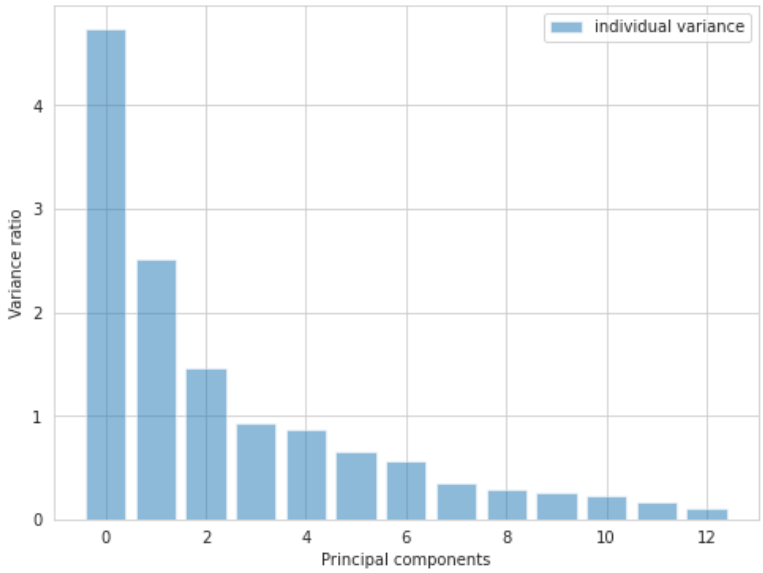
Pattern Recognition and Machine Learning
Report for : Lab 9

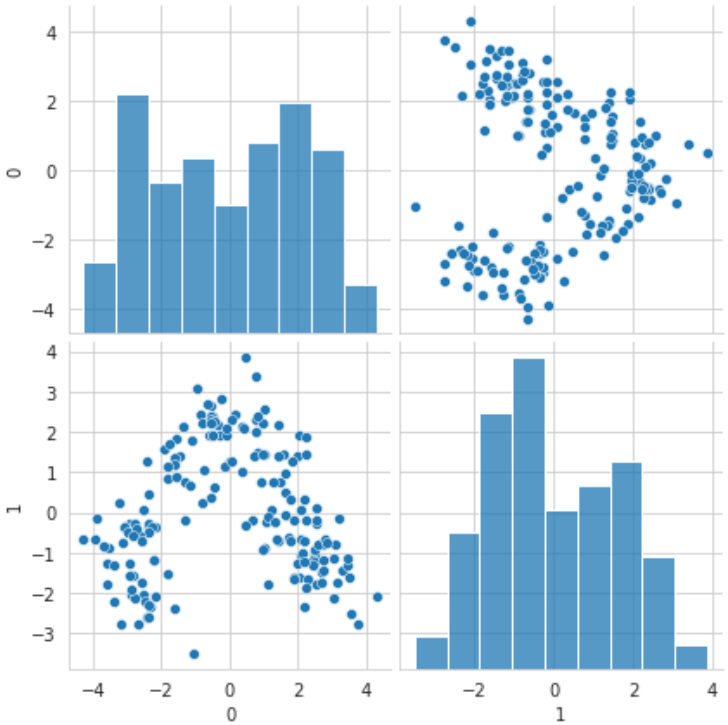
Name : Khushi Parikh
Roll No : B20EE029

Question 1:

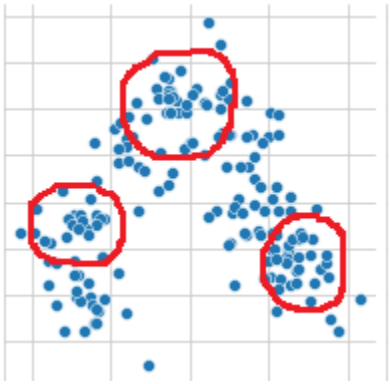
- **(Pre-processing)** After importing the necessary libraries, I imported the data. I looked at the features of the data - data types, shape, scaled the data and divided it into X(features) and Y(labels). The data had 178 rows and 14 columns.
- **(1.1)** The dimension reduction technique used is principal component analysis. I chose the top 2 features obtained from the graph -



- Visualizing the reduced data -



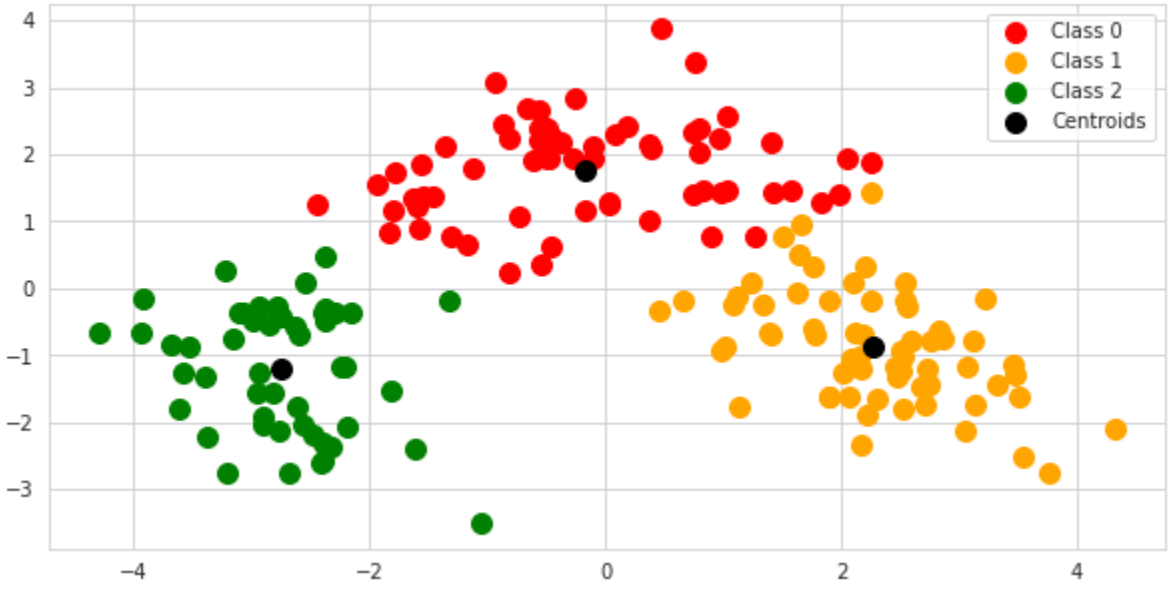
- Looking at this data, it seems that the data can be divided into three clusters, one with the left data points, one with the top data points and one with the right side data points as three clusters are formed as follows -



- **(1.2)** Using value of clusters as 3, following are the predicted class labels -

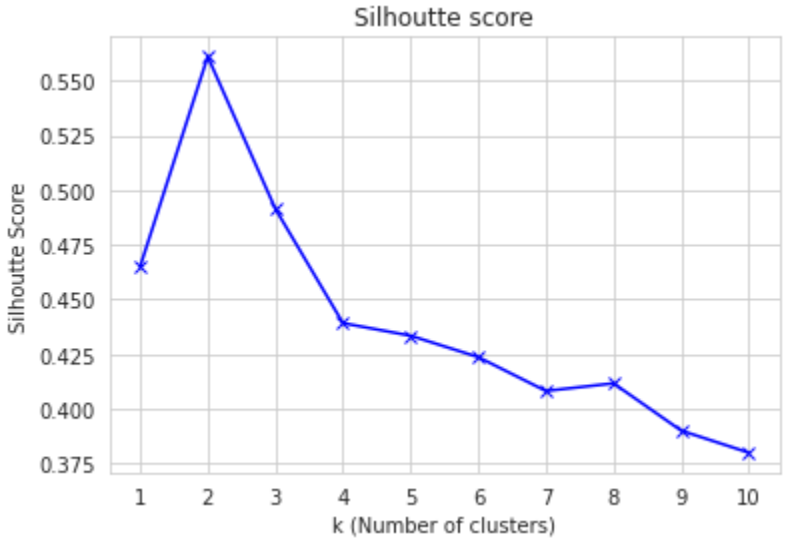
```
Predicted Class labels :-
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

Following is the graph visualization with centroids -



- **(1.3)** Checking silhouette score for number of clusters in range 2 to 11, we get that for 3 clusters, we get the maximum score -

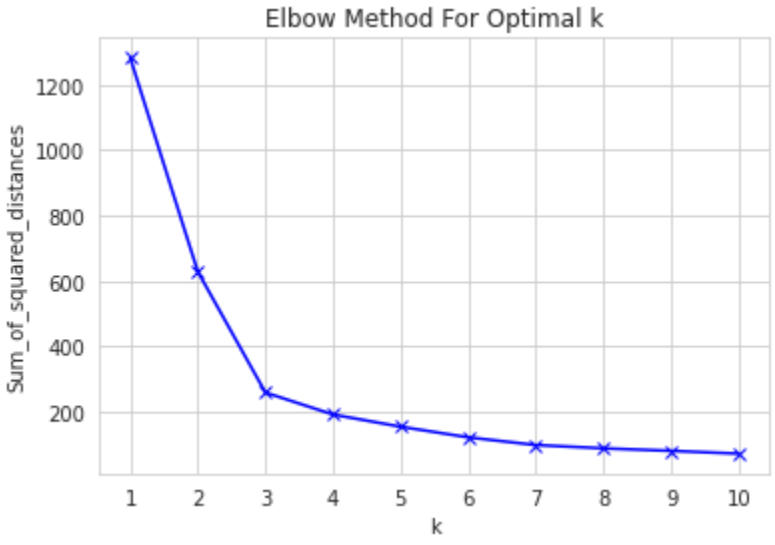
```
Silhouette Score for clusters = 2 : 0.4649140908920152
Silhouette Score for clusters = 3 : 0.5610505693103246
Silhouette Score for clusters = 4 : 0.4914213395710318
Silhouette Score for clusters = 5 : 0.439095014033523
Silhouette Score for clusters = 6 : 0.4333614590199158
Silhouette Score for clusters = 7 : 0.423570606607218666
Silhouette Score for clusters = 8 : 0.40808048784150297
Silhouette Score for clusters = 9 : 0.4115651897285956
Silhouette Score for clusters = 10 : 0.38993709690700834
Silhouette Score for clusters = 11 : 0.3798547872196581
```



Silhouette score is used to see how well separated the clusters are, more the score - more well separated the clusters are. A silhouette score of 1 means perfect distinction of clusters.

- **(1.4)** To find the best number of clusters, we can also use the elbow method. Graph -

The shape of the graph forms an arm. Imagining this to be an actual hand, the place where the elbow would be gives the optimal value of k and hence the name - Elbow method. The plot represents the sum of squared distances of each point from their cluster mean.

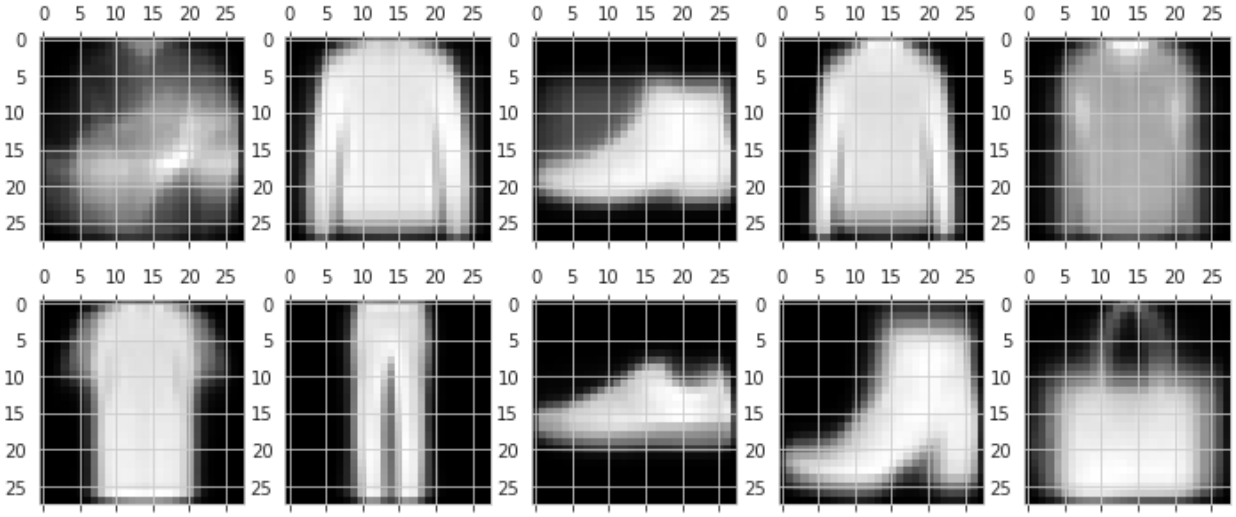


Question 2:

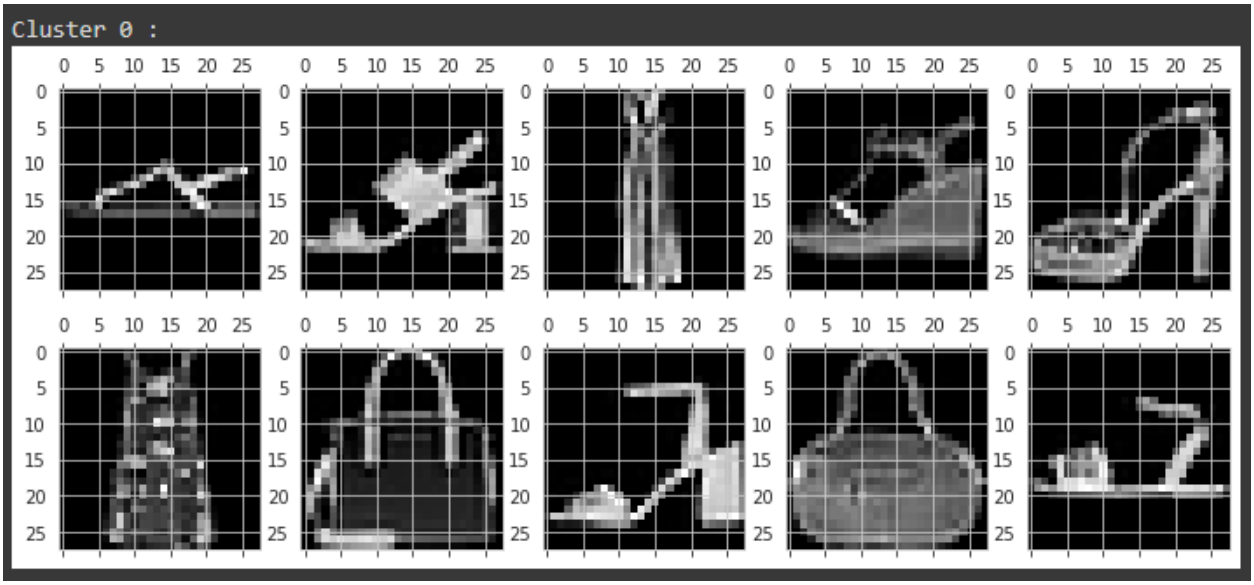
- **(2.1 + 2.2)** KMeans clustering was implemented from scratch as follows -
- The maximum number of iterations, number of clusters and initial centroids was taken from the user. While we do not reach the maximum iterations, we first get the closest centroid for each point in the dataset and assign then a particular cluster number.
- The new cluster centroid is calculated by taking the mean of the points in a particular cluster.
- This process is repeated until either the new centroids are the same as the previous ones (convergence - best possible centroids) or the maximum number of iterations is reached.
- **(2.3)** The range of the input, being an image, varies from 0 to 255. An array of random integers of size (10,784) was generated as centroids.
  Output value counts -

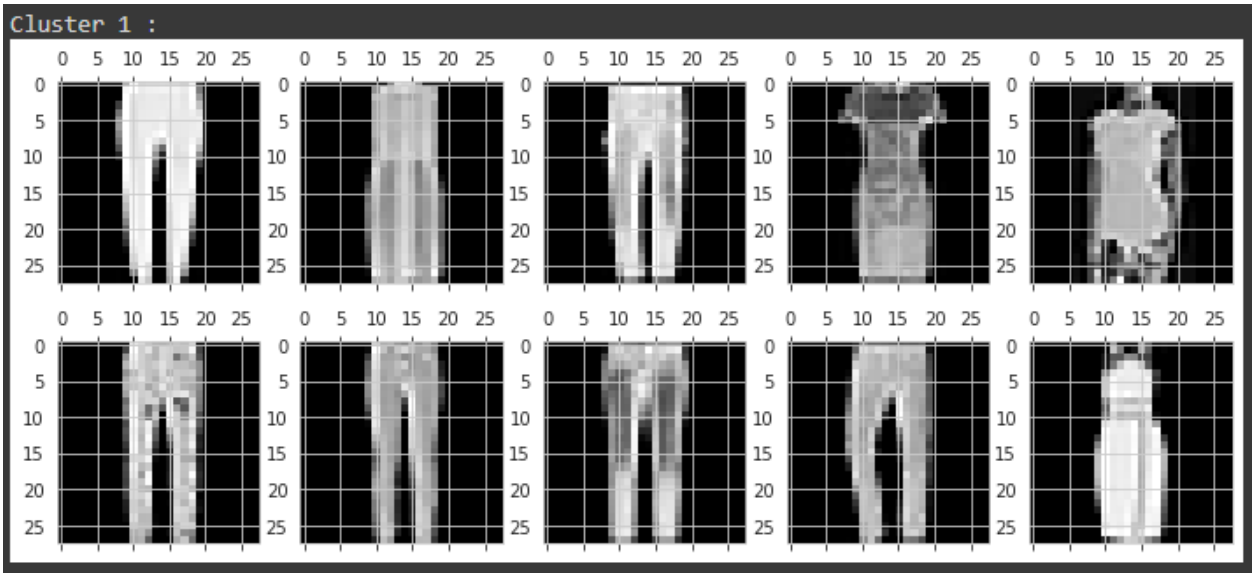
```
1    1383
0    1250
4    1205
5    1154
7    1065
8     983
2     863
6     646
3     520
9     403
Name: cluster, dtype: int64
```

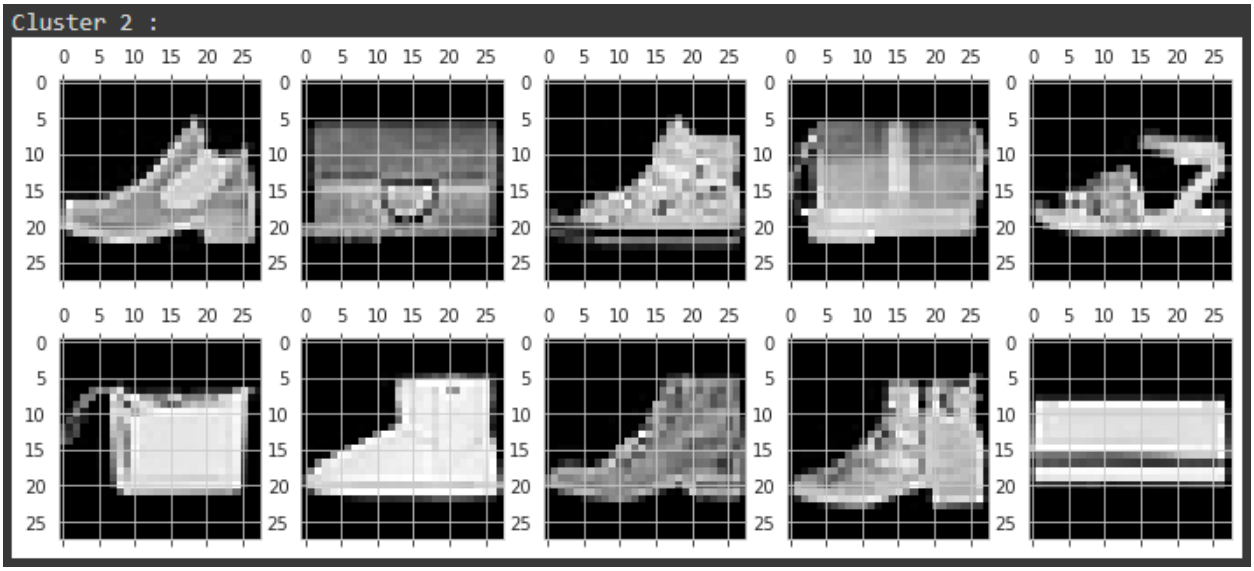
- **(2.4)** Visualized cluster centers -



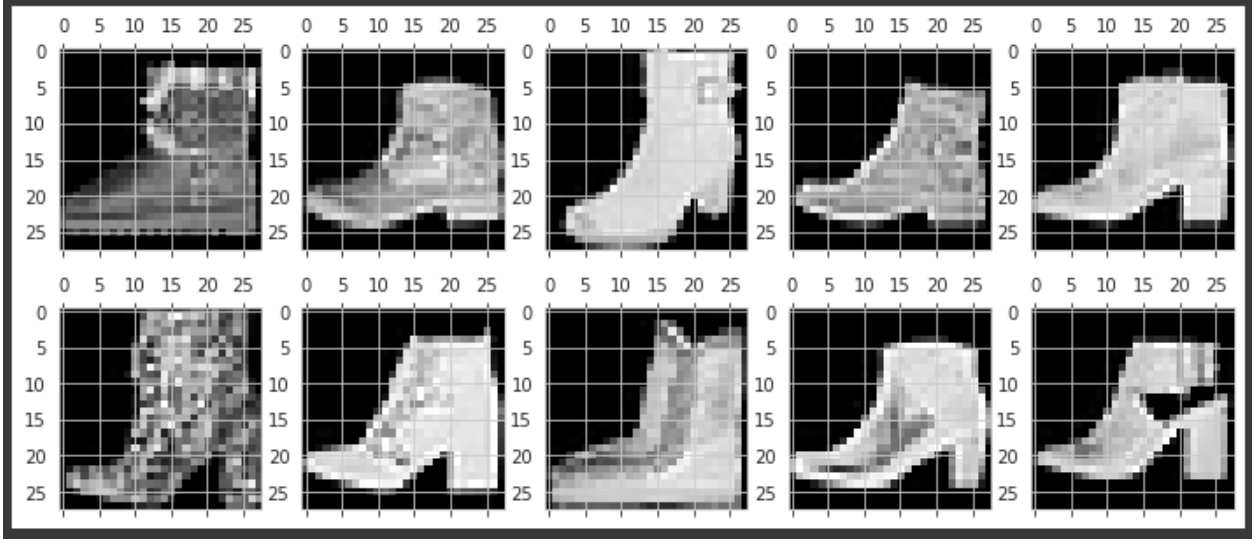
- **(2.5)** 10 images corresponding to each cluster -



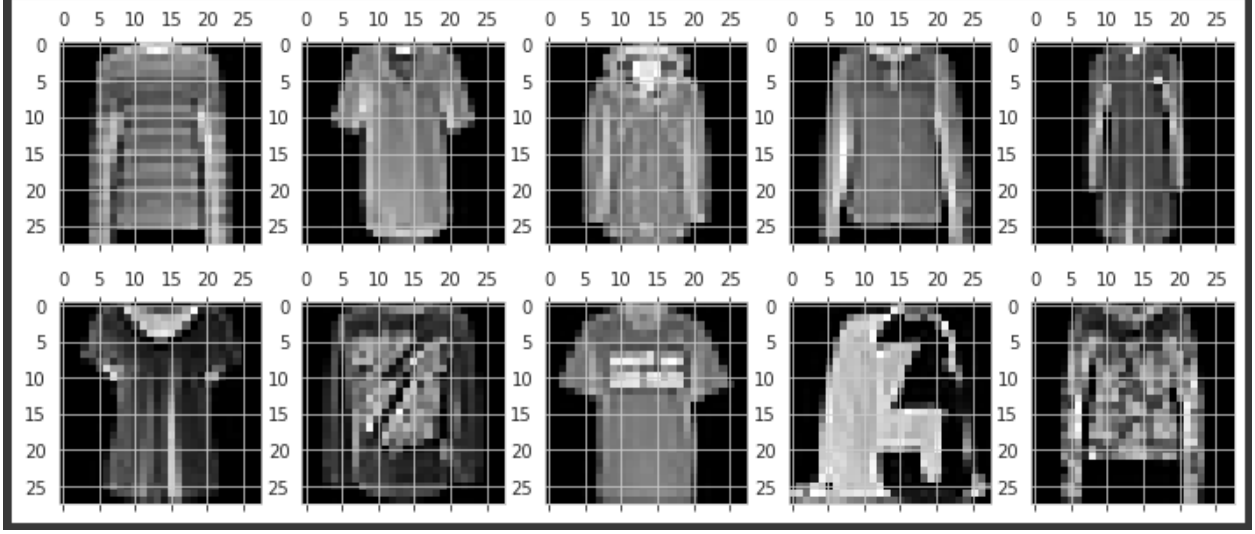




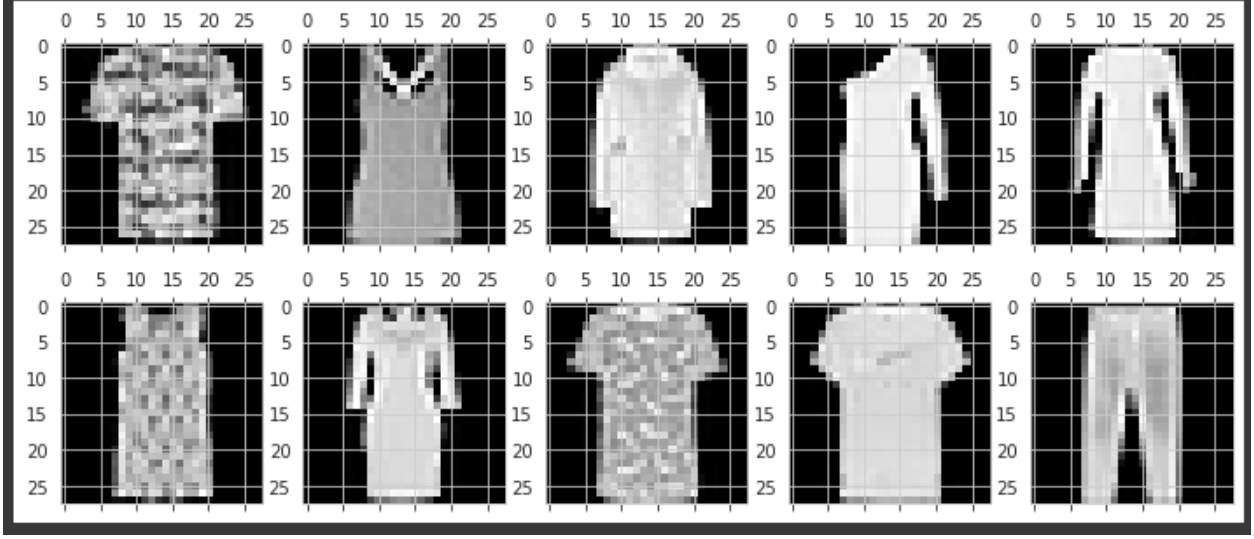
Cluster 3 :



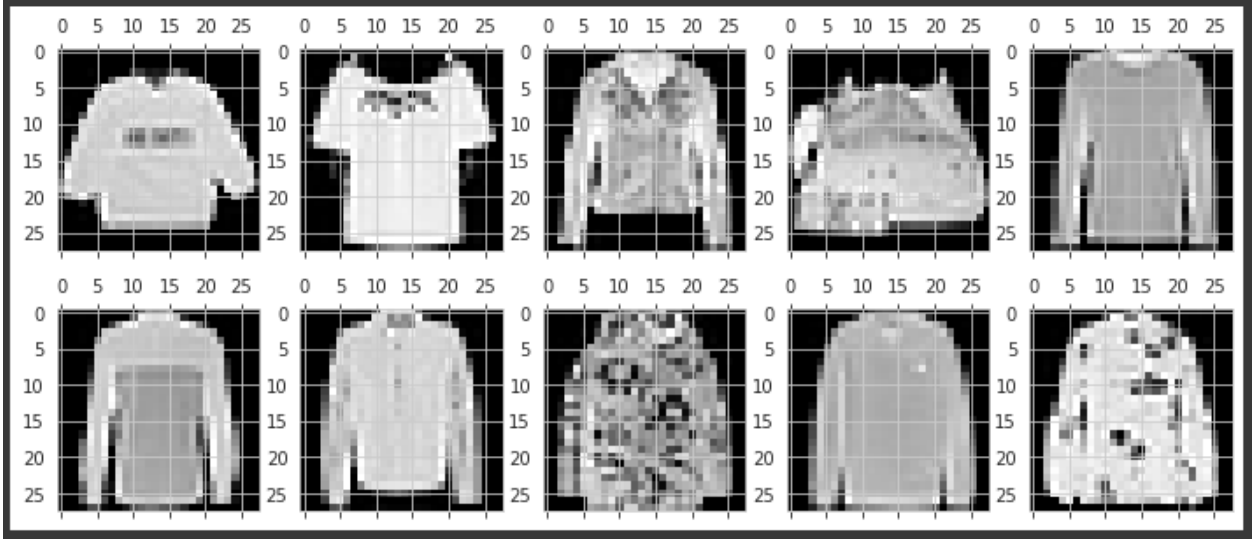
Cluster 4 :



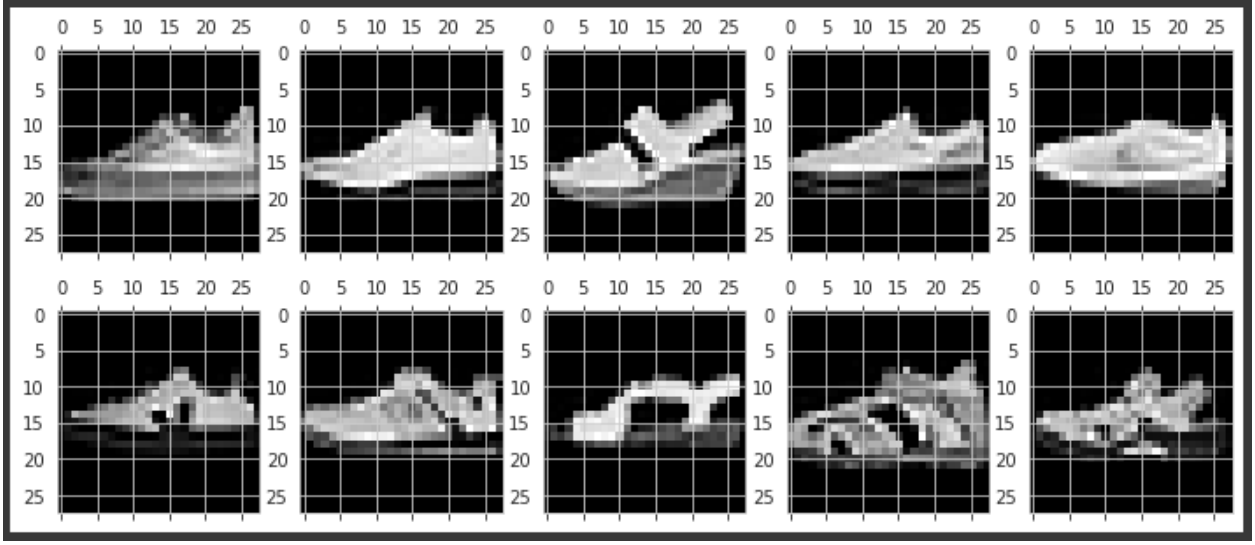
Cluster 5 :



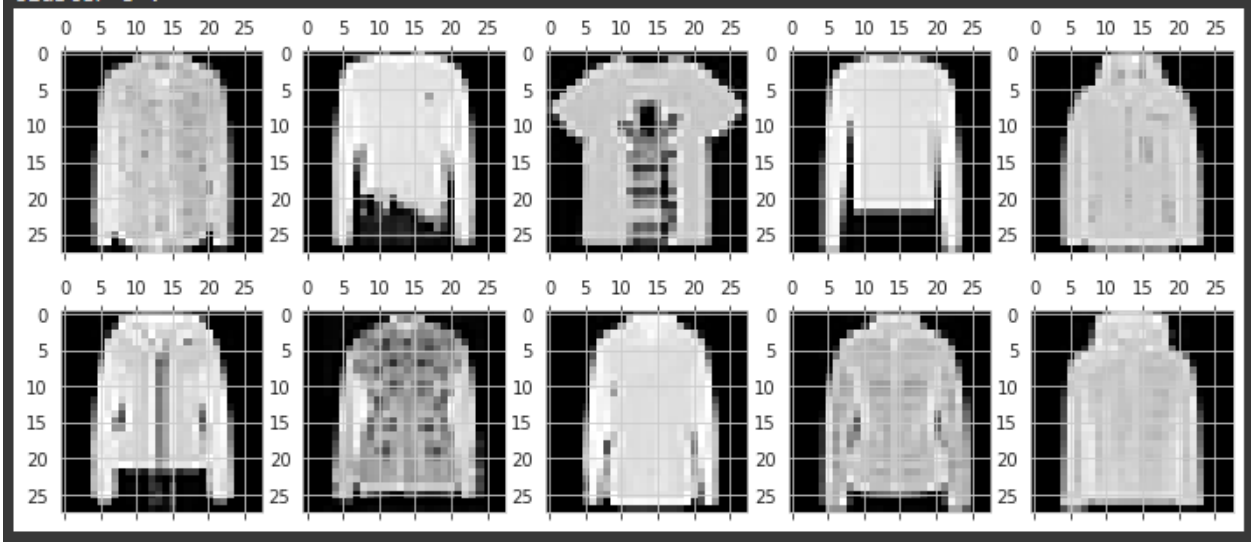
Cluster 6 :

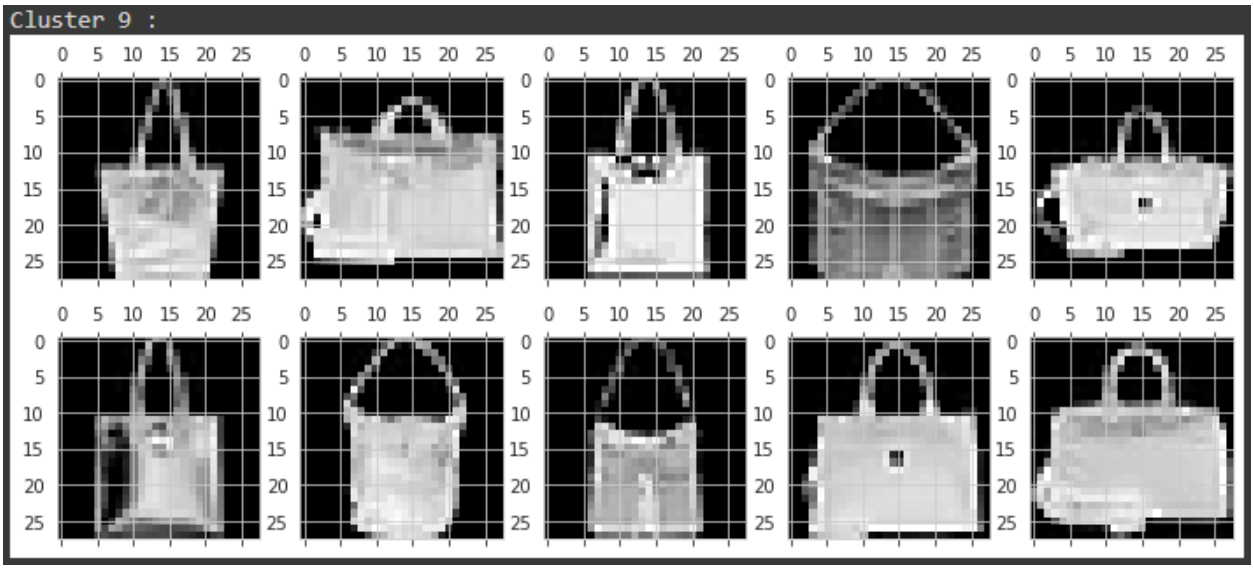


Cluster 7 :



Cluster 8 :

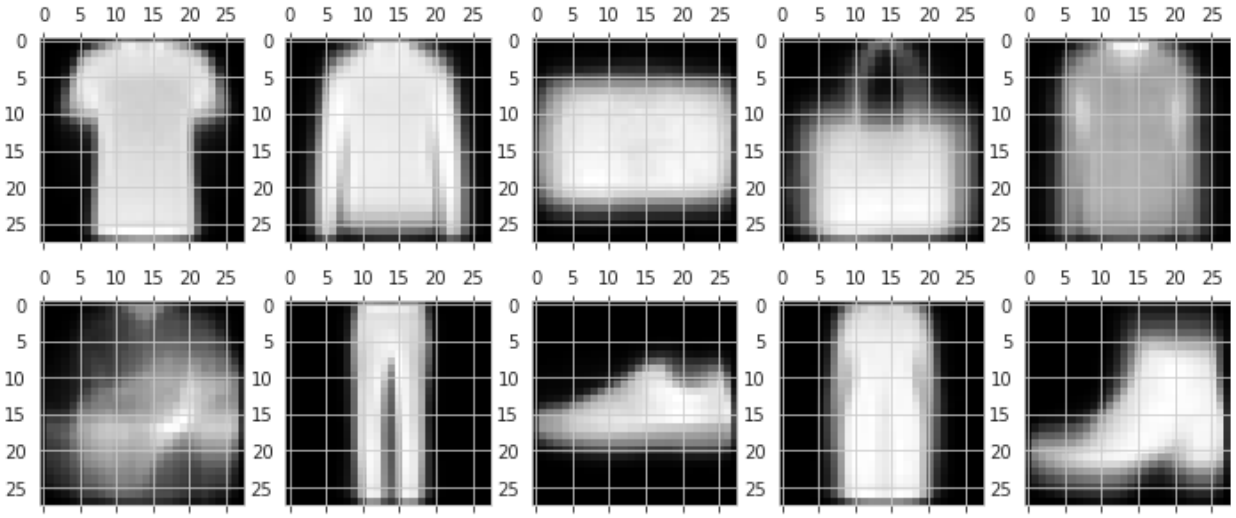




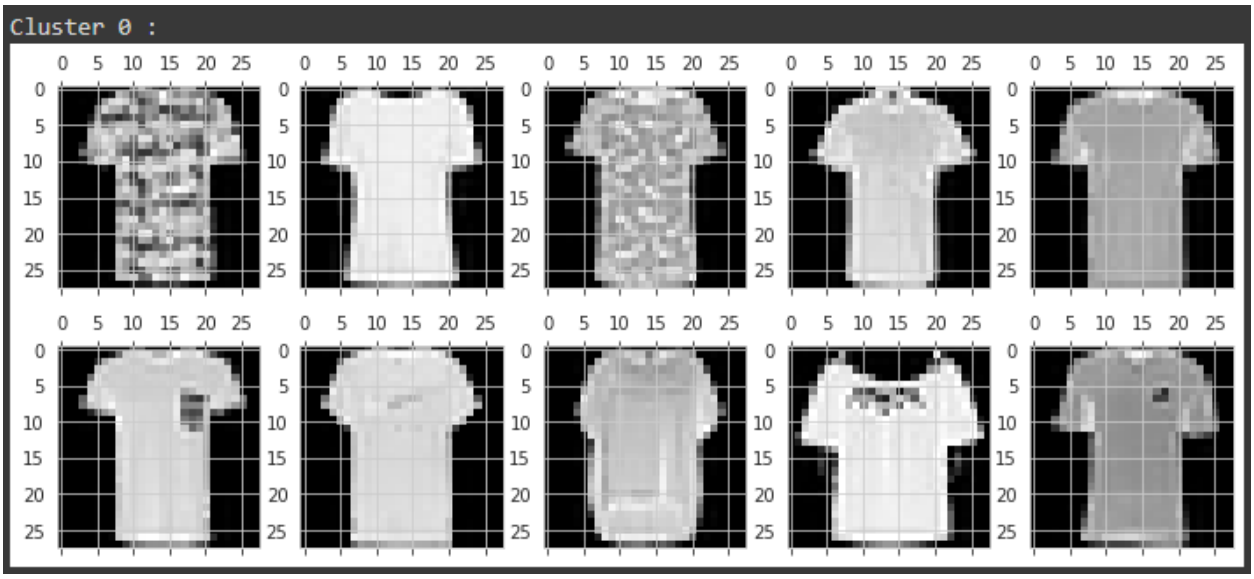
- **(2.6)** 1 image was chosen from each cluster, a total of 10 images. Value Counts -

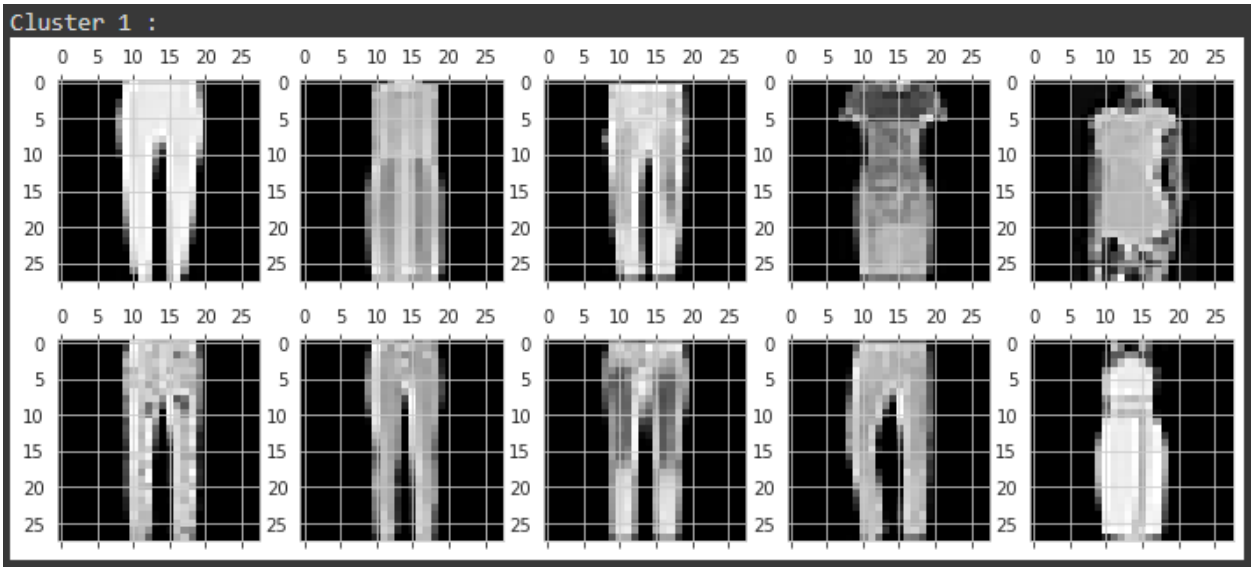
```
2     345
8     394
0     616
3     807
9     937
4    1173
1    1186
7    1205
5    1313
6    1496
Name: cluster, dtype: int64
```

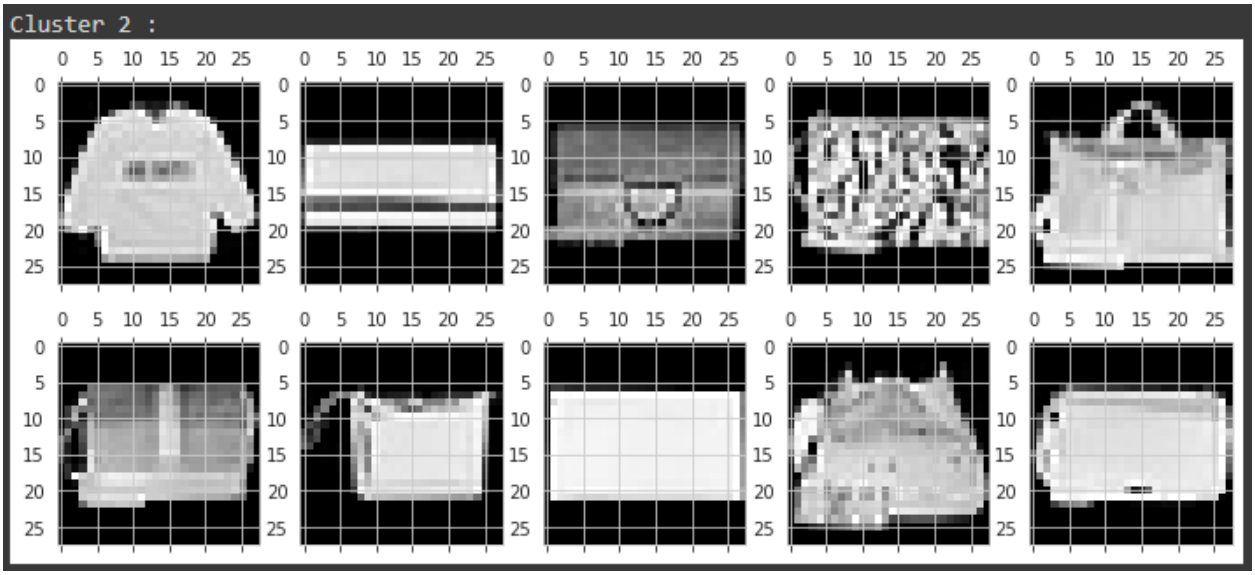
- Cluster Centers -



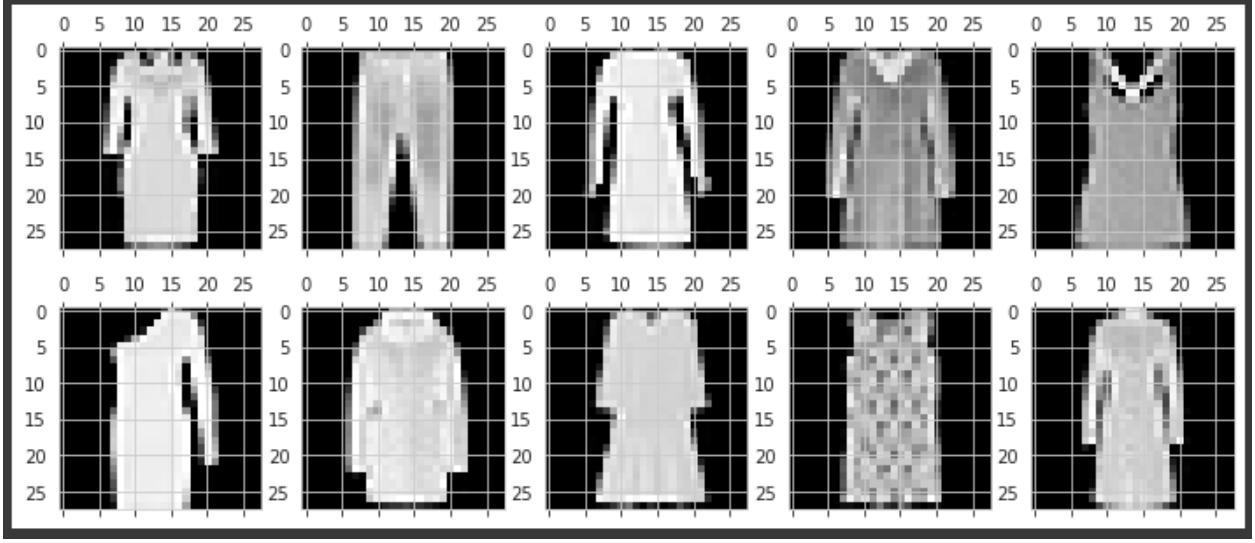
- **(2.7)** Cluster visualizations -



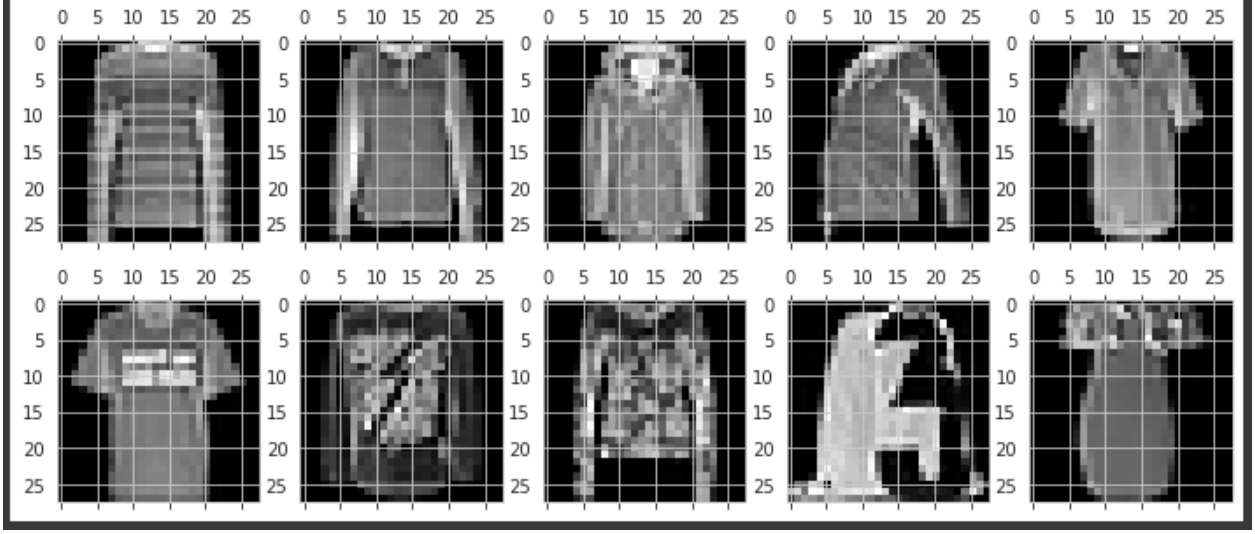




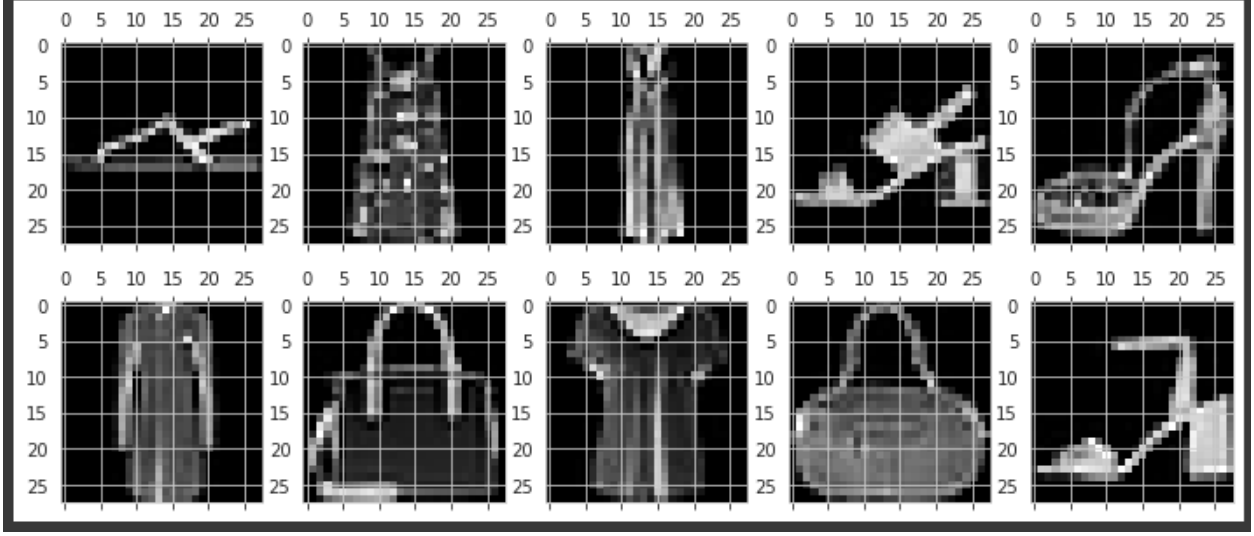
Cluster 3 :



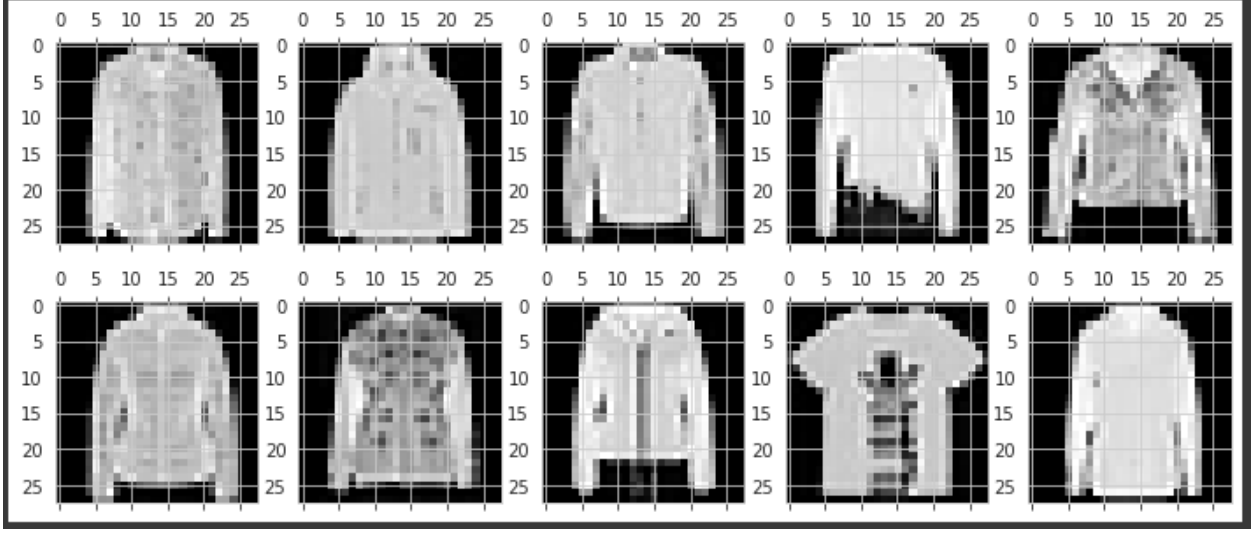
Cluster 4 :



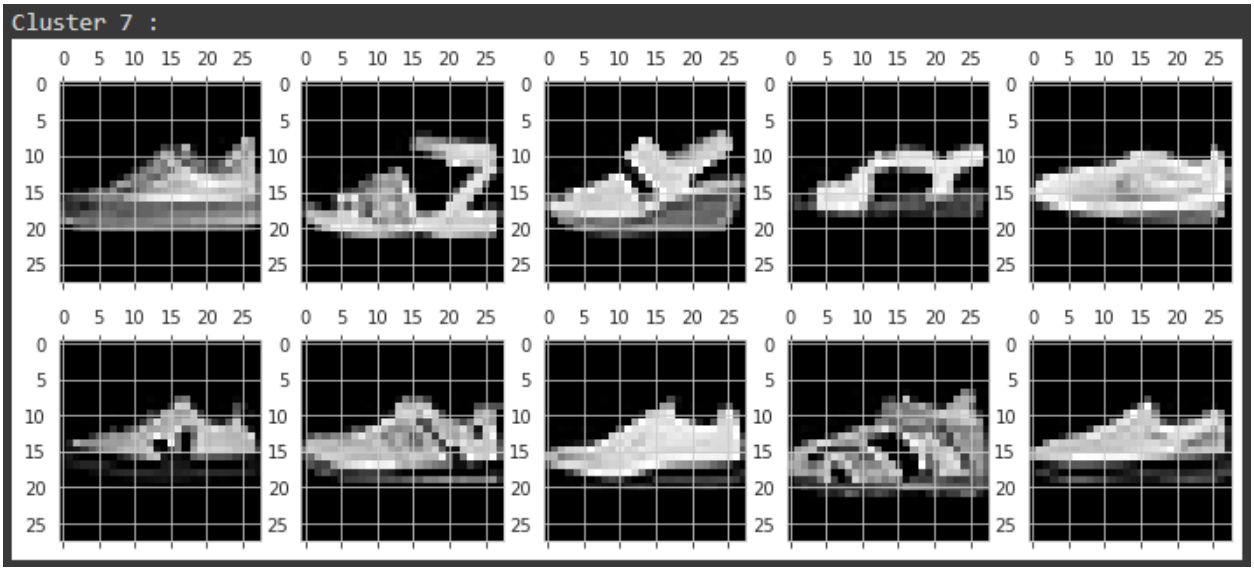
Cluster 5 :



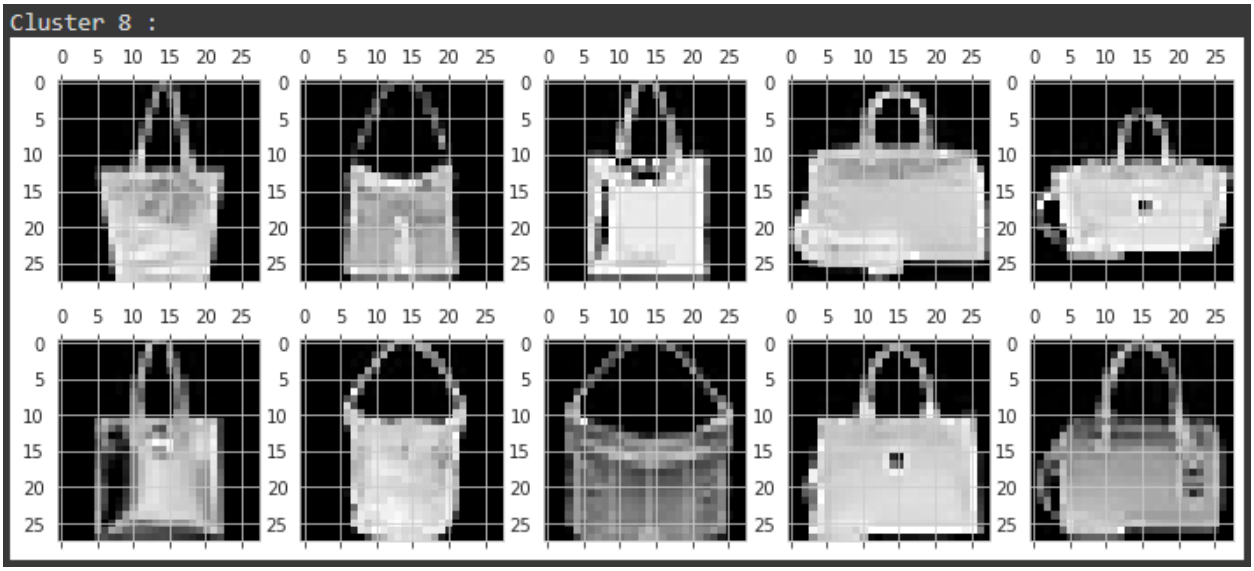
Cluster 6 :

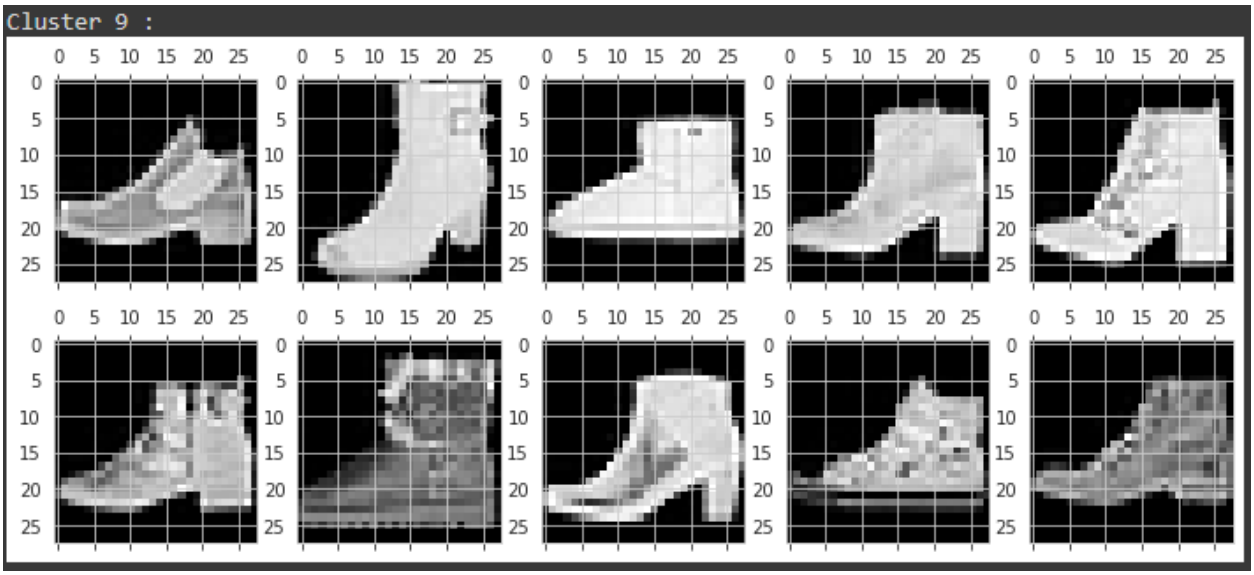


Cluster 7 :



Cluster 8 :





- **(2.8)** Clusterwise sse for randomly selected indexes -

```
[17152777922.040844,
 13017300830.96737,
 7155391997.855265,
 5543063472.022339,
 11637967821.252577,
 13146292890.080254,
 5336902517.427129,
 10974774257.616354,
 8696729923.202917,
 4719751949.028095]
```

Clusterwise sse for indexes selected from classes -

```
[5880546823.395722,
 11713921221.480042,
 3225897818.1066165,
 9972098255.27228,
 10291618795.365217,
 17862090828.88845,
 13390944622.449337,
 12390356785.368975,
 4734607499.3945675,
 7930975215.13806]
```

Comparing the sse to see which is better, the one with the less sse is-

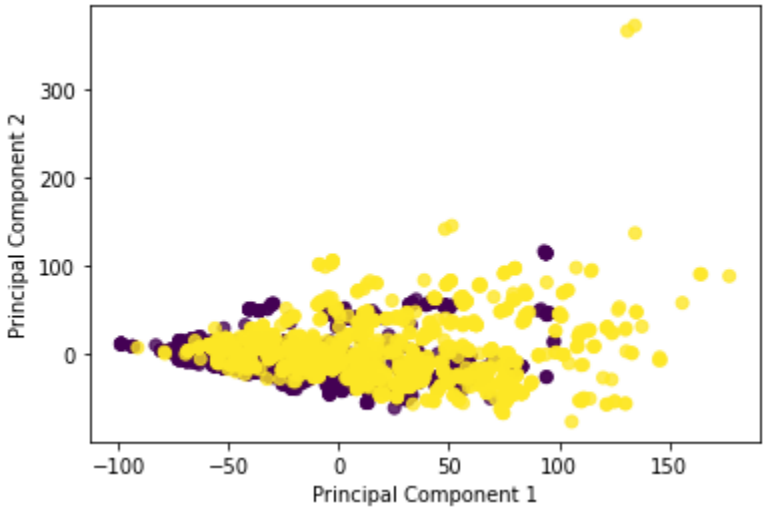
```
print(sum(clusterwise_sse))
print(sum(clusterwise_sse1))
sum(clusterwise_sse)/sum(clusterwise_sse1)
```

```
97393057864.85927
97380953581.49313
1.000124298262863
```

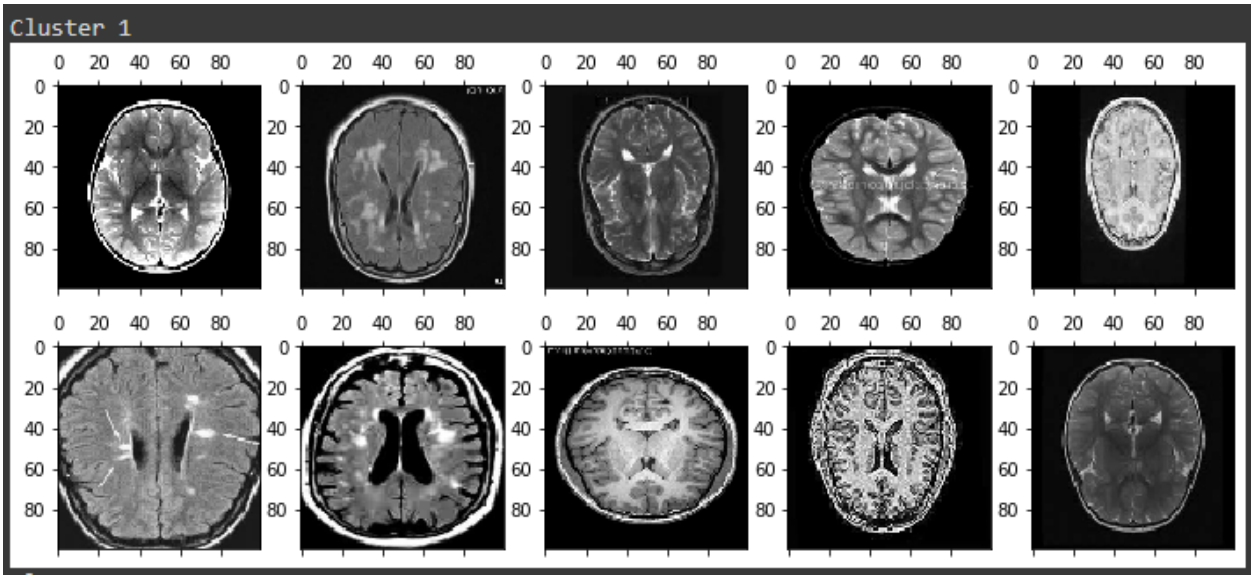
- This shows that the sum of sse for second initialisation(class-wise initialisation) is less than that of the first initialisation(randomly selected initialisations). Hence, this shows that the initializing using one point from each cluster creates better clusters for the same number of iterations as it has a less sse(sum of squared errors).

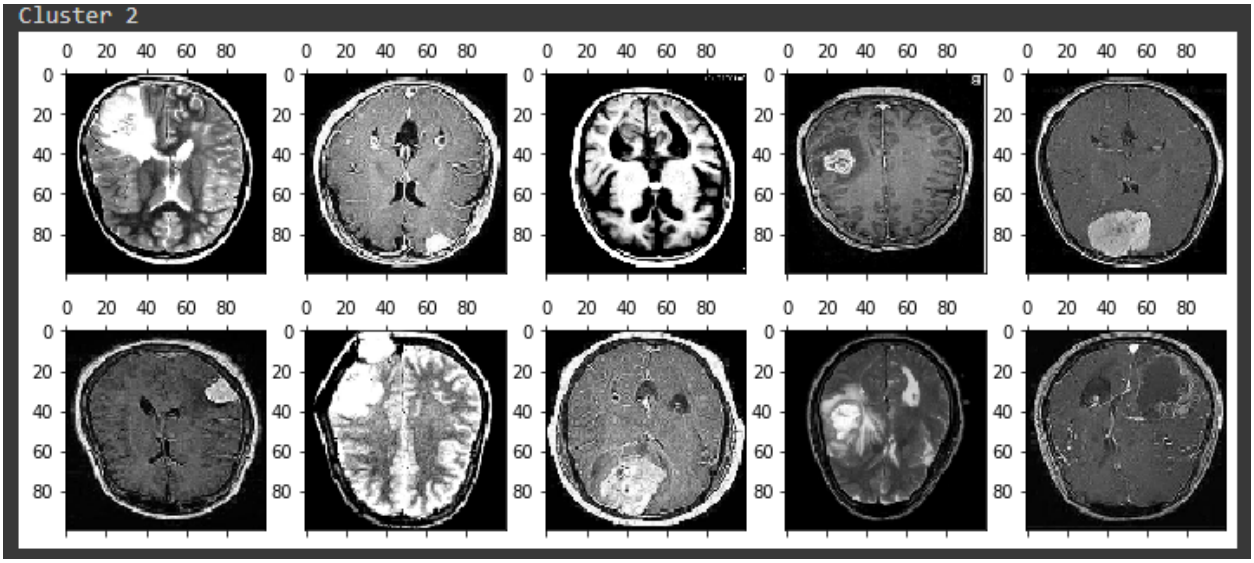
Question 3:

- **(3.1)** All images were imported and reshaped to a size of (100,100) and then this array was flattened. Hence the pixels of one image make one row of the final dataframe. There were a total of 3000 images, 1500 from yes and 1500 from no. This whole data was then normalized using standard scaler.
- **(3.2)** The dimensions reduction technique used was Principal Component Analysis. The two major components were chosen so that the final data could be reduced to it. The final data was then visualized as follows -



- **(3.3)** The communities were visualized as follows -



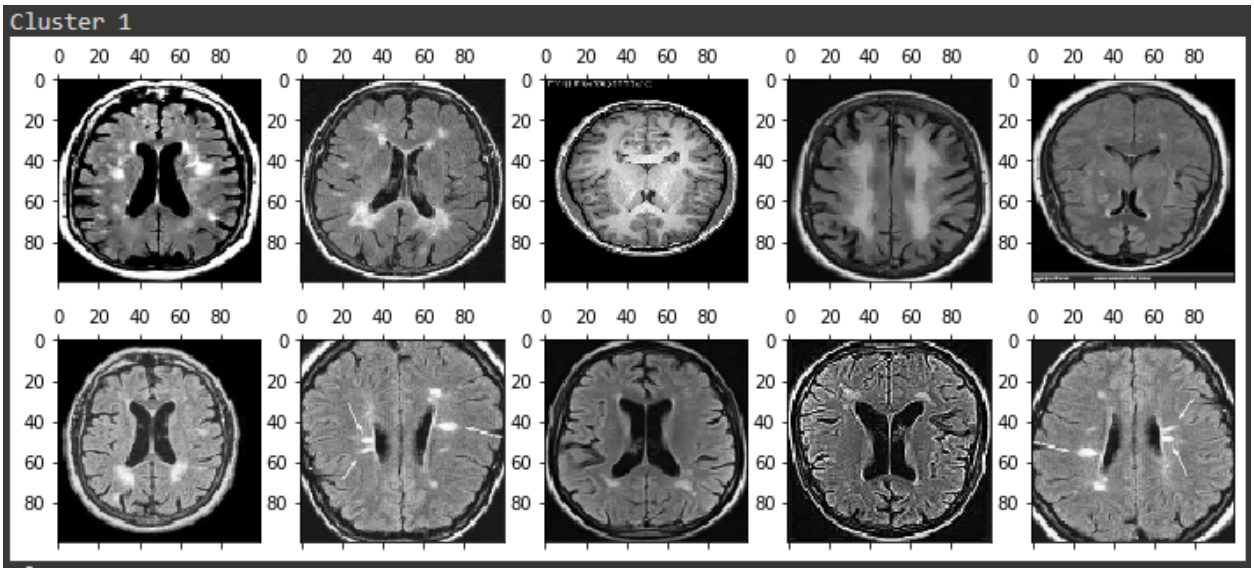


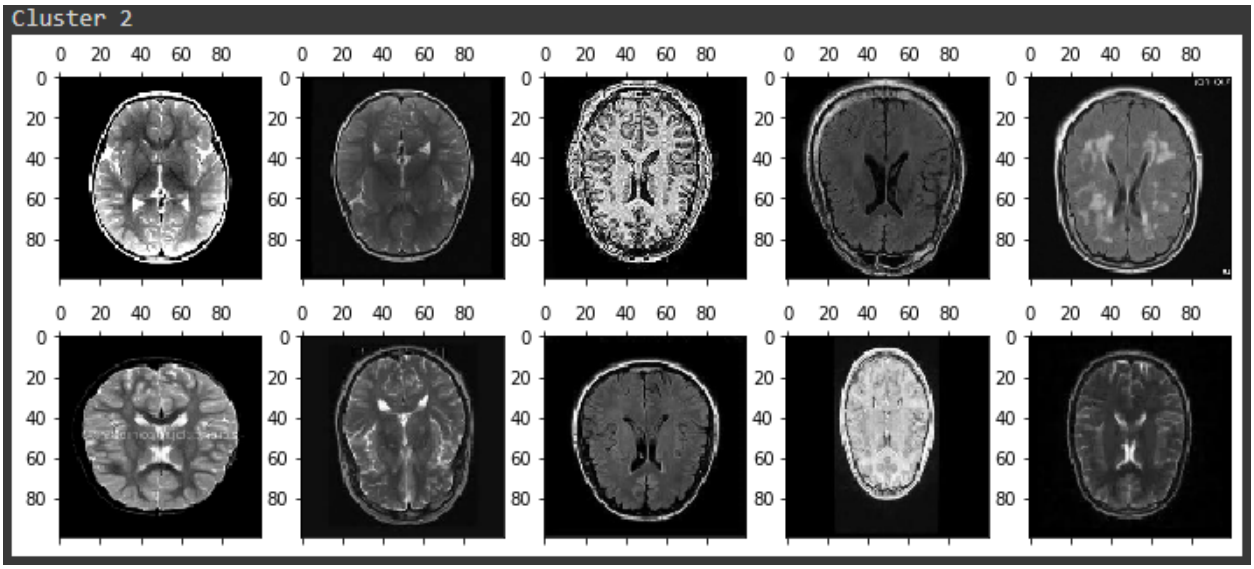
- **(3.4)** Agglomerative clustering was applied after importing it from sklearn. Result -

```
Predicted Class labels :
(array([0, 1], dtype=int64), array([1250, 1750], dtype=int64))
Silhouette Score : 0.14999214652657833
```

  There were two predicted labels - 0 and 1 with counts of its values as 1250 and 1750 respectively.

- Output clusters from agglomerative clustering are -



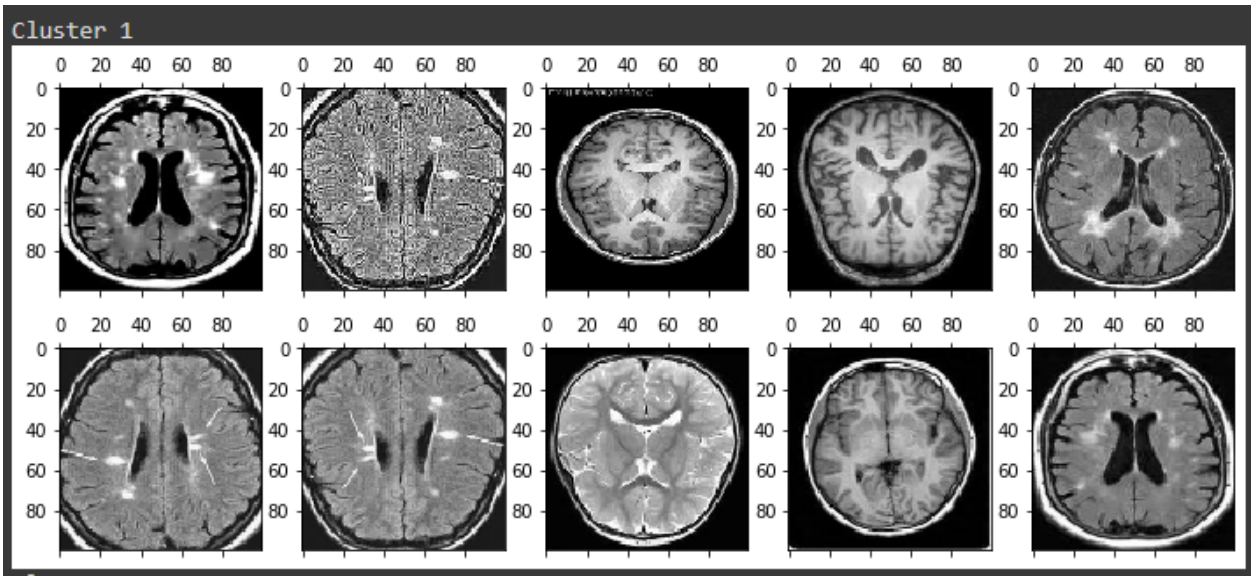


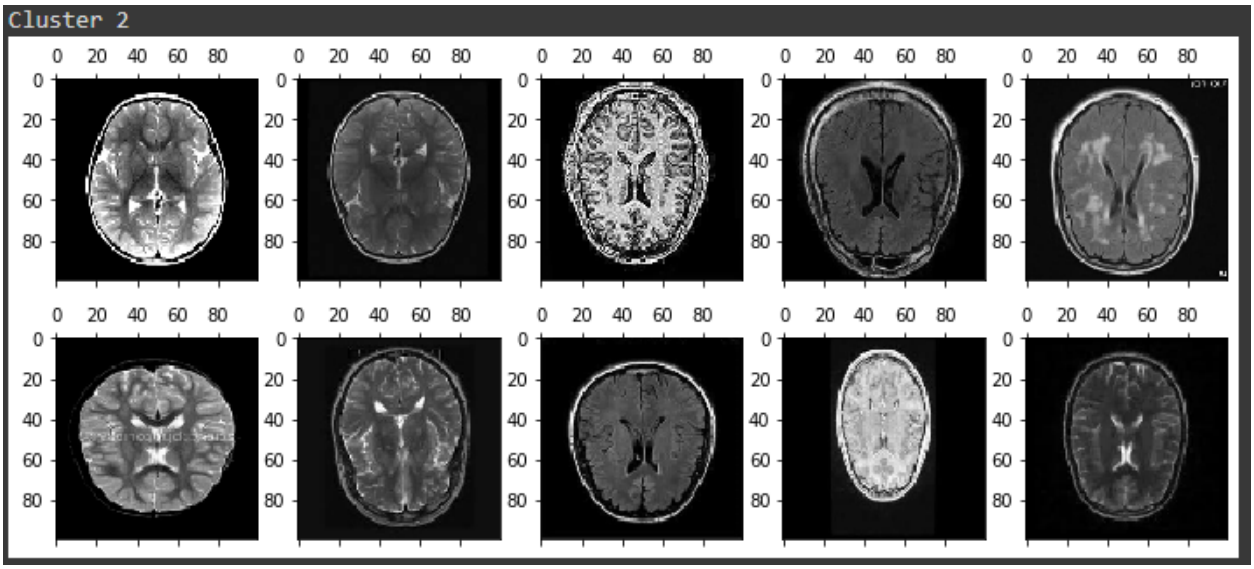
- **(3.5)** Using KMeans, we get -

```
Predicted Class labels :
(array([0, 1]), array([1115, 1885], dtype=int64))
Silhouette Score : 0.18737755126152753
```

There are two output labels, 0 and 1 with counts of their values as 1115 and 1885 respectively.

- Output clusters from KMeans -

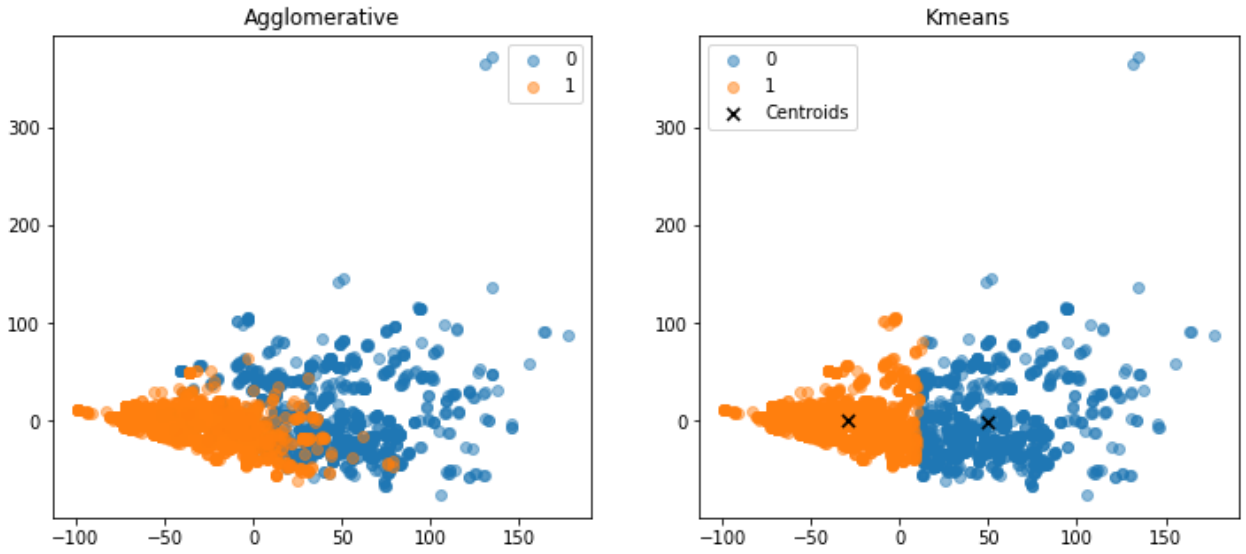




- Comparing the two methods - agglomerative clustering and KMeans using parameters such as homogeneity score, completeness score and silhouette score and v-measure score, we get -

	Agglomerative	Kmeans
Silhouette :	0.14999214652657833	0.18737755126152753
Completeness :	0.09997134460778857	0.08551951606169576
Homogeneity :	0.09795879714157775	0.08140972288157741
V_measure :	0.0989548391149624	0.08341402797523288

- From output visualizations of the clusters, we can see that KMeans has done a better job at separating two clusters -



- As we can see both from the scores and visualizations, KMeans has done a very good job at separating the two clusters, compared to agglomerative clustering.