République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté de : Génie électrique et d'informatique
Département : Informatique

Mémoire de fin d'études

En vue de l'obtentiondudiplôme de

Master en Informatique

Spécialité : Systèmes Informatiques

Présenté par :

Imane KHELOUATet **Katia MOUALI**

Sujet : Anticipation de la valeur des crypto-monnaies grâce à l'apprentissage automatique

Proposé et dirigé par : Samy SADI

Soutenu le 11 novembre 2020 devant le jury composé de:

M. Idir FILALI	Président du Jury
M. Mohamed RAMDANI	Membre du Jury
M. Samy SADI	Directeur de mémoire

Dédicaces

À mes chers parents qui ne cessent de me donner l'amour nécessaire pour que je puisse arriver à ce que je suis aujourd'hui.

À mes chers grand-parents qui m'ont doté d'une éducation digne.

À mes sœurs, ma source de joie et de bonheur.

À mon frére.

À vous, chers lecteurs.

.

Imane

Dédicaces

À mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.

À mon très cher frére Adel.

À tous les membres de ma famille, petits et grands.

À vous, chers lecteurs.

.

Katia

Remerciements

Nous remercions notre encadreur, Monsieur Sadi, pour tout ce qu'il nous a appris dans le monde de l'intelligence artificielle ainsi que toutes ses remarques pertinentes.

Nous remercions l'ensemble des membres du jury, qui nous ont fait l'honneur de bien vouloir étudier avec attention notre travail.

Nous remercions toutes les personnes qui ont contibuer à la realisation de ce travail.

Résumé

Ce mémoire s'inscrit dans le cadre de l'application de l'apprentissage automatique dans la prédiction des prix des crypto-monnaies.

Depuis 2013, plus de 400 crypto-monnaies ont été recensées. Bien comprendre l'évolution de la valeur de ces monnaies permettrait d'établir des stratégies de trading efficaces. Pour ce faire, nous avons d'abord collecté des données pertinentes en quantités suffisantes, ces dernières sont nettoyées, elles sont ensuite regroupées de sorte à former une large base de données organisée qui seront reparti plus tard en données d'entrainement, de validation et de teste. Puis, nous avons implémenté les algorithmes et les modèles les plus utilisées dans la plupart des études de marché financier. Ainsi, nous avons conçu trois modèles basés sur les réseaux de neurones artificiels sélectionnés pour leurs performances lors de la phase d'entrainement. Enfin, nous avons testé la qualité de ces modèles en réalisant des comparaisons entre les modèles lors duquel nous avons eu de bons résultats. L'un des succès est l'évaluation obtenue à l'aide du premier modèle, qui a confirmé ses statistiques lors de l'entrainement.

Mots clés : Crypto-monnaies, prediction, intelligence artificielle, apprentissage automatique, apprentissage profond, réseaux de neurones artificiels.

Abstract

This dissertation is part of the application of machine learning in the prediction of crypto-currency prices.

Since 2013, more than 400 crypto-currencies have been identified. Understanding the evolution of the value of these currencies would allow the establishment of effective trading strategies. To do this, we first collected relevant data in sufficient quantities, these are cleaned, they are then grouped together so as to form a large organized database which will later be divided into training, validation and of test. Then, we implemented the algorithms and models most used in most of the financial market studies. We have therefore designed three models based on artificial neural networks selected for their performance during the training phase. Finally, we tested the quality of these models by making comparisons between models during which we had good results. One of the successes was the rating obtained using the first model, which confirmed her stats in training.

Keywords: Crypto-currencies, prediction, artificial intelligence, machine learning, deep learning, artificial neural networks.

Glossaire

AA : Apprentissage Automatique.
IA : Intelligence Artificielle.
RNN : Recurrent Neural Network.
RNA : Réseau de Neurones Artificiels.
ANN : Artificial Neural Network.
ResNet : Residual Network.
LSTM : Long Short-Term Memory.
ConvNet : Convolutional Network.
GRU : Gated Recurrent Unit.
K-NN : K nearest neighbors en anglais.
CNN : Convolutionnal Neural Network.
ReLU : Rectified Linear Unit.
LIMICS : Laboratoire d'Informatique Médicale et d'Ingénierie des Connaissances en e-Santé.
PC : Personal Compute.r
Ags : Algorithme du gradient stochastique.
ETH: la crypto-devise Etherium.
BTC : la crypto-devise Bitcoin.
USD : Dollar.
LTC : la crypto-devise Litecoin.
BPTT : Backpropagation through time.

Table des matières

Table des tableaux et des figures

Introduction générale

1 Contexte du travail

Notre travail s'inscrit dans le cadre de l'anticipation des coûts des crypto-monnaies grâce à l'apprentissage automatique.

En effet, il existe plusieurs stratégies de trading qui permettent d'obtenir des signaux d'achat ou de vente tel que l'analyse technique, mais le marché de la crypto-monnaies est en plein essor, plus de 400 crypto-monnaies ont été recensées depuis 2013, et la complexité des unités de temps disponibles ne rendent la tâche que plus complexe aux analystes, ainsi le travail est long et fastidieux, et les analystes ratent de nombreuses opportunités de trading.

D'autre part, le domaine de l'apprentissage automatique a connu beaucoup de développement, et notamment celui des réseaux de neurones qui permettent de réaliser des programmes plus performants que ceux qui existent auparavant dans le domaine de la prédiction en général, et la prédiction financière plus exactement.

2 Problématique

Le succès des techniques d'apprentissage automatique pour la prévision des marchés boursiers implique que ces méthodes pourraient également être efficaces pour prédire les prix des crypto-monnaies. Cependant, l'application des algorithmes de l'apprentissage automatique sur le marché de la crypto-monnaie jusqu'à présent, s'est limitée à l'analyse des prix du Bitcoin enutilisant des forêts aléatoires, des réseaux de neurones bayésien, et d'autres algorithmes. Ces études ont pu anticiper à des degrés divers, les fluctuations du prix de Bitcoin, et ont révélé que les meilleurs résultats ont été obtenus par des algorithmes basés sur les réseaux de neurones.

La problématique qui se pose alors est celle du choix des modèles de réseaux de neurones à utiliser pour la prédiction des coûts des crypto-monnaies. Il faut pouvoir choisir des modèles performants, puis les évaluer et enfin choisir un modèle de prédiction final.

3 Contribution

Notre contribution dans le cadre de ce projet se situe à plusieurs niveaux:

1. Nous avons proposé trois nouveaux modèles de réseaux de neurones pour la prediction des prix des crypto-monnaies.
2. Nous avons entrainé les modèles proposés.
3. Enfin, nous avons testé les modèles en effectuant des prédictions et en comparant entre les résultats obtenus et les resultats attendus.

4 Organisation du mémoire

Ce mémoire est organisé en quatre chapitres.

Dans le premier chapitre, nous avons introduit la notion de l'intelligence artificielle et l'apprentissage automatique en général.

Dans le deuxième chapitre, nous avons présenté les réseaux de neurones artificiels, leurs fonctionnements, leurs différentes architectures et leurs types d'entrainements.

Dans le troisième chapitre, nous avons décrit les quatres modèles proposés de réseaux de neurones artificiels.

Dans le quatrième chapitre, nous avonsprésenté la conception et l'implémentation de notre application ainsi que son déploiement où nous avons présenté les résultats de nos modèles.

Pour finir, nous avons réalisé une synthèse générale de notre mémoire ainsi qu'une liste de perspectives de notre application que nous comptons implémenter à l'avenir. Par la suite, nous avons conclu avec une bibliographie qui rassemble toutes nos références théoriques.

Chapitre 1: Généralités sur l’apprentissage automatique

1 Introduction

L'apprentissage automatique *(AA)* permet à une machine d'évoluer grâce à un processus d'apprentissage, et lui permet ainsi de remplir des taches qu'il est difficile ou impossible de remplir avec les algorithmes classiques.

L'apprentissage automatique touche à de nombreux domaines notamment au domaine financier et plus précisément au *trading*[1], dans lequel la machine est capable de prédire l'état du marché financier. Nous souhaitons anticiper les valeurs des crypto-monnaies en utilisant l'apprentissage automatique, pour comprendre comment la machine arrive à effectuer cette tâche, des connaissances de base en apprentissage automatique sont nécessaires.

Dans ce chapitre nous présentons des généralités sur l'Apprentissage automatique. Pour ce faire, nous commençons par une petite introduction à l'intelligence artificielle *(IA).* Puis nous donnons une brève définition de l'apprentissage automatique. Apres quoi, nous exposons son fonctionnement global et ses types. Enfin, nous concluons avec quelques techniques d'apprentissage automatique et les recentes applications de ce dernier.

2 Intelligence artificielle

L'intelligence artificielle *(IA)* regroupe un ensemble de théories et techniques dont le but est de créer des machines capables de simuler l'intelligence. Son origine remonte en 1936, avec le concept de la « machine universelle » d'Alain Turing, ainsi qu'à l'article « l'ordinateur et l'intelligence » qu'il a publié un peu plus tard en 1950.Une autre origine probable est la publication en 1949, d'un mémorandum sur la « traduction automatique des langues » par Warren Weaver qui suggère qu'une machine puisse accomplir une tâche qui relève de l'intelligence humaine.

Le développement des techniques informatique aboutit à plusieurs avancées. Ainsi en 2011 l'intelligence artificielle a franchi une grande étape grâce à trois grandes ruptures quasi concomitantes à savoir : l'introduction d'algorithme bien plus sophistiqués (réseaux de neurones convolutifs), l'arrivée sur le marché de processeurs graphiques à bas coûts capables d'effectuer d'énormes quantités de calculs et la disponibilité de très grandes bases de données correctement annotées permettant un apprentissage plus fin.Neanmoins les

[1] Mot anglais, désigne les opérations d'achats et de ventes effectuées sur les marchés financiers.

promesses non tenues des débuts de l'intelligence artificielle ont amené à distinguer deux catégories différentes a savoir : l'intelligence artificielle Forte qui se distingue par des machines qui mettent en œuvres des raisonnements semblables aux raisonnements humains, qui auraient également une réelle conscience d'elle-même et l'intelligence artificielle Faible : se distingue par des machines qui rendent de nombreux sévices aux humains en simulant l'intelligence humaine. (1)

3 Apprentissage automatique

3.1 Définition

L'apprentissage automatique (AA) est un sous domaine de l'intelligence artificielle (IA). Il se base sur des approches mathématiques et statistiques, afin que la machine puisse apprendre elle-même sans qu'elle soit programmée pour. (2)

3.2 Fonctionnement global

Le fonctionnement global de l'apprentissage automatique repose sur quatre phases séquentielles. Ces phases sont les suivantes :

1. Collection de données : après l'acquisition de données pertinentes en quantités suffisantes, ces dernières sont nettoyées, elles sont ensuite regroupées de sorte à former une large base de données organisée selon le type d'apprentissage (voir type d'apprentissage automatique). En generale, les données sont séparées avec les proportions suivantes (3) : 20% pour le test et dans les 80% restant : 20% pour la validation et le rest pour l'entrainnement.
2. Entrainement : est une phase très importante dans laquelle les paramètres du model sont ajustés afin d'optimiser les résultats et réduire l'erreur en utilisant des méthodes telles que la descente de gradient … (voir section suivante).
3. Validation : permet d'évaluer sur un deuxième jeu de données les performances du model obtenu lors de la phase d'entrainement, et ainsi vérifier que le model ne fait pas de sur-apprentissage[2].
4. Test : permet de tester le model avec de nouvelle données.

[2] Un model statistique qui contient plus de paramètres que ne peuvent le justifier les données.

Mis à part la phase de collecte de données, les autres phases se répètent indéfiniment jusqu'à atteindre la performance optimale voulue.

3.3 Type d'apprentissage automatique

L'apprentissage automatique touche à plusieurs domaine et se confronte à divers problèmes, donc les approches à adopter et les algorithmes à utilisés diffèrent en fonction des taches à réaliser et du type de données (données étiquetées et/ou non étiquetées) disponibles. C'est pourquoi il y a plusieurs types d'apprentissage automatique on peut citer ce qui suit.

3.3.1 Apprentissage supervisé

Il est appelé apprentissage supervisé car la machine est assistée par un expert. Dans ce type d'apprentissage automatique, on dispose de données préalablement étiquetées (données pour lesquelles on connait déjà la réponse cible) par un expert. La machine doit comparer ses sorties Yobtenues en appliquant un algorithme A, aux données d'entrées X et sorties fournies par l'expert. La machine doit alors trouver une fonction réduisant l'écart d'erreur entre les résultats obtenus et les résultats attendus. Le but des méthodes d'apprentissage supervise est de bien généraliser, c'est-à-dire d'apprendre une fonction qui fasse des prédictions correctes sur des données non présentes dans l'ensemble d'apprentissage. Ce type d'apprentissage automatique est principalement utilisé pour résoudre les problèmes de régression[3] et de classification[4].

3.3.2 Apprentissage non-supervisé

L'apprentissage non-supervise est une discipline dont le concept est : à partir de données non étiquetées X, la machine doit apprendre à regrouper au sein d'un même ensemble des données ayant des similarités communes.

Il est principalement utiliser pour la clusterisation[5]. La machine fait alors elle-même les rapprochements en fonction de ces caractéristiques qu'elle est en mesure de repérer sans nécessiter d'intervention externe. De cette capacité à effectuer de la clusterisation découle également la possibilité de mettre un système de recommandation, ainsi que la possibilité de mettre au point un système de détection d'anomalie.

[3] Caractérisée par des sorties indéfinies sous forme de valeur numériques continues.

[4] Caractérisée par des sorties finies sous forme de catégories (discrètes) allant de binaire jusqu'à n-aires.

[5] Procédé destiné à regrouper un ensemble d'éléments hétérogènes, sous forme de sous groupe homogènes, ou lier par des caractéristiques communes :

3.3.3 Apprentissage semi-supervisé

Cette approche d'apprentissage fait intervenir en entrée aussi bien des données étiquetées que non étiquetées. Elle représente un bon compromis entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non-supervisé qui n'utilise que des données non étiquetées. L'utilisation des données non étiquetées en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage.

3.3.4 Apprentissage par renforcement

L'apprentissage par renforcement fait référence à une classe de problème d'apprentissage automatique, dont le but est d'apprendre a partir d'expérience successive de façon a trouver la meilleure solution.

Dans un tel problème, on dit qu'un agent interagit avec l'environnement pour trouver la solution optimale. L'apprentissage par renforcement diffère fondamentalement des problèmes supervises et non supervises par ce cote interactif et itératif : l'agent essaie plusieurs solution (exploration), observe la réaction de l'environnement et adapte son comportement (les variables) pour trouver la meilleure stratégie (il exploite le résultat de ses explorations).

Cette méthode est particulièrement adaptée aux problèmes nécessitant un compromis entre la quête de récompense à court terme et celle de récompense à long terme. Parmi les exemples de problèmes traités de cette façon : apprendre à un robot à marcher en terrain difficile, à conduire (voiture autonome), piloter un agent a travers un labyrinthe, etc.

3.4 Techniques d'apprentissage automatique

L'apprentissage automatique est étroitement lié aux statistiques; disposer d'une connaissance approfondie en statistique est donc utile pour comprendre et exploiter les algorithmes d'apprentissage automatique.

Les approches de l'apprentissage automatique sont continuellement développées. Dans ce qui suit nous citons quelques techniques utilisées par l'apprentissage automatique.

3.4.1 Arbre de décision

L'arbre de décision est un model prédictif, utilisépour représenter visuellement les décisions et montrer ou éclairer la prise de décision.

Dans ce model, les nœuds représentent les tests et les feuilles représentent les valeurs prédites, tandis que les arêtes désignent une réponse au test du nœud. La figure ci-dessous décrit une situation où une banque peut accorder ou non un prêt à un candidat, chaque individu est évalué sur un ensemble de variables testées dans les nœuds internes, les décisions sont prises dans les feuilles. (4)

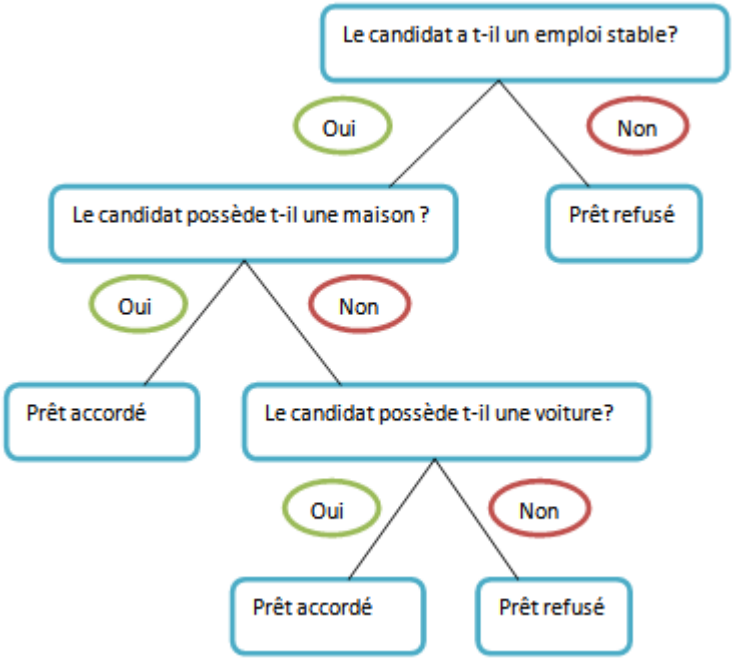


Figure 1 : Exemple arbre de décision.

3.4.2 Forêt d'arbres decisionnels

Forêt d'arbres décisionnels ou forêt aléatoire (*Random forest* en anglais) est algorithme d'apprentissage automatique, il est nommé ainsi car il regroupe un ensemble d'arbres de décisions, il est utilisé aussi bien pour les problèmes de classification et de régression.

L'apprentissage s'effectue sur plusieurs arbres, et chacun est entrainé sur un sous ensemble de données aléatoires différentes, les prédictions sont ensuite moyennées lorsque les données sont quantitatives dans le cas de la régression, ou utilisées pour un vote lorsque les données sont qualitatives dans le cas de la classification.

L'objectif de cette méthode est d'améliorer les performances prédictives et limiter le sur-apprentissage.

3.4.3 K plus proches voisins

K plus prochesvoisins (*K-NN : K nearest neighbors* en anglais) est un algorithme d'apprentissage supervise, qui permet de faire des prédictions sur des variables qualitatives dans le cas de la classification ou quantitative dans le cas de la régression.

Pour pouvoir effectuer une prédiction, l'algorithme se base sur un jeu de données en entier. Pour une nouvelle entrée X, la méthode de K-plus proches voisins prend les K-échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée X en se basant sur la distance. Dans la figure ci-dessous, un exemple qui prédit la classe d'un nouvel élément, en se basant sur ses K voisins les plus proches et en choisissant la classe des voisins majoritaires, dans cet exemple le nouvel élément appartiendra à la classe B car elle est plus majoritaire que la calsse A (nombre d'éléments de B > nombre d'éléments de A). (5)

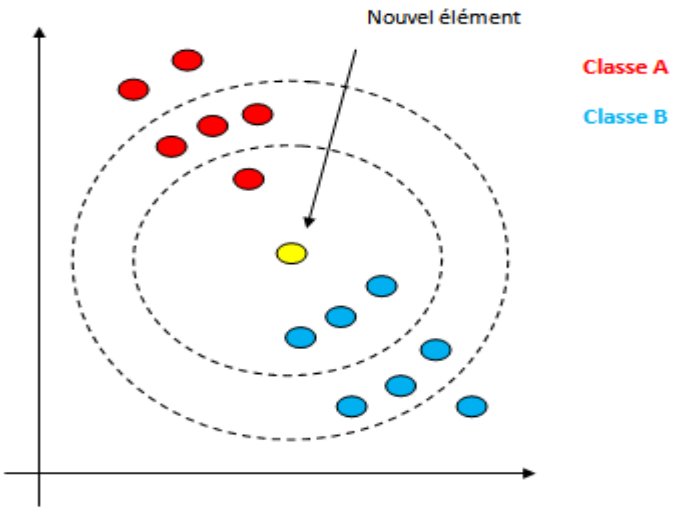


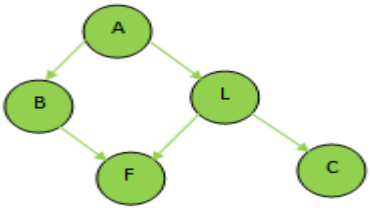
Figure 2 : Exemple K-plus proches voisins.

3.4.4 Réseaux bayésiens

Un réseau bayésien (6) est une représentation des connaissances sous forme de graphe acyclique. Les nœuds représentent les variables aléatoires et les arcs représentent les relations (si possible) causale[6] entre ces variables, l'absence d'arc signifie une indépendance conditionnelle. La figure ci-dessous illustre un exemple de réseau bayésien, nous prenons la viriable B son parent est le A et son successeur est le F, comme il n'existe pas d'arc reliant le B aux autres variables (C'est-à-dire un arc allant de B vers les autres variables), alors il existe une indépendance conditionnelle entre la varibale B,C,L et A. La même méthode est appliquée aux autres variables.

[6]Un lien statistique qui affirme qu'une variable agit sur une autre.

Le réseau bayésien représente toutes les relations entre les variables. Il est ainsi le meilleur outil pour la prise de décisions conditionnées par plusieurs facteurs complexes.



Nœuds	Parents	Indépendances conditionnelles
B	{A}	I (B, {L, C} | {A})
C	{L}	I (C, {A, B, F} | {L})
F	{B, L}	I (F, {A, C} | {B, L})
L	{A}	I (L, {B} |{A})

Figure 3 : Exemple réseau bayésien.

3.4.5 Descente de gradient

L'algorithme descente de gradient ou algorithme de la plus forte pente/plus profonde descenteest un algorithme d'optimisation différentiel qui permet de trouver la meilleure valeur des paramètres d'apprentissage en utilisant la fonction de coûts. Il permet d'indiquer à un modèle donné la direction à suivre dans la courbe tracée en fonction de l'un des paramètres d'apprentissage et une fonction objective. Cela se réalise en calculant la dérivée première de cette fonction et en trouvant de ce fait le vecteur jacobien inverse.Résultant pour le guider dans la recherche de telle sorte à avoir la borne minimale locale ou globale.

3.4.6 Reseaux de neurones artificiels

Un réseau de neurones artificiel est système qui s'inspire du fonctionnement des neurones biologique, qui consiste en un nombre de neurones qui communiquent entre pour réaliser une tâche précise. Ce réseau prend corps dans un ordinateur sous forme d'algorithme. Il capable de se modifier lui-même en fonction des résultats de ses actions, ce qui permet l'apprentissage et la résolution de problèmes sans algorithme, donc sans programmation classique.

Un réseau de neurone artificiel se compose de trois couches élémentaires avec un nombre bien définis de neurones, elles sont reliées entre elles par des connexions pondérées (voir la figure ci-dessous).

1. La couche d'entrée *(Input layer)* : reçoit les entrées d'information brutes provenant de l'environnement.
2. La couche cachée *(Hidden layer)* : traite les données reçus par la couche d'entrée pour extraire les informations importantes.

3. La couche de sortie *(Output layer)* : produit le résultat final.

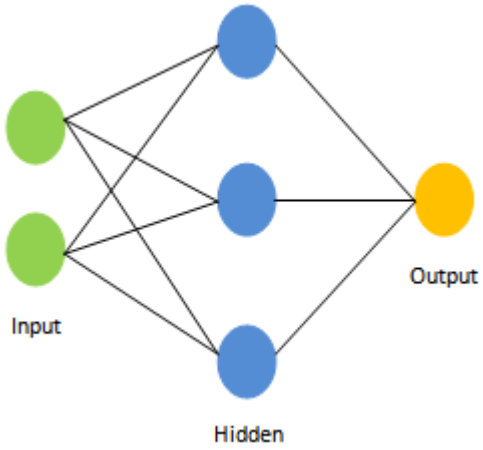


Figure 4 : Exemple réseaux de neurones.

3.5 Application de l'apprentissage automatique

L'apprentissage automatique se répand et se développe rapidement qu'il conquit un grand nombre de secteurs. Récemment L'apprentissage automatique a permis à bon nombre de domaine de franchir un grand pas en avant, en passant par le domaine médicale avec le système d'aide a la décision en analyse d'échographies pour les grosses extra utérine réalisée par le LIMICS[7] et l'hôpital Trousseau ; la cyber-sécurité avec la plateforme Sure Sense réalisée par Deep Instinct [8]qui permet de détecter les virus pour le PC HP Elite Book 800 G6 ; la finance avec les solutions contre les fraudes bancaires proposées par Ercom Analytics[9] ; la robotique avec Dactyl la main robotisée réalisée par des chercheur OpenAI[10] qui a mis 4 minutes pour réussir un Rubik's Cube ; ou l'astronomie avec Morpheus développée par l'UC Santa Cruz [11], capable d'analyser automatiquement les images astronomique pour identifier les astres et les galaxies,… Ces exemples relatent l'évolution spectaculaire de l'apprentissage automatique, et nous donne un aperçue sur un avenir prometteur.

4 Conclusion

Dans ce chapitre nous avons présenté les grandes lignes de l'apprentissage automatique. Nous avons exploré son fonctionnement, et relaté ses types, puis nous avons

[7] Laboratoire d'Informatique Médicale et d'Ingénierie des Connaissances en e-Santé.
[8] Compagnie de cyber-sécurité.
[9] Entreprise spécialisée en sécurité numérique.
[10] Société de recherche et de déploiement d'intelligence artificielle.
[11] University of California Santa Cruz.

exposé ses différentes techniques. Enfin nous avons énoncé quelques-unes de ses récentes applications.

Bien que l'apprentissage automatique soit puissant, sa capacité d'apprentissage est limitée, et pour anticiper l'évolution de la valeur des crypto-monnaies et ainsi établir des stratégies de *trading* efficaces, l'apprentissage automatique fait appel àl'apprentissage profond avec les réseaux de neurones artificiels. Ainsi le chapitre suivant est consacré aux réseaux de neurones artificiels, nous exposons plus de détails sur cette technologie, ses types, son fonctionnement, son architecture.

Chapitre 2: Les réseaux de neurones artificiels

1 Introduction

Certains domaines exigent la manipulation d'un grand nombre de données et une grande précision de calcul comme le *trading*, mais les algorithmes classiques ne peuvent pas répondre à ces exigences.

En effet, l'apprentissage limité des algorithmes classiques de l'apprentissage automatique, représentent un vrai défi pour les problèmes complexes, ainsi l'apprentissage profond vient palier ce problème. Quelque soit la quantité de données acquise, les systèmes d'apprentissage profond peuvent améliorer leurs performances grâce aux réseaux de neurones artificiels. En accédant à d'avantage de données, la machine acquiert assez d'expérience pour être utilisée à des taches spécifiques.

Dans ce chapitre nous présentons des généralités sur les réseaux de neurones. Par la suite nous exposons les types d'entrainement des réseaux de neurones. Puis nous explorons les différentes architectures des réseaux neuronaux. Enfin, nous concluons avec l'entrainement d'un model de réseau de neurones.

2 Généralités sur les réseaux de neurones artificiels

2.1 Neurone formel

Un neurone formel appelé aussi neurone artificiel est l'élément minimal d'un réseau de neurones artificiels. C'est une représentation mathématique et informatique prenant les principes de fonctionnement d'un neurone biologique[1]. Il est donc avant tout un operateur qui a partir d'entrees X_i et de poids W_i produit une valeur numérique Y en sortie. (7)

2.2 Fonctionnement

Un neurone formel traite m entrées qui sont externes (si c'est les premiers neurones) ou des sorties d'autres neurones (si c'est des neurones intermédiaires). Le traitement donne en sortie une valeur numérique Y, le processus de traitement consiste en quatre étapes principales :

[1]Il s'agit d'une cellule dans le système nerveux qui reçois, traite et communique des informations, il est à l'origine du fonctionnement du cerveau humain.

1- La pondération : effectue un produit entre chaque entrée $X_i$et le poids qui lui est associéW_i.
2- La somme : une somme pondérée des coefficients synaptiques $\sum_{i=1}^{m}(X_i * W_i)$.
3- L'ajout du biais : un biais (valeur numérique) est ajouté à la somme précédente $\theta + \sum_{i=1}^{m}(X_i * W_i)$.
4- Activation : le résultat obtenu précédemmentest transformé avec une fonction d'activation φ (voir section 2.3) : $\varphi(\theta + \sum_{i=1}^{m}(X_i * W_i))$.

2.3 Fonction d'activation

La fonction d'activation est une fonction mathématique utilisee au niveau de chaque neurone. Elle peut être catégorisé selon plusieurs facteurs comme : la non-linéarité, l'étendue (plage activation), la dérivée monotone, la linéarité. La fonction d'activation la plus utilisee est la non-linéaire, elle offre une grande puissance au réseau de neurones et une capacité à généraliser des problèmes très complexes avec des résultats efficaces. (8)

Le tableau ci-dessous présente les fonctions d'activations les plus utilisées :

Fonction	Equation	Description
RELU	$F(X) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	- Fonction non-linéaire, très éfficace, sa principale limite est la disparition du gradient.
Leaky RELU	$F(X) = \begin{cases} 0.01 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	- Optimisation de RELU pour prévenir la perte du gradient.
ELU	$F(X) = \begin{cases} a(e^x - 1) si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	- Alternative solide à RELU, elle produit des sorties négatives, son seul inconvénient est la possibilité d'avoir des sorties dans $[0; \infty[$pour$x > 0$.
Sigmoïde	$F(X) = \frac{1}{1 + e^{-x}}$	- Fonction non-linéaire, très utilisée pour les problèmes de classification binaire, les sorties sont dans une plage (0 ; 1), mais elle est sensible à la disparition du gradient, et couteuse lors de l'exécution.

Tanh	$F(X) = \frac{2}{1+e^{-2x}} - 1$	- Fonction non-linéaire, valeurs normalisées entre -1 et 1, sortie centrée sur 0, elle a les même inconvenients que la fonction sigmoïde.
Softmax	$F(X)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} / j \in \{1, \dots, k\}$	- Fonction non-linéaire, généralization de sigmoïde, valeurs normalisées entre 0 et 1, utilisée pour les problemes de classification.

Tableau 1: Fonctions d'activation.

3 Types d'entrainements des réseaux de neurones artificiels

Il existe deux types d'entrainement des réseaux de neurones, selon le problème étudié : avec rétro-propagation ou sans rétro-propagation.

3.1 Entrainement Acyclique (Feed Forward Propagation)

Un réseau de neurones non bouclé ou bien réseau statique est représenté par une structure multicouche ou l'information circule des entrées vers la sortie sans retour en arrière. Si on représente ce réseau par un graphe ou les neurones représentent des nœuds et les connexions des arêtes, on aura un graphe acyclique. La figure ci-dessous illustre un réseau de neurone non bouclé.

