Construction d'un modèle de prédiction de cours de l'action NTT

Kalalou Ilias

22 mars 2025

Table des matières

1 Introduction

Ce rapport décrit le projet de construction d'un modèle de prédiction de cours de l'action NTT dans le cadre d'un stage. L'objectif est de développer une solution d'analyse et de prévision des séries temporelles en utilisant des méthodes de machine learning. Le projet s'est déroulé en plusieurs phases :

— Compréhension des données et Analyse Exploratoire (EDA)
— Prétraitement des données et Feature Engineering
— Sélection et entraînement des modèles de prédiction
— Évaluation des performances et analyse des résultats
— Amélioration et itérations sur les modèles

2 Analyse Exploratoire des Données (EDA)

2.1 Présentation des données

Les données utilisées dans ce projet proviennent des cours de l'action NTT. Chaque enregistrement comporte les informations suivantes :

— (Date)
— (Cours de clôture)
— (Cours d'ouverture)
— (Cours le plus haut)
— (Cours le plus bas)
— (Volume)
— % (Pourcentage de variation)

Un premier aperçu a été obtenu en affichant les cinq premières lignes ainsi que la dimension de la matrice de données. Par la suite, plusieurs tableaux ont été générés afin d'examiner en détail la structure des données et d'en tirer des conclusions sur leur qualité. Par exemple, nous avons affiché un graphique du cours de clôture, ce qui nous a permis d'observer la tendance globale de l'action.

2.2 Visualisations et Analyse Descriptive

Afin de mieux comprendre le comportement du cours de l'action, plusieurs graphiques ont été réalisés :

— **Trend in NTT closing Price :** Ce graphique (Figure 1) présente la courbe des prix de cloture depuise le debut du lancement de l'action NTT en bourse. On peux ainsi voir que NTT a fait unne entré trés réussit en bourse mais a ensuite subit une grosse baisse pour ensuite se stabiliser et croitre lentement mais surement.
— **Distribution du Cours de Clôture :** Ce graphique (Figure 2) présente l'histogramme et la courbe de densité des valeurs de clôture. Il révèle que les valeurs les plus fréquentes se situent entre 30 et 60 dollars, donnant ainsi une estimation de la valeur moyenne de l'action.
— **Variation du Cours de Clôture par Mois :** Le graphique présenté dans la Figure 3 montre les variations mensuelles des cours de clôture. On observe une forte volatilité durant les mois de mars à juin ainsi qu'en octobre. Cette information suggère qu'une attention particulière pourrait être portée à ces périodes lors de la modélisation.

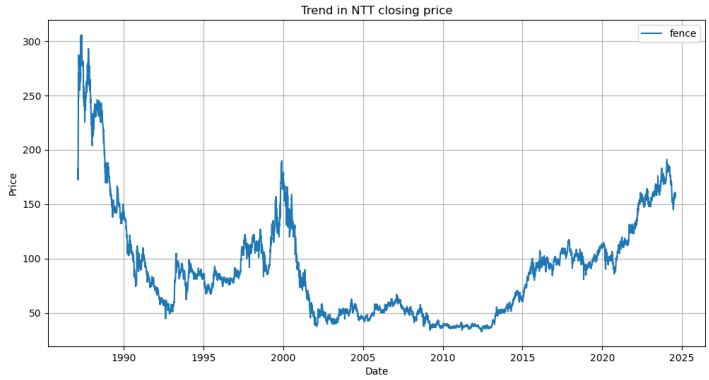

FIGURE 1 – Trend in NTT closing Price

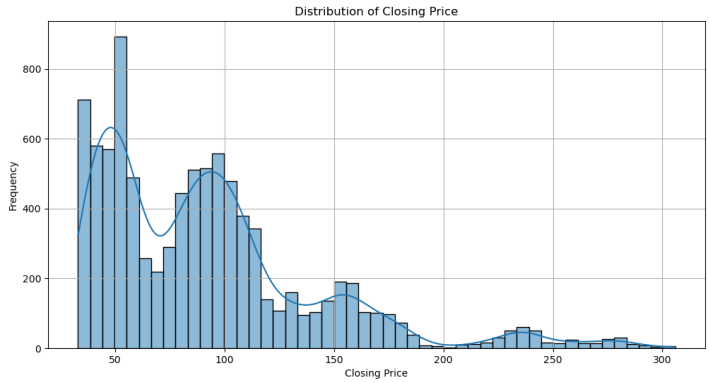

FIGURE 2 – Distribution du Cours de Clôture

FIGURE 3 – Variation du Cours de Clôture par Mois

Ces visualisations, combinées aux statistiques descriptives (moyenne, médiane, écart-type, etc.), ont permis d'identifier des tendances globales et des anomalies potentielles dans les données, formant ainsi la base pour le choix des modèles de prédiction.

3 Prétraitement des Données et Feature Engineering

3.1 Nettoyage et Réindexation

Le prétraitement des données a été réalisé en plusieurs étapes :

— **Conversion des Dates :** La colonne a été convertie en format `datetime` pour faciliter les opérations temporelles.

— **Réindexation :** Le DataFrame a été réindexé pour inclure l'ensemble des dates ouvrées (business days) comprises entre la date minimale et la date maximale. Les valeurs manquantes (du fait des weekends et des jours fériés) ont ensuite été interpolées à l'aide d'une méthode linéaire, permettant d'obtenir une série temporelle continue.

3.2 Normalisation

Afin de garantir que toutes les variables soient sur une échelle comparable, les colonnes numériques suivantes ont été normalisées à l'aide du `MinMaxScaler` :

— (Cours de clôture)

— (Cours d'ouverture)

— (Cours le plus haut)

— (Cours le plus bas)

— (Volume)

La colonne a été préalablement convertie (par exemple, en transformant «79.15M» en un nombre absolu) à l'aide d'une fonction dédiée. Par ailleurs, afin de stabiliser la variance et de faciliter la modélisation, une transformation logarithmique a été appliquée à la cible :

$$\texttt{log()} = \ln\Big(+ 10^{-6}\Big)$$

3.3 Création des Séquences Temporelles

Pour exploiter les modèles de réseaux de neurones, la série temporelle a été découpée en séquences. Dans notre approche, des séquences de longueur 20 ont été utilisées pour prédire la valeur suivante de la cible (i.e. `log()`). Cette étape est cruciale pour capturer les dépendances temporelles et fournir au modèle une fenêtre contextuelle suffisante pour effectuer des prédictions précises.

4 Modèles de Prédiction et Entraînement

4.1 Modèle ARIMA

Le modèle ARIMA (AutoRegressive Integrated Moving Average) a été choisi pour plusieurs raisons :

— **Avantages :**

  — *Simplicité et interprétabilité* : ARIMA est un modèle statistique classique qui permet d'identifier les composantes auto-régressives (AR) et de moyenne mobile (MA) ainsi que la nécessité d'intégration (I) pour rendre la série stationnaire.

  — *Efficacité pour la prévision one-step-ahead* : En se basant sur les valeurs passées et sur les erreurs antérieures, ARIMA peut fournir des prévisions robustes pour le prochain pas de temps.

— **Inconvénients :**
  — *Hypothèse de linéarité* : ARIMA est un modèle linéaire et peut être limité dans sa capacité à capturer des dynamiques non linéaires présentes dans les données financières.
  — *Détérioration des prévisions multi-steps* : Pour des horizons de prévision multiples, la méthode de *rolling forecast* utilisée par ARIMA tend à accumuler les erreurs, conduisant souvent à des prévisions qui convergent vers une moyenne.

Ces caractéristiques en font un modèle pertinent pour établir une baseline et analyser la dynamique linéaire des séries temporelles, tout en servant de référence pour comparer d'autres approches plus avancées.

4.1.1 Implémentation du modèle ARIMA

Pour implémenter le modèle ARIMA, les étapes suivantes ont été réalisées :

1. **Transformation de la cible :** La colonne (cours de clôture) a été transformée par une fonction logarithmique pour stabiliser la variance :

$$\texttt{\_log} = \ln\Big(+ 10^{-6}\Big)$$

   Ceci permet de mieux gérer les variations importantes et de rendre la série plus stationnaire.

2. **Test de stationnarité :** Un test d'Augmented Dickey-Fuller (ADF) a été réalisé sur la série log-transformée afin de vérifier la stationnarité. Les résultats obtenus (0.34) ont confirmé la nécessité d'appliquer une différenciation (d=1).

3. **Séparation Train/Test :** La série log-transformée a été divisée en deux ensembles, avec environ 95% des données utilisées pour l'entraînement et 5% pour le test. Cette répartition permet d'entraîner le modèle sur un grand volume de données historiques et de tester sa capacité à prévoir de nouveaux points.

4. **Rolling Forecast :** La méthode de *rolling forecast* a été adoptée pour simuler un scénario de prévision en temps réel. Le modèle ARIMA (de type ARIMA(0,1,1)) est réentraîné à chaque itération sur l'historique disponible pour effectuer une prévision one-step-ahead. À chaque étape, la vraie valeur observée est ajoutée à l'historique, permettant ainsi une mise à jour continue du modèle.

$$Prédiction y_{t+1} = ARIMA(y_1, y_2, \ldots, y_t)$$

   Les prévisions sont ensuite inversées (via l'exponentielle) pour revenir à l'échelle originale.

4.1.2 Test de fiabilité et évaluation du modèle ARIMA

Pour évaluer la fiabilité du modèle ARIMA, plusieurs métriques ont été calculées :
— **RMSE (Root Mean Squared Error)** et **MAE (Mean Absolute Error)** : Ces métriques mesurent l'écart global entre les prévisions et les valeurs réelles.
— **Évaluation personnalisée de la tendance et de la proximité** : Un algorithme a été développé pour mesurer la capacité du modèle à prédire la direction (hausse ou baisse) du cours et la proximité des prévisions par rapport aux valeurs réelles. Pour chaque jour (à partir du deuxième), la variation journalière réelle et prédites est comparée :

— *Précision de la tendance* : Pourcentage de fois où le signe de la variation (hausse ou baisse) est correctement prédit.
— *Précision de proximité* : Un score est attribué pour chaque jour en fonction de l'écart absolu entre la prévision et la valeur réelle, avec un score maximal (100%) pour une différence nulle et un score nul pour une différence supérieure ou égale à 1 $.

Le score global est ensuite défini comme la moyenne des deux précisions.

4.1.3 Résultats obtenus

Les résultats du modèle ARIMA sont les suivants :
— **RMSE :** 1.71
— **MAE :** 1.32

De plus, l'évaluation personnalisée a permis d'obtenir :
— **Précision de la tendance :** 63.91% si on le teste sur 1% des données mais baisse a 55.1 % sur 5% des donné se qui est plus representatife

Ces résultats indiquent que, bien que le modèle ARIMA soit performant pour prédire la prochaine valeur, sa capacité à prédire sur des horizons plus longs reste limitée en raison de l'accumulation d'erreurs.

4.1.4 Visualisation des résultats

La Figure 4 ci-dessous illustre la courbe de prévision obtenue par la méthode de rolling forecast du modèle ARIMA, comparée aux valeurs réelles. On peut constater que la courbe prévisionnelle suit globalement la tendance générale de l'action, bien que certains écarts subsistent. Ces écarts sont principalement dus à la nature linéaire du modèle ARIMA et à l'accumulation progressive des erreurs lors des prévisions multiples.

Après plusieurs tests et expérimentations, nous avons constaté que le modèle ARIMA offrait des performances nettement meilleures lorsque les données étaient complètes – c'est-à-dire après avoir complété les valeurs manquantes par interpolation – et lorsqu'elles n'étaient pas normalisées. En effet, la normalisation tendait à modifier légèrement l'échelle des valeurs, ce qui impactait la précision des prévisions, tandis que l'utilisation de données complètes permettait de mieux capturer les dynamiques intrinsèques du cours de l'action. Nous avons aussi essayer de supprimer les valeur

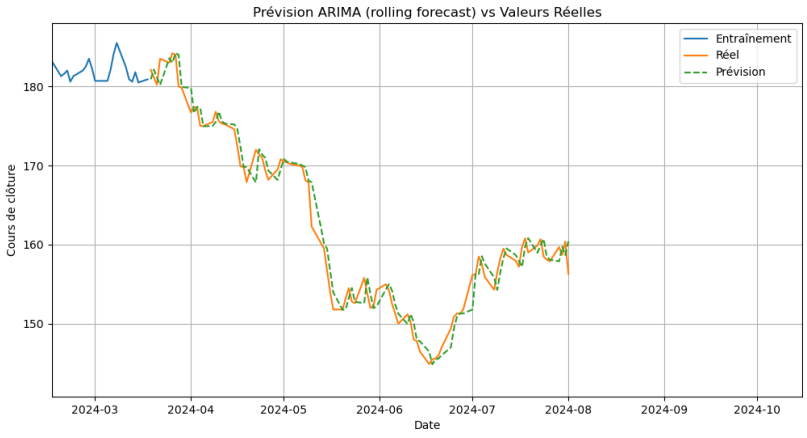

FIGURE 4 – Prévision ARIMA (rolling forecast) comparée aux valeurs réelles

4.2 Modèles LSTM

Un modèle LSTM univarié a été développé pour servir de référence et comparer les performances avec le modèle ARIMA. Le choix d'un LSTM (Long Short-Term Memory) repose sur sa capacité à capturer les dépendances à long terme dans les séries temporelles, en surmontant notamment le problème de l'explosion ou de la disparition du gradient qui affecte les réseaux de neurones récurrents standards.

Avantages :

- *Gestion des dépendances à long terme* : La structure en mémoire interne des LSTM permet de conserver des informations sur de longues périodes, ce qui est particulièrement utile dans les séries financières présentant des dynamiques complexes.
- *Adaptabilité aux non-linéarités* : Contrairement aux modèles linéaires comme ARIMA, les LSTM peuvent modéliser des relations non linéaires entre les variables, ce qui offre un potentiel d'amélioration dans la prévision de séries présentant une forte volatilité.

Inconvénients :

- *Complexité computationnelle* : L'entraînement des LSTM nécessite généralement un temps de calcul plus important et des ressources matérielles accrues, notamment en cas d'hyperparamétrage inadapté.
- *Nécessité de grands volumes de données* : Pour obtenir des performances fiables, les LSTM demandent souvent un nombre important d'observations, ce qui peut poser problème pour des séries temporelles de taille limitée.

4.2.1 Implémentation du modèle LSTM

L'implémentation du modèle LSTM s'est articulée autour des étapes suivantes :

1. **Préparation des données :** La série temporelle correspondant à la colonne (cours de clôture) a été extraite soit en version brute, soit en version normalisée. Ensuite, des séquences temporelles de longueur fixe (par exemple, 10 pas de temps) ont été générées pour constituer l'ensemble des entrées du réseau.
2. **Séparation Train/Test :** La série a été découpée en deux ensembles, par exemple avec 95% des séquences utilisées pour l'entraînement et 5% pour le test. Ce fractionnement permet d'entraîner le modèle sur la majeure partie des données historiques tout en évaluant sa capacité à prédire des valeurs non vues.
3. **Architecture du modèle :** Le modèle LSTM utilisé comporte deux couches récurrentes de 50 neurones chacune, suivies d'une couche entièrement connectée pour produire la prévision. Les états initiaux du LSTM sont initialisés à zéro pour chaque mini-lot lors de l'entraînement.
4. **Entraînement et optimisation :** L'entraînement a été réalisé sur 50 époques avec l'optimiseur Adam et une fonction de coût basée sur l'erreur quadratique moyenne (MSE). Une barre de progression a été affichée afin de suivre l'évolution de la perte durant l'entraînement.
5. **Prédiction et évaluation :** Une fois le modèle entraîné, des prédictions ont été effectuées sur l'ensemble de test. Les métriques RMSE (Root Mean Squared Error) et MAE (Mean Absolute Error) ont été calculées pour quantifier l'écart global entre

les prévisions et les valeurs réelles. Par ailleurs, une évaluation personnalisée a été réalisée pour mesurer la précision de la tendance (capacité à prédire correctement la direction de la variation) et la proximité moyenne, permettant ainsi de calculer un score global de fiabilité.

4.2.2 Test de fiabilité et évaluation du modèle LSTM

L'efficacité du modèle LSTM a été évaluée à la fois sur les données brutes et sur les données normalisées. Pour chaque cas, le modèle a généré des prévisions one-step-ahead, et les erreurs ont été mesurées par RMSE et MAE. En complément, un algorithme d'évaluation a permis de mesurer :

- *La précision de la tendance* : Le pourcentage de jours où le signe de la variation journalière (hausse ou baisse) est correctement prédit.
- *La précision de proximité* : Un score attribué pour chaque jour en fonction de l'écart entre la valeur prédite et la valeur réelle, avec un score de 100% en cas d'égalité et un score dégressif pour des écarts plus importants.

Le score global de fiabilité est la moyenne de ces deux précisions.

4.2.3 Résultats obtenus

Les résultats expérimentaux obtenus pour le modèle LSTM sont les suivants :

- **Sur les données non normalisées :**
  - *RMSE :* 159.84
  - *MAE :* 159.49
  - *Précision de la tendance :* 47.74%
- **Sur les données normalisées :**
  - *RMSE :* 0.14
  - *MAE :* 0.14
  - *Précision de la tendance :* 49.59%

Ces résultats indiquent que le modèle LSTM est très sensible aux étapes de prétraitement des données. Les très faibles valeurs d'erreur obtenues sur les données normalisées résultent principalement de la réduction d'échelle due à la normalisation, et non d'une véritable performance prédictive. En revanche, les erreurs élevées sur les données non normalisées montrent que le modèle ne capture pas efficacement la dynamique sous-jacente de la série. Ainsi, bien que le LSTM présente un potentiel pour modéliser des relations non linéaires, ses performances globales dans ce contexte restent limitées. Ces constats suggèrent qu'il est nécessaire d'envisager des améliorations supplémentaires, telles qu'un réglage plus fin des hyperparamètres, l'utilisation d'un jeu de données d'entraînement plus important ou l'exploration d'architectures plus sophistiquées. Le modèle LSTM sert ainsi de référence pour comparer les performances des méthodes plus classiques, telles qu'ARIMA.

4.2.4 Visualisation des résultats

La Figure 5 illustre la comparaison entre les valeurs réelles et les prévisions générées par le modèle LSTM, tant pour les données non normalisées que pour les données normalisées. Bien que le modèle parvienne parfois à suivre la tendance générale, des écarts significatifs sont observés, soulignant les limites de l'approche LSTM pour cette série

temporelle spécifique. Cette visualisation, combinée aux métriques d'erreur et aux scores d'évaluation, permet de mieux comprendre les forces et les faiblesses du modèle.

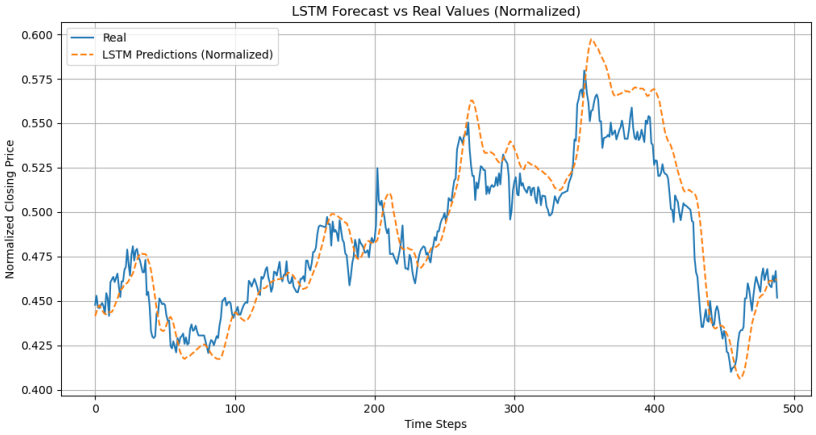

FIGURE 5 – Prévision LSTM comparée aux valeurs réelles

4.3 GRU Multivarié

Ce modèle exploite plusieurs features pour prédire la transformation logarithmique de . Contrairement à un modèle univarié qui ne considère que la série cible, le GRU multivarié intègre diverses informations afin de mieux capturer les interactions et la dynamique des données financières.

4.3.1 Avantages :

— *Exploitation de l'information multivariée* : L'utilisation simultanée de plusieurs features permet au modèle de détecter des interactions complexes et d'améliorer la précision des prévisions.
— *Efficacité en termes de paramètres* : Les GRU requièrent généralement moins de paramètres que les LSTM, facilitant ainsi leur entraînement sur des jeux de données volumineux.
— *Stabilité de l'entraînement* : L'application de techniques telles que le clipping des gradients aide à stabiliser l'entraînement, particulièrement important dans le cas de séries financières très volatiles.

4.3.2 Inconvénients :

— *Dépendance aux hyperparamètres* : La performance du modèle est fortement influencée par le choix des hyperparamètres (longueur des séquences, taux d'apprentissage, ratio train/test, etc.), nécessitant une phase d'expérimentation approfondie.
— *Complexité du prétraitement* : La préparation des données multivariées requiert une normalisation rigoureuse pour éviter que des échelles disparates ne biaisent l'apprentissage.

4.3.3 Implémentation et Entraînement :

L'implémentation du modèle GRU multivarié s'est déroulée en plusieurs étapes :

1. **Préparation des données :** La colonne a été extraite (soit en version brute, soit normalisée) et les autres features ont été intégrées dans le DataFrame. La série est ensuite découpée en séquences temporelles de longueur fixe (par exemple, 10 pas de temps) grâce à une fonction dédiée.
2. **Séparation Train/Test :** Les séquences ont été divisées en un ensemble d'entraînement et un ensemble de test, par exemple en utilisant un ratio de 95% pour l'entraînement et 5% pour le test. Cette répartition permet d'entraîner le modèle sur une grande partie de l'historique tout en évaluant sa capacité à généraliser.
3. **Architecture du modèle :** Le modèle GRU multivarié est construit avec deux couches GRU ayant chacune 100 neurones cachés, suivies d'une couche entièrement connectée qui génère la prévision. Pour stabiliser l'entraînement, un clipping des gradients est appliqué.
4. **Entraînement :** L'entraînement s'est déroulé sur 50 époques avec un taux d'apprentissage de 0.001. À chaque époque, le modèle est optimisé pour minimiser l'erreur quadratique moyenne (MSE) entre les valeurs prédites et les valeurs réelles.

4.3.4 Évaluation et Résultats :

Les performances du modèle GRU multivarié ont été évaluées à l'aide de métriques classiques telles que le RMSE (Root Mean Squared Error) et le MAE (Mean Absolute Error), ainsi qu'une évaluation personnalisée qui mesure la capacité du modèle à prédire correctement la direction des variations quotidiennes (précision de la tendance) et la proximité entre les prévisions et les valeurs réelles. Par exemple, en utilisant la série normalisée de la colonne , le modèle a donné les résultats suivants :

— **Sur les données non normalisées :**
  — *RMSE :* 155.58
  — *MAE :* 155.22
  — *Précision de la tendance :* 48.77%
— **Sur les données normalisées :**
  — *RMSE :* 0.017
  — *MAE :* 0.014
  — *Précision de la tendance :* 48.15%

Ces résultats montrent que l'intégration de plusieurs variables via le GRU multivarié permet, dans une certaine mesure, de réduire l'erreur de prédiction par rapport aux approches univariées. Toutefois, les très faibles valeurs d'erreur obtenues sur les données normalisées traduisent principalement l'impact de la normalisation sur l'échelle des données, plutôt qu'une performance prédictive réellement satisfaisante. De plus, la précision de la tendance demeure autour de 48–49%, indiquant que le modèle peine à anticiper correctement la direction des variations journalières. Ces constats soulignent que, malgré une certaine stabilité d'entraînement (grâce au clipping des gradients), le modèle GRU multivarié nécessite encore des améliorations (par exemple, une meilleure sélection des features ou un réglage plus fin des hyperparamètres) pour optimiser sa capacité à capturer la dynamique complexe des séries financières.

4.3.5 Visualisation des résultats :

La Figure 6 compare les valeurs réelles aux prévisions générées par le modèle GRU multivarié. On peut observer que, malgré quelques écarts ponctuels, le modèle parvient

globalement à suivre la tendance générale de la série temporelle. Cependant, les résultats révèlent aussi des limites dans la capacité du modèle à prévoir précisément les variations quotidiennes des cours de clôture.

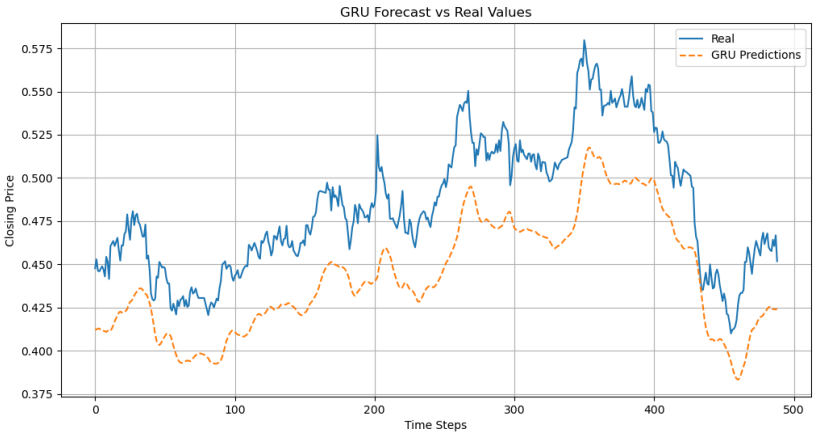

FIGURE 6 – GRU Multivariate Forecast vs. Actual Values

4.4 TCN (Temporal Convolutional Network)

Le modèle TCN utilise des blocs temporels à convolution dilatée pour capturer des dépendances temporelles sur plusieurs échelles. La classe `TCN` empile ces blocs, chacun constitué de deux couches de convolution dilatée, suivies d'opérations de dropout et d'une fonction d'activation ReLU, puis applique une couche linéaire sur la sortie du dernier pas temporel pour générer la prévision.

4.4.1 Avantages :

— *Captation multi-échelle* : Les convolutions dilatées permettent de saisir à la fois des dépendances à court et à long terme, sans recourir à des mécanismes récurrents.
— *Parallélisation* : Les opérations de convolution peuvent être parallélisées, réduisant ainsi le temps d'entraînement par rapport aux modèles récurrents.
— *Stabilité des gradients* : La structure convolutionnelle atténue les problèmes d'explosion ou de disparition des gradients souvent rencontrés dans les architectures récurrentes.

4.4.2 Inconvénients :

— *Dépendance aux hyperparamètres* : La performance du TCN est sensible aux choix de la taille du noyau, du nombre de couches, du facteur de dilatation et du dropout, ce qui nécessite une optimisation minutieuse.
— *Complexité d'interprétation* : Les opérations de convolution et de dilatation rendent parfois l'interprétation des coefficients moins intuitive comparée aux modèles linéaires classiques.

4.4.3 Implémentation et Entraînement :

L'implémentation du modèle TCN s'est réalisée selon les étapes suivantes :

1. **Préparation des données :** La série temporelle correspondant à la colonne a été extraite (soit en version brute, soit normalisée). Des séquences temporelles de longueur fixe (par exemple, 10 pas de temps) ont été générées via une fonction dédiée pour servir d'entrées au modèle.
2. **Architecture du modèle :** Le TCN est construit en empilant plusieurs blocs temporels. Chaque bloc applique deux convolutions dilatées avec un padding calculé pour préserver la longueur de la séquence, suivies d'une activation ReLU et d'un dropout. Une connexion résiduelle est ajoutée pour faciliter la propagation des gradients. La sortie du dernier bloc est passée à une couche linéaire pour obtenir la prévision finale.
3. **Entraînement :** Le modèle a été entraîné sur 50 époques avec un taux d'apprentissage de 0.001. La fonction de coût utilisée est l'erreur quadratique moyenne (MSE). Une barre de progression a été affichée pour suivre l'évolution de la perte, et les pertes ont été rapportées toutes les 10 époques.
4. **Évaluation :** Après l'entraînement, le modèle a été évalué sur un ensemble de test (par exemple, 10% des données). Les métriques RMSE et MAE ont été calculées pour quantifier l'erreur globale entre les prévisions et les valeurs réelles. De plus, une évaluation personnalisée de la tendance (prédiction correcte de la direction des variations) et de la proximité (écart entre la valeur prédite et la valeur réelle) a été réalisée pour obtenir un score global de fiabilité.

4.4.4 Résultats obtenus :

Les tests effectués sur le modèle TCN, entraîné sur la série normalisée de , ont permis d'obtenir les métriques suivantes :

— **Sur les données non normalisées :**
  — *RMSE :* 8.791
  — *MAE :* 8.354
  — *Précision de la tendance :* 49.28%
— **Sur les données normalisées :**
  — *RMSE :* 0.213
  — *MAE :* 0.199
  — *Précision de la tendance :* 51.63%

Ces résultats montrent que le TCN est capable de capturer efficacement les dépendances temporelles à plusieurs échelles et fournit des prévisions compétitives par rapport aux modèles récurrents traditionnels. Toutefois, l'écart relatif entre les résultats obtenus sur les données non normalisées et normalisées souligne la forte sensibilité du modèle à l'échelle des données. Cela suggère qu'une normalisation adéquate est cruciale, mais également qu'une optimisation fine des hyperparamètres (nombre de couches, taille du noyau, taux de dropout, etc.) pourrait être nécessaire pour améliorer la performance globale du modèle.

4.4.5 Visualisation des résultats :

La Figure 7 ci-dessous présente la comparaison entre les valeurs réelles et les prévisions générées par le modèle TCN. Bien que le modèle suive globalement la tendance de la série temporelle, certains écarts ponctuels révèlent des marges d'amélioration pour affiner sa capacité prédictive.

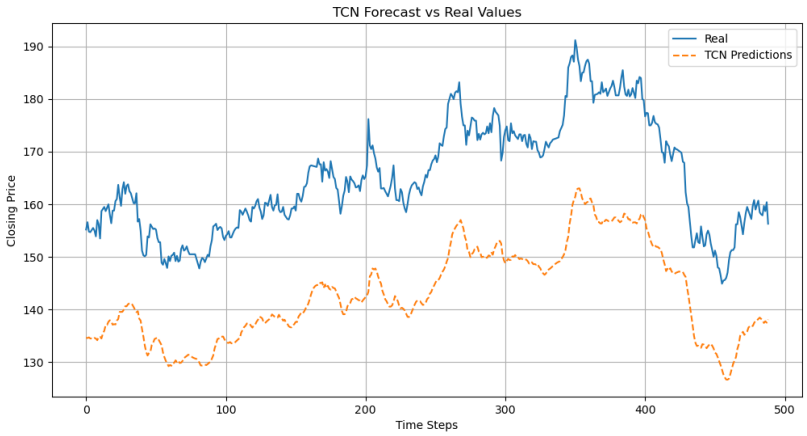

FIGURE 7 – TCN Forecast vs. Actual Values

5 Résumé des résultats et préparation des supports de présentation

Cette section synthétise l'ensemble des résultats obtenus lors de l'entraînement des différents modèles de prévision (ARIMA, LSTM, GRU Multivarié et TCN) et décrit la préparation des supports de présentation destinés à communiquer ces résultats de manière claire et accessible.

5.1 Résumé des résultats

Les différents modèles de prévision ont été évalués à l'aide de métriques classiques telles que le RMSE (Root Mean Squared Error) et le MAE (Mean Absolute Error), ainsi qu'à travers une évaluation personnalisée de la précision de la tendance (capacité à prédire correctement la direction de la variation journalière) et de la proximité entre les prévisions et les valeurs réelles. Les résultats clés sont les suivants :

- **ARIMA :**
  - *RMSE :* 1.71
  - *MAE :* 1.32
  - *Précision de la tendance :* entre 54% et 63% (selon le fractionnement Train/-Test)
- **LSTM (univarié)** :
  - Sur les données non normalisées : RMSE = 159.84, MAE = 159.49, Précision de la tendance = 47.74%
  - Sur les données normalisées : RMSE = 0.14, MAE = 0.14, Précision de la tendance = 49.59%
- **GRU Multivarié** :
  - Sur les données non normalisées : RMSE = 155.58, MAE = 155.22, Précision de la tendance = 48.77%
  - Sur les données normalisées : RMSE = 0.017, MAE = 0.014, Précision de la tendance = 48.15%
- **TCN :**

— Sur les données non normalisées : RMSE = 8.79, MAE = 8.35, Précision de la tendance = 49.28%
— Sur les données normalisées : RMSE = 0.213, MAE = 0.199, Précision de la tendance = 51.63%

Globalement, pour notre projet, le modèle ARIMA s'est révélé être le plus adapté. En effet, ses performances (RMSE de 1.71 et MAE de 1.32) témoignent d'une robustesse notable dans la prévision de la série temporelle sur l'échelle originale. De plus, une précision de la tendance variant entre 54% et 63% démontre une capacité relativement satisfaisante à anticiper la direction des variations quotidiennes. En comparaison, les modèles basés sur des réseaux de neurones (LSTM, GRU multivarié et TCN) offrent théoriquement une meilleure capacité à modéliser des relations non linéaires, mais dans notre cas, leurs résultats sont très faibles sur les données brutes, souvent quasi nuls. Ces faibles performances semblent être liées à une sensibilité excessive aux étapes de normalisation et à un surajustement sur l'échelle normalisée, limitant ainsi leur capacité à fournir des prévisions fiables sur l'échelle d'origine.

Ainsi, bien que les modèles de réseaux de neurones présentent un intérêt théorique certain, dans le contexte de notre projet, ARIMA s'est avéré être la solution la plus efficace et robuste.

5.2 Visualisation des résultats

Les graphiques générés illustrent la comparaison entre les valeurs réelles et les prévisions pour chacun des modèles. Ils montrent que, malgré quelques écarts ponctuels, le modèle ARIMA fournit une prévision plus cohérente et robuste, tandis que les modèles basés sur des réseaux de neurones peinent à fournir des résultats significatifs sur l'échelle brute. Ces visualisations confirment l'adéquation d'ARIMA pour ce projet, tout en mettant en évidence les limites des autres approches dans notre cas d'étude.

5.3 Pistes d'amélioration pour chaque modèle

Afin d'optimiser les performances et la robustesse des modèles de prévision, plusieurs pistes d'amélioration ont été identifiées pour chaque approche :

— **ARIMA :**
  — *Intégration de variables exogènes :* En passant à un modèle ARIMAX, il serait possible d'intégrer des facteurs externes influençant le cours (tels que des indicateurs macroéconomiques ou des signaux sectoriels) afin d'améliorer la qualité des prévisions.
  — *Modèles saisonniers :* L'utilisation de SARIMA (Seasonal ARIMA) permettrait de mieux capturer les variations périodiques présentes dans les données financières.
  — *Optimisation du fractionnement Train/Test :* Expérimenter avec différents ratios et techniques de pondération des erreurs pour atténuer l'accumulation des erreurs dans les prévisions multi-steps.
— **LSTM (univarié) :**
  — *Architecture plus profonde ou bidirectionnelle :* Explorer des réseaux plus profonds ou l'utilisation de LSTM bidirectionnels pour mieux capturer les dépendances temporelles complexes.

- *Mécanismes d'attention :* L'ajout d'un mécanisme d'attention pourrait permettre au modèle de se concentrer sur des périodes clés, améliorant ainsi la précision des prévisions.
- *Optimisation des hyperparamètres :* Affiner le nombre de neurones, le taux d'apprentissage et la régularisation (dropout, L2) pour améliorer la généralisation, notamment en cas de séries volatiles.

— **GRU Multivarié :**
- *Sélection et ingénierie des features :* Améliorer la qualité des features utilisées (par exemple, en incorporant des indicateurs techniques supplémentaires) afin d'exploiter pleinement l'information disponible.
- *Réglage fin des hyperparamètres :* Affiner la taille de la couche cachée, le nombre de couches et le clipping des gradients pour stabiliser davantage l'entraînement.
- *Régularisation accrue :* Intégrer des techniques supplémentaires de régularisation, telles que le dropout ou la régularisation L2, pour éviter le surapprentissage.

— **TCN :**
- *Architecture et noyaux variés :* Expérimenter avec des architectures plus profondes et des tailles de noyaux variées afin de capturer des dépendances à différentes échelles.
- *Ajustement du facteur de dilatation et du dropout :* Optimiser ces paramètres pour trouver un compromis optimal entre flexibilité et stabilité du modèle.
- *Amélioration des connexions résiduelles :* Intégrer ou renforcer les connexions résiduelles et utiliser des techniques de normalisation supplémentaires pour faciliter la convergence et améliorer les prévisions sur des horizons plus longs.

6 Conclusion Globale

Ce projet a constitué un véritable défi tant sur le plan technique que conceptuel. La mise en œuvre et l'évaluation de divers modèles de prévision, allant d'approches classiques comme ARIMA à des techniques avancées basées sur des réseaux de neurones (LSTM, GRU Multivarié, TCN), ont nécessité une phase d'apprentissage approfondie. En effet, il a fallu se documenter longuement pour identifier et comprendre les modèles d'intelligence artificielle adaptés aux séries temporelles, ainsi que leurs avantages et limites respectifs.

Par ailleurs, l'implémentation de ces modèles a présenté plusieurs difficultés. La complexité du prétraitement des données, incluant la normalisation, la création de séquences temporelles et le fractionnement Train/Test, ainsi que la compréhension et l'interprétation des graphiques générés, ont constitué des étapes particulièrement exigeantes. L'ajustement des hyperparamètres et l'optimisation des architectures, notamment pour les modèles récurrents et les réseaux de convolution temporelle, ont également demandé un investissement considérable en temps et en expertise.

En dépit de ces obstacles, le projet a permis de développer une approche comparative robuste des différentes techniques de prévision. Chaque modèle a apporté des éclairages complémentaires sur la dynamique des séries financières, tout en mettant en évidence les compromis entre complexité, interprétabilité et performance prédictive.

Enfin, ce travail a non seulement renforcé mes compétences en matière de modélisation prédictive et d'analyse de séries temporelles, mais il a également souligné l'importance de la communication des résultats. La préparation de supports de présentation clairs et structurés s'est avérée indispensable pour vulgariser des concepts techniques complexes

et pour faciliter la prise de décision basée sur les données.

En somme, ce projet illustre la richesse et la diversité des approches en intelligence artificielle appliquées aux séries temporelles, tout en mettant en lumière les défis inhérents à leur implémentation et à leur interprétation.

7 Organisation du Code et Structure du Dépôt GitHub

Pour faciliter la maintenance et la lisibilité, le code a été structuré en plusieurs fichiers, chacun dédié à une étape spécifique du processus de prévision. Voici un aperçu des principaux fichiers et de leurs fonctions :

- **data_preprocessing.py** :
  - *load_and_preprocess_data* : Charge les données depuis un fichier CSV, convertit la colonne des dates en datetime, effectue le nettoyage (notamment la conversion de la colonne) et réindexe le DataFrame pour inclure toutes les dates ouvrées.
  - *normalize_data* : Normalise les colonnes spécifiées à l'aide de MinMaxScaler.
  - *create_sequences* : Découpe une série temporelle en séquences de longueur fixe pour la préparation des entrées des modèles.
  - *visualisation* : Génère des graphiques d'analyse exploratoire des données (EDA), tels que la tendance, la distribution et les variations mensuelles.
  - *information* : Fournit des statistiques descriptives et identifie les dates manquantes.
  - *convert_volume* : Convertit les valeurs de volume (ex. "79.15M") en valeurs numériques absolues.
  - *plot_forecast* : Affiche les courbes de prévision comparées aux valeurs réelles.
  - *adjust_monthly_volatility* : Ajuste la volatilité mensuelle en réduisant les pics anormaux pour lisser la série.
- **model_arima.py** :
  - Implémente le modèle ARIMA.
  - *rolling_forecast_arima* : Réalise une prévision en rolling forecast en réentraînant le modèle à chaque itération.
- **model_lstm.py** :
  - Définit la classe `LSTMForecast` qui implémente le modèle LSTM univarié.
  - *train_and_evaluate* : Entraîne le modèle LSTM et retourne les prédictions ainsi que les métriques d'évaluation (RMSE, MAE).
- **model_gru.py** :
  - Définit la classe `GRUMultivariate` pour le modèle GRU multivarié.
  - Contient également une fonction *create_sequences* (similaire à celle de `data_preprocessing.p`) et la fonction *train_and_evaluate_gru* pour entraîner et évaluer le modèle.
- **model_tcn.py** :
  - Contient la classe `TemporalBlock` qui définit un bloc de convolution dilatée avec padding et dropout.
  - Définit la classe `TCN` qui empile plusieurs `TemporalBlock` et applique une couche linéaire pour générer la prévision.
  - Inclut la fonction *create_sequences* et *train_and_evaluate_tcn* pour l'entraînement et l'évaluation du modèle TCN.

— **evaluation.py** :
  — Contient la fonction *evaluate_forecast* qui calcule les métriques d'évaluation personnalisées, telles que la précision de la tendance et la proximité moyenne.
  — Calcule également les métriques classiques comme le RMSE et le MAE.
— **main.py** :
  — Ordonne l'exécution de toutes les parties du projet : chargement des données, prétraitement, entraînement et évaluation de chaque modèle (ARIMA, LSTM, GRU, TCN), et génération des graphiques.
  — Coordonne les appels aux différentes fonctions définies dans les autres fichiers pour produire les résultats finaux.
— **stock_price.csv** :
  — Fichier de données brutes contenant les informations sur les cours (ex. , , , , , etc.).

Chaque fichier est soigneusement commenté pour expliquer le rôle des différentes fonctions et faciliter la maintenance et l'extension du code.

8 Annexes

Remarques finales

— Le dépôt GitHub inclura tous les fichiers ci-dessus, ainsi qu'un README détaillé expliquant comment exécuter le code et reproduire les résultats.
— Les présentations (slides) s'appuieront sur ce rapport pour illustrer le processus complet : de la compréhension des données jusqu'à la validation du modèle.