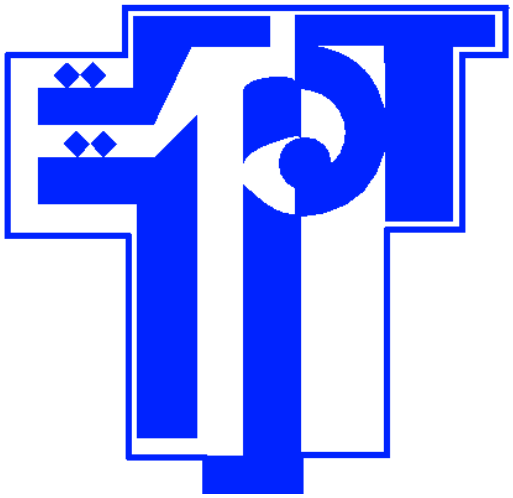
Université de Carthage



Ecole Polytechnique de Tunisie

Rapport de Projet

Analyse et prédictions des prix des actions de CRUDE OIL

Travail réalisé par : Rahma KHARRAT
Myriam EL AMRI
Supervisé par : Mme. Ines BOUSNINA

AU : 2023-2024

Introduction

Dans le cadre du module 'Théorie des signaux et systèmes', nous avons eu l'occasion de concevoir un projet exploitant divers concepts enseignés au cours de la formation, permettant ainsi d'approfondir nos connaissances. Notre équipe, composée de Myriam EL AMRI et Rahma KHARRAT , a choisi de se concentrer sur le cours "Processus aléatoires", avec pour objectif la prédiction de séries temporelles représentant les cours boursiers du pétrole brut (CRUDE OIL). Ce domaine d'application est notoirement réputé et revêt une grande importance pour les investisseurs en bourse. Ainsi, nous avons consacré nos efforts à relever ce défi majeur. Notre projet comprend des codes écrits en Python, utilisés tant pour l'analyse du signal que pour la prédiction des prix. Il se divise en deux cas d'études distincts, mettant en œuvre diverses méthodes telles que les modèles AR, MA, ARMA, ARIMA, ainsi que les réseaux de neurones LSTM.

Remarque : Ce rapport est accompagné de 3 notebooks qui donnent les codes détaillés de toutes les parties.

Table des matières

Table des figures

A Introduction sur les notions des séries temporelles

Les séries temporelles constituent un domaine singulier, une discipline à part entière. Les experts disent :"une bonne prévision est une bénédiction, tandis qu'une mauvaise prévision peut s'avérer dangereuse"

A.1 Définitions générales

Les données en série temporelle représentent les observations d'une variable de réponse Y(t) à différents moments dans le temps t. Ces données sur la variable sont recueillies à des intervalles réguliers et dans un ordre chronologique. Tout ce qui est observé de manière séquentielle au fil du temps constitue une série temporelle. Par exemple, les données collectées sur les ventes de smartphones à plusieurs intervalles de temps, le produit intérieur brut (PIB) d'une nation chaque année, la production d'électricité chaque année/mois, etc., sont tous des exemples de données en série temporelle. L'objectif de la prévision des données en série temporelle est de comprendre comment la séquence d'observations va se poursuivre dans le futur.
La bibliothèque 'yfinance' est un module Python permettant aux utilisateurs de télécharger des données financières depuis Yahoo Finance. Elle est largement utilisée pour obtenir des informations sur les cours des actions, les indices, les devises, les matières premières, ainsi que d'autres instruments financiers.
Yahoo Finance est un service en ligne fournissant des informations financières, des données sur les marchés boursiers, des actualités et d'autres ressources liées à la finance.

A.2 Exemple de données

On utilise la bibliothèque yfinance pour télécharger les données boursières de l'entreprise APPLE pour la période allant du 1er janvier 2022 au 1er janvier 2023 depuis Yahoo Finance. Ensuite, on affiche les cinq premières lignes de données :

Date	Open	High	Low	Close	Adj Close	Volume
2022-01-03	177.830002	182.880005	177.710007	182.009995	179.953888	104487900
2022-01-04	182.630005	182.940002	179.119995	179.699997	177.669998	99310400
2022-01-05	179.610001	180.169998	174.639999	174.919998	172.944000	94537600
2022-01-06	172.699997	175.300003	171.639999	172.000000	170.056976	96904000
2022-01-07	172.889999	174.139999	171.029999	172.169998	170.225052	86709100

TABLE 1 – Données boursières pour le symbole AAPL.

.
Les informations enregistrées le 3 janvier 2022 fournissent un aperçu détaillé des performances de l'action.Par exemple, le cours d'ouverture (Open) s'est établi à 177.830002, tandis que le cours le plus élevé

(High) atteint au cours de la journée a été de 182.880005. Le cours le plus bas (Low) enregistré pendant la journée a été de 177.710007. À la clôture (Close) de la journée, le cours de l'action s'est fixé à 182.009995. Le cours de clôture ajusté (Adj Close) était de 179.953873.

Visualisation des données :

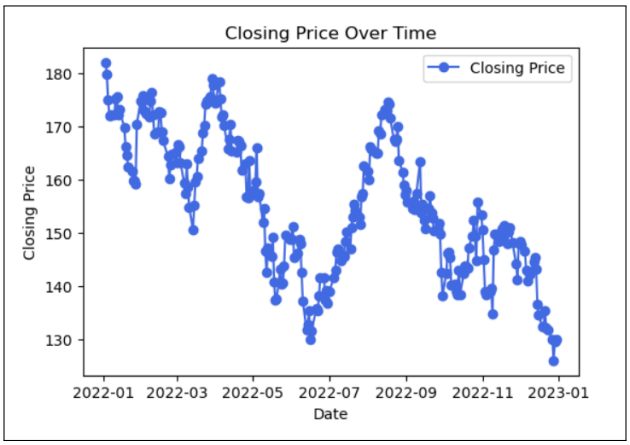


FIGURE 1 – Visualisation des données de APPLE

A.3 Composantes d'une série temporelle

Une série temporelle peut se composer de plusieurs éléments, parmi lesquels figurent les principaux :

Composante de Tendance (Tt) : Elle représente la direction générale ou la trajectoire à long terme des données, pouvant adopter une tendance linéaire ou non linéaire. Diverses méthodes de lissage, telles que la moyenne mobile, peuvent être employées pour présenter cette composante. La moyenne mobile implique le calcul de la moyenne d'un certain nombre de points de données consécutifs, atténuant ainsi les variations à court terme afin de mettre en relief les tendances plus générales.

Composante Saisonnière (St) : Elle capture les variations régulières ou périodiques qui surviennent à des intervalles fixes, souvent associées aux saisons de l'année.

Composante Cyclique (Ct) : Elle représente des variations qui se manifestent à des intervalles irréguliers, généralement liées à des cycles économiques ou d'autres cycles de long terme.

Terme d'Erreur (ϵ_t) (également appelé bruit blanc) : Il symbolise la variabilité aléatoire ou le bruit qui ne peut pas être expliqué par les composantes précédentes. Si le bruit blanc constitue la seule composante des données de la série temporelle, celles-ci ne peuvent pas être utilisées pour la prédiction.

Types de décompositions :

Décomposition additive de la série temporelle : $Y_t = T_t + S_t + C_t + \varepsilon_t$. La décomposition additive est la plus appropriée lorsque l'amplitude des fluctuations saisonnières, ou la variation autour de la tendance-cycle, ne varie pas en fonction du niveau de la série temporelle.

Décomposition multiplicative : $Y_t = T_t \times S_t \times C_t \times \varepsilon_t$. Lorsque la variation dans le schéma saisonnier, ou la variation autour de la tendance-cycle, semble être proportionnelle au niveau de la série temporelle, une décomposition multiplicative est plus appropriée. Les décompositions multiplicatives sont courantes avec les séries temporelles économiques.

Une alternative à l'utilisation d'une décomposition multiplicative consiste d'abord à transformer les données jusqu'à ce que la variation dans la série semble stable au fil du temps, puis à utiliser une décomposition additive. Lorsqu'une transformation logarithmique est appliquée, cela équivaut à utiliser une décomposition multiplicative, car $Y_t = T_t \times S_t \times C_t \times \varepsilon_t$ est équivalent à $\log(Y_t) = \log(S_t) + \log(T_t) + \log(C_t)$.

visualisation d'un exemple de décomposition :

La décomposition saisonnière repose sur l'idée que les séries temporelles peuvent être décomposées en diverses composantes afin de mieux appréhender les tendances, les variations saisonnières et les résidus. Cette opération peut être réalisée au moyen de différentes méthodes, telles que la méthode de décomposition X-12-ARIMA, la méthode STL (Seasonal and Trend decomposition using LOESS), ou tout simplement en utilisant la décomposition additive de la bibliothèque statsmodels en Python.

