CUSTOMER CHURN PREDICTION

Step 1: Import Libraries

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
```

Step 2: Load Dataset

```
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")  # Download
from Kaggle
print("Data loaded successfully!")
```

```
Data loaded successfully!
```

```
df.head(3)
```

```
   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

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
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

      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 rows x 21 columns]
```

Step 3: Basic Exploration

```
print(df.shape)
print(df.columns)
print(df["Churn"].value_counts())
```

```
(7043, 21)
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner',
'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

Step 4: Handle Missing Values

```
print(df.isnull().sum())
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"],
errors='coerce')  # Convert to numeric
df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```

```
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

C:\Users\Dell\AppData\Local\Temp\ipykernel_13684\4084226285.py:3: 
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```

Step 5: Drop Irrelevant Columns

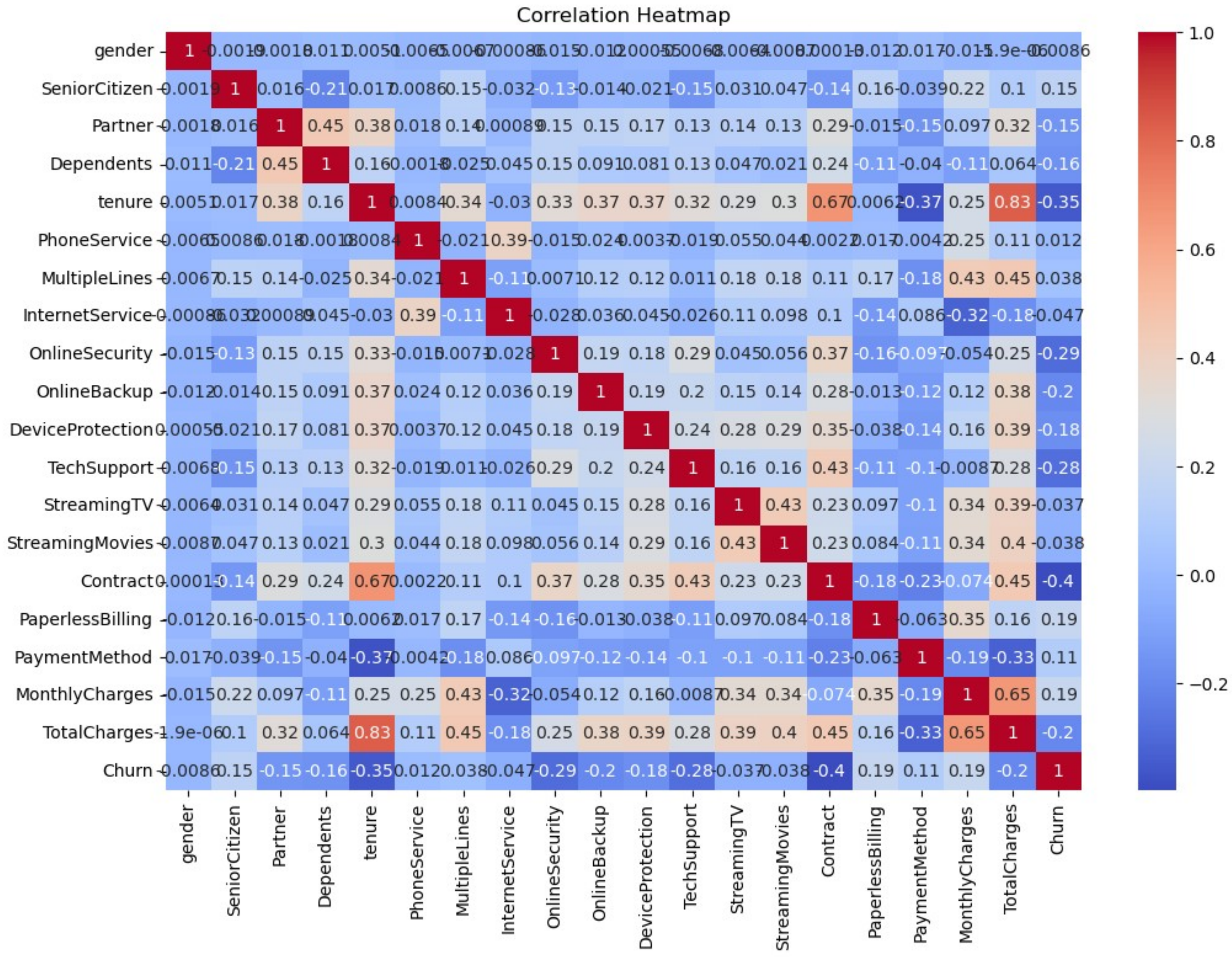
```
df.drop(['customerID'], axis=1, inplace=True)
```

Step 6: Convert Categorical Variables to Numeric

```
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Step 7: Correlation Matrix

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



Step 8: Feature and Target Split

```
X = df.drop("Churn", axis=1)
y = df["Churn"]
```

Step 9: Train-Test Split

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

Step 10: Train Model

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

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

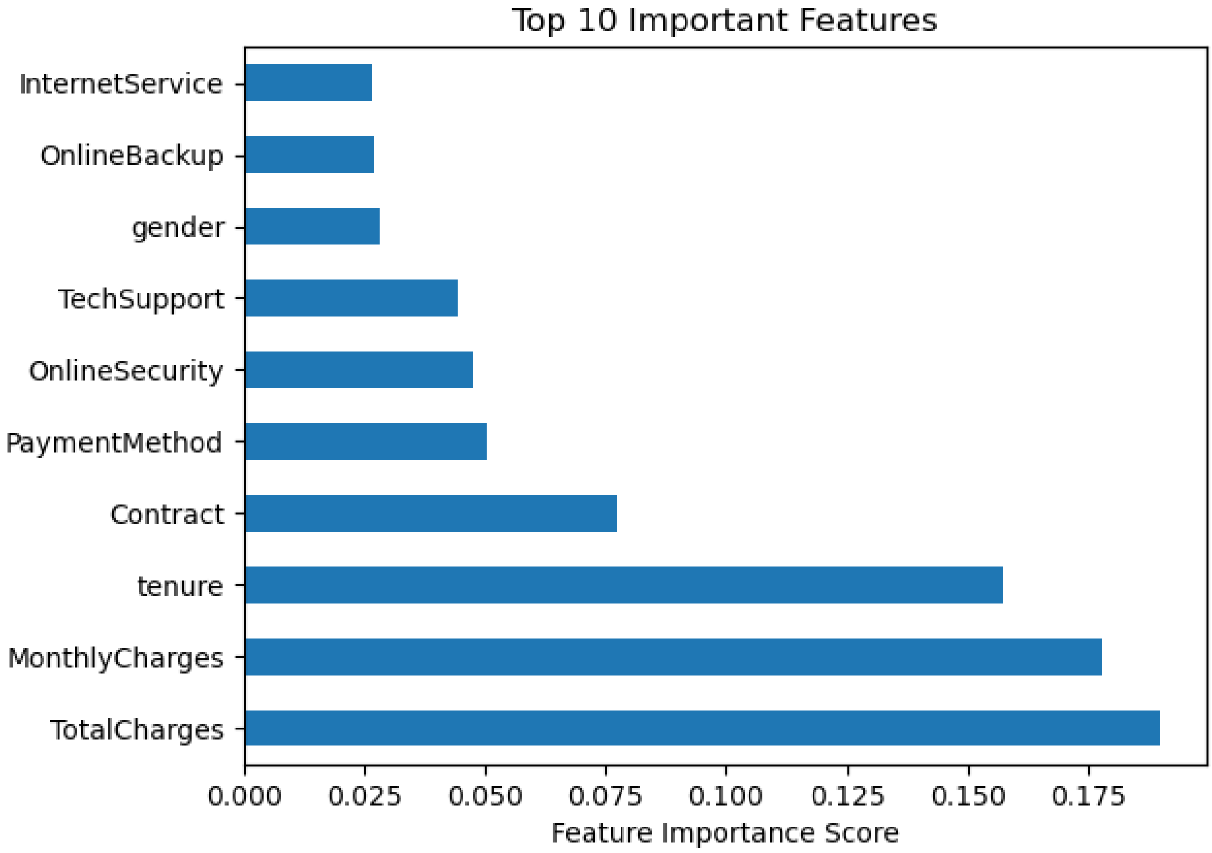
Step 11: Evaluate Model

```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,
y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.7955997161107168
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.66      0.47      0.55       373

    accuracy                           0.80      1409
   macro avg       0.74      0.69      0.71      1409
weighted avg       0.78      0.80      0.78      1409

Confusion Matrix:
 [[945  91]
 [197 176]]
```

Step 12: Feature Importance

```
importances = pd.Series(model.feature_importances_, index=X.columns)
importances.nlargest(10).plot(kind='barh', title="Top 10 Important
Features")
plt.xlabel("Feature Importance Score")
plt.show()
```


Top 10 Important Features
InternetService
OnlineBackup
gender
TechSupport
OnlineSecurity
PaymentMethod
Contract
tenure
MonthlyCharges
TotalCharges
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Feature Importance Score