République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Informatique
Spécialité : Big Data Analytics

Module : TIAD

Rapport N° 03 :
Etude comparative entre l'algorithme itératif, k-means et dbscan

---

Travail présenté à M Necir

Travail réalisé par le binôme :
GOUMEZIANE Kenza
AISSANI Anouar

Année Universitaire : 2022/2023

Table des matières

Table des figures

Introduction

Dans le TP précédent, nous avons implémenté un algorithme itératif permettant de partitionner les données d'un jeu de données en k clusters, k étant inconnu au départ.

Nous avons utilisé le jeu de données Iris pour tester l'efficacité de notre algorithme, mais cela ne suffit pas pour décider de cette dernière.

Ainsi, dans le cadre de ce TP nous utiliserons un autre jeu de données qui est « **diabetic_data** », nous y appliquerons notre algorithme et nous comparerons les résultats obtenus avec ceux de k-means et dbscan.

Dans le présent rapport, nous expliquons les étapes de réalisation suivies.

Pré-traitement des données

Avant de pouvoir manipuler le jeu de données **diabetic_data,** nous devons nous assurer que les données qu'il contient sont exploitables, pour cela nous y appliquons quelques changement dont :

- La suppression des colonnes qui sont vides à plus de 40%.

```
#Supprimer les colonnes avec plus de 40% de valeurs manquantes
df.drop(['weight','payer_code','medical_specialty'],axis=1,inplace=True)
```

- La suppression des lignes avec beaucoup de données manquantes.

    - Nous commencons par chercher les lignes qui ne sont pas utiles comme celles dont le sexe du patient est inconnu.

```
drop_Idx = drop_Idx.union(set(df['gender'][df['gender'] == 'Unknown/Invalid'].index))
```

    - Nous terminons par supprimer toutes les lignes à la fois.

```
#Nouveaux ids
new_Idx=list(set(df.index)-set(drop_Idx))
#Jeu de données final
df=df.iloc[new_Idx]
```

A l'issue de cette étape nous avons réduit le jeu de données de `101766 lignes et 50 colonnes` à `96446 lignes et 47 colonnes.`

L'objectif du dataset étudié est de prévoir si un patient sera hospitalisé à nouveau après 30 jours de sa sortie de l'hôpital et ce, en se basant sur les données présentes dans le jeu de données.

Pour notre part, nous exploitons les deux colonnes **«time in hospital»** et **«num medications»** pour connaître le nombre de clusters créés selon les trois algorithmes dont nous disposons tout en gardant à l'esprit que nous sommes face à un problème de classification.

Normalisation des données

Afin d'assurer un résultat qui n'est pas faussé à cause des données qui se trouvent à des intervalles éloignés, nous avons effectué une normalisation de nos données

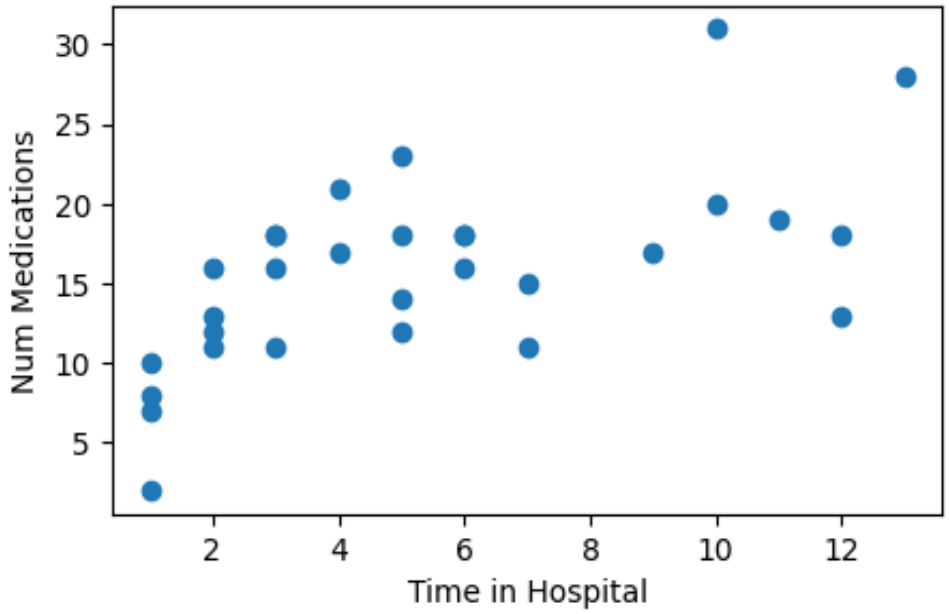

Figure 1: Représentation des données avant normalisation

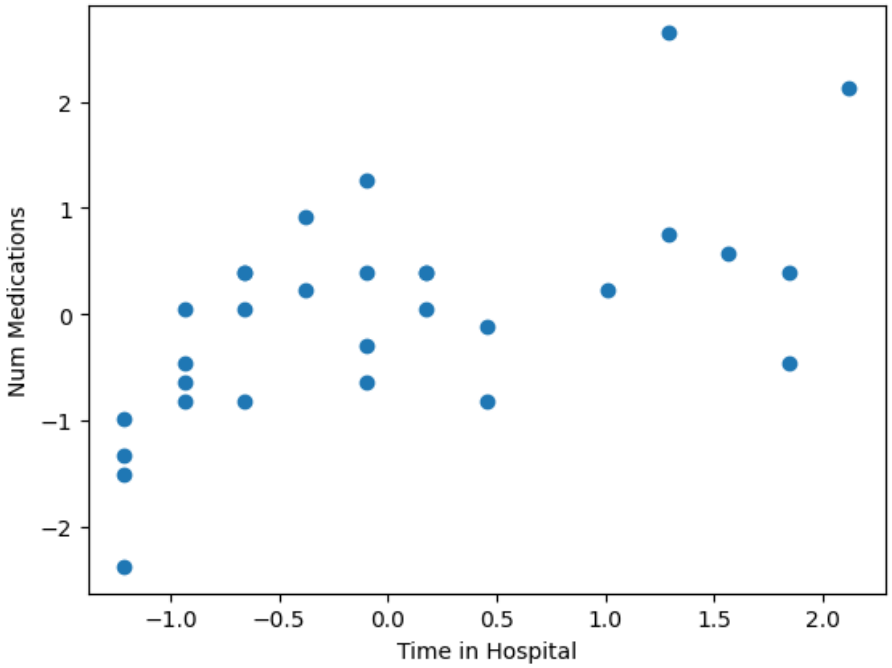

Figure 2: Représentation des données après normalisation

Application de l'algorithme itératif

Commençons par appliquer l'algorithme itératif pour pouvoir le comparer aux deux autres algorithmes.

Notre algorithme a donné le résultat suivant à deux clusters :

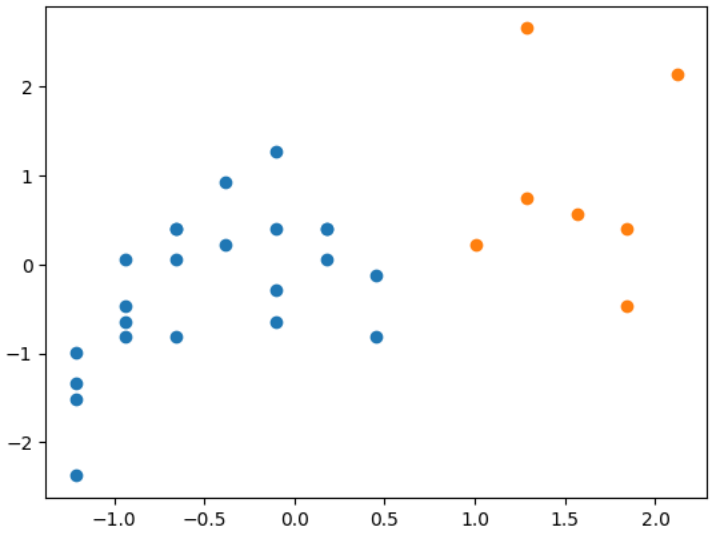

Figure 3: Résultat de l'algorithme itératif

Nous interprétons le résultat obtenu par rapport aux données utilisées comme suit :
Plus le nombre de médicament ainsi que le nombre de jours passés à l'hôpital augmentent et plus la chance que le patient revienne à l'hôpital augmentent et le contraire reste vrai c'est-à-dire que *plus le nombre de médicament ainsi que le nombre de jours passés à l'hôpital diminuent et plus la chance que le patient revienne à l'hôpital diminue.*

Nous voyons bien que nous pouvons tracer une frontière de décision entre les deux clusters obtenus ce qui prouve l'efficacité de l'algorithme.

Voyons ce que dbscan et k-means vont donner.

Application de l'algorithme DBSCAN

Pour décider du rayon de voisinage et du nombre de voisins à donner comme entrées à cet algorithme, nous avons effectué des recherches sur cette approche et nous sommes arrivées au conclusions suivantes :

- Lorsque nous avons deux colonnes à étudier, nous optons pour un **minp = 4**.
- Pour calculer la valeur du rayon de voisinage, nous calculons la distance entre chaque point de données et son voisin le plus proche en utilisant Nearest Neighbours. Après cela, nous les trions et enfin traçons le résultat comme le montre la figure suivante.
  À partir du graphe, nous identifions la valeur maximale à la courbure du graphique.
  Cette valeur est notre rayon et donc dans notre cas **R= 0.8.**

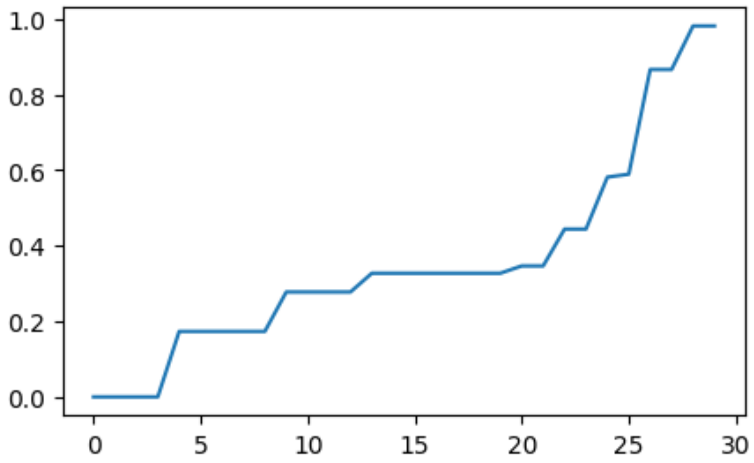

Figure 4: Approximation du rayon de voisinage pour dbscan

L'application de l'algorithme donne le résultat suivant :

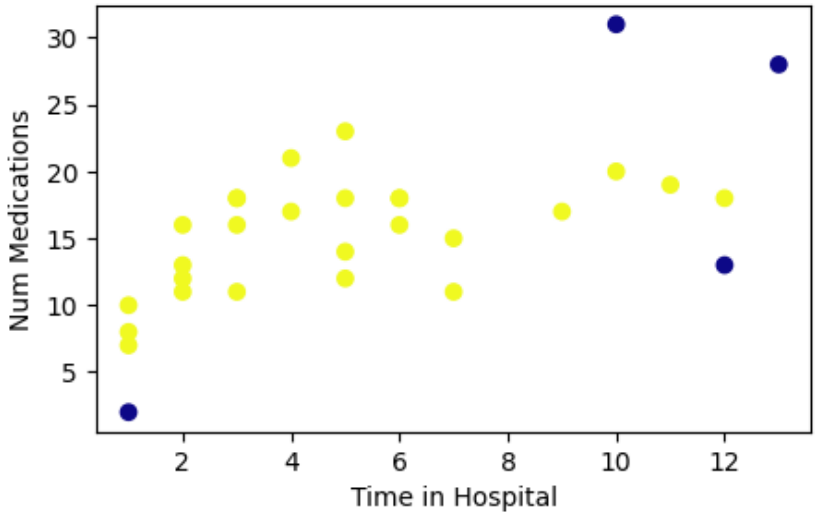

Figure 5: Résultat de l'algorithme dbscan

Le résultat montre un seul cluster et quatre points qui peuvent être considérés aberrants.

Comparaison entre l'algorithme itératif et DBSCAN

Le nombre de clusters étant différent, nous ne pouvons pas comparer les distances intra et inter.

Nous constatons en revanche que l'algorithme basé sur la densité a généré un seul cluster alors que dans un problème de classification nous cherchons à séparer au mieux nos données en deux ensembles.

D'un autre côté, l'algorithme itératif a généré deux clusters qui peuvent être répartis de la même sorte à l'œil nue.

Conclusion :

Nous concluons à ce niveau **que l'algorithme itératif a donné de meilleurs résultats par rapport à dbscan.**

Application de l'algorithme k-means

Nous appliquons maintenant l'algorithme k-means pour k = 2.

Nous avons défini k égal à deux car nous sommes face à un problème de classification et qu'ainsi nous pourrons comparer le résultat obtenu avec celui de l'algorithme itératif.

Le clustering avec k-means est comme le montre la figure suivante :

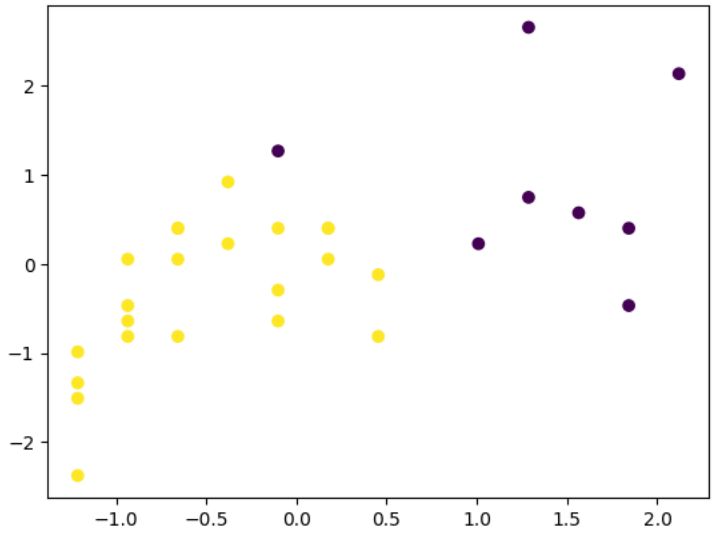

Figure 6: Résultat de l'algorithme k-means

Nous remarquons que l'algorithme a généré deux clusters séparés entre eux mais qu'il y a un individu dans le deuxième cluster qui semble plus proche du premier cluster.

Comparaison entre l'algorithme itératif et k-means

Nous commençons par comparer le nombre d'individus contenus dans les deux clusters

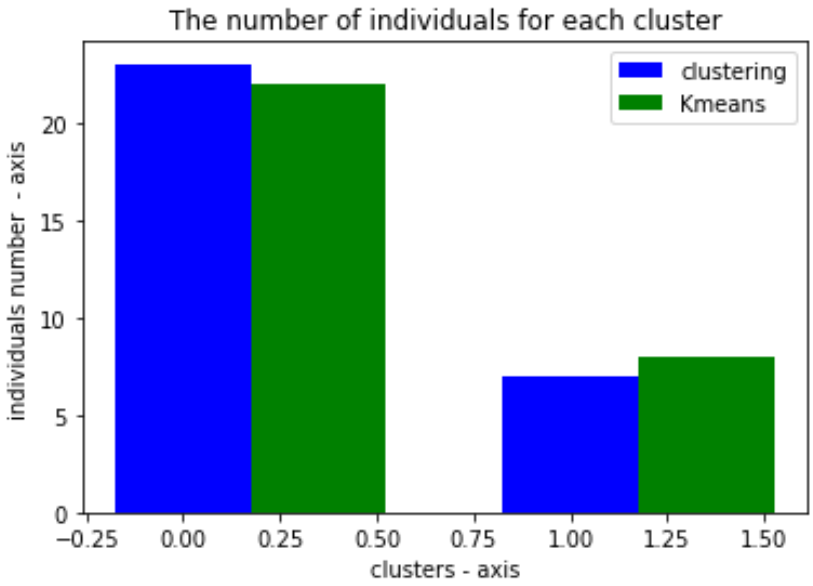

Figure 7: Comparaison entre le nombre d'individus : k-means vs itératif

Nous voyons bien que les résultats sont pareils à un individu près. Cet individu est celui que nous avons constaté dans le graphe précédent.

Comme nous avons là deux clusters dans chacun des deux algorithmes, nous pouvons comparer les distances entre les individus du même cluster obtenue dans les deux cas ainsi que les distances entre les clusters.

Distance inter-clusters :

Nous rappelons qu'un bon clustering maximise la distance entre les clusters.

Dans le graphe suivant, nous constatons que le clustering itératif a réalisé une distance entre les clusters qui est supérieur à celle réalisé par k-means.

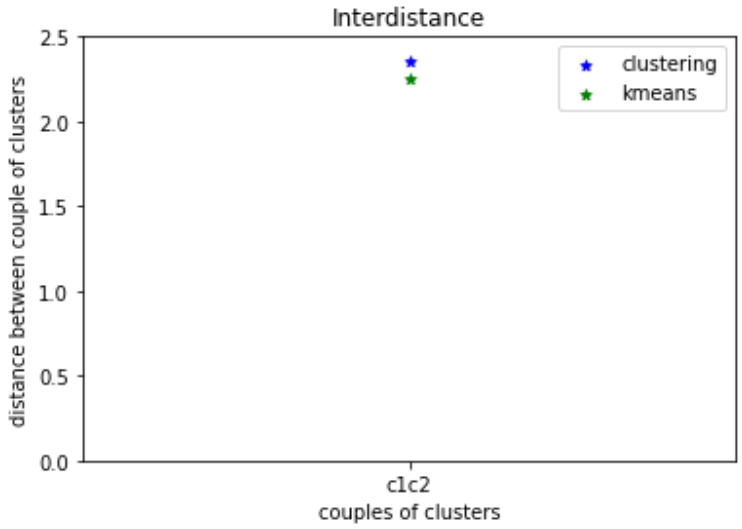

Figure 8: Comparaison distance inter : k-means vs itératif

Distance intra-clusters :

Nous rappelons qu'un bon clustering minimise la distance entre les éléments d'un même cluster.

Dans le graphe suivant, nous constatons que pour le premier cluster, l'algorithme itératif a réalisé une distance entre les clusters qui est inférieur à celle réalisé par k-means. Par contre, pour le deuxième cluster c'est k-means qui a minimisé la distance intra.

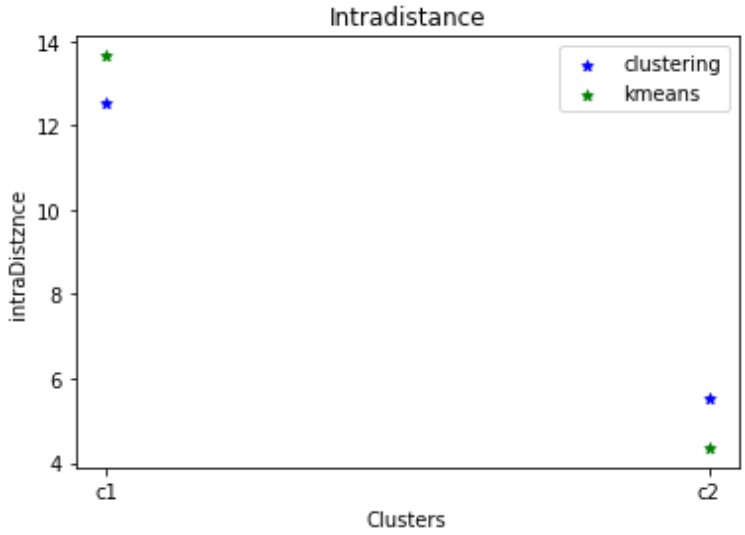

Figure 9: Comparaison distance intra : k-means vs itératif

Conclusion :

L'algorithme itératif et k-means ont donné tous les deux de bons résultats. Leurs distances intra sont presque les mêmes tandis que la distance inter est meilleure dans l'algorithme itératif.

Conclusion

Nous arrivons à la fin de cette étude comparative entre les résultats du clustering obtenus par l'algorithme réalisé durant le TP et les deux algorithmes k-means et dbscan vus en cours.

A l'issue de cette étude, nous avons constaté l'efficacité de notre algorithme qui a bien su séparer notre jeu de données en clusters homogènes.

L'algorithme itératif a donné un meilleur résultat par rapport à dbscan dans la mesure où ce dernier n'a généré qu'un seul cluster. Aussi, l'algorithme itératif a généré une distance inter plus grande comparée à celle de k-means et des distances inter quasi-pareilles.