**Data Mining Portfolio Entry for Clustering**

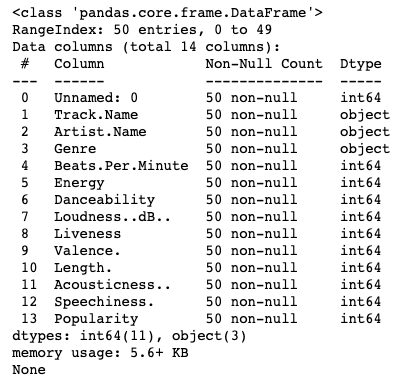
**Data acquisition**

For this DMP, I used a total of 4 datasets.

The **FIRST** dataset I used in this chapter is the Iris\_student.csv.dat.csv data provided by Dr. Santago. This dataset has 150 data objects and 5 columns. This dataset has 4 numerical features and 1 categorical class variable.

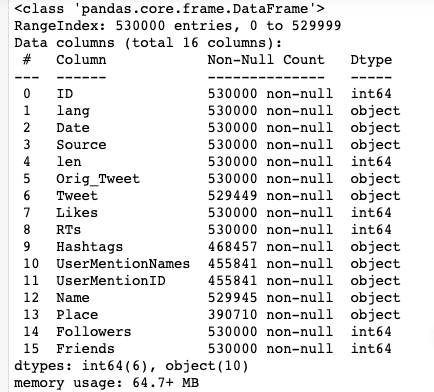
The **SECOND** dataset I used in this chapter is the separated\_2d.csv data provided by Dr. Santago. This dataset has 61 data objects and 3 columns. This dataset has 2 numerical features and 1 categorical class variable.

The **THIRD** dataset is called the top50.csv dataset that I found on Kaggle. This dataset contains the top50 most listened songs in the world and their corresponding features. The dataset has 50 rows and 14 columns. Some basic information about this dataset is shown below:



The features I am going to focus on are: Genre, Beats.Per.Munite, Energy, Danceability, Loudness..dB.., Liveness, Valence., Length., Acousticness.., Speechiness.., Popularity. I want to cluster the data objects, based on some of the features above, to their correct genres.

The **FOURTH** dataset is called FIFA.csv dataset that I found on Kaggle. This dataset contains a random collection of 530k tweets starting from the Round of 16 till the World Cup Final that took place on 15 July, 2018 & was won by France. Multiple features are available in this dataset as shown in the figure below:



Despite the fact that we have multiple features at hand, the specific feature of my interest is the “Tweet” feature which is essentially the content of each tweet. This feature is categorical and I will further process this feature as will be demonstrated later.

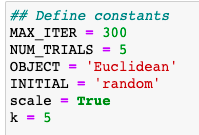
**Program development**

In this program development section, I implemented two Algorithms. One is the K-means algorithm (including the bisecting K-means); the other is the hierarchical clustering algorithm.

The **FIRST** one is the K-means (and some variation of k-means) algorithm. Here below is a summary of things I implemented in my code:

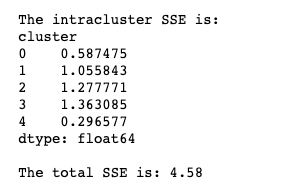
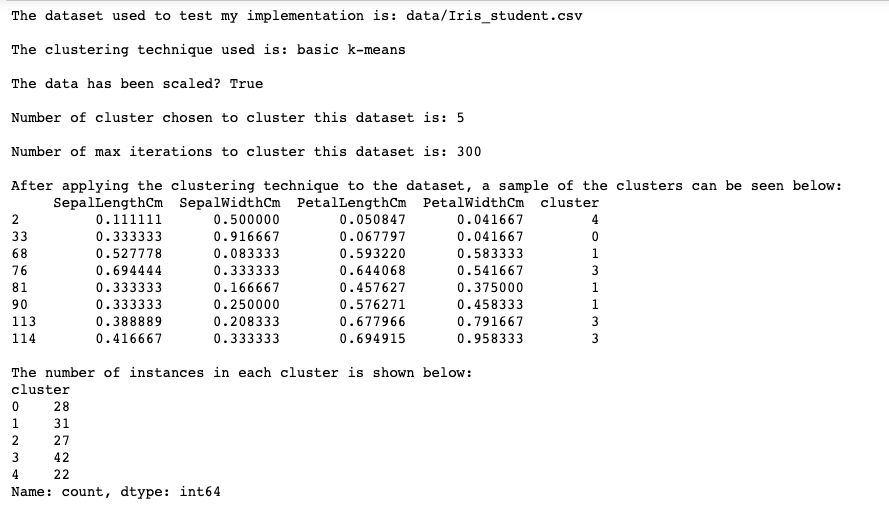
1. Data is read and stored as a pandas dataframe. Users may select to scale or not scale the data.
2. Two algorithms are implemented in this section.
   1. Basic k-means
   2. Bisecting k-means
3. My implementation allows the following distance metric: Euclidean, One-norm (Manhattan distance), and Infinity norm
4. Users may choose to initialize the centroid assignments randomly or using the K-means++ algorithm.
5. Data structures in this algorithm is implemented as below:
   1. The K-centroids object is implemented as a DataFrame object with the first few columns being the features values and the last column being the cluster this object belongs to. Dataset for clustering is stored also in this way.
6. The final clusters are evaluated using several evaluation metrics. The metrics I implemented are: Cohesion, Separation, Silhouette Coefficient, and Correlation Coefficient.

Implementation details can be found in the file named K\_means implementation.ipynb. To demonstrate the code I implemented, I used the dataset Iris\_student.csv to test my K-means algorithm. I started with the hyperparameters:

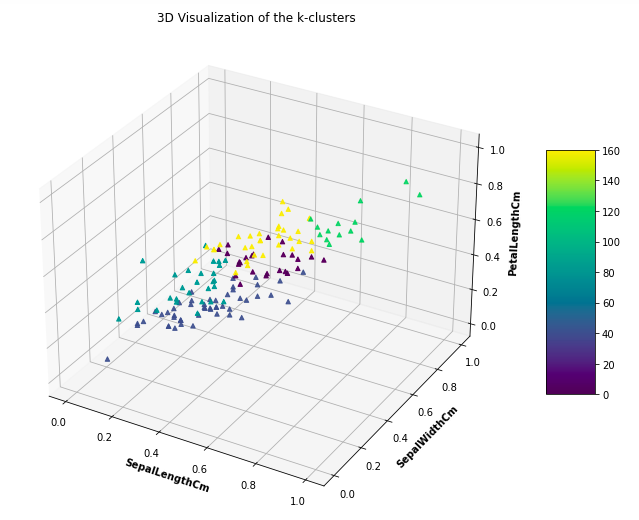


1. MAX\_ITER: The maximum iterations of the K-means algorithm.
2. NUM\_TRIALS: The number of trails executing the basic K-means algorithm when bisecting the selected cluster
3. OBJECT: The objective function, currently set as Euclidean distance. It can also be the Manhattan distance or the Infinity-norm.
4. INITIAL: How to form initial centroids--random assignment or k-means++.
5. scale: Should the dataset be scaled or not before K-means clustering.
6. k: The number of clusters

Some basic information about the k-means algorithm is printed to the console:

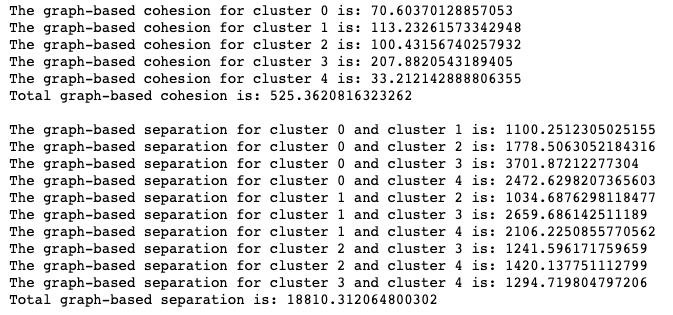
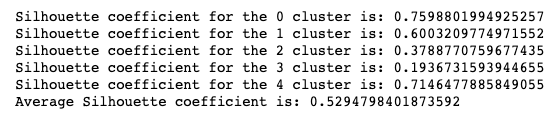
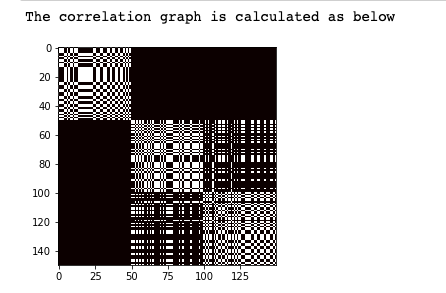


I also graphed the final clusters based on three of the features in the dataset. The features are: SepalLengthCm, SepalWidthCm, and PetalLengthCm. Shown below is the 3D plot demonstrating the geographical distribution of the clusters:



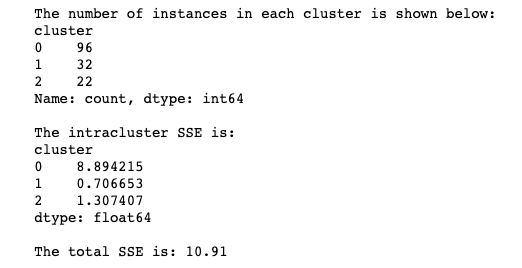
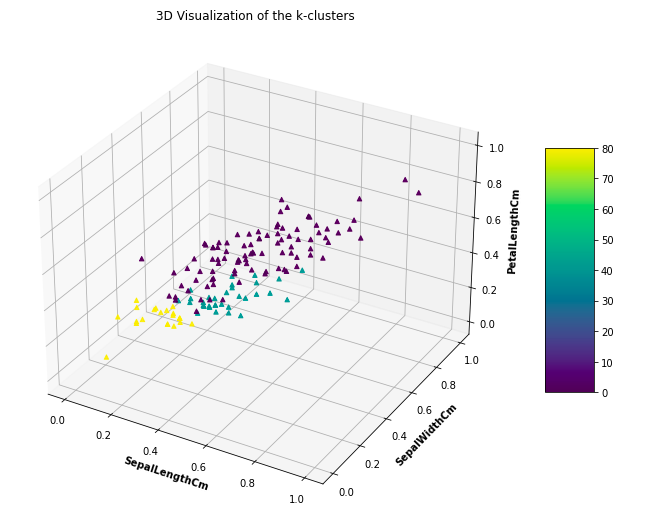
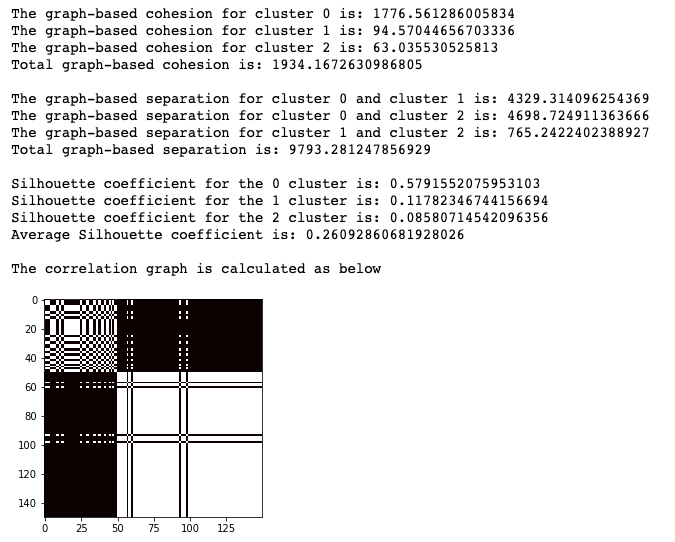
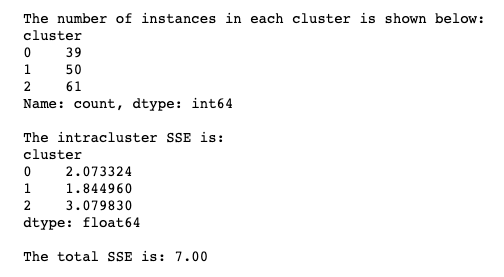
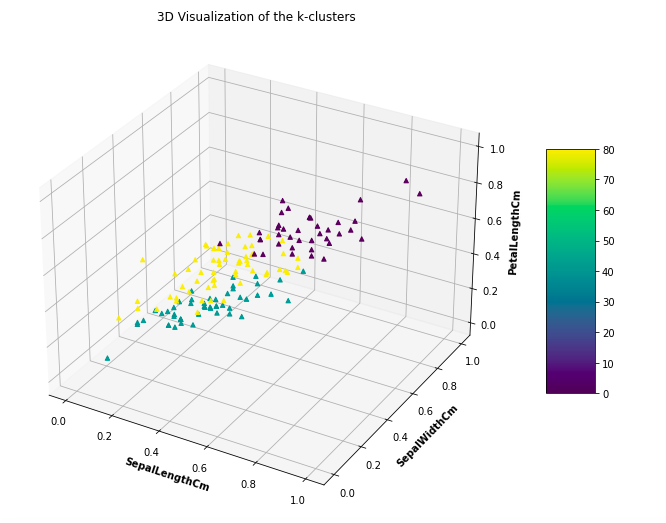
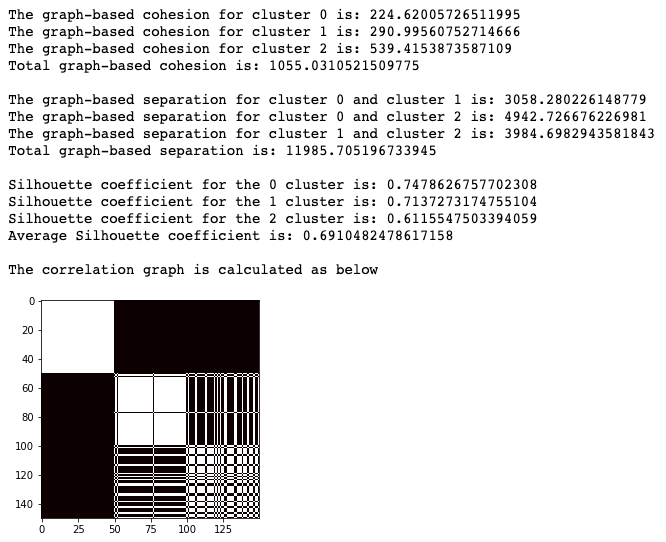
From the graph, it may be concluded that the geographic distribution of the clusters are visually separable, even though there is indeed another feature that is excluded in this graph. Therefore, the k-means clustering I implemented makes sense from the first glance.

Still, more evaluation criteria need to be calculated in order to conclude that this clustering is effective. To do so, I implemented codes to calculate several evaluation metrics and the results are shown below:

1. Graph-based cohesion, separation: 
2. Silhouette Coefficient:
3. Correlation coefficient matrix: The correlation coefficient matrix can be hard to show but through plotting each individual entry as either black or white, the effectiveness of the clusters can still be seen:

Evaluation: Even though both the Graph-based cohesion, separation and the silhouette coefficient are showing pretty good clustering structures, the correlation graph suggests that a cluster size of 3 should suffice for this problem. Therefore, I should change the hyperparameter k to 3 instead of 5. In this following section, I experimented with changing the hyperparameters and how that affects the performance of the clustering conducted.

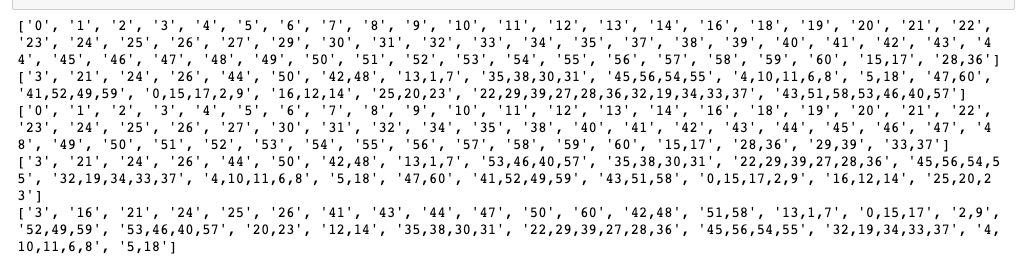
Modifying the hyperparameters:

1. Set k = 3 (decreasing the number of clusters):
   1. The intracluster SSE and also the total SSE increases compared to when k=5. This is consistent with our understanding since having less clusters means greater errors. 
   2. The 3D scatter plot shows separated structure of clustering even though one cluster seems to cluster a large number of clusters compared to the other two.
   3. The evaluation metrics also change. Cohesion increases while the separation decreases. Silhouette coefficient is also smaller but the correlation graph looks pretty good. 
2. Setting INITIAL = ‘k-means++’ (use k-means++ algorithm to set initial centroids):
   1. The intracluster SSE and also the total SSE decreases. This is consistent with our understanding since setting appropriate initial centroids have significant impacts on the final clustering results. And the k-means++ algorithm generally performs well for initial centroid selection.
   2. The 3D scatter plot shows an even better structure of clustering. 
   3. Changing the initial assignment method also decreases the cohesion and increases the separation greatly. The silhouette coefficient shows pretty good clustering. 
   4. Still, it can easily be noticed that through using K-means++ initialization, the processing time required to execute the k-means algorithm is longer.

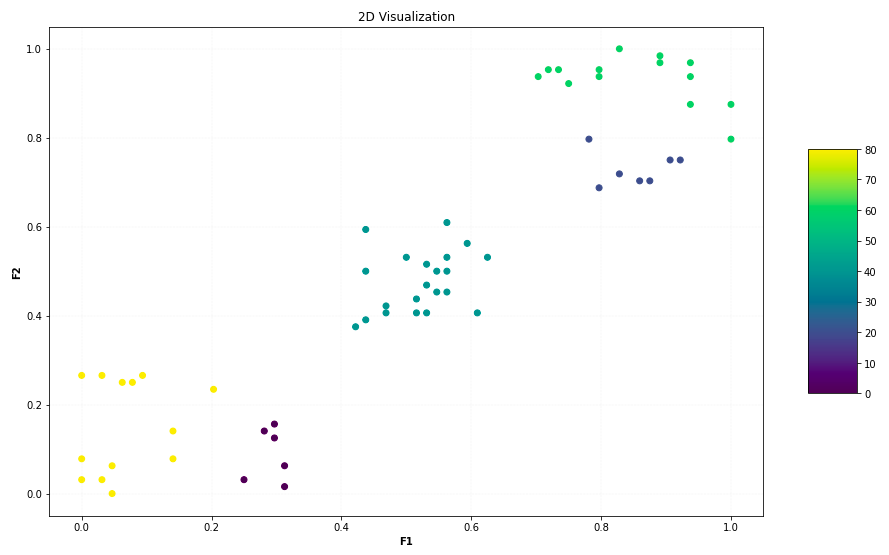
The **SECOND** algorithm I implemented is the Hierarchical Clustering algorithm. Here below is a summary of things I implemented in my code:

1. Data is read and stored as a pandas dataframe. Users may select to scale or not scale the data.
2. My implementation allows the following proximity measures: single link, complete link, and group average.
3. Data structures in this algorithm is implemented as below:
   1. Similarity matrix is implemented as a DataFrame object with each entry indicating the proximity between the objects indicated by index number and column number.
   2. Data structure to store the final clusters is implemented as a nested list where each first level list indicates a level of cluster in hierarchical clustering.

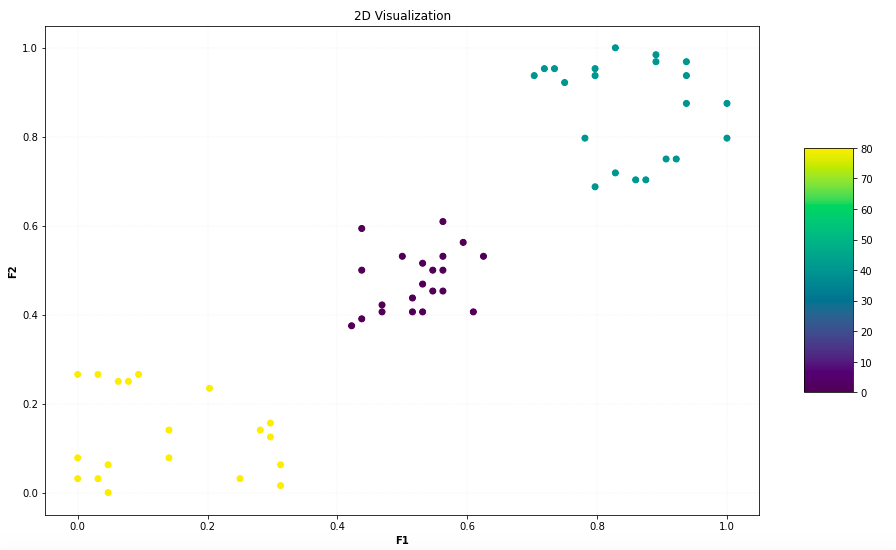
Since the Iris\_student.csv dataset has 150 data objects and it would be time consuming to go through the entire dataset, I used the dataset named separated\_2d.csv instead to test my hierarchical clustering. After scaling the dataset, hierarchical clustering is employed to cluster the dataset. Several runs will lead to the clusters as shown below:



These clusters can be visualized using 2D plots. For instance, in the case of 5 clusters, the 5 clusters look like:



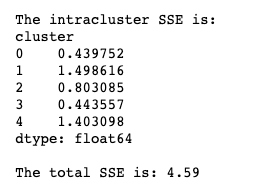
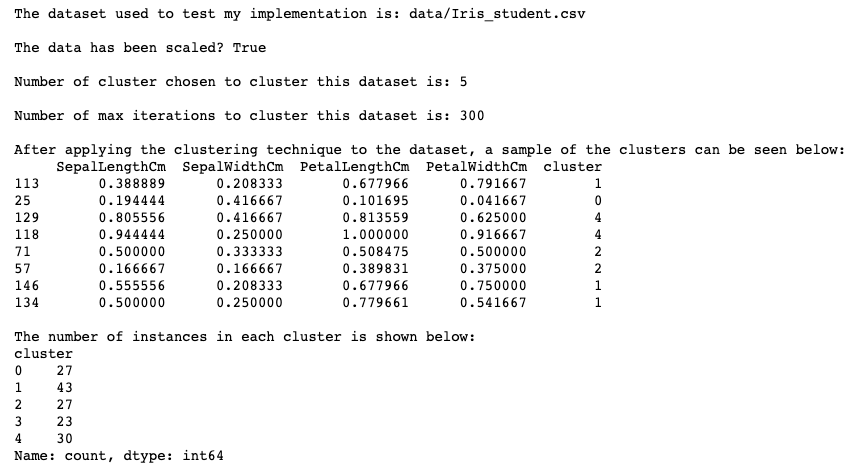
The clustering structure seems to separate the data objects pretty well. Another instance I tried is to see how 3 clusters look like in this dataset since from the visualization above it seems that the data clusters in top right and bottom left could both be clustered together. Here below are the 3 clusters fit by hierarchical clustering:



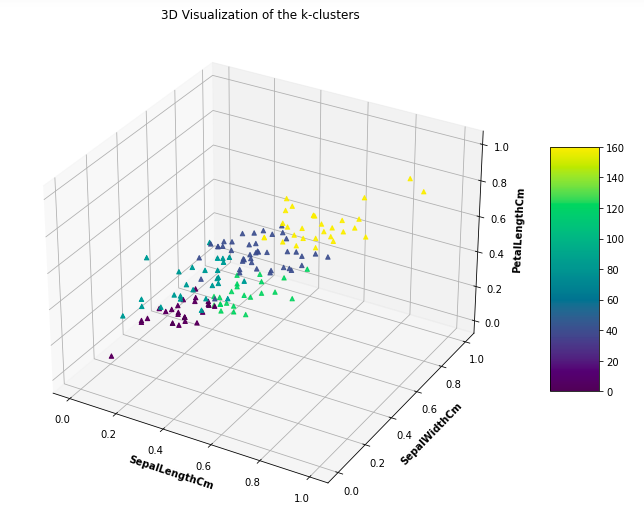
**Data analysis and package use**

In this chapter, I use the package sklearn to develop three programs related to clustering. All detailed implementations of these programs can be found in the file package use.ipynb.

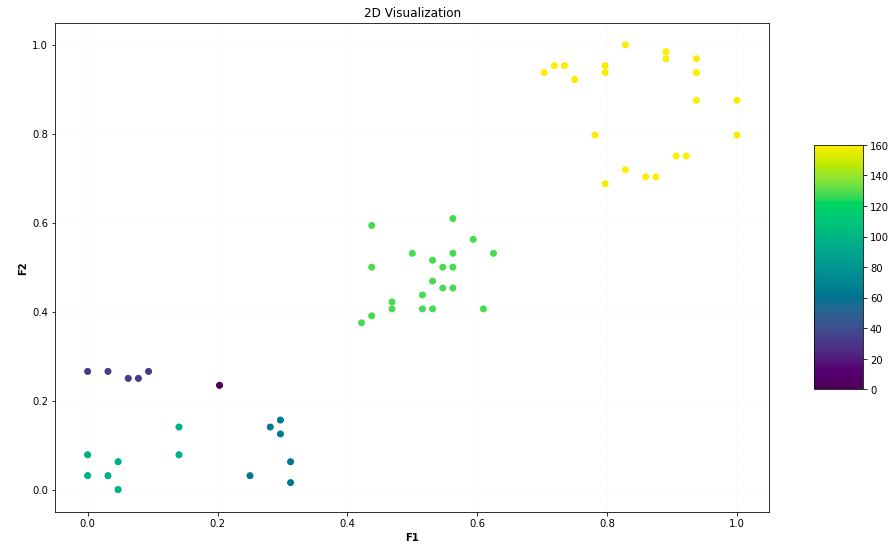
In the **FIRST** program, I used the package sklearn to conduct K-means clustering on the dataset Iris\_student.csv. This result is used to validate the result I get through running the K-means algorithm I implemented in the code development section. Similar to the code development section, I output the results in an organized way and visualized the clusters. Here shown below are the results from using K-means in sklearn to construct the clusters:



The result I get through choosing 5 clusters and random initial centroid assignment is almost exactly the same as I get from using my own implemented algorithm. The total SSE is 4.59, with only a 0.01 difference from the SSE I get in the code development section (SSE = 4.58). This demonstrates that my implementation in the code development section is effective. I also visualized the result:



Since I’ve already tested hierarchical clustering in the code development section, I would like to use the DBSCAN algorithm to cluster the dataset separate\_2d.csv once again and compare the results. Therefore, in the **SECOND** program, I used the DBSCAN algorithm to cluster the dataset. Here shown below is the visualization of results when using EPs = 0.1 and MinPts = 3:

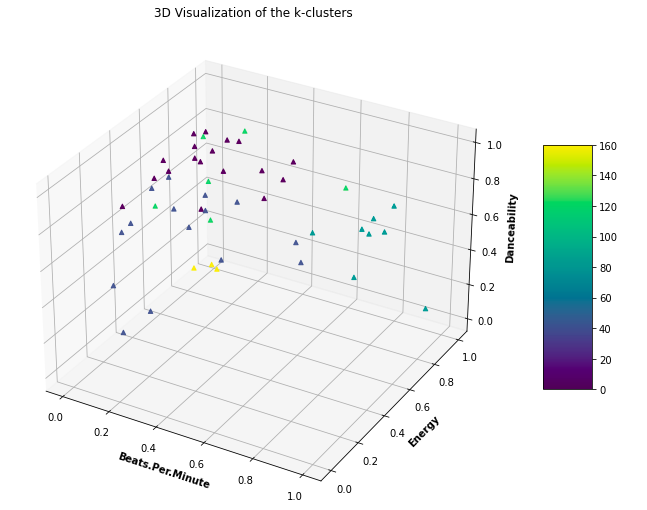


Although both the hierarchical clustering and the DBSCAN give 5 clusters, the cluster distribution varies to some extent. The clusters given by DBSCAN creates 3 clusters on bottom left while leaving lots of data objects on top right in one cluster. Result given by hierarchical clustering, on the other hand, separates the data objects on the top right into two different clusters. This difference is due to the fundamental difference in how both clustering techniques separate data into clusters. Data points on the top right have at least some connections that are close to each other, thus they are put into the same cluster.

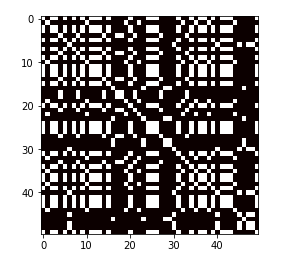
In the **SECOND** program, I once again used the package sklearn to conduct K-means clustering on the dataset 50top.csv. Since there are 21 different genres among the 50 songs, it would be unrealistic to create 21 clusters using 50 data objects, I manually created 5 broader genres for clustering:

1. Pop songs
2. Rap songs
3. Edm
4. Latin
5. Other

I then used K-means to cluster the dataset. In order to evaluate the effectiveness of my clustering, I used 3 of the features: Beats.Per.Munite, Energy, Danceability to paint a 3d graph and to see if the distribution makes sense according to clustering. Here shown below is the 3d graph:



According to the graph shown above, a few clusters are separated well enough even though on the top the clusters are still hard to distinguish. In those areas, the clustering criteria should depend on some other features. Also, to further evaluate my clusters, I sorted data objects in sequence according to their clustering, and created a block-diagonal matrix based on whether or not the two data objects originally belong to the same cluster. Here shown below is my observation. The structure may not be highly significant even though there are certain blocks of entries which indicates that certain objects that are put into the same cluster originally belong to the same genre. Still, this clustering exercise has the weakness that it only has 50 observations and I need to cluster them all into 5 clusters. However, in the next program development, I will use the FIFA dataset which contains 500k data objects.



The **THIRD** program I developed is related to a bigger dataset. To experiment with datasets having more data objects, I decided to conduct clustering on the FIFA.csv dataset which have more than 500k data objects. However, there are some difficulties when working with this dataset. To begin with, this dataset intends to analyze textual data in order to extract information, which means we will need to conduct clustering based on textual data. Another limitation is that it would take us pretty long to cluster 500k+ data objects. To better solve these problems, I need to preprocess the data properly.

In the data preprocessing part, since the dataset is too large for the purpose of this program, I sampled 2000 data points from the dataset. Also, I only kept the 'Orig\_Tweet’ feature which is the only thing I want to analyze for this problem. Still, the text information in the ‘Orig\_Tweet’ feature contains many extraneous information, stopwords, etc. that are irrelevant to our analysis. Therefore, I also wrote codes to eliminate them. Steps that I took are:

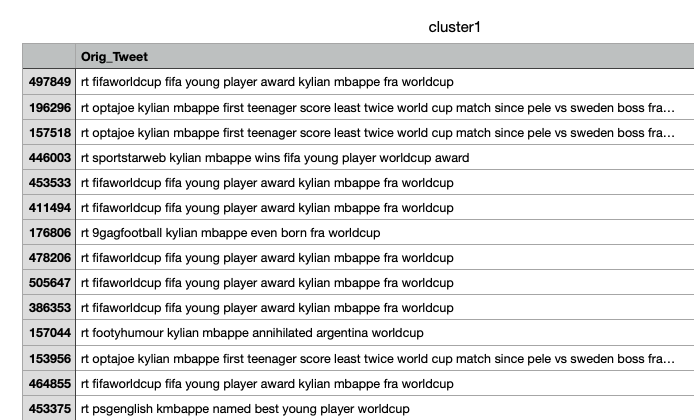
1. Remove URL: the regular expression I used is 'http\S+'
2. Change capital letters into lowercase ones
3. Remove digits: the regular expression is ‘ \d+’
4. Remove stop words in English

Now we have the preprocessed dataset. The next step is to transform textual data into numerical information such that we may apply clustering techniques to process them. Method I used for this program is the **TfidfVectorizer** feature in sklearn.feature\_extraction.text. This feature converts a collection of raw documents to a matrix of TF-IDF features. And the tf–idf features are values that are proportional to the number of times a word appears in the document but are offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. This feature allows us to transform a textual data into numeric “frequency” value, and thus allowing clustering. Here shown below are a set of “frequent” words after the transformation:

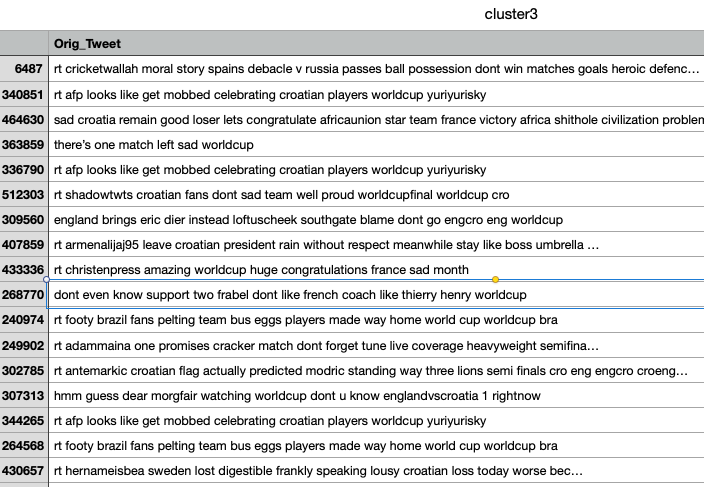


I then applied the K-mean algorithm to cluster the data points. In this part, I started with k=2 because I don’t want the clusters to be too complicated. However, I can barely distinguish between the textual information from both clusters since the k-means clustering is unable to find some deterministic properties that allows the separation of the dataset into only two clusters. I then tried out k=5, k=10, but none of those worked as I expected. Finally, when I used the value k=50 (which is a large cluster number), I am able to find some very significant clusters. Here below are two instances of the clusters:

Cluster 1: Data objects in this cluster almost all contain some information about the player Kylian Mbappe. This feature is quite significant.



Cluster 3: Information in this cluster is more obscure to see than in cluster 1, but we do observe that lots of the tweets in this cluster have words like “dont”, “yuriyurisky”... This may be one feature in this cluster even though it is hard to interpret them in common sense.



More information about these two clusters can be found in file cluster1.csv and cluster3.csv.

**Student learning summary and self-assessment**

Major takeaway:

1. Implementation of both K-means clustering and hierarchical clustering. I did put great effort into implementing these two algorithms such that they accommodate quite a few interactive inputs. For instance, K-means algorithm I implemented allows both random and kmeans++ initial centroid assignments. Such implementations deeply improve my understanding of these clustering techniques.
2. Different types of evaluation measures. There are so many different ways to evaluate the goodness of clustering. Among them, I am now most capable of using Silhouette Coefficient, cohesion/separation, and correlation to evaluate some clusters.
3. How different algorithms differ from each other. One of such instances is that I compared the clusters produced by the DBSCAN algorithm to the clusters produced by hierarchical algorithms. The results do differ from each other.

Another aspect of this chapter that I found interesting is how clustering techniques can be applied to textual analysis. Even though this is not covered in class, I’ve become interested in relating the subjects in class to natural language processing starting from the last chapter (Association Analysis). Through researching online and using already existing packages, I was capable of applying k-means clustering to textual data. The resulting clusters are also quite useful as we can see from the first cluster that it includes all tweets that are talking about Mbappe. With this work, I am now confident in doing more textual analysis in the future.

Besides that, something I want to explore more in this chapter are:

1. Try out more supervised clustering techniques, and use supervised evaluation metrics to test their performance.

**Self-evaluation:**

I believe that my understanding of notions related to clustering is solid. Comprehensively, I would like to give myself an A/A- on this chapter.