In [1]:
```
from sklearn.datasets import load_boston
```

In [2]:
```
boston = load_boston()
```

In [3]:
```
import pandas as pd
```

In [4]:
```
data = pd.DataFrame(boston.data, columns = boston.feature_names)
```

In [5]:
```
data["MEDV"] = pd.DataFrame(boston.target)
```

In [6]:
```
x = data["RM"]
y = data["MEDV"]
```

In [7]:
```
pd.DataFrame([x,y]).transpose().head()
```

Out[7]:

	RM	MEDV
0	6.575	24.0
1	6.421	21.6
2	7.185	34.7
3	6.998	33.4
4	7.147	36.2

In [8]:
```
from sklearn.linear_model import LinearRegression
```

In [9]:
```
model1 = LinearRegression()
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x, y, test_size = 0.2
```

In [14]:
```
print(x_train_1.shape)
```

```
(404,)
```

In [15]:
```
print(x_test_1.shape)
```

```
(102,)
```

In [19]:
```
linearRegression_train = LinearRegression()
```

In [22]:
```
type(x)
```

Out[22]:
```
pandas.core.series.Series
```

In [23]:
```
type(x_train_1)
```

Out[23]:
```
pandas.core.series.Series
```

In [24]:
```
x=pd.DataFrame(x)
y=pd.DataFrame(y)
```

In [25]:
```
x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x, y, test_size = 0.2
```

In [26]:
```
print(x_train_1.shape)
```

```
(404, 1)
```

In [27]:
```
print(x_test_1.shape)
```

```
(102, 1)
```

In [28]:
```
linearRegression_train.fit(x_train_1, y_train_1)
```

Out[28]:
```
LinearRegression()
```

In [29]:
```
yTestPredict = linearRegression_train.predict(x_test_1)
```

In [30]:
```
print(y_test_1)
```

```
     MEDV
242  22.2
23   14.5
216  23.3
501  22.4
480  23.0
..    ...
249  26.2
138  13.3
99   33.2
385   7.2
266  30.7

[102 rows x 1 columns]
```

In [31]:
```
print(yTestPredict)
```

```
[[23.11172454]
 [18.23798372]
 [18.908682  ]
 [25.21324582]
 [22.07437787]
 [30.69508642]
 [21.93129557]
 [17.96076176]
 [12.6309461 ]
 [28.34317112]
 [17.23640762]
 [24.26532558]
 [23.63934052]
 [18.75665705]
 [19.55255234]
 [19.74034786]
 [16.7624475 ]
 [25.05227823]
 [22.70036293]
 [17.96970441]
 [36.18586966]
 [18.42577924]
 [21.35896638]
 [20.48258729]
 [18.78348498]
 [21.79715592]
 [33.22585459]
 [27.93180951]
 [19.06070694]
 [20.52730051]
 [21.40367959]
 [29.80976469]
 [21.86869707]
 [21.84186914]
 [28.91550032]
 [20.75980925]
 [18.908682  ]
 [20.76875189]
 [21.90446764]
 [32.60881218]
 [20.27690649]
 [20.0354551 ]
 [16.48522555]
 [21.04597385]
 [ 3.25905549]
 [17.99653234]
 [11.1375246 ]
 [23.76453754]
 [10.10017793]
 [10.96761437]
 [25.49941042]
 [18.6225174 ]
 [23.02229811]
 [22.88815845]
```

```
[19.56149499]
[23.5320288 ]
[20.49152993]
[26.30424835]
[ 6.66620274]
[31.77714631]
[18.18432785]
[20.92971948]
[19.33792889]
[20.24113591]
[23.3889465 ]
[16.95918566]
[13.68617806]
[27.52939054]
[16.27954474]
[21.06385913]
[19.69563464]
[22.28900132]
[20.63461223]
[27.50256261]
[26.90340548]
[22.72719086]
[21.9581235 ]
[21.77032799]
[26.68878203]
[19.48101119]
[16.35108589]
[24.47100639]
[15.02757462]
[38.09959542]
[22.72719086]
[23.62145524]
[27.35948031]
[14.53572922]
[23.32634799]
[22.1816896 ]
[30.17641308]
[19.39158476]
[27.01071721]
[24.7392857 ]
[34.30791448]
[27.7261287 ]
[21.80609856]
[26.33107628]
[18.63146004]
[32.5730416 ]
[13.44472668]
[28.97809882]]
```

In [33]:
```
from sklearn.metrics import mean_squared_error
```

In [34]:
```
import numpy as np
```

In [37]:
```
yTrainPredict = linearRegression_train.predict(x_test_1)
```

In [38]:
```
np.sqrt(mean_squared_error(y_test_1, yTrainPredict))
```

Out[38]: 5.5064341682109985

In [39]:
```
linearRegression_train.score(x_test_1,y_test_1)
```

Out[39]: 0.5743934390546408

In [40]:
```
from sklearn.linear_model import Ridge
```

In [41]:
```
ridge_reg = Ridge(alpha=1)
```

In [42]:
```
ridge_reg.fit(x,y)
```

Out[42]: Ridge(alpha=1)

In [45]:
```
yTestPredict = ridge_reg.predict(x_test_1)
```

In [54]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[54]: 5.522934223326744

In [47]:
```
ridge_reg2 = Ridge(alpha=0.5)
```

In [48]:
```
ridge_reg2.fit(x_train_1, y_train_1)
```

Out[48]: Ridge(alpha=0.5)

In [52]:
```
yTestPredict = ridge_reg2.predict(x_test_1)
```

In [50]:
```
ridge_reg3 = Ridge(alpha=5)
```

In [51]:
```
ridge_reg3.fit(x_train_1, y_train_1)
```

Out[51]: Ridge(alpha=5)

In [53]:
```
yTestPredict = ridge_reg3.predict(x_test_1)
```

In [55]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[55]: 5.522934223326744

In [56]:
```
ridge_reg4 = Ridge(alpha=50)
```

In [57]:
```
ridge_reg4.fit(x_train_1, y_train_1)
```

Out[57]:
```
Ridge(alpha=50)
```

In [58]:
```
yTestPredict = ridge_reg4.predict(x_test_1)
```

In [59]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[59]:
```
5.741623366591704
```

In [60]:
```
ridge_reg5 = Ridge(alpha=10)
```

In [61]:
```
ridge_reg5.fit(x_train_1, y_train_1)
```

Out[61]:
```
Ridge(alpha=10)
```

In [62]:
```
yTestPredict = ridge_reg5.predict(x_test_1)
```

In [63]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[63]:
```
5.541852072732345
```

In [65]:
```
from sklearn.linear_model import Lasso
```

In [66]:
```
lasso_reg = Lasso(alpha=0.1)
```

In [67]:
```
lasso_reg.fit(x_train_1, y_train_1)
```

Out[67]:
```
Lasso(alpha=0.1)
```

In [68]:
```
yTestPredict = lasso_reg.predict(x_test_1)
```

In [69]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[69]:
```
5.521586452694054
```

In [70]:
```
lasso_reg1 = Lasso(alpha=1)
```

In [72]:
```
lasso_reg1.fit(x_train_1, y_train_1)
```

Out[72]:
```
Lasso(alpha=1)
```

In [73]:
```
yTestPredict = lasso_reg1.predict(x_test_1)
```

In [74]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[74]: 5.785466416724749

In [75]:
```
lasso_reg2 = Lasso(alpha=10)
```

In [76]:
```
lasso_reg2.fit(x_train_1, y_train_1)
```

Out[76]: Lasso(alpha=10)

In [77]:
```
yTestPredict = lasso_reg2.predict(x_test_1)
```

In [78]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[78]: 8.445799622846943

In [79]:
```
lasso_reg3 = Lasso(alpha=5)
```

In [80]:
```
lasso_reg3.fit(x_train_1, y_train_1)
```

Out[80]: Lasso(alpha=5)

In [81]:
```
yTestPredict = lasso_reg3.predict(x_test_1)
```

In [82]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[82]: 8.445799622846943

In [83]:
```
from sklearn.linear_model import ElasticNet
```

In [86]:
```
elasticNet_reg = ElasticNet(alpha=0.1, l1_ratio=0.5)
```

In [87]:
```
elasticNet_reg.fit(x_train_1, y_train_1)
```

Out[87]: ElasticNet(alpha=0.1)

In [88]:
```
yTestPredict = elasticNet_reg.predict(x_test_1)
```

In [89]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[89]: 5.598200157334269

In [90]:
```
elasticNet_reg1 = ElasticNet(alpha=1, l1_ratio=0.5)
```

In [91]:
```
elasticNet_reg1.fit(x_train_1, y_train_1)
```

Out[91]:
```
ElasticNet(alpha=1)
```

In [92]:
```
yTestPredict = elasticNet_reg1.predict(x_test_1)
```

In [93]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[93]:
```
6.66737831783719
```

In [94]:
```
elasticNet_reg2 = ElasticNet(alpha=10, l1_ratio=0.5)
```

In [95]:
```
elasticNet_reg2.fit(x_train_1, y_train_1)
```

Out[95]:
```
ElasticNet(alpha=10)
```

In [96]:
```
yTestPredict = elasticNet_reg2.predict(x_test_1)
```

In [97]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[97]:
```
8.445799622846943
```

In [98]:
```
elasticNet_reg3 = ElasticNet(alpha=0.05, l1_ratio=0.5)
```

In [99]:
```
elasticNet_reg3.fit(x_train_1, y_train_1)
```

Out[99]:
```
ElasticNet(alpha=0.05)
```

In [100]:
```
yTestPredict = elasticNet_reg3.predict(x_test_1)
```

In [101]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[101]:
```
5.547080057864505
```

In [102]:
```
elasticNet_reg4 = ElasticNet(alpha=0.1, l1_ratio=0.3)
```

In [103]:
```
elasticNet_reg4.fit(x_train_1, y_train_1)
```

Out[103]:
```
ElasticNet(alpha=0.1, l1_ratio=0.3)
```

In [105]:
```
yTestPredict = elasticNet_reg4.predict(x_test_1)
```

In [106]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[106]:
```
5.633533386042302
```

In [107]:
```
elasticNet_reg5 = ElasticNet(alpha=0.1, l1_ratio=0.7)
```

In [108]:
```
elasticNet_reg5.fit(x_train_1, y_train_1)
```

Out[108]:
```
ElasticNet(alpha=0.1, l1_ratio=0.7)
```

In [109]:
```
yTestPredict = elasticNet_reg5.predict(x_test_1)
```

In [110]:
```
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[110]:
```
5.5650641025410525
```

In [ ]:
```

```