**Projet Machine Learning**

HILLAIRET Oceane  
DAMIN Thomas  
FURTADO LEAL Carla

# Description du jeu de données

Le jeu de données se compose de **1319 observations et 9 caractéristiques**. En examinant les types de données, nous constatons que la majorité des caractéristiques sont numériques, avec une seule caractéristique de type objet représentant la variable cible. Cela suggère que nous devrons utiliser des techniques spécifiques pour manipuler les données et en extraire des informations pertinentes.

Ces données fournissent une base solide pour évaluer la santé cardiovasculaire des patients, offrant un aperçu précieux de leurs caractéristiques physiologiques et des résultats des tests médicaux. Il y a des caractéristiques classiques comme le sexe l’âge ou encore la fréquence cardiaque, mais aussi des aspects plus techniques.

La glycémie fournit des indications sur le taux de sucre dans le sang des patients. Les variables liées à la pression artérielle, à savoir la pression artérielle systolique (Systolic blood pressure) et la pression artérielle diastolique (Diastolic blood pressure), sont également présentes. Ces deux mesures, sont essentielles pour avoir un aperçu sur la santé cardiaque en indiquant la pression exercée par le sang sur les parois des artères lors des battements cardiaques.

Dans le contexte médical. CK-MB est une enzyme cardiaque, tandis que Troponin est une protéine libérée lors de lésions cardiaques. Ces deux variables peuvent fournir des informations cruciales sur les dommages cardiaques. Et enfin, nous avons une variable cible "Résultat" qui indique si le test effectué sur le patient est positif ou négatif.

Une image contenant texte, reçu, capture d’écran, Police

Description générée automatiquement

Figure: Extrait du jeu de données

En examinant les types de données, nous constatons que la majorité des caractéristiques sont numériques, avec une seule caractéristique de type objet représentant la variable cible. Cela suggère que nous devrons utiliser des techniques spécifiques pour manipuler les données et en extraire des informations pertinentes.

Une image contenant texte, capture d’écran, Rectangle, diagramme

Description générée automatiquement

Figure 2 : Répartition des cas

Les données fournies comprennent neuf variables médicales clés, allant de l'âge des patients à leurs résultats de tests. L'âge moyen des patients est d'environ 56 ans, avec une répartition majoritairement masculine. La fréquence cardiaque moyenne est de 78 battements par minute, tandis que les mesures de pression artérielle systolique et diastolique affichent des moyennes de 127.17 mmHg et 72.27 mmHg respectivement. Le taux moyen de sucre dans le sang est d'environ 146.63 mg/dL, avec des niveaux d'enzymes cardiaques (CK-MB et Troponin) présentant des moyennes de 15.27 et 0.36 respectivement.

L'analyse statistique initiale révèle plusieurs éléments clés. Tout d'abord, nous notons une distribution variée des variables numériques, chacune présentant des moyennes, des écarts-types et des plages de valeurs différents. En examinant la répartition des résultats des tests, nous constatons que près de **61% des tests sont positifs**, tandis que **39% sont négatifs**. De plus, la répartition entre les hommes et les femmes dans le jeu de données est déséquilibrée, avec environ **66% d'hommes** et **34% de femmes**.

Une image contenant texte, capture d’écran, Police, reçu

Description générée automatiquement

Figure 3 : Description du jeu de données

De plus, un nuage de points a été utilisé pour examiner la relation entre la pression artérielle systolique et l'âge, différenciée par le résultat du test, ainsi que des boîtes à moustaches qui elles ont été utilisées pour détecter et visualiser les valeurs aberrantes dans les variables numériques.

Dans le cadre de l'analyse des données, une étape cruciale consiste à identifier les relations entre les différentes variables en utilisant la corrélation. Pour ce faire, nous avons éliminé la variable cible "Result" du jeu de données initial, créant ainsi un nouveau DataFrame appelé ne contenant que les caractéristiques afin de calculer la matrice de corrélation entre celles-ci. Enfin nous avons créé un nouveau DataFrame contenant uniquement les caractéristiques pertinentes.

Pour visualiser et interpréter ces relations de corrélation, nous avons utilisé un graphique qui affiche la matrice de corrélation des caractéristiques pertinentes. La visualisation finale nous permet de rapidement identifier les paires de caractéristiques ayant des relations fortes et significatives entre elles, dans le cas dessous les variables **systolic blood pressure** et **diastolic blood pressure** ont un coefficient de corrélation proche de **0,6 et sont donc fortement corrélées**. On supprime donc pour la suite de l’analyse la colonne systolic blood pressure.

Une image contenant texte, capture d’écran, diagramme, carré

Description générée automatiquement

Figure 4 : Matrice de corrélation

La division de l'ensemble de données en ensembles d'entraînement et de test est une étape cruciale dans le processus d'apprentissage automatique.

Cette division est effectuée en tenant compte de la distribution des classes dans l'ensemble de données, afin de garantir une représentation équilibrée des différentes catégories dans chaque ensemble.

Dans notre cas, la distribution des classes dans l'ensemble d'entraînement montre que 57% des données appartiennent à la classe 0 et 43% à la classe 1. De même, dans l'ensemble de test, la distribution des classes est légèrement différente avec 56% des données appartenant à la classe 0 et 44% à la classe 1.

Cette répartition équilibrée des classes entre l'ensemble d'entraînement et l'ensemble de test est cruciale pour garantir une évaluation juste et précise des performances du modèle. Une répartition inégale des classes peut entraîner un biais dans l'évaluation du modèle, affectant sa capacité à généraliser correctement sur de nouvelles données.

En prenant en compte cette distribution des classes, nous nous assurons que notre modèle est entraîné et évalué de manière impartiale, ce qui permet d'obtenir des résultats fiables et généralisables dans des situations réelles.

# Preprocessing

Le preprocessing, ou le pré-traitement des données, est une étape cruciale. L’objectif est de rendre les données utilisables pour l’apprentissage et l’implémentation des modèles.

Le prétraitement des données inclue :

* La suppression des valeurs aberrantes
* La gestion des valeurs manquantes
* La normalisation ou la standardisation des variables numériques
* L’encodage des variables qualitatives / catégorielles

Il s’agit donc de corriger les erreurs (outliers, bruits…) pour obtenir des données précises et cohérentes pour des modèles fiables.

Dans notre cas voici les traitements opérés :

* Nettoyage des valeurs aberrantes

Le premier preprocessing que nous avons fait faisait qu’il restait que peu de données, nos modèles devenaient très sensibles au sur apprentissage. Nous avons alors décidé de garder toutes les données en ne supprimant que les valeurs aberrantes de battement par minute du cœur (1111, ce qui n’est pas humainement possible). Ainsi que l’une des variables corrélées.

* Encodage

L’encodage consiste à convertir la façon dont l’information est présentée. En claire, c’est la façon de passer d’une variable qualitative à une variable de type numérique, tout en préservant l’information.

La seule variable dont nous avons changé l’encodage est notre variable cible « Résult ». Nous avons utilisé la « LabelEncoder » de la bibliothèque sklearn qui, suivant la documentation, est réservé à la variable cible. Il permet de transformer les catégories en valeurs binaires.

Une autre étape importante avant de placer à l’implémentation des données est la séparation entre données d’apprentissages et données de test. Cela permet de limiter l’effet de l’overffiting. On entraine les modèles sur une partie des données (train-set) et on évalue sa performance sur les données qui n’ont pas été utilisées lors de l’apprentissage (test-set). En effet, sinon il suffit que le modèle « apprenne » les données pour pouvoir les classer correctement, mais il sera incapable de classer correctement des données qu’il n’aura jamais vu.

# Analyse comparative

On cherche le modèle qui obtiendra les meilleurs résultats. Etant donné que l’on veut pouvoir prédire la présence ou l’absence de maladie cardiaque, il faut définir le type d’erreur que l’on cherche à minimiser :

* Les faux positifs : prédire la présence de maladie cardiaque chez une personne à tort.
* Les faux négatifs : prédire l’absence de maladie cardiaque chez une personnes à tort.

Dans notre cas l’erreur la plus grave serait de prédire l’absence de maladie cardiaque à tort, donc on va chercher à minimiser le nombre de faux négatifs.

Ainsi pour évaluer nos modèles nous utiliserons le f1-score, une métrique qui fait la moyenne entre les valeurs du recall et de la précision.

## Apprentissage-supervisé

* Arbre de décision

Cette classification repose sur des règles déterminées par une suite de tests sur la valeur des attributs qui permettent de créer des sous-ensembles de données. L’objectif est de déterminer les tests, les règles qui permettent à l’issue de tous les tests d’obtenir des sous-ensembles de données appartenant à la même classe (ou presque).

L’implémentation du modèle nous permet d’obtenir l’arbre ci-dessous :

Une image contenant diagramme, ligne, conception

Description générée automatiquement

Figure 5: Arbre de décision

Nous pouvons constater que l’ensemble des variables sont utilisées dans l’arbre.

Une image contenant capture d’écran, Caractère coloré, diagramme, texte

Description générée automatiquement

Figure 6 : matrice de confusion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.99 | 0.98 | 0.99 | 102 |
| 1 | 0.99 | 0.99 | 0.99 | 162 |

Figure 7: Rapport de classification

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 100 cas négatifs sur 102 et 161 cas positifs sur 162.
* 99% des cas positifs sont correctement prédits
* 99% des prédictions positives sont effectivement des cas positifs
* Il n’y a qu’un seul faux négatif

Une image contenant texte, Tracé, diagramme, capture d’écran

Description générée automatiquement

Figure 8 : Courbes d'apprentissage de l'arbre de décision

Néanmoins ce graphique nous permet de dire qu’il y a un peu d’overfitting. La courbe bleu présente l’évolution du F1 score avec le nombre de données, sur les données d’apprentissage, la courbe orange sur les données de test.

Le modèle ne fait aucune erreur sur les données d’apprentissage (toujours à 100%), mais sur les données de test on peut voir que les performances sont plus faibles même si elles restent au-dessus de 97%. On constate également que les performances augmentent avec le nombre de données.

* Forêt aléatoire

Le RandomForest est une méthode d’ensemble qui consiste a entrainer plusieurs arbres de décisions et à voter pour la classe majoritaire.

Une image contenant texte, capture d’écran, diagramme, Tracé

Description générée automatiquement

Figure 9 : Importance des variables de l'arbre de décision

Le graphique ci-dessus, présente l’importance accordée à chaque variable par le modèle. Parmi les 7 variables, Troponin est celle qui a le plus d’importance avec CK-MB.

Une image contenant capture d’écran, Caractère coloré, carré, Rectangle

Description générée automatiquement

Figure 10 : Matrice de confusion du random forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.98 | 0.98 | 0.98 | 102 |
| 1 | 0.99 | 0.99 | 0.99 | 162 |

Figure 11 : Rapport de classification du random forest

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 100 cas négatifs sur 102 et 160 cas positifs sur 162.
* 99% des cas positifs sont correctement prédits
* 99% des prédictions positives sont effectivement des cas positifs
* Il n’y a que 2 faux négatif

Une image contenant texte, capture d’écran, Tracé, ligne

Description générée automatiquement

Figure 12 : Courbes d'apprentissage du random forest

On constate également de l’overfitting.

* Adaboost

C’est un modèle de Boosting, c’est-à-dire qu’il entraine un classifieurs sur des échantillons différents en prenant en compte et en corrigeant les erreurs à chaque fois. Ici le modèle utilisé est celui par défaut, l’arbre de décision.

Le graphique si dessous nous permet de visualiser l’importance des classifieurs faibles qui est la même à chaque fois. Les variables les plus importantes sont :

* CK-MB et Troponin

Comme pour l’arbre de décision, mais cette fois les autres variables ont plus d’impact sur les résultats.

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Figure 13 : Importances des variables adaboost

Une image contenant capture d’écran, Caractère coloré, texte, diagramme

Description générée automatiquement

Figure 14 : Matrice de confusion adaboost

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.99 | 0.97 | 0.98 | 102 |
| 1 | 0.98 | 0.99 | 0.99 | 162 |

Figure 15: Rapport de classification adaboost

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 99 cas négatifs sur 102 et 161 cas positifs sur 162.
* 98% des cas positifs sont correctement prédits
* 99% des prédictions positives sont effectivement des cas positifs
* Il n’y a qu’un seul faux négatif

Une image contenant texte, capture d’écran, Tracé, diagramme

Description générée automatiquement

Figure 16 : courbes d'apprentissage adaboost

De la même façon que pour les précédents modèles, il y a de l’overfitting, bien que les performances soient très bonnes.

* Machine à vecteur de support

Il s’agit d’une famille de classifieurs qui sépare les données grâce à une frontière qui maximise la distance entre les classes. Mais cela suppose que les données soient séparables de manière linéaire.

Une image contenant texte, capture d’écran, jaune, diagramme

Description générée automatiquementUne image contenant texte, capture d’écran, Caractère coloré, Graphique

Description générée automatiquement

Une image contenant texte, capture d’écran, Caractère coloré, Graphique

Description générée automatiquementUne image contenant texte, capture d’écran, graphisme, Graphique

Description générée automatiquement

Figure 17 : espace caratéristique

Ces graphiques nous permettent de visualiser la séparation des classes dans l’espace caractéristique. On peut voir qu’en fonction des variables, la séparation n’est pas nécessairement linéaire. Ce qui pourrait expliquer les faibles performances du modèle.

Une image contenant capture d’écran, texte, diagramme, Rectangle

Description générée automatiquement

Figure 18 : Matrice de confusion svm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.639 | 0.40 | 0.49 | 102 |
| 1 | 0.98 | 0.85 | 0.76 | 162 |

Figure 19 : rapport de classification svm

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 41 cas négatifs sur 102 et 138 cas positifs sur 162.
* 98% des cas positifs sont correctement prédits
* 85% des prédictions positives sont effectivement des cas positifs
* Il y a 24 faux négatifs

Une image contenant texte, diagramme, Tracé, capture d’écran

Description générée automatiquement

Figure 20 : Courbes d'apprentissage svm

Il n’y a pas d’overfitting ici, mais les performances sont moins bonnes. Il y a plus de faux négatifs.

* K-nearest neighbors (KNN)

Cet algorithme consiste de manière simple à attribuer à une observation, la classe majoritaire parmi ses voisins. Ici on considère 5 voisins.

A la différence des SVM, lorsque l’on visualise la région de décision on constate qu’elle s’adapte assez bien aux données, même si cela n’est pas toujours optimale. Il ne s’agit plis de faire une séparation linéaire.

Une image contenant texte, capture d’écran, jaune, diagramme

Description générée automatiquementUne image contenant texte, capture d’écran, jaune, carte

Description générée automatiquement

Figure 21 : Région de décision KNN

Une image contenant capture d’écran, texte, Caractère coloré, diagramme

Description générée automatiquement

Figure 22 : matrice de confusion knn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.47 | 0.46 | 0.47 | 102 |
| 1 | 0.68 | 0.67 | 0.67 | 162 |

Figure 23: Rapport de classification svm

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 47 cas négatifs sur 102 et 109cas positifs sur 162.
* 68% des cas positifs sont correctement prédits
* 67% des prédictions positives sont effectivement des cas positifs
* Il y a 53 faux négatifs

Une image contenant texte, capture d’écran, Tracé, diagramme

Description générée automatiquement

Figure 24 : Courbes d'apprentissage svm

Les performances de ce modèle diminuent avec le nombre de données et les performances sur le test-set sont moins bonnes que sur le train-set.

* Naïve Bayes

Ce classifieur s’appuie sur le théorème de Bayes, qui s’appuie sur les probabilités conditionnelles, en prenant en compte plusieurs variables. Mais en supposant qu’elles sont indépendantes les unes des autres.

Une image contenant capture d’écran, texte, Caractère coloré, diagramme

Description générée automatiquement

Figure 25 : matrice de confusion naif bayes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.53 | 1 | 0.69 | 102 |
| 1 | 1 | 0.43 | 0.60 | 162 |

Figure 26 : Rapport de classification naif bayes

La matrice de confusion et le rapport de classification nous apprennent que :

* Le modèle prédit correctement 102 cas négatifs sur 102 et 70 cas positifs sur 162.
* 100% des cas positifs sont correctement prédits
* 0.43% des prédictions positives sont effectivement des cas positifs
* Il y a 92 faux négatifs

Ce modèle prédit essentiellement les cas négatifs, puisque qu’il classe 194 observations dans cette classe et 70 dans l’autre. De ce fait, même si tous les cas négatifs sont bien identifiés, il y a beaucoup de faux négatifs. Or c’est le type d’erreur que nous cherchons à éviter.

On peut expliquer ces résultats en visualisant les probabilités à priori et à postériori.

A priori, il y a plus de cas positifs dans le jeu de données et donc plus de chances de tomber sur un individu atteint de maladies cardiaques. Mais lorsque que l’on prend en compte les variables, on voit que la probabilité de tomber sur individu sain est plus élevé.

Une image contenant capture d’écran, Tracé, diagramme, Rectangle

Description générée automatiquement

Figure 27 : probabilités à priori et à postériori

Une image contenant texte, diagramme, Tracé, ligne

Description générée automatiquement

Figure 28 : Courbes d'apprentissage naif bayes

L’analyse des performances dans le train et le test set ne révèle pas d’overfitting, mais une performance qui diminue avec le nombre de données.

## Sélection du modèle

## Une image contenant texte, diagramme, ligne, capture d’écran Description générée automatiquement

Figure 29 : Les scores des différents modèles

De manière générale ce les modèles à base d’arbre et AdaBoost qui obtiennent les meilleures performances. Leurs accuracy sont tous égaux (raison de plus pour ne pas l’utiliser comme métrique). Néanmoins, un modèle se détache des autres, l’Arbre de Décision avec un f1 de 99.08%.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Arbre de Décision | Random Forest | AdaBoost |
| F1 score | 0.9898 | 0.9877 | 0.9877 |

Figure 30 : les f1-scores des 3 meilleurs modèles

Afin de confirmer ce choix et d’améliorer si possibles les performances de notre classifieur nous avons décidé d’ajuster les hyperparamètres, de faire un apprentissage sur des données équilibrées à l’aide du sur-échantillonnage. En effet, notre jeu de donnée est déséquilibré. Même si le déséquilibre concerne la classe positive. De plus cela pourrait réduire l’overfitting observé.

On procède à un sur-échantillonnage qui utilise les données existantes et les duplique pour avoir autant de cas positifs que négatifs. Nous avons fait ce choix car nous avons une variable catégorielle dans nos données.

Une image contenant texte, capture d’écran, Tracé, ligne

Description générée automatiquementUne image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquementUne image contenant texte, capture d’écran, Tracé, diagramme

Description générée automatiquement

Figure 31 : les courbes d'apprentissage des 3 meilleurs modèles après amélioration

Ces améliorations ont permis de résoudre le problème d’overfitting, mais comme nous nous y attendions cela n’a pas significativement améliorer les performances des modèles.

Le meilleur modèle reste l’arbre de décision.

# IV- Analyse comparative

## Apprentissage non-supervisé

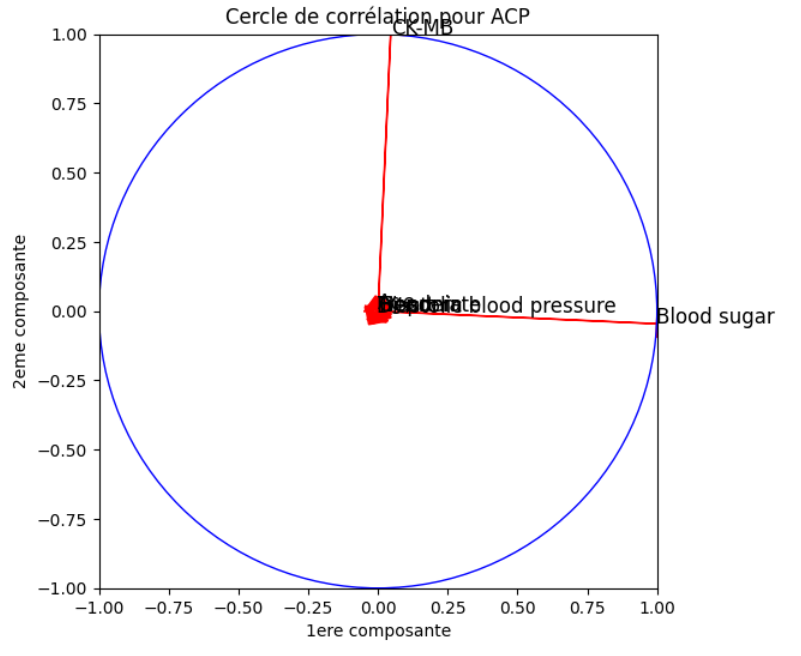
Dans ce cas-là, on cherche le modèle qui obtiendra les meilleurs groupements. Pour savoir si le groupement est bien réalisé, nous utiliserons le score de silhouette qui représente la différence l’inertie intra-classe et l’inertie inter-classe.

Pour répondre à notre problématique, nous souhaitons regrouper les patients en deux catégories, avoir une maladie cardiaque ou non. Nous essayerons aussi avec plus de classes.

Afin de pouvoir visualiser le clustering, nous décidons d’utiliser une ACP (analyse en composante principale) afin de réduire le nombre de dimensions. Cette ACP sera réutilisé pour visualiser les autres résultats d’algorithmes. Les algorithmes en questions seront toujours appliqués sur les données réelles.

Nous choisissons deux composantes car la première explique 67 % de la variance et la seconde 25%.

Suite à l’ACP, nous obtenons le cercle de corrélation suivant :



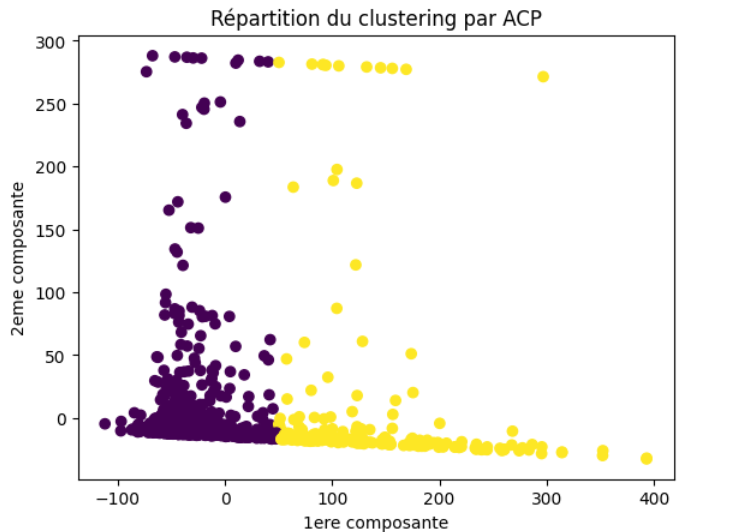
On peut alors en déduire que la première composante est grandement impacté par la variable *Blood sugar* et que la seconde par *CK-MB.*

* K-Means Clusturing

Algorithme de clustering son fonctionnement consiste à initialiser K centröides (centre des clusters) aléatoirement. Ensuite un calcul de distance est réalisé entre chaque point et chaque centröides. Les points sont alors désigner au centroïde le plus proche. Les centres sont alors déplacés en calculant la moyenne des distances entre chaque point faisant partie d’un cluster.

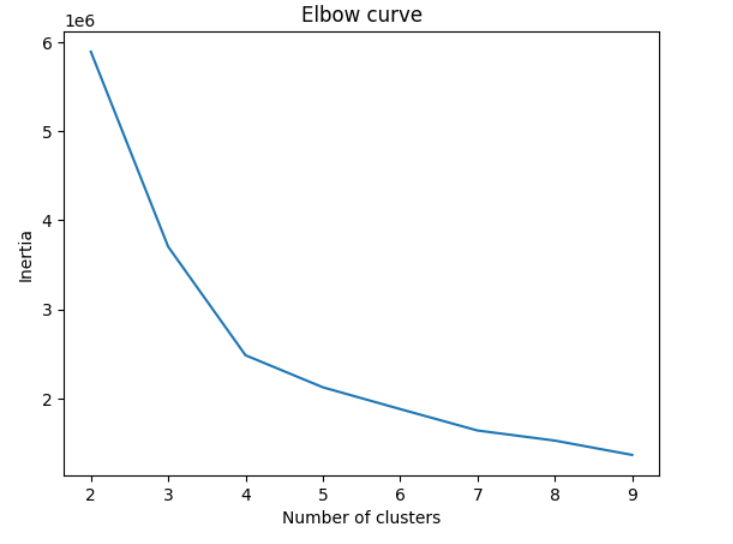
L’algorithme itère jusqu’à ce que les centröides soient immobiles entre deux itérations.

Suite à l’ACP nous pouvons alors afficher le graphique des deux regroupements réalisé par K-Means :

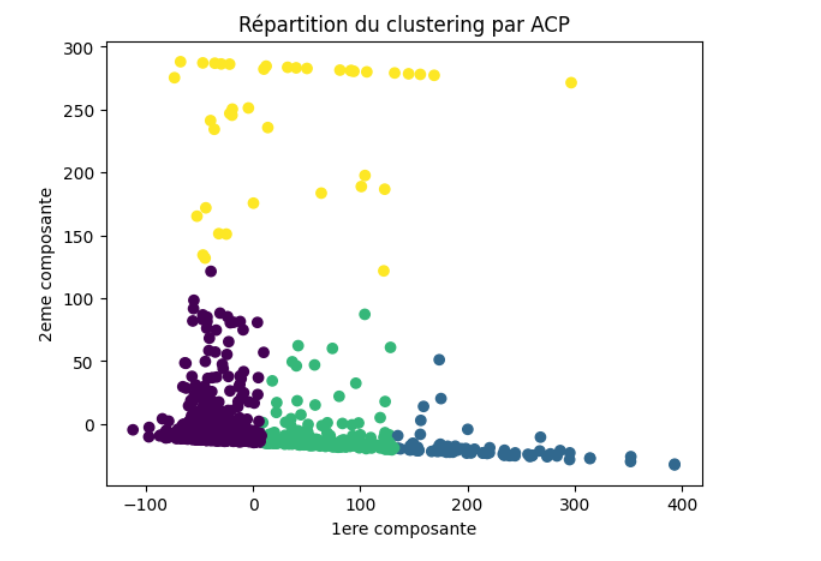


Notre modèle de K-Means possède un score de silhouette de 0.57.

Même si notre problématique consiste à regrouper en deux catégories, il peut être pertinent de cherche le nombre de clusters le plus appropriés. Pour faire ceci nous utilisons la méthode du coude en appliquant plusieurs fois le modèle.



On remarque alors que le nombre de clusters optimal est quatre. En réappliquant K-Means avec quatre clusters nous obtenons ce graphique :



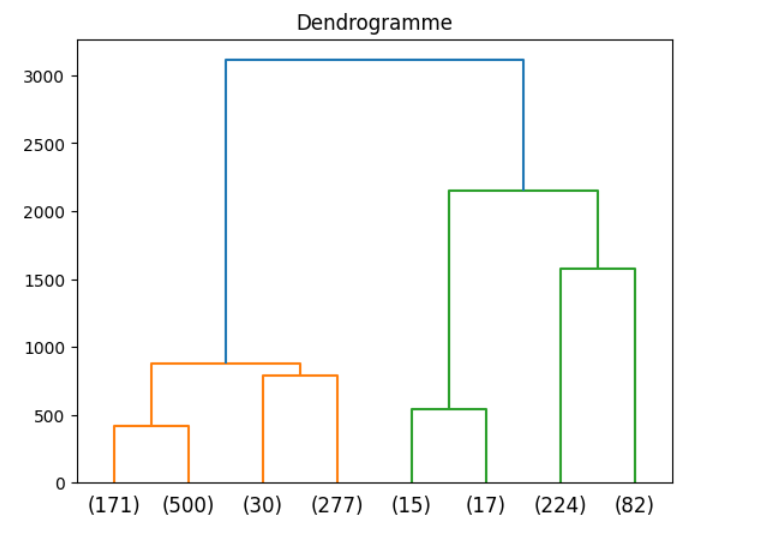
Avec un score de silhouette de 0.50.  
Ce modèle aurait pu classer les patients en fonction du type de maladie cardiaque par exemple. Cela restant une hypothèse. Cependant avec quatre clusters le modèle n’est pas plus performant et ne sera donc pas sélectionné.

* Clusturing Hierarchique Agglomératif

Un inconvénient potentiel du clustering *K*-means est qu’il nous oblige à spécifier à l’avance le nombre de clusters *K*. Le clustering hiérarchique est une approche alternative qui n’exige pas que nous nous engagions à un choix particulier de *K*.

Le clustering hiérarchique consiste en une série de fusions d’observations. Elle commence par les observations individuelles, chacune étant traitée comme un cluster distinct, et fusionne successivement les paires de cluster les plus proches, une paire à la fois. Le processus se poursuit de manière itérative jusqu’à ce que tous les clusters soient finalement fusionnés en un seul cluster. (<https://act6100.netlify.app/clustering/agglom.html>)

Pour le ce type de modèle, nous n’avons pas besoin de préciser le nombre de cluster de départ pour visualiser le dendrogramme. C’est en se basant dessus que nous allons choisir ce nombre.



On voit alors que, comme pour l’algorithme précédent, le choix de deux ou quatre classe semble pertinent. Nous avons donc décidé de réappliquer le clustering en fixant le nombre de cluster.

Pour deux classes, nous avons un score à 0.545  
Et pour quatre classes, un score à 0.50

De la même façon que K-Means, le nombre de cluster semble plus pertinent quand fixé à deux.

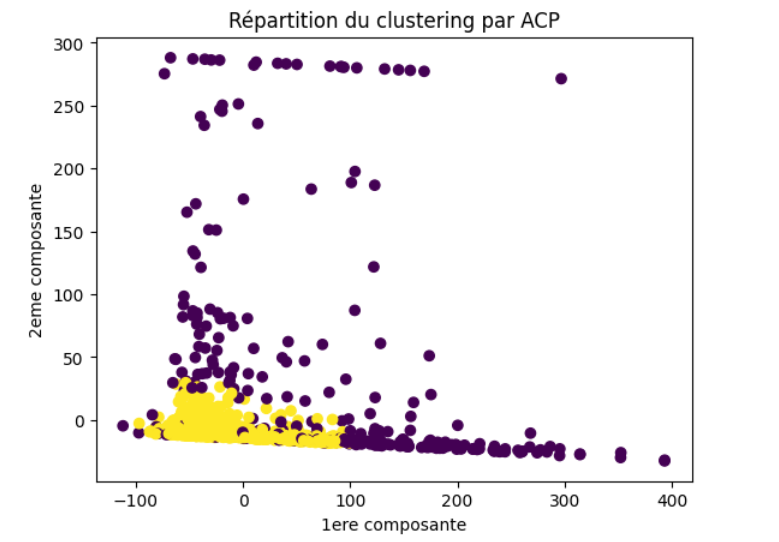
* Clusturing DBSCAN

Le DBSCAN (**Density-based Spatial Clustering of Applications with Noise)** est un algorithme simple qui définit des clusters en utilisant l’**estimation de la densité locale**. On peut le diviser en 4 étapes :

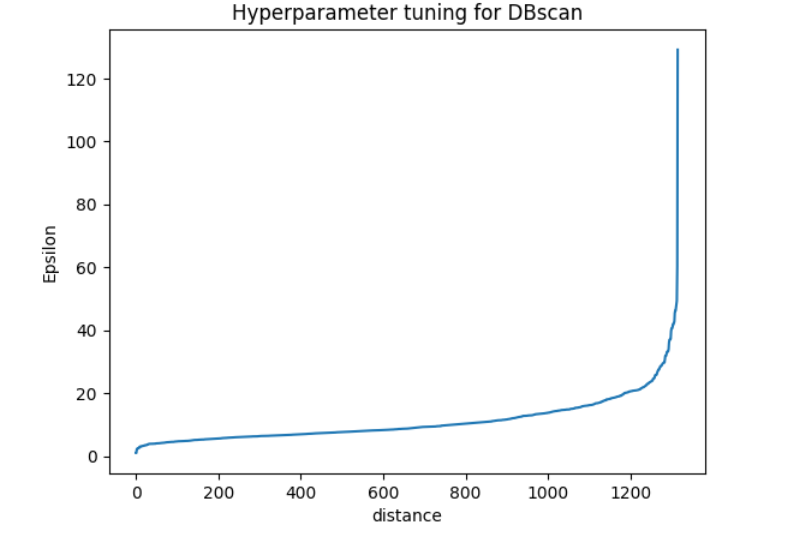
* Pour chaque observation on regarde le nombre de points à au plus une distance ε de celle-ci. On appelle cette zone le **ε-voisinage de l’observation**.
* Si une observation compte au moins un certain nombre de voisins y compris elle-même, elle est considérée comme une **observation cœur**. On a alors décelé une **observation à haute densité**.
* Toutes les observations au voisinage d’une observation cœur appartiennent au même cluster. Il peut y avoir des observations cœur proche les unes des autres. **Par conséquent de proche en proche on obtient une longue séquence d’observations cœur qui constitue un unique cluster**.
* Toute observation qui n’est pas une observation cœur et qui ne comporte pas d’observation cœur dans son voisinage est considérée comme une **anomalie**.

(<https://datascientest.com/machine-learning-clustering-dbscan>)

Pour l’implémentation du modèle, nous allons choisir comme hyperparamètre par défaut un epsilon de 25 et nombre de poids minimal de 40. Suite à cela nous obtenons ce graphique grâce à l’ACP :

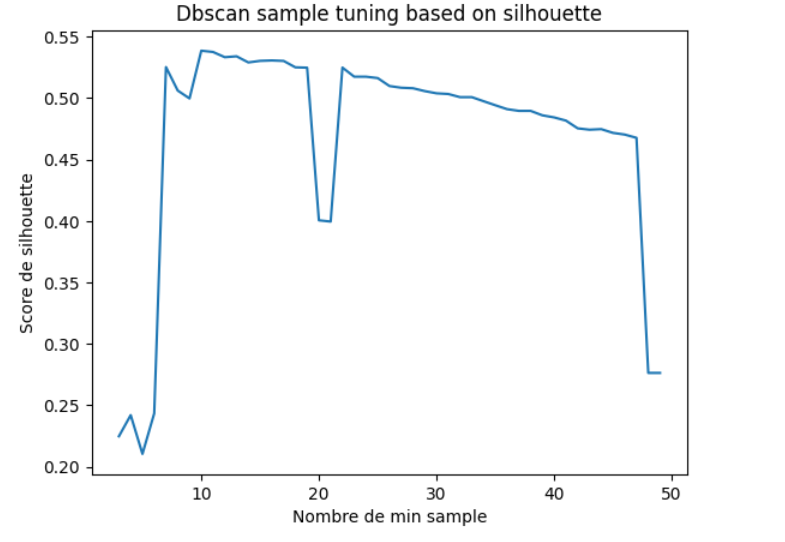


Maintenant, nous souhaitons optimiser les hyperparamètres de ce modèle pour qu’il soit le plus pertinent possible. Dans un premier temps nous allons optimiser le paramètre epsilon. En utilisant un calcul de distances du voisinage. Epsilon sera optimisé quand la courbe commencera à augmenter (de la même façon que la méthode du coude).



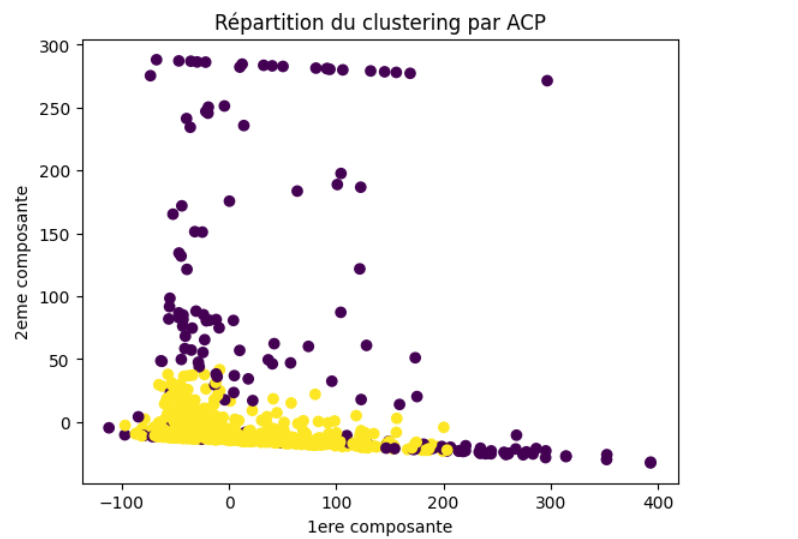
Nous pouvons donc remarquer que epsilon = 25 semble un bon choix d’hyperparamètre.

Il reste maintenant à optimiser un second hyperparamètre, le nombre de de points minimal pour qu’un point soit considéré core point. Pour optimiser celui-ci, nous lançons N modèles avec des valeurs différentes en voulant obtenir le meilleur score de silhouette.



Nous l’obtenons alors quand la valeur de l’hyperparamètre est fixée à dix.

En réappliquant le modèle avec les nouveaux hyperparamètres nous obtenons ce scatter plot par ACP :



Avec comme score de silhouette 0.538

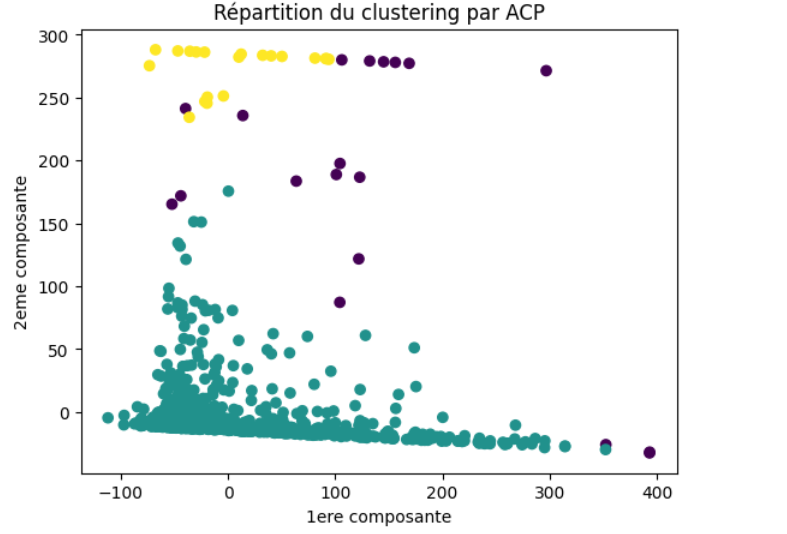
* Clusturing HDBSCAN

Algorithme hiérarchique de DBSCAN, il part du principe que :

* Les algorithmes de clustering basés sur la densité, comme DBSCAN, ne clusterisent que selon un seuil de densité global, ce qui va empecher de trouver des clusters de densité trop variables.
* Les algorithmes de type clustering hierarchique sont aussi intéressants, mais peuvent avoir une hiérarchie trop complexe, difficilement interprétable.
* Un autre problème rencontré est aussi la multiplication de paramètres, influençant grandement le résultat (par exemple le nombre de classes est à spécifier pour k-means)

Notons qu'un algorithme de clustering est différent d'un algorithme de partitionnement, comme k-means. Le but de ce dernier est d'associer à tout élément un des k regroupements, en minimisant la distance intra-regroupement. Dans notre définition du clustering, on s'autorise à avoir des points qui n'appartiennent à aucun regroupement: ils sont considérés comme étant du bruit.

Pour HDBCSAN, nous allons tout d’abord utiliser la même valeur pour savoir si un point est considéré comme core point (soit dix). La visualisation du modèle est la suivante :



Avec un score de silhouette de 0.66

Ce modèle semble être le plus performant (avec le score de silhouette le plus élevé). Nous allons alors essayer de l’optimiser le plus possible. Notamment pour le paramètre de la taille minimale d’un cluster. Pour ce faire nous allons l’optimiser pour maximiser le score de silhouette.



On remarque alors que cet hyperparamètre n’améliore pas la performance du modèle dans notre cas.

## Sélection du modèle

Pour sélectionner notre meilleur modèle, nous choisissons celui qui possède le meilleur score de silhouette. Dans ce cas c’est le modèle généré par l’algorithme HDBSCAN. On peut interpréter ce clustering de plusieurs façons. Une chose sûre est que les points verts correspondent sûrement aux patients malades (voir répartitions des classes dans la description du jeu de données). Les points violets ou jaunes sont alors l’autre population de patients, en prenant en compte que l’une des deux classes est alors un ensemble de points considérés comme étant du « bruit ».(voir définition de HDBSCAN).