République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER RECHERCHE

En
Informatique

Option
Intelligence artificielle

Thème

Prédiction de la confiance dans les réseaux sociaux avec des méthodes de l'IA

Présenté par : M. AMMAR KHODJA Yanni
Mlle. OUDJEHANI Lynia

Soutenu le 25 septembre 2022 devant le jury composé de :

Président	M. H. SLIMANI	Professeur	U. A/Mira Béjaia.
Examinateur	M. K. BEDJOU	MAA	U. A/Mira Béjaia.
Encadrant	M. K. AKILAL	MCB	U. A/Mira Béjaia.

⁂ *Remerciements*

Nous tenons à remercier sincèrement notre encadrant AKILAL Karim, pour la pertinence de son encadrement, ses précieux conseils et encouragements, et surtout pour ses qualités humaines, qui nous ont permis de mener à terme ce travail de recherche.
Nous adressons également nos remerciements aux membres du jury d'avoir accepté de juger notre travail.
Enfin nous exprimons notre profonde reconnaissance à tous les responsables et enseignants de l'université de Bejaia qui ont contribué à notre formation.

⁕ *Dédicaces* ⁕

Je dédie ce modeste travail **à** mes parents qui m'ont épaulée couragement durant ces 5 dernières années de formation.
Plus spécialement à ma Mère et ma sœur, sans qui mon rêve de poursuivre mes études n'aurait pu être possible. Merci de votre accompagnement à lire et relire mes écrits.
A mon frère pour sa patience.
A tous mes proches, pour chacune de vos petites attentions qui ne sont pas passées inaperçues.

Mlle. OUDJEHANI Lynia

⁂ *Dédicaces* ⁂

Je remercie ma mère et mon père, pour leur amour, leurs efforts ainsi que leur soutien inconditionnel, à la fois moral et économique, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.
A mes deux grands frères Amazigh, Karim, ainsi que mon cousin Bahmane pour tous les conseils de vie que j'ai fort apprécié.
A mes trois chers amis, Foufi, Youva, Mustapha.
Je tiens également à rendre hommage à mon ancien enseignant ALOUI Abdelouhab à travers ce mémoire.

*M. **AMMAR KHODJA Yanni***

Table des matières

Table des figures

Liste des tableaux

Notations et symboles

ACP	Analyse en composantes principales.
AGR	Agreement.
ANN	Artificial Neural Network.
CER	Controversy, eclecticism, and reciprocity.
CPU	Central Processing Unit.
DT	Decision Tree.
GCR	Gullibility, competence, and reciprocity.
$GFtrust$	Friendship Graph trust.
$GridSearchCV$	Recherche de grille.
$HITS$	Hyperlink-Induced Topic Search.
$HPLP$	Prédiction de liens haute performance.
IA	Intelligence artificielle.
KNN	K-Nearest Neighbors.
MAE	Mean Absolut error.
ML	Machine learning.
MLR	Multiple linear regression.
PC	Composant principal.
PCC	Pearson correlation coefficient.
RF	Random Forest.
REC	Reciprocal.
$RLMP$	Régression linéaire multiple polynomiale
$RMSE$	Root mean squared error.
$sonLP$	Social network link prediction by principal component regression.
$STAR$	Semiring trust inference for trust-aware social recommenders.
SVM	Support Vector Machines.
SVR	Support Vector Regression.
$XGBoost$	eXtreme Gradient Boosting.

Introduction générale

L'être humain étant sociable, cherche toujours à nouer de nouveaux liens, d'où l'existence des réseaux sociaux. Ces derniers sont des services en ligne qui facilitent le développement des relations sociales entre personnes qui sont représentées par des profils. Toutefois ces profils ne reflètent pas de façon fidèle les personnalités et les intentions des personnes derrières. D'où l'intérêt de prendre des précautions devant une demande d'amitié. Une fois acceptée, cette personne aura accès à certaines informations privées.

Si la personne est mal intentionnée, c'est une porte ouverte à la cyberattaque. Celle-ci ne cause pas que des dommages financiers mais aussi des dommages psychologiques, qui peuvent être irréversibles à ses victimes. Pour atténuer ces risques, deux options s'offrent :

- **(a)** : Ne pas faire confiance et ne pas accepter la demande d'amitié, donc ne prendre aucun risque mais passer à côté d'une opportunité d'une croissance sociale et une future belle amitié florissante.
- **(b)** : Apprendre quand, à qui et à quelle mesure faire confiance.

Aujourd'hui, avec les milliards d'utilisateurs que comptent les réseaux sociaux, la question ne se pose plus, mesurer la confiance est la solution. Elle peut paraître intuitive dans la vie réelle, mais en ligne, il est plus complexe et difficile de la prédire. Des difficultés telles que l'absence de proximité et le flux de données énorme et incessant de données échangées.

Parmi les méthodes d'inférence statiques, celles basées sur l'intelligence artificielle et plus précisément l'apprentissage automatique (Machine Learning) sont parmi les méthodes les mieux adaptées pour traiter les problèmes de grandes quantités de données.

Ce projet de fin d'étude vise à construire des modèles d'apprentissage automatique, permettant de détecter la confiance et la méfiance qu'un utilisateur donné peut attribuer à un autre utilisateur dans les réseaux sociaux.

Ce mémoire est organisé comme suit :

Le premier chapitre est consacré aux notions de base sur la confiance plus particulièrement dans les réseaux sociaux. Nous présentons quelques définitions et travaux connexes à la mesure de confiance.

Le deuxième chapitre est un aperçu sur l'intelligence artificielle en général et sur l'apprentissage automatique en particulier. Le but ici est de comprendre l'approche d'apprentissage automatique qui va être modélisée et implémentée dans les chapitres suivants.

Dans le troisième chapitre, nous présenterons notre approche dans le but de résoudre le problème en question, commençant par une vue d'ensemble des outils et de l'environnement. Nous passons à la description de l'ensemble de données utilisés et à leurs pré-traitement.

Le quatrième chapitre est, quant à lui, dédié à la mise en œuvre de notre approche. Nous présenterons un certain nombre d'algorithmes adoptés. Ensuite, nous les appliquons sur l'ensemble de données préparé dans le chapitre précédent. Puis, nous discuterons et comparerons les résultats obtenus.

Nous conclurons ainsi ce mémoire par une conclusion générale et quelques perspectives futures.

Chapitre 1

Notions de base sur la confiance

"La conviction tirée de la confiance est plus forte que toutes les assurances appuyées sur des preuves." Claire De Lamirande

1.1 Introduction

Faire confiance, on fait ça tous les jours. Quand on réfléchit un peu à la relation avec autrui, c'est le concept le plus évident qui nous vient à l'esprit.

Quand nous confions nos enfants à l'école, quand nous achetons des produits. Lorsque les citoyens votent, ils confient à leurs représentants une parcelle de souveraineté. La confiance est présente dans l'ensemble de nos activités sociales, elle imprègne toute notre existence sociale.

Lorsqu'un virus a fait surface et a chamboulé notre existence du jour au lendemain, nous avons dû faire confiance à des virologues que nous ne connaissons pas, à des politiciens pour prendre des décisions qui servent le bien collectif, aux **réseaux sociaux** pour le maintien d'un lien social. Dans une situation aussi extrême, la peur mène aux conflits tandis que la confiance permet l'entraide nécessaire à la survie, en jouant son rôle de mécanisme de réduction de la complexité sociale [1].

En effet, faire confiance, c'est se sentir en sécurité, mais cette dernière n'est pas stable dans le temps. Elle est très sensible au contexte de la relation. Si le contexte change, la confiance change aussi. Elle ne se décrète pas, ne se commande pas et ne se contrôle pas.

Aujourd'hui, où le monde n'a jamais été aussi **connecté**, ou l'économie et les relations n'ont jamais été aussi prospères, nos sociétés ne sont pas fondées sur quelque chose d'aussi peu fiable que la confiance mais sur la recherche de sécurité et de maîtrise [2].

Dans le présent chapitre, nous allons présenter les principales définitions de la confiance et les travaux menés sur la mesure de confiance.

1.2 Définition

Qu'est-ce que la confiance ? Un savoir hypothétique sur une conduite future ! À quoi sert-elle ? À faciliter les rapports sociaux, permettre le vivre-ensemble ! Pourquoi l'étudier ? Parce qu'il s'agit d'une importante catégorie de la pratique et qu'elle est, jusqu'alors, (paradoxalement) peu construite comme une catégorie à part ! Luhmann,1988 [1]

Malgré la popularité montante du concept de la confiance, d'être le capital social, il n'existe pas aujourd'hui de consensus sur ce qu'est la confiance, et encore moins sûr ce qui la détermine.

Mais de nombreux travaux ont cherché à définir voire même à modéliser la confiance et dans divers domaines. Parmi eux :

La confiance peut être une attente sociale [1] [3] , aussi bien qu'un état psychologique [4] [5], une volonté de se rendre vulnérable [6], ou comme une certitude ou un sentiment de sérénité [7].

En économie, la confiance est modélisée par des modèles qui privilégient généralement la **coopération** entre agents [8].

Selon Fukuyama [9] la confiance est liée aux **attentes** des membres d'une communauté dans laquelle les individus partagent habituellement certaines normes et adoptent un comportement **prévisible** reposant sur des valeurs **partagées**.

En approfondissant le lien entre les notions de coopération, de partage, et de confiance, Demolombe [10], a étendu ces modèles avec sincérité. L'auteur fait remarquer que la confiance d'un individu A envers un individu B n'est possible que si B est sincère aux yeux de A.

Nooteboom et al. [11], eux postulent que la confiance comporte une dimension **altruiste** permettant de supposer la **réciprocité** de ce sentiment, nonobstant tout intérêt personnel.

Ainsi, selon ces définitions, la confiance est associée aux sources **altruistes** de la **coopération**, de la **loyauté** et de la **sincérité** envers le partenaire, comme le reflètent aussi les définitions présentées dans le Tableau 1.1.

Définitions mettant particulièrement en avant les attentes positives	**Auteurs**
Etat impliquant des attentes assurément positives quant aux motivations d'une autre partie à notre égard dans une situation comportant des risques.	Lewicki et Bunker, 1995 [12]
Croyance individuelle ou une croyance commune parmi un groupe d'individus qu'un autre individu ou groupe a) fait des efforts de bonne foi pour se conduire en accord avec des engagements explicites ou implicites, b) est honnête quelles que soient les négociations qui ont précédé de telles implications, c) et ne tire pas d'avantages excessifs des autres même lorsque l'opportunisme est possible.	Cummings et Bromiley, 1996 [13]
Ensemble d'attitudes mentales qui caractérisent l'esprit de l'agent X qui veut déléguer, qui préfère qu'un agent Y fasse l'action. X croit que Y a l'intention de faire l'action et qu'il persévérera pour y parvenir, en présupposant de l'accord d'Y avec l'objectif et de la volonté de Y de l'atteindre.	Castelfranchi et Falcone, 2000 [14]
Expression de foi et d'assurance qu'une personne ou une institution sera juste, fiable, éthique, compétente et non menaçante.	Caldwell et Clapham, 2003 [15]
Processus d'anticipation imaginative de la fiabilité des actions de l'autre partie à partir de "1" la réputation du partenaire et de l'acteur "2" de l'évaluation des circonstances actuelles de l'action "3" des suppositions à propos du partenaire et la croyance dans l'honnêteté et la moralité de l'autre partie.	Khodyakov, 2007 [16]
Sentiment de sérénité qui émane de la relation à un acteur sur qui l'on se repose dans une situation donnée en espérant qu'il prendra soin de nos intérêts.	Karsenty, 2013 [17]

Tableau 1.1 – Sélection de définitions de la confiance interpersonnelle ou de la confiance en général

Toutefois, lorsqu'on évoque la confiance entre deux partenaires, il existe une relation d'agence dans laquelle le principal, appelé *trustor* (En anglais, celui qui fait confiance), fait confiance à un agent ou *trustee* (En anglais, celui à qui on accorde sa confiance).

En principe, le trustor accorde sa confiance au trustee pour que ce dernier réalise le mandat dans le respect des attentes du premier. Et c'est cette relation entre trustor et trustee, que nous allons étudier pour pouvoir mesurer la confiance.

Nous avons discuté de la confiance, mais qu'en est-il de la méfiance ? Comme le suggèrent Massa et Avesani [18], la prévision de la méfiance est tout aussi importante que la prévision de la confiance, et pourrait être même plus importante.

Plusieurs théoriciens ignorent la méfiance, c'est-à-dire, la considèrent comme l'absence de confiance, une neutralité. Cependant, dans notre réalité, se méfier de quelqu'un est une attitude si différente que d'être neutre à son égard.

La méfiance est fondamentalement caractérisée par un sentiment de malaise, de pessimisme, et dans une certaine mesure, de peur, comme l'expliquent Lerwick et al. [19]. Des caractéristiques loin de la définition de "la neutralité". Nous énumérons dans le Tableau 1.2 d'autres définitions de la méfiance :

Définition de la méfiance	**Auteurs**
La méfiance se développe dans un but de défense et de recherche de sécurité. Pourtant, lorsqu'elle tend à l'exagération, la méfiance provoque souvent l'effet inverse, et entretient au contraire un sentiment d'insécurité et la peur des risques de mensonges ou de trahison.	Le-Parisien [20]
La confiance permet de construire le lien social. La méfiance quant à elle consacre la rupture du lien précité. Elle ne permet pas de se fier ni pour autant de défier. Elle renvoie à l'idée de doute, d'être en attente de quelque chose éventuellement.	Yann Algan, Pierre Cahuc [2]
État d'esprit de quelqu'un qui se tient sur ses gardes face à quelqu'un d'autre ou à propos de quelque chose : Éveiller la méfiance de quelqu'un.	Larousse [21]
Disposition contraire à la confiance et en vertu de laquelle on craint d'être trompé et on se tient sur ses gardes.	Lafaye 1861 [22]
Essentiellement soupçonneuse et inquiète, elle se fait prendre en mauvaise part. Elle touche à la misanthropie.	Lafaye 1861 [22]

Tableau 1.2 – Sélection de définitions de la méfiance.

1.3 Mesures de confiance

Mesurer la confiance, nous permettra de prédire si cette personne est digne de confiance, ou au contraire, s'il faut s'en méfier, et ne pas rester dans l'incertitude.

Comme le dit Alphonse Karr [23] : *"L'incertitude est le pire de tous les maux"*, ou par l'une des fameuses citations de Diane de Beausacq [24] : *"Même devant le malheur arrivé, le moment où cesse l'incertitude, produit toujours en nous une sorte d'apaisement "*.

Nous avons cité plus haut, que nos sociétés sont fondées sur la sécurité, s'avérant être tout l'inverse de l'incertitude. Donc, pour la sécurité de certains aspects de notre vie, il n'y a pas de place pour cette dernière. Le défi ici consiste à pouvoir sortir de l'incertitude en arrivant à prédire la confiance et la méfiance.

Plusieurs études ont été menées dans cette voie, pour évaluer notre capacité à détecter les individus dignes de confiance et les individus dont nous devrions nous méfier. Mais il s'avère que sans connaître l'historique d'**interaction** d'un inconnu, sans connaître sa **réputation** ou ses projets, et simplement en interagissant pendant une courte période, nous avons peu de signaux, d'indices trahissant ses dispositions et ses émotions.

Or, il est permis de croire que notre capacité à identifier les individus susceptibles de notre confiance, imparfaite qu'elle soit, est bien réelle et offre des résultats.

Selon Golbeck [25] en fonction de l'environnement, la confiance peut avoir différents attributs. Cependant, dans la plupart des cas, en particulier dans les environnements distribués, tel que les **réseaux sociaux**, la confiance suit certaines propriétés qui sont illustrées dans la Figure 1.1.

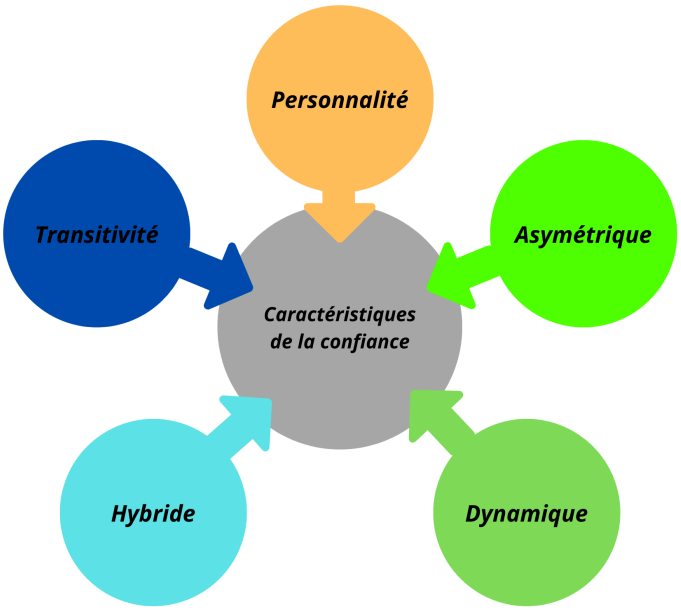

FIGURE 1.1 – Propriétés de la confiance [25].

Pour mieux illustrer notre travail, nous allons par ce qui suit, nous concentrer sur les travaux déjà réalisés sur la prédiction de confiance dans les réseaux sociaux.

Parmi les propriétés de la confiance utilisées dans ces certains travaux, nous pouvons citer :

Wang et al. [26] qui ont développé une méthode pour mesurer la confiance entre utilisateurs basée sur leurs goûts communs. Les données sont regroupées dans différents groupes, un ensemble de goûts personnalisé est construit pour l'utilisateur et est utilisé pour déduire la confiance basée sur les goûts communs entres utilisateurs.

Maheswari & Karpagam [27] ont évalué la confiance en utilisant une approche basée sur la réputation. Trois propriétés ont été utilisées dans leur approche : la transitivité, la composabilité et l' asymétrie.

La transitivité qui signifie que pour deux nœuds A et B, A doit faire confiance à B pour faire des recommandations à A sur les autres [25].

La composabilité décrit la situation lorsqu'un certain nombre de recommandations sur la fiabilité d'un nœud sont reçues ; là, les valeurs de confiance dans ces recommandations devraient alors être composables en une seule croyance sur la fiabilité du dit nœud.

L'asymétrie de la confiance consiste en le fait que la confiance ne soit pas nécessairement identique entre 2 personnes. Parce que les individus ont des antécédents et des expériences différentes, il est compréhensible que deux personnes puissent se faire confiance chacun avec des valeurs différentes.

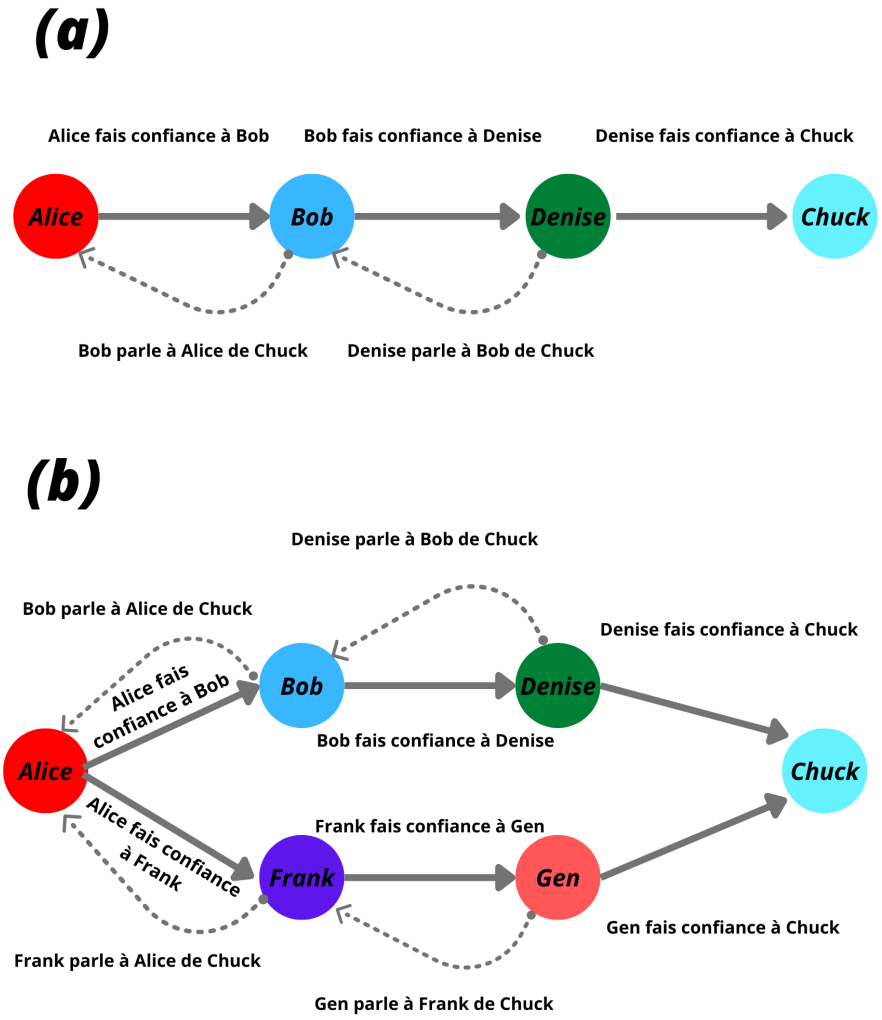

FIGURE 1.2 – Structures de chemin de réseau pour trouver la confiance.

La partie (a) montre une chaîne simple de gens où les caractéristiques transitives de la confiance permettent à Alice de se faire une opinion de Chuck sur la base des informations que Denise donne à Bob et Bob, à son tour, donne à Alice.

La partie (b) montre une structure plus complexe où Alice reçoit des informations de deux personnes et elle doit se faire une opinion de Chuck en composant les informations dont elle dispose.

Parmi les méthodes existantes sur la prédiction de la confiance, certaines permettent de prédire à la fois la confiance et la méfiance. Certaines de ces méthodes sont dites supervisées, utilisant des techniques d'apprentissage automatique, d'autres non supervisées. Certaines utilisent les données d'interaction entre les utilisateurs [28] d'autres les émotions [29], et d'autres la similarité des intérêts [30].

1.3.1 Prédiction non supervisée de la confiance

Il existe deux manières différentes de procéder à la prédiction non supervisée de la confiance basée sur les graphes. La première utilise des métriques locales, c'est-à-dire, comment un nœud fait confiance à un autre nœud. La seconde, est l'utilisation de métriques globales, qui, elles, décrivent le degré de confiance d'un nœud donné, si un nœud est digne de confiance ou pas en général.

1.3.1.1 Prédiction de la confiance par métriques locales

La plupart des approches par métriques locales utilisent des **règles de propagation** de la confiance à travers un graphe de confiance, d'une source **u** à un puits **v** [31].

Dans un graphe, un nœud représente un utilisateur, et un lien (c'est-à-dire un arc) représente la relation sociale entre deux utilisateurs. Les réseaux sociaux sont dynamiques, donc leurs graphiques changent avec le temps. Les graphes de réseaux sociaux sont dits signés s' ils incluent des relations de confiance et de méfiance, sinon s' il y a que des relations de confiance, ils sont dits non-signés. (Voir la Figure 1.3)

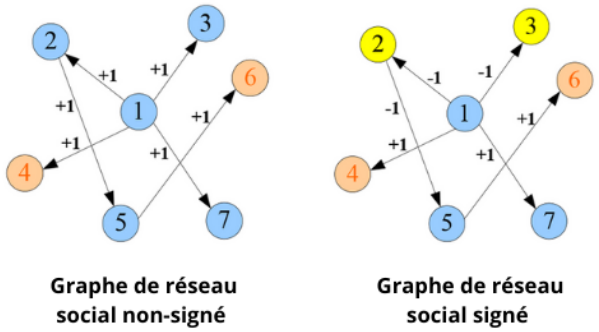

FIGURE 1.3 – Exemples de graphes de réseau social signé et non signé.

Parmi les divers algorithmes non supervisés utilisant seulement les métriques locales, nous pouvons citer les plus populaires tels que :

L'algorithme Tidal Trust [25] qui estime la valeur de confiance entre deux membres d'un réseau social. Il calcule les valeurs de confiance en utilisant les poids moyens de tous les voisins du membre.

Mole Trust [18], cet algorithme effectue une recherche en profondeur pour propager et déduire la confiance dans le réseau de confiance. Et enfin l'algorithme GFTrust [32].

Appliquer ces algorithmes tels quels sur des réseaux signés n'est pas simple. Car selon Gao et al. [32] la méfiance n'est pas transitive.

Pour la prédiction de la méfiance, les algorithmes tels que :

STAR [32], un algorithme propagatif. Un algorithme propagatif permet de propager la confiance comme le montre la Figure 1.4 [33]. Les lignes pleines indiquent les relations d'approbation existantes et les lignes pointillées indiquent la confiance propagée.

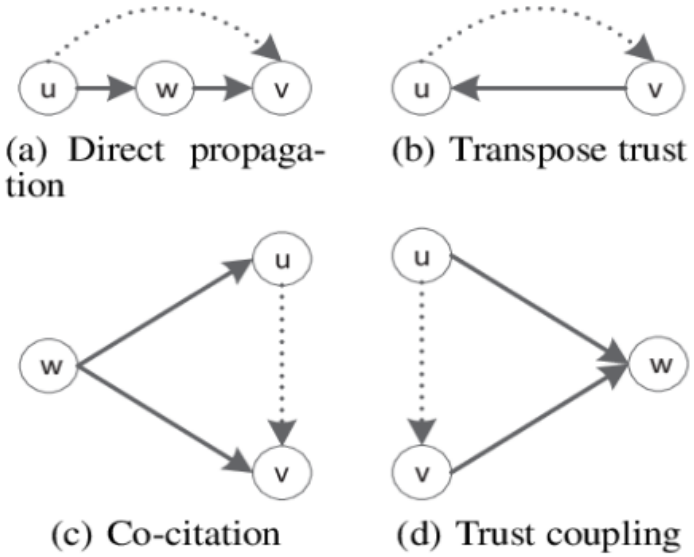

FIGURE 1.4 – Exemple d'opérateurs de propagation [33].

L'algorithme Appleseed [34], où la confiance est considérée comme un lien positif passant des nœuds à leurs trustees, et la méfiance comme un lien négatif.

Cependant, la plupart de ces algorithmes, en plus de leur complexité temporelle, souffrent de dégradation de la confiance sur les longs chemins et de dépendances [35].

Une autre approche, celle-ci ne se reposant pas sur la transitivité de la confiance, un algorithme basé sur le filtrage collaboratif, proposé par Akilal et al. [30]. Cet algorithme utilise uniquement les informations provenant des voisins directs des trustors et des trustees, et en utilisant l'accord comme métrique de la similarité, il peut déduire à la fois les relations de confiance et de méfiance.

Au lieu d'exploiter l'aspect transitif de la confiance ; çàd comme sentiment qui se propage à travers un réseau social. D'autres approches considèrent que la confiance est le résultat d'une confrontation entre les traits sociaux des deux acteurs (trustor/trustee), ou les traits les plus dominants l'emportent et affectent donc la valeur de la confiance [35] [36].

1.3.1.2 Prédiction de la confiance par métriques globales

Plusieurs approches ont été proposées, parmi ces algorithmes, on peut citer :

L'algorithme PageRank [37] utilisé pour calculer la popularité d'une page web par rapport à certains mots-clés. Cet algorithme a été exploité pour prédire la notoriété dans les réseaux sociaux. L'algorithme HITS et l'algorithme EigenTrust. Conçues pour les réseaux non signés.

Pour les réseaux signés, les algorithmes PageRank et HITS ont également été revisités pour prendre en compte les liens négatifs.

Les limites de ces approches, et le fait que la confiance soit subjective [18], font que les métriques globales ne soient pas aptes à décrire très précisément comment un trustor **U** devrait se fier, ou se méfier d'un trustee **V**.

1.3.2 Prédiction supervisée de la confiance

Les approches d'apprentissage supervisé s'appuient sur la riche littérature d'apprentissage automatique. Ces méthodes sont appliquées sur des données représentées par des caractéristiques, qui lient chaque paire d'utilisateurs dépendant du voisinage ou du nombre de liens les séparant, calculés en fonction de la structure du graphe. La définition de ces caractéristiques joue un rôle essentiel. Elle est la partie la plus critique de tout algorithme d'apprentissage automatique.

La confiance entre les utilisateurs est évaluée via des mesures de similarité basées sur la structure du graphe. Des auteurs comme Naderan et al. [38], Nowell et al. [39], Al Hasan et al. [40], ont proposés certaines caractéristiques, les plus populaires :

— Le voisinage commun.
— La taille des chemins séparant les utilisateurs.

Les méthodes d'apprentissage supervisées les plus appliquées sont :

Les réseaux de neurones artificiels, le modèle de Markov caché, les machines à vecteurs de support (Support Vector Machines) ou encore des techniques des K plus proches voisins et d'arbres de décision. L'hypothèse est que les nœuds ou les liens qui ont des caractéristiques similaires tendent à appartenir aux mêmes classes.

Naderan et al. [38] ont étudié l'évaluation de la confiance dans le réseau social Epinions. Après le traitement des données brutes de l'ensemble de données, ils les convertissent en un vecteur de caractéristique. L'étiquette du vecteur de caractéristiques, c'est-à-dire, la valeur de confiance, est convertie en valeurs floues avec deux, trois et cinq classes. Enfin, le vecteur de caractéristiques et les étiquettes sont alimentés en trois classificateurs : SVM , arbre de décision, et KNN pour trouver le modèle de confiance et de méfiance. Parmi les trois classificateurs, l'arbre de décision réalise le meilleur classement.

Aouay et al. [41] ont étudié la prédiction de la confiance en utilisant deux jeux de données, leur appliquant la méthode des k-plus proches voisins et celles des arbres de décision, en conjointement avec une méthode des attributs de sélection, basée sur l'analyse en composantes principales (ACP), la précision de prédiction augmente plus, et donc donne de meilleures performances.

Bao et al [42] , proposent sonLP un prédicteur de lien de réseau social. Il utilise l'ACP pour déterminer les composantes principales (PC) les caractéristiques. Chaque PC est une somme pondérée des caractéristiques. L'ACP détermine les pondérations de sorte que les composantes soient statistiquement indépendantes et classées par contribution décroissante à la variance de la variable de résultat, c'est-à-dire, si un lien se forme. sonLP sélectionne les variables avec les plus grandes valeurs propres, puis les regroupe en n classes.

Lichtenwalter et al. [43] proposent le HPLP (prédiction de liens haute performance), une méthode d'apprentissage supervisée de pointe qui utilise l'algorithme Random forest. Il faut 16 caractéristiques du lien topologique (mesure de chemins, in/out degree,...). Bien que certaines caractéristiques faibles soient parfois utiles, Random forest évite de les suradapter. Puis revisité (HPLP+) hautement évolutif.

Selon Naderan et al. [38] et Al Hasan et al. [40], les classificateurs (méthode d'apprentissage supervisé) sont des solutions beaucoup plus appropriées pour classer la confiance dans les réseaux sociaux. A noter que pour la résolution de la question sur la méfiance, les réseaux signés sont tout aussi un problème de classification.

1.4 Conclusion

Ce chapitre présente une formalisation de la confiance à utiliser dans la prédiction dans les réseaux sociaux. Partant d'un bagage sociologique et psychologique, on a présenté une définition de la confiance sociale conçue pour être utilisée dans les réseaux sociaux. Puis on a présenté un aperçu des travaux émis sur la prédiction de la confiance dans les réseaux sociaux, ainsi que la prédiction de la confiance et de la méfiance. Partant de ces travaux, nous avons décrit deux des approches les plus étudiées dans la littérature : la prédiction non supervisée de la confiance et la prédiction supervisée de la confiance. Ainsi, on a pu comprendre comment les caractéristiques de la confiance se traduisent en algorithmes.

Ce chapitre contribue à lier les aspects algorithmiques de ce travail avec la nature de la confiance dans les réseaux sociaux. Le prochain chapitre introduira les méthodes d'intelligence artificielle, plus précisément à celle de l'apprentissage automatique supervisé.

Chapitre 2

Méthodes de l'intelligence artificielle

2.1 Introduction

À travers l'histoire de l'évolution de l'homme, on peut voir que pour de nombreuses tâches il est le plus performant de son écosystème jusqu'à l'arrivée des machines. Celles-ci n'ont cessé de le rattraper. Elles sont plus puissantes physiquement, calculent plus vite et peuvent stocker les informations de manière plus fiable, plus durable, et en plus grandes quantités. Elles sont capables de communiquer à travers le monde en quelques millisecondes et avec un débit d'information énorme.

Ceci dit, pour l'instant, certaines facultés humaines comme le bon sens ou la capacité d'apprentissage demeurent hors de portée des machines. Mais depuis quelques années une révolution semble s'être mise en place et cette révolution a été incarnée par l'Intelligence Artificielle (IA).

La naissance de l'IA fut une nouvelle ère pour l'homme. Une ère où les données sont le nouvel or noir, où Amazon, Google et Facebook, sont les entreprises les plus riches et les puissantes que l'histoire ait jamais connues. Une poignée de personnes d'une poignée d'entreprises influencent l'opinion de milliards de gens et cela grâce à l'IA.

l'IA est partout. Quand on navigue sur internet et sur les réseaux sociaux, l'IA décide de ce qui va nous être recommandé. Par exemple sur le fil d'actualité de facebook c'est l'IA qui choisit les publications à voir en premier et celles qui vont passer à la trappe. Quand on fait une recherche sur google ou un autre moteur de recherche, la manière dont sont ordonnés les résultats, l'IA est utilisée pour filtrer l'information qui n'est pas pertinente et en revanche mettre les résultats qui sont plus susceptibles d'être intéressants.

Dans ce chapitre, nous allons décrire qu'est ce que l'intelligence artificielle, son histoire entre fiction et réalité. Puis nous verrons quelques méthodes de l'IA, qui seront utilisées dans les chapitres suivants.

2.2 Définition de l'IA

Qu'est ce que l'Intelligence artificielle ?

L'expression intelligence artificielle est composé de deux mot :

— ce qui n'est pas innée, résultat produit par le travail de l'homme [44].
— un ensemble des processus retrouvés dans des systèmes, plus ou moins complexes, qui permettent de comprendre, d'apprendre, ou de s'adapter à des situations nouvelles [45].

Bien qu'il n'y ait pas de définition générale, les chercheurs s'accordent sur le fait que l'intelligence artificielle ou bien l'IA, est une jeune discipline qui réunit des sciences, des théories, et des techniques (logique mathématique, statistiques, probabilités, informatique ...) [46], dans le but de permettre aux machines d'exécuter des fonctions normalement associées à l'intelligence humaine, telles que la perception, le raisonnement, etc...

Ces définitions diffèrent significativement dans leur façon de définir l'intelligence. Parmi les grands axe qui y découlent [47] :

— Créer des systèmes qui se comportent comme les êtres humains : cette définition a été introduite par le test d'Alan Turing, selon lequel une machine est considérée intelligente seulement si elle ne peut pas être distinguée d'un être humain durant un interrogatoire.
— Créer des systèmes qui pensent comme des êtres humains.
— Créer des systèmes qui pensent rationnellement : les systèmes doivent raisonner.

2.3 Histoire de l'IA

L'intelligence artificielle en tant que discipline scientifique n'est bien sûr pas née ex nihilo. En 1956, le jour où on lui a attribué son nom. Elle est le fruit d'un long processus de réflexion scientifique dans l'histoire de l'humanité.

Selon Marquis et al. [48] pour brosser ce panorama historique, il distinguent schématiquement quelques grandes périodes :

La première allant de l'Antiquité au XVIe siècle, où les créatures artificielles peuplent l'imaginaire collectif comme le reflètent les mythes, les contes, et les œuvres littéraires de la plupart des cultures. Bien avant les robots, le désir de libération de la servitude du travail, l'utilisation de la créature artificielle à des fins belliqueuses, animer les esprits...

Puis une période de transition vers la modernité au XVIIe et au XVIIIe siècles, avant la mathématisation de la logique au XIXe siècle. Où il eut la naissance de la logique moderne et qui voit aussi le développement progressif des probabilités. Elle est aussi marquée par l'émergence de premières machines.

Puis la naissance de l'informatique, de la théorie de la calculabilité à la cybernétique, dans la première moitié du XXe siècle. Une période marquée le plus par les travaux de Claude Shannon [49] sur les fondements de la théorie de l'information, et ceux d'Alan Turing [50] sur les fonctions calculables par machine. Ces auteurs discutent aussi la possibilité de construire des machines pensantes « thinking machines » [51]. L'année 1950 voit ainsi la publication de plusieurs articles faisant référence à l'idée de machines pensantes, et le plus célèbre article « l'ordinateur et l'intelligence » d'Alan Turing [52] (qui décrit comment déterminer si une machine s'approche d'une intelligence humaine), méthode appelée "le jeu de l'imitation" plus connu aujourd'hui sous le nom de *"Test de turing"*.

Et enfin le développement de l'IA dans la seconde moitié du XXe siècle. Après une conférence tenue à l'été 1956 à Dartmouth College (Hanover, New Hampshire, USA) [53] sur le thème des machines pensantes, certains chercheurs se sont davantage intéressés et organisent plus de rencontres. C'est à l'occasion de ces rencontres que l'expression « artificial intelligence » (défendue par McCarthy) fut utilisée pour la première fois de manière systématique pour désigner le nouveau champ de recherche [54].

Depuis, les recherches sur l'IA ont connu successivement des périodes d'essor et de gel. Dans les années 80, les investissements sur le développement de l'intelligence artificielle reprennent. Au cours de la dernière décennie, l'intelligence artificielle prend un nouveau tournant. La puissance de calcul des ordinateurs, la capacité de stockage, et l'accumulation des données augmentent de façon extraordinaire (Big data). Les améliorations techniques développent ainsi la performance des algorithmes. C'est l'ère du machine learning (ML).

2.4 Apprentissage automatique

2.4.1 Définition

L'apprentissage automatique ou machine learning (ML) est une technologie d'intelligence artificielle qui permet à une machine d'évoluer de manière appropriée aux situations inconnues, pouvant prendre des décisions de manière indépendante, et le tout par la capacité " d'apprendre " à partir des données d'entrées et sans être explicitement programmés pour cela [55]. La machine

peut apprendre comme un être humain, une imitation presque parfaite grâce aux méthodes du ML.

Le machine learning est constitué d'un ensemble d'algorithmes avancés par lesquels le traitement de données de valeur parmi d'immenses sources d'information complexe est rendu possible, et ce, sans avoir à faire appel aux humains.

2.4.2 Approches de l'apprentissage automatique

En général, deux grandes approches en apprentissage automatique sont utilisées aujourd'hui : l'apprentissage non supervisé et l'apprentissage supervisé (voir la Figure 2.1 [56]).

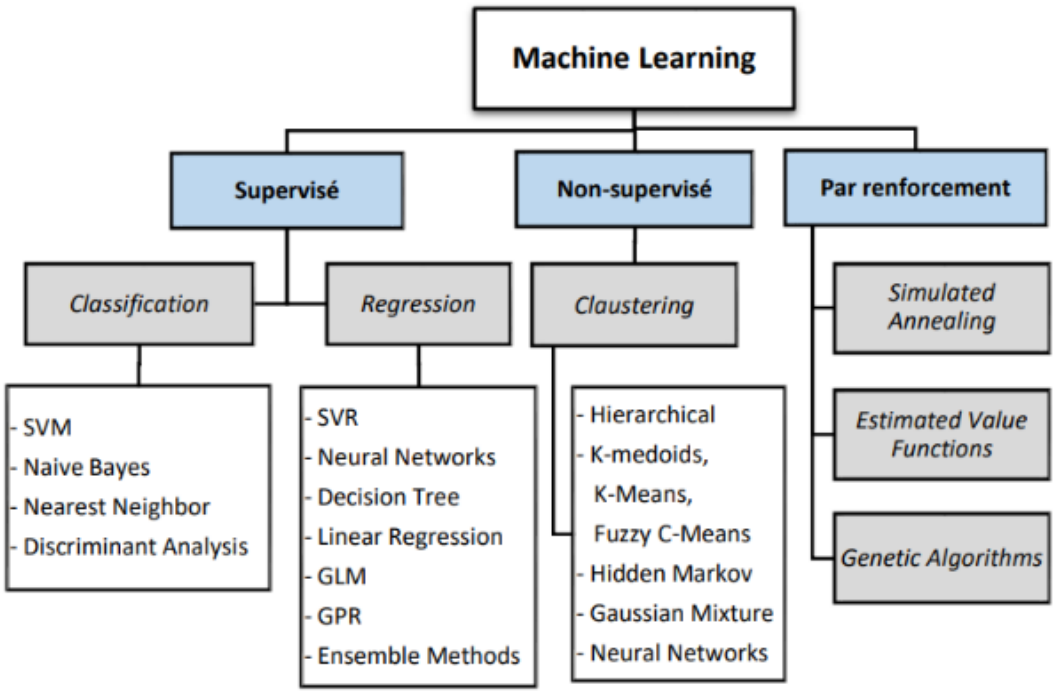

FIGURE 2.1 – Différentes approches de l'apprentissage automatique [56]

2.4.2.1 Apprentissage supervisé

L'apprentissage supervisé est la méthode la plus utilisée en Machine Learning. On parle de celle-ci lorsque l'on fournit à une machine beaucoup d'exemples, c'est-à-dire, un jeu de données prédéfinis, qu'elle doit étudier. La machine est « entraînée ». Elle crée un modèle capable de prédire les valeurs de réponse quand de nouvelles données lui sont fournies. Un ensemble de données de test est souvent utilisé pour valider le modèle. (Voir Figure 2.2).

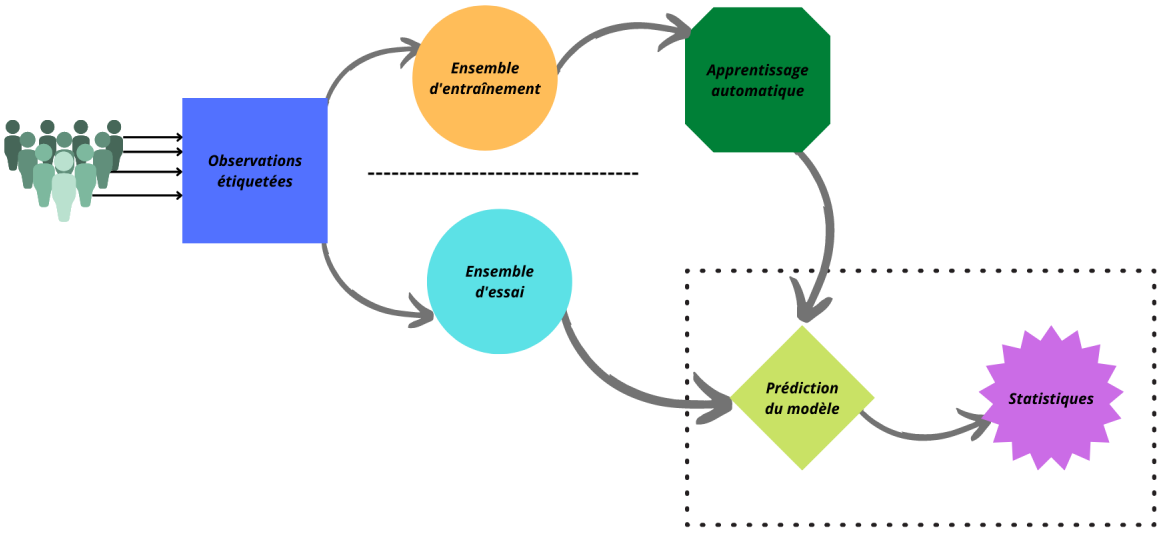

FIGURE 2.2 – Workflow d'un apprentissage supervisé.

Les méthodes d'apprentissage supervisé sont généralement divisées en deux catégories, la classification et la régression. La classification est apparentée par exemple lorsque la valeur de sortie est une catégorie telle que, vrai ou faux. Un problème de régression se pose lorsque la production est une valeur réelle, calculée, comme le prix ou le poids. Ainsi, nous avons les algorithmes de classification et les algorithmes de régression :

2.4.2.1.1 Régression linéaire

En statistique, en économétrie et en apprentissage automatique, un modèle de **régression linéaire** est un modèle qui cherche à établir une relation linéaire entre une variable X (variable explicative, indépendante) et une variable Y (variable expliquée, dépendante).

Pour décrire la relation linéaire entre les deux variables ou bien pour prédire Y pour une valeur donnée de X, on utilise une droite de régression, dont la formule est :

$$Y = \beta_0 + \beta_1 X \tag{2.1}$$

β_0 : L'intercepteur
β_1 : La pente

Les deux éléments (l'intercepteur, la pente) sont appelés coefficients du modèle ou paramètres du modèle.

β_1 représente la valeur de changement de Y si nous changeons X d'une unité.

Donc, fondamentalement, la construction du modèle de régression linéaire simple est la résolution de cette équation pour des valeurs de coefficients du modèle.

En regardant l'exemple de la Figure 2.3 ci-dessous, il est impossible de tracer une droite à travers tous les points de données. Puisque tout modèle statistique n'est qu'une approximation, il y a toujours une erreur, notée ε [57].

Ici, c'est la distance entre le point prédit et le point observé qui est l'erreur ε. Elle est aussi appelée perturbation.

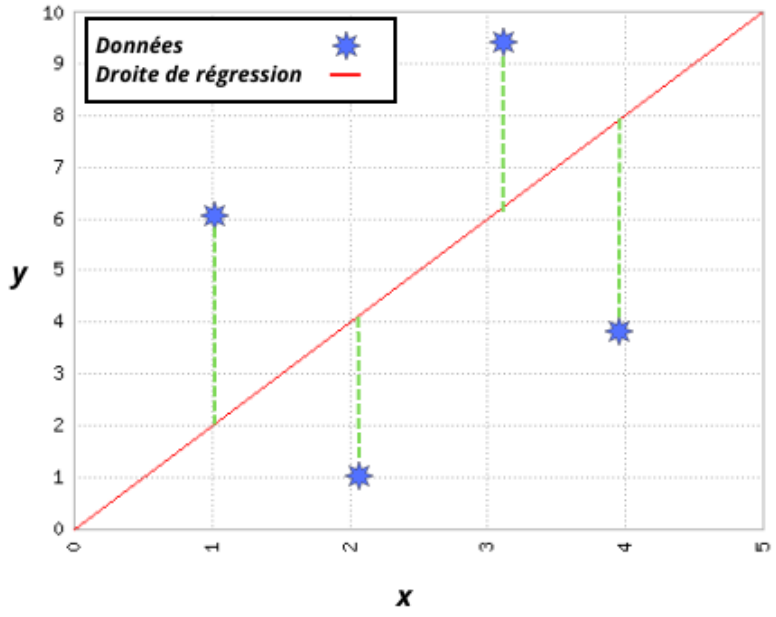

FIGURE 2.3 – Un modèle de régression linéaire simple.

On dit que la machine apprend quand elle trouve quels sont les coefficients du modèle qui minimisent l'erreur. La régression linéaire utilise les méthodes des moindres carrés ou de descente de gradient pour trouver les meilleurs coefficients du modèle. La méthode des moindres carrés y parvient en minimisant la somme de l'erreur quadratique entre les valeurs ajustées et réelles de chaque observation dans les données d'apprentissage. La descente de gradient trouve les paramètres optimaux du modèle en mettant à jour les paramètres à chaque itération.

2.4.2.1.2 Régression linéaire multiple

On distingue le plus souvent deux types d'algorithmes de régression linéaire : la régression linéaire simple et la **régression linéaire multiple**. La régression linéaire multiple est une généralisation, à p variables explicatives, de la régression linéaire simple. Dans ce cas précis, il est possible d'accéder à plusieurs informations en partant d'un facteur unique. Nous supposons donc que les données collectées suivent le modèle suivant :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p \tag{2.2}$$

p : Nombre de fonctionnalités

Dans une régression multiple, il se peut que le nombre p de variables disponibles soit grand. Cette quantité d'information est parfois superflue ou redondante. Ainsi, la diminution du nombre de variables réellement intéressantes dans la régression est envisageable. Soit on part du modèle complet et on retire des variables, soit on part d'une régression simple et on ajoute des variables qui enrichissent le modèle [58].

2.4.2.1.3 Arbre de décision

Les arbres de décision (Decision Tree) DT sont des algorithmes flexibles et puissants pour les problèmes de classification et de régression. Ils s'adaptent aux données complexes et peuvent résoudre des problèmes linéaires ou non-linéaires.

Comme son nom l'indique, la structure de ces algorithmes ressemble à des arbres constitués de nœuds, de branches et de feuilles (Voir Figure 2.4 [59]). Chacun des nœuds constituant l'arbre représente une règle de classification préalablement déterminée. Généralement, la décision pars d'un nœud d'où découlent plusieurs résultats possibles. Chacun de ces résultats mène à d'autres nœuds, d'où émanent d'autres possibilités.

Essentiellement, ils apprennent une hiérarchie de questions if/else, menant à une décision.

Les arbres de décision sont souvent utilisés, car ils sont très interprétables. Ils peuvent être utiles avec ou sans données concrètes, et les données ne nécessitent pas beaucoup de traitement [60].

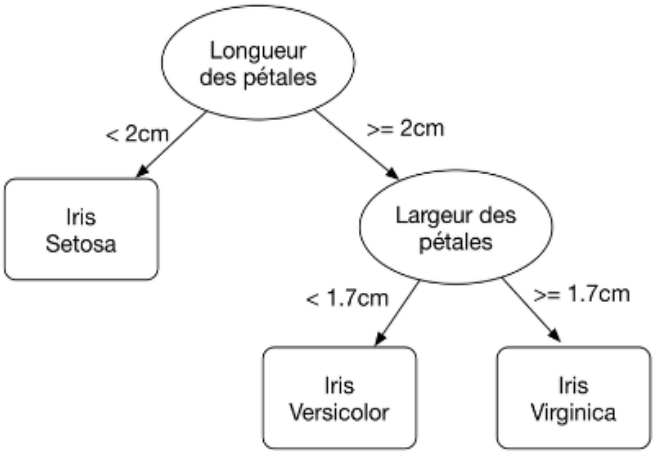

FIGURE 2.4 – Exemple d'arbre de décision [59].

Il existe deux principaux types d'arbres de décision :

— Les arbres de régression (Regression Tree) permettent de prédire une quantité réelle, une valeur numérique.
— Les arbres de classification (Classification Tree) permettent de prédire à quelle classe la variable de sortie appartient.

Ce modèle très populaire a donné naissance à des algorithmes puissants tels que XGBoost ou Random Forest. (Voir Figure 2.5 [60])

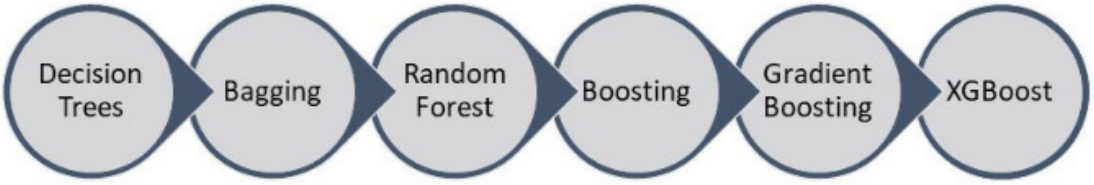

FIGURE 2.5 – Évolution de l'arbre de décision [60].

2.4.2.1.4 Random Forest

Random Forest (RF) est un algorithme d'apprentissage supervisé pour les tâches de classification et de régression. Comme son nom l'indique, forêt aléatoire, un RF crée un ensemble (une forêt) avec plusieurs arbres de décision aléatoires.

Fondamentalement, l'algorithme de forêt aléatoire construit de nombreux arbres de décision, même si chacun d'eux suradapte les données, il prend la moyenne de leurs résultats pour réduire le surajustement (Voir Figure 2.6 [61]).

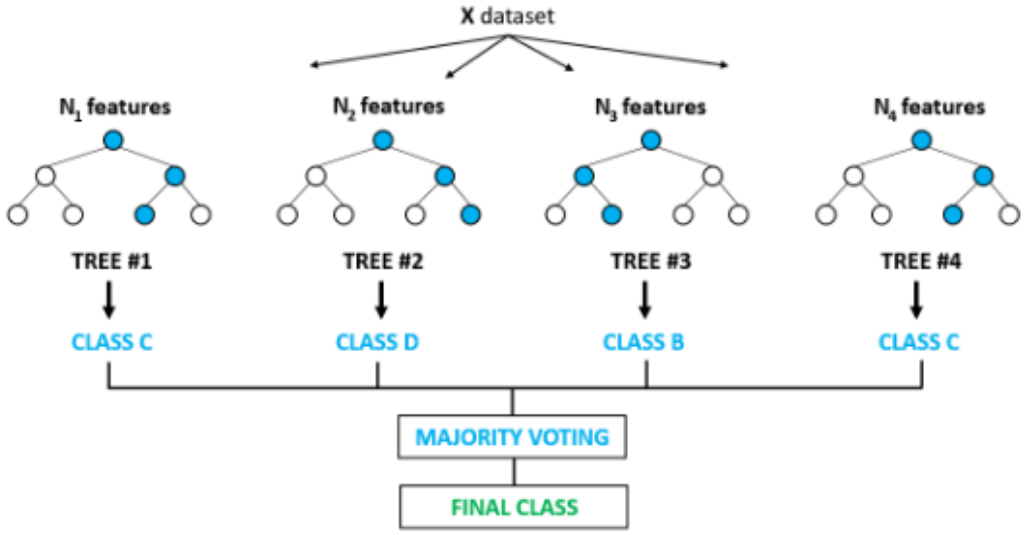

FIGURE 2.6 – Fonctionnement de Random Forest [61].

2.4.2.1.5 XGBoost

XGBoost (ou contraction de eXtreme Gradient Boosting) apparu en 2015, est l'un des modèles de machine Learning les plus efficaces. Utilisé pour la régression ou la classification [62].

XGBoost est basé sur l'apprentissage d'ensemble séquentiel et les arbres de décision. Comme son nom l'indique, il utilise le boosting de gradient.

Le Gradient Boosting est un algorithme particulier de Boosting. Sa particularité est que dans la classification, l'actualisation des poids se calcule de la même façon que la descente de gradient stochastique, et dans la régression, la fonction de coût globale aura aussi la même structure que la descente de gradient stochastique.

Le Gradient Boosting est la plupart du temps utilisé avec des algorithmes d'Arbre de Décision utilisés dans un ordre déterminé (considérés comme des « weak learners »). Pour une observation,

chaque arbre donne un résultat, et la prédiction finale est obtenue en additionnant chacune des valeurs obtenues par les arbres.

Le Boosting quant à lui consiste à assembler plusieurs « weak learners » pour en faire un « strong learner », c'est-à-dire assembler plusieurs algorithmes ayant une performance peu élevée pour en créer un beaucoup plus efficace et plus satisfaisant.

La particularité d'XGBoost réside dans le type de "weak learner" utilisé. Les "weak learners" sont des arbres décisionnels. Les arbres qui ne sont pas assez bons sont "élagués", c'est-à-dire qu'on leur coupe des branches, jusqu'à ce qu'ils soient suffisamment performant. Sinon ils sont complètement supprimés. Cette méthode est appelée le "pruning" (élagage). Ainsi, XGBoost s'assure de ne conserver que de bons weak learners.

De plus, XGBoost est informatiquement optimisé pour rendre les différents calculs nécessaires à l'application d'un Gradient Boosting rapides et il propose un panel d'hyperparamètres très important. Il est ainsi possible grâce à cette diversité de paramètres, d'avoir un contrôle total sur l'implémentation du Gradient Boosting (voir la Figure 2.7 [63]).

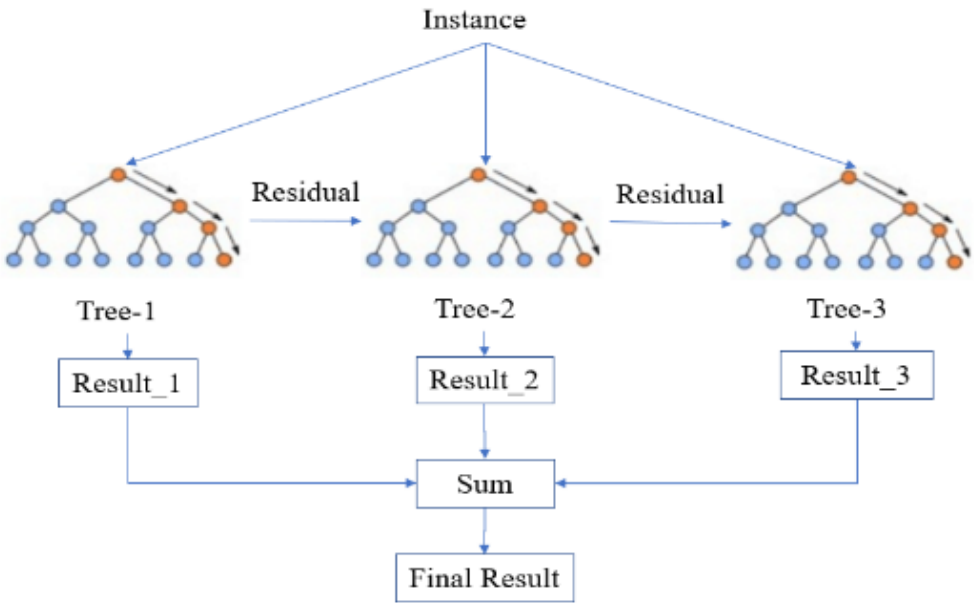

FIGURE 2.7 – Structure simplifiée de XGBoost [63].

Les inconvénients de XGBoost est que comme pour tous les autres algorithmes basés sur des arbres de décision, il faut faire attention à l'overfitting (sur-apprentissage). Pour cela XGBoost n'est pas le meilleur dans le cas de jeux de données très volumineux. Il nécessite plus de temps pour s'entraîner, et n'est pas facile à interpréter.

2.4.2.1.6 Les K-plus proches voisins

K-Nearest Neighbors (KNN), ou les k-plus proches voisins, est l'un des algorithmes le plus simple d'apprentissage automatique supervisé. Il permet à la fois de résoudre les problèmes de classification et de régression.

C'est une méthode non paramétrique dans laquelle le modèle mémorise les observations de l'ensemble d'apprentissage pour la classification de données de l'ensemble test.

L'algorithme KNN est qualifié comme paresseux (lazy learning) car il n'apprend rien pendant la phase d'entraînement. Pour prédire la classe d'une nouvelle donnée d'entrée, il va chercher ses K voisins les plus proches (en utilisant la distance euclidienne ou autres) et choisira la classe des voisins majoritaires.

Le choix du paramètre K est très crucial dans cet algorithme et le meilleur choix dépend des données.

Dans la Figure 2.8, si on choisit K=3, l'algorithme cherche les trois plus proches voisins de l'étoile rouge, pour pouvoir soit le classer dans la classe A ou soit dans la classe B. Dans ce cas, les trois plus proches voisins de l'étoile rouge sont deux cercles de la classe B et un cercle de la classe A. Par conséquent, l'algorithme classera l'étoile rouge dans la classe B.

Par contre, si K=6, les six plus proches voisins de l'étoile rouge sont quatre cercles de la classe A et deux cercles de la classe B. Dans cet exemple, l'algorithme classera l'étoile rouge dans la classe A.

En général, des valeurs plus élevées de k réduisent l'influence du bruit sur la classification [64].

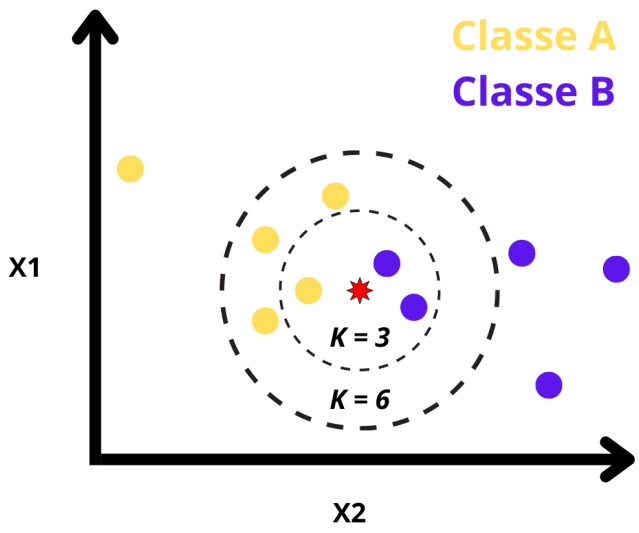

FIGURE 2.8 – Exemple de classification par l'algorithme KNN.

2.4.2.1.7 Machine à vecteurs de support (SVM)

SVM (Support Vector Machine ou Machine à vecteurs de support) est une famille d'algorithmes d'apprentissage automatique qui permettent de résoudre des problèmes tant de classification que de régression.

Les machines à vecteurs de support sont utilisées lorsque les données ont exactement deux classes. L'objectif de SVM est de mapper l'ensemble de données d'entrée dans un espace de grande dimension et de créer une frontière de décision (hyperplan de séparation) en apprenant à classer correctement les classes. L'hyperplan peut être défini comme la ligne linéaire dans un espace de grande dimension.

L'hyperplan choisi doit maximiser sa distance avec les données des deux classes les plus proches de l'hyperplan (Voir la Figure 2.9). Les données les plus proches de l'hyperplan sont d'ailleurs appelées vecteurs support et la distance la marge. L'hyperplan avec une marge maximale est appelé l'hyperplan optimal [65].

L'hyperplan peut être linéaire (classificateur linéaire) le SVM est donc dit linéaire. Comme il peut être non linéaire (classificateur non linéaire).

SVM linéaire est utilisé pour les données séparables linéairement, ce qui signifie que si un ensemble de données peut être classé en deux classes en utilisant une seule ligne droite.

SVM non linéaire est utilisé pour les données séparées non linéairement, ce qui signifie que si un ensemble de données ne peut pas être classé à l'aide d'une ligne droite, il existe plusieurs méthodes d'adaptation des SVM aux problèmes multi classes tels que les méthodes one vs one, one vs all etc...

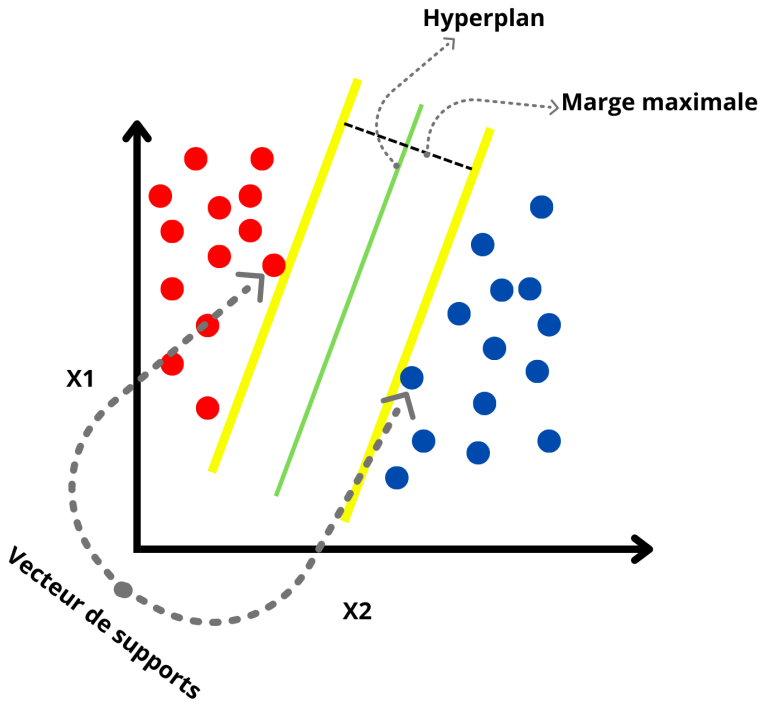

FIGURE 2.9 – Exemple de machine à vecteurs de support.

2.4.2.1.8 Réseaux de neurones artificiels

Inspiré des réseaux de neurones du cerveau humain, les réseaux de neurones artificiels ou Artificial Neural Networks (ANNs) sont des modèles bien plus complexes que tous les autres modèles du ML. Ils se composent d'un certain nombre de nœuds de traitement interconnectés appelés neurones. Les neurones sont généralement organisés en une séquence de couches, comprenant une couche d'entrée, une seule ou un ensemble de couches intermédiaires, et une couche de sortie.

La couche d'entrée reçoit les données d'entrée du réseau, mais n'effectue aucun calcul. La couche de sortie donne la réponse du réseau à l'entrée spécifiée. Les couches intermédiaires, ou généralement appelées couches cachées, sont généralement connectées aux couches d'entrée et de sortie (Voir Figure 2.10).

Chaque nœud, ou neurone artificiel, de la couche cachée, se connecte à un autre et possède un poids et un seuil associés. Si la sortie d'un nœud est supérieure à la valeur du seuil spécifiée, ce nœud est activé et envoie des données à la couche suivante du réseau. Sinon, aucune donnée n'est transmise à la couche suivante du réseau. La sortie de chaque neurone du réseau de neurones utilise généralement la fonction d'activation sigmoïde suivante :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Il existe cependant d'autres fonctions d'activation qui peuvent être utilisées dans les réseaux de neurones, telles que la gaussienne, la tangente hyperbolique, le seuil linéaire, et même une simple fonction linéaire.

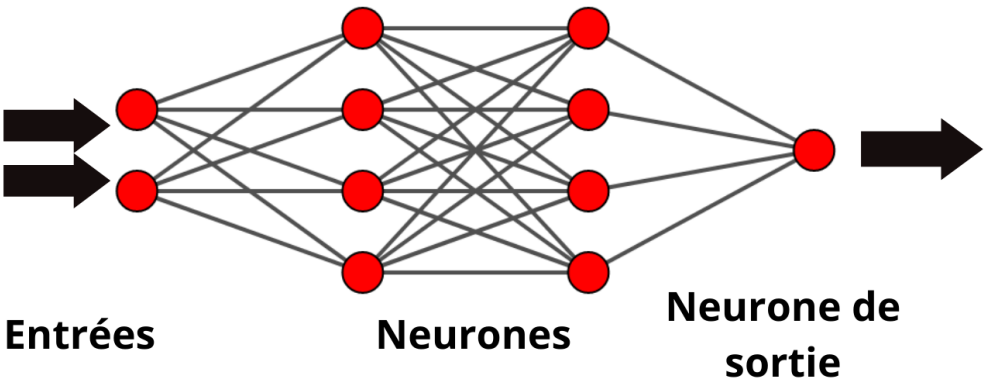

FIGURE 2.10 – Exemple de réseaux de neurones artificiels.

Les réseaux de neurones artificiels sont généralement adoptés pour les problèmes de régression et de classification. Le développement du modèle se fait en deux étapes : apprentissage et test.

Le ANN est d'abord formé ou alimenté en grandes quantités de données. La formation consiste à fournir des données d'entrée et à indiquer au réseau ce que doit être la sortie. Chaque entrée est accompagnée de l'identification correspondante. Le fait de fournir les réponses permet au modèle d'ajuster ses pondérations internes pour apprendre à mieux faire son travail. En définissant les règles et en faisant des déterminations les réseaux neuronaux utilisent plusieurs principes. Ceux-ci comprennent l'entraînement par gradient, la logique floue, les algorithmes génétiques, et les méthodes bayésiennes [66].

D'autres techniques basées sur l'ANN sont les techniques d'apprentissage en profondeur (deep learning) ou ANN purement multicouches. L'apprentissage en profondeur élimine une partie du prétraitement des données qui est généralement impliqué dans l'apprentissage. Ces données peuvent être non structurées, telles que du texte et des images. Le deep learning peut déterminer quelles caractéristiques sont les plus importantes. Alors qu'en machine learning, cette hiérarchie de fonctionnalités est établie manuellement par un expert humain.

2.4.2.2 Apprentissage non supervisé

La technique de l'**apprentissage non supervisé** (ou unsupervised learning) consiste à entraîner des modèles, sans réaliser d'étiquetage manuel ou automatique des données au préalable. Le programme reçoit directement des jeux de données et doit y trouver des modèles et des relations.

Les algorithmes regroupent les données en fonction de leur similitude, et éloignent ceux qui ont le moins de caractéristiques communes, sans aucune intervention humaine. L'apprentissage par la machine se fait de façon totalement autonome.

Les tâches non supervisées les plus courantes sont le clustering, dues au fait que les clusters (labels) ne sont pas connus à l'avance.

L'apprentissage non supervisé est fréquemment appliqué dans l'analyse de graphes, à savoir les systèmes de recommandation... Pour la prédiction de la confiance dans les réseaux sociaux, l'apprentissage non supervisé est la solution la plus populaire. Paramétrer la confiance au préalable est une tâche complexe pour l'homme, contrairement à la machine qui n'est pas subjective.

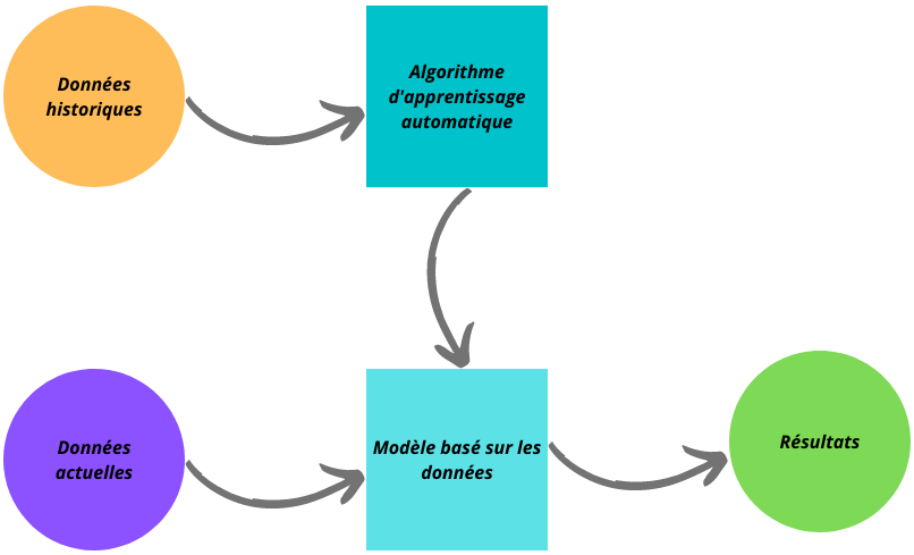

FIGURE 2.11 – Apprentissage automatique non supervisé.

2.5 Conclusion

Dans ce chapitre, nous avons défini ce qu'est l'intelligence artificielle et vu partiellement son panorama historique. Puis nous nous sommes plus concentré sur la description de ses approches d'apprentissage automatique, plus particulièrement les différents algorithmes d'apprentissage supervisé, qui nous serviront dans les chapitres qui suivent, pour la prédiction de la confiance dans les réseaux sociaux.

Chapitre 3

Description et prétraitement des données

3.1 Introduction

Dans ce chapitre, nous allons présenter une description des jeux de données, de l'outil Python et les configurations nécessaires de l'environnement, utilisés pour la prédiction de la confiance dans les réseaux sociaux.

Ensuite, nous allons expliquer l'étape de prétraitement des données, après l'extraction de leurs informations pertinentes, ici les différentes caractéristiques de la confiance. Une étape essentielle avant de passer à la phase d'apprentissage.

3.2 Environnement de développement

3.2.1 Environnement matériel

Les expériences ont été réalisées sur un ordinateur portable dont les caractéristiques de l'environnement matériel sont rapportées dans le Tableau 3.1.

	caractéristiques
CPU	Intel® Core™ i3-M380 CPU 2.53 GHz
RAM	8.00 Go
HDD	512 Go

Tableau 3.1 – Caractéristiques de l'environnement matériel utilisé.

3.2.2 Environnement logiciel

De nombreux outils logiciels sont en développement depuis 25 ans dont l'objectif commun est de faciliter le processus complexe d'analyse des données et de proposer des environnements intégrés en plus des langages de programmation standard. Un certain nombre d'entre eux sont orientés vers le traitement rapide et le streaming de données à grande échelle, tandis que d'autres sont spécialisés dans l'implémentation des algorithmes de ML.
Les différents outils logiciels utilisés durant ce projet sont fournis par Anaconda (voir Figure 3.1). Il s'agit d'un environnement logiciel open source de développement d'applications dédié à la science des données et à l'apprentissage automatique (Langages Python et R, outils PyCharm et Spyder et Jupyter, Bibliothèque de ML, etc...).

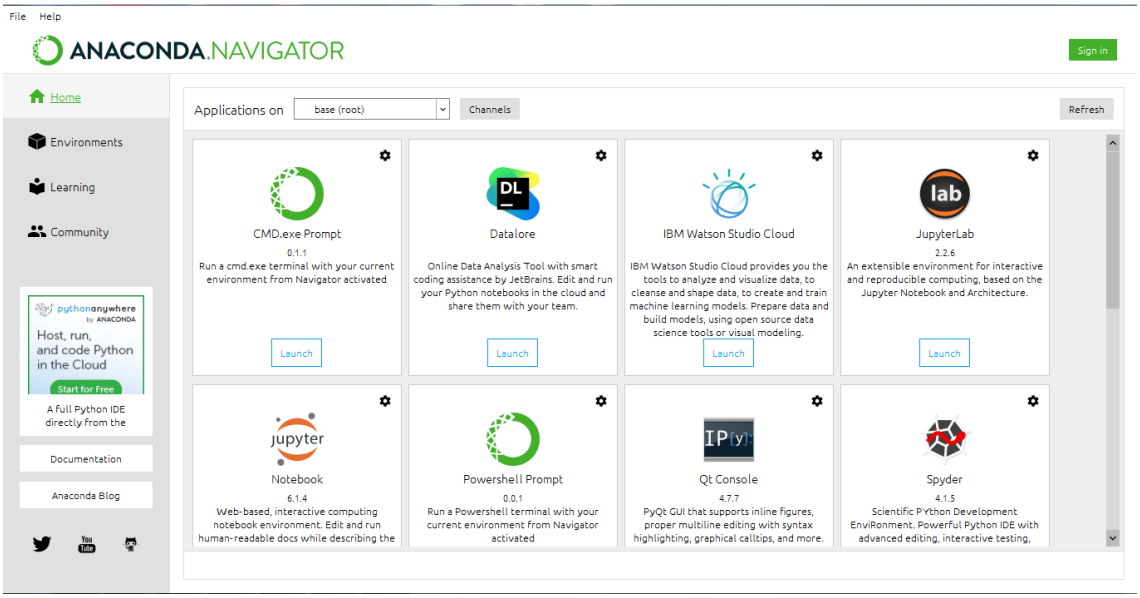

FIGURE 3.1 – Environnement Ananconda

.

Les outils utilisés pour la réalisation de la phase de prétraitement des données, l'analyse des données, ainsi que la phase d'apprentissage automatique supervisée, sont :

3.2.2.1 Python

Python est un langage de programmation interprété, utilisant le concept d'orientée objet et de haut niveau avec une sémantique dynamique. La combinaison entre l'intégration de ces structures de données de haut niveau et le typage dynamique, à la liaison dynamique le rendent très attrayant pour le développement rapide d'applications, ainsi que pour une utilisation en tant que langage de script ou en tant que liaison des composants existants entre eux [67].

3.2.2.2 NetworkX

NetworkX est un package Python qui permet d'analyser des réseaux de graphes de façon complexe. Les fonctionnalités de NetworkX requièrent des connaissances en théorie des graphes. Le package traite avec des graphes et contient des fonctions de navigation des bords et des nœuds pour découvrir et comprendre des relations complexes et/ou optimiser les chemins entre les données liées dans un réseau [68].

3.2.2.3 Pandas

Pandas est un package Python concevant le travail avec des données « relationnelles » ou « étiquetées » à la fois basiques et intuitives en fournissant des structures de données instantanées, flexibles et expressives. Visant à être le composant de construction fondamentale de haut niveau pour effectuer une analyse de données pratique et réelle en Python.

3.2.2.4 Numpy

Numpy est le package principal pour le calcul scientifique en Python. Une bibliothèque Python permettant de fournir un objet tableau de dimension multiple, divers objets dérivés (tels que des tableaux masqués et des matrices) et une collections de fonctions pour des opérations efficaces sur des tableaux, y compris l'arithmétiques, tri, E/S, logiques, gestion de forme, sélection, algèbre linéaire de base, opérations statistiques de base, simulation aléatoire. . .etc.

3.2.2.5 XGBoost

XGBoost qui signifie Extrême Gradient Boosting, est une bibliothèque d'apprentissage automatique qui ne cesse de s'améliorer et distribuée à arbre de décision à gradient optimisé (Gradient-Boosted Decision Trees GBDT). Elle est basée sur le développement d'arbre parallèle et connu pour être la principale bibliothèque d'apprentissage automatique pour les problèmes de régression, de classification et de classement [69].

3.2.2.6 Scikit-learn

Scikit-learn ou Sklean est une bibliothèque d'apprentissage automatique open source qui est utilisée principalement dans l'apprentissage supervisé et non supervisé. Il fournit également divers outils pour l'adaptation du modèle, le prétraitement des données, la sélection du modèle, l'estimation d'un modèle et inclut en plus de nombreux autres utilitaires.

3.2.2.7 Matplotlib

Matplotlib est une bibliothèque riche et variée. Son but consiste à créer des visualisations statiques, animées et interactives en Python. Matplotlib permet une interprétation facile des statistiques et rend les choses difficiles à comprendre possibles.

3.2.2.8 Scipy

Scipy est une panoplie d'algorithmes mathématiques et de fonctions pratiques construites sur l'extension NumPy du langage de programmation Python. Il s'accompagne d'une puissance significative à la session interactive de Python en fournissant à l'utilisateur des fonctions et des classes de haut niveau afin de manipuler et de visualiser les données.

3.2.2.9 Statistics

Statistics est un module fournissant des fonctions de calcul de statistiques mathématiques de données numériques.

3.3 Description des jeux de données

Un jeu de données est un ensemble ordonné ou une collection de données cohérentes, qui est associée à un attribut ou une observation particulière et qui est représentable sous différents formats (vidéos, images, texte, son, etc.).

Les ensembles de données ont un rôle très important et crucial lors du processus d'apprentissage automatique, du fait que les données générées sur l'ensemble des données aideront à former notre modèle avec les données d'apprentissages afin d'évaluer ses performances avec les données de validation et de test.

À travers les réseaux sociaux, notre recherche nous a amené à recueillir plusieurs jeux de données afin de représenter le mieux possible la mesure de confiance entre les utilisateurs d'un réseau social. Nous avons réuni 4 jeux de données de divers sites web, qui seront utilisés dans notre travail : **Bitcoin-Alpha**, **Bitcoin-OTC**, **Wikipedia-Rfa** et **Advogato**.

Ces jeux de données contiennent généralement trois colonnes :

— **Index** : Index de la ligne.
— **Source** : Représente le numéro d'un utilisateur qui fait confiance (trustor).
— **Cible** : Représente le numéro d'un utilisateur à qui l'ont fait confiance (trustee).
— **Poids** : Représente la mesure de confiance donnée par la **Source** vers la **Cible**.

Dans le Tableau 3.2, en prenant exemple de la ligne dont l'index = 0, on a comme information que le nœud source avec le numéro d'utilisateur 2 fait confiance à la cible avec le numéro d'utilisateur 402 avec une mesure de 0.1. Cela signifie un manque de confiance du trustor envers le trustee (mais pas de méfiance pour autant).

Index	Source	Cible	Poids
0	2	402	0.1
1	10	970	0.8
2	113	54	0.4
3	10	271	0.8
4	119	2	0.8
....	....	....	....
24181	281	3450	0.1
24182	1202	604	0.1
24183	114	7370	-0.1
24184	3451	98	0.5
24185	15	3451	0.1

TABLEAU 3.2 – Jeu de données représentant la mesure de confiance entre utilisateurs dans le réseau social **Bitcoin-Alpha**.

Une représentation en forme de graphe orienté est nécessaire afin d'exploiter le mieux possible notre jeu de données.

Les graphes sont des structures mathématiques utilisées pour modéliser de nombreux types de relations et de processus dans les systèmes physiques, biologiques, sociaux et d'information [68]. Un graphe orienté est formé de deux ensembles :
Un ensemble $X = x_1, x_2...x_n$ dont les éléments sont appelés sommets, un ensemble $A = a_1, a_2...a_m$, partie du produit cartésien $X \times X$, dont les éléments sont appelés arcs. On notera $G = (X, A)$.

Si $a = (x, y)$ est un arc du graphe G, x est l'extrémité initiale de a et y l'extrémité finale de a [70].

Avec cette représentation, plusieurs caractéristiques globales peuvent être calculées afin de catégoriser les jeux de données :

3.3.1 Distribution empirique des poids

Une distribution empirique des poids sert à décrire la répartition de la valeur de confiance données par les utilisateurs dans un réseau social. Selon les résultats obtenus, si la répartition est plutôt positive, cela pourrait représenter une forte naïveté de la part des utilisateurs ou que l'environnement du réseau social est sain.

Tandis, qu'une grande méfiance ou un environnement toxique dans un réseau social serait déduit par une forte répartition négative. En conséquence, l'interprétation des données en termes de distribution empirique des poids reste ambiguë (voir Figure 3.2).

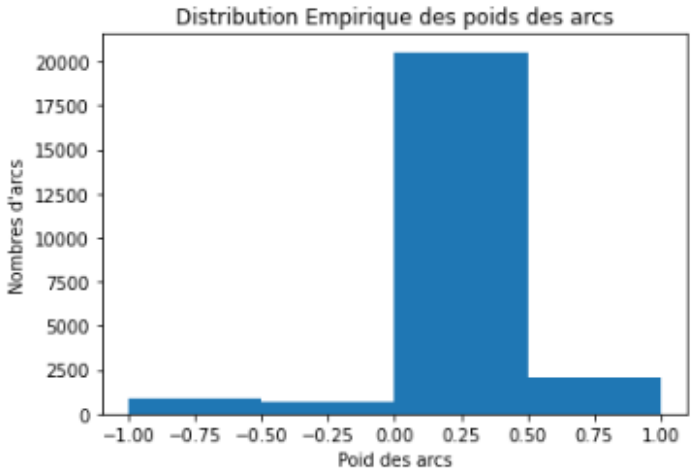

FIGURE 3.2 – Représentation de la distribution empirique des poids des arcs du graphe orienté utilisant le jeu de données **Bitcoin-Alpha**

.

3.3.2 Caractéristiques principales des graphes

Chaque graphe orienté utilisant les jeux de données des réseaux sociaux contiennent des caractéristiques principales tel que le nombre de nœuds, le nombre d'arcs, l'intervalle de la mesure de confiance (Voir le Tableau 3.3).

Jeu de données	**Nombre de noeuds**	**Nombres d'arcs**	**Taux de réciprocité (%)**	**Intervalle mesure de confiance**	**Description**
BTC-Alpha Net	3783	24 186	59.57	[-1,1]	Confiance de l'utilisateur bitcoin P à l'utilisateur Q.
BTC-OTC Net	5881	35 592	56.89	[-1,1]	Confiance de l'utilisateur bitcoin P à l'utilisateur Q.
Advogato	5417	51 312	16.89	[0.1,0.9]	Confiance de l'utilisateur Advogato P à l'utilisateur Q.
RFA Net	9654	104 554	0.07	[-1,1]	Soutien ou opposition à l'élection d'un utilisateur Q en tant qu'administrateur Wikipédia, par un autre utilisateur P.

Tableau 3.3 – Caractéristiques principales de chaque jeu de données.

3.4 Prétraitement des données

Le jeu de données est divisé aléatoirement en trois parties indépendantes avec chacune son propre objectif spécifique.

La partie apprentissage, les données appelées données d'apprentissages sont utilisées pour apprendre les exemples d'apprentissages en ajustant les paramètres du modèle.

La partie validation, les données appelées données de validation, permettent d'ajuster les hyper-paramètres du modèle afin d'avoir à généraliser sur les données de test, tout en gardant les meilleures performances du modèle.

Et finalement dans la partie test, les données appelées "données de test" sont utilisées pour calculer les performances réelles du modèle.

Afin d'aboutir à de meilleures performances des algorithmes d'apprentissage supervisé, des fonctionnalités (features) pertinentes seront nécessaires, càd une représentation des caractéristiques ayant pour but de guider notre modèle à généraliser sur l'ensemble de tests. De ce fait, l'extraction de nouvelles fonctionnalités à partir des ces jeux de données est primordiale pour appliquer un modèle d'apprentissage supervisé.

Du fait que nous avons déjà appréhendé les concepts de la confiance dans le Chapitre 1, nous pouvons d'ores et déjà choisir des caractéristiques qui pourront influencer positivement et optimiser le modèle le plus possible. Les caractéristiques de confiance dans les réseaux sociaux que nous avons choisies d'utiliser sont : la **notoriété**, la **réputation** , l'**appréciation personnelle**, la **sociabilité**.

3.4.1 Caractéristiques de la confiance

La **notoriété** est la possibilité pour les internautes de trouver des informations nous concernant. Cela signifie que les gens remarquent notre présence (notoriété publique). Il semblerait que la notoriété soit un moyen efficace d'assurer un jugement concret sur le degré de confiance et de méfiance pour une personne, car qui dit grande notoriété dit plusieurs avis sur la personne [71].

Cependant, la notoriété ne suffit pas à juger une personne, du fait qu'une personne peut avoir une bonne, comme une mauvaise notoriété dans les réseaux sociaux, c'est exactement la caractéristique principale de la **réputation**, qu'est le fait que vous soyez connu en bien ou en mal. Grande réputation, bonne réputation, mauvaise réputation.

La **sociabilité** est l'une des caractéristiques les plus importantes dans l'élaboration des fonctionnalités, sachant que plus une personne est sociable, plus elle a des chances d'être sujet à des expériences soit positives, soit négatives. C'est précisément l'attribut définissant le plus l'**appréciation personnelle**, qui est de pouvoir estimer la relation interpersonnelle avec un individu, lui faire confiance ou bien le contraire, en être méfiant, tout cela dépend de l'expérience vécue.

3.4.2 Création des fonctionnalités

Dans ce qui suit, nous utiliserons de simple notation pour représenter la structure du graphe orienté :

— **U** : pour désigner le nœud source de la ligne en cours dans le jeu de données (trustor).
— **V** : pour désigner le nœud cible de la ligne en cours dans le jeu de données (trustee).

— **(U,V)** : pour désigner l'arc allant du nœud source vers le nœud cible de la ligne en cours dans le jeu de données.
— **W(U,V)** : pour désigner le poids de l'arc allant du nœud source vers le nœud cible de la ligne en cours dans le jeu de données.

Afin de pouvoir intégrer ces caractéristiques de la confiance, nous devons déterminer leurs spécification dans les réseaux sociaux :

— **Notoriété** : nombre d'utilisateurs de l'entourage donnant une note d'appréciation à un utilisateur spécifique.
— **Réputation** : moyenne des mesures de confiance données par les utilisateurs de l'entourage envers un utilisateur spécifique.

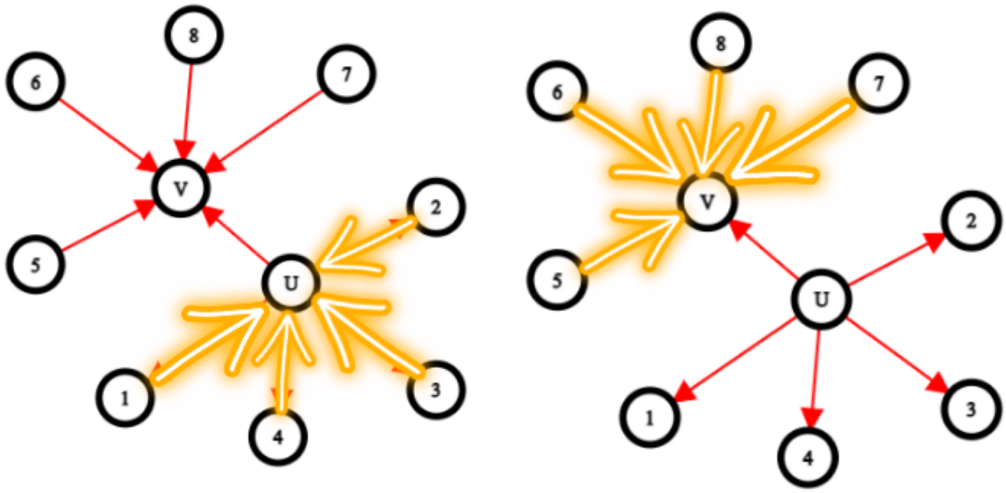

FIGURE 3.3 – Exemple de notoriété et de réputation dans un graphe ordonné. Le graphe à gauche désigne la notoriété et la réputation du nœud **U**. Le graphe à droite désigne la notoriété et la réputation du nœud **V**.

— **Appréciation personnelle** : moyenne des mesures de confiance que donne l'utilisateur aux autres utilisateurs de son entourage.
— **Sociabilité** : nombre de personnes qu'un utilisateur a su noter dans son entourage.

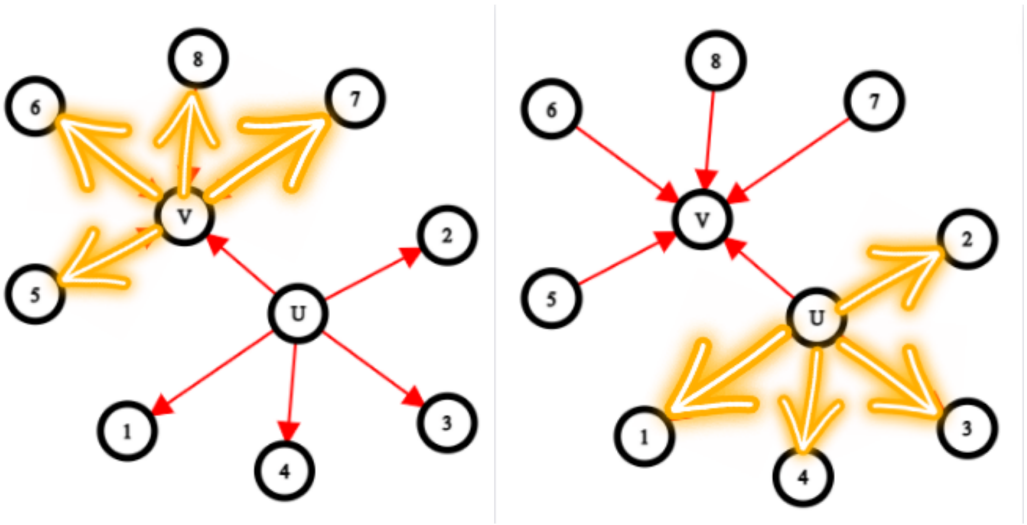

FIGURE 3.4 – Exemple d'appréciation personnelle et de sociabilité dans un graphe ordonné. Le graphe à gauche désigne l'appréciation personnelle et la sociabilité du nœud **V**. Le graphe à droite désigne l'appréciation personnelle et la sociabilité du nœud **U**.

.

Maintenant que les fonctionnalités ont étaient choisies, la création du nouveau jeu de données pourra aboutir avec comme fonctionnalités les suivantes :

- **U** : Le nœud source.
- **V** : Le nœud cible.
- **U_nbr_degree_filtr_out** $(d_G^+(U)^{-1})$: Le nombre d'arcs sortants du nœud **U**, sauf l'arc **(U,V)**.
- **V_nbr_degree_out** $(d_G^+(V))$: Le nombre d'arcs sortants du nœud **V**.
- **U_nbr_degree_in** $(d_G^-(U))$: Le nombre d'arcs entrants vers le nœud **U**.
- **V_nbr_degree_filtr_in** $(d_G^-(V)^{-1})$: Le nombre d'arcs entrants vers le nœud **V**, sauf l'arc **(U,V)**.
- **U_moy_filtr_out** : La moyenne des arcs sortants du nœud **U**, sauf l'arc **(U,V)**.
- **V_moy_filtr_in** : La moyenne des arcs entrants vers le nœud **V**, sauf l'arc **(U,V)**.
- **U_moy_in** : La moyenne des arcs entrants vers le nœud **U**.
- **V_moy_out** : La moyenne des arcs sortants du nœud **V**.
- **W(U,V)** : Le poids de l'arc **(U,V)**.

Dans la Figure 3.5 une représentation du résultat obtenu après extraction des nouvelles fonctionnalités au réseau BTC-Alpha :

	U	V	U_nbr_degree_filtr_out	V_nbr_degree_out	U_nbr_degree_in	V_nbr_degree_filtr_in	U_moy_filtr_out	V_moy_filtr_in	U_moy_in	V_moy_out	W(U,V)
0	2.0	402.0	0.395112	0.030550	0.513784	0.030075	0.158333	0.204124	0.358537	0.160000	0.1
1	2.0	1127.0	0.395112	0.012220	0.513784	0.012531	0.100000	0.204124	0.358537	0.100000	0.1
2	2.0	168.0	0.395112	0.042770	0.513784	0.037594	0.233333	0.200000	0.358537	-0.085714	0.9
3	2.0	1632.0	0.395112	0.004073	0.513784	0.002506	0.100000	0.203608	0.358537	0.250000	0.2
4	2.0	10.0	0.395112	0.360489	0.513784	0.408521	0.174847	0.203093	0.358537	0.160452	0.3
...	...	...	...	...	...	...	...	...	...	...	...
24181	3446.0	906.0	0.000000	0.034623	0.002506	0.020050	0.100000	0.000000	0.100000	0.076471	0.1
24182	1907.0	499.0	0.000000	0.010183	0.002506	0.010025	0.175000	0.000000	0.300000	0.180000	1.0
24183	3447.0	15.0	0.000000	0.401222	0.002506	0.305764	0.201639	0.000000	0.100000	0.032995	0.3
24184	838.0	7335.0	0.000000	0.034623	0.002506	0.015038	0.150000	0.000000	1.000000	0.517647	-1.0
24185	3451.0	98.0	0.000000	0.075356	0.002506	0.082707	0.206061	0.000000	0.100000	0.202703	0.5

FIGURE 3.5 – Exemple de jeu de données après l'extraction des fonctionnalités du jeu de données **Bitcoin-Alpha**.

3.4.3 Matrice de corrélations

Les matrices de corrélation permettent d'évaluer les dépendances entre plusieurs fonctionnalités à la fois. Le résultat est un Tableau avec des coefficients de corrélation entre les fonctionnalités (voir Figure 3.6). Une faible dépendance entre les fonctionnalités d'un jeu de données est primordiale pour aboutir à de bonnes performances du modèle d'apprentissage supervisé.

	U_nbr_degree_filtr_out	V_nbr_degree_out	U_nbr_degree_in	V_nbr_degree_filtr_in	U_moy_filtr_out	V_moy_filtr_in	U_moy_in	V_moy_out	W(U,V)
U_nbr_degree_filtr_out	1.000000	-0.156390	0.978556	-0.163484	-0.086884	-0.075683	0.144751	-0.020767	-0.056797
V_nbr_degree_out	-0.156390	1.000000	-0.154611	0.980144	0.179635	-0.015106	-0.065907	-0.090157	0.086465
U_nbr_degree_in	0.978556	-0.154611	1.000000	-0.161702	-0.079905	-0.036329	0.134693	-0.015575	-0.039918
V_nbr_degree_filtr_in	-0.163484	0.980144	-0.161702	1.000000	0.166042	-0.015772	-0.069120	-0.059727	0.077089
U_moy_filtr_out	-0.086884	0.179635	-0.079905	0.166042	1.000000	0.078792	0.077626	0.170269	0.392572
V_moy_filtr_in	-0.075683	-0.015106	-0.036329	-0.015772	0.078792	1.000000	0.232777	0.077402	0.237168
U_moy_in	0.144751	-0.065907	0.134693	-0.069120	0.077626	0.232777	1.000000	0.141383	0.127944
V_moy_out	-0.020767	-0.090157	-0.015575	-0.059727	0.170269	0.077402	0.141383	1.000000	0.129868
W(U,V)	-0.056797	0.086465	-0.039918	0.077089	0.392572	0.237168	0.127944	0.129868	1.000000

FIGURE 3.6 – Matrice de corrélation des fonctionnalités du jeu de données **Bitcoin-Alpha** après extraction.

.

D'après cette matrice de corrélation, nous pouvons voir une forte dépendances entre deux fonctionnalités, tels que :

U_nbr_degree_filtr_out à une forte dépendance avec **U_nbr_degree_in** et **V_nbr_degree_filtr_in** à une forte dépendances avec **V_nbr_degree_out**.

De ce fait, retirer l'une de ces fonctionnalités ne diminue pas énormément les performances du modèle. Mais afin d'optimiser notre modèle en terme de précision, toutes les fonctionnalités seront utilisées.

U_nbr_degree_filtr_out et **V_moy_filtr_in** ont une forte dépendance avec la sortie désirée **W(U,V)**, cela veut dire que ces deux fonctionnalités impacteront grandement en améliorant les performances du modèle.

3.4.4 Division du jeu de données

Durant notre phase d'apprentissage automatique, nous passerons par les trois sous-phases, qui sont la phase d'apprentissage, la phase de validation, et la phase de test. En guise d'illustration, nous allons aussi manipuler et utiliser le jeu de données **Bitcoin-Alpha**.

En conséquence, à chaque déroulement d'un algorithme d'apprentissage automatique, nous devons diviser aléatoirement notre nouveau jeu de données en trois parties, ensemble d'entraînements, ensemble de validation, et ensemble de tests (voir la Figure 3.7).

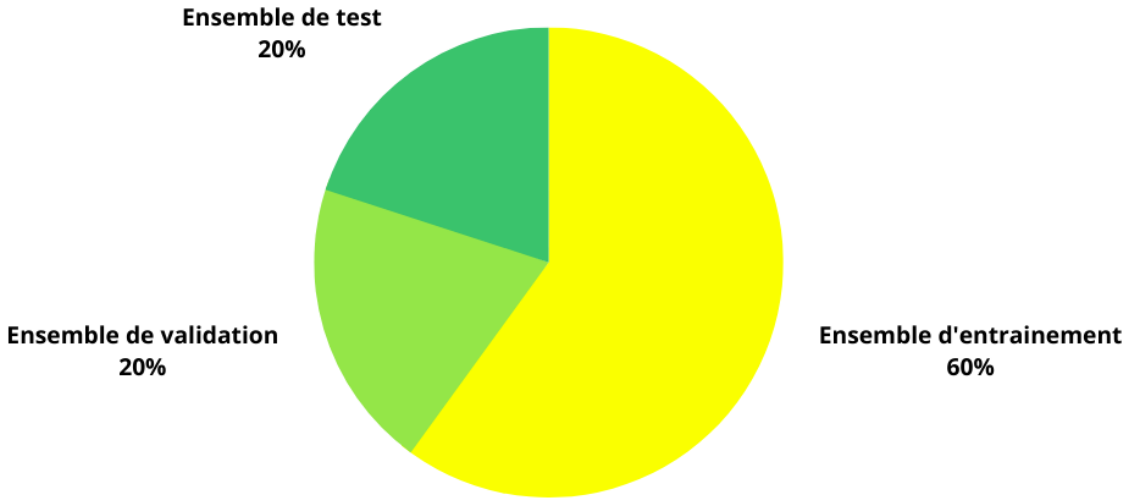

FIGURE 3.7 – Pourcentage standard de la répartition des données pour l'utilisation en apprentissage automatique.

.

3.5 Conclusion

Dans ce chapitre, nous avons décrit l'ensemble de données (jeux de données) initiales et choisi les caractéristiques de la confiance, utilisées pour tester notre approche. Ensuite, nous avons énuméré les différentes étapes de pré-traitement sur l'ensemble de données tout en lui créant de nouvelles fonctionnalités, afin de le préparer aux phases d'apprentissage automatique.

Le chapitre suivant sera consacré à l'application des méthodes d'apprentissage automatique, ainsi qu'à la discussion et à la comparaison des résultats.

Chapitre 4

Application et résultats

4.1 Introduction

Dans ce chapitre, nous allons présenter notre contribution à la prédiction de confiance dans les réseaux sociaux. Pour ce, nous allons mettre en place les différentes méthodes de l'apprentissage supervisé sur les trois phases de l'apprentissage automatique en utilisant les jeux de données, après le prétraitement de ces dernièrs.

Ensuite, nous discuterons les résultats obtenus de nos modèles entraînés, et nous effectuerons également une comparaison entre les performances des modèles générés. Puis, nous comparerons l'un de nos modèles ayant les meilleures performances, avec certains algorithmes classiques utilisés dans la prédiction de la confiance.

Les algorithmes d'apprentissage supervisé que nous allons utiliser sont : l'algorithme de Régression linéaire multiple polynomial, l'algorithme XGBoost, la méthode SVR(Support vector regression) et l'algorithme KNN.

4.2 Phase d'apprentissage Automatique

Après l'extraction des nouvelles fonctionnalités pertinentes utilisant des caractéristiques de la confiance, à partir du jeu de données initial, et la création du nouveau jeu de données, nous pouvons passer à l'étape d'utilisation des méthodes d'apprentissage automatique pour prédire le poids des arcs représentant la mesure de confiance entre deux utilisateurs distincts.

Durant la phase d'apprentissage, et la phase de validation nous manipulons et utilisons le jeu de données **Bitcoin-Alpha**, afin de donner exemple des étapes suivis.

4.2.1 Phase d'apprentissage

La phase d'apprentissage consiste à entraîner notre modèle à apprendre nos exemples d'entraînement en modifiant ses paramètres et en minimisant le plus possible le coût d'erreur entre la sortie désirée(valeur réelle) et la sortie prédite.

L'ensemble d'apprentissage contient 14511 observations et 8 fonctionnalités.

Pour tester les performances des méthodes d'apprentissage automatique, nous utiliserons des algorithmes linéaires tels que la régression linéaire multiple, qui s'améliorera en utilisant le degré polynomial. Ainsi, que les algorithmes non linéaires tels que : SVR, et KNN, l'utilisation des méthodes ensemblistes tel que XGBoost, permettent de prédire des valeurs continues.

4.2.1.1 XGBoost

Vu précédemment dans le Chapitre 2, XGBoost (eXtrême Gradient Boosting) est un algorithme utilisé dans l'apprentissage supervisé pour la régression (prédiction de valeurs continues) et faisant partie des algorithmes **Ensemble learning** impliquant l'utilisation de plusieurs arbres de décisions afin de construire le modèle.

Avant de dérouler l'algorithme quelques annotations sur les hyper-paramètres de l'algorithme XGBoost doivent être énumérés :

- λ**(lambda)** : hyper-paramètre utilisé pour la régularisation du modèle, plus est grande plus la régularisation sera importante [72].
- γ **(gamma)** : hyper-paramètre permettant de limiter le nombre de divisions de nœuds. En soustrayant le gain par rapport à la valeur donnée à gamma, s'elle est positive une division sera possible. Un hyper-paramètre important lors de la régularisation [72].
- **Eta (Learning rate)** : après chaque étape de boosting, nous pouvons directement obtenir le poids des nouvelles fonctionnalités, et Eta réduit le pas permettant la mise à jour du poids des fonctionnalités pour rendre le processus de boosting plus conservateur [72].
- **max_depth** : profondeur maximale d'un arbre. L'augmentation de cette valeur rendra le modèle plus complexe et plus susceptible de sur-ajustement [72].
- **n_estimators** : est le nombre d'arbres boostés par gradient [72].
- **colsample_bytree** : définit le pourcentage d'entités (colonnes) qui sera utilisé pour construire chaque arbre. Dans tous les cas, l'ensemble de fonctionnalités pour chaque arbre est susceptible d'être différent [72].

Cet algorithme sera divisé en 3 étapes différentes :

— **étape 0** : Calcul du 1er estimateur qui est une constante $\hat{y}(0)$ (moyenne des valeurs réelles).
— **étape 1** : calcul de résidus (différence entre y_i et la dernière estimation $\hat{y}(0)$).
— **étape 2** : construction de l'arbre #1 permettant de calculer les estimations $\hat{y}_i(0)$.
- A chaque niveau de l'arbre, on identifie le split optimal défini par le gain maximum.
- Lorsque l'arbre est construit, on calcule les estimations de chacune des feuilles.
- On itère jusqu'à ce qu'il n'y ait plus de split possible pour l'arbre qui est en cours de construction.
— **étape 3** : on calcule les résidus permettant de construire un nouvel arbre, l'arbre #2

— **étape 4** : refaire les étapes 1 à 3 jusqu'à l'itération finale.

Afin d'expliquer au mieux les étapes de l'algorithme XGBoost, nous donnerons comme exemple les étapes de la phase d'apprentissage sur un seul arbre de décision avec un niveau de profondeur de 3 que nous construirons.

1. **étape 0** : 1er estimateur qui est une constante $\hat{y}(0)$:

   Pour initialiser l'estimateur $\hat{y}(0)$, nous devons d'abord l'initialiser a une constance moyennant les sorties désirée de l'ensemble d'entraînement. La valeur prise sera la moyenne des valeurs réelles. On aura $\hat{y}(0) = 0.145$.
2. **étape 1** : calcul de résidus (différence entre y_i et la dernière estimation $\hat{y}(0)$:

   Le calcul des résidus se fait à travers cette formule :

$$residut = y_i - \hat{y}(0) \tag{4.1}$$

   y_i : Les sorties réelles de l'ensemble de test.
   $\hat{y}(0)$: La moyenne des sorties réelles de l'ensemble de test.

   Selon le calcul de tous les résidus de chaque exemple d'entrainement, nous aboutirons à ce tableau ci-dessous illustrant les résidus calculées(voir le Tableau 4.1).

Index	U	V	W(U,V)	Résidus
3216	129	1080	0.5	0.354524
1713	45	431	0.1	-0.045476
6218	155	73	0.1	-0.045476
16167	491	7603	-1.0	-1.145476
23398	603	1009	0.1	-0.045476
....	....	....	....	....
4315	1098	84	0.1	-0.045476
20309	49	545	0.1	-0.045476
611	99	353	0.1	-0.045476
19777	374	186	0.8	0.654524
5351	65	40	0.1	-0.045476

TABLEAU 4.1 – Jeu de données illustrant les résidus calculés pour l'ensemble d'entraînement du jeu de données **Bitcoin-Alpha**.

3. **étape 2** : Construction du 1er arbre à partir des résidus :

   On cherche quel split en X va donner le score maximum, défini par :

$$Score = \underbrace{\frac{\left(\sum(y_i - \hat{y}_i)\right)^2}{\text{Nb Instances } + \lambda}}_{\text{Feuille Gauche}} + \underbrace{\frac{\left(\sum(y_i - \hat{y}_i)\right)^2}{\text{Nb Instances } + \lambda}}_{\text{Feuille droite}} - \underbrace{\frac{\left(\sum(y_i - \hat{y}_i)\right)^2}{\text{Nb Instances } + \lambda}}_{\text{Feuille initiale}} - 2 \cdot \gamma \qquad (4.2)$$

   y_i : Les sorties réelles de l'ensemble de test.
   $\hat{y}_i$: La moyenne des sorties réelles de l'ensemble de test.
   γ : Hyper-paramètre permettant de limiter le nombre de divisions de nœuds.
   λ : Hyper-paramètre utilisé pour la régularisation du modèle.

   Pour calculer ces scores, il nous faut donc calculer les scores de la feuille initiale (F.I), et des feuilles gauche (F.G) et droite (F.D) après split. On modulera également avec les hyper-paramètres λ et γ.

Split du 1er niveau de l'arbre 1 – comparons le score de chacun des splits. On a 14511 valeurs dans cet exemple, donc 14510 splits sont possibles. Mais nous nous contenterons de 3 exemples de splits :

— **Split 1.1** : U_moy_filtr_out<-0.0970588252

— **Split 1.2** : U_moy_filtr_out<0.1

— **Split 1.3** : U_moy_filtr_out<-0.03

	Split 1.1	Split 1.2	Split 1.3
Feuilles initiales - Nombres de résidus	résidus de 14 511 exemples		
Feuille initiale – Score ($\lambda = 0$)	2.1950421332825138e-32		
Feuille gauche - Nombres de Résidus	633	2762	759
Feuille gauche – Score ($\lambda = 0$)	0.00042879908950134443	8.276731613824675e-06	0.0002673707811789769
Feuille droite - Nombres de Résidus	13 878	11 749	13 752
Feuille droite – Score ($\lambda = 0$)	1.955828099541368e-05	1.9457258249539322e-06	0.0002821275018679562
Total Score ($\lambda = 0$, $\gamma = 0$)	0.0004483573704967581	1.0222457438778606e-05	0.0002821275018679562

TABLEAU 4.2 – Résultats obtenus lors du Split du 1er niveau de l'arbre 1 utilisant l'algorithme XGBoost.

Ces calculs sont répétés sur les autres fonctionnalités, calculant l'estimateur, les résidus pour la feuille gauche et droite. Ainsi que le calcul **du score**(le Gain) qui permettra de choisir quelle fonctionnalité (et sa condition) on conservera.

On conclut très rapidement que **le split Split 1.1 (U_moy_filtr_out < -0.0970588252)** a obtenu le meilleur score comparé aux autres splits, il est donc conservé dans la construction de l'arbre au premier niveau.

On passe maintenant au Split du 2^e niveau de l'arbre, de la feuille droite, ou on devra calculer le Gain, après les calculs on aboutit au fait que **(U_moy_in $<$ 0.29868421)** soit la fonctionnalité ayant le plus grand score (gain). Cette étape est répétée jusqu'à l'atteinte du niveau maximum de profondeur initié au début de l'algorithme. On calcule maintenant les estimations optimales par feuille. On se sert des résidus restant par feuille :

$$w_j = \frac{\sum_{i \in j}(y_i - \hat{y}_i)}{\text{Nb Instances } + \lambda} \tag{4.3}$$

y_i : Les sorties réelles de l'ensemble de test.
$\hat{y}_i$: La moyenne des sorties réelles de l'ensemble de test.
λ : Hyper-paramètre utilisé pour la régularisation du modèle.
w_j : Les estimations optimales par feuille.

En conséquent on obtient les valeurs de chaque feuilles représentez ci-dessous dans la Figure 4.1 :

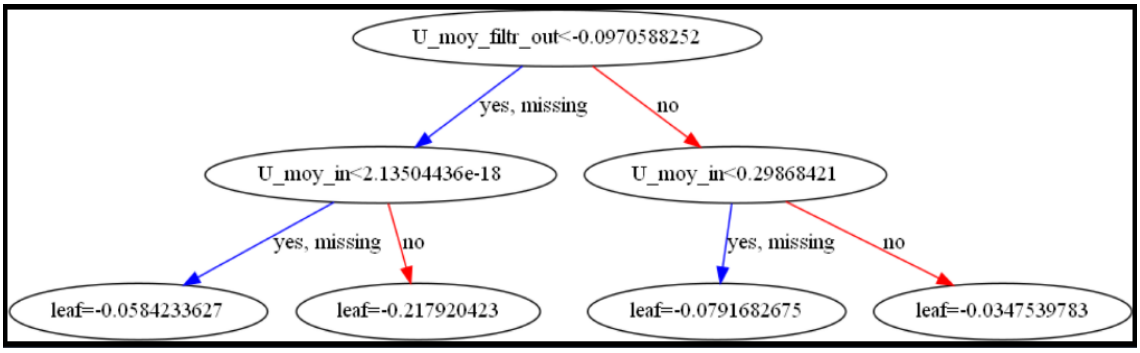

FIGURE 4.1 – Arbre illustrant les étapes et les résultats de l'algorithme XGBoost sur le jeu de données **Bitcoin-Alpha**

.

4. **étape 3** : On calcule les nouveaux résidus suite à la construction du nouvel arbre :
   L'affectation des résidus se fait selon les valeurs des fonctionnalités de chaque exemple d'entraînement. Chaque résidus d'ensemble d'entraînement aura additionné la feuille le correspondant à travers l'arbre.

   Après avoir calculé les nouveaux résidus pour chaque ensemble d'entraînement, nous réitérons les étapes 1-3 en créant un nouvel arbre, en calculant les valeurs nécessaires jusqu'à atteindre la limite d'itération donnée dans les paramètres de l'algorithme XGBoost, ayant comme objectif de minimiser le plus possible les résidus et permettre une prédiction précise.

4.2.1.2 Regression linéaire multiple

Le deuxième algorithme est la régression linéaire multiple. Son objectif consiste à calculer la distance entre chaque donnée et la ligne de régression et à chercher à la minimiser en utilisant la méthode des moindres carrés.

Au lieu d'utiliser la descente de gradient dans notre algorithme, nous utiliserons l'équation normale, qui est utilisée pour trouver la ligne la mieux adaptée aux données observées et cela en une seule itération.

Lorsque nous utilisons la régression linéaire multiple, l'équation de la régression linéaire simple :

$$Y = \beta_0 + \beta_1 X \tag{4.4}$$

β_0 : Désigne l'intercepteur.
β_1 : Désigne le coefficient.

sera reconvertie en :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p \tag{4.5}$$

β_0 : Désigne l'intercepteur.
β_i : Désigne l'i^e coefficients, $i = 1, 2, 3, ..., p$.
p : Désigne le nombre de features utilisés.

Le calcul des coefficients et de l'intercepteur se fait à travers l'équation normale :

$$[\beta_0, \beta_1, \beta_2, ..., \beta_p] = (X^T X)^{-1} X^T y \tag{4.6}$$

On obtient les résultats suivants :

$$\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \end{pmatrix} = \begin{pmatrix} -0.02464086 \\ -0.35730367 \\ 0.31727761 \\ 0.32226644 \\ -0.24530115 \\ 0.59768056 \\ 0.3874096 \\ 0.10181779 \\ 0.11488721 \end{pmatrix}$$

4.2.1.3 Support vector regression (SVR)

Support Vector Regression (SVR) est une fonction de régression généralisée du SVM. Un modèle d'apprentissage automatique utilisé pour la classification des données sur des données continues.

Cependant, comparé au SVM, le rôle de la marge maximale est différent. La fonction permettant de calculer les coordonnées de l'hyperplan dans SVR est de type linéaire sous la forme :

$$Y = wX + b \tag{4.7}$$

Quelques notations :

— w : Les coefficients de l'hyperplan.

— ε : L'écart entre l'hyperplan et les deux lignes de démarcations.

— ξ_i **et** ξ_i^* : L'écart entre les deux lignes de démarcations et les points hors du niveau de tolérances.

— ξi : L'écart entre les deux lignes de démarcations et les exemples se trouvant hors de l'hyperplan optimal.

— C : l'hyper-paramètre utilisé pour la régularisation du modèle de tel sorte que s'il est grand, il sera sujet à un sur-apprentissage qui, contrairement à une valeur minime proche de zéro, aura plus de chance de souffrir de sous-apprentissage.

— noyau : est une fonction utilisée dans l'apprentissage automatique pour trouver un classificateur non linéaire.

— γ : Le paramètre gamma est l'inverse de l'écart type du noyau. **RBF** (fonction gaussienne), qui sert de mesure de similarité entre deux points. Intuitivement, une petite valeur gamma définit une fonction gaussienne avec une grande variance. Dans ce cas, deux points peuvent être considérés comme similaires même s'ils sont éloignés l'un de l'autre. En revanche, une grande valeur gamma signifie une fonction gaussienne avec une petite variance et dans ce cas, deux points sont considérés comme similaires juste s'ils sont proches l'un de l'autre.

Notre objectif est essentiellement de trouver l'hyperplan qui contient le maximum d'observations d'entraînement dans la marge ε (niveau de tolérance) ce qui donnerai comme directive de maximiser la marge ou de minimiser la fonction de minimisation ci-dessous et sans enfreindre les contraintes ci-dessous :

1. Minimiser :

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{4.8}$$

2. Contraintes :

$$\begin{aligned} &y_i - wx_i - b \leq \varepsilon + \xi_i \\ &wx_i + b - y_i \leq \varepsilon + \xi_i^* \\ &\xi_i, \xi_i^* \geq 0 \end{aligned} \tag{4.9}$$

Dans notre cas, les données sont non linéaires, en conséquence, nous utiliserons le noyau **RBF**, qui est une fonction noyau utilisée dans l'apprentissage automatique pour trouver un classificateur non linéaire. Utilisant cette fonction :

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{4.10}$$

En appliquant le noyau sur SVR, on aboutit aux équations suivantes :

1. Minimiser :

$$\text{Min}_{w,b}\,\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{I}(\xi_i + \xi_i^*) \tag{4.11}$$

2. Contraintes :

$$\begin{aligned}
&\left(\sum_{k=1}^{K} w_k\phi\left(x_{i,k}\right) + b\right) - y_i \leq \epsilon + \xi_i^* \\
&y_i - \left(\sum_{k=1}^{K} w_k\phi\left(x_{i,k}\right) + b\right) \leq \epsilon + \xi_i \\
&\xi_i, \xi_i^* \geq 0 \qquad\qquad \forall_i
\end{aligned} \tag{4.12}$$

Vous trouverez ci-dessous la Figure 4.2 représentant un hyperplan optimal du SVR utilisant un noyau.

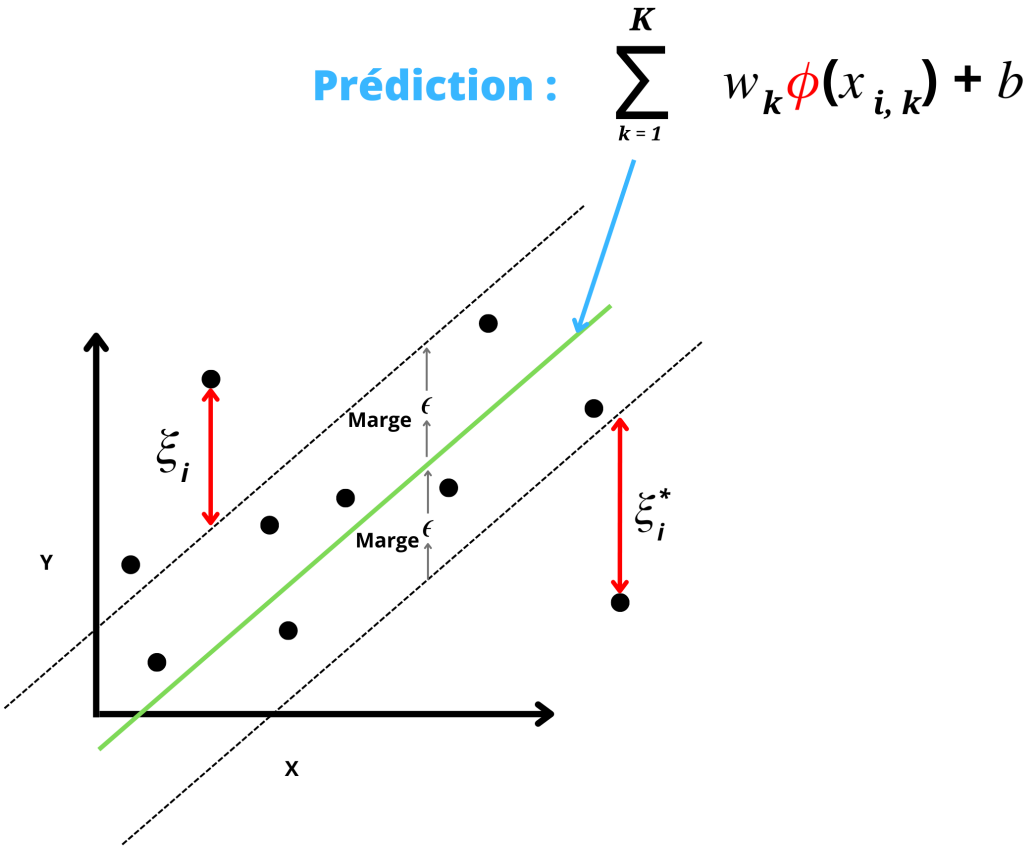

FIGURE 4.2 – Représentation d'un hyperplan optimal du SVR utilisant un noyau

.

4.2.1.4 K - Nearest Neighbors en régression (KNN regression)

KNN régression est très proche de KNN classification, l'algorithme KNN utilise la « similarité des caractéristiques » afin de faire une prédiction des valeurs de nouveau points de l'ensemble de test. Cela signifie qu'une valeur est attribuée au nouveau point en fonction de sa corrélation avec les points de l'ensemble d'apprentissage.

L'algorithme KNN régression est composé de 3 étapes :

- Calculer les distances entre le point à prédire et chaque point de l'ensemble d'entraînement en utilisant la métrique de Manhattan.
- Les K points de données d'entraînement les plus proches du point à prédire sont sélectionnés (en fonction de la distance).
- La moyenne de ces points de données sélectionnées sera la prédiction finale pour le point à prédire. l'équation utilisé pour calculer la moyenne est :

$$\hat{f}(x_{te}) = \frac{1}{k} \sum_{i \in N0} y_i \tag{4.13}$$

(x_{te} : Le point à prédire.
y_i : Le point sélectionné.
$N0$: Ensemble des i^e points sélectionnés.

4.2.2 Phase de validation

Précédemment, nous avons divisés les données de notre nouveau jeu de données en trois ensembles distincts. L'ensemble de validation est utilisé pour :

- Éviter le sur-apprentissage et le sous-apprentissage
- Optimiser et généraliser sur cet ensemble en utilisant le modèle généré précédemment dans la phase d'entraînement.

Afin d'aboutir à une bonne prédiction, un ajustement des hyper-paramètres est nécessaire, chaque hyper-paramètre se caractérise par sa capacité à influencer les résultats du modèle.

Pour évaluer les performances de notre modèle en termes de régression, l'utilisation d'une métrique est nécessaire pour calculer la distance entre la valeur prédite et les vrais valeurs.

— **L'erreur quadratique moyenne** :

L'erreur quadratique moyenne **RMSE (Root mean squared error)** est une métrique populaire pour mesurer le taux d'erreur d'un modèle de régression. Cependant, il ne peut être comparé qu'entre des modèles dont les erreurs sont mesurées dans les mêmes unités.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}} \tag{4.14}$$

a : Cible réelle.
p : Cible prévue.

Nous présenterons l'**erreur absolue moyenne**, et le **coefficient de corrélation de Pearson**, et qui sont deux métriques de régression dans la section suivante afin de discuter des résultats des modèles les comparer.

4.2.2.1 XGBoost

L'algorithme XGBoost est l'une des méthodes d'apprentissage automatique qui peut contenir plusieurs hyper-paramètres, ici, le paramètre **max_depth**, **n_estimators**, λ**(lambda)** , **colsample_bytree.**

Nous avons utilisé la **validation croisée k-fold** consistant à :

— Diviser aléatoirement l'ensemble d'entraînement en k sous-ensemble.
— Former un modèle sur tous les sous-ensemble sauf un (k-1).
— Évaluer le modèle sur ce sous-ensemble (k-1).

Ces étapes sont répétés k fois tout en modifiant à chaque fois le (k-1) sous-ensemble par un autre sous-ensemble de l'ensemble d'entraînement, finalement une moyenne de scores des résultats obtenus pour chaque itération sera calculée.

Nous avons aussi utilisé la **recherche de grille** (GridSearchCV), qui est une technique ayant pour objectif de rechercher les meilleures valeurs de paramètre à partir de l'ensemble donné de la grille de paramètre. Les meilleures valeurs de paramètres du modèles sont sauvegardées.

En conséquent, nous avons choisis 6 grilles de paramètre avec les valeurs ascendantes suivantes :

— **n_estimators : [400, 700, 1000]**
— **colsample_bytree : [0.7, 0.8]**
— **max_depth : [15, 20, 25]**
— **reg_lambda : [1.1, 1.2, 1.3]**

La moyenne RMSE des résultats obtenus sur les ensembles de validation est calculée après entraînement du modèle sur plusieurs combinaisons des paramètres cités au-dessus (voir la Figure 4.3).

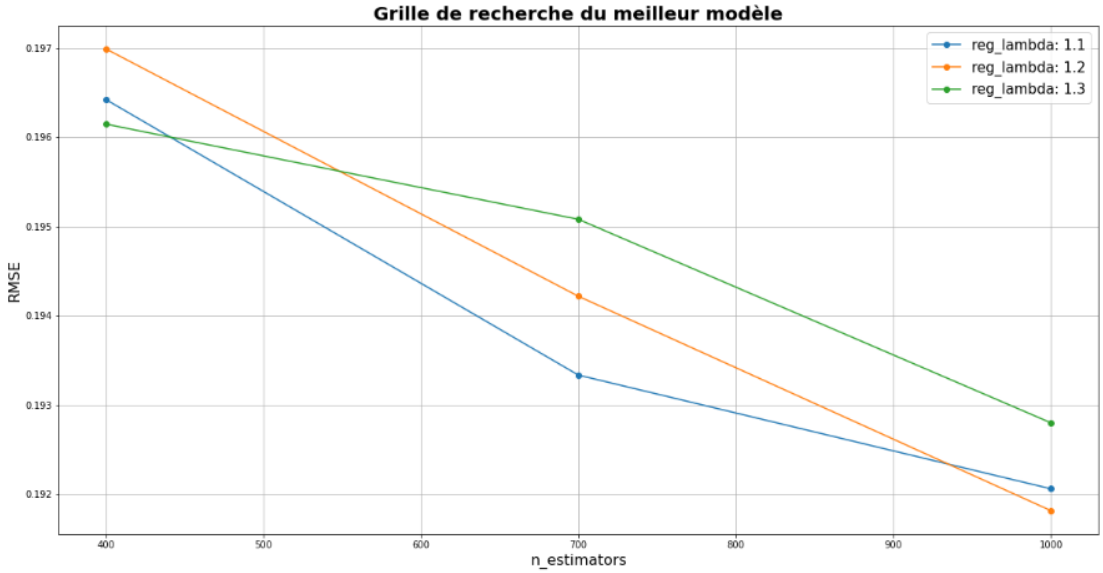

FIGURE 4.3 – Représentation des moyennes **RMSE** du modèle **XGBoost** obtenus sur les ensembles de validations sur λ et **n_estimators** sur le dataset **Bitcoin-Alpha**.

Avec la recherche de grille, nous avons abouti à une configuration optimale sur l'ensemble de validation. Partant des résultats obtenus dans la Figure 4.3, nous pouvons conclure que la configuration optimale pour le modèle XGBoost sur le jeu de données Bitcoin-Alpha est : **max_depth = 15, n_estimators = 1000, λ(lambda) = 1.2, colsample_bytree = 0.7**.

4.2.2.2 Régression linéaire multiple polynomial (RLMP)

La régression linéaire multiple reste limitée en termes d'utilisation de données non linéaires, de ce fait, le concept du polynomial a été utilisé afin d'adapter notre modèle durant la phase d'apprentissage et optimiser les résultats sur l'ensemble de validation. L'hyper-paramètre **degree** a été testé sur les différentes valeurs ascendantes suivantes :

— **degree : [1, 2, 3, 4, 5]**

Afin de déduire la configuration optimale, une visualisation des résultats par rapport à l'hyper-paramètre est nécessaire (voir la Figure 4.4).

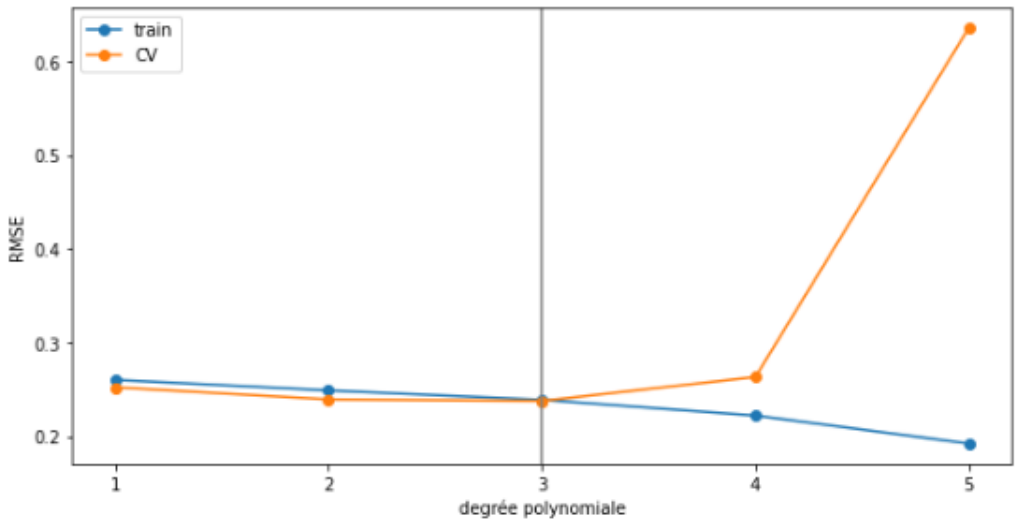

FIGURE 4.4 – Représentation des valeurs **RMSE** du modèle de **régression linéaire multiple polynomial** sur le paramètre **degree** du jeu de données **Bitcoin-Alpha**.

Nous pouvons voir que le modèle s'adapte jusqu'au degré 3. A partir du 4^e degré, le modèle entre dans l'état de sur-apprentissage avec une légère augmentation de la valeur RMSE, puis d'une plus grande augmentation à partir du degré 5.
Donc la meilleure valeur choisie est le paramètre **degree = 2** pour le modèle de régression linéaire multiple sur le jeu de données Bitcoin-Alpha.

4.2.2.3 Support vector regression (SVR)

Support vector regression est un algorithme utilisant deux hyper-paramètres principaux **gamma** et **C**. Chacun d'eux contribue au performances du modèle, durant la phase de validation. Nous avons utilisé la validation croisée k-fold et la recherche de grille (GridSearchCV) afin d'aboutir à la meilleure configuration possible pour notre modèle.

De ce fait, nous avons sélectionné deux grilles de paramètre avec les valeurs ascendantes suivantes :

— **C : [0.1, 1, 10, 100]**

— **gamma : [1, 0.1 ,0.01, 0.001]**

La Figure 4.5 illustre la moyenne RMSE des résultats obtenus sur les ensembles de validation après avoir entraîné le modèle sur plusieurs combinaisons des paramètres.

FIGURE 4.5 – Représentation des moyennes **RMSE** du modèle **SVR** obtenues sur les ensembles de validations sur chaque combinaison du jeu de données **Bitcoin-Alpha**.

La meilleure configuration des hyper-paramètres dépend des valeurs RMSE calculées, plus cette valeurs est faible. Plus la prédiction est bonne et permet de généraliser. D'après la Figure 4.5, nous pouvons considérer que la meilleur configuration pour notre modèle SVR sur le jeu de donnée Bitcoin-Alpha est **C = 10, gamma = 1**.

4.2.2.4 K - Nearest Neighbors en régression (KNN regression)

L'hyper-paramètre principal de l'algorithme KNN régression, est le nombres K plus proches, nous avons testé les valeurs ascendantes suivantes :

— **K : [1, 2, 3, ..., 30]**

Dans la Figure 4.6, nous pouvons constater qu'à K = 1, le modèle subit un sur-apprentissage, diminus au fur et à mesure que le K augmente, et finalement se stabilise à partir de K = 8. Nous avons pris la valeur **K = 18** comme optimum pour optimiser en terme de résultats et de temps de calcul.

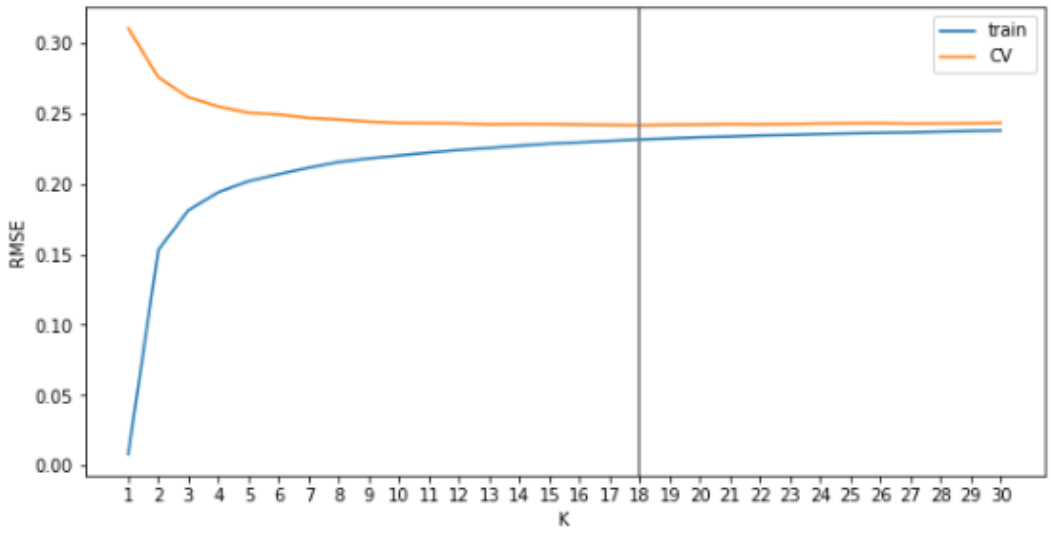

FIGURE 4.6 – Représentation des valeurs **RMSE** du modèle de **KNN regression** sur le paramètre K du jeu de données **Bitcoin-Alpha**.

4.2.3 Phase de test

Cette phase consiste à prédire les données de l'ensemble de test, en utilisant le modèle obtenus lors de la phase de validation. Nous comparons ensuite nos prédictions avec celles de l'ensemble de test(valeurs réelles). La phase de test est utilisée pour évaluer les performances du modèle dans la vie réelle.

4.3 Résultats

Dans le but de représenter les performances de nos modèles, nous utilisons la métrique d'erreur quadratique moyenne présenté durant la section précédente, ainsi que deux autres métriques :

— **L'erreur absolue moyenne** :

L'erreur absolue moyenne **MAE (Mean Absolut error)** a la même unité que les données d'origine et ne peut être comparée qu'entre des modèles dont les erreurs sont mesurées dans les mêmes unités. Son ampleur est généralement similaire à celle du RMSE, mais légèrement plus petite. a et p ayant les même significations que l'erreur quadratique moyenne (Eq. 4.14).

$$MAE = \frac{\sum_{i=1}^{n} |p_i - a_i|}{n} \tag{4.15}$$

— **Coefficient de corrélation de pearson :**

Le coefficient de **corrélation de Pearson PCC** (Pearson correlation coefficient) permet de calculer le taux de corrélation entre les valeurs prédites et les valeurs réelles. Une valeur PCC proche de 1, désigne une forte corrélation, tandis qu'un PCC proche de -1, signifie une faible corrélation.

$$PCC = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{4.16}$$

4.3.1 Comparaison des modèles IA construits

Afin de comparer les résultats entre des modèles, le Tableau 4.3 récapitule tous les résultats obtenus durant la phase de test de telle sorte que dans chaque cellule de tableau se trouve un tuple (MAE, RMSE, PCC) obtenus à partir d'un algorithme (ligne) sur le jeu de donnée (colonne).

	Bitcoin-Alpha	Bitcoin-OTC	Wikipedia-Rfa	advogato
XGBoost	**(0.08, 0.18, 0.76)**	**(0.08, 0.18, 0.85)**	**(0.11, 0.16, 0.78)**	**(0.05, 0.09, 0.92)**
SVR	(0.14, 0.23, 0.57)	(0.15, 0.26, 0.66)	(0.16, 0.21, 0.55)	(0.11, 0.15, 0.74)
RLMP	(0.14, 0.24, 0.50)	(0.16, 0.28, 0.60)	(0.16, 0.21, 0.55)	(0.11 ,0.15, 0.74)
KNN	(0.13 ,0.23, 0.56)	(0.14, 0.26, 0.66)	(0.16, 0.21, 0.53)	(0.10, 0.15, 0.75)

Tableau 4.3 – Résultats obtenus de chaque modèle sur nos jeux de données (Les valeurs en gras sont les meilleures).

Selon le Tableau 4.3 , le modèle ensembliste XGBoost a donné de meilleurs résultats que les modèles de la famille non linéaire, SVR, Régression linéaire multiple utilisant le degré polynomial et KNN.

Plus les erreurs MAE et RMSE déminuent, plus la valeur PCC augmente, et plus le modèle est performant. Nous avons remarqué que les trois algorithmes non linéaires sont similaires par rapport aux résultats des métriques d'évaluation.

Le modèle XGBoost a clairement surpassé les autres modèles dans les trois métriques, obtenant une valeur MAE réduite de ±0.06 de différences, avec la métrique RMSE une différence approximative de ±0.05, et enfin un gain de la valeur PCC de ±0.2.

Durant la phase de validation, un ajustement des hyper-paramètres a été établi en utilisant la méthode de validation croisée k-fold et la recherche de grille (GridSearchCV), afin de choisir les meilleurs hyper-paramètres possibles pour nos modèles ce qui a permis d'améliorer nos résultats.

La Figure 4.7 illustre une comparaison graphique des modèles sur le jeu de données **Wikipedia-Rfa**. L'axe des ordonnées représente les valeurs prédite, l'axe des abscisses représente les valeurs réelles (ensemble de test). Le modèle parfait, càd un score de 100%, sera interprété comme une ligne droite. De ce fait, plus les points de notre modèle s'approchent de la ligne droite, plus le modèle est optimal.

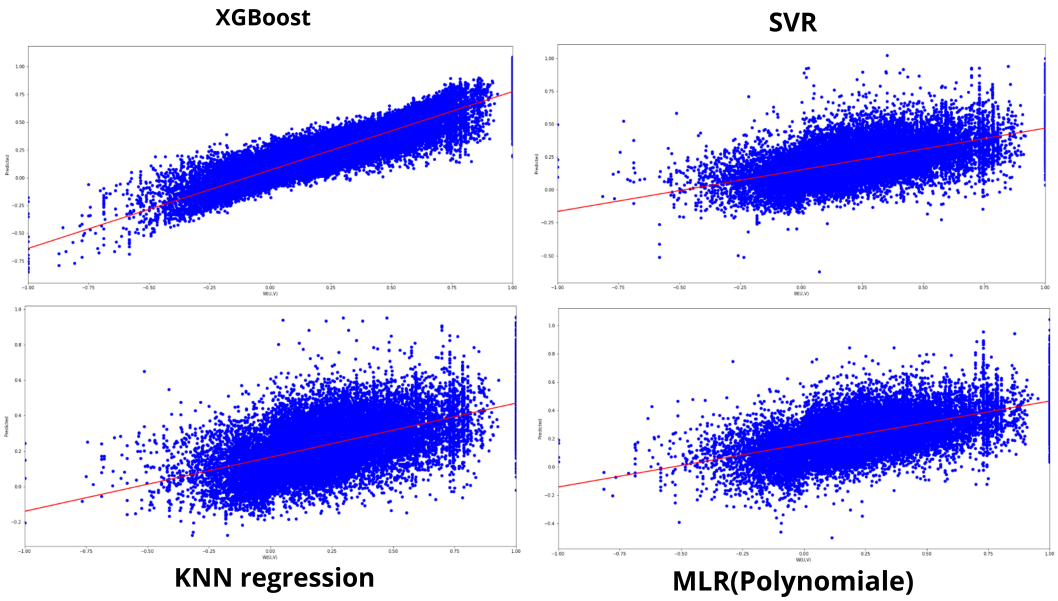

FIGURE 4.7 – Représentation des corrélation entre les valeurs prédite et les valeurs réelles des modèles sur le jeu de données **Wikipedia-Rfa**.

L'importance d'une fonctionnalité du jeu de données est mesurée à travers sa contribution à l'amélioration des performances du modèle, comme indiqué au chapitre 3. Le graphe de la Figure 4.8 montre l'importance des indicateurs pour la prédiction de la sortie, en utilisant le modèle XGBoost qui a surpassé les autres. Les trois fonctionnalités les plus intéressantes sont :

— La moyenne des arcs sortants du trustor **U** sauf l'arc **(U, V)**.
— La moyenne des arcs entrants du trustee **V** sauf l'arc **(U, V)**.
— Le nombre d'arcs sortants du trustor **U** sauf l'arc **(U, V)**.

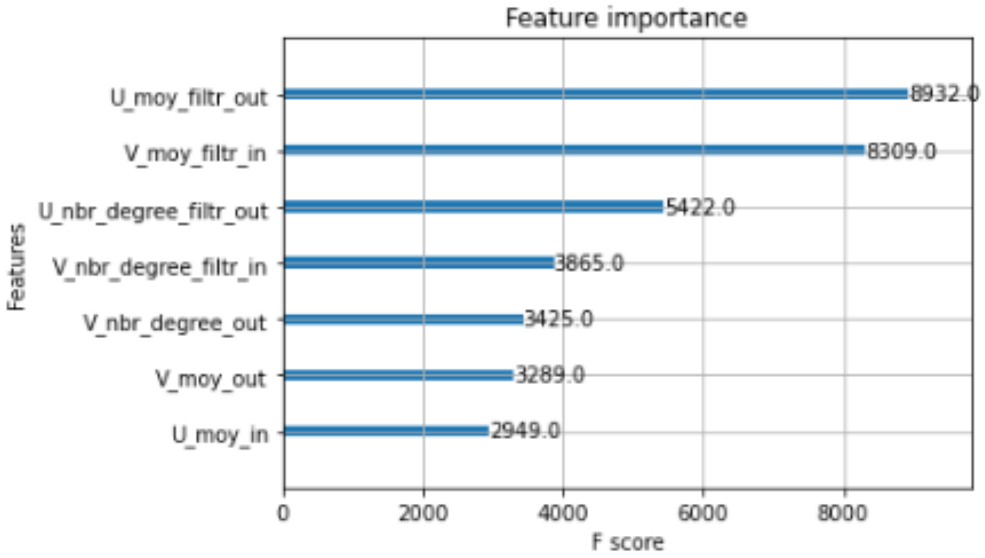

FIGURE 4.8 – Représentation de l'importance des fonctionnalités sur le jeu de données **Wikipedia-Rfa**.

4.3.2 Comparaison avec des algorithmes classiques

Précédemment, nous avons déduit que l'algorithme XGBoost est un algorithme puissant permettant de faire de bonnes prédictions. Ayant obtenu les meilleures performances lors de la comparaison avec trois autres modèles utilisés, cet algorithme a été choisi pour une comparaison avec d'autres travaux connexes afin de situer notre travail ainsi que notre contribution.

Une répartition standard des ensembles de données est utilisée dans les travaux connexes liée à la prédiction supervisée. De telle sorte que l'**ensemble d'entraînement** est de 60% et l'**ensemble de test** de 40%. Du fait que l'algorithme XGBoost est un algorithme d'apprentissage supervisé, une nouvelle répartition des données a été mise en place. En conséquence, l'**ensemble d'entraînement** est réparti en 50%, **ensemble de validation** 10%, **ensemble de test** 40%.

Les algorithmes classiques utilisent sans prétraitement les jeux de données initiaux.

Dans le but d'évaluer l'approche proposée, nous avons comparé les performances de notre modèle avec les algorithmes suivants :

— **Reciprocal (REC)** est un algorithme utilisant la réciprocité, arborant l'hypothèse que si un nœud U fait confiance a un nœud V, alors V fera probablement confiance au nœud U autant qu'il lui fait confiance. Autrement dit, $W(U,V) = W(V,U)$ si l'arc de V à U existe, et 0 sinon.

— **STAR** est une approche propagative proposée par Gao et al. (2016) [32]. L'algorithme STAR a pour but de prédire la confiance et la méfiance en définissant un semi-anneau 2D (la confiance et la certitude). L'interprétation de la confiance qu'accorde le nœud U à V se traduit par un couple $(W_{u,v}, C_{u,v})$. De telle sorte que $W_{u,v} \in [-1, 1]$ exprime le degré de confiance/méfiance , $C_{u,v} \in [0, 1]$ est la valeur de certitude de $W_{u,v}$. Les auteurs ont utilisé la longueur du chemin vers un nœud cible et le degré de ce nœud cible pour l'évaluation de la dimension de certitude. Ce semi-anneau favorise les arcs avec de plus grandes valeurs de certitude et contourne l'intransitivité de la méfiance en ignorant simplement les chemins avec deux liens négatifs successifs.

— **Agreement (AGR)** est un algorithme basé sur le filtrage collaboratif proposé par Akilal et al. (2019) [30]. Cet algorithme utilise uniquement les informations provenant des voisins directs des trustors et des trustees, et en utilisant l'accord entre nœuds comme métrique de similarité. Il peut déduire à la fois les relations de confiance et de méfiance.

— **Controversy, eclecticism, and reciprocity (CER)** est une approche proposé par Akilal et al. (2019) [36]. Cet approche considère que la confiance est le résultat d'une confrontation entre les traits des deux acteurs (trustor/trustee). Les traits (la contreverse, l'éclectisme, la réciprocité) les plus dominants l'emportent et affectent donc la valeur de la confiance.

— **Gullibility, competence, and reciprocity (GCR)** est une autre approche proposé par Akilal et al. (2020) [35]. Cet approche se base sur la confrontations de trois traits (la crédulité, la compétence, la réciprocité) des deux acteurs (trustor/trustee). Les traits les plus dominants l'emportent et affectent donc la valeur de la confiance.

Le Tableau 4.4 regroupe les résultats obtenus durant la phase de test. Chaque cellule de tableau se trouve un tuple (MAE, RMSE, PCC) obtenus à partir d'un algorithme (ligne) sur le jeu de données (colonne).

	Bitcoin-Alpha	Bitcoin-OTC	Wikipedia-Rfa
XGBoost	**(0.08, 0.18, 0.76)**	**(0.09, 0.19, 0.83)**	**(0.11, 0.16, 0.78)**
GCR	(0.12, 0.23, 0.61)	(0.14, 0.27, 0.67)	(0.17, 0.22, 0.53)
CER	(0.12, 0.24, 0.60)	(0.14, 0.27, 0.66)	(0.17, 0.22, 0.54)
AGR	(0.14, 0.24, 0.56)	(0.14, 0.26, 0.69)	(0.17, 0.22, 0.53)
STAR	(0.21, 0.32, 0.25)	(0.23, 0.35, 0.22)	(0.24, 0.32, 0.22)
REC	(0.12, 0.27, 0.47)	(0.15, 0.32, 0.46)	(0.56, 0.63, 0.071)

TABLEAU 4.4 – Résultats obtenus de notre modèle avec les algorithmes classiques sur nos jeux de données.

D'après le Tableau 4.4, l'algorithme **XGBoost** surpasse les algorithmes classiques dans les trois métriques utilisés pour l'évaluation. En termes de MAE, l'algorithme **REC** se rapproche plus au moins de XGBoost sur le jeu de données Bitcoin-Alpha et Bitcoin-OTC, étant réputé pour avoir un fort taux de réciprocité. Cependant, en termes de RMSE, l'algorithme **XGBoost** est plus précis que **REC**.

GCR et **CER** sont les deux approches qui se rapprochent le plus du modèle **XGBoost** avec une différence de ± 0.04 en MAE et RMSE, de ± 0.15 pour PCC. Cette différence s'explique du fait que notre modèle apprend du jeu de données directement sans distinctions sur l'interprétation des données, tandis que les deux algorithmes **GCR** et **CER** se base sur des concepts de la confiance. **GCR** et **CER** prouvent également que les concepts utilisés se rapprochent le plus de la réalité.

Il faut aussi noter que la comparaison faite, reste toute fois biaisée. Les tuples (MAE, RMSE, PCC) sont calculés sur **100% des données** par les algorithmes classiques énumérés précédemment, alors que les algorithmes d'apprentissage supervisé ne sont testés que sur **40% des données**.

Finalement, l'algorithme **XGBoost** obtient de meilleurs résultats par rapport à sa capacité à apprendre et à généraliser sur de nouveaux ensembles de données. Néanmoins, le temps de prédiction est loin d'être optimal par rapport aux algorithmes classiques. Le temps de prédiction dépend principalement de la taille du jeu de données, du nombre d'arbres et de la profondeur d'un arbre.

4.4 Conclusion

Dans ce chapitre, nous avons concrétisé notre travail en suivant trois étapes de l'apprentissage automatique sur nos données, à savoir la phase d'apprentissage, la phase de validation et la phase de test. Durant chaque phase, des modèles ont été appliqués dans l'objectif final de prédire une valeur de confiance entre deux utilisateurs dans un réseau social.

Une étude comparative a été mise en œuvre, par nos soins, en calculant des métriques de corrélations et d'erreurs. Avec les résultats obtenus et après la discussion de ces derniers, un fait incontestable a été établi sur quel modèle est le plus performant, le modèle XGBoost.

Conclusion et perspectives

Avec l'avènement des réseaux sociaux, prédire la confiance (et la méfiance) qu'un utilisateur devrait accorder à un autre devient un besoin pressant. En effet, prédire le degré de confiance attribué à un utilisateur quelconque permet de savoir si une mise en relation pourrait s'avérer dangereuse ou non, leur permettant ainsi d'avoir plus de sécurité et une meilleure présence et interaction en ligne.

Les réseaux sociaux ne sont qu'un outil, ils ne sont ni bons ni mauvais, ils sont ce qu'on en fait. Pour répondre à cette problématique de confiance dans les réseaux sociaux, le projet présenté dans ce mémoire propose une solution permettant d'établir le potentiel des modèles d'intelligence artificielle, et plus précisément les méthodes d'apprentissage automatique.

Nous avons vu quelques méthodes de prédiction de la confiance déjà réalisées. La plupart d'entre elles souffrent souvent de certaines limites, telles que l'incapacité à gérer une grande quantité de données, et une faible efficacité. Notre approche, consiste en la création de nouvelles fonctionnalités et leur utilisation sur quatre algorithmes supervisés, KNN, XGBoost, la régression linéaire multiple, et le SVR, pour la prédiction de la confiance, nous a donné des résultats meilleurs, plus pertinents.

Les algorithmes ont été évalués sur quatre ensemble de données, prennent en paramètre des traits sociaux qui sont des caractéristiques de la confiance. En l'occurence, la notoriété, la réputation, l'appréciation personnelle, et la sociabilité. Les quatre algorithmes donnent des résultats plus ou moins similaires. Les résultats de notre étude comparative montrent sans équivoque que l'algorithme XGBoost donne de meilleurs résultats en terme de précision bien que sa complexité temporelle laisse à désirer.

Ce projet était bénéfique pour nous dans plusieurs sens. Il nous a permis de consolider tout ce que nous avons acquis durant notre parcours, nous perfectionner en améliorant nos connaissances en programmation, et en conception et de la mise en œuvre des systèmes de prédiction (régression).

En guise de perspectives futures, s'inscrivant dans la continuité de nos recherches, nous souhaitons étudier d'autres aspects sociaux et pouvoir créer un modèle capable de comprendre ce qui affecte la confiance et pouvoir la prédire en temps réel.

Bibliographie

[1] Niklas Luhmann. *La confiance : un mécanisme de réduction de la complexité sociale*. Economica, 2006.

[2] Yann Algan, Pierre Cahuc, et al. La société de défiance : comment le modèle social français s' auto-détruit. Technical report, HAL, 2007.

[3] Mark Granovetter. Economic action and social structure : The problem of embeddedness. In *The sociology of economic life*, pages 22–45. Routledge, 2018.

[4] Eric Campoy and Valérie Neveu. Proposition d'une échelle de mesure de la confiance organisationnelle. *Revue de gestion des ressources humaines*, (62) :21–38, 2006.

[5] Denise M Rousseau, Sim B Sitkin, Ronald S Burt, and Colin Camerer. Not so different after all : A cross-discipline view of trust. *Academy of management review*, 23(3) :393–404, 1998.

[6] Roger C Mayer, James H Davis, and F David Schoorman. An integrative model of organizational trust. *Academy of management review*, 20(3) :709–734, 1995.

[7] Tushar Kanti Das and Bing-Sheng Teng. Trust, control, and risk in strategic alliances : An integrated framework. *Organization studies*, 22(2) :251–283, 2001.

[8] Stephen Paul Marsh. Formalising trust as a computational concept. 1994.

[9] Francis Fukuyama. *Trust : The social virtues and the creation of prosperity*. Simon and Schuster, 1996.

[10] Robert Demolombe. To trust information sources : a proposal for a modal logical framework. In *Trust and deception in virtual societies*, pages 111–124. Springer, 2001.

[11] Bart Nooteboom. Trust, opportunism and governance : A process and control model. *Organization studies*, 17(6) :985–1010, 1996.

[12] Barbara Benedict Bunker, Jeffrey Z Rubin, et al. *Conflict, cooperation, and justice : Essays inspired by the work of Morton Deutsch*. Jossey-Bass San Francisco, CA, 1995.

[13] Larry L Cummings and Philip Bromiley. The organizational trust inventory (oti). *Trust in organizations : Frontiers of theory and research*, 302(330) :39–52, 1996.

[14] Cristiano Castelfranchi and Rino Falcone. Trust and control : A dialectic link. *Applied Artificial Intelligence*, 14(8) :799–823, 2000.

[15] Cam Caldwell and Stephen E Clapham. Organizational trustworthiness : An international perspective. *Journal of business ethics*, 47(4) :349–364, 2003.

[16] Dmitry Khodyakov. Trust as a process : A three-dimensional approach. *Sociology*, 41(1) :115–132, 2007.

[17] Laurent Karsenty. Comment appréhender la confiance au travail. *La confiance au travail*, pages 13–51, 2013.

[18] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics : An experimental study on epinions. com community. In *AAAI*, volume 1, pages 121–126, 2005.

[19] Roy J Lewicki, Daniel J McAllister, and Robert J Bies. Trust and distrust : New relationships and realities. *Academy of management Review*, 23(3) :438–458, 1998.

[20] Le Parisien. Être méfiante vis-à-vis de l'autre. `http://pratique.leparisien.fr/forme-bien-etre/psycho/moi-autres/etre-mefiante-vis-a-vis-de-l-autre`. (page consultée le 30 mai 2022).

[21] Larousse. (s. d.). Méfiance. `https://www.larousse.fr/dictionnaires/francais/mefiance`. (page consultée le 30 mai 2022).

[22] Pierre Benjamin Lafaye. *Dictionnaire des synonymes de la langue française : avec une introduction sur la théorie des synonymes, ouvrage qui a obtenu de l'Institut le prix de linguistique en 1843 et en 1858*. L. Hachette et cie, 1861.

[23] Alphonse Karr. *Les guêpes*, volume 5. Calmann Lévy, 1888.

[24] Diane de Beausacq. *Les pensées et maximes de la vie*. 1883.

[25] Jennifer Ann Golbeck. *Computing and applying trust in web-based social networks*. University of Maryland, College Park, 2005.

[26] Jing Wang, Jian Yin, Yuzhang Liu, and Chuangguang Huang. Trust-based collaborative filtering. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 4, pages 2650–2654. IEEE, 2011.

[27] S Maheswari and GR Karpagam. Empirical evaluation of reputation based trust in semantic web. *International Journal of Engineering Science and Technology*, 2(10) :5672–5678, 2010.

[28] Hong Huang, Yuxiao Dong, Jie Tang, Hongxia Yang, Nitesh V Chawla, and Xiaoming Fu. Will triadic closure strengthen ties in social networks ? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(3) :1–25, 2018.

[29] Ghazaleh Beigi, Jiliang Tang, Suhang Wang, and Huan Liu. Exploiting emotional information for trust/distrust prediction. In *Proceedings of the 2016 SIAM international conference on data mining*, pages 81–89. SIAM, 2016.

[30] Karim Akilal, Hachem Slimani, and Mawloud Omar. A robust trust inference algorithm in weighted signed social networks based on collaborative filtering and agreement as a similarity metric. *Journal of Network and Computer Applications*, 126 :123–132, 2019.

[31] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412, 2004.

[32] Peixin Gao, Hui Miao, John S Baras, and Jennifer Golbeck. Star : Semiring trust inference for trust-aware social recommenders. In *Proceedings of the 10th ACM conference on Recommender systems*, pages 301–308, 2016.

[33] Yuan Yao, Hanghang Tong, Xifeng Yan, Feng Xu, and Jian Lu. Matri : a multi-aspect and transitive trust inference model. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1467–1476, 2013.

[34] Cai-Nicolas Ziegler. Trust propagation models. In *Social Web Artifacts for Boosting Recommenders*, pages 99–131. Springer, 2013.

[35] Karim Akilal, Mawloud Omar, and Hachem Slimani. Characterizing and using gullibility, competence, and reciprocity in a very fast and robust trust and distrust inference algorithm for weighted signed social networks. *Knowledge-Based Systems*, 192 :105345, 2020.

[36] Karim Akilal, Hachem Slimani, and Mawloud Omar. A very fast and robust trust inference algorithm in weighted signed social networks using controversy, eclecticism, and reciprocity. *Computers & Security*, 83 :68–78, 2019.

[37] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking : Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[38] Marjan Naderan, Ehsan Namjoo, and Somayeh Mohammadi. Trust classification in social networks using combined machine learning algorithms and fuzzy logic. *iranian journal of electrical and electronic engineering*, 15(3) :294–309, 2019.

[39] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7) :1019–1031, 2007.

[40] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06 : workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805, 2006.

[41] Saoussen Aouay, Salma Jamoussi, and Faiez Gargouri. Feature based link prediction. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pages 523–527. IEEE, 2014.

[42] Zhifeng Bao, Yong Zeng, and YC Tay. sonlp : social network link prediction by principal component regression. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 364–371. IEEE, 2013.

[43] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252, 2010.

[44] Larousse. (s. d.). Artificiel. `https://www.larousse.fr/dictionnaires/francais/artificiel`. (page consultée le 7 juin 2022).

[45] Larousse.(s. d.). Intelligence. `https://www.larousse.fr/dictionnaires/francais/intelligence` (page consultée le 7 juin 2022).

[46] Conseil de l'Europe. C'est quoi l'ia ? `https://www.coe.int/fr/web/artificial-intelligence/what-is-ai` (page consultée le 15 juin 2022).

[47] Jean-Louis Laurière. *Intelligence artificielle : résolution de problèmes par l'homme et la machine*. Eyrolles Paris, 1987.

[48] Pierre Marquis, Odile Papini, and Henri Prade. Panorama de l'intelligence artificielle. *Ses Bases Méthodologiques, ses Développements*, 1, 2014.

[49] Claude E Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12) :713–723, 1938.

[50] Alan M Turing. Intelligent machinery, a heretical theory. *The Turing test : Verbal behavior as the hallmark of intelligence*, 105, 1948.

[51] Alan M Turing. Can a machine think. *The world of mathematics*, 4 :2099–2123, 1956.

[52] Computing Machinery. Computing machinery and intelligence-am turing. *Mind*, 59(236) :433, 1950.

[53] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4) :12–12, 2006.

[54] John McCarthy. Defending ai research : a collection of essays and reviews. 1996.

[55] Innocent Mateyaunga. Predictive maintenance using machine learning. master thesis, Université de Tlemcen, Faculté de technologie, 2020.

[56] Thang Le Duc, Rafael García Leiva, Paolo Casari, and Per-Olov Östberg. Machine learning methods for reliable resource provisioning in edge-cloud computing : A survey. *ACM Computing Surveys (CSUR)*, 52(5) :1–39, 2019.

[57] Jean-Marc Azaïs and Jean-Marc Bardet. Le modèle linéaire par l'exemple régression, analyse de la variance et plans d'expériences illustrations numériques avec les logiciels r, sas et splus, 2006.

[58] Alboukadel Kassambara. *Machine learning essentials : Practical guide in R*. Sthda, 2018.

[59] Antoine Mazieres. Cartographie de l'apprentissage artificiel et de ses algorithmes. *Manuscrit de thèse, Université Paris Diderot*, 2016.

[60] Oded Z Maimon and Lior Rokach. *Data mining with decision trees : theory and applications*, volume 81. World scientific, 2014.

[61] Ankit Chauhan. Random forest classifier and its hyperparameters, 2021.

[62] Jason Brownlee. *XGBoost With python : Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery, 2016.

[63] Yuanchao Wang, Zhichen Pan, Jianhua Zheng, Lei Qian, and Mingtao Li. A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, 364(8) :1–13, 2019.

[64] Wend-Benedo Arnaud Bienvenue Zoungrana. *Application des algorithmes d'apprentissage automatique pour la détection de défauts de roulements sur les machines tournantes dans le cadre de l'Industrie 4.0*. PhD thesis, Université du Québec à Chicoutimi, 2020.

[65] Lipo Wang. *Support vector machines : theory and applications*, volume 177. Springer Science & Business Media, 2005.

[66] Philippe Lacomme, Gérard Fleury, Matthieu Gondran, and Chafik Samir. Découverte du machine learning-les outils de l'apprentissage automatique, 2021.

[67] Python.org. What is python ? executive summary. `https://www.python.org/doc/essays/blurb/` (page consultée le 2 aout 2022), 2022.

[68] Nvidia. What is networkx ? `https://www.nvidia.com/en-us/glossary/data-science/networkx/` (page consultée le 2 aout 2022), 2022.

[69] Nvidia. What is xgboost ? `https://www.nvidia.com/en-us/glossary/data-science/xgboost` (page consultée le 2 aout 2022), 2022.

[70] Eric Sigward. Introduction à la théorie des graphes. *Académie Metz-Nancy*, 2002.

[71] larousse. (s. d.). Notoriété. `https://www.larousse.fr/dictionnaires/francais/notoriete` (page consultée le 11 aout 2022), 2022.

[72] XGBoost.io. Xgboostparameters. `https://xgboost.readthedocs.io/en/stable/parameter.html` (page consultée le 25 aout 2022), 2022.

RÉSUMÉ

Les réseaux sociaux sont omniprésents dans notre quotidien. Ce pouvoir d'interagir avec des milliers de personnes de cultures et milieux différents est une lame à double tranchant. En effet, accorder sa confiance à un utilisateur dont on ne connait pas précisément sa vraie identité et la nature de ses attentions, est un gros risque à prendre. Les travaux d'aujourd'hui traitent de plus en plus la prédication de la confiance et encore plus la méfiance, longtemps ignorée, elle suscite plus d'intérêt que la confiance en soi.

Dans ce contexte, nous visons à travers ce travail à réaliser un modèle de prédiction de la confiance et méfiance dans les réseaux sociaux, qui s'appuie sur l'utilisation des méthodes d'apprentissage supervisé et comme métriques de confiance les traits sociaux tels que la notoriété et la sociabilité. De plus, nos expériences sur quatre ensembles de données montrent que, en plus de leur simplicité et de leur extensibilité, les quatre algorithmes d'apprentissage supervisé utilisés fournissent des résultats précis.

Mots clés : Réseaux sociaux ; Confiance ; Méfiance ; Prédiction de la confiance ; Méthodes d'apprentissage supervisé ; Métrique de confiance ; Trait social.

ABSTRACT

Social networks are ubiquitous in our daily lives. This power to interact with thousands of people from different cultures and places is a double-edged sword. Indeed, trusting a user whose true identity and the nature of his attentions is not precisely known, is a big risk to take. Today's work increasingly requires the preaching of trust and even more mistrust, long ignored, it arouses more interest than self-confidence.

In this context, we aim through this work to achieve a predictive model of trust and mistrust in social networks, which is based on the use of supervised learning methods and as trust metrics social traits such as notoriety and sociability. Moreover, our experiments on four datasets demonstrated that, in addition to their simplicity and scalability, the four supervised learning algorithms used provide accurate results.

Key words : Social networks ; Trust ; Mistrust ; Trust prediction ; Trust metric ; Social trait.