الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Salhi Ahmed- Naama

Institut des sciences et technologies

Département de Mathématiques et Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master
En : Mathématiques

Spécialité : Probabilités, Statistique et Application

Intitulé

Prévision par les modèles ARIMA et ANN

Présenté par :
SLAMANI Boubakeur

Soutenu : le 03 Juillet 2022

Devant le jury composé de :

M. **MOULAY KHATIR Smain**,	M.C.B.	C.U. Naâma	Président du jury
Mme **KHALOUATI Hafidha**,	M.A.A.	C.U.Naâma	Examinateur
M. **GASMI Laid**,	M.C.A.	Univ. Adrar	Rapporteur.
M. **KENOUZA Jamel**,	M.C.A.	C.U.Naâma	Co-encadreur.

Année universitaire : 2021/2022

Remerciement

Je tiens à remercier en premier lieu et avant tout ***Allah*** *le tout puissant, qui nous a donné la force et la patience d'accomplir notre travail dans les meilleures conditions.*

Je tiens également à remercier du coeur mon encadreur Dr. ***Gasmi Laid****, son aide précieuse et ses conseilles judicieuses. Un remerciement très particulier va au Dr.* ***Kenouza Jamel*** *mon co-encadreur pour l'attention qu'il a porté à mon travail.*

Je voudrai également remercier les membres de jury Dr. ***MOULAY KHATIR Smain*** *et Mme.****KHALOUATI Hafidha*** *maîtres de conférences au centre universitaire Salhi Ahmed de Naâma, qui m'ont fait l'honneur de juger ce modeste travail.*

le 9 juillet 2022.

Liste des Abréviations

$ARIMA$: Autoregressive Integrated Moving Average

DQN : Deep Q-Network

GAN : Generative adversarial networks

MLP : Multi-Layer Perceptron

$PACF$: Fonction d'auto-corrélation partielle

PMC : Perceptrons Muiti-Couches

RNN : Recurrent Neural Network

ρ_l : Le coefficient d'auto corrélation d'ordre l

ANN : Artificial Neural Network.

$AR(p)$: auto-régressifs d'ordre p.

CNN : convolutional neural networks

MA : Moyenne mobile.

$N(0, {\sigma_\varepsilon}^2)$: Loi normal de moyen $\mu = 0$, et de $var = {\sigma_\varepsilon}^2$

$RBFN$: Radial Basis Function Networks

ACF : Fonction d'auto corrélatio

LSTM : Long Short-Term Memory

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

Introduction

L'exploration de séries temporelles est un domaine très important de l'analyse de données, extrayant des connaissances à partir d'observations passées pour identifier l'évolution d'un phénomène dans le présent et faciliter sa projection dans le futur. Cette exploration met également en évidence certaines corrélations qui peuvent sembler incontournables à l'œil nu et identifie certaines caractéristiques nécessaires pour comprendre l'état actuel de ce phénomène. Beaucoup d'efforts ont été faits par la communauté scientifique pour améliorer l'exploration et l'analyse des séries chronologiques.

L'un des modèles les plus célèbres et les plus populaires pour ce type d'analyse est le modèle *ARIMA* introduit par Box et Jenkins en 1976 [1]. La popularité de ce modèle vient principalement de sa flexibilité, de sa qualité et de sa polyvalence. Il est disponible pour les processus auto-régressifs (AR), les processus à moyenne mobile (MA) et la combinaison des deux ($ARMA$). *ARIMA* fonctionne sur des processus dits stationnaires, mais peut s'adapter à des processus qui ne sont pas dans sa version intégrée (Le « I » dans $ARIMA$) . Enfin, ce modèle peut également avoir des fluctuations saisonnières dans la série chronologique *SARIMA*. Un important La limite de ce modèle est son taux d'erreur élevé dès que la normalité et/ou la linéarité des données est perdue.

Récemment, l'évolution des technologies de l'information a contribué à générer des quantités massives de données d'une part, et à préparer le support informatique nécessaire aux calculs d'autre part. Cette évolution a également introduit la mise en place de modèles d'apprentissage plus performants que ceux proposés par la statistique classique.

Les réseaux de neurones atteignent alors leur apogée et surpassent les modèles traditionnels, qu'il s'agisse de régression, de classification ou plus des problèmes avancés tels que le traitement de données multimédia (image, son) grâce au *deep learning*. La particularité des réseaux de neurones est leur tendance à s'adapter aux caractéristiques créées par les données elles-mêmes.

Cette approche orientée données facilite l'apprentissage malgré la perte de certaines caractéristiques essentielles en statistique classique telles que la linéarité, la normalité, l'homosédasticité ou l'indépendance d'observation. Comme le modèle *ARIMA*, les réseaux

de neurones se déclinent en plusieurs architectures correspondant chacune à un type de problématique (*RNN*, *CNN*, *DQN*, *GAN*, etc.).

Ce mémoire s'organise en quatre chapitres traitant :

1. Le problème d'identification de modèles des séries temporelles linéaires ;
2. Les différents types de réseaux de neurones artificielles ;
3. Application sur les données réelles , comparaissant et hybridation ;
4. Conclusion.

Dans ce manuscrit, nous essayons de coupler l'effet de ces deux formes de modélisation pour produire un résultat qui pourrait dépasser les performances individuelles de chaque modèle. Nous présentons une approche de modélisation hybride pour l'analyse de séries chronologiques. Ce dernier combine *ARIMA* et réseaux de neurones artificiels. Nous présentons une approche de modélisation hybride pour l'analyse de séries chronologiques. Ce dernier combine *ARIMA* et réseaux de neurones artificiels *LSTM* . Combiner les deux nous aiderait à capturer certains motifs qui seraient inaccessibles à l'un des deux modèles sans le support de l'autre et fournir ainsi des résultats satisfaisants dans la prédiction des séries chronologiques. Le reste de ce travail est organisé comme suit : Le placement des deux modèles dans leurs cadres théoriques respectifs.

Le chapitre 3 décrive la partie empirique de l'étude en donnant un aperçu de la nature des données impliquées et des métriques utilisées pour mesurer la performance de notre modèle sur ces données. Le dernière chapitre résume les résultats remarquables de l'étude.

CHAPITRE 2

Processus classiques (ARIMA)

Les modèles ARIMA offrent une autre approche de la prévision des séries chronologiques. Le lissage exponentiel et les modèles ARIMA sont les deux approches les plus largement utilisées pour la prévision des séries chronologiques et fournissent des approches complémentaires au problème. Alors que les modèles de lissage exponentiel sont basés sur une description de la tendance et de la saisonnalité des données, les modèles ARIMA visent à décrire les autocorrélations dans les données.

Avant d'introduire les modèles ARIMA, nous étudions les caractéristiques statistiques-en terme de stationnarité-des séries temporelles en présentant les différents tests(Dickey-Fuller,corrélogramme, etc...) s'y rapportant. Puis, nous présentons différentes classes de modèles (AR, MA, ARMA) en étudiant leurs propriétés. Enfin, la méthode de Box et Jenkins qui systématise une démarche d'analyse des séries temporelles.

Nous devons d'abord discuter du concept de stationnarité et de la technique de différenciation des séries temporelles.

2.1 Introduction et premières définitions

L'étude des séries temporelles, ou séries chronologiques, correspond à l'analyse statistique d'observations régulièrement espacées dans le temps. Elles ont été utilisées en astronomie ('on the periodicity of sunspots', 1906), en météorologie ('time-seires regression of sea level on weather', 1968), en théorie du signal ('Noise in FM receivers', 1963), en biologie ('the autocorrelation curves of schizophrenic brain waves and the power spectrum', 1960), en économie (' time-series analysis of imports, exports and other economic variables ', 1971)...etc.[2]
Pour la majorité de ces phénomènes, il existe souvent une dépendance temporelle entre les observations, ce qui a des modélisations de type autorégressif : on utilise le passé pour expliquer le présent et prédire le futur. Modéliser et prédire une série chronologique suppose, dans la plupart des cas, de faire des hypothèses sur son comportement ; Puiqu'il s'agit de séries non-déterministes, il va falloir que la comporsante aléatoire ' varie, mais pas trop'. Cette condition se traduira par la « stationnarité », qui implique une certaine

régularité du processus et permet de dériver ses propriétés asymptotiques.
Dans toute la suite, on se place dans un cadre paramétrique : trouver un modèle pour les observations revient à déterminer les paramètres d'une fonction dont la forme est connue 'a priori' et qui vérifie

$$y_t = f(\epsilon_t, y_{t-1}, ...)$$

Où y_t représente l'observation à l'instant t et ϵ_t est le bruit aléatoire.
De puis les années 20(par exemple l'étude de **Yule** sur les taches solaires publiée en 1927) et jusqu 'aux années 80, les modèles linéaires à bruit gaussien AR ont tenu la tête d'affiche. Ils modélisent le présent par une combinaison linéaire d'un nombre fini de retards plus un bruit :

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + a_p y_{t-p} + \epsilon_t$$

Largement utilisés pendant ce temps. Cependant, de puis une vingtaine d'années, les limitations des modèles linéaires ont été soulignées. Les données réelles ayant souvent des caractéristiques non-linéaires qui ne sont pas prises en compte. Dans les séries financières, par exemple, on observe des périodes très fragiles, suivies de périodes plus stables. Ce phénomène appelé "volatility clustering".
Pour résoudre les problèmes évoqués, des modèles plus complexes ont été proposés à partir des années 80 : Les modèles à volatilité conditionnelle de type ARCH, GARCH [Engle (1982), Bollerslev (1986)], les modèles à seuil ou linéaires par morceaux (Tong provenant de l'itelligence artificielle comme les perceptrons multicouches qui ont la propriété pratique d'approximateurs universels [Hornit et alii (1989)].

Définition 2.1.1. *(serie temporelle)*
Une série temporelle (ou série chronologique) à temps discret est une suite réelle finie $(x_t)_{1\leq t\leq n}$ *,où t représente le temps (en minute, jour, année...).*

Voici quelques exemples de séries temporelles [3].

Exemple 2.1.1. *Nombre de passagers par mois (en milliers) dans les transports aériens, de 1949 à 1960 (la Figure(2.1)).*

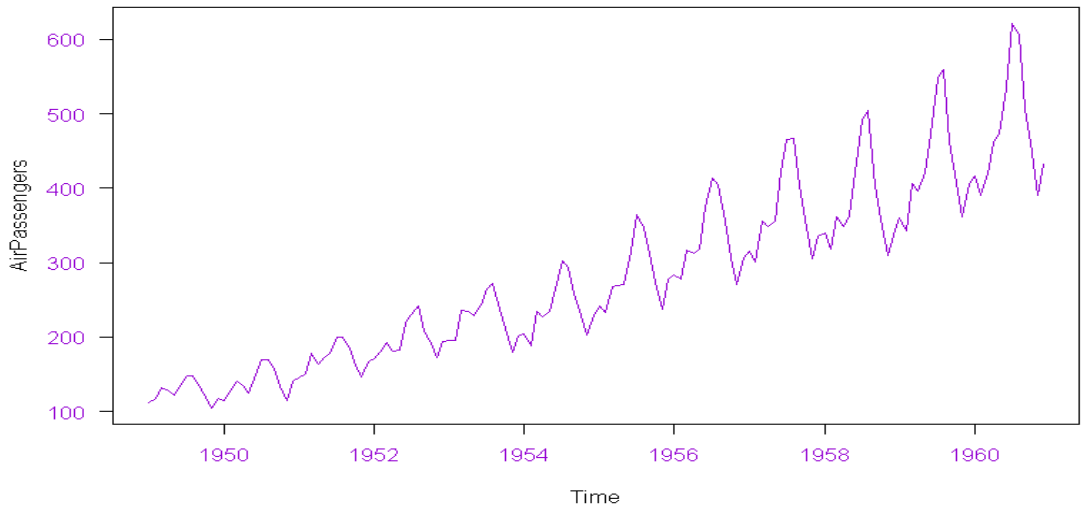


Fig. 2.1 – Nombre de passagers (en milliers) dans les transports aériens

Un des objectifs principaux de l'étude d'une série temporelle est la prévision des réalisations futures, très souvent pour des raisons économiques (prévoir l'évolution de la vente d'un produit pour ajuster au mieux les moyens de production, prévoir l'évolution d'un marché financier ...). Bien entendu, aucun modèle ne correspond exactement à la réalité, et il est impossible de prévoir parfaitement le devenir d'une série temporelle. Lorsque cela sera possible, nous donnerons des intervalles de prévisions, afin de pouvoir apporter une information quant à la précision de la prévision.
Pour ce faire, il existe un large choix de modèles utilisables :

- les modèles de régression, comme par exemple :

$$x_t = \alpha_1 t + \alpha_2 + \epsilon_t,\ t = 1, ..., n$$

  Une fois les coefficients de ce modèle estimés, la prévision de x_{t+1} sera

$$\hat{x}_{t+1} = \hat{\alpha}_1 (t+1) + \hat{\alpha}_2$$

  .
- les lissages exponentiels qui sont très simples à mettre en œuvre ;
- les modèles de type ARMA, qui consistent à enlever de la série les tendances et saisonnalités (ou périodicités) évidentes et à modéliser le résidu restant. Ces méthodes sont plus sophistiquées et plus lourdes numériquement (temps de calcul) que les précédentes, mais également plus performantes.

Parmi les exemples précédents , celui relatif au nombre de passagers dans les transports aériens (la Figure(2.1)) est une série assez typique de ce que l'on rencontre en économétrie, et elle donne lieu à de bonnes prévisions pour toutes les méthodes classiques.
Les défis que nous allons devoir relever sont les suivants :

- définir un modèle avec un nombre fini de paramètres ;
- estimer les paramètres de ce modèle ;
- vérifier la qualité d'ajustement du modèle, comparer différents modèles (partage de l'échantillon d'observations en 80% pour l'apprentissage et 20% pour le test) ;
- effectuer des prédictions.

Exemple 2.1.2. *Nombre annuel de captures de lynx pour 1821–1934 au Canada. Tirée de Brockwell & Davis (1991).*

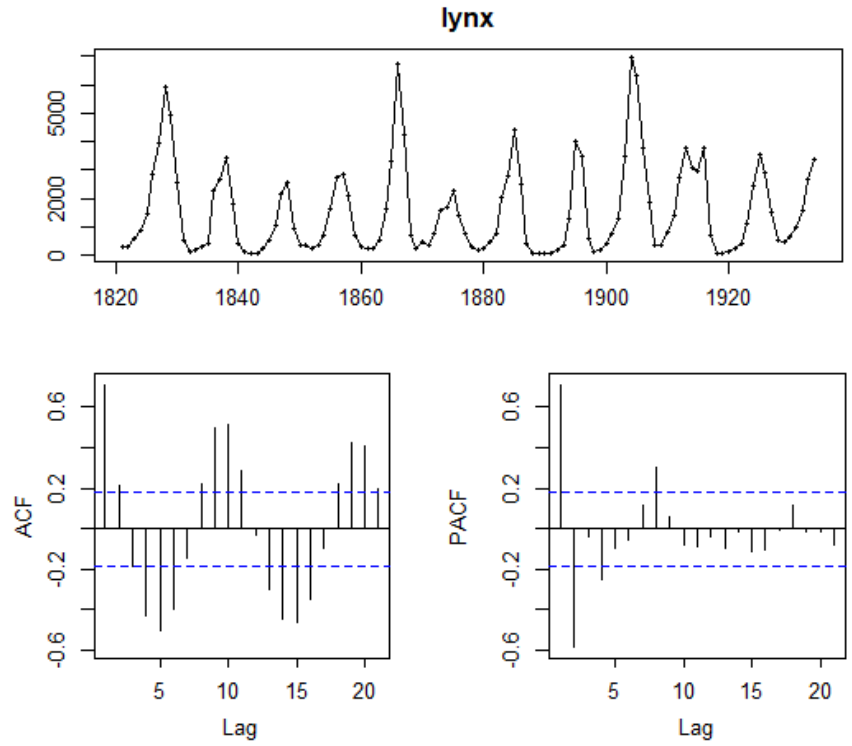


Fig. 2.2 – Les trois graphes de la séries lynx et de ACF, PACF

On a modéliser cette séries, par le modèle $ARIMA(4,0,0)$ *:*

```
Best model: ARIMA(4,0,0) with non-zero mean
Series: lynx
ARIMA(4,0,0) with non-zero mean
Coefficients:
ar1      ar2     ar3      ar4       mean
1.1246  -0.7174  0.2634  -0.2543  1547.3859
s.e.  0.0903   0.1367  0.1361   0.0897   136.8501
sigma^2 estimated as 748457:  log likelihood=-931.11
AIC=1874.22   AICc=1875.01   BIC=1890.64
```

Ce modèle peut s'écrire comme :

$$\begin{aligned} y_t &= 1547.39 + +\eta_t \\ \eta_t &= 1.12\eta_{t-1} - 0.72\eta_{t-2} + 0.26\eta_{t-3} + 0.14\eta_{t-4} + \epsilon_t \\ \epsilon_t &\sim \mathcal{N}(0,\, 0.07.10^7). \end{aligned}$$

Pour validé le modèle, en va étudier les résidus :

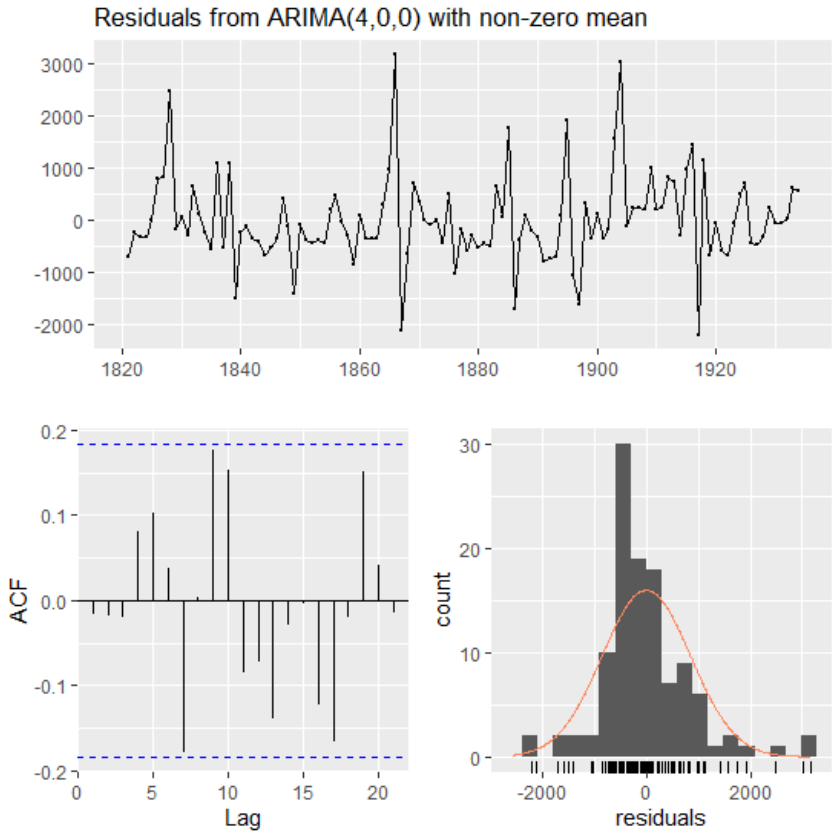


Fig. 2.3 – Graphiques des résidus pour le model AR(4)

```
checkresiduals(myarima)
Ljung-Box test
data:  Residuals from ARIMA(4,0,0) with non-zero mean
Q* = 13.201, df = 5, p-value = 0.02157

Model df: 5.   Total lags used: 10
```

$p - value = 0.02157$, cela suggère que la variation annuel de captures de lynx est essentiellement un montant aléatoire qui n'est pas corrélé avec celui des années précédents. Le modèle est validé, passant au prévision :

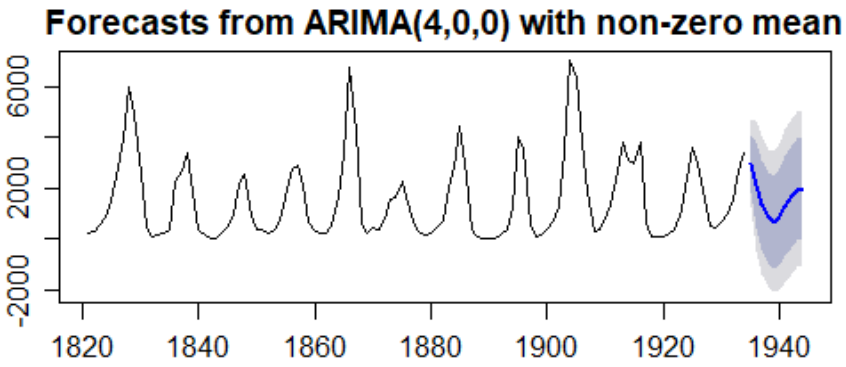


Fig. 2.4 – Prévision de 10 pts.

2.1.1 Tendances et composantes saisonnières

On parle de tendance lorsque la série $(x_t)_{1\leq t\leq n}$ peut s'écrire, à une erreur d'ajustement ε_t près, comme une combinaison linéaire de m fonctions du temps, choisies a priori (par exemple fonction puissance, exponentielle, logarithmique...) :

$$x_t = \sum_{j=1}^{m} \alpha_j f_j(t) + \varepsilon_t \quad 1 \leq t \leq n\,.$$

Lorsque $x_t = \alpha t + \beta + \varepsilon_t$ la tendance est linéaire ($m = 1$ et $f(t) = \alpha t + \beta$). Une tendance polynomiale se traduira par

$$x_t = \alpha_1 t^p + \alpha_2 t^{p-1} + \cdots + \alpha_{p+1} + \varepsilon_t\,.$$

On parle de composante périodique lorsque la série $x_t = \alpha t + \beta + \varepsilon_t$ peut se décomposer en :

$$x_t = s_t + \varepsilon_t \; 1 \leq t \leq n\ ,$$

où s_t est périodique, c'est-à-dire $s_{t+T} = s_t$, avec T la période (supposée entière). Lorsque la période est de *6 mois ou 1 an*, on parle généralement de composante saisonnière.
Il est fréquent qu'une série comporte à la fois une tendance et une composante périodique voir (la figure 2.1).

2.1.2 Indices descriptifs d'une série temporelle

Indices de tendances centrales

Considérons un ensemble d'observations $x_1, \ldots, x_n$.
Nous utilisons comme indicateur de la tendance centrale la moyenne :

$$\bar{x}_n = \frac{1}{n}\sum_{t=1}^{n} x_t.$$

Indices de dispersion

Nous utilisons comme indicateur de dispersion la variance empirique :

$$\hat{\sigma}_n(0) = \frac{1}{n}\sum_{1}^{n}(x_i - \bar{x}_n)^2.$$

Indices de dépendance

La fonction d'autocovariance empirique est donnée par :

$$\hat{\sigma}_n(h) = \frac{1}{n-h}\sum_{t=1}^{n-h}(x_t - \bar{x}_n)(x_{t-h} - \bar{x}_n),$$

et **la fonction d'autocorrélation empirique** est donnée par

$$\hat{\rho}_n(h) = \frac{\hat{\sigma}_n(h)}{\hat{\sigma}_n(0)}\,.$$

Ce sont les auto-corrélations empiriques que nous utiliserons pour caractériser la dépendance entre les variables[3].

2.2 Estimation et élimination de la tendance(et saisonnalité)

Une série temporelle $(x_t)_{1\leq t\leq n}$ est l'observation des n premières réalisations d'un processus stochastique $(X_t)_t$. C'est ce processus que l'on cherche désormais à modéliser. Pour cela, la démarche suivante doit être adoptée :

- représenter graphiquement la série afin de repérer les tendances et saisonnalités ;
- estimer et supprimer les tendances et saisonnalités (partie déterministe du processus stochastique) ;
- choisir un modèle pour les résidus (partie aléatoire du processus stochastique) et l'estimer ;
- prédire les réalisations futures à l'aide de ce modèle.

2.2.1 Bruit blanc

Définition 2.2.1. *Un processus de* ***bruit blanc****est une suite de variables aléatoires* $(X_t)_t$ *indépendantes, d'espérance et de variance constantes. Si l'espérance est nulle, le bruit blanc est* ***centré****, et si les variables aléatoires sont gaussiennes, le bruit blanc est* ***gaussien****.*

2.2.2 Processus stationnaire

Un processus aléatoire $(X_t)_t$ est dit **stationnaire** si et seulement si :

1) $E(X_t) = \mu \quad \forall\, t$,c'est-à-dire que la moyenne du processus est de μ, donc **constante** ;

2) $var(X_t) < \infty, \forall t$, c'est-à-dire que la variance est finie et invariable dans le temps ;

3) $cov(X_t, X_{t+h}) = E[(X_t - \mu)(X_{t+h} - \mu)] = \gamma(h) \quad \forall t$, la covariance est indépendante du temps.

Il apparaît, à partir de ces propriétés, qu'un processus de bruit blancε_t dans lequel les ε_t et de même loi $N(0, {\sigma_\varepsilon}^2)$ est stationnaire [4].

2.2.3 Elimination de la tendance et de la saisonnalité

Cette méthode permet de supprimer les tendances et saisonnalité d'une série temporelle sans les estimer.
Soit Δ_T l'opérateur qui associe $(X_t - X_{t-T})$ à (Xt) :

$$\Delta_T = X_t - X_{t-T}.$$

On note Δ l'opérateur Δ_1 , et Δ_T^k l'opérateur $\underbrace{\Delta_T \circ \cdots \circ \Delta_T}_{k\, fois}$.

Proposition 2.2.1. *Soit un processus admettant une tendance polynomiale d'ordre k :*

$$X_t = \underbrace{\sum_{j=0}^{k} a_j t^j}_{m_t} + \varepsilon_t \,.$$

Le processus ΔX_t *admet une tendance polynomiale d'ordre k-1.*

Ainsi, en appliquant k fois Δ, on élimine la tendance.

2.3 Modélisation des séries stationnaires

Après avoir examiné des outils d'exploration, nous abordons la modélisation des séries temporelles. On considère qu'une série temporelle observée $\{y_t,\ t = 1,\ldots,\ T\}$ est la réalisation de v.a. (variables aléatoires)$\{Yt,\ t = 1,\ldots,\ T\}$ qui forment elles-mêmes une portion d'un processus aléatoire $\{Y_t,\ t = \ldots,\ 0,1,2,\ldots\}$, c'est-à -dire *d'une série infinie de v.a.* Pour pouvoir faire de l'estimation (de moyenne, de variance,...), il faut que le phénomène ait une certaine stabilité. La notion de stationnarité, est la clef de l'analyse des séries temporelles[5].

2.3.1 Corrélation, fonction d'auto corrélation et fonction d'auto cor-rélation partielle

Fonction d'auto corrélation (ACF). Considérons une série (faiblement) stationnaire X_t. On est souvent intéressé par décrire la dépendance de X_t par rapport à son passé, notamment pour expliquer le niveau actuel de la série par le niveau à une date précédente. Si une dépendance est linéaire, elle est bien décrite par *le coefficient d'auto corrélation*. Par définition, le coefficient d'auto corrélation d'ordre l est :

$$\rho_l = \frac{Cov\left(X_t, X_{t-l}\right)}{\sqrt{V\left(X_t\right) V\left(X_{t-l}\right)}}$$

Mais, $V(X_{t-l}) = V(X_t) = \gamma_0$ et $E(X_t) = \mu$ donc :

$$\rho_l = \frac{Cov\left(X_t, X_{t-l}\right)}{V\left(X_t\right)} = \frac{E\left[\left(X_t - \mu\right)\left(X_{t-l} - \mu\right)\right]}{E\left(X_t - \mu\right)^2} = \frac{\gamma_l}{\gamma_0}$$

Fonction d'auto-corrélation partielle(PACF) de X_t, est définie par :

$$\begin{cases} \alpha\left(1\right) = corr\left(X_2, X_1\right) = \rho\left(1\right) \\ \alpha\left(k\right) = corr\left(X_{k+1} - P_{\overline{sp}\{1,X_2,\ldots,X_k\}} X_{k+1}, X_1 - P_{\overline{sp}\{1,X_2,\ldots,X_k\}} X_1\right) \end{cases}$$

où $P_{\overline{sp}\{1,X_2,\ldots,X_k\}}$ est la projection sur le sous-espace engendré par $1, X_2, \ldots, X_k$.
$\alpha(k) = \phi_{kk}$, $k \geq 1$ avec ϕ_{kk} tel que :

$$\begin{pmatrix} \rho_0 & \rho_1 & \cdots & \rho_{k-1} \\ \rho_1 & \rho_2 & \cdots & \rho_{k-2} \\ \vdots & \ldots & \ddots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \cdots & \rho_0 \end{pmatrix} \begin{pmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_k \end{pmatrix}$$

2.3.2 Les processus auto-régressifs $AR(p)$

Les processus auto-régressifs, construits à partir de l'idée que l'observation au temps t s'explique linéairement par les observations précédentes.

Définition 2.3.1. *On dit que (X_t) est un processus auto-régressif d'ordre p (centré) s'il s'écrit*

$$X_t = \varepsilon_t + \sum_{j=1}^{p} a_j X_{t-j}\,, \tag{2.1}$$

où ε_t est un bruit blanc centré de variance σ_2.

ε_t est l'**innovation** contenue dans le processus au temps t
Les coefficients a_j doivent vérifier la contrainte suivante pour assurer la stationnarité du processus :
le polynôme caractéristique du processus (2.1), $A(z) = 1 - a_1 z - ... - a_p z^p$, ne doit avoir que des racines (réelles ou complexes)de module strictement supérieur à 1.

2.3.3 Les processus en moyenne mobile $MA(q)$

Définition 2.3.2. *On appel **le moyenne mobile**(Moving Average) d'ordre q un processus de la forme*

$$X_t = \epsilon_t - \theta_1 \epsilon_{t-1} - ... - \theta_q \epsilon_{t-q},$$

où les ϵ_j pour $t - q \leq j \leq t$ sont des bruits blancs centrés de variance σ^2.

où $Xt = \theta(L)\epsilon_t$ On suppose bien évidement que q est inférieur au nombre d'observations. Autocorrélation de X_t :

2.3.4 Les processus mixtes $ARMA(p, q)$

Définition 2.3.3. *Un processus auto-régressif moyenne mobile d'ordres p et q est de le forme :*

$$X_t = \sum_{k=1}^{p} a_k X_{t-k} - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}\ .$$

où les ϵ_j pour $t - q \leq j \leq t$ sont des bruits blancs centrés de variance σ^2.

La stationnarité d'un $ARMA$
est assurée si *toutes les racines* du polynôme
$A(z) = 1 - a_1 z - \cdots - a_P z^p$, sont de module strictement supérieur à 1. Ce polynôme forme avec $B(z) = 1 + \theta_1 z + \cdots + \theta_q z^q$, les deux polynômes caractéristiques du processus. On supposera également que les polynômes A et B n'ont pas de racine commune, afin de s'assurer qu'il n'y a pas de représentation plus courte.
On peut écrire le processus $ARMA(p, q)$ sous la forme suivante :

$$X_t - a_1 X_{t-1} - \cdots - a_p X_{t-p} = \epsilon_t - \theta_1 \epsilon_{t-1} - \cdots - \theta_{t-q} \epsilon_{t-q},$$

où $\Phi(L)X_t = \Theta(L)\epsilon_t$
$\Phi(L)$ est l'opérateur de autorégressif
$\Theta(L)$ est l'opérateur de moyenne mobile

2.3.5 Les processus $ARIMA(p, d, q)$

Lorsque l'on a une série (X_t) à non stationnarité stochastique, il convient de la modéliser à l'aide d'un processus $ARIMA(p, d, q)$ où d *désigne l'ordre de différenciation (ou d'intégration).*

Définition 2.3.4. *Un processus $ARIMA(p, d, q)$ ou "**A**uto**r**egressive **I**ntegrated **M**oving **A**verage" d'ordre $p, d,$ et q pour la série (X_t) est un processus de la forme suivante :*

$$(1 - \phi_1 B - \cdots - \phi_p B^p) \nabla^d x_t = (1 - \theta_1 B - \cdots - \theta_q B^q) \varepsilon_t$$

ou encore

$$(1 - \phi_1 B - \cdots - \phi_p B^p) (1 - B)^d x_t = (1 - \theta_1 B - \cdots - \theta_q B^q) \varepsilon_t$$

où ε_t est un bruit blanc centré de variance σ^2,B est l'opérateur de retard tel que $Bx_t = x_{t-1}$ et $B^p x_t = x_{t-p}$,∇^d est l'opérateur de différence de degré d ($d \geq 0$ est un entier positif), $(\phi_1, \cdots, \phi_p)$ et $(\theta_1, \cdots, \theta_q)$ sont des coefficients à estimer.

La série (X_t) est une série *non stationnaire* alors que la série $w_t = \nabla^d x_t$ est une série stationnaire. Estimer les paramètres du processus $ARIMA(p, d, q)$ pour la série (X_t) *non stationnaire* revient à estimer les coefficients du processus $ARMA(p, q)$ pour la série (W_t) *stationnaire.*

2.3.6 Identification et estimation de modèles ARMA

En pratique, lorsque l'on doit ajuster un modèle AR, MA ou $ARMA$ à des données réelles la première question qui se pose est celle du choix des ordres p et q du modèle $ARMA$ (on considère que les AR et MA sont un cas particulier d'$ARMA$ avec respectivement q=*0* et p=0). Pour choisir ces ordres, nous pouvons exploiter les résultats suivants :

table 2.1 – Les propriétés des fonctions d'autocorrélation et d'autocorrélation partielle

Type de processus	Définition	autocorrélations	autocorrélations partielles
$AR(p)$	$\Phi(L)X_t = \epsilon_t$	$\rho(h) \searrow 0$	$r(h) = 0$ pour $h \geq p + 1$
$MA(q)$	$X_t = \Theta(L)\epsilon_t$	$\rho(h) = 0$ pour $h \geq q + 1$	$r(h)$ rien de particulier
$ARMA(p, q)$	$\Phi X_t = \Theta(L)\epsilon_t$	$\rho(h) \searrow 0$, $h \geq q + 1$	$r(h) \searrow 0$, $h \geq \max(q + 1, p)$

Si on trouve que les ACF et les $PACF$ ne sont pas significatifs pour tous les retards, *alors les résidus obéissent à un processus de bruit blanc* (voir [6]).
Pour un modèle AR(1) :

- quand $\phi_1 = 0, y_t$ est équivalent au bruit blanc ;
- quand $\phi_1 = 1$ et $c = 0, y_t$ est équivaut à une marche aléatoire ;
- quand $\phi_1 = 1$ and $c \neq 0, y_t$ est équivaut à une marche aléatoire avec dérive ;
- quand $\phi_1 < 0, y_t$ tend à osciller autour de la moyenn.

Nous restreignons normalement les modèles auto-régressifs aux données stationnaires, auquel cas certaines contraintes sur les valeurs des paramètres sont nécessaires.

- Pour un modèle AR(1) : $-1 < \phi_1 < 1$.
- Pour un modèle AR(2) : $-1 < \phi_2 < 1, \phi_1 + \phi_2 < 1, \phi_2 - \phi_1 < 1$.

Quand $p \geq 3$,les restrictions sont beaucoup plus compliquées. L'ogicielle (R, python,...) prend en compte ces restrictions lors de l'estimation d'un modèle.

- Pour un modèle *MA(1)* : $-1 < \theta_1 < 1$.
- Pour un modèle *MA(2)* :$-1 < \theta_2 < 1, \theta_2 + \theta_1 > -1, \theta_1 - \theta_2 < 1$.

Des conditions plus compliquées sont valables pour $q \geq 3$. Encore une fois, (R, python,...) prendra soin de ces contraintes lors de l'estimation des modèles.

2.4 Tests sur les résidus

2.4.1 Tests d'autocorrélation

Il existe un grand nombre de tests d'autocorrélation, les plus connus sont ceux de Box et Pierce (1970) et Ljung et Box (1978).
Soit une autocorrélation des erreurs d'ordre $K(K > 1)$:

$$\varepsilon_t = \sum_{i=1}^{K} \rho_i \varepsilon_{t-i} + \upsilon_t \text{ où } \upsilon_t \rightsquigarrow \mathcal{N}(0, \sigma_{\upsilon^2})$$

Les hypothèses du test de Box-Pierce sont les suivantes :

$$\begin{cases} H_0 : & \rho_1 = \rho_2 = \ldots = \rho_K = 0 \\ H_1 : & \textit{il existe au moins un } \rho_i \textit{ significativement diffrent de } 0. \end{cases}$$

Pour effectuer ce test, on a recours à la statistique Q qui est donnée par :

$$Q = n \sum_{i=1}^{K} \hat{\rho}_i^2$$

Sous l'hypothèse H_0 vraie, Q suit l*a loi du Khi-Deux* avec Kdegrés de liberté :
$Q \rightsquigarrow \chi^2(K)$ si $Q > k^*$ où k^* est la valeur donnée par *la table du Khi-Deux* pour un risque fixé et un nombre K de degrés de liberté. Alors , on rejette H_0 et on accepte H_1 *(autocorrélation des erreurs).*

2.4.2 Tests de normalité

Pour calculer des intervalles de confiance prévisionnels et aussi pour effectuer les tests de Student sur les paramètres, il convient de vérifier la normalité des erreurs. Le test de Jarque et Bara (1984), fondé sur la notoin de Skewness(asymétrie) et de Kurtosis(aplatissement), permet de vérifier la normalité d'une distribution statistique.

Les tests du Skewness et du Kurtoris

Soit $\mu_k = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^k$ le moment centré d'ordre k, le coefficient de Skewness $\left(\beta_1^{1/2}\right)$est égal à : $\left({\beta_1}^{1/2}\right) = \frac{\mu_3}{{\mu_2}^{3/2}}$ et le coefficient de Kurtoris :$\beta_2 = \frac{\mu_4}{\mu_2^2}$. Si la distribution est normale et le nombre d'observation grand(n>30) : $\beta_1^{1/2} \rightarrow N\left(0; \sqrt{\frac{6}{n}}\right)$ *et* $\beta_2 \rightarrow N\left(3; \sqrt{\frac{24}{n}}\right)$

On construit alors les statistiques :

$$\nu_1 = \frac{\left|\beta_1^{1/2} - 0\right|}{\sqrt{\frac{6}{n}}} \quad et \quad \nu_1 = \frac{|\beta_2 - 3|}{\sqrt{\frac{24}{n}}}$$

que l'on compare à 1,96(valeur de la loi normale au seuil de 5%). Si les hypothèses H_0 : $\nu_1 = 0$ *et* $\nu_2 = 0$ sont vérifiées, alors $\nu_1 \leq 1,96$ *et* $\nu_2 \leq 1,96$; dans le cas contraire, l'hypothèse de normalité est rejetée.

Le test de Jarque et Bara

Il s'agit d'un test qui synthétise les résultats précédents ; si $\beta_1^{1/2}$ *et* β_2 obéissent à des lois normales alors la quantité s : $s = \frac{n}{2}\beta_1 + \frac{n}{24}(\beta_2 - 3)^2$ suit un χ^2 à deux degrés de liberté.
Donc si $s > \chi^2_{1-\alpha}(2)$, on rejettel'hypothèse H_0 de normalité des résidus au seuil α.

Le Test de Kolmogorov-Smirnov

Le test de Kolmogorov-Smirnov est un test non paramétrique de conformité de deux populations régies par des variables quantitatives continues. Il est plus puissant que le test de χ^2 et ne nécessite pas de regroupement en classes ce qui le rend efficace dans le cas de petits échantillons. Le principe est de comparer les fréquences relatives cumulées (FRC) de l'échantillon E et celui, théorique, de la population P. La fonction descriminante est l'écart maximal observé (qui est aussi une distance entre deux probabilités) :

$$D_c = \max_{1 \leq i \leq n} |FRC_{obs}(x_i) - FRC_{th}(x_i)|$$

Les hypothèses relatives à un test bilatéral sont les suivantes :
$\begin{cases} H_0 : FRC_{obs} = FRC_{th} \quad pour\, tout\, x\, réel; \\ H_1 : FRC_{obs} \neq FRC_{th} \quad pour\, au\, moins\, une\, valeur\, de\, x\, réel. \end{cases}$
La distribution de la variable D est, sous l'hypothèse H_0 connue et tabulée.
Dès que $n > 40$, on peut considérer les valeurs seuils suivantes :

$$D_{5\%} = \frac{0.886}{\sqrt{n}} \quad D_{1\%} = \frac{1.031}{\sqrt{n}}$$

On rejetera H_0 dès que $D_c \leq D_s$.

2.4.3 Tests de stationnarité

Tests de Dickey-Fuller augmenté

Soit Y_t notre série initiale centrée. Le test de Dickey-Fuller augmenté(ADF,1981)consiste à effectuer la régression 2.2 et tester la signifficativité du coefficient $\rho - 1$ à l'aide des seuils de MacKinnon [7] . $D_t = \mu + \alpha t$ est la partie déterministe de la série, $\alpha\, et\, \mu$ pouvant être nuls. Selon les valeurs de $\alpha\, et\, \mu$, on obtient les différentes versions du test DFA du tableau2.2

$$\Delta Y_t = D_t + (\rho - 1)Y_{t-1} + \sum_{i=1}^{p-1} \psi_i \Delta Y_{t-i} + \varepsilon_t \tag{2.2}$$

DICKEY-FULLER augmenté : test de racine unitaire
H_0 : $\rho - 1 = 0$ dans la regression2.2 . Racine unitaire, non stastionnarité.
H_1 : $\rho - 1 < 0$, stationnarité de la série.
statistique de test : statistique de STUDENT du coefficient $(\rho - 1)$.
rejet de H_0 : seuils de Mac KINNON, tableau2.2.

table 2.2 – Différentes versions du test DFA.

version du test		valeur critique à 5%
Tendance+ constante	$\alpha \neq 0, \mu \neq 0$	-3.454
Constante	$\alpha = 0, \mu \neq 0$	-2.89
ϕ	$\alpha = 0, \mu = 0$	-1.944

Remarque 2.4.1. *Il faut aussi tester la normalité du résidu (au moins en dessinant un histogramme). Sans cette normalité , les statistiques de test sur les coefficients n'ont pas les lois asymptotiques habituelles et les intervalles de confiance sur les coefficients, appuyés sur le théorème de la limite centrale, sont très approximatifs.*

Cependant, on montre qu'un modèle AR(p) choisi par le critère AIC permet une prédicition avec une plus petite erreur $\hat{\sigma}_\varepsilon^2(p)$ (on dit que ce critèreest asymptotiquement éfficace : le rapport entre l'erreur de prédiction évaluée au minimum théorique et l'erreur se basant sur la valeur choisie par le critère AIC tend vers 1 si T tend vers l'infini) [8],[9].

2.4.4 Critères de choix des modèles

Après examen des coefficients et des résidus, certains modèles sont écartés. Pour départager les modèles restants, on fait appel aux critères standards et aux critères d'information.

Critères standards :

- L'erreur absolue moyenne (Mean Absolute Error) :

$$MAE = \frac{1}{n}\sum_t |e_t|$$

  où e_t est le résidu du modèle $ARMA$ étudié et n le nombre d'observation.

- l'erreur quadratique moyenne (Mean Squared Error) :

$$MSE = \frac{1}{n}\sum_1^n e_t^2.$$

- la racine carrée de l'erreur quadratique moyenne(Root Mean Square Error) :

$$RMSE = \sqrt{\frac{1}{n}\sum_1^n e_t^2}.$$

- Ecart absolu moyen en pourcentage (Mean Absolute Percent Error) :

$$MAPE = 100\frac{1}{n}\sum_{1}^{n}|\frac{e_t}{X_t}| \ .$$

Plus la valeur de ces critères est faible, plus le modèle estimé est proche des observations.
Critères d'information :

- Le critère d'Akaike : $AIC = \ln\sigma_\epsilon^2 + \frac{2(p+q)}{n}$
- Le critère de Schwarz : $SIC = \ln\sigma_\varepsilon^2 + (p+q)\frac{\ln(n)}{n}$
- Le critère d'information de Hannan - Quinn : $HQ = \ln\sigma_\varepsilon^2 + \alpha(p+q)\ln[\frac{\ln(n)}{n}]$

où $\alpha(>2)$ est une constante [10].

Conclusion

Dans ce chapitre, nous avons d'abord introduit l'intégration des modèles *ARIMA* pour la prévision des séries temporelles. En résumé la démarche empirique pour effectuer une modélisation *ARIMA(p,d,q)* est la suivante :

1. Stationnariser la série (régression, différenciation, transformation), vérifier avec l'autocorrélogramme.
2. Déterminer des ordres p et q plausibles à l'aide de respectivement l'autocorrélogramme partiel et l'autocorrélogramme.
3. Estimer les paramètres, si plusieurs modèles candidats les départager par l'AIC ou le BIC.
4. Valider ou non le modèle par un diagnostique des résidus (test, représentation graphique, autocorrélogramme).
5. Confirmer votre choix en simulant de la prévision (échantillon test).

Dans le cas de modèles SARIMA[(p,d,q) ;(P,D,Q)] :

1. Identifier la saisonnalité s (autocorrélogramme, spectrogramme)
2. Stationnariser la série, en commençant par D puis d
3. Déterminer des ordres p et q plausibles à l'aide de l'autocorrélogramme partiel et l'autocorrélogramme. Les ordres P et Q en regardant les ordre mulitples de s de ces autocorrélogrammes.
4. Estimer les paramètres, si plusieurs modèles candidats les départager par l'AIC ou le BIC.
5. Valider ou non le modèle par un diagnostique des résidus (test, représentation graphique, autocorrélogramme).
6. Confirmer votre choix en simulant de la prévision (échantillon test).

CHAPITRE 3

Les Réseaux de neurones récurrents

Dans ce chapitre nous nous pencherons sur les méthodes les plus connues conçues pour la prévision des séries temporelles. Cette étude comportera des modèles et des algorithmes d'apprentissage provenant de différentes familles. On s'intéressa plus particulièrement aux réseaux de neurones artificiels, en particulier les *LSTMs*. Nous verrons qu'il existe de nombreuses optimisations et adaptations de ces méthodes.

Ces dernières années ont vu la résurgence de l'utilisation des réseaux de neurones récurrents pour le traitement de séquences, et en particulier dans des tâches prédictives. Les réseaux de neurones récurrents disposent d'une mémoire sur ce qui a été calculé par le passé et c'est ce qui les rend particulièrement adaptés au traitement de séquences. En théorie, les réseaux de neurones récurrents peuvent garder en mémoire l'information vue dans une séquence arbitrairement grande mais en pratique perdent de leur efficacité sur des dynamiques à très long terme. C'est dans ce sens que de récents travaux ont vu l'émergence d'architectures de réseaux de neurones récurrents disposant de mécanismes de "gate" permettant d'améliorer considérablement les capacités de mémorisation des modèles. Plus spécifiquement, certains types de réseaux récurrents, les LSTMs [11] et les Prophets [30] se sont avérés particulièrement efficaces pour modéliser des séquences dont la dynamique pouvait s'étaler loin dans le temps ; ce n'est pas le cas de l'approche autorégressive qui de fait, au moment de l'inférence ne se base que sur une fenêtre temporelle de taille fixe dans le passé.
Finalement, en apprentissage automatique, l'essentiel des progrès réalisés ces dernières années en modélisation de séquence a été avec des architectures récurrentes à "gate" voir [12] . Dans cette section, nous allons passer rapidement en revue les principes de base des réseaux de neurones.

3.1 Principes des Réseaux de Neurones Artificiels

Le neurone formel est l'entité de base qui compose un réseau de neurones. Ce paragraphe présente le modèle de neurones le plus souvent utilisé et qui sera considéré dans le reste de ce travail. Mathématiquement un neurone peut être représenté par une

fonction à plusieurs variables avec une sortie unique. La figure 3.1 représente le schéma classique d'un neurone formel. Les valeurs $x_1, \ldots, x_n$ représentent les entrées du neurone i. Les valeurs w_{ij} représentent les poids associés à chaque entrée x_j avec $j \in \{1, \ldots, n\}$.

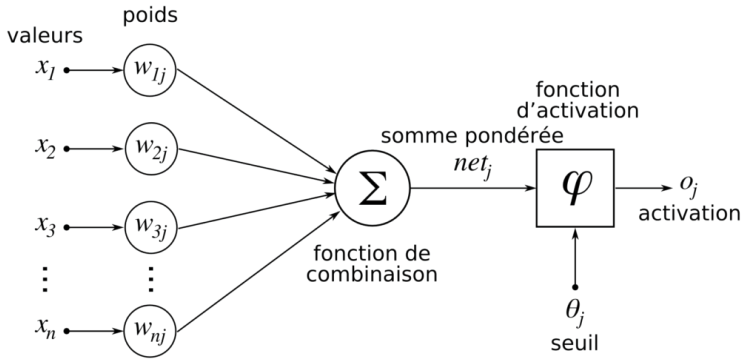


Fig. 3.1 – architecture d'un neurone formel

Un neurone peut être également caractérisé par quatre aspects :

— La nature des entrées qui peut être soit binaire soit réelle ; dans notre cas ce sont toujours des réels.

— La fonction de sommation des entrées qui peut être linéaire ou non linéaire. En pratique, ainsi que dans nos expérimentations, c'est une combinaison pondérée par les poids w_{ij} des entrées qui est utilisée.

— La nature de la fonction de transfert utilisée (linéaire ou non-linéaire). En pratique, on trouve des fonctions à seuil (équation (3.1)) ou des fonctions sigmoïdes (équation (3.2)) ou encore des fonctions gaussiennes (équation (3.3)).

— La nature de la sortie (binaire ou réelle). Dans notre cas, elle sera toujours réelle.

Fonction à seuil :

$$f(x) = \begin{cases} 1 & six \geq a \\ -1 & sinon \end{cases} \tag{3.1}$$

Fonction sigmoide :

$$f(x) = \frac{1}{1 + e^{(-ax)}} \tag{3.2}$$

Fonction gaussienne :

$$f(x) = e^{(\frac{-ax^2}{2})} \tag{3.3}$$

Nous pouvons remarquer que la fonction de transfert sigmoïde possède un régime quasilinéaire au voisinage de 0 et nonlinéaire ailleurs. Cette particularité offre au réseau la possibilité de s'adapter aussi bien aux problèmes linéaires qu'aux problèmes non linéaires.

3.2 Les Réseaux de Neurones à Propagation Avant

La nécessité d'utiliser des FFNN non linéaires découle naturellement des limites des méthodes statistiques linéaires pour la modélisation et la prédiction de séquences non linéaires. Même le remplacement de la fonction de transfert du neurone de sortie par une

fonction non linéaire ne peut effectivement dépasser cette limite. Un système entièrement non linéaire est obtenu en ajoutant au moins une couche intermédiaire, généralement appelée couche cachée.

Dans ce paragraphe, nous détaillerons principalement deux types de réseaux à propagation avant : le Perceptron Multi Couche, en anglais Multi-Layer Perceptron (MLP), et les Réseaux à Fonction à Base Radiale, en anglais Radial Basis Function Networks (RBFN) .

3.2.1 Le perceptron multi couche

Dans e paragraple, nous allons détailler le principe des réscaux à concles en insistant sur les avantages qu'ils pussedent par rapport à d'autres architectures. Nous donnerons également leur représentation mathématique.