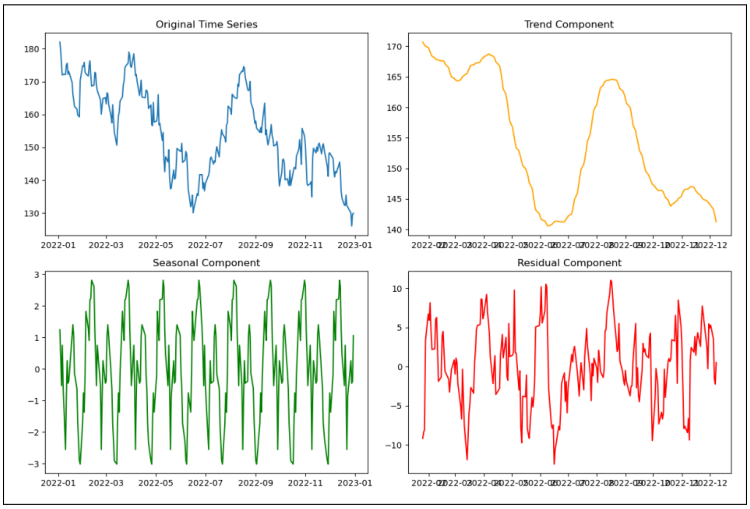


FIGURE 2 – Décomposition d'une série temporelle

A.4 Stationnarité

Définition

Une série temporelle est considérée comme stationnaire si elle présente les trois propriétés suivantes :

- **Moyenne constante** : La moyenne reste constante au fil du temps.
- **Variance constante** : La variance demeure constante au fil du temps.
- **Covariance constante** : La covariance entre deux périodes dépend du décalage entre ces périodes et non du moment où la covariance est calculée. Ainsi, la covariance entre les séries au moment $t = 2$ et $t = 4$ devrait être approximativement la même que celle entre les séries au moment $t = 7$ et $t = 9$.

Importance de la stationnarité

Travailler avec des séries stationnaires est essentiel pour plusieurs raisons lors de l'analyse de données de séries temporelles :

— **Modélisation plus facile** : Les séries stationnaires sont plus simples à modéliser et à prévoir, de nombreux modèles de séries temporelles supposent la stationnarité.
— **Interprétation significative** : Les données stationnaires permettent une interprétation plus claire des tendances, motifs, et relations au sein des données.
— **Tests statistiques** : De nombreux tests et techniques statistiques reposent sur l'hypothèse de la stationnarité, tels que les tests d'autocorrélation et de cointégration.

Les défis des séries temporelles non stationnaires

Trend - Saisonalité - Hétéroscédasticité

Comment rendre une série stationnaire ?

La plupart des séries temporelles économiques présentent des tendances influencées par des facteurs tels que la croissance économique, l'inflation ou les changements démographiques. Pour appliquer efficacement des techniques de séries temporelles, il est crucial d'éliminer la tendance. Le détrending consiste à éliminer la composante de tendance pour se concentrer sur les variations sous-jacentes.
Le détrending peut être réalisé par des techniques telles que :

— **Différenciation** : Calculer les différences entre les valeurs successives de la série temporelle.
— **Décomposition** : Décomposer la série temporelle en composantes telles que la tendance, la saisonnalité, et le résidu pour les analyser séparément.
— **Régression** : Modéliser la tendance par régression puis examiner les résidus.

Vérification de la Stationnarité

L'utilisation de plusieurs tests de stationnarité offre une meilleure compréhension des propriétés de la série.

Outil Graphique : la fonction d'autocorrélation (ACF)

Une méthode pour vérifier la stationnarité consiste à analyser l'ACF des données de la série temporelle. L'ACF mesure la corrélation entre un point de données et ses valeurs décalées dans le temps. Dans une série stationnaire, les valeurs de l'ACF devraient décroître rapidement, indiquant une faible corrélation entre les observations à différents décalages temporels.

Outils statistiques : les tests de stationnarité

Deux tests couramment utilisés sont le Test d'Augmented Dickey Fuller (ADF) et le test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS). Ces deux tests sont souvent employés ensemble pour obtenir une évaluation plus robuste de la nature de la série temporelle.

Test d'Augmented Dickey Fuller (ADF)

Ce test vérifie si la série temporelle a une racine unitaire, c'est-à-dire si elle est non stationnaire.

H_0 : Il y a une racine unitaire, la série temporelle est non stationnaire.

H_a : Il n'y a pas de racine unitaire, la série temporelle est stationnaire.

Deux types de séries temporelles stochastiques non stationnaires peuvent être observés : le modèle de Random Walk sans dérive et le modèle de Random Walk avec dérive.

Modèle de Random Walk sans dérive

Dans ce modèle, la moyenne de la série est constante, mais la variance augmente avec le temps. La première différence du Random Walk sans dérive est une série temporelle stationnaire.

Modèle de Random Walk avec dérive

Dans ce modèle, à la fois la moyenne de la série et la variance augmentent avec le temps. La série n'est pas stationnaire.

Test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS)

Les hypothèses nulle et alternative sont opposées au test ADF
H_0 : Le processus est stationnaire par tendance.
H_a : La série a une racine unitaire (la série n'est pas stationnaire autour d'une tendance déterministe).
L'utilisation de ces deux tests ensemble permet de conclure l'une des quatre situations suivantes :

1. Les deux tests concluent à la stationnarité. La série est stationnaire.
2. Les deux tests concluent à la non-stationnarité. La série n'est pas stationnaire.
3. Le test ADF conclut à la stationnarité mais le test KPSS conclut à la non-stationnarité. Cela indique qu'il est nécessaire d'utiliser la différenciation pour rendre la série stationnaire.
4. Le test ADF conclut à la non-stationnarité mais le test KPSS confirme la stationnarité. Cela indique qu'il est nécessaire d'utiliser la détendance pour rendre la série stationnaire.

Remarque : L'application des logarithmes avant la différenciation est souvent utilisée lorsque les données présentent une croissance exponentielle ou une variabilité non constante au fil du temps. Après avoir calculé les premières différences, on vérifie si la nouvelle série ainsi obtenue est stationnaire. Sinon, on procède à la différenciation jusqu'à obtenir la série stationnaire désirée.

A.5 Exemple d'un série stationnaire : bruit blanc

Distribution :
En raison de sa nature stationnaire intrinsèque, il ne présente pas de tendance prévisible à long terme, ce qui le rend dépourvu de mémoire. Sa distributuon est comme suit :

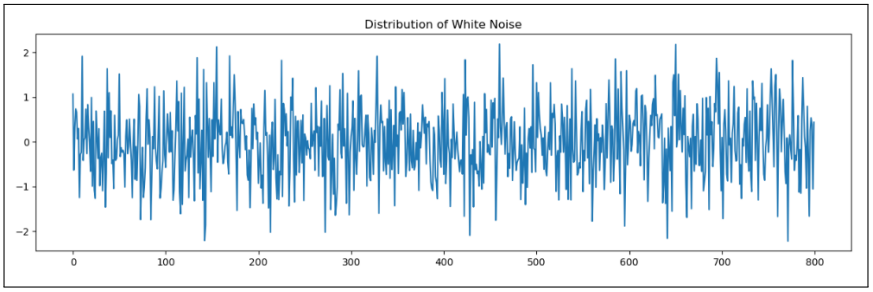


FIGURE 3 – Distribution de bruit blanc

On observe que la distribution reste constante autour de la moyenne et est totalement aléatoire. Prédire

le prochain mouvement de la série temporelle devient difficile. En examinant l'autocorrélation de cette série, on constatera une autocorrélation totalement nulle, indiquant que la corrélation entre la série à n'importe quel moment t et ses valeurs retardées est nulle.

autocorrélation :

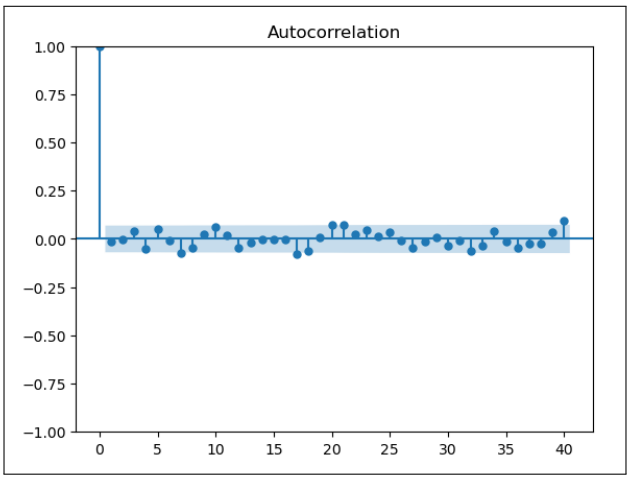


FIGURE 4 – autocorrélation

On observe que tous les décalages se situent à l'intérieur de la zone surlignée en bleu. Cela suggère qu'il existe peu, voire aucune corrélation entre les observations à différents décalages. Si un ou plusieurs pics se trouvent à l'extérieur de cette plage, ou si plus de 5 % des pics se situent en dehors de ces limites, alors nous pouvons conclure que la série n'est pas un "Bruit Blanc".

B Analyse et prédictions avec AR, MA, ARMA, ARIMA

"Comprendre la théorie derrière un modèle est la moitié de la tâche à accomplir"

B.1 ACF & PACF

B.1.1 Fonction d'autocorrélation (ACF)

La corrélation pour une série temporelle et une version retardée d'elle-même. Il s'agit de la corrélation entre l'observation au moment actuel et les observations aux moments précédents. La fonction d'autocorrélation commence avec un décalage de 0, qui est la corrélation de la série temporelle avec elle-même et donne donc une corrélation de 1.
Le graphique de la fonction d'autocorrélation (ACF) indique une tendance lorsque les autocorrélations pour de petits retards sont positives et importantes, puis diminuent lentement. En revanche, une saisonnalité est indiquée par des autocorrélations plus élevées pour les retards saisonniers.
Nous utiliserons la fonction `plot_acf` de la bibliothèque `statsmodels.graphics.tsaplots`.
Le graphique d'autocorrélation peut répondre aux questions suivantes :

- La série temporelle observée est-elle un bruit blanc/aléatoire ?
- Une observation est-elle liée à une observation adjacente, une observation décalée de deux, et ainsi de suite ?
- La série temporelle observée peut-elle être modélisée avec un modèle MA (moyenne mobile) ? Si oui, quel est l'ordre ?

B.1.2 Fonction d'autocorrélation partielle (PACF)

La corrélation partielle au décalage k est la corrélation entre X_t et X_{t-k} qui n'est pas expliquée par les décalages de 1 à $k-1$.
Nous utiliserons la fonction `plot_pacf` de la bibliothèque `statsmodels.graphics.tsaplots` avec le paramètre `method="ols"` (régression de la série temporelle sur ses décalages et sur la constante).
(Voir `statsmodels.tsa.stattools.pacf`)
Note : Le paramètre par défaut pour la méthode est `yw` (Yule-Walker avec ajustement de la taille d'échantillon dans le dénominateur pour `acovf`). Cependant, cette valeur par défaut provoque des autocorrélations implausibles supérieures à 1 sur les données d'échantillon. Par conséquent, nous modifions le paramètre de méthode pour éviter ce problème. `ywmle` fonctionnerait également bien, comme suggéré dans cette publication sur StackExchange.
Le graphique de corrélation partielle peut répondre à la question suivante :

- La série temporelle observée peut-elle être modélisée avec un modèle AR (autorégressif) ? Si oui, quel est l'ordre ?

B.2 Modèles autorégressifs (AR) et modèles de moyenne mobile (MA)

B.2.1 Modèles autorégressifs(AR)

Présentation

La représentation mathématique du modèle AR d'ordre p, noté $AR(p)$, est donnée par :

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \varepsilon_t$$

où :

- X_t est la valeur de la série temporelle à l'instant t,
- c est une constante,
- $\phi_1, \phi_2, \ldots, \phi_p$ sont les coefficients du modèle,
- $X_{t-1}, X_{t-2}, \ldots, X_{t-p}$ sont les valeurs passées de la série temporelle,
- ε_t est une composante d'erreur qui capture le bruit ou l'erreur aléatoire.

Le modèle AR suppose que la valeur actuelle d'une série temporelle est une combinaison linéaire de ses valeurs passées, augmentée d'une composante d'erreur. La performance du modèle dépend de la capacité de ces coefficients à capturer les tendances et les motifs de la série temporelle.

Comment déterminer l'ordre du modèle AR ?

Pour déterminer l'ordre p du modèle AR, nous pouvons utiliser la Fonction d'Autocorrélation (ACF) et la Fonction d'Autocorrélation Partielle (PACF). L'ordre p peut être défini si :

- La fonction d'autocorrélation (ACF) décroît de manière exponentielle ou a une forme sinusoïdale,
- La fonction d'autocorrélation partielle (PACF) présente un pic significatif au retard p, mais aucun pic après.

Si les graphiques ACF et PACF ne sont pas concluants, nous pouvons tester différents ordres p et choisir le modèle en fonction du Critère d'Information d'Akaike (AIC).

Limites du modèle

L'autorégression devrait être utilisée pour les séries temporelles sans tendance et/ou saisonnalité, en contrôlant l'ordre de l'autorégression. Un ordre élevé indique que nous devrions utiliser des paramètres supplémentaires, par exemple, en ajoutant une moyenne mobile (MA). Il est également important de noter que nous ne pouvons prédire une cible que jusqu'à l'horizon que nous choisissons.

Implémentation en Python

Nous pouvons mettre en œuvre une autorégression en Python en utilisant la classe `AutoReg` du package `statsmodels`. Pour notre prévision effective, nous appelons la méthode `predict()`. Pour choisir le bon ordre, nous pouvons également utiliser la classe `ar_select_order`.

B.2.2 Modèle Moyenne mobile (MA)

Présentation :

La prévision de la série temporelle peut être réalisée à l'aide d'un modèle de moyenne mobile (MA) représenté par l'équation :

$$\hat{y}_t = \varepsilon_t + \beta_1 \varepsilon_{t-1} + \ldots + \beta_q \varepsilon_{t-q}$$

où

— y_t est la valeur de la série temporelle au temps t,
— c est le terme d'interception,
— $\theta_1, \theta_2, \ldots, \theta_q$ sont les coefficients du modèle,
— ε_t est le terme d'erreur ou bruit blanc (différence entre la valeur originale et la valeur prédite).

Les coefficients du modèle, $\beta_1, \beta_2, \ldots, \beta_q$, sont estimés à l'aide de diverses méthodes, telles que l'estimation des moindres carrés. Le modèle de moyenne mobile suppose que la valeur actuelle y_t dépend des termes d'erreur, y compris l'erreur actuelle ε_t, ainsi que les erreurs passées ε_{t-1}, ε_{t-2}, ε_{t-3}, et ainsi de suite.

Comme les termes d'erreur sont aléatoires, il n'y a pas de relation linéaire entre la valeur actuelle et les termes d'erreur.

Comment déterminer l'ordre q du modèle MA ?

L'ordre q de la moyenne mobile représente la largeur de la fenêtre de la moyenne mobile et peut être déterminé en utilisant le graphique de la fonction d'autocorrélation (ACF). Nous pouvons déterminer l'ordre si :

— La fonction d'autocorrélation (ACF) présente un pic significatif au retard q, mais aucun après, et
— La fonction d'autocorrélation partielle (PACF) décroît de manière exponentielle ou a une forme sinusoïdale.

Limites du modèle

En général, la moyenne mobile peut être utilisée pour des séries temporelles stationnaires.

Implémentation en Python

Nous pouvons mettre en œuvre une moyenne mobile (MA) en Python en utilisant la classe `ARIMA` du module `statsmodels` de Python. Comme l'objectif est de modéliser uniquement une moyenne mobile, il est nécessaire de fixer l'ordre des termes autorégressifs et d'intégration à zéro. Le modèle `ARIMA` utilise les paramètres d'ordre sous la forme d'un tuple (p, d, q), et dans ce cas, les deux premiers ordres sont définis à zéro.

L'utilisation de la classe `ARIMA` est similaire à la classe `AutoReg`. Les méthodes `fit()` et `predict()` sont utilisées pour entraîner le modèle et effectuer des prédictions.

B.3 Cas d'étude 1

Nous allons analyser une série temporelle représentant les prix des actions du pétrole brut. Nous choisissons un intervalle de temps spécifique, du 13 décembre 2022 au 13 mars 2023. Notre objectif est de modéliser ce signal à l'aide du modèle AR ou MA approprié. On obtient les données suivantes :

Date	Open	High	Low	Close	Volume
2022-12-13	73.290001	76.370003	73.209999	75.389999	352858
2022-12-14	75.269997	77.750000	74.900002	77.279999	292488
2022-12-15	77.370003	77.769997	75.330002	76.110001	224663
2022-12-16	76.370003	76.570000	73.330002	74.290001	99205
2022-12-19	74.500000	76.410004	73.809998	75.190002	69348

TABLE 2 – Prix d'actions de Crude Oil

Visualisation des données :

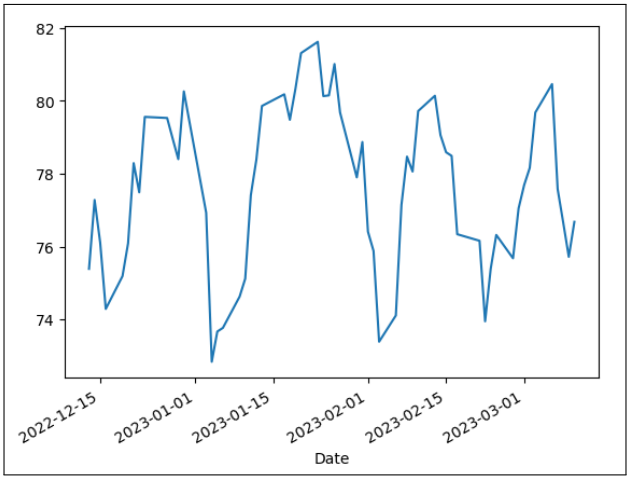


FIGURE 5 – Visualition des données cas d'étude 1

On observe que notre signal présente une tendance non linéaire. De plus, il exhibe des pics et des creux, indiquant ainsi des variations rapides des prix au fil du temps, ce qui correspond à une dérivée importante. Nous pouvons donc conclure que les composantes de hautes fréquences de ce signal ne sont pas nulles.

B.3.1 Analyse de fourier

Le spectre de fourier donne :

En analysant le spectre de fréquence, nous pouvons identifier les tendances à différentes échelles de

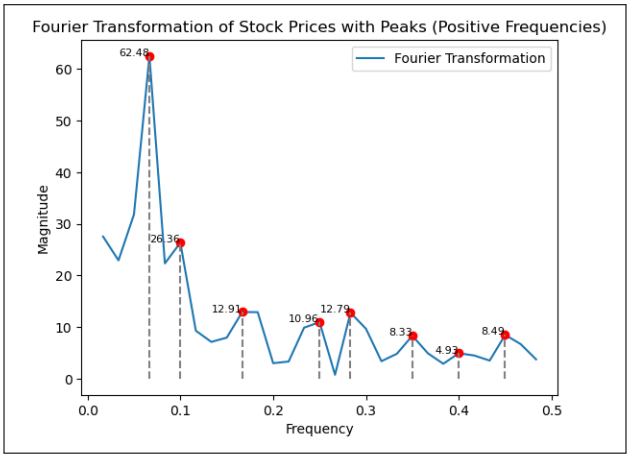


FIGURE 6 – Spectre de fourier cas d'étude 1

temps. On remarque qu'il y a une seule composante dominante (correspondant à une amplitude de 60.48) à basse fréquence, suggérant ainsi la présence d'une tendance à long terme. Les autres composantes représentent les fluctuations à court terme.

Pour déterminer s'il existe une périodicité dans le signal, nous allons éliminer les fluctuations correspondant aux petits pics dans le spectre de fréquence.

On visualise le signal reconstitué on obtient :

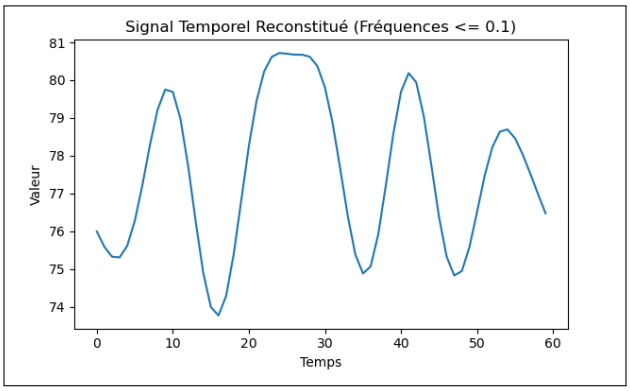


FIGURE 7 – Sinal reconstitué cas d'étude 1

Le signal temporel reconstitué révèle la présence d'une certaine périodicité, avec une période d'environ dix jours.

B.3.2 ACF & PACF cas d'étude 1

La fonction d'autocorrélation donne :

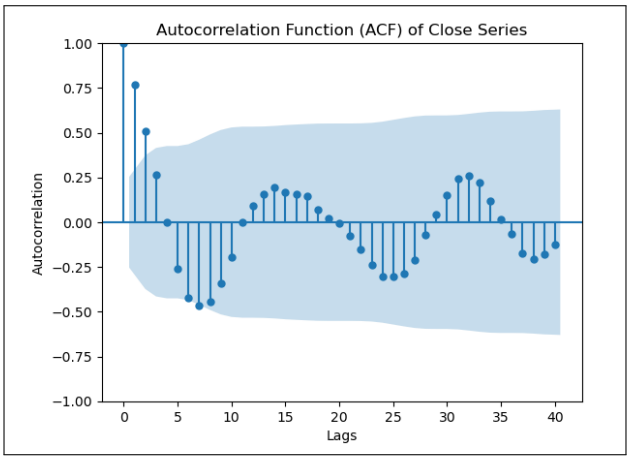


FIGURE 8 – Autocorrélation cas d'étude 1

On observe que le diagramme ACF présente une forme sinusoïdale, attribuée à la présence d'une composante saisonnière. De plus, aucun pic significatif n'est observé après le 7 eme décalage, suggérant que la série étudiée est stationnaire. Pour confirmer cette hypothèse de stationnarité, nous allons utiliser les tests ADF et KPSS.

Pour la PACF :

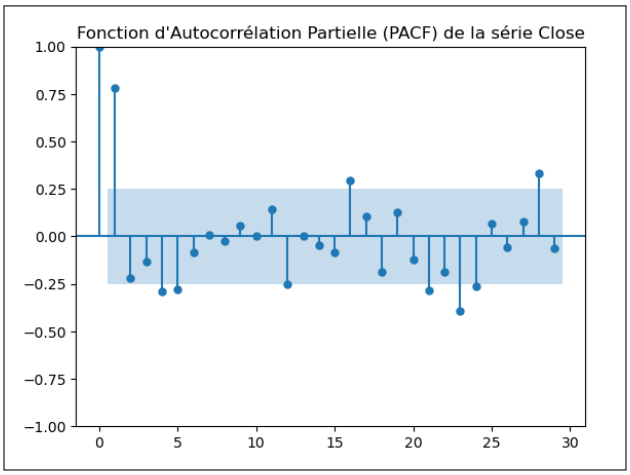


FIGURE 9 – Autocorrélation partielle cas d'étude 1

On observe que le diagramme PACF présente des composantes significatives de manière plus périodique, cohérent avec l'analyse de Fourier précédente, confirmant ainsi la présence de la composante saisonnière.

B.3.3 Test de stationnarité

Résultats du test ADF (Augmented Dickey-Fuller) :
Hypothèse nulle : La série possède une racine unitaire (non stationnaire)
Statistique ADF : -4.507173188554421
Valeur p : 0.0001908538510862125
Nombre de retards : 4
Nombre d'observations : 55
Valeurs critiques : '1%' : -3.5552728880540942, '5%' : -2.9157312396694217, '10%' : -2.5956695041322315
La statistique KPSS est de 0.07901674006420926 avec une valeur $p = 0.1$. On observe que $p > 0.05$, donc on accepte H_0. La série est stationnaire autour d'une tendance déterministe.

En conclusion, d'après les résultats des deux tests, nous nous trouvons dans le cas du Scénario I : la série est stationnaire.

B.3.4 Choix du modèle et de l'ordre

D'après les diagrammes ACF et PACF, on conclut qu'un modèle MA serait plus approprié qu'un modèle AR, vu la présence de la composante saisonnière. Puisque nous n'avons aucun pic significatif après le lag 7, le modèle MA serait d'ordre 7.

Prédictions MA :

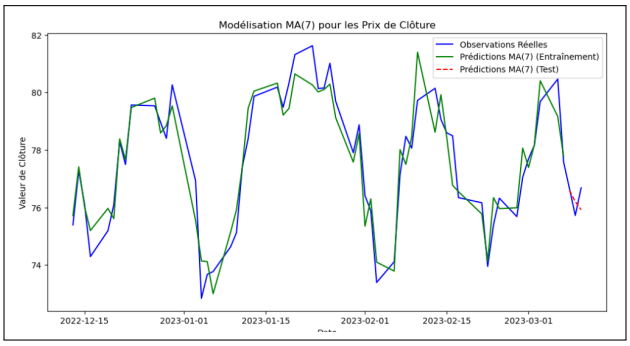


FIGURE 10 – Prédictions avec MA cas d'étude 1

Bien que les modèles MA soient généralement utilisés pour modéliser la composante irrégulière ou le bruit dans une série temporelle, et qu'ils ne capturent pas explicitement la composante saisonnière, on observe une certaine cohérence entre les valeurs réelles et les valeurs prédites sur une période de 5 jours.

Si les diagrammes ACF et PACF suggèrent que le modèle autorégressif (AR) n'est pas optimal pour la prédiction du signal financier, nous explorons une approche alternative. Nous chercherons à modéliser les données en utilisant un nombre restreint de retards, en particulier ceux qui présentent des pics significatifs en dehors de la zone bleue dans le diagramme PACF.

Prédictions MA :

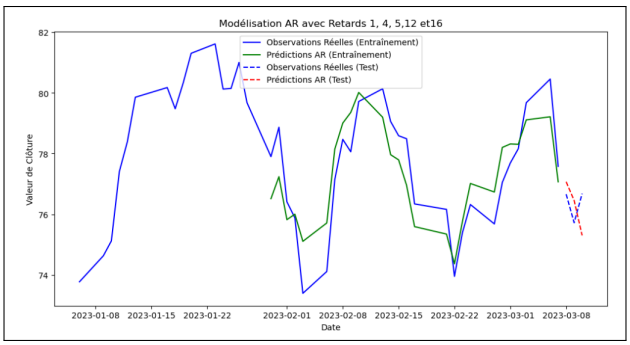


FIGURE 11 – Prédictions avec AR cas d'étude 1

On observe que les prédictions correspondent bien aux valeurs réelles au cours des quatre premiers jours, mais à partir du cinquième jour, on constate un écart croissant entre les deux, atteignant 0.9 $ à la fin de la période.

B.4 Modèle ARIMA

Modèle ARMA

Le modèle Autorégressif à Moyenne Mobile (ARMA) combine l'autorégression d'ordre p avec la moyenne mobile d'ordre q. Ainsi, l'approche décrit la relation d'une série temporelle avec elle-même et le bruit

aléatoire des pas de temps précédents :

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

Ici, le premier terme de la sommation représente la partie autorégressive et le deuxième terme de la sommation représente la moyenne mobile.

Choisir les bons ordres pour les composantes p et q de notre modèle ARMA peut être assez difficile, car les graphiques de la fonction d'autocorrélation (ACF) et de la fonction d'autocorrélation partielle (PACF) pourraient ne pas nous aider beaucoup si les deux composantes sont présentes. Cependant, ils pourraient indiquer l'ordre et nous aider à trouver un bon point de départ pour l'ajustement de nos hyperparamètres. Nous pouvons également utiliser une recherche sur grille dans laquelle nous testons différentes combinaisons de p et q, puis choisissons les ordres en fonction d'un critère choisi, tel que l'AIC (criterium d'information d'Akaike).
Tout comme l'autorégression et la moyenne mobile, la méthode ARMA fonctionne bien uniquement pour les séries temporelles stationnaires.

Modèle ARIMA

Les modèles AutoRegressive Integrated Moving Average (ARIMA) sont largement utilisés pour les séries temporelles non stationnaires. Au lieu de réaliser la différenciation dans une étape distincte, nous pouvons l'inclure directement dans le modèle ARMA en ajoutant un ordre de différenciation d, ce qui nous conduit à un modèle AutoRegressive Integrated Moving Average (ARIMA). Un ordre d de 1 signifie que la série temporelle est différenciée une fois, tandis qu'un ordre d de 2 signifie que nous différencions la série temporelle deux fois.

B.5 Cas d'étude 2

Nous allons analyser une série temporelle représentant les prix des actions de Netflix. Notre objectif est de modéliser ce signal à l'aide du modèle ARIMA approprié.

Visualisation des données :

Date	Price
2018-04-02	280.290009
2018-04-03	283.670013
2018-04-04	288.940002
2018-04-05	293.970001
2018-04-06	288.850006

Il est clairement évident que la série n'est pas stationnaire, exhibant une tendance non linéaire. Par conséquent, on en déduit qu'un modèle ARMA ne sera efficace dans notre cas qu'avec une étape de différenciation pour rendre la série stationnaire. En alternative, pour simplifier le processus, nous pouvons

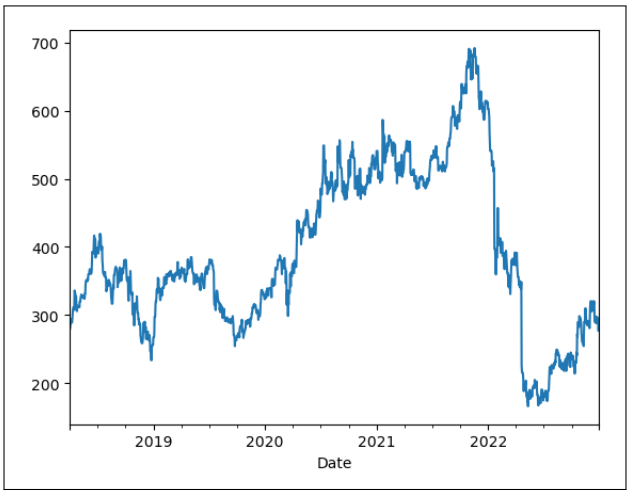


FIGURE 12 – Les données cas d'étude 2

directement opter pour le modèle ARIMA.

B.5.1 Choix de p et q

Pour cela, nous allons procéder à l'"ajustement des hyperparamètres". Ainsi, pour les valeurs possibles de p, nous allons créer une plage de 0 à 29 par incréments de 4. Ensuite, pour les paramètres de q, nous allons utiliser une plage plus restreinte, allant de 0 à 3 par incréments de 1.
Si vous vous interrogez sur la raison pour laquelle nous avons plus de paramètres p que de paramètres q, c'est parce que c'est ainsi que fonctionnent généralement les modèles ARMA. Lorsque nous considérons l'autorégression (AR), nous regardons souvent dans le passé pour prédire le présent ou le futur, d'où la nécessité d'avoir plusieurs paramètres p. En revanche, pour q, il s'agit plutôt de chocs à court terme dans le système. Ainsi, nous avons généralement moins de paramètres q, que nous maintenons à des valeurs modestes telles que un, deux, ou trois.

Ensuite, nous prendrons le dictionnaire et le transformerons en une DataFrame.
Nous explorerons ces différentes combinaisons d'hyperparamètres pour déterminer celles qui offrent les meilleures performances pour notre modèle, en utilisant une grille.
Ainsi, pour chaque combinaison de p et q, nous commencerons par entraîner un modèle avec ces hyperparamètres spécifiques. Ensuite, nous calculerons l'erreur absolue moyenne d'entraînement pour ce modèle particulier et sauvegarderons ce résultat. Nous répéterons cette étape pour chaque combinaison unique de la grille. Une fois cela accompli, nous examinerons les différentes valeurs de l'erreur absolue moyenne (MAE) et choisirons celle qui présente la meilleure performance.

	0	8	16
0	100.2119	5.4661	5.5042
1	52.2175	5.4668	5.5149
2	31.5860	5.4749	5.5177
3	22.5469	5.4995	5.5415

Donc, nous avons toutes les valeurs possibles pour p en haut de l'index, puis toutes les valeurs possibles pour q vers le côté. Ensuite, au centre, nous avons les différentes erreurs absolues moyennes pour les modèles entraînés avec ces ensembles d'hyperparamètres.On obtient la grille suivante :

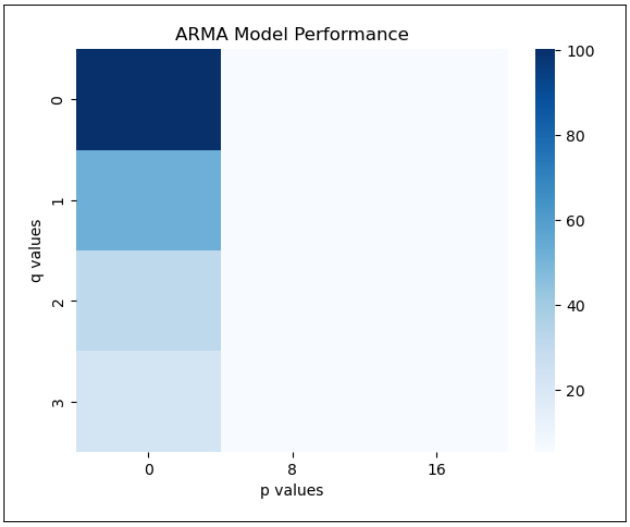


FIGURE 13 – grille des paramètres cas d'étude 2

On constate que tous les modèles avec p=0 présentent des performances médiocres. Une deuxième conclusion importante est que l'augmentation de la valeur de p n'a pas entraîné de changement significatif dans la performance du modèle. Le modèle optimal est celui où p est égal à 8 et q est égal à 0. En examinant le DataFrame de l'erreur absolue moyenne (MAE), on remarque que notre meilleur modèle, avec une MAE de 5.4661, et notre deuxième meilleur modèle, avec une MAE de 5.4668, présentent une différence presque négligeable.

B.5.2 Diagnostic du modèle

Maintenant, il reste une dernière étape à accomplir avant d'évaluer notre modèle. Nous devons examiner les résidus obtenus. Cela s'inscrit dans le cadre du diagnostic du modèle, visant à vérifier les hypothèses de normalité des erreurs et la distribution des résidus. Si les erreurs suivent une distribution normale et ne présentent pas de corrélation entre elles, cela confirmera la qualité de notre modèle.

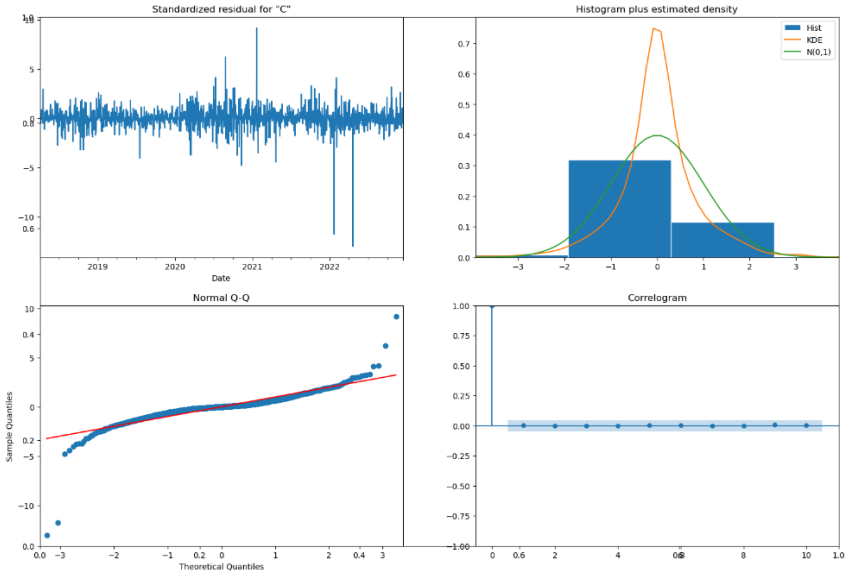


FIGURE 14 – diagnostic cas d'étude 2

Le premier graphique : représente les résidus de notre modèle. Ces résidus semblent assez bons. Ils sont centrés autour de zéro, bien qu'il y a des événements, ils restent centrés autour de zéro.
2ème graphique : (histogramme des résidus) Ici, nous voulons que les barres de l'histogramme suivent la courbe verte de la distribution normale, et il semble qu'elles le fassent plus ou moins correctement.
3ème graphique : Dansle graphique de la Normal Q-Q, tous les points devraient parfaitement s'aligner avec la ligne rouge. Des écarts significatifs impliqueraient une distribution asymétrique.
4ème graphique : Le dernier que nous voulons montrer est le graphique de corrélogramme, le graphique ACF des résidus. Il semble conforme à nos attentes. Au niveau de décalage zéro, le coefficient de corrélation a une valeur de un, puis tout descend en dessous juste à l'intérieur de la barre bleue.

C'est une excellente façon d'avoir rapidement une idée de la performance de notre modèle. Maintenant, passons à la phase d'évaluation.

B.5.3 Prédictions

Prédictions ARMA

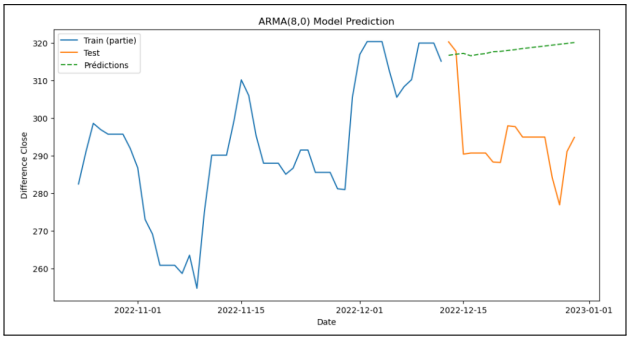


FIGURE 15 – ARMA cas d'étude 2

La valeur de l'erreur absolu moyen (MAE) s'élève à 24.301993712667244. Comme observé, le modèle ARMA ne permet pas une prédiction efficace de la série temporelle non stationnaire, présentant ainsi une MAE élevée. Cela souligne la nécessité de l'étape de différenciation, qui sera intégrée dans le modèle ARIMA.

Prédictions ARIMA

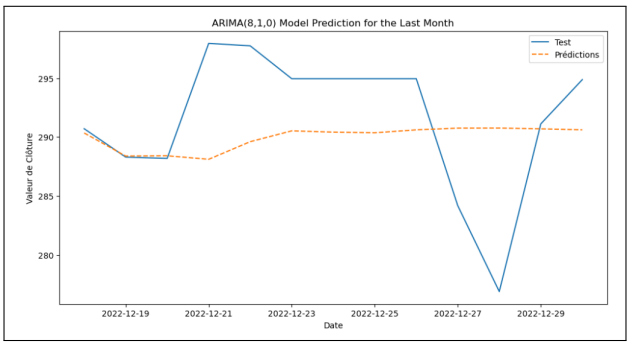


FIGURE 16 – ARIMA cas d'étude 2

La valeur de l'erreur absolue moyenne (MAE) s'élève à 4.748915402860982. Les performances du modèle ARIMA se révèlent satisfaisantes dans la prédiction de notre série sur une période de 15 jours, démontrant des prédictions pratiquement exemptes d'erreurs pour les trois premiers jours.

C prédictions avec LSTM

Dans cette partie on utilise les données du cas d'étude 2.

C.1 Introduction générale et définitions

Les LSTM (Long Short-Term Memory) introduisent une nouvelle approche dans les réseaux de neurones. La cellule mémoire d'un LSTM est constituée de plusieurs portes, notamment une porte d'entrée, une porte de sortie et une porte d'oubli. Ces portes régulent le flux d'informations à l'intérieur de la cellule mémoire, permettant ainsi de contrôler quelles informations doivent être mémorisées et quelles informations doivent être oubliées. Cette capacité confère aux LSTM la faculté de retenir des informations importantes sur de longues séquences, tout en ignorant les éléments moins pertinents. La composante clé de ce mécanisme est la cellule mémoire, qui autorise le réseau à stocker et à accéder à des informations sur une période prolongée.

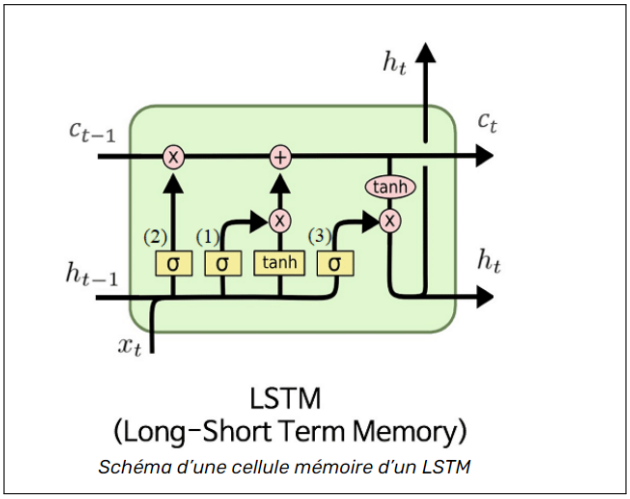


FIGURE 17 – Schéma d'une cellule mémoire d'un LSTM

La porte d'entrée (*Input Gate*) détermine quelles informations doivent être mises à jour dans la cellule mémoire. Elle prend en compte l'entrée actuelle x_t ainsi que l'état précédent de la cellule mémoire c_{t-1} et génère un vecteur d'activation qui représente les informations à ajouter à la cellule c_{t-1}. Cet ajout d'informations se traduit par une opération mathématique effectuée entre ce vecteur d'activation et l'état précédent c_{t-1}.

La porte d'oubli (*Forget Gate*) (2) permet au LSTM de supprimer les informations obsolètes de la cellule mémoire. Elle utilise à la fois l'entrée actuelle et l'état précédent pour générer un vecteur d'activation qui détermine quelles informations doivent être oubliées. Cela se traduit également par une opération mathématique effectuée entre ce deuxième vecteur d'activation et l'état précédent c_{t-1}. C'est à partir des deux opérations précédentes sur c_{t-1} que l'on obtient l'état actuel c_t.

Enfin, la porte de sortie (*Output Gate*) (3) détermine la sortie du LSTM à un instant donné. Elle utilise l'entrée actuelle x_t et l'état actuel de la cellule mémoire c_t pour générer un vecteur d'activation qui re-

présente la sortie du LSTM. C'est ainsi que l'on obtient h_t.

La combinaison de ces trois portes permet au réseau LSTM de gérer efficacement les dépendances à long terme. Lors de la rétropropagation du gradient, les LSTM peuvent maintenir un flux d'informations constant à travers le temps, évitant ainsi le problème du "vanishing gradient" et permettant un apprentissage plus stable et plus précis.

Fonctionnement :

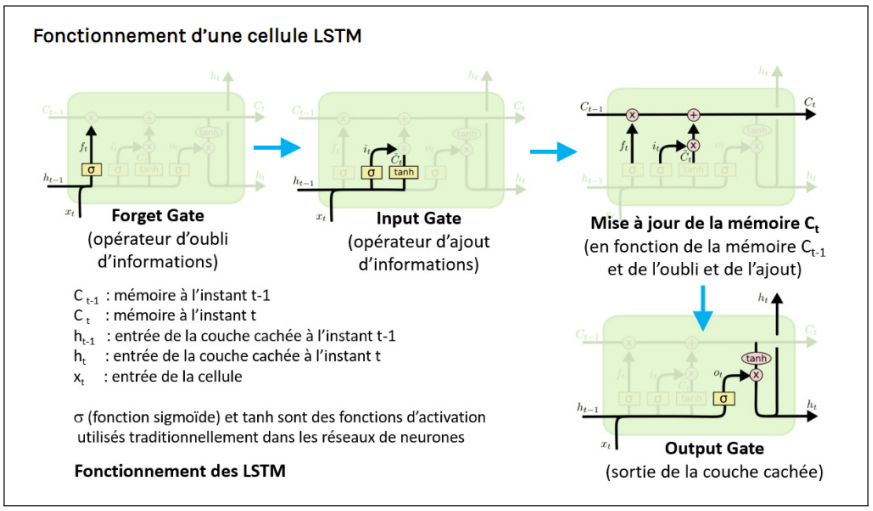


FIGURE 18 – Fonctionneemnt d'un LSTM

Stateful vs Stateless LSTM

Stateless (Sans état) : Les LSTM mettent à jour les paramètres sur le lot 1 puis initient les états des cellules (c'est-à-dire la mémoire, généralement avec des zéros) pour le lot 2.
Stateful (Avec état) : Il utilise les derniers états de sortie du lot 1 comme états initiaux pour le lot 2.

Quand utiliser quoi ?

Lorsque les séquences dans les lots sont liées les unes aux autres (par exemple, les prix d'une même marchandise), il est préférable d'utiliser le mode stateful. Sinon, lorsque chaque séquence représente une phrase complète, il est conseillé d'utiliser le mode stateless.

Taille du lot (Batch-size) : quelle taille de lot choisir ?

Imaginez, vous devez apprendre à reconnaître un oiseau... On vous présente des images d'oiseaux différents.
Que préféreriez-vous :

— Voir une image à la fois, prendre des notes sur les caractéristiques particulières de l'oiseau (définir vos poids) et ensuite voir un autre oiseau, et ainsi de suite.
— OU peut-être préféreriez-vous apprendre plus rapidement si vous voyiez - disons 5 - images d'oiseaux à la fois. Peut-être que vous pourriez alors remarquer plus rapidement les propriétés intrinsèques de l'oiseau ?

Cette analogie reflète le choix de la taille du lot (*batch size*) dans les réseaux de neurones. Le batch size représente le nombre d'échantillons d'entraînement utilisés dans une itération pour mettre à jour les poids du réseau. En ajustant cette taille de lot, on peut influencer la vitesse d'apprentissage et la capacité du modèle à généraliser à partir des données d'entraînement. Un petit batch size peut conduire à une mise à jour de poids plus fréquente mais peut être plus coûteux en termes de calcul, tandis qu'un batch size plus grand peut accélérer le processus d'apprentissage, mais au détriment de la fréquence des mises à jour des poids.
On choisit donc le paramètre *batch size* avec une valeur de 64. Le *batch size* indique le nombre d'échantillons d'entraînement utilisés dans une itération pour mettre à jour les poids du modèle. Dans ce cas, chaque mise à jour des poids se fait sur un lot de 64 échantillons.

Pour le paramètre *epochs*, nous le fixons à 120. Les *epochs* représentent le nombre de fois où l'ensemble complet des données d'entraînement est passé à travers le modèle. Chaque epoch se compose d'une série d'itérations où le modèle est ajusté avec les données du lot. Un nombre élevé d'époques offre au modèle davantage d'occasions d'apprendre des données d'entraînement.

Concernant le paramètre *timesteps*, nous le réglons sur 30. Les *timesteps* représentent la longueur de la séquence temporelle ou la longueur de la fenêtre que le modèle considère à chaque itération. Pour les données temporelles, chaque séquence peut être découpée en tranches de *timesteps*. Cela est particulièrement important pour les modèles récurrents, tels que les LSTM, qui maintiennent une mémoire à long terme à travers les pas de temps. Un choix approprié de *timesteps* dépend de la nature des données temporelles et de la complexité du modèle. **Anatomie d'un noeud :**

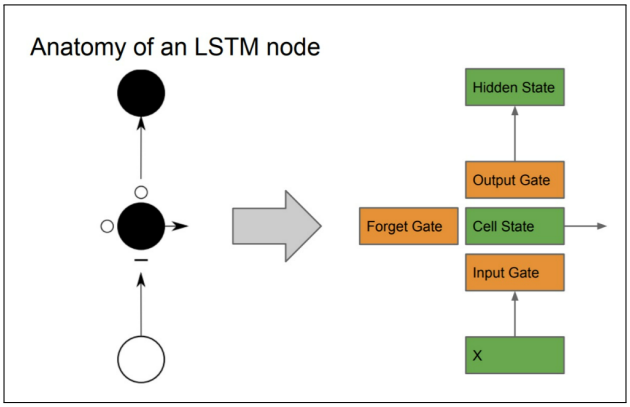


FIGURE 19 – Anatomie d'un noeud LSTM

Comment le nombre de paramètres LSTM est calculé ?

— Pour décider comment gérer la mémoire, chaque cellule LSTM possède, comme cité au début, **3**

portes :
  - entrée (ce qu'il faut laisser entrer),
  - oubli (ce qu'il faut oublier) et
  - sortie (ce qu'il faut écrire en sortie).
- L'état de la cellule LSTM est sa **mémoire**.
- L'état caché de LSTM est équivalent à la sortie de la cellule :
  - taille_de_l'état_caché_lstm (nombre de neurones = cellules de mémoire) = taille_des_sorties_lstm.
- Paramètres :
  - poids pour les entrées (taille_des_entrées_lstm),
  - poids pour les sorties (taille_des_sorties_lstm),
  - variable de biais.
- Résultat du point précédent - pour les 3 portes et pour l'état de la cellule (= 4)

$$\textbf{PARAMÈTRES} = 4\times \textbf{Taille des sorties LSTM}\times(\textbf{taille des poids d'entrée LSTM}+\textbf{taille des poids de sortie} \tag{1}$$

C.2 Configuration du modèle

Voici le résumé du modèle pour lequel on a opté :

Layer (type)	Output Shape	Param #	
input_1 (InputLayer)	[(64, 30, 1)]	0	
lstm	(64, 30, 10)	480	
lstm_1	(64, 30, 10)	840	
dense	(64, 30, 1)	11	
Total params :	**1331 (5.20 KB)**		
Trainable params :	**1331 (5.20 KB)**		
Non-trainable params :	**0 (0.00 Byte)**		

.

On a opté pour l'utilisation de l'API fonctionnelle de Keras afin de construire un modèle de réseau de neurones récurrents (LSTM) dédié à la prédiction temporelle. La configuration du modèle comprend deux couches LSTM, chacune équipée de 10 unités de mémoire, suivies d'une couche dense pour la sortie. L'optimiseur Adam a été sélectionné pour ajuster les poids du modèle lors de l'entraînement, et la fonction de perte Mean Absolute Error (MAE) est employée pour évaluer la disparité entre les prédictions du modèle et les valeurs réelles. L'utilisation de l'API fonctionnelle confère la possibilité de définir des architectures de modèles complexes et flexibles. Le résumé du modèle expose en détail chaque couche, y compris le nombre de paramètres, fournissant ainsi une vue d'ensemble de la structure du réseau neuronal.

C.3 Prédictions et évaluations

On construit l'ensemble de test pour évaluer la performance du modèle. Tout d'abord, les données sont subdivisées pour extraire la partie correspondant à l'ensemble de test. Ensuite, la colonne contenant les

valeurs à prédire est isolée et les valeurs sont normalisées. Par la suite, une séquence d'entrée, X_{test}, est construite en utilisant des fenêtres glissantes de la taille spécifiée par la variable timesteps. Chaque fenêtre représente une séquence temporelle qui sera fournie au modèle pour effectuer une prédiction. Enfin, les dimensions de X_{test} sont ajustées pour convenir au format d'entrée du modèle LSTM. L'ensemble de test ainsi construit est prêt à être utilisé pour évaluer la performance du modèle sur des données qu'il n'a pas vues pendant l'entraînement. Les opérations de sous-ensemble, de normalisation et de préparation des séquences sont des étapes cruciales pour garantir une évaluation précise du modèle sur des données réelles. On trouve les prédictions suivantes :

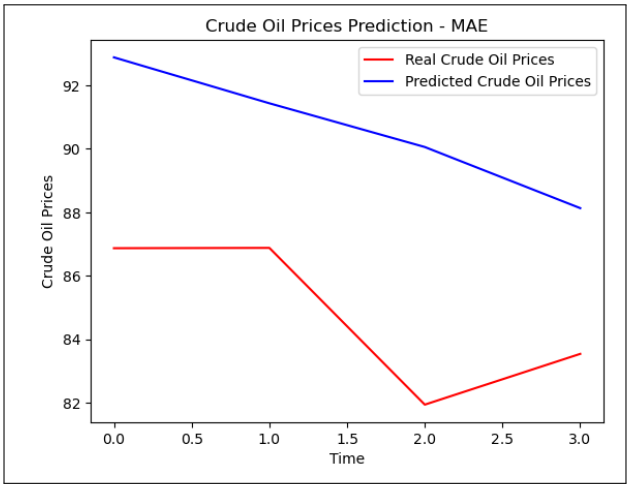


FIGURE 20 – Visualitions des prédictions

Pour évaluer la précision des prédictions d'un LSTM après la phase d'entraînement, on procède à une phase de test au cours de laquelle on mesure la disparité entre les observations réelles et les prédictions du modèle. Différentes métriques de mesure sont utilisées, et fréquemment le RMSE (Root Mean Square Error), qui représente la moyenne arithmétique des carrés des écarts entre les prévisions du modèle et les observations. Dans ce cas particulier, on obtient une valeur de 5.999044745678759 pour le RMSE.
La RMSE (Root Mean Square Error) et la MSE (Mean Squared Error) sont définies par les expressions suivantes :
RMSE (Root Mean Square Error) :

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

MSE (Mean Squared Error) :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Ces métriques de performance ont plusieurs propriétés importantes à prendre en considération :

— Elles donnent une importance plus élevée aux grandes erreurs qu'aux petites. En raison du carré dans la somme, une erreur trois fois plus grande aura neuf fois plus d'impact sur la moyenne, les rendant ainsi sensibles aux outliers.

— Les outliers, qui représentent des valeurs aberrantes dans les données réelles, peuvent fortement influencer ces métriques, car la prédiction est souvent très éloignée de ces valeurs aberrantes.
— Elles sont facilement optimisables, car ce sont des métriques dérivables. Elles peuvent être utilisées avec des algorithmes basés sur le gradient, et dans certains cas, une formule explicite peut être obtenue sans nécessiter d'optimisation stochastique, comme dans le cas de la régression par moindres carrés ordinaires (MCO).

Maintenant, nous modifions le modèle pour l'initialiser avec la loss MSE. Alors :

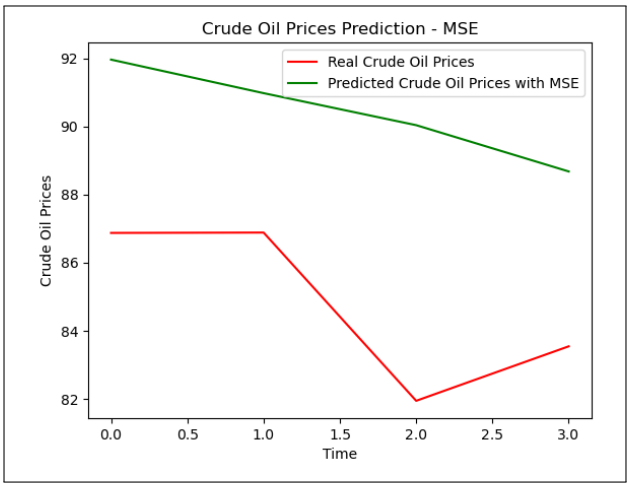


FIGURE 21 – Visualisation des prédictions aprés modification

La valeur de l'erreur absolue moyenne (MAE) est de 5.602266311645508, indiquant ainsi une amélioration de la précision de la prédiction.

Conclusion

En conclusion, notre projet axé sur la prédiction des cours boursiers du pétrole brut a été l'occasion d'appliquer divers concepts de la théorie des signaux et systèmes, notamment en explorant les modèles AR, MA, ARMA, ARIMA, ainsi que les réseaux de neurones LSTM. Cette expérience nous a permis d'approfondir nos compétences en analyse de séries temporelles et en modélisation prédictive. Les résultats obtenus et les comparaisons entre les différentes approches nous ont offert des perspectives intéressantes sur l'efficacité des modèles étudiés dans le contexte financier. Ce projet a été une opportunité enrichissante pour appliquer nos connaissances théoriques à des données réelles et renforce notre compréhension des techniques de prédiction dans le domaine des marchés financiers.

D Webographie

https ://medium.com/towards-data-science/a-brief-introduction-to-time-series-forecasting-using-statistical-methods-d4ec849658c3

https ://www.kaggle.com/code/iamleonie/time-series-interpreting-acf-and-pacf/notebook

https ://ilyasbinsalih.medium.com/what-are-acf-and-pacf-plots-in-time-series-analysis-cb586b119c5d

https ://medium.com/@sawsanyusuf/forecasting-time-series-data-with-arma-model-949fb9ef8460
https ://medium.com/@evertongomede/forecasting-non-stationary-time-series-03b638a7cd50

https ://towardsdatascience.com/introduction-to-time-series-forecasting-part-2-arima-models-9f47bf0f476b

https ://towardsdatascience.com/introduction-to-time-series-forecasting-part-1-average-and-smoothing-models-a739d832315

https ://medium.com/@JDEconomics/how-to-test-for-stationarity-in-time-series-data-using-python-44d82890aa9d