Data Science Exploration 4- Linear Regression

December 2, 2022

0.1 ECE 131A Data science exploration 4: Please complete this jupyter notebook by filling out the code blocks. Once you have completed the notebook, generate a PDF of the completed notebook and upload the PDF to Gradescope by 11:59 PM on 12/2/2022.

0.2 In this data science exploration, we will be performing linear regression on Boston Housing Dataset. There are 506 samples and 13 feature variables in this data-set. The objective is to predict the value of prices of the house using the given features.

The description of all the features is given below:

CRIM: Per capita crime rate by town

ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

INDUS: Proportion of non-retail business acres per town

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX: Nitric oxide concentration (parts per 10 million)

RM: Average number of rooms per dwelling

AGE: Proportion of owner-occupied units built prior to 1940

DIS: Weighted distances to five Boston employment centers

RAD: Index of accessibility to radial highways

TAX: Full-value property tax rate per $10,000

B: $1000(Bk - 0.63)^2$, where Bk is the proportion of [people of African American descent] by town

LSTAT: Percentage of lower status of the population

MEDV: Median value of owner-occupied homes in $1000s

[1]:
```
## Importing the necessary packages
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns
```

```
%matplotlib inline
```

0.3 Loading the Boston Housing DataSet from scikit-learn

[2]:
```
from sklearn.datasets import load_boston

boston_dataset = load_boston()

boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston['MEDV'] = boston_dataset.target
boston.head()
```

```
/home/lawrence/.local/lib/python3.10/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function load_boston is
deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
```

[2]:

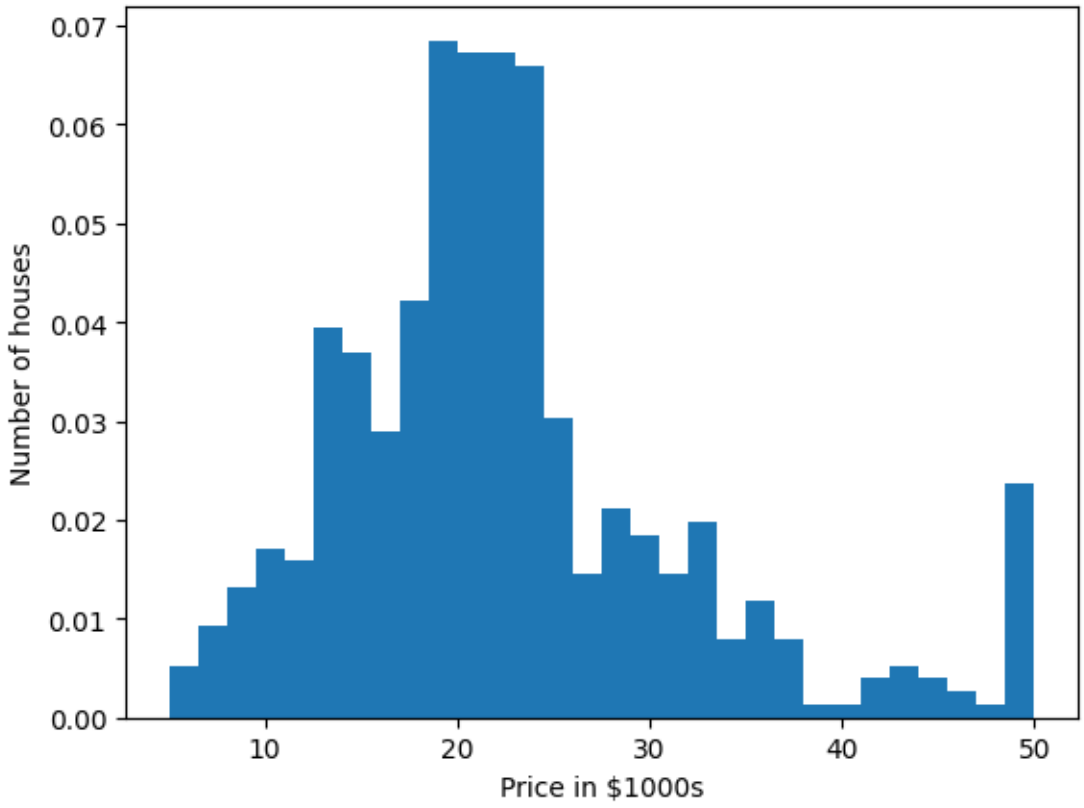
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

