RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU
Faculté de Génie Électrique et d'Informatique
Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Spécialité: *Ingénierie des systèmes d'information (ISI)*

Thème

Classification des arythmies ECG avec des méthodes de Machine Learning et de Deep Learning

Réalisé par :
Lydia DEKKICHE

Encadré par :
M. **CHEBOUBA**

Année universitaire : 2019-2020

Remerciements

Avant tout, je rends grâce à DIEU de m'avoir accordé la volonté et le courage pour réaliser ce mémoire.

Au terme de ce travail je tiens tout d'abord à remercier mon promoteur M. CHEBOUBA qui m'a toujours soutenu par son aide et ses précieux conseils.

Je remercie ma famille pour son aide, sa générosité et son soutien moral qui ont été pour moi une source de courage et de confiance.

Je remercie aussi les honorables membres du jury qui ont accepté d'évaluer ce travail.

Enfin, un grand remerciement à tous ceux qui, par un mot, m'ont donné la force de continuer à travailler afin d'atteindre mes objectifs.

Résumé

Le signal électrocardiogramme (ECG) est trés largement utilisé comme l'un des outils les plus importants dans la pratique clinique afin d'évaluer l'état cardiaque des patients. Il représente les variations de l'activité électrique du cœur en fonction du temps. La classification des battements du signal ECG en différents cas pathologiques est une tâche de reconnaissance très complexe et le taux élevé de mortalité dans le monde dû aux problèmes liés au dysfonctionnement de l'appareil cardiaque a poussé les chercheurs à développer des techniques de classification automatique des maladies cardiovasculaires pour un bon diagnostic.

Dans ce mémoire, nous proposons un système pour la classification automatique des arythmies ECG en utilisant différents algorithmes de Machine Learning et de Deep Learning. Pour faire la classifcation des signaux ECG on va passer par deux étapes, la première est la classification binaire qui définit si une personne est malade ou pas. Pour cela, on utilise une base de données qui contient deux classes 1 (malade), 0 (pas malade), puis si le résultat est égale à 0 la procédure est terminé, sinon on passe à la classification multi classe, on utilise une base de données qui contient 4 classes qui correspondent à des types de maladies cardiaques. La classification multi classe va nous permettre de trouver le type de la maladie.

Mots Clés : ECG, Arythmie, Classification, Machine Learning, Deep Learning.

Abstract

The electrocardiogram (ECG) signal is widely used as one of the most common important tools in clinical practice to assess the cardiac status of patients. It represents variations in the electrical activity of the heart as a function of time. The classification of beats of the ECG signal in different pathological cases is a complex recognition task and the high mortality rate worldwide due to problems related to dysfunction of the heart system prompted researchers to develop techniques for automatically classifying cardiovascular diseases for a good diagnostic.

In this thesis, we propose a system for the automatic classification of ECG arrhythmias using different Machine Learning and Deep Learning algorithms. To classify the ECG signals we will go through two stages the first is the binary classification which defines whether a person is ill or not. For this, we use a database which contains two classes 1 (ill), 0 (not ill), then if the result is 0 the procedure is terminated, otherwise we go to the multi-class classification, we use a database which contains 4 classes that correspond to types of heart disease. The multi-class classification will allow us to find the type of the disease.

Key Words : ECG, Arrhythmia, Classification, Machine Learning, Deep Learning.

Table des matières

Table des figures

Introduction

Actuellement le domaine médical exige de nouvelles techniques et technologies, afin d'évaluer l'information d'une manière objective. Cela est dû aux développements récents dans l'électronique qui a poussé l'informatique à un stade de plus en plus avancé. Ceci a permis d'avoir des machines de plus en plus performantes permettant d'exécuter des algorithmes complexes et de tester de nouvelle approches de l'intelligence artificielle (IA) qui s'avérait impossible auparavant.

Ainsi la médecine comme plusieurs autres domaines a bénéficié de cette révolution en informatique particulièrement l'IA. Dans ce contexte, l'informatique est devenu un outil incontournable dans la pratique médicale moderne en générale et comme support d'aide au diagnostic en particulier. Plusieurs techniques de l'IA, de la logique floue (LF), réseaux de neurones (RN), les algorithmes génétiques (AG), sont couramment utilisées aux applications médicales en vue d'améliorer la performance des systèmes d'aide au diagnostic médical.

Selon l'Organisation mondiale de la santé (OMS), les maladies cardiovasculaires (MCV) sont la première cause de décès aujourd'hui. Plus de 17,7 millions de personnes sont mortes de MCV en 2017 dans le monde entier, ce qui représente environ 31% de tous les décès, et plus de 75% de ces victimes arrive dans des pays à revenu faible ou intermédiaire [1] .

Par conséquent, le diagnostic de ces maladies dangereuses semble une tâche vitale. Dans les services de cardiologie au niveau des hôpitaux, le signal électrocardiogramme (ECG) reste encore l'un des outils prédominants et les plus largement utilisés pour le diagnostic et l'analyse des arythmies cardiaques.

En réalité, l'examen ECG est un outil non invasif effectué par le médecin en vue d'explorer le fonctionnement du cœur par l'emploi des électrodes externes mises en contact de la peau. Il s'agit d'un signal qui reflète l'activité électrique du cœur.

À partir du signal ECG, certains paramètres importants peuvent être extraits. En règle générale, les durées et les formes des différentes ondes sont considérés comme des signes indicateurs de certaines anomalies cardiaques.

Cependant, la détection manuelle des ondes caractéristiques du signal ECG et la classification des battements cardiaques sont des tâches difficiles et ennuyeuses surtout pour l'analyse des enregistrements de longue durée comme dans l'examen Holter et les cas ambulatoires pour le suivi continu dans les salles de réanimation et de soins intensifs.

Par conséquence, les systèmes automatiques d'analyse du signal ECG, capables d'aider les médecins à faire le diagnostic, semble indispensable en raison du grand nombre de patients dans les unités de soins intensifs et de la nécessité d'une observation continue. C'est ainsi qu'apparu plusieurs systèmes automatiques d'aide au diagnostic cardiaque à travers le signal ECG. Ces systèmes devraient être facilement applicables, évolutifs, précis, robustes, et stables.

Pour cette raison, cette problématique a attiré beaucoup d'intérêt et de nombreuses méthodes ont été proposées par différents chercheurs pour trouver des solutions . Généralement, ces méthodes sont développées en deux grandes étapes : la caractérisation du signal ECG et le module de classification.

La première étape qui concerne la caractérisation de l'ECG peut être réalisée : Soit dans le domaine temporel afin d'obtenir des caractéristiques morphologiques (telles que la largeur, hauteur et la surface du complexe QRS, variabilité du rythme cardiaque, etc),Soit dans le domaine fréquentiel afin de trouver des changements dans les spectres de puissance du complexe QRS entre les battements normaux et anormaux.
Pour la deuxième étape qui concerne la classification, plusieurs techniques de l'IA ont été développées.

Dans le cadre de ce travail, nous proposons une approche automatisée pour la classification des arythmies ECG. Cette approche est basée sur des méthodes de Machine Learning et de Deep Learning.

Ce mémoire est organisé en 4 chapitres , le premier chapitre concerne la présentation de l'électrocardiogramme, le deuxième chapitre décrit l'apprentissage automatique et l'apprentissage profond, le troisième chapitre sur l'état de l'art qui décrit les études récentes déjà réalisées et ayant un lien avec notre problématique, puis on abordera le dernier chapitre qui est le système réalisé où on va expliquer en détails les différentes étapes et méthodes utilisées pour le traitement de la problématique.

Chapitre 1

Électrocardiogramme

1.1 Introduction :

La cardiologie est une branche de la médecine qui traite des troubles du cœur ainsi que de certaines parties du système circulatoire. Le domaine comprend le diagnostic médical et le traitement des malformations cardiaques congénitales, des maladies coronariennes, de l'insuffisance cardiaque, des cardiopathies valvulaires et de l'électrophysiologie . L'ECG (Electrocardiogramme) est l'un des outils les plus importants dans le domaine. Les cardiologues doivent étudier les ECG afin de prendre des décisions vitales concernant leurs patients. Ce qui fait de l'automatisation de l'analyse ECG un excellent outil pour la cardiologie.

1.2 Anatomie du Cœur :

1.2.1 Présentation :

Le cœur est un organe creux et musculaire comparable à une pompe, qui assure la circulation du sang dans les veines et les artères. Sa forme est similaire à un cône inversé (sa base vers le haut et à droite et son apex en bas et à gauche).

Le cœur se situe dans le médiastin, c'est la partie médiane de la cage thoracique délimitée par les deux poumons, le sternum et la colonne vertébrale. Il se trouve un peu à gauche du centre du thorax. Le cœur pèse environ 300 grammes chez l'homme adulte, 250 grammes chez la femme. Il est capable de propulser, au repos, 4 à 5 litres de sang par minute.

Le cœur est considéré comme une pompe musculaire subdivisée en quatre cavités : deux oreillettes et deux ventricules permettant de propulser le sang vers toutes les cellules du corps humain. Les paires oreillette–ventricule droite et gauche constituent respectivement les cœurs droit et gauche [2].

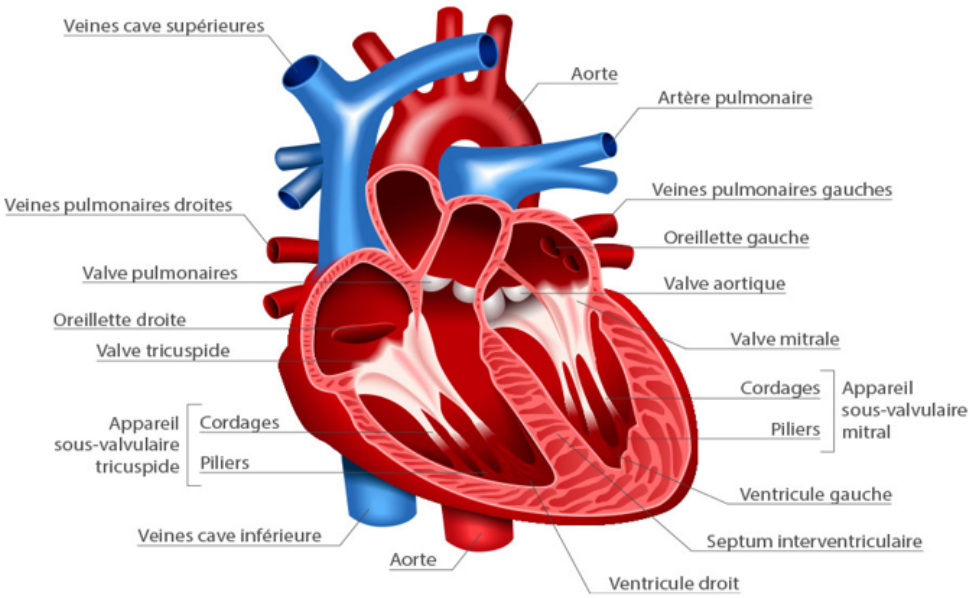


FIGURE 1.1 – Schéma général du cœur.

1.2.2 Fonctionnement du cœur :

Pour faire circuler le sang dans le corps, le cœur se contracte et se dilate. Cette action de pompage s'illustre bien par l'alternance du serrement et du desserrement d'un poing. Chaque battement, le cœur expulse du sang dans les artères. C'est ce qui crée le pouls [3].

- D'abord, l'oreillette droite se remplit du sang appauvri en oxygène provenant du corps (des muscles, des organes, du cerveau et même du cœur). Lorsque l'oreillette est pleine, elle se contracte. À la contraction, la valvule tricuspide reliant l'oreillette droite et le ventricule droit s'ouvre. Le sang entre alors dans le ventricule droit.

- Lorsque le ventricule droit est plein, il se contracte à son tour pour pousser le sang dans les poumons par la valvule pulmonaire.

- Les poumons remplacent le dioxyde de carbone présent dans le sang par de l'oxygène. Le sang, maintenant oxygéné, est expulsé vers l'oreillette gauche.

- Lorsque l'oreillette gauche se contracte, la valvule mitrale reliant cette dernière au ventricule gauche s'ouvre. Ainsi, le sang pénètre dans ce ventricule.

- Le ventricule gauche expulse le sang oxygéné par la valvule aortique vers l'aorte, qui alimente le reste du corps cœur.

- Le sang riche en oxygène circule dans tout le corps. Enfin, les veines ramènent le sang pauvre en oxygène à l'oreillette droite, qui s'en remplit, et le cycle recommence.

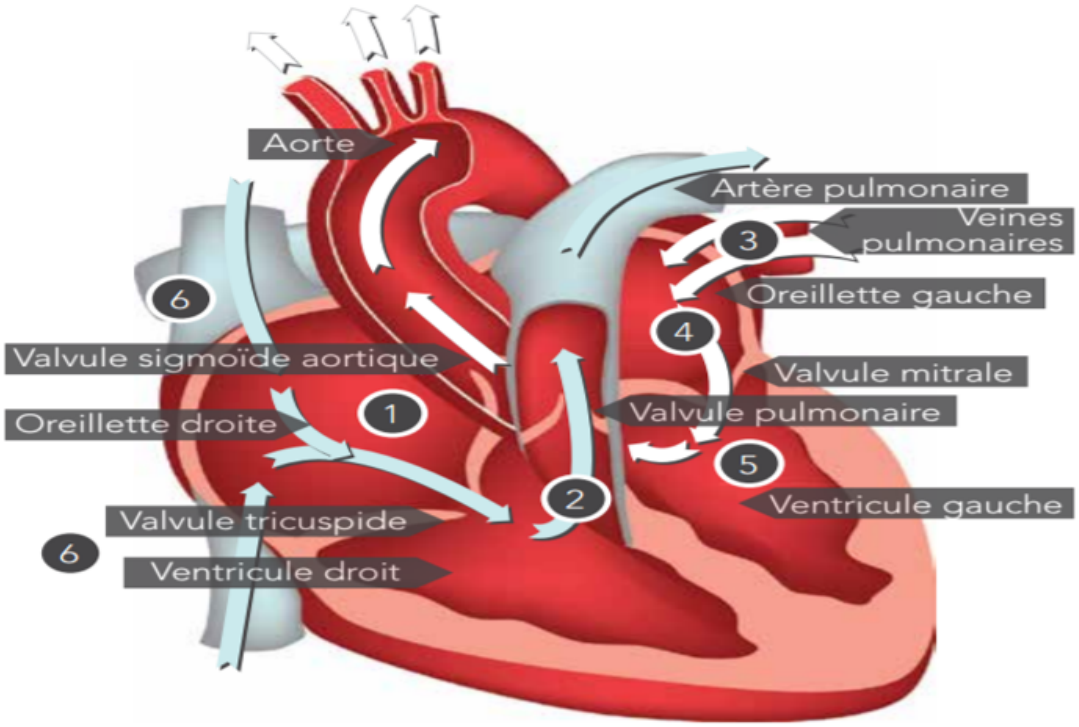


FIGURE 1.2 – Circulation sanguine dans le cœur .

1.3 L'électrocardiographie :

1.3.1 Électrocardiographe :

Un électrocardiographe est un appareil qui mesure et enregistre l'activité électrique qui traverse le cœur :

- Le tracé de cet enregistrement s'appelle un électrocardiogramme (souvent noté ECG).
- Cet appareil a été inventé par Willem Einthoven en 1895 (Prix Nobel en 1924).
- L'électrocardiogramme est aujourd'hui l'examen le plus courant pour étudier le fonctionnement du cœur et pour déceler les troubles cardiaques.

1.3.2 Électrocardiogramme :

Un électrocardiogramme (ECG) est un test qui étudie le fonctionnement du cœur en mesurant son activité électrique. À chaque battement cardiaque, une impulsion électrique (ou « onde ») traverse le cœur. Cette onde fait contracter le muscle cardiaque afin qu'il expulse le sang du cœur. Un ECG représente de ce fait un outil de mesure et enregistre l'activité électrique du cœur [4].
Un médecin peut déterminer si l'activité électrique ainsi observée est normale ou irrégulière.

Pourquoi passer un électrocardiogramme ?

L'ECG est un examen essentiel en cardiologie. Il est peut-être prescrit dans de nombreuses situations :

— À l'occasion d'un bilan médical avant une intervention chirurgicale.
— Lorsqu'un patient se plaint de palpitations ou de douleurs à la poitrine.
— En cas d'urgence (arrêt cardiaque, suspicion d'infarctus du myocarde).
— Pour détecter des arythmies (troubles du rythme cardiaque)
— Lors de la prise de certains médicaments qui peuvent agir sur le fonctionnement du cœur.
— Pour surveiller l'activité cardiaque en cas de maladie cardiaque connue.

Dans toutes ces situations, l'électrocardiogramme est anormal.

Types d'électrocardiogrammes :

L'électrocardiogramme constitue un examen indolore, sans risques. Il existe plusieurs types d'électrocardiogrammes :

- ***L'électrocardiogramme de repos*** est le cas le plus fréquent en pratique :
  — Il est réalisé au cours d'une consultation médicale lorsque le patient est allongé sur le dos.
  — Mais il peut aussi être effectué lors d'une intervention d'urgence par les pompiers ou le SAMU grâce à des appareils portatifs.

- ***L'électrocardiogramme en continu pendant un effort*** est enregistré pendant toute la durée d'un effort physique du patient (sur un vélo ou sur un tapis roulant) :

  — Cet ECG est souvent appelé le test à l'effort.
  — Cet électrocardiogramme est prescrit lorsque le patient décrit des palpitations ou des douleurs à la poitrine, mais que son électrocardiogramme de repos est normal.

- ***Le Holter-ECG*** est enregistré pendant 24 heures au cours des activités quotidiennes du patient.

  — Cet électrocardiogramme est notamment utilisé pour déterminer à quel moment de la journée les troubles cardiaques du patient sont les plus importants.

Comment se préparer à passer un électrocardiogramme ?

Aucune préparation n'est requise avant l'examen :

— Il est inutile d'être à jeun.
— Aucun produit n'est injecté ni avant, ni au cours de l'examen.
— Si le patient prend des médicaments, il n'est pas nécessaire d'interrompre le traitement, mais le patient doit absolument indiquer au médecin l'ensemble des médicaments pris. En effet, certains médicaments peuvent modifier le tracé de l'électrocardiogramme.
— Il est toutefois recommandé de s'abstenir de fumer.

Déroulement de ECG :

L'examen peut être réalisé au cabinet du cardiologue, à l'hôpital et parfois même à domicile. Pour effectuer l'électrocardiogramme, le médecin applique sur la peau une dizaine de petites électrodes (petits disques métalliques collés sur la peau grâce à des patchs) disposées au niveau des bras, des jambes et de la poitrine. Pour une meilleure adhérence des électrodes, il peut être nécessaire de raser quelques zones de la peau. Puis l'examen se déroule de la manière suivante selon le type d'ECG :

- L'enregistrement de l'électrocardiogramme de repos dure **entre 5 et 10 minutes :**

  — Il est important pendant l'enregistrement de ne pas parler, de rester calme et détendu, pour éviter de perturber le tracé.
  — Le médecin peut demander au patient de retenir son souffle à certains moments de l'enregistrement.

- L'électrocardiogramme en continu pendant un effort dure **entre 10 et 30 minutes :**

  — L'intensité de l'effort demandé au patient est augmentée par paliers.
  — L'examen dure jusqu'à ce que le patient signale l'apparition de la fatigue au médecin.
  — Le patient ne doit pas parler pendant l'enregistrement, sauf pour signaler tout symptôme anormal.
  — L'enregistrement est suivi par une période de récupération pendant laquelle le patient reste sous la surveillance du médecin.

- Le Holter-ECG dure **au minimum 24 h :**

  — Un petit appareil portatif d'enregistrement de l'ECG est relié aux électrodes et accompagne le patient dans ses activités quotidiennes.
  — Le patient doit noter les moments où il ressent des symptômes particuliers pour faciliter l'interprétation des résultats de l'examen.

1.3.3 Les ondes du signal ECG :

le signal ECG est représenté par des lettres d'alphabet « P, Q, R, S et T », ils correspondent à l'activation électrique des diverses parties du cœur qui est formé par plusieurs ondes, le signal représente la morphologie et l'amplitude de ces ondes mais la durée change selon les dérivations ECG utilisées :

- ***Onde P :*** C'est la première onde détectable, qui représente la dépolarisation auriculaire. Il s'agit d'une petite déviation positive (ou négative) avant le complexe QRS, qui se propage à partir du nœud SA et est conduit dans toutes les cellules de l'oreillette par des jonctions intermédiaires qui connectent ces cellules.

- ***Le complexe QRS :*** C'est un ensemble de déflexions positives et négatives qui correspondent à la contraction des ventricules. Pour un cas normal, il a une durée inférieure à 0.12 seconde et son amplitude variable est comprise entre 5 et 20 mV.

- ***L'onde Q :*** Est la première déviation initiale vers le bas ou « négative ».

- ***L'onde R :*** Est alors la déviation suivante vers le haut (à condition qu'elle croise la ligne isoélectrique et devienne « positive »).

- ***L'onde S :*** Est alors la déviation suivante vers le bas, à condition qu'elle traverse la ligne isoélectrique pour devenir brièvement négative avant de revenir à la ligne de base isoélectrique.

- ***L'onde T :*** Elle correspond à la repolarisation des ventricules. L'onde T normale à une amplitude plus faible que le complexe QRS [5].

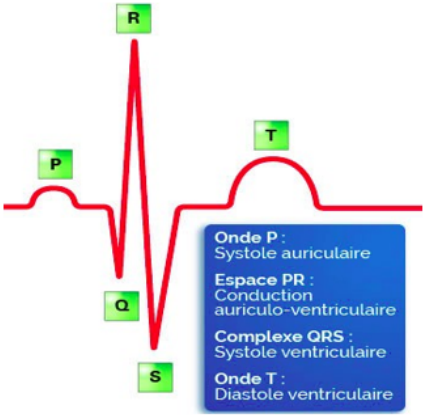


FIGURE 1.3 – Les ondes du signal ECG.

1.3.4 Les segments et intervalles qui caractérisent un ECG normal :

En plus des différentes ondes qui sont les paramètres de base pour une bonne caractérisation d'un signal ECG, il existe un certain nombre d'intervalles et de segments qui portent des informations très utiles sur la vitesse de conduction de l'impulsion électrique dans les différentes parties du cœur.
Les intervalles et les segments les plus importants sont :

- ***Intervalle RR :*** L'intervalle RR correspond au délai entre deux dépolarisations des ventricules. C'est cet intervalle qui permet de calculer la fréquence cardiaque.

- ***Segment PR :*** (pause du nœud AV) Le segment PR correspond au délai entre la fin de la dépolarisation des oreillettes et le début de celle des ventricules. C'est le temps pendant lequel l'onde de dépolarisation est bloquée au niveau du nœud AV [6].

- ***Intervalle PR :*** (durée de conduction auriculo-ventriculaire) L'intervalle PR correspond à la durée de propagation de l'onde de dépolarisation du nœud sinusal jusqu'aux cellules myocardiques ventriculaires.

- ***Intervalle QT :*** (durée de systole ventriculaire) Cet intervalle correspond au temps de systole ventriculaire, qui va du début de l'excitation des ventricules jusqu'à la fin de leur relaxation.

- ***Segment ST :*** (durée de stimulation complète des ventricules) Le segment ST correspond à la phase pendant laquelle les cellules ventriculaires sont toutes dépolarisées, le segment est alors isoélectrique.

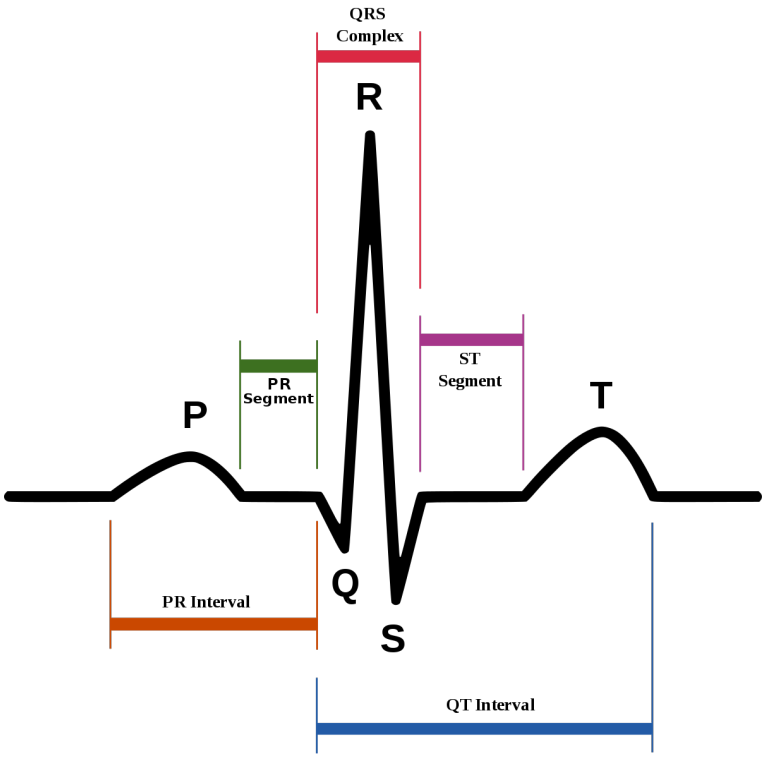


FIGURE 1.4 – Segments et intervalles d'un ECG normal.

1.3.5 Différentes pathologies cardiaques :

Notre objectif dans cette partie n'est pas d'analyser précisément les origines des pathologies cardiaques et leurs conséquences sur le fonctionnement cardiaque ni de décrire les traitements que ces pathologies nécessitent, mais simplement de mettre en relation certaines observations anormales du tracé ECG avec ces pathologies . Parmi les maladies les plus rencontrées, ceux qui affectent le rythme cardiaque et qui sont appelées les arythmies cardiaques. Mais avant de parler des arythmies, il est intéressant de connaître les caractéristiques du rythme normale appelé aussi rythme sinusal.

- **Rythme sinusal :** Le rythme sinusal est le rythme normal cardiaque. Il correspond à une activation physiologique des oreillettes, puis des ventricules, à partir du nœud sinusal. Son rythme est compris entre 60 à 80 battements par minute avec un intervalle régulier entre des battements normaux.

- **Les arythmies cardiaques :** L'arythmie est une anomalie qui affecte la fréquence cardiaque normale. Plusieurs types d'arythmie ne présentent aucun problème de santé ; cependant, elles peuvent causer divers symptômes gênants, comme des étourdissements ou une douleur dans la poitrine. D'autres formes d'arythmies, plus dangereuses, ont des répercussions sur l'apport sanguin et nécessitent de ce fait une prise en charge médicale.
  Dans notre projet, on va utiliser 4 types d'arythmies :

  — ***Contraction ventriculaire prématurée (PVC) :*** Sont des battements cardiaques anormaux qui proviennent des ventricules du cœur. Elles sont causées par une irritabilité électrique du muscle cardiaque des ventricules et perturbent le rythme normal du cœur, provoquant une bascule ou un battement sauté dans la poitrine. Les contractions ventriculaires prématurées sont très fréquentes et sont susceptibles d'apparaître chez les personnes âgées, les personnes hypertendues et les personnes souffrant de maladies cardiaques.

  — ***Contraction auriculaire prématurée (APC)*** : Est une dépolarisation prématurée prenant naissance en un point quelconque des oreillettes, habituellement mais non constamment propagée aux ventricules. La morphologie de l'onde P extrasystolique est différente de celle de l'onde P sinusale et dépend de la situation du foyer auriculaire ectopique.

  — ***Bloc de branche droit (RBB) :*** Lorsqu'il existe un bloc de branche droit, le ventricule droit n'est pas directement activé par les impulsions électriques progressant au travers la branche droite du faisceau de His. Le ventricule gauche, quant à lui, est normalement activé par la branche gauche. Ces impulsions électriques progressent alors au travers du myocarde, du ventricule gauche vers le ventricule droit, activant finalement celui-ci. La prévalence du bloc de branche droit augmente avec l'âge.

  — ***Battement du rythme (Paced Beat : PAB) :*** Est en général la manifestation d'un dysfonctionnement du muscle cardiaque. Il peut apparaître à tout âge, mais il est fréquemment la conséquence d'un infarctus du myocarde. Il se manifeste par la désynchronisation des battements qui conduit à une diminution du pompage du sang [7].

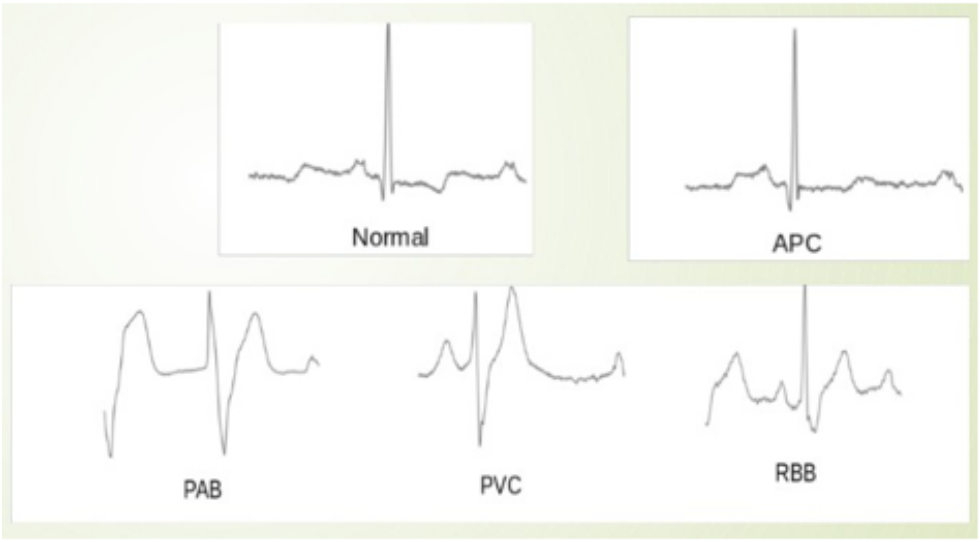


FIGURE 1.5 – Les types des résultats ECG (1 normal, 4 anormaux).

1.4 Conclusion :

Dans ce chapitre, nous avons parlé de l'électrocardiogramme. Nous avons commencé par donner une brève introduction de l'anatomie du cœur, puis présenter l'électrocardiogramme, ses différents types, son déroulement et les différentes ondes et les différents segments et intervalles qui le compose. Enfin, nous avons décrit quelques arythmies cardiaques.

Chapitre 2

Machine Learning et Deep Learning

2.1 Introduction :

Avant d'arriver à ce qu'est l'apprentissage automatique (ML : Machine Learning) et l'apprentissage profond (DL : Deep Learning), nous devons introduire le concept d'intelligence artificielle.

L'Intelligence Artificielle (IA) est un vaste domaine, où nous essayons d'imiter le comportement humain dans le but de rendre les machines si puissantes pour accomplir de nombreux types de tâches telles que la résolution de problèmes, la représentation des connaissances, la reconnaissance vocale, et autres. L'idée de base est de mettre les connaissances dans la machine.

Le Machine learning et le Deep learning font partie de l'intelligence artificielle. Ces approches ont toutes deux pour résultat de donner aux ordinateurs la capacité de prendre des décisions intelligentes.

La relation entre les trois concepts IA, ML et DL est résumée par les auteurs dans la figure suivante :

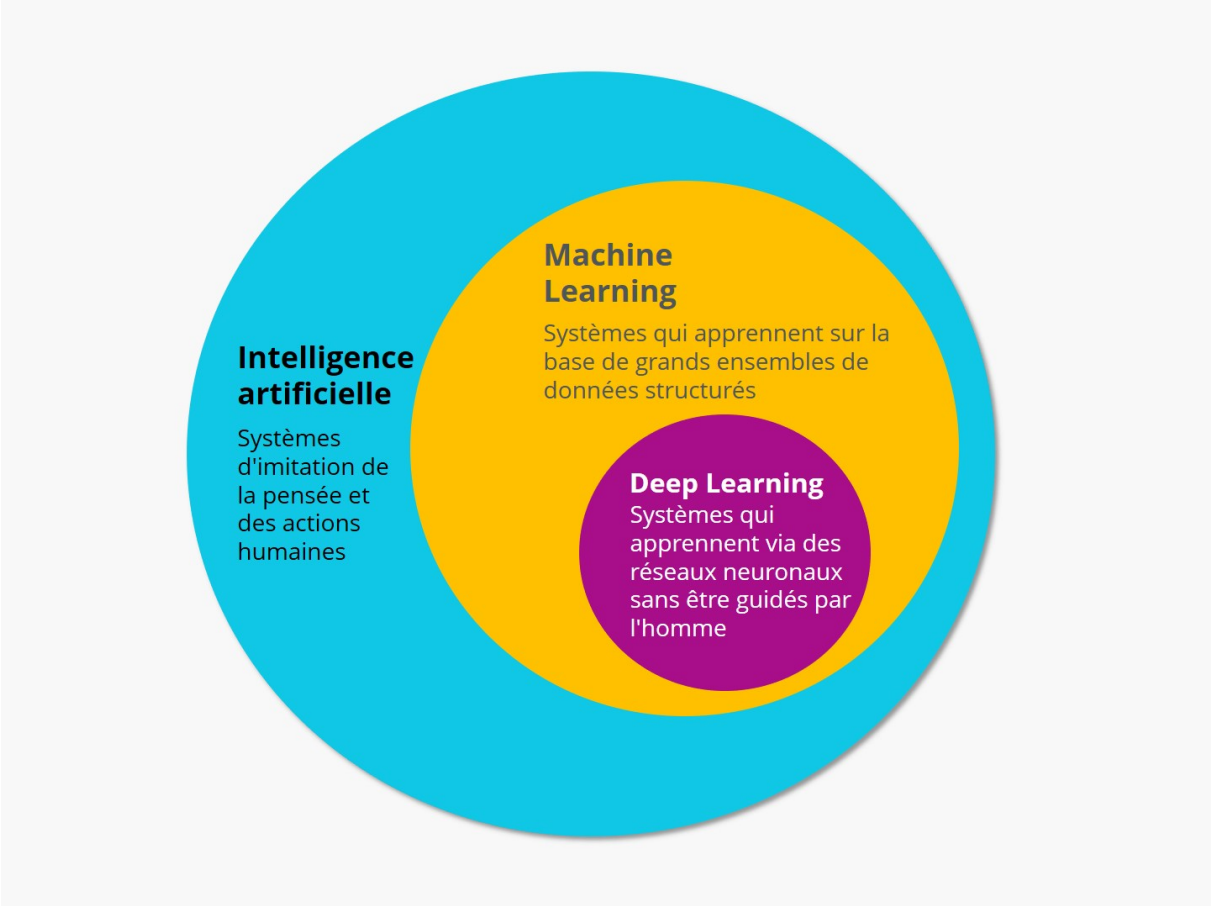


FIGURE 2.1 – La relation entre IA,ML et DL

2.2 Machine Learning

2.2.1 Définition :

L'apprentissage machine (ou apprentissage automatique, Machine Learning en anglais) est un sous-domaine de l'intelligence artificielle, qui donne à un système une capacité de compréhension grâce à ses algorithmes. Il est basé sur l'idée de faire apprendre des algorithmes à partir de données et de faire des prédictions avec ces données et par cela les ordinateurs apprennent à résoudre des tâches spécifiques, sans avoir besoin de les programmer.

L'objectif du Machine Learning est de reconnaître parmi des données des structures souvent trop difficiles à détecter ou à mesurer manuellement. À partir de ces structures, on peut chercher à classifier des individus, des objets, à prédire la valeur d'une variable à un certain horizon, à expliquer l'apparition ou non d'une caractéristique.

2.2.2 Les différents types d'apprentissage :

- **Apprentissage Supervisé** :

  L'apprentissage supervisé (supervised learning en anglais) est certainement la forme de machine learning la plus répandue. Elle consiste à entraîner le programme sur des exemples dont on connaît la catégorie (c'est à cette connaissance qu'elle doit le nom "supervisé").

  Le programme a une base de données avec des classes connues. Il aura pour objectif de ranger une/des valeur(s) qui lui sont attribuée(s) dans les classes définies. L'algorithme va classer les nouvelles valeurs attribuées par l'utilisateur. Quelle que soit la valeur, elle sera attribuée à l'une des classes existantes par l'algorithme. Cet algorithme n'ayant pas été développé pour créer de nouvelles classes, il s'avère très utile pour des tâches de classifications ciblées.

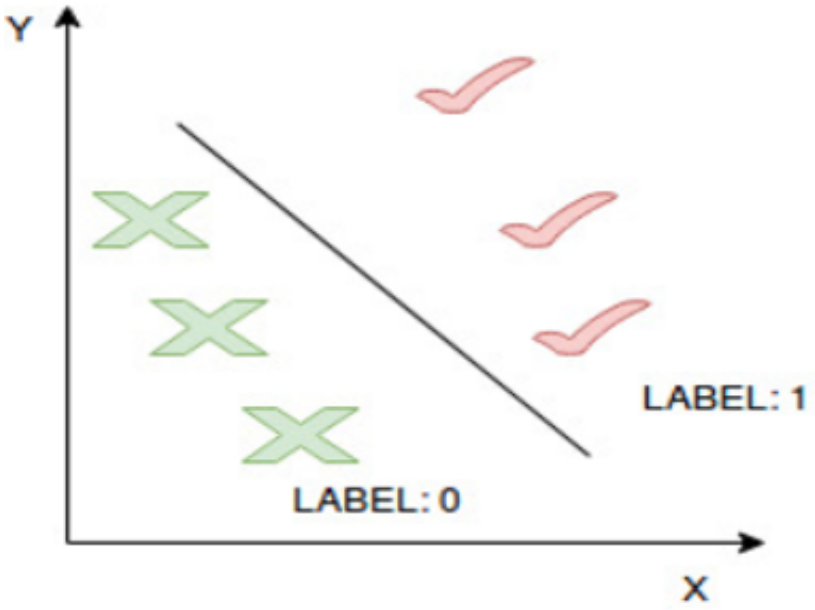


FIGURE 2.2 – L'apprentissage supervisé

- **Apprentissage non supervisé** :

  Dans ce cas, le programme ne reçoit que des valeurs et doit créer les classes dans lesquelles les attribuer. Il va donc "décider" lui-même le nombre de classes à créer pour ensuite ranger les données dans chaque classe. Ces algorithmes sont utilisés lorsque nous n'avons pas d'échantillon à disposition [8].
  (À noter que l'on peut faire cette méthode pour se constituer un échantillon puis utiliser l'apprentissage supervisé pour classer de nouvelles données).

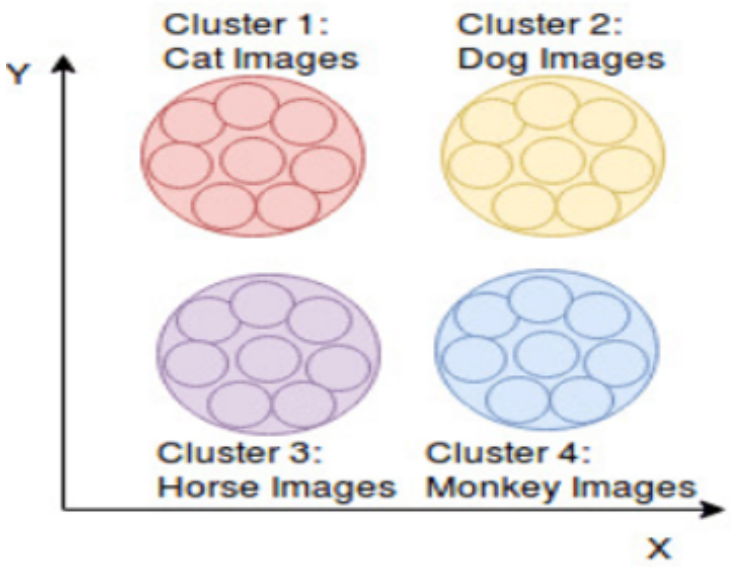


FIGURE 2.3 – L'apprentissage non supervisé

- **Apprentissage par renforcement** :

  Ce dernier type d'apprentissage automatique a pour moteur le renforcement. C'est-à-dire que la machine a la capacité d'apprendre sur la base de différents tests et erreurs donnés dans différents contextes. Que vous connaissiez ou pas les résultats dès le départ, la machine ne connaît toujours pas les meilleures décisions à prendre pour obtenir les bons résultats. Dans ce cas, l'algorithme associe, seul, les schémas de réussite. Et c'est leur répétition qui permet à l'algorithme d'obtenir constamment des résultats exacts et infaillibles.

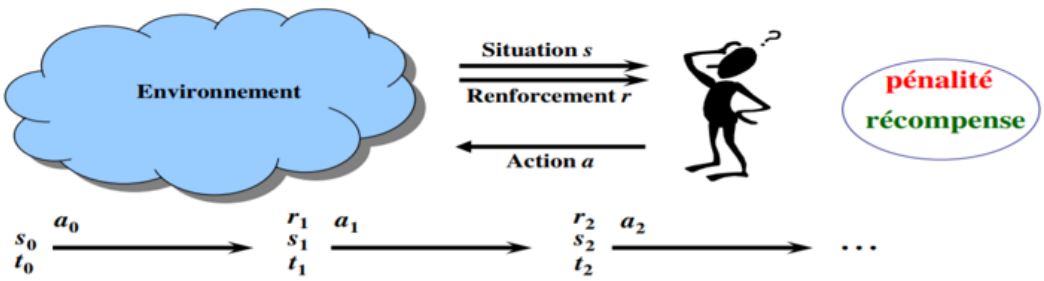


FIGURE 2.4 – L'apprentissage par renforcement

2.2.3 Domaines d'application de l'apprentissage automatique :

- ***Automatisation :*** L'apprentissage automatique, qui fonctionne de manière entièrement autonome dans n'importe quel domaine sans aucune intervention humaine. Par exemple, des robots exécutant les étapes essentielles du processus dans les usines de fabrication.

- ***Industrie de la finance :*** L'apprentissage automatique gagne en popularité dans le secteur financier. Les banques utilisent principalement le ML pour trouver des modèles dans les données mais aussi pour prévenir la fraude.

- ***Organisation gouvernementale :*** Le gouvernement utilise le ML pour gérer la sécurité publique et les services publics. Prenons l'exemple de la Chine avec la reconnaissance faciale massive. Le gouvernement utilise l'intelligence artificielle pour empêcher le jaywalker.

- ***L'industrie de la santé :*** La santé a été l'une des premières industries à utiliser l'apprentissage automatique avec détection d'images.

- ***Transports :*** Dans l'industrie des transports, les données sont analysées pour identifier des patterns et des tendances. Ainsi, les itinéraires sont plus efficients et les problèmes potentiels peuvent être prédits pour augmenter la rentabilité. L'analyse de données et les modèles du Machine Learning sont utilisés comme de précieux outils par les entreprises de livraison, les transports publics et les autres entreprises de transport.

2.2.4 Algorithmes de Machine Learning :

- **La régression logistique**

  La régression logistique est un algorithme de classification d'apprentissage supervisé utilisé pour prédire la probabilité d'une variable cible. La nature de la variable cible ou dépendante est dichotomique, ce qui signifie qu'il n'y aurait que deux classes possibles. En termes simples, la variable dépendante est de nature binaire, les données étant codées soit 1 (signifie succès / oui) ou 0 (signifie échec / non) [9] .

  Mathématiquement, un modèle de régression logistique prédit P (Y) en fonction de X. C'est l'un des algorithmes ML les plus simples qui peut être utilisé pour divers problèmes de classification tels que la détection de spam, la détection de maladies, etc...

  Généralement, la régression logistique signifie une régression logistique binaire ayant des variables cibles binaires, mais il peut y avoir deux autres catégories de variables cibles qui peuvent être prédites par elle. Sur la base de ce nombre de catégories, la régression logistique peut être divisée en types suivants :

— ***Régression logistique binaire :***
Dans un tel type de classification, une variable dépendante n'aura que deux types possibles, soit 1 et 0. Par exemple, ces variables peuvent représenter un succès ou un échec, oui ou non, une victoire ou une perte, etc.

— ***Régression logistique multinomiale :***
Dans un tel type de classification, la variable dépendante peut avoir 3 types ou plus, ou les types n'ayant aucune signification quantitative. Par exemple, ces variables peuvent représenter « Type A » ou « Type B » ou « Type C », dans notre cas c'est le type de la maladie.

— ***Régression logistique ordinale :***
Dans ce genre de classification, la variable dépendante peut avoir 3 types possible ou plus, ou les types ayant une signification quantitative. Par exemple, ces variables peuvent représenter « mauvais » ou « bon », « très bon », « excellent » et chaque catégorie peut avoir des scores tels que 0,1,2,3.

- **Random Forest :**

Random forest est un algorithme d'apprentissage supervisé. Il peut être utilisé à la fois pour la classification et la régression. C'est également l'algorithme le plus flexible et le plus facile à utiliser. Une forêt est composée d'arbres. On dit que plus il y a d'arbres, plus la forêt est robuste.

Les forêts aléatoires créent des arbres de décision sur des échantillons de données sélectionnés au hasard, obtiennent des prédictions à partir de chaque arbre et sélectionnent la meilleure solution par vote. Il fournit également un assez bon indicateur de l'importance des fonctionnalités [10].

Les forêts aléatoires ont une variété d'applications, telles que les moteurs de recommandation, la classification d'images et la sélection de caractéristiques. Il peut être utilisé pour classer les demandeurs de prêt fidèles, identifier les activités frauduleuses et prédire les maladies.

Nous pouvons comprendre le fonctionnement de l'algorithme Random Forest à l'aide des étapes suivantes :

— ***Étape 1 :*** Commencez par sélectionner des échantillons aléatoires à partir d'un ensemble de données.

— ***Étape 2 :*** Ensuite, cet algorithme construira un arbre de décision pour chaque échantillon. Ensuite, il obtiendra le résultat de la prédiction de chaque arbre de décision.

— ***Étape 3 :*** Dans cette étape, le vote sera effectué pour chaque résultat prévu.

— ***Étape 4 :*** Enfin, sélectionnez le résultat de prédiction le plus voté comme résultat de prédiction final.

Le schéma suivant illustre son fonctionnement :

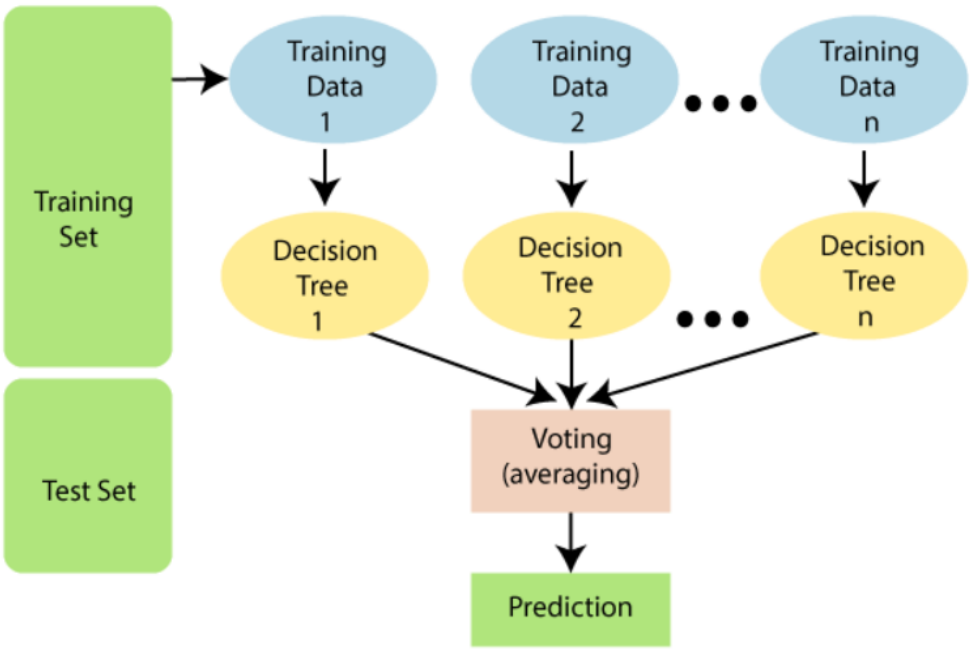


FIGURE 2.5 – Fonctionnement de Random Forest

De manière générale, la modélisation random forest est utilisé avec la librairie scikit learn sur python. On retrouve des paramètres comme :

- ***n-estimators :*** Le nombre d'arbres dans la forêt.
- ***max-depth :*** La profondeur maximale de l'arbre
- ***min-samples-split :*** Le nombre minimum d'échantillons requis pour diviser un nœud interne
- ***max-features :*** Il s'agit du nombre maximum de fonctionnalités que Random Forest est autorisé à essayer dans un arbre individuel.

- **Les Machines à Vecteurs de Support (SVM) :**

Les SVM sont des algorithmes d'apprentissage automatique supervisé puissants qui sont utilisés à la fois pour la classification et la régression. Mais généralement, ils sont utilisés dans les problèmes de classification. Dans les années 1960, les SVM ont été introduits pour la première fois, mais ils ont ensuite été affinés en 1990 [11]. Dernièrement, ils connaissent un très grand succès, pour de multiples raisons :

- Elles peuvent travailler avec des données disposant d'un très grand nombre de paramètres.
- Elles utilisent peu d'hyperparamètres
- Elles garantissent de bons résultats théoriques
- Elles peuvent égaler ou dépasser en performance les réseaux de neurones ou modèles gaussiens.

Les éléments suivants sont des concepts importants dans SVM :

— ***Vecteurs de support*** : Les points de données les plus proches de l'hyperplan sont appelés vecteurs de support. La ligne de séparation sera définie à l'aide de ces points de données.

— ***Hyperplan :*** Un hyperplan est un plan de décision qui sépare un ensemble d'objets ayant différentes appartenances à une classe.

— ***Marge :*** Une marge est un espace entre les deux lignes sur les points de classe les plus proches. Ceci est calculé comme la distance perpendiculaire à la ligne pour prendre en charge les vecteurs ou les points les plus proches. Si la marge est plus grande entre les classes, alors elle est considérée comme une bonne marge, une plus petite marge est une mauvaise marge.

L'objectif principal est de séparer le jeu de données de la meilleure façon possible. La distance entre les deux points les plus proches est appelée marge. L'objectif est de sélectionner un hyperplan avec la marge maximale possible entre les vecteurs du support.

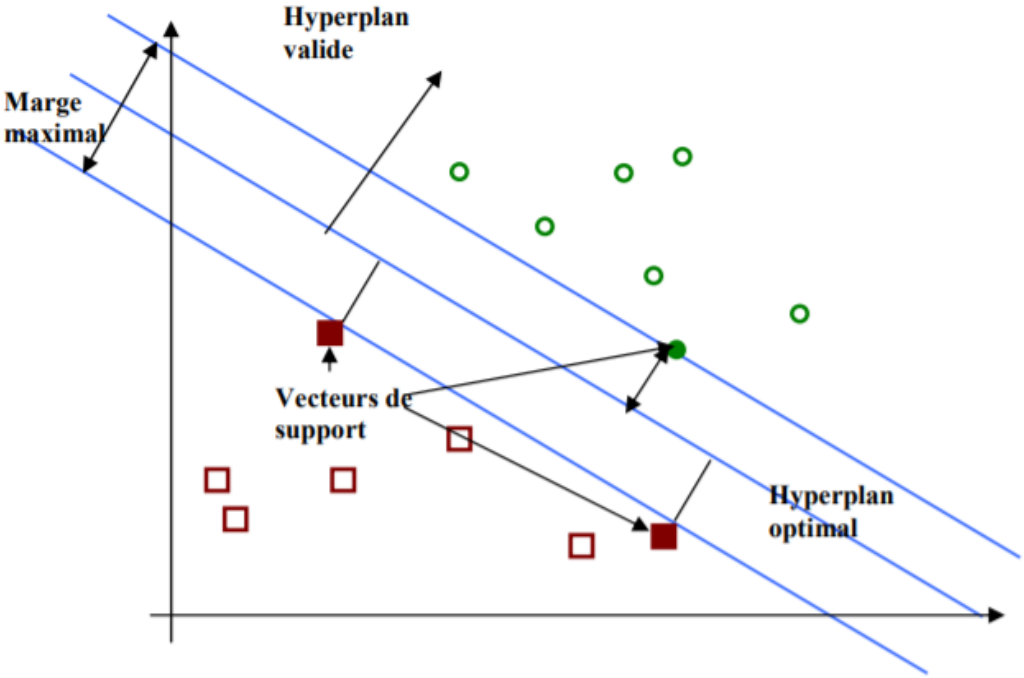


FIGURE 2.6 – Exemple de marge maximal (hyperplan valide).

2.3 Deep Learning

2.3.1 Définition :

Le Deep Learning ou apprentissage profond est un sous-domaine de l'apprentissage automatique (et un sous-sous-domaine de l'IA en général) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. Semblable à l'apprentissage automatique, l'apprentissage profond contient également un apprentissage supervisé, non supervisé et par renforcement.

Dans [12], un article représente un aperçu du contexte historique de DL, il présente les étapes majeures qui mènent à ce que nous avons maintenant. Ces étapes sont résumées dans le tableau suivant :

L'année	Contributeur	Contribution
300 AC	Aristotle	-Introduction de l'associationnisme, début de l'histoire des humains qui essayent de comprendre le cerveau
1873	Alexander Bain	-Introduction du Neural Groupings comme les premiers modèles de réseaux de neurones
1943	McCulloch et Pitts	-Introduction du McCulloch Pitts (MCP) modèle considérer comme L'ancêtre des réseaux de neurones artificielles
1949	Donald Hebb	-Considérer comme le père des réseaux de neurones, il introduit la règle d'apprentissage de Hebb qui servira de fondation pour les réseaux de neurones modernes.
1958	Frank Rosenblatt	-Introduction du premier perceptron
1974	Paul Werbos	-Introduction de la retro propagation
1980	Teuvo Kohonen	-Introduction des cartes auto organisatrices
1980	Kunihiko Fukushima	-Introduction du Neocognitron, qui a inspiré les réseaux de neurones convolutif
1982	John Hopfield	-Introduction des réseaux de Hopfield
1985	Hilton et Sejnowski	-Introduction des machines de Boltzmann

1986	Paul Smolensky	-Introduction de Harmonium, qui sera connu plus tard comme machines de Boltzmann restreintes
1986 1990	Michael I. Jordan Yann LeCun	-Définition et introduction des réseaux de neurones récurrent -Introduction de LeNet et montra la capacités des réseaux de neurones profond
1997	Schuster et Paliwal	-Introduction des réseaux de neurones récurrent bidirectionnelles
1997	Hochreiter et Schmidhuber	-Introduction de LSTM, qui ont résolu le problème du vanishing gradient dans les réseaux de neurones récurrent
2006	Geoffrey Hinton	-Introduction des Deep Belief Network
2009	Salakhutdinov et Hinton	-Introduction des Deep Boltzmann Machines
2012	Geoffrey Hinton	-Introduction de AlexNet qui remporta le challenge ImageNet

FIGURE 2.7 – Le résumé de l'histoire de deep learning [12].

2.3.2 Applications du Deep Learning :

- ***La reconnaissance faciale :*** Les yeux, le nez, la bouche, tout autant de caractéristiques qu'un algorithme de DL va apprendre à détecter sur une photo. Il va s'agir en premier lieu de donner un certain nombre d'images à l'algorithme, puis à force d'entraînement, l'algorithme va être en mesure de détecter un visage sur une image.

- ***Le traitement automatique de langage naturel :*** Le traitement automatique de langage naturel est une autre application du DL. Son but étant d'extraire le sens des mots, voire des phrases pour faire de l'analyse de sentiments. L'algorithme va par exemple comprendre ce qui est dit dans un avis Google, ou va communiquer avec des personnes via des chatbots. La lecture et l'analyse automatique de textes est aussi un des champs d'application du DL avec le Topic Modeling : tel texte aborde tel sujet.

- ***Voitures autonomes :*** Les entreprises qui construisent de tels types de services d'aide à la conduite, ainsi que des voitures autonomes telles que Google, doivent apprendre à un ordinateur à maîtriser certaines parties essentielles de la conduite à l'aide de systèmes de capteurs numériques au lieu de l'esprit humain. Pour ce faire, les entreprises commencent généralement par entraîner des algorithmes utilisant une grande quantité de données. Vous pouvez imaginer comment un enfant apprend grâce à des expériences constantes et à la réplication. Ces nouveaux services pourraient fournir des modèles commerciaux inattendus aux entreprises.

- ***Recherche vocale et assistants à commande vocale :*** L'un des domaines d'utilisation les plus populaires de DL est la recherche vocale et les assistants intelligents à commande vocale. Avec les grands géants de la technologie ont déjà fait d'importants investissements dans ce domaine, des assistants à commande vocale peuvent être trouvés sur presque tous les smartphones. Le Siri d'Apple est sur le marché depuis octobre 2011. Google aujourd'hui, l'assistant à commande vocale pour Android, a été lancé moins d'une année après Siri. Le plus récent des assistants intelligents à commande vocale est Microsoft Cortana.

- ***Reconnaissance d'image :*** Un autre domaine populaire en matière de DL est la reconnaissance d'image. Son objectif est de reconnaître et d'identifier les personnes et les objets dans les images, ainsi que de comprendre le contenu et le contexte. La reconnaissance d'image est déjà utilisée dans plusieurs secteurs tels que les jeux, les médias sociaux, la vente au détail, le tourisme, etc. Cette tâche nécessite la classification des objets d'une photo parmi un ensemble d'objets connus auparavant. Une variante plus complexe de cette tâche, appelée détection d'objet, consiste à identifier spécifiquement un ou plusieurs objets dans la scène de la photo et à dessiner un cadre autour d'eux.

- ***La détection du cancer du cerveau :*** Une équipe de chercheurs français a noté qu'il était difficile de détecter les cellules cancéreuses du cerveau invasives au cours d'une intervention chirurgicale, en partie à cause des effets de l'éclairage dans les salles d'opération. Ils ont découvert que l'utilisation de réseaux de neurones conjointement avec la spectroscopie Raman pendant les opérations leur permettait de détecter les cellules cancéreuses plus facilement et de réduire le cancer résiduel après l'opération.

- ***Analyse des sentiments du texte :*** De nombreuses applications ont des commentaires ou des systèmes de révision basés sur des commentaires intégrés à leurs applications. La recherche sur le traitement du langage naturel et les réseaux de neurones récurrents ont parcouru un long chemin et il est maintenant tout à fait possible de déployer ces modèles sur le texte de votre application pour extraire des informations de niveau supérieur. Cela peut être très utile pour évaluer la polarité sentimentale dans les sections de commentaires ou pour extraire des sujets significatifs à l'aide de modèles de reconnaissance d'entités nommées.

- ***Recherche en marketing :*** En plus de rechercher de nouvelles fonctionnalités susceptibles d'améliorer votre application, DL peut également être utile en arrière-plan. La segmentation du marché, l'analyse des campagnes marketing et bien d'autres peuvent être améliorés à l'aide de modèles de régression et de classification DL. Cela vous aidera vraiment beaucoup si vous avez une grande quantité de données. Sinon, vous ferez probablement mieux d'utiliser des algorithmes traditionnels d'apprentissage automatique pour ces tâches plutôt que DL [13].

2.3.3 Le réseau neuronal :

La pratique, de tous les algorithmes de DL sont des réseaux neuronaux [14]. Les réseaux neuronaux, aussi appelés ANN, sont des modèles de traitement de l'information qui simulent le fonctionnement d'un système nerveux biologique. C'est similaire à la façon dont le cerveau manipule l'information au niveau du fonctionnement. Tous les réseaux neuronaux sont constitués de neurones inter connectés qui sont organisés en couches.

Le neurone :

Ce qui forme les réseaux de neurones, ce sont les neurones artificiels inspirés du vrai neurone qui existe dans notre cerveau. Les 2 figures suivantes montrent une représentation d'un neurone réel et d'un neurone artificiel :

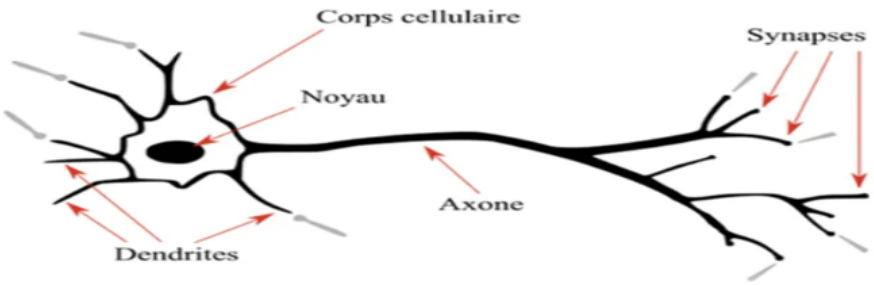


FIGURE 2.8 – Un neurone réel.

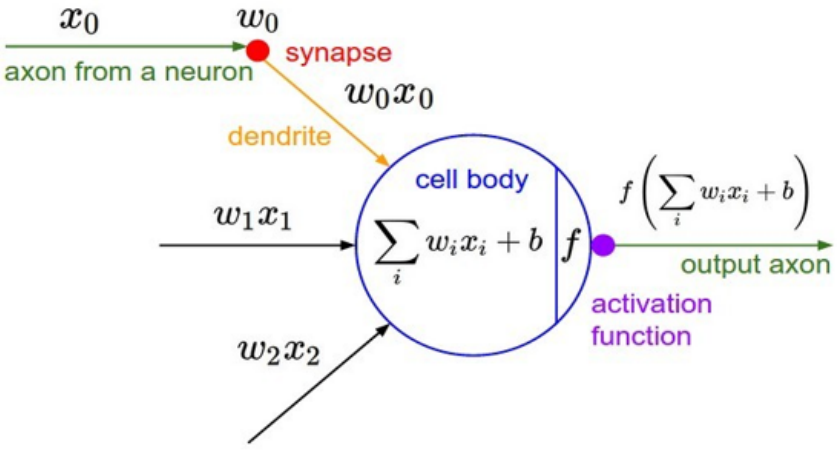


FIGURE 2.9 – Un neurone artificiel.

Les *Xi* sont des valeurs numériques qui représentent soit les données d'entrée, soit les valeurs sorties d'autres neurones.
Les poids *Wi* sont des valeurs numériques qui représentent soit la valeur de puissance des entrées, soit la valeur de puissance des connexions entre les neurones.
Il existe des opérations qui se passent au niveau du neurone artificiel. Le neurone artificiel fera un produit entre le poids (w) et la valeur d'entrée (x), puis ajouter un biais (b), le résultat est transmis à une fonction d'activation (f) qui ajoutera une certaine non-linéarité.

Les fonctions d'activation :

Après que le neurone a effectué le produit entre ses entrées et ses poids, il applique également une non-linéarité sur ce résultat. Cette fonction non linéaire s'appelle la fonction d'activation. La fonction d'activation est une composante essentielle du réseau neuronal. Ce que cette fonction a décidé est si le neurone est activé ou non. Il calcule la somme pondérée des entrées et ajoute le biais. C'est une transformation non linéaire de la valeur d'entrée.

Après la transformation, cette sortie est envoyée à la couche suivante. La non-linéarité est si importante dans les réseaux de neurones, sans la fonction d'activation, un réseau de neurones est devenu simplement un modèle linéaire. Il existe de nombreux types de ces fonctions, parmi lesquelles nous trouvons :

- **La fonction Sigmoïde :** Cette fonction est l'une des plus couramment utilisées. Il est borné entre 0 et 1, et il peut être interprété stochastiquement comme la probabilité que le neurone s'active, et il est généralement appelé la fonction logistique ou le sigmoïde logistique.

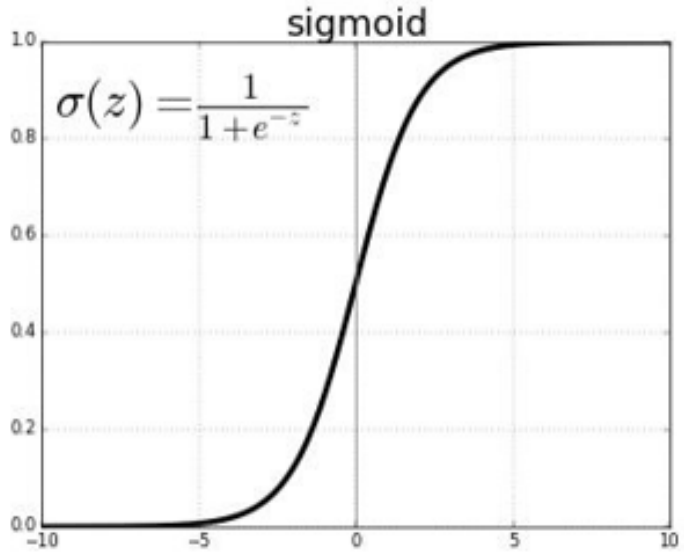


FIGURE 2.10 – Représentation graphique de la fonction Sigmoïde

- **La fonction ReLu :** La fonction RELU est probablement la plus proche de sa correspondante biologique [15]. Cette fonction est récemment devenue le choix de nombreuses tâches (notamment en computer vision) [16]. Comme dans la formule ci-dessus, cette fonction renvoie 0 si l'entrée z est inférieure à 0 et retourne z lui-même s'il est plus grand que 0.

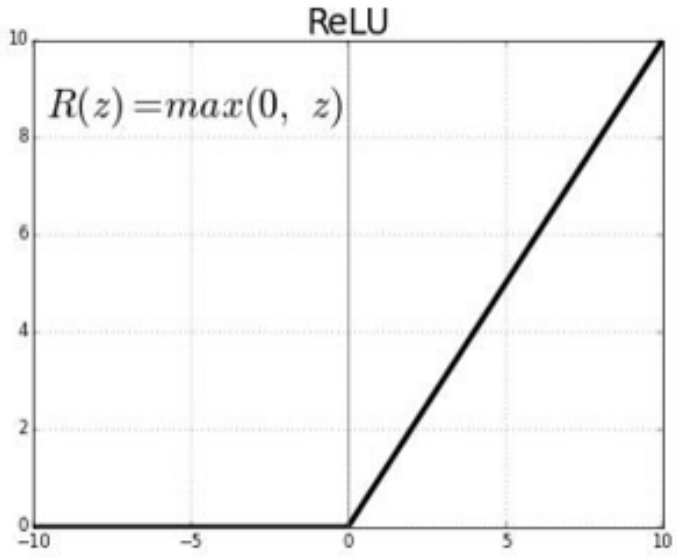


FIGURE 2.11 – Représentation graphique de la fonction ReLu

- **La fonction Softmax :** Régression Softmax (synonymes : Logistique multinomiale, Classificateur d'entropie maximale, ou simplement Régression logistique multi-classe) est une généralisation de la régression logistique que nous pouvons utiliser pour la classification multi-classes [17]. Contrairement à d'autres types de fonctions, la sortie d'un neurone d'une couche utilisant la fonction softamx dépend des sorties de tous les autres neurones de sa couche. Cela s'explique par le fait qu'il nécessite que la somme de toutes les sorties soit égale à 1.

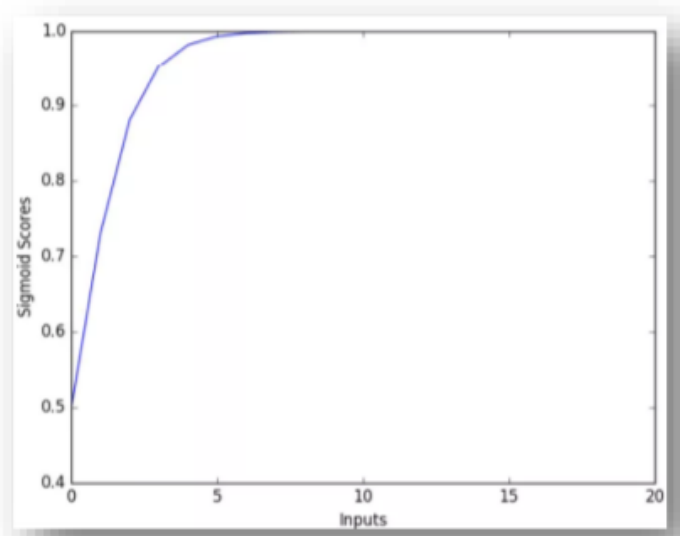


FIGURE 2.12 – Représentation graphique de la fonction Softmax.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ pour tout } j \in \{1, \ldots, K\}.$$

Les types des réseaux de neurones :

Il existe différents types de réseaux de neurones. Nous pouvons ainsi citer :

- **Perceptron multicouche (Multi Layer Perceptron : MLP) :**

  Un perceptron multicouche (MLP) est une classe de réseau neuronal artificiel à action directe. Un MLP se compose d'au moins trois couches de nœuds : couche d'entrée, couche cachée et couche de sortie.

  — ***La couche d'entrée (input layer) :*** Un ensemble de neurones qui portent le signal d'entrée. Tous les neurones de cette couche sont ensuite reliés à ceux de la couche suivante.

  — ***La couche cachée (hidden layer) :*** Ou plus souvent les couches cachés (couche cachée 1, couche cachée 2, ...). Il s'agit du cœur de perceptron.

  — ***La couche de sortie (output layer) :*** Cette couche représente le résultat final du réseau, c'est-à-dire sa prédiction.

  À l'exception des nœuds d'entrée, chaque nœud est un neurone qui utilise une fonction d'activation non linéaire. MLP utilise une technique d'apprentissage supervisé appelée rétropropagation pour la formation. Ses multiples couches et son activation non linéaire distinguent le MLP d'un perceptron linéaire. Il peut distinguer des données qui ne sont pas linéairement séparables.

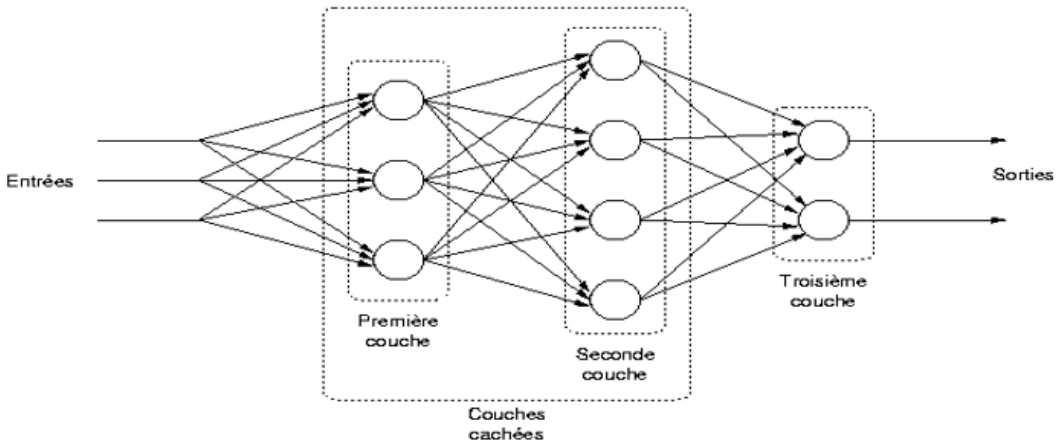


FIGURE 2.13 – perceptron multicouche

- **Réseaux de neurones convolutifs :**

  Les réseaux de neurones convolutifs (Convolutional Neural Network : CNN) sont un type de réseau de neurones spécialisés pour le traitement de données ayant une topologie semblable à une grille. Les exemples comprennent des données de type série temporelle, qui peuvent être considérées comme une grille 1D en prenant des échantillons à des intervalles de temps réguliers et des données de type image, qui peuvent être considérées comme une grille 2D de pixels.

Les réseaux convolutifs ont connu un succès considérable dans les applications pratiques. CNN est une séquence de couches, et chaque couche transforme un volume d'activation en un autre par une fonction différentiable [18]. Les trois principaux types de couches pour construire ce type de réseau sont : couche convolutive, couche de pooling et couche entièrement connectée.

— ***La couche convolutive :*** C'est la couche la plus importante et le cœur des éléments constitutifs du réseau convolutif, et c'est aussi elle qui effectue le plus de calculs lourds.

— ***La couche de pooling :*** Il est courant d'insérer périodiquement une couche Pooling dans ce type d'architecture. Sa fonction est de réduire progressivement la taille spatiale de la représentation pour réduire le nombre de paramètres et de calculs dans le réseau, et donc de contrôler également le overfitting (le sur-apprentissage).

— ***La couche entièrement connectée :*** Cette couche constitue toujours la dernière couche d'un réseau de neurones, elle est entièrement connectée à tous les neurones de sorties (d'où le terme entièrement connectée : fully-connected).

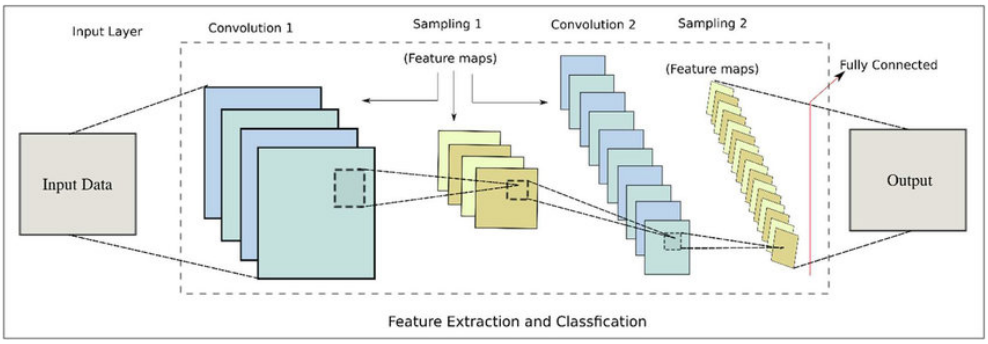


FIGURE 2.14 – Architecture du modèle CNN.

- **Les réseaux de neurones récurrents :**

Les réseaux de neurones récurrents (RNN) est un type de réseau neuronal dans lequel la sortie de la couche précédente est alimentée en entrée de la couche en cours. Dans les réseaux de neurones traditionnels, toutes les entrées et sorties sont indépendantes les unes des autres, mais dans des cas comme lorsqu'il est nécessaire de prédire le mot suivant d'une phrase, les mots précédents sont nécessaires et il est donc nécessaire de se souvenir des mots précédents. C'est ainsi que RNN a vu le jour, ce qui a résolu ce problème à l'aide d'une couche cachée.
Les RNN sont appelés récurrents car ils exécutent la même tâche pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents. Voici à quoi ressemble un RNN typique :

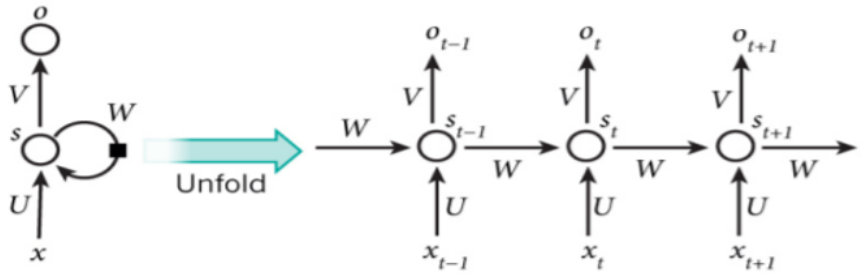


FIGURE 2.15 – (à gauche) un RNN (à droite) sa version déroulée

Le schéma ci-dessus montre un RNN déroulé. En déroulant, nous signifions simplement qu'on montre le réseau pour la séquence complète. Par exemple, si la séquence qui nous intéresse est une phrase de 3 mots, le réseau serait déroulé en un réseau de neurones de 3 couches, une couche pour chaque mot. Les formules qui dirigent les calculs dans un RNN sont les suivantes :

— xt est l'entrée au moment t.
— U, V, W sont les paramètres que le réseau va apprendre des données de l'apprentissage.
— st est l'état caché au moment t. C'est la « mémoire » du réseau. st est calculé en fonction de l'état caché précédent et de l'entrée à l'étape actuelle :

$$st = f(Uxt + W\ st\text{-}1)$$

Où f est une fonction non linéaire telle que : ReLu ou Hyperbolic tangent (tanh).

— ot est la sortie au moment t. Par exemple, si on veut prédire le prochain mot dans une phrase, ce serait un vecteur de probabilités dans un vocabulaire.

$$ot = softmax(V\ st)$$

Les RNN ont des difficultés à apprendre des dépendances sur le long terme et les interactions entre des mots qui sont éloignés les uns des autres [19]. C'est problématique parce que la signification d'une phrase est souvent déterminée par des mots qui ne sont pas très proches. Dans ce cas précis si la fonction d'activation est une tanh ou bien une sigmoïde alors nous aurons le problème de la disparaition du gradient.

C'est pour cette raison qu'on se tourne vers la fonction d'activation ReLu qui elle ne souffre pas de ce problème, mais il existe une solution encore plus utilisée c'est l'utilisation des architectures Long Short term Memory (LSTM) ou Gated Recurrent Unit (GRU) :

RNN LSTM (Long Short-Term Memory) :

Les réseaux de mémoire à long terme à court terme généralement appelés simplement (LSTM : Long Short Term Memory) sont un type spécial de RNN.

Les Réseaux neuronaux récurrents présentés dans la section précédente sont capables d'apprendre des règles de mise à jour de séquence arbitraire en théorie. Dans la pratique, cependant, ces modèles oublient généralement rapidement le passé [20] . C'est ce qu'on appelle le problème de la disparition de gradient et c'est pourquoi ils ont inventé le LSTM. La cel-

lule LSTM est une adaptation de la couche récurrente qui permet aux signaux plus anciens des couches profondes de se déplacer vers la cellule du présent . Dans un réseau LSTM, trois portes sont présentes :

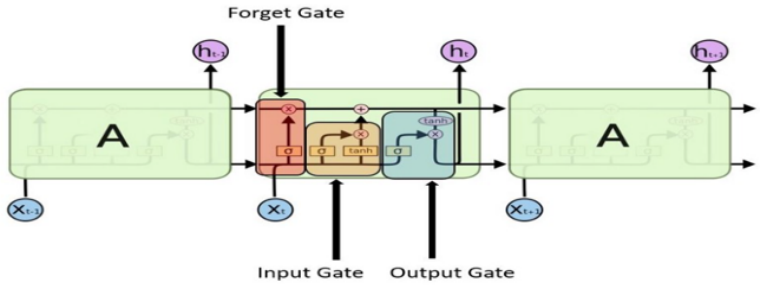
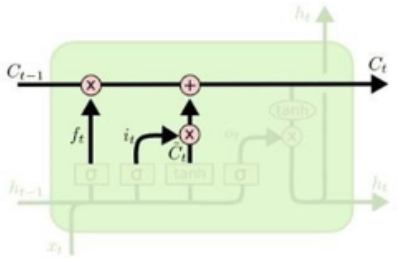


FIGURE 2.16 – LSTM avec ses portes.

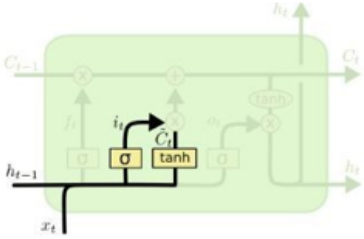
— ***Forget gate (Porte d'oubli) :*** Cette porte décide quelles informations doivent être conservées ou jetées.il est décidé par la fonction sigmoïde : il regarde l'état précédent (ht-1) et l'entrée (Xt), si la sortie de la sigmoïde est proche de 0, cela signifie que l'on doit oublier l'information et si on est proche de 1 alors il faut la mémoriser pour la suite.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

FIGURE 2.17 – La porte d'oubli .

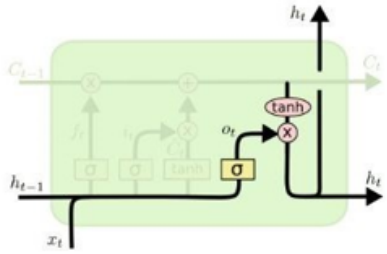
— ***Input gate (Porte d'entrée) :*** Cette porte décide si l'entrée doit modifier le contenu de la cellule. La fonction sigmoïde décide des valeurs à laisser passer par 0 et 1,et la fonction tanh donne un poids aux valeurs qui sont passées en décidant de leur niveau d'importance allant de -1 à 1.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

FIGURE 2.18 – La porte d'entrée .

— ***Output gate (porte de sortie) :*** Cette porte détermine si une cellule peut envoyer une valeur aux autres cellules ou non. En effet, quand sa valeur est proche de zéro, la cellule ne fournit aucune information en sortie Cette porte décide quelle partie de la cellule actuelle parvient à la sortie.

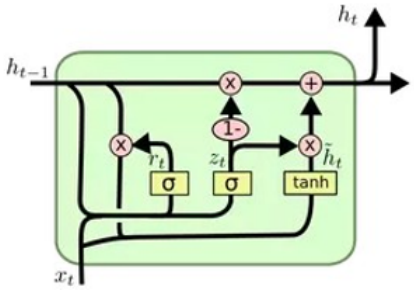


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

FIGURE 2.19 – La porte de sortie.

RNN GRU (Gated Recurrent Unit) **:**

RNN GRU est un autre type de réseaux de neurones récurrents [21], vise à résoudre le problème de la disparition de gradient qui vient avec un réseau neuronal récurrent standard. GRU est une modification de la couche cachée RNN qui permet de mieux capturer les connexions longue portée.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

FIGURE 2.20 – Cellule GRU.

GRU est similaire au LSTM mais sa structure est simplifiée. Comme LSTM, pour résoudre le problème de la disparition du gradient d'un RNN standard, GRU utilise, ce qu'on appelle, la porte de mise à jour et la porte de réinitialisation. Fondamentalement, ce sont deux vecteurs qui décident quelles informations doivent être transmises à la sortie.

— ***La porte de réinitialisation :*** Essentiellement, cette porte est utilisée à partir du modèle pour décider de la quantité d'informations passées à oublier lors du calcul des informations présentes. La réinitialisation de la porte rt est calculée comme indiqué dans la formule.

— ***La porte de mise à jour :*** La porte de mise à jour agit de la même manière que la porte d'entrée et d'oubli d'un LSTM. Il décide des informations à jeter et des nouvelles informations à ajouter.

Les réseaux neuronaux mentionnés, ne sont pas tous les réseaux qui existent, une excellente ressource qui résume peut-être toute les architectures est en [22]. La figure suivante montre une représentation de leur travail :

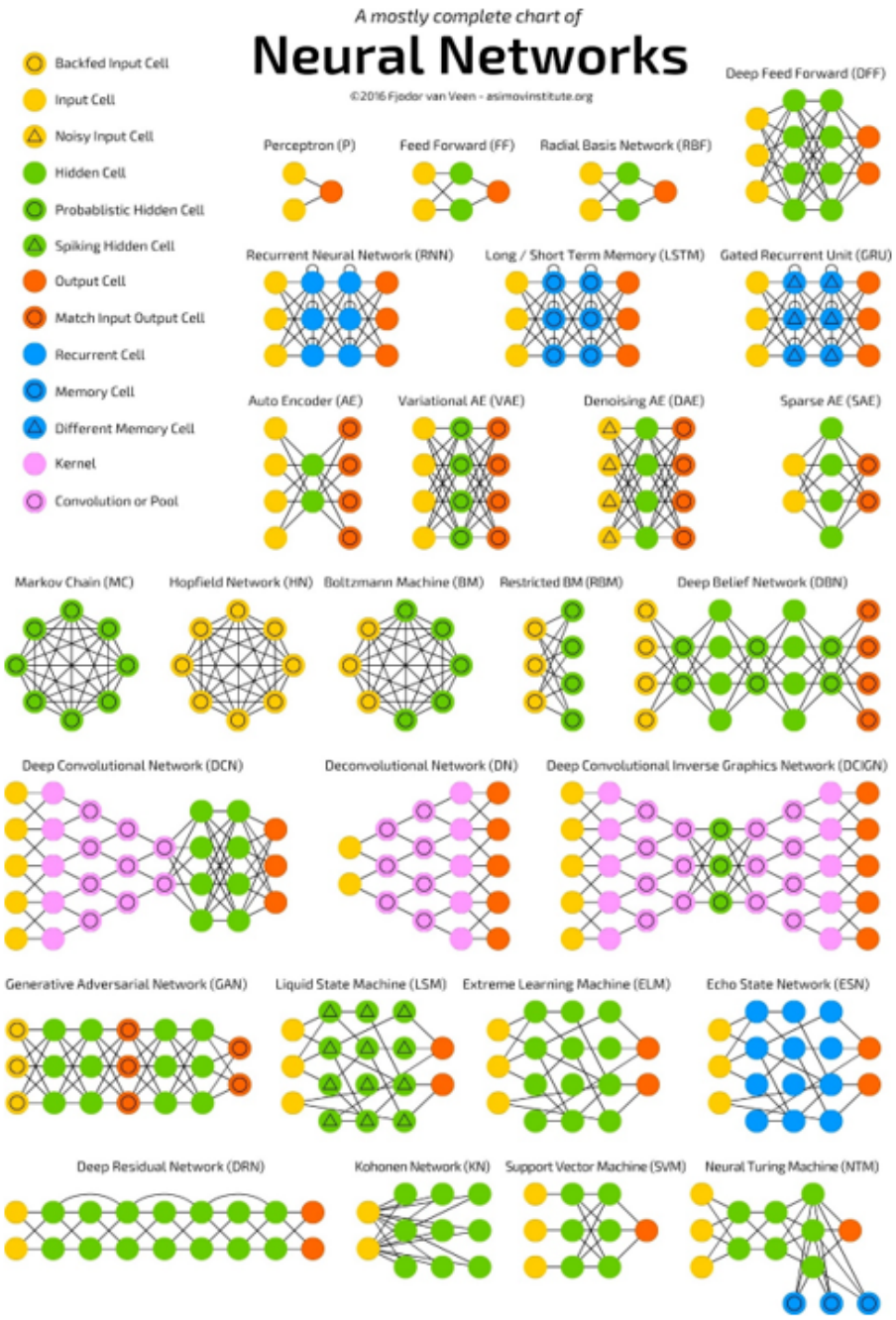


FIGURE 2.21 – Un résumé des types d'architectures de réseaux de neurones [22].

2.3.4 L'apprentissage en Deep Learning

- **Introduction :**

Le processus d'apprentissage dans le Deep Learning revient à l'entraînement du réseau neuronal en utilisant des optimiseurs itératifs, qui ne font que conduire la fonction de coût à une très faible valeur [23]. Nous pouvons utiliser différents algorithmes pour effectuer l'apprentissage , mais l'algorithme le plus utilisé est l'algorithme itératif d'optimisation par la descente de gradient qu'est la méthode la plus utilisée presque sur tous les réseaux neuronaux avec ses différents modèles.
Le processus d'apprentissage revient au problème d'optimisation où il s'agit de minimiser ou de maximiser une fonction f(x). Cette fonction est appelée la fonction objective, les auteurs dans [23] l'ont appelée aussi la fonction de coût, la fonction de perte et la fonction d'erreur. Pendant l'entrainement, l'algorithme tente d'identifier le minimum global sans tomber dans le piège du minimum local. Le schéma suivant réalisé par les auteurs en [23] est une illustration de ce concept :

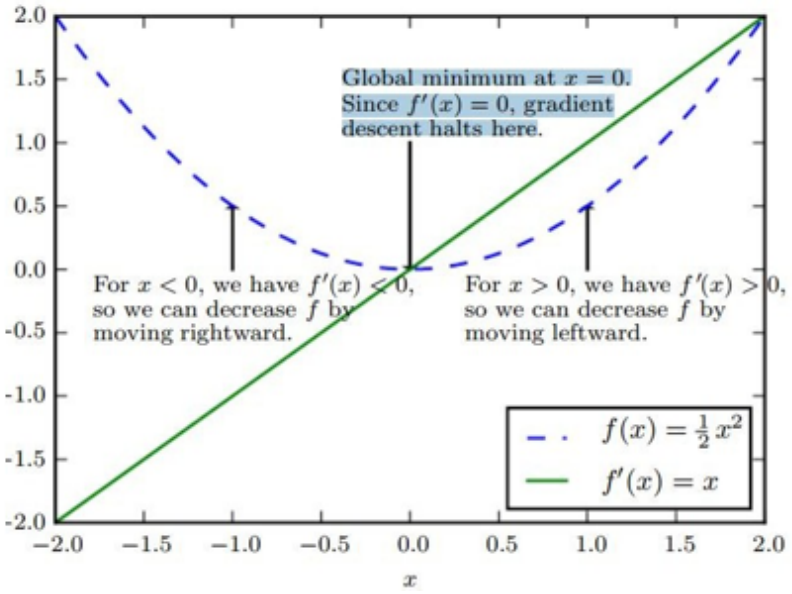


FIGURE 2.22 – Une illustration du processus de recherche de l'optimum [23].

- **Les variantes de la descente de gradient :**

Il existe trois principaux types de variantes de l'algorithme de descente de gradient. La principale différence entre eux est la quantité de données que nous utilisons lorsque nous calculons le gradient pour chaque étape d'apprentissage . Les algorithmes d'optimisation qui utilisent tout l'ensemble de l'apprentissage sont appelés les méthodes de gradients déterministes ou batch descente de gradient, car ce sont des méthodes où tous les exemples d'apprentissage sont traités simultanément dans un grand batch, tandis que le terme "batch" pour décrire un groupe d'exemples.
Les algorithmes d'optimisation qui n'utilisent qu'un seul exemple à la fois sont parfois appelés méthodes stochastiques ou en online descente de gradient.

La plupart des algorithmes utilisés pour l'apprentissage profond se situent quelque part entre les deux, utilisant plus d'un mais moins que tous les exemples d'entrainement. Ils sont appelés les méthodes stochastiques de minibatch ou de minibatch descente de gradient.

- **Les Algorithmes d'optimisation de la descente de gradient :**

  - ***Adam :*** Adam est un algorithme d'optimisation présenté en 2015. Le nom de cet algorithme est dérivé de Adaptive Moment Estimation. Lors de l'introduction de cet algorithme, les auteurs ont présenté les avantages de l'utilisation d'Adam sur des problèmes d'optimisation non convexes, comme suit :

    — Simplicité de mise en œuvre.
    — Efficacité du calcul.
    — Peu de mémoire requise.
    — Bien adapté aux problèmes volumineux en termes de données et/ou de paramètres.
    — Les hyper-paramètres nécessitent généralement peu de réglages.

    Il existe d'autres optimiseurs avec différents mécanismes de fonctionnement, comme :

  - ***Adagrad***
  - ***RMSProp***
  - ***Adadelta***

2.4 Comparaison entre Machine Learning et Deep Learning :

- ***Fonctionnement*** : l'apprentissage profond est un sous-ensemble de l'apprentissage automatique qui prend des données comme entrée et prend des décisions intuitives et intelligentes à l'aide d'un réseau neuronal artificiel empilé par couches. D'un autre côté, l'apprentissage automatique en tant que super-ensemble d'apprentissage profond prend les données comme entrée, analyse ces données, essaie de leur donner un sens (décisions) en fonction de ce qu'il a appris pendant l'entraînement.

- ***Dépendance des données :*** les algorithmes d'apprentissage automatique fonctionnent souvent bien même si l'ensemble de données est petit, mais l'apprentissage profond est gourmand en données, plus vous avez de données, meilleures sont les performances.

- ***Temps d'entraînement et d'inférence :*** le temps d'entraînement pour l'apprentissage automatique prend relativement peu de temps pour s'entraîner, allant de quelques secondes à quelques heures. Alors que la formation d'un réseau d'apprentissage profond prend généralement beaucoup de temps car un algorithme d'apprentissage profond implique de nombreuses couches.

- ***Résultat :*** le résultat d'un apprentissage automatique traditionnel est généralement une valeur numérique, tel qu'un score ou une classification. Alors que le résultat d'un apprentissage profond peut être un score, un élément, du texte, de la parole, etc.

2.5 Conclusion :

Nous avons présenté au cours de ce chapitre l'apprentissage automatique avec ses différents types, supervisé, non- supervisé et avec renforcement, ses domaines d'application et ses différents algorithmes, ensuite nous avons présenté l'apprentissage profond, ses applications, les réseaux de neurones où on a parlé sur le neurone biologique et formel, les fonctions d'activation et on a cité quelques types des réseaux de neurones. À la fin de ce chapitre, on a fait une comapraison entre l'apprentissage automatique et l'apprentissage profond.

Chapitre 3

État de l’art

3.1 Introduction :

Dans le but de bien mener et aborder notre problématique, on a commencé par chercher et lire plusieurs articles publiés qui traitent des problématiques sur la classification des arythmies ECG avec les méthodes de Machine Learning et de Deep Learning.

3.2 Approche Machine Learning :

3.2.1 Approche : *Classification des signaux ECG avec SVM*

- ***Explication de la méthode utilisée :***

  Dans le but de faire la classification des données ECG d'une façon automatique avec les outils de machine learning, dans [24], ils ont utilisé un modèle de machine learning qui est SVM. La figure suivante montre le système global de classification des signaux ECG :

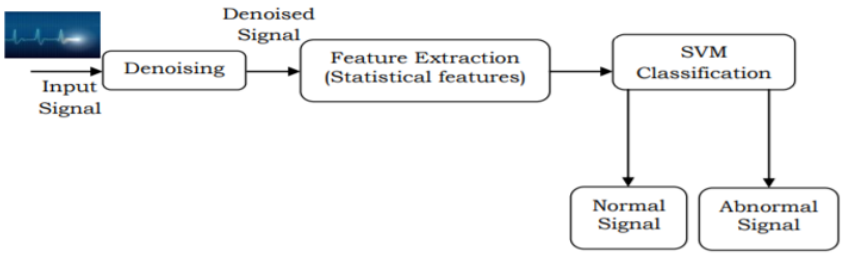


FIGURE 3.1 – Système de classification des signaux ECG.

Le système de classification des signaux ECG se compose de trois étapes :

La première étape est le prétraitement des données. C'est un nom commun pour les opérations avec des signaux au niveau le plus bas d'abstractions et c'est une amélioration des données de signal qui supprime ou améliore certaines caractéristiques du signal. Dans cette étape, le signal débruitage est employé en utilisant le filtre médian.

La deuxième étape est l'extraction de caractéristiques qui utilise des caractéristiques statistiques pour la classification. Les parties intéressantes d'un signal sont représentées par des dimensions à ce stade. C'est un type de réduction dimensionnelle car tout le signal est représenté par un vecteur de caractéristiques compact et cette étape est très utile dans une situation où la dimension d'un signal d'entrée est trop grande. Afin de compléter la tâche de classification rapidement dans la récupération et la correspondance de signaux, les fonctionnalités d'entrée devrait être sous une forme compacte. Les mesures statistiques suivantes : la moyenne, l'écart type, la variance et l'asymétrie sont calculés.

La dernière étape du système est la classification. Il analyse les propriétés numériques des caractéristiques de signal inconnus afin de les classer en différentes classes en utilisant les fonctionnalités d'entraînement (training features) des classes connues. Il utilise généralement deux phases : entraînement et test (train et test). En entraînement, les caractéristiques des échantillons sont connues et leurs classes sont introduits dans le classificateur SVM. Ensuite,

le classificateur formé est testé avec le test échantillon. Sur la base des résultats des tests avec des données de vérité, les performances du système sont analysées.

- ***Les caractéristiques statistiques :***

  Les caractéristiques statistiques extraites du signal ECG d'entrée sont données comme suit :

  — ***Moyenne :*** C'est la moyenne d'un ensemble de valeurs et définie comme le rapport entre les sommes de ces valeurs et du nombre d'éléments de l'ensemble.

  — ***Écart type :*** La quantité de dispersion ou de variation d'un ensemble de données donné est quantifiée en utilisant cette mesure.

  — ***Asymétrie :*** C'est la somme des valeurs de la distribution des données divisée par le nombre de valeurs dans la distribution.

  — ***Variance :*** Considérons un ensemble de nombres aléatoires X et sa moyenne $\bar{X}$. La variance mesure la distance entre X et $\bar{X}$. En outre, il est calculé à partir de la déviation au carré de X par rapport à son $\bar{X}$.

- ***Résultats et discussions :***

  Dans cette étude, le jeu de données d'arythmie MIT-BIH est utilisé pour l'analyse. Un exemple d'enregistrement normal et anormal dans la base de données MIT-BIH est illustrée sur la figure 3.2 et 3.3 respectivement. La base de données contient deux canaux enregistrement ambulatoire de 48 extraits d'une demi-heure de 47 sujets. Elles sont numérisées à 360 échantillons par seconde.

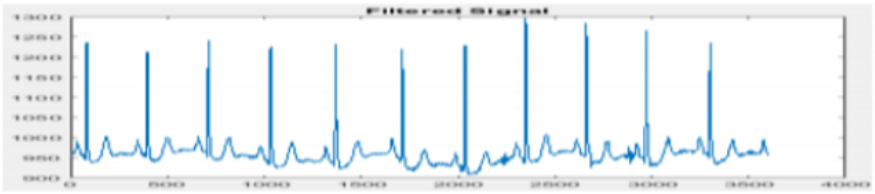


FIGURE 3.2 – Un signal normal.

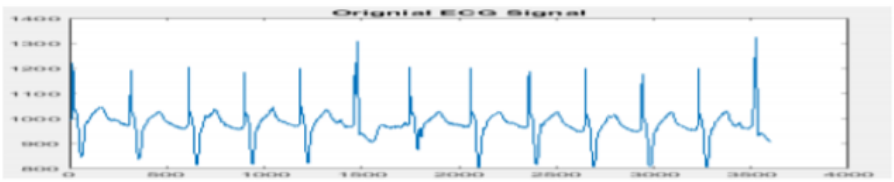


FIGURE 3.3 – Un signal anormal

Avant d'extraire les caractéristiques du signal, tous les signaux ECG sont désactivés en utilisant filtrage médian simple. La taille de la fenêtre utilisée dans cette étude est 3. Figure 3.4 et 3.5 montrent la version sans bruit des enregistrements normaux de la figure 3.2 et les enregistrements anormaux de la figure 3.3 respectivement.

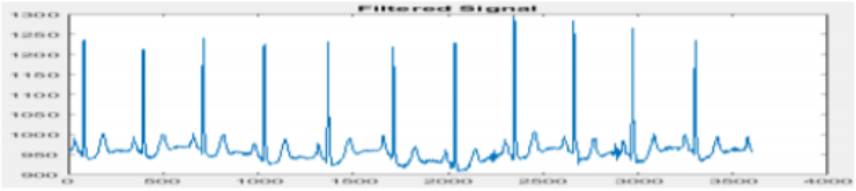


FIGURE 3.4 – Un signal normal sans bruit.

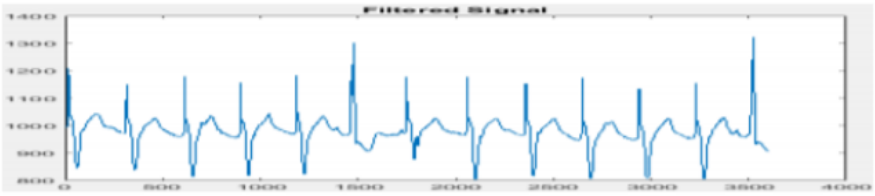


FIGURE 3.5 – Un signal anormal sans bruit.

Enfin, la classification d'un signal à l'aide du classificateur SVM est analysée avec l'aide des caractéristiques statistiques extraites de la version sans bruit des enregistrements ECG d'entrée. La validation croisée du facteur k (k-fold) est utilisée. La figure 3.6 et 3.7 montrent la courbe ROC et la matrice de confusion de la classification du signal ECG analysé.
Il est observé que le système ECG fournit une précision de 90% en termes de sensibilité, de spécificité et d'exactitude également.

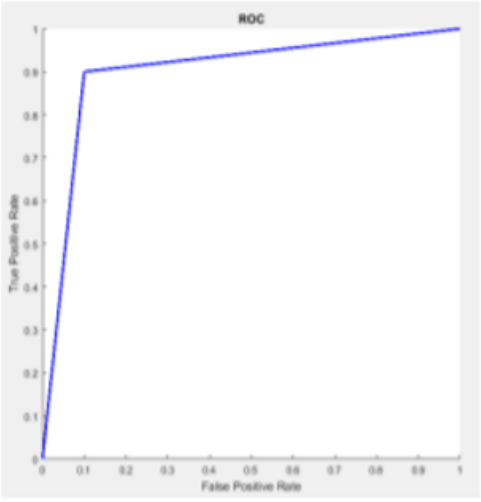


FIGURE 3.6 – La courbe ROC

FIGURE 3.7 – La matrice de confusion

- ***Conclusion :***

  Dans cet article, la classification des signaux ECG à l'aide de caractéristiques statistiques est analysée. Les enregistrements de la base de données d'arythmie du MIT-BIH sont utilisés pour la classification. Tout d'abord, le prétraitement est effectué à l'aide du filtre médian, et les caractéristiques statistiques sont extraites. Enfin, la classification basée sur SVM-RBF est utilisé pour classer les signaux. Les résultats expérimentaux montrent les performances du système de classification des signaux ECG avec des résultats prometteurs. Le classificateur SVM-RBF classe correctement 90% du signal ECG donné avec des fonctionnalités statistiques simples.

3.3 Approches Deep Learning :

3.3.1 Approche 1 : *Classification des arythmies ECG en utilisant un réseau neuronal convolutif 2-D*

- ***Explication de la méthode utilisée :***

  Dans le but de faire la classification des données ECG d'une façon automatique avec les outils de deep learning, dans [25], ils ont utilisé un modèle de deep learning qui est le CNN et cette procédure comprend deux étapes :
  Le prétraitement des données et le classificateur d'arythmies ECG. Dans cet article, ils ont utilisé la base de données sur les arythmies MIT-BIH [26] pour l'entraînement et le test des modèles CNN.
  Comme le modèle CNN traite une image bidimensionnelle en tant que donnée d'entrée, les signaux ECG sont transformés en images ECG pendant l'étape de prétraitement des données ECG. Avec ces images d'ECG obtenues, la classification de huit types d'ECG est effectuée dans l'étape de classificateur CNN. Les procédures globales sont indiquées dans la figure suivante :

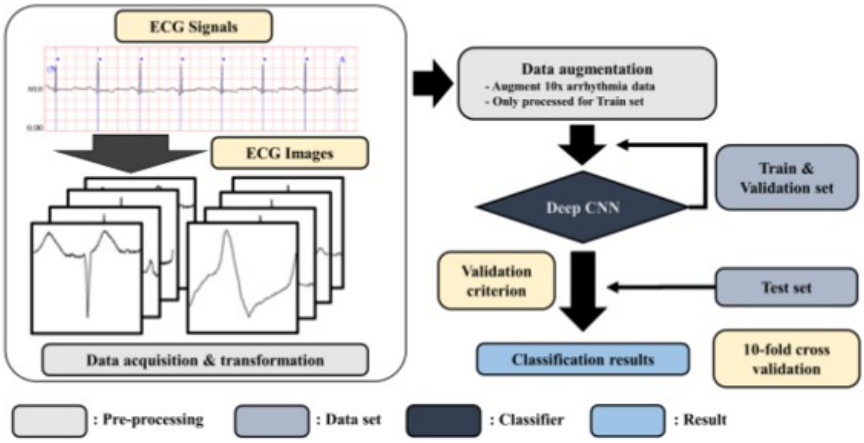


FIGURE 3.8 – Procédures globales traitées dans la classification des arythmies ECG.

— **ETAPE 1 : Prétraitement des données ECG** :

CNN à deux dimensions nécessite une image en tant que donnée d'entrée. Par conséquent, ils ont transformé les signaux ECG en images ECG en traçant chaque battement ECG sous la forme d'une image individuelle de 128 x 128 en niveaux de gris. Dans la base de données sur les arythmies du MIT-BIH, chaque battement ECG est découpé en tranches en fonction du temps de pointe de l'onde Q. Plus précisément le type de l'arythmie est étiquetée au moment du pic d'onde Q de chaque battement ECG. Ainsi, on définit une seule image de battement ECG en centrant le signal de crête de l'onde Q en excluant le premier et les 20 derniers signaux ECG des signaux précédents et suivants.

— **ETAPE 2 : Classificateur d'arythmies ECG** :

Dans cet article, ils ont adopté CNN comme classificateur d'arythmie ECG. La raison pour laquelle ils ont appliqué la 2D CNN en convertissant le signal ECG en forme d'image ECG est que les couches de convolution et de regroupement 2D sont plus adaptées pour filtrer l'espace localité des images ECG.

Il existe plusieurs modèles CNN réussis de ImageNet Large Visual Perception Challenge (ILSVRC) [27], une compétition pour détecter et classer des objets dans un ensemble d'images donné.
AlexNet [28], annoncé en 2012, a pris la première place avec des performances écrasantes comme premier modèle à utiliser le CNN et GPU à ILSVRC.

En 2014, GoogLeNet [29] et VGGNet [30] sont les premiers et deuxième dans l'ILSVRC, respectivement. Bien que VGGNet occupe la deuxième place, la structure VGGNet avec structure répétitive de filtres 3x3 et sous-échantillonnage est plus souvent utilisé dans la reconnaissance d'images car la structure est beaucoup plus simple et les performances ne sont pas très différentes de celles de GoogLeNet.

ResNet [31] et DenseNet [32], qui sont apparus récemment, ont été proposé pour résoudre le problème que l'effet des premières caractéristiques de l'image sur la sortie finale est faible lorsque le modèle CNN est plus profond.

Dans l'article, ils suivent les principes de base structure de VGGNet et optimiser le modèle CNN pour montrer des performances optimales pour la classification des arythmies ECG. La comparaison des performances de la proposition du modèle CNN a été réalisé avec AlexNet et VGGNet sans comparer avec GoogLeNet, ResNet et DenseNet. En effet, l'image ECG de ce papier est une image relativement simple de 128x128 niveaux de gris, il n'est donc pas nécessaire d'avoir une couche profonde, et une augmentation des paramètres libres peut entraîner le sur-apprentissage des performances.

La figure suivante montre l'architecture globale du modèle CNN proposé :

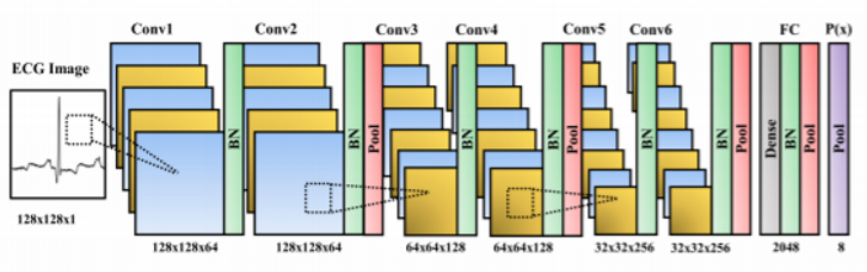


FIGURE 3.9 – Architecture du modèle CNN proposé.

Data augmentation :

L'augmentation des données signifie l'augmentation du nombre de points de données. En termes d'images, cela peut signifier l'augmentation du nombre d'images dans le jeu de données. Nous pouvons atteindre à la fois une spécificité et une sensibilité élevées en augmentant et en équilibrant les données d'entrée. On augmente sept battements d'arythmie ECG (PVC, PAB, RBB, LBB, APC, VFW, VEB) avec neuf méthodes de recadrage différentes : gauche haut, centre haut, droite haut, centre gauche, centre, centre droite, gauche bas, centre bas et en bas à droite. Chaque méthode de recadrage donne deux des trois tailles d'une image ECG, c'est-à-dire 96 x 96. Ensuite, ces images augmentées sont redimensionnées à la taille originale de 128 x 128.

- ***Résultats obtenus :***

Les résultats d'évaluation résumés des modèles CNN sont présentés dans le tableau suivant où Native fait référence aux données d'entraînement sans augmentation et Augmented fait référence à l'augmentation. Le modèle CNN proposé a atteint 0,989 AUC, 99,05% précision moyenne, 99,57% spécificité, sensibilité moyenne de 97,85% et 98,55 valeur prédictive positive.

Method	Grade	AUC	Acc(%)	Sp(%)	Se(%)	+P(%)
Proposed	Native	0.986	98.90	99.64	97.20	98.63
	Augmented	**0.989**	**99.05**	99.57	**97.85**	98.55
AlexNet	Native	0.985	98.81	**99.68**	96.81	**98.63**
	Augmented	0.986	98.85	99.62	97.08	98.59
VGGNet	Native	0.984	98.77	99.43	97.26	98.08
	Augmented	0.984	98.63	99.37	96.93	97.86

FIGURE 3.10 – Les résultats des différent modèles CNN.

À partir du tableau, le modèle CNN proposé avec data augmentation montre les meilleurs résultats d'AUC, de précision et de sensibilité.
AlexNet sans augmentation présente la meilleure spécificité et une valeur prédictive positive. Discrimination de sensibilité entre la méthode proposée et la deuxième plus élevé est de 0,59% alors que la discrimination de spécificité de la méthode proposée et la plus élevée est de 0,11%. Cela indique que le modèle proposé a obtenu la meilleure précision dans la classification de l'arythmie.

VGGNet affiche le pire résultat par rapport aux deux autres modèles CNN. La principale différence entre le VGGNet et les autres modèles est le nombre de couches de regroupement. Après avoir contourné quatre couches de pooling, la taille de l'image ECG est réduite à 8 x 8. Bien que le VGGNet ait deux fois plus de nombre de kernels à la dernière couche convolutive, il semble que la dernière caractéristique de taille 16 x 16 est plus stable. Par conséquent, lorsqu'on utilise une image ECG de taille 128 x 128, il est recommandé de ne pas utiliser plus de trois couches de pooling dans le modèle CNN. Entre le modèle proposé et le AlexNet avec augmentation des données, la différence entre les sensibilités moyennes est de 0,77%. C'est une très grande différence lorsqu'on examine les sensibilités détaillées des sept arythmies différentes.

3.3.2 Approche 2 : *Classification des arythmies ECG en utilisant les réseaux de neurones récurrents :*

- ***Explication de la méthode utilisée :***

Dans [33], ils ont appliqué les réseaux de neurones récurrents pour faire la classification des arythmies des signaux ECG. Le signal ECG est composé de seize types de pulsations cardiaques divisées en deux groupes de battements de coeur normaux et d'arythmie.

La base de données d'arythmie MIT-BIH est utilisée pour examiner la performance de classification des battements. Pour l'entraînement et pour le test, l'un des ensembles de données d'arythmies MIT-BIH les plus populaires a été utilisé. Il contient 47 enregistrement de 30 minutes chacun et 40 % des enregistrements étaient ceux des patients cardiaques. Les enregistrements contenaient différents types de signaux basés sur différents emplacements de câbles : ML2, V1, V2, V4 et V5 [34].

Tout d'abord, un fractionnement de 70-30 des données a été effectué. Les signaux ML2 ont été sélectionnés et pour chaque enregistrement, le signal a été segmenté en morceaux de 720-720 de longueur. Le taux d'échantillonnage était de 360, 360 segments avant la valeur de R-Peak et 360 segments après la même ont assuré que presque 3 temps seront là en un seul morceau et le classement du battement central devait être fait contre 15 étiquettes arythmiques possibles.

la figure suivante montre l'architecture globale du systéme :

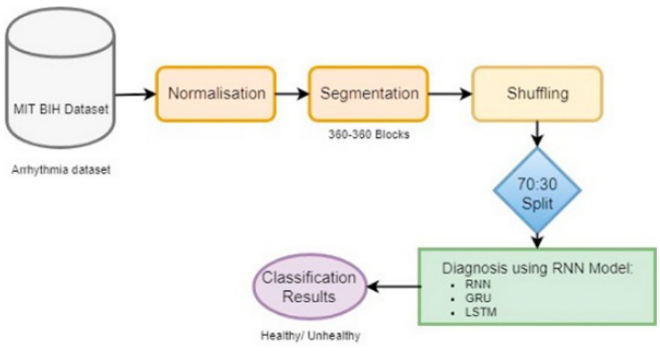


FIGURE 3.11 – Les étapes de détection de l'arythmie ECG proposées.

- ***Résultats obtenus :***

  Le RNN LSTM a montré une précision de 88,1% lorsque nous considérons le nombre d'itérations comme étant 5 et le nombre de couche cachées comme étant 3, il y a 64, 256 et 100 neurones par couche cachée respectivement qui montre mieux la détection d'arythmie que RNN simple et GRU car la précision de RNN est de 85.4% et GRU est de 82.5% ce qui est moins par rapport à la précision de LSTM. Le modèle est implémenté directement en utilisant les signaux de la base de données MIT BIH et aucun prétraitement n'a été effectué. Par conséquent, la complexité du modèle implémenté est beaucoup moindre que les algorithmes traditionnels d'apprentissage automatique.

  Dans cet article, la classification binaire de l'arythmie a été réalisée et les résultats peuvent être améliorés en l'étendant à une classification multi classe. Depuis peu de travail a été fait dans ce domaine de classification binaire (détection de l'arythmie), le modèle proposé permet la même chose et offre la possibilité de poursuivre les travaux dans ce domaine. La précision de la classification peut être augmentée plus loin en augmentant le nombre d'époques. L'article montre que le LSTM donne le meilleur résultat pour la classification binaire de l'arythmie ECG

	Algorithm	Accuracy	Sensitivity	Specificity
ECG Classification	RNN	85.4	80.6	85.7
ECG Classification	RNN GRU	82.5	78.9	81.5
LSTM ECG Classification	RNN LSTM	88.1	92.4	83.35

FIGURE 3.12 – Les résultats des différent modèles RNN.

3.4 Comparaison des approches

D'après les résultats obtenus des approches de Machine Learning et de Deep Learning utilisées pour faire la classification des données ECG, on remarque que les modèles de CNN donnent des meilleurs résultats avec un taux de précision qui dépasse 98%, que le modèle SVM qui a donné un taux de précision de 90% et les modèles RNN qui ont donné pour leur meilleur modèle un résultat de 88%.

3.5 Conclusion :

On a pu voir dans ces différents articles qu'il existe plusieurs méthodes pour la problématique de classification des arythmies ECG. Plus généralement, cette problématique se traduit par un problème de classification multi classe avec des approches qui se basent sur des algorithmes de Machine Learning et de Deep Learning.
Pour répondre à notre problématique, on va s'inspirer des articles lus en testant plusieurs algorithmes de Machine Learning et de Deep Learning et sélectionner le plus optimal pour notre problématique.

Chapitre 4

Système réalisé

4.1 Architecture générale :

Notre projet porte sur la classification des arythmies ECG avec les méthodes de machine Learning et de Deep Learning. Pour atteindre cet objectif et pour obtenir la meilleure performance possible, on propose le système suivant :
Pour faire la classification des données ECG, on va passer par deux étapes :

- ***La classification binaire :*** qui définit juste si la personne est malade ou pas, pour cela on utilise une base de données qui contient deux classes : 1 pour « malade », 0 pour « pas malade », puis si le résultat est égal à 0 la procédure est finie et la personne est saine, sinon si le résultat est égal à 1, on passe à la classification multi classe.

- ***La classification multi classe :*** on utilise une base de données qui contient 4 classes qui correspondent à des types de maladies cardiaques. La classification multi classe va nous permettre de trouver le type de la maladie donc soit : APC, PAB, PVC ou RBB.

  On applique les mêmes étapes pour tous les algorithmes utilisés dans ce projet, par exemple avec les algorithmes de Machine Learning, on fait la classification binaire puis toujours avec les mêmes algorithmes on fait la classification multi classe, et c'est la même chose pour les algorithmes de Deep Learning.

  La figure suivante montre l'architecture globale de la classification des arythmies ECG :

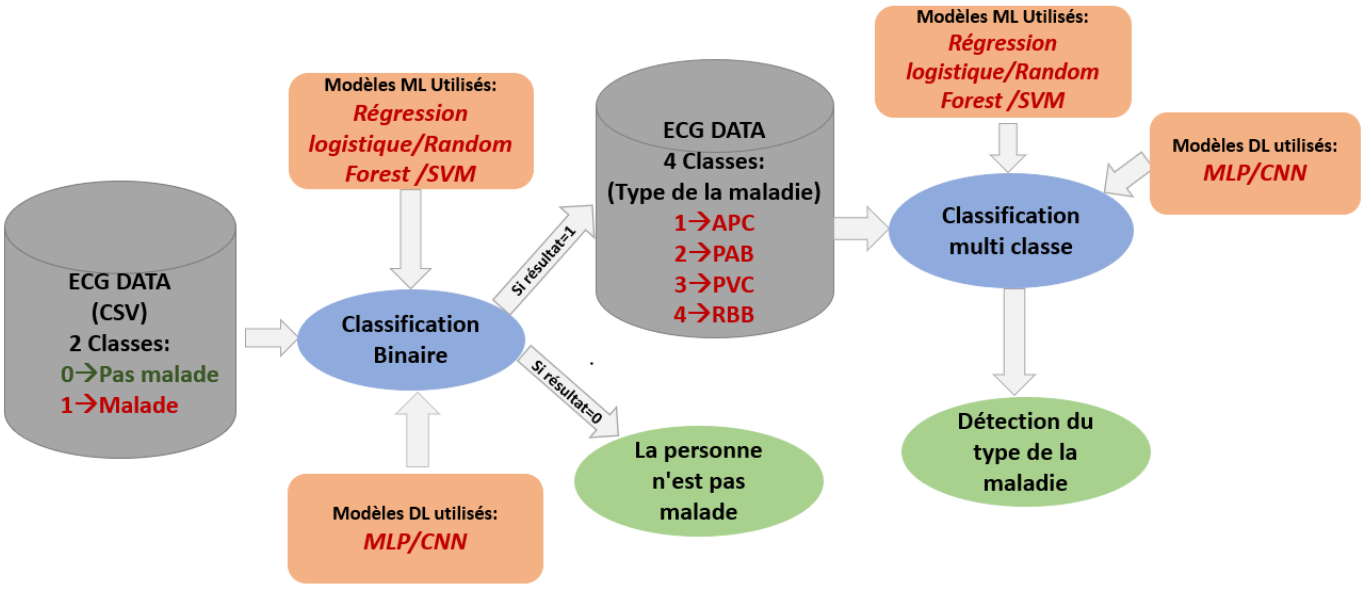


FIGURE 4.1 – L'architecture globale de la classification des arythmies ECG.

4.1.1 La raison de faire deux classifications (binaire et multi classe) :

La raison de choisir de passer par deux étapes est par rapport au nombre de classes qui n'est pas équitable : on a beaucoup plus de classes 0 que d'autres classes, donc si on fait juste une seule classification multi classe, le résultat des algorithmes sera faible et ne sera pas fiable pour un domaine qui est très sensible où la précision est très importante, en plus de ça, en général la classification binaire donne toujours un meilleur résultat que la classification multi classe si on utilise la même base de donnée.
En plus de tout ça, c'est plus prioritaire de définir d'abord si une personne est malade ou pas, puis si elle est saine on n'aura pas besoin d'aller plus loin.

4.2 La base de données utilisée :

La base de données utilisée est un fichier CSV qui contient :

— *Nombre d'échantillons :* 109446.
— *La fréquence d'échantillonnage :* 125Hz.
— *Nombre de catégories :* 5.
— *Source de données :* Ensemble de données sur l'arythmie MIT-BIH de Physionet [35], cet ensemble de données a été utilisé pour explorer la classification des pulsations cardiaques à l'aide d'architectures des réseaux neuronaux profonds et pour observer certaines des capacités d'apprentissages par transfert associées. Les signaux correspondent aux battements du cœur sous forme d'électrocardiogramme (ECG) pour le cas normal et aux cas affectés par différentes arythmies et infarctus du myocarde. Ces signaux sont prétraités et segmentés, chaque segment correspondant à un battement du cœur.

4.2.1 Prétraitement de données :

- ***Classification binaire :*** On sait que les classes 1, 2, 3 et 4 correspondent à des signaux des personnes malades, et pour faire la classification binaire il faut qu'il y ait que deux classes : 0 et 1.
  Donc on remplace toutes les classes qui ont 2, 3 et 4 par 1, pour avoir qu'une seule classe des signaux des personnes malades. Donc en tout on aura deux classes 0 et 1 et donc on pourra faire une classification binaire.

- ***Classification multi classe :*** Le but de la classification multi classe est de déterminer le type de la maladie vu qu'on sait si on passe à la classification multi classe, donc la personne est malade.
  Donc on prend la base de données originale qui avait 5 classes et on supprime toutes les lignes qui contiennent la classe 0 puis on n'aura à la fin que 4 classes qui correspondent aux types des maladies cardiaques.

4.2.2 Aperçu sur la base de données :

La base de données contient 109446 lignes et 188 colonnes.
— De 0 jusqu'a 186 se sont des attributs (features).
— La colonne 187 représente la classe cible (label) de la base de données qui peut être soit 0 ou 1 lors de la classification binaire et soit 1, 2, 3 ou 4 lors de la classification multi classe.

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049020	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082621	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045946	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058011	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.150235	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 188 columns

FIGURE 4.2 – Les 5 premières lignes de la base de données.

4.3 Les méthodes d'optimisation et les mesures de performance utilisées :

4.3.1 Les méthodes d'optimisation :

- ***Hyperparamètre :*** En apprentissage automatique, un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage. En revanche, les valeurs des autres paramètres (généralement les poids des nœuds) sont dérivées via l'apprentissage.

- ***Cross validation :*** La validation croisée (CV) est l'une des techniques utilisées pour tester l'efficacité d'un modèle d'apprentissage automatique, c'est aussi une procédure de rééchantillonnage utilisée pour évaluer un modèle si nous avons des données limitées. Cette méthode comprend les étapes suivantes :

  1. Divise les n observations de l'ensemble de données en k sous-ensembles de taille mutuellement exclusifs et égaux ou presque égaux appelés « plis ».
  2. Ajustez le modèle en utilisant les plis k-1 comme jeu d'entraînement et un pli (k ième) comme jeu de test. Une fois chaque itération terminée, enregistrez l'erreur du modèle.
  3. Répétez ce processus k fois en utilisant un pli différent à chaque fois comme ensemble de test et les plis restants (k-1) comme ensemble d'apprentissage.
  4. Une fois toutes les itérations terminées, prenez la moyenne des k modèles. Ce serait l'erreur quadratique moyenne du modèle.

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

Une considération importante de cette approche est la sélection du nombre de plis. Le choix du nombre de plis doit être fait en partant du principe que chaque pli doit avoir suffisamment de points de données pour fournir une estimation juste des performances du modèle. D'autre part, le nombre k ne doit pas être aussi petit que 2, afin d'avoir suffisamment de modèles entraînés pour évaluer les performances du modèle.

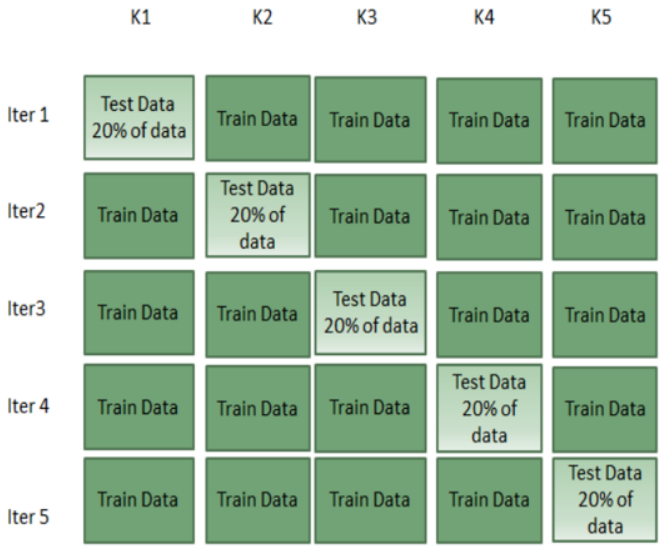


FIGURE 4.3 – Méthode de validation croisée K-Fold.

- ***RandomizerSearch :*** La recherche aléatoire établit une grille de valeurs hyperparamétriques et sélectionne des combinaisons aléatoires pour former le modèle et le score. Cela vous permet de contrôler explicitement le nombre de combinaisons de paramètres qui sont tentées. Le nombre d'itérations de recherche est défini en fonction du temps ou des ressources. Scikit-Learn propose la fonction RandomizedSearchCV pour ce processus.

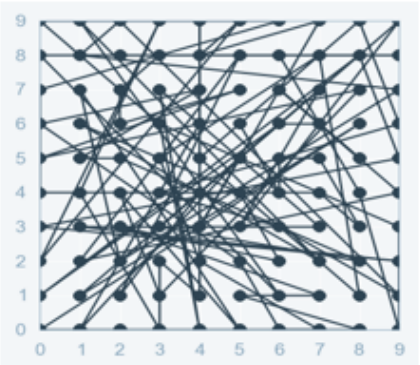


FIGURE 4.4 – RandomizerSearch.

- ***GridSearch :*** La recherche de grille est le processus qui consiste à effectuer un réglage hyperparamétrique afn de déterminer les valeurs optimales pour un modèle donné. Ceci est important car les performances de l'ensemble du modèle sont basées sur les valeurs d'hyper paramètre spécifiés.

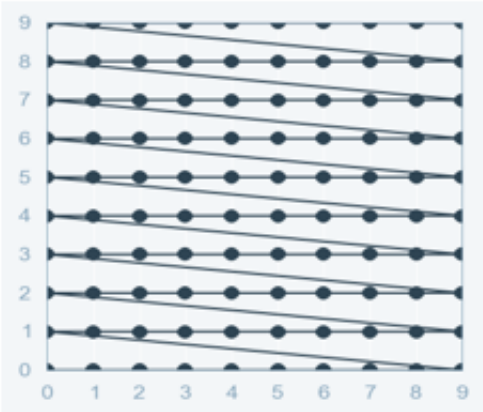


FIGURE 4.5 – GridSearch

- ***Overfitting :*** Le sur-apprentissage fait référence à un modèle qui modélise trop bien les données d'entraînement. Le sur-apprentissage se produit lorsqu'un modèle apprend les détails et le bruit dans les données d'apprentissage dans la mesure où cela a un impact négatif sur les performances du modèle sur de nouvelles données. Cela signifie que le bruit ou les fluctuations aléatoires dans les données d'apprentissage sont captés et appris en tant que concepts par le modèle. Le problème est que ces concepts ne s'appliquent pas aux nouvelles données et ont un impact négatif sur la capacité des modèles à se généraliser.

- ***Underfitting :*** Le sous-apprentissage fait référence à un modèle qui ne peut ni modéliser les données d'entraînement ni généraliser à de nouvelles données. Un modèle d'apprentissage automatique non adapté n'est pas un modèle approprié et sera évident car il aura de mauvaises performances sur les données d'entraînement. L'un des indicateurs les plus alarmants d'un modèle d'apprentissage automatique peu performant est un test de précision des données d'entraînement et de test. Un test de données indiquera si le modèle est sur-appris, sous-appris ou équilibré. La raison pour laquelle on divise l'ensemble d'entraînement et l'ensemble de test est pour qu'on puisse déterminer et ajuster les performances de nos modèles. Sinon, on entraîne aveuglement nos modèles à prédire sans aucun aperçu des performances du modèle.

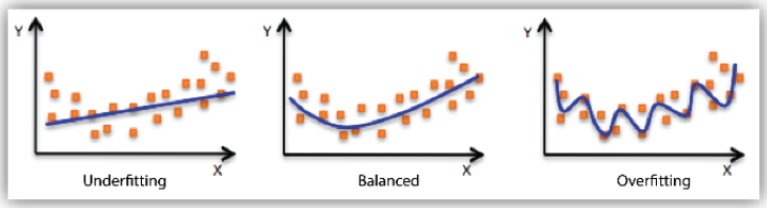


FIGURE 4.6 – Le sous-apprentissage, équilibré et le sur-apprentissage.

- ***Le sur-échantillonnage*** aléatoire implique la duplication aléatoire d'exemples de la classe minoritaire et les ajouter à l'ensemble de données d'entraînement. Les exemples de l'ensemble de données d'entraînement sont sélectionnés au hasard avec remplacement. Cela signifie que des exemples de la classe minoritaire peuvent être choisis et ajoutés plusieurs fois au nouvel ensemble de données d'entraînement « plus équilibré » ; ils sont sélectionnés à partir de l'ensemble de données d'entraînement d'origine, ajoutés au nouvel ensemble de données d'entraînement, puis renvoyés ou « remplacés » dans l'ensemble de données d'origine, ce qui leur permet d'être à nouveau sélectionnés.

- ***Le sous-échantillonnage*** aléatoire implique la sélection aléatoire d'exemples de la classe majoritaire à supprimer de l'ensemble de données d'apprentissage. Cela a pour but de réduire le nombre d'exemples dans la classe majoritaire dans la version transformée de l'ensemble de données d'apprentissage. Ce processus peut être répété jusqu'à ce que la distribution de classe souhaitée soit obtenue, par exemple un nombre égal d'exemples pour chaque classe.

4.3.2 Les mesures de performances :

- ***Matrice de confusion :*** Une matrice de confusion, aussi appelée matrice d'erreur est une matrice N x N utilisée pour évaluer les performances d'un modèle de classification, où N est le nombre de classes cibles. La matrice compare les valeurs cibles réelles avec celles prédites par le modèle d'apprentissage automatique. Cela nous donne une vision globale de la performance de notre modèle de classification et des types d'erreurs qu'il commet. Elle comporte 4 valeurs essentielles :

  - ***Vrais positifs (True Positif : TP)*** Ce sont des cas dans lesquels nous avons prédit oui (la personne est malade), et elle est malade
    — La valeur prédite correspond à la valeur réelle.
    — La valeur réelle était positive et le modèle a prédit une valeur positive.

  - ***Vrais négatifs (True Négatif : TN)*** Ce sont des cas dans lesquels nous avons prédit non (la personne n'est pas malade), et elle n'est pas malade.
    — La valeur prédite correspond à la valeur réelle.
    — La valeur réelle était négative et le modèle a prédit une valeur négative.

  - ***Faux positif (False positif : FP) - Erreur de type 1***
    — La valeur prédite a été faussement prédite.
    — La valeur réelle était négative mais le modèle prédit une valeur positive.

  - ***Faux négatif (False Négatif : FN) - Erreur de type 2***
    — La valeur prédite a été faussement prédite.
    — La valeur réelle était positive mais le modèle a prédit une valeur négative.

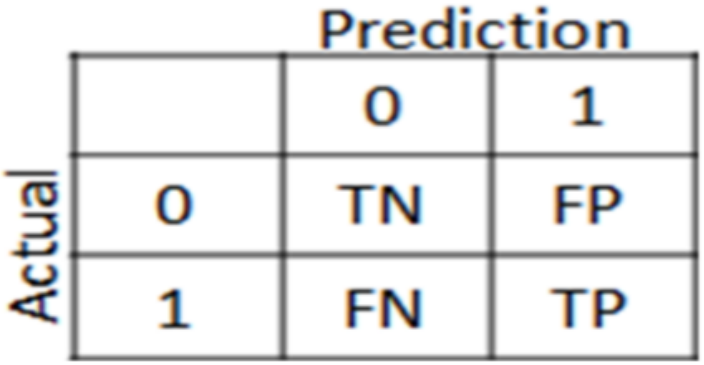


FIGURE 4.7 – Matrice de confusion.

- ***Balanced accuracy :*** Est une métrique que l'on peut utiliser pour évaluer la qualité d'un classificateur. Il est particulièrement utile lorsque les classes sont déséquilibrées, c'est-à-dire que l'une des deux classes apparaît beaucoup plus souvent que l'autre. Cela se produit souvent dans de nombreux contextes tels que la détection d'anomalies et la présence d'une maladie. Balanced accuracy normalise les prévisions vraies positives et vraies négatives par le nombre d'échantillons positifs et négatifs, respectivement, et divise leur somme par deux.

$$\text{Balanced accuracy} = \frac{TPR + TNR}{2}$$

- ***Précision :*** La précision est le rapport des observations positives correctement prédites au total des observations positives prévues.

- ***Recall :*** Le rappel est le rapport des observations positives correctement prédites à toutes les observations dans la classe réelle - oui .

- ***F1 Score :*** F1 score est la moyenne pondérée de la précision et du rappel. Par conséquent, ce score prend en compte à la fois les faux positifs et les faux négatifs.

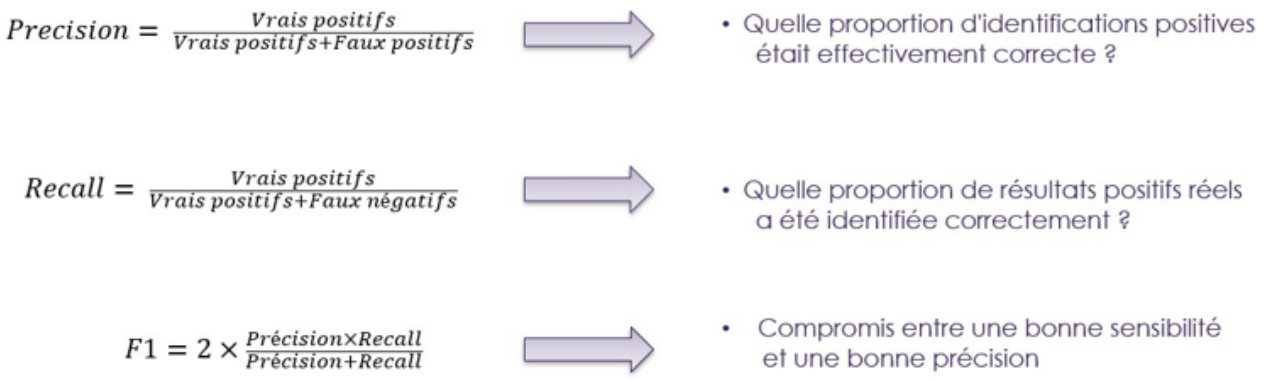


FIGURE 4.8 – Métriques de mesure.

4.4 Présentation des outils utilisés :

4.4.1 Anaconda :

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS [36].

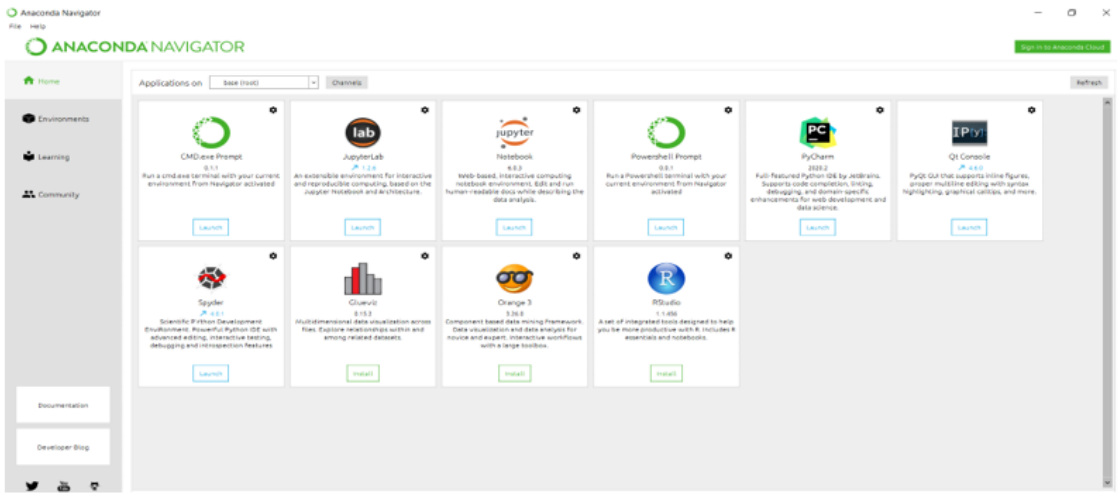


FIGURE 4.9 – Interface Anaconda.

4.4.2 Jupyter :

Jupyter est une application web utilisée pour programmer, initialement développés pour les langages de programmation Julia , Python et R (d'où le nom Jupyter), et supporte près de 40 langages. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks qui sont utilisés en science des données pour explorer et analyser des données. La cellule est l'élément de base d'un notebook jupyter. Elle peut contenir du texte formaté au format markdown ou du code informatique qui pourra être exécuté.

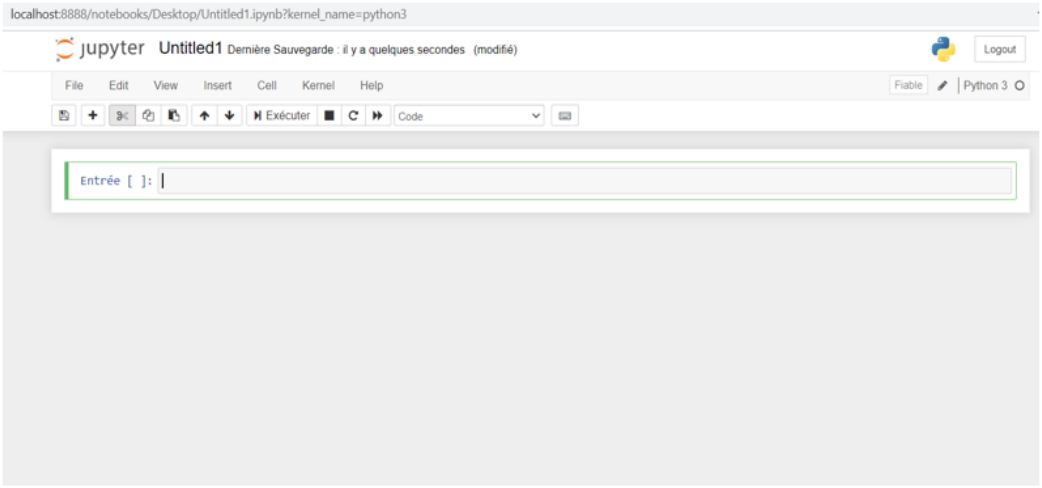


FIGURE 4.10 – Interface jupyter.

4.4.3 Python :

Python est un langage de programmation interprété. Il est utilisé pour de nombreuses applications différentes. Il est utilisé dans certains lycées et collèges comme langage de programmation d'introduction car Python est facile à apprendre, mais il est également utilisé par des développeurs de logiciels professionnels dans des endroits tels que Google, la NASA et Lucasfilm Ltd. Ainsi python est le langage le plus utilisé dans le domaine du Machine Learning et de la science des données.

FIGURE 4.11 – Logo Python.

4.4.4 TensorFlow :

TensorFlow est une bibliothèque open source de logiciels pour le flux de données et la programmation différentielle pour diverses tâches. De même, TensorFlow est utilisé dans l'apprentissage automatique par les réseaux de neurones. Développé par Google en 2011 sous le nom de DistBelief, TensorFlow a été officiellement lancé en 2017 gratuitement. La bibliothèque est capable de fonctionner sur plusieurs processeurs et disponible sur plusieurs plateformes, y compris mobiles. Le nom vient de tableaux multidimensionnels appelés tenseurs (Tensor), couramment utilisés dans les réseaux de neurones. Aujourd'hui, cette bibliothèque est utilisée par les grandes entreprises [37] comme Intel, Twitter, Lenovo...

FIGURE 4.12 – Logo TensorFlow.

4.4.5 Keras :

Keras est une bibliothèque open source d'apprentissage profond pour le Python, capable de s'exécuter sur TensorFlow. Il est écrit par Francis Chollet, membre de l'équipe Google. Keras est utilisé dans un grand nombre de startups, de laboratoires de recherche [38] (dont le CERN, Microsoft Research et la NASA) et de grandes entreprises telles que Netflix, Yelp, Square, Uber, Google, etc. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro Electronic Intelligent Robot Operating System). En 2017, l'équipe TensorFlow de Google a pris la décision de fournir un support pour Keras et de l'intégrer dans la bibliothèque principale de TensorFlow. Il présente un ensemble d'abstractions de plus haut niveau et plus intuitives qui facilitent la configuration des réseaux neuronaux [39].

FIGURE 4.13 – Logo Keras.

4.4.6 Scikit-learn :

Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle comprend des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment NumPy et SciPy.

FIGURE 4.14 – Logo scikit-learn.

4.5 Algorithmes utilisés :

Après avoir préparé notre architecture de travail, et lu plusieurs articles qui traitent des problématiques qui ressemble à la nôtre, on a choisi de tester trois algorithmes de Machine Learning qui sont la Régression Logistique, Random Forest et SVM et deux algorithmes de Deep Learning qui sont le perceptron multicouche (MLP) et CNN, afin de les comparer et de choisir le plus optimal pour notre problématique.

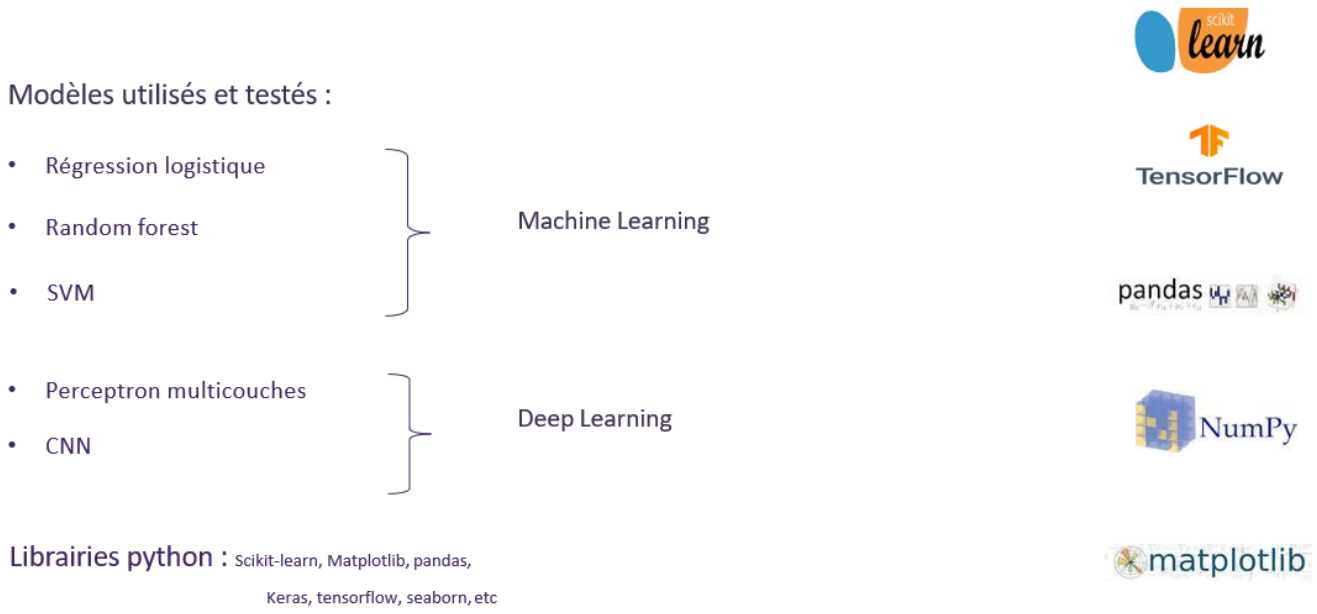


FIGURE 4.15 – Différents algorithmes utilisés.

4.5.1 Algorithmes de Machine Learning :

- **APPROCHE 1 : Algorithme Random Forest**
  - ***Utilisation de Random Forest pour la classification binaire :***

    Pour définir si une personne est malade ou pas, on va utiliser l'algorithme de Random Forest pour faire la classification binaire sur la base de données qui contient deux classes : 0 qui correspond à « pas malade », et 1 qui correspond à « malade », pour faire cela il faut :

    — ***Charger les bibliothèques nécessaires :***

```
import numpy
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, label_ranking_average_precision_score, label
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
import matplotlib.pyplot as plt

# fix random seed for reproducibility
numpy.random.seed(7)

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.utils import shuffle
```

FIGURE 4.16 – Chargement des bibliothèques en python.

— ***Séparer les features et les labels :***

— *Les features :* les features sont définies comme des variables indépendantes individuelles qui agissent en tant qu'entrée. Les modèles de prédiction utilisent les features pour effectuer des prédictions.

— *Les labels :* les labels sont la sortie finale, c'est-à-dire le résultat de la prédiction.

Dans notre cas les features prennent les colonnes de 0 à 186 et les labels la colonne 187.

```
X = data.iloc [:, 0:187] .values
y = data.iloc [:, 187] .values
```

FIGURE 4.17 – Séparation des features et labels.

— ***Diviser l'ensemble de données en 80% pour l'entraînement et 20% pour le test :***

Nous allons diviser notre ensemble de données en un échantillon d'entraînement de 80% qui permet à l'algorithme de s'entrainer dessus et un échantillon de test de 20% qui est inconnu pour l'algorithme, qui consiste à tester et à évaluer l'algorithme pour sa capacité de prédiction sur des données nouvelles, pour faire cela on utilise la fonction : `train_test_split`.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.2, random_state = 0,
                                                     stratify = y)
```

FIGURE 4.18 – Séparation de données en train et en test.

— ***Entrainer le modèle :***

Pour entrainer notre modèle prédictif, on a passé par une étape qui s'appelle la recherche des hyperparamètres, qui consiste à chercher les paramètres les plus optimaux de chaque algorithme avec des méthodes de Gridsearch ou de RandomizerSearch.
Pour l'algorithme random forest, la recherche d'hyperparamètres nous a sélectionné les paramètres suivants :
— n_estimators : 15
— criterion='entropy'
— max_depth=10
— class_weight="balanced"

pour l'entraînement du modèle, on utilise la fonction : `fit()`

```
from sklearn.ensemble import RandomForestClassifier

classifier= RandomForestClassifier(n_estimators= 15,criterion='entropy',
                                   max_depth=10,class_weight="balanced")
classifier.fit(X_train, y_train)
```

FIGURE 4.19 – Entraînement du modèle Random Forest.

— ***Prédire le résultat du test :***

Pour prédire la classe des nouvelles instances de données à l'aide de notre modèle de classification finalisé dans scikit-learn, on utilise la fonction : `predict()`.

```
y_pred= classifier.predict(X_test)
```

FIGURE 4.20 – Prédiction du résultat du test.

— ***Résultas Obtenus :***

— ***La matrice de confusion :*** La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. ¨

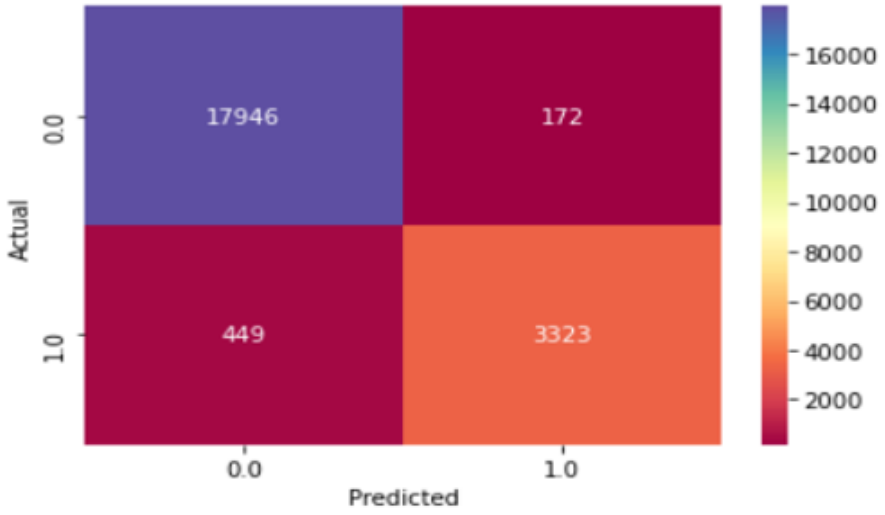


FIGURE 4.21 – La matrice de confusion Random Forest Binaire.

Les différentes valeurs de la matrice de confusion sont les suivantes :

Vrai positif (TP) = 3323; ce qui signifie que 3323 personnes de classe positive ont été correctement classés par le modèle.

Vrai négatif (TN) = 17946; ce qui signifie que 17946 personnes de classe négative ont été correctement classés par le modèle.

Faux positif (FP) = 449; ce qui signifie que 449 personnes de classe négative ont été incorrectement classés comme appartenant à la classe positive par le modèle.

Faux négatif (FN) = 172; ce qui signifie que 172 personnes de classe positive ont été incorrectement classés comme appartenant à la classe négative par le modèle.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.935735841871778
```

— ***Le rapport de classification :***

```
print(classification_report( y_test.round(), y_pred.round()))
```

	precision	recall	f1-score	support
0.0	0.98	0.99	0.98	18118
1.0	0.95	0.88	0.91	3772
accuracy			0.97	21890
macro avg	0.96	0.94	0.95	21890
weighted avg	0.97	0.97	0.97	21890

FIGURE 4.22 – Résultats Random Forest Binaire.

Pour la classification binaire, l'algorithme Random Forest montre une précision de 95 % , un recall de 88% et un f1-score de 91% pour les personnes malades, ce qui est un bon résultat, qui va nous permettre de savoir si une personne est malade.

- ***Utilisation de Random Forest pour la classification multi classe :***

Pour définir le type de la maladie d'une personne qui était déjà désigné comme malade lors de la classification binaire, on va utiliser l'algorithme Random Forest pour faire la classification multi classe sur la base de données qui contient 4 classes : 1, 2, 3 et 4 qui correspondent à un type de maladie, pour faire cela il faut :
les étapes sont les mêmes que celles de la classification binaire sauf :

— ***Le chargement de la base de données :*** on fait un filtre pour récupérer seulement les lignes qui ont leurs classes qui correspondent soit à 1, 2, 3 ou 4.

```
ndf= df[df [187] == 1.0 ]
ndff= df[df [187] == 2.0 ]
ndfff= df[df [187] == 3.0 ]
ndffff= df[df [187] == 4.0 ]
```

FIGURE 4.23 – Sélectionner seulement les labels[1,2,3,4].

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

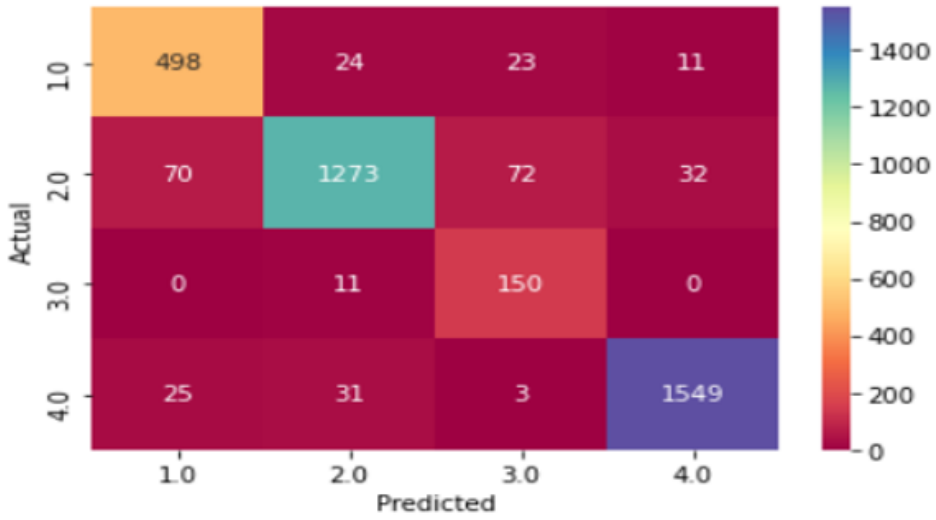


FIGURE 4.24 – La matrice de confusion Random Forest Multi.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.913715919969794
```

— ***Le rapport de classification :***
La figure suivante montre le taux de précision, de recall et de f1-score de chaque type de maladie :

```
print(classification_report( y_test.round(), y_pred.round()))
```

	precision	recall	f1-score	support
1.0	0.84	0.90	0.87	556
2.0	0.95	0.88	0.91	1447
3.0	0.60	0.93	0.73	161
4.0	0.97	0.96	0.97	1608
accuracy			0.92	3772
macro avg	0.84	0.92	0.87	3772
weighted avg	0.93	0.92	0.92	3772

FIGURE 4.25 – Résultats Random Forest Multi.

- **APPROCHE 2 : Algorithme Régression Logistique :**

  - ***Utilisation de la Régression Logistique pour la classification binaire :***

    Pour définir si une personne est malade ou pas, on va utiliser l'algorithme de Régression Logistique pour faire la classification binaire, pour faire cela :
    Il faut passer par les mêmes étapes que celles de la classification binaire avec l'algorithme Random forest, sauf le modèle qui sera différent :

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(C=1.0, class_weight="balanced",penalty='l2')
clf.fit(X_train, y_train)
```

FIGURE 4.26 – Entraînement du modèle Régression Logistique.

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

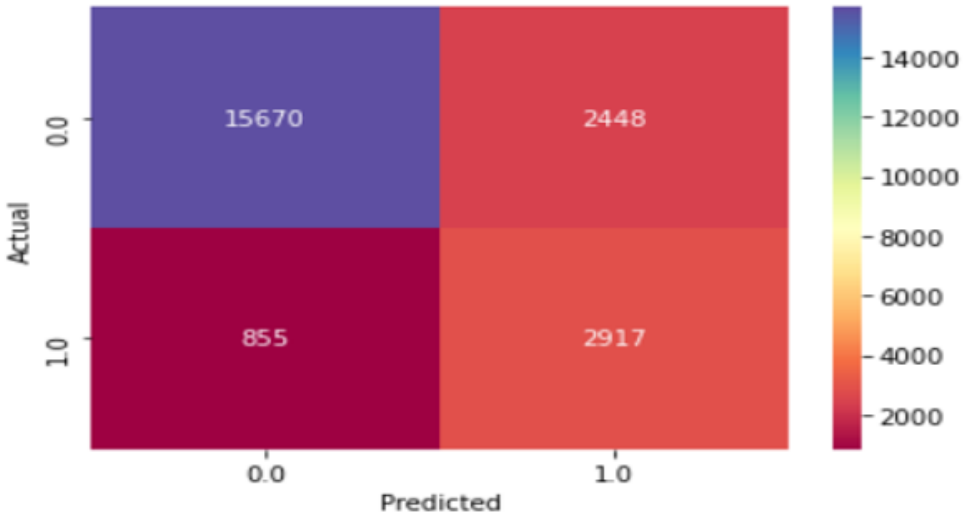


FIGURE 4.27 – La matrice de confusion Régression Logistique Binaire.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.8201514795724084
```

— ***Le rapport de classification :***

```
print(classification_report( y_test.round(), y_pred.round()))
```

```
              precision    recall  f1-score   support

         0.0       0.95      0.87      0.91     18118
         1.0       0.55      0.77      0.64      3772

    accuracy                           0.85     21890
   macro avg       0.75      0.82      0.77     21890
weighted avg       0.88      0.85      0.86     21890
```

FIGURE 4.28 – Résultats Régression Logistique Binaire.

Pour la classification binaire avec l'algorithme Régression logistique, le taux de précision est de 55 %, le Recall est de 77% et le f1-score est de 64% pour les personnes malades, qui n'est pas un bon résultat par rapport à l'algorithme Random Forest.

- ***Utilisation de la Régression Logistique pour la classification multi classe :***

Pour définir le type de la maladie d'une personne qui était déjà désigné comme malade lors de la classification binaire, on va utiliser l'algorithme Régression Logistique pour faire la classification multi classe :
les étapes sont les mêmes que celles de la classification multi classe avec l'algorithme Random Forest.

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

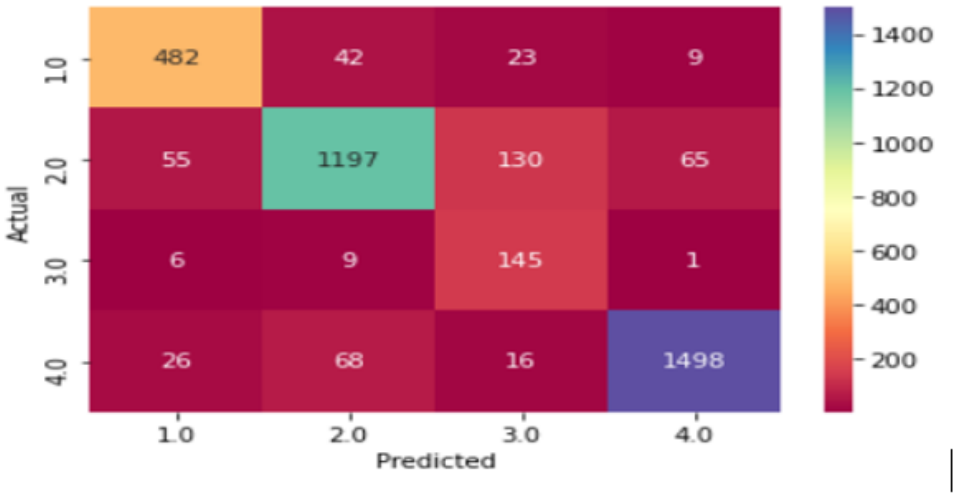


FIGURE 4.29 – La matrice de confusion Régression Logistique Multi.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.8815870954424263
```

— ***Le rapport de classification :***
La figure suivante montre le taux de précision, de recall et de f1-score de chaque type de maladie :

```
print(classification_report( y_test.round(), y_pred.round()))
```

	precision	recall	f1-score	support
1.0	0.85	0.87	0.86	556
2.0	0.91	0.83	0.87	1447
3.0	0.46	0.90	0.61	161
4.0	0.95	0.93	0.94	1608
accuracy			0.88	3772
macro avg	0.79	0.88	0.82	3772
weighted avg	0.90	0.88	0.89	3772

FIGURE 4.30 – Résultats Régression Logistique Multi.

- **APPROCHE 3 : Algorithme SVM :**

  - ***Utilisation de SVM pour la classification binaire :***

    Pour définir si une personne est malade ou pas, on va utiliser l'algorithme SVM pour faire la classification binaire, pour faire cela :
    Il faut passer par les mêmes étapes que celles de la classification binaire avec l'algorithme Random forest et Régression Logistique, sauf le modèle qui sera différent :

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

FIGURE 4.31 – Entraînement du modèle SVM .

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

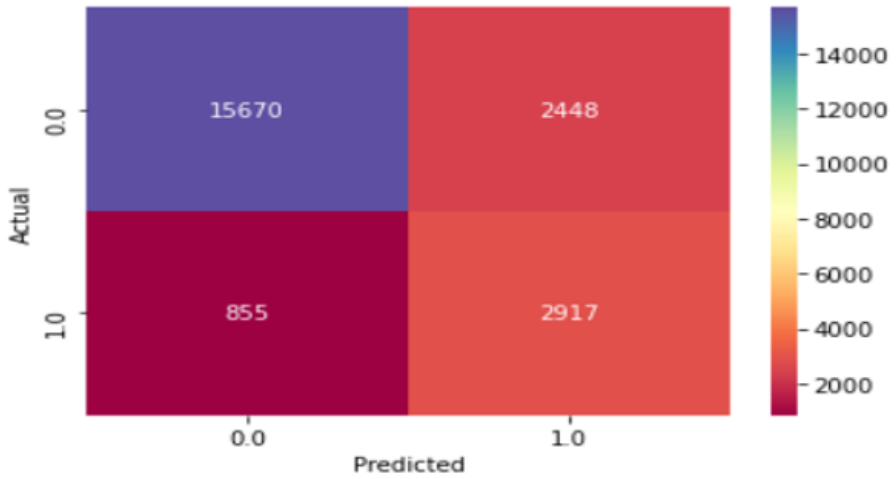


FIGURE 4.32 – La matrice de confusion SVM Binaire.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.9204457885779298
```

— ***Le rapport de classification :***

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0.0	0.97	1.00	0.98	18118
1.0	0.98	0.84	0.91	3772
accuracy			0.97	21890
macro avg	0.97	0.92	0.94	21890
weighted avg	0.97	0.97	0.97	21890

FIGURE 4.33 – Résultats SVM Binaire.

..

Pour la classification binaire avec l'algorithme SVM, le taux de précision est de 98 % , le recall est de 84% et le f1-score est de 91% pour les personnes malades, ce qui est un bon résultat, qui va nous permettre de savoir si une personne est malade.

- ***Utilisation de SVM pour la classification multi classe :***

Pour définir le type de la maladie d'une personne qui était déjà désigné comme malade lors de la classification binaire, on va utiliser l'algorithme SVM pour faire la classification multi classe :
les étapes sont les mêmes que celles de la classification multi classe avec Random Forest et Régression Logistique.

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

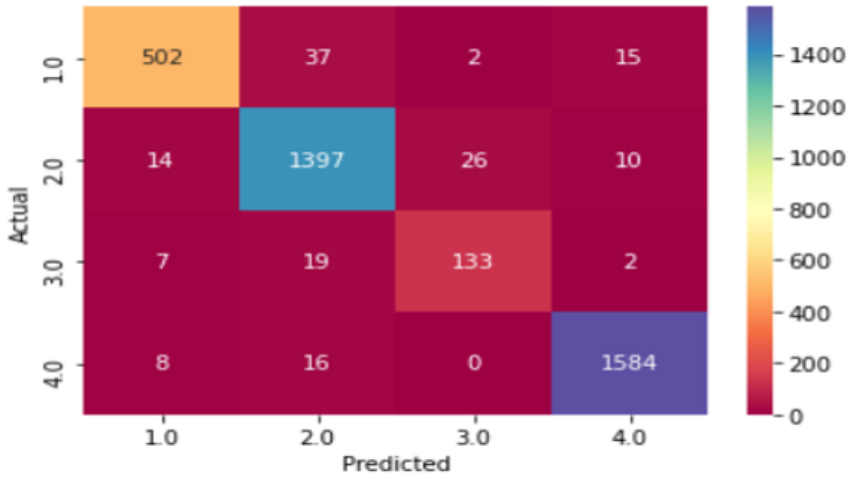


FIGURE 4.34 – La matrice de confusion SVM Multi.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test, y_pred)
```

```
0.9198712577640915
```

— ***Le rapport de classification :***
La figure suivante montre le taux de précision, de recall et de f1-score de chaque type de maladie :

```
print(classification_report( y_test.round(), y_pred.round()))
```

	precision	recall	f1-score	support
1.0	0.95	0.90	0.92	556
2.0	0.95	0.97	0.96	1447
3.0	0.83	0.83	0.83	161
4.0	0.98	0.99	0.98	1608
accuracy			0.96	3772
macro avg	0.93	0.92	0.92	3772
weighted avg	0.96	0.96	0.96	3772

FIGURE 4.35 – Résultas SVM Multi.

4.5.2 Algorithmes de Deep Learning :

- **APPROCHE 1 : Algorithme MLP**

  - ***Utilisation de MLP pour la classification binaire :***

    Pour définir si une personne est malade ou pas, on va utiliser l'algorithme MLP pour faire la classification binaire pour faire cela :
    On va suivre les mêmes étapes que celles de la classification binaire avec les algorithmes de machine learning sauf :

    — ***Le modèle MLP :***

```
from keras.models import Sequential

# Import `Dense` from `keras.layers`
from keras.layers import Dense

# Initialize the constructor
model = Sequential()

# Add an input layer
model.add(Dense(12, activation='relu', input_shape=(187,)))

# Add one hidden layer
#model.add(Dense(12, activation='relu'))
#model.add(Dense(8, activation='relu'))
model.add(Dense(8, activation='relu'))

# Add an output layer
model.add(Dense(1, activation='sigmoid'))
```

FIGURE 4.36 – Le modèle MLP.

— ***Compiler le modèle :***

Avant d'entraîner le modèle, nous devons configurer le processus d'apprentissage en appelant la méthode compile qui contient trois arguments :

- ***Loss :*** Il s'agit de la fonction de coût que le modèle va utiliser pour minimiser les erreurs. Nous utiliserons l'entropie croisée binaire « binary_crossentropy » pour la classification binaire et « catégorical_crossentropy » pour la classification multi classe.
- ***Optimizer :*** Nous utiliserons l'algorithme efficace de descente de gradient stochastique « Adam », car il se règle automatiquement et donne de bons résultats dans un large éventail de problèmes.
- ***Metrics :*** Dans le cas d'un problème de classification, nous utilisons metrics=['accuracy'].

```
model.compile(loss='binary_crossentropy',optimizer='adam',
              metrics=['accuracy'])
```

FIGURE 4.37 – Compilation du modèle.

— ***Entrainer le modèle :***

La recherche d'hyperparamètres avec les méthodes de Gridsearch nous a sélectionné 12 époques pour l'entrainement du modèle et un batch size de 50.

```
his = model.fit(X_train,y_train,
            epochs=12,
            batch_size=50,
            validation_data=(X_test, y_test),
            verbose=1)
```

FIGURE 4.38 – Entraînement du modèle MLP.

— ***Résultats Obtenus :***

— **La matrice de confusion :**

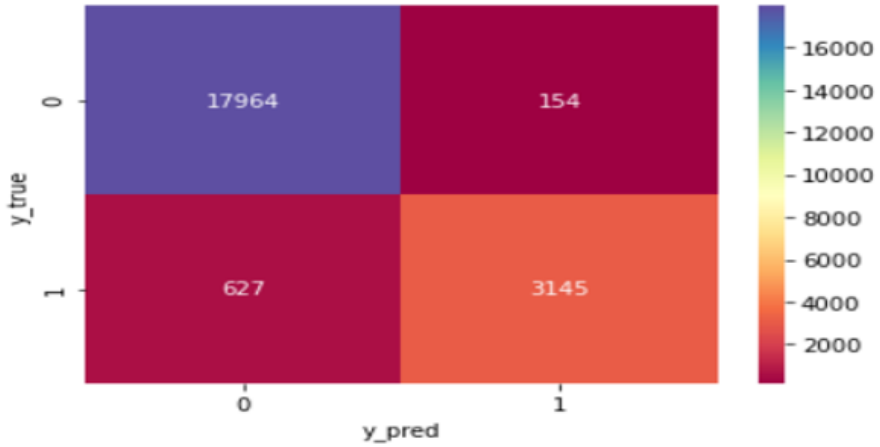


FIGURE 4.39 – La matrice de confusion MLP Binaire.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test.round(), y_pred.round())
```

```
0.9055511489016799
```

— **Le rapport de classification :**

```
print(classification_report( y_test.round(), y_pred.round()))
```

```
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98     18118
         1.0       0.96      0.83      0.89      3772

    accuracy                           0.96     21890
   macro avg       0.96      0.91      0.93     21890
weighted avg       0.96      0.96      0.96     21890
```

FIGURE 4.40 – Résultats MLP Binaire .

Pour la classification binaire avec l'algorithme MLP, le taux de précision est de 96 %, le Recall est de 83% et le f1-score est de 89% pour les personnes malades, ce qui est un bon résultat, qui va nous permettre de savoir si la personne est malade.

- ***Utilisation de MLP pour la classification multi classe :***

  Pour définir le type de la maladie d'une personne qui était déjà désigné comme malade lors de la classification binaire, on va utiliser l'algorithme MLP pour faire la classification multi classe :
  les étapes sont les mêmes que celles de la classification multi classe avec les algorithmes de Machine Learning.

  — ***Résultats Obtenus :***

  — ***La matrice de confusion :***

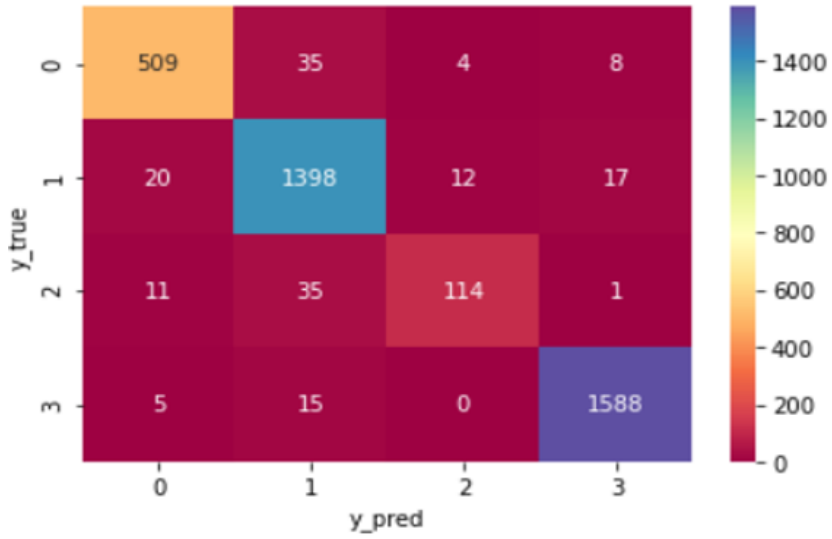


FIGURE 4.41 – La matrice de confusion MLP Multi.

  — ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test.argmax(axis=1), y_pred.argmax(axis=1))
```

```
0.9023500778823496
```

— ***Le rapport de classification :***
La figure suivante montre le taux de précision, de recall et de f1-score de chaque type de maladie :

```
print(classification_report(y_test.argmax(axis=1), y_pred.argmax(axis=1)))
```

	precision	recall	f1-score	support
0	0.93	0.92	0.92	556
1	0.94	0.97	0.95	1447
2	0.88	0.71	0.78	161
3	0.98	0.99	0.99	1608
accuracy			0.96	3772
macro avg	0.93	0.89	0.91	3772
weighted avg	0.96	0.96	0.96	3772

FIGURE 4.42 – Résultats MLP Multi.

- **APPROCHE 2 : Algorithme CNN**

  - ***Utilisation de CNN pour la classification binaire :***

    Pour définir si une personne est malade ou pas, on va utiliser l'algorithme CNN pour faire la classification binaire, pour faire cela :
    On va suivre les mêmes étapes que celles de la classification binaire avec l'algorithme MLP sauf le modèle qui sera différent :

    — ***Le modèle CNN :***

```
K.clear_session()

inp = Input(shape=(feature, depth))
C = Conv1D(filters=32, kernel_size=2, strides=1)(inp)

C11 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(C)
A11 = Activation("relu")(C11)
C12 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(A11)
S11 = Add()([C12, C])
A12 = Activation("relu")(S11)
M11 = MaxPooling1D(pool_size=2, strides=2)(A12)


C21 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(M11)
A21 = Activation("relu")(C21)
C22 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(A21)
S21 = Add()([C22, M11])
A22 = Activation("relu")(S11)
M21 = MaxPooling1D(pool_size=2, strides=2)(A22)


C31 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(M21)
A31 = Activation("relu")(C31)
C32 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(A31)
S31 = Add()([C32, M21])
A32 = Activation("relu")(S31)
M31 = MaxPooling1D(pool_size=2, strides=2)(A32)


C41 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(M31)
A41 = Activation("relu")(C41)
C42 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(A41)
S41 = Add()([C42, M31])
A42 = Activation("relu")(S41)
M41 = MaxPooling1D(pool_size=2, strides=2)(A42)


C51 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(M41)
A51 = Activation("relu")(C51)
C52 = Conv1D(filters=32, kernel_size=2, strides=1, padding='same')(A51)
S51 = Add()([C52, M41])
A52 = Activation("relu")(S51)
M51 = MaxPooling1D(pool_size=2, strides=2)(A52)

F1 = Flatten()(M51)

D1 = Dense(32)(F1)
A6 = Activation("relu")(D1)
D2 = Dense(32)(A6)
D3 = Dense(2)(D2)
A7 = Activation("sigmoid")(D3)

model = Model(inputs=inp, outputs=A7)
```

FIGURE 4.43 – Le modèle CNN 1D.

— ***Résultats Obtenus :***

— ***La matrice de confusion :***

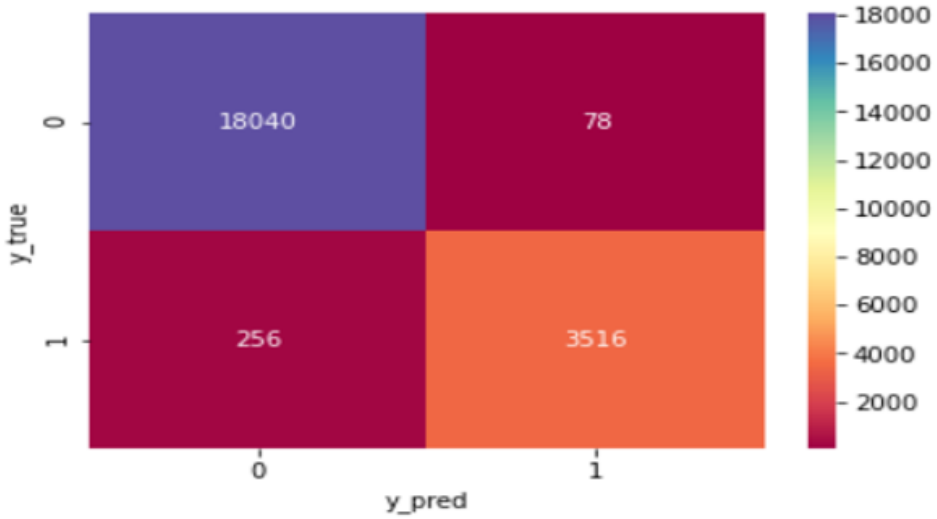


FIGURE 4.44 – La matrice de confusion CNN Binaire.

— ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_test.argmax(axis=1), y_pred.argmax(axis=1))
```

```
0.9632345814295984
```

— ***Le rapport de classification :***

```
print(classification_report(y_test.argmax(axis=1), y_pred.argmax(axis=1)))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     18118
           1       0.98      0.93      0.95      3772

    accuracy                           0.98     21890
   macro avg       0.98      0.96      0.97     21890
weighted avg       0.98      0.98      0.98     21890
```

FIGURE 4.45 – Résultats CNN Binaire.

Pour la classification binaire avec l'algorithme CNN, le taux de précision est de 98 %, le recall est de 93% et le f1-score est de 95% pour les personnes malades, ce qui est un très bon résultat qui va nous permettre de savoir si la personne est malade.

- ***Utilisation de CNN pour la classification multi classe :***

  Pour définir le type de la maladie d'une personne qui était déjà désigné comme malade lors de la classification binaire, on va utiliser l'algorithme CNN pour faire la classification multi classe :
  les étapes sont les mêmes que celles de la classification multi classe avec l'algorithme MLP.

  — ***Résultats Obtenus :***

  — ***La matrice de confusion :***

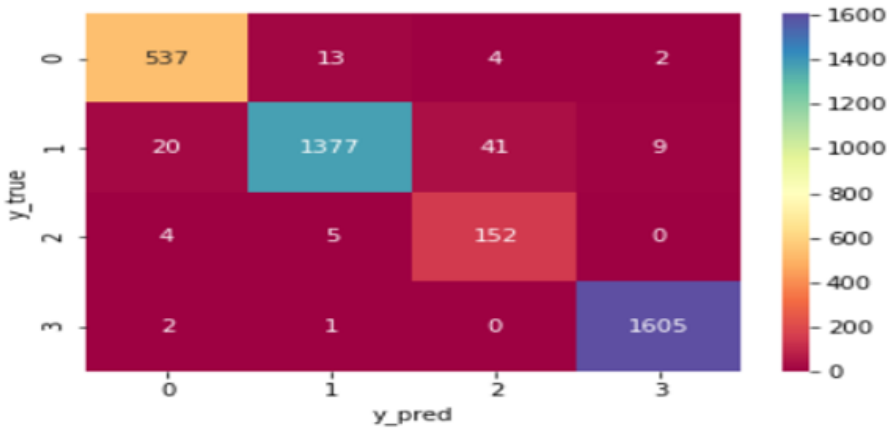


FIGURE 4.46 – La matrice de confusion CNN Multi.

  — ***Balanced accuracy :***

```
from sklearn.metrics import balanced_accuracy_score

balanced_accuracy_score(y_testt.argmax(axis=1), y_predd.argmax(axis=1))
```

```
0.9351806808769887
```

  — ***Le rapport de classification :***
  La figure suivante montre le taux de précision, de recall et de f1-score de chaque type de maladie :

```
print(classification_report(y_testt.argmax(axis=1), y_predd.argmax(axis=1)))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       556
           1       0.96      0.99      0.97      1447
           2       0.95      0.75      0.84       161
           3       0.99      0.99      0.99      1608

    accuracy                           0.98      3772
   macro avg       0.97      0.92      0.94      3772
weighted avg       0.98      0.98      0.97      3772
```

FIGURE 4.47 – Résultats CNN Multi.

4.5.3 Comparaison des algorithmes utilisés :

- **Comparaison pour la classification binaire :**

Algorithmes / Métriques	Random Forest		Régression logistique		SVM		MLP		CNN	
	0	1	0	1	0	1	0	1	0	1
Précision	0.98	0.95	0.95	0.55	0.97	0.98	0.97	0.96	0.99	0.98
Recall	0.99	0.88	0.87	0.77	1.00	0.84	0.99	0.83	1.00	0.93
F1 score	0.98	0.91	0.91	0.64	0.98	0.91	0.98	0.89	0.99	0.95
Matrice de confusion →Vrai Positif (VP)	3323		2917		3186		3145		**3516**	
Balanced Accuracy	0.93		0.82		0.92		0.90		**0.96**	

FIGURE 4.48 – Tableau comparatif des algorithmes utilisés pour la classification binaire.

Après avoir testé plusieurs algorithmes de Machine Learning et de Deep Learning pour la classification binaire, on remarque que la plupart des algorithmes donnent des bons résultats, mais le plus optimal et le plus performant qui répond le mieux à notre problématique c'est le CNN qui nous a donné une meilleure balanced accuracy de **96%** comparé aux autres algorithmes testés : Random Forest, Régression Logistique, SVM et MLP qui ont donné une balanced accuracy de 93%, 82%, 92% et 90% respectivement .
Pour la première étape qui concerne la classification binaire dans notre architecture, on va choisir le ***CNN***.

- **Comparaison pour la classification multi classe :**

Algorithmes / Métriques	Random Forest				Régression logistique				SVM				MLP				CNN			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Précision	0.84	0.95	0.60	0.97	0.85	0.91	0.46	0.95	0.95	0.95	0.83	0.98	0.93	0.94	0.88	0.98	0.97	0.96	0.95	0.99
Recall	0.90	0.88	0.93	0.96	0.87	0.83	0.90	0.93	0.90	0.97	0.83	0.99	0.92	0.97	0.71	0.99	0.96	0.99	0.75	0.99
F1 score	0.87	0.91	0.73	0.97	0.86	0.87	0.61	0.94	0.92	0.96	0.83	0.98	0.92	0.95	0.78	0.99	0.96	0.97	0.84	0.99
Matrice de confusion →Vrai Positif (VP)	498	1273	150	1549	482	1197	145	1498	502	1397	133	1584	509	1398	114	1588	537	1377	152	1605
Balanced Accuracy	0.91				0.88				0.91				0.90				**0.93**			

FIGURE 4.49 – Tableau comparatif des algorithmes utilisés pour la classification multi classe.

Après avoir testé plusieurs algorithmes de Machine Learning et de Deep Learning pour la classification multi classe, on remarque que la plupart des algorithmes donnent des bons résultats, mais le plus optimal qui répond le mieux à notre problématique c'est le CNN qui nous a donné une meilleure balanced accuracy de **93%** comparé aux autres algorithmes testés : Random Forest, Régression Logistique, SVM et MLP qui ont donné une balanced accuracy de 91%, 88%, 91% et 90% respectivement .
D'après cette analyse, l'algorithme le plus optimal pour la classification multi classe dans notre problématique est le ***CNN***.

Après avoir choisi le meilleur algorithme qui répond le mieux à notre problématique, notre architecture finale se base sur l'algorithme CNN pour la classification binaire et multi classe des arythmies ECG, comme la montre la figure suivante :

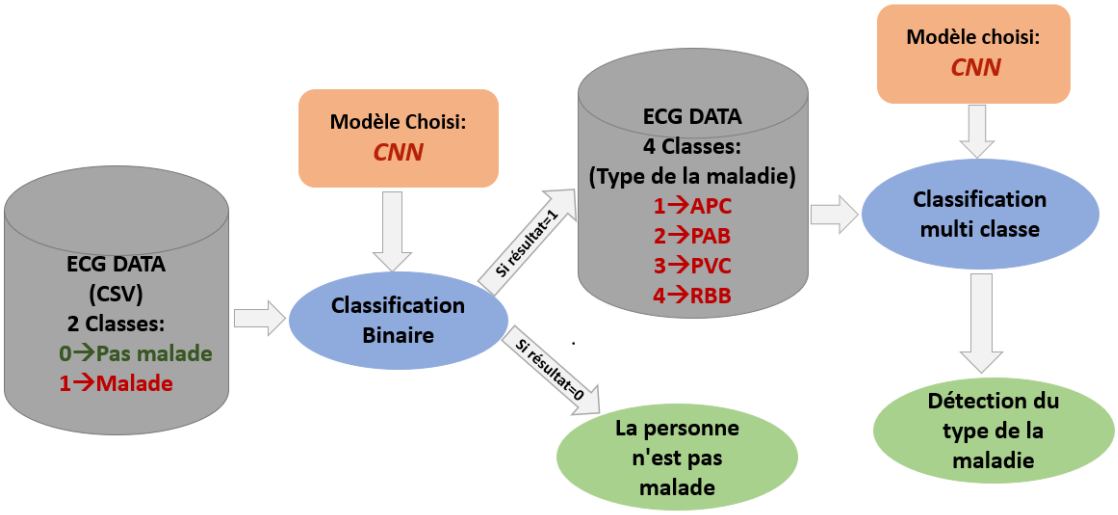


FIGURE 4.50 – L'architecture finale de la classification des arythmies ECG .

4.6 Comparaison entre les méthodes de l'état de l'art et celles du système utilisé :

On remarque d'après les méthodes utilisées dans l'état de l'art et celles utilisées dans ce projet, les deux méthodes ont donné les mêmes résultats dans le côté où le CNN qui a eu le meilleur résultat de la classification des données ECG, ce qui prouve encore plus l'efficacité de CNN pour résoudre ce type de problématique.
L'importance de tester plusieurs algorithmes pour une problématique, pour connaître l'algorithme le mieux adapté, parce qu'il n'existe pas une règle générale pour dire qu'un algorithme est meilleur qu'un autre.

4.7 Conclusion :

Nous avons présenté dans ce chapitre notre système réalisé pour la classification des arythmies ECG basée sur deux étapes, classification binaire et multi classe avec les algorithmes de Machine Learning et de Deep Learning, pour cela on a utilisé trois algorithmes de Machine Learning qui sont Random Forest, Régression Logistique et SVM et deux algorithmes de Deep Learning qui sont MLP et CNN, et on a afficher et analyser les différents résultats pour chaque algorithme. La comparaison des résultats trouvés a montré que l'algorithme CNN donne les meilleurs résultats pour la classification binaire et multi classe.

Conclusion

Dans ce travail, on a proposé une approche pour la classification automatique des arythmies ECG qui se base sur deux étapes, une classification binaire pour définir si une personne est malade ou pas et une classification multi classe afin de détecter le type de la maladie cardiaque pour les personnes malades, en testant différents algorithmes de Machine Learning et de Deep Learning et en sélectionnant les algorithmes les plus optimaux pour notre problématique.
Dans notre cas, on a pu constater que l'algorithme CNN a donné des meilleurs résultats concernant la classification binaire et multi classe comparé aux autres algorithmes testés.

Afin d'aboutir à ces résultats, j'ai passé beaucoup de temps à lire et à étudier les publications et les articles pour voir ce qui se fait de mieux en matière de classification et pour pouvoir concevoir ma propre approche.
D'après les expériences vues dans les articles ou par ma propre expérience, on remarque que c'est important de tester plusieurs algorithmes pour savoir qui est le mieux adapté pour une certaine problématique, parce qu'il n'y a pas une règle générale qui dit qu'un algorithme est meilleur qu'un autre.

Grâce à l'avancée de l'intelligence artificielle, comme le Machine Learning et le Deep Learning ça nous a permet de traiter des problématiques plus complexes et d'avoir de bons résultats ce qui est très important dans le domaine de la santé où la précision peut sauver des vies humaines.
L'intelligence artificielle offre de nombreux avantages pour explorer le paysage de la santé. Cela promet d'innover dans le système de santé pour un avenir meilleur.

Pour finir, avant de passer aux perspectives, ce travail m'a permis d'élargir mes champs de connaissances, de confronter la formation théorique à la pratique et le temps passé à lire des articles m'a servi d'une bonne initiation à la recherche.
La formation ingénierie des systèmes d'information (ISI) acquise à travers mon cursus à l'université de UMMTO, m'a été utile sur le plan fonctionnel et technique. En effet, la réalisation de ce mémoire a été une expérience très enrichissante. Ce dernièr était une opportunité de mettre en pratique le savoir que j'ai acquis, et de travailler en toute autonomie sur un projet de recherche qui est intéressant et qui a de la valeur dans l'IA et dans le domaine médical.

Comme perspectives nous pouvons citer :

- Tester nos modèles sur d'autres ensembles de données.
- Créer une API et intégrer le modèle prédictif a l'intérieur.
- Intégrer le modèle prédictif dans un système embarqué afin de suivre le rythme cardiaque d'une façon continue et en temps réel.

Bibliographie

[1] «Cardiovascular diseases (CVDs),» World Health Organization, 17 May 2017. [En ligne]. Lien : https ://www.who.int/en/news-room/fact-sheets/detail/cardiovasculardiseases-(cvds), 24 juin 2020

[2] Obraska P., Perlemuter L., Quevauvilliers J Médecine, appareille cardiovasculaire » Edition-Masson Tome II 1968.

[3] H. Staff, «How the Heart Works,» Alberta, 22 July 2018. [En ligne]. Lien : https ://my-health.alberta.ca/Health/pages/conditions.aspx ?hwid=tx4097abc, 28 juin 2020

[4] GOLDBERGER, Ary L., et al.Goldberger : clinical electrocardiography : a simplified approach. 2013.

[5] H. MEZIANE, « Acquisition de signaux Electrocardiogrammes (ECG) à l'aide de la carte DSPACE », mémoire de fin d'études, Université Abou Bekr Belkaid –Tlemcen, le 02 Juillet 2003

[6] R. BENALI, « Analyse du signal ECG par réseau adaptif d'ondelettes en vue de la reconnaissance de pathologies cardiaques », Doctorat en Sciences, Université Abou Bekr Belkaid, Avril 2013

[7] D.ATTIAS, N.LELLOUCHE « ikb CARDIOLOGIE VASCULAIRE »Edition VG,7e edition 2016.

[8] Chlo-AgatheAzencott « Introduction au Machine Learning », 2013

[9] David G.Kleinbaum, MitchelKlein.« Introduction to Logistic Regression », 2010

[10] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001

[11] P. Mahé :" Noyaux pour graphes et Support Vector Machines pour le criblage virtuel de molécules ". Rapport de stage, DEA MVA 2002/2003, Septembre 2003

[12] On the Origin of Deep Learning. Haohan Wang, Bhiksha Raj. 2017.

[13] Top 15 Deep Learning applications that will rule the world in 2018 and beyond, Vartul Mittal, 3 Oct 2017.

[14] Python Deep Learning. Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca. 2019

[15] Python Deep Learning. Valentino Zocca, Gianmario Spacagna, Daniel Slater, Peter Roelants. 2017.

[16] Fundamentals of Deep Learning Designing Next-Generation Machine Intelligence Algorithms. Nikhil Buduma. 2017

[17] https ://sebastianraschka.com/faq/docs/softmax regression.html, 20 Juillet 2020.

[18] http ://cs231n.github.io/convolutional-networks/overview, 25 juillet 2020.

[19] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks.," ICML (3), vol. 28, pp. 1310–1318, 2013.

[20] TensorFlow for Deep Learning. Reza Bosagh Zadeh, Bharath Ramsundar. 2017

[21] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078.

[22] http ://www.asimovinstitute.org/neural-network-zoo/, 30 juillet 2020.

[23] Deep Learning. Ian Goodfellow, Yoshua Bengio, Aaron Courville. MIT Press. 2016

[24] Vijaya Arjunan R, School of Engineering and IT « ECG SIGNAL CLASSIFICATION BASED ON STATISTICAL FEATURES WITH SVM CLASSIFICATION », 2016

[25] Tae Joon Jun, Hoang Minh Nguyen, Daeyoun Kang et al « ECG arrhythmia classification using a 2-D convolutional neural network »,2017

[26] Moody GB, Mark RG (2001). The impact of the MIT-BIH arrhythmia database. IEEE Engineering in Medicine and Biology Magazine 20(3) :45-50

[27] Russakovsky O, Deng J, Su H et al (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115(3) :211-252

[28] Krizhevsky A, Sutskever I,Hinton GE (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems pp 1097-1105

[29] Szegedy C, Liu W, Jia Y et al (2015). Going deeper with convolutions. IEEE conference on computer vision and pattern recognition pp 1-9

[30] Simonyan K, Zisserman A (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv :1409.1556.

[31] He K, Zhang X, Ren S et al (2016). Deep residual learning for image recognition. IEEE conference on computer vision and pattern recognition pp 770-778

[32] Wiley Sons 14. Huang G, Liu Z, Weinberger KQ et al (2017). Densely connected convolutional networks. IEEE conference on computer vision and pattern recognition pp 3

[33] Shraddha Singh, Saroj Kumar Pendey, Urja Pawar et al, « Classification of ECG Arrhythmia using Recurrent Neural Networks» Data Science (ICCIDS 2018)

[34] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.

[35] https ://www.physionet.org/content/mitdb/1.0.0/

[36] https ://www.anaconda.fr/

[37] https ://www.tensorflow.org/

[38] https ://www.linkedin.com/in/fchollet, 12 août 2020

[39] http ://KERAS.io