COMPTE RENDU TP CLUSTERING K-MEANS

Étudiant :
Sami EZZAHID
5A – IE – ILC 1

ESIREM – Polytech Dijon

ESIREM – Département Informatique/Electronique — Année 2023-2024

Table des matières

TABLE DES MATIERES

INTRODUCTION

1.1. INTRODUCTION

Introduction

Le présent rapport expose le travail réalisé dans le cadre du TP Projet du cours de Big Data à l'ESIREM, sous la supervision de Natalia Kharchenko et Sergey Kirgizov. Ce TP se concentre sur l'exploration des techniques de clustering et de visualisation de données, avec une attention particulière sur l'algorithme k-Means. Les objectifs du projet sont doubles : l'implémentation de l'algorithme k-Means et l'exploration d'un jeu de données du monde réel à l'aide de techniques de clustering et de réduction de la dimension.

Pour mener à bien le projet d'exploration des jeux de données, le présent rapport se focalisera sur le jeu de données "Toy datasets" de scikit-learn, une bibliothèque populaire en machine learning. Le processus d'analyse comprendra le traitement préliminaire des données, l'utilisation de l'algorithme k-Means avec différentes valeurs du paramètre k, le choix du meilleur k via la méthode du coude ou de la silhouette, et enfin, la présentation graphique des résultats obtenus.

Les technologies recommandées pour la réalisation de ce projet incluent Python, accompagné des bibliothèques telles que NumPy, scikit-learn, PCA (Principal component analysis) et Matplotlib. Les participants sont également encouragés à explorer d'autres langages de programmation ou technologies de visualisation selon leurs préférences et leurs compétences.

Dans la suite de ce rapport, nous détaillerons le processus d'exploration du jeu de données "Toy datasets" en suivant les étapes mentionnées dans l'énoncé du TP.

2 EXPLORATION DES JEUX DE DONNEES AVEC CLUSTERING

1.2. 1. TRAITEMENT PRELIMINAIRE

Le jeu de données "Toy datasets" de scikit-learn est un ensemble de données synthétiques généralement utilisé à des fins d'apprentissage et de prototypage. Il se compose de plusieurs sous-ensembles, chacun simulant des situations spécifiques et offrant des caractéristiques variées. Pour cet exemple, nous allons utiliser le sous-ensemble "make_blobs", qui génère des clusters de points bien séparés.

Propriétés du jeu de données "Toy datasets" - make_blobs :

- Caractéristiques (Features) : Chaque exemple dans le jeu de données synthétique généré par "make_blobs" possède un certain nombre de caractéristiques. Dans le cas de "make_blobs", ces caractéristiques sont souvent bidimensionnelles pour faciliter la visualisation, mais le nombre de caractéristiques est ajustable.

- Clusters (Centres) : Le jeu de données simule la présence de clusters définis par des centres (centroids). Ces centres sont générés aléatoirement et contrôlables par l'utilisateur.

- Déviation standard (Cluster_std) : Cette propriété contrôle la dispersion des points autour de chaque centre de cluster. Une valeur plus élevée entraîne une plus grande dispersion des points au sein de chaque cluster.

Conversions Possibles pour le Calcul de Distance :

- Standardisation des Caractéristiques : Le jeu de données soit synthétique et généralement bien équilibré, il peut être utile de standardiser les caractéristiques (z-score) si nécessaire, surtout si on utilise des algorithmes de clustering sensibles à l'échelle des variables.

- Calcul de Distance Euclidienne : Le "make_blobs" génère des clusters bien définis, la distance euclidienne est généralement suffisante pour mesurer la similarité entre les points. Aucune conversion particulière n'est nécessaire pour ce jeu de données, car il est conçu pour être relativement simple et compréhensible.

1.3. 2. ANALYSE DES DONNEES

1.3.1. TEST DE DIFFERENTS VALEURS DE K

Le code commence par utiliser le StandardScaler de scikit-learn pour standardiser les données (X) avant d'appliquer l'algorithme k-Means. La standardisation est importante pour les algorithmes basés sur la distance, comme k-Means, afin de garantir que toutes les caractéristiques contribuent de manière équitable au calcul de la distance.

```
14  # Chargement du jeu de données Iris
15  iris = load_iris()
16  X, y = iris.data, iris.target
17
18  # Standardisation des données
19  scaler = StandardScaler()
20  X_scaled = scaler.fit_transform(X)
```

Une liste de valeurs de k est définie (k_values), allant de 2 à 6 dans cet exemple. Ces valeurs représentent le nombre de clusters que l'algorithme k-Means va tenter de former.

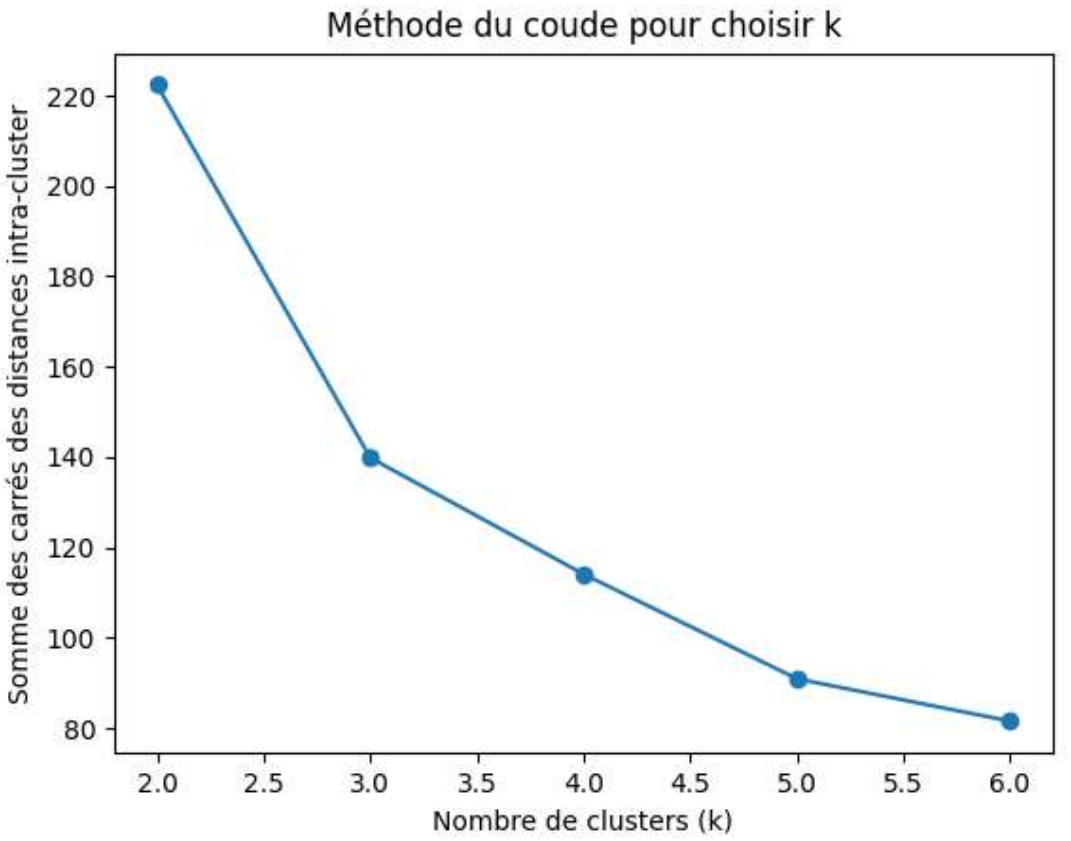
En fonction de la nature spécifique de vos données, une plage de valeurs de k entre 2 et - pourrait être un bon point de départ pour expérimenter. Si les données sont relativement simples, un nombre plus bas de clusters pourrait suffire, mais si elles sont plus complexes, vous pourriez avoir besoin d'explorer des valeurs plus élevées.

```
22  # Essai de différentes valeurs de k
23  k_values = range(2, 10)
24  inertia_values = []
25
26  for k in k_values:
27      kmeans = KMeans(n_clusters=k, random_state=42)
28      kmeans.fit(X_scaled)
29      inertia_values.append(kmeans.inertia_)
```

Une boucle for itère sur chaque valeur de k. À chaque itération, l'algorithme k-Means est ajusté sur les données standardisées, et la somme des carrés des distances intra-cluster (inertia) est calculée et enregistrée dans la liste inertia_values.

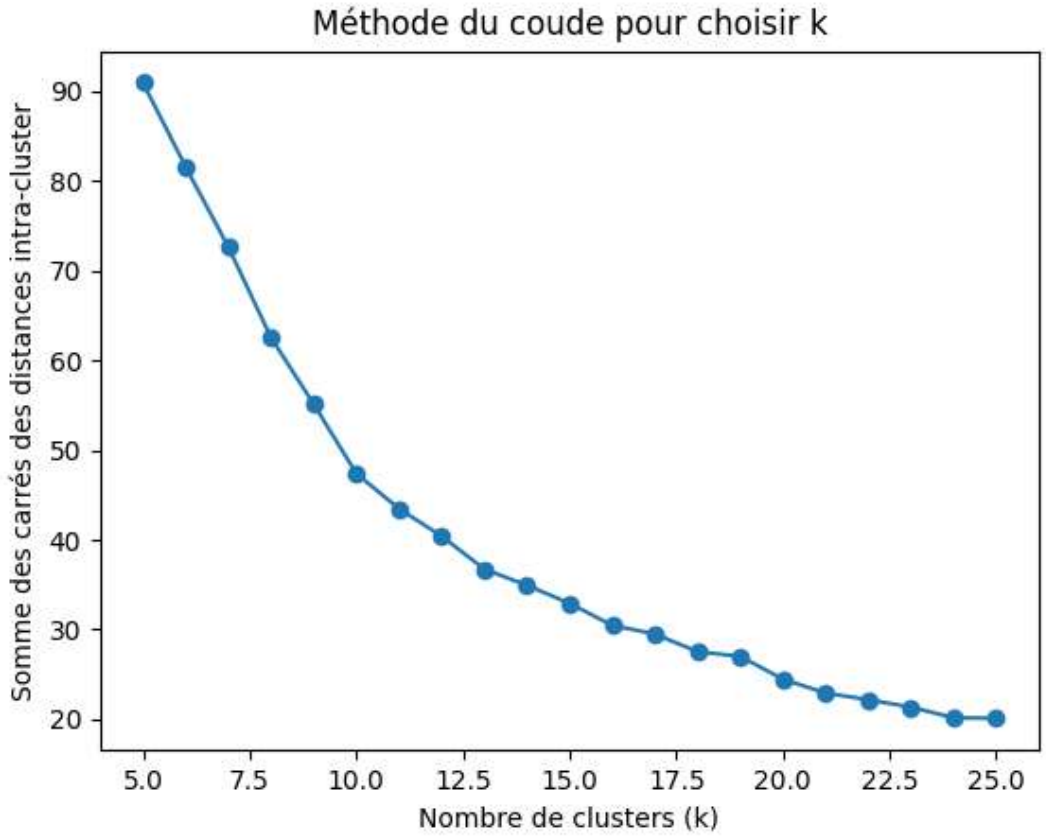
1.3.2. METHODE DU COUDE POUR CHOISIR K

Une fois que la somme des carrés des distances intra-cluster est calculée pour chaque valeur de k, le résultat est tracé graphiquement. La méthode du coude vise à identifier le point du graphique où l'ajout de clusters supplémentaires n'apporte qu'une amélioration minimale . Ce point est considéré comme le nombre optimal de clusters.



On observe sur le graphe que la courbe du nombre de clusters en fonction de somme des carrées des distances à une allure décroissante, ce qui signifie le point optimal est plus grand que 6.

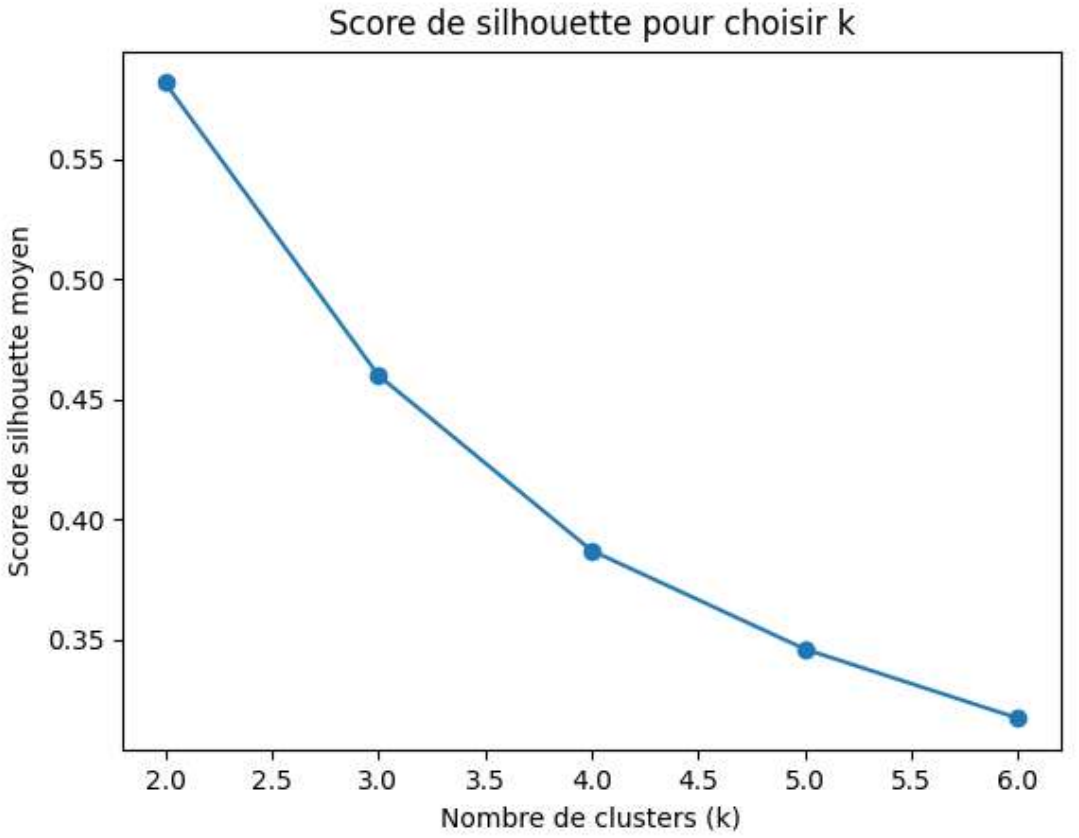
Essayons un intervalle plus large pour les valeurs de test : [2 : 26]



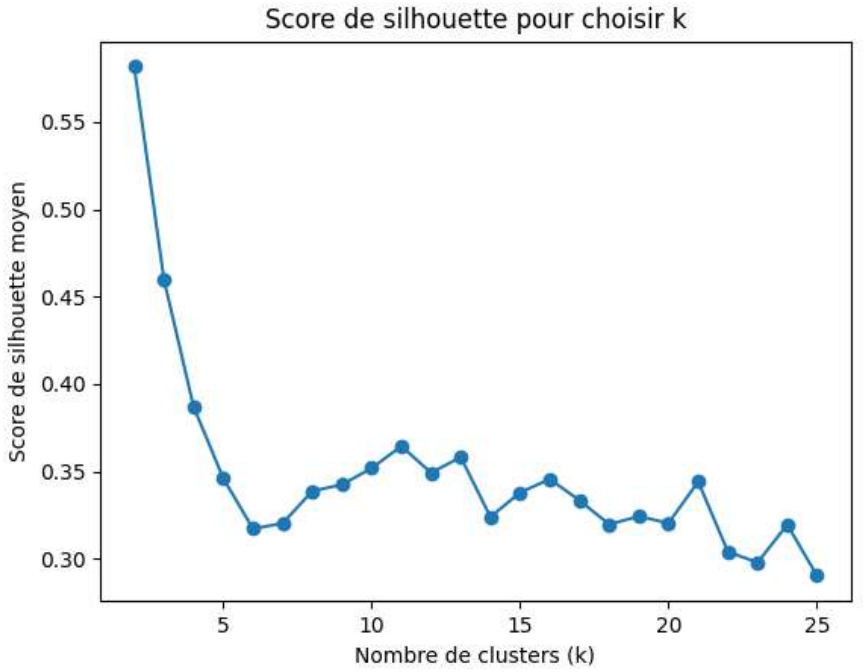
On observe que le meilleur K pour la méthode coude est : 24

1.3.3. METHODE DE LA SILHOUETTE POUR CHOISIR K

Elle évalue la qualité du regroupement des données en mesurant à quel point chaque point de données est similaire aux points de son propre cluster par rapport aux points du cluster le plus proche (le cluster voisin le plus proche). Le score de silhouette moyen est ensuite calculé pour chaque valeur de k, et le nombre optimal de clusters est celui qui maximise ce score.



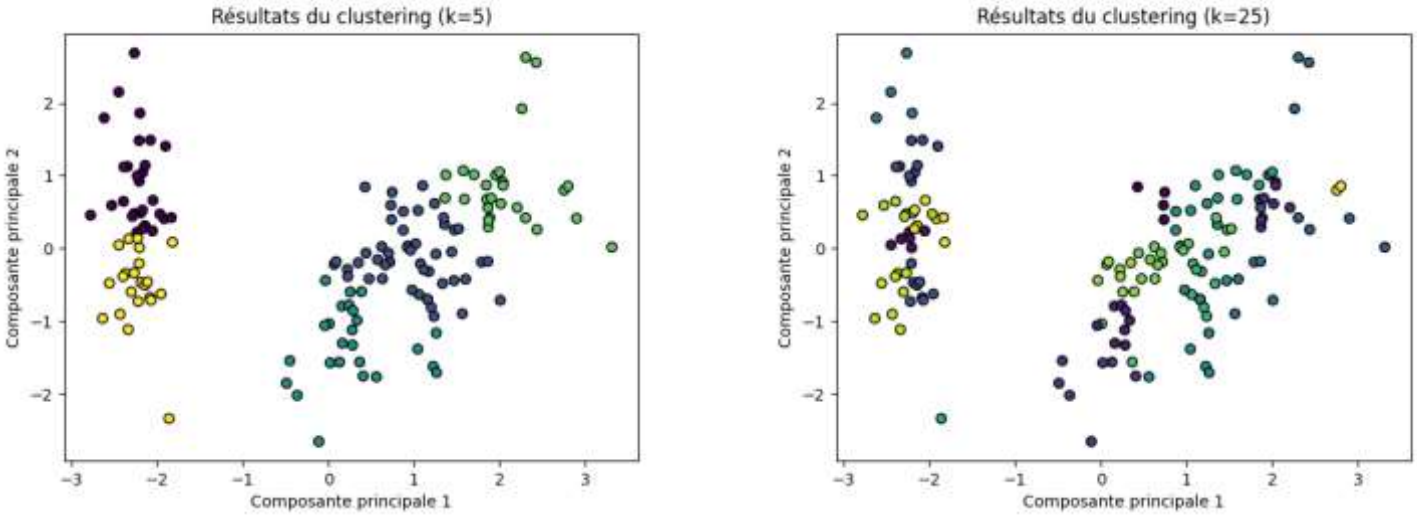
On observe des résultats contradictoires par rapport à la méthode « coude » car pour selon le graphe ci-dessus, la valeur optimale de k est la valeur la plus petit de l'intervalle de test !



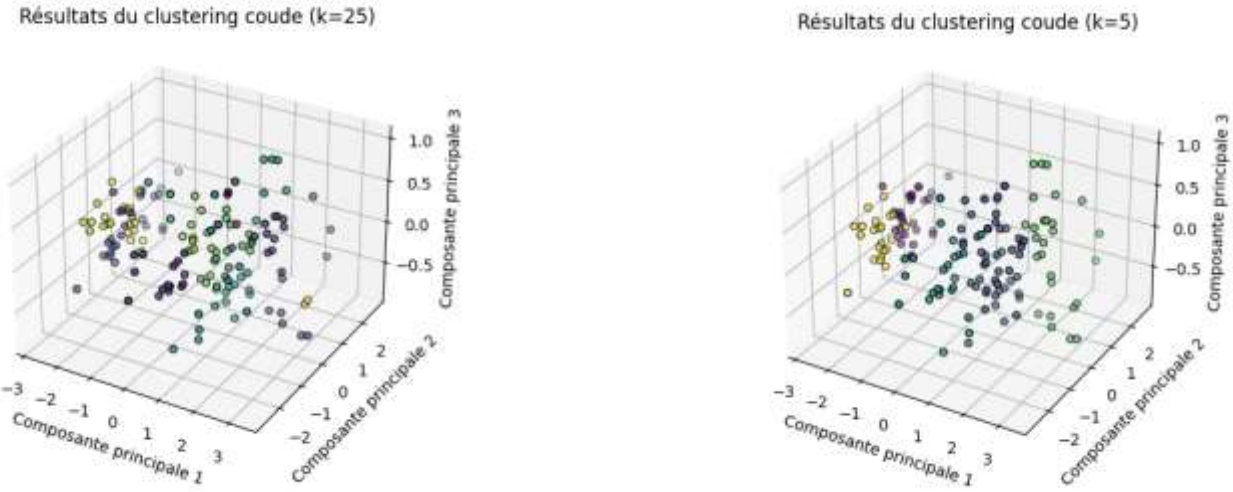
En augmentant l'intervalle de test, la valeur de k optimale reste la plus petite : 2

1.4. REPRESENTATION GRAPHIQUE DES RESULTATS

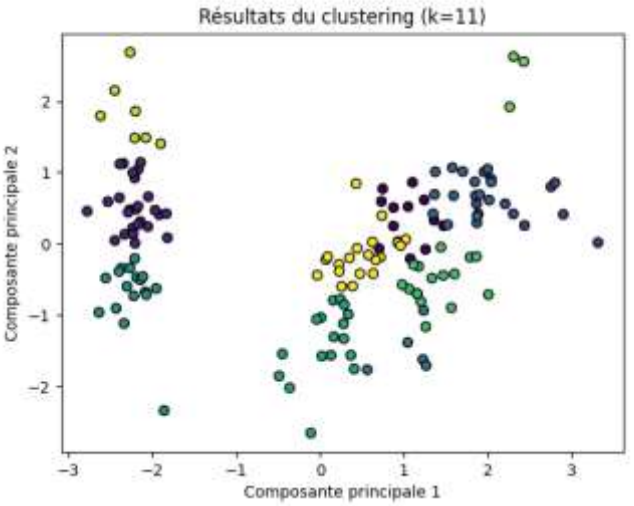
Pour représenter graphiquement les résultats du clustering dans un espace tridimensionnel, facilitant l'interprétation visuelle des clusters identifiés par l'algorithme K-Means avec le meilleur nombre de clusters. Et donc le k qui donne la somme des carrées des distances intra-cluster la plus petite et le score le plus haut pour la proximité d'un point d'objet avec les autres points du même cluster par rapport aux clusters voisins les plus proches.

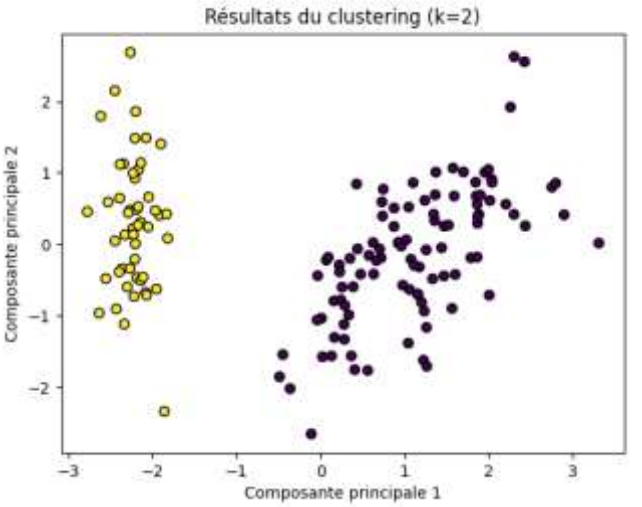


Représentation bidimensionnelle de la méthode coude

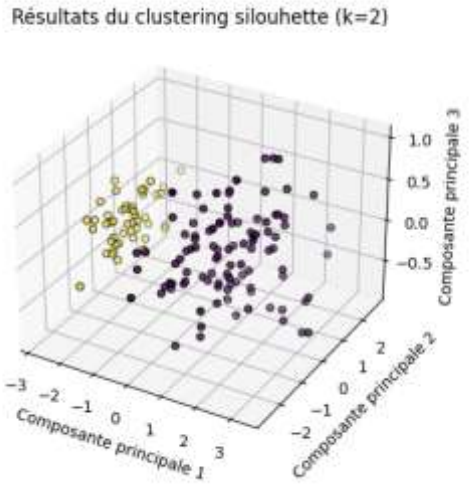


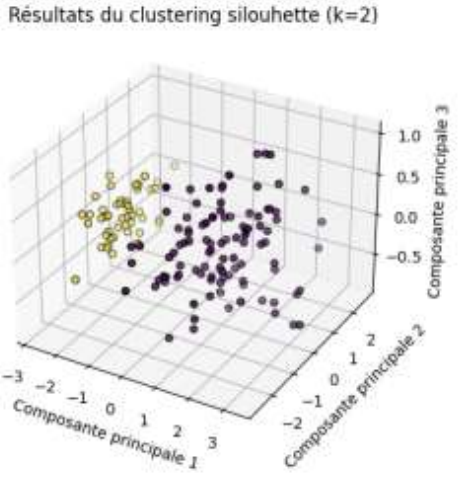
Représentation tridimensionnelle de la méthode coude





Représentation bidimensionnelle de la méthode silhouette





Représentation tridimensionnelle de la méthode silhouette

1.5. COMPARAISON DES RESULTATS AVEC DES VRAIES CLASSES D'OBJETS

La logique derrière cette comparaison repose sur l'utilisation de l'indice de Rand ajusté (ARI) pour évaluer la similarité entre les clusters obtenus à partir de l'algorithme K-Means et les vraies classes d'objets du jeu de données Iris.

Utilisation de la fonction adjusted_rand_score de scikit-learn pour mesurer la similarité entre les clusters obtenus et les vraies classes d'objets (y).

```
Indice de Rand ajust? (Coude): 0.14567937624130647
Indice de Rand ajust? (Silhouette): 0.5681159420289855
```

Plus l'ARI est proche de 1, plus les clusters obtenus correspondent aux vraies classes d'objets.