# **RAPPORT**

# **Projet Machine Learning**



# SOMMAIRE

[SOMMAIRE 2](#_Toc1076106509)

[Jeu de Données : 3](#_Toc1583203233)

[Tache de Classification : 3](#_Toc1111221910)

[Préparation des données : 3](#_Toc46815812)

[Algorithmes de Classification: 3](#_Toc1971968501)

[Arbre de Décision ; 4](#_Toc509059278)

[Foret Aleatoire : 4](#_Toc1177366008)

[AdaBoost : 4](#_Toc795009728)

[Machine a Vecteurs de Support (SVM) 4](#_Toc106605647)

[K-Nearest Neighbors (KNN) 4](#_Toc1392375718)

[Naive Bayes : 4](#_Toc1483923934)

[Taches de Clustering 4](#_Toc13321918)

[Chargement des données et Visualisation : 5](#_Toc1470183981)

[Algorithmes de Clustering : 5](#_Toc912965590)

[K-Means Clustering : 6](#_Toc1991149879)

[Clustering Hiérarchique Agglomératif : 6](#_Toc1352635986)

[Clustering DBSCAN : 6](#_Toc1006076553)

[Clustering HDBSCAN : 6](#_Toc1866532225)

[Analyse Comparative : 6](#_Toc1952417144)

[Analyse de la classification 7](#_Toc464856850)

[Analyse du clustering 7](#_Toc1144493778)

## Jeu de Données :

Explication du jeu de données :

* Le jeu de données est une collection d'enregistrements médicaux comportant les colonnes suivantes :
* Age : L'âge du patient.
* Genre: Le genre du patient, où 1 pourrait indiquer les hommes et 0 les femmes, bien que cela ne soit pas spécifié explicitement.
* Fréquence cardiaque: La fréquence cardiaque du patient, probablement mesurée en battements par minute.
* Pression artérielle systolique : La pression artérielle systolique du patient
* Pression artérielle diastolique : La pression artérielle diastolique du patient
* Taux de sucre dans le sang: Le taux de sucre dans le sang du patient
* CK-MB : Une mesure de la créatine kinase-MB, une enzyme libérée dans le sang lors de l'endommagement ou de la mort des cellules du muscle cardiaque, souvent utilisée pour diagnostiquer les crises cardiaques.
* Troponine : Une mesure de la troponine, une autre enzyme libérée lors de dommages au muscle cardiaque. Des niveaux élevés peuvent indiquer une crise cardiaque.

Le résultat du diagnostic, indiquant si le patient a été diagnostiqué comme positif ou négatif pour une certaine condition, probablement liée à une maladie cardiovasculaire ou à un risque de crise cardiaque.

Ce jeu de données pourrait être utilisé pour analyser des facteurs de risque, diagnostiquer des conditions médicales, ou pour la recherche médicale en général.

## 

## Tache de Classification :

### Préparation des données :

Pour préparer les données pour une tâche de classification, nous devons suivre plusieurs étapes pour nous assurer que les données sont dans un format approprié pour l'entraînement d'un modèle. Voici un aperçu des étapes principales :

1. Nettoyage des données:

* Valeurs manquantes
* Les valeurs aberrantes, ou les erreurs dans les données en les remplaçant par la moyenne/médiane, ou en utilisant une imputation).

2. Encodage des variables catégorielles :

* Pour le résultat, qui est une variable catégorielle binaire (positive/négative), nous pouvons également l'encoder en 0 et 1.

3. Normalisation/Standardisation:

* La normalisation (mise à l'échelle des valeurs entre 0 et 1)
* la standardisation (transformation des données pour avoir une moyenne de 0 et un écart-type de 1)

peut être nécessaire, surtout pour les variables comme la pression artérielle, le taux de sucre dans le sang, CK-MB, et la troponine.

4. Séparation des données : Diviser le jeu de données en un ensemble d'entraînement et un ensemble de test. Cela permet d'évaluer la performance du modèle sur des données non vues lors de l'entraînement. Typiquement, on utilise une répartition comme 80% pour l'entraînement et 20% pour le test.

5. Définition de la variable cible: Notre variable cible est le "Résultat". Nous devons nous assurer qu'elle est correctement formatée et séparée des variables explicatives (features).

Avec ces étapes complétées, les données sont maintenant prêtes à être utilisées pour entraîner un modèle de classification.

### Algorithmes de Classification:

// TO DO Tache de Thomas

#### Arbre de Décision ;

#### Foret Aleatoire :

### Explication du fonctionnement des modèles de classification utilisés avec leurs avantages et inconvénients.

Voici les explications du fonctionnement des modèles SVM, KNN et Naive Bayes qu’on a utilisé pour notre projet.

#### Explication Machine a vecteurs de support SVM :

* Avantages : Efficace dans les espaces de grande dimension c’est-à-dire quand il y a beaucoup de données à traiter, peut gérer des ensembles de données avec peu de données d'entraînement.
* Inconvénients : Le choix du noyau et des hyperparamètres peut être difficile, peut être lent pour de grandes quantités de données d'entraînement.

#### Explication K-Nearest Neighbors KNN :

* Avantages : Facile à comprendre, ne nécessite pas d'entraînement, peut gérer des données non linéaires et peut être efficace avec des données de grande dimension.
* Inconvénients : Le temps de prédiction peut être élevé et sensible aux valeurs aberrantes.

#### Explication Naive Bayes :

* Avantages : Simple, rapide, fonctionne bien avec des ensembles de données de grande dimension et peut être efficace même avec des violations mineures des hypothèses d'indépendance conditionnelle.
* Inconvénients : Fait l'hypothèse forte de l'indépendance conditionnelle entre les caractéristiques, ce qui peut ne pas être réaliste pour certains ensembles de données complexes.

#### Machine a Vecteurs de Support (SVM)

Voici le processus d'implémentation, d'entraînement, d'évaluation et d'interprétation du modèle de classification SVM (Machine à Vecteurs de Support).

Nous avons utilisé la bibliothèque `***scikit-learn***` pour implémenter le modèle SVM. La classe `***SVC***` a été importée depuis `***sklearn.svm***`.

Nous avons créé une instance de `SVC` avec un noyau linéaire et avons utilisé les données d'entraînement pour entraîner le modèle. Les données d'entraînement ont été préalablement définies comme `X\_train` et `y\_train`.

Nous avons utilisé le modèle entraîné pour prédire la présence ou l'absence de maladies cardiaques à partir du jeu de données.

Ensuite, nous avons évalué les performances du modèle en calculant son accuracy à l'aide de la fonction ***`accuracy\_score`*** et en générant un rapport de classification complet (incluant precision, recall, f1-score, et support) à l'aide de la fonction `classification\_report’.

#### K-Nearest Neighbors (KNN)

Voici le processus d'implémentation, d'entraînement, d'évaluation et d'interprétation du modèle KNN, nous avons utilisé la classe `KNeighborsClassifier` de la bibliothèque `scikit-learn

Nous avons créé une instance de `KNeighborsClassifier` sans spécifier de paramètres supplémentaires, ce qui utilise par défaut une valeur de k de 5. Par défaut, la valeur de k est définie à 5 pour le modèle k-plus proches voisins (KNN).

Ensuite, nous avons entraîné ce modèle sur l'ensemble de données d'entraînement en utilisant la méthode `fit`. En laissant les paramètres supplémentaires par default et on la couplant avec la méthode FIT on a un meilleur résultat.

Nous avons évalué les performances du modèle en calculant son accuracy (précision) à l'aide de la fonction `accuracy\_score`. De plus, nous avons généré un rapport de classification (precision, recall, f1-score, support) en utilisant la fonction `classification\_report`, ainsi qu'une matrice de confusion en utilisant la fonction `confusion\_matrix`.

#### Naive Bayes :

Pour implémenter le classifieur Naive Bayes, nous avons utilisé la classe `GaussianNB` de la bibliothèque `scikit-learn`. Nous avons importé la classe nécessaire à partir du module `sklearn.naive\_bayes`.

Puis nous avons créé une instance de `GaussianNB` sans spécifier de paramètres supplémentaires. Ensuite, nous avons entraîné ce modèle sur l'ensemble de données d'entraînement en utilisant la méthode `fit`comme pour le modèle KNN.

Ensuite nous avons utilisé le modèle entraîné pour prédire la présence ou l'absence de maladies cardiaques pour l'ensemble de test en utilisant la méthode `predict`.

Et pour finir nous avons évalué les performances du modèle en calculant son accuracy (précision) à l'aide de la fonction `accuracy\_score` fournie par `scikit-learn`. Et nous avons généré un rapport de classification (precision, recall, f1-score, support) en utilisant la fonction `classification\_report`, ainsi qu'une matrice de confusion en utilisant la fonction `confusion\_matrix`.

## Taches de Clustering

Les algorithmes de clustering travaillent en calculant la distance ou la dissimilarité entre les points de données pour déterminer leur proximité. Les critères et métriques spécifiques varient en fonction de l'algorithme utilisé. Parmi les algorithmes de clustering les plus populaires, on trouve K-means, le clustering hiérarchique, DBSCAN, et HDBSCAN, chacun ayant ses propres forces et faiblesses et étant adapté à différents types de données et exigences.

### Chargement des données et Visualisation :

Charger et visualiser vos données est une étape cruciale avant d'appliquer des algorithmes de clustering. Cette étape vous permet d'obtenir une compréhension préliminaire de la structure de vos données, d'identifier les outliers potentiels, et d'évaluer si un prétraitement supplémentaire est nécessaire.

#### **Étapes pour le Chargement et la Visualisation des Données :**

1. **Chargement des Données** :
2. **Examen Préliminaire** :

Examinez les premières lignes de votre dataset pour comprendre sa structure, les types de données et s'il y a des valeurs manquantes.

1. **Visualisation des Données**
2. **Réduction de Dimensionnalité pour Visualisation** : Pour les datasets avec de nombreuses caractéristiques, l'analyse en composantes principales (PCA) ou t-SNE peut être utilisée pour réduire les données à deux ou trois dimensions pour une visualisation facile.

### Algorithmes de Clustering :

*1. K-means :*

*- Fonctionnement: Divise l'ensemble de données en K clusters en minimisant la variance intra-cluster.*

*- Avantages :*

*- Simple et rapide.*

*- Efficace pour des ensembles de données de taille moyenne à grande avec des clusters de forme sphérique.*

*- Limitations :*

*- Sensible à l'initialisation des centroïdes et aux valeurs aberrantes.*

*- Nécessite de spécifier le nombre de clusters à l'avance.*

*- Applicabilité: Convient aux données avec des clusters de forme sphérique et de taille égale.*

*Une image contenant capture d’écran, Caractère coloré

Description générée automatiquement*

*2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) :*

*- Fonctionnement: Identifie les clusters en se basant sur la densité des points.*

*- Avantages :*

*- Peut détecter des clusters de formes arbitraires et de densités variables.*

*- Ne nécessite pas de spécifier le nombre de clusters à l'avance.*

*- Limitations:*

*- Sensible aux valeurs des paramètres `eps` et `min\_samples`.*

*- Peut avoir du mal avec des clusters de densités différentes.*

*- Applicabilité : Utile pour détecter des clusters de formes et de densités variables.*

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

Description générée automatiquement*

*3. HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) :*

*- Fonctionnement : Extension de DBSCAN qui utilise une approche hiérarchique.*

*- Avantages :*

*- Peut identifier des clusters de formes et de densités variées.*

*- Robuste aux valeurs des paramètres.*

*- Limitations :*

*- Peut être plus lent que DBSCAN en raison de sa complexité.*

*- Nécessite plus de mémoire pour stocker la structure hiérarchique.*

*- Applicabilité: Recommandé lorsque le nombre de clusters n'est pas connu à l'avance et pour des données complexes avec des clusters de formes et de densités variées.*

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

Description générée automatiquement*

*4. Clustering hiérarchique agglomératif :*

*- Fonctionnement: Construit une hiérarchie de clusters en fusionnant successivement les clusters les plus proches.*

*- Avantages :*

*- Peut capturer la structure hiérarchique intrinsèque des données.*

*- Ne nécessite pas de spécifier le nombre de clusters à l'avance.*

*- Limitations:*

*- Coûteux en termes de temps de calcul, surtout pour de grands ensembles de données.*

*- Sensible aux choix de la mesure de distance et de la méthode de liaison.*

*- Applicabilité: Utile pour explorer la structure hiérarchique des données sans spécifier le nombre de clusters à l'avance.*

*Une image contenant diagramme, texte, Plan, Dessin technique

Description générée automatiquement*

***Conclusion :***

*Le choix de l'algorithme de clustering dépendra de la structure des données, de la nature des clusters que vous recherchez et des performances spécifiques que vous recherchez en termes de temps de calcul et de précision du clustering. Il est souvent judicieux d'expérimenter plusieurs algorithmes et d'évaluer leurs performances sur votre ensemble de données spécifique pour trouver le meilleur choix.*

## Analyse Comparative :

### Analyse de la classification (SVM, KNN, Naive Bayes)

En analysant les résultats des trois modèles (SVM, KNN, Naive Bayes) sur la base des métriques d'évaluation suivantes : accuracy, classification report (précision, rappel, f1-score) et matrice de confusion, voici quelques observations :

**Résultat obtenu :**

|  |  |  |
| --- | --- | --- |
| SVM | KNN | Naive Bayes |
| - Accuracy : 93.67%    - Précision pour la classe False : 93%    - Précision pour la classe True : 95%    - Rappel (recall) pour la classe False : 97%    - Rappel pour la classe True : 89%    - F1-score pour la classe False : 95%    - F1-score pour la classe True : 92% | - Accuracy : 78.48%    - Précision pour la classe False : 77%    - Précision pour la classe True : 83%    - Rappel pour la classe False : 91%    - Rappel pour la classe True : 60%    - F1-score pour la classe False : 83%    - F1-score pour la classe True : 70% | - Accuracy : 94.94%    - Précision pour la classe False : 95%    - Précision pour la classe True : 95%    - Rappel pour la classe False : 97%    - Rappel pour la classe True : 92%    - F1-score pour la classe False : 96%    - F1-score pour la classe True : 94% |

*Note : l'accuracy mesure la performance globale du modèle, tandis que la précision et le rappel fournissent des informations sur la performance du modèle pour chaque classe individuellement. Le F1 Score est une mesure qui prend en compte à la fois la précision et le rappel pour fournir une mesure globale de la performance du modèle. Ces métriques sont toutes importantes pour évaluer la qualité des prédictions d'un modèle de classification.*

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

Description générée automatiquement

#### Conclusion (SVM, KNN, NB) :

Après avoir évalué les performances des trois modèles SVM, KNN et Naive Bayes pour la prédiction de la présence ou de l'absence de maladies cardiaques dans notre dataset nous pouvons tirer les conclusions suivantes :

Dans notre cas, les ***SVM*** ont montré des performances globalement élevées, avec un équilibre raisonnable entre la précision et le rappel pour les deux classes donc False et True. Mais, le rappel pour la classe positive (présence de maladies cardiaques True) était légèrement inférieur, ce qui peut nous laisser croire qu’il a tendance à sous-estimer certains cas de maladies cardiaques.

Dans notre analyse, les performances des ***KNN*** étaient relativement inférieures par rapport aux autres modèles. Ils ont montré une précision inferieur pour la classe négative (absence de maladies cardiaques donc False), et leur rappel pour la classe positive (présence de maladies cardiaques True) était plus bas aussi comparer au Naive Bayes et SVM.

Le modèle ***Naive Bayes*** a démontré les meilleures performances en termes d'accuracy, de précision, de rappel et de F1-score pour les deux classes True et False. Il a présenté une capacité solide à prédire avec précision surtout la présence ou l'absence de maladies cardiaques avec notre dataset, ce qui en fait le choix le plus approprié pour notre contexte de prediction.

### Analyse du clustering:

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

Description générée automatiquement

Ce graphique compare la performance de divers algorithmes de clustering en se fondant sur le score de silhouette et le temps de calcul. Il offre un aperçu direct de l'efficacité des algorithmes en termes de qualité de séparation des clusters et de rapidité d'exécution pour un jeu de données médical.

* Interprétation du graphique📊 :
* Score de Silhouette :

- K-Means montre le score de silhouette le plus élevé, ce qui implique que les clusters générés par K-Means sont plus cohérents et mieux séparés par rapport aux autres méthodes. Cela indique une bonne adéquation de cet algorithme pour les caractéristiques de vos données.

- DBSCAN a un score de silhouette légèrement inférieur à celui de l'agglomération, ce qui peut être dû à sa sensibilité aux paramètres `eps` et `min\_samples`. Un ajustement fin de ces paramètres pourrait améliorer ce score.

- HDBSCAN présente un score de silhouette plus faible, suggérant que les clusters formés peuvent ne pas être aussi distincts. Cela peut être lié à la nature des données ou à la capacité de l'algorithme à détecter des clusters de densité variable.

- Agglomération a un score de silhouette équivalent à celui de DBSCAN, ce qui indique qu'elle est également une méthode de clustering assez efficace pour cet ensemble de données.

* Temps de Calcul

- K-Means a le temps de calcul le plus long, bien qu'il reste relativement court. Cela reflète sa complexité de calcul un peu plus élevée due à l'itération sur les centres de clusters pour atteindre la convergence.

- DBSCAN et Agglomération montrent des temps de calcul très courts, ce qui indique que ces algorithmes sont particulièrement efficaces du point de vue computationnel pour cet ensemble de données.

- HDBSCAN a un temps de calcul intermédiaire, ce qui peut être dû à des calculs supplémentaires nécessaires pour évaluer la densité des données et les relations hiérarchiques.

* Conclusion :

Sur la base de ce graphique, K-Means semble être le choix optimal en termes de qualité de clustering mesurée par le score de silhouette. Cependant, il est également le plus coûteux en temps de calcul. Les algorithmes DBSCAN et Agglomération présentent un bon équilibre entre efficacité computationnelle et qualité de clustering, avec des scores de silhouette légèrement inférieurs mais des temps de calcul considérablement réduits. HDBSCAN semble être moins performant en termes de score de silhouette mais offre une méthode flexible, bien adaptée pour détecter des clusters de densités variables, ce qui pourrait être utile pour certaines applications spécifiques.

Remarque 📌: Il est important de considérer le contexte de vos données et vos objectifs spécifiques pour choisir l'algorithme de clustering le plus approprié. Si la distinction claire des clusters est prioritaire, K-Means peut être la meilleure option, tandis que pour des données complexes ou pour des applications nécessitant une rapidité de traitement, DBSCAN ou l'agglomération pourraient être préférables.