# Milestone 3: Preliminary Analysis

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# importing warning to surpress the future warnings
import warnings
warnings.simplefilter(action='ignore')
```

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

In [3]:
```
df.head(10)
```

Out[3]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Interne |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | Fib |
| **5** | 9305-CDSKC | Female | 0 | No | No | 8 | Yes | Yes | Fib |
| **6** | 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes | Yes | Fib |
| **7** | 6713-OKOMC | Female | 0 | No | No | 10 | No | No phone service | |
| **8** | 7892-POOKP | Female | 0 | Yes | No | 28 | Yes | Yes | Fib |
| **9** | 6388-TABGU | Male | 0 | No | Yes | 62 | Yes | No | |

10 rows × 21 columns

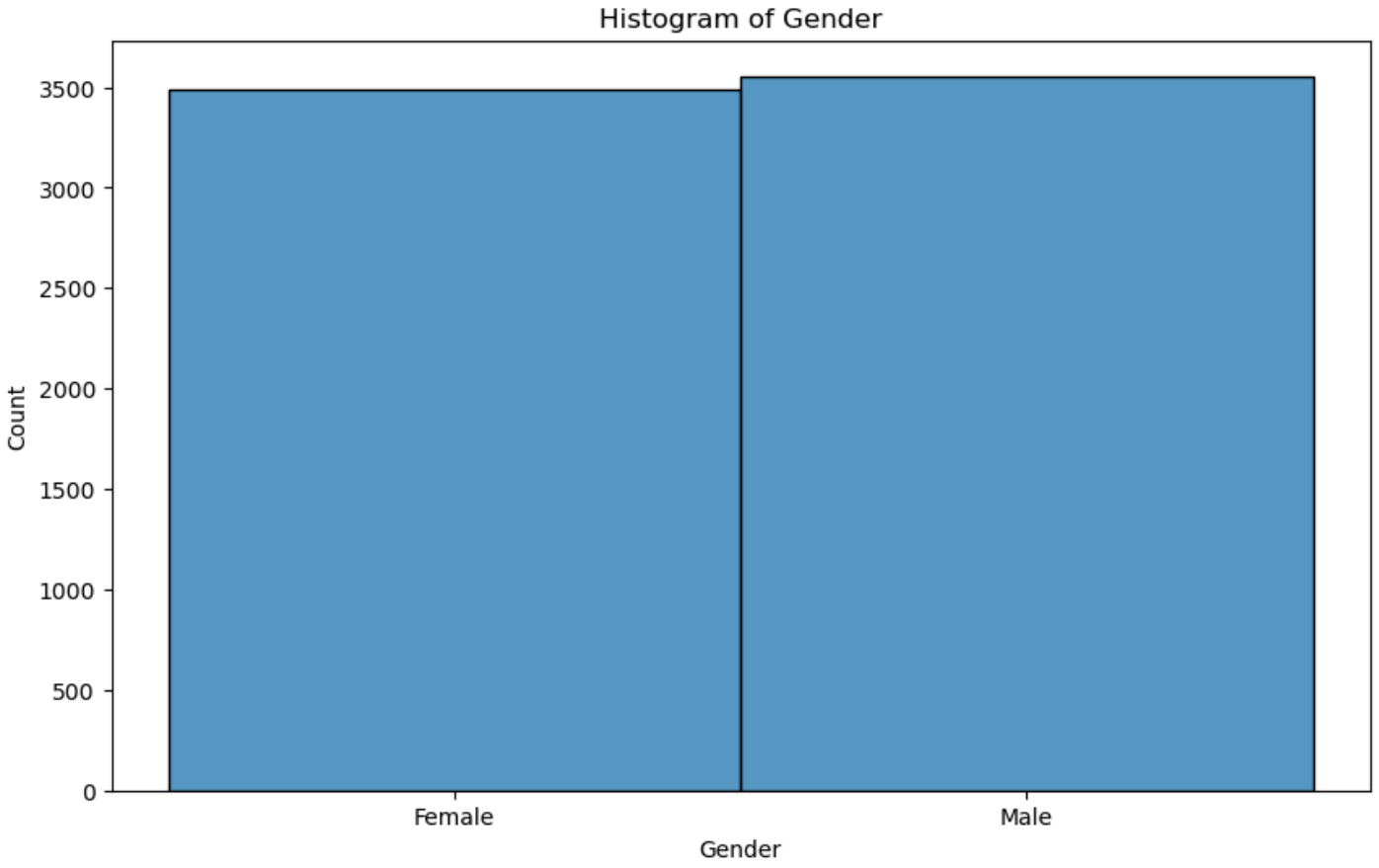
In [4]:
```
df.shape
```

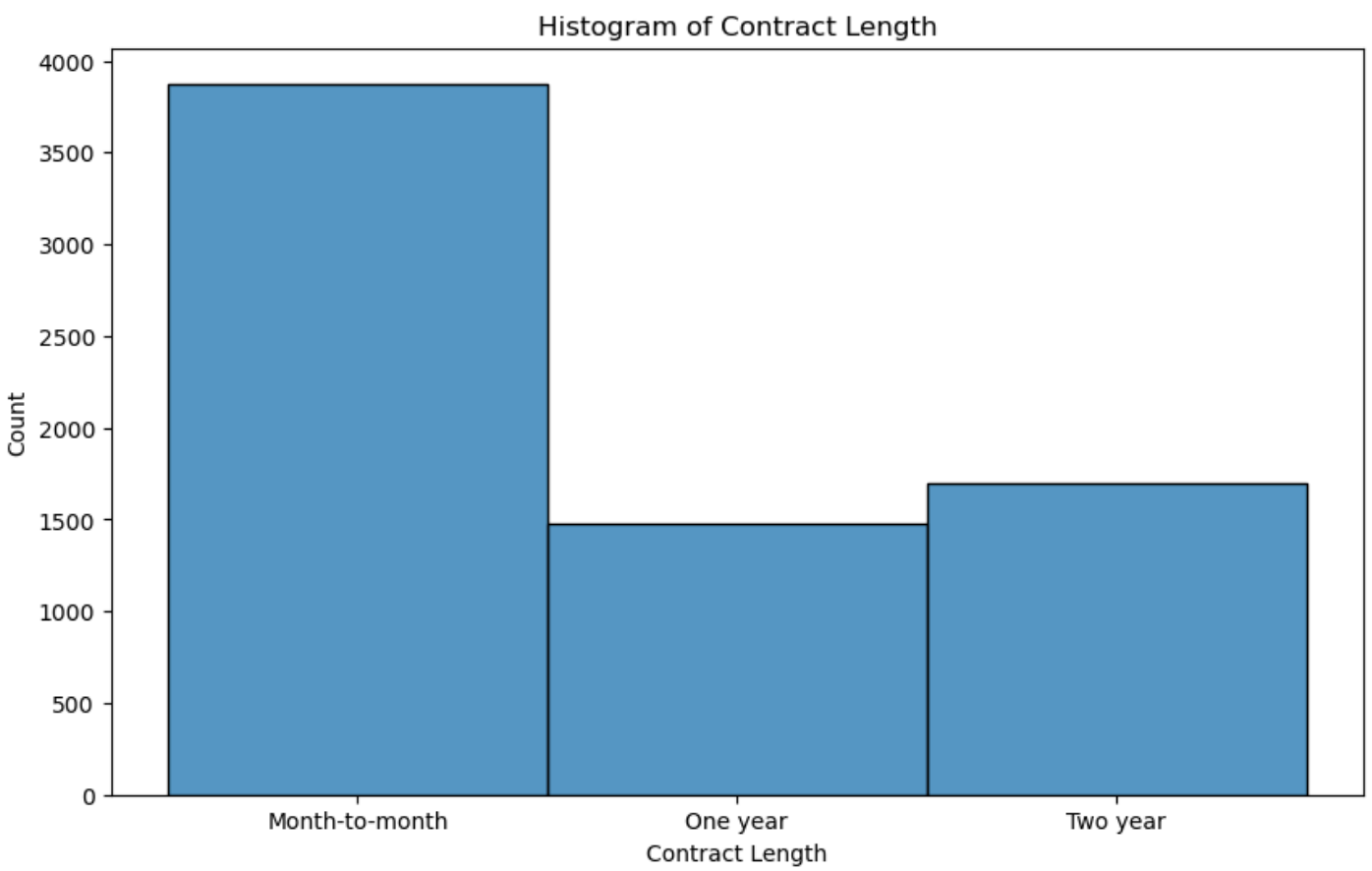
Out[4]: (7043, 21)

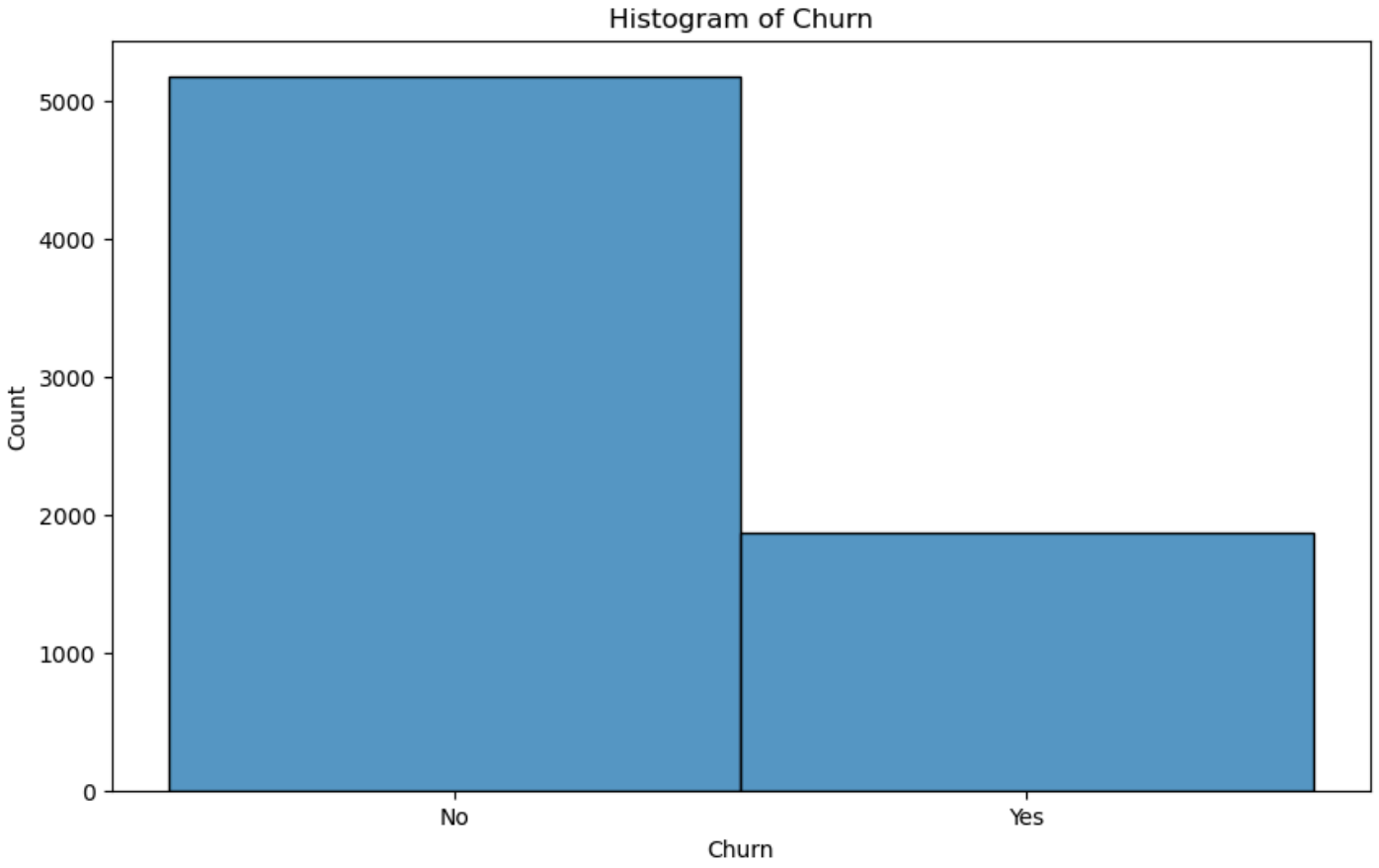
In [5]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Histogram of Gender')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Contract')
plt.xlabel('Contract Length')
plt.ylabel('Count')
plt.title('Histogram of Contract Length')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Churn')
plt.xlabel('Churn')
plt.ylabel('Count')
plt.title('Histogram of Churn')
plt.show()
```






# Milestone 4: Data Preperation, Models, and Analysis

At Milestone 4 I'll be building and completeing the model as well as starting to provide some analysis and final thoughts.

# Data Review

In [6]:
```
# Review the data frame again
df.head()
```

Out[6]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Interne |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | Fib |

5 rows × 21 columns

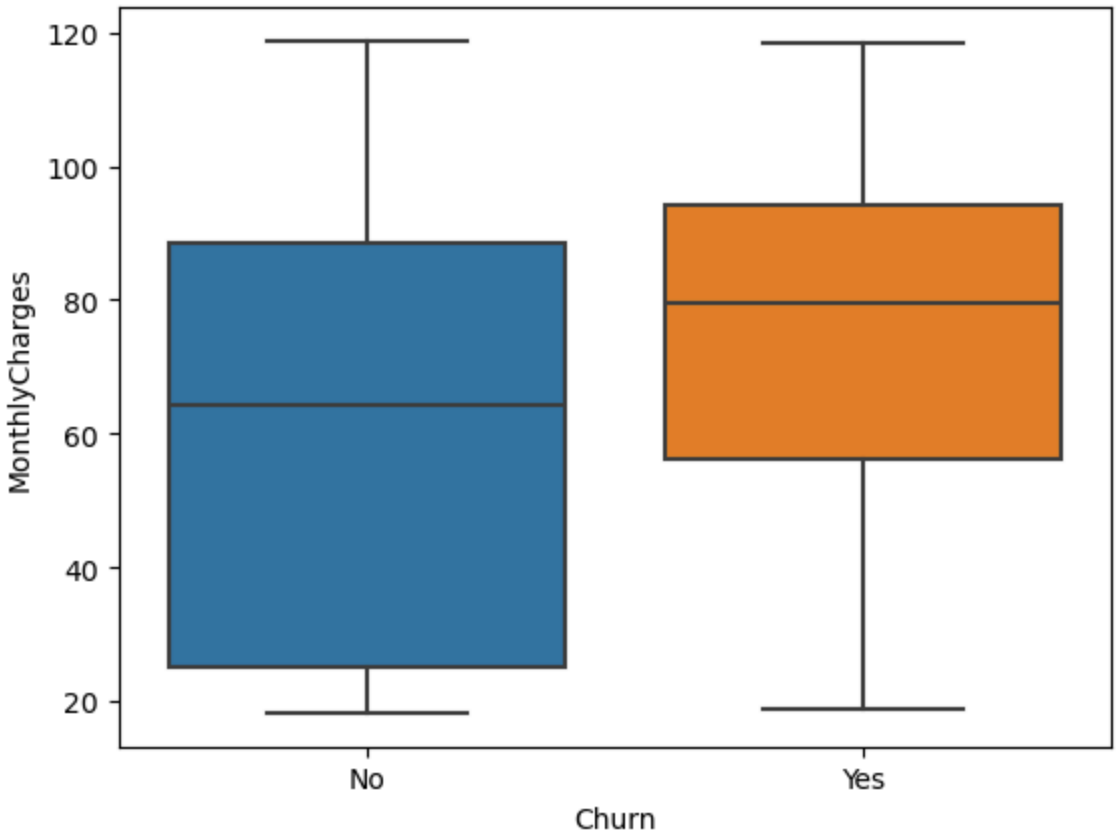
In [7]:
```
df.shape
```

Out[7]:
```
(7043, 21)
```

In [8]:
```
df.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]:
```
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.show()
```




In [10]:
```
# Counting how many customer have churned
df["Churn"].value_counts()
```

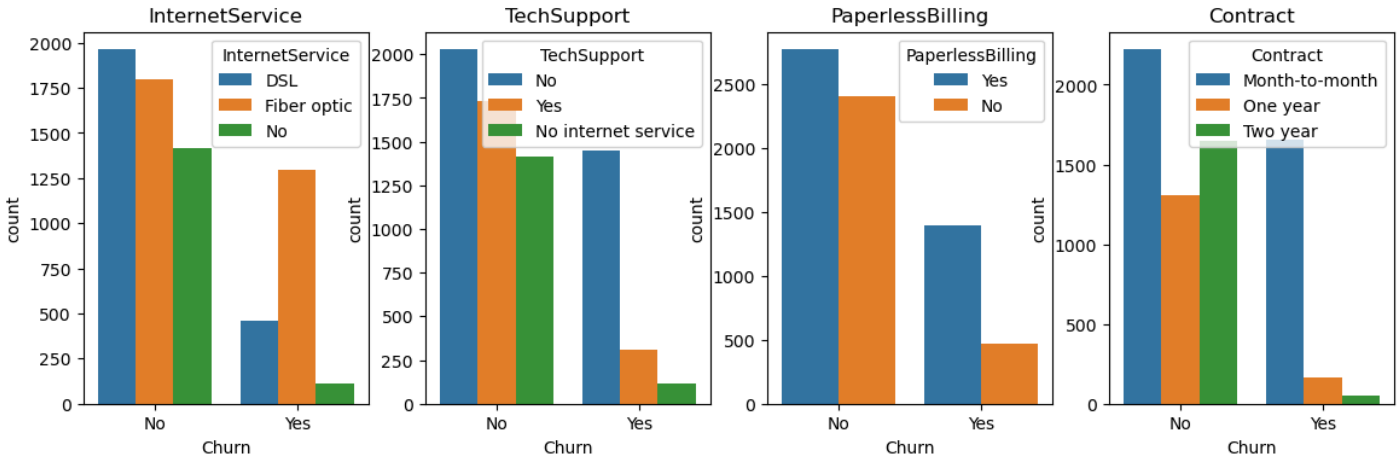
Out[10]:
```
Churn
No      5174
Yes     1869
Name: count, dtype: int64
```

We can see only a small number of customers have acually 'churned'. This indicates an unbalanced classification problem. We'll use oversampling to correct during the train and test data set portion of the project.

In [11]:
```
# plotting churn against other categorical variables
cols = ['InternetService',"TechSupport","PaperlessBilling","Contract"]

plt.figure(figsize=(14,4))

for i, col in enumerate(cols):
    ax = plt.subplot(1, len(cols), i+1)
    sns.countplot(x ="Churn", hue = str(col), data = df)
    ax.set_title(f"{col}")
```




# Data Preperation

In the data preperation section I will be preparing the data for modeling.

In [12]:
```
# converting the totalcharges column from object to float64, this will match the monthlycharges c
df['TotalCharges'] = df['TotalCharges'].apply(lambda x: pd.to_numeric(x, errors='coerce')).dropn
```

Since this data set was created specifically to practice customer churn analysis, there are not any NaN values in the data set. There is minimal data cleanup to complete. However, there will be a few pre-processing steps to prep the data for the model.

In [13]:
```
# reviewing categorical variables from data frame
cat_features = df.drop(['customerID','TotalCharges','MonthlyCharges','SeniorCitizen','tenure'],a
cat_features.head()
```

Out[13]:

| | gender | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBack |
|---|---|---|---|---|---|---|---|---|
| **0** | Female | Yes | No | No | No phone service | DSL | No | |
| **1** | Male | No | No | Yes | No | DSL | Yes | |
| **2** | Male | No | No | Yes | No | DSL | Yes | |
| **3** | Male | No | No | No | No phone service | DSL | Yes | |
| **4** | Female | No | No | Yes | No | Fiber optic | No | |

In [14]:
```
# importing additional library for pre-processing
from sklearn import preprocessing
```

In [15]:
```
# creating a lebel encoder to tranforma and fit the categorical features
label_encoder = preprocessing.LabelEncoder()
df_cat = cat_features.apply(label_encoder.fit_transform)
df_cat.head()
```

Out[15]:

| | gender | Partner | Dependents | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBack |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | |
| **1** | 1 | 0 | 0 | 1 | 0 | 0 | 2 | |
| **2** | 1 | 0 | 0 | 1 | 0 | 0 | 2 | |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 | 2 | |
| **4** | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |

In [16]:
```
# combining the original data frame with the tranformed categorical data frame
num_features = df[['TotalCharges','MonthlyCharges','SeniorCitizen','tenure']]
finaldf = pd.merge(num_features, df_cat, left_index=True, right_index=True)
```

In [17]:
```
# import additional libraries
from sklearn.model_selection import train_test_split
```

In [19]:
```
# splitting the data set into train and test sets
finaldf = finaldf.dropna()

X = finaldf.drop(['Churn'],axis=1)
y = finaldf['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [20]:
```
# import additional libraries to oversmaple the training data
from imblearn.over_sampling import SMOTE
```

In [21]:
```
# using smote to oversample the train data set
oversample = SMOTE(k_neighbors=5)
X_smote, y_smote = oversample.fit_resample(X_train, y_train)
X_train, y_train = X_smote, y_smote
```

In [22]:
```
# checking the retured value counts of the train data set
y_train.value_counts()
```

Out[22]:
```
Churn
0    3452
1    3452
Name: count, dtype: int64
```

# Model Building

It seems that a random forest classifier may be more useful in this case.

In [23]:
```
# import model library
from sklearn.ensemble import RandomForestClassifier
```

In [24]:
```
rf_model = RandomForestClassifier(random_state=46)
rf_model.fit(X_train,y_train)
```

Out[24]:
```
RandomForestClassifier
RandomForestClassifier(random_state=46)
```

## Evaluation

In [25]:
```
# import library for evaluating model
from sklearn.metrics import accuracy_score
```

In [26]:
```
predictions = rf_model.predict(X_test)
print(accuracy_score(predictions,y_test))
```

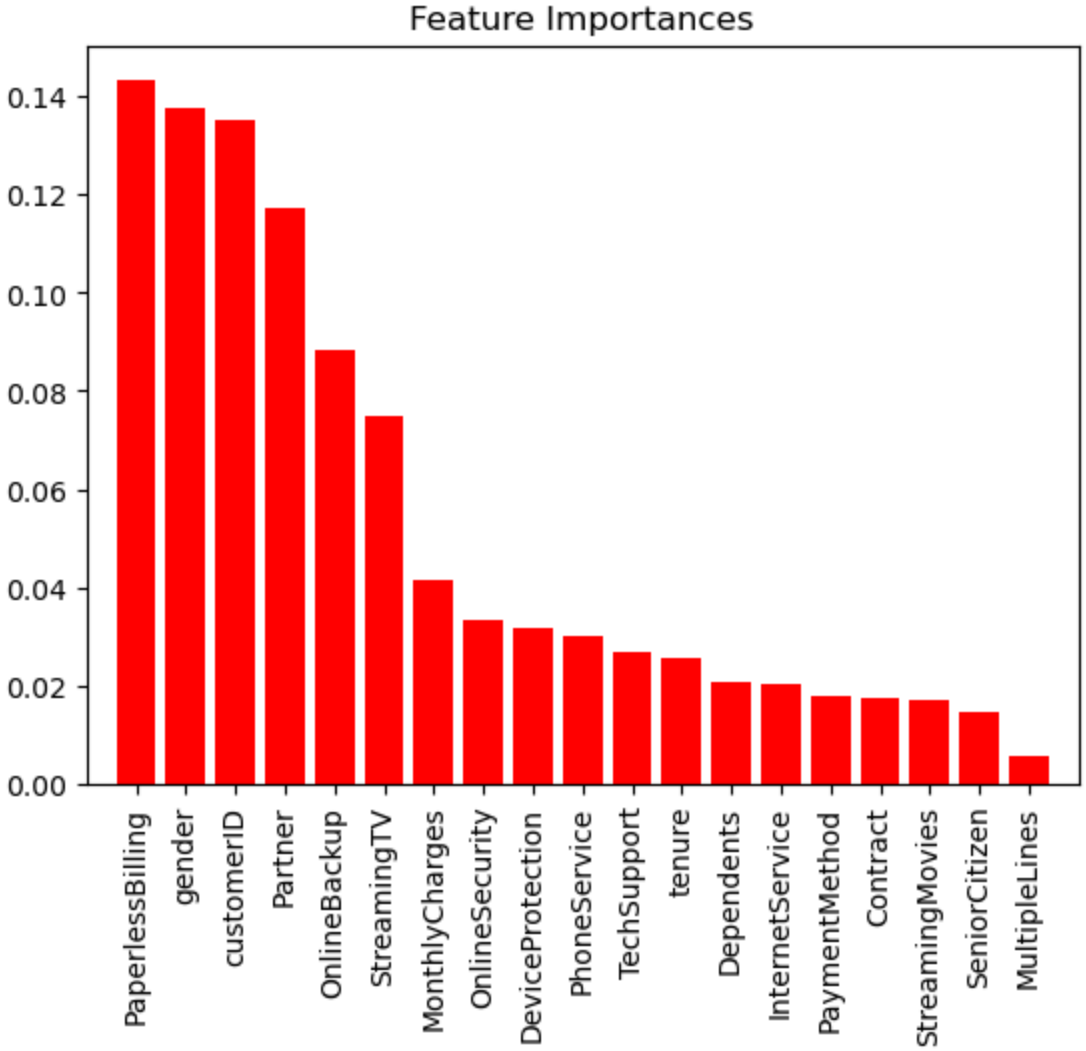
```
0.7699267557087462
```

In [27]:
```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
```

In [28]:
```
feature_names = df.columns.tolist()
```

In [29]:
```
# Feature Importance
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure()
plt.title("Feature Importances")
plt.bar(range(X.shape[1]), importances[indices], color="r", align="center")
plt.xticks(range(X.shape[1]), [feature_names[i] for i in indices], rotation=90)
plt.xlim([-1, X.shape[1]])
plt.show()
```


Feature Importances
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
PaperlessBilling
gender
customerID
Partner
OnlineBackup
StreamingTV
MonthlyCharges
OnlineSecurity
DeviceProtection
PhoneService
TechSupport
tenure
Dependents
InternetService
PaymentMethod
Contract
StreamingMovies
SeniorCitizen
MultipleLines