Telecom Customer Churn Prediction

◇ Imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import  missingno as msno
import seaborn as sns
```

◇ Load Dataset

```
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head()

   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
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

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
No
3  No phone service             DSL            Yes  ...
Yes
4                No     Fiber optic             No  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0          No          No              No  Month-to-month
Yes
1          No          No              No        One year
No
```

```
2          No          No               No  Month-to-month
Yes
3         Yes          No               No        One year
No
4          No          No               No  Month-to-month
Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

Getting information about the dataset

```
df.shape
```

```
(7043, 21)
```

```
df.dtypes
```

```
customerID           object
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
```

```
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
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
```

◇ Preprocessing

Handling Missing Values

Removing irrelevant columns

```
df.drop(columns='customerID', inplace=True)
```

Check to see if there are any duplicates

```
df.duplicated().sum()
```

```
22
```

```
df.drop_duplicates(inplace=True)
```

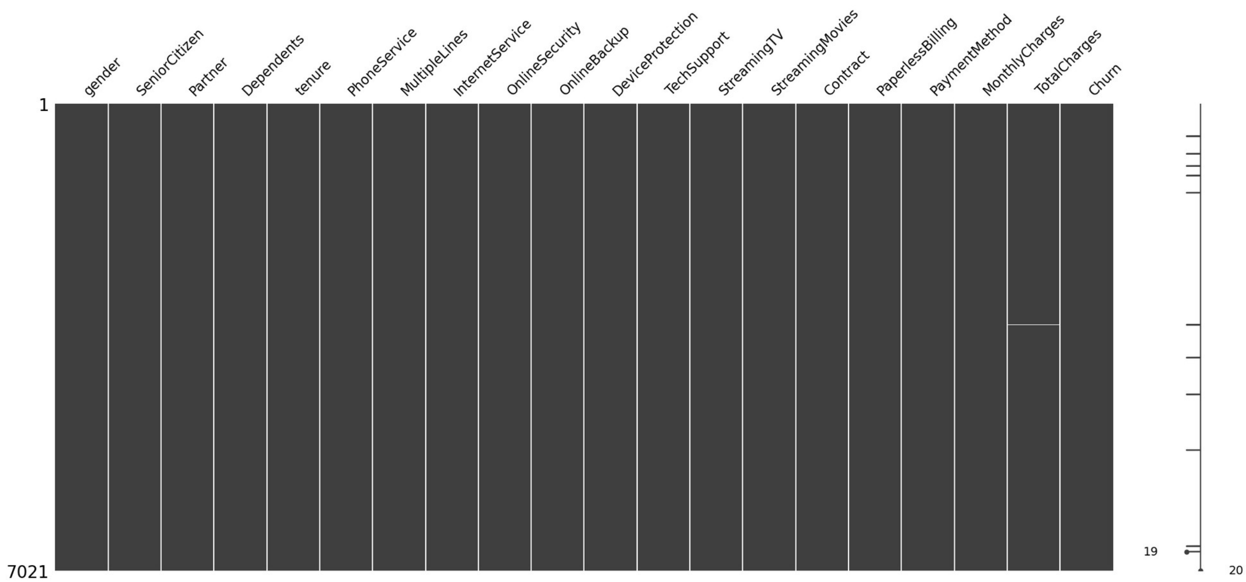
setting errors to 'coerce', so that if pandas encounters a value that it cannot convert to a numeric datatype, it will replace that value with NaN.

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

Check to see if there are any missing values

```
msno.matrix(df)
```

```
<Axes: >
```




```
df.isnull().sum()

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

df.dropna(inplace=True)
```

Check to see what unique values each feature contains

```
for col in df.columns:
    print(col, df[col].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']
```

```
df.dtypes
```

```
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        float64
Churn                object
dtype: object
```

◇ Exploratory Data Analysis (EDA)

Single Variable Analysis

Plotting count plot for features 'Gender', 'SeniorCitizen', 'Partner' and 'Dependents' to get a good understaning of Customer Demography

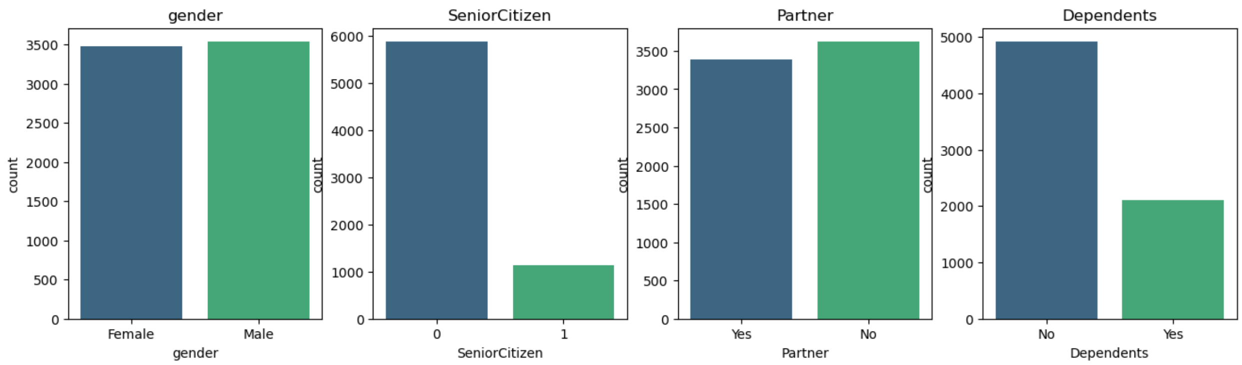
Based on the plotted features we can indicate the following factors:

- The proportion of male and female users are about the same
- The majority of users are not seniors
- The majority of users have no dependents

```
cols = ['gender','SeniorCitizen', 'Partner', 'Dependents']

fig, axes = plt.subplots(1, 4, figsize=(16, 4))

for i, col in enumerate(cols):
    ax = axes[i]
    sns.countplot(x=col, data=df, palette='viridis', ax=ax)
    ax.set_title(col)
```

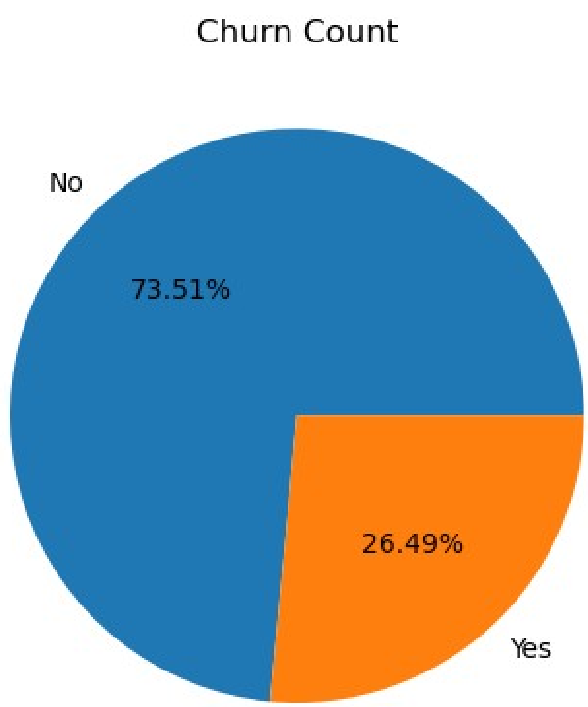


Plotting pie chart for Churn distribution

Only 26.49% of customers of the telecom co. have churned from the company which proves the company has a high rate of meeting its users desires

```
plt.pie(df['Churn'].value_counts(), labels=df['Churn'].unique(),
autopct = '%1.2f%%')
plt.title('Churn Count')
```

```
Text(0.5, 1.0, 'Churn Count')
```



Plotting count plot for features'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV' and 'StreamingMovies'

According to the plotted features:

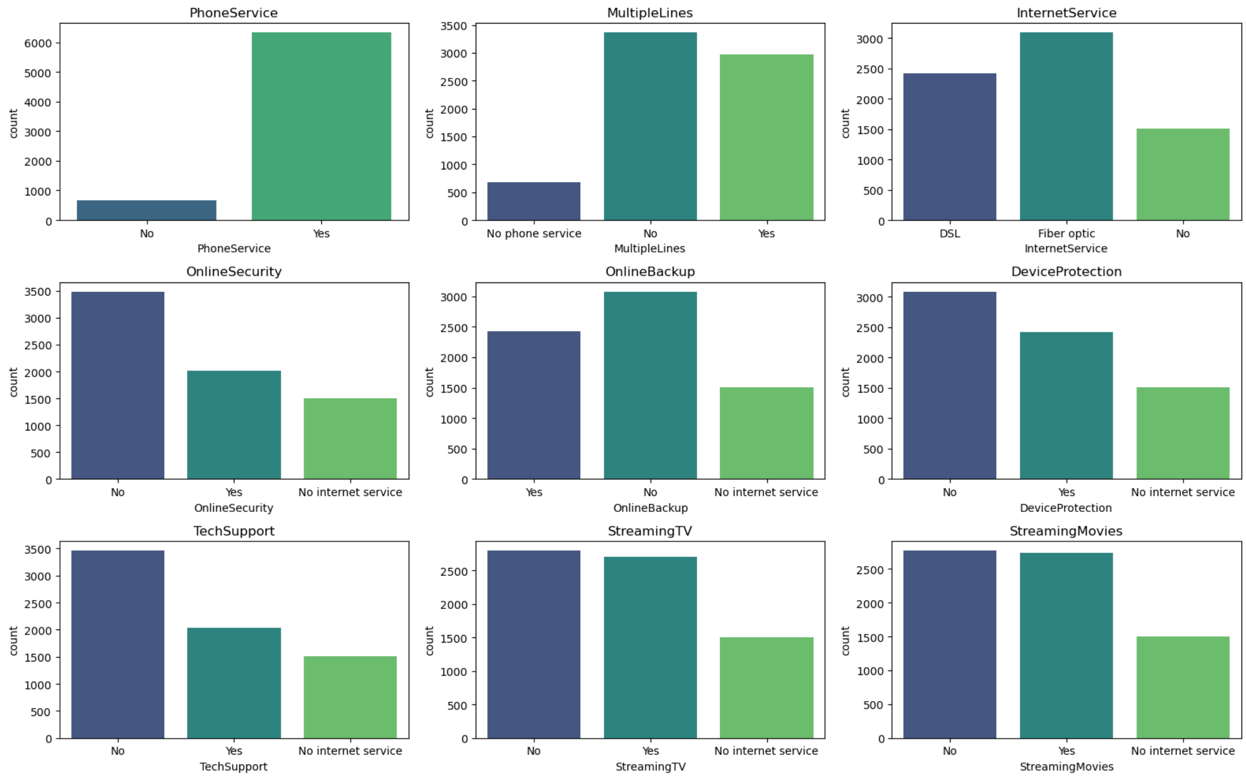
- The majority of users recieve PhoneService from the company.
- The internet service recieved by customers are mostly fiber optic but sill a large number would rather DSL.
- The majority of users do not use services regarding device protections, online security and tech support which higlights the customers concern regarding their device safety and data protection.
- Streaming services such as streaming TVs and Movies are the most popular services among users with more than 2500 users.

```
cols = ['PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']

fig, axes = plt.subplots(3, 3, figsize=(16, 10))

for i, col in enumerate(cols):
    ax = axes[i//3, i%3]
    sns.countplot(x=col, data=df, palette='viridis', ax=ax)
    ax.set_title(col)

plt.tight_layout()
plt.show()
```



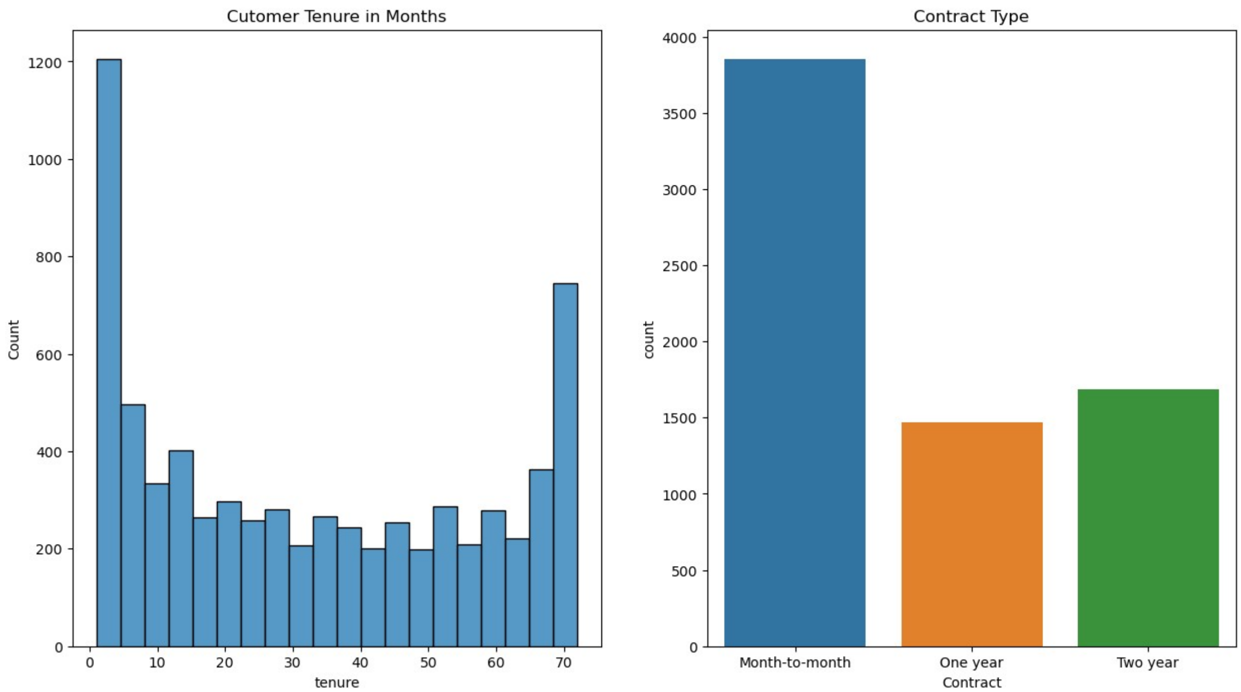
Contract Type vs Tenure

The following plots considering two features of 'Contract Type' and 'Customer Tenure' suggest that:

- The company seems to be attracting a lot of new customers with contract type of 'Month to Month'. Also a large number of customers have a tenure of 70 months, which might indicate that the company has a lot of long-term customers.

```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))
sns.histplot(x = 'tenure', bins = 20, data = df, ax =
ax[0]).set_title('Cutomer Tenure in Months')
sns.countplot(x = 'Contract', data = df, ax =
ax[1]).set_title('Contract Type')
```

```
Text(0.5, 1.0, 'Contract Type')
```



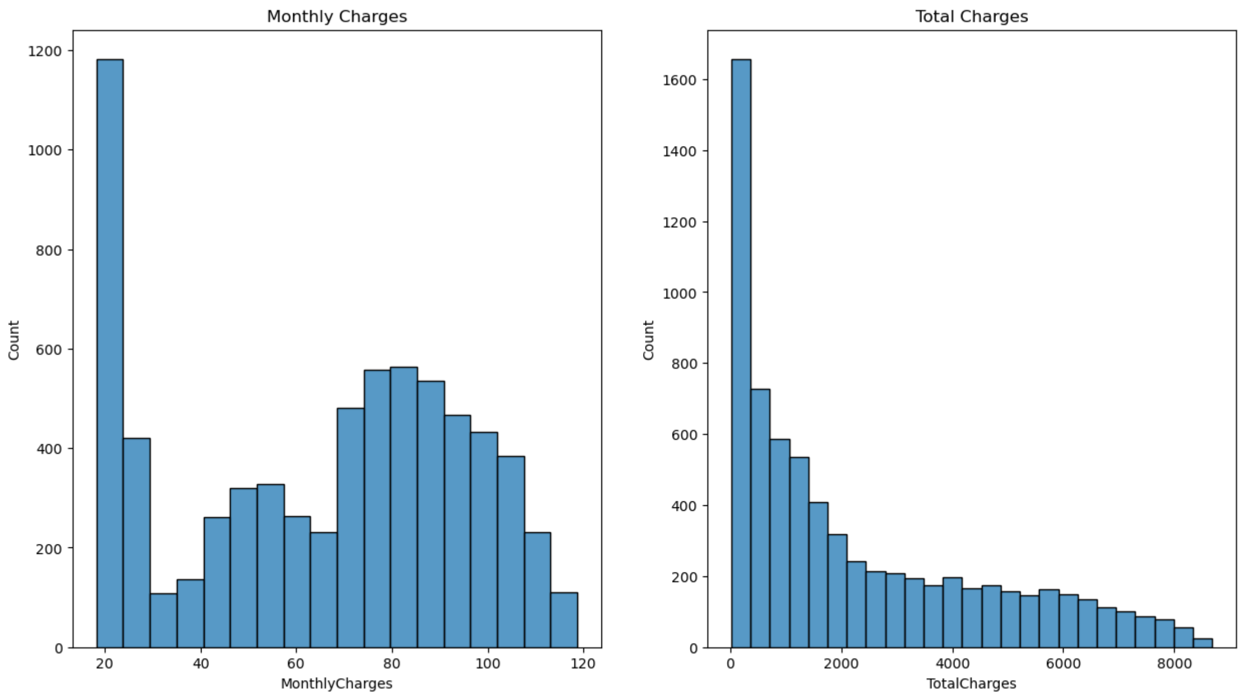
Billing vs Charges

Majority of customers pay almost 20 dollars for the monthly services and are also having total charges of less than 200 dollars. However, there are considerable number of customers having monthly charges between 70 to 100 dollars and total charges of 200-800 dollars. Many customers also have a total bill of more than 4000. This could be possible, if the customer has a long tenure or uses alot of services.

```python
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

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

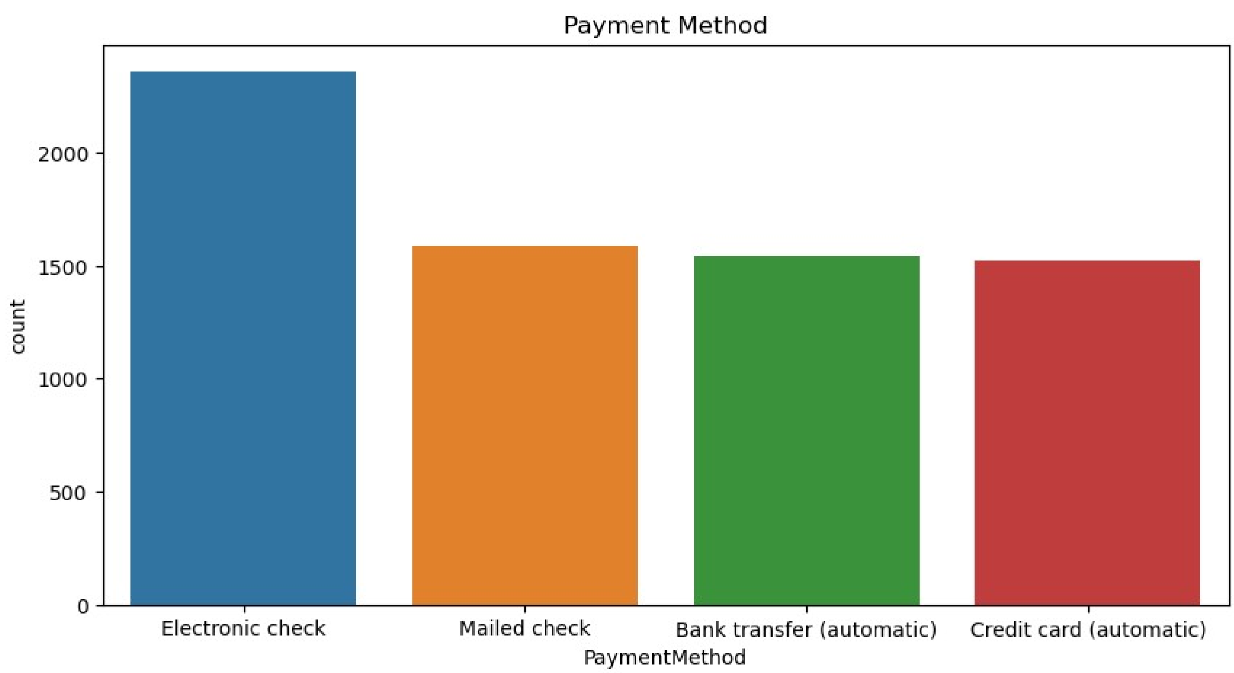
```
Text(0.5, 1.0, 'Total Charges')
```



As shown in the plotted features, the majority of users prefer Electronic checks as their payment method. The following most preferred payment methods are mailed check, bank transfer and credit card with roughly 1500 users.

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.countplot(x = df['PaymentMethod']).set_title('Payment Method')
```

```
Text(0.5, 1.0, 'Payment Method')
```



Two Variable Analysis

Customer Demogrpahics and Churn

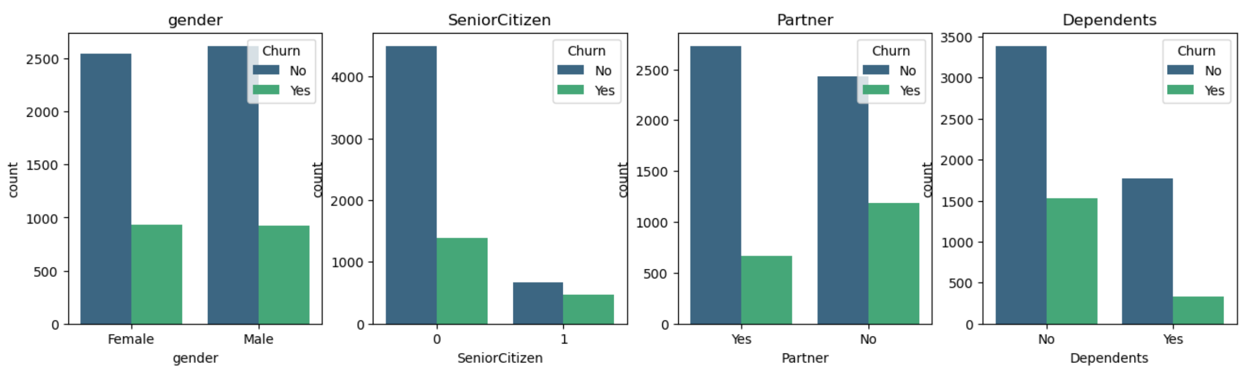
The following graphs indicate that:

- It appears that the churn rates for females and males are quite similar, indicating that gender may not be a significant factor in customer churn.
- It is obviously seen that senior citizens have a higher churn rate. This could suggest that senior citizens might be finding the service less satisfactory or perhaps more difficult to use.
- Customers without partners have a higher churn rate. This could be because individuals with partners may have more stable usage patterns or benefit more from certain services or discounts.
- Customers without dependents tend to churn more frequently. This could be due to the flexibility and less commitment required by individuals without dependents

```
cols = ['gender','SeniorCitizen', 'Partner', 'Dependents']

fig, axes = plt.subplots(1, 4, figsize=(16, 4))

for i, col in enumerate(cols):
    ax = axes[i]
    sns.countplot(x=col, data=df,hue='Churn', palette='viridis',
ax=ax)
    ax.set_title(col)
```



Services Churn

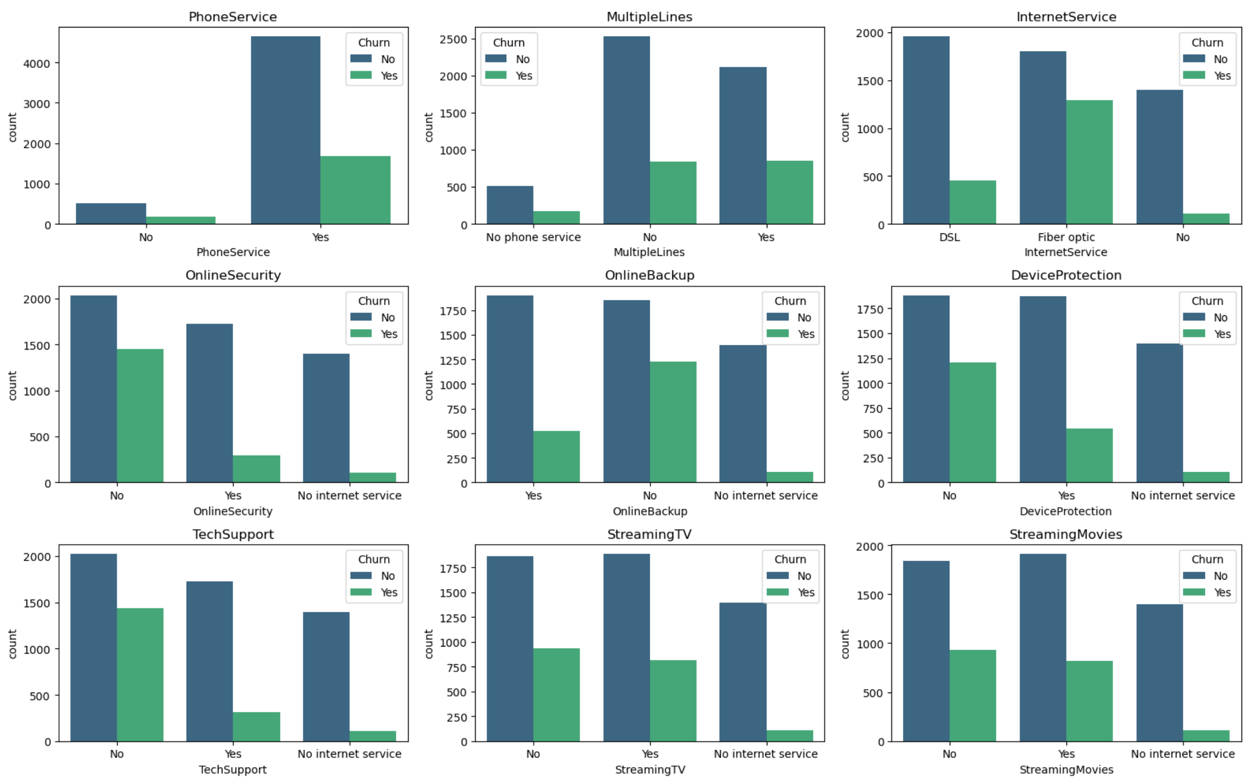
- It seems like customers who recieve internet service especially in fiber optic field tend to churn more frequently.
- The customers who have no access to security and data protection programms offered by the company seem to churn more frequently compared to those who use streaming services.

```
cols = ['PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']

fig, axes = plt.subplots(3, 3, figsize=(16, 10))

for i, col in enumerate(cols):
    ax = axes[i//3, i%3]
    sns.countplot(x=col, data=df, hue='Churn', palette='viridis',
ax=ax)
    ax.set_title(col)

plt.tight_layout()
plt.show()
```



