École Centrale Lyon

PE12
Rapport Final

PE12 : Prédiction du cours boursier par une intelligence artificielle combinant indicateurs financiers et analyse des sentiments

Élèves :
Jérémie Hareau
Raphael Davin
Alceu Diniz
Nada Braham
Amaury Truchetet
Thomas Henriot

Tuteur scientifique :
Christian de Peretti

Conseiller :
Denis Bouillet

Président de jury :
Stéphane Dumarty

Date d'écriture :
mercredi 5 juin 2024

Résumé

Le rêve de tout investisseur financier est de prévoir avec certitude le cours du lendemain d'une action boursière. Cependant la tâche s'avère extrêmement difficile, notamment à cause de l'irrationalité de l'être humain.

Mais avec l'essor de l'intelligence artificiel, l'impossible ne pourrait-il pas devenir réalisable ? En effet, en créant un algorithme combinant respectivement les aspects quantitatifs des données historiques financières avec les éléments qualitatifs des émotions humaines, tous les aspects nécessaires à la prévision du cours d'une action- seraient regroupés.

L'algorithme de prédiction que nous avons créé est constitué de deux intelligences artificielles : une traitant de l'analyse des sentiments et l'autre s'occupant de la prédiction à partir d'indicateurs financiers. Nous sommes capable de prévoir une baisse, une hausse ou une stagnation du cours d'une action avec une précision de 58%.

Abstract

The dream of every financial investor is to predict with certainty the next day's stock price. However, this task is extremely challenging, particularly due to the irrationality of human behavior.

But with the rise of artificial intelligence, could the impossible become achievable ? Indeed, by creating an algorithm that combines the quantitative aspects of historical financial data with the qualitative elements of human emotions, all the necessary aspects for predicting a stock's price would be integrated.

The prediction algorithm we have developed consists of two artificial intelligences : one handling sentiment analysis and the other focused on prediction based on financial indicators. We are capable of forecasting a decrease, increase, or stagnation in a stock's price with an accuracy of 58%.

Remerciements

Nous tenons à exprimer notre profonde gratitude à Monsieur de Peretti et à Monsieur Bouillet pour leur soutien tout au long de l'année. En effet, lors des soutenances et réunion, notre travail a été mis en avant et valorisé, ce qui nous a permis de persévérer même dans les moments les plus difficiles du projet.

Nous adressons un remerciement particulier à Monsieur de Peretti pour ses cours de finances en début d'année et pour les nombreuses réunions en méthode agile tout au long de l'année. Ces rencontres régulières nous ont permis d'obtenir un feedback constant !

Enfin, nous remercions chaleureusement Monsieur Bouillet pour le partage de son expérience précieuse. Ses nombreuses techniques nous ont été extrêmement utiles pour mener à bien notre projet.

Table des matières

Table des figures

1 Introduction

Dans le domaine financier, l'idéal serait d'acheter lorsque le cours d'une action est au plus bas pour revendre lorsqu'il est au plus haut. Cette stratégie, bien que simple à comprendre, s'avère extrêmement difficile à mettre en œuvre. En effet, la nature humaine étant souvent imprévisible et irrationnelle, anticiper les comportements des acteurs du marché relève d'un défi complexe.

Alors que la capacité de prévision semblait avoir été reléguée au second plan dans l'esprit de nombreux investisseurs, l'avènement de l'intelligence artificielle a ravivé l'intérêt pour cette notion. Nous aspirons ainsi à développer un algorithme de prédiction, alliant des aspects mathématiques et une analyse des sentiments de marché, dans le but de prédire le cours d'une action.

L'algorithme envisagé fait intervenir deux intelligences artificielles différentes : une traitant de l'analyse des sentiments et l'autre s'occupant de la prédiction à partir d'indicateurs financiers.

Pour être plus précis, la première intelligence artificielle sera conçue pour évaluer le sentiment actuel du marché en analysant des articles financiers et des données issues des réseaux sociaux. La seconde intelligence artificielle, quant à elle, se concentrera sur la prédiction des mouvements futurs du marché en se basant exclusivement dans un premier temps sur des indicateurs financiers. L'algorithme final intégrera les résultats de la première IA, en utilisant le sentiment du marché comme une nouvelle entrée pour la seconde IA en plus des indicateurs financiers déjà utilisés. Cette approche combinée vise à améliorer la précision des prédictions en intégrant l'analyse des sentiments actuels avec les données financières historiques (confère figure 13).

Pour illustrer son fonctionnement, prenons l'exemple de la prédiction du cours de l'action Google pour demain.

Dans un premier temps, la première intelligence artificielle sera alimentée avec une variété d'articles financiers et de messages provenant des réseaux sociaux. L'objectif est d'extraire le sentiment global de la journée à partir de ces sources diverses. Une fois ce sentiment extrait, il est ensuite pris en considération par notre seconde intelligence artificielle. Cette dernière utilise également une série d'indicateurs financiers spécifiques liés à l'action Google. En combinant ces données, la seconde intelligence artificielle est en mesure de générer ses propres prédictions quant au cours de l'action pour le lendemain. Ces prédictions peuvent indiquer une baisse, une stagnation ou une hausse du cours de l'action.

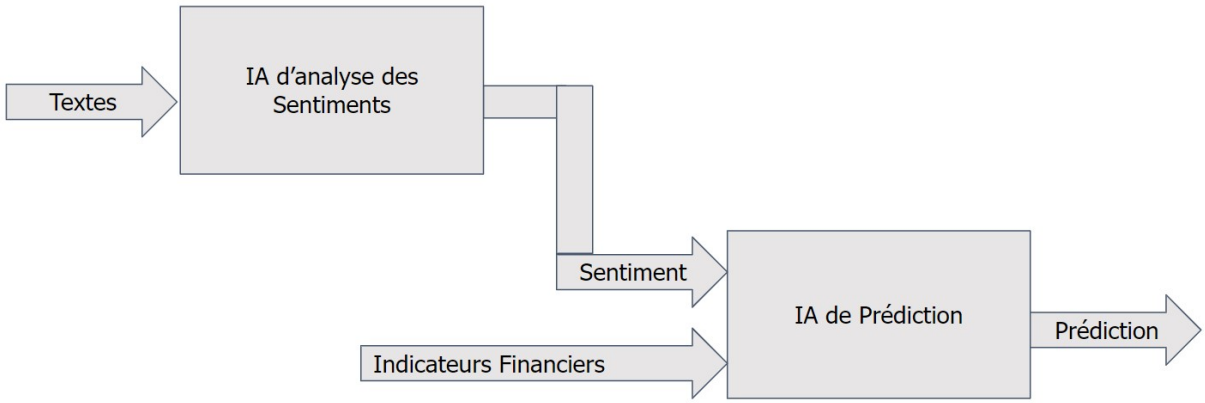


FIGURE 1 – Schéma de synthèse de notre algorithme

L'algorithme devra être en capacité de prédire correctement l'évolution du cours d'une action : hausse, baisse ou stagnation avec un taux de réussite supérieur ou égale à 60%.

Le rapport s'articulera comme suit : tout d'abord nous énumérerons les différents objectifs et tâches sélectionnés en début d'année pour la réalisation de notre projet. Ensuite, un état de l'art présentera toutes les connaissances essentielles. Puis nous détaillerons la méthodologie employée pour créer l'algorithme, et enfin nous présenterons les résultats obtenus.

2 Cahier des charges fonctionnels

Chacun des 3 sous-objectif peut se décomposer en plusieurs tâches, comme présenté ci-dessous :

2.1 Objectif 1 : Création d' une intelligence artificielle d'analyse des sentiments

- **Tâche 1 :** Déterminer les sources à analyser
- **Tâche 2 :** Déterminer les sentiments à analyser et la pondération associée
- **Tâche 3 :** Créer un algorithme de récupération quotidienne des données extraites d'un site
- **Tâche 4 :** Créer une base de données pour entraîner notre intelligence artificielle
- **Tâche 5 :** Développer une intelligence artificielle d'analyse des sentiments

2.2 Objectif 2 : Création d'une intelligence artificielle de prédiction

- **Tâche 1 :** Récupération des données historiques
- **Tâche 2 :** Choix des indicateurs en entrée
- **Tâche 3 :** Création de notre intelligence artificielle

2.3 Objectif 3 : Création de notre algorithme de prédiction final regroupant les deux intelligences artificielles précédentes

— **Tâche 1 :** Créer une nouvelle base de données
— **Tâche 2 :** Optimiser notre algorithme de prédiction en intégrant l'analyse des sentiments obtenue grâce à notre IA

3 État de l'art synthétique

3.1 Objectif 1 : Création d' une intelligence artificielle d'analyse des sentiments

Tâche 1 : Déterminer les sources à analyser

Pour tirer profit des inefficiences du marché, il est possible de prendre en compte le facteur psychologique et les biais comportementaux des différents acteurs financiers afin de faire des prédictions. Ainsi pour optimiser une gestion de portefeuille, une idée pourrait être de chercher à dégager l'humeur générale des investisseurs via des articles financiers et les réseaux sociaux. En effet plusieurs études ont montré qu'il existait des relations entre les nouvelles financières récentes et l'évolution du cours de la bourse[1].

Pour identifier le sentiment général du marché, il existe une discipline informatique, nommée "opinion mining", qui consiste à extraire grâce à des algorithmes le sentiment général d'un texte. Ces outils d'opinion mining ne sont efficaces que s'ils sont utilisés sur des sources textuelles pertinentes. On distingue deux grands types de sources : les réseaux sociaux, qui donnent directement accès à ce que pensent les acteurs, et la presse, qui n'est pas une expression directe de l'opinion des investisseurs, mais donne une bonne indication de la température globale du marché. Différentes études cherchent à exploiter ces deux types de sources pour appliquer des algorithmes d'opinion mining : beaucoup ont utilisé la presse spécialisée dans la finance (PRNewsWire...) ou même des réseaux sociaux financiers (le plus connu étant StockTwits)[2]. Les résultats obtenus sont assez variable selon les sources et les méthodes utilisées.

Tâche 2 : Déterminer les sentiments à analyser et la pondération associée

Si certaines études ne parviennent pas à obtenir de résultats significatifs permettant de démontrer un lien entre le sentiment du marché et les variations des indices boursiers, d'autres peuvent nous fournir plusieurs enseignements utiles. Par exemple, un aticle publié en 2021 montre qu'en faisant la distinction entre plusieurs émotions (colère, peur, tristesse, joie) et en les étudiant et notant séparément, on pouvait établir des corrélations entre la variation du prix des actions et les différentes notes de ces sentiments[3].

