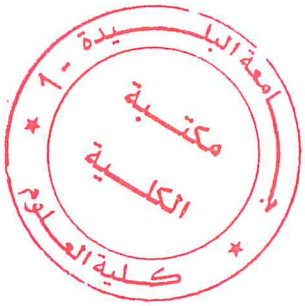
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saad Dahleb Blida1

Faculté des Sciences

Département d'Informatique



Mémoire présenté par :

AMEUR Abdelkader

BENACHOUR Amira

Pour l'obtention du diplôme de Master en Informatique

Option : Ingénierie des logiciels

Thème :

Développement d'un outil d'analyse prédictive

Encadreur : **Mme ABED Hafida.**

Soutenu le :

Année universitaire : 2016/2017

Remerciement

Toute louange est à ALLAH l'unique et le seul… qui nous a guidé vers le chemin du savoir
…donné le pouvoir et la patience à chaque fois qu'il le fallait…

Et c'est avec un grand plaisir que nous réservons ces lignes à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail.

Nous tenons à remercier sincèrement madame ABED Hafida, pour ses conseils avisés, ses remarques pertinentes, et aussi pour sa disponibilité et son encouragement.

Nous exprimons notre grande reconnaissance aux professeurs de département d'informatique de l'université de Blida 1 qui nous ont tant donné pour être ce que nous sommes aujourd'hui.

Nous adressons notre gratitude aux membres du jury qui nous feront l'honneur de juger notre travail.

Merci à tous et à toute

A mes chers parents

A notre promotrice Madame Abed pour son aide et son soutien

A mon binôme pour son travail, sa patience et sa persévérance

Pour monsieur Baouya et monsieur Bellatreche pour leur aide.
Pour toutes les personnes ayant contribué de loin ou de prés,
directement ou indirectement à la réalisation de ce travail.

Abdelkader

Dédicaces

Je dédie ce mémoire :
À ma mère,
À mon père,
À ma sœur Sarah,
À mes frères Abderrahmane et Youcef,
À toute ma famille,
À tous mes amis,
À mon binôme Abdelkader,
À tous ceux et toutes celles qui m'ont accompagné et soutenu durant cette année de formation.

Amira

ملخص

منذ ظهوره أحدث التحليل التنبئي ثورة في مجال الاعلام الالي وذلك بفضل متانته
وكونه طريقة سهلة لدعم اتخاذ القرار.
في السنوات الأخيرة، تم مجال جديد يقف أمام التحليل التنبئي: البيانات الكبيرة.
حتى كونه وسيلة فعالة لتحليل البيانات الضخمة، تطبيق خوارزميات التنبؤ على البيانات الكبيرة لا يزال شيء جديد ومهيأ لبحوث في المستقبل.
عملنا في هذا المجال يهدف إلى استخدام التحليلات التنبئية على البيانات الكبيرة من خلال دراسة الخوارزميات التنبئية، اختيار احد هذه الخوارزميات واختباره على البيانات الكبيرة
سنختتم هذه الدراسة مع برنامج دعم اتخاذ القرار ينطبق على البيانات الكبيرة والذي يمكن استخدامه في العديد من المجالات

كلمات مفتاحية: التحليلات التنبئية، البيانات الكبيرة، الخوارزمية.

Résumé

Depuis son apparition, l'analyse prédictive a révolutionnée l'industrie informatique grâce à sa robustesse et en étant un moyen facile d'aide à la décision.

Dans les dernières années, un nouveau domaine d'application s'est dressé devant l'analyse prédictive : les big data.

Même en étant un moyen d'analyse efficace pour les big data, l'application des algorithmes prédictives sur des big data est encore quelque chose de nouveau et est destiné à plusieurs futures recherches.

Notre travail s'inscrit dans ce contexte, on vise à utiliser l'analyse prédictive sur des big data en développant une version apte de traiter ces gros volumes de données avec plus d'efficacité.

L'étude se conclura par un logiciel d'aide à la décision applicable sur des big data et utilisable dans plusieurs domaines.

Mots clés : Analyse prédictive, big data, algorithme.

Abstract

Since its inception, predictive analysis has revolutionized the IT industry through its robustness and because it is an easy way to make decision.

In recent years, a new field of application has arisen in front of predictive analysis: big data.

Even it is efficient to analyze big data, the application of predictive algorithms on big data is still something new and may be the subject of future researches.

In this context is our study project, we aim to use predictive analysis on big data by developing a version able to analyze these large volumes of data with more efficiency.

At the end of this study, we will make decision support software that is applicable to big data and can be used in several domains.

Key words: Predictive analysis, big data, algorithm.

TABLE DES MATIERES :

INTRODUCTION GENERALE

CHAPITRE I : ETUDE BIBLIOGRAPHIQUE

CHAPITRE II : L'ALGORITHME CART

LISTE DES FIGURES :

LISTE DES TABLEAUX :

LISTE DES EQUATIONS :

INTRODUCTION GENERALE

INTRODUCTION GENERALE :

De nos jours, l'analyse prédictive est devenue un module essentiel dans tout système informatique existant. Comme preuve, une de ses applications les plus connues « l'évaluation du risque-client, » est utilisée dans l'ensemble des services financiers au niveau des entreprises. Les modèles d'évaluation traitent les antécédents de crédit d'un client, les demandes de prêt, les données client, etc, afin de classer les individus selon la probabilité de rembourser leurs crédits en temps voulu. Un exemple bien connu est le FICO Score[1] **[1]**.

Un autre cas montrant l'impact de l'analyse prédictive sur l'industrie d'aujourd'hui est la mise au point de la police prédictive, un système pouvant à l'avance prédire des endroits de crime en fonction du temps.

L'analyse prédictive regroupe un ensemble d'algorithmes et de méthodes basées sur les statistiques et la théorie des jeux et qui permettent de connaitre le comportement futur d'un système donné en se basant sur des données existantes. Ce concept qui était autrefois de la pure science-fiction trouve ses applications dans différents domaines.

Les premières applications de ce concept étaient dans le domaine du *marketing*, afin de comprendre les modèles de churn[2] **[2]**, aussi appelés les modèles d'attrition. Chez un opérateur de téléphonie mobile, la mise en place de l'analyse prédictive a par exemple permis de remettre à plat les réseaux de distribution, et de diminuer la proportion de clients abandonnistes.

Les banquiers ont aussi adopté la méthode pour multiplier par deux les taux de retour d'actions marketing, et pour diminuer de moitié le coût de ces campagnes. Pour proposer la bonne offre à la bonne personne au bon moment, mais aussi pour prendre un avantage concurrentiel grâce à la réponse instantanée que leur propose le système en matière, par exemple, d'octroi de crédit.

Mais bien sûr, le marketing et les services clients ne sont pas les seuls domaines d'application des algorithmes prédictifs.

Par exemple dans le domaine médical, une application prédictive peut être utilisée afin de déterminer la maladie d'une personne en se basant sur ses symptômes et sur les symptômes d'autres patients.

[1] : type de pointage de crédit, créé par la « Fair Isaac Corporation ».

[2] : Churn est un terme anglais qui désigne la perte de clientèle ou d'abonnés

Dans la finance, l'analyse prédictive peut aider à optimiser les sommes allouées au recouvrement pour encaisser les sommes dues en identifiant les agences les plus efficaces, les stratégies de contact, les actions judiciaires et autres pour chaque client, afin d'augmenter le taux de recouvrement tout en réduisant les coûts.

Dans la sécurité civile comme mentionné en dessus l'analyse prédictive peut être appliquée afin de déterminer en fonction du temps les endroits où il y a plus d'agression, cela permet de renforcer la sécurité dans les zones concernées et à partir de là sauver des vies.

Aussi dans le domaine de l'assurance sociale plusieurs types de fraudes ont une approche prévisible et peuvent être identifiés à l'aide de modèles statistiques, ce qui aide la prévention, les enquêtes après fraude et le recouvrement.

L'analyse prédictive peut être appliquée dans le domaine du génie civil, elle est utilisée pour prédire le degré des dommages que des constructions peuvent subir après une catastrophe naturelle donnée.

Bien sûr, les domaines cités ne sont qu'un exemple des milieux dans lesquels les modèles prédictifs peuvent être appliqués.

Plusieurs raisons ont permis à l'analyse prédictive de monter en importance de manière exponentielle ces dernières années. L'amélioration des outils d'analyse de données et les réussites démontrées dans le cadre des nouvelles applications ont donné aux modèles prédictifs leurs places d'aujourd'hui mais ce qui a vraiment contribué à l'avènement des algorithmes de prédiction est le phénomène des données massives ou Big Data.

L'analyse prédictive montre toute sa grandeur et son efficacité dans le milieu des Big Data, plus on possède de données mieux sont les résultats obtenus.

Malgré l'efficacité des algorithmes de l'analyse prédictive existants, certains ne sont pas adaptées à analyser les grands volumes de données, les études visant à appliquer ces algorithmes sur des Big Data ne sont pas nombreux. Ceci demeure toujours un sujet de recherche d'actualité.

Appliquer un algorithme prédictif sur des Big Data revient à dire : définir une version parallèle de l'algorithme.

Il existe plusieurs manières de paralléliser un algorithme, parmi lesquelles on trouve le Map-Reduce.

PROBLEMATIQUE :

Avec le développement du Big Data, l'utilisation des algorithmes d'analyse prédictive est devenue un enjeu crucial. Cependant, derrière la notion simplificatrice et vulgarisée d'algorithme prédictif se dissimule un processus analytique complexe et polyvalent. Le choix d'un algorithme par rapport à un autre est aussi une question récurrente au niveau de la littérature.

En fait, à la question « Quel algorithme dois-je utiliser ? » la réponse est toujours « Cela dépend. ». Cela dépend de la taille, de la qualité et de la nature des données. Cela dépend de ce que vous voulez faire avec la réponse. Cela dépend de la conversion des calculs de l'algorithme en instructions pour l'ordinateur que vous utilisez. Cela dépend du temps que vous avez, cela dépend de la plate-forme de déploiement, etc.

Nous avons donc tenté au cours de ce travail de comprendre les deux concepts clés de notre problématique : Analyse prédictive et Big Data. L'objectif de ce mémoire est donc d'étudier les principales techniques permettant de faire de la prédiction sur des données volumineuses. D'un autre côté, et vu que les plates-formes de déploiement de système de stockage de données sont nombreuses : Cloud, Map-Reduce, Machine parallèle, etc. Nous nous sommes focalisés sur la plate-forme Map-Reduce.

A partir de là, on peut résumer nos objectifs dans ce qui suit :

OBJECTIFS :

- Etat de l'art sur les algorithmes d'analyse prédictive
- Définition de critères de choix d'un algorithme
- Implémentation d'une version parallèle Map-Reduce de l'algorithme choisi
- Test sur des données réelles Big Data.

Ces objectifs nous ont permis de définir notre plan de travail :

- Introduction.
- Etude des différents algorithmes d'analyse prédictive.
- Choix de l'algorithme à appliquer.
- Conception et implémentation d'une version parallèle de l'algorithme choisi.

- Application de l'algorithme sur une étude de cas.
- Validation des résultats.
- Conclusion.

Ce plan va être exécuté via 4 chapitres :

- Le premier chapitre fait un état de l'art sur les méthodes les techniques existantes de l'analyse prédictive avec pour conclusion l'algorithme exact qui va être implémenté.
- Le deuxième chapitre fait la description détaillée de l'algorithme qui va être implémenté avec la définition, l'historique, le principe…
- Le troisième chapitre représente l'étude conceptuelle dans laquelle on va principalement définir la version Map-Reduce de notre algorithme.
- Le quatrième chapitre représente la partie implémentation et tests.

CHAPITRE I :

ETUDE BIBLIOGRAPHIQUE

1. INTRODUCTION :

Le but de ce chapitre est de présenter l'analyse prédictive : quelques généralités, des définitions, une explication de processus de conception d'un outil de prédiction, depuis la définition du projet jusqu'au développement du programme.

Le choix du modèle prédictif est la phase la plus importante, nous entamerons cette partie avec l'explication des modèles les plus répandus.

2. GENERALITES :

L'analyse prédictive moderne est née dans les années 1940, lorsque les gouvernements ont commencé à employer les premiers modèles informatiques (simulations de Monte-Carlo, modèles informatiques pour réseaux de neurones, programmation linéaire) pour décoder les messages allemands pendant la Deuxième guerre mondiale. Dans les années 1960, les entreprises et les établissements de recherche sont entrés dans l'ère de la commercialisation de l'analyse avec la programmation non linéaire, et de la résolution heuristique de problèmes via l'informatique : premiers modèles capables de prévoir les conditions météorologiques, résoudre le « problème du plus court chemin » pour améliorer les questions de transport aérien et de logistique et appliquer la modélisation prédictive aux demandes de crédit en fonction des risques. Puis dans les années 1970– 1990, l'analyse a été utilisée de façon plus répandue dans les entreprises et l'analyse prescriptive en temps réel est devenue réalité dans les startups technologiques.

Aujourd'hui, l'analyse prédictive a finalement investi beaucoup d'entreprises, qui y ont recours au quotidien dans des cas d'utilisation toujours plus nombreux **[3]**.

Il est important de situer l'analyse prédictive et savoir la différence avec les autres types d'analyses. Selon Gartner on a 4 types d'analyses :

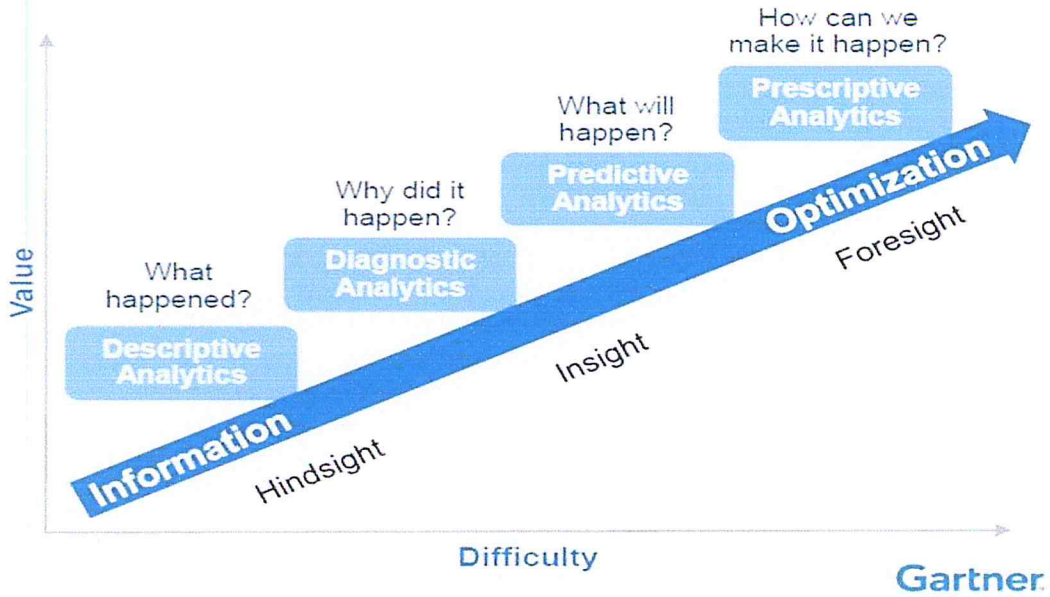


Figure 1 « Types d'analyse selon Gartner » sur le site « gartner.com »

Analyse Descriptive : Qu'est-il arrivé ? L'analyse descriptive traite le passé et essaye de comprendre les effets sur le présent (Business Intelligence).

Analyse diagnostic : Pourquoi cela s'est produit ? L'analyse diagnostic fournit des réponses aux questions sur les raisons, les effets, les interactions ou les séquences d'événements. Ici le terme Business Analytics cadrerait bien.

L'analyse prédictive : Qu'est-ce qui va se passer ? L'analyse prédictive regarde vers l'avenir et livre sur la base de l'exploration de données, l'apprentissage machine et d'autres méthodes statistiques la probabilité d'événements futurs.

Analyse prescriptive : Comment nous devons agir de telle sorte qu'un événement futur (ne) se produise (pas) ? L'analyse Prescriptive va un pas plus loin que l'analyse prédictive. Elle fournit des recommandations sur la façon d'affecter une tendance particulière dans une direction souhaitée, prévenir un événement prévu ou pour répondre à un événement futur.

3. DEFINITION DE L'ANALYSE PREDICTIVE :

Selon Siegel, l'analyse prédictive : Technologie qui apprend de l'expérience (données) pour prédire le comportement futur des individus afin de conduire de meilleures décisions **[4]**.

Gutierrez, définit l'analyse prédictive, qui parfois appelée analyse avancée, comme une série de techniques analytiques et statistiques permettant de prédire des actions ou des comportements futurs **[3]**.

Pour bien résumé : L'Analyse Prédictive est une technique d'exploration, d'extraction et d'analyse de données historiques et actuelles pour prédire les tendances et événements futurs ainsi que les motifs de comportements. L'exactitude des résultats dépendra énormément de l'analyse des données et de la qualité des hypothèses. Le cœur de l'analyse prédictive se fonde sur la capture des relations entre des points de données issues du passé et sur l'utilisation de ces relations pour prédire les résultats futurs. Afin de pouvoir faire des prédictions sur la base d'un ensemble de données spécifique, on utilise une ou plusieurs variables prédictives pour prédire une variable réponse. Sous sa forme la plus simple, l'analyse prédictive est un support permettant d'établir des prévisions pour la prise de décision.

4. LE PROCESSUS DE L'ANALYSE PREDICTIVE :

L'analyse prédictive combine les techniques de statistiques, du data mining et d'apprentissage automatique pour trouver un sens à partir de données. Pour concevoir un modèle prédictif, il existe un cycle à suivre :

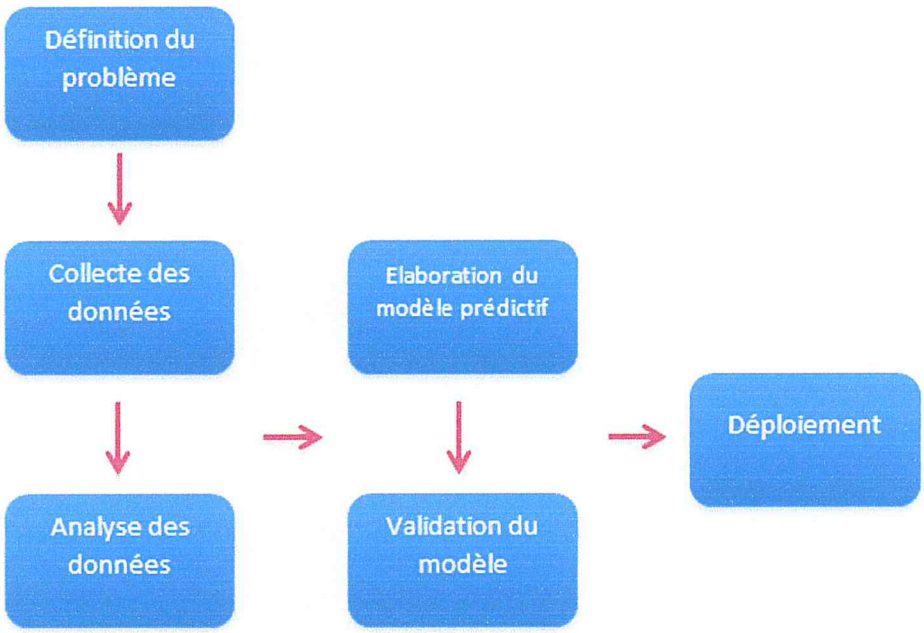


Figure 2 Processus de l'analyse prédictive

4.1. Définir le problème :

Cette phase est la plus importante, la définition bien détaillée d'un projet dès le départ permet une meilleure maitrise et donc une meilleure étude qui aboutit à la réussite du projet. Il faut :

- Bien comprendre la problématique.
- Définir le cadre d'étude : quel est le projet à réaliser ? Pour qui ?
- Identifier les ensembles des données qui vont être utilisées.
- Définir les résultats du projet et les résultats attendus ou objectifs …

Les réponses issues de cette étape entraîneront des exigences différentes pour les phases suivantes du cycle de vie.

4.2. La collecte des données :

Quel que soit le type d'évaluation menée, il est essentiel de bien choisir les méthodes de collecte et d'analyse des données et de les appliquer correctement.

La collecte de données doit en effet permettre d'obtenir l'ensemble des données nécessaires pour porter des jugements appropriés sur le programme ou la politique.

Dès que le but de l'évaluation est clairement défini, il est nécessaire de convenir d'un petit nombre de questions clés d'évaluation. Le fait d'y répondre devrait garantir le respect du but de l'évaluation. Disposer d'un ensemble reconnu de questions clés d'évaluation fournit une orientation sur les données à recueillir et la manière de les analyser et d'en rendre compte dans les résultats de l'évaluation.

Les méthodes de collecte et d'analyse de données doivent être choisies en fonction de l'évaluation concernée, de ses questions clés d'évaluation et des ressources disponibles.

Les évaluations d'impact doivent utiliser autant que possible les données existantes, puis s'appuyer sur de nouvelles données pour combler les lacunes.

Les méthodes de collecte et d'analyse de données doivent être sélectionnées de manière à assurer la complémentarité entre les points forts et les faiblesses des unes et des autres.

Après avoir passé en revue les informations disponibles, il convient de créer une matrice d'évaluation précisant les méthodes de collecte et d'analyse de données utilisées pour répondre à chaque question clé d'évaluation, puis d'identifier et de hiérarchiser les lacunes devant être comblées en recueillant de nouvelles données. Ceci permet de confirmer que la collecte de données prévue (et le regroupement de données existantes) recouvre l'ensemble des questions clés d'évaluation, de déterminer si la triangulation entre les différentes

sources de données est suffisante, et de contribuer à la conception d'outils pour la collecte de données afin de s'assurer que les informations nécessaires sont recueillies.

4.3. L'analyse des données :

Cette étape consiste en l'analyse et l'interprétation des données recueillies. Que les données soient quantitatives ou qualitatives, l'analyse peut se révéler plus ou moins complexe selon les méthodes utilisées et la quantité de données compilées.

Il est nécessaire de vérifier qu'aucune donnée n'est inexacte ou manquante. Ce processus s'appelle le nettoyage des données, et il consiste à détecter et à traiter toutes les erreurs qui se produisent lors de l'écriture, de la lecture, du stockage, de la transmission ou du traitement des données informatisées **[5]**.

Assurer la qualité des données signifie également garantir des analyses et une présentation des données appropriées avec par conséquent des résultats clairs. Il est par ailleurs nécessaire de rendre les données accessibles afin qu'elles puissent être utilisé par le modèle prédictive, soit dans la phase d'apprentissage ou pour l'évaluation.

4.4. Elaboration du modèle prédictif (Techniques de l'analyse prédictive) :

C'est la phase de création, test et validation du modèle prédictif. Il existe différentes techniques en fonction de la variable étudiée si celle-ci est quantitative ou qualitative, continue ou discrète.

D'une manière générale, on considère que les problèmes de prédiction d'une variable quantitative sont des problèmes de régression tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification. Certaines méthodes, comme la régression logistique, sont à la fois des méthodes de régression, au sens où il s'agit de prédire la probabilité d'appartenir à chacune des classes, et des méthodes de classification. Nous présentons en ce qui suit les modèles les plus connus :

4.4.1. Régression : [6]

La régression est un ensemble de méthodes statistiques très utilisées pour analyser la relation d'une variable par rapport à une ou plusieurs autres, On étudie en régression deux variables quantitatives, dont l'une, appelée variable expliquée, est considérée comme dépendante de l'autre, appelée variable explicative ou indépendante. On note habituellement la variable expliquée Y, et la variable explicative X. Dans cette relation, la

valeur de la variable dépendante (expliquée) est traitée comme étant fonction de la valeur de la ou des variables indépendantes (x).

Il existe différents types de techniques de régression disponibles pour faire des prédictions. Ces techniques sont principalement motivées par trois paramètres (nombre de variables indépendantes, type de variables dépendantes et la forme de la ligne de régression).

Le modèle de régression le plus connu est le modèle de régression linéaire :

Dans cette technique, la variable dépendante est continue, la variable indépendante peut être continue ou discrète , et la nature de la ligne de régression est linéaire (la fonction qui relie les variables explicatives à la variable expliquée est linéaire).

Un modèle de régression linéaire simple est défini par une équation de la forme :

Dans le modèle linéaire, la relation bi variée entre y et x est décrite par l'équation suivante :

$$y=\alpha + \beta x \quad (1)$$

Autrement dit, la relation entre x et y est modélisée par une ligne droite dont la pente est β

En généralisant aux cas avec plusieurs variables indépendantes, le modèle linéaire décrit la relation au moyen de l'équation suivante :

$$Y=\alpha + \beta 1x1 + \beta 2x2 \ldots \beta kxk \quad (2)$$

Où :

- Y est la variable dépendante
- x1, x2… xk sont les variables indépendantes
- α est un coefficient de valeur constante
- $\beta 1, \beta 2 \ldots \beta k$ sont les coefficients des variables indépendantes

La méthode dite des moindres carrés est la plus couramment utilisée pour estimer les paramètres de l'équation de régression linéaire.

Lorsque le modèle n'est pas linéaire, on peut effectuer une régression approchée par des algorithmes itératifs, on parle de régression non linéaire.

Si on s'intéresse au quantile conditionnel de la distribution de la variable aléatoire y sachant le vecteur de variables aléatoires x, on utilise un modèle de régression quantile (donne par approximation soit la médiane soit les autres quantiles de la variable réponse).

Si la variable expliquée est une variable aléatoire binomiale (le résultat est deux classes), il est courant d'utiliser une régression logistique ou un modèle probit.

Si la forme fonctionnelle de la régression est inconnue, on peut utiliser un modèle de régression non paramétrique.

4.4.2. L'apprentissage par classement [7] :

Contrairement à l'étape précédente consistant à calculer une valeur numérique probabiliste relative à un ensemble de données, le classement permet d'ordonner de nouvelles données ajoutées dans des classes prédéfinies ou non.

Si les classes sont déjà définies depuis le début on parle d'apprentissage supervisé.

Dans le cas contraire, ça veut dire dans le cas où les classes sont générées au même temps que l'apprentissage et ne sont donc pas définies depuis le début, on parle d'apprentissage non supervisé.

Dans les deux cas, il existe plusieurs structures de données et algorithmes qui permettent de réaliser un modèle prédictif efficace :

1. *Les arbres de décision* [8] :

C'est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches (les « feuilles » de l'arbre), et sont atteints en fonction de décisions prises à chaque étape.

Les arbres de décision représentent un moyen d'apprentissage supervisé.

Du point se vu apprentissage, on distingue deux types d'arbres :

- Arbre de classification : Arbre qui génère des classes d'appartenance.
- Arbre de régression : Arbre qui revoie une valeur probabiliste.

Critère de segmentation : Usuellement, les algorithmes pour construire les arbres de décision sont construits en divisant l'arbre du sommet vers les feuilles en choisissant à chaque étape une variable d'entrée qui réalise le meilleur partage de l'ensemble d'objets. Pour choisir la variable de séparation sur un nœud, les algorithmes testent les différentes variables d'entrée possibles et sélectionnent celle qui maximise un critère donné.

Dans le cas d'arbre de classification il s'agit d'un problème de classification automatique. Le critère d'évaluation des partitions caractérise l'homogénéité (ou le gain en homogénéité) des sous-ensembles obtenus par division de l'ensemble. Ces métriques sont appliquées à chaque sous-ensemble candidat et les résultats sont combinés (par exemple, moyennés) pour produire une mesure de la qualité de la séparation.

Il existe un grand nombre de critères de ce type, les plus utilisés sont l'entropie de Shannon, le coefficient de Gini et leurs variantes.

- Indice de diversité de Gini : utilisé par l'algorithme CART, le coefficient de Gini mesure avec quelle fréquence un élément aléatoire de l'ensemble serait mal classé si son étiquette était sélectionnée aléatoirement depuis la distribution des étiquettes dans le sous-ensemble. Le coefficient de Gini peut être calculé en sommant la probabilité pour chaque élément d'être choisi, multipliée par la probabilité qu'il soit mal classé. Il atteint sa valeur minimum (zéro) lorsque tous les éléments de l'ensemble sont dans une même classe de la variable-cible. Pratiquement, si l'on suppose que la classe prend une valeur dans l'ensemble 1,2,...m et si f_i désigne la fraction des éléments de l'ensemble avec l'étiquette i dans l'ensemble, on aura :

$$I_g(f)= \sum_{i=1}^{m} fi(1-fi)=1-\sum_{i=1}^{m} fi^2 \quad (3)$$

- Gain d'information : utilisé par les algorithmes ID3 et C4.5, le gain d'information est basé sur le concept *l'entropie de Shannon* en théorie de l'information.

Pour une source, qui est une variable aléatoire discrète X comportant n symboles, chaque symbole x_i ayant une probabilité P_i d'apparaître, l'entropie de la source X est définie comme :

$$\text{Entropie(X)}= -\sum_{i=1}^{n} Pi * \log(Pi) \quad (4)$$

A partir de là, on peut définir le gain d'information comme suit :

$$\text{Gain(X, Y)= Entropie(X)-} \sum_{i=1}^{m} \left(\frac{ai}{A}\right) * \text{Entropie(X, Y = ai)} \quad (5)$$

- Y représente la variable aléatoire pour laquelle on va calculer le gain.
- Les a_1, a_2,.....a_m représentent les modalités de Y.
- A représente le nombre total de lignes d'enregistrements pour lesquels on calcule les gains.
- Entropie (X, Y=a_i) représente l'entropie de X mais calculés pour le sous-ensemble d'enregistrements pour lesquels Y=a_k.

Pour le cas des arbres de régression, le même schéma de séparation peut être appliqué, mais au lieu de minimiser le taux d'erreur de classification, on cherche à maximiser la variance interclasses (avoir des sous-ensembles dont les valeurs de la variable-cible soient les plus dispersées possibles). En général, le critère utilise le *test du chi carré.*

Dans le cas de variables continues le critère de segmentation choisi doit être adéquat. En général, on trie les données selon la variable à traiter, puis on teste les différents points de coupure possibles en évaluant le critère pour chaque cas, le point de coupure optimal sera celui qui maximise le critère de segmentation.

Définir la taille de l'arbre **:** En apprentissage automatique sur base d'arbre de décision, un élément très important doit être considérer dans le processus : La taille de l'arbre.

Quant en parle de la taille, deux éléments surviennent à l'esprit : les valeurs de chaque attribut et le nombre total d'attributs utilisés pour la construction de l'arbre.

Bien sur ces deux éléments ne sont pas définis par l'algorithme qui va générer l'arbre mais par autre choses, par l'expert ou par d'autres techniques de statistiques selon le domaine d'étude, mais il est primordial de choisir le moins d'attributs possibles et qui donne la plus grande vision possible du système à modéliser.

Plus l'arbre devient très grand et complexe, plus l'on court le risque de voir ce modèle incapable d'être extrapolé à de nouvelles données, c'est-à-dire de rendre compte de la réalité que l'on cherche à appréhender, c'est ce qu'on appelle le problème de *sur-apprentissage.*

Afin de régler ce problème on fait de l'élagage qui consiste à supprimer les branches d'arbres moins importantes. Ce travail peut être exécuté durant ou après la construction de l'arbre. Ces opérations sont appelées respectivement : *pré-élagage* et *post-élagage* de l'arbre.

Donc il faut toujours essayer de construire l'arbre le plus simple possible avec la performance la plus haute possible.

Algorithmes utilisés **:** il existe plusieurs algorithmes qui permettent de construire un arbre de décision, en générale ces algorithmes possèdent tous une forme générale qui peut être résumé en ce qui suit :

Tableau 1 Algorithme générique de construction d'arbre de décision

Algorithme d'apprentissage générique Entrée : échantillon S Debut Initialiser l'arbre courant à l'arbre vide ; la racine est le nœud courant Répéter Décider si le nœud courant est terminal Si le noeud est terminal alors Lui affecter une classe Sinon Sélectionner un test et créer autant de nouveaux nœuds fils qu'il y a de réponses possibles au test FinSi Passer au nœud suivant non exploré s'il en existe Jusqu''à obtenir un arbre de décision Fin

En général, on décide qu'un nœud est terminal lorsque tous les exemples associés à ce nœud, ou du moins la plupart d'entre eux sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante.

En général, on attribue au nœud la classe majoritaire (éventuellement calculée à l'aide d'une fonction de cout lorsque les erreurs de prédiction ne sont pas équivalentes). Lorsque plusieurs classes sont en concurrence, on peut choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard.

La sélection d'un test à associer à un nœud est plus délicate. Puisqu'on cherche à construire un arbre de décision le plus petit possible rendant compte au mieux des données, une idée naturelle consiste à chercher un test qui fait le plus de progression possible dans la tache de classification des données d'apprentissage. Comment mesurer cette progression ? Il y a des algorithmes qui utilisent l'indice de Gini et d'autres utilisent la notion d'entropie.

En ce qui va suivre, on va résumer les algorithmes les plus connus dans la littérature permettant la génération d'arbre de décision :

- Algorithme ID3 (Iterative Dichotomiser 3) :

Développé à l'origine par Ross Quinlan .Il a tout d'abord été publié dans le livre ''Machine Learning'' en 1986.

ID3 permet de construire un arbre de décision en calculant le gain d'information de chaque attribut.

Il commence par calculer l'entropie d'information ensuite le gain global de chaque attribut, et celui dont la valeur est max est choisi comme racine, ensuite pour chaque modalité de la racine on calcule les gains des attributs restants et à chaque fois c'est celui présentant le gain max qui est choisi comme prochain fils.

L'algorithme peut être résumé en ce qui suit **[9]** :

Tableau 2 Algorithme ID3

ID3
Entrées : ensemble d'attributs A ; échantillon E ; classe c
Début
Initialiser l'arbre vide ;
Si tous les exemples de E ont la même classe c
Alors étiqueter la racine par c ;
Sinon si l'ensemble des attributs A est vide
Alors étiqueter la racine par la classe majoritaire dans E ;
Sinon soit *a* le meilleur attribut choisi dans A en calculant le gain ;
Étiqueter la racine par *a* ;
Pour toute valeur *v* de *a*
Construire une branche étiquetée par *v* ;
Soit Eav l'ensemble des exemples tels que e(*a*) = *v* ;
Ajouter l'arbre construit par ID3 (A-{*a*}, Eav, c) ;
finpour
finsinon
finsinon
Retourner racine ;
Fin

- Algorithme C4.5 :

Il a été développé aussi par Quinlan **[10]**, il est similaire à ID3 auquel il apporte plusieurs améliorations :

→ une adaptation de la fonction de gain qui n'a plus tendance à aller vers l'attribut avec le plus de valeurs possibles ;

→ la possibilité de gérer des attributs avec des valeurs manquantes ;

→ la possibilité de post-élaguer son arbre pour éviter le « sur apprentissage » ;

→ la possibilité de manipuler des valeurs continues (en les "discrétisant" lors de la mise en arbre).

C4.5 Utilise une nouvelle fonction de gain différente de celle utilisée par ID3 :

Ratio Gain (S, A) =Gain (S, A)/Split Entropie (S, A) (6)

Tel que :

Split Entropie (S, A) = $\sum_{v \in valeurs(A)} (|Sv|/|S|) \times \log2 (|Sv|/|S|)$

- S : Ensemble des exemples sur lequel le calcul est fait.
- A : Attribut qu'on veut calculer son gain d'information dans l'ensemble S.
- S_v : Sous ensemble de S pour lequel A prend la valeur v.

Quinlan continua avec les versions C5.0 et See5 (C5.0 pour les systèmes UNIX et See5 pour Windows) qu'il commercialisa. C5.0 améliore C4.5 sur plusieurs points dont :

→ La rapidité

→ L'utilisation de la mémoire

→ Des arbres de décision plus petits

C5.0 est un produit commercial dont le code source est disponible gratuitement pour l'interprétation et l'utilisation des arbres de décision et l'ensemble des règles qu'il produit.

- Algorithme CHAID: (CHI-Squared Automatic Interaction Detector)

Publié en 1980 par Gordon V. Kass **[11].** Il peut être utilisé pour la prédiction comme la régression linéaire ou pour la détection d'interaction entre variables.

En pratique, il est souvent utilisé en marketing direct pour sélectionner un groupe de consommateurs et prédire comment leurs réponses à certaines variables affectent d'autres variables.

Comme avec les autres arbres de décision, ses avantages sont un résultat essentiellement visuel et facilement interprétable. À cause de la segmentation de la population lors de l'analyse, l'échantillonnage doit être suffisamment large de manière que la taille de chaque groupe ne devienne pas trop petite, ce qui rendrait l'analyse peu fiable.

CHAID détecte l'interaction entre variables dans un jeu de données. En utilisant cette technique on peut établir des relations de dépendance entre variables. En prenant l'abonnement à un journal, par exemple, il sera possible d'étudier l'influence de variables explicatives comme le prix, la taille, les suppléments, etc. CHAID identifie des groupes discrets puis, en examinant les réactions aux variables explicatives, cherche à prédire l'impact sur la variable initiale.

CHAID est souvent utilisé comme technique d'exploration et est une alternative aux multiples régressions, en particulier quand le jeu de données n'est pas parfaitement adapté aux analyses par régression.

L'algorithme CHAID se déroule en trois étapes :

- Préparation des prédicteurs
- Fusion des classes
- Sélection de la variable de séparation

Dans cette dernière étape, l'algorithme utilise les probabilités pour estimer si une catégorie peut être divisée. L'algorithme « *Exhaustive* » CHAID (ECHAID) utilise une technique plus complexe à cette même fin.

- Algorithme CART : (Classification And Regression Trees)

Il est semblable à ID3 à l'exception du fait que l'arbre généré ici est binaire et le critère de segmentation utilisé est l'indice de Gini. **[12].**

- QUEST : (acronyme anglais pour quick, unbiased, efficient, classification tree)

Permet d'expliquer une variable qualitative par plusieurs variables qualitatives ayant un grand nombre de modalités **[13].** Il a été inventé par Wei-Yin Loh et Yu-Shan Shih en 1997 **[14].**

Il est utilisé dans le package lohTools **[15]** du logiciel R.

Il existe bien sur d'autres méthodes qui ne vont pas être détaillés ici comme : SLIQ, VFDT, UFFT…

2. ***Les réseaux de neurone*** : [16]

Définition *:* Un neurone formel est un modèle caractérisé par des signaux d'entrée (les variables explicatives par exemple), une fonction d'activation *f*, $f\ (\alpha_0+\Sigma i\alpha_i.x_i)$. *f* peut être linéaire, à seuil, stochastique et le plus souvent sigmoïde. Le calcul des paramètres se fait par apprentissage.

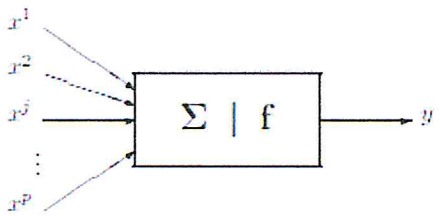


Figure 3 Neurone formel

Les neurones sont ensuite associés en couche. Une couche d'entrée lit les signaux entrant, un neurone par entrée xi, une couche en sortie fournit la réponse du système. Une ou plusieurs couches cachées participent au transfert. Un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.

De façon usuelle et en régression (Y quantitative), la dernière couche est constituée d'un seul neurone muni de la fonction d'activation identité tandis que les autres neurones (couche cachée) sont munis de la fonction sigmoïde.

De là, on distingue deux types de réseaux de neurones :

- Perceptron : C'est un réseau de neurone qui ne possède pas des couches intermédiaires. Il possède donc des neurones d'entrées directement reliés au neurone de sortie.

Dans ce cas de figure, on calcule la somme $S=x_1*w_1 + x_2*w_2 + \ldots. + x_n*w_n$ (7) tel que les x_i représentent les entrées et w_i les poids des arrêtes reliant les entrées au neurone de sortie. Ensuite on applique dans le neurone de sortie une fonction d'activation (Step function) f sur S et on fournit la sortie f(S).

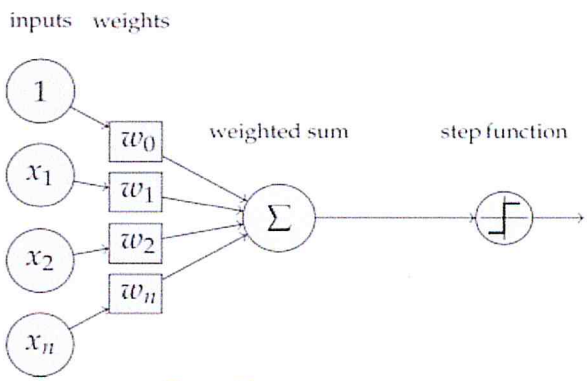


Figure 4 Perceptron

Dans un perceptron, la fonction d'activation est en général une fonction à seuil :

F(S)=1 si S > θ et 0 sinon. (8)

Le seuil θ est défini de plusieurs méthodes et on peut le remplacer par une entrée x_0=1 comme montrer dans la figure 2 avec un poids w_0=- θ.

Dans ce cas la définition précédente devient :

F(S)= 1 si S>0 et 0 sinon avec

$S= x_0*w_0 + x_1*w_1 + x_2*w_2 + \ldots. + x_n*w_n$

- Réseau multicouches : C'est un réseau neuronal possédant des couches intermédiaires de neurones reliant la couche d'entrée à la couche de sortie.

Chaque neurone d'une couche interne doit fournir à la prochaine couche une sortie calculée comme celle des perceptrons, la somme des x_i*p_i tel que les x_i sont les entrées que le neurone reçoit et les p_i sont les poids des arrêtes reliant les entées x_i au neurone considéré.

Bien sûr, les x_i peuvent être des sorties de neurones d'une couche précédente.

Ce qui est important de noter dans ce cas, c'est que chaque neurone interne doit disposer d'une fonction d'activation f qui est en générale une fonction sigmoïde.

Contrairement au perceptron, un réseau neuronal multicouche peut posséder plusieurs sorties; il est donc considéré comme la concaténation de plusieurs perceptrons.

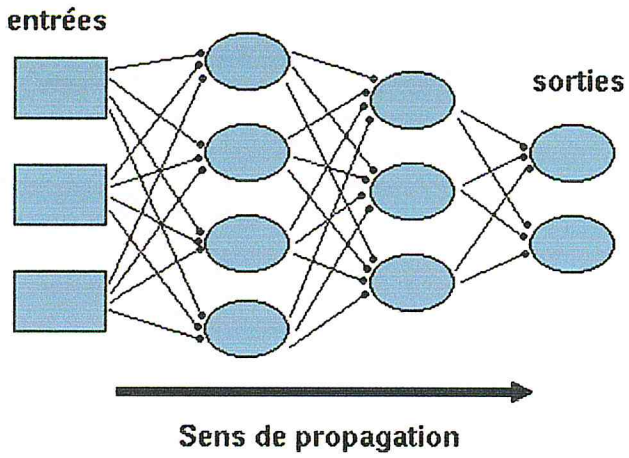


Figure 5 Réseau multicouche

Apprentissage : Dans les réseaux de neurones l'apprentissage est supervisé si le réseau doit converger vers une sortie bien définie. Dans le cas contraire, quand le réseau est libre de converger vers une sortie quelconque, là l'apprentissage est non supervisé.

Dans tous les cas, l'apprentissage dans un réseau neuronal consiste à calculer et adapter les poids synaptiques de manière est ce que le réseau devient capable de fournir les bonnes sorties en fonction bien sur des entrées adéquates. Ce problème au cœur des réflexions sur l'apprentissage a connu un début de réponse grâce aux travaux du physiologiste canadien Donald Hebb sur l'apprentissage en 1949 décrits dans son ouvrage *The Organization of Behaviour*. Hebb a proposé une règle simple qui permet de modifier la valeur des coefficients synaptiques en fonction de l'activité des unités qu'ils relient. Cette règle aujourd'hui connue sous le nom de « *règle de Hebb* » est presque partout présente dans les modèles actuels, même les plus sophistiqués. Cette règle suggère que lorsque deux neurones sont excités conjointement, il se crée ou renforce un lien les unissant.

Dans le cas d'un neurone artificiel seul utilisant la fonction signe comme fonction d'activation cela signifie que :

$$W_i^{'} = W_i + \alpha\,(Y * X_i)$$

Où :

- $W_i^{'}$ est le poids modifié,
- W_i le poids d'origine,
- Y la sortie calculée,
- α le pas d'apprentissage.

Les règles d'apprentissage présentent dans les algorithmes d'apprentissage à base de réseaux de neurones sont fondées à partir de cette règle à l'exception du fait qu'elles prennent en considération l'erreur observée en sortie.

D'autres problèmes se posent aussi quand on parle de l'apprentissage à base des réseaux de neurones, le sur apprentissage et la propagation d'erreurs figurent parmi les plus importants.

→ Sur-apprentissage : Il arrive souvent que les exemples de la base d'apprentissage comportent des valeurs approximatives ou bruitées. Si on oblige le réseau à répondre de façon quasi parfaite relativement à ces exemples, on peut obtenir un réseau qui est biaisé par des valeurs erronées.

Pour éviter le sur apprentissage, il existe une méthode simple : il suffit de partager la base d'exemples en 2 sous-ensembles. Le premier sert à l'apprentissage et le second sert à l'évaluation de l'apprentissage. Tant que l'erreur obtenue sur le deuxième ensemble diminue, on peut continuer l'apprentissage, sinon on arrête.

On peut utiliser aussi l'élagage, qui consiste à supprimer des connexions (ou synapses), des entrées ou des neurones du réseau une fois l'apprentissage terminé. En pratique, les éléments qui ont la plus petite influence sur l'erreur de sortie du réseau sont supprimés.

→ Propagation d'erreurs : Ce phénomène se déroule quand l'erreur commise par un neurone se propage à ses synapses et aux neurones qui y sont reliés. Pour les réseaux de neurones, on utilise habituellement la *rétropropagation du gradient de l'erreur*, qui consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs : les poids synaptiques qui contribuent à engendrer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale.

Nous allons maintenant donner un algorithme qui permet de faire l'apprentissage par les réseaux de neurones basé sur la loi de Hebb et la rétropropagation du gradient.

Cet algorithme va être adapté pour les deux types de réseaux **[17].**

Dans le cas d'un perceptron :

Apprentissage par descente de gradient : algorithme
- Initialiser aléatoirement les coefficients wi.
- Répéter :
- Pout tout i :
- $\Delta wi = 0$
- Fin Pour
- Pour tout exemple (x, c) dans S
- Calculer la sortie o du réseau pour l'entrée x
- Pout tout i :
- $\Delta wi = \Delta wi + \varepsilon * (c - o) * xi * \sigma'(x.w)$
- Fin Pour
- Fin Pour
- Pour tout i :
- $wi = wi + \Delta wi$
- Fin Pour
- Fin Répéter

O est la sortie calculée, σ'(x.w) est la dérivée de la *fonction d'activation* σ(x.w) et ε est une valeur bien choisie.

