In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [5]:
```
df = pd.read_csv("Boston.csv")
```

In [9]:
```
df.shape[0]
```

Out[9]:
```
506
```

In [10]:
```
df.shape[1]
```

Out[10]:
```
15
```

In [11]:
```
df.shape
```

Out[11]:
```
(506, 15)
```

In [13]:
```
df.isnull()
```

Out[13]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	..
501	False	False	False	False	False	False	False	False	False	False	False	False	False
502	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	False	False	False	False	False	False	False

506 rows × 15 columns

In [12]:
```
df.isna()
```

Out[12]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	..
501	False	False	False	False	False	False	False	False	False	False	False	False	False
502	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	False	False	False	False	False	False	False

506 rows × 15 columns

In [14]:
```
df.isna().sum()
```

Out[14]:
```
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
```

In [5]:
```
df
```

Out[5]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99
502	503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90
503	504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90
504	505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45
505	506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90

506 rows × 15 columns

In [6]:
```
print(f'There are {df.shape[0]} - rows and {df.shape[1]} - columns in dataset'
```

```
There are 506 - rows and 15 - columns in dataset
```

In [7]:
```
df = df.drop('Unnamed: 0',axis=1)
df[:3]
```

Out[7]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	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	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	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

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'ta
x',
       'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

In [11]:
```
df.rename(columns={'medv':'price'},inplace=True)
df[:3]
```

Out[11]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	price
0	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	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	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

In [12]:
```
df.describe()
```

Out[12]:

	crim	zn	indus	chas	nox	rm	age	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.

In [13]:
```
df.corr()
```

Out[13]:

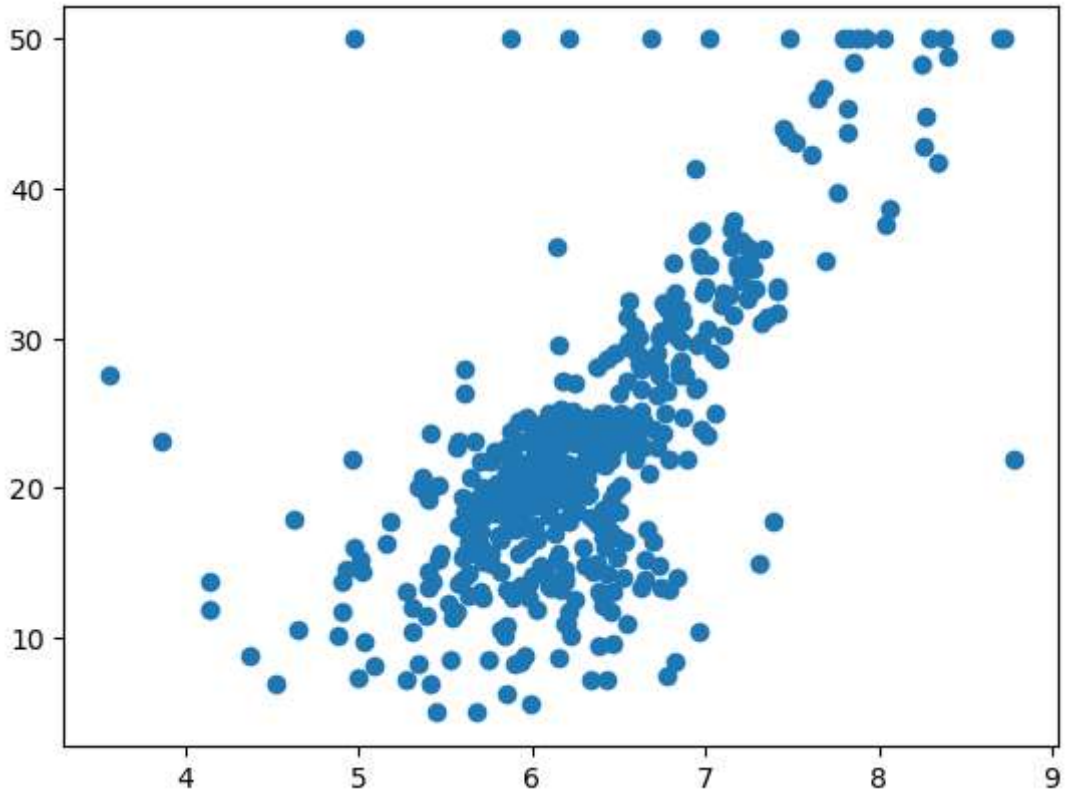
	crim	zn	indus	chas	nox	rm	age	dis	
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.6
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.3
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.5
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.0
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.6
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.2
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.4
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.4
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.0
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.9
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.4
black	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.4
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.4
price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.3

In [14]:
```
plt.figure(figsize=(12,10))
sns.heatmap(df.corr(),annot=True)
```

Out[14]: <AxesSubplot:>

In [15]:
```
plt.scatter(df.rm,df.price)
```

Out[15]: <matplotlib.collections.PathCollection at 0x234172ee5b0>



In [19]:
```
X = df[['rm']]
y = df['price']
```

In [20]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
```

In [21]:
```
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(354, 1) (354,)
(152, 1) (152,)
```

In [22]:
```
from sklearn.linear_model import LinearRegression

reg_model = LinearRegression()
reg_model.fit(X_train,y_train)
```

Out[22]:
```
LinearRegression()
```

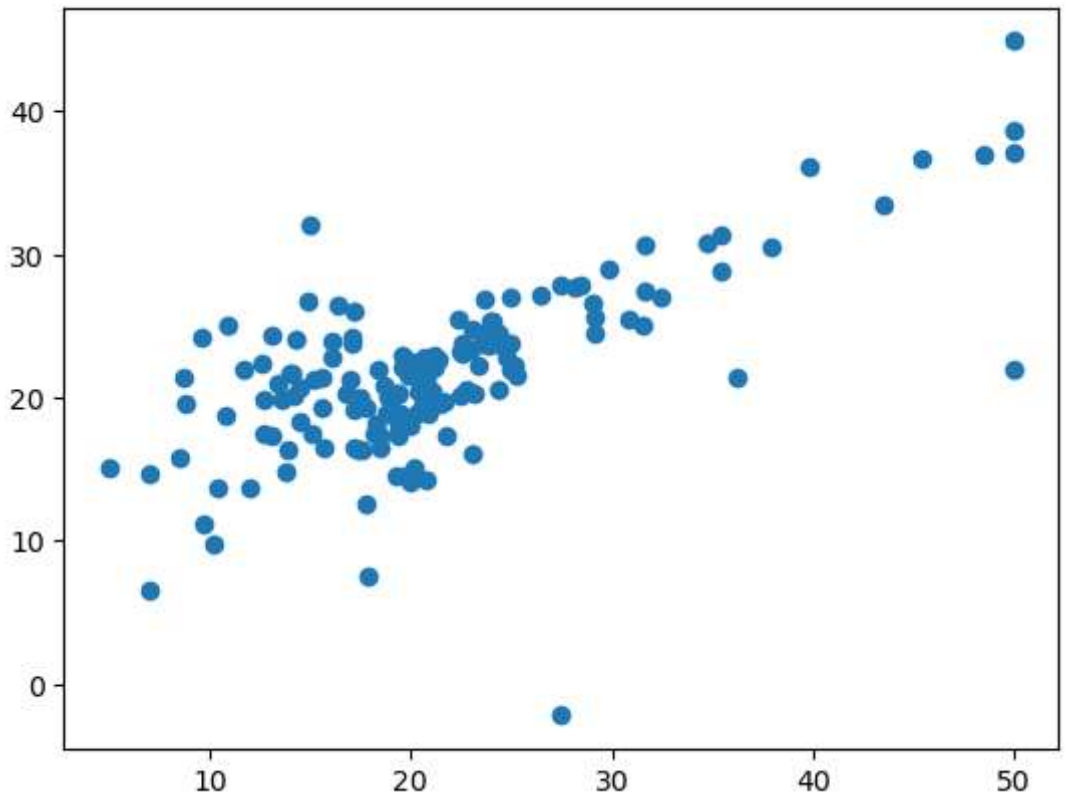
In [23]:
```
ypred = reg_model.predict(X_test)
```

In [24]:
```
reg_model.score(X_test,y_test)*100
```

Out[24]:
```
45.846499343030686
```

In [25]:
```
plt.scatter(y_test,ypred)
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x234170c6250>
```



In [26]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from math import sqrt
```

In [27]:
```
mean_absolute_error(y_test,ypred)
```

Out[27]:
```
4.314224104076755
```

In [28]:
```
mean_squared_error(y_test,ypred)
```

Out[28]:
```
40.35144969787304
```

In [29]:
```
sqrt(mean_squared_error(y_test,ypred))
```

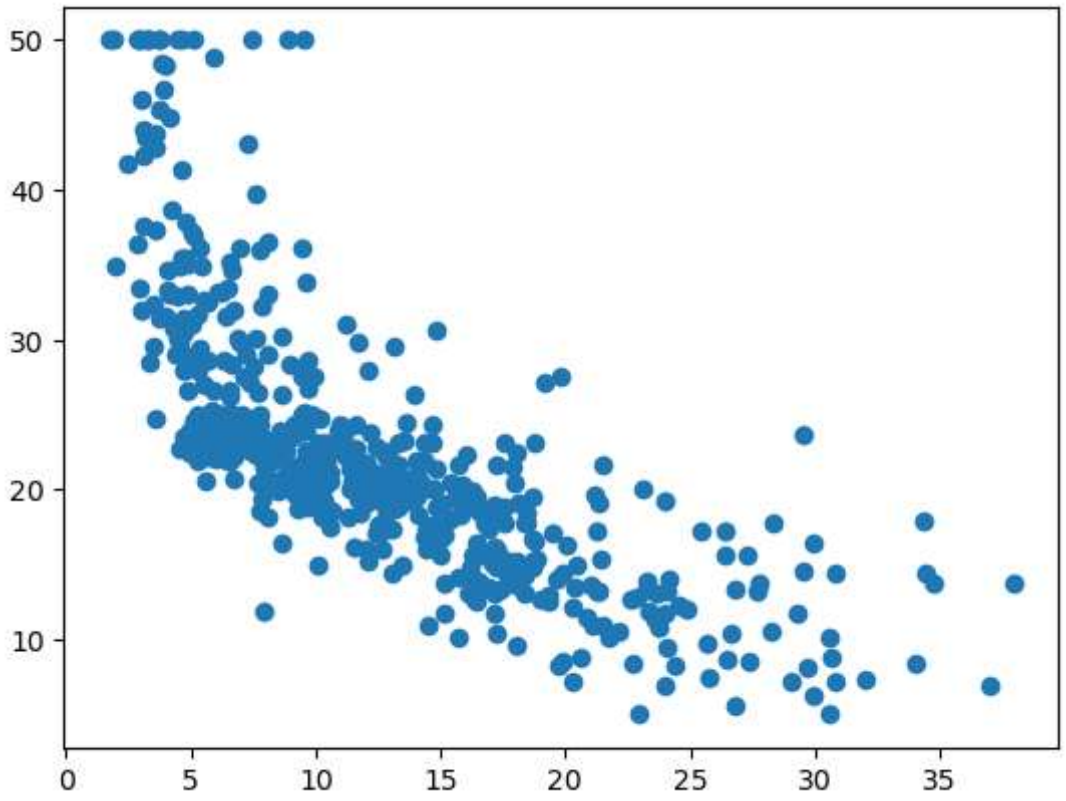
Out[29]:
```
6.352279094771658
```

In [30]:
```
r2_score(y_test,ypred)
```

Out[30]:
```
0.4584649934303069
```

In [31]:
```
plt.scatter(df.lstat,df.price)
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x2341711e850>
```



In [32]:
```
X = df[['lstat']]
y = df['price']
```

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
```

In [34]:
```
reg_model = LinearRegression()
reg_model.fit(X_train,y_train)
```

Out[34]: LinearRegression()

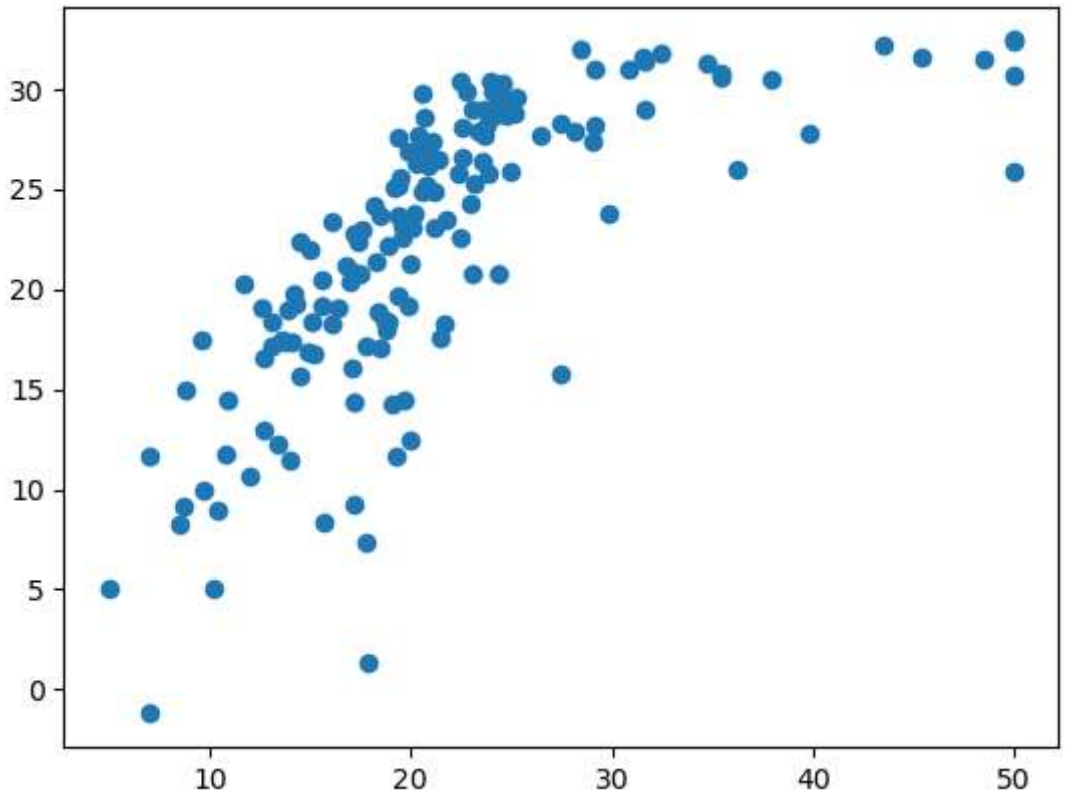
In [35]:
```
ypred = reg_model.predict(X_test)
```

In [36]:
```
reg_model.score(X_test,y_test)*100
```

Out[36]: 48.86979007906852

In [37]:
```
plt.scatter(y_test,ypred)
```

Out[37]: <matplotlib.collections.PathCollection at 0x23417329d00>



In [38]:
```
mean_absolute_error(y_test,ypred)
```

Out[38]: 4.75210051143785

In [39]:
```
mean_squared_error(y_test,ypred)
```

Out[39]: 38.09870218243471

In [40]:
```
sqrt(mean_squared_error(y_test,ypred))
```

Out[40]: 6.172414615240514

In [41]:
```
r2_score(y_test,ypred)*100
```

Out[41]: 48.86979007906852

In [42]:
```
X = df[['rm', 'lstat']]
y = df.price
```

In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
```

In [44]:
```
reg_model = LinearRegression()
reg_model.fit(X_train,y_train)
```

Out[44]: LinearRegression()

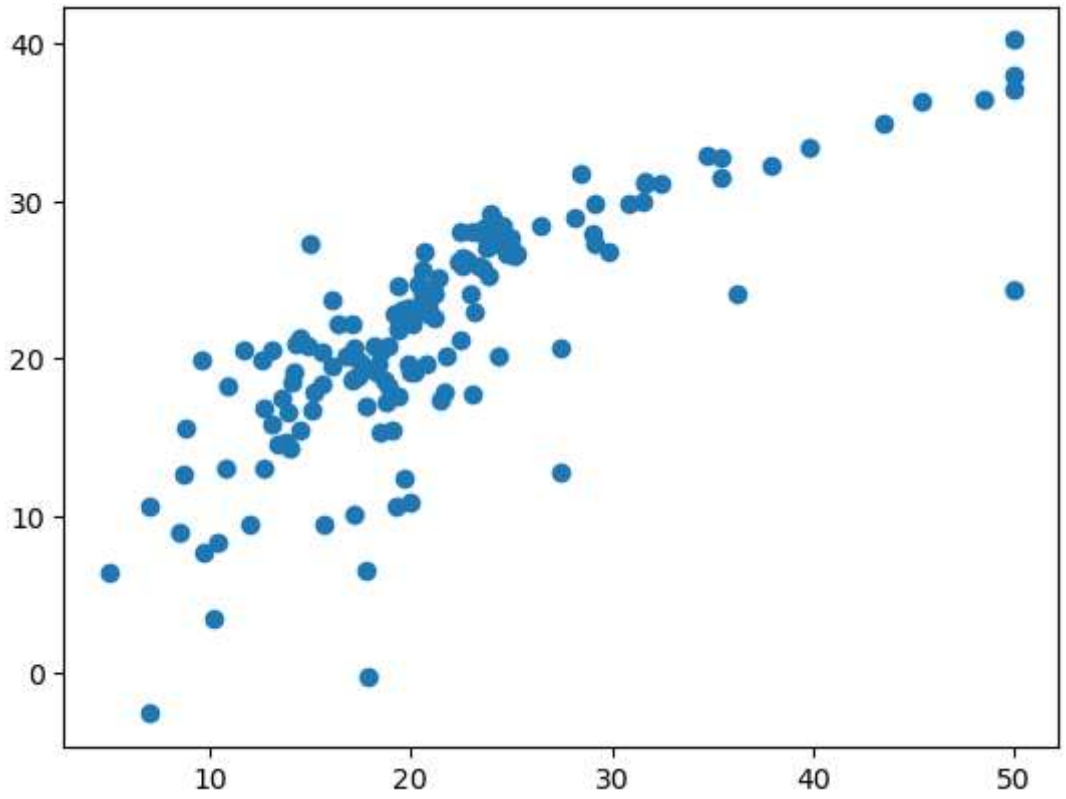
In [45]:
```
ypred = reg_model.predict(X_test)
```

In [46]:
```
reg_model.score(X_test,y_test)*100
```

Out[46]: 59.985184477155975

In [47]:
```
plt.scatter(y_test,ypred)
```

Out[47]:
```
<matplotlib.collections.PathCollection at 0x23417191f70>
```



In [48]:
```
mean_absolute_error(y_test,ypred)
```

Out[48]:
```
4.0880457454485155
```

In [49]:
```
mean_squared_error(y_test,ypred)
```

Out[49]:
```
29.816277731842458
```

In [50]:
```
r2_score(y_test,ypred)*100
```

Out[50]:
```
59.985184477155975
```

In [51]:
```
X = df.iloc[:,:-1]
y = df.price
```

In [52]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
```

In [53]:
```
reg_model = LinearRegression()
reg_model.fit(X_train,y_train)
```

Out[53]: LinearRegression()

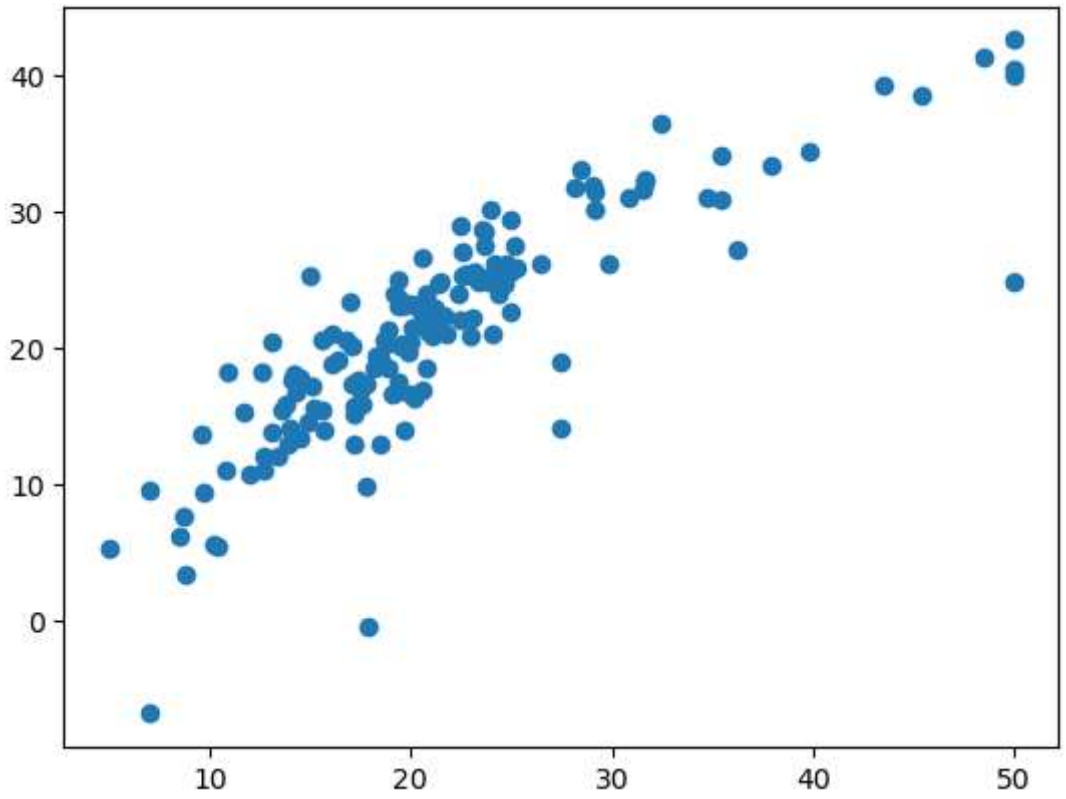
In [54]:
```
ypred = reg_model.predict(X_test)
```

In [55]:
```
reg_model.score(X_test,y_test)*100
```

Out[55]: 71.12260057484903

In [56]:
```
plt.scatter(y_test,ypred)
```

Out[56]: <matplotlib.collections.PathCollection at 0x234171ef7c0>



In [57]:
```
mean_absolute_error(y_test,ypred)
```

Out[57]: 3.1627098714574253

In [58]:
```
mean_squared_error(y_test,ypred)
```

Out[58]: 21.517444231177432

In [59]:
```
sqrt(mean_squared_error(y_test,ypred))
```

Out[59]: 4.6386899261728445

In [60]:
```
r2_score(y_test,ypred)*100
```

Out[60]: 71.12260057484903

In [ ]:
```

```