الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

Faculté : Sciences de L'ingéniorat جامعة باجي مختار عنابـــة

Département : Informatique

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Gestion et analyse des données massives

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème :

Classification Multi Label des catégories fonctionnelles des gènes de levures

Présenté par : *Kissoum Sourour*

Encadrant : Mohamed Benali Yamina *Grade : Pr* *UBMA*

Jury de Soutenance :

Azizi Nabiha	Pr	UBMA	Président
Mohamed BenAli Yamina	Pr	UBMA	Encadrant
Zenakhra Djamel	MAA	UBMA	Examinateur

Année Universitaire : 2020/2021

Remerciement

Ce projet de fin d'étude, n'avait jamais été possible sans l'orientation et le soutiende mon encadreur, alors je tiens à exprimer ma gratitude et reconnaissance à Mme MOHAMED BENALI YAMINA pour son suivi depuis le début jusqu'à la fin dece mémoire.

Je tiens à exprimer mes sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Mes remerciements aux différents membres du jury spécialement à l'examinateurd'avoir accepté l'évaluation de ce travail.

Mes remerciements les plus sincères s'adressent aussi à mes parents, mes frères et mes proches pour leurs encouragements et leurs efforts durant toute l'année.

Dédicaces

Je dédie ce modeste travail à ma famille avec tous mes sentiments de respect, d'amour, de gratitude et de reconnaissance pour tous les sacrifices déployés pour m'élever dignement et assurer mon éducation dans les meilleures conditions.

A ma mère et mon père pour l'éducation qu'ils m'ont prodigué ; avec tous les moyens et au prix de toutes les sacrifices qu'ils ont consentis à mon égard, pour le sens du devoir qu'ils mon enseigné depuis mon Enfance.

À mes chers frères Newfel et Haitem qui m’ont encouragé sans cesse et cru en moi.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur : machers cousine Mayada.

Je clos ce dédicace en dédiant ce travail au peu d'amis que j'ai eus la chance d'avoir les connaître en particulier : mes amies Maya, Linda, Hiba et chaima.

À tous les étudiants de ma promotion en Gestion et Analyse de Données Massives.

Résumé

La classification multi-label est une extension de la classification traditionnelle dans laquelle les classes ne sont pas mutuellement exclusives, chaque exemple pouvant appartenir à plusieurs classes simultanément. Ce type de classification est requis par un grand nombre d'applications actuelles telles que la classification de texte, d'images et l'annotation de vidéos. Elle est très utilisée dans le domaine de la bioinformatique, par exemple la classification des gènes de levures.

Selon la littérature, plusieurs approches avec différentes stratégies ont été développées pour résoudre les problèmes de l'apprentissage Multi-Label. Ces approches ont été organisées en trois grandes familles, à savoir : les approches par transformation, les approches par adaptation et les approches d'ensemble. La première catégorie de méthodes représente des algorithmes indépendants, qui transforment le problème d'apprentissage Multi-Label en un ou plusieurs problèmes de classification ou de régression mono-label. La deuxième catégorie de méthodes adapte des algorithmes traditionnels pour des données Multi-Label. Tandis que, la dernière catégorie de méthode utilise des techniques d'ensemble pour résoudre les problèmes produits par les deux catégories précédentes.

Dans ce contexte, l'apprentissage profond a été utilisé. Dans cet article, nous avons utilisé une technique basée sur le Deep Learning et plus particulièrement les réseaux de neurones profonds DNN pour la classification des gènes de levures. Les résultats expérimentaux montrent que notre modèle de DNN donne de meilleurs résultats avec un taux de 79% dans la phase de test.

Mots-clés: classification multi label, DNN, l'apprentissage profond, la machine Learning

Abstract

Multi-label classification is an extension of traditional classification in which the classes are not mutually exclusive, each example can belong to several classes simultaneously. This type of classification is required by a large number of current applications such as text classification, image classification and video annotation. It is widely used in the field of bioinformatics, for example for the classification of yeast genes.

According to the literature, several approaches with different strategies have been developed to solve the problems of Multi-Label Learning. These approaches have been organized into three main families, namely: transformation approaches, adaptation approaches and ensemble approaches. The first category of methods represents independent algorithms, which transform the Multi-Label learning problem into one or more single-label classification or regression problems. The second category of methods adapts traditional algorithms for multi-label data. While, the last category of methods uses ensemble techniques to solve the problems produced by the two previous categories.

In this context, deep learning has been used. In this paper, we used a technique based on Deep Learning and more specifically DNNs for yeast gene classification. The experimental results show that our DNN model performs better with a rate of 79% in the test phase.

Keywords: multi-label classification, DNN, Deep Learning, Machine Learning

ملخص

التصنيف متعدد العلامات هو امتداد للتصنيف التقليدي حيت لا تكون الفئات متنافية حيت يكون كل متال قادر عل انتماء الى عدة فئات في وقت واحد. هدا النوع من التصنيف المطلوب من قبل العديد من التطبيقات الحالية متل تصنيف النصوص و الصور و التعليقات التوضيحية للفيديو .يستخدم على نطاق واسع في مجال المعلوماتية الحيوية علء سبيل المتال لتصنيف جينات الخميرة.

وفقا للأدبيات تم تطوير العديد من الاساليب دات استراتيجيات المختلفة لحل مشاكل التعلم متعدد العلامات . تم تنظيم هده الاساليب في تلات عائلات رئيسية و هي نهج التحول نهج التكيف و النهج الشامل. تمتل الفئة الأولى من الاساليب الخوارزميات المستقلة و التي تحول مشكلة التعلم متعدد التسميات الى واحد او اكتر من مشاكل التصنيف او الانحدار دات التسمية الواحدة. الفئة الثانية من الاساليب تتكيف مع الخوارزميات التقليدية لبيانات متعددة التسميات .في حين ان الفئة الاخيرة من الطريقة تستخدم تقنيات المجموعات لحل المشكلات الناتجة عن الفئتين السابقتين في هدا السياق تم استخدام التعلم العميق. في هده المقالة استخدمنا تقنية تعتمد علئ التعلم العميق و بشكل اكتر تحديدا الشبكات العصبية العميقة . تظهر النتائج التجريبية ان هدا النمودج الخاص بنا يعطي نتائج افضل بمعدل 79 في مرحلة الاختبار .

كلمات مفتاحية تصنيف متعدد التسمية. شبكة عصبية عميقة. التعلم العميق. التعلم الالي

La table de matière

Liste des illustrations

Liste des tables

Abréviations

AI	Intelligence artificielle
AA	Apprentissage automatique
DNN	Deep Neural Network
ANN	Artificiel neural network
MLC	Multi Label classification
MLL	Multi Label Learning
CNN	Convolutional neural network
RNN	Recurrent neural network
RELLU	Rectified Linear Unit

Introduction générale

Contexte

La croissance exponentielle des données numériques hétérogènes et l'enrichissement permanent d'Internet par de nouveaux contenus via les multiples avancées technologiques telles que : les chats, les médias sociaux, les réponses aux sondages, le e-learning, le e-santé…, ont mis un accent considérable sur la nécessaire de disposer de solutions efficaces et absolues pour conserver, chercher et classer les données afin d'extraire une information pertinente dans un temps très réduit. Ceci a suscité les spécialistes du domaine de s'orienter vers la classification de texte, afin d'assister les utilisateurs à trouver leurs besoins et faciliter leur travail dont l'intervention humaine est devenue d'aucune utilité.

Avec les récentes percées dans le traitement du langage naturel (NLP) et le texte mining, de nombreux chercheurs sont désormais intéressés au développement d'applications exploitant les méthodes de classification de texte. Ce type de classification est l'un des problèmes les plus fondamentaux de Machine Learning. Il consiste à analyser automatiquement le texte, puis attribuer un ensemble de balises ou de catégories prédéfinies en fonction de son contenu via des algorithmes d'apprentissage supervisé.

Depuis des années, la classification automatique des documents a suscité beaucoup d'attention par les chercheurs du domaine en raison des récents progrès technologiques. Ces derniers, ont permis de partager en ligne de gros volumes de données relatives aux textes appartenant à divers domaines. La tâche de la classification de documents ou de textes, constitue un problème typique à la classification Multi-Label par le fait qu'un document peut appartenir à plusieurs catégories simultanément (Sport, science, médecine).

La classification Multi-Label, était obligatoirement introduite à des applications en catégorisation de texte, ensuite elle s'est étalée vers d'autres types de problématiques tels que la bioinformatique, l'étiquetage des images, la classification des vidéos.

Pour la classification multi-label, notre choix s'est porté sur l'apprentissage profond avec un réseau de neurones profonds (Deep Neural Network - DNN) car depuis quelques années ces réseaux ont montré de bonnes performances et sont devenus l'état de l'art dans le domaine de la vision par ordinateur.

Problématique

La classification multi-label ou classification multi-étiquettes est une variante du problème de classification ou plusieurs étiquettes cibles doivent être affectées à chaque instance. La classification multi- étiquettes ne doit pas être confondue avec celle du multi-classes, qui est le problème de la catégorisation des cas de plus de deux classes .En effet, ce problème peut affecter une ou plusieurs labels a une instance, c'est pour cela, il serait

intéressant de voir si une approche d'apprentissage profond peut faire un bon travail de classifacation.

Objectifs

La classification multi-label est appliquée dans plusieurs domaines tels que la catégorisation des nouvelles (news), l'étiquetage d'image et la classification musicale. Comparativement parlant, la classification multi-label est une tâche plus complexe que la classification multi-classes puisque le classifieur doit apprendre la présence de différentes sorties à la fois à partir du même ensemble de variables prédictives. Donc notre objectif est de trouver une solution pour ce problème. Nous avons pris l'ensemble de données des gènes de levure comme cas d'étude.

Le but majeur de notre recherche est de construire un modèle de classification multi label efficace qui est le réseau de neurones profonds (DNN) afin de prédire les différentes étiquettes.

Organisation du mémoire

Le mémoire est organisé en suivant le plan suivant :

- Le premier chapitre consiste à présenter quelques concepts de base de l'apprentissage machine.
- Le deuxième chapitre consiste à présenter les méthodes et techniques liées à la classification multi-label.
- Dans le troisième chapitre nous présentons le Deep Learning, une description plus détaillée sur les réseaux de neurones profonds (DNNs) qui est le modèle choisie dans notre projet.
- Le quatrième chapitre présente l'implémentation et interprétations des résultats.
- Enfin nous clôturons ce mémoire par une conclusion générale sur le travail effectué ainsi quelques perspectives.

1 CHAPITRE : L'apprentissage Automatique

1.1 Introduction

L'intelligence artificielle est née dans les années 50, quand une poignée de pionniers du domaine naissant de l'informatique, ont commencé à se demander si les ordinateurs pouvaient être amenés à « penser », une question dont nous explorons encore aujourd'hui les ramifications. Une définition concise du champ serait la suivante : l'effort d'automatiser les tâches intellectuelles normalement effectuées par les humains. En tant que tel, l'IA est un domaine général qui englobe l'apprentissage automatique et l'apprentissage en profondeur, mais qui comprend également beaucoup plus d'approches qui n'impliquent aucun apprentissage. Les programmes d'échecs initiaux, par exemple, ne concernaient que des règles codées en dur élaborées par des programmeurs et ne se qualifiaient pas comme apprentissage automatique. Pendant un temps assez long, de nombreux experts ont estimé que l'intelligence artificielle au niveau humain, pouvait être obtenue en faisant en sorte que les programmeurs fabriquent à la main un ensemble suffisamment large de règles explicites pour manipuler les connaissances. .Cette approche est connue sous le nom d'IA symbolique et elle était le paradigme dominant de l'IA des années 50 et à la fin des années 80. Elle a atteint son pic de popularité durant le boom des systèmes experts des années 80. Bien que l'IA symbolique se soit révélée appropriée pour résoudre des problèmes logiques bien définis, comme jouer aux échecs, il était difficile de trouver des règles explicites pour résoudre des problèmes flous plus complexes, tels que la classification des images, la reconnaissance de la parole et la traduction. Une nouvelle approche est apparue pour prendre la place de l'IA symbolique : l'apprentissage automatique [1].

1.2 L'apprentissage Automatique

L'apprentissage automatique (Machine Learning) est un domaine de recherche en informatique qui traite des méthodes d'identification et de mise en œuvre de systèmes et algorithmes par lesquels un ordinateur peut apprendre, ce domaine a souvent été associé à l'intelligence artificielle [2].

Il existe différents types d'apprentissage automatique :

- Apprentissage supervisé.
- Apprentissage non supervisé.
- Apprentissage par renforcement.

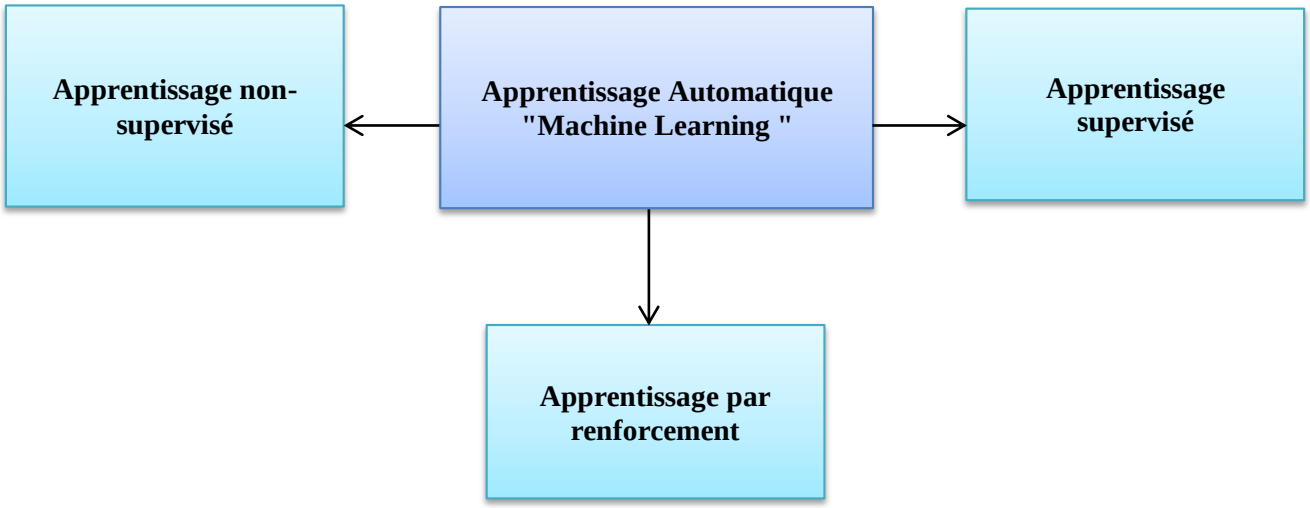


Figure 1 : Les différents types d'apprentissage

1.2.1 Historique

Depuis l'antiquité, le sujet des machines pensantes préoccupe les esprits. Ce concept est la base de pensées pour ce qui deviendra ensuite l'intelligence artificielle, ainsi qu'une de ses sous-branches l'apprentissage automatique. L'historique suit le chainage suivant [3]:

- En 1936 la concrétisation de cette idée est principalement due à Alan Turing (mathématicien et cryptologue britannique) et à son concept de la " machine universelle", qui est à la base des ordinateurs d'aujourd'hui.

- Il continuera à poser les bases de l'apprentissage automatique, avec son article sur « L'ordinateur et l'intelligence » en 1950, dans lequel il développe, entre autres, le test de Turing.

- En 1943, le neurophysiologiste Warren McCulloch et le mathématicien Walter Pitts publient un article décrivant le fonctionnement de neurones en le représentant à l'aide de circuits électriques.

- Cette représentation sera la base théorique des réseaux neuronaux Arthur Samuel, informaticien américain pionnier dans le secteur de l'intelligence artificielle, est le premier à faire usage de l'expression *machine Learning* (en français, « apprentissage automatique ») en 1959 à la suite de la création de son programme pour IBM en 1952. Le programme jouait au Jeu de Dames et s'améliorait en jouant. À terme, il parvint à battre le 4^{e} meilleur joueur des États-Unis.

- En 1997, une avancée majeure dans le secteur de l'intelligence machine est le succès de l'ordinateur développé par IBM, Deep Blue, qui est le premier à vaincre le champion mondial d'échecs Garry Kasparov .

- En 2011 le projet Deep Blue en inspirera nombre d'autres dans le cadre de l'intelligence

artificielle, particulièrement un autre grand défi : IBM Watson, l'ordinateur dont le but est de gagner au jeu Jeopardy. Ce but est atteint quand Watson gagne à Jeopardy. En répondant aux questions par traitement de langage naturel.

- En 2012, un réseau neuronal développé par Google parvient à reconnaître des visages humains ainsi que des chats dans des vidéos YouTube**.**

- En 2014, 64 ans après la prédiction d'Alan Turing, le dialogueur Eugene Goostman est le premier à réussir le test de Turing en parvenant à convaincre 33 % des juges humains au bout de cinq minutes de conversation qu'il est non pas un ordinateur, mais un garçon ukrainien de 13 ans**.**

- En 2015, une nouvelle étape importante est atteinte lorsque l'ordinateur « AlphaGo » de Google gagne contre un des meilleurs joueurs au jeu de Go, jeu de plateau considéré comme le plus dur du monde**.**

- En 2016, un système d'intelligence artificielle à base d'apprentissage automatique nommé LipNet parvient à lire sur les lèvres avec un grand taux de succès [3]**.**

1.2.2 Qu'est-ce que l'apprentissage automatique ou machine Learning ?

L'apprentissage automatique, également appelé apprentissage machine ou apprentissage artificiel et en anglais machine learning, est une forme d'intelligence artificielle (IA) qui permet à un système d'apprendre à partir des données et non à l'aide d'une programmation explicite. Cependant, l'apprentissage automatique n'est pas un processus simple. Au fur et à mesure que les algorithmes ingèrent les données de formation, il devient possible de créer des modèles plus précis basés sur ces données.

Un modèle de machine learning est le résultat généré lorsque vous entraînez votre algorithme d'apprentissage automatique avec des données. Après la formation, lorsque vous fournissez des données en entrée à un modèle, vous recevez un résultat en sortie. Par exemple, un algorithme prédictif crée un modèle prédictif. Ensuite, lorsque vous fournissez des données au modèle prédictif, vous recevez une prévision qui est déterminée par les données qui ont servi à former le modèle [4]

La figure suivante résume les trois types d'apprentissage avec les problèmes connexes à résoudre :

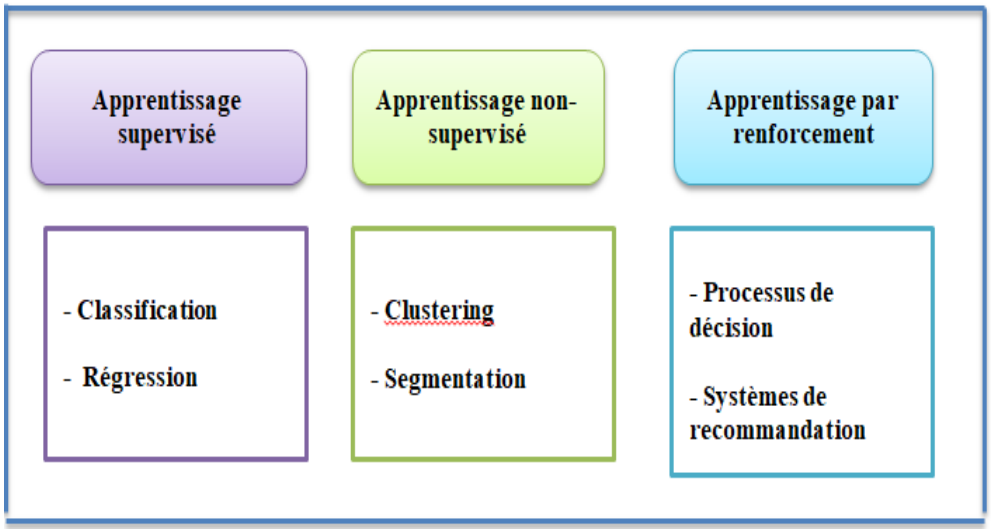


Figure 2 : Types d'apprentissage automatique

1.2.3 Les types d'apprentissage automatique

1.2.3.1 Apprentissage non-supervisé

La Première classe d'algorithmes d'apprentissage automatique est appelée apprentissage non supervisé, dans ce cas, nous n'étiquetons pas les données au préalable, nous laissons plutôt l'algorithme arriver à sa conclusion.

Ce type d'apprentissage est important car il est beaucoup plus commun dans le cerveau humain que l'apprentissage supervisé [2]

Les algorithmes d'apprentissage non supervisé sont particulièrement utilisés dans les problèmes de clustering, dans lesquels, étant donné une collection d'objets, nous voulons être en mesure de comprendre et de montrer leurs relations. Une approche standard consiste à définir une mesure de similarité entre deux objets, puis à rechercher tout groupe d'objets plus similaires les uns aux autres, par rapport aux objets des autres clusters. Par exemple, dans le cas des e-mails spam/ non spam, l'algorithme peut être capable de trouver des éléments communs à tous les spam (par exemple, la présence de mots mal orthographiés) [5].

Bien que cela puisse fournir une classification meilleure qu'aléatoire, il n'est pas clair que les spam/non spam puissent être facilement séparés [6].

L'apprentissage non supervisé comprend deux catégories d'algorithmes: Algorithmes de regroupement et d'association.

Dans la figure 3, on reconnait trois clusters : (un en rouge, un bleu et un vert).

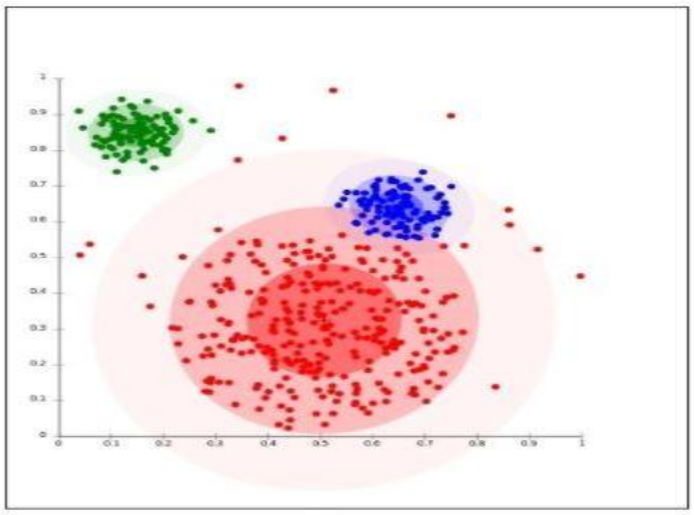


Figure 3 : Exemple d'apprentissage automatique non supervisé (Clustering) [7]

1.2.3.1.1 Regroupement ou Clustering

La mise en cluster consiste à séparer ou à diviser un ensemble de données appartenant aux mêmes groupes se ressemblent d'avantage que ceux d'autres groupes. En terme simples l'objectif est de séparer les groupes ayant des traits similaires et de les assigner en grappes [8].

1.2.3.1.2 Association

L'association consiste à découvrir des relations intéressantes entre des variables dans de grandes bases de données. Par exemple, les personnes qui achètent une nouvelle maison ont aussi tendance à acheter de nouveaux meubles. Il découvre la probabilité de co-occurrence d'éléments dans une collection [8].

1.2.3.2 Apprentissage par renforcement

L'apprentissage par renforcement est une approche de l'intelligence artificielle qui met l'accent sur l'apprentissage du système à travers ses interactions avec l'environnement. Avec l'apprentissage par renforcement, le système adapte ses paramètres en fonction des réactions reçues de l'environnement, qui fournit ensuite un retour d'information sur les décisions prises [2].

Par exemple, un système qui modélise un joueur d'échecs qui utilise le résultat des étapes précédentes pour améliorer ses performances, est un système qui apprend avec le renforcement.

La recherche actuelle sur l'apprentissage avec renforcement est hautement interdisciplinaire et comprend des chercheurs spécialisés dans les algorithmes génétiques, les réseaux de neurones, la psychologie et les techniques de contrôle [6].

1.2.3.3 Apprentissage Supervisé

L'apprentissage supervisé est basé sur un certain nombre d'exemples pré classifiés, dans lesquels est connu à priori la catégorie à laquelle appartient chacune des entrées utilisées comme exemples. Dans ce cas, la question cruciale est le problème de généralisation, après l'analyse d'un échantillon d'exemples, le système devrait produire un modèle qui devrait fonctionner pour toutes les entrées possibles.

L'ensemble de données pour l'entrainement, est constitué de données étiquetées, c'est-à-dire d'objets et de leurs classes associées. Cet ensemble d'exemples étiquetés constitue donc l'ensemble d'apprentissage. Afin de mieux comprendre ce concept, prenons un exemple : un utilisateur reçoit chaque jour un grand nombre d'e-mails, certains sont des e-mails d'entreprises importants et d'autres sont des e-mails indésirables non sollicités ou des spam.

Un algorithme supervisé sera présenté avec un grand nombre d'e-mails qui ont déjà été étiquetés par l'utilisateur comme spam ou non spam. L'algorithme fonctionnera sur toutes les données étiquetées, faire des prédictions sur l'e-mail et voir si c'est un spam ou non. Cela signifie que l'algorithme examinera chaque exemple et fera une prédiction pour chacun pour savoir si l'e-mail est un spam ou pas. La première fois, l'algorithme fonctionne sur toutes les données non étiquetées, la plupart des e-mails seront mal étiquetés car il peut fonctionner assez mal au début. Cependant, après chaque exécution, l'algorithme compare sa prédiction au résultat souhaité (l'étiquette). Au fur et à mesure, l'algorithme apprendra à améliorer ses performances et sa précision [2].

Dans certains cas, le résultat n'est pas nécessairement discret et il se peut que nous n'ayons pas un nombre fini de classes dans lesquelles classer nos données. Par exemple, nous essayons peut-être de prédire l'espérance de vie d'un groupe de personnes en fonction de paramètres de santé préétablis. Dans ce cas, comme le résultat est une fonction continue (nous pouvons spécifier une espérance de vie comme un nombre réel exprimant le nombre d'années que la personne devrait vivre), nous ne parlons pas d'une tâche de classification mais plutôt d'un problème de régression [5].

Pour cet apprentissage, nous avons des données en entrée (Features) et le résultat attendu (Label). Il nous permet de faire des prédictions basées sur un modèle qui est obtenu à partir de données d'historique et de l'algorithme choisi.

Il tente de répondre à une question : Classification : "quelle classe"?

Dans un problème de régression, l'ensemble d'apprentissage est une paire formée par un objet et une valeur numérique associée. Il existe plusieurs algorithmes d'apprentissage supervisé qui ont été développés pour la classification et la régression. Parmi tous, les arbres de décision, les règles de décision, les réseaux de neurones et les réseaux bayé siens [6].

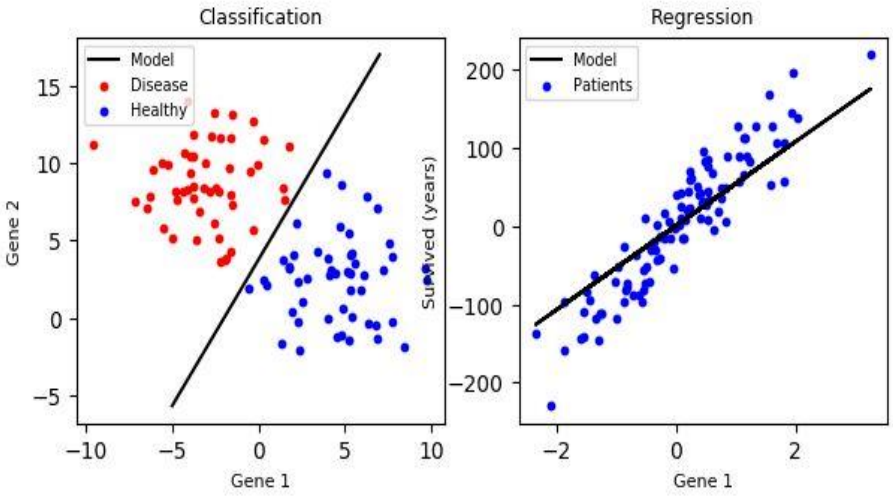


Figure 4 : La classification / La régression [9]

1.3 Classification

L'apprentissage automatique est un domaine d'étude qui s'intéresse aux algorithmes qui apprennent à partir d'exemples. La classification est une tâche qui nécessite l'utilisation d'algorithmes d'apprentissage automatique qui apprennent à attribuer une étiquette de classe aux exemples du domaine problématique [10].

Un exemple facile à comprendre est la classification des courriels comme "spam" ou "non spam".

Il existe de nombreux types de tâches de classification qu'on peut rencontrer dans l'apprentissage automatique et des approches spécialisées de la modélisation qui peuvent être utilisées pour chacune [10]. Les types de classification sont :

- La classification binaire
- La classification multi-classes
- La classification multi-label

1.3.1 La classification binaire (Binary classification)

La classification binaire (ou la classification binomiale) est une transformation de données qui vise à répartir les membres d'un ensemble dans deux groupes disjoints selon que l'élément possède ou non une propriété / fonctionnalité donnée. Par exemple, un modèle d'apprentissage automatique qui classe les courriers électroniques en tant que «indésirable» ou «légitime») est une classification binaire [11].

Dans la classification binaire, il n'y a que deux classes, la classe positive et la classe négative.

La classification binaire consiste à classer les éléments d'un ensemble en deux groupes sur la base d'une règle de classification. Les problèmes typiques de classification binaire sont les suivants [11]:

- Tests médicaux visant à déterminer si un patient est atteint d'une certaine maladie ou non.
- Le contrôle de la qualité dans l'industrie, qui consiste à déterminer si une spécification a été respectée.
- Dans le domaine de la recherche d'informations, il s'agit de décider si une page doit ou non figurer dans l'ensemble des résultats d'une recherche.

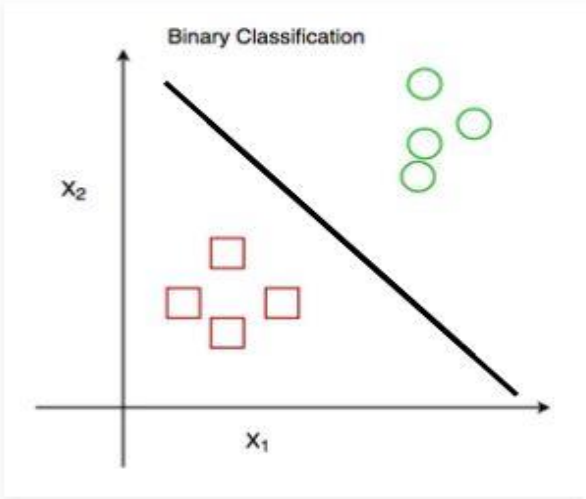


Figure 5 : Exemple de problème de classification binaire [10]

1.3.2 La classification multi classes

La classification multi classes désigne une tâche de classification comportant plus de deux classes, par exemple, classer un ensemble d'images de fruits qui peuvent être des oranges, des pommes ou des poires. La classification multi classes part du principe que chaque échantillon est affecté à une et une seule étiquette : un fruit peut être soit une pomme, soit une poire, mais pas les deux en même temps [12] .

La classification multi-classes fait référence aux tâches de classification qui ont plus de deux étiquettes de classe.

Exemples :

- La classification des visages.
- Classification des espèces végétales.
- Reconnaissance de caractères optiques.

Dans ce type de classification, les points sont classés en trois classes ou plus (Figure 4).

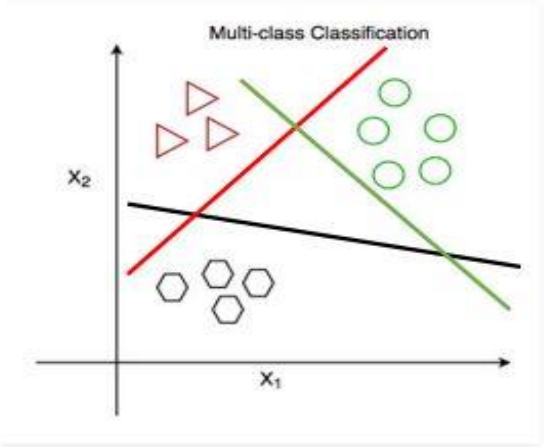


Figure 6 : Exemple de problème de classification multi classes [10]

La classification multi-classes peut être transformée en un problème de classification binaire en utilisant les deux approches suivantes :

1.3.2.1 One Vs One

Dans cette approche, un classifieur binaire est formé pour chaque paire de classes (figure 7). Pour un problème de classification avec N classes N * (N-1) / 2 classifieurs binaires sont entraînés. Au moment de la prédiction, un schéma de vote est appliqué, chaque classifieur vote pour une classe et la classe avec le plus de votes serait la prédite [13].

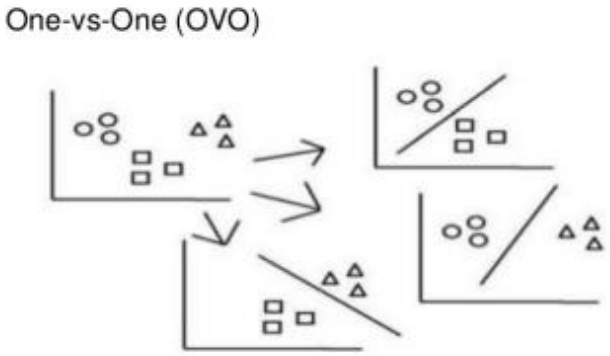


Figure 7 : Illustration de l'approche OneVsOne [13]

1.3.2.2 One Vs Rest

Dans One Vs Rest (également appelé One Vs All), un classifieur binaire est formé pour chacune des classes et dans chaque processus de formation du classifieur. Les étiquettes des données appartenant à la classe cible représentent la classe positive et tous les autres points qui appartiennent au reste de la classe représentent la classe négative comme le montre la figure 8. Lors de la prédiction de la

classe d'un nouvel échantillon, tous les classifieurs formés sont invités à prédire la classe de l'échantillon et le classifieur avec la probabilité la plus élevée détermine la classe [13].

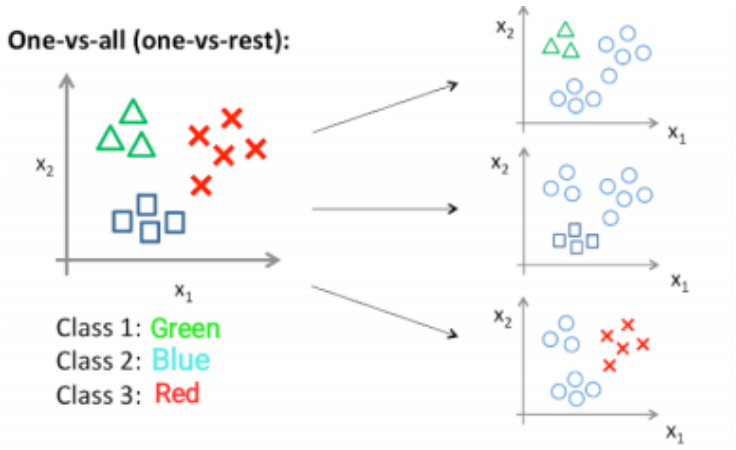


Figure 8: Illustration de l'approche OneVsRest [13]

1.3.3 La classification multi label

Jusqu'à présent, chaque instance a toujours été affectée à une seule classe. Dans certains cas notre classificateur peut produire plusieurs classes pour chaque instance. Par exemple, considérons un classificateur de reconnaissance de visage : que doit-il faire s'il reconnaît plusieurs personnes sur la même image ? Bien sûr, il devrait attacher une étiquette à chaque personne qu'il reconnaît. Disons que le classificateur a été entraîné à reconnaître trois visages, Alice, Bob et Charlie ; alors lorsqu'on lui montre une image d'Alice et de Charlie.

Il devrait produire [1, 0, 1] (ce qui signifie "Alice oui, Bob non, Charlie oui"). Un tel système de classification qui produit plusieurs binaires multiples est appelé un système de classification multi-label [12].

Table 3

x	y
x_1	$[t_2, t_5]$
x_2	$[t_1, t_2, t_3, t_4]$
x_3	$[t_3]$
x_3	$[t_2, t_4]$
x_3	$[t_1, t_3, t_4]$

Figure 9 : Exemple de problème de classification multi label [7]

1.4 La différence entre la classification multi-classes et la classification multi-label

La différence entre la classification multi-classes et la classification multi-labels est que dans les problèmes multi-classes, les classes s'excluent mutuellement, alors que dans les problèmes multi-labels, chaque étiquette représente une tâche de classification différente, mais les tâches sont en quelque sorte liées [12] .

Par exemple, la classification multi-classes part du principe que chaque échantillon est affecté à une et une seule étiquette : un fruit peut être soit une pomme, soit une poire, mais pas les deux en même temps. En revanche, un exemple de classification multi-label peut être qu'un texte peut porter sur la religion, la politique, la finance ou l'éducation en même temps ou sur aucun de ces sujets [12].

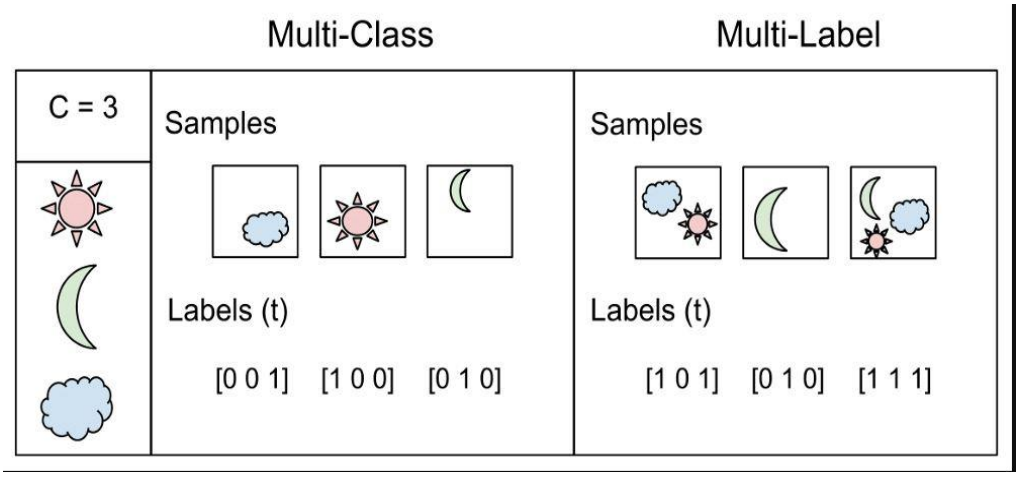


Figure 10 : La différence entre la classification multi-classes et la classification multi-label [14]

1.5 Conclusion

Dans ce chapitre nous avons présenté le Machine Learning qui est un des champs de l'Intelligence Artificielle (IA) qui consiste en l'automatisation de l'apprentissage d'un algorithme, notamment par l'analyse, la sélection et le traitement de données.

Nous avons donné un bref historique sur l'apprentissage automatique, ces différents types (non-supervisé, Supervisé, par renforcement). Aussi nous avons présenté la classification, ces types (La classification binaire, la classification multi classes et la classification multi-label).

2 CHAPITRE :

Classification multi label

2.1 Introduction

La classification est l'une des principales tâches de la fouille de données. Cette formulation du problème implique la restriction d'une seule étiquette par motif, néanmoins, de plus en plus de problèmes de classification sont envisagés aujourd'hui, tels que la catégorisation de textes et de sons, la classification de scènes sémantiques, le diagnostic médical ou la classification de fonctions de gènes et de protéines, où un motif peut avoir plusieurs étiquettes simultanément associées. Par exemple, dans le domaine de la classification sémantique de scènes, une image contenant un paysage avec à la fois une plage et une montagne pourrait être associée aux catégories plage et montagne simultanément. Ce type de problème est appelé multi-label par rapport à l'apprentissage supervisé classique (également appelé single-label). La résolution d'un problème avec des données multi-label pose de nouveaux défis en raison de la croissance exponentielle des combinaisons de labels à prendre en compte et du coût de calcul pour construire et interroger les modèles. De plus, les données multi-label possèdent généralement des caractéristiques spécifiques, telles que la dimensionnalité élevée des données non équilibrées et la corrélation entre les labels [15].

2.2 Définition du problème et approches d'apprentissage multi-label

2.2.1 Définition formelle

Nous définissons la tâche de l'apprentissage multi-label comme suit [13]:

- Un espace d'exemples X qui consiste en des vecteurs de valeurs de types de données primitives (booléennes, discrètes ou continues), c'est-à-dire ∀xiεX = (xi1, xi2,…, xiD) où D est la taille du vecteur (ou nombre d'attributs descriptifs) ;
- Un espace d'étiquettes L = {y1, y2, . . ., yQ} qui est un vecteur de Q variables discrètes (avec valeurs 0 ou 1) ;
- Un ensemble d'exemples E, où chaque exemple est une paire de vecteurs de l'espace des exemples et de celui des étiquettes, respectivement, c'est-à-dire E = {(xi, Yi)\ xi εX, Yi εL, 1 ≤ i ≤ N} et N est le nombre d'exemples de E (N = |E|) ;
- Un critère de qualité q, qui récompense des modèles à haute précision prédictive et faible complexité.

La classification multi-label consiste à définir une fonction h(xi) qui renvoie l'ensemble des étiquettes pertinentes. Alors pour tout xεX = X1 × . . . × XD, un espace d'entrée de dimension D de caractéristiques numériques ou catégorielles, on a une bipartition (Y, $\bar{Y}$) de l'ensemble des étiquettes L, où Y = h(x) est l'ensemble des étiquettes pertinentes et $\bar{Y}$ est l'ensemble de celles non pertinentes. Ici, $\bar{Y}$ = L\Y désigne le complément théorique de Y dans L.

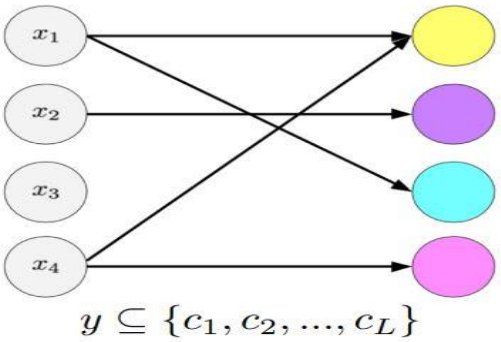


Figure 11 : Exemple de classification multi label [16]

2.3 Approches d'apprentissage multi label

L'apprentissage multi-label est une généralisation de l'apprentissage mono-label classique où chaque exemple xi peut être étiqueté par un ou plusieurs labels simultanément.

L'apprentissage à partir de données multi-label est un problème qui a suscité beaucoup d'attention ces dernières années par la communauté d'apprentissage automatique et les communautés connexes et a conduit au développement de plusieurs approches d'apprentissage. Ces approches peuvent être organisées en trois grandes familles [17].

1. ***Approches d'apprentissage par transformation :*** elles transforment le problème d'apprentissage multi-label en un ou plusieurs problèmes de classification ou de régression mono-label.

2. ***Approches d'apprentissage par adaptation :*** elles adaptent des algorithmes d'apprentissage pour des données multi-label.

3. ***Approches d'apprentissage ensemble :*** elles utilisent un ensemble de classifieurs issus de la première ou de la deuxième famille d'approches.

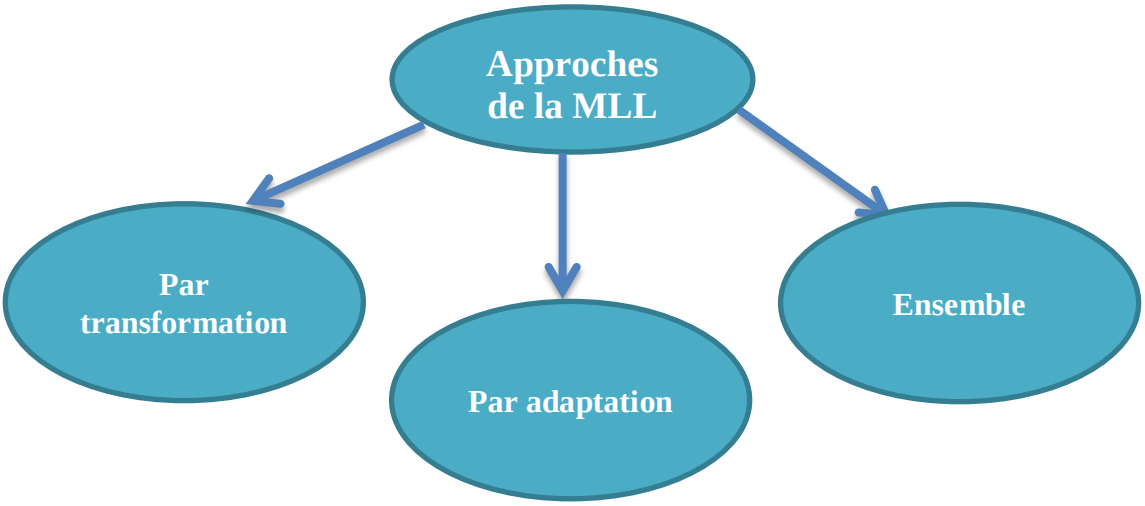


Figure 12 : Les approches de la MLL (Multi Label Learning)

Dans ce qui suit, nous présentons neuf méthodes d'apprentissage les plus représentatives des trois grandes familles. Nous précisons le critère qu'elles optimisent implicitement ou explicitement et rappelons leurs complexités d'apprentissage et de prédiction dans le pire des cas ainsi que leurs avantages et inconvénients.

La figure si dessous montre les différentes méthodes utilisées dans chaque approche

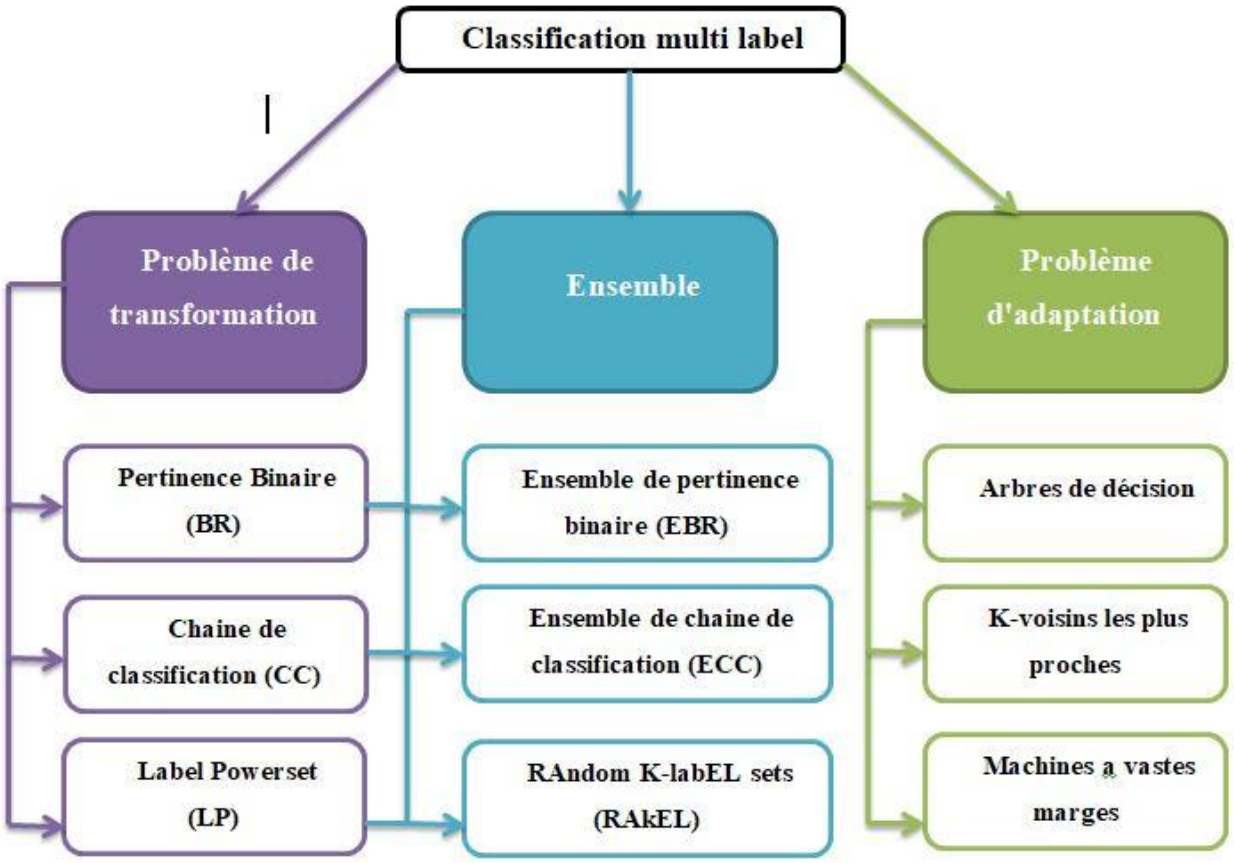


Figure 13 : Les approches de l'apprentissage Multi-Label [18]

2.3.1 Approches d'apprentissage par transformation

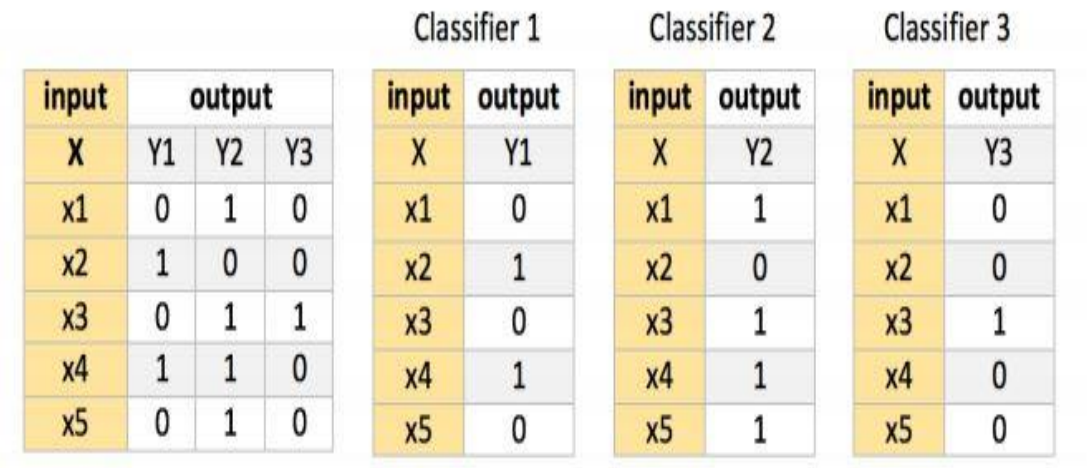
Les méthodes de transformation du problème sont des méthodes d'apprentissage multilabel qui transforment le problème d'apprentissage multi-label en un ou plusieurs problèmes de classification ou de régression à un seul marqueur d'étiquettes [17].

2.3.1.1 Méthode de pertinence binaire (BR)

La méthode de la pertinence binaire est un algorithme élémentaire qui décompose (ou bien qui transforme) un problème d'apprentissage à étiquettes multiples en un ensemble de problèmes de classification binaires indépendants. Cette approche est connue sous le nom de méthode de

pertinence binaire (Binary Relevance : BR).BR est la méthode la plus populaire et la plus simple de cette classe d'approches.

Elle traite du problème d'apprentissage multi-label en apprenant un classifieur pour chaque étiquette, en utilisant tous les exemples étiquetés avec ce label comme exemples positifs et tous les autres exemples comme négatifs. Lors d'une prédiction, chaque classifieur binaire prédit si son étiquette est pertinente pour l'exemple donné ou non, ce qui donne à la fin, un ensemble d'étiquettes pertinentes. Dans le scénario de classement, les étiquettes sont classées selon la probabilité associée à chaque étiquette par le classifieur binaire respectif [17].



input	output		
X	Y1	Y2	Y3
x1	0	1	0
x2	1	0	0
x3	0	1	1
x4	1	1	0
x5	0	1	0

Classifier 1

input	output
X	Y1
x1	0
x2	1
x3	0
x4	1
x5	0

Classifier 2

input	output
X	Y2
x1	1
x2	0
x3	1
x4	1
x5	1

Classifier 3

input	output
X	Y3
x1	0
x2	0
x3	1
x4	0
x5	0

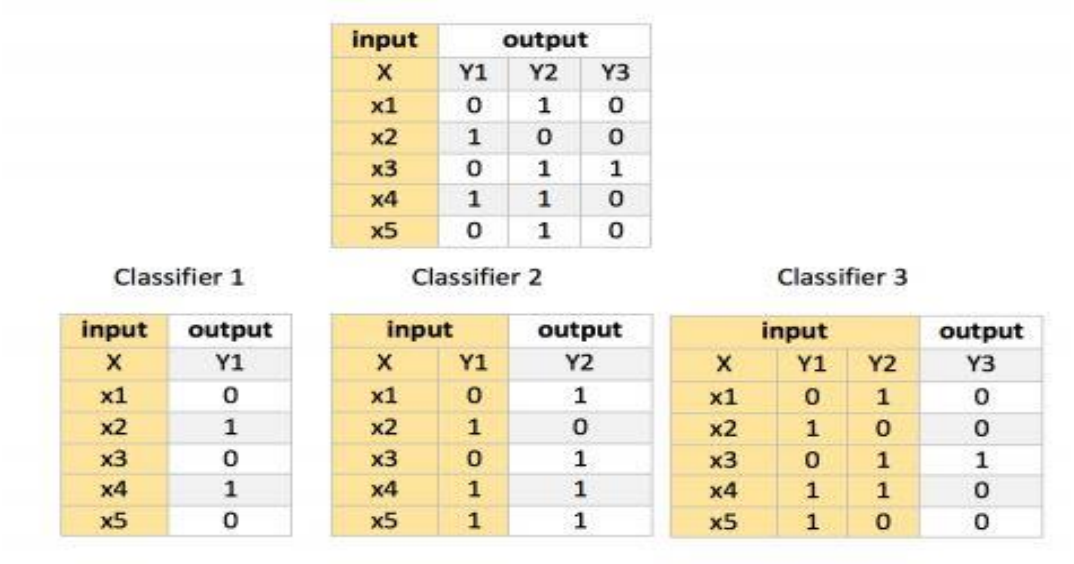
Figure 14 : Un exemple de transformation d'un problème de classification multi-étiquettes dans un problème de classification binaire en utilisant l'approche de pertinence binaire [19]

2.3.1.2 Méthode de chaine de classification (CC)

Pour obtenir de meilleures performances, la méthode de chaine de classification "CC" a été introduite en multi-label. Elle transforme un problème d'apprentissage multi-labels en une chaîne de classificateurs binaires basée sur la dépendance des étiquettes.

La méthode de chaine de classification est une amélioration de la méthode BR qui transforme également le problème d'apprentissage multi-label en q problèmes de classification ou de régression mono-label.

Cependant, les classifieurs sont entraînés dans un ordre aléatoire défini avant la phase d'apprentissage [1.., j, Q] tel que chaque classifieur binaire hj apprenant un label yj ajoute tous les labels associés aux classifieurs qui le précédent dans la chaîne (i.e. y1, . . ., yj−1) dans son espace d'attributs. Comme BR, pour un nouvel exemple, CC retourne l'ensemble des prédictions générées par l'ensemble des classifieurs [19].



input	output		
X	Y1	Y2	Y3
x1	0	1	0
x2	1	0	0
x3	0	1	1
x4	1	1	0
x5	0	1	0

Classifier 1

input	output
X	Y1
x1	0
x2	1
x3	0
x4	1
x5	0

Classifier 2

input		output
X	Y1	Y2
x1	0	1
x2	1	0
x3	0	1
x4	1	1
x5	1	1

Classifier 3

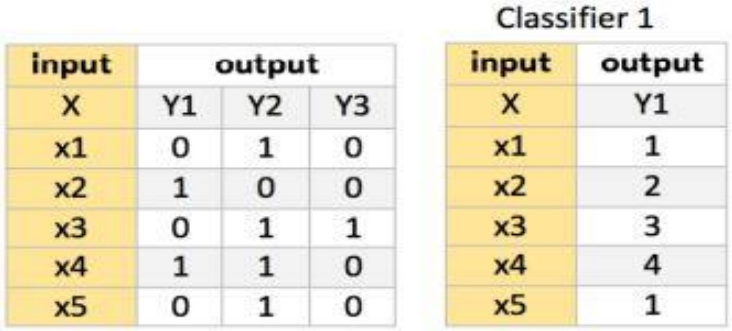
input			output
X	Y1	Y2	Y3
x1	0	1	0
x2	1	0	0
x3	0	1	1
x4	1	1	0
x5	1	0	0

Figure 15 : Approches chaine de classifieurs [13]

2.3.1.3 Méthode d'étalonnage des étiquettes LP (Label Powerset)

Une deuxième méthode de transformation du problème est la méthode de combinaison d'étiquettes, ou méthode d'étalonnage des étiquettes (Label Powerset : LP), qui a fait l'objet de plusieurs études. LP transforme le problème d'apprentissage multi-label en un seul problème d'apprentissage mono-label à plusieurs classes.

LP considère chaque combinaison de labels présente dans l'ensemble d'apprentissage comme une classe et apprend ensuite un classifieur multi-classes h [19].



input	output		
X	Y1	Y2	Y3
x1	0	1	0
x2	1	0	0
x3	0	1	1
x4	1	1	0
x5	0	1	0

Classifier 1

input	output
X	Y1
x1	1
x2	2
x3	3
x4	4
x5	1

Figure 16 : Approches d'étalonnage d'étiquettes [13]

2.3.2 Approches d'apprentissage par adaptation

Les méthodes multi-label qui adaptent, élargissent et personnalisent un algorithme d'apprentissage machine existant pour la tâche d'apprentissage multi-label sont appelées méthodes d'adaptation d'algorithme [20] .

Les méthodes d'adaptation des algorithmes pour la classification multi-labels se concentrent sur l'adaptation des algorithmes de classification à un seul label au cas multi-labels, généralement par des changements dans les fonctions de coût/décision [20].

Nous présentons ici une des méthodes multi-label proposées dans la littérature qui est basée sur les algorithmes d'apprentissage machine comme les k- plus proches voisins.

2.3.2.1 Arbres de décision

"Clare" est un algorithme C4.5 adapté pour la classification multi-label ; la modification implique les calculs d'entropie.MMC, MMDT, et SSC MMDT raffiné, peuvent classer des données multi-labellisées basées sur des attributs multi-valeurs sans transformer les attributs en valeurs uniques.

Elles sont également appelées méthodes de classification par arbre de décision multi-valué et multi-étiqueté.

Multi-Label C4.5 (ML-C4.5) est une adaptation de l'algorithme C4.5 bien connu pour l'apprentissage multi-label permettant plusieurs étiquettes dans les feuilles de l'arbre. Cette méthode est devenue une référence et a été principalement utilisée comme classifieur de base dans des ensembles de méthodes d'apprentissage multi-label.

Le concept d'arbres de regroupement prévisionnel est proposé (Predictive Clustering Trees : PCT). Ils peuvent également être utilisés dans le contexte de l'apprentissage multi-label, où chaque étiquette est une composante du vecteur cible.

Les PCTs sont des arbres de décision considérés comme une hiérarchie de grappes: le nœud supérieur correspond à un cluster contenant toutes les données, qui est divisé de façon récursive en grappes plus petites tout en descendant dans l'arborescence. Les PCTs sont construits en utilisant un algorithme standard d'induction de haut en bas des arbres de décision, où la variance et la fonction prototype peuvent être instanciées en fonction de la tâche à accomplir. Les PCTs ont donné de très bons résultats de classification combinés avec des forêts aléatoires [20].

2.3.2.2 K-voisins les plus proches

Multi-label K Nearest Neighbors utilise k-Nearest Neighbors pour trouver les exemples les plus proches d'une classe de test et utilise l'inférence bayésienne pour prédire les étiquettes.Cette méthode est basée sur la distance et fonctionne bien lorsqu'il existe une relation entre la distance et les étiquettes.

En phase d'apprentissage, ML-kNN estime les probabilités a priori et a posteriori de chaque label à partir des exemples d'apprentissage. Pour un nouvel exemple xi , ML-kNN calcule ses k plus proches voisins puis mesure la fréquence de chaque label dans ce voisinage. Cette fréquence est ensuite combinée avec les probabilités estimées dans la phase d'apprentissage pour déterminer son ensemble de labels en suivant le principe du maximum a posteriori (MAP) [21].

2.3.2.3 Machines à vastes marges

De nombreuses approches ont utilisé des SVM à étiquette unique avec une approche un contre-tous. De plus deux mécanismes ont été présentés pour améliorer la qualité de la marge des SVM dans un environnement un-contre-tous.

Le premier, la méthode de suppression de bandes (BandSVM), fonctionnait au niveau de l'instance. Une fois qu'une SVM un-contre-tous avait été apprise, elle a enlevé des exemples négatifs semblables qui se trouvaient à une distance seuil (bande) de l'hyperplan de décision appris.

L'approche d'adaptation d'algorithme a également été utilisée, ainsi ont proposé Rank-SVM, une méthode de classement, basée sur les SVM, qui est devenue une référence dans l'apprentissage multi-label. La fonction de coût qu'ils utilisent est la fraction moyenne de paires d'étiquettes incorrectement ordonnées. Rank-SVM définit un ensemble de Q classifieurs linéaires qui sont optimisés pour minimiser une mesure qui évalue la fraction moyenne de paires d'étiquettes qui sont ordonnées inversement pour l'instance (c'est-à-dire la perte de classement empirique (Ranking-loss) définie parmi les mesures d'évaluation du classement) [22].

2.3.3 Approches d'ensembles

Les méthodes d'apprentissage d'ensemble sont des méthodes qui combinent les méthodes par adaptation et les méthodes par transformations. Nous présentons ici trois de ces méthodes : RAkEl , ECC et EBR.

2.3.3.1 RAndom k-labEL sets (RAkEL)

Le RAndom k-labELsets (RAkEL) est une méthode d'ensemble pour la classification multi-label. Il dessine m sous-ensembles aléatoires d'étiquettes de taille k à partir de toutes les étiquettes L et forme un classifieur d'étalonnage d'étiquettes à l'aide de chaque jeu d'étiquettes. Un processus de vote simple détermine le jeu final d'étiquettes pour un exemple donné [20].

2.3.3.2 Ensemble de chaine de classification (ECC)

Les ECC sont des ensembles de chaînes de classification qui représentent une technique de classification multi-label d'ensemble à base de classifieurs. Entraîne m CC classifieurs C1, C2, ..., Cm. Chaque Ck est entraîné avec un ordre aléatoire de chaîne (de L) et un sous-ensemble aléatoire de X. Par conséquent, chaque modèle Ck est susceptible d'être unique et capable de donner des prédictions multi-label différentes.

Ces prédictions sont additionnées par étiquette afin que chaque étiquette reçoive un certain nombre de votes. Un seuil est utilisé pour sélectionner les étiquettes les plus populaires qui forment le jeu prédit final d'étiquettes multiples. La prévision finale est obtenue en additionnant les prédictions par étiquette, puis en appliquant un seuil pour sélectionner les étiquettes pertinentes [17].

2.3.3.3 Ensemble de pertinence binaire (EBR)

L'ensemble de classificateurs de pertinence binaire (EBR) est généré en utilisant le bagging pour chaque classificateur BR. La génération d'un ensemble de RB, chacun avec une sélection aléatoire d'instances, améliore les performances des RB grâce à la diversité des classificateurs de base [23].

Le tableau si dessous montre les différentes méthodes d'apprentissage multi-label ainsi que leurs avantages et inconvénients.

Approches	Méthodes	Avantage	Inconvénient
Par Transformation	Pertinence binaire (BR) [17]	Faible complexité en apprentissage (relative à un classifieur de base)	La limitation de l'étiquetage des informations sur les relations. Si toutes les étiquettes ne soient indépendantes, la méthode perd les corrélations entre elles.
	Chaine de classification (CC) [19]	Son avantage est sa vitesse d'apprentissage du modèle et sa modélisation des corrélations entre les labels	Sa définition aléatoire de l'ordre d'apprentissage des modèles reste une faiblesse
	Méthode d'étalonnage des étiquettes LP (Label Powerset) [19]	Faible complexité de calcul du modèle mais aussi son exploitation naturelle des corrélations entre labels	Elle ne permet pas de prédire de nouvelles classes (combinaisons de labels) qui n'existent pas dans l'ensemble d'apprentissage

Par adaptation	Arbres de décision [20]	Ils peuvent être calculés automatiquement à partir de bases de données par des algorithmes d'apprentissage supervisé	L'apprentissage par arbre de décision peut amener des arbres de décision très complexes, qui généralisent mal l'ensemble d'apprentissage
	K-voisins les plus proches [21]	Bonnes performances en général	L'algorithme devient beaucoup plus lent à mesure que le nombre d'exemples d'apprentissage augmente.
	Machines à vastes marges [22]	Le SVM fonctionne relativement bien lorsqu'il existe une marge de séparation claire entre les classes.	L'algorithme SVM ne convient pas aux grands ensembles de données.
Ensemble	RAndom K-labEL sets (RAkEL) [20]	Elle permet de prédire de nouvelles combinaisons de labels qui n'existent pas dans l'ensemble d'apprentissage	Elle n'explore pas suffisamment tous les sous-ensembles de labels pour capturer toutes les corrélations et se focalise uniquement sur l'apprentissage de quelques sous-ensembles de taille k
	Ensemble de chaine de classifieur (ECC) [17]	Elle tente d'améliorer les performances de son classifieur de base (CC) en multipliant les modèles.	Son complexité d'apprentissage croît linéairement avec le nombre de labels.
	Ensemble de pertinence binaire (EBR) [23]	Elle tente d'améliorer les performances de son classifieur de base (BR) en multipliant les modèles.	Son complexité d'apprentissage croît linéairement avec le nombre de labels.

Tableau 1 : Les différentes méthodes d'apprentissage multi labels

2.4 Conclusion

Dans ce chapitre nous avons présenté les différentes méthodes et techniques liées à la classification multi-label., tant au niveau de la classification. . Comme nous avons pu le constater, la classification multi-label fait appel à plusieurs techniques pour la résolution des taches. Nous commençons par une définition de l'apprentissage multi-label. Ensuite, nous présentons un aperçu des méthodes les plus récentes d'apprentissage multi-label, ainsi que leurs avantages et inconvénients.

3 CHAPITRE : L'apprentissage profond (Deep Learning)

3.1 Introduction

L'apprentissage profond (Deep Learning) est une technique d'apprentissage automatique (Machine Learning) qui a considérablement amélioré les résultats dans de nombreux domaines tels que la vision par ordinateur, la reconnaissance de la parole et la traduction automatique. Les techniques d'apprentissage profond permettent, à l'aide de données, de résoudre de nombreux problèmes dans de nombreux domaines de l'économie tels que la santé, le transport, le commerce, la finance ainsi que l'énergie. L'apprentissage en profondeur est une forme d'apprentissage automatique qui permet des ordinateurs pour apprendre de l'expérience et comprendre le monde en termes de hiérarchie des concepts. Un graphique de ces hiérarchies serait profond de plusieurs couches [24].

3.2 Qu'est-ce qu'un réseau de neurones profond ?

Un réseau de neurones est un ensemble d'algorithmes inspirés par le cerveau humain. Le but de cette technologie est de simuler l'activité du cerveau humain, et plus spécifiquement la reconnaissance de motifs et la transmission d'informations entre les différentes couches de connexions neuronales [25].

Un Deep Neural Network, ou réseau de neurones profond, se distingue par une particularité : il est composé d'au moins deux couches. Ceci lui permet de traiter les données de manière complexe, en employant des modèles mathématiques avancés [25].

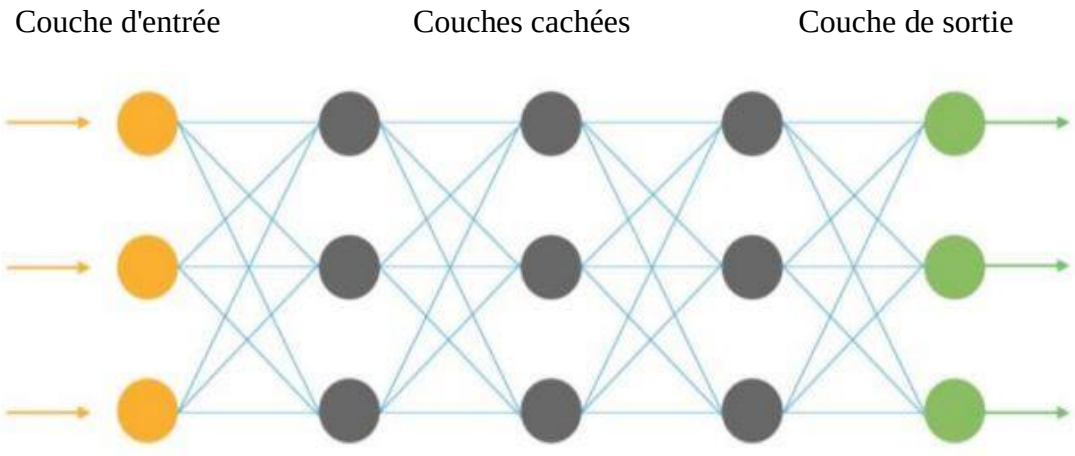


Figure 17: Schéma d'un réseau de neurone profond [26]

3.3 Pourquoi profond ?

Un réseau neuronal est une sorte de logiciel cérébral ou de machine virtuelle composée de milliers d'unités (les neurones) qui effectuent des calculs. Plus précisément, des unités logiques

et de prises de décisions (neurones, perceptrons) relient les données d'entrées et de sorties entre elles par l'intermédiaire d'un réseau complexe (réseau, cerveau) capable de prendre des décisions complexes [27].

A l'origine, ces systèmes portaient le nom de réseaux artificiels de neurones (ANN, Artificial Neural Networks) afin de les différencier des systèmes biologiques. Ils se composent en général d'un certain nombre de données d'entrées et de sorties (input / output layer), d'un réseau étroit de neurones et de plusieurs strates intermédiaires (hidden layers).

Ces couches intermédiaires permettent de traiter des problèmes complexes ; sans elles, le système ne résout que des calculs simples. Le nombre de couches est donc un facteur décisif pour la complexité du système, et de l'apprentissage ; les données s'associent d'une couche à l'autre, les résultats d'une première couche servant d'entrée à la prochaine, et ainsi de suite afin d'aboutir à une prise de décision complexe.

Ce fonctionnement en strates donne toute sa profondeur au réseau et à l'apprentissage. L'adjectif « profond » s'entend ici dans tous les sens du terme [27].

3.4 A quoi sert ?

Les réseaux de neurones profonds représentent en réalité le dernier maillon dans l'évolution de l'intelligence artificielle. À l'origine, le Machine Learning permet d'automatiser les modèles statistiques par le biais d'algorithmes pour réaliser de meilleures prédictions [28].

Un modèle de Machine Learning est en mesure d'établir des prédictions pour une seule tâche. Il se contente d'apprendre en modifiant ses poids à chaque prédiction erronée pour gagner en précision.

Par la suite, les réseaux de neurones artificiels ont vu le jour. Ces réseaux utilisent une couche cachée pour stocker et évaluer l'impact de chaque " input " sur la production finale. Des informations sur l'impact de chaque entrée sont stockées et dissimulées, au même titre que des associations entre les données [28].

Enfin, les réseaux de neurones profonds ont été inventés. Plutôt que de se contenter d'une seule couche cachée, les Deep Neural Networks vont encore plus loin en combinant de multiples couches cachées pour encore plus de bénéfices.

3.5 Comment fonctionne-il ?

En général, un Deep Neural Network a une couche d'entrée, une couche de sortie et au moins une couche entre les deux. Plus le nombre de couches est élevé, plus un réseau est dit " profond ". Chacune de ces couches effectue différents types de tri et de catégorisation spécifique dans un processus nommé " hiérarchie de caractéristique " [28]

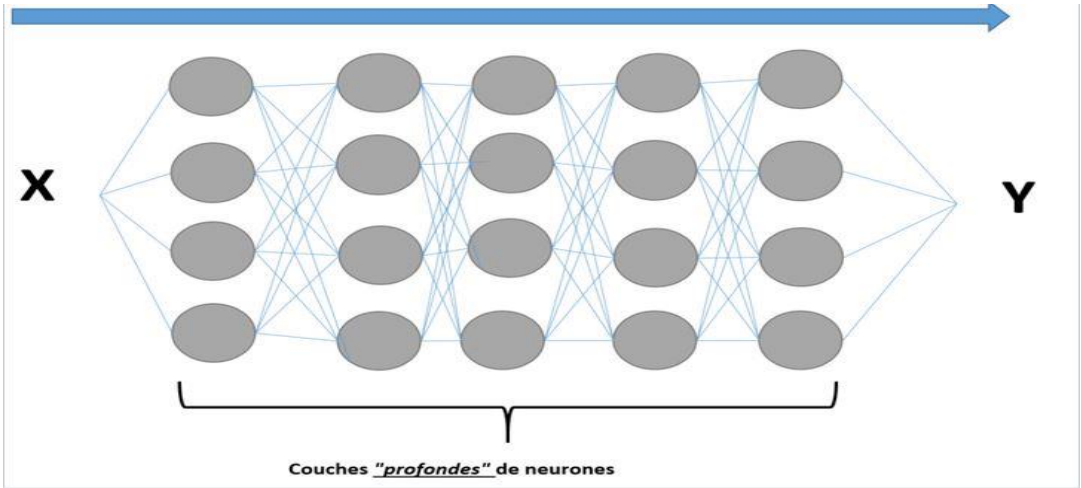


Figure 18 : Réseau de neurone profond [29]

3.6 L'apprentissage des réseaux de neurones profonds (Deep Learning)

Afin de pouvoir imiter le fonctionnement du cerveau humain pour classifier les données qui lui sont fournies, un réseau de neurones doit être entraîné au préalable. C'est le Deep Learning : une forme spécifique de Machine Learning [28].

Lors de la phase d'entraînement, des données d'exemple sont fournies à l'IA. Elle apprend ainsi à reconnaître les motifs et les caractéristiques.

Lorsqu'il reçoit de nouvelles données, sans étiquettes, le réseau de neurones artificiels tente de les catégoriser. Sa prédiction peut être correcte ou incorrecte, et c'est au programmeur de le corriger en cas d'erreur. Au fil des essais, le système apprend de ses erreurs et son taux de réussite s'améliore jusqu'à ce que ses prédictions deviennent infaillibles [28].

3.7 Les différents types de réseaux de neurones dans l'apprentissage profond

Il existe trois types importants de réseaux neuronaux qui constituent la base de la plupart des modèles pré-entraînés en apprentissage profond.

- Réseau de neurones artificiels (ANN)
- Réseau de neurones convolutifs (CNN)
- Réseau de neurones récurrents (RNN)

3.7.1 Réseau neuronal Artificiel (ANN)

Un seul perceptron (ou neurone) peut être imaginé comme une régression logistique. Le réseau neuronal artificiel, ou ANN, est un groupe de plusieurs perceptrons/neurones à chaque couche. L'ANN est également connue sous le nom de réseau neuronal à alimentation directe, car les entrées sont traitées uniquement dans le sens direct.

La figure 3 représente un réseau de neurones artificiel ANN qui se compose de 3 couches : entrée, couche cachée et sortie. La couche d'entrée accepte les entrées, la couche cachée traite les entrées, et la couche de sortie produit le résultat. Essentiellement, chaque couche essaie d'apprendre certains poids [27].

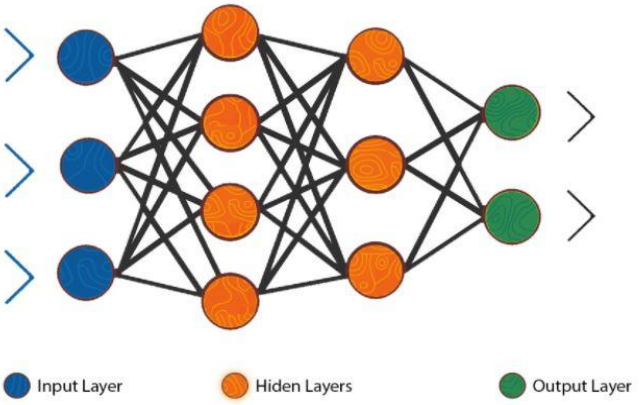


Figure 19 : Réseau de neurone artificiel [30]

L'ANN peut être utilisé pour résoudre des problèmes liés à :

- Données tabulaires
- prédiction de classification
- Prédiction de régression

3.7.1.1 Modélisation des réseaux de neurones artificiels

Un réseau de neurones peut être considéré comme un modèle mathématique de traitement, composé de plusieurs éléments de calcul non linéaire (neurones), opérant en parallèle et connectés entre eux par des poids. Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Les neurones artificiels sont souvent utilisés sous forme de réseaux qui diffèrent selon le type de connections entre les neurones, une cinquantaine de types peut être dénombrée. Nous distinguons trois types de couches [29]:

Couche d'entrée : les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur. Donc les neurones sont liés à chaque neurone de la couche suivante. Ces liaisons disposent d'un poids (w) permettant de pondérer la valeur.

Les neurones appliquent une fonction mathématique basique. Elle s'appelle : fonction d'activation du neurone. Il en existe plusieurs, On utilise en général RELU.

RELU, est une fonction de redressement, elle va suivre la courbe suivante :

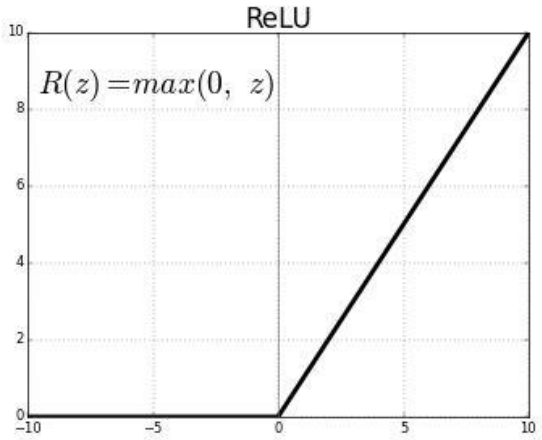


Figure 20 : Fonction d'activation du neurone RELU [31]

Couches cachées : chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectue la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation qui est en général une fonction sigmoïde. Par la suite, il envoie cette réponse aux neurones de la couche suivante [29].

Couche de sortie : elle joue le même rôle que les couches cachées, la seule différence entre ces deux types de couches est que la sortie des neurones de la couche de sortie n'est liée à aucun autre neurone [29].

La couche de sortie dépend du nombre de valeur que nous souhaitons prédire. Le neurone de sortie va également contenir une fonction d'activation, mais cette fois ci, elle sera différente. Pour obtenir une probabilité d'appartenance de données d'entrées au groupe, nous utiliserons la fonction 'Sigmoïd'

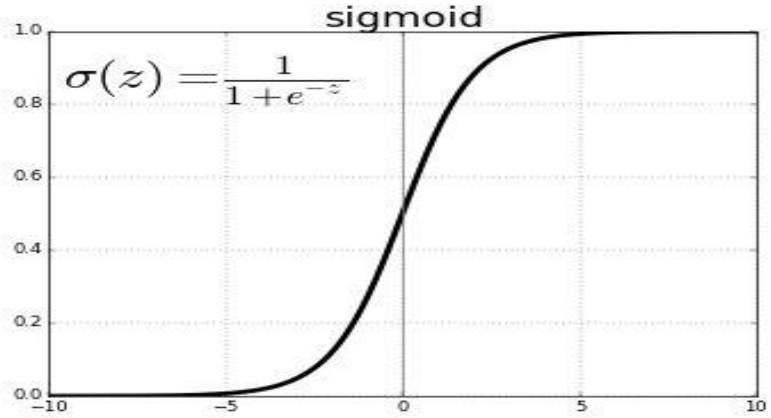


Figure 21 : Fonction d'activation sigmoïde [31]

3.7.1.2 Domaines d'application des réseaux de neurones artificiels

- Traitement du signal : filtrage, classification, identification de source, traitement de la Parole…etc.
- Traitement d'images : reconnaissance de caractères et de signatures, compression d'images, reconnaissance de forme, cryptage, classification, etc.
- Contrôle : commande de processus, diagnostic, contrôle qualité, asservissement des robots, systèmes de guidage automatique des automobiles et des avions…etc.

3.7.2 Réseau neuronal Convolutif (CNN)

L'apprentissage en profondeur (DL) est un domaine de recherche en apprentissage automatique (ML), il s'appuie sur mes réseaux de neurones artificiels, et conçu pour traiter des massives quantités d'informations en augmentant des couches au réseau. En termes simples, il s'agit d'un apprentissage automatique utilisant des réseaux de neurones profonds. Le CNN (réseaux de neurones convolutifs ou réseaux de neurones à convolution), il est considéré parmi les grandes familles des réseaux de neurones profonds, il est l'amélioration des réseaux de neurones artificiels traditionnels, et la connexion entre leurs neurones est inspirée du cortex visuel animal [32].

CNN est utilisé dans la reconnaissance d'image et vidéo, les systèmes de recommandation, et le traitement du langage naturel. Il y a deux parties, la première partie est appelée la partie convolution du modèle, et la deuxième partie est la partie classification.

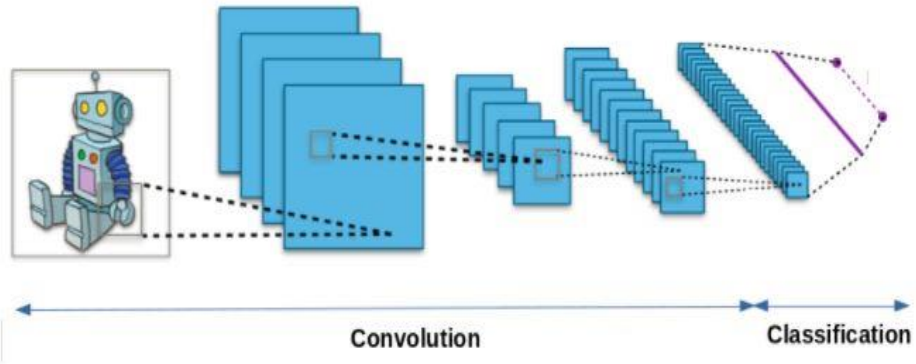


Figure 22 : Architecture du CNN en 2 parties "Convolution et Classification" [32]

CNN peut être utilisé pour résoudre des problèmes liés à :

- Données d'images
- Prédiction de classification
- Prédiction de régression

La couche convolutive (CONV) : Dans cette couche pour que puisse détecter des caractéristiques de l'image il s'agit d'appliquer un filtre de convolution à l'image. L'image passe par une série de filtres ou de noyaux de convolution, pour créer une image appelée carte de convolution. Enfin, les cartes de convolution sont concaténées en un vecteur de caractéristiques [33].

La couche de Pooling (POOL) : La couche de Pooling est une opération généralement appliquée entre deux couches de convolution. Celle-ci reçoit en entrée les features maps formées en sortie de la couche de convolution et son rôle est de réduire la taille des images, tout en préservant leurs caractéristiques les plus essentielles. Parmi les plus utilisés, on retrouve le max-pooling mentionné précédemment ou encore l'average pooling dont l'opération consiste à conserver à chaque pas, la valeur moyenne de la fenêtre de filtre [33].

La couche d'activation RELU : Cette couche remplace toutes les valeurs négatives reçues en entrées par des zéros. L'intérêt de ces couches d'activation est de rendre le modèle non linéaire et de ce fait plus complexe [33].

Couche entièrement connectée (Fully connected) : Ces couches sont placées en fin d'architecture de CNN et sont entièrement connectées à tous les neurones de sorties (d'où le terme fully-connected). Après avoir reçu un vecteur en entrée, la couche FC applique successivement une combinaison linéaire puis une fonction d'activation dans le but final de classifier l'input image (voir schéma suivant). Elle renvoie enfin en sortie un vecteur de taille d correspondant au nombre de classes dans lequel chaque composante représente la probabilité pour l'input image d'appartenir à une classe [33].

3.7.3 Réseau neuronal Récurrent (RNN)

Un réseau de neurones récurrent (RNN, Recurrent Neural Network) est un type de réseau de neurones artificiels principalement utilisé dans la reconnaissance vocale et le traitement automatique du langage naturel. Les RNN sont conçus de manière à reconnaître les caractéristiques séquentielles et les modèles d'utilisation des données requis pour prédire le scénario suivant le plus probable [34].

Les RNN sont utilisés dans le cadre de l'apprentissage profond et dans le développement de modèles qui simulent l'activité du système cérébral humain. Ils sont particulièrement puissants dans les scénarios faisant intervenir le contexte dans la prédiction d'un résultat [34].

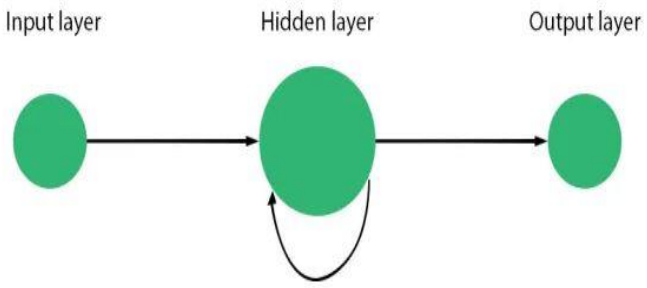


Figure 23 : Réseau de neurone récurrent [35]

RNN peut être utilisé pour résoudre des problèmes liés à [34] :

- Données du texte
- Données vocales
- Prédiction de classification
- Prédiction de régression

3.8 Conclusion

Dans ce chapitre, nous présenterons l'apprentissage en profondeur (DL) et le réseau de neurones profonds (DNN), c'est-à-dire des réseaux de neurones avec plusieurs couches cachées.

Ainsi que ces différents types le réseau de neurones artificiels "ANN", Réseau de neurones Convolutifs "CNN" et Réseau de neurones récurrents "RNN".

4 CHAPITRE : Conception et Implémentation

4.1 Introduction

Dans ce quatrième chapitre du mémoire nous mettons en avant notre conception. Nous allons montrer comment réalisé notre classificateur et créer notre modèle. Dans la partie implémentationet interprétation, de l'importation des bibliothèques nécessaires et importantes jusqu'à l'obtention des résultats. Et avant ça nous allons définir brièvement l'environnement du développement et les outils utilisés pour réaliser ce travail.

4.2 Conception

Avant d'entrer dans l'implémentation, Nous allons expliquer comment concevoir notre programme.

La figure ci-dessus est l'architecture proposée pour les différentes étapes du modèle.

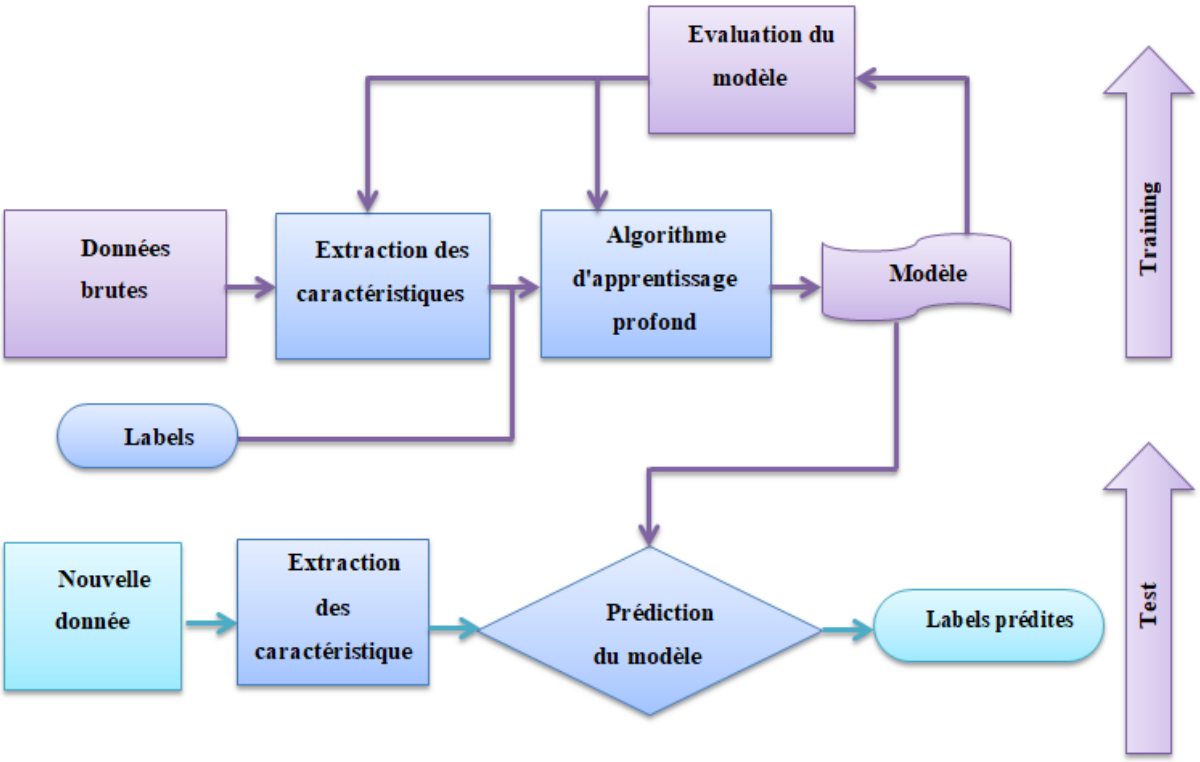


Figure 24 : Architecture de l'application

On remarque qu'il Ya deux grands processus qui englobent notre conception :

a) **Phase d'apprentissage :** consiste à générer la base de données des gènes de levures comme des données brutes. Ensuite, Nous ferons l'extraction des caractéristiques de la base d'apprentissage afin d'appliquer l'algorithme d'apprentissage profond(DNNs) pour la création du modèle qui va par la suite prédire les labels.

b) **Phase test :** Pour toute nouvelle donnée X à classifier au futur, nous faisons l'extraction des caractéristiques de la base de test, pour pouvoir mieux tester notre modèle et avoir une bonne évaluation.

4.3 Outils utilisés

Le travail a été établi avec un micro-ordinateur ASSUS 5éme génération avec une Ramde 8GB et un processeur i5.

4.3.1 Anaconda Navigator

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

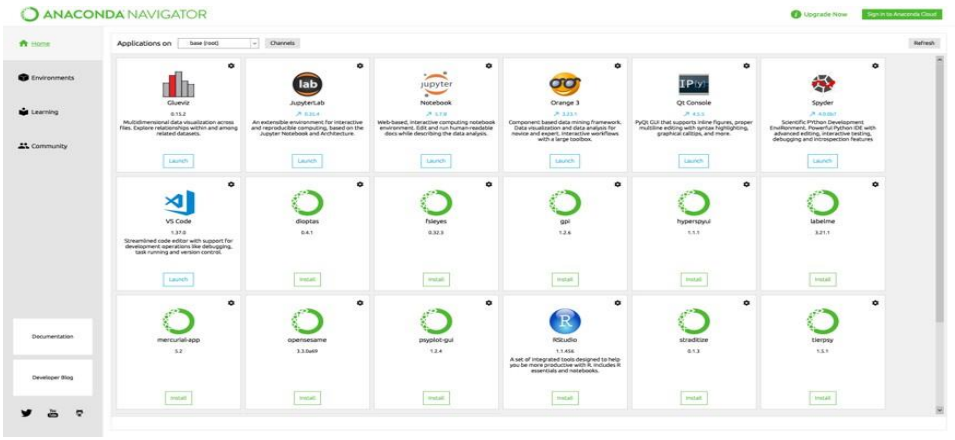


Figure 25 : Interface Anaconda Navigator

4.3.2 Jupyter Notebook

Jupyter Notebook est une application Web Open Source permet de créer et de partager des documents contenant du code (exécutable directement dans le document), peut rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web. Il est possible de faire du traitement de données, de la modélisation statistique, de la visualisation de données, du Machine Learning, etc. Ainsi, que Google Colaboratory c'est sur cet outil qu'est basé [37].

L'interface se présentera ainsi :

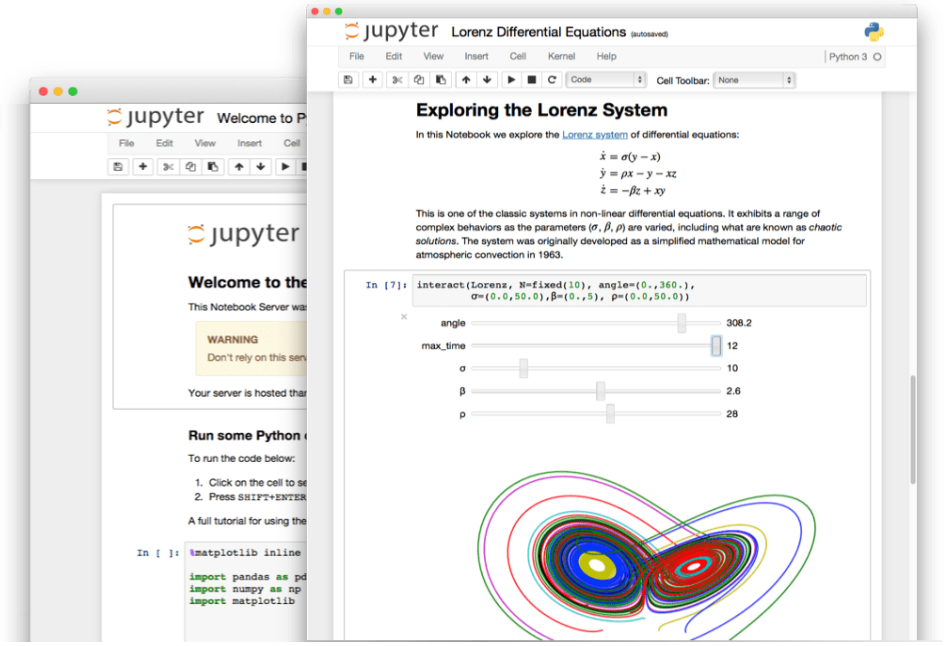


Figure 26: Fenêtre représentant Jupyter Notebook [37]

4.3.3 Python

Python est un langage de programmation multiplateforme open source, orienté objet cela, par rapport à ses concurrents directs (par exemple, C ++ et Java), est très concis.

Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. C'est également un langage à usage général, et il est en effet très flexible en raison d'une grande variété de packages disponibles qui résolvent un large éventail de problèmes et de nécessités. L'interpréteur Python et sa vaste bibliothèque standard sont disponibles librement, sous forme de sources ou de binaires, pour toutes les plateformes majeures depuis son site officiel et peuvent être librement redistribués [38].

4.4 Etapes de l'implémentation

La phase implémentation se compose de différentes étapes :

- Importation des bibliothèques.
- Importation du dataset.
- Prétraitement des données.
- Phase d'apprentissage.
- Phase de test.
- Evaluation et interprétation.

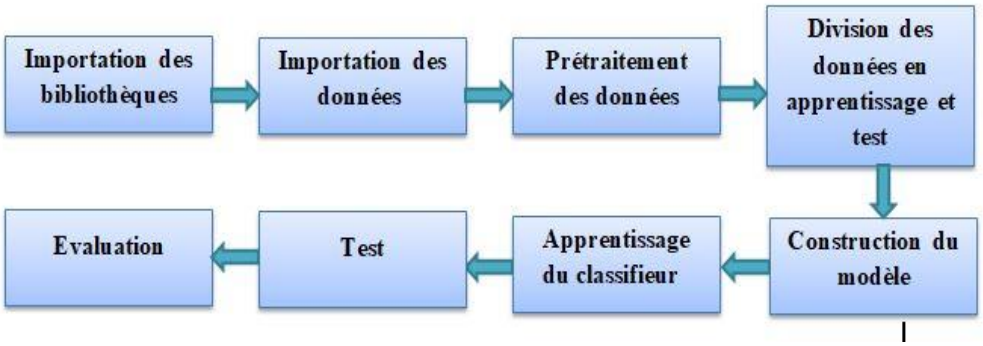


Figure 27 : Etapes d'implémentation

4.4.1 Importation des bibliothèques nécessaires

La première partie du programme est l'importation des bibliothèques nécessaires pour mener à bien les différentes tâches. Pour ce faire nous avons eu recours à des bibliothèques principales ci-dessous

a) Numpy

Numpy est une bibliothèque open source associée au langage Python, fournit à l'utilisateur des tableaux multidimensionnels, tout en offrant de multiples fonctions permettant la création et la manipulation de matrices, vecteurs, etc [32].

b) Pandas

Le package pandas traite de tout ce que NumPy ne peut pas faire. Grâce à ses structures de données d'objets spécifiques, DataFrames et Series, pandas permet de gérer des tableaux complexes de données de différents types (ce que les tableaux de NumPy ne peuvent pas faire) et de séries chronologiques [39].

c) Scikit-Learn

Scikit-learn est une librairie pour Python spécialisée dans la machine Learning (apprentissage automatique), Il offre des outils de prétraitement des données, implémente de nombreux algorithmes de classification, d'apprentissage supervisé et non supervisé, de sélection de modèle, de validation et de mesures d'erreur [32].

d) Keras

Keras est une API de réseau neuronal de haut niveau écrite en Python. Le but de son développement est de permettre des expérimentations rapides et de prendre en charge les réseaux convolutifs et récurrents et une combinaison des deux [39].

e) Matplotlib

Matplotlib est une bibliothèque Python capable de générer des graphiques de qualité. Matplotlib peut être utilisé dans les scripts Python, Python et IPython Shell, les notebooks Jupyter et les serveurs d'applications Web. Elle essaie et de rendre les choses complexes possibles. Vous pouvez utiliser quelques lignes de code pour générer des graphiques, des histogrammes, des graphiques à barres, des graphiques en nuage de points, etc [40] .

Toutes ces bibliothèques fait partie de la distribution Anaconda et peut être installé via pip depuis Anaconda prompt [41].

- ***Pip install scikit-learn***
- ***Pip install pandas***
- ***Pip install Matplotlib***

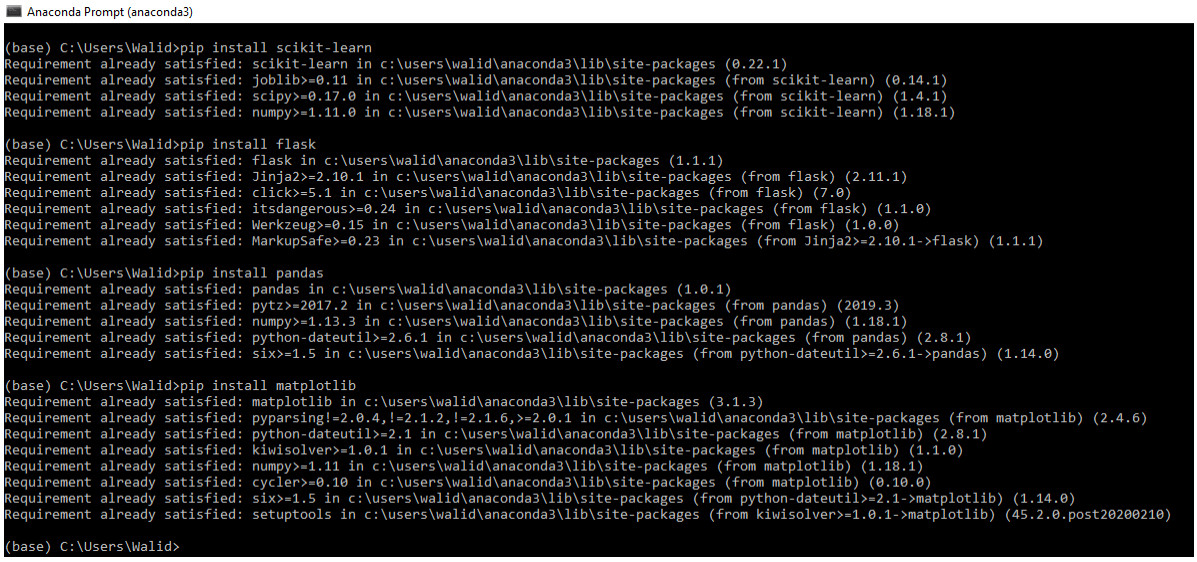


Figure 28: Installation des bibliothèques

4.4.2 Importation des données

La classification multi-label s'applique à plusieurs types de données de plusieurs domaines (médicine, biologie, chimie, marketing etc.).

Dans ce mémoire nous allons utiliser la base de données **des gènes de levures,** Il s'agit d'un ensemble de données qui contient des expressions de micro-array et des profils phylogénétiques pour 1500 gènes de levures. Chaque gène est annoté avec un sous-ensemble de 14 catégories fonctionnelles (par exemple, métabolisme, énergie, etc.) du niveau supérieur du catalogue fonctionnel.

Vous pouvez télécharger l'ensemble de données directement à partir de ce lien (Multi-Label Classification Dataset Repository - Knowledge Discovery and Intelligent Systems - KDIS - University of Córdoba)

Il y a deux fichiers de données :

1. csv_result-yeast-train.xlsx
2. csv_result-yeast-test.xlsx

csv_result-yeast-train.xlsx : Un fichier de taille 1.57 Mo, qui contient 1500 lignes (un identifiant unique pour chaque instance), et 118 colonnes dont 103 concernant les attributs et les 14 colonnes pour les classes (labels).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
id	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13	Att14	Att15
1	0.0937	0.139771	0.062774	0.007698	0.083873	-0.119156	0.073305	0.00551	0.027523	0.043477	-0.024946	0.061221	0.147377	0.082805	-0.011043
2	-0.022711	-0.050504	-0.035691	-0.065434	-0.084316	-0.37856	0.038212	0.08577	0.182613	-0.055544	0.029267	0.042597	-0.107352	0.019207	0.047062
3	-0.090407	0.021198	0.208712	0.102752	0.119315	0.041729	-0.021728	0.019603	-0.063853	-0.053756	0.078468	0.130276	0.082742	-0.041696	-0.028589
4	-0.085235	0.00954	-0.013228	0.094063	-0.013592	-0.030719	-0.116062	-0.131674	-0.165448	-0.123053	-0.088342	-0.01067	0.03059	0.120052	0.129741
5	-0.088765	-0.026743	0.002075	-0.043819	-0.005465	0.004306	-0.055865	-0.071484	-0.159025	-0.111348	-0.113015	-0.151546	-0.224416	-0.193918	-0.097713
6	0.052386	-0.077969	-0.065555	-0.044628	-0.005428	0.120818	0.05185	0.072627	0.107119	0.034214	0.067433	-0.029847	0.027771	0.021908	0.094492
7	-0.162278	-0.052898	-0.046867	0.021493	0.008256	0.050265	-0.136019	0.035562	-0.035445	-0.068137	-0.03787	0.028775	-0.042031	0.044432	-0.050701
8	-0.075051	-0.062277	0.21707	0.250216	0.068413	0.001721	-0.076456	-0.094136	-0.182685	0.012269	0.017489	0.240197	0.08313	0.012974	-0.114525
9	0.014962	0.063235	-0.001107	0.06586	-0.032007	0.054317	-0.048934	0.030914	0.037152	0.09808	0.031204	-0.011976	0.028459	0.09424	0.085284
10	0.087604	0.032477	0.143112	0.150504	0.167052	0.060122	-0.085561	0.052064	0.079	-0.102476	0.147377	-0.231194	-0.002659	-0.031625	0.117188
11	0.372552	0.247687	0.275981	0.086757	0.037926	-0.030495	-0.007161	-0.021243	-0.040741	0.098187	-0.0462	0.203584	-0.010554	-0.103253	0.016765
12	-0.000628	-0.063052	0.062563	-0.001431	0.005861	0.050742	0.001052	-0.086762	-0.01616	-0.038526	-0.062467	-0.198941	-0.074132	-0.054846	-0.008952
13	0.015089	0.073954	0.023296	-0.093508	0.02538	-0.062028	0.040267	-0.060581	0.032808	0.095193	-0.028791	-0.131345	-0.053908	-0.027933	0.052347
14	-0.084183	-0.129782	-0.195234	-0.124294	-0.021362	-0.070877	0.095374	-0.11933	-0.035861	-0.090628	-0.134259	-0.163965	-0.071448	-0.154969	-0.089646
15	0.015501	-0.100177	-0.225022	-0.075194	-0.09573	-0.109448	-0.073858	-0.067576	0.05209	0.051013	0.102447	-0.014787	-0.042075	-0.041252	0.092385
16	0.119198	0.143875	0.24599	0.051389	-0.005277	0.031344	0.000589	0.054193	0.071066	0.044431	-0.014258	-0.011458	-0.088157	-0.0407	-0.22499
17	-0.003018	-0.037259	-0.130894	-0.082866	-0.042685	-0.134401	-0.024021	0.097234	0.000858	0.105475	-0.02414	0.180063	0.073987	0.10628	0.024177
18	0.140251	-0.025495	0.005217	0.018019	0.026697	-0.014894	0.080701	0.185063	0.0728	0.081616	0.102023	0.098425	0.099434	0.082942	0.025323
19	0.069755	-0.056199	0.142758	-0.147294	0.185114	-0.00504	0.070308	0.076543	0.050768	0.014497	0.389611	0.019378	-0.064873	-0.017034	0.036598

DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN
Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Class10	Class11	Class12	Class13	Class14
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	1	1	0
0	1	1	0	0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	0	0	0	0	1	1	0
0	1	1	0	1	1	1	1	0	0	0	1	1	0
0	1	1	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	1	1	1
0	0	0	1	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	1	0

Figure 29 : Aperçue de la dataset "Train"

csv_result-yeast-test.xlsx : Un fichier de taille 1.05 Mo, qui contient 917 lignes (un identifiant unique pour chaque instance), et 118 colonnes dont 103 concernant les attributs et les 14 colonnes pour les classes (labels)**.**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
id	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13	Att14	Att15
1	0.004168	-0.170975	-0.156748	-0.142151	0.058781	0.026851	0.197719	0.04185	0.066938	-0.056617	-0.02723	-0.137411	0.067776	0.047175	0.155671
2	-0.103956	0.011879	-0.098986	-0.054501	-0.00797	0.049113	-0.03058	-0.077933	-0.080529	-0.016267	-0.215304	-0.009885	-0.155843	-0.059522	-0.098836
3	0.509949	0.401709	0.293799	0.087714	0.011686	-0.006411	-0.006255	0.013646	-0.040666	-0.024447	-0.040576	0.014326	-0.074968	0.141365	-0.015182
4	0.119092	0.004412	-0.002262	0.072254	0.044512	-0.051467	0.074686	-0.00767	0.079438	0.062184	-0.013027	0.045538	0.080412	-0.010042	0.013029
5	0.042037	0.007054	-0.069483	0.081015	-0.048207	0.089446	-0.004947	0.064456	-0.133387	0.068878	-0.139371	0.041487	-0.058531	0.021264	-0.101382
6	0.003286	-0.023601	-0.023104	-0.064171	-0.017719	-0.11351	-0.035212	-0.077675	-0.024348	-0.012458	0.013203	-0.033329	0.047343	0.03111	0.066102
7	0.076606	-0.025478	0.072419	-0.065383	0.123539	0.12013	0.102248	0.008324	0.094147	-0.014666	0.159352	-0.005371	0.071226	0.110422	0.17914
8	-0.106801	-0.069277	-0.132254	-0.099246	-0.188413	-0.110359	-0.15512	-0.168094	-0.182473	0.023208	-0.130712	-0.079053	-0.171518	-0.064736	-0.153275
9	0.17765	0.136159	0.153211	0.190312	0.158541	0.219521	0.079914	0.232238	0.127932	0.089142	0.214214	0.186596	0.080546	0.214809	0.157049
10	0.122075	-0.055828	-0.034775	-0.039235	0.075027	0.087308	0.088062	0.122385	0.091848	-0.099132	0.041763	-0.024225	0.118928	0.057956	0.027388
11	0.027604	0.147904	0.040762	-0.062932	0.073776	0.042128	0.035798	-0.032456	-0.023777	0.058981	-0.097121	-0.037276	0.04938	0.08271	0.060235
12	-0.060498	-0.045939	0.005694	-0.00803	-0.053844	-0.14502	-0.120767	-0.204572	-0.135548	-0.223845	-0.108189	-0.090217	-0.122463	-0.123234	-0.071434
13	-0.107407	-0.008558	-0.049246	-0.073517	-0.173126	-0.047933	-0.112671	-0.150253	-0.04629	-0.023223	-0.052437	-0.113562	-0.172783	-0.216828	-0.163574
14	0.040022	0.055742	-0.039405	0.050275	0.013602	0.050647	0.029556	-0.055486	0.018907	0.095781	0.021203	0.046395	0.043224	0.150822	0.079715
15	-0.096449	0.046531	-0.052701	-0.020124	-0.166082	0.079298	-0.148116	0.068355	-0.064213	0.05836	-0.014423	-0.059671	-0.089284	-0.007239	-0.082862
16	-0.194145	-0.08313	-0.094892	-0.046649	-0.137844	0.112074	-0.26929	-0.078088	0.031018	0.050656	-0.091382	0.133081	-0.156543	-0.165285	-0.198917
17	0.130601	0.068018	0.191105	-0.003937	0.044457	-0.15091	0.010224	-0.071669	0.054135	0.042311	0.007292	0.088242	0.074419	-0.075424	-0.054123
18	-0.018454	0.060603	0.07227	0.07788	-0.086323	-0.078543	-0.198041	-0.014758	-0.112972	-0.146407	-0.025857	-0.096086	-0.061599	-0.040123	-0.039163
19	0.053627	0.086194	-0.048808	0.064847	-0.039971	0.046727	0.02186	0.084702	0.077895	0.031315	0.043681	0.067544	0.116453	0.093862	0.146331

DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN
Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Class10	Class11	Class12	Class13	Class14
0	0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	1	1	0
1	0	1	1	0	0	0	0	0	0	0	1	1	0
1	1	0	0	0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	1	1	1	1	0
0	0	0	0	1	1	0	0	0	0	0	1	1	0
1	1	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	0	0	0	0	1	1	0

Figure 30 : Aperçue de la dataset "Test"

4.4.3 Prétraitement des données

Le prétraitement des données est une étape très importante qui consiste à nettoyer, éliminer, identifier et corriger les informations altérées, inexactes ou non pertinentes. Cette phase fondamentale du traitement des données a pour but d'améliorer la cohérence, fiabilité et valeur des données.

4.4.3.1 Importation des données et la lecture du data set

La figure suivante contient un pseudo code de la lecture de la base et l'affichage de sa taille, la variable **"Yeast"** c'est le nom de dataset.

```
Yeast = pd.read_excel('csv_result-yeast-train.xlsx')
Yeast.head()
```

]:

	id	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	...	Class5	Class6	Class7	Class8	Class9
0	1	0.093700	0.139771	0.062774	0.007698	0.083873	-0.119156	0.073305	0.005510	0.027523	...	0	0	0	0	0
1	2	-0.022711	-0.050504	-0.035691	-0.065434	-0.084316	-0.378560	0.038212	0.085770	0.182613	...	0	0	1	1	0
2	3	-0.090407	0.021198	0.208712	0.102752	0.119315	0.041729	-0.021728	0.019603	-0.063853	...	0	0	0	0	0
3	4	-0.085235	0.009540	-0.013228	0.094063	-0.013592	-0.030719	-0.116062	-0.131674	-0.165448	...	0	0	0	0	0
4	5	-0.088765	-0.026743	0.002075	-0.043819	-0.005465	0.004306	-0.055865	-0.071484	-0.159025	...	0	0	0	0	0

5 rows × 118 columns

```
print("Yeast.shape: " + str(Yeast.shape))
```

```
Yeast.shape: (1500, 118)
```

Figure 31 : La taille de la base

L'indexeur iloc pour Pandas Dataframe est utilisé pour l'indexation basée sur les emplacements entiers / la sélection par position.

La syntaxe de l'indexeur iloc est data.iloc[<sélection de ligne>, <sélection de colonne>].

Dans pandas, "iloc" est utilisé pour sélectionner les lignes et les colonnes par numéro, dans l'ordre où elles apparaissent dans le cadre de données.

Il y a deux "arguments" à iloc - un sélecteur de ligne, et un sélecteur de colonne.

```
X = Yeast.iloc[:, :104]
y = Yeast.iloc[:, 104:]
print("X.shape: " + str(X.shape))
display(X.head())
print("y.shape: " + str(y.shape))
display(y.head())
print("Descriptive stats:")
X.describe()
```

```
X.shape: (1500, 104)
```

	id	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	...	Att94	Att95	Att96	Att97
0	1	0.093700	0.139771	0.062774	0.007698	0.083873	-0.119156	0.073305	0.005510	0.027523	...	0.039048	-0.018712	-0.034711	-0.038675
1	2	-0.022711	-0.050504	-0.035691	-0.065434	-0.084316	-0.378560	0.038212	0.085770	0.182613	...	-0.001198	0.030594	-0.021814	0.010430
2	3	-0.090407	0.021198	0.208712	0.102752	0.119315	0.041729	-0.021728	0.019603	-0.063853	...	0.195777	0.022294	0.012583	0.002233
3	4	-0.085235	0.009540	-0.013228	0.094063	-0.013592	-0.030719	-0.116062	-0.131674	-0.165448	...	0.001189	-0.066241	-0.046999	-0.066604
4	5	-0.088765	-0.026743	0.002075	-0.043819	-0.005465	0.004306	-0.055865	-0.071484	-0.159025	...	-0.035045	-0.080882	0.028468	-0.073576

5 rows × 104 columns

```
y.shape: (1500, 14)
```

	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Class10	Class11	Class12	Class13	Class14
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	1	1	0
2	0	1	1	0	0	0	0	0	0	0	0	1	1	0
3	0	0	1	1	0	0	0	0	0	0	0	1	1	1
4	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32 : Sélection de données" iloc Pandas"

4.4.4 Division de données en apprentissage et test

Cette étape du programme consiste à la division des données en deux parties : une partie pour l'apprentissage et une partie pour le test. Pour ce faire nous avons utilisé la fonction train_test_split de sklearn dont nous avons paramétré de tel sorte que la partie test concerne 30% des données et donc 70% pour la partie d'apprentissage (train).

Ainsi une fois la division faite, les parties x_train et x_test concerneront la colonne qui contient le résumé (prétraité) des attributs et les parties y_train et y_test concerneront les labels.

```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

```
print("X_train.shape: " + str(x_train.shape))
print("X_test.shape: " + str(x_test.shape))
print("y_train.shape: " + str(y_train.shape))
print("y_test.shape: " +  str(y_test.shape))
```

```
X_train.shape: (1050, 104)
X_test.shape: (450, 104)
y_train.shape: (1050, 14)
y_test.shape: (450, 14)
```

Figure 33 : Partie des données Train et Test

4.4.4.1 Convertir un tableau Numpy en liste

Nous utilisons la méthode tolist () pour convertir un tableau Numpy en liste .Aussi nous pouvons avoir une liste d'éléments de données qui est convertie à partir d'un tableau en utilisant cette méthode.

```
y_train_list = y_train.values.tolist()
y_train = np.array(y_train_list)
```

```
y_train
```

```
array([[0, 0, 1, ..., 1, 1, 0],
       [0, 0, 0, ..., 1, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 1, 1, 0],
       [1, 0, 0, ..., 1, 1, 0],
       [0, 0, 0, ..., 1, 1, 0]])
```

```
y_test_list = y_test.values.tolist()
y_test = np.array(y_test_list)
```

```
y_test
```

```
array([[0, 1, 1, ..., 1, 1, 0],
       [0, 1, 1, ..., 1, 1, 0],
       [0, 1, 1, ..., 1, 1, 0],
       ...,
       [0, 0, 1, ..., 1, 1, 0],
       [0, 0, 0, ..., 1, 1, 0],
       [1, 0, 0, ..., 1, 1, 0]])
```

Figure 34 : Convertir tableau Numpy en liste

4.4.5 Phase d'apprentissage du modèle

En Deep Learning, l'algorithme se construit une "représentation interne" afin de pouvoir effectuer les tâches requises (prédiction, reconnaissance, etc.), vous devez d'abord entrer un ensemble d'exemples de données afin qu'il puisse s'entraîner et s'améliorer, ce jeu de données s'appelle le training set.

La construction du modèle DNN en python se compose de plusieurs couches (couche d'entrée, couche cachées, couche de sortie) Keras propose :

a. **Couche d'entrée :** les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transmettent aux neurones cachés

b. **Couche dense cachée :**

Définir le nombre de nœuds qu'on veut, on a choisir 64 neurones dans la première couche cachée et 128 neurones dans la deuxième couche cachée.

c. **Couche dense de sortie :**

Pour une couche de sortie on a choisir 14 neurones (classes).

```
Entrée [11]:
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense

Using TensorFlow backend.

Entrée [12]:
model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=len(x_train.columns)))
model.add(Dense(units=128, activation='relu'))

model.add(Dense(len(y_train[0]), activation='sigmoid'))

Entrée [13]:
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Entrée [14]:
model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                6720
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_3 (Dense)              (None, 14)                1806
=================================================================
Total params: 16,846
Trainable params: 16,846
Non-trainable params: 0
```

Figure 35 : Les différentes couches utilisées pour notre modèle

Maintenant que le modèle est défini, nous pouvons le compiler avant d'entraîner le modèle, il faut configurer le processus d'apprentissage en appelant la méthode compile qui accepte trois arguments

Loss : il s'agit de la fonction de coût que le modèle va utiliser pour minimiser les erreurs. Elle peut être définie par son appellation (exemple : binary_crossentropy).

Optimizer : a pour but de réduire l'erreur actuelle qui est définit dans loss (modifier les poids de neurones).

Metrics : les fonctions métriques sont similaires aux fonctions de perte, à quelle point notre réseau de neurone a juste sur les prédictions qui la faites.

4.4.6 Lancer l'apprentissage

Une fois le modèle construit et sauvegardé nous passons à l'étape suivante pour ce faire nous allons additionner une partie pour l'apprentissage et une partie pour le test que nous avons préparé et ensuite nous faisons l'apprentissage avec le total de ces données, X_train et X_test les données d'apprentissage et de test, Y_train et Y_test les données d'apprentissage et de test des classes.

Nous avons défini notre modèle et l'avons compilé pour qu'il soit prêt pour un calcul efficace, il est maintenant temps d'exécuter le modèle sur quelques données nous pouvons entraîner ou ajuster notre modèle sur nos données chargées en appelant la fonction fit () sur le modèle, l'entraînement se fait sur :

Epoch : on passe par toutes les lignes du jeu de données d'entraînement (les itérations) et chaque

Epoch est composée d'un ou plusieurs Batch, en fonction de la taille du Batch choisi et le modèle est adapté à de nombreuses époques.

Batch : un ou plusieurs échantillons considérés par le modèle au cours d'une Epoch avant que les poids ne soient mis à jour.

Le processus de formation se déroulera pendant un nombre fixe d'itérations à travers Epoch, nous devons également définir le nombre de lignes de l'ensemble de données qui sont prises en compte avant la mise à jour des poids du modèle dans chaque Epoch appelée taille du Batch et définie à l'aide de l'argument batch_size, pour ce problème nous allons fonctionner pour un petit nombre d'Epoch (150) et utiliser une taille de Batch (128).

Nous voulons entraîner le modèle suffisamment pour qu'il apprenne à faire une bonne (ou assez bonne) correspondance entre les lignes de données d'entrée et la classification de sortie. Le modèle comportera toujours une certaine erreur et précision, mais la quantité d'erreur se stabilisera après un certain point pour une configuration donnée du modèle. C'est ce qu'on appelle la convergence des modèles.

```
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=.3)
history = model.fit(x_train, y_train, validation_data=(x_test,  y_test), epochs=150, batch_size=128)
```

```
Train on 1050 samples, validate on 450 samples
Epoch 1/150
1050/1050 [==============================] - 1s 666us/step - loss: 10.1452 - accuracy: 0.5306 - val_loss: 2.6164 - val_ac
curacy: 0.7532
Epoch 2/150
1050/1050 [==============================] - 0s 51us/step - loss: 2.6108 - accuracy: 0.7000 - val_loss: 2.2380 - val_accu
racy: 0.7173
Epoch 3/150
1050/1050 [==============================] - 0s 50us/step - loss: 1.7036 - accuracy: 0.7363 - val_loss: 1.4666 - val_accu
racy: 0.5841
Epoch 4/150
1050/1050 [==============================] - 0s 49us/step - loss: 1.1108 - accuracy: 0.7003 - val_loss: 0.9372 - val_accu
racy: 0.7083
Epoch 5/150
1050/1050 [==============================] - 0s 57us/step - loss: 0.8533 - accuracy: 0.7037 - val_loss: 0.8540 - val_accu
racy: 0.7378
Epoch 6/150
1050/1050 [==============================] - 0s 53us/step - loss: 0.7191 - accuracy: 0.7392 - val_loss: 0.6891 - val_accu
racy: 0.7529
```

```
Epoch 145/150
1050/1050 [==============================] - 0s 49us/step - loss: 0.4479 - accuracy: 0.7972 - val_loss: 0.5338 - val_accu
racy: 0.7606
Epoch 146/150
1050/1050 [==============================] - 0s 46us/step - loss: 0.4484 - accuracy: 0.7965 - val_loss: 0.5175 - val_accu
racy: 0.7679
Epoch 147/150
1050/1050 [==============================] - 0s 49us/step - loss: 0.4474 - accuracy: 0.8003 - val_loss: 0.5343 - val_accu
racy: 0.7511
Epoch 148/150
1050/1050 [==============================] - 0s 50us/step - loss: 0.4571 - accuracy: 0.7904 - val_loss: 0.4864 - val_accu
racy: 0.7794
Epoch 149/150
1050/1050 [==============================] - 0s 48us/step - loss: 0.4388 - accuracy: 0.7989 - val_loss: 0.4844 - val_accu
racy: 0.7790
Epoch 150/150
1050/1050 [==============================] - 0s 51us/step - loss: 0.4355 - accuracy: 0.8031 - val_loss: 0.4928 - val_accu
racy: 0.7779
```

Figure 36 : L'entrainement du modèle en appelant la fonction fit ( )

La fonction fit () retournera une liste avec deux valeurs. La première sera la perte du modèle sur l'ensemble de données et la seconde sera la précision du modèle sur l'ensemble de données, on voit un message pour chacune des 150 Epochs imprimant la perte et la précision, suivi de l'évaluation finale du modèle formé sur l'ensemble de données de formation.

Idéalement, nous aimerions que la perte soit nulle et la précision de 1.0 (par exemple, 100%). Au lieu de cela, nous aurons toujours une certaine erreur dans notre modèle. L'objectif est de choisir une configuration de modèle et une configuration d'apprentissage qui permettent d'obtenir la perte la plus faible et la précision la plus élevée possible pour un ensemble de données donné, nous ne voulons que rapporter la précision, donc nous ignorerons la valeur de la perte.

4.4.6.1 Validation

Afin de montrer les résultats obtenus pour notre modèle, Nous illustrons dans ce qui suit les résultats en termes de précision et d'erreur.

Apres la création du model, Nous allons faire un test, et nous obtenons les deux graphes suivants :

Graphe de précision (Accuracy)

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

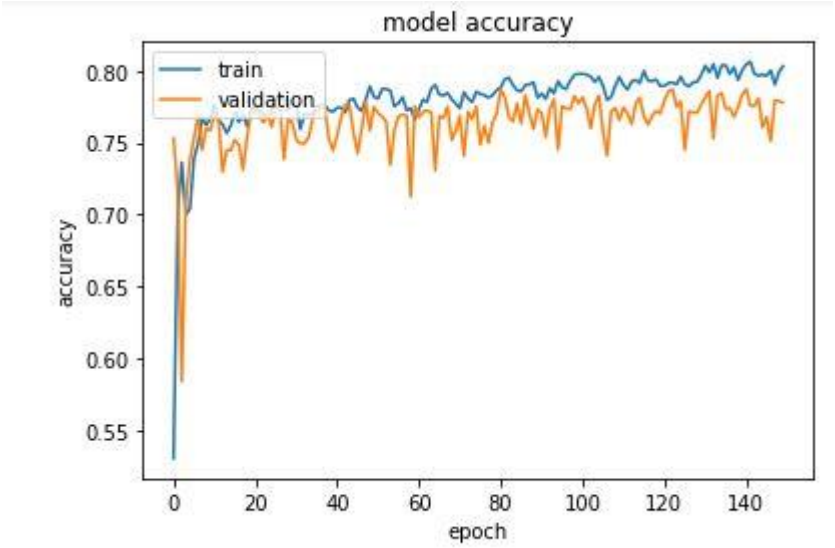


Figure 37 : Graphe de précision pour le modèle

D’après la Figure 37 La précision de l’apprentissage et de la validation augmente avec le nombre d’époques.

Graphe de perte (Loss)

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

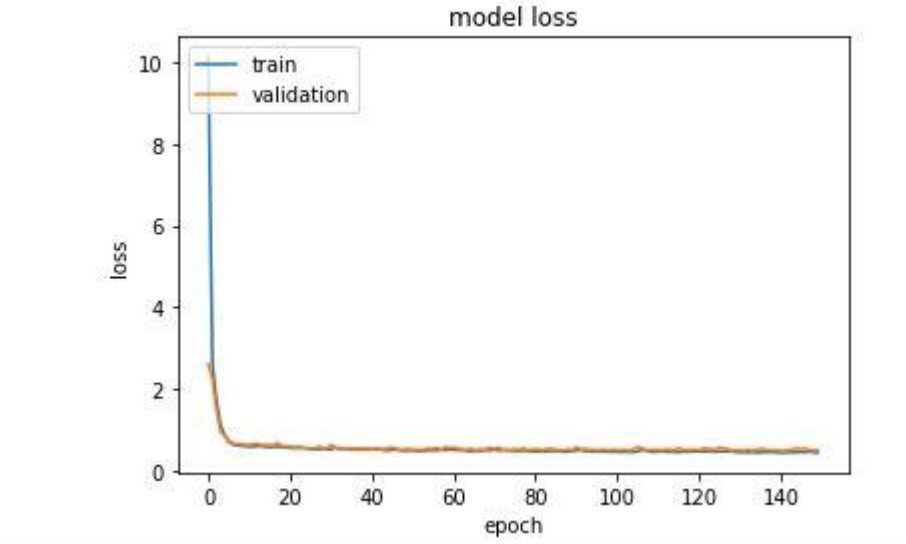


Figure 38 : Trace de la perte du modèle sur les ensembles de données d'entraînement et de validation

Ce graphe (figure 38) montre que, l’erreur d’apprentissage et de la validation diminue avec le nombre d’époques.

```
results = model.evaluate(x_test, y_test, verbose = 0)
print("Test Accuracy","{:.2%}".format(results[1]))
```

```
Test Accuracy 77.79%
```

Figure 39 : Précision du modèle

4.4.7 L'évaluation

Une fois qu’on a formé le modèle, comment puisse-nous l'utiliser pour faire des prévisions sur de nouvelles données ? Il s'agit d’appliquer un nouvel ensemble de données que le modèle n’a jamais

vu auparavant, faire des prédictions est aussi simple que d'appeler la fonction predict () sur le modèle. Le but de l'évaluation est d'estimer au mieux les performances d'un modèle sur de nouvelles données.

```
# Predict on test data using the model
labelsPredict = model.predict(xx)

# I'm going to call a label 1 if its value > 0.5 and so, im going to use round function to get labels
# Threshold could be set dynamically for better performance of the model
labelsPredict = labelsPredict.round()
labelsPredict
```

```
]: array([[0., 0., 0., ..., 1., 1., 0.],
       [0., 1., 1., ..., 1., 1., 0.],
       [0., 1., 1., ..., 1., 1., 0.],
       ...,
       [0., 0., 0., ..., 1., 1., 0.],
       [0., 0., 0., ..., 1., 1., 0.],
       [0., 0., 0., ..., 1., 1., 0.]], dtype=float32)
```

Figure 40 : Prédiction des labels

```
results = model.evaluate(xx,yy,verbose = 0)
print("Test Accuracy","{:.2%}".format(results[1]))
```

```
Test Accuracy 79.19%
```

Figure 41 : la précision totale du modèle sur les données d'apprentissage et de test

On observe au-dessus (Voir la Figure 41) que la précision de notre modèle sur les données d'entrainement est **79,19%.** D'après la figure ci-dessus on peut dire que notre modèle est très bon pour classifier.

4.5 Conclusion

Dans cette section du mémoire nous avons présenté les outils utilisés et les différentes parties de notre phase d'implémentation. Ainsi nous avons présenté quelques bibliothèques qui ont été indispensables et une approche de classification basée sur les réseaux neurones profonds.

Conclusion Générale et Perspectives

La classification multi-label est le domaine approprié où des méthodes dédiées sont étudiées et proposées pour remplir la tâche d'étiquetage des ressources en plusieurs catégories et où l''information est un atout essentiel pour améliorer la qualité de vie et le progrès social.

L'objectif principal de ce mémoire est de construire un modèle de classification multi label efficace qui est le réseau de neurones profonds (DNN), afin de prédire les différentes étiquettes.

Les démarches que nous avons eu à mener à savoir de la définition de l'apprentissage automatique qui est un sous domaine de l'intelligence artificielle et plus précisément l'apprentissage supervisé. Dans la partie classification, nous avons vu les différentes techniques et approches de la classification multi-label.

Pour réaliser notre travail de classification. Nous avons utilisé le Deep Learning, la méthode d`apprentissage qui a montré ses performances ces dernières années et nous avons choisi la méthode DNNs comme méthode de classification, ce choix est justifié par la simplicité et l'efficacité de la méthode. Le modèle obtenu a été testé sur une base de données qui est celle des gènes de levure.

L'implémentation a été faite avec le langage de programmation python et on a utilisé des bibliothèques pour faciliter la tâche de création de notre modèle et pour l'accélération du training.

Notre travail n`est que dans sa version initiale, nous pouvons dire que ce travail reste ouvert pour des travaux de comparaison avec d'autres méthodes de classification.

Comme perspectives de recherches futures, nous envisageons de :

1. De tester sur notre modèle d'autre base de données autre que celle de gènes de levures.
2. D'augmenter de nouvelles couches afin de tester les performances du système.
3. De travailler avec les modèles déjà entrainés et faire une comparaison.

Bibliographie

[1] F. e. a. Chollet, Deep learning with Python, New York: Manning , 2018.

[2] M. R. K. A. M. Giancarlo Zaccone, Deep Learning with TensorFlow: Explore neural networks with Python, Packt, 24 avril 2017.

[3] D. B. A. Z. LI, «Une brève histoire de l'apprentissage automatique en cybersécurité,» Infowatch de sécurité, 14 novembre 2019. [En ligne]. Available: https://stellarcyber.ai/fr/a-brief-history-of-machine-learning-in-cybersecurity/. [Accès le 15 05 2021].

[4] D. k. Judith Hurwitz, Machine Learning, IBM, 2018.

[5] A. E. DJOKHRAB, «Planification et Optimisation de Trajectoire d'un Robot Manipulateur à 6 D. D. L. par des Techniques Neuro-Floues,» Masters thesis, Université Mohamed Khider, Biskra, 2015.

[6] O. M. KALOUNE Salim, «Classification d'objets avec le Deep Learning,» Université Akli Mohand Oulhadj de Bouira Faculté des Sciences et des Sciences Appliquées, Bouira, 2018.

[7] Y. Benzaki, «L'apprentissage non supervisé – Machine Learning,» 01 Mars 2017. [En ligne]. Available: https://mrmint.fr/lapprentissage-non-supervise-machine-learning. [Accès le 17 05 2021].

[8] «apprentissage-supervise-vs-non-supervise,» [En ligne]. Available: https://analyticsinsights.io/apprentissage-supervise-vs-non-supervise/. [Accès le 18 Mai 2021].

[9] «machine-learning-classification-vs-regression,» 19 Juillet 2019. [En ligne]. Available: https://dev.to/petercour/machine-learning-classification-vs-regression-1gn. [Accès le 19 Mai 2021].

[10] «Getting-started-with-classification,» 09 Février 2018. [En ligne]. Available: https://www.geeksforgeeks.org/getting-started-with-classification/. [Accès le 20 Mai 2021].

[11] «Classification binaire,» 01 Avril 2021. [En ligne]. Available: https://datafranca.org/wiki/Classification_binaire. [Accès le 21 05 2021].

[12] «Multiclass and multioutput algorithms,» 2017. [En ligne]. Available: https://scikit-learn.org/stable/modules/multiclass.html. [Accès le 22 05 2021].

[13] O. Aristide, «Classification de Texte multi-label en utilisant la,» Université Badji Mokhtar Département d'informatique, annaba, 2020.

[14] «Multi-Label classification with One-Vs-Rest strategy,» 20 Février 2019. [En ligne]. Available: https://prakhartechviz.blogspot.com/2019/02/multi-label-classification-python.html. [Accès le 22 05 2021].

[15] V. S. GIBAJA Eva, « A tutorial on multilabel learning,» 2015.

[16] Z. C. Lipton, «Multilabel Classification,» Université de Californie, San Diego.

[17] N.-Y. NAIR-BENREKIA, «Classification interactive multi-label pour l'aide à l'organisation personnalisée des données,» Université de Nantes, France, 2015.

[18] P. Brandt, «AN INVESTIGATION OF MULTI-LABEL CLASSIFICATION,» University of KwaZulu-Natal, Durban, South Africa, 2014.

[19] E. Rostom, «Unsupervised Clustering and Multi-Label,» L'université libre Département de mathématiques-'informatique, Berlin, 2018.

[20] B. Assia, «Contribution en apprentissage semi-supervisé,» Université des Sciences et de la Technologie Département Informatique, Oran, 2018.

[21] «Initiez-vous-au-machine-learning,» 08 Avril 2021. [En ligne]. Available: https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4022441-entrainez-votre-premier-k-nn. [Accès le 22 05 2021].

[22] A. Fatima, «Optimisation des SVM multiclasses par des m´ethodes,» Universit´e des Sciences et de la Technologie Departement d'Informatique, Oran.

[23] E. L. G. J. C. V. Jose M. Moyano, «Review of ensembles of multi-label classifiers: Models, experimental studyand prospects,» vol. 44, n° %133-45, 2018.

[24] L. R. N. Y. Tapp Alain, «Introduction à l'apprentissage profond,» 06 Juin 2018. [En ligne]. Available: https://catalogue.edulib.org/fr/cours/IVADO-IA-101/. [Accès le 23 Mai 2021].

[25] «Réseau de neurones artificiels : quelles sont leurs capacités ?,» 10 Mars 2020. [En ligne]. Available: https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-quun-reseau-neuronal-artificiel/. [Accès le 23 Mai 2021].

[26] B. K. MERDOUD Kenza, «Détection de maladies par traitement d'image,» Faculté des Sciences et des Sciences Appliquées Département d'informatique, Bouira, 2018.

[27] «apprentissage-automatique-et-apprentissage-profond,» [En ligne]. Available: https://www.stemmer-imaging.com/fr-ch/conseil-techniqu/apprentissage-automatique-et-apprentissage-profond/. [Accès le 25 Mai 2021].

[28] «Deep Neural Network : qu'est-ce qu'un réseau de neurones profond ?,» 06 Avril 2021. [En ligne]. Available: https://datascientest.com/deep-neural-network. [Accès le 26 Mai 2021].

[29] «RÉSEAUX DE NEURONES ARTIFICIELS – ANN,» 24 Octobre 2019. [En ligne]. Available: https://www.tellaw.org/ia-machine-learning/reseaux-de-neurones-artificiels-ann-partie-1-la-theorie/. [Accès le 28 Mai 2021].

[30] Z. Lynda, «Contribution à la prévision de la séchresse par le modèle du réseau de neurones autorégressif,» ECOLE NATIONALE SUPERIEURE D'HYDRAULIQUE -ARBAOUI AbdellahDEPARTEMENT AMENAGEMENT ET GENIE HYDRAULIQUE, Blida, 2017.

[31] «Les réseaux de neurones,» [En ligne]. Available: https://ml4a.github.io/ml4a/fr/neural_networks/. [Accès le 29 Mai 2021].

[32] A. Mendjel, «La prédiction des familles de protéines en utilisant le réseau,» Université Badji Mokhtar Département d'informatique, Annaba, 2020.

[33] «Convolutional-Neural-Network,» 25 Juin 2020. [En ligne]. Available: https://datascientest.com/convolutional-neural-network. [Accès le 01 Juin 2021].

[34] «Réseaux de neurones récurrents,» La Rédaction TechTarget, Juillet 2019. [En ligne]. Available: https://www.lemagit.fr/definition/Reseaux-de-neurones-recurrents. [Accès le 01 Juin 2021].

[35] B. Dickson, «What are recurrent neural networks (RNN)?,» 08 Juin 2020. [En ligne]. Available: https://bdtechtalks.com/2020/06/08/what-is-recurrent-neural-network-rnn/. [Accès le 02 Juin 2021].

[36] «Anaconda,» IA, 01 Octobre 2018. [En ligne]. Available: https://intelligence-artificielle.agency/anaconda/. [Accès le 03 Juin 2021].

[37] H. Michel, «Google Colab : Le Guide Ultime,» 04 Novembre 2019. [En ligne]. Available: https://ledatascientist.com/google-colab-le-guide-ultime/. [Accès le 04 Juin 2021].

[38] «Python,» 25 Février 2021. [En ligne]. Available: https://www.lebigdata.fr/python-langage-definition. [Accès le 04 Juin 2021].

[39] «Pandas,» [En ligne]. Available: https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/. [Accès le 05 Juin 2021].

[40] «Matplotlib,» 2017. [En ligne]. Available: https://he-arc.github.io/livre-python/matplotlib/index.html#:~:text=matplotlib%20est%20une%20biblioth%C3%A8que%20Python,quatre%20outils%20d'interface%20graphique.&text=Pour%20des%20graphiques%20simples%2C%20le%20module%20matplotlib.. [Accès le 5 Juin 2021].

[F. Fasmeyer, «matplotlib,» 2017. [En ligne]. Available: https://he-arc.github.io/livre-
4 python/matplotlib/index.html. [Accès le 06 Juin 2021].
1
]

Webographie

[F. e. a. Chollet, Deep learning with Python, New York: Manning , 2018.
1
]
[M. R. K. A. M. Giancarlo Zaccone, Deep Learning with TensorFlow: Explore neural networks with
2 Python, Packt, 24 avril 2017.
]
[D. B. A. Z. LI, «Une brève histoire de l'apprentissage automatique en cybersécurité,» Infowatch de
3 sécurité, 14 novembre 2019. [En ligne]. Available: https://stellarcyber.ai/fr/a-brief-history-of-machine-
] learning-in-cybersecurity/. [Accès le 15 05 2021].
[D. k. Judith Hurwitz, Machine Learning, IBM, 2018.
4
]
[A. E. DJOKHRAB, «Planification et Optimisation de Trajectoire d'un Robot Manipulateur à 6 D. D. L. par
5 des Techniques Neuro-Floues,» Masters thesis, Université Mohamed Khider, Biskra, 2015.
]
[O. M. KALOUNE Salim, «Classification d'objets avec le Deep Learning,» Université Akli Mohand
6 Oulhadj de Bouira Faculté des Sciences et des Sciences Appliquées, Bouira, 2018.
]
[Y. Benzaki, «L'apprentissage non supervisé – Machine Learning,» 01 Mars 2017. [En ligne]. Available:
7 https://mrmint.fr/lapprentissage-non-supervise-machine-learning. [Accès le 17 05 2021].
]
[«apprentissage-supervise-vs-non-supervise,» [En ligne]. Available:
8 https://analyticsinsights.io/apprentissage-supervise-vs-non-supervise/. [Accès le 18 Mai 2021].
]
[«machine-learning-classification-vs-regression,» 19 Juillet 2019. [En ligne]. Available:
9 https://dev.to/petercour/machine-learning-classification-vs-regression-1gn. [Accès le 19 Mai 2021].

]
[«Getting-started-with-classification,» 09 Février 2018. [En ligne]. Available:
1 https://www.geeksforgeeks.org/getting-started-with-classification/. [Accès le 20 Mai 2021].
0
]
[«Classification binaire,» 01 Avril 2021. [En ligne]. Available:
1 https://datafranca.org/wiki/Classification_binaire. [Accès le 21 05 2021].
1
]
[«Multiclass and multioutput algorithms,» 2017. [En ligne]. Available: https://scikit-
1 learn.org/stable/modules/multiclass.html. [Accès le 22 05 2021].
2
]
[O. Aristide, «Classification de Texte multi-label en utilisant la,» Université Badji Mokhtar Département
1 d'informatique, annaba, 2020.
3
]
[«Multi-Label classification with One-Vs-Rest strategy,» 20 Février 2019. [En ligne]. Available:
1 https://prakhartechviz.blogspot.com/2019/02/multi-label-classification-python.html. [Accès le 22 05 2021].
4
]
[V. S. GIBAJA Eva, « A tutorial on multilabel learning,» 2015.
1
5
]
[Z. C. Lipton, «Multilabel Classification,» Université de Californie, San Diego.
1
6
]
[N.-Y. NAIR-BENREKIA, «Classification interactive multi-label pour l'aide à l'organisation personnalisée
1 des données,» Université de Nantes, France, 2015.
7
]
[P. Brandt, «AN INVESTIGATION OF MULTI-LABEL CLASSIFICATION,» University of KwaZulu-
1 Natal, Durban, South Africa, 2014.
8
]
[E. Rostom, «Unsupervised Clustering and Multi-Label,» L'université libre Département de mathématiques-
1 'informatique, Berlin, 2018.
9
]
[B. Assia, «Contribution en apprentissage semi-supervisé,» Université des Sciences et de la Technologie
2 Département Informatique, Oran, 2018.
0
]
[«Initiez-vous-au-machine-learning,» 08 Avril 2021. [En ligne]. Available:
2 https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4022441-entrainez-
1 votre-premier-k-nn. [Accès le 22 05 2021].
]
[A. Fatima, «Optimisation des SVM multiclasses par des m´ethodes,» Universit´e des Sciences et de la
2 Technologie Departement d'Informatique, Oran.
2

]
[E. L. G. J. C. V. Jose M. Moyano, «Review of ensembles of multi-label classifiers: Models, experimental
2 studyand prospects,» vol. 44, n° %133-45, 2018.
3
]
[L. R. N. Y. Tapp Alain, «Introduction à l'apprentissage profond,» 06 Juin 2018. [En ligne]. Available:
2 https://catalogue.edulib.org/fr/cours/IVADO-IA-101/. [Accès le 23 Mai 2021].
4
]
[«Réseau de neurones artificiels : quelles sont leurs capacités ?,» 10 Mars 2020. [En ligne]. Available:
2 https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-quun-reseau-neuronal-
5 artificiel/. [Accès le 23 Mai 2021].
]
[B. K. MERDOUD Kenza, «Détection de maladies par traitement d'image,» Faculté des Sciences et des
2 Sciences Appliquées Département d'informatique, Bouira, 2018.
6
]
[«apprentissage-automatique-et-apprentissage-profond,» [En ligne]. Available: https://www.stemmer-
2 imaging.com/fr-ch/conseil-techniqu/apprentissage-automatique-et-apprentissage-profond/. [Accès le 25
7 Mai 2021].
]
[«Deep Neural Network : qu'est-ce qu'un réseau de neurones profond ?,» 06 Avril 2021. [En ligne].
2 Available: https://datascientest.com/deep-neural-network. [Accès le 26 Mai 2021].
8
]
[«RÉSEAUX DE NEURONES ARTIFICIELS – ANN,» 24 Octobre 2019. [En ligne]. Available:
2 https://www.tellaw.org/ia-machine-learning/reseaux-de-neurones-artificiels-ann-partie-1-la-theorie/.
9 [Accès le 28 Mai 2021].
]
[Z. Lynda, «Contribution à la prévision de la séchresse par le modèle du réseau de neurones autorégressif,»
3 ECOLE NATIONALE SUPERIEURE D'HYDRAULIQUE -ARBAOUI AbdellahDEPARTEMENT
0 AMENAGEMENT ET GENIE HYDRAULIQUE, Blida, 2017.
]
[«Les réseaux de neurones,» [En ligne]. Available: https://ml4a.github.io/ml4a/fr/neural_networks/. [Accès
3 le 29 Mai 2021].
1
]
[A. Mendjel, «La prédiction des familles de protéines en utilisant le réseau,» Université Badji Mokhtar
3 Département d'informatique, Annaba, 2020.
2
]
[«Convolutional-Neural-Network,» 25 Juin 2020. [En ligne]. Available:
3 https://datascientest.com/convolutional-neural-network. [Accès le 01 Juin 2021].
3
]
[«Réseaux de neurones récurrents,» La Rédaction TechTarget, Juillet 2019. [En ligne]. Available:
3 https://www.lemagit.fr/definition/Reseaux-de-neurones-recurrents. [Accès le 01 Juin 2021].
4
]
[B. Dickson, «What are recurrent neural networks (RNN)?,» 08 Juin 2020. [En ligne]. Available:
3 https://bdtechtalks.com/2020/06/08/what-is-recurrent-neural-network-rnn/. [Accès le 02 Juin 2021].
5

]
[36] «Anaconda,» IA, 01 Octobre 2018. [En ligne]. Available: https://intelligence-artificielle.agency/anaconda/. [Accès le 03 Juin 2021].
]
[37] H. Michel, «Google Colab : Le Guide Ultime,» 04 Novembre 2019. [En ligne]. Available: https://ledatascientist.com/google-colab-le-guide-ultime/. [Accès le 04 Juin 2021].
]
[38] «Python,» 25 Février 2021. [En ligne]. Available: https://www.lebigdata.fr/python-langage-definition. [Accès le 04 Juin 2021].
]
[39] «Pandas,» [En ligne]. Available: https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/. [Accès le 05 Juin 2021].
]
[40] «Matplotlib,» 2017. [En ligne]. Available: https://he-arc.github.io/livre-python/matplotlib/index.html#:~:text=matplotlib%20est%20une%20biblioth%C3%A8que%20Python,quatre%20outils%20d'interface%20graphique.&text=Pour%20des%20graphiques%20simples%2C%20le%20module%20matplotlib.. [Accès le 5 Juin 2021].
]
[41] F. Fasmeyer, «matplotlib,» 2017. [En ligne]. Available: https://he-arc.github.io/livre-python/matplotlib/index.html. [Accès le 06 Juin 2021].
]