**ASSIGNMET 3 - Predicting House Prices 🏠💰**

**Importing Libraries**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import fetch\_california\_housing

**Loading the Dataset**

boston = load\_boston() # This will raise a warning; consider using fetch\_california\_housing instead

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

data['PRICE'] = boston.target

**Exploring the Data**

print(data.head())

**Visualizing Correlations**

plt.figure(figsize=(10, 8))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()

**Scatter Plot of RM vs PRICE**

plt.figure(figsize=(6, 4))

plt.scatter(data['RM'], data['PRICE'])

plt.xlabel("Average number of rooms per dwelling (RM)")

plt.ylabel("House Price")

plt.title("RM vs Price")

plt.show()

**Checking for Missing Values**

print(data.isnull().sum())

**Data Preparation**

X = data.drop('PRICE', axis=1) # Features

y = data['PRICE'] # Target variable

**Standardizing the features**

scaler = StandardScaler()

X = scaler.fit\_transform(X)

**Splitting 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)

**Initializing the Models**

lr\_model = LinearRegression()

dt\_model = DecisionTreeRegressor(random\_state=42)

rf\_model = RandomForestRegressor(random\_state=42)

**Training the Models**

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

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

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

**Making Predictions**

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

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

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

**Evaluating the Models**

models = {

"Linear Regression": lr\_pred,

"Decision Tree": dt\_pred,

"Random Forest": rf\_pred

}

for name, pred in models.items():

mae = mean\_absolute\_error(y\_test, pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, pred))

print(f"{name} - MAE: {mae:.2f}, RMSE: {rmse:.2f}")

