All                  2850   483  3333      14.491449
```

[23]:
```
int_plan_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, no to All
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   False          3 non-null      int64
 1   True           3 non-null      int64
 2   All            3 non-null      int64
 3   churn percent  3 non-null      float64
dtypes: float64(1), int64(3)
memory usage: 120.0+ bytes
```

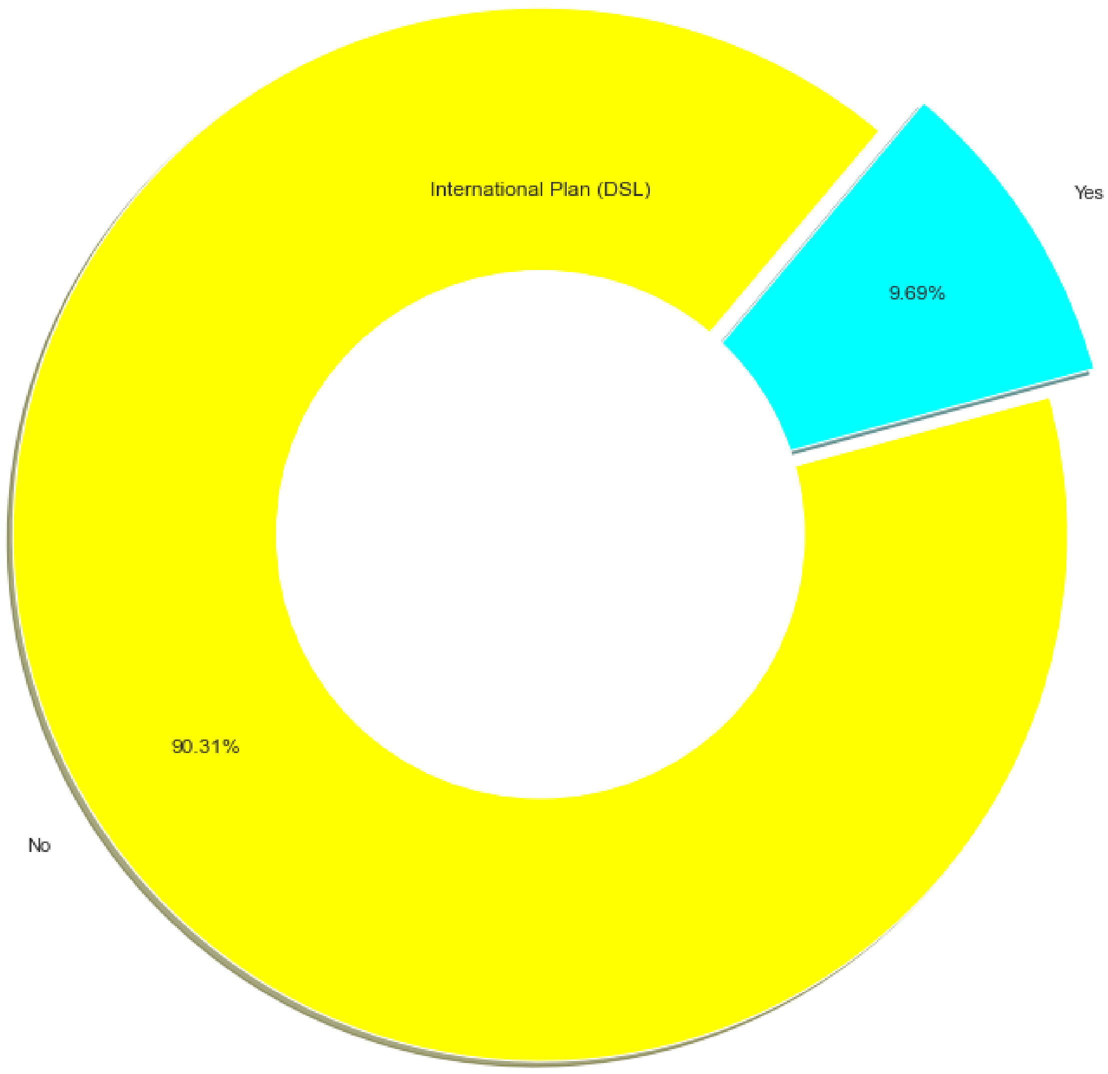
[69]:
```
i = data['international plan'].value_counts()
i

