République Algérienne Démocratique et Populaire
Université Abderrahmane MIRA de Béjaïa
Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire Présenté pour L'obtention du Diplôme de Master
en Mathématiques Appliquées

Spécialité : mathématique financières

Méthode LSTM versus méthodes traditionnelles de séries temporelles : étude comparative sur la consommation électrique — Cas de l'entreprise Sonelgaz

Présenté par :
HADJARA TAHAR AMINE
Encadré par :
M'LAMIA DJERROUD

Composition du Jury :

M. S. Amroun	Président du jury	MCB	Université de Béjaïa
Mme B. Takhedmit	Examinateur	MCA	Université de Béjaïa
M. S. L. Abbassi	Examinateur	MCA	Université de Béjaïa

Année Universitaire 2024 – 2025

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'énergie joue un rôle fondamental dans le développement économique et social d'un pays. Parmi les différentes formes d'énergie, l'électricité occupe une place centrale dans la vie quotidienne des citoyens, le fonctionnement des services publics, ainsi que dans l'activité industrielle et commerciale. Pour répondre aux besoins croissants en électricité, il est indispensable pour les opérateurs du secteur énergétique d'assurer une gestion efficace de la production, du transport et de la distribution de cette ressource.

L'un des défis majeurs dans cette gestion est la capacité à anticiper la demande. Une bonne prévision de la consommation électrique permet non seulement de garantir un approvisionnement stable et sécurisé, mais aussi d'optimiser les ressources, de réduire les pertes et de limiter les coûts. Cette capacité de prévision est d'autant plus cruciale dans un contexte où la consommation devient de plus en plus variable et dépendante de multiples facteurs économiques, climatiques et sociaux.

Depuis plusieurs décennies, des méthodes statistiques ont été utilisées pour modéliser et prévoir les séries temporelles liées à la consommation d'électricité. Ces méthodes, telles que les modèles de lissage ou les modèles ARIMA, sont basées sur l'analyse des valeurs passées d'une série pour en déduire une tendance future. Elles sont bien établies et offrent des résultats satisfaisants dans de nombreux cas.

Cependant, avec l'évolution technologique et l'émergence de nouvelles techniques issues de l'intelligence artificielle, des approches plus performantes ont vu le jour. Parmi elles, les réseaux de neurones, en particulier les modèles LSTM (Long Short-Term Memory), se sont imposés comme des outils puissants capables de capturer des relations complexes au sein des données temporelles, en tenant compte des dépendances à long terme.

Dans ce mémoire, nous proposons une étude comparative entre les méthodes statistiques classiques et les techniques d'apprentissage profond pour la prévision de la consommation électrique. Cette étude est appliquée à des données réelles issues de la Direction de Distribution de Béjaïa, une filiale du groupe SONELGAZ.

L'objectif est d'analyser les avantages et les limites de chaque approche afin de déterminer les conditions dans lesquelles l'une pourrait s'avérer plus adaptée que l'autre. Pour cela, plusieurs modèles seront construits, évalués et comparés à l'aide d'indicateurs de performance standard.

Ce mémoire est structuré en plusieurs chapitres. Le premier présente l'entreprise SONELGAZ et le contexte de l'étude. Le second chapitre est consacré aux méthodes statistiques de prévision. Le troisième porte sur les méthodes d'apprentissage profond, en particulier le modèle LSTM. Le quatrième chapitre présente l'étude comparative sur des données réelles. Enfin, une conclusion générale vient clore ce travail en résumant les principaux résultats.

Chapitre 1

Présentation de l'entreprise SONELGAZ

1.1 Introduction

Ce chapitre vise à présenter l'environnement dans lequel s'inscrit notre travail de mémoire, à savoir le groupe SONELGAZ et plus précisément la Direction de Distribution de Béjaïa. Il s'agit d'introduire l'entreprise, sa structure organisationnelle, ses missions et son rôle stratégique dans le secteur énergétique national. En outre, le chapitre présente le contexte pratique et les enjeux de notre mémoire, qui porte sur la prévision de la consommation électrique à l'aide de méthodes traditionnelles et de techniques d'intelligence artificielle. Une compréhension approfondie du fonctionnement de SONELGAZ permet de mieux cerner les besoins opérationnels auxquels répond cette étude.

1.2 Historique de SONELGAZ

SONELGAZ est créée le 28 juillet 1969, en remplacement de l'entité précédente Électricité et gaz d'Algérie (EGA), et on lui a donné un monopole de la distribution et de la vente de gaz naturel dans le pays, de même pour la production, la distribution, l'importation, et l'exportation d'électricité. En 2002, le décret présidentiel n° 02-195, la convertit en une société par actions SPA entièrement détenue par l'État.

En septembre 2013, SONELGAZ achète neuf centrales électrique à General Electric pour un montant de 2,7 milliards de dollars. La puissance totale de ces six centrales est de plus de 8 000 mégawatts, permettant d'augmenter la capacité de production de l'Algérie en électricité de 70 % . Un partenariat entre les deux groupes est prévu dans le cadre de ce contrat pour la fabrication en Algérie d'équipements de production d'électricité.

Le 19 mars 2014, SONELGAZ et General Electric ont signé un accord de partenariat à long terme pour la construction d'un complexe industriel en Algérie, appelé General Electric Alegria Turbine (GEAT). Il a une capacité de fabrication de matériel de production d'électricité (turbines à gaz, turbines à vapeur, alternateurs et systèmes de contrôle-commande) représentant 2 000 MW par an. Le complexe permettra également de créer environ un millier d'emplois directs sur le territoire de Aïn Yagout dans la wilaya de Batna. La première pierre a été posée en septembre 2014. Le projet est lancé au deuxième semestre de 2016 et il est inauguré en 2020. En 2021, l'entreprise a conclu un marché avec un client du Moyen-Orient pour la vente de deux turbines à gaz pour la production de l'électricité avec leurs équipements connexes d'une capacité de production de 500 mégawatts. Il s'agit de la première commande à l'international.

1.3 Présentation de SONELGAZ

SONELGAZ est l'opérateur historique dans le domaine de la fourniture des énergies électrique et gazière en Algérie. Crée en 1969, SONELGAZ, œuvre depuis un demi-siècle au service du citoyen algérien en lui apportant cette source énergétique essentielle à la vie quotidienne. A la faveur de la promulgation de la loi sur l'électricité et la distribution du gaz par canalisations, SONELGAZ est passée d'une entreprise verticalement intégrée à une holding pilotant un Groupe industriel multi-sociétés et multi-métiers.

SONELGAZ a toujours joué un rôle majeur dans le développement économique et social du pays. Sa contribution dans la concrétisation de la politique énergétique nationale est à la mesure des importants programmes réalisés, en matière d'électrification rurale et de distribution publique gaz ; ce qui a permis de hisser le taux de couverture en électricité à 98 % pour 10 494 000 clients et un taux de pénétration du gaz à 65 % pour 6 451 000 clients

SONELGAZ est aujourd'hui érigé en Groupe industriel composé de 35 filiales et 6 sociétés en participation. Ainsi, les filiales métiers de base assurent la production, le transport et la distribution de l'électricité ainsi que le transport et la distribution du gaz par canalisations. On compte :

- La Société Algérienne de Production de l'Électricité (SPE),
- La Société Algérienne de Gestion du Réseau de Transport de l'Électricité (GRTE),
- L'Opérateur Système électrique (OS), chargée de la conduite du système Production / Transport de l'électricité,
- La Société Algérienne de Distribution de l'électricité et du gaz d'Alger (SDA),
- La Société Algérienne de Distribution de l'électricité et du gaz du Centre (SDC),
- La Société Algérienne de Distribution de l'électricité et du gaz de l'Est (SDE),
- La Société Algérienne de Distribution de l'électricité et du gaz de l'Ouest (SDO).

1.4 Présentation de la Direction de Distribution de Béjaïa (CD)

La Direction de Distribution de Béjaïa est rattachée à la société Algériennes de Distribution de l'électricité et du gaz de l'Est (SDE), dont le siège se trouve à Constantine, cette dernière est composée d'une Direction à laquelle sont reliés directement :

- Le secrétariat.
- Les assistants du Directeur de Distribution.
- Le chargé des affaires juridiques.
- Le chargé de la communication.
- 9 Divisions (DRC, DTE, DTG, DEET, DAM, DFC, DRH, DGSI, SAG).
- 10 Agences commerciales (BEJAIA CITE TOBAL, BEJAIA 4 CHEMAIN, EL KSEUR, AMIZOUR, SIDI AICH, SEDDOUK, AKBOU, TAZMALT, AOKAS, KHERRATA).
- 5 Districts (BEJAIA, AKBOU, SIDI AICH, AMIZOUR, KHERRATA)

Chaque division est composée de plusieurs services qui assurent des missions précises : maintenance des réseaux, traitement des raccordements, suivi de la consommation, gestion des ressources humaines et comptabilité.

1.5 Organisation de la Sonalgaz Distribution CD Béjaïa

La structure organisationnel des différentes divisions de la Sonelgaz Distribution CD BEJAIA et présenté dans la figure 1.1

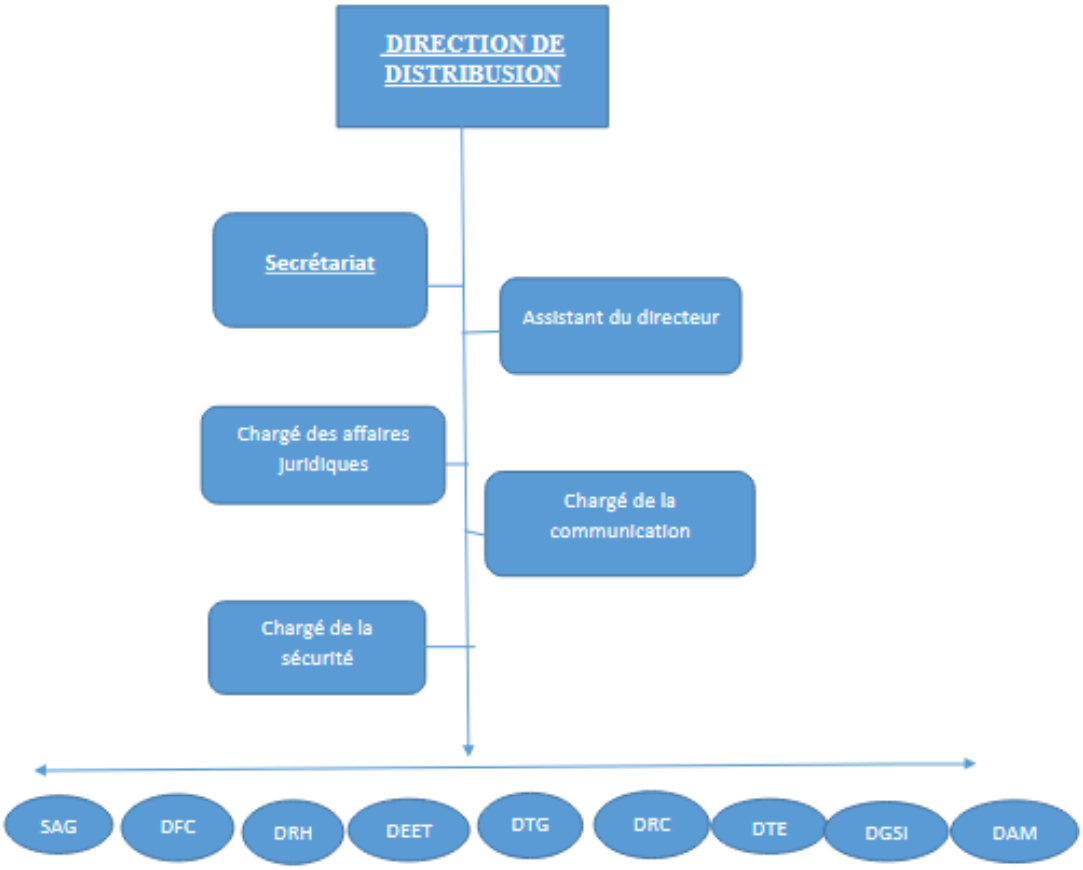

FIG. 1.1 : Organigramamme de la Direction de Distribution de Béjaïa

1.6 Contexte du mémoire

Dans un contexte marqué par une croissance continue de la demande énergétique, il devient crucial pour SONELGAZ d'améliorer la qualité de ses prévisions de consommation, afin d'optimiser la planification de l'offre, de réduire les pertes et d'éviter les surcharges du réseau.

Le présent mémoire s'inscrit dans cette optique et vise à évaluer la performance des méthodes classiques de prévision face aux approches plus modernes basées sur les réseaux de neurones profonds.

L'étude est menée à partir de données réelles de consommation électrique fournies par la Direction de Distribution de Béjaïa. Elle compare les résultats des modèles, dans le but de proposer des solutions adaptées aux besoins opérationnels de SONELGAZ. **Enjeux :**

— Améliorer la précision des prévisions à court et moyen terme,
— Aider à la planification énergétique régionale,
— Valoriser l'usage des techniques d'intelligence artificielle dans les systèmes énergétiques.

1.7 Conclusion

Ce premier chapitre a permis d'établir un cadre général autour de notre mémoire. La présentation de SONELGAZ et de sa Direction de Distribution de Béjaïa nous a permis de mieux comprendre l'importance stratégique de la prévision de la consommation électrique dans le fonctionnement de l'entreprise. À travers ce contexte, nous avons introduit les motivations et les objectifs de notre étude, qui s'inscrit pleinement dans une démarche d'optimisation des outils de prévision énergétique.

Chapitre 2

les méthodes statistiques de la prévision.

2.1 Introduction

Prévoir l'évolution d'un phénomène est une tâche essentielle dans de nombreux domaines comme l'économie, la finance ou la gestion des ressources. Pour cela, on utilise des méthodes statistiques qui permettent d'analyser les données passées afin d'anticiper les valeurs futures.

Ces méthodes reposent sur des calculs et des modèles qui aident à mieux comprendre les variations des données et à faire des estimations plus précises. Elles sont largement utilisées pour aider à la prise de décision et optimiser la gestion des ressources.

Dans ce chapitre, nous allons introduire les principes fondamentaux de l'analyse des séries temporelles. Ces notions sont essentielles pour comprendre le fonctionnement des principales méthodes statistiques de prévision, ainsi que leurs avantages et leurs limites.

2.2 les concepts fondamentaux

2.2.1 Séries temporelles

Définition 2.1. Une série temporelle (ou chronologique) est une suite d'observations réalisées séquentiellement dans le temps [5]. Elle peut être définie sur des intervalles réguliers ou irréguliers. Si les observations sont faites à des intervalles de temps constants, on parle de série temporelle régulière ; sinon, la série est dite irrégulière.

Les séries temporelles peuvent être de deux types : discrètes, lorsque les données sont relevées à des moments précis, ou continues, lorsque les données sont enregistrées de façon continue dans le temps. En analyse statistique, on travaille le plus souvent avec des séries discrètes.

Exemple 2.1
les températures relevées chaque jour, La consommation de gaz enregistrée chaque jour (ou chaque mois), le chiffre d'affaires mensuel d'une entreprise.

On note généralement une série temporelle par :

$$Y_1, Y_2, Y_3 \ldots, Y_N$$

où chaque Y_t représente la valeur observée à l'instant t. la figure 2.1 présente les données de consommation de fuel lourd en France de janvier 1983 à juin 1999 sont présentées par [32].

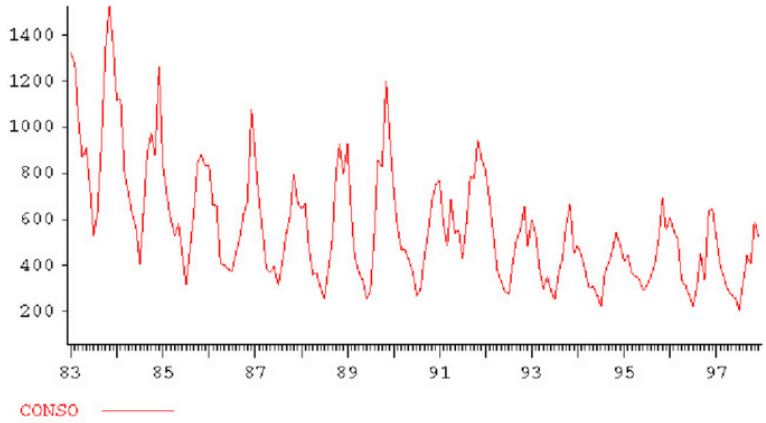

FIG. 2.1 : Les données de consommation de fuel lourd en France

2.2.2 Les différentes composantes d'une série temporelles

Une série chronologique a trois composantes : une composante tendance, une composante saisonnière et une composante résiduell [26] :

- **la composante tendance** (ou trend) (T_t) représente le mouvement général de la série sur le long terme. Elle peut être croissante, décroissante ou stable. Elle ne doit pas nécessairement être linéaire. Parfois, on dira d'une tendance qu'elle "change de direction", lorsqu'elle passe d'une tendance à la hausse à une tendance à la baisse , exemple dans la figure 2.2.
- **La composante saisonnière** (ou saisonnalité) (S_t) correspond à un phénomène qui se répète à intervalles de temps réguliers (périodiques) exemple dans la figure 2.2 . comme les cycles annuels (pensez aux ventes de glaces qui augmentent en été). En général, c'est un phénomène saisonnier d'où le terme de variations saisonnières.
- **La composante résiduelle** (ou bruit aléatoire ou résidu) (ε_t) correspond à des fluctuations irrégulières, en général de faible intensité mais de nature aléatoire exemple dans la figure 2.2.

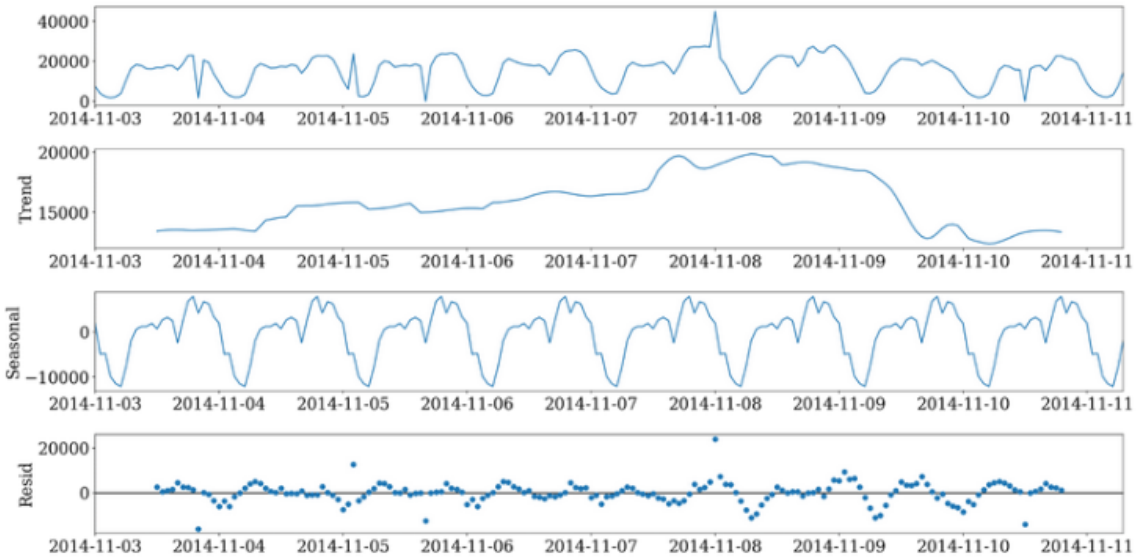

FIG. 2.2 : Résultat de décomposition d'une série temporelle en ses composants

2.2.3 Schémas d'une série temporelle

Il existe deux principaux types de schémas pour modéliser les séries temporelles :

Modèle additif

Ce modèle est souvent utilisé lorsque les fluctuations saisonnières sont indépendantes du niveau de la série. Une observation Y_t est alors décomposée en trois composantes :

$$Y_t = T_t + S_t + \varepsilon_t \tag{2.1}$$

où T_t représente tendance, S_t est la saisonnalité et ε_t est le bruit aléatoire ou résidu.

Ce modèle est pertinent lorsque l'amplitude des variations saisonnières reste constante (la figure 2.3), quelle que soit l'évolution de la tendance.

Modèle multiplicatif

Ce modèle est préférable lorsque les fluctuations saisonnières varient proportionnellement avec le niveau de la série. La série est alors modélisée ainsi :

$$Y_t = T_t \times S_t \times \varepsilon_t \tag{2.2}$$

La décomposition additive est la plus appropriée si l'ampleur des fluctuations saisonnières autour du tendance ne varie pas avec le niveau de la série temporelle. Lorsque la variation du modèle saisonnier autour du tendance semble proportionnelle au niveau de la série temporelle, une décomposition multiplicative est alors plus appropriée [22].

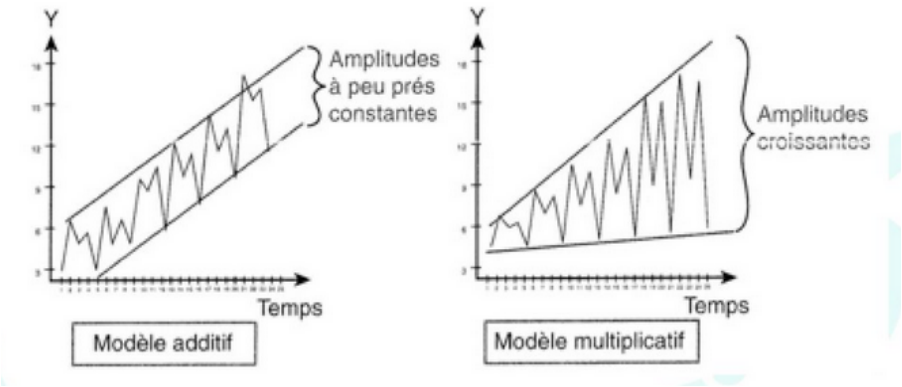

FIG. 2.3 : Exemples de schémas additif et multiplicatif .

Il peut être difficile de travailler directement avec une relation multiplicative. Une solution alternative est d'appliquer une transformation logarithmique aux données. Cela permet de transformer le produit en une somme, ce qui facilite les calculs et l'analyse :

$$Y_t = T_t \cdot S_t \cdot \varepsilon_t$$

est équivalent à

$$\log Y_t = \log S_t + \log T_t + \log \varepsilon_t.$$

2.3 Modèles à base de lissage

Les modèles à base de lissage sont parmi les méthodes les plus anciennes et les plus simples utilisées pour analyser et prévoir les séries temporelles. Leur idée principale est d'atténuer les variations brusques ou aléatoires dans les données en « lissant » la courbe des observations passées. Cela permet une meilleure visualisation des tendances et parfois les effets saisonniers.

Historiquement, ces méthodes ont commencé à être utilisées dès le début du XX^{e} siècle. Le lissage par moyenne mobile a d'abord été utilisé dans l'analyse économique pour suivre les cycles d'activité. Plus tard, dans les années 1950 et 1960, des chercheurs comme Charles Holt, Peter Winters et Robert G. Brown ont proposé des méthodes plus avancées, comme le lissage exponentiel, qui donnent plus de poids aux données récentes.

Ces méthodes sont encore très utilisées aujourd'hui, car elles sont simples, rapides à appliquer, et efficaces lorsque la structure de la série n'est pas trop complexe.

2.3.1 Lissage par la moyenne mobile

Le lissage par la moyenne mobile [11] est la méthode la plus simple pour lisser une série. Elle consiste à remplacer chaque valeur par la moyenne des k dernières observations. Cela permet de réduire l'effet des fluctuations aléatoires.

La formule est :

$$\hat{y}_t = \frac{1}{k} \sum_{i=0}^{k-1} y_{t-i} \tag{2.3}$$

où k est la taille de la fenêtre (par exemple 3 mois), y_t est la valeur observée à l'instant t, et $\hat{y}_t$ est la valeur lissée.

Il existe plusieurs types de moyennes mobiles, notamment :

— **Simple :** toutes les valeurs dans la fenêtre ont le même poids.
— **Pondérée :** les valeurs récentes ont plus de poids que les anciennes.
— **Centrée :** la moyenne est centrée autour du point observé (utile en analyse, mais pas pour la prévision).

Cette méthode est utile pour visualiser la tendance, mais elle ne permet pas de prédire ce qui va se passer plus tard.

Exemple 2.2
Considérons la série suivante représentant la consommation mensuelle d'électricité (en kWh) pour une entreprise sur six mois :

$$y = [120,\ 130,\ 125,\ 140,\ 135,\ 145]$$

Appliquons une moyenne mobile simple avec une fenêtre $k = 3$. On calcule alors :

$$\hat{y}_3 = \frac{1}{3}(120 + 130 + 125) = \frac{375}{3} = 125$$

$$\hat{y}_4 = \frac{1}{3}(130 + 125 + 140) = \frac{395}{3} \approx 131.67$$

$$\hat{y}_5 = \frac{1}{3}(125 + 140 + 135) = \frac{400}{3} \approx 133.33$$

$$\hat{y}_6 = \frac{1}{3}(140 + 135 + 145) = \frac{420}{3} = 140$$

Ainsi, la série lissée devient :

$$\hat{y} = [\text{NA},\ \text{NA},\ 125,\ 131.67,\ 133.33,\ 140.00]$$

Les deux premières valeurs ne sont pas calculées car il faut au moins trois données pour faire une moyenne avec une fenêtre de taille 3. La série lissée devient alors plus régulière, ce qui aide à mieux voir la tendance générale en réduisant les variations d'un mois à l'autre.

2.3.2 Les lissages exponentiels

Les lissages exponentiels sont des méthodes simples utilisées pour lisser une série temporelle et produire des prévisions. Leur principe repose sur l'attribution d'un poids décroissant aux anciennes valeurs : les données récentes ont plus d'influence que les anciennes.

Contrairement à la moyenne mobile simple, où toutes les valeurs ont le même poids, ou à la moyenne mobile pondérée, où les poids sont fixés manuellement, le lissage exponentiel diminue automatiquement les poids de manière exponentielle.

En 1958, **Charles Holt** a introduit une méthode tenant compte de la tendance. En 1960, **Winters** a intégré la saisonnalité. Le livre de **Brown** [7], publié en 1963, a ensuite popularisé ces méthodes dans les domaines de la prévision et de la gestion des stocks.

Il existe plusieurs variantes du lissage exponentiel, adaptées à différents types de séries :

— Lissage exponentiel simple .

— Lissage exponentiel double (Holt) .

— Lissage exponentiel triple (Holt-Winters).

Nous allons voir ces méthodes dans les sections suivantes.

2.3.3 Lissage exponentiel simple

Le lissage exponentiel simple [6] constitue une amélioration de la moyenne mobile, dans laquelle les pondérations décroissent de manière exponentielle avec le temps. Ce modèle est adapté aux séries temporelles sans tendance ni saisonnalité.

Il repose sur une seule composante appelée **niveau** ℓ_t, qui représente l'estimation actuelle de la série. Ce niveau est mis à jour à chaque pas de temps selon la formule suivante :

$$\ell_t = \alpha y_t + (1 - \alpha)\ell_{t-1}$$

Dans la pratique, on écrit souvent la formule de prévision sous la forme suivante, équivalente :

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t \tag{2.4}$$

avec :

— $0 < \alpha < 1$: paramètre de lissage,

— y_t : valeur observée à l'instant t,

— $\hat{y}_t$ ou ℓ_t : estimation du niveau à l'instant t.

Exemple 2.3

Considérons une série de consommation mensuelle (y_t) sur 10 mois :

t	1	2	3	4	5	6	7	8	9	10
y_t	120	127	125	130	128	131	129	134	137	135

Prenons $\alpha = 0{,}3$ et $\hat{y}_1 = y_1 = 120$.

On applique la formule (2.4) :

$$\hat{y}_2 = 0{,}3 \times 127 + 0{,}7 \times 120 = 38{,}1 + 84 = 122{,}1$$
$$\hat{y}_3 = 0{,}3 \times 125 + 0{,}7 \times 122{,}1 = 37{,}5 + 85{,}47 = 122{,}97$$
$$\hat{y}_4 = 0{,}3 \times 130 + 0{,}7 \times 122{,}97 = 39 + 86{,}079 = 125{,}079$$
$$\hat{y}_5 = 0{,}3 \times 128 + 0{,}7 \times 125{,}079 = 38{,}4 + 87{,}555 = 125{,}955$$
$$\hat{y}_6 = 0{,}3 \times 131 + 0{,}7 \times 125{,}955 = 39{,}3 + 88{,}1685 = 127{,}4685$$
$$\hat{y}_7 = 0{,}3 \times 129 + 0{,}7 \times 127{,}4685 = 38{,}7 + 89{,}228 = 127{,}928$$
$$\hat{y}_8 = 0{,}3 \times 134 + 0{,}7 \times 127{,}928 = 40{,}2 + 89{,}5496 = 129{,}7496$$
$$\hat{y}_9 = 0{,}3 \times 137 + 0{,}7 \times 129{,}7496 = 41{,}1 + 90{,}8247 = 131{,}9247$$
$$\hat{y}_{10} = 0{,}3 \times 135 + 0{,}7 \times 131{,}9247 = 40{,}5 + 92{,}3473 = 132{,}8473$$

Prévision pour le mois suivant ($t = 11$) :

$$\hat{y}_{11} = 0{,}3 \times 135 + 0{,}7 \times 132{,}8473 = 40{,}5 + 92{,}9931 = 133{,}4931$$

Le paramètre α joue un rôle essentiel dans le lissage :
— Si α est proche de 1, le modèle donne plus d'importance aux nouvelles données. Cela permet de suivre rapidement les changements, mais peut aussi rendre les prévisions plus instables.
— Si α est proche de 0, le modèle change lentement. Cela permet d'obtenir une courbe plus lisse, en supprimant les petites variations. Mais il met plus de temps à réagir quand les données changent vraiment.

En pratique, on choisit α en testant plusieurs valeurs et en gardant celle qui donne les meilleurs résultats.

2.3.4 Lissage exponentiel double (Holt)

Le lissage exponentiel double [20](ou méthode de Holt) est une extension du lissage exponentiel simple. Il permet de modéliser des séries temporelles qui présentent une tendance linéaire. Pour cela, il introduit deux composantes :
— ℓ_t : le niveau estimé de la série au temps t,
— b_t : tendance estimée au temps t.

Les formules de mise à jour sont les suivantes :

$$\begin{cases} \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\ \hat{y}_{t+h} = \ell_t + h b_t \end{cases}$$

où :
— α est le coefficient de lissage du niveau,
— β est le coefficient de lissage de la tendance,
— $\hat{y}_{t+h}$ est la prévision à h pas de temps dans le futur.

Ce modèle est bien adapté lorsque la série présente une tendance, mais sans saisonnalité.

Exemple 2.4
On reprend la même série temporelle que celle présentée dans l'exemple 2.3. Prenons les paramètres suivants : $\alpha = 0{,}3$, $\beta = 0{,}2$, $\ell_1 = y_1 = 120$ et $b_1 = y_2 - y_1 = 7$.

On calcule alors pas à pas :

$$\ell_2 = 0{,}3 \times 127 + 0{,}7 \times (120 + 7) = 38{,}1 + 88{,}9 = 127{,}0$$
$$b_2 = 0{,}2 \times (127{,}0 - 120) + 0{,}8 \times 7 = 1{,}4 + 5{,}6 = 7{,}0$$
$$\ell_3 = 0{,}3 \times 125 + 0{,}7 \times (127{,}0 + 7{,}0) = 37{,}5 + 93{,}8 = 131{,}3$$
$$b_3 = 0{,}2 \times (131{,}3 - 127{,}0) + 0{,}8 \times 7{,}0 = 0{,}86 + 5{,}6 = 6{,}46$$
$$\ell_4 = 0{,}3 \times 130 + 0{,}7 \times (131{,}3 + 6{,}46) = 39 + 96{,}902 = 135{,}902$$
$$b_4 = 0{,}2 \times (135{,}902 - 131{,}3) + 0{,}8 \times 6{,}46 = 0{,}9204 + 5{,}168 = 6{,}0884$$

etc.

La prévision à $t = 5$ est donc $\hat{y}_5 = \ell_4 + b_4 \approx 141{,}99$.

Ce modèle est particulièrement efficace pour les séries avec une tendance, mais sans composante saisonnière.

2.3.5 Méthode de Holt-Winters

Holt (1957) et Winters (1960) [43] prolonge le modèle de Holt en ajoutant une composante saisonnière. La méthode saisonnière de Holt-Winters comprend trois composantes : Le niveau l_t, et La tendance b_t, et La saisonnalité s_t.

Il existe deux variantes selon le type de saisonnalité : additive ou multiplicative.

Modèle additif

Ce modèle est approprié lorsque l'amplitude de la saisonnalité reste constante. la mise à jour des composantes :

$$\begin{aligned} l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - l_t) + (1-\gamma)s_{t-m} \end{aligned}$$

où : s_t : composante saisonnière à l'instant t et m Période de la saisonnalité (exemple :m = 12 pour des données mensuelles) et γ Coefficient de lissage de la saisonnalité ($0 \leq \gamma \leq 1$) .

Pour la prévision à horizon h :

$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m} \tag{2.5}$$

Modèle multiplicatif

Ce modèle est utilisé lorsque l'amplitude de la saisonnalité varie proportionnellement à la tendance. La mise à jour de ses composants :

$$\begin{aligned} l_t &= \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma\frac{y_t}{l_t} + (1-\gamma)s_{t-m} \end{aligned}$$

Pour la prévision à horizon h :

$$\hat{y}_{t+h} = (l_t + hb_t) \cdot s_{t+h-m} \tag{2.6}$$

Les modèles de lissage exponentiel sont souvent utilisés pour prévoir les séries temporelles, car ils sont simples et efficaces.

Ils ont plusieurs points forts :

— Faciles à utiliser et à comprendre ;
— Peuvent suivre la tendance et la saisonnalité ;
— Utilisent peu de paramètres.

Ils fonctionnent bien quand les données changent de façon régulière.

Mais ils ont aussi des limites :

— Leur performance dépend du bon choix des paramètres (α, β, γ) ;
— Moins adaptés aux séries irrégulières ou influencées par des événements imprévus ;
— Peu flexibles pour des situations plus complexes.

Après avoir exploré les modèles à base de lissage , nous allons maintenant nous intéresser à une autre approche classique et puissante pour la modélisation des séries temporelles : la méthodologie Box-Jenkins.

2.4 Les modèles à base de la Méthodologie Box-Jenkins

La méthodologie de Box-Jenkins, développée dans les années 1970 par George Box et Gwilym Jenkins [5], propose une approche structurée pour analyser et prévoir les séries temporelles. Elle repose sur l'idée que les valeurs futures peuvent être estimées à partir des valeurs passées et des erreurs précédentes.

Cette méthode est particulièrement adaptée aux séries stationnaires, c'est-à-dire dont les caractéristiques statistiques restent stables dans le temps. Si la série ne l'est pas, elle peut souvent être transformée pour le devenir.

Les modèles Box-Jenkins combinent trois éléments principaux :

— la composante **autorégressive** (AR) : basée sur les valeurs passées de la série ;
— la composante **moyenne mobile** (MA) : basée sur les erreurs passées de prévision ;
— la composante **d'intégration** (I) : utilisée pour rendre la série stationnaire si nécessaire.

Ensemble, ces éléments forment les modèles ARIMA (AutoRegressive Integrated Moving Average), qui peuvent être adaptés pour gérer également la saisonnalité (modèles SARIMA).

La démarche comprend généralement quatre étapes : l'identification du modèle,l'estimation des paramètres,la validation du modèle et enfin la prévision.

Avant d'entrer dans les détails, il est important de présenter quelques notions clés, comme les processus stochastiques, la stationnarité ou encore les fonctions d'autocorrélation.

2.4.1 Processus stochastique

Un processus stochastique est une suite de valeurs aléatoires dans le temps, notée $\{X_t\}$. Cela signifie que, pour chaque instant t, la valeur observée est le résultat d'un phénomène en partie aléatoire.

Dans le contexte des séries temporelles, on observe une seule suite de données : par exemple, la consommation d'électricité mois par mois. On suppose que cette série observée est une réalisation particulière d'un processus stochastique. Cela nous permet de modéliser les dépendances dans le temps et de faire des prévisions, en considérant comment les valeurs passées influencent les valeurs futures.

Bruit blanc

Le bruit blanc est un type particulier de processus stochastique qui ne présente aucune structure dans le temps. On le note souvent $\{\varepsilon_t\}$. Il se caractérise par trois propriétés :

— une espérance nulle : $\mathbb{E}[\varepsilon_t] = 0$;
— une variance constante : $\mathrm{Var}(\varepsilon_t) = \sigma^2$;
— aucune corrélation entre les valeurs : $\mathrm{Cov}(\varepsilon_t, \varepsilon_{t+k}) = 0$ pour $k \neq 0$.

Ainsi, dans la méthodologie Box-Jenkins, après ajustement du modèle, on vérifie que les erreurs (différence entre les valeurs observées et les valeurs prédites) forment un bruit blanc pour s'assurer que toute l'information contenue dans la série a été capturée.

Décalages temporels

Un décalage temporel, ou *lag*, correspond à la valeur passée d'une série temporelle. Si l'on note une série $\{X_t\}$, alors X_{t-1} désigne la valeur observée à la période précédente.

Exemple 2.5
si X_t représente la température aujourd'hui, alors X_{t-1} est la température d'hier, X_{t-2} celle d'avant-hier, etc.

Opérateur retard

L'opérateur retard (ou *backshift operator*), noté B, est un outil formel utilisé pour simplifier l'écriture des modèles de séries temporelles. Il est défini par :

$$BX_t = X_{t-1}$$

c'est-à-dire qu'il décale la série d'une période vers le passé. On a aussi : $B^2X_t = X_{t-2}$, $B^3X_t = X_{t-3}$, etc.

Autocorrélation et autocorrélation partielle

L'**autocorrélation**, aussi appelée fonction d'autocorrélation (ACF pour *Autocorrelation Function*) mesure la dépendance entre les valeurs d'une série temporelle à différents décalages (ou lags). Elle permet d'observer dans quelle mesure les valeurs passées influencent les valeurs futures.

Pour un processus stationnaire $\{X_t\}$, l'autocorrélation au décalage k est définie par :

$$\rho(k) = \frac{\mathrm{Cov}(X_t, X_{t-k})}{\mathrm{Var}(X_t)}$$

Elle varie entre -1 et 1. Une valeur proche de 1 ou de -1 indique une forte corrélation à ce lag, tandis qu'une valeur proche de 0 indique une absence de dépendance.

L'**autocorrélation partielle**, notée PACF (*Partial Autocorrelation Function*), mesure la corrélation entre X_t et X_{t-k} en supprimant l'effet des lags intermédiaires 1 à $k-1$. Elle permet ainsi d'isoler l'effet direct d'un lag spécifique.

Les fonctions ACF et PACF sont souvent représentées sous forme de graphiques appelés corrélogrammes. Pour mieux interpréter ces graphes, on trace généralement des bandes de confiance (souvent à 95 %). Si une barre dépasse cette bande, la corrélation à ce lag est statistiquement significative.

2.4.2 Processus stationnaire

Un processus stochastique est dit **stationnaire** lorsque ses propriétés statistiques ne changent pas au cours du temps [5] [9].

Plus précisément, un processus $\{X_t\}$ est *stationnaire au sens faible* si les trois conditions suivantes sont vérifiées :

— L'espérance mathématique est constante : $\mathbb{E}[X_t] = \mu$ pour tout t.
— La variance est constante : $\text{Var}(X_t) = \sigma^2$ pour tout t.
— $\text{Cov}(X_t, X_{t+k}) = \gamma(h)$ pour tout t et tout k, La covariance entre deux observations ne dépend que du décalage entre elles.

Exemple 2.6
Le *bruit blanc* $\{\varepsilon_t\}$ est défini comme une suite de variables aléatoires indépendantes et identiquement distribuées (i.i.d.) de moyenne nulle et de variance constante . Il satisfait les trois conditions ci-dessus et constitue donc un exemple de processus stationnaire .

La méthodologie Box-Jenkins supposent que la série est stationnaire. Si la série ne l'est pas, il est souvent nécessaire de la transformer pour pouvoir la modéliser correctement.

2.4.3 Modèle à moyenne mobile MA(q)

Le modèle à moyenne mobile d'ordre q, ou MA(q), est un processus stochastique stationnaire dans lequel chaque observation est exprimée comme une combinaison linéaire d'un bruit blanc présent et de ses valeurs passées [15].

Sous l'hypothèse usuelle où la série est centrée, i.e. $\mathbb{E}[X_t] = 0$, le modèle MA(q) s'écrit :

$$X_t = \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q} \tag{2.7}$$

où : ε_t est un bruit blanc et $\theta_1, \ldots, \theta_q$ sont les paramètres du modèle.

À l'aide de l'opérateur retard B, défini par $B^k\varepsilon_t = \varepsilon_{t-k}$, On peut donc réécrire le modèle MA(q) sous forme compacte :

$$X_t = (1 + \theta_1 B^1 + \theta_2 B^2 + \cdots + \theta_q B^q)\varepsilon_t = \theta(B)\varepsilon_t$$

Le processus MA(q) est toujours **stationnaire** car il dépend d'un nombre fini de perturbations aléatoires .

Exemple 2.7
Le modèle MA(1) est défini par :

$$X_t = \varepsilon_t + \theta_1\varepsilon_{t-1}$$

2.4.4 Modèle autorégressif AR(p)

Le modèle autorégressif d'ordre p, ou AR(p), est un processus stochastique dans lequel chaque observation est exprimée comme une combinaison linéaire de ses propres valeurs passées et d'un bruit blanc.

le modèle AR(p) s'écrit :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t \tag{2.8}$$

où : ε_t est un bruit blanc et $\phi_1, \ldots, \phi_p$ sont les paramètres autorégressifs. De même que pour les processus moyennes mobile cette relation s'écrit :

$$X_t = \left(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p\right) X_t + \varepsilon_t \quad \text{soit} \quad \phi(B)X_t = \varepsilon_t$$

avec $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$.

Le processus AR(p) est stationnaire si les racines du polynôme $\phi(B)$ sont toutes à l'extérieur du cercle unité (i.e. de module strictement supérieur à 1) [15].

Exemple 2.8
Le modèle AR(2) est défini par :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t$$

2.4.5 Modèle ARMA(p, q)

Un modèle ARMA (*Autoregressive Moving Average*) est obtenu en combinant un modèle autorégressif d'ordre p (AR) et un modèle à moyenne mobile d'ordre q (MA). Ce type de modèle est utilisé pour modéliser une série temporelle stationnaire en capturant à la fois la dépendance de la variable avec ses propres valeurs passées et avec les chocs aléatoires passés.

Le modèle ARMA(p, q) s'écrit comme suit :

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \tag{2.9}$$

soit, avec les notations précédentes

$$\phi(B) X_t = \theta(B) \varepsilon_t$$

avec :

$$\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p \quad \text{et} \quad \theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$$

Exemple 2.9
Un modèle ARMA(2,3) s'écrit sous la forme :

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3},$$

2.4.6 Test de Dickey-Fuller (ADF)

Le test de Dickey-Fuller est un outil statistique permettant de vérifier si une série temporelle est stationnaire ou non. Il repose sur la détection d'une racine unitaire [4], ce qui indiquerait une non-stationnarité.

Le point de départ est un modèle autorégressif d'ordre 1 :

$$X_t = \phi X_{t-1} + \varepsilon_t,$$

où ε_t est un bruit blanc.

Si $\phi = 1$, la série suit une marche aléatoire, et donc elle n'est pas stationnaire. Pour tester cela, on reformule le modèle en soustrayant X_{t-1} de chaque côté :

$$\Delta X_t = \gamma X_{t-1} + \varepsilon_t, \quad \text{avec } \gamma = \phi - 1.$$

Le test consiste à vérifier si le paramètre γ est nul :
— $H_0 : \gamma = 0$ (la série a une racine unitaire, donc elle est non stationnaire),
— $H_1 : \gamma < 0$ (la série ne possède pas de racine unitaire, donc elle est stationnaire).

Version augmentée du test ADF. Dans la réalité, les valeurs d'une série peuvent être influencées non seulement par la valeur précédente X_{t-1}, mais aussi par les variations récentes. De plus, il peut exister une corrélation entre les résidus du modèle, ce qui fausse les résultats du test de base.

Pour corriger cela, on utilise la version dite *augmentée* du test ADF. Elle consiste à ajouter plusieurs retards des différences passées ΔX_{t-i} dans le modèle. Cela permet de mieux capter la dynamique de la série et de s'assurer que les résidus ne sont pas autocorrélés.

Le modèle devient alors :

$$\Delta X_t = \gamma X_{t-1} + \sum_{i=1}^{p} \alpha_i \Delta X_{t-i} + \varepsilon_t, \tag{2.10}$$

où : p est le nombre de retards ajoutés et γ est toujours le paramètre testé pour vérifier la stationnarité.

— Si la statistique de test est inférieure à la valeur critique, on rejette H_0 et on conclut que la série est stationnaire.
— Sinon, on ne rejette pas H_0 : la série est alors considérée comme non stationnaire.

2.4.7 Transformations des séries temporelles

Lorsque le test de Dickey-Fuller indique que la série temporelle est non stationnaire, il devient nécessaire de lui appliquer des transformations pour satisfaire les hypothèses des modèles ARIMA. Ces transformations visent généralement à supprimer la tendance, à réduire la variabilité, ou à éliminer la saisonnalité.

Les transformations les plus courantes sont les suivantes :

La différenciation (ou différentiation temporelle)

Elle consiste à remplacer chaque observation X_t par la différence entre deux observations consécutives. Elle est utilisée pour éliminer une tendance linéaire :

$$Y_t = \Delta X_t = X_t - X_{t-1} \tag{2.11}$$

Si la série reste non stationnaire après une première différenciation, on peut appliquer une différenciation d'ordre supérieur :

$$Y_t = \Delta^d X_t = (1 - B)^d X_t$$

où d est l'ordre de différenciation minimal permettant d'obtenir une série stationnaire.

La différenciation saisonnière

Lorsque la série présente un comportement périodique (saisonnalité), il est possible d'éliminer cet effet en utilisant la transformation suivante :

$$Y_t = X_t - X_{t-s} \tag{2.12}$$

où s est la période saisonnière (par exemple, $s = 12$ pour des données mensuelles ou $s = 7$ pour des données quotidiennes).

Une fois les transformations appropriées appliquées, la stationnarité de la série peut être vérifiée de nouveau à l'aide de tests statistiques (comme ADF).

2.4.8 Modèle ARIMA(p, d, q)

ARIMA est un acronyme qui signifie Auto-Regressive Integrated Moving Average [41], Le modèle ARIMA est une extension des modèles AR et MA appliquée aux séries temporelles non stationnaires. Il permet de modéliser une série en combinant trois composantes :

— **AR(p)** : une partie autorégressive d'ordre p ;
— **I(d)** (*Integrated*, d) : une différenciation d'ordre d pour rendre la série stationnaire ;
— **MA(q)** : une partie à moyenne mobile d'ordre q.

Ainsi, un modèle ARIMA(p, d, q) s'écrit, après d différenciations de la série Y_t, comme suit :

$$\phi(B)(1-B)^d Y_t = \theta(B)\varepsilon_t \tag{2.13}$$

où :
— $\phi(B)$ est un polynôme autorégressif d'ordre p : $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$,
— $\theta(B)$ est un polynôme de moyenne mobile d'ordre q : $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$,

Exemple 2.10
Supposons qu'on analyse la série Y_t représentant la consommation mensuelle de gaz d'une entreprise. Un test de stationnarité (par exemple Dickey-Fuller) indique que la série n'est pas stationnaire. Après une différenciation ($d = 1$), la série devient stationnaire.

On ajuste alors un modèle ARIMA$(1, 1, 1)$:

$$(1 - \phi_1 B)(1 - B)Y_t = (1 + \theta_1 B)\varepsilon_t$$

Ce qui revient à modéliser :

$$\Delta Y_t = \phi_1 \Delta Y_{t-1} + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

où $\Delta Y_t = Y_t - Y_{t-1}$.

2.4.9 Modèle SARIMA (Seasonal ARIMA)

Les séries temporelles présentent souvent des schémas saisonniers, c'est-à-dire des variations qui se répètent à intervalles réguliers (mois, trimestres, années). Le modèle SARIMA (*Seasonal AutoRegressive Integrated Moving Average*) est une extension du modèle ARIMA qui permet de capturer à la fois les dynamiques non saisonnières et saisonnières [15].

Il est noté généralement :

$$SARIMA(p, d, q)(P, D, Q)_s$$

où :
— (p, d, q) sont les paramètres non saisonniers du modèle ARIMA,
— (P, D, Q) sont les paramètres saisonniers,
— s est la période de saisonnalité (par exemple $s = 12$ pour des données mensuelles avec un cycle annuel).

L'équation générale du modèle est :

$$\Phi(B^s)\phi(B)(1-B)^d(1-B^s)^D Y_t = \Theta(B^s)\theta(B)\varepsilon_t \tag{2.14}$$

avec :
— $\phi(B)$ et $\theta(B)$: polynômes non saisonniers (AR et MA),
— $\Phi(B^s) = 1 - \Phi_1 B^s - \cdots - \Phi_P B^{Ps}$: polynôme AR saisonnier,
— $\Theta(B^s) = 1 + \Theta_1 B^s + \cdots + \Theta_Q B^{Qs}$: polynôme MA saisonnier,
— $(1-B)^d$: différenciation ordinaire,
— $(1-B^s)^D$: différenciation saisonnière
— ε_t : bruit blanc.

2.4.10 Identification du modèle : méthode de Box-Jenkins

L'identification du modèle constitue la première étape essentielle de la méthodologie de Box-Jenkins. Elle repose principalement sur l'analyse des corrélations internes d'une série temporelle afin de déterminer la structure adéquate du modèle $ARIMA(p, d, q)$ ou $SARIMA(p, d, q)(P, D, Q)_s$ à partir de ses propriétés statistiques.

Avant toute modélisation, il est nécessaire de s'assurer que la série est stationnaire [17]. Si la série ne l'est pas, on applique une ou plusieurs différenciations pour la rendre stationnaire.

Une fois la série rendue stationnaire, l'identification des ordres p (composante autorégressive) et q (composante moyenne mobile) s'effectue à l'aide de l'analyse des fonctions ACF et PACF.

Détermination de p et q

1. Choix de p pour un processus AR(p) : Si l'ACF diminue de façon exponentielle tandis que la PACF présente une coupure nette après un certain retard p, un modèle **AR(p)** est suggéré.

Exemple 2.11
la Figure 2.4 illustre l'ACF et la PACF d'un processus AR(1). On observe que l'ACF décroît progressivement tandis que la PACF présente un pic net au lag 1, ce qui confirme la nature autorégressive du modèle. Les **intervalles de confiance à 95 %** (en bleu), souvent générés automatiquement par les logiciels de statistiques, permettent d'identifier les retards significatifs.

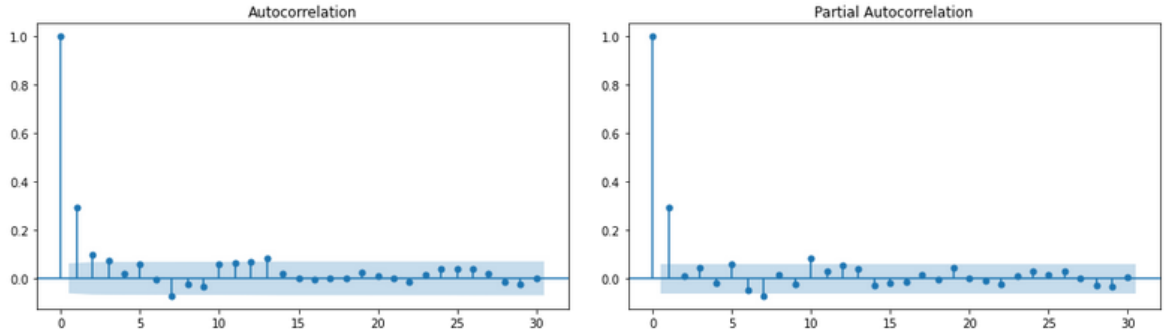

FIG. 2.4 : Interprétation des graphes ACF et PACF pour un processus AR(1)

2. Choix de q pour un processus MA(q) : Si la PACF décroît exponentiellement (ou de manière amortie), et que l'ACF présente une coupure nette après un certain retard q, cela suggère un modèle MA(q).

Exemple 2.12
la Figure 2.5 illustre un processus MA(2). L'ACF montre deux retards significatifs avant de s'annuler, tandis que la PACF décroît de manière continue, ce qui est caractéristique d'un modèle MA(2).

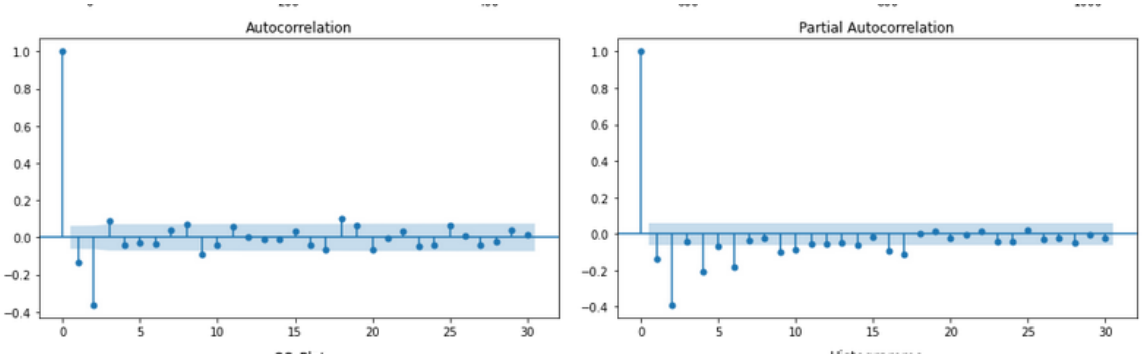

FIG. 2.5 : Interprétation des graphes ACF et PACF pour un processus MA(2)

Le tableau 2.1 présente une synthèse de l'interprétation des fonctions ACF et PACF selon le type de modèle. Les symboles ϕ et θ désignent respectivement les coefficients AR et MA, ρ_k est l'ACF au lag k, et ψ_{kr} est la PACF.

Modèle	**Autocorrélation (ACF)** ρ_k	**Autocorrélation partielle (PACF)** ψ_{kr}
AR(1)	Décroissance exponentielle (si $\phi_1 > 0$) ou oscillatoire (si $\phi_1 < 0$).	Pic significatif au lag 1, puis $\psi_{kr} = 0$ pour $k > 1$.
AR(2)	Décroissance exponentielle ou sinusoïdale amortie.	Pics significatifs aux deux premiers retards, $\psi_{kr} = 0$ pour $k > 2$.
MA(1)	Pic significatif au lag 1, puis $\rho_k = 0$ pour $k > 1$.	Décroissance exponentielle ou amortie.
AR(p)	Décroissance exponentielle ou amortie.	p pics significatifs, $\psi_{kr} = 0$ pour $k > p$.
MA(q)	q pics significatifs, $\rho_k = 0$ pour $k > q$.	Décroissance exponentielle ou amortie.
ARMA(p,q)	Décroissance mixte (effets AR et MA).	Décroissance mixte.
ARIMA(1,1,0)	Décroissance hyperbolique lente.	Pic significatif au premier retard.
ARIMA(1,0,1)	Décroissance amortie.	Décroissance amortie.
ARIMA(p,d,q)	Décroissance hyperbolique.	Décroissance hyperbolique.
SARIMA(P,D,Q,s)	Motifs saisonniers dans l'ACF.	Pics saisonniers dans la PACF.

TAB. 2.1 : Lecture des graphes ACF et PACF pour l'identification du modèle.

2.4.11 Estimation des paramètres

Une fois les paramètres du modèle estimés, on passe à l'étape d'estimation des coefficients du modèle en utilisant des algorithmes de calcul tels que l'estimation par maximum de vraisemblance [10].

Maximum de vraisemblance

Considérons un échantillon $X_1, ..., X_T$ supposé **iid** (indépendamment et identiquement distribués) suivant une loi de densité $f(x|\theta)$, où $\theta = (\theta_1, ..., \theta_k) \in \mathbb{R}^k$ est un vecteur de paramètres inconnus. La fonction de vraisemblance est alors définie comme :

$$L(\theta) = \prod_{t=1}^{T} f(X_t|\theta).$$

L'estimateur du maximum de vraisemblance (EMV) est la valeur $\hat{\theta}_{EMV}$ qui maximise cette fonction $L(\theta)$. Pour obtenir cet estimateur, on peut souvent résoudre le système d'équations obtenu en dérivant $L(\theta)$ par rapport à chaque paramètre θ_j et en annulant ces dérivées :

$$\frac{\partial L(\theta)}{\partial \theta_j} = 0, \quad \text{pour } j = 1, ..., k.$$

En pratique, il est plus courant de travailler avec le logarithme de la vraisemblance, appelé fonction de log-vraisemblance :

$$\ell(\theta) = \log L(\theta) = \sum_{t=1}^{T} \log f(X_t|\theta).$$

Maximiser $L(\theta)$ revient alors à maximiser $\ell(\theta)$, ce qui simplifie souvent les calculs puisque la dérivation du logarithme transforme le produit en somme.

Remarque 2.1. Dans la pratique, on utilise des logiciels comme R, Python (statsmodels), ou Gretl pour estimer ces paramètres automatiquement par la méthode du maximum de vraisemblance.

Il existe autre méthodes pour estimer comme la Méthode des moindres carrés inconditionnel (CLSM) ou Méthodes des estimations non linéaire.

Critères de choix des modèles

Il n'est pas toujours facile de déterminer un modèle unique qui s'ajuste au mieux à une série temporelle. En pratique, le choix repose souvent sur la comparaison de plusieurs modèles candidats à l'aide de **critères d'information**. Ces critères évaluent la qualité de l'ajustement tout en pénalisant les modèles trop complexes. Le modèle retenu est généralement celui qui **minimise** l'un de ces critères sur les T observations disponibles.

Principaux critères d'information :

1. **AIC (Critère d'information d'Akaike)** [1] :

$$AIC(p,q) = \log(\hat{\sigma}_\varepsilon^2) + \frac{2(p+q)}{T}$$

   où $\hat{\sigma}_\varepsilon^2$ est la variance des résidus.

2. **BIC (Critère bayésien de Schwarz)** [38] :

$$BIC(p,q) = \log(\hat{\sigma}_\varepsilon^2) + \frac{(p+q)\log T}{T}$$

3. **HQ (Critère de Hannan-Quinn)** [18] :

$$HQ(p,q) = \log(\hat{\sigma}_\varepsilon^2) + \frac{(p+q)\log(\log T)}{T}$$

l'AIC est souvent utilisé pour les prévisions à court terme, tandis que le BIC, plus strict, est préféré pour éviter les modèles trop complexes.

Remarque 2.2. **Le meilleur modèle est généralement celui qui présente la plus petite valeur du critère (AIC, BIC ou HQ).**

2.4.12 Validation du modèle

Une fois un modèle estimé, il est indispensable de vérifier la validité statistique du modèle. La validation repose sur plusieurs tests : la significativité des paramètres, l'analyse des résidus, et la détection d'hétéroscédasticité éventuelle.

Tests sur les paramètres

Après l'estimation du modèle, chaque paramètre $\hat{\theta}_i$ est soumis à un test de significativité. On teste l'hypothèse nulle $H_0 : \theta_i = 0$ contre l'alternative $H_1 : \theta_i \neq 0$ à l'aide de la statistique de test de Student du coefficient est :

$$t_i = \frac{\hat{\theta}_i}{\text{SE}(\hat{\theta}_i)}$$

qui suit une loi t de student à $(n - k)$ degrés de liberté.

où $\text{SE}(\hat{\theta}_i)$ est l'erreur standard de l'estimateur. La p-valeur associée permet de décider : si p-valeur $< 0{,}05$, le paramètre est significatif ; sinon, il est considéré comme non significatif et peut être retiré du modèle.

Tests sur les résidus

La validation du modèle repose principalement sur l'analyse des résidus, c'est-à-dire la différence entre les valeurs observées et les valeurs ajustées par le modèle.

Les résidus sont définis comme :

$$\hat{\varepsilon}_t = Y_t - \hat{Y}_t,$$

où $\hat{Y}_t$ est la valeur ajustée par le modèle. pour qu'un modèle soit valide, ses résidus doivent idéalement se comporter comme un bruit blanc.

Pour tester l'absence d'autocorrélation, on utilise la statistique de Ljung-Box (1978) :

$$Q = n(n+2) \sum_{k=1}^{h} \frac{\hat{\rho}_k^2}{n-k}$$

où n est la taille de l'échantillon, $\hat{\rho}_k$ est l'autocorrélation des résidus au lag k, et h est le nombre de retards testés. Sous l'hypothèse nulle, Q suit une loi χ^2 à $h - p - q$ degrés de liberté [**LjungBox1978**] Un p-valeur élevée (supérieure à 5%) suggère que les résidus sont bien un bruit blanc.

Tests d'homoscédasticité

Test de Jarque-Bera [23] : Ce test permet de vérifier la normalité des résidus. La statistique du test est :

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right)$$

où S est le coefficient d'asymétrie (skewness), et K la kurtose est le nombre de paramètres à estimés. Sous l'hypothèse nulle de normalité, JB suit une loi χ^2 à 2 degrés de liberté.

Test ARCH d'Engle (1982) : Pour détecter une hétéroscédasticité conditionnelle c'est-à-dire une dépendance dans la variance des erreurs, on estime la régression suivante :

$$\hat{\varepsilon}_t^2 = \alpha_0 + \alpha_1 \hat{\varepsilon}_{t-1}^2 + \cdots + \alpha_p \hat{\varepsilon}_{t-p}^2 + u_t$$

On teste $H_0 : \alpha_1 = \cdots = \alpha_p = 0$. La statistique nR^2 suit une loi χ^2 à p degrés de liberté. Un rejet de H_0 indique un effet ARCH significatif [13].

Ces tests permettent d'assurer que le modèle est adéquat pour la prévision. Si ces conditions ne sont pas remplies, il convient de revoir la spécification du modèle, par exemple en modifiant les ordres (p, d, q) ou en choisissant une autre classe de modèles.

2.4.13 Prédiction à partir du modèle validé

Une fois le modèle sélectionné et validé, il peut être utilisé pour effectuer des prévisions sur les périodes futures. Ces prévisions sont calculées à partir des paramètres estimés et des valeurs passées de la série. Il est également possible d'estimer l'incertitude associée à ces prévisions à l'aide d'intervalles de confiance.

La figure 2.6 illustre la méthodologie de Box-Jenkins appliquée aux séries temporelles

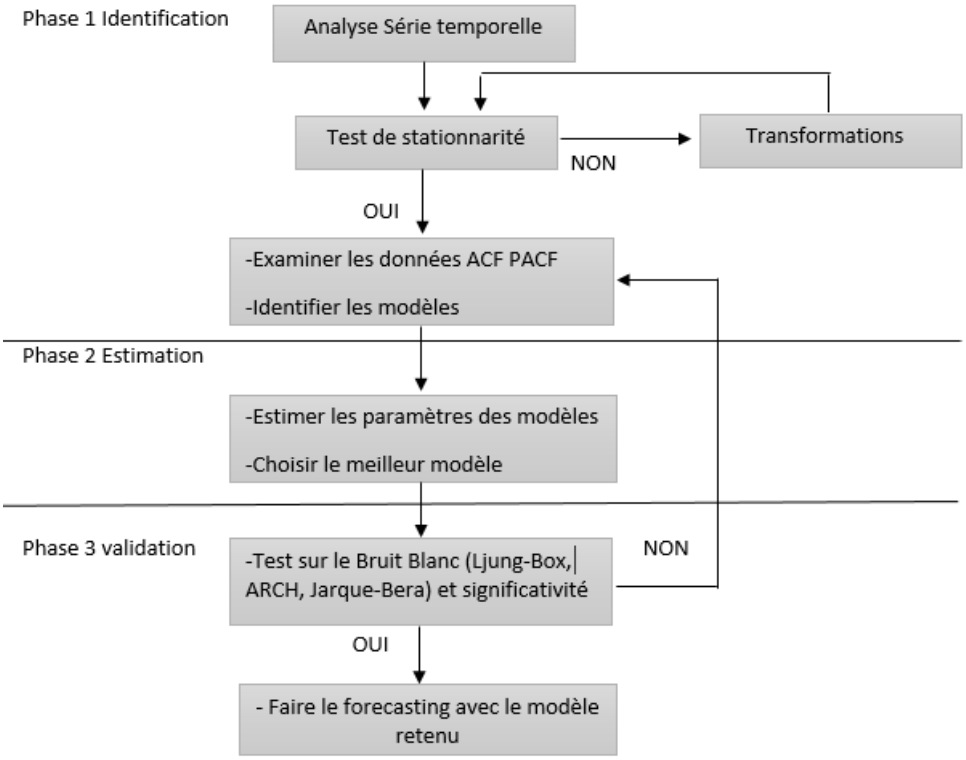

FIG. 2.6 : Etapes de la méthodologie Box-Jenkins

2.5 Conclusion

Les modèles ARIMA et SARIMA, basés sur la méthode de Box-Jenkins, ont plusieurs avantages. Ils permettent de bien représenter la structure d'une série temporelle en tenant compte de la tendance, de la saisonnalité et des dépendances entre les valeurs passées.

Cependant, ces modèles ont aussi des inconvénients. Ils supposent que la série est stationnaire, ce qui oblige souvent à transformer les données (différenciation, logarithme). De plus, il peut être difficile de bien choisir les bons paramètres du modèle, comme les ordres p, d et q.

Par rapport aux modèles de lissage, comme le lissage exponentiel ou le modèle de Holt-Winters, les modèles ARIMA sont plus puissants pour analyser les séries avec des dépendances complexes. Mais les modèles de lissage sont plus simples à utiliser et demandent moins de transformations sur les données. Ils sont donc souvent préférés pour des prévisions à court terme ou lorsque les données changent peu.

Chapitre 3

Méthodes d'apprentissage profond pour la prévision avec LSTM

3.1 Introduction

L'apprentissage automatique s'est imposé comme un outil central dans le traitement et l'analyse de données complexes. Parmi ses différentes branches, l'apprentissage profond (deep learning) se distingue par sa capacité à modéliser des relations non linéaires à partir de grandes quantités de données, notamment grâce aux réseaux de neurones artificiels.

Dans le cadre de la prévision de séries temporelles, les méthodes statistiques traditionnelles comme ARIMA ou les modèles de lissage ont montré leurs limites, en particulier face à des dynamiques complexes ou non stationnaires. Pour répondre à ces défis, des modèles issus de l'apprentissage profond, tels que les réseaux de neurones récurrents (RNN) et leur variante améliorée, les réseaux à mémoire longue courte (Long Short-Term Memory, LSTM), ont été développés.

Ce chapitre vise à introduire les fondements théoriques nécessaires à la compréhension des modèles LSTM. Nous commencerons par une présentation générale de l'apprentissage automatique, avant d'aborder l'apprentissage profond, les réseaux de neurones, les architectures récurrentes, puis les spécificités des modèles LSTM et leurs variantes.

3.2 Apprentissage automatique

L'apprentissage automatique (*Machine Learning* ou ML) est une branche de l'intelligence artificielle (IA) qui vise à développer des algorithmes capables d'apprendre à partir des données sans être explicitement programmés. Ces algorithmes identifient des motifs dans les données et utilisent ces connaissances pour effectuer des prédictions ou prendre des décisions.

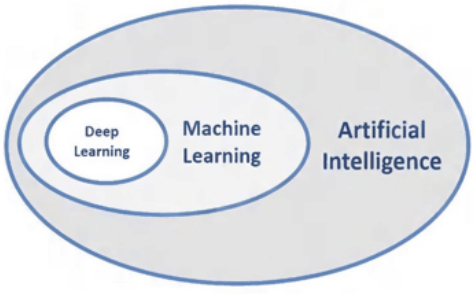

Fig. 3.1 : Relation entre IA, ML et DL [39].

La Figure 3.1 illustre la relation hiérarchique entre l'intelligence artificielle (IA), le machine learning (ML) et l'apprentissage profond (DL). L'IA constitue le domaine général, au sein duquel le ML est un sous-ensemble, et l'apprentissage profond un sous-ensemble du ML.

Les principales approches de l'apprentissage automatique

L'apprentissage automatique regroupe différentes méthodes permettant à un modèle d'apprendre à partir de données. On distingue généralement trois grandes approches :

- **Apprentissage supervisé** : Le modèle apprend à partir d'un ensemble de données pour lesquelles les réponses sont connues. Par exemple, dans le cas de la prévision de la consommation de gaz ou d'électricité, on dispose des consommations passées associées à certaines caractéristiques (mois, température, etc.). Le modèle utilise ces exemples pour apprendre à prédire la consommation future.
- **Apprentissage non supervisé** : Contrairement à l'apprentissage supervisé, ici les données ne sont pas accompagnées de réponses. Le modèle cherche à en découvrir la structure cachée, comme des regroupements ou des tendances. Par exemple, il peut identifier des groupes de clients ayant des comportements de consommation similaires, sans que ces groupes soient définis à l'avance.
- **Apprentissage par renforcement** : Cette approche repose sur l'interaction entre un agent (un programme ou un robot) et un environnement. L'agent prend des décisions, observe les résultats, et reçoit des récompenses ou des pénalités. Il apprend ainsi progressivement à adopter les actions les plus bénéfiques. Ce type d'apprentissage est utilisé, par exemple, dans les systèmes de contrôle intelligent pour optimiser la consommation d'énergie dans un bâtiment.

Ces trois approches sont utilisées dans beaucoup de domaines : reconnaissance d'images, voitures autonomes, traitement du langage, ou encore prévision de séries temporelles [2].

Classification des méthodes d'apprentissage automatique

La Figure 3.2 présente une classification générale des méthodes d'apprentissage automatique. On y distingue deux grandes familles : les méthodes de *machine learning* classique et les méthodes d'*apprentissage profond* (*deep learning*).

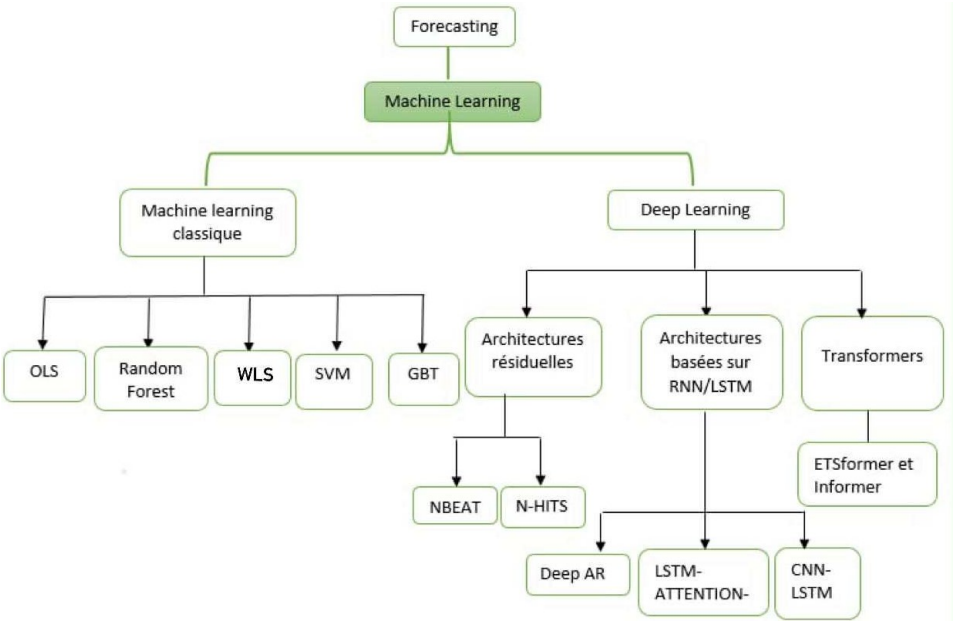

FIG. 3.2 : Classification des méthodes d'apprentissage automatique.

Remarque 3.1. Les méthodes classiques illustrées dans la figure comprennent :

- **OLS** : Régression linéaire par moindres carrés ordinaires.
- **WLS** : Régression par moindres carrés pondérés, utilisée lorsque les erreurs n'ont pas la même variance.
- **SVM** : Support Vector Machine, utilisé pour la classification et la régression.
- **GBT** : Gradient Boosted Trees, modèle performant basé sur un ensemble d'arbres de décision successifs.

Dans ce chapitre, nous nous concentrons principalement sur l'exploration des approches basées sur l'apprentissage profond, en particulier les LSTM.

3.3 Apprentissage profond

L'apprentissage profond (deep learning) est un domaine de l'intelligence artificielle qui repose sur l'utilisation de réseaux de neurones artificiels pour modéliser des relations complexes dans les données. Il est particulièrement puissant pour des tâches telles que la reconnaissance d'images, le traitement du langage naturel, et bien sûr, la prévision de séries temporelles [27].

3.3.1 Du neurone biologique au neurone artificiel

Les réseaux neuronaux artificiels s'inspirent largement du fonctionnement des neurones biologiques, qui sont les cellules responsables du traitement de l'information dans le cerveau [8].

En comprenant le fonctionnement des neurones biologiques, les chercheurs ont pu développer des modèles simplifiés, appelés neurones artificiels.

Le neurone biologique

Le neurone biologique est une cellule spécialisée dans la transmission de signaux électriques et chimiques dans le système nerveux.

Voici la figure 3.3 d'illustration d'un neurone biologique [42] :

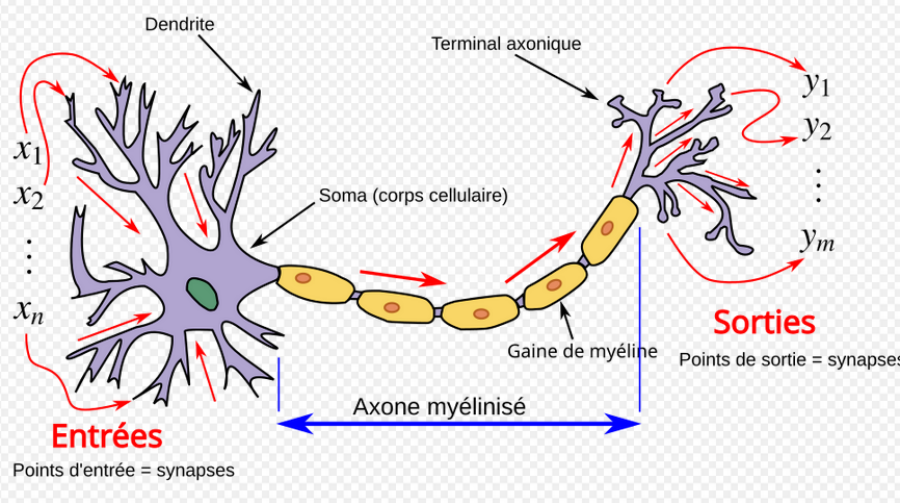

FIG. 3.3 : représentation d'un neurone biologique .

Chaque neurone se compose de plusieurs parties essentielles :

Le soma (ou corps cellulaire) qui contient le noyau de la cellule, **Les dendrites** qui reçoivent les signaux provenant d'autres neurones, **L'axone** qui transmet les signaux vers d'autres

neurones ou muscles, **La synapse**, qui est la jonction entre deux neurones où se produit la transmission du signal.

Le neurone fonctionne de manière électrochimique : lorsqu'un neurone reçoit un signal de ses dendrites, celui-ci se propage jusqu'à l'axone. Si l'intensité du signal dépasse un seuil donné, le neurone génère un potentiel d'action qui est transmis à d'autres neurones via les synapses.

Le neurone artificiel

Inspiré du neurone biologique, le neurone artificiel, bien qu'il soit une simplification, cherche à imiter certaines de ses fonctions de base. Un neurone artificiel reçoit des entrées, applique des poids à ces entrées, les additionne, puis applique une fonction d'activation pour produire une sortie.

Voici la fiqure 3.4 d'illustration d'un neurone artificiel [39] :

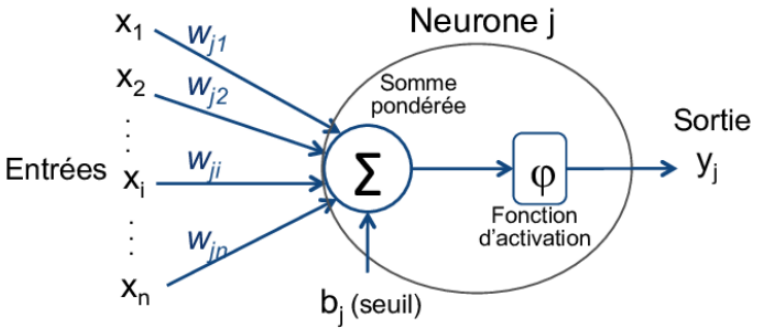

FIG. 3.4 : La structure d'un neurone artificiel.

Dans sa forme la plus simple, un neurone artificiel calcule la somme pondérée de ses entrées, y ajoute un biais, puis applique une fonction d'activation, généralement non linéaire, à cette somme. La valeur obtenue représente la sortie finale du neurone, comme l'illustre l'équation suivante :

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.1}$$

- x_i : Les valeurs d'entrée du neurone (x_1 à x_n)
- w_i : Les poids est le coefficient qui contrôle le signal d'entrée (la force de connexion), en d'autres termes le poids décide l'influence de l'entrée sur la sortie.
- $\sum$: La somme pondérée des entrées.
- b : Le terme de biais dans un neurone est un nombre ajouté à la somme des entrées pour aider le neurone à mieux ajuster sa sortie. Il permet au modèle de mieux s'adapter aux données, même si les entrées ne sont pas idéales. C'est comme un ajustement supplémentaire qui rend le modèle plus flexible.
- f : La fonction d'activation (non linéaire en général).
- y : La sortie du neurone.

Fonctionnement :

Cette opération peut se décomposer en deux étapes :

1. Calcul du potentiel d'activation :

$$z = \sum_{i=1}^{n} w_i x_i + b$$

2. Application de la fonction d'activation :

$$y = f(z)$$

La fonction d'activation est une fonction mathématique appliquée après la somme des entrées et l'ajout du biais. Elle détermine la sortie du neurone et influence la performance du réseau . Le choix de la fonction d'activation est crucial pour le bon apprentissage du réseau.dans la fiqure 3.5 on montre quelques fonctions d'activation avec leurs courbes [33].

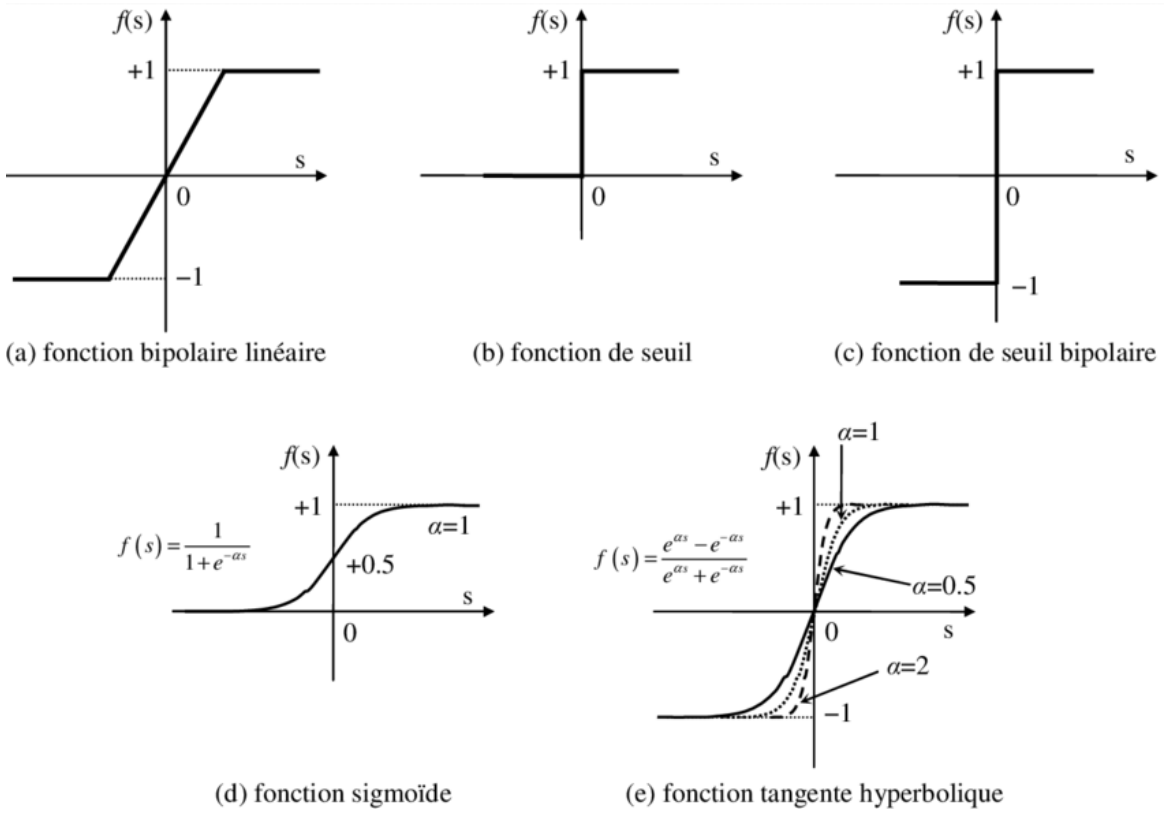

FIG. 3.5 : Fonctions d'activation d'un neurone artificiel .

Exemple 3.1
Imagine un neurone artificiel reçoit des entrées $x_1, x_2, \ldots, x_n$, qui sont multipliées par des poids $w_1, w_2, \ldots, w_n$ respectifs, et un biais b est ajouté pour obtenir une valeur agrégée :

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n + b$$

Cette valeur y, représentant le potentiel d'activation du neurone, est ensuite introduite dans une fonction d'activation. Dans le cas de la **fonction de seuil**, la sortie du neurone est déterminée par :

$$f(y) = \begin{cases} 1 & \text{si } y \geq 0 \\ 0 & \text{si } y < 0 \end{cases}$$

Ainsi, le neurone s'active et produit une sortie de 1 lorsque le signal total y atteint ou dépasse un certain seuil (ici fixé à zéro). Dans le cas contraire, il reste inactif, produisant une sortie de 0. Ce comportement de type **binaire** est analogique à un interrupteur, où la sortie du neurone est activée ou désactivée en fonction de l'intensité du signal d'entrée.

Exemple 3.2
Dans le cas de la **fonction sigmoïde**, la sortie du neurone est comprise entre 0 et 1. On

considère généralement que le neurone est **activé** si la sortie est supérieure à 0,5, et **inactif** sinon.

Par exemple, si la valeur d'entrée du neurone donne une sortie sigmoïde supérieure à 0,5, alors le neurone est activé. Dans le cas contraire, la sortie est trop faible pour entraîner une activation.

La fonction sigmoïde est définie par la formule suivante :

$$f(y) = \frac{1}{1 + e^{-y}}$$

3.3.2 Réseaux de neurones artificiels

Le premier modèle de neurone artificiel a été proposé par Frank Rosenblatt en 1958 sous la forme du *Perceptron* [35]. vient aprés la decouverte de neurone artificiels par Walter Pitts et Warren McCulloche en 1943. Il s'agit d'un modèle linéaire capable de classer des données en deux catégories. Ce modèle a marqué le début de l'étude des réseaux neuronaux artificiels.

Cependant, le Perceptron présente une limitation majeure : il est incapable de résoudre certains problèmes non linéaires simples. Cette limitation a été démontrée par Minsky et Papert en 1969 dans leur ouvrage [29], ce qui a entraîné une perte d'intérêt pour ces modèles pendant plusieurs années.

C'est à partir des années 1980 que l'étude des réseaux multicouches a redémarré, notamment grâce au développement du Perceptron multicouche (*Multilayer Perceptron* ou MLP). Cette avancée a été rendue possible par l'introduction de l'**algorithme de rétropropagation du gradient** (*backpropagation*), proposée notamment par Rumelhart, Hinton et Williams [36]. Cet algorithme constitue une avancée fondamentale, car il permet d'entraîner efficacement des architectures profondes en ajustant les poids du réseau en fonction de l'erreur mesurée à la sortie.

Architecture d'un réseau de neurones artificiel.

Un réseau de neurones artificiels (RNA) est structuré en couches : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie voir la fiqure 3.6. chaque couche joue un rôle bien précis dans le traitement des données :

Couche d'entrée (Input layer) : C'est la première couche du réseau. Elle reçoit les données initiales. Chaque neurone de cette couche correspond à une variable d'entrée. Par exemple, si l'on cherche à prédire la consommation de électrique, chaque neurone peut représenter la consommation à un moment donné.

Couches cachées (Hidden layers) : Ce sont les couches situées entre l'entrée et la sortie. Elles réalisent des calculs pour détecter des motifs ou des relations dans les données. Grâce à ces couches, le réseau peut apprendre des informations complexes.

Couche de sortie (Output layer) : C'est la dernière couche du réseau. Elle fournit le résultat final, comme une prédiction. Par exemple, dans un modèle de prévision, elle pourrait donner la consommation de électrique estimée pour une date future.

Chaque neurone est relié aux neurones des couches voisines par des connexions avec des poids. Pendant l'apprentissage, le réseau ajuste ces poids pour améliorer ses résultats.

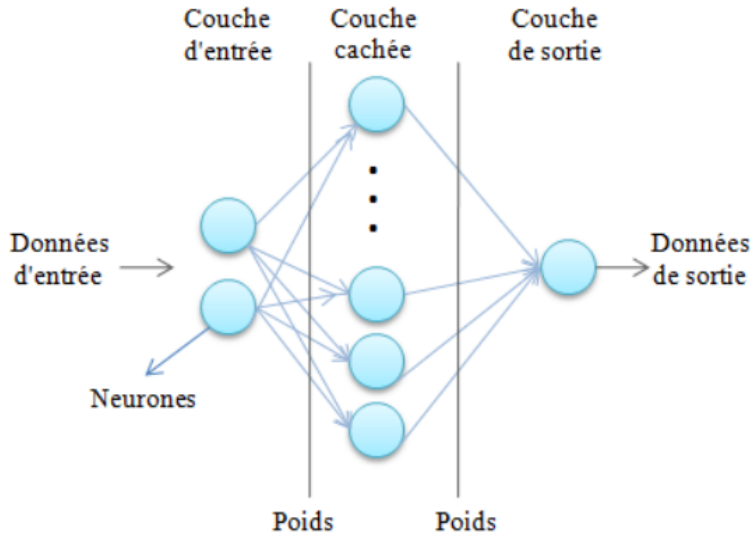

FIG. 3.6 : Structure de modèle de réseau de neurones artificiels [39].

3.3.3 Rétropropagation du gradient

L'algorithme de rétropropagation du gradient est au cœur de l'apprentissage supervisé des réseaux de neurones. Il permet d'optimiser les poids et les biais du réseau en minimisant une fonction de coût à l'aide de une méthode d'optimisation (comme la descente de gradient).

Voici les Comme le décrivent Goodfellow et al. dans leur ouvrage de référence sur l'apprentissage profond [14], l'entraînement d'un réseau repose sur les étapes suivantes :

1. **Initialisation :** les poids w et les biais b du réseau sont initialisés, en général de manière aléatoire.
2. **Propagation avant (forward propagation)** : Les données d'entrée traversent les couches du réseau, où chaque neurone effectue un calcul de la forme :

$$z = w \cdot x + b \quad \text{puis} \quad \hat{y} = f(z)$$

où x est l'entrée, f est la fonction d'activation (par exemple ReLU ou sigmoïde), z la somme pondérée, et $\hat{y}$ la sortie prédite.

3. **Calcul de la fonction de coût :** on mesure l'écart entre la prédiction $\hat{y}$ et la valeur réelle y à l'aide d'une fonction de perte (ou du coût), qui dépend du type de problème :
   — Dans les problèmes de régression, on utilise souvent l'erreur quadratique moyenne (MSE) :

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   — Pour la classification binaire, la perte d'entropie croisée (log loss) est utilisée :

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

4. **Rétropropagation du gradient :** on calcule les gradients de la fonction de coût par rapport aux poids du réseau, en appliquant la règle de la chaîne. Cela permet d'obtenir :

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

5. **Mise à jour des paramètres :** les poids et les biais sont ajustés en suivant la direction du gradient négatif avec un taux d'apprentissage η :

$$w \leftarrow w - \eta \cdot \frac{\partial \mathcal{L}}{\partial w}$$

6. **Itération sur plusieurs époques :** ce processus est répété sur plusieurs itérations (ou *époques*) à travers les données d'entraînement jusqu'à convergence du modèle.

3.3.4 Limites des réseaux de neurones classiques

Les réseaux de neurones classiques (Perceptron Multicouche (MLP)), dits *feedforward*, sont efficaces pour modéliser des relations complexes entre des variables en entrée et en sortie. Cependant, ils présentent plusieurs limites lorsqu'il s'agit de traiter des données séquentielles ou temporelles :

— **Absence de mémoire** : ces réseaux ne conservent aucune information sur les données précédentes. Chaque entrée est traitée de manière indépendante, sans tenir compte du contexte passé. cela limite leur performance sur des tâches comme la prédiction de texte ou l'analyse de séries temporelles.
— **Taille d'entrée fixe** : les réseaux classiques nécessitent des vecteurs d'entrée de taille constante. cela rend difficile le traitement de séquences de longueurs variables, comme des phrases de différentes tailles ou des séries temporelles incomplètes.
— **Incapacité à modéliser les dépendances temporelles** : Dans les séries temporelles (par exemple la consommation de gaz, le climat, etc.), une observation dépend souvent des valeurs passées. Les réseaux classiques n'ont aucun mécanisme pour capturer cette dynamique.Ils voient chaque observation comme indépendante, ce qui peut fortement réduire la qualité des prévisions.

Ces limitations ont conduit au développement de nouvelles architectures, telles que les réseaux convolutifs (CNN) pour les données spatiales et les réseaux récurrents (RNN) pour les données temporelles. Ces variantes permettent d'étendre les capacités des réseaux de neurones classiques à des contextes plus variés et complexes.

3.4 Types de réseaux de neurones

Il existe plusieurs types de réseaux de neurones adaptés à des tâches spécifiques. Les plus utilisés sont :

— **Les réseaux de neurones récurrents (RNN)** : conçus pour traiter des données séquentielles (texte, séries temporelles, etc.). Ils sont capables de conserver une mémoire des états précédents. Une variante importante est le LSTM (Long Short-Term Memory).
— **Les réseaux de neurones convolutifs (CNN)** :Spécialement conçus pour le traitement d'images, les CNN sont capables d'extraire automatiquement des caractéristiques spatiales importantes grâce à l'utilisation de filtres et de convolutions [28].
— **Transformer** : utilisés pour Traduction, résumé de texte, modèles de langage (comme GPT).
— **les réseaux de neurones résiduels (ResNet)** :utilisés principalement dans des tâches complexes où il est nécessaire d'avoir un réseau très profond, c'est-à-dire avec beaucoup de couches, sans perdre en performance.

Dans ce mémoire, nous mettrons l'accent sur les RNN et leurs variantes, en particulier les LSTM, que nous présentons dans la section suivante.

3.5 Réseaux de neurones récurrents (RNN)

Les premiers travaux significatifs sur Les réseaux de neurones récurrents (RNN, pour Recurrent Neural Networks) remontent aux années 1980, et notamment aux recherches de Elman [12]. Contrairement aux réseaux de neurones classiques, les RNN sont capables de conserver une mémoire interne qui leur permet de modéliser les dépendances temporelles dans les séries temporelles ou d'autres types de données séquentielles.

Principe de fonctionnement :

Un réseau de neurones récurrent repose sur une structure où les connexions entre les neurones incluent des boucles récurrentes. Ainsi, à chaque étape temporelle, l'état caché du réseau dépend non seulement de l'entrée actuelle, mais aussi de l'état précédent. Cela permet au réseau de conserver une mémoire de court terme sur les observations passées(chapitre 10 dans [14]).

À chaque pas de temps t, le réseau reçoit une entrée x_t et met à jour son état caché h_t en fonction de cette entrée et de l'état caché précédent h_{t-1}. Cette mise à jour est réalisée selon la formule suivante :

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.2}$$

où W_{xh} représente la matrice des poids entre l'entrée et l'état caché, W_{hh} est la matrice des poids reliant l'état caché précédent, b_h est un vecteur biais, et f est une fonction d'activation non linéaire (comme la tangente hyperbolique ou ReLU).

L'état caché h_t agit comme une mémoire courte du réseau. Il contient des informations sur les entrées passées et est essentiel pour permettre au modèle de capturer les dépendances temporelles. À partir de cet état caché, le réseau génère une sortie y_t définie par :

$$y_t = W_{hy}h_t + b_y \tag{3.3}$$

où W_{hy} est la matrice des poids reliant l'état caché à la sortie, et b_y est le biais de sortie. Ainsi, à chaque instant t, la sortie y_t dépend à la fois de l'entrée actuelle x_t et des informations des étapes précédentes mémorisées dans h_t.

Comme illustré dans la figure 3.7, un réseau de neurones récurrent traite séquentiellement les données temporelles en utilisant des connexions cycliques qui permettent de prendre en compte l'état précédent du réseau .

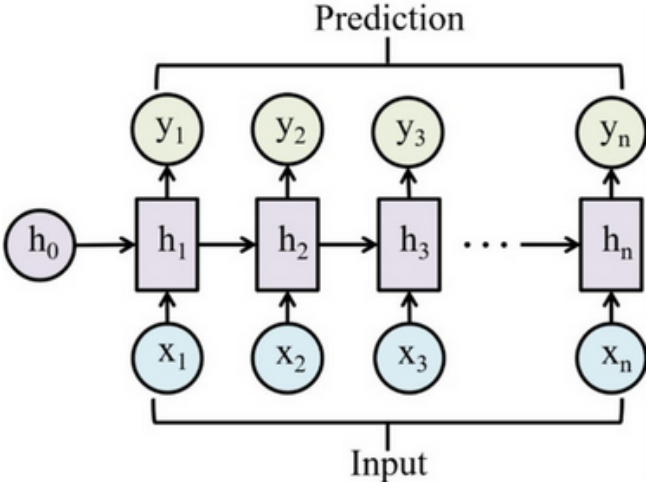

Fig. 3.7 : Architecture RNN de base [14].

Limites des RNN classiques. Malgré leur utilité, les RNN souffrent de problèmes lors de l'apprentissage de longues séquences, notamment à cause de l'extinction ou l'explosion du gradient lors de la rétropropagation dans le temps (*Backpropagation Through Time*, BPTT). Cela rend difficile l'apprentissage de dépendances à long terme [14].

Pour pallier ces limitations, des architectures améliorées ont été développées, notamment :

— Les réseaux **LSTM** (*Long Short-Term Memory*) [19] ;

— Les réseaux **GRU** (*Gated Recurrent Unit*).

Ces variantes intègrent des mécanismes de contrôle permettant de conserver ou d'oublier sélectivement l'information, ce qui améliore la performance sur les séquences longues.

3.6 Réseaux de neurones LSTM (Long Short-Term Memory)

Les réseaux de neurones à mémoire long terme (*Long Short-Term Memory*, LSTM), introduits par Hochreiter et Schmidhuber en 1997 [19], sont une extension des réseaux de neurones récurrents (RNN) conçue pour résoudre les problèmes liés à la disparition et à l'explosion du gradient. Ils intègrent une architecture composée de cellules mémoire capables de conserver l'information sur de longues périodes grâce à un mécanisme de portes : la porte d'oubli, la porte d'entrée, et la porte de sortie. Ces portes régulent respectivement l'oubli de l'ancienne information, l'ajout de nouvelle information, et la production de la sortie.

Architecture d'un bloc LSTM : Un bloc LSTM repose sur une cellule de mémoire C_t qui stocke l'information est régulée par trois portes principales :

1. **La porte d'oubli** f_t **:** qui détermine quelles informations de l'état précédent C_{t-1} doivent être oubliées.
2. **La porte d'entrée** i_t **:** qui décide quelles nouvelles informations sont ajoutées à la cellule.
3. **La porte de sortie** o_t **:** qui décide quelles parties de la mémoire doivent être utilisées pour générer la sortie cachée h_t.

Le fonctionnement interne d'un bloc LSTM est régi par les équations suivantes :

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

où : x_t est l'entrée à l'instant t, h_{t-1} est la sortie cachée précédente, W et b représentent les poids et biais appris par le réseau , σ est la fonction sigmoïde et tanh est la tangente hyperbolique , $(*)$ désigne le produit élément par élément (produit de Hadamard). La figure 3.8 illustre de manière détaillée l'architecture interne d'un LSTM [34], en montrant comment les données circulent entre les différentes composantes, ainsi que les équations mathématiques associées à chaque étape du traitement.

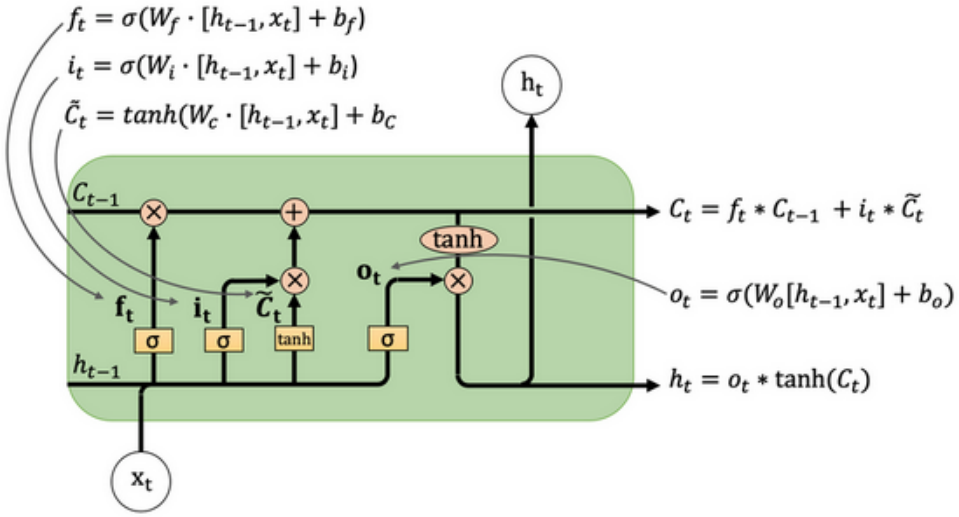

FIG. 3.8 : Architecture complète d'un LSTM avec les équations internes .

3.7 Variantes des LSTM

Bien que les réseaux LSTM classiques aient démontré une grande efficacité pour la modélisation des séries temporelles, plusieurs variantes ont été développées afin d'améliorer leurs performances, d'augmenter leur capacité à capturer des dépendances complexes, ou d'intégrer d'autres types de données. Nous présentons ici quelques variantes pertinentes pour les séries temporelles : DeepAR, LSTM avec attention et LSTM-CNN.

3.7.1 LSTM avec attention (LSTM-Attention)

Le modèle LSTM-Attention est une extension du LSTM classique, conçu pour améliorer la prise en compte des dépendances temporelles dans les séries longues ou complexes. Dans le cadre de la prévision de la consommation de électrique , certains jours ont plus d'influence que d'autres (ex. : journée chaude, jours fériés). Le mécanisme d'attention permet au modèle de se concentrer sur ces moments importants.

Le fonctionnement est le suivant [24] et [31] :

— Le LSTM génère une séquence d'états cachés $h_1, h_2, \ldots, h_T$.

— Un score d'importance u_t est calculé pour chaque état :

$$u_t = v^\top \tanh(W_h h_t + b_h)$$

— Ces scores sont transformés en poids d'attention par une normalisation softmax :

$$\alpha_t = \frac{\exp(u_t)}{\sum_{k=1}^{T} \exp(u_k)}$$

— Le vecteur de contexte V est obtenu comme une moyenne pondérée des h_t :

$$V = \sum_{t=1}^{T} \alpha_t h_t$$

Ce vecteur V résume les informations importantes de la séquence et est ensuite utilisé pour prédire la valeur cible. Grâce à ce mécanisme, le modèle apprend à se concentrer sur les moments passés qui influencent le plus la prévision.

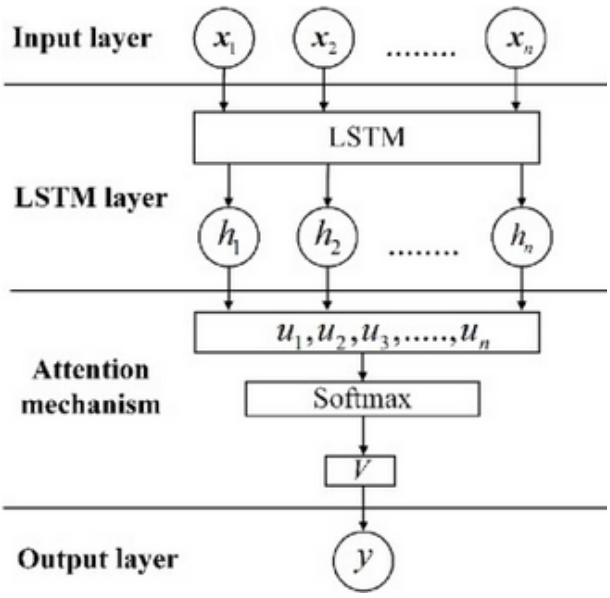

FIG. 3.9 : Schéma simplifié d'un LSTM avec mécanisme d'attention [24].

Ce type de réseau a été introduit initialement dans le contexte de la traduction automatique [3], mais il est désormais largement utilisé pour les séries temporelles.

3.7.2 DeepAR

DeepAR est un modèle développé par Amazon qui repose aussi sur les réseaux LSTM [37]. Sa particularité est qu'il peut apprendre à prédire plusieurs séries temporelles en même temps. Cela est très utile dans un contexte comme celui de la consommation électrique, où l'on dispose de plusieurs séries similaires (par exemple, une série pour chaque région ou chaque client).

Principe de fonctionnement

DeepAR utilise un LSTM qui, à chaque instant t, reçoit :

— la valeur précédente z_{t-1} (ex. : la consommation du jour précédent),

— des variables supplémentaires x_t (par exemple : jour de la semaine, température),

— l'état interne précédent du réseau.

Contrairement aux modèles classiques qui prédisent une seule valeur, DeepAR produit une **distribution de probabilité** pour la valeur future. Cela permet d'obtenir à la fois une prédiction et une estimation de l'incertitude.

Le but est que le modèle apprenne à deviner les valeurs futures en se basant sur les anciennes. Il s'entraîne pour que ses prédictions soient les plus proches possibles des vraies valeurs.

Cela revient à maximiser la probabilité suivante :

$$\log p(z_{t_0:T} \mid z_{1:t_0-1}, x_{1:T}) = \sum_{t=t_0}^{T} \log p(z_t \mid z_{1:t-1}, x_{1:T}) \tag{3.4}$$

Ou : x_t représente les variables explicatives disponibles (par exemple, la température ou le jour de la semaine).

La valeur future z_t est modélisée comme une variable aléatoire suivant une loi normale :

$$z_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$$

où μ_t et σ_t sont générés par le LSTM, représentant respectivement la moyenne et l'écart-type de la distribution prédite.

La figure 3.10 illustre le processus d'entraînement du modèle DeepAR, basé sur une cellule LSTM utilisée de manière autoregressive. Ce schéma est tiré d'une analyse appliquée à la pandémie de COVID-19 en Inde, présentée dans [40].

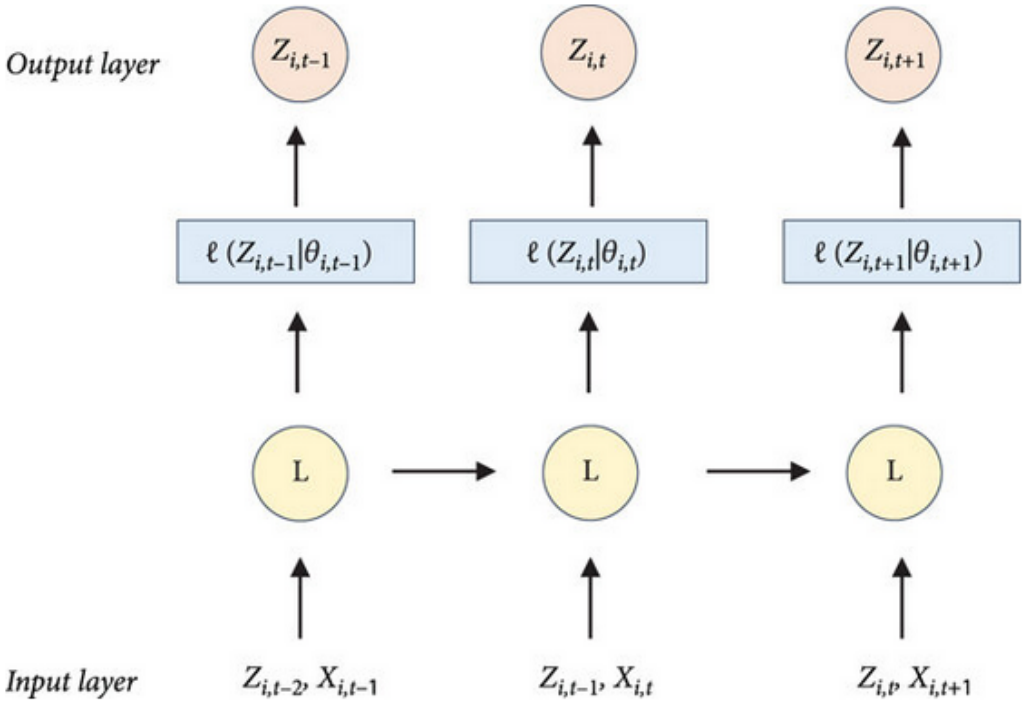

FIG. 3.10 : Schéma du processus d'entraînement du modèle DeepAR.

DeepAR présente plusieurs avantages importants : il donne des prévisions avec des intervalles de confiance, il apprend à partir de plusieurs séries similaires pour mieux généraliser, et il peut facilement intégrer des variables explicatives. pour plus de détails techniques, voir [37].

3.7.3 LSTM-CNN (LSTM combiné avec des réseaux convolutifs)

Le modèle LSTM-CNN (aussi appelé *CNN-LSTM*) est une architecture hybride qui combine deux types puissants de réseaux de neurones : les **réseaux convolutifs (CNN)** et les **réseaux à mémoire long terme (LSTM)**. Cette combinaison permet de tirer parti à la fois de la capacité des CNN à extraire des motifs locaux dans les séries temporelles, et de la capacité des LSTM à capturer des dépendances à long terme.

Dans ce modèle, les données temporelles sont d'abord traitées par une ou plusieurs couches convolutives (1D), qui détectent automatiquement des caractéristiques locales importantes (par exemple des pics de consommation). Les représentations extraites sont ensuite transmises aux couches LSTM, qui modélisent les relations séquentielles dans le temps . la figure 3.11 représente l'architecture de base du réseau CNN-LSTM [16].

Ce type d'architecture est particulièrement utile dans des contextes où les séries temporelles présentent à la fois des *variations locales rapides* et des *tendances globales*.

Plusieurs travaux ont montré l'efficacité de cette approche. Par exemple dans [30], cette architecture a été utilisée pour prédire l'évolution des cas de COVID-19 à partir de données temporelles. Les résultats ont montré une amélioration significative par rapport aux modèles LSTM seuls, grâce à la combinaison des deux mécanismes d'apprentissage .

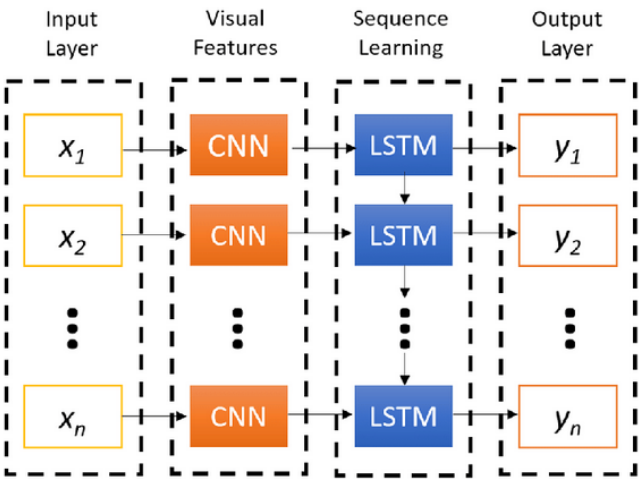

FIG. 3.11 : L'architecture de base du réseau CNN-LSTM.

En plus des variantes explorées, telles que LSTM avec attention, CNN-LSTM et DeepAR, il existe d'autres adaptations des LSTM, comme les LSTM bidirectionnels (BiLSTM), les LSTM empilés (Stacked LSTM) ou encore les Peephole LSTM.

3.8 Processus d'apprentissage des modèles LSTM et de leurs variantes

Les modèles LSTM (Long Short-Term Memory) et leurs variantes sont conçus pour capturer les dépendances temporelles dans les séries chronologiques. Le processus d'apprentissage de ces modèles repose sur plusieurs étapes clés que nous décrivons ci-après.

3.8.1 Prétraitement des données

Le prétraitement permet d'adapter les données au format requis par les modèles LSTM.

1. Normalisation des données

Les valeurs brutes peuvent avoir des ordres de grandeur très différents. Pour éviter que certaines variables ne dominent l'apprentissage, on les normalise :

$$x_{\text{norm}}^{(i)} = \frac{x^{(i)} - \mu}{\sigma} \tag{3.5}$$

où :

— $\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$ est la moyenne,

— $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu)^2}$ est l'écart-type.

— N = la taille des données

Cette opération donne à la série une moyenne nulle et un écart-type de 1.

Une autre méthode est la normalisation min-max :

$$x_{\text{min-max}}^{(i)} = \frac{x^{(i)} - x_{\min}}{x_{\max} - x_{\min}} \tag{3.6}$$

Celle-ci ramène les valeurs dans l'intervalle $[0, 1]$.

2. Création des séquences (fenêtres temporelles)

Les LSTM ne peuvent pas apprendre directement sur une série entière. On découpe la série en séquences de taille fixe W, appelées fenêtres :

$$X_t = [x_{t-W+1}, \ldots, x_t], \quad y_t = x_{t+1} \tag{3.7}$$

Chaque séquence X_t permet au modèle de prédire la valeur suivante y_t.

3. Séparation chronologique des données

On divise les données en trois groupes :
— **Ensemble d'entraînement** : pour apprendre les modèles.
— **Ensemble de validation** : pour ajuster les hyperparamètres et éviter le surapprentissage.
— **Ensemble de test** : pour évaluer les performances finales.

3.8.2 Conception de l'architecture du modèle

La conception de l'architecture d'un modèle LSTM consiste à décider comment organiser les différentes parties du réseau pour qu'il puisse bien apprendre à partir des données temporelles. En général, on utilise une ou plusieurs couches LSTM pour mémoriser les informations importantes dans le temps, puis une couche finale pour faire la prédiction. Selon le problème à résoudre, on peut aussi ajouter d'autres éléments comme l'attention (pour aider le modèle à se concentrer sur les moments importants) ou des couches convolutionnelles (pour détecter des motifs dans les données). L'objectif est de construire un modèle adapté aux données et capable de faire de bonnes prévisions.

3.8.3 Réglage des hyperparamètres

Les hyperparamètres sont les paramètres que l'on fixe avant l'entraînement. Ils influencent directement la qualité de l'apprentissage. Les plus importants sont :
— **Taille des séquences** : combien de valeurs passées le modèle regarde pour faire une prédiction.
— **Nombre de couches et de neurones** : plus on en met, plus le modèle est puissant, mais cela augmente aussi le risque de surapprentissage.
— **Taux de dropout** : utilisé pour éviter que le modèle apprenne trop précisément les données d'entraînement.
— **Taux d'apprentissage**.
— **Nombre d'époques** et **taille du batch** : combien de fois on passe sur toutes les données et combien de données sont utilisées en même temps pendant l'apprentissage.

Ces paramètres peuvent être choisis à la main ou testés automatiquement (par exemple avec une recherche par grille).

3.8.4 Entraînement du modèle

Après avoir choisi les bons hyperparamètres, on peut passer à l'entraînement du modèle LSTM. Cette étape a pour but d'apprendre à prédire correctement les valeurs futures à partir des données passées. Voici les principales étapes :

1-Création du modèle

On construit le modèle LSTM avec l'architecture définie (nombre de couches, nombre de neurones, etc.). Les poids sont initialisés automatiquement.

2-Apprentissage sur les données
Le modèle reçoit des séquences d'entrée et essaie de prédire la valeur suivante. Après chaque prédiction, il calcule une erreur et ajuste ses poids pour mieux prédire la prochaine fois. Cette technique s'appelle la *rétropropagation à travers le temps* (BPTT).

3-Utilisation de l'optimiseur Adam
Pour ajuster les poids, on utilise un algorithme appelé *Adam* [25], très efficace pour les séries temporelles. Il permet un apprentissage plus rapide et plus stable.

4-Évaluation sur les données de validation
Après chaque passage sur l'ensemble d'entraînement (appelé une *époque*), on teste le modèle sur des données de validation. Cela permet de vérifier que le modèle ne s'adapte pas trop uniquement aux données d'apprentissage.

5-Arrêt automatique si nécessaire
Si la performance sur les données de validation diminue, on peut arrêter l'entraînement automatiquement. Ce mécanisme s'appelle *early stopping*.

6-Sauvegarde du meilleur modèle
Le modèle qui obtient les meilleurs résultats pendant l'entraînement est sauvegardé. Il servira ensuite pour faire des prédictions sur de nouvelles données.

Grâce à ce processus, le modèle LSTM apprend à bien capter les relations temporelles présentes dans les séries chronologiques.

3.8.5 Prédiction et évaluation

Après l'entraînement, le modèle est utilisé pour faire des prévisions sur de nouvelles données. Cette étape permet de vérifier si le modèle fonctionne bien.

1. Faire des prédictions On utilise le modèle entraîné pour prédire les prochaines valeurs d'une série temporelle. Ces prédictions sont faites à partir de données qu'il n'a jamais vues (appelées données de test).

2. Comparer avec les vraies valeurs Une fois que le modèle donne des prédictions, on les compare avec les vraies valeurs réelles pour voir si elles sont proches. Plus elles sont proches, meilleur est le modèle.

3. Mesurer la performance du modèle Pour savoir si les prédictions sont bonnes, on compare les valeurs prédites $\hat{y}_t$ aux valeurs réelles y_t, à l'aide d'indicateurs d'erreur standards [21] :

— **RMSE (erreur quadratique moyenne)** :

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{3.8}$$

Plus la RMSE est petite, mieux c'est.

— **MAE (erreur absolue moyenne)** :

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{3.9}$$

C'est l'erreur moyenne entre les vraies valeurs et les valeurs prédites.

— **MAPE (erreur pourcentage absolue moyenne)** :

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{3.10}$$

Elle donne une idée de l'erreur en pourcentage.

4. Choisir le meilleur modèle On peut comparer plusieurs modèles (LSTM, LSTM avec attention, etc.) grâce à ces mesures. Celui qui a les plus petites erreurs est souvent le meilleur.

En somme, la phase de prédiction et d'évaluation est essentielle pour valider la qualité du modèle appris et sélectionner la meilleure approche pour la tâche de prévision.

3.9 Conclusion

Ce chapitre a permis de comprendre les fondements des modèles LSTM ainsi que leurs principales variantes appliquées à la prévision de séries temporelles.

Les modèles LSTM offrent plusieurs avantages importants. Ils sont capables de capturer les dépendances à long terme dans les données, ce qui les rend particulièrement efficaces pour les séries temporelles complexes. Leurs variantes, comme les modèles avec attention ou combinés avec des réseaux de neurones convolutifs (LSTM-CNN), permettent d'améliorer encore les performances dans certains contextes spécifiques.

Cependant, ces modèles présentent aussi des inconvénients. Ils nécessitent généralement un grand volume de données pour bien fonctionner, leur entraînement est parfois coûteux en temps de calcul, et leur mise en œuvre requiert un réglage minutieux des hyperparamètres. De plus, leur fonctionnement interne est souvent difficile à interpréter, ce qui limite leur transparence par rapport aux méthodes statistiques classiques.

En résumé, les modèles LSTM et leurs variantes constituent des outils puissants pour la prévision des séries temporelles, à condition de bien maîtriser leur apprentissage et leur mise en œuvre.

Chapitre 4

Étude comparative sur des données réelles

4.1 Introduction

Ce chapitre présente la mise en œuvre concrète des méthodes de prévision étudiées dans les chapitres précédents. Il s'agit d'une étape essentielle du mémoire, car elle permet d'évaluer les performances réelles des modèles dans un cas pratique. L'objectif est de comparer deux approches : les modèles statistiques traditionnels, représentés ici par le modèle SARIMA, et les modèles d'intelligence artificielle, en particulier le modèle hybride LSTM-CNN.

4.2 Présentation des données

Les données utilisées dans cette étude proviennent d'un ensemble réel de mesures de consommation d'électricité collectées auprès de plusieurs entreprises sur une période allant de 2006 à 2020. Chaque entreprise (ou client) est identifié par un numéro unique.

Dans cette étude comparative, nous nous concentrons sur un seul client, identifié par le numéro **1033**. Cela permet une analyse détaillée de l'évolution de sa consommation électrique au cours du temps.

Les données sont structurées sous forme de séries chronologiques mensuelles, avec deux colonnes principales : `Date`, représentant le mois et l'année du relevé, et `Consom`, représentant la consommation d'électricité exprimée en kilowattheures (kWh).

Le traitement des données et la génération des graphiques ont été réalisés dans l'environnement **Google Colab**, en utilisant les bibliothèques Python telles que `pandas` pour la manipulation des données et `matplotlib` pour la visualisation.

La figure 4.1 illustre l'évolution de la consommation mensuelle d'électricité du client 1033 sur la période allant de 2006 à 2020.

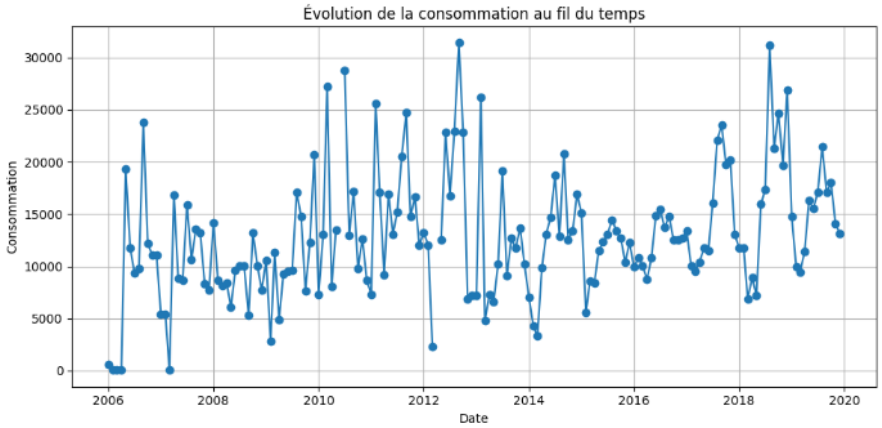

FIG. 4.1 : Série temporelle de la consommation électrique mensuelle du client 1033

On remarque que la consommation varie beaucoup d'un mois à l'autre. Il semble aussi y avoir des irrégularités ou des valeurs anormales. À ce stade, les données n'ont pas encore été nettoyées. Elles peuvent contenir des valeurs manquantes, des doublons ou des erreurs. Ces problèmes seront traités dans la section suivante, qui porte sur le prétraitement des données.

4.3 Prétraitement des données

Le prétraitement des données est une étape indispensable pour garantir la qualité et la fiabilité des modèles de prévision. Il comprend le nettoyage, la transformation et le découpage des données.

4.3.1 Nettoyage des données

Les données brutes de consommation peuvent contenir des valeurs manquantes, des doublons ou des erreurs. Pour corriger ces problèmes, nous avons appliqué plusieurs opérations en Python via Google Colab :

— Suppression des doublons avec `drop_duplicates()` ;
— Interpolation des valeurs manquantes avec `interpolate()` ;
— Vérification des statistiques avec `describe()` pour détecter d'éventuelles anomalies.

Figure 4.2 illustre la série temporelle de consommation du client 1033 après le nettoyage. On constate une structure plus régulière et cohérente des données par rapport à la figure brute présentée précédemment dans la section 4.2.

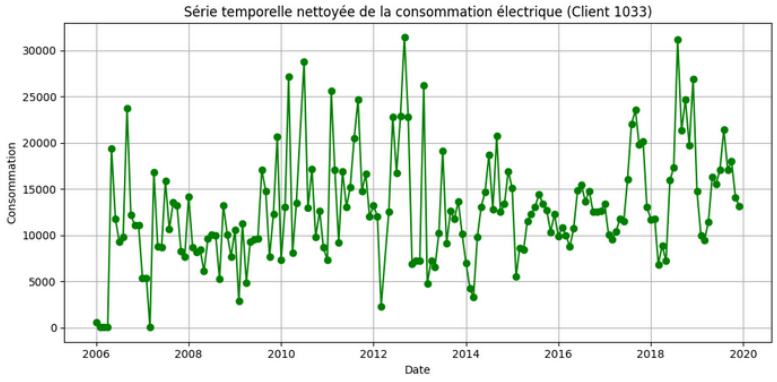

FIG. 4.2 : Série de consommation après nettoyage

4.3.2 Transformation des données

À l'issue du nettoyage des données, il est nécessaire de vérifier la stationnarité de la série temporelle, condition essentielle à l'application des modèles ARIMA ou SARIMA.

1. Test de stationnarité sur la série nettoyée

Le test de Dickey-Fuller augmenté (ADF) a été appliqué à la série mensuelle corrigée du client 1033, afin de détecter la présence d'une racine unitaire (non-stationnarité).

Résultats du test ADF (série nettoyée) : à l'aide de la fonction " adfuller " de la bibliothèque " statsmodels.tsa.stattools "

— Statistique ADF : `-2.137`
— p-value : `0.230`
— Seuil critique à 1% : `-3.473`
— Seuil critique à 5% : `-2.880`
— Seuil critique à 10% : `-2.577`

La statistique ADF étant supérieure aux seuils critiques et la p-value largement supérieure à 0.05, on ne peut pas rejeter l'hypothèse nulle de non-stationnarité. La série n'est donc pas stationnaire.

2. Différenciation de la série

Pour rendre la série stationnaire, deux types de différenciation ont été appliqués :

— La **différenciation simple d'ordre 1**, qui calcule la variation mensuelle (voir Figure 4.3).
— La **différenciation saisonnière** avec un décalage (*lag*) de 12 mois, afin d'éliminer la saisonnalité annuelle (voir Figure 4.4).

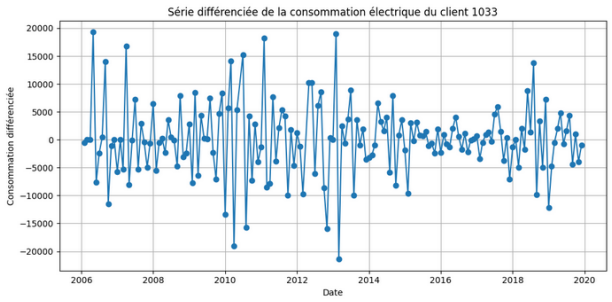

FIG. 4.3 : Série différenciée d'ordre 1

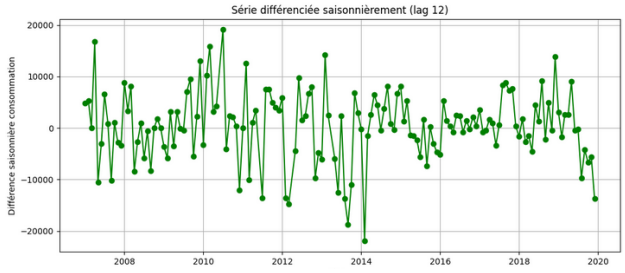

FIG. 4.4 : Série différenciée saisonnièrement (lag 12)

FIG. 4.5 : Transformation de la série de consommation mensuelle du client 1033

3. Test de stationnarité après différenciation

Le test ADF (Augmented Dickey-Fuller) a été appliqué à chaque série transformée :

Résultats pour la série différenciée d'ordre 1 :

— Statistique ADF : `-9.031`
— p-value : 5.44×10^{-15}
— Seuil critique à 1% : `-3.473`
— Seuil critique à 5% : `-2.880`
— Seuil critique à 10% : `-2.577`

Résultats pour la série différenciée saisonnièrement (lag 12) :

- Statistique ADF : `-4.354`
- p-value : 3.58×10^{-4}
- Seuil critique à 1% : `-3.477`
- Seuil critique à 5% : `-2.882`
- Seuil critique à 10% : `-2.578`

Dans les deux cas, la statistique ADF est inférieure aux seuils critiques, et la p-value est très faible, ce qui permet de rejeter l'hypothèse nulle de non-stationnarité. La série différenciée est donc stationnaire, confirmant l'efficacité de ces transformations pour la modélisation.

4.3.3 Prétraitement spécifique aux modèles LSTM

Les modèles LSTM nécessitent une structuration particulière des données temporelles pour exploiter efficacement les dépendances séquentielles. Cette section décrit les étapes spécifiques mises en œuvre, à savoir : la normalisation de la série, la création de séquences temporelles, et la séparation chronologique en ensembles d'entraînement, de validation et de test.

1-Normalisation de la série. Afin d'assurer une convergence rapide et stable de l'apprentissage, les données de consommation sont transformées via une normalisation min-max sur l'intervalle $[0, 1]$, en se basant uniquement sur les données d'entraînement pour éviter toute fuite d'information. Cette transformation homogénéise les échelles des entrées, ce qui est crucial pour les réseaux de neurones.

La Figure 4.6 illustre la série de consommation électrique après l'application d'une normalisation Min-Max. nous avons utilisé la fonction *"MinMaxScaler"* de la bibliothèque *sklearn*.

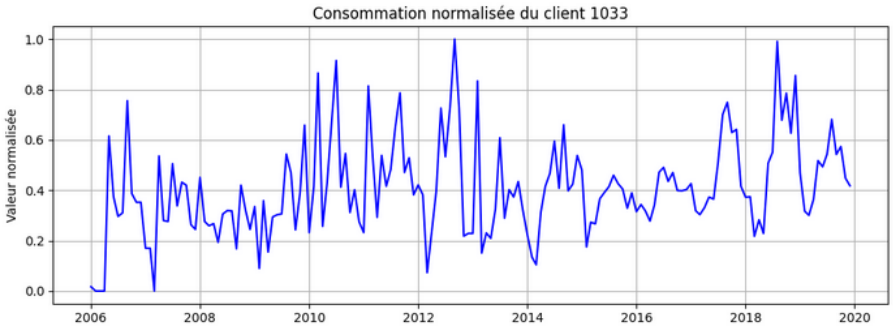

FIG. 4.6 : Série de consommation normalisée

2 - Création des séquences temporelles Après la normalisation des données, celles-ci sont découpées en séquences glissantes à l'aide d'une fenêtre de taille 12 mois ($W = 12$) et d'un pas de 1 mois ($H = 1$). Cela signifie que pour chaque séquence de 12 valeurs mensuelles consécutives, le modèle doit prédire la valeur du mois suivant.

Cette méthode permet au modèle LSTM de capturer les dépendances temporelles dans les données pour faire des prévisions plus précises.

4.4 Découpage des données

Avant d'entraîner les modèles de prévision, il est important de diviser les données en deux parties distinctes : une partie pour l'**entraînement** du modèle, et une autre pour tester sa capacité à bien prédire les valeurs futures.

Cette séparation est faite après le nettoyage des données, en respectant l'ordre chronologique : les données les plus anciennes servent à l'apprentissage, et les plus récentes à l'évaluation.

Sur un total de **165** mois de consommation, nous avons utilisé la fonction `iloc()` de la bibliothèque `pandas` pour effectuer la séparation suivante :

— **155** mois pour l'entraînement du modèle (*train*), soit environ **94%** des données ;

— **10** mois pour le test du modèle (*test*), correspondant à la période de **mars à décembre 2019**.

4.5 Modélisation avec SARIMA

4.5.1 Décomposition de la série temporelle

Avant de procéder à la modélisation, nous décomposons la série temporelle de consommation afin d'identifier visuellement ses composantes : tendance, saisonnalité et bruit. Cette étape permet de mieux comprendre la structure de la série et de motiver l'utilisation d'un modèle SARIMA, qui prend en compte à la fois les composantes saisonnières et non saisonnières.

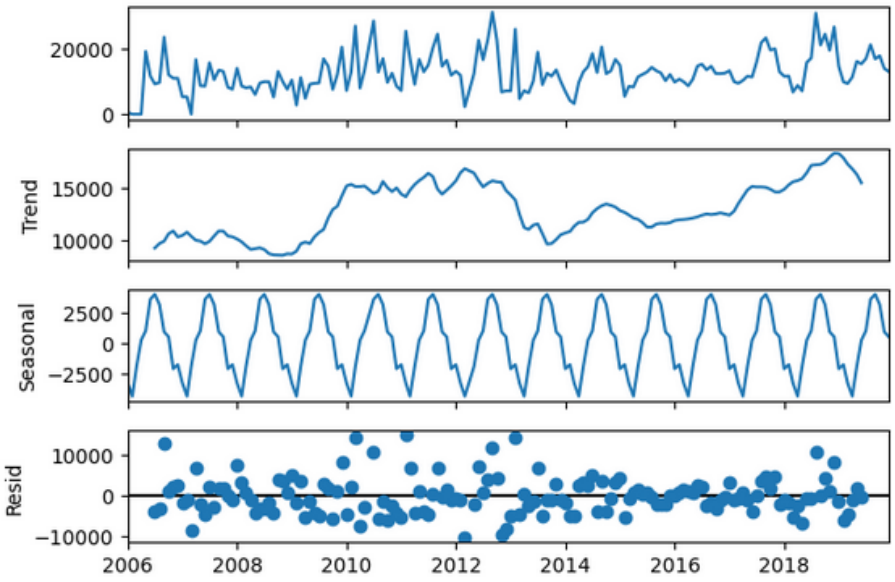

Fig. 4.7 : Décomposition de la série temporelle (tendance, saisonnalité, résidus)

La décomposition présentée dans la figure 4.7, réalisée à l'aide de la fonction `seasonal_decompose` de la bibliothèque `statsmodels.tsa.seasonal`, montre une forte composante saisonnière annuelle, ce qui justifie l'utilisation d'un modèle SARIMA avec une périodicité de 12 mois."

4.5.2 Identification des composantes

Après différenciation de la série, nous analysons les fonctions d'autocorrélation (ACF) et d'autocorrélation partielle (PACF) pour identifier les ordres des composantes AR et MA, ainsi que leurs composantes saisonnières.

Les graphiques de l'ACF et de la PACF sont obtenus dans la fiqure 4.8 à l'aide des fonctions `plot_acf()` et `plot_pacf()` de la bibliothèque `statsmodels.graphics.tsaplots`.

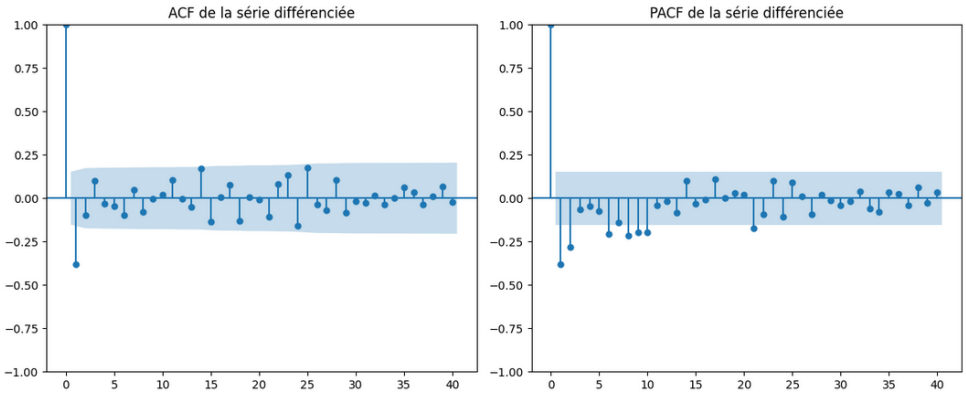

FIG. 4.8 : Fonctions ACF et PACF de la série différenciée

4.5.3 Estimation du modèle

L'estimation du modèle consiste à ajuster différentes spécifications SARIMA aux données d'apprentissage, dans le but de minimiser le critère d'information d'Akaike (AIC), qui équilibre la qualité de l'ajustement et la complexité du modèle

Pour cela, plusieurs combinaisons de paramètres (p, d, q) et (P, D, Q, s) ont été évaluées à l'aide de la fonction `SARIMAX` de la bibliothèque `statsmodels`. Le meilleur modèle a été sélectionné en fonction de la valeur minimale du critère AIC.

$$\textbf{SARIMA(1, 1, 2)} \times \textbf{(1, 1, 1, 12)} \quad \text{avec } \textbf{AIC} = \textbf{2566.63}$$

L'estimation des paramètres a été réalisée sur la série nettoyée et transformée du client 1033.

SARIMAX Results

Dep. Variable:	Consom	No. Observations:	156
Model:	SARIMAX(1, 1, 2)x(1, 1, [1], 12)	Log Likelihood	-1277.316
Date:	Fri, 06 Jun 2025	AIC	2566.633
Time:	21:46:22	BIC	2583.745
Sample:	0	HQIC	2573.585
	- 156		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.6450	0.319	2.025	0.043	0.021	1.269
ma.L1	-1.4615	0.341	-4.287	0.000	-2.130	-0.793
ma.L2	0.4998	0.299	1.674	0.094	-0.085	1.085
ar.S.L12	0.0008	0.113	0.007	0.994	-0.220	0.222
ma.S.L12	-1.1567	0.052	-22.104	0.000	-1.259	-1.054
sigma2	2.273e+07	2.28e-08	9.96e+14	0.000	2.27e+07	2.27e+07

Ljung-Box (L1) (Q):	0.06	Jarque-Bera (JB):	19.61
Prob(Q):	0.80	Prob(JB):	0.00
Heteroskedasticity (H):	0.41	Skew:	0.68
Prob(H) (two-sided):	0.00	Kurtosis:	4.34

FIG. 4.9 : Résumé de l'ajustement du modèle SARIMA

4.5.4 Validation du modèle

Pour vérifier si le modèle SARIMA s'ajuste correctement aux données, nous analysons les résidus à l'aide de tests statistiques. Ces tests permettent de détecter des problèmes comme l'autocorrélation, la non-normalité ou l'hétéroscédasticité.

— **Test de Jarque-Bera** (`scipy.stats.jarque_bera`)
Statistique : 40.0764 p-value : 0.0000
les résidus ne sont pas normalement distribués.
— **Test de Ljung-Box (lags=10)** (`statsmodels.stats.diagnostic.acorr_ljungbox`)
Statistique : 5.3109 p-value : 0.8695
Pas d'autocorrélation significative.
— **Test ARCH d'Engle** (`arch.unitroot.het_arch`)
Statistique : 23.8283 p-value : 0.0081
Hétéroscédasticité (effet ARCH) détectée.

Ces résultats montrent que les résidus ne sont pas parfaitement distribués, mais qu'ils ne présentent pas d'autocorrélation, ce qui est un critère important pour passer à l'étape de prévision.

La figure 4.10 illustre les graphiques de diagnostic des résidus du modèle nous avons utilisé la fonction *plot.diagnostics()* de la bibliothèque *matplotlib* :

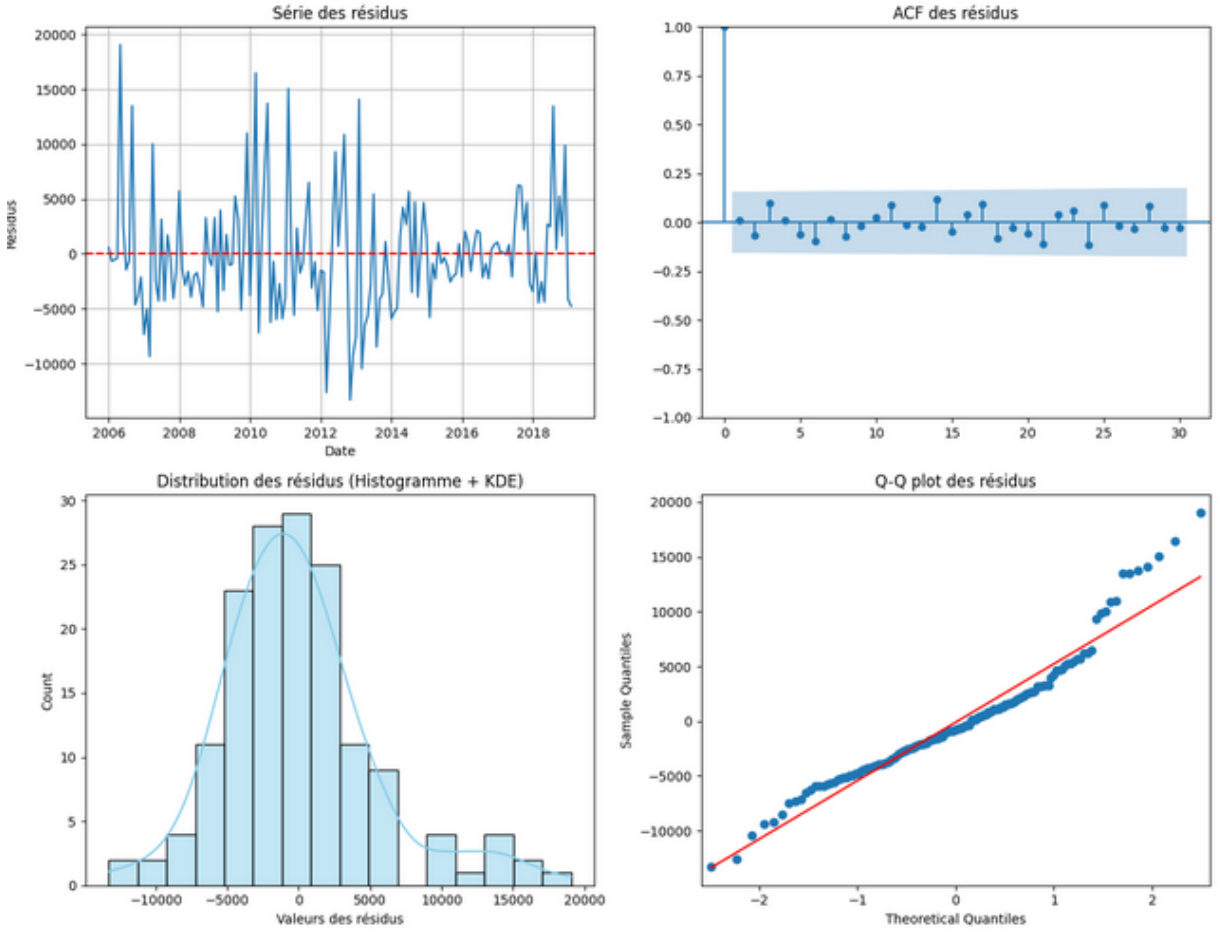

FIG. 4.10 : Analyse diagnostique des résidus du modèle SARIMA

4.5.5 Prévisions

Le modèle SARIMA a été utilisé pour prédire la consommation mensuelle du client 1033 pour les mois de mars à décembre 2019. Les résultats sont présentés ci-dessous.

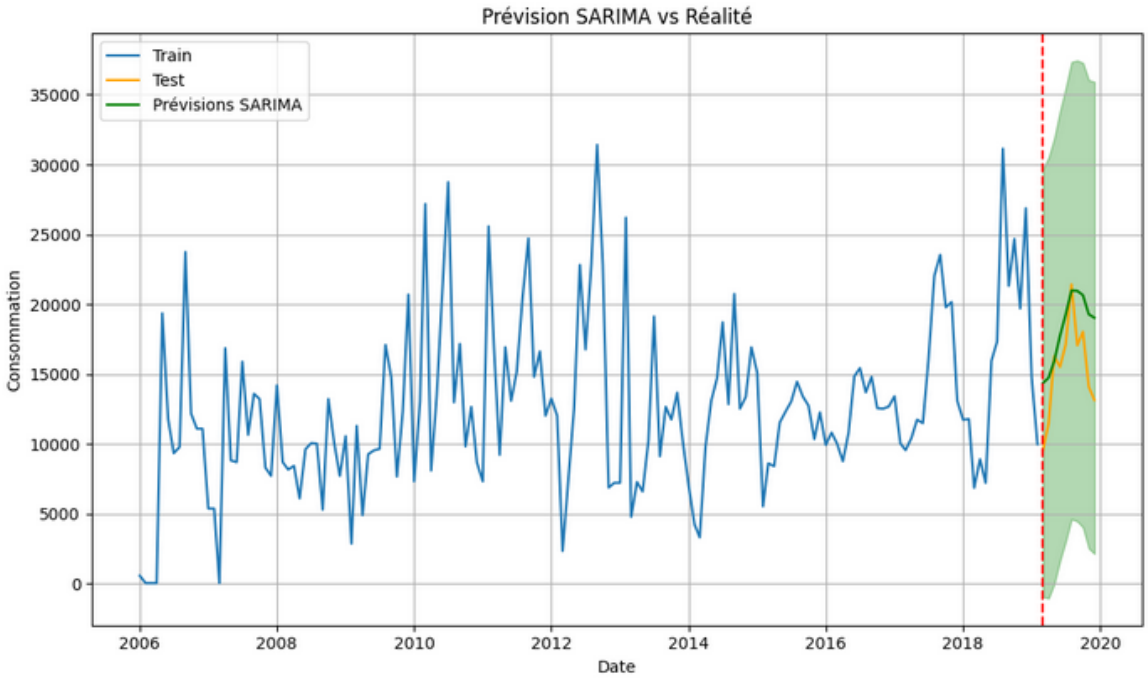

FIG. 4.11 : Prévisions SARIMA vs valeurs réelles

Le tableau 4.1 présente les valeurs réelles, les valeurs prédites et le pourcentage d'erreur absolu moyen (MAPE) pour la prédiction de la consommation mensuelle du client 1033 pour les mois de mars à décembre 2019.

TAB. 4.1 : Résultats de la prédiction de la consommation d'électricité avec SARIMA

Date	**Valeur réelle**	**Valeur prédite**	**MAPE (%)**
2019-03-01	9475	14346.63	51.42
2019-04-01	11445	14733.30	28.73
2019-05-01	16279	15934.62	2.12
2019-06-01	15514	17753.56	14.44
2019-07-01	17107	19256.92	12.57
2019-08-01	21423	20988.25	2.03
2019-09-01	17055	20958.72	22.89
2019-10-01	18039	20653.09	14.49
2019-11-01	14096	19280.76	36.78
2019-12-01	13148	19035.17	44.78
Moyenne	–	–	**23.02**

Analyse des résultats La Figure 4.11 montre la comparaison entre les vraies valeurs et celles prédites par le modèle SARIMA. On voit que le modèle suit bien la tendance générale de la consommation d'électricité.

Cependant, certaines prévisions sont loin des valeurs réelles, comme en mars, novembre et décembre, où l'erreur dépasse 40%. À l'inverse, entre mai et août, les prédictions sont très proches des vraies valeurs, avec une erreur faible.

En moyenne, l'erreur (MAPE) est de 23.02%. Cela signifie que le modèle donne des résultats globalement acceptables, mais il a du mal à bien prédire certains mois où la consommation change fortement.

4.6 Modélisation avec le modèle LSTM-CNN

Dans cette section, nous appliquons une architecture combinée LSTM-CNN pour prédire la consommation mensuelle d'électricité du client 1033. Cette combinaison vise à exploiter les avantages des réseaux convolutifs pour l'extraction de caractéristiques locales et ceux des LSTM pour la modélisation des dépendances temporelles de long terme.

4.6.1 Architecture du modèle

L'architecture du modèle est séquentielle et a été construite avec les couches de la bibliothèque `tensorflow.keras.layers` de Keras. Elle se compose des éléments suivants :

— Une couche `LSTM` avec 64 unités pour capturer la dynamique temporelle ;
— Une couche `Dropout` (taux de 0,2) pour limiter le surapprentissage ;
— Une couche `Conv1D` avec 32 filtres, une taille de noyau de 3, et l'activation `ReLU` ;
— Une couche `MaxPooling1D` pour réduire la dimension temporelle ;
— Une couche `Flatten` pour vectoriser les données issues des couches précédentes ;
— Une couche `Dense` (fully connected) avec 32 neurones et une activation `ReLU` ;
— Une seconde couche `Dropout` ;
— Une couche `Dense` finale avec une unité pour la sortie.

Ce modèle compte un total de **28 257 paramètres entraînables**. Il est compilé avec l'optimiseur `Adam` avec la bibliothèque *Keras* et la fonction de perte `Log-Cosh Loss`.

Layer (type)	Output Shape	Param #
lstm_6 (LSTM)	(None, 12, 64)	16,896
dropout_12 (Dropout)	(None, 12, 64)	0
conv1d_6 (Conv1D)	(None, 10, 32)	6,176
max_pooling1d_6 (MaxPooling1D)	(None, 5, 32)	0
flatten_6 (Flatten)	(None, 160)	0
dense_12 (Dense)	(None, 32)	5,152
dropout_13 (Dropout)	(None, 32)	0
dense_13 (Dense)	(None, 1)	33

FIG. 4.12 : Architecture du modèle LSTM-CNN

4.6.2 Entraînement du modèle

Le modèle est entraîné sur les 155 premières observations de la série temporelle, durant 50 époques, avec une taille de batch fixée à 16.L'entraînement a été effectué à l'aide de la méthode `model.fit()` de la bibliothèque *Keras*.

La figure 4.13 montre comment la perte change pendant l'entraînement du modèle LSTM-CNN. On a deux courbes : la courbe bleue pour la perte d'entraînement, et la courbe orange pour la perte de validation.

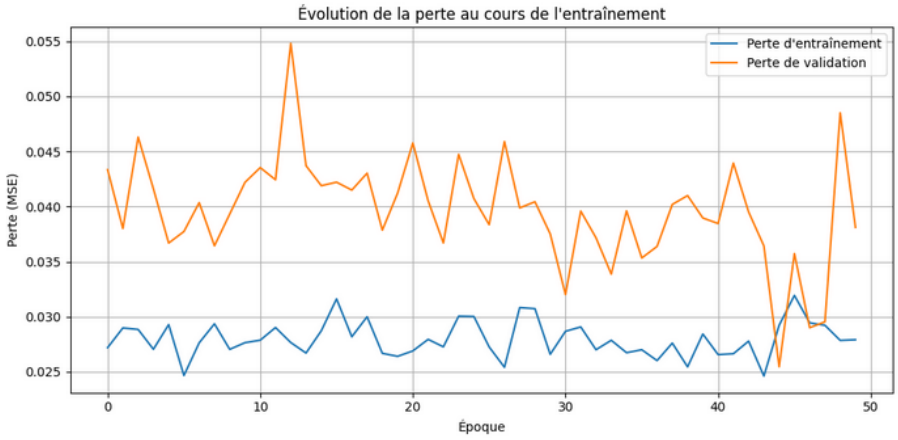

FIG. 4.13 : Évolution de la fonction de perte durant l'apprentissage du modèle LSTM-CNN

Analyse de l'évolution de la perte

On voit que la perte d'entraînement est assez stable et reste basse (entre 0,025 et 0,032). Cela veut dire que le modèle apprend bien à reproduire les données qu'il voit pendant l'entraînement.

En revanche, la perte de validation change beaucoup d'une époque à l'autre. Par exemple, à la 12^{e} époque, elle monte très haut, au-dessus de 0,055. Cela montre que le modèle a parfois du mal à bien prédire sur les nouvelles données (celles qu'il n'a pas vues pendant l'entraînement).

Même si la courbe orange monte et descend beaucoup, elle ne monte pas tout le temps. Donc le modèle ne fait pas vraiment de surapprentissage, mais il a un peu de mal à rester stable.

4.6.3 Résultats de la prévision

Les prédictions sur la période de test (mars à décembre 2019) sont comparées aux valeurs réelles dans la figure 4.14, illustrant la capacité du modèle à suivre la tendance de consommation.

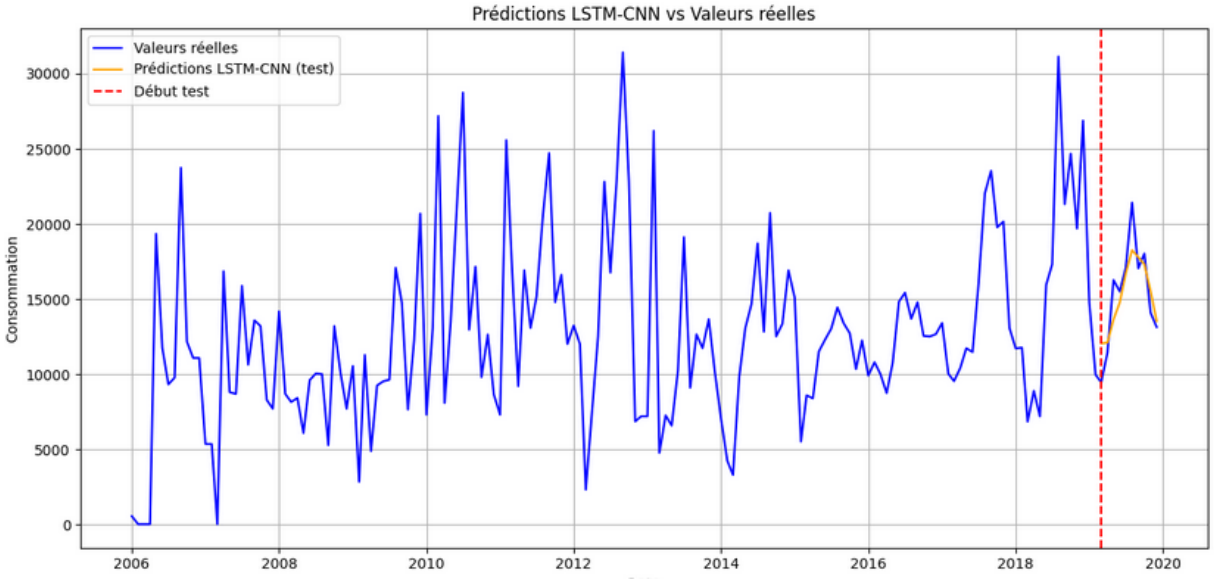

FIG. 4.14 : Prévisions du modèle LSTM-CNN comparées aux valeurs réelles

Le tableau 4.2 présente les valeurs observées, prédites, ainsi que l'erreur absolue en pourcentage pour chaque mois. Le **MAPE global obtenu est de 9,26%**.

Tab. 4.2 : Résultats de la prédiction avec le modèle LSTM-CNN

Date	Valeur réelle	Valeur prédite	Erreur absolue (%)
2019-03-01	9475	12062.03	27.30
2019-04-01	11445	12116.94	5.87
2019-05-01	16279	13649.76	16.15
2019-06-01	15514	14780.74	4.73
2019-07-01	17107	16727.00	2.22
2019-08-01	21423	18262.45	14.75
2019-09-01	17055	17799.53	4.37
2019-10-01	18039	17318.92	3.99
2019-11-01	14096	15537.81	10.23
2019-12-01	13148	13544.64	3.02
MAPE global	–	–	**9.26**

4.6.4 Analyse des résultats

Le modèle LSTM-CNN donne un bon résultat global avec une erreur moyenne (MAPE) de **9,26%**. Cela veut dire que, dans l'ensemble, les prédictions sont assez proches des vraies valeurs de consommation.

Mais si on regarde les résultats mois par mois, on voit que les erreurs sont plus grandes au début. Par exemple, en mars 2019, l'erreur atteint **27,30%**, ce qui montre que le modèle a eu du mal à bien prédire ce mois-là. Cela peut être à cause d'un changement brutal dans les données que le modèle n'a pas bien appris. En mai aussi, l'erreur est un peu élevée avec **16,15%**.

Par contre, à partir de juin 2019, les résultats s'améliorent. Les erreurs deviennent plus petites. En juillet, l'erreur est très faible, seulement **2,22%**, et dans d'autres mois comme octobre ou novembre, les erreurs restent aussi assez basses.

En résumé, le modèle LSTM-CNN arrive bien à suivre la tendance de la consommation d'électricité, surtout après les premiers mois. Il est un peu sensible aux variations brusques, mais donne de bonnes prévisions dans les périodes plus stables.

4.7 Comparaison des résultats

Les performances des modèles SARIMA et LSTM-CNN ont été comparées en utilisant le MAPE (Mean Absolute Percentage Error), un indicateur qui mesure l'erreur moyenne en pourcentage entre les valeurs réelles et les valeurs prédites.

Les résultats de la prédiction sont présentés dans les tableaux 4.1 et 4.2, pour les mois de mars à décembre 2019. Le tableau 4.1 montre que le modèle SARIMA atteint un MAPE global de **23.02 %**, tandis que le tableau 4.2 montre que le modèle LSTM-CNN obtient un MAPE global plus faible, égal à **9.26 %**.

Ces résultats indiquent que le modèle LSTM-CNN donne des prédictions plus proches des valeurs réelles que le modèle SARIMA, avec une erreur moyenne environ deux fois plus petite.

4.8 Analyse et discussion

Les résultats montrent que le modèle LSTM-CNN est plus performant que le modèle SARIMA pour prédire la consommation d'électricité. Avec un MAPE global de **9.26 %**, il fait moins d'erreurs que SARIMA, qui a un MAPE de **23.02 %**.

Cette différence s'explique par la nature des modèles. Le modèle SARIMA repose sur des relations statistiques simples, ce qui fonctionne bien quand les données sont régulières. Mais lorsque les données varient beaucoup ou de manière complexe, SARIMA devient moins précis.

Le modèle LSTM-CNN, quant à lui, est basé sur des réseaux de neurones. Il peut apprendre des schémas complexes dans les données. Les couches LSTM sont capables de capter la tendance à long terme, et les couches CNN permettent d'identifier des variations locales. Cette combinaison aide le modèle à mieux suivre les changements dans la consommation.

Par exemple, en avril 2019, le modèle LSTM-CNN fait une erreur de seulement **5.87 %**, contre **28.73 %** pour le modèle SARIMA. Dans plusieurs mois, le modèle LSTM-CNN donne des résultats plus proches de la réalité.

Cependant, il faut noter que le modèle LSTM-CNN est plus difficile à mettre en œuvre. Il demande plus de données, plus de temps de calcul, et ses résultats sont moins faciles à expliquer. En revanche, SARIMA est plus simple à utiliser et à interpréter, même s'il est moins précis.

En conclusion, si l'objectif est d'avoir des prédictions plus précises, le modèle LSTM-CNN est préférable. Mais si on cherche un modèle plus rapide et facile à utiliser, SARIMA peut suffire.

4.9 Conclusion

Ce chapitre a permis d'appliquer, sur un cas réel, les méthodes de prévision étudiées dans le cadre de ce mémoire. À partir des données de consommation électrique du client 1033, nous avons mis en œuvre deux approches distinctes : le modèle SARIMA, basé sur des principes statistiques classiques, et le modèle LSTM-CNN, issu des techniques modernes d'apprentissage profond.

Les résultats obtenus montrent que le modèle LSTM-CNN offre de meilleures performances que le modèle SARIMA, avec un taux d'erreur moyen plus faible. Cela confirme la capacité des réseaux de neurones à modéliser des dynamiques complexes et à s'adapter à des comportements non linéaires dans les séries temporelles. En revanche, le modèle SARIMA, bien qu'il soit plus simple à interpréter et à implémenter, se montre moins précis face à des séries présentant une forte variabilité.

L'étude met en évidence l'intérêt d'introduire des méthodes avancées dans les systèmes de prévision de la consommation énergétique, notamment dans un contexte industriel tel que celui de SONELGAZ. Elle ouvre également la voie à d'autres travaux qui pourraient intégrer davantage de données exogènes, explorer d'autres architectures de réseaux, ou encore étendre l'analyse à un ensemble plus large de clients afin de généraliser les résultats obtenus.

Conclusion générale

Ce mémoire a présenté une étude comparative entre deux grandes familles de méthodes de prévision appliquées à la consommation électrique : les méthodes statistiques classiques, comme les modèles SARIMA, et les approches modernes basées sur l'apprentissage profond, notamment les modèles LSTM .

Les résultats obtenus ont montré que les méthodes traditionnelles restent des outils fiables, faciles à interpréter, et efficaces dans des contextes simples, où la structure des données temporelles est relativement régulière. Toutefois, elles atteignent leurs limites face à des séries plus complexes, marquées par des variations non linéaires ou des dépendances à long terme.

À l'inverse, les modèles basés sur les réseaux de neurones, en particulier le modèle LSTM-CNN utilisé dans cette étude, ont démontré une meilleure capacité à capturer la dynamique des données réelles. Grâce à leur architecture, ces modèles peuvent exploiter à la fois les dépendances temporelles et les motifs locaux, ce qui améliore la précision des prévisions. Dans le cas étudié, le modèle LSTM-CNN a surpassé le modèle SARIMA en termes de performance, mesurée notamment par l'erreur MAPE.

Néanmoins, cette amélioration s'accompagne d'une complexité plus importante : le prétraitement des données, le choix de l'architecture du modèle et le réglage des hyperparamètres nécessitent un travail rigoureux et des ressources informatiques adaptées.

En conclusion, le choix de la méthode de prévision doit tenir compte du contexte d'application, de la qualité des données disponibles et des moyens techniques. Cette étude confirme l'intérêt croissant des méthodes d'apprentissage profond dans le domaine énergétique, et ouvre la voie à de futures recherches intégrant des données exogènes (météo, événements, etc.) ou explorant d'autres variantes des modèles LSTM.

Bibliographie

[1] H. Akaike. "A New Look at the Statistical Model Identification". In : *IEEE Transactions on Automatic Control* (1974), p. 716-723.

[2] C.-A. Azencott. *Introduction au Machine Learning – 2e éd.* Dunod, 2022.

[3] D. Bahdanau, K. Cho et Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In : *arXiv preprint arXiv :1409.0473* (2014).

[4] A. Bensalma. "Introduction aux tests de racine unitaire". Mémoire de Master, Chapitre 3, p. 37. Mém. de mast. Pôle universitaire de Koléa, Tipaza, Algérie, 2022.

[5] G. E. P. Box et al. *Time Series Analysis : Forecasting and Control*. John Wiley & Sons, 2015.

[6] R. G. Brown. "Exponential Smoothing for Predicting Demand". In : *Journal of the Operational Research Society* (1956), p. 25-29.

[7] R. G. Brown. *Smoothing, Forecasting and Prediction of Discrete Time Series*. Courier Corporation, 2004.

[8] S. R. y Cajal. "Histology of the Nervous System of Man and Vertebrates". In : *Oxford University Press* (1899), p. 1-800.

[9] J. Carreau. *Analyse des séries temporelles*. Chapitre 2 : Stationnarité des processus temporels. Rennes : Presses Universitaires de Rennes, 2019.

[10] G. Casella et R. L. Berger. *Statistical Inference*. 2e éd. Duxbury, 2002.

[11] C. Chatfield. "The Analysis of Time Series : An Introduction". In : (2004).

[12] J. L. Elman. "Finding Structure in Time". In : *Cognitive Science* 14 (1990), p. 179-211.

[13] R. F. Engle. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In : *Econometrica* (1982), p. 987-1007.

[14] I. Goodfellow, Y. Bengio et A. Courville. *Deep Learning*. MIT Press, 2016.

[15] Y. Goude. *Séries temporelles 2A*. français. Support de cours, Master 2, Université Paris-Saclay. 2021.

[16] D. Gupta et al. *A Hybrid CNN-LSTM Model for Pre-miRNA Classification – Scientific Figure on ResearchGate*. `https://www.researchgate.net/figure/The-basic-architecture-of-the-CNN-LSTM-network_fig2_353080439`. 2021.

[17] J. D. Hamilton. "Time Series Analysis". In : *Princeton University Press* (1994).

[18] E. J. Hannan et B. G. Quinn. "The Determination of the Order of an Autoregression". In : *Journal of the Royal Statistical Society : Series B* (1979), p. 190-195.

[19] S. Hochreiter et J. Schmidhuber. "Long Short-Term Memory". In : *Neural Computation* (1997), p. 1735-1780.

[20] C. C. Holt. *Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages*. OTexts, 2004.

[21] R. J. Hyndman et G. Athanasopoulos. *Forecasting : Principles and Practice*. OTexts, 2018.

[22] Inconnu. *Initiation à l'analyse des séries temporelles et à la prévision*. 2025.

[23] C. M. Jarque et A. K. Bera. "Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals". In : *Economics Letters* (1980), p. 255-259.

[24] Q. Kang et al. "Attention-Based LSTM Predictive Model for the Attitude and Position of Shield Machine in Tunneling". In : *Underground Space* (2023), p. 335-350. issn : 2467-9674.

[25] D. P. Kingma et J. Ba. *Adam : A Method for Stochastic Optimization*. 2014.

[26] A. Lagnoux. *Renforcement Statistique : Séries Chronologiques*. Université de Toulouse le Mirail, 2015.

[27] Y. LeCun, Y. Bengio et G. Hinton. "Deep Learning". In : *Nature* (2015), p. 436-444.

[28] Y. LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In : *Proceedings of the IEEE* (1998), p. 2278-2324.

[29] M. Minsky et S. Papert. *Perceptrons : An Introduction to Computational Geometry*. Cambridge, MA : MIT Press, 1969, p. 104.

[30] PseudoLab. *CNN-LSTM Tutorial — Time Series Forecasting*. 2020.

[31] Y. Qin et al. "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction". In : *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)* (2017), p. 2627-2633.

[32] ResearchGate. *Initiation à l'analyse des séries temporelles et à la prévision*. 2025. url : https://www.researchgate.net/figure/Les-donnees-de-consommation-de-fuel-lourd-en-France_fig3_228643642 (visité le 11/06/2025).

[33] ResearchGate. *Utilisation des techniques avancées pour l'observation et la commande d'une machine asynchrone : application à une éolienne*. https://www.researchgate.net/figure/Fonctions-dactivation-dun-neurone-artificiel_fig2_322194356.

[34] ResearchGate contributors. *Complete LSTM Architecture with Equations Showing How Information Moves Through the Cell*. 2020. url : https://www.researchgate.net/figure/Complete-LSTM-architecture-with-equations-showing-how-information-moves-through-the-cell_fig2_351291905.

[35] F. Rosenblatt. "The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain". In : *Psychological Review* (1958), p. 386-408.

[36] D. E. Rumelhart, G. E. Hinton et R. J. Williams. "Learning Representations by Back-Propagating Errors". In : *Nature* (1986), p. 533-536.

[37] D. Salinas et al. "DeepAR : Probabilistic Forecasting with Autoregressive Recurrent Networks". In : *International Journal of Forecasting* (2020), p. 1181-1191.

[38] G. Schwarz. "Estimating the Dimension of a Model". In : *The Annals of Statistics* (1978), p. 461-464.

[39] Scientific Figure on ResearchGate. *La segmentation des images médicales en utilisant les champs de Markov cachés et la technique Deep Learning*. 2025. url : https://www.researchgate.net/figure/La-relation-entre-AI-ML-et-DL-1_fig12_344266838.

[40] R. Sharma, P. Agarwal et D. Kaushik. *COVID-19 Epidemic Analysis in India with Multi-Source State-Level Datasets*. 2022.

[41] R. H. Shumway et D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer Texts in Statistics. New York : Springer, 2000.

[42] Wikimedia Commons. *Neurone*. 2025.

[43] P. R. Winters. “Forecasting Sales by Exponentially Weighted Moving Averages”. In : *Management Science* (1960), p. 324-342.

Résumé

Ce mémoire présente une étude comparative entre deux grandes approches de **prévision** des **séries temporelles** appliquées à la **consommation d'électricité** : les méthodes statistiques classiques, comme les modèles **SARIMA**, et les méthodes récentes basées sur l'apprentissage profond, notamment les **réseaux de neurones LSTM** et leurs variantes. L'objectif est d'évaluer leur efficacité sur des données réelles. Les résultats montrent que les modèles **LSTM-CNN** offrent de meilleures performances dans les situations complexes, bien qu'ils nécessitent davantage de ressources de calcul. Ce travail permet ainsi de mieux comprendre les avantages et les limites de chaque approche.

Mots-clés : prévision, séries temporelles, LSTM, SARIMA, consommation électrique.

ملخص

يتناول هذا البحث دراسة مقارنة بين طريقتين رئيسيتين في مجال **التنبؤ** باستخدام **السلاسل الزمنية** لتحليل **استهلاك الكهرباء**، وذلك من خلال استخدام النماذج الإحصائية التقليدية مثل **نموذج ساريما (SARIMA)** من جهة، ومن جهة أخرى، النماذج الحديثة المبنية على التعلم العميق، خاصة **الشبكات العصبية من نوع LSTM (الذاكرة طويلة وقصيرة الأمد)** وتفرعاتها. يهدف هذا العمل إلى تقييم فعالية هذه النماذج عند تطبيقها على بيانات حقيقية. وقد أظهرت النتائج أن نماذج **LSTM-CNN** تحقق أداءً تنبؤياً أفضل في الحالات المعقدة، بالرغم من حاجتها إلى موارد حسابية أكبر. ويساهم هذا البحث في توضيح نقاط القوة والضعف في كل نهج، ما يساعد على اختيار النموذج الأنسب حسب طبيعة البيانات.

الكلمات المفتاحية: التنبؤ، السلاسل الزمنية، الشبكة العصبية LSTM، نموذج ساريما، استهلاك الكهرباء.

Abstract

This thesis presents a comparative study between two main approaches to **forecasting time series** related to **electricity consumption** : traditional statistical models such as **SARIMA**, and modern deep learning methods, especially **LSTM neural networks** and their extensions. The aim is to assess the effectiveness of these models on real data. The results show that **LSTM-CNN** models provide better predictive performance in complex situations, although they require greater computational resources. This work contributes to understanding the strengths and weaknesses of each approach, and to guiding model selection based on the data characteristics.

Keywords : forecasting, time series, LSTM, SARIMA, electricity consumption.