Rudra_Ronit_HW3_Practicum_Q3

November 9, 2016

1 Question 3

Load the Boston dataset (sklearn.datasets.load_boston()) into Python using a Pandas dataframe. Perform a K-Means analysis on unscaled data, with the number of clusters ranging from 2 to 6. Provide the Silhouette score to justify which value of k is optimal. What information do the values of Homogeneity/Completeness provide as well? Calculate the mean values for all features in each cluster for the optimal clustering - how do these values differ from the centroid coordinates?

2 Answer

Load required modules

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn import metrics
        from sklearn.datasets import load_boston
        from sklearn.cluster import KMeans
        import pandas as pd
```

Loading the data onto an object

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

Converting the data into a dataframe

```
In [3]: boston = pd.DataFrame(data.data, columns = data.feature_names)
        boston['target'] = data.target
```

```
In [4]: boston
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0

7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0
10	0.22489	12.5	7.87	0.0	0.524	6.377	94.3	6.3467	5.0	311.0
11	0.11747	12.5	7.87	0.0	0.524	6.009	82.9	6.2267	5.0	311.0
12	0.09378	12.5	7.87	0.0	0.524	5.889	39.0	5.4509	5.0	311.0
13	0.62976	0.0	8.14	0.0	0.538	5.949	61.8	4.7075	4.0	307.0
14	0.63796	0.0	8.14	0.0	0.538	6.096	84.5	4.4619	4.0	307.0
15	0.62739	0.0	8.14	0.0	0.538	5.834	56.5	4.4986	4.0	307.0
16	1.05393	0.0	8.14	0.0	0.538	5.935	29.3	4.4986	4.0	307.0
17	0.78420	0.0	8.14	0.0	0.538	5.990	81.7	4.2579	4.0	307.0
18	0.80271	0.0	8.14	0.0	0.538	5.456	36.6	3.7965	4.0	307.0
19	0.72580	0.0	8.14	0.0	0.538	5.727	69.5	3.7965	4.0	307.0
20	1.25179	0.0	8.14	0.0	0.538	5.570	98.1	3.7979	4.0	307.0
21	0.85204	0.0	8.14	0.0	0.538	5.965	89.2	4.0123	4.0	307.0
22	1.23247	0.0	8.14	0.0	0.538	6.142	91.7	3.9769	4.0	307.0
23	0.98843	0.0	8.14	0.0	0.538	5.813	100.0	4.0952	4.0	307.0
24	0.75026	0.0	8.14	0.0	0.538	5.924	94.1	4.3996	4.0	307.0
25	0.84054	0.0	8.14	0.0	0.538	5.599	85.7	4.4546	4.0	307.0
26	0.67191	0.0	8.14	0.0	0.538	5.813	90.3	4.6820	4.0	307.0
27	0.95577	0.0	8.14	0.0	0.538	6.047	88.8	4.4534	4.0	307.0
28	0.77299	0.0	8.14	0.0	0.538	6.495	94.4	4.4547	4.0	307.0
29	1.00245	0.0	8.14	0.0	0.538	6.674	87.3	4.2390	4.0	307.0
..	...	...	...	...	...	...	...	...	...	...
476	4.87141	0.0	18.10	0.0	0.614	6.484	93.6	2.3053	24.0	666.0
477	15.02340	0.0	18.10	0.0	0.614	5.304	97.3	2.1007	24.0	666.0
478	10.23300	0.0	18.10	0.0	0.614	6.185	96.7	2.1705	24.0	666.0
479	14.33370	0.0	18.10	0.0	0.614	6.229	88.0	1.9512	24.0	666.0
480	5.82401	0.0	18.10	0.0	0.532	6.242	64.7	3.4242	24.0	666.0
481	5.70818	0.0	18.10	0.0	0.532	6.750	74.9	3.3317	24.0	666.0
482	5.73116	0.0	18.10	0.0	0.532	7.061	77.0	3.4106	24.0	666.0
483	2.81838	0.0	18.10	0.0	0.532	5.762	40.3	4.0983	24.0	666.0
484	2.37857	0.0	18.10	0.0	0.583	5.871	41.9	3.7240	24.0	666.0
485	3.67367	0.0	18.10	0.0	0.583	6.312	51.9	3.9917	24.0	666.0
486	5.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0
487	4.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0
488	0.15086	0.0	27.74	0.0	0.609	5.454	92.7	1.8209	4.0	711.0
489	0.18337	0.0	27.74	0.0	0.609	5.414	98.3	1.7554	4.0	711.0
490	0.20746	0.0	27.74	0.0	0.609	5.093	98.0	1.8226	4.0	711.0
491	0.10574	0.0	27.74	0.0	0.609	5.983	98.8	1.8681	4.0	711.0
492	0.11132	0.0	27.74	0.0	0.609	5.983	83.5	2.1099	4.0	711.0
493	0.17331	0.0	9.69	0.0	0.585	5.707	54.0	2.3817	6.0	391.0
494	0.27957	0.0	9.69	0.0	0.585	5.926	42.6	2.3817	6.0	391.0
495	0.17899	0.0	9.69	0.0	0.585	5.670	28.8	2.7986	6.0	391.0
496	0.28960	0.0	9.69	0.0	0.585	5.390	72.9	2.7986	6.0	391.0
497	0.26838	0.0	9.69	0.0	0.585	5.794	70.6	2.8927	6.0	391.0
498	0.23912	0.0	9.69	0.0	0.585	6.019	65.3	2.4091	6.0	391.0
499	0.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6.0	391.0

500	0.22438	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6.0	391.0
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0

	PTRATIO	B	LSTAT	target
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2
5	18.7	394.12	5.21	28.7
6	15.2	395.60	12.43	22.9
7	15.2	396.90	19.15	27.1
8	15.2	386.63	29.93	16.5
9	15.2	386.71	17.10	18.9
10	15.2	392.52	20.45	15.0
11	15.2	396.90	13.27	18.9
12	15.2	390.50	15.71	21.7
13	21.0	396.90	8.26	20.4
14	21.0	380.02	10.26	18.2
15	21.0	395.62	8.47	19.9
16	21.0	386.85	6.58	23.1
17	21.0	386.75	14.67	17.5
18	21.0	288.99	11.69	20.2
19	21.0	390.95	11.28	18.2
20	21.0	376.57	21.02	13.6
21	21.0	392.53	13.83	19.6
22	21.0	396.90	18.72	15.2
23	21.0	394.54	19.88	14.5
24	21.0	394.33	16.30	15.6
25	21.0	303.42	16.51	13.9
26	21.0	376.88	14.81	16.6
27	21.0	306.38	17.28	14.8
28	21.0	387.94	12.80	18.4
29	21.0	380.23	11.98	21.0
..	...	...	...	...
476	20.2	396.21	18.68	16.7
477	20.2	349.48	24.91	12.0
478	20.2	379.70	18.03	14.6
479	20.2	383.32	13.11	21.4
480	20.2	396.90	10.74	23.0
481	20.2	393.07	7.74	23.7
482	20.2	395.28	7.01	25.0
483	20.2	392.92	10.42	21.8
484	20.2	370.73	13.34	20.6

```
485     20.2  388.62  10.58    21.2
486     20.2  392.68  14.98    19.1
487     20.2  388.22  11.45    20.6
488     20.1  395.09  18.06    15.2
489     20.1  344.05  23.97     7.0
490     20.1  318.43  29.68     8.1
491     20.1  390.11  18.07    13.6
492     20.1  396.90  13.35    20.1
493     19.2  396.90  12.01    21.8
494     19.2  396.90  13.59    24.5
495     19.2  393.29  17.60    23.1
496     19.2  396.90  21.14    19.7
497     19.2  396.90  14.10    18.3
498     19.2  396.90  12.92    21.2
499     19.2  395.77  15.10    17.5
500     19.2  396.90  14.33    16.8
501     21.0  391.99   9.67    22.4
502     21.0  396.90   9.08    20.6
503     21.0  396.90   5.64    23.9
504     21.0  393.45   6.48    22.0
505     21.0  396.90   7.88    11.9

[506 rows x 14 columns]
```

Saving the data into separate variables for attributes and targets.

```
In [5]: n_samples, n_features = boston.iloc[:,0:13].shape
        boston_data = boston.iloc[:,0:13]
        labels = boston['target']
```

Applying K-Means Clustering on the data with clusters ranging from 2 to 6. The "k_means" list stores all the 5 models.

```
In [6]: k_means = []
        for cluster_size in range(2,7):
            k_means.append(KMeans(n_clusters = cluster_size).fit(boston_data))
```

Calculating the Silhouette Score of all the models using the euclidean distance metric. Saving the Silhouette Score to a list.

```
In [7]: maxi = []
        for model in k_means:
            print(metrics.silhouette_score(boston_data,model.labels_,
                                           metric='euclidean'))
            maxi.append(metrics.silhouette_score(boston_data,model.labels_,
                                                 metric='euclidean'))
```

```
0.691398118833
0.723403034161
```

```
0.568219170853
0.570738665513
0.501258930507
```

Printing out the Completeness Score

```
In [8]: for model in k_means:
            print(metrics.completeness_score(labels,model.labels_))
```

```
0.627029136728
0.639770837023
0.601684007444
0.620034051928
0.629506604287
```

Printing out the Homogeneity Score

```
In [9]: for model in k_means:
            print(metrics.homogeneity_score(labels,model.labels_))
```

```
0.070186194715
0.0921583560761
0.135137885887
0.148658835525
0.187370799835
```

Saving the model with highest Silhouette Score

```
In [10]: best_fit = k_means[int(np.where(maxi >= max(maxi))[0])]
```

Displaying Cluster Centers for the best model

```
In [11]: best_fit.cluster_centers_
```

```
Out[11]: array([[  3.74992678e-01,   1.57103825e+01,   8.35953552e+00,
                   7.10382514e-02,   5.09862568e-01,   6.39165301e+00,
                   6.04133880e+01,   4.46074481e+00,   4.45081967e+00,
                   3.11232240e+02,   1.78177596e+01,   3.83489809e+02,
                   1.03886612e+01],
                [  1.09105113e+01,   5.32907052e-15,   1.85725490e+01,
                   7.84313725e-02,   6.71225490e-01,   5.98226471e+00,
                   8.99137255e+01,   2.07716373e+00,   2.30196078e+01,
                   6.68205882e+02,   2.01950980e+01,   3.71803039e+02,
                   1.78740196e+01],
                [  1.49558803e+01,  -5.32907052e-15,   1.79268421e+01,
                   2.63157895e-02,   6.73710526e-01,   6.06550000e+00,
                   8.99052632e+01,   1.99442895e+00,   2.25000000e+01,
                   6.44736842e+02,   1.99289474e+01,   5.77863158e+01,
                   2.04486842e+01]])
```

Displaying assigned clusters of each data point based on the best model

```
In [12]: best_fit.predict(boston_data)
```

```
Out[12]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 1,
                1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Adding Predicted Clusters to the dataset

```
In [13]: boston['class'] = best_fit.predict(boston_data)
```

```
In [14]: boston
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.
10	0.22489	12.5	7.87	0.0	0.524	6.377	94.3	6.3467	5.0	311.
11	0.11747	12.5	7.87	0.0	0.524	6.009	82.9	6.2267	5.0	311.
12	0.09378	12.5	7.87	0.0	0.524	5.889	39.0	5.4509	5.0	311.
13	0.62976	0.0	8.14	0.0	0.538	5.949	61.8	4.7075	4.0	307.
14	0.63796	0.0	8.14	0.0	0.538	6.096	84.5	4.4619	4.0	307.

15	0.62739	0.0	8.14	0.0	0.538	5.834	56.5	4.4986	4.0	307.0
16	1.05393	0.0	8.14	0.0	0.538	5.935	29.3	4.4986	4.0	307.0
17	0.78420	0.0	8.14	0.0	0.538	5.990	81.7	4.2579	4.0	307.0
18	0.80271	0.0	8.14	0.0	0.538	5.456	36.6	3.7965	4.0	307.0
19	0.72580	0.0	8.14	0.0	0.538	5.727	69.5	3.7965	4.0	307.0
20	1.25179	0.0	8.14	0.0	0.538	5.570	98.1	3.7979	4.0	307.0
21	0.85204	0.0	8.14	0.0	0.538	5.965	89.2	4.0123	4.0	307.0
22	1.23247	0.0	8.14	0.0	0.538	6.142	91.7	3.9769	4.0	307.0
23	0.98843	0.0	8.14	0.0	0.538	5.813	100.0	4.0952	4.0	307.0
24	0.75026	0.0	8.14	0.0	0.538	5.924	94.1	4.3996	4.0	307.0
25	0.84054	0.0	8.14	0.0	0.538	5.599	85.7	4.4546	4.0	307.0
26	0.67191	0.0	8.14	0.0	0.538	5.813	90.3	4.6820	4.0	307.0
27	0.95577	0.0	8.14	0.0	0.538	6.047	88.8	4.4534	4.0	307.0
28	0.77299	0.0	8.14	0.0	0.538	6.495	94.4	4.4547	4.0	307.0
29	1.00245	0.0	8.14	0.0	0.538	6.674	87.3	4.2390	4.0	307.0
..	...	...	...	...	...	...	...	...	...	...
476	4.87141	0.0	18.10	0.0	0.614	6.484	93.6	2.3053	24.0	666.0
477	15.02340	0.0	18.10	0.0	0.614	5.304	97.3	2.1007	24.0	666.0
478	10.23300	0.0	18.10	0.0	0.614	6.185	96.7	2.1705	24.0	666.0
479	14.33370	0.0	18.10	0.0	0.614	6.229	88.0	1.9512	24.0	666.0
480	5.82401	0.0	18.10	0.0	0.532	6.242	64.7	3.4242	24.0	666.0
481	5.70818	0.0	18.10	0.0	0.532	6.750	74.9	3.3317	24.0	666.0
482	5.73116	0.0	18.10	0.0	0.532	7.061	77.0	3.4106	24.0	666.0
483	2.81838	0.0	18.10	0.0	0.532	5.762	40.3	4.0983	24.0	666.0
484	2.37857	0.0	18.10	0.0	0.583	5.871	41.9	3.7240	24.0	666.0
485	3.67367	0.0	18.10	0.0	0.583	6.312	51.9	3.9917	24.0	666.0
486	5.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0
487	4.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0
488	0.15086	0.0	27.74	0.0	0.609	5.454	92.7	1.8209	4.0	711.0
489	0.18337	0.0	27.74	0.0	0.609	5.414	98.3	1.7554	4.0	711.0
490	0.20746	0.0	27.74	0.0	0.609	5.093	98.0	1.8226	4.0	711.0
491	0.10574	0.0	27.74	0.0	0.609	5.983	98.8	1.8681	4.0	711.0
492	0.11132	0.0	27.74	0.0	0.609	5.983	83.5	2.1099	4.0	711.0
493	0.17331	0.0	9.69	0.0	0.585	5.707	54.0	2.3817	6.0	391.0
494	0.27957	0.0	9.69	0.0	0.585	5.926	42.6	2.3817	6.0	391.0
495	0.17899	0.0	9.69	0.0	0.585	5.670	28.8	2.7986	6.0	391.0
496	0.28960	0.0	9.69	0.0	0.585	5.390	72.9	2.7986	6.0	391.0
497	0.26838	0.0	9.69	0.0	0.585	5.794	70.6	2.8927	6.0	391.0
498	0.23912	0.0	9.69	0.0	0.585	6.019	65.3	2.4091	6.0	391.0
499	0.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6.0	391.0
500	0.22438	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6.0	391.0
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0

	PTRATIO	B	LSTAT	target	class

0	15.3	396.90	4.98	24.0	0
1	17.8	396.90	9.14	21.6	0
2	17.8	392.83	4.03	34.7	0
3	18.7	394.63	2.94	33.4	0
4	18.7	396.90	5.33	36.2	0
5	18.7	394.12	5.21	28.7	0
6	15.2	395.60	12.43	22.9	0
7	15.2	396.90	19.15	27.1	0
8	15.2	386.63	29.93	16.5	0
9	15.2	386.71	17.10	18.9	0
10	15.2	392.52	20.45	15.0	0
11	15.2	396.90	13.27	18.9	0
12	15.2	390.50	15.71	21.7	0
13	21.0	396.90	8.26	20.4	0
14	21.0	380.02	10.26	18.2	0
15	21.0	395.62	8.47	19.9	0
16	21.0	386.85	6.58	23.1	0
17	21.0	386.75	14.67	17.5	0
18	21.0	288.99	11.69	20.2	0
19	21.0	390.95	11.28	18.2	0
20	21.0	376.57	21.02	13.6	0
21	21.0	392.53	13.83	19.6	0
22	21.0	396.90	18.72	15.2	0
23	21.0	394.54	19.88	14.5	0
24	21.0	394.33	16.30	15.6	0
25	21.0	303.42	16.51	13.9	0
26	21.0	376.88	14.81	16.6	0
27	21.0	306.38	17.28	14.8	0
28	21.0	387.94	12.80	18.4	0
29	21.0	380.23	11.98	21.0	0
..	...	...	...	...	...
476	20.2	396.21	18.68	16.7	1
477	20.2	349.48	24.91	12.0	1
478	20.2	379.70	18.03	14.6	1
479	20.2	383.32	13.11	21.4	1
480	20.2	396.90	10.74	23.0	1
481	20.2	393.07	7.74	23.7	1
482	20.2	395.28	7.01	25.0	1
483	20.2	392.92	10.42	21.8	1
484	20.2	370.73	13.34	20.6	1
485	20.2	388.62	10.58	21.2	1
486	20.2	392.68	14.98	19.1	1
487	20.2	388.22	11.45	20.6	1
488	20.1	395.09	18.06	15.2	1
489	20.1	344.05	23.97	7.0	1
490	20.1	318.43	29.68	8.1	1
491	20.1	390.11	18.07	13.6	1
492	20.1	396.90	13.35	20.1	1

```
493     19.2  396.90  12.01    21.8      0
494     19.2  396.90  13.59    24.5      0
495     19.2  393.29  17.60    23.1      0
496     19.2  396.90  21.14    19.7      0
497     19.2  396.90  14.10    18.3      0
498     19.2  396.90  12.92    21.2      0
499     19.2  395.77  15.10    17.5      0
500     19.2  396.90  14.33    16.8      0
501     21.0  391.99   9.67    22.4      0
502     21.0  396.90   9.08    20.6      0
503     21.0  396.90   5.64    23.9      0
504     21.0  393.45   6.48    22.0      0
505     21.0  396.90   7.88    11.9      0

[506 rows x 15 columns]
```

For each cluster, finding mean of each attribute

```
In [15]: boston.groupby('class').mean()
```

```
Out[15]:              CRIM         ZN      INDUS      CHAS       NOX         RM  \
         class
         0        0.374993  15.710383   8.359536  0.071038  0.509863  6.391653
         1       10.910511   0.000000  18.572549  0.078431  0.671225  5.982265
         2       14.955880   0.000000  17.926842  0.026316  0.673711  6.065500

                      AGE       DIS        RAD         TAX    PTRATIO           B
         class
         0      60.413388  4.460745   4.450820  311.232240  17.817760  383.489809
         1      89.913725  2.077164  23.019608  668.205882  20.195098  371.803039
         2      89.905263  1.994429  22.500000  644.736842  19.928947   57.786316

                   LSTAT     target
         class
         0      10.388661  24.931694
         1      17.874020  17.429412
         2      20.448684  13.126316
```

3 Inferences

- Model with **k=3** is the best fit as it has the highest silhouette score of **0.7234**
- *Homogeinity* gives a measure of how much each cluster contains members of a single class and would suggest amount of separation between clusters. A high score would mean that most classes got their own separate clusters.
- *Completeness* gives a measure of spread of classes among clusters. A high score is desirable as it would mean that objects of the same class fall into the same cluster.

- The best fit model has the highest homogeinity and completeness score as well.
- After calculating the mean values of each attribute for each cluster, it was observed that the centroid coordinates are the same as the means which was to be expected as the K-means algorithm updates the centroid coordinates by calculating the means of the datapoints falling into the clusters.