TELCO CUSTOMER CHURN DATASET PROJECT

NAME: KASINATHAN T
COURSE NAME: DATA ANALYTICS
BATCH NO: 2024-11989

INTRODUCTION

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

IMPORT LIBRARIES

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns
- from sklearn.model_selection import train_test_split
- from sklearn.linear_model import LogisticRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.svm import SVC
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.model_selection import GridSearchCV
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.metrics import accuracy_score
- import warnings
- warnings.filterwarnings('ignore')

READ THE DATASET AND FIND THE DATASET SHAPE AND DESCRIBE

- df = pd.read_csv('Telco-Customer-Churn.csv')
- df.head()
- SHAPE AND DESCRIBE FIND:
- df.shape
- `(7043, 21)`
- df.describe()
- SeniorCitizentenureMonthlyChargescount7043.0000007043.0000007043.000000mean0.16214732.37114964.761692std0.36861224.55948130.090047min0.0000000.00000018.25000025%0.0000009.00000035.50000050%0.00000029.00000070.35000075%0.00000055.00000089.850000max1.00000072.000000118.750000

As the datatype of Total charges column is wrongly interpreted as object, we change it to float

- df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')

FIND NULL VALUES

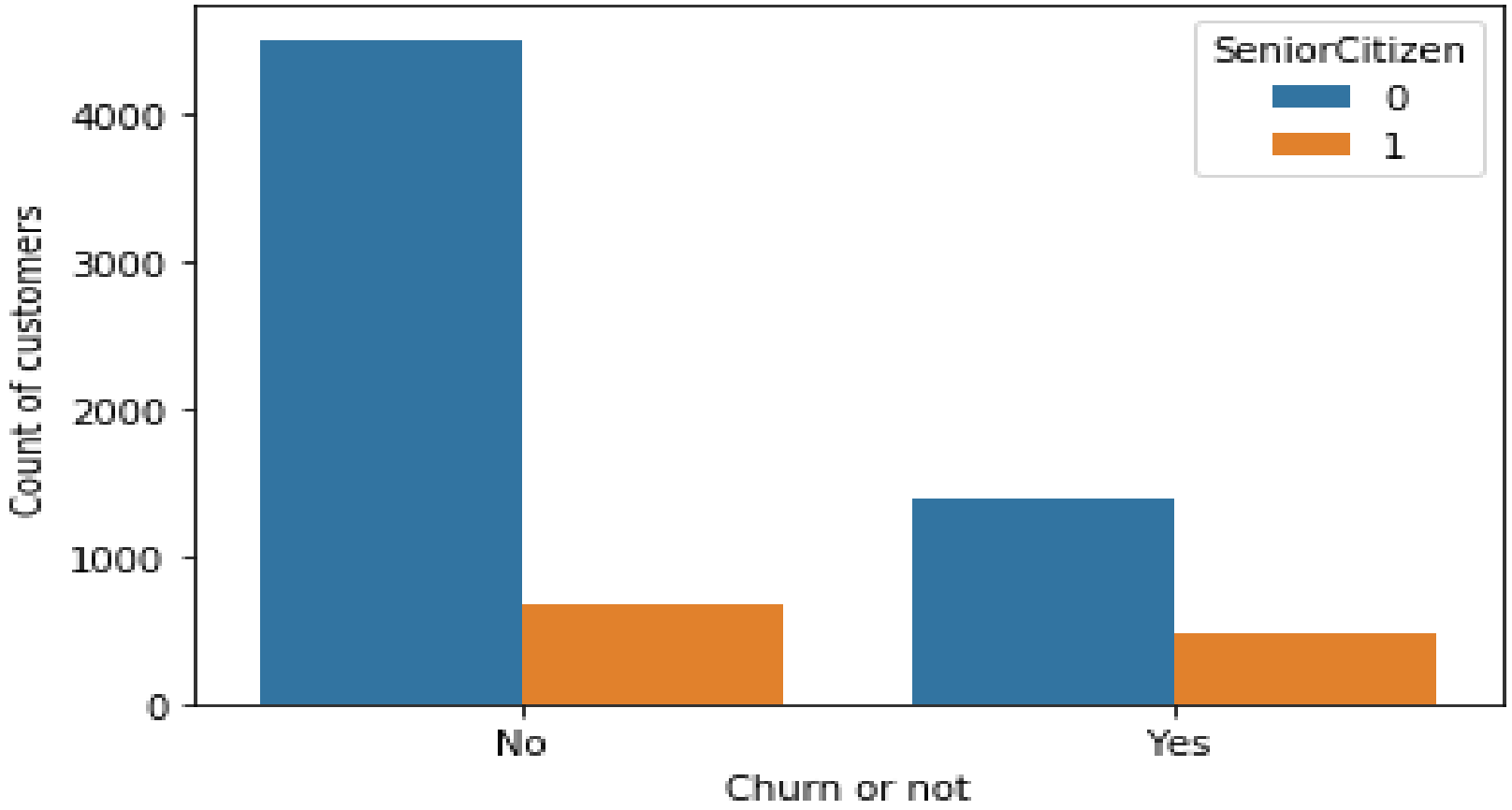
- df.isna().sum()
- customerID 0
- gender 0
- SeniorCitizen 0
- Partner 0
- Dependents 0
- tenure 0
- PhoneService 0
- MultipleLines 0
- InternetService 4
- OnlineSecurity 0
- OnlineBackup 0
- DeviceProtection 0
- TechSupport 0
- StreamingTV 0
- StreamingMovies 0

replacing null values with mean

- df['TotalCharges'].fillna(np.mean(df['TotalCharges']),inplace=True)
- df['MonthlyCharges'].fillna(np.mean(df['MonthlyCharges']),inplace=True)
- df.InternetService.fillna('Fiber optic',inplace=True)

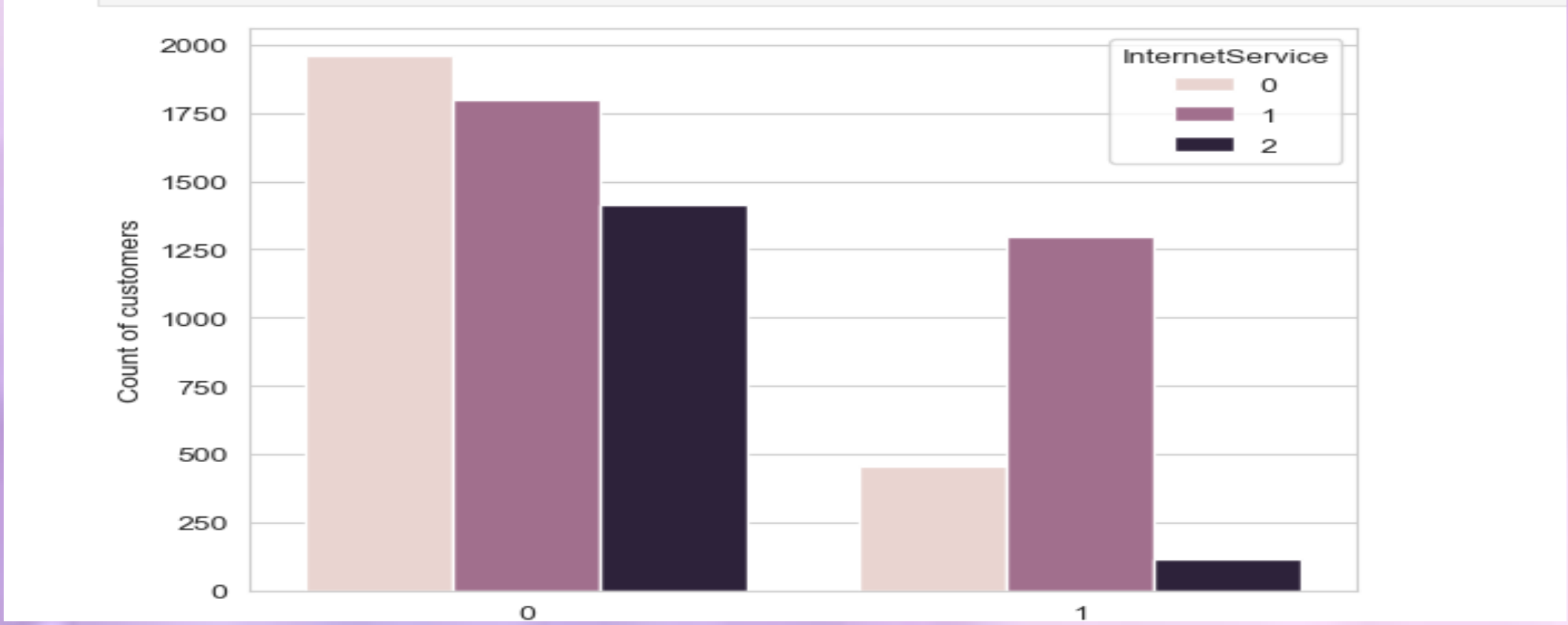
EDA OF Telco-Customer-Churn

```
#Senior citizen impact on churn
sns.countplot(x='Churn', hue="SeniorCitizen",data=df)
plt.xlabel("Churn or not")
plt.ylabel('Count of customers')
plt.show()
```



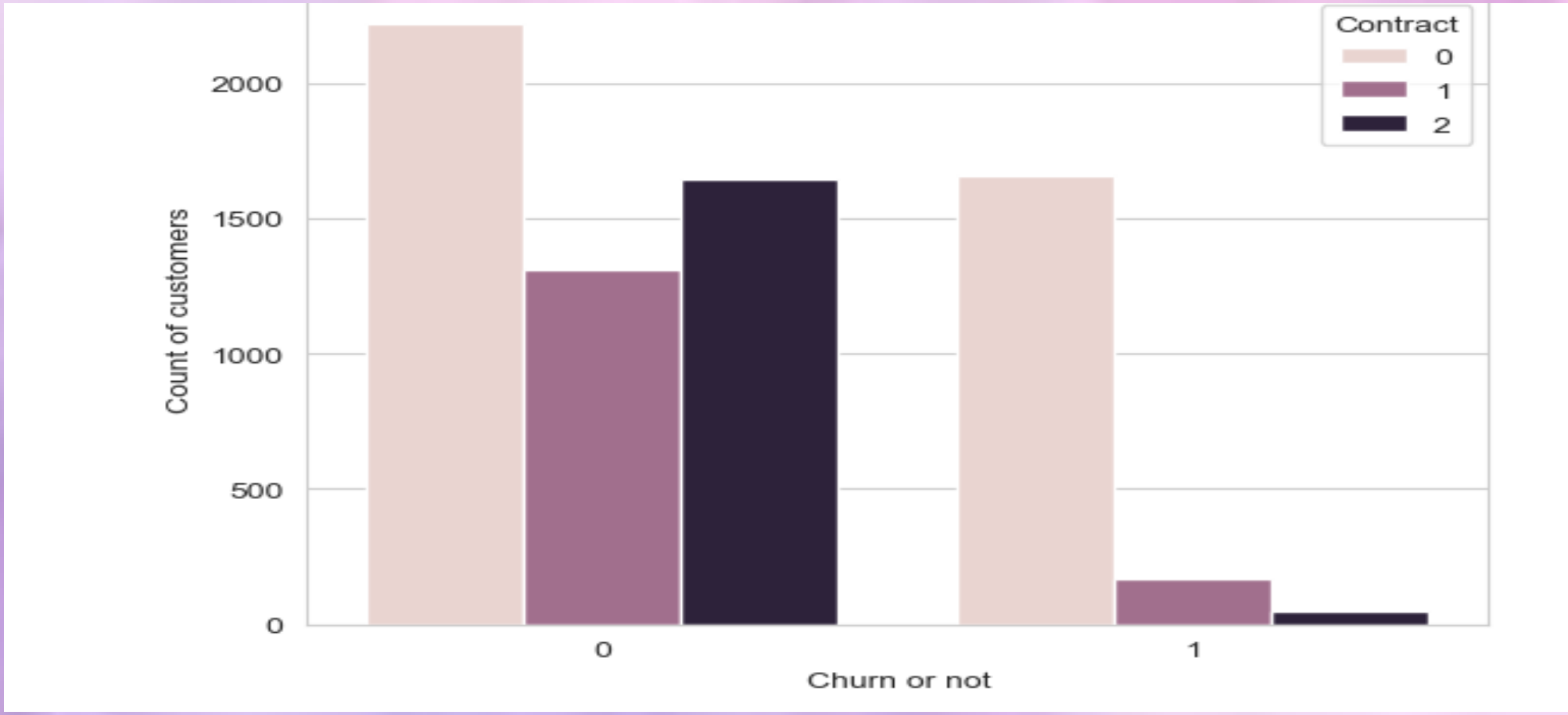
Internet service impact on churn

```
sns.countplot(x='Churn', hue="InternetService",data=df)
plt.xlabel("Churn or not")
plt.ylabel('Count of customers')
plt.show()
```



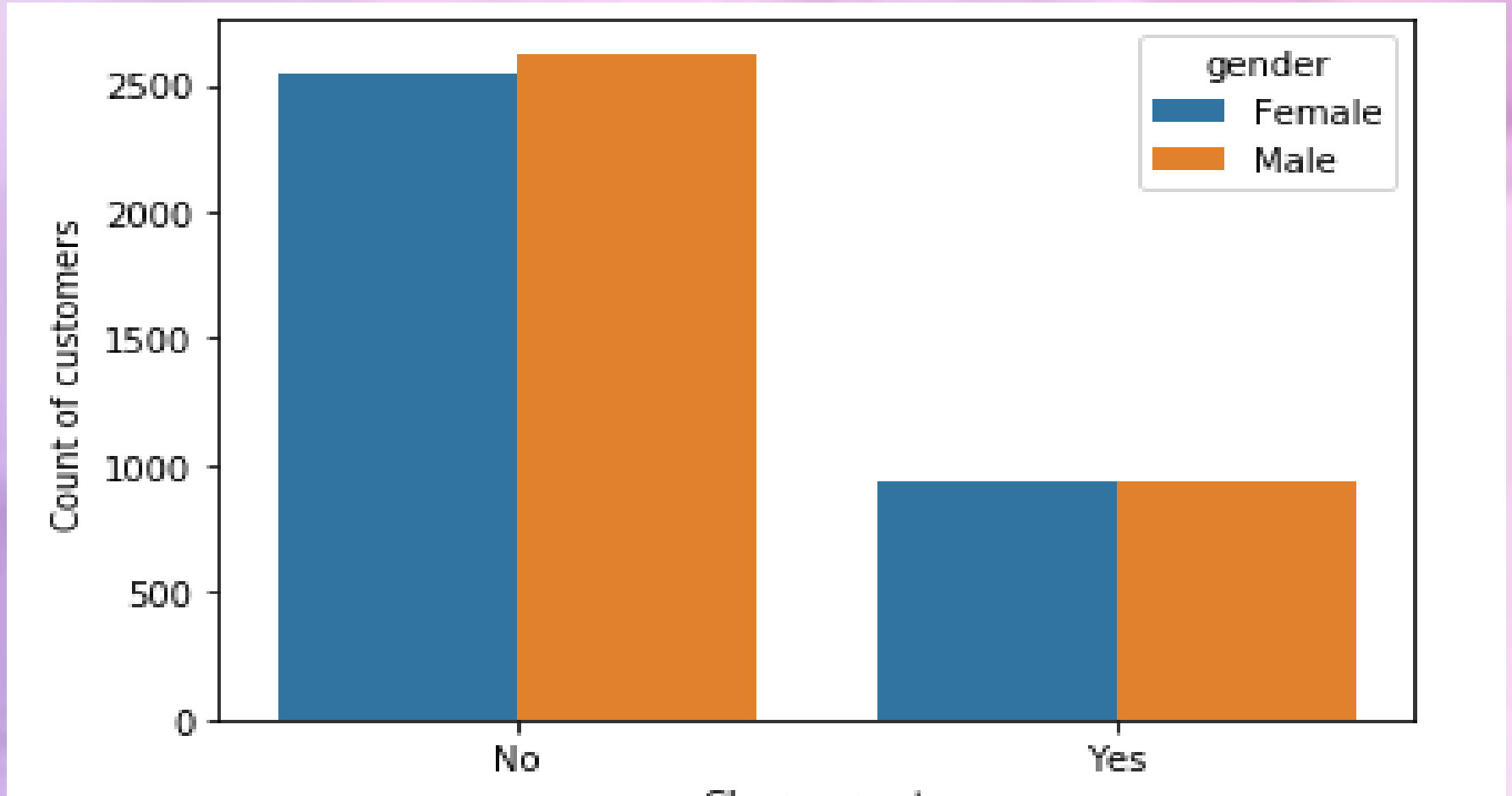
Contract impact on churn

```
sns.countplot(x='Churn', hue="Contract",data=df)
plt.xlabel("Churn or not")
plt.ylabel('Count of customers')
plt.show()
```

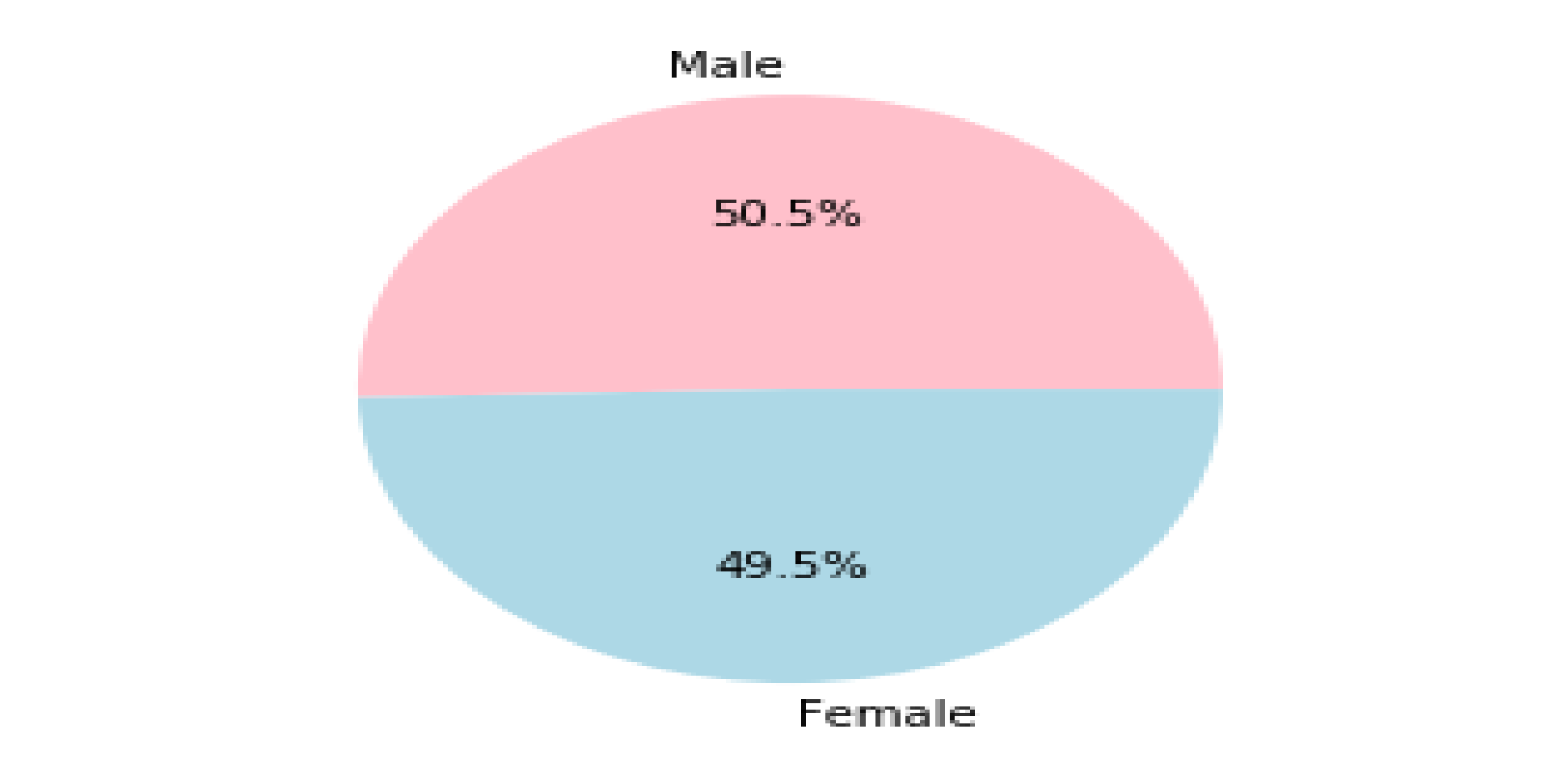


Gender impact on churn

```
sns.countplot(x='Churn', hue="gender",data=df)
plt.xlabel("Churn or not")
plt.ylabel('Count of customers')
plt.show()
```

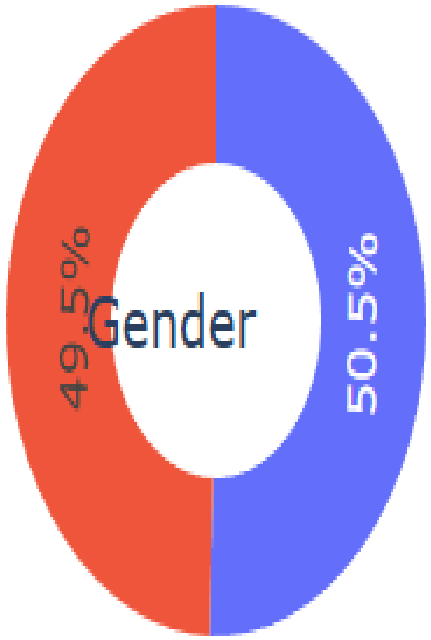


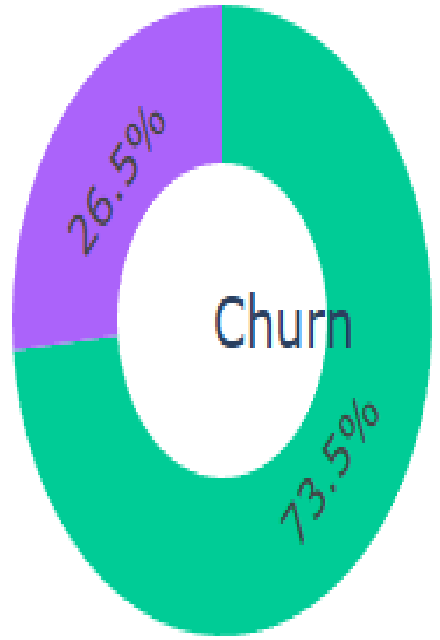
```
gender_count = df.gender.value_counts()
plt.pie(gender_count.values, labels=gender_count.index, autopct='%0.1f%%', colors =
['Pink','Lightblue'])
plt.show()
```



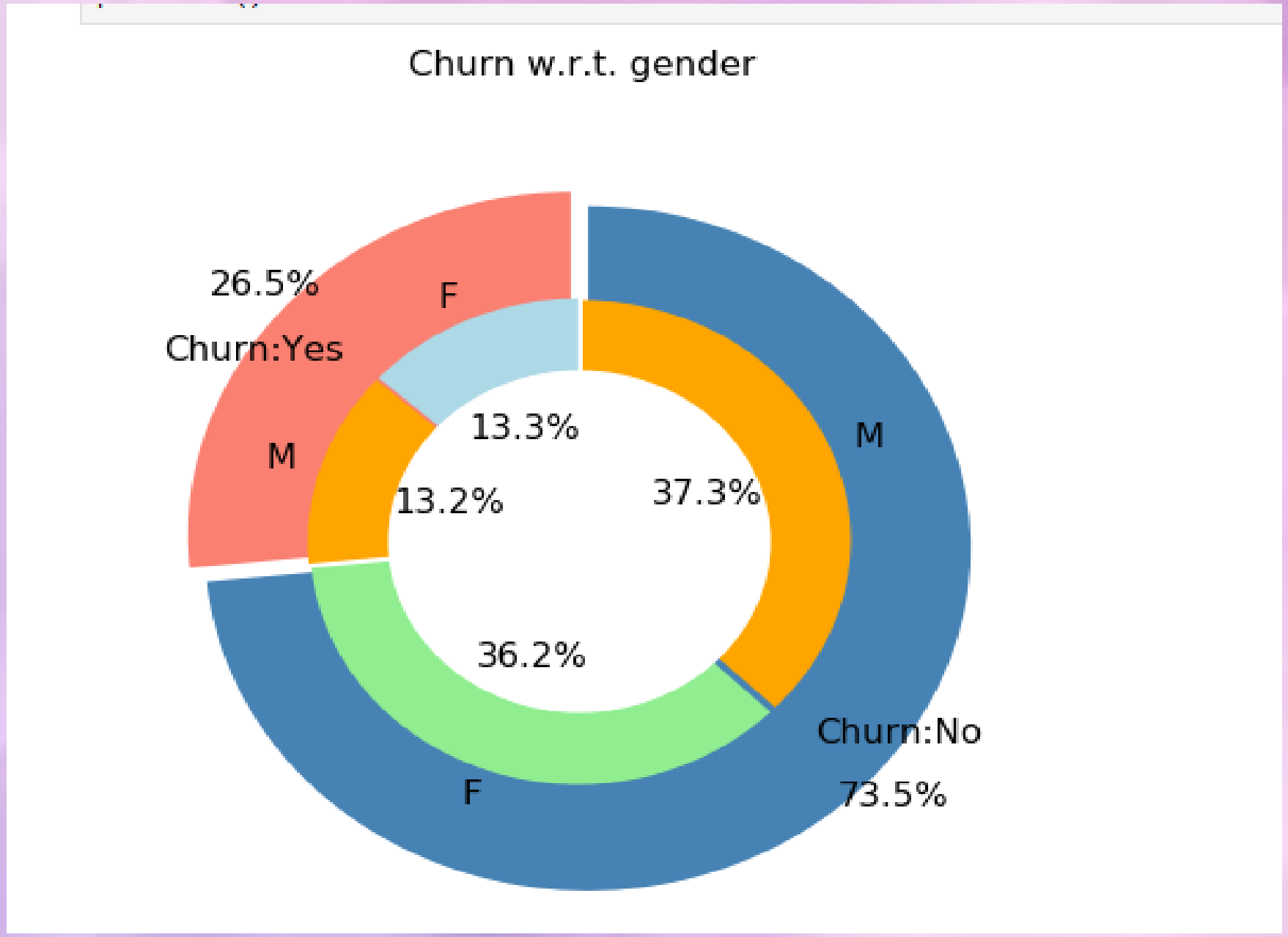
```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
gen_labels = ['Male','Female']
churn_labs = ['No Churn','Churned']

fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'},{'type':'domain'}]])
fig.add_trace(go.Pie(labels = gen_labels, values = df['gender'].value_counts(),name = 'Gender'),
              1,1)

fig.add_trace(go.Pie(labels=churn_labs, values = df['Churn'].value_counts(), name = 'Churn'),
              1,2)
plt.show()

#Converting above pie chart into doughnut chart
fig.update_traces(hole = .5, hoverinfo = 'label+percent+name', textfont_size = 16)

fig.update_layout(title_text = 'Gender Vs Churn Distribution',
                  annotations = [dict(text='Gender',x=0.16, y=0.5, font_size = 20, showarrow = False),
                                 dict(text = 'Churn', x=0.84, y=0.5, font_size = 20, showarrow = False)])

fig.show()
```

Gender Vs Churn Distribution



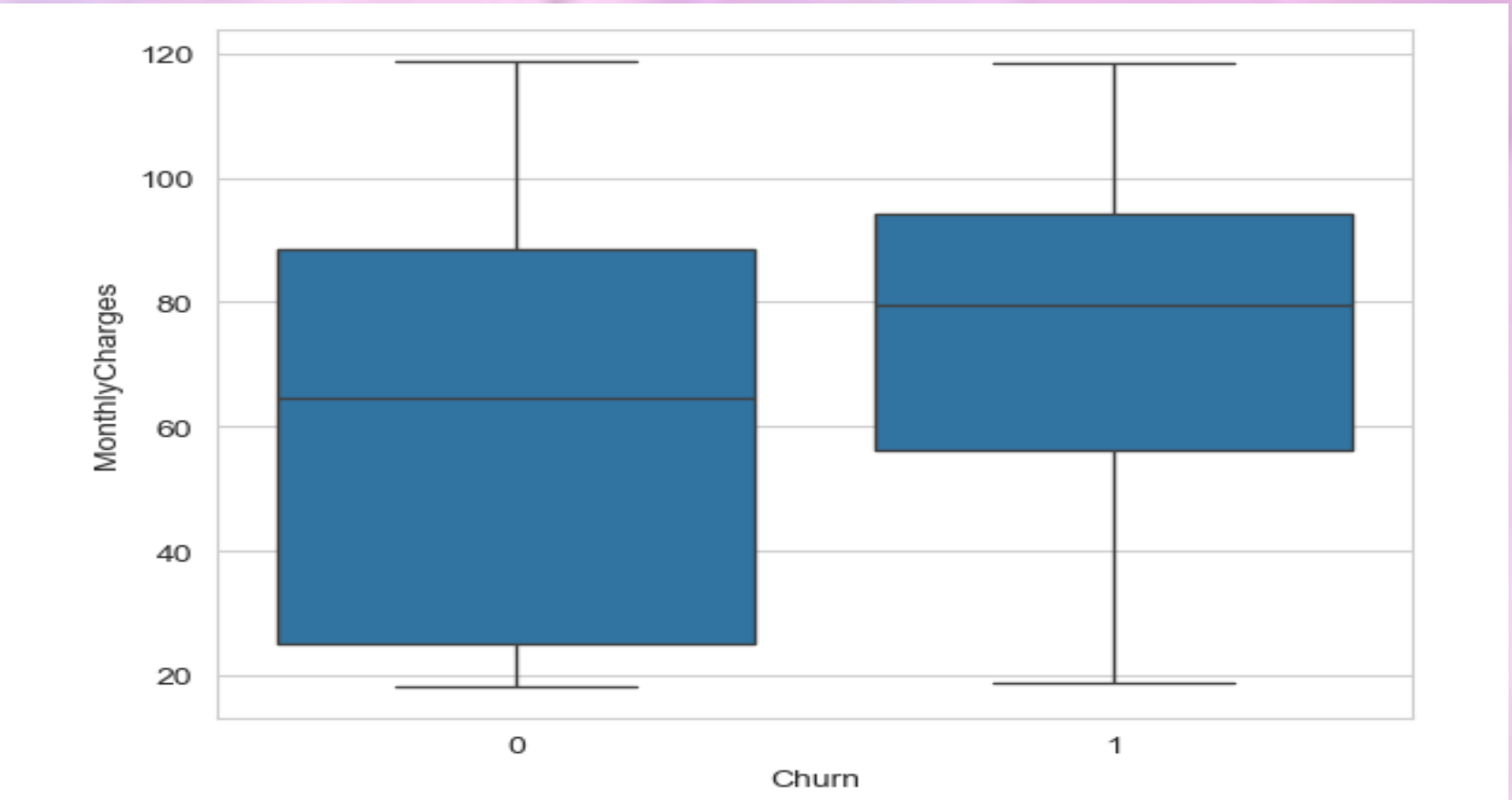


```
plt.figure(figsize=(6,6))
labels = ['Churn:Yes','Churn:No']
values = [1869,5174]
gen_lab = ['F','M','F','M']
gen_count = [939,930,2549,2625]
colors = ['Salmon','Steelblue']
color_gen = ['LightBlue','Orange','Lightgreen','Orange']
explode = (0.3,0.3)
explod_gen = [0.1,0.1,0.1,0.1]
textprop = {'fontsize':15}

#Plot Pie charts
plt.pie(values,labels=labels,autopct="%1.1f%%", pctdistance=1.08,
    labeldistance=0.8, colors = colors,startangle=90, frame=True,
    explode=explode, radius=10, textprops=textprop)

plt.pie(gen_count, labels=gen_lab, autopct="%1.1f%%", pctdistance=0.5,
    colors=color_gen, startangle=90, explode=explod_gen, radius=7, textprops=textprop)

centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth = 0)
fig=plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title("Churn w.r.t. gender", fontsize=15, y=1.1)

plt.axis('equal')
plt.tight_layout()
plt.show()
```

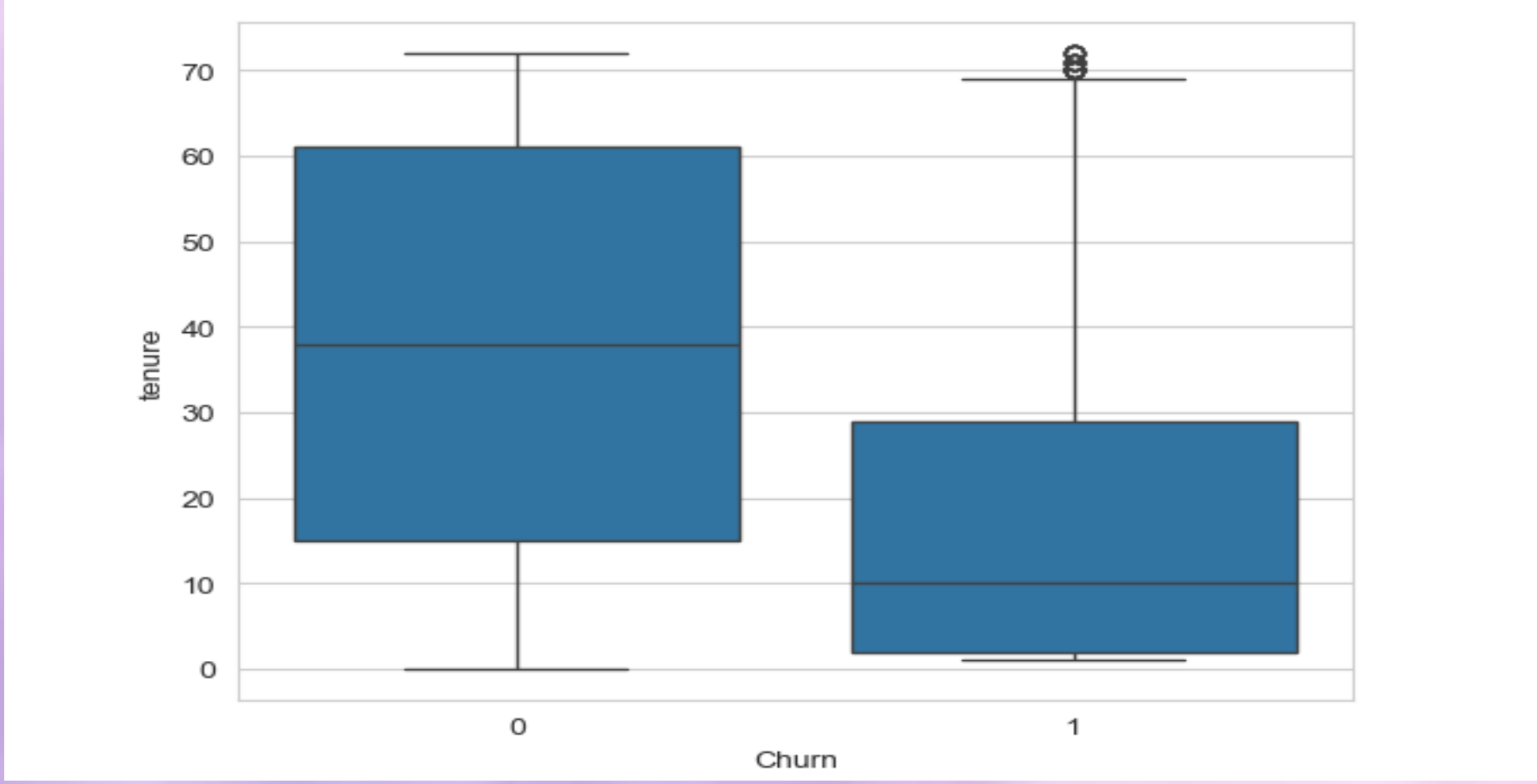

Churn w.r.t. gender
26.5%
F
Churn:Yes
M
13.3%
13.2%
37.3%
M
36.2%
F
Churn:No
73.5%

Montly charges distribution usign box plot for churn and not churn

```
sns.boxplot(x = 'Churn', y='MonthlyCharges', data=df)
plt.show()
```



```
sns.boxplot(x = 'Churn', y='tenure', data=df)
plt.show()
```



Apply for mechine learning algorithm in Telco-Customer-Churn

- splitting into input and output

- x=df.iloc[: ,1 : -1]
- y=df['Churn']
- print(x.shape)
- print(y.shape)
- (7043, 19)
- (7043,)
- x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
- print(x_train.shape)
- print(x_test.shape)
- (5282, 19)
- (1761, 19)

Apply for Logistic regression

- Accuracies=[]
- cls= LogisticRegression()
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('average logistic regression',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `average logistic regression 81.09028960817717`

Apply for KNeighborsClassifier

- cls= KNeighborsClassifier(n_neighbors=10)
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('Accuracy of KNeighborsClassifier',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `Accuracy of KNeighborsClassifier 78.08063600227145`

Apply for DecisionTreeClassifier

- cls= DecisionTreeClassifier(max_depth=3)
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('Accuracy of DecisionTreeClassifier',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `Accuracy of DecisionTreeClassifier 78.87563884156728`

Apply for SVM

- cls= SVC()
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('Accuracy of SVM',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `Accuracy of SVM 73.93526405451448`

Apply for naive bayes

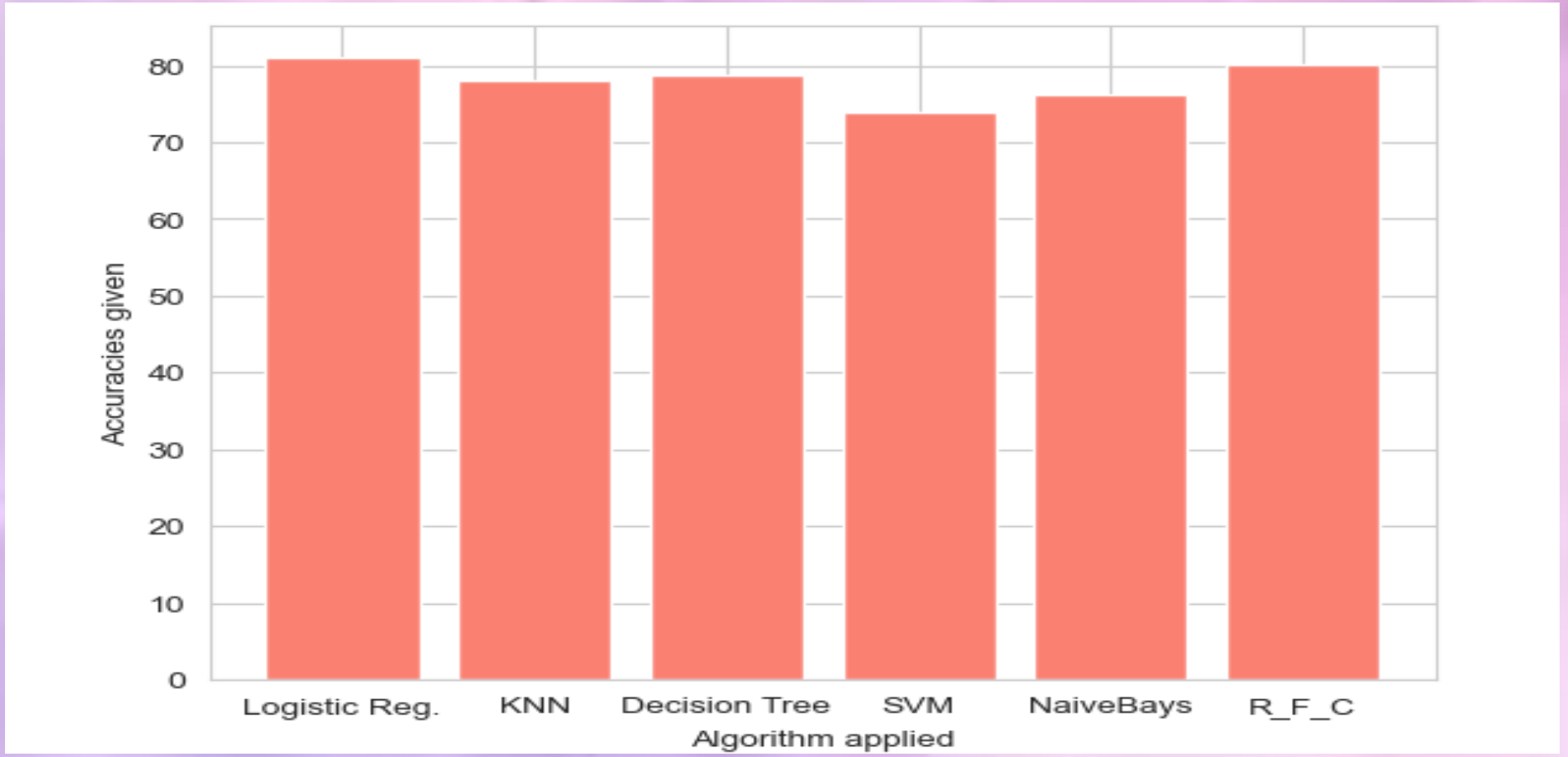
- cls= GaussianNB()
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('Accuracy of SVM',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `Accuracy of SVM 76.32027257240205`

Apply for RandomForestClassifier

- cls= RandomForestClassifier(max_depth=5)
- cls.fit(x_train,y_train)
- y_pred=cls.predict(x_test)
- print('Accuracy of RandomForestClassifier',accuracy_score(y_test,y_pred)*100)
- Accuracies.append(accuracy_score(y_test,y_pred)*100)
- `Accuracy of RandomForestClassifier 80.23850085178876`

comparision of accuracy score

```
algos=['Logistic Reg.','KNN','Decision Tree','SVM','NaiveBays','R_F_C']
plt.bar(algos,Accuracies,color='salmon')
plt.xlabel('Algorithm applied')
plt.ylabel('Accuracies given')
plt.show()
```



THANK YOU

GIVE A OPPORTUNITY