UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR

Projet de Machine Learning

ML/S6/2024 – P N°1

Prédictibilité de l'indice boursier S&P 500 : Une analyse comparative des modèles de machine learning

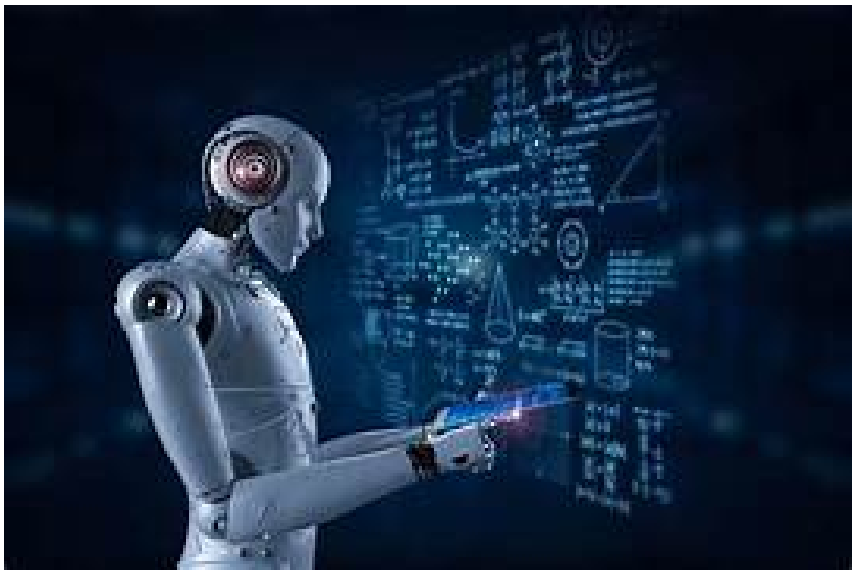
Réalisé par :

Boubacar KANDE

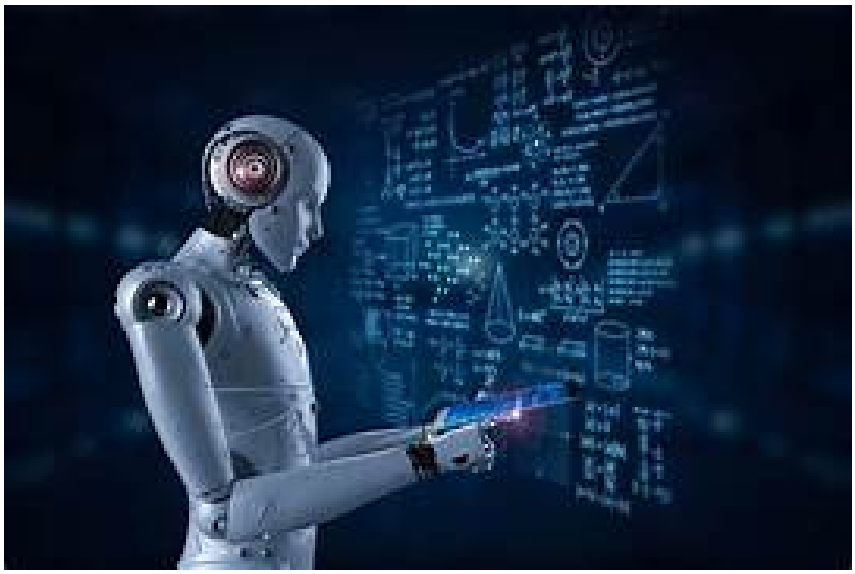
Responsables du cours :

Jamal BOUOIYOUR

Hamou ELOTMANY



Année Universitaire : 2023/2024



Date de remise du rapport : 30/05/2024

Référence du projet : ML/S6/2024 – P N°1.

Intitulé du projet : Prédictibilité de l'indice boursier S&P500 : Une analyse comparative des modèles de machine learning

Type de projet : Simulation et modélisation des données

Objectifs du projet : Ce projet de Machine Learning se concentre sur la prédiction de l'indice boursier S&P 500 en utilisant différents modèles, notamment ARMA, RNN, LSTM et GRU. Après avoir entraîné et testé ces modèles, les résultats montrent que les réseaux neuronaux surpassent largement le modèle ARMA en termes de précision. Plus spécifiquement, les modèles LSTM et GRU se révèlent légèrement meilleurs que le RNN. En conclusion, ce projet démontre l'efficacité des modèles de réseaux neuronaux pour la prédiction des séries temporelles financières, avec une préférence notable pour LSTM et GRU.

Mots-clés du projet : simulation, prévision, réseau neurone, machine learning

N° cahier de laboratoire associé : UMR 6031 TREE

Table des matières

NOTATIONS, ACRONYMES

ADF Augmented Dickey-Fuller

ARMA Auto-Regressive Moving Average

ELU Exponential Linear Unit

GRU Gated Recurrent Unit

Leaky ReLU Leaky Rectified Linear Unit

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MSE Mean Squared Error

R2 Coefficient of Determination

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

PReLU Parametric Rectified Linear Unit

sec Seconde

Tanh Tangente Hyperbolique

Liste des tableaux

Table des figures

1 Introduction

La prévision des marchés financiers a longtemps captivé l'intérêt des chercheurs et des praticiens, étant donné son potentiel pour générer des rendements élevés et pour informer les décisions économiques. L'indice S&P 500, en particulier, est un baromètre clé de la santé économique des États-Unis, représentant 500 des plus grandes entreprises cotées en bourse. La capacité de prédire les mouvements de cet indice est donc d'une importance cruciale pour les investisseurs, les gestionnaires de fonds et les décideurs politiques.

Historiquement, les modèles de séries temporelles traditionnels, tels que les modèles ARMA (AutoRegressive Moving Average), ont été largement utilisés pour la prévision des marchés financiers. Ces modèles se fondent sur des hypothèses linéaires et des structures de dépendance temporelle bien définies (Box & Jenkins, 1976). Cependant, les dynamiques des marchés financiers sont souvent non linéaires et influencées par une multitude de facteurs exogènes, rendant les approches traditionnelles parfois insuffisantes pour capturer la complexité des mouvements de marché.

Avec l'avènement des techniques de machine learning, de nouveaux modèles ont émergé, promettant des améliorations significatives en termes de précision prédictive. Parmi ces modèles, les réseaux de neurones récurrents (RNN), les réseaux de neurones à mémoire à long terme (LSTM) et les unités récurrentes fermées (GRU) se sont distingués par leur capacité à modéliser des séquences temporelles complexes et à capturer des dépendances à long terme (Hochreiter & Schmidhuber, 1997 ; Cho et al., 2014). Ces modèles ont montré des performances prometteuses dans divers domaines, allant de la reconnaissance vocale à la prévision météorologique, et plus récemment, à la prédiction des marchés financiers.

Les RNN, conçus pour traiter des données séquentielles, sont capables de retenir des informations sur de longues séquences temporelles. Cependant, ils sont souvent limités par le problème de gradient évanescent, rendant difficile l'apprentissage de dépendances à long terme (Bengio et al., 1994). Pour remédier à cela, les LSTM et les GRU ont été développés, intégrant des mécanismes de mémoire sophistiqués permettant de conserver des informations sur de plus longues périodes et de gérer efficacement les dépendances temporelles complexes.

L'objectif de cette étude est de réaliser une analyse comparative de la performance prédictive de différents modèles de machine learning pour la prévision de l'indice S&P 500. En particulier, nous comparerons les performances des modèles ARMA, RNN, LSTM et GRU. Cette comparaison permettra d'évaluer dans quelle mesure les modèles avancés de machine learning surpassent les modèles traditionnels de séries temporelles dans le

contexte de la prédiction des marchés financiers.

2 Méthodologie et organisation du travail

La méthodologie de cette étude est structurée de manière à permettre une comparaison rigoureuse et approfondie des performances prédictives des différents modèles de séries temporelles et de machine learning appliqués à l'indice S&P 500. La démarche adoptée se divise en plusieurs étapes clés :

- Nous définissons quelques concepts clés et faire une étude des fonctions d'activations en réseau neurone
- Une analyse exploratoire des données
- Application sur les différents modèles et comparaison en terme de prévision

Ainsi, l'organisation et les étapes suivies pour réaliser le travail de prédiction de l'indice boursier S&P 500 à l'aide des modèles ARMA, RNN, LSTM et GRU est la suivante.

2.1 Organigramme des Tâches Réalisées

Le travail a été structuré en plusieurs tâches clés, comme illustré dans les organigrammes ci-dessous. Nous aurons une partie de simulation qui nous servira pour évaluer la performance des fonctions d'activations. Car les fonctions d'activations jouent un role crucial dans les algorithmes de machine learning. Pour éviter les choix arbitraire de fonctions d'activations dans des projets de machines learning, nous étudions d'abord ces dernières. Dans un second temps, nous construisons des modèles de machine learning sur une série unvariée qui est notre cas l'indice boursier S&P 500

FIGURE 1 – L'organigramme pour la partie simulation et etude de fonction d'activation

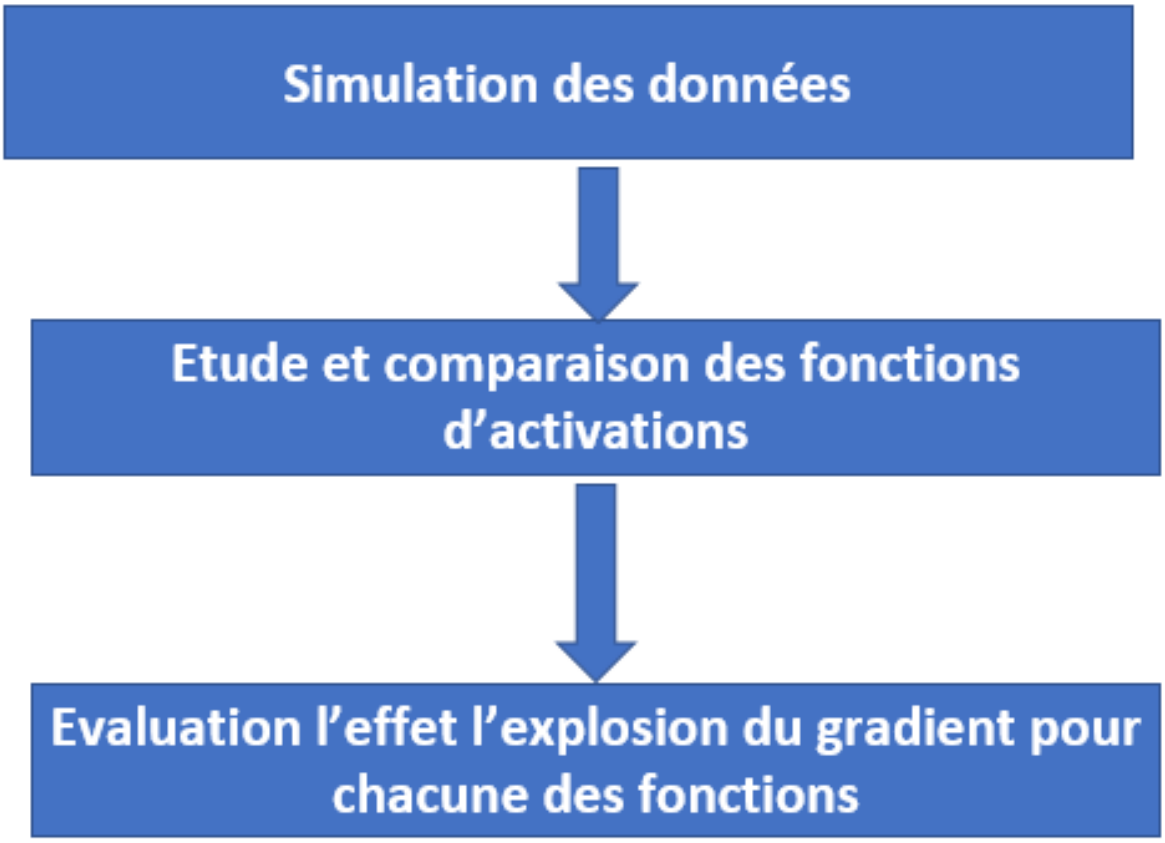
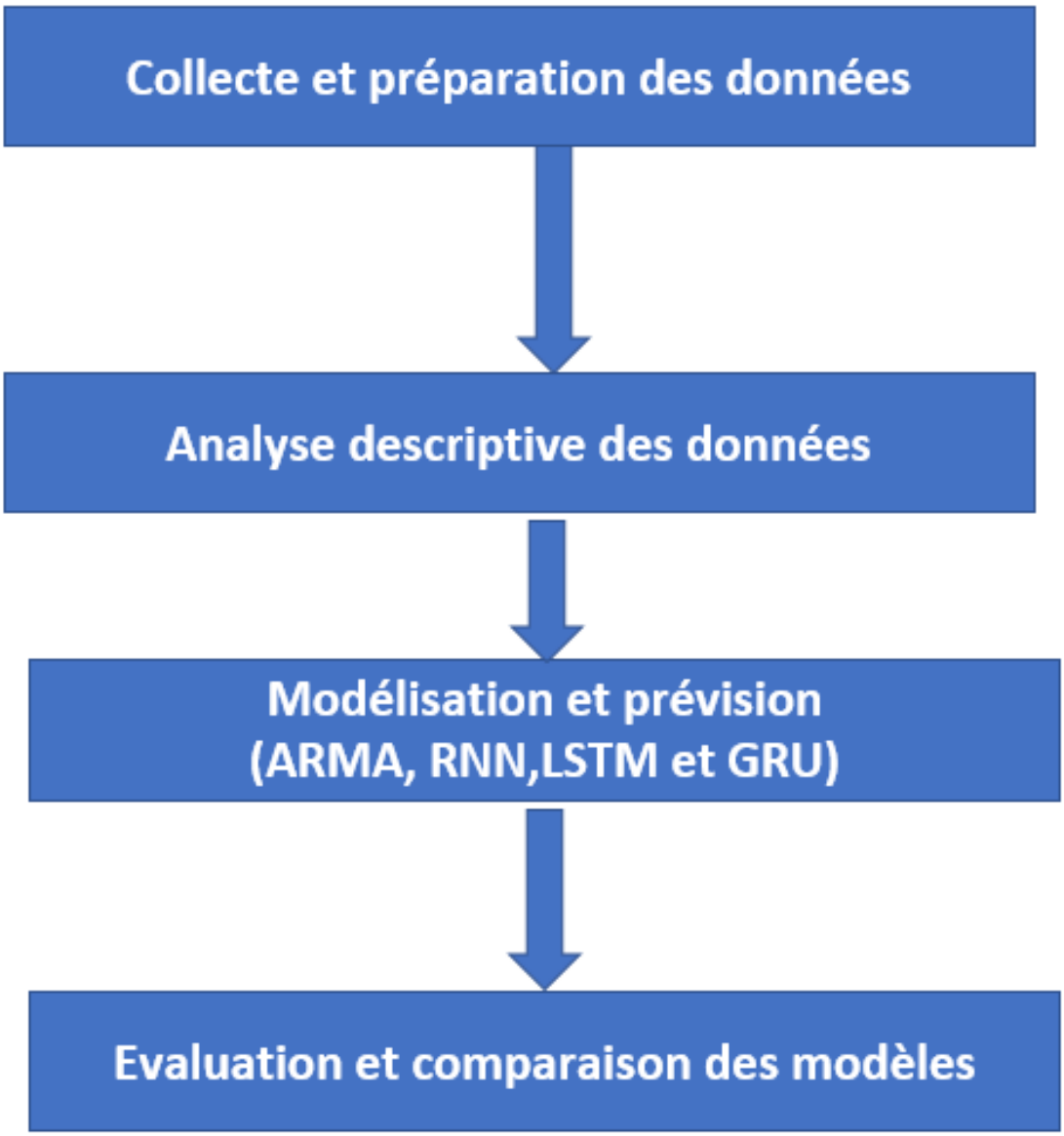


FIGURE 2 – L'organigramme pour la partie application sur les données réelles



2.2 Description des Données et Source de Données

Les données utilisées dans cette étude sont les prix de clôture journaliers de l'indice S&P 500. Les principales caractéristiques des données incluent les valeurs de clôture, les volumes de transactions et entre autes. Les données ont été collectées à partir de Yahoo Finance de façon automatisée via le package python **yfinance**. La période couverte par les données s'étend sur plusieurs années de 3 janvier 2000 à 30 avril 2024 soit 6120 observations pour assurer une analyse robuste et représentative.

2.3 Modèles à Utiliser pour la Modélisation et la Prévision

Pour aborder la problématique de prédiction de l'indice S&P 500, une compréhension approfondie des concepts suivants est nécessaire :

Modèles ARMA (AutoRegressive Moving Average) : Ces modèles combinent les aspects autorégressifs (AR) et de moyenne mobile (MA) pour capturer les dynamiques temporelles linéaires des séries de données.

Réseaux de Neurones Récurrents (RNN) : Les RNN sont adaptés pour modéliser les séquences temporelles grâce à leur capacité à conserver les informations de l'état précédent, permettant ainsi de capturer les dépendances temporelles.

Réseaux de Neurones à Longue Mémoire à Court Terme (LSTM) : Les LSTM sont une variante des RNN conçus pour surmonter les problèmes de disparition et d'explosion des gradients, facilitant la modélisation des dépendances à long terme dans les séries temporelles.

Unités Récurrentes Gated (GRU) : Les GRU sont une autre variante des RNN qui simplifient l'architecture des LSTM tout en conservant la capacité à gérer les dépendances temporelles de longue durée.

2.4 Outils et Techniques Utilisés pour Répondre aux Besoins du Projet

Les outils et techniques suivants ont été utilisés pour mener à bien ce projet :

Environnements de Développement : Python avec des bibliothèques comme Pandas, NumPy, Scikit-learn pour la manipulation et l'analyse des données, et TensorFlow pour la mise en œuvre des modèles de machine learning.

Techniques de Prétraitement des Données : Nettoyage des données, normalisation, et création de caractéristiques pour améliorer la qualité et la pertinence des données pour la modélisation.

Métriques d'Évaluation : Utilisation de métriques telles que la Root Mean Squared Error (RMSE) et la Mean Absolute Error (MAE) pour évaluer et comparer les performances des différents modèles.

Visualisation des Données : Utilisation de bibliothèques comme Matplotlib et Seaborn pour visualiser les tendances, les distributions et les résultats des prédictions.

En adoptant cette organisation méthodique et en utilisant ces outils et techniques, l'étude vise à fournir une évaluation comparative rigoureuse et approfondie des modèles de prédiction appliqués à l'indice S&P 500.

3 Définition et étude des fonctions d'activations en réseau neurone

3.1 Définitions

3.1.1 Couche (layer)

Une couche dans un réseau de neurones est une collection de neurones qui opère simultanément à une même étape du traitement des données. Les réseaux de neurones sont constitués de plusieurs couches empilées les unes sur les autres, permettant ainsi une hiérarchisation et une abstraction progressive des caractéristiques. Les principaux types de couches sont :

- **Couche d'entrée (Input Layer)** : Reçoit les données brutes à traiter.
- **Couches cachées (Hidden Layers)** : Effectuent des transformations intermédiaires sur les données, capturant des caractéristiques complexes.
- **Couche de sortie (Output Layer)** : Produit le résultat final du réseau.

Chaque couche est connectée à la couche suivante par des poids, et les valeurs des neurones dans une couche sont calculées par une fonction d'activation appliquée aux sommes pondérées des entrées de la couche précédente.

3.1.2 Neurone (Neuron)

Un neurone dans un réseau de neurones artificiels est une unité de base qui reçoit des entrées, les transforme à l'aide de poids, applique une fonction d'activation et produit une sortie. Les neurones sont inspirés des neurones biologiques dans le cerveau. La sortie d'un neurone est calculée comme suit :

$$y = f(\sum_i w_i x_i + b)$$

où x sont les entrées, w_i les poids associés, b le biais, et f la fonction d'activation. Les neurones dans les couches cachées et la couche de sortie permettent au réseau d'apprendre et de généraliser des relations complexes dans les données.

3.1.3 Vanishing Gradient (Disparition du Gradient)

Le phénomène de "vanishing gradient" ou disparition du gradient se produit principalement lors de l'entraînement de réseaux de neurones profonds à l'aide de l'algorithme de rétropropagation. Il survient lorsque les gradients des couches d'un réseau deviennent extrêmement petits, ralentissant considérablement l'apprentissage ou l'arrêtant complètement. Ce problème est particulièrement fréquent dans les réseaux récurrents et les réseaux de neurones profonds.

Lorsque les gradients deviennent trop petits, les poids des couches précédentes ne se mettent pas à jour de manière significative pendant l'entraînement. Cela est souvent dû à l'utilisation de fonctions d'activation comme la sigmoid ou la tanh, dont les dérivées sont très faibles pour des valeurs extrêmes des entrées.

Des techniques comme les réseaux de neurones à mémoire à long terme (LSTM) et les unités récurrentes fermées (GRU) ont été développées pour atténuer ce problème en introduisant des mécanismes de régulation des gradients, permettant ainsi un apprentissage plus efficace sur de longues séquences.

3.2 Etude des fonctions d'activations

Une fonction d'activation dans un réseau de neurones artificiels est une fonction mathématique appliquée au signal de sortie d'un neurone. Son rôle est de faire en sorte à ce que le neurone soit activé ou non.

La fonction d'activation est essentielle pour apprendre et comprendre des données complexes, car elle introduit des propriétés non-linéaires au réseau.

TABLEAU 1 – Tableau des fonctions d'activation avec équations

Fonctions	**Équation**	**Plage de Valeurs**	**Dérivée**
ReLU	$f(x) = \max(0, x)$	$x \geq 0$	$f'(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$
Leaky ReLU	$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha x & \text{sinon} \end{cases}$	$-\infty < x < \infty$	$f'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ \alpha & \text{sinon} \end{cases}$
PReLU	$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha x & \text{sinon} \end{cases}$	$-\infty < x < \infty$	$f'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ \alpha & \text{sinon} \end{cases}$
ELU	$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha(e^x - 1) & \text{sinon} \end{cases}$	$-\infty < x < \infty$	$f'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ f(x) + \alpha & \text{sinon} \end{cases}$
Swish	$f(x) = \frac{x}{1+e^{-x}}$	$-\infty < x < \infty$	$f'(x) = f(x) + \frac{1-f(x)}{1+e^{-x}}$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$0 < f(x) < 1$	$f'(x) = f(x)(1 - f(x))$
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$-1 < f(x) < 1$	$f'(x) = 1 - (f(x))^2$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	$0 < f(x_i) < 1$	$\frac{\partial f(x_i)}{\partial x_i} = f(x_i)(1 - f(x_i))$

Pour comparer la performance de chacune des fonctions d'activations, nous avons effectuer une simulation de données avec 10 variables entrées, une variable de sortie pour 1000 observations. Les résultats des simulations sont consigné dans le tableau suivent (note book voir mon Github).

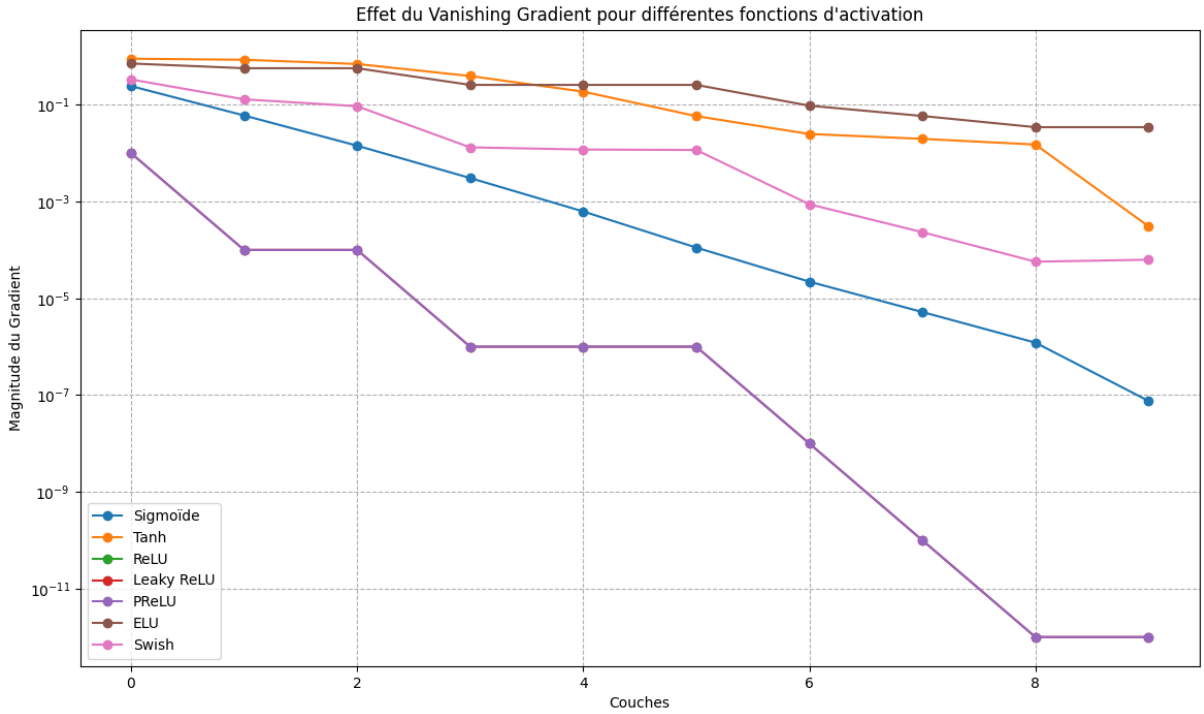
TABLEAU 2 – Performances des différentes fonctions d'activation

Fonction d'Activation	**Précision (Accuracy)**	**Perte (Loss)**	**Temps d'Entraînement (sec)**
Sigmoid	0.4700	0.7036	5.89
Tanh	0.5550	0.6912	6.93
ReLU	0.4850	0.7521	5.68
Leaky ReLU	0.5250	0.7069	6.23
PReLU	0.4600	0.7203	5.12
ELU	0.5350	0.7193	6.17
Swish	0.5000	0.7314	6.02

Nous constatons que le modèle utilisant la fonction d'activation Tanh a la meilleure

précision, suivie par ELU, Leaky ReLU, ReLU, Swish, Sigmoid et enfin PReLU. Tanh a également la perte (Loss) la plus basse, ce qui indique qu'il pourrait mieux généraliser les données que les autres fonctions d'activation dans ce contexte.

FIGURE 3 – Effet du vanishing gradien pour les différentes fonctions d'activations



L'objectif de ce graphique est de montrer comment différentes fonctions d'activation affectent la stabilité de l'apprentissage en profondeur. Les fonctions Sigmoid et PReLU ont tendance à provoquer un "vanishing gradient" (diminution rapide de la magnitude du gradient) à mesure que l'on s'enfonce dans les couches du réseau. En revanche, les fonctions ReLU, Leaky ReLU, Swish et Tanh sont moins sujettes à ce problème.

En somme, ce graphique peut aider à choisir la meilleure fonction d'activation pour un réseau de neurones profond, en tenant compte de la stabilité de l'apprentissage et de la propagation des gradients.

4 Description du problème et l'objectif

4.1 Problématique et objectif, description des données

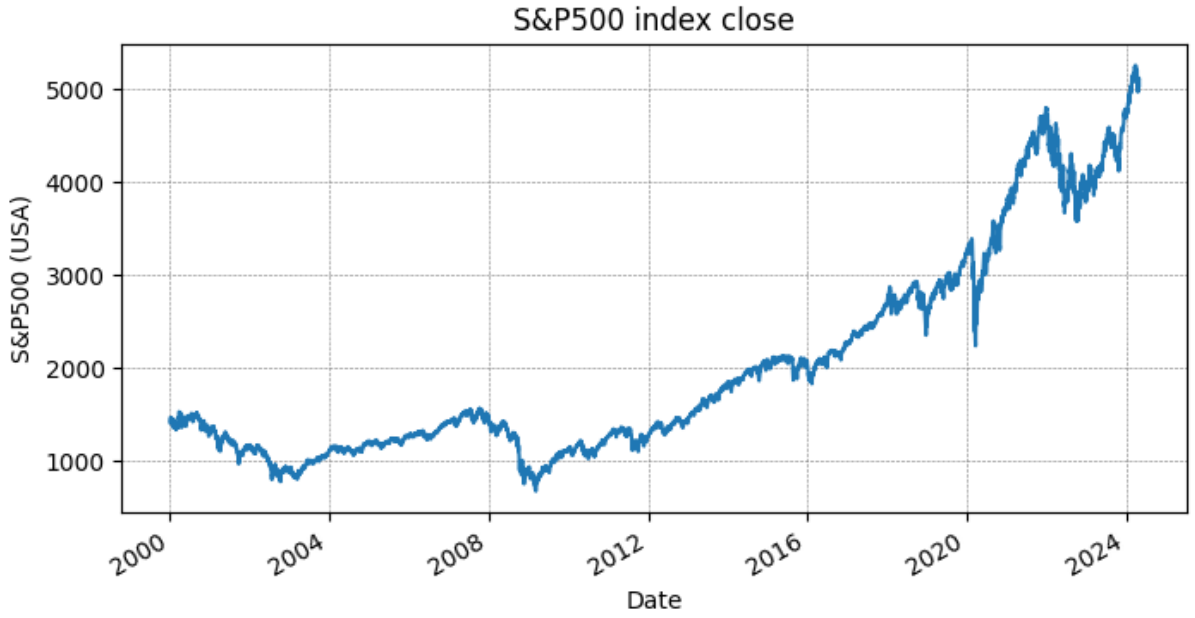
La problématique de cette étude est de déterminer quel modèle parmi ARMA, RNN, LSTM et GRU offre les meilleures performances prédictives pour l'indice S&P 500. L'objectif principal est de comparer ces modèles en termes de précision prédictive et de robus-

tesse, afin d'identifier celui qui peut le mieux capter les dynamiques complexes du marché financier.

4.2 Analyse descriptive de la base de données

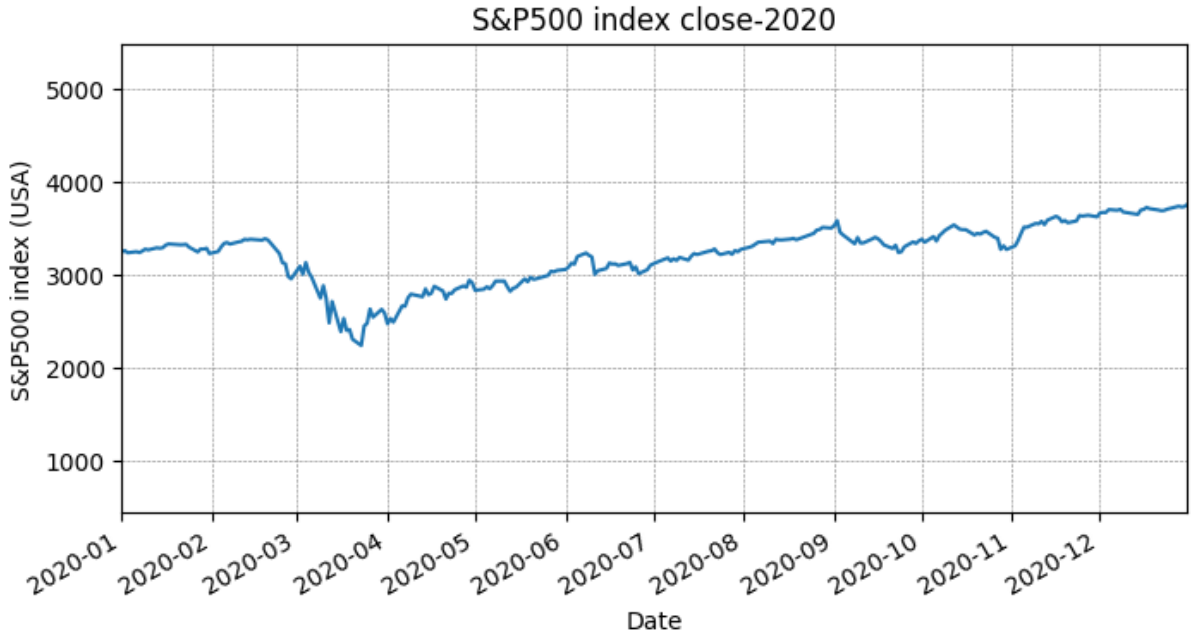
4.2.1 Visualisation de la série

FIGURE 4 – Evolution de notre série sur la période 2000-01-03 à 2024-04-30



La courbe montre une tendance générale à la hausse, avec quelques fluctuations. Cela indique une augmentation de la valeur de l'indice S&P 500 au fil de la période observée. L'indice S&P 500 est souvent utilisé comme indicateur économique et comme référence pour la performance des investissements.

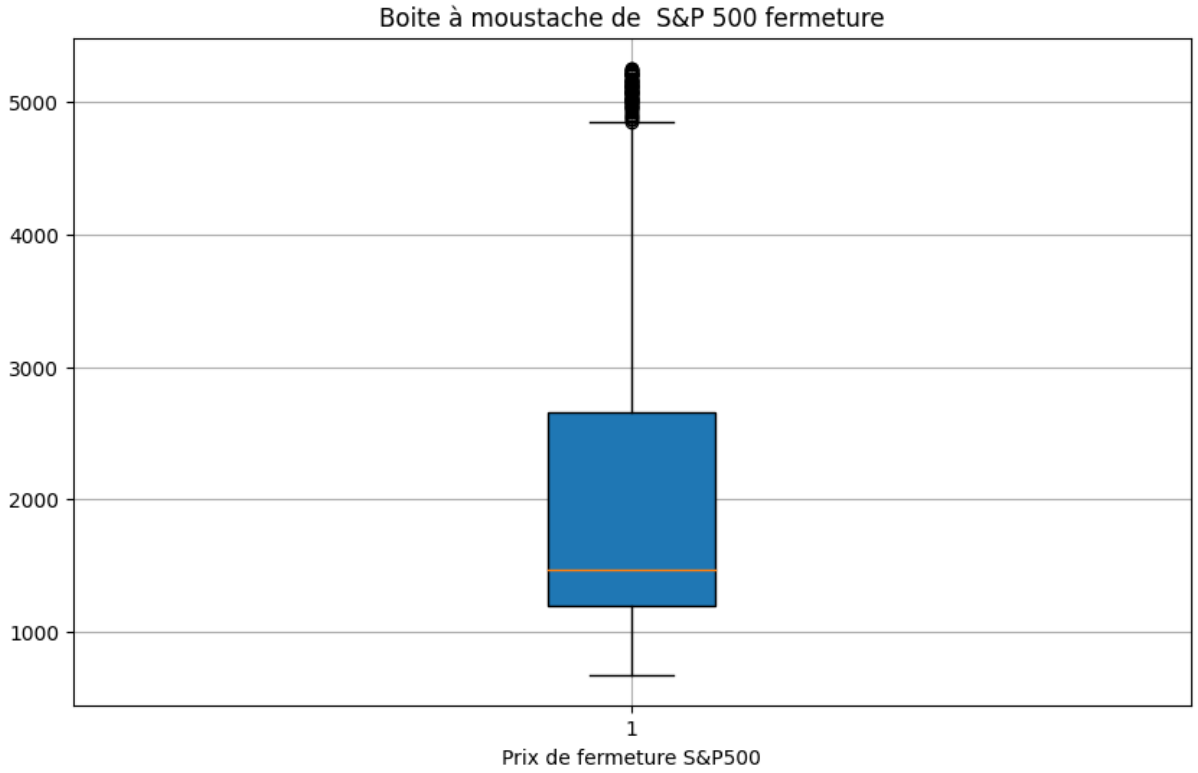
FIGURE 5 – Evolution de notre série au cours de l'année 2020



Ce graphique montre la volatilité du marché boursier au cours de 2020, mettant particulièrement en évidence l'impact des événements autour de mars, probablement liés aux changements économiques mondiaux dus à la pandémie de COVID-19.

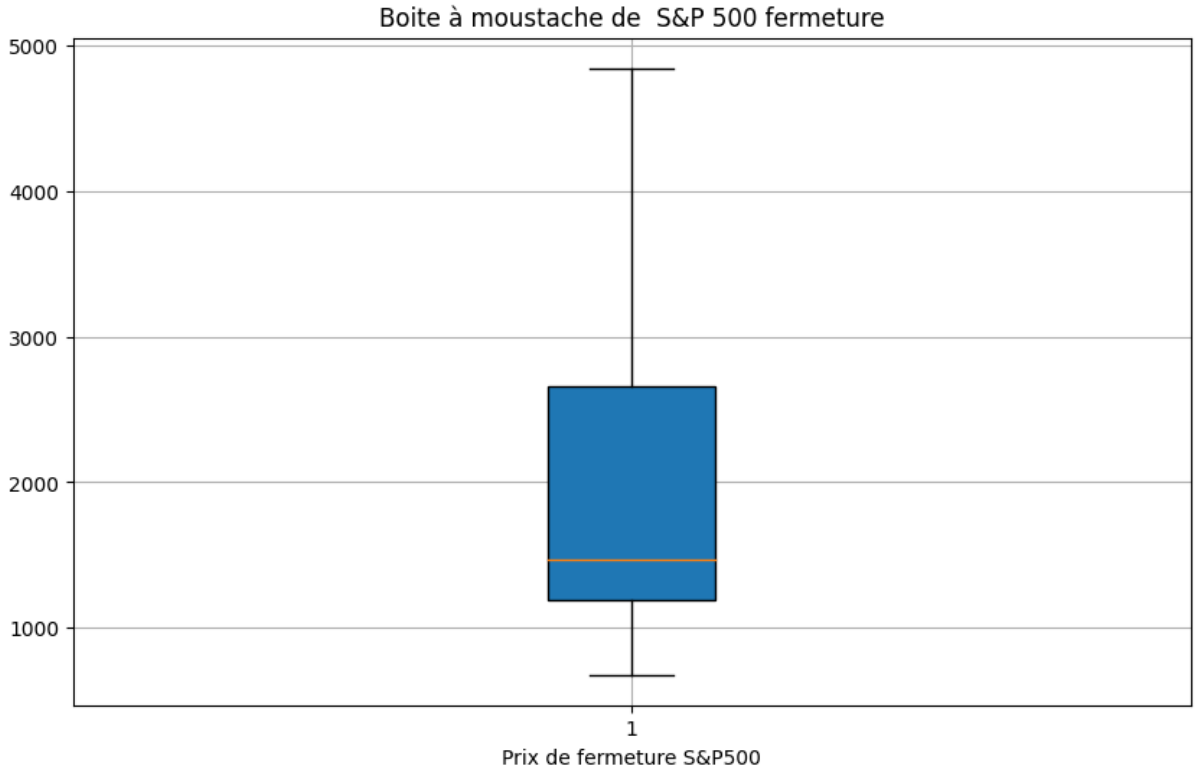
4.2.2 Traitement des données aberrantes

FIGURE 6 – Boxplot avant traitement des données aberrantes



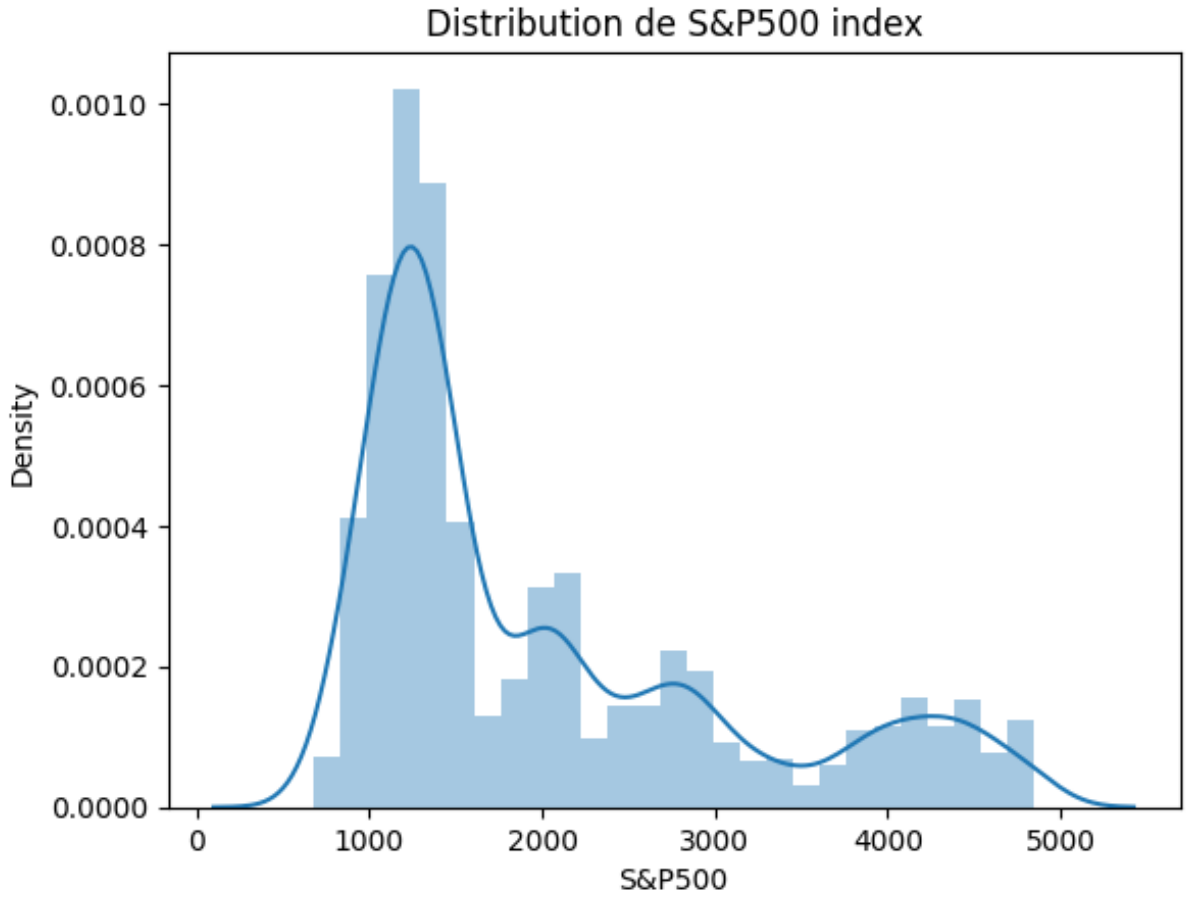
Le boxplot de l'indice S&P 500 nous donne une vue d'ensemble de la distribution des prix de clôture, des valeurs centrales et des valeurs aberrantes. Les points situés en dehors des moustaches sont des valeurs aberrantes. Elles peuvent indiquer des événements exceptionnels ou des erreurs de mesure. Pour résoudre ce problème, nous avons utilisé les méthodes d'imputations par interpolation linéaire.

FIGURE 7 – Boxplot après traitement des données aberrantes



4.2.3 La distribution de notre série

FIGURE 8 – Vérification de la normalité de la série



Ce graphique nous montre que notre série n'est pas normalement distribuée, alors qu'il est important de normalisé les données d'entrées afin d'assurer la convergence de l'algorithme.

4.3 Choix des modèles ARMA, RNN, LSTM et GRU

Le choix des modèles ARMA, RNN, LSTM et GRU permet de comparer les approches classiques et modernes pour la prédiction de l'indice boursier S&P 500. Tandis que les modèles ARMA offrent une base solide pour les relations linéaires, les RNN, LSTM et GRU apportent des solutions avancées pour capturer les dynamiques non linéaires et les dépendances à long terme. Cette comparaison est essentielle pour déterminer la méthode la plus efficace et la plus précise pour la prédiction des séries temporelles financières.

5 Approche classique des séries temporelles ARMA

Dans cette section, nous nous inspirons de l'approche de Box-Jonkins (1976). Cette d'approche suit quelques étapes : test de stationnarité, identification des ordres p et q, estimation du modèle ARMA(p,q) et enfin faire des prévisions.

5.1 Test de stationarité ADF

TABLEAU 3 – Résultats du test de Dickey-Fuller augmenté (ADF) pour le S&P500

	A niveau	**Différence première**
ADF Test Statistic	1.575044	-15.53379
p-value	0.997781	2.203307e-28
Conclusion	Acception de l'hypothèse nulle. Série non stationnaire.	Rejet de l'hypothèse nulle. Série est stationnaire.

Les données du S&P500, à niveau, montrent des tendances ou des variations qui ne sont pas constantes au fil du temps. Il y a une présence de racine unitaire et du coup la série n'est pas stationnaire. Mais elles deviennent stationnaires après avoir pris la différence première.

5.2 Résultats de prévisions du modèle ARMA

Étant donné que notre objectif est de faire des prévisions, nous n'allons pas présenté les estimations du modèle (voir détail sur les estimations du modèle). Nous rappelons qu'on a divisé l'échantillon en données d'entrainement (train data) 90% et en données de test (test data) 10%.

FIGURE 9 – Division de l'échantillon en train/test

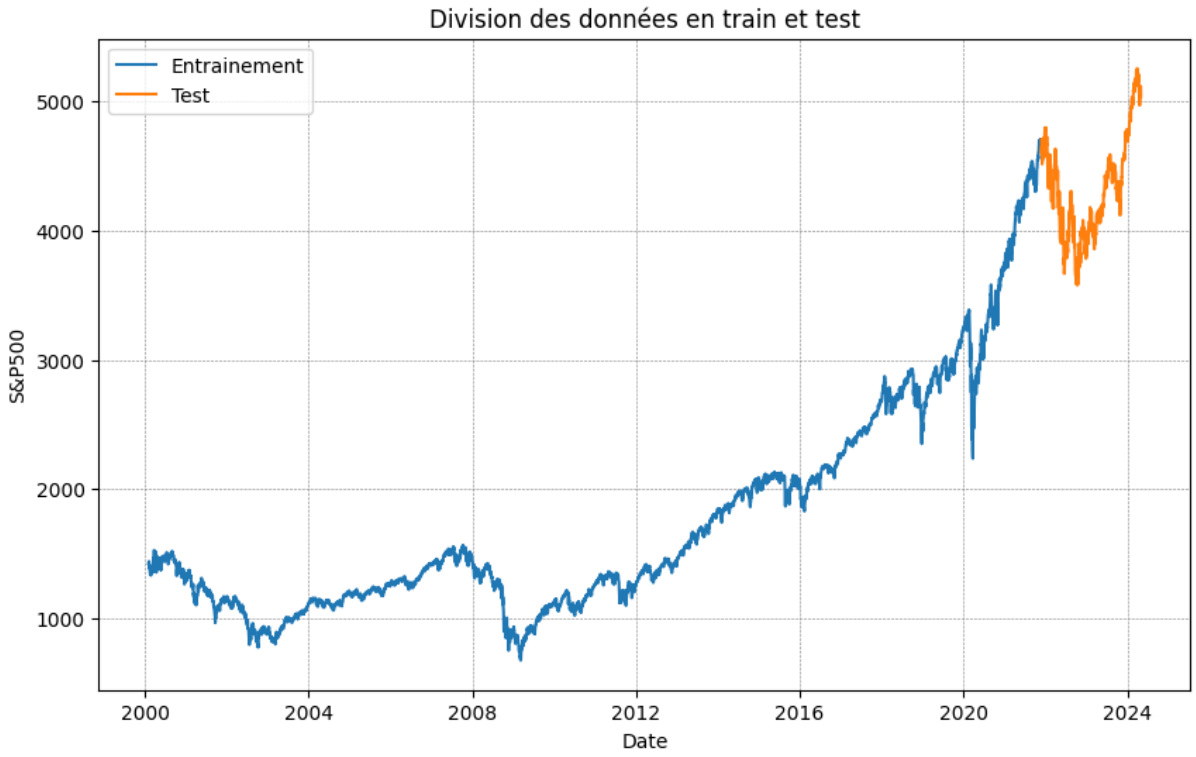
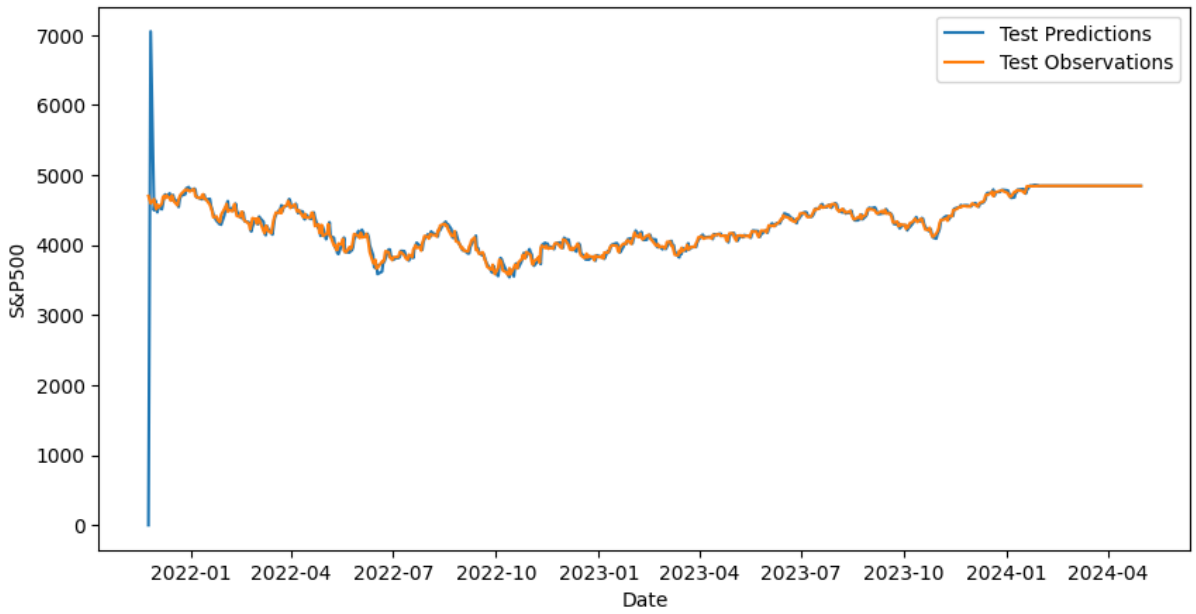


FIGURE 10 – Prévision du modèle ARMA sur les données test



6 Machine learning

6.1 Introduction et l'utilisation de machine learning en économie

Le machine learning (apprentissage automatique) est une sous-discipline de l'intelligence artificielle qui permet aux ordinateurs d'apprendre à partir de données et de faire des prédictions ou des décisions sans être explicitement programmés pour chaque tâche. En utilisant des algorithmes et des modèles statistiques, le machine learning peut identifier des motifs cachés dans de grandes quantités de données et adapter ses prédictions en fonction de nouvelles informations. Cette capacité d'apprentissage autonome a révolutionné de nombreux domaines, y compris l'économie.

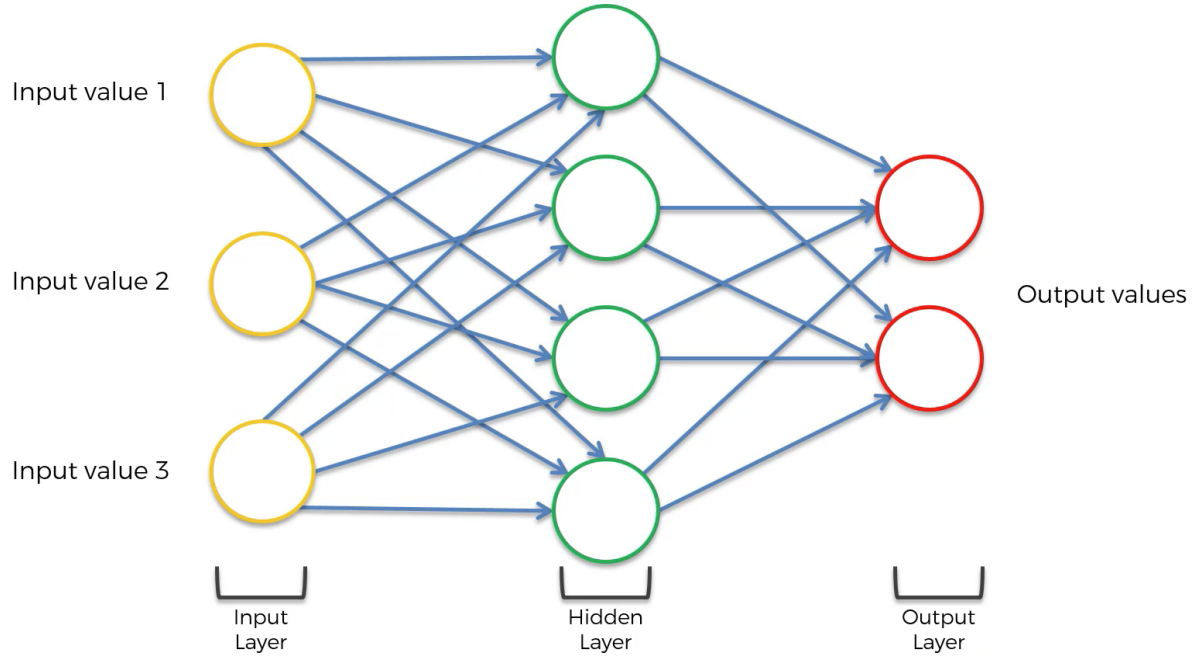
L'application du machine learning en économie a ouvert de nouvelles perspectives pour l'analyse des données, la prédiction et la prise de décision. Les économistes et les chercheurs utilisent ces techniques pour aborder des problèmes complexes et pour améliorer l'efficacité et la précision des modèles économiques traditionnels.

Le machine learning est devenu un outil essentiel dans le domaine de l'économie, offrant des moyens avancés pour analyser des données complexes, faire des prévisions précises et prendre des décisions éclairées. En intégrant ces techniques, les économistes peuvent mieux comprendre les dynamiques économiques, optimiser les politiques et les stratégies, et répondre aux défis actuels avec une efficacité accrue. La capacité du machine learning à apprendre et à s'adapter continuellement à de nouvelles informations en fait une technologie incontournable pour l'avenir de l'analyse économique et de la prise de décision.

6.2 Comment fonctionne le machine learning

6.2.1 Architecture d'un RNN

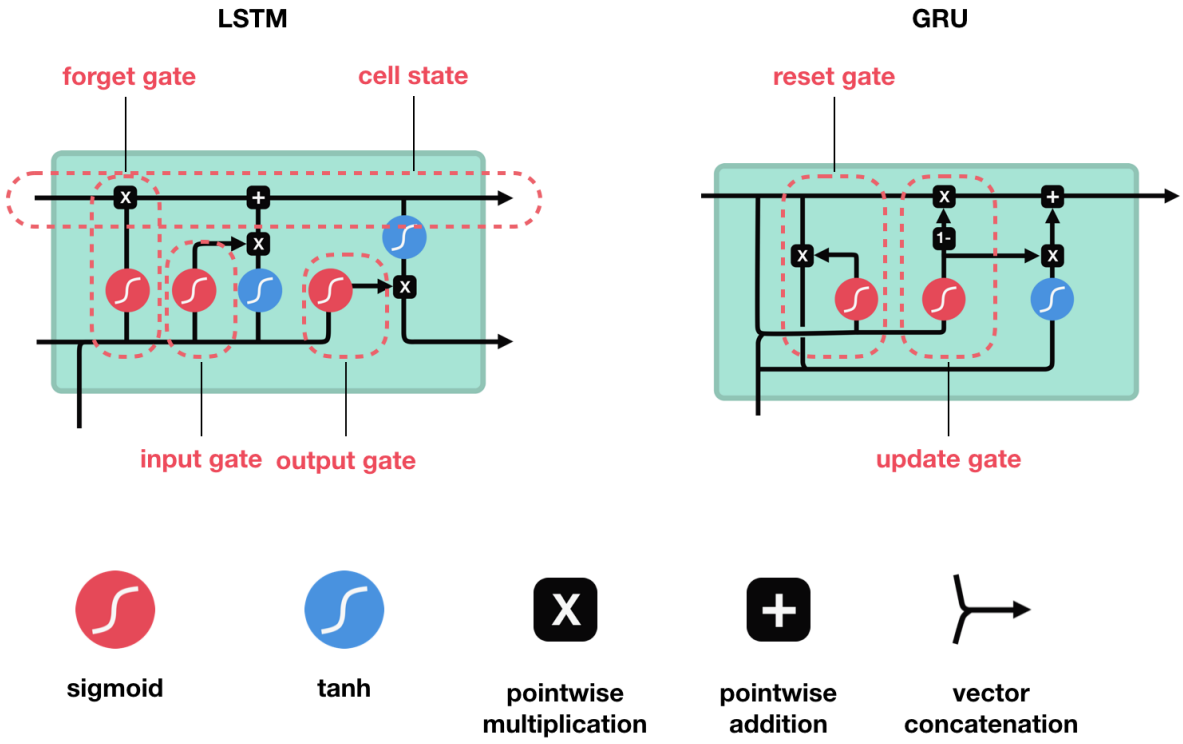
FIGURE 11 – Réseau neurone récurrent



Les RNN sont constitués de neurones ; des noeuds de traitement de données qui travaillent ensemble pour effectuer des tâches complexes. Les neurones sont organisés en couches d'entrée (input layer), couches cachée (hidden layer) et couches de sortie (Output layer). La couche d'entrée reçoit les informations à traiter (input value ou les variables explicatives) et la couche de sortie fournit le résultat. Le traitement et l'analyse et la prévision des données ont lieu dans la couche cachée.

6.2.2 Architecture d'un LSTM et GRU

FIGURE 12 – Réseaux neurones LSTM et GRU



Un réseau LSTM est composé de cellules LSTM, chacune ayant trois portes principales : la porte d'entrée (input gate), la porte d'oubli (forget gate), et la porte de sortie (output gate). Voici les composants essentiels :

- **Porte d'entrée (Input Gate)** : Contrôle combien d'informations de l'entrée actuelle et de l'état caché précédent doivent être conservées dans la cellule.

- **Porte d'oubli (Forget Gate)** : Décide quelles informations de l'état précédent doivent être oubliées.

- **Porte de sortie (Output Gate)** : Détermine quelle partie de l'état de la cellule doit être utilisée comme sortie.

Le GRU est une variante simplifiée de LSTM avec moins de portes, ce qui le rend moins complexe et souvent plus rapide à entraîner. Il possède deux portes principales : la porte de mise à jour (update gate) et la porte de réinitialisation (reset gate).

- **Porte de mise à jour (Update Gate)** : Détermine dans quelle mesure l'état précédent

doit être conservé et combien du nouvel état doit être ajouté.

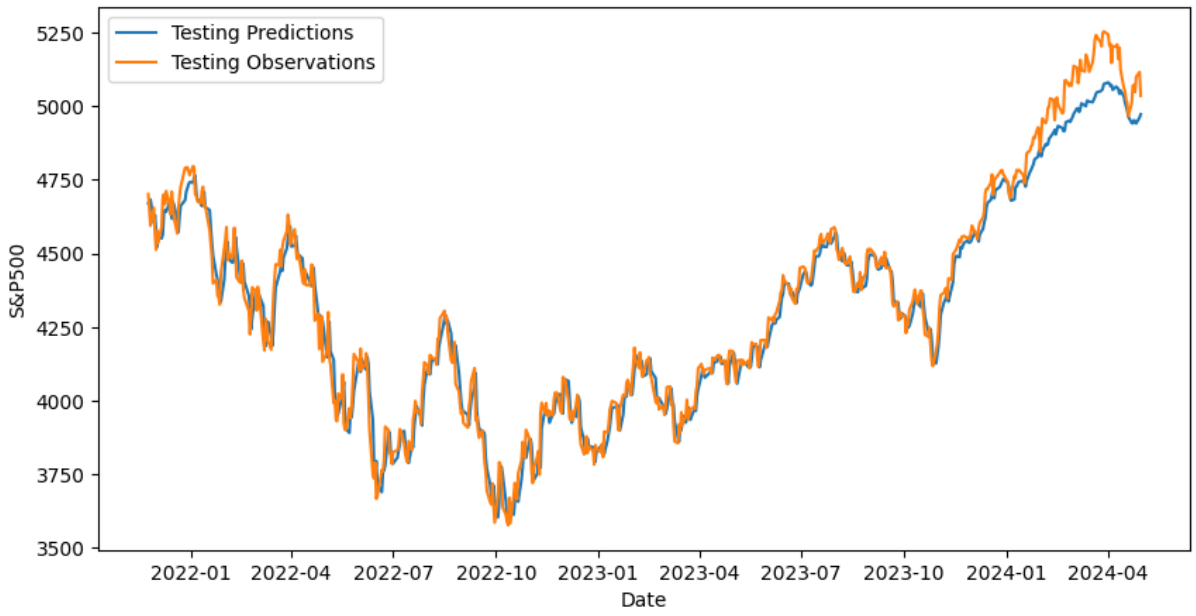
- **Porte de réinitialisation (Reset Gate)** : Contrôle combien de l'état précédent doit être oublié.

7 Application sur la série de l'indice S & P500

Dans cette section, nous présentons les résultats des différents modèles de réseaux neurones.

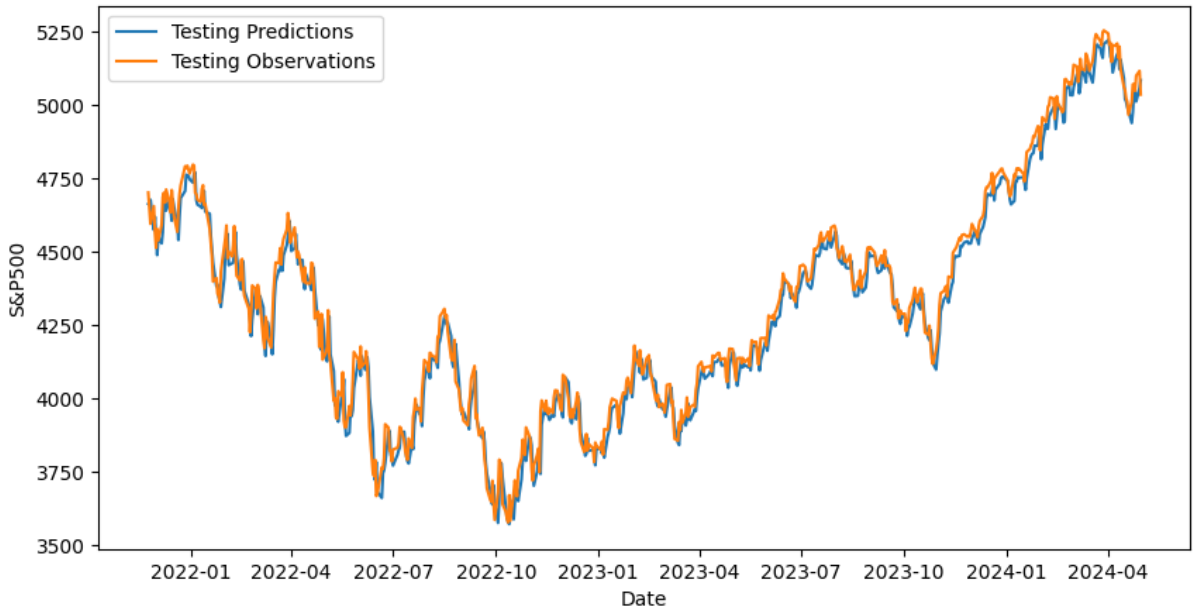
7.1 Modèle RNN

FIGURE 13 – Prévision du modèle RNN sur les données de test



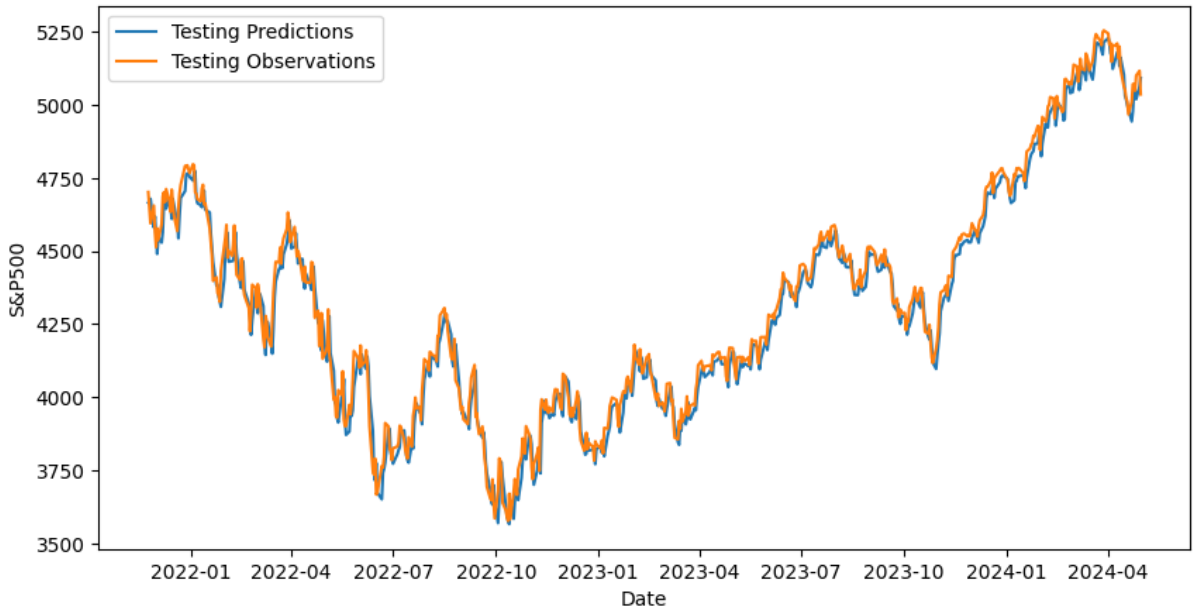
7.2 Modèle LSTM

FIGURE 14 – Prévision du modèle LSTM sur les données de test



7.3 Modèle GRU

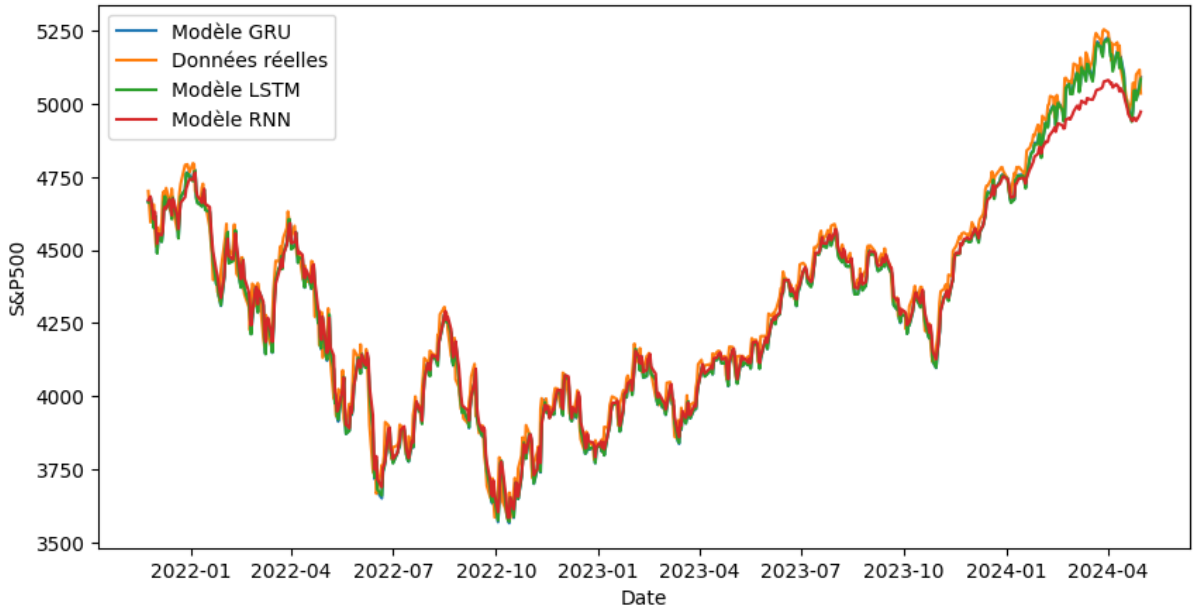
FIGURE 15 – Prévision du modèle GRU sur les données de test



8 Comparaison des modèles

8.1 Comparaison des modèles LSTM, GRU et RNN

FIGURE 16 – Comparaison des modèles de réseaux neurones sur notre échantillons de test



Tous les modèles LSTM, GRU et RNN semblent suivre une tendance similaire, mais avec des variations. L'objectif est d'obtenir des prédictions aussi proches que possible des données réelles. Ce graphique nous montre que les modèles LSTM et GRU donnent des résultats similaires et sont plus performants que le modèle RNN (réseau neurone récurent). Ce résultat est conforme avec les études précédentes sur les réseaux neurones, car LSTM et GRU sont des versions améliorés du RNN qui souvre d'un problème de vanishing gradiant que nous avons évoqué ci-haut.

8.2 Comparaison des modèles de réseaux neurones et la méthode classique (ARMA)

Dans cette partie, nous comparons les métriques que nous avons calculé pour chacun des modèles (ARMA, LSTM, GRU et RNN) sur l'échantillon d'entrainement et celui de test.

TABLEAU 4 – Comparaison des métriques de performance pour les modèles ARMA, LSTM, RNN et GRU.

Modèles	**Métriques**	**Entraînement**	**Test**
ARMA	Erreur absolue moyenne	14.889	48.494
	Erreur quadratique moyenne	1027.436	48775.117
	Erreur quadratique moyenne racine	32.053	220.850
	Coefficient de détermination (R2)	0.998	0.586
LSTM	Erreur absolue moyenne	0.0035	0.0106
	Erreur quadratique moyenne	3.0593e-05	0.0001
	Erreur quadratique moyenne racine	0.0055	0.0134
	Coefficient de détermination (R2)	0.9992	0.9809
RNN	Erreur absolue moyenne	0.0035	0.0125
	Erreur quadratique moyenne	2.9312e-05	0.0001
	Erreur quadratique moyenne racine	0.0054	0.0160
	Coefficient de détermination (R2)	0.9993	0.9728
GRU	Erreur absolue moyenne	0.0039	0.010
	Erreur quadratique moyenne	3.3285e-05	0.0002
	Erreur quadratique moyenne racine	0.0057	0.0133
	Coefficient de détermination (R2)	0.9992	0.9812

En comparant les métriques de performance des différents modèles sur les échantillons d'entraînement et de test, plusieurs points se dégagent

Les modèles de réseaux neuronaux (LSTM, RNN, GRU) surpassent largement le modèle ARMA en termes de MAE, MSE et RMSE, et ont des coefficients de détermination (R^2) proches de 1, indiquant une très bonne capacité de modélisation des données d'entraînement. Le modèle ARMA, bien que performant sur l'échantillon d'entraînement (R^2 de 0.998), montre des erreurs beaucoup plus élevées en comparaison avec les modèles de

réseaux neuronaux.

Les modèles de réseaux neuronaux continuent de surpasser le modèle ARMA sur l'échantillon de test. Leurs erreurs (MAE, MSE, RMSE) restent relativement faibles, et leurs coefficients de détermination (R^2) restent élevés (au-dessus de 0.97), ce qui montre une bonne généralisation. Le modèle ARMA, en revanche, présente une dégradation significative de la performance sur l'échantillon de test avec un R^2 de 0.586 et des erreurs nettement plus élevées, indiquant qu'il généralise moins bien que les modèles de réseaux neuronaux.

Parmi les modèles de réseaux neuronaux, les performances sont très proches, mais le GRU et LSTM semblent légèrement meilleur sur l'échantillon de test avec un R^2 de 0.9812 et des erreurs légèrement plus faibles que celle du RNN. Le RNN, bien que performant, a des erreurs légèrement plus élevées que les LSTM et GRU sur l'échantillon de test.

En conclusion, les modèles de réseaux neuronaux (LSTM, RNN, GRU) offrent une performance supérieure à celle du modèle ARMA pour les deux échantillons (entrainement et test). Le GRU et le LSTM montrent une légère supériorité parmi les réseaux neuronaux, ce qui pourrait en faire les modèles privilégiés pour ce type de données de séries temporelles.

9 Conclusion

Ce projet a permis de comparer efficacement plusieurs modèles de prédiction, notamment ARMA, RNN, LSTM et GRU, pour anticiper les variations de l'indice boursier S&P 500. Les résultats obtenus révèlent que les modèles de réseaux neurones, en particulier LSTM et GRU, surpassent significativement le modèle ARMA en termes de précision et de capacité de généralisation. Cette constatation confirme l'efficacité des techniques de machine learning pour la prédiction des séries temporelles financières.

Cette étude a représenté une expérience enrichissante du point de vue personnel. En travaillant sur ce projet, nous avons pu approfondir nos connaissances en matière de modélisation des séries temporelles et d'application des techniques de machine learning en finance. De plus, nous avons acquis une compréhension plus approfondie des avantages et des limites de différents modèles de prédiction, ce qui a renforcé notre capacité à analyser et à interpréter les données financières.

Pour aller plus loin dans ce projet, plusieurs perspectives peuvent être envisagées.

Nous enrichirons cette étude en incorporant des données externes pertinentes, telles que les données sur l'incertitude de politique climatique, pour capturer des influences externes sur les marchés financiers.

Nous développerons une version opérationnelle des modèles capable de fournir des prédictions en temps réel, ce qui serait utile pour les investisseurs et les analystes financiers.

Références

[1] Afouda, J. (2020). Machine learning par la pratique avec python

[2] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166.

[3] Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation : Encoder-Decoder Approaches. arXiv preprint arXiv :1409.1259.

[4] Reinhart, C. M., & Rogoff, K. S. (2009). This Time is Different : Eight Centuries of Financial Folly. Princeton : Princeton University Press.

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

[6] Box, G.E.P., & Jenkins, G.M. (1976). Time Series Analysis : Forecasting and Control. Holden-Day.

[7] https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/

[8] https://ichi.pro/fr/synthese-complete-des-principaux-avantages-et-inconvenients-des-fonctions-d-activa

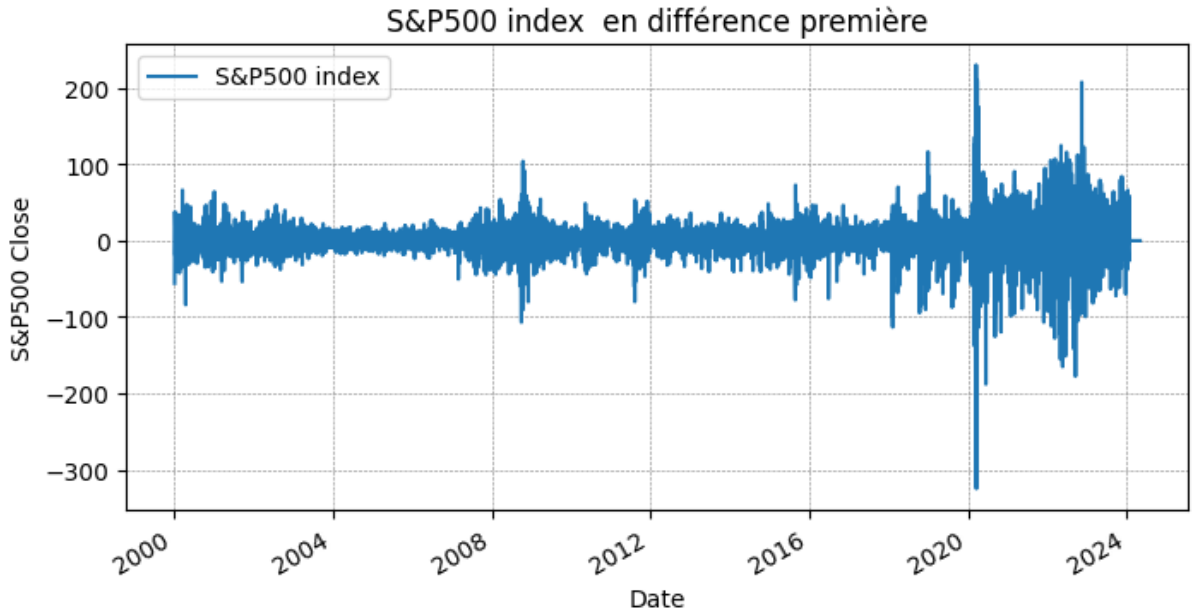
[9] https://www.freecodecamp.org/news/the-ultimate-guide-to-recurrent-neural-networks-in-python/

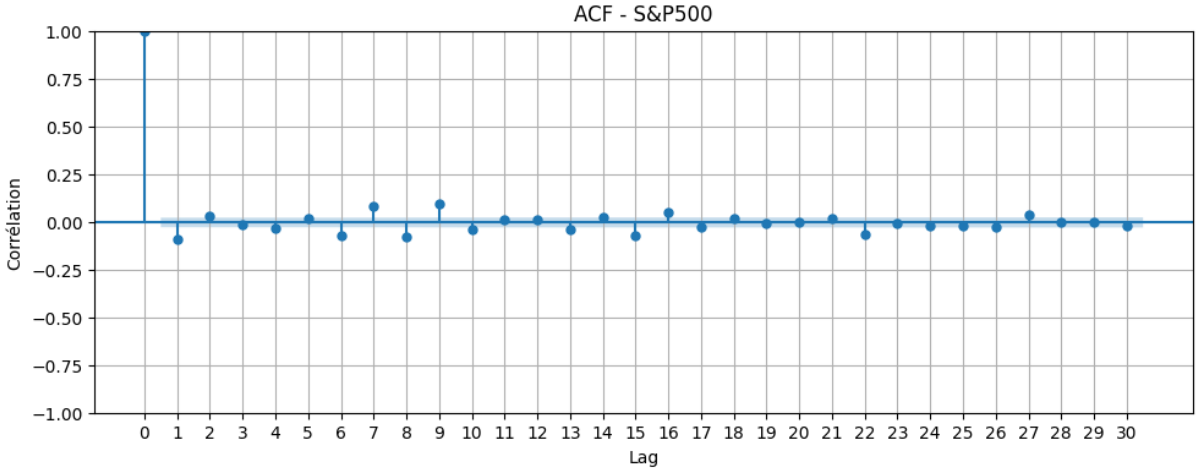
[10] https://www.kaggle.com/code/kanncaa1/recurrent-neural-network-with-pytorch

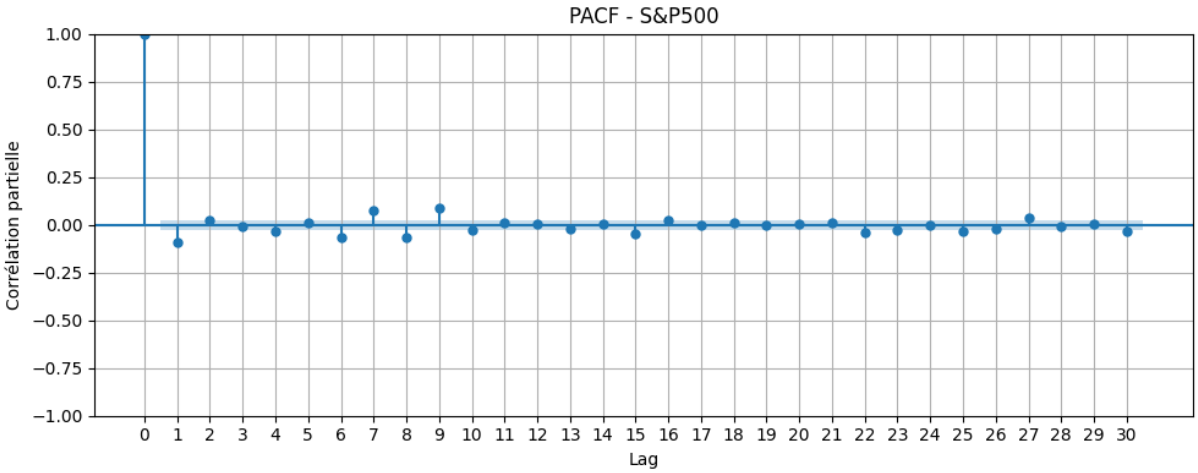
Code Python

Liens vers mon github pour la reproductibilité du travail :
Code python 1
Code python 2

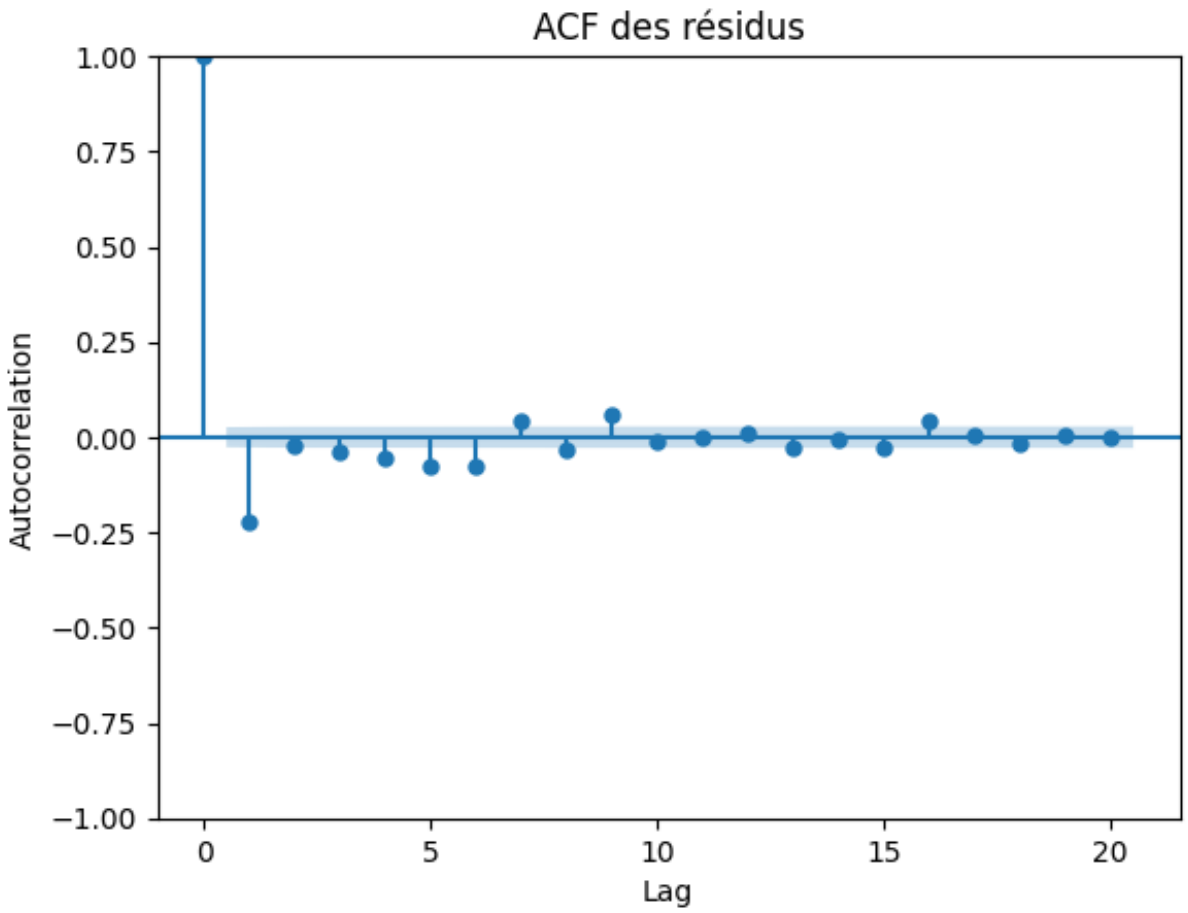
Annexes

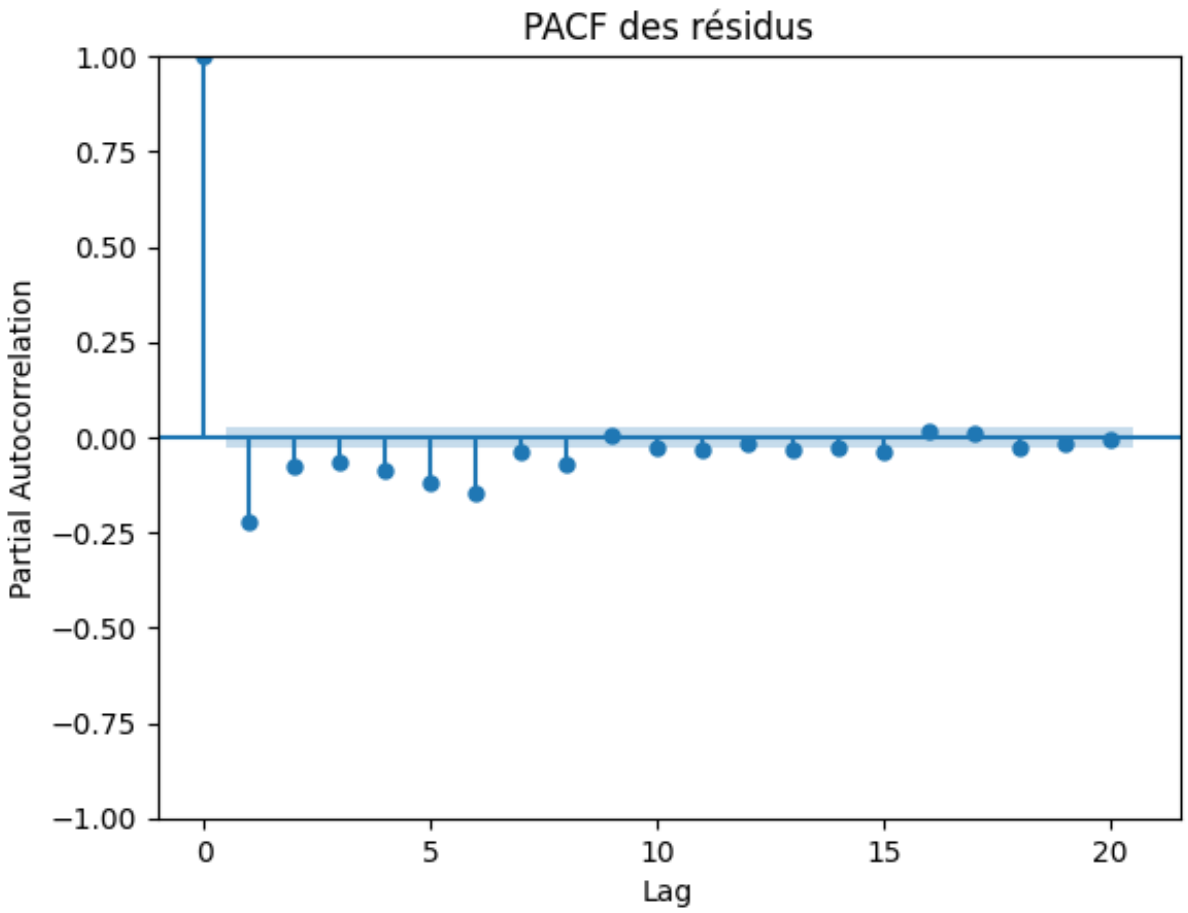


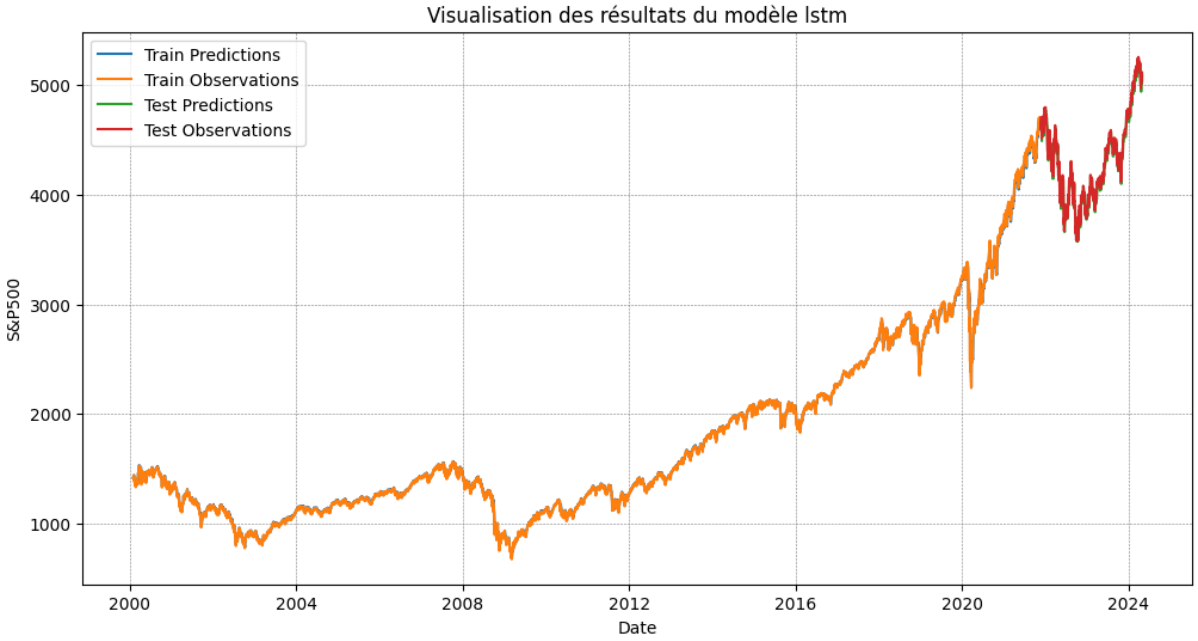


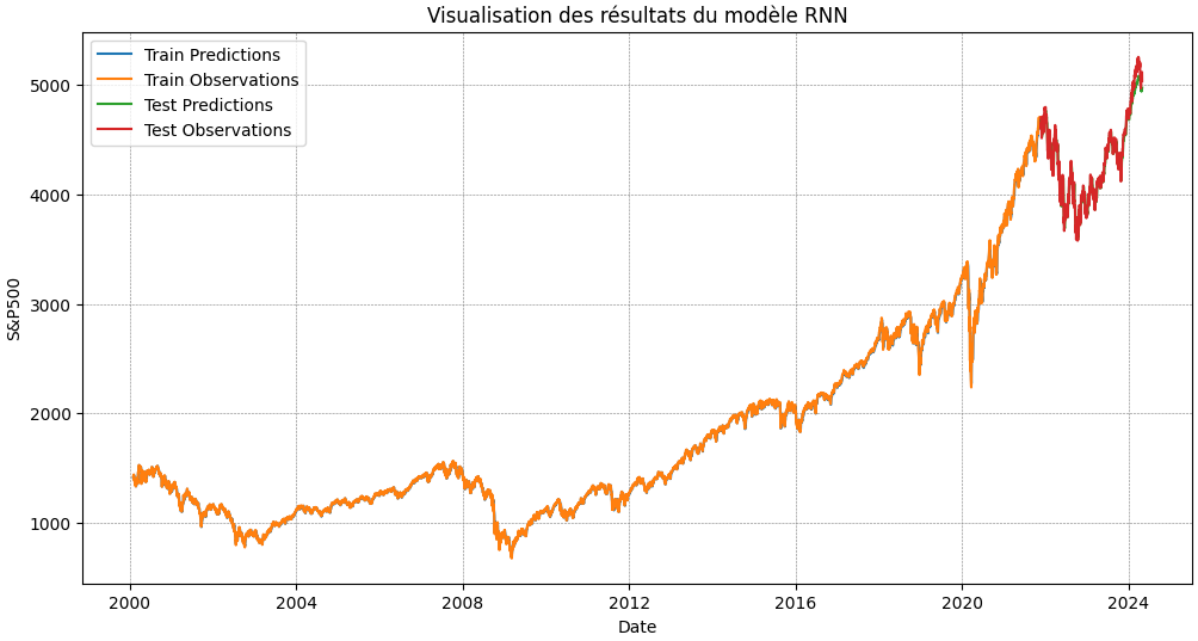


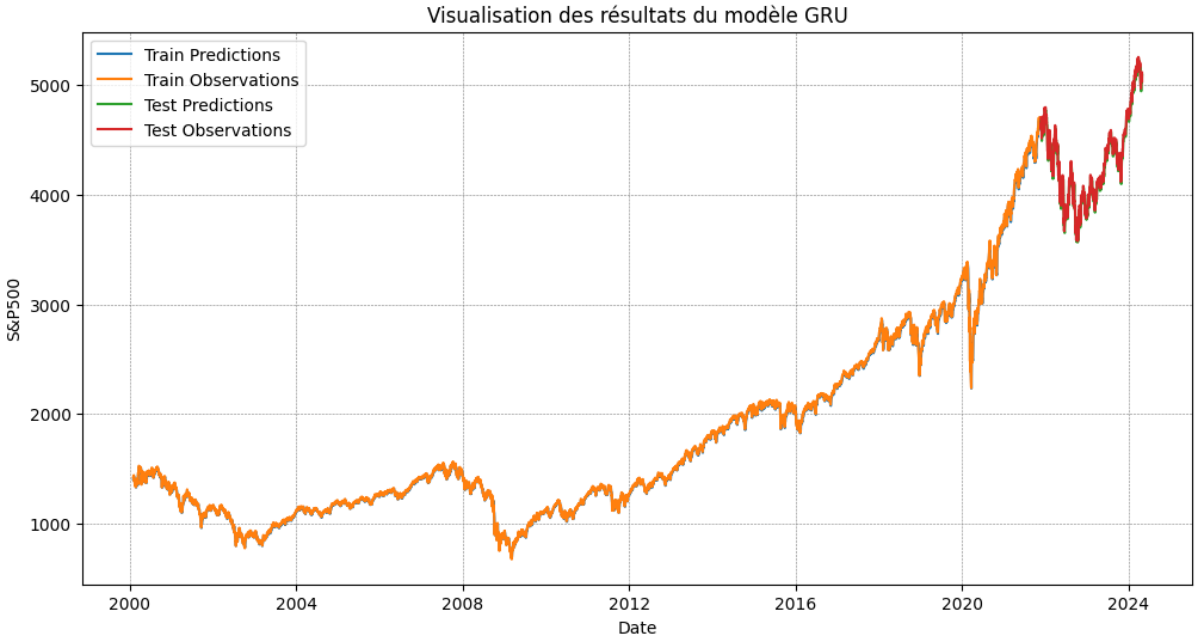
Série différenciée


ACF des résidus
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Autocorrelation
0
5
10
15
20
Lag


PACF des résidus
Partial Autocorrelation
Lag
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


Visualisation des résultats du modèle lstm
Train Predictions
Train Observations
Test Predictions
Test Observations
S&P500
Date
5000
4000
3000
2000
1000
2000
2004
2008
2012
2016
2020
2024


Visualisation des résultats du modèle RNN
Train Predictions
Train Observations
Test Predictions
Test Observations
S&P500
Date
1000
2000
3000
4000
5000
2000
2004
2008
2012
2016
2020
2024


Visualisation des résultats du modèle GRU
Train Predictions
Train Observations
Test Predictions
Test Observations
S&P500
Date
1000
2000
3000
4000
5000
2000
2004
2008
2012
2016
2020
2024