République de Côte d'Ivoire

Union - Discipline - Travail

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ministère de la Communication et de l'Economie Numérique

INTERNATIONAL DATA SCIENCE INSTITUTE

Thème :

Prédiction du cours d'une action du S&P 500

RAPPORT DU PROJET

Présenté par :

- **Konan Guillaume David**
- **Kouiahon Tosseu Larissa Dorine**
- **Sagoe Christian Brice Yan**
- **Sylla Oumarou**

Année académique : 2025-2026

TABLE DES MATIÈRES

1. INTRODUCTION ET CONTEXTE

1.1 Présentation du Projet

Ce projet a pour objectif de construire un système complet de prédiction des cours boursiers de l'indice S&P500 en utilisant les techniques de Deep Learning, plus précisément les réseaux de neurones récurrents de type LSTM (Long Short-Term Memory). Le projet se compose de deux parties principales :

- Un modèle prédictif : Réseau LSTM entraîné sur 5 ans de données historiques
- Un dashboard interactif : Application Streamlit pour visualiser et interagir avec les prédictions

1.2 Qu'est-ce que le S&P500 ?

Le Standard & Poor's 500 (S&P500) est l'un des indices boursiers les plus importants au monde. Il regroupe les 500 plus grandes entreprises cotées aux États-Unis, sélectionnées selon leur capitalisation boursière. Cet indice est considéré comme un baromètre de l'économie américaine et inclut des entreprises majeures telles que :

- Apple Inc.
- Microsoft Corporation
- Amazon.com Inc.
- Alphabet Inc. (Google)
- Tesla Inc.

Pourquoi choisir le S&P500 pour ce projet ?

- Données abondantes : Historique riche et accessible gratuitement
- Représentativité : Reflète l'économie américaine dans son ensemble
- Liquidité : Volumes d'échanges importants garantissant des données fiables
- Référence académique : Largement utilisé dans la littérature scientifique

1.3 Problématique

La prédiction des marchés financiers est un problème complexe qui a fasciné les chercheurs et praticiens depuis des décennies. Les défis principaux sont :

1. Non-stationnarité : Les propriétés statistiques changent dans le temps
2. Bruit élevé : Rapport signal/bruit faible
3. Facteurs externes : Événements géopolitiques, économiques, psychologiques
4. Dépendances temporelles complexes : Les patterns se répètent sur différentes échelles de temps

Question de recherche : *Peut-on utiliser un réseau LSTM pour capturer les dépendances temporelles dans les données du S&P500 et produire des prédictions à court terme avec une précision acceptable ?*

1.4 Objectifs du Projet

Objectifs principaux :

5. Construire un modèle LSTM capable de prédire le prix de clôture du S&P500 le lendemain
6. Atteindre une erreur de prédiction (MAPE) inférieure à 3%
7. Créer un dashboard interactif pour visualiser et comprendre les prédictions

Objectifs pédagogiques :

8. Comprendre les réseaux de neurones récurrents (RNN) et leurs limitations
9. Maîtriser l'architecture LSTM et ses mécanismes de portes
10. Apprendre le prétraitement des séries temporelles
11. Savoir évaluer et interpréter les performances d'un modèle de régression
12. Développer une application complète de bout en bout (end-to-end)

2. FONDEMENTS THÉORIQUES

2.1 Séries Temporelles

2.1.1 Définition

Une série temporelle est une suite d'observations mesurées à des instants successifs, généralement à intervalles réguliers. Mathématiquement, on la note :

$$X = \{x_1, x_2, x_3, ..., x_t, ..., x_n\}$$

où x_t représente l'observation au temps t.

2.1.2 Caractéristiques des Séries Temporelles Financières

Les séries temporelles financières, comme le S&P500, présentent des propriétés spécifiques :

13. Tendance (Trend) : Mouvement général à long terme (haussier, baissier, stable)
14. Saisonnalité : Patterns qui se répètent à intervalles réguliers
15. Cycles : Fluctuations de moyen terme
16. Bruit : Variations aléatoires non prévisibles
17. Autocorrélation : Corrélation avec les valeurs passées

2.2 Réseaux de Neurones Récurrents (RNN)

2.2.1 Pourquoi les RNN pour les Séries Temporelles ?

Les réseaux de neurones traditionnels (feedforward) traitent chaque entrée de manière indépendante. Ils ne peuvent pas capturer les dépendances temporelles car ils n'ont pas de « mémoire ».

Les RNN (Recurrent Neural Networks) résolvent ce problème en introduisant des connexions récurrentes qui permettent de conserver l'information des pas de temps précédents.

2.2.2 Problème du Gradient Vanishing

Les RNN classiques souffrent d'un problème majeur : le gradient vanishing (disparition du gradient). Lors de la rétropropagation à travers le temps (BPTT), le gradient est multiplié de nombreuses fois par des valeurs souvent inférieures à 1, ce qui le fait tendre vers zéro.

Conséquences :

- Incapacité à apprendre des dépendances à long terme
- Le réseau « oublie » rapidement l'information passée
- L'entraînement devient très difficile

2.3 Architecture LSTM (Long Short-Term Memory)

2.3.1 Principe et Motivation

Les LSTM, introduits par Hochreiter & Schmidhuber en 1997, ont été conçus spécifiquement pour résoudre le problème du gradient vanishing. L'idée clé est d'introduire un mécanisme de mémoire à long terme contrôlé par des portes.

2.3.2 Architecture d'une Cellule LSTM

Une cellule LSTM est composée de :

18. Cell State (C_t) : La « mémoire à long terme » de la cellule
19. Hidden State (h_t) : La sortie de la cellule
20. Trois portes : Forget, Input, Output

2.3.3 Les Trois Portes du LSTM

1. Forget Gate (Porte d'oubli) - f_t

Décide quelle information de la mémoire précédente doit être oubliée. σ = fonction sigmoïde (sortie entre 0 et 1). 0 = oublier complètement, 1 = garder complètement.

2. Input Gate (Porte d'entrée) - i_t

Décide quelle nouvelle information doit être ajoutée à la mémoire. i_t représente l'importance de la nouvelle information (0 à 1).

3. Output Gate (Porte de sortie) - o_t

Décide quelle partie de la mémoire doit être utilisée pour la sortie.

2.3.4 Avantages du LSTM

21. Capture des dépendances à long terme grâce au cell state
22. Évite le gradient vanishing : les gradients peuvent circuler sans multiplication excessive
23. Flexibilité : les portes s'adaptent automatiquement selon les données
24. Robustesse : fonctionne bien sur de nombreux problèmes de séquences

2.4 Deep Learning avec PyTorch

2.4.1 Pourquoi PyTorch ?

PyTorch est un framework de Deep Learning développé par Meta (Facebook) qui offre :

25. Définition dynamique du graphe : Plus intuitif et flexible
26. Interface pythonique : Facile à apprendre et debugger
27. Support GPU : Accélération matérielle automatique
28. Écosystème riche : Nombreuses bibliothèques et ressources
29. Adoption académique : Standard dans la recherche

3. MÉTHODOLOGIE

3.1 Vue d'Ensemble du Pipeline

Notre projet suit un pipeline de Machine Learning classique adapté aux séries temporelles :

30. Collecte des Données : Yahoo Finance (yfinance) - 5 ans de S&P500
31. Exploration (EDA) : Statistiques, visualisations, corrélations
32. Prétraitement : Normalisation (MinMaxScaler), création des séquences, Split Train/Test (80/20)
33. Construction du Modèle : Architecture LSTM (2 couches, 64 neurones)
34. Entraînement : Loss MSE, Optimizer Adam, 250 epochs, batch size 32
35. Évaluation : Métriques RMSE, MAE, MAPE, visualisations
36. Déploiement : Sauvegarde du modèle, Dashboard Streamlit

3.2 Choix de Configuration

3.2.1 Horizon de Prédiction

Choix : Prédiction à 1 jour (J+1)

Justification :

- Complexité maîtrisable : Prédire plusieurs jours est exponentiellement plus difficile
- Précision maximale : L'erreur s'accumule sur les prédictions multi-horizons
- Utilité pratique : Pertinent pour le trading à court terme
- Évaluation claire : Comparaison directe réalité vs prédiction

3.2.2 Période Historique

Choix : 5 ans de données

Justification :

- Équilibre quantité/pertinence : ~1250 jours de trading, assez récent pour être pertinent
- Capture de différents régimes de marché : Période COVID-19 (2020), Reprise post-COVID (2021-2022), Volatilité récente (2023-2026)

3.2.3 Features Utilisées

Choix : 5 features OHLCV

Feature	Description	Justification
Open	Prix d'ouverture	Gap overnight, sentiment initial

High	Plus haut du jour	Résistance, volatilité
Low	Plus bas du jour	Support, volatilité
Close	Prix de clôture	Variable cible, prix de référence
Volume	Nombre d'actions	Liquidité, force de la tendance

3.2.4 Longueur de Séquence

Choix : 60 jours (environ 3 mois de trading)

Justification :

- Empirique : Valeur couramment utilisée dans la littérature
- Cycles de marché : Capture des tendances à moyen terme
- Compromis : Trop court (10 jours) = manque de contexte, trop long (200 jours) = perte de pertinence

3.3 Stack Technologique

Bibliothèque	Version	Rôle
PyTorch	2.0+	Framework Deep Learning
NumPy	1.24+	Calcul numérique
Pandas	2.0+	Manipulation de données
Scikit-learn	1.3+	Prétraitement, métriques
Matplotlib	3.7+	Visualisation statique
yfinance	0.2+	Téléchargement données financières
Streamlit	1.28+	Dashboard interactif

4. COLLECTE ET EXPLORATION DES DONNÉES

4.1 Source de Données : Yahoo Finance

yfinance est une bibliothèque Python open-source qui permet de télécharger gratuitement des données financières historiques depuis Yahoo Finance.

Avantages :

- Gratuit et sans API key
- Données fiables et mises à jour quotidiennement
- Interface simple et intuitive
- Large couverture (actions, indices, cryptos, devises)

4.2 Statistiques Descriptives

Période couverte : 2021-01-19 à 2026-01-16 (5 ans)

Nombre de jours de trading : 1256 jours

Observations clés :

37. Tendance haussière : Prix min $3577 (2021) → Prix max $6977 (2026), soit +95% sur 5 ans
38. Volatilité : Écart-type ~$907 (≈18.5% du prix moyen)
39. Volume moyen : 4.4 milliards de dollars/jour
40. Aucune valeur manquante : Pas besoin d'imputation

4.3 Insights de l'Analyse Exploratoire

Conclusions de l'EDA :

- ✓ Données de qualité : Complètes, cohérentes, sans anomalies majeures
- ✓ Variabilité suffisante : Différents régimes de marché représentés
- ✓ Tendances identifiables : Patterns visuellement détectables
- ⚠ Non-stationnarité : Moyenne et variance changent dans le temps
- ⚠ Volatilité hétéroscédastique : Périodes calmes et agitées alternent

5. PRÉTRAITEMENT DES DONNÉES

5.1 Normalisation avec MinMaxScaler

5.1.1 Pourquoi Normaliser ?

La normalisation est une étape cruciale pour les réseaux de neurones. Sans elle, les performances et la stabilité de l'entraînement sont gravement compromises.

Problèmes sans normalisation :

41. Échelles différentes : Close ~3000-7000, Volume ~3-10 milliards → Le Volume dominerait l'apprentissage
42. Gradients instables : Grandes valeurs → gradients explosifs, Petites valeurs → gradients évanescents
43. Convergence lente : L'optimiseur doit naviguer dans un espace déformé
44. Activations saturées : Valeurs > 1 saturent la sigmoïde/tanh

5.1.2 MinMaxScaler : Principe

Formule mathématique :

```
X_normalized = (X - X_min) / (X_max - X_min)
```

Propriétés : Output entre 0 et 1, préserve la forme de la distribution, sensible aux outliers.

5.2 Création des Séquences (Sliding Window)

Un LSTM ne traite pas les données point par point, mais par séquences. L'idée est de créer des échantillons qui contiennent l'historique récent.

Principe : *« Pour prédire le prix de DEMAIN, utilisons les N derniers jours »*

Résultat après création des séquences :

- X shape : (1196, 60, 5) → 1196 échantillons, 60 jours d'historique, 5 features
- y shape : (1196,) → 1196 cibles (prix de clôture du jour suivant)

5.3 Split Train/Test

Pour les séries temporelles, le split doit respecter l'ordre chronologique. On ne peut pas mélanger (shuffle) les données comme dans un problème classique de ML.

Raison :

On veut évaluer la capacité du modèle à prédire le futur. Utiliser des données futures pour prédire le passé serait de la triche (data leakage).

Résultat :

- Train : 956 samples (80%)

- Test : 240 samples (20%)

6. ARCHITECTURE DU MODÈLE LSTM

6.1 Conception de l'Architecture

6.1.1 Choix des Hyperparamètres

Hyperparamètre	Valeur	Justification
input_size	5	Nombre de features (OHLCV)
hidden_size	64	Capacité du modèle (compromis complexité/overfitting)
num_layers	2	Profondeur (capture de patterns hiérarchiques)
dropout	0.2	Régularisation (20% de neurones désactivés aléatoirement)
output_size	1	Prédiction d'une seule valeur (Close demain)

6.1.2 Schéma de l'Architecture

Le modèle LSTM complet est structuré comme suit :

- INPUT : [batch_size, 60, 5] - Séquences de 60 jours avec 5 features
- LSTM Layer 1 : 64 neurones, traite la séquence de 60 timesteps
- Dropout (20%) : Régularisation entre les couches
- LSTM Layer 2 : 64 neurones, raffine les patterns
- Extraction : On prend seulement la sortie du dernier timestep (t=60)
- Fully Connected Layer : Transformation linéaire 64 → 1
- OUTPUT : [batch_size, 1] - Prédiction du Close demain

6.2 Analyse du Nombre de Paramètres

Calcul théorique :

- LSTM Layer 1 : $4 \times [(5 + 64 + 1) \times 64] = 17,920$ paramètres
- LSTM Layer 2 : $4 \times [(64 + 64 + 1) \times 64] = 33,024$ paramètres
- Couche Linear : $(64 \times 1) + 1 = 65$ paramètres
- Total : ~51,000 paramètres entraînables

Interprétation : 51k paramètres représente un modèle de taille moyenne, ni trop petit (underfitting), ni trop grand (overfitting). Suffisant pour capturer les patterns sans surapprentissage excessif.

7. ENTRAÎNEMENT DU MODÈLE

7.1 Configuration de l'Entraînement

7.1.1 Fonction de Perte : MSELoss

Formule mathématique :

$$MSE = (1/n) \times \Sigma_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Pourquoi MSE pour ce problème ?

45. Régression : On prédit une valeur continue (prix)
46. Pénalisation des grandes erreurs : Le carré amplifie les erreurs importantes
47. Différentiabilité : Dérivée simple pour la rétropropagation
48. Standard de l'industrie : Largement utilisé pour les prédictions de prix

7.1.2 Optimiseur : Adam

Adam (Adaptive Moment Estimation) combine les avantages de Momentum (moyenne des gradients passés) et RMSprop (adapte le learning rate par paramètre).

Pourquoi Adam ?

49. Adaptatif : Learning rate différent pour chaque paramètre
50. Robuste : Fonctionne bien sans tuning excessif
51. Convergence rapide : Accélération grâce au momentum
52. Standard : Choix par défaut dans la communauté

7.1.3 Hyperparamètres d'Entraînement

Hyperparamètre	Valeur	Justification
epochs	250	Convergence observée vers epoch 200-250
batch_size	32	Compromis vitesse/stabilité, standard
learning_rate	0.0001	Convergence stable avec Adam
shuffle	True	Évite l'overfitting sur l'ordre des batches

7.2 Résultats de l'Entraînement

Évolution de la loss pendant l'entraînement :

- Epoch 10 : Loss = 0.001843

- Epoch 50 : Loss = 0.000973
- Epoch 100 : Loss = 0.000740
- Epoch 150 : Loss = 0.000628
- Epoch 200 : Loss = 0.000557
- Epoch 250 : Loss = 0.000457

Observations positives :

- ✓ Décroissance monotone : La loss diminue régulièrement
- ✓ Pas d'oscillations : Courbe lisse, entraînement stable
- ✓ Convergence atteinte : Plateau vers epoch 200
- ✓ Réduction significative : 0.001843 → 0.000457 (75% de réduction)

8. ÉVALUATION ET ANALYSE DES RÉSULTATS

8.1 Métriques d'Évaluation

Une seule métrique ne suffit pas pour évaluer complètement un modèle de régression. Chaque métrique apporte un éclairage différent.

- RMSE (Root Mean Squared Error) : Sensible aux outliers, même unité que la cible
- MAE (Mean Absolute Error) : Robuste aux outliers, interprétation intuitive
- MAPE (Mean Absolute Percentage Error) : Indépendant de l'échelle, en pourcentage

8.2 Résultats Obtenus

Métrique	Train	Test
RMSE ($)	66.31	107.62
MAE ($)	49.96	80.30
MAPE (%)	1.12	1.30

8.3 Interprétation des Résultats

Sur le Test Set (données non vues) :

53. RMSE = $107.62 : Le modèle se trompe d'environ $108 en moyenne (< 2% du prix). ÉVALUATION : BON
54. MAE = $80.30 : L'erreur absolue médiane est de $80 (< 1.5% du prix). ÉVALUATION : TRÈS BON
55. MAPE = 1.30% : Le modèle se trompe de 1.3% en moyenne. ÉVALUATION : EXCELLENT (benchmark < 3%)

8.4 Benchmark et Contexte

Que signifie un MAPE de 1.30% en finance ?

Modèle	MAPE typique	Commentaire
Naïve (prix d'hier)	1.5-2%	Baseline la plus simple
Moyenne Mobile	2-3%	Méthode classique
ARIMA	1.5-2.5%	Méthode statistique

Notre LSTM	1.30%	Comparable ou meilleur
Modèles professionnels	0.8-1.5%	Ensembles complexes

Conclusion : Notre modèle atteint des performances professionnelles pour un projet pédagogique.

8.5 Forces et Faiblesses du Modèle

Forces :

- Excellente précision globale (MAPE 1.30%)
- Généralisation correcte (pas d'overfitting sévère)
- Capture de la tendance générale du marché
- Stabilité (pas de prédictions aberrantes)
- Convergence rapide (~5 minutes sur CPU)

Faiblesses :

- Difficulté avec la volatilité (erreurs plus élevées lors de mouvements brusques)
- Prédictions conservatrices (sous-estime les pics, sur-estime les creux)
- Horizon limité (J+1 uniquement)
- Absence de données externes (pas de sentiment de marché, news, etc.)

9. DASHBOARD INTERACTIF

9.1 Présentation du Dashboard

Le dashboard Streamlit transforme notre modèle LSTM en une application interactive accessible sans connaissances en programmation. Il permet de :

56. Visualiser les données historiques du S&P500
57. Consulter les prédictions du modèle
58. Évaluer les performances avec des métriques détaillées
59. Comprendre le modèle grâce aux explications intégrées

9.2 Structure des Fichiers

```
dashboard.py              # Application principale
├── src/
│   ├── model.py          # Définition et chargement du LSTM
│   ├── data_loader.py    # Fonctions de manipulation de données
│   └── utils.py          # Fonctions utilitaires
├── models/
│   ├── lstm_model.pth    # Poids du modèle entraîné
│   ├── scaler.pkl        # Scaler pour normalisation
│   └── config.json       # Configuration et métadonnées
└── data/
    └── sp500_data.csv    # Données historiques
```

10. CONCLUSIONS

10.1 Récapitulatif des Objectifs Atteints

- ✓ Objectif 1 : Construire un modèle LSTM fonctionnel - Architecture à 2 couches, 64 neurones, 51,521 paramètres entraînés sur 5 ans de données S&P500
- ✓ Objectif 2 : Atteindre une précision acceptable - MAPE de 1.30% (excellent, < 3% visé), MAE de $80.30 (< 1.5% du prix moyen)
- ✓ Objectif 3 : Créer un dashboard interactif - Application Streamlit avec visualisations Plotly
- ✓ Objectif 4 : Approche pédagogique complète - Documentation détaillée, code commenté, reproductible

10.2 Perspectives d'Amélioration

60. Ajouter des indicateurs techniques (SMA, RSI, MACD) comme features supplémentaires
61. Intégrer des données externes (VIX, taux d'intérêt, sentiment de marché)
62. Tester des architectures plus avancées (Transformer, Attention mechanisms)
63. Implémenter l'early stopping et la validation croisée temporelle
64. Étendre à la prédiction multi-horizon (J+1, J+5, J+10)

11. RÉFÉRENCES

11.1 Articles Scientifiques

65. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780 → Article fondateur du LSTM
66. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press → Référence complète sur le Deep Learning
67. Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. arXiv:1308.0850
68. Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669

11.2 Documentation Technique

- PyTorch Documentation : https://pytorch.org/docs/stable/index.html
- Streamlit Documentation : https://docs.streamlit.io/
- Scikit-learn Documentation : https://scikit-learn.org/stable/
- yfinance Documentation : https://pypi.org/project/yfinance/

11.3 Ressources en Ligne

- Understanding LSTM Networks (Colah's Blog) : https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Time Series Forecasting with Deep Learning (TensorFlow) : https://www.tensorflow.org/tutorials/structured_data/time_series

ANNEXES

A. Configuration Complète


```
{
  "config": {
    "sequence_length": 60,
    "hidden_size": 64,
    "num_layers": 2,
    "dropout": 0.2,
    "batch_size": 32,
    "epochs": 250,
    "learning_rate": 0.0001,
    "train_split": 0.8
  },
  "metrics": {
    "train_rmse": 66.31,
    "train_mae": 49.96,
    "train_mape": 1.12,
    "test_rmse": 107.62,
    "test_mae": 80.30,
    "test_mape": 1.30
  }
}
```


B. Glossaire

- Adam : Algorithme d'optimisation adaptatif combinant momentum et RMSprop
- Autograd : Système de calcul automatique des gradients en PyTorch
- Backpropagation : Algorithme pour calculer les gradients dans un réseau de neurones
- Batch : Sous-ensemble de données traité simultanément
- Cell State : Mémoire à long terme d'une cellule LSTM
- Dropout : Technique de régularisation désactivant aléatoirement des neurones
- Epoch : Une passe complète sur l'ensemble des données d'entraînement
- Gradient Vanishing : Problème où les gradients deviennent trop petits
- Hidden State : État caché d'un RNN, sortie de la cellule
- Learning Rate : Taille du pas lors de la mise à jour des poids
- Loss Function : Fonction mesurant l'erreur du modèle
- LSTM : Long Short-Term Memory, type de RNN
- MAE : Mean Absolute Error, erreur absolue moyenne
- MAPE : Mean Absolute Percentage Error, erreur en pourcentage
- Normalisation : Mise à l'échelle des données (ex: 0-1)
- Overfitting : Modèle qui mémorise les données d'entraînement
- RNN : Recurrent Neural Network, réseau de neurones récurrent
- RMSE : Root Mean Squared Error, racine de l'erreur quadratique moyenne