# Boston Housing Dataset Linear Regression Analysis

This document provides a detailed overview of the steps taken to perform a linear regression analysis on the Boston Housing Dataset using Python. The workflow includes data loading, exploration, visualization, model training, and evaluation. We also generate synthetic data for an additional linear regression example.

## Prerequisites

Ensure you have the necessary libraries installed:

pip install numpy pandas seaborn scikit-learn matplotlib

## Importing Libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.datasets import fetch\_openml

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

from sklearn.linear\_model import LinearRegression

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

## Loading the Dataset

boston\_dataset = fetch\_openml(data\_id=531, as\_frame=True)

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

boston['MEDV'] = boston\_dataset.target

boston.head()

## Exploratory Data Analysis

### Descriptive Statistics

Display the first few rows of the dataset:

print(boston.head())

A screenshot of a graph

Description automatically generated

### Distribution of the Target Variable

Visualize the distribution of the target variable MEDV (Median Value of Owner-Occupied Homes):

sns.set(rc={'figure.figsize': (11.7, 8.27)})

sns.histplot(boston['MEDV'], bins=30)

plt.show()

A graph of a graph

Description automatically generated

### Correlation Matrix

Generate a heatmap to show the correlation between features:

cm = boston.corr()

sns.heatmap(data=cm, annot=True)

plt.show()

A screenshot of a computer screen

Description automatically generated

### Feature Selection and Data Splitting

We select the features LSTAT (Percentage of Lower Status Population) and RM (Average Number of Rooms per Dwelling):

X = pd.DataFrame(np.c\_[boston['LSTAT'], boston['RM']], columns=['LSTAT', 'RM'])

Y = boston['MEDV']

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=5)

### Model Training and Evaluation

### Training the Model

Create and train the linear regression model:

lin\_model = LinearRegression()

lin\_model.fit(X\_train, Y\_train)

### Evaluating Model Performance

Evaluate the model using RMSE and R² score for both training and testing sets:

# Training set evaluation

y\_train\_predict = lin\_model.predict(X\_train)

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

r2\_train = r2\_score(Y\_train, y\_train\_predict)

print("The model performance for training set")

print("--------------------------------------")

print(f'RMSE is {rmse\_train}')

print(f'R2 score is {r2\_train}')

print("\n")

# Testing set evaluation

y\_test\_predict = lin\_model.predict(X\_test)

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

r2\_test = r2\_score(Y\_test, y\_test\_predict)

print("The model performance for testing set")

print("--------------------------------------")

print(f'RMSE is {rmse\_test}')

print(f'R2 score is {r2\_test}')

A screenshot of a performance

Description automatically generated

## Visualizing Predictions

Plot the actual vs. predicted prices for the testing set:

A graph showing a number of blue dots

Description automatically generated

## Additional Linear Regression Example with Synthetic Data

### Generating Synthetic Data

Create synthetic data for house sizes and prices:

np.random.seed(42)

house\_sizes = np.random.randint(1000, 2500, 100)

house\_prices = 100 + (house\_sizes \* 0.1) + (np.random.randn(100) \* 50)

plt.scatter(house\_sizes, house\_prices)

plt.xlabel('House Size (sqft)')

plt.ylabel('House Price ($k)')

plt.title('House Prices vs Size')

plt.show()

A graph showing the size of a house

Description automatically generated

### Training the Model on Synthetic Data

Prepare the data and train the model:

### X = house\_sizes.reshape(-1, 1)

### y = house\_prices

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

### model = LinearRegression()

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

### Predicting House Prices

Predict the price for a house with 2000 sqft:

predicted\_price = model.predict([[2000]])

print(f"The predicted price for a 2000 sqft house is: ${predicted\_price[0]:.2f}k")

