Lab Assignment 04

The objective of this lab assignment is to explore a dataset that contains information from customers of a telephone company (`data_lab_04.csv`). We will analyze the features in the dataset and try to determine which of these features are good indicators of customer churn (that is, loss of customers).

Instructions:

Complete each task and question by filling in the blanks (`...`) with one or more lines of code or text. Each task and question is worth **0.5 points** (out of **10 points**).

Submission:

This assignment is due **Wednesday, September 25, at 11:59PM (Central Time)**.

This assignment must be submitted on Gradescope as a **PDF file** containing the completed code for each task and the corresponding output. Late submissions will be accepted within **0-12** hours after the deadline with a **0.5-point (5%) penalty** and within **12-24** hours after the deadline with a **2-point (20%) penalty**. No late submissions will be accepted more than 24 hours after the deadline.

This assignment is individual. Offering or receiving any kind of unauthorized or unacknowledged assistance is a violation of the University's academic integrity policies, will result in a grade of zero for the assignment, and will be subject to disciplinary action.

Part 1: Exploring the Dataset

In [1]:
```
# Load libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [2]:
```
# Load dataset
data = pd.read_csv('data_lab_04.csv')
```

In [3]:
```
# Display the first three rows of the dataset
data.head(3)
```

Out[3]:

	State	Account length	Area code	International plan	Voice mail plan	Number voice mail 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	To ni cha
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.0
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.4
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32

Task 01 (of 15): Display the first three rows and the first three columns of the dataset using the `iloc` and `loc` methods. *Hint:* Remember that the `iloc` method is used for indexing by integer position and the `loc` method is used for indexing by label.

In [4]:
```
data.iloc[0:3,0:3]
```

Out[4]:

	State	Account length	Area code
0	KS	128	415
1	OH	107	415
2	NJ	137	415

In [5]:
```
data.loc[[0,1,2], ['State','Account length','Area code']]
```

Out[5]:

	State	Account length	Area code
0	KS	128	415
1	OH	107	415
2	NJ	137	415

Task 02 (of 15): Determine the dimensionality of the dataset. Then, display information (data types, number of values) about the features in the dataset. *Hint:* Use methods `shape` and `info`.

In [6]:
```
data.shape
```

Out[6]:
```
(3333, 20)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
State                          3333 non-null object
Account length                 3333 non-null int64
Area code                      3333 non-null int64
International plan             3333 non-null object
Voice mail plan                3333 non-null object
Number voice mail messages     3333 non-null int64
Total day minutes              3333 non-null float64
Total day calls                3333 non-null int64
Total day charge               3333 non-null float64
Total eve minutes              3333 non-null float64
Total eve calls                3333 non-null int64
Total eve charge               3333 non-null float64
Total night minutes            3333 non-null float64
Total night calls              3333 non-null int64
Total night charge             3333 non-null float64
Total intl minutes             3333 non-null float64
Total intl calls               3333 non-null int64
Total intl charge              3333 non-null float64
Customer service calls         3333 non-null int64
Churn                          3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
```

Question 01 (of 05): How many observations and how many features are in the dataset? What are the data types of the features? Are there any missing values?

Answer: There are 3333 observations with 20 features. The datatypes are object, int64, float64, and bool. There are no missing values because the non-null are equivalent to the number of observations.

Part 2: Transforming the Features

Task 03 (of 15): Change the data type of feature 'Churn' from bool to int64 and change the values of feature 'International plan' from Yes/No to True/False. *Hint:* Use methods `astype` and `map`.

In [8]:
```
data['Churn'] = data['Churn'].astype('int64')
change_values = {'No' : False, 'Yes' : True}
data['International plan'] = data['International plan'].map(change_values)
data.head(3)
```

Out[8]:

	State	Account length	Area code	International plan	Voice mail plan	Number voice mail 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	To ni cha
0	KS	128	415	False	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.0
1	OH	107	415	False	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.4
2	NJ	137	415	False	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32

Task 04 (of 15): Create a new numerical feature named 'Total charge' that contains the sum of the day, evening, and night charges. Then, sort the dataset in descending order by total charge. *Hint:* Use method `sort_values`.

In [9]:
```
data['Total charge'] = data['Total day charge'] + data['Total eve charge'] + data['Total night charg
e']
data = data.sort_values(by='Total charge', ascending=False)
data.head(3)
```

Out[9]:

	State	Account length	Area code	International plan	Voice mail plan	Number voice mail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	...	Total eve charge	Total night minutes	Total night calls	Tot nig char
985	NY	64	415	True	No	0	346.8	55	58.96	249.5	...	21.21	275.4	102	12.39
15	NY	161	415	False	No	0	332.9	67	56.59	317.8	...	27.01	160.6	128	7.23
365	CO	154	415	False	No	0	350.8	75	59.64	216.5	...	18.40	253.9	100	11.43

3 rows × 21 columns

Part 3: Summarizing the Features

Task 05 (of 15): Compute summary statistics for all numerical features and all non-numerical features. *Hint:* Use method `describe` with the appropriate parameters.