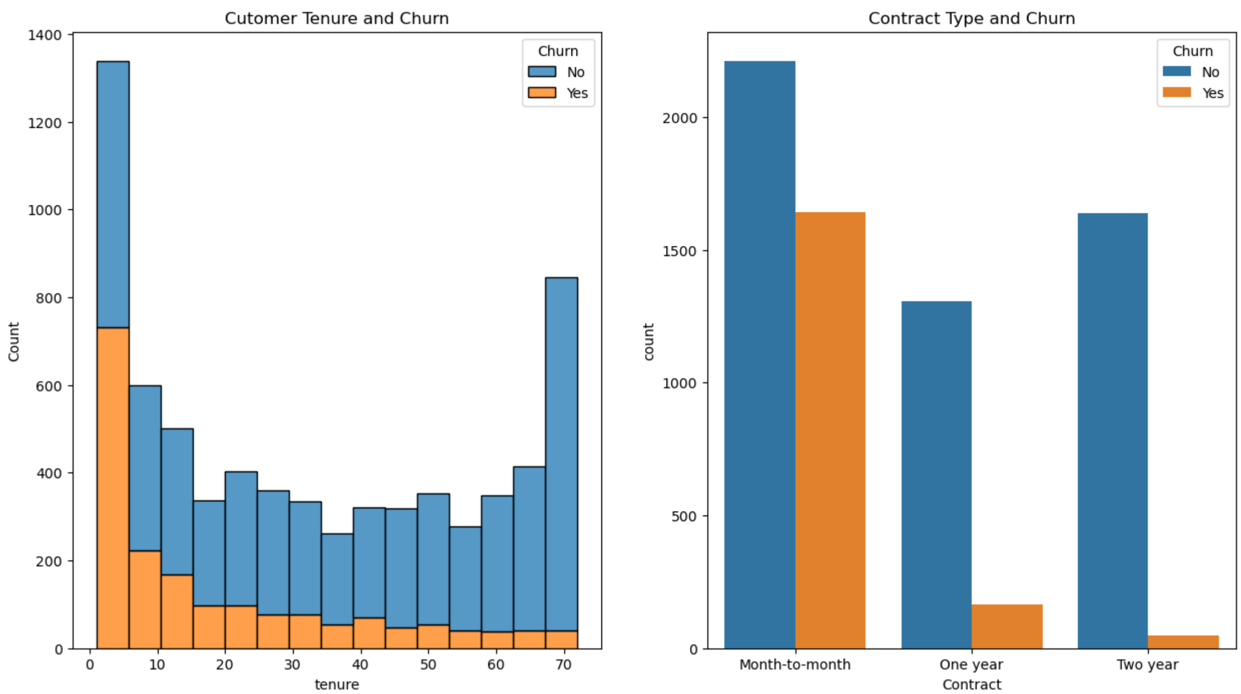
Tenur/Contract and Churn

- Customers with shorter tenures (less than a year) are more likely to churn. This could suggest that newer customers are less satisfied or more volatile.
- Customers on a month-to-month contract have a higher churn rate compared to those on one-year or two-year contracts. This could be because longer-term contracts provide a sense of stability and commitment that reduces the likelihood of churn

```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))
sns.histplot(x='tenure', data=df, ax= ax[0], hue='Churn',
multiple='stack').set_title('Cutomer Tenure and Churn')
```

```
sns.countplot(x='Contract', data=df, ax= ax[1],
hue='Churn').set_title('Contract Type and Churn')

Text(0.5, 1.0, 'Contract Type and Churn')
```



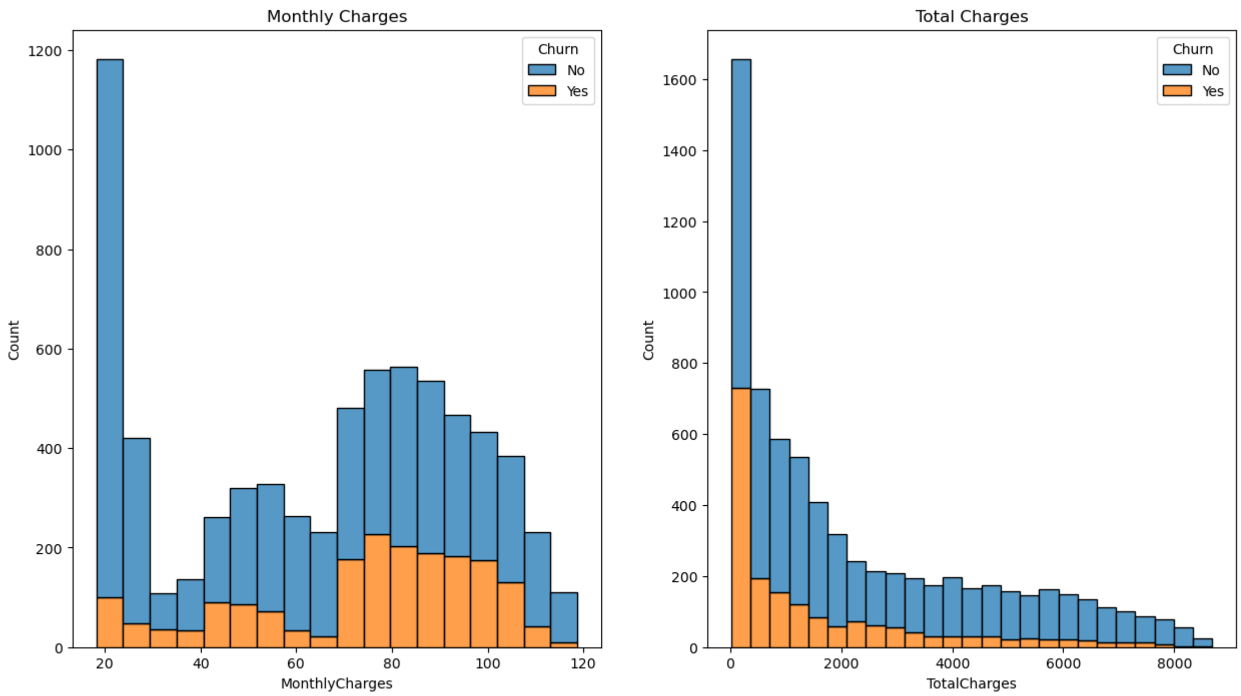
Billing/Monthly Charges and Churn

- It appears that a large number of customers with low monthly charges did not churn. However, for mid-range monthly charges, there is a significant number of customers who churned. This could suggest that customers are more likely to churn when their monthly charges are neither too low nor too high.
- There is a decreasing number of customers as total charges increase. Very few customers with high total charges churned. This could indicate that customers who have been with the company longer and therefore have higher total charges are less likely to churn.

```
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

sns.histplot(x='MonthlyCharges', data=df, ax=ax[0], hue='Churn',
multiple= 'stack').set_title('Monthly Charges')
sns.histplot(x='TotalCharges', data=df, ax=ax[1], hue='Churn',
multiple= 'stack').set_title('Total Charges')

Text(0.5, 1.0, 'Total Charges')
```



◇ Data Preprocessing part 2

Label Encoding

```
from sklearn.preprocessing import LabelEncoder

cols = df.columns[df.dtypes == 'object']

le = LabelEncoder()

for i in cols:
    le.fit(df[i])
    df[i] = le.transform(df[i])

df.dtypes

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        float64
Churn                 int32
dtype: object
```

Feature Scaling

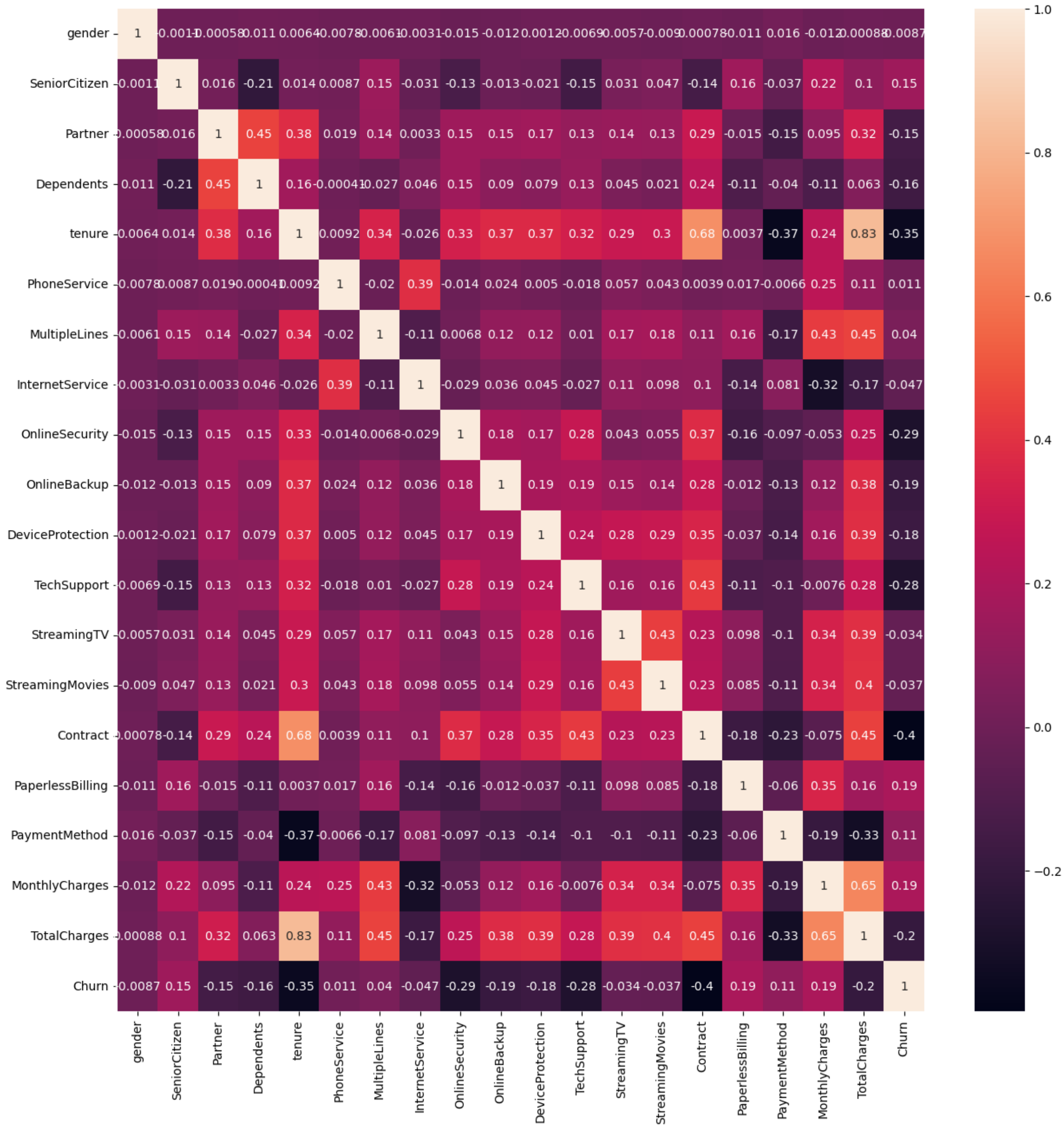
```
from sklearn.preprocessing import StandardScaler

cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
sc = StandardScaler()
df[['tenure', 'MonthlyCharges', 'TotalCharges']] =
sc.fit_transform(df[['tenure', 'MonthlyCharges', 'TotalCharges']])
```

Confusion Matrix

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

<Axes: >
```



Train Test Split

```
from sklearn.model_selection import train_test_split

y = df['Churn']
X = df.drop(columns='Churn')

x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

◇ Model Training

```
from sklearn.tree import *
from sklearn.ensemble import *
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Decision Tree

```
dtree = DecisionTreeClassifier(max_depth=4, random_state=42,
ccp_alpha=0.001)

dtree.fit(x_train, y_train)

DecisionTreeClassifier(ccp_alpha=0.001, max_depth=4, random_state=42)

d_pred = dtree.predict(x_test)
print('Training Accuracy: ', dtree.score(x_train, y_train))

Training Accuracy:  0.7810271041369472
```

Random Forest

```
rfc = RandomForestClassifier(max_depth=4, random_state=42,
ccp_alpha=0.001)

rfc.fit(x_train, y_train)

RandomForestClassifier(ccp_alpha=0.001, max_depth=4, random_state=42)

r_pred = rfc.predict(x_test)
print('Training Accuracy: ', rfc.score(x_train, y_train))

Training Accuracy:  0.793509272467903
```

KNN

```
knn = KNeighborsClassifier(algorithm='ball_tree', n_neighbors=6,
weights='uniform')

knn.fit(x_train, y_train)

KNeighborsClassifier(algorithm='ball_tree', n_neighbors=6)

k_pred = knn.predict(x_test)
print('Training Accuracy: ', knn.score(x_train, y_train))

Training Accuracy:  0.8215049928673324
```

Ensemble Training

```
model = BaggingClassifier(estimator=dtree, n_estimators=50,
max_samples=0.7, oob_score=True, random_state=42)

model.fit(x_train, y_train)

BaggingClassifier(estimator=DecisionTreeClassifier(ccp_alpha=0.001,
max_depth=4,
                                                   random_state=42),
                  max_samples=0.7, n_estimators=50, oob_score=True,
                  random_state=42)

e_pred = model.predict(x_test)
print('Training Accuracy: ', knn.score(x_train, y_train))

Training Accuracy:  0.8215049928673324
```

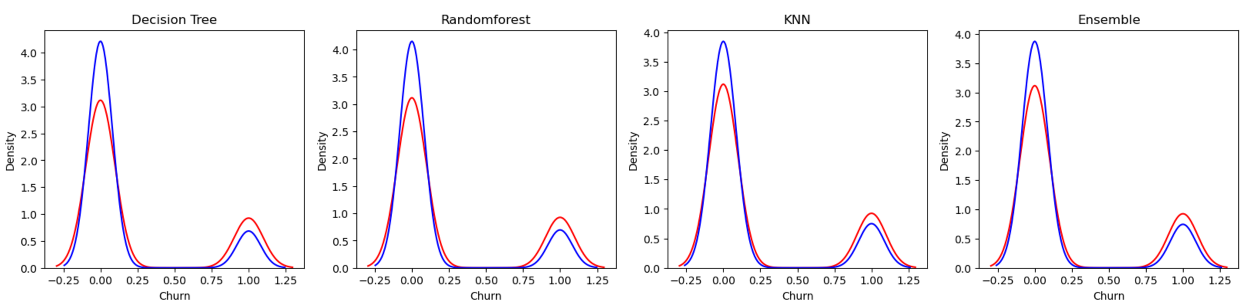
◇ Evaluation

```
from sklearn.metrics import accuracy_score, mean_squared_error,
mean_absolute_error, f1_score
```

Distribution Plot

```
models = {'Decision Tree' : d_pred, 'Randomforest': r_pred, 'KNN':
k_pred, 'Ensemble': e_pred}
fig, ax = plt.subplots(1, 4, figsize=(20, 4))

for i, model_name in enumerate(models):
    sns.kdeplot(y_test, color='r', label="Actual Value",
ax=ax[i]).set_title(model_name)
    sns.kdeplot(models[model_name], color='b', label="Fitted Values" ,
ax=ax[i])
```



Metrics

```
fig, ax = plt.subplots(2,2, figsize=(20, 10))

sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y =
[accuracy_score(y_test, d_pred), accuracy_score(y_test, r_pred),
```

```
accuracy_score(y_test, k_pred)], ax=ax[0,0]).set_title('Accuracy
Score')

sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y =
[mean_squared_error(y_test, d_pred), mean_squared_error(y_test,
r_pred), mean_squared_error(y_test, k_pred)],
ax=ax[0,1]).set_title('Mean Squared Error')

sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y =
[mean_absolute_error(y_test, d_pred), mean_absolute_error(y_test,
r_pred), mean_absolute_error(y_test, k_pred)],
ax=ax[1,0]).set_title('Mean Absolute Error')

sns.barplot(x = ['Decision Tree', 'Random Forest', 'KNN'], y =
[f1_score(y_test, d_pred), f1_score(y_test, r_pred), f1_score(y_test,
k_pred)], ax=ax[1,1]).set_title('F1 Score')
```

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