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

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder  # only for 2 classes if more than 2 classes then use OneHotEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score


df = pd.read_csv("Telco-Customer-Churn.csv")  # Ensure file is downloaded from the given link
print("Dataset Shape:", df.shape)
print(df.head())
```

```
Dataset Shape: (7043, 21)
   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]
```

```
df.drop('customerID', axis=1, inplace=True)
```

```
df
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No
...	...	...	...	...	...	...	...	...	...	...
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No

7043 rows × 20 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce') #coerce is convert data in NaN if not convert in numeric
```

```
df
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No
...	...	...	...	...	...	...	...	...	...	...
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No

7043 rows × 20 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df = df.dropna()
```

```
df
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No
...	...	...	...	...	...	...	...	...	...	...
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No

7032 rows × 20 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
for col in df.columns:
    if df[col].dtype == 'object':
```

```
    if df[col].nunique() == 2:
        df[col] = LabelEncoder().fit_transform(df[col])
    else:
        df = pd.get_dummies(df, columns=[col])
```

```
/tmp/ipython-input-2470158310.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = LabelEncoder().fit_transform(df[col])
/tmp/ipython-input-2470158310.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = LabelEncoder().fit_transform(df[col])
/tmp/ipython-input-2470158310.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = LabelEncoder().fit_transform(df[col])
/tmp/ipython-input-2470158310.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = LabelEncoder().fit_transform(df[col])
```

```
df
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...
0	0	0	1	0	1	0	1	29.85	29.85	0	...
1	1	0	0	0	34	1	0	56.95	1889.50	0	...
2	1	0	0	0	2	1	1	53.85	108.15	1	...
3	1	0	0	0	45	0	0	42.30	1840.75	0	...
4	0	0	0	0	2	1	1	70.70	151.65	1	...
...	...	...	...	...	...	...	...	...	...	...	...
7038	1	0	1	1	24	1	1	84.80	1990.50	0	...
7039	0	0	1	1	72	1	1	103.20	7362.90	0	...
7040	0	0	1	1	11	0	1	29.60	346.45	0	...
7041	1	1	1	0	4	1	1	74.40	306.60	1	...
7042	1	0	0	0	66	1	1	105.65	6844.50	0	...

7032 rows × 41 columns

```
X = df[[
    'Contract_Month-to-month',
    'InternetService_Fiber optic',
    'TotalCharges',
    'tenure',
    'MonthlyCharges',
    'TechSupport_No',
    'PaymentMethod_Electronic check',
    'Contract_One year',
    'StreamingMovies_Yes',
    'MultipleLines_No phone service'
]]
```

```
X
```

	Contract_Month-to-month	InternetService_Fiber optic	TotalCharges	tenure	MonthlyCharges	TechSupport_No	PaymentMethod_Electronic check	Contra
0	True	False	29.85	1	29.85	True	True	
1	False	False	1889.50	34	56.95	True	False	
2	True	False	108.15	2	53.85	True	False	
3	False	False	1840.75	45	42.30	False	False	
4	True	True	151.65	2	70.70	True	True	
...	...	...	...	...	...	...	...	
7038	False	False	1990.50	24	84.80	False	False	
7039	False	True	7362.90	72	103.20	True	False	
7040	True	False	346.45	11	29.60	True	True	
7041	True	True	306.60	4	74.40	True	False	
7042	False	True	6844.50	66	105.65	False	False	

7032 rows × 10 columns

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
y = df['Churn']
```

```
y
```

	Churn
0	0
1	0
2	1
3	0
4	1
...	...
7038	0
7039	0
7040	0
7041	1
7042	0

7032 rows × 1 columns

dtype: int64

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
dt_model = DecisionTreeClassifier(random_state=42, max_depth=5)
dt_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

```
y_pred_dt = dt_model.predict(X_test)
```

```
print("\nDecision Tree Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("ROC-AUC:", roc_auc_score(y_test, dt_model.predict_proba(X_test)[:, 1]))
print("\nClassification Report:\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Performance:
Accuracy: 0.7903340440653873
ROC-AUC: 0.828923596191975

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

```
              0       0.86      0.86      0.86      1033
              1       0.61      0.61      0.61       374

       accuracy                           0.79      1407
      macro avg       0.73      0.73      0.73      1407
   weighted avg       0.79      0.79      0.79      1407
```

```
dt_importance = pd.Series(dt_model.feature_importances_, index=X.columns).sort_values(ascending=False)
```

Random Forest

```
rf_model = RandomForestClassifier(n_estimators=100, max_depth=8, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=8, random_state=42)
```

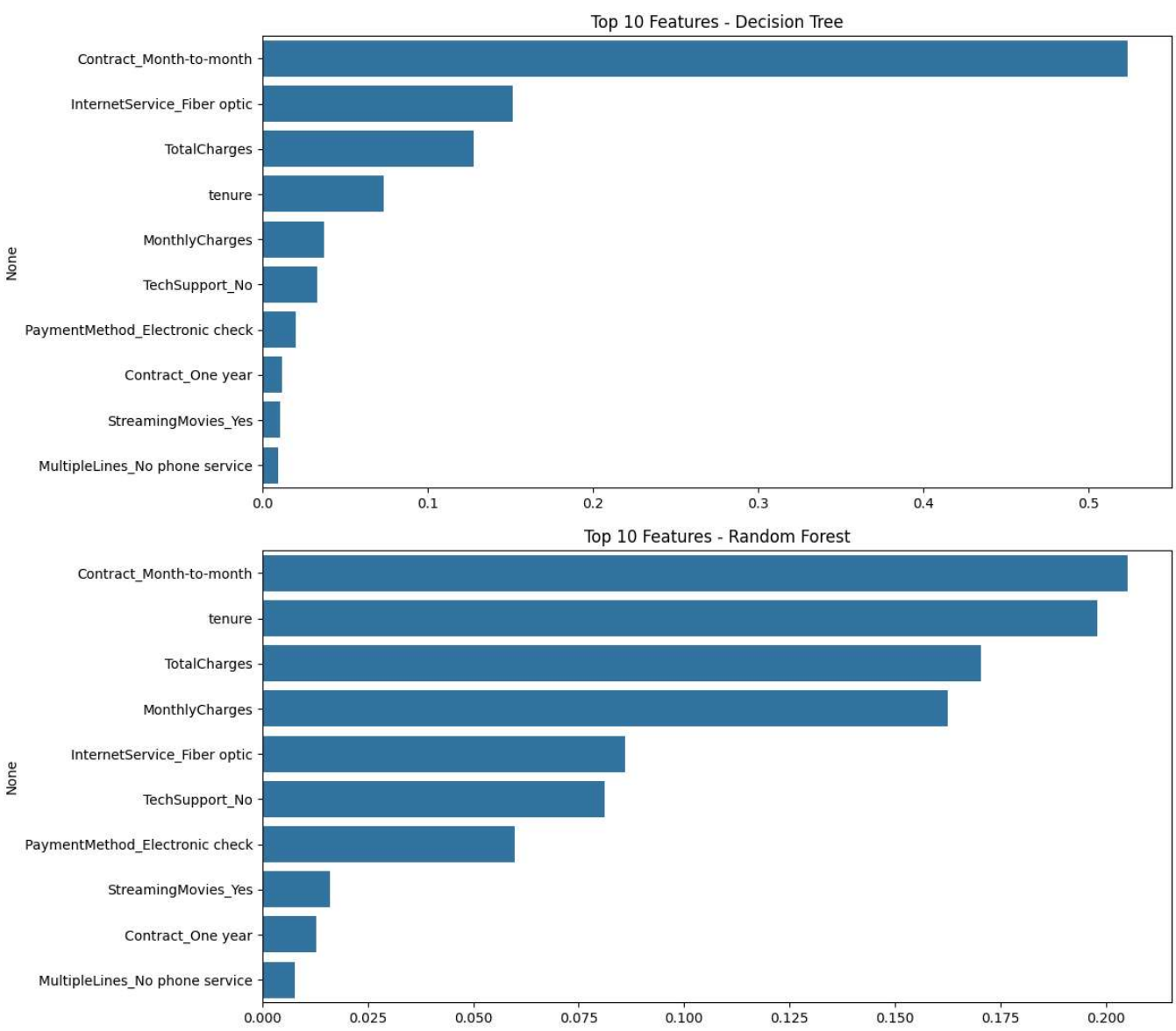
```
y_pred_rf = rf_model.predict(X_test)
```

```
print("\nRandom Forest Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("ROC-AUC:", roc_auc_score(y_test, rf_model.predict_proba(X_test)[:, 1]))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
```

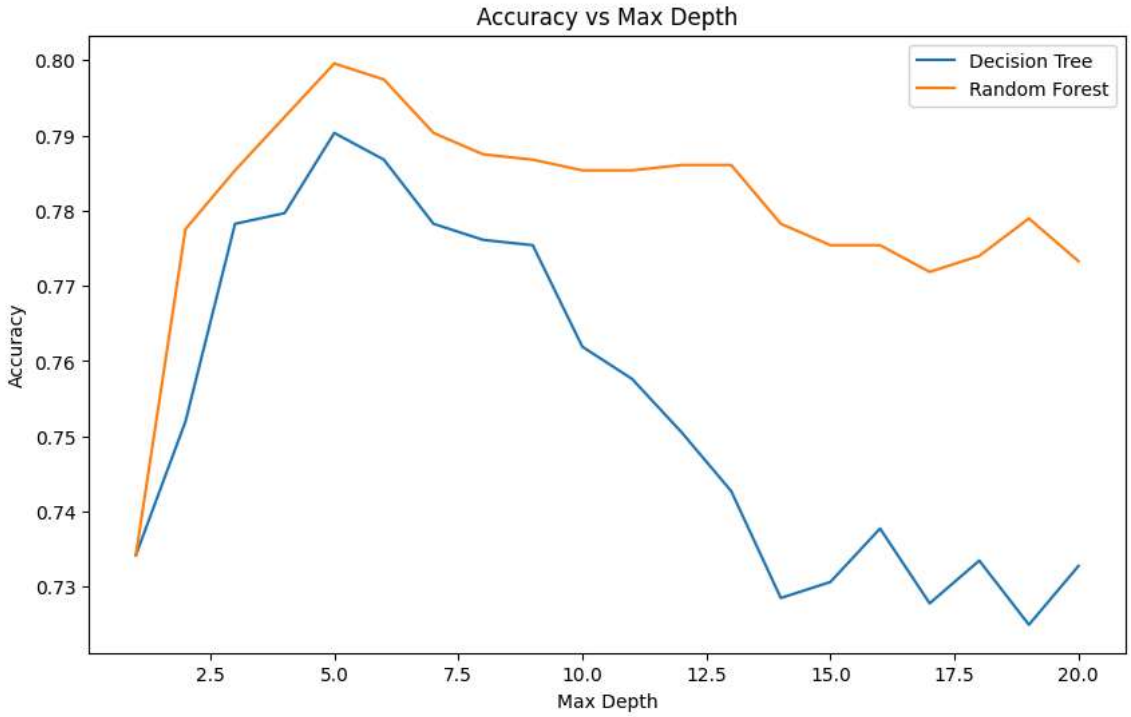
```
Random Forest Performance:
Accuracy: 0.7874911158493249
ROC-AUC: 0.8350192316652084

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1033
           1       0.63      0.50      0.56       374

    accuracy                           0.79      1407
   macro avg       0.73      0.70      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

```
rf_importance = pd.Series(rf_model.feature_importances_, index=X.columns).sort_values(ascending=False)
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x=dt_importance[:10].values, y=dt_importance[:10].index)
plt.title("Top 10 Features - Decision Tree")
plt.show()

plt.figure(figsize=(12, 6))
sns.barplot(x=rf_importance[:10].values, y=rf_importance[:10].index)
plt.title("Top 10 Features - Random Forest")
plt.show()
```




```
depths = range(1, 21)
dt_accuracies = []
rf_accuracies = []


for d in depths:
    dt_temp = DecisionTreeClassifier(max_depth=d, random_state=42)
    dt_temp.fit(X_train, y_train)
    dt_accuracies.append(accuracy_score(y_test, dt_temp.predict(X_test)))

    rf_temp = RandomForestClassifier(n_estimators=100, max_depth=d, random_state=42)
    rf_temp.fit(X_train, y_train)
    rf_accuracies.append(accuracy_score(y_test, rf_temp.predict(X_test)))


plt.figure(figsize=(10, 6))
plt.plot(depths, dt_accuracies, label='Decision Tree')
plt.plot(depths, rf_accuracies, label='Random Forest')
plt.xlabel("Max Depth")
plt.ylabel("Accuracy")
plt.title("Accuracy vs Max Depth")
plt.legend()
plt.show()
```




```
estimators = [10, 50, 100, 200, 300]
rf_scores = []

for n in estimators:
    rf_temp = RandomForestClassifier(n_estimators=n, max_depth=8, random_state=42)
    rf_temp.fit(X_train, y_train)
    rf_scores.append(accuracy_score(y_test, rf_temp.predict(X_test)))

plt.figure(figsize=(10, 6))
plt.plot(estimators, rf_scores, marker='o')
plt.xlabel("Number of Trees")
plt.ylabel("Accuracy")
plt.title("Random Forest Accuracy vs Number of Trees")
plt.show()
```