# To show the count of the unique values of the column 'international plan',␣
↪and saving in variable 'i'
```

[69]:
```
no     3010
yes     323
Name: international plan, dtype: int64
```

8.0.1 Donut Chart

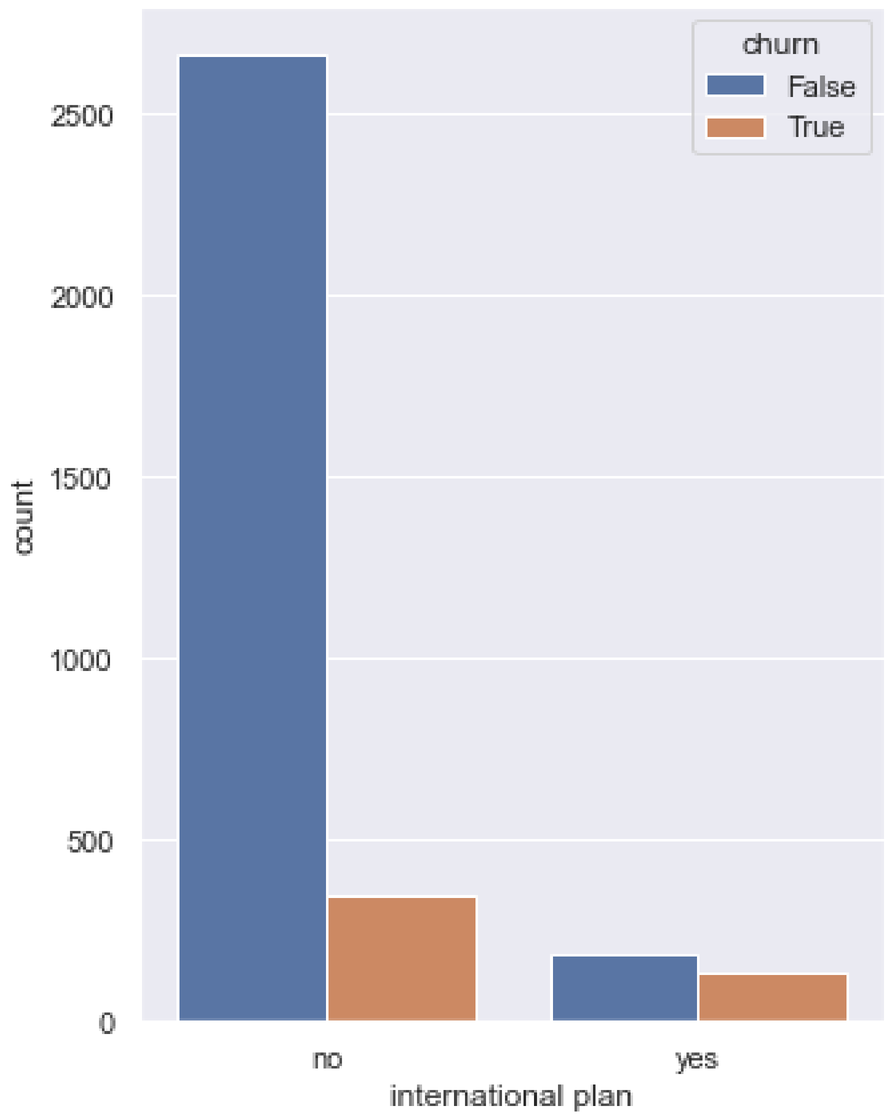
[70]:
```
# Creating a Donut Chart of customers count 'having international plan' and␣
↪'not having international plan'

plt.pie(i, labels=['No', 'Yes'], colors=['yellow','cyan'], startangle=50,␣
↪shadow=True, radius=2,
        explode=(0,0.2), autopct='%1.2f%%',  pctdistance=0.75);

circle = plt.Circle((0,0), 1, color='white')
c = plt.gcf()

c.gca().add_artist(circle)

plt.title("International Plan (DSL)")
plt.rcParams['figure.figsize'] = (5,7)
plt.show()

# Donut chart is a modified Pie chart, with an area from center cut out
#1 First, drawing a pie plot, using Matplotlib library
# The data is taken from the variable 'i' .... labels are given as 'No' & 'Yes'␣
↪...
# ... color is given to each label 'yellow' & 'cyan' ... startangle means the␣
↪angle for slicing, set as 50 ...
```

```
# ... shadow is True means it will drop some shadow of the chart ... radius of␣
↪the circle is set as 2 ...
# ... explode is used to cut the slice out of the figure ...
# ... autopct is used to show the % on the chart upto required decimal points ..
↪.
# ... pctdistance is given for distance of % from the center
#2 Second, using plt.Circle...we will create a circle and save it in the␣
↪variable name 'circle' ...
# ... putting (0,0) by default ... 1 is radius of circle ... and color of␣
↪circle is 'white'
# plt.gcf() is used to get the current figure ... and we are saving it in␣
↪variable 'c'
#3 Third, we will add the 'circle' at the center of pie chart ... using gca().
↪add_artist()
# We have given the title to the chart using plt.title()
# Using rcParams from matplotlib library, setting the size of the figure as 5x7
# plt.show() - To show the chart
```



Countplot

```
[71]: sns.countplot(data['international plan'], hue='churn', data=data)
      plt.rcParams['figure.figsize'] = (10,4)
      plt.show()

      # Drawing a countplot for the column 'international plan' from the dataset␣
       ↪using seaborn library...
      # ...selecting the Churn column for hue
      # 'hue=churn' represents that we want to use column 'churn' for color encoding
      # i.e. color the bars for Churn and Not-Churn differently
      # Blue color representing False (Not Churn) and Orange color representing True␣
       ↪(Churn)
```



9 6. Analyzing the 'Voice Mail Plan' Column

[24]:
```
data.head()
```

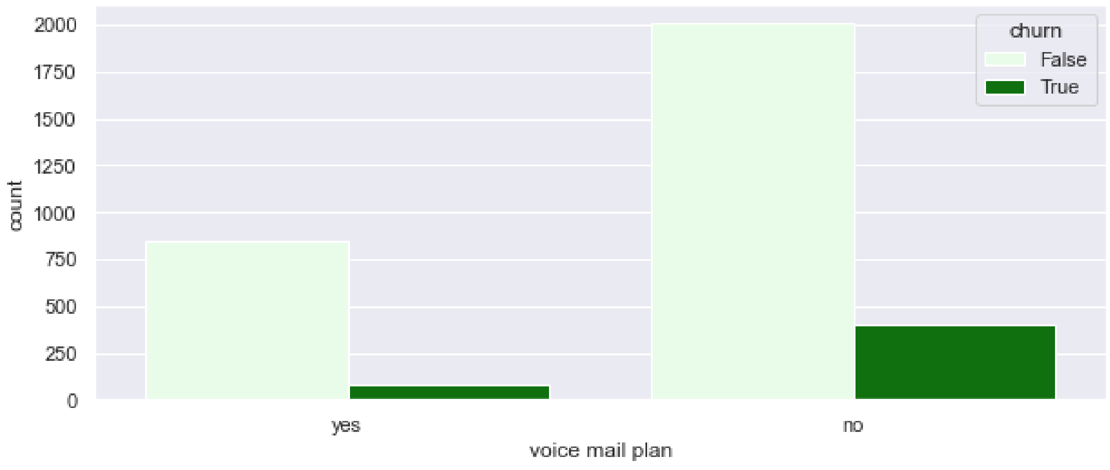
[24]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
```

```
2    NJ              137        415     358-1921                no
3    OH               84        408     375-9999               yes
4    OK               75        415     330-6626               yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

Countplot

[73]:
```python
sns.countplot(data['voice mail plan'], hue="churn", data=data, color='green')
plt.rcParams['figure.figsize']=(10,5)
plt.show()
```

```
# Drawing a countplot for the column 'voice mail plan' from the dataset using␣
↪seaborn library...
# ...selecting the 'churn' column for hue .... color is set as 'green' for bars
# 'hue=churn' represents that we want to use column 'churn' for color encoding
# i.e. color the bars for Churn and Not-Churn differently
# Light Green color representing False (Not Churn) and Dark Green color␣
↪representing True (Churn)
```



[74]:
```
vmplan = pd.crosstab(data['voice mail plan'], data['churn'], margins=True)
                        # Using crosstab function from pandas library to compute␣
↪a simple cross-tabulation...
                        # ...of two columns i.e., 'voice mail plan' & 'churn'␣
↪and saving the output in varibale 'vmplan'
                        # ... margins=True means - to show the sum of both␣
↪values in a new column 'All'
vmplan
```

[74]:
```
churn            False  True   All
voice mail plan
no                2008   403  2411
yes                842    80   922
All               2850   483  3333
```

[75]:
```
vmplan['churn percent'] = vmplan[True] / vmplan['All'] * 100
vmplan

#creating a new column 'churn percent'- to show the churn percentage of the␣
↪customers who were using voice mail plan
```

```
[75]: churn            False  True   All  churn percent
      voice mail plan
      no                2008   403  2411      16.715056
      yes                842    80   922       8.676790
      All               2850   483  3333      14.491449
```

```
[76]: data['voice mail plan'].unique()              # To show all the unique values␣
       ↪of the column 'voice mail plan'
```

```
[76]: array(['yes', 'no'], dtype=object)
```

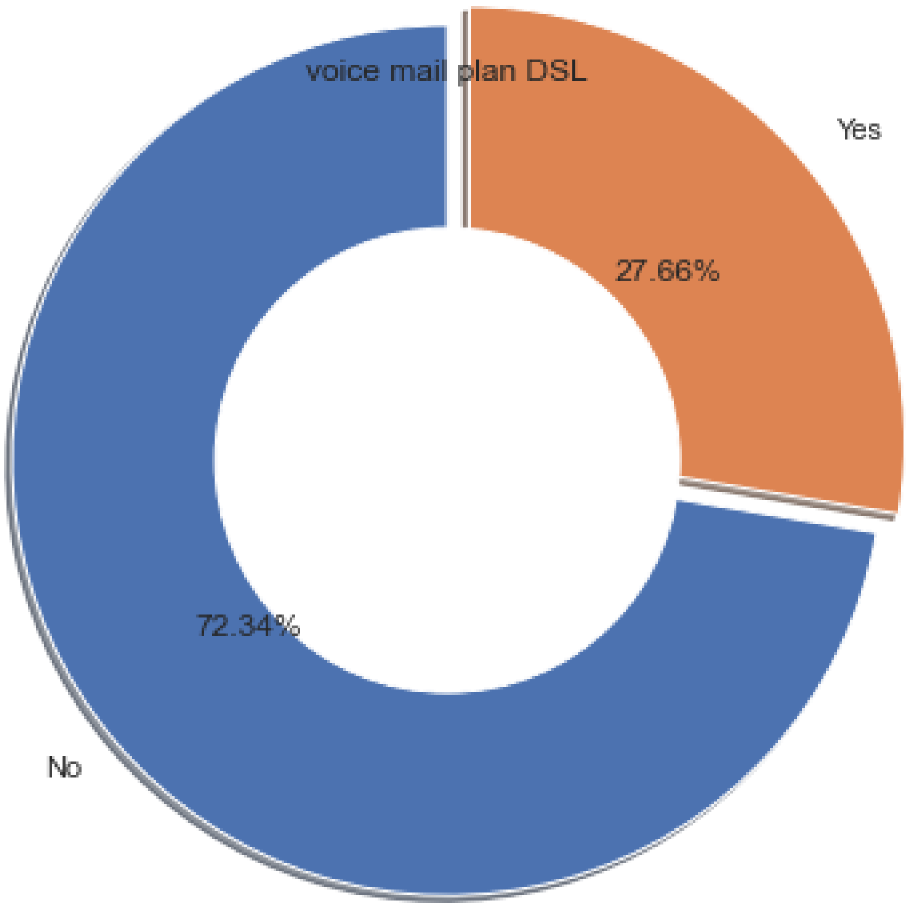
```
[77]: v = data['voice mail plan'].value_counts()
      v
      #To show the occurance/count of all unique values of the column 'voice mail␣
       ↪plan', and saving the output in variable 'v'
```

```
[77]: no     2411
      yes     922
      Name: voice mail plan, dtype: int64
```

9.0.1 Donut Chart

```
[78]: # Creating a Donut Chart of customers count 'having voice mail plan' and 'not␣
       ↪having voice mail plan'

      plt.pie(v, labels=['No','Yes'], startangle=90, shadow=True, radius=1.5,␣
       ↪explode=(0,0.1), autopct='%1.2f%%')

      circle = plt.Circle((0,0),0.8, color='white')
      c = plt.gcf()

      c.gca().add_artist(circle)

      plt.title("voice mail plan DSL")
      plt.show()

      # Donut chart is a modified Pie chart, with an area from center cut out
      #1 First, drawing a pie plot, using Matplotlib library
      # The data is taken from the variable 'v' .... labels are given as 'No' & 'Yes'␣
       ↪...
      # ... startangle means the angle for slicing, set as 90 ...
      # ... shadow is True means it will drop some shadow of the chart ... radius of␣
       ↪the circle is set as 1.5 ...
      # ... explode is used to cut the slice out of the figure ...
      # ... autopct is used to show the % on the chart upto required decimal points ..
       ↪.
```

```
#2 Second, using plt.Circle...we will create a circle and save it in the␣
 ↪variable name 'circle' ...
# ... putting (0,0) by default ... 0.8 is radius of circle ... and color of␣
 ↪circle is 'white'
# plt.gcf() is used to get the current figure ... and we are saving it in␣
 ↪variable 'c'
#3 Third, we will add the 'circle' at the center of pie chart ... using gca().
 ↪add_artist()
# We have given the title to the chart using plt.title()
# plt.show() - To show the chart
```



10 7. Analyzing the 'Number Vmail Messages' Column

[25]:
```
data.head()
```

[25]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[80]:
```
data['number vmail messages'].unique()     # To show the unique values present␣
↪in the column 'number vmail messages'
```

[80]:
```
array([25, 26,  0, 24, 37, 27, 33, 39, 30, 41, 28, 34, 46, 29, 35, 21, 32,
       42, 36, 22, 23, 43, 31, 38, 40, 48, 18, 17, 45, 16, 20, 14, 19, 51,
       15, 11, 12, 47,  8, 44, 49,  4, 10, 13, 50,  9], dtype=int64)
```

[81]:
```
data['number vmail messages'].value_counts().head()

# Showing the occurance/count of top 5 unique values of the column 'number␣
↪vmail messages'
```

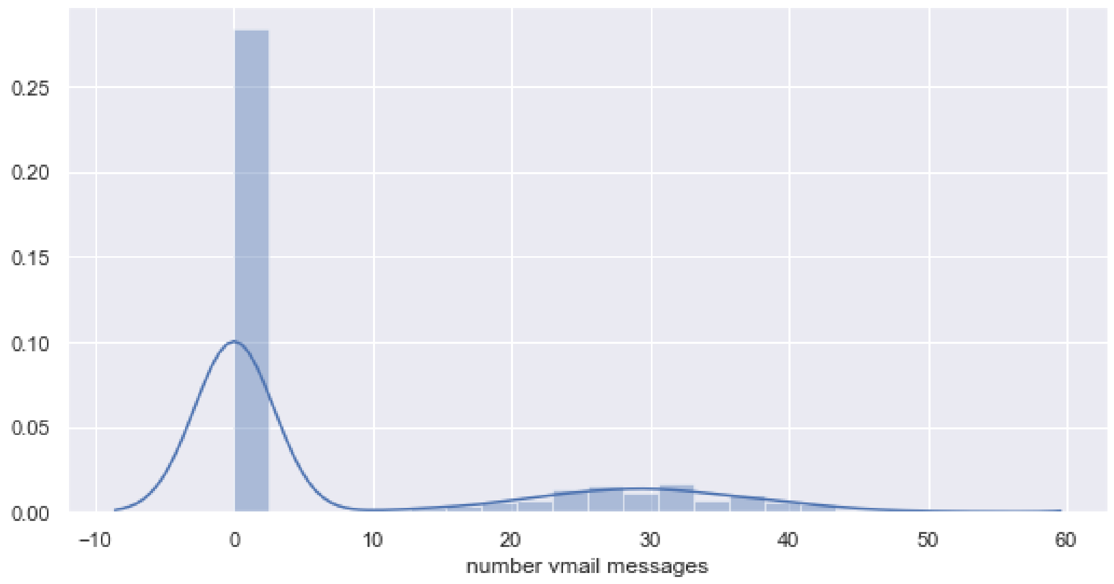
[81]:
```
0     2411
31      60
29      53
28      51
33      46
Name: number vmail messages, dtype: int64
```

[82]:
```
data['number vmail messages'].describe()

# To check the summary statistics of the column. It checks extreme outliers and␣
↪large deviations etc.
# It shows the count of non-null values,mean,std,minimum,maximum,25th,50th,75th␣
↪percentile value.
# Percentile means - how many of the values are less than the given percentile.
```

[82]:
```
count    3333.000000
mean        8.099010
std        13.688365
min         0.000000
25%         0.000000
50%         0.000000
75%        20.000000
max        51.000000
Name: number vmail messages, dtype: float64
```

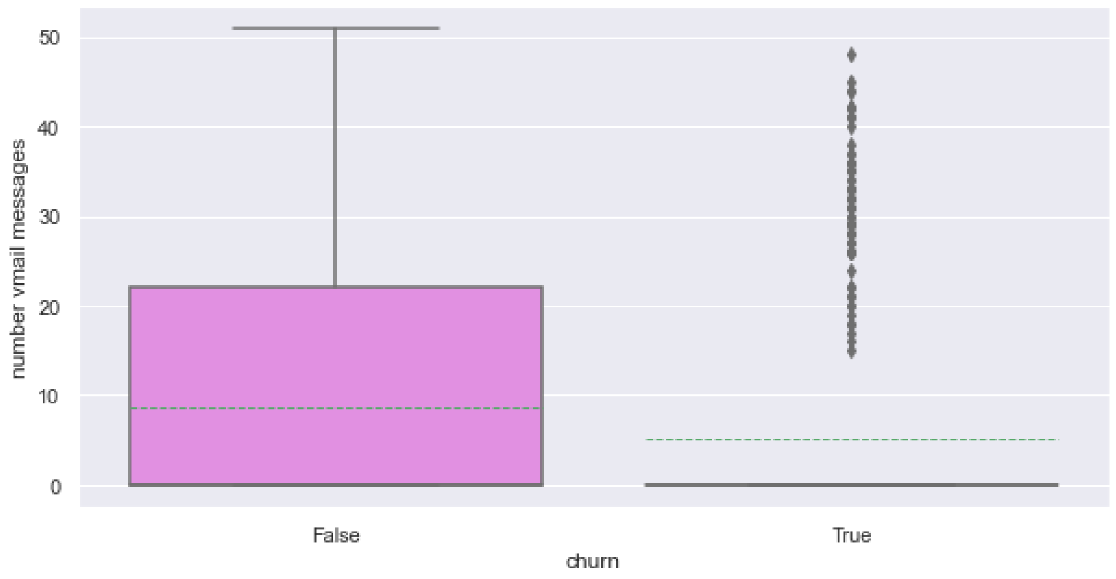
Distribution Plot

[83]:
```
sns.distplot(data['number vmail messages']);