Mais l'article le plus édifiant est celui de Thomas Renaut[4] : en étudiant le sentiment global sur StockTwits (quantifié par un score de sentiment allant de -1 à 1, et reflétant la positivité du tweet étudié), il est parvenu à établir un lien significatif entre l'indicateur de sentiment et l'évolution des rendements.

Tâche 3 : Créer un algorithme de récupération quotidienne des données extraites d'un site

Le web scraping consiste à extraire de façon automatisée, et grâce à un langage de programmation [5], des données textuelles provenant de diverses sources comme des sites web ou des réseaux sociaux. Un moyen élémentaire de faire du web scraping est de travailler directement sur le code html de la page que l'on veut "scrapper", car ce dernier est facilement accessible et tout le contenu à extraire (c'est à dire le texte de la page) y est

repérable grâce aux balises.

La bibliothèque python Beautiful soup est intéressante pour effectuer du web scrapping car elle est efficace et simple à prendre main. Cependant, elle ne fonctionne que pour les pages web statiques, c'est à dire dont le code html est généré dans son intégralité à l'ouverture de la page. Pour les pages dynamiques, comme c'est le cas pour les réseaux sociaux, l'utilisation en plus de Beautiful Soup de la bibliothèque Selenium est nécessaire pour interagir avec la page et avoir accès à tout le code html.

Tâche 4 : Créer une base de données pour entraîner notre intelligence artificielle

Les bases de données sont essentielles pour l'intelligence artificielle car elles permettent le stockage et l'organisation de vastes quantités de données nécessaires à l'entraînement des modèles. Ces bases de données se doivent d'être les plus exhaustives possible afin d'avoir un maximum d'exemples pour que l'IA comprenne au mieux la complexité de ce qui lui est demandé.

Pour créer une base de données, plusieurs options s'offrent à nous : créer la base de données manuellement, ce qui permet de la personnaliser exactement selon nos besoins mais est extrêmement chronophage ; utiliser des bases de données existantes, à condition qu'elles correspondent bien à nos critères pour les données d'entrée et les types de sortie ; ou utiliser des IA préexistantes ayant les mêmes types de données d'entrée et de sortie que notre projet, ce qui est la méthode la plus simple à mettre en œuvre. Cette dernière approche consiste à utiliser ces IA pour générer les données nécessaires en leur fournissant nos entrées spécifiques et en récupérant leurs sorties pour constituer notre base de données d'entraînement. Attention cette méthode bien que simplé nécessite l'utilisation de plusieurs IA pour éviter de reproduire simplement l'IA d'origine voir pire, d'obtenir une version moins performante.

Tâche 5 : Développer une intelligence artificielle d'analyse des sentiments

Une intelligence artificielle est un algorithme créé dans le but d'imiter l'intelligence humaine. Il existe de nombreux types d'IA, et le choix de l'IA à utiliser est basé sur la tâche qu'il lui est demandé d'exécuter : le principe de cette tâche (classification ou régression) et sa complexité.

L'analyse de sentiment est une tâche complexe pour un programme. Un algorithme «simple» n'est pas en mesure de pouvoir comprendre un texte écrit dans une langue aussi «complexe» que le français. Pour réaliser une tâche de régression complexe (une régression consiste à la prédiction d'une valeur, ici le sentiment du texte), les réseaux de neurones semblent être des modèles d'IA adaptés. Les réseaux de neurones sont des modèles mathématiques avec des milliers de paramètres. Ils sont une solution à la recherche d'un modèle trop complexes pour être calculés. Les paramètres du réseaux, se séparent en deux groupes : les paramètres architecturaux, qui sont immuables et choisi par le créateur du réseau, et les paramètres scalaires, les valeurs du réseau.

Pour simplifier, le réseau peut se symboliser par l'équation $Y = f(X, s)$ où X est le texte donné en entrée, Y le sentiment du texte donné par le réseau au texte X, f est la fonction liée aux paramètres architecturaux du réseau, et s contient l'ensemble des paramètres scalaires. Pour obtenir la réponse la plus satisfaisante possible, il convient donc d'avoir les paramètres architecturaux et scalaires les mieux adaptés et les plus précis possibles.

Il existe au sein des réseaux de neurones de nombreuses familles. Pour effectuer de l'analyse de texte, une famille semble sortir du lot : les réseaux dits «récurrents»[6]. La spécificité de ces réseaux est de découper les données en entrée pour les traiter en plusieurs fois, tout en gardant en mémoire la partie déjà traitée précédemment. Une mise en équation de ce principe serait en gardant les notations précédentes, et en notant $X = X_1 + X_2 + X_3 \ldots + X_n$ et $Y_{i+1} = f(X_{i+1}, s, Y_i)$, où la réponse est $Y = Y_n$ (il existe également d'autres manières de déterminer Y comme : somme, moyenne ou encore maximum des Y_i obtenus). L'architecture des réseaux récurrents est un atout dans le traitement de données séquentielles, où les différents éléments de l'entrée sont liés entre eux, comme les mots d'un texte.

Parmi les réseaux récurrents il existe deux types de cellules que sont les cellules LSTM et GRU[6]. Leur but est de palier à l'un des principaux problèmes des réseaux récurrents, qui est la disparition du gradient lors de l'apprentissage (concrètement, un réseaux récurrents a des problèmes d'apprentissages sur des entrées de taille conséquente.

Les cellules LSTM sont un type de cellule pour les réseaux récurrents. Les cellules LSTM se représentent ainsi :

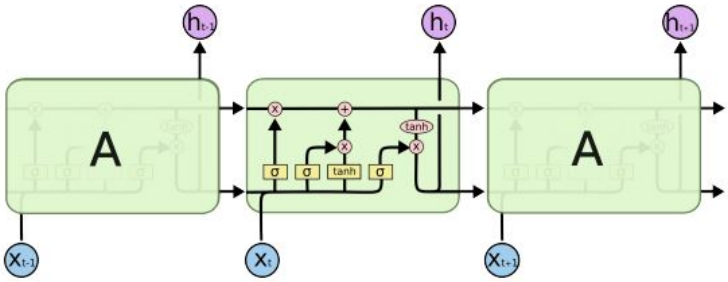


FIGURE 2 – Schéma d'une cellule LSTM

Les cellules LSTM sont une forme particulière de cellules récurrentes. Elles se distinguent par leur capacité de "mémorisation" : elles peuvent décider de ce qui est important où non à garder en mémoire (dans un vecteur "état" noté c). Ce choix se fait notamment par la porte d'oubli, ou "forget gate", qui multiplie les éléments inutiles par une valeur proche de 0, et les éléments utiles par une valeur proche de 1. Les cellules LSTM sont formées de 4 portes, qui possèdent chacune leurs poids de propagation (matrices notées avec W) et de récurrence (matrices notées U) ainsi que leurs biais (vecteurs notés b).[6]

$$
\begin{aligned}
F_t &= \sigma(W_F x_t + U_F h_{t-1} + b_F) \\
I_t &= \sigma(W_I x_t + U_I h_{t-1} + b_I) \\
O_t &= \sigma(W_O x_t + U_O h_{t-1} + b_O) \\
c_t &= F_t \circ c_{t-1} + I_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= O_t \circ \tanh(c_t) \\
o_t &= f(W_o h_t + b_o)
\end{aligned}
$$

FIGURE 3 – Calculs effectués par une cellule LSTM

Ici, σ correspond à la fonction sigmoïde, h_t la réponse au temps t, et $\odot$ à la multiplication matricielle terme à terme. Ce système revient à dire que la cellule aura modifié son état c_t et aura renvoyé pour une entrée x_t une sortie h_t .

Un réseau récurrent dit bidimensionnel [7] a la particularité de traiter l'entrée dans les deux sens. Un exemple classique de l'efficacité de ce principe est le suivant.

Complétez les textes :

1. un, deux, ..., quatre, cinq
2. un, deux, ..., huit, seize

Comme illustré par cet exemple, lire un texte uniquement de droite à gauche peut être source d'erreur. En effet, dans les deux cas nous serions tentés de répondre trois mais en lisant de droite à gauche la deuxième phrase on prend conscience que la réponse attendue est quatre et non deux. Ainsi, lire le texte dans les sens permet au réseau une meilleure compréhension du texte en question.

Pour réunir les deux lectures de l'entrée, une cellule en fin du réseau aura la responsabilité de les pondérer et de les additionner pour obtenir le sentiment final.

Le dernier élément pour définir les bases de notre réseau est la façon dont celui-ci va apprendre. Comme écrit dans la partie précédente, le principe d'apprentissage du réseau sera l'apprentissage supervisé, c'est à dire qu'il sera donné au réseau de grandes quantités d'exemples, chacun pourvu de la réponse attendue (ils sont dits étiquetés). Par ces exemples, le réseau ajustera ses paramètres scalaires pour approcher le plus possible des étiquettes. L'ajustement des paramètres se fait par la méthode de la descente de gradient.[8]

Pour que le réseau donne des résultats satisfaisants, il est nécessaire de l'entraîner. Le principe de l'apprentissage par rétropropagation du gradient est intuitif : les paramètres scalaire du réseau sont initialisés au hasard, puis le réseau traite de nombreux exemples de textes. Pour chaque texte, il calcule l'influence de chacun de ses paramètre sur son erreur (la différence entre la réponse attendue et la réponse qu'il a donné). Puis, il modifie ses paramètres en fonction de cette influence. Cette modification se fait en fonction d'un pas,

qui doit être choisi assez grand pour que l'apprentissage ne demande pas trop d'exemples, et assez faible pour que le modèle que représente le réseau puisse converger vers un modèle convenable. L'équation représentant la rétropropagation du gradient est la suivante. Mathématiquement, la rétropropagation se formule ainsi [8].

$$w_{ij}^{(n)} = w_{ij}^{(n)} - \lambda e_i^{(n)} x_j^{(n-1)}$$

FIGURE 4 – Calcul des gradients

Pour initier les poids des cellules, il existe la méthode Xavier/Glorot, qui repose sur le calcul des valeurs aléatoires suivant une distribution uniforme de probabilité situées entre -1/n et 1/n, où n est le nombre d'entrées. Cette méthode de répartitions aléatoire des poids permet une convergence plus rapide pour l'apprentissage par rétropropagation, adaptée à l'utilisation des fonctions sigmoïde et tangente hyperbolique, qui sont les fonctions utilisées par la cellule LSTM[9].

Le calcul de l'influence de chacun de ses paramètre sur l'erreur se fait par des calculs de gradients via la règle de la chaîne. Pour ce réseau, l'erreur sera la différence entre la réponse attendue et la réponse du réseau, le tout au carré. On peut ensuite calculer les gradients via ces équations.

- $dE/dw_{xo} = dE/d_o * (d_o/dw_{xo}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)) * x_t$
- $dE/dw_{ho} = dE/do * (do/dw_{ho}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)) * h_{t-1}$
- $dE/db_o = dE/do * (do/db_o) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)$
- $dE/dw_{xf} = dE/df * (df/dw_{xf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * x_t$
- $dE/dw_{hf} = dE/df * (df/dw_{hf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * h_{t-1}$
- $dE/db_o = dE/df * (df/db_o) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)$
- $dE/dw_{xi} = dE/di * (di/dw_{xi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * x_t$
- $dE/dw_{hi} = dE/di * (di/dw_{hi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * h_{t-1}$
- $dE/db_i = dE/di * (di/db_i) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)$
- $dE/dw_{xg} = dE/dg * (dg/dw_{xg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * x_t$
- $dE/dw_{hg} = dE/dg * (dg/dw_{hg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * h_{t-1}$
- $dE/db_g = dE/dg * (dg/db_g) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g))$

FIGURE 5 – Calcul des gradients

Un pré-traitement des entrées doit aussi être effectué. En effet, même si cette partie n'est pas directement liée au réseau, elle reste essentielle pour résoudre la plupart des problèmes de machine learning. Un réseau de neurones est un modèle mathématique, qui s'applique sur des entrées qui sont des vecteurs de nombres. Or sachant que l'entrée est un texte, il faut une étape transformant les textes en vecteurs. Le pré-traitement de texte se divise ainsi :

D'abord, le code est "nettoyé". Certains mots jugés peu utiles -que l'on appelle des "stopwords" - ainsi que certaines ponctuations sont enlevés. En effet,e tout ce qui peut être considéré comme inutile pour de l'analyse de sentiment, ou alors particulièrement dur

à traiter pour le réseau est supprimé.

Ensuite, les phrases nettoyées sont "tokenisées", c'est à dire fragmentées, souvent selon des mots ou sous-mots. Par exemple, "Bonjour, comment allez-vous" pourrait devenir "Bonjour comment allez vous" après nettoyage, puis ["Bonjour","comment","allez","vous"] voir "Bon","jour","comment","allez","vous"] après tokenisation, suivant si celle-ci est effectuée selon les mots ou les sous-mots.

Une fois le texte nettoyé et tokenisé, il est vectorisé, c'est l'étape de "l'embedding". L'embedding consiste à représenter chaque token par un vecteur de nombres réels, chaque nombre représentant une caractéristique du token ainsi vectorisé. Pour cette étape, il est intéressant de faire appel à une aide externe. Les étapes de tokenisation et l'embedding sont effectuées à l'aide du module SpaCy.

3.2 Objectif 2 : Création d'une intelligence artificielle de prédiction

Tâche 2 : Choix des indicateurs en entrée

En finance, les traders utilisent des indicateurs financiers pour tenter de prédire les variations du cours d'une action.[10] Voici, certains des plus utilisés :

Volatilité : La volatilité est une mesure de la variation du prix d'un actif financier sur une période donnée. Elle indique l'amplitude et la fréquence des fluctuations de prix. Les investisseurs utilisent la volatilité pour évaluer le niveau de risque associé à un actif. Une volatilité élevée signifie que le prix peut varier considérablement sur une courte période, tandis qu'une faible volatilité indique des variations de prix plus stables.

SMA (Simple Moving Average) : La moyenne mobile simple est une méthode d'analyse technique qui calcule la moyenne des prix d'un actif sur une période spécifique en ajoutant les prix de clôture pour chaque période et en divisant par le nombre total de périodes. Les SMA sont utilisées pour lisser les données de prix et identifier les tendances à moyen et long terme. Elles sont également utilisées comme signaux d'achat ou de vente lorsqu'elles croisent le prix actuel.

RSI (Relative Strength Index) : L'indice de force relative est un oscillateur qui mesure la force et la vitesse des variations de prix d'un actif financier en comparant les gains et les pertes moyens sur une période spécifique. Le RSI varie de 0 à 100 et est généralement utilisé pour identifier les conditions de surachat et de survente sur un marché. Un RSI supérieur à 70 est considéré comme indiquant une surachat, tandis qu'un RSI inférieur à 30 indique une survente.

Momentum : Le momentum est une mesure de la vitesse à laquelle le prix d'un actif change sur une période donnée. Il est calculé en soustrayant le prix de clôture actuel du prix d'une période précédente. Un momentum positif indique une tendance à la hausse, tandis qu'un momentum négatif indique une tendance à la baisse. Les investisseurs utilisent le momentum pour confirmer les tendances de marché et identifier les points d'entrée

et de sortie potentiels.

TRIX (Triple ExponentIAl Moving Average) : Le TRIX est un indicateur de momentum qui lisse les données de prix en utilisant une moyenne mobile exponentielle triple. Il mesure la vitesse à laquelle le prix d'un actif change sur une période donnée, similaire au momentum. Cependant, le TRIX est plus sensible aux changements de tendance à court terme en raison de sa méthode de lissage plus complexe. Les croisements de la ligne TRIX avec sa ligne de signal sont souvent utilisés comme signaux d'achat ou de vente.

Tâche 3 : Réseaux de Neurones Profond

Les réseaux de neurones profonds (DNN) représentent une classe puissante de modèles d'apprentissage automatique qui ont révolutionné de nombreux domaines, de la reconnaissance d'images à la traduction automatique en passant par les véhicules autonomes. Leur popularité découle de leur capacité à capturer des modèles complexes dans des données non structurées, permettant ainsi des performances remarquables sur une gamme de tâches.[11]

Au cœur des DNN se trouvent les neurones artificiels, des unités computationnelles qui agissent comme des approximations des neurones biologiques. Ces neurones prennent des entrées pondérées, les combinent et les passent à travers une fonction d'activation pour produire une sortie. L'apprentissage dans un DNN implique l'ajustement de ces poids et biais pour minimiser une fonction de perte, généralement par des algorithmes tels que la descente de gradient.

Un aspect clé des DNN est leur capacité à modéliser des représentations hiérarchiques des données. Avec plusieurs couches cachées, ces réseaux peuvent apprendre des caractéristiques de plus en plus abstraites à mesure que les données passent à travers le réseau. Cette hiérarchie de représentations permet aux DNN de capturer des caractéristiques complexes et des relations non linéaires dans les données, les rendant particulièrement efficaces pour des tâches telles que la reconnaissance d'images et le traitement du langage naturel.

Cependant, l'efficacité des DNN dépend fortement du choix judicieux des hyperparamètres. Ces paramètres, tels que le taux d'apprentissage, le nombre de couches et les fonctions d'activation, ne sont pas appris directement à partir des données, mais doivent être spécifiés avant l'entraînement. Trouver les valeurs optimales des hyperparamètres est souvent une tâche ardue et nécessite une expérimentation minutieuse. [12]

Un défi majeur lors de l'entraînement des DNN est l'overfitting, où le modèle s'adapte trop bien aux données d'entraînement et ne généralise pas bien aux nouvelles données. Pour lutter contre l'overfitting, diverses techniques de régularisation sont utilisées, telles que L1 et L2, qui ajoutent des termes de pénalité aux poids du modèle, ainsi que le dropout, qui supprime aléatoirement des neurones pendant l'entraînement.

Enfin, l'évaluation des performances des DNN nécessite une attention particulière à la manière dont les données sont divisées en ensembles d'entraînement, de validation et de

test. Une division inappropriée peut conduire à des estimations biaisées des performances du modèle.

Pour expliquer les différences entre le jeu d'entrainement, de validation et de test une analogie peut être faite avec l'école. L'ensemble d'entraînement est comme les cours et exercices qu'un élève utilise pour apprendre. Pendant cette phase, l'IA apprend à partir des données fournies. Ensuite, l'ensemble de validation est utilisé de manière similaire aux quiz intermédiaires, permettant de vérifier si l'IA comprend bien les concepts ou si elle commence à mémoriser les réponses spécifiques, ce qui peut indiquer un surapprentissage (overfitting). Enfin, l'ensemble de test est comme l'examen final, où l'IA est évaluée sur des données inédites pour mesurer ses performances réelles et vérifier son efficacité dans des conditions nouvelles.

L'objectif principal de l'entraînement est d'obtenir la plus grande précision possible tout en réduisant la fonction coût. La fonction coût, ou fonction de perte, mesure l'erreur entre les prédictions du modèle et les résultats réels. En réduisant cette fonction, on améliore la performance du modèle.

La précision est définie comme le nombre de prédictions correctes (vraies positives) parmi toutes les prédictions positives faites par le modèle. En d'autres termes, elle mesure la capacité du modèle à ne pas faire de fausses prédictions positives. Elle est définie par la formule [13] :

$$\text{Précision} = \frac{\text{Vraies Positives}}{\text{Vraies Positives} + \text{Faux Positifs}} \tag{1}$$

avec

1. Vraies Positives : Nombre d'échantillons correctement prédits comme positifs
2. Faux Positifs : Nombre d'échantillons incorrectement prédits comme positifs

Pendant l'entraînement, l'algorithme ajuste ses paramètres pour minimiser cette erreur. Cela se fait en utilisant des techniques d'optimisation, comme la descente de gradient, qui ajustent les poids du modèle de manière itérative. Chaque itération de ce processus est appelée une époque (epoch). Une époque correspond à un passage complet à travers l'ensemble d'entraînement.

En pratique, au lieu de passer par toutes les données d'entraînement en une seule fois, l'ensemble d'entraînement est souvent divisé en lots plus petits appelés "batches". Chaque batch est utilisé pour une mise à jour partielle des paramètres du modèle. Cette méthode, appelée descente de gradient stochastique par mini-batch (mini-batch stochastic gradient descent), permet une convergence plus rapide et une utilisation plus efficace de la mémoire.

Pour coder et créer son propre réseau de neurones, il est courant d'utiliser Python, notamment avec la bibliothèque TensorFlow. Cette dernière permet de construire efficacement et rapidement une architecture d'IA performante.[14]

En conclusion, les réseaux de neurones profonds représentent une classe de modèles d'apprentissage automatique extrêmement puissante, mais leur efficacité dépend de nombreux facteurs, notamment la sélection appropriée des hyperparamètres, la régularisation

efficace et le choix d'algorithmes d'optimisation appropriés. Avec une attention méticuleuse à ces aspects, les DNN continueront à repousser les limites de ce qui est possible dans de nombreux domaines d'application.

Cependant, d'autres options peuvent être utilisés autre que coder un Réseaux de Neurones sur Tensorflow. En effet, il est aussi possible d'utiliser la bibliothéque sklearn et d'utliser l'algorithme de classification Random Forest. Random Forest peut être comparé métaphoriquement à une équipe de décideurs. Plutôt que de dépendre d'une seule personne pour prendre une décision, il demande l'avis de plusieurs experts (arbres de décision) dans la forêt. Chacun donne son opinion, et le Random Forest combine toutes ces opinions pour prendre la meilleure décision. Cela rend le modèle robuste et moins susceptible de se tromper sur de nouvelles situations, ce qui en fait un choix puissant pour la classification dans l'apprentissage machine.[15]

Afin d'améliorer la precision du modéle, on va utiliser une méthode du Backtesting.Le Backtesting est une méthode utilisée pour tester une stratégie de trading en appliquant cette stratégie à des données historiques. En utilisant des données passées, on peut simuler des transactions pour voir comment la stratégie aurait performé dans le passé. Cela permet d'évaluer l'efficacité de la stratégie sans risquer de l'argent réel.[16]

4 Méthodologie

4.1 Objectif 1 : Création d' une intelligence artificielle d'analyse des sentiments

L'objectif de cette partie est de développer une IA d'analyse des sentiments, dont les résultats serviront d'entrées pour une IA de prédiction. Cela nécessite de nombreuses réflexions et tâches à accomplir :

1. Définir les entrées de l'IA : textes, articles, mais quelles sources spécifiques et performantes choisir ?
2. Déterminer comment collecter ces textes ?
3. Spécifier la sortie de l'IA : un sentiment parmi plusieurs ? Une note pour chaque sentiment ? Une note générale ?
4. Comment créer une base de données pour entraîner l'IA ?
5. Quelle architecture d'IA choisir ?

En nous basant sur notre état de l'art, nous avons estimé qu'il serait plus simple de réaliser nos prédictions sur les indices américains, étant donné que le marché américain est très liquide et représente bien l'économie du pays. Par conséquent, les sources utilisées pour appliquer nos algorithmes de mining d'opinions seront américaines.

Nous avons décidé d'utiliser à la fois les médias traditionnels et les réseaux sociaux. Les médias traditionnels même s'ils ne reflètent pas directement le sentiment des investisseurs offrent une vue d'ensemble de l'état du marché. Les réseaux sociaux, quant à eux, permettent d'accéder directement aux opinions des acteurs du marché financier ainsi qu'à celles du grand public.

Pour les réseaux sociaux, nous allons utiliser StockTwits, qui est un réseau social financier, et Twitter. Même si Twitter n'est pas spécialisé dans la finance, une sélection adéquate des tweets utilisés peut nous permettre d'obtenir des informations pertinentes (par exemple en ne prenant en compte que les tweets contenant des mots-clés tels que "I feel" ou "mood"). L'avantage de Twitter est qu'il soit utilisé par une très grande partie de la population ce qui offre un volume de publications extrêmement conséquent et permet d'avoir une idée plus représentative du sentiment du marché.

Pour les médias traditionnels, nous avons sélectionné 2 sources d'informations en continu spécialisées dans la finance : PRNewsWire et YahooFinance. Toutes ces sources sont en anglais.

A partir de ces sources, le but de notre intelligence artificielle sera d'extraire une valeur entre -1 et 1 qui sera à l'image du sentiment général du marché (cette idée nous vient directement des travaux de Thomas Renault, cf état de l'art). Cette note globale sera la moyenne de toutes les notes données aux articles et aux tweets. En effet, nous avons pensé qu'il était plus simple, et pas forcément réducteur, de faire un indicateur général traduisant la positivité ou non du sentiment du marché plutôt que de faire des distinctions

entre plusieurs sentiments spécifiques.

Les sources à analyser étant déterminées, il s'agit maintenant de les récupérer et ce de manière automatisée. En effet, pour prédire le cours de l'action à la journée t+1 il nous faut analyser les sources au jour t pour en extraire le sentiment et donc les récupérer quotidiennement.

Ainsi, pour récupérer de façon automatisée du contenu textuel sur les réseaux sociaux ou les sites web, nous avons utilisé la bibliothèque python BeautifulSoup (qui nous permet d'effectuer des traitements sur le code html de la page) ainsi que Selenium afin de pouvoir interagir avec les pages (ce qui est nécessaire étant donné le caractère dynamique des pages web StockTwits et Twitter).

StockTwits

Pour StockTwits, la tâche était relativement simple. Après avoir ouvert la page d'accueil avec la bibliothèque Selenium, il fallait faire défiler la page pour charger tout le code HTML complet (et non juste les premières lignes), ce qui a été rendu possible grâce à Selenium.

Ensuite, nous avons extrait le code HTML complet de la page, puis utilisé BeautifulSoup pour filtrer ce code et récupérer uniquement le contenu textuel - les tweets- présent sur la page (environ une centaine). Ce traitement nous a permis d'obtenir une liste où chaque élément correspondait au contenu d'un tweet. Il ne restait plus qu'à écrire le contenu de cette liste dans un fichier Excel pour obtenir le résultat voulu.

Twitter

Pour obtenir le même résultat sur Twitter, la tâche a été un peu plus difficile car Twitter demande une authentification pour accéder au contenu. Nous avons donc dû automatiser l'étape de connexion à l'aide de Selenium, qui permet d'interagir avec la page. Pour ce faire, nous avons stocké les informations de connexion (l'adresse mail, le nom d'utilisateur et le mot de passe) dans trois variables, puis mis en place un code permettant de fournir successivement ces informations à la page.

Une fois connecté, nous avons suivi les mêmes étapes que pour StockTwits afin d'obtenir un fichier Excel contenant toutes les publications de la page.

Articles

Pour les deux journaux considérés, PRNewsWire et Yahoo Finance, la procédure pour récupérer les articles consistait à se rendre sur la page d'accueil des sites et à récupérer tous les liens présents. Ensuite, nous avons filtré ces liens pour exclure les publicités, les vidéos et les articles déjà traités les jours précédents, afin de ne conserver que les nouveaux articles. Une fois tous les liens pertinents collectés, nous avons effectué des requêtes HTTP pour obtenir les pages complètes en format HTML. Cela nous a permis de naviguer sur chaque page et, en utilisant les identifiants spécifiques qui indiquent l'emplacement du corps du texte ou des annonces, d'extraire précisément le contenu désiré.

Pour automatiser la récupération quotidienne du contenu textuel publié, nous avons utilisé le gestionnaire des tâches de Windows. Nous avons programmé une tâche qui exécute nos algorithmes de web scraping à une heure donnée chaque jour, garantissant ainsi l'automatisation complète du processus.

Nous sommes donc en mesure de récupérer nos entrées, à savoir des articles et des tweets, et nous connaissons notre sortie : un sentiment compris entre -1 et 1. Ainsi, il ne manque plus que la base de données pour créer notre intelligence artificielle d'analyse des sentiments. Idéalement, nous souhaiterions trouver une base de données correspondant à ces critères. Cependant, cela n'étant pas possible, nous avons décidé de créer notre propre base de données. Pour ce faire, nous allons collecter de nombreux articles et tweets correspondant à nos types d'entrée, puis les faire analyser par différentes IA d'analyse de sentiments existantes afin de constituer notre propre base de données, même si nous aurions aimé noter nous-même chaque article manuellement pour nous garantir un résultat satisfaisant. Cependant, ne disposant pas d'une quantité de temps et d'une patience suffisante pour annoter des dizaines de milliers d'articles et tweets nous nous sommes contentés des IA déjà existantes.

Pour les articles financiers, nous avons récupéré de nombreux articles sur PRNewsWire et Yahoo Finance. Il nous restait alors à leur attribuer une note entre -1 et 1 en utilisant des intelligences artificielles existantes : NLTK, l'IA de Python, et Watson Natural Language d'IBM.

Nous avons principalement utilisé NLTK car il est directement intégré à Python et libre de droit, ce qui facilite son utilisation et permet d'analyser un grand nombre de textes. En revanche, Watson Natural Language, bien que performant, est limité par la version gratuite qui ne permet d'analyser que 15 000 textes par mois, rendant son utilisation limitée pour obtenir des données suffisamment exhaustives pour notre programme.

Pour ce qui est des tweets, nous nous sommes basés sur une base de données existante trouvée sur le site Kaggle [17], comprenant environ 25 000 tweets. Cette base de données attribuait à chaque tweet un sentiment (positif, neutre ou négatif) ainsi qu'une note entre 0 et 1. Nous avons donc dû convertir ces notes pour les situer entre -1 et 1, selon les critères suivants : entre -1 et $-\frac{1}{3}$ pour les sentiments négatifs, entre $-\frac{1}{3}$ et $\frac{1}{3}$ pour les sentiments neutres, et entre $\frac{1}{3}$ et 1 pour les sentiments positifs. 25000 tweets n'étant pas suffisants nous avons procédé de la même façon que pour les articles financiers pour renforcer la base de données avec des tweets provenant de Twitter et de StockTwits évalués par une IA.

Nous avons finalement obtenu une base de données constituée de 75 000 données, ce qui nous permettra d'entraîner notre intelligence artificielle d'analyse des sentiments. Idéalement, nous aurions souhaité disposer de davantage de données, car plus le volume de données est important, plus l'intelligence artificielle est performante.

Maintenant que les entrées et sorties ont été déterminés et que la base de données a été créée, il est temps de parler de l'IA en elle-même.

Au vu de l'état de l'art, nous avons choisi une architecture de réseaux récurrents uti-

lisant des cellules LSTM. En effet, les LSTM sont plus connues et plus anciennes que les cellules GRU, ce qui les rend plus simples à utiliser. De plus, notre réseau sera bidimensionnel et les poids seront initialisés grâce à la méthode Xavier/Glorot.

La première étape pour coder le réseau était l'implémentation des classes de base que sont les cellules, les couches, et la cellule finale, dont le rôle est de pondérer les 2 sorties du réseau bidimensionnel. Puis, la classe réseau fait le lien entre les différentes couches, et contient les fonctions principales : initialisation du réseau, propagation vers l'avant et vers l'arrière.

La classe réseau contient également les fonctions de chargement et de sauvegarde des paramètres du réseau, via le module numpy et les documents en ".npy". Les fichiers doivent être nommés de telle sorte à ce que le programme puisse charger les paramètres et les placer convenablement dans le réseau. La dernière fonction à implémenter concerne la lecture des textes à analyser, et leur pré-traitement. Les textes sont lus via des fichiers CSV à l'aide du module python panda, nettoyés puis vectorisé via le module SpaCy.

Une fois ces fonctions implémentées, le réseau doit être entraîné. Le module réseau est donc muni d'une fonction d'apprentissage, qui, pour un nombre d'itération n donné, effectue une propagation puis une rétropropagation du gradient sur le réseau, pour les n premières lignes du document d'entraînement. L'entraînement terminé, il supprime ces lignes (le CSV original étant déjà enregistré et intact, de sorte à pouvoir refaire l'entraînement), affiche les résultats liés à l'entraînement, et sauvegarde les nouveaux paramètres du réseau. Cette fonction d'apprentissage est munie de tests qui empêche le réseau de s'entraîner sur des textes qui amènerait un potentiel problème (notamment des lignes où l'étiquette est considéré comme NaN : Not a Number pour python, qui amènerait les paramètres du programme à être bloqué sur l'état NaN après rétropropagation du gradient).

4.2 Objectif 2 : Création d'une intelligence artificielle de prédiction

L'objectif était clair : développer une intelligence artificielle capable de prédire avec la plus grande précision le cours du lendemain de n'importe quelle action boursière. Avant de commencer le développement de l'IA, il était essentiel de définir les paramètres clés : les données d'entrée, les données de sortie et la base d'apprentissage. Ces éléments sont cruciaux pour assurer la précision et l'efficacité de l'IA dans ses prédictions.

Données de sortie

Nous avons rapidement écarté l'idée de prévoir précisément le prix du jour n+1 avec les données du jour n, car cela nous semblait extrêmement improbable d'obtenir un résultat satisfaisant. De plus, étant donné que la notion de rendement est plus couramment utilisée que celle de prix, nous avons décidé de nous concentrer sur la prévision d'une hausse ou d'une baisse du rendement. Toutefois, pour éviter de prendre en compte les légères fluctuations quotidiennes, nous avons ajouté l'option de stagnation. Ainsi, notre

programme se concentrera uniquement sur les grandes fluctuations. Notre IA doit donc produire une sortie de 1 lorsque le rendement futur prévu est supérieur au 80ème percentile (c'est-à-dire parmi 20 % les plus élevés des rendements passés observés), de -1 lorsque le rendement futur est inférieur au 20ème percentile, et de 0 entre les deux. Ainsi, notre IA va prédire si le rendement du jour n+1 va subir une hausse, une stagnation ou une baisse par rapport au rendement du jour n.

Données d'entrée

Nous aurions pu choisir d'utiliser uniquement des données brutes telles que les prix d'ouverture, de fermeture et le volume, en laissant l'IA découvrir les corrélations entre ces données et le rendement du lendemain. Cependant, en raison de la complexité limitée de notre IA, contrainte par la puissance de nos ordinateurs, nous avons préféré utiliser des indicateurs financiers couramment employés par les traders pour anticiper les mouvements d'un actif. Il n'y a en effet aucune utilité à gaspiller notre puissance de calcul en laissant l'IA chercher des moyens de prédiction alors que certains indicateurs ont été spécialement créés à cet effet. Ainsi, nous avons opté pour les indicateurs suivants : Volatilité, RSI, SMA, EMA, Momentum et TRIX.

Base de données

Maintenant que nous avons défini les données d'entrée et de sortie, il est nécessaire de créer une base de données sur laquelle notre IA pourra s'entraîner. Cette base de données doit contenir les indicateurs financiers sélectionnés ainsi que la réponse que notre IA doit fournir (-1, 0, 1).

En utilisant Yahoo Finance, il est possible de récupérer les prix d'ouverture, de fermeture et le volume de vente des actions depuis l'année 2000, ce qui fait plus de 24 ans de données. À partir de ces données, nous pouvons dériver toutes les informations souhaitées, car les indicateurs que nous avons sélectionnés se calculent à partir des données fournies par Yahoo Finance. En connaissant le prix de fermeture, nous pouvons également calculer les rendements futurs et ainsi déterminer les valeurs de sortie (-1, 0 ou 1) pour notre IA.

Une seule action n'étant pas suffisante pour créer une base de données robuste, nous avons décidé d'en récupérer beaucoup plus. Pour garantir une cohérence dans notre base de données, nous avons sélectionné les actions de manière systématique. Ainsi, nous avons choisi cinq actions pour chacun des domaines suivants : agro-alimentaire, assurance, divertissement, distribution, énergie, technologie, pharmaceutique, industriel, mode, sport, business et télécommunication (confère Annexe pour plus de détails).

Cette base de 275 551 données est ensuite séparée en 3 : une base d'entraînement (70% des données), une base de validation (20% des données) et une base de test contenant le reste des données.

Intelligence Artificielle

Pour élaborer notre IA, nous avions deux options :

1. Créer notre Intelligence artificielle en suivant des tutoriels en ligne. Cependant, bien que cette approche puisse garantir un résultat, la capacité à personnaliser le code initial pour qu'il corresponde exactement à nos attentes est limitée, ce qui réduit notre flexibilité.
2. Créer notre propre réseau de neurones profonds de A à Z en utilisant TensorFlow en Python. Cependant, cette méthode comporte le risque de ne pas garantir des résultats satisfaisants.

Par conséquent, nous avons décidé d'explorer les deux approches et de retenir l'algorithme final qui répondrait le mieux à nos besoins et exigences.

Intelligence Artificielle en suivant des tutoriels

Dans de nombreux jeux de données, certaines classes peuvent être naturellement beaucoup plus fréquentes que d'autres ce qui rend le modèle de machine learning biaisé en faveur de la classe majoritaire d'où l'importance de la méthode de ré-échantillonnage des classes. En effet, dans le domaine de l'intelligence artificielle une classe fait référence aux différentes catégories ou étiquettes dans lesquelles les données peuvent être regroupées. On sépare les données d'entraînement en 2 parties : les échantillons de la classe majoritaire (Target = 0) et les échantillons des classes minoritaires positives (Target = 1) et négatives (Target = -1). Ensuite, les classes minoritaires sont augmentées en échantillonnant avec remplacement, de sorte que leur taille soit égale à celle de la classe majoritaire. Enfin, les ensembles équilibrés sont combinés, créant un nouvel ensemble d'entraînement équilibré. Après, on précède à la méthode de normalisation en attribuant à chaque entrée une moyenne de 0 et un écart-type de 1 ce qui permet de standardiser les échelles des différentes caractéristiques.

On va définir et entraîner un modèle de forêt aléatoire, nous allons utiliser l'algorithme de classification RandomForest. À cette fin, nous ferons appel à la classe RandomForestClassifier du module sklearn pour construire notre modèle de classification en utilisant l'algorithme de forêt aléatoire. Le modèle est configuré avec 100 arbres et un critère de division nécessitant au moins 100 échantillons pour diviser un nœud. Le modèle est ensuite entraîné sur l'ensemble d'entraînement équilibré et normalisé en utilisant les entrées spécifiées et les cibles. Une fois le modèle entraîné, il est utilisé pour faire des prédictions sur l'ensemble de test. Ces prédictions sont stockées dans la variable 'preds' qui représentent les valeurs prédites du target pour les données de test.

Ensuite, nous évaluons les performances du modèle sur l'ensemble de test en utilisant la précision comme métrique d'évaluation en comparant les prédictions avec les valeurs réelles de la colonne 'Target'.

Par ailleurs, afin d'améliorer la précision, nous incorporons des fonctions pour la prédiction et l'évaluation sur plusieurs périodes temporelles (backtesting) ce qui peut fournir une évaluation plus robuste de la performance du modèle dans différents contextes temporels. Au lieu de diviser les données en utilisant une simple partition fixe on utilise une

fenêtre glissante, ce qui permet d'utiliser des périodes différentes pour l'entraînement et le test. Dans chaque itération de la boucle de backtesting, on reéchantillonne les différentes classes on normalise les données,on entraîne le modèle, et on fait des prédictions. Les résultats du backtesting sont ensuite combinés et la précision du modèle est calculée pour évaluer sa performance globale.

Réseaux de Neurones profond

Maintenant que nos entrées et sorties sont définies et notre base de données préparée, il ne nous restait plus qu'à créer notre IA. Nous avons opté pour un réseau de neurones profond (DNN) avec une dernière couche composée de trois neurones et une fonction d'activation softmax. Cette configuration permet de calculer la probabilité de chacune des sorties potentielles (-1, 0, 1) et de sélectionner celle qui est la plus probable.

Ensuite, il nous fallait déterminer le nombre de couches cachées, le nombre de neurones par couche, les fonctions d'activation, ainsi que le taux d'apprentissage. Cependant, il n'existe pas de méthode prédéfinie pour fixer ces paramètres. Nous avons donc commencé par des estimations approximatives, entraîné l'IA, puis évalué la précision obtenue sur l'ensemble de test (c'est-à-dire le nombre de fois où notre IA a correctement prévu le résultat souhaité). Le but était de trouver la configuration optimale des couches pour obtenir la meilleure précision possible.

Pour limiter le phénomène de sur-apprentissage (overfitting), nous avons mis en place des techniques de régularisation L2 et des Dropouts, dont les paramètres devaient également être optimisés.

4.3 Objectif 3 : Création de notre algorithme de prédiction final regroupant les deux intelligences artificielles précédentes

L'objectif de cette partie est d'améliorer la précision de notre Réseaux de Neurones de prédiction en considérant les sentiments grâce à l'incorporation du score donné par notre première intelligence artificielle comme une entrée supplémentaire. Cependant, pour inculquer cette nouvelle entrée : il faut entraîner à nouveaux le Réseaux de Neurones qui ne connaît pas cette nouvelle entrée. Ainsi, une nouvelle base de données doit donc être créée.

Par conséquent, étant donné que le réseau de neurones a été entraîné en utilisant des données financières couvrant les 24 dernières années, l'idéal serait de récupérer des articles et tweets sur la même période. Ainsi, des textes anciens seront prélevés à l'aide d'algorithmes de web scraping pour les soumettre à notre intelligence artificielle d'analyse de sentiments, afin de déterminer le sentiment du marché à n'importe quelle date. Nous combinerons ensuite tous ces scores avec la base de données contenant les indicateurs financiers pour créer la base de données finale.

Les articles disponibles sur une page web à un jour donné ne sont pas forcément accessibles dans le futur. Après une certaine période de temps, les articles le plus anciens sont substitués par des plus récents, ce qui exige, si l'on veut construire une base de données, l'accès complet à la base de données du site.

Le journal américain The Wall Street Journal rend disponible des outils du type "développeur" [18] qui, à partir des requêtes http identifiées avec une clé qu'ils fournissent, donnent accès à des informations directement issues de leur base de données d'articles. On peut notamment utiliser des filtres de date, de section (i.e. sports, finances, santé) ou des mot-clés. Le journal en question est le plus grand aux États-Unis, et donc une source pertinente et représentative de l'humeur ou de la perception collective de la population et du marché américain.

Le corps de la totalité des articles n'est pas disponible, celui-ci est disponible uniquement en payant. Cependant, le titre, le "snippet" (la preview) et le "lead-paragraph" sont renvoyés en tant que réponses. Ces trois éléments sont bien représentatifs du sentiment général exprimé dans le texte.

Nous avons cherché également à extraire d'anciennes publications postées sur les réseaux sociaux. Si nous n'avons trouvé aucun moyen d'effectuer ceci sur StockTwits, nous avons pu en revanche utiliser la fonction "recherche avancée" de Twitter, qui permet de rechercher les tweets contenant n'importe quel mot-clé et publiés à la date spécifiée. Grâce à cela, nous avons pu mettre au point un algorithme qui, pour une plage de dates donnée, renvoie un fichier excel pour chaque jour appartenant à la plage contenant le nombre de tweets choisis publiés ce jour.

Cependant, dans le cas où la base de données ne serait pas prête à temps, une solution de secours devait être trouvée. Ainsi, nous avons utilisé les données partagées par le projet de recherche Hedonometer, qui vise à mesurer le bonheur et l'émotion collective à partir de données textuelles : tweets et articles. Toutes les données disponibles entre 2009 et 2023 ont été récupérées et normalisées entre -1 et 1.[19]

Cependant, ceci implique que les indicateurs financiers ne pourront eux aussi être utilisé qu'entre 2009 et 2023 et non plus entre 2000 et 2024 ce qui implique une perte de données considérable : environ 100 000 ce qui est non négligeable vu la taille de notre nouvelle base de données 173 856.

5 Résultats

Nous sommes en capacité de récupérer de manière automatique quotidiennement des articles financiers provenant de YahooFinance et PrNews et des tweets extraits de Twitter et StockTwits.

Au niveau de l'intelligence artificielle d'analyse des sentiments le réseau retenu est un réseau composé de 20 couches, de 96 neurones chacune (taille qui correspond à la dimension des vecteurs obtenus par pré-traitement des textes via SpaCy).
Les fonctions de sauvegarde et de chargement du réseau sont fonctionnelles et se réalisent en un temps satisfaisant, en O(n), n le nombre de paramètres du réseau, ce qui prend moins d'une dizaine de secondes.
Les fonctions de propagation vers l'avant et de rétropropagation du gradient sont aussi fonctionnelles. Le pas d'apprentissage choisi est de $5 * 10^{-5}$. L'apprentissage a été effectué sur une base de données d'une taille de 65 000 éléments, dont la répartition de -1 à 1 est montrée ci-dessous. C'est un nombre très faible compte tenu du nombre de paramètres présents au sein du réseau. Mais compte tenu de la vitesse d'apprentissage du réseau, qui est de 1000 rétropropagations à l'heure, du nombre de tests à effectuer pour trouver une taille et un pas d'apprentissage satisfaisant, et de la puissance de l'ordinateur sur lequel a été effectué l'apprentissage, il est difficile d'augmenter le nombre de données sur lequel il est effectué.

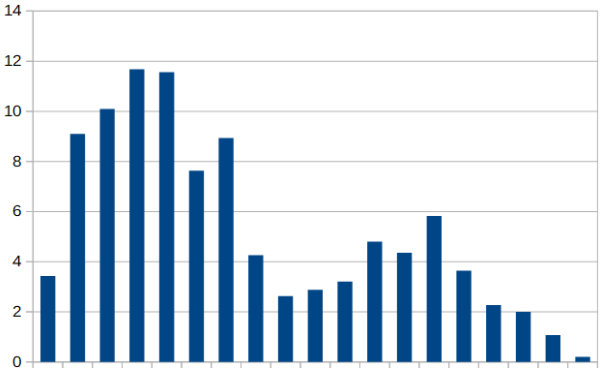


FIGURE 6 – Répartition des étiquettes dans le dataset d'entraînement et de test, classées de -1 à 1 au dixième près

On constate dans le dataset une répartition peu homogène des étiquettes, qui sont les valeurs associées aux textes du dataset d'entraînement. Plus de 65% des étiquettes sont négatives. Il convient donc d'ajuster le pas quand les étiquettes sont positives pour éviter que le réseau ne soit bloqué dans le négatif.

Si la répartition non-homogène des étiquettes peut poser un problème dans certains cas, ici cette répartition peut être considérée comme représentative des données qui seront fournies au réseau une fois celui-ci fonctionnel, ce qui ne constitue donc pas un gros désavantage.

Pour caractériser les performances du réseau, deux indicateurs sont utilisés :
— la distance moyenne à l'étiquette
— la proportion de réponse ayant le bon signe

Pour une étude plus précise des performances du réseau, il aurait été possible de calculer une distance relative en fonction de la valeur de l'étiquette : la différence entre 0.6 et 0.9 est bien plus faible que celle entre 0.2 et -0.1, car dans le premier cas l'erreur n'est que sur la "puissance" du sentiment, et dans l'autre le programme attribut un caractère "un peu négatif" à un message "plutôt positif". C'est pour cela que la proportion de réponses dans le "bon côté" de l'intervalle est également mesurée.

Les performances du réseau étaient au départ de 0.57 de distance moyenne, pour 67% de réponses avec le bon signe (dû à l'asymétrie du dataset, avec une initialisation qui amenait le réseau à répondre dans le négatif, les performances sont supérieures à 0.5). Une fois l'entraînement terminé, les performances du réseau sont de : 0.44 de distance moyenne, et 72% de réponses avec le bon signe.

Les performances du réseau se sont améliorées à la suite de l'apprentissage. Mais plusieurs éléments sont à prendre en compte :

- les indicateurs de performances peuvent être améliorés (avec une fonction de calcul de l'erreur plus adéquate, et la proportion de réponses avec le bon signe est à remettre dans le contexte d'un dataset asymétrique)
- l'entraînement aurait dû être réalisé sur un ensemble plus conséquent de données, ce qui aurait pu permettre de mieux étudier les potentiels points faibles de la fonction d'apprentissage du réseau, par exemple si le réseau arrive à converger vers un état stable, si cette convergence n'est pas trop rapide ou trop tardive...
- un pas d'apprentissage mieux choisi aurait pu permettre des améliorations plus significatives des performances du réseau, mais le choix de ce pas aurait nécessité trop de temps dans le cadre du projet.

Pour résumer les résultats relatifs à la partie "analyse de sentiments" du projet : toutes les fonctions relatives au réseau sont utilisables. L'évolution des performances du réseau est encourageante, mais elle reste critiquable. Le manque de temps et de moyens techniques ne permet pas de pousser l'entraînement, et donc ne permet pas une analyse complète des performances. Certaines caractéristiques du réseau aurait également pu être plus optimisées, mais cela demanderait une augmentation des moyens alloués et du temps attribué au projet.

Au niveau des deux programmes de prédiction utilisant uniquement les indicateurs financiers, voici les résultats obtenus :

Intelligence Artificielle en suivant des tutoriels

Pour évaluer la robustesse de notre modèle, nous calculons le score de prédiction, souvent appelé métrique de performance. Ce score sert à évaluer la qualité des prédictions faites par le modèle par rapport aux valeurs réelles des données de test. Nous obtenons les résultats suivants :

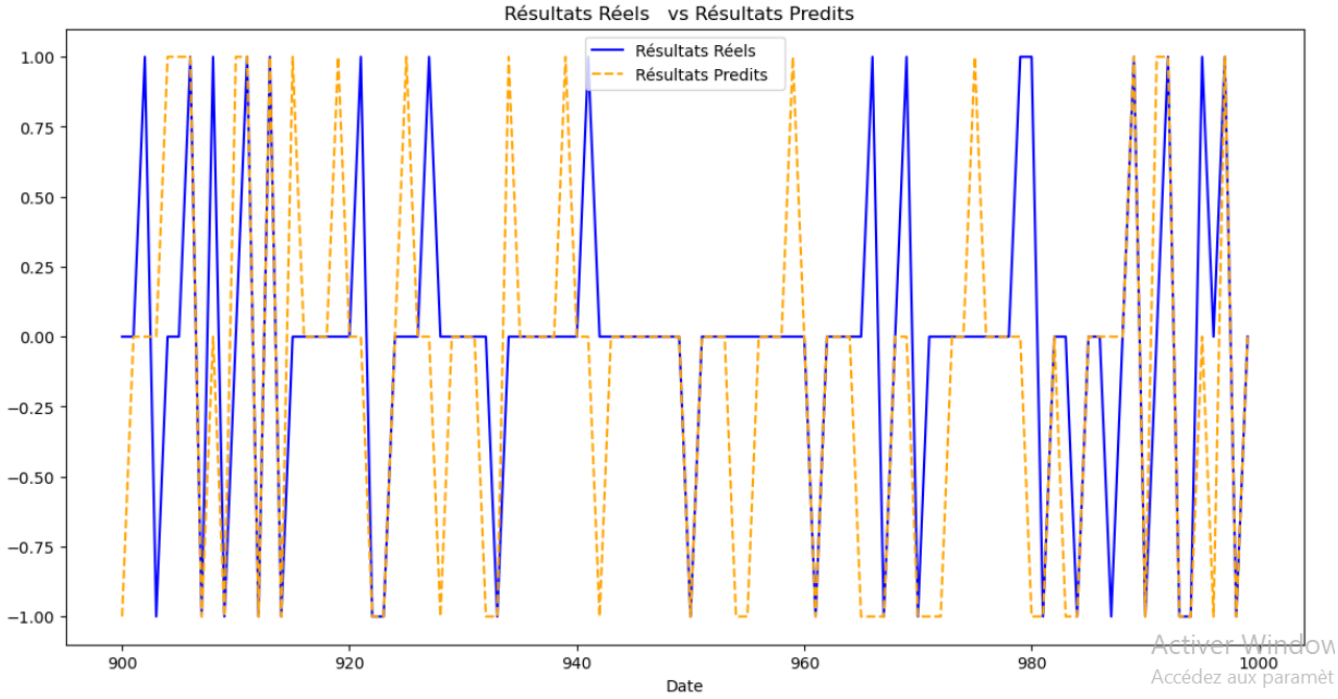


FIGURE 7 – Visualisation du score de prédiction

En combinant la méthode du backtesting avec les indicateurs financiers nous trouvons :

```
Précision du modèle : 0.5825396825396826
```

FIGURE 8 – Score de prédiction avec la méthode du backtesting et les indicateurs financiers

Réseaux de Neurones Profond

Après avoir longuement entraîné notre réseau de neurones profond (DNN) avec différentes configurations, nous avons identifié l'architecture optimale : trois couches distinctes et 16 neurones par couche avec un taux d'apprentissage de $\alpha = 10^{-5}$.

Nous avons obtenu une précision de 58% sur le jeu d'entraînement, 60% sur le jeu de validation et 58% sur le jeu de test (confère figure 10) . Malheureusement, nous devons signaler que notre programme atteint rapidement un plafond et cesse de s'améliorer : confère figure 9.

