```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings("ignore")
```

```
data=pd.read_csv("/content/sample_data/customer_churn.csv")
```

```
data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Dev
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	

5 rows × 21 columns

```
data.shape
```

```
(7043, 21)
```

```
data.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
```

```
data.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

```
data.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
```

```
data.columns
```

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

```
data[data['TotalCharges'].isin([''])]
data['TotalCharges'].value_counts() ## to check white spaces
```

```
TotalCharges
           11
20.2       11
19.75       9
20.05       8
19.9        8
           ..
6849.4      1
692.35      1
130.15      1
3211.9      1
6844.5      1
Name: count, Length: 6531, dtype: int64
```

```
for i in data.columns:
  data[data[i].isin([''])]

      ###for loop to check white spaces, ? etc
```

```
data['TotalCharges'].value_counts()  ## to check white spaces
```

```
TotalCharges
           11
20.2       11
```

```
19.75      9
20.05      8
19.9       8
           ..
6849.4     1
692.35     1
130.15     1
3211.9     1
6844.5     1
Name: count, Length: 6531, dtype: int64
```

Double-click (or enter) to edit

```
data['TotalCharges']=(data['TotalCharges'].replace(' ',np.nan))  ## replace white spaces with nan
```

```
data.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        11
Churn                0
dtype: int64
```

```
data.TotalCharges.dtypes
```

```
dtype('float64')
```

```
data['TotalCharges']=data['TotalCharges'].astype(float)
```

```
data.TotalCharges.dtypes
```

```
dtype('float64')
```

```
data=data.dropna()
```

```
data.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
```

```
data=data.drop(['customerID'],axis=1)
```

```
data.columns
```

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

```
col_list =[]
for i in data.columns:
  if data[i].dtypes=='object':
    col_list.append(i)
```

```
col_list
```

```
['gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'Churn']
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
for i in col_list:
  data[i]=le.fit_transform(data[i])
```

```
data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	Device
0	0	0	1	0	1	0	1	0	0	2	
1	1	0	0	0	34	1	0	0	2	0	
2	1	0	0	0	2	1	0	0	2	2	
3	1	0	0	0	45	0	1	0	2	0	
4	0	0	0	0	2	1	0	1	0	0	

```
x=data.iloc[:,:-1]
```

```
x.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	Device
0	0	0	1	0	1	0	1	0	0	2	
1	1	0	0	0	34	1	0	0	2	0	
2	1	0	0	0	2	1	0	0	2	2	
3	1	0	0	0	45	0	1	0	2	0	
4	0	0	0	0	2	1	0	1	0	0	

```
y=data['Churn']
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=999)
```

```
rf=RandomForestClassifier(n_estimators=100)
```

```
rf.fit(xtrain,ytrain)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
ypred=rf.predict(xtest)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
accuracy_score(ytest,ypred)
```

```
0.8123667377398721
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid={
    "n_estimators":[100,200,300],
    'max_depth': [None,5,10],
    "min_samples_split": [2,5,10],
    "min_samples_leaf":[1,2,4]
}
```

```
rfc=RandomForestClassifier(random_state=42)
```

```
gs=GridSearchCV(estimator=rfc,param_grid=param_grid,cv=2)
```

```
gs.fit(xtrain,ytrain)
```

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

Start coding or generate with AI.

```
print("best hyperparameter", gs.best_params_)
best_dt=gs.best_estimator_
print(best_dt)
```

```
best hyperparameter {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 200}
RandomForestClassifier(max_depth=10, min_samples_leaf=4, n_estimators=200,
                       random_state=42)
```

```
prediction=best_dt.predict(xtest)
```

```
accuracy_score(prediction,ytest)
```

```
0.8187633262260128
```

```
from sklearn.linear_model import LogisticRegression
```

```
lr=LogisticRegression()
```

```
lr.fit(xtrain,ytrain)
```

```
LogisticRegression
LogisticRegression()
```

```
pred1=lr.predict(xtest)
```

```
accuracy_score(pred1,ytest)
```

```
0.8187633262260128
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
```

```
dt.fit(xtrain,ytrain)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred_dt=dt.predict(xtest)
```

```
accuracy_score(pred_dt,ytest)
```

```
0.7327647476901208
```

Start coding or generate with AI.