# From Seaborn library we are generating a distribution plot for the column␣
↪'number vmail messages'
```



10.0.1 Box Plot

[84]:
```
sns.boxplot(x='churn', y='number vmail messages', color='violet',␣
 ↪meanline=True, showmeans=True, sym='r+', data=data)
                                 # creating a box plot for the cdolumn 'number␣
 ↪vmail messages' ... grouped by column 'churn'
plt.figure(figsize =(5, 5))   # Setting the size of the figure as 5x5, using␣
 ↪matplotlib library
plt.show()                       # to show the box plot

# In box plot, a box is plotted from Ist quartile to IIIrd quartile
# here, in this boxplot ... we have values from 'churn' column on x-axis and␣
 ↪distribution at y-axis ...
# there are no outliers ... the boundaries are at 0 and 51 ... the minimum␣
 ↪value is at 0 ...
# the first quartile (25%) is at 0 ... the second quartile (50%) is also at 0 ..
 ↪.
# the third quartile (75th percentile) is at 20 ... the maximum value is at 51 .
 ↪..
# color is set as 'violet' ... meanline=True indicates the mean line ...␣
 ↪showmeans=True indicates the mean point ...
# sym='r+' to change the symbol to + sign with red color ... data is taken from␣
 ↪our original dataframe 'data'
```



```
<Figure size 360x360 with 0 Axes>
```

11 8. Analyzing the 'Customer Service Calls' Column

[26]:
```
data.head()
```

[26]:
```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
```

```
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[86]:
```
data['customer service calls'].nunique()

# To show the total number of unique values present in column 'customer service␣
 ↪calls'
```

[86]:
```
10
```

[87]:
```
data['customer service calls'].unique()     # To show all the unique values of␣
 ↪the column 'customer service calls'
```

[87]:
```
array([1, 0, 2, 3, 4, 5, 7, 9, 6, 8], dtype=int64)
```

[88]:
```
data['customer service calls'].value_counts()

# To show the count of the unique values of the column 'customer service calls'␣
 ↪from original dataframe
```

[88]:
```
1    1181
2     759
0     697
```

```
3     429
4     166
5      66
6      22
7       9
8       2
9       2
Name: customer service calls, dtype: int64
```

[89]:
```
churn_data = data[data['churn']]   # creating a new dataframe 'churn_data',␣
 ↪with the records of all churned customers
churn_data
```

[89]:
```
     state  account length  area code phone number international plan  \
10      IN              65        415     329-6603                 no
15      NY             161        415     351-7269                 no
21      CO              77        408     393-7984                 no
33      AZ              12        408     360-1596                 no
41      MD             135        408     383-6029                yes
…     …              …        …          …                  …
3301    CA              84        415     417-1488                 no
3304    IL              71        510     330-7137                yes
3320    GA             122        510     411-5677                yes
3322    MD              62        408     409-1856                 no
3323    IN             117        415     362-5899                 no

     voice mail plan  number vmail messages  total day minutes  \
10                no                      0              129.1
15                no                      0              332.9
21                no                      0               62.4
33                no                      0              249.6
41               yes                     41              173.1
…               …                     …                 …
3301              no                      0              280.0
3304              no                      0              186.1
3320              no                      0              140.0
3322              no                      0              321.1
3323              no                      0              118.4

      total day calls  total day charge  …  total eve calls  \
10                137             21.95  …               83
15                 67             56.59  …               97
21                 89             10.61  …              121
33                118             42.43  …              119
41                 85             29.43  …              107
…                …               …  …              …
3301              113             47.60  …               90
```

```
3304              114              31.64  …             140
3320              101              23.80  …              77
3322              105              54.59  …             122
3323              126              20.13  …              97

      total eve charge  total night minutes  total night calls  \
10               19.42                208.8                111
15               27.01                160.6                128
21               14.44                209.6                 64
33               21.45                280.2                 90
41               17.33                122.2                 78
…                  …                  …                …
3301             17.19                156.8                103
3304             16.88                206.5                 80
3320             16.69                120.1                133
3322             22.57                180.5                 72
3323             21.19                227.0                 56

      total night charge  total intl minutes  total intl calls  \
10                  9.40                12.7                 6
15                  7.23                 5.4                 9
21                  9.43                 5.7                 6
33                 12.61                11.8                 3
41                  5.50                14.6                15
…                  …                  …                …
3301                7.06                10.4                 4
3304                9.29                13.8                 5
3320                5.40                 9.7                 4
3322                8.12                11.5                 2
3323               10.22                13.6                 3

      total intl charge  customer service calls  churn
10                 3.43                       4   True
15                 1.46                       4   True
21                 1.54                       5   True
33                 3.19                       1   True
41                 3.94                       0   True
…                 …                      …    …
3301               2.81                       0   True
3304               3.73                       4   True
3320               2.62                       4   True
3322               3.11                       4   True
3323               3.67                       5   True

[483 rows x 21 columns]
```

```
[90]: churn_data['customer service calls'].value_counts()

      # To show the count/occurance of the unique values of the column 'customer␣
       ↪service calls'
```

```
[90]: 1    122
      0     92
      2     87
      4     76
      3     44
      5     40
      6     14
      7      5
      9      2
      8      1
      Name: customer service calls, dtype: int64
```

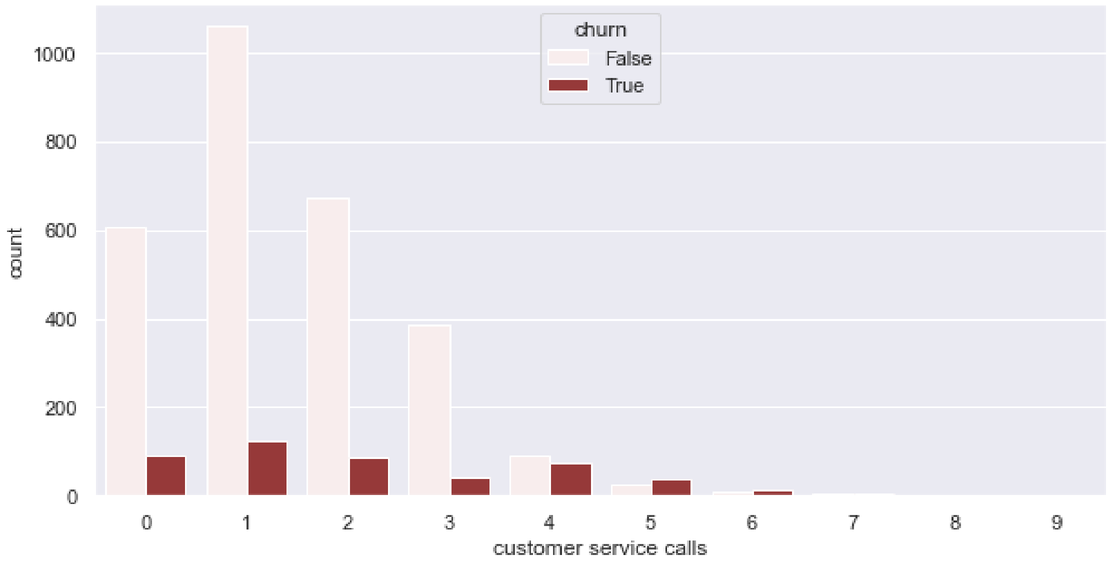
crosstab( )

```
[91]: cscalls = pd.crosstab(data['customer service calls'], data['churn'],␣
       ↪margins=True)
                          # Using crosstab function from pandas library to compute a␣
       ↪simple cross-tabulation...
                          # ...of two columns i.e., 'customer service calls' &␣
       ↪'churn' and saving the output in varibale 'c'
                          # ... margins=True means - to show the sum of both values␣
       ↪in a new column 'All'
      cscalls['churn percent'] = cscalls[True] / cscalls['All'] * 100
                                          # Creating a new column 'churn percent',␣
       ↪which shows the...
                                          # ...percentage of churn customers against␣
       ↪each count of customer service calls
      cscalls
```

```
[91]: churn                   False  True   All  churn percent
      customer service calls
      0                         605    92   697      13.199426
      1                        1059   122  1181      10.330229
      2                         672    87   759      11.462451
      3                         385    44   429      10.256410
      4                          90    76   166      45.783133
      5                          26    40    66      60.606061
      6                           8    14    22      63.636364
      7                           4     5     9      55.555556
      8                           1     1     2      50.000000
      9                           0     2     2     100.000000
      All                      2850   483  3333      14.491449
```

