PAM: introduction

The Partitioning Around Medoids (PAM) algorithm (also known as k-medoids) is a clustering technique aiming at minimizing the sum of dissimilarities between points in a dataset and the nearest medoid, where a medoid is the most centrally located point in a cluster. The main objective function is to minimize the total cost:

$$\sum_{i=1}^{n} \min_{j \in \{1,2,\ldots,k\}} d(x_i, m_j),$$

where $d(x_i, m_j)$ is the dissimilarity between point x_i and medoid m_j, and k is the number of clusters which has to be assessed beforehand. PAM is a better alternative than k-means when dealing with non-Euclidean distances or datasets with noise and outliers, as it is more robust to such irregularities.

'iris' dataset

iris: dataset of 150 observations x 5 variables.

Sepal.Length: continuous variable
Sepal.Width: continuous variable
Petal.Length: continuous variables
Petal.Width: continuous variable
Species.: discrete variables, setosa, virginica, versicolor

```
# load dataset
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)
0 5.1 3.5 1.4 0.2
1 4.9 3.0 1.4 0.2
2 4.7 3.2 1.3 0.2
3 4.6 3.1 1.5 0.2
4 5.0 3.6 1.4 0.2
```

Summary

In order to perform a PAM, we call the function 'KMedoids()' from $'sklearn\_extra.cluster'$, containing three essential arguments: 'method', 'metric' and 'n_clusters'.

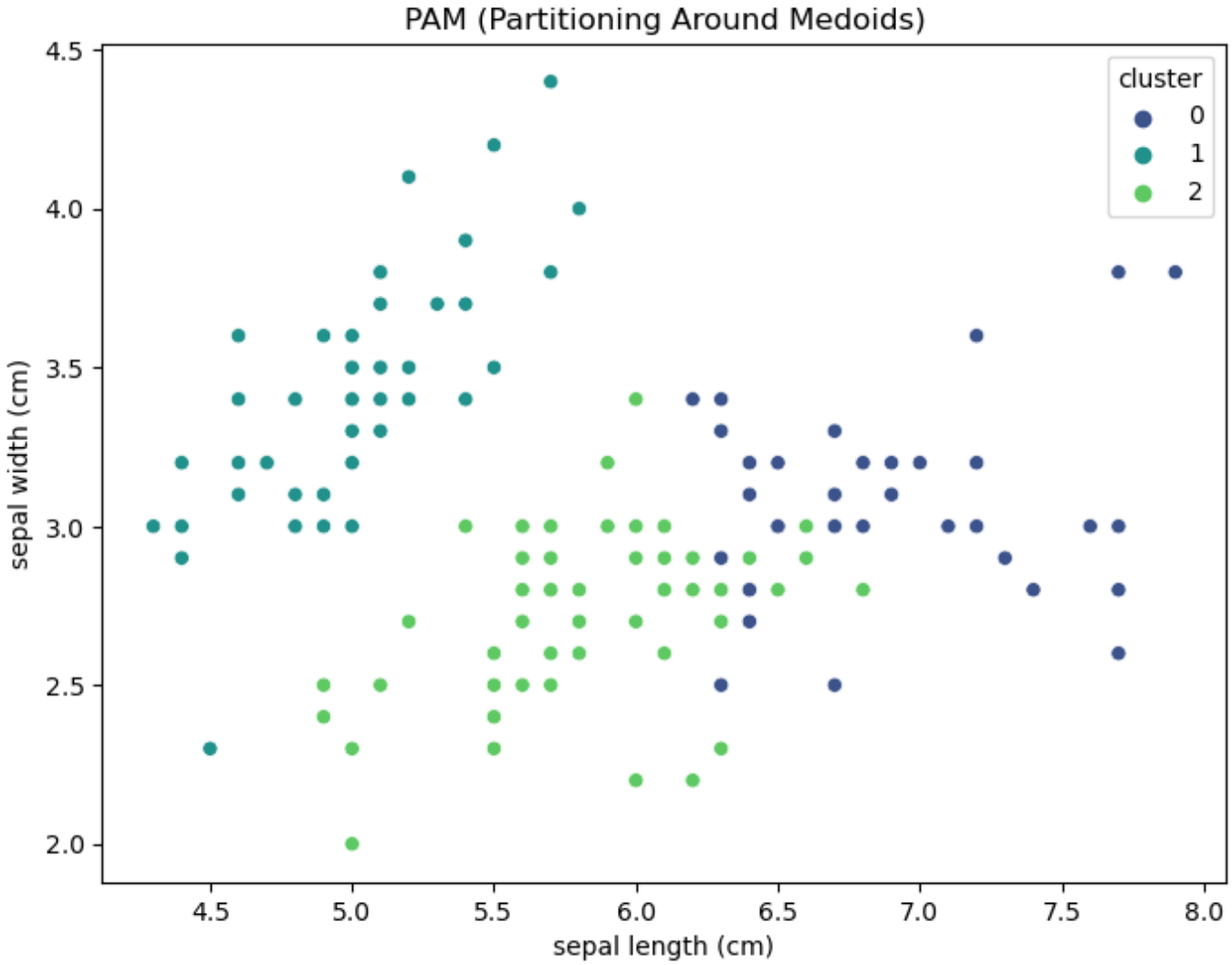
```
# Calculate pairwise distances
metric = 'manhattan'
dissimilarity_matrix = pairwise_distances(iris_df, metric='manhattan')

# Perform PAM clustering using KMedoids from sklearn_extra
pam = KMedoids(init='random', method='pam', metric='precomputed', n_clusters=3,
          random_state=2024)
pam.fit(dissimilarity_matrix)

# Extract medoids and clustering results
medoids = pam.cluster_centers_
clustering_results = pam.labels_
clustering_results
```

Visualizing PAM clusters

On Iris dataset, using Manhattan distance metric



Main observations

- The PAM algorithm successfully grouped the Iris dataset into three clusters, with each cluster roughly corresponding to one of the true species.
- Some overlap is observed between the clusters, particularly between the clusters for versicolor and virginica, indicating difficulty in perfectly separating these two species.
- The comparison of true species labels with PAM clusters reveals that the algorithm captures the overall structure of the dataset but may misclassify some points, especially where species characteristics are similar.

References

https://www.datanovia.com/en/lessons/k-medoids-in-r-algorithm-and-practical-examples/

Python
https://www.python.org/