import pandas as pd

import numpy as np

# The Boston housing dataset is no longer available in scikit-learn directly.

# We will load it from a trusted online source.

data\_url = "http://lib.stat.cmu.edu/datasets/boston"

raw\_df = pd.read\_csv(data\_url, sep="\s+", skiprows=22, header=None)

# --- FIX STARTS HERE ---

# The original data loading can fail if the file has an odd number of lines.

# We make it robust by ensuring we only process complete pairs of rows.

even\_rows = raw\_df.values[::2]

odd\_rows = raw\_df.values[1::2]

# Find the number of complete data points (pairs of rows)

num\_samples = min(len(even\_rows), len(odd\_rows))

# Combine the corresponding even and odd rows into a single data array

data = np.hstack([even\_rows[:num\_samples], odd\_rows[:num\_samples, :2]])

target = odd\_rows[:num\_samples, 2]

# --- FIX ENDS HERE ---

# Feature names (as per the dataset's documentation)

feature\_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']

# Create a pandas DataFrame

boston\_df = pd.DataFrame(data, columns=feature\_names)

# Add the target variable (Median Value in $1000s) to the DataFrame

boston\_df['MEDV'] = target

# Check for any missing values (like the 'NA' in your screenshot)

print("Missing values in each column:")

print(boston\_df.isnull().sum())

# Note: The standard online file has no missing values. If your file does,

# you would need to handle them here (e.g., by filling them with the mean).

# Display the first 5 rows of the dataset

print("\nFirst 5 rows of the dataset:")

print(boston\_df.head())

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

# --- 1. Exploratory Data Analysis (EDA) ---

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

correlation\_matrix = boston\_df.corr().round(2)

sns.heatmap(data=correlation\_matrix, annot=True, cmap='coolwarm')

plt.title("Correlation Matrix of Boston Housing Features")

plt.show()

# --- 2. Splitting the Data ---

# Separate features (X) from the target (y)

X = boston\_df.drop('MEDV', axis=1)

y = boston\_df['MEDV']

# Split 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)

# --- 3. Feature Scaling ---

scaler = StandardScaler()

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

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

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

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

print("\nModel training completed!")

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

# Make predictions on the test set

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

# Evaluate the model

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

rmse = np.sqrt(mse)

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

print(f"\nMean Squared Error (MSE): {mse:.2f}")

print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

print(f"R-squared (R²): {r2:.2f}")

# Visualize the predictions vs actual values

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, alpha=0.7)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='red', linestyle='--')

plt.title("Actual vs. Predicted Median Values")

plt.xlabel("Actual MEDV ($1000s)")

plt.ylabel("Predicted MEDV ($1000s)")

plt.grid(True)

plt.show()

**OUTPUT:**

Missing values in each column:

CRIM 0

ZN 0

INDUS 0

CHAS 0

NOX 0

RM 0

AGE 0

DIS 0

RAD 0

TAX 0

PTRATIO 0

B 0

LSTAT 0

MEDV 0

dtype: int64

First 5 rows of the dataset:

CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX \

0 0.00632 18.0 2.31 0.0 0.538 6.575 65.2 4.0900 1.0 296.0

1 0.02731 0.0 7.07 0.0 0.469 6.421 78.9 4.9671 2.0 242.0

2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0

3 0.03237 0.0 2.18 0.0 0.458 6.998 45.8 6.0622 3.0 222.0

4 0.06905 0.0 2.18 0.0 0.458 7.147 54.2 6.0622 3.0 222.0

PTRATIO B LSTAT MEDV

0 15.3 396.90 4.98 24.0

1 17.8 396.90 9.14 21.6

2 17.8 392.83 4.03 34.7

3 18.7 394.63 2.94 33.4

4 18.7 396.90 5.33 36.2

Model training completed!

Mean Squared Error (MSE): 24.29

Root Mean Squared Error (RMSE): 4.93

R-squared (R²): 0.67