Lovemore Tenha
Project5 Report

In this report, we explore the performance of various clustering algorithms on 2 different datasets; the iris dataset and the faults.csv dataset from Kaggle. We use the clustering algorithms from the scikit-learn and scipy libraries in python. The clustering algorithms are Kmeans, Hierarchical (from scikit learn and scipy libraries) and DBSCAN. We also implemented the Kmeans elbow approach for determining the optimal number of clusters and the method of choosing the parameters eps and min_samples for DBSCAN. The table below shows the results after running the algorithms on the iris dataset. The clustering quality metric in the table measures the proportion of correctly classified points to the total number of points relative to ground truth; with 1 being the highest performance and 0 being the lowest performance.

Regressor	**Clustering Quality**	**Runtime**
Kmeans	0.89	0.0172
Hierarchical (Scipy)	0.89	0.0011
Hierarchical (Scikit-Learn)	0.89	0.0009
DBSCAN	0.67	0.0020

Performance and Runtime for each clustering algorithm on iris dataset

All clustering algorithms, except for DBSCAN, had relatively high clustering quality value at 0.89. DBSCAN has a clustering quality of 0.67. The reason for this lower value is due to DBSCAN labeling some of the points as outliers. Both Hierarchical clustering algorithms had the fastest runtime, and kmeans was the slowest.

The following results were obtained after applying the clustering algorithms on the faults.csv dataset.

Regressor	**Runtime**
Kmeans	0.7597
Hierarchical (Scipy)	0.1261
Hierarchical (Scikit-Learn)	0.1084
DBSCAN	0.0302

The faults.csv has 34 attributes and the last 7 are class labels. Only the first 27 were used with the clustering algorithms. The table above shows the running time of the algorithms on the dataset. DBSCAN had the fastest running time and Kmeans has the slowest time. The Elbow approach for Kmeans shows that the optimal number of clusters is 4 in this dataset.