Reconnaissance de chiffres manuscrits
Base MNIST

MAT4102
Tristan Amadei

Introduction

Le but de cet TP est d'entrainer un algorithme à reconnaître des chiffres écrits à la main, à partir de la base de chiffres MNIST.
On va pour cela mettre en place deux approches. Premièrement, on va considérer les images comme étant la seule source de données qu'on possède, on ne prend pas en compte les labels auxquelles elles sont associées. Pour mettre en place cette approche, on implémentera un algorithme K-Means, puis un algorithme K-Medoids, dont on comparera les résultats.
Enfin, nous utiliserons les labels des images sources pour évaluer la qualité des clusters.
La base de données MNIST est une base de chiffres manuscrits. Elle contient 60,000 images, soit 6000 images par chiffre. Chaque image est libellée par le chiffre qu'elle représente.
On peut visualiser certaines images de cette base pour se donner une idée de ce avec quoi on travaille.

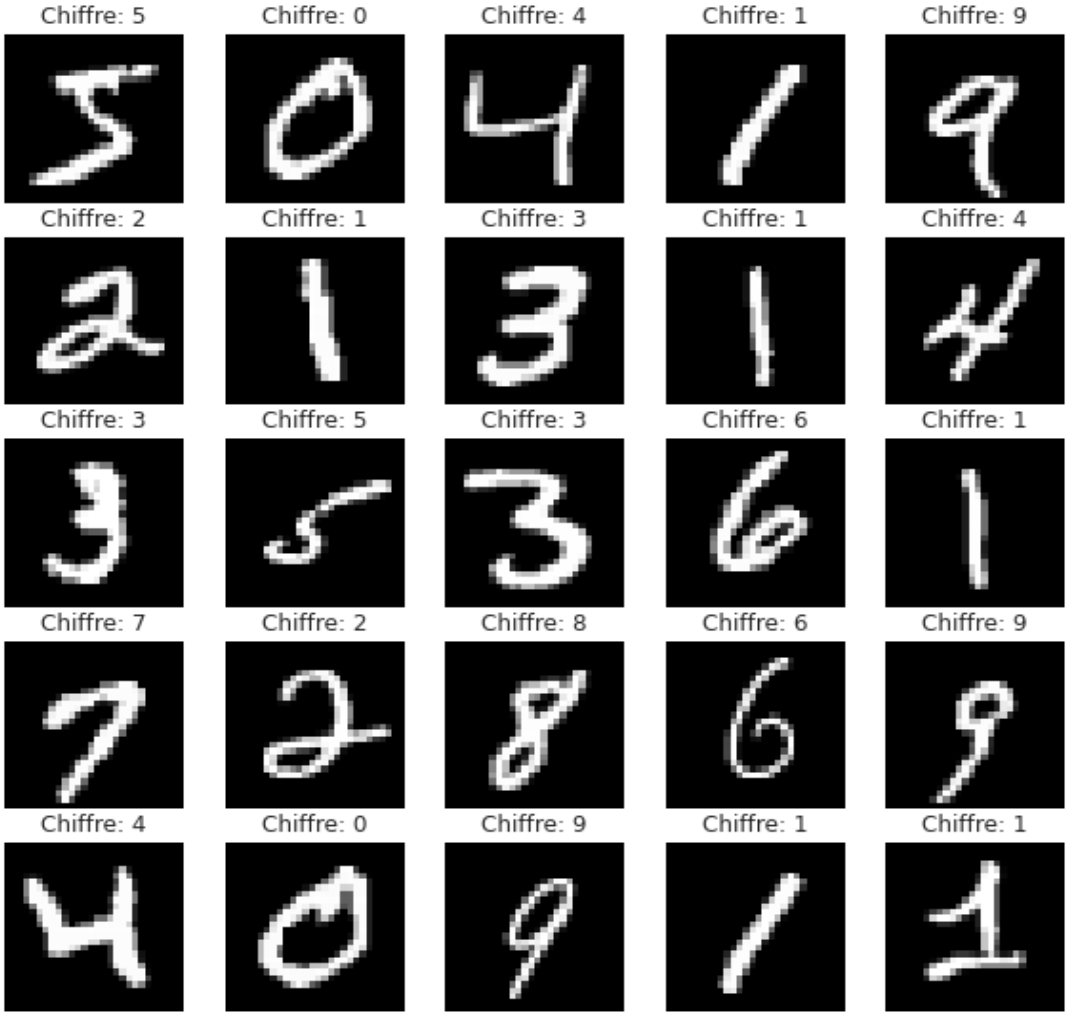

Figure 1 : images de MNIST

Partie 1 : Considérer les images de chiffres indépendamment de leurs labels et réaliser des regroupements de ces chiffres

• K-Means

On cherche à regrouper les images dans des clusters. On commence par implémenter la méthode K-Means.
Il y a 10 classes différentes dans les données sources, correspondant aux 10 différents chiffres, on serait donc tenté de choisir 10 clusters pour notre algorithme K-Means.
Cependant, on va chercher à être plus précis. On va faire varier le nombre de clusters et calculer, pour chaque cas, la fonction de coût de notre méthode de clustering. Le but étant, à terme, de choisir le nombre de clusters optimal grâce à la méthode du coude.
On se sert également d'autres indices pour chercher le nombre de clusters optimal, comme le score de silhouette, l'indice de Calinski-Harabasz et l'entropie moyenne pondérée.

On cherche la valeur du nombre de clusters qui maximise la silhouette du modèle, tout en maximisant également l'indice de Calinski-Harabasz et l'entropie moyenne pondérée. Cette valeur doit aussi respecter le critère de la méthode du coude.

On commence par calculer le coût du modèle pour différents nombres de clusters.

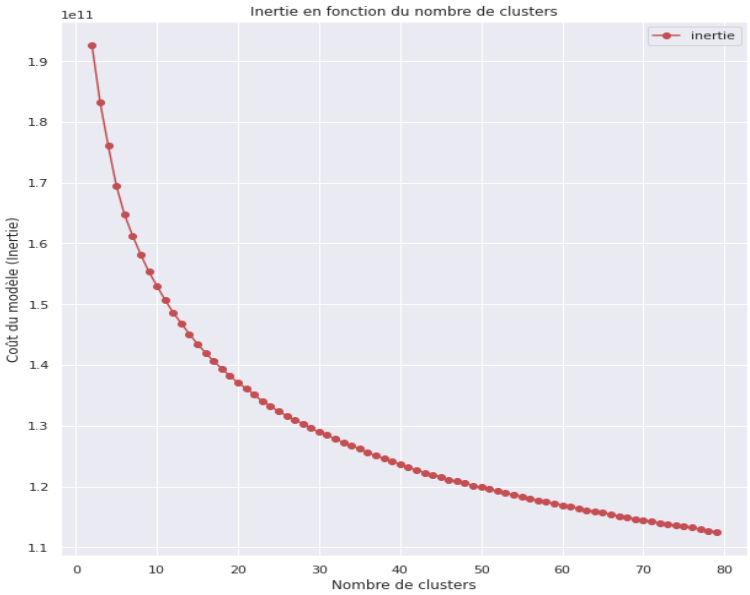

Figure 2 : tracé de l'inertie en fonction du nombre de clusters

Ce tracé, ci-dessus, nous permet, grâce à la technique du coude, de considérer que le nombre optimal de clusters se trouve environ entre 20 et 30.

On trace ensuite les autres indices en fonction du nombre de clusters. Le but est de trouver leurs maximas, afin de trouver le nombre de clusters optimal maximisant tous ces indices.

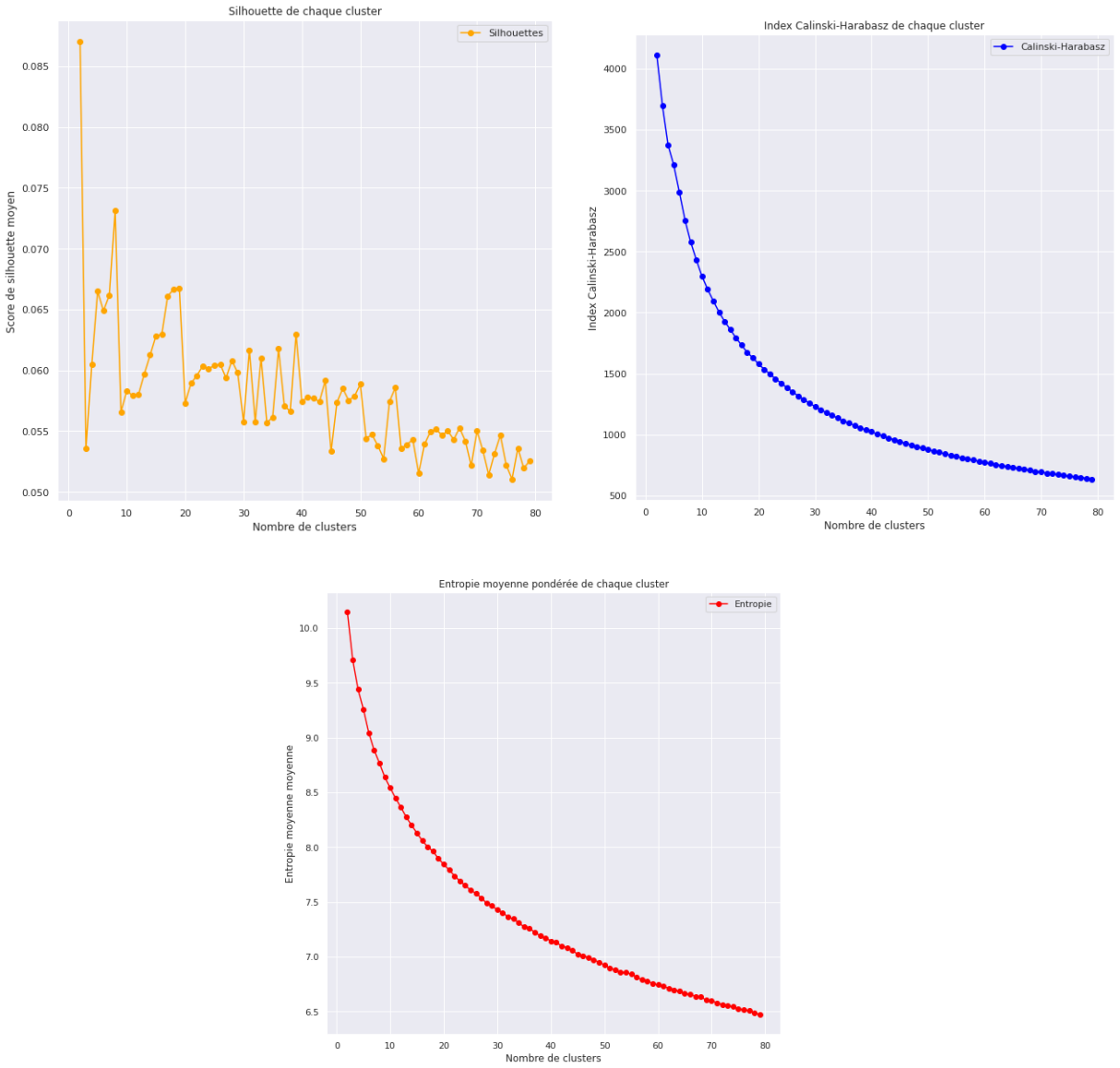

Figure 3 : tracés des indices en fonction du nombre de clusters

L'un des maximas des silhouettes se trouve pour n_clusters = 19.
On choisit donc 19 clusters comme nombre optimal de clusters.

Maintenant qu'on a déterminé le nombre optimal de clusters, on peut implémenter notre méthode optimale.
On peut alors visualiser les premières images de certains clusters qui ont été déterminés. On choisit au hasard de visualiser les images des clusters n°0 et n°8.

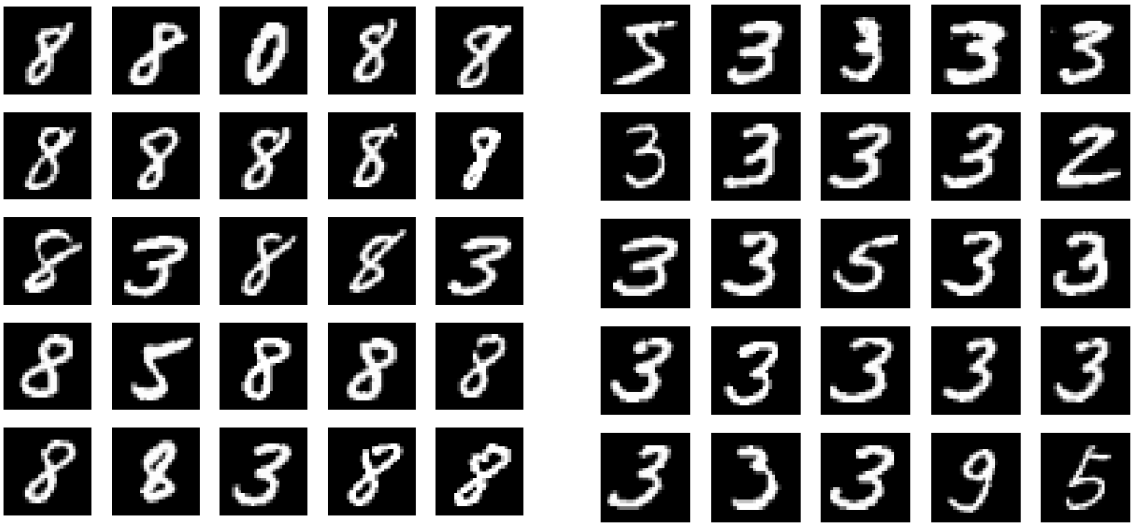

Figure 4 : images des clusters n°0 et n°8

Pour se rendre compte de la précision de notre modèle, on peut afficher les images des centres de nos clusters. Cela permet de vérifier que chaque image est bien différente des autres et que chacune possède une particularité.

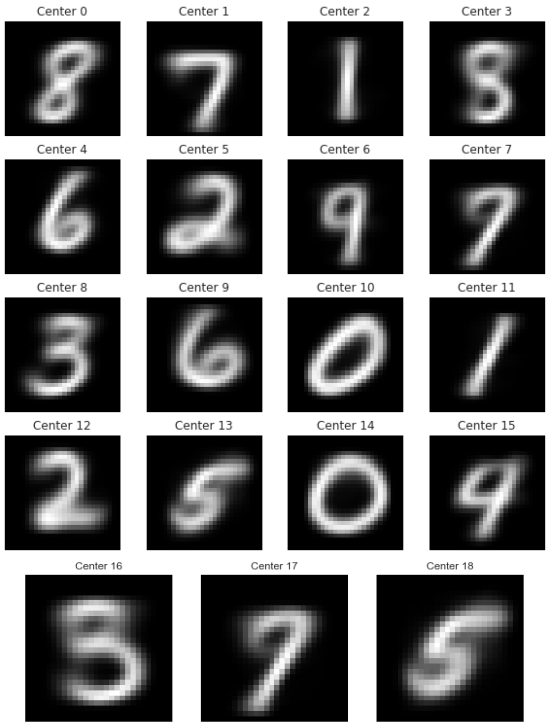

Figure 5 : images des centres des clusters

On remarque bien que les centres ne sont pas des points appartenant à la base MNIST, ce sont des points créés par l'algorithme K-Means.
Cela nous permet de comprendre comment a fonctionné le modèle, et déterminer quels clusters sont très homogènes, et lesquels ont pose plus de difficultés à créer. Par exemple, les centres n°10 et n°14, qui sont des 0, semblent être très homogènes, c'est-à-dire que la grande majorité des images dans ce cluster sont bien des 0. On s'en rend compte car le centre est pratiquement un 0 et n'est que très peu flou. En revanche, le centre n°15 semble être le centre d'un cluster composé de

beaucoup de 4 et de 9, donc ce cluster contient plus d'imprécision et est moins homogène que ceux cités précédemment.
On peut s'en rendre compte en affichant les première images.

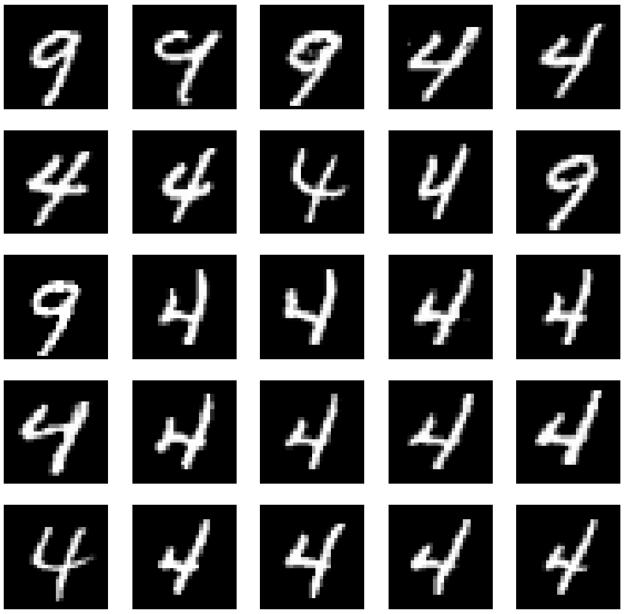

Figure 6: images du cluster n°15

Effectivement, il y une grande partie des images qui représentent le chiffre 4, mais un nombre non négligeable de ce cluster représente le chiffre 9.
Cependant, on se rend compte, en regardant les centres de nos clusters, que le modèle a bien classifié les images car chaque chiffre est le centre d'au moins un cluster. Cela veut dire que, par exemple, les 9 ont majoritairement été reconnus comme tels, et le modèle ne les a pas considérés comme des 3.

• **K-Medoids**

On effectue les mêmes opérations que précédemment, mais cette fois-ci en implémentant la méthode d'apprentissage K-Medoids.
On peut commencer par supposer que le bon nombre de clusters pour l'algorithme K-Medoids est le même que celui qu'on avait trouvé pour K-Means avec la méthode du coude.
On choisit ainsi 19 clusters. La première chose qu'on remarque en implémentant cet algorithme est la différence de temps d'exécution par rapport à K-Means. En effet, pour 19 clusters, K-Means met un peu moins de 1000 secondes, soit environ 16-17 minutes, alors que K-Means met seulement un peu plus de 50 secondes.
Cependant, l'avantage de K-Medoids est que les centres correspondent à des images du set d'entrainement de la base MNIST, on peut donc les afficher et nous faire une idée des clusters créés.

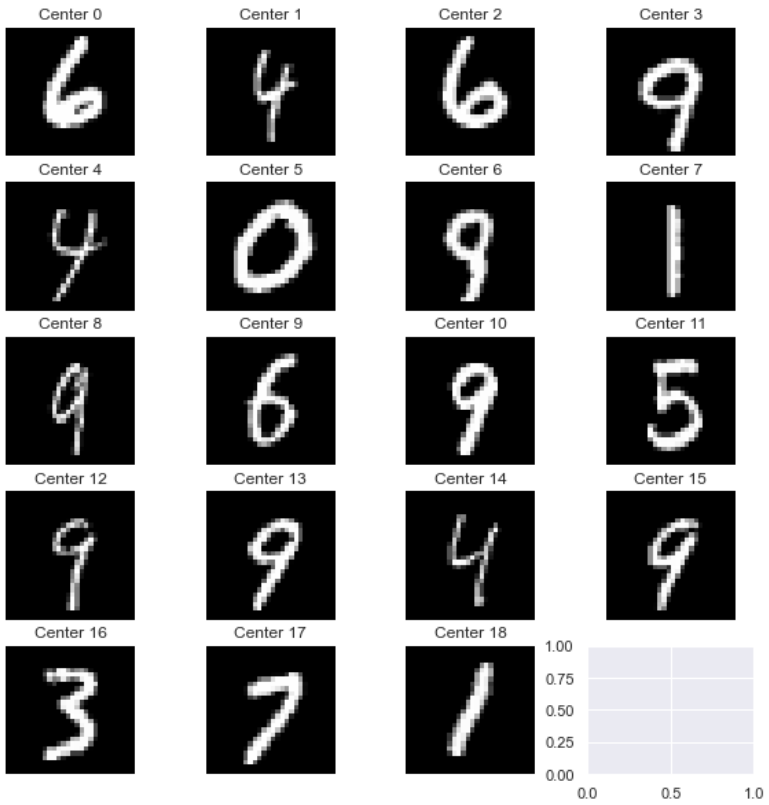

Figure 7: images des centres des clusters pour K-Medoids avec 19 clusters

Les centres sont ici bel et bien des images de la base d'entrainement.
Cependant, ici les clusters semblent moins bien représenter les données que les clusters de K-Means. En effet, les chiffres 2 et 8 ne sont jamais des centres de clusters, donc ces chiffres-là seront souvent mal classés.
De plus, l'indice de Calinski-Harabasz est égal à 1135.06 pour K-Medoids, et son score d'entropie moyenne pondérée est égal à 7.94. Alors que pour le même nombre de clusters, K-Means possède un score d'entropie proche de celui de K-Medoids, égal à 7.9 ; par contre, il possède un indice de Calinski-Harabasz bien plus important à 1628.47, ce qui indice que les clusters créés par K-Means, avec les mêmes paramètres et données d'entrée, sont plus denses et séparés entre eux.
Il serait donc judicieux de chercher le nombre optimal de clusters propre à K-Medoids, avec la méthode du coude.

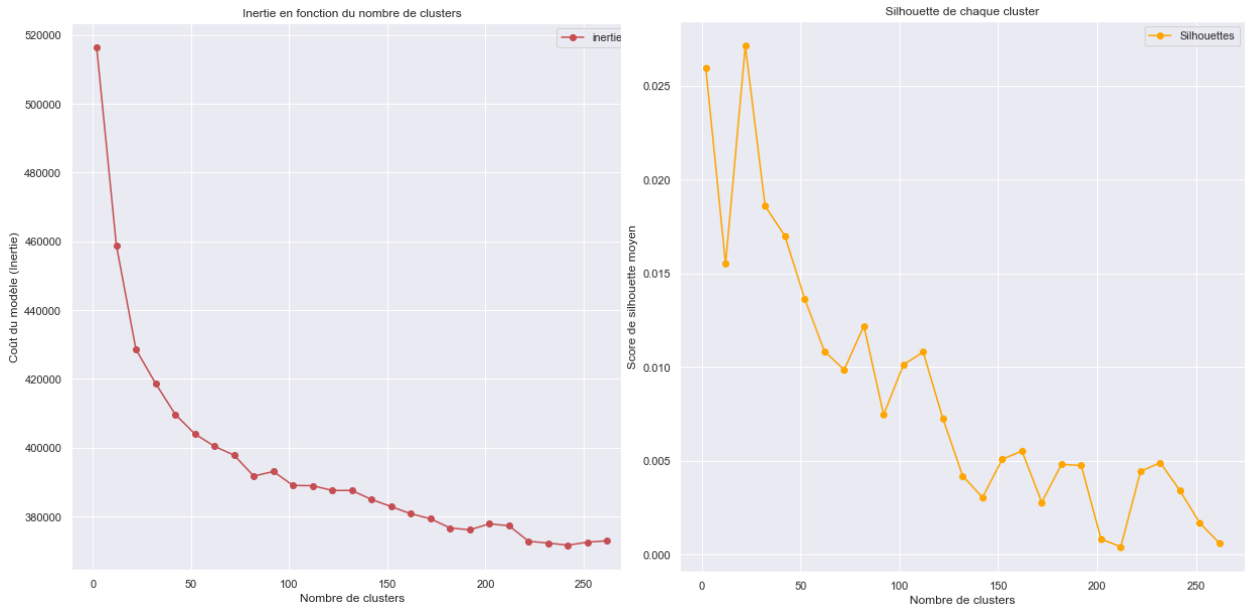

Figure 8: tracés de l'inertie et du score de silhouette pour K-Medoids en fonction du nombre de clusters

Le score de silhouette nous indique un clair maximum pour 22 clusters, ce qui est bien corroboré par le tracé de l'inertie sur la figure de gauche.
On en déduit donc par cette méthode, que le nombre optimal de clusters pour l'algorithme K-Medoids associé à cette méthode est égal à 22.
On va alors implémenter K-Medoids avec ce nombre de clusters, et visualiser les centres des clusters créés.

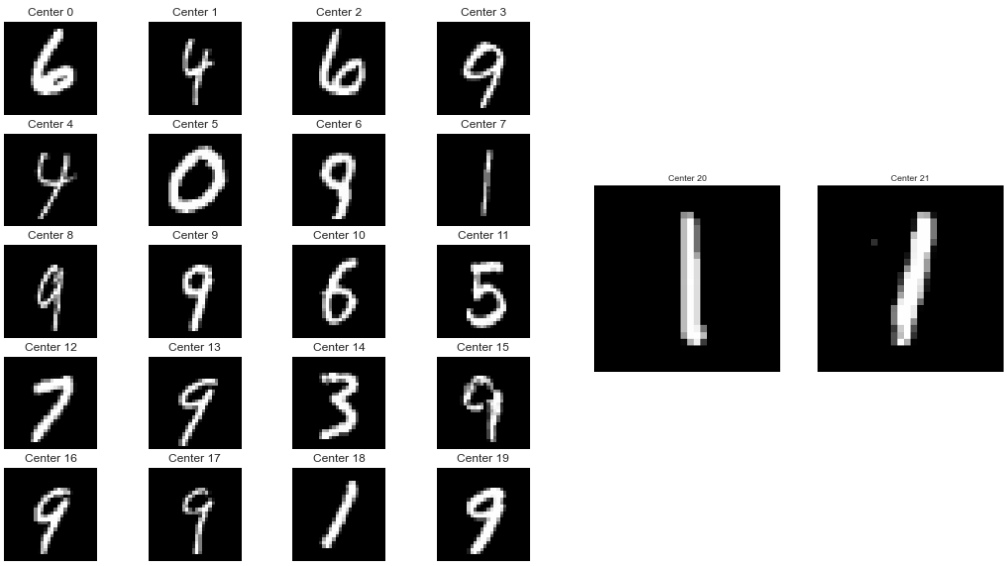

Figure 9: images des centres des clusters pour K-Medoids avec 22 clusters

Avec la visualisation de ces centres, on se rend compte que l'algorithme parvient à bien classifier certains chiffres, comme le 0 ou le 6, mais à des difficultés à faire de même pour d'autres, comme le 2 ou le 8.
En affichant les premières images des clusters, on peut vérifier ce point.

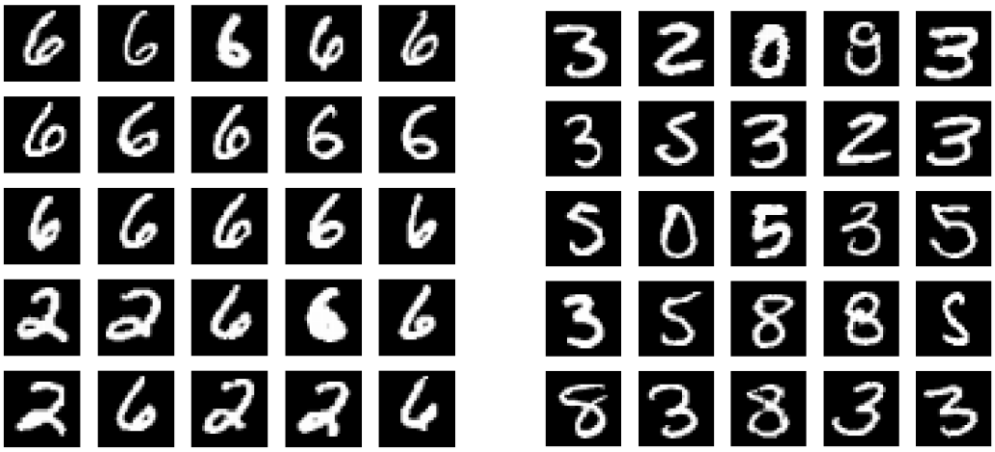

Figure 10: images des clusters n°0 et n°12 pour K-Medoids avec 22 clusters

Cependant, les résultats de K-Medoids, moins bons que ceux de K-Means, peuvent s'expliquer par le fait que K-Medoids est efficace pour gérer des bases de données contenant des *outliers*. Cependant, dans la base MNIST, même si certains chiffres semblent difficiles à lire ou déchiffrer, ils ne représentant pas des outliers par rapport aux autres données. Ainsi, dans ce contexte, K-Medoids est moins efficace que K-Means, en plus du fait qu'il soit plus long à exécuter.
Cependant, dans un autre contexte, K-Medoids peut présenter des avantages sur le nombre de clusters à choisir. En effet, la complexité de l'algorithme K-Means est en O(N.k.t) avec t le nombre d'itérations de l'algorithme, k le nombre de clusters et N le nombre d'éléments dans les données d'entrée. Ainsi, cet algorithme voit son temps d'exécution varier selon le nombre de clusters, comme on peut le voir sur les figures plus loin dans ce rapport. De son côté, K-Medoids a une complexité en $O(N^2.kT)$. En considérant N >> k, t, ce qui est majoritairement le cas, il sera très peu sensible aux variations du nombre de clusters. Ainsi, même si son temps d'exécution restera probablement plus long que celui de K-Means, l'utilisateur peut l'approximer avant de lancer l'algorithme et il restera relativement constant quelque soit le nombre de clusters.

Partie 2 : Prendre en compte les labels des classes

On va maintenant considérer les images et leurs labels comme un set. On peut alors utiliser ces labels pour évaluer la qualité des clusters que nous avons construits avec les méthodes précédentes.

Avec ces étiquettes, on peut calculer l'homogénéité des clusters qu'on a créés. Plus l'homogénéité est proche de 1, plus les clusters du modèle créé ne contiennent des données qui ne sont membres que d'une classe, c'est-à-dire que les clusters que contiennent qu'un seul type de chiffre.
On peut aussi mettre en place une fonction qui calcule le nombre d'images mal classées et ainsi calculer le pourcentage de précision de notre modèle.

On peut alors remarquer que les résultats qu'on trouve avec ces nouvelles données diffèrent des résultats précédents sur le nombre optimal de clusters. En effet, on avait trouvé précédemment que, pour la méthode K-Means, le nombre optimal était 19. Or, on remarque ici que pour 19 clusters, la précision du modèle n'est que d'environ 0.75 et l'homogénéité est égale à environ 0.65.

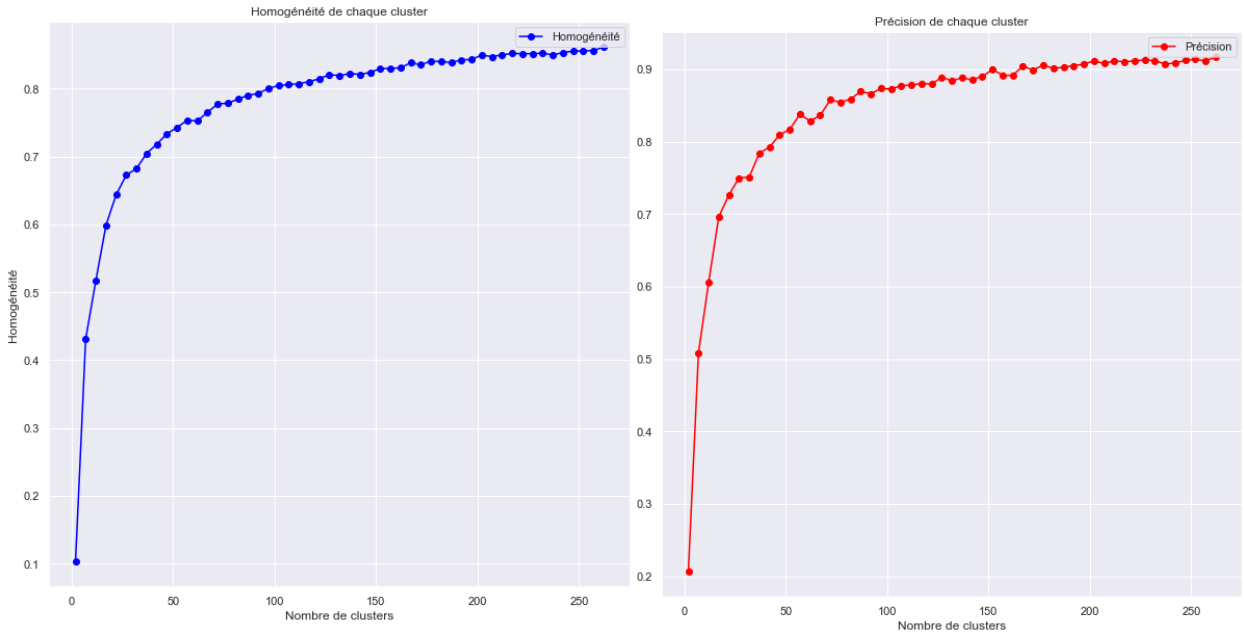

Figure 11 : tracé de l'homogénéité et précision en fonction du nombre de clusters

Mais on remarque aussi que ces deux paramètres augmentent quand le nombre de clusters augmente, et il faut prendre 150 clusters pour obtenir au moins 90 % de précision.
On pourrait alors penser qu'il suffit de prendre un nombre très grand de clusters, ce qui nous octroierait une précision proche de 1. Mais il y a deux problèmes à cela. Premièrement, le temps d'exécution augmente proportionnellement au nombre de clusters. Comme on peut le voir sur le graphique ci-dessous (Figure 7), plus le nombre de clusters est important, plus l'est également le temps d'exécution du K-Means associé.

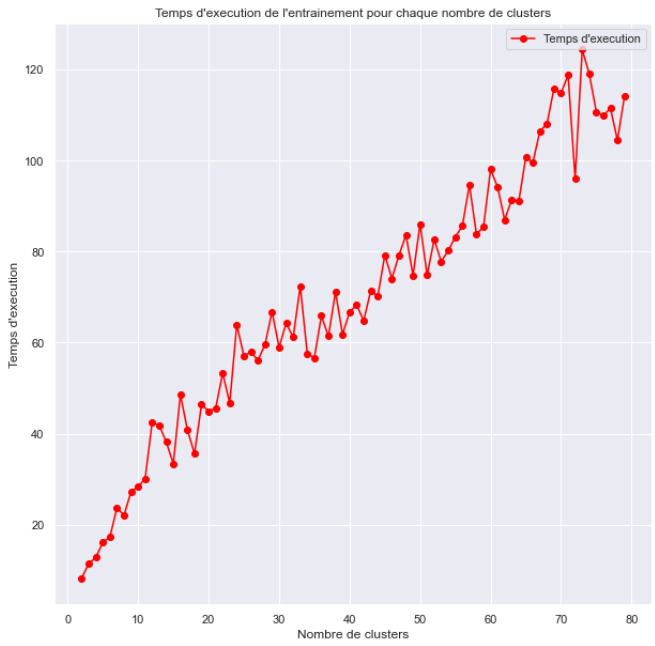

Figure 12 : tracé du temps d'exécution en fonction du nombre de clusters

Et cette base est relativement petite en termes de nombre d'entrées. Pour une base significativement plus grande, le temps d'exécution exploserait.
Le second problème est celui du sur-apprentissage. En effet, créer un grand nombre de clusters nous permet d'obtenir une très bonne précision sur nos données d'apprentissage, mais cela ne garantit pas que le modèle soit capable de coller aux datas réelles.
Il faut donc être prudent sur le nombre de clusters à implémenter. Cependant, cela ne veut pas dire forcément qu'il faut rester dans un nombre de clusters proche de 10. En effet, par exemple, pour 150 clusters, on obtient une précision de 0.895 pour les données d'entrainement, mais avec les données de test, on garde une précision haute, égale à 0.896. Ainsi, si le temps d'exécution ne pose pas de problème, il est utile dans ce cas de faire augmenter le nombre de clusters.
On remarque aussi que, plus le nombre de clusters augmente, plus les indices de densité des clusters diminuent, comme la valeur de l'entropie moyenne. Ainsi, les clusters créés par un K-Means avec un grand nombre de clusters sont de mauvaise qualité, les images ne sont pas très rapprochées les unes des autres et les distances entre clusters ne sont pas suffisamment grandes pour que l'algorithme soit réellement efficace sur des données futures.

Avec l'ajout des labels associés aux images, on peut également s'intéresser à l'homogénéité des clusters créés par K-Medoids, qui est censé prioriser cet aspect.
Mais sur ce point également, K-Medoids semble moins performant que K-Means, chacun pour son nombre de clusters optimal.
En effet, pour 22 clusters, K-Medoids produit ces indices :

- Indice Calinski-Harabasz = 1032.70
- Entropie moyenne pondérée = 7.8

Ces indices, que l'on cherche à maximiser, sont inférieurs à ceux fournis par les clusters de K-Means.

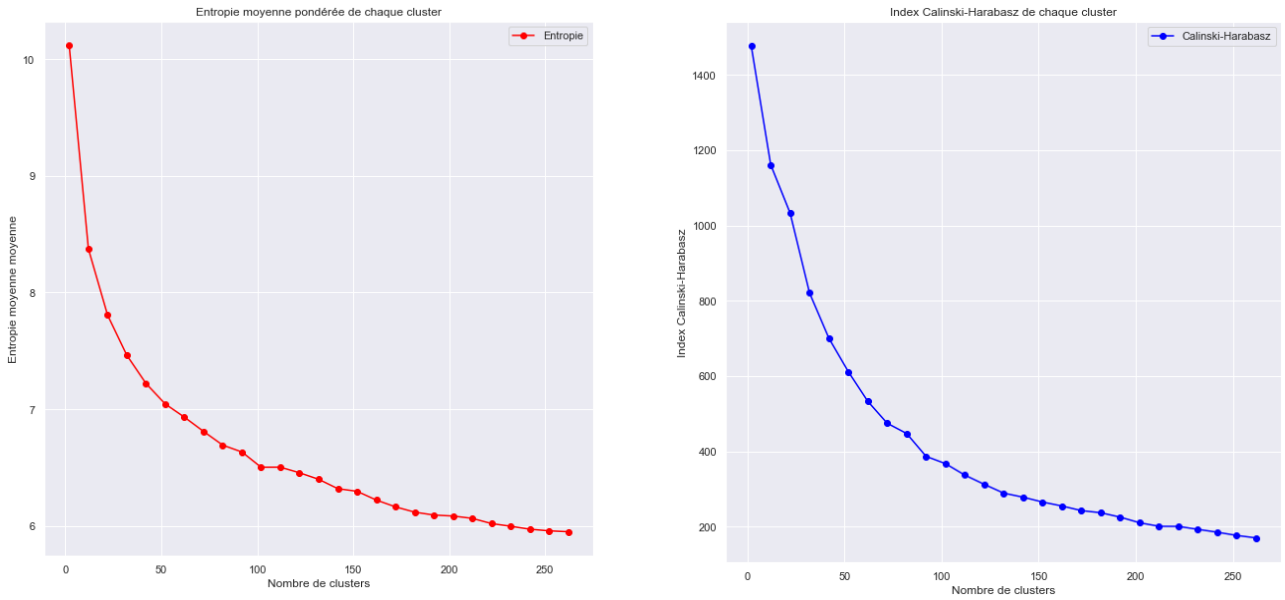

Figure 13 : tracés de l'entropie moyenne et de l'indice CH en fonction du nombre de clusters

Néanmoins, si l'on compare ces courbes avec celles produites par K-Means, l'entropie de K-Medoids diminue moins rapidement que celle de K-Means.
Cependant, l'homogénéité des clusters de cet algorithme croit plus faiblement que celle de K-Means. Ainsi, les clusters mis en place par K-Medoids sont moins homogènes, c'est-à-dire qu'ils contiennent plus de familles de données.

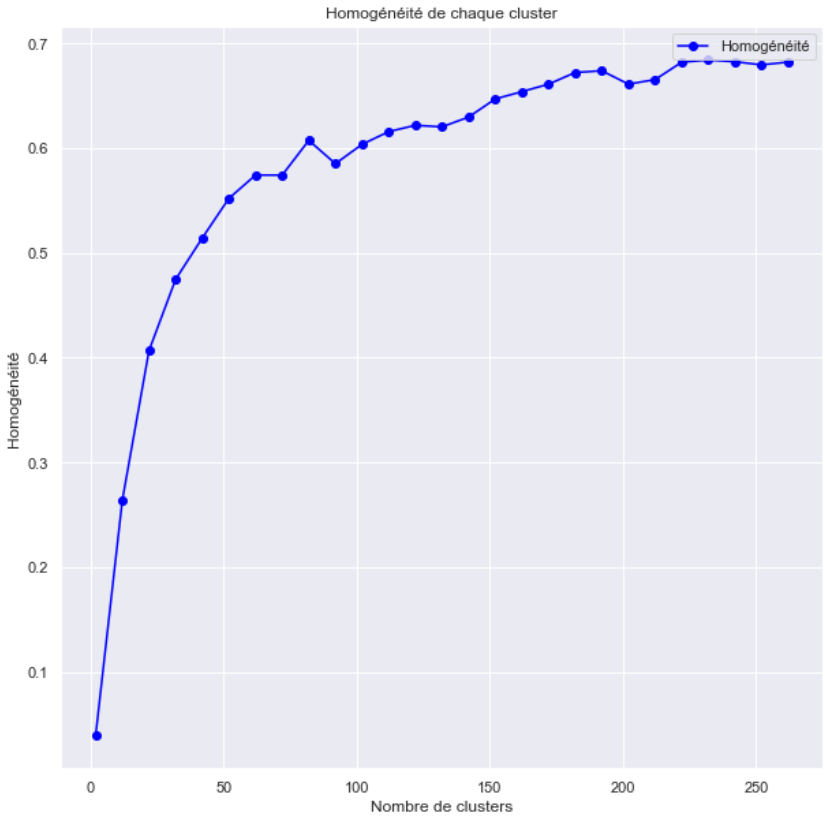

Figure 14 : tracé de l'homogénéité en fonction du nombre de clusters

Enfin, sans grande surprise à la vue des autres indices de K-Medoids, la précision de cet algorithme en fonction du nombre de clusters en paramètre est également plus faible que celle de K-Means.

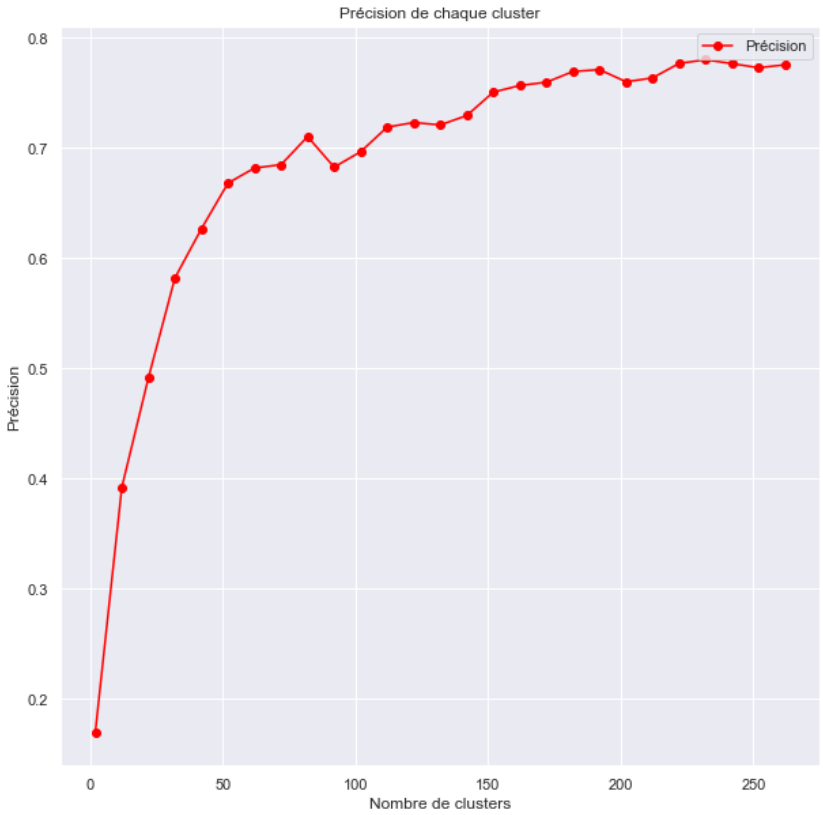

Figure 15 : tracé de la précision en fonction du nombre de clusters

Partie 3 : Une autre approche

On a obtenu de bons résultats avec les méthodes précédentes. Cependant, il nous fallait pour cela accepter en contrepartie des exécutions longues et lourdes, demandant beaucoup de ressources. Cela était dû au fait que nous avions choisi de travailler sur les 784 pixels de chaque image.
On peut essayer une autre approche.
Sur chaque ligne, on compte le nombre de pixels noirs, et on fait la moyenne des valeurs des pixels non noirs. On fait la même chose pour chaque colonne. On se retrouve alors avec 28 * 2 + 28 * 2 = 112 valeurs par image, au lieu de 784.
Dans cette partie, on veut juste comparer les résultats obtenus précédemment avec les résultats que peut nous donner cette nouvelle approche simplifiée.
On commence par s'intéresser à l'algorithme K-Means.
Comme on s'y attendait, le temps d'exécution est plus faible que si l'on travaille sur l'ensemble des données.

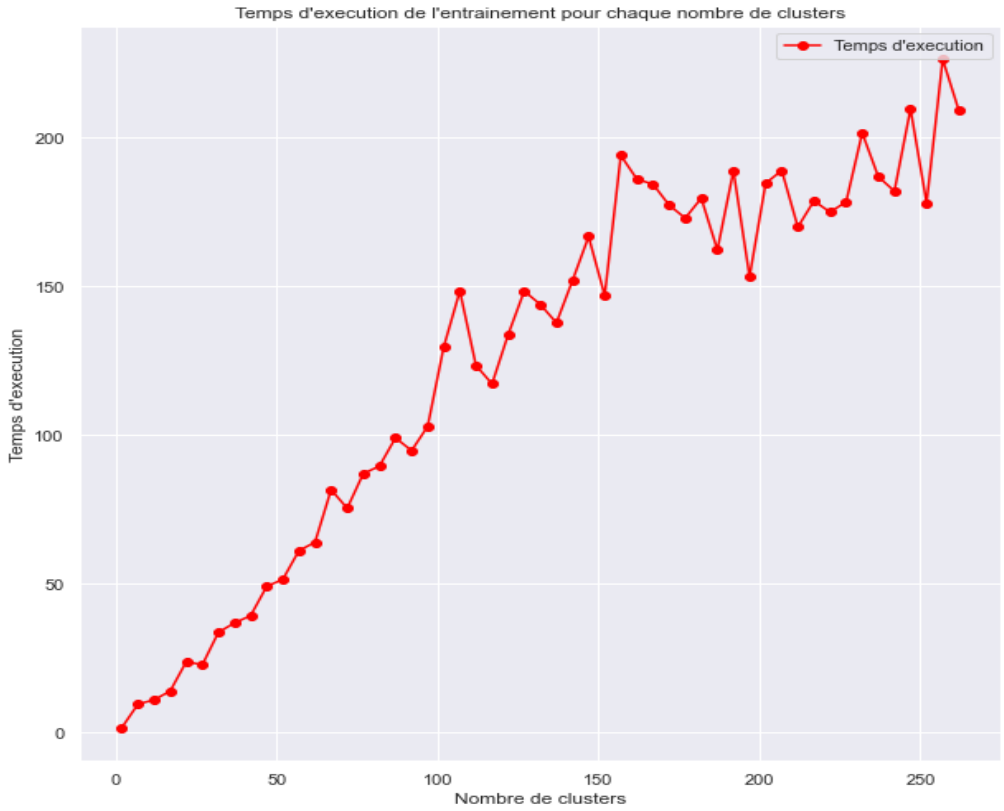

Figure 16 : tracé du temps d'exécution en fonction du nombre de clusters

Sur la méthode précédente, on atteignait jusqu'à 120 secondes pour 80 clusters. Avec cette méthode, pour 80 clusters, le temps d'exécution a diminué de plus de 20 secondes.
On remarque aussi que la valeur de l'indice de Calinski-Harabasz diminue fortement et rapidement.

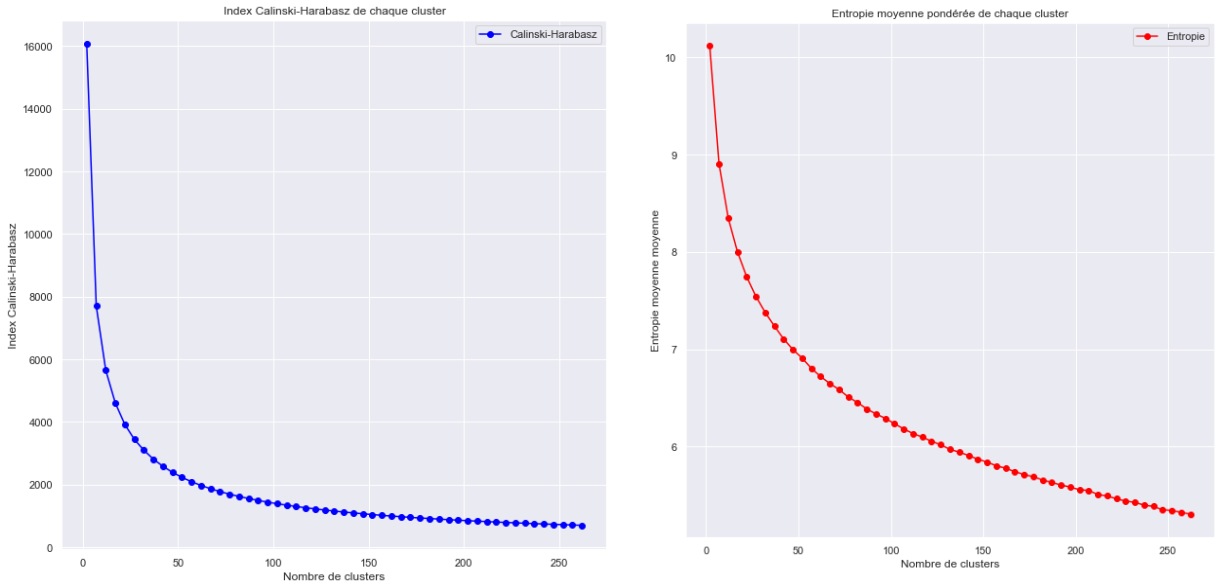

Figure 17 : tracés des indices en fonction du nombre de clusters

Les valeurs d'entropie semblent relativement proches de celles de la méthode précédente cependant.
Enfin, comme on pouvait s'y attendre, l'homogénéité des clusters est moins bonne que précédemment.

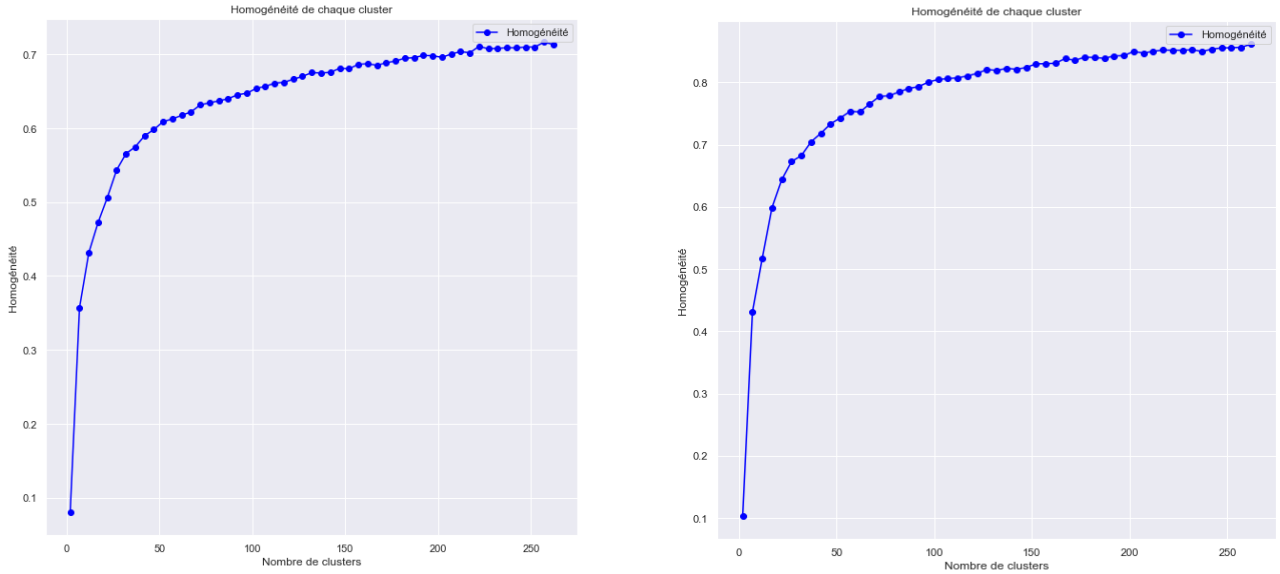

Figure 18 : comparaison des tracés de l'homogénéité en fonction des clusters. A droite, le tracé correspond à la méthode de l'histogramme ; à gauche, le tracé est celui de la méthode sans perte d'information

Les clusters initiaux sont bien plus homogènes et denses que ceux produits par cette nouvelle méthode. Il en va de même pour la précision des clusters.

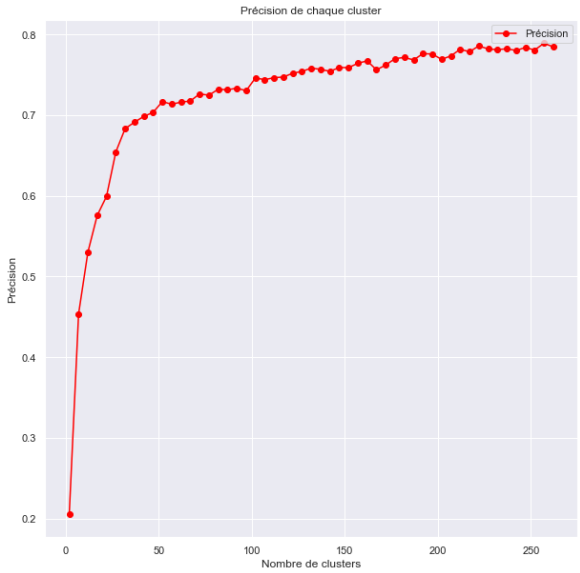

Figure 19 : tracé de la précision en fonction du nombre de clusters

Cependant, cette méthode n'est pas forcément à écarter, elle peut être préférée à la méthode initiale, sans perte d'information. En effet, si la densité et l'homogénéité des clusters n'ont pas besoin d'être aussi élevées que possible, il peut être préférable de choisir la deuxième méthode, moins exigeante en termes de ressources nécessaires et en temps d'exécution.

Les mêmes observations et conclusions peuvent être tirées si on décide d'utiliser l'algorithme K-Medoids, comme on peut le constater sur les graphes ci-dessous.

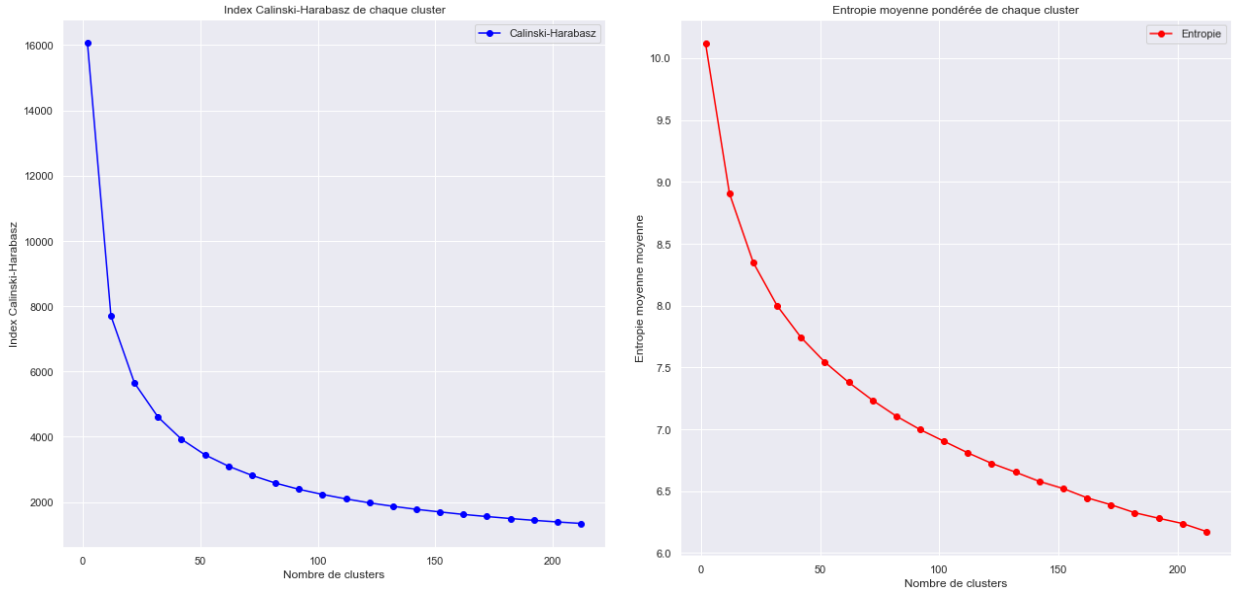

Figure 20 : tracé des indices de K-Medoids en fonction du nombre de clusters

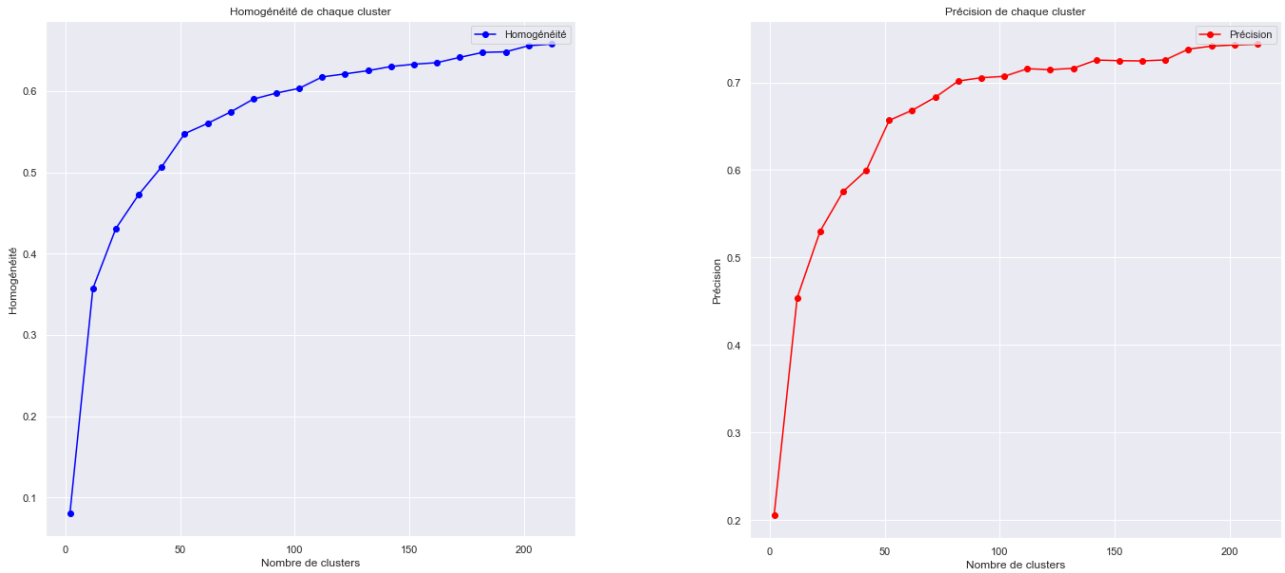

Figure 21 : tracé de la précision et homogénéité de K-Medoids en fonction du nombre de clusters

Cependant, si on accepte des résultats en-dessous de la méthode sans perte d'information et qu'on veut implémenter un algorithme K-Medoids, cette méthode peut s'avérer très intéressante. En effet, l'un des inconvénients auxquels on avait fait face avec la méthode précédente était le temps d'exécution. Avec cette méthode de l'histogramme, les temps d'exécution sont considérablement réduits. Cependant, cette méthode a un autre point négatif handicapant : avec K-Means, on ne peut pas du tout visualiser les centres car on ne peut pas les reconstruire.

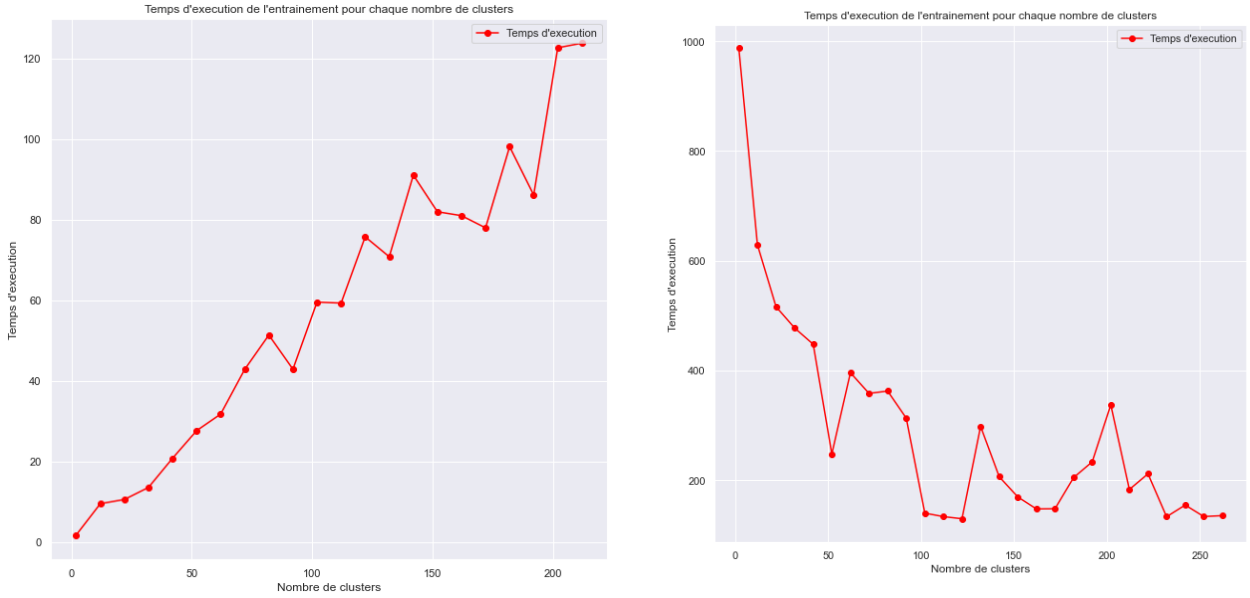

Figure 22 : comparaison des tracés du temps d'exécution en fonction des clusters. A droite, le tracé correspond à la méthode de l'histogramme ; à gauche, le tracé est celui de la méthode sans perte d'information

On remarque bien que le minimum du temps d'exécution du K-Medoids sur la figure de droite reste supérieur au maximum sur la figure de gauche.