Il existe d'autres variantes pour cet algorithme :

Variante de la Règle Delta généralisée :
• On ne calcule pas les variations de coefficients en sommant sur tous les exemples de S mais on modifie les poids à chaque présentation d'exemple.
Initialiser aléatoirement les coefficients wi.
• Répéter :
•Prendre un exemple (x, c) dans S
•Calculer la sortie o du réseau pour l'entrée x
•Pout i de 1 à n :
• $wi = wi + \varepsilon * (c - o) * xi * \sigma'(x.w)$
•Fin Pour
•Fin Répéter

Aussi :

Apprentissage : algorithme de Widrow-Hoff (adaline / règle delta)
La fonction de transfert est linéaire (purelin) : $\sigma(y) = y$
Donc : $\sigma'(y) = 1$
• Initialiser aléatoirement les coefficients wi.
• Répéter :
•Prendre un exemple (x, c) dans S
•Calculer la sortie o du réseau pour l'entrée x
•Pout i de 1 à n :
• $wi = wi + \varepsilon * (c - o) * xi$
•Fin Pour Fin Répeter

Dans le cas d'un réseau multi couches :

Algorithme de rétropropagation du gradient :
• Initialiser aléatoirement les coefficients wij dans [-0.5 ; 0.5]
• Répéter
• Prendre un exemple (x, c) de S
• Calculer la sortie o
• Pour toute cellule de sortie i
• $\delta i = \sigma'(yi) * (ci - oi) = oi * (1 - oi) * (ci - oi)$
• Fin Pour
• Pour chaque couche de $q - 1$ à 1
• Pour chaque cellule i de la couche courante
• $\delta i = \sigma'(yi) * [\Sigma(k \in Succ(i)) (\delta k * wki)]$
$= oi * (1 - oi) * [\Sigma(k \in Succ(i)) (\delta k * wki)]$
• Fin Pour
•Fin Pour
• Pour tout poids wij $wij = wij + \varepsilon * \delta i * xij$ Fin Pour Fin Répéter

3. ***Machine à vecteurs de support :*** [18]

Les Support Vector Machines souvent traduites par l'appellation de Séparateur à Vaste Marge (SVM) sont une classe d'algorithmes d'apprentissage initialement définis pour la prévision d'une variable qualitative binaire. Ils ont été ensuite généralisés à la prévision d'une variable quantitative. Dans le cas de la discrimination d'une variable dichotomique, ils sont basés sur la recherche de l'hyperplan de marge optimale qui, lorsque c'est possible, classe ou sépare correctement les données tout en étant le plus éloigné possible de toutes les observations. Le principe est donc de trouver un classifieur, ou une fonction de discrimination, dont la capacité de généralisation (qualité de prévision) est la plus grande possible.

En gros, elles permettent de séparer des instances appartenant à deux classes différentes en les séparant dans un plan par un hyperplan. Par la suite, on a généralisé le problème de discrimination en plusieurs classes.

Les SVM regroupent plusieurs notions compliquées en mathématiques, nous allons donc nous contenter de citer une SVM simple : le séparateur linéaire.

Séparateur linéaire : C'est le cas simple des SVM, un séparateur linéaire est une fonction de classification (discrimination) obtenue par combinaison linéaire d'un vecteur d'entrée $X=(x_1, x_2,\ldots,x_n)^T$ avec un vecteur de poids $W=(w_1,w_2,\ldots,w_n)^T$:

$$H(X)=W^TX + W_0 \quad (9)$$

Il est donc décidé que X est de classe 1 si H(X)>0, et de classe -1 sinon.

La frontière de décision {\displaystyle h(x)=0}H(X)=0 est un hyperplan, appelé hyperplan séparateur, ou séparatrice.

Le but d'un algorithme d'apprentissage supervisé est d'apprendre la fonction H(X) par le biais d'un ensemble d'apprentissage :

$$\{(x_1, c_1), (x_2, c_2),\ldots\ldots,(x_m, c_m),\} \subset R^n \times \{-1, 1\}$$

Où

- les c_i sont les labels,
- m est la taille de l'ensemble d'apprentissage,
- n la dimension des vecteurs d'entrée.

Si le problème est linéairement séparable, on doit alors avoir :

$C_k h(x_k) \geq 0 \qquad 1 \leq k \leq m$

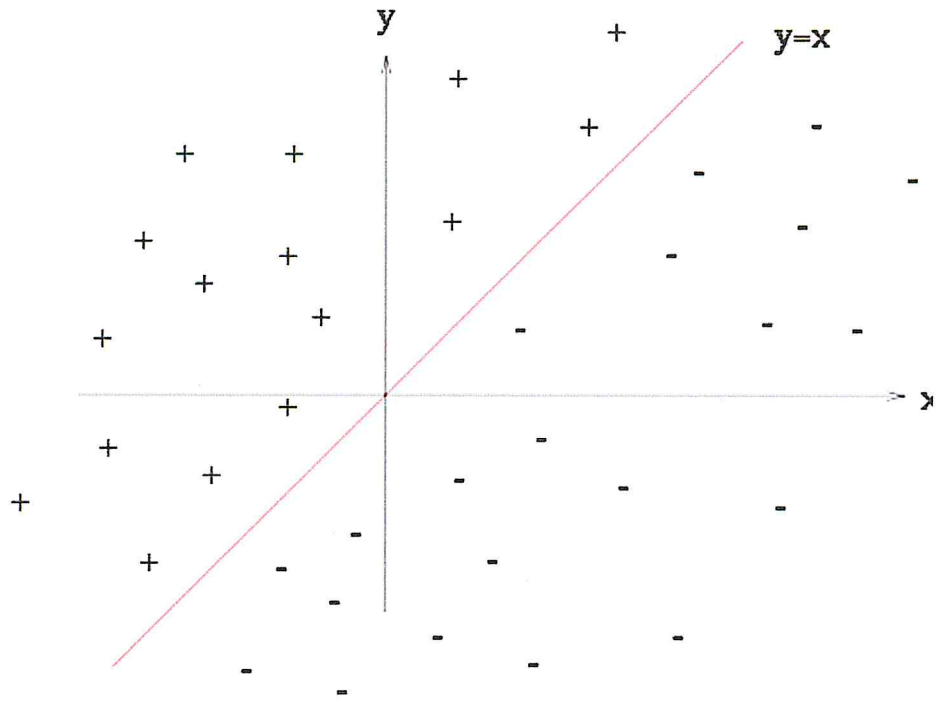


Figure 6 Séparateur linéaire

Le problème le plus connu dans la discrimination linéaire est de trouver l'hyperplan séparateur qui divise au mieux l'ensemble d'apprentissage. En effet, il y a toujours dans n'importe quel cas une infinité de droites qui permettent la séparation, donc le problème est de trouver l'hyperplan optimal.

Pour résoudre ce problème, il a été montré qu'il existe un unique hyperplan optimal, défini comme l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur. A partir de ce résultat, il n'est plus difficile d'attendre l'hyperplan optimal grâce à des calculs de distances et des formulations duales. **[19]**.

4. *Le boosting :*

C'est une technique d'apprentissage qui permet de rendre une méthode de classification encore plus puissante qu'elle est déjà.

Le principe de la méthode est de prendre une technique d'apprentissage quelconque un peu meilleure que le hasard (à 55% prés) et un ensemble d'exemples difficiles à classifier avec la méthode choisie.

Ensuite, on divise les données collectées en échantillons, on applique la méthode considérée sur chacun des échantillons et à chaque fois on tire une décision d'apprentissage.

A la fin, on combine les différentes règles tirées en utilisant un système de vote pondéré de ces règles et on déduit des règles de classification plus efficaces.

Ceci n'est que le principe théorique de la méthode. Le boosting possède plusieurs algorithmes qui ont implémentés son principe de divers manière, on cite par exemple : le boostraping, le bagging, le rankboost…et principalement l'AdaBoosting. **Adapté de [20], [21], [22].**

Du point de vue implémentation, le boosting n'est pas proprement dit une technique d'apprentissage mais un moyen de renforcer les méthodes d'apprentissage connus.

5. *Méthode des K plus proches voisins :*

C'est une méthode d'apprentissage supervisé. En abrégé k-NN ou KNN, de l'anglais *k-nearest neighbor*. **[23]**

Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de N couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir.

Par exemple, dans un problème de classification, on retiendra la classe la plus représentée parmi les k sorties associées aux k entrées les plus proches de la nouvelle entrée x.

Pour savoir le nombre k représentant le nombre de voisins à utiliser, on peut utiliser par exemple *la validation croisée*.

6. *Classifieur naïf de bayes :*

Le classifieur naïf bayésien est l'une des méthodes les plus simples en apprentissage supervisé basée sur le théorème de Bayes.
Il est peu utilisé par les ingénieurs du data mining au détriment des méthodes traditionnelles que sont les arbres de décision ou les régressions logistiques.
Un avantage de cette méthode est la simplicité de programmation, la facilité d'estimation des paramètres et sa rapidité (même sur de très grandes bases de données). Malgré ses avantages, il est peu utilisé en pratique en raison de l'absence d'un modèle explicite simple de la méthode. Il est dit naïf car les variables explicatives sont supposées être libres entre elles conditionnellement par rapport à la variable cible.

Principe : Ce classifieur est essentiellement basé sur la théorie de Bayes, pour une variable de classe C et un ensemble de variables explicatives E_1, E2,..... dont elle dépend la probabilité d'apparition d'un classe c sachant e_1, e2,..... e_n est :

$$P(c \mid e_1,e_2,.....,e_n)=P(c)*P(e_1,e_2,.....e_n \mid c)/P(e_1,e_2,.....e_n) \quad (10)$$

Grace à un peu de simplification et en appliquant la propriété de naïveté du modèle (Indépendance entre les variables explicatives E) on obtient :

$$P(c \mid e_1, e_2,..... e_n)= P(c)*\prod_{i=1}^{n} P(e_i \mid c)/ P(e_1, e_2,.....e_n)$$

Le classifieur couple ce modèle de probabilités avec une règle de décision. Une règle couramment employée consiste à choisir l'hypothèse la plus probable. Il s'agit de la règle du maximum a posteriori ou MAP. En gros, on calcule les probabilités d'appartenance de la variable considérée à toutes les classes prédéfinies, ensuite on choisit celle avec la plus grande valeur. Le classifieur correspondant à cette règle est la fonction {\displaystyle \mathrm {classifieur} } Classifieur suivante :

$$\text{Classifieur (e1,.....en)}=\text{argmax}P(C=c)\prod_{i=1}^{n} p(Ei=ei \mid C=c) \quad (11)$$

Malgré la simplicité de cette méthode, son efficacité dans les cas pratiques a été démontrée plus qu'une fois. En 2004 un article a montré qu'il existe des raisons théoriques derrière cette efficacité inattendue **[24]**.

7. *Autres méthodes :*

Jusqu'à maintenant nous avons présenté que les méthodes les plus connues et les plus utiliser en apprentissage automatique. Mais bien sûr il y a plusieurs autres méthodes non citées comme : les réseaux bayésiens fondés sur la théorie de Bayes et la condition de Markov **[25]**, les forets aléatoires dont le premier algorithme de décision a été introduit par Breiman **[26]**,les algorithmes génétiques initialement introduits par David Goldberg à

travers son livre *Genetic Algorithms in Search, Optimization, and Machine Learning* **[27]**, Modèle de mélanges gaussiens **[28]**, courbe ROC **[29]**…..etc.

5. ETUDE COMPARATIVE ENTRE LES TECHNIQUES EXISTANTES DE L'ANALYSE PREDICTIVE :

Tableau 3 Comparaison entre les méthodes d'analyse prédictive

	Avantages	Inconvénients
Arbre de décision	1. Simple à mettre en œuvre et efficace. 2. Facilement compréhensible en raison de sa logique booléenne (si alors sinon). 3. Peu de préparation des données (pas de normalisation, de valeurs vides à supprimer…). 4. Le modèle peut gérer à la fois des valeurs numériques et des catégories. 5. Il est possible de valider un modèle à l'aide de tests statistiques, et ainsi de rendre compte de la fiabilité du modèle.	1. L'apprentissage de l'arbre de décision optimal est NP-complet concernant plusieurs aspects de l'optimalité. En conséquence, les algorithmes d'apprentissage par arbre de décision sont basés sur des heuristiques telles que les algorithmes gloutons cherchant à optimiser le partage à chaque nœud, et de tels algorithmes ne garantissent pas d'obtenir l'optimum global. Certaines méthodes visent à diminuer l'effet de la recherche gloutonne. 2. Certains concepts sont difficiles à exprimer à l'aide d'arbres de décision comme le XOR par exemple. 3. L'apprentissage par arbre de décision peut amener des arbres de décision très complexes, qui généralisent mal l'ensemble d'apprentissage (il s'agit du problème de sur apprentissage précédemment évoqué). On utilise des procédures d'élagage pour contourner ce problème, certaines approches comme l'inférence conditionnelle permettent de s'en affranchir

Réseaux de neurones	1. Capacité de représenter n'importe quelle fonction, linéaire ou pas, simple ou complexe. 2. Faculté d'apprentissage à partir d'exemples représentatifs, par « rétro propagation des Erreurs ». L'apprentissage (ou construction du modèle) est automatique 3. Résistance au bruit ou au manque de fiabilité des données. 4. Comportement moins mauvais en cas de faible quantité de données.	1. L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la (ou les) couche(s) cachée(s). 2. Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage, qui jouent un rôle important dans la vitesse de convergence. 3. Le problème du sur apprentissage.
Machines à vecteurs de support	1. Capacité à traiter de grandes dimensionnalités. 2. Robuste et puissante. 3. Traitement des problèmes non linéaires. 4. Non paramétrique 5. Souvent performant dans les comparaisons avec les autres approches	1. Difficulté à identifier les bonnes valeurs des paramètres (et sensibilité aux paramètres). 2. Difficulté à traiter les grandes bases de données. 3. Problème lorsque les classes sont bruitées. 4. Difficulté d'interprétations (ex : pertinence des variables). 5. Le traitement des problèmes multi-classes reste une question ouverte
Classifieur naïf de Bayes	1. Modèle robuste et rapide. 2. Capable de traiter de grands volumes de données. 3. C'est un modèle linéaire même au niveau des performances.	1. Nombre de règles égale au nombre de combinaisons de descripteurs (dans la pratique, les règles ne sont pas formées, nous conservons les probabilités conditionnelles que nous appliquons pour chaque individu à classer. 2. Pas de modèle explicite.
Méthode des k plus proches voisins	1. Simplicité, pas d'apprentissage d'un modèle (lazy learning). 2. Incrémentalité (garder à disposition les individus de la base). 3. Bonnes performances en général.	1. Paramétrage difficile (choix de la taille du voisinage). 2. Impossibilité d'interprétation d'un classement proposé. 3. Nécessité de garder sous la main la base de données. 4. Sensibilité à la dimensionnalité (et aux variables non pertinentes).

Adapté de [30], [31], [32], [33] et [34].

A partir de ce tableau nous avons décidé d'adopter pour notre travail la méthode basée sur les arbres de décision et cela est principalement due au fait que les arbres de décision sont faciles à comprendre grâce à leurs structures conditionnelles.

De ce fait même un non informaticien ou un non mathématicien pourra comprendre le modèle contrairement aux autres structures qui sont un peu plus complexes.

En plus de cet avantage nous trouvons l'avantage de l'efficacité et la capacité d'adopter le modèle à différentes situations car les arbres de décision peuvent gérer différents types de variables et grâce à cela on peut trouver différents jeux de données qui permettent de tester la validité et la puissance de notre outil.

Comme expliqué en haut, il y a plusieurs algorithmes qui permettent de construire des arbres de décision : C4.5, CART, CHAID…. Nous avons décidé d'adopter l'algorithme CART. C'est un algorithme utiliser pour les cas de classification comme ceux de régression, il est alors d'une large utilisation. De plus, il permet l'utilisation de variables de tous types (continues, discrètes et catégoriques). A mentionner aussi que des travaux concernant les traitements parallèles des algorithmes comme C4.5 ont déjà été réalisés, ce qui n'est pas le cas de l'algorithme CART.

Notre travail consistera donc à implémenter une version parallèle Map-Reduce adaptée à l'application de l'algorithme CART sur des Big Data.

Dans ce qui suit, nous expliquerons en détails les concepts de Big Data et Map-Reduce, ce sont deux notions importantes dans le contexte de notre travail. Nous montrerons aussi l'impact de ces deux notions sur le monde de l'analyse prédictive.

6. ANALYSE PREDICTIVE, BIG DATA ET PROGRAMMATION PARALLELE :

Comme déjà mentionné, les algorithmes de l'analyse prédictive sont applicables dans le cas des Big Data. Cela se fait en parallélisant certaines instructions de ces algorithmes, des instructions qui peuvent s'exécuter en même temps.

Il existe plusieurs manières et techniques pour paralléliser des traitements séquentiels d'un algorithme, parmi lesquelles on trouve le Map-Reduce.

Maintenant se pose les questions :

- C'est quoi le Map-Reduce ?
- C'est quoi les Big Data ?

Avant d'expliquer ces deux notions, on va se pencher sur la notion de « programmation parallèle » pour mieux l'éclaircir.

6.1. La programmation parallèle : [35]

On entend par programmation parallèle ou parallélisme l'exécution simultanée d'un ensemble d'instructions ou taches d'un algorithme. Ces instructions doivent être exécutables parallèlement sans causer des erreurs dans les résultats finaux ou des inter blocages.

On peut désigner aussi par parallélisme l'exécution d'un même programme sur plusieurs fragments d'un fichier ou d'un système, puis renvoyer les résultats au système père, on parle ici de l'aspect architectural.

Un exemple d'exécution de taches parallèles est la lecture d'un ensemble de processus à partir d'un même fichier, ou par exemple l'écriture dans deux fichiers différents…

Le concept est apparue en 1962 dans la thèse de Carl Adam Petri sur les réseaux de Petri **[36]** ou il parlait d'accès simultanées à un une mémoire partagée.

Dès lors, plusieurs méthodes de parallélisme ont vu le jour. Nous, dans ce mémoire on va s'intéresser au Map-Reduce qui est conçu pour des cas de big data.

6.2. Les Big Data :

Définition :

Littéralement, le terme « Big Data » signifie méga données, grosses données ou encore données massives. Il désigne des volumes impressionnants de données qui sont créés quotidiennement par les particuliers, les entreprises et aussi les objets et machine connectés.

Cependant, aucune définition précise ou universelle ne peut être donnée au Big Data. Etant un objet complexe polymorphe, sa définition varie selon les communautés qui s'y intéressent en tant qu'usager ou fournisseur de services.

Afin de clarifier ce concept, une variété de définitions et de caractéristiques des grandes données est présentée :

Selon Knapp **[37]**, les grandes données se réfèrent à « outils, processus et procédures qui permettent à une organisation de créer, de manipuler et de gérer des ensembles de données et des installations de stockage très importants ».

Ohlhorst **[38]**, définit les Big Data comme des ensembles de données extrêmement volumineux, difficile, pour ne pas dire impossible, de gérer et d'analyser avec les outils de traitement de données traditionnels. Et plus l'ensemble de données est grand, plus il est difficile d'en tirer une valeur. Phillips-Wren et Hoskisson **[39]** se réfère à Laney **[40]** faisant

la première tentative de définir les Big Data, en les décrivant de grandes données ayant le volume, la vitesse et la variété comme caractéristiques principales.

Nous allons maintenant décrire ces caractéristiques.

Caractéristiques des big data :

Comme déjà mentionné, les big data sont par de nombreux chercheurs définit comme une source de données avec trois caractéristiques : un volume de données extrêmement important, une très grande variété de données et une haute vélocité, d'où le modèle des 3 V, définit par l'analyste Doug Laney **[40]** :

- Le volume : décrit la taille des données, qui est une caractéristique centrale de Big data aujourd'hui. Un grand volume de données se réfère simplement à la grande quantité de données créées chaque seconde.

Le second V représente :

- La variété : indiquant les différents types de données, telles que les données structurées, non structurées et semi-non structurées, Et les différentes façons d'appliquer les données.
- La vélocité : signifie que les données sont rapidement collectées et traitées en temps réel.

Certains savants notent également un quatrième V :

- La véracité (Elgendy & Elragal **[41]** ; Hurwitz et al. **[42]** ; Ohlhorst **[38]**) se référant à la précision et à la crédibilité des données et des analyses.

Le traitement de ces grands volumes de données ne pourrait se faire à l'aide des techniques classiques existantes, cela aurait un coût très important en matière de temps d'exécution. C'est pour quoi, une des voies praticables pour passer à l'échelle est le parallélisme.

Dans ce contexte, on trouve Map-Reduce qui est un pattern de répartition des taches des plus utiliser pour l'utilisation et l'analyse des Big Data.

6.3. Le Map-Reduce : [43]

Map-Reduce est un modèle de programmation qui permet le développement et le test de programmes dédiés à l'analyse des grandes masses de données distribuées sur un ensemble de nœuds. Son objectif est d'automatiser le parallélisme et la distribution du calcul sans exiger (de l'utilisateur) une supervision du système ni une expertise dans le calcul parallèle. Étant donné que le système gère l'exécution parallèle, la coordination et l'échec d'exécution des taches, le rôle du programmeur est de définir et d'implémenter les deux fonctions Map et Reduce. La figure 7 résume les étapes d'exécution d'un programme Map-Reduce qui sont:

1. La phase Map : le nœud analyse un problème, le découpe en sous-problèmes, et les délègue à d'autres nœuds (qui peuvent en faire de même récursivement). Les sous-problèmes sont ensuite traités par les différents nœuds à l'aide de la fonction Map qui à un couple (clé, valeur) associe un ensemble de nouveaux couples (clé, valeur) :
   Map (clé1, valeur1) → list (clé2, valeur2)
2. Les paires (clé, valeur) produites par les différents Mappers sont regroupées et triées suivant leurs clés. Ensuite, elles sont dirigées vers les différents nœuds Reduce de façon que toutes les paires qui ont la même clé soient dans le même nœud Reduce. Cette étape est appelée « Shuffle and Sort ».
3. La phase Reduce : Chaque Reducer traite les valeurs associées à chaque clé à la fois. Ce traitement est fixé par la fonction Reduce écrite par le programmeur.
   Ensuite, les résultats sont envoyés au nœud père qui fait de même.
   À la fin du processus, le nœud d'origine peut recomposer une réponse au problème qui lui avait été soumis : Reduce (clé2, list(valeur2)) → valeur2

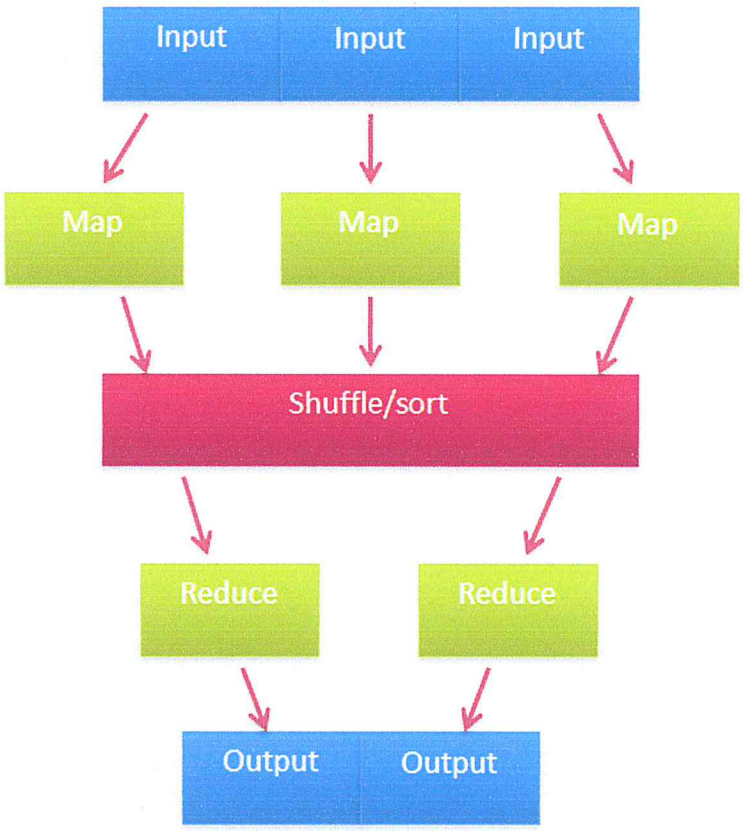


Figure 7 « Etapes d'exécution d'un algorithme Map/Reduce » sur le site « deltarhho.org »

Pour résumer, la fonction Map doit décrire le traitement qui peut être exécuté sur des parties des données initiales indépendamment des résultats sur les autres parties. Alors que la fonction Reduce décrit comment agréger des résultats de la fonction Map pour atteindre l'objectif fixé.

Map-Reduce possède quelques caractéristiques :

- Le modèle de programmation Map-Reduce est simple et très utile. Bien qu'il ne possède que deux fonctions, Map () et Reduce (), elles peuvent être utilisées pour de nombreux types de traitement des données, les fouilles de données, les graphes… Il est indépendant du système de stockage et peut manipuler de nombreux types de variable.
- Le système découpe automatiquement les données en entrée en bloc de données de même taille. Puis, il planifie l'exécution des tâches sur les nœuds disponibles.
- Il fournit une tolérance aux fautes à grain fin grâce à laquelle il peut redémarrer les nœuds ayant rencontré une erreur ou affecter la tâche à un autre nœud **[44]**.
- La parallélisation est invisible à l'utilisateur afin de lui permettre de se concentrer sur le traitement des données **[45]**.

Une fois qu'un nœud a terminé une tâche, on lui affecte un nouveau bloc de données. Grâce à cela, un nœud rapide fera beaucoup plus de calculs qu'un nœud plus lent. Le nombre de tâches Map ne dépend pas du nombre de nœuds, mais du nombre de blocs de données en entrée. Chaque bloc se fait assigner une seule tâche Map. De plus, toutes les tâches Map n'ont pas besoin d'être exécutées en même temps en parallèle, les tâches Reduce suivent la même logique. Par exemple, si des données en entrée sont divisées en 400 blocs et qu'il y a 40 nœuds dans le cluster, le nombre de tâches Map sera de 400. Il faudra alors 10 vagues de Map pour réaliser le mapping des données **[46]**.

Le Map-Reduce est apparu en 2004. La technologie est encore jeune. Elle souffre de quelques points faibles **[47]** :

- Elle ne supporte pas les langages de haut niveau comme le SQL
- Elle ne gère pas les index. Une tâche Map-Reduce peut travailler une fois les données en entrée stockées dans sa mémoire. Cependant, Map-Reduce a besoin d'analyser chaque donnée en entrée afin de la transformer en objet pour la traiter, ce qui provoque des baisses de performance.
- Elle utilise un seul flot de données. Map-Reduce est facile à utiliser avec une seule abstraction mais seulement avec un flot de donnée fixe. Par conséquent, certains algorithmes complexes sont difficiles à implémenter avec seulement les méthodes Map () et Reduce (). De plus, les algorithmes qui requièrent de multiples éléments en entrée ne sont pas bien supportés car le flot de données du Map-Reduce est prévu pour lire un seul élément en entrée et génère une seule donnée en sortie.
- Quelques points peuvent réduire les performances de Map-Reduce. Avec sa tolérance aux pannes et ses bonnes performances en passage à l'échelle, les opérations de Map-Reduce ne sont pas toujours optimisées pour les entrées/sorties. De plus, les méthodes Map () et Reduce () sont bloquantes. Cela signifie que pour passer à l'étape suivante, il faut attendre que toutes les tâches de l'étape courante soient terminées. Map-Reduce n'a pas de plan spécifique d'exécution et n'optimise pas le transfert de données entre ses nœuds.

On va maintenant donner un exemple d'un calcul qui peut se faire dans un nœud Map-Reduce :

Supposons qu'on a un fichier contenant la ligne suivante :

1/ "A A A B C D A V B C C K"

Le but est de calculer le nombre d'occurrence de chaque lettre en utilisant le principe Map-Reduce.

On suppose que l'ensemble d'entrée de la fonction Map est le couple <clé, valeur> ou la clé correspond au numéro de la ligne (1 dans cet exemple) et la valeur est le contenu de la ligne ("A A A B C D A V B C C K").

La phase « Map » va générer des couples <clé, valeur> à partir du couple d'entrée ou chaque clé dans ce cas correspond à une lettre lue, et la valeur correspond à 1.

On aura dans cette étape ce qui suit :

<A, 1>
<A, 1>
<A, 1>
<B, 1>
<C, 1>
<D, 1>
<A, 1>
<V, 1>
<B, 1>
<C, 1>
<C, 1>
<K, 1>

1. La phase « Shuffle/sort » va ordonner les couples ayant la même clé et les mettre dans le même nœud de calcul :

A, [1, 1, 1, 1]
B, [1, 1]
C, [1, 1, 1]
D, [1]
V, [1]
K, [1]

2. La phase « Reduce » va sommer les valeurs d'une même clé :

<A, 4>
<B, 2>
<C, 3>
<D, 1>
<V, 1>
<K, 1>

6.4. Application de l'analyse prédictive dans le domaine du big data :

En définissant des modèles Map-Reduce ou autre modèles de programmation parallèle sur les algorithmes prédictifs, on peut directement les utiliser sur des Big Data. Et c'est par ailleurs ce qu'on est en train de faire : définir une version Map-Reduce de CART et l'appliquer sur des Big Data.

L'application de l'analyse prédictive sur des Big Data a été plus qu'un succès dans les années passées. Les algorithmes prédictifs se sont retrouvés capables de prédire des résultats fiables en se basant sur des données de types Big Data. Ceci est dû, premièrement à l'efficacité de ces algorithmes qui ont été mis au point sur la base de fondements mathématiques infaillibles. Et deuxièmement à la puissance des méthodes de parallélisme tel le Map-Reduce. On peut donc dire que les Big Data et la programmation parallèle ont rendu l'analyse prédictive plus opérationnelle **[48].**

7. CONCLUSION :

Ce chapitre a servi à présenter les techniques utilisées dans l'analyse prédictive et généralement l'apprentissage statistique. Nous avons choisi à la fin de ce chapitre l'algorithme qu'on va utiliser pour développer un outil prédictif qui est l'algorithme CART.

Nous avons aussi expliqué les principaux concepts qui seront utilisés pour le développement d'une version parallèle de l'algorithme.

Dans le prochain chapitre, nous expliquerons en détail CART avec tous les principes et les notions mathématiques qui lui sont reliées.

CHAPITRE II :

L'ALGORITHME CART

1. INTRODUCTION :

Il existe beaucoup d'algorithmes qui permettent de faire de l'apprentissage automatique, cela veut dire apprendre le comportement d'un système sur la base d'ensembles d'enregistrements de données.

Ces algorithmes permettent de construire des classifieurs ou prédicteurs, des structures de données qui peuvent prévoir un résultat précis si elles reçoivent l'entrée correspondante. On distingue plusieurs classifieurs : les arbres de décision, les réseaux de neurones artificiels, le classifieurs de Bayes…

L'apprentissage à base d'arbres de décision est le plus connu et le plus utilisé en raison de sa simplicité et de sa logique conditionnelle facilement compréhensible.

Les algorithmes d'apprentissage à base d'arbres de décision sont variés mais ils prennent la même démarche globale :

- Construire un arbre qui fait de la prédiction en utilisant un critère de segmentation pour définir le corps de l'arbre (comme l'indice de Gini, l'entropie de Shannon, le test du khi-deux …)
- Simplifier l'arbre en supprimant les branches les moins informatives tout en gardant un taux d'efficacité élevée, c'est le processus d'élagage.

Le processus d'élagage peut se faire avant le début de construction de l'arbre en fixant des règles ou des conditions d'arrêt (Pré-élagage), ou après la construction de l'arbre en remplaçant certains nœuds internes par des feuilles (post-élagage).

Les algorithmes générateurs d'arbres de décision se différencient dans le critère de segmentation et le processus d'élagage.

Parmi ces algorithmes on trouve CART qui est l'objet de notre étude.

CART (Classification And Régression Tree) publié par Breiman et al **[12]** en 1984 construit des arbres binaires avec l'indice de Gini comme critère de segmentation et une opération de post-élagage, il est utilisé en classification tout comme en régression. C'est un algorithme très efficace.

Dans notre mémoire, on ne va s'intéresser qu'au cas de classification (prédire les valeurs d'une variable qualitative) car c'est le plus répandu parmi les problèmes de prédiction.

2. PRINCIPE DE L'ALGORITHME CART :

Comme expliqué en dessus CART utilise l'indice de Gini, qu'on va détailler par la suite, pour construire un arbre maximal à partir d'un ensemble d'enregistrements de données. Ensuite, il utilise une opération de post élagage pour simplifier l'arbre maximal.

On peut représenter notre algorithme comme suit :

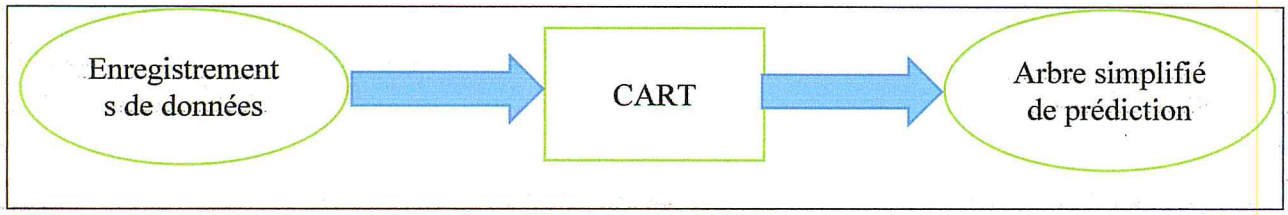


Figure 8 Représentation de CART

On peut le représenter de manière plus détaillé comme suit :

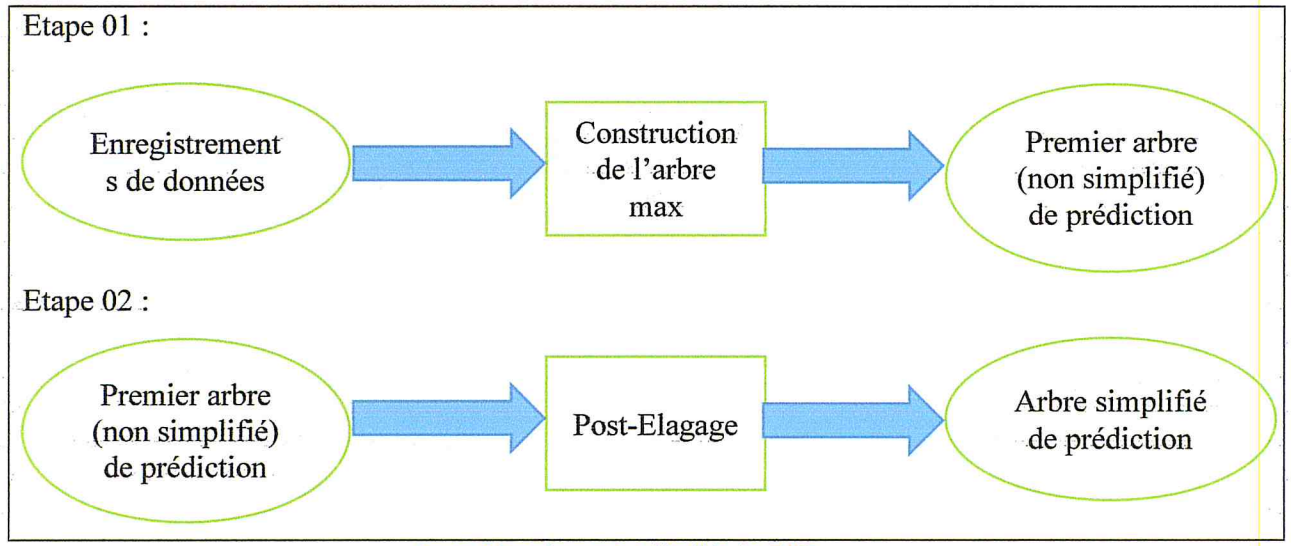


Figure 9 Représentation détaillée de CART

On va maintenant expliquer toutes les notions et les opérations de notre algorithme. A savoir :

1. L'explication du critère de Gini utilisé dans la construction du premier arbre non élagué.
2. L'explication des phases de l'algorithme : l'étape de génération de l'arbre maximal et l'étape de post-élagage de cet arbre.

2.1. Critère de Gini [49] :

Le coefficient de Gini est une mesure statistique de la dispersion d'une distribution dans une population donnée, développée par le statisticien italien Corrado Gini. Le coefficient de Gini est un nombre variant de 0 à 1, où 0 signifie l'égalité parfaite et 1 signifie une inégalité parfaite (par exemple un seul salarié dispose de tous les revenus et les autres n'ont aucun revenu).

Ce coefficient est très utilisé pour mesurer l'inégalité des revenus dans un pays.

Mathématiquement, le coefficient de Gini est équivalent à l'*écart moyen relatif* (l'écart moyen divisé par la moyenne pour le mettre à l'échelle) : il s'agit donc bien d'une mesure de dispersion de valeurs numériques ; dans le cas de revenus d'une mesure d'inégalité économique.

Cette mesure se calcule comme suit :

Considérons un ensemble d'enregistrements de données de la forme (x_1, x_2, …, x_n, y) tel que les x_i sont des variables de prédiction et y une variable à prédire.

En supposant que y prend m valeurs possibles (donc on a m classes possibles) et S désigne l'ensemble des enregistrements à disposition et S_i est le groupe des enregistrements appartenant à la classe i on a :

$$\text{Gini(S)} = \sum_{i=1}^{m} fi(1 - fi) = 1 - \sum_{i=1}^{m} fi^2$$

Avec f_i = card(S_i)/card(S).

Exemple :

Considérons un tableau représentant les résultats de matchs d'une équipe de football en fonction de certaines conditions :

Tableau 4 Résultats de matchs d'une équipe de football

Match à domicile	Balance positive	Mauvaises conditions climatiques	Match précédemment gagné	Match gagné
v	v	v	f	v
f	v	v	f	v
f	f	f	v	f

Là on a card(S) = 3 et on a deux classes : match gagné et match perdu, donc

- m=2,
- Card (Match gagné) = 2
- card (match perdu) = 1.

L'indice de Gini correspondant à cet échantillon S est :

Gini(S)= $1-((2/3)^2+(1/3)^2) = 1-(4/9) -(1/9) = 4/9$.

2.2. Phases de l'algorithme CART [50] :

2.2.1. Construction de l'arbre maximal de CART :

Comme évoqué plus haut CART construit des arbres binaires et il considère tout type de variables : quantitative ou qualitative pour les variables de prédiction ou la variable à prédire.

Le cas qui nous intéresse ici est le cas d'une variable qualitative à prédire, en d'autre mot on s'intéresse au cas de classification.

Nous allons maintenant expliquer les démarches de CART dans le cas de classification.

CART commence par construire un arbre maximal non simplifié qui prend en considération tous les enregistrements de données disponibles, le processus est le suivant :

L'arbre construit par notre algorithme est généré à partir d'un ensemble d'enregistrements de données appelées données d'apprentissage, elles sont de la forme : $(x_1, x_2, \ldots, x_n, y)$, ou chaque x_i est appelé attribut et sera par la suite un nœud interne ou racine dans l'arbre maximal T_{max}.

Les valeurs prises par y sont les classes d'appartenance et sont représentées dans les feuilles de l'arbre.

Par exemple, on veut prédire si un étudiant va réviser ses cours dans la soirée en fonction de certains paramètres :

- P_1 : est-ce que l'étudiant a une séance de sport à 17h.
- P_2 : est-ce que l'étudiant a des affaires urgentes à régler après l'école.
- P_3 : est-ce que l'étudiant est fatigué.
- P_4 : est-ce que l'étudiant est malade.

Donc, nos x_i sont les P_i et y, la variable à prédire prend deux valeurs :

→ Y=réviser.

→ Y=ne pas réviser.

Les valeurs de y sont donc nos classes, on a de ce fait deux : « réviser » et « ne pas réviser ».

CART commence par poser toutes les questions binaires possibles pour chaque attribut x_i afin de déterminer la racine de l'arbre et la question choisie est celle qui maximise le gain d'information exprimé comme suit :

Gain (x_j)= Gini(S)- fg.Gini(S_g)-f_d.Gini(S_d)

- S : est l'ensemble de tous les enregistrements.
- x_j : l'attribut x_j, il prend deux valeurs : x_j=a ou x_j=non a.
- S_g : représente les enregistrements pour lesquels x_j=a.
- S_d : représente les enregistrements pour lesquels x_j=non a.
- f_d =card(S_d)/card(S).
- f_d =card(S_d)/card(S).

On fait ce procédé pour chaque attribut pour déterminer la question à poser pour chacun d'eux, après ça la question posée dans la racine de l'arbre sera celle présentant le gain maximum.

On refait récursivement le même procédé pour les deux branches de l'arbre.

On arrête quand l'un des trois cas suivants se présente :

1. Quand on arrive à un nœud ou tous les exemples correspondants ont la même classe (On dit dans ce cas que l'ensemble est pur et l'indice de gini qu'il lui correspond est nul), alors ce nœud devient terminal (une feuille) et on lui attribue la classe correspondante.
2. Quand on arrive à un nœud possédant peu d'exemples (Environ 10% ou moins des enregistrements totaux **[51]**) alors ce nœud deviendra terminal et on lui attribue la classe majoritaire. Si on n'a pas de classe majoritaire dans l'ensemble considéré on prend celle de tout l'ensemble sinon on lui attribue une classe aléatoire.
3. Quand on arrive à un cas ou on n'a plus d'attributs pour diviser, le nœud devient terminal et on lui attribue une classe de la même manière que dans 2.

Exemple :

On va considérer un exemple qui nous permet de savoir est ce qu'on peut jouer au ballon en fonctions de certaines conditions :

Tableau 5 Exemple 2

Condition	Température	Humidité	Vent	Jouer
Soleil	Chaud	Elevée	Faible	Non
Soleil	Chaud	Elevée	Fort	Non
Nuage	Chaud	Elevée	Faible	Oui
Pluie	Doux	Elevée	Faible	Oui
Pluie	Froid	Normale	Faible	Oui
Pluie	Froid	Normale	Fort	Non
Nuage	Froid	Normale	Fort	Oui
Soleil	Doux	Elevée	Faible	Non
Soleil	Froid	Normale	Faible	Oui
Pluie	Doux	Normale	Faible	Oui
Soleil	Doux	Normale	Fort	Oui
Nuage	Doux	Elevée	Fort	Oui
Nuage	Chaud	Normale	Faible	Oui
Pluie	Doux	Elevée	Fort	Non

On a dans cet exemple 4 attributs : Condition, Température, Humidité et Vent et 2 classes : jouer « Oui » et ne pas jouer « Non ».

CART commence par générer toutes les questions binaires possibles pour chaque attribut, ensuite il calcule le gain pour chaque cas, la question qui maximise ce gain sera choisie.

Par exemple pour le premier attribut si on pose la question est ce qu'on a soleil ou pas notre ensemble de départ va être divisé en deux sous ensemble :

Un ensemble gauche S_g Contenant que les attributs ou Condition=Soleil.

Un ensemble droit S_d contenant les attributs ou Condition=Nuage ou Pluie.

Card(S_g)=5, Card(S_d)=9

Gain=Gini(S)-(5/14)*Gini(S_g)-(9/14)*Gini(S_d)=0.065.

Donc pour S_g=Soleil on a un gain de 0.065.

De la même manière on calcule les deux autres cas restants :

→ Pour S_g=Nuage, Gain=0.102

→ Pour S_g=Pluie, Gain=0.002

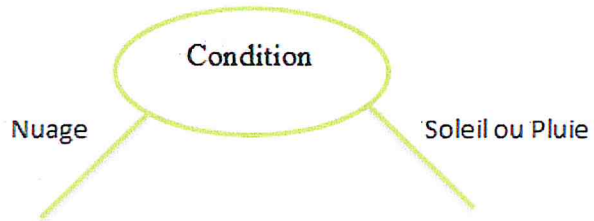
On voit que c'est dans le cas S_g =Nuage ou on a le gain max donc la division qu'on va choisir pour l'attribut température s'il venait à être choisi comme racine est par rapport à la valeur Nuage.

De la même manière, on calcule les meilleures divisions pour les autres attributs, on obtient :

Tableau 6 Résultats des calculs

Attribut	Ensemble de gauche	Ensemble de droit	Gain
Température	Chaud	Doux ou froid	0.0163
Humidité	Elevée	Normale	0.091
Vent	Faible	Fort	0.031

On voit que l'attribut qui maximise le gain est condition, donc c'est lui qui va être choisi comme racine avec la division correspondante.

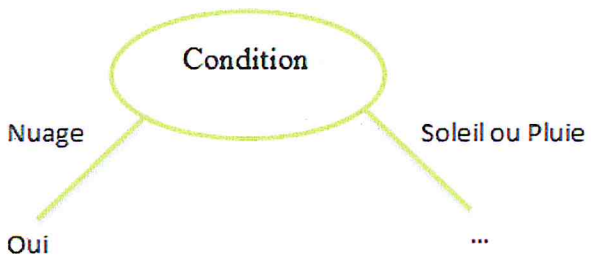


Deux nœuds en résultent de cette division, pour continuer nous devons d'abord voir si l'un deux est terminal en utilisant les trois conditions citées en haut.

Pour le fils droit, on a encore des attributs sur lesquels on peut diviser, le nœud est impur et on a beaucoup d'enregistrements pour lesquels Condition = Soleil ou pluie (8 > à 10% de 14) donc le nœud n'est pas terminal.

Pour le fils gauche, on remarque que tous les exemples appartiennent à la même classe « Oui » donc le nœud est pur, il est donc un nœud terminal et on lui affecte la classe « Oui ».

A ce niveau notre arbre prend la forme suivante :



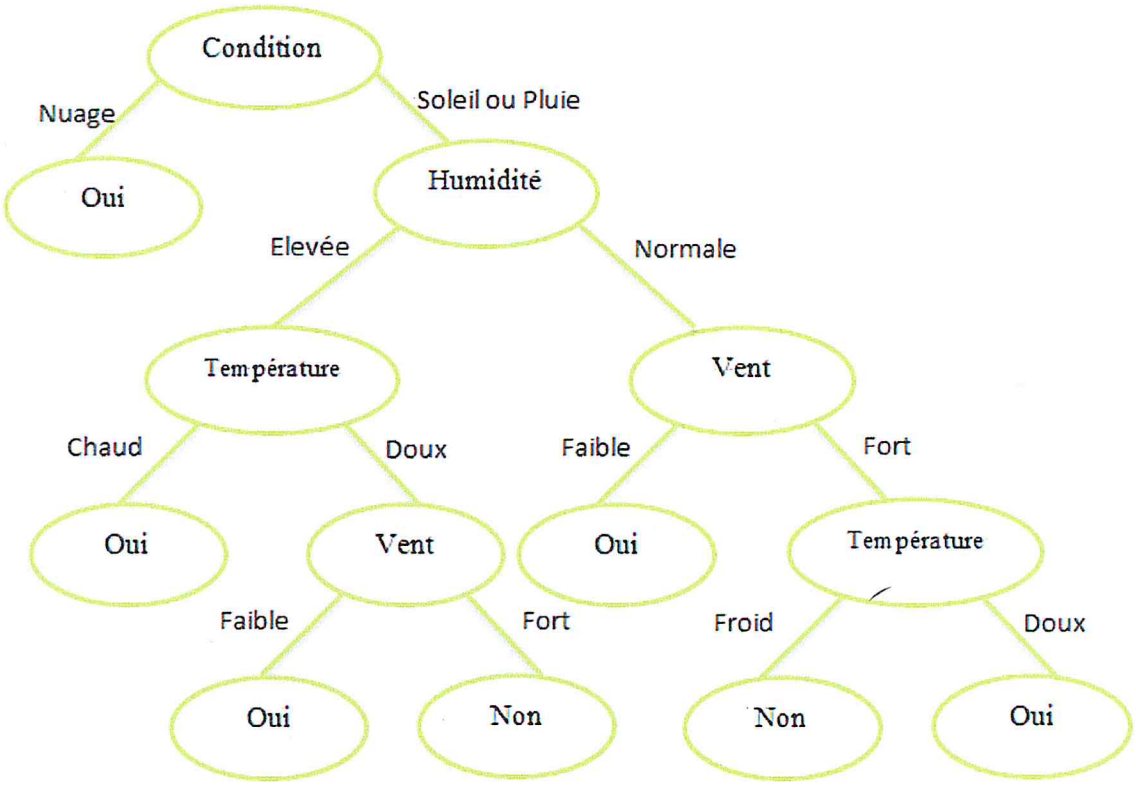


Figure 10 Arbre max CART

2.2.2. Post-élagage CART :

Une fois l'arbre T_{max} construit CART calcule un critère de complexité à partir de l'erreur produite par le jeu d'apprentissage pour chaque nœud de l'arbre.