Ce phénomène peut s'expliquer par deux principales raisons :

1. Un manque flagrant de données : nous disposons de moins de 200 000 échantillons.
2. Une architecture de réseau trop simple : des réseaux de neurones plus sophistiqués tels que les réseaux récurrents (RNN) ou les LSTM (Long Short Term Memory) sont plus performants pour réaliser ce genre de tâche prédictive. Cependant, ils demandent aussi plus de données car le nombre de paramètres à entraîner est plus conséquent.

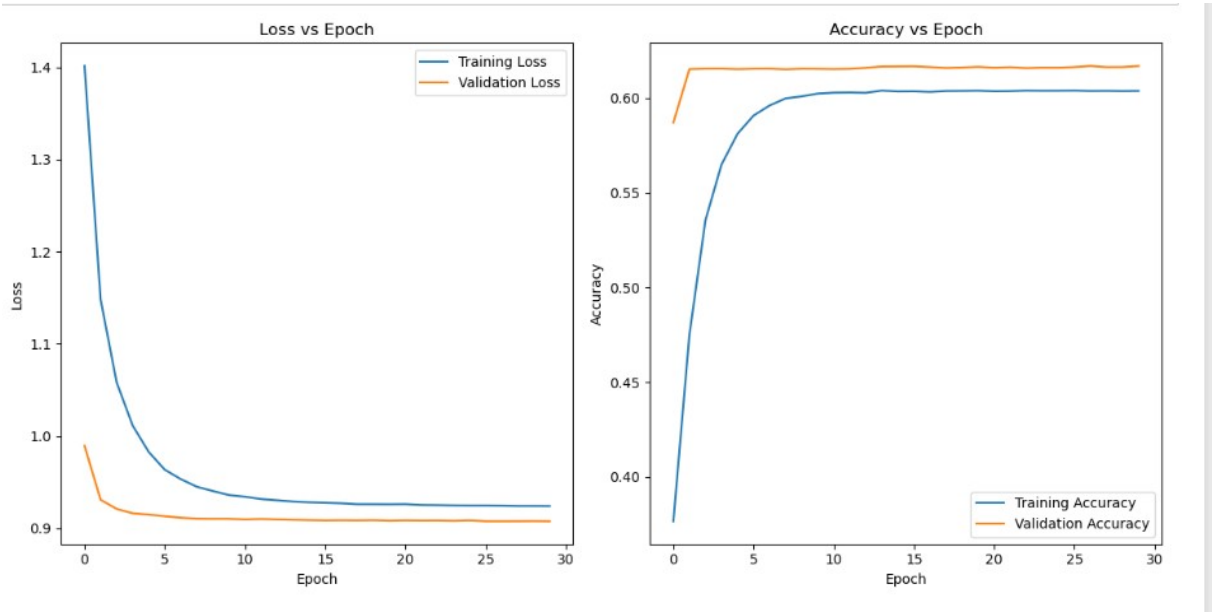


FIGURE 9 – Courbes montrant l'évolution de la fonction coût et de la précision en fonction du nombre d'epochs

```
Test accuracy: 0.582308828830719
```

FIGURE 10 – Résultat du test avec les indicateurs financiers

Bien qu'une précision de 58% sur le jeu de test soit modeste, elle reste encourageante concernant notre objectif de 60%. Cependant, nous avons décidé de ne pas nous attarder sur les 2% restants pour nous concentrer sur l'avancement du programme final.

Cependant, il fallait tout d'abord choisir quelle IA de prédiction nous comptions utiliser pour l'algorithme final et sachant que les deux donnent des résultats équivalant nous avons opté pour celle dont nous avons la meilleure compréhension : le Réseaux de Neurones.

Ainsi pour entraîner notre Réseaux de Neurones en rajoutant les sentiments du marchés aux indicateurs financiers il fallait avoir à disposition une nouvelle base de données. L'idéal aurait été de créer cette nouvelle base en utilisant l'intelligence artificielle d'analyse des sentiments. Cependant même si cette dernière est opérationnelle, par manque de temps nous n'avons pas pu créer notre base de données.

Nous avons donc été contraints de passer au plan B : utiliser la base de données Hedonometer.

Les résultats obtenus sont décevants mais compréhensibles. En effet, même si à première vue tout marchait à la perfection : précision de 64 % en phase d'entraînement, 59% en phase de validation (confère figure 11). Nous obtenons à la fin une précision de 52% en phase de test : confère figure 12.

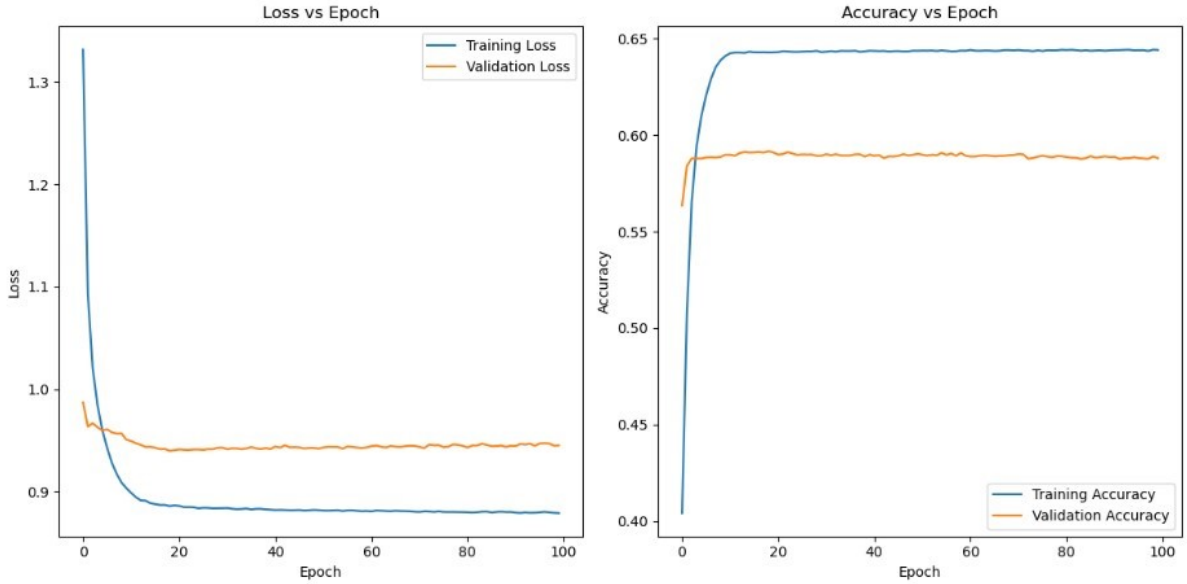


FIGURE 11 – Entraînement en considérant l'analyse des sentiments

```
Test Accuracy: 0.5206037759780884
```

FIGURE 12 – Résultat du test en considérant l'analyse des sentiments

Ceci est principalement dû à 2 raisons :

— Un manque considérable de données car seules celles entre 2009 à 2023 ont été utilisées ce qui équivaut à 173856 données .

— Le sentiment pour un jour donné contrairement aux indicateurs financiers - qui eux sont propres à chaque action - est fixe. En effet, dans notre base de données, nous avons 40 cours d'actions différents. Lors de l'entraînement, l'IA essaie de prédire la variation du cours pour chaque action au jour t+1 en utilisant les indicateurs financiers liés au cours au jour t. Les indicateurs financiers du jour t sont bien sûr différents pour chaque action. Cependant, la note de sentiment n'étant pas liée à une action ou à un cours spécifique, elle reste la même pour les 40 actions. Par conséquent, le programme observe qu'une même note de sentiment conduit à des variations totalement différentes, ce qui complique l'apprentissage correct de l'IA.

Ainsi, pour éviter ce phénomène de score global : il serait nécessaire de réaliser une analyse de sentiment spécifique à chaque domaine pour éviter d'avoir deux résultats opposés pour le même score. Ce qui est tout à fait possible comme le montre l'exemple du Covid. En effet pour une même note négative nous avons constaté : une baisse importante des actions liés aux compagnies aériennes cotées en bourse contre une hausse significative des compagnies pharmaceutiques.

Par conséquent il faudrait opter pour un vecteur de sentiments et non plus une simple note. Et il y aurait autant de notes dans le vecteurs de sentiments que de secteurs qui nous intéresse : technologie, divertissement ...

6 Conclusion

Pour expliquer les résultats obtenues il est indispensable de se remettre ce schéma en tête.

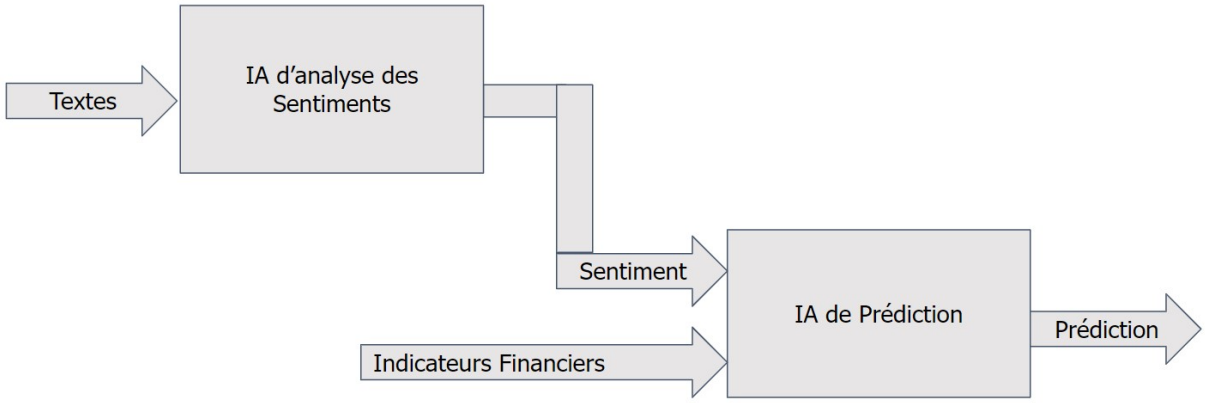


FIGURE 13 – Schéma de synthèse de notre algorithme

Notre objectif était de prédire les variations du cours d'une action boursière pour le jour t+1. Pour ce faire, nous avons mis en place une méthode structurée en plusieurs étapes. Tout d'abord, nous récupérons automatiquement des tweets et des articles financiers. Ces informations sont ensuite analysées par notre intelligence artificielle de traitement des sentiments, qui leur attribue une note comprise entre -1 et 1. La moyenne de ces notes est calculée et transmise à notre seconde intelligence artificielle dédiée à la prédiction. Cette dernière importe alors toutes les données disponibles sur le cours de l'action pour le jour t et calcule divers indicateurs financiers. Enfin, en utilisant les indicateurs financiers et la note de sentiment, une prédiction des variations du cours de l'action au jour t+1 est effectuée avec une probabilité de succès de 58%. Ce qui est très satisfaisant au vue de l'efficience des marchés : il est normalement impossible de prévoir les rentabilités futures.

Les perspectives futures incluraient l'augmentation de nos bases de données pour améliorer considérablement la précision de nos intelligences artificielles. De plus, nous envisagerions d'utiliser un vecteur de sentiments, c'est-à-dire plusieurs notes comprises entre -1 et 1, chacune correspondant à un secteur différent : technologie, énergie... Enfin, nous souhaiterions perfectionner la méthode d'obtention de la note globale. Plutôt que de simplement calculer la moyenne de toutes les notes obtenues au jour t, nous chercherions à déterminer une pondération plus judicieuse pour chaque note.

7 Bibliographie

Références

[1] Hsinchun Chen Rob Shumaker. *Textual analysis of stock market prediction using breaking financial news : The AZFin text system.*

[2] Xiaojun Zeng Johan Bollena Huina Maoa. *Twitter mood predicts the stock market.*

[3] Indranil Bose Suparna Dhar. *Emotions in Twitter communication and stock prices of firms : the impact of Covid-19 pandemic.*

[4] Thomas Renault. *Intraday online investor sentiment and return patterns in the U.S. stock market.*

[5] *A Practical Introduction to Web Scraping in Python.* URL : https://realpython.com/python-web-scraping-practical-introduction/.

[6] *Les réseaux de neurones récurrents, des RNN simples aux LSTM.* URL : https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm.

[7] *Modern reccurent network : bidirectionnal-RNN.* URL : https://d2l.ai/chapter_recurrent-modern/bi-rnn.html.

[8] *LSTM derivation of back propagation throught time.* URL : https://www.geeksforgeeks.org/lstm-derivation-of-back-propagation-through-time/.

[9] *What is Xavier initialization ?* URL : https://365datascience.com/tutorials/machine-learning-tutorials/what-is-xavier-initialization/.

[10] Ludovic Araye-Elouan Bisson-Achille-Henrotte-Thomas-Sulpice-Ulysse Koechlin et Maxime Moreau. *PE 46 : Optimisation d'un portefeuille d'actions à l'aide d'un réseau de neurones.* 2023.

[11] Yann Le Cunn. *Quand la machine apprend.* 2019.

[12] Andrew Ng. *Deep Learning Specialization.* URL : https://www.coursera.org/specializations/deep-learning.

[13] *precision.* URL : https://kobia.fr/classification-metrics-accuracy/?fbclid=IwZXh0bgNhZW0CMTAAAR2GDi55Na8wzk5IlvA8V85EuYgZ4A2ql2nmpHfFdL76K5MJZz1x90iaem_AfbHLgjSJT69BRpK7vvmE4wxIgH88GtEKe_kX0ddl2hO6RYxtlnAHGetdT325I4H_wAk-R9psZarNBYvtohto3DL.

[14] Dr Kevin Webster. *Getting Started with Tensorflow.* URL : https://www.coursera.org/learn/probabilistic-deep-learning-with-tensorflow2.

[15] *RandomForestClassifier.* URL : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[16] *Backtesting.* URL : https://www.avatrade.fr/education/online-trading-strategies/backtesting-trading-strategies#:~:text=Le%20backtesting%20d%27un%20portefeuille,couvrir%20plusieurs%20classes%20d%27actifs..

[17] *base de données d'origine issue de Kaggle.* URL : https://www.kaggle.com/datasets/ankitkumar2635/sentiment-and-emotions-of-tweets.

[18] *The New York Times Developper's Portal*. URL : https://developer.nytimes.com/apis.

[19] Peter DODDS et Chris DANFORTH. *Hedonometer*. URL : https://hedonometer.org/timeseries/en_all/?from=2022-01-01&to=2023-06-30.

8 Annexe

Actions constituant la base de donnée

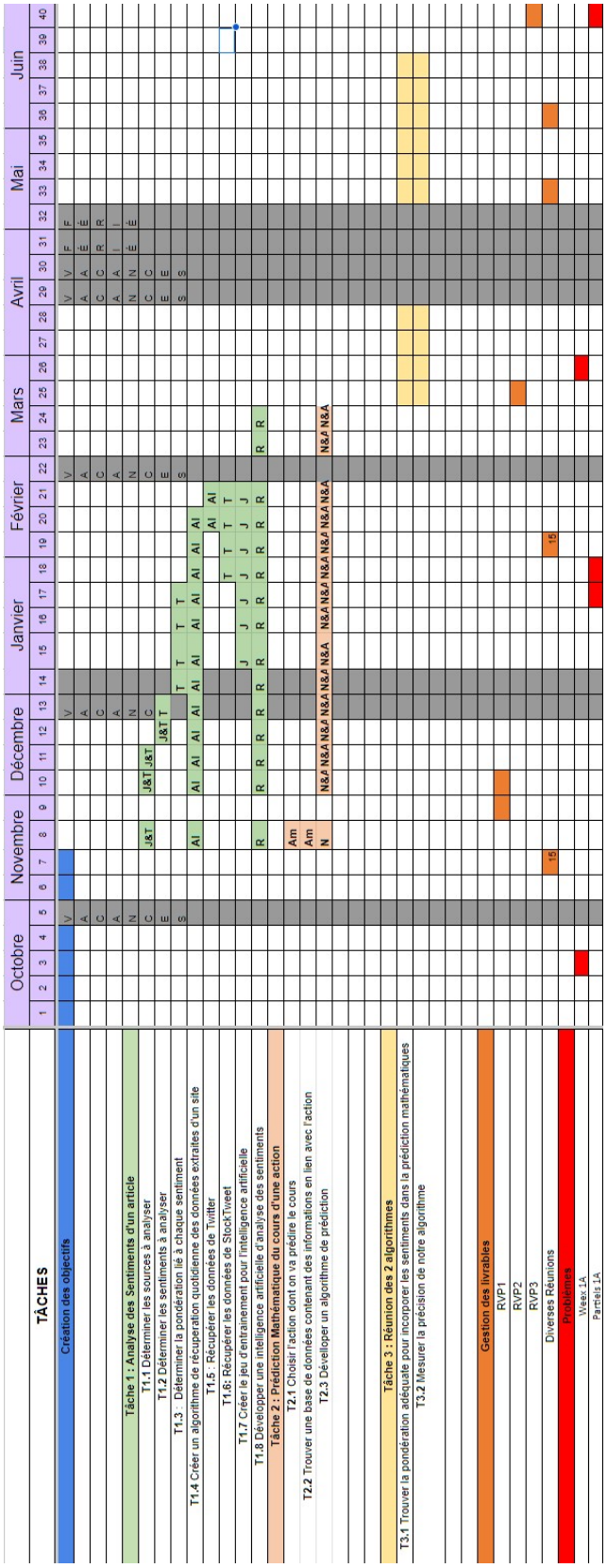
Agroalimentaire	Assurance	Divertissement	Distribution	Énergie	Technologie
The Coca-Cola Company	MetLife	The Walt Disney Company	Walmart	ExxonMobil	Apple
PepsiCo	Prudential Financial	Netflix	Amazon	Chevron	Microsoft
Mondelez International	Aflac	Comcast	Target	BP	Alphabet (Google)
Tyson Foods	American International Group	Live Nation Entertainment	Costco		Nvidia
Archer-Daniels-Midland	Allstate		Home Depot		

FIGURE 14 – Actions sélectionnées

Pharmaceutique	Industriel	Mode	Sport	Télécommunication	Business
Pfizer	3M	Nike	Nike	AT&T	Salesforce
Merck	Honeywell	Lululemon Athletica	Under Armour	Verizon	MercadoLibre
GlaxoSmithKline	General Electric	PVH Corp.	Adidas	T-Mobile	
	Caterpillar	Gap Inc.	The Walt Disney Company		
	Boeing	Foot Locker			

FIGURE 15 – Actions sélectionnées

GANTT


TÂCHES
Octobre
Novembre
Décembre
Janvier
Février
Mars
Avril
Mai
Juin
Création des objectifs
Tâche 1 : Analyse des Sentiments d'un article
T1.1 Déterminer les sources à analyser
T1.2 Déterminer les sentiments a analyser
T1.3 : Déterminer la pondération lié à chaque sentiment
T1.4 Créer un algorithme de récuperation quotidienne des données extraites d'un site
T1.5 : Récupérer les données de Twitter
T1.6: Récupérer les données de StockTweet
T1.7 Créer le jeu d'entrainement pour l'intelligence artificielle
T1.8 Développer une intelligence artificielle d'analyse des sentiments
Tâche 2 : Prédiction Mathématique du cours d'une action
T2.1 Choisir l'action dont on va prédire le cours
T2.2 Trouver une base de données contenant des informations en lien avec l'action
T2.3 Développer un algorithme de prédiction
Tâche 3 : Réunion des 2 algorithmes
T3.1 Trouver la pondération adéquate pour incorporer les sentiments dans la prédiction mathématiques
T3.2 Mesurer la précision de notre algorithme
Gestion des livrables
RVP1
RVP2
RVP3
Diverses Réunions
Problèmes
Week 1A
Partiels 1A

CheckList

Contenu

Résumé en français	X	
Résumé en anglais	X	
Table des matières	X	
Table des figures	X	
Introduction	X	
Conclusion générale	X	
Bibliographie	X	
Citation des références dans le texte	X	

Forme

Vérification orthographe	X	
Pagination	X	
Homogénéité de la mise en page	X	
Lisibilité des figures	X	

FIGURE 16 – CheckList

Budget :

Au niveau du budget, nous n'avons rien dépenser des 300 € mis à notre disposition. Cependant, si nous devions calculer le coût complet de notre projet d'étude en considérant les tarifs horaires dépensés pour une entreprise :

- **Monsieur de Peretti** nous a délivré 3 cours de 3h et 5 réunions d'une durée de 1h. Ainsi nous avons utilisé 14h de son temps. Si nous prenons un tarif horaire 90 €, un total de 1260 € aurait été à déboursé.
- **Monsieur Bouillet :** Monsieur Bouillet participera à 8 réunions ce qui fait un total de 8h pour notre groupe, soit en se basant sur le même tarif horaire que Monsieur De Peretti un total de 840 €
- **Nous :** Sachant que notre projet d'étude s'est déroulé sur une durée de 9 mois, vacances comprises car nous ne nous sommes pas arrêtés. Nous avons donc travaillé

36 semaines à un rythme de 5 heures par semaine. Cela représente un total de 180 heures de travail individuel. En considérant un salaire horaire de 50 €, le coût total s'élève à 54 000 €.

Ce qui fait un total de 56 100 €.