import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

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

from sklearn.metrics import mean\_squared\_error

from sklearn.linear\_model import LinearRegression

# Load dataset

df = pd.read\_csv('../Data/BostonHousing.csv')

print(df.head())

print(df.isnull().sum()) # Check for missing values

# Scale the feature columns (except the target)

scaler = MinMaxScaler()

df.iloc[:, :-1] = scaler.fit\_transform(df.iloc[:, :-1])

# Correlation heatmap

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

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.title("Feature Correlation Heatmap")

plt.show()

# Features and target variables

x = df.iloc[:, :-1] # All columns except the last one

y = df[["rm", "b", "zn"]] # Predicting 3 columns

# Train-test split

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

# Linear regression model

model = LinearRegression()

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

# Predictions

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

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

# Evaluation: RMSE

rmse\_train = np.sqrt(mean\_squared\_error(y\_train, y\_train\_pred))

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

print(f"Train RMSE: {rmse\_train:.4f}")

print(f"Test RMSE: {rmse\_test:.4f}")

# Comparison DataFrame

comparison = pd.DataFrame({

"Actual": y\_test.values.flatten(),

"Predicted": y\_test\_pred.flatten()

})

print(comparison.head())

# Visualization: Actual vs Predicted for each target column

columns = ["rm", "b", "zn"]

train\_colors = ['blue', 'green', 'orange']

test\_colors = ['cyan', 'lightgreen', 'gold']

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

for i, col in enumerate(columns):

axes[i].scatter(y\_train[col], y\_train\_pred[:, i], color=train\_colors[i], label='Train')

axes[i].scatter(y\_test[col], y\_test\_pred[:, i], color=test\_colors[i], label='Test')

min\_val = min(y[col].min(), y[col].min())

max\_val = max(y[col].max(), y[col].max())

axes[i].plot([min\_val, max\_val], [min\_val, max\_val], color='black', linestyle='--')

axes[i].set\_xlabel("Actual Values")

axes[i].set\_ylabel("Predicted Values")

axes[i].set\_title(f"Actual vs Predicted for {col}")

axes[i].legend()

axes[i].grid(True)

plt.tight\_layout()

plt.show()