```
In [10]: data.describe(include=[np.number])
```

Out[10]:

	Account length	Area code	Number voice mail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total e char
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.0000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.114311	17.083540
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.922625	4.310668
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.000000	14.160000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.000000	17.120000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.000000	20.000000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.000000	30.910000

```
In [11]: data.describe(exclude=[np.number])
```

Out[11]:

	State	International plan	Voice mail plan
count	3333	3333	3333
unique	51	2	2
top	WV	False	No
freq	106	3010	2411

Task 06 (of 15): Group the data by feature 'Churn' and compute summary statistics for all numerical variables again. *Hint:* Use method `groupby`.

In [12]:
```
data.groupby(['Churn']).describe(include=[np.number])
```

Out[12]:

	Account length								Area code		...	Total night charge		Total night minutes	
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean
Churn															
0	2850.0	100.793684	39.88235	1.0	73.0	100.0	127.0	243.0	2850.0	437.074737	...	10.570	17.77	2850.0	200.133193
1	483.0	102.664596	39.46782	1.0	76.0	103.0	127.0	225.0	483.0	437.817805	...	10.795	15.97	483.0	205.231677

2 rows × 136 columns

Task 07 (of 15): Compute the percentage of churned and non-churned customers. *Hint:* Use method `value_counts` with the appropriate parameters.

In [13]:
```
data['Churn'].value_counts(normalize=True)
```

Out[13]:
```
0    0.855086
1    0.144914
Name: Churn, dtype: float64
```

Task 08 (of 15): Compute the mean values of all numerical features for churned and non-churned customers. Notice the differences and similarities between both groups.

In [14]:
```
data[data.Churn == 0].mean()
```

Out[14]:
```
Account length              100.793684
Area code                   437.074737
International plan            0.065263
Number voice mail messages    8.604561
Total day minutes           175.175754
Total day calls             100.283158
Total day charge             29.780421
Total eve minutes           199.043298
Total eve calls             100.038596
Total eve charge             16.918909
Total night minutes         200.133193
Total night calls           100.058246
Total night charge            9.006074
Total intl minutes           10.158877
Total intl calls              4.532982
Total intl charge             2.743404
Customer service calls        1.449825
Churn                         0.000000
Total charge                 55.705404
dtype: float64
```

In [15]:
```
data[data.Churn == 1].mean()
```

Out[15]:
```
Account length                102.664596
Area code                     437.817805
International plan              0.283644
Number voice mail messages      5.115942
Total day minutes             206.914079
Total day calls               101.335404
Total day charge               35.175921
Total eve minutes             212.410145
Total eve calls               100.561077
Total eve charge               18.054969
Total night minutes           205.231677
Total night calls             100.399586
Total night charge              9.235528
Total intl minutes             10.700000
Total intl calls                4.163561
Total intl charge               2.889545
Customer service calls          2.229814
Churn                           1.000000
Total charge                   62.466418
dtype: float64
```

Question 02 (of 05): What is the percentage of churned customers? What is the mean total charge for churned customers? What is the percentage of non-churned customers? What is the mean total charge for non-churned customers

Answer:

85.5 percent, mean of 55.705,

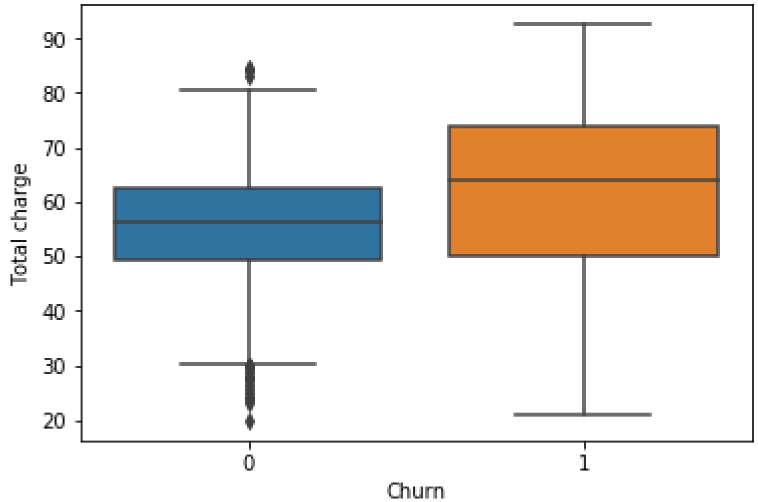
14.5 percent, mean of 62.47

Part 4: Visualizing the Features

Task 09 (of 15): Visualize the summary statistics of churned and non-churned customers for feature 'Total charge'. *Hint:* Use function `seaborn.boxplot()` with the apropriate parameters. Make sure you group customers by feature 'Churn'!

```
In [16]: sns.boxplot(data=data, y='Total charge', x='Churn')
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1a437358>
```



Question 03 (of 05): What do you observe in the plot?

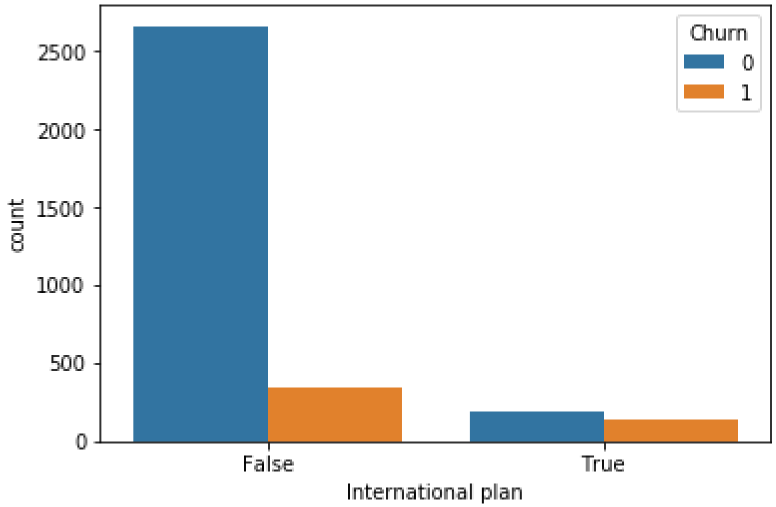
Answer:

There is a wider variance in the total charge for Churn == True

Task 10 (of 15): Visualize the number of churned and non-churned customers in each category of feature 'International plan'. *Hint:* Use function `seaborn.countplot()` with the apropriate parameters. Make sure you group customers by feature 'Churn'!

In [17]:
```
sns.countplot(data=data, hue='Churn', x='International plan')
```

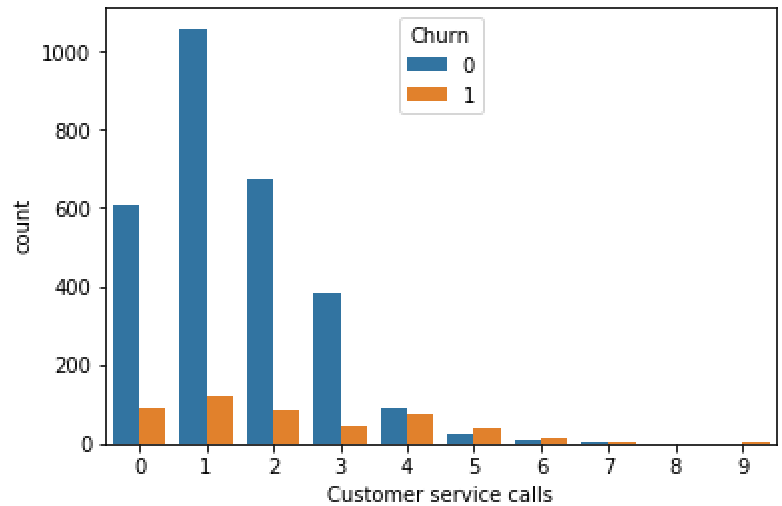
Out[17]: `<matplotlib.axes._subplots.AxesSubplot at 0x1135c0cc0>`



Task 11 (of 15): Visualize the number of churned and non-churned customers in each category of feature 'Customer service calls'. *Hint:* Use function `seaborn.countplot()` with the apropriate parameters. Make sure you group customers by feature 'Churn'!

In [18]:
```
sns.countplot(data=data, hue='Churn', x='Customer service calls')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x113574668>



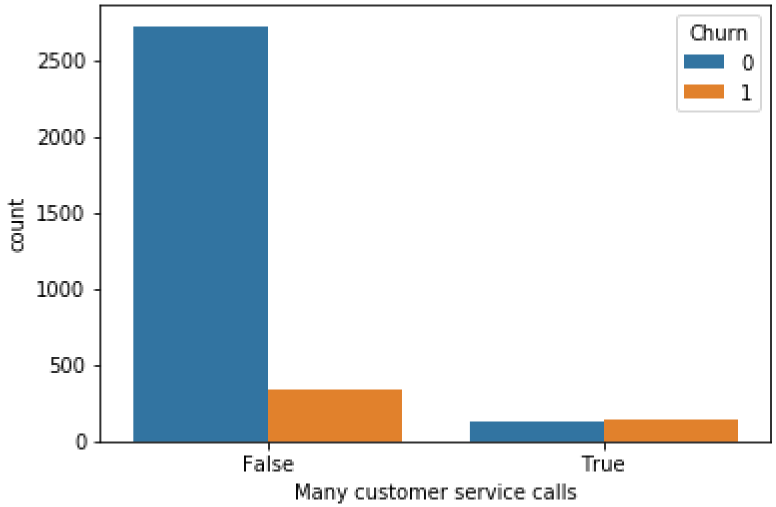
Task 12 (of 15): Create a new Boolean feature named 'Many customer service calls' that indicates whether a user has made more than 3 customer service calls.

In [47]:
```
data['Many customer service calls'] = data['Customer service calls'].apply(lambda x:
                                                                            True if x > 3
                                                                            else False)
```

Task 13 (of 15): Visualize the number of churned and non-churned customers in each category of feature 'Many customer service calls'. *Hint:* Use function `seaborn.countplot()` with the apropriate parameters. Make sure you group customers by feature 'Churn'!

In [48]:
```
sns.countplot(data=data, hue='Churn', x='Many customer service calls')
```

Out[48]: `<matplotlib.axes._subplots.AxesSubplot at 0x113876ef0>`



Question 04 (of 05): What do you observe in the plots?

Answer: In the original plot based on just number of customer calls, we can see that the greatest showing of churn == False is when there are 1-3 calls. This drops dramatically after 4+ calls. We see this represented similarly when we take a look at the second plot, which is 1-3, and 4+. For if False, we know it is 1-3, where the churn == False rate is significantly higher.

Part 5: Making Conclusions

Task 14 (of 15): Compute the churn rate (percentage of churned customers) for customers without international plan and for customers with international plan. *Hint:* Use method `value_counts`.

In [49]:
```
# Compute churn rate for customers without international plan
num_churned = data[(data['International plan'] == 0) & (data['Churn'] == 1)].Churn.value_counts()
num_nonchurned = data[(data['International plan'] == 0) & (data['Churn'] == 0)].Churn.value_counts()
churn_rate = int(num_churned) / (int(num_churned) + int(num_nonchurned)) * 100
print(churn_rate)
```

```
11.495016611295682
```

In [50]:
```
# Compute churn rate for customers with international plan
num_churned = data[(data['International plan'] == 1) & (data['Churn'] == 1)].Churn.value_counts()
num_nonchurned = data[(data['International plan'] == 1) & (data['Churn'] == 0)].Churn.value_counts()
churn_rate = int(num_churned) / (int(num_churned) + int(num_nonchurned)) * 100
print(churn_rate)
```

```
42.414860681114554
```

Task 15 (of 15): Compute the churn rate (percentage of churned customers) for customers with 3 customer service calls or less and for customers with more than 3 service calls. *Hint:* Use method `value_counts`.

In [51]:
```
# Compute churn rate for customers with 3 customer service calls or less
num_churned = data[(data['Many customer service calls'] == 0) & (data['Churn'] == 1)].Churn.value_cou
nts()
num_nonchurned = data[(data['Many customer service calls'] == 0) & (data['Churn'] == 0)].Churn.value_
counts()
churn_rate = int(num_churned) / (int(num_churned) + int(num_nonchurned)) * 100
print(churn_rate)
```

```
11.252446183953033
```

In [52]:
```
# Compute churn rate for customers with more than 3 customer service calls
num_churned = data[(data['Many customer service calls'] == 1) & (data['Churn'] == 1)].Churn.value_cou
nts()
num_nonchurned = data[(data['Many customer service calls'] == 1) & (data['Churn'] == 0)].Churn.value_
counts()
churn_rate = int(num_churned) / (int(num_churned) + int(num_nonchurned)) * 100
print(churn_rate)
```

```
51.68539325842697
```

Question 05 (of 05): What are your final conclusions from the exploration of features 'International plan' and 'Many customer service calls'? What other tasks would you perform to explore this dataset?

Answer: The effect of the number of service calls plan looks like it plays a larger effect than the effect of whether they have an international plan. Based on the above information, we can see that the churn rate increases more based on how many received calls they have.

To explore this dataset further, I would look at some more observations to see what separates customers who churned and those that did not. Additionally, I would have an interest in seeing if the total charge, or individual charges would make an impact on the churn rate of a customer. It would also be interesting to see if the churn rates are significantly different based on regions, since we do have that sort of information.