Télécom SudParis

MAT 4202

Projet - 17 août 2023

Apprentissage, classification automatique, data mining

Analyse de signatures manuscrites pour la vérification d'identité

Élèves :
Nour Rammal
Théophile Schmutz

Enseignant :
Nesma Houmani

Résumé

Dans ce rapport, nous proposons une approche de visualisation des signatures de la base de données en les regroupant en catégories selon leur complexité. Pour ce faire, nous appliquons des techniques de classification non supervisée basées sur une modélisation statistique des signatures par un modèle à mélange de gaussiennes (GMM) couplée à la mesure d'entropie différentielle. La mesure de complexité est considérée comme un descripteur de la signature et est calculée en amont. Nous générerons des catégories de signatures selon leur complexité de manière indépendante de leur appartenance. Les résultats seront visualisés et interprétés.

Table des matières

1 Introduction

1.1 Base de données MCYT-100

Avec l'avènement des signatures électroniques dans de nombreuses actions en ligne qui nécessitent une preuve d'identité, le problème de la vérification d'identité devient plus complexe.

La base de données MCYT-100 est une collection de signatures manuscrites acquises sur un dispositif numérique, utilisée pour l'étude et l'analyse de la dynamique du geste lors de la signature. Elle comprend les signatures de 100 individus différents, avec 25 signatures pour chaque personne, soit un total de 2500 fichiers de données [1].

Chaque fichier de données contient les coordonnées du stylo utilisé pour la signature, ainsi que d'autres informations telles que l'angle avec le support et la pression exercée sur celui-ci. Ces données permettent de caractériser la dynamique du geste de signature de chaque individu, en mettant en évidence les variations de pression et d'angle du stylo pendant la signature.

1.1.1 Visualisation

Les signatures peuvent être visualisée en traçant sur un graphe la séquence de points représentant une signature donnée. Ci-dessous quelques exemples de signatures :

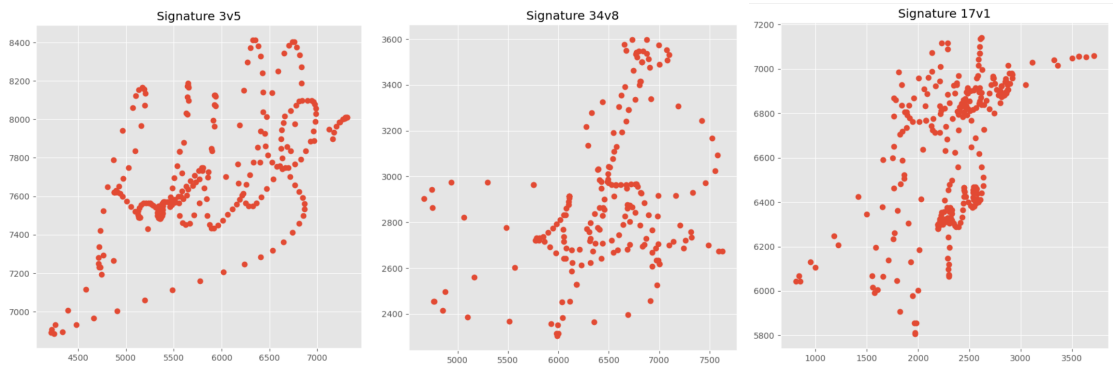

FIGURE 1 – Quelques exemples de signatures

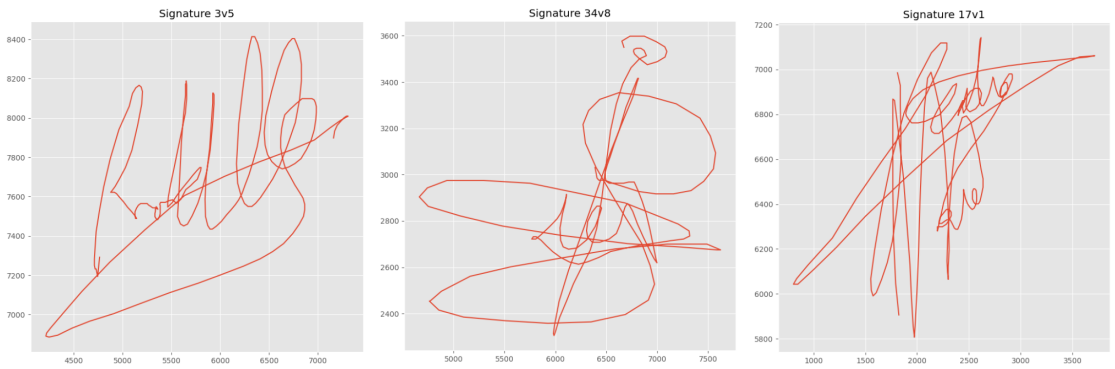

FIGURE 2 – Ces mêmes signatures en rejoignant les points

1.1.2 Mesure de complexité

Comme nous l'avions vu dans la partie précédente, une signature est représentée par une séquence de points. Si cette séquence de points est considérée comme le résultat d'une variable aléatoire, le concept d'entropie peut être utilisé pour estimer le degré de désordre associé à cette variable aléatoire, comme le suggèrent Kahindo, Garcia-Salicetti et Houmani dans leur papier [2]. C'est cette notion d'entropie qui va nous permettre de définir l'indice de complexité associé à chaque signature.

Pour chaque signature, l'indice de complexité se calcule comme suit : la composante gaussienne qui donne la probabilité la plus élevée est attribuée à chaque point $(x(t), y(t))$. Ensuite, nous attribuons au point son entropie différentielle $H(t)$ correspondante. Pour un échantillon de signature de longueur N, l'indice de complexité est défini comme suit :

$$C = \frac{\sum_{t=1}^{N} H(t)}{N} \qquad \text{Indice de Complexité} \qquad (1)$$

Ci-dessous une visualisation des indices de complexité des signatures avec des modélisation à 4, 8 et 24 gaussiennes :

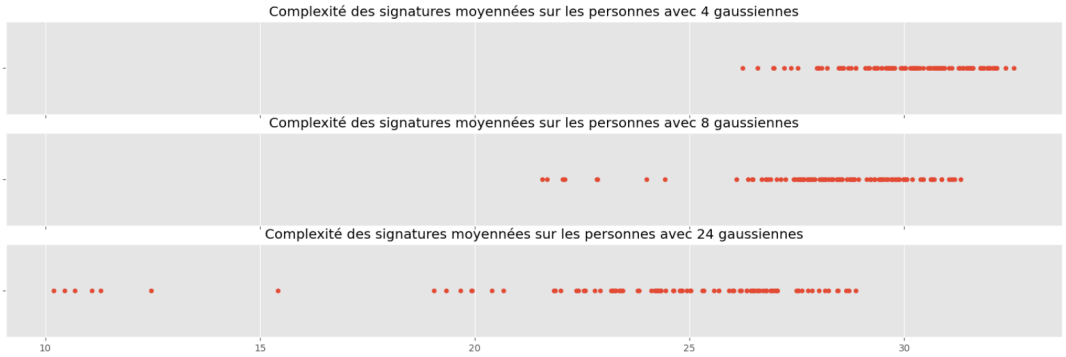

FIGURE 3 – Complexité moyennes pour les modélisations avec 4, 8 et 24 Gaussiennes

FIGURE 4 – Les complexités des 100 personnes pour les modélisations avec 4, 8 et 24 Gaussiennes

Nous pouvons d'ores et déjà constater une corrélation entre l'indice de complexité et le choix du nombre de gaussiennes utilisées dans la modélisation des signaux associés aux signatures.

1.2 Algorithmes non supervisés utilisés

Nous allons détailler ici les algorithmes utilisés dans la suite du projet. A noter que nous allons passer succinctement sur le fonctionnement des algorithmes K-Moyennes et K-Médoïdes car nous les avons déjà traités en Travaux Pratiques.

1.2.1 K-Moyennes

L'algorithme des K-Moyennes est une technique de clustering en apprentissage non supervisé, qui permet de trouver des structures dans des données en les divisant en groupes homogènes. L'objectif est de maximiser la similarité intra-clusters et de minimiser la similarité inter-clusters.

Les étapes de l'algorithme des K-Moyennes sont les suivantes :

1. Initialiser K centres de cluster.

2. Attribuer chaque point de données au cluster dont le centre est le plus proche en terme de distance euclidienne.
3. Calculer la moyenne de tous les points associés à chaque centre de cluster.
4. Déplacer chaque centre de cluster vers la moyenne calculée à l'étape 3.
5. Répéter les étapes 2 à 4 jusqu'à ce que la position des centres de cluster ne change plus ou que le nombre maximum d'itérations soit atteint.

L'utilisation de l'algorithme des K-Moyennes est intéressante lorsque le nombre de clusters est connu ou peut être estimé, ici $K = 3$. Elle l'est aussi lorsque les données sont de grandes dimension ou que le nombre d'observation est important car l'algorithme est rapide et scalable, ce qui est également le cas ici. En revanche, l'algorithme des K-Moyennes est le plus performant lorsque les données sont facilement séparables en groupes distincts, propriété qui n'est pas observable lorsque l'on observe les indices de complexité des signatures que ce soit pour des modélisations à 4, 8 ou 24 gaussiennes. Nous garderons cela à l'esprit lorsque nous interpréterons les résultats des classification des signatures en terme de complexité.

1.2.2 K-Médoïdes

L'algorithme des K-Médoïdes est une technique de clustering en apprentissage non supervisé qui est similaire à l'algorithme des K-Moyennes, mais qui utilise des points de données réels comme centres de cluster (c'est-à-dire appartenant à l'ensemble des données) appelés médoïdes, plutôt que des moyennes.

Le fonctionnement de l'algorithme des K-Médoïdes est similaire à celui des K-Moyennes, mais au lieu de calculer la moyenne de tous les points associés à chaque centre de cluster, il choisit un point de données réel qui minimise la distance totale entre les points associés à ce centre de cluster. Cette distance est mesurée en termes de distance de Manhattan, plutôt qu'en termes de distance euclidienne.

Les étapes de l'algorithme des K-Médoïdes sont les suivantes :

1. Initialiser K médoïdes.
2. Attribuer chaque point de données au médoïde le plus proche.
3. Pour chaque médoïde, sélectionner le point de données réel qui minimise la distance totale aux autres points de données associés à ce médoïde, et le définir comme nouveau médoïde.
4. Répéter les étapes 2 et 3 jusqu'à ce que la position des médoïdes ne change plus ou que le nombre maximum d'itérations soit atteint.

En comparaison avec l'algorithme des K-Moyennes, l'algorithme des K-Médoïdes est moins sensible à l'initialisation de K et le temps de calcul est un plus important. Les appréhensions concernant la séparabilité des données en classes distinctes selon leur indice de complexité sont valables pour cette méthode de classification également.

1.2.3 DBSCAN

L'algorithme DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) est un algorithme de clustering en apprentissage non supervisé qui se base sur la densité des données pour identifier les clusters. Contrairement aux algorithmes K-Moyennes et

K-Médoïdes qui nécessitent de spécifier le nombre de clusters à l'avance, DBSCAN est capable d'identifier le nombre de clusters de manière automatique.

Le fonctionnement de l'algorithme DBSCAN consiste à regrouper les points de données qui sont suffisamment proches les uns des autres en termes de densité, et à identifier les points qui sont considérés comme des valeurs aberrantes ou du bruit. L'algorithme DBSCAN utilise 2 paramètres : la distance ϵ et et le nombre minimum de points $MinPts$ devant se trouver dans un rayon ϵ pour que ces points soient considérés comme un cluster. L'idée de base de l'algorithme est ensuite, pour un point donné, de récupérer son ϵ-voisinage et de vérifier qu'il contient bien $MinPts$ points ou plus. Ce point est alors considéré comme faisant partie d'un cluster. On parcourt ensuite l'ϵ-voisinage de proche en proche afin de trouver l'ensemble des points du cluster.

Les étapes de l'algorithme DBSCAN sont les suivantes :

1. Choisir un point de données au hasard qui n'a pas encore été attribué à un cluster.
2. Trouver tous les points de données à une distance maximale (appelée "rayon epsilon") du point de départ.
3. Si le nombre de points dans cette zone est supérieur à un certain seuil (appelé "nombre minimum de points"), créer un nouveau cluster et ajouter tous les points de cette zone à ce cluster. Sinon, marquer le point de départ comme une valeur aberrante ou du bruit.
4. Répéter les étapes 1 à 3 pour tous les points de données restants qui n'ont pas encore été attribués à un cluster.

L'algorithme DBSCAN est particulièrement utile lorsque les clusters ont des formes arbitraires (non uniforme) ou complexes, car l'algorithme est capable de détecter les zones de densité dans les données. Au-vu des tracés (Fig. 3 et 4), nous pouvons nous attendre à ce que les clusters ne soient pas uniformes rendant ainsi l'utilisation de cet algorithme pertinente.

Comment fixer ϵ ? Comme dit précédemment, dans cet algorithme deux points sont clés : la métrique et ϵ. Concernant la première, nous avons fait le choix, dans la suite, d'utiliser la distance euclidienne. Reste maintenant à savoir comment choisir le bon epsilon.

Si ϵ est trop petit le ϵ-voisinage est trop faible et toutes les observations du jeu de données sont considérées comme des anomalies. C'est le cas à gauche sur la figure ci-dessous.

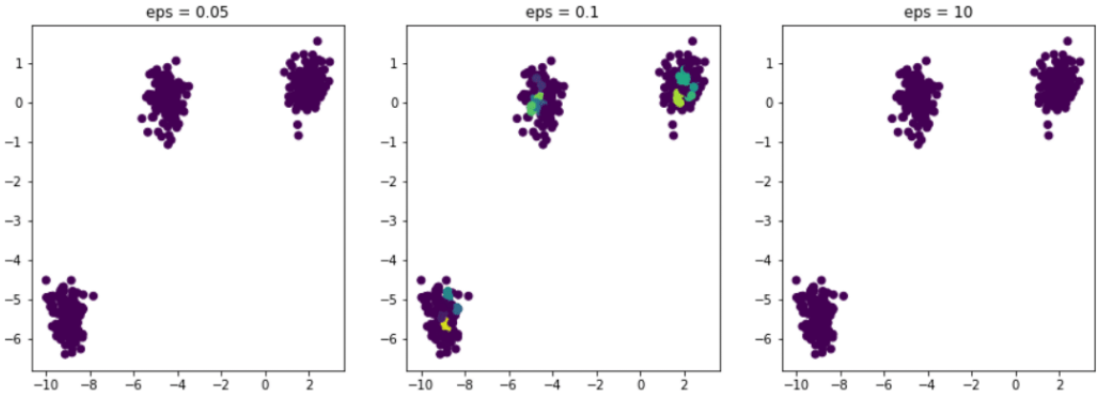

FIGURE 5 – Exemple de DBSCAN sur un exemple simple

A contrario si epsilon est trop grand chaque observation contient dans son ϵ-voisinage toutes les autres observations du jeu de données. Par conséquent nous n'obtenons qu'un unique cluster. Il est donc très important de bien calibrer le ϵ pour obtenir un partitionnement de qualité. Une méthode simple pour optimiser le ϵ consiste à regarder pour chaque observation à quelle distance se situe son voisin le plus proche. Ensuite il suffit de fixer un ϵ tel qu'une part « suffisamment grande » des observations aient une distance à son plus proche voisin inférieure à ϵ. Par « suffisamment grande » on entend 90% des observations qui doivent avoir au moins un voisin dans leur ϵ-voisinage.

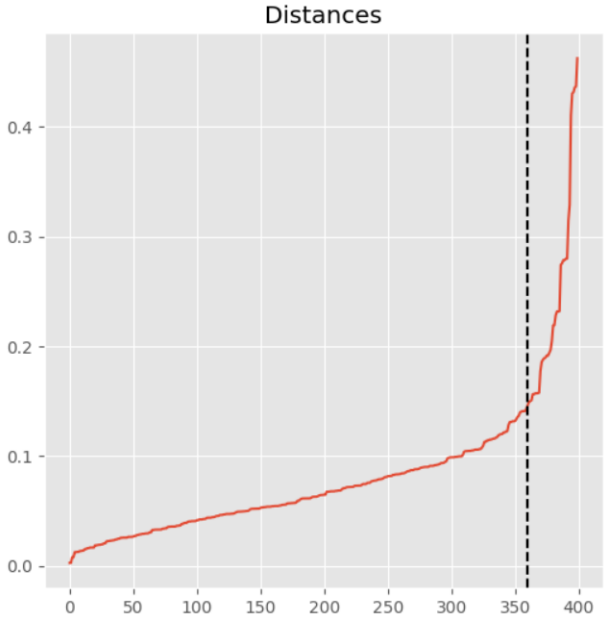

FIGURE 6 – Distance

Comme nous l'avons vu précédemment nous allons choisir un ϵ de tel sorte que 90% des observations aient une distance au proche voisin inférieure à ϵ. Dans notre exemple 0.15 semble convenir. Maintenant que nous avons optimisé notre ϵ nous pouvons effectuer notre partitionnement en utilisant DBSCAN qui est déjà implémenté dans Scikit-Learn.

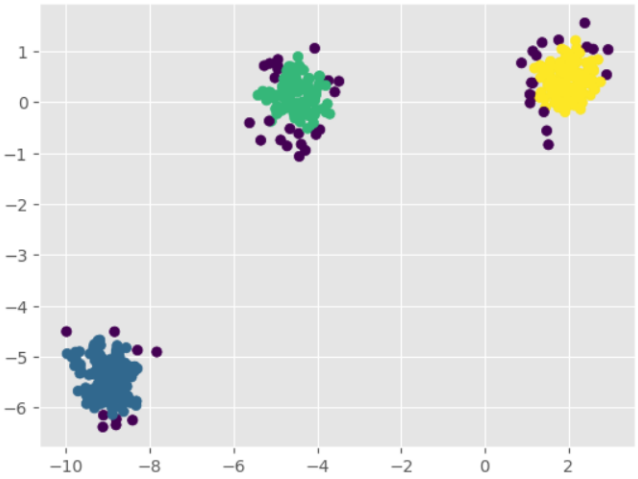

FIGURE 7 – Exemple

Cependant, un problème subsiste. En effet, dans le cadre de ce rapport nous souhaitons partitionner nos données en exactement trois clusters. Or, DBSCAN fixe lui même le nombre de clusters.

Notre méthode pour trouver la valeur optimale ϵ dans notre problème de clustering consiste en deux étapes. Tout d'abord, nous cherchons une valeur ϵ qui fonctionne de manière générale en utilisant la méthode détaillée précédemment (Fig. 6). Ces métriques nous aident à estimer une valeur initiale ϵ_0 qui pourrait être efficace pour notre ensemble de données.

Une fois que nous avons obtenu cette valeur initiale, nous cherchons à obtenir exactement trois clusters. Pour y parvenir, nous itérons sur des valeurs d'epsilon supérieures à la valeur optimale initiale ϵ_0. Nous commençons par la valeur optimale trouvée précédemment et augmentons progressivement la valeur d'epsilon jusqu'à ce que nous obtenions exactement trois clusters distincts.

L'idée derrière cette approche est d'ajuster l'échelle de distance utilisée par l'algorithme DBSCAN pour déterminer la proximité des points. En augmentant la valeur ϵ, nous permettons aux points de former des clusters plus larges, ce qui peut conduire à l'émergence de trois clusters distincts plutôt que d'un seul cluster global ou de plusieurs petits clusters.

Cette méthode itérative nous permet d'explorer différentes échelles de distance et de trouver la valeur ϵ qui produit exactement trois clusters souhaités dans notre ensemble de données. Cela nous permet d'adapter l'algorithme DBSCAN à notre problème spécifique et de répondre à notre exigence de clustering en trois clusters. Ainsi, dans la suite (Sous partie 2.4), les valeurs d'epsilon utilisées seront issues de cette procédure d'optimisation de la valeur ϵ spécifique à notre problème.

Ainsi, dans notre cas :

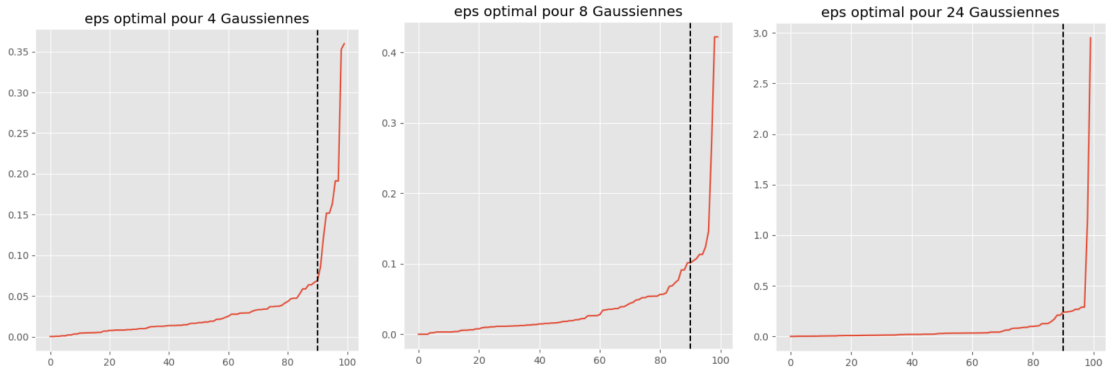

FIGURE 8 – ϵ_0 pour 4, 8 et 24 gaussiennes

Nous trouvons dans les cas initiaux :

$$\epsilon_{0\,4G} \approx 0.075 \quad \epsilon_{0\,8G} \approx 0.1 \quad \epsilon_{0\,24G} \approx 0.25$$

En itérant petit à petit pour diminuer le nombre de clusters artificiellement, nous trouvons $\epsilon_{opt} = 0.2,\ 1,\ 1$. Ainsi, pour 4 gaussiennes, nous utiliserons un epsilon de 0.2 ; pour 8 et 24 gaussiennes, nous utiliserons un epsilon de 1.

$$\epsilon_{opt\,4G} \approx 0.2 \quad \epsilon_{opt\,8G} \approx 1 \quad \epsilon_{opt\,24G} \approx 1 \tag{2}$$

2 Classification non-supervisées des personnes

La base de données MCYT-100 est utilisée pour étudier différentes méthodes de classification des signatures manuscrites, en particulier en utilisant des modèles de mélange gaussien (GMM) avec 4, 8 ou 24 composantes gaussiennes. Elle permet également de comparer les résultats de classification obtenus avec ces différents modèles, et d'évaluer leur efficacité pour la classification des individus en fonction de la complexité de leurs signatures.

2.1 K-moyennes

Nous appliquons l'algorithme des K-Moyennes aux mélanges de 4, 8 et 24 Gaussiennes et nous obtenons les résultats ci-dessous. Notons que les représentants des classes sont indiqués par une croix noire.

2.1.1 Cas avec 4 Gaussiennes

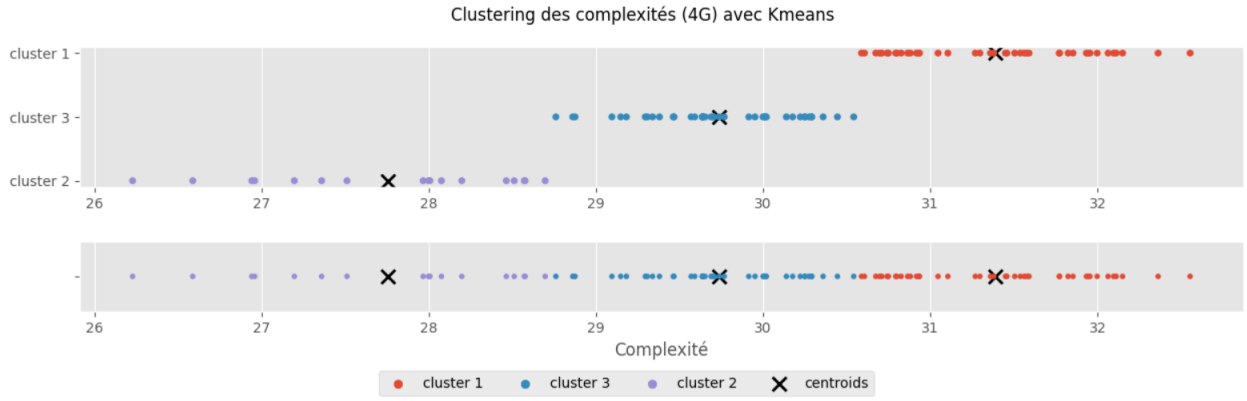

FIGURE 9 – Clustering des complexités avec KMeans et 4 Gaussiennes

2.1.2 Cas avec 8 Gaussiennes

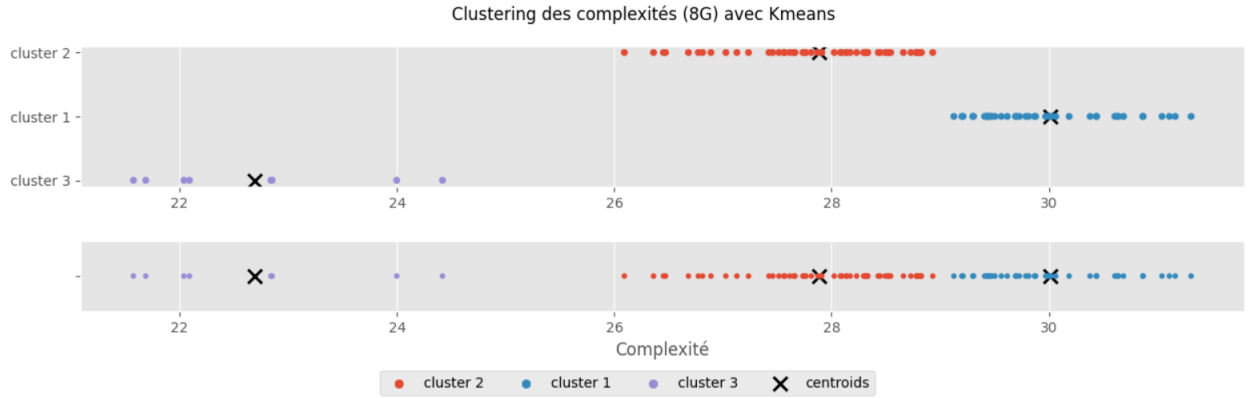

FIGURE 10 – Clustering des complexités avec KMeans et 8 Gaussiennes

2.1.3 Cas avec 24 Gaussiennes

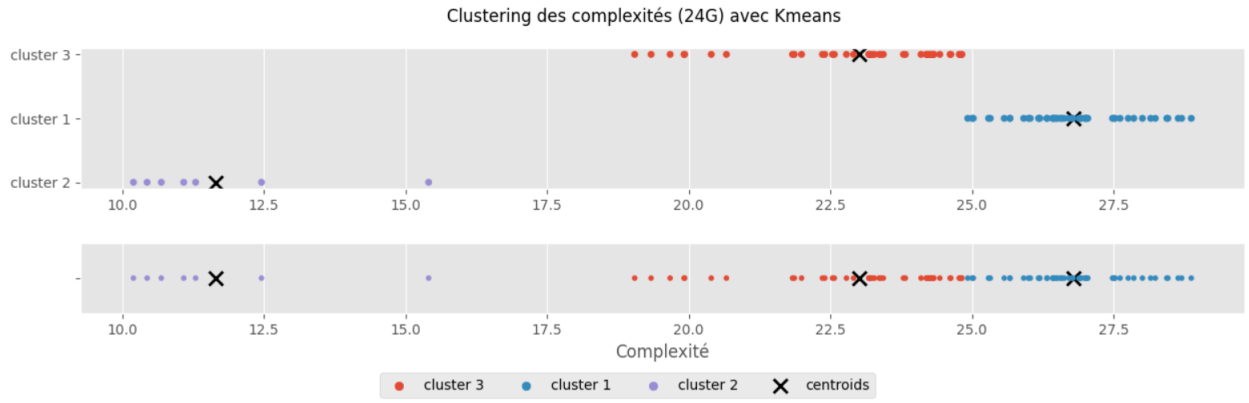

FIGURE 11 – Clustering des complexités avec KMeans et 24 Gaussiennes

2.2 K-médoïdes

Nous appliquons l'algorithme des K-Médoïdes aux mélanges de 4, 8 et 24 Gaussiennes et nous obtenons les résultats ci-dessous. Notons que les représentants des classes sont indiqués par une croix noire.

2.2.1 Cas avec 4 Gaussiennes

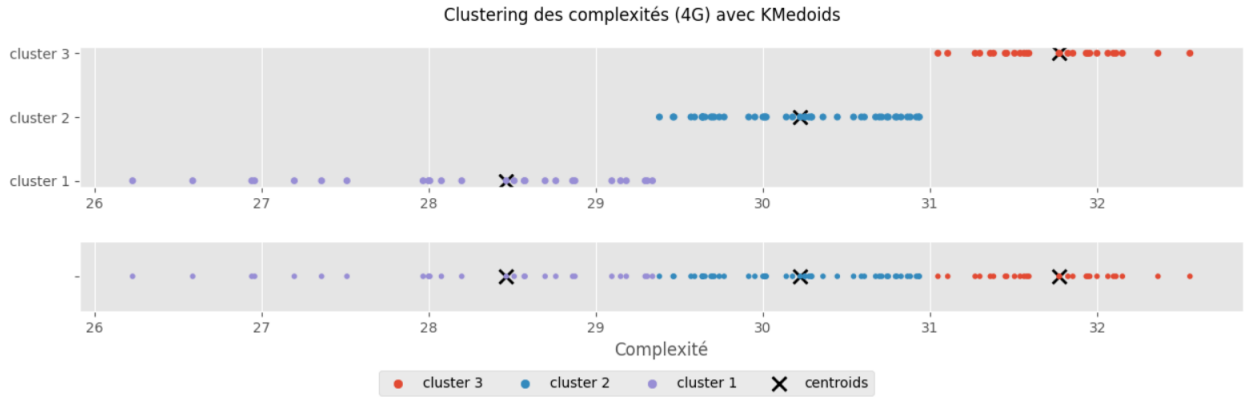

FIGURE 12 – Clustering des complexités avec KMedoids et 4 Gaussiennes

2.2.2 Cas avec 8 Gaussiennes

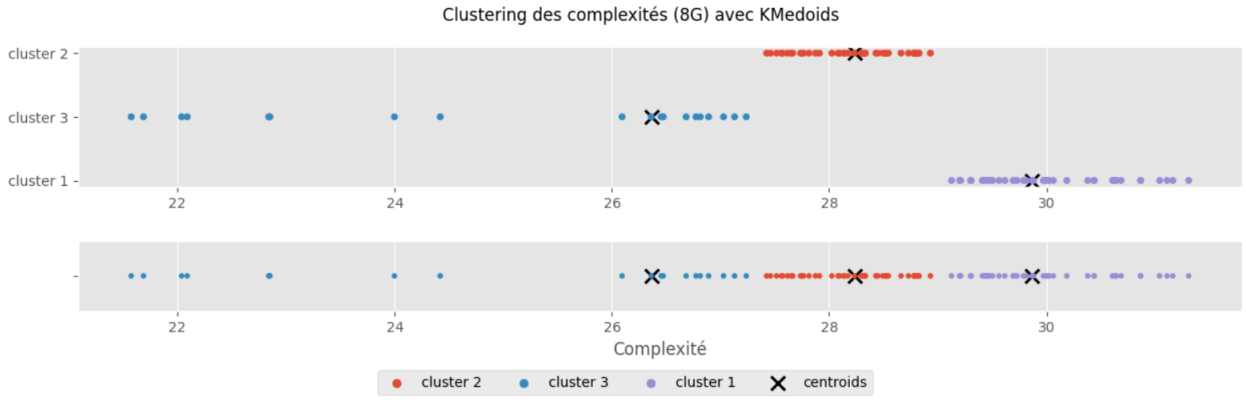

FIGURE 13 – Clustering des complexités avec KMedoids et 8 Gaussiennes

2.2.3 Cas avec 24 Gaussiennes

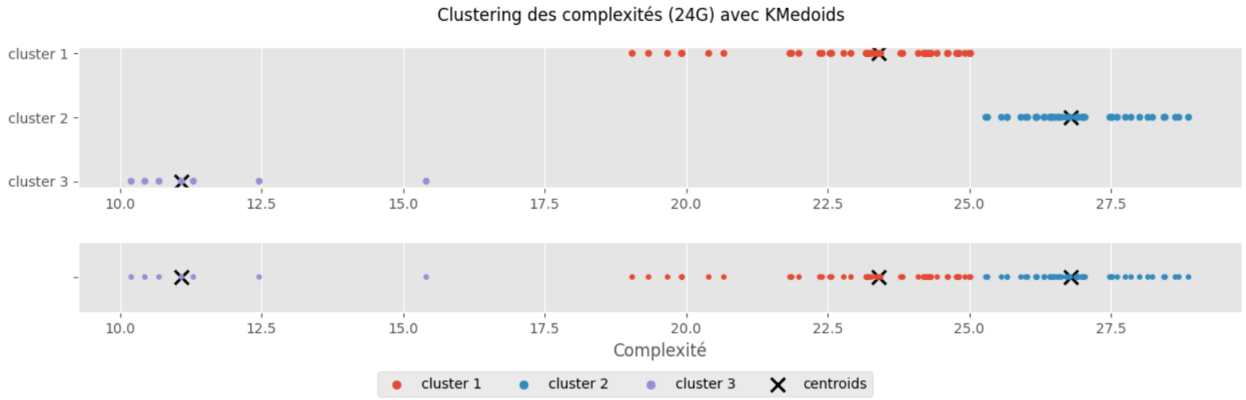

FIGURE 14 – Clustering des complexités avec KMedoids et 24 Gaussiennes

2.3 Analyse des Résultats obtenus avec K-moyennes et K-médoïdes

Pour commencer, nous examinons l'impact des différentes méthodes de clustering et des mesures d'entropie différentielle sur la distribution des individus dans chaque cluster. Nous avons utilisé la méthode de classification k-means pour créer 3 clusters correspondant à des signatures de forte, moyenne et faible complexité. Chaque individu est alors classé en fonction de sa complexité relative par rapport aux autres individus de la base de données. Nous avons utilisé ces noms de clusters pour faciliter l'interprétation et la présentation des résultats. Après l'application de la méthode k-means, nous avons constaté une forte concentration d'individus dans les classes de forte et moyenne complexité (Voir Fig. 15). Cela indique que les complexités des signatures sont principalement regroupées autour d'une complexité élevée, et que peu d'individus ont des signatures simples.

En prenant en compte la variation des modèles GMM, nous avons observé l'influence du nombre de gaussiennes utilisées. Plus le nombre de gaussiennes du modèle est élevé, plus la répartition des individus est déséquilibrée entre les complexités forte et moyenne d'un côté et faible de l'autre. Un nombre important de gaussiennes pour un modèle GMM

permet d'obtenir une meilleure représentation des petits groupes d'individus détachés de la masse, ce qui permet d'obtenir une meilleure précision sur les regroupements des individus. Cela nous amène à conclure que la base de données est principalement composée d'individus ayant une complexité moyenne plutôt élevée (environ 90 pour cent) et d'un autre petit groupe d'individus ayant une faible complexité. Les modèles GMM à 8 et 24 gaussiennes nous apportent donc plus de précisions sur les mesures de complexité des signatures, mais il existe une limite à un nombre trop important de gaussiennes, qui est le temps de calcul des mesures comparé à un modèle ayant peu de gaussiennes. L'écart constaté entre 8G et 24G est assez faible, il est donc pertinent de ne pas considérer plus de 24 gaussiennes car le gain d'information risque d'être très faible par rapport à l'augmentation en temps de calcul.

En ce qui concerne la méthode k-medoids, les mêmes résultats ont été observés lors de la classification (Voir Fig. 16). Cela indique que ce sont les gaussiennes qui influencent la répartition des individus dans les classes, plutôt que la méthode de clustering elle-même.

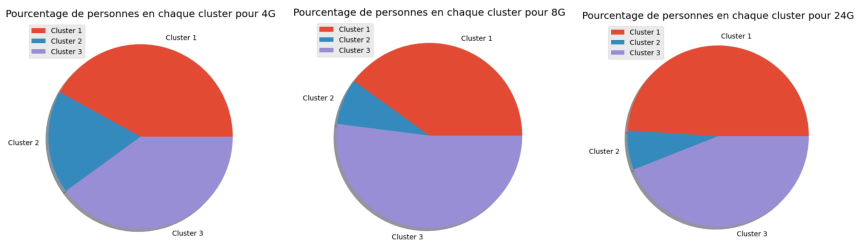

FIGURE 15 – Pourcentage de personnes en chaque cluster avec KMeans

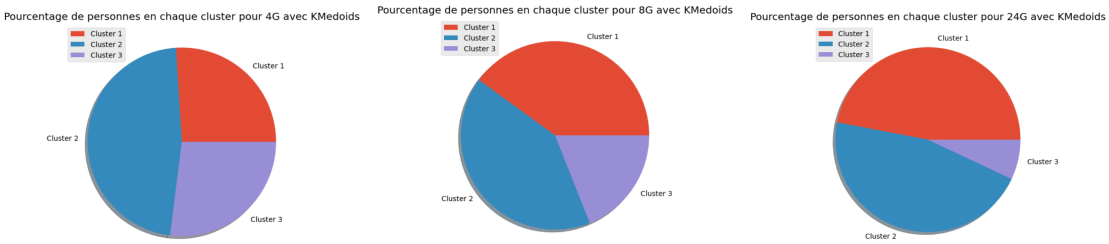

FIGURE 16 – Pourcentage de personnes en chaque cluster avec KMedoids

Nous avons ensuite procédé à une analyse des centres des classes qui ont été obtenus à partir des deux méthodes de clustering. Dans les deux cas, l'ajout de gaussiennes dans le modèle GMM a entraîné une augmentation de la distance entre les centres et donc une séparation plus importante entre les classes. Cette observation n'est pas surprenante, car c'est l'objectif principal des modèles GMM à haute précision. Visuellement, les centroïdes sont également plus éloignés les uns des autres lorsque le nombre de gaussiennes augmente. Par conséquent, le modèle final est plus pertinent et interprétable lorsque la distinction entre les classes ayant des complexités moyennes différentes est importante. Comme précédemment mentionné, plus le nombre de gaussiennes dans le modèle GMM est élevé, plus les petits regroupements d'individus sont visibles et ont du poids dans la classification. Cela signifie que la masse a moins d'influence sur les centres des classes, ce qui explique la plus grande distance observée entre eux. Dans la suite de notre analyse,

nous verrons que l'inertie entre les clusters, qui nous permet de mesurer la séparation des classes, variera en fonction des différents modèles GMM et confirmera nos interprétations concernant les représentants des clusters.

2.4 DBSCAN

Nous appliquons l'algorithme DBSCAN aux mélanges de 4, 8 et 24 Gaussiennes et nous obtenons les résultats ci-dessous. Nous avons un hyperparamètre ϵ qui corresponds à la distance maximale entre deux échantillons pour que l'un soit considéré comme étant dans le voisinage de l'autre.

2.4.1 Cas avec 4 Gaussiennes

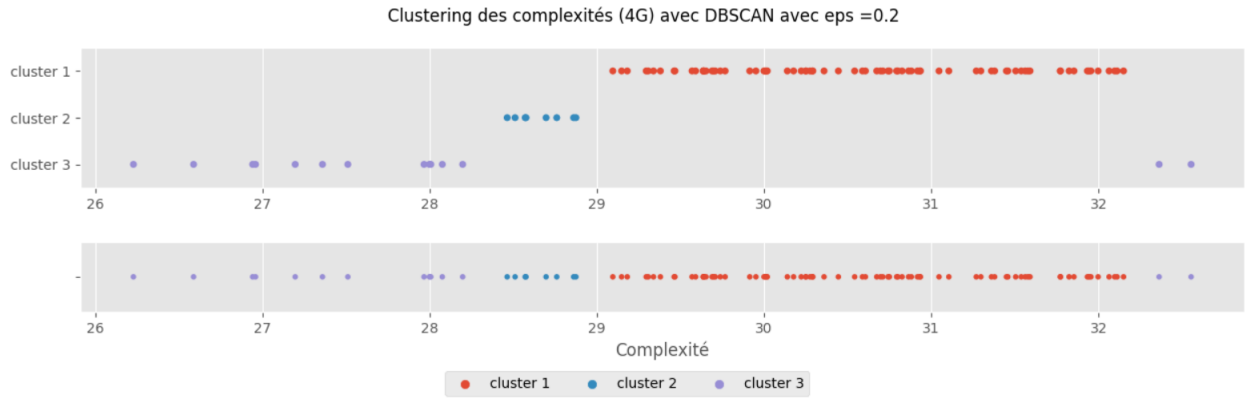

FIGURE 17 – Clustering des complexités avec DBSCAN et 4 Gaussiennes et $\epsilon = 0.2$

2.4.2 Cas avec 8 Gaussiennes

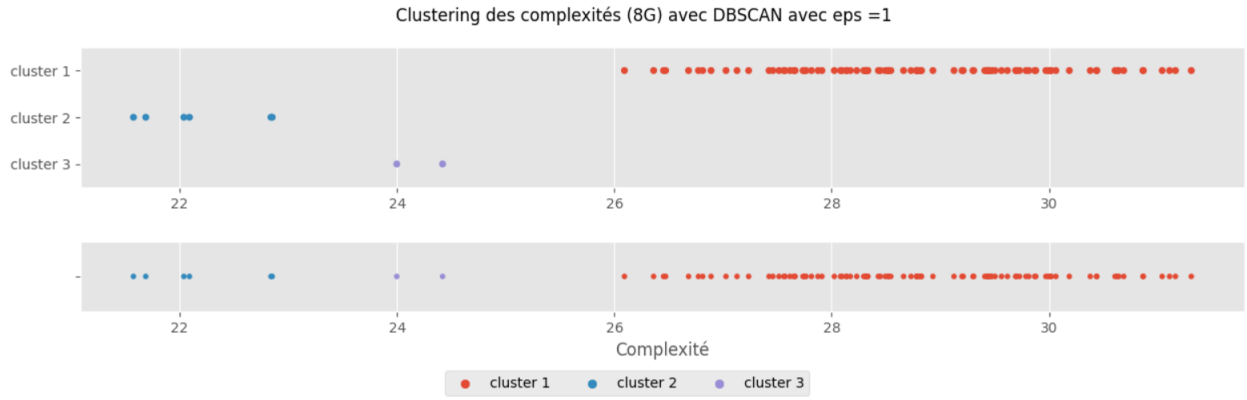

FIGURE 18 – Clustering des complexités avec DBSCAN et 8 Gaussiennes

2.4.3 Cas avec 24 Gaussiennes

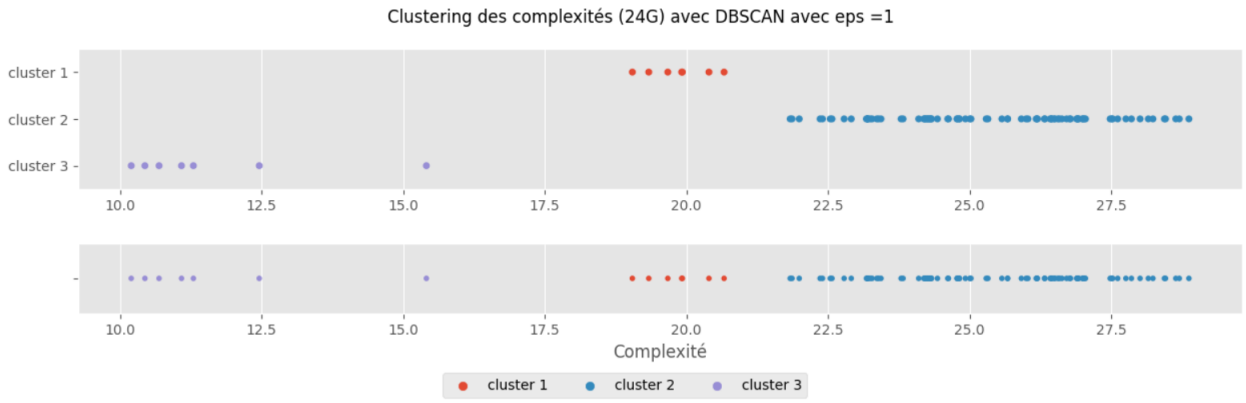

FIGURE 19 – Clustering des complexités avec DBSCAN et 24 Gaussiennes

2.5 Comparaison

Le but de l'utilisation de DBSCAN est de trouver des groupes de points qui sont proches les uns des autres et de les regrouper dans un même cluster. Cependant, contrairement à K-moyennes et K-médoïdes, DBSCAN peut également trouver des points qui ne sont pas liés à un cluster et les considérer comme du bruit (outliers), mais cela n'est pas le cas ici.

Les graphiques montrent [1] que pour les trois groupes de Gaussiennes, DBSCAN a réussi à trouver trois clusters distincts en fonction de la complexité moyenne des signatures.

Notons cependant que dans le cas à 4 gaussiennes, deux points de complexités moyennes très élevés ont été classés dans le cluster complexité faible ce qui semble aberrant. Mis à part cela, nous retrouvons les mêmes résultats qu'avec les deux méthodes précédentes, soit une population composée majoritairement de personne dont les signatures sont de complexité moyenne plutôt élevée.

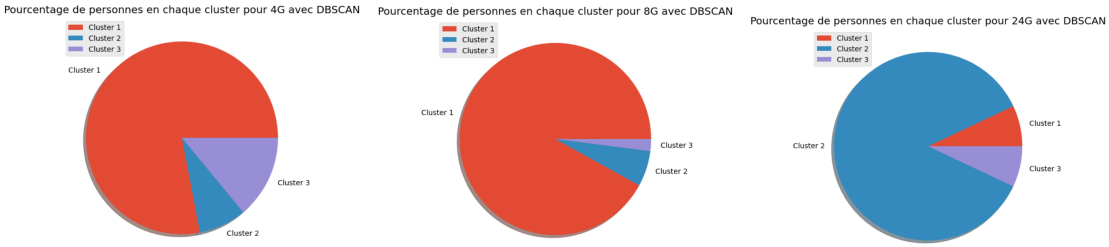

FIGURE 20 – Pourcentage de personnes en chaque cluster avec DBSCAN

1. Nous nous excusons des problèmes liés au couleurs de la légende qui change dans chaque diagramme.

2.6 Conclusion de la classification non-supervisées des personnes

Inerties intra et inter clusters Nous allons étudier l'impact des différents modèles que nous avons examinés jusqu'à présent sur les inerties intra et inter-clusters. Pour commencer, définissons rapidement ces termes.

L'inertie intra-cluster correspond à la somme de tous les écarts au carré entre les complexités moyennes des individus et la complexité du centre de leur classe. Cet indice nous informe sur la dispersion des valeurs à l'intérieur de chaque classe, elle permet de quantifier cette dispersion. L'inertie inter-classe donne un indice de dispersion plus global. En effet, l'inertie inter-classes, également appelée variance inter-classes, mesure la dispersion des moyennes des différents groupes dans un ensemble de données. Elle est calculée comme la somme pondérée des variances de chaque groupe, où la pondération correspond au nombre d'observations dans chaque groupe.

Mathématiquement, soient $G = \{e_i \mid i \in [|1; n|]\}$ est un groupe d'individus, de centre de gravité g, partitionné en k classes d'effectifs $n_1, \ldots, n_k$ qu'on appellera $G_1, \ldots, G_k$ qui ont comme centre de gravité $g_1, \ldots, g_k$. On définit l'inertie par :

$$I_t = \frac{1}{n} \sum_{e \in G} d(e, g)^2 \qquad \text{(inertie totale)} \tag{3}$$

$$I_{inter} = \frac{1}{n} \sum_{i=1}^{k} n_i d(g_i, g)^2 \qquad \text{(inertie interclasse)} \tag{4}$$

$$I_{intra} = \frac{1}{n} \sum_{i=1}^{k} \sum_{e \in G_i} d(e, g_i)^2 \qquad \text{(inertie intraclasse)} \tag{5}$$

A noter que l'inertie intra-clustes n'est pas une métrique disponible directement avec DBSCAN car cette méthode ne génère pas de cluster en tant que tel. Cependant, on peut calculer la somme des carrés des distances entre chaque point et son centre de cluster estimé (centre de gravité) pour chaque groupe identifié par DBSCAN. Cela pourrait être considéré comme une mesure similaire à l'inertie intra-cluster.

Modèle de données	**Algorithme**		
	KMeans	**KMedoids**	**DBSCAN**
4 Gaussiennes	30.08	47.13	61.13
8 Gaussiennes	50.80	69.83	147.12
24 Gaussiennes	183.04	99.78	294.20

TABLE 1 – Inertie intra-clusters pour KMeans, KMedoids et DBSCAN

Ici (Table 1) nous observons une augmentation significative de l'inertie intra classe lorsque le nombre de gaussiennes du modèle augmente. Ce résultat vient du fait que les modèles avec de nombreuses gaussiennes diminuent la compacité des individus dans chaque classe car ils étalent les valeurs mesurées. On a donc une nouvelle limite d'un modèle à grand nombre de gaussiennes qui est la dégradation de certains indices de qualité de classification. Plus la compacité des éléments d'une classe est faible, moins bon est le

clustering.

En analysant l'inertie inter-clusters pour les 3 algorithmes (Table 2), nous constatons que nous avons une meilleure séparation des clusters lorsque le modèle GMM est plus précis ce qui confirme nos propos concernant les représentants des classes. Plus le modèle possède de gaussiennes, plus il sera précis pour séparer les groupes d'individus de la base de données. Cet indice reflète la qualité du clustering, ce qui nous mène à choisir le modèle 24G pour le reste de l'étude du projet.

3 Classification non-supervisées des signatures des personnes

Une approche intéressante de la base de données MCYT-100 est de se concentrer sur la signature elle-même plutôt que sur l'individu, afin de créer un modèle d'apprentissage qui puisse être généralisé à l'ensemble de la base de données. Cela permet d'explorer les caractéristiques intrinsèques des signatures et d'évaluer leur potentiel pour la classification et l'authentification des signatures manuscrites.

3.1 K-Moyenne avec 24 Gaussiennes

Dans cette partie, nous allons classifier les 2500 signatures, indépendamment de leur auteur, à l'aide d'un clustering basé sur un modèle Kmeans à 3 clusters, en fonction de leur complexité estimée par le modèle GMM avec 24 Gaussiennes.

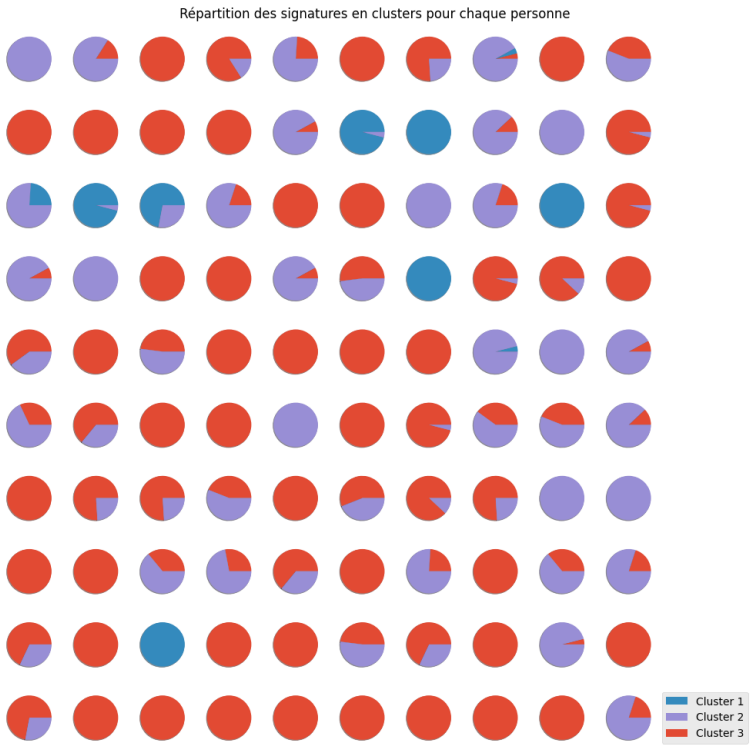

FIGURE 21 – Répartition des signatures en clusters pour chaque personne

Nous pouvons observer la part de variabilité des signatures pour chaque personne, un camembert correspondant à une personne. Visuellement, nous pouvons observer que ce sont les signature à complexité à la limite entre "moyenne" et "élevée" qui subissent le plus variabilité dans la classification, avec presque autant de signatures classées dans le cluster "Complexité moyenne" que dans celui "Complexité élevée" dans les pires des cas.

La classification de la complexité des signatures de la population reste cela-dit sensiblement similaire à celle faite avec avec la moyenne de la complexité des signatures de chaque personne : une majorité de signatures de complexité élevée. La question que l'on

se pose à présent est s'il est possible de classer les individus en s'appuyant sur les mesures de complexité de chacune de leurs signatures et non plus la moyenne. Pour ce faire, nous allons représenter une personne P par un vecteur colonne à 25 lignes (*i.e.* signatures) :

$$P = (S_1 \quad ... \quad S_{25})$$

Nous utilisons un algorithme K-Moyennes à 3 clusters pour classifier nos données. Étant donné que nous ne pouvons pas observer un vecteur à 25 dimensions, il aurait fallu une méthode pour visualiser les résultats. Cependant, puisque chaque dimension correspond à la même donnée (la complexité d'une signature) et que l'ordre des signatures n'a pas d'importance, nous avons opté pour une visualisation telle qu'elle est présentée à la Figure 22).

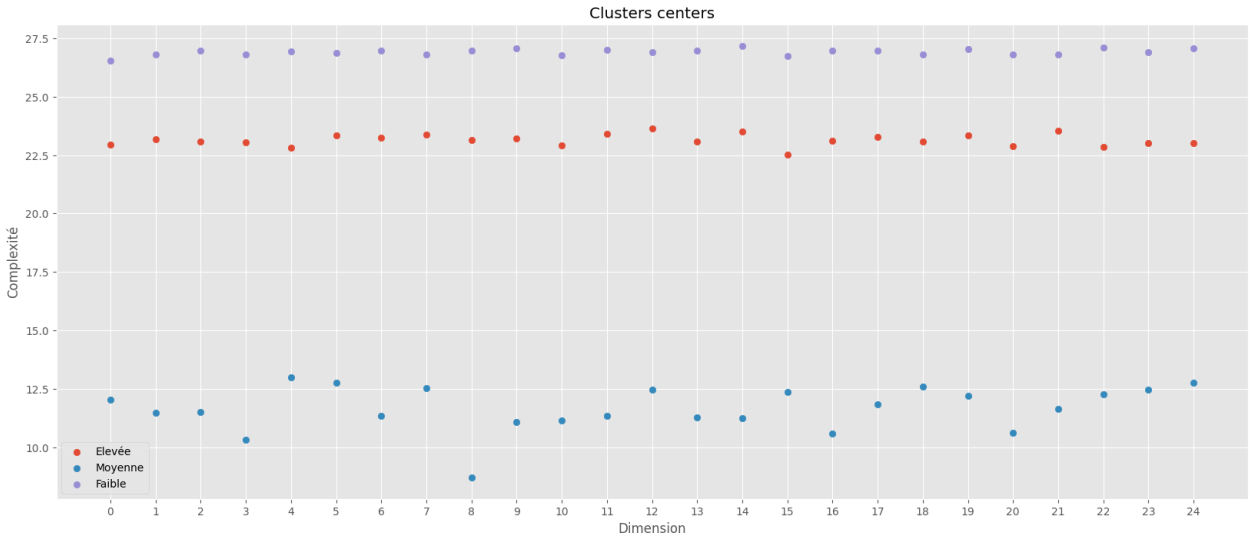

FIGURE 22 – Centres des classes pour une classification des personnes par leurs 25 signatures pour 24 gaussiennes

3.2 Conclusion de la classification non-supervisée des personnes

Le jeu de données MCYT-100 ne contient que peu de signatures simples, ce qui explique la variabilité observée dans les centres de chaque dimension des signature à faible compléxité. Toutefois, nous remarquons que les centres des signatures de complexité moyenne et élevée sont relativement constants. D'un côté, cela suggère que le clustering est efficace car les clusters sont nettement séparés. D'un autre côté, les centres sont sensiblement similaires pour les clusters des signatures de complexité moyenne et élevée en fonction de la dimension. Par conséquent, ce clustering pourrait ne pas fournir beaucoup plus d'informations qu'un clustering basé sur la moyenne de la complexité de chaque personne.

4 Apprentissage et généralisation

L'objectif est à présent de diviser les 2500 signatures en deux groupes égaux de 1250 signatures chacun. Nous avons jugé pertinent de diviser les 2500 signatures en deux sous-groupes de 1250 signatures de manière aléatoire. Cette approche permettrait d'éviter tout biais potentiel dans les résultats en évitant d'obtenir un échantillon particulier qui pourrait être biaisé, tel qu'un échantillon contenant toutes les signatures d'un seul signataire.

Un de ces groupes sera utilisé pour effectuer une classification à l'aide d'un clustering basé sur un modèle K-moyennes à 3 clusters, en fonction de leur complexité estimée par le modèle GMM avec 24 Gaussiennes. Les 1250 signatures restantes seront classées en utilisant la méthode des K plus proches voisins (K-nn ou "K nearest neighbours").

Notons que nous réalisons une première classification avec la méthode des K-moyennes sur les complexités moyennes. On fait donc un premier clustering sur 100 complexités, correspondant aux moyennes des complexités des signatures de chaque signataire.

4.1 Application des méthodes K-moyennes et K-NN sur les deux groupes

La seconde classification se fait en deux parties comme précédemment évoqué. D'abord la classification avec KMeans de 1250 signatures sélectionnées aléatoirement et indépendamment de leur complexité ou du signataire avec la méthode des K-moyennes, puis la classification des 1250 signatures restantes avec la méthode des K-NN et les centres de la première classification KMeans. La méthode des K-NN consiste à calculer la distance à chaque cluster pour chacune des 1250 complexités.

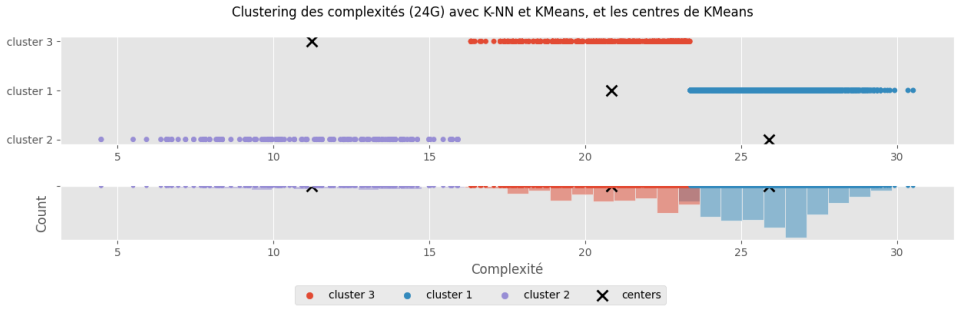

FIGURE 23 – Clustering des complexités K-NN et K-Moyennes

Rappelons que l'algorithme K-NN (*K-Nearest Neighbors*) est un algorithme d'apprentissage supervisé qui peut être utilisé pour la classification ou la régression. L'objectif de l'algorithme est de prédire la classe ou la valeur cible d'un nouvel exemple en se basant sur les K exemples les plus proches de celui-ci dans l'espace des caractéristiques.

Le fonctionnement de l'algorithme K-NN consiste à trouver les K exemples les plus proches du nouvel exemple dans l'espace des caractéristiques, et à prendre une décision en fonction de la classe majoritaire ou de la valeur moyenne de ces K exemples. Son utilisation est particulièrement utile dans les situations suivantes :

— Lorsque les données sont de faible dimension, car la distance entre les exemples peut être calculée facilement.

— Lorsque les données sont assez uniformément réparties dans l'espace, car l'algorithme peut être efficace pour identifier les régions homogènes dans l'espace des caractéristiques.

— Lorsque l'on dispose d'un grand nombre d'exemples dans la base de données.

Nous constatons que les résultats obtenus se rapprochent de ceux obtenus avec les autres méthodes avec un grand nombre de personne avec des signatures à forte complexité.

5 Conclusion

Nous pouvons en conclure que la méthode de généralisation des K-moyennes s'est faite correctement et que les méthodes de clustering implémentées sont valables.

6 Bibliographie

[1] Julian Fierrez, Marcos Faundez-Zanuy, Inmaculada Hernáez, Carlos E. Vivaracho-Pascual : "MCYT baseline corpus: a bimodal biometric database. IEE Proc Vis Image Signal Process Spec Issue Biom Internet", *IEE Proceedings - Vision Image and Signal Processing · December 2003*

[2] Christian Kahindo, Sonia Garcia-Salicetti, Nesma Houmani : "A Signature Complexity Measure to select Reference Signatures for Online Signature Verification", *IEEE*