CSE 5243 – HOMEWORK 5
VEDASRI UPPALA (.1)

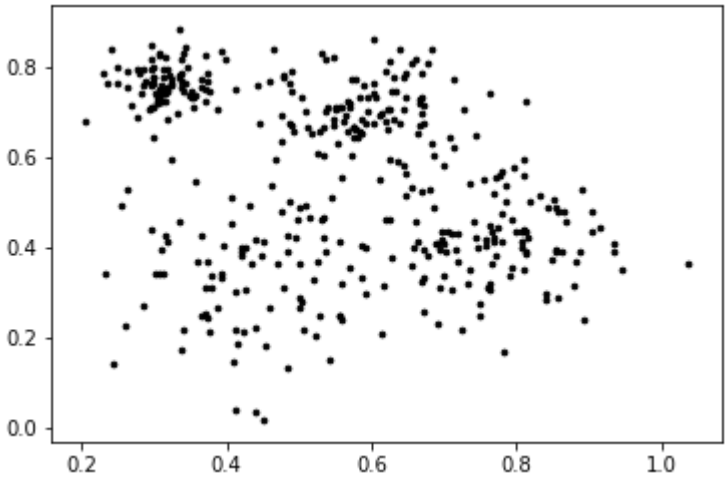
1. INTRODUCTION

K-means clustering algorithm has been implemented from scratch as a part of Homework 5. The implemented algorithm has been run on the *TwoDimHard* data set for two k values and the results are summarized in this report. A comparison has been drawn by running the off-the-shelf implementation of K-means algorithm available in *scikit-learn* machine learning library in python. In addition, the off-the-shelf implementation has also been used to cluster the wine-quality-red data set and the observations are included in the report.

2. CODE DESCRIPTION AND DESIGN CHOICES

The K-means clustering algorithm has been implemented for the *TwoDimhard* Data set having 4 attributes- *ID, X.1, X.2* and *cluster*. *ID* is the unique identifier for each point in the data set. *X.1* and *X.2* are the values of the co-ordinates of the 2-dimensional points that the dataset represents. The cluster attribute indicates the true cluster of the given data point.

The following scatter plot helps to visualize the data points in the TwoDimHard data set. The scatter plot has been plotted for the TwoDimHard data set with the attribute *X.1* on horizontal axis and *X.2* on vertical axis. By visualizing the data points through the scatter plot, we intuitively feel that the data set can be partitioned into 4 clusters. This hypothesis has been tested in the next section.



I have used Euclidean distance as the proximity measure to assign clusters to the data points. The first 4 points in the data set have been used as the initial centroids. Then for every data point the nearest cluster has been determined by calculating the Euclidean distance between the point and the current centroid of each cluster. The data point is then assigned to the cluster whose centroid is closest to the point(Minimum Euclidean distance). Then, the new centroids have been updated by calculating the mean of all the data points falling within a single cluster. This process has been repeated until the centroids of the clusters have not changed in two successive iterations.

K-means is sensitive to the scale of the features since Euclidean distance is used as proximity measure. So, it is preferable to apply normalization to the attributes before applying K-means to a data set. But, I have

observed that in the data set 'TwoDimHard' both the features have the same range. So, I have not performed any normalization.

3. ANALYSIS OF RESULTS

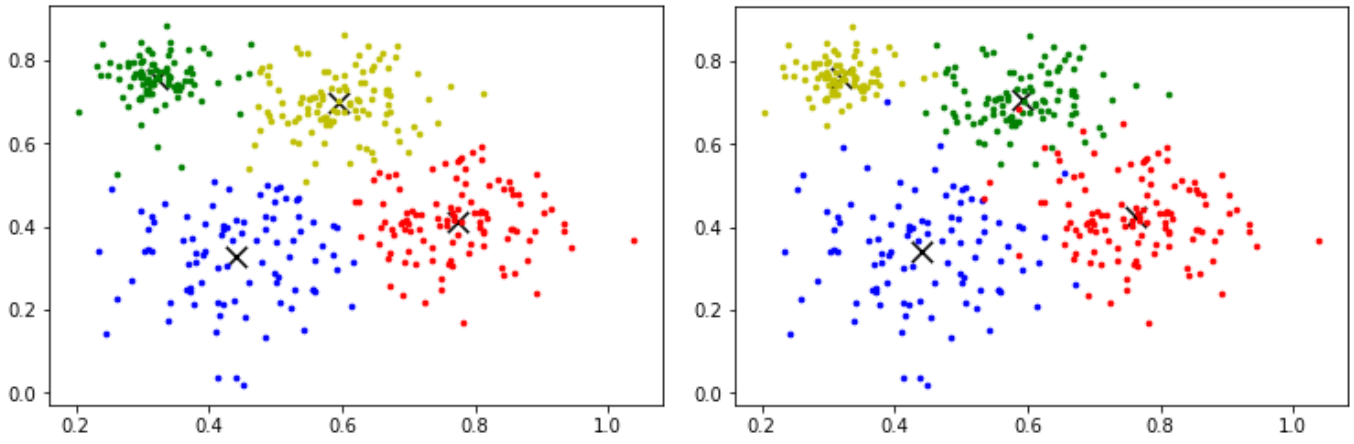
The implemented k-means algorithm has been tested on the *TwoDimHard* data set with *k=4*. The cross-tabulation matrix comparing the true clusters (using *cluster* attribute) and the predicted clusters is shown below. We can see from below that there has been fair division of points into 4 clusters and the number of points predicted in each cluster is highlighted in red.

K = 4	**Actual Clusters**				
Predicted Clusters	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
Cluster 1	0	98	2	8	108
Cluster 2	89	2	4	0	95
Cluster 3	0	0	88	2	90
Cluster 4	0	0	3	104	107
Total	89	100	97	114	400

Cross tabulation matrix for k=4

The following scatter plot on the left shows the cluster assignments that have been made using my implementation. The scatter plot on the right shows the true clusters according to the *cluster* attribute. In each scatter plot, the centroids of the clusters are shown with a 'x' symbol. By visual observation we can say that my implementation has been successful in dividing the data points into clusters.

Predicted Clusters **True Clusters**



Since we know the true cluster of each point, we can calculate true SSE (Sum of Squared Errors) and the first row of the table following shows the values of the true SSE for each cluster and the total true SSE.

Using my implementation, I have calculated the SSE with predicted clusters and the second row in the table indicates the predicted SSE for each cluster and the total predicted SSE. I have also calculated the between cluster sum of squares SSB for true and predicted clusters.

K=4	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
True SSE	0.31284	0.90253	2.43011	1.91071	5.55621
Predicted SSE	1.07648	0.50048	1.84460	1.47053	4.89210

True and predicted SSE with k=4

K=4	True SSB	Predicted SSB
	24.51317	24.412238265

True and predicted SSB with k=4

To identify a good value for k, the analysis mentioned above has been repeated by taking k = 3. The following table shows the cross-tabulation matrix comparing true and predicted clusters when k=3.

K = 3	Actual Clusters				
Predicted Cluster	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
Cluster 1	0	6	3	110	119
Cluster 2	0	0	89	3	92
Cluster 3	89	94	5	1	189
Total	89	100	97	114	400

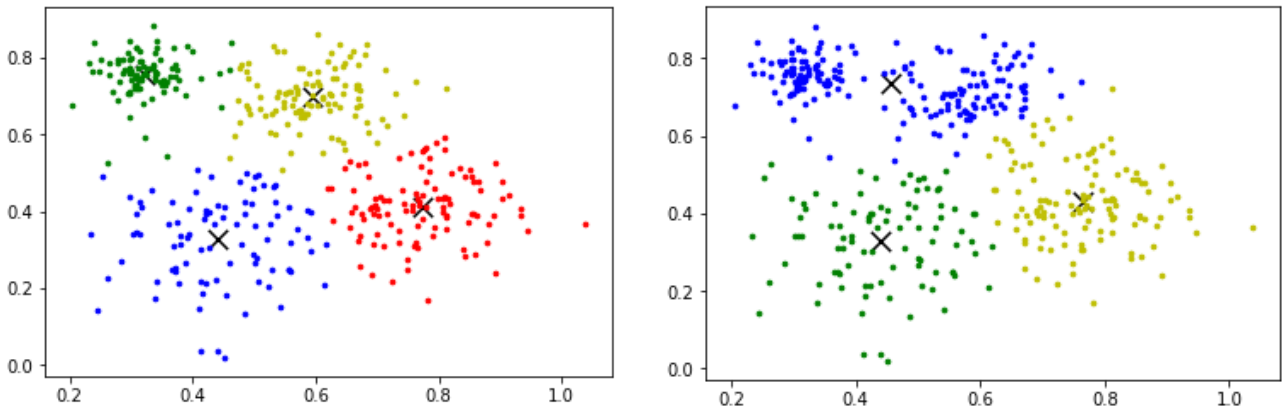
Cross Tabulation matrix with k=3

K=3	Cluster 1	Cluster 2	Cluster 3	Total
Predicted SSE	4.393772	2.208190	1.959400	8.561363

Predicted SSE when k=3

To identify better k value, I have compared the results obtained by taking k=3 and the results obtained by taking k=4.

By looking at the following scatter plots we can say that when the value of k is changed to 3, the two clusters in the top have merged with each other.



The following table summarizes the values of predicted SSE and SSB obtained by taking k=3 and k=4.

Predicted Clusters	**SSE**	**SSB**
K=3	8.561363	20.7396662919
K=4	4.89210	24.412238265

Comparison of SSE and SSB for predicted clusters using k=3 and k=4

Ultimately, we aim to have the points in a cluster very close to each other(low SSE) and the clusters to be fairly separated (high SSB). Since two clusters got merged when k=3, the SSE has increased from 4.89210 to 8.56136 and the SSB has decreased. Moreover, in the cross-tabulation matrix for k=3, we can see that 189 data points belonging to cluster 3 have been distributed almost equally. So, the distance from the centroid increases for these data points leading to high SSE and low SSB values. So, by considering these factors we can conclude that k=4 is better for the TwoDimHard data set.

4. OFF-THE-SHELF K-MEANS CLUSTERING

To analyze the performance of my implementation of the K-means algorithm, I have compared it with off-the-shelf implementation of K-means available in *scikit-learn machine learning library for python*.

K = 4	**Actual Clusters**				
Predicted Clusters	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
Cluster 1	0	98	2	8	108

Cluster 2	0	0	3	104	107
Cluster 3	0	0	88	2	90
Cluster 4	89	2	4	0	95
Total	89	100	97	114	400

Cross Tabulation matrix with k=4 for scikit-learn implementation

K = 3	**Actual Clusters**				
Predicted Cluster	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
Cluster 1	89	92	5	1	187
Cluster 2	0	8	3	110	121
Cluster 3	0	0	89	3	92
Total	89	100	97	114	400

Cross tabulation matrix with k=3 for off-the-shelf implementation

The SSE and SSB values have been calculated with the packages available KMeans library of scikit learn and a comparison has been drawn with values obtained using my implementation. The results are tabulated and are shown below.

SSE when K=3	**My Implementation**	**Off-the-shelf Implementation**
Cluster 1	4.393772	4.393772
Cluster 2	2.208190	2.208190
Cluster 3	1.959400	1.959400
Total	8.561363	8.561363

Comparison of SSE when k=3

SSE when K=4	**My Implementation**	**Off-the-shelf Implementation**
Cluster 1	1.0764850881678976	1.0764850881678976
Cluster 2	0.5004805751481358	1.4705341945503403

Cluster 3	1.8446030116907752	1.8446030116907752
Cluster 4	1.4705341945503403	0.5004805751481358
Total	4.89210	4.8921028695571485

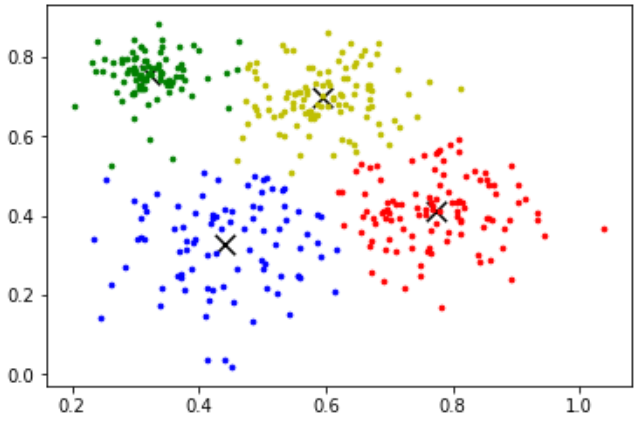
Comparison of SSE when k=4

SSB	**My implementation**	**Scikit learn**
K=3	20.7396662919	20.742977
K=4	24.412238265	24.266163739

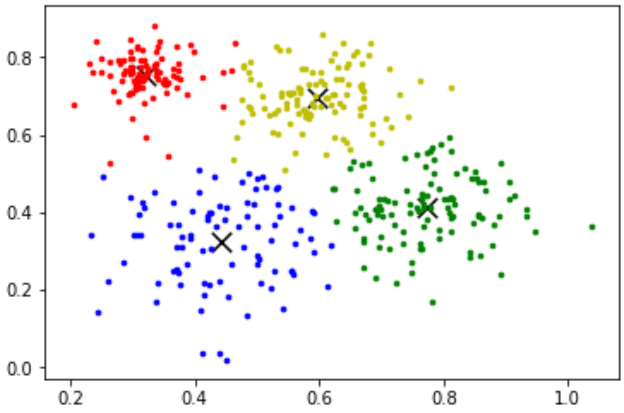
SSB analysis with k=3 and k=4

From the tables we can conclude that the SSE and SSB values are in agreement for both the implementation. Moreover, scikit-learn's implementation of K-means has lower SSE and higher SSB when k is taken as 4 suggesting that k=4 is a better choice. The same result has been derived from my implementation as well.

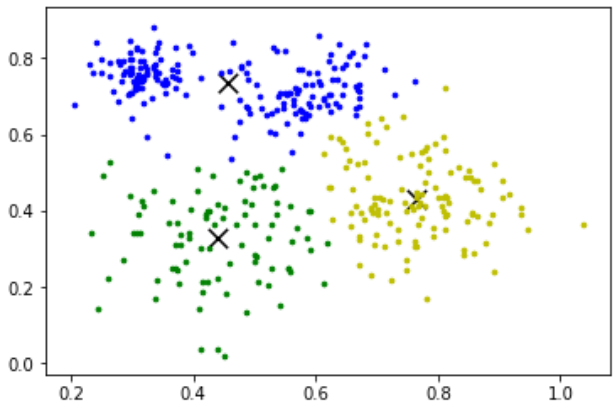
Predicted clusters with k=4



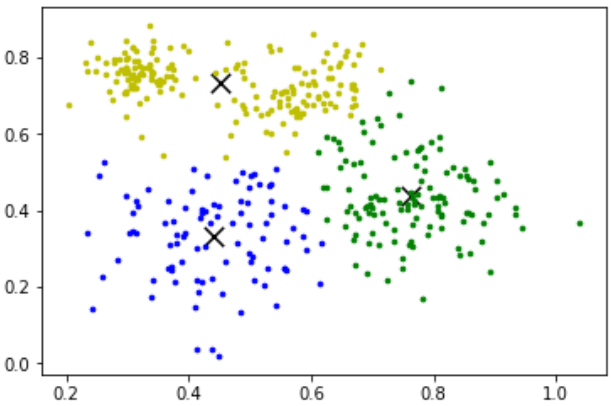
Off-the-shelf clusters with k=4



Predicted clusters with k=3



Off-the-shelf clusters with k=3



The scatter plots above help to visualize the clusters created with k=3 and k=4 using my implementation and the off-the-shelf implementation. For both the k values we can see that no significant difference can be seen in the clusters created using both the implementations.

5. CLUSTERING WINE-QUALITY DATASET

The off-the-shelf implementation in the sci-kit learn machine learning library for python has been applied to the wine-quality data set. The 'quality' attribute has been excluded while computing the clusters.

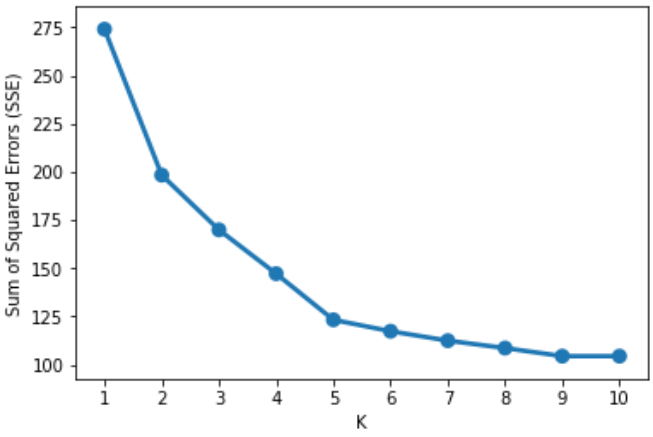
The wine quality data set has attributes with various ranges. Since we have used Euclidean distance as the proximity measure, the K-means will be sensitive to feature scaling. So, I have normalized all the attributes of the wine-quality data set to fit into range 0-1. I have used *minmaxscaler* from the pre-processing library available in scikit-learn.

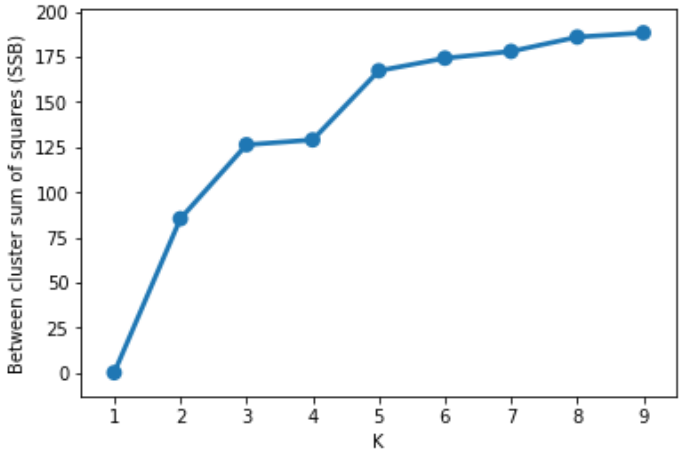
The wine data set also has many data points that are outliers. K-means algorithm is also sensitive to outliers. The outliers have been detected using the 1.5IQR Rule. The data points falling beyond 1.5*Q3 or below 1.5*Q1 are considered to be outliers. These values have been replaced. It has been observed that after handling the outliers, the SSE has improved considerably.

The following table indicates the SSE and SSB values obtained for different k values.

K	SSE	SSB
2	198.32897424204785	85.3497732469
4	147.157618823	129.064230895
6	123.22285701240155	167.3686831
8	112.40961706798255	174.346578122
9	108.567424341	186.06848899

I have created a point plot to understand how SSE is varying by changing K. The plot on the left shown below indicates SSE values for 10 different values of K ranging from 1 to 10. I have created a similar plot for SSB Vs K and is shown on the right side.





From the graph we can see that significant decrease in SSE cannot be seen when k is increased beyond 6. We can see that from k=5, SSE has almost remained constant. Similar is the case with SSB. After k=5 or 6 the increase in SSB is not significant. This suggests that k=5 or k=6 is a good value for number of clusters for the wine-quality data set.

```
SSE Cluster 1 :  23.163408958883526
SSE Cluster 2 :  22.47896085665664
SSE Cluster 3 :  17.69077890497543
SSE Cluster 4 :  17.0681033017401
SSE Cluster 5 :  26.60693386732769
SSE Cluster 6 :  16.214671122818164
Over all SSE :  123.22285701240155
```

Cluster wise SSE with k=6 for wine data set

K=6	**Actual Clusters**					
Predicted Clusters	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Cluster 1	2	4	79	115	51	4
Cluster 2	7	24	208	127	20	0
Cluster 3	0	10	143	102	6	1
Cluster 4	0	3	14	102	88	9
Cluster 5	1	4	203	83	9	0
Cluster 6	0	8	34	109	25	4

Cross tabulation matrix with k=6 for wine-quality data set

The table above shows the cross-tabulation matrix when k is chosen to be 6. When compared to the cross-tabulation matrices created with different values of k, the above matrix seems to divide points in a better way. So, we can say that k=6 is a good choice for the wine data set. Increasing k beyond this will decrease SSE and increase SSB slightly but we do not get good clusters as observed from cross tabulation matrices. So, we consider k=6 as a good value for the number of clusters for wine-quality data set. I have tried to visualize the clusters using parallel coordinate plot but I have not included them here. Since the data set has 12 features, the plot seems to be messy and does not allow proper visualization of clusters.