	PTRATIO	B	LSTAT	MEDV
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2

0.4 Data Visualization

0.5 Plot a normalized bar plot of the median value of owner occupied homes with valuation in thousands: median value in the horizontal axis and the normalized frequency in the vertical axis. For example, a house with a median value of fifty thousand dollars will be represented by 50 in the horizontal axis

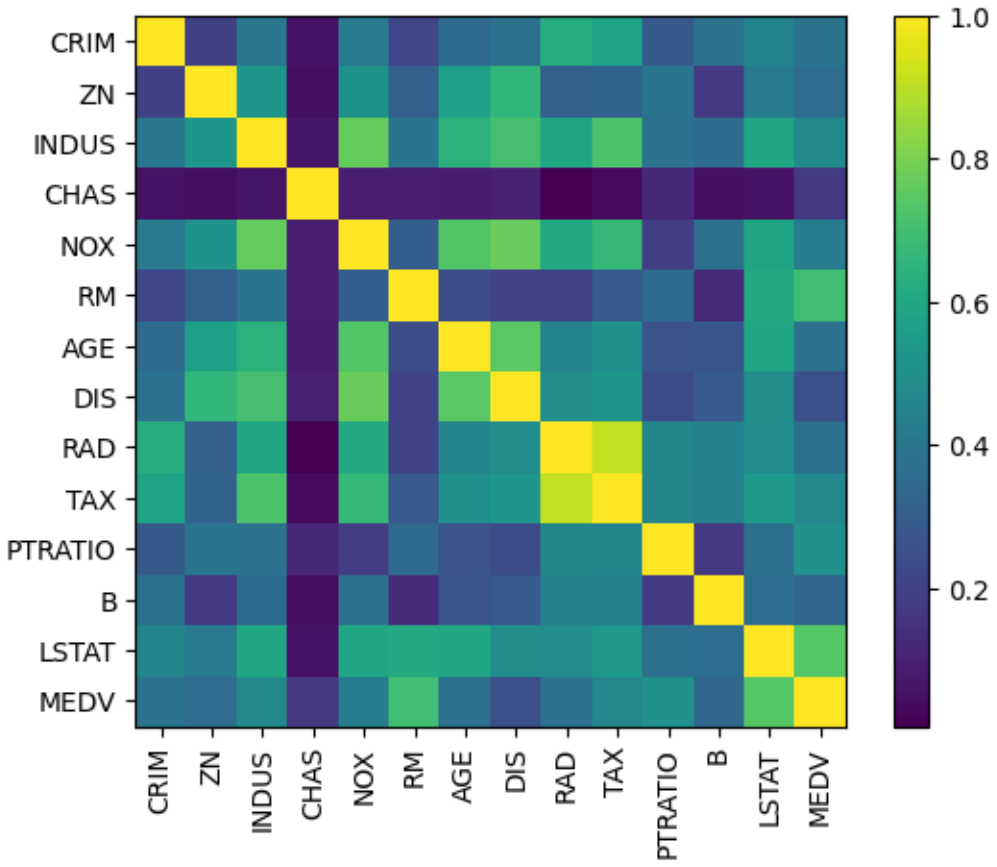
[12]:
```
# Helpful functions: sns.distplot

# Start your code here

plt.hist(boston['MEDV'], bins=30,density=True)
plt.xlabel('Price in $1000s')
plt.ylabel('Number of houses')
plt.show()

# End your code here
```



0.6 Correlation matrix: compute the pairwise correlation for all the features in the dataset and plot the correlation matrix as a heatmap. Note that since there are 13 features in the dataset, so the correlation matrix will be 13 by 13. Which two features have the strongest correlation with median home value? Please note that while finding the two features which has the strongest correlation with median home value, you need to consider the absolute correlation values.

[23]:
```
# Helpful functions: corr,sns.heatmap
# Start your code here

corr = np.abs(boston.corr())
plt.imshow(corr)
plt.colorbar()
plt.xticks(range(len(corr)), corr.columns, rotation='vertical')
plt.yticks(range(len(corr)), corr.columns)
plt.show()

#heighest two correlation with MEDV
MEDV_corr = np.abs(boston.corr()['MEDV'])
MEDV_corr = MEDV_corr.sort_values(ascending=False)
```

```
MEDV_corr = MEDV_corr[1:3]
MEDV_corr
# End your code here
```

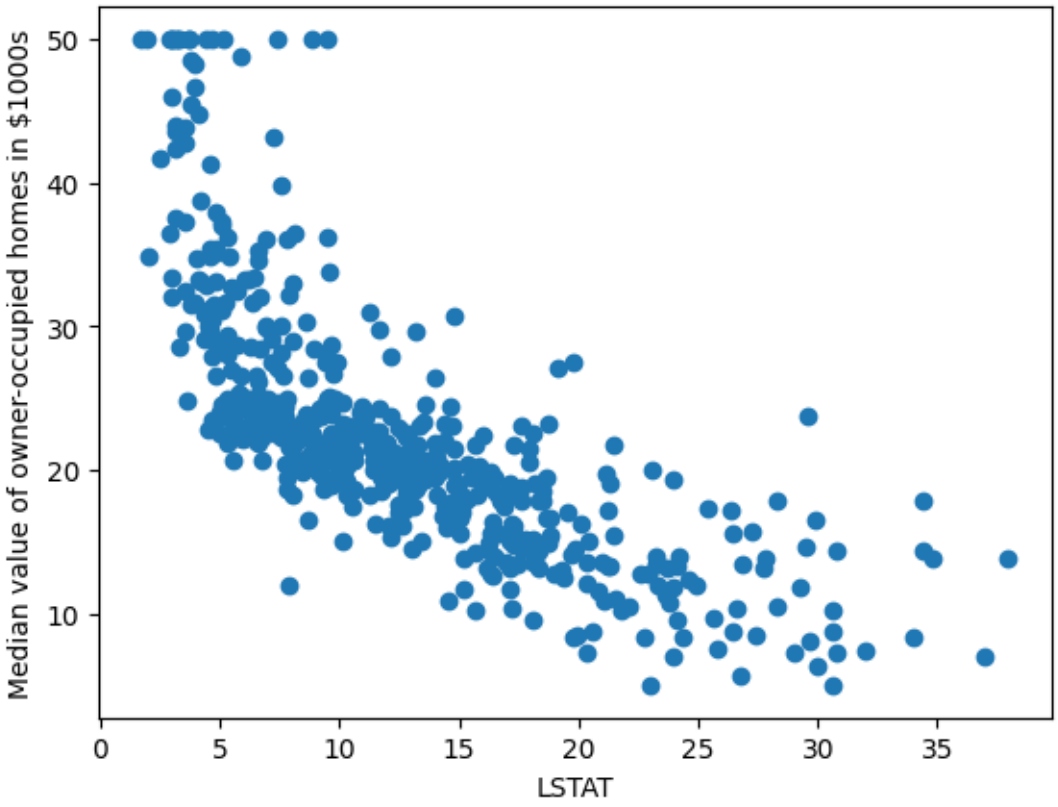


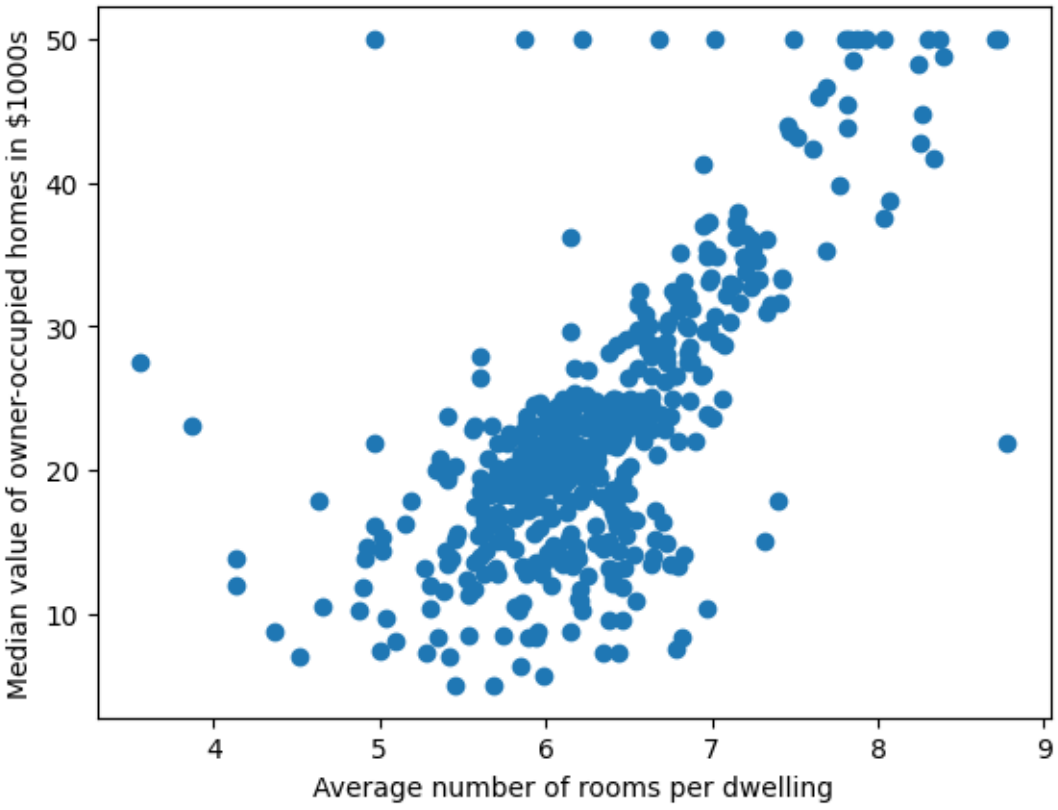
[23]:
```
LSTAT    0.737663
RM       0.695360
Name: MEDV, dtype: float64
```

0.7 Scatter plot of the feature vs the median home value: plot the feature 1 vs median home value and feature 2 vs median home value as a scatter plot where feature 1 and feature 2 are the two most strongly correlated feature found in the previous block. Note that you should have two separate plots.

[29]:
```
# Start your code here

plt.scatter(boston['LSTAT'],boston['MEDV'])
plt.xlabel('LSTAT')
plt.ylabel('Median value of owner-occupied homes in $1000s')
plt.show()
```

```
plt.scatter(boston['RM'],boston['MEDV'])
plt.xlabel('Average number of rooms per dwelling')
plt.ylabel('Median value of owner-occupied homes in $1000s')
plt.show()


# End your code here
```




Median value of owner-occupied homes in $1000s
50
40
30
20
10
4
5
6
7
8
9
Average number of rooms per dwelling

0.8 Linear regression using feature 1: we will train a linear regression model to predict the median house prices using feature 1. It will consist of the following steps:

0.9 1. Prepare a dataframe consisting of the feature 1 values and call it X_1 and a dataframe consisting of the median home values and call it Y

0.10 2. Split the X_1 and Y into training and testing sets with 80% in the training set and 20% in the testing set. After this splitting, we will have X_1_train, X_1_test, Y_train, Y_test.

