Codesoft Internship Aug 2023

Task-CUSTOMER CHURN PREDICTION

Author-Vivek Kumar

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_csv("churn.csv")
df
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Devic
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	

7043 rows × 21 columns

In [3]:
```
df.shape
```

Out[3]:
```
(7043, 21)
```

In [4]:
```
df.columns.values
```

Out[4]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [5]:
```
df.isna().sum()
```

Out[5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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

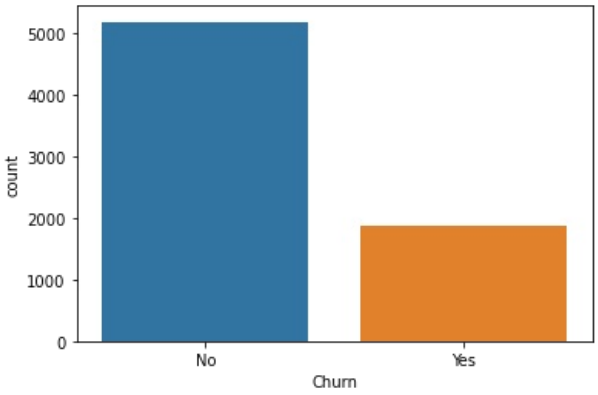
In [7]:
```
df['Churn'].value_counts()
```

Out[7]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

In [8]:
```
sns.countplot(df['Churn'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:
```
<AxesSubplot:xlabel='Churn', ylabel='count'>
```



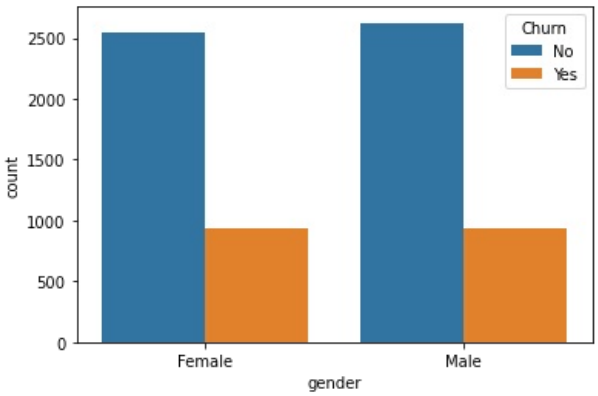
In [9]:
```
numRetained = df[df.Churn == 'No'].shape[0]
numChurned = df[df.Churn == 'Yes'].shape[0]

# print the percentage of customers that stayed
print(numRetained/(numRetained + numChurned) * 100,'% of customers stayed in the company')
# print the percentage of customers that left
print(numChurned/(numRetained + numChurned) * 100, '% of customers left with the company')
```

```
73.4630129206304 % of customers stayed in the company
26.536987079369588 % of customers left with the company
```

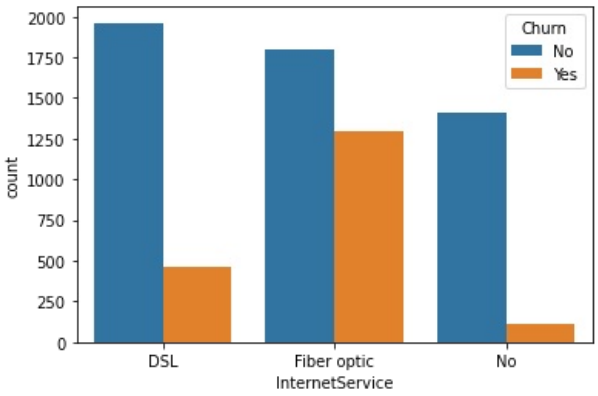
In [10]:
```
sns.countplot(x ='gender', hue='Churn', data=df)
```

Out[10]:
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



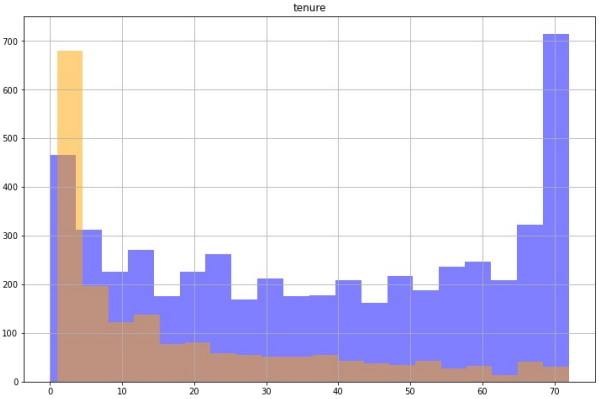
In [11]:
```
sns.countplot(x='InternetService', hue='Churn', data=df)
```

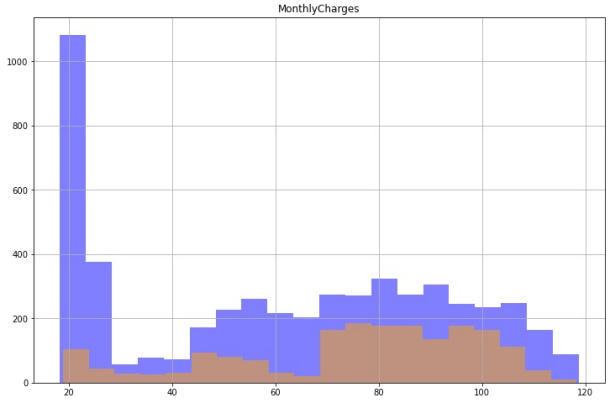
Out[11]:
```
<AxesSubplot:xlabel='InternetService', ylabel='count'>
```



In [12]:
```
numericFeatures = ['tenure', 'MonthlyCharges']
fig, ax = plt.subplots(1,2, figsize=(28, 8))
df[df.Churn == "No"][numericFeatures].hist(bins=20, color='blue', alpha=0.5, ax=ax)
df[df.Churn == "Yes"][numericFeatures].hist(bins=20, color='orange', alpha=0.5, ax=ax)
```

Out[12]:
```
array([<AxesSubplot:title={'center':'tenure'}>,
       <AxesSubplot:title={'center':'MonthlyCharges'}>], dtype=object)
```





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

In [15]:
```
#Convert all the non-numeric columns to numeric
for column in cleanDF.columns:
  if cleanDF[column].dtype == np.number:
    continue
  cleanDF[column] = LabelEncoder().fit_transform(cleanDF[column])
```

```
C:\Users\admin_a\AppData\Local\Temp\ipykernel_14584\1383960061.py:3: DeprecationWarning: Converting `np.inexact
` or `np.floating` to a dtype is deprecated. The current result is `float64` which is not strictly correct.
  if cleanDF[column].dtype == np.number:
```

In [16]:
```
cleanDF.dtypes
```

Out[16]:
```
gender                int32
SeniorCitizen         int64
Partner               int32
Dependents            int32
tenure                int64
PhoneService          int32
MultipleLines         int32
InternetService       int32
OnlineSecurity        int32
OnlineBackup          int32
DeviceProtection      int32
TechSupport           int32
StreamingTV           int32
StreamingMovies       int32
Contract              int32
PaperlessBilling      int32
PaymentMethod         int32
MonthlyCharges      float64
TotalCharges          int32
Churn                 int32
dtype: object
```

In [17]:
```
x = cleanDF.drop('Churn', axis=1)
y = cleanDF['Churn']
x = StandardScaler().fit_transform(x)
```

In [18]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.2, random_state=42)
```

In [19]:
```
model = LogisticRegression()
# Train the model
model.fit(xtrain, ytrain)
```

Out[19]:
```
LogisticRegression()
```

In [20]:
```
predictions = model.predict(xtest)

# print the predictions
print(predictions)
```

```
[1 0 0 ... 0 0 0]
```

In [21]:
```
print(classification_report(ytest, predictions))
```

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88      1036
           1       0.69      0.56      0.62       373

    accuracy                           0.82      1409
   macro avg       0.77      0.74      0.75      1409
weighted avg       0.81      0.82      0.81      1409
```

In [ ]: