Assignment 3

60-473/574

Dr. Rueda

Joel Rorseth

November 20, 2017

For this assignment, I have utilized several clustering algorithms from the *Scikit* library, specifically the *KMeans* and *GaussianMixture* clustering models from the *sklearn.cluster* and *sklearn.mixture* classes respectively.

To accomplish successful and efficient clustering, KMeans and GaussianMixture objects are instantiated to determine clusters by means of the K-Means and Expectation Maximization clustering algorithms. Both clustering models are created whilst specifying the value of *k* in their respective constructor. This is seen below in Figure 1.1**,** where the KMeans algorithm takes a parameter specifying the number of clusters (*k*), with GaussianMixture similarly accepting *k* in an equivalent manner via its parameterized number of components. After both objects request identification of *two* clusters, they are fit to the *unlabeled* sample data, and as such appease their underlying unsupervised learning algorithms. At this point, the clustering models have now been established, determined *k* clusters, and possess the ability to categorize unlabeled samples *by cluster*.

```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


# Create a KMeans clustering object where k=2
def kmeans(samples):
    return KMeans(n_clusters=2).fit(samples)


# Create an Expectation Maximization object that uses Gaussian models
def em(samples):
    return GaussianMixture(n_components=2).fit(samples)
```

Figure 1.1

To illustrate the performance and clustering results of the KMeans and Expectation Maximization algorithms, a graph is rendered in the script for question two. As seen in Figure 1.2, the clustering model objects previously instantiated are fed the original unlabeled sample dataset, and asked to predict the cluster of each sample. The returned array contains cluster predictions, where the prediction at index *i* corresponds to the sample at index *i* in the input dataset.

```
# Instantiate classifier, calculate and print efficiency
kmeans_classifier = kmeans(samples)
em_classifier = em(samples)

# Predict cluster for each sample (k=2 derived 2 classes essentially)
kmeans_pred = kmeans_classifier.fit_predict(samples)
em_pred = em_classifier.predict(samples)

# Plot the clustered samples

graph_samples("K-Means (K=2) on " + filename, samples, labels, kmeans_pred)
graph_samples("EM (K=2) on " + filename, samples, labels, em_pred)
```

Figure 1.2

A simple Python plotting library, *matplotlib.pyplot*, is then used to plot the sample dataset in its native two dimensional (feature) space.

```
# Graph samples
def graph_samples(name, samples, given_labels, clustered_labels):

    for i in range(0, len(samples)):
        color = 'b' if given_labels[i] == 1 else 'r'
        marker = '>' if clustered_labels[i] == 1 else 'o'
        plt.scatter(samples[i, 0], samples[i, 1], c=color, marker=marker)

    plt.title(name)
    plt.show()
```

Figure 1.3

As evident in Figure 1.3, the plotting procedure utilizes the original unlabeled sample dataset, original dataset labels, and the predicted clusters of each sample in the original dataset. Each sample from the original dataset, belonging to one of two distinct classes, is plotted as a blue or red point if originally labeled as being 1 or 2 respectively. To distinguish between determined clusters, a unique point shape is given to samples clustered into cluster 1 and 2 respectively. The resulting plot for each dataset over K-Means and Expectation Maximization is shown below in Figure 2.1.

```
# Determine the Calinski-Harabaz (CH) Index
def ch_index(samples, clustered_labels):
    return calinski_harabaz_score(samples, clustered_labels)
```

Figure 1.4

To compare the accuracy and general performance of both clustering models, the CH Index was calculated using *sklearn.metrics calinski_harabaz_score* function, as seen above in Figure 1.4. This takes dispersion into account, using original samples and predicted clusters from those to produce a score.

Figure 2.1

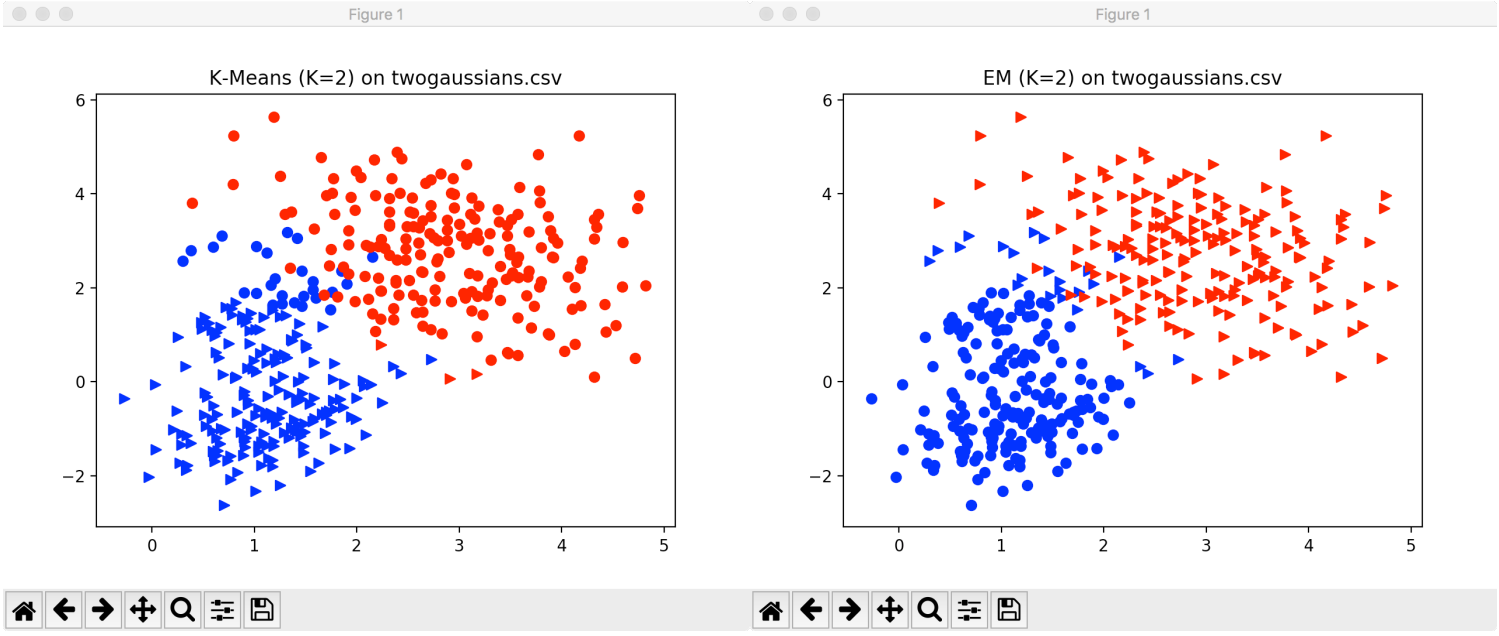


Figure 1

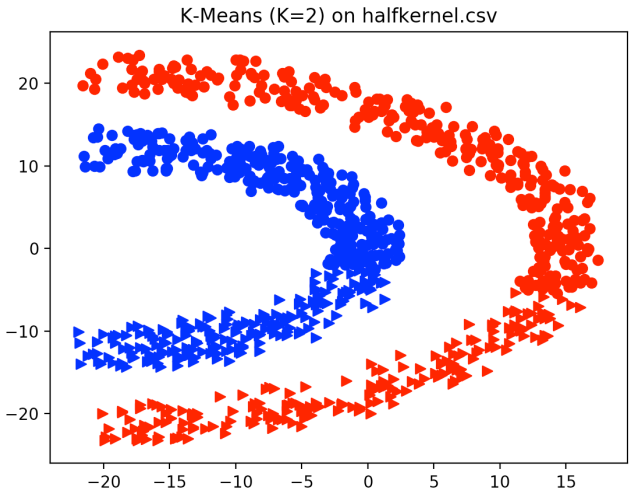


Figure 1

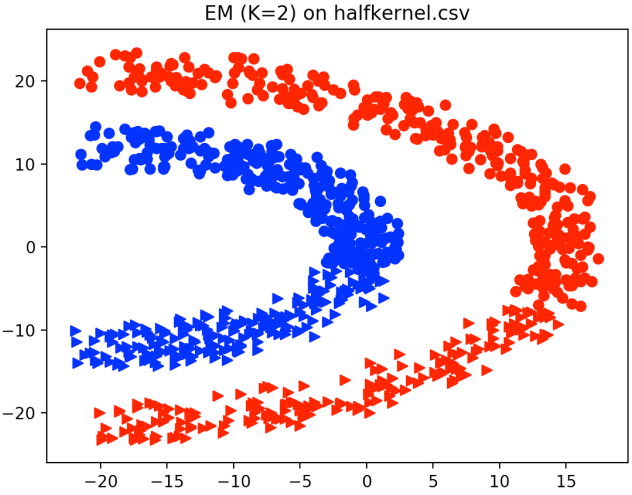
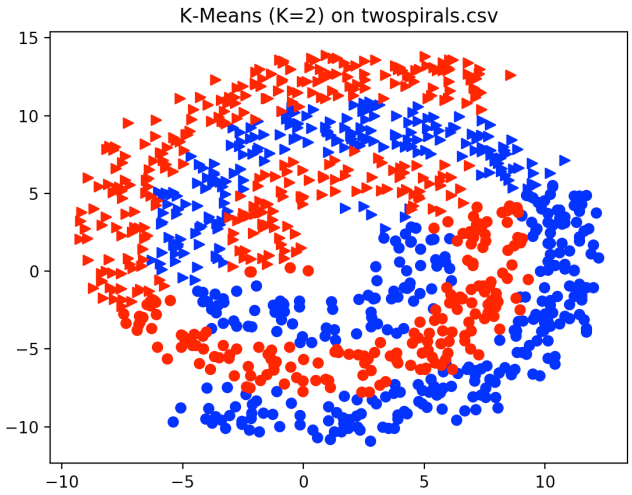


Figure 1



x=10.4716 y=-11.0672

Figure 1

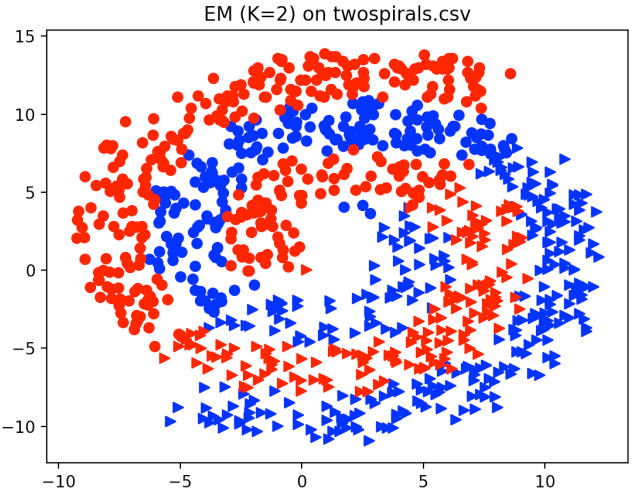


Figure 1

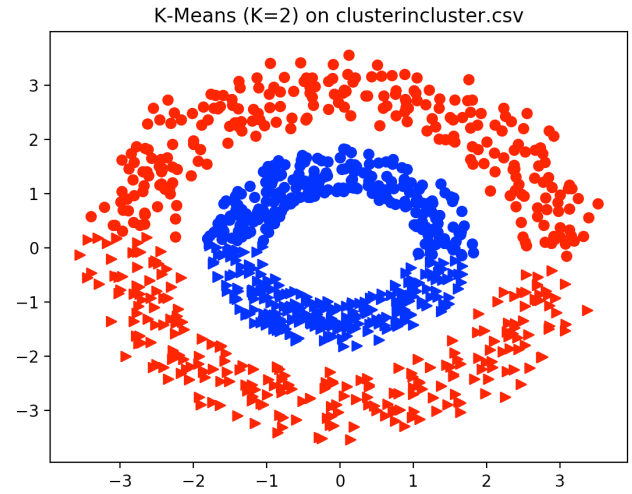
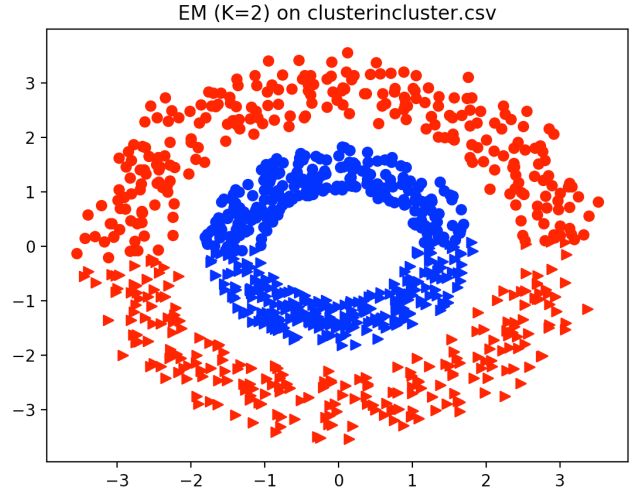


Figure 1



At a quick glance, the clustering results of K-Means and Expectation Maximization are notably similar. More specifically, it appears as though they are near identical, with the small amount of difference typically occurring near the obvious, linear decision boundary. Considering the fact that K-Means is a variant or specific case of Expectation Maximization, this is exactly the behaviour we should expect. The contested samples near the boundary are due to the nature of EM, which associates a (cluster) membership probability to these samples, which are thus designated as being *fuzzy*.

As a preliminary judgement, we may evaluate the clustering performance across clustering models and datasets by the shape and colour of the points. Given this limited information, we know that a perfect clustering would strive for a plot where samples with different colours (original class) are all of the same shape (cluster), implying a perfect one to one mapping from class to cluster. The only dataset where exemplary clustering has taken place is *twogaussians.csv*. This is primarily due to the linear nature of the K-Means and EM clustering algorithms, where due to the fact that two clusters were requested, yields a single linear decision boundary.

Moreover, it is clear that if only two clusters may be formed, it would be impossible to achieve reasonably accurate clustering for the remaining datasets. As depicted in Figure 2.1, *halfkernel.csv*, *twospirals.csv* and *clusterincluster.csv* cannot be accurately discriminated into two clusters about a single line. The K-Means and EM backing algorithms internally produce a *Voronoi Diagram*, consisting of k-1 linear decision boundaries (lines) where k is the number of clusters requested. The linear decision boundaries are also consequential of specifying the use of *Euclidean* distance as a distance metric between samples / clusters. Thus, the clustering appears to split the samples belonging to each original class, spreading them roughly evenly (due to their relatively *normal* distribution) among the two clusters formed in these three datasets.

To facilitate an opportunity to better demonstrate the power of clustering, a proper heuristic experiment was performed, testing many values of *k*. Displayed in Figure 3.1 below, the results of K-Means and EM clustering are recorded when asked to discriminate many different numbers of clusters. To offer a sufficient performance metric for comparison, the displayed scores are the Calinski-Harabasz (CH) indices for each respective clustering model over each dataset and *k* value.

Evaluating 'K' for twogaussians.csv

K	K-Means Score	EM Score
2	678.5670	643.1265
3	556.1192	482.0792
4	592.9363	543.1475
5	549.7232	520.6326
6	532.1636	503.4361
7	528.0117	507.8232
8	506.9211	460.6762
9	506.3088	451.3158
10	510.2294	435.2910
11	497.4508	442.2747
12	504.0432	469.7985
13	494.0984	411.2903
14	495.9487	426.9520
15	499.7056	404.5895
16	487.4393	426.8496
17	484.9127	401.3002
18	481.3809	364.8956
19	488.1907	389.1160
20	476.4083	411.8737
21	475.9640	450.3259
22	478.5046	351.0935
23	470.5997	306.5219
24	476.8818	396.0214
25	480.0611	354.0374
26	471.2093	352.7422
27	478.4050	321.6626
28	475.6516	379.7300
29	479.1358	336.3041
30	475.0460	355.2412
31	471.7255	270.3763
32	487.9137	289.4717
33	484.7498	266.4010
34	483.4194	212.1760
35	478.6107	310.8380
36	488.8561	242.8053
37	495.4569	246.0073
38	484.5067	287.5364
39	493.8022	306.0565
40	487.0195	213.6943
41	477.0724	192.8125
42	483.0762	337.2817
43	496.0634	300.9388
44	502.6140	212.9386
45	485.4045	352.7517
46	494.7211	254.2441
47	489.4299	376.7063
48	499.7347	260.0909
49	485.9443	229.6844
50	509.0422	254.4781
51	502.0973	283.8793
52	489.8776	339.3648
53	507.4830	325.2122
54	503.8433	221.3521
55	505.9013	346.6888
56	497.8027	276.8676
57	519.4326	394.1867
58	512.2814	266.0225
59	517.7532	253.0401
60	510.3287	186.8141
61	525.8143	305.3342
62	518.7215	229.5640
63	514.1063	301.5016
64	532.9041	181.1774
65	518.8979	339.9284
66	521.7147	336.7160
67	543.5139	335.7798
68	541.9808	341.4901
69	539.6354	340.4748
70	528.6089	339.9787
71	538.2123	304.6528
72	541.5883	295.6650
73	551.9645	325.8301
74	549.6782	144.6778
75	545.9816	385.2466
76	558.4288	312.1716
77	551.6224	341.6125
78	552.1485	310.3415
79	569.0315	391.7211
80	564.5503	345.3102
81	562.9188	377.8837
82	555.4463	341.6857
83	562.3706	374.3439
84	565.4602	382.4809
85	574.2701	406.0246
86	575.7152	344.8656
87	571.1467	382.9899
88	567.4915	380.3906
89	574.3124	360.6896
90	578.9145	388.8567
91	580.3518	424.2793
92	580.4553	370.9569
93	588.6324	365.6001
94	587.4931	464.8345
95	593.7360	437.9488
96	582.3117	404.2574
97	594.3083	444.3190
98	589.5367	450.6280
99	614.8632	427.8025
100	590.6452	439.3657

Evaluating 'K' for halfkernel.csv

K	K-Means Score	EM Score
2	823.5031	820.6276
3	1188.7766	1169.8788
4	1132.2765	916.4154
5	1204.6128	1021.2344
6	1271.1486	1088.1232
7	1303.9545	930.5080
8	1322.9455	1099.3032
9	1384.4268	1119.5087
10	1502.9955	1213.1830
11	1578.3175	1465.8370
12	1719.5058	1283.9941
13	1804.2831	1391.0743
14	1901.4791	1793.3467
15	2035.6342	1653.5096
16	2097.3950	1813.5368
17	2150.7040	2039.5563
18	2182.5294	2123.1533
19	2269.3578	2155.1050
20	2306.5420	2026.4066
21	2332.6089	2224.2900
22	2361.6055	2043.3448
23	2376.9543	2210.9434
24	2374.3272	2045.8971
25	2382.3755	2178.3042
26	2371.0232	2206.7023
27	2360.9273	2248.1678
28	2402.2308	2175.5804
29	2341.2017	2050.8998
30	2343.2940	2081.4537
31	2369.8353	2058.9528
32	2327.0442	1957.1613
33	2337.4306	2076.2752
34	2343.2661	2005.0669
35	2312.1074	1766.6537
36	2273.4644	2020.0119
37	2301.1987	1957.5774
38	2298.6724	1940.5191
39	2310.6017	2049.1671
40	2305.5014	1657.9445
41	2292.8681	2000.4228
42	2259.8778	2057.6546
43	2276.3134	2055.4193
44	2330.3614	1877.4369
45	2276.1068	1671.4216
46	2259.5062	1835.3690
47	2327.3941	2062.2349
48	2272.3759	1761.1308
49	2266.2465	1869.3107
50	2288.5724	1854.6263
51	2298.1080	1794.5961
52	2258.1599	1813.2278
53	2282.5270	1779.3467
54	2300.6615	1753.3436
55	2289.9670	2000.9675
56	2242.9702	1864.4882
57	2302.4109	1767.4625
58	2290.6213	1845.0560
59	2259.6937	1740.7127
60	2287.6517	1914.8422
61	2266.0525	1811.2811
62	2269.5573	1755.6105
63	2297.2857	1737.1976
64	2264.3169	1873.0935
65	2284.2714	1784.5065
66	2279.6267	1692.7896
67	2277.0389	2035.7597
68	2286.5678	1729.6449
69	2257.8170	1654.6964
70	2287.1715	1460.3423
71	2261.2474	1748.1675
72	2303.0141	1858.6306
73	2275.1234	1919.5425
74	2306.3312	1659.4084
75	2249.8656	1867.5274
76	2280.4081	1660.1077
77	2291.2162	1734.8088
78	2316.8679	1861.9141
79	2324.6491	1682.8383
80	2292.9991	1842.7306
81	2325.0859	2009.0458
82	2297.8678	1711.4368
83	2288.2102	1771.0978
84	2357.9993	1824.8965
85	2323.8526	1606.8707
86	2336.7462	1607.7686
87	2340.8907	1902.5248
88	2332.8055	1822.3413
89	2345.0024	1962.8568
90	2359.6047	1773.0400
91	2359.6362	1718.2368
92	2363.8273	1893.4663
93	2370.0313	1950.8745
94	2320.4140	1658.4759
95	2361.5376	1875.2013
96	2375.8497	1636.0936
97	2374.8393	1749.2637
98	2363.4732	1295.4374
99	2362.5110	1826.3576
100	2368.8948	2024.1110

Figure 3.1

Evaluating 'K' for twospirals.csv

K	K-Means Score	EM Score
2	762.9318	734.2163
3	895.5937	875.6619
4	1081.5092	1064.5025
5	1088.6067	1035.1380
6	1098.7412	1076.0370
7	1069.2539	1024.5640
8	1051.5303	949.2693
9	1065.4122	885.4680
10	1049.9948	1005.1689
11	1076.4401	925.6497
12	1104.3137	892.0335
13	1107.8864	887.5245
14	1102.1118	970.0848
15	1120.5782	1026.0236
16	1117.5205	1066.7014
17	1141.8494	1011.4863
18	1147.7129	1046.6666
19	1152.5047	938.4980
20	1155.3438	1041.9738
21	1143.9331	988.7933
22	1155.0784	866.7037
23	1151.7771	970.0262
24	1149.8741	1041.8497
25	1161.6274	1025.9818
26	1151.1154	952.2690
27	1145.3538	980.4696
28	1175.3518	980.1709
29	1145.1625	860.1371
30	1148.2493	976.8480
31	1173.7260	879.4731
32	1167.8332	934.0351
33	1206.7070	968.6402
34	1171.3010	902.9996
35	1176.6326	994.4207
36	1176.3033	1041.5604
37	1190.6330	886.9192
38	1191.1555	900.3704
39	1214.9811	951.9114
40	1204.2802	895.3899
41	1193.1121	772.6940
42	1192.1572	824.2827
43	1220.0377	1003.0968
44	1191.3710	825.0561
45	1210.1814	915.1621
46	1206.5002	907.6168
47	1207.1009	963.0283
48	1209.7835	853.4464
49	1205.1433	791.8914
50	1219.5410	891.0750
51	1212.7119	908.5446
52	1221.7713	923.0785
53	1222.3257	822.9289
54	1194.7660	845.7748
55	1223.9106	964.2964
56	1213.6647	938.1644
57	1236.7151	948.1349
58	1232.1198	961.7425
59	1214.5712	978.5104
60	1225.7469	797.0694
61	1223.2342	923.2921
62	1224.3196	890.8868
63	1231.0861	856.6131
64	1231.8303	745.8179
65	1239.4204	876.7805
66	1245.7383	920.9665
67	1244.0347	911.2641
68	1222.5553	973.2301
69	1232.0458	878.6551
70	1252.8605	818.8939
71	1255.3103	887.9635
72	1248.1826	888.7260
73	1280.6354	777.4300
74	1267.6183	883.4346
75	1263.0012	810.6029
76	1252.1199	741.2523
77	1256.4307	995.2229
78	1282.9668	801.7900
79	1253.7491	977.5398
80	1286.5424	878.0043
81	1285.3230	904.6796
82	1270.8595	799.5574
83	1254.7648	893.5010
84	1271.2016	840.0053
85	1302.7436	790.6704
86	1275.8549	786.1764
87	1310.9213	949.7547
88	1290.5207	919.9484
89	1269.8893	841.7093
90	1309.4363	909.3307
91	1267.6282	846.6954
92	1293.2460	811.3956
93	1303.5444	748.2352
94	1303.9221	951.2985
95	1276.7351	879.2751
96	1285.3179	794.4239
97	1313.2727	849.0661
98	1305.2571	924.6385
99	1305.1718	595.0655
100	1336.0525	953.1097

Evaluating 'K' for clusterincluster.csv

K	K-Means Score	EM Score
2	595.2088	592.2761
3	769.5106	761.9194
4	844.3027	833.1110
5	863.7557	857.4494
6	815.0003	789.2132
7	790.6687	753.7189
8	785.9120	714.7381
9	815.1541	759.9858
10	868.4407	813.6921
11	921.3873	859.3288
12	977.1613	900.5766
13	1049.2427	917.1680
14	1072.6060	935.8710
15	1117.7735	1046.5660
16	1163.6388	1098.0569
17	1172.5210	1138.6346
18	1199.2599	1141.2402
19	1238.8676	1187.5580
20	1243.0274	1146.4657
21	1250.0704	1165.9382
22	1282.6032	1182.3086
23	1270.7293	1189.0251
24	1288.3695	1191.8525
25	1285.2698	1233.3315
26	1287.5604	1196.7948
27	1280.5622	1203.5669
28	1291.4434	1151.3639
29	1275.2666	1181.3735
30	1278.5601	1201.0228
31	1301.8436	1094.0052
32	1270.8252	1172.9243
33	1270.5124	1103.3675
34	1271.9036	1181.3049
35	1245.3451	1167.0632
36	1237.3965	1119.1835
37	1252.0131	1146.5515
38	1238.4946	962.2246
39	1241.1250	928.5869
40	1228.7150	1062.4291
41	1223.3726	1120.2977
42	1232.8168	1102.2018
43	1230.2162	1011.1847
44	1220.3452	989.9086
45	1209.6746	1001.9531
46	1190.8012	959.2180
47	1212.6106	921.4513
48	1227.9239	886.4037
49	1217.1545	963.5158
50	1203.8368	966.8378
51	1203.3478	949.6752
52	1220.3204	1011.5360
53	1225.4371	963.3116
54	1223.9172	1018.3637
55	1213.4265	1015.9343
56	1218.8970	976.5827
57	1228.2238	908.2133
58	1215.6291	964.7223
59	1225.0829	891.4178
60	1218.8973	858.9196
61	1216.5105	948.8432
62	1232.3437	1031.1219
63	1232.6023	919.5168
64	1233.3924	1042.8350
65	1240.6376	1044.2502
66	1246.0984	1079.8121
67	1235.9746	794.5131
68	1252.5743	937.2657
69	1259.5953	894.9320
70	1269.4741	837.5209
71	1299.7846	1046.9185
72	1291.1316	951.3456
73	1260.0499	881.2846
74	1285.6402	1031.4634
75	1260.5833	955.1760
76	1283.8469	913.0930
77	1282.5283	966.4242
78	1276.3089	960.5187
79	1271.4450	1048.1826
80	1292.7705	954.1740
81	1286.9618	1006.3036
82	1286.7389	910.5392
83	1307.4215	1020.5005
84	1279.5878	782.0101
85	1286.6265	1008.9783
86	1295.1869	794.5312
87	1292.8874	815.8141
88	1288.0805	824.0101
89	1294.9951	1005.7752
90	1285.8833	986.2147
91	1300.1202	966.9722
92	1307.0833	934.9876
93	1316.2598	1068.4787
94	1299.0311	849.8387
95	1306.2296	1051.2084
96	1325.2810	939.0403
97	1313.2284	985.9255
98	1301.8395	893.5401
99	1313.4993	967.2058
100	1305.7355	985.2153

Combined with the summary of this data in Figure 3.2, a sufficient analysis may now be performed on the clustering models chosen. Generally, it is clear that the three non-linearly separated datasets have a wide, increasing range of indices. Similarly, *twogaussians.csv* shows a stable, slightly decreasing pattern as k becomes larger, with less *variance* than that of the indices across other datasets.

```
Best k for twogaussians.csv
KMeans: 2
EM: 2

Best k for halfkernel.csv
KMeans: 28
EM: 27

Best k for twospirals.csv
KMeans: 100
EM: 6

Best k for clusterincluster.csv
KMeans: 96
EM: 25
```

Figure 3.2

When run on the *twogaussians.csv* dataset, a significantly small number of clusters has been deemed as ideal for K-Means and EM by the CH Index. As discussed, requesting two clusters to be formed is ideal for this dataset, as the original classes are distinctly linearly separable (Figure 2.1). This specific dataset is a classic example of *overfitting*. Any value of *k* greater than two is simply unnecessary, and the CH Index is a reputable indicator for this condition. The difference in performance between K-Means and EM clustering is marginal, with K-Means being slightly better defined.

The clustering that has taken place on *halfkerrnel.csv* is significantly better in score than any other dataset, with CH Indices approaching 2500. This exemplary score makes perfect sense. The visual representation of the sample data in Figure 2.1 shows how the original classes in this dataset are highly (distinctly) discriminated / separable. The high score for CH Index is logical due, as it essentially measures the average *between* and *within cluster dispersion*. The notion of a good clustering prioritizes such features, specifically the distance between clusters and cluster density (larger distance and more dense being optimal). The optimal *k* value, 28, is sensibly low, as the dispersion will aid in concluding that a given cluster formation is sufficient.

With reasonable means, the analysis of the clustering model performance of *twospirals.csv* and *clusterincluster.csv* can be summarized together with similar explanation. Although *clusterincluster.csv* sports a visually superior dispersion in sample data (Figure 2.1), the CH Indices seem not to reflect this as well as *halfkernel.csv*. Along with *twospirals.csv*, the shape and sample size of the original dataset essentially restrict the better scores to higher values of *k* relative to the sample size *n*. The fact that one class is embedded visually within the other in *clusterincluster.csv* will obviously be difficult for K-Means or EM to distinguish without relatively large number of clusters. The optimal *k*, 96 and 25 for K-Means and EM, suggests that overfitting may be occurring for K-Means, as all scores past k = 32 (the approximate square root of sample size n) begin to converge / round out at this point.

Similarly, *twospirals.csv* will suffer similarly in performing high scoring clustering for small values of *k*. In particular, the dispersion and complex shape of the samples of this dataset is quite poor, as clear in Figure 2.1. This will inevitably lead to poor clustering. The standard for sufficiently large *k* is much greater for this dataset than that of any other. The optimal values of *k*, 100 and 6, attest to this fact. The CH Indices are slightly increasing with larger k, however this is possible overfitting as the ratio of requested clusters to samples is increasing at an alarming rate. Furthermore, however overfit the clustering becomes, the scores of these indices would likely increase marginally across values of *k* higher than those of the experiment.

It is also worth noting the difference in CH Indices of the K-Means and EM clustering models over the *twospirals.csv* and *clusterincluster.csv* datasets. From Figure 3.1, it is clear that while the scores are nearly equal when k = 2, as *k* increases, the scores of EM *diverge* by a significant amount to those of the K-Means model. The K-Means model, producing scores around 1300 for both datasets for k = 100, is much larger in comparison to EM, which scores around 900 on both datasets for this *k*. We may attribute this to the fact that soft assignment is taking place within the EM model, and due to the complexity (embedded/ dispersed shape) of these datasets plot, will take a hit on the score for larger *k*.

In summary, it is evident that the CH Indices of *both* clustering models are near identical when evaluated heuristically on each dataset. K-Means has proven to be slightly better by measure of CH Index, but also yields higher values of k for the same findings. We may hypothesize that K-Means is slightly more accurate, or practical in terms of clustering performance. Likewise, EM may produce reasonably similar results, yet requiring less testing for larger values of k.