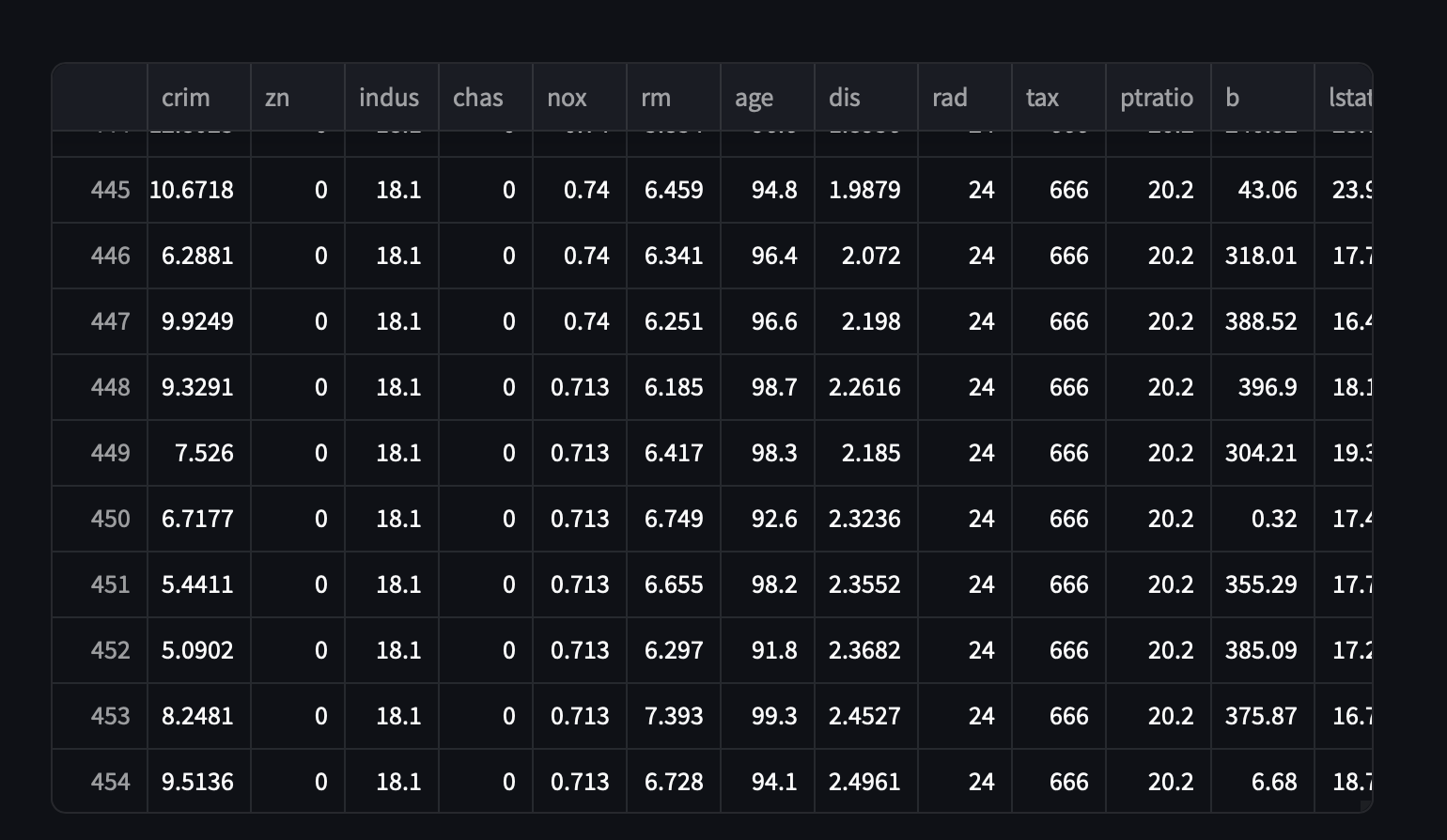
詢問

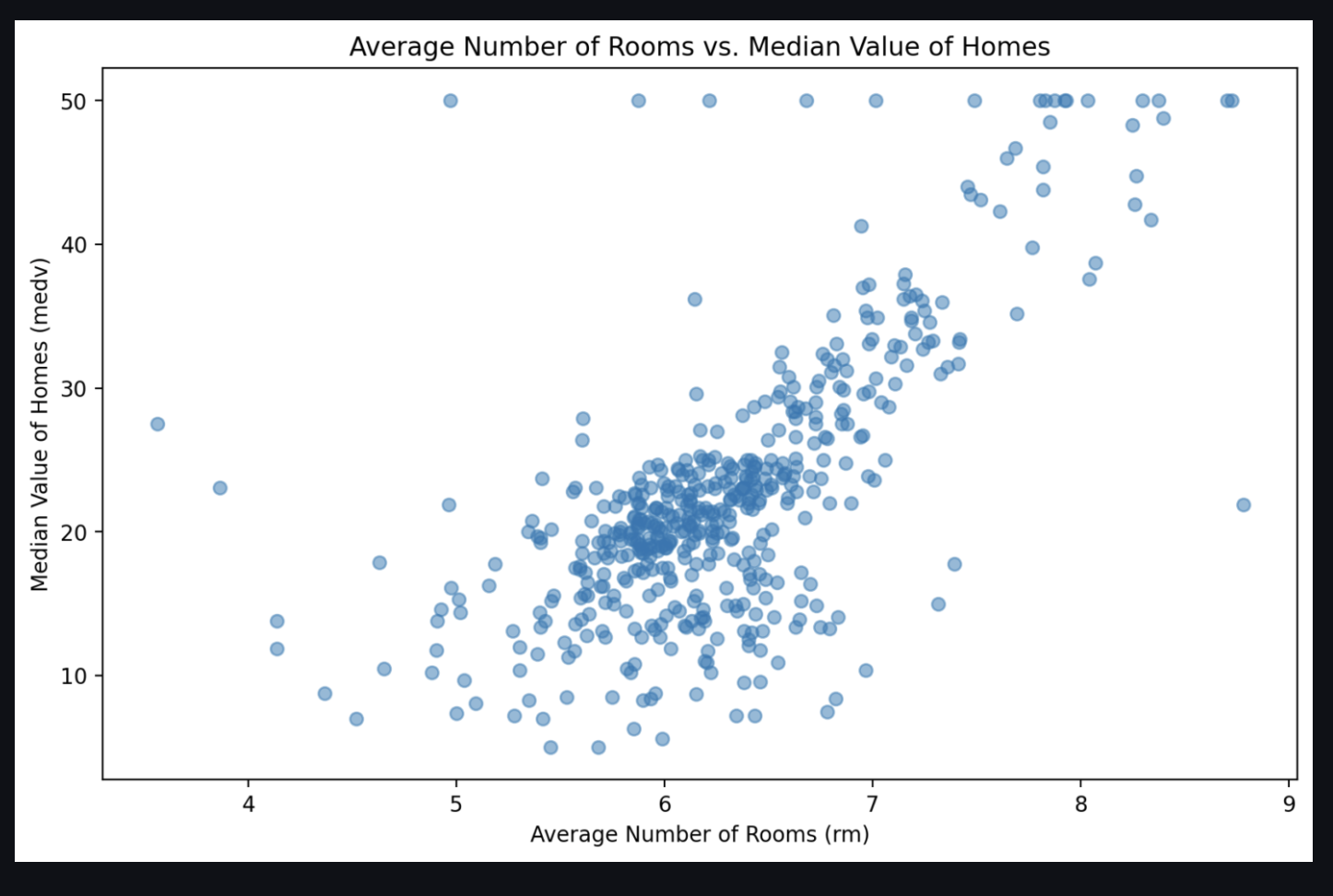
Solving the Boston Housing Problem using Scikit-Learn and CRISP-DM Step 1: 1. 再次使用chatGPT 工具利用CRISP-DM模板解決 多元回歸 Regression Problem 2. Step 1: 爬蟲抓取Boston房價 Use the URL https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv to fetch the dataset. Convert the CSV content to a pandas DataFrame and print a summary of the dataset.

### **Step 1: Data Collection**

We'll use the provided URL to fetch the Boston Housing dataset and convert it into a pandas DataFrame. Here's how you can do it:

| import streamlit as st  import pandas as pd  import matplotlib.pyplot as plt  # Fetch the dataset  url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"  boston\_housing\_df = pd.read\_csv(url)  # Display the data  st.write(boston\_housing\_df)  # Example of plotting  plt.figure(figsize=(10, 6))  plt.scatter(boston\_housing\_df['rm'], boston\_housing\_df['medv'], alpha=0.5)  plt.title('Average Number of Rooms vs. Median Value of Homes')  plt.xlabel('Average Number of Rooms (rm)')  plt.ylabel('Median Value of Homes (medv)')  st.pyplot(plt) |
| --- |



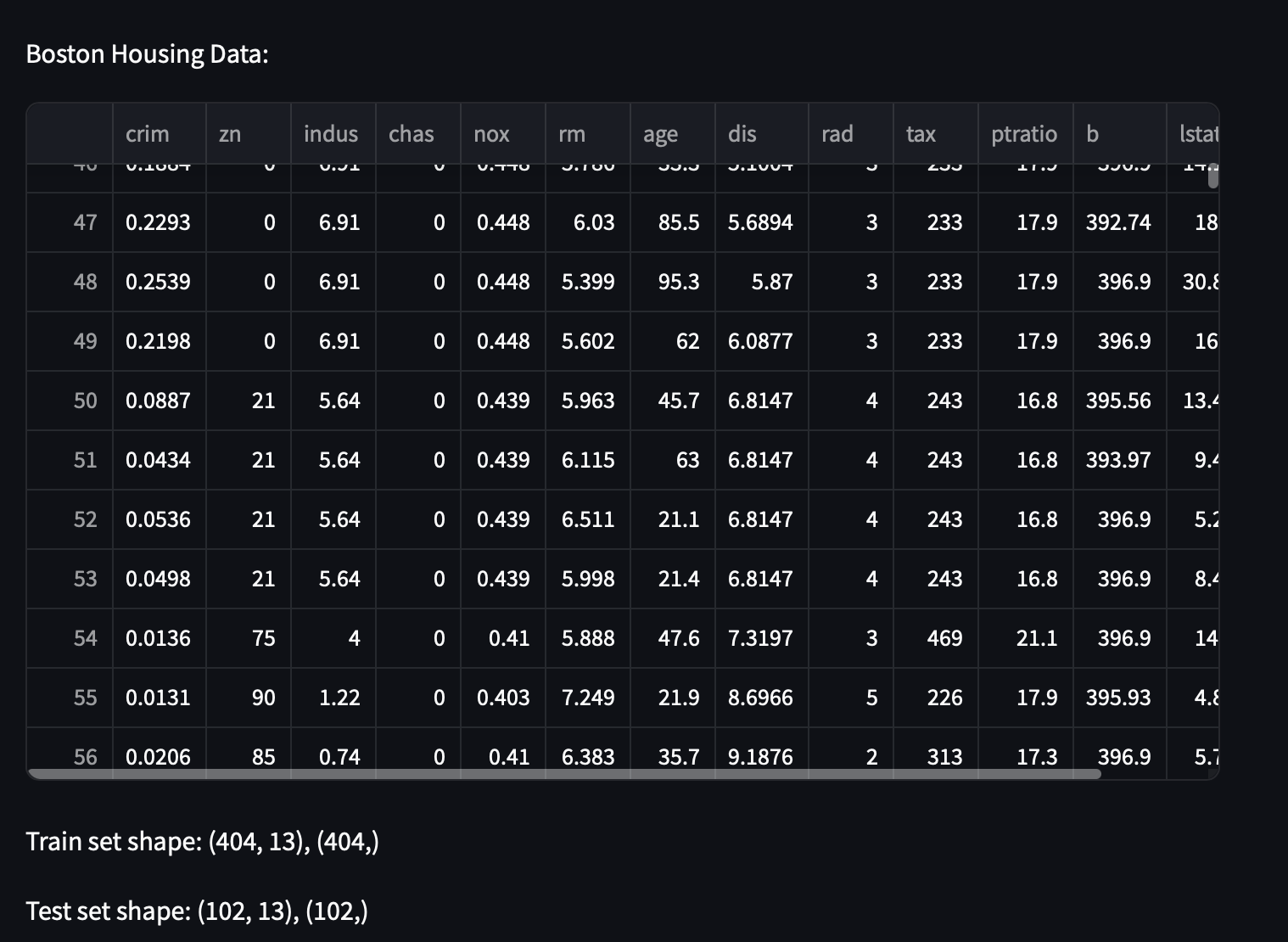


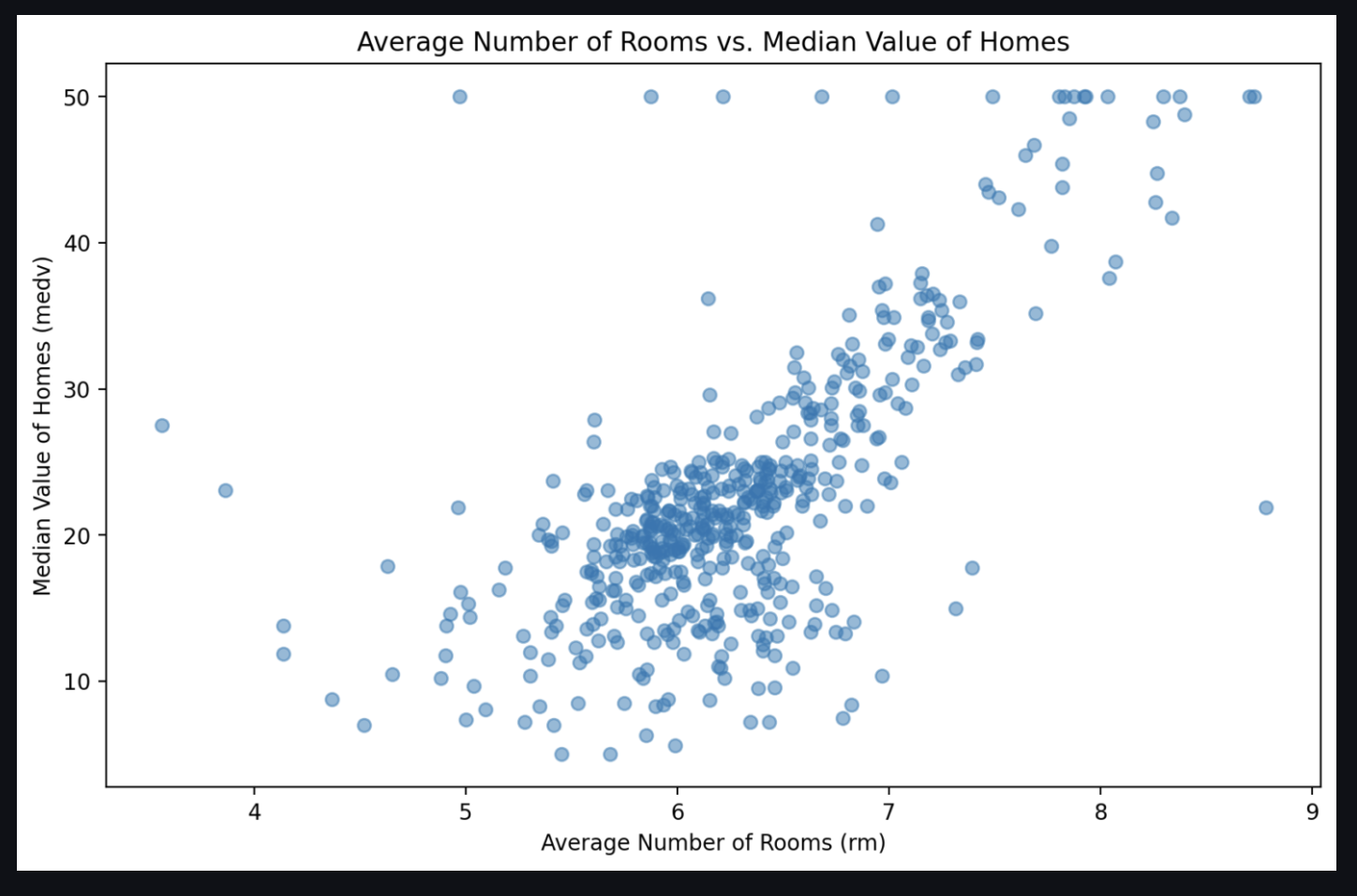
我跟他說

Step 2: Preprocessing : train test split

### **Step 2: Data Preprocessing**

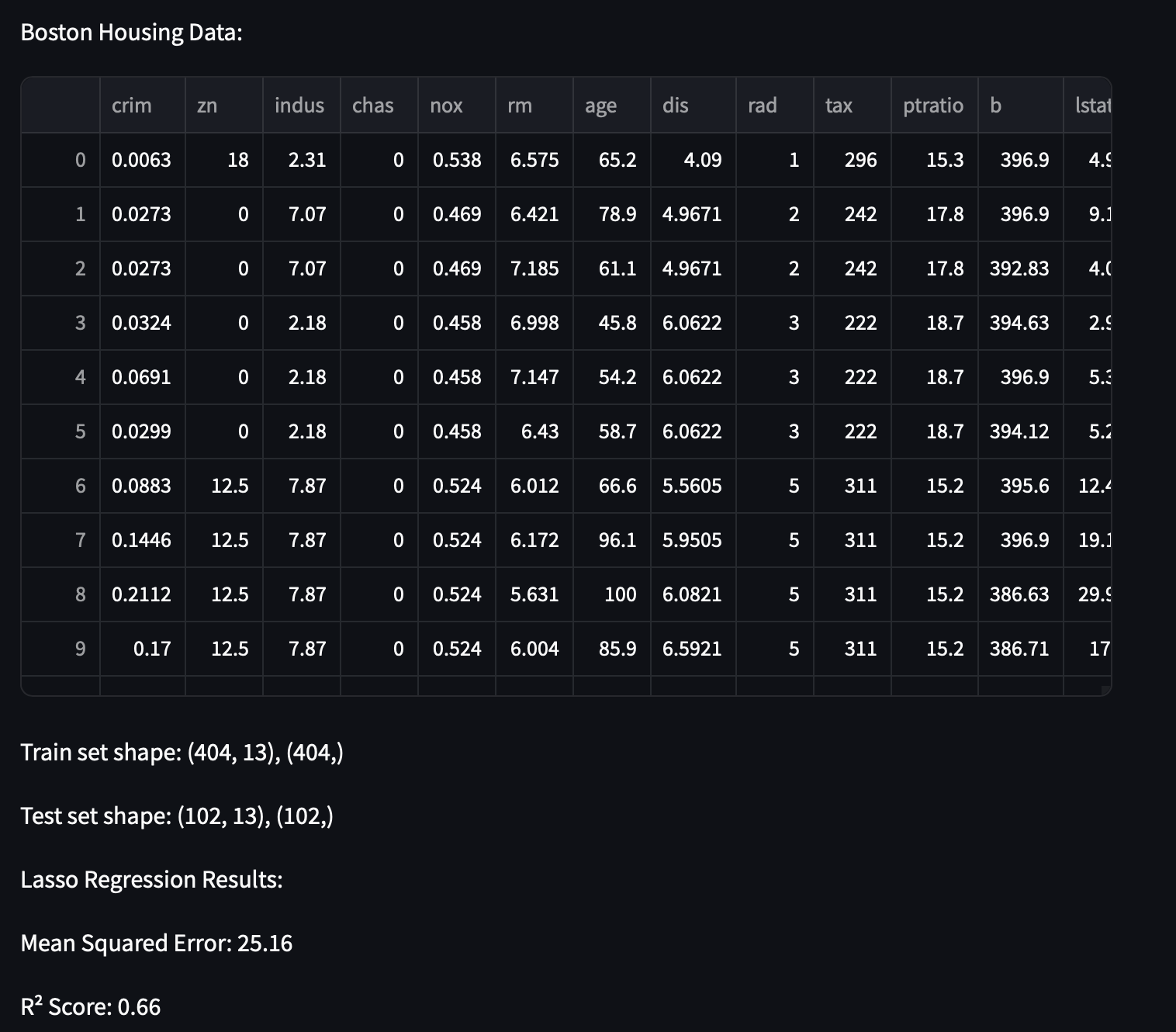
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  import streamlit as st  # Step 1: Data Collection  url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"  boston\_housing\_df = pd.read\_csv(url)  # Display the data in Streamlit  st.write("Boston Housing Data:")  st.write(boston\_housing\_df)  # Step 2: Data Preprocessing  # Separating features and target variable  X = boston\_housing\_df.drop('medv', axis=1) # Features  y = boston\_housing\_df['medv'] # Target variable  # Splitting the dataset 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)  # Output the shapes of the datasets  st.write(f'Train set shape: {X\_train.shape}, {y\_train.shape}')  st.write(f'Test set shape: {X\_test.shape}, {y\_test.shape}')  # Example of plotting the relationship between features and target  plt.figure(figsize=(10, 6))  plt.scatter(X['rm'], y, alpha=0.5)  plt.title('Average Number of Rooms vs. Median Value of Homes')  plt.xlabel('Average Number of Rooms (rm)')  plt.ylabel('Median Value of Homes (medv)')  st.pyplot(plt) |
| --- |

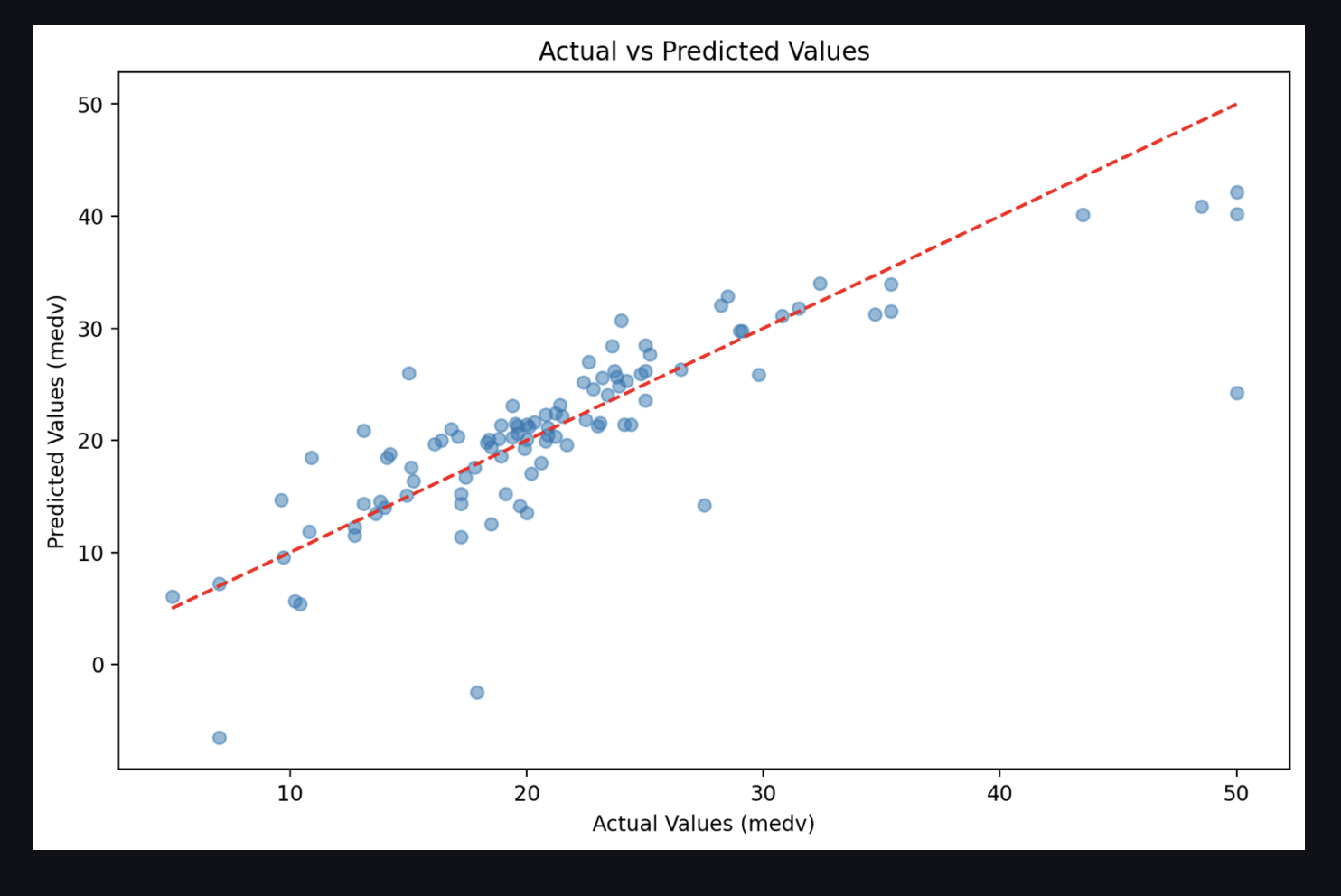




Step 3: Build Model using Lasso

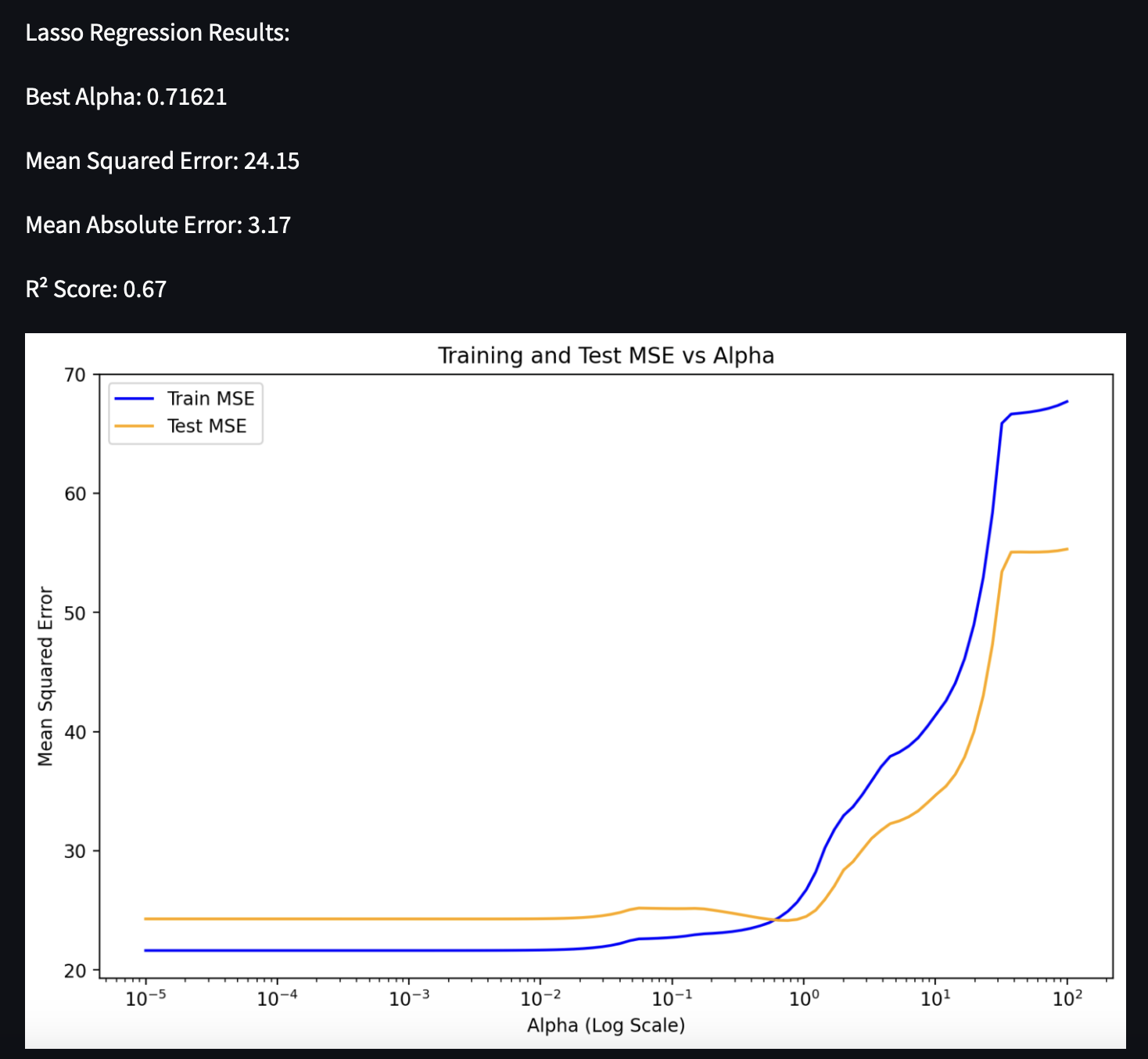
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import Lasso  from sklearn.metrics import mean\_squared\_error, r2\_score  import matplotlib.pyplot as plt  import streamlit as st  # Step 1: Data Collection  url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"  boston\_housing\_df = pd.read\_csv(url)  # Display the data in Streamlit  st.write("Boston Housing Data:")  st.write(boston\_housing\_df)  # Step 2: Data Preprocessing  # Separating features and target variable  X = boston\_housing\_df.drop('medv', axis=1) # Features  y = boston\_housing\_df['medv'] # Target variable  # Splitting the dataset 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)  # Output the shapes of the datasets  st.write(f'Train set shape: {X\_train.shape}, {y\_train.shape}')  st.write(f'Test set shape: {X\_test.shape}, {y\_test.shape}')  # Step 3: Build Model using Lasso  lasso = Lasso(alpha=0.1) # You can adjust alpha as needed  lasso.fit(X\_train, y\_train)  # Make predictions  y\_pred = lasso.predict(X\_test)  # Evaluate the model  mse = mean\_squared\_error(y\_test, y\_pred)  r2 = r2\_score(y\_test, y\_pred)  # Display the results in Streamlit  st.write("Lasso Regression Results:")  st.write(f"Mean Squared Error: {mse:.2f}")  st.write(f"R² Score: {r2:.2f}")  # Plot the actual vs predicted values  plt.figure(figsize=(10, 6))  plt.scatter(y\_test, y\_pred, alpha=0.5)  plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--') # Line for perfect prediction  plt.title('Actual vs Predicted Values')  plt.xlabel('Actual Values (medv)')  plt.ylabel('Predicted Values (medv)')  st.pyplot(plt) |
| --- |





Step 4: Evaluation: MSE, MAE, R2 metrics 的意義, overfit and underfit 的判斷（畫出 training, test curve）, 優化模型 optuna

| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import Lasso  from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score  import matplotlib.pyplot as plt  import streamlit as st  import numpy as np  import optuna  # Step 1: Data Collection  url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"  boston\_housing\_df = pd.read\_csv(url)  # Step 2: Data Preprocessing  X = boston\_housing\_df.drop('medv', axis=1)  y = boston\_housing\_df['medv']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Step 3: Build Model using Lasso  def lasso\_objective(trial):  alpha = trial.suggest\_float("alpha", 1e-5, 1e2, log=True)  lasso = Lasso(alpha=alpha)  lasso.fit(X\_train, y\_train)  y\_pred = lasso.predict(X\_test)  return mean\_squared\_error(y\_test, y\_pred)  # Optimize using Optuna  study = optuna.create\_study(direction="minimize")  study.optimize(lasso\_objective, n\_trials=100)  best\_alpha = study.best\_params['alpha']  # Train the final model  lasso = Lasso(alpha=best\_alpha)  lasso.fit(X\_train, y\_train)  # Make predictions  y\_pred = lasso.predict(X\_test)  # Evaluate the model  mse = mean\_squared\_error(y\_test, y\_pred)  mae = mean\_absolute\_error(y\_test, y\_pred)  r2 = r2\_score(y\_test, y\_pred)  # Display results  st.write("Lasso Regression Results:")  st.write(f"Best Alpha: {best\_alpha:.5f}")  st.write(f"Mean Squared Error: {mse:.2f}")  st.write(f"Mean Absolute Error: {mae:.2f}")  st.write(f"R² Score: {r2:.2f}")  # Plot training and testing curves  train\_errors, test\_errors = [], []  alpha\_values = np.logspace(-5, 2, 100)  for alpha in alpha\_values:  lasso = Lasso(alpha=alpha)  lasso.fit(X\_train, y\_train)  y\_train\_pred = lasso.predict(X\_train)  y\_test\_pred = lasso.predict(X\_test)  train\_errors.append(mean\_squared\_error(y\_train, y\_train\_pred))  test\_errors.append(mean\_squared\_error(y\_test, y\_test\_pred))  plt.figure(figsize=(10, 6))  plt.plot(alpha\_values, train\_errors, label='Train MSE', color='blue')  plt.plot(alpha\_values, test\_errors, label='Test MSE', color='orange')  plt.xscale('log')  plt.title('Training and Test MSE vs Alpha')  plt.xlabel('Alpha (Log Scale)')  plt.ylabel('Mean Squared Error')  plt.legend()  st.pyplot(plt) |
| --- |



**Step 5: Deployment**