import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Step 1: Load Data

df = pd.read\_csv('customer\_churn.csv') # Replace with your dataset

# Step 2: Preprocessing

# Handle categorical variables

label\_encoders = {}

for column in df.select\_dtypes(include=['object']).columns:

if column != 'Churn': # Assume 'Churn' is target

le = LabelEncoder()

df[column] = le.fit\_transform(df[column])

label\_encoders[column] = le

# Convert target variable

df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

# Handle missing values

df.fillna(method='ffill', inplace=True)

# Step 3: Split data

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)

# Step 4: Feature Scaling

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Step 5: Train model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Step 6: Evaluate

y\_pred = model.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

# Step 7: Uncover Hidden Patterns

importances = model.feature\_importances\_

features = X.columns

feature\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

feature\_df = feature\_df.sort\_values(by='Importance', ascending=False)

# Plot feature importance

plt.figure(figsize=(10, 6))

sns.barplot(x='Importance', y='Feature', data=feature\_df)

plt.title('Feature Importance - Hidden Patterns Behind Churn')

plt.tight\_layout()

plt.show()