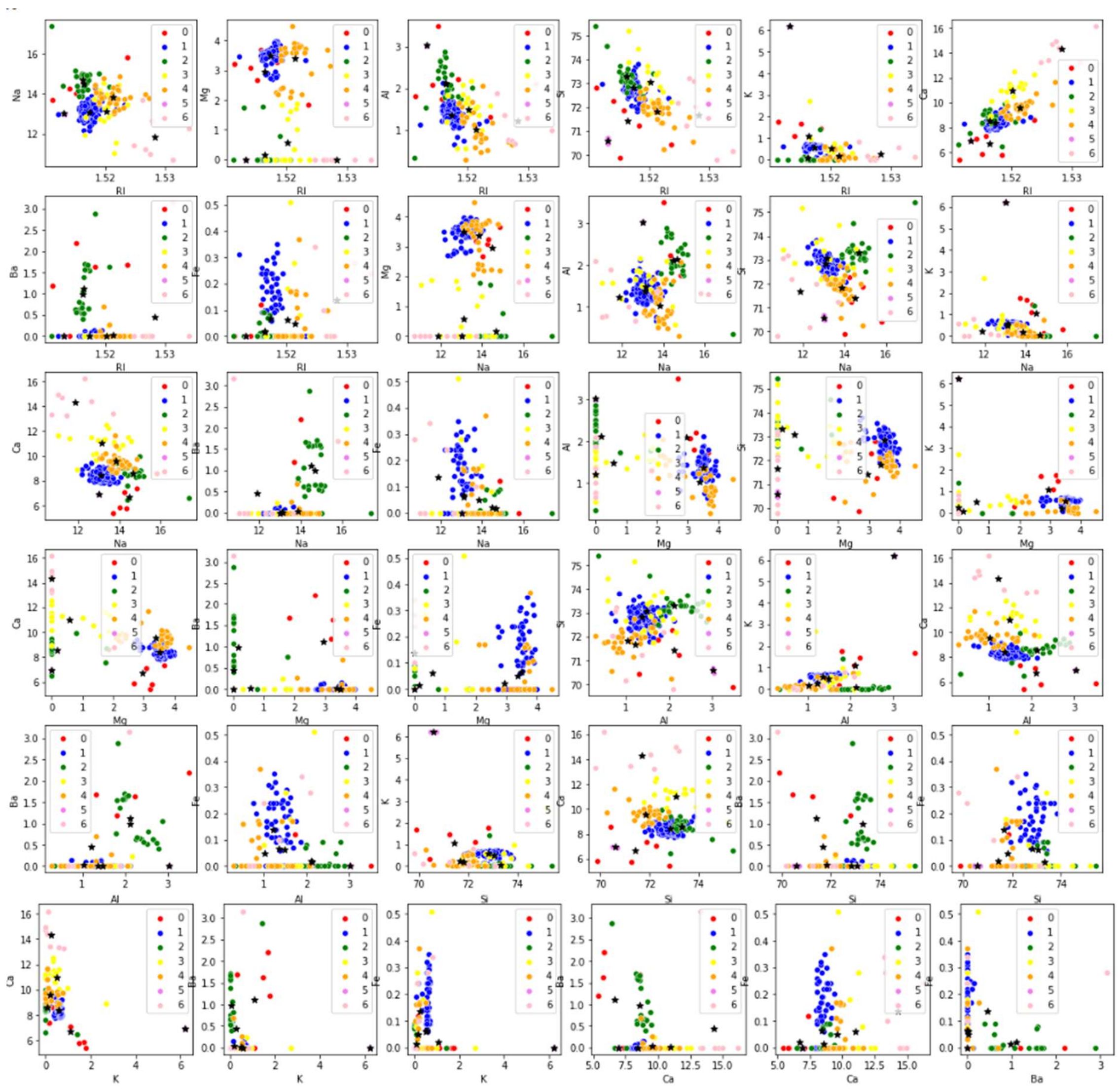
PRML LAB 6

Name: - Shalin Jain
Roll No: - B21CS070

Problem 1

Part a

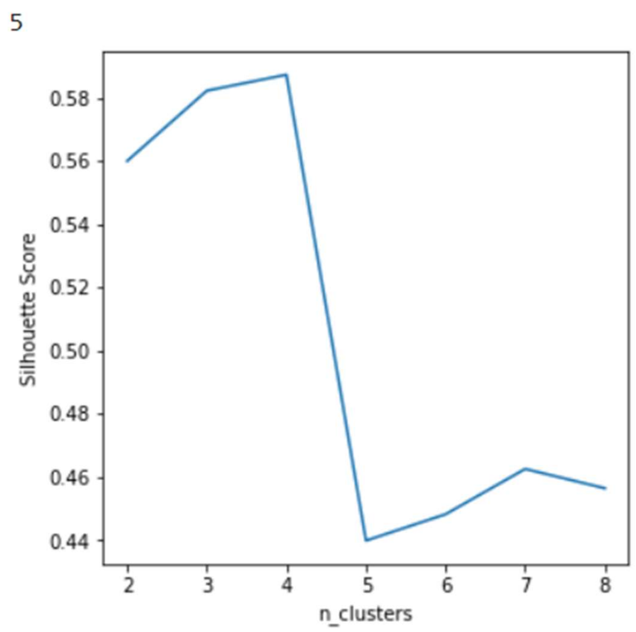
We have used KMeans with n_clusters as 7 to build the model and obtained clusters as shown in the graphs below. The black * marks in the graph represent the centroids of the respective clusters.



We can see that for different values of two features different clusters are plotted and in some cases like Si vs Mg and Ca vs Na we can see clear distinction in the clusters.

Part b)

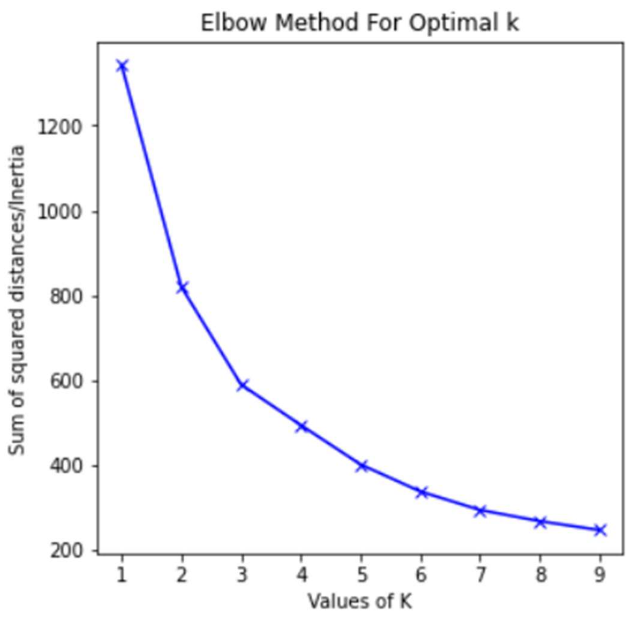
In this we have varied the value of n_clusters and saw the Silhouette Score and we know that this score should be minimum for the optimal value of n_clusters.



We can see that this score is maximum for n_clusters = 4 form the above graph.

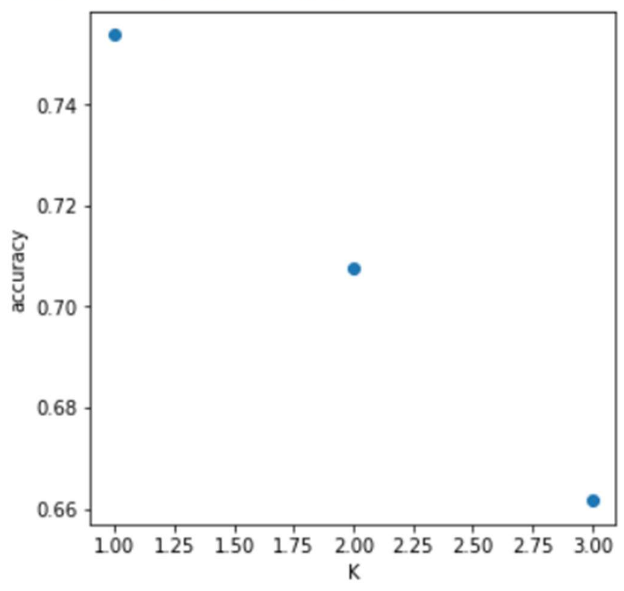
Part c

In this part we have used the elbow method to find optimal value of k i.e. n_clusters which is also near by 4 as obtained in the previous part.



Part d

```
Text(0, 0.5, 'accuracy')
```



```
0.35384615384615387
```

As we can see that the accuracy decreases with an increase number of neighbors the value of k in KNN. And accuracy increases as shown above. The initial accuracy is nearly 35% but accuracy has been increased to 70% with including KNN with bagging.

Problem 2

Part a and b

In this parts we have imported the image data and flatten the it as stated in the question.
Now we have made the KMeans class as shown in the code.

`_euclidean_distance` function: - This function gives the distance array of the distances of a point to all the other point in the given passed set of points.

`fit` function: - This function takes number of cluster and initial cluster centres as input. If the initial cluster centres are not equal to the number of clusters the it adds the remaining possible clusters to the cluster array. It also takes maximum iteration as user input and recluster till there is a very little change in centroids points.

`predict` function: - This function takes datapoint/datapoints as input and predict which cluster they may belong to.

`count_of_points` function: - This function gives the count of points in each cluster.

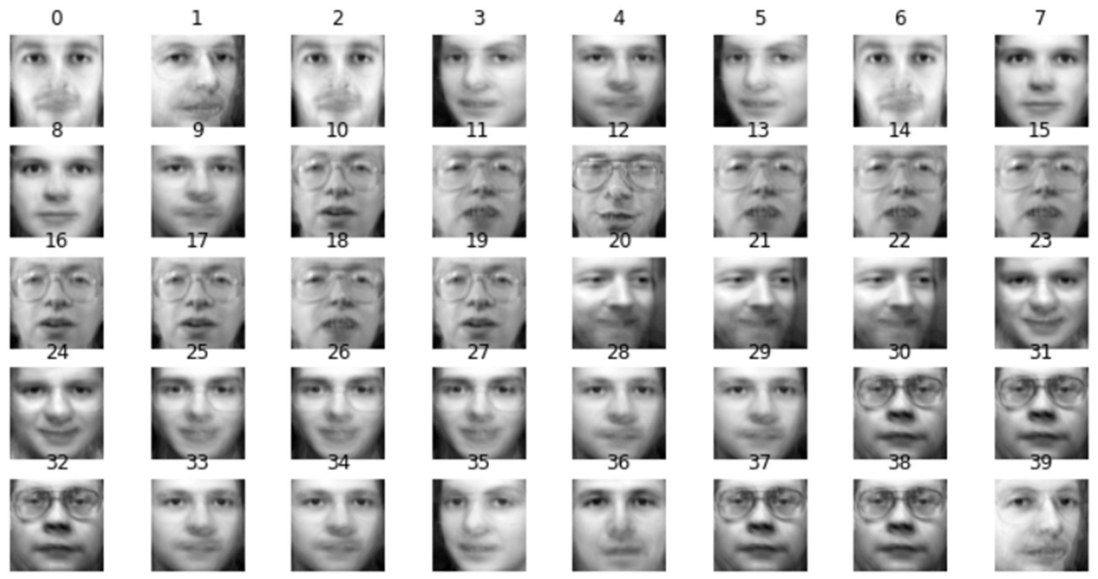
Part c

In this part we have give 40 points as initialization centres for the cluster and find out the number points in each cluster as shown. `{2: 19, 20: 6, 10: 11, 21: 17, 18: 11, 15:`

```
7, 14: 8, 13: 2, 26: 7, 34: 21, 28: 18, 30: 5, 16: 26, 19: 42, 8: 9,
24: 14, 3: 7, 35: 8, 6: 3, 39: 7, 9: 3, 29: 5, 0: 15, 33: 4, 7: 16, 5:
4, 17: 4, 4: 7, 27: 4, 36: 8, 37: 8, 22: 1, 38: 9, 1: 17, 12: 6, 31: 4,
32: 13, 25: 9, 23: 10, 11: 5}
```

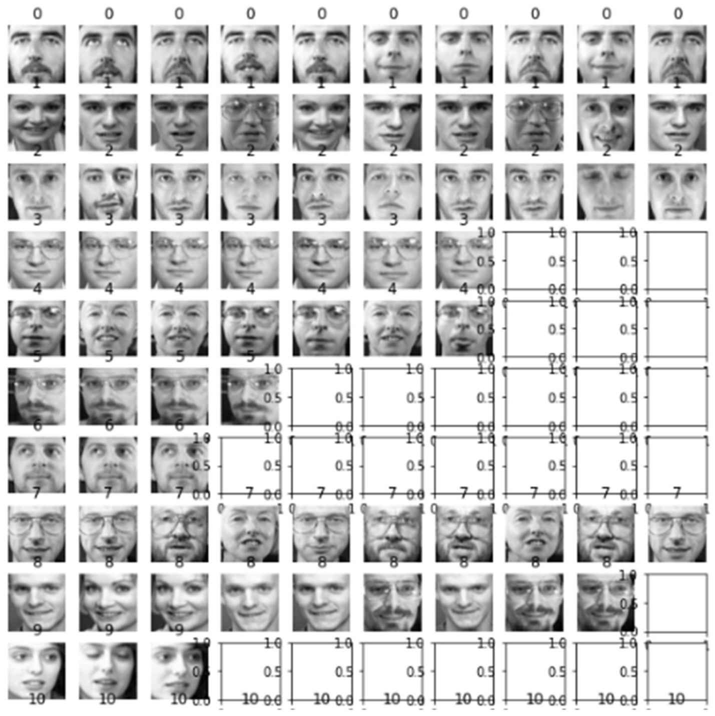
Part d

In this part we have used the cluster centres and plotted the image as shown:



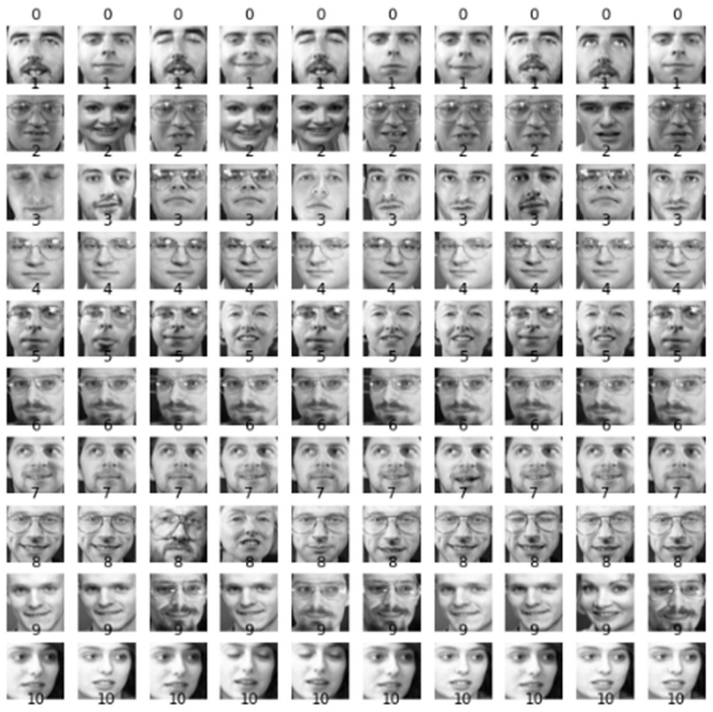
Part e

For 10 images in each class clusters(if 10 images are not present then there we left a blank box):



A sample of 10 clusters is shown

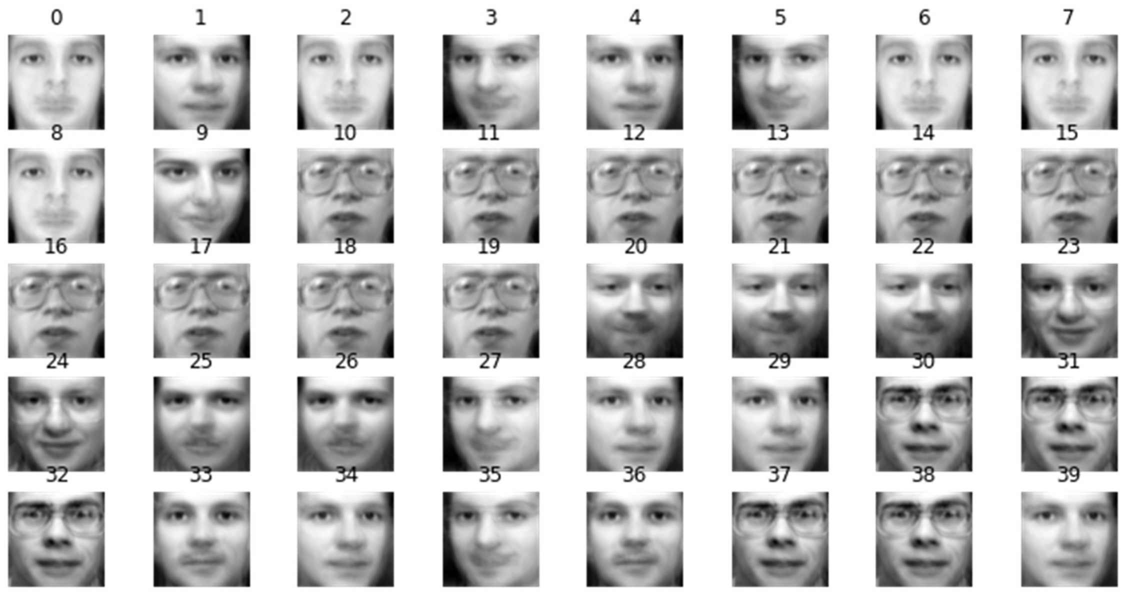
For 10 images being forced clusters are given as:



Part f

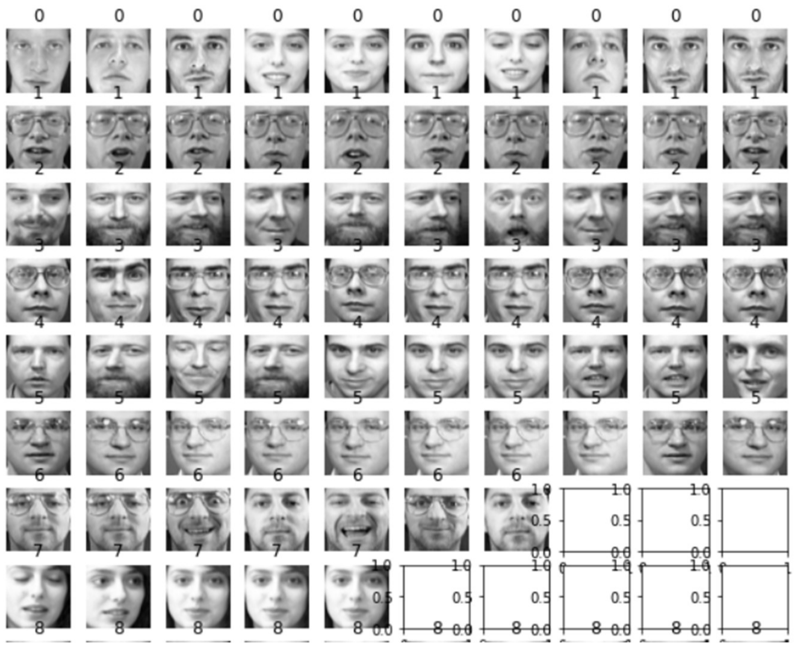
In this we have passed 40 class centers as initialization and obtained the number of images in each cluster as follows:

```
{0: 14, 17: 16, 37: 21, 39: 9, 1: 10, 2: 13, 30: 14, 4: 23, 3: 11, 22: 21,
29: 19, 5: 10, 6: 7, 7: 5, 8: 13, 9: 3, 11: 6, 10: 15, 12: 11, 13: 10, 14:
5, 15: 2, 16: 4, 18: 5, 19: 4, 21: 10, 24: 10, 23: 17, 26: 10, 27: 10, 28:
9, 20: 5, 31: 10, 32: 10, 33: 10, 34: 6, 35: 2, 36: 5, 25: 5, 38: 10}
```



Part g

Visualization of image is as follows:



Part h

In this part we have used SSE (error) to evaluate the above two models. The obtained scores are as follows:

```
SSE for original KMeans model: 13083.177734375
SSE for new KMeans model: 12182.69140625
```

Original: - datapoint center as initial centers
New: - class as initial

We can error in case of normal datapoint case is more than the classes because classes can give a more accurate depiction of labels rather than random datapoints as initial canters.

Problem 3

Part a

Preprocess the dataset and scaled it using Standard Scaler library of python also check for the null values using df.info() but there was not any null values.

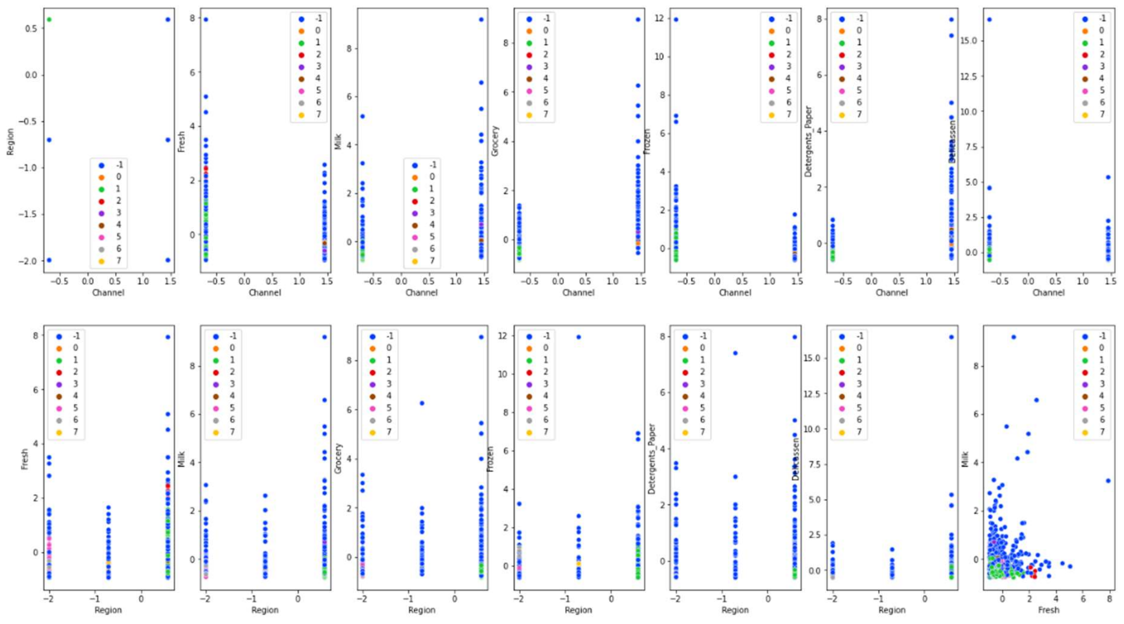
Part b

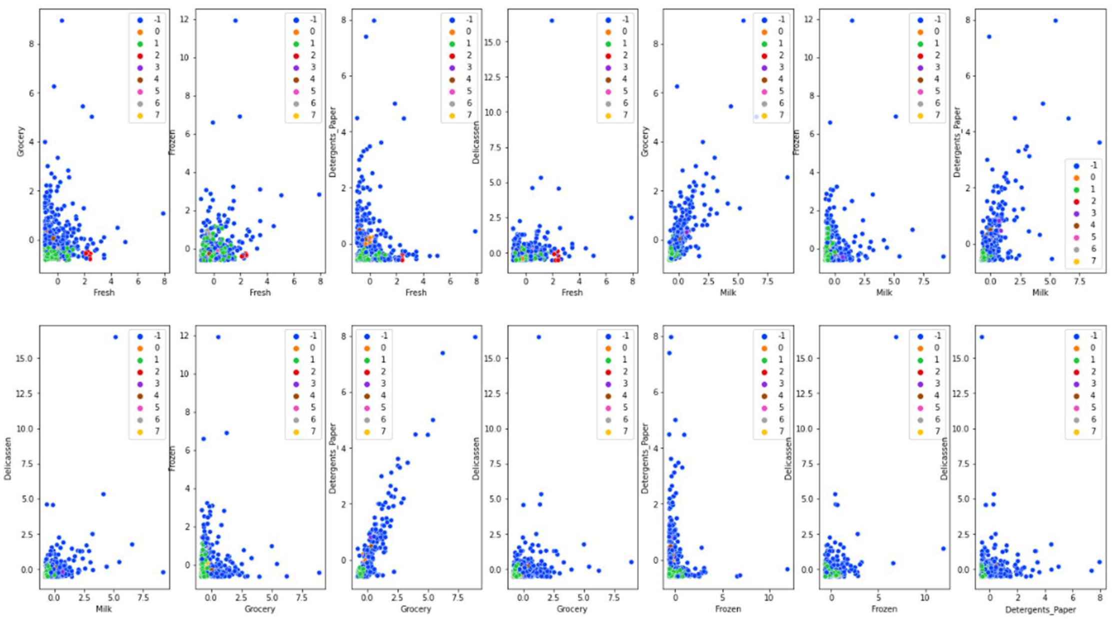
Now we found out the covariance of each pair of feature and to find out the outlier we will find the pair of distinct feature with the highest absolute covariance.

On searching we see that the Grocery and Detergents_Paper has the highest covariance between them.

Part c

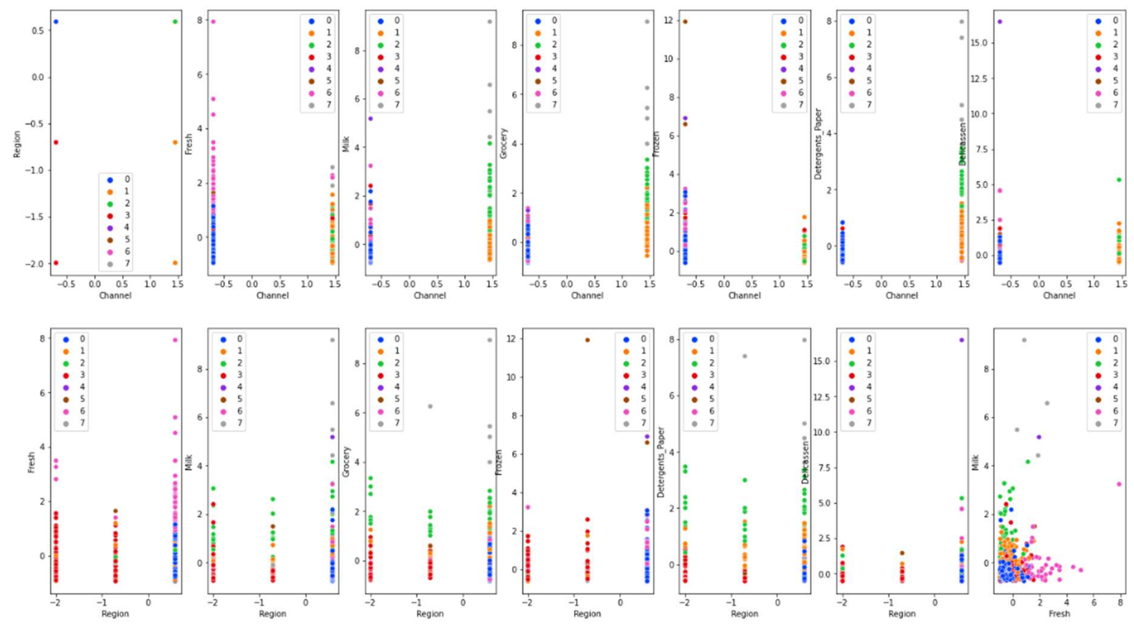
Now we trained the DBSCAN model for clustering the datapoints and plots of different clusters using two features as component axis are as shown:

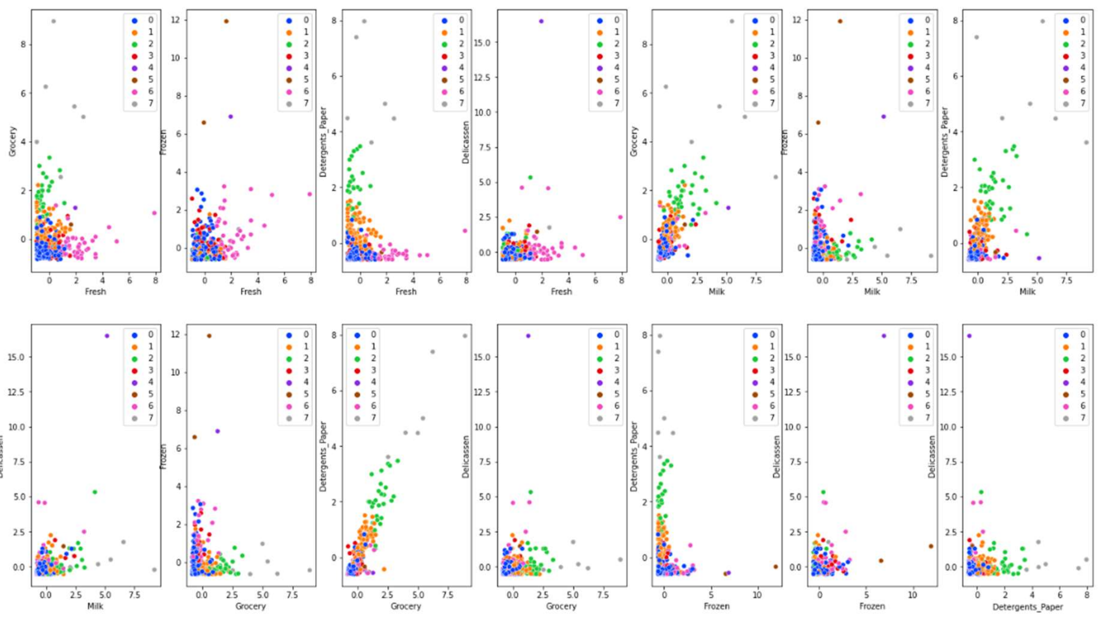




Part d

Now we trained the KMeans model for clustering the datapoints and plots of different clusters using two features as component axis are as shown:

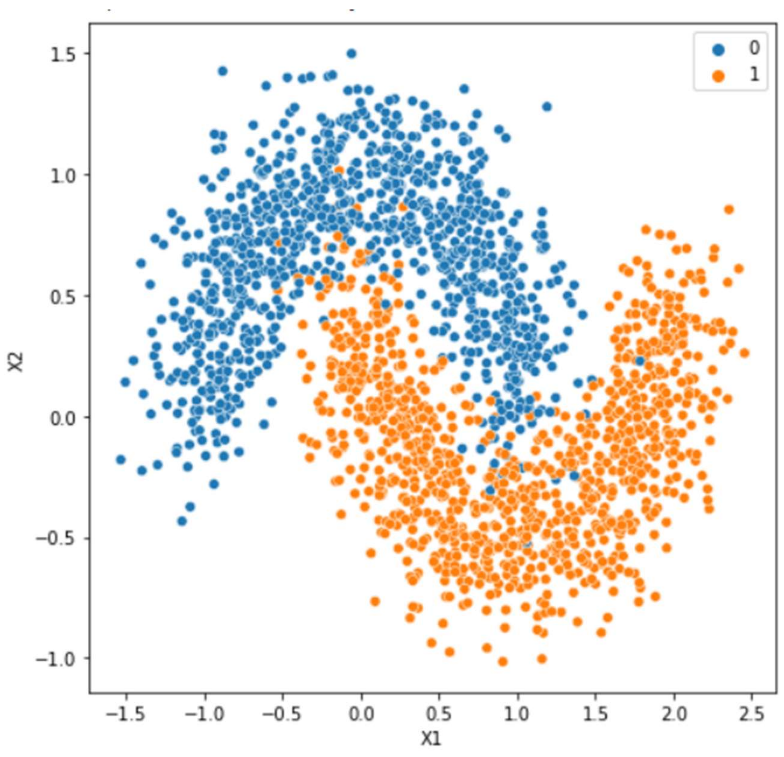




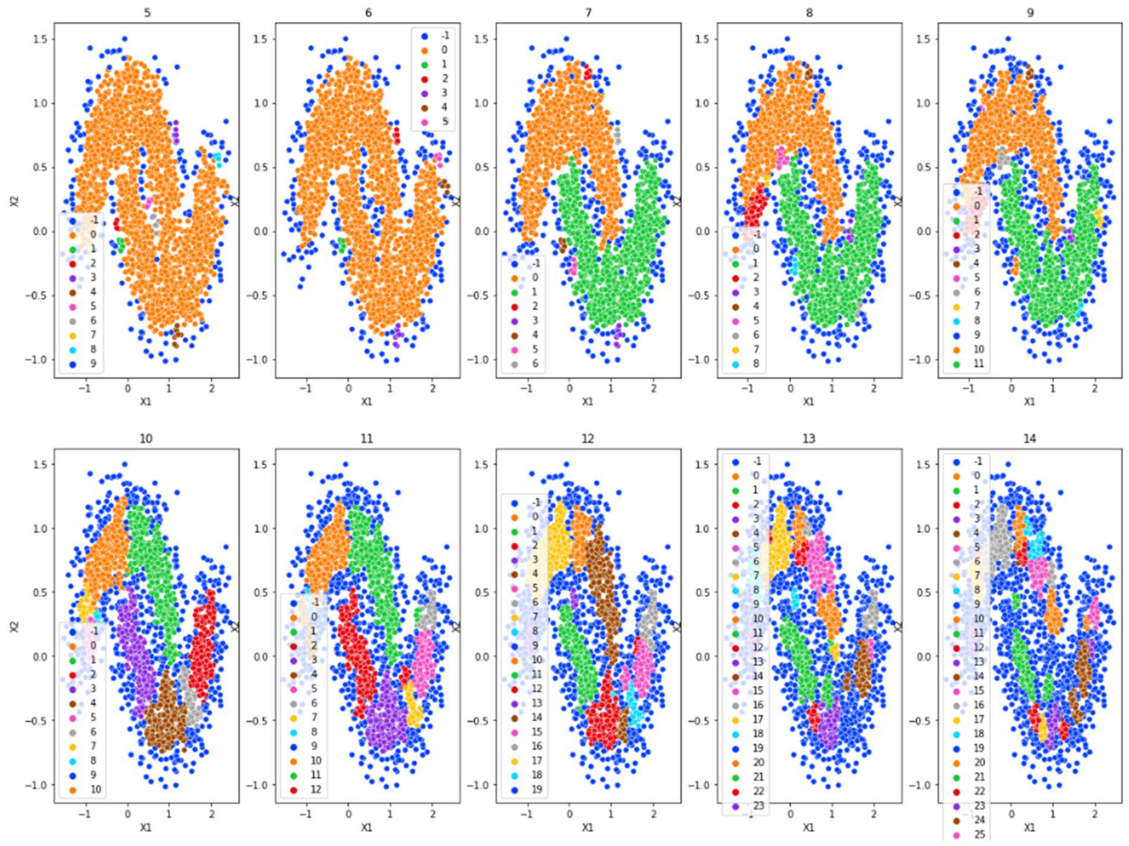
We can see both DBSCAN and the KMeans divide the datapoints into 7 clusters and DBSCAN doesn't care about the outliers of the dataset and hence it is more accurate.

Part e

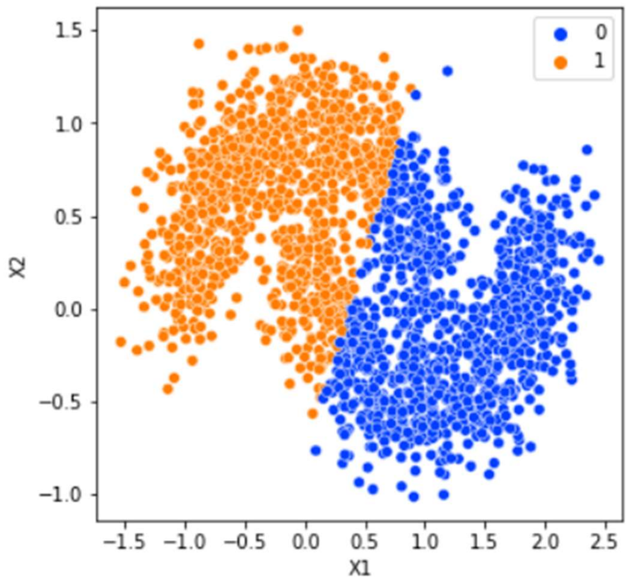
We made a make_moons dataset of 2000 points and added noise of 20% in it. The obtained dataset looks like as shown below:



When we run DBSCAN model on the dataset and varied the min_samples and took eps as 0.08 we got the following plot and we can see that at min_samples = 7 the clusters nearly separated with the boundary treated as noise.



When we ran the KMeans on the same dataset the obtained clusters was as given below:



We can see that the DBSCAN at some point clusterize the dataset more accurately