Analysis of Data

```
#Necessary imports
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style(style="darkgrid")
import plotly.express as px
from sklearn.model_selection import train_test_split

# Creating a function for churn rate
def get_churn_rate(array, include_retention=False):
    churns = sum(array)
    churn_rate = churns / len(array)
    if include_retention:
        return churn_rate, 1 - churn_rate
    else:
        return churn_rate
```

Splitting Dataframe to Come up with a Validation Set

```
df = pd.read_csv('/Users/jamesmaikara/Downloads/customer_churn.csv')
df.head()
```

```
  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]
```

```
df_train, df_validation = train_test_split(df, test_size=.1,
random_state=1)
print(df_train.shape, df_validation.shape)
```

```
(2999, 21) (334, 21)
```

```
df_validation.to_csv('/Users/jamesmaikara/Downloads/
validation_set.csv')
df_train.to_csv('/Users/jamesmaikara/Downloads/training_set.csv')
```

Validation set will be used to assess quality of our model at the end. Split it and saved to a csv file for later.

Question 1: Whether calling customer service a sign of customer unhappiness and a cause for churn.

```
#Getting churn and retention rate
get_churn_rate(df_train['churn'], include_retention=True)
```

```
(0.14338112704234746, 0.8566188729576525)
```

```python
#filters the DataFrame df_train to select rows where the 'customer service calls' column has a value greater than 1 and the 'churn' column is equal to True
df_train.loc[(df_train['customer service calls'] > 1) &
(df_train['churn'] == True)]
```

```
     state  account length  area code phone number international plan  \
3304    IL              71        510     330-7137                yes   

2402    NY              77        415     388-9285                 no   

1919    WA             100        408     382-4932                 no   

2980    KS              84        415     335-7144                 no   

3255    RI             138        510     411-6823                yes   

...    ...             ...        ...          ...                ...   

319     SD             128        510     413-9269                yes   

3287    KS             170        415     404-5840                 no   

144     VT             117        408     390-2390                yes   

905     WV             161        415     418-9036                 no   

235     MN             139        510     374-9107                 no   


     voice mail plan  number vmail messages  total day minutes  \
3304              no                      0              186.1   
2402             yes                     33              143.0   
1919              no                      0               70.8   
2980              no                      0              225.9   
3255              no                      0              286.2   
...              ...                    ...                ...   
319              yes                     32              223.5   
3287             yes                     42              199.5   
144               no                      0              167.1   
905               no                      0              191.9   
235               no                      0              134.4   

      total day calls  total day charge  ...  total eve calls  \
3304              114             31.64  ...              140   
2402              101             24.31  ...              102   
1919               94             12.04  ...              102   
2980               86             38.40  ...              105   
3255               61             48.65  ...               60   
...               ...               ...  ...              ...   
```

```
319                81              38.00  ...              74
3287              119              33.92  ...              90
144                86              28.41  ...              87
905               113              32.62  ...              87
235               106              22.85  ...              98

      total eve charge  total night minutes  total night calls  \
3304             16.88                206.5                 80
2402             18.04                104.9                120
1919             18.33                230.8                125
2980             23.43                201.4                108
3255             15.91                146.2                114
...                ...                  ...                ...
319              16.05                154.9                101
3287             11.48                184.6                 49
144              15.09                249.4                132
905               6.03                204.8                107
235              17.96                193.6                125

      total night charge  total intl minutes  total intl calls  \
3304                9.29                13.8                 5
2402                4.72                15.3                 4
1919               10.39                 9.5                 1
2980                9.06                14.3                 3
3255                6.58                11.0                 4
...                  ...                 ...               ...
319                 6.97                 9.4                 2
3287                8.31                10.9                 3
144                11.22                14.1                 7
905                 9.22                13.4                 4
235                 8.71                10.2                 2

      total intl charge  customer service calls  churn
3304               3.73                       4   True
2402               4.13                       5   True
1919               2.57                       6   True
2980               3.86                       3   True
3255               2.97                       2   True
...                 ...                     ...    ...
319                2.54                       2   True
3287               2.94                       4   True
144                3.81                       2   True
905                3.62                       4   True
235                2.75                       5   True

[238 rows x 21 columns]
```

```python
#filters the DataFrame df_train to select rows where the 'customer
service calls' column has a value greater than 1
df_train.loc[df_train['customer service calls'] > 1]
```

```
      state  account length  area code phone number international plan  \
3304    IL              71        510     330-7137                yes   
2402    NY              77        415     388-9285                 no   
756     WY              33        415     331-3202                 no   
133     TX              82        408     400-3446                 no   
366     NC             112        415     334-1872                 no   
...    ...             ...        ...          ...                ...   
144     VT             117        408     390-2390                yes   
960     AR               5        415     380-2758                 no   
2763    NC             116        408     338-7527                 no   
905     WV             161        415     418-9036                 no   
235     MN             139        510     374-9107                 no   

     voice mail plan  number vmail messages  total day minutes  \
3304              no                      0              186.1   
2402             yes                     33              143.0   
756               no                      0              213.9   
133               no                      0              200.3   
366               no                      0              193.3   
...              ...                    ...                ...   
144               no                      0              167.1   
960               no                      0              199.2   
2763             yes                     19              155.7   
905               no                      0              191.9   
235               no                      0              134.4   

      total day calls  total day charge  ...  total eve calls  \
3304              114             31.64  ...              140   
2402              101             24.31  ...              102   
756                88             36.36  ...              119   
133                96             34.05  ...              102   
366                96             32.86  ...              123   
...               ...               ...  ...              ...   
144                86             28.41  ...               87   
960               106             33.86  ...               12   
2763              104             26.47  ...              118   
905               113             32.62  ...               87   
235               106             22.85  ...               98   
```

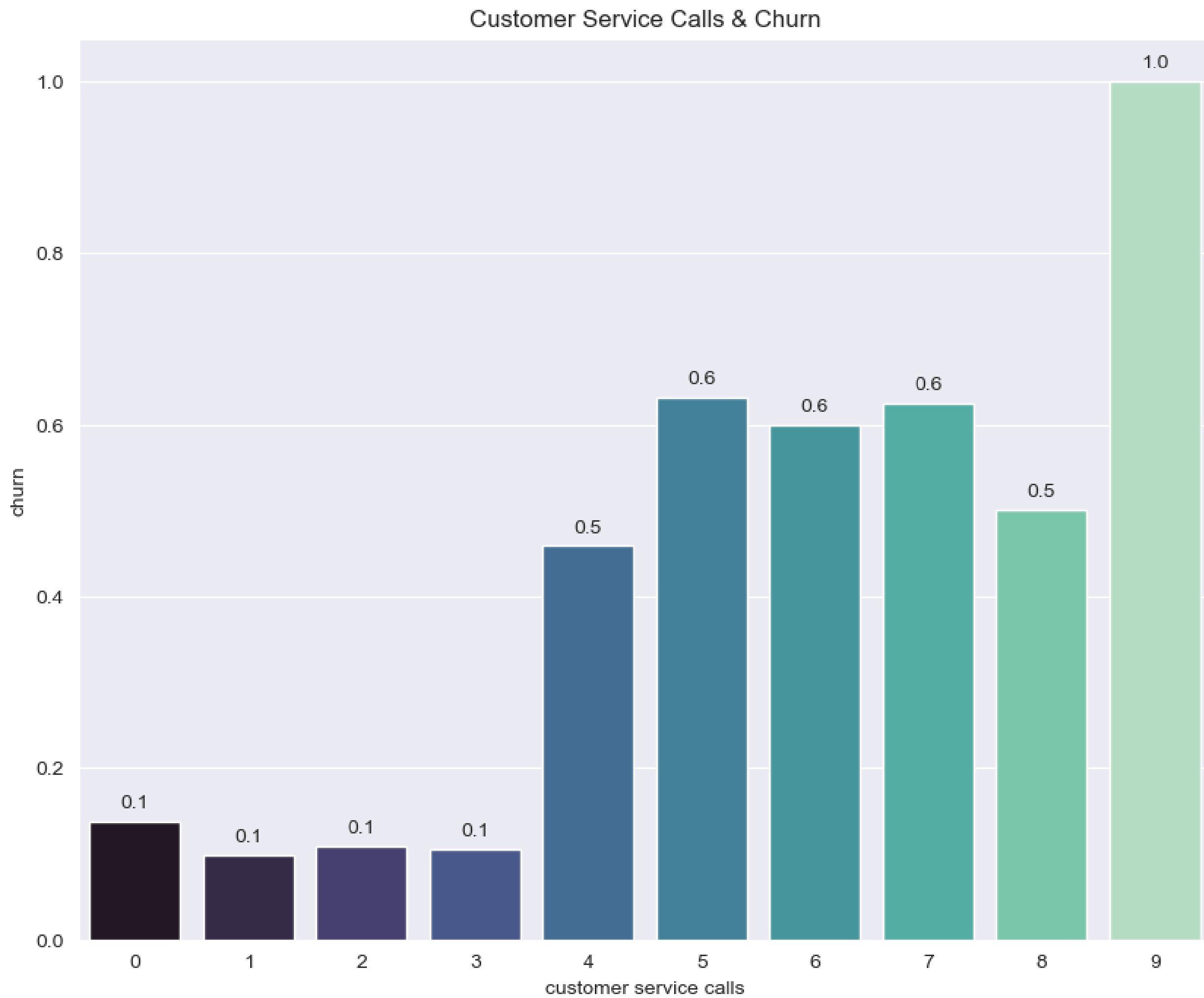
```
      total eve charge  total night minutes  total night calls  \
3304             16.88                206.5                 80
2402             18.04                104.9                120
756              20.38                148.7                 71
133              17.10                206.1                 60
366              22.45                128.6                115
...                ...                  ...                ...
144              15.09                249.4                132
960              15.92                214.0                 85
2763             15.76                192.7                116
905               6.03                204.8                107
235              17.96                193.6                125

      total night charge  total intl minutes  total intl calls  \
3304                9.29                13.8                 5
2402                4.72                15.3                 4
756                 6.69                 9.8                14
133                 9.27                 7.1                 1
366                 5.79                 9.1                 3
...                  ...                 ...               ...
144                11.22                14.1                 7
960                 9.63                13.3                 3
2763                8.67                 8.2                 2
905                 9.22                13.4                 4
235                 8.71                10.2                 2

      total intl charge  customer service calls  churn
3304               3.73                       4   True
2402               4.13                       5   True
756                2.65                       2  False
133                1.92                       4  False
366                2.46                       4  False
...                 ...                     ...    ...
144                3.81                       2   True
960                3.59                       3  False
2763               2.21                       3  False
905                3.62                       4   True
235                2.75                       5   True

[1303 rows x 21 columns]
```

```python
#creates a DataFrame customer_service that groups the 'customer
service calls' column from the df_train DataFrame and calculates the
count of each unique value in the 'churn' column.
customer_service = df_train.groupby('customer service calls')
['churn'].agg(['count'])
customer_service
```

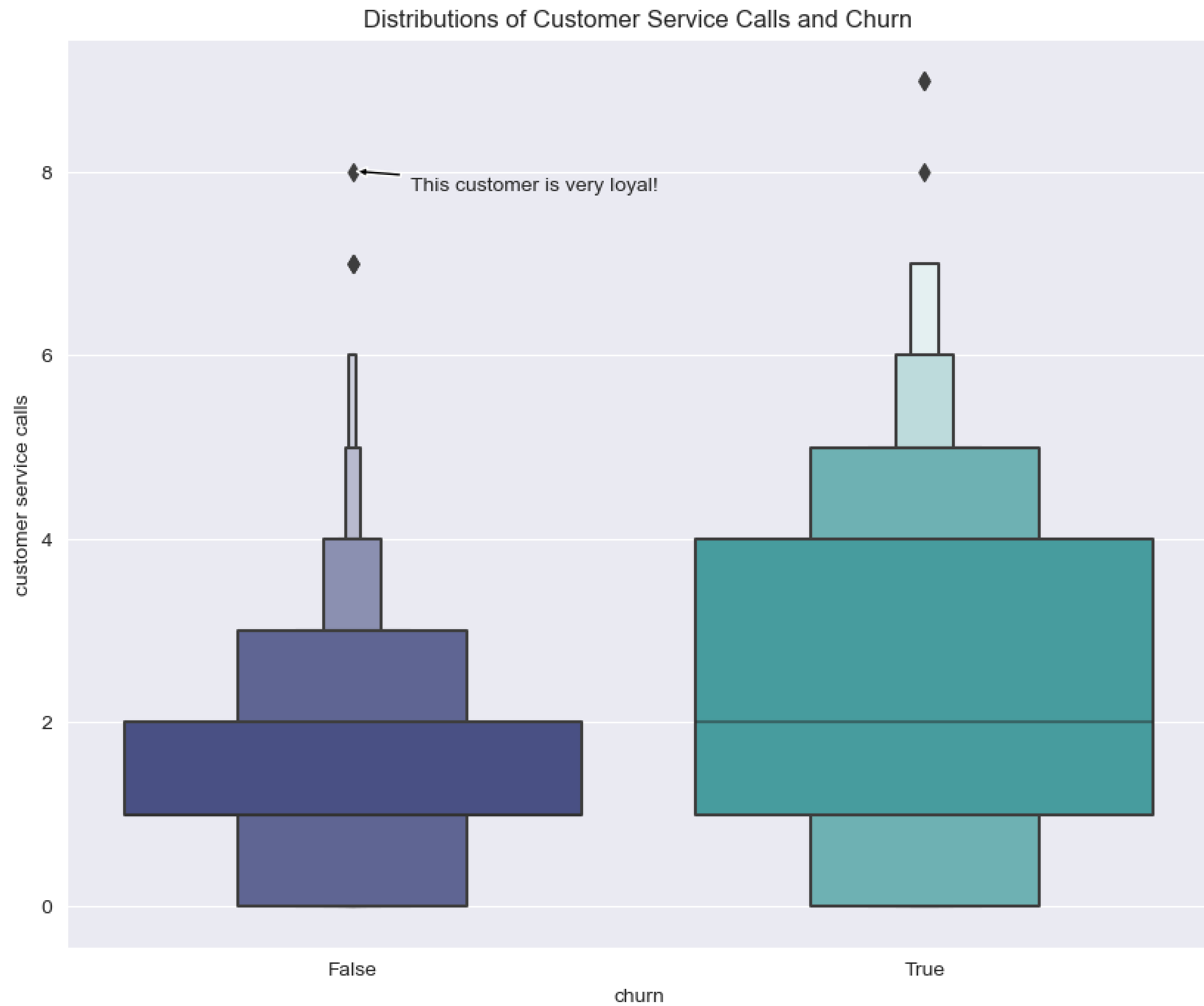
```
                        count
customer service calls
```

0	630
1	1066
2	676
3	392
4	146
5	57
6	20
7	8
8	2
9	2

```python
#creates a bar plot with annotations to show the churn rate for
different numbers of customer service calls.
plt.figure(figsize=(10, 8))
splot = sns.barplot(x='customer service calls', y='churn',
                    data=df_train, palette='mako', ci=None)
# Add annotations to bars
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.1f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha = 'center', va = 'center',
                   xytext = (0, 9),
                   textcoords = 'offset points')
plt.title('Customer Service Calls & Churn')
plt.show()
```




```
#creates a boxen plot using seaborn to visualize the distribution of
'customer service calls' for both churned and non-churned customers in
the df_train DataFrame.
plt.figure(figsize=(10, 8))
sns.boxenplot(x='churn', y='customer service calls',
              data=df_train, palette='mako')

plt.annotate('This customer is very loyal!', xy=(0, 8.01),
xytext=(0.1, 7.8),
            arrowprops=dict(facecolor='black', arrowstyle='simple'))

plt.title('Distributions of Customer Service Calls and Churn')
plt.show()
# That loyal customer is an outlier
```



Findings & Recommendations

In our analysis of the training dataset, we found that the current churn rate stands at approximately 14.5%. Notably, as we delve into the relationship between customer service calls and churn, a compelling pattern emerges. It becomes evident that with an increase in the number of customer service calls, the likelihood of a customer churning also escalates. This relationship is particularly pronounced when a customer has made at least 4 customer service calls, where the likelihood of churn surges from around 10% to a substantial 50%.

However, it's crucial to underscore that the mere occurrence of customer service calls does not serve as a definitive indicator of churn. In fact, a majority of customers who did not churn had made only 1-2 customer service calls. On the other hand, it is noteworthy that a significant proportion of customers who did churn had made 1-4 calls to customer service. Therefore, it is prudent to consider more than 3 calls to customer service as a potential red flag, indicating a heightened risk of customer churn. These insights highlight the importance of proactive measures to address and mitigate customer issues early, particularly when a customer's interactions with customer service exceed this threshold.

Based on these , I recommend relooking at our customer service protocol. It may be useful to offer a larger incentive/discount to customers making more than 3 calls to customer service.

Question 2: Are customers in certain areas more likely to churn?

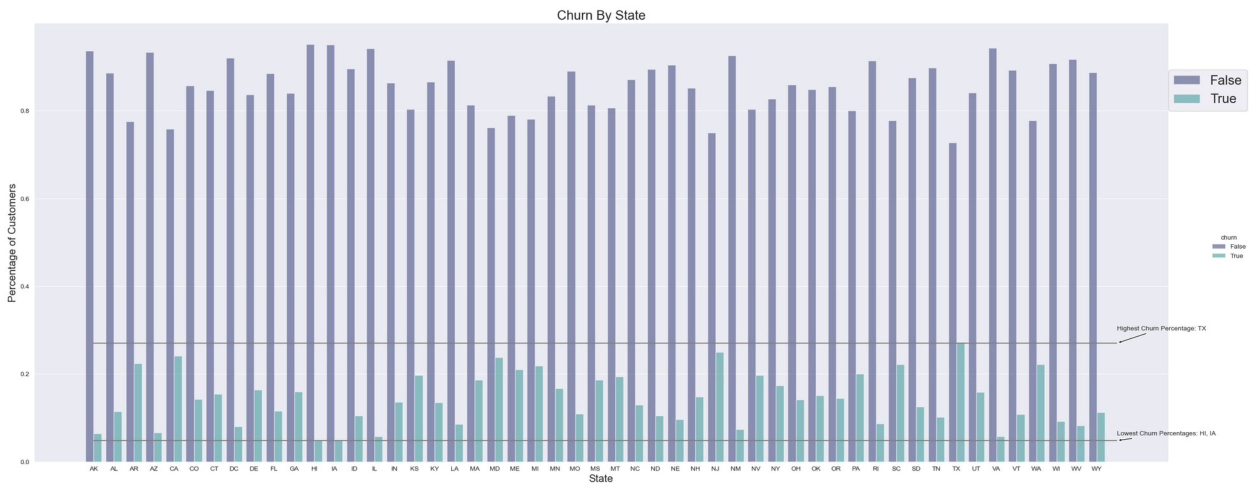
```
display(df_train['state'].unique())
display(df_train['area code'].unique())
```

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

```
array([510, 415, 408])
```

```
#groups the df_train DataFrame by the 'state' column and calculates
the normalized value counts of 'churn' for each state. Then, it
converts this information into a DataFrame
churn_by_state = df_train.groupby('state')
['churn'].value_counts(normalize=True)
churn_by_state = pd.DataFrame(churn_by_state)
churn_by_state.columns = ['value']
churn_by_state = churn_by_state.reset_index()

# Create a bar plot using seaborn
sns.catplot(data=churn_by_state, kind='bar', x='state', y='value',
hue='churn', palette='mako', alpha=0.6, height=10, aspect=2.5)

# Set plot title, y-label, and x-label
plt.title('Churn By State', fontsize=20)
plt.ylabel('Percentage of Customers', fontsize=16)
plt.xlabel('State', fontsize=16)

# Add a legend with custom location and font size
plt.legend(loc=(1, 0.8), fontsize=20)

# Add horizontal lines at specific y-values
plt.hlines(y=0.27, xmin=0, xmax=51, color='gray')
plt.hlines(y=0.048, xmin=0, xmax=51, color='gray')

# Annotate the highest and lowest churn percentages with arrows
plt.annotate('Highest Churn Percentage: TX', xy=(51, 0.27),
xytext=(51, 0.3),
             arrowprops=dict(facecolor='black', arrowstyle='simple'))
```

```python
plt.annotate('Lowest Churn Percentages: HI, IA', xy=(51, 0.048),
xytext=(51, 0.06),
             arrowprops=dict(facecolor='black', arrowstyle='simple'))

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




```python
#filters the churn_by_state DataFrame to select rows where the 'value'
column is greater than 0.23 and the 'churn' column is equal to True
churn_by_state.loc[(churn_by_state['value'] > .23) &
(churn_by_state['churn'] == True)]
```

	state	churn	value
9	CA	True	0.241379
41	MD	True	0.238095
63	NJ	True	0.250000
87	TX	True	0.272727

```python
#filters the churn_by_state DataFrame to select rows where the 'value'
column is less than or equal to 0.05.
churn_by_state.loc[churn_by_state['value'] <= .05]
```

	state	churn	value
23	HI	True	0.04878
25	IA	True	0.05000

```python
#separates the churn_by_state DataFrame into two DataFrames:
lowest_churn_percent and highest_churn_percent
lowest_churn_percentage = churn_by_state.loc[churn_by_state['value']
<= .05]
highest_churn_percentage = churn_by_state.loc[(churn_by_state['value']
> .23) & (churn_by_state['churn'] == True)]

churn_choropleth = churn_by_state.loc[churn_by_state['churn']==True]
fig = px.choropleth(data_frame=churn_choropleth, locations='state',
```

```
locationmode="USA-states",
                    color='value', scope="usa", title='States with
Highest Churn Percentage',
                    color_continuous_scale='Blues')
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"coloraxis":"coloraxis","geo":"geo","hovertemplate":"state=%
{location}<br>value=%{z}<extra></extra>","locationmode":"USA-
states","locations":
["AK","AL","AR","AZ","CA","CO","CT","DC","DE","FL","GA","HI","IA","ID"
,"IL","IN","KS","KY","LA","MA","MD","ME","MI","MN","MO","MS","MT","NC"
,"ND","NE","NH","NJ","NM","NV","NY","OH","OK","OR","PA","RI","SC","SD"
,"TN","TX","UT","VA","VT","WA","WI","WV","WY"],"name":"","type":"choro
pleth","z":[6.382978723404255e-
2,0.11428571428571428,0.22448979591836735,6.666666666666667e-
2,0.2413793103448276,0.14285714285714285,0.15384615384615385,8.0e-
2,0.16363636363636364,0.11538461538461539,0.16,4.878048780487805e-
2,5.0e-2,0.1044776119402985,5.7692307692307696e-
2,0.13636363636363635,0.19696969696969696,0.1346153846153846,8.5106382
9787234e-
2,0.1864406779661017,0.23809523809523808,0.21052631578947367,0.21875,0
.16666666666666666,0.10909090909090909,0.1864406779661017,0.1935483870
967742,0.12903225806451613,0.10526315789473684,9.615384615384616e-
2,0.14814814814814814,0.25,7.407407407407407e-
2,0.19672131147540983,0.17333333333333334,0.14084507042253522,0.150943
3962264151,0.14492753623188406,0.2,8.620689655172414e-
2,0.2222222222222222,0.125,0.10204081632653061,0.2727272727272727,0.15
873015873015872,5.714285714285714e-
2,0.1076923076923077,0.2222222222222222,9.230769230769231e-
2,8.247422680412371e-2,0.11267605633802817]}],"layout":{"coloraxis":
{"colorbar":{"title":{"text":"value"}},"colorscale":
[[0,"rgb(247,251,255)"],[0.125,"rgb(222,235,247)"],
[0.25,"rgb(198,219,239)"],[0.375,"rgb(158,202,225)"],
[0.5,"rgb(107,174,214)"],[0.625,"rgb(66,146,198)"],
[0.75,"rgb(33,113,181)"],[0.875,"rgb(8,81,156)"],
[1,"rgb(8,48,107)"]]},"geo":{"center":{},"domain":{"x":[0,1],"y":
[0,1]},"scope":"usa"},"legend":{"tracegroupgap":0},"template":{"data":
{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
```

```
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
```

```
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
```

{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"States with Highest Churn Percentage"}}}

```
#filters the churn_by_state DataFrame to select states with a churn
rate (value) greater than or equal to 0.2, and where the 'churn'
column is equal to True. Then, it extracts the unique states that meet
these criteria.
high_competition = churn_by_state.loc[(churn_by_state['value'] >= .2)
& (churn_by_state['churn'] == True)]
high_competition['state'].unique()
```

```
array(['AR', 'CA', 'MD', 'ME', 'MI', 'NJ', 'PA', 'SC', 'TX', 'WA'],
      dtype=object)
```

```
# filters the churn_by_state DataFrame to select states with a churn
rate (value) greater than or equal to 0.15 and less than 0.2, and
where the 'churn' column is equal to True. Then, it extracts the
unique states that meet these criteria.
med_high_competition = churn_by_state.loc[(churn_by_state['value'] >=
.15) & (churn_by_state['value'] < .2) & (churn_by_state['churn'] ==
True)]
med_high_competition['state'].unique()
```

```
array(['CT', 'DE', 'GA', 'KS', 'MA', 'MN', 'MS', 'MT', 'NV', 'NY',
'OK',
       'UT'], dtype=object)
```

```
#filters the churn_by_state DataFrame to select states with a churn
rate (value) less than 0.15 and greater than or equal to 0.10, and
where the 'churn' column is equal to True. Then, it extracts the
unique states
medium_competition = churn_by_state.loc[(churn_by_state['value'] <
.15) & (churn_by_state['value'] >= .10) & (churn_by_state['churn'] ==
```

```
True)]
medium_competition['state'].unique()
```

```
array(['AL', 'CO', 'FL', 'ID', 'IN', 'KY', 'MO', 'NC', 'ND', 'NH',
'OH',
       'OR', 'SD', 'TN', 'VT', 'WY'], dtype=object)
```

```
#filters the churn_by_state DataFrame to select states with a churn
rate (value) less than 0.1, and where the 'churn' column is equal to
True.
low_competition = churn_by_state.loc[(churn_by_state['value'] < .1) &
(churn_by_state['churn'] == True)]
low_competition['state'].unique()
```

```
array(['AK', 'AZ', 'DC', 'HI', 'IA', 'IL', 'LA', 'NE', 'NM', 'RI',
'VA',
       'WI', 'WV'], dtype=object)
```

```
#wo functions, categorize_state and create_competition_feat, for
encoding states based on the level of competition and adding a new
'competition' feature to the DataFrame.
def categorize_state(state):

    if state in ['AK', 'AZ', 'DC', 'HI', 'IA', 'IL', 'LA', 'NE', 'NM',
'RI', 'VA', 'WI', 'WV']:
        state = 1
    elif state in ['AL', 'CO', 'FL', 'ID', 'IN', 'KY', 'MO', 'NC',
'ND', 'NH', 'OH', 'OR', 'SD', 'TN', 'VT', 'WY']:
        state = 2
    elif state in ['CT', 'DE', 'GA', 'KS', 'MA', 'MN', 'MS', 'MT',
'NV', 'NY', 'OK', 'UT']:
        state = 3
    else:
        state = 4
    return state


def create_competition(df):

    df['competiton'] = df['state'].apply(categorize_state)
    return df

create_competition(df_train)

df_train.head(5)
```

```
     state  account length  area code phone number international plan
\
2682    DC              55         510     354-5058                yes

3304    IL              71         510     330-7137                yes
```

```
757     UT              112        415     358-5953                 no
2402    NY               77        415     388-9285                 no
792     NV               69        510     397-6789                yes

     voice mail plan  number vmail messages  total day minutes  \
2682              no                      0              106.1
3304              no                      0              186.1
757               no                      0              115.8
2402             yes                     33              143.0
792              yes                     33              271.5

      total day calls  total day charge  ...  total eve charge  \
2682               77             18.04  ...             10.50
3304              114             31.64  ...             16.88
757               108             19.69  ...             20.68
2402              101             24.31  ...             18.04
792                98             46.16  ...             21.54

      total night minutes  total night calls  total night charge  \
2682                 96.4                 92                4.34
3304                206.5                 80                9.29
757                 184.6                 78                8.31
2402                104.9                120                4.72
792                 165.4                 85                7.44

      total intl minutes  total intl calls  total intl charge  \
2682                12.9                 3               3.48
3304                13.8                 5               3.73
757                 13.1                 5               3.54
2402                15.3                 4               4.13
792                  8.2                 2               2.21

      customer service calls  churn  competiton
2682                       0  False           1
3304                       4   True           1
757                        1  False           3
2402                       5   True           3
792                        1   True           3

[5 rows x 22 columns]
```

Findings & Recommendations

Our analysis reveals significant disparities in churn rates across different states. Texas stands out with the highest churn rate, reaching a staggering 27%, while New Jersey, Maryland, and

California also exhibit elevated churn rates, exceeding 23%. In stark contrast, states like Hawaii and Iowa maintain remarkably low churn rates, hovering below 0.05%.

Several factors may underlie these variations in churn rates. First, the level of competition plays a pivotal role. Remote states like Hawaii and Iowa may experience less competition, resulting in reduced customer attrition. Conversely, states such as California, New Jersey, and Texas might boast a more competitive landscape, providing customers with alternative choices when considering a switch. Another plausible factor is the quality of service in regions within high-churn states.

Recommendations

In light of these findings, I propose several strategic considerations:

Competitor Analysis: It is imperative to scrutinize the competitive landscape in states with high churn rates, including Texas, California, and New Jersey. Investigate if competitors offer enticing introductory offers or incentives that entice our customers to churn.

Service Quality Assessment: Conduct an in-depth examination of the cellular network quality and service provision in these high-churn states. Identify potential dead zones or areas with service-related issues, as they might contribute to the elevated churn rates.

Question 3: How people are using their plan and what info it can give about churn?

```
#creates a new DataFrame df_calls by selecting specific columns from the df_train DataFrame, namely 'total day calls,' 'total eve calls,' 'total night calls,' and 'churn.'
df_calls = df_train[['total day calls', 'total eve calls', 'total night calls', 'churn']]
df_calls.head()
```

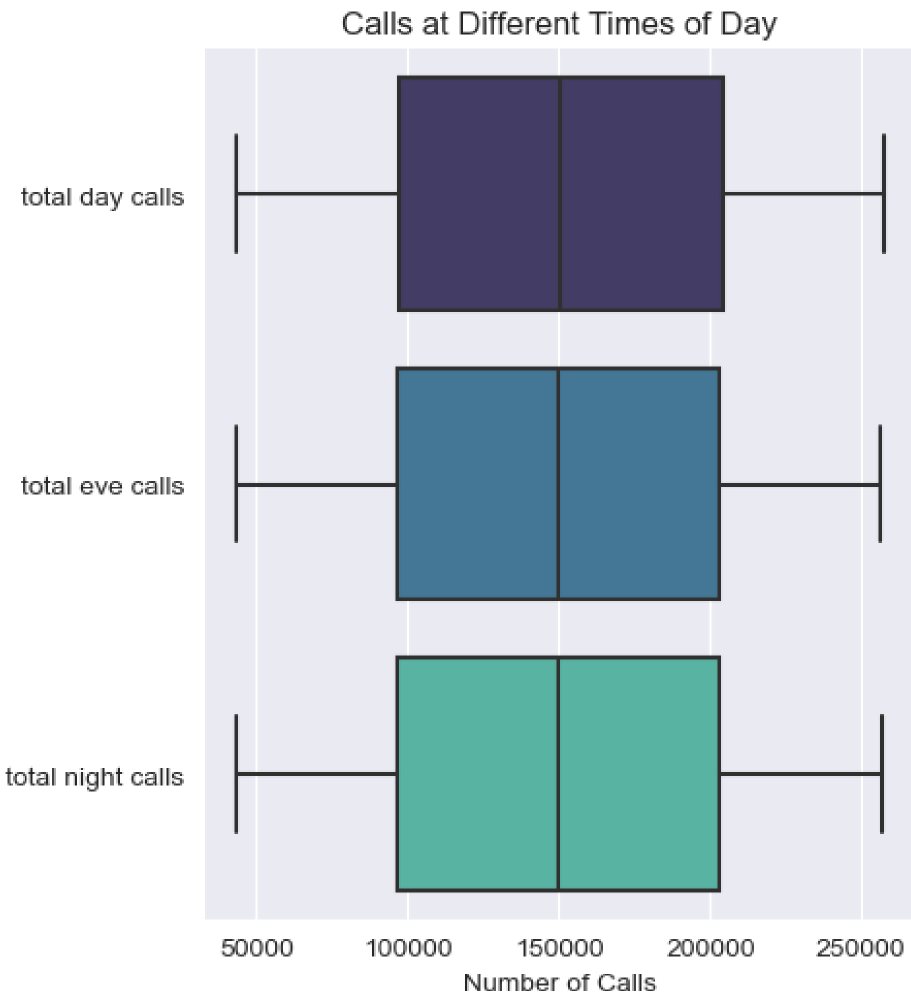
	total day calls	total eve calls	total night calls	churn
2682	77	100	92	False
3304	114	140	80	True
757	108	111	78	False
2402	101	102	120	True
792	98	102	85	True

```
#groups the df_calls DataFrame by the 'churn' column and calculates the sum of the call counts for each category of 'churn.'
calls = df_calls.groupby('churn').sum()
calls.reset_index()
```

	churn	total day calls	total eve calls	total night calls
0	False	257671	256525	256621
1	True	43432	43113	43039

```
#creates a box plot for the call counts at different times of the day, based on the 'churn' categories
sns.catplot(data=calls, orient="h", kind="box", palette='mako')
```

```
plt.title('Calls at Different Times of Day')
plt.xlabel('Number of Calls')
plt.show()
```




```
#calculates and prints calling rates for different parts of the day
and international calls. The calling rate is calculated by dividing
the total charge for a particular type of call. The median is used to
summarize these rates.
day_rate = (df_train['total day charge'] / df_train['total day
minutes']).median()
eve_rate = (df_train['total eve charge'] / df_train['total eve
minutes']).median()
night_rate = (df_train['total night charge'] / df_train['total night
minutes']).median()
intl_rate = (df_train['total intl charge'] / df_train['total intl
minutes']).median()

names = ['Day Rate', 'Eve Rate', 'Night Rate', 'Intl Rate']
```

```
for name, rate in zip(names, [day_rate, eve_rate, night_rate,
intl_rate]):
    print(f'{name}:', round(rate, 2))
```

```
Day Rate: 0.17
Eve Rate: 0.09
Night Rate: 0.04
Intl Rate: 0.27
```

```
#creates two separate DataFrames: df_intl and df_non_intl. These
DataFrames contain subsets of data from the original df_train
DataFrame, with specific conditions based on the 'international plan'
column.
df_intl = df_train.loc[df_train['international plan'] == 'yes']
df_non_intl = df_train.loc[df_train['international plan'] == 'no']

#calculates and prints the median international calling rates for two
groups of customers based on whether they have an international plan
or not.
intl_plan_rate = (df_intl['total intl charge'] / df_intl['total intl
minutes']).median()
non_intl_plan_rate = (df_non_intl['total intl charge'] /
df_non_intl['total intl minutes']).median()
print(intl_plan_rate, non_intl_plan_rate)
```

```
0.27 0.270063694267516
```

Notice the similarity!

```
df_intl.head(5)
```

```
     state  account length  area code phone number international plan  \
2682    DC              55        510     354-5058                 yes

3304    IL              71        510     330-7137                 yes

792     NV              69        510     397-6789                 yes

933     KY              74        510     368-7555                 yes

1804    CT             125        415     409-7523                 yes


     voice mail plan  number vmail messages  total day minutes  \
2682              no                      0              106.1
3304              no                      0              186.1
792              yes                     33              271.5
933               no                      0              125.8
1804              no                      0              187.3
```

```
      total day calls  total day charge  ...  total eve charge  \
2682               77             18.04  ...             10.50
3304              114             31.64  ...             16.88
792                98             46.16  ...             21.54
933               103             21.39  ...             17.65
1804              118             31.84  ...             13.66

      total night minutes  total night calls  total night charge  \
2682                 96.4                 92                4.34
3304                206.5                 80                9.29
792                 165.4                 85                7.44
933                 207.4                143                9.33
1804                263.8                112               11.87

      total intl minutes  total intl calls  total intl charge  \
2682                12.9                 3               3.48
3304                13.8                 5               3.73
792                  8.2                 2               2.21
933                 14.1                 4               3.81
1804                 9.6                 2               2.59

      customer service calls  churn  competiton
2682                       0  False          1
3304                       4   True          1
792                        1   True          3
933                        1   True          2
1804                       0   True          3

[5 rows x 22 columns]
```

```
intl_churn =
df_intl['churn'].value_counts(normalize=True).reset_index().rename(col
umns={'index': 'churn', 'churn': 'percentage'})
non_intl_churn =
df_non_intl['churn'].value_counts(normalize=True).reset_index().rename
(columns={'index': 'churn', 'churn': 'percentage'})

display(intl_churn)
display(non_intl_churn)
```

```
   percentage  proportion
0       False    0.571918
1        True    0.428082
```

```
   percentage  proportion
0       False    0.887329
1        True    0.112671
```

```
df_train.groupby('international plan')['total intl calls'].mean()
```

```
international plan
no     4.482453
yes    4.623288
Name: total intl calls, dtype: float64
```

```
#groups the df_train DataFrame by the 'international plan' column and
calculates the mean of the 'total international charge' for each
group. It shows the average international charges for customers with
and without international plans.
df_train.groupby('international plan')['total intl charge'].mean()
```