0.11 3. Train a linear regression model using X_1_train and Y_train

0.12 4. Evaluate the trained model on Y_train and Y_test. The evaluation metric that we will be using are root mean squared error (rmse) and r2_score

0.13 Print the shape of X_1_train, X_1_test, Y_train, Y_test. Also, print the rmse and r2_score for both Y_train and Y_test

[33]:
```
# Helpful functions: train_test_split,LinearRegression,fit,predict,r2_score
# Start your code here
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

X1 = boston['LSTAT']
Y1= boston['MEDV']

X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size=0.2,␣
 ↪random_state=5)

model=LinearRegression().fit(X1_train.values.reshape(-1,1),Y1_train.values.
 ↪reshape(-1,1))

Y1_train_predict = model.predict(X1_train.values.reshape(-1,1))
Y1_test_predict = model.predict(X1_test.values.reshape(-1,1))

rmse = (np.sqrt(mean_squared_error(Y1_train, Y1_train_predict)))
r2 = r2_score(Y1_train, Y1_train_predict)

print('The model performance for training set')
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

rmse = (np.sqrt(mean_squared_error(Y1_test, Y1_test_predict)))
r2 = r2_score(Y1_test, Y1_test_predict)

print('The model performance for testing set')
```

```
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

print("shape of X1_train",X1_train.shape)
print("shape of Y1_train",Y1_train.shape)
print("shape of X1_test",X1_test.shape)
print("shape of Y1_test",Y1_test.shape)

# End your code here
```

```
The model performance for training set
RMSE is 6.201452973865344
R2 score is 0.5523019908037391
The model performance for testing set
RMSE is 6.2307165730986815
R2 score is 0.5041523728903132
shape of X1_train (404,)
shape of Y1_train (404,)
shape of X1_test (102,)
shape of Y1_test (102,)
```

0.14 Linear regression using feature 2: we will train a linear regression model to predict the median house prices using feature 2. It will consist of the following steps:

0.15 1. Prepare a dataframe consisting of the feature 2 values and call it X_2 and a dataframe consisting of the median home values and call it Y

0.16 2. Split the X_2 and Y into training and testing sets with 80% in the training set and 20% in the testing set. After this splitting, we will have X_2_train, X_2_test, Y_train, Y_test.

0.17 3. Train a linear regression model using X_2_train and Y_train

0.18 4. Evaluate the trained model on Y_train and Y_test. The evaluation metric that we will be using are root mean squared error (rmse) and r2_score

0.19 Print the shape of X_2_train, X_2_test, Y_train, Y_test. Also, print the rmse and r2_score for both Y_train and Y_test

[34]:
```
# Helpful functions: train_test_split,LinearRegression,fit,predict,r2_score
# Start your code here
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

pass
```

```
X2 = boston['RM']
Y2= boston['MEDV']

X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y2, test_size=0.2,␣
 ↪random_state=5)

model=LinearRegression().fit(X2_train.values.reshape(-1,1),Y2_train.values.
 ↪reshape(-1,1))

Y2_train_predict = model.predict(X2_train.values.reshape(-1,1))
Y2_test_predict = model.predict(X2_test.values.reshape(-1,1))

rmse = (np.sqrt(mean_squared_error(Y2_train, Y2_train_predict)))
r2 = r2_score(Y2_train, Y2_train_predict)

print('The model performance for training set')
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

rmse = (np.sqrt(mean_squared_error(Y2_test, Y2_test_predict)))
r2 = r2_score(Y2_test, Y2_test_predict)

print('The model performance for testing set')
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

print("shape of X2_train",X2_train.shape)
print("shape of Y2_train",Y2_train.shape)
print("shape of X2_test",X2_test.shape)
print("shape of Y2_test",Y2_test.shape)

# End your code here
```

```
The model performance for training set
RMSE is 6.972277149440585
R2 score is 0.4340897790637215
The model performance for testing set
RMSE is 4.895963186952216
R2 score is 0.6938399401553497
shape of X2_train (404,)
shape of Y2_train (404,)
shape of X2_test (102,)
shape of Y2_test (102,)
```

0.20 **Linear regression using both features 1 and 2: we will train a linear regression model to predict the median house prices using both features 1 and 2. It will consist of the following steps:**

0.21 **1. Prepare a dataframe consisting of the feature 1 and feature 2 values and call it X_both and a dataframe consisting of the median home values and call it Y**

