Setup

Compared to the previous exercise sheets, you should install `pydataset`. If you have used the `requirements.txt` from ILIAS to set up your environment, you already have this dependency available. We will also work with `matplotlib`, `numpy`, `pandas`, `scikit-learn`, and `seaborn` again.

Task: Clustering and Outlier Detection

The aim of this exercise is to apply clustering and outlier-detection approaches. We work with the `faithful` dataset, which you can obtain from package `pydataset` with `data('faithful')`. The dataset contains eruption times and (between-eruption) waiting times for the geyser Old Faithful.

a) **[Visualization]** Load the data and normalize it with the help of `scale()` from `sklearn.preprocessing`. Create a scatter plot of the data points and create histograms of their distribution. In particular, also use two-dimensional histograms, which are available in `matplotlib` as well as `seaborn`.
Where are potential clusters? Which points might be outliers? Why have we normalized the data beforehand?

b) **[K-Means]** Train an instance of `KMeans` from `sklearn.cluster`. Visualize the results (e.g., by coloring the points according to their cluster membership in a scatter plot) for different values of k. Use `silhouette_score()` from `sklearn.metrics` to assess the quality of the k-means results.
Why does k-means produce these results? Is the silhouette coefficient a useful metric to assess clustering quality here?

c) **[DBSCAN]** Train an instance of `DBSCAN` from `sklearn.cluster` and visualize the result as before.
Why do you obtain such a result? Can you leverage the silhouette coefficient to search for hyperparameter values that yield a better clustering result?

d) **[OPTICS]** Train an instance of `OPTICS` from `sklearn.cluster` and create a reachability plot based on properties of the trained instance.
Can you use this plot to determine a good value for DBSCAN's hyperparameter ϵ?

e) **[Outlier Detection]** Detect outliers in the data with the help of `NearestNeighbors` from `sklearn.neighbors`, a distance-based approach, and `LocalOutlierFactor` from `sklearn.neighbors`, a density-based approach. Visualize the data points together with their outlier scores, e.g., by scaling the size and/or color of the points in a scatter plot accordingly.
For which data points do the results of the two approaches differ strongly, and why?

f) **[Subspace Outliers]** Calculate the outlier scores for each feature of the dataset separately. Compare these results to your previous two-dimensional approach.
Are there points that only appear to be outliers in one subspace? Are there non-trivial outliers in the data?