Ben Rbiha Yassine

Fassatoui Mohamed Amine

Année Universitaire :

2024/2025

Machine learning dans le Domaine de la Santé

Classification sur Données CSV

**Introduction :**

Dans le cadre de ce projet, nous avons réalisé une analyse approfondie des données issues d’un fichier intitulé ‘*heart.csv’*. L’objectif principal était d’appliquer des techniques de machine learning, en particulier la classification afin d’explorer et de modéliser les relations entre les variables présentes dans le jeu de données. Pour ce faire, nous avons testé et comparé plusieurs modèles de machine learning, en évaluant leurs performances à l’aide de métriques adaptées. Cette étude nous a permis de générer des résultats explicatifs et interprétables, apportant des informations précieuses sur les données tout en illustrant l’efficacité des modèles utilisés.

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

Description générée automatiquement**Data preprocessing :**

**dataset = pd.read\_csv(r"heart.csv")**

Elle permet de charger le fichier CSV heart.csv dans un DataFrame Pandas. Cela constitue la première étape avant toute analyse ou traitement des données, car elle permet d'importer les données depuis un fichier externe afin de les manipuler en Python.

Les colonnes de notre Dataset sont :   
**Age :** L'âge de la personne

**Sex :** Le sexe de la personne (M : Male , F : Female).

**ChestPainType :** Le type de douleur thoracique .

**RestingBP :** La pression artérielle au repos (en mmHg).

**Cholesterol :** Le taux de cholestérol sanguin (en mg/dl).

**FastingBS :** Le niveau de glucose sanguin à jeun (codé, 1 pour plus de 120 mg/dl et 0 pour moins de 120 mg/dl).

**RestingECG :** Les résultats de l'électrocardiogramme au repos (codés en catégories comme normal, ST,LVH)

**MaxHR :** La fréquence cardiaque maximale atteinte pendant l'exercice.

**ExerciseAngina :** Si la personne éprouve de l'angine (douleur thoracique) pendant l'exercice (Y pour oui, N pour non).

**Oldpeak :** La dépression ST induite par l'exercice par rapport au repos (indicateur d'anomalies cardiaques).

**ST\_Slope :** La pente de la portion ST du tracé ECG après l'exercice (catégorisé en Up,Flat ou Down).

**HeartDisease :** La présence ou l'absence de maladie cardiaque (probablement codé en 1 pour présence et 0 pour absence).

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

Description générée automatiquement

Nous avons dans notre Dataset 918 lignes et 12 colonnes , Cela représente un ensemble de données relativement conséquent pour l'analyse.

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

Description générée automatiquement

Le résultat de montre que **aucune valeur manquante** n'est présente dans la dataset, toutes les colonnes ont un total de 0 valeurs manquantes. Cela signifie que les données sont complètes et prêtes pour l'analyse sans avoir besoin d'un traitement d'imputation pour les valeurs manquantes.

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

Description générée automatiquement

X : Contient toutes les colonnes sauf la dernière (les variables indépendantes), c'est-à-dire les colonnes de 0 à 10 (total de 11 colonnes) : 'Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST\_Slope'.

**y** : Contient uniquement la dernière colonne (la variable cible ou dépendante) : 'HeartDisease', qui est la variable à prédire (présence ou absence de maladie cardiaque).

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

Description générée automatiquement

X.values : Convertit les caractéristiques (variables indépendantes) en un tableau NumPy.

y.values : Convertit la variable cible (la colonne 'HeartDisease') en un tableau NumPy.

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

Description générée automatiquement

label = LabelEncoder() : Crée un objet LabelEncoder qui permet de convertir les labels catégoriels en entiers.



La ligne de code suivante utilise la fonction **train\_test\_split** de **scikitlearn** pour diviser les données en ensembles d'entraînement et de test

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

Description générée automatiquement

-L'ensemble d'entraînement (X\_train et y\_train) contient 734 observations et 11 caractéristiques pour X\_train, et 734 cibles pour y\_train.

-L'ensemble de test (X\_test et y\_test) contient 184 observations et 11 caractéristiques pour X\_test, et 184 cibles pour y\_test.

Cela respecte le ratio de 80 % pour l'entraînement et 20 % pour les tests, ce qui est couramment utilisé pour diviser les données dans les projets de machine learning.

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

Description générée automatiquement

-StandardScaler est un outil de prétraitement qui normalise les données en leur donnant une distribution avec une moyenne de 0 et un écart-type de 1.

**-**fit\_transform() est utilisé pour ajuster le **scaler** sur les données d'entraînement (X\_train) et appliquer la transformation. Cela signifie qu'il calcule la moyenne et l'écart-type de chaque caractéristique dans X\_train, puis normalise les données en soustrayant la moyenne et en divisant par l'écart-type pour chaque caractéristique.

-X\_train\_sc contient maintenant les données d'entraînement mises à l'échelle.

-Cela aussi est utilisé pour appliquer la même transformation (moyenne et écart-type calculés à partir de X\_train) aux données de test (X\_test)

- L’image ci-dessous représente les données dans l’état

normal et l’autre standardisé.

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

Description générée automatiquement

**Application du K-Nearest Neighbors (KNN) pour la classification :**Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

-n\_neighbors=5 : Cette ligne définit le nombre de voisins à considérer pour la classification (ici, 5 voisins). Lorsqu'une nouvelle observation doit être classée, l'algorithme regarde les 5 voisins les plus proches dans les données d'entraînement pour déterminer la classe à prédire.

**-fit()** : Cette méthode permet d'entraîner le modèle sur les données d'entraînement. Ici, **X\_train\_sc** est la matrice des caractéristiques mises à l'échelle de l'ensemble d'entraînement, et **y\_train** est le vecteur des étiquettes (cibles) correspondantes.

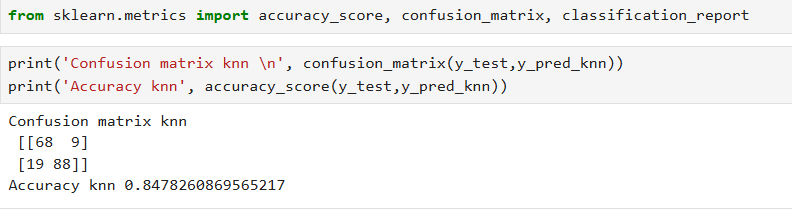


y\_pred\_knn contient les prédictions du modèle pour l'ensemble de test. C'est un tableau de valeurs prédites (par exemple, des classes comme 0 ou 1 pour la classification binaire, ou des valeurs multiples pour les problèmes de classification multiclasse).

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

Description générée automatiquement

On peut observer que la plupart des prédictions correspondent aux vraies valeurs, ce qui indique que le modèle KNN semble avoir bien performé sur ces 10 premières observations.



True Negatives (TN) = 68 : Le modèle a correctement prédit 68 fois la classe 0.

False Positives (FP) = 9 : Le modèle a prédit à tort 9 fois la classe 1 alors que la vraie classe était 0.

False Negatives (FN) = 19 : Le modèle a prédit à tort 19 fois la classe 0 alors que la vraie classe était 1.

True Positives (TP) = 88 : Le modèle a correctement prédit 88 fois la classe 1.

-La précision est de **84,78%**, ce qui signifie que **84,78%** des prédictions faites par le modèle sont correctes.

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

Description générée automatiquement

La classe 1 semble être mieux prédite en termes de précision (0.91), mais le modèle a un rappel légèrement plus faible pour cette classe (0.82) par rapport à la classe 0 (rappel de 0.88).

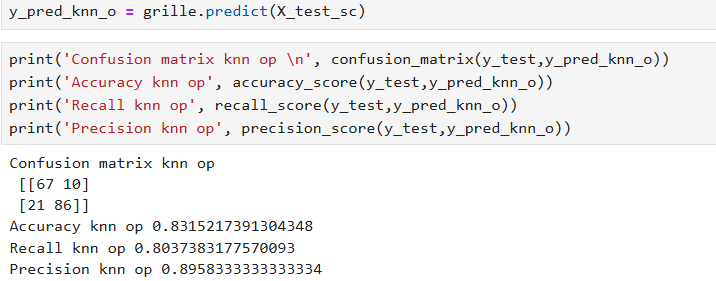
**GridSearchCV:**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Le GridSearchCV a pour optimiser le paramètre n\_neighbors du modèle K-Nearest Neighbors (KNN) en utilisant une validation croisée.

Dans notre cas , la meilleure valeur pour **n\_neighbors** est **5**, et le score d'accuracy optimal pendant la validation croisée était de **85%**.



Les résultats obtenus montrent les performances du modèle K-Nearest Neighbors optimisé (après l'utilisation de GridSearchCV ou d'un autre ajustement des paramètres).

Le modèle optimisé de **K-Nearest Neighbors** semble offrir un bon compromis entre **précision** et **rappel.**

**DecisionTreeClassifier :**

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

-L'**accuracy** est de **48.37%**, ce qui signifie que le modèle a correctement classé **48.37%** des exemples dans l'ensemble de test. Ce score est relativement faible, ce qui indique que le modèle a du mal à bien prédire la classe des exemples.

-Le **rappel** est de **20.56%** pour la classe **1**, ce qui signifie que le modèle n'a détecté que **20.56%** des instances réelles de la classe **1**. C'est un rappel relativement faible, ce qui suggère que le modèle manque une grande partie des instances positives.

-La **précision** est de **68.75%** pour la classe **1**, ce qui signifie que parmi toutes les prédictions de la classe **1**, **68.75%** étaient correctes. C'est une précision relativement raisonnable, mais elle ne compense pas le faible rappel.

Le modèle **Decision Tree** semble avoir une faible **accuracy** et un très faible **recall.**

**SVM :**

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

Description générée automatiquement

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

le modèle SVM a montré de bonnes performances globales, avec des métriques solides pour la précision et le rappel.