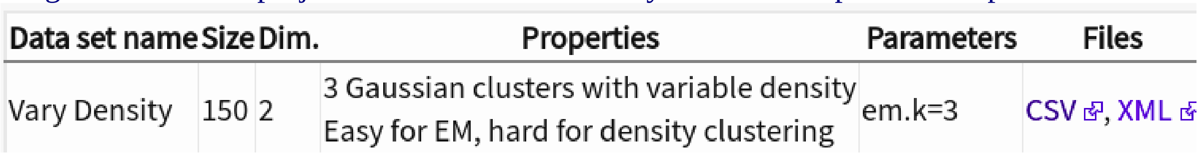
HW2 - REPORT

Canberk ARICI – 171044062

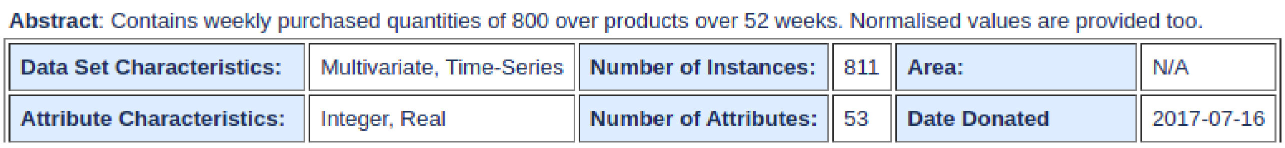
The datasets which I used for clustering are given below with descriptions.

2D Dataset :
https://github.com/elki-project/elki/blob/master/data/synthetic/ABC-publication/pov.csv



Data set name	Size	Dim.	Properties	Parameters	Files
Vary Density	150	2	3 Gaussian clusters with variable density Easy for EM, hard for density clustering	em.k=3	CSV, XML

51D Dataset:
https://archive.ics.uci.edu/ml/datasets/Sales_Transactions_Dataset_Weekly



Abstract: Contains weekly purchased quantities of 800 over products over 52 weeks. Normalised values are provided too.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	811	**Area:**	N/A
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	53	**Date Donated**	2017-07-16

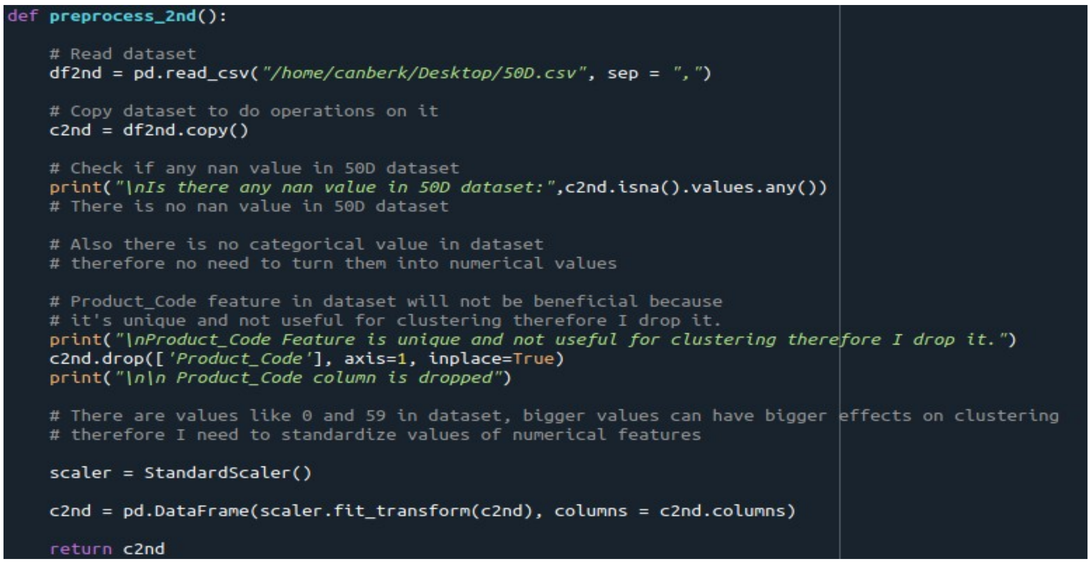
PREPROCESS

DS1:

DS1 does not include any nan value therefore no need to drop or fill any value.
Since all values are between 0 and 1, no need to normalization.
Also there is no categorical value in dataset therefore no need to turn them into numerical values.

DS2:

DS1 does not include any nan value therefore no need to drop or fill any value.
Also there is no categorical value in dataset therefore no need to turn them into numerical values.
Product_Code feature in dataset will not be beneficial because it's unique and not useful for clustering therefore I drop it.
There are values like 0 and 59 in dataset, bigger values can have bigger effects on clustering therefore I need to standardize values of numerical features.



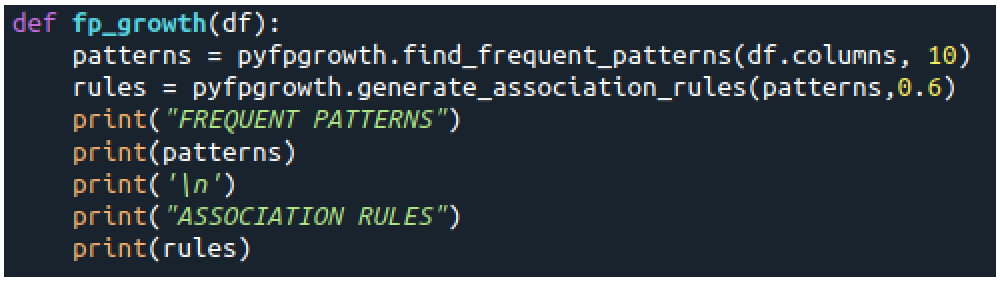
```
def preprocess_2nd():

    # Read dataset
    df2nd = pd.read_csv("/home/canberk/Desktop/50D.csv", sep = ",")

    # Copy dataset to do operations on it
    c2nd = df2nd.copy()

    # Check if any nan value in 50D dataset
    print("\nIs there any nan value in 50D dataset:",c2nd.isna().values.any())
    # There is no nan value in 50D dataset

    # Also there is no categorical value in dataset
    # therefore no need to turn them into numerical values

    # Product_Code feature in dataset will not be beneficial because
    # it's unique and not useful for clustering therefore I drop it.
    print("\nProduct_Code Feature is unique and not useful for clustering therefore I drop it.")
    c2nd.drop(['Product_Code'], axis=1, inplace=True)
    print("\n\n Product_Code column is dropped")

    # There are values like 0 and 59 in dataset, bigger values can have bigger effects on clustering
    # therefore I need to standardize values of numerical features

    scaler = StandardScaler()

    c2nd = pd.DataFrame(scaler.fit_transform(c2nd), columns = c2nd.columns)

    return c2nd
```

SOLUTIONS

1. Find clusters using Frequent Pattern Growth, k-means, DB scan and Chameleon clustering techniques. You may use data mining tools to find clusters.
2. Present the clusters of DS1 for each technique using graphics.

Frequent Pattern Growth

I tried to implement FP Growth, also made search about how can I use FP Growth for clustering but I couldn't implement more than the code below. I found frequent patterns and association rules and I couldn't find a way to visualize it, also I couldn't find Silhouette Score of FP Growth.



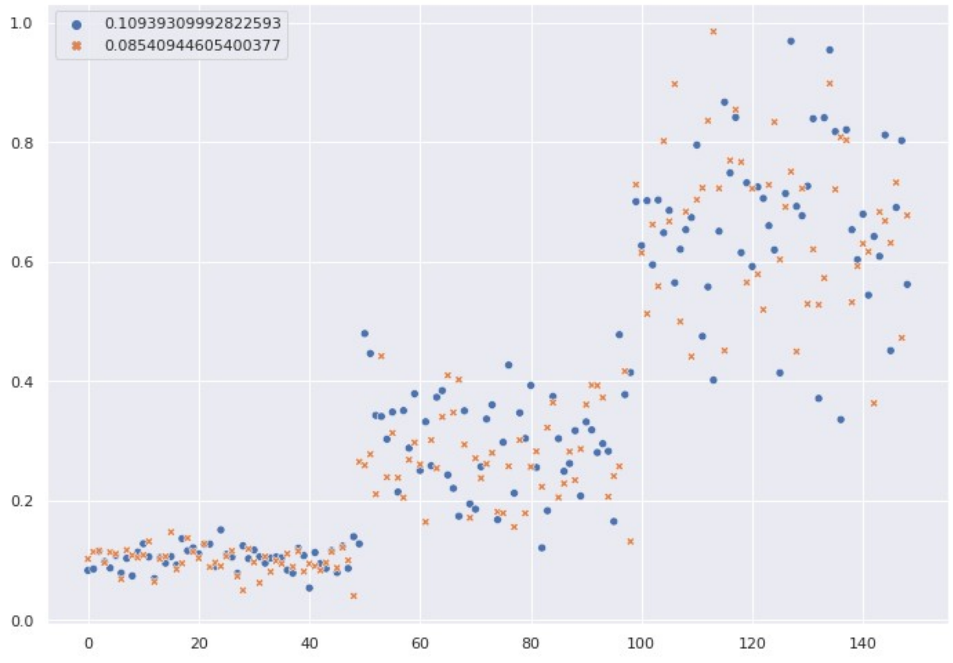
```python
def fp_growth(df):
    patterns = pyfpgrowth.find_frequent_patterns(df.columns, 10)
    rules = pyfpgrowth.generate_association_rules(patterns,0.6)
    print("FREQUENT PATTERNS")
    print(patterns)
    print('\n')
    print("ASSOCIATION RULES")
    print(rules)
```

KMeans

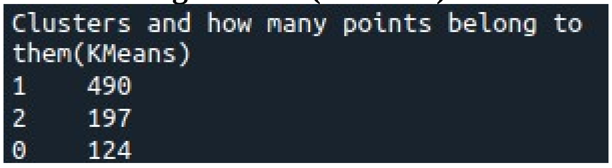
Firstly I check dimension of given dataframe then if it's less than 20, I plot graph of inertias in order to find optimal k value. Graphic gives optimal k value as 3. Then I give 3 to n_clusters parameter and use fit_predict() method by giving df to find clusters. After all, I plot clusters.

If dimension of dataframe is bigger than 20, I find optimal k value with the same way and it's 3 again. Since visualization of 2nd dataset is not wanted, I just print clusters and how many points belong to them.

Cluster Graphic of 2D Dataset:



Clusters and Number of Points Belong to Them(KMeans):



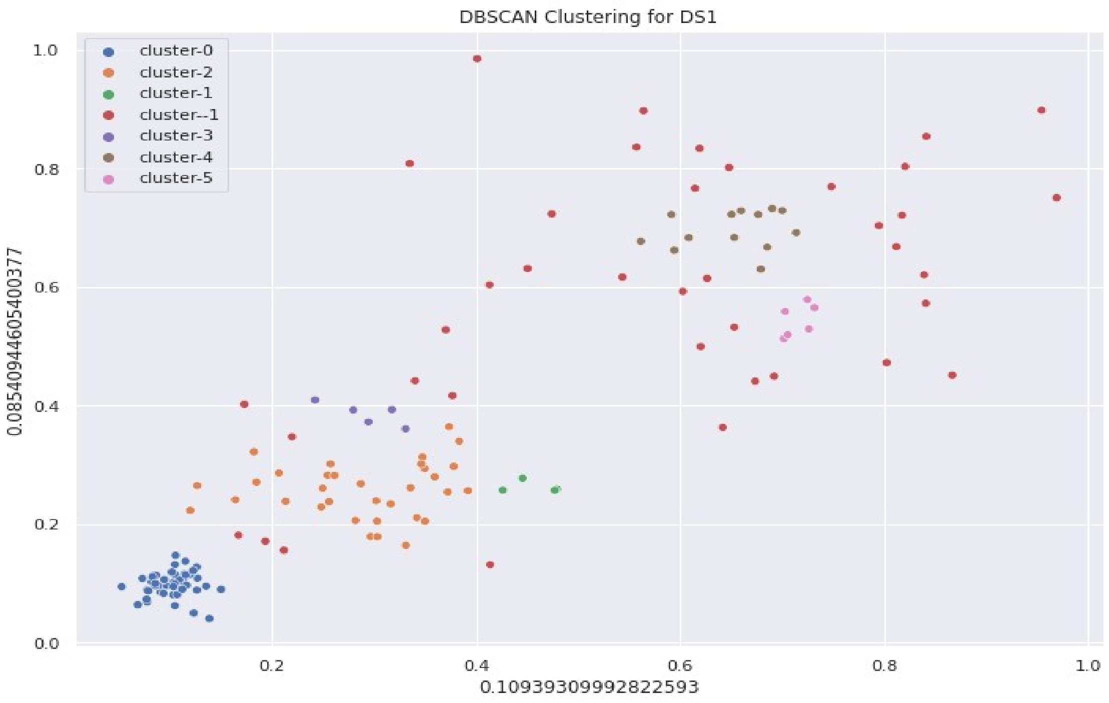
```
Clusters and how many points belong to
them(KMeans)
1    490
2    197
0    124
```

DBSCAN

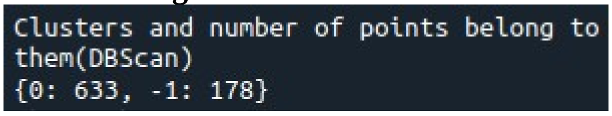
Firstly I check dimension of given dataframe then if it's less than 20, I use DBSCAN's default min_samples value which is 4 and then I find optimal epsilon value by using NearestNeighbors. It gives distances between each point and elbow point on its graph gives optimal epsilon value.
I find optimal epsilon value in find_optimal_eps function, related article which I benefited about finding optimal eps value for DBSCAN is on this link:
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf
After all, I use fit_predict() method by giving df and get labels. Then I plot graphic for 2D dataset.

If dimension of given dataframe is greater than 20, I use 2*df.shape[1] as min_samples value and I find optimal eps value as the same way. Then I find labels and print clusters and number of points belong to them because visualization for 50D dataset is not wanted.

Cluster Graphic of 2D Dataset:



Clusters and Number of Points Belong to Them:



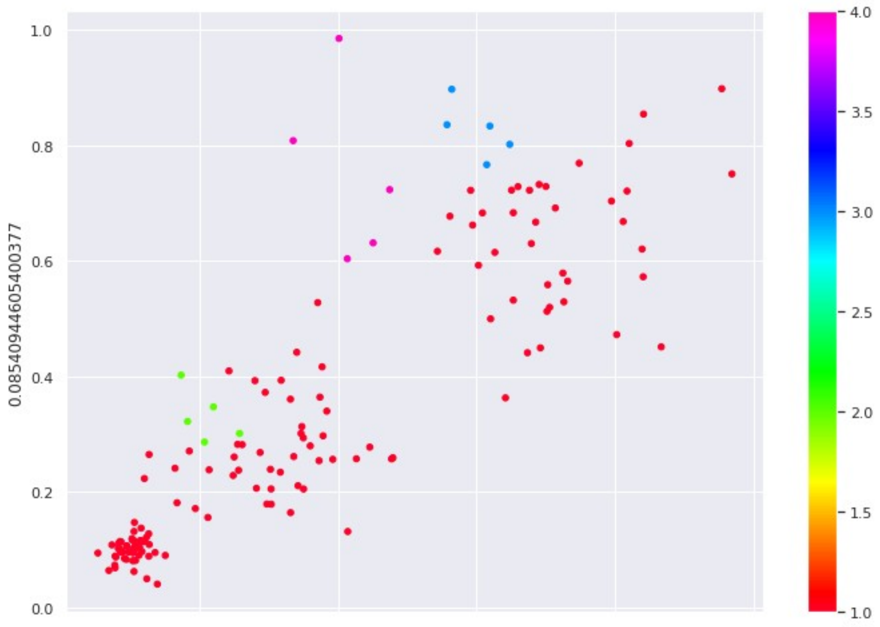
```
Clusters and number of points belong to
them(DBScan)
{0: 633, -1: 178}
```

Chameleon

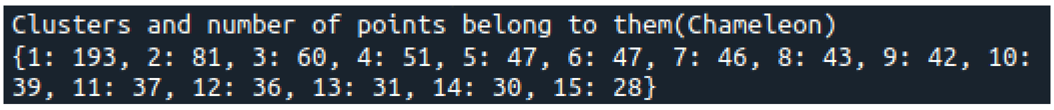
Firstly I check dimension of given dataframe then if it's less than 20, then I select k as 3 and the other variables randomly. After all I plot graphic of cluster.

If dimension of dataframe is greater than 20, I pick values randomly and since visualization for 2nd dataset is not wanted, I print clusters and number of points belong to them.

Cluster Graphic of 2D Dataset:



Cluster Graphic of 51D Dataset:

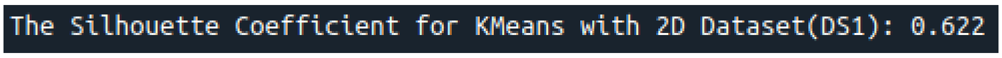


```
Clusters and number of points belong to them(Chameleon)
{1: 193, 2: 81, 3: 60, 4: 51, 5: 47, 6: 47, 7: 46, 8: 43, 9: 42, 10:
39, 11: 37, 12: 36, 13: 31, 14: 30, 15: 28}
```

3. Calculate silhouette coefficient for each clustering technique. Compare and interpret the silhouette score with the extracted clusters.

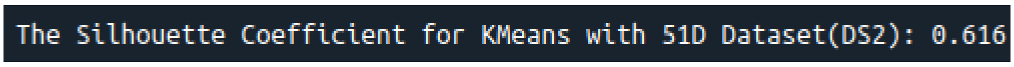
KMeans

DS1(2D Dataset):



```
The Silhouette Coefficient for KMeans with 2D Dataset(DS1): 0.622
```

DS2(51D Dataset):

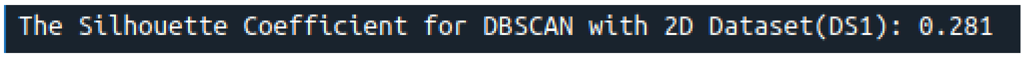


```
The Silhouette Coefficient for KMeans with 51D Dataset(DS2): 0.616
```

Silhouette Coefficient of DS1 > Silhouette Coefficient of DS2
It can be interpreted as clustering is better in small dataset(DS1) than big dataset(DS2)

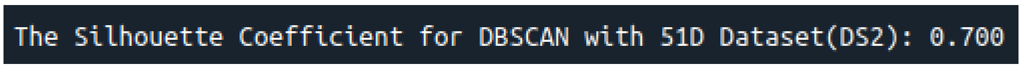
DBSCAN

DS1(2D Dataset):



```
The Silhouette Coefficient for DBSCAN with 2D Dataset(DS1): 0.281
```

DS2(51D Dataset):

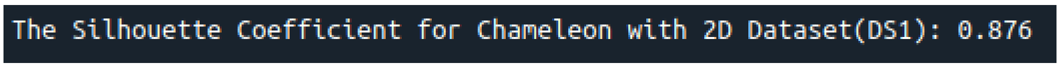


```
The Silhouette Coefficient for DBSCAN with 51D Dataset(DS2): 0.700
```

Silhouette Coefficient of DS2 > Silhouette Coefficient of DS1
It can be interpreted as clustering is better in big dataset(DS2) than small dataset(DS1)

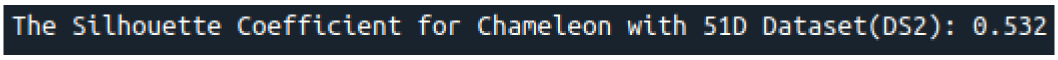
Chameleon

DS1(2D Dataset):



```
The Silhouette Coefficient for Chameleon with 2D Dataset(DS1): 0.876
```

DS2(51D Dataset):



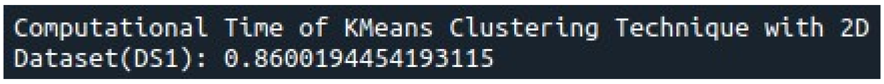
```
The Silhouette Coefficient for Chameleon with 51D Dataset(DS2): 0.532
```

Silhouette Coefficient of DS1 > Silhouette Coefficient of DS2
It can be interpreted as clustering is better in small dataset(DS1) than big dataset(DS2)

4. Present computational time and time complexity of each clustering model.

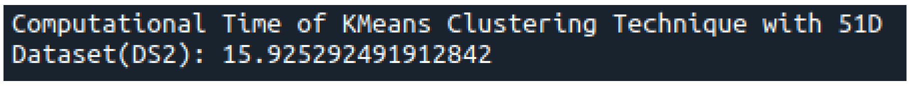
KMeans

Computational time for DS1:



```
Computational Time of KMeans Clustering Technique with 2D
Dataset(DS1): 0.8600194454193115
```

Computational time for DS2:

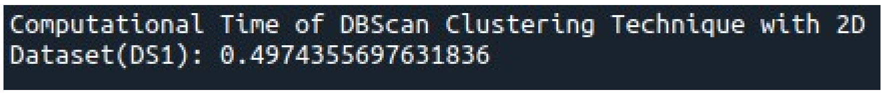


```
Computational Time of KMeans Clustering Technique with 51D
Dataset(DS2): 15.925292491912842
```

Time complexity:
Time Complexity of KMeans is O(n)

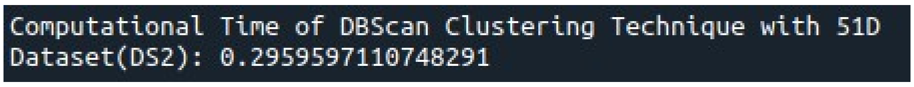
DBSCAN

Computational time for DS1:



```
Computational Time of DBScan Clustering Technique with 2D
Dataset(DS1): 0.4974355697631836
```

Computational time for DS2:

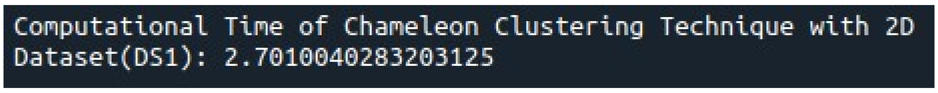


```
Computational Time of DBScan Clustering Technique with 51D
Dataset(DS2): 0.2959597110748291
```

Time complexity:
Time Complexity of DBSCAN is O(N logN)

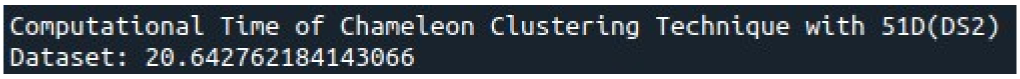
Chameleon

Computational time for DS1:



```
Computational Time of Chameleon Clustering Technique with 2D
Dataset(DS1): 2.7010040283203125
```

Computational time for DS2:



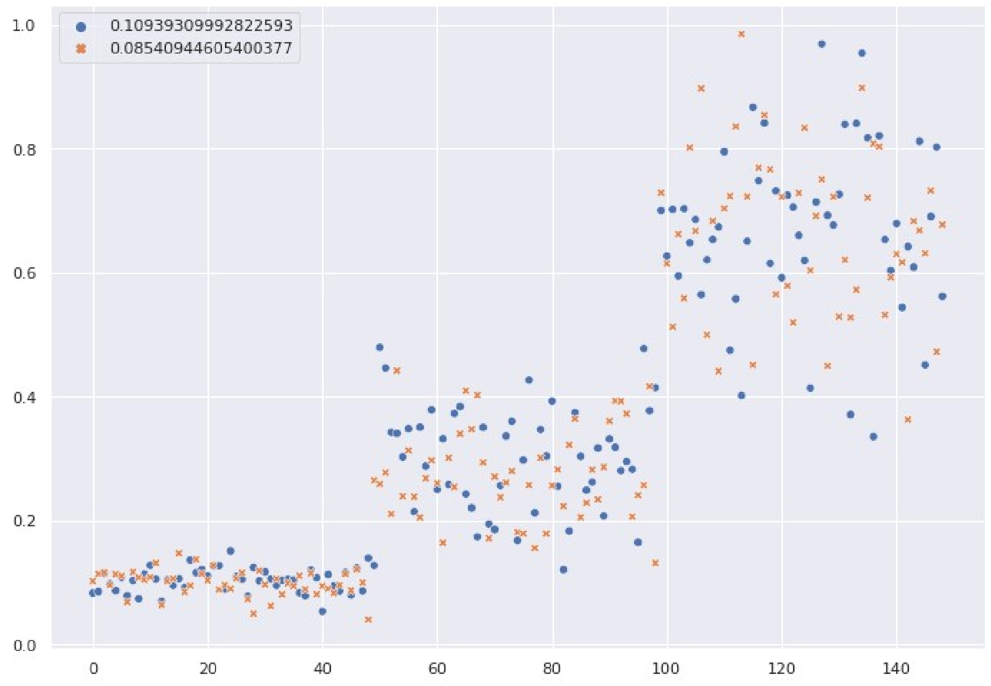
```
Computational Time of Chameleon Clustering Technique with 51D(DS2)
Dataset: 20.642762184143066
```

Time complexity:
Time Complexity of Chameleon is O(n^2)

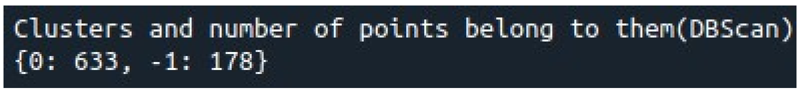
5. Which clustering technique is more suitable for your dataset? Write a discussion about it using the results mentioned above and characteristics of the clusters and the dataset.

Chameleon Technique has 0.876 Silhouette Score for small dataset (DS1) and DBSCAN has 0.700 Silhouette Score for big dataset(DS2). If we compare them in terms of complexity, Chameleon has O(n^2) complexity and it's not very good therefore we can look at other clustering technique whose Silhouette Score for small dataset (DS1) is closer to Chameleon's and it's 0.622. Besides Silhouette Score, KMeans has O(n) complexity therefore KMeans is a better choise for small dataset(DS1). DBSCAN is best solution for big dataset(DS2) in terms of both complexity and Silhouette Score. My datasets don't have spherical shape therefore KMeans and DBScan are the best solutions for them.

KMeans for DS1:



DBScan for DS2:



```
Clusters and number of points belong to them(DBScan)
{0: 633, -1: 178}
```