MACHINE LEARNING - 6363

PROJECT 3
k-MEANS CLUSTERING

Problem

- Cluster the iris data into k clusters using the k-means clustering method
- Find the optimum number of clusters that divides the data perfectly based on their characteristics

Data

- A dataset named IRIS data is provided that contains attribute information about different flower species
- Link to the dataset: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
- Contains 150 samples:
    - Each data sample has 4 attributes:
        - Petal Length (cm)
        - Petal Width (cm)
        - Sepal Length (cm)
        - Sepal Width (cm)
    - Each sample is categorized into one of the following three species:
        - Iris-setosa
        - Iris-versicolor
        - Iris-virginica

Method

- This is an unsupervised learning task where we use a k-Means clustering algorithm that tries to group data points into clusters based on their characteristics. Hence, the labels from the iris dataset are ignored for the clustering algorithm

- k-Means is an iterative approach where we try to assign each data point to a particular cluster based on the distances between the data-point and the

centroids of each cluster

- The distance metric used in my algorithm is Euclidean distance (also known as Frobenius norm for matrices).

- The k-centroids at the start of the algorithm are randomly initialized from the data-points. After every iteration, the centroids are updated based on the new clusters formed.

- The iterations stop when the change in centroids starts becoming negligible (tolerance=0.0001 in my case) or the algorithm reaches a maximum limit (e.g. 500 iterations).

- The clusters formed at the end are visualized in a 2D-plot (two consecutive features from the data-points are selected for plotting)

Results

- To understand the results, I first ran the clustering algorithm on the dataset until it converged to particular centroids.

- I tried to compare the predicted clusters with the provided labels by numbering the clusters as 0, 1, 2, … and so on. But it turned out to be a bad way to see how the algorithm works because there was no way to know which cluster belonged to which class.

- To understand the formed clusters better, I visualized the data-points into 2D plots. Since, the data-points are 4-dimensional data **(i.e. X = F1, F2, F3, F4)**, I took three different 2D combinations of the features such as (F1, F2), (F2, F3), and (F3, F4). All the plots shown below are based on this combination of features.

1. K = 3 Clusters

- These plots are observed after the clustering algorithm converges.
- We can see that the first feature combination separates the data almost perfectly into 3 clusters.
- We can also observe that the red cluster has quite distinct characteristics than the other two clusters which is why it is separated well in all three feature combinations.

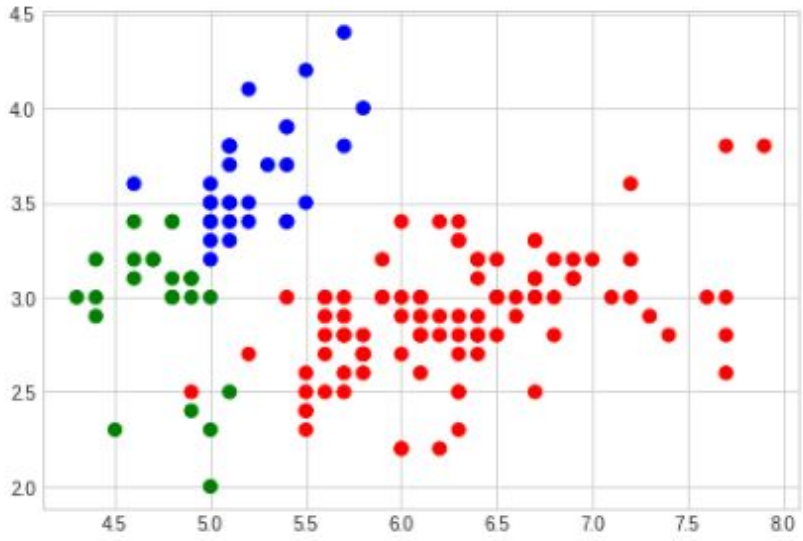

Fig 1: (F1, F2) feature-plot for 3-clusters

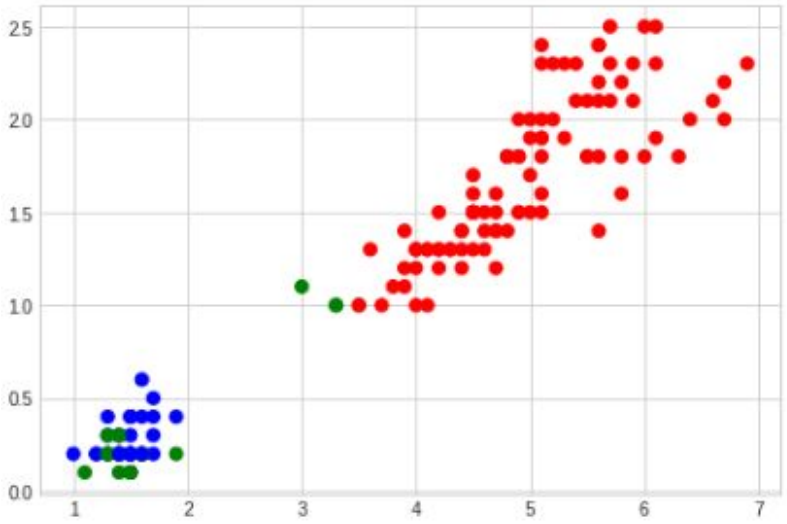

Fig 2: (F2, F3) feature-plot for 3-clusters

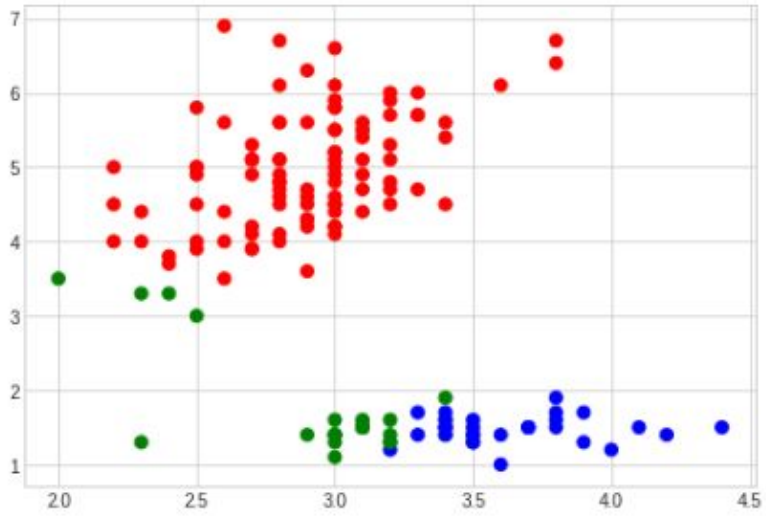

Fig 3: (F3, F4) feature-plot for 3-clusters

2. K = 5 Clusters

- When tried with 5 clusters, the data-points tend to separate well as seen in the plots below though some overlapping between the clusters can be seen from in figures 5 and 6.

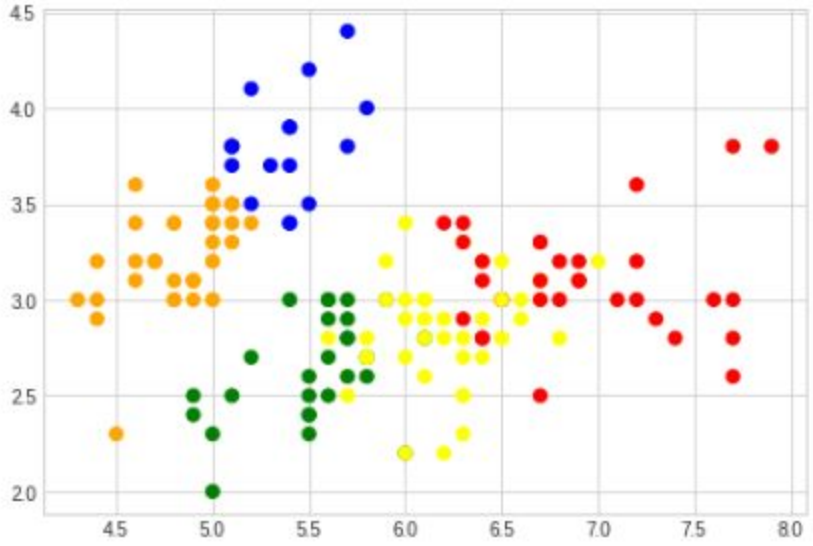

Fig 4: (F1, F2) feature-plot for 7-clusters

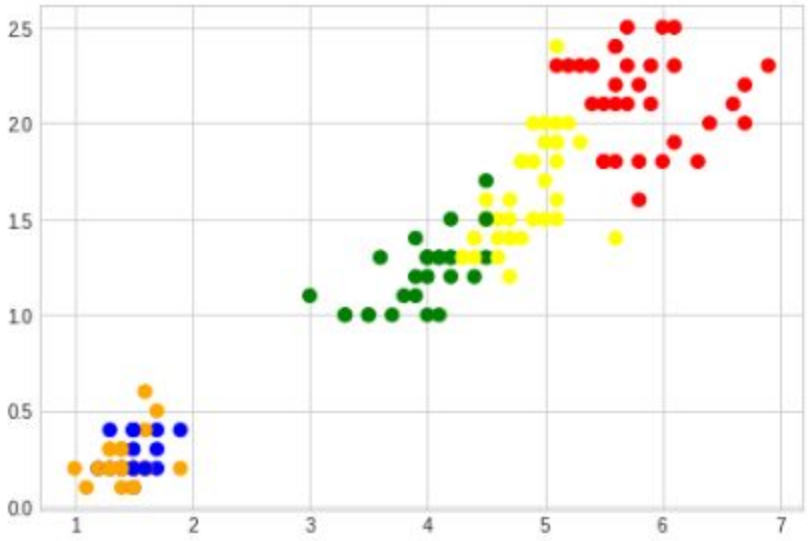

Fig 5: (F2, F3) feature-plot for 5-clusters

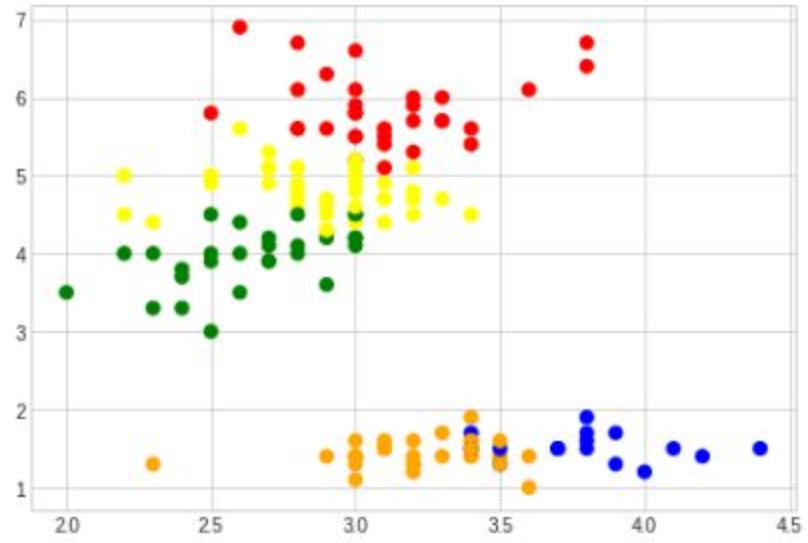

Fig 6: (F3, F4) feature-plot for 5-clusters

3. K = 7 Clusters

- On increasing the number of clusters to 7, we can already see that there is a lot of overlap between the clusters and the overlapping tends to get worse than when k=5.
- Also, the clustering plot clearly shows that one cluster (color=cyan) seems to have distinct characteristics than other clusters as seen below and above.

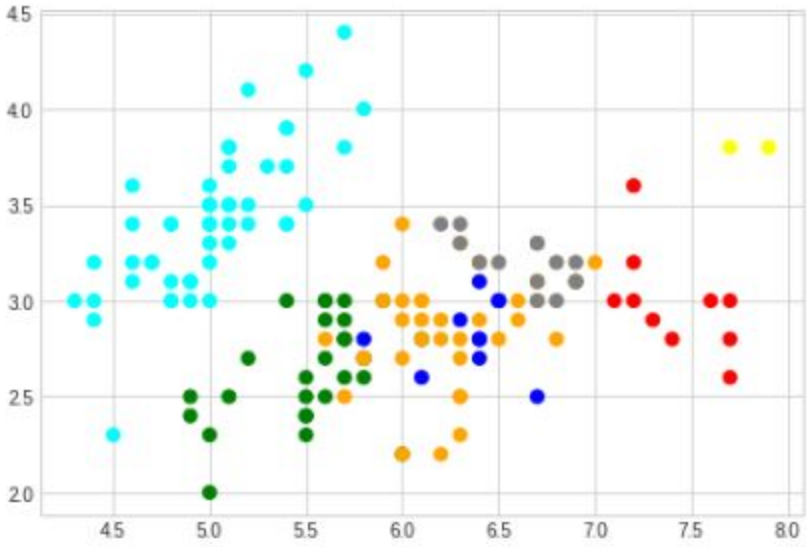

Fig 7: (F1, F2) feature-plot for 7-clusters

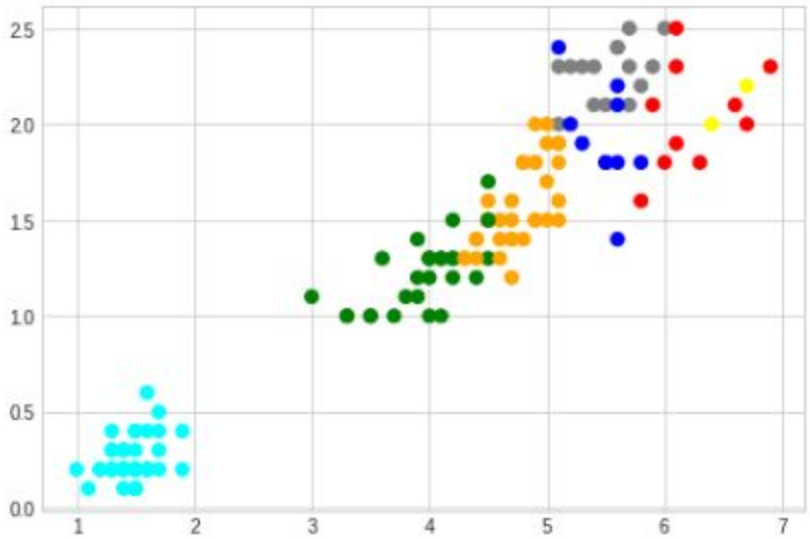

Fig 8: (F2, F3) feature-plot for 7-clusters

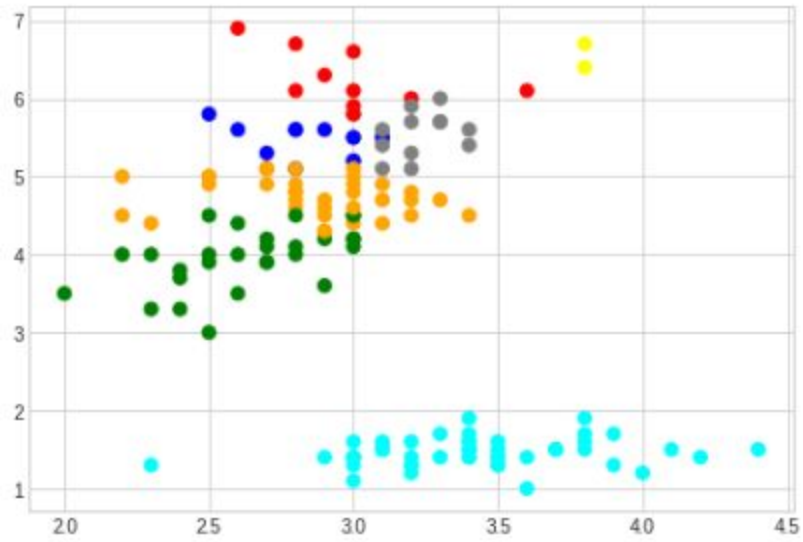

Fig 9: (F3, F4) feature-plot for 7-clusters

Conclusion

- From the plots, it is clear that one cluster is clearly distinctive from the other clusters. The problem comes when we try to separate the other data-points into separate clusters.
- When k=3, it tries to separate the other data-points into separate clusters pretty well.
- k=5 clusters also do a good job in separating the data-points, but we can see more overlapping of the clusters.
- k=7 has too many overlapping and does not do a good job in clustering the points. Hence, it is not worth it to go for a higher number of clusters.

- When compared with the labels provided, all three classes have 50 data-points.
- In k=3 clustering clusters, the clusters contained 50, 62, and 38 data points respectively. This means that 12 points out of 150 points fell in the wrong cluster.
- Also, the initialization of the centroids affected the quality of the clusters. So, we need to run this algorithm for a certain number of times to identify which centroids would be perfect for initialization.

In my opinion, both k=3 and k=5 can be used to cluster the dataset (if we do not consider the labels provided). But, if the labels are provided, we would go for 3 clusters because that would cluster the data pretty well as explained above.