Université Toulouse III - Paul Sabatier
UT3

Calcul scientifique et apprentissage automatique

Projet Battements cardiaques

Auteurs

EL AMRANI Wadie
RATABOUIL Guilhem

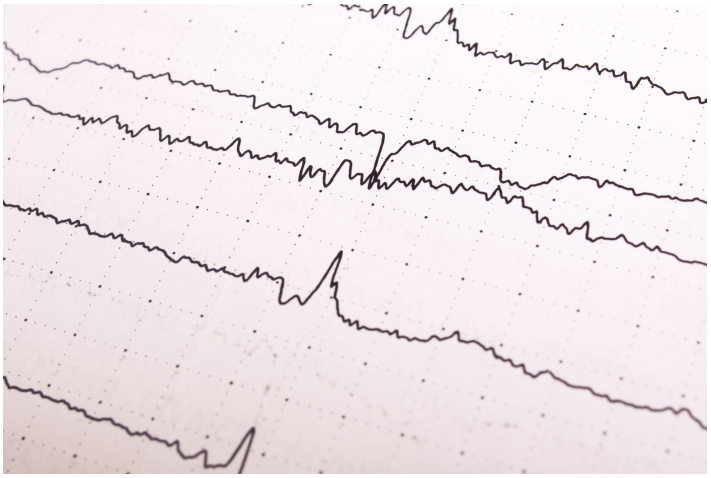
Professeurs

MOUYSSET SANDRINE

Toulouse, 17 décembre 2023

Conteúdo

1 Introduction



L'intégration de l'intelligence artificielle (IA) dans le domaine médical a ouvert de nouvelles avenues prometteuses, notamment dans le diagnostic et le suivi des pathologies cardiaques. La capacité de l'IA à analyser et interpréter les sons cardiaques représente un progrès significatif, permettant une détection plus précoce et précise de diverses anomalies cardiaques. Dans ce cadre, notre projet se concentre sur l'analyse de trois types distincts de battements cardiaques : normaux (351 échantillons), murmures (129 échantillons) et artefacts (40 échantillons). Ces catégories reflètent les variétés courantes de sons cardiaques, chacune portant des informations diagnostiques cruciales.

La première étape de notre démarche a consisté à transformer les enregistrements audio de ces battements en matrices de paramètres exploitables, en utilisant les Coefficients Céphalométriques de Fréquence de Mel (MFCC) grâce à la bibliothèque Python librosa. Cette technique permet de capturer les caractéristiques fréquentielles essentielles des sons cardiaques, transformant les enregistrements bruts en données structurées propices à l'analyse algorithmique.

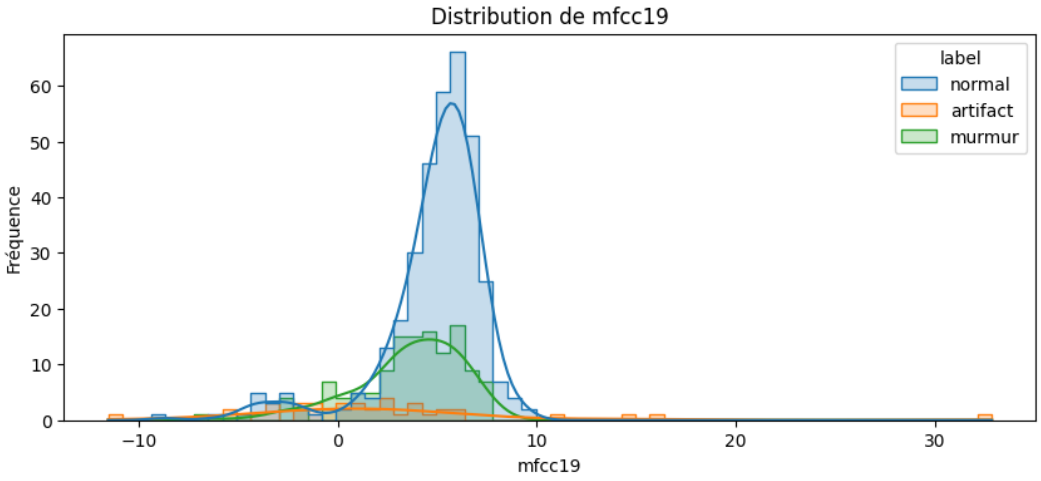
Ce projet exploite des techniques d'apprentissage machine à la fois supervisées et non supervisées pour classifier ces sons cardiaques. L'apprentissage supervisé sera utilisé pour entraîner des modèles sur des données étiquetées, permettant la reconnaissance de motifs spécifiques à chaque catégorie de battement. Parallèlement, l'apprentissage non supervisé nous aidera à découvrir des structures cachées et des relations au sein des données non étiquetées. L'objectif est de développer un système capable de classifier avec précision les différents types de battements cardiaques, offrant ainsi un outil puissant pour le diagnostic médical et la surveillance des patients.

2 Analyse Exploratoire des Données (EDA)

Avant de plonger dans la modélisation, il est utile de réaliser une analyse exploratoire des données. Cela pourrait inclure la visualisation des distributions des différentes classes (normal, murmur, artifact), l'analyse des distributions des coefficients MFCC, et la recherche de tout pattern ou anomalie.

2.1 Histogrammes des Coefficients MFCC

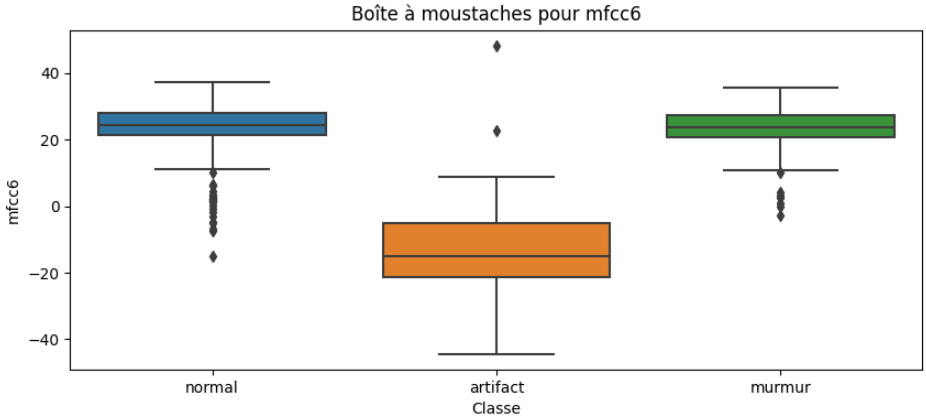
Crée des histogrammes pour chaque coefficient MFCC, en distinguant les différentes classes. Cela aide à visualiser la distribution des données pour chaque caractéristique. l y a un chevauchement considérable entre les distributions de mfcc19 pour les classes



'normal' et 'murmur', suggérant que ce coefficient à lui seul peut ne pas être suffisant pour distinguer de manière fiable ces deux classes.

2.2 Boîtes à Moustaches

Affiche des boîtes à moustaches pour chaque coefficient MFCC par classe, offrant une vue sur la dispersion et les valeurs aberrantes potentielles, par exemple : **Médiane**



: La ligne au milieu de chaque boîte représente la valeur médiane de mfcc6 pour chaque classe. On observe que la médiane pour les classes 'normal' et 'murmur' est à peu près au même niveau, tandis que la médiane pour la classe 'artifact' est nettement plus élevée, ce qui pourrait indiquer une différence dans les caractéristiques de fréquence

des sons d'artefact par rapport aux sons normaux et de murmure.
Valeurs Extrêmes et Valeurs Aberrantes :* Les "moustaches"s'étendent à partir des quartiles jusqu'aux valeurs les plus éloignées qui sont encore considérées comme non aberrantes (généralement 1,5 fois l'IQR au-dessus du troisième quartile et en dessous du premier quartile). Les points qui tombent en dehors des moustaches sont des valeurs aberrantes, indiquant des observations qui s'écartent significativement du reste des données. Toutes les classes présentent des valeurs aberrantes, suggérant la présence de battements cardiaques atypiques ou d'éventuelles erreurs dans les données.

3 Méthodes Supervisées

Dans cette sous-section, nous décrirons en détail les méthodes supervisées que nous avons choisie pour notre projet, y compris le prétraitement des données, les caractéristiques extraites, et le modèle utilisé. Nous présenterons également les résultats, y compris la matrice de confusion et le score de performance.

3.1 Random forest

Théorie

Le Random Forest est une méthode d'apprentissage supervisé qui appartient à la famille des méthodes ensemblistes. Elle opère en construisant une multitude d'arbres de décision lors de l'entraînement et en produisant le mode des classes (classification) ou la moyenne des prédictions (régression) des arbres individuels pour prédire l'issue.
Explication des paramètres :

Algorithm 1 Algorithme Random Forest

Entrée:
$\mathbf{X}$: ensemble de données d'entraînement
$\mathbf{y}$: étiquettes de l'ensemble de données d'entraînement
N : nombre d'arbres dans la forêt
M : nombre de caractéristiques à considérer pour chaque division
D : profondeur maximale de chaque arbre
Initialisation: Créer une forêt vide
for $i = 1$ to N **do**
    Générer un échantillon bootstrap $\mathbf{X}_i, \mathbf{y}_i$ à partir de $\mathbf{X}, \mathbf{y}$
    Créer un arbre de décision A_i à partir de $\mathbf{X}_i, \mathbf{y}_i$
    while non maximal D ou jusqu'à ce que toutes les feuilles soient pures **do**
        Sélectionner M caractéristiques au hasard pour la division
        Diviser le nœud pour maximiser l'information gain (ou une autre mesure de qualité)
    Ajouter l'arbre A_i à la forêt
Sortie: La forêt d'arbres $\{A_1, A_2, \ldots, A_N\}$

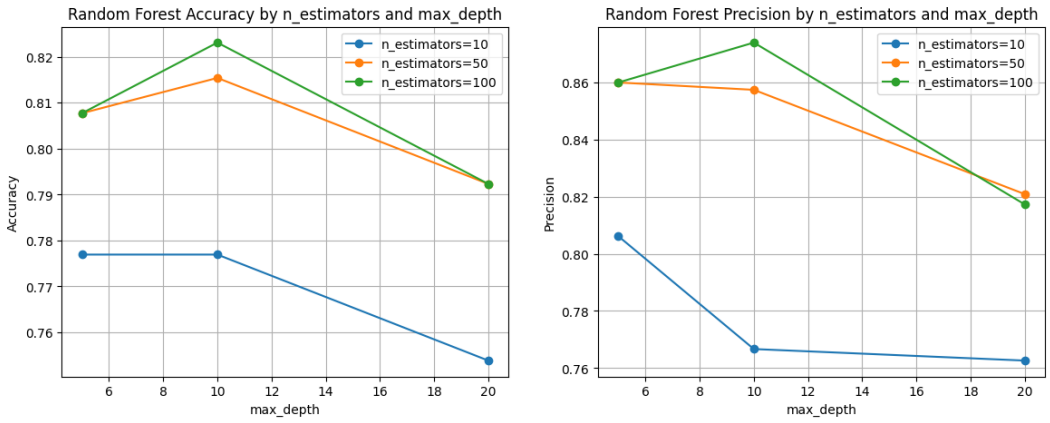
- $\mathbf{X}$: L'ensemble des observations de votre jeu de données d'entraînement. Chaque observation est un vecteur de caractéristiques.
- $\mathbf{y}$: Les étiquettes correspondant à chaque observation dans votre ensemble d'entraînement.

- N : Le nombre d'arbres que vous souhaitez dans votre forêt. Un nombre plus élevé d'arbres peut augmenter la précision et la robustesse de votre modèle, mais également le temps de calcul.
- M : Le nombre de caractéristiques à considérer lors de la recherche de la meilleure scission. Ce paramètre est crucial pour la diversité de votre forêt. Une valeur faible augmente la diversité mais peut réduire la précision.
- D : La profondeur maximale de chaque arbre. Une profondeur plus grande permettra à l'arbre de capturer plus de détails, mais augmente le risque de surapprentissage.

La fonction evaluate random forest a été conçue pour analyser et visualiser l'impact de deux hyperparamètres clés du classificateur Random Forest : n estimators, qui est le nombre d'arbres dans la forêt, et max depth, qui est la profondeur maximale de chaque arbre. En variant ces hyperparamètres, la fonction entraîne plusieurs modèles sur un ensemble d'entraînement (X train, y train) et évalue leur performance sur un ensemble de test (X test, y test) en termes d'exactitude (accuracy) et de précision (precision).

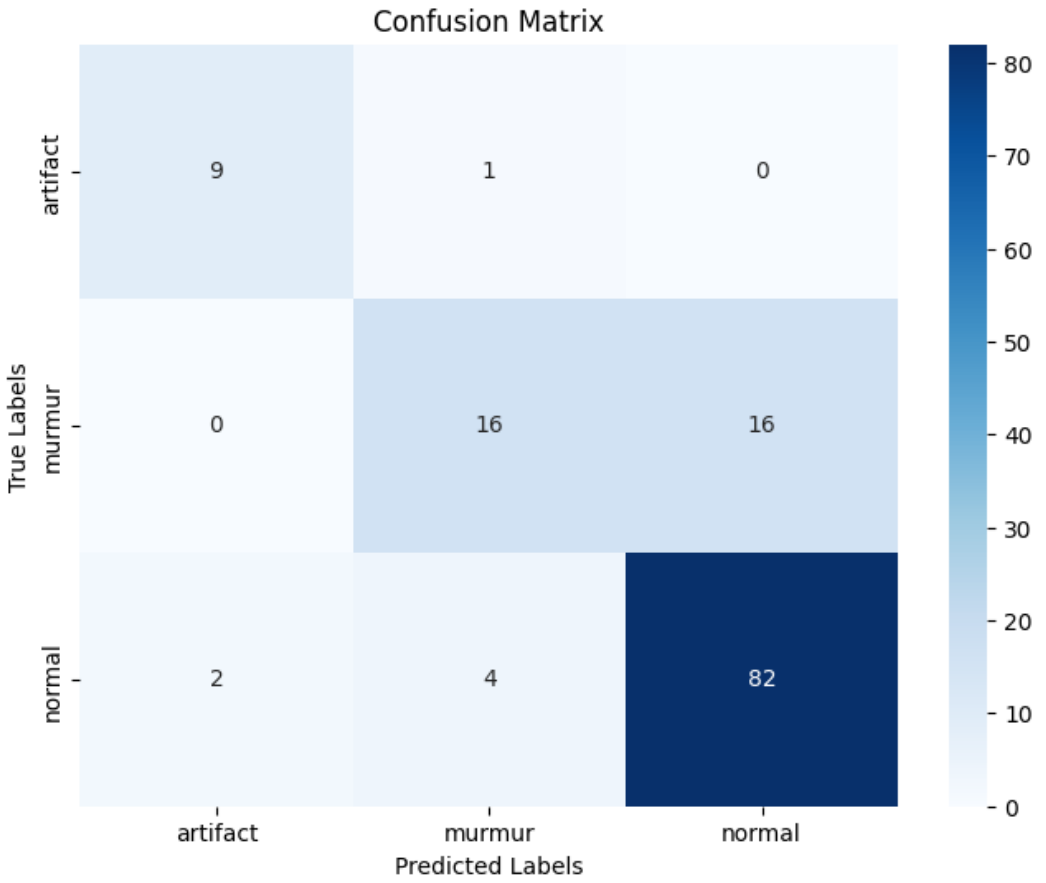
À travers une série d'itérations, la fonction ajuste la composition de la forêt aléatoire en modifiant le nombre d'arbres et leur profondeur maximale. Après l'entraînement, elle évalue chaque modèle et stocke les résultats de performance. Finalement, elle affiche deux graphiques : l'un montrant comment l'exactitude varie avec les hyperparamètres, et l'autre montrant l'évolution de la précision.

Jeu de données complet `DataMFCC`

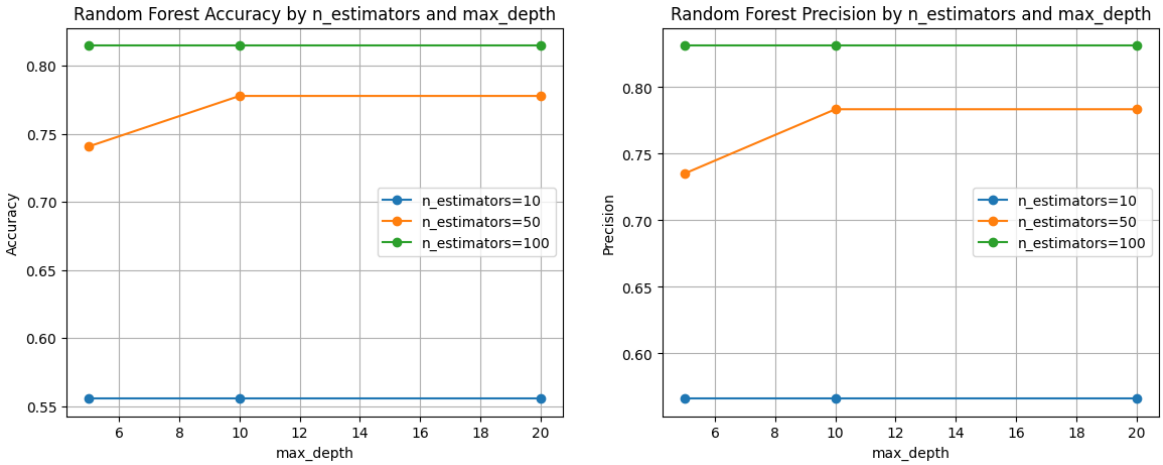


Interprétation : D'après ces observations, les meilleurs paramètres pour le modèle semblent être un nombre d'arbres (n estimators) de 50 avec une profondeur maximale (max depth) d'environ 10. Ces valeurs offrent un bon compromis entre l'exactitude et la précision et évitent les problèmes de surajustement observés avec des arbres plus profonds, surtout lorsqu'il y a moins d'arbres dans la forêt. Pour le modèle avec 100 arbres, bien que l'exactitude soit légèrement meilleure à des profondeurs plus faibles, la précision commence à chuter plus tôt, indiquant que l'augmentation du nombre d'arbres ne compense pas nécessairement pour l'augmentation de la complexité du modèle.

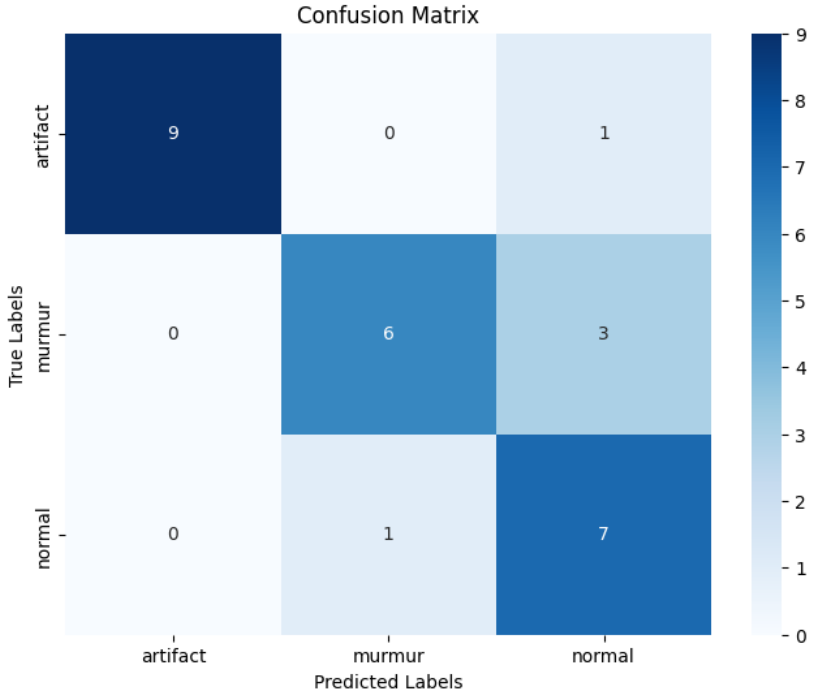
Avec estimators=100 et max depth=None :
Accuracy: 0.82
Precision: 0.80



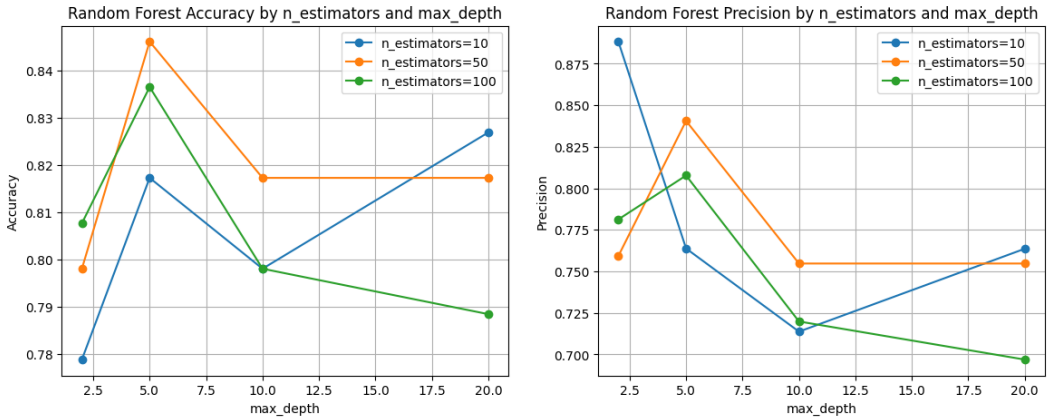
Jeu de données `DataMFCC_setA`



Ces observations indiquent que l'augmentation du nombre d'arbres et de la profondeur maximale ne se traduit pas nécessairement par une amélioration linéaire des performances et que des valeurs moyennes pour ces hyperparamètres pourraient être plus appropriées pour éviter les problèmes de surajustement tout en conservant une bonne capacité de généralisation. Pour nestimators=100 et maxdepth=10 on a :
Accuracy: 0.81
Precision: 0.83



Jeu de données `DataMFCC_setB`

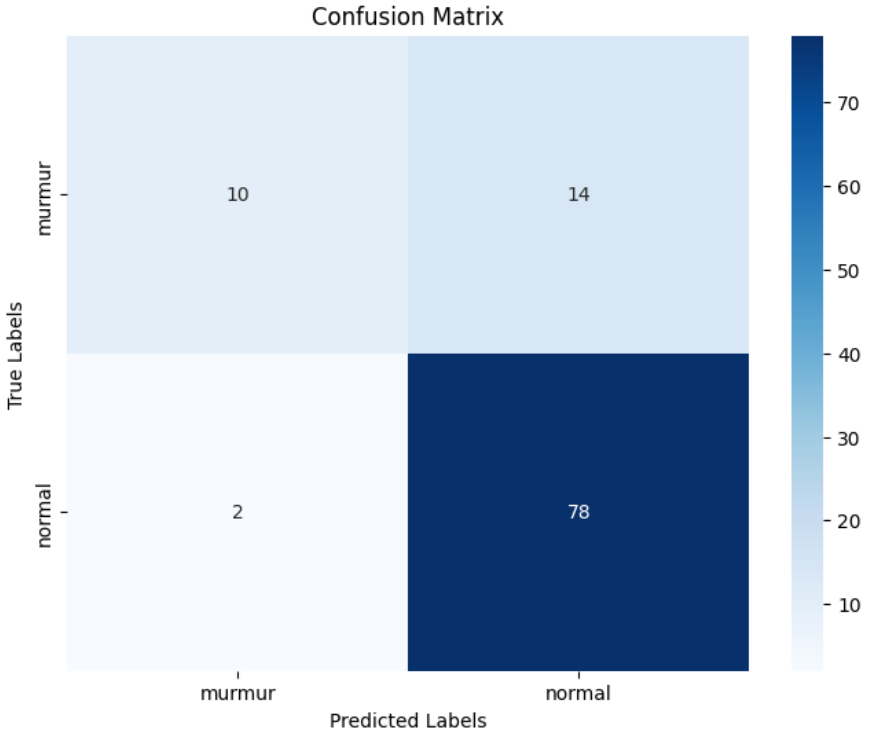


Les meilleures performances pour les deux métriques semblent être atteintes avec nestimators=50 et une maxdepth d'environ 5. Cette configuration offre un équilibre entre éviter le surajustement (comme on peut le voir avec la baisse de performance pour les profondeurs supérieures) et assurer une capacité de généralisation suffisante (contrairement aux profondeurs inférieures où la performance est moindre).

Il est important de noter que l'exactitude et la précision maximales ne coïncident pas toujours, et le choix des hyperparamètres doit tenir compte des objectifs spécifiques du modèle. De plus, la validation croisée doit être utilisée pour confirmer ces résultats sur différents sous-ensembles de données afin de garantir que les performances ne sont pas spécifiques à un seul ensemble de test.

Accuracy: 0.84

Precision: 0.84



3.2 SVM

Théorie

Les machines à vecteurs de support (SVM) sont particulièrement efficaces dans des espaces de grande dimension et peuvent trouver des frontières de décision complexes, même lorsque les données ne sont pas linéairement séparables. Leur capacité à utiliser des fonctions de noyau pour transformer l'espace des caractéristiques en un espace où une séparation linéaire est possible les rend appropriées pour les scénarios suivants révélés par notre EDA.

Évaluation

nous avons entrepris une analyse approfondie des hyperparamètres de notre modèle SVM pour déterminer la configuration optimale permettant d'obtenir la meilleure performance de classification. Pour ce faire, nous avons utilisé une méthode systématique pour explorer l'espace des hyperparamètres, en particulier en ce qui concerne les paramètres C et gamma, qui sont cruciaux dans la configuration du modèle SVM avec un noyau RBF.

La fonction svm paraman alysis a été conçue pour entraîner des modèles SVM avec une gamme de valeurs pour un hyperparamètre spécifié, soit C pour la pénalité de classification incorrecte, soit gamma qui détermine l'influence d'un seul échantillon d'entraînement. Cette fonction itère sur une liste de valeurs prédéfinies pour l'hyperparamètre choisi, entraîne le modèle SVM avec ces valeurs, effectue des prédictions sur l'ensemble de test et calcule la précision et l'exactitude pour chaque configuration. Ces métriques sont ensuite tracées pour fournir une visualisation claire de la manière dont la variation de l'hyperparamètre affecte la performance du modèle.

Pour évaluer l'influence conjointe des hyperparamètres C et gamma, nous avons employé une approche complémentaire qui permet d'analyser les performances du modèle en faisant varier simultanément les deux paramètres. En utilisant une série de graphiques indépendants pour chaque valeur de gamma, nous avons pu observer comment la

Algorithm 2 Algorithme SVM

Entrée:
$\mathbf{X}$: ensemble de données d'entraînement
$\mathbf{y}$: étiquettes de l'ensemble de données d'entraînement, $\mathbf{y}_i \in \{-1, 1\}$
C : paramètre de régularisation
Initialisation: Choisir un hyperplan initial (ou aléatoirement)
repeat
    Sélectionner un sous-ensemble d'exemples d'entraînement, appelé vecteurs de support
    Minimiser la fonction objectif:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

    sous les contraintes:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

    Mettre à jour l'hyperplan de séparation.
until convergence ou nombre d'itérations maximum atteint
Sortie: Vecteur des poids $\mathbf{w}$ et biais b qui définissent l'hyperplan optimal.

précision et l'exactitude varient avec les différentes valeurs de C. Ce processus nous a permis d'identifier les paires de valeurs C et gamma qui maximisent à la fois la précision et l'exactitude, nous guidant ainsi vers la sélection des meilleurs hyperparamètres pour notre modèle SVM.

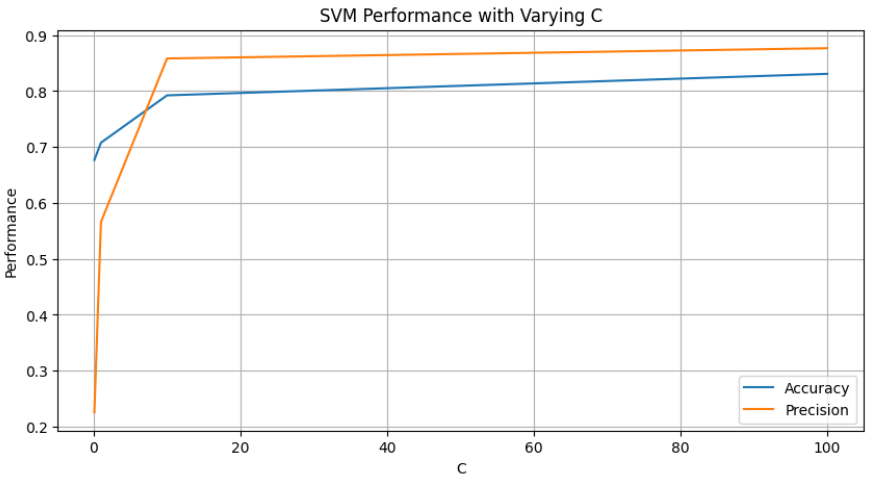
L'application de ces méthodologies a abouti à une compréhension plus nuancée de la relation entre les hyperparamètres du SVM et les performances du modèle, facilitant une optimisation efficace et éclairée pour notre projet de classification.

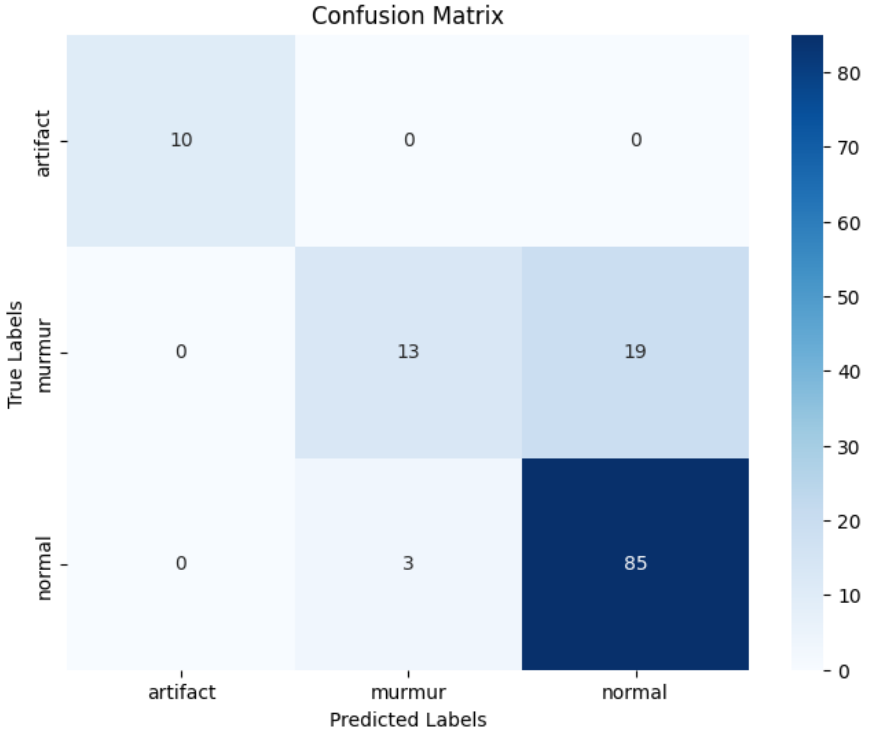
Jeu de données complet `DataMFCC`

Le paramètre gamma détermine l'influence qu'un exemple individuel de l'ensemble d'entraînement a sur la formation de la frontière de décision. Lorsque gamma est défini sur 'scale', cela signifie que la valeur de gamma est calculée automatiquement en fonction du nombre de caractéristiques (features) de l'ensemble de données d'entraînement.

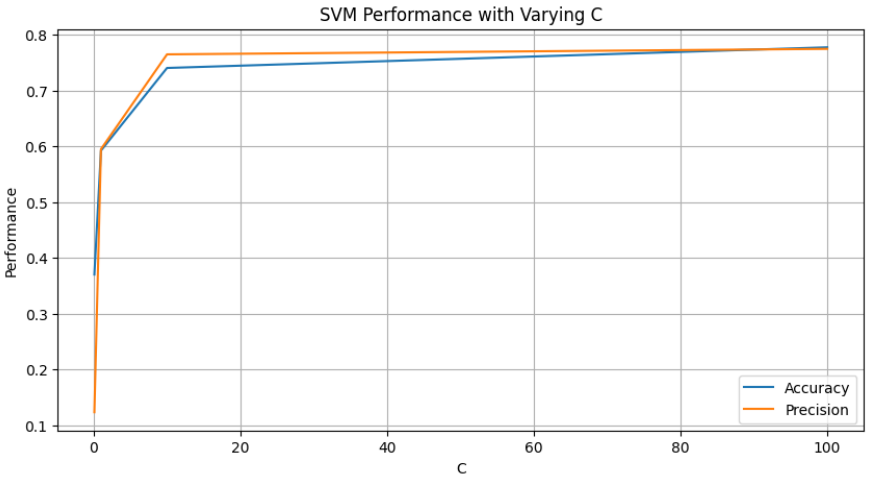
Le choix des paramètres C=100 et gamma='scale' nous donne comme résultat :

Le modèle excelle à identifier correctement les cas 'normal' et 'artifact', mais a des difficultés à distinguer entre 'murmur' et 'normal'. Des améliorations du modèle pourraient être nécessaires pour mieux classifier la classe 'murmur'.

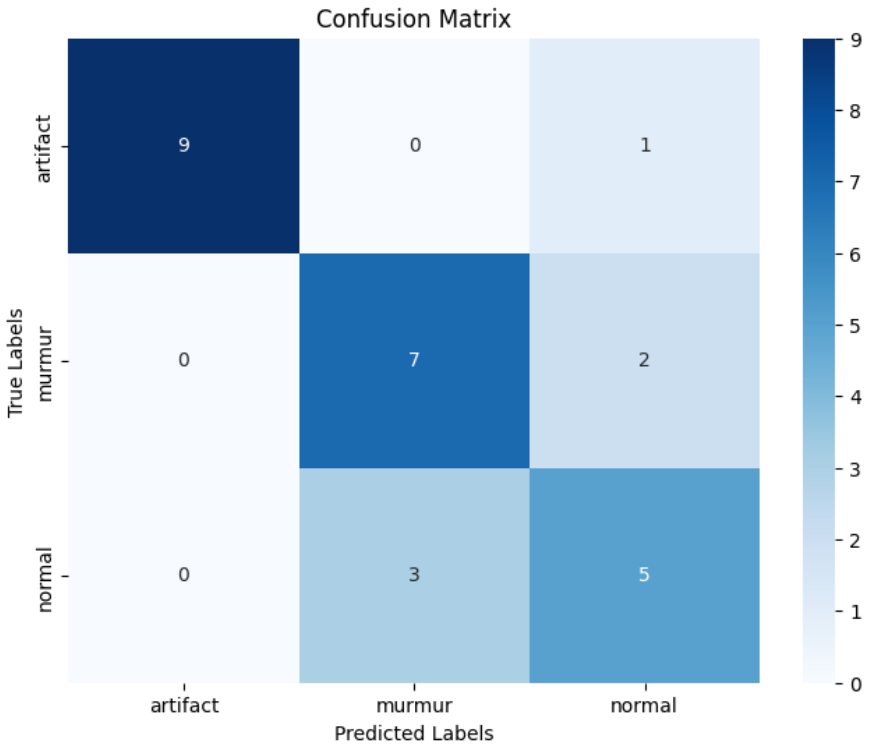




Jeu de données `DataMFCC_setA`

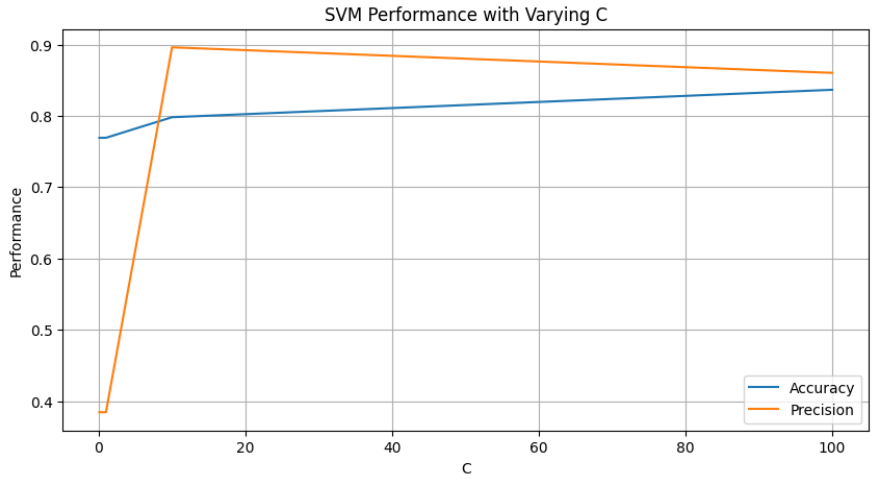


Le choix des paramètres C=100 et gamma='scale' nous donne comme résultat :
Accuracy: 0.78
Cette matrice de confusion révèle que le modèle est très fiable pour la détection des

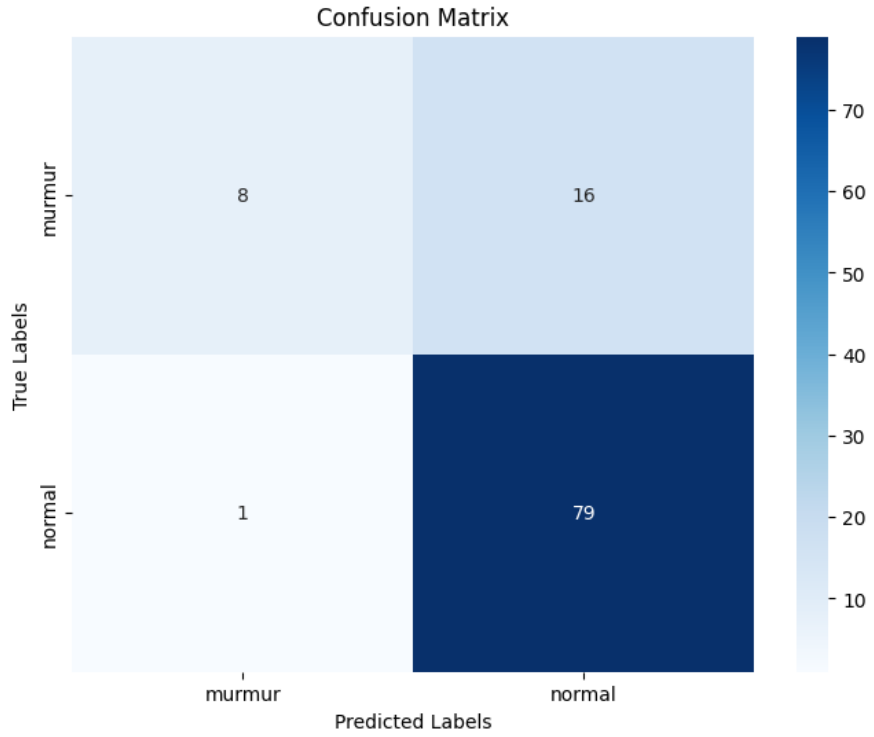


'artifacts', assez fiable pour la détection des 'murmurs', mais moins précis pour la classification des instances 'normal', montrant une tendance à les confondre avec des 'murmurs'.

Jeu de données `DataMFCC_setB`



Le choix des paramètres C=100 et gamma='scale' nous donne comme résultat : **Accuracy**: 0.84
le modèle semble être assez performant pour identifier les cas "normaux", mais il a



du mal à identifier correctement les cas de "murmur", comme le montre le nombre élevé de faux négatifs. Cela pourrait indiquer que le modèle est biaisé en faveur de la classe "normal"ou qu'il manque de caractéristiques discriminantes pour identifier correctement la classe "murmur"

3.3 k-NN

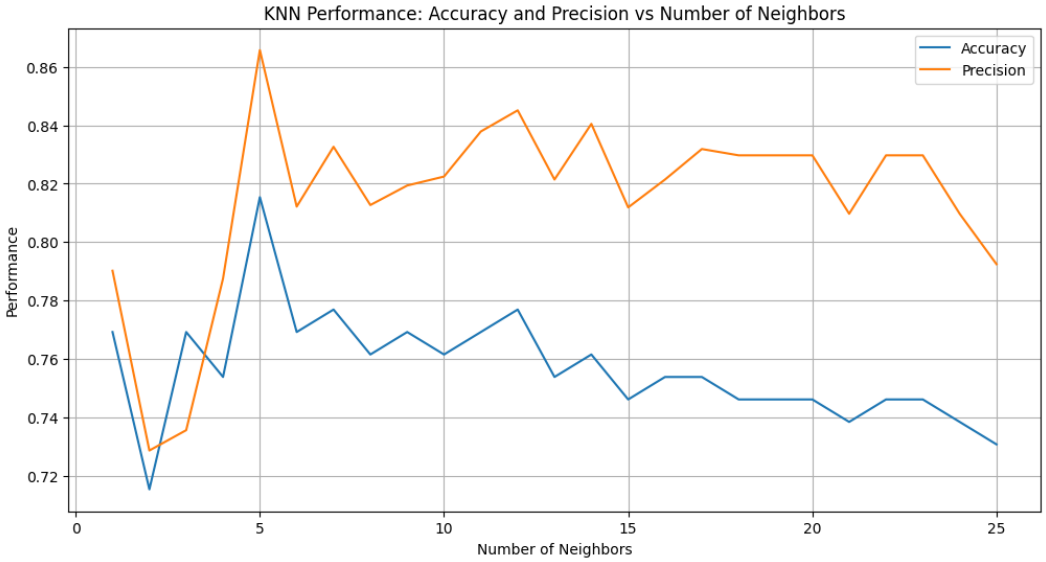
L'algorithme des k plus proches voisins (k-NN) est une méthode d'apprentissage supervisé qui peut être utilisée pour la classification et la régression. Théoriquement, pour un point de données donné, l'algorithme identifie les k points de l'ensemble

d'entraînement qui sont les plus proches de ce point, selon une mesure de distance comme la distance euclidienne. La classe ou la valeur cible du point de données est alors déterminée par un vote majoritaire des classes des k voisins pour la classification, ou par une moyenne pondérée pour la régression.

Changement de paramètres

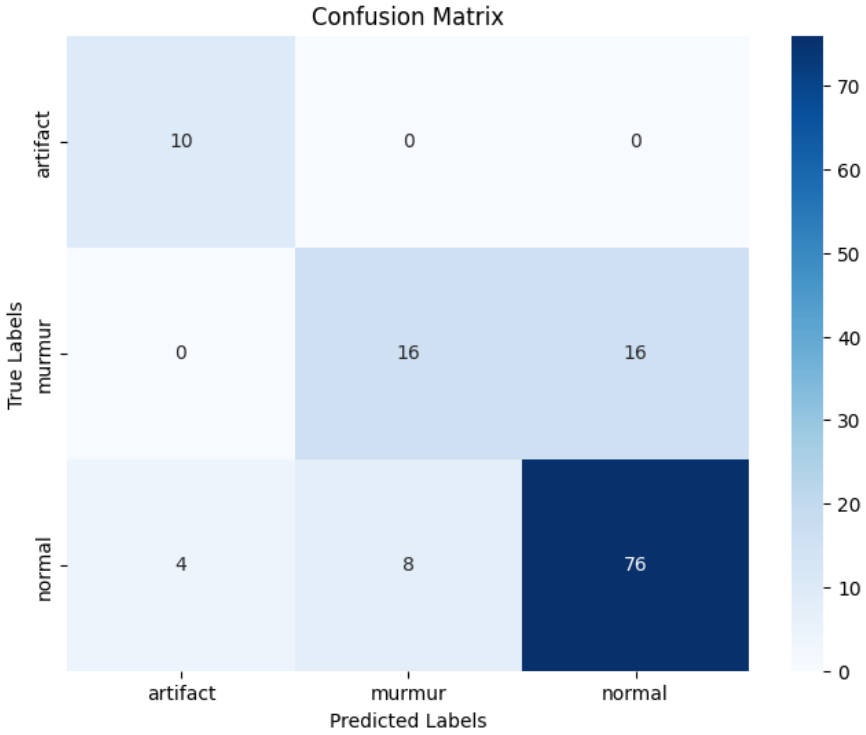
Cette fonction plot knn performance est conçue pour évaluer et visualiser les performances du classificateur des k plus proches voisins (k-NN) en fonction du nombre de voisins utilisé dans le modèle. La fonction prend en entrée des ensembles de données d'apprentissage et de test (caractéristiques et étiquettes) ainsi qu'un nombre maximal de voisins à tester. Pour chaque valeur de k dans la plage de 1 à max neighbors, la fonction entraîne un modèle k-NN, réalise des prédictions sur l'ensemble de test, puis calcule l'exactitude (accuracy) et la précision (precision) de ces prédictions.

Jeu de données complet `DataMFCC`

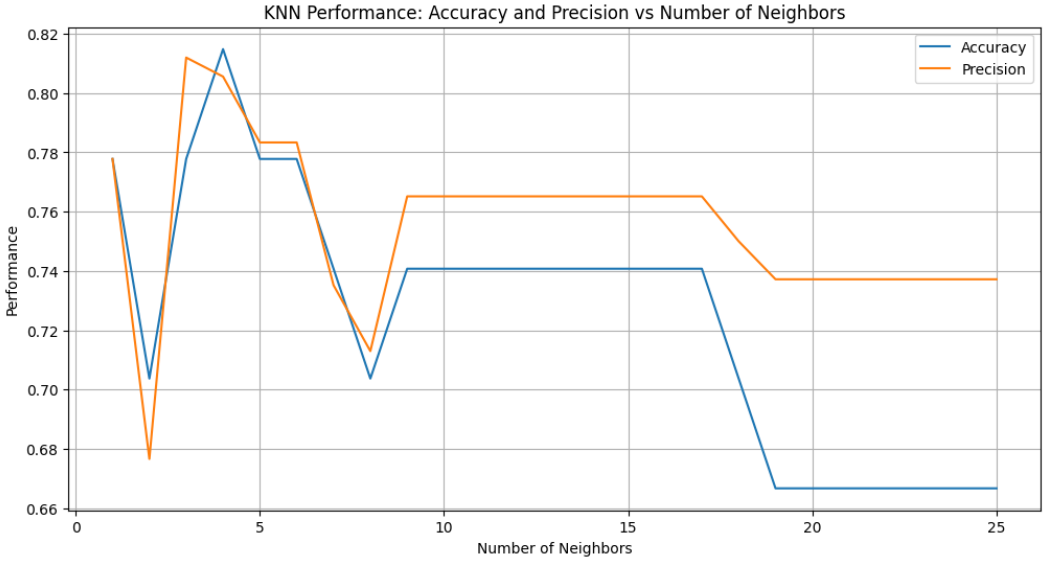


Interprétation : L'exactitude présente une tendance à la baisse en général à mesure que le nombre de voisins augmente, ce qui est typique car une valeur de k plus élevée peut lisser les frontières de décision et conduire à un sous-ajustement.

Accuracy: 0.78

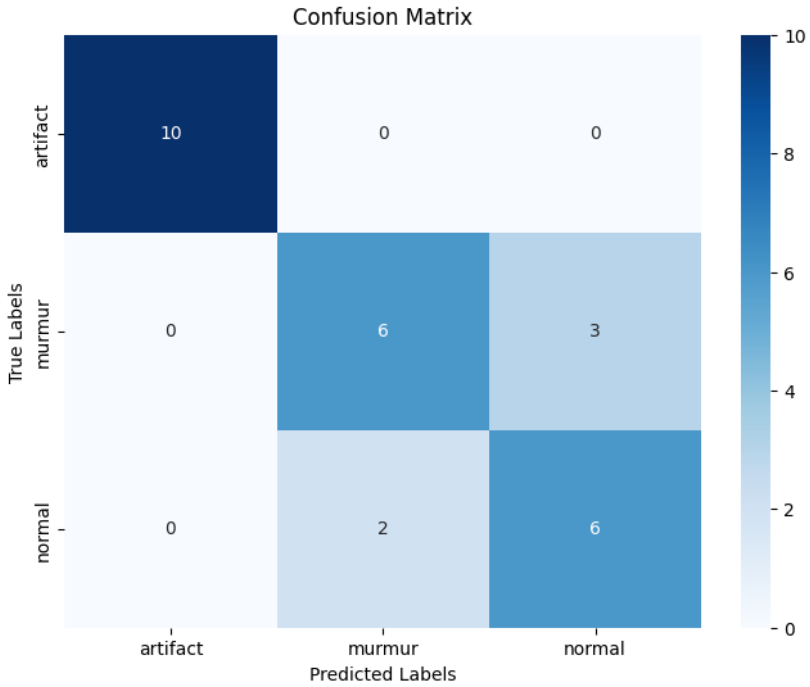


Jeu de données `DataMFCC_setA`

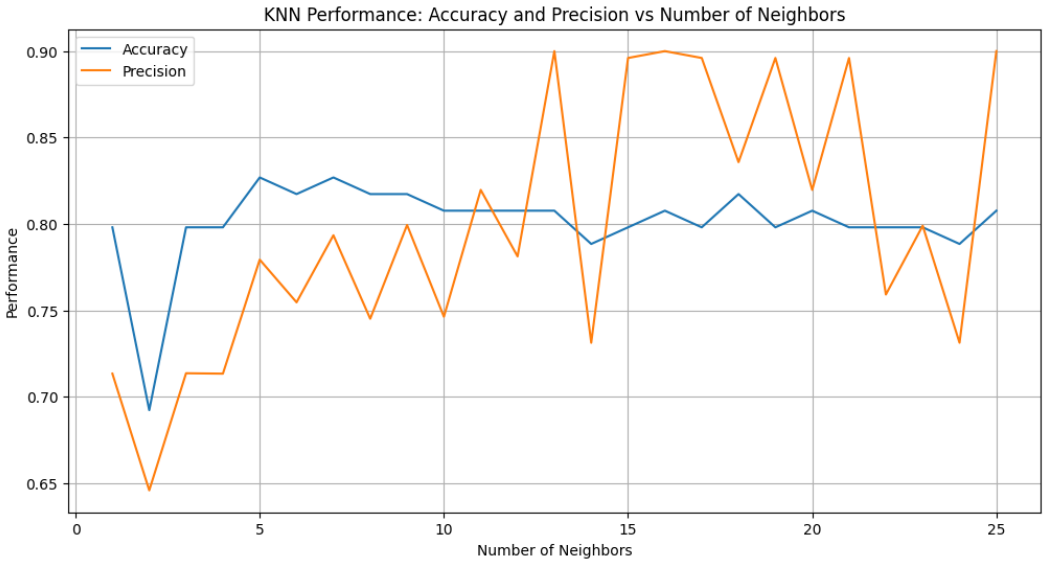


En général, le modèle semble présenter de meilleures performances avec un nombre de voisins inférieur. Un nombre optimal de voisins pour ce modèle semble être dans la gamme de 5 à 10, où la précision et l'exactitude semblent être relativement élevées et stables. Au-delà de ce point, en particulier après 20 voisins, les performances du modèle diminuent, ce qui pourrait être dû au fait que l'inclusion de plus de voisins conduit à une généralisation excessive.

Cette matrice révèle que le modèle est très efficace pour identifier les 'artifacts', mais il existe une confusion entre les classes 'murmur' et 'normal'. Les 'murmurs' sont parfois confondus avec des 'normals', et vice-versa.
Accuracy: 0.78

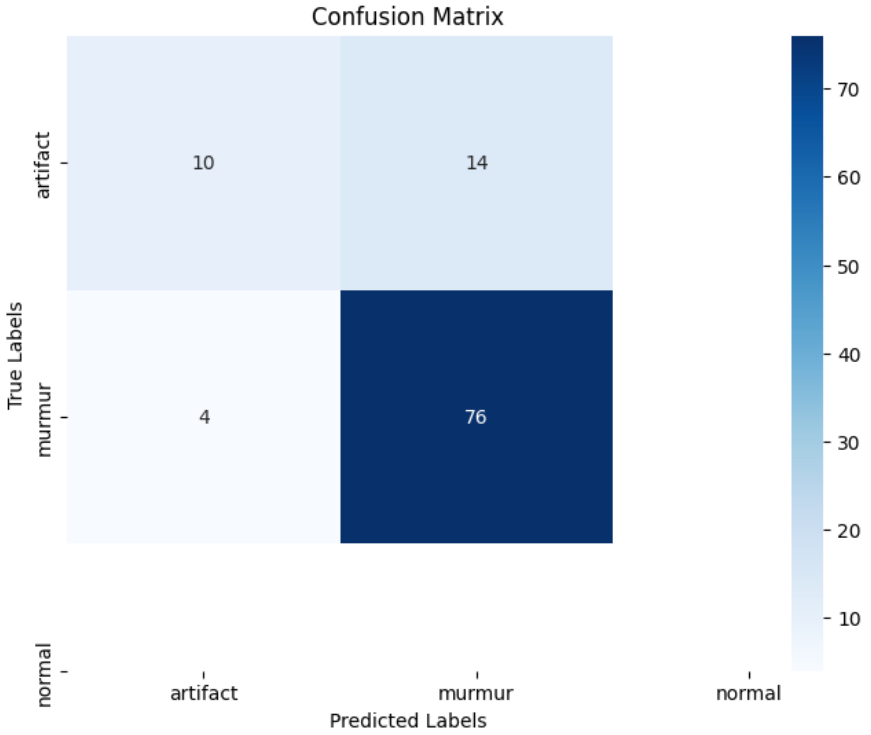


Jeu de données `DataMFCC_setB`



Il semble que le modèle atteigne des niveaux de performance élevés en termes de précision pour certains nombres de voisins spécifiques, ce qui pourrait indiquer des valeurs optimales de k. Cependant, il n'y a pas de correspondance directe entre l'exactitude et la précision, indiquant que le nombre de voisins optimal pour l'une n'est pas nécessairement optimal pour l'autre. En résumé, bien que le modèle soit très précis pour identifier les 'artifacts' et qu'il performe bien pour reconnaître les 'murmurs', il présente une faiblesse notable dans la distinction entre 'normal' et 'murmur'.
Accuracy: 0.84

Precision: 0.84



3.4 Conclusion

En conclusion, notre analyse sur les différentes datasets révèle plusieurs points clés

- Sur l'ensemble de la dataset, qui n'est pas équilibrée, les trois modèles tendent à bien prédire les classes 'artifact' et 'normal'. Cependant, la classe 'murmur' est systématiquement mal prédite et souvent confondue avec la classe 'normal'.
- Cette tendance est encore plus marquée dans la datasetB, qui ne contient que les classes 'murmur' et 'normal', où la prédiction pour la classe 'murmur' est également insatisfaisante.
- Pour la datasetA, qui est équilibrée, les résultats montrent que les prédictions, bien que non excellentes, sont supérieures à celles obtenues avec les autres datasets.

4 Méthodes Non Supervisées

Ici, nous expliquerons la méthode non supervisée que nous avons utilisée pour ce projet. Nous discuterons des détails de l'analyse non supervisée, des résultats obtenus, et de la manière dont ces résultats sont interprétés.

4.1 Modèle k-means

KMeans, l'un des algorithmes de clustering les plus répandus en apprentissage automatique non supervisé, se distingue par son objectif fondamental : regrouper un ensemble de données en plusieurs clusters distincts en fonction de leurs caractéristiques. Cette démarche vise à minimiser la variance au sein de chaque cluster, tout en maximisant la variance entre les clusters. En d'autres termes, KMeans cherche à créer des groupes

d'éléments de telle manière que les éléments à l'intérieur d'un même groupe se ressemblent davantage les uns les autres, tout en étant distincts des éléments appartenant à d'autres groupes. Cette notion de séparation optimale des données en clusters cohérents en fait un outil puissant pour l'exploration de structures sous-jacentes dans des ensembles de données non étiquetés. KMeans repose sur un processus itératif qui réaffecte les échantillons aux clusters en fonction de la proximité de leurs caractéristiques, jusqu'à ce qu'une convergence soit atteinte et que les clusters finaux soient établis.

Algorithm 3 Algorithme KMeans

Entrée:
X : ensemble de données à regrouper
K : nombre de clusters souhaité
ϵ : tolérance pour la convergence
Initialisation: Choisir K centroïdes initiaux (peuvent être aléatoires ou basés sur des heuristiques)
repeat
    for chaque point **x** dans **X do**
        Affecter **x** au cluster dont le centroïde est le plus proche
    for chaque cluster **do**
        Recalculer le centroïde du cluster comme la moyenne de ses points
until convergence (lorsque les centroïdes ne changent que de façon marginale) ou nombre d'itérations maximum atteint
Sortie: Les clusters formés et les centroïdes correspondants.

X : L'ensemble de données à regrouper.
K : Le nombre de clusters souhaité.
ϵ : La tolérance pour la convergence.

Classification avec prétraitement

Dans cette section, nous explorons la classification des enregistrements des battemnts cardiaque en utilisant l'algorithme KMeans avec prétraitement des données. Le prétraitement consiste en une réduction de dimension par Analyse en Composantes Principales (ACP) sur l'ensemble des données. L'objectif est de regrouper les données en clusters en utilisant des caractéristiques extraites par l'ACP, puis d'évaluer la performance du modèle.

4.2 Prétraitement des Données

Pour améliorer la qualité de la classification, nous utilisons l'ACP pour réduire la dimension des données tout en préservant l'essentiel de l'information. L'ACP transforme l'ensemble des données en un espace de dimension réduite en sélectionnant les composantes principales qui expliquent la variance maximale des données, ce qui signifie que KMeans opère dans un espace de caractéristiques de plus faible dimension.

4.2.1 Entraînement du modèle

Le modèle KMeans est entraîné sur les données transformées par l'ACP. Il effectue le regroupement des données en clusters, sans avoir connaissance des étiquettes de

classe réelles. Les centroïdes de chaque cluster sont calculés de manière itérative.

Jeu de données complet `DataMFCC`

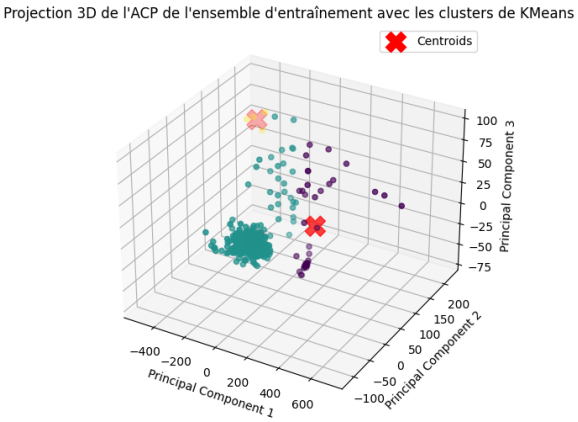


Figura 1: Matrice de confusion

Jeu de données `DataMFCC_setA`

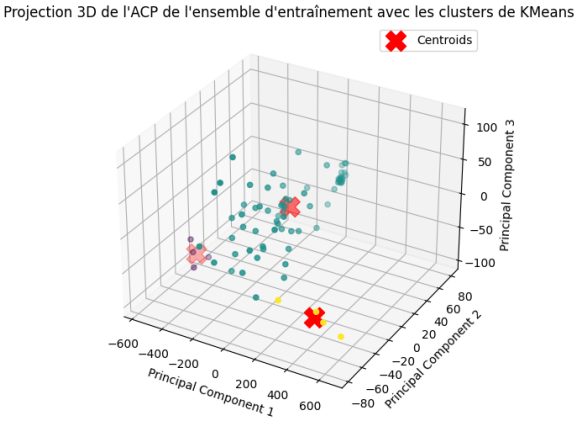


Figura 2: Matrice de confusion

Jeu de données `DataMFCC_setB`

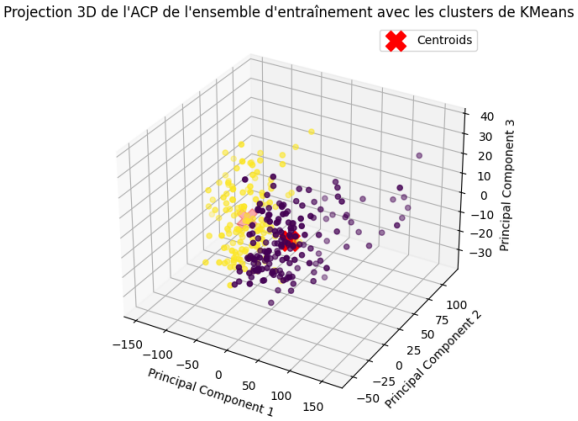


Figura 3: Matrice de confusion

4.2.2 Résultats obtenus

Les résultats de la classification KMeans sont évalués en utilisant deux mesures clés, la matrice de confusion et le score de performance.

Jeu de données complet `DataMFCC`

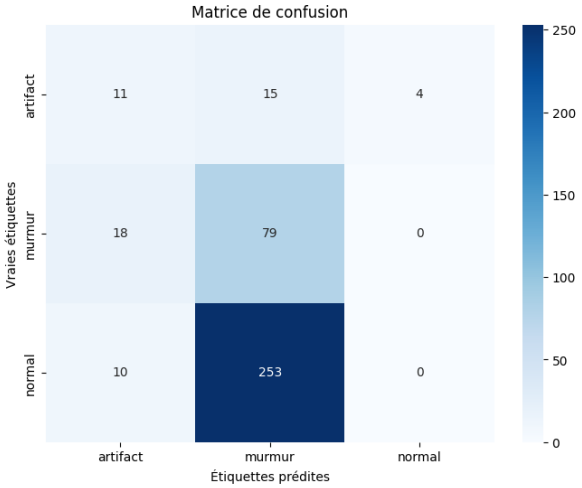


Figura 4: Matrice de confusion

Jeu de données `DataMFCC_setA`

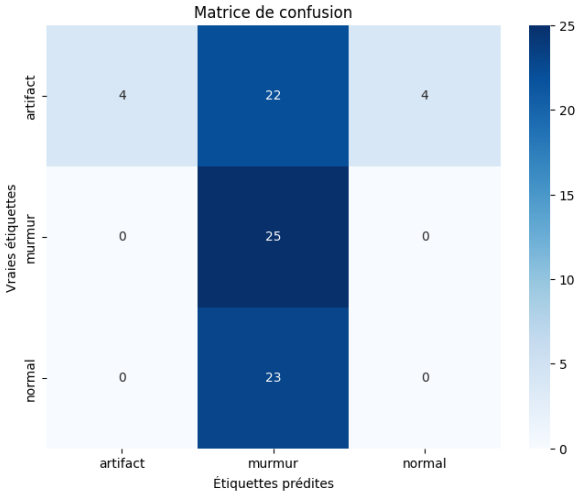


Figura 5: Matrice de confusion

Jeu de données `DataMFCC_setB`

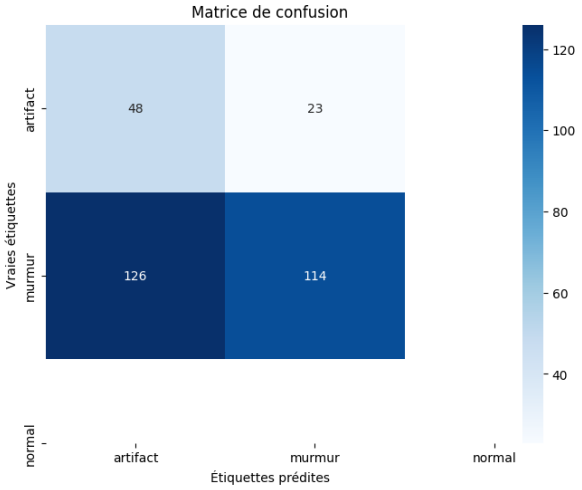


Figura 6: Matrice de confusion

La matrice de confusion est un outil essentiel pour évaluer la performance du modèle. Elle révèle comment les données réelles correspondent aux clusters prédits. Chaque entrée de la matrice indique combien d'échantillons appartiennent à une classe particulière. Ainsi, elle met en évidence les vrais positifs, les vrais négatifs, les faux positifs et les faux négatifs, permettant ainsi d'analyser en détail la qualité de la classification.

Le score de performance est une mesure globale de la précision du modèle. Il mesure la proportion d'échantillons correctement classés parmi l'ensemble des échantillons. Dans ce cas, nous avons trouvé des scores plutôt bas : 0.23 pour le data set general, 0.37 pour le data set A et 0.52 pour le data set B. Le score plus élevé du data set B par rapport aux deux autre pourrai s'expliqué par le fait qu'il contienne que deux classes. Nous avons aussi essayé de réaliser la classification sans ACP mais les resultats étaient moins bon, nous ne l'avons donc pas gardé.