# Import Libraries

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

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

import joblib

# Load the dataset

data = pd.read\_csv('Telco-Customer-Churn.csv')

# Inspect the first few rows

print(data.head())

# Drop customerID as it is not a relevant feature

data = data.drop('customerID', axis=1)

# Convert 'TotalCharges' to numeric, coerce errors and fill NaNs with mean

data['TotalCharges'] = pd.to\_numeric(data['TotalCharges'], errors='coerce')

data['TotalCharges'].fillna(data['TotalCharges'].mean(), inplace=True)

# Encode categorical variables

for column in data.columns:

if data[column].dtype == np.object:

data[column] = LabelEncoder().fit\_transform(data[column])

# Inspect the data after preprocessing

print(data.head())

# Plot the distribution of churn

sns.countplot(data['Churn'])

plt.title('Distribution of Churn')

plt.show()

# Correlation matrix

corr = data.corr()

plt.figure(figsize=(12, 8))

sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

# Define features and target variable

X = data.drop('Churn', axis=1)

y = data['Churn']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the feature variables

scaler = StandardScaler()

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

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

# Logistic Regression model

log\_model = LogisticRegression()

log\_model.fit(X\_train, y\_train)

# Random Forest model

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

rf\_model.fit(X\_train, y\_train)

# Evaluate Logistic Regression model

log\_pred = log\_model.predict(X\_test)

print("Logistic Regression Model")

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

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

print("Accuracy:", accuracy\_score(y\_test, log\_pred))

# Evaluate Random Forest model

rf\_pred = rf\_model.predict(X\_test)

print("Random Forest Model")

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

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

print("Accuracy:", accuracy\_score(y\_test, rf\_pred))

# Implementing the best model (assuming Random Forest is better in this case)

# Save the model

joblib.dump(rf\_model, 'rf\_churn\_model.pkl')

# Load the model for prediction

loaded\_model = joblib.load('rf\_churn\_model.pkl')

# Predict on new data (assuming new\_data is a preprocessed new customer data)

# Example new\_data preprocessing:

# new\_data = pd.DataFrame({

# 'SeniorCitizen': [0],

# 'tenure': [12],

# 'MonthlyCharges': [70.35],

# 'TotalCharges': [843.15],

# # add other necessary preprocessed features

# })

# new\_data = scaler.transform(new\_data) # Standardize the new data

# new\_prediction = loaded\_model.predict(new\_data)

# print("Prediction for new data:", new\_prediction)