import pandas as pd

# Example: Collect data from a CSV file

def collect\_data(file\_path):

data = pd.read\_csv(file\_path)

return data

def preprocess\_data(data):

# Handle missing values

data = data.fillna(data.mean()) # Example: Fill NaNs with mean

# Encode categorical data if necessary

data = pd.get\_dummies(data) # Convert categorical features into dummy/indicator variables

# Normalize or scale the data if required

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data)

return data\_scaled

import matplotlib.pyplot as plt

import seaborn as sns

def perform\_eda(data):

# Example: Visualize data distribution

sns.histplot(data)

plt.show()

# Example: Correlation matrix

corr\_matrix = data.corr()

sns.heatmap(corr\_matrix, annot=True)

plt.show()

def feature\_engineering(data):

# Example: Adding interaction terms or polynomial features

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)

data\_poly = poly.fit\_transform(data)

return data\_poly

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

from sklearn.linear\_model import LinearRegression

def model\_selection(data, target\_column):

# Split the data into training and testing sets

X = data.drop(target\_column, axis=1)

y = data[target\_column]

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

# Choose a model (e.g., Linear Regression)

model = LinearRegression()

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

return model, X\_test, y\_test

from sklearn.metrics import mean\_squared\_error, r2\_score

def evaluate\_model(model, X\_test, y\_test):

# Make predictions

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

# Evaluate performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

return mse, r2

def real\_time\_prediction(model, input\_data):

# Make a prediction on new data

prediction = model.predict([input\_data])

return prediction

# Visualization function for real-time results

def visualize\_prediction(prediction):

# Example: Visualize predicted result

plt.plot(prediction)

plt.title('Real-Time Prediction')

plt.show()

import streamlit as st

def deploy\_app():

st.title('Real-Time Prediction App')

st.write("Enter input data for prediction:")

# Example: Get user input for prediction

input\_data = st.text\_input('Enter features separated by commas:')

input\_data = [float(i) for i in input\_data.split(',')] # Convert to float

# Assuming the model is loaded or trained previously

# Call the model for real-time prediction

prediction = real\_time\_prediction(model, input\_data)

# Show prediction on the app

st.write('Prediction:', prediction)

if \_\_name\_\_ == "\_\_main\_\_":

# Example data collection and preprocessing steps

data = collect\_data('data.csv')

preprocessed\_data = preprocess\_data(data)

# Feature Engineering

engineered\_data = feature\_engineering(preprocessed\_data)

# Model selection

model, X\_test, y\_test = model\_selection(engineered\_data, target\_column='target')

# Model evaluation

mse, r2 = evaluate\_model(model, X\_test, y\_test)

print(f'Mean Squared Error: {mse}, R^2 Score: {r2}')

# Run the deployment application

deploy\_app()