Anthony VONG

EFREI M2 – ITF

Rattrapage Engineering Project

IT For Finance

Prévisualisation du Prix de Clôture des Actifs Financiers

Sommaire

I. Contextualisation

1. Présentation du contexte et de l'importance de la prédiction des prix des actifs financiers

L'analyse financière est un domaine qui a connu une transformation profonde avec l'avènement de l'informatique et du Big Data. La prédiction des prix des actifs financiers est un enjeu majeur pour les investisseurs, les entreprises de trading et les institutions financières. Cette tâche est complexe car elle dépend de nombreux facteurs, allant des informations économiques aux nouvelles spécifiques à une entreprise, en passant par les dynamiques du marché.

Avec le développement des technologies de l'information, une quantité de données financières est désormais disponible pour l'analyse. L'utilisation de modèles d'apprentissage automatique pour l'analyse prédictive devient de plus en plus courante. Ces modèles peuvent aider à identifier des modèles et des tendances qui seraient difficiles, voire impossibles, à détecter par des méthodes d'analyse traditionnelles.

2. Présentation de l'objectif du projet

L'objectif de ce projet est de développer un outil de prédiction des prix des actifs financiers en utilisant une approche de deep learning, plus précisément un type de réseau de neurones récurrents appelé Long Short-Term Memory (LSTM). L'outil récupère des données historiques sur les prix de clôture d'un actif financier, entraîne un modèle LSTM sur ces données, puis utilise ce modèle pour prédire les prix futurs.

3. Présentation de LSTM et une explication de pourquoi cette méthode

Les réseaux de neurones récurrents (RNN) sont une catégorie de réseaux de neurones artificiels qui sont particulièrement efficaces pour traiter des séquences de données, comme les séries temporelles ou les textes. Cependant, les RNN classiques ont du mal à gérer les dépendances à long terme dans les données, à cause des problèmes de gradient qui disparaît ou explose.

Pour résoudre ce problème, le Long Short-Term Memory (LSTM), une variante des RNN, a été proposé. Les LSTM sont conçus de manière à pouvoir apprendre et se souvenir d'informations sur de longues périodes, ce qui les rend particulièrement efficaces pour des tâches comme la prédiction de séries temporelles.

Dans notre cas, les LSTM sont appropriés car les prix des actifs financiers sont une forme de série temporelle et la prédiction de ces prix peut dépendre de l'information contenue dans des observations lointaines. En utilisant des LSTM, nous pouvons donc capturer ces dépendances à long terme et améliorer la précision de nos prédictions.

II. Introduction

La prédiction des prix des actifs financiers a toujours été un domaine d'intérêt important pour les investisseurs, les analystes et les chercheurs. La capacité à anticiper les mouvements de prix peut offrir des opportunités lucratives et permettre de prendre des décisions d'investissement plus informées. Cependant, la prédiction des prix est une tâche ardue en raison de la nature volatile et incertaine des marchés financiers. De nombreux facteurs, allant des événements macroéconomiques aux changements de sentiment des investisseurs, peuvent influencer les prix des actifs financiers, rendant leur prédiction complexe.

Dans ce contexte, l'apprentissage automatique et l'intelligence artificielle sont devenus des outils précieux pour modéliser et prédire les mouvements de prix. Ces techniques permettent de traiter de grandes quantités de données, d'apprendre de modèles complexes et de faire des prédictions sur la base de ces apprentissages.

Dans ce projet, nous allons explorer comment une forme spécifique d'apprentissage en profondeur, connue sous le nom de réseaux de neurones à longue mémoire à court terme (LSTM), peut être utilisée pour prédire les prix de clôture des actifs financiers. Les LSTM peuvent apprendre et retenir des informations sur de longues périodes, ce qui est crucial pour comprendre les tendances des prix sur le marché financier.

Le rapport suivant présente notre démarche pour la conception et l'implémentation d'un modèle LSTM pour prédire les prix de clôture des actifs financiers. Il détaille également les différentes étapes du processus, de la récupération et de la préparation des données à l'entraînement et à l'évaluation du modèle. Enfin, le rapport comprend une analyse des résultats obtenus et des perspectives pour des travaux futurs.

III. Description Technique – Code

1. Importation des bibliothèques nécessaires

```
# Importation des bibliothèques nécessaires
import math
from pandas_datareader import data as pdr
import numpy as np
import yfinance as yf
yf.pdr_override()
from datetime import date
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
from tkinter import *
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
plt.style.use('fivethirtyeight')
```

Cette section importe toutes les bibliothèques nécessaires pour exécuter le script. Ces bibliothèques incluent des packages tel que 'pandas_datareader' ou 'numpy' pour l'analyse de données, la visualisation de données 'matplotlib', la création de l'interface graphique 'tkinter', ou encore 'sklearn' et 'keras' pour le deep learning.

2. *Les fonctions principales*

```
# Calcul du rendement total et la volatilité annuel via le df et prix de cloture
def calculate_returns(df):
    # calcul le retour quotidien
    df['daily_return'] = df['Close'].pct_change()
    # calcul le rendement total sur investissement
    retour_inves = np.prod(1 + df['daily_return']) - 1
    # calcul de la volatilité annualisé en sachant que 252 = le nombre de jour moyen de trading par an
    volatilite_an = df['daily_return'].std()*np.sqrt(252)
    return retour_inves, volatilite_an
```

Cette fonction permet de calculer le rendement total et la volatilité annuelle des prix de clôture d'un actif financier.

Le rendement total est le rendement global de l'investissement sur la période donnée (ex : début_date = '2002-11-05' ; fin_date = '2017-11-05', sur cet exemple sur 15 ans).

La volatilité est une mesure de la variabilité des rendements sur cette période. Un actif trop volatile n'est pas conseiller sur du long terme, mais sur du court-terme si l'on fait du scalping par exemple c'est un bon choix.

Ces deux valeurs fournissent des informations clés sur la performance et le risque de l'actif.

```python
# Pour refresh les visualisations graphiques s'il y a plusieurs input
def clear_canvas():
    for widget in window.winfo_children():
        if isinstance(widget, Canvas):
            widget.destroy()
```

Cette fonction efface toutes les visualisations graphiques précédemment générées par l'application. Elle est utilisée pour nettoyer l'interface utilisateur avant d'afficher une nouvelle prédiction.

Prenons un exemple si je n'avais pas implémenté cette fonction :

Je fais une prédiction sur AAPL (Apple), un graph sur la prédiction de l'actif financier apparait.

Je continue et je souhaite faire une prédiction sur GOOGL (Google), un graph apparait mais le graph d'Apple persiste.

Cette fonction est plus ergonomique pour l'utilisateur et permet une interface plus claire.

```python
# Plot sur canvas (via matploblib)
def plot_on_canvas(figure):
    canvas = FigureCanvasTkAgg(figure, master=window)
    canvas.draw()
    canvas.get_tk_widget().pack()
```

Cette fonction prend une figure Matplotlib et l'affiche sur le canevas Tkinter. Comme dit précédemment Matplotlib permet la visualisation de données tandis que Tkinter permet seulement de créer une interface graphique, il faut coupler les deux afin d'avoir un résultat productif. C'est à dire une interface graphique qui puisse accueillir des données mathématiques sous graph.

```python
def plot_prediction(symbol, date_debut, date_fin):
```

Et voici le plus gros du code la fonction def plot_prediction(symbol, date_debut, date_fin):

Je vais détailler ligne par ligne cette fonction car c'est la fonction principale du code qui permet la prédiction et l'application du modèle LSTM.

```python
try:
    # Récupérer les données de l'actif financier
    df = pdr.get_data_yahoo(symbol, start=date_debut, end=date_fin)
    retour_inves, volatilite_an = calculate_returns(df)
    # + Rendement total et Volatilité annualisée
    summary_text.set(f'Rendement total: {retour_inves*100:.2f}%\n'
                     f'Volatilité annualisée: {volatilite_an*100:.2f}%')
except:
    # Message d'erreur affiché si l'actif financier n'existe pas (ex : GOOGLE au lieu de GOOGL)
    error_text.set("L'actif financier n'existe pas ou n'est pas dans la base de données.")
    return None
```

La fonction commence par un try/except, le try prendra la donnée de l'utilisateur c'est à dire l'actif financier, la date du début, et la date de fin.

Exemple : AAPL, 2021-05-05, 2023-05-05.

Si l'actif financier n'existe pas ou ne fait pas partit de la base de données (except:)alors nous afficherons un message d'erreur "L'actif financier n'existe pas ou n'est pas dans la base de données.".

La ligne df = pdr.get_date_yahoo(symbol,start=date_debut, end=date_fin) utilise la fonction 'get_data_yahoo' du module 'pandas_datareader' afin de récupérer les données historiques de l'actif financier de Yahoo Finance. Les données sont donc récupérées et stockées dans notre dataframe 'df'.

La ligne retour_inves, volatilite_an = calculate_returns(df) utilise la fonction précédemment définie 'calculate_returns' afin de calculer le rendement total et la volatilité annuelle des prix de clôture de l'actif financier.

summary_text.set(f'Rendement total: {retour_inves*100:.2f}%\n' f'Volatilité annualisée: {volatilite_an*100:.2f}%') permet de mettre à jour le texte du résumé dans l'IU pour afficher le rendement total et la volatilité annuelle calculés.

```
data = df.filter(['Close'])
```

Cette ligne permet de créer un nouveau dataframe filtré sur la colonne 'close' du dataframe original 'df'. Nous le filtrons car nous utiliserons et étudierons le prix de clôture du marché qui correspond à la colonne 'Close'.

```
dataset = data.values
```

Cette ligne convertit le dataframe 'data' en un tableau numpy afin de pouvoir l'utiliser avec la bibliothèque Keras.

```
tendance_dlong = math.ceil(len(dataset) * 0.8)
```

Cette ligne calcule le nombre de lignes de données à utiliser pour l'application de notre modèle LSTM, il utilise 80% des données pour cela (ce qui est recommandé d'après les mathématiciens!).

```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
```

La première ligne permet de crée un objet MinMaxScaler pour échelonner les données entre 0 et 1.

Par exemple un actif financier quelconque à date et prix :

2021-05-05 99$

2023-05-05 122$

Seront échelonnés entre 0 et 1.

La deuxième ligne normalise les données de clôture en utilisant l'objet précédemment crée c'est à dire MinMaxScaler.

```
tendance_data = scaled_data[0:tendance_dlong , :]
tendance_x = []
tendance_y = []
```

Cela permet de crée un nouveau tableau contenant seulement les données d'applications.

```
for i in range(60, len(tendance_data)):
    tendance_x.append(tendance_data[i-60:i, 0])
    tendance_y.append(tendance_data[i, 0])
```

Ici, tendance_x contient les 60 valeurs précédentes de la série de données tandis que tendance_y contient la valeur actuelle, cela nous permettra d'appliquer notre modèle LSTM.

```
lstm = Sequential()
lstm.add(LSTM(50, return_sequences=True, input_shape= (tendance_x.shape[1], 1)))
lstm.add(LSTM(50, return_sequences=False))
lstm.add(Dense(25))
lstm.add(Dense(1))
```

Ici nous avons la création du modèle lstm.

La première ligne initialise un modèle séquentiel pour Keras.

La deuxième ligne ajoute la première couche LSTM au modèle. Cette couche a 50 unités et retourne des séquences complètes, qui sont nécessaires lorsqu'une autre couche LSTM suit la couche actuelle.

La troisième ligne ajoute la seconde couche LSTM au modèle. Cette couche a aussi 50 unités, mais retourne seulement la dernière sortie de la séquence.

Tandis que la quatrième ligne et la cinquième ligne, ajoutent deux couches au modèle. La première a 25 unités et la seconde a 1 unité qui est la sortie finale du modèle.

```
lstm.compile(optimizer='adam', loss='mean_squared_error')
```

Ici on compile le modèle LSTM, en utilisant l'optimiseur Adam (**utilisé pour la formation de modèles d'apprentissage profond**. Il s'agit d'une extension de la descente de gradient stochastique.). La fonction perte est representée par l'erreur quadratique moyen. L'erreur quadratique moyen est utilisée pour les problèmes de régression ce qui est le cas ici.

```
lstm.fit(tendance_x, tendance_y, batch_size=1, epochs=1)
```

Cette ligne applique le modèle LSTM sur nos données.

Batch_size est fixé à 1, le modèle est mis à jour après chaque échantillon.

Epochs est aussi fixé à 1, ce qui signifie que le modèle passe une seule fois à travers l'ensemble des données.

C'est ici que tous les calculs se font.

```
predictions = lstm.predict(test_x)
predictions = scaler.inverse_transform(predictions)
```

Ces deux lignes permettent de faire les prédictions sur les données testés, c'est à dire qu'elles se rééchelonneront pour qu'elles soient le plus proches de l'échelle original des prix des clôtures.

```
tendance = data[:tendance_dlong]
pred = data[tendance_dlong:]
pred['Predictions'] = predictions
```

Ici, tendance contient les valeurs réelles de l'actif financier tandis que pred contient les valeurs durant la période de prédiction.

La colonne prédiction est ajoutée pour contenir les prédictions du modèle.

```
figure = plt.Figure(figsize=(15,8), dpi=100)
ax = figure.add_subplot(111)
ax.plot(tendance['Close'])
ax.plot(pred[['Close', 'Predictions']])
ax.set_title('LSTM Prevision')
ax.set_xlabel('Date')
ax.set_ylabel('Prix de Cloture ($)')
ax.legend(['Tendance', 'Valeur', 'Prevision'], loc='upper left')

return figure
```

Et enfin, nous avons la création du graphique de la tendance réelle et des prédictions du modèle. La fonction 'return figure' retourne la figure créée.

3. *Configuration et initialisation de l'interface utilisateur avec Tkinter*

Cette partie du script crée une interface utilisateur simple avec Tkinter. L'IU permet à l'utilisateur d'entrer le symbole d'un actif financier et une plage de dates, puis ensuite de générer une prédiction en cliquant sur un bouton.

```
# Configuration de l'interface utilisateur Tkinter
def predict():
    clear_canvas()
    error_text.set("")
    figure = plot_prediction(entry_text.get(), date_debut_text.get(), date_fin_text.get())

    if figure:
        plot_on_canvas(figure)
```

Cette fonction est appelée lorsque l'utilisateur clique sur le bouton "Prédire". Elle efface toutes les visualisations précédentes, récupère les entrées de l'utilisateur, génère une prédiction en utilisant la fonction plot_prediction et enfin affiche le graphique résultant dans l'interface graphique.
Il faut noter que plus l'écart entre les dates est important, plus les calculs seront long. Il se peut que pour une plage de 15 ans, vous deviez attendre 30 sec afin que les calculs se fassent et que le graphique de prédiction s'affiche.

```
#Création visuelle

entry_text = StringVar()
entry_label = Label(window, text="Actif Financier (ex : AAPL pour Apple)")
entry_label.pack()
entry = Entry(window, textvariable=entry_text)
entry.pack()

date_debut_text = StringVar()
date_debut_label = Label(window, text="Date de départ (YYYY-MM-DD):")
date_debut_label.pack()
date_debut_entry = Entry(window, textvariable=date_debut_text)
date_debut_entry.pack()

date_fin_text = StringVar()
date_fin_label = Label(window, text="Date de fin (YYYY-MM-DD):")
date_fin_label.pack()
date_fin_entry = Entry(window, textvariable=date_fin_text)
date_fin_entry.pack()

button = Button(window, text='Prédire', command=predict)
button.pack()

summary_text = StringVar()
summary_label = Label(window, textvariable=summary_text)
summary_label.pack()

error_text = StringVar()
error_label = Label(window, textvariable=error_text, fg="red")
error_label.pack()

window.mainloop()
```

La première fonction ‘NOM_text = StringVar()’ crée une instance, c’est une fonctionnalité de Tkinter pour stocker des chaînes de caractères qui seront utilisées. (ex : AAPL)

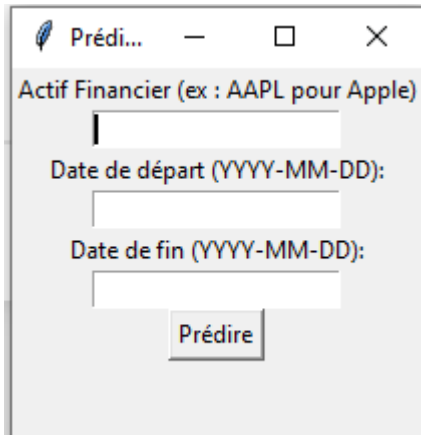
La deuxième fonction ‘NOM_label = Label(window, text=”) crée un widget Label qui est utilisé pour afficher du texte. Par exemple dans notre premier bloc de code cela permet de visualisé dans l’interface graphique : “Actif Financier (ex : AAPL pour Apple)”.

La troisième fonction ‘NOM_label.pack” permet simplement de rajouter le label à la fenêtre.

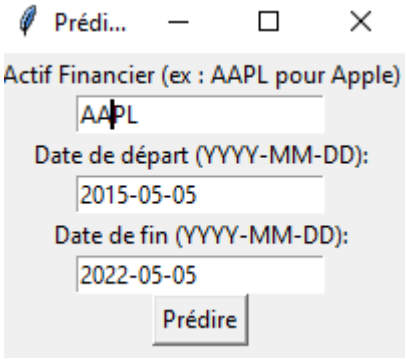
La quatrième fonction ‘NOM = Entry(window, textvariable=’var’) crée un widget qui permet une saisie de texte.

Enfin la cinquième fonction NOM.pack() permet d’ajouter le champ de saisie de texte à la fenêtre.

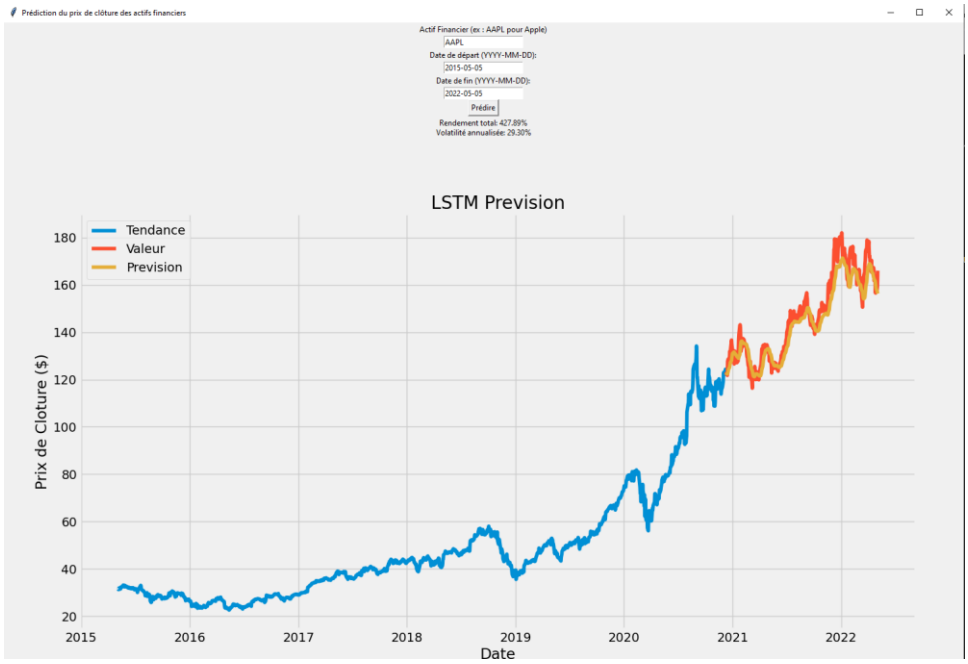
4. Visualisation du logiciel



J’aimerais faire une prédiction en utilisant le modèle LSTM avec pour premier actif AAPL (Apple) sur les plages 2015-05-05;2022-05-05.



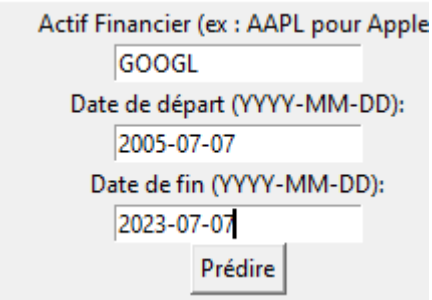
Je clique sur prédire et attend que les calculs se fassent, voici le résultat :



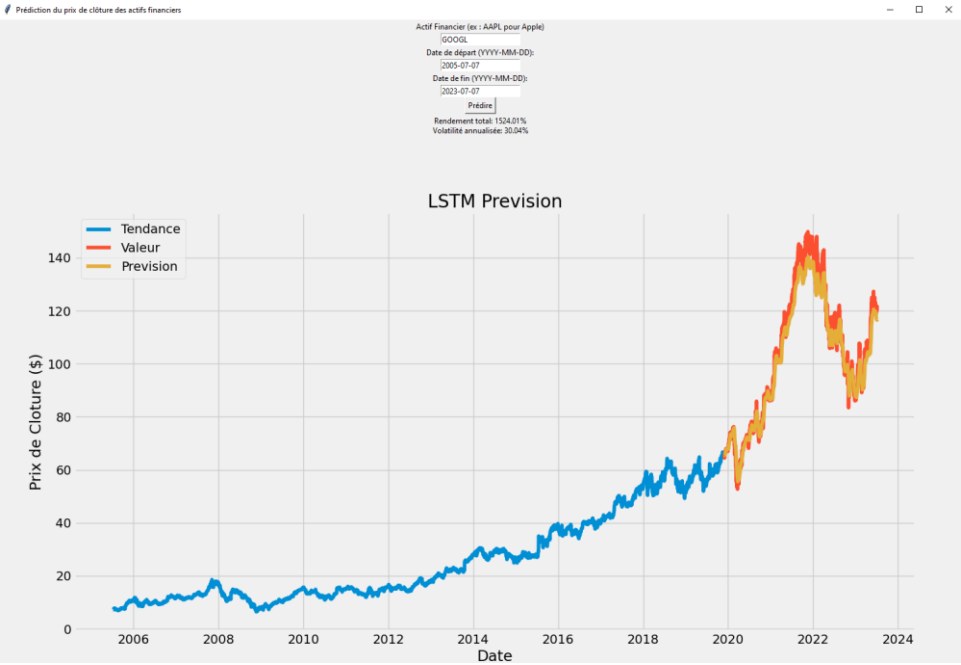
Nous apercevons donc la tendance ainsi que la valeur réelle et la prévision qui se rapproche énormément de la valeur réelle. Ce qui laisse à dire que LSTM sur des plages de données grande peut nous aider réellement à prédire un actif financier dans le futur.

Un rendement de 427.89% sur 7 ans et une volatilité de 29.30%. C’est un bon actif sur le long terme.

Maintenant sur la même fenêtre j’aimerais par exemple étudier GOOGL (Google) sur les plages 2005-07-07;2023-07-07.



Je clique sur Prédire et voici le résultat :



Voici la représentation de Google sur 18 ans et nous pouvons prendre la même conclusion que Apple.

Conclusion

Le but initial de mon projet était de prédire les prix de clôture des actifs financiers en utilisant un modèle LSTM (Long Short Term Memory). En parcourant ce processus, nous avons pu explorer différentes bibliothèques Python spécialisées dans le traitement de données, la manipulation de dates, la récupération d'informations financières en ligne et bien sûr, le machine learning.

L'application développée offre une interface conviviale qui permet aux utilisateurs de saisir un symbole d'actif financier et de choisir une plage de dates. Le programme récupère alors les données financières pertinentes et entraîne un modèle LSTM pour prédire les prix de clôture futurs. Les résultats sont ensuite affichés sous forme graphique, avec la tendance des prix réels et la prédiction du modèle.

Le modèle LSTM, avec sa capacité à retenir des informations à long terme, s'est révélé être un choix approprié pour ce type de tâche de prédiction. Cependant, comme tout modèle de machine learning, les prédictions du LSTM ne sont pas parfaites et il est important de prendre en compte d'autres facteurs lors de la prise de décisions d'investissement.

Au-delà de la précision de la prédiction, ce projet a permis de comprendre les subtilités de la préparation des données, de l'échelonnement des valeurs, de l'architecture du modèle LSTM et de l'importance de l'évaluation des performances.

Pour les futurs travaux, l'ajout de fonctionnalités supplémentaires est envisagé. Par exemple, le modèle pourrait être amélioré en intégrant d'autres facteurs, tels que des données économiques (ce qui serait dur mais je pense faisable) ou des indicateurs techniques. De plus, une analyse plus détaillée de la performance du modèle et de la significativité statistique des prédictions pourrait être effectuée.

En conclusion, bien que la prédiction des prix des actifs financiers soit une tâche complexe et que le modèle ne produise pas de prédictions parfaites, ce projet a fourni un excellent moyen d'appliquer et de comprendre les principes des réseaux de neurones LSTM et de voir comment ils peuvent être utilisés dans un contexte financier.