Telecom Customer Churn Prediction

The aim of this project is to analyze customer demographics, services, tenure and other variables to predict whether a particular customer will churn or not.

Data Dictionary

Variable	Description
CustomerID	Unique customer ID
Gender	Customer's gender
SeniorCitizen	Whether the customer is a senior citizen or not (1, 0)
Partner	Whether the customer has a partner or not (Yes, No)
Dependents	Whether the customer has dependents or not (Yes, No)
Tenure	Number of months the customer has stayed with the company
PhoneService	Whether the customer has a phone service or not (Yes, No)
MultipleLines	Whether the customer has multiple lines or not (Yes, No, No phone service)
InternetService	Customer’s internet service provider (DSL, Fiber optic, No)
OnlineSecurity	Whether the customer has online security or not (Yes, No, No internet service)
OnlineBackup	Whether the customer has online backup or not (Yes, No, No internet service)
DeviceProtection	Whether the customer has device protection or not (Yes, No, No internet service)
TechSupport	Whether the customer has tech support or not (Yes, No, No internet service)
StreamingTV	Whether the customer has streaming TV or not (Yes, No, No internet service)
StreamingMovies	Whether the customer has streaming movies or not (Yes, No, No internet service)
Contract	The contract term of the customer (Month-to-month, One year, Two year)
PaperlessBilling	Whether the customer has paperless billing or not (Yes, No)
PaymentMethod	The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
MonthlyCharges	The amount charged to the customer monthly
TotalCharges	The total amount charged to the customer
Churn	Whether the customer churned or not (Yes or No)

In [ ]:
```
#Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#Loading the dataset
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head()
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mul
0	7590-VHVEG	Female	0	Yes	No	1	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

Data Preprocessing Part 1

In [ ]:
```
#Checking the shape of the dataset
df.shape
```

Out[ ]: (7043, 21)

In [ ]:
```
#Removing the customerID column
df.drop(columns='customerID', inplace=True)
```

In [ ]:
```
#Checking for duplicate values
df.duplicated().sum()
```

Out[ ]: 22

In [ ]:
```
#Removing the duplicate values
df.drop_duplicates(inplace=True)
```

In [ ]:
```
#Checking the datatypes of the columns
df.dtypes
```

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

Type casting column Total Charges

In [ ]:
```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

In [ ]:
```
#Checking for null values
df.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
#Droping the null values
df.dropna(inplace=True)
```

In [ ]:
```
#checking number of unique values in each column
df.nunique()
```

Out[ ]:
```
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                72
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1584
TotalCharges        6530
Churn                  2
dtype: int64
```

In [ ]:
```
#Checking the unique values in each column
cols = df.columns
for i in cols:
    print(i, df[i].unique(), '\n')
```

```
gender ['Female' 'Male']

SeniorCitizen [0 1]

Partner ['Yes' 'No']

Dependents ['No' 'Yes']

tenure [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]

PhoneService ['No' 'Yes']

MultipleLines ['No phone service' 'No' 'Yes']

InternetService ['DSL' 'Fiber optic' 'No']

OnlineSecurity ['No' 'Yes' 'No internet service']

OnlineBackup ['Yes' 'No' 'No internet service']

DeviceProtection ['No' 'Yes' 'No internet service']

TechSupport ['No' 'Yes' 'No internet service']

StreamingTV ['No' 'Yes' 'No internet service']

StreamingMovies ['No' 'Yes' 'No internet service']

Contract ['Month-to-month' 'One year' 'Two year']

PaperlessBilling ['Yes' 'No']

PaymentMethod ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']

MonthlyCharges [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]

TotalCharges [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]

Churn ['No' 'Yes']
```

Descriptive Statistics

In [ ]:
```
df.describe()
```

Out[ ]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7010.000000	7010.000000	7010.000000	7010.000000
mean	0.162767	32.520399	64.888666	2290.353388
std	0.369180	24.520441	30.064769	2266.820832
min	0.000000	1.000000	18.250000	18.800000
25%	0.000000	9.000000	35.750000	408.312500
50%	0.000000	29.000000	70.400000	1403.875000
75%	0.000000	56.000000	89.900000	3807.837500
max	1.000000	72.000000	118.750000	8684.800000

In [ ]:

```
df.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
0	Female	0	Yes	No	1	No	No phone service	
1	Male	0	No	No	34	Yes	No	
2	Male	0	No	No	2	Yes	No	
3	Male	0	No	No	45	No	No phone service	
4	Female	0	No	No	2	Yes	No	

Exploratory Data Analysis

In the exploratory data analysis, I will be visualizing the data to get a better understanding of the data and to see if there are any trends or pattern in the data. First I will begin with looking at the distribution of the data and then I will look at the relationship between the independent variables and the target variable.

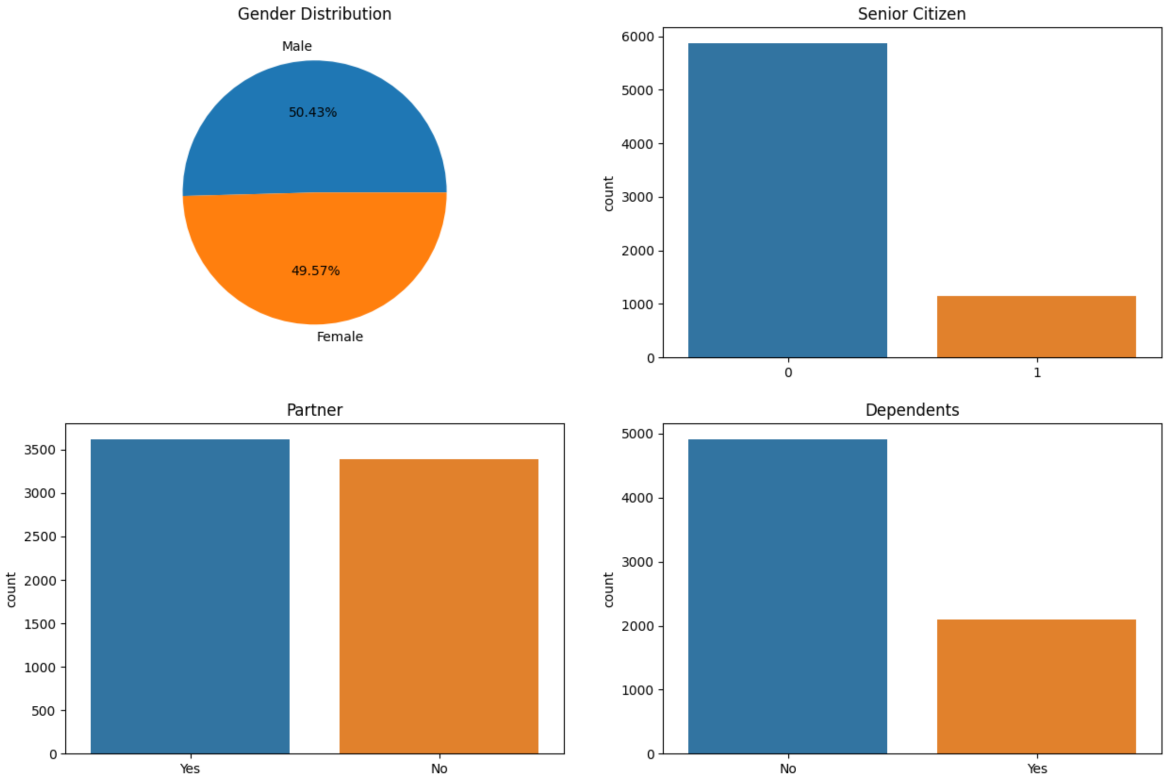
Customer Demographics

In [ ]:

```
fig, ax = plt.subplots(2, 2, figsize=(15, 10))

#Gender Distribution
ax[0,0].pie(df['gender'].value_counts(), labels  = ['Male', 'Female'], autopct =
ax[0,0].set_title('Gender Distribution')

#Senior Citizen Distribution
```

```
sns.barplot(y = df['SeniorCitizen'].value_counts(), x = df['SeniorCitizen'].unic

#Partner Distribution
sns.barplot(y = df['Partner'].value_counts(), x = df['Partner'].unique(), ax=ax[

#Dependents Distribution
sns.barplot(y = df['Dependents'].value_counts(), x = df['Dependents'].unique(),
```

Out[ ]: `Text(0.5, 1.0, 'Dependents')`



These graphs shows the customer demographics. The number of males and females is almost same, with few more males than females in the dataset. Majority of them are not senior citizen. Nearly 3500, customers have a partner and similar number of cutomers don't. Majority of the customers don't have dependents, but still a significant number does have dependents.

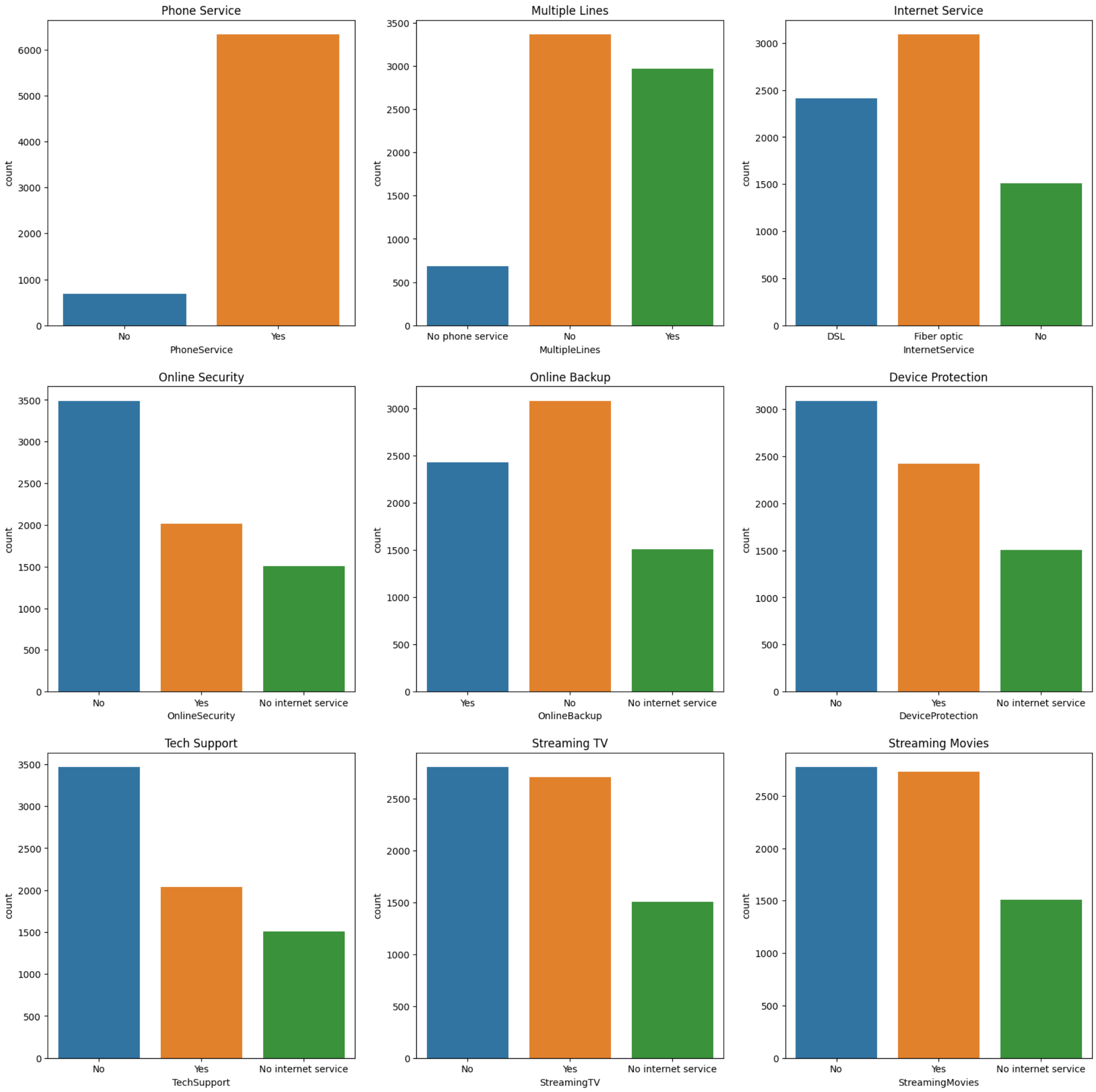
From these graphs, we get know about the customers demographics, which help us to get an idea of their psychology based on their age, relationship status, and dependents.

Services

In [ ]:
```
fig, ax = plt.subplots(3, 3, figsize=(20, 20))
#phone service
sns.countplot(x = df['PhoneService'], ax=ax[0,0]).set_title('Phone Service')
ax[0,0].set_title('Phone Service')
#Multiple Lines
sns.countplot(x = df['MultipleLines'], ax=ax[0,1]).set_title('Multiple Lines')
ax[0,1].set_title('Multiple Lines')
#Internet Service
sns.countplot(x = df['InternetService'], ax=ax[0,2]).set_title('Internet Service
ax[0,2].set_title('Internet Service')
#Online Security
sns.countplot(x = df['OnlineSecurity'], ax=ax[1,0]).set_title('Online Security')
```

```
ax[1,0].set_title('Online Security')
#Online Backup
sns.countplot(x = df['OnlineBackup'], ax=ax[1,1]).set_title('Online Backup')
ax[1,1].set_title('Online Backup')
#Device Protection
sns.countplot(x = df['DeviceProtection'], ax=ax[1,2]).set_title('Device Protecti
ax[1,2].set_title('Device Protection')
#Tech Support
sns.countplot(x = df['TechSupport'], ax=ax[2,0]).set_title('Tech Support')
ax[2,0].set_title('Tech Support')
#Streaming TV
sns.countplot(x = df['StreamingTV'], ax=ax[2,1]).set_title('Streaming TV')
ax[2,1].set_title('Streaming TV')
#Streaming Movies
sns.countplot(x = df['StreamingMovies'], ax=ax[2,2]).set_title('Streaming Movies
ax[2,2].set_title('Streaming Movies')
```

Out[ ]: Text(0.5, 1.0, 'Streaming Movies')



The above graphs visualizes the services taken by the customers from the telecom company. Nearly 6000 customers have taken phone service. However, nealry half of the cutomers have taken multiplt lines from the company. Almost 5500, have taken internet services from the company, where 3000 customers opted fibre optcs and rest of them opted DSL which could possible for business purposes. From these three major services related to telecom, the phone services and the internet services are the most popular services among the customers.

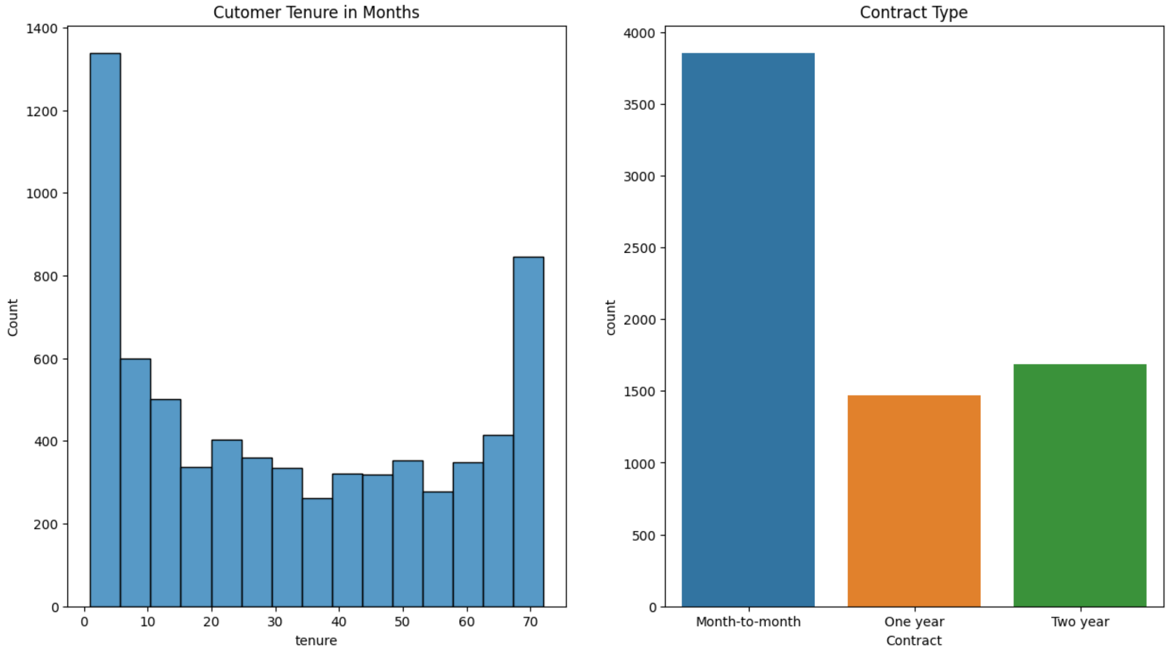
Coming to other services which includes- Online Security, Online Backup, Device Protection, Tech Support, and Streaming Services. The online backup and device protection service is opted by almost 2500 customers, which higlights the customers concern regarding their device safety and data protection. The online security and tech support is opted by almost 2000 customers which are least opted services among the customers. The streaming services are the most popular services, with more than 2500 customers opting for it.

From this,I conclude that part from the internet and phone services, the streaming services are most opted ones. Therefore, the company should focus on providing better streaming services to the customers.

Tenure and Contract

In [ ]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))
sns.histplot(x = 'tenure', data = df, ax= ax[0]).set_title('Cutomer Tenure in Mc
sns.countplot(x = 'Contract', data = df, ax= ax[1]).set_title('Contract Type')
```

Out[ ]:
```
Text(0.5, 1.0, 'Contract Type')
```

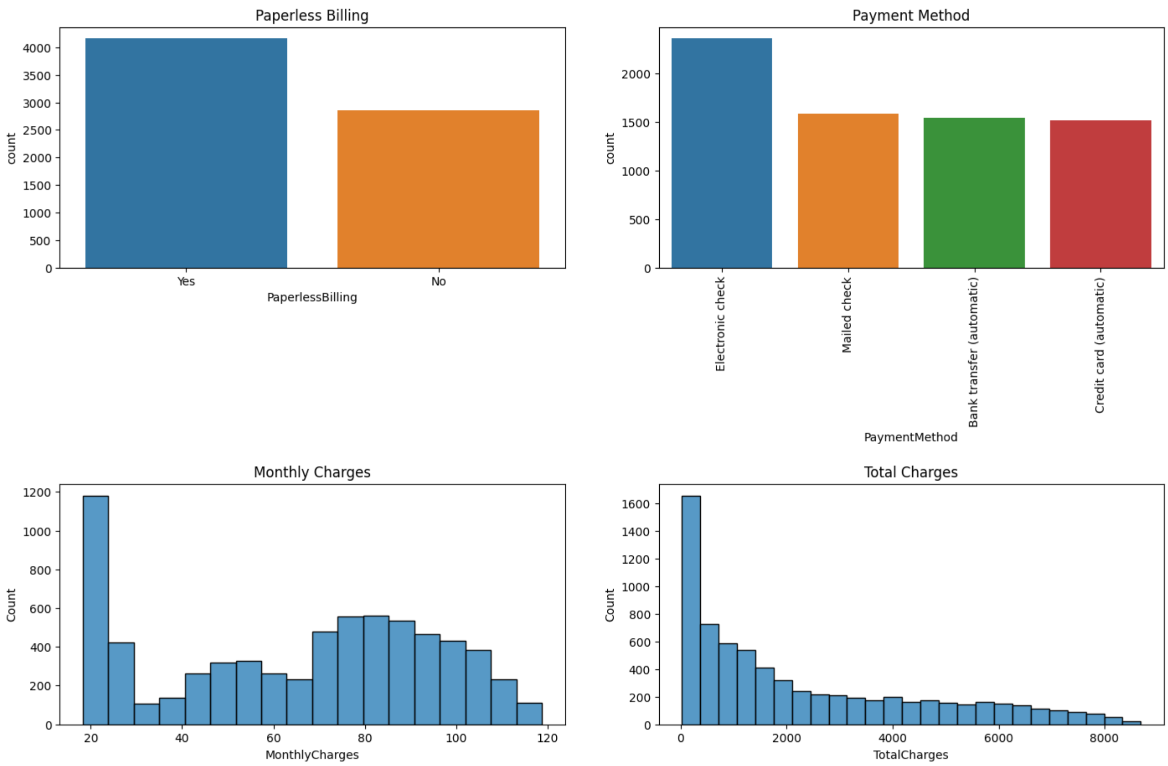


In the aboove graphs we can see the distribution of customer tenure with the comapny and the count of the type of contarct the company had with customer. Here, most if the customers had tenure less than a month, and most of the customers had a month-to-month contract with the company. Therfore, the customers with shorter tenure have month-to-month contract with the company. In addtion to that, a significant number of customers have tenure of nealry 70 months, which highlights the loyalty of the customers towards the company. Moreover, after month-to-month contract, the second most popular contract is two-year contract, which is opted by almost 1700 customers. Rest of the customers have tenure between 1-5 years.

Billing and Charges

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 10))
fig.tight_layout(pad=5.0)

#spacing between subplots
fig.subplots_adjust(hspace=0.9)


#papaerless billing
sns.countplot(x = df['PaperlessBilling'], ax=ax[0,0]).set_title('Paperless Billi

#Payment Method
sns.countplot(x = df['PaymentMethod'], ax=ax[0,1]).set_title('Payment Method')
ax[0,1].xaxis.set_tick_params(rotation=90)

#Monthly Charges
sns.histplot(x = 'MonthlyCharges', data = df, ax = ax[1,0]).set_title('Monthly C

