practical-4-dsbda

May 4, 2025

[2]:
```
#practical 4
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

[3]:
```
df = pd.read_csv("Boston.csv")
df.head()
```

[3]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64
```

```
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64
 10  tax         506 non-null    int64
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

[5]:
```
df.describe()
```

[5]:

	Unnamed: 0	crim	zn	indus	chas	nox	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	

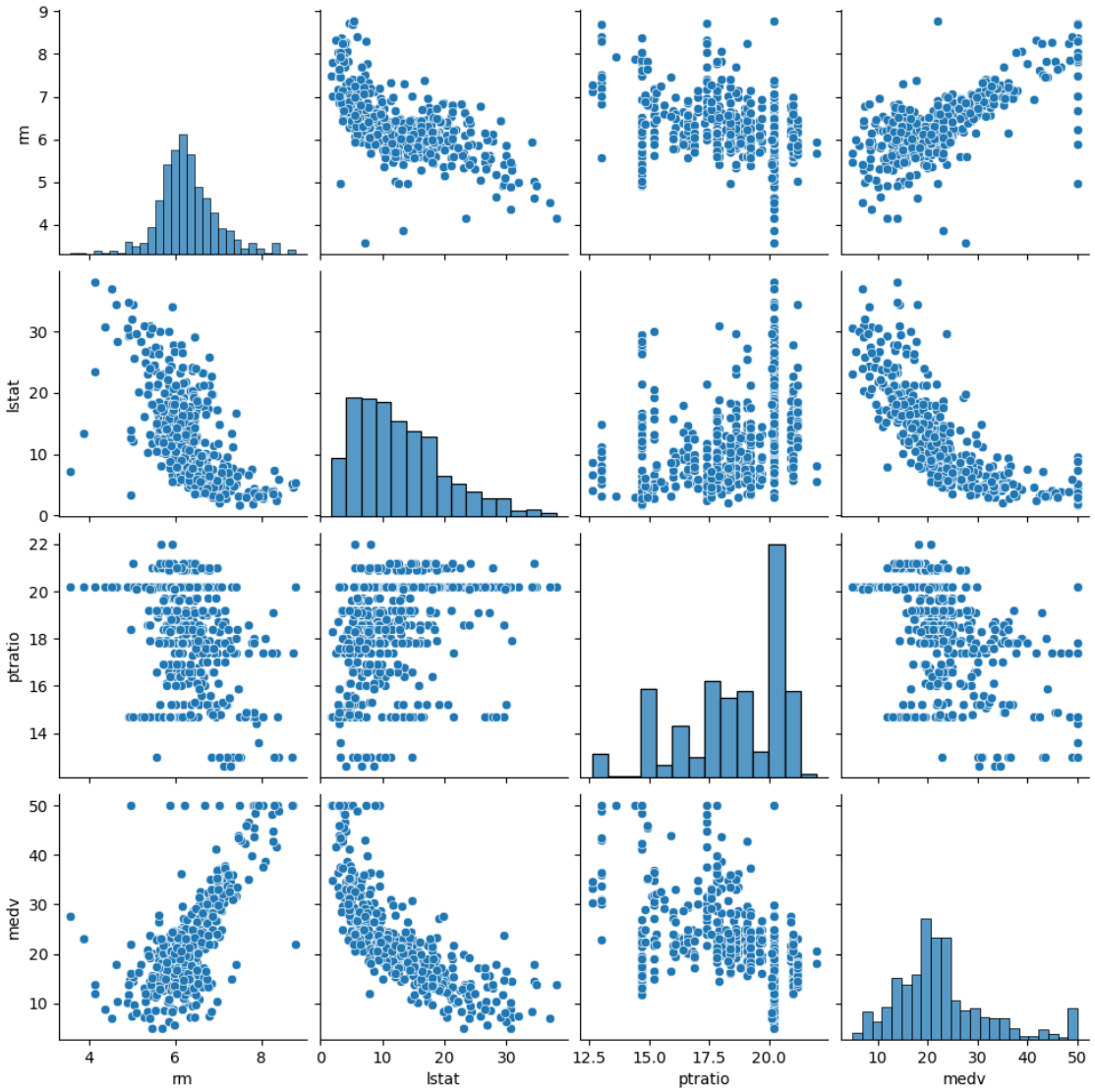
	rm	age	dis	rad	tax	ptratio	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	
std	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	
min	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	
25%	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	
50%	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	
75%	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	
max	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	

	black	lstat	medv
count	506.000000	506.000000	506.000000
mean	356.674032	12.653063	22.532806
std	91.294864	7.141062	9.197104
min	0.320000	1.730000	5.000000
25%	375.377500	6.950000	17.025000
50%	391.440000	11.360000	21.200000
75%	396.225000	16.955000	25.000000
max	396.900000	37.970000	50.000000

[6]:
```
df.isnull().sum()
```

[6]:
```
Unnamed: 0    0
crim          0
zn            0
indus         0
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
medv          0
dtype: int64
```

[8]:
```
sns.pairplot(df[['rm', 'lstat', 'ptratio', 'medv']])
plt.show()
```



[11]:
```
x = df.drop('medv',axis=1)
y = df['medv']
```

[12]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
↪2,random_state=42)
```

[13]:
```
model = LinearRegression()
model.fit(x_train,y_train)
```

[13]:
```
LinearRegression()
```

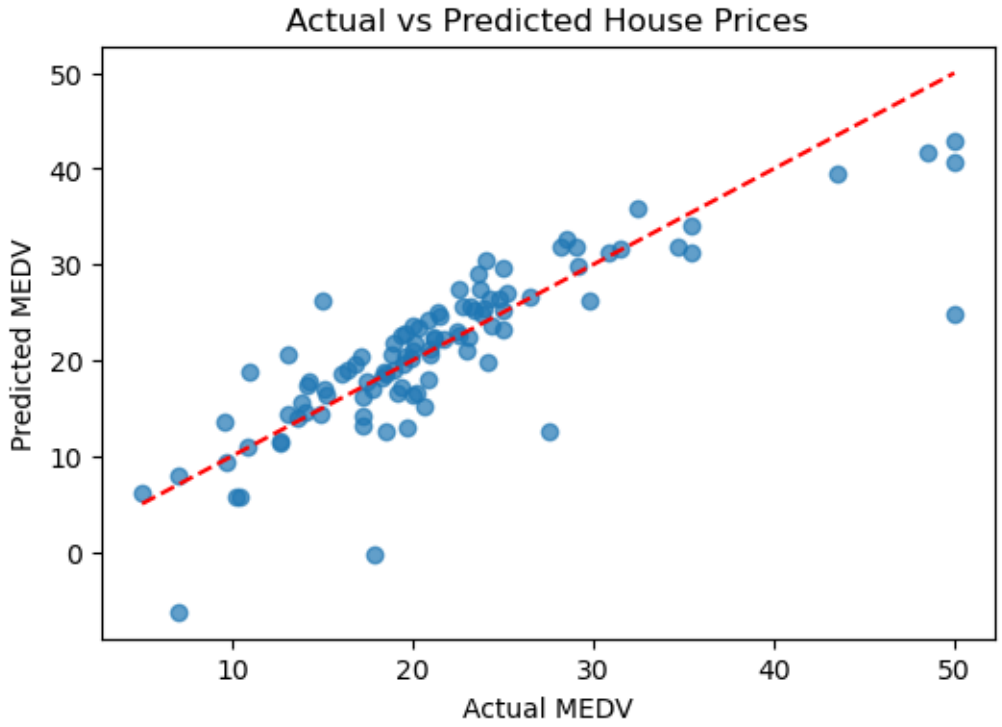
[15]:
```
y_pred = model.predict(x_test)
```

[16]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 24.497819777630365
R² Score: 0.6659408703343039
```

[17]:
```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.xlabel("Actual MEDV")
plt.ylabel("Predicted MEDV")
plt.title("Actual vs Predicted House Prices")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.show()
```



[ ]: