MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

RAPPORT DE PROJET

Option : **Ingénierie des Systèmes d'Information**

THEME :

Une approche basée sur les techniques de machine learning pour la prédiction du l'hépatite

Etudiants : « **Medjahed Youcef** »

« **Hachelaf Abdelmalek Ahmed** »

Encadrante : « Djahafi Fatiha »

Année Universitaire 2022-2023

Résumé

L'hépatite est une maladie grave répandue dans le monde, qui affecte la fonction hépatique et provoque une inflammation. Le virus est la principale cause de l'hépatite. Un diagnostic précoce et un traitement approprié peuvent guérir la maladie. Dans le cadre de cette étude, nous envisageons de mener une recherche scientifique et d'examiner différentes techniques de machine learning pour la détection de la maladie de l'hépatite à partir d'un ensemble de données sélectionné. Le but de ce projet est de proposer une approche pour la détection précise et précoce de l'hépatite afin d'améliorer l'efficacité des traitements et la qualité de vie des patients atteints d'hépatite.

Mots-clés: Hépatite, Foie, Maladie de foie, Détection Précoce, Intelligence artificielle, Apprentissage automatique, Classification.

Abstract

Hepatitis is a serious disease that is widespread around the world, affecting liver function and causing inflammation. The virus is the main cause of hepatitis. An early diagnosis and appropriate treatment can cure the disease. In this study, we plan to conduct scientific research and to examine various machine learning techniques to detect the hepatitis disease from a selected dataset. The goal of this project is to propose an approach for the accurate and early detection of hepatitis in order to improve treatment effectiveness and the quality of life for patients with hepatitis.

Keywords: Hepatitis, Liver, Liver disease, Early detection, Artificial intelligence, Machine learning, Classification.

Dédicaces

Nous dédions ce travail à :

Nos parents

Nos frangins

Notre encadreur Mme. Djahafi Fatiha

Nos amis et collègues

Remerciements

*Nous tenons à exprimer notre gratitude à **Mme DJAHAFI Fatiha** pour son soutien et ses conseils. Nous sommes reconnaissants de ses suivis, ses recommandations, ses efforts, et ses encouragements.*

Nous tenons à exprimer nos remerciements à la présidente et le jury pour avoir accepter de juger ce travail.

Nous remercions également nos parents, nos frangins, ainsi que nos amis et nos collègues.

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviation

Abréviation	Expression Complète	Page
SARSA	State-Action-Reward-State-Action	12
KNN	K nearest neighbors	18
MLP	Multi-Layer Perceptron	25
SVM	Support Vector Machines	26
RBF	Radial Basis Function	27
ILPD	Indian Liver Patient Data set	29
XGBoost	Extreme Gradient Boosting	29
LightGBM	Light Gradient Boosting Machine	29
AUC	Area Under the Curve	29
ROC	Receiver Operating Characteristic	29
SMO	Sequential Minimal Optimization	29
IBk	Instance Based with k nearest neighbor	29
AdaBoost	Adaptatif Boost	30
ILDPS	Intelligent Liver Disease Prediction Software	34

Table des matières

Introduction Générale

L'hépatite est une maladie inflammatoire du foie qui peut causer des dommages permanents et conduire à des complications graves telles que la cirrhose ou le cancer du foie. La détection précoce de la maladie est essentielle pour un traitement efficace et pour prévenir les complications graves. C'est pourquoi de nombreuses recherches ont été menées pour améliorer les méthodes de dépistage et de traitement de l'hépatite.

Avec le développement des technologies, l'apprentissage automatique qui est une branche de l'intelligence artificielle permet de résoudre plusieurs problèmes tels que la détection de maladies à partir de données médicales. Les avancées technologiques dans le domaine de l'apprentissage automatique ont permis de développer la détection de l'hépatite en utilisant différentes techniques.

Ce projet s'inscrit dans le cadre de proposer une approche pour la détection précoce de l'hépatite, en utilisant des techniques d'apprentissage automatique afin d'améliorer l'efficacité des traitements et la qualité de vie des patients. Le projet se compose de trois chapitres.

Dans le premier chapitre, nous allons présenter en détail la maladie de l'hépatite, en mettant en évidence les causes, les symptômes et les conséquences de cette pathologie.

Dans le deuxième chapitre, nous détaillerons les concepts de base de l'apprentissage automatique, et nous analyserons des travaux de recherche dans le domaine de la détection de l'hépatite en utilisant des techniques d'apprentissage automatique.

Dans le troisième chapitre, nous détaillerons la méthodologie utilisée pour développer l'approche basée sur l'apprentissage automatique proposée pour la détection précoce de la maladie de l'hépatite.

Chapitre 1

La maladie de l’hépatite

1.1 Introduction

L'hépatite est une maladie inflammatoire du foie qui peut avoir des conséquences graves sur la santé des individus. Elle peut être causée par des virus, des substances toxiques ou des maladies auto-immunes. Les symptômes de l'hépatite peuvent varier d'une personne à l'autre, allant de légers à sévères. Si elle n'est pas détectée et traitée à temps, elle peut conduire à des complications telles que la cirrhose ou le cancer du foie. C'est pourquoi une détection précoce de la maladie est essentielle pour un traitement efficace [1].

Dans ce chapitre, nous allons examiner de plus près les différentes causes de l'hépatite, les symptômes qui y sont associés, ainsi que les complications possibles. Nous verrons également comment la maladie est possiblement traitée.

1.2 L’hépatite et ses types

1.2.1 Définition de l’hépatite

L'hépatite est une inflammation du foie, un organe vital du corps humain qui joue un rôle essentiel dans le métabolisme et dans de nombreuses autres fonctions du corps. Cette inflammation peut être causée par une infection virale, une exposition à des toxines ou à des substances chimiques. Elle se manifeste par une altération de la fonction hépatique, une augmentation des enzymes hépatiques dans le sang et des symptômes tels que la fatigue, la jaunisse, et la fièvre [2].

Du point de vue scientifique, l'hépatite est caractérisée par une inflammation du tissu hépatique, qui se produit en réponse à une infection ou à une agression du foie. Cette inflammation peut entraîner une fibrose, ce qui peut à son tour conduire à une cirrhose ou à un cancer du foie [2].

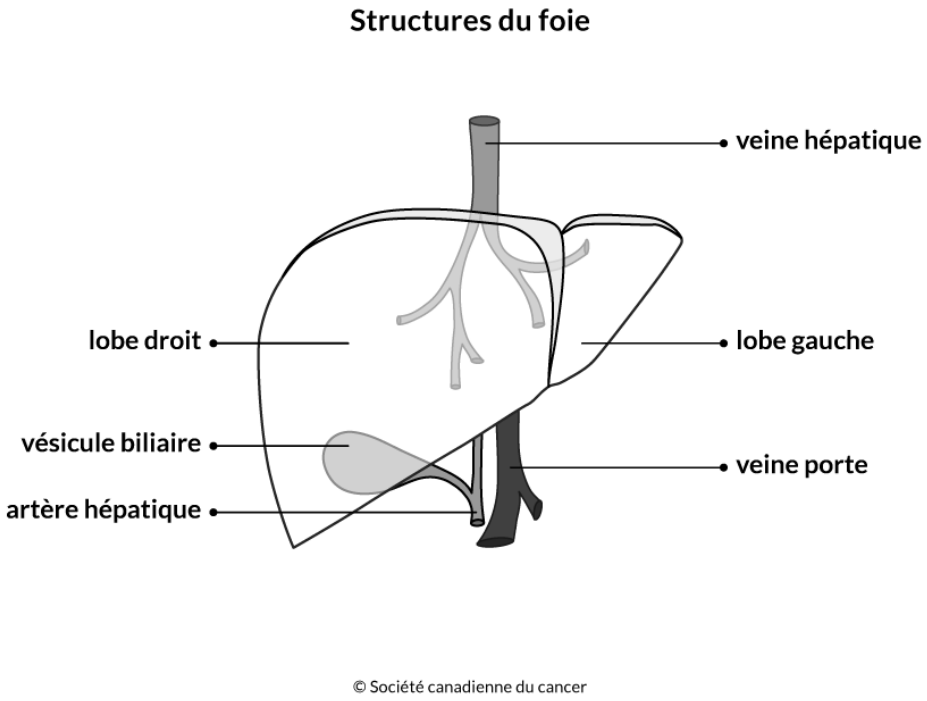


Figure 1 – Schéma des structures du foie [3]

1.2.2 Types de l'hépatite

Nous pouvons distinguer plusieurs types de l'hépatite, selon leur cause. Parmi ces types nous avons :

- L'hépatite A (causée par le virus d'hépatite A).
- L'hépatite B (causée par le virus d'hépatite B).
- L'hépatite C (causée par le virus d'hépatite C).
- L'hépatite D (causée par le virus d'hépatite D).
- L'hépatite E (causée par le virus d'hépatite E).
- L'hépatite G (causée par le virus d'hépatite G).

1.3 Symptômes et facteur de risque de l’hépatite

1.3.1 Les symptômes

Les personnes atteintes de l’hépatite peuvent avoir des symptômes comme ils peuvent être asymptotiques. Les symptômes de la maladie de l’hépatite sont nombreux et diffèrent d’un type d’hépatite à un autre [2]. Parmi les symptômes généraux nous pouvons citer :

- Fièvre.
- Fatigue.
- Perte d'appétit.
- Perte de poids.
- Maux de tête.
- Jaunisse (jaunissement de la peau et des yeux).

1.3.2 Facteurs de risque

L'hépatite est une maladie peut être causée par plusieurs facteurs. Les causes de l'hépatite et les facteurs de risque associés sont notamment des virus, des substances toxiques et des maladies auto-immunes [4].

Le tableau ci-dessous résume ces différents facteurs de risque pour l'hépatite selon leur groupe.

Table 1 – Facteurs de risque de la maladie de l'hépatite [4]

Groupe	Facteurs de risque
Facteurs virales	Exposition au sang infecté, voyage dans des zones à haut risque.
Facteurs auto-immunitaires	Antécédents familiaux de maladies auto-immunes.
Facteurs médicamenteuses	Utilisation de médicaments connus pour causer des dommages ou de la toxicité au foie, surdosage de médicaments.
Facteurs toxiques	Exposition à des produits chimiques ou des toxines connus pour causer des dommages au foie, comme les plantes toxiques.

1.4 Les complications de l'hépatite

Les séquelles et les complications de l'hépatite peuvent varier en fonction de la cause et de la gravité de l'inflammation du foie. Voici quelques exemples de séquelles possibles :

- **Fibrose hépatique :** La fibrose hépatique est une inflammation prolongée du foie pouvant causer une cicatrisation du tissu hépatique. Cette fibrose peut entraîner une hypertension portale et une insuffisance hépatique.
- **Cirrhose :** La cirrhose est une maladie chronique du foie qui est caractérisée par une fibrose étendue et une perturbation de l'architecture normale du foie. Les personnes atteintes de cirrhose peuvent présenter un cancer du foie.
- **Cancer du foie :** Le cancer du foie est une maladie qui se forme à partir des cellules du foie. La cirrhose chronique est un facteur de risque important pour le cancer du foie.

Il convient de souligner que toutes les personnes souffrant d'hépatite ne présentent pas nécessairement des séquelles à long terme. En effet, un traitement précoce et efficace de l'hépatite peut contribuer à prévenir ou à atténuer le risque de complications graves.

1.5 Les traitements médicaux de l'hépatite

Les traitements médicaux de l'hépatite varient en fonction de la cause et de la gravité de la maladie. Il existe plusieurs traitements de la maladie de l'hépatite, ces traitements peuvent être divisés en plusieurs catégories, qui dépendent de la cause de la maladie. Chacune de ces catégories est aperçu comme :

- **Traitements antiviraux :** Les traitements antiviraux sont utilisés pour traiter l'hépatite virale, en particulier les hépatites B et C. Ces médicaments sont conçus pour cibler spécifiquement le virus responsable de l'infection et aider à réduire la charge virale dans le corps [5].

- **Traitements immunosuppresseurs :** Les traitements immunosuppresseurs sont utilisés pour traiter l'hépatite auto-immune, une maladie dans laquelle le système immunitaire attaque le foie. Les médicaments immunosuppresseurs aident à réduire l'activité du système immunitaire [5].

- **Traitements de support :** Les traitements de support peuvent être utilisés pour aider à gérer les symptômes de l'hépatite et à protéger le foie. Ces traitements peuvent inclure des changements de mode de vie, tels qu'une alimentation équilibrée et une activité physique régulière. Les suppléments de vitamines et de minéraux peuvent également être recommandés pour aider à maintenir la santé du foie [5].

- **Vaccins :** Des vaccins sont disponibles pour prévenir l'infection par les virus de l'hépatite A et B [5]. Un vaccin contre le virus de l'hépatite E est également disponible dans certains pays.

1.6 Statistiques sur l'hépatite

L'hépatite est une maladie grave touchant des millions de personnes dans le monde. Chaque année, 1,5 million de personnes sont infectées par l'hépatite A et 71 millions de

personnes sont affectées par l'hépatite C. En Afrique, environ 18 millions de personnes sont atteintes d'hépatite D, tandis que l'hépatite E a infecté environ 20 millions de personnes dans le monde. Ce qui nécessite de nouvelles méthodes pour prédire l'hépatite avec précision et fiabilité et aider à améliorer la surveillance et la prévention de cette maladie.

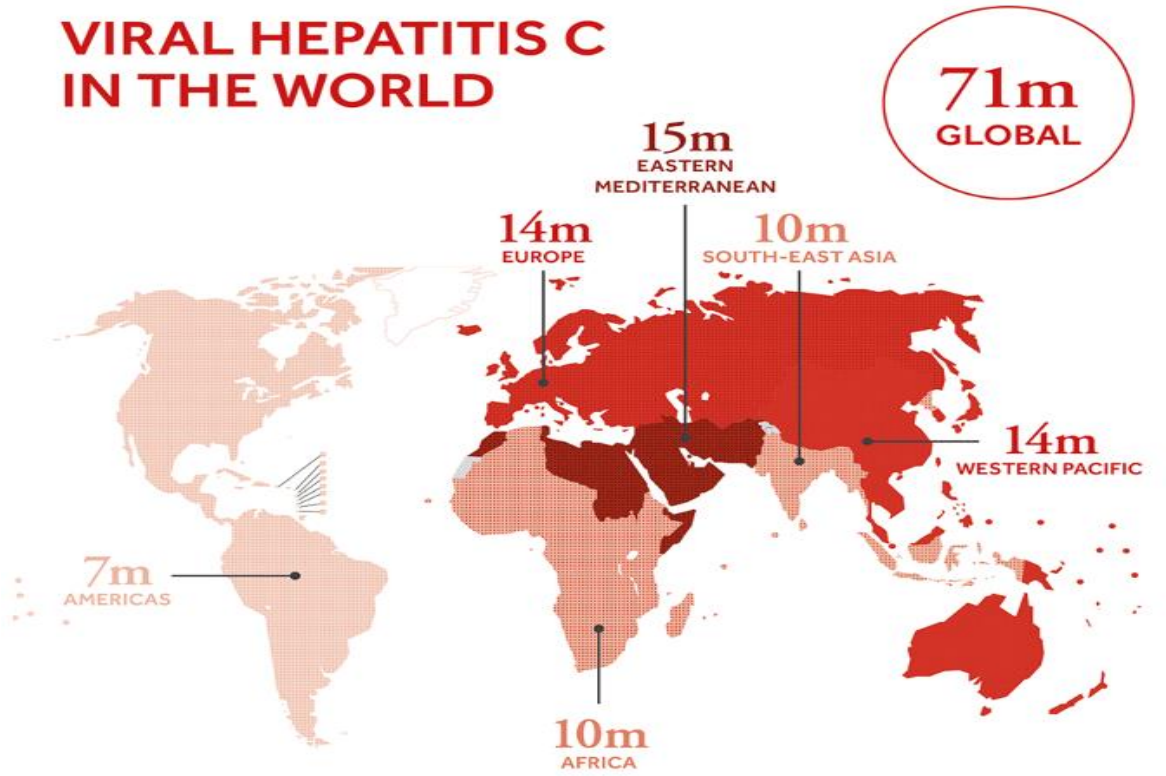


Figure 2 – Statistiques du virus de l'hépatite C dans le monde en 2017 [6]

1.7 Conclusion

Dans ce chapitre nous avons exploré des détails sur la maladie de l'hépatite, en présentant les causes, les symptômes et les complications associées, ainsi que les traitements médicaux disponibles, les statistiques de cette maladie. Toutefois, pour garantir une prise en charge optimale de cette maladie, il est primordial de disposer d'une approche diagnostique fiable et précise, qui permette aux professionnels de la santé de détecter l'hépatite le plus tôt possible afin de la traiter efficacement.

Chapitre 2

Apprentissage automatique

2.1 Introduction

L'apprentissage automatique, également connu sous le nom de machine learning, est un champ d'étude de l'intelligence artificielle qui s'intéresse à la conception, au développement, à l'analyse et à l'implémentation d'algorithmes et de modèles mathématiques qui permettent aux ordinateurs d'apprendre à partir des données.

Dans ce chapitre, nous allons présenter les concepts de base de l'apprentissage automatique. Nous commencerons par décrire les différents types d'apprentissage automatique, à savoir l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Nous verrons également le concept de la classification et des algorithmes importants de l'apprentissage automatique. A la fin, nous allons analyser des travaux de recherche dans le domaine de la détection de l'hépatite.

2.2 Types d'apprentissage automatique

2.2.1 Apprentissage supervisé

L'apprentissage supervisé permet à un modèle d'apprentissage automatique d'apprendre à partir d'exemples préétiquetés afin de prédire les étiquettes des nouveaux exemples en utilisant les caractéristiques générales découvertes à partir d'une distribution d'exemples d'entraînement. L'objectif de l'apprentissage supervisé est de trouver une fonction générale qui représente la relation entre les entrées et les sorties, afin de l'utiliser pour la classification des nouvelles données.

2.2.2 Apprentissage non supervisée

L'apprentissage non supervisé est une technique d'apprentissage automatique qui vise à explorer les relations intrinsèques entre les données non étiquetées, et faire un regroupement ou une réduction de la dimensionnalité des données.

Le regroupement consiste à identifier les ensembles de données semblables parmi les exemples, lesquels sont ensuite classés en fonction de critères de ressemblance, tels que la proximité. La réduction de dimension, quant à elle, a pour but de simplifier les données tout en conservant le maximum d'informations, par exemple en fusionnant plusieurs caractéristiques en un seul caractère.

2.2.3 Apprentissage par renforcement

L'apprentissage par renforcement est une méthode d'apprentissage automatique qui permet à un agent d'interagir avec son environnement pour prendre des décisions et maximiser une récompense cumulée au fil du temps. Toutefois, cette méthode implique la prise en compte d'actions à éviter, en plus des actions à favoriser, afin d'améliorer les performances de l'agent. L'agent doit déterminer la meilleure action à prendre en fonction de l'état actuel pour maximiser sa récompense nette, tout en prenant en compte les transitions qui entraînent un nouvel état et une récompense associée. Cette méthode est particulièrement utile dans des situations où les données étiquetées ne sont pas disponibles ou où il est difficile de modéliser la relation entre les entrées et les sorties [7].

L'apprentissage par renforcement comprend plusieurs sous-catégories, telles que les méthodes basées sur la valeur, les méthodes basées sur la politique et les méthodes basées sur la valeur et la politique combinées. Les algorithmes couramment utilisés incluent l'algorithme de Q-learning, l'algorithme SARSA, et l'algorithme de Monte Carlo [7].

2.3 La classification

La classification est un domaine de l'apprentissage automatique qui tente de trouver la catégorie ou le groupe d'un exemple de données. La classification permet de regrouper les exemples en utilisant leurs caractéristiques et propriétés.

Il existe trois types de classification qui sont : la classification supervisée, la classification non supervisée, et la classification semi supervisée.

2.3.1 Classification supervisée

La classification supervisée est une technique d'analyse et de traitement de données qui permet d'identifier les classes auxquelles appartiennent des objets en se basant sur leurs variables descriptives. Cette méthode nécessite un ensemble d'entraînement constitué d'exemples dont les classes sont connues, sur lequel un modèle est entraîné pour pouvoir généraliser et classer correctement de nouveaux exemples. La formalisation mathématique de la classification supervisée implique de considérer un ensemble d'exemples de données étiquetées, où chaque instance est représentée par deux propriétés, notées x" et x', ainsi qu'une étiquette y qui prend par exemple, une valeur de +1 ou de -1 en fonction de la catégorie à laquelle l'instance appartient. La classification supervisée vise à trouver une fonction $f(x)$ qui peut catégoriser chaque instance en classe positive C_p ou classe négative C_N, telle que :

$$y = +1 \text{ si } x \in C_p \text{ et } y = -1 \text{ si } x \in C_N \tag{2.1}$$

Plusieurs mesures de performance sont couramment utilisées pour évaluer un modèle de classification supervisée. Parmi ces mesures nous trouverons :

- **Exactitude :** La mesure de l'exactitude est la proportion d'instances correctement classées sur l'ensemble des données de test. Elle est définie comme :

$$accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{2.2}$$

Où TP est le nombre d'instances positives correctement classées, TN est le nombre d'instances négatives correctement classées, FP est le nombre d'instances négatives

classées à tort comme positives et FN est le nombre d'instances positives classées à tort comme négatives.

●**Précision :** La précision mesure la proportion d'instances positives correctement classées parmi toutes les instances positives prédites. Elle est définie comme :

$$precision = \frac{(TP)}{(TP+FP)} \quad (2.3)$$

●**Rappel :** Il s'agit d'une mesure de la proportion d'instances positives correctement classées parmi toutes les instances positives réelles. Il est défini comme :

$$recall = \frac{TP}{(TP+FN)} \quad (2.4)$$

●**F-mesure :** La F-mesure est une mesure qui combine la précision et le rappel en une seule valeur. Elle est définie comme :

$$Fscore = \frac{2 \times precision \times recall}{(precision + recall)} \quad (2.5)$$

2.3.2 Classification non supervisée

La classification non supervisée, est une méthode utilisée pour organiser les données en utilisant les structures sous-jacentes sans avoir de connaissances préalables sur les données traitées. Son principal objectif est de regrouper des exemples similaires en un nombre limité de groupes distincts, sans utiliser d'étiquettes préalables [8]. Les algorithmes de classification non supervisée visent à minimiser la similarité intra-classe tout en maximisant la similarité inter-classe. Cela permet de créer des groupes où les éléments à l'intérieur d'un même groupe sont les plus similaires possible, tandis que les groupes eux-mêmes sont séparés les uns des autres.

Il existe plusieurs types d'algorithmes de classification, mais les méthodes de classification hiérarchique et de partitionnement sont les plus couramment utilisées. La classification hiérarchique peut être ascendante ou descendante, sans qu'un nombre de classes ne soit déterminé à l'avance. Le partitionnement, quant à lui, permet une classification non hiérarchique en un nombre fixe de classes.

La classification non supervisée cherche à regrouper les exemples de données en *K* clusters, *K* étant un nombre fixé à l'avance, de sorte que les exemples de données dans le

même cluster soient similaires les uns aux autres. Chaque cluster peut être décrit par son centre de gravité u_k, qui est la moyenne des exemples de données dans ce cluster :

$$u_k = \frac{1}{|c_k|} + \sum_{x_i \in c_k}^{n} x_i \quad (2.6)$$

Il existe des mesures pour évaluer la performance d'un modèle de classification non supervisée, parmi les plus courantes nous trouvons l'indice de Davies-Bouldin, Cet indice représente la moyenne des distances entre les centres des clusters divisée par une mesure de la dispersion interne des clusters. L'indice de Davies-Bouldin est défini par la formule suivante :

$$DB = \frac{1}{K}\sum_{k=1, k<>k'}^{K} \frac{s_k + s_{k'}}{d(u_k, u_{k'})} \quad (2.7)$$

Avec :

$$s_k = \frac{1}{|c_k|} + \sum_{x_i \in c_k}^{n} d(x_i, u_k) \quad (2.8)$$

2.3.3 Classification semi supervisée

Lorsque l'on souhaite classifier des données, le processus de labellisation peut s'avérer fastidieux, complexe, coûteux voire même impossible dans certains cas. Cela peut rendre difficile l'obtention d'un classifieur performant et généralisable avec peu de données labellisées. Cependant, il arrive souvent que de nombreuses données non labellisées soient disponibles, telles que des documents textuels, des pages web ou encore des séquences protéiques. Ces données peuvent fournir des informations sur la distribution des exemples qui peuvent être exploitées pour améliorer la phase d'apprentissage. L'apprentissage semi-supervisé consiste ainsi à utiliser des données non labellisées en plus des données labellisées pour influencer l'algorithme d'apprentissage et améliorer ses performances [9].

2.4 Algorithmes d’apprentissage automatique

2.4.1 Arbres de décision

Les arbres de décision sont une méthode d'apprentissage supervisée efficace pour résoudre des tâches de classification ou de régression. L'objectif est de créer un modèle prédictif à partir de données d'entraînement qui peut ensuite être utilisé pour prédire les labels ou les valeurs cibles des exemples non vus. Le processus de construction de l'arbre consiste à diviser récursivement les données en sous-ensembles de plus en plus petits, en utilisant les caractéristiques les plus importantes des données. Le processus se répète jusqu'à ce qu'un critère d'arrêt soit atteint, par exemple, lorsque toutes les données dans un sous-ensemble appartiennent à la même classe ou lorsque la profondeur maximale de l'arbre est atteinte [10]. Cette structure en arbre a des nœuds internes qui représentent les conditions de test basées sur les caractéristiques, les arêtes qui représentent les résultats des tests et les feuilles qui représentent les prédictions finales.

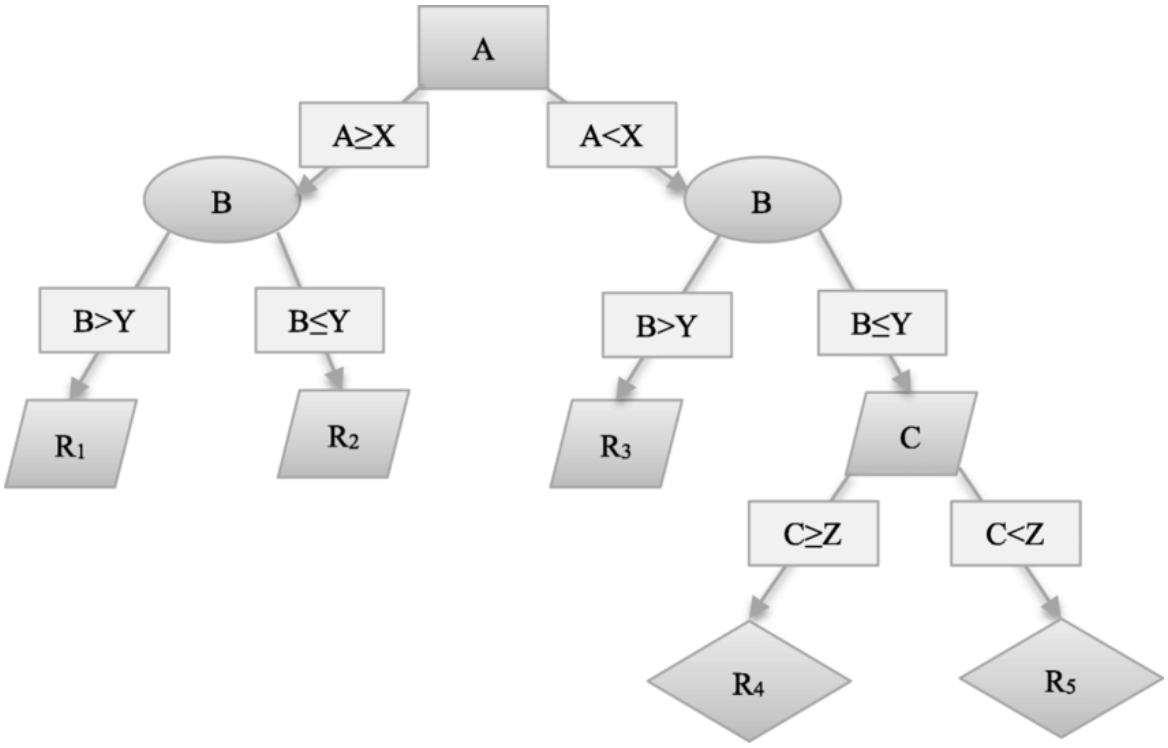


Figure 3 – Une architecture d'arbre de décision [11]

Algorithme 1 Construction d'arbre de décision binaire

Entrées :
    Matrice de données d'apprentissage *X_train*
    Vecteur de valeurs cibles associées aux données d'apprentissage *y_train*
    Critère à utiliser pour mesurer la qualité de la division *crit*
    Profondeur maximale de l'arbre *max_depth*
    Nombre minimum d'échantillons pour diviser un nœud *min_div*
    Nombre minimum d'échantillons dans une feuille de l'arbre *min_leaf*
Sorties :
    Arbre de décision construite *N*
Fonction train(*X, y, crit, depth, max_depth, min_div, min_leaf*)
    Si (toutes les valeurs cibles sont les mêmes) ou (*depth* >= *max_depth*)
    ou (nombre d'échantillons dans *X* < *min_leaf*) ou (nombre
    d'échantillons dans *X* < *min_div*) **Faire**
        Retourner un nœud feuille *N* avec l'étiquette de la valeur cible majoritaire
    Sinon
        Sélectionner la caractéristique *col* qui maximise la mesure de qualité
        Créer un nœud de décision *N* pour *col*
        Diviser *X*, *y* en sous-ensembles *X_left*, *X_right*, *y_left*, *y_right* en utilisant *col*
        Ajouter train(*X_left*, *y_left*, *crit*, *depth* + 1, *max_depth*,
        min_div, *min_leaf*) comme un enfant gauche de *N*
        Ajouter train(*X_right*, *y_right*, *crit*, *depth* + 1, *max_depth*,
        min_div, *min_leaf*) comme un enfant droit de *N*
        Retourner *N*
    Fin Si
Fin Fonction

Les mesures telles que l'impureté de Gini ou le gain d'information sont utilisées pour trouver la caractéristique et le point de division qui entraînent la plus grande réduction de l'impureté, ce qui permet de choisir la meilleure division pour améliorer la qualité des prédictions. La formule suivante permet de calculer l'impureté de Gini.

$$GINI(p) = 1 - \sum_{i=1}^{n} {p_i}^2 \quad (2.9)$$

Où p_i est la proportion d'échantillons d'une classe particulière parmi tous les échantillons du nœud.

Pour prédire la valeur cible d'un nouvel exemple, l'algorithme parcourt l'arbre à partir de la racine et suit les branches correspondant aux résultats des tests jusqu'à atteindre un nœud feuille, où la prédiction est donnée par l'étiquette de classe ou la valeur numérique attribuée au nœud feuille.

Les arbres de décision sont largement utilisés dans de nombreux domaines. Ils peuvent être utilisés pour la prédiction du risque, le diagnostic de maladies, l'analyse de la discrimination, l'identification d'objets dans des images, la compréhension des comportements des consommateurs et l'amélioration de la précision des prédictions. Les arbres de décision sont connus pour leur simplicité, leur rapidité et leur interprétabilité, ce qui les rend populaires dans diverses applications où la transparence de la méthode est importante. Malgré leurs avantages les arbres de décision présentent également certains désavantages comme la tendance au surapprentissage et les problèmes de gestion des attributs continus.

2.4.2 K plus proches voisins

L'algorithme des k plus proches voisins ou KNN (K Nearest Neighbors) est un modèle simple et facile à implémenter pour l'apprentissage automatique supervisé. Il peut être utilisé pour la classification et la régression. L'algorithme fonctionne en trouvant les K points de données les plus proches d'un nouvel élément et en utilisant la classe majoritaire ou la valeur moyenne des K voisins les plus proches pour faire une prédiction. Lors de la classification d'un nouvel élément, une mesure de distance ou de similitude est utilisée pour le comparer aux autres éléments. Les K voisins les plus proches de cet élément sont identifiés et la classe majoritaire de ces voisins ou la moyenne de leur valeurs cible sera l'étiquette du nouvel élément. Le choix correct de plusieurs paramètres, tels que le nombre de voisins et la distance utilisée pour mesurer la similitude entre les éléments, est crucial pour la performance de la méthode [12].

Il existe plusieurs fonctions de distance, parmi lesquelles les plus couramment utilisées sont :

- **Distance de Manhattan :** La distance de Manhattan est la somme des différences absolues. La distance de Manhattan est définie comme suit :

$$d(x_i, x_i') = \sum_{i=1}^{n} |x_i - x_i'| \quad (2.10)$$

Où d est la distance de Manhattan entre les vecteurs x_i^t et x_i^t.

●**Distance Euclidienne :** La distance euclidienne est la distance géométrique standard entre les points dans un espace Euclidien. Cette mesure est exprimée comme suit :

$$d(x_i, x_i') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2} \quad (2.11)$$

●**Distance Minkowski (p-distance) :** La distance de Minkowski (p-distance) est une généralisation des distances de Manhattan et Euclidienne. Elle est définie par :

$$d(x_i, x_i') = \sqrt[p]{\sum_{i=1}^{n} |x_i - x_i'|^p} \quad (2.12)$$

Avec : $p \geq 1$

Le paramètre p représente la norme utilisée pour mesurer la distance entre les vecteurs. En particulier, lorsque $p = 2$, la distance de Minkowski est équivalente à la distance Euclidienne, tandis que lorsque $p = 1$, elle est équivalente à la distance de Manhattan.

●**Distance de Tchebychev :** La distance de Tchebychev est la distance maximale entre les points dans un espace. Cette distance est décrite comme suit :

$$d(x_i, x_i') = max|x_i - x_i'| \quad (2.13)$$

●**Distance de Cosinus :** Cette distance utilise la similitude de Cosinus qui mesure la similarité des angles entre les vecteurs. La distance de Cosinus est définie comme suit :

$$d(x_i, x_i') = \frac{\sum_{i=1}^{n} x_i \times x_i'}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \times \sqrt{\sum_{i=1}^{n} (x_i')^2}} \quad (2.14)$$

Algorithme 2 K plus proches voisins

```
Entrées
    Matrice de données d'apprentissage X_train
    Vecteur de valeurs cibles associées aux données d'apprentissage y_train
    Nombre de voisins k
    Matrice de données de test X_test
    Fonction de distance d
Sorties
    Vecteur des classes prédites y_pred
Fonction predict(X_test, k, d)
    y_pred <- liste vide
    Pour chaque indice i dans X_test Faire
        dist <- liste vide
        Pour chaque indice j dans X_train Faire
            Ajouter d(x_test[i], X_train[j]) à dist
        Fin Pour chaque
        idx <- trier les distances et retourner les indices des k plus petites dist
        v <- valeur la plus fréquente dans les valeurs de y_train avec les index idx
        Ajouter v à y_pred
    Fin Pour chaque
    Retourner y_pred
Fin Fonction
```

Les algorithmes des k plus proches voisins (KNN) sont largement utilisés dans la classification, la reconnaissance d'objets, la prédiction de la qualité des données, la segmentation de clients. Les k plus proches voisins comportent des avantages tels que la robustesse face aux données bruitées, et la simplicité de les comprendre et de les mettre en œuvre. Également il existe certains inconvénients à prendre en compte comme le choix de la valeur de K et l'impact de l'échelle des attributs et des données aberrantes sur les résultats de l'algorithme.

2.4.3 Naïf Bayes

Le classificateur Naïf Bayes ou Naive Bayes est un algorithme d'apprentissage automatique supervisé, qui est utilisé pour les tâches de classification, comme la classification de texte. Il fait également partie d'une famille d'algorithmes d'apprentissage génératif, ce qui signifie qu'il cherche à modéliser la distribution des entrées d'une classe ou d'une catégorie donnée. Contrairement aux classificateurs discriminatifs, comme la

régression logistique et les machines à vecteurs de support, il n'apprend pas quelles caractéristiques sont les plus importantes pour différencier les classes [13].

Le modèle probabiliste pour un classifieur est le modèle conditionnel.

$$\rho(\mathsf{C}|F_1, \dots, F_n) \tag{2.15}$$

Où C est une variable de classe dépendante dont les instances ou classes sont peu nombreuses, conditionnée par plusieurs variables caractéristiques F_1,..., F_n.

Lorsque le nombre de caractéristiques n est grand, ou lorsque ces caractéristiques peuvent prendre un grand nombre de valeurs, baser ce modèle sur des tableaux de probabilités devient impossible. Par conséquent, nous le dérivons pour qu'il soit plus facilement soluble.

Algorithme 3 Entrainement de Naïf Bayes

```
Entrées
    Matrice de données d'apprentissage X_train
    Vecteur de valeurs cibles associées aux données d'apprentissage y_train
Sorties
    Liste des probabilités à priori P_y_priori
    Liste des probabilités conditionnelles P_x_y
Fonction train(X_train, y_train) Faire
    P_y_priori <- Liste vide
    P_x_y <- Liste vide
    Pour chaque étiquette de la valeur cible distincte y
        Ajouter la probabilité à priori de y à P_y_priori
        Pour chaque attribut i Faire
            Ajouter la probabilité conditionnelle de i étant donné y à P_x_y
        Fin Pour chaque
    Fin Pour chaque
    Retourner P_y_priori, P_x_y
Fin Fonction
```

L'algorithme Naive Bayes est un modèle d'apprentissage automatique populaire utilisé dans de nombreux domaines. Il est largement utilisé pour la classification de texte, le filtrage de contenu, la recommandation de produits, la détection de spam, l'analyse de

sentiments, la détection de fraudes et bien d'autres applications, L'un des principaux avantages de Naive Bayes est sa simplicité. Il est facile à comprendre et à mettre en œuvre, même pour les débutants en apprentissage automatique.

De plus, il est rapide à entraîner et à prédire, ce qui le rend adapté aux ensembles de données de grande taille.

Bien que l'algorithme Naive Bayes présente plusieurs avantages, il présente également quelques inconvénients, sensibilité aux données d'entraînement, incapacité à gérer les données manquantes, difficulté à modéliser les relations complexes, L'hypothèse d'indépendance conditionnelle.

2.4.4 Foret Aléatoire

La Foret Aléatoire ou le Random Forest est l'un des algorithmes les plus populaires et les plus couramment utilisés par les scientifiques des données. La forêt aléatoire est un algorithme d'apprentissage automatique supervisé largement utilisé dans les problèmes de classification et de régression. Il construit des arbres de décision sur différents échantillons et prend leur vote majoritaire pour la classification et la moyenne en cas de régression [14].

L'algorithme d'apprentissage pour les forêts aléatoires applique la technique générale d'agrégation guidée ou d'ensachage aux apprenants d'arbres. Étant donné un ensemble d'entraînement qui comprend *N* exemples, où chaque exemple est représenté par un vecteur d'entrée x_i et une étiquette correspondante y_i, avec un échantillonnage en B échantillons, chaque arbre sera ajusté à ces échantillons. La moyenne des prédictions de tous les arbres individuels sur l'échantillon X_b sera prise comme la prédiction du foret aléatoire.

$$f(X) = \frac{1}{B}\sum_{b=1}^{B} f_b(X_b) \tag{2.16}$$

Ce processus d'amorçage conduit à de meilleures performances du modèle car il réduit la variance du modèle sans augmenter le biais. Cela signifie que si la prédiction d'un seul arbre est très sensible au bruit dans son ensemble d'apprentissage, la moyenne de

nombreux arbres n'est pas si sensible tant que les arbres ne sont pas corrélés. La simple formation de plusieurs arbres sur un seul ensemble de formation produit des arbres fortement corrélés. L'échantillonnage bootstrap est un moyen d'éliminer une méthode pour les dépendances d'arbres.

De plus, une estimation de l'incertitude de prédiction peut être considérée comme l'écart type des prédictions pour tous les arbres de régression individuels :

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(X_b) - f(X))^2}{B}} \tag{2.17}$$

Algorithme 4 Construction d'une Foret Aléatoire en utilisant l'échantillonnage bootstrap

Entrées
    Matrice de données d'apprentissage *X_train*
    Vecteur de valeurs cibles associées aux données d'apprentissage *y_train*
    Nombre d'arbres dans la foret *n_tree*
    Taille de l'échantillon d'entrainement *sample_size*
Sorties
    Liste des arbres de foret *Trees*
Fonction train(*X_train, y_train, n_tree, sample_size*) **Faire**
    Trees <- Liste vide
    Pour *i* de 1 à *n_tree* **Faire**
        (*X_bootstrap*, *y_bootstrap*) <- un échantillon d'entrainement bootstrap en
        échantillonnant depuis (*X_train*, *y_train*) avec la taille *sample_size*
        T <- arbre de décision crée en utilisant (*X_bootstrap*, *y_boostrap)* comme
        données d'entrainement
        Ajouter l'arbre crée à *Trees*
    Fin Pour
    Retourner *Trees*
Fin Fonction

La Foret Aléatoire est utilisée dans une variété de domaine comme, la classification des données médicales pour le diagnostic des maladies, la prévision financière, la reconnaissance d'images. L'algorithme de la Foret Aléatoire possède des avantages qui se résident dans sa capacité à généraliser à de nouvelles données, la gestion de grand nombre de caractéristiques, et la réduction du biais. Cependant, l'algorithme peut être difficile pour interpréter.

2.4.5 Réseaux de neurones

Les réseaux de neurones sont des modèles inspirés par le fonctionnement des neurones biologiques. Ces réseaux sont constitués de plusieurs couches de neurones artificiels qui travaillent ensemble pour produire une sortie en fonction des entrées reçues. Les poids associés à ces connexions sont ajustés au cours d'un processus d'apprentissage automatique supervisé, où les erreurs produites par le réseau sont minimisées pour atteindre la meilleure performance possible sur la tâche spécifique pour laquelle il a été formé [15].

En générale, les signaux provenant d'autres neurones du réseau sont sommés en un signal en utilisant une somme pondérée.

$$a = b + \sum_{i=1}^{n} w_i x_i \tag{2.16}$$

Où *b* est le biais, x_i le signal d'entrée *i* et w_i est le poids correspondant. Une fonction d'activation *f* sera appliqué sur la somme pondérée *a* pour avoir la sortie du neurone.

$$z = f(a) = f(b + \sum_{i=1}^{n} w_i x_i) \tag{2.17}$$

La fonction d'activation d'un neurone opère comme une transformation qui combine linéairement les signaux d'entrée pour produire le signal de sortie, selon une fonction mathématique donnée. Le choix de la fonction d'activation peut être différent selon les types de réseaux de neurones, qui peuvent utiliser des fonctions distinctes comme les fonctions linéaires, les fonctions logistiques, les fonctions sigmoïdes.

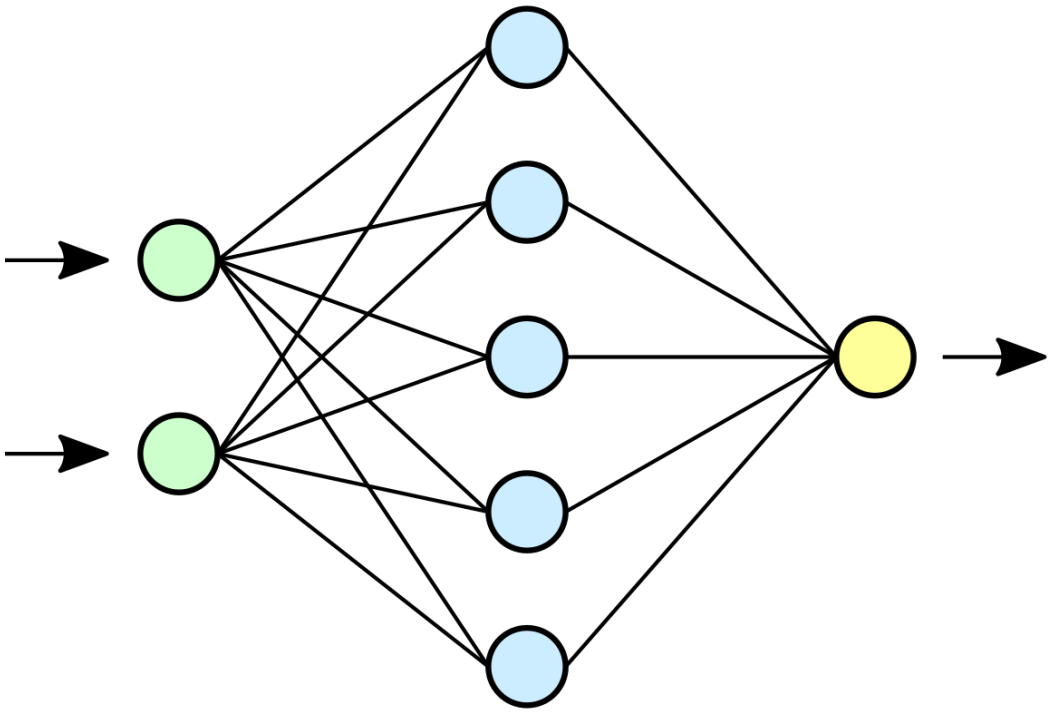


Figure 4 – Une architecture de réseau de neurones [16]

Il existe plusieurs types de réseaux de neurones artificiels qui se diffèrent en fonction de leur architecture et de leur capacité à résoudre des problèmes spécifiques Les réseaux de neurones multicouches ou MLP (Multi-Layer Perceptron) sont une forme courante de réseaux de neurones artificiels. Ils utilisent la méthode de propagation avant, et sont constitués de plusieurs couches de neurones. Les réseaux de neurones multicouches sont des modèles puissants capables d'apprendre des relations complexes entre les données.

Algorithme 5 Ajustement des poids et des biais du Réseux Neurones Multicouches

Entrées
    Matrice de données d'apprentissage *X_train*
    Vecteur de valeurs cibles associées aux données d'apprentissage *y_train*
    Nombre d'itérations *T*
    Taille des couches cachées *C*
    Fonction d'activation *f*
    Taux d'apprentissage *lr*
Sorties
    Matrice de poids *w*
    Vecteur de bais *b*
Fonction train(*X_train, y_train, T, f*) **Faire**
    w <- matrice à 0
    b <- vecteur à 0
    Pour *itr* de 1 à *T* **Faire**
        Pour chaque indice *i* dans *X_train* **Faire**
            a <- propagation_avant(*X_train*[*i*], *w*, *b*, *f*)
            delta <- retropropagation*(y_train*[*i*], *a*, *w*, *f*)
            Pour *c* de 1 à *C* **Faire**
                w[*c*] <- *w*[*c*] + *lr* * *delta*[*c*] * *a*[*c-1*]
                b[*c*] <- *b*[*c*] + *lr* * *delta*[*c*]
        Fin Pour chaque
    Fin Pour
    Retourner *w*, *b*
Fin Fonction

La capacité des réseaux de neurones à généraliser à partir de données d'entraînement en vue d'effectuer des tâches complexes avec une grande précision est une caractéristique clé de cette technique. Les réseaux de neurones sont largement utilisés dans les domaines de la reconnaissance d'images et du traitement du langage naturel, ainsi que pour la prédiction de séries chronologiques, la classification de données et la reconnaissance de la parole. Ils peuvent également être utilisés pour améliorer la compréhension des données médicales et pour prédire les maladies et les conditions médicales grâce à des analyses complexes.

2.4.6 Support Vector Machines

Les machines à vecteurs de support (SVM) sont un type d'algorithme d'apprentissage automatique supervisé utilisé pour les problèmes de classification, de

régression, et de détection d'anomalie. Les SVM permettent de trouver une frontière optimale entre deux classes de données. Cette frontière est connue sous le nom d'hyperplan et est utilisée pour prédire la classe d'un nouveau point de données en mesurant sa distance par rapport à l'hyperplan [17].

Il existe plusieurs types de fonctions noyau fréquemment utilisées dans les SVMs. Les trois fonctions noyau les plus couramment utilisées sont :

●**Noyau linéaire :** Le noyau linéaire est la fonction noyau la plus simple et la plus directe. Il correspond au scalaire sans transformation standard et est défini comme :

$$K(x, x') = x^T x' + b \tag{2.18}$$

●**Noyau polynomial :** Le noyau polynomial est un noyau non linéaire qui projette les caractéristiques d'entrée dans un espace de caractéristiques de plus haute dimension. Il est défini comme :

$$K(x, x') = (x^T x' + c)^d \tag{2.19}$$

●**Noyau de la fonction gaussienne radiale :** Le noyau de la fonction gaussienne radiale ou RBF est un noyau non linéaire qui projette les caractéristiques d'entrée dans un espace de caractéristiques de plus haute dimension. Il est défini comme :

$$K(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{2.20}$$

Où γ est une constante positive et $||x - x'||$ désigne la distance euclidienne entre x et x'.

Le noyau RBF est approprié pour les données qui ne sont pas séparables linéairement et qu'une frontière douce et non linéaire est requise.

Algorithme 6 Entrainement de Support Vector Machines

```
Entrées
    Matrice de données d'apprentissage X_train
    Vecteur de valeurs cibles associées aux données d'apprentissage y_train
    Nombre d'itérations T
    Paramètre de régularisation C
    Noyau à utiliser kernel
Sorties
    Vecteur de poids w
    Bais b
Fonction train(X_train, y_train, T, C) Faire
    w <- vecteur à 0
    b <- 0
    Pour itr de 1 à T Faire
        Pour chaque indice i dans X_train Faire
            z <- w[i] * X_train[i] + b
            Si  (z * y_train[i]) <= 0 Faire
                w[i] <- w[i] + C * y_train[i] * X_train[i]
                b <- b + C * y_train[i]
            Fin Si
        Fin Pour chaque
    Fin Pour
    Retourner w, b
Fin Fonction
```

Les machines à vecteurs de support (SVM) sont souvent utilisées pour la classification de données, la reconnaissance d'images, la reconnaissance de formes et la sélection de caractéristiques. Parmi les avantages des SVM, leurs performances élevées, leur simplicité et capacité à apprendre indépendamment de la dimension du vecteur d'entrée. Cependant ils peuvent rencontrer des problèmes liés au déséquilibrage de classe et au choix des paramètres.

2.5 Etat de l'art

La détection précoce des troubles fonctionnels hépatiques a été le sujet de nombreuses études dans la littérature scientifique, et plusieurs approches ont été proposées et appliquées. Le tableau ci-dessus présente un résumé des travaux antérieurs qui ont exploré les techniques d'apprentissage automatique pour la détection de l'hépatite. Tous ces

travaux ont utilisé le même jeu de données qui est Indian Liver Disease Patients Data set (ILPD).

Table 2 – Tableau des travaux de détection de la maladie de l'hépatite en utilisant les techniques d'apprentissage machine [18] [19] [20] [21] [22]

Nom de l'auteur	Méthodologie	Performances
Kuzhippallil, Maria Alex, JOSEPH, Carolyn, et KANNAN, A. [18]	Extension de l'algorithme XGBoost avec un algorithme génétique.	LightGBM, Random Forest, et Stacking estimator offrent les plus grandes exactitudes avec un maximum de 88%.
RAHMAN, AKM Sazzadur, SHAMRAT, FM Javed Mehedi, TASNIM, Zarrin, et al. [19]	Utilisation de l'algorithme de Régression logistique, SVM, Random forest, Naive bayes, J48 et k-PPV.	La Régression logistique a obtenu des meilleurs résultats avec exactitude de 75 %, précision de 91% et F-mesure de 83%. SVM a obtenu la plus grande AUC pour ROC.
Priya, M. Banu, P. Laura Juliet et P. R. [20]	Normalisation Min-max, sélection de fonctionnalités.	J48 a donné les meilleurs résultats avec une exactitude de 95,04 %.
SINGH, Jagdeep, BAGGA, Sachin, et KAUR, Ranjodh. [21]	Utilisation de la Régression logistique, SMO, Random Forest, Naive Bayes, J48 et IBk avec la sélection de caractéristiques.	Logistic Regression a obtenu la plus grande précision de 72,50% sans la technique de sélection de caractéristiques. Random Forest a obtenu la plus grande précision de 74,36% avec la technique de sélection de caractéristiques.
VIJAYARANI, S. et DHAYANAND, S. [22]	Utilisation de l'algorithme Naive Bayes et SVM.	Le SVM a la meilleure précision avec 76.6%. Naive Bayes

		nécessitait un temps d'exécution de 1670 millisecondes.

2.5.1 Analyse comparative des techniques d'apprentissage automatique pour les patients indiens atteints d'une maladie du foie

Le travail de Kuzhippallil, Maria Alex, JOSEPH, Carolyn, et KANNAN, A. [18] a proposé un nouveau modèle de classification pour prédire les maladies du foie en utilisant une extension de l'algorithme XGBoost avec un algorithme génétique. Le jeu de données Indian Liver Disease Patients (ILPD) provenant du référentiel UCI est utilisé pour évaluer la méthode proposée.

La détection des valeurs aberrantes est utilisée pour identifier les valeurs extrêmes déviantes et elles sont éliminées à l'aide de l'algorithme Isolation Forest. La sélection de caractéristiques a été aussi utilisée pour améliorer la performance du modèle. Les performances sont mesurées en termes d'exactitude, de précision, de rappel, de F-mesure et de complexité temporelle.

Table 3 – Résultats avant la sélection de caractéristiques et l'élimination des valeurs aberrantes [18]

Algorithme	Exactitude	Précision	Rappel	F-mesure
Multilayer Perceptron	71%	50%	71%	59%
KNN	72%	69%	72%	69%
Logistic Regression	74%	71%	74%	70%
Decision Tree	67%	68%	67%	67%
Random Forest Tree	74%	72%	74%	72%
Gradient Boosting	66%	67%	66%	66%
AdaBoost	68%	64%	68%	65%
XGBoost	70%	69%	70%	69%

Light GBM	70%	70%	70%	70%
Stacking Estimator	83%	83%	83%	83%

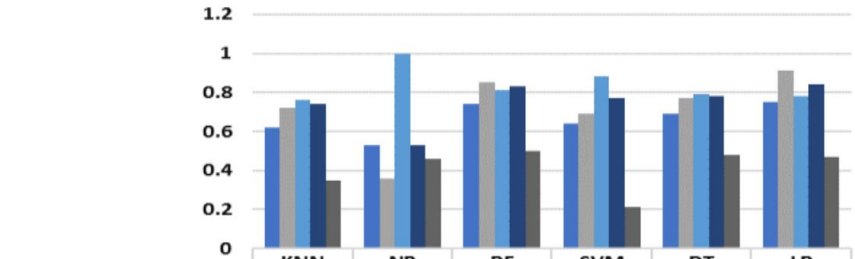
Table 4 – Résultats après la sélection de caractéristiques et l'élimination des valeurs aberrantes [18]

Algorithme	Exactitude	Précision	Rappel	F-mesure
Multilayer Perceptron	82%	81%	82%	80%
KNN	79%	77%	79%	74%
Logistic Regression	76%	72%	76%	72%
Decision Tree	84%	84%	84%	84%
Random Forest Tree	88%	88%	88%	88%
Gradient Boosting	84%	84%	84%	84%
AdaBoost	83%	83%	83%	83%
XGBoost	86%	86%	86%	86%
Light GBM	86%	85%	86%	85%
Stacking Estimator	85%	85%	85%	85%

L'extension de l'algorithme XGBoost avec un algorithme génétique a amélioré l'exactitude de la classification et réduit le temps de classification. Cependant, le fait de ne travailler que sur un seul jeu de données qui est le Indian Liver Disease Patients Dataset (ILPD) sans équilibrer ou augmenter les données, peut affecter la généralisation de cette approche aux autres données.

2.5.2 Une étude comparative sur la prédiction des maladies du foie à l'aide d'algorithmes d'apprentissage automatique supervisé

L’étude comparative de RAHMAN, AKM Sazzadur, SHAMRAT, FM Javed Mehedi, TASNIM, Zarrin, et al. [19] avait pour objectif de réduire les coûts élevés liés au diagnostic de maladies du foie en utilisant des algorithmes de régression logistique, de SVM, de forêt aléatoire, de Naive bayes, de J48 et de KNN. Les données du référentiel UCI d'apprentissage automatique comprenant 583 patients atteints de maladies du foie avec 10 paramètres et 1 paramètre comme classe cible ont été utilisées.



	KNN	NB	RF	SVM	DT	LR
Classification Accuracy	0.62	0.53	0.74	0.64	0.69	0.75
Precision	0.72	0.36	0.85	0.69	0.77	0.91
Sensitivity	0.76	1	0.81	0.88	0.79	0.78
f-1	0.74	0.53	0.83	0.77	0.78	0.84
Specificity	0.35	0.46	0.5	0.21	0.48	0.47

Figure 5 – Comparaison des performances des six techniques d'apprentissage automatique supervisé [19]

Les résultats ont montré que la régression logistique et l’algorithme SVM sont des algorithmes prometteurs pour la prédiction des maladies du foie, et que la régression logistique avait atteint la plus haute exactitude de 75 %. Cependant, il est important de noter que cette précision n'est pas encore suffisante pour une utilisation clinique fiable. Cette étude utilise un nombre limité données déséquilibrées, ce qui peut affecter la généralisation de cette approche aux autres données.

2.5.3 Analyse des performances de la prédiction des maladies du foie à l'aide d'algorithmes d'apprentissage automatique

L'étude de Priya, M. Banu, P. Laura Juliet et P. R. [20] a permis de développé un modèle de classification pour prédire les maladies du foie en utilisant des techniques d'exploration de données et des algorithmes de classification. Les données utilisées dans cette étude provenaient de l'ensemble de données de patients atteints de maladies du foie en Inde (ILPD), qui comprenait 583 enregistrements, dont 416 pour des patients atteints de maladies du foie et 167 pour des patients non atteints de maladies du foie.

Table 5 – Les résultats de l'évaluation de performances de J.48, MLP, SVM, Random Forest, Bayesnet [20]

Algorithme	Erreur absolue moyenne (PSO)	Erreur quadratique moyenne (PSO)	Erreur quadratique relative (PSO)	Exactitude (Greedy Step Wise)	Exactitude (PSO)
J.48	0.507	0.487	73.33	68.77	95.04
MLP	0.703	0.403	69.23	68.26	77.54
SVM	0.712	0.425	71.45	71.35	73.44
Random Forest	0.604	0.467	68.44	70.32	80.22
Bayesnet	0.572	0.406	74.25	67.23	90.33

Les résultats d'évaluation de la performance indiquent que l'algorithme J48 a donné les meilleurs résultats avec une précision de 95,04%. Ces résultats peuvent aider à améliorer la précision du diagnostic des maladies du foie, mais l'étude aurait pu être renforcée en équilibrant ou en augmentant les données.

2.5.4 Prévision logicielle des maladies du foie avec des techniques de sélection et de classification des caractéristiques

SINGH, Jagdeep, BAGGA, Sachin, et KAUR, Ranjodh [21] ont mené une étude comparative en utilisants six algorithmes d'apprentissage automatique supervisé qui ont été

comparés pour prédire les maladies du foie, et des techniques de sélection de caractéristiques.

L'étude a développé le logiciel Intelligent Liver Disease Prediction (ILDPS) en utilisant divers algorithmes de classification, y compris la régression logistique, SMO, Random Forest, Naive Bayes, J48 et k-plus proches voisins (IBk). Des techniques de sélection de caractéristiques ont été utilisées pour évaluer les performances des différents classificateurs. L'ensemble de données de patients atteints du foie indien (ILPD) de la base de données de l'Université de Californie à Irvine a été utilisé, et l'exactitude des prédictions a été évaluée.

Table 6 – Les résultats des différents classificateurs sans la technique de sélection des caractéristiques [21]

Classificateurs	**Instances correctement classées (%)**	**Statistiques de kappa**	**Erreur absolue moyenne**
Logistic Regression	72.50	0.2169	0.3422
Naive Bayes	55.74	0.2249	0.4407
SMO	71.35	0	0.2864
IBk	64.15	0.1664	0.3590
J48	68.78	0.1774	0.3292
Random Forest	71.53	0.2227	0.3394

Table 7 – Les résultats des différents classificateurs avec l'aide de la technique de sélection des caractéristiques [21]

Classificateurs	Instances correctement classées (%)	Statistiques de kappa	Erreur absolue moyenne
Logistic Regression	74.36	0.0133	0.4091
Naive Bayes	55.9	0.2390	0.4471
SMO	71.36	0	0.2864
IBk	67.41	0.2056	0.3266
J48	70.67	0.0306	0.3885
Random Forest	71.87	0.2499	0.3399

L’évaluation des performances montre que l'algorithme Random Forest a obtenu la plus grande précision de 74,56% avec la technique de sélection de caractéristiques et que Naive Bayes offrait la plus grande précision de 71,82% sans la technique de sélection de caractéristiques. La meilleure précision obtenue reste encore relativement faible pour une utilisation clinique fiable. Le problème de la généralisation de cette approche peut exister à cause de ne travailler que sur un seul jeu de données qui est déséquilibré.

2.5.5 Prédiction des maladies du foie à l'aide des algorithmes SVM et Naïve Bayes

L’étude de VIJAYARANI, S. et DHAYANAND, S. [22] a comparé les performances de deux algorithmes de classification, Naive Bayes et SVM, pour prédire les maladies du foie en utilisant l'ensemble de données ILPD.

Table 8 – Mesure de précision de SVM et Naive Bayes pour l'ensemble de données sur les maladies du foie [22]

Algorithme	Instances correctement classées	Instances incorrectement classées	Taux de TP	Précision	F Mesure
Naive Bayes	61.28	38.72	0.612	0.558	0.251
SVM	79.66	20.34	0.796	0.766	0.331

Table 9 – Analyse du temps d'exécution de SVM et Naive Bayes pour l'ensemble de données sur les maladies du foie [22]

Algorithme	Temps d'exécution en ms
Naive Bayes	1670.00
SVM	3210.00

Les résultats ont montré que SVM était considéré comme le meilleur algorithme de prédiction de maladies du foie en raison de sa précision de classification plus élevée. En revanche, Naïve Bayes nécessitait un temps d'exécution minimal. Les limitations de ce travail incluent le fait que les données utilisées pour entraîner et tester les algorithmes proviennent d'un ensemble de données déséquilibré (Indian Liver Disease Patients Dataset), ce qui peut affecter la généralisation des résultats aux autres données. De plus, le fait que seule une poignée d'algorithmes ait été testée pour la classification des maladies du foie laisse penser qu'il est possible que d'autres approches pourraient être plus performantes. Mais ce travail peut servir de base pour des travaux futurs qui cherchent à améliorer les performances des algorithmes de détection précoce de maladies du foie, ce qui pourrait avoir un impact important sur la santé publique en améliorant le diagnostic et le traitement des maladies du foie comme l’hépatite.

2.6 Conclusion

Dans ce chapitre nous avons abordé les notions fondamentales de l'apprentissage automatique en mettant l'accent sur la classification, un sujet pertinent dans notre étude. Ensuite, nous avons examiné plusieurs études antérieures qui ont utilisé des approches basées sur l'apprentissage automatique pour la détection des maladies du foie, en particulier l'hépatite.

Chapitre 3

Etude expérimentale de la détection de l'hépatite

3.1 Introduction

Lors de la construction d'un modèle de classification basé sur l'apprentissage automatique, plusieurs étapes préliminaires sont nécessaires afin de préparer les données et d'optimiser les performances du modèle. Le prétraitement des données et la sélection des variables sont deux étapes clés dans ce processus. Le prétraitement consiste à adapter les données pour qu'elles puissent être utilisées dans le cadre de l'apprentissage automatique, tandis que la sélection des variables permet d'identifier les caractéristiques les plus pertinentes pour détecter l'hépatite. Cependant, la construction d'un modèle d'apprentissage automatique peut présenter des défis. Les données peuvent être déséquilibrées lorsque les différentes classes sont représentées de manière inégale dans l'ensemble des données. De plus, le choix des valeurs des hyperparamètres est crucial pour obtenir un modèle optimal, car ces paramètres externes au modèle lui-même influencent des aspects tels que sa complexité, la régularisation et la vitesse d'apprentissage, ce qui impacte la performance et la capacité de généralisation du modèle.

Au niveau de ce chapitre, nous présentons la partie de la mise en œuvre de notre approche. Nous commençons par décrire l'environnement logiciel que nous avons utilisé pour le développement de notre approche et de notre application. Puis, nous détaillons l’ensemble de données utilisé en présentant ses caractéristiques. Nous abordons ensuite les différentes étapes de prétraitement des données, l’étape de sélection des variables, l'ajustement des hyperparamètres pour choisir le meilleur modèle, la classification et l’évaluation des performances des meilleurs modèles. A la fin de ce chapitre, nous présentons notre application qui offre la possibilité d’utiliser l’approche pour la détection de la maladie de l’hépatite.

3.2 Hyperparamètre

Un hyperparamètre est un paramètre dont la valeur est définie avant le début du processus d'apprentissage, et qui est spécifique au processus d'entraînement lui-même. Les hyperparamètres jouent un rôle crucial dans la configuration et le fonctionnement des modèles d'apprentissage automatique. Ils sont des paramètres externes au modèle qui doivent être réglés avant l'apprentissage contrairement aux autres paramètres qui sont ajustés par l'entraînement sur des données existantes, et ils sont des paramètres qui influencent la performance, la complexité ainsi que la capacité de généralisation du modèle [23].

3.3 L’ajustement des hyperparamètre par la recherche en grille

L'ajustement des hyperparamètres est une technique importante lors de l'optimisation des modèles d'apprentissage automatique. L’ajustement des hyperparamètres consiste à trouver les meilleures valeurs pour les hyperparamètres d'un modèle d'apprentissage automatique, ce qui peut grandement améliorer les performances du modèle [24].

Il existe plusieurs méthodes d'optimisation des hyperparamètres. Parmi celles-ci, la recherche en grille (grid search) qui est l'une des méthodes les plus couramment utilisées. La recherche en grille consiste à définir une grille de valeurs possibles pour chaque hyperparamètre, puis à évaluer le modèle pour toutes les combinaisons possibles de ces valeurs afin de trouver la configuration optimale. Bien que cette méthode soit efficace, elle peut être coûteuse en termes de temps de calcul lorsque l'espace des hyperparamètres est grand.

3.4 Environnement logiciel

Pour le développement de notre approche, nous avons utilisé le langage de programmations : Python, sous l'éditeur du texte, Visual Studio Code.

3.4.1 Python

Python est un langage de programmation orienté objet et interprété, qui offre une grande flexibilité et une facilité d'utilisation. Le langage Python est utilisé pour une variété de tâches telles que la création de sites web, le développement de logiciels et l'analyse de données [25].

Nous avons utilisé plusieurs bibliothèques du langage Python pour notre travail, notamment :

- **Matplotlib :** est une bibliothèque Python complète qui permet de créer des visualisations en Python, qu'elles soient statiques, animées ou interactives.

- **Seaborn :** est une bibliothèque de visualisation de données Python qui utilise Matplotlib. Elle propose une interface de haut niveau pour créer des graphiques statistiques.

- **Numpy :** est une extension de Python qui permet de manipuler des tableaux multidimensionnels. Elle offre des fonctions pour effectuer des opérations mathématiques et statistiques sur ces tableaux.

- **Scikit-learn :** est la bibliothèque Python la plus importante en matière d'apprentissage automatique. Elle contient de nombreux algorithmes, tels que les forêts aléatoires, les k plus proches voisins, et les machines à vecteurs de support. Scikit-learn est une bibliothèque très complète pour l'apprentissage automatique, qui permet aux développeurs d'implémenter facilement des modèles de machine learning dans leurs projets.

- **Streamlite :** est une bibliothèque open-source performante et facile à utiliser qui permet aux développeurs de créer des applications web interactives pour l'analyse de données et l'apprentissage automatique. Elle permet la création d'interfaces graphiques pour visualiser des données, interagir avec des modèles de machine learning et partager les résultats avec les utilisateurs finaux.

3.4.2 Editeur du texte Visual Studio Code

Visual Studio Code est un éditeur de texte open source performant qui prend en charge les opérations de développement. Visual Studio Code est compatible avec de nombreux langages de programmation couramment utilisés, tels que JavaScript, Python, C++ , etc. [26].

3.5 Jeu de données utilisé

Le jeu de données qui a été sélectionné pour notre travail est, Indian Liver Patient Dataset [27]. Ce jeu de données est principalement axé sur les cas de maladies du foie, notamment l'hépatite, et représente une collection de données médicales de 583 patients indiens. Parmi eux, 416 patients ont été diagnostiqués avec une maladie du foie, tandis que 167 autres n'en ont pas.

L'application de méthodes d'apprentissage automatique sur les informations contenues dans l'Indian Liver Patient Dataset pourrait permettre de développer des modèles prédictifs performants, capables de détecter l'hépatite avec rapidité et exactitude améliorées. Il est toutefois important de relever que cette base de données est confrontée à des difficultés, notamment un déséquilibre des classes où le nombre de patients atteints de la maladie est plus élevé à celui des individus qui n'en ont pas. La prise en compte de ces obstacles se révèle essentielle pour mettre au point des modèles de classification précis, aptes à s'adapter à de nouvelles données.

Table 10 – Description des variables du jeu de données Indian Liver Patients Dataset [27]

Nom	Description	Type	Valeur
Age	L'âge du patient	Numérique	En ans
Gender	Le genre du patient	Nominal	Male, Female
Total_Bilirubin	La quantité totale de bilirubine dans le sang	Numérique	En mg/dL
Direct_Bilirubin	La quantité de bilirubine directe dans le sang	Numérique	En mg/dL
Alkaline_Phosphotase	Le niveau de phosphatase alcaline dans le sang	Numérique	En U/L
Alamine_Aminotransferase	Le niveau d'alanine aminotransférase dans le sang	Numérique	En U/L
Aspartate_Aminotransferase	Le niveau d'aspartate aminotransférase dans le sang	Numérique	En U/L
Total_Protiens	La quantité totale de protéines dans le sang	Numérique	En g/dL
Albumin	Le niveau d'albumine dans le sang	Numérique	En g/dL
Albumin_and_Globulin_Ratio	Le rapport entre l'albumine et la globuline dans le sang	Numérique	En g/dL
Dataset	La classe des exemples	Nominal	1 (pour patient malade), 2 (pour

			patient qui n'est pas malade)

3.6 Prétraitement des données

L'étape du prétraitement des données est une étape importante dans le processus de modélisation, cette étape va permettre de préparer les données pour l'analyse.

3.6.1 Exploration des données

L'exploration des données est utilisée pour analyser les données afin de découvrir des informations qui n'étaient pas évidentes auparavant.

Les dix premières lignes de l'ensemble de données utilisé ont été affichées pour avoir une vue brève sur le jeu de données. Les noms des variables originales, à savoir Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase, Alamine_Aminotransferase, Aspartate_Aminotransferase, Albumin_and_Globulin_Ratio et Dataset, ont été modifiés pour des noms plus courts et plus conviviaux : Total Bili, Direct Bili, AlkPhos, Sgpt/ALT, Sgot/AST, A/G ratio et Class.

	Age	Gender	Total Bili	Direct Bili	AlkPhos	Sgpt/ALT	Sgot/AST	Total_Protien	Albumin	A/G ratio	Class
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
6	26	Female	0.9	0.2	154	16	12	7.0	3.5	1.00	1
7	29	Female	0.9	0.3	202	14	11	6.7	3.6	1.10	1
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	2
9	55	Male	0.7	0.2	290	53	58	6.8	3.4	1.00	1

Figure 6 – Dix premières lignes du jeu de données Indian Liver Patients Dataset

Ensuite, nous avons afficher le nombre d’échantillon et d’attributs et les types des données, cela peut être utile pour savoir rapidement s'il y a des données manquantes en comparant le nombre d’attribut du jeu de données avec le nombre d'exemples de chaque colonne. Nous allons ainsi, afficher les statistiques pour chaque colonne du jeu de données. La figure suivante montre les statistiques des données.

	Age	Gender	Total Bili	Direct Bili	AlkPhos	Sgpt/ALT	Sgot/AST	Total_Protien	Albumin	A/G ratio	Class
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.000000
mean	44.746141	0.756432	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	0.713551
std	16.189833	0.429603	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.452490
min	4.000000	0.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	0.000000
25%	33.000000	1.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	0.000000
50%	45.000000	1.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.000000
75%	58.000000	1.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000	1.000000
max	90.000000	1.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	1.000000

Figure 7 – Statistiques du jeu de données Indian Liver Patients Dataset

En consultant les statistiques de chaque colonne, nous pouvons déduire que, les valeurs existantes sont dans le rang réalistique, il existe quatre valeurs manquantes pour la colonne du ratio d’albumin et du globulin.

Nous avons utilisé la matrice de corrélation, un outil qui permet de visualiser la relation entre les variables. Chaque cellule de la matrice est colorée en fonction du coefficient de corrélation de la paire qu'elle représente.

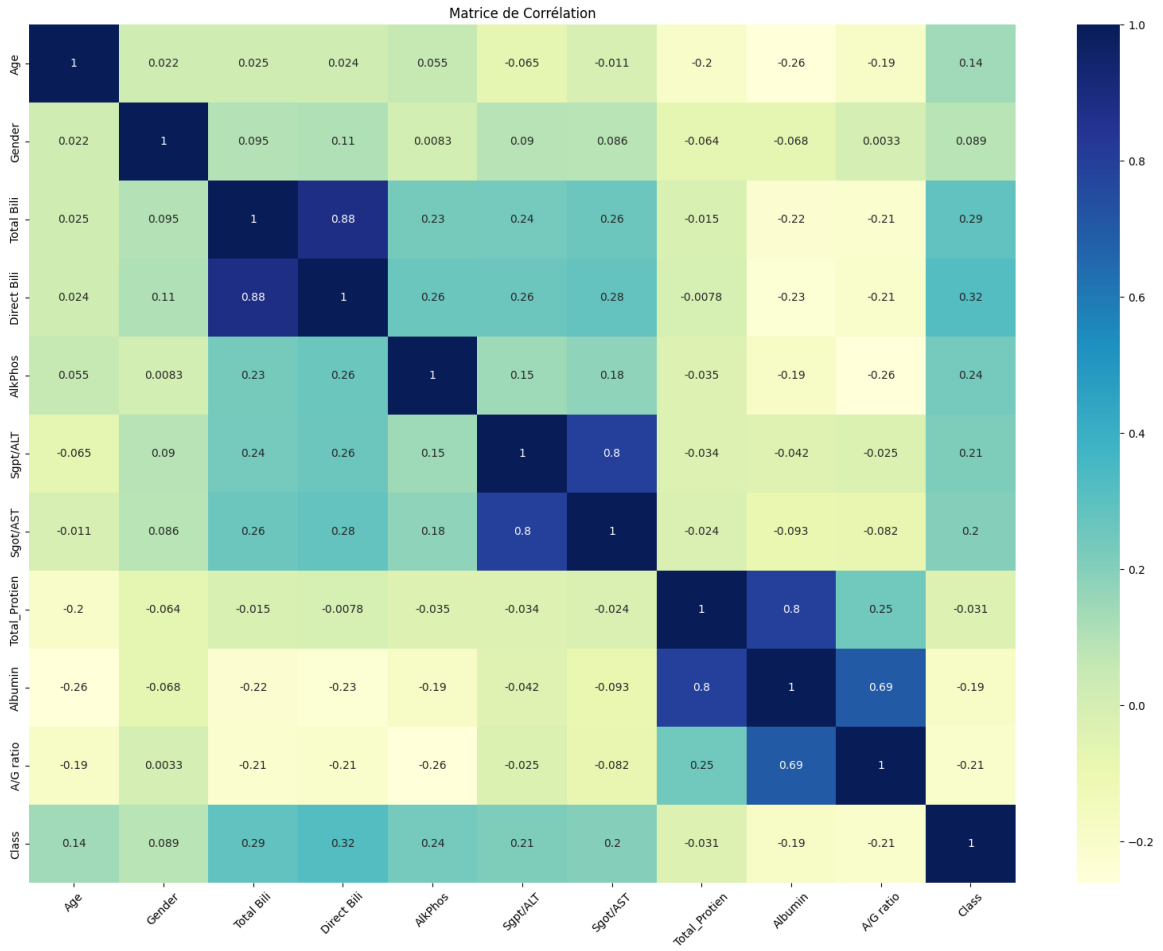


Figure 8 – Matrice de corrélation du jeu de données équilibré

Comme le montre la matrice de corrélation, les variables Total Bili et Direct Bili sont les plus fortement corrélées avec la variable cible. Les variables AlkPhos, Sgot/AST, Sgpt/ALT et l'âge sont corrélées à un degré similaire avec la variable cible (0,14-0,25). Le genre, la protéine totale sont corrélées sont similairement corrélés avec la variable cible mais avec un degré moins que le degré des autres variables.

3.6.2 Gestion des valeurs manquantes et des doublons

Après la vérification de l'existence des doublons, les données dupliquées ont étaient supprimées à l'aide de la fonction. De plus, après la vérification de la présence des valeurs

manquantes, nous avons confirmé qu'il existe quatre valeurs manquantes ont été pour la colonne du ration d'albumin et du globulin.

Il existe deux options pour gérer les valeurs aberrantes : La première option consiste à supprimer toutes les observations ayant des valeurs nulles, cette solution n'entraînera pas une perte de données importante dans ce cas, mais comme il ne manque qu'un seul attribut sur 10 pour ces lignes, cette option ne sera pas considérée. La deuxième option consiste à calculer la moyenne d'une colonne spécifique et à la remplacer dans cette colonne où nous avons des valeurs nulles. Dans ce cas, l'option choisie est la deuxième, en remplaçant chaque valeur manquante de la colonne du ration d'albumin et du globulin par la moyenne.

La figure suivante montre le nombre de valeurs manquantes pour chaque colonne avant le remplacement des valeurs.

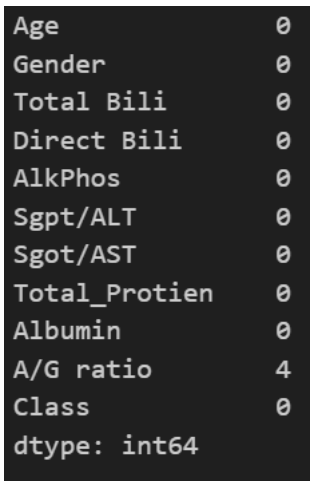


Figure 9 – Nombre de valeurs manquantes pour chaque colonne avant leur suppression

La figure suivante montre le nombre de valeurs manquantes pour chaque colonne après le remplacement des valeurs.

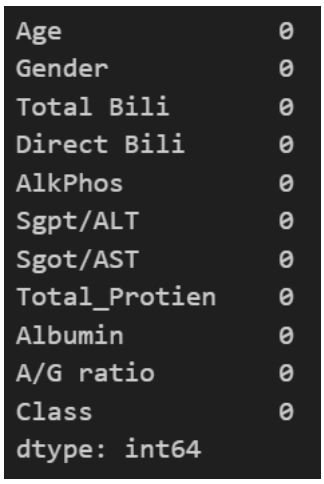


Figure 10 – Nombre de valeurs manquantes pour chaque colonne après leur suppression

3.6.3 Augmentation des données

L'augmentation de données est une méthode qui peut aider à résoudre les problèmes de sur-apprentissage en augmentant le nombre d'exemples dans un ensemble de données. Pour l'Indian Liver Patients Dataset, nous avons utilisé la technique de suréchantillonnage SMOTE (Synthetic Minority Over-sampling Technique) pour augmenter les données et équilibrer la distribution des classes et améliorer les performances du modèle de classification. La technique SMOTE est utilisée pour équilibrer les classes dans les ensembles de données d'apprentissage en générant des exemples synthétiques de la classe minoritaire. Cette technique combine des caractéristiques de l'échantillon de la classe minoritaire et ses voisins proches dans l'espace des caractéristiques. La distance entre les voisins est mesurée à l'aide d'une mesure de distance, comme la distance Euclidienne.

3.6.4 Division des données

Pour évaluer le modèle tout en augmentant ces performances, les données prétraitées sont divisé en un rapport 80:20.

L'approche consiste à diviser l'ensemble de données en deux parties : une partie d'entraînement sur laquelle le modèle est appris qui représente 80% de l'ensemble de

données initial, et une partie de test sur laquelle le modèle est testé et sa performance évaluée, le pourcentage des données à conserver pour le test est 20%.

3.6.5 Standardisation des données

Les données du Data Frame sont mises à l'échelle en utilisant la moyenne et l'écart type des données d'entrainement, ce qui peut aider à améliorer les performances de certains algorithmes d'apprentissage automatique qui peuvent être influencés par l'échelle des caractéristiques d'entrée.

3.7 Classification

Dans cette approche proposée, nous avons utilisé plusieurs algorithmes de classification pour entraîner des modèles sur les données d'entraînement, notamment :

- SVM.
- K plus proches voisins.
- Foret Aléatoire.
- Naïf Bayes.

Pour optimiser les performances des algorithmes de classification nous avons utilisé la fonction de recherche en grille « GridSearchCV » de la bibliothèque sklearn qui permet l'ajustement des hyperparamètres des classifieurs. Il s'agit d'une méthode de recherche en grille qui consiste à tester différentes combinaisons de paramètres prédéfinies afin de trouver la meilleure configuration pour notre modèle. Pour chaque combinaison de paramètres, le modèle a été entraîné et évalué à l'aide de la validation croisée, cette méthode est importée depuis la bibliothèque sklearn et consiste à diviser les données en *k* sous-ensembles différents puis elle utilise l'union de *k*-1 sous-ensembles pour l'entrainement et le sous-ensemble restant pour le test. Ce processus est répété pour chaque sous-ensemble, et l'exactitude moyenne des tests est utilisée comme exactitude finale du modèle. Dans notre étude la valeur de *k* était définie par 10.

Pour l'algorithme KNN, nous avons utilisé une grille de recherche avec des valeurs de 1 à 420 pour le nombre de voisins k. Pour l'algorithme Naïf Bayes, une liste de valeurs a été spécifiée, allant de $1e^{-2}$ à $1e^{-15}$ avec des pas de puissance de 10 pour le paramètre de régularisation. Nous avons utilisé une grille de recherche avec différentes valeurs pour le nombre d'arbres dans la Foret Aléatoire, la profondeur maximale des arbres et le critère de fractionnement de chaque nœud de l'arbre. Plus précisément, nous avons spécifié les options suivantes : nombre d'arbres dans la Foret Aléatoire : 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 580, profondeur maximale des arbres : de 1 à 22 et critère de fractionnement de chaque nœud de l'arbre : gini et entropy. En ce qui concerne l'algorithme SVM, nous avons utilisé une grille de recherche avec différentes valeurs pour le nombre d'itérations T, le paramètre de régularisation C et le noyau à utiliser (linear, rbf, sigmoid). Nous avons également spécifié une valeur pour le paramètre gamma de 1000.0.

3.8 Evaluation des résultats

Les modèles finals ont été entraînés sur les données équilibrées en utilisant les meilleurs paramètres trouvés par la recherche en grille, et leur performance a été évaluée en utilisant des métriques de performance telles que l'exactitude, la précision, le rappel et le F1-score. Nous avons obtenu les résultats présentés dans le tableau suivant.

Table 11 – Résultats de l'évaluation des algorithmes de classification pour la détection de l'hépatite sans SMOTE

Classifieur	Exactitude	Précision	Rappel	F1-score
SVM	72%	74%	47%	58%
K plus proches voisins	71%	72%	56%	61%
Naïf Bayes	73%	82%	46%	58%
Foret Aléatoire	76 %	70 %	75 %	72%

Table 11 – Résultats de l'évaluation des algorithmes de classification pour la détection de l'hépatite avec SMOTE

Classifieur	Exactitude	Précision	Rappel	F1-score
SVM	71%	70%	51%	59%
K plus proches voisins	84%	86%	74%	79%
Naïf Bayes	74%	86%	46%	60%
Foret Aléatoire	77%	73%	71%	72%

Selon les mesures d'évaluation mentionnées (exactitude, précision, rappel et F1-score), le classifieur « K plus proches voisins » avec un seul voisin avec l'utilisation de SMOTE a eu la meilleure performance parmi tous les classifieurs, ce dernier sera choisi pour la détection de l'hépatite. De plus, les performances des algorithmes après l'utilisation de SMOTE a était améliorée, et ces résultats peuvent être même plus fiables que les résultats de la performances des classifieurs avant l'utilisation de SMOTE.

En comparant les résultats de notre étude sur la détection de l'hépatite avec ceux des travaux précédents, nous constatons que notre étude a obtenu de meilleurs résultats en termes de précision (86%) par rapport à l'étude de Vijayarani et Dhayanand [25], qui a obtenu une précision de 79,66%, et à l'étude de Singh et al. [24], qui a obtenu une précision de 74,56%. De plus, notre travail a obtenu une meilleure exactitude de 84% par rapport au travail de Rahman et al. [22], qui a atteint une éxactitude maximale de 75%. Cependant, l'étude de Kuzhippallil et al. [21] a obtenu une exactitude plus élevée de 88%, et l'étude de Priya et al. [23] a atteint une exactitude de 95,04%. Néanmoins, par rapport à ces travaux antérieurs, notre étude présente des performances plus équilibrées et fiables, en évitant les déséquilibres de données. Cela représente un atout important de notre travail.

3.9 Application

Nous avons développé l'application web « ILPD App » faisant la référence à une application développée en utilisant l'ensemble de donnée Indian Liver Patients Dataset. Cette application web permet aux utilisateurs de déterminer leur risque d'avoir la maladie

de l’hépatite. Dans cette partie nous allons présenter quelques interfaces de notre application web.

3.9.1 Page d’accueil

Cette page représente la page d'accueil principale de l’application web, elle contient des informations générales sur la maladie de l’hépatite.

Figure 11 – Aperçu de la page d’accueil

3.9.2 Page de jeu de données

Cette page permet de présenter et d’afficher des détails sur le jeu de données utilisé.

Figure 12 – Aperçu de la page du jeu de données

3.9.3 Page des statistiques

Cette page va permettre de consulter des statistiques sur le jeu de données.

Figure 13 – Aperçu de la page des statistiques

3.9.4 Page de détection de l’hépatite

Cette page va permettre aux utilisateurs de saisir leurs informations médicales pour savoir s'ils sont susceptibles d’avoir la maladie de l'hépatite.

Figure 14 – Formulaire de la page de la détection de l’hépatite

3.10 Conclusion

Dans ce dernier chapitre, nous avons fait appel à la phase d'implémentation, en commençant par l'environnement logiciel utilisé et en décrivant les caractéristiques de l'ensemble de données utilisé. Nous avons ensuite abordé les différentes étapes du prétraitement des données, la sélection des variables, l'ajustement des hyperparamètres et la classification pour choisir les meilleurs modèles, ainsi que l'évaluation de leurs performances. Enfin, nous avons introduit notre application qui permet d'utiliser l'approche proposée pour la détection de la maladie de l'hépatite.

Conclusion Générale

Ce projet est une exploration approfondie des approches de l'apprentissage automatique pour la détection de l'hépatite. Les différents chapitres ont permis de couvrir des sujets variés, allant de la présentation de la maladie elle-même, aux concepts l'apprentissage automatique, ainsi qu'à l'état de l'art de ses techniques utilisées pour la détection de l'hépatite. Enfin, nous proposons notre propre approche.

Cette approche s'appuie sur des techniques d'apprentissage automatique pour améliorer l'efficacité des traitements et la qualité de vie des patients. L'approche proposée pourra être utilisée pour détecter la maladie à un stade précoce, ce qui permettrait de mettre en place un traitement plus rapidement et d'améliorer les résultats pour les patients.

Pour les perspectives de notre projet, nous proposons l'utilisation des jeux de données encore plus vastes pour améliorer la précision de la détection. Par ailleurs, nous pourrions explorer l'utilisation de techniques d'apprentissage et l'utilisation des techniques d'interprétabilité des modèles pour mieux comprendre comment les algorithmes d'apprentissage automatique prennent des décisions et comment ces décisions peuvent être expliquées aux professionnels de la santé et aux patients.

Bibliographie

[1] Hepatitis. *Hepatitis | Johns Hopkins Medicine*. https://www.hopkinsmedicine.org/health/conditions-and-diseases/hepatitis, (consulté le : 2023/01/30).

[2] Hepatitis. *National Institute of Allergy and Infectious Diseases*. https://www.niaid.nih.gov/diseases-conditions/hepatitis, (consulté le : 2023/01/30).

[3] LEE, Sid, Le Foie. *Société canadienne du cancer*. https://cancer.ca/fr/cancer-information/cancer-types/liver/what-is-liver-cancer/the-liver, (consulté le : 2023/02/04).

[4] Winchesterhospital.org. https://www.winchesterhospital.org/health-library/article?id=19580, (consulté le : 2023/02/03).

[5] Medical treatment for hepatitis. *Living with Hepatitis: Patient Care at NYU Langone Health*. https://nyulangone.org/conditions/hepatitis/treatments/medical-treatment-for-hepatitis, (consulté le : 2023/02/05).

[6] WARKAD, Shrikant Dashrath, SONG, Keum-Soo, PAL, Dilipkumar, et al. Developments in the HCV screening technologies based on the detection of antigens and antibodies. *Sensors*, 2019, vol. 19, no 19, p. 4257.

[7] Bhatt, S. (2019) *Reinforcement learning 101, Medium*. https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292, (consulté le : 2023/01/04).

[8] What is unsupervised learning? *IBM*. Available from: https://www.ibm.com/topics/unsupervised-learning, (consulté le : 2023/09/17).

[9] PALVEL, Subash, 2023, Introduction to semi-supervised learning. *Medium*. https://subashpalvel.medium.com/introduction-to-semi-supervised-learning-f1b19ad40047, (consulté le : 2023/09/15).

[10] What is a decision tree. *IBM*. https://www.ibm.com/topics/decision-trees, (consulté le : 2023/02/16).

[11] ResearchGate.net. https://www.researchgate.net/figure/A-simple-Decision-Tree-architecture-A-is-a-root-node-B-and-C-are-internal-nodes-X-Y_fig2_335102506, (consulté le : 2023/02/18).

[12] What is the K-nearest neighbors algorithm? *IBM*. https://www.ibm.com/topics/knn, (consulté le : 2023/02/16).

[13] *Naive Bayes*. https://www.ibm.com/docs/en/ias?topic=procedures-naive-bayes, (consulté le : 2023/02/19).

[14] What is Random Forest? *IBM*. https://www.ibm.com/topics/random-forest, (consulté le : 2023/02/30).

[15] What are neural networks? *IBM*. https://www.ibm.com/topics/neural-networks, (consulté le : 2023/02/18).

[16] FLORIAN GABSTEIGER, 2022, Introduction to neural networks. *Method Park by UL*. 20 December 2022. https://www.methodpark.de/blog/introduction-to-neural-networks/, (consulté le : 2023/02/18).

[17] PUPALE, Rushikesh, 2019, Support vector machines(svm) - an overview. *Medium*. 11 February 2019. https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989, (consulté le : 2022/12/01).

[18] KUZHIPPALLIL, Maria Alex, JOSEPH, Carolyn, et KANNAN, A: Comparative analysis of machine learning techniques for indian liver disease patients. Dans : *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2020, p. 778-782.

[19] RAHMAN, AKM Sazzadur, SHAMRAT, FM Javed Mehedi, TASNIM, Zarrin, et al. A comparative study on liver disease prediction using supervised machine learning algorithms. *International Journal of Scientific & Technology Research*, 2019, vol. 8, no 11, p. 419-422.

[20] PRIYA, M. Banu, JULIET, P. Laura, et TAMILSELVI, P. R. Performance analysis of liver disease prediction using machine learning algorithms. *Int. Res. J. Eng. Technol*, 2018, vol. 5, no 1, p. 206-211.

[21] SINGH, Jagdeep, BAGGA, Sachin, et KAUR, Ranjodh. Software-based prediction of liver disease with feature selection and classification techniques. *Procedia Computer Science*, 2020, vol. 167, p. 1970-1980.

[22] VIJAYARANI, S. et DHAYANAND, S. Liver disease prediction using SVM and Naïve Bayes algorithms. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 2015, vol. 4, no 4, p. 816-820.

[23] Nyuytiymbiy, K. (2022) Parameters and hyperparameters in machine learning and Deep Learning, Medium. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac, (consulté le : 2023/08/04).

[24] DAVID, Davis, 2020, Hyperparameter Optimization Techniques to Improve Your Machine Learning Model's Performance. *freeCodeCamp.org*. 12 October 2020. https://www.freecodecamp.org/news/hyperparameter-optimization-techniques-machine-learning/, (consulté le : 2023/08/04).

[25] Welcome to Python.org, *Python.org*. https://www.python.org/about/, (consulté le : 2023/06/05).

[26] *IDE et Éditeur de Code pour les Développeurs de Logiciels et les Équipes (2023) Visual Studio*. https://visualstudio.microsoft.com/fr/#vscode-section, (consulté le : 2023/06/05).

[27] ILPD (Indian Liver Patient Dataset) Data, 2022. *UCI Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets/ILPD, (consulté le : 2022/11/07).