Rapport TP 1 (Clustering) | Fouilles de données et apprentissage artificiel

Saidane Wassim

Septembre 2023

Table des matières

1 Normalisation des données

1.1 Création de la matrice X

J'ai décidé de créer un CSV pour la matrice X, pour pouvoir la modifier aisèment selon les usages.
Dans mon code une fonction permet de passer d'un CSV à une matrice numpy, j'ai également implémenté une fonction permettant de passer d'une matrice numpy à une matrice CSV.

1.2 Moyenne et variance de la matrice X

Pour cacluler la moyenne et la variance de la matrice X, j'ai utilisé les fonctions numpy.mean et numpy.var.
On trouve :
$\bar{X} = 0.4444444444444444$
$\sigma^2 = 1.1358024691358024$
On a besoin de centré-réduire la matrice X pour pouvoir appliquer l'algorithme K-means plus tard. Cette normalisation nous permettra d'avoir des données nons biasées par la suite.

1.3 Normalisation de la matrice X centrée-réduite

Pour normalisé la matrice X, la formule est la suivante :
$X_{norm} = \frac{X - \bar{X}}{\sigma}$
En python, j'ai utilisé la fonction scale de sklearn.preprocessing.
Cepandant cette fonctione ne prend pas en charge les np.matrix, je suis donc passé par des np.array.
Le résultat obtenu est le suivant :

$$x_{\text{normalisée}} = \begin{pmatrix} 0.0 & -1.224744871391589 & 1.336306209562122 \\ 1.224744871391589 & 0.0 & -0.2672612419124244 \\ -1.224744871391589 & 1.224744871391589 & -1.0690449676496976 \end{pmatrix} \tag{1}$$

La matrice ci-dessus représente les données normalisées, où chaque ligne correspond à un échantillon et chaque colonne à un attribut des données. La normalisation garantit que chaque attribut a une moyenne de zéro et un écart type de un. Cette transformation rend les attributs comparables entre eux, indépendamment de leur échelle ou de leur unité d'origine.

— **Ligne 1 :** Ses valeurs pour les trois attributs indiquent qu'il est à la moyenne pour le premier attribut, en dessous de la moyenne pour le deuxième attribut et au-dessus de la moyenne pour le troisième attribut.

— **Ligne 2 :** Il présente une valeur supérieure à la moyenne pour le premier attribut, une valeur moyenne pour le second attribut et une valeur légèrement inférieure à la moyenne pour le troisième attribut.
— **Ligne 3 :** Ses valeurs sont inférieures à la moyenne pour le premier et le troisième attributs, et supérieures à la moyenne pour le second attribut.

1.4 Moyenne et variance de la matrice X normalisée

On obtient les valeurs suivantes :
$\bar{X} = 4.9343245538895844e-17 \approx 0$
$\sigma^2 = 1.0$

La matrice a été correctement normalisée.

2 Normalisation Min-Max

2.1 Création de la matrice X2

J'ai créé une copie de la matrice X, que j'ai nommé X2.

2.2 Moyenne et variance de la matrice X2

La matrice est la même que la matrice X, donc on obtient les mêmes valeurs pour la moyenne et la variance.

2.3 Normalisation de la matrice X2 minmax

Pour normaliser la matrice X2, la formule est la suivante :
$X_{norm} = \frac{X - min(X)}{max(X) - min(X)}$
En python, j'ai utilisé la fonction MinMaxScaler de sklearn.preprocessing.

$$x_{\text{normalisée}} = \begin{pmatrix} 0.5 & 0.0 & 1.0 \\ 1.0 & 0.5 & 0.3333333333333333 \\ 0.0 & 1.0 & 0.0 \end{pmatrix} \tag{2}$$

— **Ligne 1** : Il est à la moyenne pour le premier attribut, en dessous de la moyenne pour le deuxième attribut, et au-dessus de la moyenne pour le troisième attribut.
— **Ligne 2** : Il présente une valeur supérieure à la moyenne pour le premier attribut, une valeur moyenne pour le second attribut, et une valeur légèrement inférieure à la moyenne pour le troisième attribut.
— **Ligne 3** : Ses valeurs sont inférieures à la moyenne pour le premier et le troisième attributs, et supérieures à la moyenne pour le second attribut.

2.4 Moyenne et variance de la matrice X2 normalisée

On obtient les valeurs suivantes :
$\bar{X} = 0.48148148148148145$
$\sigma^2 = 0.16941015089163236$
Contrairement à la méthode précédente, l'objectif n'est pas d'obtenir une moyenne de 0 et une variance de 1 mais plutôt de ramener les valeurs entre 0 et 1.

Comparaison entre la Normalisation Min-Max et la Standardisation

$$\text{Normalisation Min-Max} : x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$\text{Standardisation} : x' = \frac{x - \bar{x}}{\sigma}$$

	Normalisation Min-Max	**Standardisation**
Intervalle	[0, 1]	Non défini
Sensibilité aux outliers	Oui	Moins que Min-Max
Centrage	Non	Oui
Utilisation	Images, réseaux de neurones	Algorithme d'optimisation, PCA
Interprétation	Relatif aux valeurs min/max	Relatif à la moyenne

TABLE 1 – Tableau comparatif des méthodes de mise à l'échelle.

3 Analyse des Iris de Fisher avec l'algorithme K-means

3.1 Lecture du fichier data

J'ai créer une fonction me permettant de lire les datas en prenant compte des headers et qui me retourne un dataframe pandas.

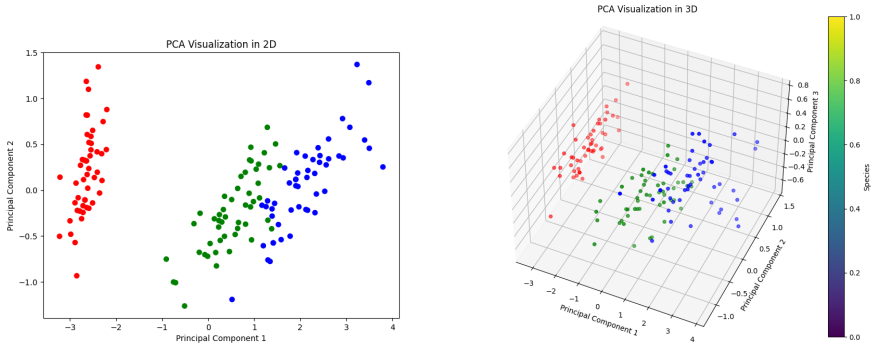
3.2 Séparation du label et des données

J'ai utilisé la fonction pop de pandas pour séparer le label des données.

3.3 Décomposition PCA

L'analyse en composantes principales (PCA) est une technique utilisée pour réduire la dimensionnalité des données tout en préservant autant d'informations que possible. Elle est principalement utilisée pour explorer et visualiser des données complexes, ainsi que pour prétraiter des données avant des analyses plus avancées.

En python j'ai utilisé la fonction PCA de sklearn.decomposition.
Pour pouvoir visualiser les données, j'ai afficher une PCA en 2D et une PCA en 3D.



(a) PCA 2D (b) PCA 3D

FIGURE 1 – Visualisations PCA 2D et PCA 3D

3.4 K-means à 3 clusters

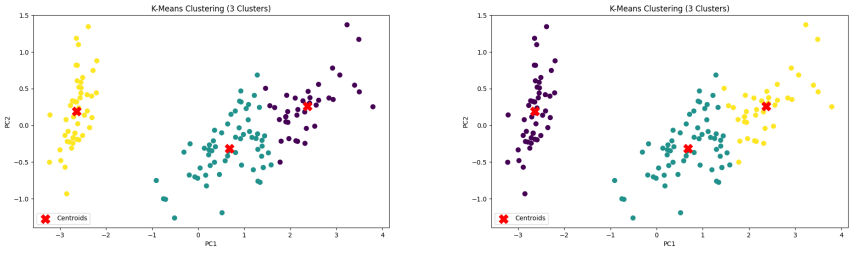
Par rapport à l'énoncé j'ai considéré que data est le de données "iris" et que "x" était le résultat de la PCA.
Pour K-means j'ai créer une fonction qui utilise la fonction KMeans de sklearn.cluster.
J'obtiens le nuage de points suivant :

FIGURE 2 – K-means à 3 clusters

trois clusters distincts issus de l'algorithme k-means sont visibles, représentant les trois espèces d'iris. Le cluster à gauche est le plus dense et compact, suggérant une grande similarité entre ses échantillons. Les deux autres clusters montrent une dispersion plus grande, indiquant une variabilité légèrement supérieure. Les frontières entre ces clusters ne sont pas nettement définies, typique de k-means qui base ses délimitations sur la distance aux centroïdes.

3.5 Répitition de K-means à 3 clusters

En répétent l'algorithme K-means 6 fois, on obtient les résultats suivants :



(a) K-means à 3 clusters (b) K-means à 3 clusters

FIGURE 3 – Visualisations K-means à 3 clusters

Ici, j'ai mis deux itérations car je retrouve à chaque fois les mêmes centroïdes. Je pense que c'est dû à ma façon d'itérer sur les données mais je n'ai pas réussi à coriger celà.

3.6 Projection des labels sur les clusters

A partir de maintenant et pour des questions de lisibilité, j'ai codé sur le programme 3.6-4.py La projection des labels sur les clusters est la suivante :

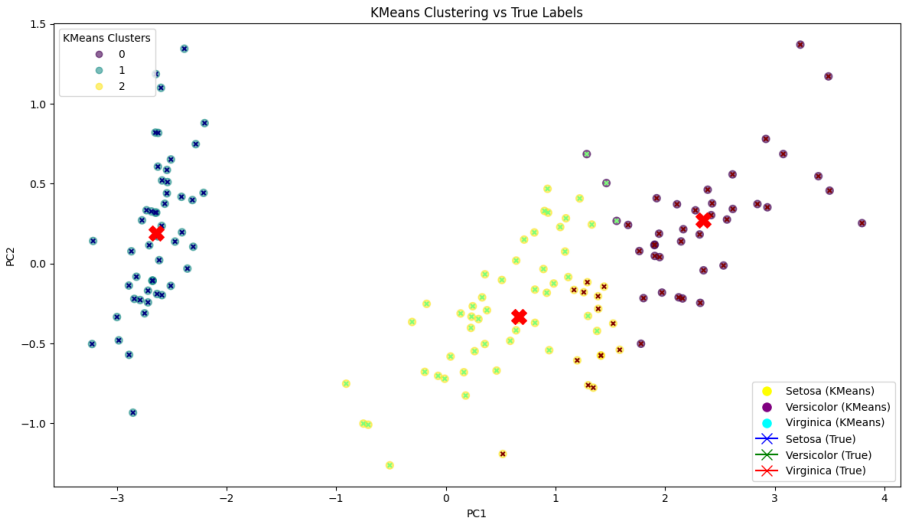


FIGURE 4 – Projection des labels sur les clusters

3.7 Tableau de contingence

Cluster	setosa	versicolor	virginica
0	50	0	0
1	0	47	14
2	0	3	36

TABLE 2 – Tableau de contingence comparant les résultats de clustering aux labels théoriques.

L'analyse du tableau de contingence pour le clustering KMeans sur les données Iris révèle que l'espèce Setosa est parfaitement distinguée des autres, étant entièrement contenue dans le cluster 0. Cependant, bien que le cluster 1 soit dominé par l'espèce Versicolor, il contient également quelques observations de Virginica, indiquant une certaine superposition des caractéristiques entre ces deux espèces. De même, le cluster 2, bien que principalement composé de Virginica, contient quelques fleurs Versicolor. Cette superposition suggère que Versicolor et Virginica ne sont pas aussi nettement séparées dans l'espace des caractéristiques que Setosa par rapport aux autres.

3.8 Indice de Silhouette

L'indice de Silhouette varie entre -1 et 1, où une valeur proche de 1 indique que les points sont bien appariés à leur propre cluster et sont éloignés des autres clusters. Une valeur de 0.6 est considérée comme modérément bonne et confirme la présence de chevauchement.

3.9 Données originales

Pour les données originales, on obtient les résultats suivants :

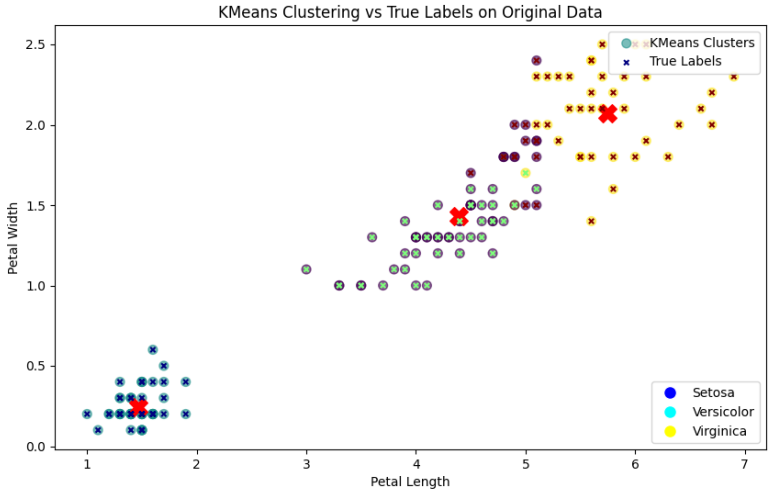


FIGURE 5 – Projection des labels sur les clusters

Cluster	setosa	versicolor	virginica
0	0	48	14
1	50	0	0
2	0	2	36

TABLE 3 – Tableau de contingence comparant les résultats de clustering aux labels théoriques.

Indice de Silhouette : 0.55
Le clustering KMeans sur les données originales et projetées d'Iris a produit des résultats similaires, bien que l'indice de Silhouette soit légèrement plus faible pour les données originales. Cela suggère que les deux méthodes capturent efficacement les structures sous-jacentes des données. L'utilisation des données

originales offre une meilleure interprétabilité, car les dimensions ont une signification claire. Cependant, la projection via PCA peut améliorer l'efficacité du clustering en réduisant la dimensionnalité et en éliminant le bruit. Dans ce cas, les différences entre les deux approches ne sont pas significatives, mais le choix dépendra de la nature des données et des objectifs de l'analyse.

4 Clustering hiérarchique sur les données Iris

4.1 AgglomerativeClustering

Cette fonction permet de faire du clustering hiérarchique.
Voici ce qu'on obtient par exemple avec le sépale :

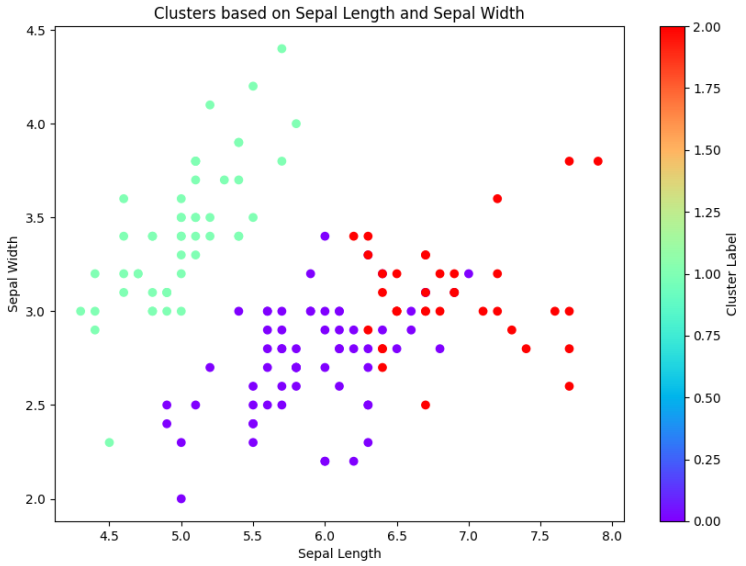


FIGURE 6 – Clustering hiérarchique sur les données Iris

4.2 Dendrogramme complet

J'ai effectué le dendrogramme grace à la fonction dendrogram de scipy.cluster.hierarchy.
J'obtiens :

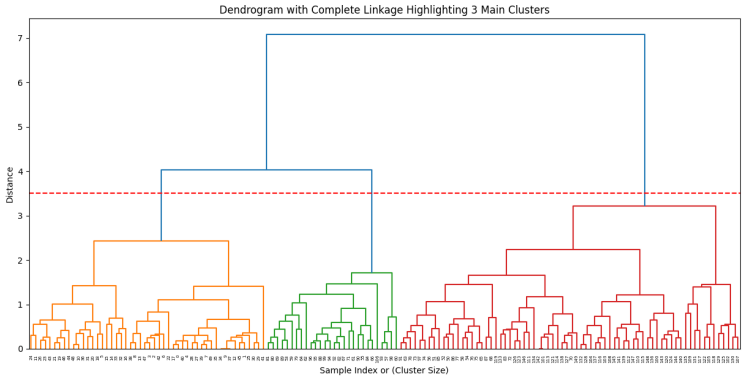


FIGURE 7 – Dendrogramme complet

Les fusions hiérarchiques des échantillons du jeu de données Iris en utilisant un linkage complet. Les points en bas représentent les échantillons individuels, et la hauteur à laquelle les branches se rejoignent indique la distance de fusion. En traçant une ligne horizontale à une hauteur spécifique, on peut déterminer le nombre de clusters principaux à cette distance. Dans le contexte d'Iris, une hauteur qui divise les données en trois branches principales pourrait correspondre aux trois classes connues : setosa, versicolor et virginica.

4.3 Coupe en 3 clusters

J'ai utilisé la fonction fcluster de scipy.cluster.hierarchy pour couper le dendrogramme en 3 clusters.
J'obtiens :

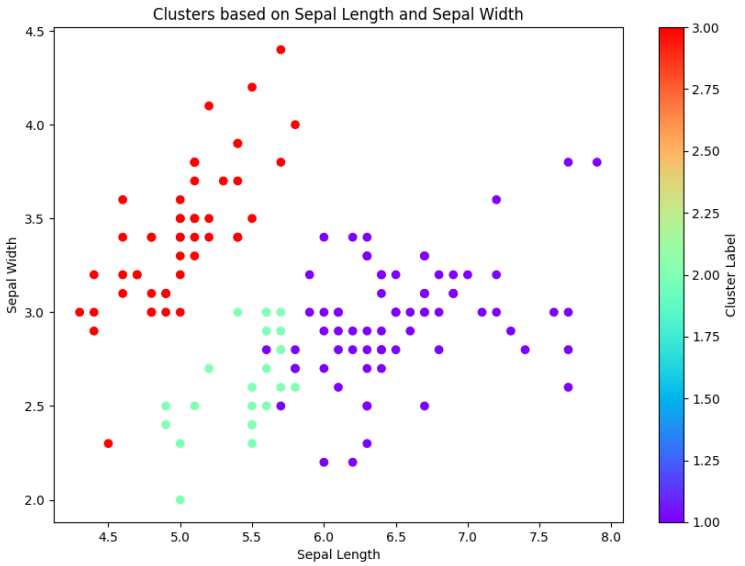


FIGURE 8 – Dendrogramme complet

Le graphique montre clairement une partition des données en trois clusters distincts, basée sur les caractéristiques de la longueur et de la largeur du sépale. Chaque couleur représente un des trois clusters obtenus grâce au clustering hiérarchique ascendant avec un linkage complet. Le clustering semble avoir bien séparé une des classes d'iris (probablement "setosa" à en juger par sa séparation nette) des deux autres. Cependant, les deux autres classes (probablement "versicolor" et "virginica") montrent une certaine superposition, ce qui est cohérent avec le fait que ces deux classes sont plus difficiles à distinguer uniquement sur la base des caractéristiques du sépale.

4.4 Tableau de contingence

	Predicted		
Actual	1	2	3
Setosa	0	0	50
Versicolor	23	27	0
Virginica	49	1	0

Les résultats du tableau de contingence montrent que le clustering hiérarchique ascendant a parfaitement distingué la classe Setosa des deux autres, en

l'assignant entièrement au cluster 3. Cependant, il y a des confusions entre Versicolor et Virginica. La majorité de Versicolor est répartie entre les clusters 1 et 2, tandis que la majorité de Virginica est assignée au cluster 1. Cette confusion entre Versicolor et Virginica est cohérente avec l'idée que ces deux classes sont plus similaires en termes de caractéristiques du sépale par rapport à Setosa.

4.5 Linkage moyen

Pour le linkage moyen on obtient le dendrogramme suivant :

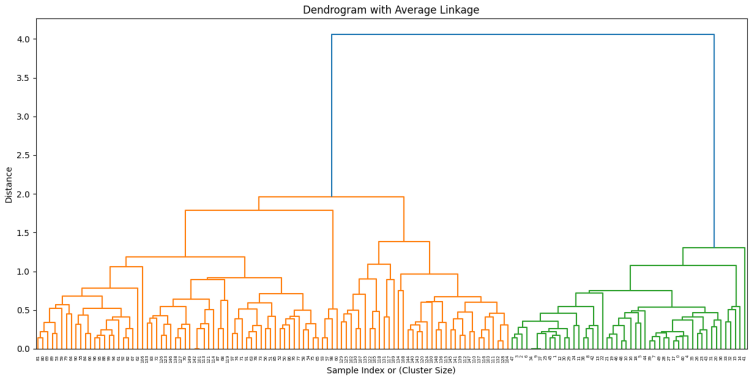


FIGURE 9 – Dendrogramme moyen

	Predicted			
Actual	1	2	3	4
Setosa	50	0	0	0
Versicolor	0	0	4	46
Virginica	0	36	0	14

Les données sont plus dispérsées avec le linkage moyen qui donne 4 clusters. Le linkage complet semble plus adapté pour les données Iris.

4.6 Indice de Silhouette

L'indice de Silhouette est de 0.5 pour le linkage complet et de 0.47 pour le linkage moyen.
Ce qui confirme que le linkage complet est plus adapté pour les données Iris.

5 Nombre optimale de clusters sur les données atmosphère d'exoplanètes

5.1 Nettoyage des données

5.2 Indice de Calinski-Harabasz et Davies-Bouldin

1. **Indice de Calinski-Harabasz (CH)** : L'indice de Calinski-Harabasz est défini comme le rapport de la dispersion entre les clusters à la dispersion intra-cluster, donné par :

$$\mathrm{CH}(k) = \frac{\mathrm{Trace}(B_k)}{\mathrm{Trace}(W_k)} \times \frac{N-k}{k-1}$$

2. **Indice de Davies-Bouldin (DB)** : L'indice de Davies-Bouldin mesure la similitude moyenne entre chaque cluster et son cluster le plus similaire, où la similitude est le rapport de la distance intra-cluster à la distance inter-cluster, donné par :

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left(\frac{s_i + s_j}{d_{ij}} \right)$$

5.3 Méthode du coude

La méthode du coude est une technique utilisée pour trouver le nombre optimal de clusters en k-means clustering. Elle consiste à exécuter le clustering k-means pour une gamme de valeurs de k (par exemple, k de 1 à 10) et à calculer la somme des distances au carré de chaque point à son centre assigné pour chaque valeur de k. Ces distances intra-cluster sont ensuite tracées en fonction des valeurs de k. L'emplacement d'un "coude" dans le graphique représente un équilibre approprié entre la précision et la généralisation et est donc considéré comme une indication du nombre optimal de clusters.
On obtiens :

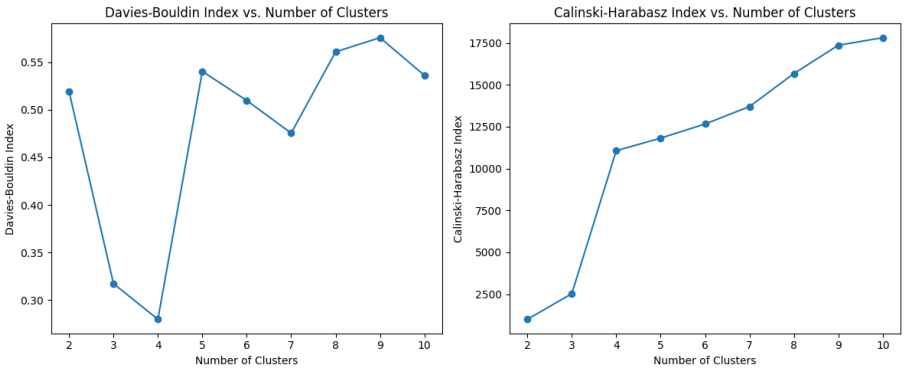


FIGURE 10 – Méthode du coude

Avec les deux indices on peut estimer le nombre de clusters optimal à 3 ou 4.

5.4 Résultats

Nous allons réaliser un clustering avec 3 et 4 clusters.

3 clusters

Nous obtenons le résultat suivant :

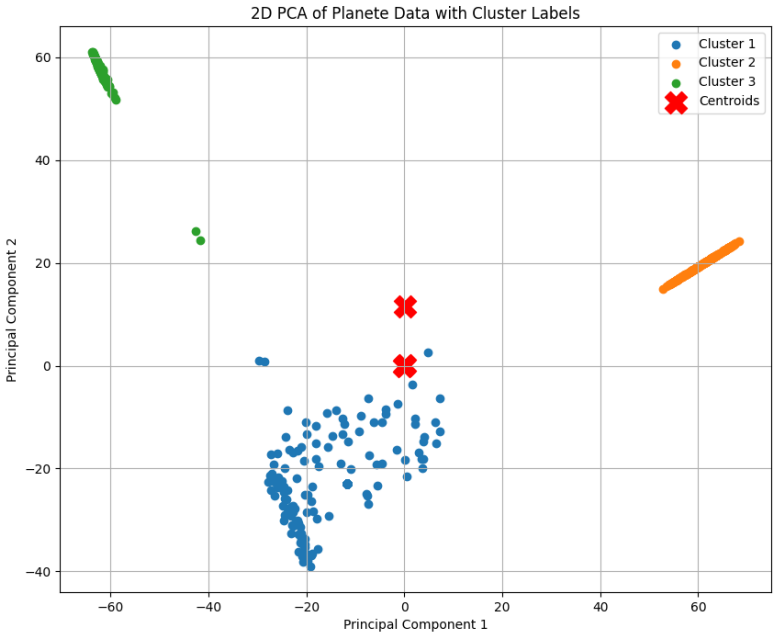


FIGURE 11 – Clustering à 3 clusters

Les clusters sont bien séparés, en revanche les centroïdes semblent bogués.

5.5 4 clusters

Nous obtenons le résultat suivant :

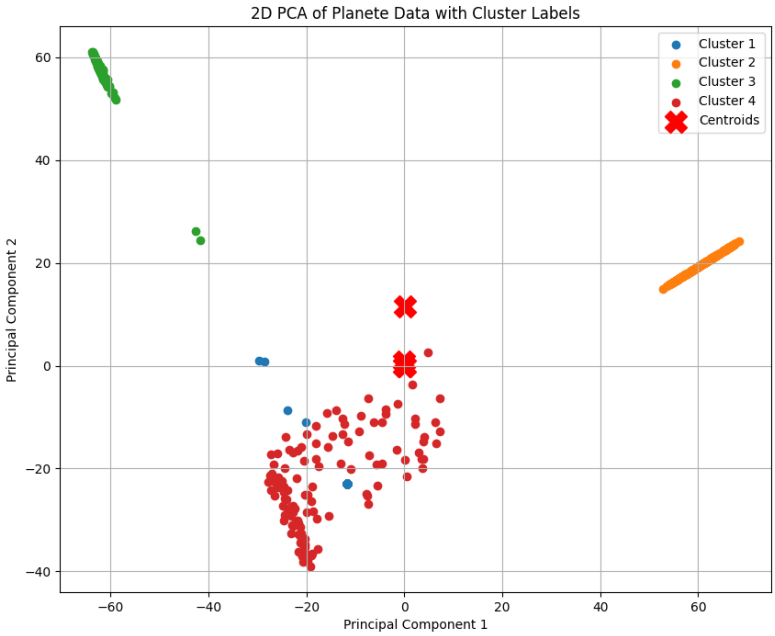


FIGURE 12 – Clustering à 4 clusters

Le résultat est clairement moins bien.