Task 04 – Customer Churn Prediction

In [1]:
```
# Step 1: Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_s
import joblib
```

```
C:\Users\pramo\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: Us
erWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version
'1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
# Step 2: Load dataset
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head()
```

Out[2]:

eLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingM
phone service	DSL	No	...	No	No	No	
No	DSL	Yes	...	Yes	No	No	
No	DSL	Yes	...	No	No	No	
phone service	DSL	Yes	...	Yes	Yes	No	
No	Fiber optic	No	...	No	No	No	

In [3]:
```
# Step 3: Basic info
df.info()
df['Churn'].value_counts()
```

```
<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
```

Out[3]:
```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

In [4]:
```
# Step 4: Data cleaning
df.columns = df.columns.str.strip()  # Remove extra spaces from column names
if 'customerID' in df.columns:
    df.drop(['customerID'], axis=1, inplace=True)

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

In [5]:
```
# Step 5: Encoding categorical variables

from sklearn.preprocessing import LabelEncoder

# Clean column names
df.columns = df.columns.str.strip()

# Encode binary columns
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Ch
for col in binary_cols:
    if col in df.columns:
        df[col] = LabelEncoder().fit_transform(df[col])

# One-hot encode multi-class columns (only if they exist)
multi_cols = [
    'gender', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBack
    'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
    'Contract', 'PaymentMethod'
]
df = pd.get_dummies(df, columns=[col for col in multi_cols if col in df.columns
```

In [6]:
```
# Step 6: Feature scaling
scaler = StandardScaler()
df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['te
```

In [7]:
```
# Step 7: Train/test split
X = df.drop('Churn', axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

In [8]:
```
# Step 8: Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[8]:
```
RandomForestClassifier(random_state=42)
```

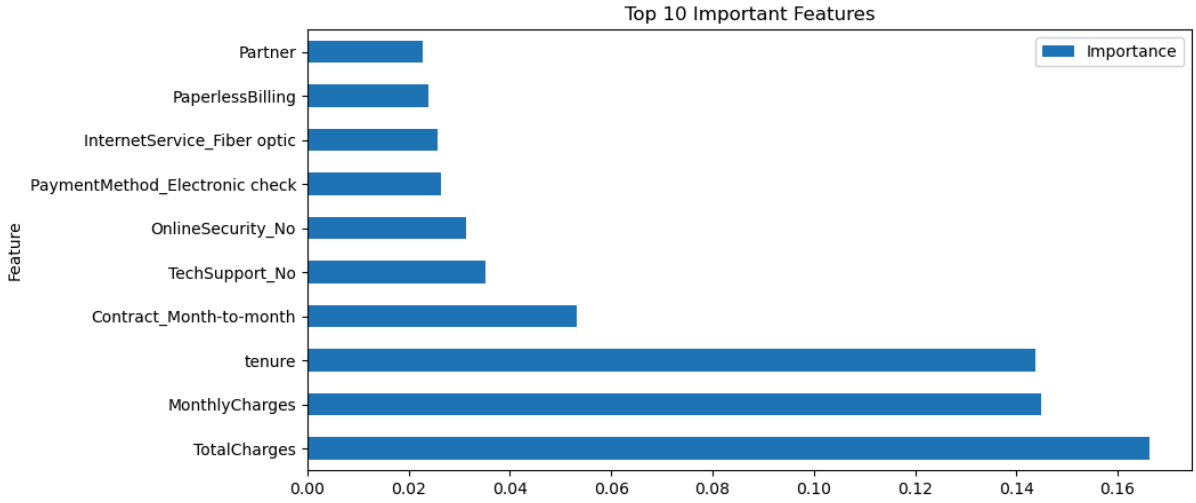
In [9]:
```
# Step 9: Evaluation
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7839374555792467
[[925 108]
 [196 178]]
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1033
           1       0.62      0.48      0.54       374

    accuracy                           0.78      1407
   macro avg       0.72      0.69      0.70      1407
weighted avg       0.77      0.78      0.77      1407
```

In [10]:
```
# Step 10: Feature importance
importances = model.feature_importances_
features = X.columns
feat_df = pd.DataFrame({'Feature': features, 'Importance': importances})
feat_df.sort_values(by='Importance', ascending=False).head(10).plot(kind='barh'
plt.title('Top 10 Important Features')
plt.show()
```



In [11]:
```
# Step 11: Save model
joblib.dump(model, 'churn_model.pkl')
```

Out[11]:
```
['churn_model.pkl']
```