In [92]:
```
import numpy as np
```

In [93]:
```
import pandas as pd
```

In [94]:
```
from sklearn.model_selection import train_test_split
```

In [147]:
```
from sklearn.linear_model import LinearRegression
```

In [148]:
```
from sklearn.metrics import mean_squared_error
```

In [162]:
```
from sklearn.impute import SimpleImputer
```

In [163]:
```
df = pd.read_csv("/home/comp/Desktop/BostonHousing.csv")
```

In [164]:
```
df
```

Out[164]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88

506 rows × 14 columns

In [165]:
```
df.columns
```

Out[165]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'r
ad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

In [166]:
```
x = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'ra
       'ptratio', 'b', 'lstat']]
```

In [171]:
```
x
```

Out[171]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88

506 rows × 13 columns

In [188]:
```
y = df['medv']
```

In [189]:
```
y
```

Out[189]:
```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: medv, Length: 506, dtype: float64
```

In [190]:
```
x_train,x_test, y_train, y_test = train_test_split(x,y,test_size = 0.
```

In [191]:
```
imputer = SimpleImputer(strategy='mean')
```

In [192]:
```
# Fit the imputer on the training data and transform it
x_train_imputed = imputer.fit_transform(x_train)
```

In [193]:
```

# Use the same imputer to transform the test data
x_test_imputed = imputer.transform(x_test)
```

In [194]:
```
# Now, train your model using the imputed data
model = LinearRegression()
model.fit(x_train_imputed, y_train)
```

Out[194]:
```
LinearRegression()
```

In [196]:
```
y_pred = model.predict(x_test_imputed)
```

In [197]:
```
y_pred
```

Out[197]:
```
array([28.82027076, 36.00002442, 15.08140519, 25.22184891, 18.87765
177,
       23.20437451, 17.58371024, 14.28385752, 23.05811861, 20.60284
153,
       24.78920462, 18.6822581 , -6.96862106, 21.82823033, 19.20416
584,
       26.27505907, 20.54740844,  5.66344926, 40.40968977, 17.64201
074,
       27.30792417, 30.0385143 , 11.13295576, 24.08674002, 17.89533
823,
       15.80395106, 22.94307302, 14.25445306, 22.26369681, 19.23442
83 ,
       22.25894487, 25.22688853, 25.67286592, 18.00872966, 16.70170
645,
       17.13523986, 31.17716724, 20.16724444, 23.71222824, 24.77802
899,
       13.93277633, 31.98011895, 42.52475588, 17.44386887, 27.12788
979,
       17.08142825, 13.87749849, 26.04848947, 20.37004142, 29.96818
018,
       21.36836548, 34.31320317, 15.86249572, 26.14644831, 39.49223
334,
       22.84555477, 18.95127793, 32.67937939, 25.00013811, 12.91574
705,
       20.8529844 , 30.54102842, 31.58870201, 15.90793456, 20.52366
577,
       16.509932  , 20.49568824, 25.99537163, 30.63125739, 11.43556
869,
       20.52716195, 27.56153658, 10.85214775, 15.98642822, 23.86536
671,
        5.66081094, 21.45315872, 41.27025577, 18.55300504,  9.09002
982,
       20.9759154 , 13.05851381, 21.01473768,  9.34912726, 23.12680
355,
       31.7893813 , 19.0980852 , 25.57551598, 29.14317814, 20.16101
848,
       25.58189606,  5.20730852, 20.16003364, 15.08613484, 12.90801
423,
       20.80149182, 24.68652099, -0.76657858, 13.33727414, 15.61464
758,
       22.19767843, 24.57259482, 10.77857391, 19.48987117, 23.23918
749,
       11.76910855, 18.35440581, 25.42242663, 20.88083146, 24.10167
328,
        7.36222362, 19.14832319, 21.92747023, 27.38678772, 32.48574
195,
       14.87418361, 35.01412703, 12.85151712, 20.81571923, 28.41804
578,
       15.6758695 , 24.6678367 ,  3.28911585, 23.79060245, 25.72372
509,
       23.03785429, 24.7447737 ])
```

In [200]:
```
model.score(x_train_imputed,y_train)
```

Out[200]:
```
0.7474263582831636
```

In [202]:
```
model.score(x_test_imputed,y_test)
```

Out[202]: 0.6833980539496451

In [204]:
```
mean_squared_error(y_test,y_pred)
```

Out[204]: 22.170729957295116

In [205]:
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[205]: 4.70858046095584