Introduction to Data Science

Clustering

Dawid Sitnik
2016118376

1.Definition

Clustering is a method of grouping objects, which are similar to each other, into one group (cluster) . It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

We shouldn't think about cluster analysis as about any special algorithm, it is rather a general problem to be solved. It can be solved in many different ways, using different algorithms, which differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. The most popular definition of include groups with small distances between cluster objects, dense areas of the space, intervals or particular statistical distributions. Clustering can be perceived as a optimization of multi-objective problem. We can not choose just one the best algorithm and its preferences. Depending on a problem, we will need to chose one that act the best for that particular case. Clustering is iterative process of getting new information or interactive multi-objective optimization that involves trial and failure rather than automatic task. Usually we need to change model parameters and data preprocessing many time until we succeed.

Besides the term clustering, there are a number of terms with similar meanings, including automatic classification, numerical taxonomy, botryology typological analysis, and community detection. The subtle differences are often in the use of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest.

2. Some types of clustering:

- K-means Clustering
- Agglomerative Clustering
- Affinity Propagation
- Guassian Mixture Modelling

2.1 K-means Clustering

This type of clustering is one of the simplest one, so it is one of the most popular as well. Using k-means, user has to define number of clusters. They are chosen, based on the closeness to the center value of the clusters. The initial center value is chosen by algorithm randomly. It is top-down approach because we decide how many clusters will we have first and than we group points to k groups.

K-means is considered by many to be the gold standard when it comes to clustering due to its simplicity and performance. When you have no idea at all what algorithm to use, K-means is usually the first choice. K-means works by defining spherical clusters that are separable in a way so that the mean value converges towards the cluster center. Because of this, K-Means may under perform sometimes.

2.2 Agglomerative Clustering

This type, also known as Hierarchical clustering, doesn't except user to define the number of clusters. At the beginning every point is considered as an independent cluster and then they are recursively clustered together. Clustering depends on the distances between the points. Agglomerative clustering tries to make distances between the points in one cluster as small as possible. In the same time it tries to maximize distances between clusters. Unlike k-means, this approach is bottom-up. Commonly used distance measures are Euclidean distance, Manhattan distance or Mahalanobis distance.

2.3 Affinity Propagation

Similarly to the previous algorithm, affinity propagation also doesn’t need number of clusters in its argument. It deduce it on its own. It is based on message passing between points. Like k-means, affinity propagation finds "exemplars", members of the input set that are representative of clusters.

Affinity propagation performs really well on several computer vision and biology problems, such as clustering pictures of human faces and identifying regulated transcripts, but we'll soon find out it doesn't work well for my dataset.

2.4 Guassian Mixture Modeling

It is probabilistic based clustering or kernel density estimation based clustering. The clusters are formed based on the Gaussian distribution of the centers

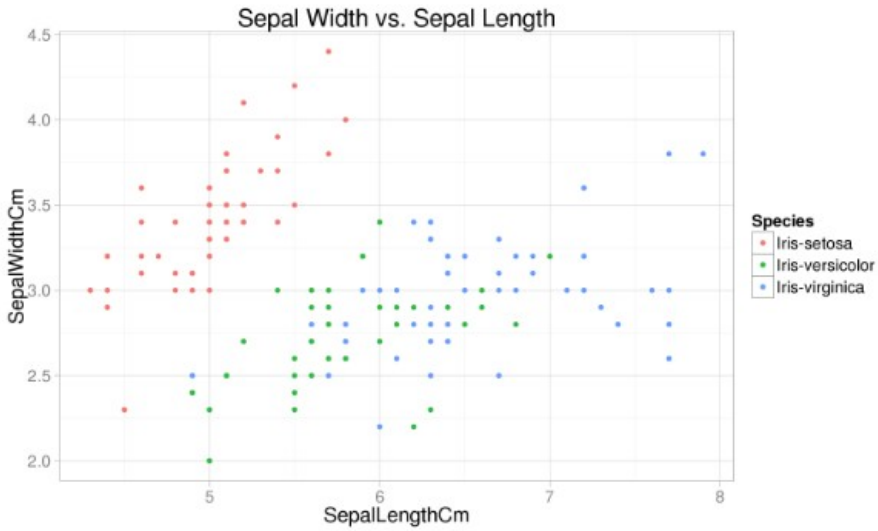
3. Description of my data set

The Iris data set was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species



4. Idea of the project

The main purpose is to make classification using different methods, described above. After getting results I am going to compare them with proper classification to see which method perform the best.

So to make my project works I have created four independent Python files (each one for single algorithm). Each Python file will fetch my data set, but it will not fetch the last column which is the proper class of the object. The algorithm will need to find it out on its own.

```
#Call required libraries
import time                    # To time processes
import warnings                # To suppress warnings

import numpy as np             # Data manipulation
import pandas as pd            # Dataframe manipulatio
import matplotlib.pyplot as plt                    # For graphics
import seaborn as sns
import plotly.plotly as py #For World Map
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

from sklearn.preprocessing import StandardScaler  # For scaling dataset
from sklearn.cluster import KMeans, AgglomerativeClustering, AffinityPropagation #For clustering
from sklearn.mixture import GaussianMixture #For GMM clustering

import os                      # For os related operations
import sys                     # For data size

wh = pd.read_csv("./Iris.csv") #Read the dataset
wh.describe()

print("Dimension of dataset: ")
print(wh.dtypes)

wh1 = wh[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] #Subsetting the data
```

I am considering four parameters sepal length, sepal width, petal length, petal width for clustering the flowers. Since the clustering is sensitive to range of data It is advisable to scale the data before proceeding, which is the next step of my programs.

```
#Scaling of data
ss = StandardScaler()
ss.fit_transform(wh1)
```

After that I am running certain algorithm and showing plot of my output classification. To make the result more readable I am restricting myself to draw just two dimensional plots with sepal width on Y axis and sepal height on X.

```
def doGMM(X, nclust=2):
    model = GaussianMixture(n_components=nclust,init_params='kmeans')
    model.fit(X)
    clust_labels3 = model.predict(X)
    return (clust_labels3)

clust_labels3 = doGMM(wh1,3)
gmm = pd.DataFrame(clust_labels3)
wh1.insert((wh1.shape[1]),'gmm',gmm)
```

```
#Plotting the cluster obtained using GMM
fig = plt.figure()
ax = fig.add_subplot(111)
scatter = ax.scatter(wh1['SepalLengthCm'],wh1['SepalWidthCm'],
                     c=gmm[0],s=50)
ax.set_title('GMM')
ax.set_xlabel('SepalLengthCm')
ax.set_ylabel('SepalWidthCm')
plt.colorbar(scatter)
```

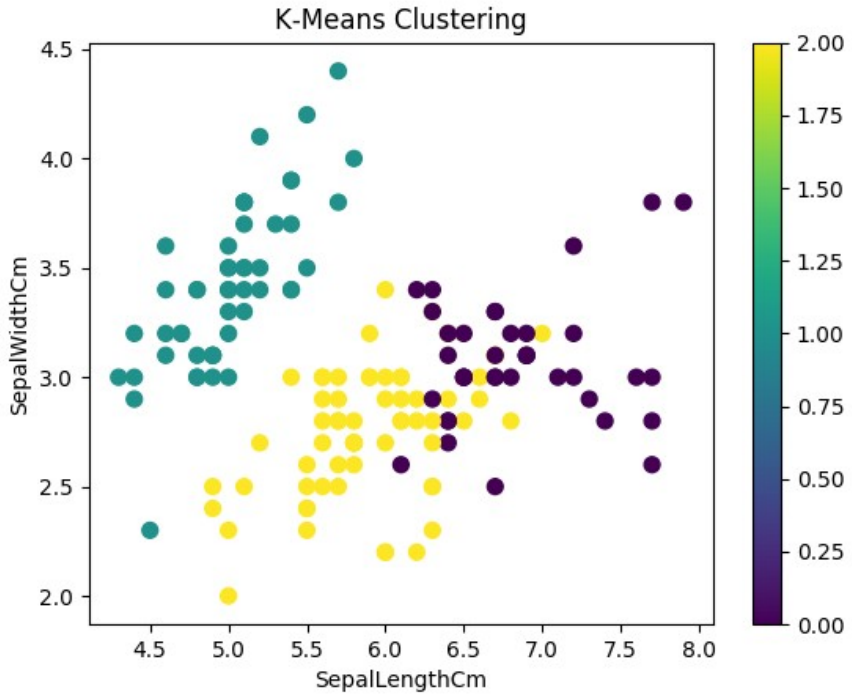
At the end I am writing my results to the output file and display the plot.

```
classification = pd.DataFrame(wh1, columns=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'gmm']).round(3).to_csv('./outputs/gmm.csv')
plt.show();
```

To decide which algorithm is the best for my case I am comparing output files with my input. What I am doing exactly is calculating an error, which is: number of misstakes / number of objects and looking for the smallest one.

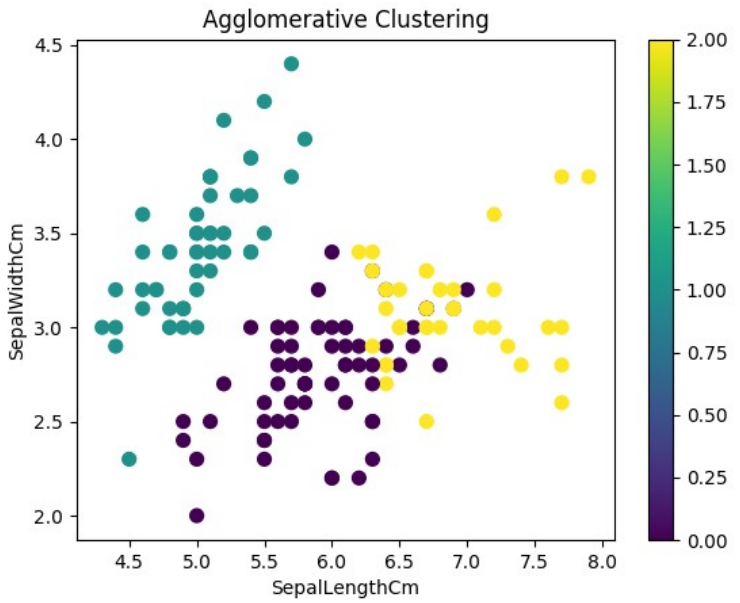
5.Results

5.1 K-means Clustering



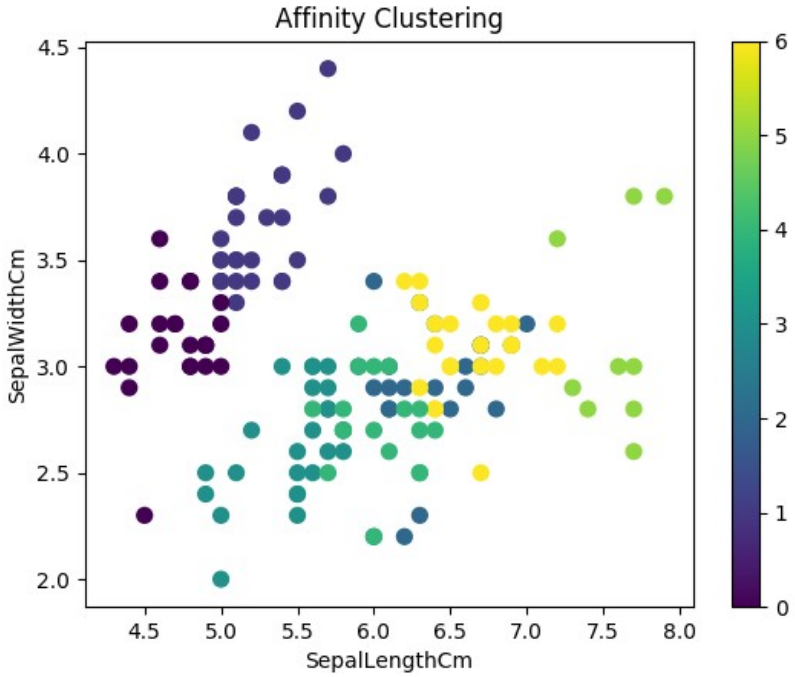
As we can notice, we got exactly 3 different classes of classification (as described in argument of our algorithm function). After analyzing output file and comparing it with the result which we should get, we can see that algorithm made mistakes in 16 fields. So its error rate is 0.106

