Clustering

# 1 Fundamentals

1. Clustering: getting started
   (a) Download the synthetic clustering datasets (fruits) from the class webpage. Load the first one.
   (b) Using sklearn library, apply `KMeans` algorithm on the dataframe, with 2 clusters, on the numerical columns. You can check the documentations to see how to do it, it should be something like `clusters = KMeans(n_clusters=2).fit_predict(df[["weight","diameter"]])`.
   (c) Add the clusters found as a "cluster" column to the dataframe
   (d) To observe the clusters, we can plot (with `seaborn` library for instance, `sns.scatterplot`) with dot colors corresponding to clusters (hue="cluster"). Compare with using the `fruit` information as colors (ground truth clusters). You should see that the clusters are not the expected ones. Guess why.
2. Limits of k-means
   (a) Normalize the data, and retry on the first dataset. It should solve the problem.
   (b) Do the same with the second example. The result should not be as expected. Do you understand why?
   (c) Try to solve the issue using Gaussian Mixture (class `BayesianGaussianMixture`. Check the `covariance_type` parameter.
   (d) Also try the DBscan approach
   (e) Do the same (comparing ground truth, k-means, GM, DBScan) on the other examples. Every time, try to understand why each of the method succeed or fail. Try to play with the parameters to make the methods succeed.
3. Intuition on Generative Models

   We have seen that gaussian mixtures are powerful models. We will code a naive version of it to get an intuition of how it works.
   (a) We will first generate some realistic data for height of individuals. We assume that a population is composed of two types of people (e.g., men/women) with 2 distinct properties. Group 1: height is distributed with mean=160cm and std=5, while for Group2, mean=175,std=8. Generate a dataset, for instance with `np.random.normal`, and 50 individuals from each group.
   (b) Write a function that compute the **pdf** (probability density function) of a data point given a gaussian distribution defined by its mean and std. This is direclty obtained using `scipy.norm.pdf`. This value can be interpreted as how likely it is to obtain an observed point, given the gaussian generator.
   (c) Write a function computing the mixture probability of generating a point given 2 possible generators. If both generators contribute equally, this is $0.5 * p1 + 0.5 * p2$. $p1$ and $p2$ are computing using the function written above
   (d) Write a function computing the likelihood of generating a set of points given two normal distributions, i.e., mean1, mean2, std1,std2. The likelihood of a set of points is defined as the product of the probability of generating each of the point, i.e., $p_1 * p_2 * ... * p_i$.
   (e) First, let us assume that there is a single point generator, i.e., a single normal law. Explore the space of parameters (mean, std) with a for loop, and find the mean and std maximizing the likelihood.
   (f) Check that the parameters found correspond to the mean and std of the dataset. They are thus different from the *real* model used to generate the data.

(g) Do the same parameter exploration but assuming two generators. (Be careful, it might take some time! Limit the parameter space)

(h) Check that the parameters you found are good matches for the real parameters you used to generate the data.

4. Interpreting clusters

   An important part of using clustering is to make sense of the clusters obtained.

   (a) Download the cleaned toy car dataset from the class website. Keep only the numerical values (e.g., `df.select_dtypes(include='number')`

   (b) Use k-means with 3 clusters.

   (c) Compute the centroid (mean values for each feature), and the size for each cluster. A flexible way to proceed is to extract the clusters ( `fit_predict` ), add the resulting list as a new column (e.g., "cluster") in a copy of the feature dataframe, then compute statistics by cluster in that dataframe, for instance with `.groupby("cluster").agg(['mean',"count"])`

   (d) If you had to give a manual label to those clusters, to describe the cars they contain, what would it be ? (e.g.: "large and old expensive cars"...)

   (e) Check the difference with and without normalization

5. Evaluation and number of clusters

   (a) Compute the silhouette score using method `silhouette_visualizer` from package `yellowbrick`, plot the silhouette score and interpret it.

   (b) We would like to find the optimal number of clusters. Apply the silhouette score method: plot the relation between $k$ and the silhouette score, and search for a maximum value. You can use the `kelbow_visualizer`, with the `metric` option set to silhouete. (By curiosity, you can check other cluster evaluation metrics)

# 2 Going Further

(a) Using this knowledge, explore the proposed Wine dataset: https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering