Assignment 3

Clustering: practical issues (Python version)

The objective of this assignment is to address some typical problems related to the practical usage of clustering algorithms. We will consider again Iris data, but this time there are no experts available to label them. Therefore you want to use an automated approach to assign the flowers to different groups based on their characteristics.

TASK 0: warm up) Read the iris_clusters.csv data. This file has been sampled (it contains 300 observations) and does not contain any label information. You can use import pandas as pd and data = pd.read_csv("iris_clusters.csv", delimiter=";").

TASK 1: k-Means clustering) You can start the analysis with the simplifying assumption that you know that there are three species present in the data. Knowing the number of clusters, you can thus try clustering your flowers using k-Means with k=3. For this, you can use the scikit-learn package and the commands kmeans=KMeans(n_clusters=3) and kmeans.fit(data).

Make sure to look at the documentation on page https://scikit-learn.org/stable/modules/clustering.html#k-means, as some functions provided will prove useful for tasks later on. This implementation of k-means assumes Euclidean distance as the (dis)similarity measure, but there are other implementations in libraries such as pyclustering that allow the usage of other distance functions. Keep this in mind, particularly if you want to use the same library for the optional Task 6.

By accessing kmeans.labels_ you can see the cluster to which each flower has been assigned, and the cluster centroids' coordinates at kmeans.cluster_centers_. To see how the data has been assigned, you can simply plot two of the data features, using the labels as the color hue. You can do this using matplotlib (or seaborn, if you are familiar with it from the labs), for example:

```
import matplotlib.pyplot as plt

plt.scatter(data["sw"], data["sl"], c=kmeans.labels_, cmap='viridis')

plt.show()
```

Do the identified clusters correspond to the ones you expected to find, that is, the three Iris species you know from the lectures and from the first lab?

TASK 2: preprocessing) To improve the analysis, try to perform some data preprocessing before applying the k-Means operator. More specifically:

1) A min-max rescaling, scaling all the regular attributes to the interval [0, 1]. You can import sklearn.preprocessing and use normalize() for this.

2) Detect Outliers. Here you can use the simple operator based on Distances, inspecting the 10 nearest neighbors of each observation and marking 5 of them as

outliers. Here you can use various outlier detection methods here from the first lab, e.g. Tukey Test (scipy), Local Outlier Factor (sklearn)…

3) Filter out the identified outliers.

Is it better to rescale before or after detecting and filtering out the outliers? (If you are unsure, try both and look at the statistics of the data, in particular the min, max and average values in each column.)

Use again a scatterplot chart to visualize the clusters, coloring the observations by cluster id. Do the identified clusters correspond to the ones you expected to find, that is, the three Iris species? How many records have been included in each cluster? What are the coordinates of the three centroids representing the three clusters?

TASK 3: choice of k and internal evaluation) You will now put yourselves in the more realistic scenario where you do not know the number of clusters in advance. In this assignment we want to inspect the performance of the algorithm for each value of K. Therefore, you can write a loop to try every value of K from 2 to 10 (do not use k=1, which would be useless).

As you have no labels to check how well k-Means is performing, you need to use an *internal* (i.e., only based on your data) measure computing the cohesion and separation of your clusters. Here, you can use the Davies-Bouldin index. This index has a lower value when the clusters are better separated. For this, you can use sklearn.metrics.davies_bouldin_score(). What is the value of K leading to the most compact and separated clusters? Set that parameter as K.

TASK 4: Hierarchical clustering) Now you should try to use a different approach, in particular agglomerative hierarchical clustering. Such an implementation is also found in sklearn, under AgglomerativeClustering. Have a look at the documentation here: https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

You can start using the SingleLink method, which implements the approach you have seen during the lectures applying the MIN cluster similarity function. You can change between the Single-, Average- and CompleteLink methods by setting the parameter linkage as either single, average, or complete respectively: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering

Apart from the root of the dendrogram, trivially containing all the observations, how many records have been included in each of the two top (largest) clusters?

You can follow the instructions here to have the dendrogram generated visualized: https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html

TASK 5: DB-Scan) As a last algorithm, you should try using DB-SCAN. The model is applied similarly to the previous algorithms on sklearn. You should be able to find more about DBSCAN here: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

1) How many clusters does DB-SCAN find using eps=1, min_samples=5? (Keep in mind that the default option for eps=0.5, so you will need to set the parameter manually when creating the DBSCAN object.)
2) Leaving min_samples unchanged (5), can you manually find a value for epsilon leading to two clusters (plus noise)?

Now we can try to (semi-)automatically find a good value for the parameter epsilon. In particular, as seen during the lectures, you will use the method to estimate epsilon based on the distance to the N_{th} nearest neighbors. To estimate epsilon automatically, you need to create the k-distances plot. For that purpose, a snippet of code is provided at a3-kdist.py. Paste this code into your process to generate the plot.

You should notice some points of discontinuity in the curve, that is, points where the curve has a less regular shape, like a sudden increase/decrease of the distance: think of the effect of setting an epsilon threshold corresponding to those points, and check whether your intuition is correct by running DB-Scan with that epsilon.

[OPTIONAL] TASK 6: Document clustering) This last task is intended to be performed independently, so we only provide limited instructions. The knowledge acquired during the lectures and from the assignments should be sufficient for you to set up this process from the beginning to the end.

You will use the data *bbc.zip*, containing five folders (business, tech, sport, ...), each containing several news articles about the same topic of the folder. The objective of the task is to see what clusters you would obtain by applying k-means to a vectorial representation of these documents. You can use k=5 (and can also try other values of k, or try to find the best value for k, if you have time – this is not requested), but of course you should not let the clustering algorithm know about the topics.

As this task is optional (and to be done independently), feel free to use libraries of your choice for text preprocessing and clustering. As a starting point, we suggest looking at text preprocessing libraries like nltk or gensim, offering, among others, algorithms for stopword removal, stemming and tokenization.

CONCLUSION) With this and the previous assignments you have used a number of methods to preprocess the data and several data mining algorithms. We have tried to give you a “well guided” plan to emphasize the features of these approaches.

However, remember that when you apply these methods to new data you will need to take each of the steps that you have taken during these assignments and think if it is appropriate for the problem at hand. Here are a few additional questions, so that you can reflect about what you have done. You can think about them yourself, and also discuss them with your course and group mates.

1) In which of the tasks performed during this assignment would dimensionality reduction techniques be useful?

2) What would be the consequence of not removing outliers before using DB-SCAN?

3) Although we know that there are three clusters in the data, the best clustering seem to contain only two groups of records. What does this mean? Is it our assumption that there are three clusters wrong? Or are the clustering algorithms not working well? Or, do you have any other explanation for this sub-optimal behavior (if you think it is sub-optimal)?