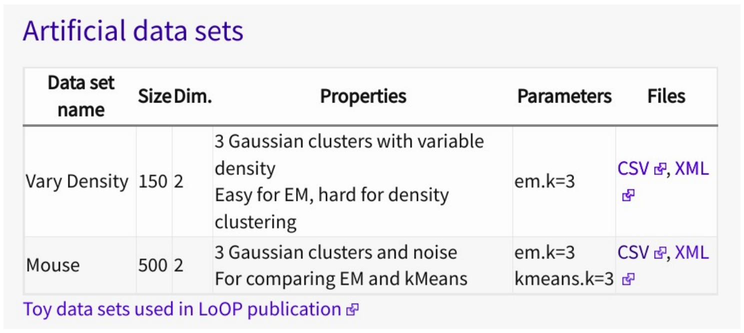
HW2 – REPORT

Can Duyar – 171044075

* I used a 2-dimensional data set from the website that I explained in below:
https://elki-project.github.io/datasets/ (I used the Mouse.csv dataset)



Artificial data sets

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
Mouse	500	2	3 Gaussian clusters and noise For comparing EM and kMeans	em.k=3 kmeans.k=3	CSV, XML

Toy data sets used in LoOP publication

* I used a data set that includes more than 20-dimension from the Kaggle. This data set is related with Breast cancer. I cleaned the data set to avoid from the unnecessary data.
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

Dataset

Breast Cancer Wisconsin (Diagnostic) Data Set

Predict whether the cancer is benign or malignant

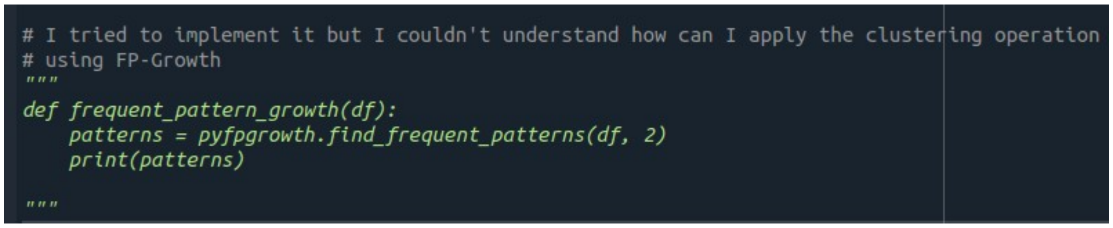
UCI Machine Learning • updated 5 years ago (Version 2)

Solutions of the Questions:

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.

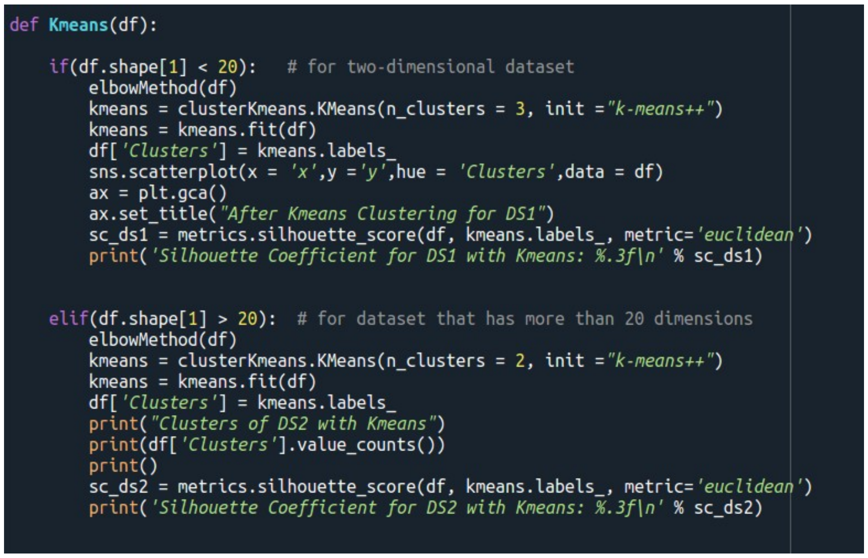
I tried to implement Fp-Growth but I couldn't understand how can I apply clustering operation using Fp-Growth, I just used it to find frequent patterns instead of clusters but my other implementations are totally done.

