Experiment 4. Build a linear regression model using python on given data set by

i. Prepare the data for ML model.

ii. Splitting Training data and Test data.

iii. Evaluate the model (intercept and slope).

iv. Visualize the training set and testing set using Matplotlib, Seaborn.

v. predicting the test set result.

vi. compare actual output values with predicted values.

Step 1: Import Libraries and Load the Dataset

We'll load the Boston Housing dataset, which is accessible online. We'll load it directly into a DataFrame using its URL.

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset from a URL
url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
data = pd.read_csv(url)

# Display the first few rows of the dataset
print(data.head())
```

```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3   
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8   
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8   
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7   
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7   

        b  lstat  medv  
0  396.90   4.98  24.0  
1  396.90   9.14  21.6  
2  392.83   4.03  34.7  
3  394.63   2.94  33.4  
4  396.90   5.33  36.2  
```

Step 2: Prepare Data for Modeling

The dataset contains various columns, with MEDV (Median value of owner-occupied homes) as the target variable. We'll use RM (average number of rooms per dwelling) as the input feature to predict MEDV.

```
# Step 2: Prepare data for ML model
# We are predicting the target variable 'medv' (median value of owner-occupied homes in $1000s)
# and using 'rm' (average number of rooms per dwelling) as the input feature.
X = data[['rm']]
Y = data['medv']
```

Step 3: Train and Test Split

We'll split the data into training and testing sets for model training and evaluation

```
# Step 3: Splitting Training Data and Test Data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

Step 4: Train and Evaluate the Model

Train a linear regression model on the training data and evaluate it by calculating the intercept and slope.

```
# Step 4: Train the model and evaluate the model (Intercept and Slope)
model = LinearRegression()
model.fit(X_train, Y_train)

# Model parameters
intercept = model.intercept_
slope = model.coef_[0]

print("Model Intercept:", intercept)
print("Model Slope:", slope)
```

```
Model Intercept: -36.24631889813795
Model Slope: 9.348301406497727
```

Step 5: Visualize and Compare Results

We'll visualize both training and test predictions and compare actual values with predicted values.

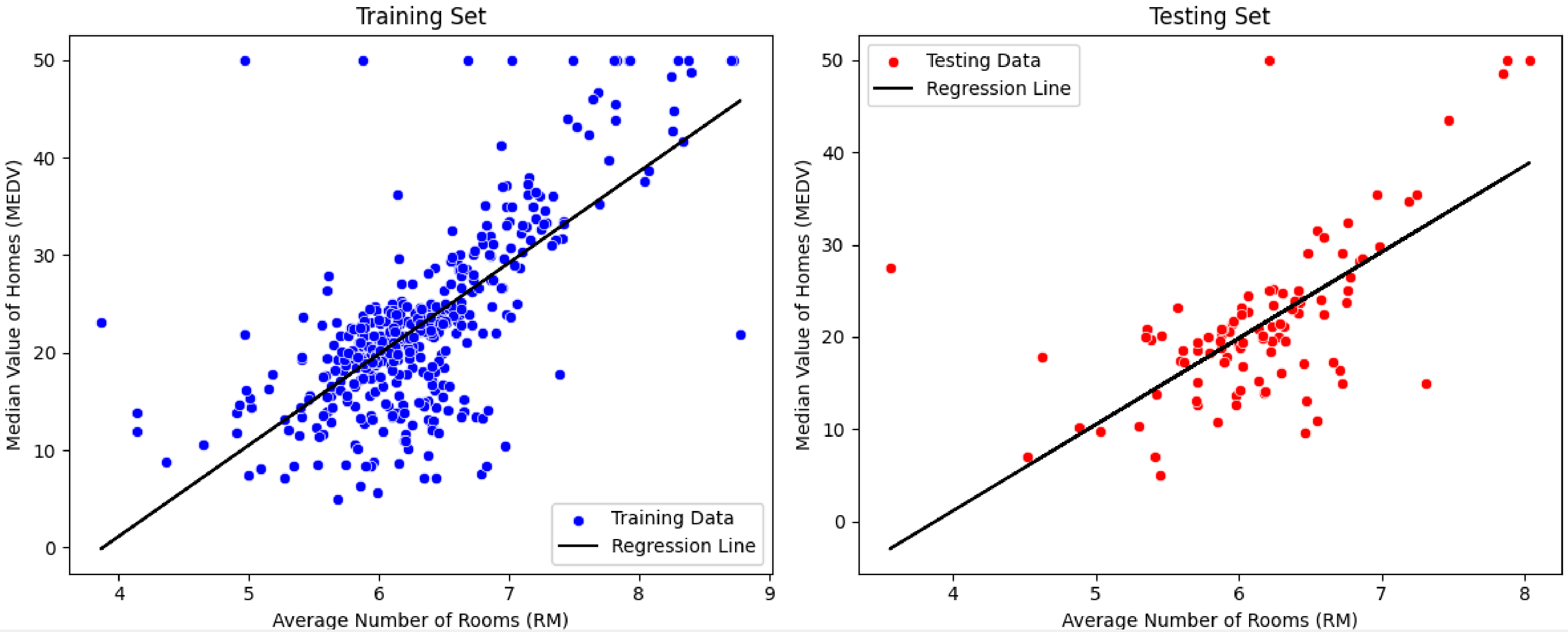
```
# Step 5: Visualize the Training Set and Testing Set
plt.figure(figsize=(12, 5))

# Training set visualization
```

```
plt.subplot(1, 2, 1)
plt.title("Training Set")
sns.scatterplot(x=X_train['rm'], y=Y_train, color='blue', label="Training Data")
plt.plot(X_train, model.predict(X_train), color='black', label="Regression Line")
plt.xlabel("Average Number of Rooms (RM)")
plt.ylabel("Median Value of Homes (MEDV)")
plt.legend()

# Testing set visualization
plt.subplot(1, 2, 2)
plt.title("Testing Set")
sns.scatterplot(x=X_test['rm'], y=Y_test, color='red', label="Testing Data")
plt.plot(X_test, model.predict(X_test), color='black', label="Regression Line")
plt.xlabel("Average Number of Rooms (RM)")
plt.ylabel("Median Value of Homes (MEDV)")
plt.legend()

plt.tight_layout()
plt.show()
```



```
# Predict the Test Set Results
Y_pred = model.predict(X_test)

# Compare Actual Output Values with Predicted Values
comparison = pd.DataFrame({'Actual': Y_test.values, 'Predicted': Y_pred})
print(comparison.head())

# Calculate model evaluation metrics
mse = mean_squared_error(Y_test, Y_pred)
mae = mean_absolute_error(Y_test, Y_pred)
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
   Actual  Predicted
0    23.6  23.732383
1    32.4  26.929502
2    13.6  19.684568
3    22.8  20.451129
4    16.1  22.619935
Mean Squared Error: 46.144775347317264
Mean Absolute Error: 4.478335832064149
```