0.22 **2. Split the X_both and Y into training and testing sets with 80% in the training set and 20% in the testing set. After this splitting, we will have X_both_train, X_both_test, Y_train, Y_test.**

0.23 **3. Train a linear regression model using X_both_train and Y_train**

0.24 **4. Evaluate the trained model on Y_train and Y_test. The evaluation metric that we will be using are root mean squared error (rmse) and r2_score**

0.25 **Print the shape of X_both_train, X_both_test, Y_train, Y_test. Also, print the rmse and r2_score for both Y_train and Y_test**

[36]:
```
# Helpful functions: train_test_split,LinearRegression,fit,predict,r2_score
# Start your code here
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

pass

X_both= boston[['RM','LSTAT']]
Y_both= boston['MEDV']

X_both_train, X_both_test, Y_both_train, Y_both_test = train_test_split(X_both,␣
 ↪Y_both, test_size=0.2, random_state=5)

model=LinearRegression().fit(X_both_train,Y_both_train)

Y_both_train_predict = model.predict(X_both_train)
Y_both_test_predict = model.predict(X_both_test)

rmse = (np.sqrt(mean_squared_error(Y_both_train, Y_both_train_predict)))
r2 = r2_score(Y_both_train, Y_both_train_predict)

print('The model performance for training set')
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))


rmse = (np.sqrt(mean_squared_error(Y_both_test, Y_both_test_predict)))
```

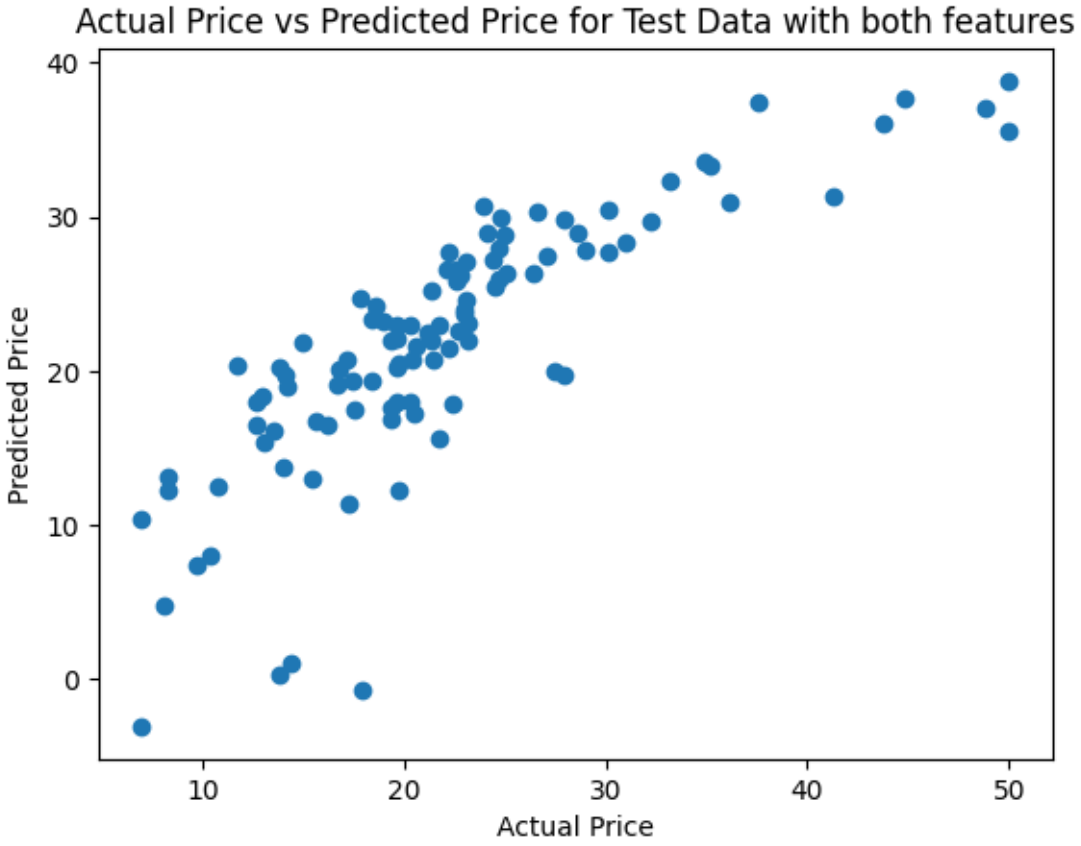
```
r2 = r2_score(Y_both_test, Y_both_test_predict)

print('The model performance for testing set')
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

print("shape of X_both_train",X_both_train.shape)
print("shape of Y_both_train",Y_both_train.shape)
print("shape of X_both_test",X_both_test.shape)
print("shape of Y_both_test",Y_both_test.shape)

# End your code here
```

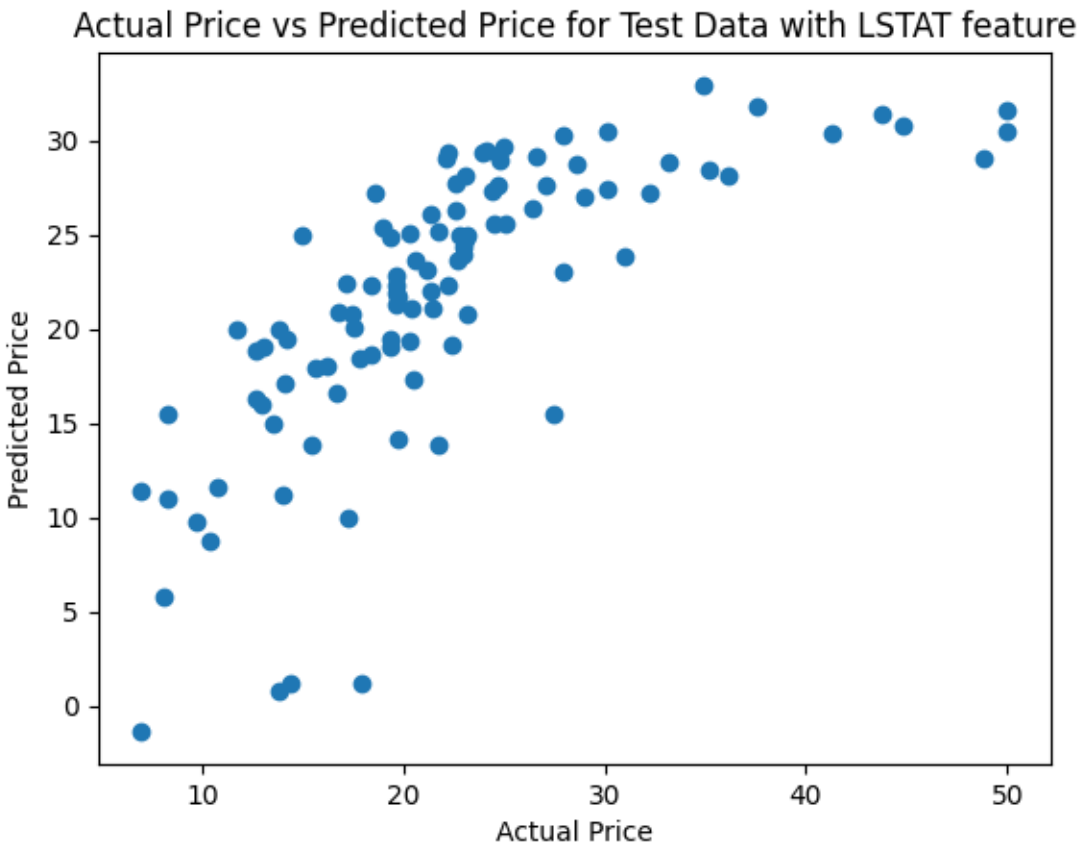
```
The model performance for training set
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701
The model performance for testing set
RMSE is 5.137400784702911
R2 score is 0.6628996975186953
shape of X_both_train (404, 2)
shape of Y_both_train (404,)
shape of X_both_test (102, 2)
shape of Y_both_test (102,)
```

