# Pattern Recognition and Machine Learning

**(Winter 2022)**

**Assignment 9**

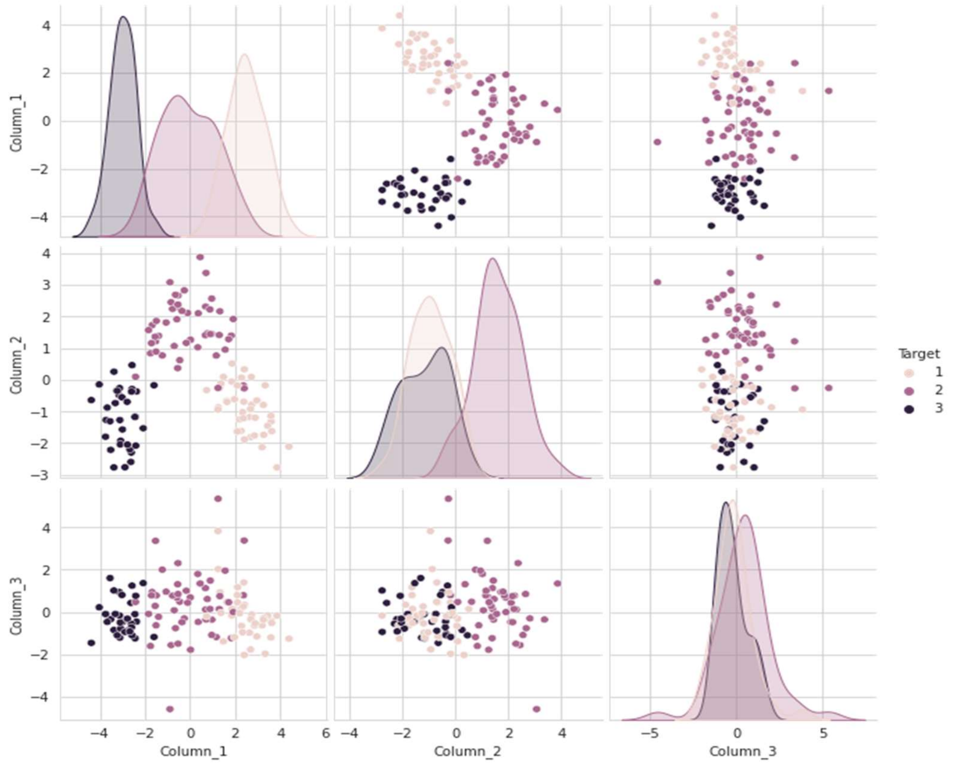
**Report**

## <u>Question - 1:</u>

First, I did the pre-processing of the dataset. I dropped any Nan/null values present in the dataset. Then, with the help of Standard Scaler, I scaled the dataset. Then, for further task, I separated the target column from the dataset.

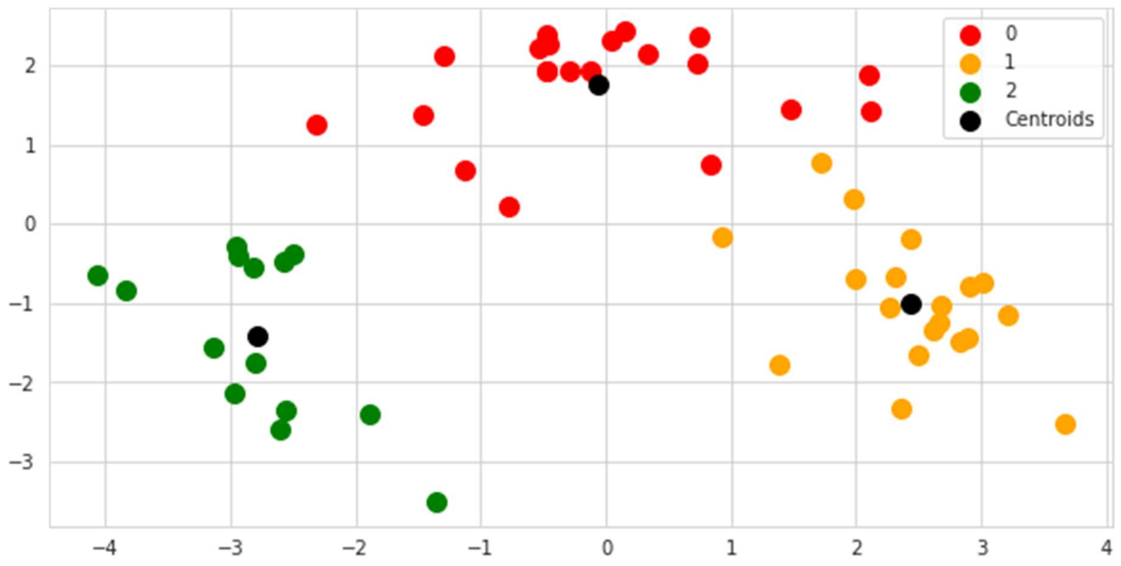
**<u>Task - 1:</u> Use any dimension reduction technique of your choice, visualize the data and by looking at the plot tell which value of k will be best suited for k-means clustering and why? (no need to use any method to find optimal k).**

For this task, I used the PCA dimension reduction technique to reduce the dimensions of the dataset. Then, I visualized the dataset using the scatter plots. It is visible form the plots that 3 clusters can be made and hence, k = 3 will be best suited for the k-means clustering.



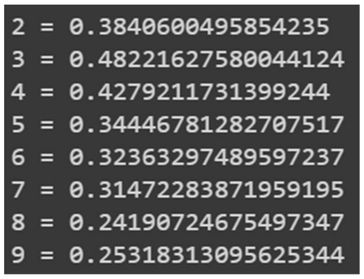
**<u>Task - 2:</u> Build a k-means clustering algorithm( can use sklearn library) and implement using the value of k what you have chosen above. Visualize part b by showing the clusters along with the centroids.**

For this task, I imported the KMeans model from the sklearn library and instantiated. Then, I passed the parameters as  n_clusters = 3. Then, I made the scatter plots of the clusters and also plotted the centroids. The plot that I got is:



**<u>Task - 3:</u> Use different values of k and find the Silhouette Score and then tell which value of k will be optimal and why?**

For this task, I iterated over a range of values and trained the KMean model and printed the silhouette_score. The silhouette_score I got as:

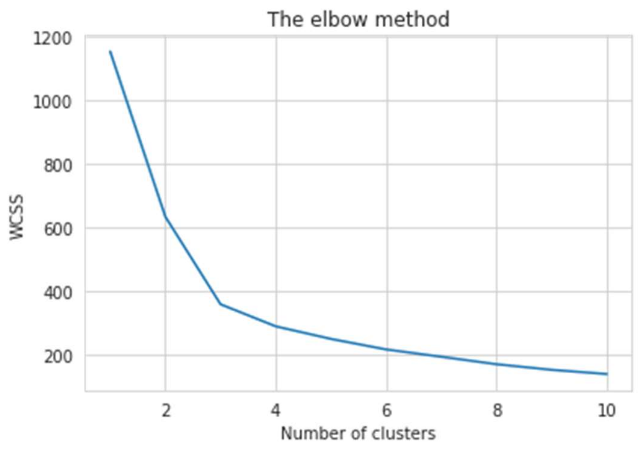


```
2 = 0.3840600495854235
3 = 0.48221627580044124
4 = 0.4279211731399244
5 = 0.34446781282707517
6 = 0.32363297489597237
7 = 0.31472283871959195
8 = 0.24190724675497347
9 = 0.25318313095625344
```

This clearly specifies that silhouette_score = 3 gives the best silhouette_score.

**<u>Task - 4:</u> There are few methods to find the optimal k value for k-means algorithm like the Elbow Method . Use the above method to find the optimal value of k.**

For this task, I plotted the curve to visualize the within cluster sum of squares. The plot that I got is:



From this graph/plot, it is observed that for 3 clusters it is giving the best results.

## Question - 2:

**Task - 1: Implement a k-means clustering algorithm from scratch.**

**Task - 2: Make sure that it should:**

**i) Be a class which will be able to store the cluster centers.**

**ii) Take a value of k from users to give k clusters.**

**iii) Be able to take initial cluster center points from the user as its initialization.**

**iv) Stop iterating when it converges (cluster centers are not changing anymore) or, a maximum iteration (given as max_iter by user) is reached.**

For the task 1 and task 2, I implemented the K Means Clustering Algorithm from scratch. For this, I made a class KMeansAlgorithm() and initialized K, max_iterations, centroids as empty list and clusters as list of lists. Then, I wrote the code for fit(), getClusterLabels(), createClusters(), closestCentroid(), getNewCentroids() and isConverged().

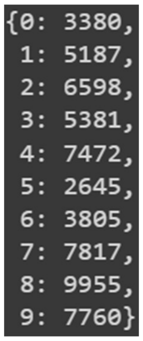
The fit() method has the flexibility of having the initializations as randomly from the entire dataset or from each class. Then, after initializations, it iterated over the user input max iterations or while it is converged (i.e., the distance between the current centroid and the new centroid is equal) and hence, returns the cluster labels.

The getClusterLabels method helps to return the labels of the clusters.

The createClusters method helps to return the clusters and getNewCentroids to get new centroids and the isConverged method to check the convergence condition.

<u>Task - 3:</u> **Train the k-means model on f-MNIST data with k = 10 and 10 random 784-dimensional points (in input range) as initializations. Report the number of points in each cluster.**

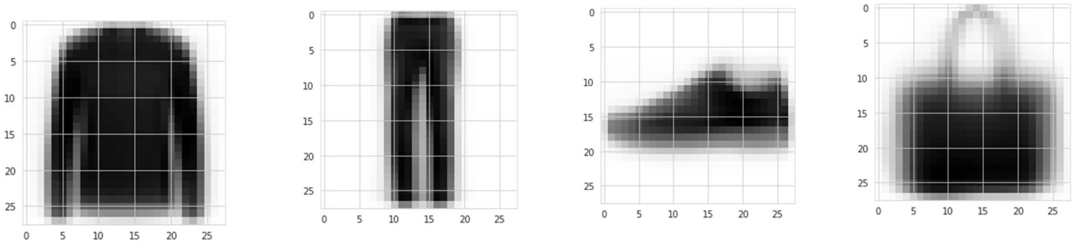
For this task, I took the input of the given dataset. Then, I passed the given dataset to the implemented KMeansAlgorithm() and took 10 random points as initializations. The number of points in each cluster are:



```
{0: 3380,
 1: 5187,
 2: 6598,
 3: 5381,
 4: 7472,
 5: 2645,
 6: 3805,
 7: 7817,
 8: 9955,
 9: 7760}
```

<u>Task - 4:</u> **Visualize the cluster centers of each cluster as 2-d images of all clusters.**

Then, I visualized the cluster centres of each cluster as 2-d images of all clusters. I got the images as:

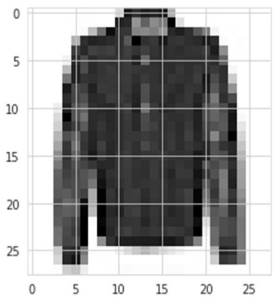


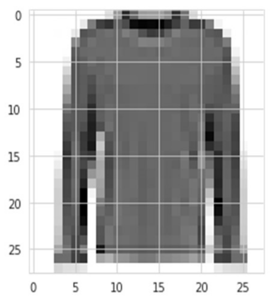
Similary, I got the image for each cluster centre in my colab file.

**<u>Task - 5:</u> Visualize 10 images corresponding to each cluster.**

For this task, I plotted the 10 images from each cluster that we got in previous task:

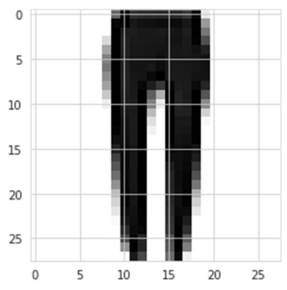
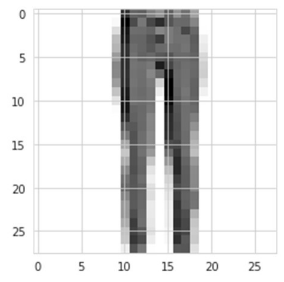
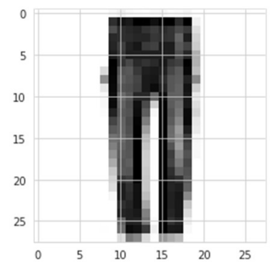
For example, for Cluster 0:





And other 7 images in this cluster.

For cluster 1:

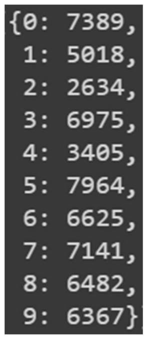




And other 7 images in this cluster.

Similarly, for each cluster I plotted the 10 images in my colab file.

**<u>Task - 6:</u> Train another k-means model with 10 images from each class as initializations , report the number of points in each cluster and visualize the cluster centers.**

For this task, I took each image from each label as initializations and trained the K Means Clustering algorithm. The number of points in each cluster are:

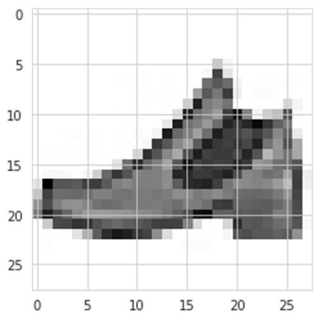
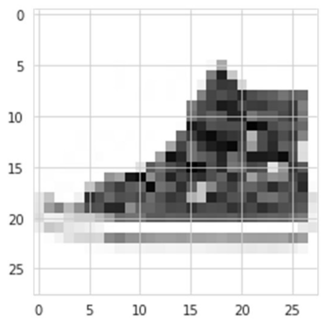
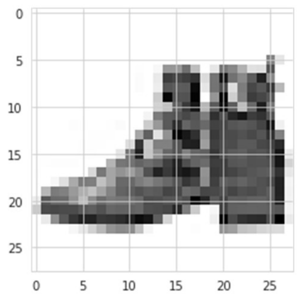


```
{0: 7389,
 1: 5018,
 2: 2634,
 3: 6975,
 4: 3405,
 5: 7964,
 6: 6625,
 7: 7141,
 8: 6482,
 9: 6367}
```

**<u>Task - 7:</u> Visualize 10 images corresponding to each cluster.**

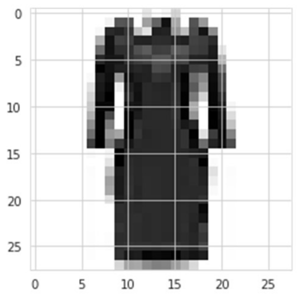
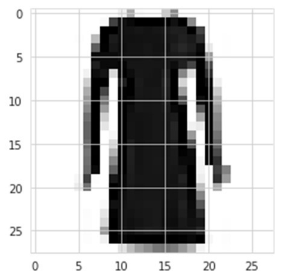
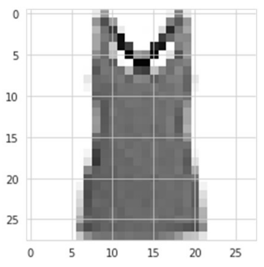
For this task, I plotted the 10 images from each cluster that we got in previous task:

For example, for Cluster 0:

And similar other 7 images.

For Cluster 1:

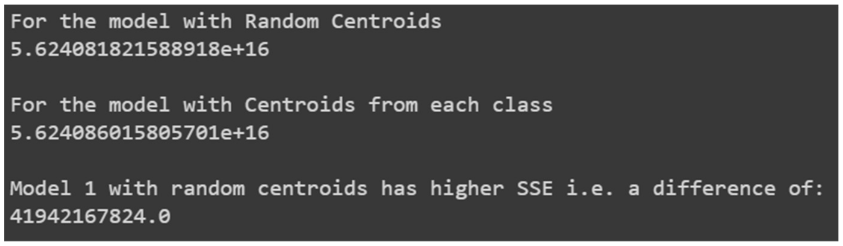




And similar other 7 images.

Similarly, for each cluster I plotted the 10 images in my colab file.

**<u>Task - 8:</u> Evaluate Clusters of part c and part f with Sum of Squared Error (SSE) method. Report the scores and comment on which case is a better clustering.**

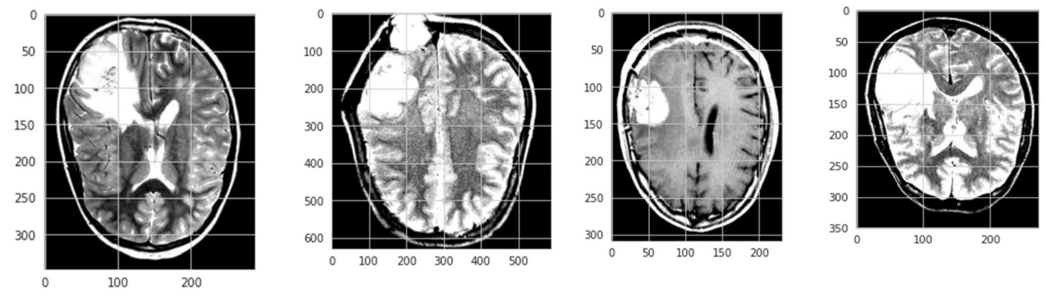
For this task, I wrote a function to calculate the SSE. Then, I passed this function to both the part c and part f and calculated the SSE scores.



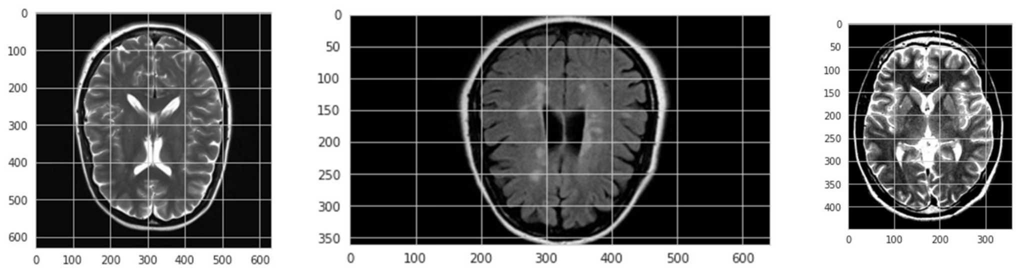
```
For the model with Random Centroids
5.624081821588918e+16

For the model with Centroids from each class
5.624086015805701e+16

Model 1 with random centroids has higher SSE i.e. a difference of:
41942167824.0
```

## Question - 3:

For the visualization of the given dataset. I plotted some images from both the folders. For brain tumor present:



For brain tumor absent:



**Task-1: Check out the dataset & normalize the data so that the scale of each variable will be the same.**

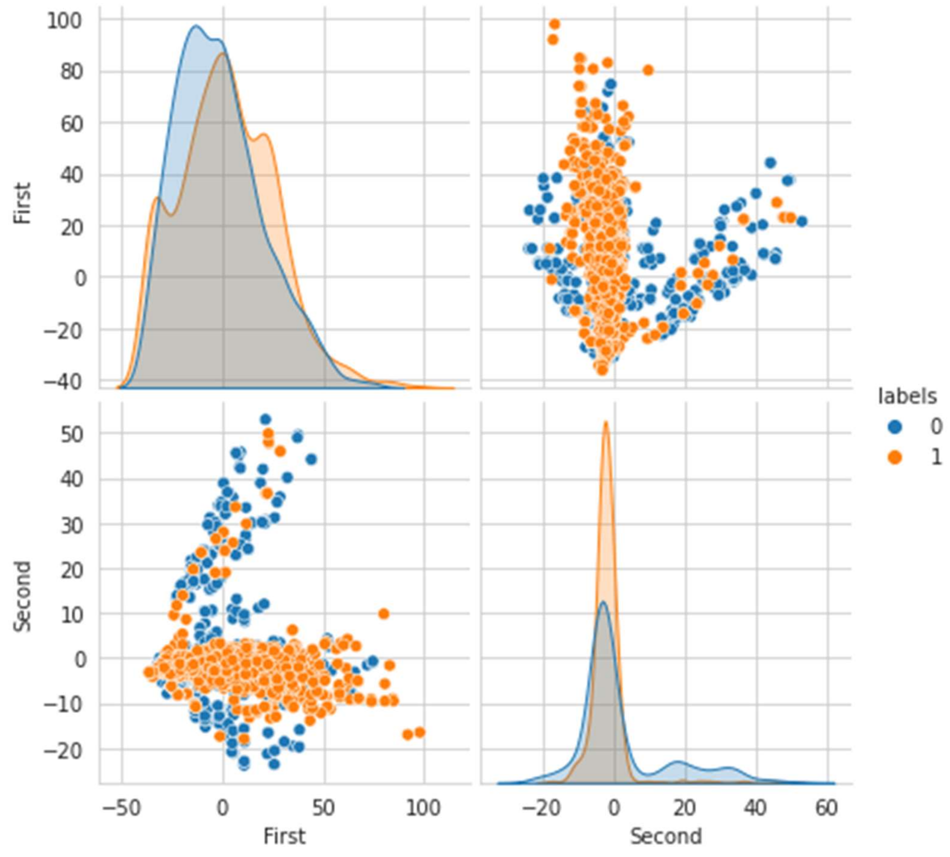
For this task, I first made the dataset using the given images using the cv2 library to read then flatten the Gray image. I did this for all the images present in both the yes and no folders and hence, prepared the dataset.

**Task-2: Use any dimension reduction technique and visualize the dataset & find out the number of communities available.**

For this task, I used the PCA dimension reduction technique to reduce the dimensions. I transformed the dataset into 2 principal components. Then, I made the data frame out of the two principal components.

**<u>Task-3</u>: Visualize the communities from part A.**

For this task, I visualized the communities from the dataset. I got the plot as:

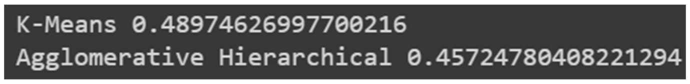


**<u>Task-4</u>: Apply Agglomerative hierarchical clustering (using sklearn).**

For this task, I imported the AgglomerativeClustering form sklearn.cluster library and trained the model with 2 as number of clusters.

**<u>Task-5</u>: Apply K-means (sklearn) and make a comparison between these two approaches & justify your results.**

For this task, I applied K-means to the given dataset and trained the model. Then, for the comparison of the two approaches I calculated the silhouette_score to compare the approaches. I got the result as:



```
K-Means 0.48974626997700216
Agglomerative Hierarchical 0.45724780408221294
```

Hierarchical clustering can't handle big data well but K Means clustering can. This is because the time complexity of K Means is linear i.e. O(n) while that of hierarchical clustering is quadratic i.e. O(n2). Hence, K-Means performed well.