FP-GROWTH:



```
# I tried to implement it but I couldn't understand how can I apply the clustering operation
# using FP-Growth
"""
def frequent_pattern_growth(df):
    patterns = pyfpgrowth.find_frequent_patterns(df, 2)
    print(patterns)

"""
```

Kmeans:

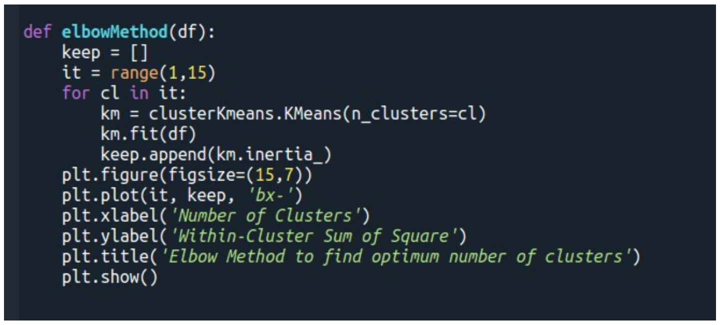


```
def Kmeans(df):

    if(df.shape[1] < 20):   # for two-dimensional dataset
        elbowMethod(df)
        kmeans = clusterKmeans.KMeans(n_clusters = 3, init ="k-means++")
        kmeans = kmeans.fit(df)
        df['Clusters'] = kmeans.labels_
        sns.scatterplot(x = 'x',y ='y',hue = 'Clusters',data = df)
        ax = plt.gca()
        ax.set_title("After Kmeans Clustering for DS1")
        sc_ds1 = metrics.silhouette_score(df, kmeans.labels_, metric='euclidean')
        print('Silhouette Coefficient for DS1 with Kmeans: %.3f\n' % sc_ds1)


    elif(df.shape[1] > 20):  # for dataset that has more than 20 dimensions
        elbowMethod(df)
        kmeans = clusterKmeans.KMeans(n_clusters = 2, init ="k-means++")
        kmeans = kmeans.fit(df)
        df['Clusters'] = kmeans.labels_
        print("Clusters of DS2 with Kmeans")
        print(df['Clusters'].value_counts())
        print()
        sc_ds2 = metrics.silhouette_score(df, kmeans.labels_, metric='euclidean')
        print('Silhouette Coefficient for DS2 with Kmeans: %.3f\n' % sc_ds2)
```

-I wrote only one Kmeans method that works with two separate conditions for both DS1 and DS2.

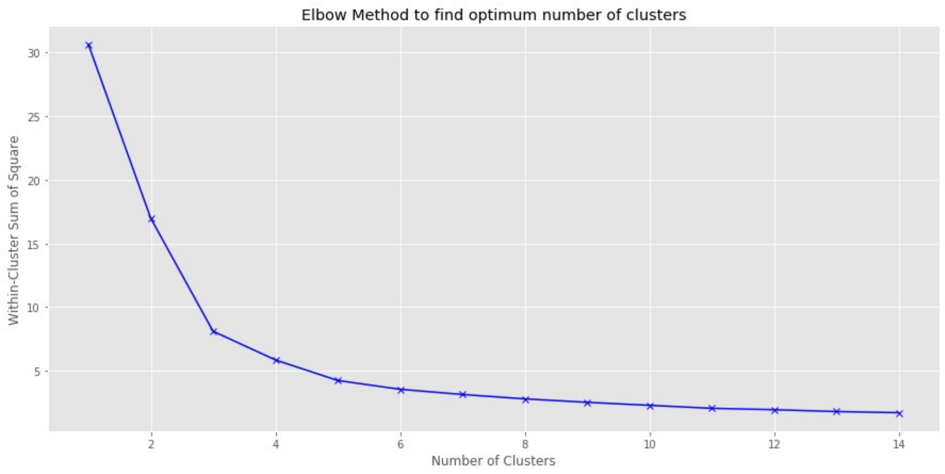
-I applied the elbow method for both DS1 and DS2 and found the optimum number of clusters with this method.

Implementation:

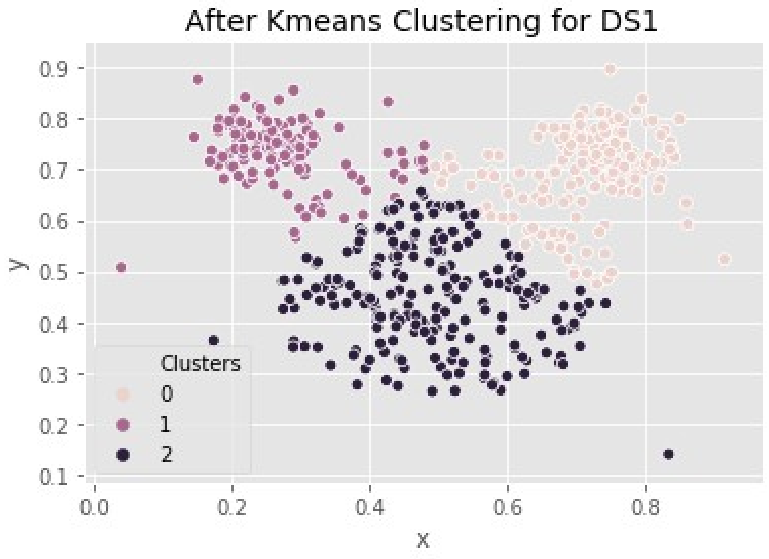


```
def elbowMethod(df):
    keep = []
    it = range(1,15)
    for cl in it:
        km = clusterKmeans.KMeans(n_clusters=cl)
        km.fit(df)
        keep.append(km.inertia_)
    plt.figure(figsize=(15,7))
    plt.plot(it, keep, 'bx-')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Within-Cluster Sum of Square')
    plt.title('Elbow Method to find optimum number of clusters')
    plt.show()
```

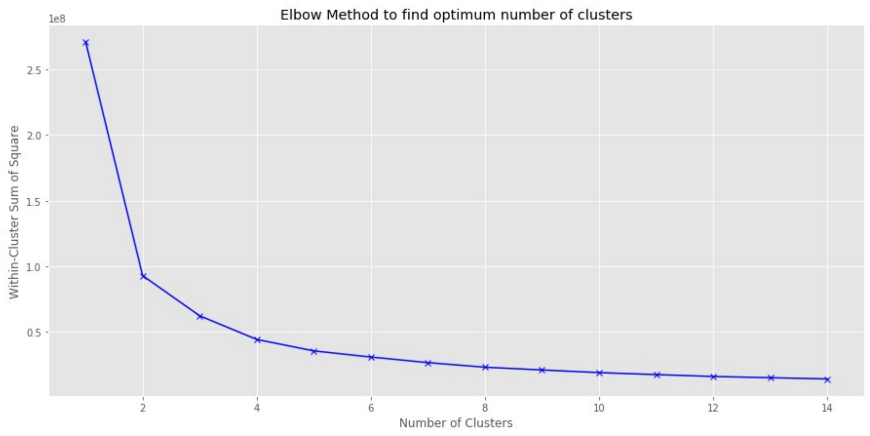
Output for DS1: Optimum number of clusters is around 3 for DS1 in this case.



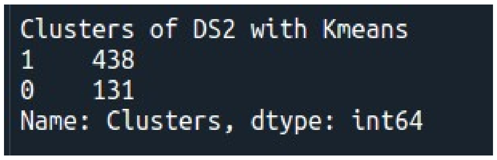
Clusters of DS1:



Output of DS2: Optimum number of clusters is 2 for DS2 in this case.

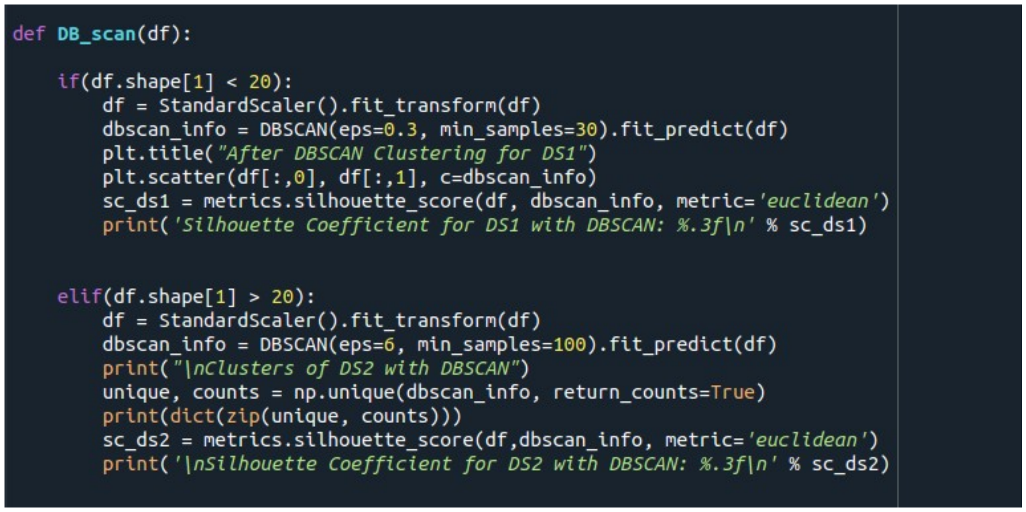


Clusters of DS2 (In the assignment the visualization was only demanded for DS1 so I expressed it numerically for DS2)



```
Clusters of DS2 with Kmeans
1    438
0    131
Name: Clusters, dtype: int64
```

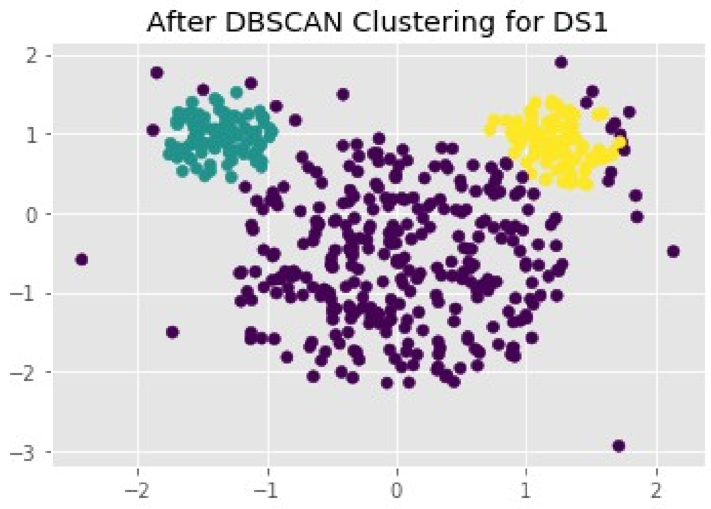
DBSCAN:



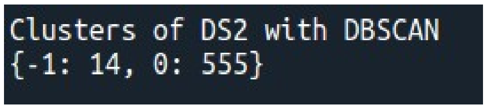
```
def DB_scan(df):

    if(df.shape[1] < 20):
        df = StandardScaler().fit_transform(df)
        dbscan_info = DBSCAN(eps=0.3, min_samples=30).fit_predict(df)
        plt.title("After DBSCAN Clustering for DS1")
        plt.scatter(df[:,0], df[:,1], c=dbscan_info)
        sc_ds1 = metrics.silhouette_score(df, dbscan_info, metric='euclidean')
        print('Silhouette Coefficient for DS1 with DBSCAN: %.3f\n' % sc_ds1)


    elif(df.shape[1] > 20):
        df = StandardScaler().fit_transform(df)
        dbscan_info = DBSCAN(eps=6, min_samples=100).fit_predict(df)
        print("\nClusters of DS2 with DBSCAN")
        unique, counts = np.unique(dbscan_info, return_counts=True)
        print(dict(zip(unique, counts)))
        sc_ds2 = metrics.silhouette_score(df,dbscan_info, metric='euclidean')
        print('\nSilhouette Coefficient for DS2 with DBSCAN: %.3f\n' % sc_ds2)
```

-I wrote only one DBSCAN method that works with two separate conditions for both DS1 and DS2. I picked min_samples and eps values randomly.

Clusters of DS1:

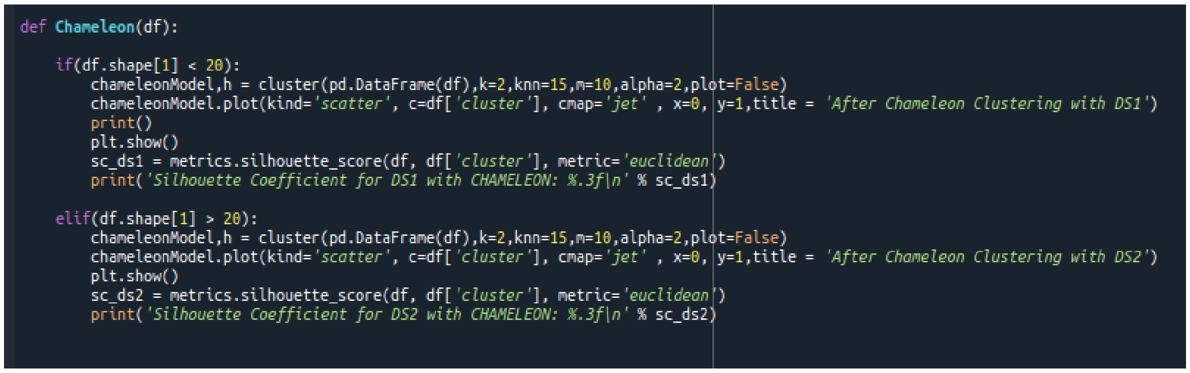


<u>Clusters of DS2:</u> (In the assignment the visualization was only demanded for DS1 so I expressed it numerically for DS2)



```
Clusters of DS2 with DBSCAN
{-1: 14, 0: 555}
```

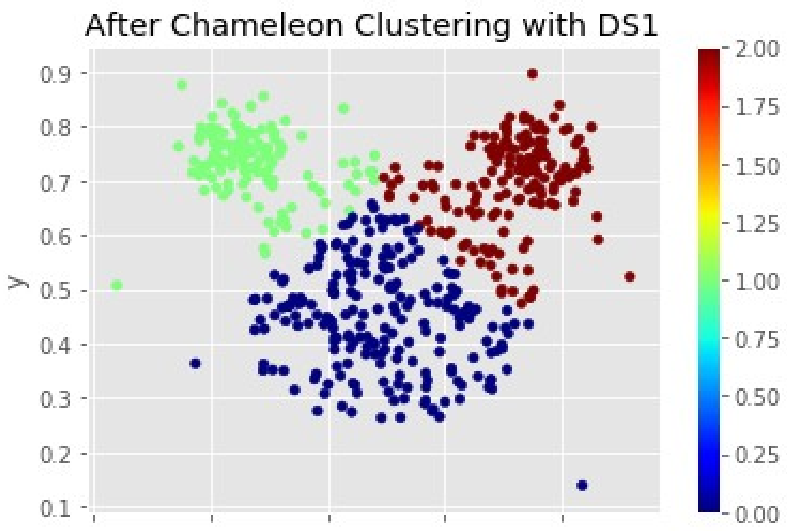
CHAMELEON:



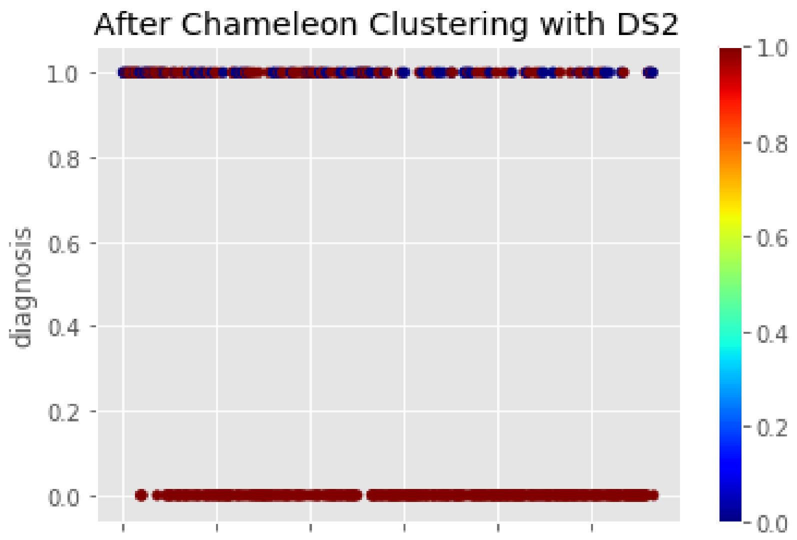
```python
def Chameleon(df):

    if(df.shape[1] < 20):
        chameleonModel,h = cluster(pd.DataFrame(df),k=2,knn=15,m=10,alpha=2,plot=False)
        chameleonModel.plot(kind='scatter', c=df['cluster'], cmap='jet' , x=0, y=1,title = 'After Chameleon Clustering with DS1')
        print()
        plt.show()
        sc_ds1 = metrics.silhouette_score(df, df['cluster'], metric='euclidean')
        print('Silhouette Coefficient for DS1 with CHAMELEON: %.3f\n' % sc_ds1)

    elif(df.shape[1] > 20):
        chameleonModel,h = cluster(pd.DataFrame(df),k=2,knn=15,m=10,alpha=2,plot=False)
        chameleonModel.plot(kind='scatter', c=df['cluster'], cmap='jet' , x=0, y=1,title = 'After Chameleon Clustering with DS2')
        plt.show()
        sc_ds2 = metrics.silhouette_score(df, df['cluster'], metric='euclidean')
        print('Silhouette Coefficient for DS2 with CHAMELEON: %.3f\n' % sc_ds2)
```

-I wrote only one CHAMELEON method that works with two separate conditions for both DS1 and DS2. I picked parameters randomly.

<u>Clusters of DS1:</u>



Clusters of DS2: (It was not in the assignment to visualize this part, but I wanted to visualize it to see it better.)

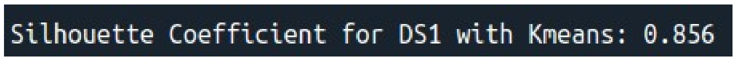


- There are two clusters for malignant and benign tumor cells. (1 = malignant , 0 = benign)

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

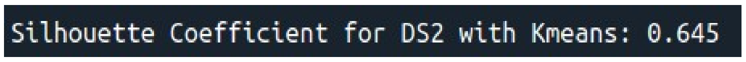
Kmeans:

DS1:



```
Silhouette Coefficient for DS1 with Kmeans: 0.856
```

DS2:



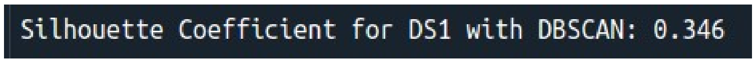
```
Silhouette Coefficient for DS2 with Kmeans: 0.645
```

silhouette_coefficient_DS1 > silhouette_coefficient_DS2

- Clustering is better in the DS1(small data set) than DS2(big data set)

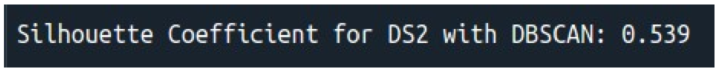
DBSCAN:

DS1:



```
Silhouette Coefficient for DS1 with DBSCAN: 0.346
```

DS2:



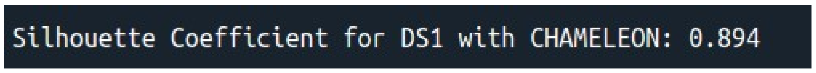
```
Silhouette Coefficient for DS2 with DBSCAN: 0.539
```

silhouette_coefficient_DS2 > silhouette_coefficient_DS1

- Clustering is better in the DS2 than DS1

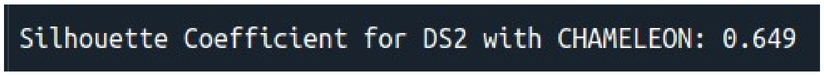
CHAMELEON:

DS1:



```
Silhouette Coefficient for DS1 with CHAMELEON: 0.894
```

DS2:



```
Silhouette Coefficient for DS2 with CHAMELEON: 0.649
```

silhouette_coefficient_DS1 > silhouette_coefficient_DS2

- Clustering is better in the DS1 than DS2

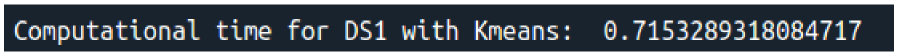
Silhouette Coefficient Comparison for DS1:
Chameleon(0.894) > Kmeans(0.856) > DBSCAN(0.346)

Silhouette Coefficient Comparison for DS2:
Chameleon(0.649) > Kmeans(0.645) > DBSCAN(0.539)

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

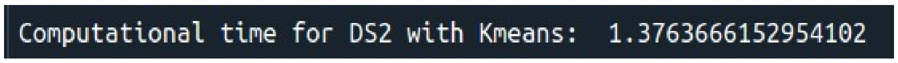
Kmeans:

Computational time for DS1:



```
Computational time for DS1 with Kmeans:  0.7153289318084717
```

Computational time for DS2:

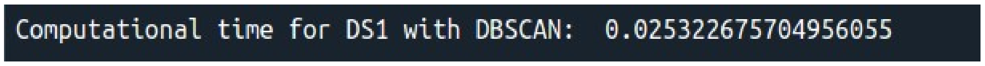


```
Computational time for DS2 with Kmeans:  1.3763666152954102
```

Time complexity:
-Time Complexity of Kmeans is O(n)

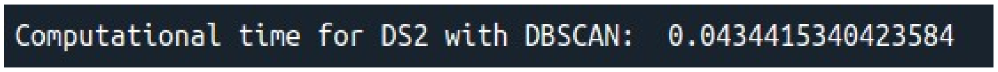
DBSCAN:

Computational time for DS1:



```
Computational time for DS1 with DBSCAN:  0.025322675704956055
```

Computational time for DS2:

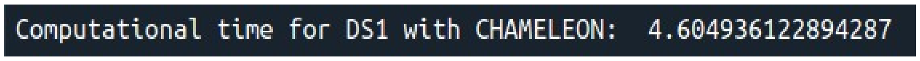


```
Computational time for DS2 with DBSCAN:  0.0434415340423584
```

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

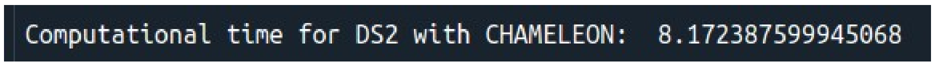
CHAMELEON:

Computational time for DS1:



```
Computational time for DS1 with CHAMELEON:  4.604936122894287
```

Computational time for DS2:

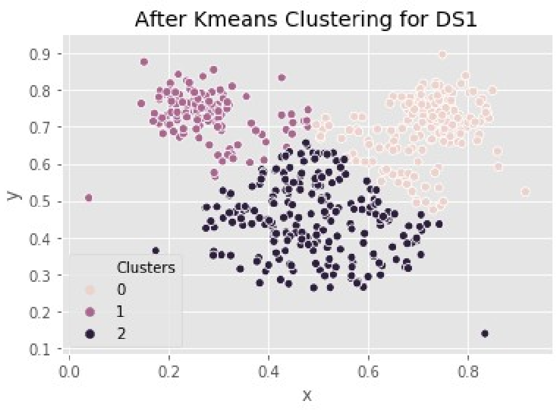


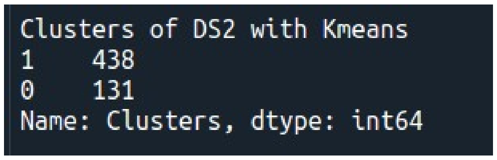
```
Computational time for DS2 with CHAMELEON:  8.172387599945068
```

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.

If we compare the silhouette coefficients(scores) and time complexities then we can say that kmeans is better than others. Chameleon looks like better because of silhouette coefficient values but if we calculate the time complexity of the Chameleon then we can see that It's O(n^2). It's slower than others so we need to chose an better algorithm in terms of time complexity and silhouette coefficients. Kmeans has 0.856 silhouette value for DS1 and 0.645 for DS2. It's so close to Chameleon and its time complexity is O(n). (better than Chameleon in this case). As a result we can say that Kmeans it the best algorithm for my data sets. My data points have a spherical shape that's why kmeans is one of the good solutions to cluster them.




```
Clusters of DS2 with Kmeans
1    438
0    131
Name: Clusters, dtype: int64
```