Date - 17/10/2023

Team ID - 721

Project Title - Customer Churn Prediction

1.Import Libraries required to create the Customer Churn Model

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load Churn Prediction Dataset

In [2]:
```
data = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

3.Exploring Dataset

1. Displaying the top 5 rows

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

Out[4]:

ineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	Paperl
No	...	No	No	No	No	Month-to-month	
Yes	...	Yes	No	No	No	One year	
Yes	...	No	No	No	No	Month-to-month	
Yes	...	Yes	Yes	No	No	One year	
No	...	No	No	No	No	Month-to-month	

2. Displaying the bottom 5 rows

In [34]:
```
data.tail()
```

Out[34]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multiple
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No s
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	
7042	3186-AJIEK	Male	0	No	No	66	Yes	

5 rows × 21 columns

3. Displaying the colomns

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

Out[6]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

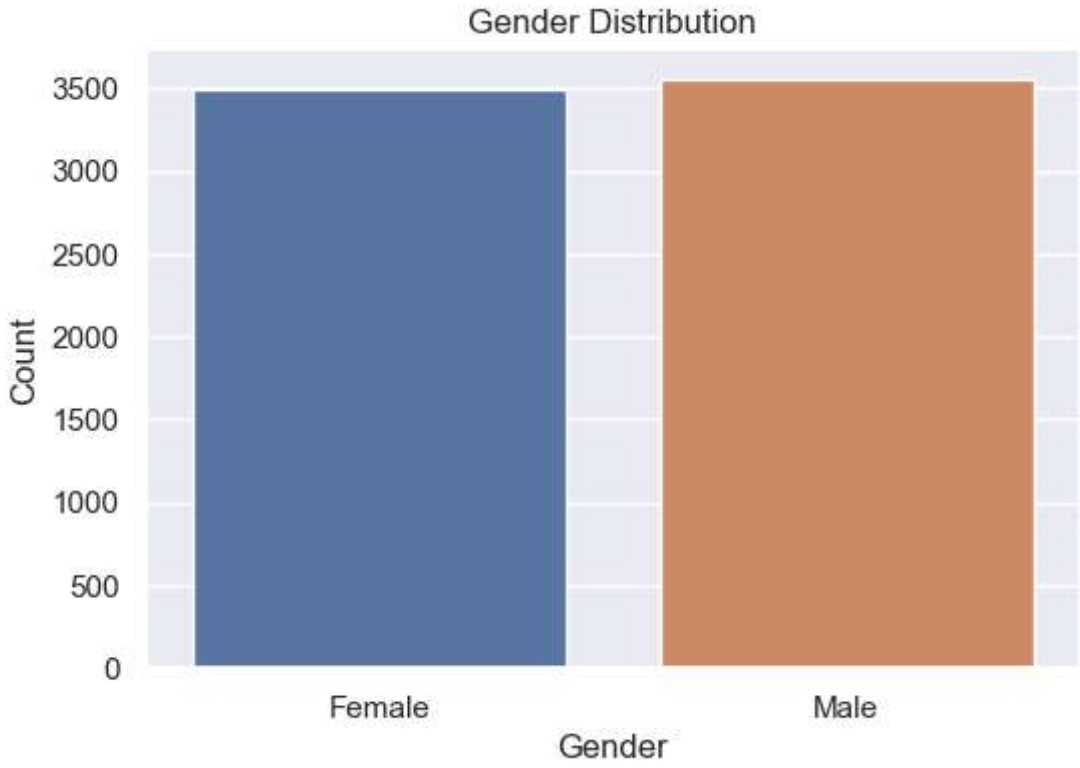
4. Displaying the shape

In [36]:
```
data.shape
```

Out[36]:
```
(7043, 21)
```

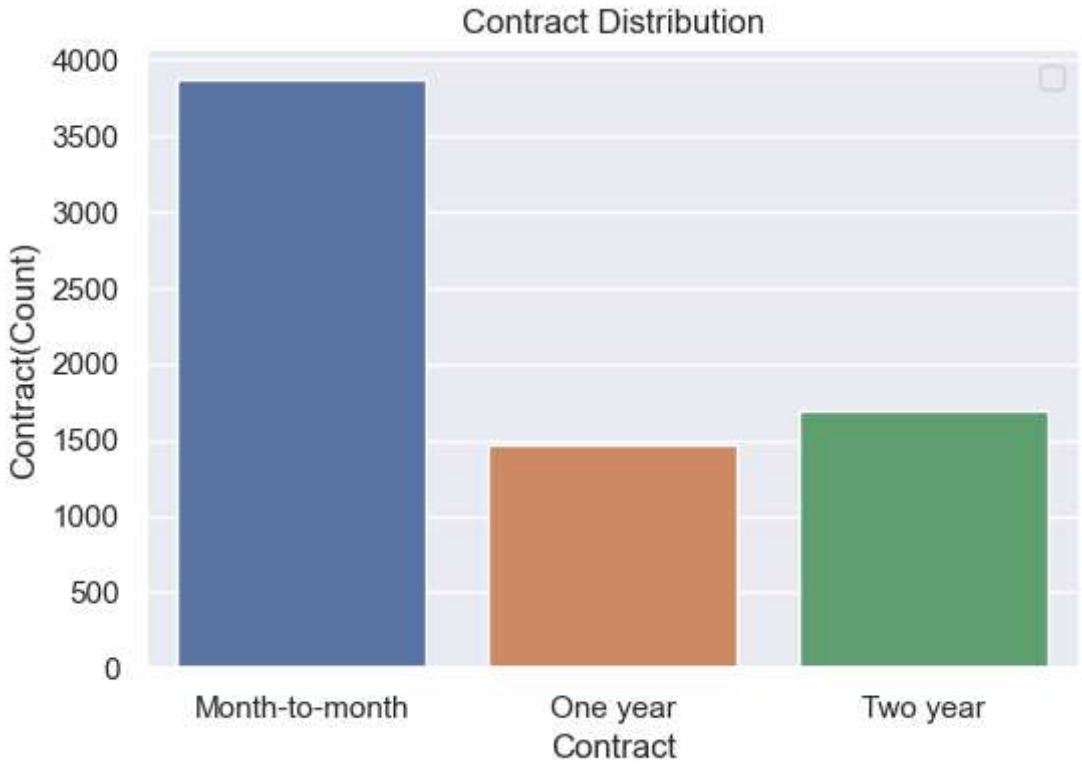
4.Data Visualization

In [44]:
```
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='gender')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