Count Plot

```
[92]: sns.countplot('customer service calls', hue='churn', color='brown', data=data)

      # Drawing a countplot for the column 'customer service calls' from the dataset␣
       ↪using seaborn library...
      # ...selecting the 'churn' column for hue .... color is set as 'black' for bars
      # 'hue=churn' represents that we want to use column 'churn' for color encoding
      # i.e. color the bars for Churn and Not-Churn differently
      # Light color representing False (Not Churn) and Dark color representing True␣
       ↪(Churn)

      plt.show()   # to show the plot
```



12 9. Analyzing the 'Per Minute Charge'

```
[27]: data.head()
```

```
[27]:   state  account length  area code phone number international plan  \
      0    KS             128        415     382-4657                 no
      1    OH             107        415     371-7191                 no
      2    NJ             137        415     358-1921                 no
      3    OH              84        408     375-9999                yes
```

```
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  …  total eve calls  total eve charge  \
0             45.07  …               99             16.78
1             27.47  …              103             16.62
2             41.38  …              110             10.30
3             50.90  …               88              5.26
4             28.34  …              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 21 columns]
```

[94]:

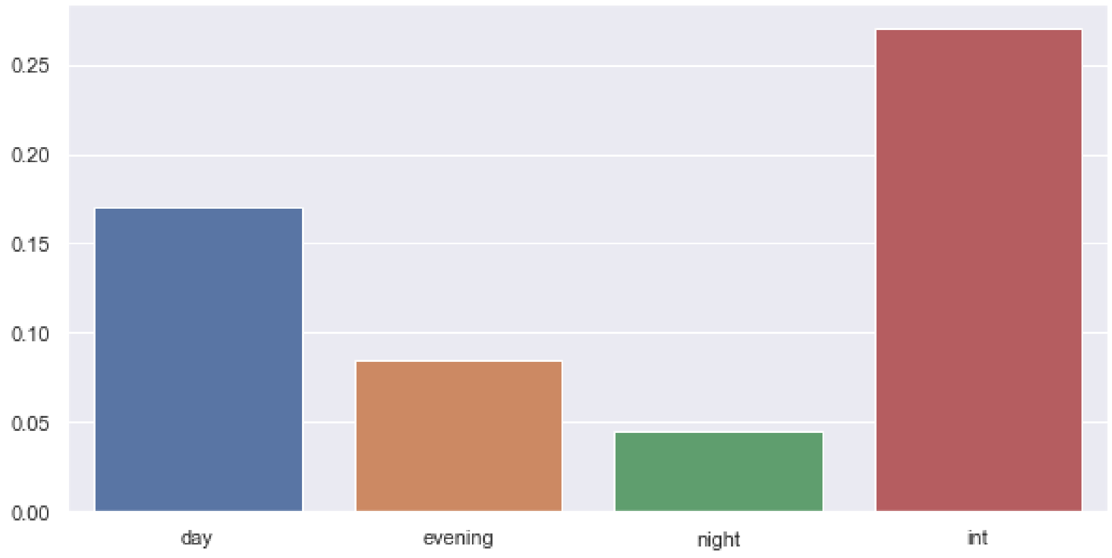
```
dc_pm = data['total day charge'].mean()/data['total day minutes'].mean()
# checking per minute day charge
ec_pm = data['total eve charge'].mean()/data['total eve minutes'].mean()
# checking per minute evening charge
nc_pm = data['total night charge'].mean()/data['total night minutes'].mean()
# checking per minute night charge
intc_pm = data['total intl charge'].mean()/data['total intl minutes'].mean()
# checking per minute international charge
```

```
[95]: print(dc_pm)     # printing per minute day charge
      print(ec_pm)     # printing per minute evening charge
      print(nc_pm)     # printing per minute night charge
      print(intc_pm)   # printing per minute international charge
```

```
0.1700030073913066
0.08500104871485774
0.04500041448440013
0.2700500279887098
```

Bar Plot

```
[96]: sns.barplot(x=['day','evening', 'night', 'int'], y=[dc_pm, ec_pm, nc_pm,␣
        ↪intc_pm])
      plt.show()

      # drawing a bar plot to represent different 'per minute call chagres'
      # on x-axis we have given the naming .... on y-axis we have given the charges
      # plt.show() - to show the plot
```


day
evening
night
int