Définition 3.2.1. ***(Architecture)*** *Les réseaux à couches (figure 3.2), appelés également Perceptrons Muiti-Couches (PMC) sont des réseaux de neurones pour les quelles les neurones sont organisés en couches successives. L'information, provenant des entrées, circule vers les sorties sais retour en arrière et, toute connexion entre neurones d'une même couche est interdite. Un neurone ne peut donc transmettre son état qu'aux neurones appartenant à une couche postérieure à la sienne. La couche recevant l'information à traiter de l'environnement extérieur est appelée couche d'entrée. La couche de sortie est composée d'autant de neurones que le nombre de sorties du réseau. Les couches constituées des neurones effectuant des calculs intermédiaires sont appelées couches cachées.*

Choisir ainsi l'architecture d'un PMC consiste a fixer le nombre de couches, le nombre de neurones par couche, la nature des différentes connexions entre les neurones et la fonction d'activation des neurones dans chaque couche.

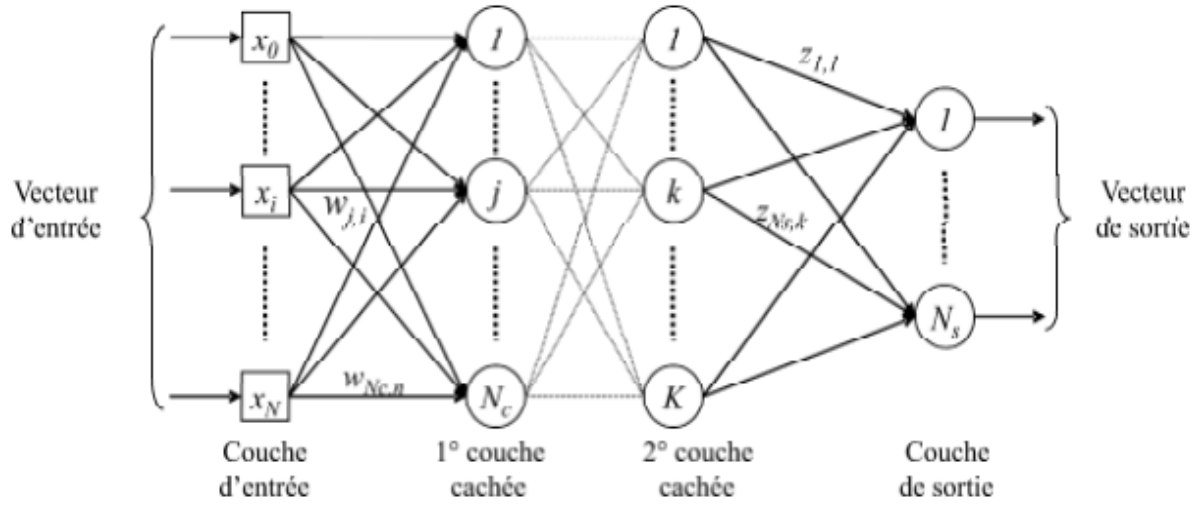


Fig. 3.2 – Structure d'un réseaux de neurones de type PMC à deux couches cachées.

Propriété 3.2.1. ***(Approximation universelle)*** *La propriété d'approximation universetle a été démontrée par Cybenko [13] et Funuhashi [14].*

et peut s'énoncer de la façon suivante : Toute fonction dornée suffisamment régulière peut etre approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un reseau de neurones comportant une couchke de neurones cuchés en nombre fini, possédant tous la mene fonction d'activation, et un neurone de sortie linénire.

Remarque 3.2.1. *Cette propriété s'applique pour les réscaux préscutés au paragraplhe précécient, mais aussi pour les fonetions ondelettes et radiales on la sortie d'un neurone j peut s'écrire :*

$$s_j = \exp\left[\frac{\sum_{i=1}^{n}(x_i - w_{j,i)^2}}{2w_{j,0}^2}\right]. \tag{3.4}$$

avec w_i sont les coordonnées du centre de la gaussienne et $w_{j,0}$ son écart type.

Cette propriété justifie notre choix des réscaux a une seule couche cachée dans la suite de notre travail. Le seui degré de liberté qui subsiste pour déterminer l'architecture du réseau est le nombre de neurones cachés. Ce nombre doit être choisi convenablement selon la nature du problème pour obtenir la précision souhaitée.

Architecture de base

La figure 3.3 montre la composition générique d'un perceptron multi couche, en anglais Multi-Layer Perceptron (MLP). Dans ce type de réseaux, les neurones sont organisés de manière à constituer plusieurs couches. En pratique, le nombre de couches varie de 3 à 4 dont une couche d'entrée, une couche de sortie et une ou deux couches cachées.

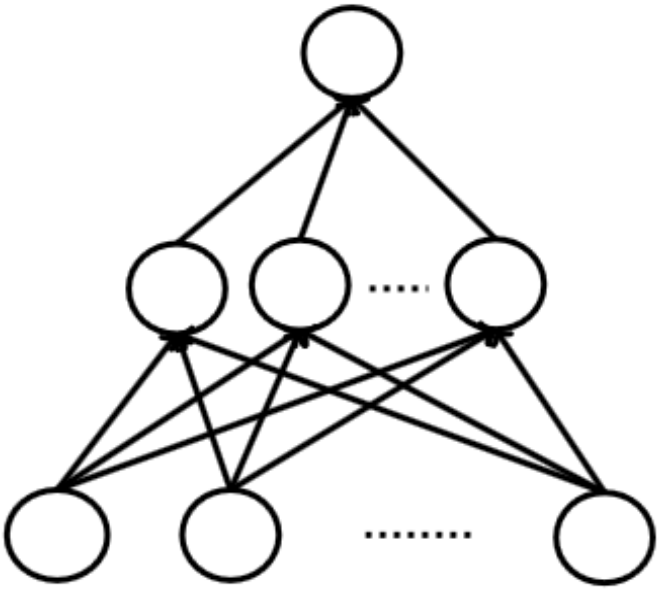


Fig. 3.3 – Architecture générique d'un MLP

La couche d'entrée contient autant de neurones que la dimension de l'espace d'entrée. La couche de sortie contient un neurone unique pour la valeur de la prévision. Enfin, une ou plusieurs couches cachées contiennent un nombre variable de neurones.

Caractéristiques de base

Fonctionnellement, comme les modèles linéaires, les MLP effectuent de nombreuses opérations algébriques. Le MLP effectue une composition de fonction sur l'entrée en raison de la fonction de transfert des neurones du réseau. Par conséquent, la sortie du réseau dépendra de l'entrée de courant et de la structure des connexions. La seule mémoire dont dispose un MLP est la mémoire de l'exemple passé en entrée. On appelle la fenêtre temporelle de dimension d un vecteur colonne, noté X(t) :

$$X(t) = t(x(t), x(t-1), ..., x(t-d)) \tag{3.5}$$

Il est mathématiquement démontré que les MLP possèdent un pourvoir d'approximation universelle (voir [13]). Autrement dit, lorsque les données sont normalisées cest-à-dire que la fonction objectif est comprise dans la boule unité, un MLP possédant une seule couche cachée est capable d'approximer n'importe quelle fonction mathématique. Il suffit d'avoir le nombre de neurones cachés suffisant. Cette caractéristique peut être suffisante pour expliquer le grand nombre de MLP à une seule couche cachée utilisée dans la littérature. Mais une autre explication est liée à de la nature de l'algorithme de rétropropagation du gradient utilisé. En effet, la valeur du gradient tend à disparaitre assez rapidement au fur et à mesure que l'on avance dans des couches profondes, rendant l'algorithme d'apprentissage inefficace ou trop lent [15].

3.3 Résoudre le problème de dépendance à long terme

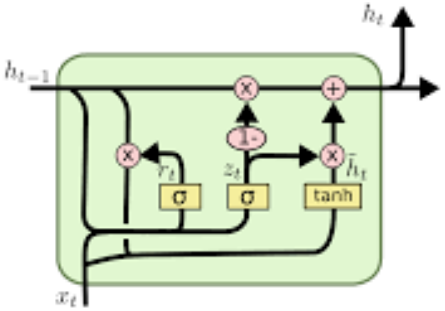
Nous avons vu dans la section précédente qu'il est difficile pour les RNN vanille d'apprendre efficacement les dépendances à longue portée en raison de la disparition et de l'explosion des gradients. Pour résoudre ce problème, le réseau Long Short Term Memory a été développé en 1997 par Sepp Hochreiter et Jürgen Schmidhuber[16]. Gated Recurrent Unit a été introduit en 2014 et donne une version plus simple de LSTM. Voyons comment *LSTM* et *GRU* résolvent le problème de l'apprentissage des dépendances à longue portée.

3.3.1 Mémoire à long terme *LSTM*

LSTM introduit des calculs supplémentaires à chaque pas de temps. Cependant, il peut toujours être traité comme une unité de boîte noire qui, pour le pas de temps (h_t), renvoie l'état interne (f_t) et celui-ci est transmis au pas de temps suivant. Cependant, en interne, ces vecteurs sont calculés différemment. LSTM introduit trois nouvelles portes : les portes d'entrée (input o_t) , d'oubli (forget g_t) , et de sortie (output c_t) . Chaque pas de temps a également un état caché interne (h_t) et une mémoire interne $\left(\sigma\left(W^h x_t + U^h s_{t-1}\right)\right)$. Ces nouvelles unités sont calculées comme suit :

$$\begin{aligned}
f_t &= \sigma\left(W^f x_t + U^f S_{t-1}\right) \\
o &= \sigma\left(W^o x_t + U^o s_{t-1}\right) \\
g_t &= \tanh\left(W^{g_{x_t} + U^{g S_{t-1}}}\right) \\
c_t &= f_t \odot c_{t-1} + h_t \odot g_t \\
s_t &= \tanh\left(c_t\right) \odot o_t \\
h_t &= f_t
\end{aligned}$$

Maintenant, comprenons les calculs. Les portes $o_t, \sigma(\cdot)$, et $[0, 1]$ sont générées par activations *sigmoïdes* (h_t) qui limitent leurs valeurs dans f_t . Par conséquent, ceux-ci agissent comme des portes en laissant ne produit qu'une fraction de la valeur lorsqu'elle est multipliée par une autre variable. La porte d'entrée (o_t) contrôle la fraction de l'entrée nouvellement calculée à conserver. La porte oubliée (g_t) détermine la effet du pas de temps précédent et la porte de sortie (c_t) contrôle la quantité de état de laisser sortir. L'état masqué interne est calculé de l'entrée au pas de temps actuel et la sortie du pas de temps précédent. Notez que c'est la même chose que le calcul de l'interne état dans un *RNN* vanille. h_t est l'unité de mémoire interne du pas de temps courant et considère la mémoire de l'étape précédente mais réduite d'une fraction S_t et l'effet de la état caché interne mais mélangé avec la porte d'entrée (o_t). Enfin, Z_t serait passé à la prochain pas de temps et calculé à partir de la mémoire interne actuelle et de la porte de sortie. Le les portes d'entrée, d'oubli et de sortie sont utilisées pour inclure sélectivement la mémoire précédente et la état caché actuel qui est calculé de la même manière que dans les *RNN* vanille. Le portail Le mécanisme de *LSTM* permet le transfert de mémoire à partir de pas de temps à longue portée.



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

Fig. 3.4 – Lstm architecture

3.3.2 Unités récurrentes fermées (Gated Recurrent Units)

GRU est plus simple que *LSTM* et n'a que deux portes internes, à savoir la porte de mise à jour (z_t) et la porte de réinitialisation (r_t). Les calculs des portes de mise à jour et de réinitialisation sont les suivants :

$$z_t = \sigma\left(W^z X_t + U^z s_{t-1}\right)$$
$$r_t = \sigma\left(W^r x_t + U^r s_{t-1}\right)$$

L'état (s_t) du pas de temps (t) est calculé à l'aide de l'entrée (x_t), de l'état (s_{t-1}) du pas de temps précédent, de la mise à jour et des portes de réinitialisation :

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \tanh\left(\left(W^h x_t + U^h\left(r_t \circ s_{t-1}\right)\right)\right.$$

La mise à jour calculée par une fonction *sigmoïde* détermine la quantité de mémoire de l'étape précédente à conserver dans l'étape actuelle. La porte de réinitialisation contrôle la façon de combiner la mémoire précédente avec l'entrée du pas en cours.

LSTM et *GRU* sont tous deux capables de gérer la mémoire sur de longs *RNN*. Cependant, une question courante est laquelle utiliser ? *LSTM* a longtemps été préféré comme

premier choix pour les modèles de langage, ce qui ressort de leur utilisation intensive dans la traduction de la langue, la génération de texte et la classification des sentiments. *GRU* a l'avantage distinct de moins de poids entraînables par rapport à *LSTM*. Il a été appliqué à des tâches où *LSTM* dominait auparavant. Cependant, des études empiriques montrent qu'aucune des deux approches ne surpasse l'autre dans toutes les tâches. Le réglage des hyperparamètres du modèle, tels que la dimensionnalité des unités cachées, améliore les prédictions des deux. Une règle générale consiste à utiliser *GRU* dans les cas où moins de données d'entraînement car il nécessite moins de poids entraînables. *LSTM* s'avère efficace dans le cas de grands ensembles de données tels que ceux utilisés pour développer des modèles de traduction linguistique.

3.3.3 Performances et temps d'apprentissage d'un LSTM

Compte tenu de la structure plus complexe, en particulier la présence de 3 matrices de poids (Forget gate, Input gate et poids de calcul de la mémoire), il est logique que la phase d'apprentissage des *LSTM* requiert plus de temps que celle d'un réseau de neurones conventionnel ou de RNN. Mais, un *LSTM* donne de bien meilleures performances. Dans les faits, la plupart des résultats intéressants obtenus aujourd'hui par les réseaux récurrents sont en fait obtenus par des *LSTM*, en raison de leur capacité à gérer la dépendance à long terme.

Remarque 3.3.1. *Ces réseaux de neurones récurrents LSTM à large « mémoire court-terme » ont révolutionné la reconnaissance de la voix (Speech Recognition), la compréhension et la génération de texte (Natural Langage Processing) ou la prédiction sur des séries numériques de données temporelles. Il est clair que pour la prédiction de séries temporelles, les LSTM donnent de bien meilleurs résultats que les réseaux de neurones traditionnels qui ne sont pas adaptés à ce type de tâche. Le temps de calcul en phase d'apprentissage est cependant nettement augmenté avec les LSTM du fait d'un nombre de poids du réseau fortement augmenté. A noter que de nouvelles technologies, les Transformers, basés sur le mécanismes d'attention, pourraient à termes, prendre une place de plus en plus importante pour le traitement des séries temporelles.*

3.3.4 Conclusion

Dans ce chapitre, nous avons décrit deux approches basées sur l'apprentissage profond pour développer des modèles de prévision de séries temporelles. Les réseaux de neurones conviennent dans les cas où il y a peu d'informations sur les propriétés sous-jacentes telles que la tendance à long terme et la saisonnalité ou celles-ci sont trop complexes pour être modélisées avec un degré de précision acceptable par les méthodes statistiques traditionnelles. Différentes architectures de réseaux neuronaux telles que *MLP*, *RNN* extraire des modèles complexes à partir des données. Si les modèles de réseaux neuronaux sont entraînés avec des mesures appropriées pour éviter le sur-ajustement des données d'entraînement, ces modèles se généralisent bien sur des données de validation ou de test invisibles. Pour éviter le sur-ajustement, nous avons appliqué le décrochage, qui est largement utilisé dans les réseaux de neurones profonds pour une variété d'ensembles de données et d'applications.

CHAPITRE 4

Applications

Dans ce chapitre, nous présentons deux applications, pour la première nous avons comparer le modèle classique en utilisant la méthodologie de Box-Jenkins (ARIMA modèle),et le modèle *LSTM*, pour la deuxième application nous avons hybrider les deux modèles pour bénéficier de la performance des deux modèles. Nous avons utilisé le puissant logiciel **python3.7**, voir ce lien (cliquer sur l'image) :

https://www.anaconda.com/products/distribution

En particulier nous avons travailler sur **jupyter notebook(anconda3)** ; **Jupyter** est un notebook de calcul (computational notebook) open source, gratuit et interactif. C'est une application web basée client permettant de créer et de partager du code, des équations, des visualisations ou du texte, voir :

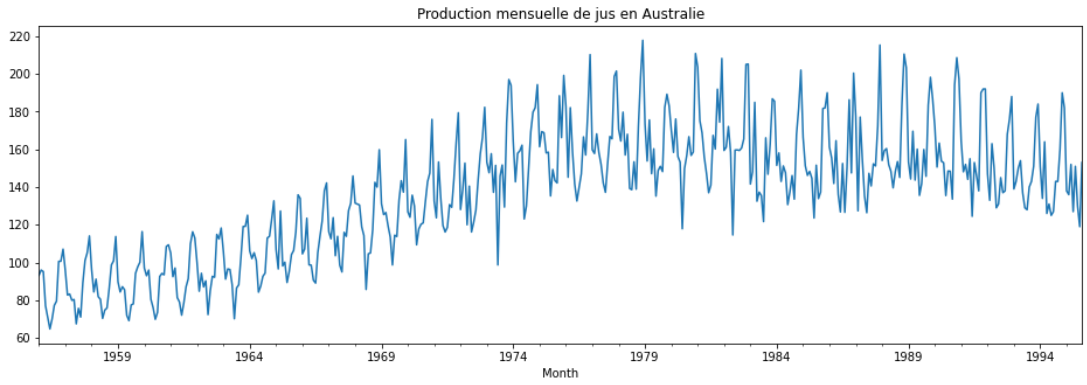
https://www.anaconda.com/products/distribution

L'objectif des expérimentations est d'évaluer la pertinence des deux méthodes puis Dans un premier temps, nous décrivons les jeux de données utilisés, puis le montage expérimental et enfin les résultats obtenus.

4.1 Première application

Nous allons comparer la qualité de prévision de production mensuelle de jus en Australie entre (1956/01/01 et 1995/08/01) ; en utilisant les deux modèles : $ARIMA$ et $LSTM$:

Fig. 4.1 – Production mensuelle de jus en Australie



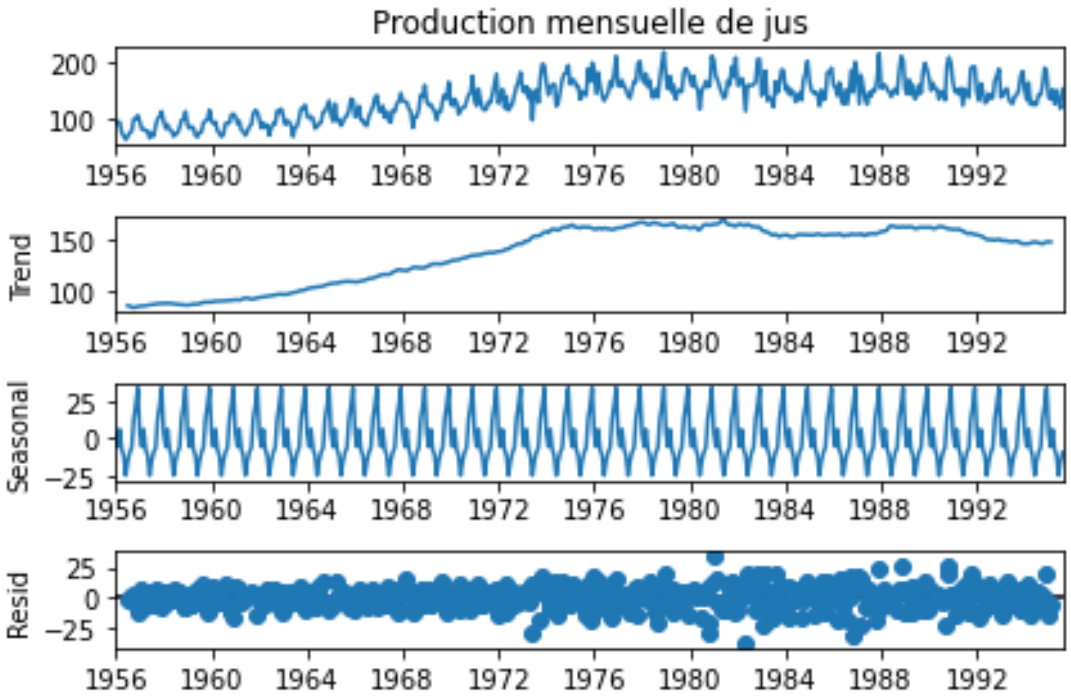
Lorsque nous regardons le graphique de cette séries, nous pouvons voir qu'il y a une saisonnalité dans les données. C'est pourquoi nous utiliserons *SARIMA (ARIMA saisonnier)* au lieu d'$ARIMA$.

ARIMA saisonnier est une extension d'*ARIMA* qui prend explicitement en charge les données de séries chronologiques uni-variées avec une composante saisonnière. Il ajoute trois nouveaux hyperparamètres pour spécifier l'autorégression (AR), la différenciation (I) et la moyenne mobile (MA) pour la composante saisonnière de la série, ainsi qu'un paramètre supplémentaire pour la période de la saisonnalité.
Il existe quatre éléments saisonniers qui ne font pas partie d'ARIMA et qui doivent être configurés ; ils sont :

- P : ordre auto-régressif saisonnier ;
- D : Ordre des différences saisonnières ;
- Q : Ordre moyen mobile saisonnier ;
- m : Le nombre de pas de temps pour une seule période saisonnière.

Décomposer :
Dans cette approche, la tendance et la saisonnalité sont modélisées séparément et la partie restante de la série est renvoyée :

Fig. 4.2 – Décomposition saisonnale de la série.



Ici, nous pouvons voir que la tendance, la saisonnalité sont séparées des données et nous pouvons modéliser les résidus.
Prévisions *SARIMA* :
Exécutons la fonction *auto_arima*() pour obtenir les meilleures valeurs *p,d,q,P,D,Q*, En obtient :

SARIMAX

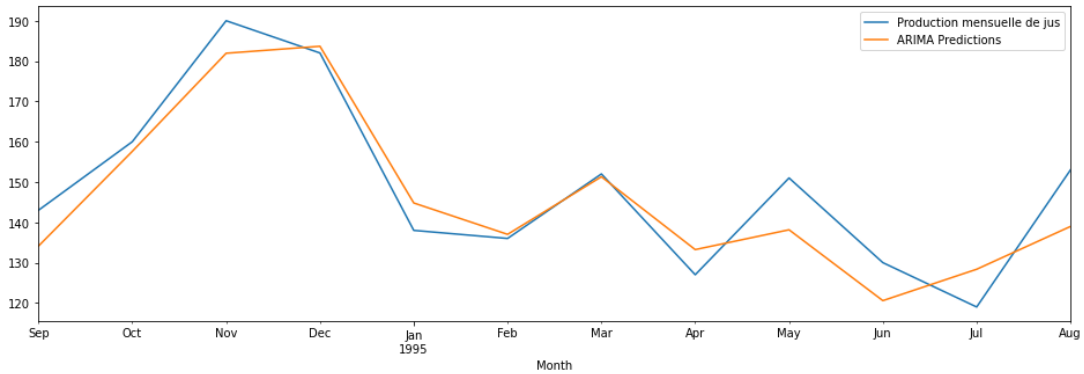
Dep. Variable :	Production mensuelle de jus	No. Observations :	464
Model :	SARIMAX(2, 1, 1)x(4, 0, [1, 2, 3], 12)	Log Likelihood	-1707.822
Date :	Sat, 16 Apr 2022	AIC	3437.644
Time :	13 :04 :29	BIC	3483.159
Sample :	01-01-1956	HQIC	HQIC
	- 08-01-1994		
Covariance Type :	opg		

	$coef$	$stderr$	z	$P > \|z\|$	[0.025	0.975]
$ar.L1$	-0.1184	0.041	-2.861	0.004	-0.200	-0.037
$ar.L2$	-0.1671	0.046	-3.628	0.000	-0.257	-0.077
$ma.L1$	-0.8493	0.027	-31.403	0.000	-0.902	-0.796
$ar.S.L12$	1.7597	0.096	18.249	0.000	1.571	1.949
$ar.S.L24$	-1.6245	0.177	-9.165	0.000	-1.972	-1.277
$ar.S.L36$	0.8024	0.131	6.108	0.000	0.545	1.060
$ar.S.L48$	0.0612	0.046	1.329	0.184	-0.029	0.151
$ma.S.L12$	-1.5589	0.113	-13.851	0.000	-1.779	-1.338
$ma.S.L24$	1.4056	0.182	7.725	0.000	1.049	1.762
$ma.S.L36$	-0.6737	0.104	-6.508	0.000	-0.877	-0.471
$sigma2$	84.1171	4.886	17.216	0.000	74.541	93.694
Ljung-Box (L1) (Q) :		02	Jarque-Bera (JB) :			41.58
Prob(Q) :		0.88	Prob(JB) :			0.00
Heteroskedasticity (H) :		4.08	Skew :			-0.30
Prob(H) (two-sided) :		0.00	Kurtosis :			4.34

Warnings :
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Comme nous pouvons le voir, le meilleur modèle arima choisi par $auto\_arima()$ est $SARIMAX(2,1,1)x(4,0,3,12)$ Séparons les données en ensemble d'entraînement et de test, pour voir la performance du modèle ; pour tester le modèle en trace le graphe suivant :

Fig. 4.3 – Sarima vs actuelles data



Les erreurs du test de modèle $SARIMA(2,1,1)x(4,0,3,12)$:

MSE Error :	64.26465003119232
RMSE Error :	8.016523562691768
Mean :	136.39537815126045

LSTM Forecast :

```
lstm_model.summary()
Model: "sequential"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200)               161600

dense (Dense)                (None, 1)                 201

=================================================================
Total params: 161,801
Trainable params: 161,801
Non-trainable params: 0
_________________________________________________________________
```

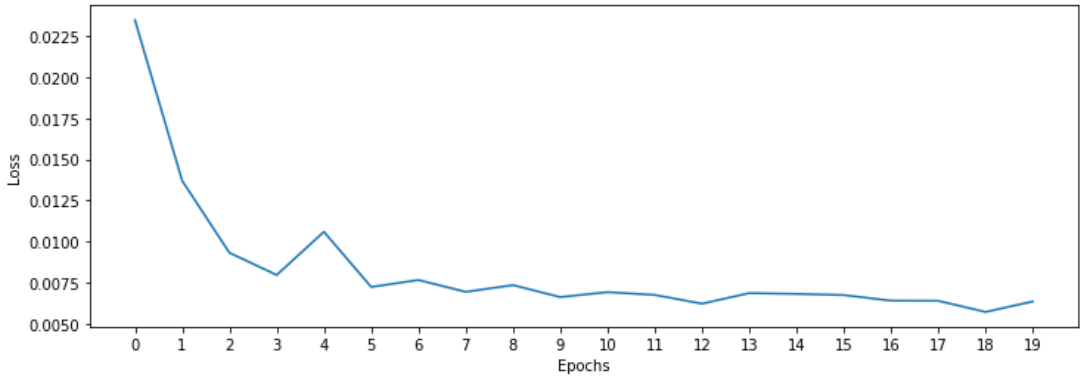


Fig. 4.4 – lstm_model.history : loss(perte).

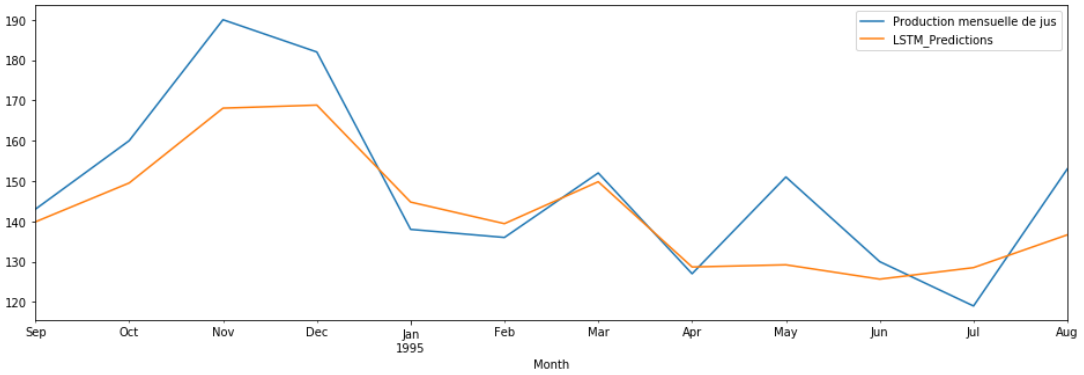


Fig. 4.5 – LSTM vs actuelles data

Pour comparé les deux méthodes , voir le tableau 4.1 ci-dessous

table 4.1 – ARIMA vs LSTM Prédiction

Month	Production mensuelle de jus	ARIMA_Predictions	LSTM_Predictions
1994-09-01	143.0	134.126256	139.863850
1994-10-01	160.0	157.500197	149.512329
1994-11-01	190.0	181.952293	168.067998
1994-12-01	182.0	183.676790	168.822054
1995-01-01	138.0	144.716191	144.772883
1995-02-01	136.0	137.097364	139.419548
1995-03-01	152.0	151.263858	149.818280
1995-04-01	127.0	133.318011	128.673290
1995-05-01	151.0	138.304727	129.214232
1995-06-01	130.0	120.557719	125.662228
1995-07-01	119.0	128.290150	128.515422
1995-08-01	153.0	139.028525	136.651676

Les erreurs du test de modèle LSTM :

MSE Error :	140.90580161735517
RMSE Error :	11.870374956898168
Mean :	136.39537815126045

le graphique suivant résume la performance des deux modèles ; *ARIMA* et *LTSM* :

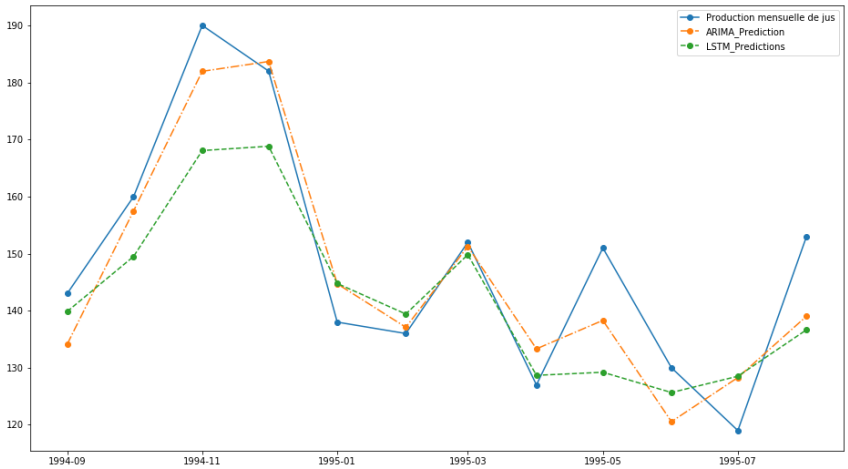


Fig. 4.6 – Comparaison entre ARIMA et LTSM modèle

	Models	RMSE Errors	MSE Errors
0	ARIMA	8.016524	64.264650
1	LSTM	11.870375	140.905802

table 4.2 – ARIMA vs LSTM

Nous avons vu beaucoup de choses différentes en commençant par différents concepts de modèles statistiques et de manipulations de séries temporelles (différentes transformations, stationnarité dans les séries temporelles, fonctions d'autocorrélation). Suivi d'un modèle statistique (ARIMA) et de réseaux de neurones récurrents (LSTM) appliqués à notre ensemble de données particulier, voir le tableau (4.2).

4.2 Deuxième application

La météo est un processus continu, gourmand en données, multidimensionnel, dynamique et chaotique et ces propriétés font de la prévision météorologique un défi de taille. L'objectif ici est de construire une approche hybride, capable de traiter les relations linéaires et non linéaires qui convient le mieux à la plupart des applications.
La prévision météorologique est devenue un domaine de recherche important au cours des dernières décennies. Il est toujours préférable d'établir une relation linéaire entre les données météorologiques d'entrée et les données cibles correspondantes. Mais avec la découverte de la non-linéarité dans la nature des données météorologiques, l'attention s'est déplacée vers la prédiction non linéaire des données météorologiques.

Ainsi, il comporte deux parties comme mentionné ci-dessous :
1. Prédiction de la pluie(rain), c'est-à-dire s'il pleuvra ou non.
2. Prévision de différents facteurs météorologiques qui nous aideront à prévoir les précipitations. Pour atteindre ce deuxième objectif, un modèle de prédiction de séries temporelles hybrides est construit, qui tentera d'apprendre la relation entre le facteur météo et le temps.

Les données sur les précipitations annuelles moyennes sur la région de *Bhubaneshwar* à *Odisha*, en *Inde*, ont été utilisées pour l'étude. Il se compose des valeurs de données des facteurs tels que la date, la température, le point de rosée(dew point), l'humidité(hum), la vitesse du vent(wind speed), la rafale de vent(wind gust), la direction, la viscosité(viscosity), la pression(), les précipitations, le taux de précipitations, les précipitations totales, l'état, le brouillard, la pluie, la neige, la grêle, tonnerre et tornade. Les données sont enregistrées à un intervalle de temps de 30 minutes et consistent en des données d'un an, de janvier 2017 à janvier 2018.

La série suivante représente , pour voire les composantes de cette data en affiche les premières lignes :

	date	temp	dew_pt	hum	wind_spd	vis	pressure	rain
0	2018-01-01 02 :30 :00	69	64	80	2.3	1	29.87	0
1	2018-01-01 05 :30 :00	68	65	87	4.6	1	29.84	0
2	2018-01-01 08 :30 :00	71	64	71	6.9	1	29.92	0
3	2018-01-01 11 :30 :00	79	69	64	4.6	2	29.89	0
4	2018-01-01 14 :30 :00	79	64	51	6.9	2	29.82	0

```
ARMA Model Results
==============================================================
Dep. Variable:          y   No. Observations:         304
Model:          ARMA(7, 0)   Log Likelihood       -787.148
Method:           css-mle   S.D. of innovations     3.218
Date:                  Sa   30 Apr 2022   AIC    1592.296
Time:            13:49:18   BIC                   1625.749
Sample:                 0   HQIC                  1605.678

==============================================================
coef    std err      z    P>|z|  [0.025    0.975]
--------------------------------------------------------------
const    81.4445  0.745   109.335   0.000  79.984   82.904
ar.L1.y   0.7037  0.057    12.308   0.000   0.592    0.816
ar.L2.y  -0.0666  0.069    -0.967   0.334  -0.202    0.068
ar.L3.y   0.0959  0.068     1.411   0.158  -0.037    0.229
ar.L4.y   0.0501  0.068     0.735   0.462  -0.083    0.184
ar.L5.y  -0.1864  0.068    -2.726   0.006  -0.320   -0.052
ar.L6.y   0.2252  0.069     3.243   0.001   0.089    0.361
ar.L7.y  -0.0665  0.058    -1.155   0.248  -0.179    0.046
```

Les colonnes devant être prédites sont temp,dew_pt,hum,wind_spd,vis,pressure. Mais pour l'instant nous allons essayer de prédire ***la température***. On a regrouper les données de chaque colonne par jour, et on a éliminer le dernière colonne.

	dew_pt	hum	pressure	temp	vis	wind_spd
2018-01-01	65.250	69.25	29.86125	73.375	1.625	5.1750
2018-01-02	59.125	54.00	29.91375	74.375	1.625	5.1875
2018-01-03	69.625	67.00	29.84500	79.500	1.750	3.8875
2018-01-04	75.000	78.75	29.79875	80.625	1.625	8.6250
2018-01-05	76.000	91.50	29.76625	78.250	1.375	4.3125

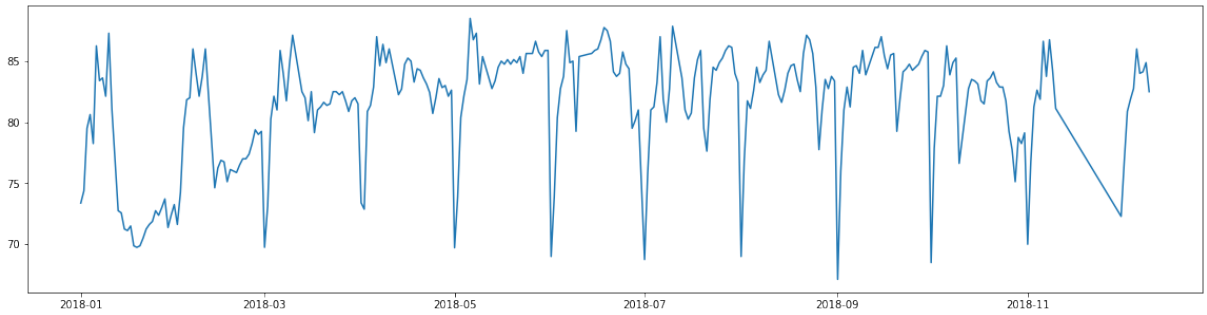


Fig. 4.7 – La température journalière de l'année 2017

Nous avons diviser les données pour l'entraînement et le test, c'est-à-dire 70%, 30%, le meilleur modèle été $ARIMA(7, 0, 0)$:

Tracé des données prédites et des données de test pour savoir comment les données sont couvertes par l'arima.

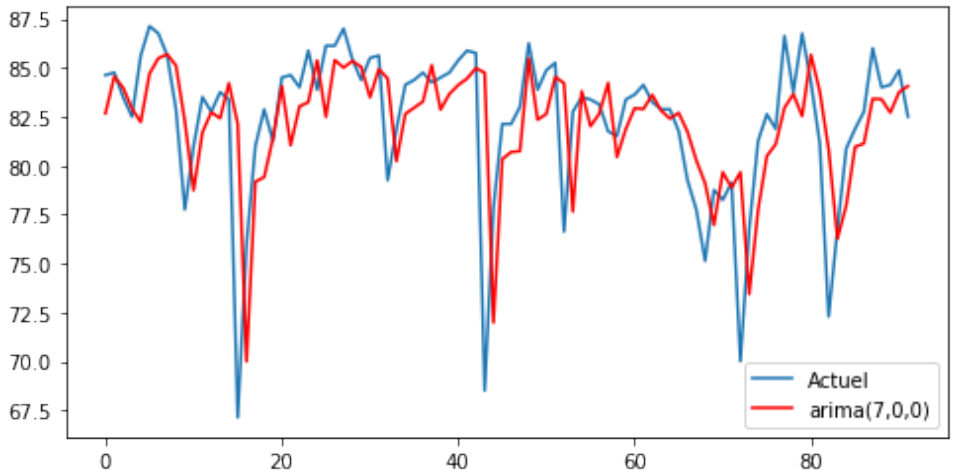


Fig. 4.8 – Le modèle ARIMA(7) vs actuel test

$$TestMSE : 12.361$$

Construisons un modèle ANN pour prédire l'erreur en voyant l'erreur de prédiction des 7 jours précédents :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation
import tensorflow.keras.backend
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error as mse

model = Sequential()
model.add(Dense(100, input_dim=7, bias_initializer="uniform",activation="
  tanh"))
model.add(Dense(50, bias_initializer="uniform", activation="tanh"))
model.add(Dense(25, bias_initializer="uniform", activation="tanh"))
model.add(Dense(1))
model.add(Activation("linear"))
model.compile(loss='mean_squared_error', optimizer='adam')
# train_scaled-----------------------------------------------------
train_X,train_Y = [],[]
for i in range(0 , len(training_error) - 7):
train_X.append(training_error[i:i+7])
train_Y.append(training_error[i+7])
#-----------------------------------------------------------------
new_train_X,new_train_Y = [],[]
for i in train_X:
new_train_X.append(i.reshape(-1))
for i in train_Y:
new_train_Y.append(i.reshape(-1))
new_train_X = np.array(new_train_X)
new_train_Y = np.array(new_train_Y)
#-----------------------------------------------------------------
model.fit(new_train_X,new_train_Y, epochs=400, batch_size=20,verbose=1)
```

On obtient le modèle :

```
Model: "sequential_1"

Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 100)               800

dense_5 (Dense)              (None, 50)                5050

dense_6 (Dense)              (None, 25)                1275

dense_7 (Dense)              (None, 1)                 26

activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 7,151
Trainable params: 7,151
Non-trainable params: 0
```

Une fois le modèle ANN (MLP) créé, l'algorithme de rétro-propagation du gradient de l'erreur permettant de mettre à jour l'ensemble des paramètres du réseau.

Validation du modèle *ANN* :

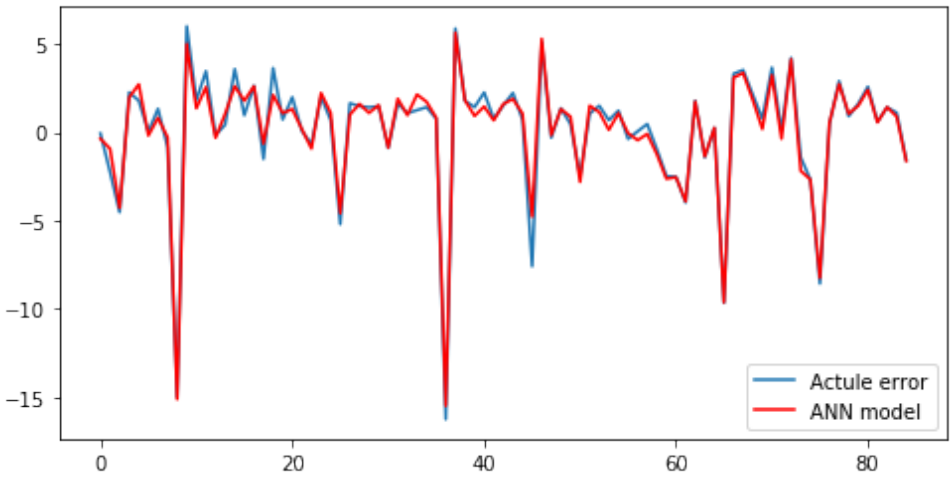


Fig. 4.9 – Le modèle ANN vs actuel error

$TestMSE$: 0.323

Maintenant passons au modèle Hybride. L'archetécture de ce modèle est de la forme :

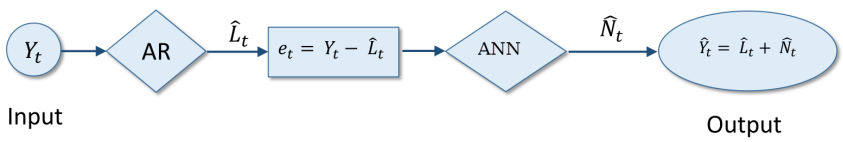


Fig. 4.10 – Hybride modèle : $(ARIMA+ANN)$

```
pred_final = predictions + predicted_list[7:]
```

Traçons le graphique linéaire pour représenter à quel point les données sont couvertes par le modèle Hybride (*ARIMA+ANN*).

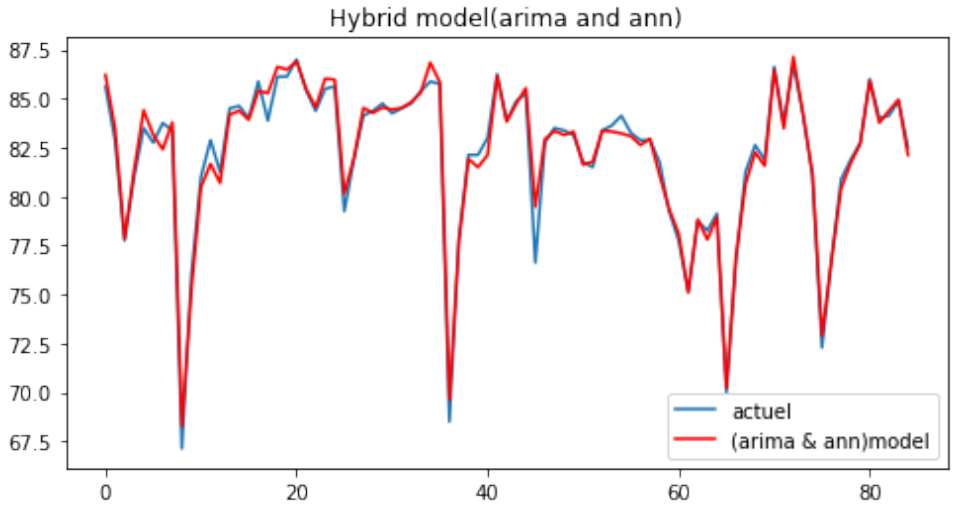


Fig. 4.11 – Hybride modle ARIMA&ANN

Pour la comparaison visuelle de *ARIMA(7)* et Hybride*ARIMA&ANN* modèles, on trace le graphique suivant :

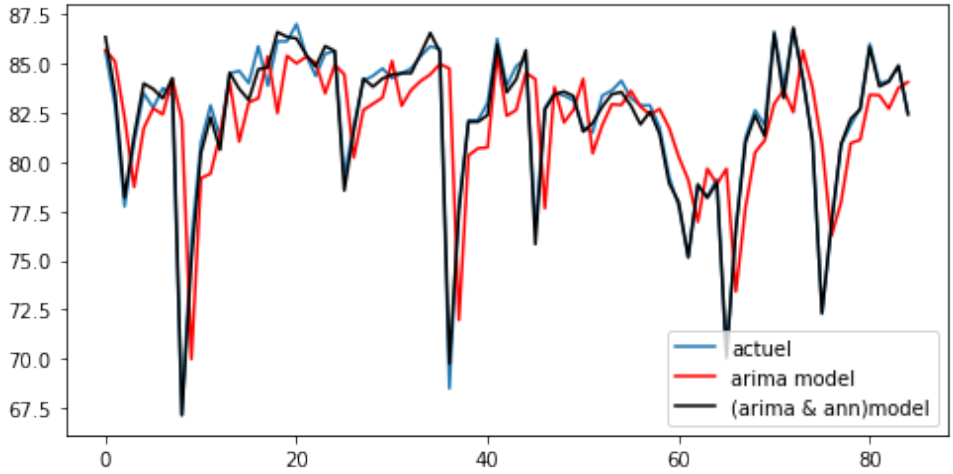


Fig. 4.12 – Comparaison Arima, Hyb_arimaann avec Actuel

Sur la base des valeurs MSE, nous pouvons voir qu'ARIMA-ANN fonctionne mieux avec notre ensemble de tests avec une erreur quadratique moyenne plus petite de 0.172 par rapport à 12.361 fournie par le modèle ARIMA.

Conclusions et Perspectives

Dans cette mémoire nous avons présenté un panorama des méthodes de prévision des séries temporelles. Nous avons présenté l'avantage de l'hbridation ARIMA-LSTM pour améliorer la précision des prévisions des séries chronologiques.
Ce travail propose une technique de prévision en séparant un ensemble de données de séries chronologiques en composants linéaires et non linéaires. Ensuite, les modèles de moyenne mobile intégrée autorégressive (ARIMA) et de réseau neuronal artificiel (ANN) sont utilisés pour reconnaître et prédire séparément les composants détaillés et approximatifs reconstruits, respectivement.

De cette manière, l'approche proposée utilise les forces uniques de ARIMA et LSTM pour améliorer la précision des prévisions. Les résultats montrent clairement que la méthode proposée atteint les meilleures précisions de prévision pour la série étudier.

Il est noté que le modèle *ARIMA* a produit de meilleurs résultats avec une plus petite quantité de données dans les études universitaires précédentes. Cependant, la grande quantité de données dans les modèles générés dans le cadre de cette étude montre que les algorithmes basés sur l'apprentissage en profondeur, tels que *LSTM* (Long-Short-Term-Memory), surpassent les algorithmes traditionnels, tels que le modèle *ARIMA*. Il est fortement recommandé de refaire cette étude avec des données plus réelles, et de comparer nos résultats avec les résultats d'autres études afin de confirmer cette conclusion.

Bibliographie

[1] G. E. Box, "Jenkins. gm time series analysis : Forecasting and control. revised edition," 1976.

[2] A. Charpentier, "Cours de séries temporelles : théorie et applications," *Université Paris Dauphine*, 2006.

[3] J. Jacques, "Introduction aux séries temporelles," *Polytech Lille*, 2013.

[4] R. Bourbonnais, *Économétrie-10e éd. : Cours et exercices corrigés*. Dunod, 2018.

[5] A. Yves, "Séries temporelles avec r : Méthodes et cas," *Springer&Verlang, France*, 2011.

[6] A. Lagnoux, "Renforcement statistique séries chronologiques," *Université de Tou% louse Le Mirail*, 2015.

[7] J. G. MacKinnon, "Approximate asymptotic distribution functions for unit-root and cointegration tests," *Journal of Business & Economic Statistics*, vol. 12, no. 2, pp. 167–176, 1994.

[8] R. von Sachs and S. Van Bellegem, "Stat 2414 séries chronologiques," *STAT*, vol. 2414, no. 1, 2005.

[9] W. N. Venables and D. M. Smith, "The r development core team," *An Introduction to R, Version*, vol. 1, no. 0, 2003.

[10] P. Besse, "Apprentissage statistique & data mining," *Toulouse : INSA*, 2009.

[11] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[12] T. Zia and U. Zahid, "Long short-term memory recurrent neural network architectures for urdu acoustic modeling," *International Journal of Speech Technology*, vol. 22, no. 1, pp. 21–30, 2019.

[13] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[14] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks*, vol. 2, no. 3, pp. 183–192, 1989.

[15] L. Gasmi, Z. C. Elmezouar, and M. K. Attouch, "Wind power forecasting using neural network and arima models (field of" kabertene", in southern algeria)," *INTERNATIONAL JOURNAL OF ECOLOGICAL ECONOMICS & STATISTICS*, vol. 39, no. 2, pp. 71–79, 2018.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

الملخص

يعد التنبؤ بالسلاسل الزمنية موضوعًا مهمًا في المالية ، الاقتصاد والأعمال. يدرس هذا العمل إمكانية تطبيق وإمكانات تقنيات التنبؤ مثل ، $ARIMA$ و $SARIMA$ وكذلك الشبكات العصبية التراجعية (RNN) و على الخصوص النموذج $(LSTM)$ على سلاسل حقيقية. النموذج المقترح في التطبيق الثاني$(ARIMA - LSTM)$، حقق أفضل درجات دقة في التنبؤ للسلسلة المدروسة.

الكلمات الرئيسية: السلاسل الزمنية ، التنبؤ، $LSTM, RNN, ARIMA$.

Abstract

Time series forecasting is an important topic in finance, economics, business. This work studies the applicability and potential of forecasting techniques such as ARIMA, SARIMA, as well as recurrent neural networks (RNN), in particular the (LSTM) model, on real data. The proposed approach uses the unique strengths of ARIMA and LSTM to improve forecast accuracy. The results clearly show that the proposed method (ARIMA-LSTM); in the second series (the second application), achieves the best prediction accuracies for the series studied.

***Keywords**: Time series, Forecasting, ARIMA, RNN, LSTM.*

Résumé

La prévision des séries temporelles est un sujet important en finance, en économie, en affaires. Ce travail étudie l'applicabilité et le potentiel des techniques de prévision telles que ARIMA, SARIMA, ainsi que des réseaux de neurones récurrents (RNN), en particulier le modèle (LSTM), sur des données réelles. L'approche proposée utilise les forces uniques de ARIMA et LSTM pour améliorer la précision des prévisions. Les résultats montrent clairement que la méthode proposée (ARIMA-LSTM) ; dans la deuxième application, atteint les meilleures précisions de prévision pour la série étudier.

***Mots clés** : Séries temporelles, Prévision, ARIMA, RNN, LSTM.*