Maniya Yash Rajeshbhai
B20CS033

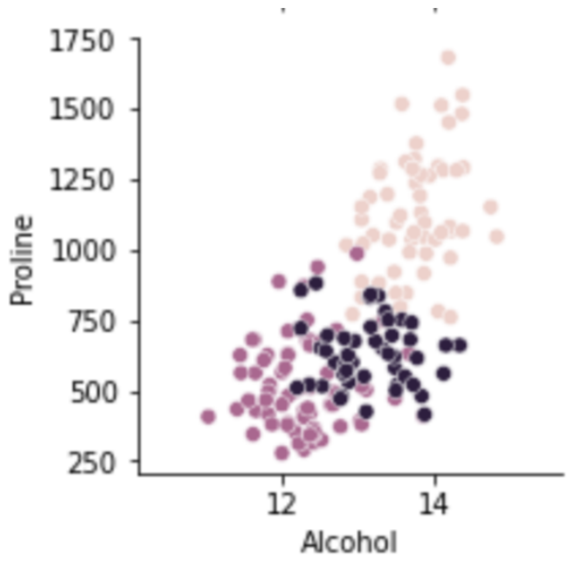
ML Lab 9 Report

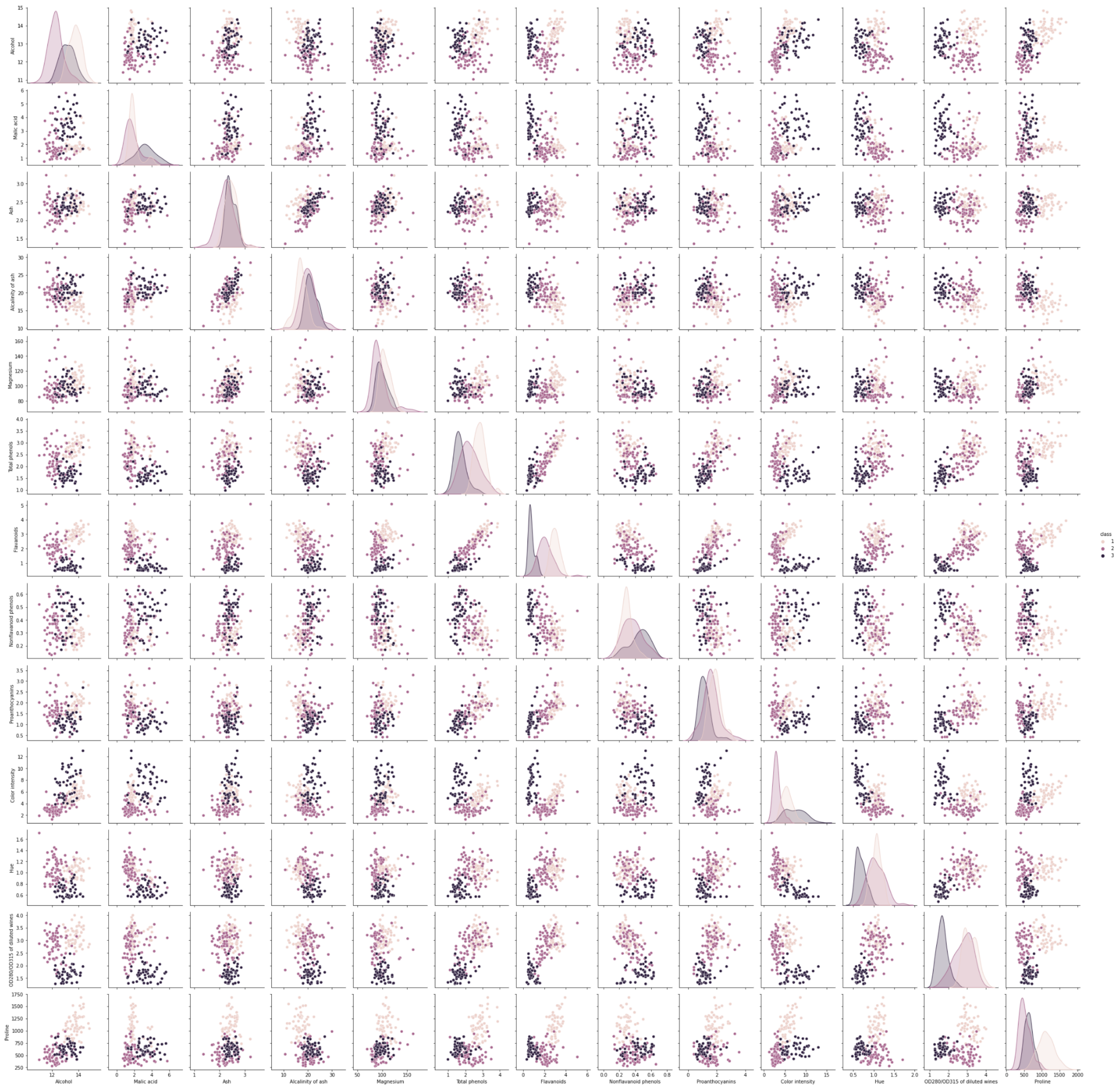
Question 1 :

1. Dimensional Reduction, Visualisation, optimal K:

- Did PCA for dimensionality reduction, and visualised the data using pairplot.
- For most of the pairs, it is visible that 3-4 clusters can be formed which seems optimal.

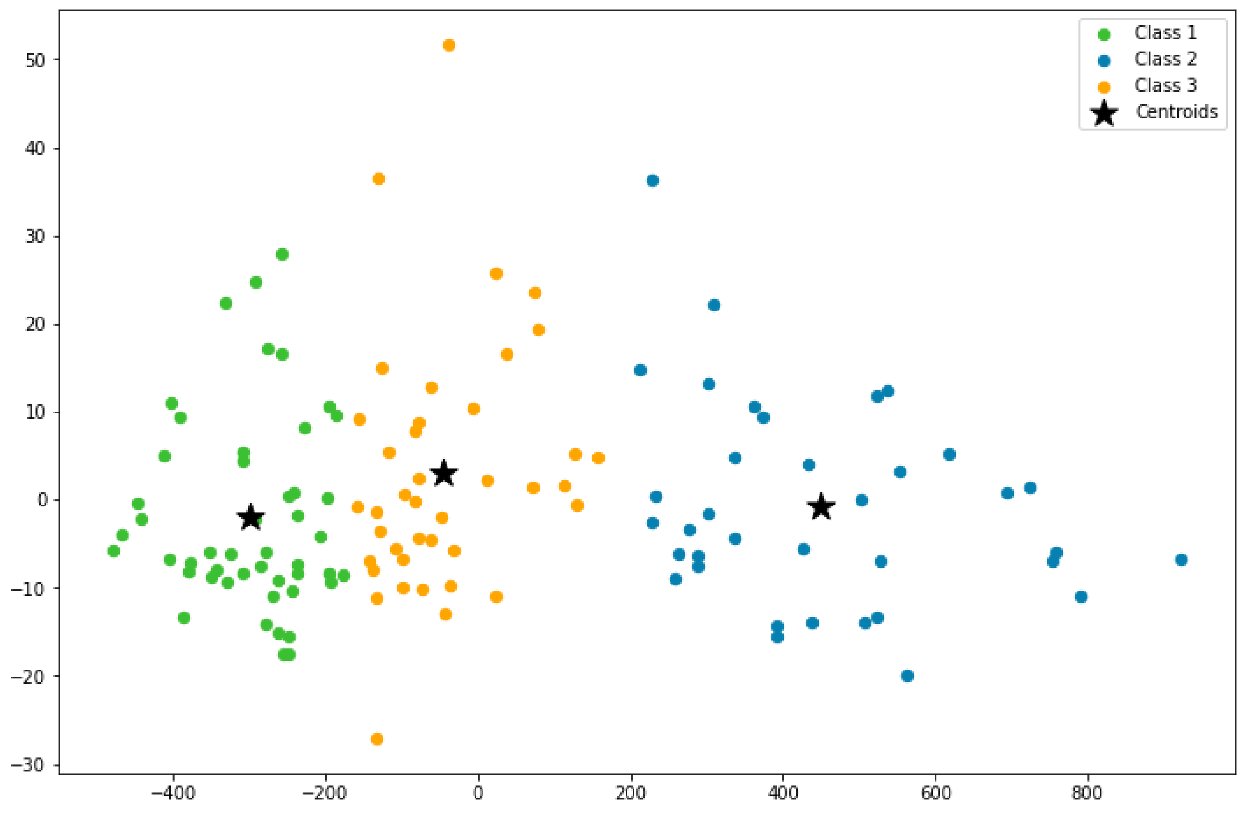
Let's take the proline-vs-alcohol plot as an example, we see that k=3 clusters can be separated, however clearly separate clusters can't be seen from these plots.





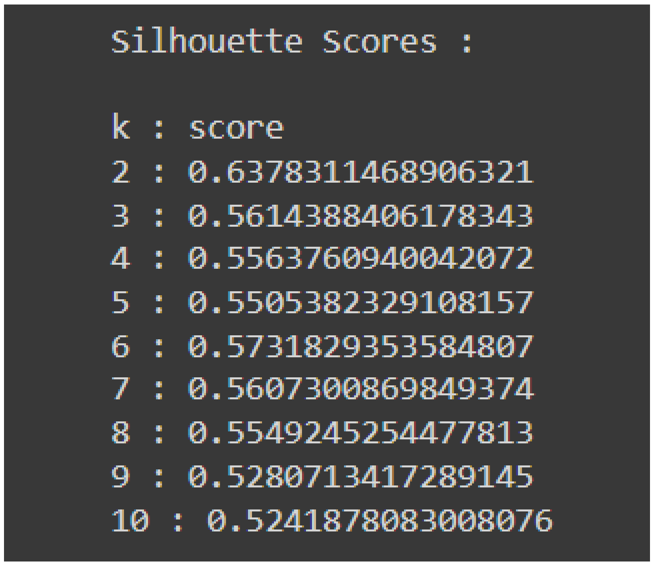
2. K means (sklearn) :

- Applied k means algorithm to the dataset with k=3, and found the following cluster separation.



3. Silhouette Scores on varying k:

- Varied parameter k for k means and found their Silhouette Scores to see which value gives the optimal number of clusters.
- We find that k=2 has maximum Silhouette Score, and k=3 follows next.

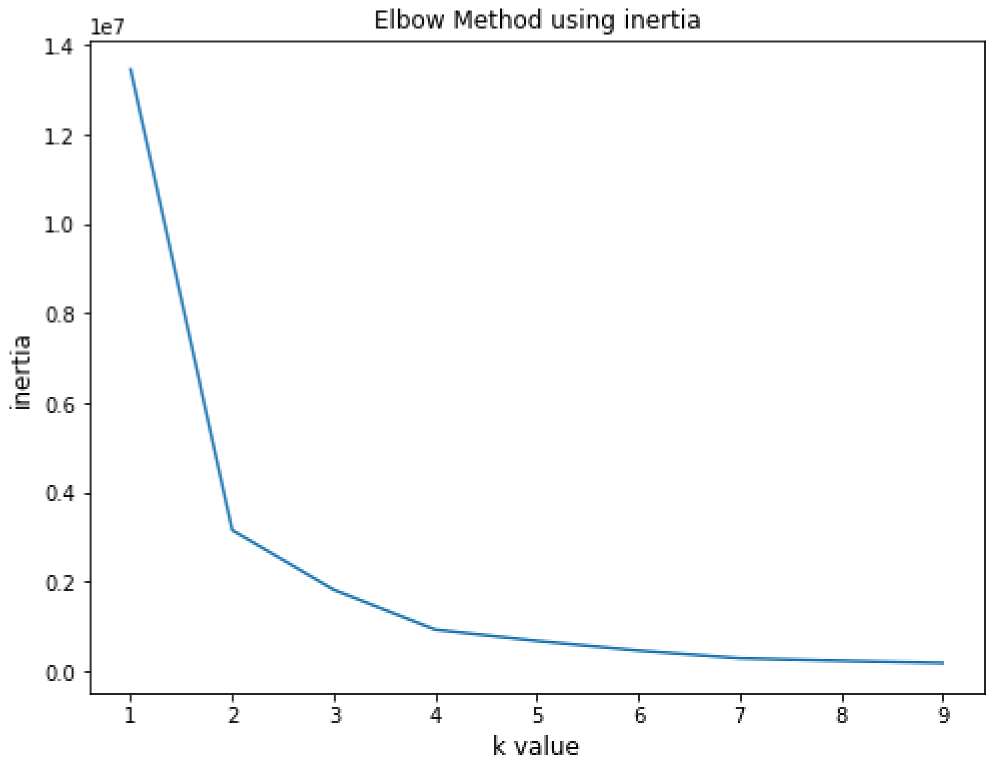


```
Silhouette Scores :

k : score
2 : 0.6378311468906321
3 : 0.5614388406178343
4 : 0.5563760940042072
5 : 0.5505382329108157
6 : 0.5731829353584807
7 : 0.5607300869849374
8 : 0.5549245254477813
9 : 0.5280713417289145
10 : 0.5241878083008076
```

4. Elbow Method :

- In the elbow method, we vary *k* and find the inertia (sum of squared distance of samples to their closest cluster center) value for each *k*.
- Then we see at which value of *k*, the inertia starts to linearise or change in slope is pretty much high and choose that *k* value as optimal.



We see that upto k=3, the slope was steep. However from k>3, the slope gets steady and lesser in magnitude. (less decrease in inertia)
So we choose k=3 as the optimal number of clusters.

Question 2 :

A. K Means from scratch :

B. In part A, K Means class should hold :

- Be a class which will be able to store the cluster centers
  - Made class and stored centroids as self.centroids attribute to the class.

- Take a value of k from users to give k clusters.
  - Made k a parameter in __init__ method and stored it as an attribute to class, so k is defined for each instance of the model.

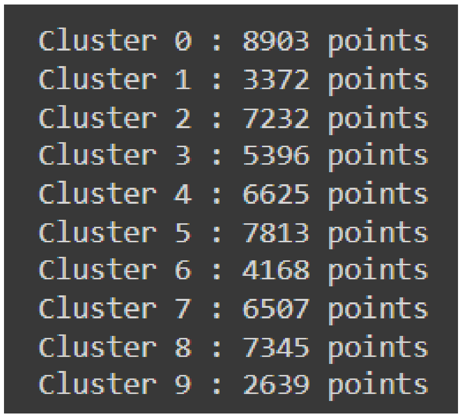
- Be able to take initial cluster center points from the user as its initialization.
  - Made a parameter init_mode which can take two values : 'random', 'manual'.
  - Set to random, it initialises the centroids randomly.
  - Set to manual, it takes another parameter init_cluster_centers of shape (k, n_features) to initialize the centroids manually.

- Stop iterating when it converges (cluster centers are not changing anymore) or, maximum iteration (given as max_iter by user) is reached.
  - Added break condition for termination of loop with max_iter steps.
  - We check if the sum of the distances of previous centroids and current centroids is 0 or not, if 0 (no changes in centroids), we terminate the loop.

Made a model which can do all of the above mentioned things along with some other functionalities like plotting clusters.

C. Training model with random initializations :

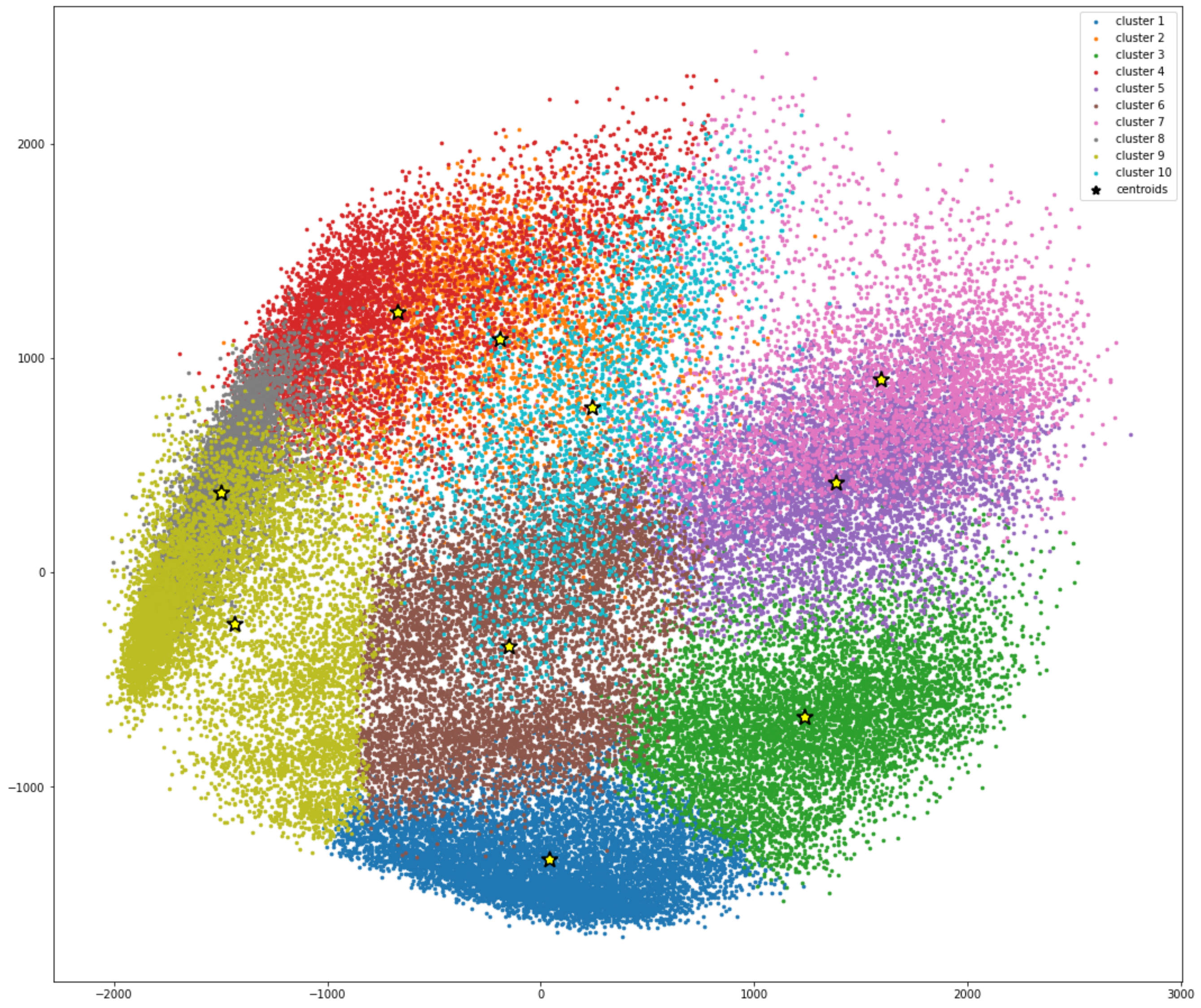
- Trained the K_Means model with random centroids with **init_mode='random'** parameter.
- Calculated number of points in each cluster.



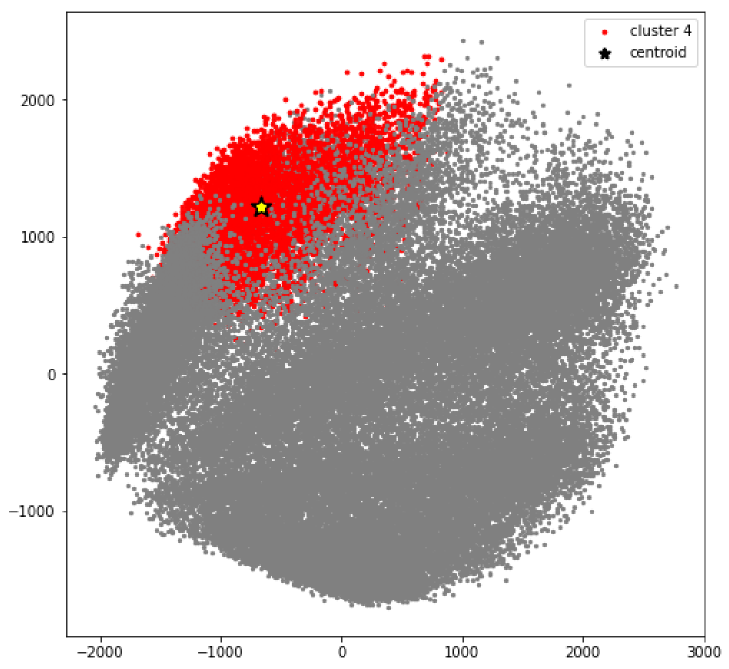
```
Cluster 0 : 8903 points
Cluster 1 : 3372 points
Cluster 2 : 7232 points
Cluster 3 : 5396 points
Cluster 4 : 6625 points
Cluster 5 : 7813 points
Cluster 6 : 4168 points
Cluster 7 : 6507 points
Cluster 8 : 7345 points
Cluster 9 : 2639 points
```

D. Visualising the clusters :

- Since there are 784 features, we cannot visualise the clusters in 2 dimensions.
- Also we cannot take any random 2 features to visualise as it wouldn't show correct separation of clusters because it won't take other dimensions into account.
- So we apply PCA and reduce the dimensions of features to 2.
- Below are the results. Even if it seems that clusters aren't quite separable, they are in fact. It happens as due to PCA, there is some data loss and PCA.



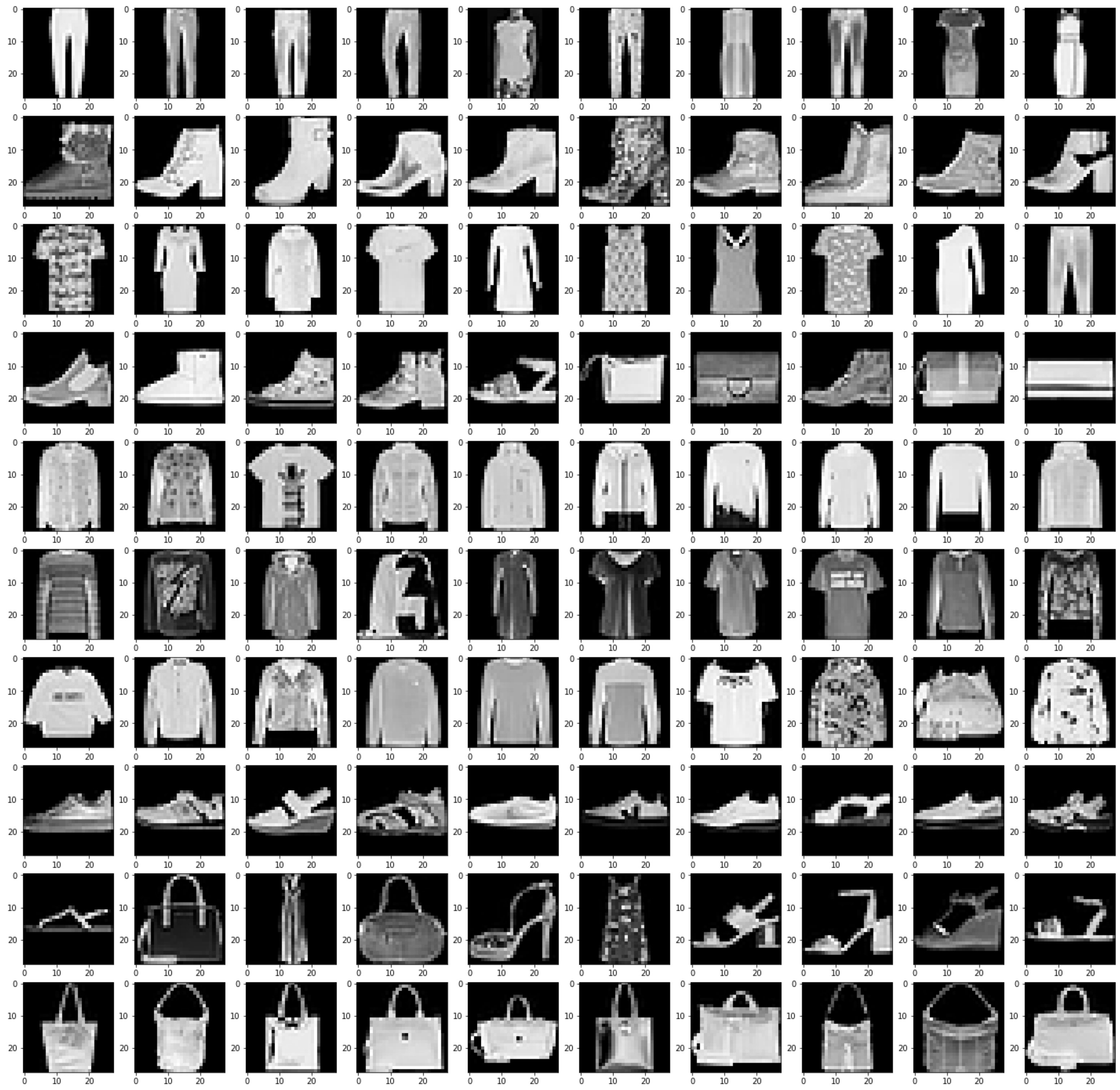
It seems as if the data is forcefully forged into 2 dimensions. (due to PCA)



Also made such 10 plots to visualise individual clusters.

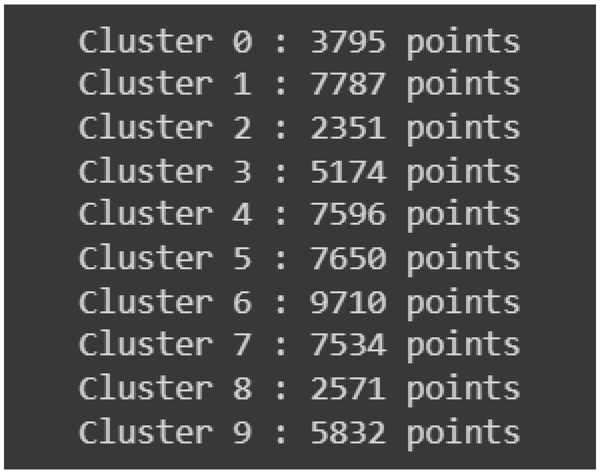
E. Images corresponding to each cluster :

Converted the arrays to images after reshaping them into matrices of shape 28*28. Here are the images corresponding to each class/clusters :



F. Manual centroid initialisation :

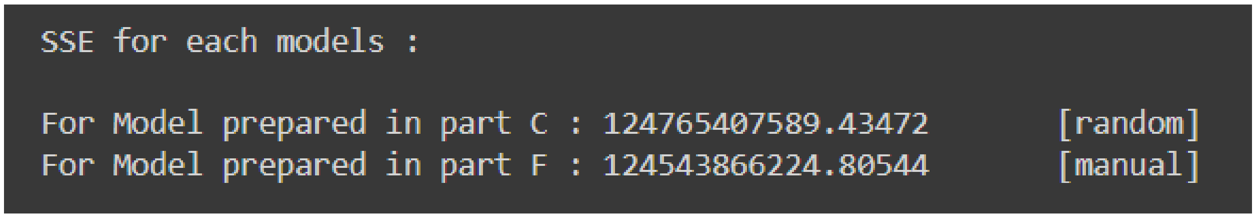
Took 10 images (784 features for each image) from each class and initialised the centroids of the K_Means class manually using init_mode='manual' parameter.
Found following results :



```
Cluster 0 : 3795 points
Cluster 1 : 7787 points
Cluster 2 : 2351 points
Cluster 3 : 5174 points
Cluster 4 : 7596 points
Cluster 5 : 7650 points
Cluster 6 : 9710 points
Cluster 7 : 7534 points
Cluster 8 : 2571 points
Cluster 9 : 5832 points
```

H. Comparing SSE for both models :

We calculated the sum of the squared distance between centroid and each member of the cluster, and found following results :



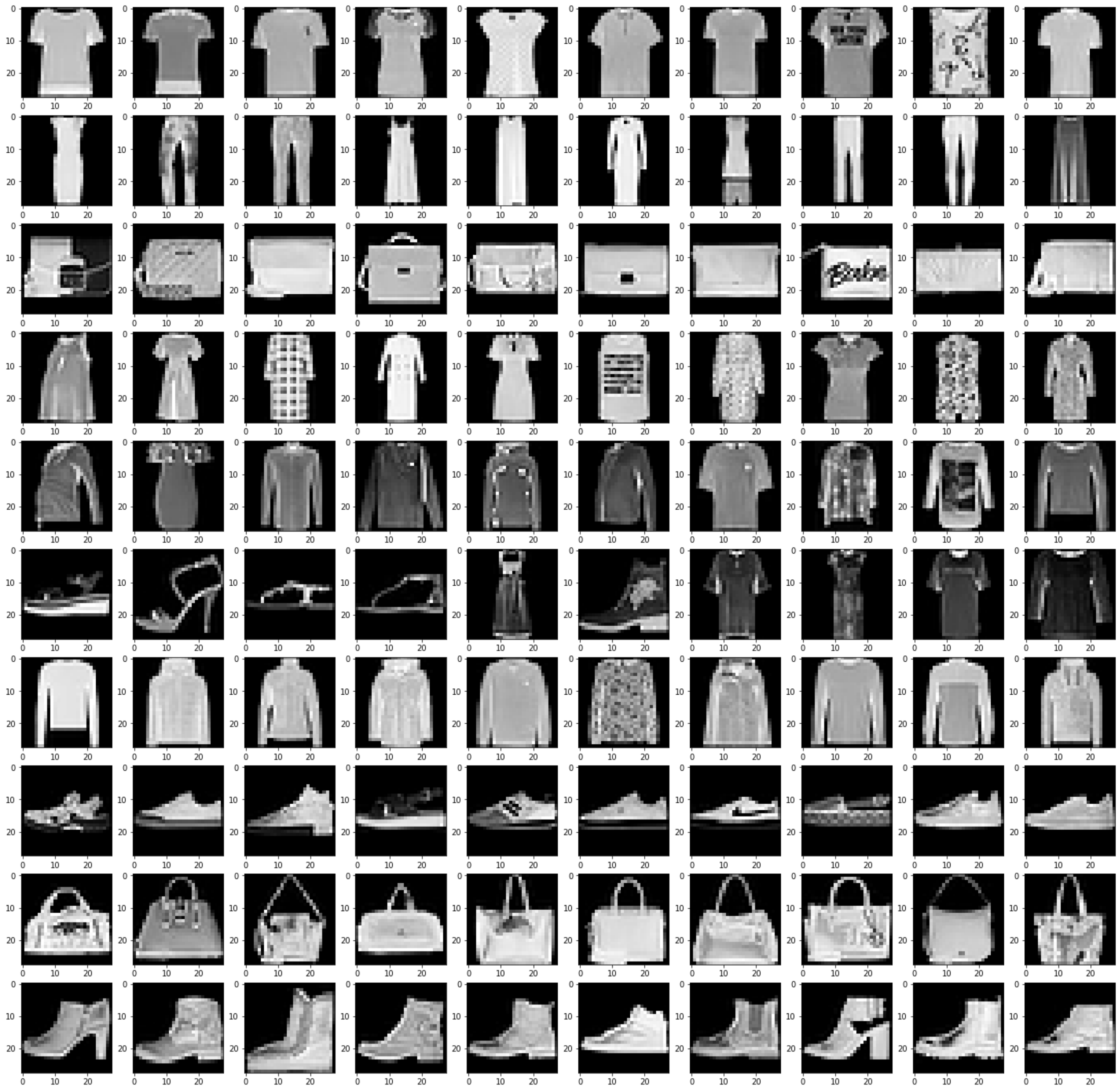
```
SSE for each models :

For Model prepared in part C : 124765407589.43472       [random]
For Model prepared in part F : 124543866224.80544       [manual]
```

SSE is lesser in Model of subpart F, hence it is better than the other model.

We realise that manually setting the initial centroids, from each of the classes will help minimise SSE and hence result in better clustering.

G. Images corresponding to each cluster [manual initialisation] :



Question 3 :

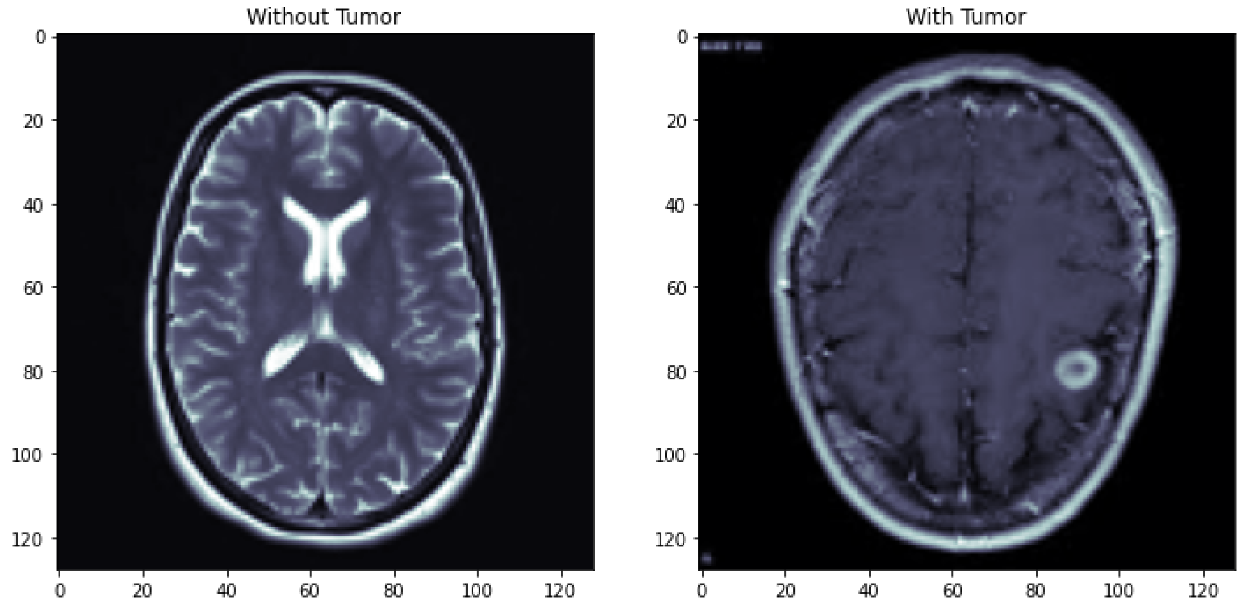
A. Preprocessing :

- Used PIL library to convert images to arrays of shape (28,28) by reading data, converting to grayscale and then compressing.
- Scaled the data as asked in the question.

B. Dimension Reduction :

- Since the images were in high resolution, we converted them to 28x28 and hence we got 784 features after using PCA.

C. Visualising Classes :

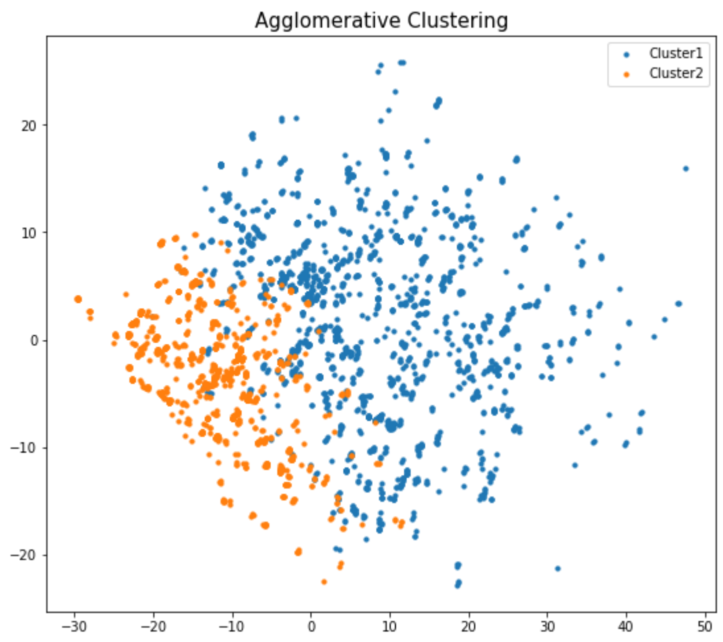


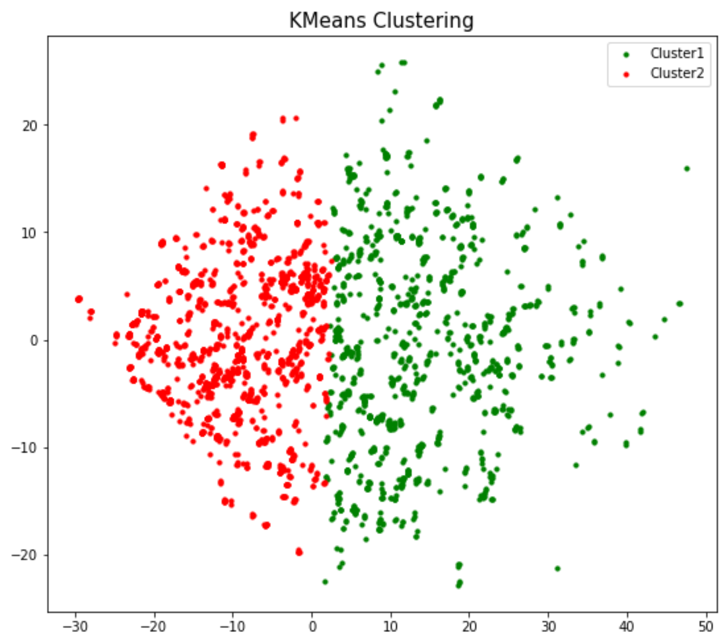
D. Agglomerative hierarchical clustering :

Applied Agglomerative clustering using sklearn.

E. Applying KMeans and comparing with the former:

Following were the results if we compare both the models.





From plots, it seems that KMeans is better than Agglomerative Clustering, however it is not the case (we plot after performing PCA which reduces dimensions, which mixes up clusters a bit).

By comparing the accuracies, we see that Agglomerative Clustering performs better.

```
Accuracies :

Agglomerative Clustering : 0.7323
KMeans Clustering : 0.6643
```

End of the Report !