Celui qui minimise ce critère est remplacé par une feuille dont la classe est la classe majoritaire à son niveau.

Ensuite, on fait le même travail sur l'arbre résultant jusqu'à ce qu'on arrive à la racine.

Un fois le processus terminé, on obtient une suite d'arbre emboités : T_1, T_2, …,T_n tel que T_n est la racine de T_{max}, ensuite on utilise un ensemble de validation différent de celui de l'apprentissage. L'arbre qui commet le moins d'erreurs sur cet ensemble est retourné par l'algorithme.

Critère (T_k,d)= (MC (d,k)-MCT(d,k)) / (N(k)* (Nt(d,k)-I)) (12)

Où MC (d,k) est le nombre d'exemples mal classés du jeu d'apprentissage par le nœud d de l'arbre T_k quand on fait l'hypothèse qu'il a été transformé en feuille,

MCT (d,k) est le nombre d'exemples mal classés par les feuilles du nœud T_k situé sous le nœud d, N(k) représente le nombre de feuilles de T_k et Nt (d,k) représente le nombre de feuilles du sous arbre de Tk situé sous le nœud d.

Exemple :

Considérons l'arbre construit dans l'exemple précédant :

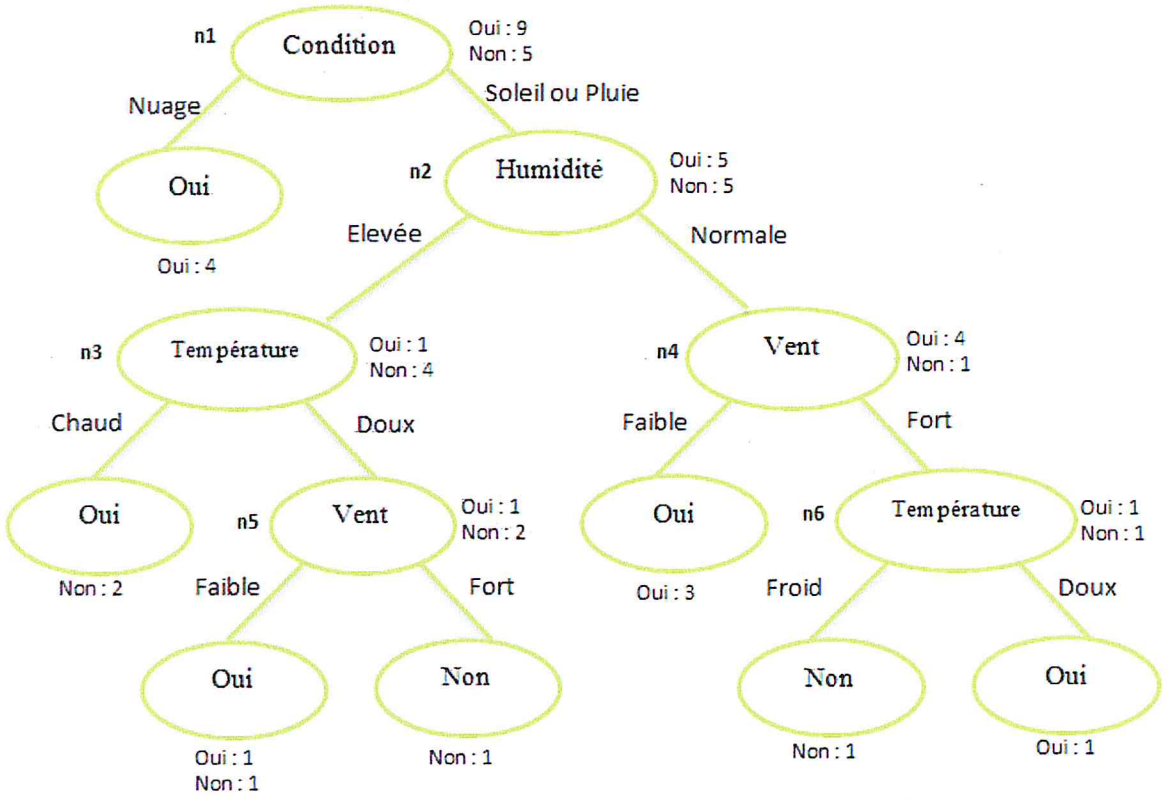


Figure 11 Arbre T0

Les valeurs devant chaque nœud indiquent combien d'exemples ont été attribués aux classes « Oui » et « Non » respectivement par les feuilles se trouvant en dessous du nœud considéré.

Calculons pour chaque nœud la valeur de son critère :

Critère $(T_0, n_1) = (5-1) / (7*(7-1)) = 0.0095238$

Critère $(T_0, n_2) = (5-1) / (7*(6-1)) = 0.112857$

Critère $(T_0, n_3) = (1-1) / (7*(3-1)) = 0$

Critère $(T_0, n_4) = (1-0) / (7*(3-1)) = 0.007142$

Critère (T_0, n_5) = (1-1) / (7*(2-1)) =0

Critère (T_0, n_6) = (1-0) / (7*(2-1)) =0.1428714

On voit que le minimum est obtenu dans deux nœuds : n_3 et n_5 et étant donné que si on supprime la branche de n_3 l'arbre sera plus simple que lorsqu'on supprime la branche de n_5 alors c'est n_3 qui va être remplacé par une feuille à quelle on attribue la classe « Non » car c'est la classe majoritaire (4 Non et 1 Oui).

Ainsi nous obtenons l'arbre T_1 :

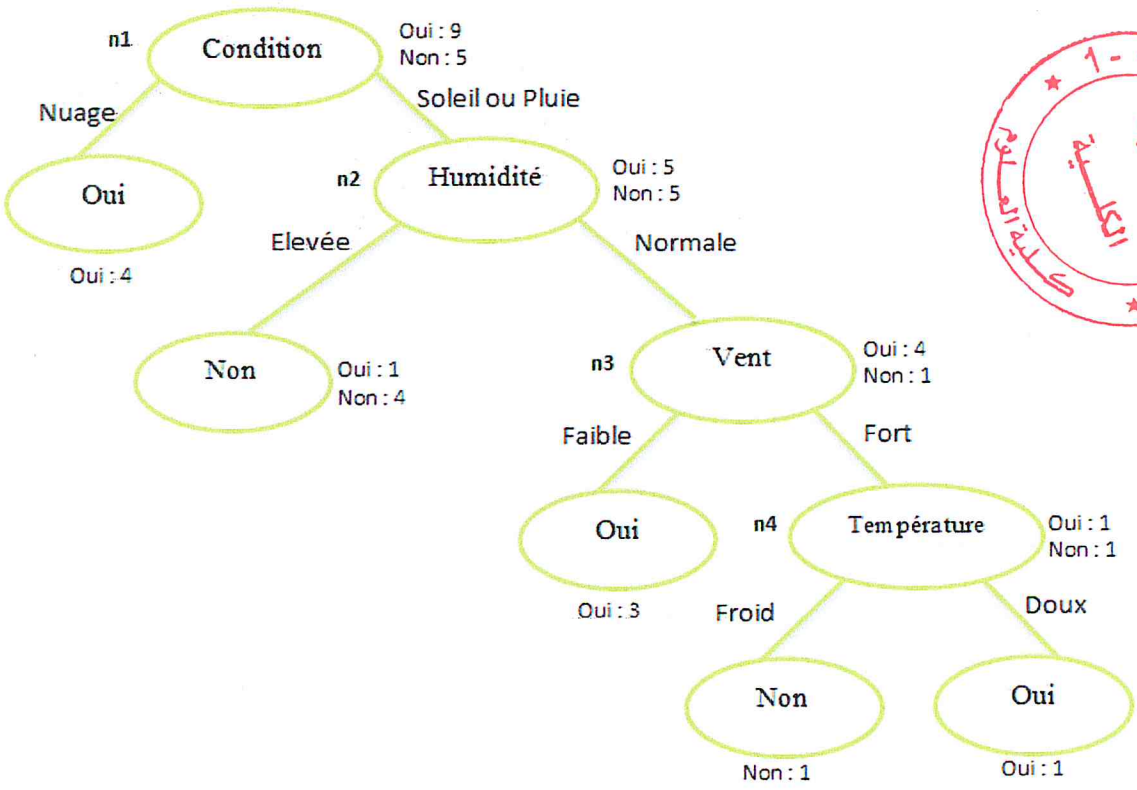


Figure 12 Arbre T1

En faisant la même opération sur cet arbre on obtient l'arbre T_2 suivant :

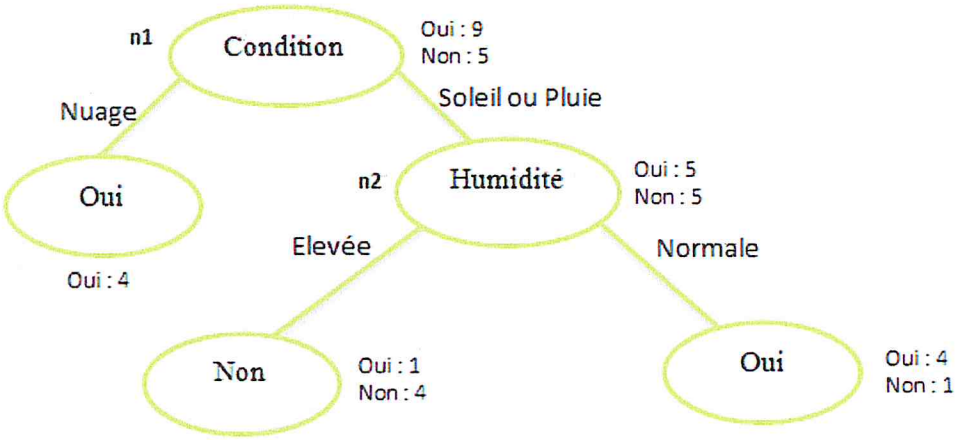


Figure 13 Arbre T2

En exécutant l'algorithme une autre fois on obtient l'arbre T_3 :

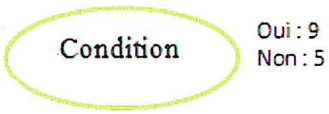


Figure 14 Arbre T3

Etant donné que T_3 est la racine alors l'algorithme s'arrête.

Maintenant pour choisir l'arbre optimal parmi les trois obtenu (T_1, T_2 et T_3), nous utiliserons l'ensemble de validation présenté dans le tableau 5. L'arbre qui commet le moins d'erreurs de prédiction est notre arbre.

Tableau 8 Ensemble de validation

Condition	Température	Humidité	Vent	Jouer
Soleil	Chaud	Elevée	Fort	Non
Soleil	Chaud	Elevée	Faible	Oui
Pluie	Froid	Normal	Fort	Non
Nuage	Doux	Normal	Faible	Non

Dans ce cas c'est l'arbre T_1 qui commet le moins d'erreurs donc c'est lui qui va être retourné par l'algorithme.

L'exemple qu'on a traité montre comment prédire une variable qualitative en utilisant des variables de prédictions qualitatives aussi. On va maintenant montrer comment faire si nos x_i

(variables de prédiction) étaient continues, quantitatives ou discrets ou si on a des enregistrements incohérents ou manquants.

2.3. Cas des variables discrètes, quantitatives et continues [50] :

De manière générale, quel que soit le type des variables utilisées dans la prédiction (qualitatives, quantitatives, discrètes ou continues) la procédure reste la même.

La seule différence c'est dans la manière à utiliser pour générer les questions binaires propres à chaque attribut. On va expliquer pour chaque type d'attribut le principe à suivre.

Une variable discrète est une variable qui prend un nombre fini de valeurs numériques.

Dans ce cas, CART identifie toutes les valeurs possibles puis pose toutes les questions binaires possibles sur la valeur prise par cet attribut, la question choisie est celle qui maximise le gain d'information. C'est le même principe que précédemment.

Exemple : Supposons qu'on a un attribut x qui prend 3 valeurs possibles a, b et c. Dans ce cas on a 3 questions possibles :

Tableau 9 Liste des questions possibles

Ensemble de gauche	Ensemble de droite
a	b et c
b	a et c
c	a et b

Ensuite, on prend la question qui maximise le gain.

Dans le cas général, pour n valeurs possibles on a 2^{n-1} -1 questions possibles.

Si on tombe dans un cas où un attribut prend une infinité de valeurs numériques (variable quantitative), on suit le même principe. On pose toutes les questions binaires possible pour chaque attribut quantitatif ($x_i<a$ ou $x_i\leq a$) et on choisit la question qui maximise le gain. Le nombre de questions à poser dépend du nombre de valeurs prises par l'attribut considéré dans l'ensemble d'apprentissage.

Dans le cas d'une variable continue, l'approche la plus utilisée est de trier le jeu d'apprentissage selon la variable courante, de séparer les exemples par classes et ensuite de trouver un seuil ($d>s$ et $d\leq s$) qui convient le mieux pour éclater le plus équitablement le jeu d'apprentissage. Un seuil est déterminé avec le point milieu entre 2 classes différentes.

$$(v_1 + v_2)/2 \quad (13)$$

Dans l'équation 11, v_1 représente la borne supérieure limite entre deux classes et v_2 représente la borne inférieure de la deuxième classe. En général, lorsqu'on analyse un attribut numérique par rapport aux classes, il y a plusieurs seuils de segmentation, on doit choisir le seuil qui favorise la répartition du jeu d'apprentissage.

Exemple : On a un jeu d'apprentissage qui contient un attribut qui est continu. Voici les valeurs de cet attribut et la classe qui lui est associée.

Tableau 10 Exemple d'attributs continues

Valeur	Classe
70	1
72	2
72	1
75	1
80	1
83	2
85	1

Pour cet exemple de la table 10, on a 4 seuils possibles, soient 71, 72,81.5 ou 84.

On doit évaluer la mesure de segmentation Gini en utilisant chacun des seuils et on garde le meilleur résultat de classification.

2.4. Les valeurs manquantes et incohérentes [50] :

Une des manières de traiter les valeurs manquâtes ou incohérentes est de les remplacer par la valeur majoritaire de l'ensemble considéré. On va adopter cette méthode dans cette étude.

Les parties précédentes ont servi à la présentation de CART dans le cas de classification. Mais on a déjà expliqué que CART peut être utilisé pour la régression aussi, c'est pour ça qu'on va expliquer brièvement comment utiliser CART pour la régression.

2.5. Cas de la régression [52] :

Comme déjà mentionné, CART peut être utilisé pour la régression.

Dans ce cas, CART génère l'arbre initial en posant des questions du type $x_i < V$ ou $x_i > V$ avec V un nombre réel. Bien sûr, l'algorithme considère toutes les questions possibles pour

un attribut donné (On pose une question pour chaque valeur prise par l'attribut courant). Pour choisir la question à placer dans l'arbre celle-ci doit minimiser le gain suivant :

Hétérogénéité de subdivision=$R(t_g) + R(t_d)$ (14)

Tel que t_g et t_d sont les deux fils résultant de la division.

$$R(t) = \sum_{e \in t} (y_e - u_t)^2 \quad (15)$$

$$u_t = (\sum_{e \in t} ye) / card(t) \quad (16)$$

avec y_e est la classe lue sur l'ensemble des enregistrements d'une ligne faisant partie du nœud t.

Une fois la question à poser pour chaque attribut est déterminée, celle placée au niveau de la racine est déterminée par le gain minimum de la relation (5).

Le processus est ensuite exécuté de manière récursive pour les deux nœuds fils jusqu'ce que l'arbre soit formé. Une fois arrivé au feuilles, les valeurs qui leurs sont affectées sont les moyennes empiriques u_t de leurs données.

La phase d'élagage se fait sur le même principe que précédemment, on construit une succession d'arbres emboités et on cherche à minimiser l'erreur globale d'apprentissage.

Pour un arbre A donné, on note K le nombre de feuilles de A, la valeur de K exprime la complexité de A. La mesure de qualité de discrimination d'un arbre A s'exprime par le critère :

$$E(A) = \frac{1}{n} \sum_{i=1}^{n} (y_i - A(X_i))^2 \quad (17)$$

Où E(A) est l'erreur globale que commet l'arbre A.

N est le nombre d'exemples d'apprentissage.

y_i est la sortie (la classe) de l'exemple X_i. Et $A(X_i)$ est la sortie calculée pour X_i par A.

La construction de la séquence d'arbres emboîtés repose sur une pénalisation de la complexité de l'arbre :

$$C(A) = E(A) + \gamma k \quad (18)$$

Avec γ un entier positif.

Pour $\gamma = 0$, $A_{max} = A_K$ minimise $C(A)$. En faisant croître γ, l'une des divisions de A_K, celle pour laquelle l'amélioration de D est la plus faible (inférieure à γ), apparaît comme superflue

et les deux feuilles obtenues sont regroupées (élaguées) dans le nœud père qui devient terminal ; A_K devient A_{K-1}.

Le procédé est itéré pour la construction de la séquence emboîtée :

$A_{max} = A_K \supset A_{K-1} \supset \cdots A_1$

Où A_1, le nœud racine, regroupe l'ensemble de l'échantillon.

Une fois la séquence obtenue, on peut par exemple utiliser l'erreur sur un ensemble de validation pour trouver l'arbre optimal.

3. CONCLUSION :

Ce chapitre a servi à expliquer l'algorithme CART avec toutes ses étapes.

Nous avons expliqué l'étape de construction de l'arbre maximal ainsi que la phase d'élagage en prenant en compte tous les types de variables qu'on peut rencontrer.

Dans le chapitre suivant nous aborderons l'explication de la version parallèle que nous avons proposée pour, qui sera adaptée à l'utilisation de la prédiction sur des Big Data, en se basant sur les étapes expliquées en dessus.

CHAPITRE III :

CART POUR LES PROBLEMES BIG DATA

1. INTRODUCTION :

Nous avons déjà abordé le principe de l'algorithme CART dans le chapitre précédant.

La recherche bibliographique nous a permis de savoir qu'il y a des travaux qui ont été réalisé sur l'implémentation d'une version séquentielle de CART. Cependant nous n'avons trouvé aucune documentation concernant l'implémentation parallèle.

Ce chapitre va permettre de présenter l'approche Map-Reduce de CART que nous proposons.

2. NOTRE PROPOSITION : MODELE PARALLELE DE CART :

La version parallèle qu'on va proposer repose sur la parallélisation avec Map-Reduce des calculs effectués par l'algorithme. On distingue comme calculs principaux :

- Le calcul de classe majoritaire au niveau des tests d'arrêt.
- Le calcul du critère de Gini.
- Le calcul du critère d'erreur de chaque nœud dans l'arbre dans la phase d'élagage.
- Le calcul du nombre d'erreurs commises par chaque arbre de la liste d'arbres générés par le processus d'élagage.

De plus des calculs, d'autres traitements nécessitent aussi une parallélisation :

- Les traitements relatifs à chaque attribut :

Etant donné que les attributs d'un ensemble de données sont indépendants les uns des autres, on peut effectuer les différents calculs les concernant en même temps :

- Dans la phase de construction de l'arbre max, on peut générer les questions binaires des attributs en même temps, c'est-à-dire : pendant qu'on génère les questions pour l'attribut 1, les mêmes traitements se font pour les autres attributs, simultanément.
- Dans la phase d'élagage, les gains d'erreurs des attributs peuvent être calculés en même temps aussi.
- La validation dans l'étape d'élagage de la liste d'arbres emboités en utilisant l'ensemble de validation :

De la même manière, les tests sur les arbres se font parallèlement.

Sur la base de ces points, on peut schématiser notre version en ce qui suit :

2.1. Calcul de l'arbre max :

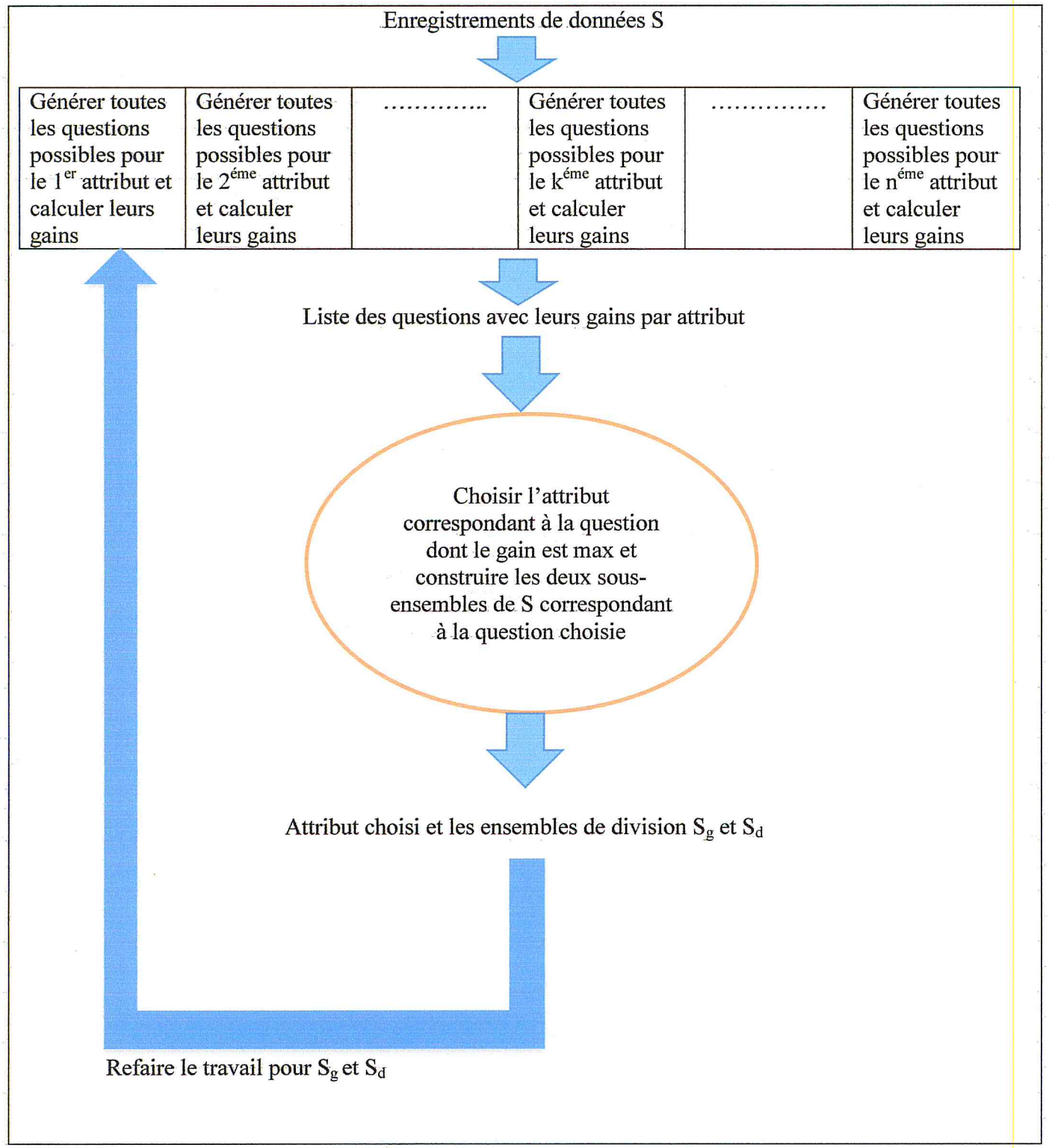
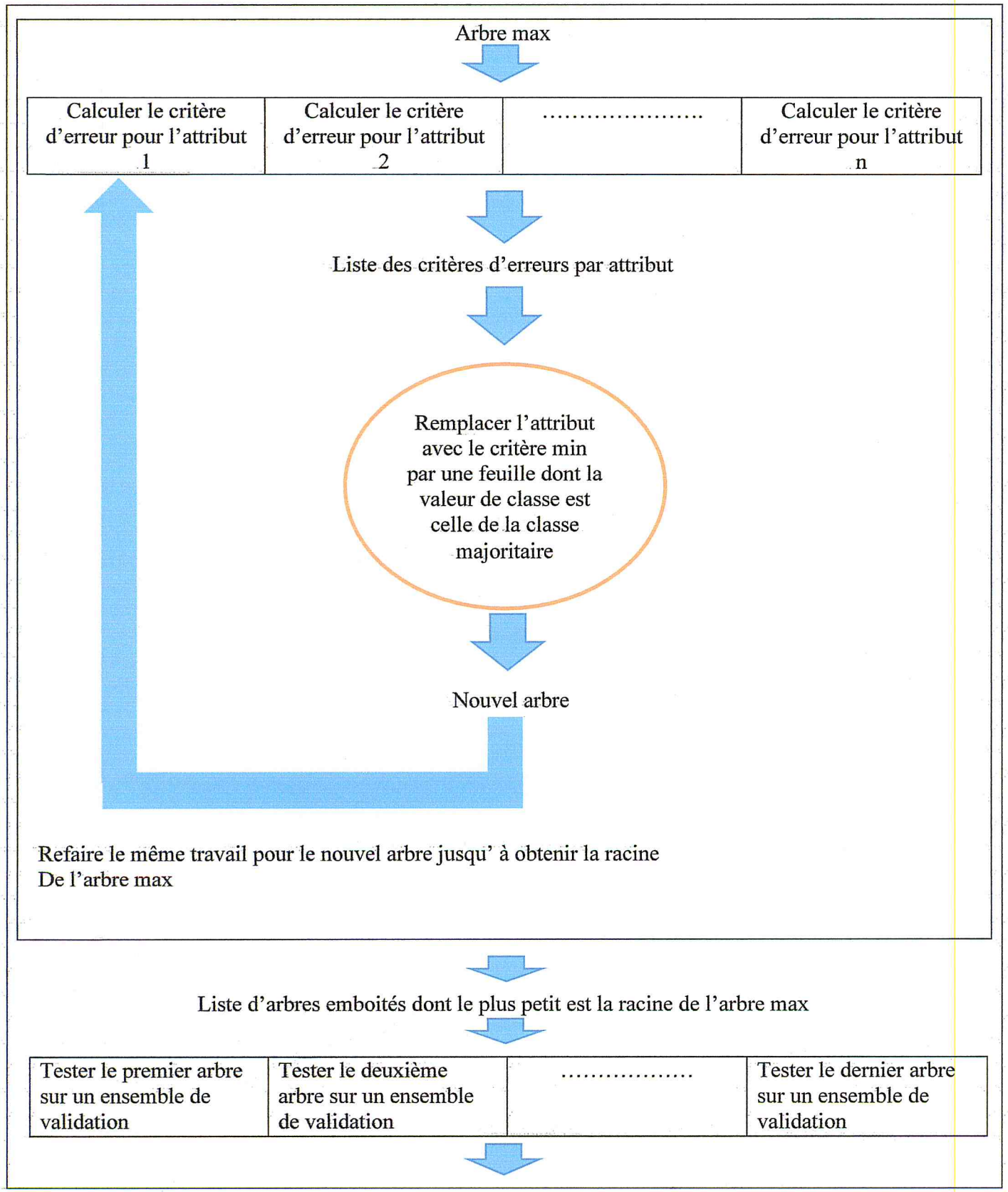


Figure 15 Construction de l'arbre CART max

Les instructions représentées cote à cote s'exécutent en parallèle.

A la fin on aura un arbre maximal.

2.2. Elagage :


Arbre max
Calculer le critère d'erreur pour l'attribut 1
Calculer le critère d'erreur pour l'attribut 2
……..
Calculer le critère d'erreur pour l'attribut n
Liste des critères d'erreurs par attribut
Remplacer l'attribut avec le critère min par une feuille dont la valeur de classe est celle de la classe majoritaire
Nouvel arbre
Refaire le même travail pour le nouvel arbre jusqu' à obtenir la racine De l'arbre max
Liste d'arbres emboités dont le plus petit est la racine de l'arbre max
Tester le premier arbre sur un ensemble de validation
Tester le deuxième arbre sur un ensemble de validation
……
Tester le dernier arbre sur un ensemble de validation

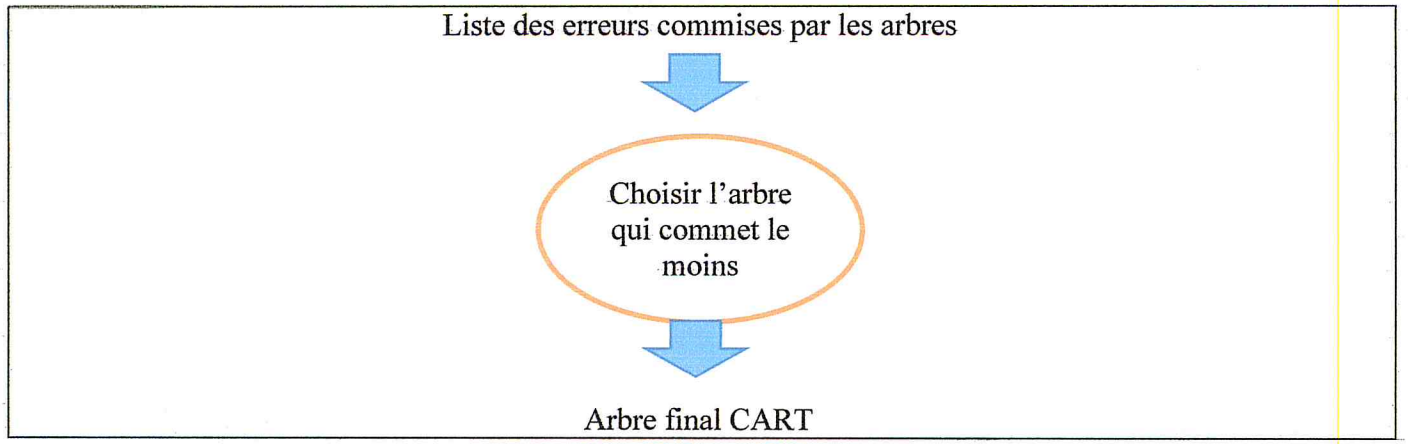


Figure 16 Elagage CART

Nous allons maintenant écrire les parties principales de notre algorithme :

```
Algorithme CART
  Varaibles
    Arbre A ; Chaine_de_caractéres DE ; Résultat Résultats ;Fichier texte S, S';
    Debut
      S = fichier contenant les données d'apprentissage ;
      S'=fichier contenant les données de validation ;
      A = Apprentissage ( S,S') ;
      DE = Lire les données d'entrée ;
      Résultats = Prédire ( A, DE ) ;
      Ecrire(Résultats) ;
    Fin
```

- La Fonction Apprentissage (S, S') :

```
Fonction Arbre Apprentissage (S : Fichier texte d'apprentissage, S': Fichier texte de
validation)
 Variables
   Arbre A ; Attribut Att ; Tableau_de_Chaine_de_caractére Val ;
   Début
     A.racine = Construction_Arbre_Max (S, Att, Val) ;
     Elagage (A, S, S') ;
     Retourner A ;
Fin
```

La fonction Construction_Arbre_Max construit un arbre non élagué à partir d'un fichier de données d'apprentissage S. Mais, la fonction doit toujours savoir quel est la partie de S à partir de laquelle elle doit générer le prochain nœud. Est-ce qu'elle travaille avec tout S ou une partie de S.

Exemple : Dans l'exemple qu'on a traité dans le précédent chapitre, on avait le tableau suivant qui est notre fichier S :

Tableau 11 Exemple du chapitre 2

Condition	Température	Humidité	Vent	Jouer
Soleil	Chaud	Elevée	Faible	Non
Soleil	Chaud	Elevée	Fort	Non
Nuage	Chaud	Elevée	Faible	Oui
Pluie	Doux	Elevée	Faible	Oui
Pluie	Froid	Normale	Faible	Oui
Pluie	Froid	Normale	Fort	Non
Nuage	Froid	Normale	Fort	Oui
Soleil	Doux	Elevée	Faible	Non
Soleil	Froid	Normale	Faible	Oui
Pluie	Doux	Normale	Faible	Oui
Soleil	Doux	Normale	Fort	Oui
Nuage	Doux	Elevée	Fort	Oui
Nuage	Chaud	Normale	Faible	Oui
Pluie	Doux	Elevée	Fort	Non

Le but était de savoir est ce qu'on peut jouer dehors en fonction de certaines conditions climatiques.

Après avoir calculé le gain d'information de S pour chaque question binaire possible à poser, le maximum était pour l'attribut « Condition » avec les sous-ensembles :

→ Sg= Les lignes de S ou Condition = Nuage.

→ Sd = Les lignes de S ou Condition = Pluie ou Soleil.

Le même travail a été refait pour Sg et Sd.

En projetant cet exemple sur notre algorithme on aura ce qui suit :

On applique la fonction Constrcution_Arbre_Max sur S, avec un certain attribut : Att qui est égal à une certaine valeur val, et puisqu' au début on travaille sur tout l'ensemble S alors : Att==null et val==null.

A la fin de cette première étape, la fonction renvoie un nœud possédant la valeur Condition.

Dans une deuxième étape la fonction Construction_Arbre_Max va être appliqué sur Sg et Sd avec comme paramètres d'entrée : Sg, Condition, (Nuage) et Sd, Condition, (Pluie, Soleil).

Donc on aura comme phases d'exécution :

- A.racine = Construction_Arbre_Max (S,null, null).
- A.racine.fils_gauche = Construction_Arbre_Max(Sg, Condition, (Nuage))
- A.racine.fils_droite = Construction_Arbre_Max(Sd, Condition, (Pluie, Soleil)).

Donc, les paramètres Att et Val servent à définir le sous ensemble de S avec lequel la fonction Construction_Arbre_Max travaille dans la prochaine itération.

Avant d'expliquer la fonction Construction_Arbre_max il est primordial d'éclaircir la variable de type « question » qui va être utilisé dans cette fonction.

Une question est un type contenant un tableau de valeurs correspondant aux valeurs sur lesquelles la question est posée.

Exemple :

Dans l'exemple précédant, on a posé la question sur « Condition », sur les lignes ou Condition==Nuage, donc si on déclare une variable q de type question le tableau val de q va contenir « Nuage ».

On va maintenant définir cette fonction plus en détails :

- **Constrcution_Arbre_Max :**

```
Fonction Nœud Construction_Arbre_ Max (S0, Att, val)
 Variables
Fichier_texte : S ; Tableau_dynamique : T_valeurs, Question_gain, Attribut_question ; Nœud : nœud ;
 Début
   • On construit un nouveau document provisoirement S à partir de S0 pour travailler plus à l'aise :
     S=Construction_Document (S0, Att, Val) ;
   • On supprime l'ancien document si ce n'est pas notre document original :
   Si Att ≠ null alors supprimer (S0) ;
   Si S==null alors renvoyer null
   Sinon
    Debut
     // Tester si S correspond à un nœud terminal
     Si taille(S)==10% de N ou la classe majoritaire de S se trouve dans 80% ou plus des lignes de S
     Alors nœud=Générer_Noeud_Terminal(S) ;
     // avec N le nombre de lignes total de tous les enregistrements de données existants
     Sinon
      Début
        • Faire pour chaque attribut simultanément
          Début
        - T_valeurs=Toutes les valeurs possibles de chaque attribut dans S.
        - Générer toutes les questions possibles pour l'attribut en cours
        - Calculer pour chaque question son gain (S,q) et le mettre dans le tableau Question_gain
        - Attribut_question=max(Question_gain) ;
          Fin
      Fin
     nœud=Attribut Att2 dont le gain de question est le plus grand dans le tableau Attribut_question
     Récupérer la question q correspondant au gain max.
     nœud. fils_gauche=Constrcution_Arbre_Max (S, Att2, q.val) ;
     noeud. fils_droite=Constrcution_Arbre_Max (S, Att2, Liste des valeurs prises par Att2-q.val) ;
    Fin
  Retourner (nœud) ;
Fin
```

On va maintenant expliquer les variables et les fonctions principales utilisées dans cette fonction :

1. ***Les variables*** :

- T_valeurs : C'est un tableau contenant toutes les valeurs prises par un attribut donné, il est utilisé dans la génération des questions à poser pour un attribut. Chaque attribut possède son propre tableau
- Question_gain : Tableau contenant des couples (q, g) tel que q est une question et g le gain qu'il lui est associé pour un ensemble S
- Attribut_question : Tableau contenant des couples (Att, q) tel que Att est un attribut donné et q est une question relative à l'attribut Att et dont le gain est maximum.

2. ***Les fonctions*** :

On va s'intéresser à la fonction gain (S,q) qui permet de calculer le gain d'une question au niveau de l'ensemble S et la fonction Générer_Noeud_Terminal (S) qui permet de renvoyer un nœud terminal dont la classe est la classe majoritaire de S :

Fonction réel gain (S : Fichier texte, q : question)
Variables
S_g et S_d : deux fichiers textes.
Début
- On construit deux ensembles S_g et S_d à partir de S correspondant à la division que va provoquer la question q (Un ensemble S_g qui satisfait q et un autre S_d qui ne la satisfait pas.
- Renvoyer (Gini(S)-(taille(S_g)/taille(S)) *Gini(S_g) -(taille(S_d)/taille(S)) *Gini(S_d))

Fin

Fonction réel Gini (S : Fichier texte)
Variables Gini=0 : Réel
Début
//Dans ce cas, on va utiliser le principe de mapreduce. On va définir map () et reduce () comme suit :
- Map () = génère des couples <classe,1>, on lit tous les fragments de S et pour chaque ligne on génère le couple (classe, 1).
- Reduce () =Somme les couples <classe,1> pour renvoyer le nombre d'occurrence de chaque classe lue en entrée.
- Une fois cela fait, on parcourt le fichier de sortie de reduce « f » comme suit :
  Tant que f n'est pas fini
  Faire Début
  - Lire un nombre d'occurrence d'une classe « occ ».
  - Gini=Gini + (occ/taille(S)) *(1-(occ/taille(S))).

  Fin
  Fait
- Renvoyer (Gini).

Fin

Maintenant, la fonction Générer_Noeud_Terminal (S) :

```
Fonction Nœud Générer_Noeud_Terminal (S : Fichier texte)
Variables Nœud nd ; Chaine_de_caractéres : cl ;
Début
  // Là aussi on va utiliser le principe mapreduce :
Map () = génère des couples <classe ,1>
Reduce () = somme des couples <classe,1> pour renvoyer le nombre d'occurrence de chaque classe.
Ensuite, on lit le fichier de sortie et on renvoie dans « cl » la classe dont son nombre d'occurrence est
max.
nd.classe=cl.
Renvoyer nœud
Fin
```

- **Elagage :**

Après la construction de l'arbre max vient la phase du post élagage de l'arbre résultant, la procédure « Elagage (A, S) » prend en paramètre l'arbre retourné dans la phase précédente et un ensemble de données de validation.

Le résultat de son exécution est l'arbre final résultat de l'apprentissage sur les données, qui sera par la suite utilisé pour la prédiction.

On constate deux étapes principales pour cette phase :

- La construction d'un ensemble d'arbres Ti, en remplaçant progressivement un nœud qui minimise le critère de complexité d'erreur par un nœud terminal ayant comme valeur la classe majoritaire, jusqu'à arriver à la racine.
- La validation : pour chacun des arbres Ti construits, dérouler l'ensemble de validation puis, calculer l'erreur commise par chaque arbre. L'arbre qui minimise le plus l'erreur est celui choisi et sera retourné par la fonction « Elagage ».

Cet algorithme éclaircira mieux la fonction :

```
Procédure Elagage (A : arbre, S : ensemble de données d'apprentissage, S': ensemble de
données de validation)
Variables
L : liste d'arbre ; n : nœud ; Cr, Err : Tableaux ; bool : booléan
Début
Bool : vrai ;
// construction de l'ensemble d'arbres L
Tant que (bool)
faire
 Pour chaque nœud interne ni // non feuille
  Faire simultanément
    Cr=Calcul_critère (A, S, ni) ;
  Fait
  Pour le nœud n dont le critère d'erreur dans Cr est le minimum // n=min(Cr)
  Remplacer (A, n, classMaj(n)) ;// On remplace l'arbre au niveau de n par une feuille dont la
classe est majoritaire.
  Ajouter_Liste_Arbre (L, A) ; // On produit une copie de A et on la met dans L.
Si taille (A)==1 alors bool=faux ;
Fait

// validation
Pour chaque arbre construit de L
 Faire simultanément
  Err=Calcul_Erreur (Ti,S') ; //
 Fait
Retourner min_err(Err) // On retourne l'arbre dont l'erreur commise sur l'ensemble de
validation S' est minimum
Fin
```

On va maintenant détailler quelques fonctions :

- Calcul_critère (A, S, ni) :

Cette fonction calcule pour un nœud donné ni (non terminal) d'un arbre T, le critère de complexité de l'erreur produite. Ce critère est calculé comme suit :

Critère (T, S, ni) = (MC (T, S, ni) -MCT (T, S, ni)) / (N (T)* (Nt (T, ni) -1))

Où :

MC (T, S, ni) : est le nombre d'exemples mal classés du jeu d'apprentissage par le nœud ni de l'arbre T quand on fait l'hypothèse qu'il a été transformé en feuille.

Fonction entier MC (T : arbre, S : Fichier d'apprentissage ; ni : nœud)
Variables entier : Max ;
Début
// remplacer ni par la classe majoritaire
Map () =

- On parcourt l'arbre pour chaque ligne dans tous les fragments de S simultanément.
- A la fin de chaque ligne parcourue on génère un couple <c, 1> si le chemin que l'arbre a emprunté pour parcourir la ligne passe par ni. « c » est la classe renvoyée par l'arbre T

Reduce () = regroupe les couples <c, 1> pour renvoyer le nombre d'occurrence de chaque classe générée par l'arbre
Max = Le nombre max de la liste renvoyée par reduce ().
Renvoyer Somme (De tous les p des couples <c, p>) – max.
Fin

MCT (T, S, ni) : est le nombre d'exemples mal classés par les feuilles de l'arbre T situés sous le nœud ni

Fonction entier MCT (T : arbre, S : Fichier d'apprentissage, ni : nœud)
Début
Map () =

- On parcourt l'arbre pour chaque ligne dans tous les fragments de S simultanément.
- A la fin de chaque ligne parcourue on génère un couple < 'erreur', 1> si le chemin que l'arbre a emprunté pour parcourir la ligne passe par ni et si la classe renvoyée par l'arbre n'est pas celle du jeu d'apprentissage.

Reduce () = regroupe les couples <'erreur', 1> pour renvoyer une seul valeur « e » représentant le nombre d'erreurs commises par les feuilles se trouvant en dessous de ni.
Retourner e.
Fin

N (T) et N (T, ni) représentent respectivement le nombre de feuilles de l'arbre T et le nombre de feuilles de T se trouvant en dessous de ni.

- Calcul_Erreur (Ti, S') :

Cette fonction permet de calculer l'erreur commise par chacun des arbres issus de l'étape précédente, pour pouvoir choisir l'arbre optimale. Nous utiliserons un ensemble de validation S' sur chacun des arbres Ti.

Fonction Entier Calcul_erreur (Ti : arbre, S': ensemble de données de validation)
Début
Map () =

- On parcourt l'arbre Ti pour chaque ligne dans tous les fragments de S' simultanément.
- A la fin de chaque ligne parcourue on génère un couple < 'erreur', 1> si la classe renvoyée par l'arbre n'est celle dans le jeu d'apprentissage.

Reduce () = regroupe les couples <'erreur', 1> pour renvoyer une seul valeur « e » représentant le nombre d'erreurs commises par l'arbre Ti.
Retourner e
Fin

3. CONCLUSION :

La version parallèle de CART présentée dans ce chapitre repose principalement sur le modèle de programmation Map-Reduce au niveau des calculs. L'algorithme fait ses différents calculs au même temps sur tous les fragments du fichier sur lequel il travail, de cette manière, les traitements deviennent plus rapides.

Dans le dernier chapitre, nous expliquerons la phase d'implémentation de la version proposée.

CHAPITRE IV :

IMPLEMENTATION ET TESTS

1. INTRODUCTION :

Ce chapitre est consacré à la présentation de notre logiciel dédié à l'implémentation de la version Map-Reduce de CART que nous avons proposé.

On va expliquer l'environnement de développement, les outils que nous avons utilisés, ainsi que notre programme.

On va aussi décrire l'interface utilisateur et les tests que nous avons effectués.

Tous les tests qui vont être expliqués ont été fait sur Intel(R) Core(TM) i3-4005U CPU @ 1.7GHz avec 4Go de RAM.

2. PRESENTATION DU LOGICIEL :

2.1 Environnement de développement :

Nous avons utilisé Eclipse comme plateforme de développement sous le langage java :

Eclipse est un environnement de développement extensible et polyvalent, initié par IBM en 2001 et écrit en java. Conçu autour d'une plateforme commune à laquelle s'agrègent des composants dérivatifs, le projet est ainsi constitué de nombreux sous-projets spécifiques aux technologies sous-jacentes. L'objet de la solution Eclipse est de fournir des outils favorisant la productivité, mais pas seulement celle qui concerne le codage logiciel. On y trouve des environnements de développement intégré mais également de conception, de modélisation, de tests, de reporting, etc. Eclipse a beau être écrit en Java, il peut être utilisé pour développer sous n'importe quel langage de programmation.

Eclipse est publié sous la licence EPL (Eclipse Public License). La fondation Eclipse en est désormais le porteur officiel. **[53]**

Nous avons utilisé Eclipse pour plusieurs raisons **[54]** :

- Il est simple.
- Il est extensible et permet une évolution continue des applications web développées sur sa base grâce à l'ajout des plugins.
- De nombreux tutoriels et documentation du développeur existent pour débuter la programmation avec ce framework. Cela est très important pour pouvoir résoudre n'importe quel problème relatif à Eclipse.
- Il gère mieux les raccourcis que d'autres environnements de développement.

Pour le langage de programmation, nous avons adopté java en raison de sa simplicité et beaucoup d'autres raisons comme **[55]** :

- Java est facile à apprendre. Java a été conçu pour être facile à utiliser et il est donc facile à écrire, compiler, déboguer et apprendre que les autres langages de programmation.
- Java est orienté objet, il permet donc d'écrire des programmes modulaires et un code réutilisable.
- Java est indépendant de la plateforme. L'un des avantages les plus importants de Java est sa capacité de se déplacer facilement d'un système informatique à un autre. La capacité à exécuter le même programme sur de nombreux systèmes différents est essentielle pour le logiciel World Wide Web, et Java succède à cela en étant indépendant de la plateforme à la fois la source et le niveau binaire.
- Java est distribuée. Ce langage a été conçu pour rendre l'informatique répartie facile avec la fonctionnalité de réseau qui est intrinsèquement intégrée. Écriture de programmes de réseau en Java, c'est comme envoyer et recevoir des données vers et à partir d'un fichier.
- Java est sécurisée. Java considère la sécurité dans le cadre de son design. Le langage Java, le compilateur, interprète, et l'environnement d'exécution ont chacun été développés avec la sécurité à l'esprit.
- Java est robuste, il met beaucoup d'accent sur la vérification rapide pour d'éventuelles erreurs, que des compilateurs Java sont en mesure de détecter de nombreux problèmes qui seraient d'abord montrer au cours de délai d'exécution dans d'autres langues.

C'est un langage multi-thread, il permet l'exécution de plusieurs taches au même temps via les threads **[56]**.

2.2 Bibliothèques utilisées :

Une bibliothèque est un ensemble de classes fournies à l'utilisateur pour une utilisation directe sans encombrement avec les détails d'implémentation.

Eclipse fournit à ses utilisateurs la possibilité d'importer des bibliothèques prêtes à l'emploie pour les utiliser dans leurs programmes.

Les bibliothèques que nous avons utilisées sont présentées ci-dessous :

2.2.1. JGraphx :

C'est une bibliothèque qui permet d'afficher des graphes. On l'a utilisé afin de pouvoir afficher et voir la forme finale de notre arbre de décision. **[57]**

2.2.2. Jxl :

Jxl **[58]** est une bibliothèque qui permet la création et la manipulation de fichiers xls que ce soit en lecture ou en écriture. On l'a utilisé pour pouvoir traiter des fichiers d'entrée au format Excel.

2.2.3. Les bibliothèques HADOOP :

Hadoop est un framework libre et open source écrit en Java destiné à faciliter la création d'applications distribuées (au niveau du stockage des données et de leur traitement) et échelonnables (scalables) permettant aux applications de travailler avec des milliers de nœuds et des pétaoctets de données. Ainsi chaque nœud est constitué de machines standard regroupées en grappe. Tous les modules de Hadoop sont conçus dans l'idée fondamentale que les pannes matérielles sont fréquentes et qu'en conséquence elles doivent être gérées automatiquement par le framework.

Hadoop a été inspiré par la publication de Map-Reduce, GoogleFS et BigTable de Google. Hadoop a été créé par Doug Cutting et fait partie des projets de la fondation logicielle Apache depuis 2009.

Le noyau d'Hadoop est constitué d'une partie de stockage : HDFS (Hadoop Distributed File System), et d'une partie de traitement appelée MapReduce. Hadoop fractionne les fichiers en gros blocs et les distribue à travers les nœuds du cluster. Pour traiter les données, Hadoop transfère le code à chaque nœud et chaque nœud traite les données dont il dispose. Cela permet de traiter l'ensemble des données plus rapidement et plus efficacement que dans une architecture super calculatrice plus classique qui repose sur un système de fichiers parallèle où les calculs et les données sont distribués via les réseaux à grande vitesse. **[59]**

Le framework Hadoop de base se compose des modules suivants **[60]** :

- Hadoop Common
- Hadoop Distributed File System (HDFS) : le système de fichiers
- Hadoop YARN
- Hadoop MapReduce

Tableau 12 Architecture Hadoop

Module	Description
Hadoop Common	Contient les bibliothèques et les utilitaires nécessaires aux autres modules Hadoop
HDFS	Le système de gestion de fichiers distribués permet de stocker les données sur les machines du cluster
Hadoop YARN	Une plate-forme chargée de la gestion des ressources informatiques du clusters et de les utiliser pour la planification des applications des utilisateurs
Hadoop Mapreduce	Une implémentation du modèle de programmation MapReduce pour le traitement des données à grande échelle.

HDFS repose dans son fonctionnement sur deux modules essentiels **[59]** :

NameNode :

Nœud de noms, ce composant gère l'espace de noms, l'arborescence du système de fichiers et les métadonnées des fichiers et des répertoires. Il centralise la localisation des blocs de données répartis dans le cluster. Il est unique mais dispose d'une instance secondaire qui gère l'historique des modifications dans le système de fichiers (rôle de backup). Ce NameNode secondaire permet la continuité du fonctionnement du cluster Hadoop en cas de panne du NameNode d'origine.

DataNode :

Nœud de données, ce composant stocke et restitue les blocs de données. Lors du processus de lecture d'un fichier, le NameNode est interrogé pour localiser l'ensemble des blocs de données. Pour chacun d'entre eux, le NameNode renvoie l'adresse du DataNode le plus accessible, c'est-à-dire le DataNode qui dispose de la plus grande bande passante. Les DataNodes communiquent de manière périodique au NameNode la liste des blocs de données qu'ils hébergent. Si certains de ces blocs ne sont pas assez répliqués dans le cluster, l'écriture de ces blocs s'effectue en cascade par copie sur d'autres.

Chaque DataNode sert de bloc de données sur le réseau en utilisant un protocole spécifique au HDFS. Le système de fichiers utilise la couche TCP/IP pour la communication. Les clients utilisent le Remote Procedure Call **[61]** pour communiquer entre eux. Le HDFS stocke les fichiers de grande taille sur plusieurs machines. Il réalise la fiabilité en répliquant les données sur plusieurs hôtes et par conséquent ne nécessite pas de stockage RAID **[62]** sur les hôtes. Avec la valeur par défaut de réplication, les données sont stockées sur trois nœuds : deux sur le même support et l'autre sur un support différent. Les DataNodes peuvent

communiquer entre eux afin de rééquilibrer les données et de garder un niveau de réplication des données élevé.

Le HDFS n'est pas entièrement conforme aux spécifications POSIX **[63]**, en effet les exigences relatives à un système de fichiers POSIX diffèrent des objectifs cibles pour une application Hadoop. Le compromis de ne pas avoir un système de fichiers totalement compatible POSIX permet d'accroître les performances du débit de données.

Le HDFS a récemment amélioré ses capacités de haute disponibilité, ce qui permet désormais au serveur de métadonnées principal d'être basculé manuellement sur une sauvegarde en cas d'échec (le basculement automatique est en cours d'élaboration). Les NameNodes étant le point unique pour le stockage et la gestion des métadonnées, ils peuvent être un goulot d'étranglement pour soutenir un grand nombre de fichiers, notamment lorsque ceux-ci sont de petite taille. En acceptant des espaces de noms multiples desservis par des NameNodes séparés, le HDFS limite ce problème.

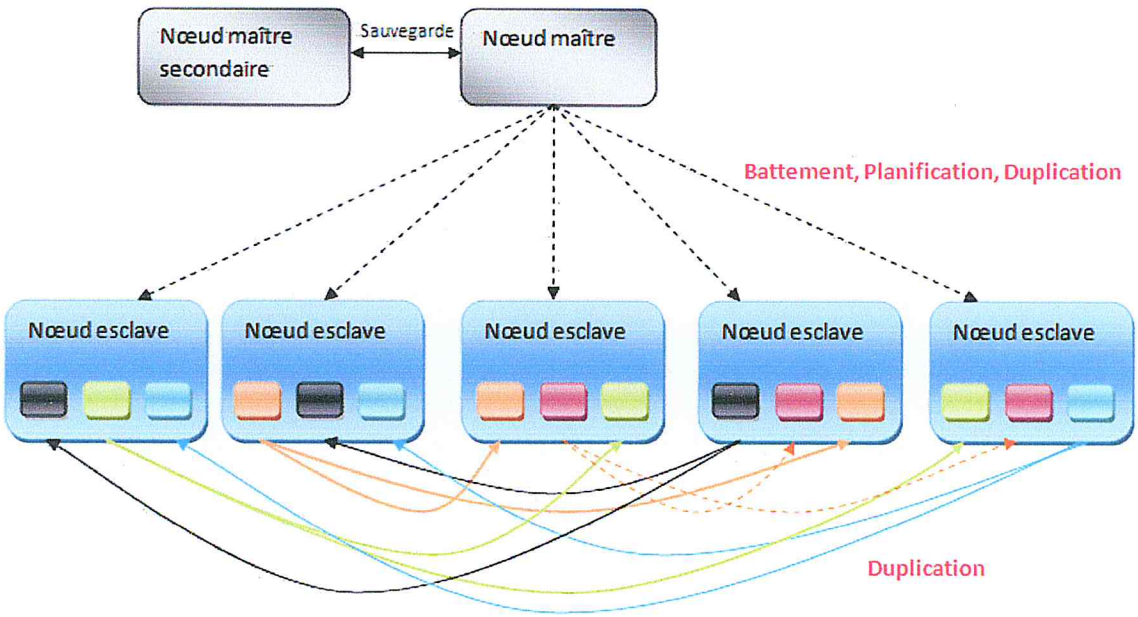


Figure 17 « Schema de principe du HDFS » sur le site « wikipedia.org »

Le terme Hadoop se réfère non seulement aux modules de base ci-dessus, mais aussi à son écosystème et à l'ensemble des logiciels qui viennent s'y connecter comme Apache Pig **[64]**, Apache Hive **[65]**, Apache HBase **[66]** …autres **[67]**.

Au niveau de notre travail, ce n'est pas Hadoop qui exécute les taches mapreduce de notre programme mais Eclipse. Cela a nécessité seulement l'importation d'un plugin (hadoop-eclipse-plugin) dans Eclipse afin que celui-ci devienne capable d'exécuter des jobs mapreduce et l'importation des bibliothèques qui permettent à Eclipse de reconnaitre les

méthodes et les classes mapreudce. Ces bibliothèques ont été importées du fichier d'installation de hadoop.

2.3 Description du logiciel :

Nous donnons dans ce paragraphe une description de notre logiciel en expliquant son principe et en décrivant l'interface utilisateur.

2.3.1. Principe :

Notre programme suit les étapes suivantes :

- Vérifier si le fichier d'apprentissage n'est pas vide.
- Si ce n'est pas le cas, il vérifie l'un des critères d'arrêt expliqués dans le chapitre 2.
- Sinon, il pose toutes les questions binaires possibles pour tous les attributs après avoir lue le fichier d'apprentissage et stocker toutes les valeurs prises par chaque attribut.
- Il calcule ensuite pour chaque question son gain et sauvegarde le couple (question, gain) dans une structure X.
- Il identifie le couple (question, gain) dont le gain est max.
- Il crée un nœud Y ou le nom est le nom de l'attribut dont la question a été choisi, on associe aussi la question du couple choisi au nœud.
- On applique récursivement le processus sur les deux sous-ensembles de données résultant de la question. Les nœuds renvoyés par les deux appels vont être les fils du nœud Y qui sera la racine de l'arbre s'il a été généré lors du premier Appel du processus.

Ensuite, le programme prend l'arbre résultant et lui applique le processus d'élagage :

- Il calcule pour chaque nœud interne son erreur de classement.
- Il remplace le nœud dont l'erreur est minimale par une feuille dont la classe associée est la classe majoritaire des exemples classés par le nœud à remplacer.
- Il crée une copie de l'arbre résultant, il l'a sauvegarde et il refait le même processus jusqu'à atteindre la racine.
- Il lit un deuxième fichier de données qui va servir à la validation.
- Il teste la séquence d'arbres résultants sur l'ensemble de validation et renvoie la racine de l'arbre qui commet le moins d'erreurs.

2.3.2. Interface graphique :

Pour faciliter l'utilisation de l'algorithme que nous avons développé, on a mis en œuvre une interface graphique, simple d'utilisation qui permettra à l'utilisateur de faire des prédictions, d'ajouter ou de mettre à jour des nouvelles données.

Dans cette section, nous donnerons un aperçu sur cette interface :

- La page d'accueil :

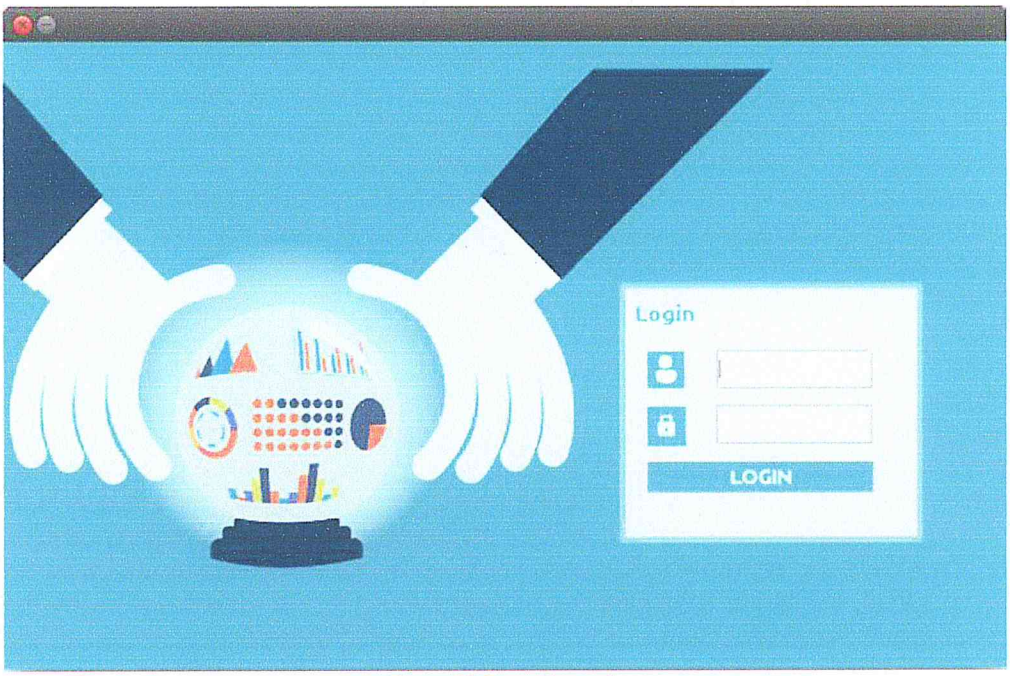


Figure 18 Page d'accueil

Apres l'authentification, deux choix seront présentés : Apprentissage ou prédiction.

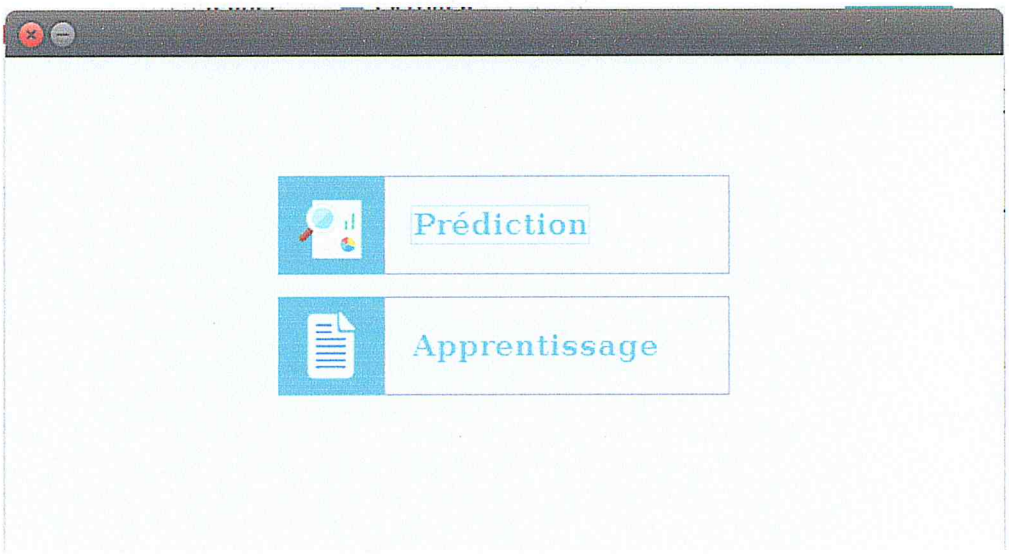


Figure 19 Menu principal

- Apprentissage :

Dans cette partie on peut retrouver les différentes données traitées auparavant, un menu nous permet d'ajouter un nouvel ensemble de données, de mettre à jour un ensemble déjà existant ou de le supprimer.

Pour l'ajout et la mise à jour, l'utilisateur indique le fichier contenant les données d'apprentissage et celle de la validation.

Chaque fois que des données sont ajoutées, l'arbre résultant de l'application de l'algorithme est sauvegardé dans une base de données : le programme est connecté à un serveur de base de données, dans lequel sont stockés les arbres générés par l'algorithme d'apprentissage. De cette manière, les résultats sont utilisables pour les prédictions futures.

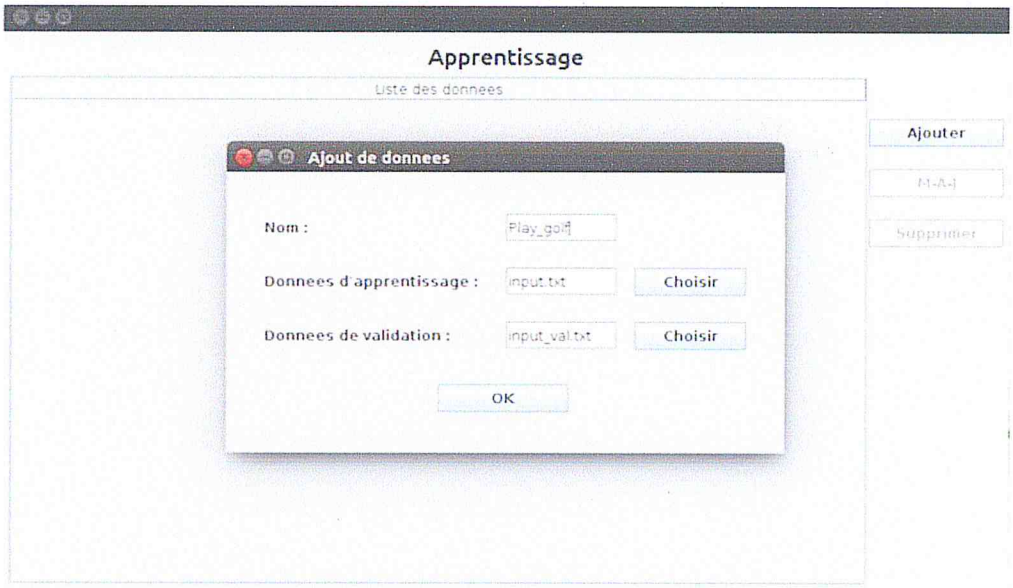


Figure 20 Ajouter des nouvelles données de prédiction

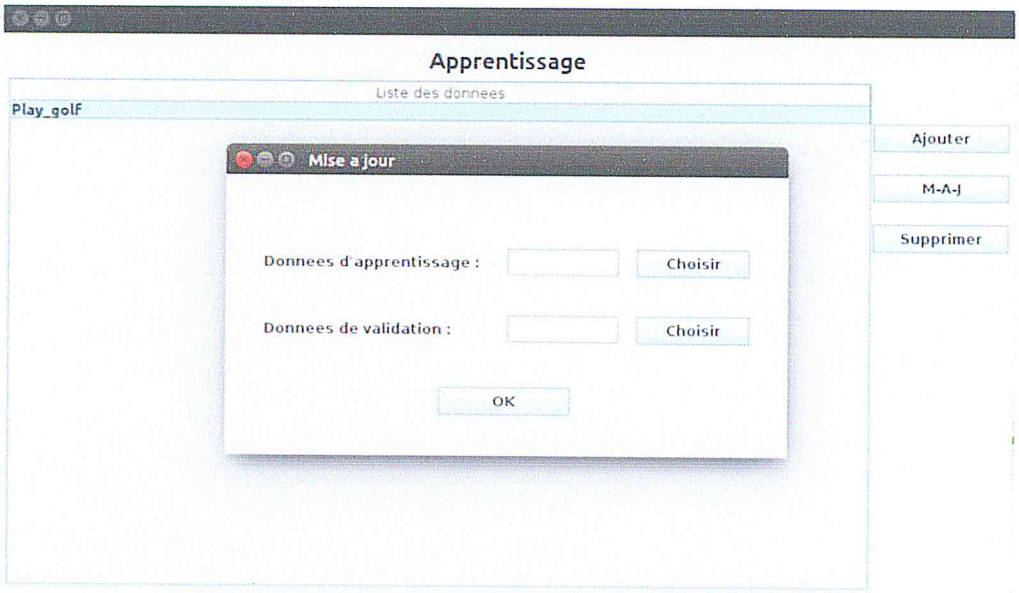


Figure 21 Mettre à jour des données de prédiction

- Prédiction :

La prédiction se base sur les résultats de la phase précédente, encore une fois, l'utilisateur choisi le domaine qui correspond à la prédiction qu'il veut faire. L'arbre stocké dans la base de données est chargé et est utilisé pour calculer le résultat.

L'utilisateur sera dirigé à introduire les valeurs nécessitants la prédiction.

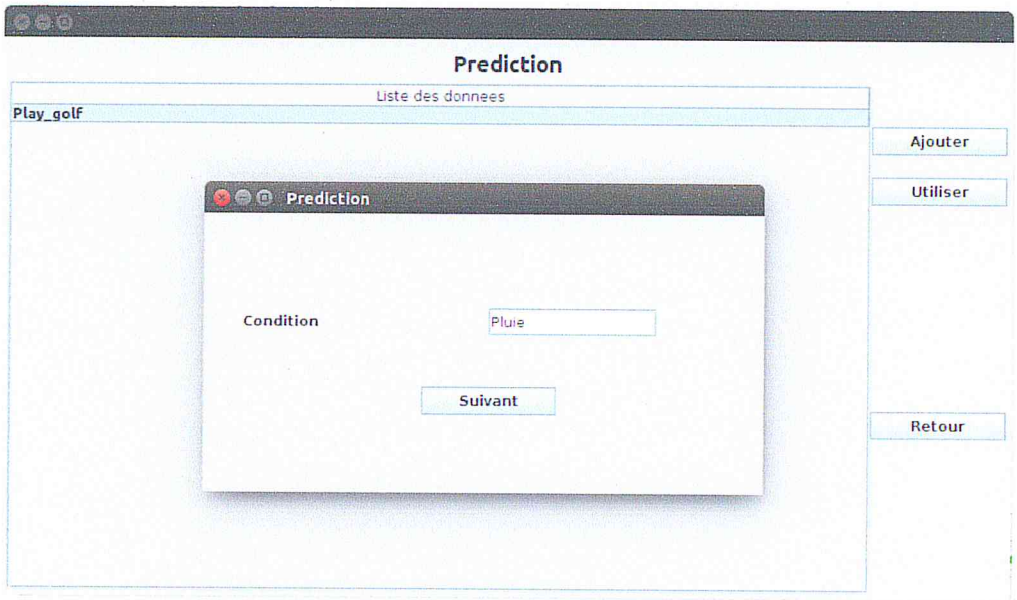


Figure 22 Introduire les valeurs de prédiction

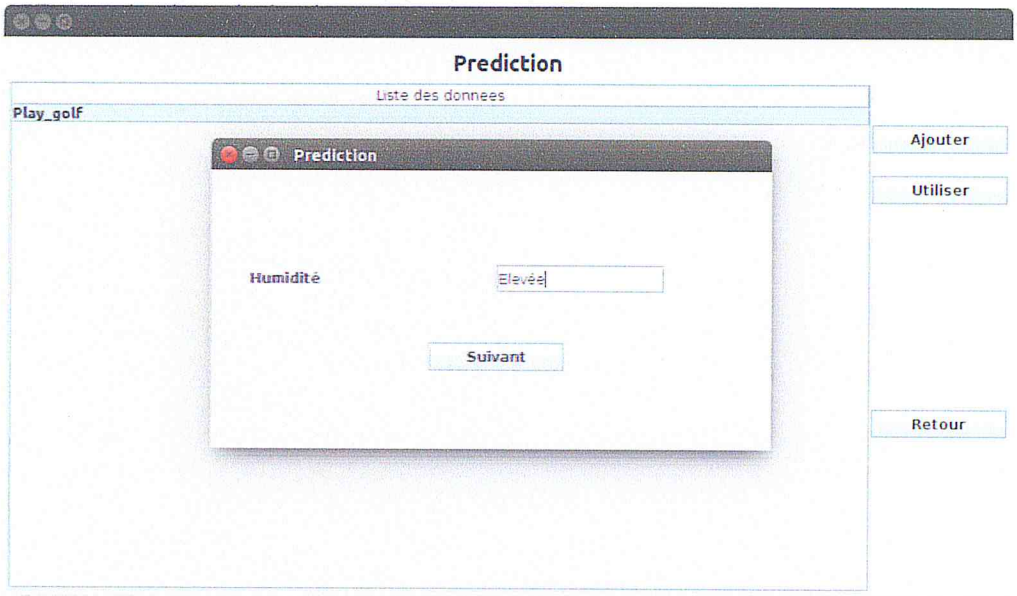


Figure 23 Introduire les données de prédiction

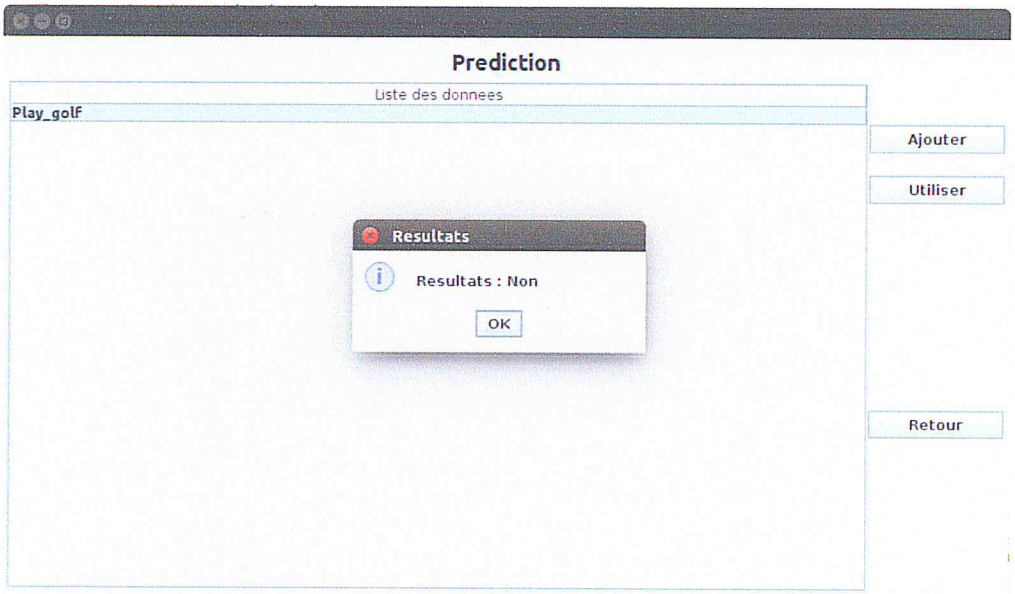


Figure 24 Résultats de la prédiction

Ces opérations sont appliquées autant de fois que nécessaire. L'algorithme étant général, il peut être appliqué plusieurs fois sur des données diverses.

