Telecom Customer Churn machine learning and analysis

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.offline import plot
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [2]:
```
data = pd.read_csv('C:\\Users\\windows 10 pro\\Documents\\Telecom+Customer+Churn\\teleco
zipcode = pd.read_csv('C:\\Users\\windows 10 pro\\Documents\\Telecom+Customer+Churn\\tel
```

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

Out[9]:

	Customer_id	Gender	Age	Married	Number_of_Dependents	City	Zip_Code	Latitude	Longitude	Nu
0	0002-ORFBO	Female	37	Yes	0	Frazier Park	93225	34.827662	-118.999073	
1	0003-MKNFE	Male	46	No	0	Glendale	91206	34.162515	-118.203869	
2	0004-TLHLJ	Male	50	No	0	Costa Mesa	92627	33.645672	-117.922613	
3	0011-IGKFF	Male	78	Yes	0	Martinez	94553	38.014457	-122.115432	
4	0013-EXCHZ	Female	75	Yes	0	Camarillo	93010	34.227846	-119.079903	

5 rows × 38 columns

In [4]:
```
zipcode.head()
```

Out[4]:

	Zip_Code	Population
0	90001	54492
1	90002	44586
2	90003	58198
3	90004	67852
4	90005	43019

In [5]:
```
churn_df = pd.merge(data, zipcode[['Zip_Code','Population',]], on = 'Zip_Code')
```

In [10]:
```
churn_df.head()
```

Out[10]:

	Customer_id	Gender	Age	Married	Number_of_Dependents	City	Zip_Code	Latitude	Longitude	Nu
0	0002-ORFBO	Female	37	Yes	0	Frazier Park	93225	34.827662	-118.999073	
1	5183-SNMJQ	Male	32	No	0	Frazier Park	93225	34.827662	-118.999073	
2	6847-KJLTS	Female	72	Yes	0	Frazier Park	93225	34.827662	-118.999073	
3	8788-DOXSU	Male	46	No	0	Frazier Park	93225	34.827662	-118.999073	
4	0003-MKNFE	Male	46	No	0	Glendale	91206	34.162515	-118.203869	

5 rows × 39 columns

In [11]:
```
churn_df.describe()
```

Out[11]:

	Age	Number_of_Dependents	Zip_Code	Latitude	Longitude	Number_of_Referrals	Tenu
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	
mean	46.509726	0.468692	93486.070567	36.197455	-119.756684	1.951867	
std	16.750352	0.962802	1856.767505	2.468929	2.154425	3.001199	
min	19.000000	0.000000	90001.000000	32.555828	-124.301372	0.000000	
25%	32.000000	0.000000	92101.000000	33.990646	-121.788090	0.000000	
50%	46.000000	0.000000	93518.000000	36.205465	-119.595293	0.000000	
75%	60.000000	0.000000	95329.000000	38.161321	-117.969795	3.000000	
max	80.000000	9.000000	96150.000000	41.962127	-114.192901	11.000000	

In [12]:
```
churn_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 39 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Customer_id                        7043 non-null   object 
 1   Gender                             7043 non-null   object 
 2   Age                                7043 non-null   int64  
 3   Married                            7043 non-null   object 
 4   Number_of_Dependents               7043 non-null   int64  
 5   City                               7043 non-null   object 
 6   Zip_Code                           7043 non-null   int64  
 7   Latitude                           7043 non-null   float64
 8   Longitude                          7043 non-null   float64
 9   Number_of_Referrals                7043 non-null   int64  
 10  Tenure_in_Months                   7043 non-null   int64  
 11  Offer                              7043 non-null   object 
 12  Phone_Service                      7043 non-null   object 
 13  Avg_Monthly_Long_Distance_Charges  6361 non-null   float64
 14  Multiple_Lines                     6361 non-null   object 
 15  Internet_Service                   7043 non-null   object 
 16  Internet_Type                      5517 non-null   object 
 17  Avg_Monthly_GB_Download            5517 non-null   float64
 18  Online_Security                    5517 non-null   object 
 19  Online_Backup                      5517 non-null   object 
 20  Device_Protection_Plan             5517 non-null   object 
 21  Premium_Tech_Support               5517 non-null   object 
 22  Streaming_TV                       5517 non-null   object 
 23  Streaming_Movies                   5517 non-null   object 
 24  Streaming_Music                    5517 non-null   object 
 25  Unlimited_Data                     5517 non-null   object 
 26  Contract                           7043 non-null   object 
 27  Paperless_Billing                  7043 non-null   object 
 28  Payment_Method                     7043 non-null   object 
 29  Monthly_Charge                     7043 non-null   float64
 30  Total_Charges                      7043 non-null   float64
 31  Total_Refunds                      7043 non-null   float64
 32  Total_Extra_Data_Charges           7043 non-null   int64  
 33  Total_Long_Distance_Charges        7043 non-null   float64
 34  Total_Revenue                      7043 non-null   float64
 35  Customer_Status                    7043 non-null   object 
 36  Churn_Category                     7043 non-null   object 
 37  Churn_Reason                       7043 non-null   object 
 38  Population                         7043 non-null   int64  
dtypes: float64(9), int64(7), object(23)
memory usage: 2.1+ MB
```

In [13]:

```
churn_df.isna().any()
```

Out[13]:
```
Customer_id                          False
Gender                               False
Age                                  False
Married                              False
Number_of_Dependents                 False
City                                 False
Zip_Code                             False
Latitude                             False
Longitude                            False
Number_of_Referrals                  False
Tenure_in_Months                     False
Offer                                False
Phone_Service                        False
Avg_Monthly_Long_Distance_Charges     True
Multiple_Lines                        True
Internet_Service                     False
Internet_Type                         True
Avg_Monthly_GB_Download               True
Online_Security                       True
Online_Backup                         True
Device_Protection_Plan                True
Premium_Tech_Support                  True
Streaming_TV                          True
Streaming_Movies                      True
Streaming_Music                       True
Unlimited_Data                        True
Contract                             False
Paperless_Billing                    False
Payment_Method                       False
Monthly_Charge                       False
Total_Charges                        False
Total_Refunds                        False
Total_Extra_Data_Charges             False
Total_Long_Distance_Charges          False
Total_Revenue                        False
Customer_Status                      False
Churn_Category                       False
Churn_Reason                         False
Population                           False
dtype: bool
```

EDA and visualization

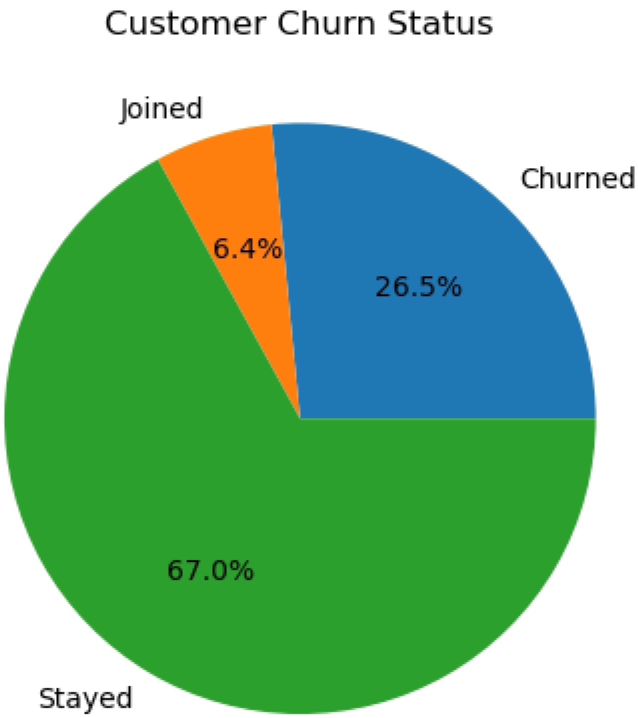
In [14]:
```
cus = churn_df.groupby('Customer_Status')[['Customer_Status']].count()
cus
```

Out[14]:

Customer_Status	Customer_Status
Churned	1869
Joined	454
Stayed	4720

In [89]:
```
cus.plot.pie(autopct='%1.1f%%')
plt.title('Customer Churn Status')
plt.ylabel('')
```

Out[89]:
```
Text(0, 0.5, '')
```



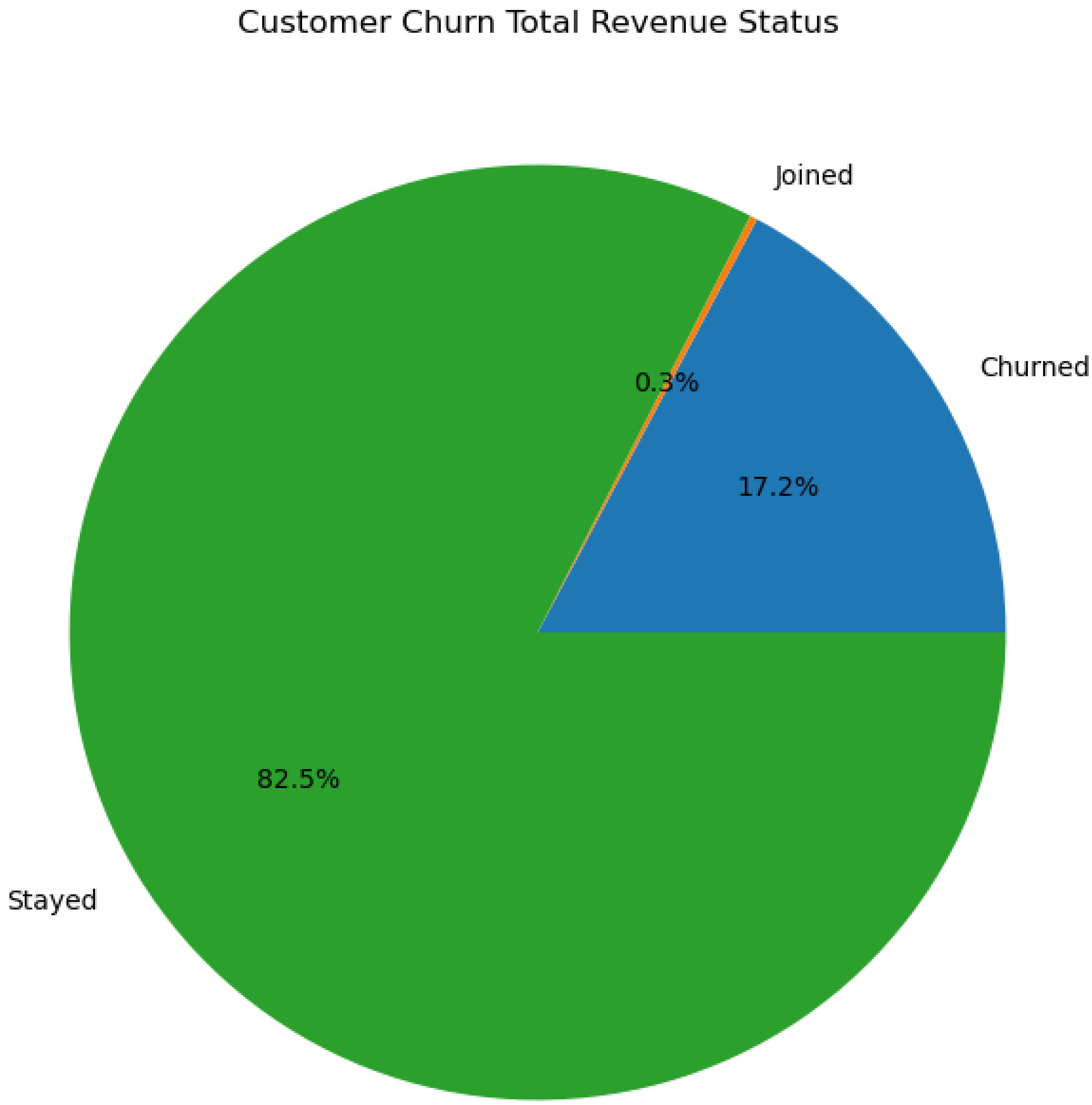
In [113…
```
churn_total_revenue = churn_df.groupby('Customer_Status')[['Total_Revenue']].sum()
churn_total_revenue
```

Out[113]:

	Total_Revenue
Customer_Status	
Churned	3684459.82
Joined	54279.75
Stayed	17632392.12

In [112…
```
churn_total_revenue.plot.pie(autopct='%1.1f%%')
plt.title('Customer Churn Total Revenue Status')
plt.ylabel('')
```

Out[112]:
```
Text(0, 0.5, '')
```



In [120…

```
churn_monthly_charge = churn_df.groupby('Customer_Status')[['Monthly_Charge']].sum()
churn_monthly_charge
```

Out[120]:

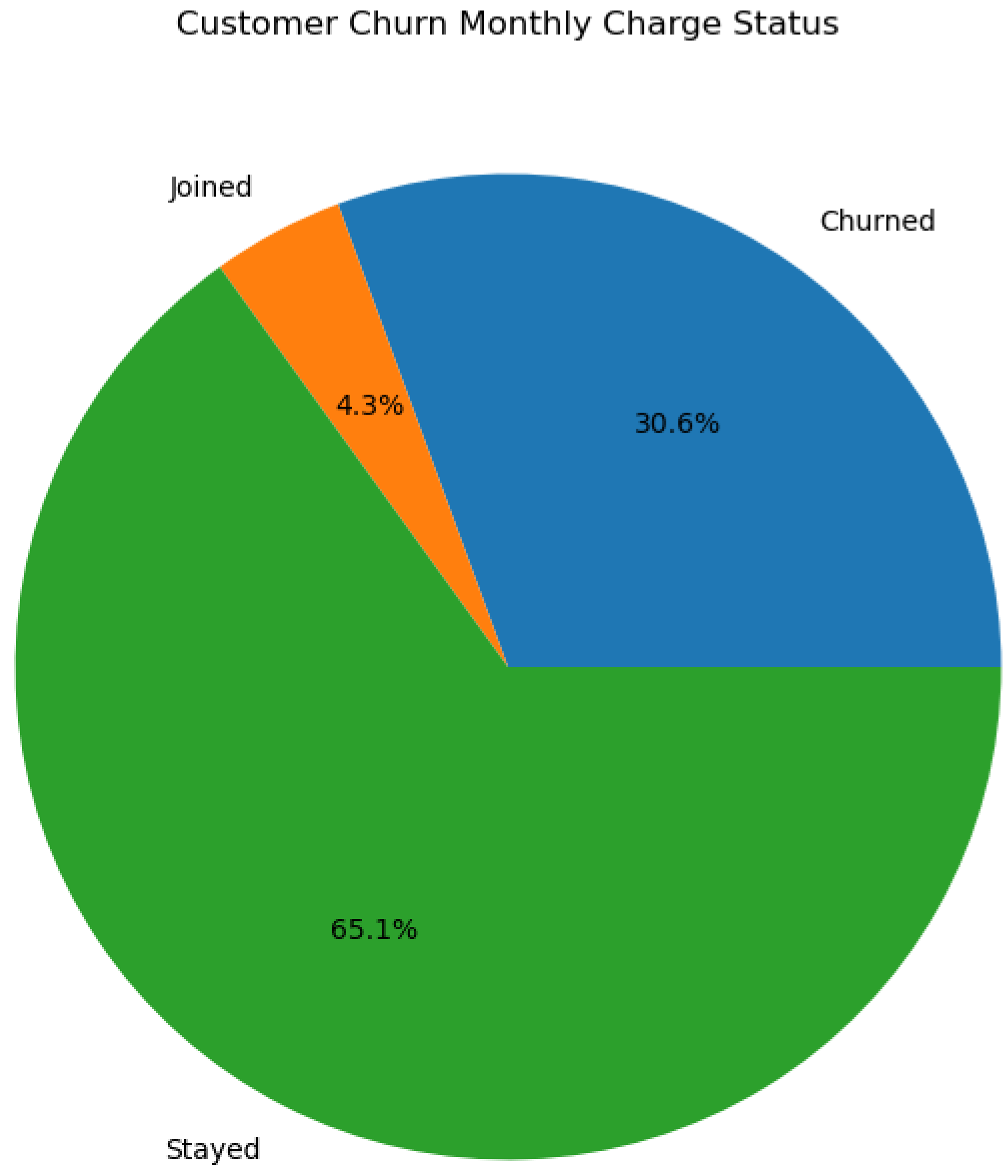
	Monthly_Charge
Customer_Status	
Churned	137086.65
Joined	19420.30
Stayed	291400.60

In [119…

```
churn_monthly_charge.plot.pie(autopct='%1.1f%%')
plt.title('Customer Churn Monthly Charge Status')
plt.ylabel('')
```

Out[119]: Text(0, 0.5, '')

Loading [MathJax]/extensions/Safe.js



In [124…

```
churn_total_long_distance_charges = churn_df.groupby('Customer_Status')[['Total_Long_Dis
churn_total_long_distance_charges
```

Out[124]:

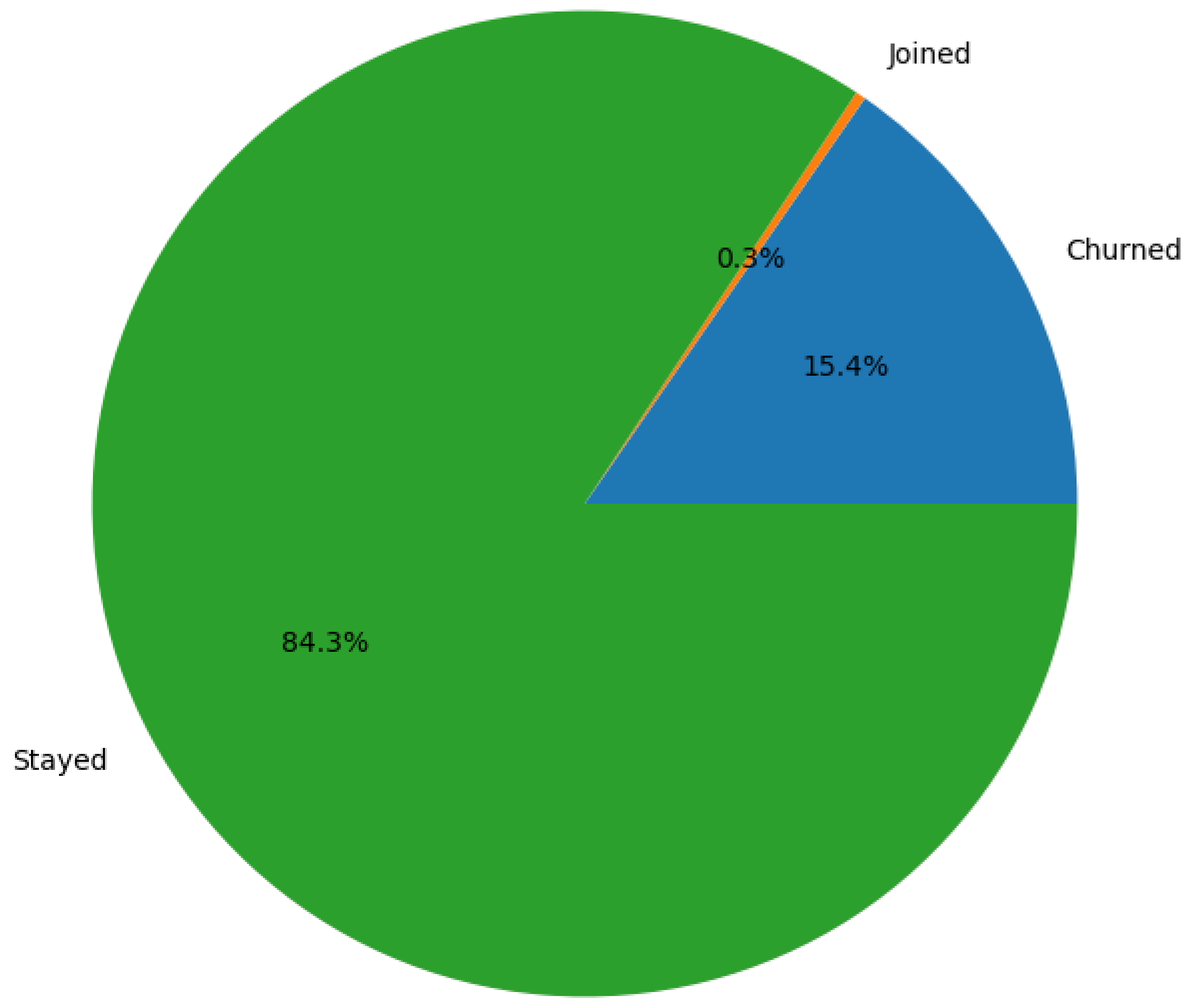
	Total_Long_Distance_Charges
Customer_Status	
Churned	810991.9
Joined	17309.2
Stayed	4447605.0

In [123…

```
churn_total_long_distance_charges.plot.pie(autopct='%1.1f%%')
plt.title('Customer Churn Total Long Distance Charges')
plt.ylabel('')
```

Out[123]: Text(0, 0.5, '')

Customer Churn Total Long Distance Charges



In [92]:
```
churn_cat = churn_df.groupby('Churn_Category')[['Churn_Category']].count()
churn_cat
```

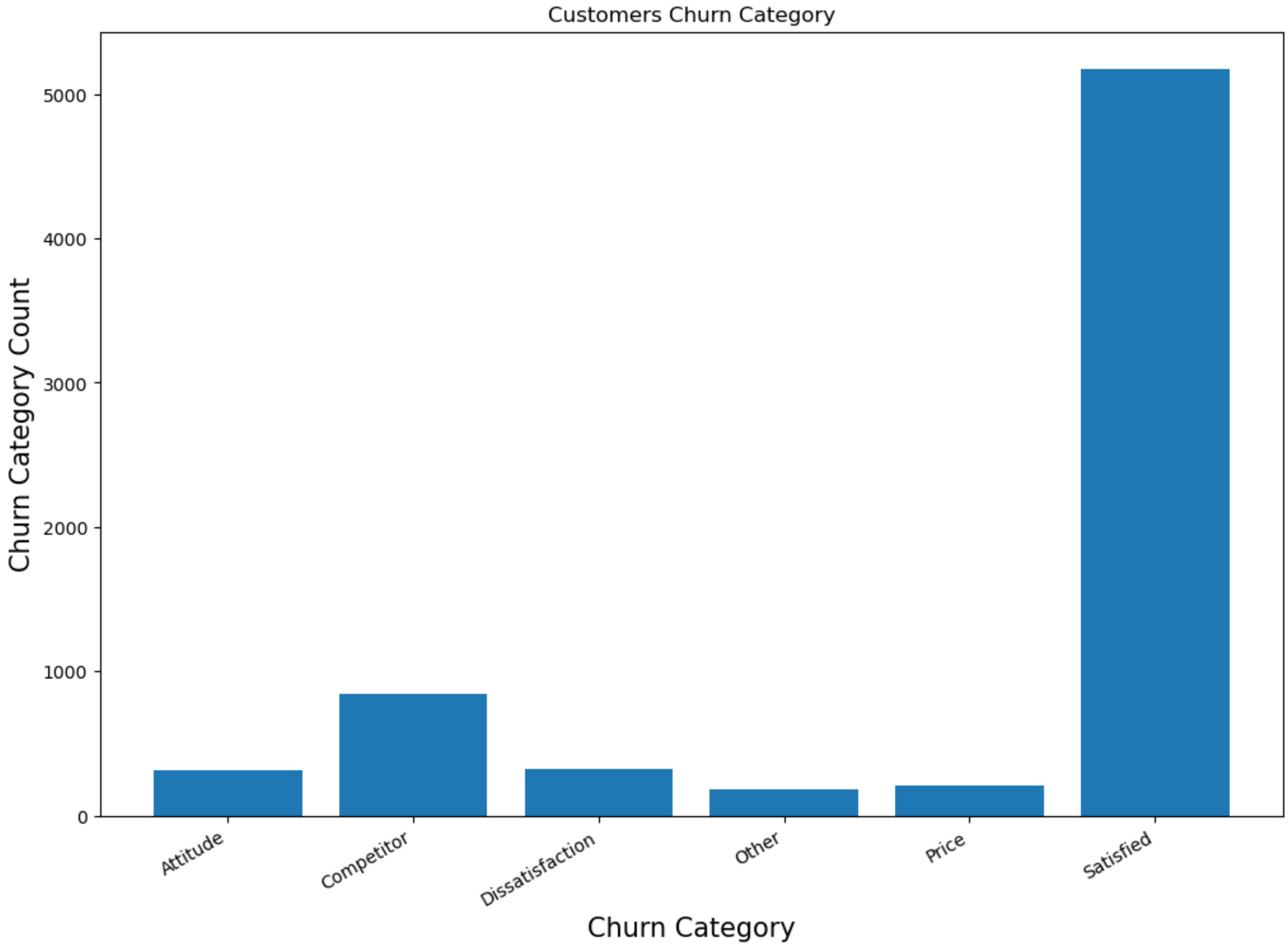
Out[92]:

Churn_Category	Churn_Category
Attitude	314
Competitor	841
Dissatisfaction	321
Other	182
Price	211
Satisfied	5174

In [97]:
```
plt.rcParams['figure.figsize']=(12,8)
plt.title('Customers Churn Category')
plt.xlabel('Churn Category', fontsize = 15)
plt.ylabel('Churn Category Count', fontsize = 15)
plt.xticks(rotation = 30, ha = 'right')
_cat.index, churn_cat.Churn_Category)
```

Loading [MathJax]/extensions/Safe.js

Out[97]: <BarContainer object of 6 artists>



In [102…

```
churn_rea = churn_df.groupby('Churn_Reason')[['Churn_Reason']].count()
churn_rea
```

Loading [MathJax]/extensions/Safe.js

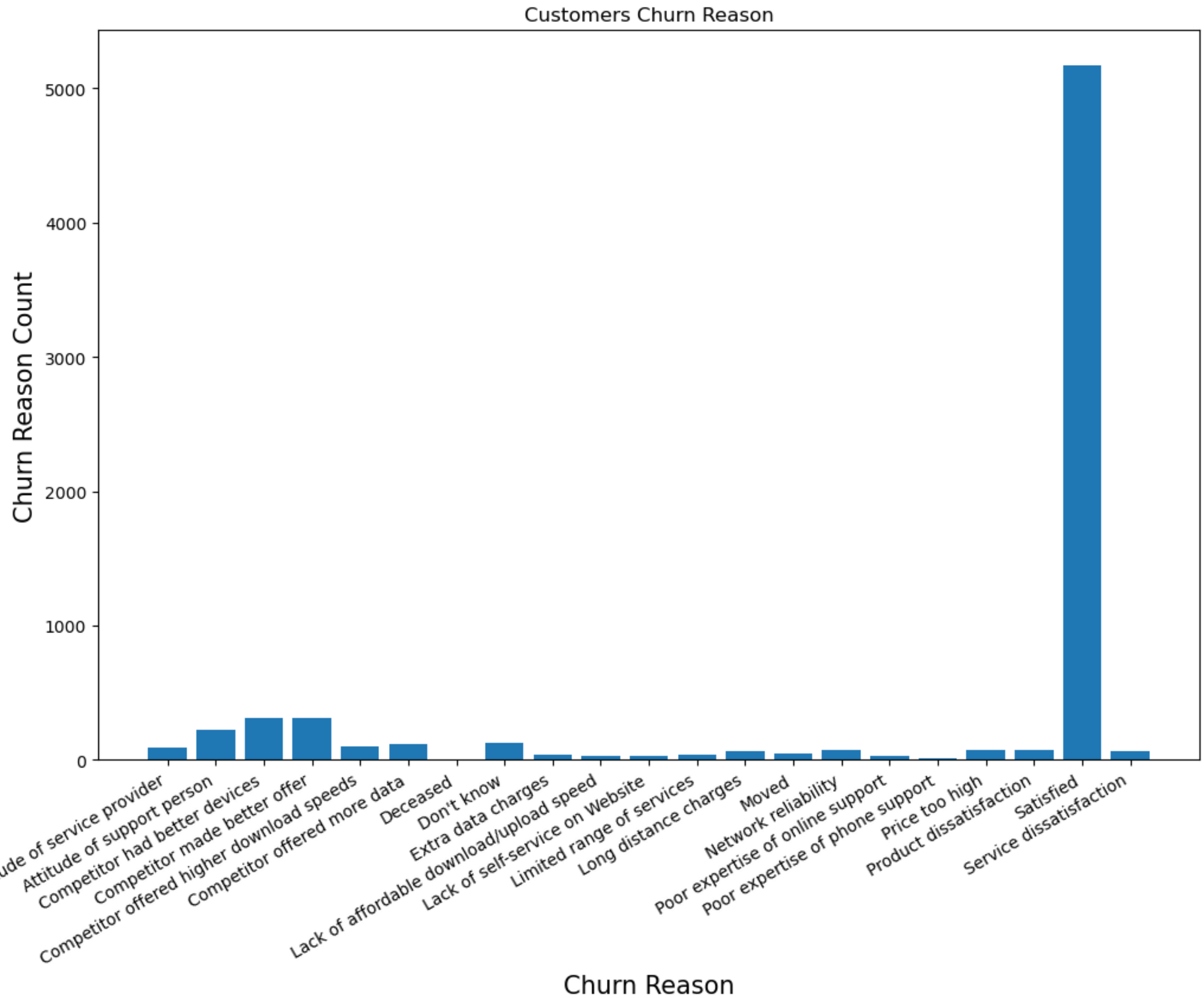
Out[102]:

Churn_Reason	Churn_Reason
Attitude of service provider	94
Attitude of support person	220
Competitor had better devices	313
Competitor made better offer	311
Competitor offered higher download speeds	100
Competitor offered more data	117
Deceased	6
Don't know	130
Extra data charges	39
Lack of affordable download/upload speed	30
Lack of self-service on Website	29
Limited range of services	37
Long distance charges	64
Moved	46
Network reliability	72
Poor expertise of online support	31
Poor expertise of phone support	12
Price too high	78
Product dissatisfaction	77
Satisfied	5174
Service dissatisfaction	63

In [103…

```
plt.rcParams['figure.figsize']=(12,8)
plt.title('Customers Churn Reason')
plt.xlabel('Churn Reason', fontsize = 15)
plt.ylabel('Churn Reason Count', fontsize = 15)
plt.xticks(rotation = 30, ha = 'right')
plt.bar(churn_rea.index, churn_rea.Churn_Reason)
```

Out[103]: <BarContainer object of 21 artists>



In [132…

```
# Total closing rate by Region
total_City = churn_df.groupby('City')[['Total_Revenue']].sum()
total_City
```

Out[132]:

City	Total_Revenue
Acampo	18107.96
Acton	12156.36
Adelanto	18235.49
Adin	5539.38
Agoura Hills	10641.88
...	...
Yreka	17467.67
Yuba City	17867.82
Yucaipa	29765.80
Yucca Valley	12374.69
Zenia	12733.38

1106 rows × 1 columns

Loading [MathJax]/extensions/Safe.js

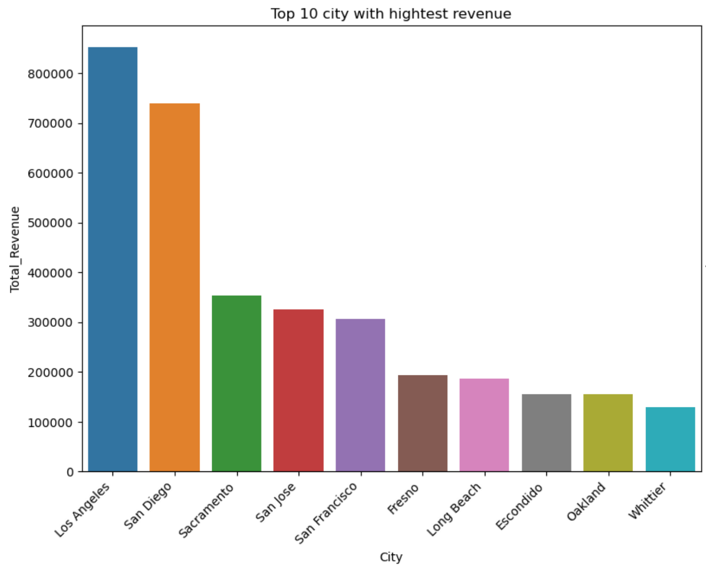
In [134…
```
top_10 = total_City.sort_values('Total_Revenue', ascending=False).head(10)
print(top_10)
bottom_10 = total_City.sort_values('Total_Revenue', ascending=False).tail(10)
print(bottom_10)
```

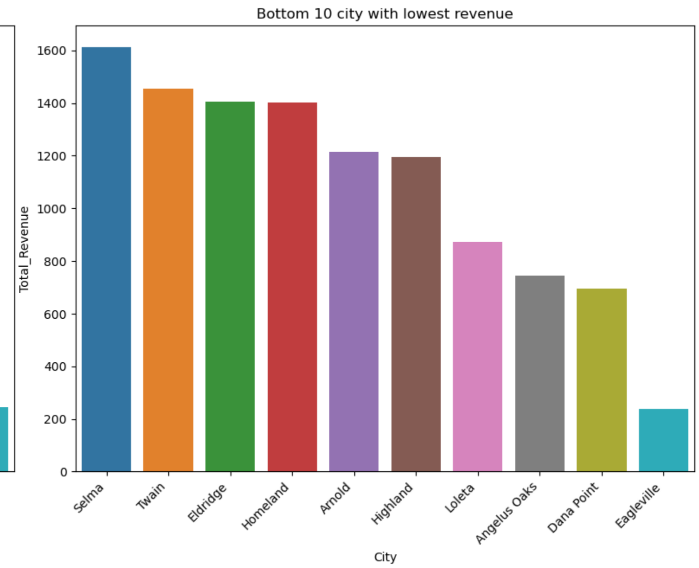
```
               Total_Revenue
City
Los Angeles        852725.23
San Diego          738416.01
Sacramento         353371.84
San Jose           326478.36
San Francisco      306995.99
Fresno             194430.25
Long Beach         185937.12
Escondido          155899.80
Oakland            154564.36
Whittier           128858.77
              Total_Revenue
City
Selma               1613.53
Twain               1453.58
Eldridge            1405.04
Homeland            1400.54
Arnold              1215.61
Highland            1193.64
Loleta               873.05
Angelus Oaks         744.72
Dana Point           694.86
Eagleville           238.57
```

In [135…
```
# Total Revenue by City Visualization
fig, axes = plt.subplots (1,2, figsize = (16,6))
plt.tight_layout(pad=2)
xlabels = top_10.index
axes[0].set_title('Top 10 city with hightest revenue')
axes[0].set_xticklabels(xlabels, rotation = 45, ha = 'right')
sns.barplot(x=top_10.index, y=top_10.Total_Revenue, ax=axes[0])
axes[0].set_xlabel('City')
axes[0].set_ylabel('Total_Revenue')

xlabels = bottom_10.index
axes[1].set_title('Bottom 10 city with lowest revenue')
axes[1].set_xticklabels(xlabels, rotation = 45, ha = 'right')
sns.barplot(x=bottom_10.index, y=bottom_10.Total_Revenue, ax=axes[1])
axes[1].set_xlabel('City')
axes[1].set_ylabel('Total_Revenue')
```

```
C:\Users\windows 10 pro\AppData\Local\Temp\ipykernel_5232\906439376.py:6: UserWarning:

FixedFormatter should only be used together with FixedLocator

C:\Users\windows 10 pro\AppData\Local\Temp\ipykernel_5232\906439376.py:13: UserWarning:

FixedFormatter should only be used together with FixedLocator
```

Out[135]:
```
Text(838.9608585858584, 0.5, 'Total_Revenue')
```





In [15]:
```
churn_df1 = churn_df.iloc[:, [13,17,29,30,31,32,33,35]]
churn_df1.head(5)
```

Out[15]:

	Avg_Monthly_Long_Distance_Charges	Avg_Monthly_GB_Download	Monthly_Charge	Total_Charges	Total_Refu
0	42.39	16.0	65.60	593.3	
1	45.69	11.0	95.10	865.1	4
2	47.34	28.0	100.40	5749.8	
3	9.70	6.0	61.35	3645.5	
4	10.69	10.0	-4.00	542.4	3

In [16]:
```
churn_df1['Avg_Monthly_Long_Distance_Charges'].fillna(churn_df1['Avg_Monthly_Long_Distan
churn_df1['Avg_Monthly_GB_Download'].fillna(churn_df1['Avg_Monthly_GB_Download'].mean(),
```

```
C:\Users\windows 10 pro\AppData\Local\Temp\ipykernel_1124\74051609.py:1: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\windows 10 pro\AppData\Local\Temp\ipykernel_1124\74051609.py:2: SettingWithCopy
Warning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

In [17]:
```
churn_df1.isna().any()
```

Loading [MathJax]/extensions/Safe.js

```
Out[17]: Avg_Monthly_Long_Distance_Charges    False
         Avg_Monthly_GB_Download              False
         Monthly_Charge                       False
         Total_Charges                        False
         Total_Refunds                        False
         Total_Extra_Data_Charges             False
         Total_Long_Distance_Charges          False
         Customer_Status                      False
         dtype: bool
```

```
In [18]: churn_df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 8 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Avg_Monthly_Long_Distance_Charges  7043 non-null   float64
 1   Avg_Monthly_GB_Download            7043 non-null   float64
 2   Monthly_Charge                     7043 non-null   float64
 3   Total_Charges                      7043 non-null   float64
 4   Total_Refunds                      7043 non-null   float64
 5   Total_Extra_Data_Charges           7043 non-null   int64  
 6   Total_Long_Distance_Charges        7043 non-null   float64
 7   Customer_Status                    7043 non-null   object 
dtypes: float64(6), int64(1), object(1)
memory usage: 495.2+ KB
```

```
In [19]: X = churn_df1.drop('Customer_Status', axis=1)
         y = churn_df1['Customer_Status']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
In [ ]:
```

```
In [20]: # Feature scaling
         sc = MinMaxScaler(feature_range=(0,1))
         sc1 = sc.fit_transform(X_train)
```

```
In [21]: clf1 = RandomForestClassifier(n_jobs=3, random_state=0)
         clf1.fit(X_train, y_train)
```

```
Out[21]: ▼ RandomForestClassifier
         RandomForestClassifier(n_jobs=3, random_state=0)
```

```
In [22]: # Make predictions
         y_pred = clf1.predict(X_test)
```

```
In [23]: accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.7885024840312278
```

```
In [35]: con_matrix = confusion_matrix(y_test, y_pred)
         print(f"Accuracy: {con_matrix}")
```

```
Accuracy: [[112  30 234]
 [ 46  40   8]
 [ 25   0 914]]
```

```
In [24]: report = classification_report(y_test, y_pred)
         print(f"Accuracy: {report}")
```

```
Accuracy:                precision    recall  f1-score   support

     Churned       0.65      0.48      0.55       376
      Joined       0.58      0.49      0.53        94
      Stayed       0.84      0.94      0.89       939

    accuracy                           0.79      1409
   macro avg       0.69      0.64      0.66      1409
weighted avg       0.77      0.79      0.77      1409
```

Application of 20% hypothesis discount on the charge

In [25]:
```
X_test_20_percent_discount = X_test
X_test_20_percent_discount.head()
```

Out[25]:

	Avg_Monthly_Long_Distance_Charges	Avg_Monthly_GB_Download	Monthly_Charge	Total_Charges	Total_F
2200	23.13	24.0	109.90	7332.40	
4627	1.15	23.0	70.10	70.10	
3225	13.40	17.0	44.00	44.00	
2828	33.69	15.0	80.00	1029.35	
3768	1.38	16.0	69.95	320.40	

In [26]:
```
X_test_20_percent_discount.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1409 entries, 2200 to 3065
Data columns (total 7 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Avg_Monthly_Long_Distance_Charges  1409 non-null   float64
 1   Avg_Monthly_GB_Download            1409 non-null   float64
 2   Monthly_Charge                     1409 non-null   float64
 3   Total_Charges                      1409 non-null   float64
 4   Total_Refunds                      1409 non-null   float64
 5   Total_Extra_Data_Charges           1409 non-null   int64  
 6   Total_Long_Distance_Charges        1409 non-null   float64
dtypes: float64(6), int64(1)
memory usage: 88.1 KB
```

The 20% discount hypothesis is applied using X_test columu Monthly_Charge and Total_Long_Distance_Charges

In [27]:
```
X_test_20_percent_discount['Monthly_Charge'] = X_test_20_percent_discount['Monthly_Charg
X_test_20_percent_discount['Total_Long_Distance_Charges'] = X_test_20_percent_discount['
```

In [28]:
```
X_test_20_percent_discount.head()
```

Out[28]:

	Avg_Monthly_Long_Distance_Charges	Avg_Monthly_GB_Download	Monthly_Charge	Total_Charges	Total_F
2200	23.13	24.0	87.92	7332.40	
4627	1.15	23.0	56.08	70.10	
3225	13.40	17.0	35.20	44.00	
2828	33.69	15.0	64.00	1029.35	
3768	1.38	16.0	55.96	320.40	

In [39]:
```
# Make predictions
y_pred_20 = clf1.predict(X_test_20_percent_discount)
```

In [40]:
```
accuracy = accuracy_score(y_test, y_pred_20)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.7565649396735273
```

In [41]:
```
con_matrix1 = confusion_matrix(y_test, y_pred_20)
print(f"Accuracy: {con_matrix}")
```

```
Accuracy: [[112  30 234]
 [ 46  40   8]
 [ 25   0 914]]
```

In [42]:
```
report = classification_report(y_test, y_pred_20)
print(f"Accuracy: {report}")
```

```
Accuracy:               precision    recall  f1-score   support

     Churned       0.61      0.30      0.40       376
      Joined       0.57      0.43      0.49        94
      Stayed       0.79      0.97      0.87       939

    accuracy                           0.76      1409
   macro avg       0.66      0.57      0.59      1409
weighted avg       0.73      0.76      0.72      1409
```

From the above before and after 20% hypothesis discount, the classification report show that the support in the model

is same. This proof that the 20% do not have any impact on churn.

Findings and Recommendations

churn is indeed high in the internet division

* 25.5% across 1869 out of 7043 customers

Predictive model is able to predict churn but the main driver is not customer charge sensitivity

* Attitude of support person, Competitor had better devices, and Competitor made better offer are the largest drivers

Discount strategy of 20% discount is very less effective and not necessary.

* Application of 20% discount can be used to motivate new subcriber to join

In [ ]:

Loading [MathJax]/extensions/Safe.js