Neelkumar S Bhuva
K-Means Report

1) INTRODUCTION :

K-Means is used for unsupervised clustering. It partitions n observations into K clusters where points in each cluster are close to each other. The algorithm is as follows :
Step 1 : Choose k centroids for k clusters.
Step 2: Assign all input points to one of the k clusters based on a euclidean distance.
Step 3: Update the centroid of all clusters by calculating the mean of points in each cluster.
Step 4: If the centroid has changed then go back to step 2. Repeat until none of the centroid changes.
There is no guarantee that the global optimum is found with K-means. There are multiple local optimum solutions and the results depends on the initial clusters. So, step 1 here is very crucial.

Missing Values : There are no missing values.
Normalization : Attributes "fx_acidity", "resid_sugar", "free_sulf_d", "tot_sulf_d", "pH" and "alcohol" are normalized so that all attributes have same scale (0-1). MinMax scaling was used.

2) DESIGN CHOICES :

A) Initialize Centroids :

Random Uniform Initialization : In step 1 k centers are selected randomly from n input points assuming uniform distribution. Following are the overall SSE's from **running K-means 10 different times on TwoDimHard dataset for K = 4.**

Table 2.1 : Overall SSE's :

SSE	10.07	9.38	5.81	14.16	12.23	8.95	9.20	8.29	8.62	11.32
SSB	19.22	19.92	23.49	15.13	17.07	20.35	20.09	21.01	20.67	17.97

Table 2.2 : Best, Average and Worst SSE by running K-means 10 times

Best SSE	Average SSE	Worst SSE
5.81	10.91	14.16

K-means++ :

a) Select one center randomly from the input points.
b) Calculate the distance d of all points to this center.
c) Use probability distribution d_i^2/∑d_i^2 for i = 1,2,3...N to select second center from the N input points.
d) Update the distance d for each point x as min [$(x - c_1)$, $(x - c_2)$]
e) Repeat c and d until k centers are selected.

Table 2.3 : Running k-means 10 times for K = 10: Overall SSE's :

SSE	7.97	9.37	8.53	5.41	6.00	5.49	9.32	9.29	5.36	7.19
SSB	21.32	19.92	20.77	23.89	23.29	23.81	19.97	20.01	23.94	22.10

Table 2.4 : Best, Average and Worst SSE by running K-means 10 times

Best SSE	Average SSE	Worst SSE
5.36	7.39	9.37

From the above results K-means with K-means++ to select initial centroids gives lower overall SSE's and higher SSB's. So, for this experiment K-means++ is used to select k initial centroids. K-means++ speeds up convergence.

3) True Cluster :

TwoDimHard dataset :

True Cluster SSE : 1: 0.31, 2: 0.90, 3: 2.43, 4: 1.91
True Overall SSE : 5.55 True SSB : 23.75

Wine dataset :

Using quality attribute as true cluster :
True Cluster SSE : 3: 2.49, 4: 13.23, 5: 127.39, 6: 123.57, 7: 37.27, 8: 3.58
True Overall SSE : 307.53 True SSB : 29.48

4) Analysis of TwoDimHard dataset:

Cluster SSE measures how close the data points are within each cluster. SSB measures how well separated are the clusters from each other. We want overall SSE to be minimum indicating that the points within each cluster are close to each other and SSB to be maximum indicating that the clusters are well separated.

A) <u>For K = 4 :</u>

Table 4.1 : Cluster SSE, Overall SSE and SSB

Metris	Iteration 1	Iteration 2	Iteration 3
SSE (Cluster 1)	0.616	2.023	0.945
SSE (Cluster 2)	2.118	1.208	2.021
SSE (Cluster 3)	1.122	1.462	1.574
SSE (Cluster 4)	1.908	0.312	0.397
Overall SSE	**5.765**	**5.006**	**4.939**
SSB	**23.538**	**24.297**	**24.364**

The results are as good as true clusters (Section 3). The overall SSE decreased and SSB increased for successive iterations. Even though these results are not true everytime, K-means++ does perform relatively well as seen in section 2.

Selecting the best SSE and SSB (Iteration 3) from table 4.1 :

Table 4.2 : Cross Tabulation Matrix for Iteration 3

Cluster	1 (Actual)	2 (Actual)	3 (Actual)	4 (Actual)
1 (Predicted)	98	1	5	0
2 (Predicted)	0	90	3	0
3 (Predicted)	0	4	106	0
4 (Predicted)	2	2	0	89

From Table 4.2 and Fig 4.1, the true clusters and predicted clusters are not very different from each other.

Scatter Plot :

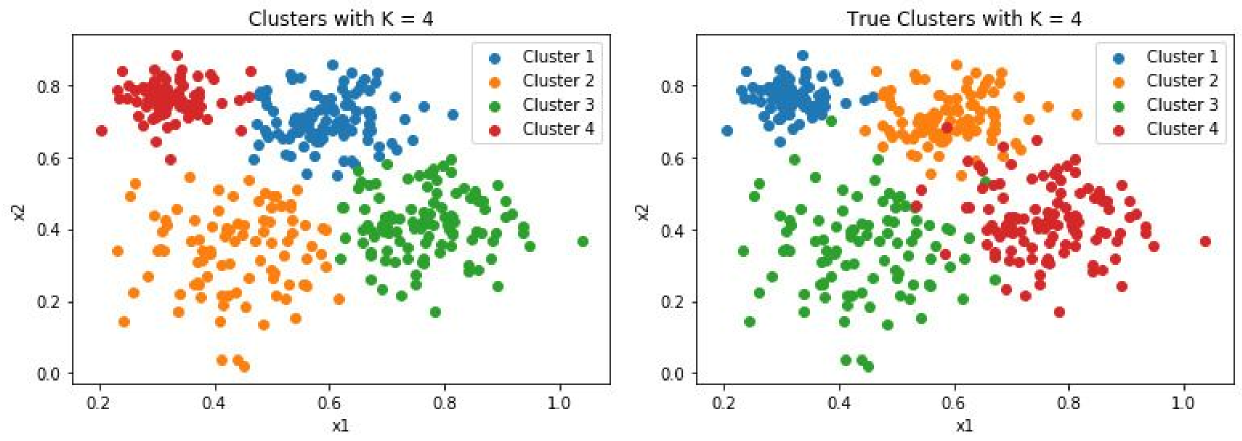


Fig 4.1 : Image on the left shows predicted clusters and image on right shows true clusters

B) <u>For K = 3 :</u>

Table 4.3 : Cluster SSE, Overall SSE and SSB

Metris	Iteration 1	Iteration 2	Iteration 3
SSE (Cluster 1)	5.195	6.150	5.484
SSE (Cluster 2)	1.430	1.251	2.582
SSE (Cluster 3)	2.265	1.988	2.895
Overall SSE	**8.891**	**9.390**	**10.961**
SSB	**20.412**	**19.913**	**18.342**

Table 4.4 : Cross Tabulation Matrix for Iteration 1

Cluster	1 (Actual)	2 (Actual)	3 (Actual)	4 (Actual)	Total
1 (Predicted)	89	100	2	6	197
2 (Predicted)	0	0	2	103	105
3 (Predicted)	0	0	93	5	98
Total	89	100	97	114	400

Scatter Plot for Iteration 1:

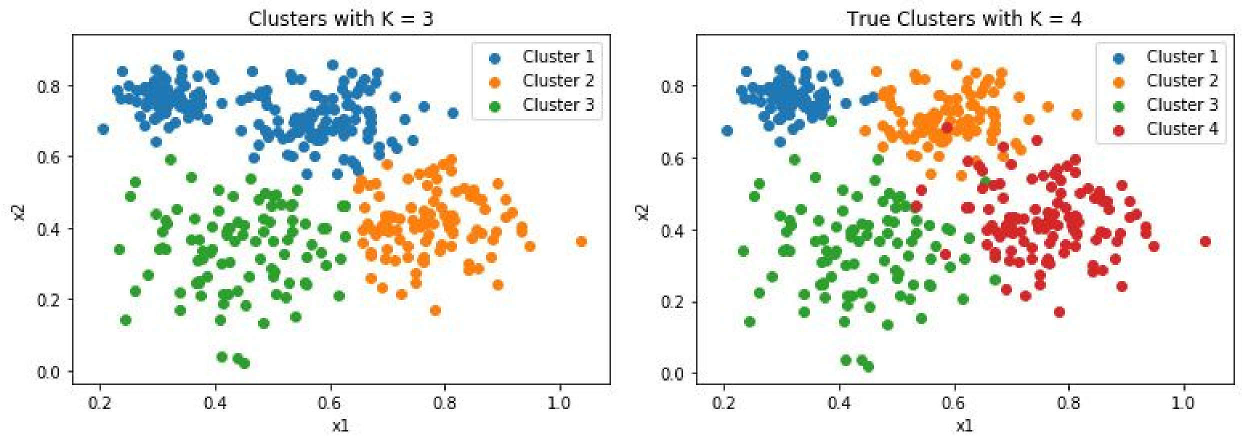


Fig 4.2 : Image on the left shows predicted clusters and image on right shows true clusters

Changing K from 4 to 3 changed the results. From table 4.3, SSE increased and SSB decreased for K = 3 compared to K = 4 for all 3 iterations indicating that K = 3 performs poorly. From the true scatter plot there are 4 dense areas. Using K = 3 combines two of the dense clusters into one depending on the initial centroids. Doing this, naturally increases the sum of square errors as the distance of points from the centroid increases.

To double check the conclusion, K-means is run 10 times and results are found to be worse for K = 3 :
SSE's : [9.561, 9.592, 10.598, 11.471, 10.280, 12.075, 16.667, 10.195, 10.554, 9.321]
Best SSE : 9.321 **Average SSE :** 11.032 **Worst SSE :** 16.667
SSB's : [19.743, 19.711, 18.705, 17.833, 19.023, 17.228, 12.636, 19.108, 18.750, 19.983]
Clearly the SSE's are higher compared to K = 4 and SSB's are lower.

5) Analysis of Wine dataset:

The number of clusters was varied from 2 to 11. SSE and SSB were calculated for each cluster. Fig 5.1 shows how SSE and SSB varies with K. Quality attribute is used for external validation.

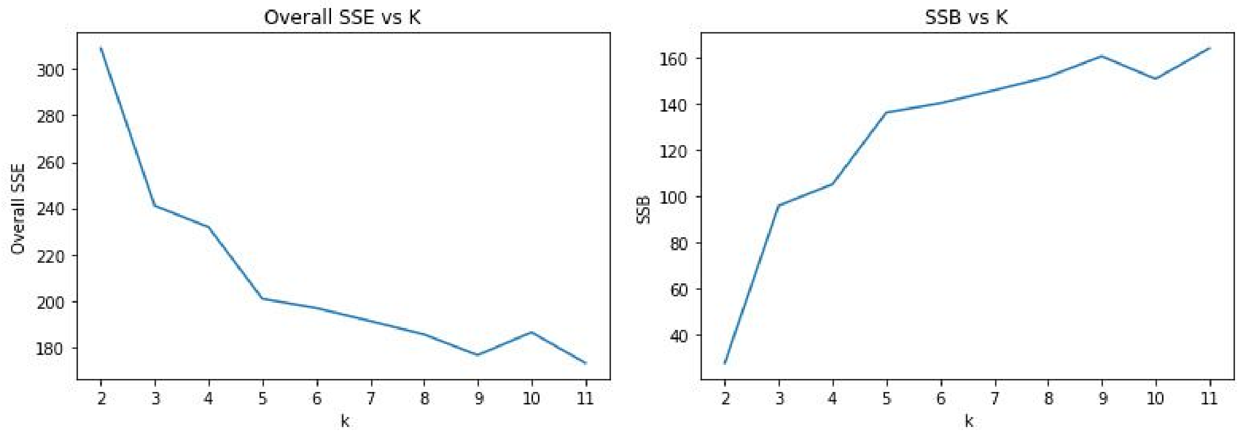


Fig 5.1 : Image on left shows overall SSE vs K and image on right shows SSB vs K

	K = 4	K = 5	K = 8	K = 9	K = 10	K = 11
SSE	231.793	208.798	185.453	176.491	186.333	173.101
SSB	105.226	128.221	151.566	160.527	150.685	163.917

From Fig 5.1 K = 11 has the lowest SSE and highest SSB. But, K = 9 has SSE and SSB values very close to K = 10. So, K = 9 is better as compared to K = 11 because K = 11 increases complexity.

The increase of SSB and decrease in SSE is slow after K = 5. So, the **best K would be 5**. K = 9 might result in overfitting where multiple smaller clusters are formed within one big cluster splitting similar points into different groups.

For K = 5 : Cluster SSE : {1: 53.26603377196223, 2: 8.14462004178295, 3: 19.10853517499216, 4: 60.46434719322064, 5: 67.81458631745834}

Table 4.2 : Cross Tabulation Matrix for K = 5

Cluster	3	4	5	6	7	8	Total
1 (Predicted)	0	16	295	173	23	1	508
2 (Predicted)	0	1	19	14	2	0	36
3 (Predicted)	7	23	90	60	7	0	187
4 (Predicted)	1	4	134	157	52	4	352
5 (Predicted)	2	9	143	234	115	13	516
Total	10	53	681	638	199	18	1599

True cluster 3 is insignificant as it has only 10 points (Could be noise cluster). With predicted clusters, each cluster has significant number of points.

6) Off The Shelf K-Means :

Scikit in python was used to implement K-Means for this dataset.
Package : sklearn.cluster.KMeans
This package uses k-means++ to initialize centroids. The K-means algorithm is run 10 times with different centroid seeds and the best output is selected. Euclidean distance measure is used.

A) TwoDimHard dataset :

For K = 4 :
Table 6.1 : Cluster SSE for K = 4

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
SSE	1.470	1.076	0.500	1.844

Overall SSE : 4.892 SSB : 24.412
This result is close to SSE : 4.93 and SSB : 24.36 in Section 4 (implemented K-means). But we cannot say that implemented K-means performs as good as sklearn K-means, since this result is not true for all iterations. Sklearn K-means is much more stable and better.

Table 6.2 : Cross Tabulation Matrix for K = 4

Cluster	1 (Actual)	2 (Actual)	3 (Actual)	4 (Actual)	Total
1 (Predicted)	89	0	2	4	95
2 (Predicted)	0	104	0	3	107
3 (Predicted)	0	8	98	2	108
4 (Predicted)	0	2	0	88	90
Total	89	114	100	97	400

Parameter Change : Initial centroids were randomly selected instead of using k-means++ and the result was exactly the same as k-means++. Further changing the number of iterations from 10 to 20 and then to 50 did not change the SSE and SSB results.

Scatter Plot :

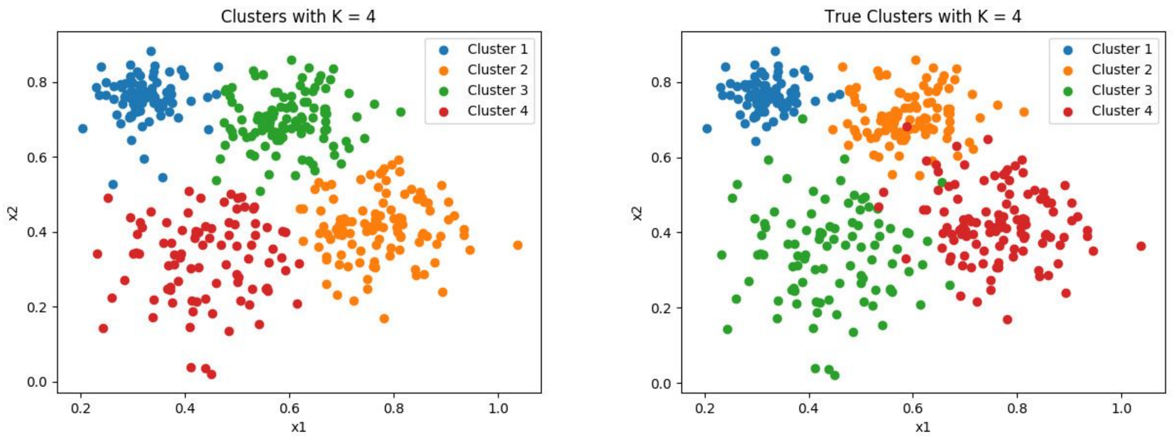


Fig 6.1 : Image on the left shows predicted clusters and image on right shows true clusters

For K = 3 : Overall SSE increased to 8.56 and SSB decreased to 20.74.

Conclusion : **K = 4 is better compared to K = 3. This result is consistent with implemented K-means.**

Also, K was varied from 2 to 12, K = 4 gave the best result. K = 11 has lowest value of SSE and highest value of SSB. But the decrease in SSE and increase in SSB is slow after K = 4.

B) Wine dataset :

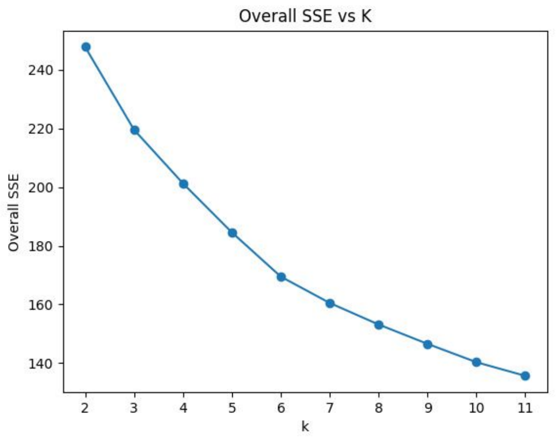
The number of clusters was varied from 2 to 11. SSE and SSB were calculated for each cluster. Fig 6.2 shows how SSE and SSB varies with K.

From Fig 6.2 below, K = 11 has lowest value of SSE and highest value of SSB. But the decrease in SSE and increase in SSB is slow after K = 6. **K = 6 is selected (as opposed to K = 5 in Section 5) as it is a simpler model compared to K = 11 which might result in overfitting.**

For K = 6 : **Cluster SSE :** {1: 24.201622833019297, 2: 11.720304731930792, 3: 25.074860170471887, 4: 28.104334190618385, 5: 37.84901290100405, 6: 42.542944200496365}

Overall SSE : 169.49307902754077 **SSB :** 167.52636348118762

Compared to result in section 6 (SSE: 208.79 SSB : 128.22) SSE here is significantly lower and SSB is significantly higher. Sklearn K-means on wine data gives much better result.



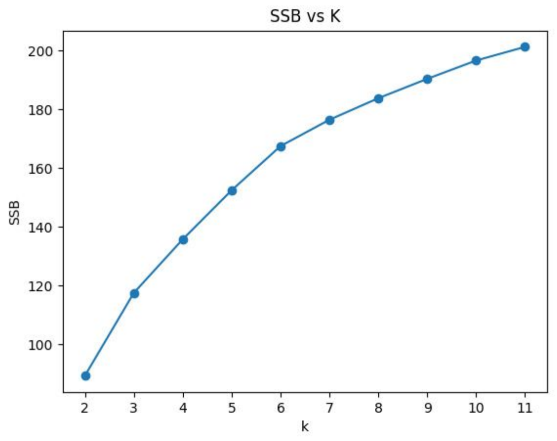


Fig 6.2 : Image on left shows overall SSE vs K and image on right shows SSB vs K

Table 6.3 : Cross Tabulation Matrix for K = 6

Cluster	3	4	5	6	7	8	Total
1 (Predicted)	0	10	52	126	30	4	222
2 (Predicted)	0	2	33	15	2	1	53
3 (Predicted)	2	5	74	105	46	4	236
4 (Predicted)	0	4	38	145	97	9	293
5 (Predicted)	7	28	273	158	13	0	479
6 (Predicted)	1	4	211	89	11	0	316
Total	10	53	681	638	199	18	1599

From Table 6.3, there is no insignificant cluster and hence there is no noise cluster.

Parameter Change : Initial centroids were randomly selected instead of using k-means++ and the result was exactly the same as k-means++. Further changing the number of iterations from 10 to 20 and then to 50 did not change the SSE and SSB results.