Predict Customer Conversion (Churn) with Machine Learning

Importing necessary libraries

```
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
```

Reading and exploring the dataset

```
df = pd.read_csv("customer-churn-dataset.csv")
df
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService

```
df.shape
```

```
(7043, 21)
```

```
df.columns.values
```

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

... ...

To check for missing values or (NA)

```
df.isna().sum()
```

```
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
```

Dataset Statistics

```
df.describe()
```

	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

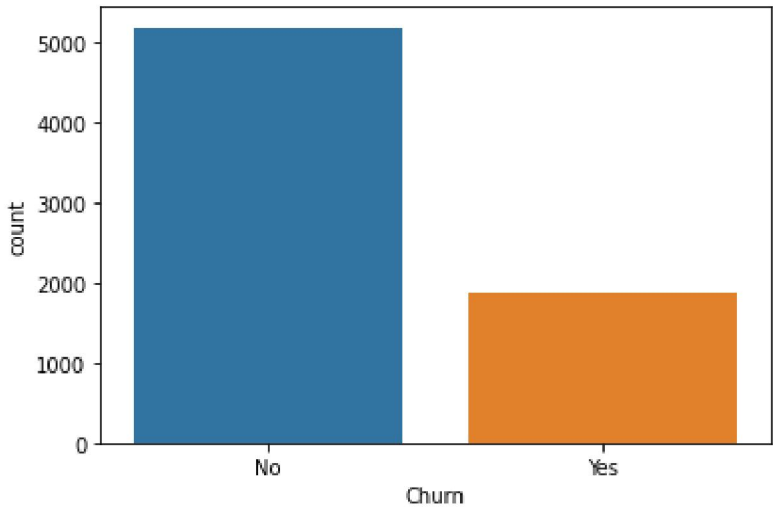
```
df['Churn'].value_counts()
```

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

Visualizing the conversion

```
sns.countplot(df['Churn'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fdae08d7a50>
```



Percentage-wise results

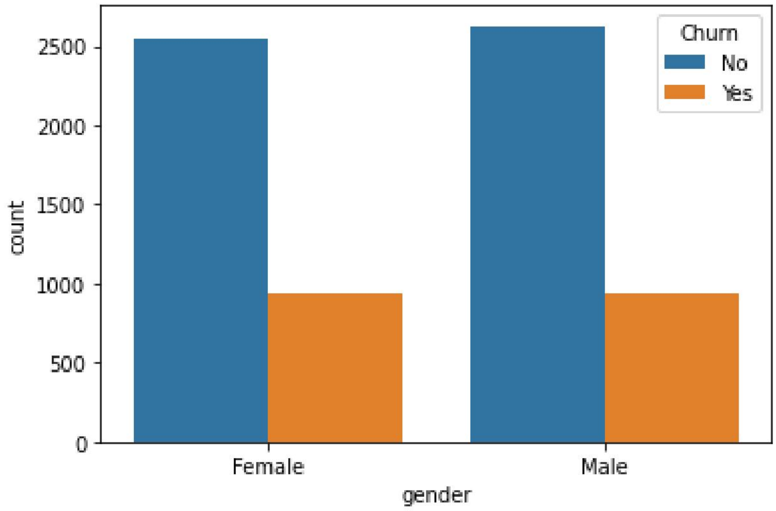
```
numRetained = df[df.Churn == 'No'].shape[0]
numChurned = df[df.Churn == 'Yes'].shape[0]

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

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

Gender-wise visualization of customer conversion

```
sns.countplot(x ='gender', hue='Churn', data=df)
```

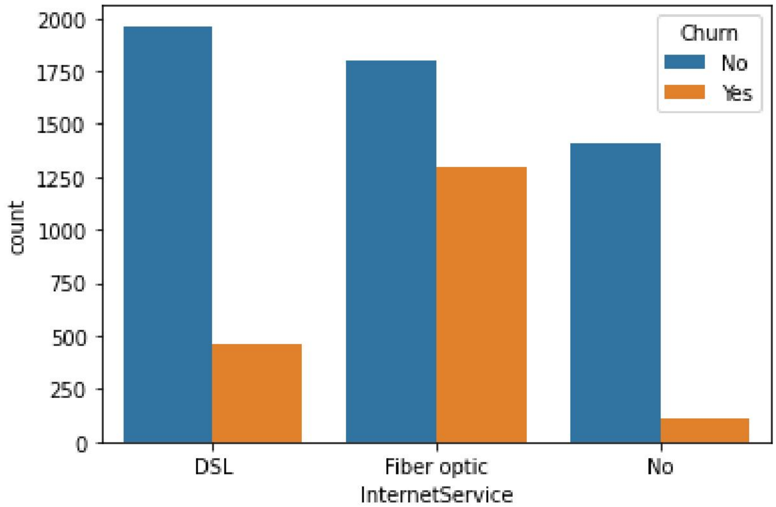
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdadf621ad0>
```



Visualization of customer conversion for the internet service

```
sns.countplot(x='InternetService', hue='Churn', data=df)
```

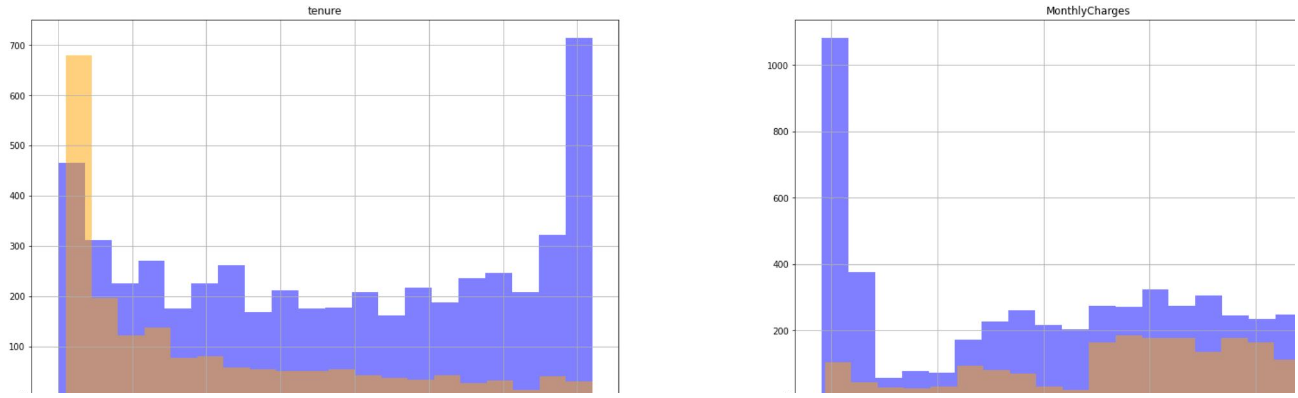
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdadf15ba50>
```



Visualization of Numerical data

```
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)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fdadf19c490>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdadf098a50>],
      dtype=object)
```



Dropping unnecessary columns from the dataset

```
cleanDF = df.drop('customerID', axis=1)
```

```
# 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])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarning:
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
cleanDF.dtypes
```

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

Scaling of data

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

Split the data into 80% for training and 20% for testing

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.2, random_state=42)
```

Creating and Training the Logistic Regression Model

```
model = LogisticRegression()
# Train the model
model.fit(xtrain, ytrain)
```

```
LogisticRegression()
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
LogisticRegression()
```

Creating predictions on the Test data

```
predictions = model.predict(xtest)

# print the predictions
print(predictions)
```

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

Final scores - precision, recall and f1-score

```
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
```