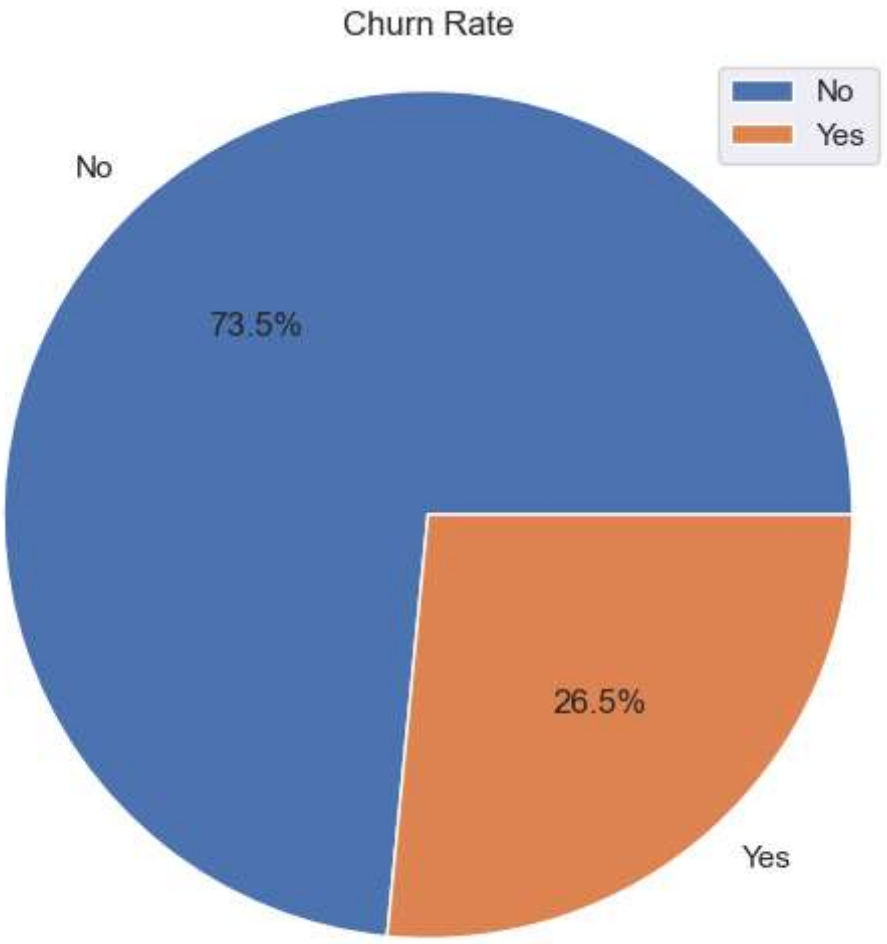
In [58]:
```
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='Contract')
plt.title('Contract Distribution')
plt.xlabel('Contract')
plt.ylabel('Contract(Count)')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



In [48]:
```
# Count the number of customers for each churn category
churn_counts = data['Churn'].value_counts()
```

In [54]:
```
# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(churn_counts, labels=churn_counts.index, autopct='%1.1f%%')
plt.title('Churn Rate')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circu
plt.legend()
plt.show()
```



5.Preprocess Dataset

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [9]:
```
missing_values = data.isnull().sum()
```

In [10]:
```
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 customerID          0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges         0
Churn                0
dtype: int64
```

In [11]:
```
data = data.dropna()
```

In [12]:
```
data.describe()
```

Out[12]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [13]:
```
df = data.drop('customerID',axis=1)
```

In [15]:
```
df
```

Out[15]:

Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	Onlin
Yes	No	1	No	No phone service	DSL	No	
No	No	34	Yes	No	DSL	Yes	
No	No	2	Yes	No	DSL	Yes	
No	No	45	No	No phone service	DSL	Yes	
No	No	2	Yes	No	Fiber optic	No	
...	...	...	...	...	...	...	
Yes	Yes	24	Yes	Yes	DSL	Yes	
Yes	Yes	72	Yes	Yes	Fiber optic	No	
Yes	Yes	11	No	No phone service	DSL	Yes	
Yes	No	4	Yes	Yes	Fiber optic	No	
No	No	66	Yes	No	Fiber optic	Yes	

In [16]:
```
#count of string value into the column.
count=0
for i in df.TotalCharges:
    if i==' ':
        count+=1
print('count of empty string:- ',count)
#we will replace this empty string to nan values
df['TotalCharges'] = df['TotalCharges'].replace(" ",np.nan)
# typecasting of the TotalCharges column
df['TotalCharges'] = df['TotalCharges'].astype(float)
```

```
count of empty string:-  11
```

6. Checking Null Values in Customer Churn Data

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

In [18]:
```
# fill null values with mean
df['TotalCharges'] = df['TotalCharges'].fillna(df['TotalCharges'].mean())
```

In [19]:
```
#numerical variables

num = list(df.select_dtypes(include=['int64','float64']).keys())

#categorical variables

cat = list(df.select_dtypes(include='O').keys())

print(cat)

print(num)
```

```
['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'Inte
rnetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechS
upport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod', 'Churn']
['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']
```

In [20]:
```
# value_counts of the categorical columns
for i in cat:
    print(df[i].value_counts())
# as we see that there is extra categories which we have to convert it into
df.MultipleLines = df.MultipleLines.replace('No phone service','No')
df.OnlineSecurity = df.OnlineSecurity.replace('No internet service','No')
df.OnlineBackup = df.OnlineBackup.replace('No internet service','No')
df.DeviceProtection = df.DeviceProtection.replace('No internet service','No
df.TechSupport = df.TechSupport.replace('No internet service','No')
df.StreamingTV = df.StreamingTV.replace('No internet service','No')
df.StreamingMovies = df.StreamingMovies.replace('No internet service','No')
```

```
gender
Male      3555
Female    3488
Name: count, dtype: int64
Partner
No     3641
Yes    3402
Name: count, dtype: int64
Dependents
No     4933
Yes    2110
Name: count, dtype: int64
PhoneService
Yes    6361
No      682
Name: count, dtype: int64
MultipleLines
No                  3390
Yes                 2971
No phone service     682
Name: count, dtype: int64
InternetService
Fiber optic    3096
DSL            2421
No             1526
Name: count, dtype: int64
OnlineSecurity
No                     3498
Yes                    2019
No internet service    1526
Name: count, dtype: int64
OnlineBackup
No                     3088
Yes                    2429
No internet service    1526
Name: count, dtype: int64
DeviceProtection
No                     3095
Yes                    2422
No internet service    1526
Name: count, dtype: int64
TechSupport
No                     3473
Yes                    2044
No internet service    1526
Name: count, dtype: int64
StreamingTV
No                     2810
Yes                    2707
No internet service    1526
Name: count, dtype: int64
StreamingMovies
No                     2785
Yes                    2732
No internet service    1526
Name: count, dtype: int64
Contract
Month-to-month    3875
Two year          1695
One year          1473
Name: count, dtype: int64
```

```
PaperlessBilling
Yes    4171
No     2872
Name: count, dtype: int64
PaymentMethod
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: count, dtype: int64
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

7.Handling categorical Variables in Customer Churn Data

In [21]:
```
# we have to handel this all categorical variables
# there are mainly Yes/No features in most of the columns
# we will convert Yes = 1 and No = 0
for i in cat:
    df[i] = df[i].replace('Yes',1)
    df[i] = df[i].replace('No',0)
```

In [26]:
```
df
```

Out[26]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Interr
0	Female	0	1	0	1	0	0	
1	Male	0	0	0	34	1	0	
2	Male	0	0	0	2	1	0	
3	Male	0	0	0	45	0	0	
4	Female	0	0	0	2	1	0	
...	...	...	...	...	...	...	...	...
7038	Male	0	1	1	24	1	1	
7039	Female	0	1	1	72	1	1	
7040	Female	0	1	1	11	0	0	
7041	Male	1	1	0	4	1	1	
7042	Male	0	0	0	66	1	0	

7043 rows × 20 columns

In [31]:
```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
#from sklearn.preprocessing import OrdinalEncoder

# Convert all values in 'InternetService' to strings
df['InternetService'] = df['InternetService'].astype(str)

# Create and fit the OrdinalEncoder
oe = OrdinalEncoder()
df['InternetService'] = oe.fit_transform(df[['InternetService']])

df['Contract'] = oe.fit_transform(df[['Contract']])
df['PaymentMethod'] = oe.fit_transform(df[['PaymentMethod']])
df
```

Out[31]:

OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperI
1	0	0	0	0	0.0	
0	1	0	0	0	1.0	
1	0	0	0	0	0.0	
0	1	1	0	0	1.0	
0	0	0	0	0	0.0	
...	...	...	...	...	...	
0	1	1	1	1	1.0	
1	1	0	1	1	1.0	
0	0	0	0	0	0.0	
0	0	0	0	0	0.0	
0	1	1	1	1	2.0	

In [32]:
```
scale_cols = ['tenure','MonthlyCharges','TotalCharges']
# now we scling all the data
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
df[scale_cols] = scale.fit_transform(df[scale_cols])
df
```

Out[32]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inte
0	Female	0	1	0	0.013889	0	0	
1	Male	0	0	0	0.472222	1	0	
2	Male	0	0	0	0.027778	1	0	
3	Male	0	0	0	0.625000	0	0	
4	Female	0	0	0	0.027778	1	0	
...	...	...	...	...	...	...	...	
7038	Male	0	1	1	0.333333	1	1	
7039	Female	0	1	1	1.000000	1	1	
7040	Female	0	1	1	0.152778	0	0	
7041	Male	1	1	0	0.055556	1	1	
7042	Male	0	0	0	0.916667	1	0	

7043 rows × 20 columns