RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE Dr. TAHAR MOULAY SAIDA

FACULTE : TECHNOLOGIE

DEPARTEMENT : INFORMATIQUE

MÉMOIRE DE MASTER

Option: Modélisation Informatique des Connaissances et du Raisonnement (MICR)

Sélection de modèles pour l'apprentissage automatique

Présenté par :

ROUDANE Saliha

Encadré par :

Dr. BENAMARA Djilali

Année Universitaire 2020-2021

Remerciements

Tout d'abord je tiens à remercier ALLAH le tout puissant de m'avoir donné la santé, la volonté, le courage et la patience pour mener à terme ma formation et pouvoir réaliser ce travail de recherche.

Je tiens à remercier, très sincèrement, le professeur **Mr. BENAMARA Djillali** j'ai l'honneur et la chance de bénéficier de ses connaissances et compétence.

J'adresse également des remerciements à tous les enseignants

Département Informatique Univ-Saida

en générale.

Dédicace

Je dédie ce mémoire:
A mon Père et ma Mère **FELLAG Fatima** pour leur amour inestimable, leur sacrifices leur confiance, leur soutien et toutes les valeurs qu'ils ont su m'inculquer
Qui je souhaite une bonne santé
A mes sœurs : **Fatima**, **Faiza**, **Amina**, **Lwiza**, **Linda**
A mon frère : **Khaled**
Pour ses soutiens moral et leurs conseils précieux tout au long de mes études.
A tous la famille:
ROUDANE et **FELLAG**

Saliha

Table des matières

Chapitre III : Réalisation

Liste des Figures

Liste des Tableaux

Liste des abréviations:

MS: Model Selection

ML: Machine learning

ERM: Minimisation des risques empiriques

AIC: Critère d'information d'Akaike.

BIC: Critère d'information bayésien.

CV: Cross-Validation.

IA: d'intelligence artificielle.

DL: Deep Learning.

RN : Réseau Neurone.

CNN: Réseau De Neurones Convolutifs.

RNN: Réseau De Neurones Recurrents.

LSTM: Long Short-Term Memory

GRU: Unité récurrente fermée

ملخص

الهدف من اختيار النموذج هو العثور على بنية الشبكة مع أفضل خصائص التعميم، أي تلك التي تقلل الخطأ في الحالات المحددة لمجموعة البيانات (خطأ التحديد). هناك مشكلتان متكررتان في تصميم الشبكة العصبية تسمى النقص والتركيب. يتم تحقيق أفضل تعميم باستخدام نموذج يكون تعقيده هو الأنسب لإنتاج ملاءمة مناسبة للبيانات. يسمى خطأ الشبكة العصبية في حالات التدريب لمجموعة البيانات خطأ التدريب. وبالمثل، يسمى الخطأ الموجود في المثيلات المحددة خطأ التحديد. يقيس خطأ التدريب قدرة الشبكة العصبية على ملاءمة البيانات التي تراها. لكن خطأ الاختيار يقيس قدرة الشبكة العصبية على التعميم على البيانات الجديدة. الهدف من العمل الحالي هو تقديم أفضل نموذج للعمل العصبي العصبي لملاءمة البيانات مع خطأ التدريب الأدنى باستخدام التحقق من صحة الطيات k.

الكلمات المفتاحية: اختيار النموذج، التجهيز غير المناسب، التجهيز الزائد، التحقق المتقاطع، التعلم العميق، الشبكة العصبية، شبكة عصبية تلافيفية، شبكة عصبية متكررة.

Résumé

L'objectif de la sélection de modèle est de trouver l'architecture de réseau avec les meilleures propriétés de généralisation, c'est-à-dire celle qui minimise l'erreur sur les instances sélectionnées de l'ensemble de données (l'erreur de sélection). Deux problèmes fréquents dans la conception d'un réseau de neurones sont appelés sous-apprentissage et sur-apprentissage. La meilleure généralisation est obtenue en utilisant un modèle dont la complexité est la plus appropriée pour produire un ajustement adéquat des données. L'erreur d'un réseau de neurones sur les instances d'apprentissage de l'ensemble de données est appelée erreur d'apprentissage. De même, l'erreur sur les instances sélectionnées est appelée erreur de sélection. L'erreur d'apprentissage mesure la capacité du réseau de neurones à s'adapter aux données qu'il voit. Mais l'erreur de sélection mesure la capacité du réseau de neurones à se généraliser à de nouvelles données. L'objectif du présent travail est de donner le meilleur modèle de réseau neuronal pour ajuster les données avec l'erreur d'entraînement minimale en utilisant la validation croisée k-fold.

Mots clés: Sélection de modèle, sous-apprentissage, sur-apprentissage, validation croisée, Deep Learning, réseau de neurones, Réseau neuronal convolutif, réseau neuronal récurrent.

Abstract

The objective of model selection is to find the network architecture with the best generalization properties, that is, that which minimizes the error on the selected instances of the data set (the selection error). Two frequent problems in the design of a neural network are called underfitting and overfitting. The best generalization is achieved by using a model whose complexity is the most appropriate to produce an adequate fit of the data. The error of a neural network on the training instances of the data set is called the training error. Similarly, the error on the selected instances is called the selection error. The training error measures the ability of the neural network to fit the data that it sees. But the selection error measures the abilit of the neural network to generalize to new data. The aim of the present work is to give the best model of neural neutwork to fit data with the minimun training error by using the k-fold cross validation.

Key Words: Model selection, underfitting, overfitting, cross validation, Deep Learning, neural network, Convolutional neuronal network, recurrent neural network.

Introduction générale

La sélection du modèle est un problème important dans de nombreux domaines, y compris l'apprentissage automatique. Il s'agit de sélectionner un modèle statistique à partir d'un ensemble de modèles candidats, en fonction des données. Dans le cas le plus simple, un ensemble de données préexistant est considéré. Cependant, la tâche peut également inclure la conception d'expériences afin que les données collectées soient bien adaptées au problème de sélection de modèle. Compte tenu des modèles candidats ayant un pouvoir prédictif ou explicatif similaire, un modèle plus simple est susceptible d'être le meilleur choix. Le but de la sélection du modèle est de choisir le modèle qui se rapproche le mieux des données observées.

Plan du mémoire:

Ce mémoire est divisé en trois principaux chapitres:

Dans le premier chapitre « Sélection du modèle » nous allons présenter les concepts de sélection de modèle.

Dans le second chapitre « Deep Learning» nous allons présenter la technique utilisée dans ce mémoire pour résoudre ce problème qui est le deep leaning son principe, ses différentes architectures et ses différentes domaines d'application.

Le troisième chapitre « Réalisation » sera une description les outils utilisés et les résultats obtenus.

Et enfin, nous terminerons ce mémoire par une conclusion générale.

Chapitre I
Sélection du Modèle

I.1. Introduction:

Sélectionner le modèle d'apprentissage automatique parfait est à la fois un art et une science. En ce qui concerne les modèles d'apprentissage automatique, il n'existe pas de solution ou d'approche unique qui convienne à tous. Il y a plusieurs facteurs qui peuvent affecter votre choix d'un modèle d'apprentissage automatique.

MS est le processus de sélection du modèle ML final à partir de l'ensemble des modèles d'apprentissage automatique candidats pour l'ensemble de données d'apprentissage. MS est un processus qui peut être appliqué à différents types de modèles en même temps. MS est une partie importante de toute analyse statistique, et en fait, il est fondamental à la poursuite de la science. En général, il existe de nombreuses techniques et algorithmes qui permettent de sélectionner un modèle pour un ensemble particulier de données. Il a été développé et amélioré au fil des ans.

I.2.Minimisation des risques empiriques:

ERM est un principe de la théorie de l'apprentissage statistique qui définit une famille d'algorithmes d'apprentissage et est utilisé pour donner des limites théoriques à leurs performances. L'idée de base est que nous ne pouvons pas savoir exactement dans quelle mesure un algorithme fonctionnera dans la pratique (le vrai "risque") car nous ne connaissons pas la vraie distribution des données sur laquelle l'algorithme fonctionnera, mais nous pouvons à la place mesurer ses performances sur un ensemble connu de données d'entraînement (le risque « empirique »).[1]

La fonction de risque empirique, alias erreur dans l'échantillon, ou erreur de train, est la perte moyenne de l'échantillon d'un prédicteur h:

$$R_{emp}(h) = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i)$$

Le risque empirique permet d'obtenir une approximation imparfaite du risque réel. Le risque empirique peut être diminué en effectuant un sur apprentissage qui augmente lui le risque réel. Cette situation n'étant pas souhaitable, il convient de choisir un biais suffisamment fort pour empêcher le sur apprentissage. La figure ci-dessous montre l'évolution simultanée des deux risques.

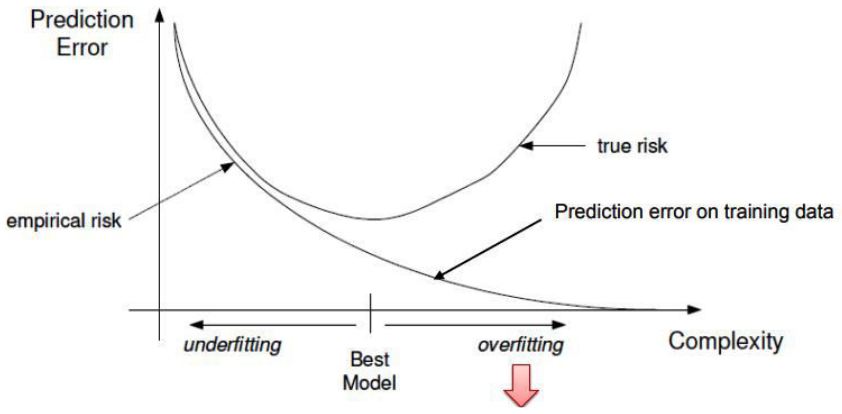


Figure I.1: L'évolution simultanée des deux risques.

I.3. Overfitting et Underfitting:

Un problème courant dans l'apprentissage automatique est l'overfitting, qui est défini par l'apprentissage d'une fonction qui explique parfaitement les données d'entraînement à partir desquelles le modèle a appris, mais ne se généralise pas bien aux données de test invisibles. L'overfitting se produit lorsqu'un modèle apprend à partir des données d'entraînement au point de commencer à détecter des idiosyncrasies qui ne sont pas représentatives des modèles du monde réel. [6]

Les concepts d'overfitting et d'underfitting sont étroitement liés au compromis biais-variance. Le biais fait référence à l'erreur due à des hypothèses trop simplistes ou à des hypothèses erronées dans l'algorithme d'apprentissage. Le biais entraîne un sous-ajustement des données. Un biais élevé signifie que notre algorithme d'apprentissage manque des tendances importantes parmi les fonctionnalités. [6]

La variance fait référence à l'erreur due à un modèle trop complexe qui essaie d'ajuster les données d'apprentissage aussi étroitement que possible. Dans les cas de variance élevée, les valeurs prédites du modèle sont extrêmement proches des valeurs réelles de l'ensemble d'apprentissage. Une variance élevée donne lieu à un overfitting. En fin de compte, pour avoir un bon modèle, nous avons besoin d'un faible biais et d'une faible variance. [6]

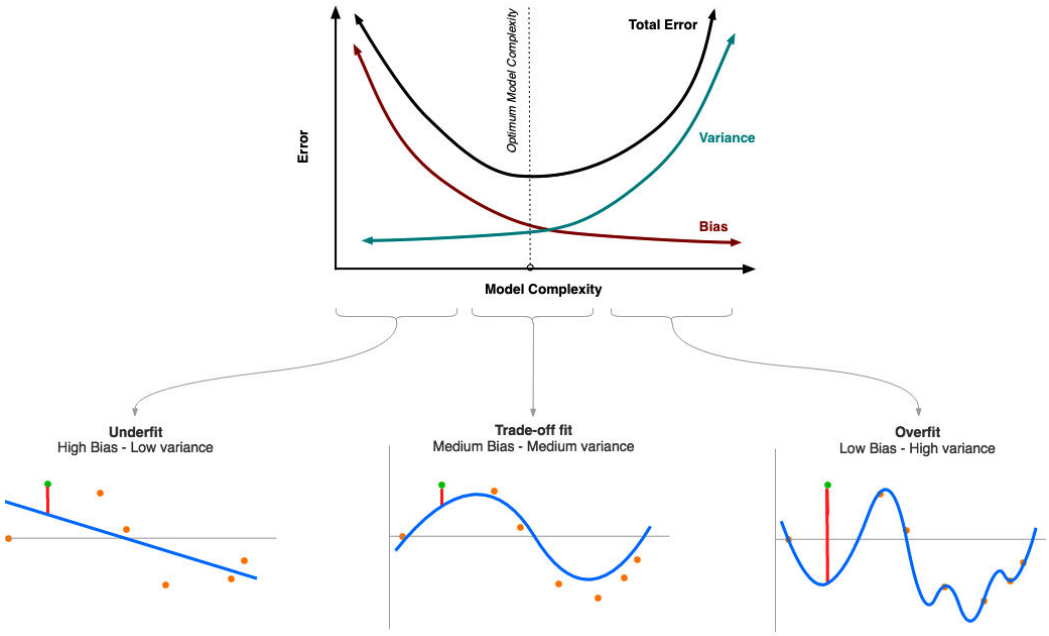


Figure I.2: Un exemple d'overfitting, d'underfitting et d'optimisation

✍ **High Bias:** également connu sous le nom d'underfitting, cela se produit lorsque le modèle n'apprend pas à partir de l'ensemble d'apprentissage, ce qui se traduit par des performances médiocres du modèle pour les trois ensembles (training, validation, et testing sets), ainsi que pour les données invisibles. [6]

✍ **High Variance:** également connue sous le nom d'overfitting, cette condition fait référence à l'incapacité du modèle à bien fonctionner sur des données différentes de celles de l'ensemble d'apprentissage. [6]

I.4. Biais-variance tradeoff:

Le biais et la variance sont des erreurs de prédiction en ML. En ML, on essaie de minimiser ces erreurs pour améliorer la précision tout en évitant overfitting et underfitting. Le biais est la différence entre la prédiction moyenne du modèle et la valeur réelle qu'il essaie de prédire. La variance est la variabilité de la prédiction du modèle, nous indiquant à quel point les modèles sont sensibles aux fluctuations trop faibles des ensembles de données. Le biais et la variance sont considérés comme un compromis, un modèle plus complexe et plus puissant réduit le biais et augmente la variance, et vice versa. En ML, nous essayons de trouver l'équilibre entre biais et variance qui minimise l'erreur de généralisation. **[2]**

✍ **Mathématiquement:**

L'erreur totale peut être écrite sous la forme d'une équation mathématique; **[5]**

$$\boldsymbol{Err(x) = Bias^2 + Variance + Irre\ ducible\ Error}$$

Ou :

$$\boldsymbol{E\left[y - \hat{f}(x))^2\right] = bias[\hat{f}(x)]^2 + var\left(\hat{f}(x)\right) + \sigma_\epsilon^2}$$

Cette équation suggère que nous devons trouver un modèle qui atteint simultanément un faible biais et une faible variance. La variance est un terme non négatif et le biais au carré est également non négatif, ce qui implique que l'erreur totale ne peut jamais descendre en dessous de l'erreur irréductible.

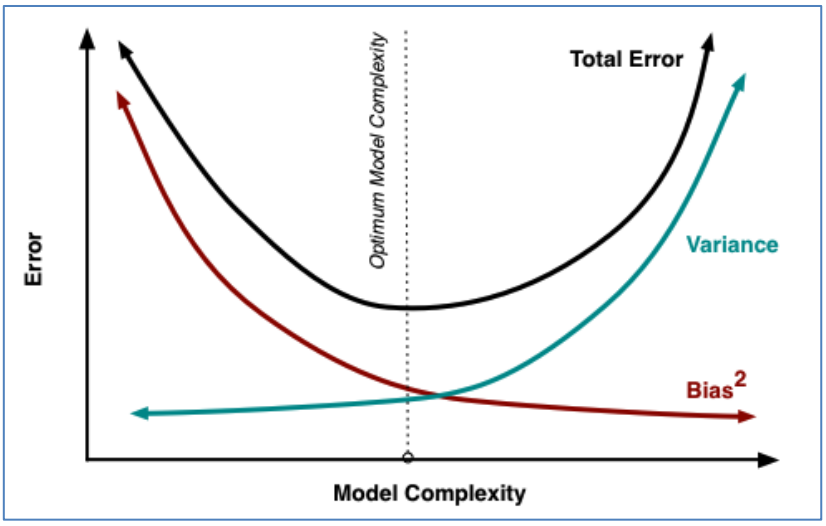


Figure I.3: Variance Bias Trade-off

Comme le montre la figure ci-dessus, il existe un point où l'erreur de validation croisée commence à augmenter en raison de l'augmentation de la variance et de la diminution du biais. C'est le point exact où le modèle doit cesser d'augmenter sa complexité et utiliser tous les paramètres définis par ce point de la courbe. Habituellement, c'est là que les courbes de Biais et de Variance se croisent, créant le point optimal de complexité du modèle. À ce stade, le modèle a un faible biais et une faible variance sans conduire à un underfitting ou à un overfitting du modèle.

✍ **Dérivation de l'équation:**

Définir les différentes notations utilisées : ont des variables indépendantes x qui marque la valeur d'une variable dépendante y. Fonction f désigne la véritable relation entre x et y. Dans les problèmes de la vie réelle, il est très difficile de connaître cette relation. y est donné par cette formule avec du bruit qui est représenté par la variable aléatoire ϵ avec une moyenne nulle et une variance ² :

$$\boldsymbol{y = f(x) + \epsilon}$$

Mathématiquement, a les propriétés suivantes:

$$\boldsymbol{E[\epsilon] = 0, var(\epsilon) = E[\epsilon^2] = \sigma_\epsilon^2}$$

Maintenant, lorsque nous essayons de modéliser le problème réel sous-jacent, nous essayons de trouver une fonction $\hat{f}$ qui peut prédire avec précision la vraie relation f. Le but est de rapprocher la prédiction le plus possible de la valeur réelle ($y \approx \hat{f}(x)$) pour minimiser l'erreur.

Venons-en maintenant à l'équation de compromis biais-variance,

$$\boldsymbol{E\left[y - \hat{f}(x))^2\right] = bias[\hat{f}(x)]^2 + var\left(\hat{f}(x)\right) + \sigma_\epsilon^2}$$

Ici, $\boldsymbol{E\left[y - \hat{f}(x))^2\right]}$ est l'erreur quadratique moyenne, communément appelée **MSE**. Ceci est défini comme la différence quadratique moyenne d'une prédiction $\boldsymbol{\hat{f}(x)}$ *à* partir de sa vraie valeur y.

Le biais est défini comme la différence entre la valeur moyenne de la prédiction et la vraie fonction de relation $f(x)$.

$$\boldsymbol{bias[\hat{f}(x)] = E\left[\hat{f}(x)\right] - f(x)}$$

La variance est définie comme l'espérance de l'écart au carré de $\boldsymbol{\hat{f}(x)}$ par rapport à sa valeur attendue E[f̂(x)].

$$\boldsymbol{var(\hat{f}(x)) = E\left[(\hat{f}(x) - E[\hat{f}(x)])^2\right]}$$

En partant de la gauche de l'équation, $\boldsymbol{E\left[y - \hat{f}(x))^2\right]}$,

$$\begin{aligned}
\boldsymbol{E\left[y - \hat{f}(x))^2\right]} &= \boldsymbol{E[(f(x) + \epsilon - \hat{f}(x))^2]} \\
&= \boldsymbol{E[f((x) - \hat{f}(x))^2] + E[\epsilon^2] + 2E[\left(f(x) - \hat{f}(x)\right)\epsilon]} \\
&= \boldsymbol{E[f((x) - \hat{f}(x))^2] + \underbrace{E[\epsilon^2]}_{=\sigma_\epsilon^2} + 2E\left[\left(f(x) - \hat{f}(x)\right)\right]\underbrace{E[\epsilon]}_{=0}} \\
&= \boldsymbol{E[f((x)) - \hat{f}(x))^2] + \sigma_\epsilon^2}
\end{aligned}$$

En remplaçant y par $f(x) + \epsilon$ dans la première ligne, nous procédons en développant davantage en utilisant la propriété linéaire d'espérance et d'indépendance des variables aléatoires ϵ et $\hat{f}$. Puis en utilisant les propriétés de ϵ et le fait que lorsque deux variables aléatoires sont indépendantes, l'espérance de leur produit est égale au produit de leurs espérances.

Maintenant, en élargissant encore le terme : $\boldsymbol{E\left[\left(f(x)-\hat{f}(x)\right)^2\right]}$

$$\begin{aligned}
E\left[\left(f(x)-\hat{f}(x)\right)^2\right] &= E\left[\left(f(x)-E[\hat{f}(x)]\right)-\left(\hat{f}(x)-E[\hat{f}(x)]\right)^2\right] \\
&= E\left[\left(E[\hat{f}(x)]-f(x)\right)^2\right]+E\left[\left(\hat{f}(x)-E[\hat{f}(x)]\right)^2\right] \\
&-2E\left[\left(f(x)-E[\hat{f}(x)]\right)\left(\left(\hat{f}(x)-E[\hat{f}(x)]\right)\right)\right] \\
&= \underbrace{\left(E[\hat{f}(x)]-f(x)\right)^2}_{=bias[\hat{f}(x)]}+\underbrace{E\left[\left(\hat{f}(x)-E[\hat{f}(x)]\right)^2\right]}_{=var(\hat{f}(x))} \\
&-2\left(f(x)-E[\hat{f}(x)]\right)E\left[\left(\hat{f}(x)-E[\hat{f}(x)]\right)\right] \\
&= bias[\hat{f}(x)]^2+var\left(\hat{f}(x)\right) \\
&-2\left(f(x)-E[\hat{f}(x)]\right)\left(E[\hat{f}(x)]-E[\hat{f}(x)]\right) \\
&= bias[\hat{f}(x)]^2+var(\hat{f}(x))
\end{aligned}$$

$\boldsymbol{E\left[\hat{f}(x)-f(x)\right]}$ est une constante puisque l'on soustrait $f(x)$, une constante, de $\boldsymbol{E\left[\hat{f}(x)\right]}$ qui est aussi une constante. Don, $\boldsymbol{E\left[E(\hat{f}(x)\right]-f(x)^2]=\left(E\left[\hat{f}(x)\right]-f(x)\right)^2}$**.** En développant davantage en utilisant la propriété de linéarité de l'espérance, nous obtenons la valeur de $\boldsymbol{E\left([f(x)]-\hat{f}(x)\right)^2}$. En rebranchant cette valeur dans l'équation pour $\boldsymbol{E\left[y-\hat{f}(x))^2\right]}$, nous arrivons à notre équation finale.

$$\boldsymbol{E\left[y-\hat{f}(x))^2\right]=bias[\hat{f}(x)]^2+var\left(\hat{f}(x)\right)+\sigma_\epsilon^2}$$

I.5. Sélection du modèle par comparaison:

La sélection de modèle fait référence au problème de l'utilisation des données pour sélectionner un modèle dans la liste des modèles concurrents. Essentiellement, cela implique l'utilisation d'un critère de sélection de modèle pour trouver le modèle le mieux adapté aux données. La sélection de modèles à l'aide de critères d'information a été développée pour résumer les données probantes en faveur d'un modèle. Plus précisément, les techniques de critères d'information mettent l'accent sur la minimisation de la quantité d'informations requises pour exprimer les données et le modèle. Il en résulte une sélection de modèles qui sont une représentation efficace des données. **[3]**

I.5.1. Critère d'information d'Akaike (AIC):

L'AIC est l'un des critères d'information les plus couramment utilisés. L'idée de l'AIC est de sélectionner le modèle qui minimise la vraisemblance négative pénalisée par le nombre de paramètres comme spécifié dans l'équation: **[3]**

$$AIC = -2 \log p(L) + 2p$$

Où:

- L: Fait référence à la vraisemblance sous le modèle ajusté.
- p: Est le nombre de paramètres dans le modèle.

Plus précisément, l'AIC vise à trouver le meilleur modèle d'approximation du processus de génération de données réel inconnu et de ses applications (Akaike, 1973; Bozdogan, 1987; Zucchini, 2000). **[3]**

I.5.2. Critère d'information bayésien (BIC):

Un autre critère d'information largement utilisé est le BIC. Contrairement aux critères d'information d'Akaike, le BIC est dérivé dans un cadre bayésien en tant qu'estimation du facteur de Bayes pour deux modèles concurrents (Schwarz, 1978; Kass et Raffery, 1995). Le BIC est défini comme: **[3]**

$$BIC = -2 \log p(L) + 2p \log(n)$$

Où:

- L: Fait référence à la vraisemblance sous le modèle ajusté.
- p: Est le nombre de paramètres dans le modèle.

Superficiellement, BIC ne diffère d'AIC que par le second terme qui dépend désormais de la taille de l'échantillon n. Les modèles qui minimisent les critères d'information bayésiens sont sélectionnés. D'un point de vue bayésien, BIC est conçu pour trouver le modèle le plus probable compte tenu des données.

I.6. Validation et test du modèle:

I.6.1. Le partitionnement des données:

Est un processus consistant à diviser un ensemble de données en trois sous-ensembles afin que chaque ensemble peut être utilisé à des fins différentes. Ainsi, le développement d'un modèle n'est pas affecté par l'introduction de biais. Ce qui suit est une explication de chaque sous-ensemble: **[4]**

✍ **Ensemble d'entraînement (Training set):**

Comme son nom l'indique, il s'agit de la partie de l'ensemble de données utilisée pour l'entraînement du modèle. Il se compose des données d'entrée (les observations) associées à un résultat (la classe d'étiquettes). Cet ensemble peut être utilisé pour entraîner autant de modèles que souhaité, en utilisant différents algorithmes. Cependant, l'évaluation des performances n'est pas effectuée sur cet ensemble car, puisque cet ensemble a été utilisé pour entraîner le modèle, la mesure serait biaisée.

✍ **Ensemble de validation (Validation set):**

Également appelé ensemble de développement, cet ensemble est utilisé pour effectuer un évaluation impartiale de chaque modèle tout en affinant les hyperparamètres. L'évaluation des performances est fréquemment effectuée sur cet ensemble de données pour tester différentes configurations des hyperparamètres.

Bien que le modèle n'apprenne pas de ces données (il apprend des données de l'ensemble d'apprentissage), il est indirectement affecté par les données de cet ensemble en raison de sa participation au processus de décision des modifications apportées aux hyperparamètres. Après avoir exécuté différentes configurations d'hyperparamètres basées sur les performances du modèle sur l'ensemble de validation, un modèle gagnant est sélectionné pour chaque algorithme. **[4]**

✍ **Ensemble de test (Testing set):** il est utilisé pour effectuer l'évaluation finale des performances du modèle (après apprentissage et validation) sur des données invisibles. Cela permet de mesurer les performances du modèle avec des données réelles pour les prévisions futures. L'ensemble de test est également utilisé pour comparer des modèles concurrents. Considérant que l'ensemble d'apprentissage a été utilisé pour entraîner différents modèles et l'ensemble de validation a été utilisé pour affiner les hyperparamètres de chaque modèle afin

de sélectionner une configuration gagnante, le but de l'ensemble de test est d'effectuer une comparaison impartiale des modèles finaux. **[4]**

Le schéma suivant montre le processus de sélection du modèle idéal et d'utilisation des ensembles mentionnés précédemment: **[4]**

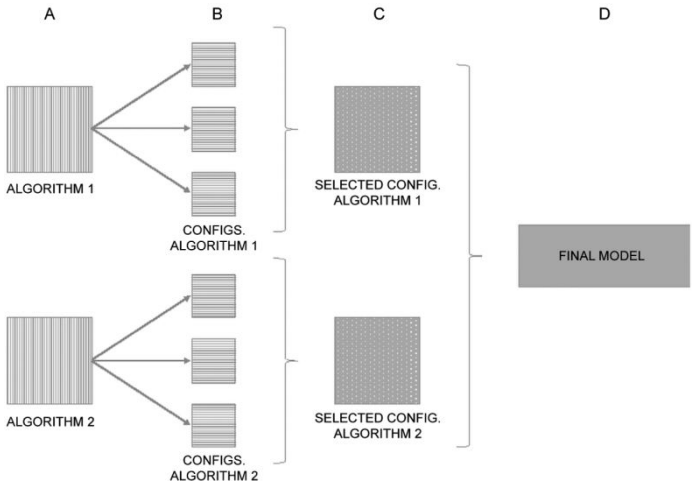


Figure I.4: Objectifs de la partition de l'ensemble de données

Les sections « **A** à **D** » illustrées dans le schéma précédent sont décrites comme suit: **[4]**

✍ La section « **A** » fait référence au processus d'apprentissage du modèle pour les algorithmes souhaités en utilisant les données contenues dans l'ensemble d'apprentissage.
✍ La section « **B** » représente le processus de réglage fin des hyperparamètres de chaque modèle. La sélection de la meilleure configuration d'hyperparamètres est basée sur la performance du modèle sur l'ensemble de validation.
✍ La section « **C** » montre le processus de sélection du modèle final en comparant la configuration finale de chaque algorithme en fonction de ses performances sur l'ensemble de test.
✍ Enfin, la section « **D** » représente le modèle sélectionné qui sera appliqué aux données réelles pour la prédiction.

Le diagramme suivant montre la partition proportionnelle de l'ensemble de données en trois sous-ensembles. Il est important de souligner que l'ensemble d'entraînement doit être plus grand que les deux autres, car c'est celui qui doit être utilisé pour entraîner le modèle. De plus, il est possible d'observer que les ensembles d'apprentissage et de validation ont un effet sur le modèle, tandis que l'ensemble de test est principalement utilisé pour valider les performances réelles du modèle avec des données invisibles. Compte tenu de cela, les ensembles d'apprentissage et de validation doivent provenir de la même distribution: **[4]**

Figure I.5: Visualisation du pourcentage de division

Initialement, les problèmes d'apprentissage automatique étaient résolus en partitionnant uniquement les données en deux ensembles : un ensemble d'apprentissage et un ensemble de test. Cette approche consistait à utiliser l'ensemble d'apprentissage pour entraîner le modèle, ce qui est identique à l'approche à trois ensembles. Cependant, l'ensemble de test a été utilisé pour affiner l'hyperparamètre ainsi que pour déterminer les performances ultimes de l'algorithme. **[4]**

Compte tenu de cela, il existe un moyen d'obtenir un modèle moins biaisé tout en divisant l'ensemble de données en deux ensembles, ce qu'on appelle une division de validation croisée (cross-validation). **[4]**

I.6.2. Cross-Validation:

Est également une procédure utilisée pour partitionner les données en rééchantillonnant les données utilisées pour entraîner et valider le modèle. Il se compose d'un paramètre, K, qui représente le nombre de groupes dans lesquels l'ensemble de données sera divisé.

La procédure est également appelée validation croisée K-fold, où K est généralement remplacé par un nombre de votre choix. Par exemple, un modèle créé à l'aide d'une procédure de validation croisée 10 fois signifie un modèle où les données sont divisées en 10 sous-groupes. La procédure de CV est illustrée dans le schéma suivant:

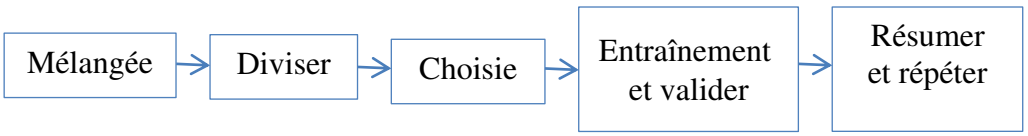


Figure I.6: Procédure de validation croisée

Le schéma précédent affiche la procédure générale suivie pendant le CV:

1. Les données sont mélangées de manière aléatoire, en considérant que le processus de validation croisée est répété.
2. Les données sont divisées en K sous-groupes.
3. L'ensemble de validation/test est sélectionné comme l'un des sous-groupes qui ont été créés. Le reste des sous-groupes devient l'ensemble d'apprentissage.
4. Le modèle est entraîné sur l'ensemble d'entraînement, comme d'habitude. Le modèle est évalué à l'aide de l'ensemble de validation/test.
5. Le résultat de cette itération est enregistré. Les paramètres sont ajustés en fonction des résultats et le processus recommence en remaniant les données. Ce processus est répété K nombre de fois.

I.7. Conclusions:

Dans ce chapitre nous avons vu la notion de risques et surtout la notion de fonction de perte, ensuite nous avons présenté Overfitting et Underfitting, Biais-variance trade-off (Mathématiquement, Dérivation de l'équation), ensuite nous avons présenté Sélection du modèle en utilisant des critères d'information (AIC, BIC), Finalement, nous avons présenté Validation et test du modèle (Le partitionnement des données, Cross-Validation).

Chapitre II
Deep Learning

II.1. Introduction:

Avant d'arriver à ce qu'est l'apprentissage profond (en anglais (DL): Deep Learning), nous devons introduire deux concepts principaux : le premier est le concept d'intelligence artificielle (IA), et le second est le concept l'apprentissage automatique (en anglais ML: Machine Learning).

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

ML est un sous-domaine d'IA, qui donne à un système une capacité de compréhension grâce à ses algorithmes. Il est basé sur l'idée de faire apprendre des algorithmes à partir de données et de faire des prédictions avec ces données et par cela les ordinateurs apprennent à résoudre des tâches spécifiques, sans avoir besoin de les programmer.

DL est un nouveau domaine de recherche du Machine Learning, qui a été introduit dans le but de rapprocher le ML de son objectif principal: l'intelligence artificielle. Il concerne les algorithmes inspirés par la structure et le fonctionnement du cerveau. Ils peuvent apprendre plusieurs niveaux de représentation dans le but de modéliser des relations complexes entre les données.

II.2. Histoire du Deep Learning:

Les débuts de l'IA remontent à Alan Turing dans les années 1950, c'est une branche de l'Informatique fondamentale s'est développée avec pour objectif la simulation des comportements du cerveau humain. Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'ère informatique. En effet, dans l'imaginaire commun, lorsqu'on parle d'intelligence artificielle on désigne par là un programme qui peut effectuer des taches d'humain, en apprenant toute seule. Or, l'IA telle que définie dans l'industrie est plutôt « des algorithmes plus ou moins évolués qui imitent des actions humaines ». L'IA a beaucoup évolué grâce notamment à l'émergence du Cloud Computing et du Big Data, soit d'une puissance de calcul peu couteuse et de l'accessibilité à un grand nombre de données. Ainsi, les machines ne sont plus programmées; elles apprennent.

En 1980 apparait le terme d'apprentissage automatique (machine learning en anglais ML), est un sous-domaine de l'IA qui s'intéresse en particulier aux capacités d'apprentissage. Le principe est de reproduire un comportement non pas en le programmant à la main dans un ordinateur, mais en concevant un système plus général capable d'apprendre à partir d'exemples à résoudre votre problème.

En 2010 un nouveau terme fait son apparition et c'est l'apprentissage profond"(en anglais deep Learning DL) et c'est l'abréviation du terme\apprentissage dans les réseaux de neurones profonds". et comme son nom l'indique il se base sur les réseaux de neurones artificielles. [2]

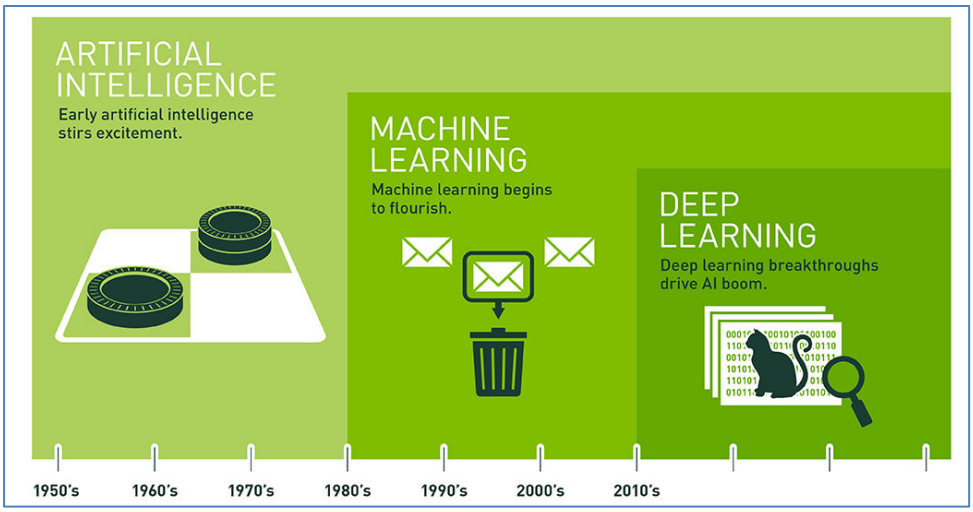


Figure II.1: Echelle de temps pour l'intelligence artificielle, apprentissage automatique et l'apprentissage profond

II.3. Définition:

DL est basé sur l'idée des réseaux de neurones artificiels et il est taillé pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de DL a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime.

Le terme "DL" a été introduit pour la première fois au ML par Dechter (1986), et aux réseaux neuronaux artificiels par Aizenberg et al (2000).

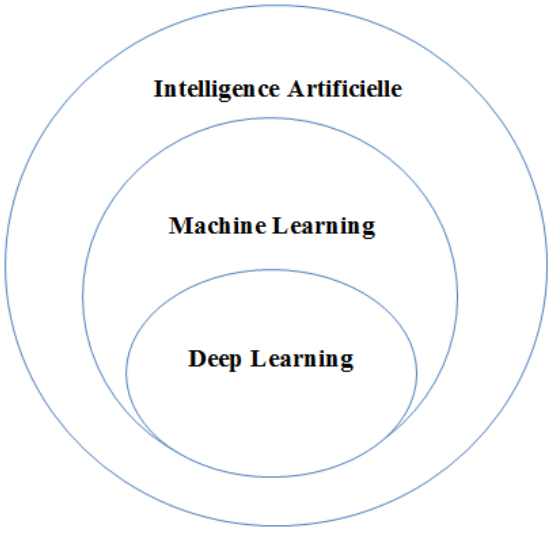


Figure II.2: La relation entre AI, ML et DL

II.4. Domaines d'application de Deep Learning:

INTERNET & NUAGE:

- Classification des images.
- Reconnaissance vocale.
- Traduction du langage.
- Traitement du langage.
- Analyse des sentiments.
- Recommandation.

MACHINES AUTONOMES:

- Détection de piétons.
- Suivi de voie.
- Reconnaître les panneaux de signalisation.

SÉCURITÉ & DÉFENSE:

- Détection de visage.
- Vidéosurveillance.
- Imagerie satellite.

MÉDECINE & BIOLOGIE:

- Détection des cellules cancéreuses.
- Classement des diabétiques.
- Découverte de médicaments.

MÉDIAS ET DIVERTISSEMENT:

- Sous-titrage vidéo.
- Traduction en temps réel.
- Recherche vidéo. **[3]**

II.5. Le réseau neuronal (RN):

II.5.1. Définition:

Les RN sont un sous-ensemble de ML et sont au cœur des algorithmes du DL. Le cerveau humain inspire leur nom et leur structure. Les réseaux de neurones profonds [Figure II.2] compromis de couches de nœuds, contenant une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque nœud (également appelé perceptron) est connecté à un autre et possède un poids et un seuil associés. Si la sortie d'un perceptron individuel est supérieure à la valeur seuil spécifiée, ce perceptron est activé, envoyant des données à la couche suivante du réseau. Si non, aucune donnée ne transite par la couche suivante du réseau. **[4]**

RN s'appuient sur les données d'entraînement pour apprendre et améliorer leur précision au fil du temps. Cependant, une fois que ces algorithmes d'apprentissage atteignent une grande précision, ce sont des outils puissants en informatique et en IA, nous permettant de classer et de regrouper les données à grande vitesse. **[4]**

II.5.2. Les neurones biologiques:

Aussi appelé cellule nerveuse, le neurone est une cellule appartenant au système nerveux et dont le plus grand nombre est présent dans le cortex cérébral. Son rôle est de recevoir, analyser puis transmettre des informations sous la forme de signaux bioélectriques appelés influx nerveux et ce sans distinction du rôle du message envoyé qui peut être perceptif, moteur, émotif ou cognitif. **[7]**

➢ Composition:

✍ **Corps cellulaire (soma) :** il est centré par un noyau, toutes les informations recueillies par les synapses sont acheminées vers le corps cellulaire.

✍ **Synapse :** qui est une jonction entre deux neurones; et généralement entre l'axone d'un neurone et une dendrite d'un autre neurone.

✍ **Dendrites :** ce sont de fines extensions tubulaires qui se ramifient autour du neurone et forment une sorte de vaste arborescence. Elles captent les signaux envoyés au neurone.

✍ **L'axone :** qui est la partie qui s'occupe de la transmission de l'information issue du corps cellulaire; conduisant des signaux électriques de la sortie d'un neurone vers l'entrée d'un autre neurone.

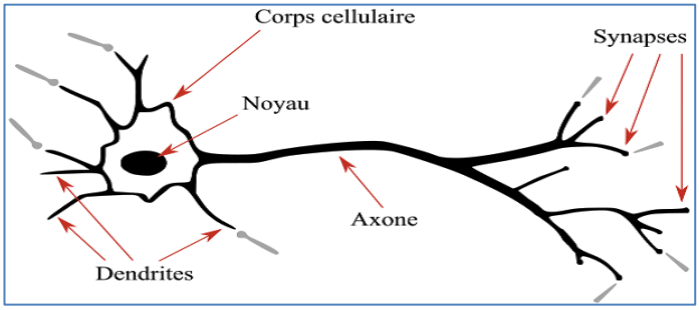


Figure II.3: Représentation schématique d'un neurone biologique

II.5.3. Les neurones formels (artificiel):

Un "neurone formel" (ou simplement "neurone") est une fonction algébrique non linéaire et bornée, dont la valeur dépend des paramètres appelés coefficients ou poids. Les variables de cette fonction sont habituellement appelées "entrées" du neurone, et la valeur de la fonction est appelée sa "sortie". **[1]**

Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de logiciel. On a pris l'habitude de représenter graphiquement un neurone comme indiqué sur la figure II.4. **[1]**

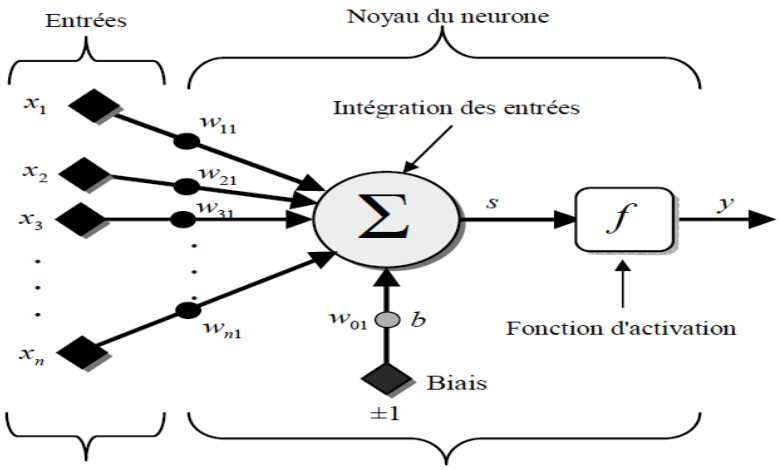


Figure II.4: Modèle d'un neurone artificiel

Le modèle du neurone formel est proposé par W:M:Culloch et W:P itts en 1943 on en trouve: **[1]**

- Les $\boldsymbol{x_i}$ C'est les vecteurs d'entrées.
- Les $\boldsymbol{w_{ij}}$ sont les poids synaptiques du neurone $\boldsymbol{j}$.
- Ces **poids** pondèrent les entrées et peuvent être modifiés par apprentissage.
- **Biais:** entrée prend souvent les valeurs -1 ou +1 qui permet d'ajouter de la flexibilité au réseau. en permettant de varier le seuil de déclenchement du neurone par l'ajustement des poids et du biais lors de l'apprentissage.
- **Noyau:** intègre toutes les entrées et le biais et calcul la sortie du neurone selon une fonction d'activation qui est souvent non linéaire pour donner une plus grande flexibilité d'apprentissage.

Un neurone est essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées. Le résultat s de cette somme est ensuite transformé par une fonction de transfert f qui produit la sortie y du neurone. Les n entrées du neurone correspondent au vecteur $x = [x_1, x_2, \dots, x_n]^T$, alors que $w = [w_{11}, w_{21}, \dots, w_{n1}]^T$ représente le vecteur des poids du neurone. La sortie s de l'intégrateur est donnée par l'équation suivante:**[1]**

$$\boldsymbol{s = \sum_{i=1}^{n} w_{i1} x_1 \mp b}$$

Cette sortie correspond à une somme pondérée des poids et des entrées plus ce qu'on nomme le biais b du neurone. Le résultat s de la somme pondérée s'appelle le niveau d'activation du neurone. Le biais b s'appelle aussi le seuil d'activation du neurone. Lorsque le

niveau d'activation atteint ou dépasse le seuil b, alors l'argument de f devient positif (ou nul). Sinon, il est négatif. **[1]**

Les réseaux de neurones sont caractérisés par l'architecture (l'organisation des neurones), l'apprentissage (méthode de détermination des poids de connexions), et par leur fonction d'activation.

II.5.3.1. Fonctions d'activation:

Différentes fonctions d'activation pouvant exister pour activer le neurone dont elles sont énumérées au tableau 2. Les fonctions d'activations les plus utilisées sont «seuil» (en anglais «hard limit»), «linéaire» et «sigmoïde». **[1]**

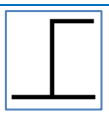
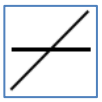
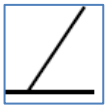
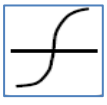
Nom de la fonction	Relation entrée/sortie	Icône
Seuil	$y = 0 \quad \boldsymbol{Si} \quad s < 0$ $y = 1 \quad \boldsymbol{Si} \quad s \geq 0$	
Seuil symétrique	$y = -1 \quad \boldsymbol{Si} \quad s < 0$ $y = 1 \quad \boldsymbol{Si} \quad s \geq 0$	
Linéaire	$y = s$	
Linéaire saturée	$y = 0 \quad \boldsymbol{Si} \quad s \leq 0$ $y = s \quad \boldsymbol{Si} \quad 0 \leq s \leq 1$ $y = 1 \quad \boldsymbol{Si} \quad s \geq 0$	
Linéaire saturée symétrique	$y = -1 \quad \boldsymbol{Si} \quad s < -1$ $y = s \quad \boldsymbol{Si} \quad -1 \leq s \leq -1$ $y = 1 \quad \boldsymbol{Si} \quad s > 1$	
Linéaire positive (ReLU)	$y = 0 \quad \boldsymbol{Si} \quad s \leq 0$ $y = s \quad \boldsymbol{Si} \quad s \geq 0$	
Sigmoïde	$y = \frac{1}{1 + exp^{-s}}$	
Tangente hyperbolique (Tanh)	$y = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}}$	

Tableau II.1: Différentes fonctions d'activations

Pour mettre en œuvre un système de réseau de neurones artificiel il suffit de correspondre chaque élément du neurone biologique au neurone formel le tableau II.3 suivant pourra résumer cette modélisation: **[1]**

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dentrites	Signal d'entrée
Noyau ou Somma	Fonction d'activation

Tableau II.2: Analogie entre le neurone biologique et le neurone formel

II.5.3.2. Architecture des réseaux de neurones:

On distingue deux grands types d'architectures de réseaux de neurones: les réseaux non bouclés (statiques ou Feed-forward) et les réseaux bouclés (dynamiques ou Feed-back).

✍ Réseaux de neurones non bouclés:

Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux. L'information circulant des entrées vers les sorties sans retour en arrière. C'est à dire si l'on se déplace dans le réseau à partir d'un neurone quelconque en suivant les connexions, on ne peut pas revenir au neurone de départ. Les réseaux de neurones non bouclés sont des outils statiques, utilisés principalement pour effectuer des tâches d'approximation de fonctions non linéaires, de modélisation de processus statiques non linéaires. **[6]**

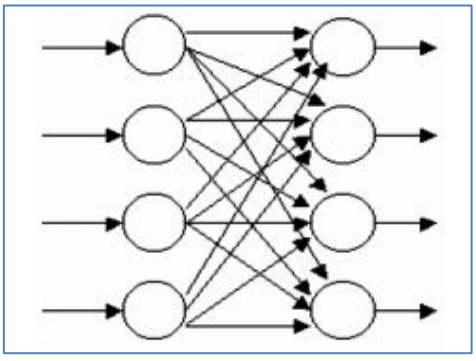


Figure II.5: Réseau non bouclé à connexions totales

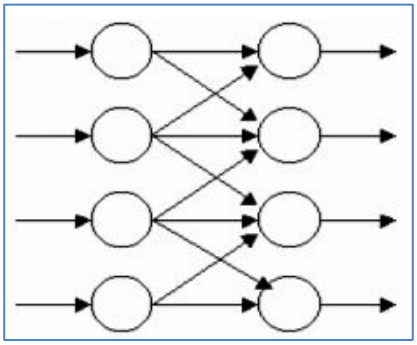


Figure II.6: Réseau non bouclé à connexions partielles

Quelques modèles des réseaux de neurone non bouclés:

1. Perceptron mono couche:

C'est historiquement le premier RNA, c'est un réseau simple, puisque il ne se compose que d'une couche d'entrée et d'une couche de sortie. Il est calqué, à la base, sur le système visuel et de ce fait a été conçu dans un but premier de reconnaissance des formes.

2. Le perceptron multi couche (MLC):

Le perceptron Multi Couches PMC ou MLP (Multi Layer Perceptron) en anglais est une extension du perceptron monocouche, avec une ou plusieurs couches cachées entre l'entrée et la sortie. L'idée principale est de grouper des neurones dans une couche. En place ensuite bout à bout plusieurs couches et on connecte complètement les neurones de deux couches adjacentes. Les entrées des neurones de la deuxième couche sont donc en fait les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée. Ils calculent alors leurs sorties qui sont transmises aux neurones de la deuxième couche, etc. Les sorties des neurones de la dernière couche forment la sortie du réseau.

✍ Les réseaux de neurones bouclés:

Ce sont des réseaux qui ont un ou plusieurs rebouclages internes, leurs sorties à un instant dépendront des entrées aux mêmes instants, et aux instants antérieurs. Ces connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation. Les rebouclages rajoutent donc un effet de mémorisation du passé. Ces réseaux de neurones bouclés constituent un système dynamique "à temps discret", régi par une (ou plusieurs) équation(s) aux différences non linéaires, résultant de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions. Ils sont utilisés pour effectuer des tâches de modélisation et d'adaptation de systèmes dynamiques, de commande de processus, ou de filtrage. **[6]**

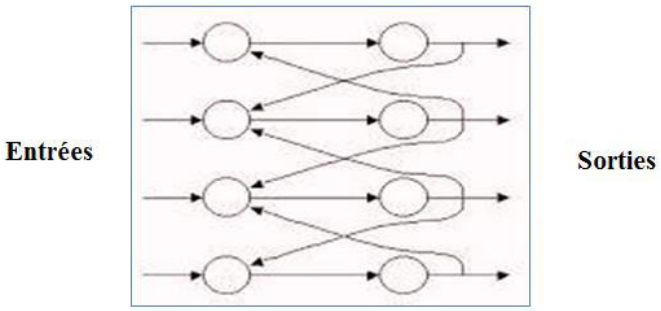


Figure II.7: Structure d'un réseau de neurones dont les connexions sont récurrentes (bouclées)

II.5.3.3. Architecture des réseaux de neurones profonds:

Il existe un grand nombre de variantes d'architectures profondes. La plupart d'entre elles sont dérivées de certaines architectures parentales originales. Il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données. DL est un domaine à croissance rapide, et de nouvelles architectures, variantes ou algorithmes apparaissent toutes les semaines.

II.5.3.3.1. Les réseaux neuronaux convolutionnels (CNN):

Les réseaux de neurones convolutionnels (CNN) sont un type de réseau de neurones artificiel. En particulier pour les données de grande dimension (par exemple, les images et les vidéos). Les premiers CNN ayant eu du succès ont été inventés en 1989 par LeCun dans et appliqués à la reconnaissance d'écriture manuscrite. Le nom « convolution » est dérivé d'une opération mathématique impliquant la convolution de différentes fonctions.

Les différentes couches de CNN:

Un CNN est composé de plusieurs blocs de construction de base, appelés couches CNN. Ces couches implémentent des fonctionnalités de base telles que: pooling, convolution, fully connected layers. **[8]**

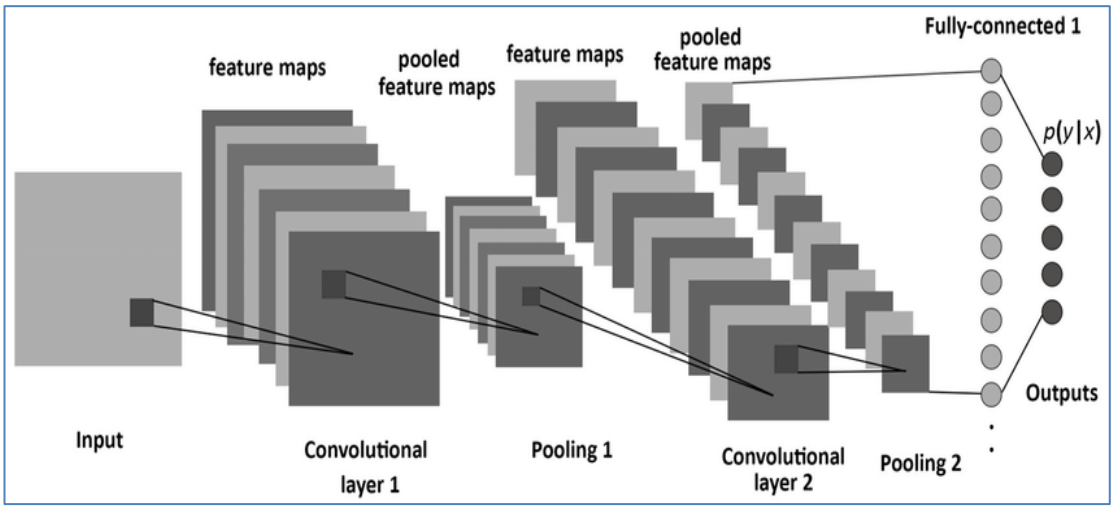


Figure II. 8: Les réseaux de neurones convolutifs.

1. Couche Convolution:

Une couche convolutive est le composant le plus important d'un CNN. Il comprend un ensemble de filtres (également appelés noyaux convolutifs) qui sont convolués avec une entrée donnée pour générer une carte de caractéristiques en sortie. **[8]**

✍ **Filtre:** chaque filtre d'une couche convolutive est une grille de nombres discrets. A titre d'exemple, considérons un filtre 2×2 illustré à la Figure.II.9. Les poids de chaque filtre (les nombres dans la grille) sont apprises lors de la formation de CNN. Ensuite, étant donné les paires d'entrées-sorties, les poids du filtre sont réglés dans un certain nombre d'itérations différentes au cours de la procédure d'apprentissage. **[8]**

2	**0**
-1	**3**

Figure II.9: Un exemple de filtre d'image 2D.

✍ **Opération de convolution:** Nous avons mentionné précédemment que la couche de convolution effectue une convolution entre les filtres et l'entrée de la couche. Considérons une convolution 2D sur la figure II.10 pour développer un aperçu du fonctionnement de la couche. Étant donné une carte de caractéristiques d'entrée 2D et un filtre de convolution de tailles de matrice 4×4 et 2×2, respectivement, une couche de convolution multiplie le filtre 2×2 avec un patch en surbrillance (également 2×2) de la carte de caractéristiques d'entrée et résume toutes les valeurs pour générer une valeur dans la carte des caractéristiques en sortie. Notez que le filtre glisse le long de la largeur et de la hauteur de la carte des caractéristiques en entrée et ce processus se poursuit jusqu'à ce que le filtre ne puisse plus glisser plus loin. **[8]**

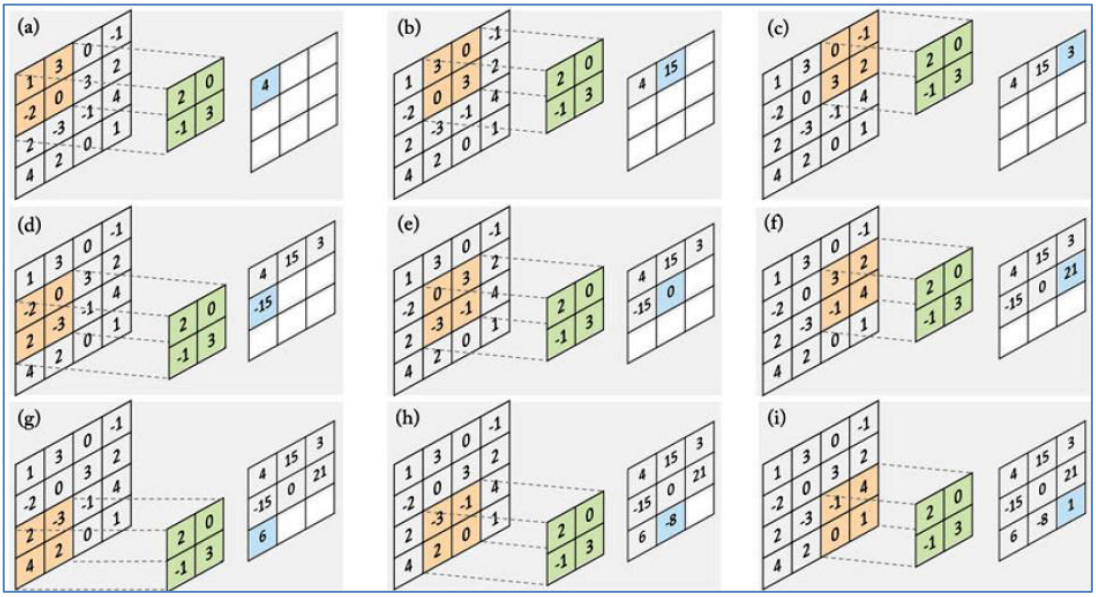


Figure II.10: Le fonctionnement d'une couche de convolution

Le fonctionnement d'une couche de convolution est illustré dans la figure ci-dessus. (a) à (i) montrent les calculs effectués à chaque étape, lorsque le filtre est glissé sur la carte des caractéristiques en entrée pour calculer la valeur correspondante dans la carte des caractéristiques en sortie. Le filtre 2×2 (affiché en vert) est multiplié par la même région de taille (affichée en orange) dans une carte de caractéristiques d'entrée 4×4 et les valeurs résultantes sont additionnées pour obtenir une entrée correspondante (affichée en bleu) dans la sortie carte des caractéristiques à chaque étape de convolution. **[8]**

Dans l'exemple ci-dessus, afin de calculer chaque valeur de la carte des caractéristiques en sortie, le filtre effectue un pas de 1 le long de la position horizontale ou verticale (c'est-à-dire le long de la colonne ou de la ligne de l'entrée). Cette étape est appelée la foulée (Stride) du filtre de convolution, qui peut être réglé sur une valeur différente (de 1) si nécessaire. Par exemple, l'opération de convolution avec une foulée de 2 est illustrée à la figure II.11. Par rapport à la foulée de 1 dans l'exemple précédent, la foulée de 2 donne une carte des caractéristiques en sortie plus petite. Cette réduction des dimensions est appelée opération de sous-échantillonnage. Une telle réduction des dimensions fournit une invariance modérée à l'échelle et à la pose des objets, ce qui est une propriété utile dans des applications telles que la reconnaissance d'objets. **[8]**

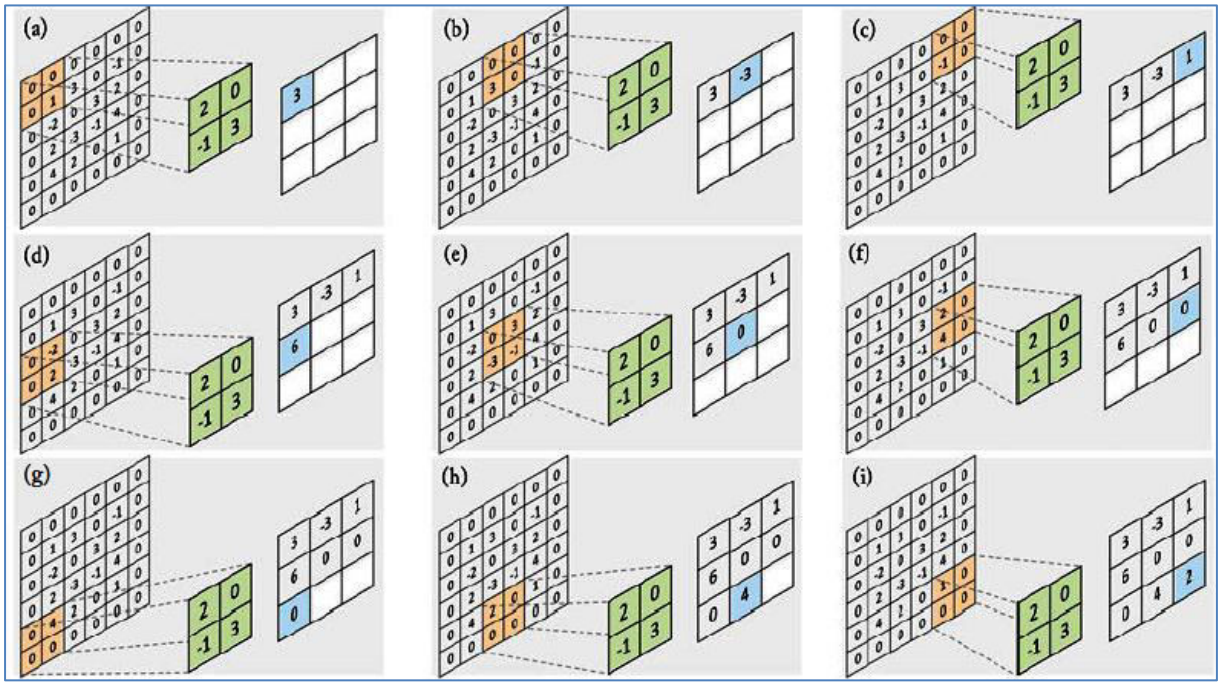


Figure II.11: Le fonctionnement d'une couche de convolution avec un remplissage nul de 1 et une foulée de 2.

Le fonctionnement d'une couche de convolution avec un remplissage nul de 1 et une foulée de 2 est illustré dans la figure ci-dessus. (a) à (i) montrent les calculs effectués à chaque étape, lorsque le filtre est glissé sur la carte des caractéristiques en entrée pour calculer la valeur correspondante de la carte des caractéristiques en sortie. Le filtre 2×2 (affiché en vert) est multiplié par la même région de taille (affichée en orange) dans une carte de caractéristiques d'entrée 6×6 (y compris le remplissage de zéro) et les valeurs résultantes sont additionnées pour obtenir une entrée correspondante (affichée en bleu) dans la carte des caractéristiques en sortie à chaque étape de convolution. **[8]**

Nous avons vu sur la figure II.10 que la taille spatiale de la carte des caractéristiques en sortie est réduite par rapport à la carte des caractéristiques en entrée. Précisément, pour un filtre de taille $f \times f$, une carte des caractéristiques en entrée de taille $h \times w$ et une longueur de foulée s, les dimensions des caractéristiques en sortie sont données par: **[8]**

$$h' = \left\lfloor \frac{h-f+s}{s} \right\rfloor, \quad w' = \left\lfloor \frac{w-f+s}{s} \right\rfloor,$$

Où, $\lfloor . \rfloor$ désigne l'opération de plancher. Cependant, dans certaines applications, telles que le débruitage d'image, super-résolution, ou segmentation, nous voulons garder les tailles spatiales constantes (ou même plus grandes) après la convolution. Ceci est important car ces applications nécessitent des prédictions plus denses au niveau du pixel. De plus, cela nous permet de concevoir des réseaux plus profonds (c'est-à-dire avec plus de couches de poids) en évitant un effondrement rapide des dimensions des caractéristiques de sortie. Cela permet d'obtenir de meilleures performances et des étiquetages de sortie à plus haute résolution. Ceci peut être réalisé en appliquant un remplissage de zéro autour de la carte des caractéristiques en entrée. Comme le montre la Figure. II.11, zéro remplissage (zero-padding) de l'horizontale et verticale dimensions nous permet d'augmenter les dimensions de sortie et donne donc plus de flexibilité dans la conception de l'architecture. L'idée de base est d'augmenter la taille de la carte des caractéristiques en entrée de manière à obtenir une carte des caractéristiques en sortie, avec les dimensions souhaitées. Si p désigne l'augmentation de la carte des caractéristiques en entrée le long de chaque dimension (en remplissant des zéros), nous pouvons représenter les dimensions de la carte des caractéristiques en sortie modifiées comme suit: **[8]**

$$h' = \left\lfloor \frac{h-f+s+p}{s} \right\rfloor, \quad w' = \left\lfloor \frac{w-f+s+p}{s} \right\rfloor,$$

Sur la figure II.13, p = 2 et donc les dimensions de sortie ont été augmentées de 6×6 à 3×3. Si les couches convolutives ne remplissaient pas les entrées et n'appliquaient que des convolutions valides, alors la taille spatiale de la sortie les caractéristiques seront réduites d'un petit facteur après chaque couche de convolution et les informations aux frontières seront « lavées » trop rapidement. **[8]**

2. Couche de Pooling:

Une couche de mise en commun opère sur des blocs de la carte de caractéristiques d'entrée et combine les activations de caractéristiques. Cette opération de combinaison est définie par une fonction de mise en commun telle que la moyenne ou la fonction max. Semblable à la couche de convolution, nous devons spécifier la taille de la région regroupée et la foulée. La figure II.12 montre l'opération de regroupement maximal, où l'activation maximale est choisie dans le bloc de valeurs sélectionné. Cette fenêtre est glissée sur les cartes de caractéristiques en entrée avec une taille de pas définie par la foulée (1 dans le cas de la figure II.12). Si la taille de la région regroupée est donnée par $f \times f$, avec une foulée s, la taille de la carte des caractéristiques en sortie est donnée par: **[8]**

$$h' = \left\lfloor \frac{h-f+s}{s} \right\rfloor, \quad w' = \left\lfloor \frac{w-f+s}{s} \right\rfloor,$$

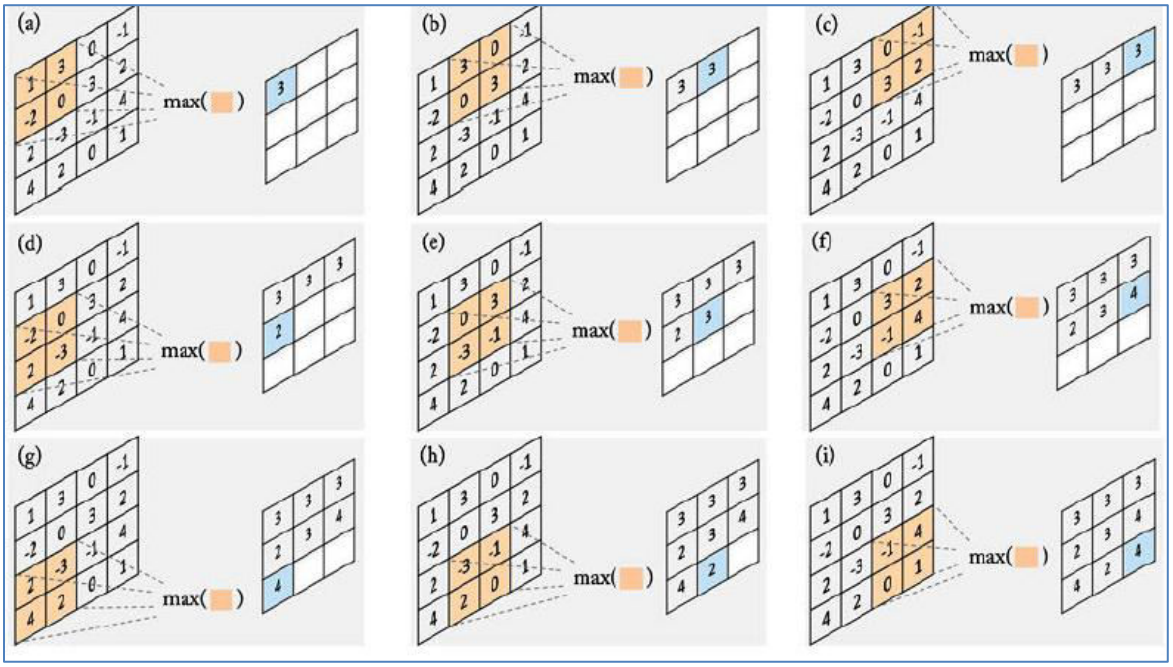


Figure II.12: Le fonctionnement de la couche max-pooling lorsque la taille de la région de pooling est de 2×2 et la foulée est de 1.

(a)–(i) montre les calculs effectués à chaque étape lorsque la région regroupée dans la carte des caractéristiques en entrée (affichée en orange) est glissée à chaque étape pour calculer la valeur correspondante dans la carte des caractéristiques en sortie (affichée en bleu).

3. Couche Fully Connected:

Les couches entièrement connectées correspondent essentiellement à des couches de convolution avec des filtres de taille 1×1. Chaque unité d'une couche entièrement connectée est densément connectée à toutes les unités de la couche précédente. Dans un CNN typique, les couches entièrement connectées sont généralement placées vers la fin de l'architecture. Cependant, certaines architectures réussies sont rapportées dans la littérature qui utilisent ce type de couche à un emplacement intermédiaire au sein d'un CNN. Son fonctionnement peut être représenté comme une simple multiplication matricielle suivie de l'ajout d'un vecteur de termes de biais et de l'application d'une fonction non linéaire par élément $f(.)$:

$$\mathbf{y} = \mathbf{f}(\mathbf{W}^{\mathbf{T}}\mathbf{x} + \mathbf{b})$$

Où $\boldsymbol{x}$ et $\boldsymbol{y}$ sont respectivement le vecteur des activations d'entrée et de sortie, $\boldsymbol{W}$ désigne la matrice contenant les poids des connexions entre les unités de couche, et $\boldsymbol{b}$ représente le vecteur de terme de biais.

✍ CNN Architectures:

1. LeNet [LeCun et al, 1998] **:** L'architecture est l'une des formes les plus anciennes et de base des CNN qui a été appliqué à l'identification des chiffres manuscrits. Une variante réussie de ce style d'architecture s'appelle le modèle LeNet-5, car il comprend 5 couches de poids au total. **[9]**

2. **AlexNet** [Krizhevsky et al, 2012] : A été le premier modèle CNN à grande échelle qui a conduit à la résurgence des réseaux de neurones profonds dans la vision par ordinateur. Cette architecture a remporté le prix ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) en 2012 par une large marge. **[9]**

3. **VGGNet** [Simonyan and Zisserman, 2014]: Est l'un des modèles CNN les plus populaires depuis son introduction en 2014, même s'il n'a pas été le vainqueur de l'ILSVRC'14. La raison de sa popularité réside dans la simplicité de son modèle et l'utilisation de noyaux convolutifs de petite taille qui conduisent à des réseaux très profonds. Les auteurs ont introduit un ensemble de configurations de réseau, parmi lesquelles les configurations D et E (communément appelées VGGnet-16 et VGGnet-19) sont les plus réussies. **[9]**

4. **GoogleNet** [Szegedy et al, 2015] : Est le premier modèle populaire qui utilise une architecture plus complexe avec plusieurs branches de réseau. Par la suite, plusieurs versions améliorées et étendues de GoogleNet ont également été proposées. **[9]**

5. **ResNet** [He et al, 2016] : Le réseau résiduel de Microsoft a remporté le défi ILSVRC 2015 avec un grand saut de performance. La caractéristique remarquable de l'architecture du réseau résiduel est le saut d'identité des connexions dans les blocs résiduels, ce qui permet de former facilement des architectures CNN très profondes. **[9]**

6. ResNeXt (2017) **:** L'architecture ResNeXt combine les points forts des conceptions GoogleNet et ResNet. Plus précisément, il utilise des connexions à saut qui ont été proposées pour les réseaux résiduels et les combine avec l'architecture multi-branches dans le module de création. **[9]**

II.5.3.3.2. Les réseaux neuronaux récurrents (RNN):

Un réseau de neurones récurrents est un type de réseau de neurones artificiels qui est le meilleur adapté à la reconnaissance de modèles dans des séquences de données, telles que du texte, de la vidéo, données sur la parole, le langage, les génomes et les séries chronologiques. Un RNN est extrêmement algorithme puissant qui peut classer, regrouper et faire des prédictions sur données, en particulier les séries chronologiques et le texte. **[10]**

RNN peut être vu comme un réseau MLP avec ajout de boucles à l'architecture. Dans la figure II.14 vous pouvez voir qu'il existe une couche d'entrée (avec des nœuds tels que x1, x2, etc.), une couche cachée (avec des nœuds tels que h1, h2, etc.) et une couche de sortie (avec des nœuds tels que y1, y2, etc.). Ceci est similaire à l'architecture MLP. La différence est que les nœuds des couches cachées sont interconnectés. Dans un RNN/LSTM vanille (de base), les nœuds sont connectés dans une direction. Cela signifie que h2 dépend de h1 (et x2), et h3 dépend de h2 (et x3). Le nœud dans la couche cachée est déterminé par le nœud précédent dans la couche cachée.

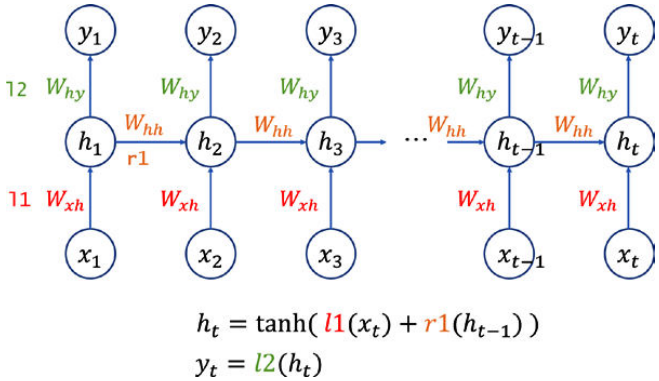


Figure II.14: Un RNN

✍ Types de réseaux de neurones récurrents:

Il existe quatre types de réseaux de neurones récurrents: **[11]**

1. Un à un RNN:

Ce type de réseau de neurones est connu sous le nom de réseau de neurones Vanilla. Il est utilisé pour les problèmes généraux d'apprentissage automatique, qui ont une seule entrée et une seule sortie.

2. Un à plusieurs:

Ce type de réseau de neurones a une seule entrée et plusieurs sorties. Un exemple de ceci est la légende de l'image.

3. Plusieurs à un:

Ce RNN prend une séquence d'entrées et génère une seule sortie. L'analyse des sentiments est un bon exemple de ce type de réseau où une phrase donnée peut être classée comme exprimant des sentiments positifs ou négatifs.

4. Plusieurs à plusieurs:

Ce RNN prend une séquence d'entrées et génère une séquence de sorties. La traduction automatique en est un exemple.

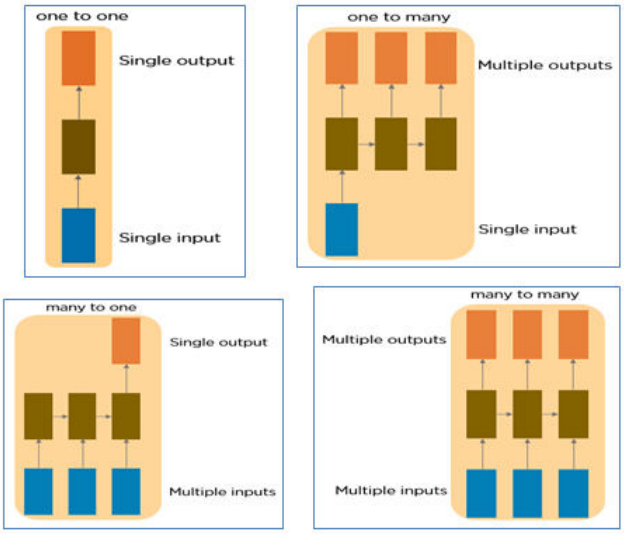


Figure II.15: Les types de séquences d'entrée pour un réseau récurrent

✍ Architectures RNN variantes:

1. Réseau de neurones récurrent simple (RNN simple):

Est essentiellement un ensemble de réseaux de neurones communs agencés ensemble, chacun d'eux transmettant un message à un autre. En d'autres termes, ces réseaux ont une mémoire qui stocke des connaissances sur les données vues, mais leur mémoire est à court terme et ne peut pas maintenir des séries temporelles à long terme. Un réseau récurrent simple n'a qu'une seule mémoire interne. **[12]**

2. Mémoire longue à court terme (LSTM):

Est une sorte de modèle ou de structure de données séquentielles développé par pour l'avancement de RNN. Il utilise une combinaison spéciale d'unités cachées, de produits par élément et de sommes entre les unités pour mettre en œuvre des portes qui contrôlent les «cellules de mémoire». Ces cellules sont conçues pour conserver des informations sans modification pendant de longues périodes. La plus grande caractéristique de LSTM réside dans sa capacité à apprendre la dépendance à long terme. Pour prédire l'étape suivante, les valeurs de poids sur le réseau doivent être mises à jour, ce qui nécessite la maintenance des informations des étapes initiales. LSTM peut apprendre correctement ces dépendances à long terme, et LSTM a trois portes : entrée, oubli et sortie. La porte d'oubli est intégrée pour indiquer combien la mémoire précédente se souvient et combien elle a oublié. **[12]**

3. Unité récurrente fermée (GRU):

Est un type simple de LSTM suggéré par Cho et al. La différence avec LSTM est que GRU fusionne les portes d'entrée et d'oubli et les convertit avec une porte de mise à jour. Par conséquent, GRU a moins de paramètres que LSTM, ce qui facilite la formation. **[12]**

II.6. Conclusion:

Dans ce chapitre, nous avons introduit des concepts importants liés à l'apprentissage en profondeur (Histoire, Définition, Domaines d'application), ensuite nous avons présenté Le réseau neuronal (Définition, Les neurones biologiques, Les neurones formels), À la fin de ce chapitre, Les réseaux neuronaux convolutionnels (Les différentes couches, les Architectures) et Les réseaux neuronaux récurrents (les Types, les Architectures)

Chapitre III
Réalisation

III.1. Présentation des outils de travail:

III.1.1. Software:

III.1.1.1. Le Langage de Programmation Python:

Python est un langage orienté objet de haut niveau, il est l'un des langages de programmation les plus efficaces pour le développement Web. Python est utilisé dans différents domaines d'application (industrie, application de recherche et scientifique…).

Voici quelques raisons de la popularité de Python:

- Syntaxe de haut niveau (par rapport aux langages de niveau inférieur C, Java et C++).
- Les applications peuvent être développées en écrivant moins de lignes de code, ce qui rend Python attrayant pour les débutants comme pour les programmeurs avancés.
- Cycle de vie de développement efficace.
- Vaste collection de bibliothèques open source gérées par la communauté.
- Forte portabilité.

La simplicité de Python a attiré de nombreux développeurs pour créer de nouvelles bibliothèques pour l'apprentissage automatique, conduisant à une forte adoption de Python.[1]

III.1.1.2. Jupyter Notebook:

Créé à partir de Python en 2014, **Jupyter est un notebook de calcul** (computational notebook) open source, gratuit et interactif. C'est une application web basée client permettant de créer et de partager du code, des équations, des visualisations ou du texte. Son nom est en fait une **référence à trois langages de programmation : Julia, Python et R**. En effet, Jupyter prend en charge plus d'une quarantaine de langages de programmation.

III.1.1.3. Packages pour ML:

Les principaux packages Python utilisés pour l'apprentissage automatique sont mis en évidence dans Figure III.1.

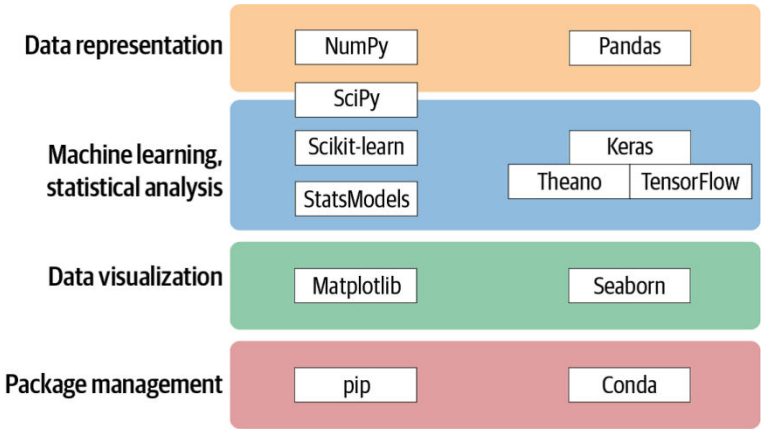


Figure III.1: Python packages

Voici un bref résumé de chacun de ces forfaits: [1]

- **NumPy:** Prend en charge les grands tableaux multidimensionnels ainsi qu'une vaste collection de fonctions mathématiques.
- **Pandas :** Une bibliothèque pour la manipulation et l'analyse des données. Entre autres fonctionnalités, il offre des données structures pour manipuler les tables et les outils pour les manipuler.
- **Matplotlib:** Une bibliothèque de traçage qui permet la création de graphiques et de tracés 2D.
- **Scikit-learn (ou sklearn):** Une bibliothèque d'apprentissage automatique offrant une large gamme d'algorithmes et d'utilitaires.
- **Seaborn:** Une bibliothèque de visualisation de données basée sur Matplotlib. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs.
- **Pip et Conda :** Ce sont des gestionnaires de packages Python. pip est un gestionnaire de packages qui facilite l'installation, la mise à niveau et la désinstallation des packages Python. Conda est un paquet gestionnaire qui gère les packages Python ainsi que les dépendances de bibliothèque en dehors de les packages Python.

III.1.2.Hardware:

Un ordinateur personnel PACKARD BELL:

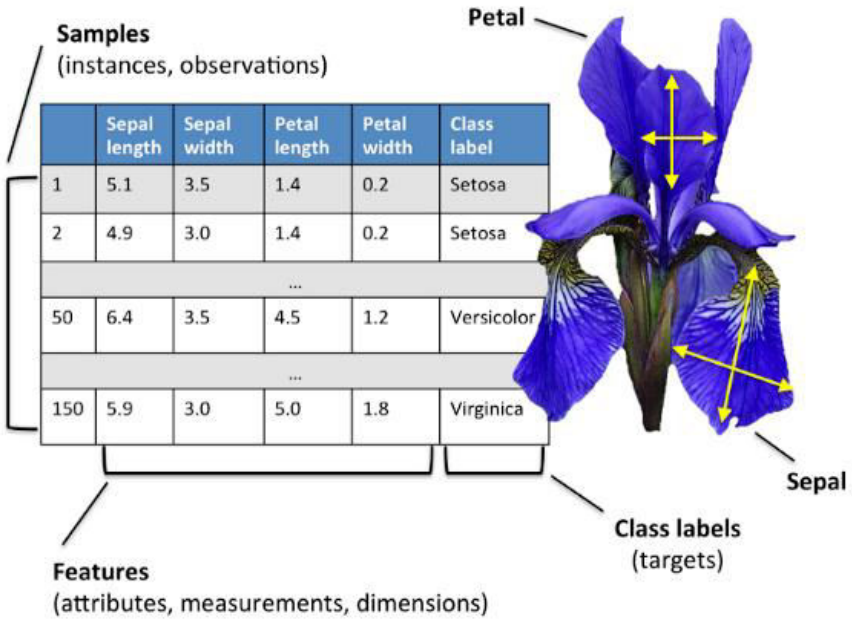
- Processeur : Intel(R) Core (TM) i3 CPU M 380 @ 3.53GHz 3.53 GHz.
- Mémoire installée (RAM) : 2.00 Go.
- Système d'exploitation 64 bits.
- Microsoft Windows 7.

III.3. Dataset:

L'ensemble de données sur l'iris contient quatre caractéristiques (longueur et largeur des sépales et des pétales) de 50 échantillons de trois espèces d'iris (Iris setosa, Iris virginica et Iris versicolor). Ces mesures ont été utilisées pour créer un modèle discriminant linéaire pour classer les espèces. L'ensemble de données est souvent utilisé dans des exemples d'exploration de données, de classification et de clustering et pour tester des algorithmes.[2]

✍ Fonctionnalités de l'ensemble de données Iris :

1. longueur des sépales en cm (sepal length).
2. largeur des sépales en cm (sepal width).
3. longueur des pétales en cm (petal length).
4. largeur des pétales en cm (petal width).



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Figure III.2: Description visuelle des diverses caractéristiques des espèces d'iris

✍ Classes cibles à prévoir :

1. Iris Setosa.
2. Iris Versicolor.
3. Iris Virginie.

Figure III.3. Fleurs de trois espèces de plantes d'iris

Information about Dataset:

Les cinq premières lignes de données pour chaque classe:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Ensemble de données complet sur l'iris:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

Exemples d'entrées dans l'ensemble de données Iris:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Statistiques de base :

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

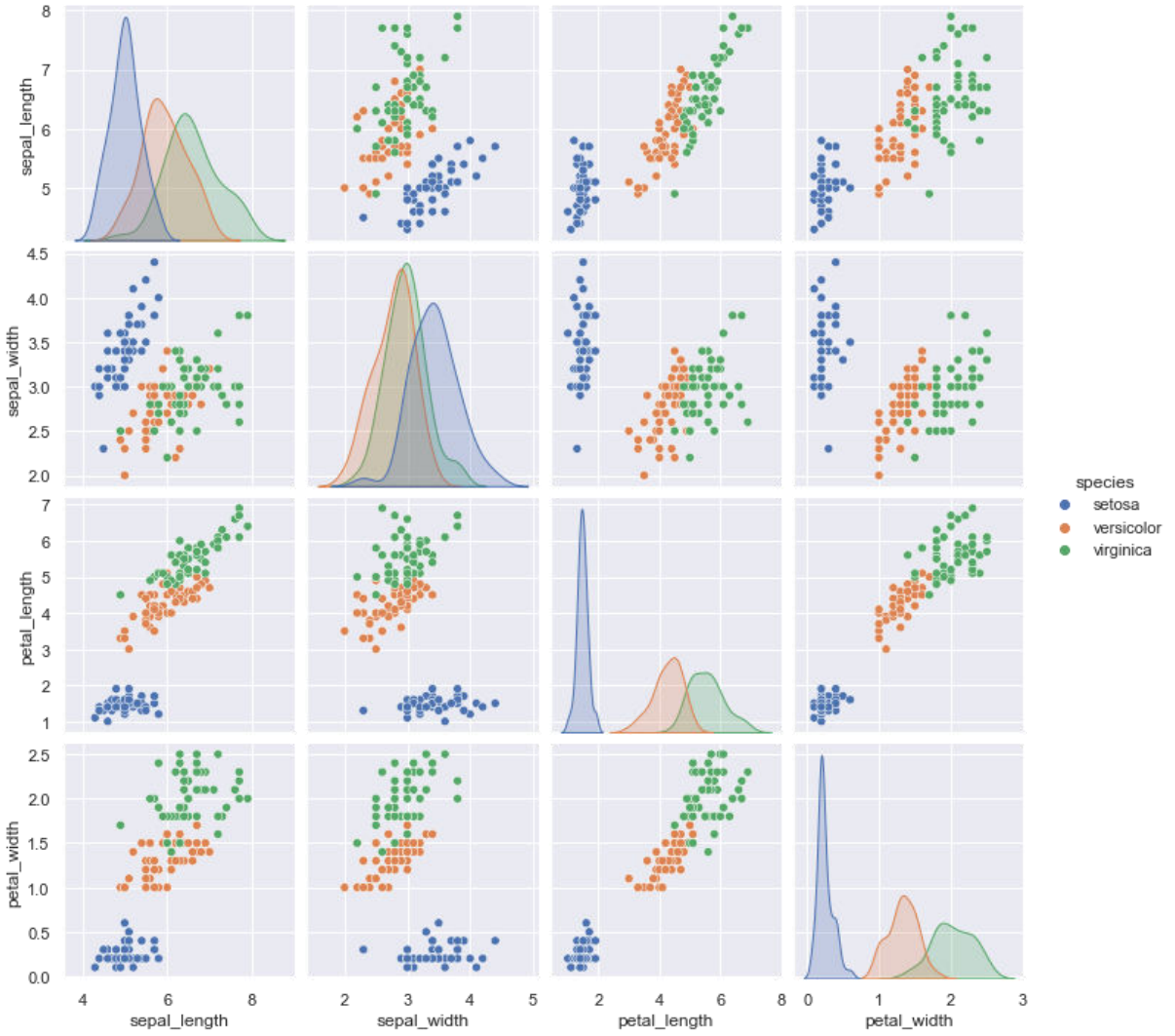
Informations à partir de données:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

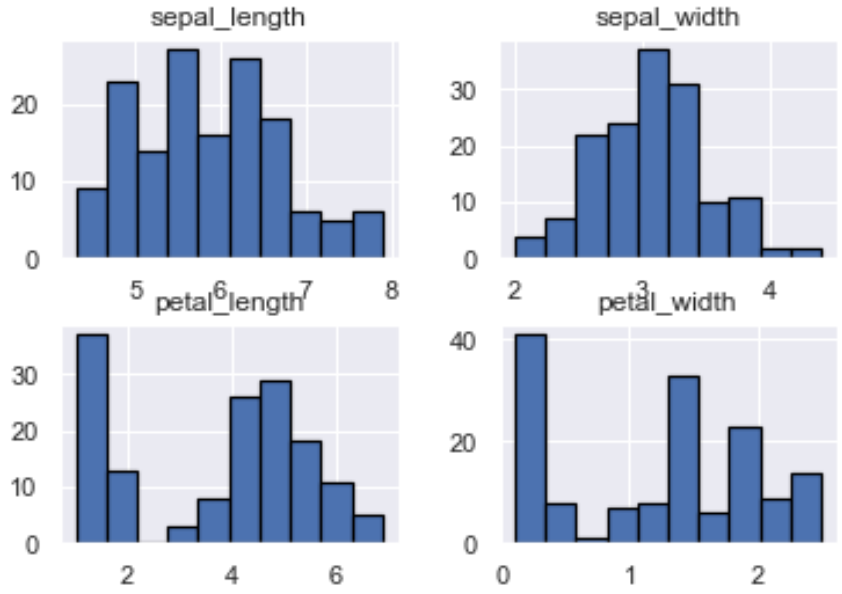
Les caractéristiques individuelles comptent :

```
species
setosa        50
versicolor    50
virginica     50
dtype: int64
```

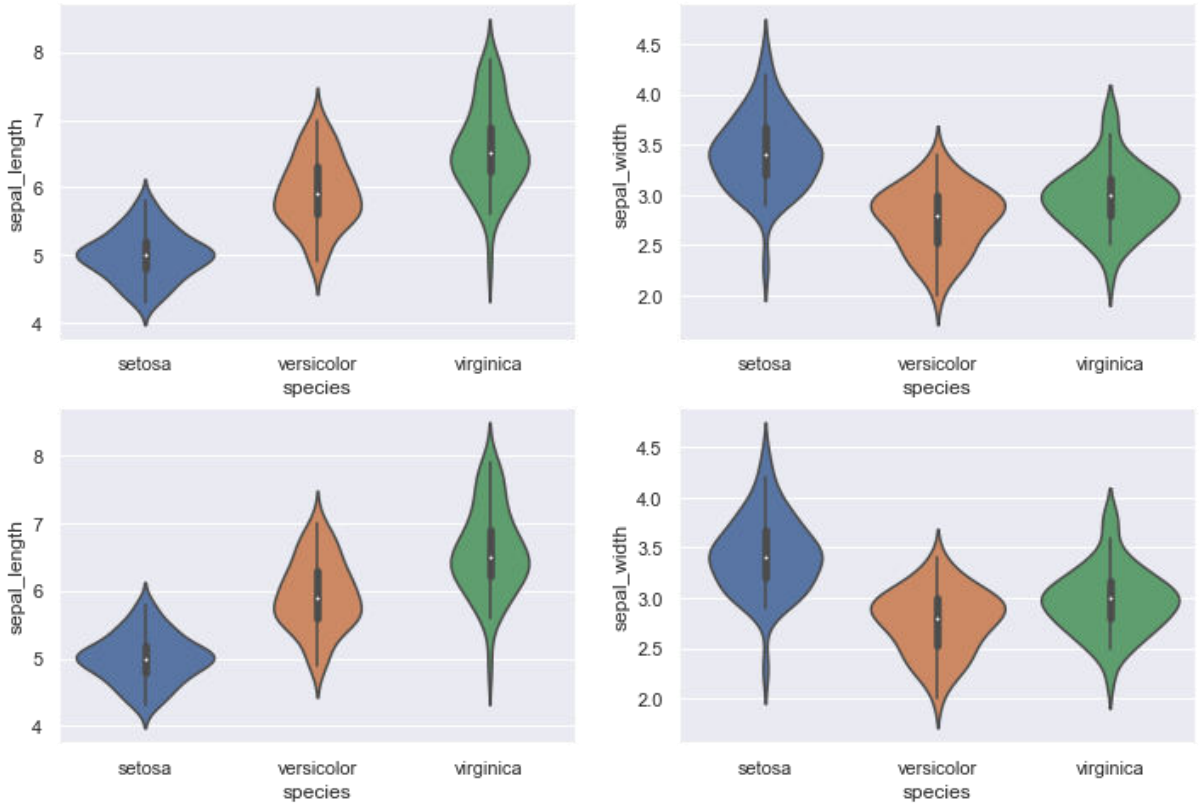
Visualisation des données



Répartition de la longueur et de la largeur :



La longueur et la largeur diffèrent selon les espèces :



III.4. Expérimentations et interprétations:

✍ k-fold cross validation:

Utilisation de la validation croisée (K-Fold cross validation) pour partitionner la rame en une formation et un ensemble de validation.

1. Importer tous les éléments requis:

```
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
```

2. Lire les données de l'iris et créer X (caractéristiques) et y (réponse) :

```
# read in the iris data
iris = load_iris()
# create X (features) and y (response)
X = iris.data
y = iris.target
```

3. Divisez les données en ensembles d'apprentissage et de test à l'aide de la fonction train_test_split :

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.10)
```

4. Instanciez la classe K-Fold avec une configuration 10 fois :

```
kf = KFold(n_splits = 10)
```

5. Appliquez la méthode split aux données dans X. Cette méthode crée 10 configurations de fractionnement différentes. Avec la sortie enregistrée dans une variable appelée divisions:

```
splits = kf.split(X)
```

6. Effectuez une boucle "**for**" qui passe par les différentes configurations de division. Dans le corps de la boucle, des variables sont créées qui contiendront les données d'apprentissage et les ensembles de validation :

```
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Résultats du k-fold cross validation:

TRAIN: [15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [30 31 32 33 34 35 36 37 38 39 40 41 42 43 44]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [45 46 47 48 49 50 51 52 53 54 55 56 57 58 59]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [60 61 62 63 64 65 66 67 68 69 70 71 72 73 74]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [75 76 77 78 79 80 81 82 83 84 85 86 87 88 89]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [90 91 92 93 94 95 96 97 98 99 100 101 102 103 104]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [105 106 107 108 109 110 111 112 113 114 115 116 117 118 119]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]
TEST: [120 121 122 123 124 125 126 127 128 129 130 131 132 133 134]

TRAIN: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134]
TEST: [135 136 137 138 139 140 141 142 143 144 145 146 147 148 149]

Figure III.4 : Résultats du k-fold cross validation

III.5. Conclusion:

Dans ce chapitre nous avons donné les outils nécessaires pour la réalisation de ce travail (Software, Hardware). Nous avons aussi présenté l'environnement de développement (Le Langage de Programmation Python, Jupyter Notebook, Packages pour ML), à la fin nous avons montré Information about Dataset les résultats obtenus ainsi que quelques captures d'écran qui explique le déroulement de l'entrainement.

Conclusion générale

Conclusion Générale

Dans cette mémoire, j'ai exploré le défi de la sélection de modèles pour l'apprentissage automatique.

L'utilisation correcte des techniques de sélection de modèles et de sélection d'algorithmes est vitale dans la recherche en apprentissage automatique. Il existe plusieurs techniques et algorithmes différents qui peuvent être utilisés, parmi lesquels des techniques de validation croisée courantes telles que la validation croisée de sortie une fois et la validation de pli k.

Pour réaliser ce travail nous avons utilisé le deep learning comme méthode, et le choix de cette méthode se justifie par la simplicité et l'efficacité de ses méthodes. et en fournissant le meilleur modèle de travail neuronal pour ajuster les données avec une erreur d'entraînement minimale en utilisant la validation k-fold.

Référence

Les Références

Référence chapitre I:

[1] Voire la source: https://datascience.stackexchange.com/questions/66729/difference-between-empirical-risk-minimization-and-structural-risk-minimization, Consulté le juillet 2021.

[2] Machine learning and electronic health records Sivert Stavland, University of Bergen, 2020. Voire la source: https://bora.uib.no/bora-xmlui/handle/1956/24107?show=full, Consulté le juillet 2021.

[3] Henry de-Graft Acquah, Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of an asymmetric price relationship, 2009, Voire la source: https://academicjournals.org/journal/JDAE/article-abstract/650D3294276, Consulté le juillet 2021.

[4] Hyatt Saleh, The Machine Learning Workshop, Second Edition, Get ready to develop your own high-performance machine learning algorithms with scikit-learn.

[5] Mansi Goel, The Bias-Variance Trade-Off : A Mathematical View, Jul 2020, Voire la source: https://medium.com/snu-ai/the-bias-variance-trade-off-a-mathematical-view-14ff9dfe5a3c, Consulté le juillet 2021.

[6] Hariom Tatsat, Sahil Puri & Brad Lookabaugh, Machine Learning & Data Science Blueprints for Finance, From Building Trading Strategies to Robo-Advisors Using Python.

Référence chapitre II:

[1] Yousef Djeriri, Les Réseaux de Neurones Artificiels, Septembre 2017. Voire la source: https://www.researchgate.net/publication/319939107_Les_Reseaux_de_Neurones_Artificiels, Consulté le juin 2021.

[2] Haohan Wang, Bhiksha Raj, On the Origin of Deep Learning, Voire la source: https://arxiv.org/abs/1702.07800, Consulté le juin 2021.

[3] Voire la source: https://www.slideshare.net/akira-ai/introduction-of-machine-learning-deep-learning-and-its-types, Consulté le juin 2021.

[4] Samuel Rikli, Wasserstein Gan: Deep Generation Applied On Financial Time Series, Voire la source: https://arxiv.org/pdf/2107.06008, Consulté le juin 2021

[5] Nacima OUKACINE : "Utilisation des réseaux de neurones pour la reconstitution de défauts en évaluation non destructive". Mémoire de Magister. Université Mouloud Mammeri, Tizi-ouzou, 2012.

[6] G. Dreyfus, J.-M. Martinez, M. Samuelides M. B. Gordon, F. Badran, S. Thiria, Apprentissage statistique», Eyrolles, Paris, 2004.

[7] Ehrlich, Thery, L'apport de l'analyse de sentiments dans l'amélioration des services de support, Haute École de Gestion de Genève (HEG-GE), Octobre 2019, Voire la source: http://doc.rero.ch/record/327853/files/TB_Thery_Ehrlich.pdf. Consulté le juin 2021.

[8] Salman Khan, A Guide to Convolutional Neural Networks for Computer Vision (SYNTHESES LECTURES ON COMPUTER VISION), 2018, Voire la source: http://libgen.li/ads.php?md5=57b8ce1f6209368cc51c1ef4511a1736. Consulté le juin 2021

[9] Voire la source: https://www.jeremyjordan.me/convnet-architectures/amp/, Consulté le juin 2021.

[10] Tarry Singh, Deep Learning with Applications Using Python, Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras.

[11] Voire la source: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn, Consulté le juin 2021.

[12] Halit Apaydin and Hajar Feizi, Comparative Analysis of Recurrent Neural Network Architectures for Reservoir Inflow Forecasting, Publié May 2020. Voire la source: https://www.researchgate.net/publication/341609757_Comparative_Analysis_of_Recurrent_Neural_Network_Architectures_for_Reservoir_Inflow_Forecasting, Consulté le juin 2021.

Référence chapitre III:

[1] Hariom Tatsat, Sahil Puri & Brad Lookabaugh, Machine Learning & Data Science Blueprints for Finance, From Building Trading Strategies to Robo-Advisors Using Python.

[2] Voire la source: https://archive.ics.uci.edu/ml/datasets/Iris, Consulté le août 2021.