5.2 Agglomerative Clustering



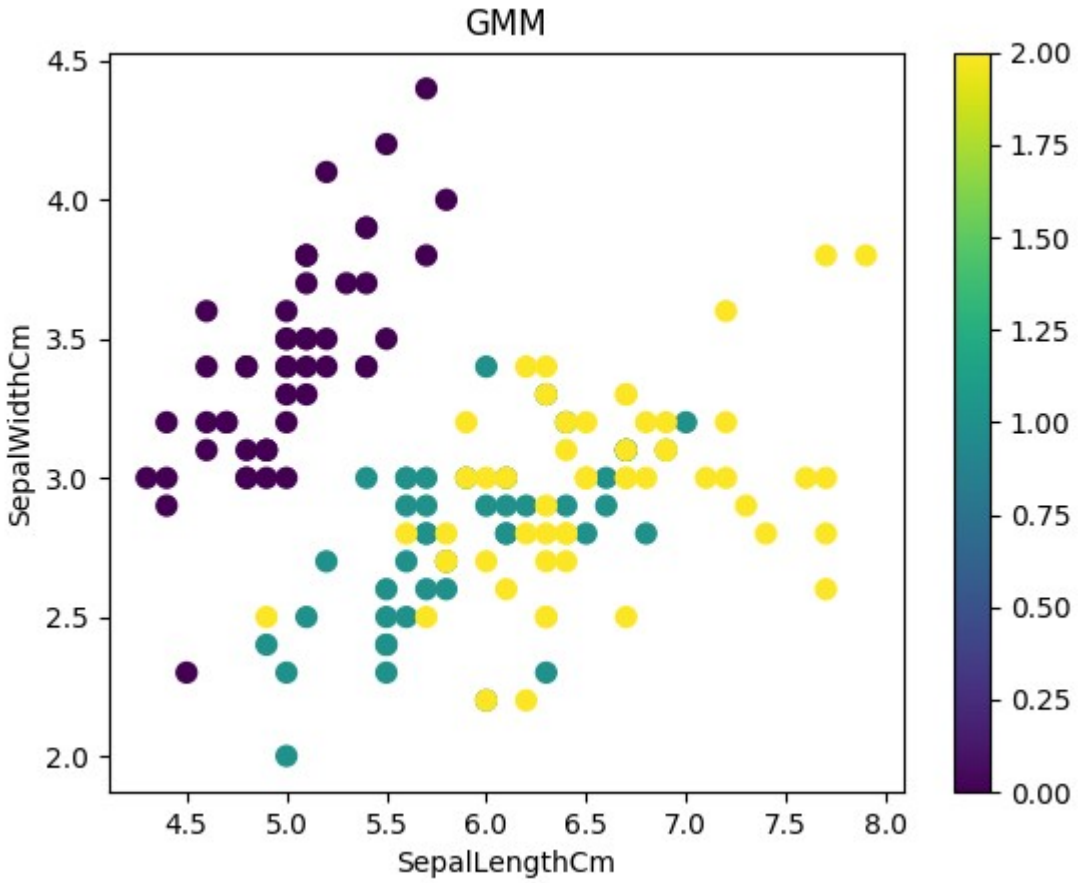
The result is very similar to the previous one. In this case, after comparing our output data with the expected output we also get error rate of 0.106.

5.3 Affinity Propagation



This algorithm differs from 2 previous because we didn't pass number of expected classes as its argument. It decides on that number itself. In out case it didn't make the best decision, because it divided our dataset to 6 classes. In that case I am not even counting the error rate, it is obviously the biggest.

5.4 Guassian Mixture Modelling



The GMM algorithm gives as the best result, its error rate is only 0.046.

6. Conclusion

As we can notice from above, the best algorithm for our dataset case was Guassian Mixture Modeling. Our data didn't have to many features and we just had 150 objects. We shouldn't conclude that this algorithm will always perform the best, but for our example it won.

References:

https://en.wikipedia.org/wiki/Cluster_analysis
https://www.kaggle.com/dhanyajothimani/basic-visualization-and-clustering-in-python
https://www.learndatasci.com/tutorials/k-means-clustering-algorithms-python-intro/
https://en.wikipedia.org/wiki/Affinity_propagation