Classification de battements cardiaques
Data Analysis 2 and classification

Compte Rendu

Bertrand Tom, Lafargue Héloïse

Analyse de données - ENSEEIHT, Toulouse
Année 2023-2024

Contents

1 Introduction

Ce projet vise à développer un système de reconnaissance des battements cardiaques en utilisant des enregistrements audios recueillis de deux manières distinctes : via une application smartphone accessible au grand public et à travers un essai clinique utilisant des stéthoscopes numériques.

À travers différentes techniques de classification, l'étude vise à développer un système de reconnaissance robuste et à en évaluer la performance.

2 Récupération des MFCC des fichiers audios

Pour l'extraction des MFCC (Mel Frequency Cepstral Coefficients) des fichiers audio, nous avons réalisé trois étapes principales avec la bibliothèque librosa :
1. Chargement de l'audio : chaque fichier audio est chargé en mémoire, récupérant le signal et le taux d'échantillonnage.
2. Calcul des MFCC : pour chaque fichier, les MFCC sont calculés à partir du signal audio, produisant un ensemble de coefficients qui résumé les propriétés spectrales de l'audio.
3. Moyennes et étiquetage : les 20 premiers coefficients MFCC moyens sont calculés sur chaque fenêtre de 10ms, pour chaque enregistrement, et associés à un label correspondant à leur catégorie (par exemple, "normal", "murmure"), avant d'être ajoutés à un tableau pour une analyse ultérieure.

Nous obtenons les MFCC et leur label associé présentés dans la figure 1.

	mfcc0	mfcc1	mfcc2	mfcc3	mfcc4	...	mfcc15	mfcc16	mfcc17	mfcc18	mfcc19	Label
0	-401.90738	164.81062	-58.317463	13.229384	10.860508	...	-1.284082	-4.098318	-9.13165	-0.07470109	-4.0142293	artifact
1	54.625393	70.46708	-39.60718	13.036337	-4.0314426	...	-0.23129234	-0.19199274	0.41576678	-1.6389968	-1.7073883	artifact
2	-351.49493	150.23912	67.18877	15.069814	-4.9765363	...	-17.578962	-14.562918	-11.977705	-8.320749	-2.302865	artifact
3	-425.01157	101.98345	6.312329	4.4396434	3.0431933	...	-5.0433974	-2.3123517	-4.572983	-2.42044	-4.426513	artifact
4	-486.64224	93.6632	-25.499802	12.071202	-4.054235	...	-4.3552685	0.7392053	-5.504707	-1.2949178	-5.20937	artifact
...	...	...	...	...	...	...	...	...	...	...	...	...
100	-532.7327	41.8584	19.42001	13.994753	13.489591	...	-0.9217793	-2.749417	-4.079781	-3.5684144	-3.6402516	normal
101	-576.4376	96.39936	28.279396	3.3105412	8.912554	...	-2.961211	-1.5593976	-3.27879	-1.9508821	-3.3353627	normal
102	-459.18704	116.94609	13.236116	27.06791	12.2739525	...	4.556873	2.5409005	4.1454716	3.6600945	3.8791988	normal
103	-596.28906	120.040535	28.13429	28.43375	34.78845	...	-1.2334136	-3.2238798	-2.0361598	-3.103184	-2.78883	normal
104	-387.26135	152.52327	39.222904	1.9184633	8.930495	...	-2.095475	-1.6742053	-2.2538097	-2.5495098	-3.128308	normal

105 rows × 21 columns

Figure 1: MFCC pour le setA

3 Pré-traitement

Pour le pré-traitement nous avons :
- observé la distribution des variables pour savoir si une normalisation serait pertinente (mfcc0 et mfcc1 trop différentes du reste : normaliser),
- effectué une normalisation sur le dataset,
- analysé la répartition des données par classe ("normal" bien représenté, le reste non : ok "normal", possibilité d'écarter "artifact"),
- séparer les données en 75% entraînement et 25% test.

3.1 Observation des données

Le jeu de données contient 20 variables (les 20 premières mfcc) avec 3 classes : "artifact", "murmur", "normal". Il y a 520 observations pour chaque attribut.

Comme nous pouvons le voir sur la figure.2, la répartition des variables est assez distinctes surtout entre les 2 premières MFCC et le reste. Il semble donc utile de normaliser le dataset.

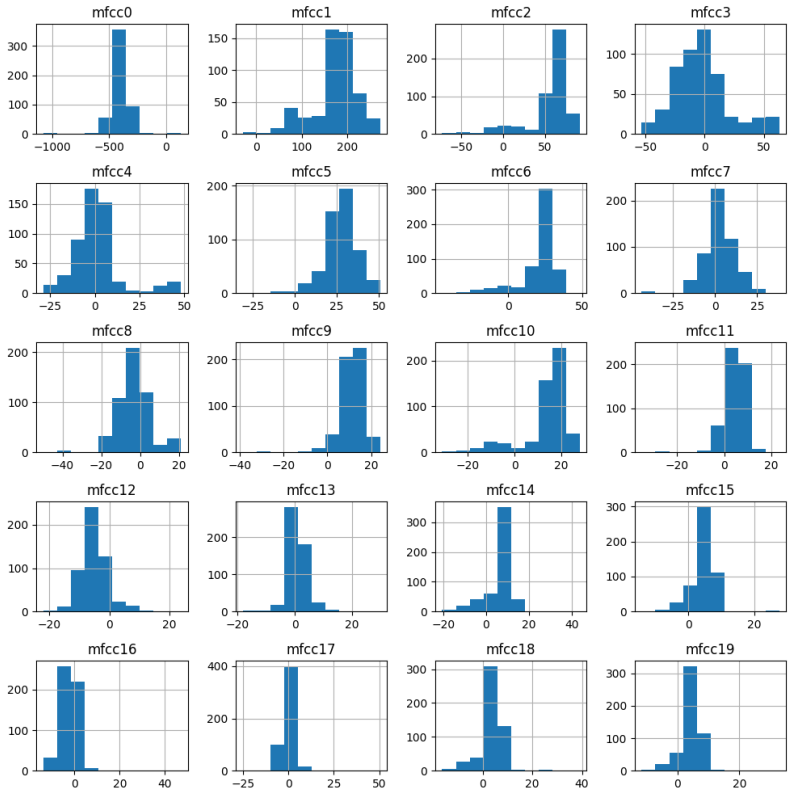

Figure 2: Distribution des données

Cette observation est confortée en regardant les moyennes des variables : il y en a beaucoup autour de valeurs faibles (entre -5 et 5, quelques unes à 10-27) et puis les deux premières moyennes sont bien plus éloignées, à -402 et 171.

3.2 Normalisation

Nous procédons donc à une normalisation du jeu de données (fig.3).

Maintenant toutes les variables prennent des valeurs entre -10 et 10, ce qui va aider à être moins sensibles aux échelles différentes des caractéristiques, améliorant ainsi la performance et la stabilité lors de l'entraînement.

3.3 Analyse des données

Nous nous intéressons à la répartition de nos données selon les différentes classes (figure.4).

Dans notre cas, on dispose de beaucoup d'exemplaires de "normal" (67.5%) et peu de "murmur" (25%) et très peu de "artifact" (7.5%). Les classes ne sont pas équilibrées, l'entraînement fonctionnera donc bien pour la classe "normal" mais potentiellement pas très bien pour les autres qui manquent de représentativité. On pourrait par exemple envisager de ne s'intéresser qu'à "normal" et "murmur" et donc écarter "artifact" de notre recherche (comme pour le setB).

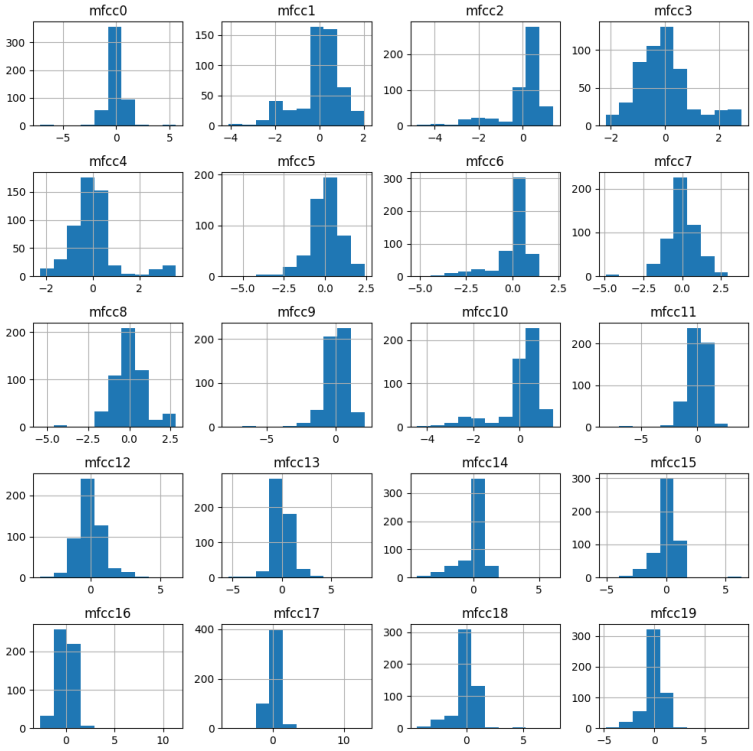

Figure 3: Distribution des données normalisées

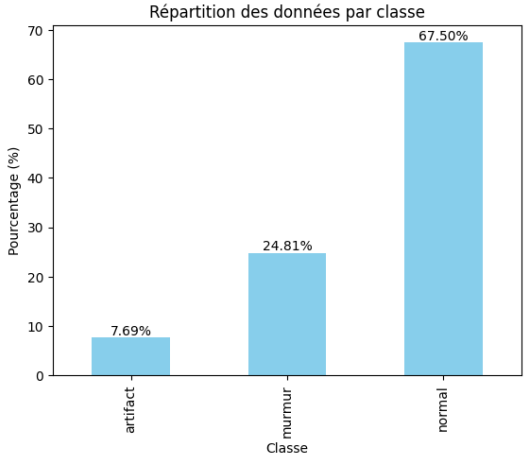

Figure 4: Distribution des données normalisées

3.4 Séparation des données en train-test

On sépare ensuite ce jeu de données normalisées en ensemble d'entraînement (75%) et de test (25%).

Pour information, nous avons aussi testé avec les répartitions suivantes mais nous obtenions les mêmes résultats par la suite : 80-20, 90-10.

4 Méthodes d'apprentissage non supervisé

Dans cette section, nous abordons les méthodes d'apprentissages non-supervisé. Nous en avons étudié 3 ci-dessous : K-means, DBScans et Classification Hiérarchique

4.1 K-Means

Ici, on crée des centroïdes (initialement 3, on souhaite obtenir 3 classes) composés de la moyenne arithmétique des membres du cluster qu'ils représentent. A chaque itération, on ajoute aux clusters les données dans leur voisinage et on met à jour le centroïde, l'opération est répétée jusqu'à convergence.

Les meilleurs résultats obtenus ont été avec l'utilisation de la normalisation : environ 70.5% de précision sans et 74.4% avec. La différence s'est faite de façon notable sur les artefacts, beaucoup mieux classés grâce à la normalisation. Néanmoins, les murmurs sont très souvent confondus avec des battements cardiaques normaux.

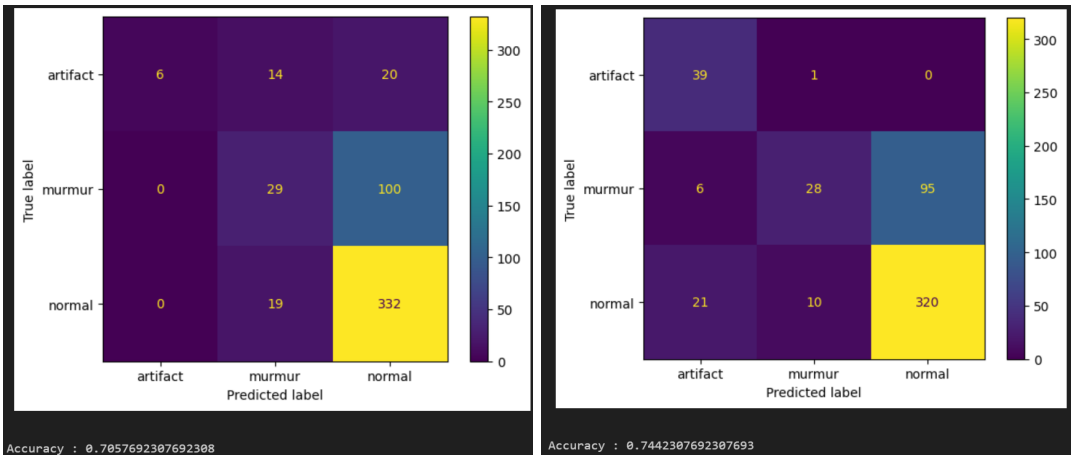

Figure 5: KMeans avec données de base (à gauche) et normalisées (à droite)

4.2 DBScans

Avec DBScans, on regroupe les données dans un rayon ϵ autour d'eux à chaque itération. Le rayon est agrandi à chaque nouvelle données ajoutée en fonction du poids choisi pour les données et ce jusqu'à convergence.

La recherche de paramètres optimaux a dû être faite manuellement car DBScans étant non supervisé il faut adapter les étiquettes à chaque fois, il était plus simple de faire ça à la main. De plus, DBScans renvoie des -1 au lieu des label de classe s'il ne parvient pas à classer la données (si elle est trop bruitée) ce qui a pu poser problème pour les données de bases et a entraîné un petit sur-traitement des données après prédiction (remplacement des -1 par le label artefact qui est probablement leur classe si trop bruitée). Ce traitement n'a pas été nécessaire pour les données normalisées puisqu'elles étaient toutes classables par DBScans.

Nous avons utilisé cette méthode sur le jeu de données de base puis sur le jeu de données normalisées ce qui nous a permis de passer de 72.8% de précision à 79.2%. On peut voir par exemple ici que la normalisation a permis de beaucoup mieux reconnaître la classe "artifact" en passant de 12 à 36 vrais positifs sur 40 exemplaires d'artifact. Cependant, nous ne sommes toujours pas capables de bien distinguer les murmurs qui sont principalement considérés par notre modèle comme des battements cardiaques normaux (fig.9).

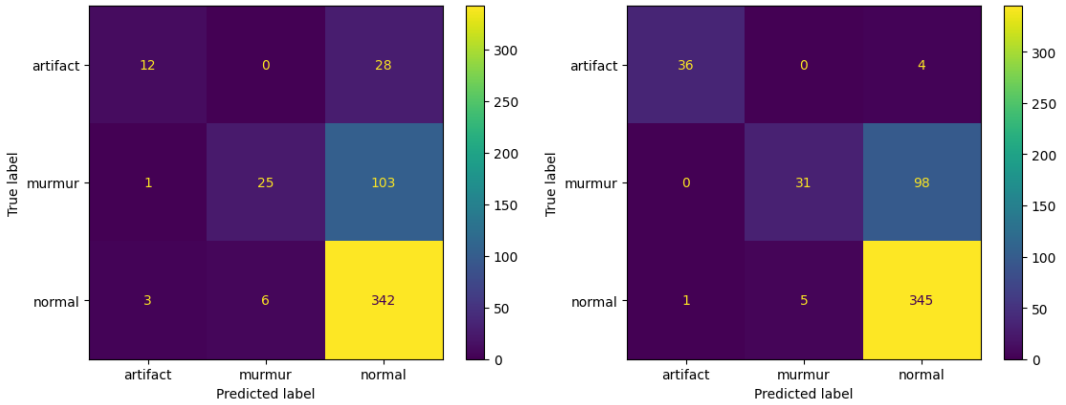

Figure 6: DBScans avec données de base (à gauche) et normalisées (à droite)

4.3 Classification Hiérarchique

La classification hiérarchique est une technique de clusterisation dans laquelle on crée de multiples clusters et dont le nombre qu'on obtient finalement dépend de la dissimilarité choisie. Plus la dissimilarité acceptable augmente dans notre modèle, moins on a de classes. Ici on choisi la mesure de dissimilarité correspondant à la liaison de Ward, qui minimise la variance au sein des clusters formés, et une distance de coupure à 51 qui correspond à la création de 3 clusters (plus haut on en aurait 2 puis 1), conduisant à une précision de classification de 74.23%.

Pour que cela fonctionne, il a fallut changer certaines étiquettes par une autre car cet algorithme ne renvoie pas de "0" ce qui pose problème avec correspondance(), il renvoie à partir de 1 les étiquettes. Comme on a choisi une distance avec 3 classes, on obtenait des 1,2 et 3. Il a suffit de remplacer les 3 par des 0 pour que les étiquettes soient traitables par correspondance().

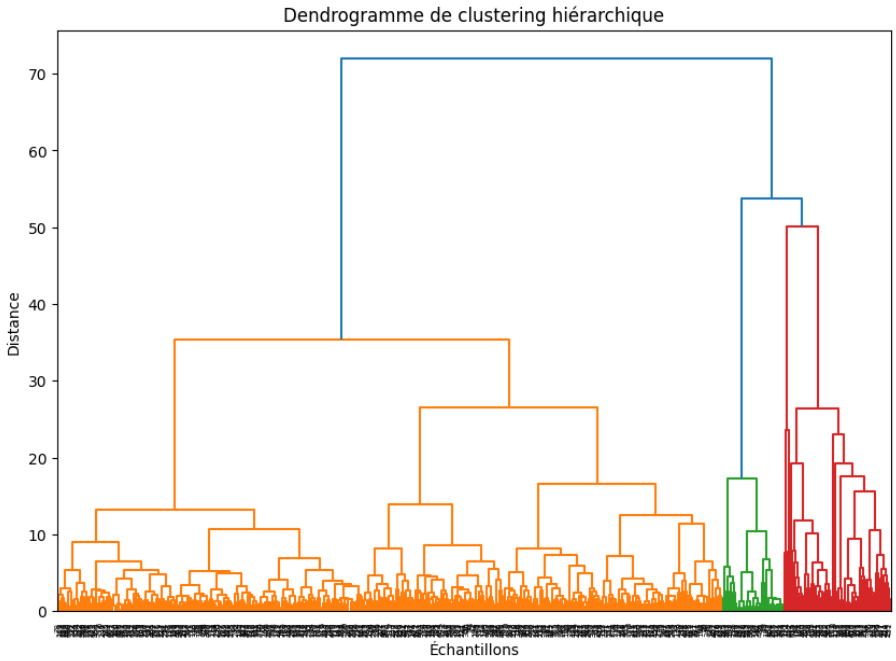

Figure 7: Dendrogramme de clustering hiérarchique

Les résultats obtenus (fig.8) montrent que cette méthode fonctionne relativement bien pour reconnaître les "artifact" et les "normal" mais confond beaucoup les "murmur" en "normal" ce qui est très inquiétant dans le cadre de notre sujet.

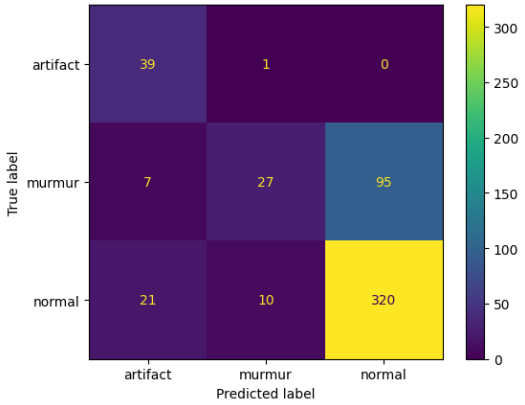

Figure 8: Matrice confusion obtenue avec la classification hiérarchique

5 Méthodes d'apprentissage supervisé

5.1 Forêts aléatoires

Pour ce dernier essai, nous avons décidé d'essayer d'utiliser une classification par forêt aléatoire avec le jeu de données normalisées. La classification par forêt aléatoire étant une méthode efficace, les résultats étaient déjà plutôt bon dans l'étude préliminaire sans normalisation avec 81.5% de précision et d'assez bons résultats sur la séparation des classes. Avec la normalisation des données, on obtient 85.5% de précision et la classification est encore meilleure : plus aucune erreur sur les artefacts et moins d'erreurs en particulier sur les murmures.

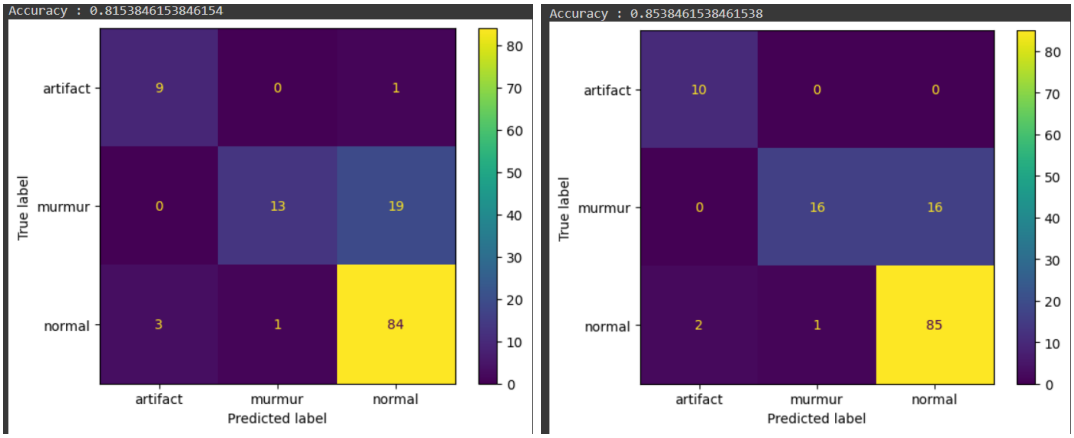

Figure 9: Forêt aléatoire avec données de base (à gauche) et normalisées (à droite)

6 Conclusion

Dans une partie préliminaire où nous avons testé diverses méthodes de classification supervisée sur des ensembles de données non pré-traitées ou avec ACP seule, nous avons constaté qu'il était difficile pour les différentes méthodes de séparer les données correspondant à un murmure des données normales. La méthode par k-ppv avait un bon taux de réussite pour reconnaître les murmures mais prenait aussi certains normaux pour des murmures. En non supervisé, DBScans et la Classification Hiérarchique ont donné de plutôt bon résultats une fois la normalisation ajoutée. Les meilleures résultats ayant été obtenus avec Forêt Aléatoire (supervisé) sur des données normalisées, mais là encore on a seulement 50 % de réussite pour la détection des murmures même si on a presque 100% de réussite sur les autres classes nous donnant environ 85.4% de précision globale.