CONTENTS

INTRODUCTION

For this assignment, I have implemented KNN and KMeans algorithms and calculated their various evaluation metrics. The dataset used for both of them was ‘Avila’, taken from the UCI Machine Learning Repository, and is suitable for both classification and clustering. The Avila data set has been extracted from 800 images of the 'Avila Bible', an XII century giant Latin copy of the Bible. The prediction task consists in associating each pattern to a copyist.

K-NEAREST NEIGHBORS (KNN)

KNN is a classification algorithm whose purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. It is based on feature similarity – a given data point is classified on the basis of how closely its features resemble the features of our training set. The algorithm outputs a discrete class in which it thinks an object belongs, with this class being the one most common among its k nearest neighbours.

IMPLEMENTATION

KNN is implemented in the following steps:

- The dataset is loaded and sorting into training and testing datasets and their respective labels/classes
- The Euclidean distance is calculated between each row of the test data and the training data
- These distances are sorted in ascending order and their indexes stored
- We only need to see k neighbours, so we take k number of indices and use it to retrieve the k nearest labels
- From these k nearest labels, the most common one is taken as the predicted class
- This is applied to every row in the testing dataset and an array of corresponding labels/classes is generated and returned

EVALUATION METRICS

For each class that the model has predicted, several scores are calculated that give a measure of the algorithm's performance. These scores are calculated using several values:

- **True Positive (TP)**: It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
- **True Negative (TN)**: It refers to the number of predictions where the classifier correctly predicts the negative class as negative.
- **False Positive (FP)**: It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
- **False Negative (FN)**: It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

ACCURACY

It gives you the overall accuracy of the model, meaning the fraction of the total samples (for a particular class) that were correctly classified by the algorithm.

Formula: (TP+TN)/(TP+TN+FP+FN)

PRECISION

It tells you what fraction of predictions as a positive class were actually positive. A perfect precision would be one.

Formula: TP/(TP+FP).

RECALL

It tells you what fraction of all positive samples of a class were correctly predicted as positive by the classifier.

Formula: TP/(TP+FN)

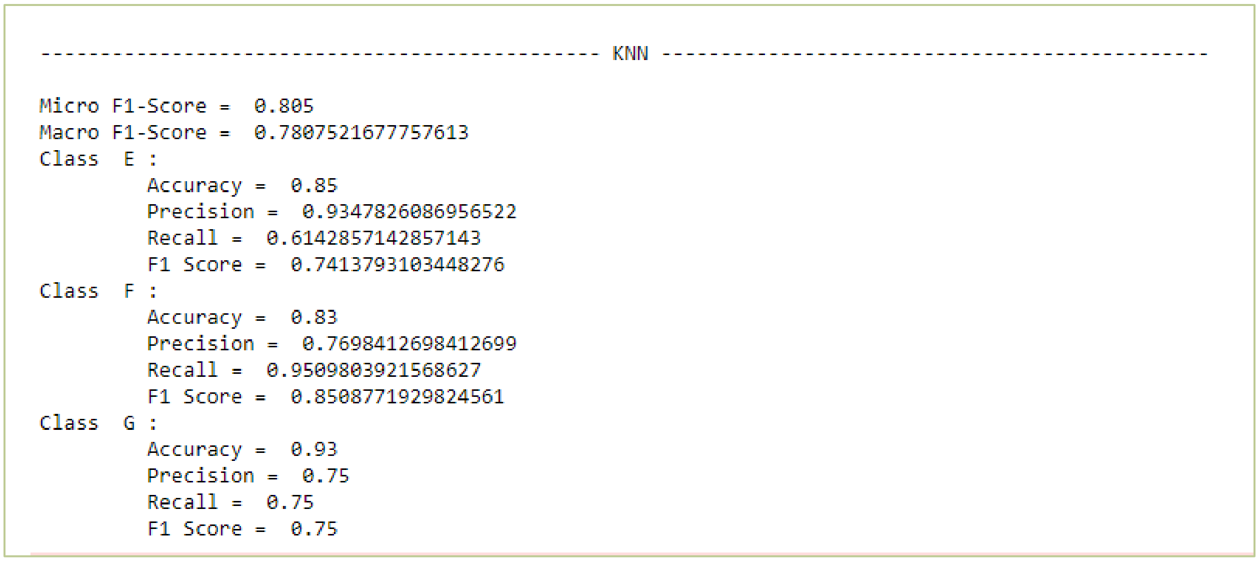
F1-SCORE

It combines precision and recall into a single measure. Mathematically it's the harmonic mean of precision and recall. An F1-score of 1 indicates a 100% accuracy for the algorithm.

Formula: 2 * (precision * recall / precision + recall)

The micro F1-score is also taken by considering the total TP, total FP and total FN of the model (instead of considering each class individually.)

The macro F1-score is taken by finding the unweighted mean of the F1-scores of all the classes.



```
----------------------------------------------- KNN -----------------------------------------------

Micro F1-Score =  0.805
Macro F1-Score =  0.7807521677757613
Class  E :
         Accuracy =  0.85
         Precision =  0.9347826086956522
         Recall =  0.6142857142857143
         F1 Score =  0.7413793103448276
Class  F :
         Accuracy =  0.83
         Precision =  0.7698412698412699
         Recall =  0.9509803921568627
         F1 Score =  0.8508771929824561
Class  G :
         Accuracy =  0.93
         Precision =  0.75
         Recall =  0.75
         F1 Score =  0.75
```

K-MEANS CLUSTERING

K-means is a centroid-based algorithm, in which datapoints are sorted into clusters on the basis of how close they are to a cluster's centroid. Every cluster is associated with a centroid. K-Means tries to minimize the sum of the distances between the points and the centroid of their respective cluster. Each point can only belong to one cluster.

IMPLEMENTATION

K-Means is implemented in the following steps:

- The number of clusters (k) is chosen
- k random points are selected from the data and chosen to be centroids
- All points are assigned to the closest cluster centroid (by finding the Euclidean distance between the point and the centroid)
- Centroids of the newly formed clusters are recalculated
- This process keeps repeating until maximum number of iterations is reached (or the centroids of newly formed clusters do not change)

DAVIES-BOULDIN INDEX

In classification, measuring the quality of the classification is very straightforward and simple. Quantifying clustering quality, however, is a little different. The performance metric being used here is the 'Davies Bouldin Index'. It is defined as a ratio between the cluster scatter and the clusters' separation; a lower value will mean that the clustering is better. The idea is that no cluster should be similar to another – so the best clustering algorithm would be minimizing the Davies-Bouldin index as clusters would be separate from each other.

$$DB = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i, \text{ where}$$

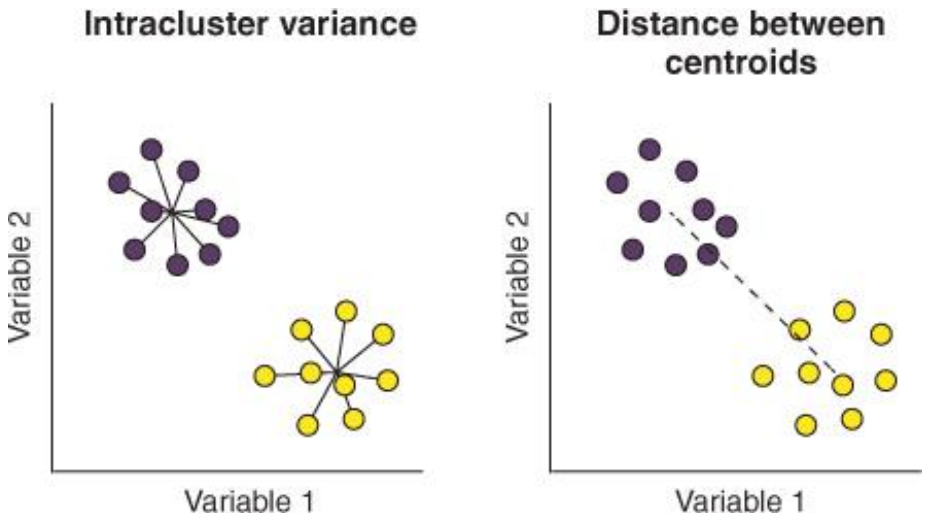
$$R_i = \max_{j=1...nc, i \neq j}\left(R_{ij}\right), \ i = 1...n_c$$

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

$$d_{ij} = d\left(v_i, v_j\right), \ s_i = \frac{1}{\|c_i\|}\sum_{x \in c_i} d\left(x, v_i\right)$$

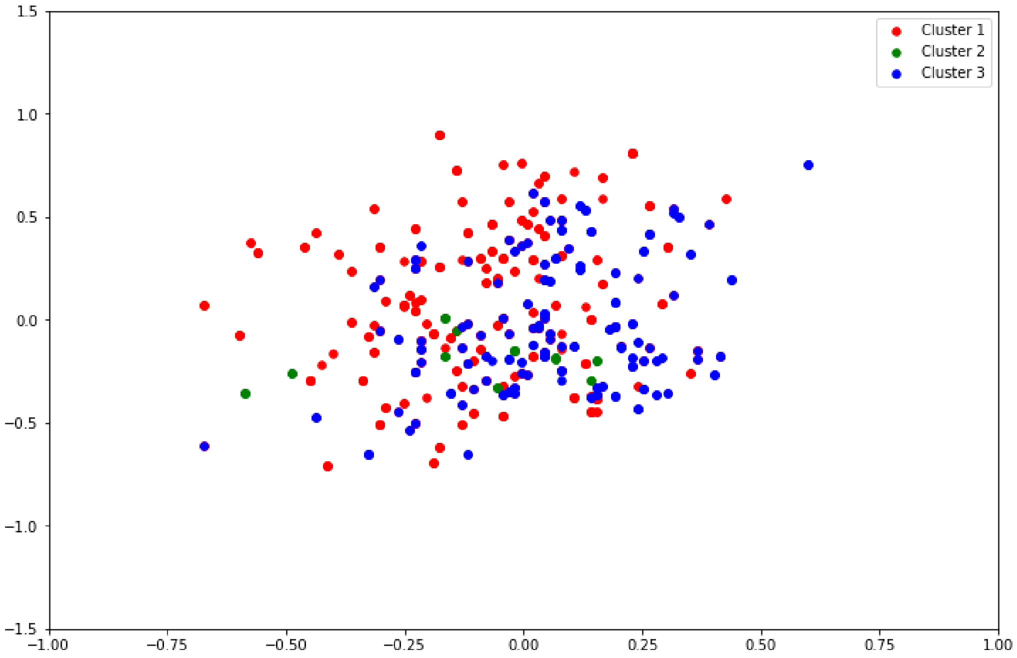
Where,

- d(x,y) is the Euclidean distance between x and y.
- c_i is the cluster i.
- v_i is the centroid of cluster c_i
- $\|c_i\|$ refers to the norm of c_i

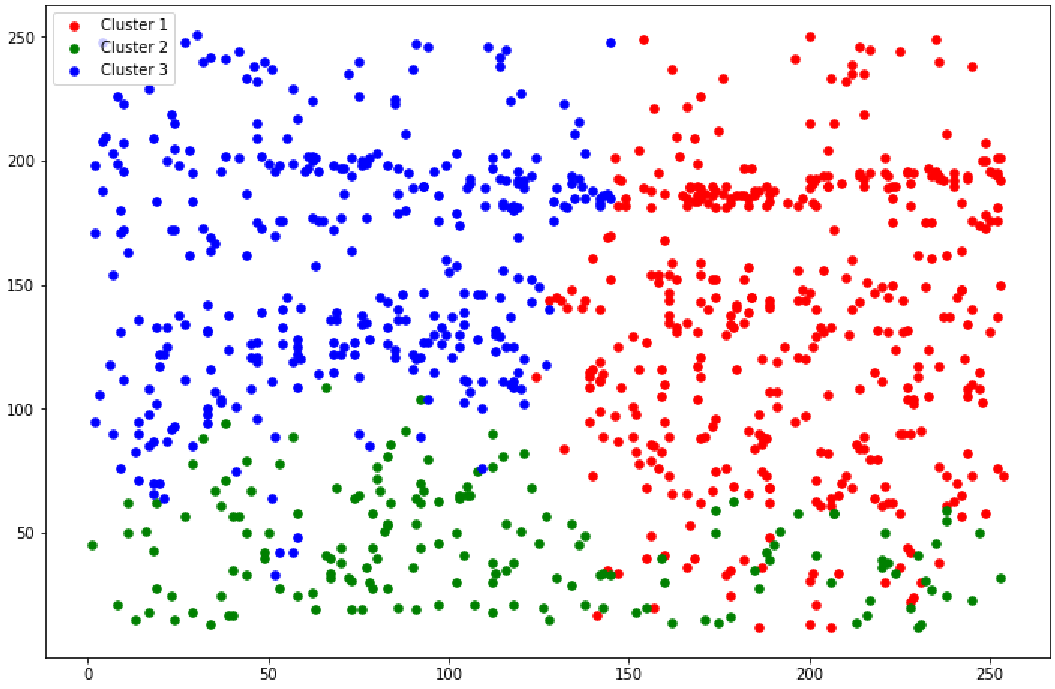


DATA VISUALIZATION AFTER CLUSTERING

DATASET 1 (AVILA)



DATASET 2 (SEGMENTATION)


Cluster 1
Cluster 2
Cluster 3