ASSIGNMENT 2 - K Means Clustering

GROUP 48:

19CS30032 - P Anurag Reddy

19CS10061 - Shashank Suroju

DATA:

The given data contained information about patients with liver disorder. The data was provided in .csv format. (Link to data set - Indian Liver Patient Dataset: https://www.kaggle.com/jeevannagaraj/indian-liver-patient-dataset.)

The data has the following attributes:

1.	*age* :	Age of the patient
2.	*gender* :	Gender of the patient
3.	*tot_bilirubin* :	Total Bilirubin
4.	*direct_bilirubin* :	Direct Bilirubin
5.	*alkphos* :	Alkaline Phosphotase
6.	*sgpt* :	Alamine Aminotransferase
7.	*sgot* :	Aspartate Aminotransferase
8.	*tot_proteins* :	Total Protiens
9.	*albumin* :	Albumin
10.	ag_ratio :	Albumin and Globulin Ratio
11.	is_patient :	Ground Truth of the data

The attribute gender had values ‘Male’ and ‘Female’. We changed them to 1 and 2 respectively.

The attribute alkphos had missing data. We filled the missing positions with its mean value.

FUNCTIONS:

Part1:

euclidean_distance:

The function takes two n dimensional points and returns their euclidean distance.

get_closest_centroid:

The function takes all the data points and centroids as input. Then, it calculates the distance between each data point and centroids in order to cluster them to the centroid which is closest to them. Returns a list of indexes of clusters to which data point is closest to.

create_cluster:

It takes centroids as input. Then calls closest_centorid to get the cluster index of each data point. Then appends the data points to respective cluster lists and returns the list.

get_centroids:

Takes the cluster list as input. Then computes the new centroids of a cluster by calculating the mean of the data points corresponding to that cluster. Returns the new centroids.

is_converged:

Takes the old centroids and new centroids as input. If the distance between them is zero then it returns true else it returns false.

predict:

Takes data as input. Calls create_cluster, get_centoid and is_converged iteratively until maximum number of iterations has been reached or the centroids have converged.

assign_cluster_labels:

Takes final clustering information as input and returns a numpy array which contains the labels of each data point.

Part4:

find_centroids:

Takes data points and k-centroids as input and calls the K Means algorithm on them. Returns the final centroids computed by using the data points and the initial centroids.

TestA:

Evaluating the average NMI by choosing k random centroids ,splitting data, applying K-means on training data and labeling the test data using the final centroids on training data, finally calculates the clustering accuracy through NMI.

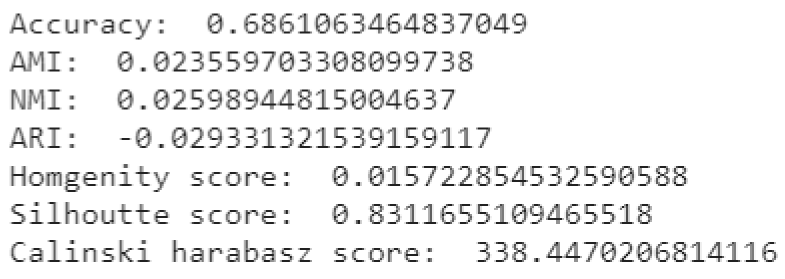
The NMI is computed 2500 times as we are doing 50 iterations of TestA and in each iteration we are computing NMI 50 times for each random initialization. Average NMI is computed for each iteration of TestA giving a total of 50 average NMI values.

RESULTS:

We are taking K=2 in our models as the given data is divided into 2 labels.

1. **Accuracy(Ground Truth):**
   We are using the accuracy_score function from sklearn.metrics to measure the accuracy of our model. It compares the predicted labels with ground truth and returns the accuracy.
   Note: The accuracy may differ for some iterations as the cluster labels can be either 0,1 or 1,0. This affects the outcome.

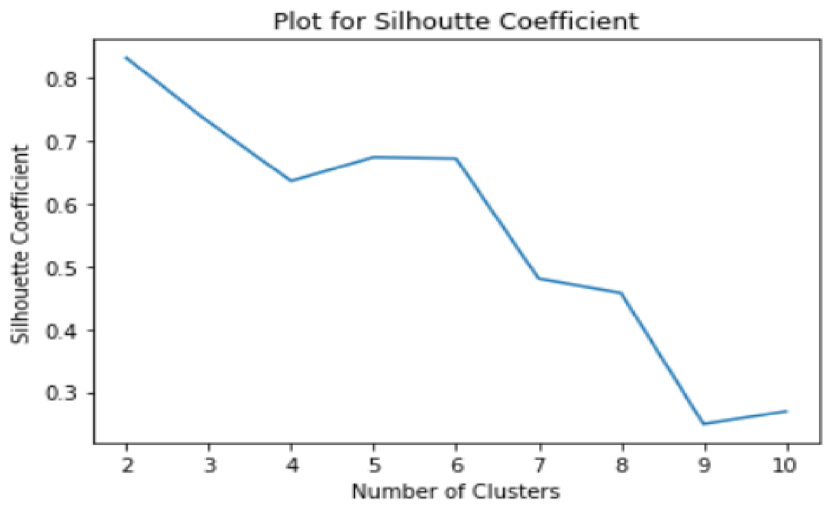
2. **Accuracy(Without Ground Truth):**
   We measured performance using various metrics such as ARI, NMI, AMI etc.



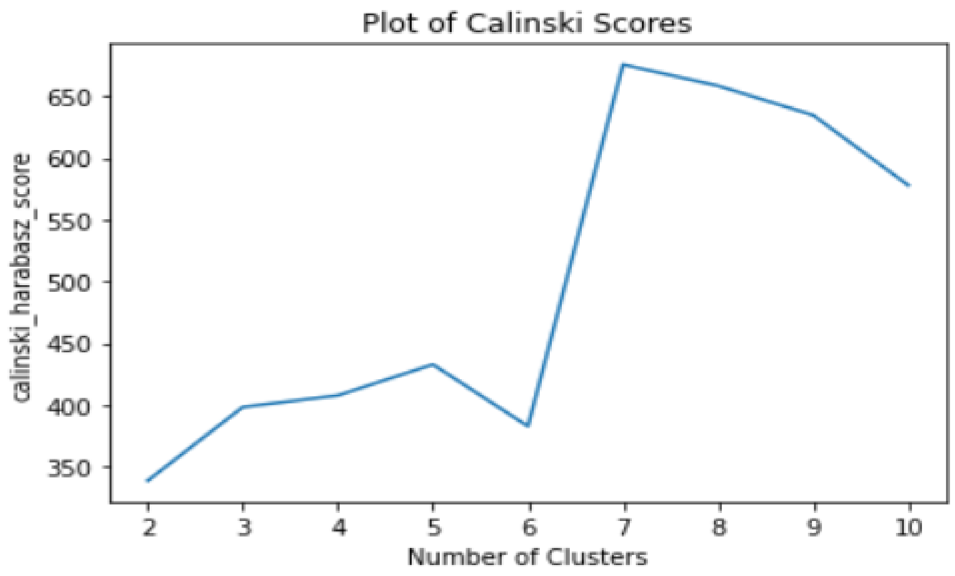
```
Accuracy:  0.6861063464837049
AMI:  0.023559703308099738
NMI:  0.02598944815004637
ARI:  -0.029331321539159117
Homgenity score:  0.015722854532590588
Silhoutte score:  0.8311655109465518
Calinski harabasz score:  338.4470206814116
```

Performance Measures for K = 2

3. **Finding Optimal K:**
   We used silhouette index and calinski harabasz score as a measure for finding optimal K for the given data.



Optimal number of clusters found from the silhouette index is 2.

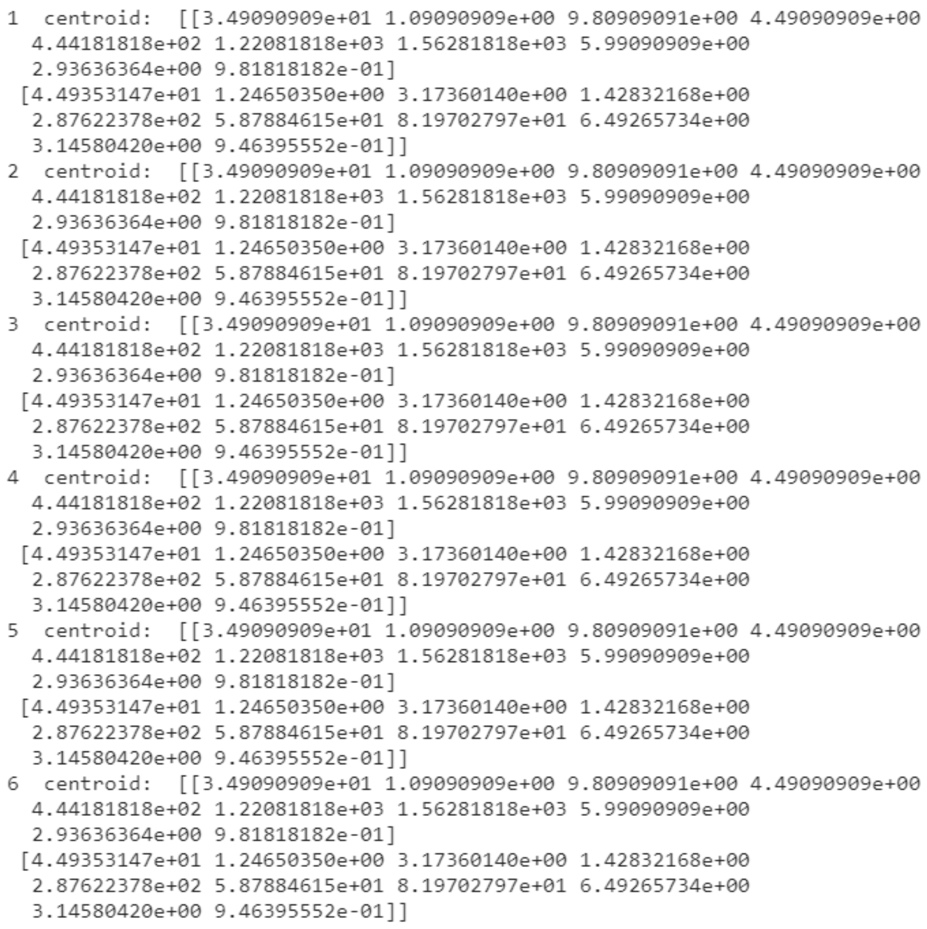


Optimal number of clusters found from the silhouette index is 7.

4. **TestA:**
   Note: We have considered k=2 as the most suitable k.

We observed no change in the outcome of clustering as for all the iterations the clusters have converged to the same centroids. Hence, we didn’t use any heuristics as there is no observed variation and performed TestA with just random initialisation of K points.

Below is a screen shot of the centroids of the first few iterations. We can observe that all of them are the same points. All the remaining centroids are also the same. This is to prove that we had no observed variation in outcome.

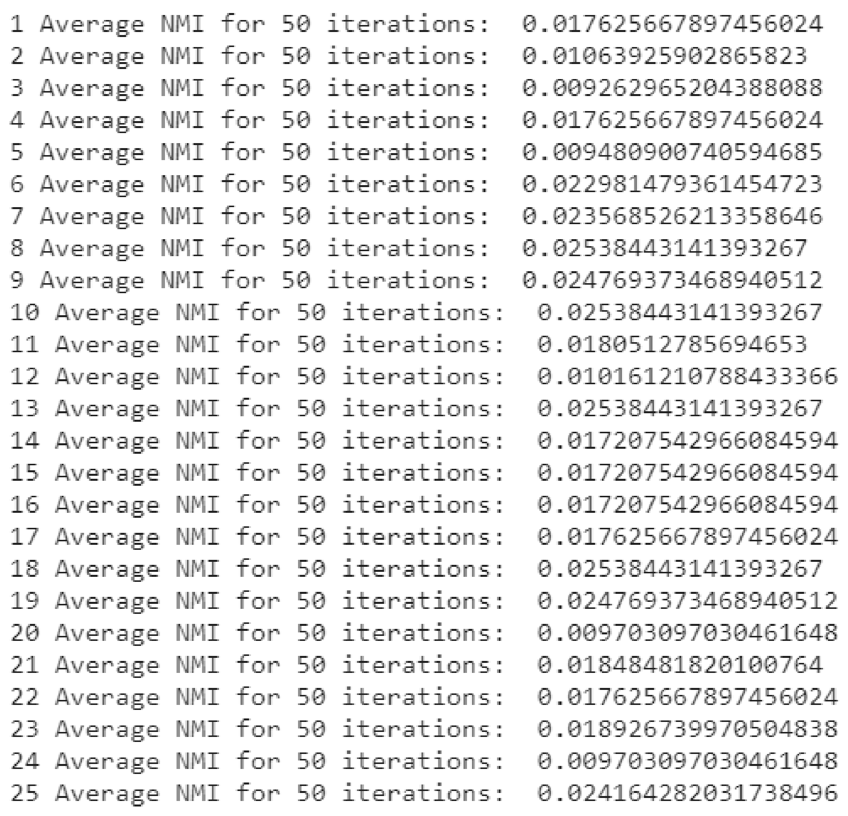


```
1  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
2  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
3  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
4  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
5  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
6  centroid:  [[3.49090909e+01 1.09090909e+00 9.80909091e+00 4.49090909e+00
  4.44181818e+02 1.22081818e+03 1.56281818e+03 5.99090909e+00
  2.93636364e+00 9.81818182e-01]
 [4.49353147e+01 1.24650350e+00 3.17360140e+00 1.42832168e+00
  2.87622378e+02 5.87884615e+01 8.19702797e+01 6.49265734e+00
  3.14580420e+00 9.46395552e-01]]
```

Centroids for the first 6 iterations of TestA

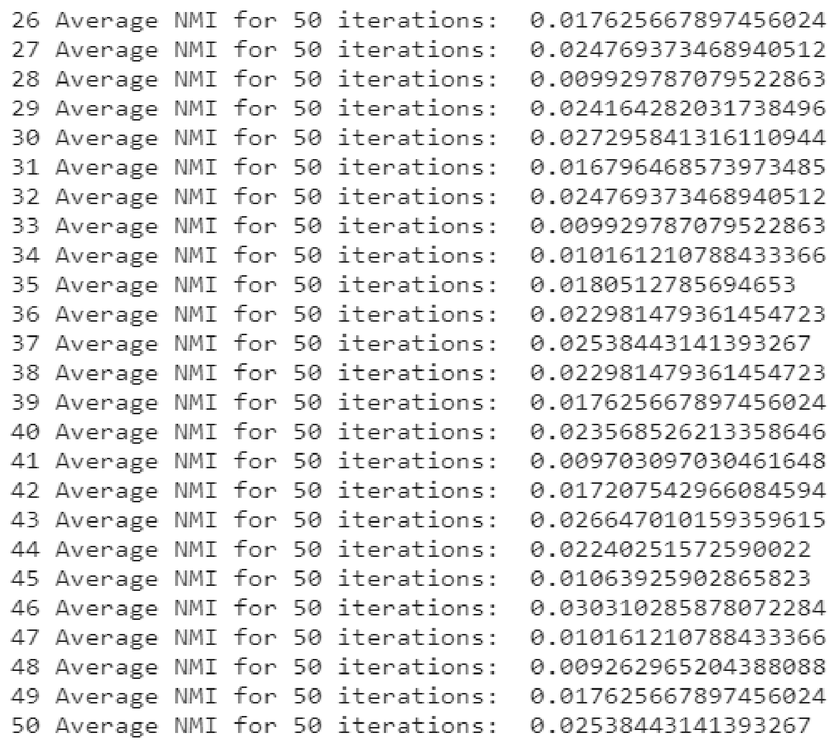
NMI’s:

The difference in the NMI values is observed due to different splitting of data at each random initialization and also the variation in NMI values is minimal.



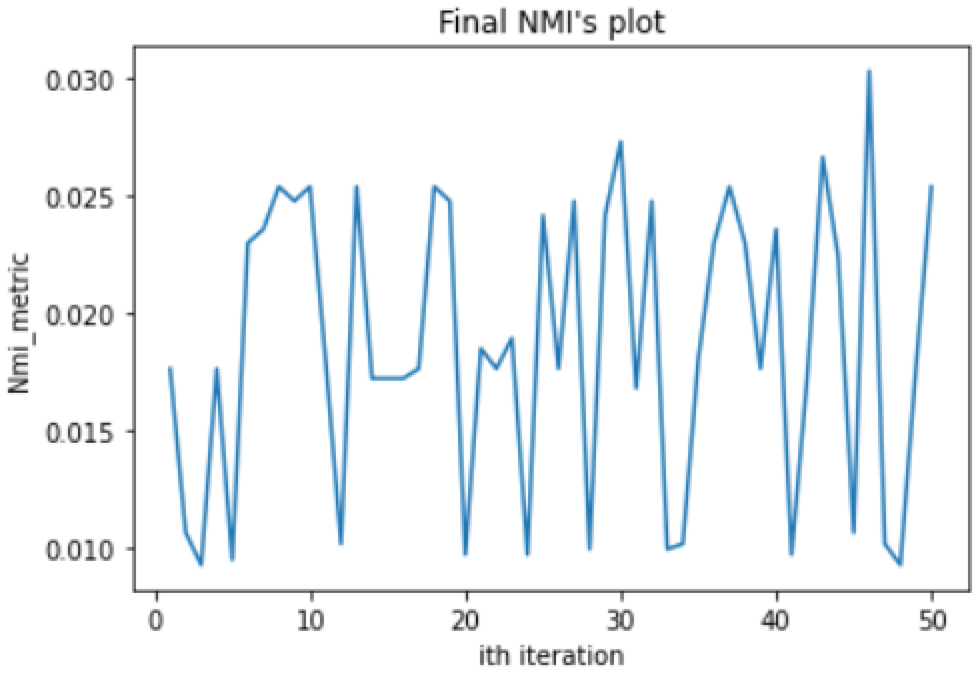
```
1 Average NMI for 50 iterations:  0.017625667897456024
2 Average NMI for 50 iterations:  0.01063925902865823
3 Average NMI for 50 iterations:  0.009262965204388088
4 Average NMI for 50 iterations:  0.017625667897456024
5 Average NMI for 50 iterations:  0.009480900740594685
6 Average NMI for 50 iterations:  0.022981479361454723
7 Average NMI for 50 iterations:  0.023568526213358646
8 Average NMI for 50 iterations:  0.02538443141393267
9 Average NMI for 50 iterations:  0.024769373468940512
10 Average NMI for 50 iterations:  0.02538443141393267
11 Average NMI for 50 iterations:  0.0180512785694653
12 Average NMI for 50 iterations:  0.010161210788433366
13 Average NMI for 50 iterations:  0.02538443141393267
14 Average NMI for 50 iterations:  0.017207542966084594
15 Average NMI for 50 iterations:  0.017207542966084594
16 Average NMI for 50 iterations:  0.017207542966084594
17 Average NMI for 50 iterations:  0.017625667897456024
18 Average NMI for 50 iterations:  0.02538443141393267
19 Average NMI for 50 iterations:  0.024769373468940512
20 Average NMI for 50 iterations:  0.009703097030461648
21 Average NMI for 50 iterations:  0.01848481820100764
22 Average NMI for 50 iterations:  0.017625667897456024
23 Average NMI for 50 iterations:  0.018926739970504838
24 Average NMI for 50 iterations:  0.009703097030461648
25 Average NMI for 50 iterations:  0.024164282031738496
```

NMI values for the first 25 iterations



```
26 Average NMI for 50 iterations:  0.017625667897456024
27 Average NMI for 50 iterations:  0.024769373468940512
28 Average NMI for 50 iterations:  0.009929787079522863
29 Average NMI for 50 iterations:  0.024164282031738496
30 Average NMI for 50 iterations:  0.027295841316110944
31 Average NMI for 50 iterations:  0.016796468573973485
32 Average NMI for 50 iterations:  0.024769373468940512
33 Average NMI for 50 iterations:  0.009929787079522863
34 Average NMI for 50 iterations:  0.010161210788433366
35 Average NMI for 50 iterations:  0.0180512785694653
36 Average NMI for 50 iterations:  0.022981479361454723
37 Average NMI for 50 iterations:  0.02538443141393267
38 Average NMI for 50 iterations:  0.022981479361454723
39 Average NMI for 50 iterations:  0.017625667897456024
40 Average NMI for 50 iterations:  0.023568526213358646
41 Average NMI for 50 iterations:  0.009703097030461648
42 Average NMI for 50 iterations:  0.017207542966084594
43 Average NMI for 50 iterations:  0.026647010159359615
44 Average NMI for 50 iterations:  0.02240251572590022
45 Average NMI for 50 iterations:  0.01063925902865823
46 Average NMI for 50 iterations:  0.030310285878072284
47 Average NMI for 50 iterations:  0.010161210788433366
48 Average NMI for 50 iterations:  0.009262965204388088
49 Average NMI for 50 iterations:  0.017625667897456024
50 Average NMI for 50 iterations:  0.02538443141393267
```

NMI values for the second 25 iterations



NMI dispersion plot