Lab Assignment 04 (Solutions)

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).

Part 1: Exploring the Dataset

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

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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110

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]:
```python
data.loc[0:3, 'State':'Area code']
```

Out[5]:

	State	Account length	Area code
0	KS	128	415
1	OH	107	415
2	NJ	137	415
3	OH	84	408

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]:
```python
data.shape
```

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

In [7]:
```python
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: The dataset contains 3333 observations and 20 features. There is 1 feature of type bool (i.e., Boolean) (Churn), 3 features of type object (i.e., String) (State, International plan, Voice mail plan), and 16 features of type int64 or float64 (i.e., numerical) (Account length, Area code, Number voice mail messages, etc.). There does not seem to be any explicit missing values, since all features have 3333 values and there are 3333 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
0	KS	128	415	False	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	False	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	False	No	0	243.4	114	41.38	121.2	110

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]:
```python
data['Total charge'] = data['Total day charge'] + data['Total eve charge'] + d
ata['Total night charge']
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	...
0	KS	128	415	False	Yes	25	265.1	110	45.07	197.4	...
1	OH	107	415	False	Yes	26	161.6	123	27.47	195.5	...
2	NJ	137	415	False	No	0	243.4	114	41.38	121.2	...

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]:
```python
data.describe(exclude = ['object', 'bool'])
```

Out[10]:

	Account length	Area code	Number voice mail messages	Total day minutes	Total day calls	Total day charge	Total e minu
count	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	200.980
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.6000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.4000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.3000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.7000

In [11]:
```
data.describe(include = ['object', 'bool'])
```
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()
```
Out[12]:

	Account length								Area code		...	To ch
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75
Churn												
0	2850.0	100.793684	39.88235	1.0	73.0	100.0	127.0	243.0	2850.0	437.074737	...	10
1	483.0	102.664596	39.46782	1.0	76.0	103.0	127.0	225.0	483.0	437.817805	...	10

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]:
```python
data[data['Churn'] == 1].mean()
```

Out[14]:
```
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
```

In [15]:
```python
data[data['Churn'] == 0].mean()
```

Out[15]:
```
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
```

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

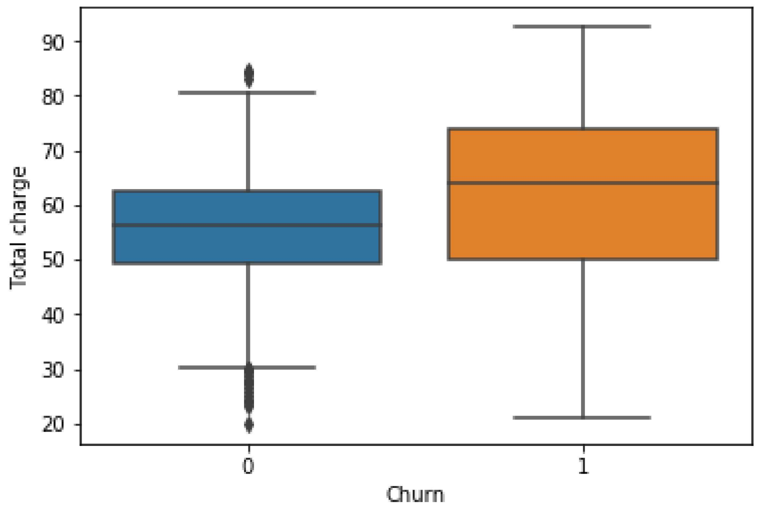
Answer: The percentage of churned customers is 14.49%. The mean total charge for churned customers is 62.47. The percentage of non-churned customers is 85.51%. The mean total charge for non-churned customers is 55.71.

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(x = 'Churn', y = 'Total charge', data = data)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x270e0921c88>
```



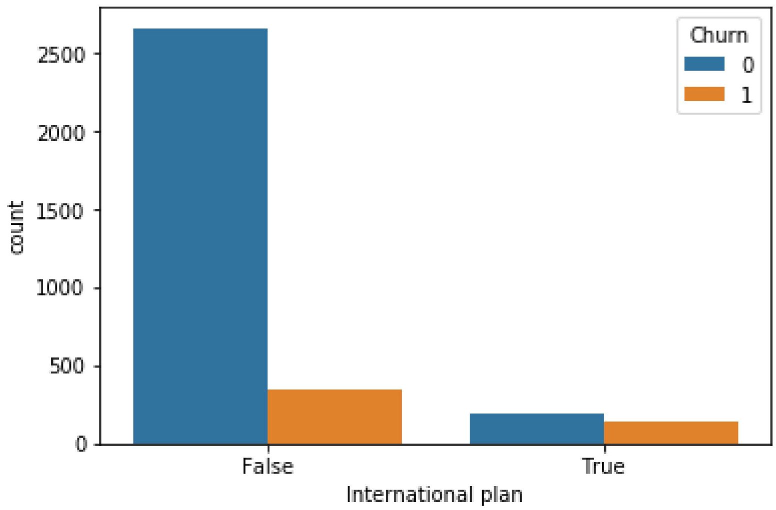
Question 03 (of 05): What do you observe in the plot?

Answer: We observe that the median total charge is higher for churned customers.

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(x = 'International plan', hue = 'Churn', data = data)
```

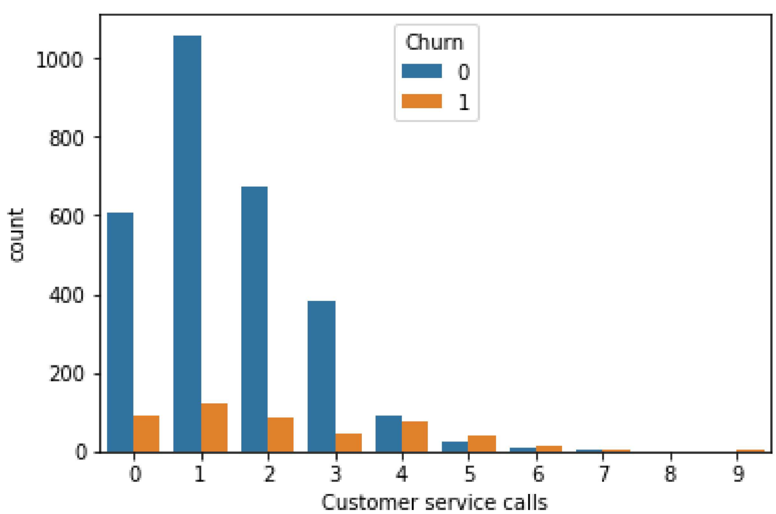
Out[17]: `<matplotlib.axes._subplots.AxesSubplot at 0x270e09d8f98>`



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(x = 'Customer service calls', hue = 'Churn', data = data)
```

Out[18]: `<matplotlib.axes._subplots.AxesSubplot at 0x270e0ac9470>`



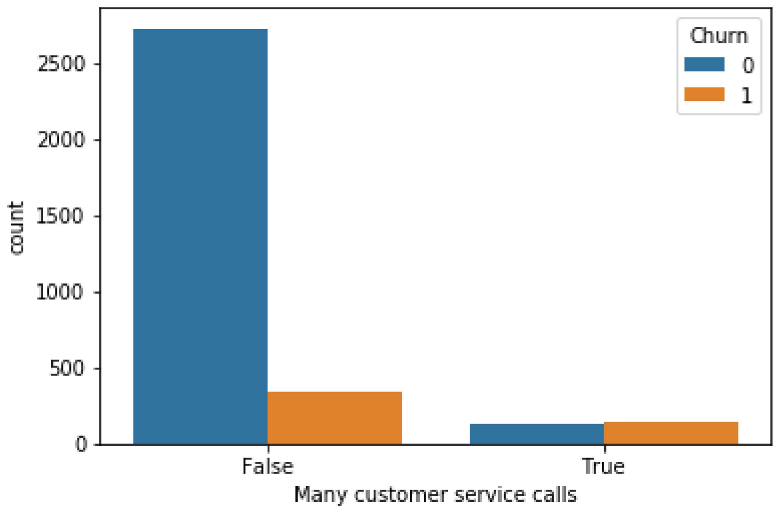
Task 12 (of 15): Create a new Boolean feature named 'Many customer service calls' that indicates whether a user has made more than 3 customer service calls.

In [19]:
```
data['Many customer service calls'] = data['Customer service calls'] > 3
```

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 [20]:
```
sns.countplot(x = 'Many customer service calls', hue = 'Churn', data = data)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x270e1d68898>
```



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

Answer: We observe that the churn rate (percentage of churned customers) seems to be much higher for customers with an international plan and with more than 3 customer service calls.

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 [21]:
```
# Compute churn rate for customers without international plan
num_churned = data[data['International plan'] == False]['Churn'].value_counts
()[1]
num_nonchurned = data[data['International plan'] == False]['Churn'].value_coun
ts()[0]
churn_rate = num_churned/(num_churned + num_nonchurned)
print(churn_rate)
```

```
0.11495016611295682
```

```
In [22]: # Compute churn rate for customers with international plan
num_churned = data[data['International plan'] == True]['Churn'].value_counts()[1]
num_nonchurned = data[data['International plan'] == True]['Churn'].value_counts()[0]
churn_rate = num_churned/(num_churned + num_nonchurned)
print(churn_rate)
```

```
0.4241486068111455
```

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 [23]: # Compute churn rate for customers with 3 customer service calls or less
num_churned = data[data['Many customer service calls'] == False]['Churn'].value_counts()[1]
num_nonchurned = data[data['Many customer service calls'] == False]['Churn'].value_counts()[0]
churn_rate = num_churned/(num_churned + num_nonchurned)
print(churn_rate)
```

```
0.11252446183953033
```

```
In [24]: # Compute churn rate for customers with more than 3 customer service calls
num_churned = data[data['Many customer service calls'] == True]['Churn'].value_counts()[1]
num_nonchurned = data[data['Many customer service calls'] == True]['Churn'].value_counts()[0]
churn_rate = num_churned/(num_churned + num_nonchurned)
print(churn_rate)
```

```
0.5168539325842697
```

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: Features 'International plan' and 'Many customer service calls' seem to be good indicators of customer churn. If our goal is to build a model to predict customer churn, these two features are good candidates for predictors. We should analyze the rest of the features (for example, by creating more box plots and/or count plots) to identify other good indicators of customer churn.