telco

June 25, 2025

1 Telco Customer Churn Prediction

2 Step 1. Import Libraries

[1]:

```
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_score
import joblib
```

3 Step 2. Load Dataset

[4]:

```
df = pd.read_csv(r'C:\Users\sintu.
 ↪DESKTOP-HVVV5FJ\Desktop\Sintupy\telco\\WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

df.head()

4 Step 3.Data cleaning

[6]:

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df.dropna(inplace=True)
df.drop('customerID', axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7032 non-null   object
 1   SeniorCitizen     7032 non-null   int64
 2   Partner           7032 non-null   object
 3   Dependents        7032 non-null   object
 4   tenure            7032 non-null   int64
 5   PhoneService      7032 non-null   object
 6   MultipleLines     7032 non-null   object
 7   InternetService   7032 non-null   object
 8   OnlineSecurity    7032 non-null   object
 9   OnlineBackup      7032 non-null   object
 10  DeviceProtection  7032 non-null   object
 11  TechSupport       7032 non-null   object
 12  StreamingTV       7032 non-null   object
 13  StreamingMovies   7032 non-null   object
 14  Contract          7032 non-null   object
 15  PaperlessBilling  7032 non-null   object
 16  PaymentMethod     7032 non-null   object
 17  MonthlyCharges    7032 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7032 non-null   object
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

5 Step 4. Encode Categorical Variables

[7]:

```
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'PaperlessBilling',␣
 ↪'Churn']
for col in binary_cols:
    df[col] = LabelEncoder().fit_transform(df[col])
df['gender'] = df['gender'].map({'Male': 1, 'Female': 0})
multi_cols = ['MultipleLines', 'InternetService', 'OnlineSecurity',␣
 ↪'OnlineBackup',
              'DeviceProtection', 'TechSupport', 'StreamingTV',␣
 ↪'StreamingMovies',
              'Contract', 'PaymentMethod']
df = pd.get_dummies(df, columns=multi_cols)
df.head()
```

[7]:

```
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0              0        1           0       1             0
1       1              0        0           0      34             1
2       1              0        0           0       2             1
3       1              0        0           0      45             0
4       0              0        0           0       2             1

   PaperlessBilling  MonthlyCharges  TotalCharges  Churn  …  \
0                 1           29.85         29.85      0  …
1                 0           56.95       1889.50      0  …
```

```
2                 1           53.85        108.15      1  …
3                 0           42.30       1840.75      0  …
4                 1           70.70        151.65      1  …

   StreamingMovies_No  StreamingMovies_No internet service  \
0                True                                False   
1                True                                False   
2                True                                False   
3                True                                False   
4                True                                False   

   StreamingMovies_Yes  Contract_Month-to-month  Contract_One year  \
0                False                     True              False   
1                False                    False               True   
2                False                     True              False   
3                False                    False               True   
4                False                     True              False   

   Contract_Two year  PaymentMethod_Bank transfer (automatic)  \
0              False                                    False   
1              False                                    False   
2              False                                    False   
3              False                                     True   
4              False                                    False   

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                  False                            True   
1                                  False                           False   
2                                  False                           False   
3                                  False                           False   
4                                  False                            True   

   PaymentMethod_Mailed check  
0                       False  
1                        True  
2                        True  
3                       False  
4                       False  

[5 rows x 41 columns]
```

6 Step 5. Feature Scaling

[8]:

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

7 Step 6. Train-test Split

[9]:

```
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_state=42)
```

8 Step 7. Train Model

[11]:

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

[11]:

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

9 Step 8. Evaluate Model

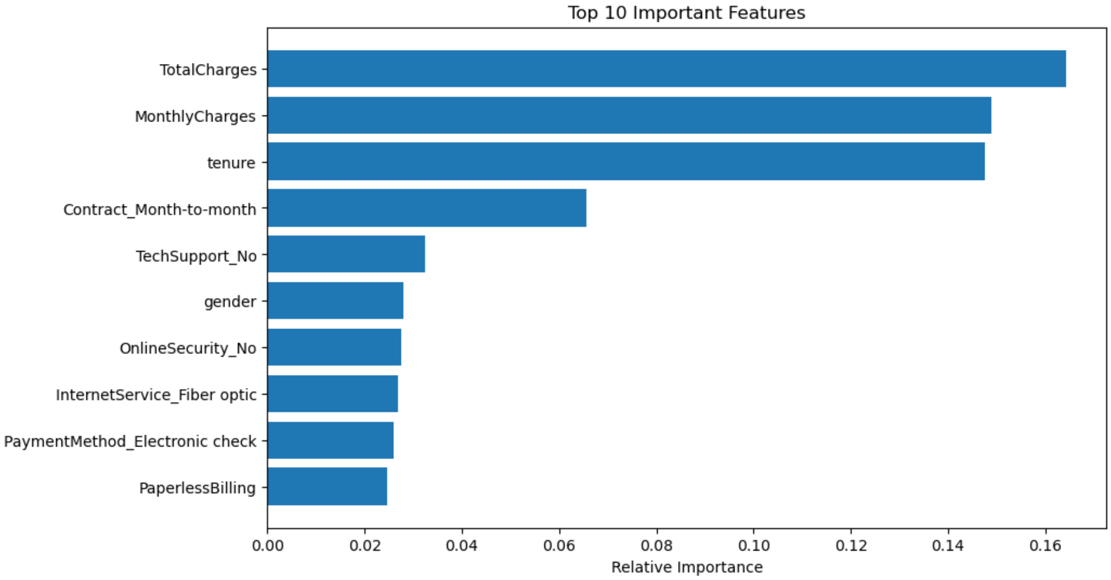
[13]:

```
y_pred = model.predict(X_test)

print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy Score: 0.7867803837953091

Confusion Matrix:
 [[923 110]
 [190 184]]

Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.89      0.86      1033
           1       0.63      0.49      0.55       374

    accuracy                           0.79      1407
   macro avg       0.73      0.69      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

10 Step 9. Feature Importance Visualization

```
[14]: importances = model.feature_importances_
features = X.columns
indices = np.argsort(importances)[-10:]  # Top 10

plt.figure(figsize=(10, 6))
plt.title('Top 10 Important Features')
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



11 Step 10. Save the Model

```
[15]: joblib.dump(model, 'telco_churn_model.pkl')
```

```
[15]: ['telco_churn_model.pkl']
```