3. RESULTATS ET TESTS :

Nous avons réalisé trois tests :

Les données du premier test **[68]** sont des paramètres de mesure de la qualité d'un véhicule automobile.

Dans cet exemple la qualité d'un véhicule dépend d'un ensemble d'éléments comme : le nombre de places, la sécurité, maintenance …ainsi que d'autres critères cités dans l'exemple.

Donc, notre programme va apprendre à prédire la qualité d'une voiture en fonction des paramètres cités dans l'exemple.

Le deuxième exemple **[69]** est un jeu anonyme dans lequel des joueurs qui sont les attributs effectuent certaines actions qui sont les valeurs des attributs et chaque série d'actions cause leur réussite ou leur échec.

Ici le programme va apprendre les séries d'actions qui permettent de gagner le jeu.

Le troisième exemple **[70]** est également un jeu, il contient toutes les possibilités de dispersion des pièces dans une partie de tic-tac-toe entre deux joueurs.

Le programme va prédire est ce qu'il y a un vainqueur en fonction de la répartition des pièces dans le jeu.

Nous avons divisé chaque ensemble de données dans les trois exemples en trois parties :

- Une partie utilisée pour l'apprentissage.
- Une partie utilisée pour la validation dans l'étape d'élagage afin de choisir l'arbre optimal.
- Une partie dédiée à la prédiction, l'arbre prédit le résultat et on le compare avec celui des données d'entrée. Cette partie est utilisée pour mesurer la performance du classifieur généré par le programme.

Les résultats ainsi que les différentes informations liées aux tests peuvent être résumé dans ce qui suit :

Tableau 13 Résultats des tests

	Nombre de lignes dédiées à l'apprentissage	Nombre de lignes dédiées à la validation	Nombre de lignes dédiées à la prédiction	Nombre d'attributs	Temps d'exécution en minutes	Nombre d'erreurs commis par le classifieur	Taux d'erreur
Test 1	1609	79	71	6	10	10	14.08%
Test 2	2999	115	71	36	55	2	2.8%
Test 3	410	46	71	9	25	22	30.9%

Dans chaque exemple, les attributs prennent de 4 à 6 valeurs différentes.

Pour calculer le nombre d'erreurs commises par le classifieur généré par le programme, on initialise d'abord un compteur d'erreur à 0 (un entier). Ensuite, on compare la sortie du classifieur pour un enregistrement donné avec celle présente sur les données. S'il y a différence on incrémente le compteur d'erreur de 1. Incrémenter le compteur d'erreur signifie que le programme n'a pas prédit correctement le résultat, le résultat prédit n'est pas celui qui a été observé dans l'expérience.

On refait le même travail pour tous les autres enregistrements.

3.1 Interprétation des résultats :

Le tableau nous permet de tirer 4 conclusions importantes :

1. Plus l'ensemble d'apprentissage est grand, plus le classifieur apprend plus et commet moins d'erreur.

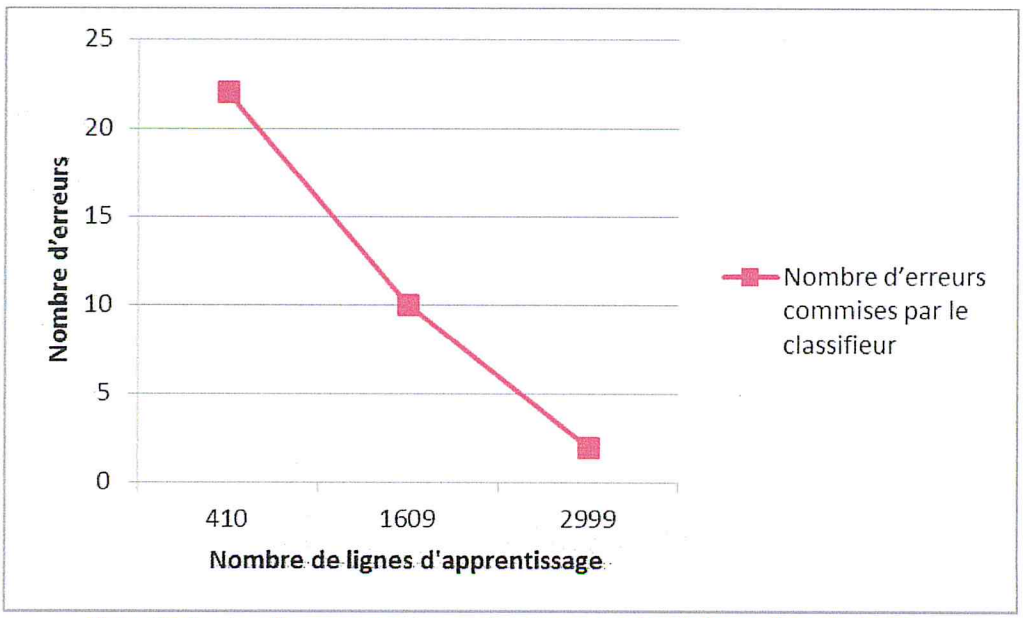


Figure 25 Nombre d'erreurs commises en fonction du nombre de lignes d'apprentissage

2. Plus l'exemple est complexe plus le temps d'exécution est grand.

   Par complexité, on veut dire 3 choses : nombre de lignes d'apprentissage et de validation, nombre d'attributs, nombre de valeurs prises par chaque attribut et difficulté d'apprentissage.

3. Le temps long que prend le programme pour s'exécuter s'explique par le fait qu'on n'a pas parallélisé tous les traitements de CART, on a parallélisé seulement les calculs principaux que fait l'algorithme et cela est due au manque de temps.

   Ainsi, plusieurs traitements importants que fait l'algorithme reste en séquentiel comme : la génération des questions à poser pour chaque attribut, la division des ensembles de données en sous-ensembles, la suppression des fichiers après traitements… C'est ces traitements qui se déroulement en séquentiel qui rendent notre programme un peu lent.

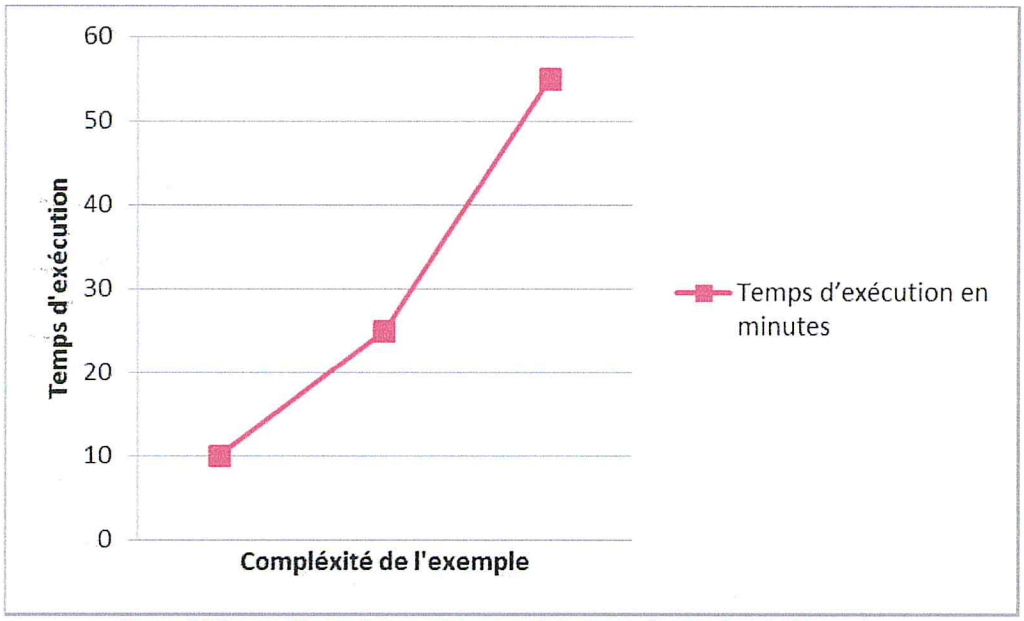


Figure 26 Temps d'exécution en minutes en fonction de la complexité d'exemples

   Comme preuve, on a déroulé l'un des jobs Map-Reduce que nous avons défini sur l'ensemble de 2999 lignes sans faire appels aux autres traitements séquentiels, alors le programme a pris moins d'une seconde pour renvoyer les résultats.

4. Notre programme est performant car à chaque fois, il prédit correct plus de 65% des exemples qu'il teste. Par exemple, pour le premier exemple il a fait 10 erreurs sur 71 exemples testés, donc il a prédit correctement 61 sorties sur 71. Le nombre maximum d'erreurs qu'il a commis c'est dans l'exemple 3, 22 erreurs sur 71 lignes et même ici il a prédit correctement plus de 65% des exemples (49 lignes sur 71).

5. CONCLUSION :

Nous avons présenté dans ce chapitre notre programme, qui peut être utilisé comme un outil d'aide à la décision dans différentes entreprises

Nous avons présenté dans une première partie l'ensemble des outils que nous avons utilisé pour la réalisation de notre projet. Et dans une deuxième partie, des tests qui ont permis de valider notre travail.

CONCLUSION GENERALE

CONCLUSION GENERALE :

Depuis quelques années, l'analyse prédictive connait un essor remarquable à cause du fait qu'elle est l'une des outils permettant l'analyse, l'interprétation et l'utilisation des big data.

L'efficacité de l'analyse prédictive est due essentiellement à la puissance de ses algorithmes qui permettent d'établir des modèles de prédiction infaillibles.

Dans le cas des big data, il est primordial de définir des modèles parallèles de ces algorithmes pour pouvoir les utiliser. Qui dit modèle parallèle, veut dire qu'on doit définir une autre version de l'algorithme considéré, une version dans laquelle certaines instructions de cet algorithme s'exécutent en même temps.

Dans la littérature, plusieurs méthodes de parallélisation d'algorithmes ont été définies, parmi lesquelles on trouve le « mapreduce ».

Notre travail se situait entre ces deux axes : analyse prédictive et mapreduce. Notre but était de prendre un des algorithmes prédictifs et de lui définir un modèle mapreduce afin qu'il devient applicable sur des big data.

L'algorithme en question était CART, un algorithme permettant de construire des arbres de décision binaires.

Du point de vue théorique, on peut dire qu'on a atteint notre objectif, on a défini une version mapreduce opérationnelle de CART, bien que le coté expérimentation reste à approfondir.

Sur la base des résultats des tests effectués, on peut dégager plusieurs perspectives :

On peut paralléliser tous les traitements « parallélisables » du programme et pas seulement les calculs afin d'augmenter la vitesse de traitement de notre programme. Cela peut se faire en définissant des modèles mapreduce pour toutes les autres parties du programme. Ce travail n'a pas été fait à cause du manque de temps.

On peut aussi exécuter le programme dans un environnement hadoop multi-nœud pour la même fin. Dans ce cas, on aura plus de nœuds principaux de calculs distribués sur plusieurs machines. Cela permet de grandement augmenter la vitesse de traitements de notre programme.

En d'autres mots, cette architecture nous permet d'exécuter notre programme sur plusieurs machines au même temps. Aussi, on n'a pas réussi à mettre au point cette architecture à cause du manque de temps et le manque de matériel.

On peut aussi faire la même étude sur un autre algorithme plus performant que CART car l'étude qu'on a menée nous a permis de percevoir un inconvénient majeur dans CART.

CART construit des arbres binaires, or dans plusieurs cas un arbre binaire est insuffisant pour décrire tous les aspects d'études et tous les axes de prédiction d'un problème donné d'où la nécessité de générer des arbres n-ère.

A la fin de ce travail, nous pouvons affirmer que nos objectifs ont été en grande partie atteints : Nous avons appris une nouvelle méthode de programmation, une méthode très efficace qui est le mapreduce et nous avons plongé dans l'océan du Big Data, un domaine très intéressant et un domaine du futur.

Toutefois, nous aurions souhaité terminé notre travail jusqu'au bout pour le tester sur des données réelles du big data.

BIBLIOGRAPHIE

REFERENCES BIBLIOGRAPHIQUES :

[1] Carey, P. (2013, Janvier 4). *FICO moves headquarters to San Jose.* Consulté le Avril 30, 2017, sur Mercury News.

[2] Berry and Linoff, M. J.A, Gordon S. (2000). *Mastering Data Mining: The Art and Science of Customer Relationship Management.* John Wiley & Sons.

[3] Gutierrez, D. D. (2015). *Guide de l'analyse prédictive.* Consulté le Janvier 02, 2017, sur InsideBIGDATA: https://www.celge.fr/wp-content/uploads/2015/12/guide-analyse-pr%C3%A9dictive.pdf

[4] Siegel, E. (2013). *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die.* John Wiley & Sons, p107.

[5] Peersman, G. (2014, Septembre). *Greet Peersman, Note méthodologique,Évaluation d'impact n° 10,Présentation des méthodes de collecte et d'analyse de données dans l'évaluation d'impact.* Consulté le Janvier 02, 2017, sur UNICEF, office of Research-Innocenti: https://www.unicef-irc.org/publications/pdf/brief_10_data_collection_analysis_fre.pdf

[6] Cornillon P. A., Matzner-Løber E. (2010). *Régression avec R.* Springer.

[7] Trevor H., Tibshirani R. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction* (éd. 2e). Springer.

[8] Mitchell, T. (1997). Decision Tree Learning. Dans M. Hill (Éd.), *Machine Learning* (pp. 52-80).

[9] Quinlan, J. R. (1986). Induction of Decision Trees. Dans M. Kaufmann (Éd.), *Machine Learning* (pp. 81-106).

[10] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* (S. M. Morgan Kaufmann, Éd.) San Francisco, CA, USA.

[11] Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data, Journal of Applied Statistics. *Journal of the Royal Statistical Society, 29*(2), 119-127.

[12] Breiman, L. F. (1984). *Classification and Regression Tree.* Pacific California: Breiman, L., Friedman, J.H., Olshen, R., and Stone, C.Wadsworth & Brooks/Cole Advanced Books & Software.

[13] Tufféry, S. (2012). *Data Mining et statistique décisionnelle* (éd. 3e). TECHNIP.

[14] Wei-Yin Loh, Y.-S. S. (1997). Split Selection Methods for Classification Trees. *Statistica Sinica, 7*, 815-840.

[15] Zoonekynd, V. (2009). Use R. *Article and book summaries by Vincent Zooneky*, (p. 5).

[16] Gerstner, A. A. (2004). *Cognitive Navigation Based on Nonuniform Gabor Space Sampling, Unsupervised Growing Networks, and Reinforcement Learning* (Vol. 15).

[17] Rumelhart, D. E. (1986). *Learning Internal Representations by Error Propagation‖ Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*

[18] Cristanini, N. Shawe-Taylor, J. (2000). *Cristanini, NAn Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press.

[19] Vapnik V., Kots S. (1982). *Estimation of Dependences Based on Empirical Data.* Springer Series in Statistics.

[20] Grete Heinz, L. J. (2003). Exploring Relationships in Body Dimensions. *Journal of Statistics Education, 11*(2).

[21] Sanjeev Arora, E. H. (2012). *The Multiplicative Weights Update Method: a Meta Algorithm and Applications.*

[22] Avi Levy, H. R. (2016). *Deterministic Discrepancy Minimization via the Multiplicative Weight Update Method.*

[23] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175-185.

[24] Zhang, H. (2004). FLAIRS. *The Optimality of Naive Bayes.*

[25] Pearl, J. (1985). Conference of the Cognitive Science Society. *Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning*, (pp. 329-334).

[26] Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32.

[27] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* (A. Wesley, Éd.)

[28] Peel, G. J. (2000). *Finite Mixture Models.* New York: Wiley.

[29] Delacour, H. e. (2005). *La Courbe ROC (receiver operating characteristic) : principes et principales applications en biologie clinique* (Vol. 63). Delacour, H., et al. La Courbe ROC (receiver operating characterAnnales de biologie clinique.

[30] T. Hothorn, K. Hornik, A. Zeileis. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. *T. Hothorn, K. Hornik et A. Zeileis, « Unbiased Recursive Journal of Computational and Graphical Statistics, 15*(3), 651-674.

[31] DREYFUS, G. (1998). les réseaux de neurones. Ecole supérieure de physique et de chimie industrielle de la ville de Paris (ESPCI) laboratoire d'électronique.

[32] Breiman, L. F. (1984). *Classification and Regression Tree.* Pacific California: Breiman, L., Friedman, J.H., Olshen, R., and Stone, C.Wadsworth & Brooks/Cole Advanced Books & Software.

[33] Rish, I. (2001). *An empirical study of the naive Bayes classifier.* IJCAI.

[34] E. DIDAY, L. L. (1982). *Éléments d'analyse de données.* DUNOD.

[35] Culler, D. E. (1999). *Parallel Computer Architecture - A Hardware/Software Approach.* Morgan Kaufmann Publishers.

[36] Janette Cardoso, H. C. (1999). *Fuzziness in Petri Nets.* Physica-Verlag.

[37] Knapp, M. (2013). Big Data. *Journal of Electronic Resources in Medical Libraries, 10*(4).

[38] Ohlhorst, F. J. (2012). *Big Data Analytics : Turning Big Data into Big Money.* Somerset, NJ, USA: John Wiley & Sons.

[39] Phillips-Wren, G. &. (s.d.). An analytical journey towards big data. *Journal of Decision Systems*, 87-102.

[40] Laney, D. (2001). *Application Delivery Strategies.* Meta group.

[41] Elgendy, N., & Elragal, A. (2014). Advances in Data Mining. Applications and Theoretical Aspects. (P. Perner, Éd.) *Big Data Analytics, 8557*, 214-227.

[42] Hurwitz, J., Nugent, A., Halper, & Kaufman. (2013). *Big Data for Dummies.* New Jersey: John Wiley & Sons, Inc.

[43] Dean, J., & Ghemawat, S. (2008). Mapreduce : Simplified data processing on large clusters. *51*(1), 107-113.

[44] Dawei, J. (2010). *The performance of MapReduce : An In-Depth Study.* p. 472.

[45] Lee, K.-H., Lee, Y.-J., Choi, H., Chung, Y. D., & Moon, B. (2011). *Parallel Data Processing with MapReduce : A Survey.* p. 11-20.

[46] Kavulya, S., Tan, J., Gandhi, R., & Narasimhan, P. (2009). *An analysis of traces from a production MapReduce cluster.* p. 94-103.

[47] Lee, K.-H., Lee, Y.-J., Choi, H., Chung, Y. D., & Moon, B. (2011). *Parallel Data Processing with MapReduce : A Survey.* p. 13.

[48] Zdnet. (2015). *Big Data : Quels intéréts pour l'analyse prédictive.* Consulté le avril 26, 2017, sur Zdnet: http://www.zdnet.fr/actualites/big-data-quel-interet-pour-l-analyse-predictive-39824666.htm

[49] *Définiitons, méthode et qualité * Indice de Gini.* (s.d.). Consulté le mars 25, 2017, sur Insee: insee.fr

[50] Alain, G. (2007). *Exploration d'un algorithme génétique et d'un arbre de décision à des fins de catégorisation.* Québec: Université de Québec à trois rivières.

[51] Wolfgang, H. (2004). *Classification and Regression Trees (CART) Thoery and applicatoin.* Berlin: Centre d'economie et de statistiques appliquées, Université de Humboldt Berlin.

[52] Tibshirani, R. (1996). *Regression analysis and selection via the Lasso.* Journal Royal Statist.Soc. B, 58, pages 267 à 288, 1996.

[53] *Eclipse.* Consulté le 29/05/2017 sur open source guide : http://www.open-source-guide.com/Solutions/Developpement-et-couches-intermediaires/Outils-de-developpement/Eclipse

[54] Jeff, M., Jean-Michel, L. & Chris, A. Eclipse Rish Client Plateform.

[55] La plupart des avantages significatifs du langage java. Consulté le 29/05/2017 sur nopanda.com.

[56] Laurent, G. (2012). *Techniques audiovisuelles et multimédias - 3e éd : T2 : Systèmes micro-informatiques et réseaux, diffusion, distribution, réception.* Dunod.

[57] Patrick, B. (2012). JGraphX : Les bases. pbriand.developpez.com. Consulté le 29/05/2017.

[58] How to Set Data into Excel sheet using jxl. Consulté le 31/05/2017 sur *Selenium Easy.*

[59] Hadoop Releases. Consulté 30-05-2017 sur Apache.org. Apache Software Foundation.

[60] Vassilina, B. (2016). Hadoop : une plateforme de référence pour faire du Big Data. Consulté le 30/05/2017 sur blog.octo.com.

[61] Bruce Jay Nelson, Remote Procedure Call, PARC CSL-81-9 (Also CMU-CS-81-119), Xerox Palo Alto Research Center, Palo Alto, CA 94306, May 1981, other keywords: IPC, interprocess communication, Emissary, RPC, PhD thesis.

[62] Jakob Østergaard, Emilio Bueso et al., The Software-RAID HOWTO, 6 mars 2010, 1.1.1 éd., 46 p.

[63] Richard,S. The origin of the name POSIX.

[64] Hadoop : Apache Pig. Consulté le 29/05/2017 sur Apache: apache.org.

[65] Hadoop. Consulté le 29/05/2017 sur https://hive.apache.org.

[66] HBase Architecture. Consulté le 29/05/2017 sur whatsbigdata.be.

[67] Hadoop Tutorial. Consulté le 30/50/2017 sur TutorialsPoint.com.

[68] Qingping Tao Ph. D. MAKING EFFICIENT LEARNING ALGORITHMS WITH EXPONENTIALLY MANY FEATURES. Qingping Tao A DISSERTATION Faculty of The Graduate College University of Nebraska In Partial Fulfillment of Requirements. 2004.

[69] Manuel Oliveira. Library Release Form Name of Author: Stanley Robson de Medeiros Oliveira Title of Thesis: Data Transformation For Privacy-Preserving Data Mining Degree: Doctor of Philosophy Year this Degree Granted. University of Alberta Library. 2005.

[70] Esmeir S., Markovitch S. (2004). Lookahead-based algorithms for anytime induction of decision trees. ICML.