In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:

```
df=pd.read_csv("Boston.csv")
df
```

Out[6]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	39
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	39
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	39
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	39
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	39
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	39
502	503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	39
503	504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	39
504	505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	39
505	506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	39

506 rows × 15 columns

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
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 [8]:

```
target_feature="medv"
y=df[target_feature]   #output dependent
x=df.drop(target_feature,axis=1)   #Input Indepedent
```

In [9]:

```
x.head()
```

Out[9]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	blac
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.8
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.6
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9

In [10]:

```
y.head()
```

Out[10]:

```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: medv, dtype: float64
```

In [11]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [12]:

```
from sklearn.linear_model import LinearRegression
regression=LinearRegression()
regression.fit(x_train,y_train)
```

Out[12]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:

```
train_score=round(regression.score(x_train,y_train)*100,2)
print(train_score)
```

```
75.5
```

In [14]:

```
y_pred=regression.predict(x_test)
df1=pd.DataFrame({'Actual':y_test,'Predicted':y_pred,'Variance':y_test-y_pred})
df1
```

Out[14]:

	Actual	Predicted	Variance
337	18.5	18.591297	-0.091297
48	14.4	9.917991	4.482009
261	43.1	36.341328	6.758672
222	27.5	32.567436	-5.067436
320	23.8	24.311757	-0.511757
...	...	...	...
126	15.7	14.212018	1.487982
371	50.0	24.338774	25.661226
281	35.4	33.599213	1.800787
360	25.0	22.186173	2.813827
204	50.0	43.166925	6.833075

102 rows × 3 columns

In [15]:

```
from sklearn.metrics import r2_score
score=round(r2_score(y_test,y_pred)*100,2)
print('r_2 score',score)
```

```
r_2 score 68.6
```

In [16]:

```
from sklearn import metrics
print("Mean absolute error on test data of linear regression",metrics.mean_absolute_erro
print("Mean squared error on test data of linear regression",metrics.mean_squared_error(
print("Root mean squared error on test data of linear regression",np.sqrt(metrics.mean_s
```

```
Mean absolute error on test data of linear regression 3.8636366706139937
Mean squared error on test data of linear regression 31.21308173724438
Root mean squared error on test data of linear regression 5.58686689811421
7
```

In [ ]:

```

```