#Total Charges
sns.histplot(x = 'TotalCharges', data = df, ax = ax[1,1]).set_title('Total Charg
```

Out[ ]:
```
Text(0.5, 1.0, 'Total Charges')
```



These graphs shows the method of billing and the bill amounts. Most of the customers, nearly 4000 prefer paperless billing, however, a little bit over half of them pays through electronic check. But still a significant number of customers prefer paper bills. Apart from electronic checks, the other modes of payment accepted by the company includes - mailed checks, bank transfer and credit cards. Nearly 4500 customers altogether prefer these modes of payment.

Now, for the montly charges, huge number of customers pays near 20 dollars for the montly services and majority of the customer having total charges less than 200 dollars.

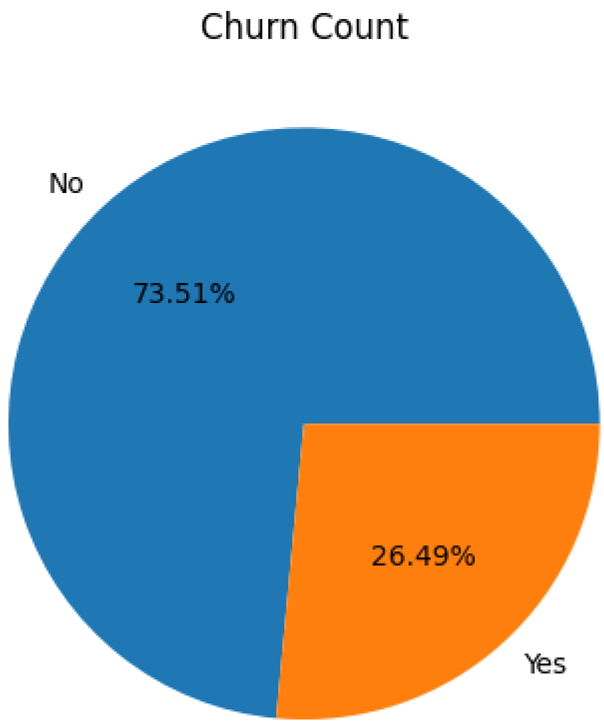
However, there are considerable number of customers having monthly charges between 70 to 100 dollars and total charges between 200-800 dollars. Interestingly, If we look at the total charges graph, we can see that some customers have a total bill more than 4000 and even 8000 as well. This could be possible, if the customer has a long tenure or uses alot of services.

Now, I conclude that company mainly have customers with low charges, which means comany should focus on these customers by providing even more afforadable services.

Churn Count

In [ ]:
```
plt.pie(x = df['Churn'].value_counts(), labels = df['Churn'].unique(), autopct =
plt.title('Churn Count')
```

Out[ ]:
```
Text(0.5, 1.0, 'Churn Count')
```



In the dataset, the number of churning customers is very less as compared to non churning. Only 26.49% churnned from the telecom company. This could be a potential proof, that company is quite good at retaning its customers.

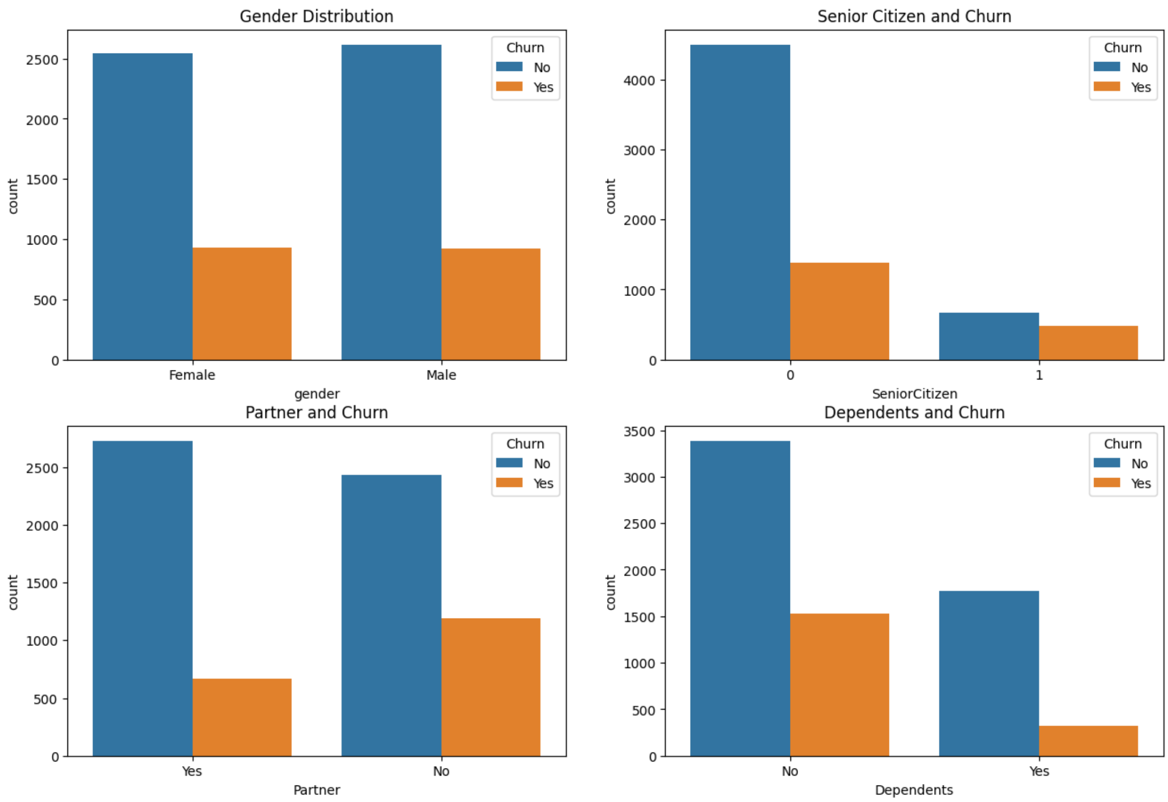
Till now, I have visualized the data and got a better understanding of the data. Now, I will look at the relationship between the independent variables and the target variable.

Customer Demogrpahics and Churn

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 10))

#Gender Distribution
sns.countplot(x = 'gender', data = df, hue = 'Churn', ax=ax[0,0])
```

```
ax[0,0].set_title('Gender Distribution')

#Senior Citizen Distribution
sns.countplot(x = df['SeniorCitizen'], ax=ax[0,1], hue = df['Churn']).set_title(

#Partner Distribution
sns.countplot( x = df['Partner'], ax=ax[1,0], hue = df['Churn']).set_title('Part

#Dependents Distribution
sns.countplot(x = df['Dependents'], ax=ax[1,1], hue = df['Churn']).set_title('De
```

Out[ ]: Text(0.5, 1.0, 'Dependents and Churn')



From these graphs, we can get know about the relation between customer demographics and customer churn. Both makes and females have equal number of churn count, so there is not relation between gender and customer churn. However, the senior citizens have a lesser churn count as compared to non senior citizens, which may be because their age and they don't want to hasle with the process of changing the telecom company. The customers with no partners have higher churn count as compared to customers with partners. Similarly, customers with no dependents have higher churn count as compared to customers with dependents.

From this I conclude that customers whom are single with no partner or have no dependents have higher churn count and senior citizens have lower churn count.

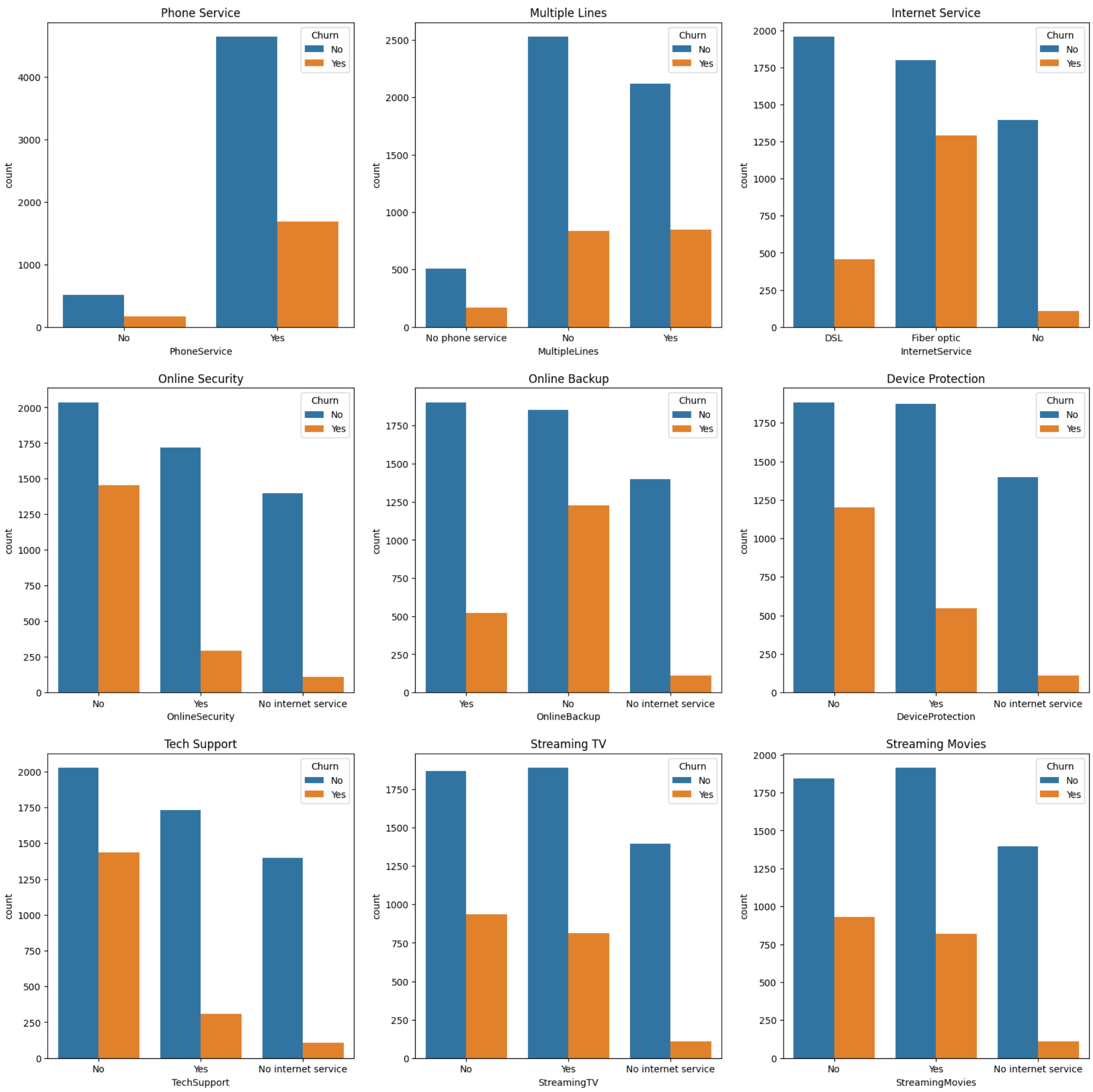
Services and Churn

In [ ]:

```
fig, ax = plt.subplots(3, 3, figsize=(20, 20))
#phone service
sns.countplot(x = df['PhoneService'], ax=ax[0,0], hue = df['Churn']).set_title('
ax[0,0].set_title('Phone Service')
#Multiple Lines
```

```
sns.countplot(x = df['MultipleLines'], ax=ax[0,1], hue = df['Churn']).set_title(
ax[0,1].set_title('Multiple Lines')
#Internet Service
sns.countplot(x = df['InternetService'], ax=ax[0,2], hue = df['Churn']).set_titl
ax[0,2].set_title('Internet Service')
#Online Security
sns.countplot(x = df['OnlineSecurity'], ax=ax[1,0], hue = df['Churn']).set_title
ax[1,0].set_title('Online Security')
#Online Backup
sns.countplot(x = df['OnlineBackup'], ax=ax[1,1], hue = df['Churn']).set_title('
ax[1,1].set_title('Online Backup')
#Device Protection
sns.countplot(x = df['DeviceProtection'], ax=ax[1,2], hue = df['Churn']).set_tit
ax[1,2].set_title('Device Protection')
#Tech Support
sns.countplot(x = df['TechSupport'], ax=ax[2,0], hue = df['Churn']).set_title('T
ax[2,0].set_title('Tech Support')
#Streaming TV
sns.countplot(x = df['StreamingTV'], ax=ax[2,1], hue = df['Churn']).set_title('S
ax[2,1].set_title('Streaming TV')
#Streaming Movies
sns.countplot(x = df['StreamingMovies'], ax=ax[2,2], hue = df['Churn']).set_titl
ax[2,2].set_title('Streaming Movies')
```

Out[ ]: Text(0.5, 1.0, 'Streaming Movies')



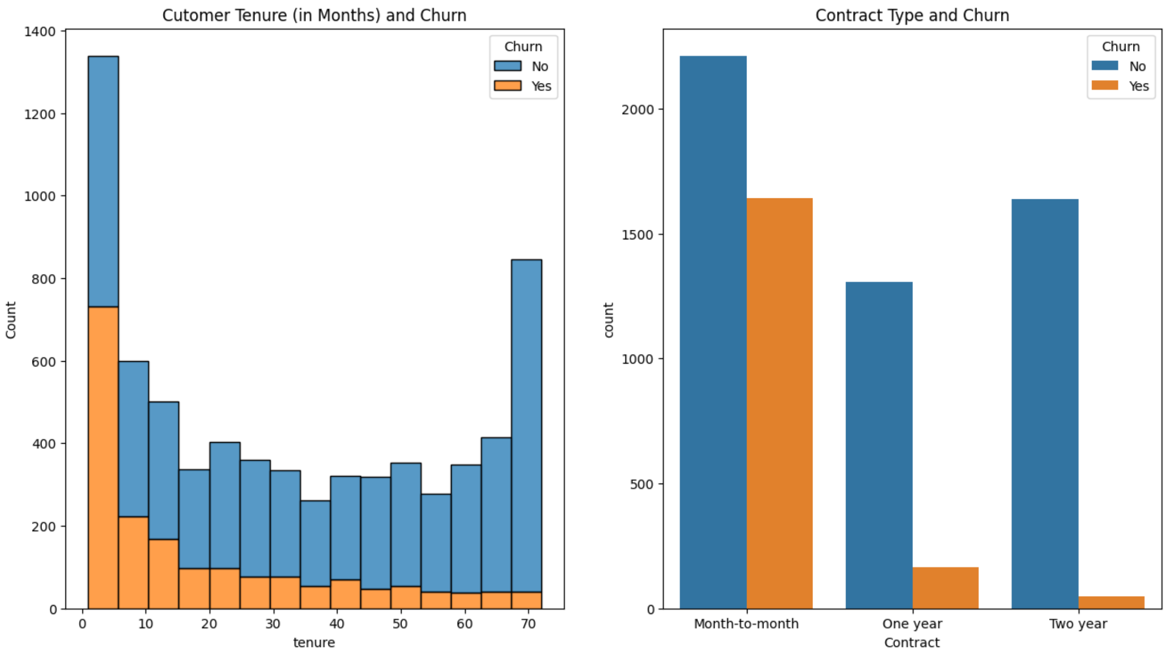
These graphs visualizes the relation between customer churn based on services opted by the customer. In the phone and internet service, there is no relation between churn and service opted, however the churn count is higher for the customers, who have taken multiple lines. Coming to other services, where customers who have not taken Online backup or Device Protection service has higher churn count, than those who have opted. Moreover, the customers with streaming services have lower churn count as compared to those who have not opted for it.

Therefore, certain services have relation with the customer churn, which are multiple lines, Online Backup, Device Protection, and Streaming Services.

Tenure/Contract and Churn

In [ ]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))
sns.histplot(x = 'tenure', data = df, ax= ax[0], hue = 'Churn', multiple = 'stac
sns.countplot(x = 'Contract', data = df, ax= ax[1], hue = 'Churn').set_title('Co
```

Out[ ]:
```
Text(0.5, 1.0, 'Contract Type and Churn')
```



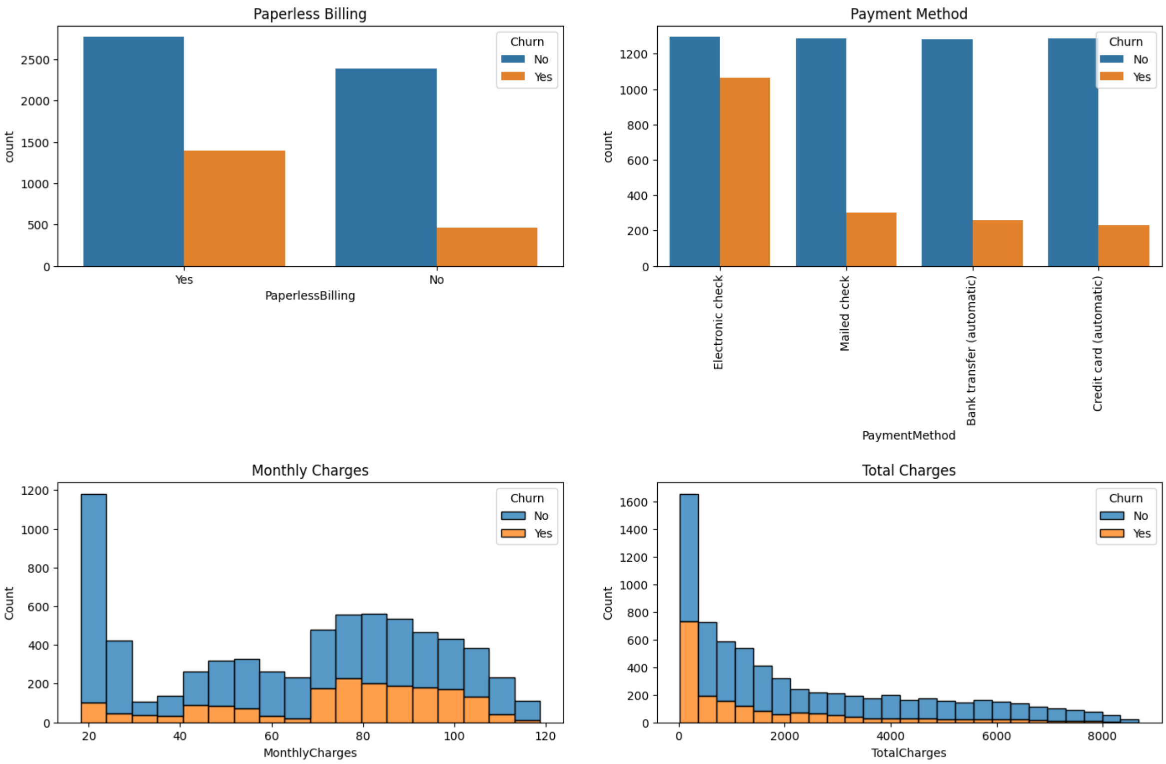
Looks like the customer tenure and contract has a inverse relation. The customers with shorter tenure or tenure less than 5 months have higher churn count. The churn count decreases with increase in tenure. Moreover, the customers with month-to-month contract have higher churn count as compared to those with one or two year contract which also proves that customer who have longer contract with the company have lower churn count.

Billing/Charges and Churn

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 10))
fig.tight_layout(pad=5.0)

#spacing between subplots
fig.subplots_adjust(hspace=0.9)
```

```
#papaerless billing
sns.countplot(x = df['PaperlessBilling'], ax=ax[0,0], hue = df['Churn']).set_tit

#Payment Method
sns.countplot(x = df['PaymentMethod'], ax=ax[0,1], hue = df['Churn']).set_title(
ax[0,1].xaxis.set_tick_params(rotation=90)

#Monthly Charges
sns.histplot(x = 'MonthlyCharges', data = df, ax = ax[1,0], hue = 'Churn', multi

#Total Charges
sns.histplot(x = 'TotalCharges', data = df, ax = ax[1,1], hue = 'Churn', multipl
```

Out[ ]: Text(0.5, 1.0, 'Total Charges')



The paperless billing and payment method have not significant relation with the customer churn. However, the montly and total charges do have a interesting relation with the customer churn. The customers with higher monthly charges have higher churn count, which is quite obvious. But, the customers with higher total charges have lower churn count, which is quite interesting. This could be possible, if the customer has a long tenure or uses alot of services. Therefore, the company should focus on lowering the monthly charges for the customers in order to reduce the churn count.

Data Preprocessing Part 2

Outlier Removal

In [ ]:
```
#Columns for outlier removal
cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
```

```
#Using IQR method to remove outliers
Q1 = df[cols].quantile(0.25)
Q3 = df[cols].quantile(0.75)
IQR = Q3 - Q1

#Removing the outliers
df = df[~((df[cols] < (Q1 - 1.5 * IQR)) |(df[cols] > (Q3 + 1.5 * IQR))).any(axis
```

Label Encoding

In [ ]:

```
from sklearn.preprocessing import LabelEncoder

#colums for label encoding
cols = df.columns[df.dtypes == 'object']

#Label encoder object
le = LabelEncoder()

#Label encoding the columns
for i in cols:
    le.fit(df[i])
    df[i] = le.transform(df[i])
    print(i, df[i].unique(), '\n')
```

```
gender [0 1]

Partner [1 0]

Dependents [0 1]

PhoneService [0 1]

MultipleLines [1 0 2]

InternetService [0 1 2]

OnlineSecurity [0 2 1]

OnlineBackup [2 0 1]

DeviceProtection [0 2 1]

TechSupport [0 2 1]

StreamingTV [0 2 1]

StreamingMovies [0 2 1]

Contract [0 1 2]

PaperlessBilling [1 0]

PaymentMethod [2 3 0 1]

Churn [0 1]
```

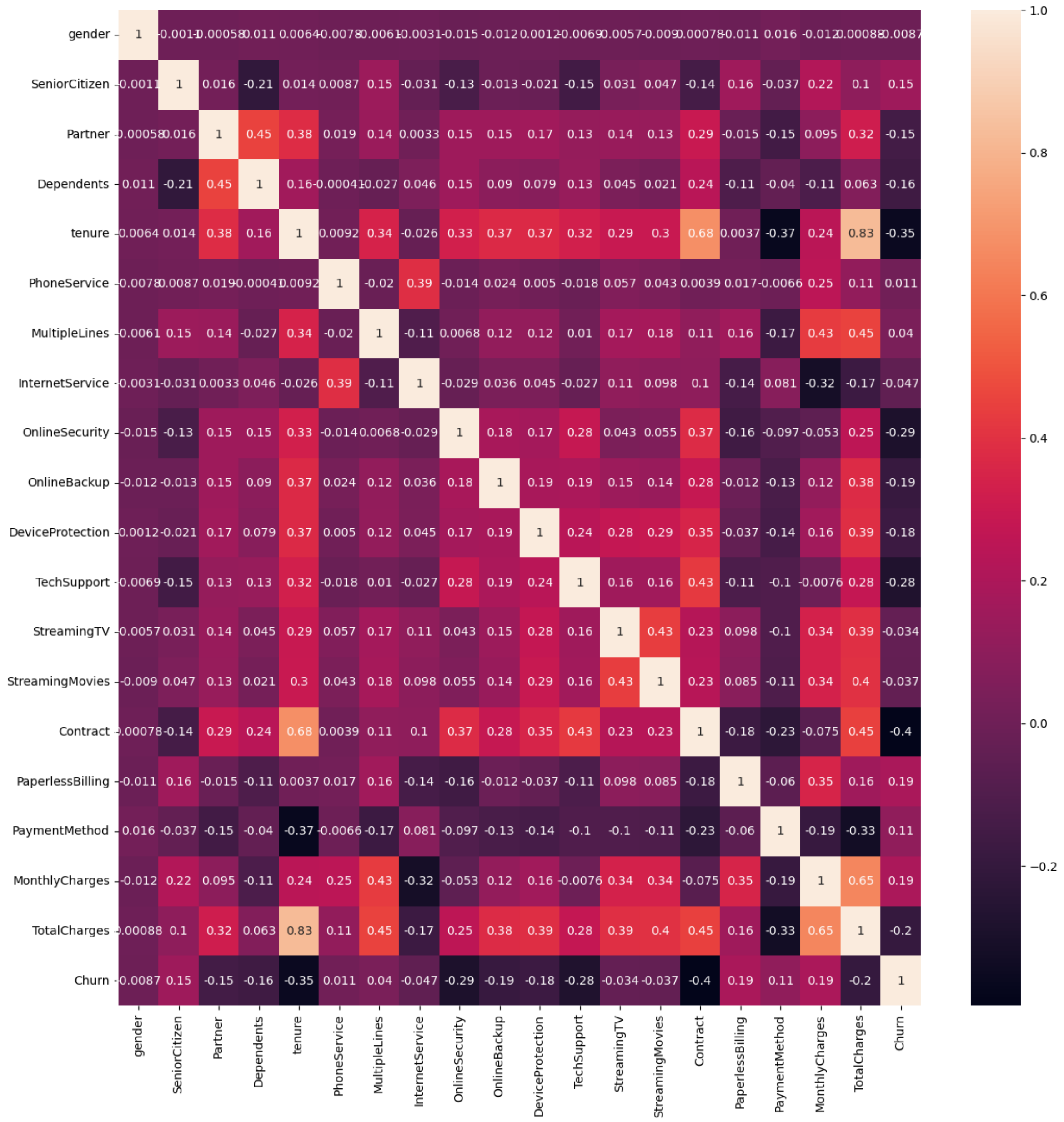
Feature Scaling

In [ ]:
```
from sklearn.preprocessing import StandardScaler

#Standardizing the data
sc = StandardScaler()
df[['tenure', 'MonthlyCharges', 'TotalCharges']] = sc.fit_transform(df[['tenure'
```

Correlation Matrix Heatmap

In [ ]:
```
plt.figure(figsize=(15, 15))
sns.heatmap(df.corr(), annot=True)
```

Out[ ]: <Axes: >



Train Test Split

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns='Churn'), df
```

Model Building

I will be using the following models to predict the customer churn:

1. Decision Tree Classifier
2. Random Forest Classifier
3. K Nearest Neighbors Classifier

Decision Tree Classifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

#Decision Tree Classifier Object
dtree  = DecisionTreeClassifier()
```

Hyperparameter Tuning using GridSearchCV

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#parameter grid
param_grid = {
    'max_depth': [2,4,6,8,10],
    'min_samples_leaf': [2,4,6,8,10],
    'min_samples_split': [2,4,6,8,10],
    'criterion': ['gini', 'entropy'],
    'random_state': [0,42]
}

#Grid Search Object with Decision Tree Classifier
grid_search = GridSearchCV(estimator = dtree, param_grid = param_grid, cv = 3, r

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 500 candidates, totalling 1500 fits
{'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 2, 'min_samples_split':
10, 'random_state': 42}
```

In [ ]:
```
#Decision Tree Classifier Object with best parameters
dtree  = DecisionTreeClassifier(criterion='gini', max_depth=6, min_samples_leaf=

#Fitting the data
dtree.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', dtree.score(X_train, y_train))

#Predicting the values
d_pred = dtree.predict(X_test)
```

```
Training Accuracy:  0.8072396576319544
```

Random Forest Classifier

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

#Random Forest Classifier Object
rfc = RandomForestClassifier()
```

Hyperparameter Tuning using GridSearchCV

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#parameter grid
param_grid = {
    'max_depth': [2,4,6,8,10],
    'min_samples_leaf': [2,4,6,8,10],
    'min_samples_split': [2,4,6,8,10],
    'criterion': ['gini', 'entropy'],
    'random_state': [0,42]
}

#Grid Search Object with Random Forest Classifier
grid_search = GridSearchCV(estimator = rfc, param_grid = param_grid, cv = 3, n_j

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 500 candidates, totalling 1500 fits
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 10, 'min_samples_sp
lit': 2, 'random_state': 42}
```

In [ ]:
```
#Random Forest Classifier Object with best parameters
rfc = RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf

#Fitting the data
rfc.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', rfc.score(X_train, y_train))

#Predicting the values
r_pred = rfc.predict(X_test)
```

```
Training Accuracy:  0.8309557774607703
```

K Nearest Neighbors Classifier

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

#KNN Classifier Object
knn = KNeighborsClassifier()
```

Hyperparameter Tuning using GridSearchCV

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#parameter grid
param_grid = {
    'n_neighbors': [2,4,6,8,10],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}

#Grid Search Object with KNN Classifier
grid_search = GridSearchCV(estimator = knn, param_grid = param_grid, cv = 3, n_j

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
{'algorithm': 'ball_tree', 'n_neighbors': 6, 'weights': 'uniform'}
```

In [ ]:
```
#KNN Classifier Object with best parameters
knn = KNeighborsClassifier(algorithm='ball_tree', n_neighbors=6, weights='unifor

#Fitting the data
knn.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', knn.score(X_train, y_train))

#Predicting the values
k_pred = knn.predict(X_test)
```

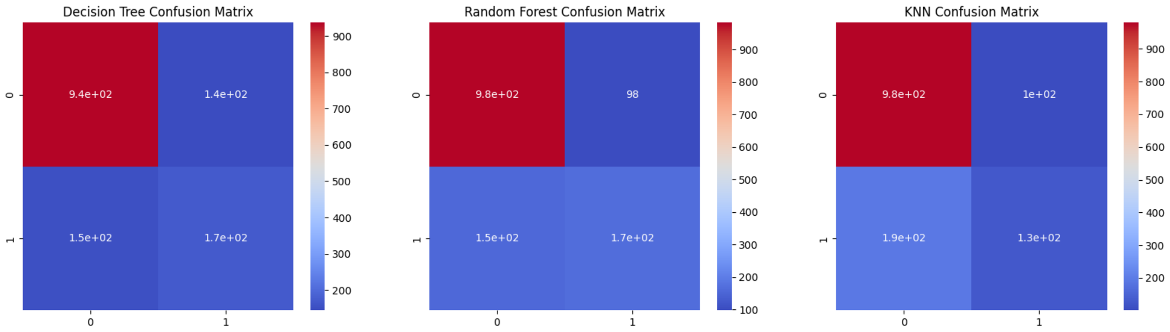
```
Training Accuracy:  0.8215049928673324
```

Model Evaluation

Confusion Matrix Heatmap

In [ ]:
```
from sklearn.metrics import confusion_matrix

fig, ax = plt.subplots(1, 3, figsize=(20, 5))

#Decision Tree Confusion Matrix
sns.heatmap(confusion_matrix(y_test, d_pred), annot=True, ax=ax[0], cmap='coolwa

#Random Forest Confusion Matrix
sns.heatmap(confusion_matrix(y_test, r_pred), annot=True, ax=ax[1], cmap='coolwa

#KNN Confusion Matrix
sns.heatmap(confusion_matrix(y_test, k_pred), annot=True, ax=ax[2], cmap='coolwa
```

Out[ ]:
```
Text(0.5, 1.0, 'KNN Confusion Matrix')
```

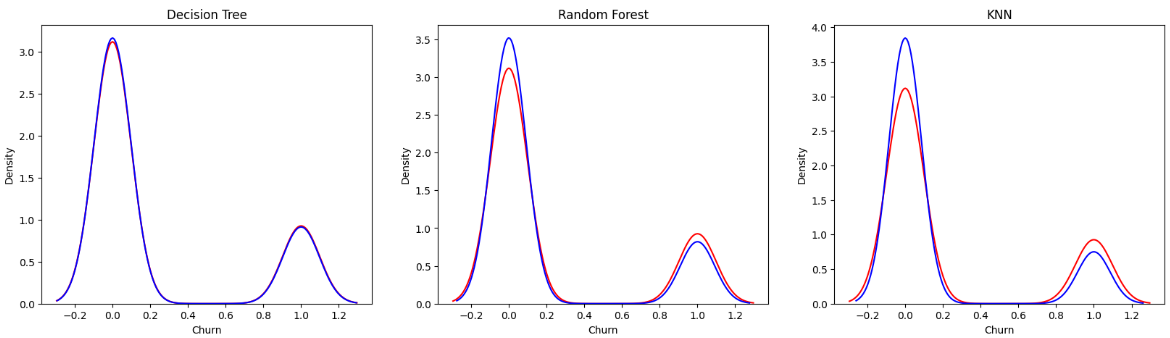


The confusion matrix heatmaps visulaizes the true positive and true negative results from the machine learning model. Here, in the above confusion matrix, when see that the Random Forest Classifier has the highest true positive and true negative results, with considerbly less false positive and false negative results. Therefore, the Random Forest Classifier is the best model for predicting the customer churn.

Distribution Plot

In [ ]:
```
fig, ax = plt.subplots(1, 3, figsize=(20, 5))

#Decision Tree
sns.distplot(y_test, hist=False, color="r", label="Actual Value", ax=ax[0]).set_
sns.distplot(d_pred, hist=False, color="b", label="Fitted Values" , ax=ax[0])

#Random Forest
sns.distplot(y_test, hist=False, color="r", label="Actual Value", ax=ax[1]).set_
sns.distplot(r_pred, hist=False, color="b", label="Fitted Values" , ax=ax[1])