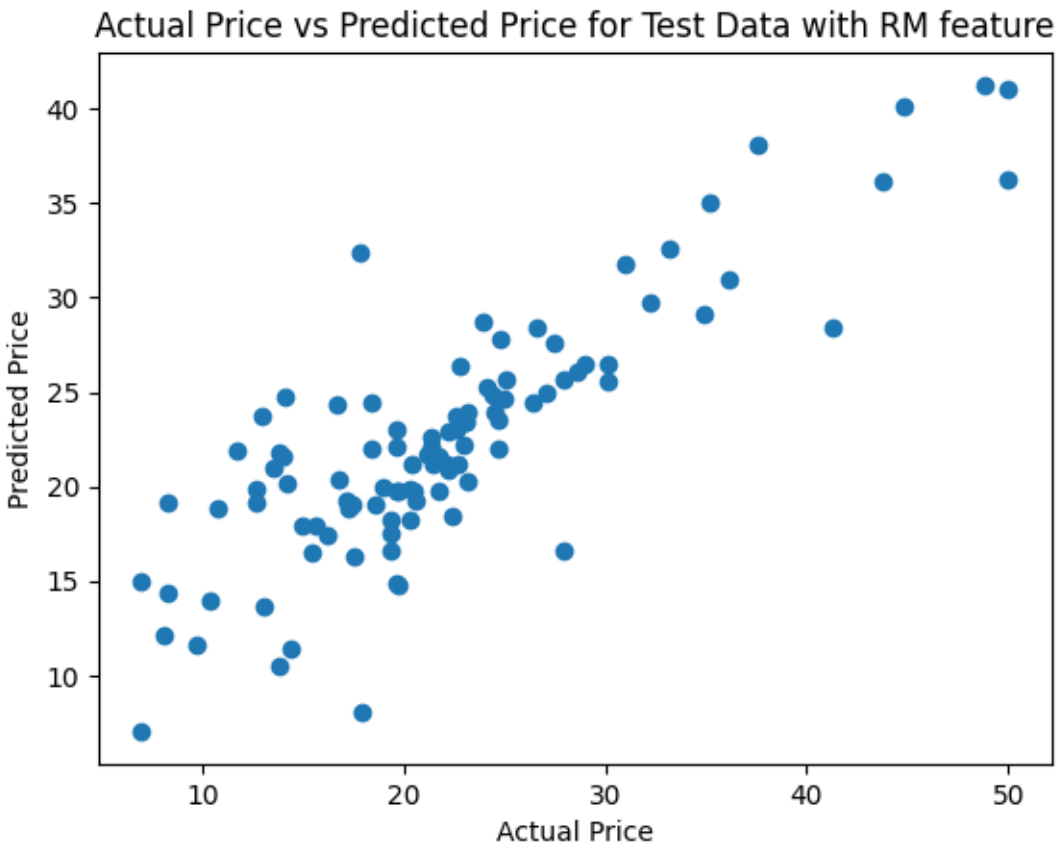
0.26 Scatter plots of the actual median home values vs predicted median home values on the test set: plot the actual vs predicted median home values on the test set for all the three linear regression models using a sctatter plot. Please note that you will have three scatter plots

[39]:
```
# Start your code here

plt.scatter(Y_both_test,Y_both_test_predict)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price vs Predicted Price for Test Data with both features")
plt.show()

plt.scatter(Y1_test,Y1_test_predict)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price vs Predicted Price for Test Data with LSTAT feature")
plt.show()

plt.scatter(Y2_test,Y2_test_predict)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price vs Predicted Price for Test Data with RM feature")
```

```
plt.show()
# End your code here
```




Actual Price vs Predicted Price for Test Data with LSTAT feature
Predicted Price
Actual Price
0
5
10
15
20
25
30
10
20
30
40
50


Actual Price vs Predicted Price for Test Data with RM feature
Predicted Price
Actual Price
10
15
20
25
30
35
40
10
20
30
40
50

[ ]: