CSL2050 PATTERN RECOGNITION AND MACHINE LEARNING

LAB REPORT-06

NAME:	SAMEER SHARMA
ROLL NUMBER:	B21CS066
LAB TITLE:	Unsupervised Learning

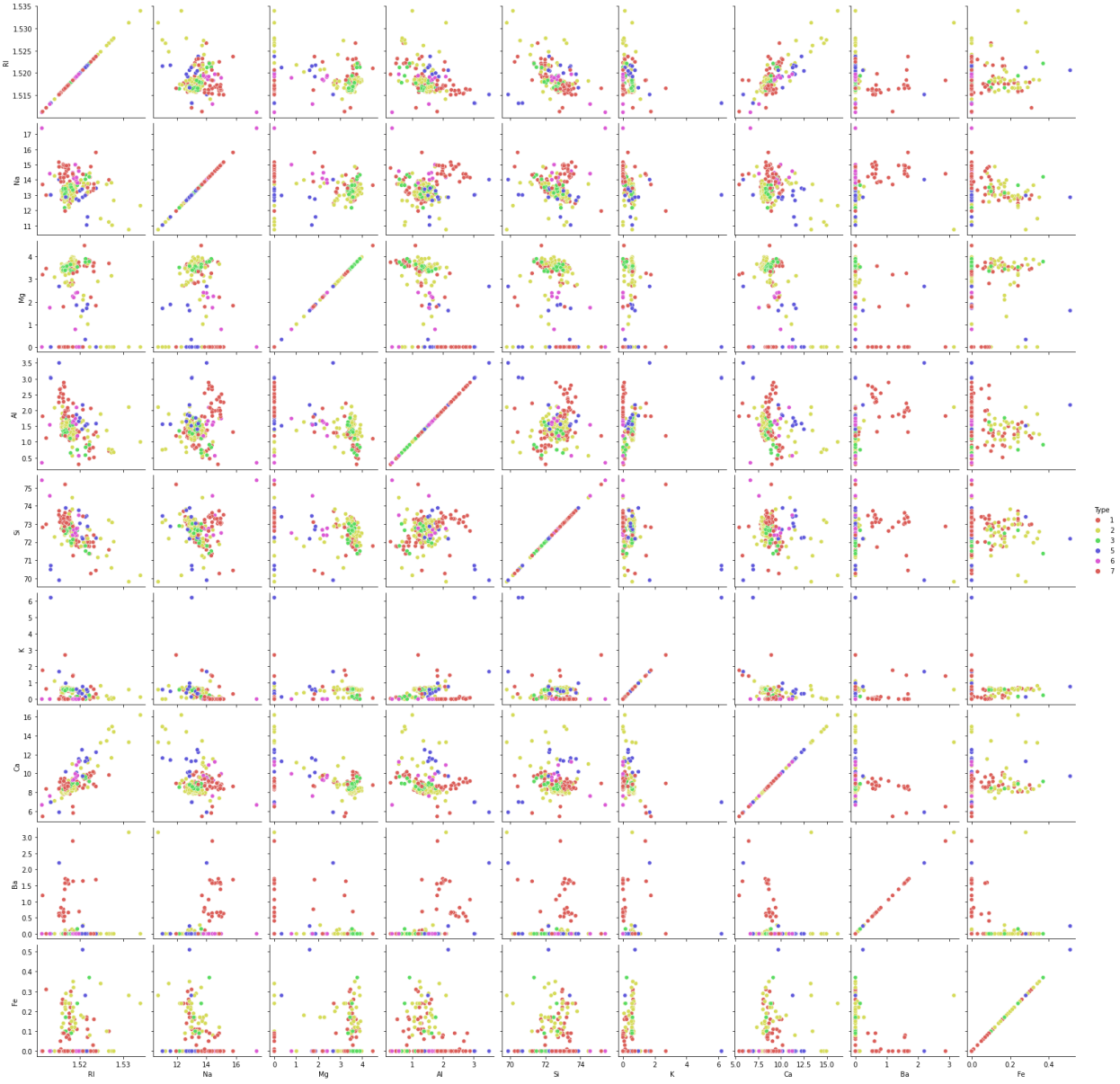
Q1)

Part 1)

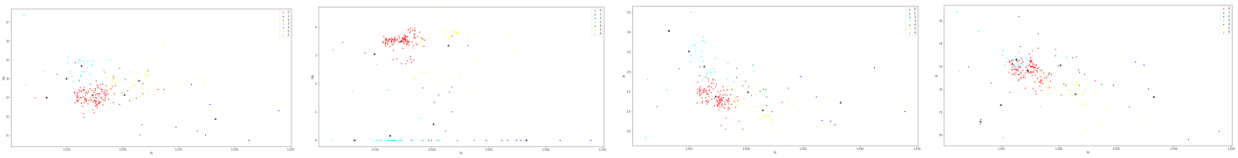
In this part, I have loaded the glass dataset. Since, there are no null values and all features are stored in their desired data types, there is no need of preprocess this data

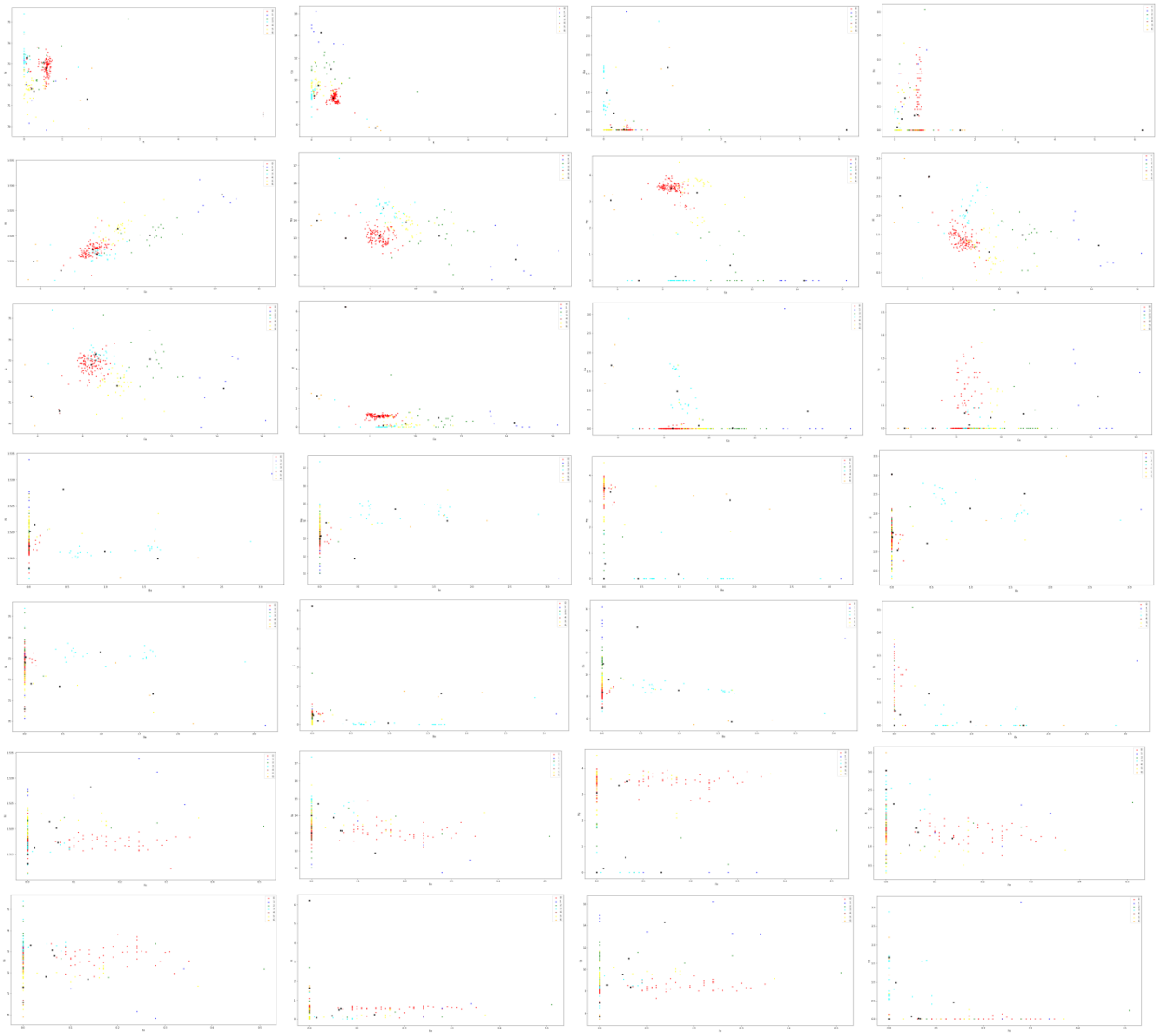
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      214 non-null    int64
 1   RI      214 non-null    float64
 2   Na      214 non-null    float64
 3   Mg      214 non-null    float64
 4   Al      214 non-null    float64
 5   Si      214 non-null    float64
 6   K       214 non-null    float64
 7   Ca      214 non-null    float64
 8   Ba      214 non-null    float64
 9   Fe      214 non-null    float64
 10  Type    214 non-null    int64
dtypes: float64(9), int64(2)
```

The visualization of the dataset is as follows. In the data visualization, I have plotted every possible feature to each possible features using the pairplot function from seaborn. Please ignore the graphs on the diagonal (Since they are plots of same feature on both x and y axis)



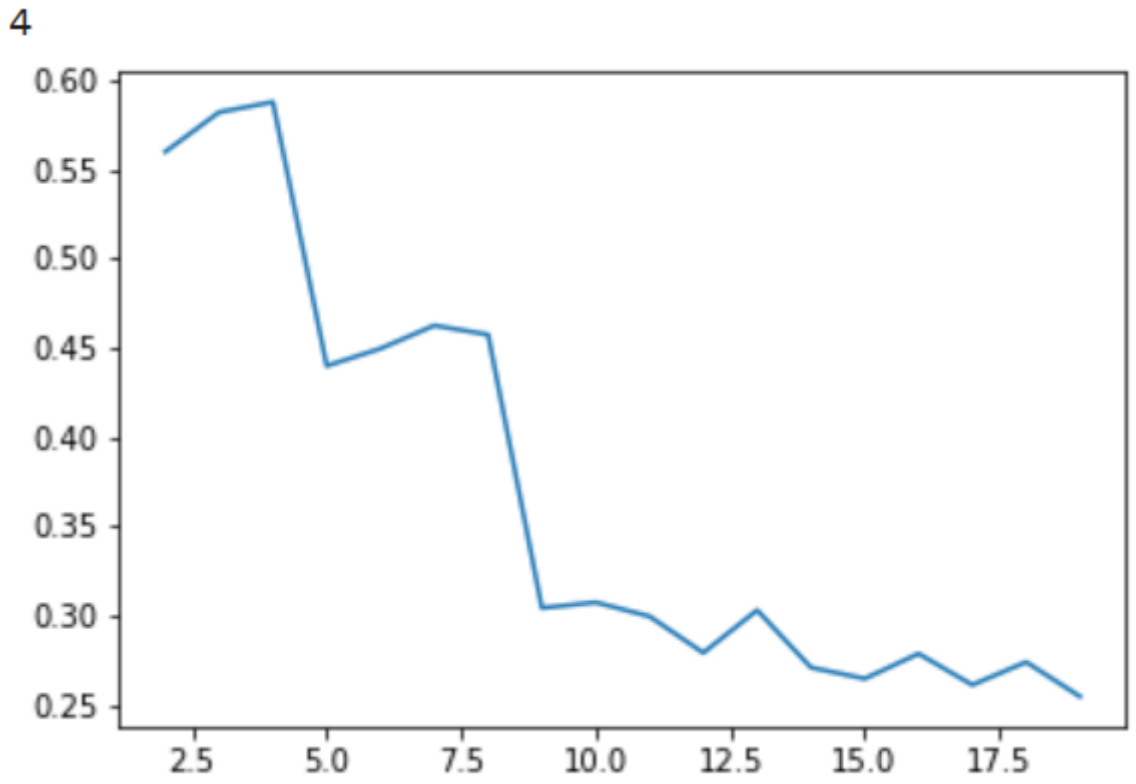
Now, from the sklearn library, I imported K means clustering classifier and initialized it with no of clusters n=7. Then, trained it on the given dataset. Visualization of datapoints with their predicted cluster and cluster points is as follows.





Part 2)

In this part, I have calculated Silhouette scores for different values of k. Got the maximum Silhouette score of around 0.6 at k=4 as shown in the figure below



Since we know that more Silhouette score correspond to more optimal value for k, most optimal value for k is 4. This is so because Silhouette score is

$$\text{Silhouette Score} = (b-a)/\max(a,b)$$

Where

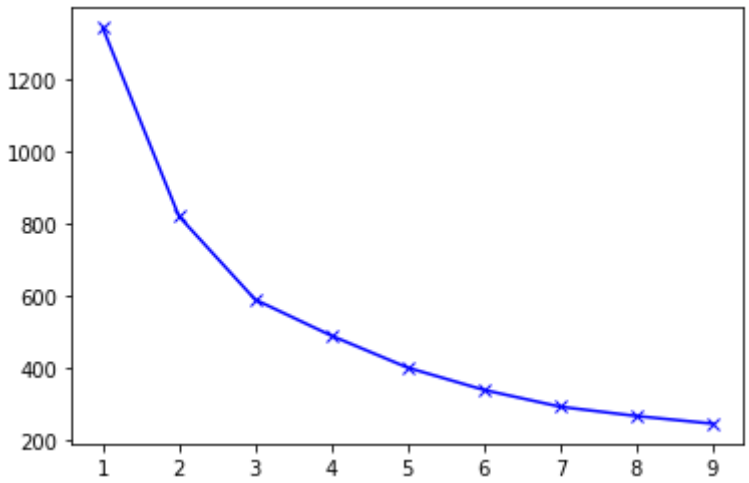
a= average intra-cluster distance i.e the average distance between each point within a cluster.

b= average inter-cluster distance i.e the average distance between all clusters.

The more the value approaches to one, the more distinguishable clusters are formed.

Part 3)

Here, I have used inertia property of K means clustering classifier to find the optimal value of k using elbow method



As we know, the optimal value of k will be that at the knee of the curve. So, optimal value for k is around 3 or 4.

Part 4)

Here are the accuracies for single knn classifier and bagging classifier consisting of 10 knn classifiers for different values of k.

```
accuracy of single knn model on k= 1 is 0.7692307692307693
accuracy of bagging classifier on k= 1 is 0.7538461538461538
accuracy of single knn model on k= 2 is 0.6615384615384615
accuracy of bagging classifier on k= 2 is 0.7230769230769231
accuracy of single knn model on k= 3 is 0.6461538461538462
accuracy of bagging classifier on k= 3 is 0.676923076923077
```

Clearly, we can see that, the accuracy of bagging classifier is slightly greater than that of single knn classifier. This is so because when we do bagging of the classifiers, their variance decrease and thus overall accuracy increases. Bagging does not affect bias of the classifier but it decreases the variance.

Q2)

Part a) and b)

Implemented a K means clustering classifier from scratch. Well commented code for the same is in the colab file.

Part c)

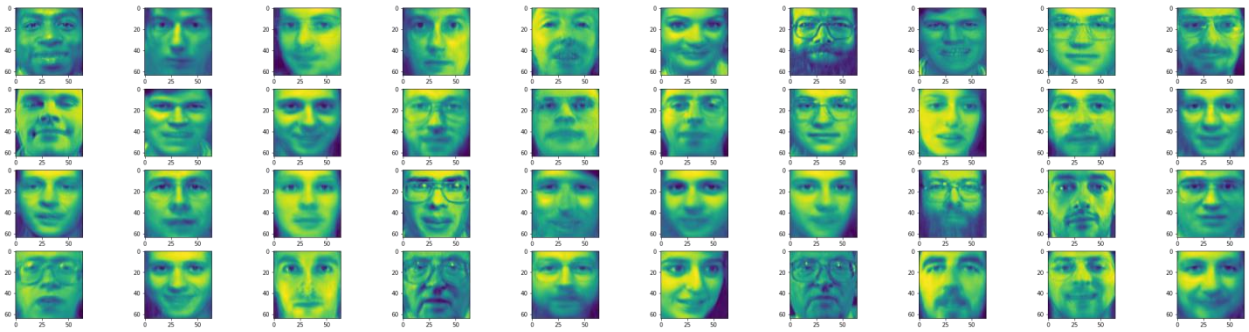
Chose 40 random points from the Olivetti dataset and used them as initial centroids for my classifier. The obtained total number of datapoints in each cluster are as follows

```
[10, 5, 14, 6, 8, 6, 5, 2, 3, 5, 2, 8, 13, 20, 8, 12, 5, 4, 12, 20, 9, 20, 19, 4, 10, 21, 23, 5, 2, 20, 10, 11, 10, 6, 22, 3, 4, 11, 6, 16]
```

Clearly all of them are near to 10 which is ideal number of data points in a cluster.

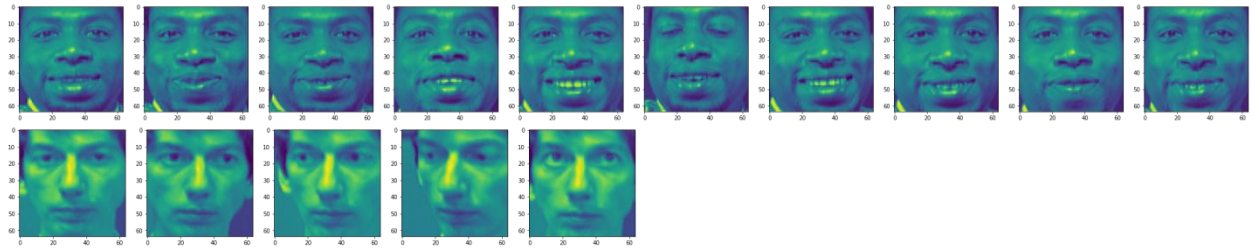
Part d)

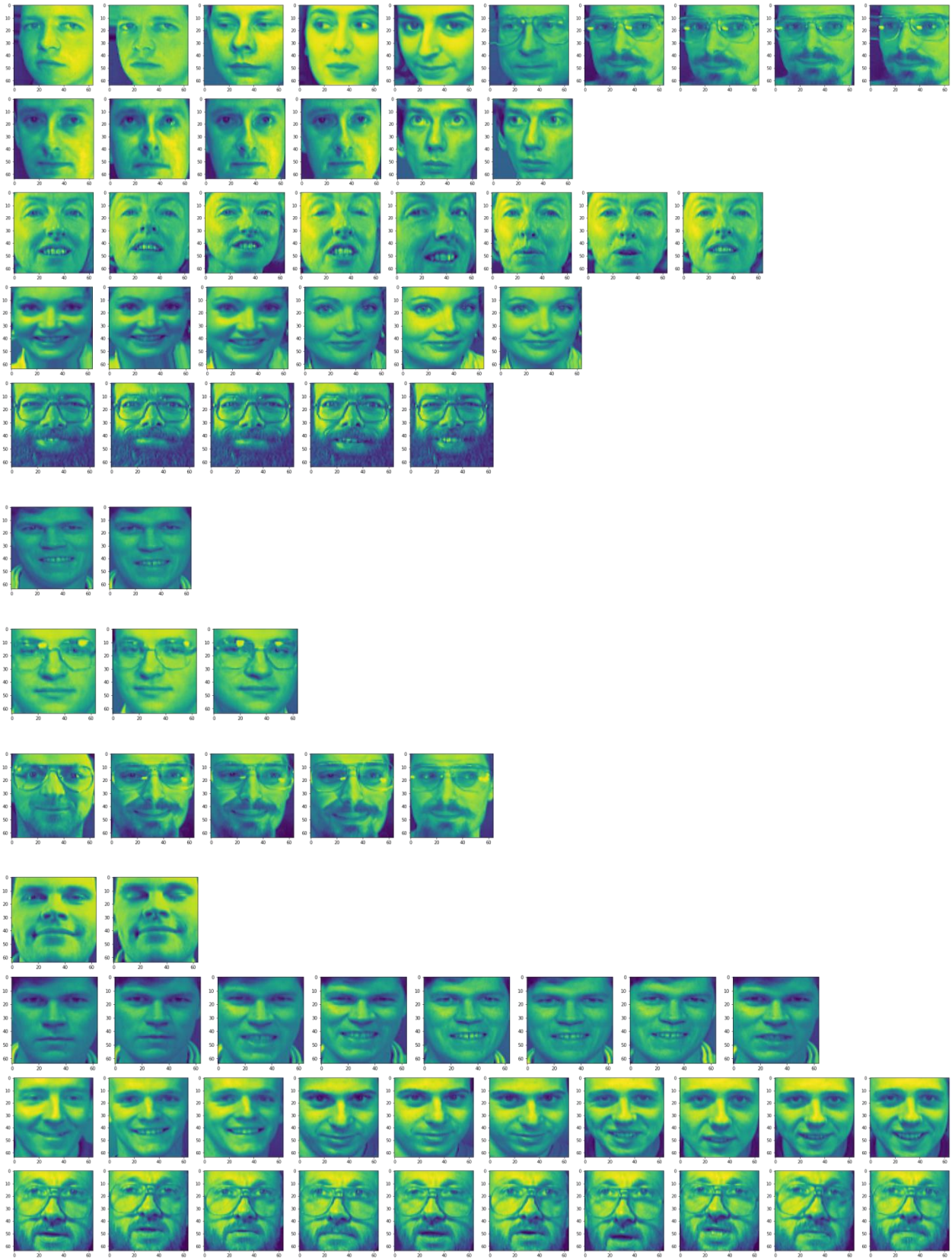
Visualization of cluster means are as follows

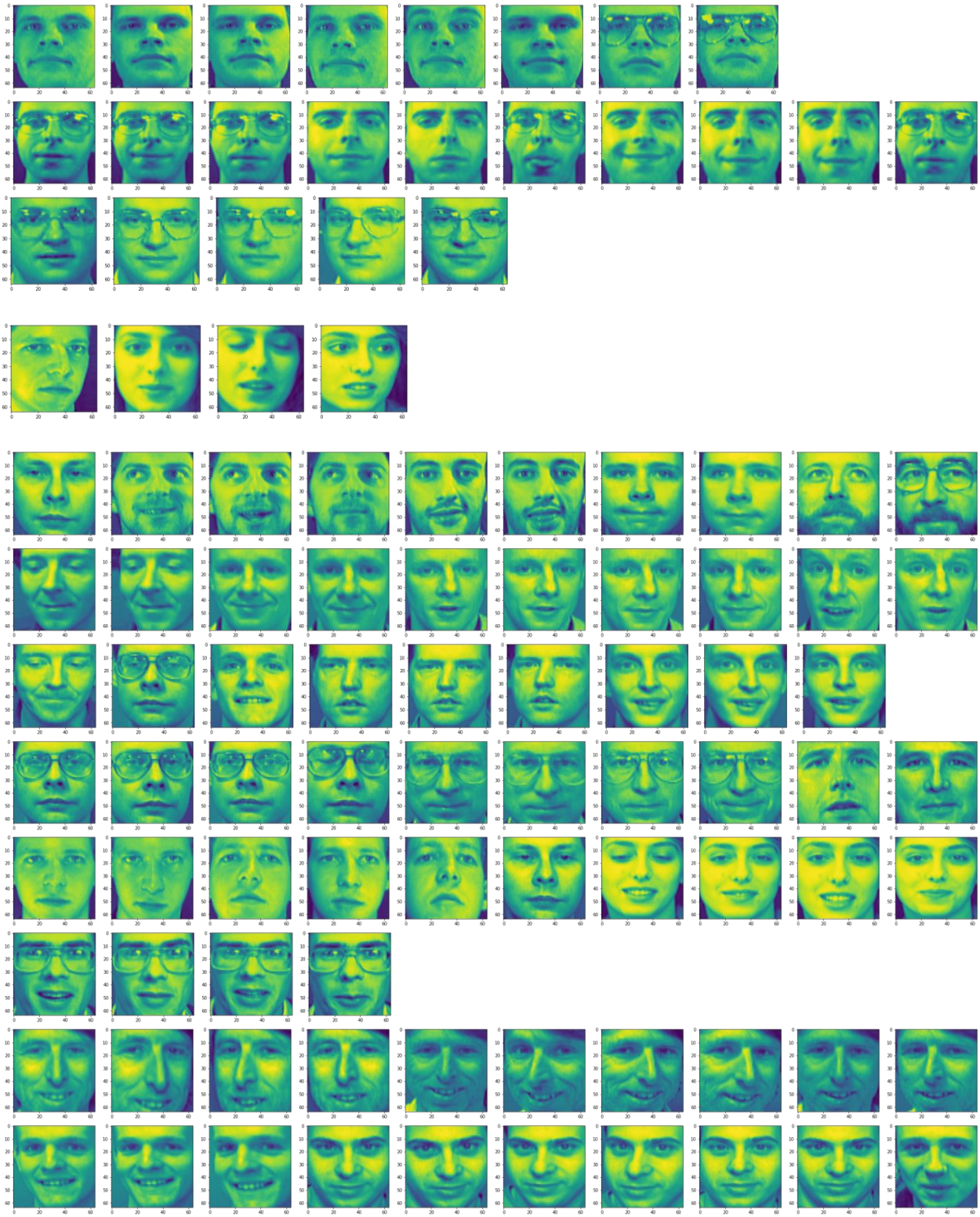


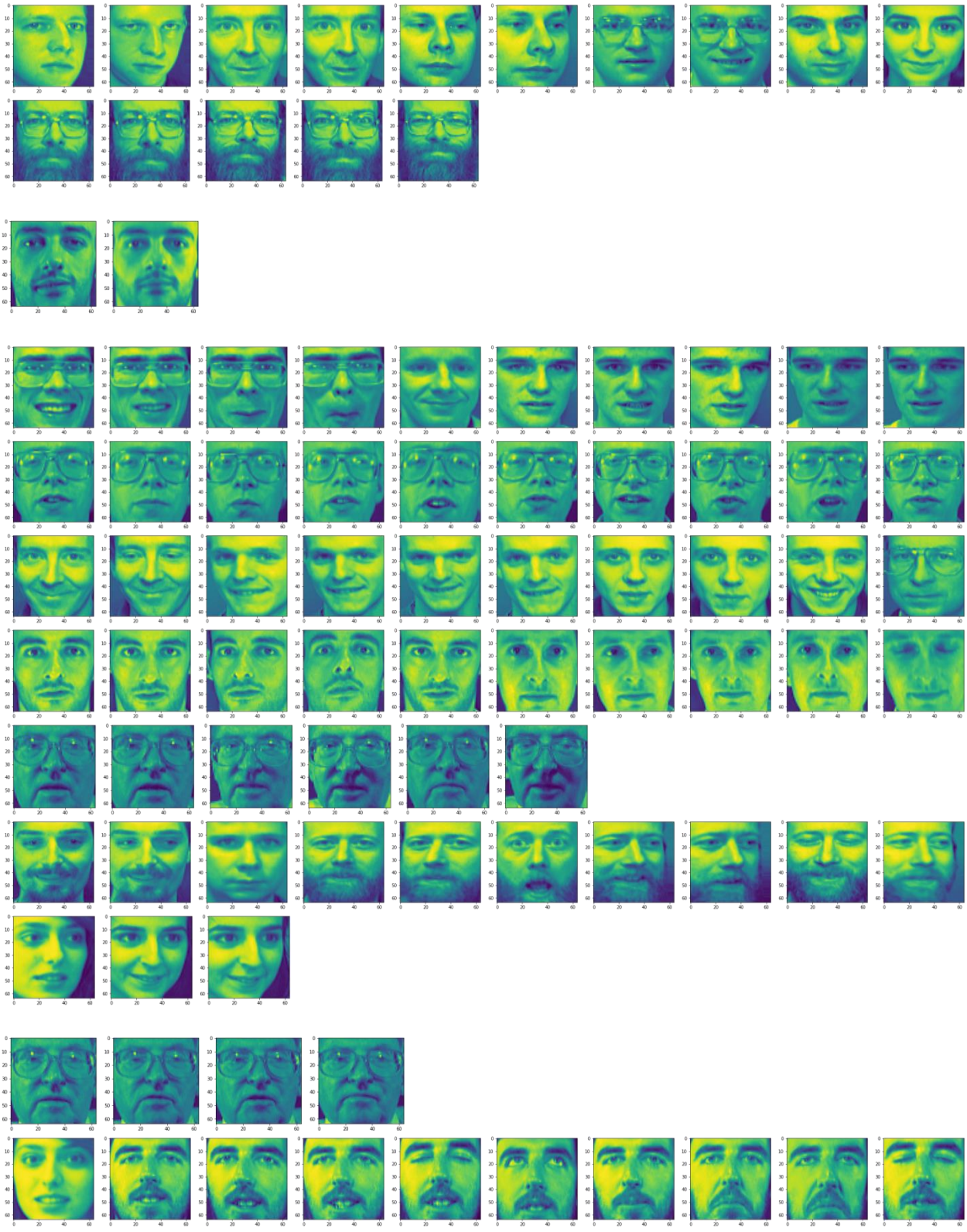
Part e)

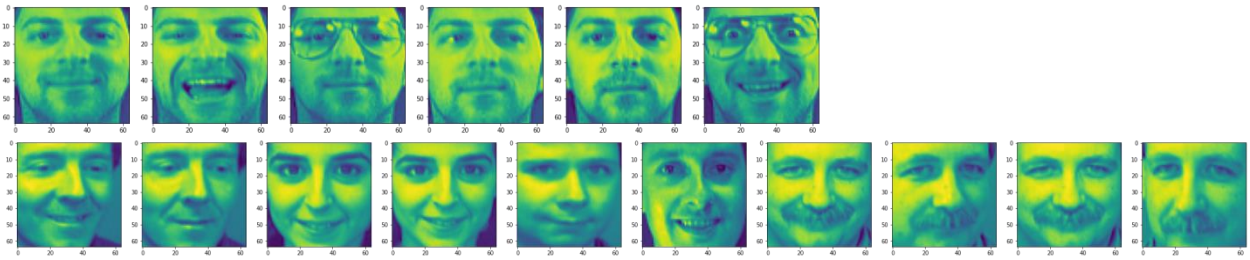
Since, in some clusters, there are less than 10 images, I have visualized all the images in those datasets. For those with more than 10 images, I have visualized first 10 images from the cluster.









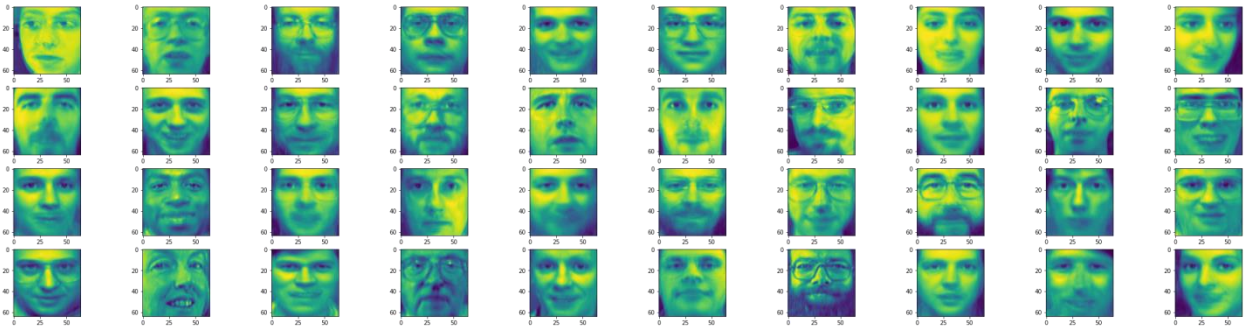


Part f)

Trained a k means classifier from sklearn library and initialized the cluster centers with one randomly chosen datapoint from each class. Number of datapoints in each cluster are as follows

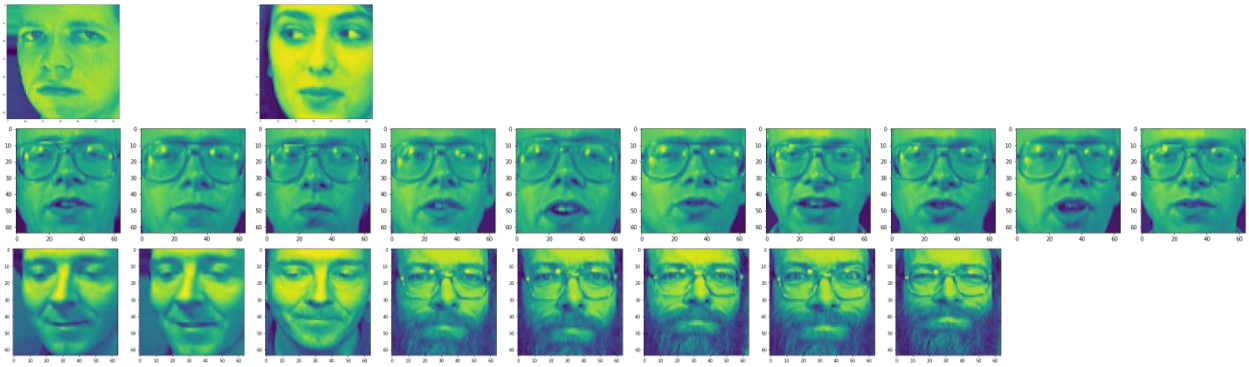
```
2 15 8 6 10 10 6 6 12 7 15 6 11 11 5 13 4 24 6 4 19 10 22 5 24 6 12 9 9 18 8 1 10 10 6 10 5 14 10 11
```

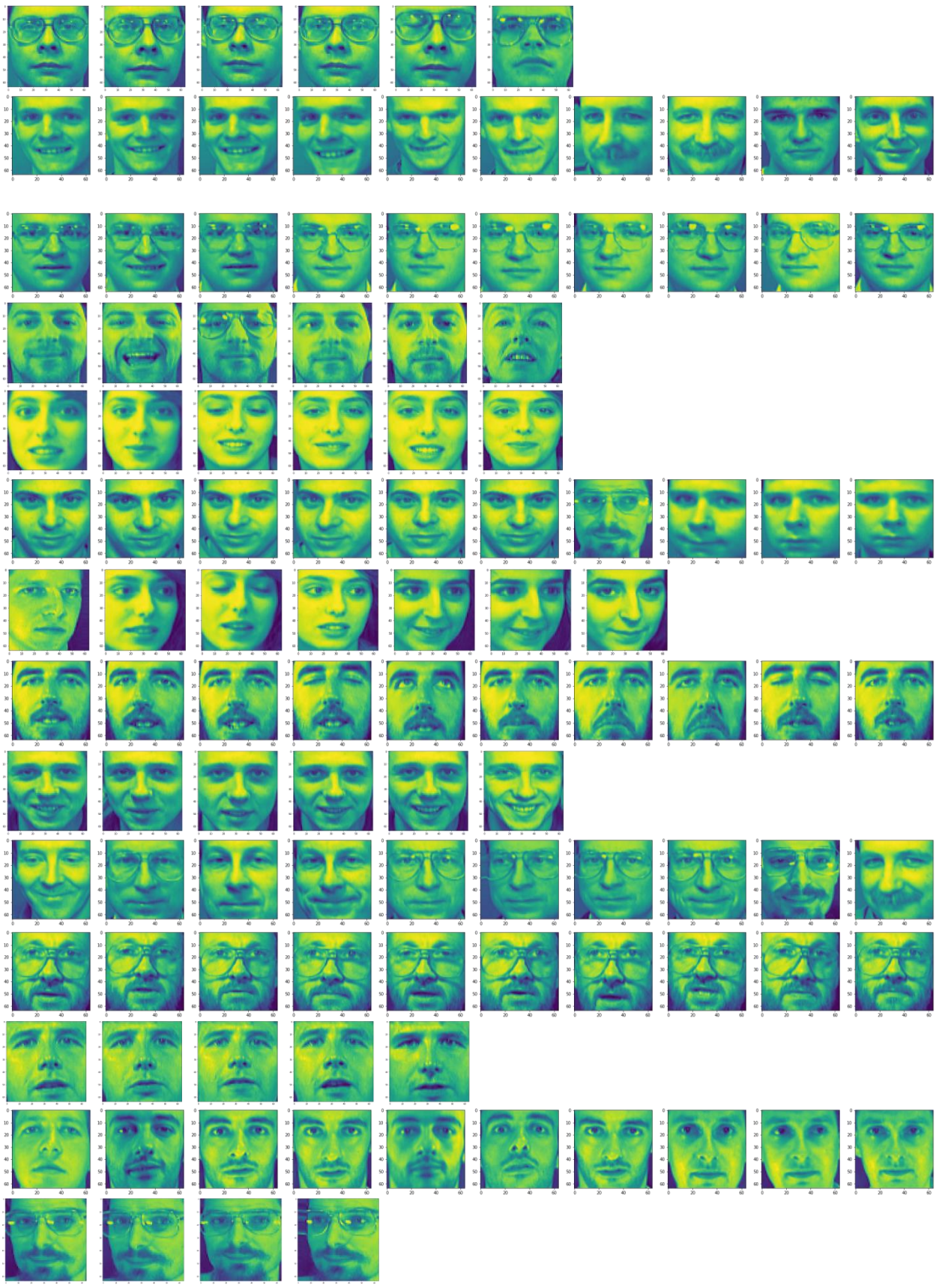
The final cluster centers are as follows

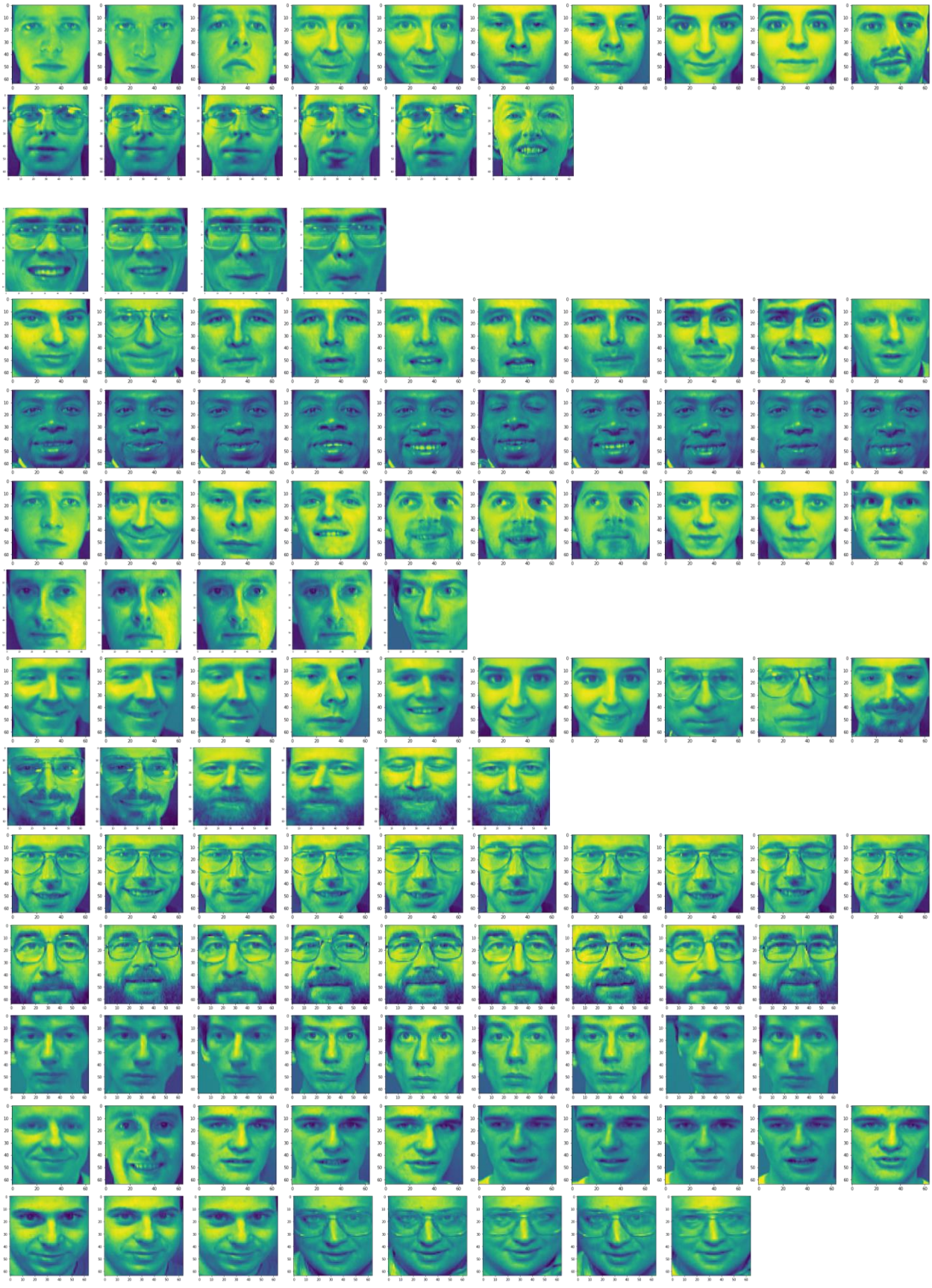


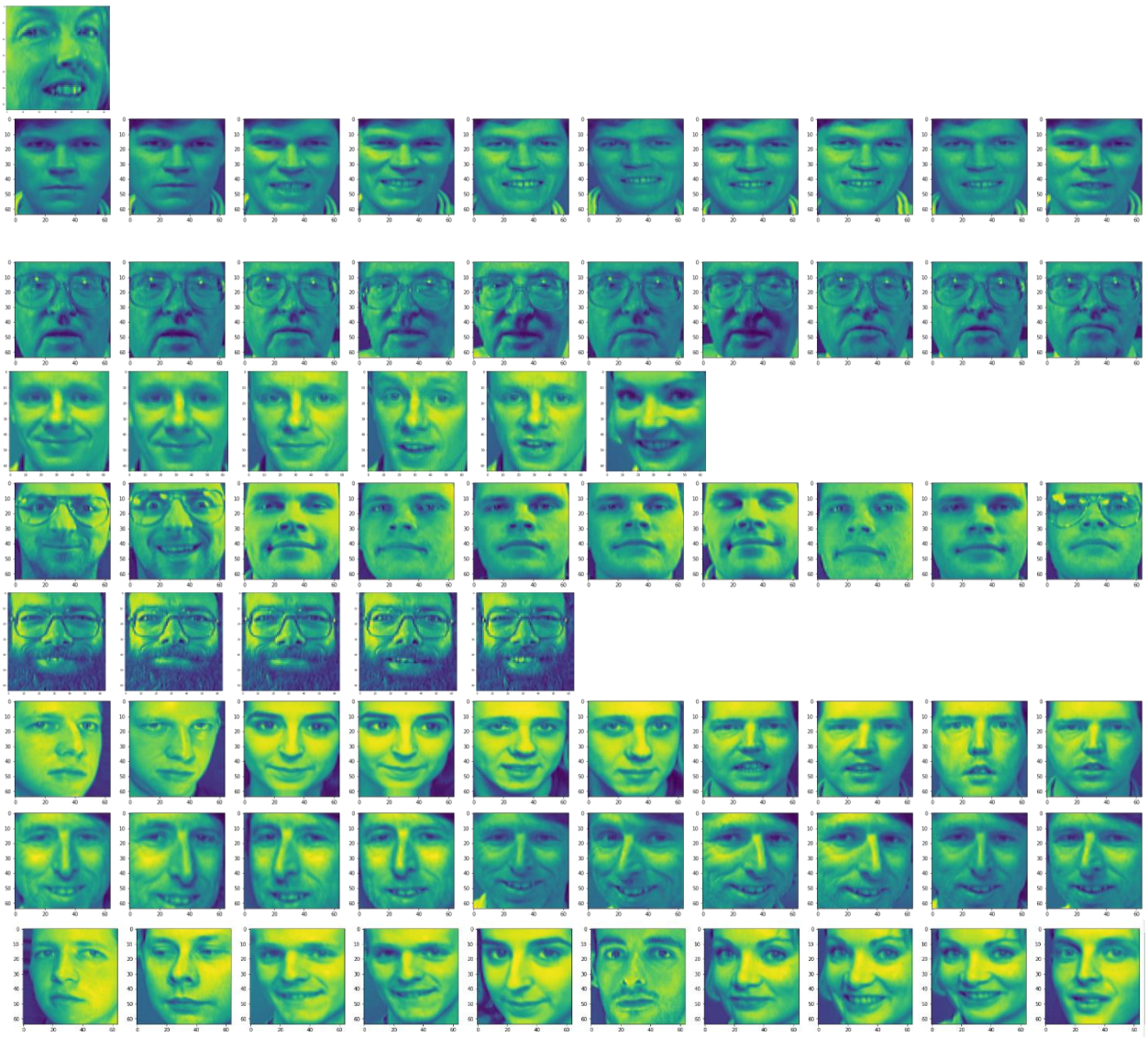
Part g)

I have done similar work here as well as done in part e) but for k means clustering classifier from scratch









Part h)

The sum of squared error for the clusters formed in part c) and f) are as follows

```
sse for K means classifier built from scratch 12724.861604690552
sse for K means classifier from sklearn library 12659.064950942993
```

Clearly, sse for K means classifier that is imported from sklearn library is less than the other one. Thus, the one imported from sklearn library (built in part f) performed better than the one built from scratch.

Q3)

Part a)

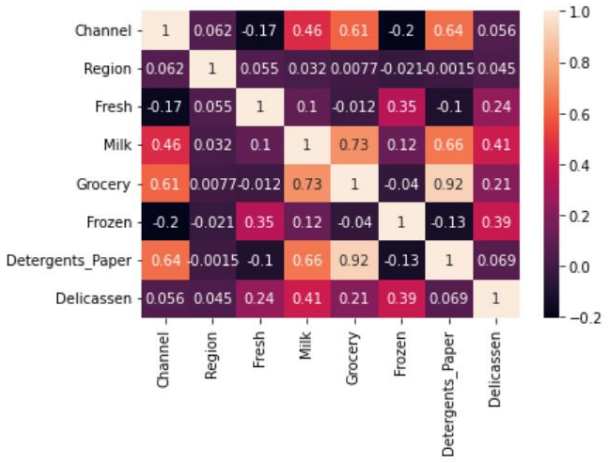
In this part, preprocessed the wholesale customers dataset. Here, to normalize the dataset, I have used min max normalization over all the features in the dataset. The preprocessed data is as follows

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	1.0	1.0	0.112940	0.130727	0.081464	0.003106	0.065427	0.027847
1	1.0	1.0	0.062899	0.132824	0.103097	0.028548	0.080590	0.036984
2	1.0	1.0	0.056622	0.119181	0.082790	0.039116	0.086052	0.163559
3	0.0	1.0	0.118254	0.015536	0.045464	0.104842	0.012346	0.037234
4	1.0	1.0	0.201626	0.072914	0.077552	0.063934	0.043455	0.108093
...	...	...	...	...	...	...	...	...
435	0.0	1.0	0.264829	0.163338	0.172715	0.215469	0.004385	0.045912
436	0.0	1.0	0.349761	0.018736	0.008202	0.073713	0.002205	0.048874
437	1.0	1.0	0.129543	0.210136	0.325943	0.006771	0.363463	0.038882
438	0.0	1.0	0.091727	0.026224	0.024025	0.016649	0.004042	0.044264
439	0.0	1.0	0.024824	0.022371	0.027022	0.000657	0.011611	0.001022

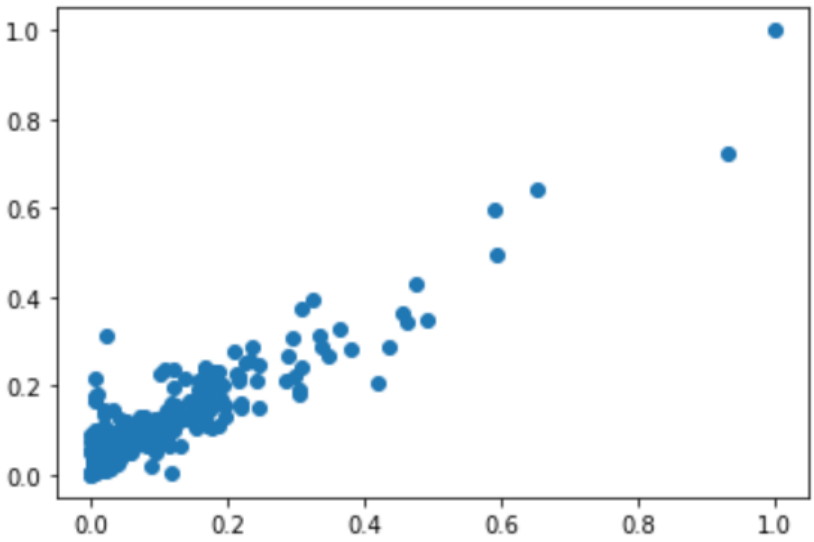
440 rows × 8 columns

Part b)

Using heat map from seaborn library, plotted the correlation between the features in the dataset

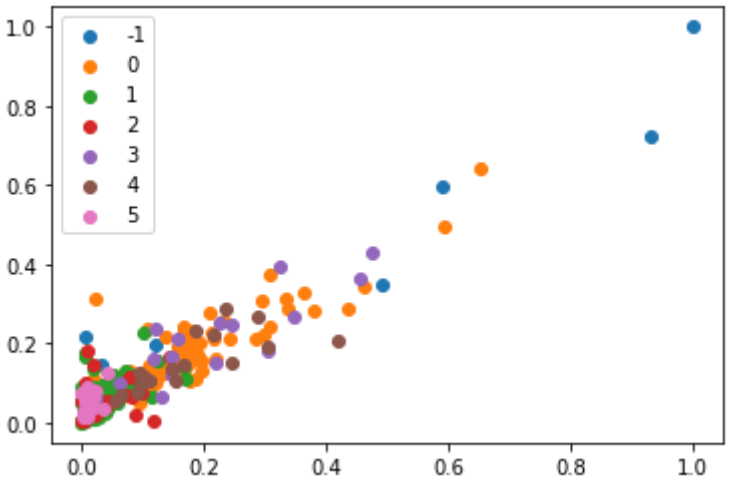


Clearly, from the heat map, the pair of features having highest correlation are Detergents_paper and Grocery. So, they will be best to visualize the clusters and outliers in the dataset. Visualization of the dataset on these two features is as follows



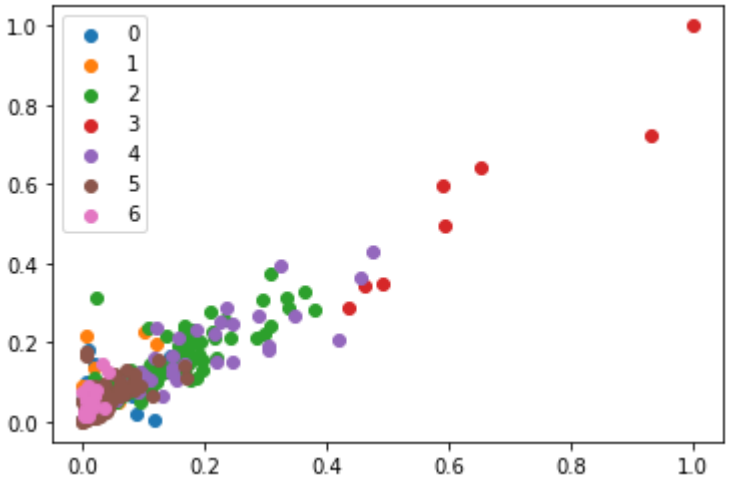
Part c)

In this part, trained a DBSCAN classifier on the dataset and visualized the clusters formed.



Part d)

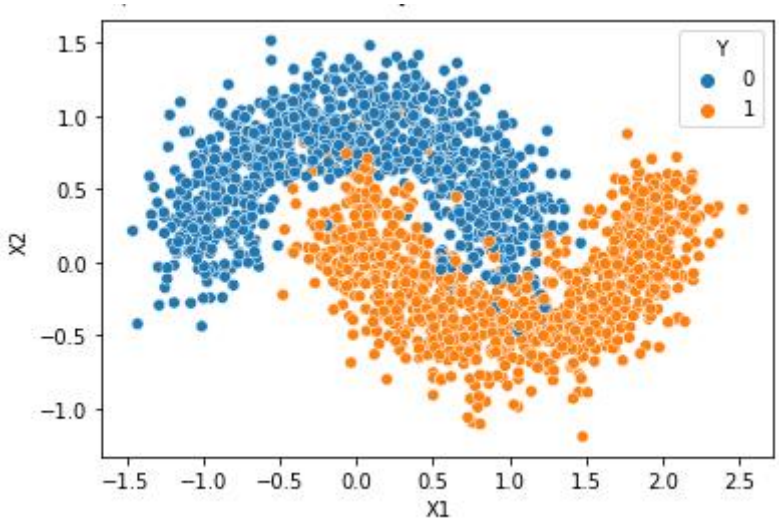
In this part, trained a Kmeans classifier on the dataset. The visualized clusters are as follows



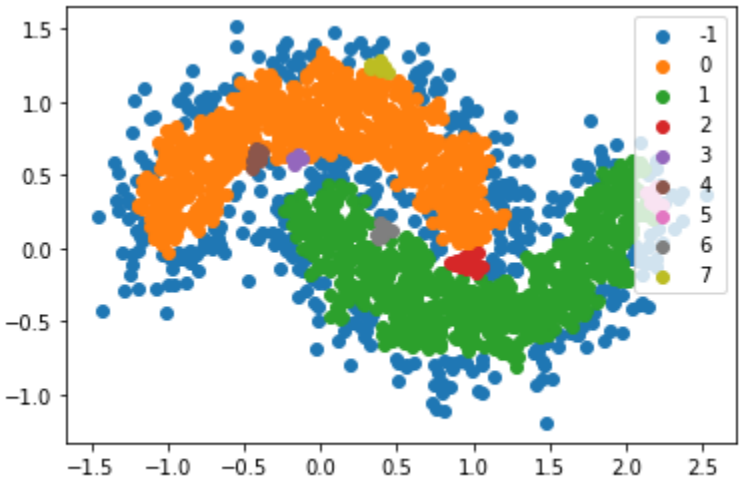
As we can see in the two graphs, clusters are more concentrated in the KMeans case while they are more spread out in DBscan case. Also, DBscan also identifying outliers and not considering them in forming clusters while this is not the case for KMeans.

Part e)

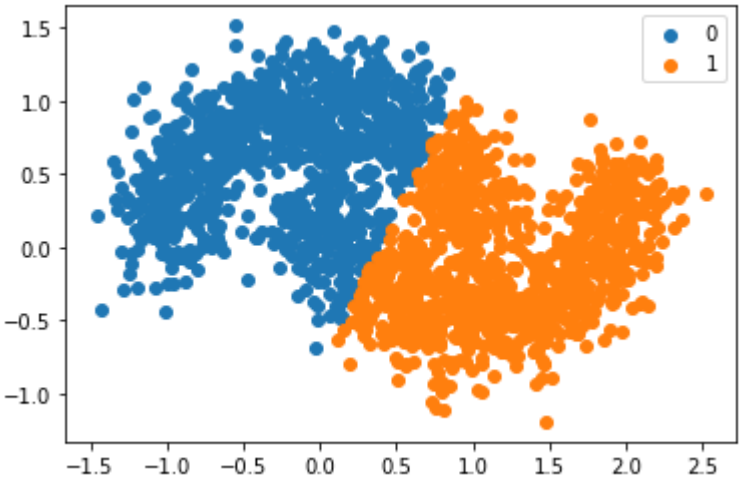
Generated a synthetic moon dataset using make_moons function from sklearn library



Now, found out the optimal values for eps and min_samples by simply running a for loop for respective ranges. The visualized clusters formed by the DBscan classifier are as follows



Again, trained a KMeans classifier on the same moon dataset with n_clusters=2 (known prior to us). The visualized clusters are as follows



Clearly, as we can see that, DBscan is separating the two moons more efficiently than KMeans. Although, DBscan is forming more clusters than two but we can group them into two groups depending on in which region maximum datapoints from that cluster lie in. Also, DBscan is identifying the outliers and not taking them into accound while forming the clusters. This removed the noise from the dataset. No outlier detection is done in the case of KMeans.