#KNN
sns.distplot(y_test, hist=False, color="r", label="Actual Value", ax=ax[2]).set_
sns.distplot(k_pred, hist=False, color="b", label="Fitted Values" , ax=ax[2])
```

Out[ ]:
```
<Axes: title={'center': 'KNN'}, xlabel='Churn', ylabel='Density'>
```



Classification Report

In [ ]:
```
from sklearn.metrics import classification_report

print('Decision Tree Classification Report: \n', classification_report(y_test, d

print('Random Forest Classification Report: \n', classification_report(y_test, r

print('KNN Classification Report: \n', classification_report(y_test, k_pred))
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86      1081
           1       0.54      0.53      0.54       321

    accuracy                           0.79      1402
   macro avg       0.70      0.70      0.70      1402
weighted avg       0.79      0.79      0.79      1402

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.91      0.89      1081
           1       0.63      0.52      0.57       321

    accuracy                           0.82      1402
   macro avg       0.75      0.71      0.73      1402
weighted avg       0.81      0.82      0.81      1402

KNN Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.91      0.87      1081
           1       0.56      0.40      0.47       321

    accuracy                           0.79      1402
   macro avg       0.70      0.65      0.67      1402
weighted avg       0.77      0.79      0.78      1402
```

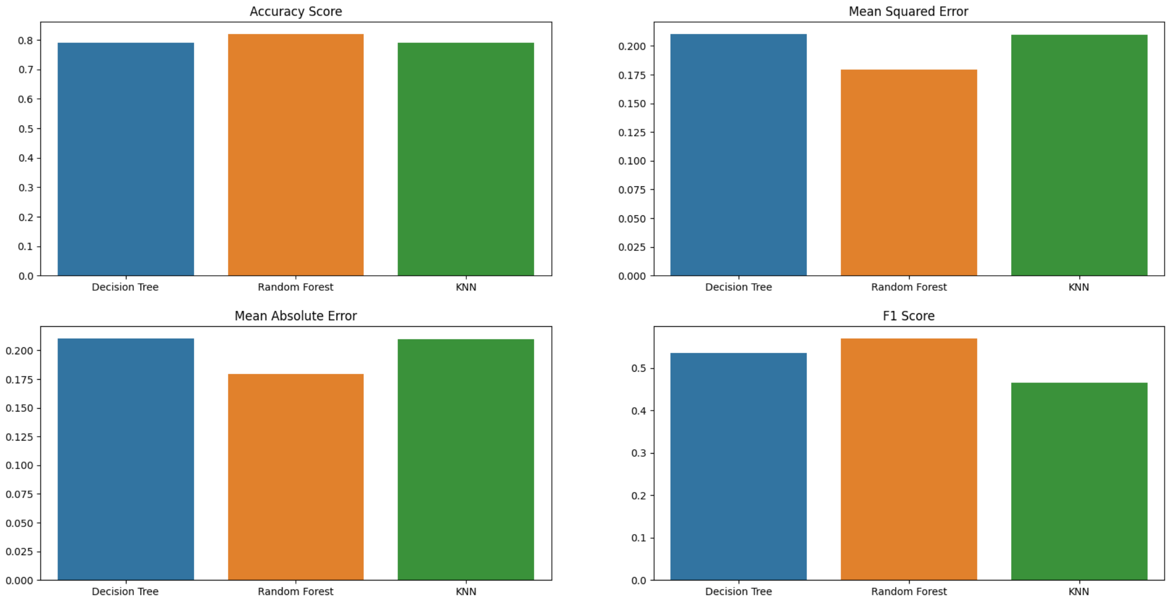
Model Metrics

In [ ]:

```
from sklearn.metrics import accuracy_score, mean_squared_error, mean_absolute_er

#Bar plots
fig, ax = plt.subplots(2,2, figsize=(20, 10))

#Accuracy Score
sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y = [accuracy_score(y

#Mean Squared Error
sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y = [mean_squared_err

#Mean Absolute Error
sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y = [mean_absolute_er

#F1 Score
sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y = [f1_score(y_test,
```

Out[ ]:

```
Text(0.5, 1.0, 'F1 Score')
```



These graphs compares the models based on the metrics such as accuracy score, mean squared error, mean absolute error and f1 score. The Random Forest Classifier has the highest accuracy score and F1 Score, and lowest mean squared error, mean absolute error. Therefore, the Random Forest Classifier is a good fit for predicting the customer churn.

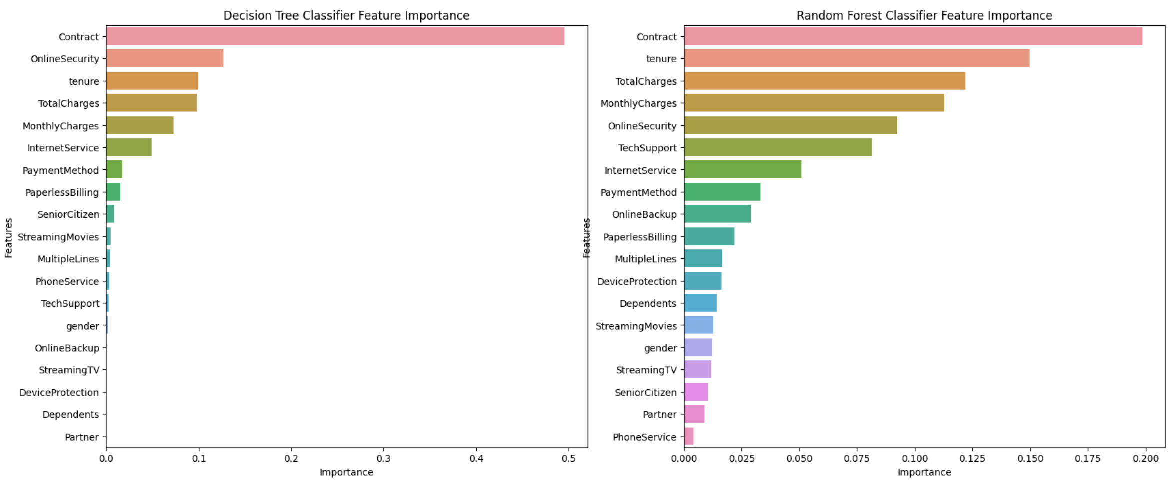
Feature Importance

In [ ]:

```
fig, ax = plt.subplots(1, 2, figsize=(20, 8))
# Decision Tree Classifier Feature Importance
feature_df = pd.DataFrame({'Features': X_train.columns, 'Importance': dtree.feat
feature_df.sort_values('Importance', ascending=False, inplace=True)
sns.barplot(x = 'Importance', y = 'Features', data = feature_df, ax=ax[0]).set_t

# Random Forest Classifier Feature Importance
feature_df = pd.DataFrame({'Features': X_train.columns, 'Importance': rfc.featur
feature_df.sort_values('Importance', ascending=False, inplace=True)
sns.barplot(x = 'Importance', y = 'Features', data = feature_df, ax=ax[1]).set_t
```

Out[ ]:

```
Text(0.5, 1.0, 'Random Forest Classifier Feature Importance')
```



From both the models, it is clear that the tenure, monthly charges, and total charges are the most important features for predicting the customer churn. Therefore, the company should focus on these features to reduce the customer churn.

Conclusion

From the exploratory data analysis, I came to know that, the senior citizens have lower churn count whereas the customers who are single or don't have dependents ahve higher churn count. In addition to that, customers are more satified with the streaming services than other services such as Online backup and Device protection, which has resulted in lower churn count in customer with streaing services than the other services.

The tenure have an inverse relation with churn count, where customer with tenure shorter than 5 months have higher churn count. Moreover, the customers with month-to-month contract have higher churn count as compared to those with one or two year contract which also proves that customer who have longer contract with the company have lower churn count.

It has been observed that the customers with higher monthly charges and lower total charges have higher churn count. Therefore, the company should focus on lowering the monthly charges for the customers in order to reduce the churn count. From the feature importance, it is clear that the tenure, contract, monthly charges, and total charges are the most important features for predicting the customer churn. Therefore, the company should focus on these features to reduce the customer churn.

Coming to the machine learning models, I have used three models - Decision Tree Classifier, Random Forest Classifier, and K Nearest Neighbors Classifier. The Random Forest Classifier has the highest accuracy i.e. 82% and F1 Score, and lowest mean squared error, mean absolute error. Therefore, the Random Forest Classifier is a good fit for predicting the customer churn.