MACHINE LEARNING GROUP-40 ASSIGNMENT-2 REPORT

Neha Dalmia 19CS30055 **Hritaban Ghosh 19CS30053**

Task:

Dataset Link : https://archive.ics.uci.edu/ml/datasets/liver+disorders

The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the dataset constitutes the record of a single male individual.

Important note: The 7th field (selector) has been widely misinterpreted in the past as a dependent variable representing presence or absence of a liver disorder. This is incorrect [1]. The 7th field was created by BUPA researchers as a train/test selector. It is not suitable as a dependent variable for classification. The dataset does not contain any variable representing presence or absence of a liver disorder. Researchers who wish to use this dataset as a classification benchmark should follow the method used in experiments by the donor (Forsyth & Rada, 1986, Machine learning: applications in expert systems and information retrieval) and others (e.g. Turney, 1995, Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm), who used the 6th field (drinks), after dichotomising, as a dependent variable for classification. Because of widespread misinterpretation in the past, researchers should take care to state their method clearly.

The task requires us to cluster the samples based on the following attributes:

Attribute information:

1. mcv mean corpuscular volume
2. alkphos alkaline phosphotase
3. sgpt alamine aminotransferase
4. sgot aspartate aminotransferase
5. gammagt gamma-glutamyl transpeptidase
6. drinks number of half-pint equivalents of alcoholic beverages drunk per day

Data Analysis and Visualisation

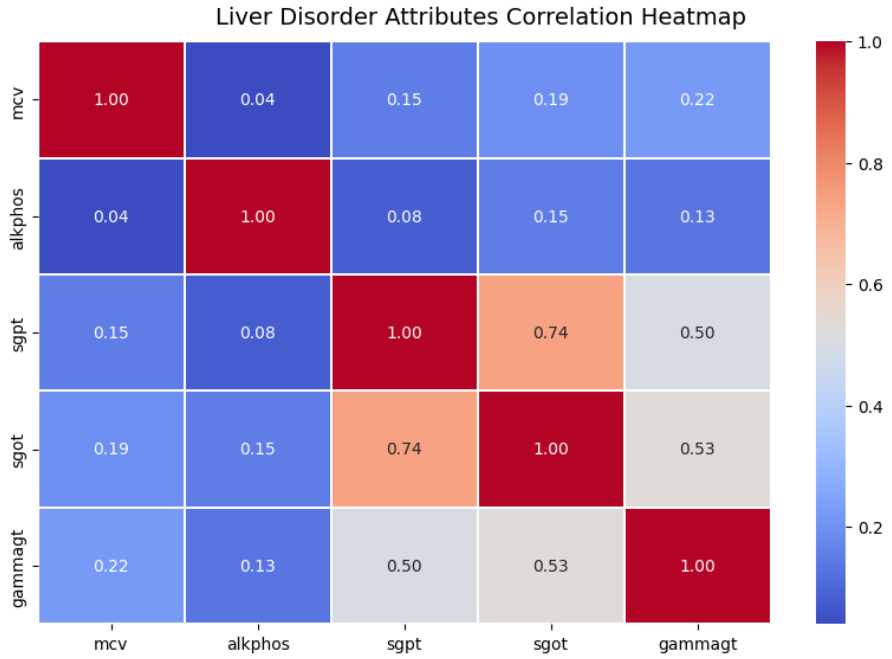
Preprocessing:

- We dropped the selector column as it was not meant for our use.
- As we were required to have a class label, we changed the last column “drinks” to drinks>=3 where it can take a binary value 0/1. which will be our class label.
- We performed Max-Min Normalization on our attribute columns to ensure there is no dominant attribute in the distance metric.

We first printed the values which can be NULL or missing in any of the samples. We did not obtain any such values hence no preprocessing was required. There was also no attribute with the same value for all classes hence we did not need to remove any attribute either.

```
>>Checking for null values in the dataframe:
mcv             0
alkphos         0
sgpt            0
sgot            0
gammagt         0
drinks >= 3     0
dtype: int64
```

The next graph that we make is the **heatmap**, this lets us know the correlation between any two attributes. This knowledge enables us to drop any columns that have high correlation with another column, so that the model doesn’t have any redundancy in it.



As you can see from the heat map that there are no two distinct columns whose correlation is greater than 0.9, therefore we avoid any redundancies in our Clustering Model

K-Means Clustering for a given K:

We ask the user to provide the value of k. We then determine the clustering performance without and with the class labels (ground truth).
Using the available ground truth we use the metrics

1. Homogeneity Score
2. Adjusted Rand Index
3. Normal Mutual Information Index
4. Fowlkes Mallows Score

Without using available ground truth we use the metrics

1. Silhouette Score
2. Calinski Harabasz score

We use the **lloyd method** for clustering which chooses k random rows from our dataset as our initial k centroids and performs repeated clustering. The algorithm stops when there is no cluster which undergoes a change from the previous clustering after an iteration or if the maximum number of iterations is exceeded. In the end, we return the cluster index of each sample.

From domain information, we can gather that when considering class labels, we will need at least two clusters (drinks >=3 has 2 values).

A sample run works as follows:

```
#######################################################
Enter the value of k which is the number of clusters (k>1): 2
Enter the method to be used for cluster centre initialization (lloyd or kmeans++): lloyd
#######################################################
```

Training the K Means Model with number of clusters 2 and initialization method lloyd as parameters...

Measuring Clustering Performance using available ground truth

homogeneity_score: 0.03842862110580102
adjusted_rand_score: 0.015557889562203782
normalized_mutual_info_score: 0.05008624123591687
fowlkes_mallows_score: 0.6321547380163345

Measuring Clustering Performance without using ground truth

silhouette_score: 0.4719650542379755
calinski_harabasz_score: 137.00107992657732

###

Conclusions:

Using Available Ground Truth:

We are likely to get relatively bad performance here as the dataset given to us did not have any labels and we had to transform a column to obtain class labels. This is a strong indicator of the fact that the data was likely to represent similar relations (like clusters) but not have very distinguishable classes in these clusters. Also, the size of the dataset is quite small so we do not have enough information to come up with very homogenous clusters as this usually needs large amounts of data.

- **Homogeneity Score**: The homogeneity score is quite less which indicates that clusters are a mix of different labels. This was expected as the dataset we were using was not meant for classification purposes so using available ground truth as one of the columns will imply correlations which do not necessarily exist.
- **Adjusted Rand Score**: This is close to 0 which is the case for random initialisations and hence matches with our initialisation technique.
- **Normalized Mutual Info Score**: Normalized Mutual Information (NMI) is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). Similar to the homogeneity score, this is close to 0 which is very less. This was also expected as the dataset was not meant for classification and hence the correlation will be low and because of that NMI should be low.
- **Fowlkes Mallows Score**: The Fowlkes-Mallows index (FMI) is defined as the geometric mean between the precision and recall. We have a decently high FM score which means that although our clusters are not very homogenous as seen through the Homogeneity Score and NMI, we are still able to get a sufficient number of true positives. This means that although the correlation between the clustering and the class labels is not very good, our distance metric is still able to get decent clusters with some relations between the attributes and the clusters.

Without Using Available Ground Truth:

We have obtained decent results without using available ground truths. This means that there was correlation between the clustering and the attributes if we do not take into consideration the explicit class labels. This makes sense as the dataset was meant to find relations and did not have class labels for classification.

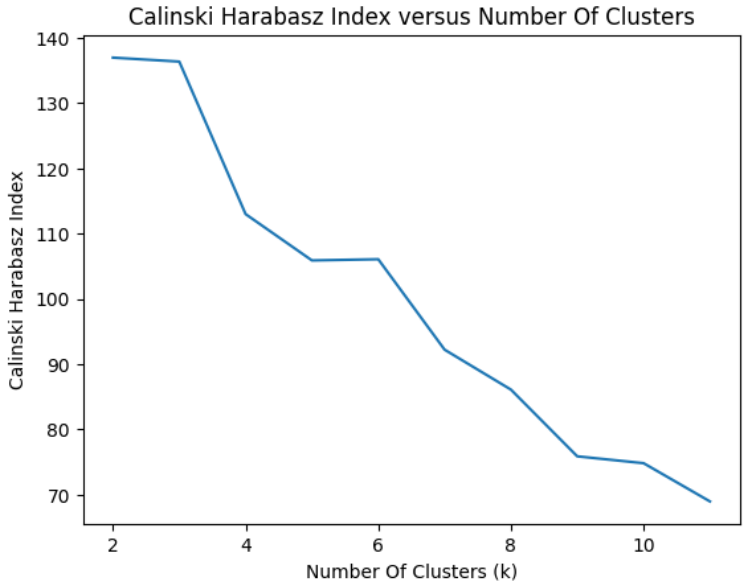
- **Silhouette Score**: The silhouette score is quite high which shows that clusters are well formed and easily distinguishable. This can be seen from the graphs obtained as well.
- **Calinski Harabasz Score**: The Calinski-Harabasz index also known as the Variance Ratio Criterion, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters, the higher the score , the better the performance. Our CH index is quite high which indicates good clustering results.

Obtaining most suitable K:

We vary the number of clusters between 2 to 11 to find the most suitable number of clusters. We use the following metrics to determine the clustering performance for a given k:

1. Silhouette Index
2. Calinski Harabasz Index
3. Wang's method of Cross Validation

Graphs obtained:

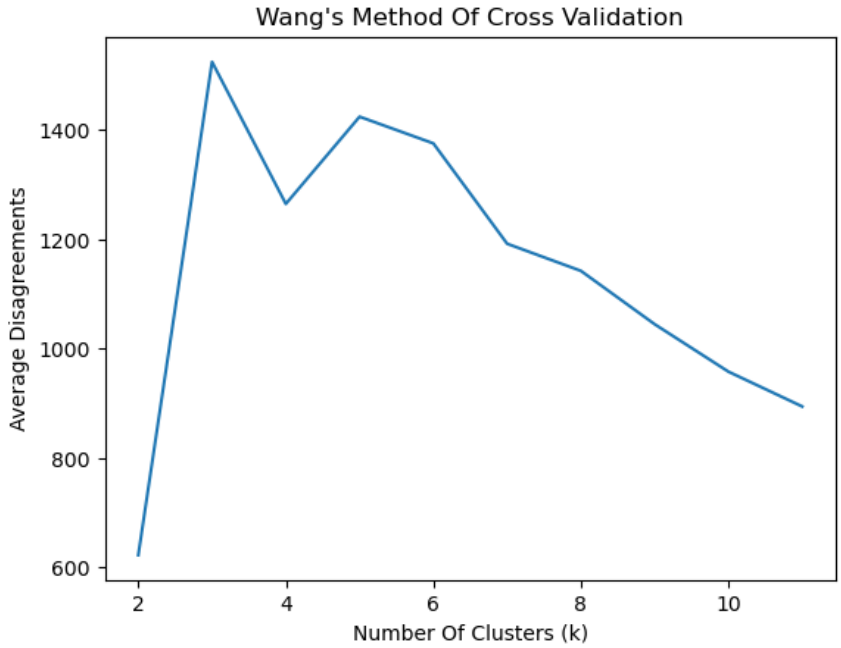


The CH index on an average decreases with increase in the number of clusters. This indicates that fewer clusters will give better performance.

Similar to CH index, Silhouette Index decreases with increase in the number of clusters.

In the case of **Wang's method of Cross Validation** we took the number of permutations of the dataset as user input. An example run has been shown below:

Perform Wang's Method Of Cross Validation Enter the value of c which is the number of permutations in Wang's Method Of Cross Validation(c>0): **20**



The number of disagreements in Wang's Method initially increases as we increase the number of clusters and then decreases but always remains larger than the initial minimum value. This decrease can be attributed to the fact that since the number of clusters is higher, the number of pairs in different clusters for both sets will be much more, so it is not a very good indicator of a better clustering hence can be taken out of consideration.

We choose the k which gives the best performance after taking into consideration all the three metrics above. All of the above indicate that k = 2 gives the best clustering performance. This is an expected result as from the domain knowledge we know that roughly the data can be segmented to two sections as there are two categories. Hence, 2 clusters should give the best performance.

Based on the above analysis , the most suitable K is: **2**

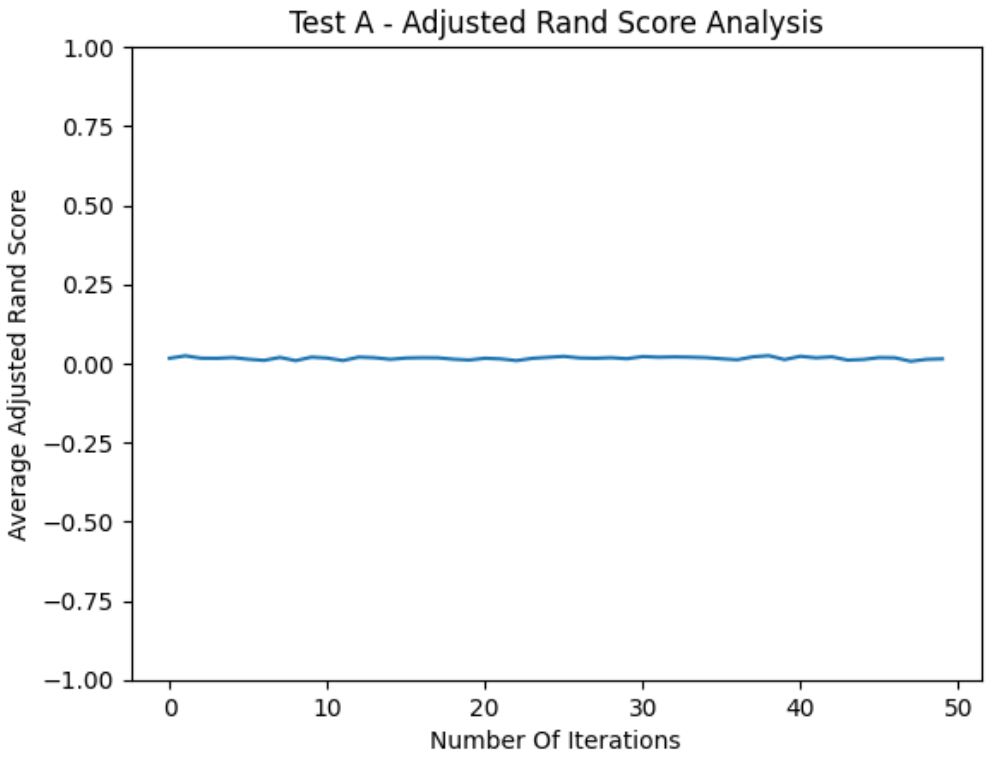
Test A:

In order to judge the stability of our clustering, we need to check if different random initializations yield similar or different results. For this purpose we perform **random testing** where we do random splitting of the dataset into train and test and choose K (the most optimal K we obtained above) random points as the cluster centers. We then perform k-means on this dataset with the given k using the **lloyd method.** We use the following metrics to analyse clustering performance:

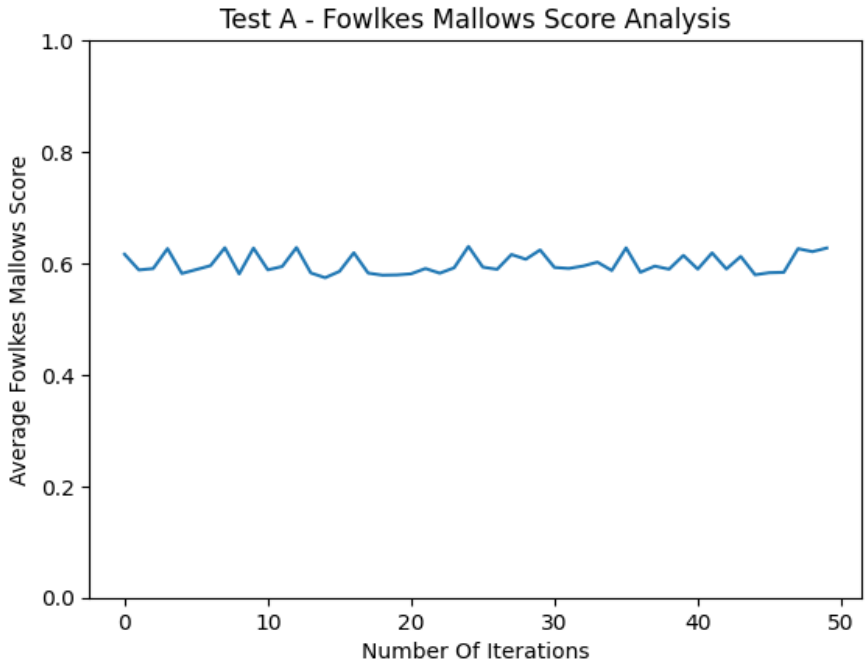
1. Average Adjusted Rand Score
2. Homogeneity Score
3. Normalized Mutual Info Score
4. Average Fowlkes Mallows Score

The metric is averaged over 50 results and the above process of random initialization is repeated 50 times.

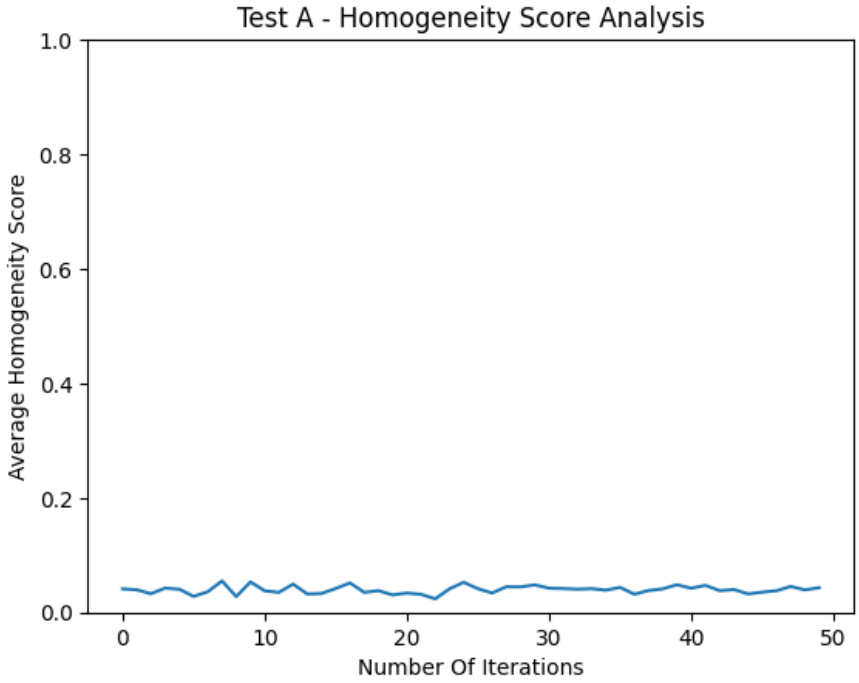
The following results are obtained:



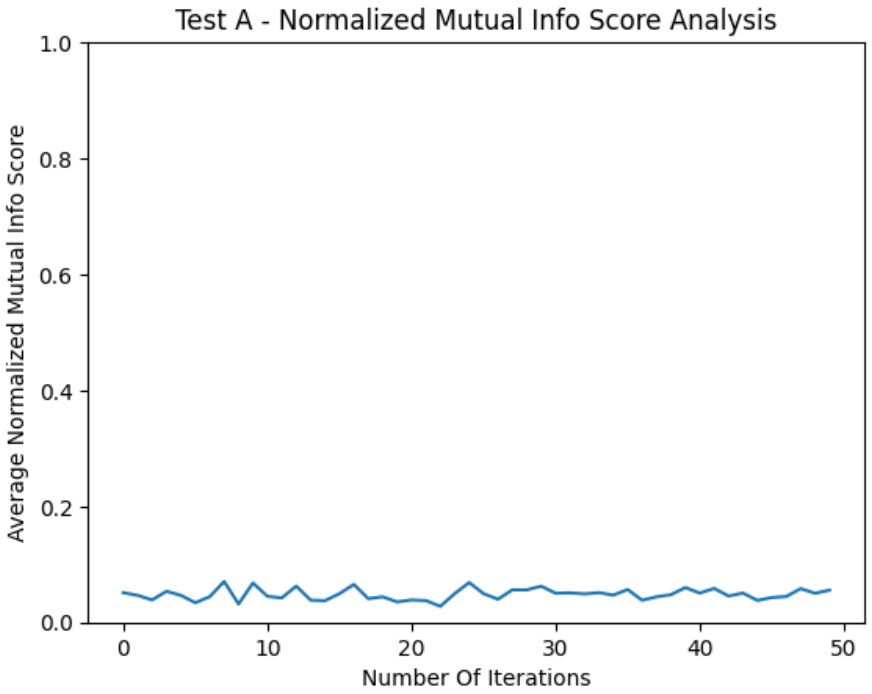
ARI in Test A:
mean: 0.01625526576127525
standard deviation: 0.004163427128441109



FM Index in Test A:
mean: 0.5990496545949999
standard deviation: 0.017461883963918904



Homogeneity Score in Test A:
mean: 0.0397484637706304
standard deviation: 0.006770327384982609



NMI Index in Test A:
mean: 0.04844832481486579
standard deviation: 0.009745712932377935

Conclusion:
As the optimal value of K was decided to be 2 which is the value we used in the first step to see clustering performance, and the method of initialisation is also the same (random initialization), the values obtained for different clustering measures are the same as the ones we initially obtained and can be explained by the same reasoning.

Stability: As we can see from the graphs, the values of the different measures do not vary with different random initialisations. Even after 50 random initialisations, we get similar values of the indices in each iteration as the standard deviation of each metric is very less (~0.00x) . This indicates that the clusters are not very sensitive to the initialisation at this stage and have obtained a good level of stability. The standard deviation of Fowlkes Mallows Score is still a little high so we will try to improve that next.

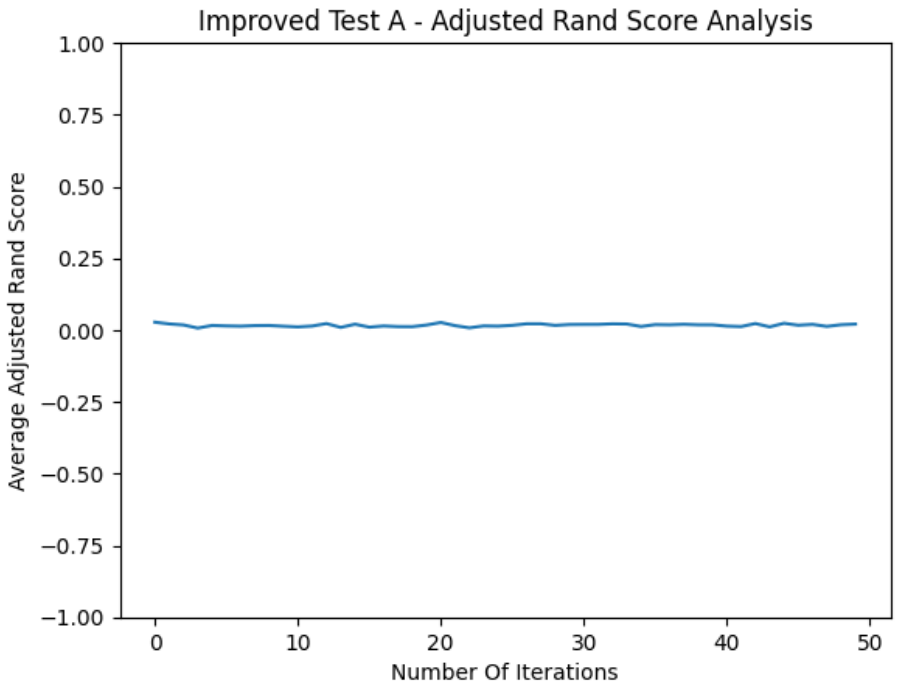
Test A with K-Means++ (Improved Test-A):

In order to obtain better clustering stability compared to the previous method, as K-Means clustering is highly dependent on the initialization of the clusters, we use a clustering method which follows a different method of initialisation for the initial k centers. This method is **k-means++** . We also aim at improving the already good stability results obtained in the previous step and potentially see an improvement in the clustering performance as well.
The **heuristic** it uses is as follows: It assigns the first cluster randomly and the following clusters are initialised with probability **inversely proportional to the distance** from the cluster centers currently initialised. Hence, the further the new cluster center from the initialised clusters, the higher the probability for it to be chosen as the next cluster center.

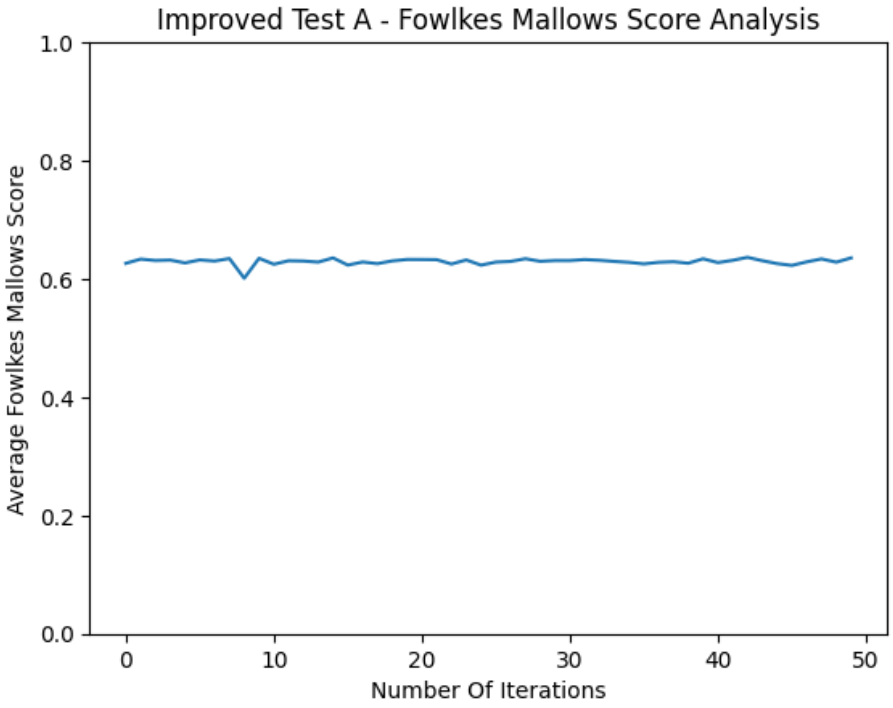
We perform **random testing** where we do random splitting of the dataset into train and test and choose K (the most optimal K we obtained above) random points as the cluster centers. We then perform k-means on this dataset with the given k using the **k-means++ method.** We use the following metrics to analyse clustering performance:

1. Average Adjusted Rand Score
2. Homogeneity Score
3. Normalized Mutual Info Score
4. Average Fowlkes Mallows Score

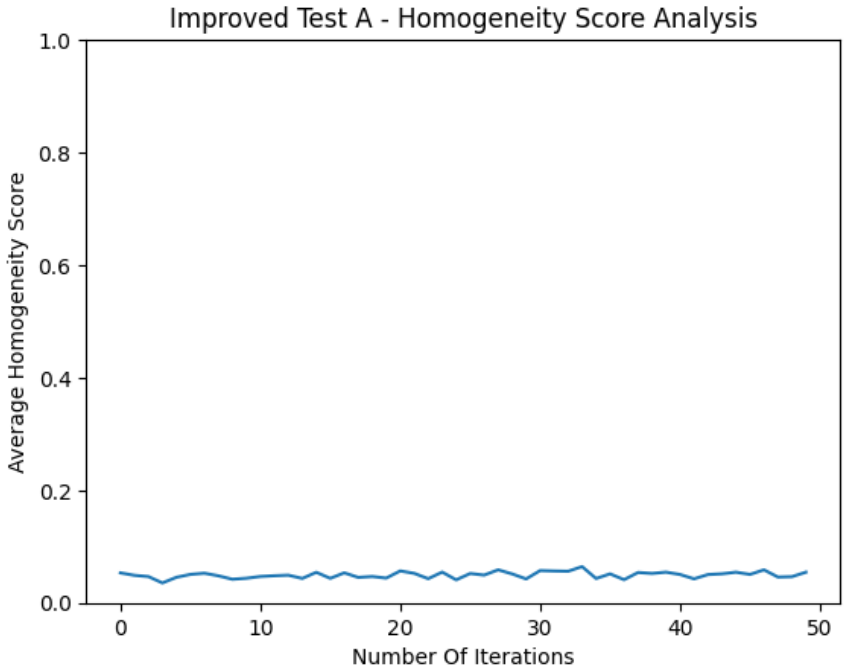
The metric is averaged over 50 results and the above process of random initialization is repeated 50 times.
The Results are as follows:



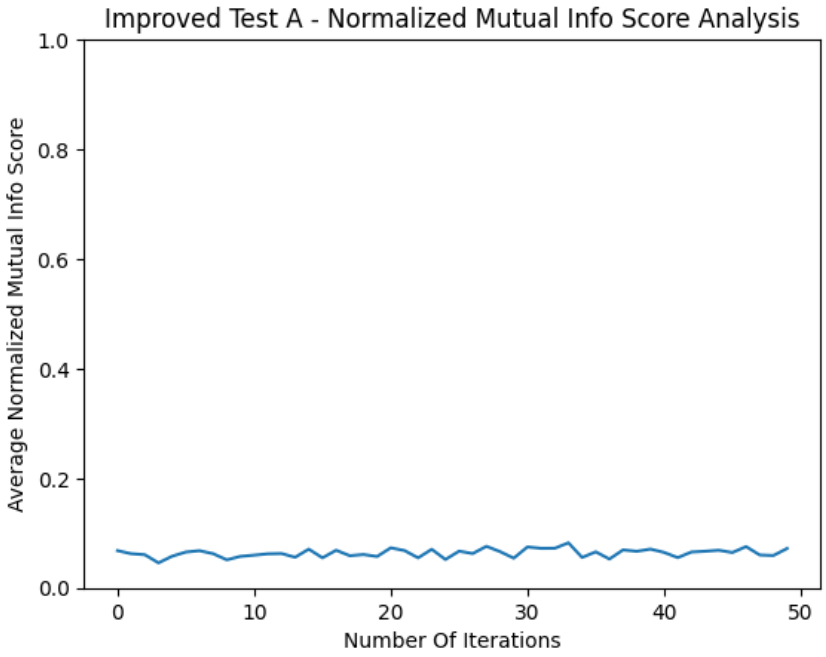
ARI in Improved Test A:
mean: 0.016776432032183722
standard deviation: 0.004575433536669217



FM Index in Improved Test A:
mean: 0.6294338470360922
standard deviation: 0.005255125419382237



Homogeneity Score in Improved Test A:
mean: 0.04956811973153139
standard deviation: 0.005685978105401102



NMI Index in Improved Test A:
mean: 0.04956811973153139
standard deviation: 0.005685978105401102

Conclusion:
As the optimal value of K was decided to be 2 which is the value we used in the first step to see clustering performance, and the method of initialisation is kmeans++, the values obtained for different clustering measures are more stable and have less dispersion.

Stability: As we can see from the graphs, the values of the different measures do not vary with different random initialisations. Even after 50 random initialisations, we get similar values of the indices in each iteration as the standard deviation is very low (~0.00x) . This indicates that the clusters have obtained a good level of stability.

Compared to **Test-A** , it is clear that we have achieved an improvement in stability as well as the performance. The mean of all the metrics has observed a slight increase which indicates a better clustering performance. The standard deviation has shown a stark decrease in case of Fowlkes Mallows measure and a slight decrease in case of the other measures which indicates that the stability of the clusters has increased. This is also apparent from the graphs.

Implementation Details:

Function: `lloyd_cluster_centre_initialization(self, X)` in class KMeans
Use : Initialises cluster centers using lloyd initialisation
Implementation: Chooses k random rows as the initial cluster centers where k is number of clusters

Function: `kmeans_plus_plus_cluster_centre_initialization` in class KMeans
Use : Initialises cluster centers using k-means++ initialisation
Implementation: Chooses the first cluster center randomly and for each successive cluster center iterates through all the rows which are not cluster centers yet and chooses the one which has maximum probability of being the next cluster center using the distance from existing clusters as the heuristic.

Function: `closest_centre(x, current_cluster_centres, distance_metric)` in class KMeans
Use : Finds the cluster closest to x using a given distance metric (like euclidean)
Implementation: Finds the distance of x from the existing cluster centers depending on the distance metric and returns the one with the least distance.

Function: `fit(self, X)` in class KMeans
Use : Performs the clustering for dataset X
Implementation: First we find the initial cluster centers depending on the metric that the object of class KMeans was initialised with. Then we repeatedly compute the clusters and update the centroids. At each iteration, every sample in the dataset belongs to a cluster indicated by its index (ranging from 0 to k-1). When we exceed the maximum number of iterations or no change in clusters is observed on recomputation the process is stopped.

Function: `fit_predict(self, X)` in class KMeans
Use : Returns the cluster indices for dataset X
Implementation: Calls fit(self,X) and returns the cluster indices for each sample.

Function: `predict(self, X)` in class KMeans
Use : Returns the cluster indices for some X
Implementation: Finds the cluster centroid closest to each row in X and assigns it the label of that cluster.

Function: `test_A(n_clusters, X, y, max_iter=50)`
Use : Checking for cluster stability
Implementation: It splits the data randomly into train and test and then forms the clusters using Lloyd initialisation. Finds out the scores of the different metrics. This is repeated 50 times and the average of the metric scores of each metric is returned.

Function: `improved_test_A(n_clusters,,y,max_iter=50,distance_metric="euclidean")`
Use : Checking for cluster stability
Implementation: It splits the data randomly into train and test and then forms the clusters using k-means++ initialisation. Finds out the scores of the different metrics. This is repeated 50 times and the average of the metric scores of each metric is returned.

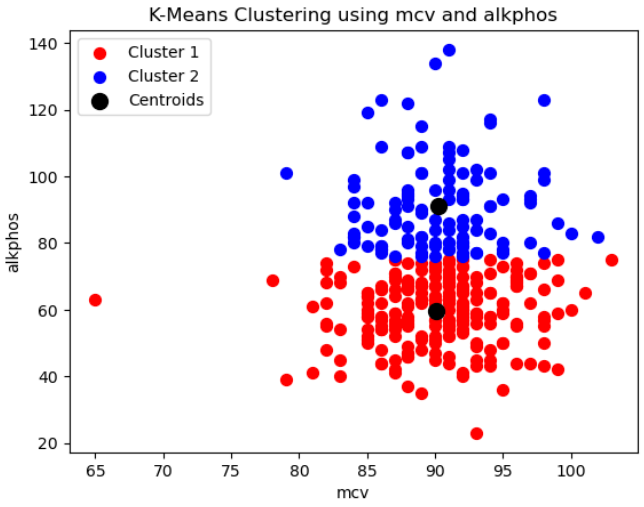
Function: `improved_test_A(n_clusters,,y,max_iter=50,distance_metric="euclidean")`
Use : Checking for cluster stability
Implementation: It splits the data randomly into train and test and then forms the clusters using k-means++ initialisation. Finds out the scores of the different metrics. This is repeated 50 times and the average of the metric scores of each metric is returned.

Function: `freqs(arr)`
Use : Obtaining unique values in arr
Implementation: Returns the unique values in arr and the count of each unique value

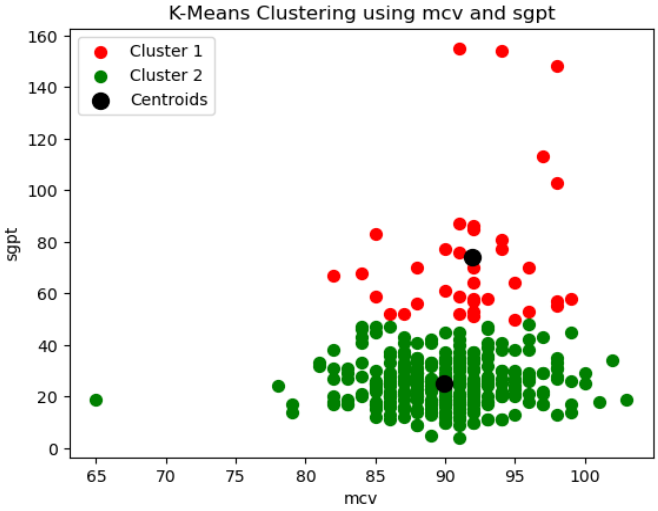
Function: `wangs_method_of_cross_validation(X, n_clusters=8, max_iter=300, distance_metric='euclidean', algorithm='lloyd', c=20)`
Use : Performing Wang's method of cross validation on a dataset X with given parameters such as number of clusters, number of permutations etc.
Implementation: In each iteration, we randomly permute the dataset and split it into 2 training and one test set. We perform k-means clustering on the two training sets. After this we use the test set to find out whether a pair of data belongs to the same cluster in the two different sets. A disagreement is defined as two rows belonging to the same cluster in one of the sets but not in the other set. It returns the average number of disagreements eventually.

Extra Analysis

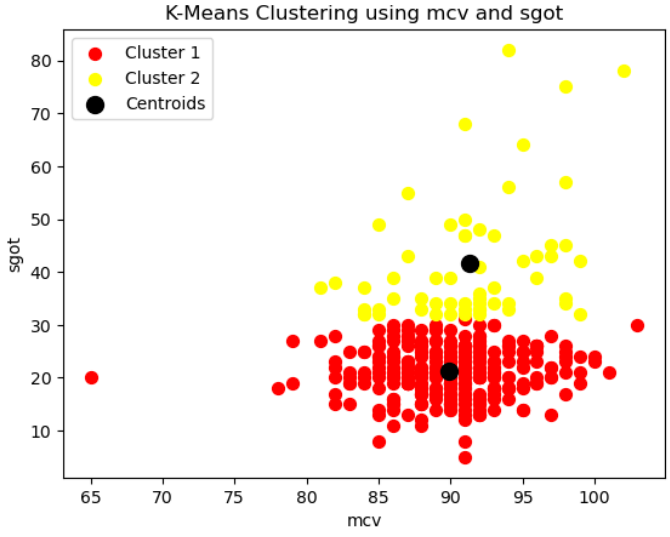
Normally, before applying K-Means clustering we do a principal component analysis which helps in visualization of the data as it reduces the dimensions. Since that is far beyond the scope of the assignment, we perform clustering taking into consideration two of the attributes at a time and note the performance, in order to see which features play a dominant role in the clustering. We use silhouette index as the metric for judging. The results are as follows:



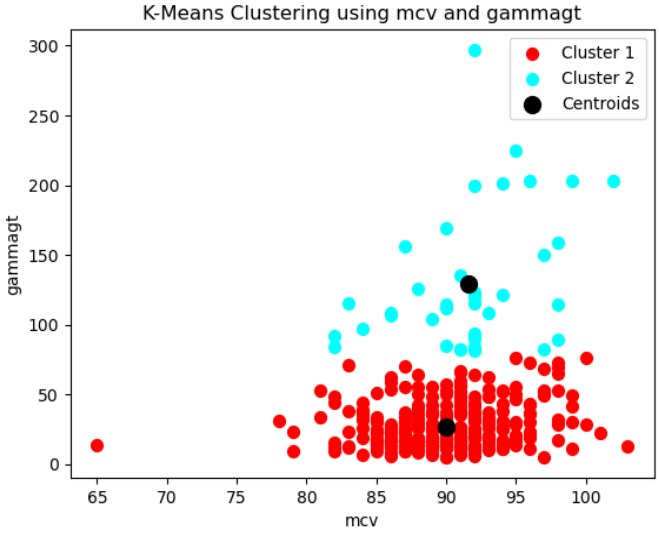
silhouette_score: 0.5299896943981719



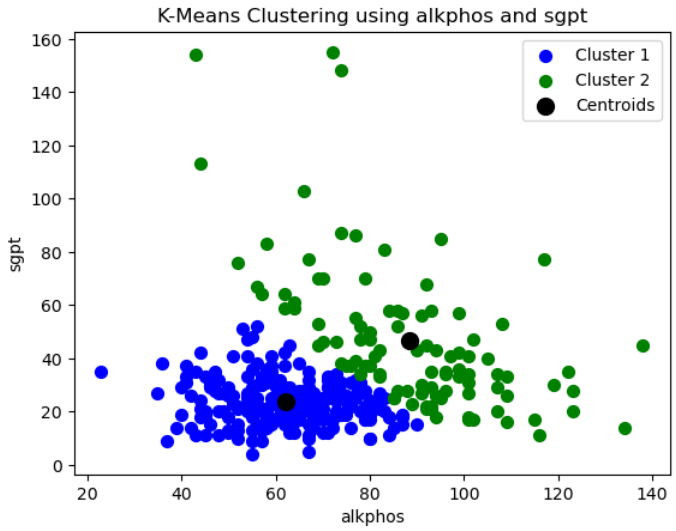
silhouette_score: 0.7018598051071286



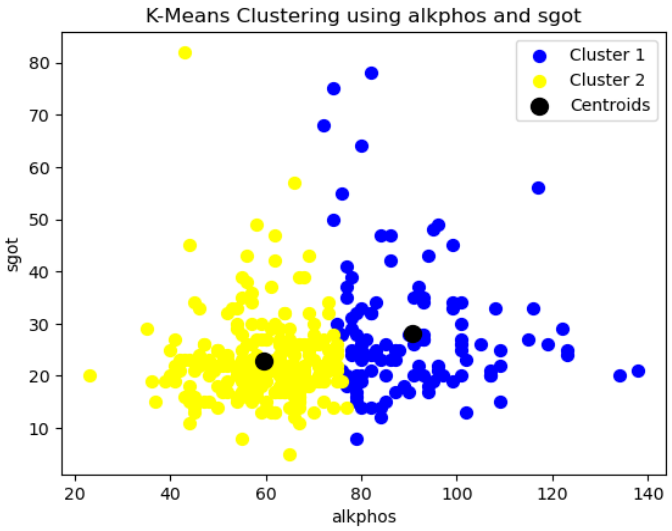
silhouette_score: 0.5662073431551423



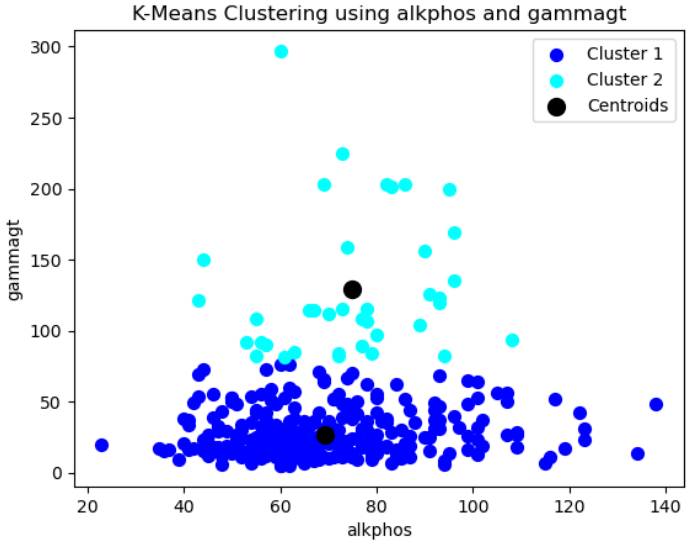
silhouette_score: 0.7564617643385075



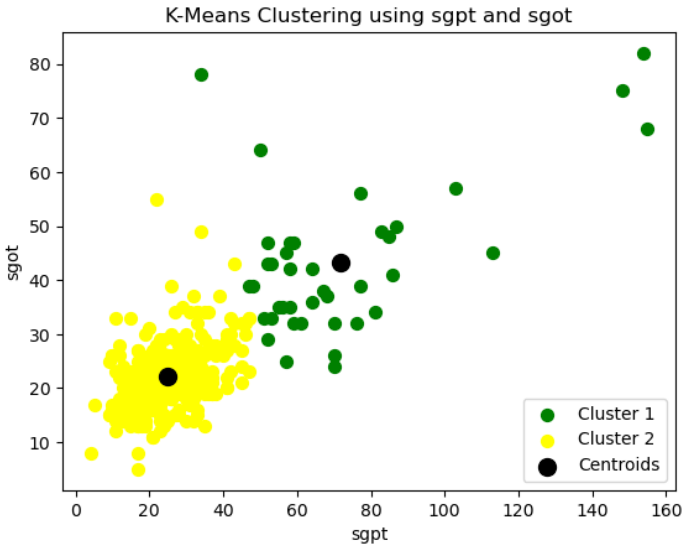
silhouette_score: 0.45301657139292206



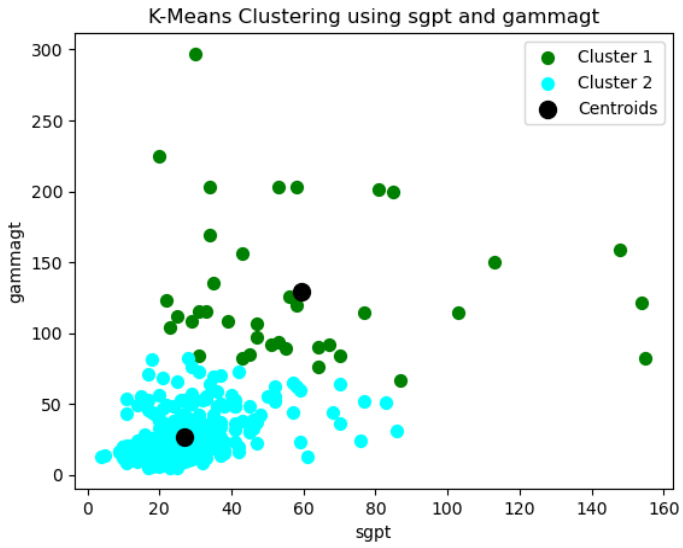
silhouette_score: 0.47624943756788896

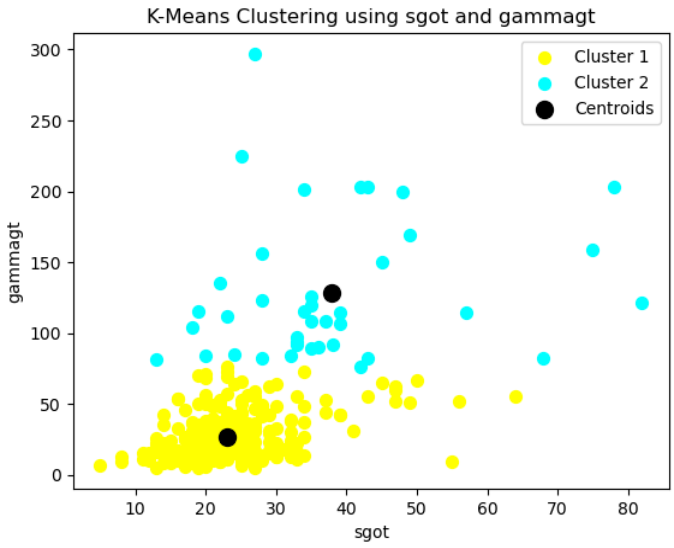


silhouette_score: 0.6671690837051212



silhouette_score: 0.6959992255614922





silhouette_score: 0.7203465968183828

silhouette_score: 0.7411796969630873

Conclusion:
Based on the above graphs, **mcv**, **gammagt** and **sgot** seem to be very determinant features for clustering.