```
international plan
no     2.753170
yes    2.878459
Name: total intl charge, dtype: float64
```

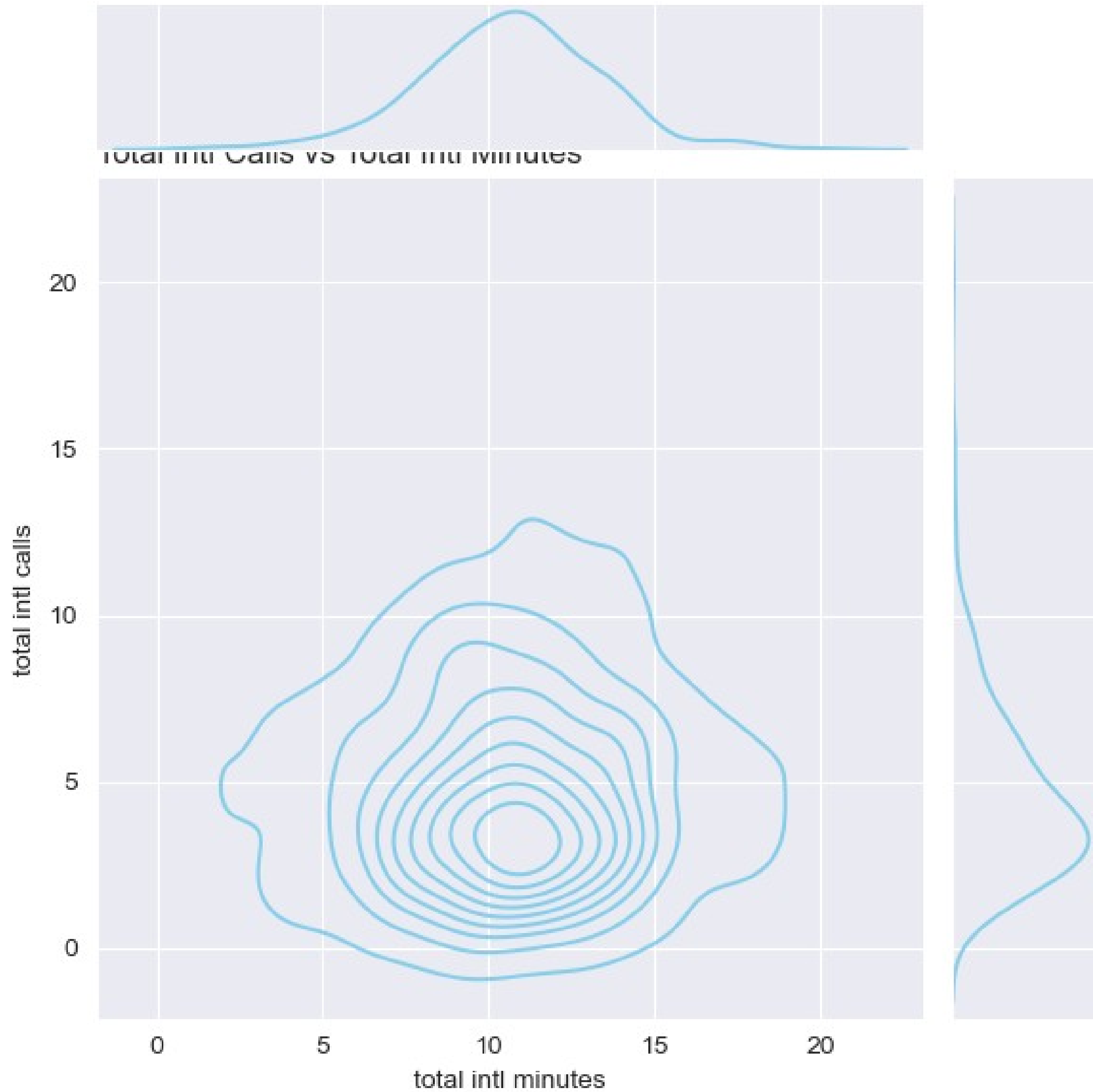
```
# Create functions to create new features so we can use them later in
a pipeline

def create_total_calls_column(df):

    df['total calls'] = df['total day calls'] + df['total eve calls']
+ df['total night calls'] + df['total intl calls']
    return df

def create_minutes_per_intl_call(df):

    df['avg minutes per intl call'] = df['total intl minutes'] /
df['total intl calls']
    return df

# Create 2 new features on the training data set
df_train = create_minutes_per_intl_call(df_train)
df_train = create_total_calls_column(df_train)

df_train.head(10)
```

```
     state  account length  area code phone number international plan
\
2682    DC              55        510     354-5058                yes

3304    IL              71        510     330-7137                yes

757     UT             112        415     358-5953                 no

2402    NY              77        415     388-9285                 no

792     NV              69        510     397-6789                yes

933     KY              74        510     368-7555                yes
```

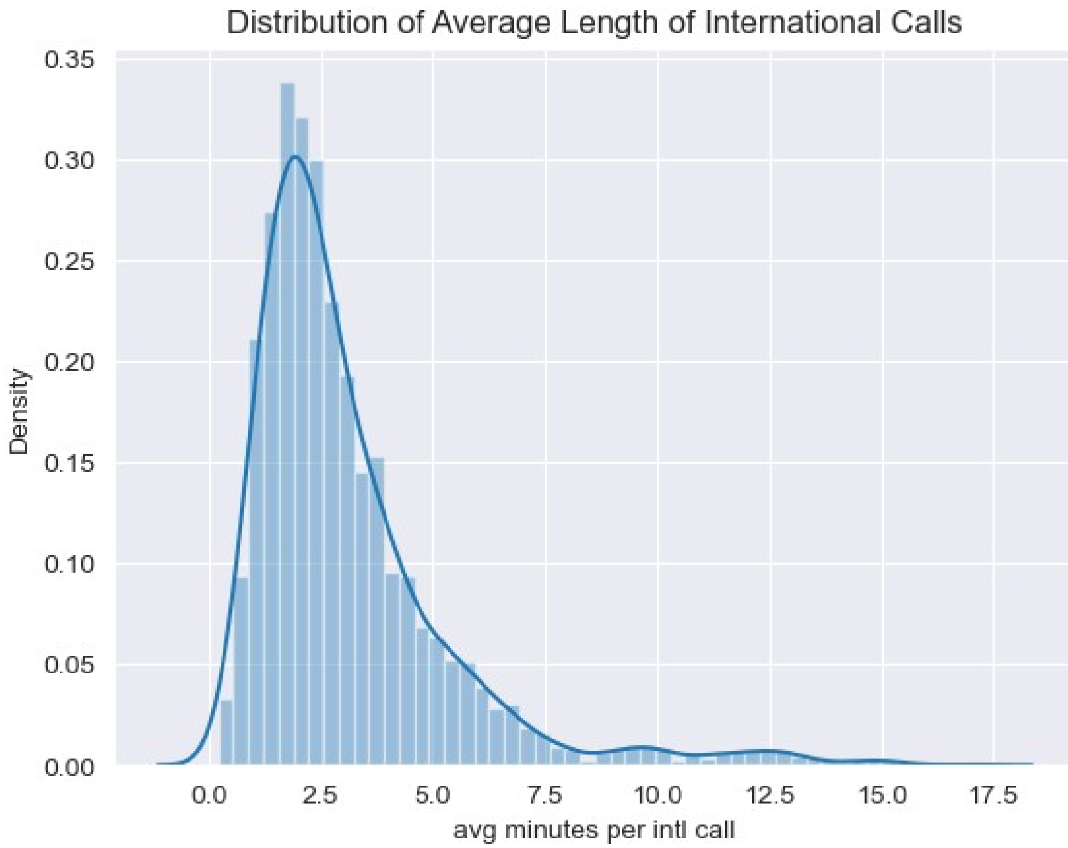
```
756     WY                33          415     331-3202                  no

804     MT                72          415     398-8385                  no

1966    NE                77          415     350-1532                  no

1804    CT               125          415     409-7523                 yes


     voice mail plan  number vmail messages  total day minutes  \
2682              no                      0              106.1
3304              no                      0              186.1
757               no                      0              115.8
2402             yes                     33              143.0
792              yes                     33              271.5
933               no                      0              125.8
756               no                      0              213.9
804               no                      0              253.0
1966              no                      0              169.4
1804              no                      0              187.3

      total day calls  total day charge  ...  total night calls  \
2682               77             18.04  ...                 92
3304              114             31.64  ...                 80
757               108             19.69  ...                 78
2402              101             24.31  ...                120
792                98             46.16  ...                 85
933               103             21.39  ...                143
756                88             36.36  ...                 71
804                73             43.01  ...                 89
1966              102             28.80  ...                 89
1804              118             31.84  ...                112

      total night charge  total intl minutes  total intl calls  \
2682                4.34                12.9                 3
3304                9.29                13.8                 5
757                 8.31                13.1                 5
2402                4.72                15.3                 4
792                 7.44                 8.2                 2
933                 9.33                14.1                 4
756                 6.69                 9.8                14
804                 9.49                 9.8                 4
1966               10.54                 2.0                 7
1804               11.87                 9.6                 2

      total intl charge  customer service calls  churn  competiton  \
2682               3.48                       0  False           1
3304               3.73                       4   True           1
757                3.54                       1  False           3
2402               4.13                       5   True           3
```

```
792                  2.21                          1   True           3
933                  3.81                          1   True           2
756                  2.65                          2  False           2
804                  2.65                          0  False           3
1966                 0.54                          1  False           1
1804                 2.59                          0   True           3

      avg minutes per intl call  total calls
2682                   4.300000          272
3304                   2.760000          339
757                    2.620000          302
2402                   3.825000          327
792                    4.100000          287
933                    3.525000          346
756                    0.700000          292
804                    2.450000          244
1966                   0.285714          342
1804                   4.800000          343

[10 rows x 24 columns]
```

```
#creates a kernel density estimation (KDE) joint plot using seaborn to
visualize the relationship between the 'total international minutes'
and 'total international calls' for customers with an international
plan ('df_intl')
sns.jointplot(x=df_intl["total intl minutes"], y=df_intl["total intl
calls"], kind='kde', color="skyblue")
plt.title('Total Intl Calls vs Total Intl Minutes', loc='left')
plt.show()
```



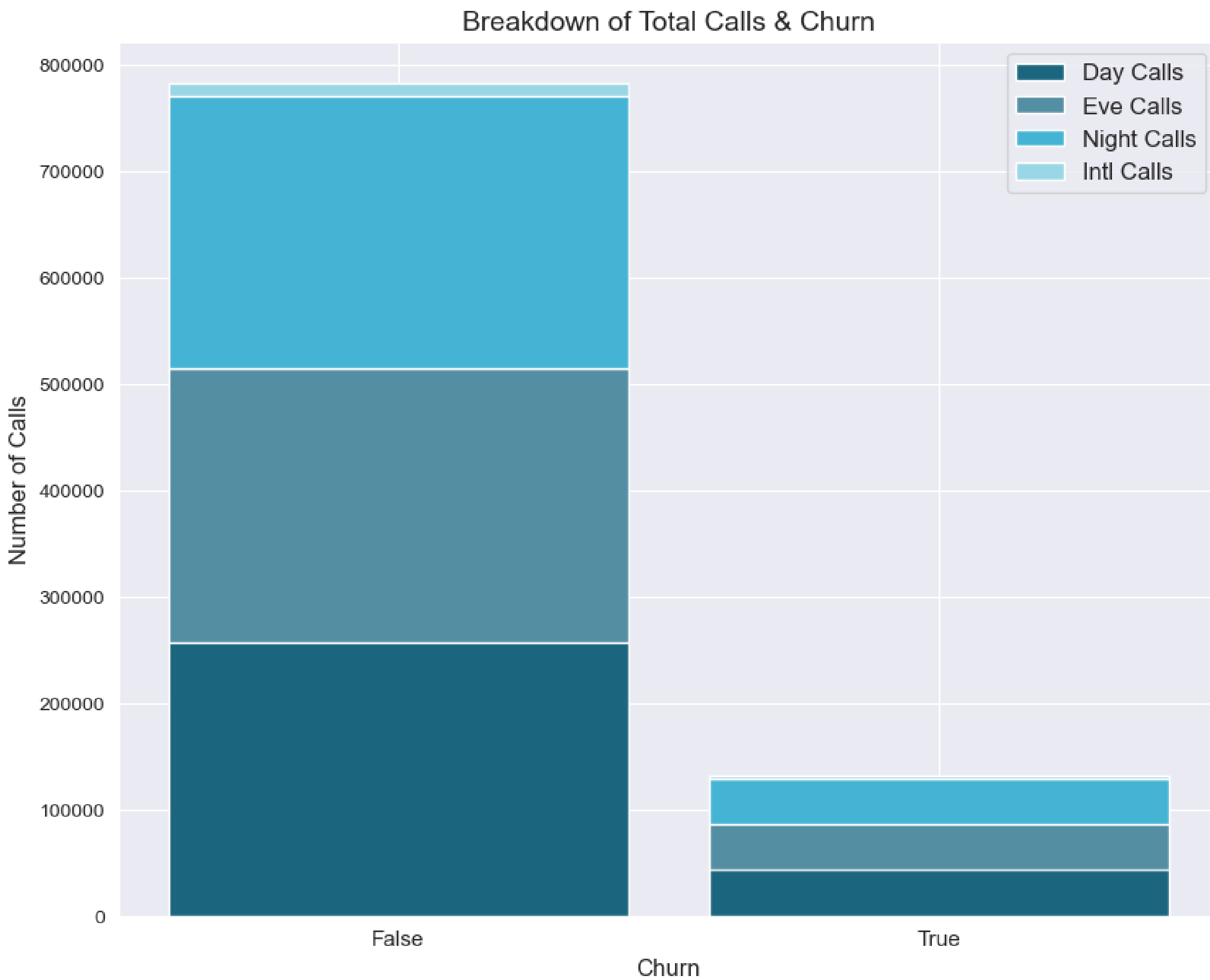

```
#create a distribution plot (histogram) to visualize the distribution
of the 'avg minutes per intl call' feature in the df_train DataFrame.
sns.distplot(df_train['avg minutes per intl call'])
plt.title('Distribution of Average Length of International Calls')
plt.show()
```




```
stacked_bar_data = df_train.groupby('churn').sum().reset_index()
stacked_bar_data

   churn                                                  state  account
length  \
0  False  DCUTWYMTNEMOSCDECOINNMKYTXFLNDALNCILMAVAVTSCKY...
258560
1   True  ILNYNVKYCTOKMDWAORGANMALKSRIOKMTWIINDEDEVTNVTX...
43692

   area code                                               phone number  \
0    1122642  354-5058358-5953331-3202398-8385350-1532375-33...
1     188415  330-7137388-9285397-6789368-7555409-7523352-69...

                                  international plan  \
0  yesnononononoyesnonononoyesnononoyesnonononoyesn...
1  yesnoyesyesyesnoyesnononononononoyesnononononoye...

                                     voice mail plan  number vmail
messages  \
0  nononononononononononononononoyesyesnononoyesnoy...
22015
1  noyesyesnononoyesnononoyesnonononononononoyesn...
2194
```

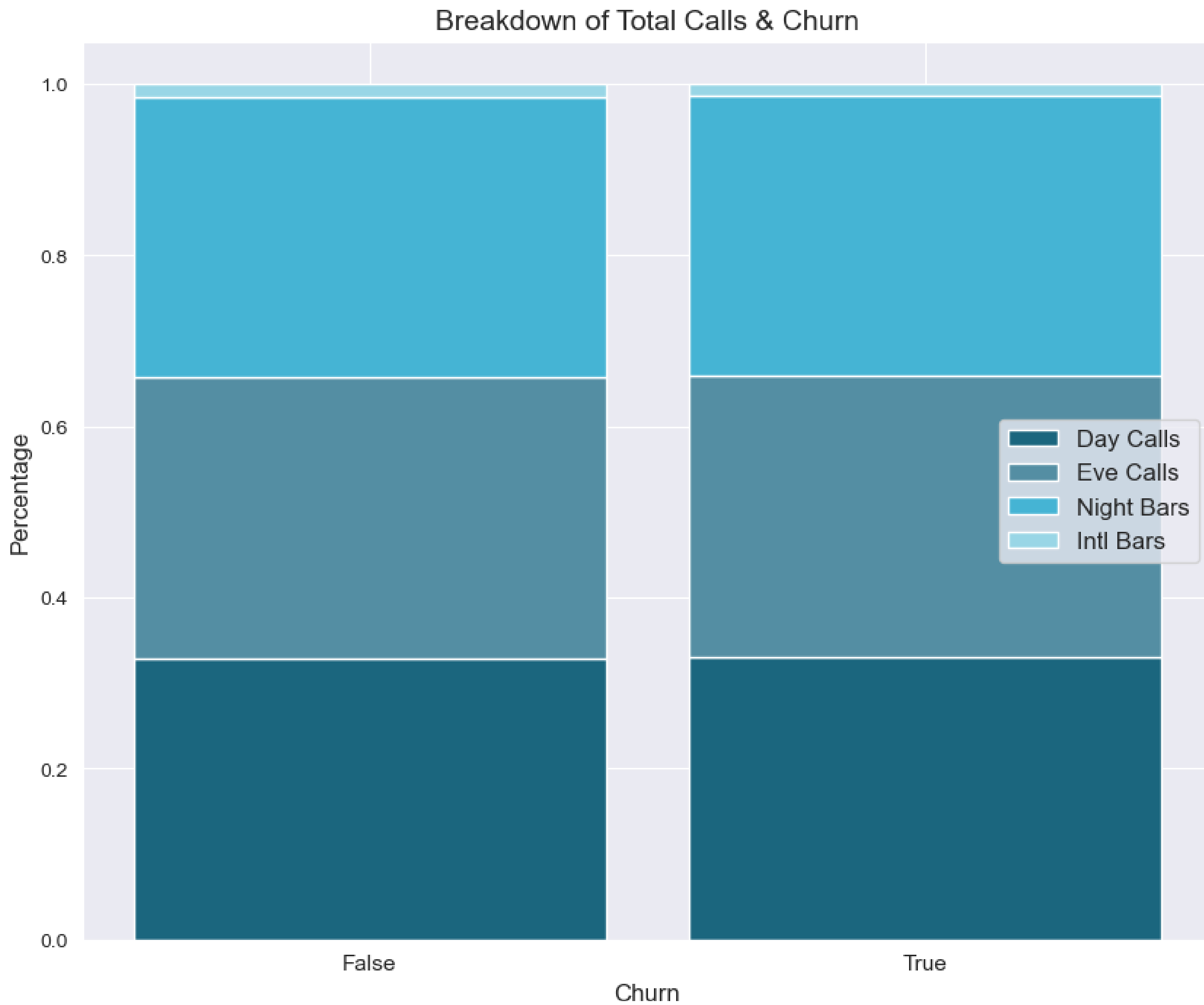
```
   total day minutes  total day calls  ...  total night minutes  \
0           449548.1           257671  ...             513363.8
1            88889.3            43432  ...              88447.3

   total night calls  total night charge  total intl minutes  \
0             256621            23101.62             26106.1
1              43039             3980.15              4604.0

   total intl calls  total intl charge  customer service calls
competiton  \
0             11709            7050.03                    3716
5907
1              1775            1243.31                     957
1186

   avg minutes per intl call  total calls
0                7694.886751       782526
1                1548.873101       131359

[2 rows x 24 columns]

#creates an awesome stacked bar graph to visualize the breakdown of
total calls by 'churn' status, where each bar represents the total
number of calls for different categories
# Create an awesome stacked bar graph of all calls
r = stacked_bar_data['churn']

# Values
totals = stacked_bar_data['total calls']
DayBars = stacked_bar_data['total day calls']
EveBars = stacked_bar_data['total eve calls']
NightBars = stacked_bar_data['total night calls']
IntlBars = stacked_bar_data['total intl calls']

# Plot
plt.figure(figsize=(10,8))
names = ('False', 'True')
barWidth = 0.85

# Create Bars
plt.bar(r, DayBars, color='#1b667e', edgecolor='white',
        width=barWidth, label='Day Calls')

plt.bar(r, EveBars, bottom=DayBars, color='#548ea3',
        edgecolor='white', width=barWidth, label='Eve Calls')

plt.bar(r, NightBars, bottom=[i+j for i,j in zip(DayBars, EveBars)],
        color='#45b4d4', edgecolor='white', width=barWidth,
label='Night Calls')
```

```
plt.bar(r, IntlBars, bottom=[i+j+k for i,j,k in zip(DayBars, EveBars, NightBars)],
        color='#99d6e6', edgecolor='white', width=barWidth, label='Intl Calls')

# Graph details
plt.xticks(r, names, fontsize=11)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Number of Calls', fontsize=12)
plt.title('Breakdown of Total Calls & Churn', fontsize=14)
plt.legend(loc=1, fontsize='large')

# Show graph
plt.show()
```




```
# Create an awesome stacked bar graph of all calls
r = stacked_bar_data['churn']
```

```python
# Turn values to percentage
totals = stacked_bar_data['total calls']
DayBars = stacked_bar_data['total day calls'] / totals
EveBars = stacked_bar_data['total eve calls'] / totals
NightBars = stacked_bar_data['total night calls'] / totals
IntlBars = stacked_bar_data['total intl calls'] / totals

# Plot
plt.figure(figsize=(10,8))
names = ('False', 'True')
barWidth = 0.85

# Create Bars
plt.bar(r, DayBars, color='#1b667e', edgecolor='white',
        width=barWidth, label='Day Calls')

plt.bar(r, EveBars, bottom=DayBars, color='#548ea3',
        edgecolor='white', width=barWidth, label='Eve Calls')


plt.bar(r, NightBars, bottom=[i+j for i,j in zip(DayBars, EveBars)],
        color='#45b4d4', edgecolor='white', width=barWidth,
label='Night Bars')

# Create Bars
plt.bar(r, IntlBars, bottom=[i+j+k for i,j,k in zip(DayBars, EveBars,
NightBars)],
        color='#99d6e6', edgecolor='white', width=barWidth,
label='Intl Bars')

# Graph details
plt.xticks(r, names, fontsize=11)
plt.xlabel('Churn', fontsize=12)
plt.ylabel('Percentage', fontsize=12)
plt.title('Breakdown of Total Calls & Churn', fontsize=14)
plt.legend(loc=7, fontsize='large')

# Show graph
plt.show()
```



Findings & Recommendations

It is evident that there is a remarkable similarity in call usage between customers who churned and those who didn't churn, encompassing day, evening, night, and international calls. Notably, the international call rates remain consistent irrespective of whether a customer has an international plan or not, with both being charged at 27 cents per minute. Interestingly, the percentage of churned customers was higher among those with international plans compared to those without. This suggests that customers with international plans might not perceive the added cost as a valuable benefit.

Recommendations:

Given these insights, I recommend revising the international call rates. Customers with international plans should be offered more competitive rates for international calls compared to those without international plans. This adjustment can enhance the attractiveness of international plans and potentially reduce churn among this group of customers.