Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2021

Détection avancée de maladie oculaire

Etudiant : Quentin Nater

Professeur : Dr Dominique Genoud

Travail rendu le : 13.08.2021

Site web : www.hevs.ch

RÉSUMÉ

Ce document, rédigé dans le contexte du travail de Bachelor de la filière d'informatique de gestion de la HESSO//Valais, a pour but de déceler des caractéristiques dans des données temporelles récoltées depuis des lentilles intelligentes posées sur l'œil de patients pouvant être atteints de glaucome, dans l'optique de prédire si la série de données récoltées est classifiée comme étant saine ou malade. Cette prédiction pourrait évaluer si les caractéristiques apprises sont relatives à des maladies oculaires spécifiques.

Ce travail de Bachelor, mené en collaboration avec l'entreprise Sensimed S.A.[1] et le groupe Dude-lab de l'Institut Informatique de Gestion du Valais, repose sur des recherches sur les différentes technologies de machine learning. Il testera des modèles de classification traditionnels dans des outils de machine learning comme KNIME et des modèles de deep learning basés sur la technologie Keras.

Finalement, ce document révélera les résultats, les analyses et les conclusions des différents outils pour démontrer l'utilisabilité des caractéristiques apprises à partir des datasets temporels.

Mots-clés : glaucome, machine learning, réseau de neurones, deep learning, CNN 1D

[1] Entreprise suisse qui vise à améliorer la prise en charge globale du glaucome : https://www.sensimed.ch/

AVANT-PROPOS

Le thème d'intelligence artificielle devient de plus en plus présent dans tous les domaines de la société. La technologie et la médecine sont particulièrement concernées par ces changements. La technologie médicale [Medtech] est une spécialité de la Suisse qui s'impose comme leader dans le domaine.

Naviguant sur cette vague innovante, Sensimed, une compagnie focalisée sur la révolution de la prise en charge du glaucome, développe des lentilles intelligentes munies de capteurs. Le système de surveillance oculaire continue (SENSIMED Triggerfish®) est donc un appareil qui transmet des informations sur les changements de volume oculaire du porteur tout au long de la journée et de la nuit. Ce dispositif capte ainsi les changements spontanés de l'œil, ce qui fournit aux médecins des informations précieuses qui peuvent les aiguiller dans la détermination du diagnostic d'un éventuel glaucome.

Afin de tirer parti des données de ces lentilles intelligentes, Sensimed souhaite utiliser des techniques d'apprentissage automatique pour déterminer des caractéristiques à partir de séries chronologiques produites par le dispositif. Finalement, leur objectif est de pouvoir évaluer si ces caractéristiques apprises sont prédictives de maladies oculaires spécifiques.

Ce thème de travail de Bachelor propose de construire une plateforme capable de gérer des datasets temporels en utilisant l'outil KNIME et de tester des outils de machine learning pour extraire des éléments de ces datasets.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont soutenu, guidé et aidé lors de la réalisation de ce document :

Dr Dominique Genoud pour m'avoir guidé en tant que responsable lors de ce travail de Bachelor. Il m'a montré la voie dans ce domaine qui m'était encore inconnu et a toujours été disponible pour m'aiguiller ou me conseiller. Il m'a permis de donner mon maximum et de dépasser mes limites dans cette épreuve.

M. Jérôme Treboux pour avoir toujours été derrière moi à chaque étape de ce projet. Il a investi de son temps sans compter pour me permettre de découvrir le sujet, de m'améliorer, de me corriger et de présenter les meilleurs résultats possible.

Mme. Michelle Constantin, M. John Nater et M. Nicolas Constantin pour leurs coups de plume et leur temps lors de la relecture de ce document.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

ANN	Artificial Neural Network
API	Application Programming Interface
AUC	Area Under the Curve
CART	Classification and Regression Trees
CNN 1D	Convolutional Neural Network 1D
CNN 2D	Convolutional Neural Network 2D
CRISP-DM	Cross Industry Standard Process for Data Mining
CSV	Comma-Separated Values
FN	False Negatives
FP	False Positives
FTRL	Follow The Regularized Leader
GAN	Generative Adversarial Network
GBDT	Stochastic Gradient Boosted Decision Trees
GBT	Gradient Boosted Tree
GS	Gaussian Noise
HES-SO	Haute École Spécialisée de Suisse Occidentale
ID3	Iterative Dichotomiser 3
Medtech	Technologie Médicale
MLP	MultiLayer Perceptron
MSE	Mean Square Error
ReLu	Rectified Linear
RL	Reinforcement Learning

RNN	Recurrent Neural Network
Rpop	Resilient Backpropagation
SE	Standard Error
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True Negatives
TP	True Positives

GLOSSAIRE

Algorithme	Suite finie d'instructions et d'opérations visant à résoudre une série de problèmes
Algorithme top-down	Une approche dite descendante part d'une matière brute et la raffine dans l'optique d'y apporter une valeur ajoutée
Application Programming Interface	Ensemble de définitions et de protocoles qui facilitent la création et l'intégration de logiciels d'applications
Apprentissage supervisé	Apprentissage avec des labels pour identifier des classes lors d'une procédure d'entraînement et de test
Artificial Neural Network	Terme générique désignant un réseau simple de neurones
Big Data	Quantité énorme de données, qui sont caractérisées par leur volume, leur vélocité et leur variété
Convolution	Opération par laquelle deux fonctions sont mises dans un rapport suggérant une sorte d'enroulement de l'une sur l'autre
Data Science	Science visant à l'extraction de connaissances d'ensembles de données
Dataset	Terme anglophone désignant un jeu de données. Ensemble de valeurs où chaque valeur est associée à un attribut et à une observation
Deep Learning	Extraction de patterns à partir de données en utilisant des réseaux de neurones
Framework	Ensemble de composants qui forment les bases d'un logiciel
Gradient	Désigne un vecteur qui représente la variation d'une fonction par rapport à la variation de ses différents paramètres
Hypermétropie	Trouble de la vision où les objets proches sont flous
Inputs	Terme anglophone désignant la ou les données d'entrée dans un réseau de neurones
Intelligence Artificielle	Toutes techniques qui permettent à un ordinateur d'imiter le

	comportement humain
Itération	Action de répéter un processus
Kernel	Dans le contexte d'un réseau de convolution, désigne une matrice qui se déplace sur des données pour en extraire des caractéristiques
Label	Étiquette référant à la classe d'une série de données
Machine Learning	Habilité d'apprendre sans être expressément programmé (apprentissage automatique)
Méthodologie Agile	Méthodologie de gestion de projet centrée sur l'idée d'itération
Métrique	Valeur ou mesure d'un paramètre
Open Source	Logiciels libres avec possibilité de redistribution, d'accès au code source et de création de travaux dérivés
Output	Terme anglophone désignant la ou les données de sortie d'un réseau de neurones
Pattern	Modèle simplifié ou répété d'une structure
TensorFlow	Plateforme open source de Google dédiée au machine learning
Weight	Terme anglophone désignant le poids d'un neurone dans un réseau. Il correspond à l'importance du neurone en question.

1. Introduction

1.1. Problématique et contexte

L'entreprise Sensimed a mis au point des lentilles intelligentes avec un capteur intégré (figure 1), qui sont capables de récolter des données capitales sur l'œil, dans le but de déterminer si un patient atteint de glaucome aurait un risque élevé de progression de la maladie. Actuellement, l'entreprise récolte les données, mais elle n'a toujours pas de processus pour en extraire des caractéristiques.

Dans ce contexte, nous allons implémenter des modèles de prédiction basés sur le machine learning pour, dans un premier temps, détecter si une suite de données est saine ou malade et dans un second temps, extraire des caractéristiques dans les résultats de ces modèles.

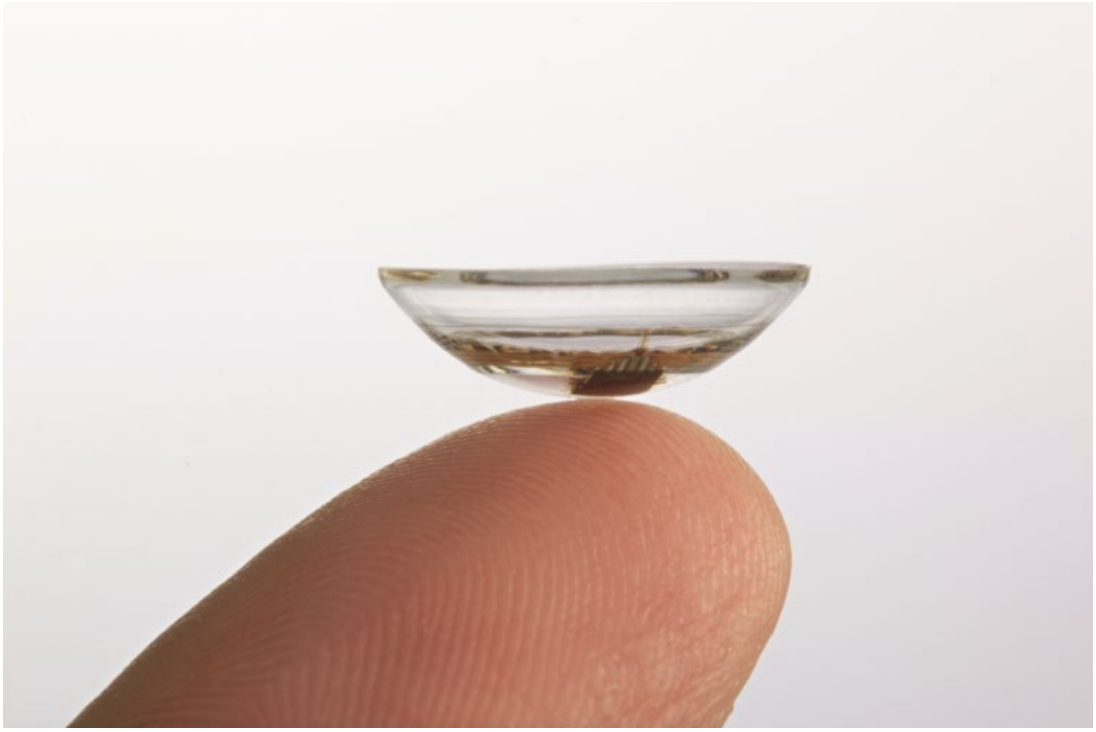


Figure 1 - SENSIMED Triggerfish®, lentilles intelligentes avec un capteur intégré - Trouvée sur Sensimed[2]

[2] Sensimed S.A.: https://www.sensimed.ch/

1.2. Glaucome

Avant de comprendre comment détecter un glaucome chez un patient, il est important de comprendre la maladie en elle-même. Nous allons donc décrire les caractéristiques de cette maladie oculaire.

1.2.1. Qu'est-ce que le glaucome ?

Le glaucome est une maladie grave qui se traduit la plupart du temps par une lésion des fibres du nerf optique. Ces derniers permettent de transmettre les informations visuelles de l'œil au cerveau. Une augmentation anormale de la pression à l'intérieur de l'œil est le plus souvent la cause de cette maladie qui peut progresser jusqu'à la cécité. A ce stade, la maladie est irréversible. (Hôpitaux Universitaires Genève, 2015).

1.2.2. Population à risque

Toute la population de plus de 40 ans peut être concernée par cette maladie. L'âge est un facteur clé, plus de 15% des personnes de plus de 70 ans sont touchées par cette maladie. (Hôpitaux Universitaires Genève, 2015).

Les facteurs suivants peuvent aussi influencer la survenue de la maladie quel que soit l'âge du patient : des antécédents familiaux, l'origine ethnique, une forte myopie ou hypermétropie, une hypertension artérielle, des problèmes vasculaires, du diabète, des traumatismes antérieurs de l'œil, des traitements prolongés à base de cortisone ou d'autres médicaments. (Lazcano-Gomez, et al., 2016).

1.2.3. Symptômes

Le problème de cette maladie est qu'elle n'a aucun symptôme (douleur, perte de vue, …) lorsqu'elle s'installe de manière progressive. Actuellement, seuls les ophtalmologues peuvent dépister cette anomalie à temps. Les patients qui se plaignent d'une diminution notable de la vision sont déjà à un stade avancé de la maladie. (Hôpitaux Universitaires Genève, 2015).

1.2.4. Traitement

Au début de la maladie, la vision périphérique est atteinte. Puis, petit à petit, la vision centrale finit par être touchée. On peut néanmoins en ralentir l'évolution grâce à des diagnostics précoces et des traitements adaptés comme des gouttes, des thérapies au laser ou de la chirurgie. (Lazcano-Gomez, et al., 2016).

1.3. Méthodologie

Aucune méthodologie n'a été imposée lors de ce projet. Un travail de Bachelor étant un travail personnel, l'utilisation de la méthodologie Agile ne pourra pas s'appliquer telle quelle. Nous en utiliserons cependant des concepts. Des réunions hebdomadaires seront organisées entre l'étudiant, l'institut de recherche et le représentant de Sensimed pour évaluer les résultats et l'avancée du projet. Des procès-verbaux seront rédigés à chaque réunion et des supports de présentations seront également mis à disposition de tous les intervenants, tout en respectant le vœu de confidentialité du client. Finalement, un journal de bord résumera chacun des travaux.

1.3.1. Cross Industry Standard Process for Data Mining

Le Cross Industry Standard Process for Data Mining [CRISP-DM] (figure 2) est un modèle de processus de datamining qui décrit une suite d'approches permettant d'orienter les travaux d'exploration de données. (IBM, 2020).

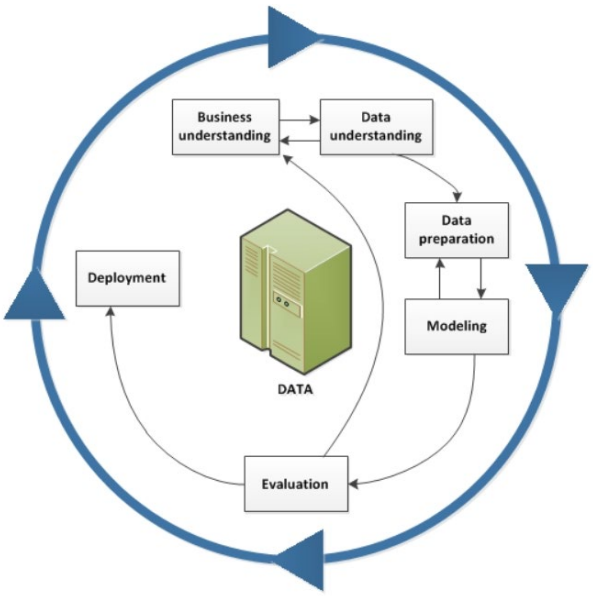


Figure 2 - Diagramme du Cross Industry Standard Process for Data Mining - Trouvée sur IBM[3]

La méthode de travail CRISP-DM sera utilisée dans chaque expérimentation de ce projet. Ce modèle est flexible et personnalisable, ce qui nous permet d'apporter plus d'importance à chacune des étapes selon le besoin. De plus, cette méthode est adaptée au machine learning.

En se basant sur les demandes du client et les données du travail de Bachelor, les étapes ci-dessous seront réalisées itérativement le long de ce projet.

[3] Présentation générale de CRISP-DM : https://www.ibm.com/docs/fr/spss-modeler/SaaS?topic=dm-crisp-help-overview

1.3.1.1. Compréhension du métier

Lors de cette étape, une analyse complète des besoins du client sera menée. Nous devons impérativement comprendre les besoins métiers et les exigences de Sensimed. De plus, un objectif clair sera fixé avec le superviseur et le client.

1.3.1.2. Compréhension des données

Une décomposition des données présentes dans les datasets fournis par le client sera opérée. Les particularités des données devront être relevées, comme le manque de données, leur corrélation et la répartition des classes.

1.3.1.3. Préparation des données

Des datasets propres à chaque expérimentation devront être créés, altérés ou affinés dans l'optique d'atteindre les meilleurs résultats en termes de temps et de qualité. Plusieurs datasets seront générés à partir des données de base conférées par Sensimed au début du projet.

1.3.1.4. Modélisation

Une fois que les datasets sur mesure ont été générés pour l'expérimentation, un modèle de classification sera construit pour atteindre l'objectif de l'expérimentation. Selon les résultats du modèle de classification, les datasets seront à nouveau analysés et altérés, jusqu'à ce que le modèle généré corresponde au besoin.

1.3.1.5. Évaluation

Au terme de chaque expérimentation, une séance sera organisée avec le superviseur et le client représentant de Sensimed. Lors de cette séance, les résultats du modèle seront présentés et analysés. Ces résultats suivront une série de prérogatives, propres à l'évaluation du modèle en question. Si les objectifs ne sont pas atteints, nous reprendrons à l'étape une, « compréhension du métier », afin de replacer des objectifs clairs pour cette nouvelle itération. Au contraire, si les objectifs sont validés par le superviseur et le client, nous procéderons à un déploiement sur le serveur du client.

1.3.1.6. Déploiement

Le déploiement de notre environnement complet de travail pourra être effectué sur le serveur du client. De plus, l'ensemble des fonctionnalités implémentées seront disponibles sous la forme d'un projet à importer. En effet, le client désire pouvoir ajouter cette brique de fonctionnalités directement dans son environnement de travail.

2. État de l'art du machine learning

Un grand nombre de modèles statistiques ont été implémentés au fil du temps. Chacun d'entre eux recèle des avantages et des inconvénients. Cette grande diversité de modèles permet de créer plusieurs représentations différentes d'un même problème. Évidemment, un modèle qui s'applique dans un contexte ne s'applique pas nécessairement aussi bien dans un autre, d'où l'utilité d'avoir une vue d'ensemble des connaissances du machine learning. (Kiwi, et al., 2018).

La figure 3 démontre une vue d'ensemble du paysage du machine learning. Dans le cadre de ce projet, nous nous concentrerons sur les aspects relatifs à notre sujet.

Figure 3 - Tâches de « machine learning » par catégories et sous-catégories - Trouvée dans le livre IA : entreprise augmentée (Kiwi, et al., 2018)

2.1. Classification

La classification se traduit par l'action d'identifier des classes avec des labels, à l'aide d'un processus d'entraînement et de test. La classification trouve son sens dans de multiples contextes, la prédiction de courrier électronique en tant que « indésirable » ou « légitime » est un exemple parmi tant d'autres. (Awad & ELseuofi, 2011).

Pour analyser, affiner et comprendre le dataset donné, nous devons tout d'abord comprendre les fondements du machine learning supervisé (voir figure 3) pour atteindre les clés de la classification. Pour ce faire, la première étape consiste à comprendre les outils et algorithmes pour prédire la classification de nos patients malades ou sains.

Face au nombre conséquent d'algorithmes dans le domaine, nous nous concentrerons seulement sur une partie, afin d'en ressortir les principales caractéristiques correspondants aux besoins de ce travail de Bachelor.

2.1.1. Decision Tree

2.1.1.1. Description d'un Decision Tree

Le Decision Tree, arbre de décision en français, est un arbre construit sur la base de modèles de régression ou de classification dans le domaine de l'apprentissage supervisé (voir figure 3). Tout un chacun a déjà expérimenté sa logique. Par exemple : est-ce qu'il pleut ? Si oui, je prends un parapluie, sinon rien.

La figure 4 illustre un exemple de Decision Tree. Par rapport à son état de santé, à la matière en question et au fait d'avoir étudié, on peut prédire si un élève réussira ou non son examen.

predicateur			label
a étudié ?	**matière**	**est malade ?**	**a réussi ?**
Oui	Science	Oui	Non
Oui	Science	Non	Oui
Oui	Français	Oui	Non
Oui	Français	Non	Oui
Non	Science	Oui	Non
Non	Science	Non	Oui
Non	Français	Oui	Non
Non	Français	Non	Non

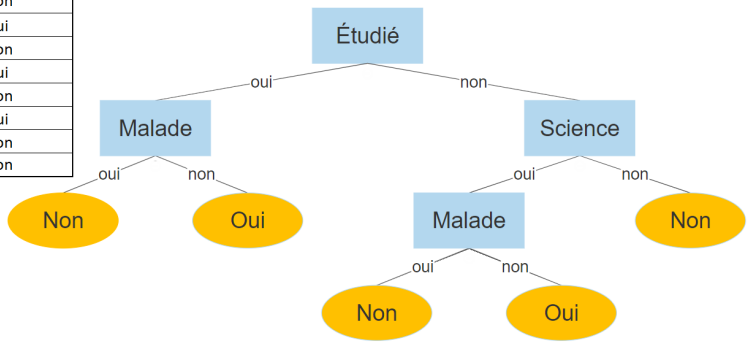


Figure 4 - Exemple d'un Decision Tree illustrant la réussite ou non d'un examen - Image de l'auteur

Comme le précise l'article « The clinical decision analysis using Decision Tree » (Bae, 2014), les Decision Trees ont été activement utilisés dans les années 90 et au début des années 2000. A présent, de nouvaux modèles ont été développés pour traiter ce genre de problèmes. Il reste

néanmoins intéressant de comprendre leur concept, celui-ci étant la base de la logique d'un Random Forest (voir chapitre 2.1.2), qui est toujours activement employé.

2.1.1.2. Algorithme de Classification and Regression Trees

Comme cité dans le papier « Comparison of Decision Tree algorithms for EMG signal classification using DWT » (Gokgoz, 2015), le cœur de l'algorithme d'un Decision Tree est basé sur l'Iterative Dichotomiser 3 [ID3] de J. R. Quinlan. L'ID3 est un algorithme de type top-down, qui part d'une matière brute et la raffine dans l'optique d'y apporter une valeur ajoutée. Voici son comportement :

1) Les données sont transmises à un nœud racine au sommet de l'arbre.

2) Chaque nœud posera une question servant à séparer un attribut du dataset duquel découlera une réponse vraie ou fausse.

3) Le nœud va ensuite partager le dataset d'entrée en deux datasets de sortie selon les réponses à la question précédente.

4) Ces nouveaux datasets vont devenir les données d'entrée de deux nouveaux nœuds enfants qui vont s'ajouter à l'arbre.

5) Nous poursuivrons le processus jusqu'à ce que toutes les données soient triées.

La figure 5 démontre un Decision Tree sur la prédiction d'une attaque cardiaque.

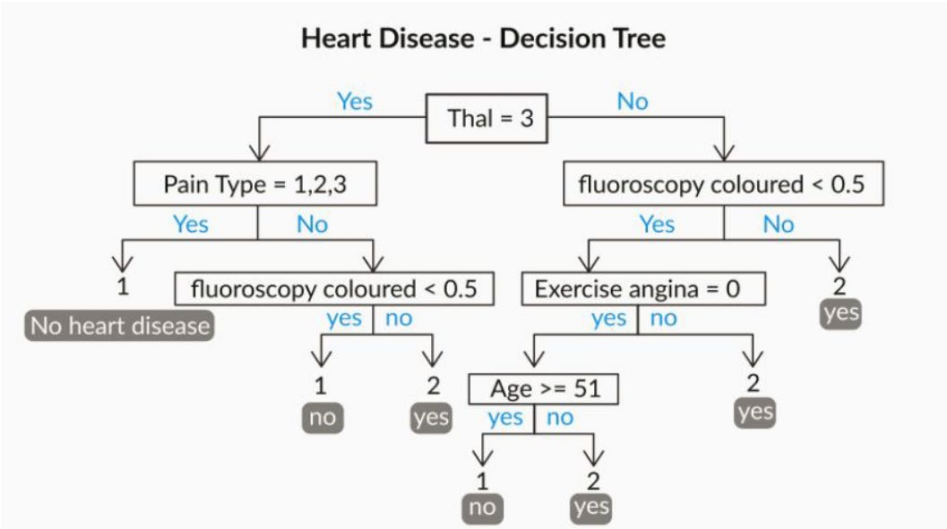


Figure 5 - Exemple de Decision Tree illustrant le diagnostic d'une maladie du cœur - Trouvée sur naivedatascientist[4]

[4] Heart Decease Decision Tree: https://naivedatascientist.co.in/2020/09/06/intro-to-decision-tree/

2.1.1.3. Gini Index

Gini Index est un algorithme de séparation de données qui se base sur l'impureté. L'impureté est définie par la chance d'être incorrect lorsqu'une classe est assignée à un attribut d'une manière aléatoire dans un même dataset. Gini Index prédit donc l'erreur. (Johnson, 2016).

2.1.1.4. Information Gain

L'Information Gain est un algorithme qui permet de trouver la meilleure question pour diviser un dataset. L'algorithme va donc tester toutes les possibilités et définir quelle séparation permet d'avoir le moins d'impureté, laquelle tend vers la branche la plus homogène. Information Gain améliore donc la précision. (Johnson, 2016).

2.1.2. Random Forest

2.1.2.1. Description d'un Random Forest

Un algorithme de classification de Random Forest est un ensemble de Decision Trees, que l'on nomme une forêt, chacun entraîné avec une partie différente des données d'un dataset. Le Random Forest maximise ainsi la variété des Decision Trees, permettant aux arbres avec de bons résultats de contrebalancer ceux avec de mauvais résultats. Chaque arbre de cette forêt aura droit à un vote, qui représente le résultat de sa classification. La majorité des voix fera pencher la classification. C'est pour ces raisons qu'un algorithme de Random Forest aura la plupart du temps une meilleure précision qu'un algorithme de Decision Tree. (Zhou, Zhou, Zhou, Yang, & Luo, 2014).

Ce modèle est très flexible, facile à entraîner et à configurer. Il pourra donc être utilisé dans de nombreux domaines de classification.

2.1.2.2. Bootstrapping

Bootstrapping définit l'action de découper le dataset d'origine en un nouveau dataset, qui est appelé bootstrapped dataset. Cette action permet de créer un nouveau dataset pour chaque arbre de la forêt.

2.1.2.3. Bagging

Bagging est le terme utilisé pour l'agrégation des décisions après un bootstrapping, soit l'action de sélectionner une partie différente des données du dataset pour chaque arbre.

2.1.2.4. Out-Of-Bag

Out-of-bag correspond à l'ensemble des données du dataset qui n'auraient pas été sélectionnées lors du bagging. Ce sont donc des données inutilisées lors de l'entraînement de l'algorithme. On peut utiliser ce out-of-bag pour tester le modèle de classification. L'out-of-bag error représente le taux d'erreurs des données testées dans un out-of-bag.

2.1.3. Gradient Boosted Tree

2.1.3.1. Description du Gradient Boosted Tree

Le Gradient Boosted Tree [GBT] est un modèle de classification supervisé de machine learning (voir figure 3). Tout comme le Random Forest, cet algorithme va combiner les résultats de plusieurs arbres individuels pour atteindre une prédiction. Toutefois, ces méthodes diffèrent sur la manière dont les arbres seront construits et sur la manière dont les résultats seront combinés.

Ce changement peut être résumé par le terme « boosting ». Le boosting est introduit dans le document de Freund et Schapire en 1990, qui définissait cette méthode par : « Converting a weak learning algorithm into one that achieves arbitrarily high accuracy » (1997). Le boosting fonctionne en appliquant séquentiellement des modules d'apprentissages faibles pour réassigner de manière répétée le poids des données d'entraînement. (Hastie, Tibshirani, & Friedman, 2009).

En d'autres termes, chaque nouvel arbre corrigera les erreurs des arbres précédents, grâce à l'évaluation d'une fonction de perte.

Après chaque itération de boost, les exemples mal classés verront leur poid augmenter et les exemples classés correctement verront leur poid diminuer. Après un nombre certain d'itération, les prédictions des séries de classifieurs faibles seront combinés par un vote majoritaire pondéré pour compter comme prédiction finale. (Krauss, AnhDo, & Huck, 2017).

Tout comme le Random Forest, le Gradient Boosted Tree est un modèle de prédiction polyvalent. Il est utilisé par exemple dans le classement de moteurs de recherche, dans l'analyse de données énergétiques ou même dans la gestion de données physiques.

2.1.3.2. Boosting Iterations

Boosting Iterations peut se résumer par le nombre d'arbres utilisés. Une valeur trop élevée pourrait pousser vers l'overfitting (voir chapitre 2.4.11). Par défaut, le nombre d'arbres est fixé à 100. Il est donc impératif de veiller à ne pas dépasser le seuil propre au dataset. (Brownlee, 2016).

2.1.3.3. Depth of the Tree

La depth of the tree, la profondeur d'un arbre en français, décrit le nombre de niveaux de la racine à la fin des branches. Comme le démontre la figure 6, une profondeur de trois nous permettrait de générer deux interactions. (Brownlee, 2016).

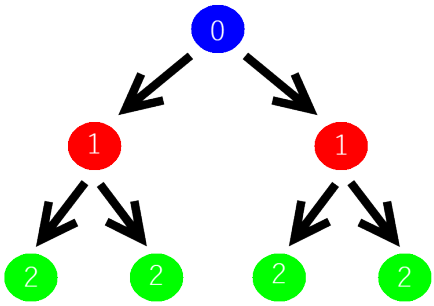


Figure 6 - Exemple d'un arbre de profondeur trois - Image de l'auteur

2.1.3.4. The Learning Rate

Le learning rate, le taux d'apprentissage en français, et le nombre d'arbres sont en relation inverse. Ils influencent ensemble le taux d'erreurs constant. (Krauss, AnhDo, & Huck, 2017).

Le document « The Elements of Statistical Learning : Data Mining, Inference, and Prediction » suggère un taux d'apprentissage standard inférieur à 0.1 (Hastie, Tibshirani, & Friedman, 2009).

2.1.3.5. The Subset of Features

Le subset of features, sous-ensemble de caractéristiques en français, correspond aux dimensions spatiales des données. Une gestion du nombre de dimensions et de classes est donc demandée selon les données. (Sourek, Hubacek, & Zelezny, 2017).

2.1.4. Support Vector Machine

2.1.4.1. Description du Support Vector Machine

Support Vector Machine [SVM] est un algorithme qui permet de gérer des problèmes de classification. Il est extrêmement utile lorsqu'une marge de séparation claire entre les données est traçable. Cet algorithme est également performant dans des espaces à multiples dimensions.

Les applications courantes du SVM sont, par exemple, la détection de visages, la catégorisation du texte, la classification d'images, la bio-informatique ou encore la reconnaissance de l'écriture manuscrite. (Issarane, 2020).

La figure 7 dévoile un exemple de SVM. Ce graphique montre les résultats des examens de différents élèves par rapport au temps passé à travailler à domicile et au temps passé à travailler en cours. La ligne L2, tracée par le SVM, se trouve à la séparation parfaite des étudiants qui ont passé leurs examens et ceux en échec. Cette ligne se nomme « decision boundry », étant la frontière qui limite les décisions. La marge définit la distance entre L1 et L3, soit la distance entre les étudiants qui ont le plus travaillé, mais qui n'ont pas passé leurs examens, et ceux qui ont le moins travaillé, mais qui ont réussi leurs examens. On appelle ces points à la limite d'une classe les

« support vectors ». Tous les autres points du dataset ne sont donc pas pertinents, n'impactant ni la marge ni la decision boundry. Le problème survient lorsque les données n'ont pas une marge nette.

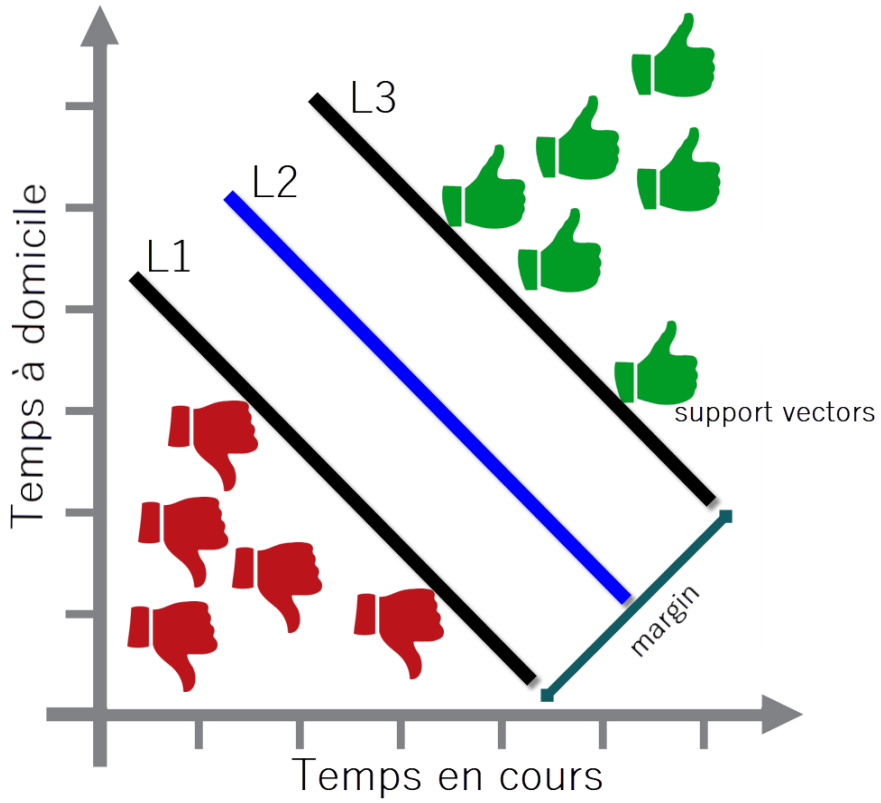


Figure 7 - SVM illustrant des élèves passant leurs examens (vert) ou non (rouge) - Image de l'auteur

Le SVM est tout à fait satisfaisant d'un point de vue théorique. L'apprentissage SV est basé sur des idées merveilleusement simples et fournit une intuition claire de ce qu'est l'apprentissage à partir d'exemples. Le Support Vector Machine peut aussi conduire à des performances élevées dans des applications pratiques. Cet algorithme peut donc, à ce sens, être à l'intersection de l'apprentissage théorique et pratique. (Hearst, Dumais, Osuna, Platt, & Scholkopf, 1998).

2.1.4.2. Soft Margin SVM

Le Soft Margin SVM fonctionne comme l'exemple ci-dessus, sauf qu'il tolère quelques erreurs. Comme le montre la figure 8, le Soft Margin SVM va conférer des pénalités d'une certaine importance à une erreur commise selon la distance entre l'erreur en question et la decision boundry.

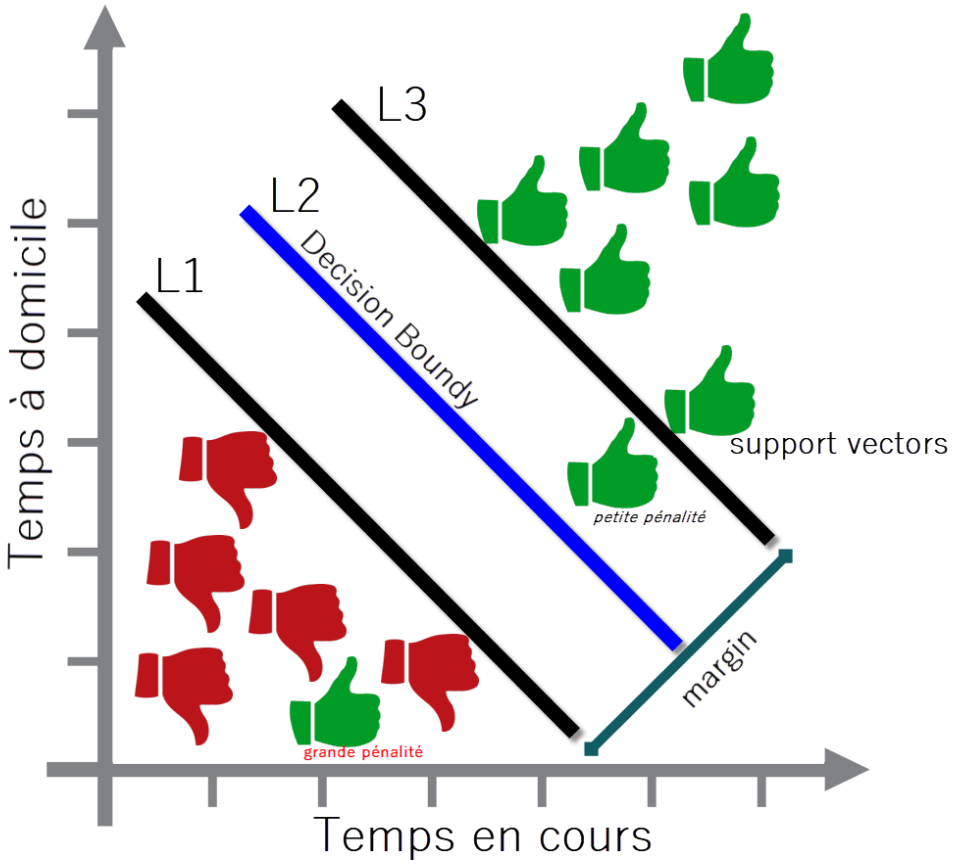


Figure 8 - Exemple d'un SVM avec des pénalités illustrant des élèves passant leurs examens (vert) ou non (rouge) - Image de l'auteur

2.1.5. Fuzzy Rule

2.1.5.1. Description du Fuzzy Rule

Fuzzy Rule est un algorithme qui offre l'aptitude de penser à la machine, utile pour la classification. Cet algorithme fonctionne sur le même principe qu'un être humain avec des réponses, « Oui », « Non », à la place de « True », « False ». Sa logique agit sous la forme d'un postulat, d'une implication et d'une conséquence. Le désavantage premier du Fuzzy Rule réside dans sa difficulté à être entraîné.

Postulat : la température est élevée.

Implication : SI la température est élevée ALORS la climatisation est activée.

Conséquence : La climatisation est active.

2.2. Deep Learning

Une autre solution de classification se nomme le deep learning. L'apprentissage profond est la troisième couche de la data science (figure 9). Le deep learning consiste à extraire des patterns depuis des données en utilisant des réseaux de neurones (Amini, 2021a). Nos recherches partant de zéro, nous allons commencer par recueillir les notions primaires de ce domaine.

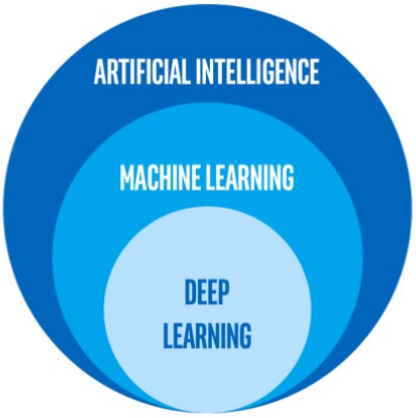


Figure 9 - Couche de la data science - Trouvée sur Intel[5]

2.2.1. Contexte du deep learning

Les réseaux de neurones existent depuis des dizaines d'années. Alors pourquoi est-ce que le domaine est autant d'actualité ? Premièrement, nous vivons à l'ère du Big Data (quantité énorme de données, qui sont caractérisées par leur volume, leur vélocité et leur variété), ce qui nous permet d'acquérir les volumes de données nécessaires au deep learning. Deuxièmement, ces modèles nécessitent généralement de grandes capacités de calcul (Kayid & Khaled, 2018). Ils ont donc besoin de matériel informatique de pointe comme les dernières cartes graphiques, qui fournissent d'excellents résultats (Intel, 2021). Finalement, de nouveaux frameworks permettant de construire et de développer facilement ces modèles, tel que TensorFlow[6] développé par Google, sont devenus accessibles aux particuliers.

Il existe plusieurs familles de deep learning, qui ont été implémentées pour résoudre des problèmes spécifiques et augmenter considérablement les résultats de leur domaine. Par exemple, le deep learning peut gérer la détection d'objets pour des voitures intelligentes (Gavrila & Philomin, 2002) ou permet à une machine d'apprendre de manière autonome pour battre les meilleurs champions d'échecs (Goodrich, 2021). Avec ses capacités, le deep learning peut apporter un plus dans tous les domaines, que ce soit en médecine, en ingénierie, en industrie et bien

[5] The Difference Between AI, ML and DL : https://www.intel.ca/content/www/ca/en/artificial-intelligence/posts/difference-between-ai-machine-learning-deep-learning.html

[6] TensorFlow: https://www.tensorflow.org/

d'autres. Nous allons donc réaliser l'état de l'art de ces différents types de deep learning. Toutefois, face au nombre importants d'algorithmes dans le domaine, nous allons nous concentrer sur une partie seulement, afin d'en ressortir les principales caractéristiques correspondants aux besoins du projet.

2.2.2. Neurone

Habituellement, comme le cite Eugene Charniak dans son livre « Introduction to Deep Learning » (2018), un neurone comporte plusieurs entrées, un corps de cellule et une seule sortie. La figure 10 illustre un neurone ordinaire.

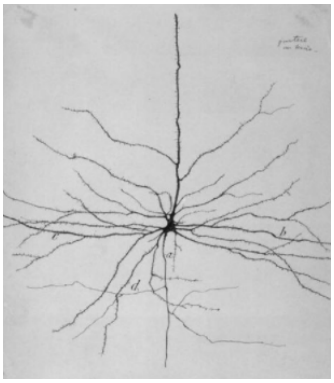


Figure 10 - Neurone ordinaire - Trouvée dans le livre Introduction to Deep Learning (Charniak, 2018)

Tout comme neurone humain, le neurone artificiel réalise un calcul simple et est connecté à d'autres neurones pour former un réseau. (Kiwi, et al., 2018).

2.2.3. Perceptron

Le perceptron est l'élément fondamental d'un réseau de neurones. Comme le démontre la figure 11, il est constitué d'un biais, de données d'entrée (inputs), de poids (weight), d'un module de somme, d'une fonction d'activation et d'une donnée de sortie (output). (Amini, 2021a).

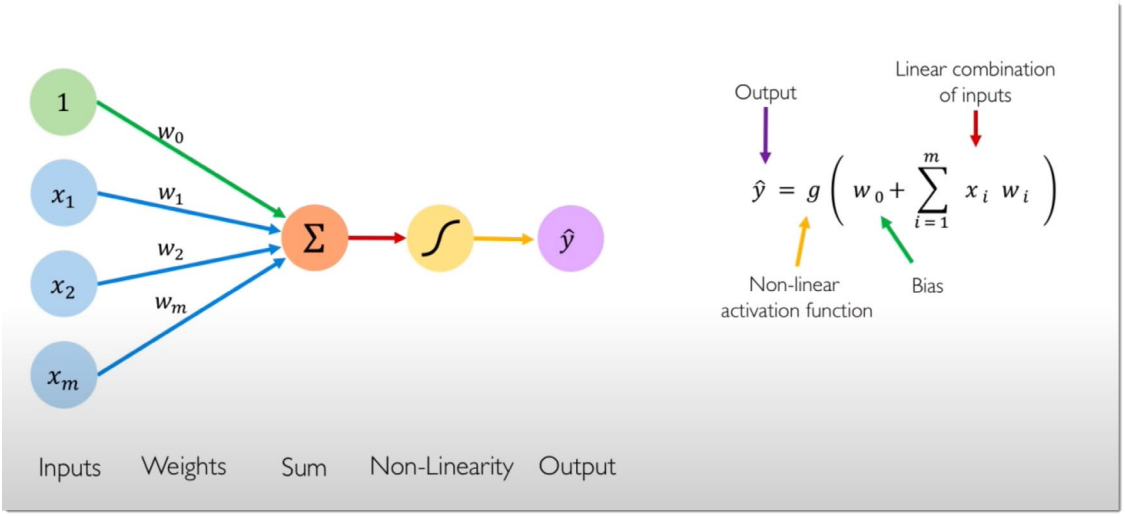


Figure 11 - Exemple d'un perceptron (Amini, 2021a)

Les données d'entrée sont les données d'un dataset qui sont conférées au réseau de neurones. Le poids d'une donnée correspond à l'importance de cette donnée par rapport aux autres pour la conclusion de la prédiction. Le biais permet de balancer la fonction d'activation vers le haut ou vers le bas sans s'attarder sur les données d'entrée. (Amini, 2021a).

La fonction somme représente simplement l'addition de toutes les données d'entrée altérées par chacun de leur poids. Une fonction de non-linéarité, par exemple une fonction sigmoïde, transforme un nombre réel en un nombre entre 0 et 1, utile pour toutes les fonctions de probabilité. Finalement, une donnée de sortie est simplement le résultat obtenu à la fin de ce processus. (Zhou, 2019).

2.2.4. Réseau de Neurones Multicouches

A partir d'un perceptron, il est possible de créer un réseau simple de neurones avec une seule ou plusieurs couches cachées (figure 12 et 13). Les couches cachées (hidden layout) sont toutes les couches qui ne sont pas des données d'entrée ou de sortie, c'est-à-dire toutes les couche de neurones qui possèdent à la fois une ou plusieurs entrées et une ou plusieurs sorties. (Amini, 2021a).

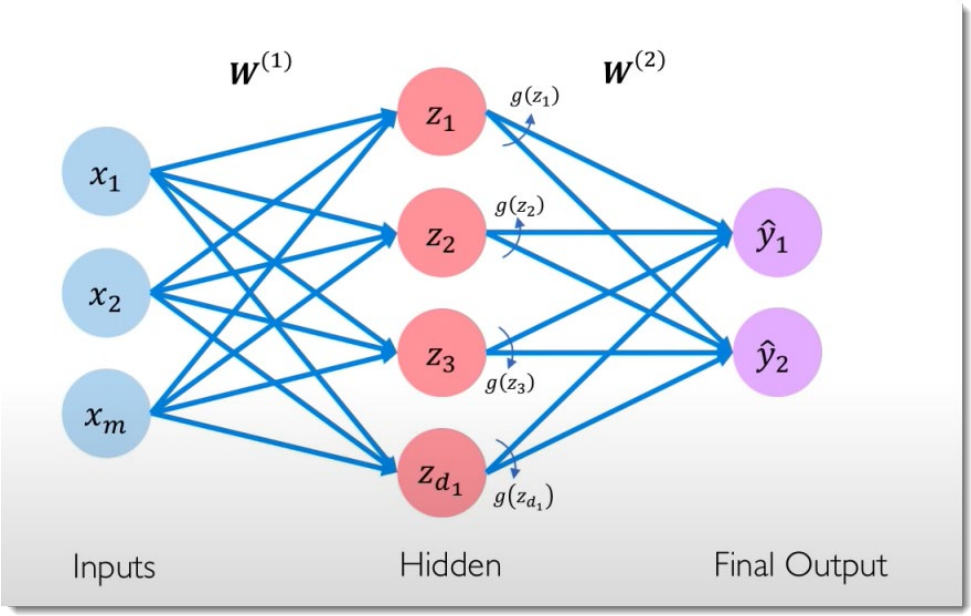


Figure 12 - Réseau simple de neurones avec une seule couche cachée (en rouge) (Amini, 2021a)

La caractéristique première des réseaux est leur habilité à apprendre à partir d'exemples. Le nombre de couches, connectées dans une structure, dépend de la complexité du problème à résoudre et va donc impacter le temps d'entraînement. (Miller, Blott, & Hames, 1992).

L'avantage d'un réseau de neurones multicouches est donc sa simplicité qui vise un gain de temps significatif pour résoudre des problèmes simples. Il pourrait démontrer une grande efficacité sur un dataset naïf sur lequel on ne connaît pas la valeur des données.

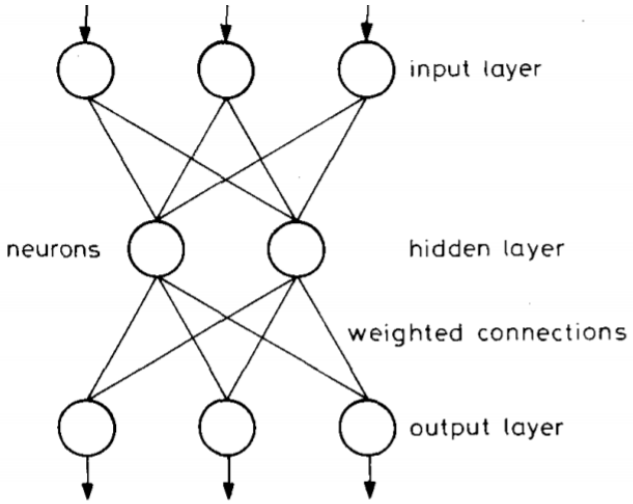


Figure 13 - Réseau de neurones à trois couches, Review of neural network applications in medical imaging and signal processing (Miller, Blott, & Hames, 1992)

2.2.5. Convolutional Neural Network 1D

Le Convolutional Neural Network 1D [CNN 1D] est un système d'apprentissage supervisé (voir figure 3), spécialisé dans les données à une dimension, comme les graphes temporels ou les signaux. (Amini, 2021b).

Par sa définition, une convolution est une opération où deux fonctions sont mises dans un rapport suggérant une sorte d'enroulement de l'une sur l'autre. (La Langue Française, s.d.).

En effet, contrairement aux Réseaux Neuronaux Artificiels [ANN] traditionnels, le CNN 1D a la capacité unique de fusionner l'extraction de caractéristiques à la classification, formant un seul et unique modèle d'apprentissage. De plus, les études récentes ont démontré qu'avec une approche appropriée, les réseaux compacts 1D peuvent atteindre des performances de pointe avec une complexité de calcul minimale. En pratique, le CNN 1D s'est distingué dans des domaines tels que la classification de l'électrocardiogramme (ECG) d'un patient, la surveillance de l'état de santé, la détection d'anomalies dans les circuits électroniques et la détection de pannes de moteur. (Kiranyaz, Ince, Abdeljaber, Avci, & Gabbouj, 2019).

Voici son fonctionnement selon une application simplifiée qui est relevée par la figure 14 :

1) Application d'un ensemble de poids, soit un filtre pour faire une extraction locale de caractéristiques.

2) Utilisation d'une multitude de filtres pour extraire différentes caractéristiques sur l'ensemble des données.

3) Partage des paramètres spatialement entre chaque filtre.

	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8	col 9
lin 1	5	2	9	8	1	6	4	7	4
lin 2	6	8	4	8	7	9	3	9	8

Figure 14 - Exemple de CNN 1D avec une table de 9x2, où chaque couleur représente un filtre différent - Image de l'auteur

Finalement, chaque étape générera une table plus floue avec les résultats de chaque filtre qui sera activé, par exemple, avec une fonction « Max Pooling » qui prendra la valeur maximale de chaque filtre (figure 15).

	col 1	col 2	col 3	col 4
lin 1	9	9	6	7
lin 2	8	8	9	9

Figure 15 - Exemple de CNN 1D avec une table de 4x2, où chaque couleur représente un filtre différent - Image de l'auteur

2.2.6. Convolutional Neural Network 2D

Le Convolutional Neural Network 2D [CNN 2D] est un système d'apprentissage supervisé (voir figure 3). Il reprend les concepts du CNN 1D, mais pour de la 2D. C'est donc un modèle spécialisé pour la reconnaissance d'image.

Comme le cite Alexander Amini (2021b), le CNN 2D est l'outil qui permet à la machine de savoir ce qu'elle regarde, de lui donner la vision. Ces modèles peuvent aussi être utilisés dans la segmentation, la capture d'image, la sécurité, la médecine et dans la robotique (2021b). La figure 16 démontre le fonctionnement d'un CNN 2D.

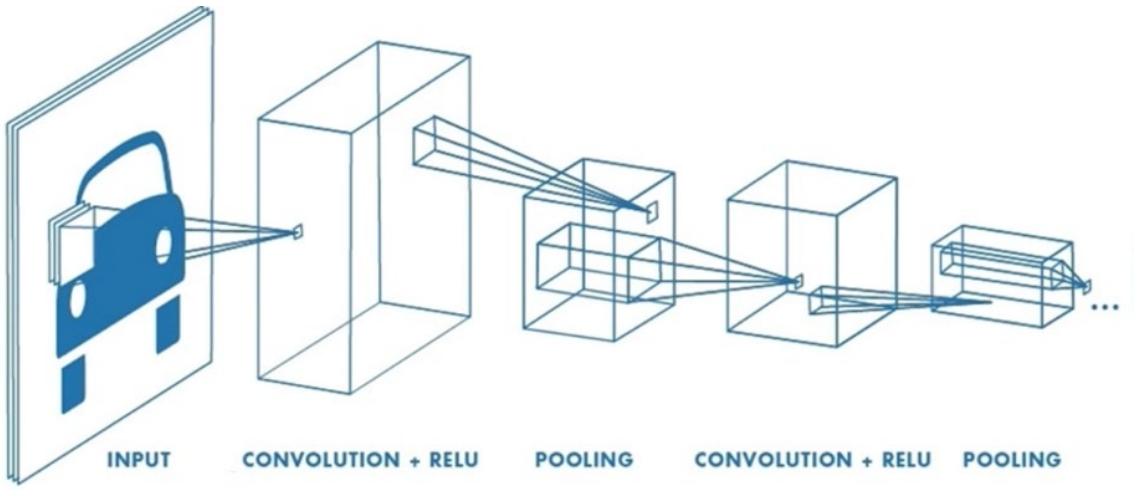


Figure 16 - Exemple d'un CNN 2D (Amini, 2021b)

Des recherches de CNN 2D ont été menées avec des cas d'analyse de signaux. Comme le démontre le document « A comparison of neural network models for wheeze detection », de Forkheim K.E., Scuse D. et Pasterkamp H. (1995), une conversion de signaux vers une image peut être entreprise pour utiliser les capacités d'un CNN 2D (figure 17).

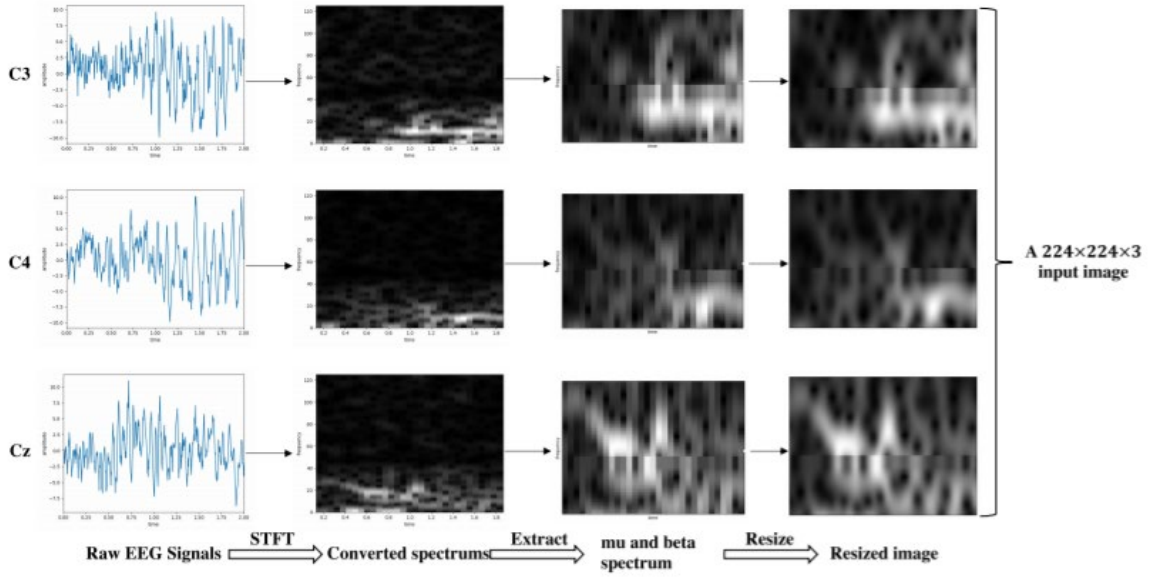


Figure 17 - Détails du processus de conversion d'un signal d'un électroencéphalogramme (Forkheim, Scuse, & Pasterkamp, 1995)

La suite de la recherche démontre que dans leur cas, le CNN 2D atteint des résultats plus précis que le ANN ou le SVM (figure 18).

Subject	Dataset	Average accuracy(%)			
		Proposed framework	CNN	ANN	SVM
1	A	73.7	71.2	66.9	68.7
	B	72.6	71.2	65.2	68.4
2	A	59.2	58.7	45.6	52.2
	B	60.3	57.7	45.1	54.2
3	A	65.1	61.5	54.1	59.2
	B	66.9	64	54.2	58.7
4	A	92.8	90.8	86.1	90.3
	B	91.2	90.2	87.2	89.4
5	A	81.3	79.6	76.2	73.2
	B	80.6	78.3	72.2	77.5
6	A	70.8	69.2	62.8	68.8
	B	70.6	70.2	59.7	68.5
7	A	73.3	69.2	66	68.2
	B	73.2	70.7	66.1	67.8
8	A	78.1	71.9	62.1	70.2
	B	77.7	69.9	62.2	68.9
9	A	76.4	72.6	60.1	65.1
	B	71.2	68.5	59.2	62.6
Average	A&B	74.2	71.4	63.9	68.4

Figure 18 - Moyenne des résultats sur la précision du Framework proposé (Forkheim, Scuse, & Pasterkamp, 1995)

Toutefois, contrairement au CNN 1D, le CNN 2D exige une transformation des données en images ou autres formats, ce qui impose une certaine perte de données ou de qualité. (Kiranyaz, Ince, Abdeljaber, Avci, & Gabbouj, 2019).

2.2.7. Recurrent Neural Network

Le Recurrent Neural Network [RNN] est un système d'apprentissage supervisé (voir figure 3) qui concerne les données temporelles et séquentielles. L'idée clé de ce modèle est de garder un état à chaque stade temporel afin de conserver une relation entre chaque réseau de neurones qui seront amenés à influencer les futurs réseaux.

Ce modèle peut être utilisé pour la traduction, la prédiction de crises financières, les études génétiques et bien d'autres domaines. (Soleimany, 2021a).

Comme le démontre la figure 19, le marqueur temps va influencer les prochains nœuds. Par exemple, si les données sont « lever en retard », « renverser son café », « embouteillage », le sentiment en sortie sera « énervé ». Cependant, si la prochaine donnée d'entrée est « gagner à la loterie », le sentiment en sortie changera.

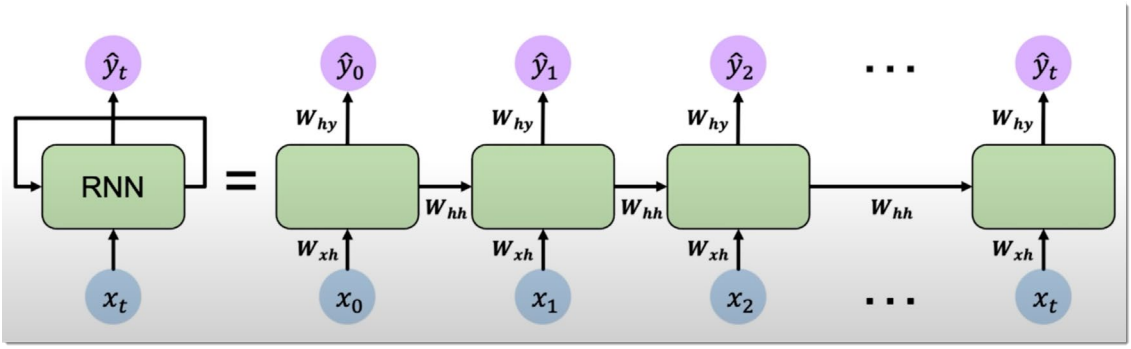


Figure 19 – Exemple d'un RNN (Soleimany, 2021a)

Les capacités du RNN sont donc pertinentes dans un projet basé sur des signaux temporels.

2.2.8. Generative Adversarial Network

Le Generative Adversarial Network [GAN] est un système d'apprentissage non-supervisé (voir figure 3). Le but principal de ce réseau est de classer des données sans avoir de label qui classe les valeurs d'un dataset et donc, d'apprendre des structures cachées des données. Ce modèle va prendre les données d'entrée qui proviennent d'une certaine distribution et générer un modèle qui représente cette distribution.

Le procédé fonctionne de la manière suivante : si nous prenons une image de base, le modèle va l'altérer en la passant dans ses différentes méthodes d'optimisation, de convolution, … comme pour un modèle classique. Puis il va reproduire le chemin inverse à partir du résultat de sortie pour reproduire l'image originale (figure 20). La différence entre l'image originale et la reconstruction

imparfaite de cette image servira lors du processus d'entraînement. En effet, lors de l'entraînement, le modèle comparera ces images et tentera de minimiser leur différence. (Soleimany, 2021b).

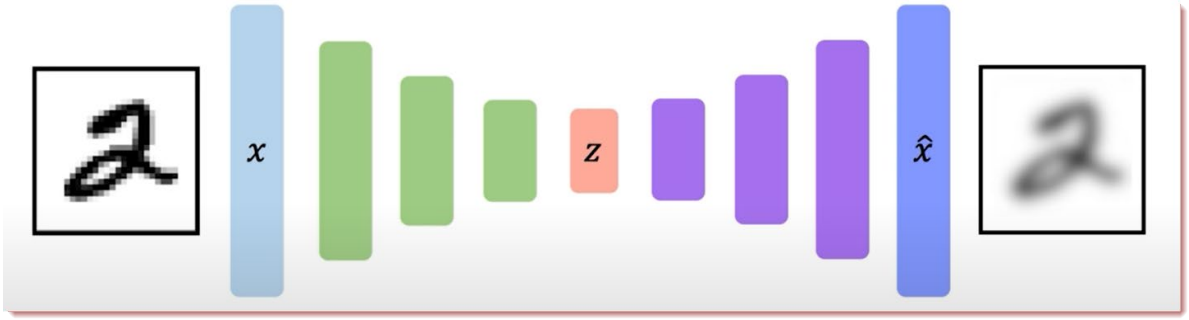


Figure 20 - Exemple d'un auto-encodeur (Soleimany, 2021b)

2.2.9. Reinforcement Learning

Le Reinforcement Learning [RL] est un système d'apprentissage sans donnée, ni label (voir figure 3). Comme le démontre la figure 21, ce modèle se base sur les notions d'agents qui interagissent avec un certain environnement. Ces agents vont donc produire des actions qui recevront toujours une récompense, un reward, qui induit une mesure de réussite ou d'échec de l'action. Plus cette récompense est grande, plus l'action produite par l'agent dans l'environnement est conséquente. (Amini, 2021d).

Figure 21 - Résumé de l'explication du Reinforcement Learning (Amini, 2021d)

L'avantage de ce modèle se résume au fait qu'il puisse générer lui-même des règles par rapport à cette gestion de récompense, ce qui permet au modèle de gérer des concepts sans donnée ni label. Ce genre de modèle est pertinent si nous avons comme objectif d'apprendre quelque chose sans en connaître les règles. (Amini, 2021d).

La figure 22 présente une voiture qui apprend à conduire tout droit sans aucune règle ni donnée. Le véhicule fera ses expérimentations et le modèle corrigera ses erreurs à travers le temps, en changeant simplement les valeurs des récompenses.

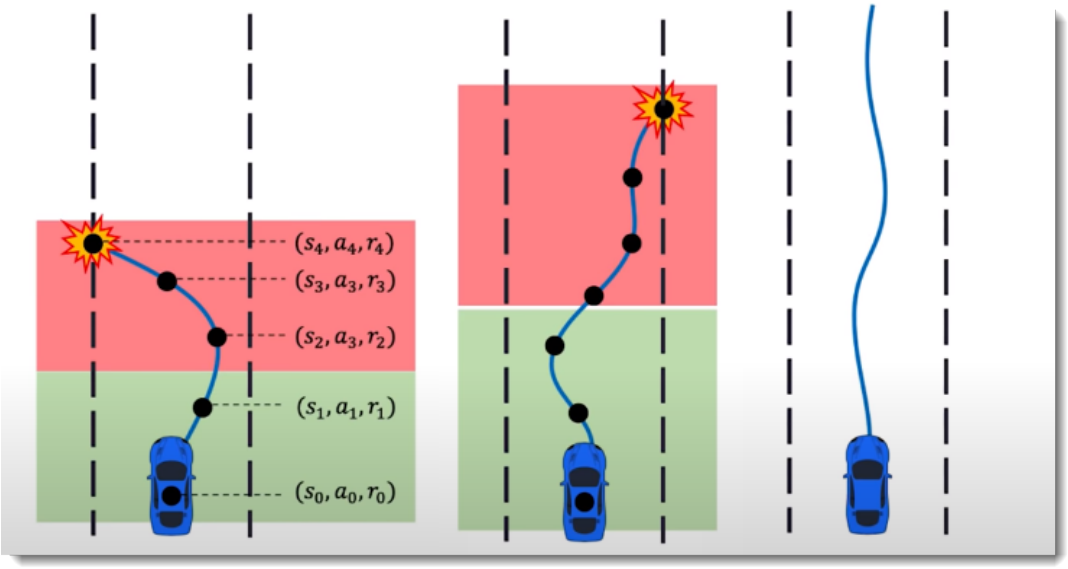


Figure 22 – Exemple de Reinforcement Learning avec une voiture autonome (Amini, 2021d)

2.3. Affinement d'un dataset

Le traitement des données est une pierre angulaire du machine learning. Les données doivent impérativement être optimisées et affinées pour être le plus pertinent possible, sans bruit ni nuisance. Le résultat de nos modèles seront fortement impactés par la gestion des données faite en amont de la modélisation. Dans ce chapitre, nous allons parcourir les différents outils pouvant se révéler pertinent pour notre problématique.

2.3.1. Backward Feature Elimination

2.3.1.1. Description de la Backward Feature Elimination

La Backward Feature Elimination est une approche itérative visant à limiter le nombre de données dans un dataset. Dans certaines situations, limiter les colonnes d'un dataset trop volumineux peut augmenter la qualité de la prédiction, tout en limitant le temps d'entraînement des modèles utilisés. (KNIME, 2021).

2.3.1.2. Fonctionnement de la Backward Feature Elimination

La Backward Feature Elimination se répétera sur chaque caractéristique (colonne) d'un dataset. A chaque itération, la caractéristique qui a le moins d'impact sur les performances du modèle est supprimée. L'algorithme va tester toutes les combinaisons possibles et va ensuite lister chaque résultat selon le nombre de suppressions et leur impact sur le modèle. La figure 23 démontre un exemple de combinaisons depuis l'outil KNIME.

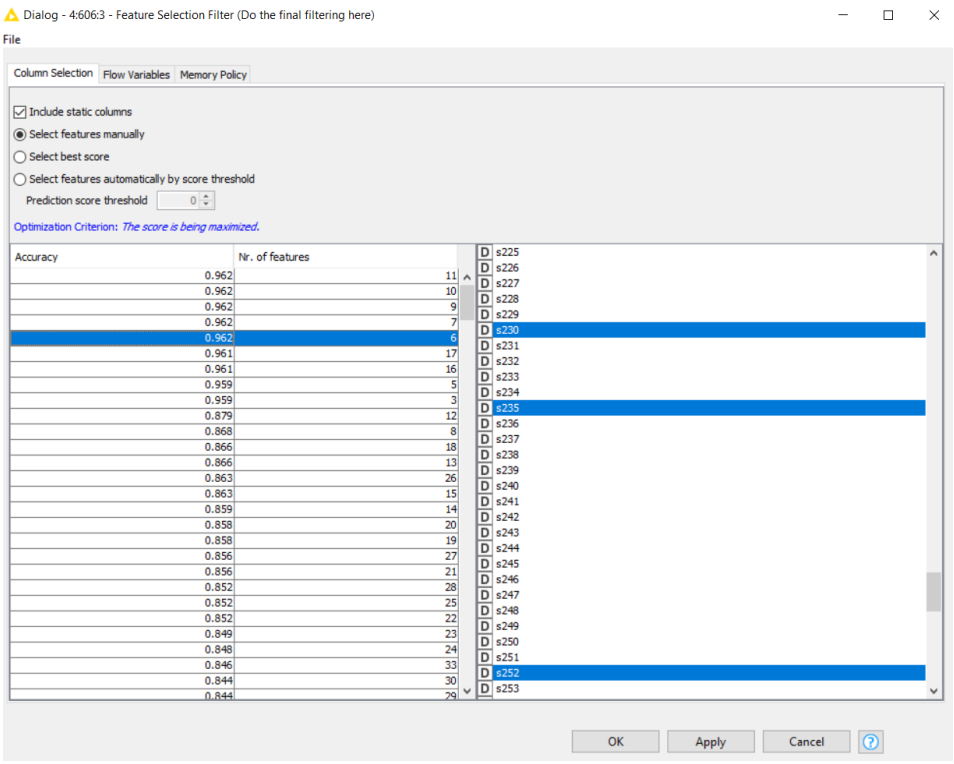


Figure 23 - Exemple Backward Feature Elimination depuis KNIME - Image de l'auteur

2.3.2. Linear Correlation

2.3.2.1. Description de la Linear Correlation

La Linear Correlation permet de placer une valeur sur la relation entre les colonnes (caractéristiques) d'un dataset. Ce rapport réciproque pourra induire si, par exemple, la notion temporelle impacte ou non la prédiction. Cet algorithme est spécialement utile pour vérifier l'état d'un dataset après l'utilisation d'un Backward Feature Elimination.

2.3.2.2. Fonctionnement de la Linear Correlation

La Linear Correlation va calculer un coefficient pour chaque paire de colonnes d'un dataset. Par exemple, cet algorithme va mesurer la corrélation de deux variables.

Le calcul de la corrélation va changer selon deux types de variables : soit des variables de type numérique vers numérique, où -1 est une corrélation négative importante et 1 est une corrélation positive importante, soit des variables de type nominal vers nominal, où 0 représente une corrélation nulle et 1 est une corrélation importante. (KNIME, 2020).

La matrice de corrélation, dévoilée dans la figure 24, est la métrique de sortie de la Linear Correlation qui permettra d'analyser le lien des colonnes d'un dataset.

Correlation matrix - 4:609 - Linear Correlation

File Edit Hilite Navigation View

Table "Correlation values" - Rows: 6 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	D s230	D s235	D s252	D s270	D s273	D Covariate5
s230	1.0	0.9953346247407...	0.9950578146995...	0.9938479709122...	0.9947016796904...	-0.479762561767...
s235	0.99533462...	1.0	0.993628338270399	0.9924439071370...	0.993442712816693	-0.478353376089...
s252	0.99505781...	0.993628338270399	1.0	0.9937114778913...	0.9942095382047...	-0.477225209641...
s270	0.99384797...	0.9924439071370...	0.9937114778913...	1.0	0.9946714704188...	-0.477315452290...
s273	0.99470167...	0.993442712816693	0.9942095382047...	0.9946714704188...	1.0	-0.475175085578...
Covariate5	-0.4797625...	-0.4783533760896...	-0.477225209641915	-0.4773154522902...	-0.4751750855789...	1.0

Figure 24 - Exemple de matrice de corrélation depuis KNIME - Image de l'auteur

2.4. Optimisation et évaluation d'un modèle de machine learning

De multiples outils existent pour optimiser des modèles de machine learning. Les modèles dévoileront leurs pleins potentiels seulement si ces outils sont paramétrés correctement.

Ces notions importantes, dont les principales sont résumées ci-dessous, seront nécessaires pour comprendre le fonctionnement et les résultats de nos modèles.

2.4.1. Fonctions d'activation

Une fonction d'activation est une fonction mathématique qui est appelée en sortie d'un neurone dans le but de réduire la perte du modèle.

2.4.1.1. Rectified Linear Unit

Le Rectified Linear Unit [ReLu] est l'une des fonctions d'activation qui permettent d'intégrer de la non-linéarité. (E. Dahl, N. Sainath, & E. Hinton, 2013).

La fonction d'activation ReLu va simplement calculer le nombre le plus grand entre 0 et la valeur d'entrée, ce qui va transformer toutes les valeurs d'entrée négatives en 0 et garder toutes les valeurs positives. Les courbes seront finalement altérées par les poids et les biais imposés par le modèle de réseau de neurones. La figure 25 expose un exemple de fonction ReLu.

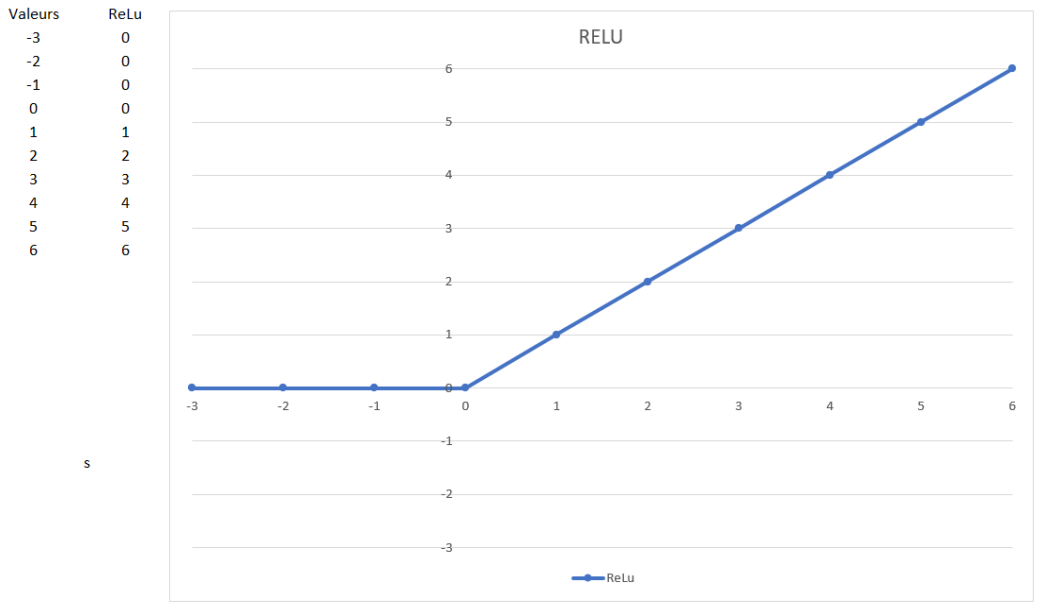


Figure 25 - Exemple de fonction ReLu avec des données d'entrée de « -3 » à « +6 » - Image de l'auteur

Le premier avantage d'une fonction d'activation ReLu est sa simplicité de calcul. Le second réside dans le fait qu'elle réduit la probabilité que le gradient disparaisse. En d'autres termes, elle empêche qu'un gradient infiniment petit n'arrive plus à altérer la valeur des poids d'un réseau. (Sharma, 2017).

L'inconvénient d'une activation ReLu réside dans sa logique même. En effet, si trop d'activations se retrouvent à 0, alors la plupart des neurones du réseau auront une sortie de valeur 0 et seront donc inactifs. De plus, la sortie de l'activation peut aller de 0 à l'infini, ce qui pourrait grandement impacter la puissance des calculs en cas de sortie trop élevée. (Himanshu, 2019).

2.4.1.2. Sigmoïde

La sigmoïde est une fonction d'activation non-linéaire dont la sortie est toujours comprise entre 0 et 1, peu importe la valeur d'entrée. Cette fonction d'activation est généralement utilisée pour calculer les mises à jour des poids dans l'apprentissage d'un modèle de réseau de neurones. (Farhadi, 2007).

La figure 26 expose une fonction sigmoïde.

Valeurs	ReLu	Sigmoïde	y
-3	0	0,04743	-3
-2	0	0,11920	-2
-1	0	0,26894	-1
0	0	0,50000	0
1	1	0,73106	1
2	2	0,88080	2
3	3	0,95257	3
4	4	0,98201	4
5	5	0,99331	5
6	6	0,99753	6

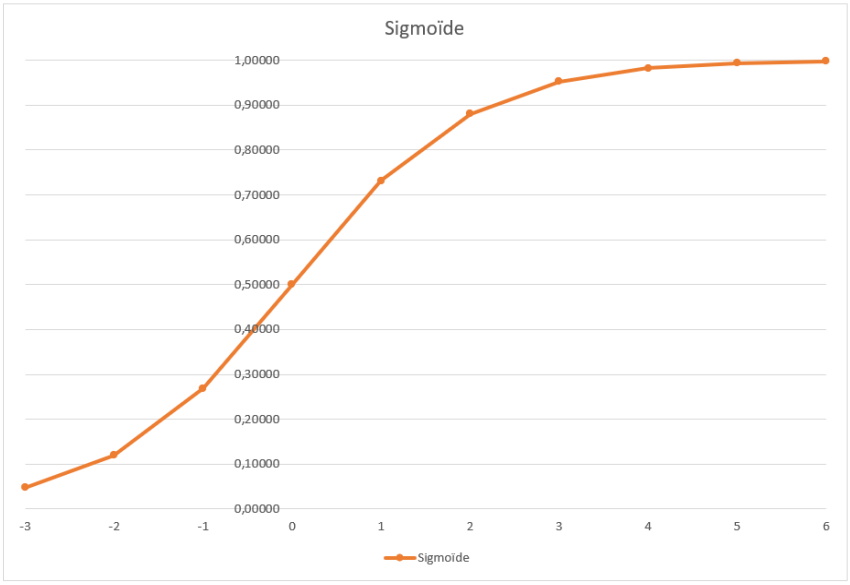


Figure 26 - Exemple d'une sigmoïde avec des données d'entrée de « -3 » à « +6 » - Image de l'auteur

L'avantage de la sigmoïde se résume par sa capacité à limiter les sorties entre 0 et 1. En effet, elle permet ainsi de contrôler la grandeur du chiffre de sortie. (Sharma, 2017).

Au contraire de l'activation ReLu, l'inconvénient de la sigmoïde réside dans le fait qu'elle pousse le gradient vers des valeurs de plus en plus faibles et un gradient infiniment petit ne peut plus altérer la valeur des poids des neurones. De plus, la vitesse d'un modèle basé sur une sigmoïde est plus lente qu'avec une activation ReLu. (Himanshu, 2019).

Finalement, la figure 27 démontre la différence des deux fonctions d'activation.

Valeurs	ReLu	Sigmoïde	y
-3	0	0,04743	-3
-2	0	0,11920	-2
-1	0	0,26894	-1
0	0	0,50000	0
1	1	0,73106	1
2	2	0,88080	2
3	3	0,95257	3
4	4	0,98201	4
5	5	0,99331	5
6	6	0,99753	6

Figure 27 - Différence entre une fonction sigmoïde (en orange) et ReLu (en bleu) - Image de l'auteur

2.4.2. Bruit

Le bruit représente l'ensemble des données de prédiction qui sont à la fois dans un espace de signal bruyant et dans un espace de signal sans bruit (Tamura, 1989). Ces données difficiles à prédire réduisent les performances du modèle de prédiction.

Le bruit peut être généré ou accentué par des nuisances externes lors de la prise de données. Dans notre cas, la pression artérielle, les clignements de l'œil ou encore la respiration peuvent générer du bruit qui brouille l'apprentissage du modèle. En effet, le bruit peut dissimuler des patterns ou des suites logiques dans les données, en les altérant de manière significative ou non.

Une analyse du bruit sur un modèle est donc importante pour vérifier la pertinence ainsi que la qualité des données, mais aussi pour améliorer la généralisation du modèle.

2.4.2.1. Gaussian Noise

Le Gaussian Noise [GS] est une couche mise à disposition par Keras lors de l'élaboration d'un modèle de prédiction. Il permet d'atténuer l'overfitting en injectant des données aléatoires. Le Gaussian Noise est un processus de corruption pour les entrées à valeur réelle. (TensorFlow, 2021).

La figure 28 est un exemple d'application d'un Gauddian Noise.

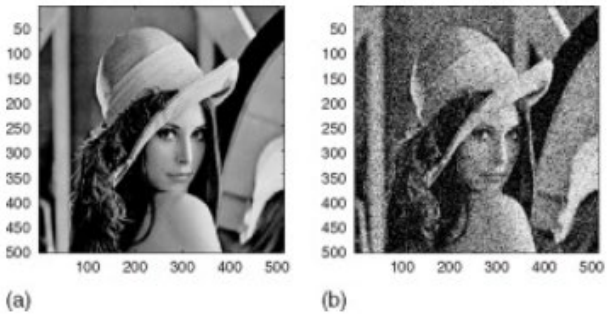


Figure 28 - (a) Image originale, (b) image corrompue par un bruit blanc - Trouvée sur sciencedirect[7]

2.4.3. Fonctions de perte et d'optimisation

Les fonctions de perte et d'optimisation sont des algorithmes appelés à la configuration du modèle de réseau de neurones qui vont permettre selon les attentes, le contexte ou encore le dataset d'augmenter la précision d'un modèle, tout en diminuant sa perte. (Amini, 2021a).

Les algorithmes de perte proposés par Keras pour de la classification binaire sont principalement la Binary Crossentropy et la Mean Square Error. (Keras, 2021a).

La Binary Crossentropy va permettre de rendre une réponse binaire. Si, par exemple, les données d'entrée sont « pluvieux » et « humide », la réponse à « est-ce qu'il pleut ? » sera « oui ». (Voir annexe IX, Formule de la Binary Crossentropy).

La Mean Square Error [MSE] va permettre de rendre une réponse concrète à des données d'entrée. Si, par exemple, les données d'entrée sont « 100 m^2 », « 2 lits » et « salle de bain », la réponse au prix sera « 160.- ». (Voir annexe IX, Formule de la Mean Square Error).

Les principales fonctions d'optimisation proposées par Keras sont Stochastic Gradient Descent [SGD], Adam, Adadelta, Adagrad, RMSProp, Nadam, Follow The Regularized Leader [FTRL] et Adamax. Elles permettent, comme la SGD par exemple, de jouer avec la valeur des poids et des données d'entrée pour atteindre de meilleurs résultats.

Le SGD est une fonction d'optimisation proposée par Keras. Cette fonction met à jour les poids et le gradient lorsque la sortie est à 0. (Keras, 2021e).

Adam est une fonction d'optimisation mis à disposition par Keras. La méthode est efficace sur le plan informatique étant peu gourmande en mémoire. De plus, elle est invariante aux changements des gradients et adaptée aux problèmes qui sont volumineux en termes de paramètres. (Ba & Kingma, 2014).

[7] Additive White Gaussian Noise https://www.sciencedirect.com/topics/computer-science/additive-white-gaussian-noise

Comme le cite la documentation de Keras, Adagrad est une fonction d'optimisation. Son taux d'apprentissage est spécifique aux paramètres qui sont adaptés en fonction de la fréquence à laquelle ces derniers sont mis à jour pendant l'apprentissage. Plus un paramètre sera mis à jour, plus ces mises à jour seront petites. (Keras, 2021f).

Adadelta est une fonction d'optimisation mis à disposition par Keras qui est basée sur un taux d'apprentissage adaptif par dimension. Elle permet de remédier aux deux inconvénients qui sont : la décroissance continue du taux d'apprentissage et le besoin de sélectionner manuellement d'un taux global d'apprentissage. (Keras, 2021g).

Le RMSProp est une fonction d'optimisation proposée par Keras, qui maintient une moyenne actualisée du carré du gradient, puis divise le gradient par la racine de cette moyenne. (Keras, 2021h).

Nadam est une fonction d'optimisation mis à disposition par Keras, qui implémente l'algorithme Nadam, qui utilise la logique d'Adam. Cette fonction a pour effet de limiter les fonctions bruyantes et d'améliorer la convergence. (Keras, 2021i).

Le Ftrl est un algorithme d'optimisation implémenté par Google pour la prédiction de clics. Cet algorithme est adapté aux modèles de réseaux de neurones peu profonds avec des espaces de caractéristiques larges et clairs. (Keras, 2021j).

Adamax est le dernier algorithme d'optimisation proposé par Keras. Il s'agit d'un variant d'Adam, qui est basé la norme de l'infini. Ses spécialités sont les modèle entrelacés. (Keras, 2021k).

2.5. Évaluation d'un modèle de machine learning

L'évaluation des modèles de machine learning est un élément primordial pour vérifier à la fois la qualité et la pertinence des résultats. Pour cette raison, une palette de métriques devront être analysées et implémentées afin de créer un filet de sécurité fiable pour les modèles.

Ces métriques, dont les principales sont résumées ci-dessous, seront nécessaires pour comprendre et valider les résultats de nos différents modèles.

2.5.1. Matrice de confusion

La matrice de confusion est un tableau à deux dimensions qui oppose les données prédites aux données réelles. Comme l'illustre la figure 29, elle résume efficacement les prédictions du modèle (positives contre négatives) et démontre rapidement ses erreurs. (Lutu, 2011).

Les True Positives [TP] correspondent aux résultats vrais prédits comme étant vrais.

Les False Positives [FP] définissent les résultats faux, prédits comme étant vrais.

Les True Negatives [TN] décrivent les résultats faux prédits comme étant faux.

Les False Negatives [FN] correspondent les résultats vrais, prédits comme étant faux.

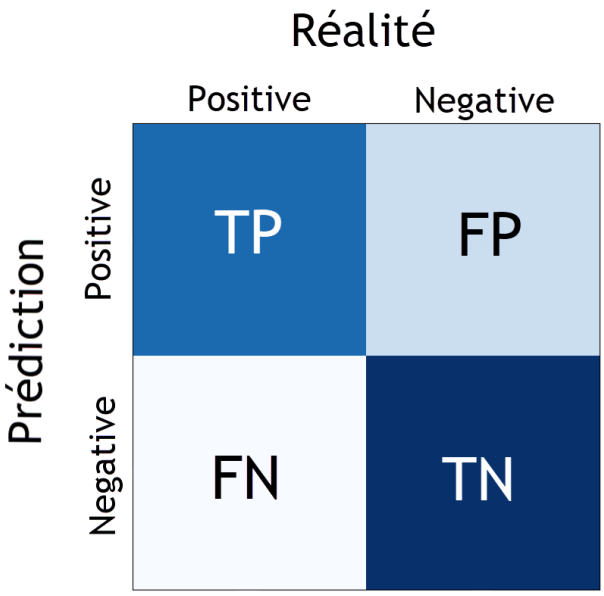


Figure 29 - Exemple de matrice de confusion - Image de l'auteur

2.5.2. Area Under the Curve

Comme l'illustre la figure 30, l'Area Under the Curve [AUC] est un score qui présente dans le temps la proportion de prédictions de True Positives (données vraies, prédites vraies) contre celles de False Positives (données fausses, prédites vraies). Un AUC de 0.5 est catastrophique. Ce score

indique que le modèle fait autant faux que juste, ce qui équivaut à prédire de manière aléatoire. Un AUC de 1 est au contraire parfait. (Kallner, 2014).

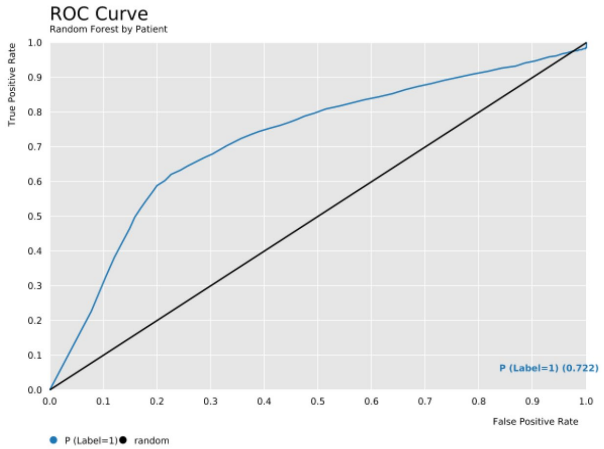


Figure 30 – Exemple d'un AUC (zone sous la ligne bleue) et d'une Roc Curve (ligne bleue) - Image de l'auteur

2.5.3. Perte

La perte, Loss en anglais, est calculé sur la base de l'entraînement et de la validation du modèle. Elle démontre comment le modèle se comporte après chaque itération d'optimisation. Plus la perte est basse, meilleure est le modèle (sauf en cas d'overfitting). La perte représente la somme de toutes les erreurs commises lors de l'entraînement et de la validation du modèle. (Keras, 2021a).

La perte est donc une métrique importante pour mesurer la qualité de l'entraînement et de l'évaluation. Idéalement, sa valeur est censée diminuer à chaque itération. La figure 31 expose le graphique de la perte d'un modèle entraîné 60 fois.

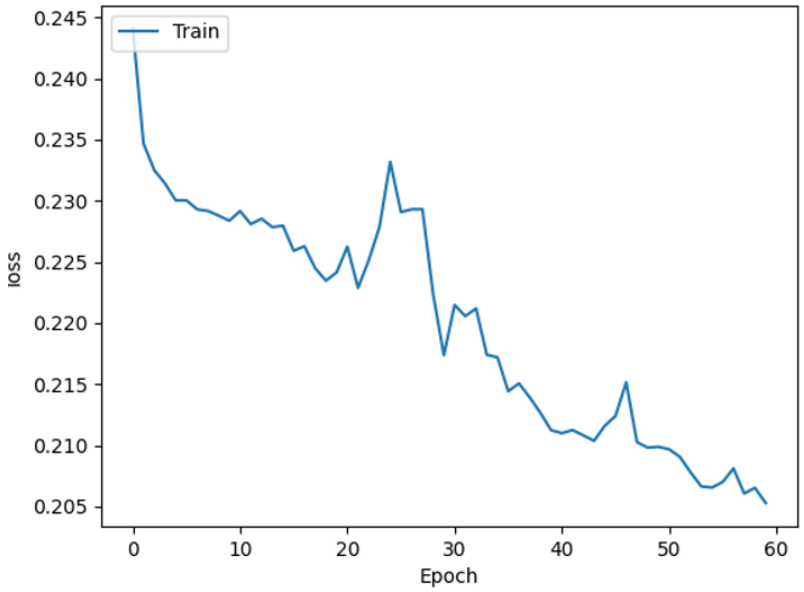


Figure 31 – Graphique de perte lors d'un entraînement de 60 itérations - Image de l'auteur

2.5.4. Accuracy

L'accuracy est une métrique qui mesure la performance de l'algorithme. Elle calcule la fréquence à laquelle les prédictions sont égales à la réalité. (Keras, 2021b).

Pour ce faire, on additionne toutes les prédictions correctes et on les divise par tous les échantillons. Le tableau 1 démontre comment l'accuracy est calculée.

Accuracy		$\frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{FP} + \mathbf{FN} + \mathbf{TN}}$
Total d'images		20
Images de « chien » prédites comme « chien »	*TP*	9
Images de « chat » prédites comme « chien »	*FP*	1
Images de « chat » prédites comme « chat »	*TN*	8
Images de « chien » prédites comme « chat »	*FN*	2
Valeur de l'accuracy		$\frac{9+8}{9+1+2+8} = 85\%$

Tableau 1 - Calcul de l'accuracy basé sur la classification de chiens et de chats

La figure 32 expose le graphique de l'accuracy d'un modèle entraîné 60 fois.

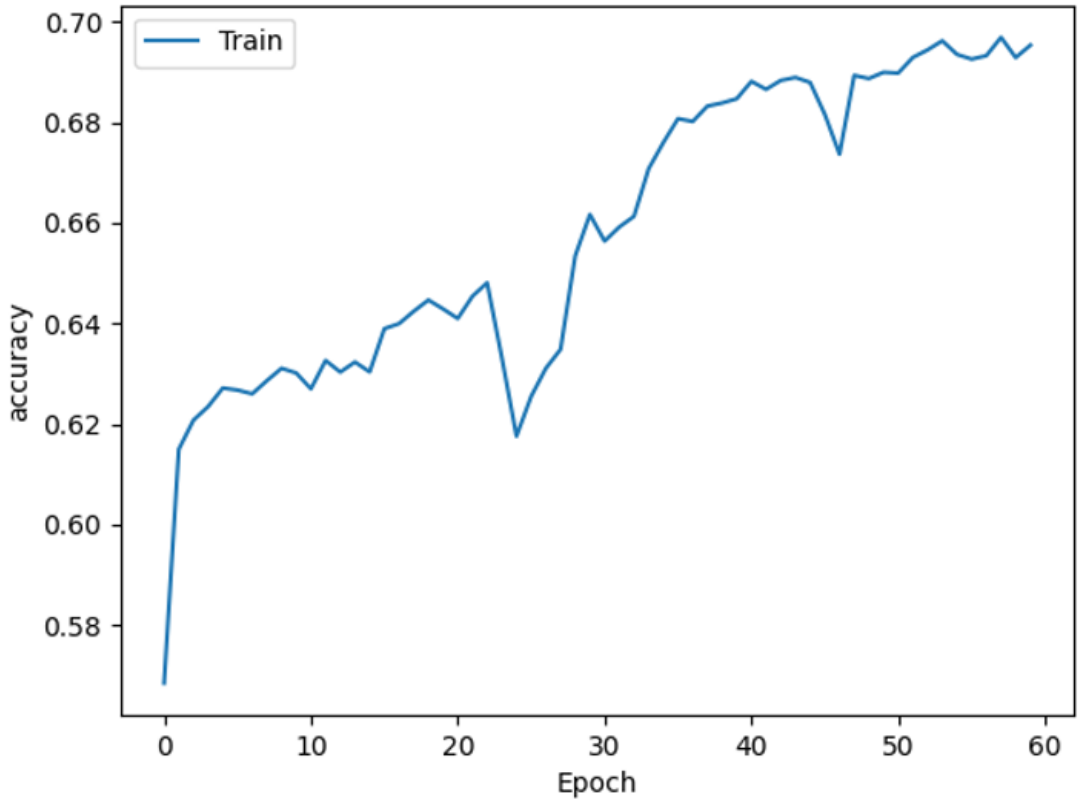


Figure 32 - Exemple d'un graphique basé sur l'accuracy - Image de l'auteur

2.5.5. Précision

La précision est une métrique utile pour mesurer la qualité d'un modèle de réseau de neurones. Elle se résume par la question suivante : sur toutes les prédictions positives, combien sont-elles correctes ? (Keras, 2021c).

Le tableau 2 démontre comment la précision est calculée (Treboux, Ingold, & Genoud, 2020).

Précision		$\frac{\mathbf{TP}}{\mathbf{TP + FP}}$
Total d'images		20
Images de « chien » prédites comme « chien »	*TP*	9
Images de « chat » prédites comme « chien »	*FP*	1
Images de « chat » prédites comme « chat »	*TN*	8
Images de « chien » prédites comme « chat »	*FN*	2
Valeur de la précision		$\frac{9}{9+1} = 90\%$

Tableau 2 - Calcul de la précision basé sur la classification de chiens et de chats

La figure 33 expose le graphique de la précision d'un modèle entraîné 60 fois.

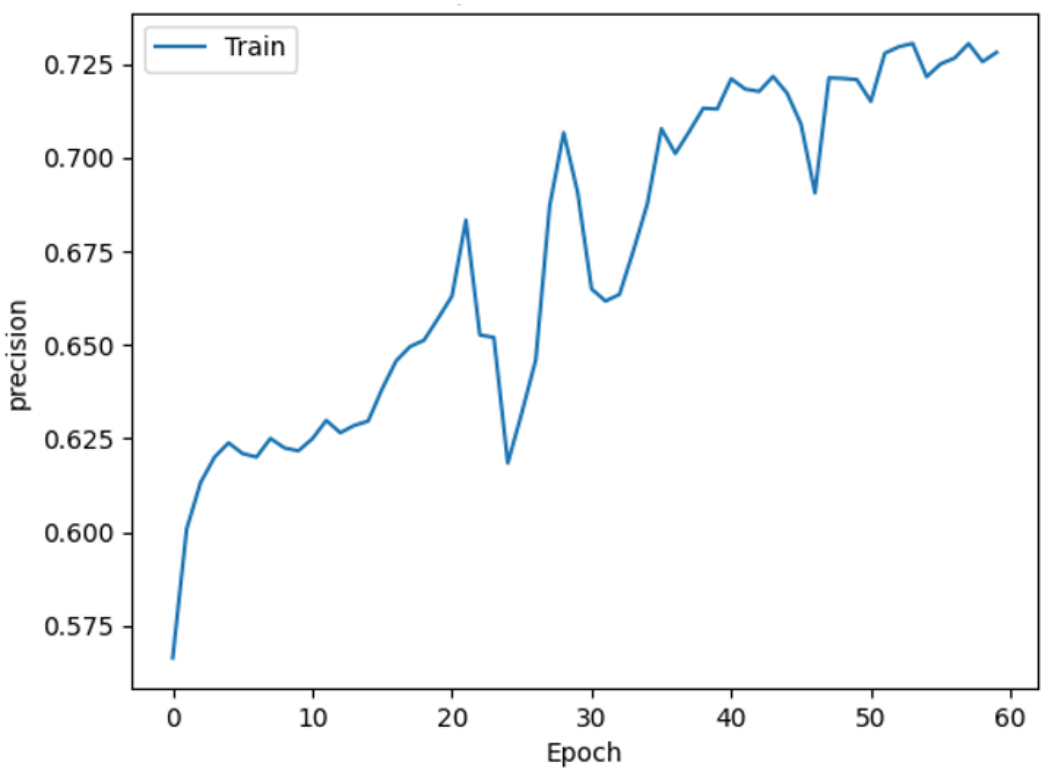


Figure 33 - Exemple de graphique basé sur la précision - Image de l'auteur

2.5.6. Recall

Le recall est une métrique permettant de mesurer la qualité d'un modèle de réseau de neurones. Il se résume par la question suivante : sur toutes les vérités prédites, combien sont-elles correctes ? (Keras, 2021c).

Le tableau 3 démontre comment le recall est calculé (Treboux, Ingold, & Genoud, 2020).

Recall		$\frac{\mathbf{TP}}{\mathbf{TP + FN}}$
Total d'images		20
Images de « chien » prédites comme « chien »	*TP*	9
Images de « chat » prédites comme « chien »	*FP*	1
Images de « chat » prédites comme « chat »	*TN*	8
Images de « chien » prédites comme « chat »	*FN*	2
Valeur du recall		$\frac{9}{9+2} = 81{,}82\%$

Tableau 3 - Calcul du recall basé sur la classification de chiens et de chats

La figure 34 expose le graphique du recall d'un modèle entraîné 200 fois.

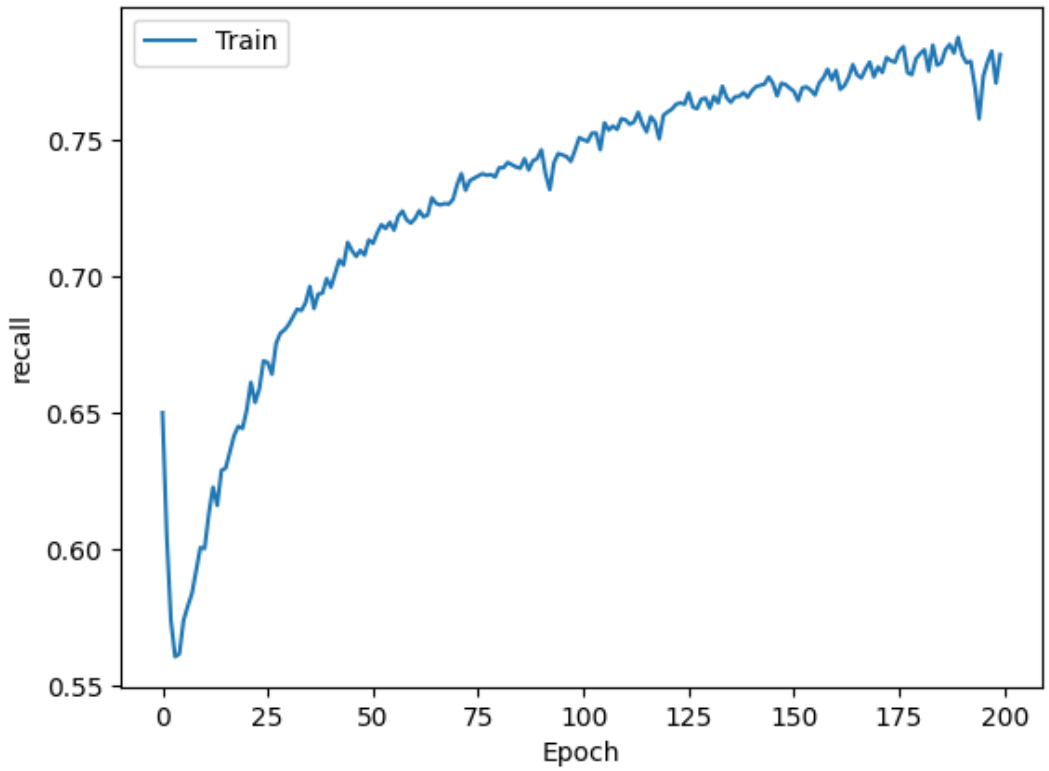


Figure 34 - Exemple de graphique basé sur le recall - Image de l'auteur

2.5.7. Standard Error

La Standard Error [SE] est une métrique statistique qui permet de définir un intervalle de confiance en une mesure donnée. Elle dépend du nombre d'échantillons et de la valeur des résultats. En d'autres termes, les intervalles de confiance permettent de comprendre dans quelle mesure il est possible de se fier aux estimations de notre échantillon. Ils fournissent la fourchette la plus probable si nous avions pu tester toutes les données. Sans cet intervalle de confiance, il serait impossible de comparer deux AUC (figure 35). (Sauro, 2014).

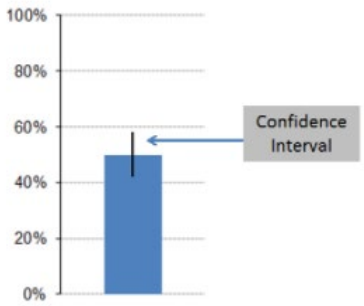


Figure 35 - Exemple d'intervalle de confiance - Trouvée sur measuringu[8]

2.5.8. Underfitting et overfitting

Idéalement, en machine learning, le but est de bâtir un modèle qui prédit parfaitement les données de test et non les données d'entraînement. L'objectif est donc de créer un modèle qui apprend des données d'entraînement tout en généralisant bien toute autre donnée. (Amini, 2021a).

Comme le montre la figure 36, si nous tirions une ligne dans un nuage de points, l'underfitting arriverait lorsqu'un modèle n'aurait pas la capacité d'apprendre des données fournies.

L'overfitting apparaîtrait lorsque le modèle est trop complexe avec trop de paramètres superflus. En effet, ce modèle ne généraliserait pas bien les nouvelles données fournies.

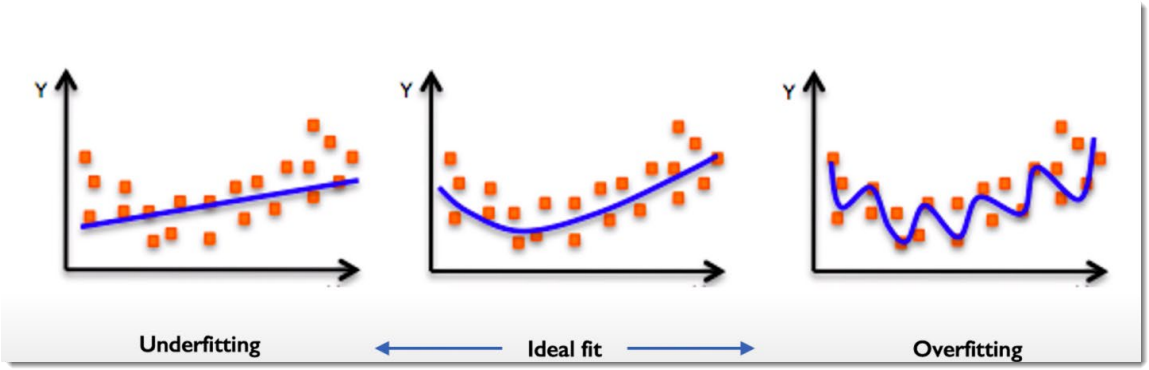


Figure 36 - Exemple d'underfitting (à gauche) et d'overfitting (à droite) (Amini, 2021a)

[8] How to Compute a Confidence Interval in 5 Easy Steps: https://measuringu.com/ci-five-steps/

3. Le Dataset

Le dataset fourni par Sensimed possède 150'353 lignes avec 4 colonnes informatives, 302 colonnes de données et 5 colonnes dites Covariates (données avec de grands impacts sur la prédiction) pour une taille de 868 MB (figure 37). Le fichier a subi une normalisation préalable de la part du client. Une analyse sur le dataset brut a été réalisée avec la conclusion que le dataset normalisé est plus pertinent pour la classification. Outre ces notions de base, dans ce chapitre, les principales caractéristiques de ce dataset seront analysées.

Output data - 0:729 - Column Resorter

File Edit Hilite Navigation View

Table "default" - Rows: 150353 Spec - Columns: 311 Properties Flow Variables

Row ID	ID	PatientId	BurstId	Label	Covaria...	Covaria...	Covaria...	Covaria...	Covaria...	s000	s001	s002	s003	s004	s005	s006	s007	s008	s009
0	0	0	3	b	a	d	b	-1.497	0.004	-0.004	-0.003	-0.009	-0.02	-0.023	-0.015	-0.011	-0.008	-0.007	-0.01
1	1	0	8	b	a	d	b	-1.497	0.004	-0.007	-0.043	-0.059	-0.064	0.137	-0.061	-0.079	-0.08	-0.075	-0.076
2	2	0	12	b	a	d	b	-1.497	0.004	-0.009	-0.013	-0.013	-0.012	-0.014	-0.016	-0.018	-0.021	-0.02	-0.019
3	3	0	15	b	a	d	b	-1.497	0.004	-0.054	-0.055	-0.054	-0.054	-0.055	-0.056	-0.057	-0.057	-0.056	-0.057
4	4	0	16	b	a	d	b	-1.497	0.004	-0.055	-0.056	-0.056	-0.057	-0.058	-0.059	-0.057	-0.057	-0.059	-0.06
5	5	0	18	b	a	d	b	-1.497	0.004	-0.069	-0.069	0.018	-0.002	-0.057	-0.062	-0.063	-0.064	-0.064	-0.066
6	6	0	19	b	a	d	b	-1.497	0.004	-0.06	-0.061	-0.062	-0.064	-0.061	-0.063	-0.063	-0.063	-0.064	-0.062
7	7	0	25	b	a	d	b	-1.497	0.004	-0.012	-0.009	-0.003	0.026	0	-0.018	-0.023	-0.018	-0.017	-0.014
8	8	0	28	b	a	d	b	-1.497	0.004	-0.059	-0.06	-0.062	-0.061	-0.063	-0.064	-0.065	-0.066	-0.068	-0.063
9	9	0	29	b	a	d	b	-1.497	0.004	-0.062	-0.077	-0.072	-0.072	-0.073	-0.075	-0.078	0.173	-0.048	-0.069
10	10	0	33	b	a	d	b	-1.497	0.004	-0.06	-0.059	-0.06	-0.061	-0.059	-0.06	-0.06	-0.061	-0.062	-0.063
11	11	0	34	b	a	d	b	-1.497	0.004	-0.052	-0.054	-0.055	-0.057	0.133	-0.036	-0.049	-0.053	-0.057	-0.057
12	12	0	35	b	a	d	b	-1.497	0.004	-0.062	-0.062	-0.061	-0.066	-0.065	-0.064	-0.065	-0.064	-0.064	-0.061
13	13	0	38	b	a	d	b	-1.497	0.004	0.121	-0.056	-0.068	0.069	0.047	-0.059	-0.069	-0.074	-0.075	-0.078
14	14	0	43	b	a	d	b	-1.497	0.004	-0.095	-0.095	-0.096	-0.094	-0.095	-0.097	-0.096	-0.097	-0.096	-0.096
15	15	0	44	b	a	d	b	-1.497	0.004	-0.083	-0.077	0.051	-0.069	-0.079	-0.083	-0.082	0.03	-0.071	-0.079
16	16	0	45	b	a	d	b	-1.497	0.004	-0.078	-0.078	-0.08	-0.083	-0.083	-0.082	-0.081	-0.081	-0.081	-0.082
17	17	0	47	b	a	d	b	-1.497	0.004	-0.077	-0.08	-0.08	-0.08	-0.082	-0.064	-0.063	-0.066	-0.069	-0.069
18	18	0	51	b	a	d	b	-1.497	0.004	-0.079	-0.079	-0.028	-0.039	-0.075	-0.079	-0.082	-0.081	-0.081	-0.083
19	19	0	60	b	a	d	b	-1.497	0.004	-0.056	-0.051	-0.052	-0.053	-0.054	-0.057	-0.06	-0.06	-0.059	-0.059
20	20	0	61	b	a	d	b	-1.497	0.004	-0.075	-0.075	-0.076	-0.078	-0.08	-0.079	-0.079	-0.078	-0.078	-0.077

Figure 37 - Affichage des 20 premières lignes du dataset depuis KNIME - Image de l'auteur

3.1. Burst

Le client a donné le nom de Burst à une série de valeurs sur la pression de l'œil qui sont récoltées par les lentilles toutes les 5 minutes pendant 30 secondes. La figure 38 est la représentation graphique du Burst numéro 0 du patient numéro 0 (malade). Ces valeurs vont de -0.067 à +0.289.

Figure 38 - Graphique du Burst numéro 0 depuis KNIME - Image de l'auteur

Le nombre de Burst par patient n'est pas rectiligne dans le dataset. Ce nombre fluctue de 1 à 283 Bursts par patient, avec une moyenne d'environ 150 Bursts par patient. La figure 39 résume l'ensemble des Bursts d'un patient.

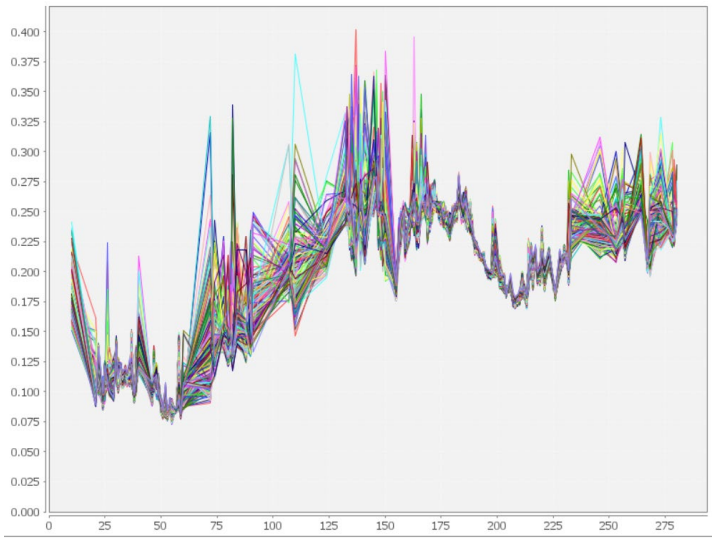


Figure 39 - Graphique de tous les Bursts du patient numéro 13 depuis KNIME - Image de l'auteur

3.2. Équilibre des données

Comme le démontre la figure 40, le nombre de personnes classées malades ou saines n'est pas équilibré. Il y a beaucoup plus de patients diagnostiqués malades que sains. Pour être précis, le dataset contient 85,4% de personnes malades (851 personnes) et 14,6% de personnes saines (145 personnes), ce qui correspond à 86,3% de Bursts malades (129'725 Bursts) et 13,7% de Bursts sains (20'628 Bursts).

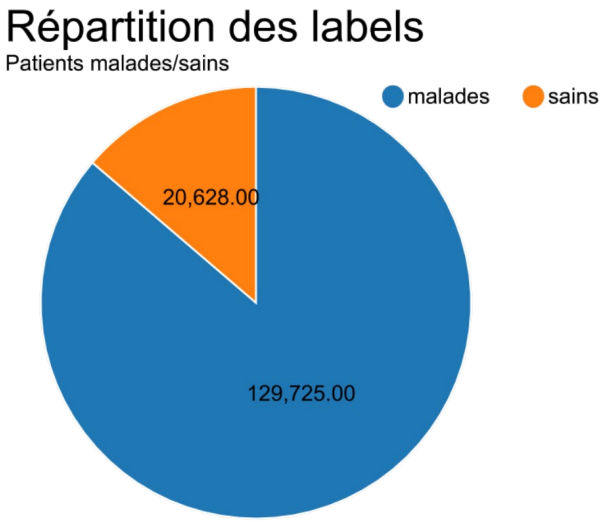


Figure 40 - Diagramme de la répartition des patients - Image de l'auteur

3.3. Séparation des données

Lors de ce projet, le dataset va être divisé en trois datasets tout en veillant à ne pas couper les Bursts d'un patient entre deux datasets différents. Le premier dataset (80,5% des données) sera dédié à l'entraînement des modèles de prédiction. Le deuxième dataset (14,53% des données) sera destiné au test des modèles et finalement, le troisième dataset (4,97% des données) concernera exclusivement la validation des modèles de classification sauvegardés.

4. Choix Technologique

Bon nombre de technologies sont disponibles pour effectuer de la classification de données. Dans le cadre de ce projet, nous allons débattre des qualités et des défauts d'applications d'analyse de données utilisant des interfaces graphiques, tout comme des différents type de machine learning et de réseaux de neurones. Le but est de trouver quelle technologie nous permettra d'atteindre le plus grand taux de classification de données, tout en relevant des patterns dans les Bursts pour expliquer certaines anomalies de la pression de l'œil lors d'un glaucome avéré.

4.1. Plateformes et outils

Nous allons tout d'abord présenter et analyser les différents outils, plateformes, langage, API ou framework qui ont été implémentés pour le machine learning, tout en restant dans le cadre et les prérequis de ce travail de Bachelor.

4.1.1. KNIME

KNIME est un logiciel libre et open-source d'analyse de données avec une interface graphique. Ce logiciel comprend une multitude d'outils pour la classification de données. Il possède le KNIME Hub, un outil qui documente toutes les fonctionnalités du produit, un grand plus pour les débutants. KNIME prône l'intuitivité de ses environnements, tout comme la facilité de son utilisation : un produit taillé pour la science des données. (KNIME, 2021).

KNIME a l'avantage de gérer facilement les datasets et d'exposer rapidement des premiers résultats grâce à ses nœuds internes déjà préconfigurés. Il faut néanmoins comprendre la logique de ces nœuds. De plus, mis à part les paramètres, nous n'avons aucun contrôle sur ses nœuds, qui pourraient se comparer à des boîtes noires. KNIME propose donc des solutions pour personnaliser des nœuds avec notre propre code.

Étant facile d'utilisation et possédant une documentation exemplaire, ce logiciel est un choix d'utilisation stratégique pour ce projet. De plus, KNIME est open source et permet de créer rapidement des modèles de classification qui seront taillés pour notre problématique. Finalement, l'utilisation de KNIME a été demandée pour la réalisation de ce travail.

4.1.2. Python

Comme le cite sa documentation officielle, python est un langage de programmation puissant et rapide. Ce langage s'intègre aisément avec les autres et s'exécute sur toutes les plateformes. Python est facile à apprendre et surtout ouvert à tous. (Python, 2021).

Dans notre cas, python nous permet de modéliser facilement des modèles complèxes qui seront affinés pour nos exigences et nos besoins. De plus, de grands outils, tels que TensorFlow ou Keras,

sont disponibles depuis ce langage pour bâtir ces modèles. Ce sont des géants dans le domaine du deep learning et des atouts majeurs dans la conception de ce projet.

Par ailleurs, python possède une grande communauté de développeurs, qui est toujours prête à aider ses membres. Si nous avons un problème, quelqu'un d'autre l'a sûrement déjà recontré.

Finalement, python est la référence du deep learning. La plupart des librairies et outils sont conçus pour ou tournent sur du python. Le choix d'un tel langage pour effectuer du deep learning coule donc de source.

4.1.2.1. TensorFlow

Comme le décrit son site officiel, TensorFlow est une bibliothèque open source pour le développement et l'entraînement de modèles de machine learning. TensorFlow nous fournit une collection riche d'outils pour bâtir facilement des réseaux de neurones. Il est utilisable à toute échelle, depuis un processeur jusqu'à des infrastructures conséquentes en passant par une simple carte graphique. TensorFlow se donne comme mission de fournir une plateforme de machine learning pour tous. (TensorFlow, 2021).

TensorFlow est donc un choix judicieux pour ce projet, étant de loin la référence actuelle pour implémenter du deep learning. Il est open source, stable, avec une grande communauté et possède une architecture simple et flexible qui permet d'entraîner des modèles sur toutes les plateformes.

4.1.2.2. Keras

En se référant à sa documentation, Keras est une Application Programming Interface [API] développée pour les humains et non pour les machines. Keras offre une API consistante et simple d'utilisation. Elle permet de minimiser le nombre d'actions utilisateur pour des cas d'emplois communs, tout en fournissant des messages d'erreur clairs et exploitables. Par ailleurs, la communauté est présente et la documentation est complète. De plus, Keras est construite sur la base de TensorFlow 2.0. (Keras, 2021d).

Keras est donc un choix d'API pertinent dans le cadre de ce projet, étant facile d'utilisation et rapide pour mettre en place une nouvelle structure. Construit au-dessus de TensorFlow, Keras symbolise le mariage parfait entre toutes les étapes d'un processus de machine learning, passant de la gestion des données à l'entraînement des modèles, jusqu'au déploiement des solutions.

4.2. Algorithmes

Pour réaliser ce projet, un grand nombre d'algorithmes est disponible. Le sens des données du dataset nous étant inconnu, le seul paramètre que nous possédons pour trier ces algorithmes se résume à la prise de données faite sous forme de signaux temporels.

Deux types d'algorithmes qui sont capables d'apprendre d'eux-mêmes à partir des données seront évalués, ceux de « Machine Learning Traditionnel » et ceux de « Deep Learning ». Le tableau 4 résume les différences clés entre le machine learning traditionnel et le deep learning.

	Machine Learning Traditionnel	**Deep Learning**	***Le meilleur*** *par rapport à ce projet*
Dataset d'entraînement	Petit	Grand	*Machine Learning*
Caractéristiques	A définir	Apprises automatiquement	*Deep Learning*
Temps d'entraînement	Court	Long	/
Analyse de données non structurées (image, …)	Difficile	Facile	*Deep Learning*
Débogage	Facile	Difficile	*Machine Learning*

Tableau 4 - Machine Learning vs Deep Learning, MATLAB[9]

Le machine learning traditionnel et le deep learning possèdent tous deux des avantages et des inconvénients. Notre dataset d'entraînement étant limité, le machine learning traditionnel débute avec un avantage. Cependant, nos datasets sont quand même assez volumineux pour implémenter une technologie de deep learning à petite échelle.

Le but premier du projet est de classer des Bursts naïvement sans intervention humaine. Le deep learning, qui permet d'entraîner un modèle sans avoir de connaissance sur les données, apporte une solution à cette problématique.

Le temps d'entraînement n'est pas une variable importante dans le cadre de ce projet. En effet, selon le client, dans le domaine médical, l'entraînement d'un modèle est rare à cause des normes de vérifications. Ainsi, même si l'entraînement du modèle est long, il serait si rare que sa durée n'impacterait pas sa qualité.

Le deep learning est capable d'analyser des données non-structurées comme des images ou des vidéos, ce qui est difficile pour du machine learning traditionnel. Les modèles de deep learning sont plus précis la plupart du temps lorsque les datasets sont volumineux. Cette qualité impose

[9] Introduction to Deep Learning: https://www.youtube.com/watch?v=-SgkLEuhfbg&ab_channel=MATLAB

néanmoins des ressources plus conséquentes et un certain travail dans la configuration des réseaux de neurones. Si le modèle n'est pas parfaitement paramétré, il ne dévoilera pas son plein potentiel.

Lors du débogage, il est plus facile de trouver des problèmes ou des erreurs dans un modèle de machine learning traditionnel que dans un modèle de deep learning. En effet, comme nous ne connaissons pas les caractéristiques relevées et la manière d'apprendre des réseaux de neurones, il est presque impossible de déboguer ce genre de modèle, comparable à une boîte noire.

Au vu des données récoltées lors des recherches, le deep learning présente des intérêts significatifs pour ce projet. Même si le deep learning sort du cadre premier de ce travail, nous prenons la décision de l'implémenter aux côtés des algorithmes traditionnels de machine learning.

4.2.1. Choix des modèles de machine learning traditionnel

Pour trier les algorithmes de machine learning traditionnel, nous nous sommes concentrés sur les algorithmes de type supervisé, qui démontre un avantage dans des données de matrices brutes.

Le Decision Tree, le Random Forest et le Gradient Boosted Tree sont des algorithmes connus pour être flexibles. Ces trois modèles se révèlent donc pertinents pour entraîner des données sans connaître leurs valeurs ou leurs caractéristiques.

Le Support Vector Machine et le Fuzzy Rule sont des algorithmes déjà plus précis et spécialisés. Ils exigent un entraînement plus long et dévoilent de plus grandes difficultés à progresser lors d'un entraînement. Cependant, leurs spécialités font leur force et pourraient révéler des résultats plus positifs que des modèles qualifiés de polyvalents.

Les algorithmes retenus sont donc : le Decision Tree, le Random Forest, le Gradient Boosted Tree, le Support Vector Machine et le Fuzzy Rule.

Ces algorithmes démontrent tous des avantages significatifs d'un point de vue d'entraînement ou de qualité. Les expérimentations pourront prouver l'effet de chacun. Il est malheureusement impossible de déceler des patterns avec une technologie classique de classification supervisée. Ces modèles seront donc pertinents pour classer les patients atteints de glaucome, mais ne seront d'aucune aide dans l'extraction de caractéristiques dans les données.

4.2.2. Choix des modèles de deep learning

De nombreux algorithmes de deep learning existent pour gérer presque chaque situation. Nous avons analysé les types de deep learning et sélectionné ceux qui sont optimisés pour gérer des traitements de signaux 1D.

Selon la configuration du réseau de neurones, le deep learning pourrait être une solution plus coûteuse en termes de temps et de ressources. Cependant, tout indique que les résultats pourraient se révéler plus précis qu'avec des modèles traditionnels.

Le premier modèle choisi est le réseau de neurones multicouches. Ce réseau de neurones est un modèle de deep learning simple qui atteint rapidement des résultats. Il permettra de se familiariser avec python et le deep learning, tout en présentant un constat simple et pertinent vis-à-vis de cette technologie.

Le second modèle choisi est le Convolutional Neural Network 1D. Le CNN 1D est un réseau de neurones taillé pour gérer les données à une dimension et les traitements de signaux. De plus, sa logique de convolution permettra de détecter des caractéristiques dans les données, un point essentiel pour le client.

Les algorithmes retenus pour le deep learning sont donc : le réseau de neurones multicouches et le Convolutional Neural Network 1D.

5. Expérimentations

Nous allons mener une série d'expérimentations pour comparer les différents algorithmes retenus, affiner toujours plus le dataset et trouver le modèle qui prédira les meilleurs résultats, tout en extrayant des caractéristiques dans les données.

Nous débuterons par une comparaison des algorithmes traditionnels de machine learning grâce à la plateforme KNIME (voir annexe V, version 4.3.2). Dans un second temps, le dataset sera au cœur de nos préoccupations lorsque nous mènerons une étude de corrélation et une élimination de données. Puis, nous construirons notre premier réseau de neurones multicouches, que nous allons ensuite optimiser. A la suite de ces résultats, nous créerons un modèle CNN 1D et nous l'optimiserons pour gérer le bruit lors de la prédiction. Une fois que nos modèles seront opérationnels, nous vérifierons leur qualité avec des techniques comme la cross-validation. Une analyse des prédictions par patient et non par Burst sera menée et finalement, nous préparerons un environnement complet pour déployer l'application chez le client. Ces expérimentation seront analysées et validées par le représentant de Sensimed durant tout le long du projet.

5.1. Comparaison des algorithmes de machine learning traditionnel

5.1.1. Expérimentation des algorithmes de machine learning traditionnel

Le but de cette expérimentation est de tester, à l'aide de l'outil KNIME, tous les algorithmes de classification traditionnels retenus, afin d'établir si cette solution est la plus adéquate pour notre problématique. Nous avons entraîné un dataset avec des données normalisées pour analyser la qualité des algorithmes. Le dataset primaire du client a été divisé et normalisé (voir chapitre 3, Dataset). La notion de Covariate (ensemble de données qui ne concernent pas la pression de l'œil) a été retirée du dataset à la demande du client. Les algorithmes testés lors de cette expérimentation sont le Gradient Boosted Tree, le Decision Tree, le Random Forest, le SVM et le Fuzzy Rule. La figure 41 expose les nœuds configurés pour le Gradient Boosted Tree.

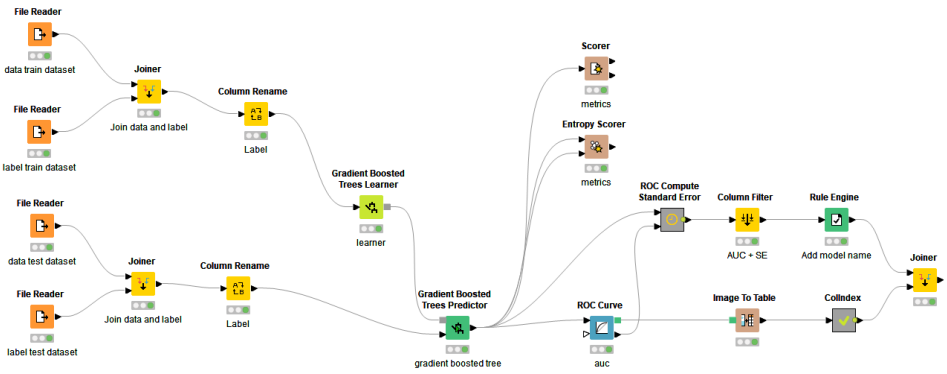


Figure 41 - Processus pour la création d'un modèle de classification depuis KNIME (voir annexe VIII) - Image de l'auteur

5.1.2. Résultats de la classification des algorithmes de machine learning traditionnel

Le tableau 5 résume les résultats de cette expérimentation, en mettant en relation l'AUC et la Standard Error des différents modèles.

Performance des différents AUC	
Algorithmes	AUC^{SE}
Gradient Boosted Tree	$0.722^{+/-0.00564}$
Decision Tree	$0.500^{+/-0.00580}$
Random Forest	$0.722^{+/-0.00564}$
Support Vector Machine	$0.484^{+/-0.00575}$
Fuzzy Rule	$0.487^{+/-0.00576}$

Tableau 5 - Résultats de l'expérimentation des algorithmes de machine learning traditionnel

Les algorithmes de Random Forest et de Gradient Boosted Tree obtiennent les meilleurs résultats avec un AUC de 0.722 et une Standard Error de 0.00564.

Les résultats négatifs du SVM, par exemple, démontrent que les données fournies par Sensimed ne sont pas totalement linéaires. Ce genre de modèle ne sera donc pas efficace pour notre dataset. Par ailleurs, ces modèles sont plus compliqués à entraîner.

La figure 42 dévoile l'AUC du Random Forest. Le résultat est déjà intéressant sachant qu'aucune optimisation n'a été effectuée sur le dataset ou sur le modèle.

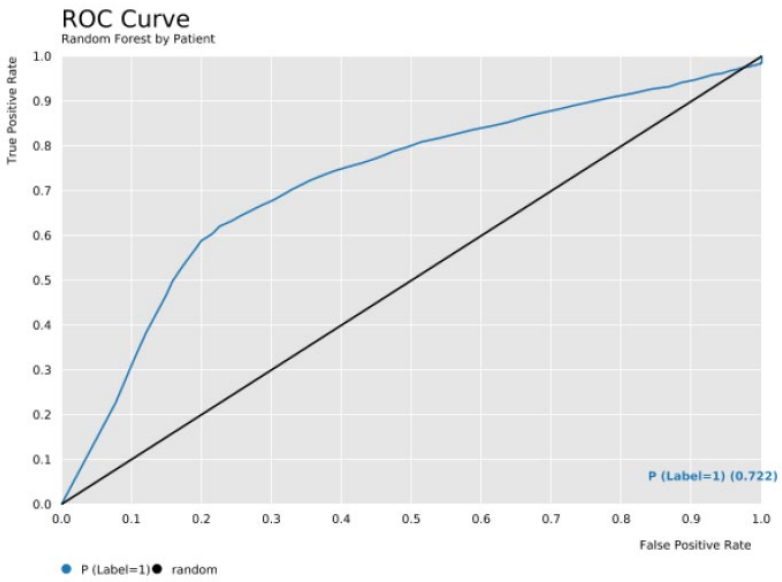


Figure 42 - Graphique de l'AUC de l'algorithme de Random Forest - Image de l'auteur

Les datasets évolueront au fil des expérimentations. Nous avons donc repris les mêmes modèles avec ces nouveaux datasets. Les résultats finaux sont exposés dans le tableau 6.

Performance des différents AUC	
Algorithmes	AUC
Gradient Boosted Tree	$0.755^{+/-0.006}$
Decision Tree	$0.644^{+/-0.007}$
Random Forest	$0.745^{+/-0.006}$
Support Vector Machine	$0.631^{+/-0.007}$
Fuzzy Rule	$0.609^{+/-0.007}$

Tableau 6 - Résultats finaux des algorithmes de machine learning traditionnel

5.1.3. Conclusion de la classification des algorithmes de machine learning traditionnel

L'algorithme du Gradient Boosted Tree permet d'établir un modèle qui détecte des Bursts malades avec un AUC de $0.755^{+/-0.006}$ et une précision de $69.6\%^{+/-0.6}$ prédictions correctes. Les algorithmes traditionnels de machine learning sont donc des technologies viables pour notre problématique.

5.2. Extraction de corrélation et possibilité d'élimination de données

Le volume de données étant important, une analyse précise des données permettra de limiter le temps de calcul des serveurs. Le but de cette expérimentation est d'affiner le dataset afin d'améliorer le volume d'apprentissage de la classification et d'éliminer ou de hiérarchiser les données à disposition.

5.2.1. Préparation des données

Le but de cette étape est de transformer le dataset originel en un dataset le plus léger et affiné possible. Pour ce faire, nous allons tester et mettre en application des nœuds préconfigurés dans KNIME. Les trois datasets ne subiront donc pas d'altération.

5.2.2. Backward Feature Elimination

Ce nœud va prédire, à partir du meilleur modèle pour le dataset, dans notre cas le Random Forest ou le Gradient Boosted Tree, chaque combinaison (ensemble de colonnes assemblées aléatoirement) possible pour le dataset d'entrée et va analyser leur précision. L'objectif final est de découvrir les données nécessaires pour atteindre une certaine précision, ce qui permettra d'éliminer les attributs moins utiles à la prédiction et donc de gagner en volume de données. La figure 43 démontre la configuration de la Feature Elimination avec le Random Forest.

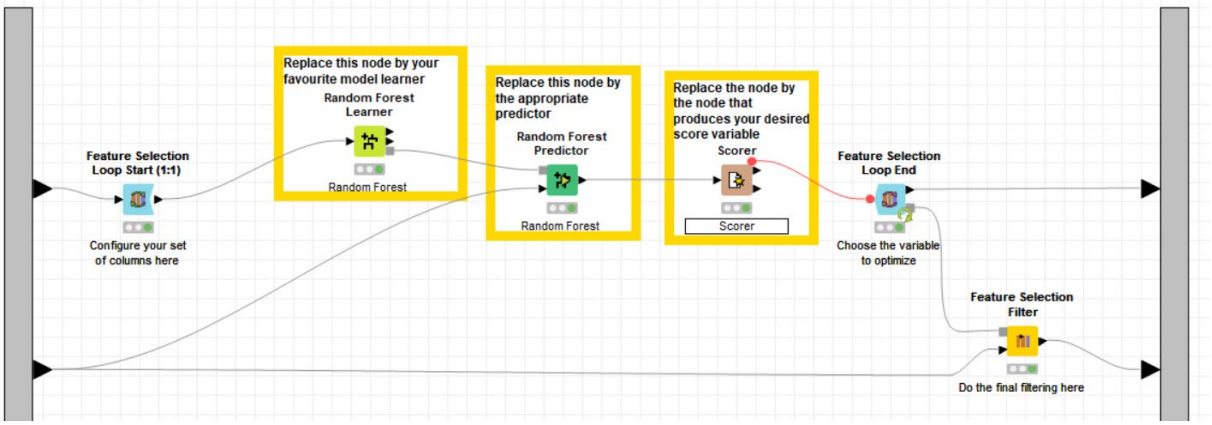


Figure 43 - Backward Feature Elimination configurée dans KNIME - Image de l'auteur

A cette heure, comme le démontre la figure 44, le meilleur résultat ne comporte que deux colonnes en plus de celles d'informations. Un nombre si faible de colonnes peut néanmoins laisser perplexe. Les résultats sur plusieurs modèles pourront prouver leur efficacité sur des algorithmes de classification traditionnels.

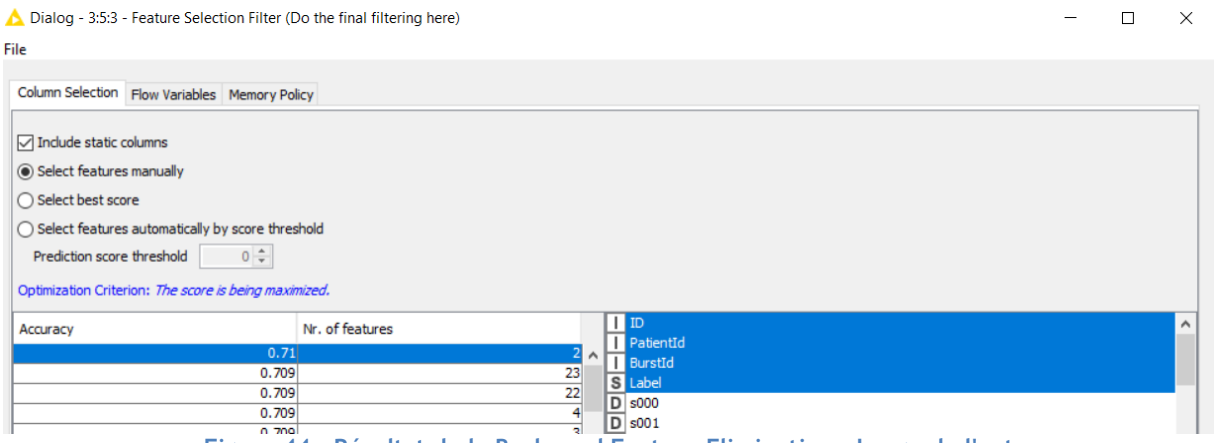


Figure 44 - Résultat de la Backward Feature Elimination - Image de l'auteur

5.2.3. Linear Correlation

Après avoir trouvé la meilleure combinaison d'attributs, il est intéressant de découvrir leur corrélation, soit à quel point ces attributs sont dépendants les uns des autres. Cette action permet de savoir si un ordre temporel est nécessaire ou non pour atteindre ce résultat. La figure 45 expose le lien entre la Backward Feature Elimination et la Linear Correlation.

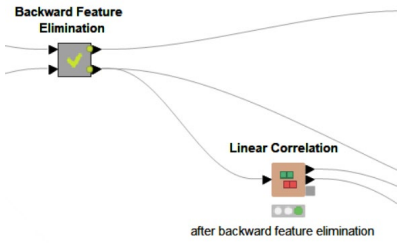


Figure 45 - Image des nœuds configurés dans KNIME (voir annexe VIII) - Image de l'auteur

En se basant sur les résultats de la figure 46, la corrélation entre les colonnes se révèle presque absolue. Le facteur temporel est donc important dans ce cas de figure. Néanmoins, vu que le nombre de colonnes a été réduit à deux, il est normal que leur corrélation soit aussi forte.

Correlation Matrix

Row ID	s201	s292
s201	1	0.9895925957145669
s292	0.9895925957145669	1

Figure 46 - Détail des corrélations entre colonnes - Image de l'auteur

5.2.4. Résultats de l'élimination et de la corrélation

Les modules d'élimination et de corrélation ont prouvé, avec le résultat de l'AUC, que le dataset pouvait être affiné avec l'élimination de 300 colonnes au prix de quelques pourcents de prédiction en moins. En termes de chiffres, comme le démontrent les graphes ci-dessous (figure 47), les résultats de l'AUC du Gradient Boosted Tree ont chuté de -0,045 et les résultats de la Standard Error

ont augmenté de +0.001, passant d'un AUC de $0.755^{+/-0.006}$ avant l'élimination à un AUC de $0.71^{+/-0.007}$ après élimination.

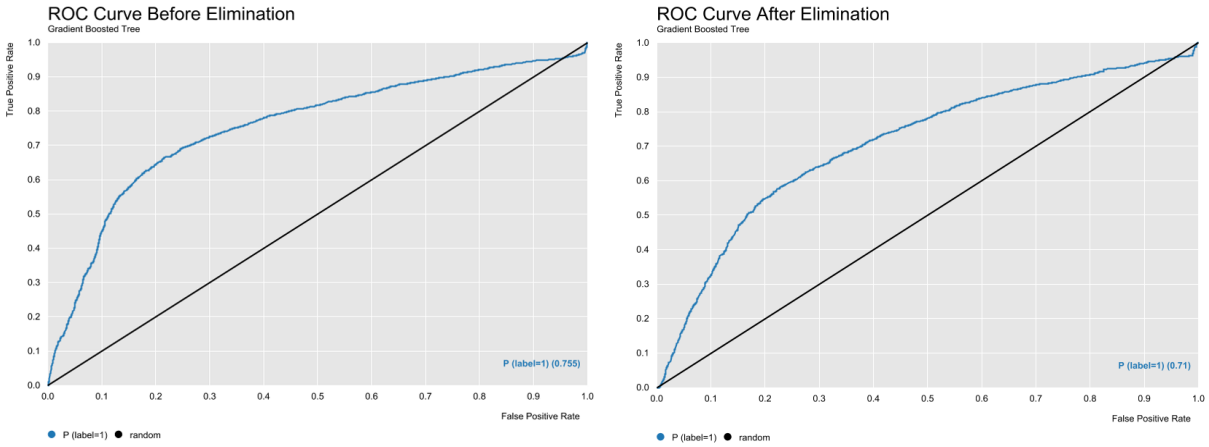


Figure 47 - Différence d'AUC avant (gauche) et après (droite) l'élimination - Image de l'auteur

Éliminer autant de données peut être une fausse bonne idée, augmentant drastiquement le risque d'overfitting, ce qui impacte la généralisation de la prédiction.

5.2.5. Conclusion de la corrélation et de l'élimination

La taille du dataset peut être drastiquement réduite avec le Random Forest ou le Gradient Boosted Tree, limitant le temps de calcul. Cependant, la qualité de la classification est impactée. Un rapport de qualité/temps doit donc être posé. Dans notre cas, perdre de la qualité contre du temps s'avère contreproductif. Par ailleurs, ce genre d'élimination ne correspond pas au deep learning, qui demande au contraire plus de données pour son entraînement. C'est pourquoi nous utiliserons l'intégralité des données lors de ce projet.

5.3. Implémentation d'un modèle de réseau de neurones à plusieurs couches

L'expérimentation suivante vise à prouver la qualité d'un réseau simple de neurones sans métrique spécifique ni optimisation. Pour ce faire, nous implémenterons à la main un système basé sur un perceptron, que nous transformerons en un réseau à plusieurs couches. Lors de cette expérimentation, nous utiliserons la librairie TensorFlow (voir annexe V, version 2.5.0) proposée par Google et l'environnement de développement python PyCharm (voir annexe V, version 2021.1) proposé par JetBrains. Tous les principaux algorithmes d'optimisation, d'activation et de perte seront testés et comparés lors de ce chapitre.

5.3.1. Préparation des données

Pour que les datasets s'accordent aux prérequis de TensorFlow, il faut diviser les trois datasets, entraînement, test et validation, en trois datasets « données » qui ne contiendront que les données et trois datasets « labels » qui contiendront les classes de chaque Burst, comme le montrent les figures 48 et 49.

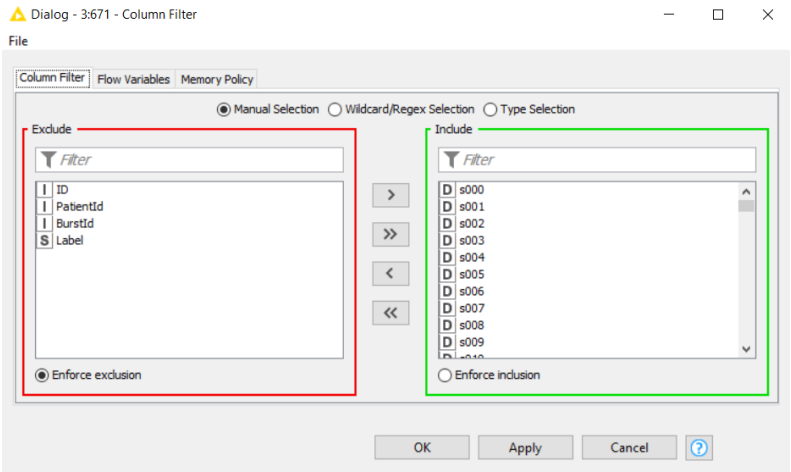


Figure 48 - Dataset d'entraînement, partie données - Image de l'auteur

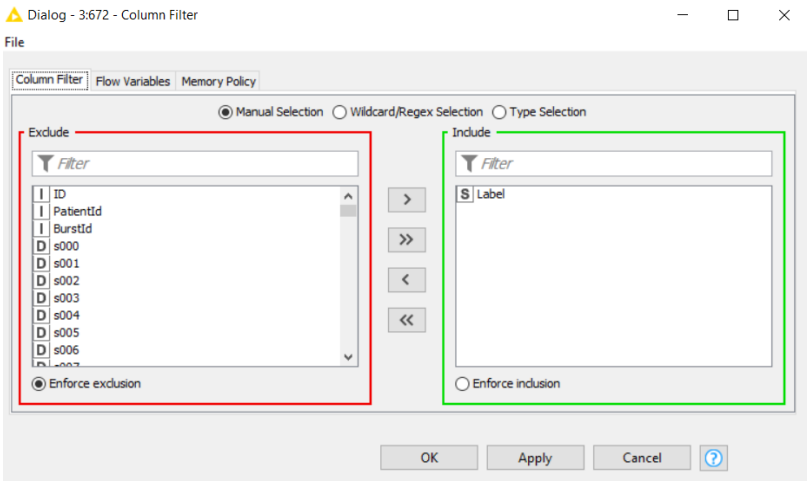


Figure 49 - Dataset des labels, partie classes - Image de l'auteur

Nous allons ensuite récupérer les fichiers CSV extraits de KNIME dans PyCharm à l'aide de la librairie Panda (voir annexe V, version 0.3.1).

5.3.2. Nombre de couches cachées et de neurones

Le modèle recherché lors de cette expérimentation est le plus simple possible. La première question à se poser est donc : combien de couches cachées et combien de neurones faut-il implémenter ?

Selon le papier « Review on Methods to Fix Number of Hidden Neurons in Neural Networks » (Gnana Sheela & Deepa, 2013), les méthodes existantes se baseraient sur un système de trail-and-error, soit utiliser le bon sens pour trouver la meilleure combinaison. Plus le nombre de neurones et de couches sont faibles, plus les chances de provoquer de l'underfitting sont importantes. Au contraire, plus le nombre de neurones et de couches sont élevées plus les chance de provoquer de l'overfitting sont grandes, sans parler du temps supplémentaire nécessaire à l'entraînement du modèle. Dans l'exemple de ce papier, les scientifiques utilisaient un réseau de neuronnes à trois couches.

Selon « Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture » (Karsoliya, 2012), en augmentant le nombre de couches à trois, l'accuracy peut augmenter, mais la demande en calcul sera aussi bien plus grande.

Pour déterminer cette différence, à la suite de la modélisation, un système à 10 couches sera comparé à un modèle à deux couches et à un modèle à six couches dont le nombre de neurones est divisé par deux à chaque couche. Le temps, représentant la demande en calcul, et l'accuracy lors du test seront comparés pour déterminer quelle architecture est la plus pertinente.

Architecture	Neurones	Paramètres	Accuracy^{SE}	Temps #itération	Temps Total
2 couches cachées de 302 neurones	907	274'821	72,00%$^{+/-0.659}$	31 secondes	~51 minutes
10 couches cachées de 302 neurones	3'323	1'006'896	69,55%$^{+/-0.693}$	64 secondes	~106 minutes
6 couches cachées de X/2 neurones	597	152'087	71,36%$^{+/-0.692}$	32 secondes	~53 minutes

Tableau 7 - Résultats de l'analyse des couches avec des modèles entraînés avec 100 itérations

Selon le tableau 7, le modèle qui ne possède que deux couches cachées de 302 neurones démontre l'accuracy la plus élevée et le temps d'entraînement le plus court. Le modèle aura donc cette configuration de base. Notons que les six couches cachées avec à chaque fois deux fois moins de neurones surpassent les 10 couches cachées de 302 neurones, même avec près de six fois moins de paramètres entraînables.

5.3.3. Modélisation du réseau de neurones

La première couche contiendra l'ensemble des données sur la pression de l'œil. Elle possédera donc les 302 colonnes du dataset comme données d'entrée.

Le modèle comportera ensuite deux couches cachées de 302 neurones, comme démontré dans le chapitre précédent. L'activation ReLu sera choisie pour faciliter le temps de calcul et éviter que le gradient ne disparaisse. Cette fonction d'activation sera utilisée à chaque couche pour rendre les données non-linéaires.

Finalement, la sortie du modèle sera activée par une fonction d'activation sigmoïde qui convertira le résultat de sortie dans un format entre 0 et 1.

Le modèle sera entraîné 100 fois avec des lots entiers pour éviter le mélange de patients.

L'extrait du code 1 illustre la construction d'un modèle de réseau multicouches.

```
…
model = Sequential()
model.add(Dense(302, activation='relu', input_dim=302))
model.add(Dense(302, activation='relu'))
model.add(Dense(302, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
…
```

Code 1 - Extrait du code du modèle de réseau de neurones multicouches (voir annexe IV)

La figure 50 démontre le résultat du modèle obtenu dans la console de l'IDE de PyCharm.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 302)               91506
_________________________________________________________________
dense_1 (Dense)              (None, 302)               91506
_________________________________________________________________
dense_2 (Dense)              (None, 302)               91506
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 303
=================================================================
Total params: 274,821
Trainable params: 274,821
Non-trainable params: 0
_________________________________________________________________
```

Figure 50 - Résumé de chaque couche depuis la console de PyCharm - Image de l'auteur

5.3.4. Implémentation de la Standard Error

La Standard Error est une métrique statistique qui permet de définir un intervalle de confiance en une mesure donnée. Sans cette métrique, il est impossible de comparer nos différents résultats. Pour remédier à ce problème, une méthode calculant la Standard Error sera implémentée.

Selon le papier de Hanley & McNeil, « The meaning and use of the area under a receiver operating characteristic (ROC) curve », il serait possible de générer la Standard Error à partir du dataset contenant tous les labels et de l'AUC généré. La formule de la figure 51 illustre la méthode permettant de calculer la SE.

$$SE = Z_\alpha \sqrt{\frac{p(1-p)}{n}}$$

Où :

- *Z_α : niveau de confiance à 95%, Z_α = 1.96*
- *p : est la précision*
- *n : est le nombre de données, inclues dans les deux classes*

Figure 51 - Formule de la Standard Error (Hanley & McNeil, 1982)

Maintenant que la Standard Error est disponible (voir annexe IV), elle accompagnera toutes les métriques de Keras mises à disposition à travers notre implémentation.

5.3.5. Détermination de la fonction de perte et d'optimisation

Les fonctions de perte et d'optimisation sont des algorithmes appelés à la configuration du modèle de réseau de neurones qui vont permettre selon les attentes, le contexte ou encore le dataset d'augmenter l'accuracy d'un modèle, tout en diminuant sa perte.

La perte est calculée sur la base de l'entraînement et de la validation du modèle. Elle démontre comment le modèle se comporte après chaque itération d'optimisation. Plus la perte est basse, meilleur est le modèle. Au contraire, l'accuracy permet de mesurer la qualité d'un modèle de réseau de neurones, soit le pourcentage de prédictions correctes.

Le but de cette expérience est de démontrer quel algorithme de perte et quel algorithme d'optimisation fonctionnent le mieux dans notre contexte avec notre architecture et notre dataset. Pour ce faire, 16 modèles, configurés avec les différentes paires d'algorithmes de perte et d'optimisation, ont été entraînés avec 100 itérations.

5.3.6. Résultats de l'implémentation du réseau de neurones multicouches

Les tableaux 8 et 9 exposent le résultat des taux de perte et des taux d'accuracy des différents algorithmes. Les lignes correspondent aux fonctions de perte et les colonnes aux différentes fonctions d'optimisation (voir chapitre 2.4.5 Fonctions de perte et d'optimisation).

Taux de perte : fonction de perte/fonction d'optimisation								
	SGD	Adam	Adadelta	Adagrad	RMSProp	Nadam	Ftrl	Adamax
Binary Crossentropy	125,9% +/-0.0072	911,6% +/-0.0064	58.98% +/-0.0063	54.55% +/-0.0061	62.37% +/-0.0073	845.4% +/-0.0067	69.31% +/-0.0076	731.1% +/-0.0071
Mean Square Error	15.5% +/-0.0064	21.06% +/-0.0066	19.07% +/-0.0063	18.49% +/-0.0061	26.81% +/-0.0070	21.06% +/-0.0066	21.08% +/-0.0076	25.00% +/-0.0076

Tableau 8 - Taux de perte par algorithme après 100 itérations

Taux d'accuracy : fonction de perte/fonction d'optimisation								
	SGD	Adam	Adadelta	Adagrad	RMSProp	Nadam	Ftrl	Adamax
Binary Crossentropy	71,62% +/-0.0072	69,10% +/-0.0064	72.15% +/-0.0063	75.23% +/-0.0061	65.85% +/-0.0073	73.19% +/-0.0067	49.59% +/-0.0076	70.43% +/-0.0071
Mean Square Error	77.96% +/-0.0064	72.71% +/-0.0066	72.05% +/-0.0063	73.44% +/-0.0061	63.51% +/-0.0070	64.65% +/-0.0066	49.59% +/-0.0076	71.31% +/-0.0076

Tableau 9 - Taux d'accuracy par algorithme après 100 itérations

5.3.7. Conclusion de l'implémentation du réseau de neurones multicouches

Malgré la simplicité du modèle, l'algorithme de perte Mean Square Error obtient déjà de meilleurs résultats, écartant la Binary Crossentropy pour le reste de l'expérimentation.

L'algorithme de perte le plus performant dans notre contexte est donc la Mean Square Error avec un algorithme d'optimisation Stochastic Gradient Descent. On peut néanmoins souligner que la différence de résultats entre les algorithmes d'optimisation est moindre, en comparaison des algorithmes de perte.

Du côté de l'accuracy, les différences entre les algorithmes sont moins pertinentes, dévoilant quand même un avantage pour le MSE et le SGD.

Comme le démontre le tableau 10, le premier résultat sur un modèle simple de réseau de neurones entraîné 200 fois avec les algorithmes de perte MSE et d'optimisation SGD a dévoilé un taux de perte moyen de $19{,}61\%^{+/-0.064}$ et un taux de précision de $72{,}74\%^{+/-0.64}$.

	Perte^{SE}	Accuracy^{SE}
Entraînement	$14{,}11\%^{+/-0.64}$	$80{,}17\%^{+/-0.64}$
Test	$21{,}78\%^{+/-0.64}$	$70{,}18\%^{+/-0.64}$
Validation	$22{,}95\%^{+/-0.64}$	$67{,}87\%^{+/-0.64}$
Moyenne	**$19{,}61\%^{+/-0.64}$**	**$72{,}74\%^{+/-0.64}$**

Tableau 10 - Résultats de l'analyse des fonctions de perte et d'optimisation

5.4. Optimisation d'un réseau de neurones multicouches

Le but de cette expérimentation est de trouver les meilleures configurations possible pour le prototype de réseau de neurones multicouches. Pour ce faire, nous devrons prouver ses résultats avec des métriques sûres du modèle, vérifier les algorithmes d'activation et trouver le nombre idéal d'itérations pour éviter le phénomène d'overfitting.

5.4.1. Modélisation du réseau de neurones multicouches optimisé

Le modèle de base sera repris de la dernière expérimentation. Ce modèle est de type séquentiel, avec une première couche créée avec les 302 colonnes du dataset comme entrées. Le modèle comporte ensuite deux couches cachées de 302 neurones. L'activation ReLu est utilisée à chaque couche pour rendre les données non-linéaires. Finalement, la sortie du modèle est activée par une sigmoïde qui transforme les résultats en valeurs comprises entre 0 et 1.

Selon nos précédentes expérimentations, l'algorithme d'optimisation est le SGD et l'algorithme de perte le MSE. Le modèle sera entraîné 200 fois avec des lots entiers pour éviter le mélange de patients. De plus, selon la documentation de Keras (2021c), les métriques nécessaires, comme l'AUC ou le recall, seront ajoutées à la compilation.

Le tableau 11 illustre la configuration du compilateur du modèle.

Optimisateur	SGD
Perte	MSE
Métriques	Accuracy
	Précision
	Recall
	AUC
	True Negatives
	False Negatives
	True Positives
	False Positives

Tableau 11 - Configuration du compilateur du modèle (voir annexe IV)

5.4.2. Évaluation du réseau de neurones multicouches

Comme le montre la matrice de confusion de la figure 52, l'algorithme classe tous les Bursts comme étant malades, peu importe leur valeur, ce qui confère une grande précision lors de l'entraînement mais qui se révèle totalement faux lors des tests.

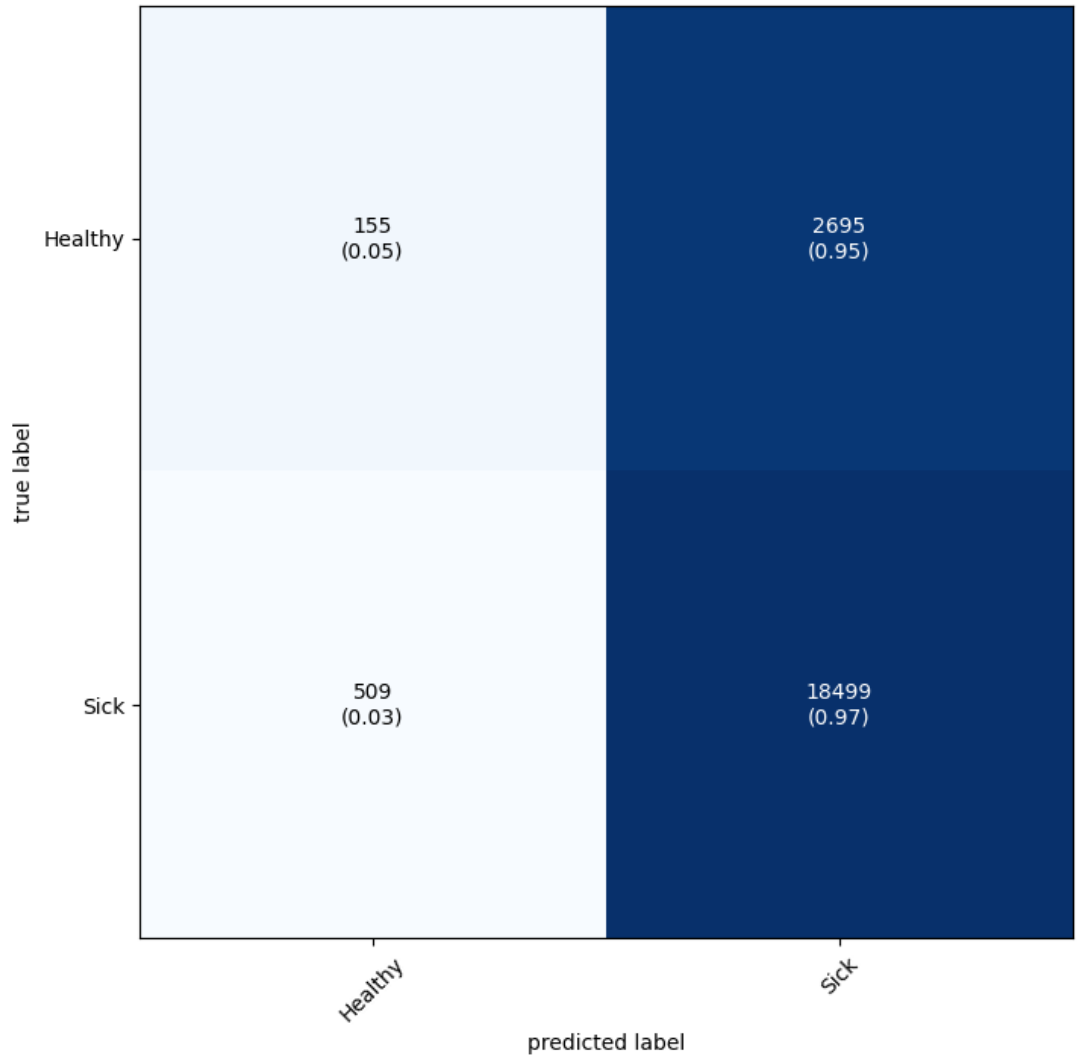


Figure 52 - Matrice de confusion prouvant la mauvaise performance du modèle - Image de l'auteur

5.4.3. Compréhension des résultats et des données

A ce stade, deux options expliquent un tel résultat.

1) Le dataset n'est pas équilibré. Il y a beaucoup plus de personnes malades que saines.

2) La configuration du modèle de réseau de neurones n'est pas optimale.

Nous testerons et documenterons donc les deux options.

5.4.4. Préparation des données pour contrer le manque d'équilibre

Pour équilibrer le nombre de patients sains et malades, nous découperons chacun des trois datasets en nombres équilibrés de Bursts. Pour ce faire, les étapes suivantes seront respectées pour chaque dataset :

1) Séparer les patients malades et sains par leur label, comme illustré dans la figure 53.

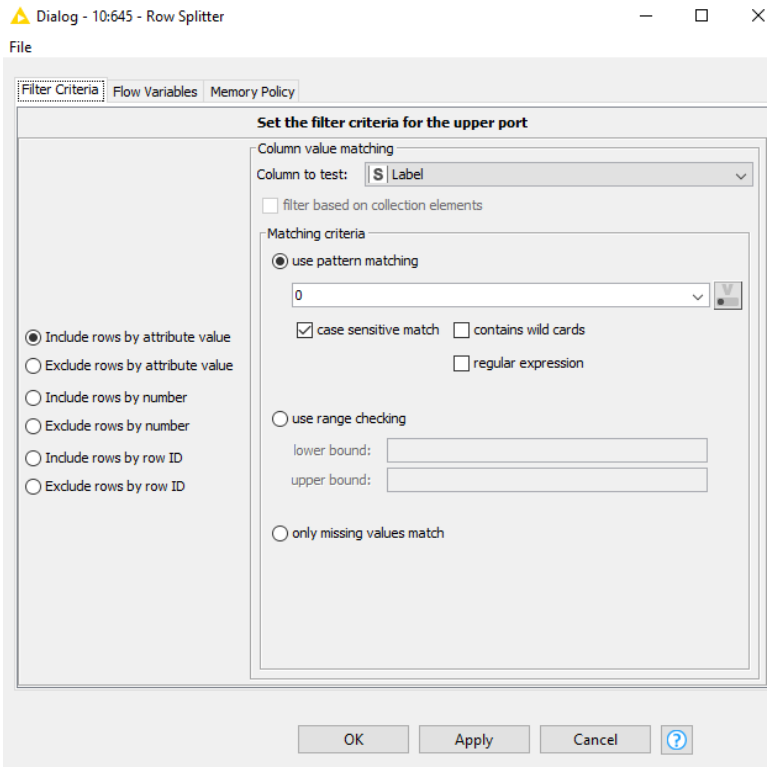


Figure 53 - Séparation des données par les labels (0) depuis KNIME - Image de l'auteur

2) Limiter le nombre de patients malades par rapport au nombre total de patients sains, tout en veillant à prendre tous les Bursts du dernier patient malade (figure 54).

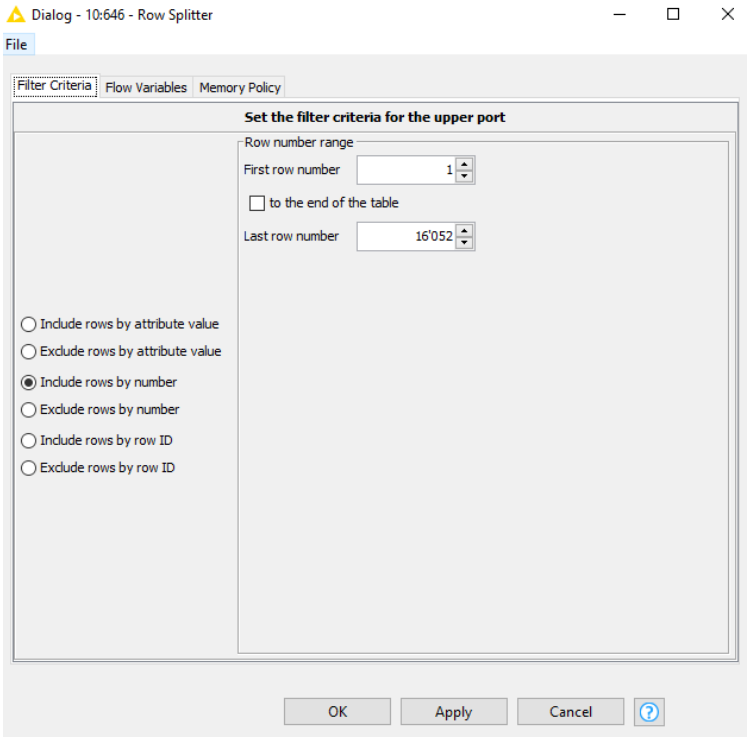


Figure 54 - Découpage du dataset d'entraînement depuis KNIME - Image de l'auteur

3) Concaténer les deux résultats.

4) Séparer les données des labels et les extraire en fichiers CSV.

Le dataset d'entraînement avait initialement 121'022 Bursts. Il se retrouve actuellement avec 32'108 Bursts, perdant environ 76,47% des données du dataset.

5.4.5. Évaluation du réseau de neurones multicouches avec des datasets équilibrés

Les résultats, démontrés par la figure 55, sont meilleurs, n'étant pas automatiquement prédits comme étant malades, mais l'ordre des Bursts, tous les sains, puis tous les malades, n'est pas optimal.

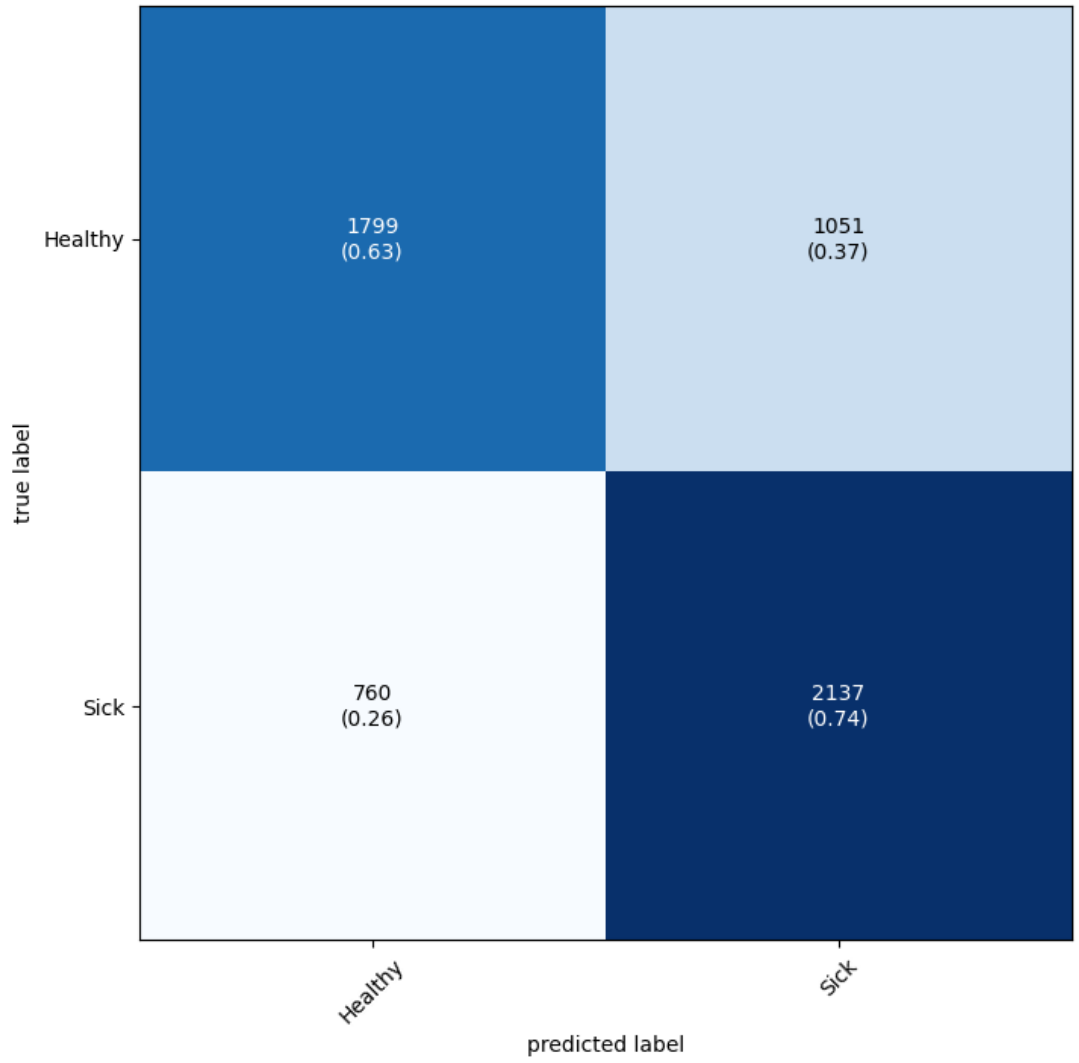


Figure 55 - Matrice de confusion d'un réseau multicouches équilibré - Image de l'auteur

5.4.6. Mélange des données équilibrées

Pour contrer le problème de l'ordre sains/malades, les trois dataset seront mélangés avec le nœud « Shuffle » de KNIME, comme dans l'exemple de la figure 56.

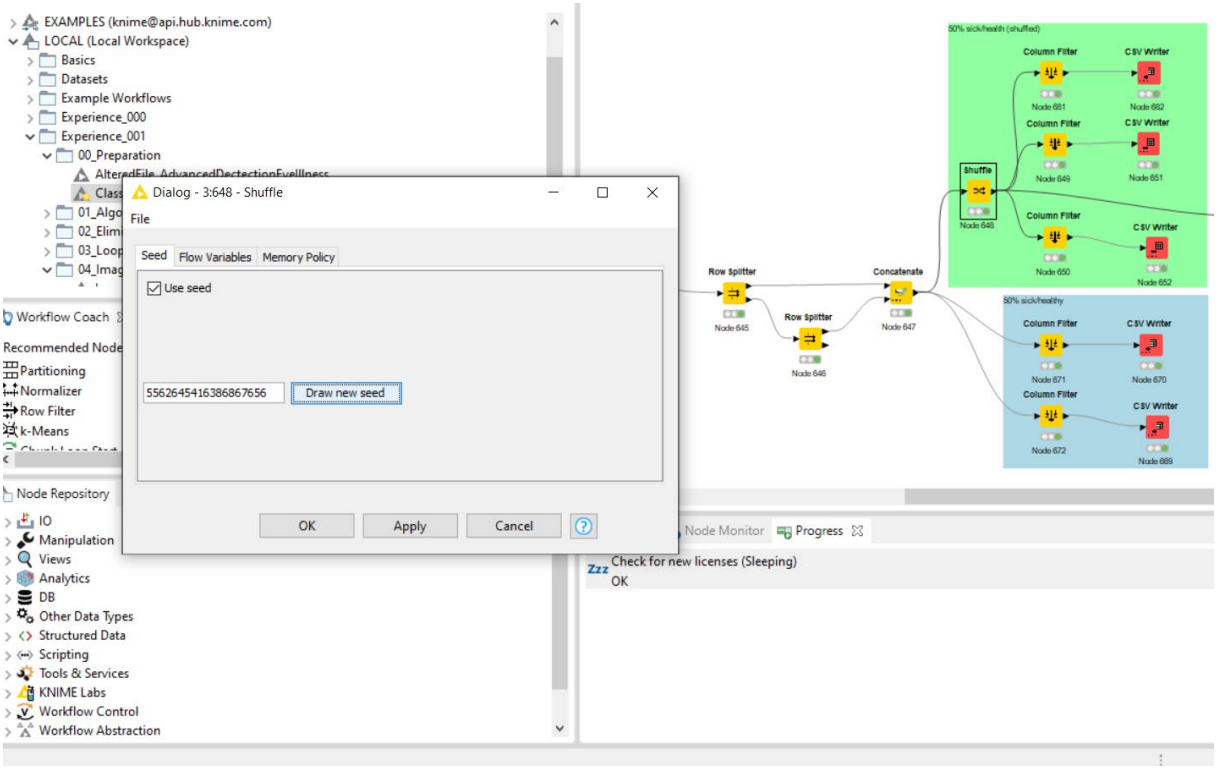


Figure 56 - Noeud Shuffle sur le dataset d'entraînement depuis KNIME - Image de l'auteur

La figure 57 expose le résultat du mélange après la transformation du dataset.

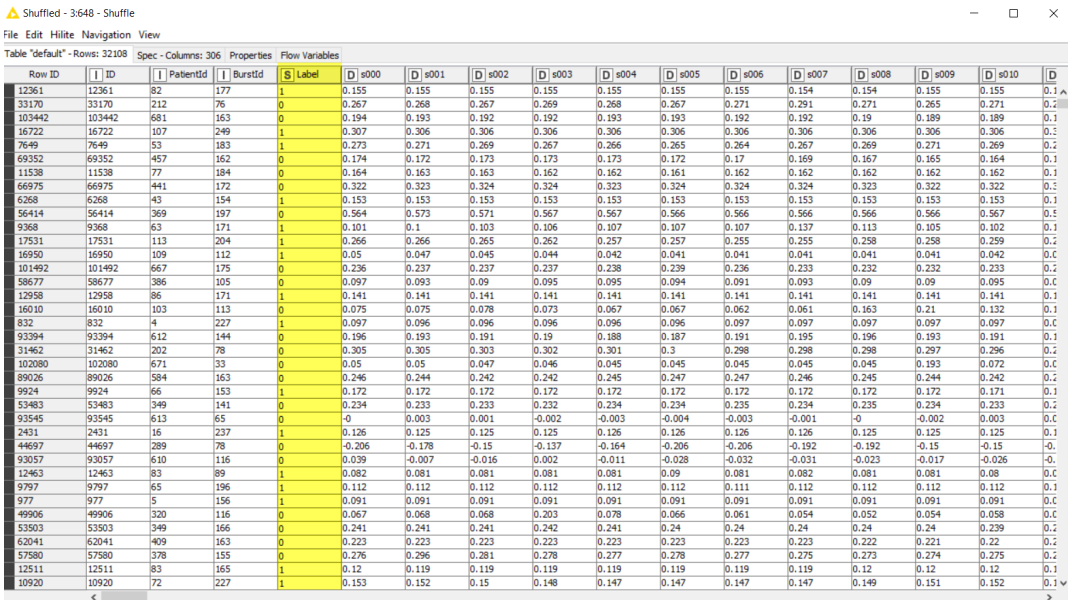


Figure 57 - Extrait du dataset d'entraînement après le mélange avec les labels (en jaune) - Image de l'auteur

5.4.7. Résultats après le mélange des données

Toujours avec le même modèle de 200 itérations, ce réseau de neurones atteint $66\%^{+/-0.6456}$ de prédictions correctes pour détecter les patients sains et $72\%^{+/-0.6456}$ de prédictions correctes pour les

patients malades sur le dataset de test. Son AUC monte à $0.7452^{+/-006456}$, pour une accuracy de $68{,}75\%^{+/-0.6456}$ (figure 58).

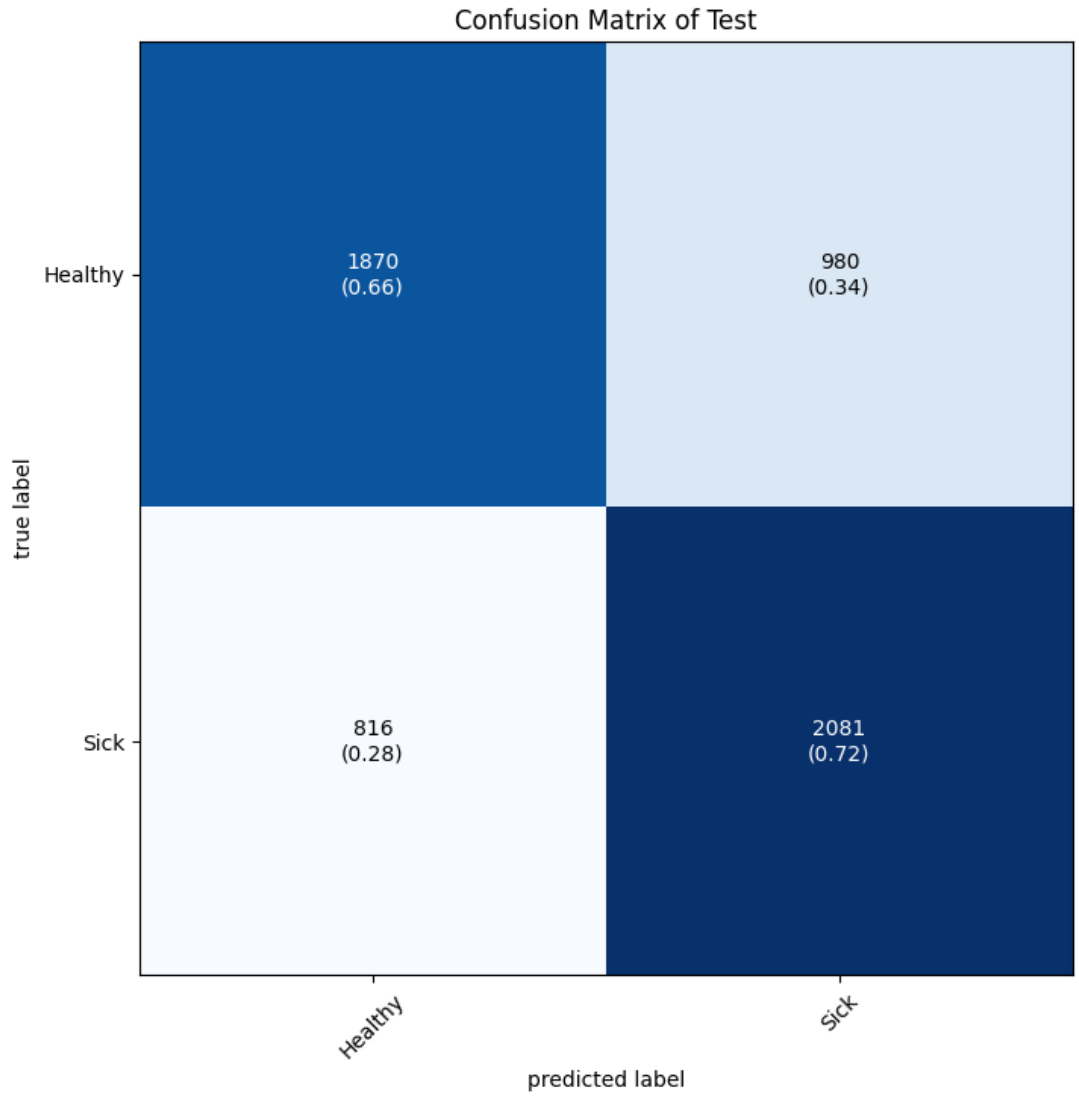


Figure 58 - Matrice de confusion après le mélange des données sur le dataset de test - Image de l'auteur

5.4.8. Activation ReLu

Maintenant que le dataset est équilibré, les configurations du modèle seront passées au peigne fin, en commençant par l'algorithme d'activation ReLu. Pour tester son utilité, l'activation sera retirée du modèle qui sera entraîné 200 fois.

5.4.9. Résultat de l'activation ReLu

Comme le montre la matrice de confusion de la figure 59, l'activation ReLu est essentielle pour le bon fonctionnement du modèle et donc obligatoire pour la suite des expérimentations. En effet, sans fonction d'activation entre chaque couche, le modèle est caduc.

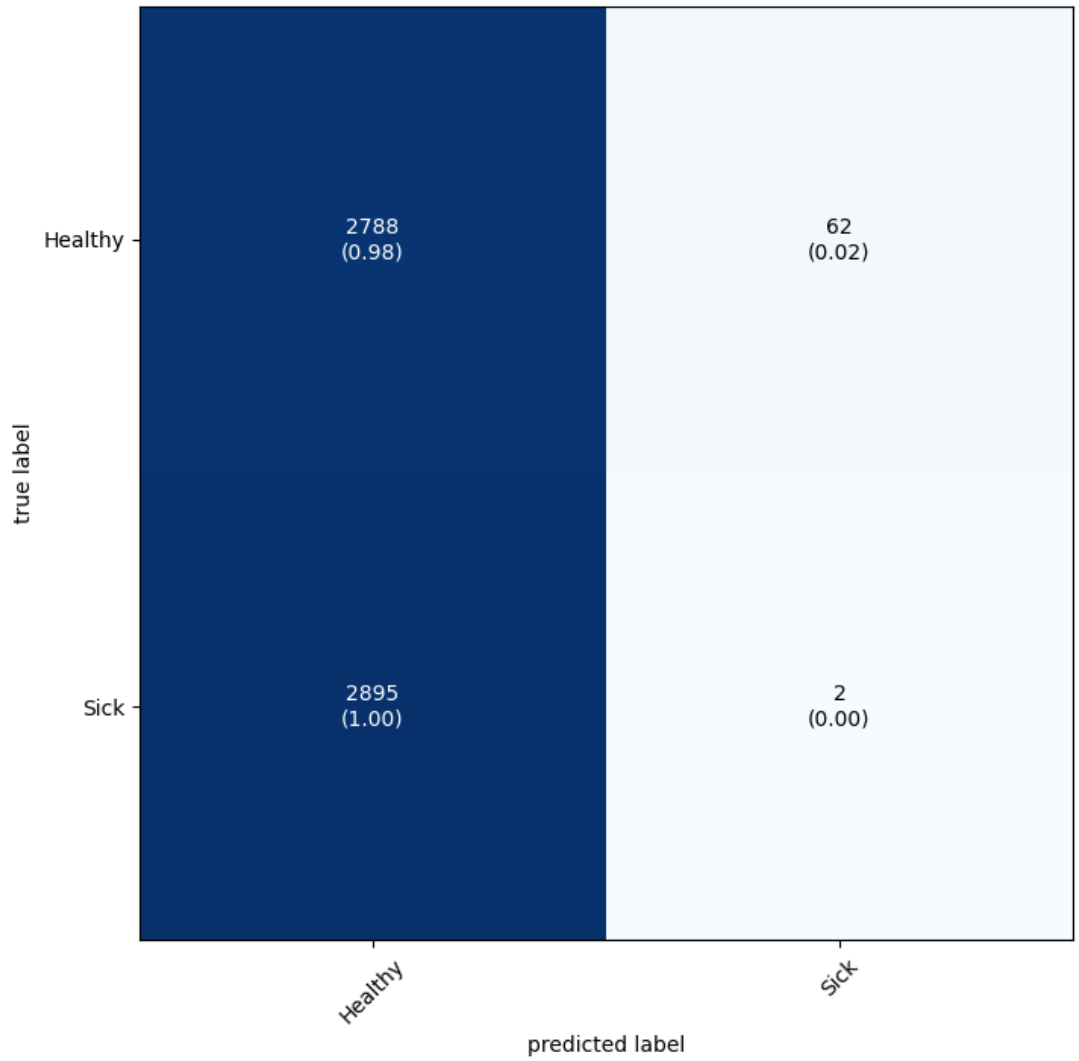


Figure 59 - Matrice de confusion d'un réseau de neurones sans l'activation ReLu - Image de l'auteur

5.4.10. Limites et résultats de l'overfitting

Le modèle est à présent configuré et équilibré. Il reste maintenant à déterminer la limite d'itérations du modèle pour avoir un entraînement optimal d'un point de vue de temps et de qualité, tout en évitant l'overfitting. Pour ce faire, le modèle sera lancé 1000 fois.

La matrice de confusion de la figure 60 démontre les impacts de l'overfitting, avec un modèle incapable de prédire un patient malade.

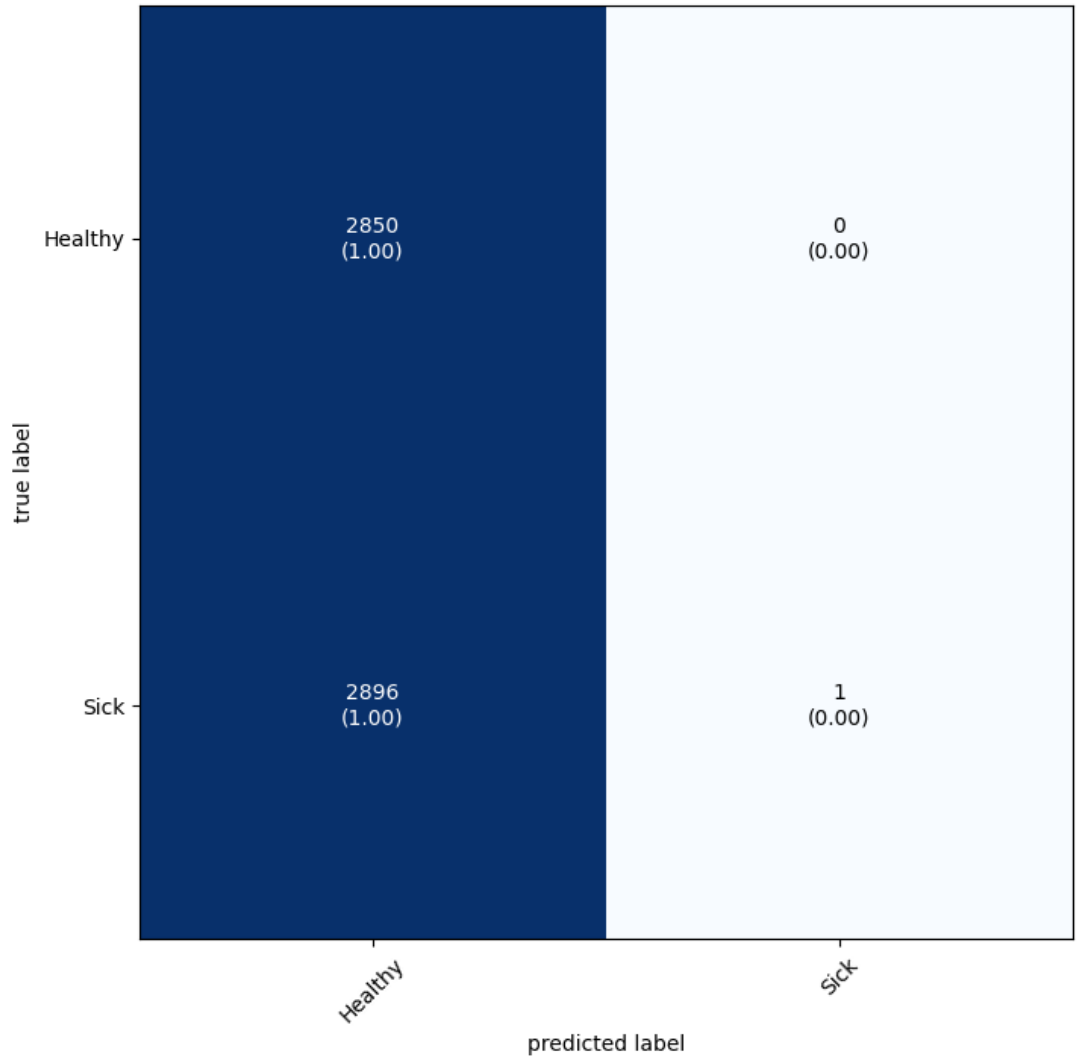


Figure 60 - Matrice de confusion avec un entraînement de 1000 itérations - Image de l'auteur

La métrique de l'AUC à chaque itération, générée graphiquement dans la figure 61 grâce à Keras, dévoile que l'overfitting surgit après environ 800 itérations. Cette unité sera la limite d'itération maximale à ne pas dépasser avec notre dataset.

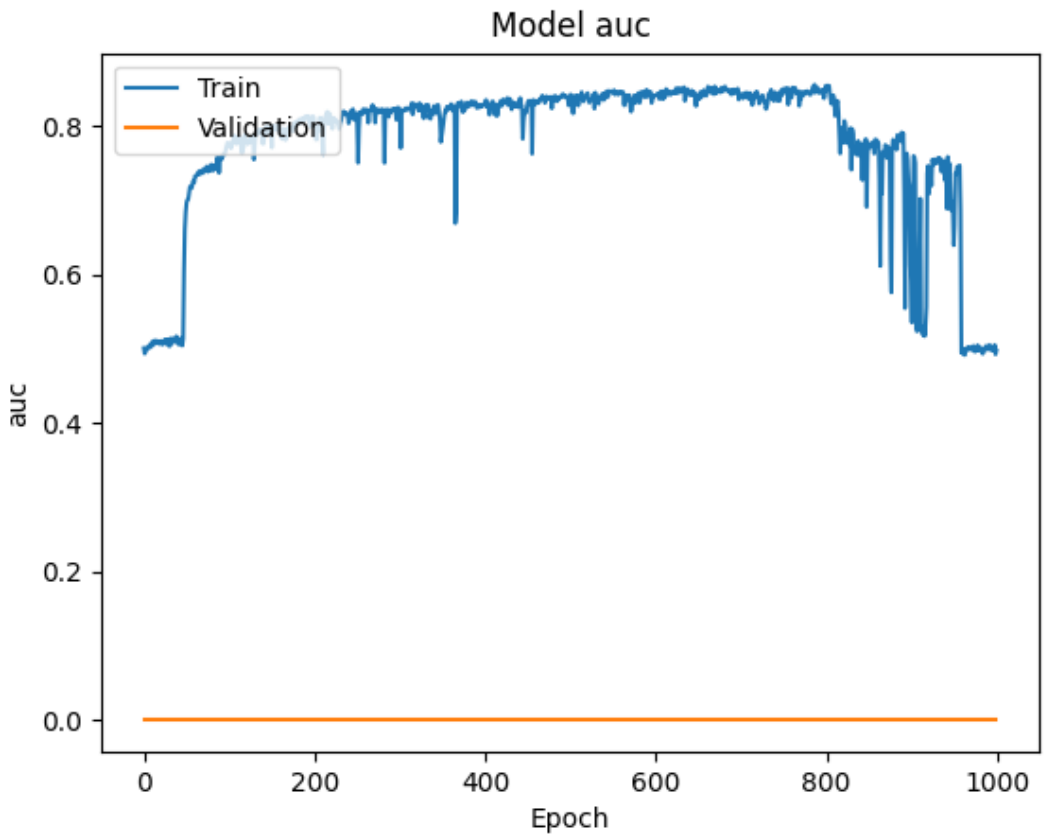


Figure 61 - Évolution de l'AUC d'un modèle entraîné 1000 fois - Image de l'auteur

Les résultats démontrent que le modèle s'améliore radicalement jusqu'à la 150ème itération, pour augmenter progressivement jusqu'à la 800ème itération et finalement, dériver de son cap jusqu'à la 1000ème itération.

200 itérations est une bonne moyenne temps/qualité pour ce modèle car il limite drastiquement l'overfitting, tout en étant entraîné convenablement.

5.4.11. Résultats des expériences sur l'optimisation du réseau multicouches

En conclusion, ce modèle de réseau de neurones multicouches simplifié est optimisé pour nos datasets et notre contexte avec :

1) Des datasets mélangés et équilibrés.

2) Une activation ReLu et sigmoïde.

3) Deux couches cachées de 302 neurones.

4) Un algorithme de perte MSE et un algorithme d'optimisation SGD.

5) Un nombre d'itération proche de 200.

La figure 62 affiche la représentation de ce modèle selon Keras.

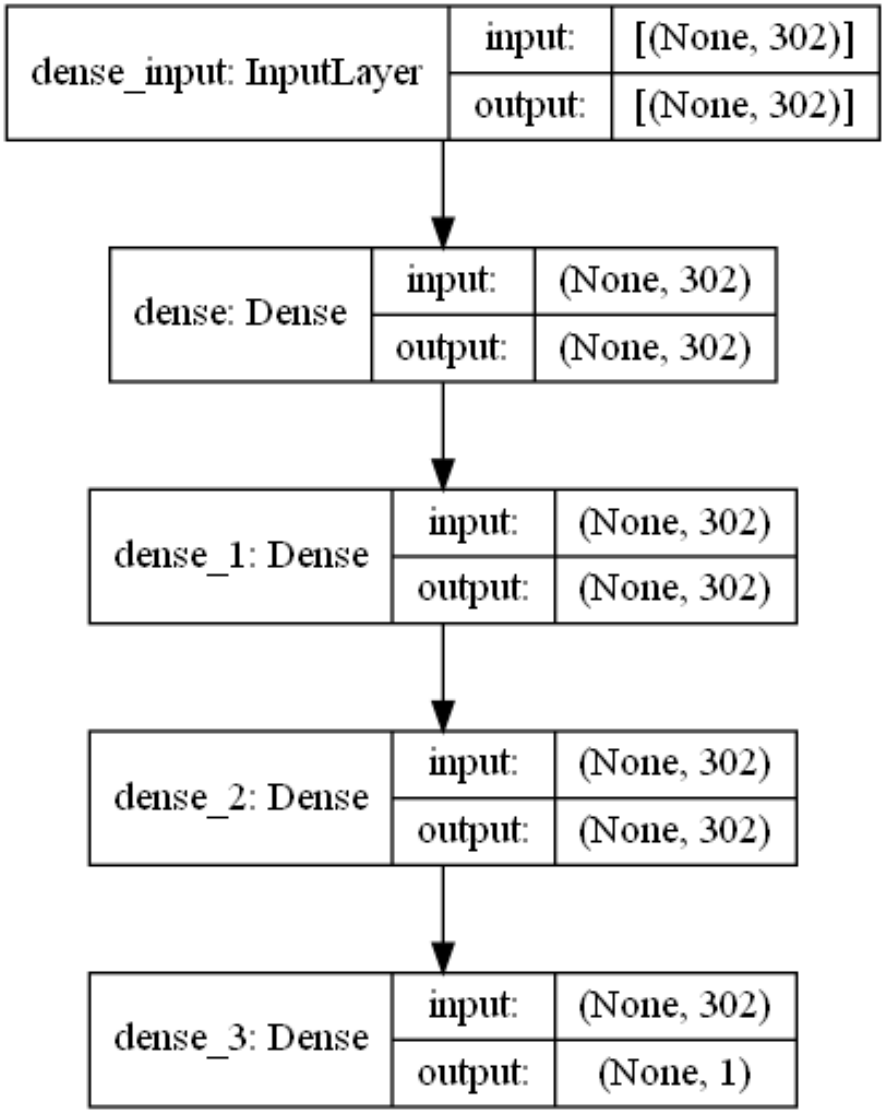


Figure 62 - Représentation du meilleur modèle multicouches - Image de l'auteur

Le tableau 12 et la figure 63 exposent le résultat du modèle avec le dataset de test.

Résultat réseau multicouches - dataset de test *(SE : 0.00646)*					
	Perte	Accuracy	Précision	Recall	AUC
200 itérations	22,62%	68,74%	67,98%	71,83%	$0.7452^{+/-0.00646}$

Tableau 12 - Résultats finaux du réseau multicouches avec le dataset de test

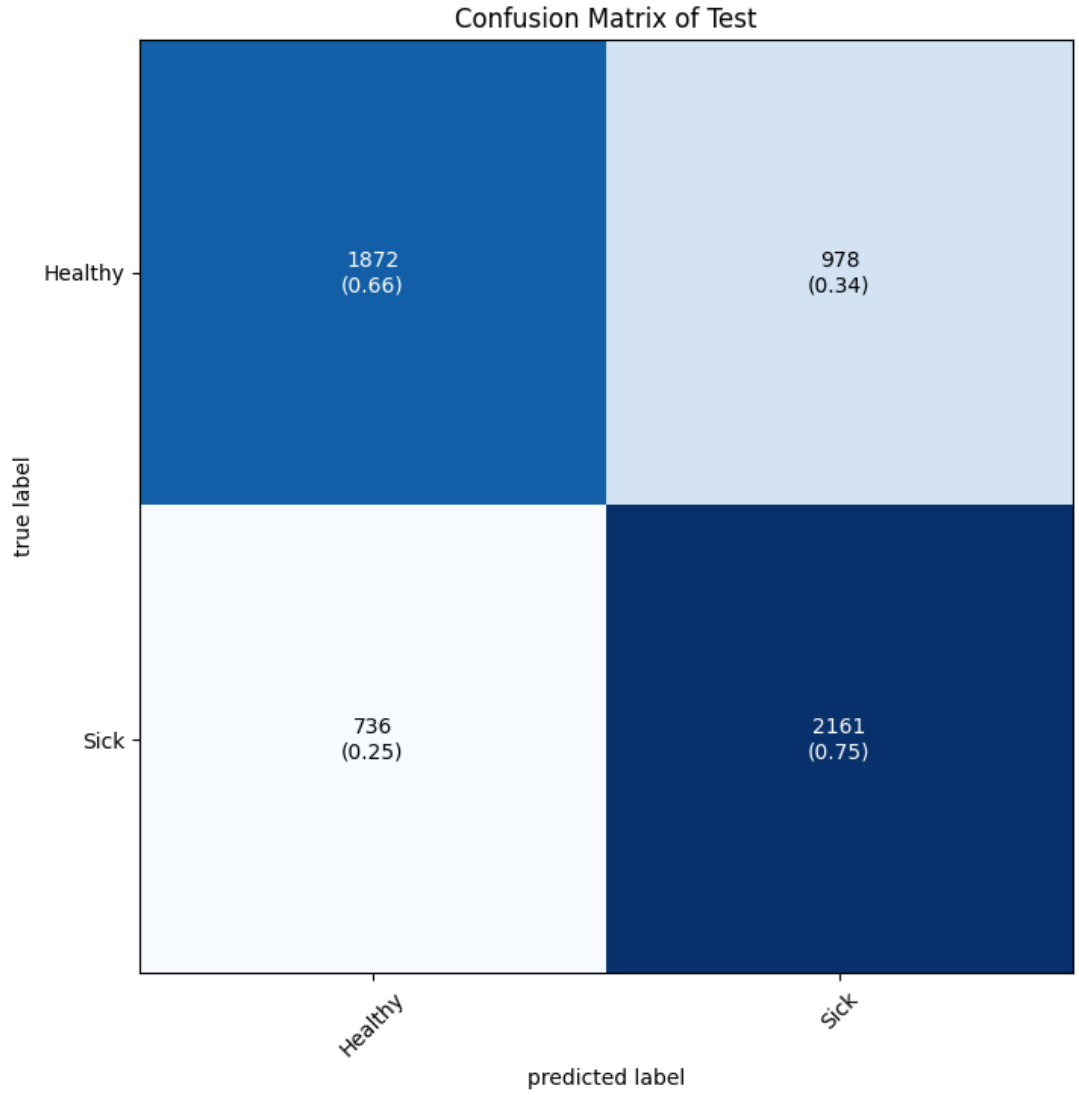


Figure 63 - Résultat du meilleur modèle multicouches avec le dataset de test - Image de l'auteur

En comparaison avec le dataset de validation, les chiffres démontrent que les résultats sont plutôt stables, comme le prouve le tableau 13 et la matrice de confusion de la figure 64.

Résultat réseau multicouches - dataset de validation *(SE : 0.00645)*					
	Perte	Accuracy	Precision	Recall	AUC
200 itérations	21,61%	70,41%	69,96%	72,05%	$0.7401^{+/-0.00645}$

Tableau 13 - Résultats finaux du réseau multicouches avec le dataset de validation

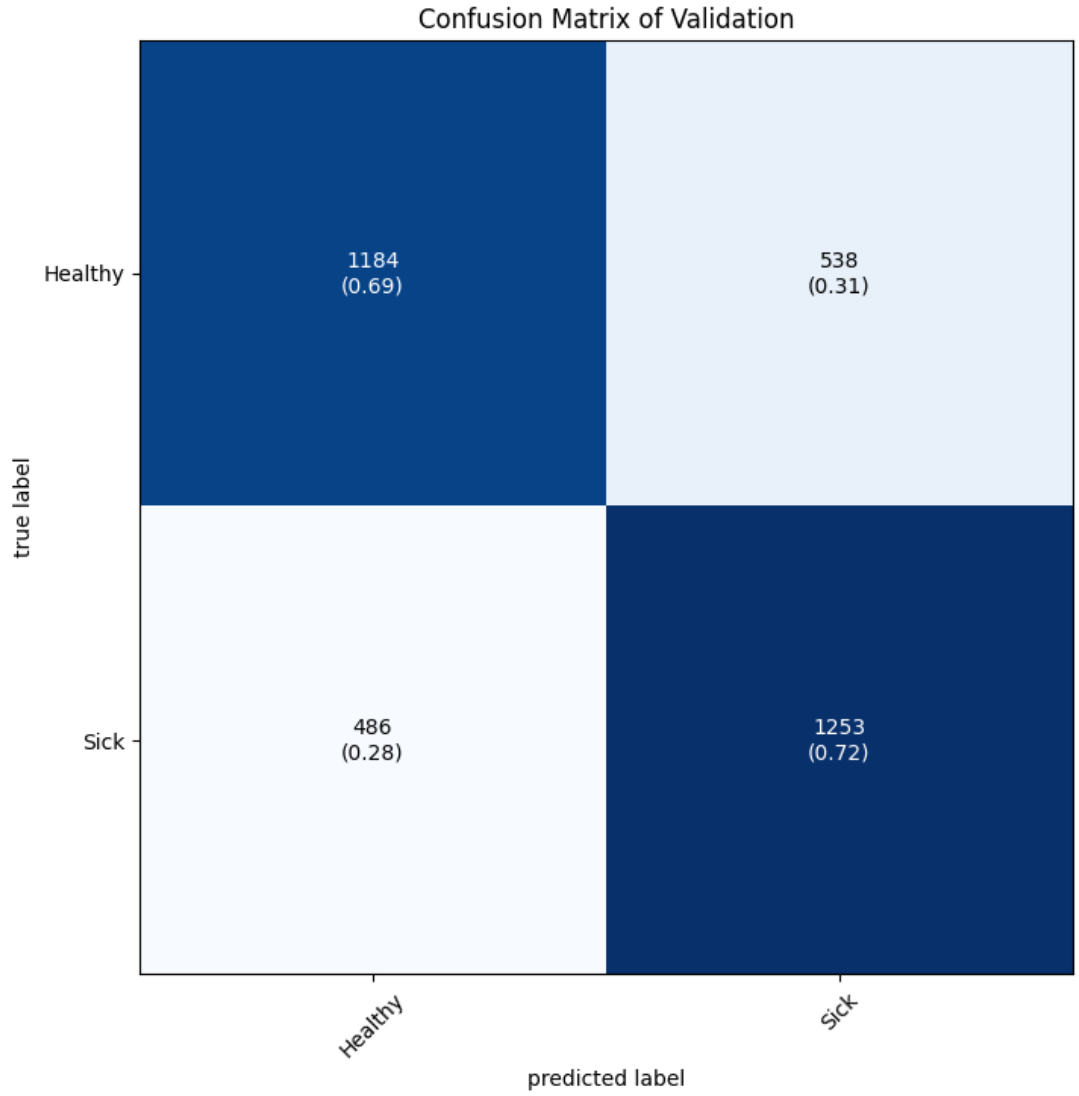


Figure 64 - Résultat du meilleur modèle multicouches avec le dataset de validation - Image de l'auteur

5.4.12. Conclusion des expériences sur le réseau de neurones multicouches

L'objectif de l'expérimentation était de faire une première immersion dans le domaine du deep learning et de prouver si un modèle simple pouvait se montrer à la hauteur d'un modèle de classification traditionnel de machine learning.

Le premier objectif est atteint. Un modèle de réseau de neurones multicouches a été implémenté et optimisé. Il est donc possible d'adapter le dataset de Sensimed avec une technologie de deep learning. Il est néanmoins important de souligner que, même si le temps consacré à construire le réseau était moindre, le temps investi à le paramétrer fut élevé. Par ailleurs, alors que l'entraînement du Gradient Boosted Tree ne prend que 3 minutes, celui d'un réseau de neurones multicouches de 200 itérations dure environ 150 minutes.

Au sujet du deuxième objectif, à ce stade, en comparant les chiffres, un réseau de neurones aussi simple atteint un AUC $0.7451^{+/-0.00646}$ contre $0.755^{+/-0.006}$ pour le meilleur résultat de machine learning traditionnel (Gradient Boosted Tree). Poursuivre sur la voie du deep learning reste une solution viable pour atteindre la meilleure classification possible avec notre dataset.

5.5. Convolutional Neural Networks 1D

Théoriquement, le CNN 1D est le modèle le plus adapté à nos datasets étant donné qu'il est spécifiquement conçu pour la classification de graphes et d'ondes. Cette expérimentation visera à démontrer que le CNN 1D est un type de deep learning plus efficace que le réseau de neurones multicouches. Le modèle sera configuré avec un algorithme d'optimisation SGD, un algorithme de perte MSE et une activation de type ReLu.

5.5.1. Préparation des données et altération

Pour cette expérimentation, nous allons tirer parti de la dernière expérience et intégrer directement un dataset équilibré et mélangé.

Pour intégrer des données à un CNN 1D, il faut transformer nos données d'un tableau en deux dimensions en une liste en trois dimensions. En effet, le CNN 1D perçoit le dataset sous trois formes, le marqueur temporel, le nombre de dimensions et le nombre possible de sorties.

La figure 65 montre tout d'abord la forme initiale du dataset d'entraînement depuis la console, puis sa transformation en 3 dimensions. Nous paramétrerons les valeurs suivantes au modèle : 302 marqueurs temporels (timesteps) correspondant aux 302 colonnes du dataset, une dimension (features) qui se réfère à la mesure de la pression de l'œil et une sortie (ouput) qui correspond à la prédiction « sain » ou « malade ».

```
Data X :(32108, 302)   /  Label Y : (32108, 1)

Transformation.................................
Data X :(32108, 302, 1)/  Label Y : (32108, 1)
```

Figure 65 - Forme originelle puis transformée du dataset d'entraînement - Image de l'auteur

Nous aurions pu avoir plusieurs dimensions, mais notre dataset ne gère que la pression des yeux. L'âge, par exemple, aurait pu être une deuxième dimension.

5.5.2. Modélisation du CNN 1D

Ce modèle, de type séquentiel, sera bâti avec :

1) Deux couches de convolution construites sur 64 filtres, un kernel d'une taille 9, une activation ReLu, un timesteps de 302 et une seule dimension. Ces couches permettront de filtrer les données selon leurs caractéristiques. Elles prendront chaque donnée d'entrée, calculeront leur poids et appliqueront chaque biais. De plus, leurs tailles (filtres, taille, nombre de couches) coïncident avec les expérimentations du dernier chapitre, limitant les risques d'overfitting et d'underfitting.

2) Une fonction MaxPooling1D(pool_size=3) qui sélectionnera les valeurs maximales entre trois éléments d'un filtre et les résumera à une valeur, ce qui permettra de réduire la taille de chaque donnée filtrée.

3) Deux couches de convolution construites sur 10 filtres, un kernel d'une taille 4 et une activation ReLu. Ces couches permettront de filtrer les données selon leurs caractéristiques.

4) Une fonction GlobalAveragePooling1D() qui sélectionnera les valeurs moyennes selon les valeurs du filtre, ce qui permettra de réduire la taille de chaque donnée filtrée. De plus, cette fonction réduira notre modèle à 2 dimensions.

5) Une fonction Dropout(0.5) qui régularisera les données en désactivant aléatoirement la moitié des neurones, ce qui améliorera la généralisation.

6) Deux couches cachées de 100 neurones avec une activation ReLu, reprenant les concepts étudiés plus tôt.

7) Une couche de sortie avec une fonction d'activation de type sigmoïde.

Le code 2 illustre la construction d'un CNN 1D.

```
…
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=9, activation='relu',
input_shape=(timesteps, features)))
model.add(Conv1D(filters=64, kernel_size=9, activation='relu'))
model.add(MaxPooling1D(pool_size=3))
model.add(Conv1D(filters=10, kernel_size=4, activation='relu'))
model.add(Conv1D(filters=10, kernel_size=4, activation='relu'))
model.add(GlobalAveragePooling1D())
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(output, activation='sigmoid'))
…
```

Code 2 - Extrait de la méthode qui implémente le modèle CNN 1D (voir annexe IV)

La figure 66 résume le modèle Keras programmé en python.

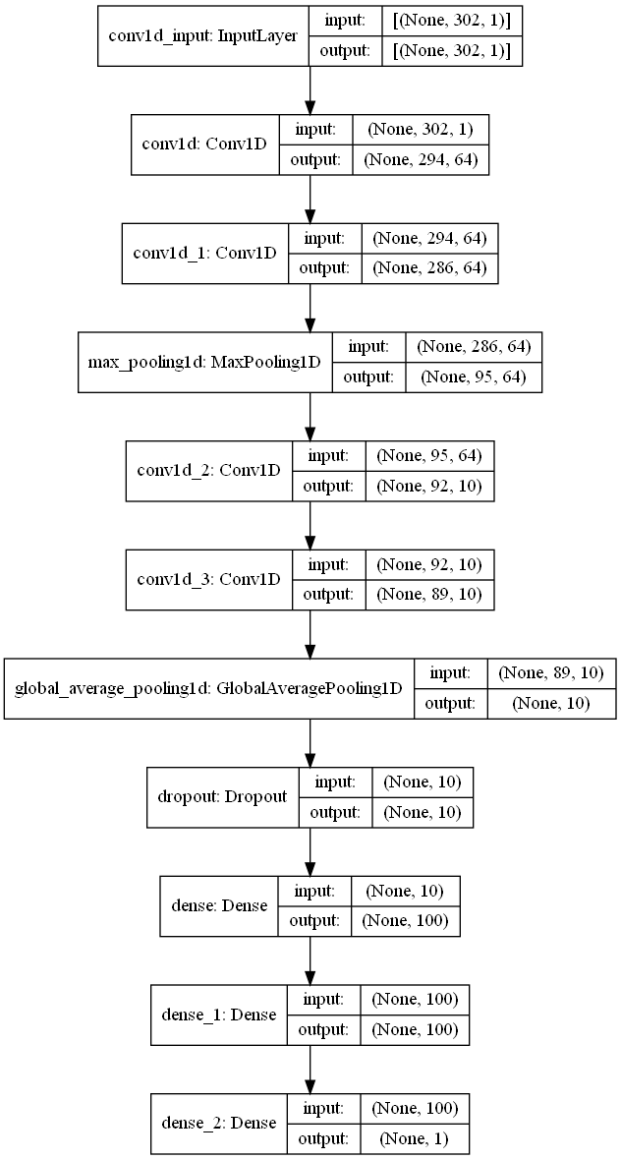


Figure 66 - Représentation graphique du modèle CNN 1D - Image de l'auteur

5.5.3. Limites et résultats de l'overfitting

Le modèle est à présent configuré. Il reste maintenant à déterminer la limite d'itérations du modèle pour avoir un entraînement optimal d'un point de vue de temps et de qualité, tout en évitant l'overfitting. Pour ce faire, le modèle sera lancé 100 fois. En reproduisant la même démarche que pour la dernière expérimentation, la limite d'itérations pour notre modèle de CNN 1D se situe à environ 60 itérations.

5.5.4. Résultats du CNN 1D

Les résultats sur le dataset de test pour un CNN 1D qui a été entraîné sur 60 itérations sont résumés dans le tableau 14.

CNN 1D - dataset de test *(SE : 0.00646)*					
	Perte	Accuracy	Precision	Recall	AUC
60 itérations	20,62%	70,54%	69,17%	74,97%	$0.7371^{+/- 0.00646}$

Tableau 14 - Résultats détaillés du CNN 1D avec le dataset de test

La matrice de confusion de la figure 67 résume les résultats du dataset de test.

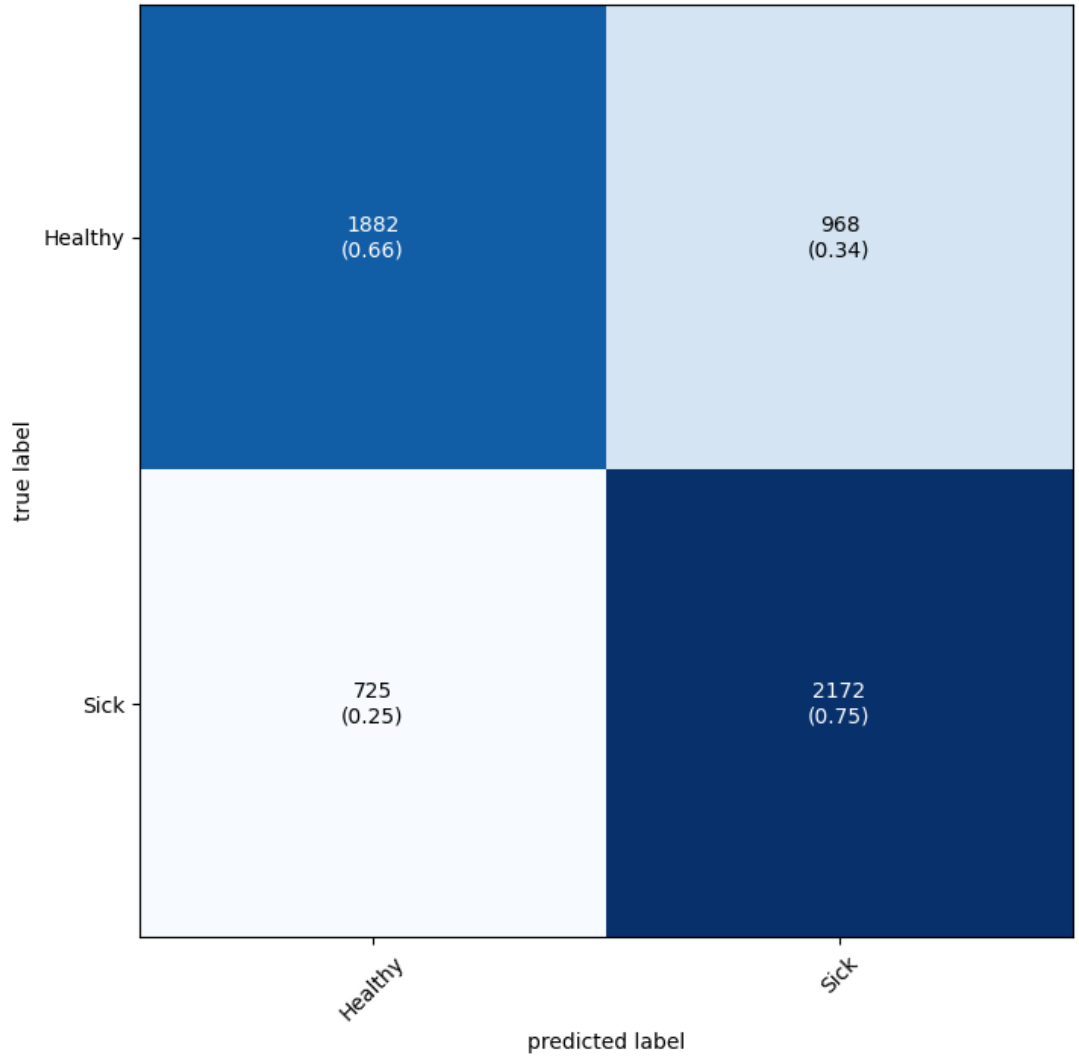


Figure 67 - Matrice de confusion du CNN 1D avec le dataset de test - Image de l'auteur

Les résultats de l'évaluation du dataset de validation, affichés dans la tableau 15, divergent quelque peu, surtout sur la métrique du recall, mais rien d'alarmant. Cette différence est due à l'écart de prédiction entre les Bursts sains (prédits à $82\%^{+/-0.646}$) et les Bursts malades (seulement prédits à $60\%^{+/-0.646}$).

CNNs 1D - dataset de validation *(SE : 0.00646)*					
	Perte	Accuracy	Precision	Recall	AUC
60 itérations	19,75%	71,25%	77,47%	60,32%	$0.7623^{+/-\ 0.00646}$

Tableau 15 - Résultats détaillés du CNN 1D avec le dataset de validation

La matrice de confusion de la figure 68 résume les résultats du dataset de validation.

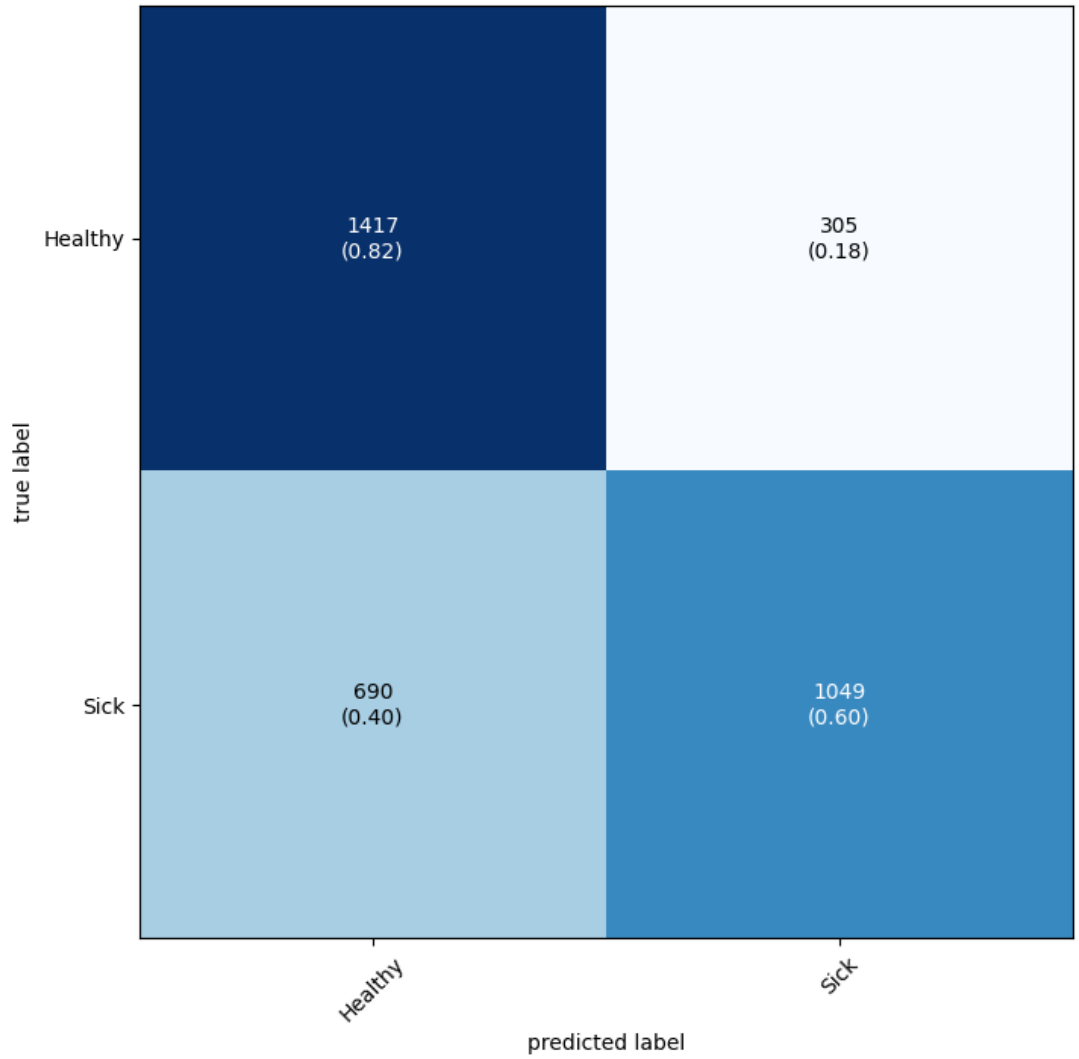


Figure 68 - Matrice de confusion du CNN 1D avec le dataset de validation - Image de l'auteur

5.5.5. Conclusion des expériences sur le CNN 1D

L'objectif de l'expérimentation était d'implémenter un modèle CNN 1D et démontrer qu'il pouvait être à la hauteur d'un modèle de classification traditionnel de machine learning.

Le premier objectif est atteint. Un modèle de Convolutional Neural Network 1D a été implémenté. Le CNN 1D est donc capable de détecter des caractéristiques dans les données transformées du dataset de Sensimed. Même si le réseau dévoile de bon résultats, il reste simple. Le modèle devra donc être optimisé pour atteindre de meilleurs résultats.

Comme ce modèle possède plus de couches et de transformations, le temps d'entraînement par itération du CNN 1D est plus long que pour le réseau de neurones multicouches. Cependant, n'ayant que 60 itérations d'environ 105 secondes à la place de 200 itérations d'environ 45 secondes, le CNN 1D s'entraîne en moyenne en 105 minutes, alors que le réseau multicouches dure 150 minutes. Toutefois, son temps d'entraînement reste environ 17 fois plus long que celui du Gradient Boosted Tree, une technologie traditionnelle de machine learning.

Au sujet du deuxième objectif, à ce stade, en comparant les chiffres, le CNN 1D atteint un AUC $0.7371^{+/-0.00646}$ contre $0.7451^{+/-0.00646}$ pour le réseau multicouches optimisé et $0.755^{+/-0.006}$ pour le Gradient Boosted Tree. Maintenant que le modèle fonctionne, il est temps de l'optimiser.

5.6. Limitation de l'impact du bruit

Nous avons à présent des résultats satisfaisants avec nos différents réseaux de neurones. Nous pouvons néanmoins les optimiser. Changer les paramètres en tâtonnant prendrait une éternité, c'est pourquoi des techniques d'optimisation comme la suppression de bruit existent. Dans cette expérimentation, nous allons procéder à une détection des seuls, éliminer une partie du bruit, en ajouter et analyser ces résultats pour déterminer s'il est possible de limiter l'impact du bruit sur les prédictions des modèles implémentés dans ce projet.

5.6.1. Détection des seuils

La première étape pour limiter le bruit est de détecter si des données sont néfastes pour la prédiction. A ce stade, le seuil de base est délimité à 0.5. Les résultats de prédiction de 0.5 et moins sont alors considérés comme sains et ceux supérieurs à 0.5 comme malades. Avec ce constat, deux scénarios sont alors envisageables.

1) Les prédictions proches du seuil du modèle sont néfastes. Les résultats de prédiction comme 0.48 ou 0.52 sont donc considérées comme du bruit qui brouille la prédiction.

2) Les prédictions proches des extrémités sont néfastes. Celles avec des résultats proche de 0 ou de 1, comme 0.15 ou 0.85 sont alors considérées comme du bruit qui brouille la prédiction.

La première étape consiste alors à détecter si le bruit est issu des valeurs proches du seuil ou des extrémités.

5.6.2. Élimination du bruit

Du moment que les seuils minimum et maximum intérieur ou extérieur ont été fixés, il faut traiter le bruit avec un nouveau dataset inconnu par le modèle. A cause du nombre limité de nos données, nous avons utilisé le dataset de validation pour déterminer les seuils (voir chapitre 5.6.1. Détection de seuils) et avons retiré le bruit propre au modèle dans le dataset de test, qui est toujours inconnu pour le réseau de neurones. Voici donc le nouveau processus :

1) Entraînement du modèle avec le dataset d'entraînement.

2) Détection des seuils avec le dataset disponible que nous avions réservé pour la validation.

3) Suppression du bruit dans le dataset de test et évaluation de ce nouveau dataset.

5.6.3. Ajout du bruit

L'ajout de bruit peut sembler négatif, mais des outils comme le Gaussian Noise permettent d'en ajouter lors de l'entraînement du modèle. Ce bruit permettra de combattre la généralisation des modèles et d'enlever les biais dus à la normalisation.

Pour ce faire, nous allons simplement insérer une ligne dans l'architecture du modèle CNN 1D. Comme l'illustre l'extrait du code 3, nous paramétrons la valeur de 0.1 au Gaussian Noise pour ajouter modérément du bruit positif.

```
…
model.add(GaussianNoise(0.1))
…
```

Code 3 - Extrait du modèle final (voir annexe IV)

5.6.4. Résultat de la gestion de bruit

Les résultats obtenus après la gestion du bruit chamboulent la comparaison du CNN 1D au traditionnel Gradient Boosted Tree. Comme le démontre le tableau 16 et la figure 69, les résultats obtenus sont pour la première fois parvenus au-dessus d'un AUC de 0.8.

CNN 1D - Après l'élimination du bruit - dataset de test *(SE : 0.00645)*					
	Perte	Accuracy	Precision	Recall	AUC^{SE}
60 itérations	18,79%	76,70%	82,80%	70,15%	$0.8109^{+/-0.00645}$

Tableau 16 - Résultat du CNN 1D après la gestion du bruit sur le dataset de test

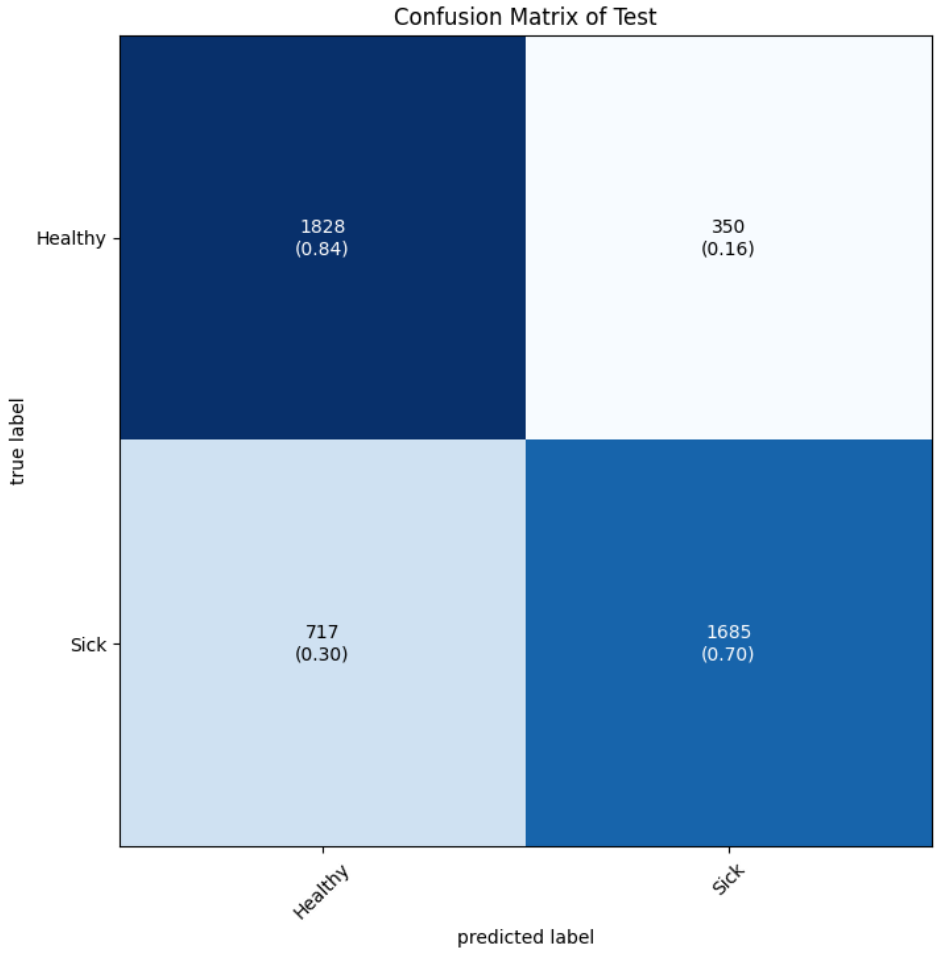


Figure 69 - Matrice de confusion d'un CNN 1D après la gestion du bruit - Image de l'auteur

5.6.5. Conclusion de la gestion de bruit

A la suite des présentations des résultats, nous avons programmé un algorithme servant à cartographier les Bursts considérés comme du bruit. Ce fichier texte permettra au client d'analyser ces Bursts sur l'échelle temporelle afin de vérifier s'ils correspondent à des moments précis de la journée et à des activités clés comme la respiration, qui pourraient influer sur la pression de l'œil.

Grâce à ces informations récoltées lors du processus d'évaluation du CNN 1D, nous sommes parvenus à extraire des caractéristiques propres aux données. La figure 70 est un extrait des informations issues de la gestion du bruit.

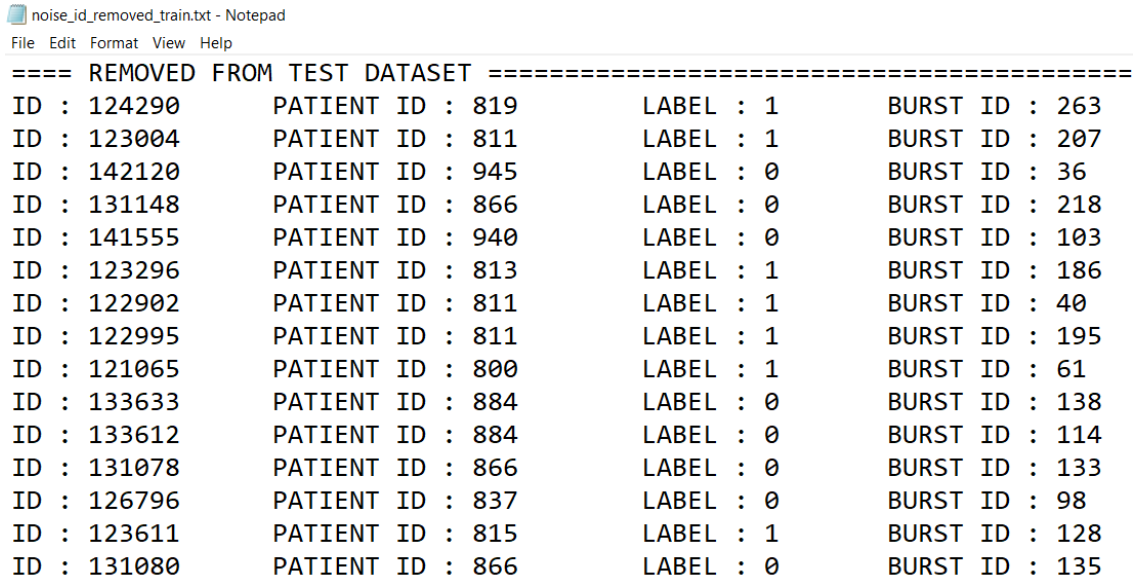

noise_id_removed_train.txt - Notepad
File Edit Format View Help
==== REMOVED FROM TEST DATASET ==
ID : 124290 PATIENT ID : 819 LABEL : 1 BURST ID : 263
ID : 123004 PATIENT ID : 811 LABEL : 1 BURST ID : 207
ID : 142120 PATIENT ID : 945 LABEL : 0 BURST ID : 36
ID : 131148 PATIENT ID : 866 LABEL : 0 BURST ID : 218
ID : 141555 PATIENT ID : 940 LABEL : 0 BURST ID : 103
ID : 123296 PATIENT ID : 813 LABEL : 1 BURST ID : 186
ID : 122902 PATIENT ID : 811 LABEL : 1 BURST ID : 40
ID : 122995 PATIENT ID : 811 LABEL : 1 BURST ID : 195
ID : 121065 PATIENT ID : 800 LABEL : 1 BURST ID : 61
ID : 133633 PATIENT ID : 884 LABEL : 0 BURST ID : 138
ID : 133612 PATIENT ID : 884 LABEL : 0 BURST ID : 114
ID : 131078 PATIENT ID : 866 LABEL : 0 BURST ID : 133
ID : 126796 PATIENT ID : 837 LABEL : 0 BURST ID : 98
ID : 123611 PATIENT ID : 815 LABEL : 1 BURST ID : 128
ID : 131080 PATIENT ID : 866 LABEL : 0 BURST ID : 135

Figure 70 - Extrait de la liste des Bursts supprimés lors de la gestion du bruit - Image de l'auteur

La prochaine étape consistera à analyser les véritables valeurs de ces données. Étant anonymisées et normalisées pour des raisons de confidentialité, ces données seront traitées directement par Sensimed après la conclusion de ce projet.

5.7. Cross-Validation et Itérations

Nous avons à présent des outils qui nous permettent de modéliser un réseau de neurones performant. Il faut néanmoins s'assurer que ses résultats sont corrects. Dans ce but, nous allons développer une méthode qui nous permettra de faire de la cross-validation et de lancer plusieurs itérations d'un modèle.

5.7.1. Cross-Validation

A la suite de l'expérimentation sur les bruits, nous sacrifions un dataset pour détecter les seuils du bruit du modèle. Pour conserver un dataset de validation, nous allons utiliser la cross-validation pour nous offrir un filet de sécurité supplémentaire. Voici la procédure que nous mettrons en place :

Découpage aléatoire des données du dataset d'entraînement de base en deux datasets. Le premier sera le dataset d'entraînement de la cross-validation (~90% des données) et le deuxième sera le dataset de test de la cross-validation (~10% des données). Il faudra néanmoins tenir compte des conditions suivantes :

a. La première nous contraint à ne pas mélanger les Bursts d'un patient dans des datasets différents, en d'autres termes, de garder des patients entiers dans chaque dataset.

b. Le nombre de patients sains et malades devra être équivalant dans les deux datasets générés dynamiquement et aléatoirement.

Pour réaliser cette tâche, nous copierons puis mélangerons le dataset qui contient les patients et tirerons les « X » premiers patients, tout en vérifiant qu'ils soient sains ou malades. Une fois les patients sélectionnés, nous ajouterons leurs Bursts dans le dataset de test de cross-validation et les supprimerons du dataset d'entraînement de cross-validation. La figure 71 résume graphiquement ce procédé.

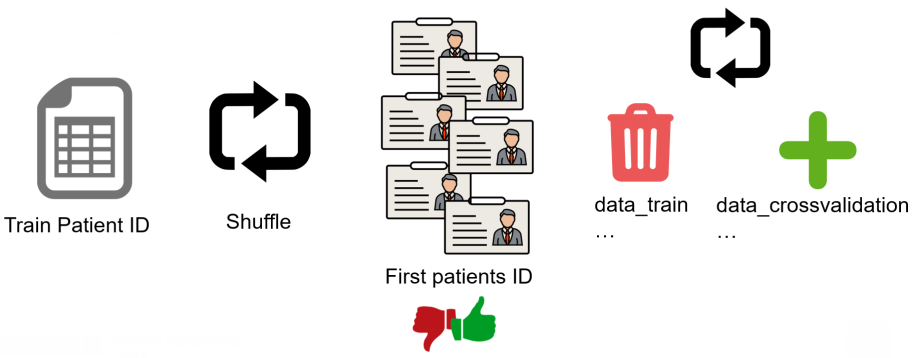


Figure 71 - Résumé de la procédure de tri des patients - Image de l'auteur

Une fois les nouveaux datasets générés, ils seront utilisés comme dataset d'entraînement et de test pour nos modèles, économisant le dataset de test de base, tout en utilisant encore le dataset de validation pour détecter les seuils du bruit.

5.7.2. Itération

Afin de pousser à ses limites nos modèles, nous allons créer une simple boucle qui permettra à l'utilisateur de lancer autant de fois qu'il le désire la création et l'entraînement d'un modèle.

5.7.3. Résultats de la cross-validation et de l'itération

Pour vérifier nos résultats, nous avons lancé 10 itérations d'un modèle CNN 1D entraîné 60 fois avec de la cross-validation. La figure 72 résume les résultats moyens des 10 entraînements des modèles qui contiennent de la cross-validation.

dataset de validation

gestion du bruit

dataset d'entraînement de base (~90%)

Moyenne des résultats du CNN 1D avec la cross-validation							
	Bursts	Perte^{SE}	AUC^{SE}	TP True Positives	FP False Positives	TN True Negatives	FN False Negatives
Entraînement	28238	0.2030$^{?}$	0.7309$^{?}$	4331	1443	5575	2769
Test	3353	18,36%$^{+/-0.804}$	0.7691$^{+/-0.00804}$	1084	302	1084	502
Validation	5162	20,51%$^{+/-0.689}$	0.7477$^{+/-0.00689}$	2107	827	1572	666

dataset de test de base

dataset d'entraînement de base (~10%)

Figure 72 - Tableau des résultats de la cross-validation et de l'itération - Image de l'auteur

Les résultats moyens sont très positifs, passant d'un AUC minimum de $0.71^{+/-0.00921}$ à un AUC maximum de $0.81^{+/-0.00892}$. La cross-validation pioche aléatoirement des données dans le dataset d'entraînement, ce qui implique que le modèle a moins de risques de généralisation.

5.8. Classification par patient

Jusqu'à présent, à la demande du client, nous avons classifié chaque Burst comme étant sain ou malade. Le but de cette expérimentation est de déterminer une moyenne à partir de l'ensemble des classifications des Bursts d'un patient. Cette moyenne indiquera si ce patient est réellement malade ou non. Le tableau 17 résume l'expérimentation.

Description	Exemple
Taux déterminant	50%
Label du patient	Malade
Nombre de Bursts du patient classés « malade »	90
Nombre de Bursts du patient	100
Calcul	50% < 90%
Résultat	Le patient est malade

Tableau 17 - Résumé du processus pour une classification par patient

La figure 73 expose le résultat du processus avec 2 Bursts malades et 1 Burst sain.

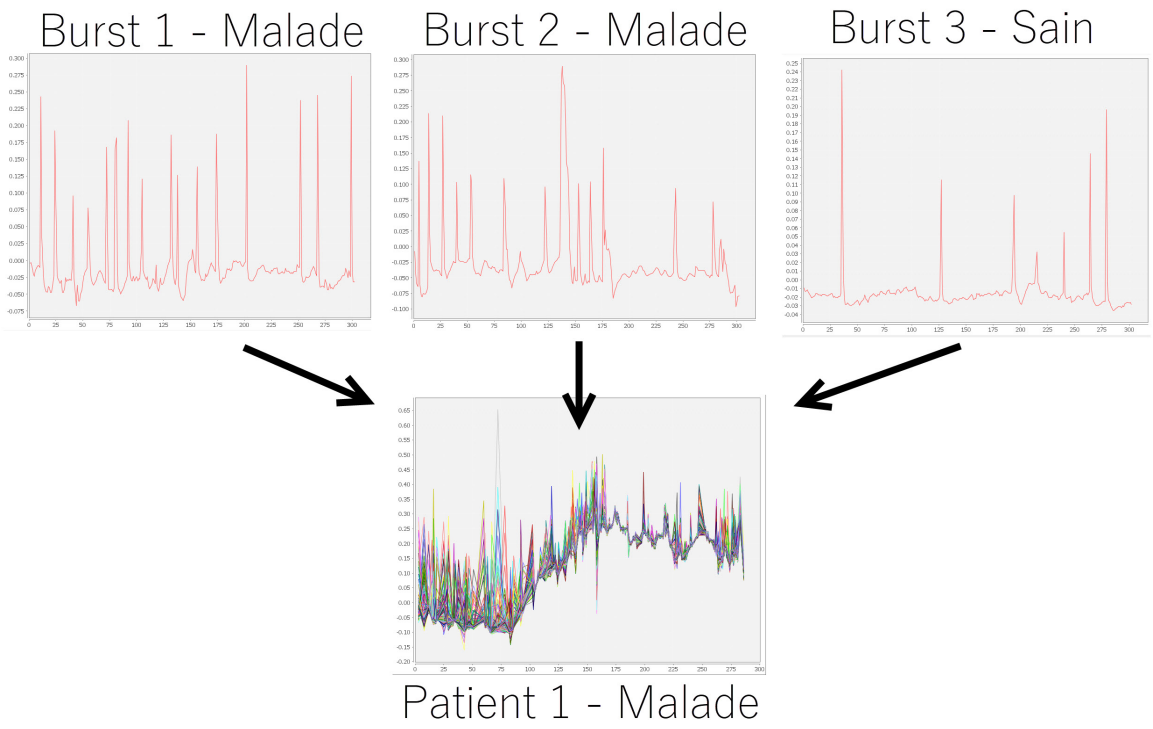


Figure 73 - Résumé de la classification par patient - Image de l'auteur

5.8.1. Préparation des données

Les datasets étant exclusivement consacrés aux données et aux labels, ils n'y a aucune information sur le patient. Pour récupérer cette information, nous allons utiliser KNIME pour créer un fichier CSV contenant tous les patients dans l'ordre des Bursts des datasets de données et de labels. La figure 74 résume le processus de préparation des données, qui a été effectué depuis la plateforme KNIME.

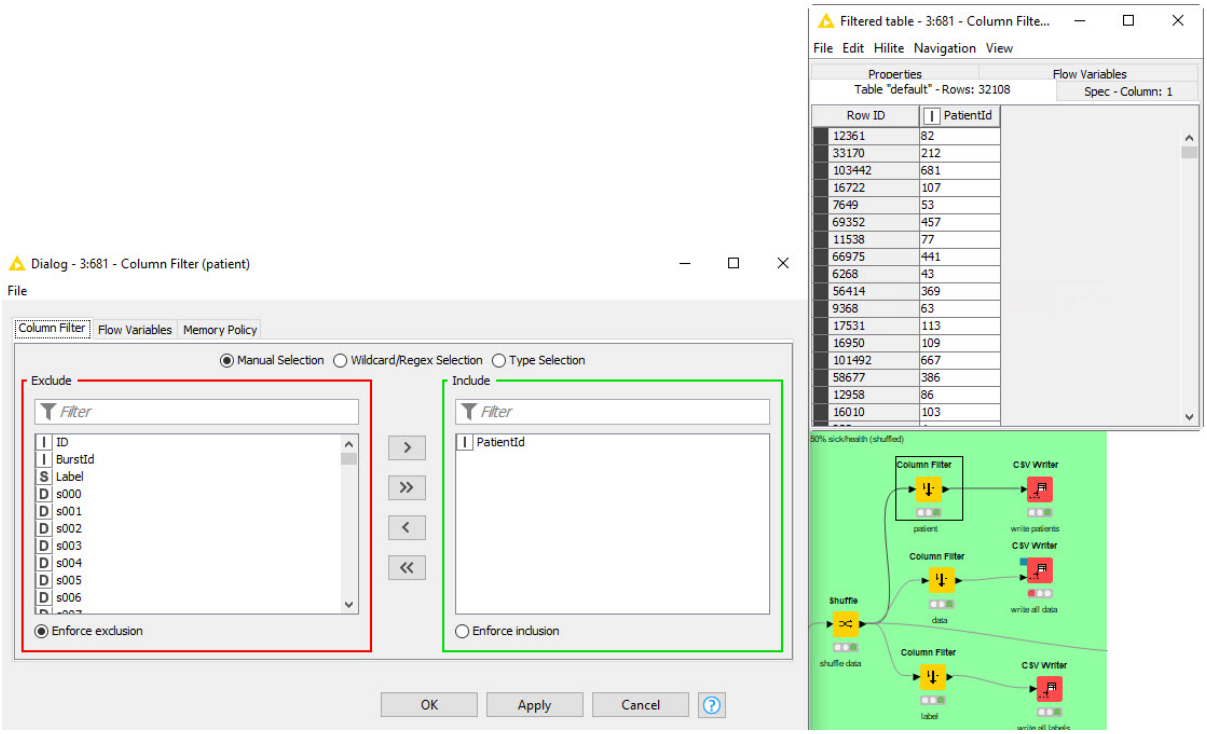


Figure 74 - Isolation des patients dans KNIME - Image de l'auteur

Un tableau de résumé sera ensuite créé en python pour pourvoir générer les métriques qui nous intéressent. La tableau 18 résume la structure et les valeurs de ce tableau python, x équivalant à un des multiples patients.

Colonne du tableau	Valeur
prediction_by_patient[x][0]	ID du patient
prediction_by_patient[x][1]	Nombre total de Bursts par patient
prediction_by_patient[x][2]	Somme de tous les Bursts prédits comme malade
prediction_by_patient[x][3]	Résultat moyen par patient
prediction_by_patient[x][4]	Label prédit par la moyenne des Bursts
prediction_by_patient[x][5]	Véritable label du patient

Tableau 18 - Détail du tableau pour la gestion de la classification par patient

5.8.2. Résultats de la classification par patient

Le tableau 19 résulte de la classification qui a été effectué sur le dataset de test avec le meilleur modèle de CNN 1D.

Classification par patient, dataset de test					
	Total	Prédit correctement	Prédit non-correctement	% Prédit correctement	% Prédit non-correctement
Par Bursts	4580	3513	1067	76,7%$^{+/-0.64}$	23,3%$^{+/-0.64}$
Par Patients	37	30	7	81,08%$^{+/-0.64}$	18,91%$^{+/-0.64}$

Tableau 19 - Résultats détaillés de la classification par patient - dataset de test

5.8.3. Conclusion de la classification par patient

Les résultats sur la classification par patient sont positifs, ayant de meilleurs scores par patient que par Burst. La nuance est néanmoins moindre la plupart du temps. Cette observation peut s'expliquer par le fait qu'un patient malade possède généralement un nombre significatif de Bursts malades.

Une autre expérience pourrait être d'altérer le seuil de la balance qui est actuellement à 50% pour orienter la décision du modèle et donc des résultats.

L'expérimentation étant concluante, les résultats de la classification des Bursts et des patients seront à présent toujours accessibles pour le client.

5.9. Déploiement, code et environnement

Le client désire pouvoir directement utiliser les méthodes implémentées lors de ce projet depuis ses propres scripts python. Pour ce faire, deux outils seront mis à disposition. Le code et toutes ses fonctionnalités seront altérés pour fonctionner de la même manière qu'une librairie, ce qui permettra au client d'appeler à sa guise les méthodes développées lors de ce projet. De plus, un environnement Miniconda sera mis à disposition pour pouvoir lancer et utiliser les méthodes du projet depuis n'importe quel poste, même avec un environnement vierge.

5.9.1. Code

Lors de l'ensemble de ce projet, 23 méthodes ont été implémentées à la main pour gérer l'ensemble des besoins des modèles, partant de la transformation des datasets pour arriver à la construction des modèles, en passant par le calcul de métriques. En plus de ce lot, trois méthodes ont aussi été développées pour répondre à tous les besoins du client. Ces méthodes permettent d'entraîner un modèle, tester un modèle existant et prédire des données non-labellisées. Un fichier de lancement sera mis à disposition du client, mettant en avant chaque utilisation possible de nos méthodes (voir annexe I).

L'intégralité du code est disponible sur la plateforme GitHub[10]. Les datasets ont néanmoins été retirés pour des questions de confidentialité.

5.9.2. Miniconda

Comme le cite la documentation officielle, Miniconda est un installateur minimal gratuit pour Conda (Conda, 2017). Miniconda va permettre de créer un environnement totalement hermétique sur notre ordinateur, où nous pourrons installer toutes les dépendances d'un projet sans aucun problème de sécurité. De plus, il sera possible d'exporter cet environnement avec toutes ses dépendances (voir Annexe VI), ce qui permettra au client de ne pas devoir les installer lui-même. En effet, Sensimed importera l'environnement (voir Annexe VII) et pourra directement lancer le projet sans faire aucune autre manipulation.

[10] Lien du projet GitHub : https://github.com/qnater/AdvancedDetectionEyeIllness

5.10. Résumé des expérimentations

En guise de synthèse, nous allons résumer les différents éléments implémentés sur notre meilleur modèle à la suite des expérimentations.

5.10.1. Entraînement d'un modèle

Le figure 75 résume la construction et l'entraînement d'un modèle en huit étapes clés, le tout durant environ 2 heures sur un ordinateur portable basique.

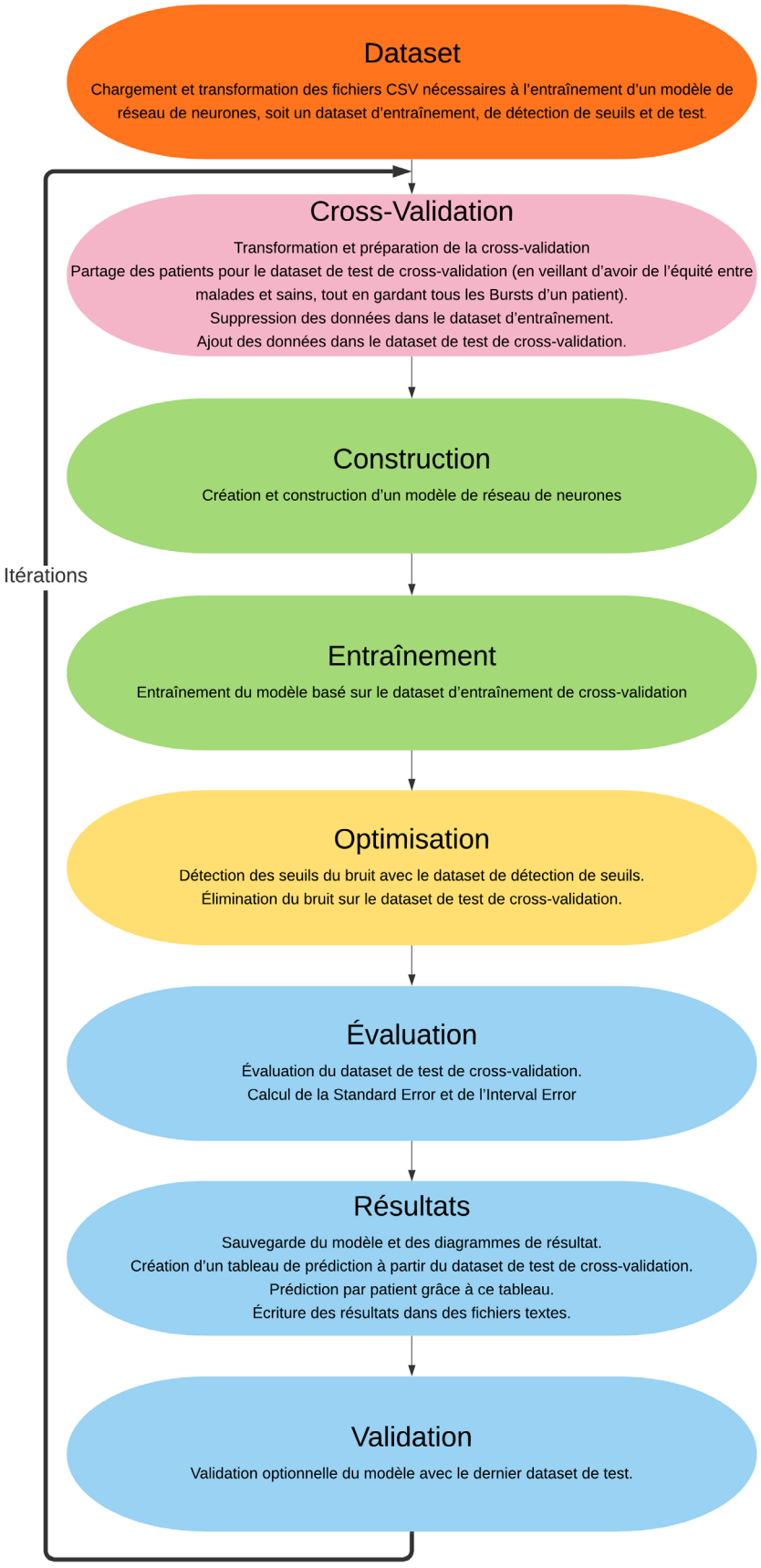


Figure 75 - Processus d'entraînement d'un modèle - Image de l'auteur

5.10.2. Test d'un modèle

Nous pouvons résumer le test d'un modèle en cinq étapes clés, le tout durant un peu moins de cinq minutes sur un ordinateur portable basique. La figure 76 expose chaque étape du processus pour réaliser le test d'un modèle.

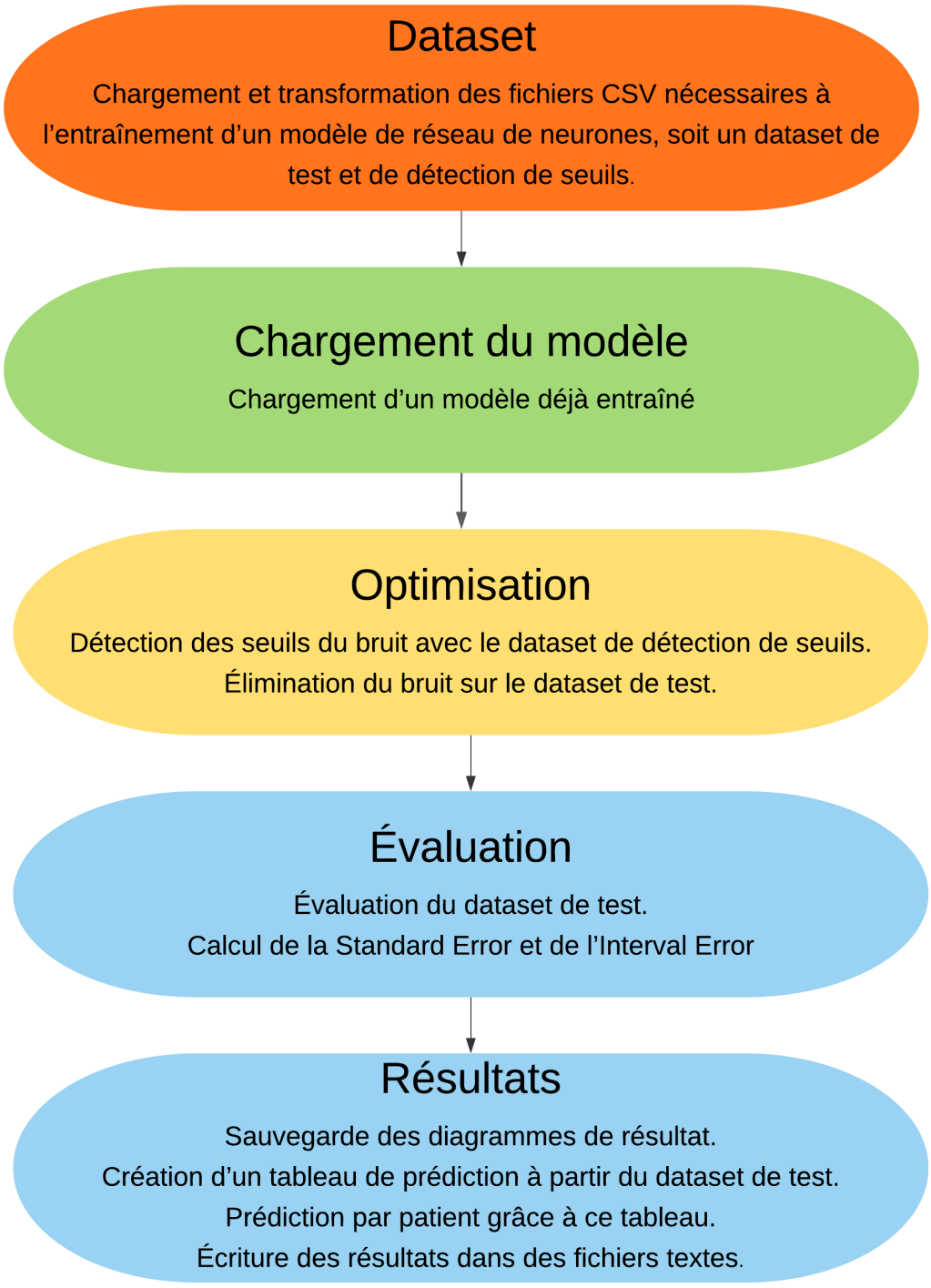


Figure 76 - Processus de test d'un modèle - Image de l'auteur

5.10.3. Prédiction de données

Le processus de prédiction de données peut être résumé en trois étapes, le tout durant un peu moins de trois minutes sur un ordinateur portable basique. La figure 77 décrit ces étapes.

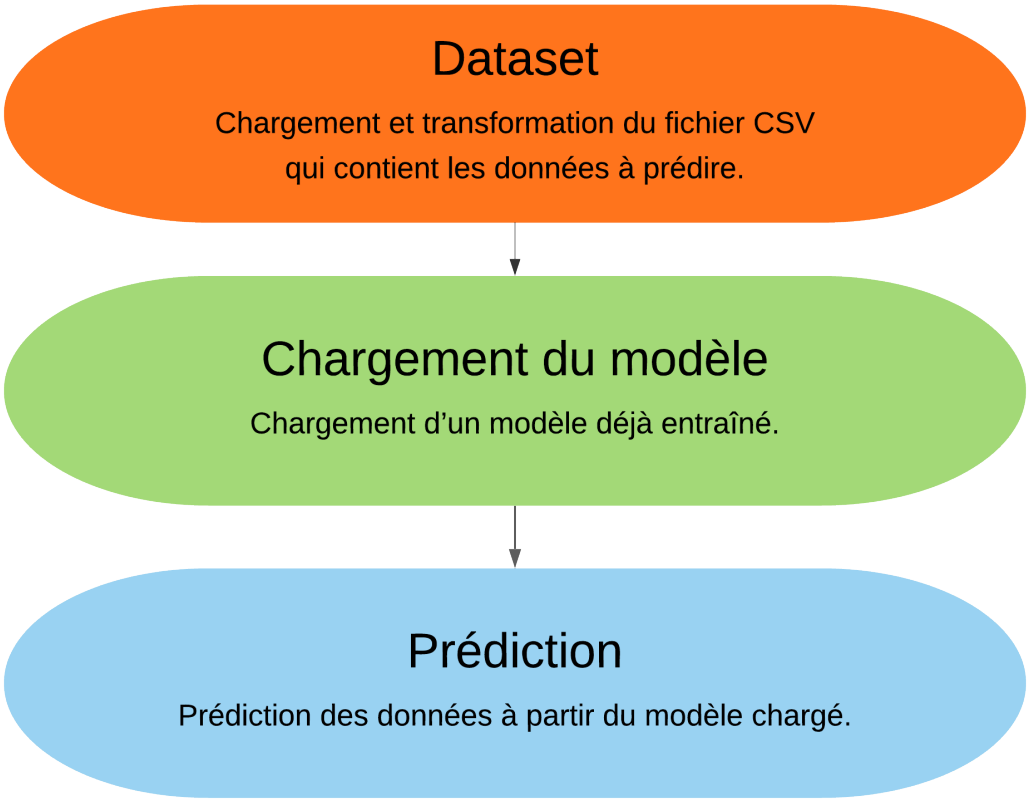


Figure 77 - Processus de prédiction de données - Image de l'auteur

6. Résultats et Conclusions

Lors de ce travail, de nombreux modèles de classification ont été explorés dans l'objectif d'atteindre la meilleure prédiction possible. Cependant, bon nombre de techniques d'optimisation et d'évaluation ont aussi gravité autour de cet objectif.

Nous avons débuté par explorer les modèles traditionnels de classification tels que le Decision Tree, le Random Forest, le Gradient Boosted Tree, le SVM et le Fuzzy Rule. Ces modèles ont démontré des résultats positifs et ont fixé la base du projet.

Puis le dataset est devenu le cœur de nos recherches lors de l'étude sur la corrélation et l'élimination de données. Nous en avons conclu qu'une telle élimination sur nos datasets était contreproductive.

Après s'être fait une idée claire sur le machine learning traditionnel, le deep learning a été évalué, en commençant par la construction d'un réseau de neurones multicouches et par son optimisation. Le CNN 1D a ensuite permis d'exploiter pleinement le potentiel du deep learning avec des résultats qui équivalaient ceux des modèles de classification traditionnels.

La prochaine étape fut l'étude du bruit qui prit toute son importance en augmentant la performance des modèles, tout en améliorant la qualité des datasets. Par ailleurs, la gestion du bruit nous a permis d'extraire des informations de nos données. Après l'élimination du bruit, la cross-validation et les itérations ont tissé un filet de sécurité pour nos modèles, validant leurs résultats.

Finalement, après la démonstration d'une classification concluante par patient, l'ensemble de l'environnement a été déployé et testé chez le client qui a validé la totalité des travaux.

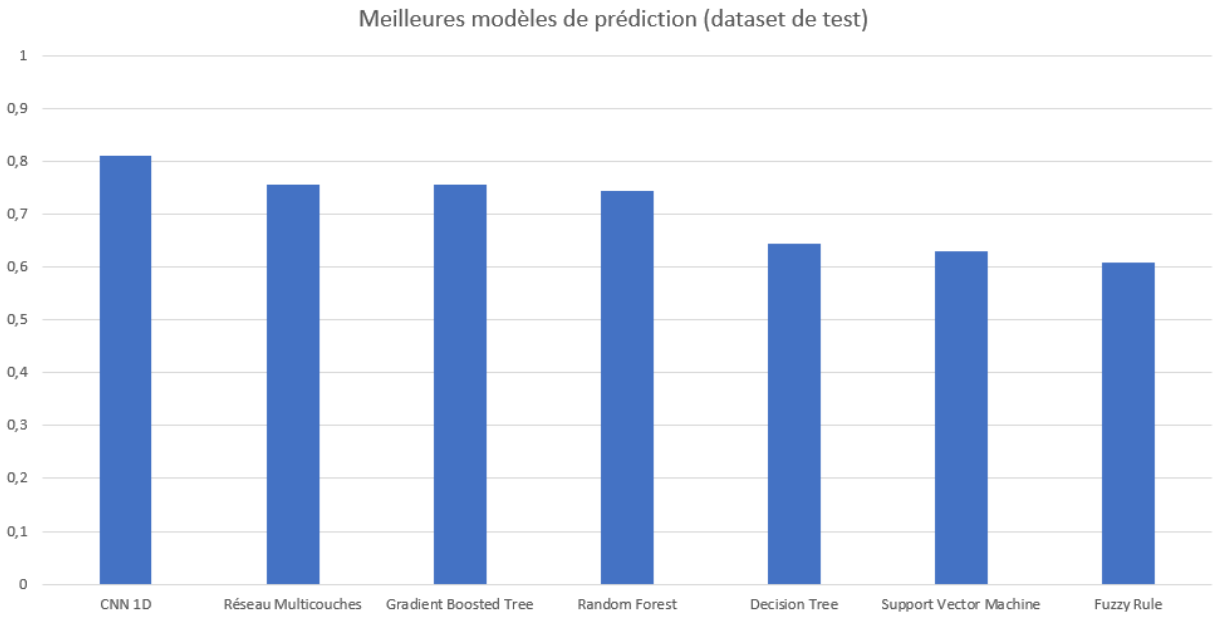
6.1. Résultats

Le meilleur modèle de prédiction de données pour notre contexte et notre dataset s'avère être le Convolutional Neural Networks 1D. Le tableau 20 résume les performances finales des différents algorithmes sur tous les aspects techniques demandés par le client.

Meilleures modèles de prédiction (dataset de test)								
	Bursts	AUC[SE]	Perte[SE]	Accuracy[SE]	TP True Positives	FP False Positives	TN True Negatives	FN False Negatives
CNN 1D	4580	0.8109$^{+/-0.0064}$	18,79%$^{+/-0.64}$	76,7%$^{+/-0.64}$	37% *1685 bursts*	8% *350 bursts*	40% *1828 bursts*	16% *717 bursts*
Réseau Multicouches	5049	0.7553$^{+/-0.0068}$	21,42%$^{+/-0.68}$	70,2%$^{+/-0.68}$	39% *1992 bursts*	17% *849 bursts*	33% *1646 bursts*	11% *562 bursts*
Gradient Boosted Tree	5747	0.7550$^{+/-0.0060}$	30,43%$^{+/-0.6}$	69,6%$^{+/-0.6}$	39% *2216 bursts*	12% *681 bursts*	31% *1782 bursts*	18% *1068 bursts*
Random Forest	5747	0.7450$^{+/-0.0060}$	29,31%$^{+/-0.6}$	70,7%$^{+/-0.6}$	37% *2137 bursts*	13% *760 bursts*	33% *1925 bursts*	16% *925 bursts*
Decision Tree	5747	0.6440$^{+/-0.0070}$	36,96%$^{+/-0.7}$	63,0%$^{+/-0.7}$	36% *2058 bursts*	15% *839 bursts*	27% *1565 bursts*	22% *1285 bursts*
SVM	5747	0.6310$^{+/-0.0070}$	40,44%$^{+/-0.7}$	59,6%$^{+/-0.7}$	42% *2418 bursts*	8% *479 bursts*	17% *1005 bursts*	32% *1845 bursts*
Fuzzy Rule	5080	0.6090$^{+/-0.0070}$	35,27%$^{+/-0.7}$	64,7%$^{+/-0.7}$	37% *1895 bursts*	13% *632 bursts*	27% *1393 bursts*	23% *1160 bursts*

Tableau 20 - Synthèse finale des résultats avec le dataset de test

La figure 78 compare les AUC des différents modèles de prédiction.



**SE : CNN 1D - 0.0064 / Réseau de neurones multicouches - 0.0068 / Gradient Boosted Tree - 0.006 / Random Forest - 0.006 / Decision Tree - 0.007 / Fuzzy Rule - 0.007 / Support Vector Machine - 0.007*

Figure 78 - Comparaison des AUC des différents modèles de classification - Image de l'auteur

6.2. Conclusion

Les objectifs de ce travail de Bachelor, soit être capable de prédire des Bursts sains ou malades selon les données récoltées par les lentilles et extraire des informations de ces données, sont totalement atteints.

6.2.1. Classification de la maladie

Avec son AUC de $0.8109^{+/-0.0064}$, le meilleur modèle parvient à détecter à $81{,}08\%^{+/-0.64}$ si un patient est sain ou malade et à $76.70\%^{+/-0.64}$ si un Burst est sain ou malade grâce à l'interface implémentée à la main et déployée chez Sensimed. Avec l'optimisation du modèle, nous avons augmenté le taux de prédiction de près de 10% en comparaison des premières implémentations du modèle, mais la route reste encore longue. Le modèle proposé pourra toujours être amélioré.

6.2.2. Bilan Technique

Le code découlant de ce projet est simple et léger, mais très complet. Il gère un nombre conséquent de métriques et permet tout autant d'entraîner des modèles, de tester des modèles que de prévoir des données. De plus, étant déployé et fonctionnel chez Sensimed, le code pourra être altéré et optimisé pour leurs besoins futurs.

Sa simplicité et sa flexibilité sont aussi dues à l'API de Keras qui simplifie grandement la construction d'un modèle et de TensorFlow qui cumule une suite d'outils efficaces.

KNIME, quant à lui, nous a permis d'entraîner, de tester, d'implémenter et de présenter des modèles de classification en un temps record.

Finalement, l'IDE PyCharm et un outil d'environnement tel que Miniconda permettent d'implémenter, de tester, de présenter et de partager rapidement nos scripts python.

6.2.3. Problèmes Rencontrés

Les problèmes principaux de ce projet ont été les capacités de calcul de nos machines. Même si un ordinateur portable basique permet de faire tourner nos modèles, nous conseillons d'utiliser des machines avec des cartes graphiques performantes pour exploiter toutes les capacités de TensorFlow et de Keras.

Un temps conséquent a été dédié à la transformation des données. En effet, chaque modèle exige un format précis de données qui découlent des datasets. Il est donc primordial de prendre le temps d'analyser et de transformer de la bonne manière nos données, avant d'implémenter un modèle de réseau de neurones, comme le CNN 1D par exemple.

Nous avons aussi perdu du temps sur le résultat des premiers modèles. Ces derniers prédisaient toujours des cas malades, un résultat invisible avec des métriques comme la perte et l'accuracy. Il

est donc primordial d'intégrer rapidement de multiples métriques importantes comme l'AUC ou le recall pour avoir une bonne vue d'ensemble le plus vite possible.

6.2.4. Améliorations Futures et Recommandations

Le modèle de classification CNN 1D est pertinent pour la problématique, cependant il est possible de l'améliorer d'encore bien des manières. Chaque paramètre pourrait améliorer la classification du modèle. De plus, la gestion de bruit pourrait être approfondie, tout comme la notion de seuil de décision entre sain et malade qui reste pour le moment fixé à 0.5.

Par ailleurs, plus les spécialistes de Sensimed analyseront les données récoltées avec leurs lentilles, plus ils seront capables de détecter et de retirer du bruit pour garder des Bursts totalement purs. De plus, une analyse pourrait être menée sur les données extraites du bruit du CNN 1D qui pourraient ouvrir une piste au sujet de la détection du glaucome.

C'est pourquoi les deux aspects que sont l'optimisation du modèle en général et l'affinement des datasets pourront être améliorés dans le futur.

6.2.5. Bilan Général

Au terme de ce projet de fin de Bachelor, tous les objectifs ont été atteints. Nous avons confectionné un modèle de classification de données unique et opérationnel, qui est capable de prédire à plus de 80% si un patient est sain ou malade. De plus, les analyses sur le bruit ont ouvert des pistes de recherche pour Sensimed dans son travail de compréhension et d'optimisation de ses données. Finalement, le client a pu recevoir l'intégralité des recherches et du code, qu'il pourra utiliser directement au sein de son environnement de travail. Ces conclusions seront une base solide pour les recherches de Sensimed et pourquoi pas, une aide, aussi modeste soit-elle, dans le combat contre le glaucome.

6.2.6. Conclusion Personnelle

J'ai toujours aspiré à travailler dans le domaine de l'Intelligence Artificielle. A la découverte de ce projet, je n'ai donc pas hésité une seconde pour le choisir.

Je suis néanmoins parti de rien, sans connaissance en deep learning ou en python. Avec mon maigre bagage, j'ai affronté ce projet et j'ai plus appris en quelques mois, qu'en plusieurs années.

Je me suis épanoui dans chaque étape de cette épreuve, aussi difficile soit-elle, que ce soit la recherche de connaissance sur le machine learning, la création de mon premier réseau de neurones ou la découverte des résultats finaux de mon modèle le plus abouti.

Ce travail de Bachelor m'a apporté la confirmation que ce domaine était fait pour moi et m'a permis d'acquérir une confiance solide en vue de mes futures épreuves professionnelles. Il sera un pilier solide qui soutiendra la suite de ma carrière et de mes études.

Références

Amini, A. (2021a, Février 5). *MIT 6.S191 (2020): Introduction to Deep Learning*. Retrieved Avril 29, 2021, from MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=njKP3FqW3Sk

Amini, A. (2021b, Février 19). *MIT 6.S191: Convolutional Neural Networks*. Consulté le Avril 29, 2021, sur MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=AjtX1N_VT9E&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=4

Amini, A. (2021c, Mars 19). *MIT 6.S191: Evidential Deep Learning and Uncertainty*. Consulté le Avril 29, 2021, sur MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=toTcf7tZK8c&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=7

Amini, A. (2021d, Mars 5). *MIT 6.S191: Reinforcement Learning*. Consulté le Avril 30, 2021, sur MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=93M1l_nrhpQ&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=5

Awad, W., & ELseuofi, S. (2011, Février 01). *Machine Learning methods for E-mail Classification*. Consulté le Août 05, 2021, sur Citeseerx: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.3185&rep=rep1&type=pdf

Ba, J., & Kingma, D. (2014, Decembre 22). *Adam: A Method for Stochastic Optimization*. Consulté le Août 03, 2021, sur Cornell University: https://arxiv.org/abs/1412.6980

Bae, J.-M. (2014, Octobre 30). *The clinical decision analysis using decision tree*. Consulté le 22 juin, 2021, sur The National Center for Biotechnology Information advances science and health: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4251295/

Breiman, L. (2021, janvier 13). *Université Paris-Saclay*. Consulté le Juin 22, 2021, sur Arbres de régression, CART: https://www.imo.universite-paris-saclay.fr/~goude/Materials/ProjetMLF/cart_beamer.pdf

Brownlee, J. (2016, Septembre 12). *How to Configure the Gradient Boosting Algorithm*. Consulté le Août 05, 2021, sur Marchine Learning Mastery: https://machinelearningmastery.com/configure-gradient-boosting-algorithm/

Charniak, E. (2018). *Introduction to Deep Learning*. Consulté le Août 03, 2021, sur https://books.google.ch/books?hl=fr&lr=&id=Z3gSEAAAQBAJ&oi=fnd&pg=PR1&dq=neurone+deep+learning&ots=m_J1AP0x_U&sig=_woTf_5JU-N8Qn7d5Z-mDgS6lvk#v=onepage&q&f=false

Conda. (2017). *Miniconda*. Consulté le Juillet 16, 2021, sur Conda: https://docs.conda.io/en/latest/miniconda.html

E. Dahl, G., N. Sainath, T., & E. Hinton, G. (2013, Octobre 12). *2013 IEEE International Conference*. (IEEE, Éd.) doi:10.1109/ICASSP.2013.6639346

Farhadi, F. (2007, Décembre). *Learning Activation Functions in Deep Neural Networks*. (PolyPublie, Éd.) Consulté le Juin 03, 2021, sur Institutional Repository of Polytechnique Montréal: https://publications.polymtl.ca/2945/

Forkheim, K., Scuse, D., & Pasterkamp, H. (1995, Mai 15). *A comparison of neural network models for wheeze detection*. (IEEE, Éd.) doi:10.1109/WESCAN.1995.493973

Freund, Y., & Schapire, R. (1997). *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**. Consulté le Mai 29, 2021, sur ScienceDirect: https://www.sciencedirect.com/science/article/pii/S002200009791504X

Gavrila, D., & Philomin, V. (2002, Août 06). *Real-Time Object Detection For "Smart" Vehicules*. (IEEE, Éd.) Consulté le Août 03, 2021, sur IEEE: https://ieeexplore.ieee.org/abstract/document/791202?casa_token=f4eK_mgQUpEAAAAA:Ev0cx-nHaMg9FSqjQfzowZKhO8AbPeqiQFbBoMj0FFmf-DQlylfgOL2mYJkhdtRTTgyhZUNBEg

Gnana Sheela, K., & Deepa, S. (2013, Juin 20). *Review on Methods to Fix Number of Hidden Neurons in Neural Networks*. Consulté le 27 Juillet, 2021, sur Hindawi: https://www.hindawi.com/journals/mpe/2013/425740/

Gokgoz, E. (2015, January 24). *Comparison of decision tree algorithms for EMG signal classification using DWT*. Consulté le Juin 22, 2021, sur ScienceDirect: https://www.sciencedirect.com/science/article/pii/S1746809414002006?casa_token=FU81zLPvCHwAAAAA:JKd_0j8Sr3jxGvZ6f8DbtzqsB_OPBd1QqwGWWG_3hzaRWONIfsuhbz75GxSNd8nwh9_8sujd_os

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Consulté le 29 Mai, 2021, sur https://www.deeplearningbook.org/

Goodrich, J. (2021, Janvier 25). *How IBM's Deep Blue Beat World Champion Chess Player Garry Kasparov*. Consulté le Aout 03, 2021, sur IEEE Spectrum: https://spectrum.ieee.org/how-ibms-deep-blue-beat-world-champion-chess-player-garry-kasparov

Gordon, J. (2017, Septembre 13). *Let's Write a Decision Tree Classifier from Scratch*. Consulté le Mars 27, 2021, sur Youtube: https://www.youtube.com/watch?v=LDRbO9a6XPU&ab_channel=GoogleDevelopers

Hackr. (2020, Mai 22). *What is Fuzzy Logic? Advantage and Disadvantage*. Consulté le Avril 29, 2021, sur Find the Best Data Science Courses & Tutorials: https://hackr.io/blog/what-is-fuzzy-logic

Hanley, J., & McNeil, B. (1982, Avril). *The meaning and use of the area under a receiver operating characteristic (ROC) curve*. doi:10.1148/radiology.143.1.7063747

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Consulté le Mai 29, 2021, sur Google Scho: https://books.google.ch/books?hl=fr&lr=&id=tVIjmNS3Ob8C&oi=fnd&pg=PR13&ots=EOD8L7I315&sig=77NcVsvbDNXJ7ubEByVuZJ9yyZ0&redir_esc=y#v=onepage&q&f=false

Hearst, M., Dumais, S., Osuna, E., Platt, J., & Scholkopf, B. (1998, Juillet). Support vector machines. *IEEE*, 28. Consulté le Mai 29, 2021, sur https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=708428

Himanshu, S. (2019, Janvier 19). *Activation Functions : Sigmoid, tanh, ReLU, Leaky ReLU, PReLU, ELU, Threshold ReLU and Softmax basics for Neural Networks and Deep Learning*. Consulté le Juillet 23, 2021, sur Medium: https://himanshuxd.medium.com/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e

Hôpitaux Universitaires Genève. (2015, Mars). *Le glaucome et ses traitements*. Consulté le Avril 14, 2021, sur Hôpitaux Universitaires Genève: https://www.hug.ch/sites/interhug/files/documents/glaucome_traitements.pdf

IBM. (2020). *Présentation générale de CRISP-DM*. Consulté le 22 juin, 2021, sur IBM: https://www.ibm.com/docs/fr/spss-modeler/SaaS?topic=dm-crisp-help-overview

Intel. (2021). *CPU vs GPU : Tirer le maximum des deux*. Consulté le Avril 03, 2021, sur Intel: https://www.intel.fr/content/www/fr/fr/products/docs/processors/cpu-vs-gpu.html

Issarane, H. (2020, Mars 2). *6 Applications De SVM*. Consulté le Juillet 27, 2021, sur Analytics & Insights: https://analyticsinsights.io/6-applications-svm/

Jerry, Y., Chow, J.-H., Chen, J., & Zheng, Z. (2009, Novembre). *Stochastic Gradient Boosted Distributed Decision Trees*. Consulté le Mai 29, 2021, sur ACM Digital Library: https://dl.acm.org/doi/pdf/10.1145/1645953.1646301

Johnson, W. (2016, Février 27). *Using Gini Split / Gini Index*. Récupéré sur Learn by Marketing: https://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

Kallner, A. (2014). *Area Under the Curve*. Consulté le Août 03, 2021, sur ScienceDirect: https://www.sciencedirect.com/topics/chemistry/area-under-the-curve

Karsoliya, S. (2012). *Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture*. Consulté le Juillet 27, 2021, sur Local Gov: http://ijettjournal.org/archive/ijett-v3i6p206

Kayid, A., & Khaled, Y. (2018, Mai 07). *Performance of CPUs/GPUs for Deep Learning workloads*. Consulté le Août 03, 2021, sur Research Gate: https://www.researchgate.net/profile/Yasmeen-Khaled/publication/325023664_Performance_of_CPUsGPUs_for_Deep_Learning_workloads/links/5ce182c992851c4eabaf8fc1/Performance-of-CPUs-GPUs-for-Deep-Learning-workloads.pdf

Keras. (2021a). *Losses*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/losses/

Keras. (2021b). *Accuracy metrics*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/metrics/accuracy_metrics/

Keras. (2021c). *Classification metrics based on True/False positives & negatives*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/metrics/classification_metrics/

Keras. (2021d). *Keras: the Python deep learning API*. Consulté le Juin 17, 2021, sur Keras: https://keras.io/

Keras. (2021e). *SGD*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/sgd/

Keras. (2021f). *Adagrad*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/adagrad/

Keras. (2021g). *Adadelta*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/adadelta/

Keras. (2021h). *RMSprop*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/rmsprop/

Keras. (2021i). *Nadam*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/Nadam/

Keras. (2021j). *Ftrl*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/ftrl/

Keras. (2021k). *Adamax*. Consulté le Août 03, 2021, sur Keras: https://keras.io/api/optimizers/adamax/

Kiranyaz, S., Ince, T., Abdeljaber, O., Avci, O., & Gabbouj, M. (2019, Mai 12). *1-D Convolutional Neural Networks for Signal Processing Applications*. (IEEE, Éd.) Consulté le Juillet 05, 2021, sur IEEExplore: https://ieeexplore.ieee.org/abstract/document/8682194?casa_token=tjRKUiM-WbIAAAAA:wlxkhyHNsbUvudH6ULuFM0MO-s6OOL15Xc1LiQBQUxCPlu99AJT_fv9a_xdaE2aEIG-yJ2sncw

Kiwi, L., Comtesse, X., Walch, D., Garbinato, A., Genoud, D., Pauletto, G., & Ramuz, G. (2018). *Entreprise Augmentée IA*. Editions G d'Encre. Consulté le Juillet 22, 2021, sur https://books.google.ch/books/about/IA_entreprise_augment%C3%A9e_m%C3%A9decine_augme.html?id=Y-VOvwEACAAJ&redir_esc=y

KNIME. (2020b, Janvier 21). *Linear correlation*. Consulté le Juin 03, 2021, sur Knime Hub: https://hub.knime.com/mlauber71/spaces/Public/latest/forum/kn_forum_correlation~GHtXhJcq8Lhqwppa

KNIME. (2021). *End to End Data Science*. Consulté le Juin 18, 2021, sur KNIME | Open for Innovation: https://www.knime.com/

KNIME. (2021a, Février 11). *Backward Feature Elimination*. Consulté le Juin 03, 2021, sur Knime Hub: https://hub.knime.com/knime/spaces/Academic%20Alliance/latest/Guide%20to%20Intelligent%20Data%20Science/Example%20Workflows/Chapter6/03_Backward_Feature_Elimination~Cim28Kbg7Rt2EsE8

Krauss, C., AnhDo, X., & Huck, N. (2017, Juin 1). *Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500*. Consulté le Mai 29, 2021, sur ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0377221716308657?casa_token=r_Ytvzim-QYAAAAA:DZtZ3LTSQdDgQfyF7cUQkhdnRyLNm139qvAsGP90i0z2MW3pFQy1jsyE2MgXU5qWVd-rwvAW0g#bib0056

La Langue Française. (s.d.). *Convolution : définition de « convolution » | La langue française*. Consulté le Juillet 27, 2021, sur La langue française: https://www.lalanguefrancaise.com/dictionnaire/definition/convolution

Lazcano-Gomez, G., De los Angeles Ramos-Cadena, M., Torres-Tamayo, M., Hernandez de Oteyza, A., Turati-Acosta, M., & Jimenez-Román, J. (2016, Novembre 28). *Cost of glaucoma treatment in a developing country over a 5-year period*. doi:10.1097/MD.0000000000005341

Lutu, P. (2011). *Using Confusion Matrices and Confusion Graphs to Design Ensemble Classification Models from Large Datasets*. Consulté le Août 03, 2021, sur https://link.springer.com/chapter/10.1007/978-3-642-23544-3_23

Miller, A., Blott, B., & Hames, T. (1992, Septembre). *Review of neural network applications in medical imaging and signal processing*. Consulté le Juin 16, 2021, sur Springer: https://link.springer.com/article/10.1007/BF02457822

Naive Data Scientist. (2020, Septembre 6). *Decision Tree – Root Nodes, Internal Nodes and Leaf Nodes*. Consulté le Mars 27, 2021, sur Naivedatascientist: https://naivedatascientist.co.in/2020/09/06/intro-to-decision-tree/

Présentation générale de CRISP-DM. (s.d.). Consulté le Mai 15, 2021, sur IBM: https://www.ibm.com/docs/fr/spss-modeler/SaaS?topic=dm-crisp-help-overview

Python. (2021). *About Python*. Consulté le Juin 18, 2021, sur Python.org: https://www.python.org/about/

Saedsayad. (s.d.). *Decision Tree - Classification*. Consulté le Mars 27, 2021, sur Saedsayad: https://www.saedsayad.com/decision_tree.htm

Sauro, J. (2014, Septembre 3). *How to Compute a Confidence Interval in 5 Easy Steps*. Consulté le Juillet 27, 2021, sur MeasuringU: https://measuringu.com/ci-five-steps/

Sharma, S. (2017, Septembre 6). *Activation Functions in Neural Networks*. Consulté le Juillet 23, 2021, sur Toward Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Soleimany, A. (2021a, Février 12). *MIT 6.S191: Recurrent Neural Networks*. Récupéré sur MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=qjrad0V0uJE

Soleimany, A. (2021b, Février 26). *MIT 6.S191: Deep Generative Modeling*. Consulté le Avril 29, 2021, sur MIT Deep Learning 6.S191: https://www.youtube.com/watch?v=BUNl0To1IVw&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=4

Sourek, G., Hubacek, O., & Zelezny, F. (2017, Mai 17). *Learning to predict soccer results from relational data with gradient boosted trees*. Consulté le Août 05, 2021, sur https://link.springer.com/content/pdf/10.1007/s10994-018-5704-6.pdf

Tamura, S. (1989, Mai 26). *An analysis of a noise reduction neural network*. (IEEE, Éd.) doi:10.1109/ICASSP.1989.266851

TensorFlow. (2021, Mai 14). *tf.keras.layers.GaussianNoise*. Consulté le Juin 15, 2021, sur TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/layers/GaussianNoise

TensorFlow. (2021). *Une plate-forme Open Source de bout en bout dédiée au machine learning*. Consulté le Juin 17, 2021, sur TensorFlow: https://www.tensorflow.org/?hl=fr

Treboux, J., Ingold, R., & Genoud, D. (2020, Juin 03). *Towards Retraining of Machine Learning Algorithms: An Efficiency Analysis Applied to Smart Agriculture*. Consulté le Août 04, 2021, sur IEEE: https://ieeexplore.ieee.org/abstract/document/9119601/authors

Westreich, D., Lessler, J., & Jonsson Funka, M. (2010, Août). Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. *Journal of Clinical Epidemiology, 63*(8), 833. Consulté le Avril 29, 2021, sur https://www.sciencedirect.com/science/article/pii/S0895435610001022

Zhou, Q., Zhou, H., Zhou, Q., Yang, F., & Luo, L. (2014, Mai 3). *Structure damage detection based on random forest recursive feature elimination*. Consulté le Juin 22, 2021, sur ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0888327013006821?casa_token=wLjSMTzIwA0AAAAA:bGSlQZ5aFMrJNHHGAZi7EZmqd-vGg__lK8eJlbPhz16sr0IRCY992cxLQsOh6XRkcf4zEyXJi38#!

Zhou, V. (2019, Mars 3). *Machine Learning for Beginners: An Introduction to Neural Networks*. Consulté le Juin 3, 2021, sur Victor Zhou: https://victorzhou.com/blog/intro-to-neural-networks/

Annexe I : AdvancedDetectionEyeIllness_LAUNCHER.py

```
# ============================================================================================ #
# Advanced Detection Eye Illness LAUNCHER ==================================================== #
# QUENTIN NATER - VERSION 27.1 =============================================================== #
# UP : 04.08.2021 - CR : 01.07.2021 ========================================================== #
# ============================================================================================ #
from AdvancedDetectionEyeIllness import test_model, train_model, prediction_data

# TEST ##################################################
# Example of a test CNN 1D
prediction_00, probability_00, history_00, history_test_00, se_00, interval_00 = test_model(
    model_type="cnn1d",
    model_directory="./ml_official/model_cnn1d",
    test_dataset="\\dataset\\data_test_glaucome.csv",
    threshold_dataset="\\dataset\\data_threshold_glaucome.csv",
    generate_plots="yes")

# Example of a test multilayers
prediction_01, probability_01, history_01, history_test_01, se_01, interval_01 = test_model(
    model_type="multilayers",
    model_directory="./ml_official/model_multilayers",
    test_dataset="\\dataset\\data_test_glaucome.csv",
    threshold_dataset="\\dataset\\data_threshold_glaucome.csv",
    generate_plots="yes")


# TRAIN #################################################
# Example of a training of CNN 1D
prediction_10, probability_10, history_10, history_test_10, se_10, interval_10 = train_model(
    model_type="cnn1d",
    epochs=60,
    number_loops=1,
    train_dataset="\\dataset\\data_train_glaucome.csv",
    test_dataset="\\dataset\\data_test_glaucome.csv",
    threshold_dataset="\\dataset\\data_threshold_glaucome.csv",
    cross_validation="no")

# Example of a training of multilayers
prediction_11, probability_11, history_11, history_test_11, se_11, interval_11 = train_model(
    model_type="multilayers",
    epochs=200,
    train_dataset="\\dataset\\data_train_glaucome.csv",
    test_dataset="\\dataset\\data_test_glaucome.csv",
    threshold_dataset="\\dataset\\data_threshold_glaucome.csv",
    number_loops=1,
    cross_validation="no",
    generate_plots="no")


# PREDICTION ############################################
# Example of prediction based on data (cnn)
prediction_20, probability_20 = prediction_data(
    model_type="cnn1d",
    model_directory="./ml_official/model_cnn1d",
    directory_filename="\\dataset\\test_prediction.csv")

# Example of prediction based on data (multilayers)
prediction_21, probability_21 = prediction_data(
    model_type="multilayers",
    model_directory="./ml_official/model_multilayers",
    directory_filename="\\dataset\\test_prediction.csv")
```

Source : Code de l'auteur (https://github.com/qnater/AdvancedDetectionEyeIllness)

Annexe II : Dataset.py

```
class Dataset:
    """
    Object with all datasets for training, testing or thresholding

    Quentin Nater 2021

    :param origin :    original dataset with ID + data /
    :param data :      dataset with all data /
    :param label :     dataset with all labels /
    :param patient :   dataset with all patient ids /
    :param row_id :    dataset with all entry ids /
    :param burst_id :  dataset with all burst ids /
    """
    def __init__(self, origin, data, label, patient, row_id, burst_id):
        self.origin = origin
        self.data = data
        self.label = label
        self.patient = patient
        self.row_id = row_id
        self.burst_id = burst_id
```

Source : Code de l'auteur (https://github.com/qnater/AdvancedDetectionEyeIllness)

Annexe III : Options.py

```
class Options:
    """
    Object with all information about training or testing

    Quentin Nater 2021


    :param experiment :                   name of the current experimentation
    :param tag :                          precision about the experimentation
    :param batch_size :                   size of a batch (in the training)
    :param verbose :                      enable to have information during the training
    :param split :                        let cross-validation inside the training
    :param directory :                    name of the directory which contains plots and models
    :param path :                         path to the plots directory
    :param ml :                           path to the machine learning model directory
    :param epochs :                       number of iteration during the training
    :param validation :                   using or not the home made cross-validation
    :param model :                        model used for the training or the testing
    """
    def __init__(self, experiment, tag, batch_size, verbose, split, directory, path, ml, epochs,
validation, model):
        self.experiment = experiment
        self.tag = tag
        self.batch_size = batch_size
        self.verbose = verbose
        self.split = split
        self.directory = directory
        self.path = path
        self.ml = ml
        self.epochs = epochs
        self.validation = validation
        self.model = model

    def to_string(self):
        print('MODEL USED', ':', str(self.model), sep="\t")
        print('EXPERIMENT', ':', str(self.experiment), sep="\t")
        print('TAG (INFO)', ':', str(self.tag), sep="\t")
        print('EPOCHS #', ':', str(self.epochs), sep="\t")
        print('BATCH_SIZE', ':', str(self.batch_size), sep="\t")
        print('VERBOSE ', ':', str(self.verbose), sep="\t")
        print('VAL SPLIT ', ':', str(self.split), sep="\t")
        print('CROSS VAL.', ':', str(self.validation), sep="\t")
        print('PLOTS PATH', ':', str(self.path), sep="\t")
```

Source : Code de l'auteur (https://github.com/qnater/AdvancedDetectionEyeIllness)

Annexe IV : AdvancedDetectionEyeIllness.py

```
# ============================================================================== #
# Advanced Detection Eye Illness ============================================== #
# QUENTIN NATER - VERSION 27.1 ================================================= #
# UP : 04.08.2021 - CR : 02.04.2021 ============================================ #
# ============================================================================== #

import os
import math
import numpy
import pandas as pd
from tqdm import tqdm
from numpy import dstack
from datetime import datetime
import matplotlib.pyplot as plot
import tensorflow.keras.metrics as tf
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from tensorflow.python.keras.layers import Dense, Conv1D, MaxPooling1D, GlobalAveragePooling1D, Dropout,
GaussianNoise
from tensorflow.python.keras.models import Sequential, save_model, load_model

from obj.Dataset import *
from obj.Options import *

print("=================================================================")
print("========== Advanced Detection Eye Illness Deep Learning ==========")
print("=============== Quentin Nater === HES-SO === 2021 ===============")
print("=================================================================")


def load_dataset(model_name, location):
    """
    Load a dataset (train, test or threshold), create all necessary datasets and convert the data format

    Quentin Nater 2021


    :param model_name :     "multilayers" Multilayers Neural Network / "cnn1d" CNN 1D
    :param location :       location of the dataset


    :returns: imported_dataset : Dataset object with all datasets (origin, data, label, patient, row_id,
burst_id)
    """
    # Loading of the dataset and copy for transformation
    data = pd.read_csv(os.getcwd() + location, header=None)
    d = pd.read_csv(os.getcwd() + location, header=None)

    # Creation of the future splitting datasets
    patient, label, ids, burst_id = numpy.zeros((1, 1), dtype=numpy.int64), \
        numpy.zeros((1, 1), dtype=numpy.int64), \
        numpy.zeros((1, 1), dtype=numpy.int64), \
        numpy.zeros((1, 1), dtype=numpy.int64)

    # loop increment and data file length
    x, data_length = 0, len(d)

    # Insert the data into the new splitting datasets
    while x != data_length:
        ids = numpy.insert(ids, ids.size, int(data[0][x]), axis=0)
        patient = numpy.insert(patient, patient.size, int(data[1][x]), axis=0)
        burst_id = numpy.insert(burst_id, burst_id.size, int(data[2][x]), axis=0)
        label = numpy.insert(label, label.size, int(data[3][x]), axis=0)
        x = x + 1

    # Delete the 1st line added in the creation
    patient = numpy.delete(patient, 0, 0)
    label = numpy.delete(label, 0, 0)
    ids = numpy.delete(ids, 0, 0)
    burst_id = numpy.delete(burst_id, 0, 0)

    # Delete all ID columns to keep only the data from the origin dataset
    data.drop(data.columns[0], axis=1, inplace=True)
    data.drop(data.columns[0], axis=1, inplace=True)
    data.drop(data.columns[0], axis=1, inplace=True)
    data.drop(data.columns[0], axis=1, inplace=True)

    # Different transformation if it's CNN 1D or Multilayers
    if model_name == "cnn1d":
```

```
        # Get the values of the array
        data = data.values
        tmp_list = list()
        tmp_list.append(data)

        # Stack group so that features are the 3rd dimension
        data = dstack(tmp_list)
    else:
        data = data.to_numpy()
    d = d.to_numpy()

    # Set the Dataset object with all different datasets to simply the process
    imported_dataset = Dataset(d, data, label, patient, ids, burst_id)

    return imported_dataset


def load_data(model_name, location):
    """
    Load a dataset (train, test or threshold), create all necessary dataset and convert the data

    Quentin Nater 2021


    :param model_name :     "multilayers" Multilayers / "cnn1d" CNN 1D
    :param location :       train / test / threshold


    :return:   data :      dataset with all data
    """

    # Loading of the dataset and copy for transformation
    data = pd.read_csv(os.getcwd() + location, header=None)

    if model_name == "cnn1d":
        # Get the values of the array
        data = data.values
        tmp_list = list()
        tmp_list.append(data)

        # Stack group so that features are the 3rd dimension
        data = dstack(tmp_list)
    else:
        data = data.to_numpy()

    return data


def model_cnn1d(timesteps, features, output):
    """
    Set the CCN 1D model for the deep learning

    Quentin Nater 2021


    :param timesteps :  Number of columns use in the dataset
    :param features :   Number of dimensions use in the dataset
    :param output :     Number of output in the classification


    :return: model :    Saved model for the training
    """

    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=9, activation='relu', input_shape=(timesteps, features)))
    model.add(Conv1D(filters=64, kernel_size=9, activation='relu'))
    model.add(GaussianNoise(0.1))
    model.add(MaxPooling1D(pool_size=3))
    model.add(Conv1D(filters=10, kernel_size=4, activation='relu'))
    model.add(Conv1D(filters=10, kernel_size=4, activation='relu'))
    model.add(GlobalAveragePooling1D())
    model.add(Dropout(0.5))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(output, activation='sigmoid'))
    print(model.summary())  # display in the console
    return model


def model_multilayers(timestep):
    """
    Set the Multilayers Neural Network model for the deep learning
```

```
    Quentin Nater 2021


    :param timestep :   Number of columns use in the dataset


    :return: model :    Saved model for the training
    """

    # Neurones creation
    model = Sequential()
    model.add(Dense(302, activation='relu', input_dim=timestep))
    model.add(Dense(302, activation='relu'))
    model.add(Dense(302, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    print(model.summary())  # display in the console
    return model


def write_results_test(history, p, name, se, interval, tag):
    """
    Write all results of the test in a text file inside the plots directory

    Quentin Nater 2021


    :param history :    Keras array with all results of the evaluation
    :param p :          Path to the plots directory
    :param name :       Unique name of the model
    :param se :         Standard Error for the AUC
    :param interval :   Interval for the AUC
    :param tag :        Tag of the file
    """

    # Path of the new file
    file = p + tag + "_model_results.txt"

    # Remove the file if it already exists
    if os.path.exists(file):
        os.remove(file)

    # Write all information
    f1 = open(file, "a")
    f1.write("==== " + name + " MODEL RESULT ================ \n")
    f1.write("==========" + tag + "========================== \n")
    f1.write("AUC........." + tag + "... : " + str(history[4]) + " \n")
    f1.write("SE........." + tag + "... : " + str(se) + " \n")
    f1.write("INTERVAL...." + tag + "... : " + str(interval) + " \n")
    f1.write("LOSS........" + tag + "... : " + str(history[0]) + " \n")
    f1.write("ACCURACY...." + tag + "... : " + str(history[1]) + " \n")
    f1.write("PRECISION..." + tag + "... : " + str(history[2]) + " \n")
    f1.write("RECALL....." + tag + "... : " + str(history[3]) + " \n")
    f1.write("TN........." + tag + "... : " + str(history[5]) + " \n")
    f1.write("FN........." + tag + "... : " + str(history[6]) + " \n")
    f1.write("TP........." + tag + "... : " + str(history[7]) + " \n")
    f1.write("FP........." + tag + "... : " + str(history[8]) + " \n")
    f1.close()


def write_results_train(history, p, name, epoch, tag):
    """
    Write all results of the training in a text file inside the plots directory

    Quentin Nater 2021


    :param history :    Keras array with all results of the evaluation
    :param p :          Path to the plots directory
    :param name :       Unique name of the model
    :param epoch :      Number of iterations in the training
    :param tag :        Tag of the file
    """

    # Path of the new file
    file = p + tag + "_model_train_results.txt"

    # Remove the file if it already exists
    if os.path.exists(file):
        os.remove(file)

    # Create the file and it's header
    f1 = open(file, "a")
    f1.write("==== " + name + " MODEL RESULT / EPOCH " + str(epoch) + " ================ \n")
```

```
    # Get the name of the columns inside the history
    list_of_keys = list(history.keys())

    # write the result of each epoch
    for inc in range(0, epoch, 1):
        f1.write(" \n")
        f1.write("EPOCH N°...." + tag + "... : " + str(inc) + " \n")
        f1.write("AUC........." + tag + "... : " + str(history[list_of_keys[4]][inc]) + " \n")
        f1.write("LOSS........" + tag + "... : " + str(history[list_of_keys[0]][inc]) + " \n")
        f1.write("ACCURACY...." + tag + "... : " + str(history[list_of_keys[1]][inc]) + " \n")
        f1.write("PRECISION..." + tag + "... : " + str(history[list_of_keys[2]][inc]) + " \n")
        f1.write("RECALL....." + tag + "... : " + str(history[list_of_keys[3]][inc]) + " \n")
        f1.write("TN........." + tag + "... : " + str(history[list_of_keys[5]][inc]) + " \n")
        f1.write("FN........." + tag + "... : " + str(history[list_of_keys[6]][inc]) + " \n")
        f1.write("TP........." + tag + "... : " + str(history[list_of_keys[7]][inc]) + " \n")
        f1.write("FP........." + tag + "... : " + str(history[list_of_keys[8]][inc]) + " \n")
        f1.write(" \n")
    f1.close()


def write_results_prediction(p, labeled, prediction, model):
    """
    Write all results of the prediction in a text file inside the plots directory

    Quentin Nater 2021

    :param p :              Directory where the prediction will be saved /
    :param labeled :        Classification array of the given dataset /
    :param prediction :     Float prediction array of the given dataset /
    :param model :          Name of the model used
    """

    # Current date to generate unique file name
    now = datetime.now()
    date = now.strftime("%d-%m-%Y-%H-%M-%S")

    # Path of the new file with dynamic date name
    file = p + model + date + ".txt"

    # Remove the file if it already exists
    if os.path.exists(file):
        os.remove(file)

    w = open(file, "a")
    w.write("==== " + model + " MODEL PREDICTION ================ \n")
    for x in range(len(labeled)):
        # Write all information
        w.write(
            "ROW : " + str(x) + "\t" +
            "LABEL : " + str(labeled[x][0]) + "\t" +
            "FLOAT : " + str(prediction[x][0]) + " \n")
    w.close()


def remove_noise(model, p, dataset, tag, min_noise, max_noise, reverse):
    """
    Remove noise of a model based on min_noise and max_noise range.
    the reverse option remove the data from outside (true) or inside (false) the min and max value

    Quentin Nater 2021


    :param model :              (1) Multilayers / CNN 1D (2)
    :param p :                  Path of the plots directory
    :param dataset :            Dataset object with all datasets (origin, data, label, patient, row_id,
burst_id)
    :param tag :                tag which gives indication of the job
    :param min_noise :          minimal limitation for the threshold
    :param max_noise :          maximal limitation for the threshold
    :param reverse :            Set if the noise elimination is outside or inside the min and max
indication


    :returns:   pred                Prediction array created to check the prediction ability /
                noise_elimination   final datasets with elimination (origin, data, label, patient, row_id,
burst_id)
                noise_id            final id dataset after noise elimination /
    """

    # Predict classification with the current trained model
    data = dataset.data
    label = dataset.label
```

```
    patient = dataset.patient
    ids = dataset.row_id
    bursts = dataset.burst_id
    pred = model.predict(data)

    # Create matrix for the ids and burst ids which will be considered as noise
    noise_id = numpy.zeros((1, 1))
    noise_burst_id = numpy.zeros((1, 1))

    file = p + "noise_id_removed.txt"
    # Create a file which contains all removed ids
    if tag == "REMOVE NOISE TEST":
        if os.path.exists(file):
            os.remove(file)

        f1 = open(file, "a")
        f1.write("==== REMOVED FROM TEST DATASET ================ \n")
        f1.close()

    # remove the data from outside (true) or inside (false) the min and max value
    if reverse:
        for i in range(len(pred)):
            value = pred[i][0]  # save the current value
            if min_noise < value < 0.5:
                pred[i][0] = 0  # healthy
            elif max_noise > value > 0.5:
                pred[i][0] = 1  # sick
            else:
                pred[i][0] = 2  # noise
    else:
        for i in range(len(pred)):
            value = pred[i][0]    # save the current value
            if value < min_noise:
                pred[i][0] = 0  # healthy
            elif value > max_noise:
                pred[i][0] = 1  # sick
            else:
                pred[i][0] = 2  # noise

    # remove useless values
    x, y = 0, 0
    for _ in pred:
        # select only noise value
        if pred[x][0] == 2:

            # Save information in the text file
            if tag == "REMOVE NOISE TEST":
                f1 = open(file, "a")
                f1.write("ID : " + str(ids[y][0]) + " \t PATIENT ID : " + str(patient[x][0]) + " \t LABEL :
" + str(
                    label[x][0]) + " \t BURST ID : " + str(bursts[y][0]) + " \n")
                f1.close()

            # delete the noise line in all datasets
            pred = numpy.delete(pred, x, 0)
            data = numpy.delete(data, x, 0)
            label = numpy.delete(label, x, 0)
            patient = numpy.delete(patient, x, 0)
            noise_id = numpy.insert(noise_id, noise_id.size, ids[y][0])
            noise_burst_id = numpy.insert(noise_burst_id, noise_burst_id.size, bursts[y][0])
            x = x - 1
        x = x + 1
        y = y + 1

    # Save all results in a Dataset object to simple the process
    noise_elimination = Dataset(None, data, label, patient, None, None)

    return pred, noise_elimination, noise_id


def evaluate_dataset(model, dataset, size, tag, display):
    """
    Set the accuracy based on the model, data, labels, the size of the batch and a comment

    Quentin Nater 2021


    :param model :              (1) Multilayers / CNN 1D (2)
    :param dataset :            Dataset object with all datasets (origin, data, label, patient, row_id,
burst_id)
    :param size :               size of the batch (should be 1)
    :param tag :                information about the job
    :param display :            boolean if you want to display information in the console about SE and
```

```
interval


    :returns:   history         keras array with all results of the evaluation /
                standard_err    unit used for the AUC /
                interval_err    unit used for the AUC
    """

    # Set the evaluation based on the current trained model
    history = model.evaluate(dataset.data, dataset.label, batch_size=size)

    standard_err, interval_err = 0, 0
    if display:
        # Calculation of the trust interval
        standard_err, interval_err = generate_standard_error(dataset.label, history[4], tag)

    return history, standard_err, interval_err


def model_save(model, p):
    """
    Save the trained model on the path p if the bool b is active

    Quentin Nater 2021


    :param model :  model to save
    :param p :      path where the plot is going to be saved
    """

    save_model(model, p)
    print("", "The model " + str(p) + " has been saved", sep="\n")


def model_load(p):
    """
    Load a trained model

    Quentin Nater 2021


    :param p :                  Path where the plot is going to be saved

    :return:    model :         Loaded model /
                model.history   keras array with all results of the evaluation
    """

    model = load_model(p, compile=True)
    return model, model.history


def create_directory(p):
    """
    Create a directory for plots and models

    Quentin Nater 2021


    :param p:   root path to save the model and plots
    """

    try:
        os.makedirs(p)
    except OSError:
        print("The directory %s was already created." % p)
    else:
        print("Successfully created the directory %s" % p)


def draw_single_plot(history, value, p, tag, level):
    """
    Draw a plot based on the history. Write the result on the path selected

    Quentin Nater 2021

    :param history :        Keras array with all results of the evaluation
    :param value :          Column of a keras array with all results of the evaluation
    :param p :              Path where the plot is going to be saved
    :param tag :            Information about the job
    :param level :          Current iteration
    """
    plot.plot(history[value])
    plot.title('Model ' + str(value) + ' - ' + tag)
```

```
    plot.ylabel(str(value))
    plot.xlabel('Epoch')
    plot.legend(['Train'], loc='upper left')
    plot.savefig(p + 'model_' + str(value) + '_' + str(level) + '.png')
    plot.show()


def draw_plots(history, p, tag, level):
    """
    Draw all necessary plots based on the history

    Quentin Nater 2021


    :param history :       Keras array with all results of the evaluation
    :param p :             Path where the plot is going to be saved
    :param tag :           Information about the job
    :param level :         Current iteration
    """

    # Get the name of the columns inside the history
    list_of_keys = list(history.keys())

    # Draw and save all useful metrics
    draw_single_plot(history, str(list_of_keys[1]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[0]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[2]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[3]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[4]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[7]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[8]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[5]), p, tag, level)
    draw_single_plot(history, str(list_of_keys[6]), p, tag, level)


def display_confusion_matrix(model, data, label, p, title, tag, display):
    """
    Draw a confusion matrix based on the model, the data (x) and the label (y). Write the result on the
path p

    Quentin Nater 2021


    :param model :      (1) Multilayers / CNN 1D (2)
    :param data :       dataset which contains the all data
    :param label :      dataset which contains the all labels
    :param p :          Path where the plot is going to be saved
    :param title :      Title of the plot
    :param tag :        Information about the job
    :param display :    Determine if a plot will be displayed


    :return: pred :      Save the prediction array for the next job
    """

    # set a prediction array with the current trained model
    pred = (model.predict(data) > 0.5).astype("int32")

    # set the matrix
    mat = confusion_matrix(label, pred)

    # build the matrix
    plot_confusion_matrix(conf_mat=mat, figsize=(8, 8), class_names=["Healthy", "Sick"], show_normed=True)

    # set the title of the matrix
    plot.title(title)

    # Save and display the result
    plot.savefig(p + '00_confusion_matrix' + tag + '.png')

    if display == "yes":
        plot.show()

    return pred


def evaluation_and_display_by_patient(label, pred, patient, tag):
    """
    Method which set statistics and information the patient in the goal to make a patient prediction

    Quentin Nater 2021
```

```
    :param label :          dataset which contains the all labels
    :param pred :           Saved array of the prediction of the model in the test dataset
    :param patient :        dataset which contains the all patient ids
    :param tag :            Information about the job
    """

    # initialization of the array of all patient ids and 6 columns
    prediction_by_patient = numpy.zeros((1000, 6))

    # Loop on the patient list and insert
    for x in range(len(patient)):
        # ID of the patient
        prediction_by_patient[patient[x][0]][0] = patient[x][0]

        # Total amount of bursts by patient
        prediction_by_patient[patient[x][0]][1] = int(prediction_by_patient[patient[x][0]][1]) + 1

        # Sum of bursts predicted as sick
        prediction_by_patient[patient[x][0]][2] = int(prediction_by_patient[patient[x][0]][2]) +
int(pred[x][0])

        # True label of the patient
        prediction_by_patient[patient[x][0]][5] = int(label[x][0])

    for x in range(len(prediction_by_patient)):
        # check if the patient exists in the array
        if prediction_by_patient[x][1] != 0:
            # Update the average result for a patient
            prediction_by_patient[x][3] = prediction_by_patient[x][2] / prediction_by_patient[x][1]

        # Check if the average is sick or healthy
        if prediction_by_patient[x][3] < 0.5:
            prediction_by_patient[x][4] = 0
        else:
            prediction_by_patient[x][4] = 1

    # Delete the empty patient
    x = 0
    for _ in prediction_by_patient:
        if prediction_by_patient[x][1] == 0:
            prediction_by_patient = numpy.delete(prediction_by_patient, x, 0)
            x = x - 1
        x = x + 1

    # display statistical results
    calc_correct_wrong(prediction_by_patient, tag, "patient")


def calc_correct_wrong(pred, label, tag):
    """
    Display the number of wrong and correct predictions inside the prediction array in comparison of the
label array

    Quentin Nater 2021


    :param pred :   Saved array of the prediction of the model in the test dataset
    :param label :  dataset which contains the all labels
    :param tag :    Information about the job
    """

    total, correct, wrong, noisy = 0, 0, 0, 0

    for i in range(len(pred)):
        if tag == "patient":
            # During a patient display, check if the prediction is the same as the expected result
            if pred[i][4] == pred[i][5]:
                correct = correct + 1
            else:
                wrong = wrong + 1
        else:
            # During a burst display, check if the prediction is the same as the expected label
            if label[i, 0] == pred[i]:
                correct = correct + 1
            elif pred[i] == 2:
                noisy = noisy + 1
            else:
                wrong = wrong + 1
        total = total + 1

    # Display of the results
    print()
    print("Total", tag, ":", str(total), sep="\t")
```

```
    print("Correct", tag, ":", str(correct), sep="\t")
    print("Wrong", tag, ":", str(wrong), sep="\t")

    if total != 0 and tag == "patient":
        print("right%", "patient", tag, ":", str((correct / total) * 100), "%", sep="\t")
        print("wrong%", "patient", tag, ":", str((wrong / total) * 100), "%", sep="\t")
    print()


def generate_standard_error(label, auc, tag):
    """
    Generate a Standard error and Interval error with a label dataset and an AUC

    Quentin Nater 2021


    :param label :          dataset which contains the all labels
    :param auc :            value of the area under the curve
    :param tag :            information about the job


    :returns:   se          standard error of the AUC /
                interval    interval error of the AUC
    """

    c_0, c_1 = 0, 0

    # Sum the amount of sick and healthy
    for i in range(len(label)):
        if label[i] == 0:
            c_0 = c_0 + 1
        elif label[i] == 1:
            c_1 = c_1 + 1

    # check if both amount are positives and process calculation of the standard error and interval
    if c_0 != 0 and c_1 != 0:
        auc2 = auc * auc
        q1 = auc / (2 - auc)
        q2 = 2 * auc2 / (1 + auc)
        se = auc * (1 - auc) + (c_0 - 1) * (q1 - auc2) + (c_1 - 1) * (q2 - auc2)
        se = se / (c_0 * c_1)
        se = math.sqrt(se)
        interval = 1.960 * se

        print("", tag + " SE       : " + str(se), tag + " Interval : " + str(interval), "", sep="\n")
        return se, interval
    else:
        print("", "Error C_0 or C_1 = 0 / Dataset without one class", "", sep="\n")
        return 0, 0


def handle_noise(model, options, threshold, test):
    """
    Handle the noise in test dataset with the threshold dataset

    Quentin Nater 2021


    :param model :              (1) Multilayers / CNN 1D (2)
    :param options :            Information about the model
    :param threshold :          Dataset object with threshold datasets (origin, data, label, patient,
row_id, burst_id)
    :param test :               Dataset object with test datasets (origin, data, label, patient, row_id,
burst_id)


    :returns:   noise_method    final noise elimination method calculated
                test            final test datasets after the noise elimination
                id              id removed after the noise elimination
    """

    # Set comparison information by removing noise out
    noise_reverse, history_noise_in, history_noise_out = False, None, None

    noise_min, noise_max, = 0.30, 0.70
    noise_out, without_noise_out, id_removed = remove_noise(model, options.path, threshold, "Remove Noise
OUT",
                                                            noise_min, noise_max, True)

    # Set comparison information by removing noise in
    noise_min, noise_max = 0.40, 0.60
    noise_in, without_noise_in, id_removed = remove_noise(model, options.path, threshold, "Remove " "Noise
IN",
```

```
                                                    noise_min, noise_max, False)

    # Evaluation of the AUC WITH noise
    history_threshold, standard_error, interval_error = evaluate_dataset(model, threshold, 1, "With noise",
False)
    auc_threshold = history_threshold[4]

    # Check if the size after the threshold elimination is still acceptable
    if without_noise_in.data.size > 1000:
        # Evaluation of the AUC without noise IN
        history_noise_in, standard_error, interval_error = evaluate_dataset(model, without_noise_in, 1,
"without IN",
                                                                            False)
        auc_in = history_noise_in[4]
    else:
        auc_in = 0
        print("", "Noise IN is too short to be used as test after noise elimination (" + str(
            without_noise_in.data.size) + ")", sep="\n")

    # Check if the size after the threshold elimination is still acceptable
    if without_noise_out.data.size > 1000:
        # Evaluation of the AUC without noise OUT
        history_noise_out, standard_error, interval_error = evaluate_dataset(model, without_noise_out, 1,
"without OUT",
                                                                             False)
        auc_out = history_noise_out[4]
    else:
        auc_out = 0
        print("", "Noise OUT is too short to be used as test after noise elimination (" + str(
            without_noise_out.data.size) + ")", sep="\n")

    # Set of checks to determine the better results (IN/OUT/NO)
    if auc_in < auc_out:
        if auc_threshold < auc_out:
            print("", "Noise OUT is up to this model", "AUC with noise......: " + str(auc_threshold),
                  "AUC IN..............: " + str(auc_in), "AUC OUT..............: " + str(auc_out),
sep="\n")
            noise_method = "out"
            noise_reverse = True
            noise_min, noise_max = 0.25, 0.75
        else:
            print("", "There are no noise to this model", "AUC with noise......: " + str(auc_threshold),
                  "AUC IN..............: " + str(auc_in), "AUC OUT..............: " + str(auc_out),
sep="\n")
            noise_method = "no"
    else:
        if auc_threshold < auc_in:
            print("", "Noise IN is up to this model", "AUC with noise......: " + str(auc_threshold),
                  "AUC IN..............: " + str(auc_in), "AUC OUT..............: " + str(auc_out),
sep="\n")
            noise_method = "in"
            noise_reverse = False
            noise_min, noise_max = 0.40, 0.60
        else:
            print("", "There are no noise to this model", "AUC with noise : " + str(auc_threshold),
                  "AUC IN : " + str(auc_in), "AUC OUT : " + str(auc_out), sep="\n")
            noise_method = "no"

    # Check if the size after the threshold elimination is still acceptable
    if without_noise_in.data.size > 1000:
        len_in = history_noise_in[5] + history_noise_in[6] + history_noise_in[7] + history_noise_in[8]
    else:
        len_in = 0

    if without_noise_out.data.size > 1000:
        len_out = history_noise_out[5] + history_noise_out[6] + history_noise_out[7] + history_noise_out[8]
    else:
        len_out = 0

    if noise_method == "in" and len_in < 1000:
        noise_method = "no"
        print("", "Noise IN is too short to be used as test after noise elimination (" + str(len_in) + ")",
sep="\n")

    elif noise_method == "out" and len_out < 1000:
        noise_method = "no"
        print("", "Noise OUT is too short to be used as test after noise elimination (" + str(len_out) +
")", sep="\n")

    # Check if the noise elimination if a good thing and remove the noise
    if noise_method != "no":
        noise, test, id_removed = remove_noise(model, options.path, test, "REMOVE NOISE TEST",
                                               noise_min, noise_max, noise_reverse)
```

```
        print("", "Noise " + str(noise_method) + " as been transform", sep="\n")

        f1 = open(options.ml + "model_noise_information.txt", "a")
        f1.write("the noise in this model is " + str(noise_method) + "\n")
        f1.close()

    return noise_method, test, id_removed


def crossvalidation(train, number):
    """
    Transform and prepare the different datasets for cross validation on the train dataset

    Quentin Nater 2021


    :param train :              Dataset object with train datasets (origin, data, label, patient, row_id,
burst_id)
    :param number :             Number of patient in the new cross validation test dataset


    :returns:   train           Dataset object with cross train datasets(origin, data, label, patient,
row_id, burst_id)
                new_dataset     Dataset object with cross test datasets (origin, data, label, patient,
row_id, burst_id)

    """

    # save the current patient dataset
    handle_patient = train.patient

    # Transform the array in DataFrame to shuffle it
    shuffle_patient = pd.DataFrame(train.origin)

    # Shuffling of the array
    shuffle_patient = shuffle_patient.sample(frac=1).reset_index(drop=True)

    # Creation of the new cross-validation test datasets
    crossvalidation_patient = numpy.zeros((1, 1), dtype=numpy.int64)
    crossvalidation_label = numpy.zeros((1, 1), dtype=numpy.int64)
    crossvalidation_row_id = numpy.zeros((1, 1), dtype=numpy.int64)
    crossvalidation_burst_id = numpy.zeros((1, 1), dtype=numpy.int64)
    random_patients = numpy.zeros(0, dtype=numpy.int64)

    # Pick random healthy patients with the shuffled array
    boundary, inc = 0, 0
    while boundary < int(number / 2):
        if int(shuffle_patient[3][inc]) == 0:
            random_patients = numpy.insert(random_patients, random_patients.size,
int(shuffle_patient[1][inc]), axis=0)
            boundary = boundary + 1
        inc = inc + 1

    # Pick random sick patients with the shuffled array
    boundary, inc = 0, 0
    while boundary < int(number / 2):
        if int(shuffle_patient[3][inc]) == 1:
            random_patients = numpy.insert(random_patients, random_patients.size,
int(shuffle_patient[1][inc]), axis=0)
            boundary = boundary + 1
        inc = inc + 1

    # Shuffle of the selected patient (convert to dataframe and convert back to panda array)
    random_patients = pd.DataFrame(random_patients)
    random_patients = random_patients.sample(frac=1).reset_index(drop=True)
    random_patients = random_patients.to_numpy()

    # Removing of the first line created in the initialization
    crossvalidation_patient = numpy.delete(crossvalidation_patient, 0, 0)
    crossvalidation_label = numpy.delete(crossvalidation_label, 0, 0)
    crossvalidation_row_id = numpy.delete(crossvalidation_row_id, 0, 0)
    crossvalidation_burst_id = numpy.delete(crossvalidation_burst_id, 0, 0)

    print("", "Cross validation patient randomly chosen : " + str(random_patients), "",
          "Creation of the new dataset... It could be long...", sep="\n")

    # Copy the current training data to the cross-validation data
    crossvalidation_data = train.data
    x, y, inc = 0, 0, 0

    # Progress bar to wait during the job
    for _ in tqdm(range(handle_patient.size), desc="Loading..."):
```

```
        value = handle_patient[x][0]

        # Check if the patient has been selected in the random process
        if value in random_patients:

            # Insert into the new cross test dataset
            crossvalidation_patient = numpy.insert(crossvalidation_patient, crossvalidation_patient.size,
                                                   int(handle_patient[x][0]), axis=0)

            crossvalidation_label = numpy.insert(crossvalidation_label, crossvalidation_label.size,
                                                 int(train.label[x][0]), axis=0)

            crossvalidation_row_id = numpy.insert(crossvalidation_row_id, crossvalidation_row_id.size,
                                                  int(train.row_id[x][0]), axis=0)

            crossvalidation_burst_id = numpy.insert(crossvalidation_burst_id,
crossvalidation_burst_id.size,
                                                    int(train.burst_id[x][0]), axis=0)

            # Remove into the old train dataset
            train.data = numpy.delete(train.data, x, 0)
            train.label = numpy.delete(train.label, x, 0)
            train.row_id = numpy.delete(train.row_id, x, 0)
            train.burst_id = numpy.delete(train.burst_id, x, 0)
            train.patient = numpy.delete(train.patient, x, 0)
            handle_patient = numpy.delete(handle_patient, x, 0)
            x = x - 1
        else:
            # Remove useless data into the cross-validation test dataset
            crossvalidation_data = numpy.delete(crossvalidation_data, y, 0)
            y = y - 1
        x = x + 1
        y = y + 1
        inc = inc + 1
    pass

    # Set the Dataset object with all different datasets to simply the process
    new_dataset = Dataset(None, crossvalidation_data, crossvalidation_label, crossvalidation_patient,
                          crossvalidation_row_id, crossvalidation_burst_id)

    return train, new_dataset


def test_evaluation_and_noise(model, options, threshold, test, display):
    """
    Manage the different noise jobs and test evaluation for training or testing models

    Quentin Nater 2021


    :param model :              (1) Multilayers / CNN 1D (2)
    :param options :            Information about the model
    :param threshold :          Dataset object with threshold datasets(origin, data, label, patient,
row_id, burst_id)
    :param test:                Dataset object with test datasets(origin, data, label, patient, row_id,
burst_id)
    :param display :            Determine is the plots will be displayed

    :returns:   y_pred :        Saved array of the prediction of the model in the test dataset /
                history_test :  keras array with all results of the evaluation /
                std_error :     standard error of the AUC /
                int_error :     interval error of the AUC /
    """

    # Noise Handler / Remove the noise from the test dataset
    print("", "= NOISES ==========================================================================", sep="\n")
    noise_result, test, id_removed = handle_noise(model, options, threshold, test)

    print("", "= Test ============================================================================", sep="\n")
    # Display and save the confusion matrix with the test dataset
    y_pred = display_confusion_matrix(model, test.data, test.label, options.path,
                                      "Confusion Matrix of Test", "test", display)

    # Generate keras results, standard error and interval error with the evaluation of the test dataset
    history_test, std_error, int_error = evaluate_dataset(model, test, 1, "test", True)

    # Display the statistic result of the test dataset
    calc_correct_wrong(y_pred, test.label, "Test")

    # Evaluate and display the statistic result of the patient prediction
    evaluation_and_display_by_patient(test.label, y_pred, test.patient, "Test")

    return y_pred, history_test, std_error, int_error, test.data
```

```
def generate_prediction(model, data_to_predict):
    """
    Generate the classification and the prediction based on data

    Quentin Nater 2021


    :param model:                   (1) Multilayers / CNN 1D (2)
    :param data_to_predict:         Data which is going to be predicted

    :returns:   labeled:            Classification array of the given dataset /
                prediction :        Float prediction array of the given dataset
    """

    # Create the array which will contain the classification
    labeled = numpy.zeros((1, 1), dtype=numpy.int64)

    # Start a Keras prediction
    prediction = model.predict(data_to_predict)

    # For each prediction, check if it is sick or healthy based on the value (0.5)
    for inc in tqdm(range(prediction.size), desc="Prediction..."):
        labeled = numpy.insert(labeled, labeled.size, (prediction[inc] > 0.5).astype("int32"), axis=0)

    # Delete the line used in the creation of the array
    labeled = numpy.delete(labeled, 0, 0)

    return labeled, prediction


def set_variables(context, model_type):
    """
    Set all variables necessary for the training and the testing of a model

    Quentin Nater 2021


    :param context:             determine if it's a training or a test
    :param model_type:          determine the type of the model (multilayer or cnn)


    :returns: opt               object with all information necessary to build or test a model
    """

    # Current date to generate unique directory name
    now = datetime.now()
    date = now.strftime("%d-%m-%Y-%H-%M-%S")

    # Initialize testing variables
    opt = Options(22, 'AdvancedDetectionEyeIllness' + context, 1, 1, 0, "", "", "", "-", "-", model_type)
    opt.directory = str(opt.experiment) + '-' + str(opt.tag) + '-' + str(model_type) + '-' + str(date)
    opt.path = './plots/' + opt.directory + '/'
    opt.ml = './ml/' + opt.directory + '/'
    opt.to_string()

    return opt


def test_model(model_type, model_directory, test_dataset, threshold_dataset, generate_plots="yes"):
    """
    Test any models already created as CNN 1D or as Multilayers Neural Network

    Quentin Nater 2021


    :param model_type: Type of the model : "multilayers" - MultiLayers / "cnn1d" CNN 1D
    :param model_directory: Name of the directory where the model has been saved
(../ml_official/model_directory)
    :param test_dataset: Path to the test dataset (CSV)
    :param threshold_dataset: Path to the threshold dataset (CSV)
    :param generate_plots: Determine if the plots will be created (default = yes)


    :returns: labeled           Classification array of the given dataset /
              prediction        Float prediction array of the given dataset /
              history_test      keras array with all results of the evaluation /
              std_error         standard error of the AUC /
              int_error         interval error of the AUC /
    """

    print("", "A model testing has been launched !", "", sep="\n")
```

```
    # Set all variables necessary to train a model
    opt = set_variables("TEST", model_type)

    # Load of the datasets
    test = load_dataset(model_type, test_dataset)
    threshold = load_dataset(model_type, threshold_dataset)

    # create the necessary directories for plots and models
    create_directory(opt.path)
    create_directory(opt.ml)

    # Load the model
    classifier, history = model_load(model_directory + "/")

    # Evaluation of the model and noise elimination
    y_pred, history_test, std_error, int_error, test.data = test_evaluation_and_noise(classifier, opt,
                                                                                       threshold, test,
generate_plots)

    # Generate the prediction array
    labeled, prediction = generate_prediction(classifier, test.data)

    # Save the result in a text file
    write_results_test(history_test, opt.path, opt.directory, std_error, int_error, "test")

    print("", "Test process over !", "", sep="\n")

    return labeled, prediction, history, history_test, std_error, int_error


def train_model(model_type, epochs, train_dataset, test_dataset, threshold_dataset,
                number_loops=1, cross_validation="yes", generate_plots="yes"):
    """
    Train CNN 1D or Multilayers Neural Network models

    Quentin Nater 2021


    :param model_type: Type of the model : "multilayers" - MultiLayers / "cnn1d" CNN 1D
    :param epochs: Number of iteration inside the training (Example : 60)
    :param train_dataset: Path to the train dataset (CSV)
    :param test_dataset: Path to the test dataset (CSV)
    :param threshold_dataset: Path to the threshold dataset (CSV)
    :param number_loops: number of time that the same model is created (default : 1)
    :param cross_validation: add cross validation option : "yes" with cross-validation / "no" without
(default = yes)
    :param generate_plots: Determine if the plots will be created (default = yes)


    :returns: labeled            Classification array of the given dataset /
              prediction         Float prediction array of the given dataset /
              history.history    keras array with all results of the training /
              history_test       keras array with all results of the evaluation /
              std_error          standard error of the AUC /
              int_error          interval error of the AUC /
    """

    print("", "A model training has been launched !", "", sep="\n")
    labeled, prediction, history, history_test, std_error, int_error = None, None, None, None, 0, 0

    number_loops = int(number_loops)
    for level_inc in range(0, number_loops, 1):

        # Set all variables necessary to test a model
        opt = set_variables("TRAIN", model_type)

        # Loading of the datasets
        train = load_dataset(model_type, train_dataset)
        test = load_dataset(model_type, test_dataset)
        threshold = load_dataset(model_type, threshold_dataset)

        # Set the shape of the training set
        n_timesteps, n_features, n_outputs = train.data.shape[1], 0, 0
        if model_type == "cnn1d":
            n_features, n_outputs = train.data.shape[2], train.label.shape[1]
            print('', 'features :' + str(n_features), sep="\n")
            print('', 'outputs :' + str(n_outputs), sep="\n")
        print('', 'timesteps :' + str(n_timesteps), sep="\n")

        if cross_validation == "yes":
            # ACTIVATE TO DO CROSS-VALIDATION
            train, test = crossvalidation(train, 25)
```

```
        # create the necessary directories for plots and models
        create_directory(opt.path)
        create_directory(opt.ml)

        classifier = None
        # creation of the Model / multilayers / cnn1d
        if model_type == "multilayers":
            classifier = model_multilayers(n_timesteps)
        elif model_type == "cnn1d":
            classifier = model_cnn1d(n_timesteps, n_features, n_outputs)

        # compile the model with all metrics
        classifier.compile(optimizer='sgd', loss='mse',
                           metrics=['accuracy', tf.Precision(), tf.Recall(), tf.AUC(), tf.TrueNegatives(),
                                    tf.FalseNegatives(), tf.TruePositives(), tf.FalsePositives()])

        # start the model and keep the history for charts
        history = classifier.fit(train.data, train.label,
                                 batch_size=opt.batch_size, epochs=int(epochs), verbose=opt.verbose)

        # save the model based on the need
        model_save(classifier, opt.ml)

        # display and write all plots
        if generate_plots == "yes":
            draw_plots(history.history, opt.path, opt.tag, level_inc)

        # Evaluation of the model and noise elimination
        y_pred, history_test, std_error, int_error, test.data = test_evaluation_and_noise(classifier, opt,
                                                                                           threshold, test,
                                                                                           generate_plots)

        # Generate the prediction array
        labeled, prediction = generate_prediction(classifier, test.data)

        # Save the result of the test in a text file
        write_results_test(history_test, opt.path, opt.directory, std_error, int_error, "test")

        # Save the result of the training in a text file
        write_results_train(history.history, opt.path, opt.directory, int(epochs), "train")

        print("", "Train process over !", "", sep="\n")

    print("", "FINAL train Process over !", "", sep="\n")

    return labeled, prediction, history.history, history_test, std_error, int_error


def prediction_data(model_type, model_directory, directory_filename):
    """
    Predict the classification based on data

    Quentin Nater 2021


    :param model_type: Type of the model : "multilayers" - MultiLayers / "cnn1d" CNN 1D
    :param model_directory: Name of the directory where the model has been saved
(../ml_official/model_directory)
    :param directory_filename: Name of the directory where the dataset has been saved
(../ml_official/model_directory)

    :returns:   labeled: Classification array of the given dataset /
                prediction : Float prediction array of the given dataset
    """

    print("", "A prediction of data has been launched !", "", sep="\n")

    print("", "Model Type.............: " + str(model_type), "Model Directory.......: " +
str(model_directory),
          "Directory filename....: " + str(directory_filename), "", sep="\n")

    # Load of the datasets
    data = load_data(model_type, directory_filename)

    classifier, history = model_load(model_directory)

    # Generate the prediction array
    labeled, prediction = generate_prediction(classifier, data)

    directory = "./prediction/"
    create_directory(directory)
    write_results_prediction(directory, labeled, prediction, model_type)
```

```
print("", "Prediction process over !", "", sep="\n")
return labeled, prediction
```

Source : Code de l'auteur (https://github.com/qnater/AdvancedDetectionEyeIllness)

Annexe V : Librairies, outils et dépendances

Librairies / Outils	Version
absl-py	0.13.0
astunparse	1.6.3
cachetools	4.2.2
charset-normalizer	2.0.2
flatbuffers	1.12
gast	0.4.0
google-auth	1.33.0
google-auth-oauthlib	0.4.4
google-pasta	0.2.0
grpcio	1.34.1
h5py	3.1.0
idna	3.2
joblib	1.0.1
keras-nightly	2.5.0.dev2021032900
keras-preprocessing	1.1.2
markdown	3.3.4
mlxtend	0.18.0
numpy	1.19.5
oauthlib	3.1.1
opt-einsum	3.3.0
panda	0.3.1
protobuf	3.17.3
pyasn1	0.4.8
pyasn1-modules	0.2.8
requests	2.26.0
requests-oauthlib	1.3.0
rsa	4.7.2
scikit-learn	0.24.2
six	1.15.0
sklearn	0.0
tensorboard	2.5.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.0
tensorflow	2.5.0
tensorflow-estimator	2.5.0
termcolor	1.1.0
threadpoolctl	2.2.0
tqdm	4.61.2
typing-extensions	3.7.4.3
urllib3	1.26.6
werkzeug	2.0.1

wrapt	1.12.1

Dépendances	Version
certifi	2021.5.30
openssl	1.1.1k
pip	21.1.3
python	3.8.10
setuptools	52.0.0
sqlite	3.36.0
vc	14.2
vs2015_runtime	14.27.29016
wheel	0.36.2
wincertstore	0.2

Outils	Version
PyCharm	2021.1
KNIME	4.3.2
Miniconda	4.9.2 (python 3.8)

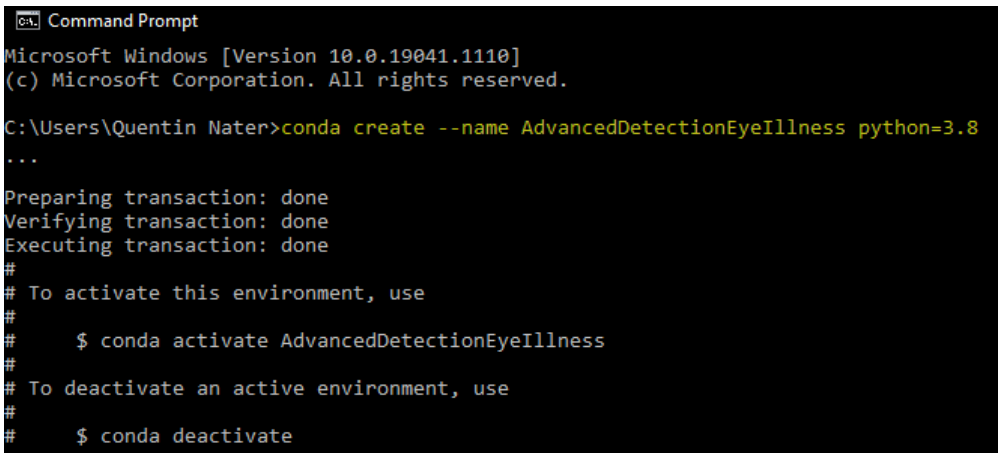
Source : Données de l'auteur

Annexe VI : Miniconda – Exportation de l'environnement

Voici les étapes à suivre pour créer et exporter un environnement :

1) Télécharger Miniconda[11] (voir annexe V, version 4.9.2).

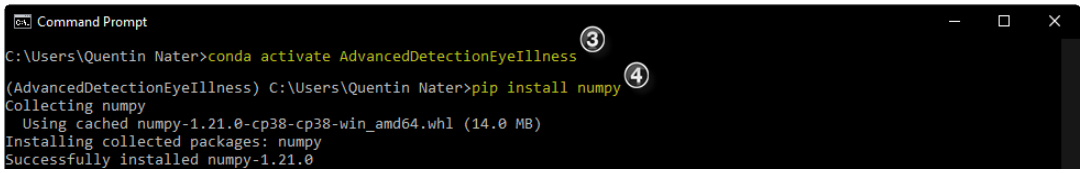
2) Une fois Miniconda installé, l'étape suivante consiste à créer un nouvel environnement dans l'invite de commande, avec la commande « *conda create --name $ENVIRONMENT_NAME python=$VERSION* ». La figure ci-dessous expose la création d'un environnement nommé « AdvancedDetectionEyeIllness ».



Source : Création d'un environnement Conda - Image de l'auteur

3) Une fois l'environnement créé, il faut s'y connecter avec la commande « *activate $ENVIRONMENT_NAME* », comme le montre le point 3 de la figure ci-dessous.

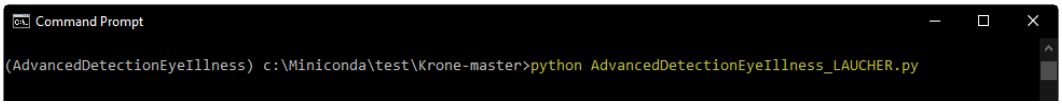
4) Il faut ensuite installer tous les paquets dépendants avec la commande : « *pip install $PACKAGE* », comme le montre le point 4 de la figure ci-dessous.



Source : Installation d'un package - Image de l'auteur

[11] Lien de téléchargement de Miniconda : https://docs.conda.io/en/latest/miniconda.html

5) Une fois que l'environnement est prêt, il est possible de lancer le programme avec la commande : « *python AdvancedDetectionEyeIllness_LAUCHER.py* ».



Source : Lancement du projet - Image de l'auteur

6) Une fois l'environnement testé, il reste à l'exporter dans un fichier avec la commande : « *conda env export > environment.yml* ».

Annexe VII : Miniconda - Importation de l'environnement

Voici la procédure d'importation d'un environnement Miniconda :

1) Télécharger Miniconda[12] (voir annexe V, version 4.9.2).

2) Télécharger le projet depuis GitHub[13].

3) Ajouter le dossier « dataset » à la racine du projet.

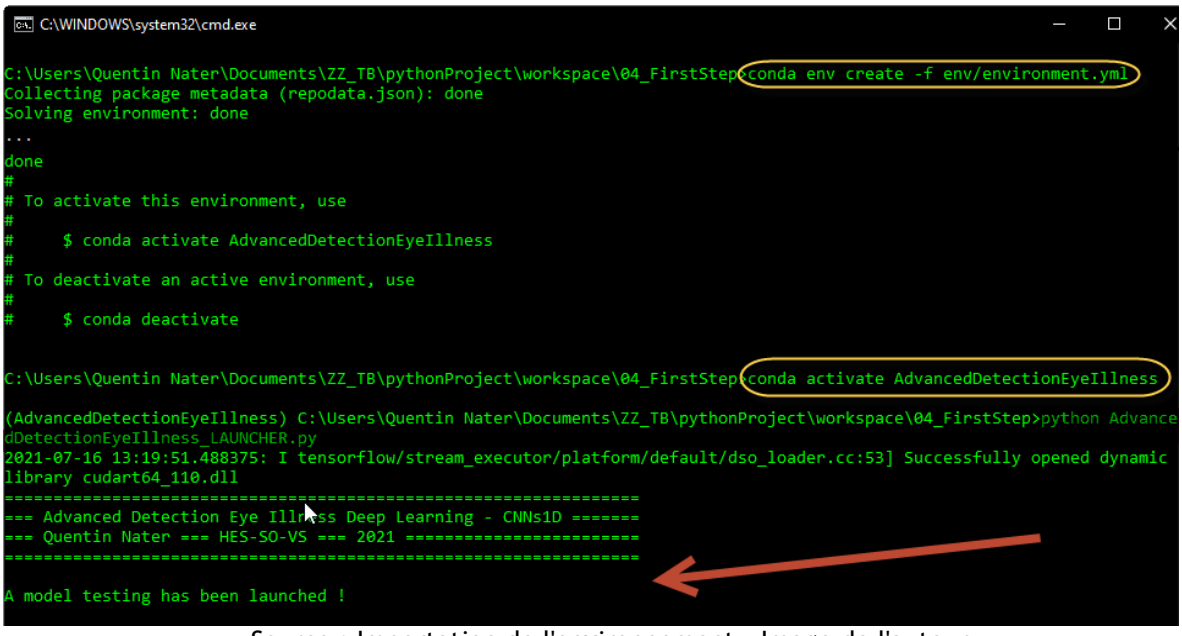
4) Ouvrir l'invite de commande de Windows et se diriger dans le dossier racine du projet.

5) Importer l'environnement avec la commande « *conda env create -f env/environment.yml* »

6) Activer l'environnement avec la commande « *activate AdvancedDetectionEyeIllness* ».

7) Lancer le script python avec la commande : « *python AdvancedDetectionEyeIllness_LAUCHER.py* »

La figure ci-dessous résume les actions nécessaires pour importer et exploiter un environnement.

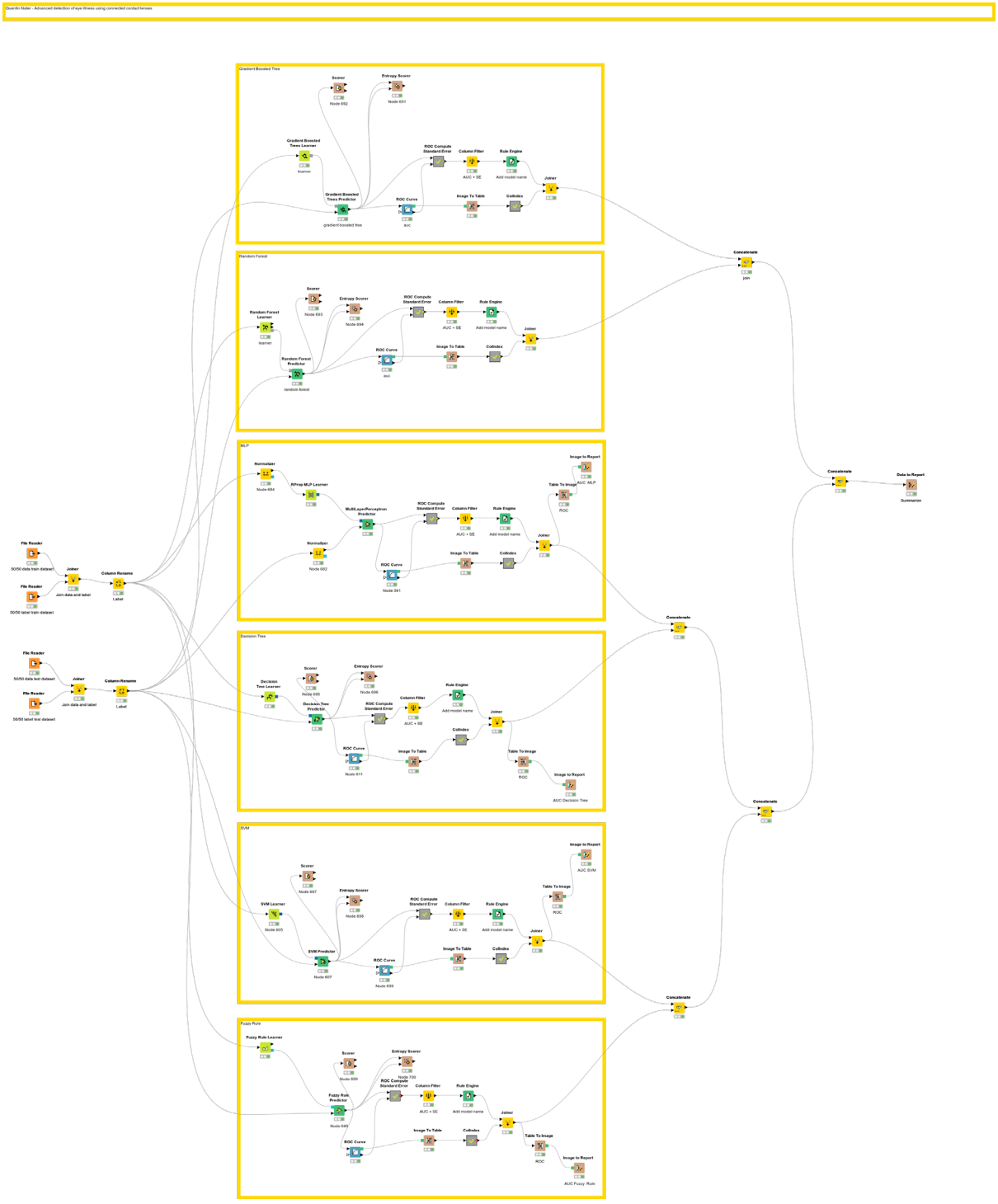


Source : Importation de l'environnement - Image de l'auteur

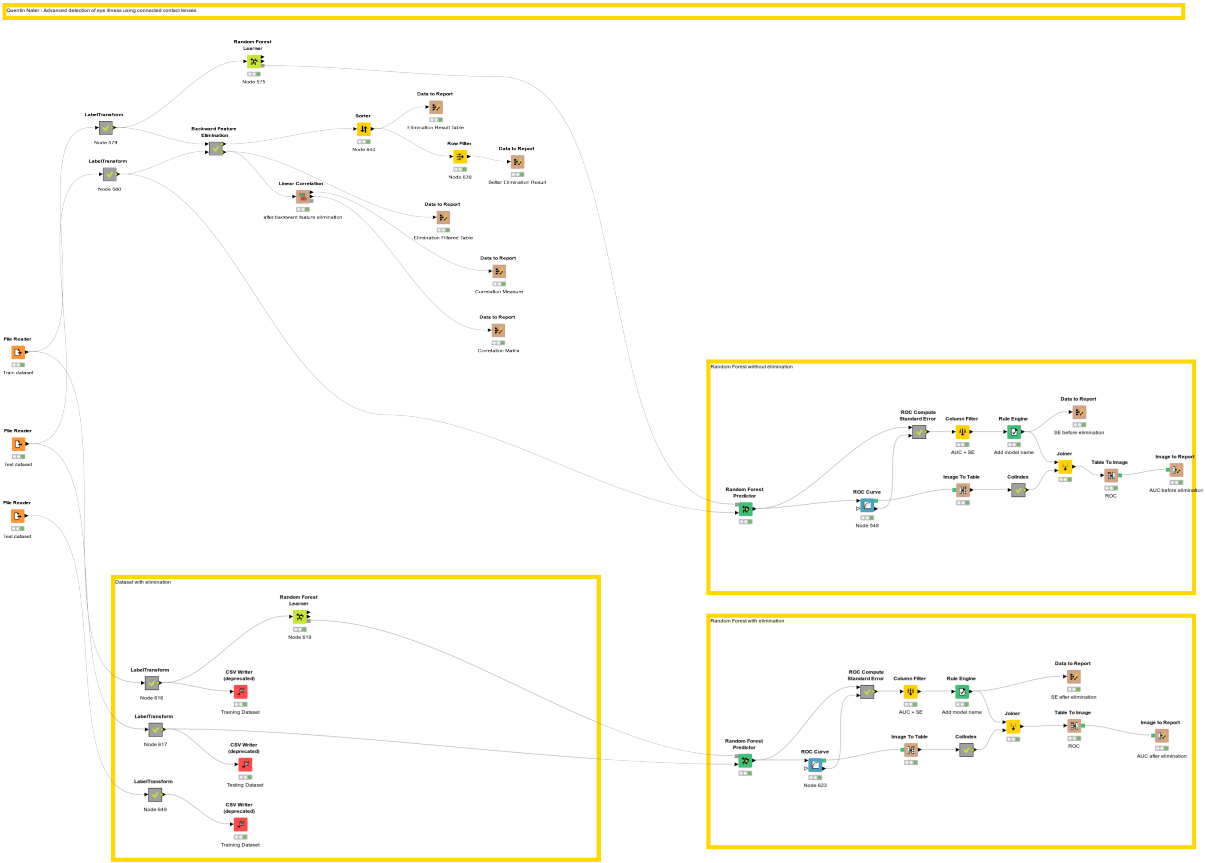
[12] Lien du projet GitHub : https://github.com/qnater/AdvancedDetectionEyeIllness

[13] Lien de téléchargement de Miniconda : https://docs.conda.io/en/latest/miniconda.html

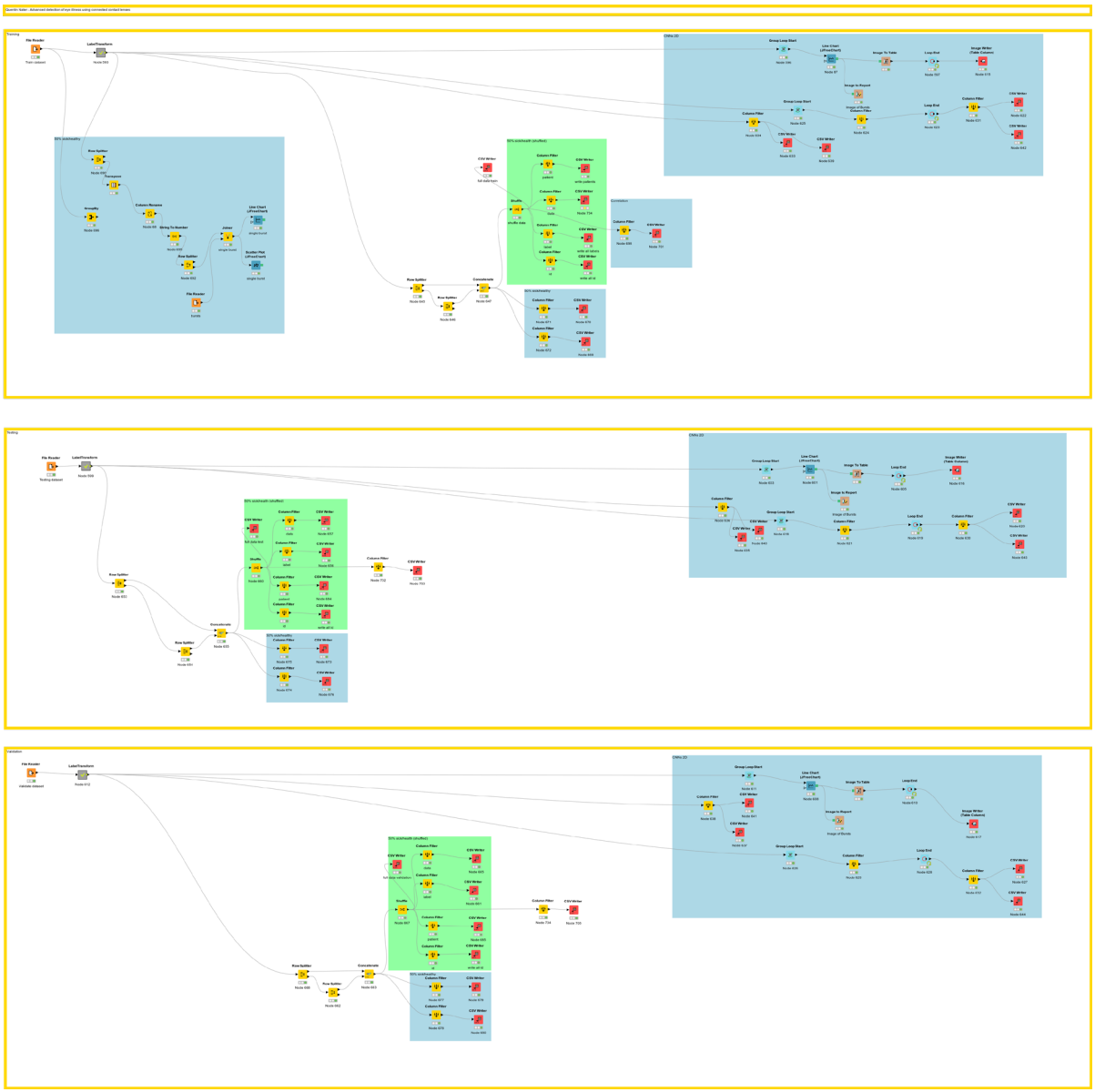
Annexe VIII : Travaux KNIME



Source : image de l'auteur



Source : image de l'auteur



Source : image de l'auteur

Annexe IX : Formules mathématiques

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicted}}\Big) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1 - \underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicted}}\Big)$$

Source : Formule mathématique de la Binary Crossentropy (Amini, 2021a)

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} \Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicted}}\Big)^2$$

Source : Formule mathématique de la MSE (Amini, 2021a)

Annexe X : Journal de bord

Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Date					362 heures	Details
27.02.2021	8	h -	13	h	5	Analyse de la demande et création des différents fichiers
27.02.2021	15	h -	18	h	3	Recherche et lecture d'articles universitaires sur le glaucome et sur le deep learning
28.02.2021	14	h -	17	h	3	Lecture d'articles universitaires sur le glaucome et sur le deep learning
28.02.2021	17	h -	21	h	4	Cours 1 MIT Deep learning
02.03.2021	8	h -	9	h	1	Réunion avec le client de Sensimed (Thierry)
06.03.2021	6	h -	14	h	8	Première analyse du dataset et entraînement des premiers algorithmes de classification
06.03.2021	14	h -	16	h	2	Ecriture des premiers rapport et séparation des datasets
09.03.2021	8	h -	9	h	1	Réunion avec le client (Sensimed)
13.03.2021	6	h -	12	h	6	Analyse des différents attributs et des différents dataset, avec précision des algorithmes de Random Forest
13.03.2021	14	h -	18	h	4	Analyse des résultats du matin et création du rapport
13.03.2021	18	h -	21	h	3	Cours 2 MIT Deep learning
13.03.2021	8	h -	9	h	1	Réunion avec le client
17.03.2021	8	h -	12	h	4	Mise en place du serveur et des fonctionnalités, création du workflow pour les boucles
17.03.2021	13	h -	17	h	4	Information et mise en place du workflow du Backward Feature Elimination
18.03.2021	8	h -	12	h	4	Analyse des résultats de l'algorithme de Backward Feature Elimination
18.03.2021	12	h -	13	h	1	Création d'un workflow pour comparer le meilleur algorithme de classification
18.03.2021	13	h -	15	h	2	Cours 3 MIT Deep learning
22.03.2021	9	h -	11	h	2	Vérification des rapports et corrections des algorithmes d'élimination et de classifications.
22.03.2021	11	h -	12	h	1	Réunion avec Jérôme (Institut)
27.03.2021	7	h -	9	h	2	Création d'un workflow pour la mise en application de la corrélation
27.03.2021	9	h -	10	h	1	Création du fichier qui va servir de rapport final à la thèse de Bachelor
27.03.2021	10	h -	12	h	2	Recherche sur l'État de l'art des Decision Trees et Random Forest
27.03.2021	14	h -	16	h	2	Rédaction de l'État de l'art des Decision Trees et Random Forest
30.03.2021	8	h -	9	h	1	Réunion avec le client de Sensimed
30.03.2021	9	h -	11	h	2	Installation et documentation sur TensorFlow
31.03.2021	9	h -	10	h	1	Réalisation du tutoriel deep learning : https://victorzhou.com/blog/intro-to-neural-networks/
31.03.2021	10	h -	12	h	2	Lecture de l'article : http://neuralnetworksanddeeplearning.com/chap1.html
31.03.2021	14	h -	15	h	1	Lecture de l'article : http://neuralnetworksanddeeplearning.com/chap1.html (2.0)
31.03.2021	15	h -	16	h	1	Installation de PyCharm et test des premiers scripts python
31.03.2021	16	h -	17	h	1	Lecture de l'article : http://neuralnetworksanddeeplearning.com/chap1.html (3.0)
02.04.2021	8	h -	9	h	1	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=njKP3FqW3Sk&ab_channel=AlexanderAmini
02.04.2021	9	h -	10	h	1	Machine Learning Zero to Hero https://www.youtube.com/watch?v=VwVg9jCtqaU&ab_channel=TensorFlow
02.04.2021	10	h -	11	h	1	Interprétation de l'image du patient dans Knime

02.04.2021	11	h -	12	h	**1**	Recherches théoriques et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=qjrad0V0uJE&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=2&ab_channel=AlexanderAmini
02.04.2021	13	h -	15	h	**2**	Essaie d'implémentation d'un réseau neuronal simple
02.04.2021	17	h -	18	h	**1**	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=AjtX1N_VT9E&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=3&ab_channel=AlexanderAmini
02.04.2021	18	h -	19	h	**1**	Essaie du tuto https://www.youtube.com/watch?v=_VTtrSDHPwU&ab_channel=TensorFlow
06.04.2021	7	h -	8	h	**1**	Customisation du code et test des différents algorithmes d'optimisation
06.04.2021	8	h -	9	h	**1**	Réunion de mise au point avec Jérôme
06.04.2021	10	h -	11	h	**1**	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=BUNl0To1IVw&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=5&ab_channel=AlexanderAmini
06.04.2021	11	h -	13	h	**2**	Test et recherche entre la loss et la précision
06.04.2021	14	h -	16	h	**2**	Premier essaie de confection d'un CNN 1D
07.04.2021	10	h -	12	h	**2**	Essaie du tutoriel de création d'un CNN : https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/
07.04.2021	14	h -	15	h	**1**	Analyse des résultats des loss/optimiser
07.04.2021	16	h -	17	h	**1**	Apprentissage de python, amélioration du code (print + fonction)
07.04.2021	18	h -	19	h	**1**	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=93M1l_nrhpQ&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=7&ab_channel=AlexanderAmini
07.04.2021	20	h -	23	h	**3**	Recherche des différences entre accuracy et loss / MSE et BCE / Préparation de la présentation
13.04.2021	7	h -	8	h	**1**	Préparation de la présentation Power Point et de la réunion avec les parties prenantes
14.04.2021	15	h -	16	h	**1**	Lecture de la thèse écrite par Olivier Arbellay (2020)
14.04.2021	16	h -	17	h	**1**	Écriture du chapitre "Glaucome" de la thèse
15.04.2021	10	h -	11	h	**1**	Documentation et implémentation d'une matrice de confusion https://www.youtube.com/watch?v=SToqP9V9y7Q&ab_channel=KGPTalkie
15.04.2021	11	h	12	h	**1**	Création de diagrammes : https://www.youtube.com/watch?v=SToqP9V9y7Q&ab_channel=KGPTalkie
15.04.2021	14	h -	15	h	**1**	Documentations et recherches à propos du Recall et de l'accuracy
15.04.2021	15	h -	16	h	**1**	Graphiques et métriques du Recall et analyse des résultats obtenus > l'IA met prédit chaque burst comme étant malade...
16.04.2021	15	h -	16	h	**1**	Ajout des métriques True Positives, True Negatives, False Positives, False Negatives et ajout des chemins dynamique dans le code
17.04.2021	8	h -	9	h	**1**	Création d'un nouveau dataset avec un équilibre de 50% de malades/sains
17.04.2021	9	h -	10	h	**1**	Installation d'un pilote graphique pour l'optimisation du GPU et l'entraînement des nouveaux réseaux de neurones avec le dataset équilibré à 50%.
17.04.2021	10	h -	11	h	**1**	Analyse des résultats et test de la différence avec ou sans la fonction d'activa RELU.
17.04.2021	11	h -	12	h	**1**	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=-boCMDouF2g&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=6&ab_channel=AlexanderAmini
17.04.2021	12	h -	13	h	**1**	Installation du NVDIA CUPDNN et test de large échelle pour l'expérimentation de 50% https://developer.nvidia.com/rdp/cudnn-download
17.04.2021	14	h -	15	h	**1**	Vérification de la valeur de la validation du RELU
17.04.2021	17	h -	18	h	**1**	Analyse du résultat et préparation du power point pour la réunion
22.04.2021	13	h -	14	h	**1**	Optimisation du code python avec git
22.04.2021	14	h -	15	h	**1**	Nouvelle tentative de construction un modèle CNN 1D
22.04.2021	15	h -	16	h	**1**	Recherche théorique et apprentissage sur le Deep Learning https://www.youtube.com/watch?v=toTcf7tZK8c&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=7&ab_channel=AlexanderAmini
24.04.2021	9	h -	10	h	**1**	Rédaction de la partie "Expérimentation 3" de la thèse
27.04.2021	7	h -	8	h	**1**	Préparation du power point pour la réunion
27.04.2021	8	h -	9	h	**1**	Réunion avec les parties prenantes
27.04.2021	17	h -	19	h	**2**	Nouvel essai pour construire un modèle CNN 1D
29.04.2021	7	h -	8	h	**1**	Création du dataset sur les patients et développement de la classification par patient
29.04.2021	9	h -	11	h	**2**	Aide de Jérôme pour comprendre les données pour les CNNs 1D

29.04.2021	11	h -	12	h	**1**	Configuration du CNN 1D (fonctionnel)
29.04.2021	13	h -	17	h	**4**	Rédaction de la partie "Deep Learning" de la thèse
01.05.2021	9	h -	11	h	**2**	Rédaction de la partie "Dataset" de la thèse
01.05.2021	11	h -	12	h	**1**	Rédaction de la partie "Knime" de la thèse
06.05.2021	7	h -	8	h	**1**	Correction de la rédaction
06.05.2021	8	h -	9	h	**1**	Corrélation avec le deep learning
06.05.2021	9	h -	11	h	**2**	Rédaction de l'expérimentation sur les réseaux de neurones multicouches
06.05.2021	11	h -	12	h	**1**	Rédaction de la partie méthodologie
06.05.2021	12	h -	14	h	**2**	Écriture et codage du CNN EXP7a 1D SIMPLE
06.05.2021	14	h -	15	h	**1**	Algorithmes pour prédire les patients malades
06.05.2021	17	h -	18	h	**1**	Boucles pour prédire les patients malades
08.05.2021	7	h -	8	h	**1**	Rédaction de la méthodologie
08.05.2021	8	h -	9	h	**1**	Expérimentation de la détection de patients malades avec le perceptron
08.05.2021	9	h -	10	h	**1**	Préparation de la réunion (PowerPoint)
08.05.2021	10	h -	12	h	**2**	Correction de la rédaction
08.05.2021	13	h -	16	h	**3**	Analyse des résultats de la classification par patients, des meilleurs réseaux multicouches et du réseau CNN 1D
11.05.2021	7	h -	8	h	**1**	Préparation du PowerPoint
11.05.2021	8	h -	9	h	**1**	Réunion avec les parties prenantes
14.05.2021	11	h -	12	h	**1**	Création d'une interface pour charger ou sauvegarder tous les modèles
20.05.2021	7	h -	9	h	**2**	Création d'un workflow avec le Gradient Boosted Tree et le Random Forest avec les nouveaux datasets
20.05.2021	9	h -	10	h	**1**	Analyse du résultat de Knime et création du power point
20.05.2021	11	h -	12	h	**1**	Création de la fonction remove_noise() pour pouvoir supprimer à volonté le bruit dans la prédiction
20.05.2021	14	h -	15	h	**1**	Test de l'algorithme avec les seuils de 0.45 et 0.55
21.05.2021	16	h -	17	h	**1**	Correction de la boucle for en une boucle for range(len())
21.05.2021	17	h -	18	h	**1**	Adaptation du code pour CNN et MultiLayer avec options Test et Train
22.05.2021	8	h -	12	h	**4**	Test et analyse du bruit sur le CNNs 1D
22.05.2021	13	h -	14	h	**1**	Préparation de la présentation
25.05.2021	7	h -	8	h	**1**	Préparation du power point pour la réunion
25.05.2021	8	h -	9	h	**1**	Réunion avec les parties prenantes
27.05.2021	8	h -	10	h	**2**	Préparation des algorithmes de Knime avec les nouveaux datasets
27.05.2021	10	h -	12	h	**2**	Calcul de l'erreur standard en Python et préparation de la présentation
29.05.2021	7	h -	10	h	**3**	Préparation du nouveau dataset donné par Sensimed
29.05.2021	10	h -	13	h	**3**	Test du modèle CNNs1D, MultiLayer, Random Forest et Gradient Boosted Tree avec le nouveau dataset donné par Sensimed.
29.05.2021	14	h -	16	h	**2**	Correction et analyse de la thèse. Ajout de la partie Gradient Boosted Tree et des références comme sources.
03.06.2021	8	h -	9	h	**1**	Test et validation du code
03.06.2021	9	h -	13	h	**4**	Rédaction de l'évaluation et des références de la thèse
03.06.2021	14	h -	17	h	**3**	Correction et rédaction de la thèse
08.06.2021	7	h -	8	h	**1**	Préparation de la présentation
08.06.2021	8	h -	9	h	**1**	Présentation avec les parties prenantes
09.06.2021	10	h -	11	h	**1**	Préparation du nouveau dataset donné par Sensimed sans covariable
09.06.2021	12	h -	15	h	**3**	Gestion du bruit avec les datasets de validation et de test
09.06.2021	15	h -	16	h	**1**	Vérification des CNNs 1D pour l'expérimentation 15
09.06.2021	9	h -	11	h	**2**	Création du tableau d'IDs pour chaque élimination de bruit et création d'un fichier avec tous les éléments supprimés.

09.06.2021	13	h -	14	h	**1**	Test avec le nouveau modèle
09.06.2021	14	h -	15	h	**1**	Information sur le bruit gaussien
11.06.2021	9	h -	10	h	**1**	Réunion question/réponse avec Jérôme
11.06.2021	10	h -	11	h	**1**	Correction de la thèse
16.06.2021	8	h -	11	h	**3**	Rédaction de la méthodologie et de l'état des connaissances
16.06.2021	11	h -	12	h	**1**	Théorie sur les réseaux neuronaux
17.06.2021	9	h -	10	h	**1**	Réunion avec Jérôme au sujet de la thèse
17.06.2021	10	h -	12	h	**2**	Création d'un script capable d'effectuer une validation croisée (exécuter 10 modèles en un seul coup)
18.06.2021	8	h -	10	h	**2**	Rédaction du choix technologique
20.06.2021	16	h -	17	h	**1**	Vérifier et exécuter la cross-validation (2)
22.06.2021	7	h -	8	h	**1**	Préparation de la présentation et analyse de la validation croisée
22.06.2021	8	h -	10	h	**2**	Réunion avec les parties prenantes
23.06.2021	8	h -	12	h	**4**	Création du code pour de multiples datasets
23.06.2021	13	h -	14	h	**1**	Vérification du nouveau dataset avec tous les types d'algorithmes.
23.06.2021	14	h -	16	h	**2**	Rectification de la thèse
28.06.2021	9	h -	10	h	**1**	Réunion sur la validation croisée
30.06.2021	8	h -	12	h	**4**	Algorithme de cross-validation
30.06.2021	13	h -	17	h	**4**	Algorithme de cross-validation et console avec un seul dataset en entrée à la place de 5
05.07.2021	6	h -	12	h	**6**	Algorithme de cross-validation (test and analyse) - Problème avec l'ensemble de teste
05.07.2021	13	h -	19	h	**6**	Rédaction de la thèse et confirmation de la validation croisée
06.07.2021	7	h -	9	h	**2**	Préparation de la présentation
06.07.2021	9	h -	10	h	**1**	Présentation avec les parties prenantes
13.07.2021	15	h -	16	h	**1**	Préparation des données de la prédiction du nouveau dataset
14.07.2021	9	h -	12	h	**3**	Développement de la méthode de prédiction
14.07.2021	13	h -	17	h	**4**	Nettoyage du code et exécution de tous les processus
15.07.2021	14	h -	15	h	**1**	Correction d'une boucle dans la détection du bruit
16.07.2021	7	h -	12	h	**5**	Installation et exportation de l'outil Conda
16.07.2021	13	h -	18	h	**5**	Correction de la thèse et rédaction du chapitre de déploiement
17.07.2021	6	h -	12	h	**6**	Correction de la thèse et rédaction de la conclusion
17.07.2021	13	h -	17	h	**4**	Test du code et préparation pour le partager avec le client
17.07.2021	17	h -	18	h	**1**	Adaptation du code pour ajouter des paramètres par défauts, comme la création ou non de graphiques
17.07.2021	18	h -	19	h	**1**	Adaptation du code pour ajouter des paramètres par défauts
20.07.2021	6	h -	9	h	**3**	Préparation du paquet pour le client et de la présentation
20.07.2021	8	h -	12	h	**4**	Correction de la thèse et des annexes
20.07.2021	13	h -	20	h	**7**	Correction de la thèse (partie résultat et conclusion)
21.07.2021	7	h -	12	h	**5**	Correction de la thèse (Réécriture du choix technologique)
21.07.2021	13	h -	17	h	**4**	Correction de la thèse (partie de l'expérimentation)
22.07.2021	6	h -	13	h	**7**	Correction du code pour le client (code smell et le dataset de test dans les prédictions)
24.07.2021	7	h -	12	h	**5**	Correction de la thèse (relecture générale)
24.07.2021	13	h -	17	h	**4**	Correction de la thèse (relecture générale)
28.07.2021	13	h -	02	h	**13**	Relecture de la thèse avec une personne tierce
03.08.2021	8	h -	12	h	**4**	Ajout des classes objets Dataset et Options qui rendent le code plus propre
03.08.2021	13	h -	16	h	**3**	Tests finaux

03.08.2021	16	h -	17	h	**1**	Tests finaux
04.08.2021	8	h -	12	h	**4**	Optimisation du code et déploiement
04.08.2021	13	h -	23	h	**10**	2ème correction de la thèse
05.08.2021	10	h -	12	h	**2**	2ème correction de la thèse (suite)
06.08.2021	07	h -	12	h	**5**	2ème correction de la thèse (détails des figures et liens)
06.08.2021	15	h -	19	h	**4**	Relecture avec une personne tierce
07.08.2021	10	h -	13	h	**3**	Correction de la relecture
07.08.2021	15	h -	17	h	**2**	Fin de la relecture avec une personne tierce
07.08.2021	17	h -	21	h	**4**	Fin de la correction de la relecture
09.08.2021	7	h -	13	h	**6**	Préparation du rapport final (rendu LAN/papier/Urkund)
10.08.2021	8	h -	9	h	**1**	Rendez-vous avec l'imprimeur

Source : données de l'auteur

Annexe XI : Product Backlog

ID	A faire	Temps *(heures)*	Priorité	Status	MosSCow
7	Rédaction de l'introduction, de l'état de l'art et du choix technologique	30	1000	3	Must
17	Rédaction de l'expérimentation	20	990	3	Must
21	Rédaction de la conclusion	10	980	3	Must
23	Implémentation du poster	2	970	3	Must
22	Test du code avec le client, correction de la thèse	10	950	3	Must
1	Analyse du dataset depuis KNIME	10	750	3	Must
2	Modélisation des différents modèles de classification traditionnels depuis KNIME	15	700	3	Must
11	Modélisation d'un modèle CNNs 1D	25	690	3	Must
12	Optimisation d'un modèle CNNs 1D	20	680	3	Must
8	Modélisation d'un réseau de neurones multicouches	15	660	3	Must
9	Optimisation d'un réseau de neurones multicouches	15	650	3	Must
4	Analyse et conclusion sur le résultat de la modélisation KNIME	2	640	3	Must
13	Test et analyse d'un réseau d'un modèle CNNs 1D	3	630	3	Must
10	Test et analyse d'un réseau de neurones multicouches	3	620	3	Must
19	Nettoyage et altération du code en vue du déploiement	10	600	3	Must
3	Affinement du dataset (élimination et corrélation)	20	500	3	Should
16	Limitation de l'impact du bruit	30	450	3	Should
14	Calcul de la Standard Error et de l'Interval Error	3	400	3	Should
5	Recherche et apprentissage du Deep Learning	30	200	3	Should
6	Recherche et apprentissage de python, Keras et TensorFlow	10	175	3	Should
15	Classification par patient	5	175	3	Could
18	Mise en place de la cross-validation et des itérations	15	100	3	Could
20	Environnement Miniconda	5	50	3	Could

Source : données de l'auteur

Annexe XII : Procès-Verbaux

Quentin Nater

PV : Advanced Detection Eyelllness
TB 2021 – Séance 01

PV de la séance du mardi 02 mars 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Thierry Varidel	Présent

Ordre du jour
Point de situation classification ; Modèle de classification Knime

Contenu de la séance

- Présentation des résultats.
  - Présentation des rapports Knime sur les différents algorithmes de classification.
  - Informations complémentaires sur les métriques de statistiques.
- Information complémentaire sur le déroulement du projet.
- Mise au propres des modèles de classification et vérification des résultats.
- Utilisation avec/sans covariate.
- Conseil sur la rédaction de la thèse.
  - Toujours garder une trace de toutes les références.

- Prochaine séance : Mardi 16 mars 2021 : 08h30

Rédaction : Quentin Nater, Mardi 02 mars 2021

Quentin Nater

PV : Advanced Detection Eyelllness
TB 2021 – Séance 02

PV de la séance du mardi 16 mars 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Non-Présent
Thierry Varidel	Présent

Ordre du jour
Point de situation classification ; Classification des modèles de machine learning

Contenu de la séance

- Présentation des résultats.
- Élimination de certains modèles et justification des résultats
- Problème lié à la taille du dataset :
  - Implémentation d'une méthode d'élimination et de corrélation dans Knime afin de limiter les données du dataset.
  - Utilisation du serveur Knime de l'institut de recherche
- Conseil de recherche : Information à propos des arbres de décision
- Prochaine séance
  - Mardi 30 mars 2021

Rédaction : Quentin Nater, Mardi 16 mars 2021

Quentin Nater

PV : Advanced Detection Eyelllness
TB 2021 – Séance 03

PV de la séance du mardi 30 mars 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Non-Présent
Thierry Varidel	Présent

Ordre du jour
Point de situation classification ; Backward Feature Elimination & Linear Correlation

Contenu de la séance

- Présentation des résultats.
- Prise de position de Thierry : **sans covariate**.
- Explication des bursts
  - Chaque point est une médiane des données.
  - Essayer d'apprendre des caractéristiques.
  - Apprendre du signal.
- Direction du Deep Learning
  - Utilisation de Keras.
  - Mise à disposition d'un ordinateur avec 2 GPU.
  - Analyse d'une image 1D, d'une image 2D ou d'un signal.
  - Utilisation de python ou Knime.
  - Utilisation d'outils comme TensorFlow.
  - Références pour l'entraînement.
    - http://neuralnetworksanddeeplearning.com/chap1.html
    - https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf
    - https://victorzhou.com/blog/intro-to-neural-networks/
- Prochaine séance
  - 13.04.2021 -> Position Deep Learning.
  - 06.04.2021 -> Mise au point.
- Information rapport
  - Ajout de la méthodologie.
  - Écrire le plus rapidement possible le plus d'informations, garder une trace.
  - Pourquoi python, Knime, Deep Learning ?
  - Anonymisation des données.

Rédaction : Quentin Nater, Mardi 30 mars 2021

Quentin Nater

PV : Advanced Detection Eyelllness

TB 2021 – Séance 04

PV de la séance du mardi 06 avril 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Non-Présent
Thierry Varidel	Non-Présent

Ordre du jour

Point de situation classification ; Deep Learning

Contenu de la séance

- Vérifier lorsque les epoch commencent à ne plus s'améliorer.
- Tester le résultat avec le set de validation.
- RNN
  - Pour la vidéo.
  - Tester pour le projet.
- Normaliser les âges
  - Détecter la tranche d'âge et pas la maladie.
- Temps d'entraînement doit être prise en compte.

Rédaction : Quentin Nater, 06 avril 2021

Quentin Nater

PV: Advanced Detection Eyelliness
TB 2021 – Séance 05

PV de la séance du mardi 27 avril 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Non-Présent
Thierry Varidel	Présent

Ordre du jour
Point de situation classification; Deep Learning 3.0 – Precision and recall

Contenu de la séance

- Présentation des résultats.
- CNNs 1D
  - CNNs1D – bust par burst (decision)
  - CNNs1D – Fusionner tous les patients + decision tree
  - CNNs1D – Mettre à 0 les données manquantes pour compenser la perte de taille
- 1 point avant RNN / tous les inputs CNN
- Implémenter un vote majoritaire pour la détection d'un patient.
- Courbe classique de données 1 à 288
- A jouer avec le 0.5 de la prédiction, discuter avec le spécialiste.
  - Pas d'avis sur l'importance de la valeur
    - Ni trop pencher sur le sain
    - Ni trop pencher vers le malade
- CNNs 1D – rectangle vertical.
- Montrer l'évolution de l'AUC
  - Mais via ROC / validation

- A faire ; CNNs1D / Vote majoritaire
- Thèse
  - Chapitre entier sur le dataset
    - EXP01 – J'ai fait ça, pour ça, ma conclusion, conclusion générale

- Prochain meeting 11.05.2021 / 08h30
- Prochaine visite 29.05.2021 / 09h00 / Technopole 1er étage (à gauche)

Rédaction : Quentin Nater, 27 avril 2021

Quentin Nater

PV: Advanced Detection Eyelllness
TB 2021 – Séance 06

PV de la séance du mardi 11 mai 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Thierry Varidel	Présent

Ordre du jour
Point de situation classification; Deep Learning 4.0 – CNNs 1D + Patient classification

Contenu de la séance

- Présentation des résultats.
- Conda offrirait un meilleur résultat que pip pour les fichiers binaires
- Pattern intra-bursts
- Covariates – 5% des patients malades
- Quand on regarde l'œil, les pulsations oculaires ressemblent à un sinus.
  - Réflexion artériel ?
  - Filtrage basse-fréquence.
  - Clignement d'œil, pointe des bursts.
  - Oscillation > respiration (effet pression de l'œil) ?
  - Deep Learning, laisser faire les paramètres.
  - Les hypothèses d'entrée sont importantes pour limiter le bruit.
- Basse fréquence : respiration
- Moyenne fréquence : mouvement de l'œil
- Début de la 2ème itération, réflexion sur la problématique.

- Prochain meeting Lundi 17.05 / 15h00
- Prochaine mise au point Mardi 25.05 / 08h30

Rédaction : Quentin Nater, 11 mai 2021

Quentin Nater

PV: Advanced Detection Eyelllness

TB 2021 – Séance 07

PV de la séance du mardi 17 mai 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Thierry Varidel	Non-Présent

Ordre du jour

Point de situation classification; Deep Learning 5.0

Contenu de la séance

- Gradient boosting tree - Knime
- Relancer avec les mêmes datasets - Knime
- Prendre que les bursts aux décisions élevées (enlever du bruit)
- Risque de détecter autre chose que la maladie
- Plage pour supprimer les valeurs proches de 0.5
  Décider que ceux qui sont entre 0.45 et 0.55 ne sont pas utilisés.
  Dans les tests

- Prochaine mise au point Mardi 25.05 / 08h30

Rédaction : Quentin Nater, 17 mai 2021

Quentin Nater

PV: Advanced Detection Eye Illness

TB 2021 – Séance 08

PV de la séance du mardi 25 mai 2021 Teams

Participants

Nom Prénom	Présence
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Non-Présent
Thierry Varidel	Présent

Ordre du jour

Point de situation classification; Deep Learning 7.0

Contenu de la séance

- Présentation des résultats.
- Les données devraient être non-définies et pas supprimées, car le dataset de test de serait plus identique.
- La standard error devrait être calculée pour le CNNs 1D.
- Partir sur un système de 3 classes à la place de la suppression.
- Le temps d'entraînement n'est pas un facteur dans le domaine de la médecine.

- Prochaine mise au point Mardi 08.06 / 08h30

Rédaction : Quentin Nater, 25 mai 2021

Quentin Nater

PV: Advanced Detection Eye Illness
TB 2021 – Séance 09

PV de la séance du mardi 08 juin 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Varidel Thierry	Présent

Ordre du jour
Point de situation classification; Deep Learning 8.0

Contenu de la séance

- Présentation des résultats.
- Les résultats du CNNs1D expliquent un cas de figure qui n'a pas assez de données ou qui est trop complexe.
- Pour éliminer le bruit, il faut utiliser un dataset neutre qui va permettre de délimiter les seuils de bruit des prédictions.
- Pour l'interface, il faut un fichier python ou une fonction, pouvoir avoir le résultat d'un seul burst.
- CNNs1D > Vérifier les valeurs du résultat.
- Une liste des tâches avec les heures requises sera rédigée par rapport aux résultats des prochaines semaines.

- Tâches des prochaines semaines
  - CNNs1D avec le nouvel dataset
    - Sans Covariate
  - 3 classes avec le dataset en plus.
  - Sur les 24 heures, est-ce qu'il y a des corrélations avec les bursts mauvais et l'heure (l'ID du Burst).

- Prochaine mise au point mardi 22.06 / 09h00

Rédaction : Quentin Nater, 08 juin 2021

Quentin Nater

PV: Advanced Detection Eye Illness

TB 2021 – Séance 10

PV de la séance du mardi 22 juin 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Varidel Thierry	Présent

Ordre du jour

Point de situation classification; Deep Learning 9.0

Contenu de la séance

- Présentation des résultats.
- Gaussian noise : Enlever les biais dû à la normalisation.
- A faire :
  - Meilleur entraînement du dataset normal avec le test du nouveau dataset.
  - Test les autres algorithmes
  - Cross Validation

- Prochaine mise au point mardi 06.07 / 09h30

Rédaction : Quentin Nater, 22 juin 2021

Quentin Nater

PV: Advanced Detection Eye Illness

TB 2021 – Séance 11

PV de la séance du mardi 06 juillet 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Varidel Thierry	Présent

Ordre du jour

Point de situation classification; Deep Learning 10.0

Contenu de la séance

- Présentation des résultats.
- Idée + analyse statistique -> Knime
- Envoyer les patients du dataset d50.

- A faire :
  - Fabrication des méthodes console et nettoyage du code
  - Chiffres statistiques sur le nouveau dataset
  - Écriture de la thèse
  - Envoyer les patients du d50

- Prochaine mise au point mardi : **20.07 / 09h30**
- Date butoir pour la thèse : **26.07.2021**

Rédaction : Quentin Nater, 06 juillet 2021

Quentin Nater

PV: Advanced Detection Eye Illness

TB 2021 – Séance 12

PV de la séance du mardi 20 juillet 2021 Teams

Participants

Nom Prénom	**Présence**
Nater Quentin	Présent
Treboux Jérôme	Présent
Genoux Dominique	Présent
Varidel Thierry	Présent

Ordre du jour

Point de situation classification; Deep Learning 11.0

Contenu de la séance

- Présentation des résultats.
- Analyse de Thierry sur le nouveau dataset : modèle très spécifique, mais pas sensible
- Le set présente un biais
  - Personnes différentes ?
  - Mesures différentes ?
    - Les données du dataset d'avèrent plus vieilles que les autres
    - Investigation de Sensimed en amont du projet

Rédaction : Quentin Nater, 20 juillet 2021

Quentin Nater

PV: Advanced Detection Eye Illness
TB 2021 – Séance 13

PV de la séance du mardi 28 juillet 2021 Teams

Participants	
Nom Prénom	**Présence**
Nater Quentin	Présent
Varidel Thierry	Présent

Ordre du jour
Point de situation sur le code – Beta Test

Contenu de la séance

- Tout le code a fonctionné du premier coup chez le client.
- Question du client :
  - Fichier de sauvegarde différent du modèle dû à la version (v2.5.0)
    - Sur la nouvelle version de TensorFlow, un fichier keras_metadata.pb est nécessaire.
  - Question sur le bruit.
    - Précision sur la décision de certains points du code
    - Mise en place d'un fichier pour définir le bruit par modèle
    - Retour du dataset de test pour les prédictions.
  - Question sur les modèles.
    - Question du client sur la raison de mes choix et précision sur les couches du CNNs 1D et du MultiLayer
  - Nom de fonction un peu confus.
    - Certains noms de méthodes sont assez confus (par exemple ceux des graphiques)
  - Question sur la construction du modèle.
    - Information sur les métriques, les fonctions de perte et d'optimisation
  - Paramètres aux fonctions, code smell
    - Vérifier le PEP Python contre le code smell
    - Utilisation de panda conseillée

Rédaction : Quentin Nater, 28 juillet 2021

HES-SO Valais-Wallis • rue de la Plaine 2 • 3960 Sierre
+41 58 606 89 11 • info@hevs.ch • www.hevs.ch

Source : images de l'auteur

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Dr Dominique Genoud
- M. Jérôme Treboux
- M. Thierry Varidel

Sierre, le 10 août 2021

Quentin Nater