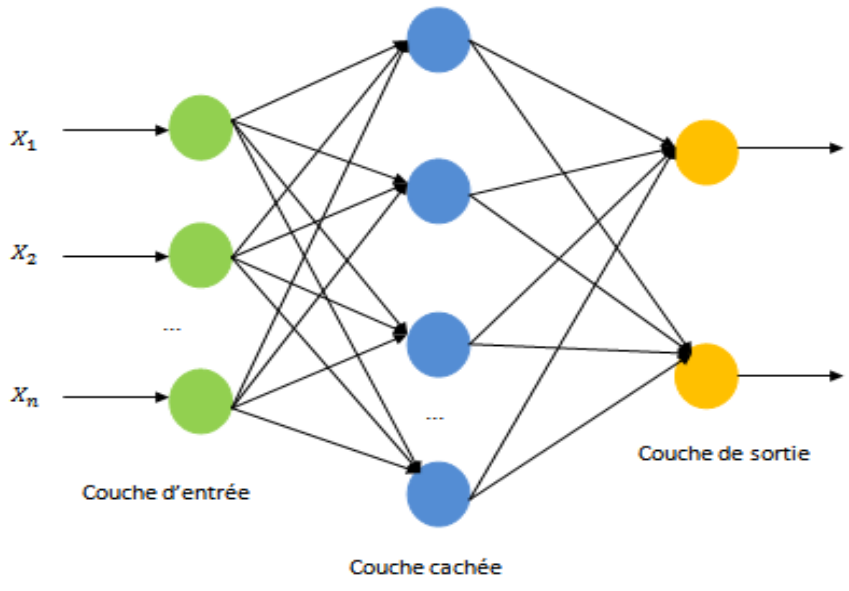


Figure 5: Réseau de neurones non bouclé.

3.2 Entrainement Cyclique *(Backward Propagation)*

Un réseau de neurones bouclé ou bien réseau dynamique / récurrent, permet des connexions arbitraires entre les neurones de toutes les couches du réseau. En suivant le sens de connexions, il existe au moins un chemin qui mène vers un neurone initial, ce chemin représente un cycle.

Les réseaux de neurones bouclés peuvent donc avoir plusieurs topologies de connexions comprenant notamment des boucles qui peuvent amener aux entrées la valeur des sorties. Cela nécessite l'intervention de la notion du temps à prendre explicitement en considération. La figure ci-dessous illustre un réseau de neurone bouclé.

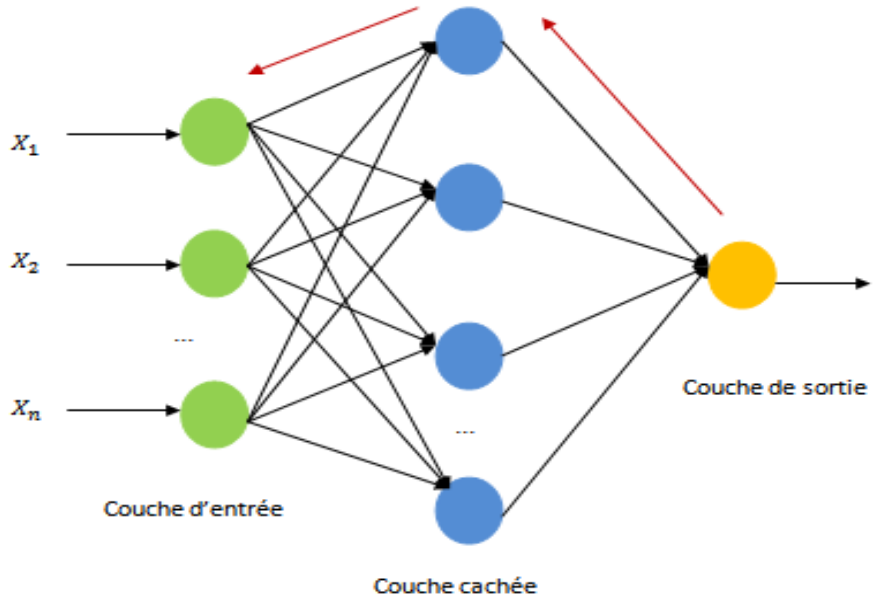


Figure 6: Réseau de neurones bouclé.

4 Architectures des réseaux de neurones artificiels

Il existe plusieurs architectures de réseau de neurones classifiées selon le type du problème traité. Dans ce qui suit, nous présentons les architectures les plus connues.

4.1 Les perceptrons

1- Perceptron monocouche : est un réseau simple, il ne se compose que d'une couche d'entrée et d'une couche de sortie. Généralement utilisé pour les systèmes visuels (reconnaissance de forme), pour la classification et pour la résolution d'opérations logiques simples (ET/OU). Sa principale limite est qu'il ne peut résoudre que des problèmes linéairement séparables (problèmes de discrimination).

2- Perceptron multicouches : est un réseau acyclique, il se compose d'une couche d'entrée, une couche de sortie et plusieurs couches cachées entre les deux, utilisé dans des problèmes complexes notamment dans l'apprentissage supervisé où il est considéré comme un bon classificateur.

4.2 Architecture complètement connectée

Dans l'architecture à connexion complète, chaque neurone est connecté aux autres neurones du réseau. Il s'agit d'un réseau à une seule couche où toutes les unités sont interconnectées. Cette architecture est également appelée modèle de *Hopfield*, elle est principalement utilisée pour l'entrepôt des connaissances, mais aussi pour la résolution de problèmes d'optimisation.

4.3 Architecture convolutionnelle

Les réseaux de neurones convolutionnels *(CNN : Convolutionnal Neural Network)*(9) sont des réseaux de neurones acycliques (non bouclé), correspondent d'une manière simplifié à l'empilement de perceptron multicouches, chacun traite une partie de l'information totale. Ils entrent dans la plupart des cas dans le cadre de l'apprentissage par transfert où le résultat de sortie peut être réutilisé comme l'entrée d'un autre modèle. Les réseaux de neurones convolutifs sont utilisés pour la reconnaissance d'images, vidéos et aussi pour le traitement naturel du langage. La figure suivante représente l'architecture convolutionnelle et les différentes couches qui la composent. (10)

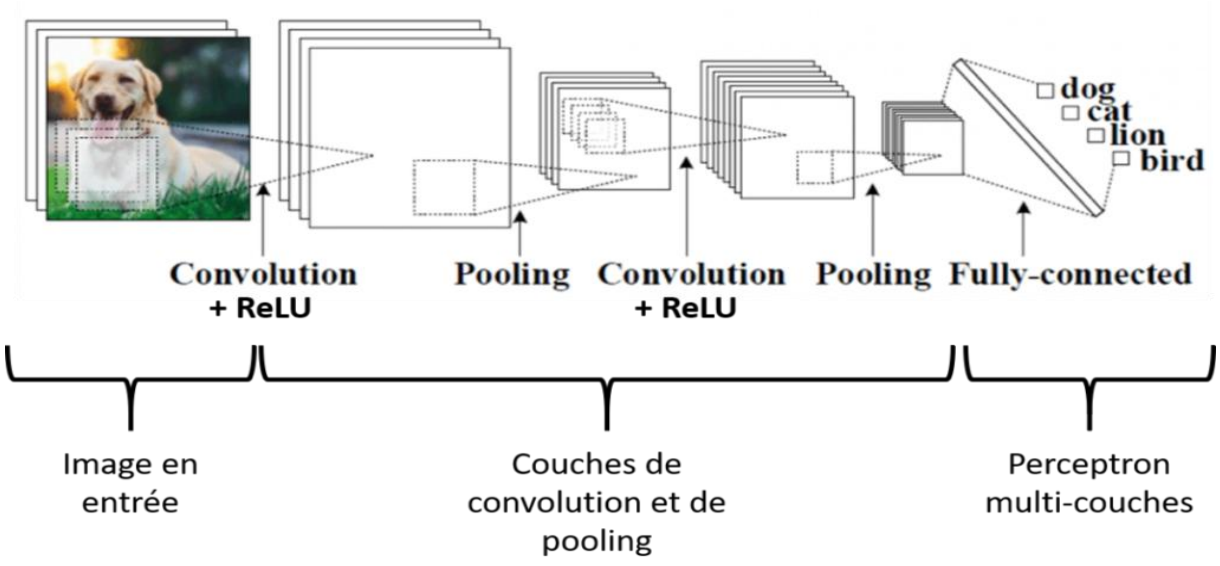


Figure 7 : Architecture convolutionnelle.

4.3.1 Couche convolutionnelle *(CONV)*

La couche de convolution est la première couche et la composante clé des réseaux de neurones convolutifs. Son but est de repérer la présence d'un ensemble de *features* dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une fenêtre représentant la *feature* sur l'image, et de calculer le produit de convolution entre la *feature* et chaque portion de l'image balayée. Les filtres correspondent exactement aux *features* que l'on souhaite retrouver dans les images. Ainsi, on obtient pour chaque paire (image, filtre) une carte d'activation, ou *feature map*, qui nous indique où se situent les *features* dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la *feature*.

4.3.2 Couche *Pooling*

La couche*pooling* est souvent placée entre deux couches de convolution : elle reçoit en entrée plusieurs *feature maps*, et applique à chacune d'entre elles l'opération de *pooling*,qui consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Ainsi, on obtient en sortie le même nombre de *feature maps* qu'en entrée, mais celles-ci sont bien plus petites.

La couche de *pooling* permet de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage.

4.3.3 Couche de correction RELU

La couche de correction RELU comme son nom l'indique se base sur la fonction réelle non-linéaire RELU : $[RELU(x) = \max(0, x)]$pour remplacer toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

4.3.4 Couche *fully-connected*

La couche *fully-connected* constitue toujours la dernière couche d'un réseau de neurones, convolutifs ou non – elle n'est donc pas caractéristique d'un *CNN*. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche *fully-connected* permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, oùN est le nombre de classes dans notre problème de

classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

4.4 Architecture récurrente

Les réseaux de neurones récurrents (*RNN : Recurrent Neural Network)* (11) sont des réseaux boucles, ils traitent l'information en cycle. Ces cycles traitent plusieurs fois l'information en la renvoyant à chaque fois au sein du même réseau, ce qui permet l'auto-entretien du réseau. Ils sont utilises pour l'apprentissage de données séquentiellement organisées et dépendantes les unes des autres, comme la reconnaissance automatique de la parole, la traduction automatique, la reconnaissance de l'écriture manuscrite etc. Les réseaux de neurones récurrents se heurtent au problème de la disparition du gradient, de nouvelles architecture répondent à ce problème en particulier les réseaux *Long Short Term Memory (LSTM)* ou *Gated Recurrent Unit (GRU)*.

4.4.1 LSTM

Les réseaux de longue mémoire à court terme *(LSTM : Long Short Term Memory)* (12) sont une version amélirée du réseau neuronal récurrent, capable d'apprendre les dépendances à long terme. Ils sont construits d'une cellule mémoire,ainsi ils peuvent lire, écrire et supprimer des informations, et de trois portes pondérées dont les poids sont appris durant l'apprentissage, ce qui détermine la bonne connexion entre elles.Ces portes sont : porte d'entrée (surveille les conditions pour qu'une nouvelle entrée soit sauvegardée dans la cellule mémoire), porte de sortie (détermine les conditions pour qu'une valeur en mémoire détermine la valeur de sortie) et la porte d'oubli (détermine les contraintes pour qu'une information reste en mémoire).

4.4.2 GRU

Les unités récurrentes gated *(GRU : Gated Recurrent Unit)*(13) sont une version améliorée du réseau neuronal récurrent, et une architecture simplifiée de *LSTM*. Contrairement aux*LSTM*, *GRU* ne possèdent pas de porte de sortie, cette dernière est remplacée par une porte de réinitialisation, ainsi ils donnent moins de contrôle sur les entrées. Cependant le nombre réduit de paramètres par rapport aux*LSTM,* permettent aux*GRU* une exécution rapide et un model plus général.

5 Entrainement d'un model

La phase de l'entrainement est le cœur du processus d'apprentissage automatique, ou une grande partie des données sera utilisé pour entrainer le modèle pour apprendre.

5.1 Fonction Coûts

La fonction coût est une composante important lors de l'entrainement d'un model. Elle offre un aperçu sur la performance du model en comparant les résultats attendus et les résultats prédits obtenus, en cas d'incohérence les résultats prédits sont corrigés par la suite. Le choix de la fonction coût dépend du problème traité ainsi la fonction coût de régression est utilisée pour les problèmes d'apprentissage supervisé de type régression, et la fonction coût de classification est utilisée pour les problèmes dont la sortie prédite est une probabilité et la sortie attendue est une valeur binaire. (14)

5.2 Algorithme d'optimisation

Un algorithme d'optimisation ou d'apprentissage est un algorithme qui permet de mettre à jour les paramètres d'un réseau de neurones et de les faire converger vers une valeur optimale. Les algorithmes d'optimisation sont classés en deux catégories :

5.2.1 Algorithme à taux d'apprentissage constant

Dans ce type d'algorithme tous les hyper-paramètres propres à l'optimisateur sont initialisés et appliqués sur l'ensemble des données durant l'entrainement.

- Algorithme du gradient stochastique : est une méthode itérative de la descente de gradient, utilisée pour la minimization d'une function objective, qui est ecrite comme une somme de fonctions differentiables. La progression de cette méthode s'avère lente et est souvent sensible et confrontéà des problèmes tels que la disparition du gradient ou le minimum local.

5.2.2 Algorithme à taux d'apprentissage adaptatif

Dans ce type d'algorithme tous les hyper-paramètres sont initialisés au début mais se modifient au fur et à mesure de la recherche pour avoir des résultats optimaux.

- ADAM : est un algorithme iteratif, qui permet de mettre à jour les poids de réseau itératif en fonction des données d'apprentissage.

- ADAMAX : C'est une variante d'Adam basée sur la norme de l'infini. Les paramètres par défaut suivent ceux fournis dans le document. Adamax est parfois supérieur à l'adam, notamment dans les modèles avec encastrement.
- SGD: Gradient descent (with momentum) optimizer.
- RMSprope : Maintenir une moyenne mobile (actualisée) du carré des gradients et diviser le gradient par la racine de cette moyenne. La version centrée maintient en outre une moyenne mobile des gradients, et utilise cette moyenne pour estimer la variance.

6 Conclusion

Dans ce chapitre nous avons présenté les réseaux de neurone artificiels. Nous avons exploré le fonctionnement d'un réseau de neurones. Puis, nous avons présenté ses différents types d'entrainement. Apres quoi, nous avons exposé les différentes architectures des réseaux de neurones. Enfin, nous avons conclu par l'entrainement d'un model.

Dans le prochain chapitre nous présentons l'application de l'apprentissage automatique dans notre projet, à savoir nos modèles de réseaux de neurones. En plus, nous analysons les architectures ainsi que leurs configurations. Enfin, nous expliquons les métriques implémentées qui nous permettent d'avoir une vue globale sur les performances de nos modèles.

Chapitre 3: Modèles proposés

1 Introduction

La popularité des crypto-monnaies a explosé en 2017 en raison de plusieurs mois consécutifs de croissance exponentielle de leur capitalisation boursière. Ainsi plusieurs projets de prédiction du prix des crypto monnaies ont été lancés. Cependant, Les marchés de la crypto-monnaie évolue sans cesse d'un rythme tres élevé (24/24h – 7/7j), ce qui les rend difficiles à résoudre ou à prédir.

Souvent, de nombreux traders de la crypto-monnaie (et négociants en actions) placent leurs transactions en se basant uniquement sur une analyse technique hors contexte, des ouï-dires d'autrui, des sentiments instinctifs ou émotions, ce qui entraîne souvent des transactions non rentables. Même si les défis du marché sont sérieux, ils présentent également de nombreuses opportunités uniques. La création d'une stratégie robuste basée sur des règles et soutenue par des données de backtest, qu'elles soient fondamentales ou techniques (ou les deux), peut aider les traders à générer des résultats plus cohérents et à éliminer les émotions pour prendre de meilleures décisions de trading avec plus de confiance.

D'un autre côté, l'intelligence artificielle est au cœur des enjeux de notre société. Elle a réussi à prédire les cours boursiers grâce à une multitude de modèles de séries temporelles.

Les modèles de l'apprentissage automatique et les réseaux de neurones, peuvent certainement donner des informations sur l'avenir de la crypto-monnaie, tel que la tendance générale et la direction dans laquelle les prix évoluent. Dans ce chapitre, nous présentons des concepts basiques, nous citons quelquestypes de trading et les strategies algorithmiques. Puis nous présentons une déscription des données exploitées par nos modéles. Ensuite nous décrivons l'approche suivie pour l'élaboration de nos modèles. Enfin, nous présentons nos quatres modéles d'apprentissage automatique permettant de prédir l'avenir de la crypto-monnaie.

2 Stratégies de trading dans la litérature

Selon Irène Aldridge (une chercheuse en finance quantitative et *Big Data* de renommée internationale), nous pouvons distinguer 3 familles de *trading* automatique :

1. Le *trading* électronique : famille classique, représente le passage des ordres par téléphone, comme le faisaient les anciens traders, ou bien les nouvelles méthodes d'exécution d'ordres par ordinateurs.
2. Le *trading* algorithmique : le concept est l'optimisation des ordres qui seront exécutés par les traders. En réalité, le trader reste maître de la décision d'investir ou pas, cependant la technique d'analyse est automatisée.
3. Le *trading* haute fréquence (THF) : dans ce cas, la décision et l'exécution totalement automatique de passage d'ordre sur le marché se fait à très haute fréquence (achat-vente en quelques secondes).

La première famille a été une évolution générale des systèmes de trading. L'avènement des réseaux et des ordinateurs fait qu'elle est aujourd'hui indispensable à toute communication inter établissements. Puis le trading algorithmique s'est rapidement généralisé au fur et à mesure que les investissements devenaient de plus en plus importants. Il a toutefois le désavantage de faire appel à des hommes et nécessite donc des moyens financiers importants pour gérer ces ressources humaines.

C'est pourquoi notre intérêt portera sur les deux autres familles. La dernière famille est particulièrement intéressante car d'une part, les techniques actuelles sont encore jeunes (elle est utilisée à New York que depuis 10 ans), et d'autre part, elle nécessite des moyens techniques abordables si l'on souhaite en avoir une utilisation à petite échelle. C'est pourquoi nous porterons notre étude sur cette famille-là. Du fait qu'elle représente le nouveau domaine de compétition des entreprises et institutions financières, l'information disponible sur les techniques de trading haute fréquence est trés peu divulguée.

Voici un graphe (15) représentant les raisons principales pour lesquelles les institutions financières ont recours au THF :

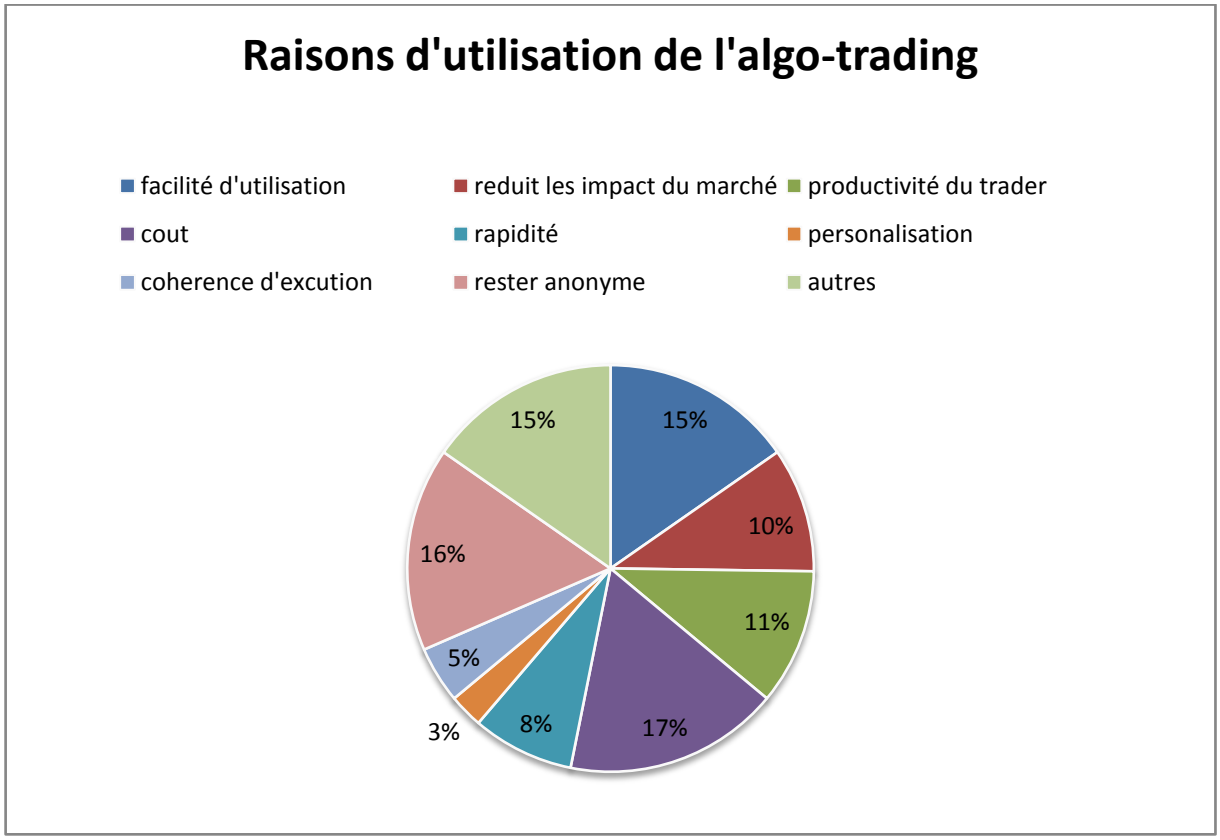


Figure 8 : Raisons d'utilisation de l'algo-trading d'après "The TRADE Annual Algorithmic Survey".

Ce graphique nous montre que les principales raisons d'utilisation de l'algo-trade sont par ordre d'intérêt:

- La facilité d'utilisation.
- Le coût. Comme nous le disions, aucune ressource humaine n'est à gérer, donc une importante réduction des coûts.
- Rester anonyme. Cela donne la possibilité aux gros investisseurs de cacher leurs intentions d'investissement en réalisant de multiples exécutions de petites tailles au cours d'une période.

2.1 Les stratégies de trading algorithmiques

Nous présentons ici les stratégies de trading les plus utilisées par les traders et les systèmes d'algo-trading. Ces stratégies vont nous guider pour l'élaboration de notre système.

- Suivre la tendance : Cette stratégie profite d'une tendance du marché pour prendre position et profiter d'un mouvement régulier à la hausse ou à la baisse d'un cours.

- Échange de paire : ici, le système va profiter d'un écart entre deux actions qui sont potentiellement corrélées. Par exemple, une chute du cours de *Coca-Cola* va avoir une incidence sur le cours de *Pepsi*. Ainsi, lors de la chute du cours de *Coca-Cola*, nous pouvons prévoir une chute moins importante sur le cours de *Pepsi* car il existe une corrélation de l'activité entre ces deux entreprises.
- Arbitrage : Autre cas, Profiter du temps de latence entre deux places boursières qui cotent une même action. Un ajustement du prix de l'action est automatiquement réalisé entre ces deux places boursières. Par exemple, l'action BNP Paribas chute sur la place boursière de Paris mais n'a pas encore été impactée à la bourse de Tokyo. Il est donc possible de profiter de ce temps de latence pour acheter à la baisse l'action BNP Paribas à Tokyo.
- Révision significative : Cette stratégie part du principe qu'une action a un prix moyen. Quand le prix de l'action est bien supérieur à ce prix moyen suite à une annonce forte sur le marché par exemple, alors il y a sur-achat ou sur-vente. Cette stratégie est très proche de l'analyse technique et fondamentale. Par exemple, sur les 14 derniers jours, la moyenne du cours de l'action d'une entreprise de vente est de 20€. Si aujourd'hui elle est à 21€ on peut conclure qu'il y a survente et que le prix va se corriger dans les jours qui viennent.
- Scalping : Cette méthode d'achat vente est élaborée pour faire des profits sur de petites différences. Pour un cours donné, on achète un peu en dessous du cours. Si une contrepartie est prête à vendre à ce prix-là, on revend tout de suite et l'on peut gagner ainsi la différence entre le prix d'achat et le prix de vente. Cela nécessite de réaliser la transaction en moins d'une minute.

2.2 Les bots de trading

Un bot de trading de crypto-monnaie est un logiciel informatique qui utilise différents indicateurs de marché pour détecter les tendances des prix, puis exécute automatiquement les transactions en fonction de paramètres prédéterminés au nom du commerçant.

Les programmes de trading algorithmique ont toujours été utilisés par les *hedge funds*[1] (des sociétés d'investissement qui investissent l'argent de leurs clients dans des

[1] *hedge funds : u*n fonds d'investissement non indexé sur la bourse. ... Ce sont des acteurs très importants du marché car ils sont capables d'investir sur des produits à risques.

investissements alternatifs, soit pour battre le marché, soit pour se protéger contre les changements imprévus du marché) ; et les bureaux de négociation exclusifs sur les marchés des matières premières, des devises et des actions.

Cependant, les robots de trading sont récemment devenus de nouveaux financements, car de plus en plus de traders de crypto utilisent des robots pour exécuter leurs stratégies de trading.

En termes simples, les robots de trading permettent de définir des paramètres spécifiques généralement basés sur une analyse technique et des indicateurs de sentiment auxquels le bot exécute des transactions pour l'utilisateur. Par exemple, si trois des indicateurs techniques choisis affichent un signal d'achat, le bot achètera automatiquement et lorsque les indicateurs afficheront un signal de vente, il se vendra.

Les robots de trading sont donc des outils que les traders existants les plus expérimentés qui peuvent être utilisé pour exécuter leurs stratégies de trading sans avoir à appuyer physiquement sur le bouton d'achat et de vente. Ils aident également à éliminer les émotions de l'équation car ils sont basés uniquement sur l'action des prix.

Actuellement, il existe plusieurs robots de trading de crypto-monnaie, nous avons choisit ceux qui ont influencé le plus nos choix pour notre application et qui sont aussi parmi les plus populaire, qui sont : Gekko , CryptoTrader, zenbot et Haasbot . (16)

Dans le tableau ci-dessous nous allons faire une comparaison entre ces bots connus.

Nom	Description	Ans d'activité	Facilité d'utilisation /10	Nombre d'echanges pris en charge pour le trading en direct
GEKKO	* Gekko est un bot de trading de crypto-monnaie populaire qui permet aux utilisateurs de backtester et d'exécuter des stratégies de trading de base. * Le bot peut être téléchargé gratuitement sur GitHub, où son développeur met régulièrement à jour le logiciel pour améliorer ses	7	5	7

	fonctionnalités.			
CryptoTrader	* CryptoTrader est un autre bot de trading Bitcoin populaire basé sur le cloud qui permet aux traders de développer, de backtester et de déployer des stratégies de trading basées sur des indicateurs techniques. * La plate-forme fournit également aux utilisateurs un marché sur lequel ils peuvent acheter et vendre des stratégies de trading rentables.	4	4	9
Zenbot	* Le Zenbot est un autre bot de trading Bitcoin gratuit et open-source qui permet aux utilisateurs de développer, de backtester et de déployer gratuitement des stratégies de trading Bitcoin automatisées. * Zenbot prend en charge une gamme de devises numériques et offre aux utilisateurs la possibilité d'exécuter des transactions à haute fréquence.	4	4	9
Haasbot	* Haasbot est sans doute le bot de trading Bitcoin le plus populaire du marché. * Haasbot permet aux traders de combiner une gamme d'indicateurs techniques pour développer, tester et déployer des stratégies de trading de crypto-monnaie simples et avancées. * Le Haasbot possède également des fonctionnalités d'arbitrage de change et de gestion des risques telles que des limites stop-loss.	4	5	7

Tableau 2 : Comparaison entre les bots de trading les plus connus.

En dehors de ces bots, il existe de nombreux autres algorithmes et modèles de séries chronologiques à expérimenter. Nous venons de faire un bref aperçu des les stratégies financière, les bots de trading automatique.

2.3 Les algorithmes de trading

2.3.1 ARMA: *Auto Regressive Moving Average*

Modèle ou processus de prévision dans lequel les méthodes d'analyse d'autorégression et de moyenne mobile sont appliquées: ARMA.

Il suppose que la série chronologique est stationnaire, fluctue plus ou moins uniformément autour d'une moyenne invariante dans le temps. Les séries non stationnaires doivent être différenciées une ou plusieurs fois pour atteindre la stationnarité. Les modèles ARMA sont considérés comme inappropriés pour l'analyse d'impact ou pour les données qui intègrent des «chocs» aléatoires.

2.3.2 ARIMA: *Auto Regressive Integrated Moving Average*

Le modèle ARIMA est une généralisation d'ARMA qui sont soit utilisés pour étudier les données, soit pour prédire / prévoir les valeurs.

Nous préférons utiliser ARIMA plutôt qu'ARMA car dans le cas de données non stationnaires, la partie intégrée d'ARIMA peut aider à mieux prédire et étudier les modèles. Comme nos données sont entièrement des données de séries chronologiques, le modèle ARIMA pourrait être le meilleur modèle pour prédire les valeurs de manière linéaire (en tenant compte de la moyenne mobile).

ARIMA est spécifié par ces trois paramètres d'ordre: (p, d, q)[2]. Le processus d'ajustement d'un modèle ARIMA est parfois appelé méthode *Box-Jenkins*[3]. (17)

3 Les données utilisées

3.1 Source de données

Les algorithmes d'apprentissage automatique et les réseaux de neurones nécessitent de grandes quantités de données d'entraînement pour révéler leur pleine puissance. La plupart des ensembles de données de trading cryptographiques actuellement disponibles ont une faible résolution temporelle, ne sont pas gratuits ou se concentrent uniquement sur un nombre limité de paires de devises.

[2]P est l'ordre d'autoregressive, d est l'ordre d'integration ou nombre de différences et q est l'ordre de moyenne mobile.

[3] Box Jenkins :sont les principaux modèles de séries temporelles.

Heureusement, il existe de nombreuses plate-forme et ressources à utiliser pour l'obtention des données historiques sur les crypto-monnaies. Certaines de ces ressources permettent le téléchargement manuel de fichiers CSV, d'autres fournissent des API à connecter au code.

Nos données ont été téléchargées sur KAGGLE[4]. Elle contient les données de trading historiques (OHLC) de plus de 400 paires de trading à 1 minute de résolution remontant jusqu'à l'année 2013. Elles ont été collectées à partir de la bourse Bitfinex[5]. (18)

3.2 Description des données

3.2.1 Pair trading

En crypto-monnaie, le terme « paires de trading » décrit un échange ou une sorte de conversion entre un type crypto-monnaie et un autre. Par exemple : ETH/BTC

Avec ETH/BTC, l'opération est : l'achat de l'Ethereum avec Bitcoin, ou bien ventre de l'Ethereum pour du Bitcoin.

La crypto-monnaie peut donc être échangée pour un autre type de crypto-monnaie mais peut aussi être échangée contre l'argent. Exemple : USD/BTC.

3.2.2 OHLC (Open-High-Low-Close)

L'OHLC designe les prix d'ouverture, de clôture, les prix hauts et bas pour chaque période. Les OHLC sont utiles car ils montrent les quatre principaux points de données sur une période, le prix de clôture étant considéré comme le plus important par de nombreux traders, il est considéré comme référence pour le prix de la crypto-monnaie dans une unité de temps.

3.2.3 Exploration des données

Le dataset contient 430 fichiers csv contenants plusieurs lignes sous la forme suivante :

4 KAGGLE : une plateforme web organisant des compétitions en science des données. https://www.kaggle.com/.

5 Bitfinix : une bourse de crypto-monnaie détenue et exploitée par iFinex .https://www.bitfinex.com/

:

time	open	close	high	low	volume
2013-04-01 00:07:00	93.25	93.30	93.30	93.25	93.300000
2013-04-01 00:08:00	100.00	100.00	100.00	100.00	93.300000
2013-04-01 00:09:00	93.30	93.30	93.30	93.30	33.676862
2013-04-01 00:10:00	93.30	93.30	93.30	93.30	33.676862
2013-04-01 00:11:00	93.35	93.47	93.47	93.35	20.000000

Figure 9 : Contenu du fichier btc/usd {btcusd.head ()}

Afin de donner une vision plus claire des données, nous comparons deux crypto monnaies, le Bitcoin BTC et l'Etherium ETH, et ceci par l'intermédiaire du Dollar Américain USD. Pour ce faire nous explorons deux de nos fichiers qui sont BTC/USD ETH/USD.

Après avoir appliqué notre algorithme de visualisation sur ces fichiers, nous avons les figures ci-dessous qui représentent le prix de clôture de la BTC/USD et de l'ensemble de données ETH/USD. Le graphique représente donc un moyen adéquat pour faire une étude sur/entre ces deux crypto-monnaies.

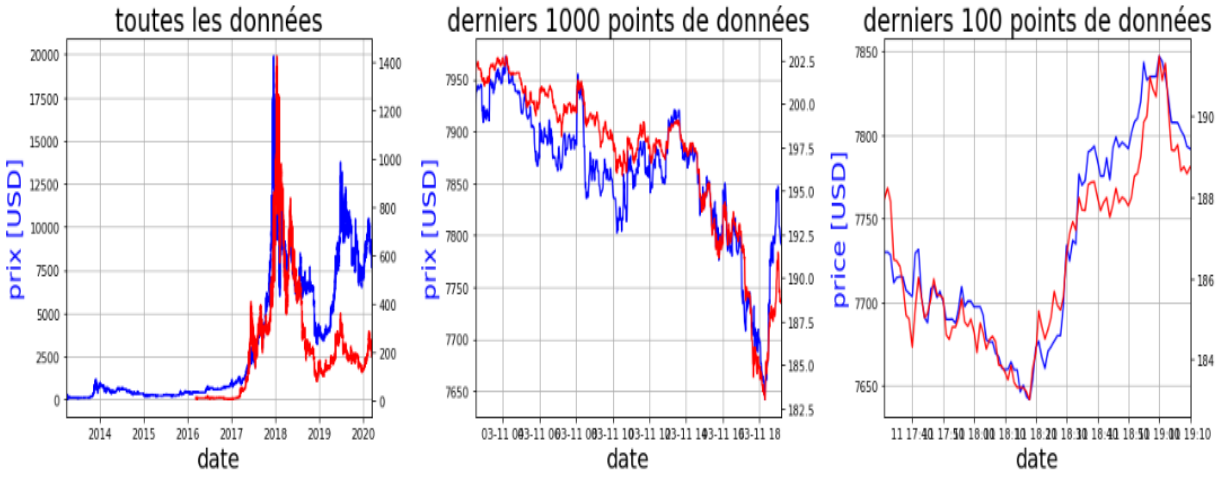


Figure 10 : Visualisation des données (BTC/ETH).

La figure ci-dessus représente une visualisation de tout l'historique des prix de Bitcoin et Etherium en fonction de Dollar.

- La courbe bleue est la visualisation de prix de Bitcoin en fonction de Dollar américain.
- La courbe rouge est la visualisation de prix d'Etherium en fonction de Dollar américain.

Comme nous pouvons voir sur ces graphiques ci-dessus, La comparaison directe des données de BTC et l'ETH sont difficile sur ces graphiques, même avec les tentatives de zoomer sur les mille et cent derniers points de données. Les résultats peuvent être convainquant mais dans le cas ou nous allons essayer de comparer plus que trois fichier (trois crypto-monnaies). Là, notre graphique sera illisible.

Alors, si nous voulons comparer plus de graphiques cryptographiques, i.e explorer plus que deux fichiers, nous devons penser à autre chose. Une méthode courante consiste à calculer les rendements du journal de chaque paire de devises au fil du temps.

Ci-dessous, nous définissons une fonction pour calculer les retours de log. On l'a appliqué sur quatre de nos fichier, qui sont : XRP/USD, LTC/USD, BTC/USD et ETH/USD, c'est-à-dire que nous comparons Ripple, Litecoin, Bitcoin et Etherium.

Afin de rendre les graphiques plus rentables, passant à une autre méthode assez courante consiste à calculer les rendements logarithmiques de chaque paire de devises au fil du temps. Nous allons l'appliquer sur quatre fichiers de notre dataset On obtient le premier graphique de la figure suivante, pour améliorer plus nous allons refaire le même graphique sur les données calculées avec une moyenne de 120 lignes on obtient le deuxième graphique, qui est assez lisible et qui permet de faire une comparaison entre les quatre crypto-monnaie.

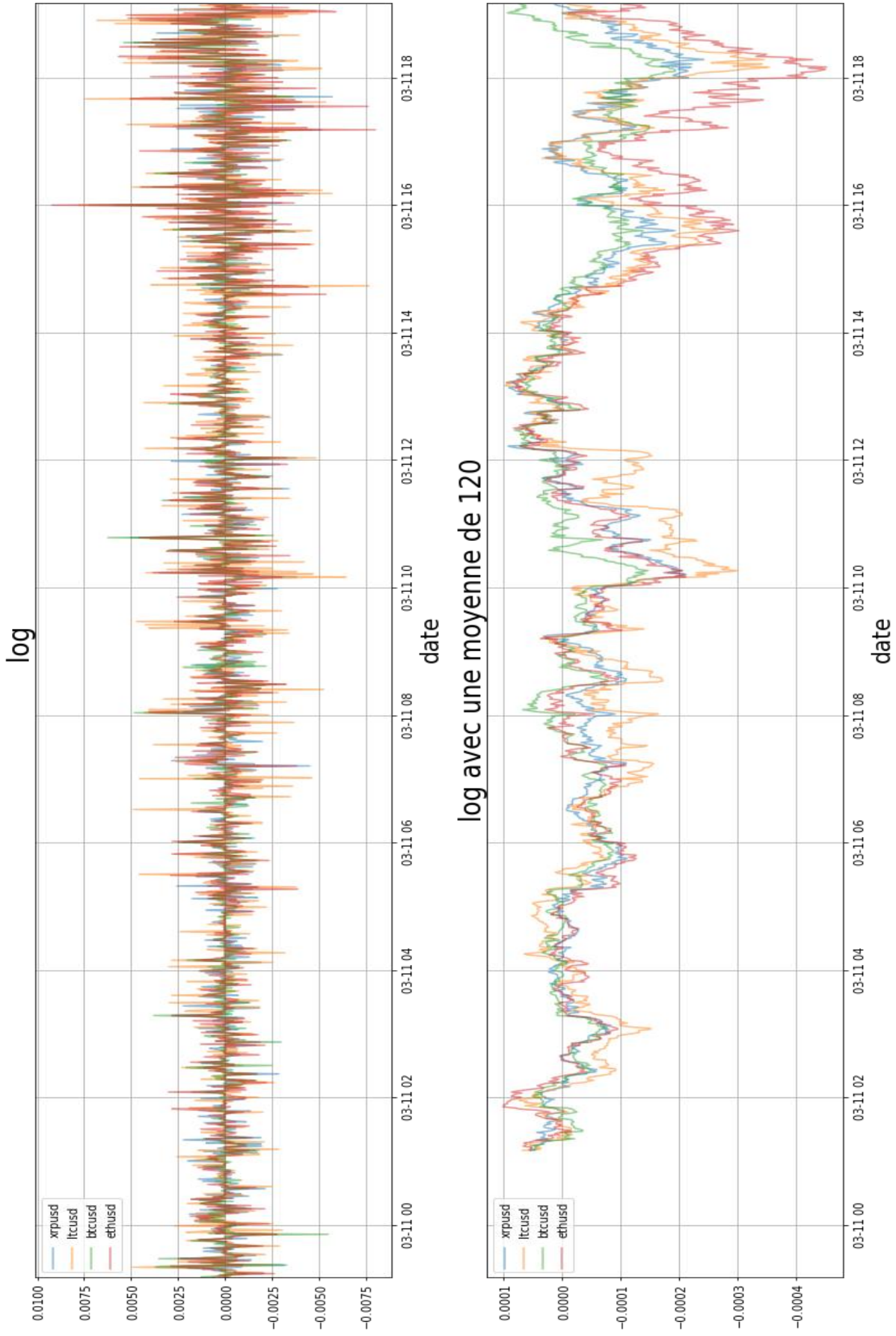


Figure 11 : Visualisation des données sur 5 fichiers csv.

4 Approche suivie

Au cours du développement de notre application, plusieurs pistes nous ont dirigés et influencés dans nos choix et nos idées, entre l'apprentissage automatique et le deep learning, et entre les différents types et modèles des réseaux de neurones.

4.1 Analyse et choix d'algorithme

Le *Deep Learning* ou apprentissage profond est un sous-ensemble d'apprentissage automatique, mais la principale différence entre les deux est que le *Deep Learning* utilise des Réseaux de Neurones qui donnent à la machine la possibilité de s'entraîner. L'apprentissage automatique nécessite des algorithmes avec des paramètres (les entrées) définis par l'utilisateur ou l'ingénieur pour entraîner la machine. En d'autres termes, l'apprentissage automatique nécessite plus d'ajustements et de corrections pratiques que le *Deep Learning*.

Mais ceci ne veut pas dire que le *Deep Learning* est meilleur que l'apprentissage automatique ; car cela dépend entièrement du problème. Dans certains cas, les modèles d'apprentissage automatique traditionnels sont bien meilleurs que les réseaux de neurones complexes. Un tel cas que nous avons trouvé est la formation et l'ajustement de données tabulaires pour la classification.

D'un autre côté, nous gardons à l'esprit que seules, les machines peuvent traiter des données en dehors du contexte social. Le sens, l'économie, la politique, les facteurs sociaux et les émotions, les sentiments ou l'intuition restent négligés dans les décisions commerciales de l'IA. Cela peut sembler un inconvénient non menaçant quand il s'agit d'opérer avec des chiffres, mais en regardant certaines des situations dramatiques passées de l'économie mondiale, parfois ce cas représente une solution idéal, et on devrait éliminer toute intervention humaine.

4.2 La modélisation

Un réseau de neurones classique permet de gérer les cas ou les expériences sont indépendantes les unes des autres. Lorsque les expériences sont sous la forme de séquences temporelles, une nouvelle structure a été inventée : les réseaux de neurones récurrents. Les RNA permettent d'utiliser la sortie du modèle comme nouvelle entrée pour le même modèle. Le processus peut être répété indéfiniment.

Une limitation sérieuse des RNA est l'incapacité de capturer les dépendances à long terme dans une séquence (par exemple, y a-t-il une dépendance entre le prix actuel et celui d'il y a 2 semaines?). Une façon de gérer la situation consiste à utiliser une variante de mémoire à court terme (LSTM) de RNA.Le comportement par défaut de LSTM est de mémoriser les informations pendant des périodes prolongées.

Ce sont donc les modéles adéquats pour traiter notre ensemble de données, caractérisés par leurs ordres chronologiques.

5 Modèles proposés

Nos propositions ont été faites en étroite relation avec l'implémentation. Ainsi, l'architecture globale de tous nos modèles est décrite par la suite. Après avoir entraîné et tester plus d'une centaine de modèles, nous avons sélectionné et proposé ces modèles parmi ceux qui semblent être les meilleurs de par leurs entraînement et leurs performances lors des tests. Sachant que le but de nos modèles est de prédire le « close ».

Tous nos modèles choisis se basent sur les RNA, et partagent les mêmes critères sauf leurs architectures.

L'apprentissage d'un RNA est similaire à la formation d'un réseau neuronal traditionnel. Nous utilisons également l'algorithme de rétro-propagation (Backpropagation ou BP en anglais), -permettant le calcul et la mise à jour des poids-, mais avec une petite torsion. Car les paramètres sont partagés tout au cours de notre réseau, le gradient (qui optimise le calcul des poids et de la loss) à chaque sortie dépend non seulement des calculs actuels, mais également des précédents.

Par exemple, pour calculer le gradient à t=4 nous aurions besoin de rétro-propagation des 3 étapes précédentes ainsi qu'un résumé des gradients. C'est ce qu'on appelle la rétro-propagation dans le temps (BPTT[6]). Cela n'a pas encore beaucoup de sens. Nous devons être conscients du fait que les RNA formés avec BPTT ont des difficultés à apprendre les dépendances à long terme; en raison de ce que l'on appelle le problème du gradient de *vanishing / exploding,* qui survient lorsque le gradient, du fait du passage sur plusieurs

[6] BPTT : Backpropagation through time, La rétropropagation dans le temps est une technique basée sur un gradient pour entraîner certains types de réseaux de neurones récurrents.

étapes de BP, s'approche de zero et il devient donc impossible d'optimiser encore plus le modèle et par conséquent de continuer l'apprentissage.

Comme nous l'avons déjà dit, Il existe des mécanismes pour traiter ces problèmes, et certains types de RNA (comme les LSTM) ont été spécifiquement conçus pour les contourner. Du fait de leur mémoire courte qui leur permet de stocker les valeurs issus des étapes précédentes (et donc stock une sorte d'historique qui leur permet de récupèrées en quelques sortes les anciennes valeurs du gradient avant qu'elle ne s'approche de zéro) et donc nous aide à éviter le *vanishing gradient problem*.

5.1 Modèle RNA 1

L'architecture de ce modèle se compose de 97 neurones répartis sur 3 couches de LSTM.

La première couche LSTM est composée de 64 neurones, qui prennent en entrée des données de dimension (1,1).

La deuxième couche LSTM est composée de 32 neuones

La troisième couche Dense est composée de 1 neuone et correspond à la couche de restitution du résulat, puisque nous souhaitons prédire la valeur de close qui corresponds à une valeur numérique, alors nous terminons par un neuronne.

Ce modèle offre les meilleurs résultats et ce sont ceux que nous avons présenté dans ce mémoire.

Pour la compilation de ce modèle nous choisissons les paramètres suivants :

1. Fonction de loss : *Mean Square Error* qui correspond à la moyenne des erreurs quadratiques commises par notre modèle. Elle est très utile car nos données sont numériques et donc permet de bien calculer la distance entre nos points d'entrès et de sorties et donc c'est un bon discriminateur. (19) (20) (21)
2. Optimizer : L'optimizer est l'algorithme qui va définir comment la valeur de la loss va évoluer au fur et à mesure de notre apprentissage, c'est-à-dire comment notre modèle va optimiser la recherche de la valeur minimal de notre loss.
   Ici, on choisit l'algorithme de descente de gradient stochastique (une variante de la descente de gradient) qui consiste à calculer la valeur de la pente vers laquelle le gradient va descendre et se rapprocher du minimum voulue (c'est-à-dire la

valeur 0), c'est d'ailleurs ce qui explique le fait que le *vanishing gradient* corresponds à un gradient qui tend vers 0 et donc qui ne s'améliore plus car proche du minimum (qui est en réalité un minimum local).

5.2 Modèle RNA 2

Ce modèle est plus basique et plus simple que le précédent, il correspond à un réseau de neuronne se composant de 13 neurones répartis sur 3 couches.

La première couche se compose 8 neurones et prend en entrée des données de dimension (1,1).

La deuxième couche LSTM est composée de 4 neurones.

La troisième couche Dense est composée de 1 neurone et correspond à la couche de restitution du résulat, puisque nous souhaitons prédire la valeur de close qui corresponds à une valeur numérique, alors nous terminons par un neuronne.

Ce neuronne offre des résultats moindres car pas complexe et donc n'apprend pas assez les caractéristiques des données.

Pour la compilation de ce modèle nous avons choisis les paramètres suivants :

1. Fonction de loss : *Mean Square Error* qui corresponds à la moyenne des erreurs quadratiques commises par notre modèle. Elle est très utile car nos données sont numériques et donc permet de bien calculer la distance entre nos points d'entrès et de sorties et donc c'est un bon discriminateur.
2. Optimizer : Ici nous avons décidé de choisir l'algorithme de RMSPROP qui se base sur un calcul des valeurs du gradient (les valeurs recente sont normalisé selon les anciennes) et ceci forme un moyen d'optimiser nos calculs.
   Cependant, les résultats obtenus avec cet algorithme sont insatisfaisants car ne répondent pas aux exigences voulu en matière de précision de prédiction.

5.3 Modèle RNA 3

L'architecture de ce modèle se compose de 260 neurones répartis sur les trois premières couches de neurones de type LSTM et une couche de sortie d'un neurone Dense.

Il est clair en vue de la complexité de ce modèle (un nombre élevé de neurones) que l'apprentissage de notre modèle sera médiocre car nos données ne sont pas assez complexes afin d'identifier et de calculer tout les poids de notre modèle avec la BP.

Observations :

En ce qui concerne l'efficacité des modèles, cela dépend de leur complexité et aussi des données utilisé : Dans notre cas nos données, sont des données à faible dimensions. Plus un modèle est complexe (beaucoup de couche et beaucoup de neurone) plus notre apprentissage risque de tomber sur du surapprentissage, car il apprendra trop parfaitement les caractéristiques de nos données (qui elles ne sont pas assez complexe) et également plus notre modèle est trop simple par rapport à nos données plus l'apprentissage ne sera pas efficace car nous n'aurons pas assez de paramètre afin d'apprendre correctement les caractéristiques de nos données.

Donc il faut toujours faire un compromis entre complexité du modèle et efficacité.

Il existe plusieurs approches, nous pouvons soit pour chaque monnaie construire un model à part et le tester, ou bien, nous allons regrouper toutes les grandes monnaies, créer un model permettant de généraliser sur ces données, ainsi :

- Nous obtenons un modèle général sur plusieurs monnaies ;
- Évitons les couts de calcul et de stockage de plusieurs modèles pour chaque type de monnaie.
- Évitons-le sur apprentissage, puisque nos données sont hétérogènes.

5.4 Etapes de l'apprentissage de notre modèle :

Dans cette section nous allons, étape par étape, explique le cheminement de l'apprentissage de notre modèle

1. Diviser nos données en 3 échantillons, un pour l'entraiînement, un pour la validation et un autre pour le test. L'ensemble d'entraînement va servir à l'entraînement de notre modèle, l'ensemble de validation va servir à valider après chaque époch les résultats obtenus par le modèle et enfin l'ensemble de test comme son nom l'indique, va nous permettre de tester les résultats de notre modèle sur un échantillon supposé nouveau.

2. Nous allons ensuite passer au *préprocessing* des données, ici nous allons procéder au filtrage des données (supressions de fausse valeur, supressions des valeurs vides, des valeurs abérantes…etc), à la transformation de nos données (si nous avons des données catégorielles (entier, chaine de caractère) il est nécessaire d'utiliser des procédées afin de les transformer en valeur numériques adaptable à notre modèle), au redimensionnement de données afin qu'elles puissent être passées en entrée à notre modèle et enfin à toute sorte de modification nécessaire selon notre cas d'usage.
3. Construction de l'architecture de notre modèle. Ici nous choisissons le nombre de couche, le nombre de neuronne ainsi que les fonctions d'activations correspondantes à chaque couche de notre réseau de neurones.
4. Lancement de la phase d'apprentissage en donnant en entrée au modèle les données et en choisissant les paramètres cités plus haut (la fonction de loss et l'optimizer pour le calcul du gradient).
5. Tester les résultats du modèle et vérifier les performances. Si les performances sont satisfaisantes alors nous sauvegardons les poids du modèle, sinon nous répétons ces étapes jusqu'à obtentions de bon résultats

6 Conclusion

Nous avons démontré dans ce chapitre les moyens mis en place par les institutions financières pour améliorer les techniques de trading, que ce soient des techniques classiques ou bien les techniques modernes, qui permettent de détecter automatiquement les opportunités qu'offrent les marchés. Nous avons expliqué les différentes stratégies adoptées, nous avons définit ce qu'est un bot de trading et nous avons comparé entre les bots de trading les plus connus.

Dans le prochain et dernier chapitre, nous offrons une analyse et une étude des outils utilisés pour la conception, l'implémentation et l'évaluation de nos modèles.

Chapitre 4: Implémentation et évaluation des modèles proposés

1 Introduction

Notre projet consiste à réaliser un modèle de prédiction, qui nous permettra d'anticiper les fluctuations du marché des crypto-monnaies, en se basant sur différents types d'architecture de réseaux de neurones.

Ces architectures ont été entièrement développées par nous même et ont été testées sur différentes crypto-monnaies. Notre application développée nous permettra ainsi de mieux comprendre les caractéristiques de ces crypto-monnaies, et d'avoir les résultats pour lesquels ce projet a été conçu. Et ceci, à travers les deux parties les plus importantes d'un projet d'apprentissage automatique, l'« Entrainement » et les « Tests » d'un modèle d'apprentissage automatique.

Dans ce chapitre, nous présentons les démarches suivies du début du projet jusqu'à son déploiement. Pour ce faire, nous commençons par introduire les outils et bibliothéques utilisées tout au long de la phase de développement suivi de son déploiement.Ensuite, nous présentons les différents types d'architectures utilisées pour la prédiction. Enfin, nous présentons les résultats obtenus lors de l'experimentation de notre application.

2 Implémentation

2.1 Outils et environnements de développement

2.1.1 Anaconda

Fondée par Peter Wang et Travis Oliphant en 2012, Anaconda[1] est une distribution gratuite et open source des langages de programmation Python et R pour le calcul scientifique (Science des données, apprentissage automatique…), qui vise à simplifier la gestion des paquets et déploiement.

2.1.2 Jupyter notebook

Jupyter Notebook (anciennement IPython Notebooks) est un environnement de calculinteractif basé sur le web pour la création de documents de bloc-notes Jupyter.

Un document Jupyter Notebook est un document JSON, suivant un schéma versionné, et contenant une liste ordonnée de cellules d'entrée / sortie pouvant contenir du code, du

[1] https://www.anaconda.com/about-us

texte, des mathématiques, des graphiques et du contenu multimédia, se terminant généralement par une extension (.ipynb).

Un bloc note Jupyter peut être converti en plusieurs formats à savoir: HTML, LATEX, PDF, Python, dispositions de présentation, ReStructured Text, Markdown.

2.1.3 TensorBoard

TensorBoard est une suite d'outils de visualisation graphique à temps réel sous forme d'une application web qui permet de lire les données exécutés sur TensorFlow. Il est utilisé comme un débogueur afin d'extraire des informations importantes des données et de suivre les résultats de l'exécution pour aboutir à une performance optimale. (23)

2.2 Bibliothèques logicielles

2.2.1 Tensorflow

Développée par Google en 2015 et publiée sous licence Apache. Tensorflow est une bibliothèque gratuite et open-source de bas niveau écrite en python, utilisée pour les applications d'apprentissage automatique. Son nom se compose de deux parties « tensor » qui désigne la représentation vectorielle des données sous forme de tenseurs 8 qui obéissent aux transformations et aux opérations algébriques, et « flow » qui désigne le flux des données représenté par ces tenseurs. Contrairement à d'autres bibliothèques numériques destinées à être utilisées dans l'apprentissage profond comme Théano, Tensor Flow a été conçu pour une utilisation à la fois dans la recherche et le développement et dans les systèmes de production, notamment Rank Brain dans la recherche Google et le projet amusant DeepDream (projet initié par Google dans le cadre de l'intelligence artificielle et plus précisément de la reconnaissance de formes et de l'apprentissage automatique). Tensor Flow est multi-plateforme, elle peut fonctionner sur des systèmes à processeur unique, des GPU ainsi que des appareils mobiles et des systèmes distribués à grande échelle de centaines de machines. (24)

2.2.2 Keras

Keras est une bibliothèque de haut niveau écrite en python, utilisée pour le développement d'applications d'apprentissage automatique. Elle est capable de fonctionner au-dessus de TensorFlow, R, Théano. Elle est la bibliothèque la plus utilisée due à la facilité d'utilisation et d'extension, la modularité et facilité de composition et la convivialité de son

interface simple et cohérente. Keras supporte la majorité des réseaux de neurones (Récurrents, Dense, Convolutionnels), et comporte deux types de modèles (Séquentiel et fonctionnel). (25)

2.2.3 Numpy

Numpy (Numerical Python) est une bibliothèque open-source destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que les fonctions, opérations arithmétiques, logiques et statistiques opérant sur ces tableaux. La bibliothèque Numpy permet de sauvegarder les données dans un fichier portant l'extension (. npy). (26)

2.2.4 Pandas

Développée par Wes McKinney en 2008, Pandas[2] dont le nom est dérivé de «panel data » est une bibliothèque écrite en python, principalement utilisée pour l'analyse de données.Ellepermet d'importer des données à partir de divers formats de fichiers tels que des valeurs séparées par des virgules, JSON, SQL, Microsoft Excel, elle permet également diverses opérations de manipulation de données telles que la fusion, le remodelage, la sélection, ainsi que le nettoyage de données et les fonctionnalités de traitement de données.

2.3 Cycle de développement

2.3.1 Méthode suivie

Dans le cadre de la bonne conduite de notre projet et du développement de notre application de prédiction des prix des crypto-monnaies, nous avons suivi une des normes utilisées dans la gestion de projet d'apprentissage automatique.

[2] https://pandas.pydata.org/about/index.html

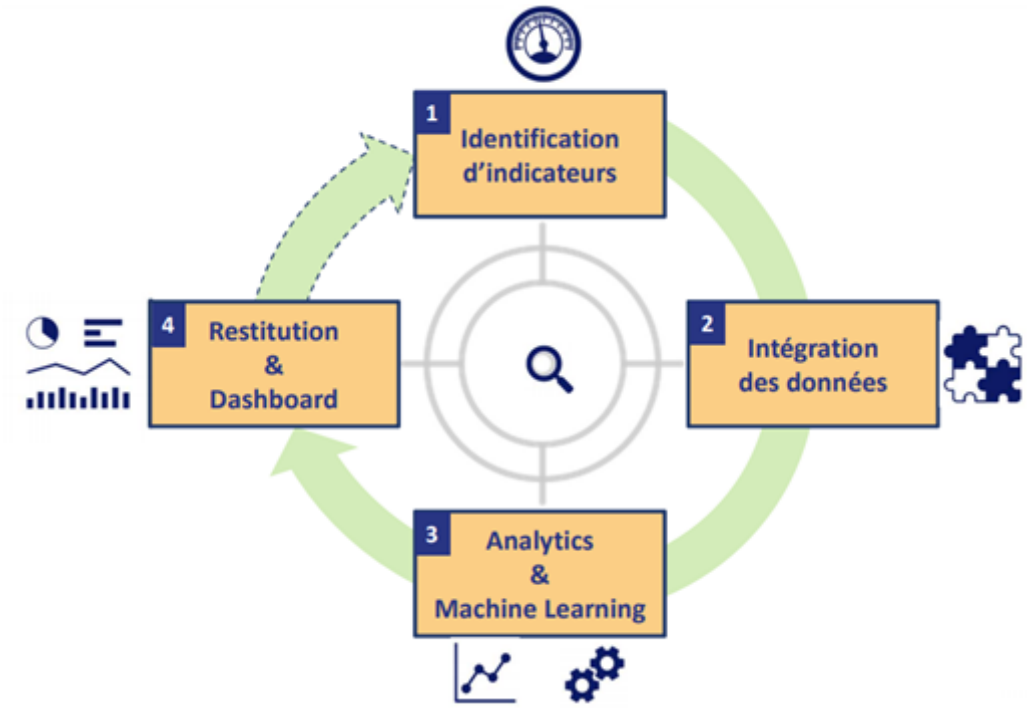


Figure 12 : Processus cyclique.

Cette méthode se divise en 4 grandes étapes différentes et forme un processus cyclique qui ne s'arrêtera que si les résultats obtenues sont assez satisfaisants :

1. Identification d'indicateur : Cette partie est parmi les plus importantes de tout projet de Data Science. C'est sur cette partie que nous allons sélectionner les données qui seront utilisées dans notre modèle. En effet, nous allons sélectionner parmi nos données les variables explicatives qui pourront le mieux expliquer et apporter de l'information sur les caractéristiques des fluctuations du marché des crypto-monnaies.Cette première partie s'effectue à travers l'analyse d'échantillons significatifs de données sur des outils statistiques (nous l'avons fait avec python, mais cela aurait pu être fait par des outils de visualisation comme Tableau par exemple). A partir de là, les variables sont choisies.
2. Intégration des données : Cette partie consiste à intégrer les données de façon à ce qu'elles s'adaptent le mieux au modèle qu'on aura choisi. Dans cette partie, les données sont téléchargées, importées sur des data frames, traitées et modifiées (conversion timestamp en Dates par exemple). Ces étapes sont nécessaires afin que les données puissent s'adapter à notre modèle.
3. Veille scientifique sur les méthodes d'apprentissage automatique : Cette partie consiste en la recherche de solutions qui pourront s'adapter à notre problème. C'est une des étapes essentielles, car il faut choisir la méthode la plus optimale

parmi celles qui pourraient s'adapter à notre problème. Plusieurs méthodes sont testées et évaluées avant d'être sélectionnées.

4. Restitution et visualisation des résultats : Dans cette partie nous allons étudier les résultats obtenus par notre modèle. Si ces résultats s'avèrent être concluants et positifs alors nous pouvons nous arrêter là et conserver les poids de notre meilleur modèle, et si ils ne s'avéraient pas concluants, alors nous allons re-boucler et tenter de choisir de meilleurs indicateurs et de refaire le processus jusqu'à obtention de résultat satisfaisant.

2.3.2 Conception

Nous présentons dans cette étape l'idée de base et l'objectif principal de notre projet à savoir une application de prédiction des prix de la crypto-monnaie basé sur un modèle de réseau de neurones.

Nous faisons une estimation de la faisabilité du projet, c'est-à-dire s'il est techniquement réalisable en fonction du temps et de la complexité de l'application.Nous avons donc analysé la complexité des données à notre disposition, leur volumétrie et leur type de codage (entier, numérique, chaîne de caractères).

En effet, chacune de ces caractéristiques sont importantes à la bonne réalisation de notre projet :

1. La complexité des données est un indicateur important afin de savoir si les données contiennent assez d'informations sur les fluctuations du marché des crypto-monnaies, afin d'être correctement apprises par un réseau de neurones. Car, s'il s'avère que ces données ne sont pas utiles, peu importe le réseau construit, peu importe son efficacité et sa complexité, nous ne pourrons jamais construire une application fiable, robuste et efficace, car tout se base sur les données.
2. La volumétrie des données est un indicateur important en ce qui concerne l'efficacité et la faisabilité de notre solution.
    a. **Efficacité** : Peu importe les données dont on dispose, si le nombre d'information (de bloc de ligne de données) importante contenant dans ces données, si elles ne sont pas assez importante ou assez diversifié afin de couvrir plusieurs situation possible (dans notre cas plusieurs variantes

de fluctuations de marché) notre modèle ne pourra pas se généraliser correctement, et un modèle non généralisé ne sera, ni efficace, ni robuste, et risque même de tomber sur du sur-apprentissage.

b. **Faisabilité** : Si nos données sont trop volumineuses, la puissance de notre machine peut ne pas supporter la complexité de ces données afin d'effectuer les calculs nécessaires et toutes les epochs d'apprentissages de notre modèle.

Nous serions tenter de juste réduire le nombre de données utilisées pour notre apprentissage, cependant, ce n'est pas aussi simple, car dans la plupart des cas, les données dont nous disposons mêmes si elles sont nombreuses, elles ne sont pas de qualité et donc il est essentiel d'utiliser le maximum possible afin de récupérer toutes les informations disponibles.

3. Le codage des données est un indicateur important en ce qui concerne la faisabilité de notre solution. En effet, les modèles de DeepLearning utilisent en exclusivité que des données numériques en entrée, car le calcul des poids entres les différentes couches ne peut se faire qu'avec des valeurs numériques.De ce fait, les données doivent toutes être transformées en des valeurs numériques, si nous avons des données en catégories (chaîne de caractères, identifiants ...), elles doivent passer par un processus qu'on appelle One Hot Encoding, qui nous permettra de les transformer sans pertes d'informations.

Après avoir effectué l'étude des données, et avoir montré leurs utilités pour la création de notre solution, nous avons par la suite étudié les différents types de modèles qui pourraient s'appliquer à nos données. En effet, Il existe plusieurs types de modèles possibles, des RNA, CNN, LSTM ... . Chaque modèle a ses caractéristiques et son utilité.

Après plusieurs tests et recherches, nous avons choisi de nous baser sur l'architecture LSTM, car grâce à son principe de mémoire courte sauvegardée entre chaque couche, c'est le modèle qui s'adapte le mieux à notre problème et à nos données.

2.3.3 Déploiement

Notre application se décompose en cinq fichiers Python + les fichiers des données (2.2 GO) regroupant les données des crypto-monnaies + des images.

Chacun de ces fichiers a un rôle spécifique, que ce soit pour la partie affichage ou pour la partie traitement de données, pre-processing des données, visualisation des données et application d'algorithme d'apprentissage automatique.

Pour l'exécution de notre application, il suffit d'ouvrir un terminal de commande, d'avoir python installé sur la machine, d'avoir les bibliothèques Dash, Plotly (visualisation), Numpy,Pandas (processing des données) ainsi que TensorFlow (apprentissage automatique), de se positionner sur le dossier contenant le code source, et exécuter la commande «$ python index.py». (Voir la figure ci-dessous)

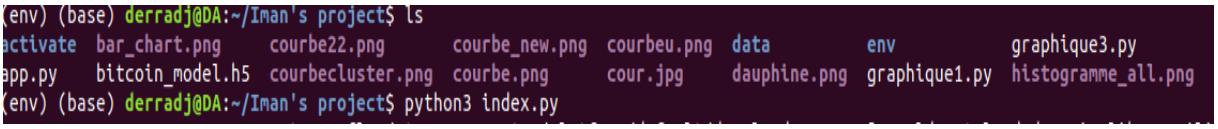


Figure 13 : Lancement de l'application.

Dès l'exécution de la commande, vous devriez vous retrouver avec un affichage comme celui ci-dessous. (Utilisez l'adresse http://127.0.0.1:8050/ sur votre navigateur web afin de le lancer l'application)

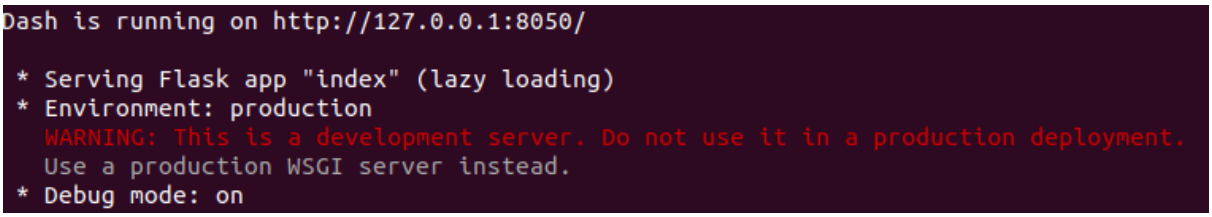


Figure 14 : Execution de la commande.

3 Utilisation de l'application Dash

Dans cette partie, nous présenterons l'application développée et comment l'utiliser.

3.1 Visualisation

Dans cette section, nous pouvons visualiser, de façon dynamique, les fluctuations de chaque crypto-monnaies selon les différentes monnaies de bases (Eur, USD ...etc)

Sur les deux boutons déroulants, nous pourrons sélectionner la monnaies que nous souhaitons afficher par rapport à une ou plusieurs monnaies de référence et inversement.

Par exemple, sur cette image, nous pouvons visualiser les fluctuations du BTC par rapport à l'Eur.

Figure 15 : Visualisation BTC - EUR.

3.2 Apprentissage automatique

Dans cette section, nous pouvons visualiser, de façon dynamique, les prédictions sur 30 jours (paramètre que nous pouvons modifier) de chaque crypto-monnaies selon les différentes monnaies de bases (Eur, USD ...etc)

Sur les deux boutons déroulants, nous pourrons sélectionner la monnaie que nous souhaitons prédire par rapport à une ou plusieurs monnaies de référence et inversement.

Par exemple, sur cette image, nous pouvons visualiser les prédictions du BTC par rapport à l'Eur.

Figure 16 : Prédiction BTC - EUR.

4 Expérimentations et résultats

4.1 Environnement d'entraînement et de test

Le modèle de réseau de neurones artificiels a besoin de ressources matérielles de qualité afin d'accroître ses performances. Dans notre cas, nous avons entraîné le modèle sur un ordinateur portable hp 205 ayant un processeur Intel(R) Core TM i5 – 4210U CPU avec une fréquence de 1.70 GHz 2.40 GHz. L'ordinateur utilise le système d'exploitation Windows10 64 bits avec notamment 6 Go de RAM et un disque dur SSD d'une capacité de 297 GB.

En ce qui concerne l'environnement de test, nous avons utilisé Jupiter décrit précédemment. Nous avons opté pour cet éditeur par rapport à sa facilité d'utilisation et les configurations offertes pour les développeurs par anaconda.

4.2 Métrique utilisée

Afin de sélectionner la métrique la plus optimale pour l'apprentissage de notre réseau de neurones, nous sommes passés par plusieurs tests de plusieurs métriques, pour enfin finir par choisir une fonction de loss et un optimiseur de calcul pour la minimisation de notre loss.

```
# Compile the model
model.compile(loss='mean_squared_error', optimizer='sgd')
model.fit(X_train, y_train, epochs=300, batch_size=8, shuffle=False)
```

Figure 17 : Métriques d'évaluations utilisées.

4.2.1 Fonction loss *(Mean Square Error)*

Elle représente la métrique la plus simple et la plus courante dans l'évaluation desproblèmes supervisés de régression, car elle est très efficace quand nous avons à faire à ce genre de données numériques.

En effet, le calcul de l'erreur quadratique offre une bonne distance d'appréhension de la différence entre les données prédites par notre modèle et les données originelles.

En plus de cela, elle joue un rôle dans la compilation de notre modèle, car facile à implémenter et peu coûteuse en calcul comparé aux autres types de données.

Plus cette valeur est élevée, plus le modèle est mauvais. Le résultat fourni est toujours positif, car nous calculons les erreurs de prévision individuelles avant de les additionner, mais serait nul pour un modèle parfait.

4.2.2 Optimiseur

Après plusieurs tests de différents optimizer (Adam, RMSPROP, SGD), nous avons décidé de sélectionner SGD car elle offre les meilleurs résultats, même si la vitesse de convergence est plus lente par rapport à celle obtenue grâce à Adam, néanmoins l'efficacité et la robustesse des résultats étaient meilleurs avec SGD qu'avec Adam.

L'optimizer SGD utilise la descente de gradient stochastique afin de retrouver les valeurs minimales retournées par la fonction de Loss utilisée (dans notre cas MSE). C'est une méthode très utilisée car elle est simple à implémenter et simple à comprendre mathématiquement.

4.2.3 Métriques d'évaluation

L'évaluation de l'algorithme d'apprentissage est une étape essentielle dans tout projet d'apprentissage automatique. Nous avons utilisé la fonction « *Mean Square Error* » *(MSE)* comme fonction de coût car c'est la plus utilisee dans les problemes de regression. Elle

permet de calculer l'erreur quadratique entre les données prédites et celles attendues. En plus, elle permet de calculer le gradient plus facilement que les autres metriques.

De plus, nous avons des métriques afin de suivre l'état et la progrssion du modèle dans la phase de validation. Par defaut, la fonction coût utilisée est aussi intégrée comme métrique d'évaluation.Nous avons utilisé la métrique RMSE qui estreprésentée comme étant la racine carré de MSE. Elle est utilisée généralement pour montreret détecter les grandes déviations et accorde un poids relativement élevé aux erreursimportantes. Cela signifie que le RMSE devrait être plus utile lorsque des erreurs importantessont particulièrement indésirables.

Aussi, nous avons appliqué la métrique MAE qui calcule la moyenne des différences absolues des erreurs et cela dans le but de pénaliser les grandes erreurs. Elle est caractérisée par sa non-ambigüité ce qui fait d'elle la métrique la plus utilisée dans tous les projetsd'apprentissage automatique.

Enfin, Métrique d'évaluation ne peut pas être utilisée comme fonction de coût de notre modèle à cause de sa non-dérivabilité. Une notion primordiale pour toutes fonctions de coûtvu qu'elle sera utilisée lors de la mise à jour des poids où nous aurons besoin de calculer ladérivée de la fonction d'erreur.

4.3 Entraînement

La phase d'apprentissage de notre modèle est la phase la plus importante dans la création de notre solution, en effet, un modèle qui aurait mal appris les caractéristiques de nos données ou qui n'aurait pas pu généraliser les informations contenues dans nos données, ne nous aurait été d'aucune utilité.

Il est alors primordial de vérifier le bon déroulement de la phase d'apprentissage, cela peut se faire à travers plusieurs approches qui nous permettrons d'éviter les erreurs classiques.

1. Eviter le sur apprentissage : Le surapprentissage est un phénomène qui survient souvent lors de la phase d'apprentissage de notre modèle, ce phénomène peut être détecté à partir des résultats obtenues par notre modèle, mais aussi à travers les statistiques obtenues lors de cette phase (Les valeurs de loss, de validation loss).

a. **Validation** : cette étape consiste à effectuer des tests sur nos modèles après chaque étape d'apprentissage, à chaque étape nous divisons les données en deux sous-ensembles, "ensemble d'entrainement" et "ensemble de validation". Tester les prédictions de notre modèle sur l'ensemble de validation nous permet de vérifier que l'apprentissage peut se généraliser et donc que le sur-apprentissage n'est pas assez important.
b. **Méthode de régularisation** : il existe un paramètre de régularisation dans le calcul de chaque couche, ce paramètre nous permet de réduire le sur-apprentissage de notre modèle en forçant tous les poids de chaque couche de neurones à ne pas apprendre toutes l'information et à ne pas être à nulle (ce qui se passe dans le sur-apprentissage, et que la majorité de l'information soit concentré que sur quelques poids et que les autres soient négligés, la régularisation viens empêcher cela.
c. **Dropout et Batch Normalisation** : ces deux sont utilisées entre de deux couches de neurones, elles permettent comme pour la régularisation, se réduire le sur apprentissage, en bloquant quelques neurones pour empêcher que ces derniers apprennent toutes l'information.
    i. Exemple Dropout : en utilisant la fonction Dropout (0.4), où 0.4 est la probabilité d'utiliser la fonction sur un neurone, supposons que nous avons une couche a 3 neurones, la fonction va à chaque étape d'apprentissage et pour chaque neurones avec une probabilité de 0.4, bloquer le neurone (geler le neurone) afin qu'il ne puisse pas mettre ces poids à jours sur cette étape.
d. **Nombre d'epoch** : Ce paramètre représente le nombre de fois que nous souhaitons effectuer une passe sur nos données pour l'apprentissage, plus ce nombre est élevé plus les paramètres et caractéristiques apprises par notre modèle vont se rapprocher le plus des caractéristiques de nos données d'entraînement. Si ce nombre est plus élevé que ce qu'il ne faut, alors nous tomberons sur du sur-apprentissage, il est alors optimisé à travers plusieurs tests de valeurs différentes.

2. Optimiser l'apprentissage : L'apprentissage peut être optimisé grâce à :

a. **La normalisation des données** : les données utilisées en entrées de notre modèle doivent être normalisées, i.e des données centrées réduites sur la moyenne et l'ecart-type.
b. **Dropout** : En plus d'éviter le sur-apprentissage cette méthode va nous permettre d'optimiser le rendement de notre modèle.
c. **Choisir le bon nombre d'epoch** : afin d'optimiser le nombre de paramètres appris par notre modèle, le nombre d'epoch est une variable importante.

4.4 Résultats obtenus

Après avoir effectué la phase d'apprentissage, nous avons testé notre modèle sur nos données afin de prédire les fluctuations des crypto-monnaies par rapport à une monnaie. Par exemple : les fluctuations des valeurs du BitCoin par rapport à l'Euro.

Sur cette partie nous allons présenter une partie des résultats obtenus afin de simplifier la compréhension aux lecteurs.

Nous avions le choix pour la conception de notre solution, de créer soit :

1. Un modèle pour chaque différentes monnaies, cela pourrait être intéressant afin de se concentrer que sur les caractéristiques d'une monnaie en particulier, cependant ce type d'approche nécessite d'avoir en notre possession un bon nombre de données sur chaque monnaie et qui contiendrait les différentes caractéristiques sur les fluctuations du marché. Ce type d'approche peut être utilisé sur des monnaies comme le BitCoin, qui dispose depuis quelques années de données conséquentes sur ses fluctuations par rapport aux changements du marché.Néanmoin, si nous choisissant une monnaie comme LTC, qui disposent de moins de données par rapport au BTC, nous ne pourrons pas construire un modèle robuste et efficace.
2. Un modèle unique pour toutes les monnaies, nous pouvons utiliser les données issues de différentes crypto-monnaies (nous utiliserons les monnaies les plus populaires, les autres n'ont pas assez de données), grâce à cette approche nous pourrons acquérir une expérience sur les différentes caractéristiques sur la fluctuation du marché.

Dans notre cas, pour la construction de notre modèle, nous avons décidé d'utiliser les données issues de trois monnaies : BTC, ETH et LTC, qui correspondent aux crypto-monnaies les plus connues en ce moment et qui disposent donc d'assez de données.

Les phases d'apprentissages se sont correctement passées, nous allons directement passer à la visualisation des résultats. Si les lecteurs souhaitent connaître les étapes d'apprentissages ou leurs statistiques, vous pouvez voir les NoteBook qui résument tout cela.

4.4.1 Entraînement du modèle sur BTC-ETH-LTC

Nous allons fusionner les données issues de ces trois monnaies, les normaliser et effectuer les phases d'apprentissages sur elles. Nous obtenons les résultats suivant :

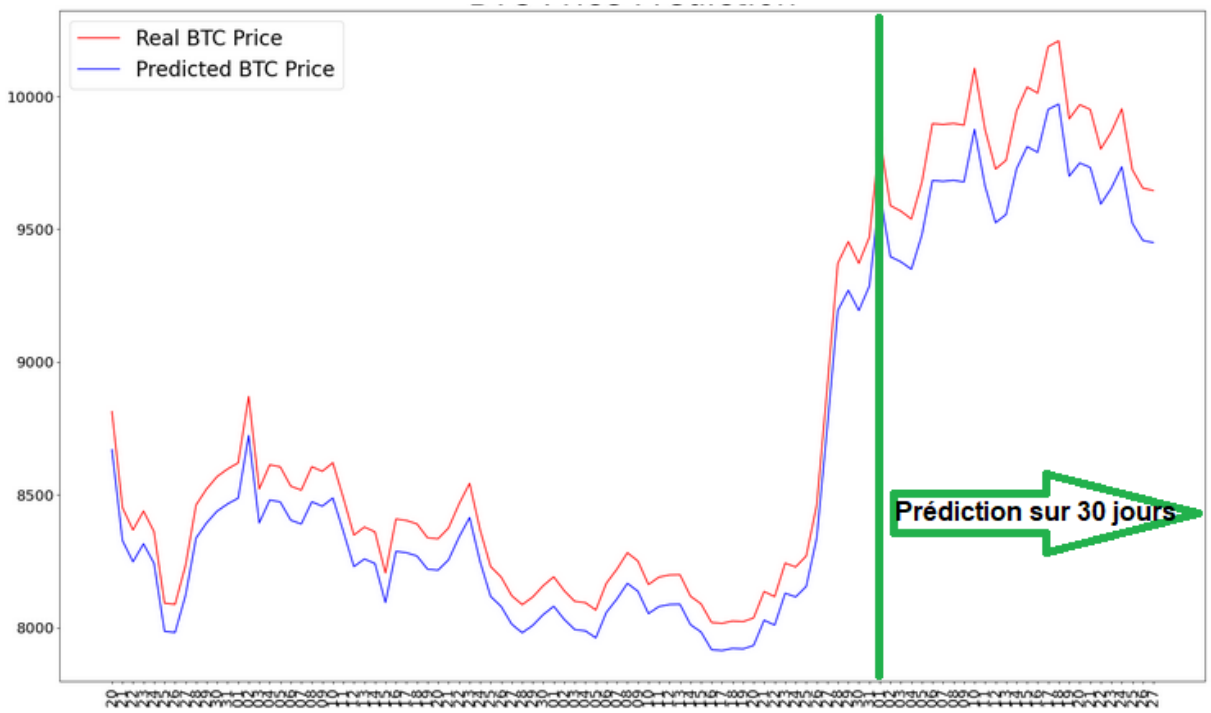


Figure 18 : Prédiction de BTC sur 30 jours.

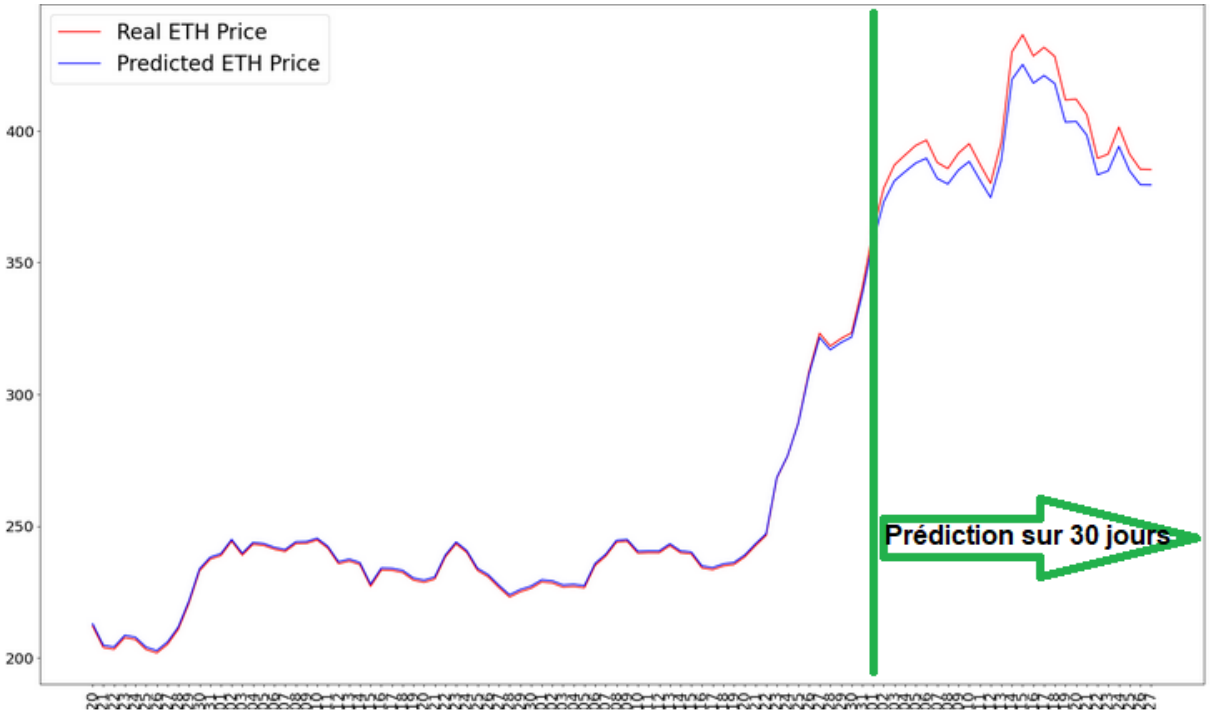


Figure 19 : Prédiction de ETH sur 30 jours.

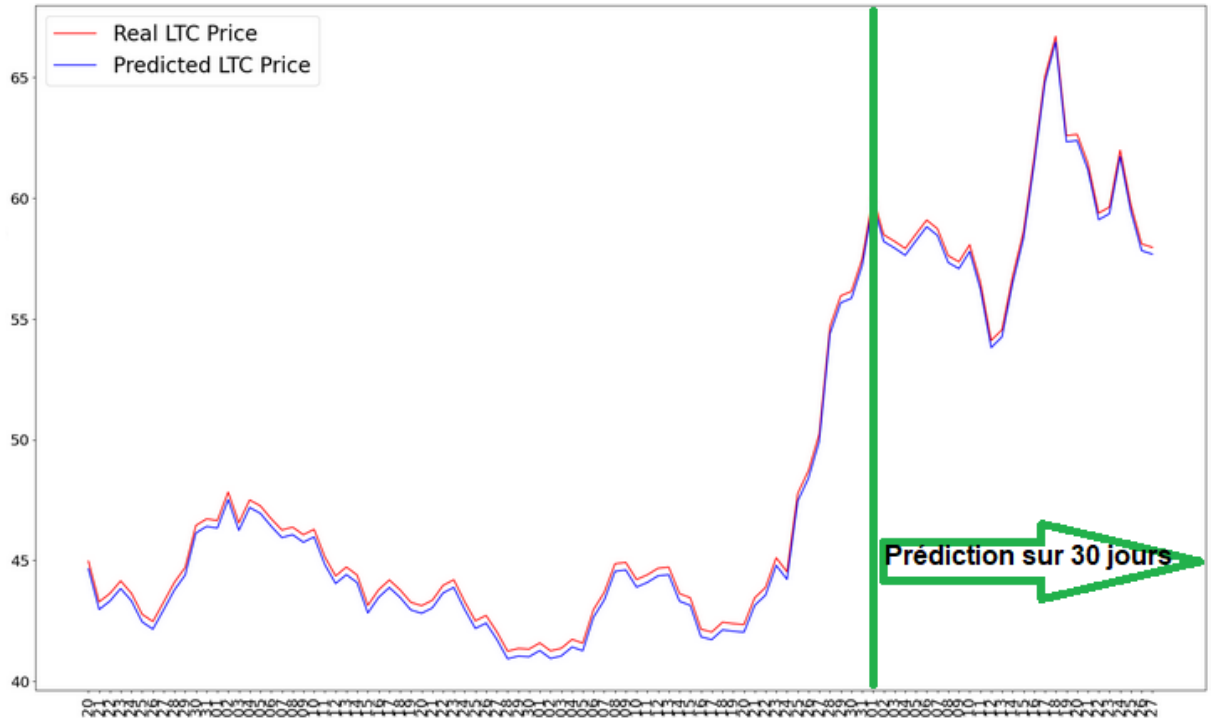


Figure 20 : Prédiction LTC sur 30 jours.

A travers ces résultats nous pouvons constater que notre modèle fonctionne assez bien et arrive à bien détecter les fluctuations de nos données sur un intervalle de 30 jours, cette intervalle peut biensure être modifié comme bon nous semble:

- Un interval grand nous permet de voir les fluctuations générales du marché, nous suggérons d'utiliser ce type d'intervalle, car il sera plus utile, puisque le but est

non pas de connaître les valeurs précises de clôtures de chaque marché, mais plutôt de connaître la tendance avec laquelle évolue notre crypto-monnaies.

- Un interval petit, nous permet de nous concentrer sur des valeurs précises dans un espace de temps court, ce type d'intervalle est à éviter car il y a de forte chance que les valeurs prédites ne soient pas exactes.

Donc le mieux est de se contenter de visualiser la tendance générale des fluctuations de notre crypto-monnaie.

5 Conclusion

Dans ce chapitre, nous avons présenté l'analyse et l'implémentation de nos modèles LSTM. Aussi nous avons parlé des différents outils et bibliothèques utilisés, qui rentrent dans le cadre de la programmation et du déploiement de notre application. Enfin, nous avons réalisé l'expérimentation de nos modèles ainsi que les résultats obtenues. Ces dernières que nous avons discutées et analysées ont bien confirmé nos prédictions faites dans le chapitre précédent.

Conclusion générale

1 Sythèse

Notre projet nous a permis d'introduire le domaine de l'apprentissage automatique et notamment celui des réseaux de neurones artificiels en raison de sa relation avec l'application mise en oeuvre. Concernant cette dernière, nous avons pu développer un dashboard doté de la plupart des méthodes de recherches connues.

Aussi, nous avons conçus trois modèles de réseaux de neurones artificiels pour la prédiction des prix de crypto-monnaies. Ensuite, nous avons expérimenté ces derniers à l'issue d'un test où nous avons vu l'évaluation du premier modèle dépasser celle des autres confirmant ses performance lors de l'apprentissage.

Ainsi, notre problématique posée auparavant est résolue, et les contributions et points positifs notés déjà acquis nous pourrons les optimiserpar par la suite en utilisant nos perspectives.

2 Perspectives

Nous envisageons plusieurs suites très prometteuses à nos travaux :

- La première perspective serait de reprogrammer le moteur avec un langage compilé plus rapide et plus performant comme le langage C ou C++. En effet, le Python est bien adapté pour l'apprentissage automatique mais est peu adapté pour les calculs intensifs. Ainsi, il serait intéressant d'entrainer les modèles sous Python, et de les utilisés ensuite dans un Dashboard codé dans un langage compilé.
- Une perspective serait d'augmenter le nombre de données d'entrainement.
- Une autre éventualité serait de tester d'avantage de combinaisons d'hyper-paramètres de notre modèle afin d'avoir des résultats encore plus satisfaisants.

Bibliographie

1. **Villani, Mission.** Qu'est-ce que l'intelligence artificielle? [En ligne] mars 2018. https://www.aiforhumanity.fr/pdfs/MissionVillani_Vulgarisation_FR-VF.pdf.

2. expertsystem. *expertsystem.* [Online] mai 6, 2020. https://expertsystem.com/machine-learning-definition/.

3. **Chaouche, Yannis.** openclassrooms. *openclassrooms.* [En ligne] 18 mars 2020. https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4020631-exploitez-votre-jeu-de-donnees.

4. **Michel Crucianu, Marin Ferecatu, Nicolas Thome, Nicolas Audebert.** Cours- Arbres de décision - Cours Cnam RCP209. *Cours- Arbres de décision - Cours Cnam RCP209.* [Online] septembre 18, 2020. http://cedric.cnam.fr/vertigo/Cours/ml2/coursArbresDecision.html.

5. **Harifi, Kenza.** Bien comprendre l'algorithme des K-plus proches voisins (Fonctionnement et implémentation sur R et Python). *Bien comprendre l'algorithme des K-plus proches voisins (Fonctionnement et implémentation sur R et Python).* [Online] september 21, 2019. https://medium.com/@kenzaharifi/bien-comprendre-lalgorithme-des-k-plus-proches-voisins-fonctionnement-et-impl%C3%A9mentation-sur-r-et-a66d2d372679.

6. **Aussem, Alexandre.** Introduction aux Réseaux Bayésiens. [Online] 2009. https://perso.liris.cnrs.fr/alain.mille/enseignements/Master_PRO/TIA/RBayesiens/Intro_RB.pdf.

7. **DREYFUS, Gérard.** Réseaux de neurones. [Online] https://www.universalis.fr/encyclopedie/reseaux-de-neurones-formels/.

8. **Maurice, Bastien.** Fonction d'activation - Deep Learning. [Online] septembre 26, 2018. https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/.

9. **Crouspeyre, Charles.** Comment les Réseaux de neurones à convolution fonctionnent. [Online] juillet 17, 2017. https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8.

10. **Juri'Predis.** Démystifier le Machine Learning, Partie 2 : les Réseaux de Neurones artificiels. [Online] https://www.juripredis.com/fr/blog/id-19-demystifier-le-machine-learning-partie-2-les-reseaux-de-neurones-artificiels.

11. **Roger Grosse, Nitish Srivastava.** Recurrent neural nets. [Online] fevrier 03, 2015. http://www.cs.toronto.edu/~rgrosse/csc321/lec9.pdf.

12. **Crouspeyre, Charles.** Comment les Réseaux de neurones récurrents et Long Short-Term Memory fonctionnent. [Online] octobre 10, 2017. https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-c25041d45921.

13. **Phi, Michael.** Illustrated Guide to LSTM's and GRU's: A step by step explanation. [Online] septembre 24, 2018. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

14. **Miller, Lachlan.** Machine Learning week 1: Cost Function, Gradient Descent and Univariate Linear Regression. [Online] janvier 10, 2018. https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd.

15. **Tremoulis, Odyssée.** *Système de trading automatique et prédiction de cours à l'aide de reseaux de neurones.* Paris : s.n., 2012.

16. A short overview of Crypto Trading Bots. *A short overview of Crypto Trading Bots.* [Online] mai 21, 2018. A short overview of Crypto Trading Bots.

17. **Peixeiro, Marco.** Analyse avancée des séries temporelles avec ARMA et ARIMA. *Analyse avancée des séries temporelles avec ARMA et ARIMA.* [Online] https://towardsdatascience.com/advanced-time-series-analysis-with-arma-and-arima-a7d9b589ed6d.

18. **Carsten.** 400+ crypto currency pairs at 1-minute resolution. *Historical crypto currency data from the Bitfinex exchange including Bitcoin.* [Online] https://www.kaggle.com/tencars/392-crypto-currency-pairs-at-minute-resolution.

19. **Raval, Siraj.** Loss Functions Explained. youtube. *Loss Functions Explained. youtube.* [Online] Juillet 24, 2018. https://www.youtube.com/watch?v=IVVVjBSk9N0.

20. **Brownlee, Jason.** How to Choose Loss Functions When Training Deep Learning Neural. *How to Choose Loss Functions When Training Deep Learning Neural.* [Online] Janvier

30, 2019. https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deeplearning-.

21. **Grover, Prince.** 5 Regression Loss Functions All Machine Learners Should Know. *5 Regression Loss Functions All Machine Learners Should Know.* [Online] juin 5, 2018. https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0.

23. **mobiny, Aryan.** How to Use TensorBoard? *How to Use TensorBoard?* [Online] juin 9, 2018. https://itnext.io/how-to-use-tensorboard-5d82f8654496.

24. **Willems, Karlijn.** TensorFlow Tutorial For Beginners. *TensorFlow Tutorial For Beginners.* [Online] janvier 16, 2019. https://www.datacamp.com/community/tutorials/tensorflowtutorial?.

25. **Rosebrock, Adrian.** Keras Tutorial: How to get started with Keras, Deep Learning, and. *Keras Tutorial: How to get started with Keras, Deep Learning, and.* [Online] septembre 10, 2018. https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-withkeras-.

26. **community, NumPy.** NumPy User Guide. *NumPy User Guide.* [Online] mai 29, 2016. https://docs.scipy.org/doc/numpy-1.11.0/numpy-user-1.11.0.pdf.