République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire Belhadj Bouchaib d'Aïn-Témouchent

Institut des Sciences
Département des Mathématiques et de l'Informatique

Mémoire

Pour l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Ingénierie des Données (RID)

Présenté par :
Anane Othmane et Ras Zin El Abidine

LA CLASSIFICATION DANS UN ENVIRONNEMENT DISTRIBUE

Encadrant :
M. BOUAFIA Zouheyr
Maitre Assistant "A" à C.U.B.B.A.T.

Soutenu en 2018

Devant le jury composé de :

Président :	M. MEDEDJEL Mansour (M.A.A)	C.U.B.B.A.T.
Examinateurs :	M. BENOMAR Mohammed Lamine (M.A.A)	C.U.B.B.A.T.
Encadrant :	M. BOUAFIA Zouheyr (M.A.A)	C.U.B.B.A.T.

Dédicace

Dédicace

Je dédie mon projet de fin d'études à : A mes chers parents.

Aucun hommage ne pourrait être à la hauteur de l'amour dont ils ne cessent de me combler. Que dieu leurs procure bonne santé et longue vie.

À tous mes frères

À mon encadreur Monsieur Z. Bouafia

À mon binôme Zin El Abidine

À mes amis

À toute la famille.

À tous ceux qui ont contribués de près ou de loin pour que ce projet soit possible, je les dis merci.

Othmane

Dédicace

Je dédie mon projet de fin d'études à : A mes chers parents.

Aucun hommage ne pourrait être à la hauteur de l'amour dont ils ne cessent de me combler. Que dieu leurs procure bonne santé et longue vie.

À tous mes frères, Fadila,Mahmoud,Rachid et Ghalima

À mon encadreur Monsieur Z. Bouafia

À mon binôme Othmane

À mes amis

À toute la famille.

À tous ceux qui ont contribués de près ou de loin pour que ce projet soit possible, je les dis merci.

Zin El Abidine

REMERCIEMENTS

REMERCIEMENTS

Nous remercions en premier lieu Dieu le tout puissant de nous avoir accordé la puissance et la volonté pour terminer ce travail.

Ce travail a été réalisé à «Centre Universitaire de Belhadj Bouchaib» à Ain Temouchent.

Nous tenons à exprimer notre reconnaissance à Monsieur Z. Bouafia Maitre Assistant "A" à l'Université d'Ain Temouchent.
Pour avoir dirigé ce mémoire, pour son suivi permanent, ses lectures attentives, ses conseils judicieux et le soutien constant qu'il nous 'a prodigué au cours de ce travail.

Nous remercions Mme. Bouhalouan chef département de math informatique et tout ensemble des enseignants de spécialité M.I remercier pour ses aides, ses encouragements, et de ses grandes qualités humaines.

En fin nous tenons à remercier tous ceux qui de près ou de loin, ont contribué à la réalisation de ce modeste travail.

Othmane et Zin El Abidine

Table des matières

1 Algorithmes de classification bases sur les arbres de décision 4
1.1 Introduction . . . 4
1.2 Data mining . . . 5
1.2.1 Définition . . . 5
1.2.2 Les données et les informations dans le Data Mining . . . 5
1.2.3 Quelques domaines d'application . . . 5
1.3 La classification . . . 6
1.3.1 Définition : . . . 6
1.3.2 Les types des classifications . . . 7
1.4 Arbre de décision . . . 9
1.4.1 Définition . . . 9
1.4.2 Algorithme de base de construction d'un arbre de décision . . . 9
1.4.3 Exemple de génération d'un arbre de décision . . . 10
1.4.4 Exemple de la génération d'un arbre de décision sous Weka . . . 10
1.5 Algorithmes de classification . . . 14
1.5.1 Algorithmes de classification séquentiels . . . 14
1.5.1.1 Algorithme ID3 (Induction of Decision Tree) . . . 14
1.5.1.1.1 Définition . . . 14
1.5.1.1.2 Description . . . 15
1.5.1.1.3 Algorithme de base ID3(Induction of Decision Tree) . . . 15
1.5.1.1.4 Un exemple de fonctionnement de l'algorithme ID3(Induction of Decision Tree) . . . 16
1.5.1.1.5 Limite . . . 16
1.5.1.2 Algorithme C4.5 . . . 16
1.5.1.2.1 Définition . . . 16
1.5.1.2.2 Les améliorations dans l'algorithme C4.5 . . . 17
1.5.1.3 Algorithme C5.0 . . . 17
1.5.1.3.1 Définition . . . 17
1.5.1.3.2 Les améliorations dans l'algorithme C5.0 . . . 17
1.5.1.4 Algorithme CART (Classification And Regression Trees) . . . 18
1.5.1.4.1 Définition . . . 18
1.5.1.4.2 Algorithme de base CART . . . 18
1.5.1.4.3 Avantages . . . 18
1.5.1.4.4 Inconvénients . . . 19
1.5.2 Algorithmes de classification distribuées . . . 19
1.5.2.1 Algorithme SLIQ (Supervised Learning In Quest) . . . 19
1.5.2.1.1 Définition . . . 19
1.5.2.1.2 Limite . . . 19
1.5.2.1.3 Algorithme de base SLIQ(Supervised Learning In Quest) 20

1.5.2.2 Algorithme SPRINT (Scalable Parallelizable Induction Of Decision Trees) . . . 20
1.5.2.2.1 Définition . . . 20
1.5.3 Comparaison entre les algorithmes de classification . . . 21
1.5.3.1 Algorithmes ID3 et C4.5 . . . 21
1.5.3.2 Algorithme C4.5 et C5.0 . . . 23
1.5.3.3 Algorithme C5.0 et CART . . . 23
1.5.3.4 Algorithme SLIQ et SPRINT . . . 23
1.5.3.5 Table comparative des algorithmes des classifications . . . 24
1.6 Conclusion . . . 24

2 Les améliorations apportées par l'algorithme C4.5 26
2.1 Introduction . . . 26
2.2 Algorithme C4.5 . . . 26
2.2.1 Les améliorations : . . . 26
2.2.2 La fonction de gain . . . 27
2.2.2.0.1 La fonction ratio de gain . . . 27
2.2.2.0.2 fonction split d'entropie . . . 27
2.2.3 Les attributs avec des valeurs discrètes . . . 27
2.2.4 Les attributs avec des valeurs inconnues ou manquantes . . . 31
2.2.5 La fonction pour compléter attribut manquant . . . 35
2.2.6 Les attributs avec des valeurs continues . . . 36
2.2.7 Le fonctionnement de l'algorithme C4.5 . . . 38
2.2.7.1 La phase d'expansion . . . 38
2.2.7.1.1 Décider si un noeud est terminal . . . 38
2.2.7.1.2 Choisir un test à associer à un noeud . . . 38
2.2.7.1.3 Affecter une classe à une feuille . . . 38
2.2.7.2 La phase d'élagage . . . 38
2.2.8 L'algorithme de base . . . 39
2.2.9 Les limites . . . 39
2.3 Conclusion . . . 39

3 Technique proposé 41
3.1 Introduction . . . 41
3.2 Le principe général de notre technique . . . 41
3.2.1 La première phase : le chargement d'un fichier textuel . . . 42
3.2.1.1 Les différents formats des fichiers textuels . . . 43
3.2.2 La seconde phase : le partitionnement d'un fichier de données . . . 44
3.2.2.1 Le partitionnement vertical à base d'attributs . . . 44
3.2.2.2 Le partitionnement horizontal à base d'enregistrements . . . 45
3.2.3 La génération des arbres de décision . . . 46
3.3 Évaluation des performances de l'algorithme proposé . . . 47
3.4 Exemples d'implémentation de notre application . . . 49
3.5 Conclusion : . . . 51

Table des figures

1.1 Domaines d'application de Data Mining 6
1.2 Un modèle de classification pour Data Mining 7
1.3 Un modèle de classification supervisée 8
1.4 Un modèle de classification non supervisée 8
1.5 Algorithme de base de construction d'un arbre de décision.[17] 9
1.6 Arbre de décision 10
1.7 Logiciel Weka 12
1.8 Chargement la base des donnée sous Weka 12
1.9 Chargement de l'algorithme J48 sous Weka 13
1.10 Arbre de décision sous Weka 14
1.11 Algorithme ID3 15
1.12 le fonctionnement de l'algorithme ID3 16
1.13 Algorithme CART [5] 18
1.14 Algorithme SLIQ_ARBRE. [6] 20
1.15 Algorithme SLIQ.[6] 20

2.1 La fonction ratio de gain. [1] 27
2.2 La fonction split d'entropie. [1] 27
2.3 Arbre de décision de la fonction XOR 29
2.4 Arbre de décision sous Weka 31
2.5 Arbre de décision de la fonction XOR 33
2.6 Arbre de décision sous Weka 35
2.7 La fonction pour compléter attribue manquante [14] 35
2.8 Algorithme C4.5 [2] 39

3.1 Le chargement d'un fichier textuel 42
3.2 les formats des fichiers textuels 43
3.3 Le partitionnement d'un fichier textuel 44
3.4 Le partitionnement vertical d'une table. 45
3.5 Le partitionnement horizontal d'une table. 45
3.6 Le chargement des données. 46
3.7 La génération des arbres des décisions. 47
3.8 Interface principale de notre application 49
3.9 Interface de la fenêtre de chargement des données 50
3.10 Interface de la fenêtre de Partitionnement des données 50
3.11 Interface de la génération des arbres des décisions par notre application de Data Mining 51

Liste des tableaux

1.1 Les données de weather 11
1.2 Comparaison du temps d'exécution,l'espace mémoire et la précision pour les algorithmes ID3 et C4.5 22
1.3 Table comparative des algorithmes des classifications[10] 24

2.1 la fonction XOR des valeurs A et B. 28
2.2 La base des données "contact-lenses" sous Weka 30
2.3 Ensemble adapté aux données manquantes de la fonction XOR des valeurs A et B 32
2.4 La base des données contact-lenses 34
2.5 état de la machine 36
2.6 état de la machine dans l'ordre croisant 37

3.1 Comparaison du temps d'exécution et l'espace mémoire pour les algorithmes proposé et C4.5 48

Introduction Générale

Introduction Générale

L'évolution des systèmes d'information conduise les entreprises à traiter de plus en plus des données issues des sources toujours plus variées. Les prévisions de taux d'augmentation des volumes des données traitées dépassent les bordures des technologies traditionnelles. On parle de pétaoctet (billiard d'octets) voire de zettaoctet (trilliard d'octets).[8]

Dans ce contexte, est apparu le terme « Big Data », la naissance de ce terme est anglo-saxonne, littéralement « grosses données », est controversée et sa traduction officielle recommandée est « Méga données », même si parfois on parle de données massives. [17]

La fouille de données (Data Mining) est une composante essentielle des technologies Big Data et des techniques d'analyse de données volumineuses. Parmi les objectifs de cette technique est de classifier des données volumineuses dans un temps et un espace mémoire réduit.[23]

L'objectif de ce mémoire est d'étudier le problème de classification basé sur les arbres des décisions. Après une synthèse des algorithmes les plus cités dans la littérature, nous proposons une version parallèle de l'un des meilleurs algorithmes séquentiels : C4.5. Le travail proposé se base sur la stratégie de partitionnement horizontal des données.

À cet effet, ce mémoire est organisé comme suit :

- Le premier chapitre : nous donnons un bref aperçu sur le Data Mining. Ensuite nous présentons un échantillon des algorithmes de classification séquentiels et distribués, en focalisant essentiellement sur l'étape de construction de l'arbre de décision avec un exemple de déroulement. Nous proposons aussi une comparaison entre ces algorithmes.

- Le deuxième chapitre : nous détaillons l'algorithme C4.5 et l'adaptation de la fonction de gain et nous intéressons aux améliorations de cet algorithme dans les deux phases : la phase d'expansion et la phase d'élagage.

- Le dernier chapitre : nous décrivons notre technique qui est une version parallèle de construction de l'arbre de décision en utilisant la stratégie de partitionnement horizontal des données. Ensuite nous montrons quelques exemples de notre application et nous analysant les performances de notre algorithme, ainsi que les gains apportés.

Enfin, nous terminons ce mémoire par une conclusion générale dans laquelle on résume notre travail, qui sera accompagnées des perspectives pour des développements futurs.

Chapitre I

Chapitre 1

Algorithmes de classification bases sur les arbres de décision

1.1 Introduction

L'augmentation du volume des données a fait qu'il est devenu urgent et vital de recourir à des techniques pour extraire des informations à partir de cette masse étendue de données. La fouille de données où le Data Mining est devenu rapidement un thème de recherche suscitant l'intérêt de la communauté scientifique et les entreprises. [21]

Parmi les problèmes des algorithmes des classifications des données sont :

- Temps d'exécution.
- Espace mémoire.
- Complexité.

Ces mesures jouent un rôle important pour déterminer l'efficacité de l'algorithme.

Dans ce chapitre, nous allons donner une idée sur le Data Mining et la classification supervisée par les arbres de décision.

Ensuite, nous allons citer quelques algorithmes de classification basés sur les arbres de décision, en nous concentrant particulièrement sur l'étape de construction de l'arbre de décision. Par exemple :

- ID3 (Induction of Decision Tree).[4]
- C4.5 (successeurs d'ID3).[4]
- C5.0(successeurs d'C4.5).[4]
- CART (Classification And Regression Tree).[4]
- SLIQ (Supervised Learning In Quest).[4]
- SPRINT (Scalable Parallelizable Induction Of Decision Trees).[4]

Et qui peuvent être fournies par le recours à des traitements :

- Séquentiels.
- Distribués.
- Parallèles.

Finalement, nous allons présenter une comparaison entre ces algorithmes, afin de choisir l'un des meilleurs algorithmes séquentiels pour une future amélioration.

1.2 Data mining

1.2.1 Définition

Le terme de « Data Mining » qui peut être traduit par « la fouille de données », « une suite d'extraction de connaissances à partir de données », « l'exploration de données » où « le forage de données », c'est une composante essentielle de la technique Big Data et des techniques d'analyse de données volumineuses.[27]

En général, le terme Data Mining montre le fait de transformer ces données en informations utiles en mettant des relations entre les données. Ces informations peuvent ensuite être utilisées par les entreprises pour réduire des coûts de l'analyse de données. [27]

1.2.2 Les données et les informations dans le Data Mining

Aujourd'hui, les entreprises traites larges quantités des données sous différents formats, dans différentes quantités des données. Parmi ces données, on distingue :

- Les données opérationnelles : telles que les données de ventes et de coûts. [27]
- Les métadonnées : comme les définitions d'un dictionnaire de données. [27]

1.2.3 Quelques domaines d'application

La figure-1.1 suivante illustre quelques domaines d'application courante de Data Mining :

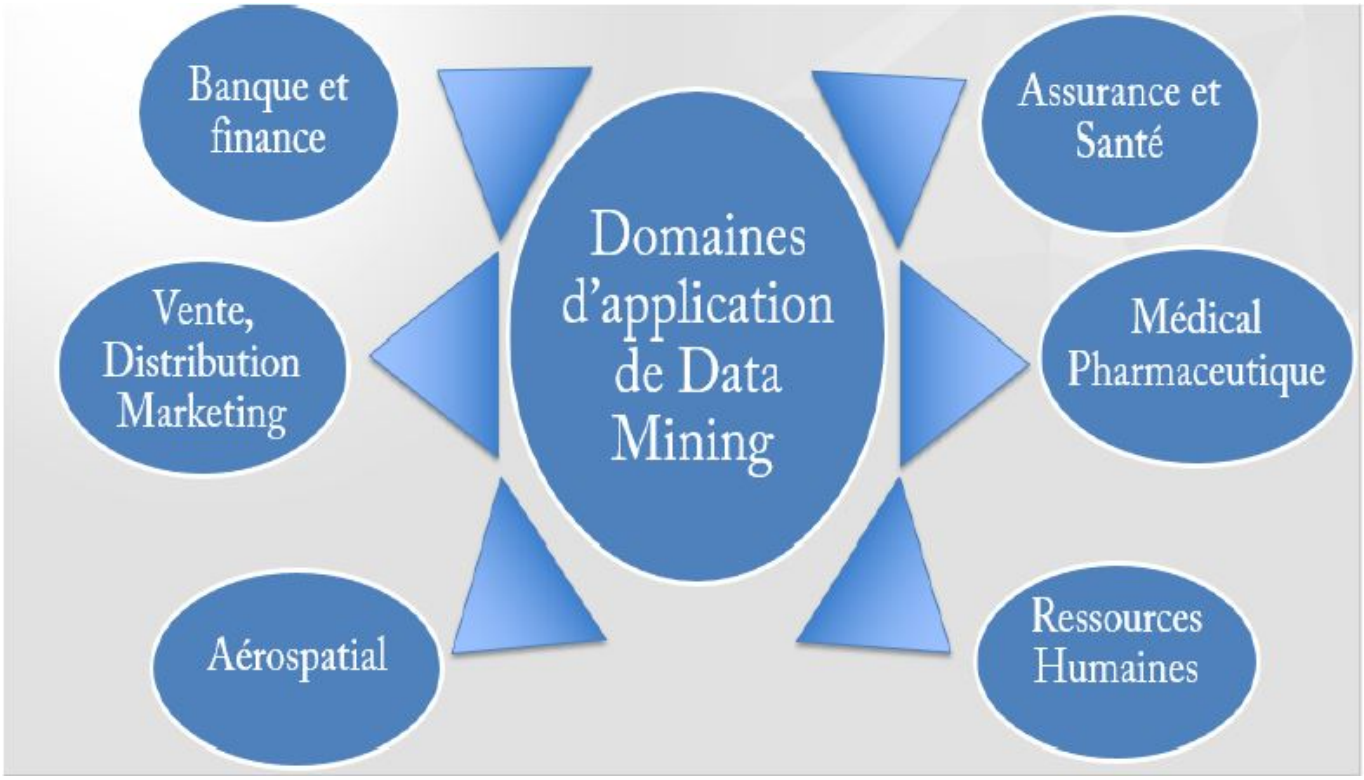


FIGURE 1.1 – Domaines d'application de Data Mining

- Banque et finance : telle que les cartes bancaires des clients. [18]
- Vente, Distribution Marketing : telles que les données de :
  - Le comportement d'achat.[18]
  - La caractéristique clientèle.[18]
- Aérospatial : comme les données relatives à la géographie d'une ville ou pays. [18]
- Assurance et Santé : tels que les données de la demande de remboursements médicaux.[18]
- Médical Pharmaceutique : comme les données de :
  - Dosage dans un traitement.[18]
  - L'apparition d'effets secondaires.[18]
  - La réponse d'un patient à un traitement. [18]
- Ressources Humaines : tel que les données de caractéristiques des employés.[18]

1.3 La classification

1.3.1 Définition :

Le terme « classification » en anglais qui peut être traduit par « l'affectation d'un individu à une classe » dans le cadre de l'analyse discriminante.[13]

En général, C'est une méthode de Data Mining qui permet de :

- Chercher des nouveaux patterns en changeant la façon organisée les données. [13]
- Produire des classes de données identiques entre elles et différentes des données d'une autre classe. [12]

La figure-1.2 suivante illustre un modèle de classification pout Data Mining :

FIGURE 1.2 – Un modèle de classification pour Data Mining

1.3.2 Les types des classifications

On distingue généralement deux types de classification :la classification supervisée et non supervisée.

- **Classification supervisée :**
  - Assigner à de nouvelles données une ou plusieurs classes, par exemple : lui attribuer une étiquette parmi économie, sport, politique, etc... [12]
  - Se base sur des exemples déjà classés dans un ensemble de classes pour déterminer un modèle de classification, par exemple : des articles en rubrique économie, sport, politique, etc... [12]

•Exemple :

La figure-1.3 suivante illustre un modèle de classification supervisée :

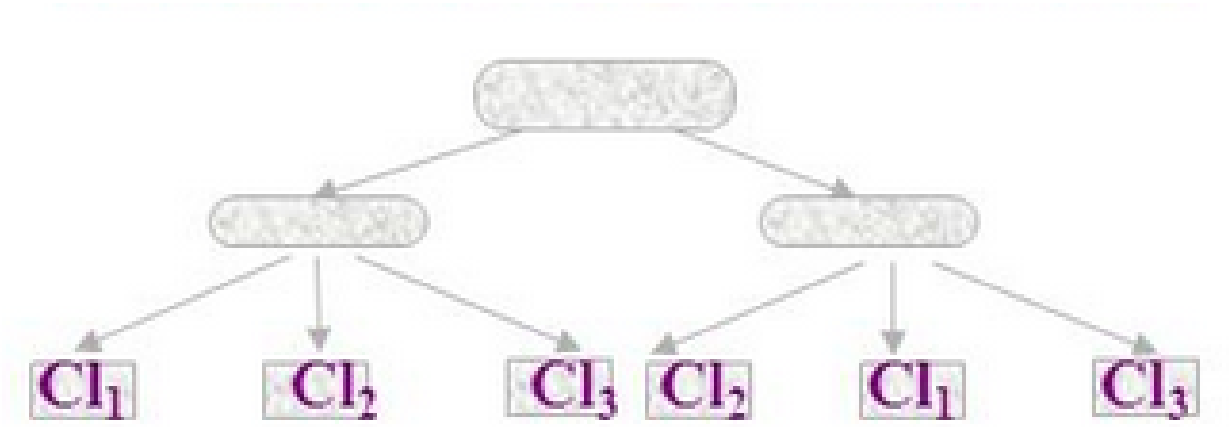
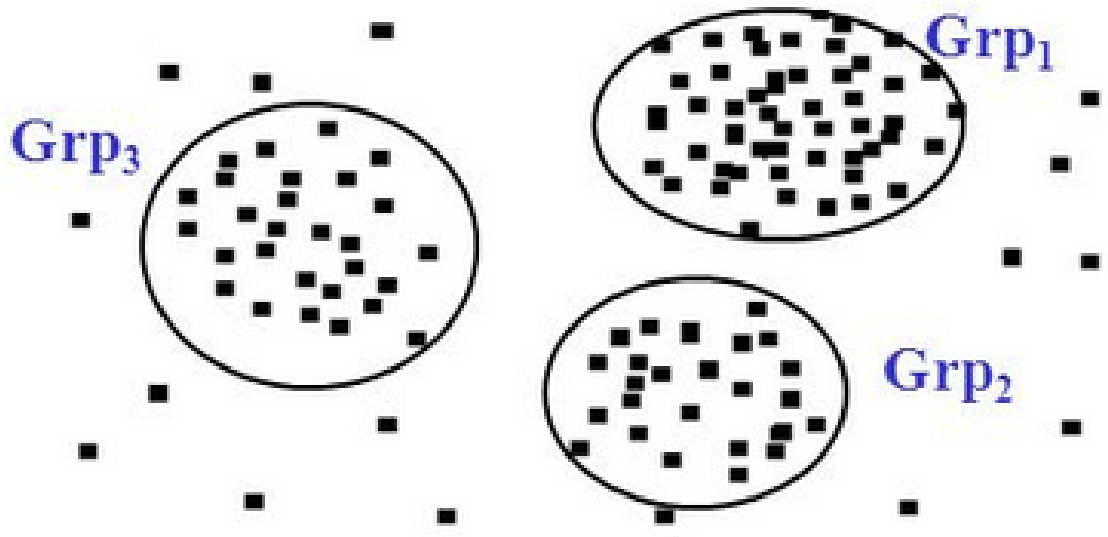


FIGURE 1.3 – Un modèle de classification supervisée

- **Classification non supervisée :**
  - Se base sur des exemples non classés, par exemple : mots d'un texte. [12]
  - On cherche à disjoindre ces données en classes, par exemple : si deux mots ont la même étiquette, ils sont en rapport avec une même thématique. [12]

•Exemple :

La figure-1.4 suivante illustre un modèle de classification non supervisée :



4

FIGURE 1.4 – Un modèle de classification non supervisée

1.4 Arbre de décision

1.4.1 Définition

L'arbre de décision 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. L'arbre de décision est un outil utilisé dans des domaines variés tels que la fouille de données. [13]

- Chaque nœud dans l'arbre de décision correspond à un attribut (un test) non cible. [13]
- Chaque arc dans l'arbre de décision à une valeur possible de cet attribut ou test. [13]
- Une feuille de l'arbre de décision diffusé une estimation espérée de l'attribut cible pour l'enregistrement éprouvé (toute description complète), en Trace le chemin de la racine de l'arbre de décision jusqu' à la feuille (la solution). [13]

1.4.2 Algorithme de base de construction d'un arbre de décision

La figure-1.5 suivante illustre l'algorithme de base de création d'un arbre de décision :

Algorithme: Algorithme AD
Entrées : N : Un nœud courant, R : Un nœud racine, C : Une Classe, A : Un attribut test, S : Un sous arbre.
Sortie : retourne un arbre de décision
N ← R
Répéter
Si N est terminal **Alors**
L'étiqueter N ← C
Sinon
Sélectionner A
Créer S
Passer au noeud suivant non exploré
Jusqu'à obtention d'un arbre

FIGURE 1.5 – Algorithme de base de construction d'un arbre de décision.[17]

1.4.3 Exemple de génération d'un arbre de décision

La figure-1.6 suivante illustre un exemple d'un arbre de décision :

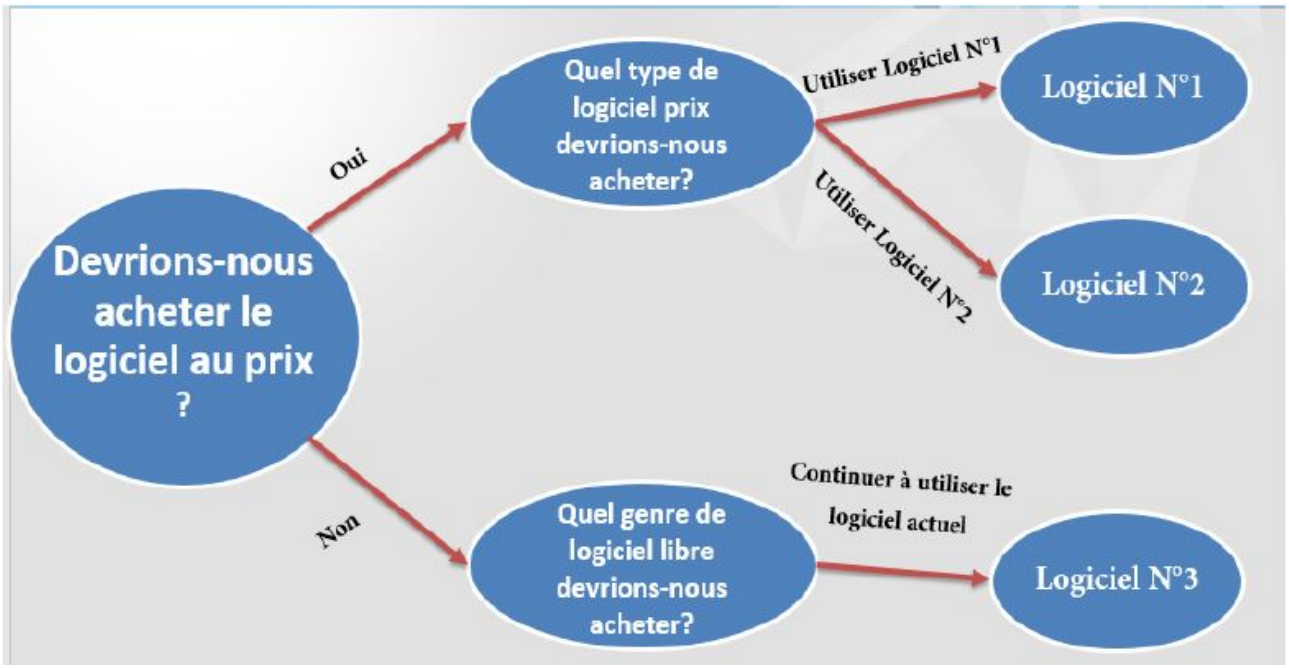


FIGURE 1.6 – Arbre de décision

1.4.4 Exemple de la génération d'un arbre de décision sous Weka

Dans cette partie, nous allons présenter un exemple de génération de l'arbre de décision par l'algorithme J48 (c'est une extension de l'algorithme C4.5 sous Weka)avec une base des données nommée « weather.arff » sous Weka.
Le tableau-1.1 suivant présente les données de weather sous « Weka » :

Weka est un logiciel gratuit dédié spécialement au Data Mining. Parmi les fonctionnalités qu'il couvre, on trouve les arbres de décision. Il permet de modéliser simplement et graphiquement des données. Ce logiciel comprend plusieurs outils dont un API (Aplication Programming Interface) qui permet d'utiliser les outils Weka dans d'autres programmes comme :

- KnowledgeFlow : qui permet de réaliser des analyses plus complexes en modélisant le flux de données au traitement à appliquer.
- Explorer : qui permet d'effectuer des analyses sur les bases des données. C'est ce dernier outil que nous utiliserons dans notre exemple.

Pour la génération d'un arbre de décision de l'algorithme J48(c'est une extension de l'algorithme C4.5) sous Weka, il faut suivre les trois phases suivantes :

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	FALSE	Yes
Sunny	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Mild	High	TRUE	No

TABLE 1.1 – Les données de weather

- **La première phase : lancement du programme « Weka » :**

Dans cette phase, on lance le logiciel Weka à partir du menu démarrer, La figure-1.7 suivante présente l'interface de ce logiciel :

FIGURE 1.7 – Logiciel Weka

- **La deuxième phase : Chargement des données**

Pour charger la base de données « weather » sous format de fichier «. arff », on clique sur « Explorer » Open file », voir figure-1.8 :

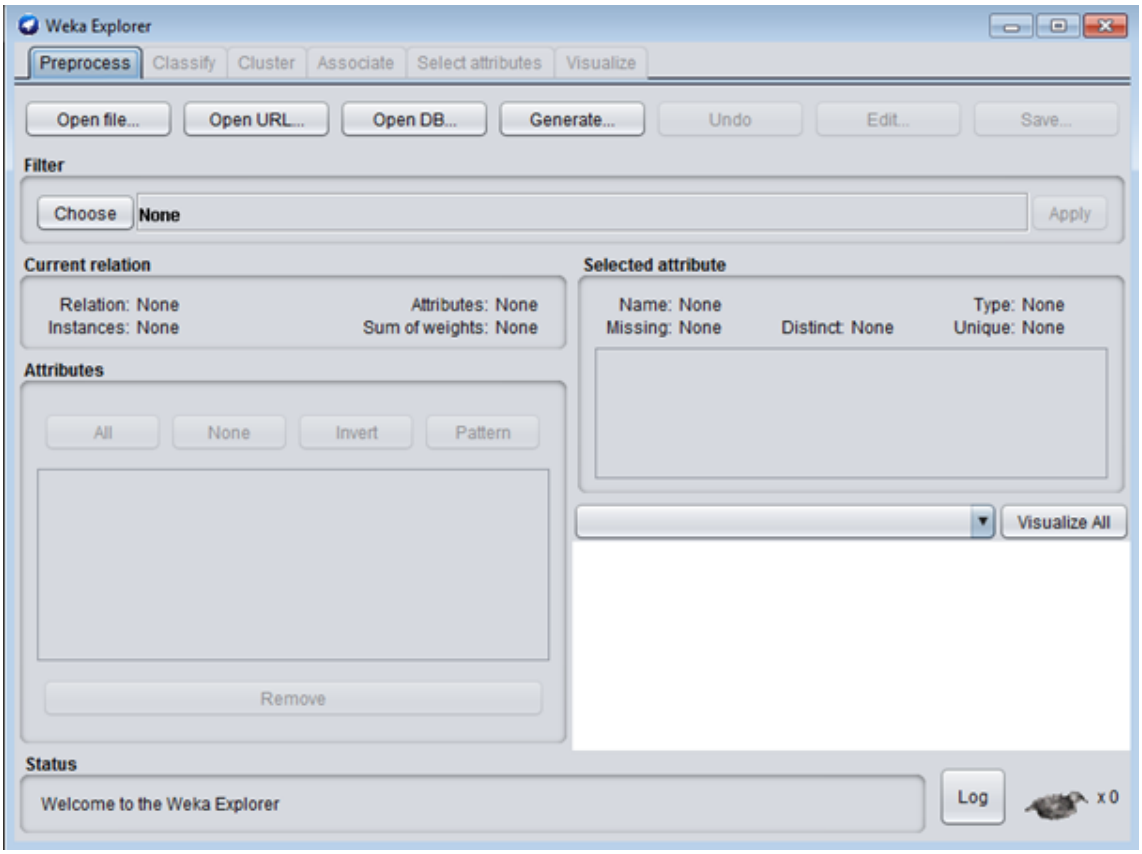


FIGURE 1.8 – Chargement la base des donnée sous Weka

- **La troisième phase : Chargement de l'algorithme J48 et Etablissement de l'arbre de décision**

Pour choisir l'algorithme J48, on clique sur « Choose » Weka » Classifiers » Trees ». Après on clique sur bouton « Start », voir figure-1.9 :

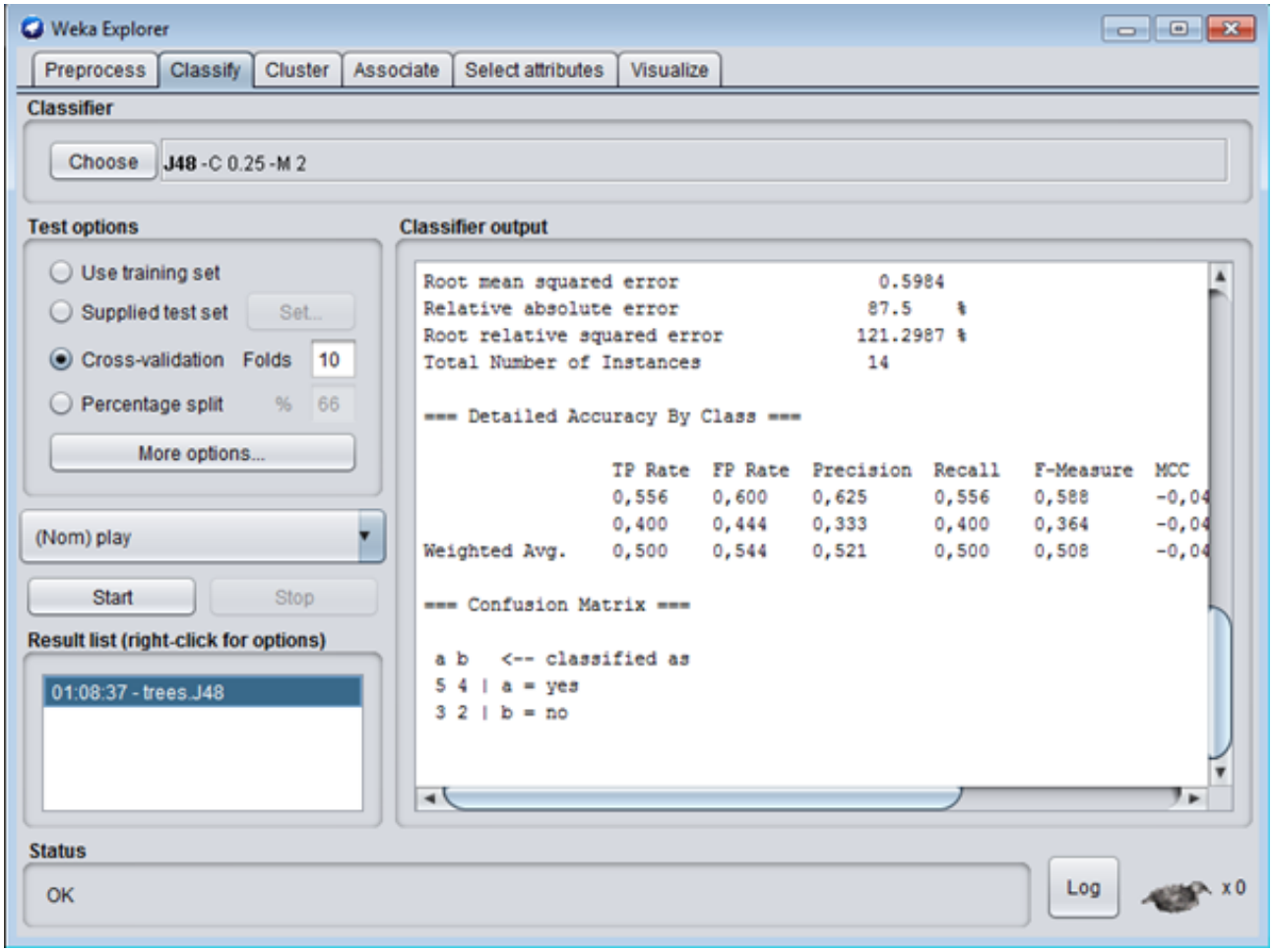


FIGURE 1.9 – Chargement de l'algorithme J48 sous Weka

Pour visualiser l'arbre de décision on clique sur « visualize tree ». La figure-1.10 suivante présente l'arbre de décision de la base « weather » générer par l'algorithme J48 sous Weka.

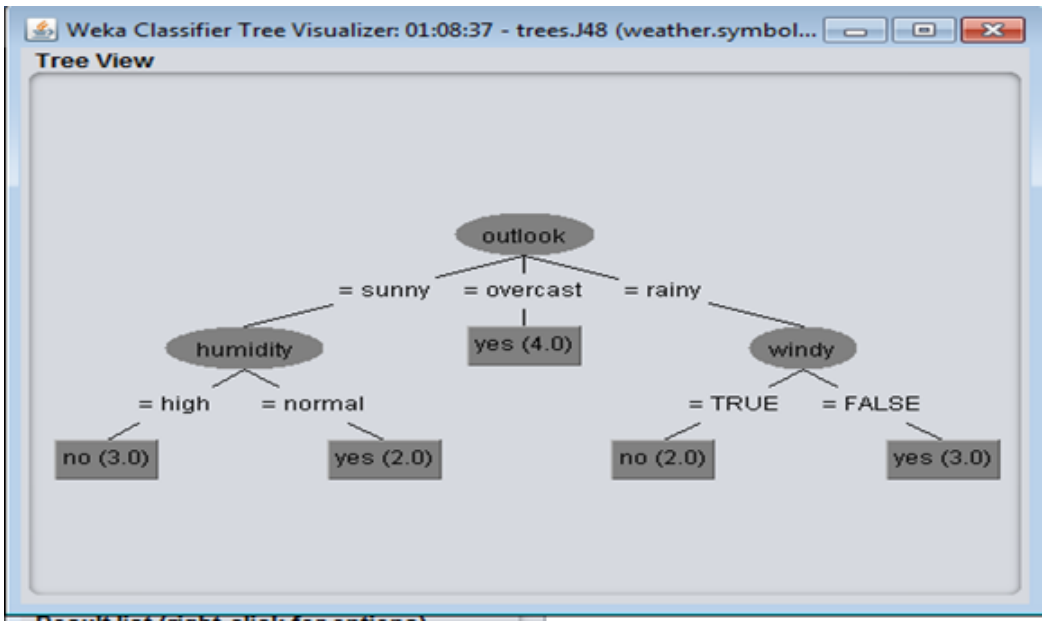


FIGURE 1.10 – Arbre de décision sous Weka

1.5 Algorithmes de classification

Dans cette partie, nous allons présenter deux types d'algorithmes de classification basés sur l'arbre de décision :

- Algorithmes de classification séquentiels.
- Algorithmes de classification distribués.

1.5.1 Algorithmes de classification séquentiels

Dans un premier temps, nous allons présenter quelques algorithmes de classification séquentiels basés sur les arbres de décision, nous concentrons sur l'étape de construction de l'arbre de décision

1.5.1.1 Algorithme ID3 (Induction of Decision Tree)

1.5.1.1.1 Définition

Cet algorithme était proposé par Ross Quinlan en 1979 et publié dans le livre ''Machine Learning" en 1986 dans le but de générer des arbres de décision à partir de données (texte, etc.).

Cet algorithme permet de classifier les données de manière supervisée, afin de génère un arbre de décision. Cet arbre aidera à classer de nouveaux modèles.
[9]

L'objectif principal de l'algorithme ID3 est de définir la variable de segmentation, c'est-à-dire prendre la variable du gain d'information maximum. [9]Donc le principe sur lequel base l'algorithme ID3 est le suivant :

- Dans l'arbre de décision, tout nœud doit être relié l'attribut non cible qui contient plus d'informations par rapport aux autres attributs qui ne sont pas encore utilisés dans le chemin depuis la racine. [20]

1.5.1.1.2 Description

L'algorithme ID3(Induction of Decision Tree) permettait de créer un arbre de décision de manière récursive en sélectionnant l'attribut qui maxime le gain d'information selon l'entropie de Shannon. [20]

L'entropie est servie pour évaluer la quantité d'informations apportées par un nœud, un autre pacage requis pour l'algorithme ID3(Induction of Decision Tree) est que les champs soient tous discrets. Il est donc attachant de discrétiser les champs continus en fonction des valeurs du champ cible avant de lancement de l'algorithme. Généralement l'attribut cible emporte seulement les valeurs vraies et fausses ou échec et succès. [20]

1.5.1.1.3 Algorithme de base ID3(Induction of Decision Tree)

La figure-1.11 montre le fonctionnement de l'algorithme ID3, nous avons un ensemble d'attributs non cibles A1, A2, ..., An, l'attribut cible A, et un ensemble E d'enregistrements d'apprentissage.

Algorithme : Algorithme ID3
Entrées : M : un ensemble d'attributs non cible, A : l'attribut cible, E : données d'apprentissage.
Sortie : retourne un arbre de décision
Début
Initialiser à l'arbre vide ;
Si E est vide **Alors**
Retourner un simple nœud de valeur Echec
Fin Si
Si E est produit uniquement de valeurs identiques pour la cible **Alors**
Retourner un simple nœud de cette valeur
Fin Si
Si M est vide **Alors**
Retourner un simple nœud avec comme une valeur la plus fréquente de l'attribut cible trouvées dans E
Fin Si
D ← l'attribut qui a le plus grand Gain (D, E) parmi tous les attributs de M
{dj avec j = 1, 2, ..., p} ← Les valeurs des attributs de D
{ej avec j = 1, 2, ..., p} ← Les sous-ensembles de E constitués respectivement des enregistrements de valeur dj pour l'attribut D
Retourner un arbre dont la racine est D et les arcs sont étiquetés par d1, d2, ..., dp et allant vers les sous arbres ID3(M-{D}, A, E1), ID3(M-{D}, A, E2), ..., ID3(M-{D}, A, Em)
Fin

FIGURE 1.11 – Algorithme ID3

1.5.1.1.4 Un exemple de fonctionnement de l'algorithme ID3(Induction of Decision Tree)

Dans la figure-1.12 qui présente le fonctionnement de l'algorithme ID3 :

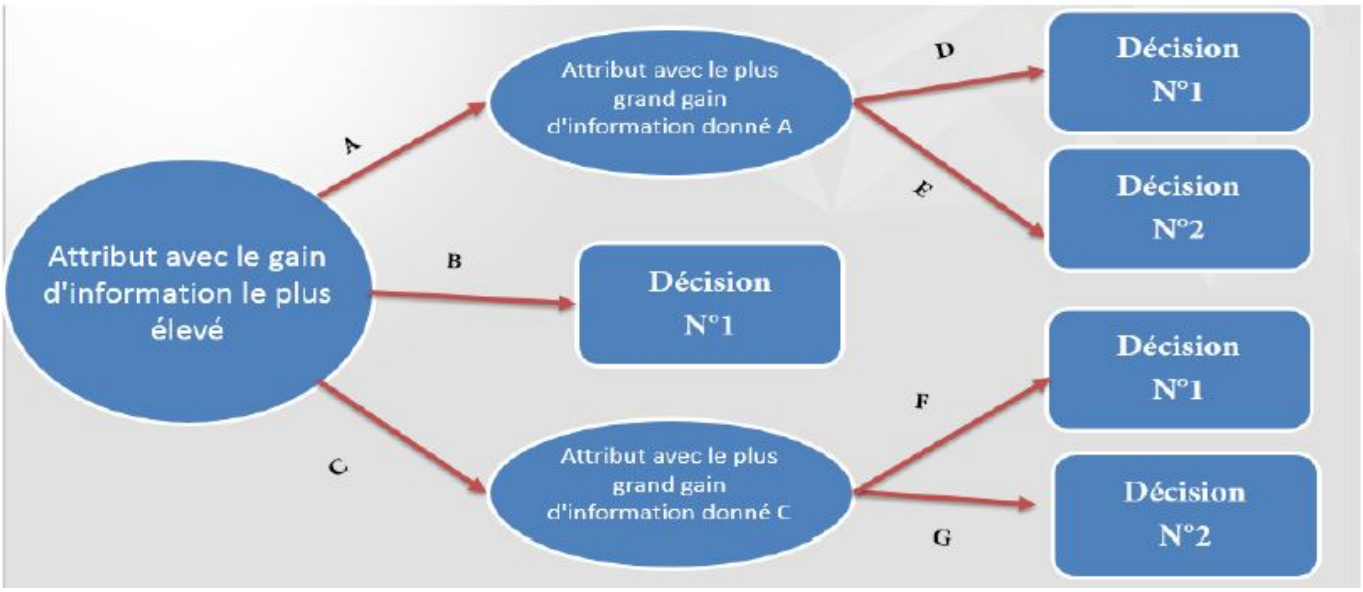


FIGURE 1.12 – le fonctionnement de l'algorithme ID3

1.5.1.1.5 Limite

Algorithme ID3(Induction of Decision Tree) reposes sur quelques problèmes comme :

- N'applique pas une étude totale sur les données, mais permet de faire confiance aux probabilités des attributs qui ont plus de chances d'aiguiller le résultat.[2]
- L'existence du « OU » va déclencher la redondance dans les tests des différents sous arbres.[2]
- Ne peut pas traiter les enregistrements incomplets. [2]

1.5.1.2 Algorithme C4.5

1.5.1.2.1 Définition

Cet algorithme a été présenté en 1993 par Ross Quinlan, c'est une amélioration et couverture des limites de l'algorithme ID3(Induction of Decision Tree) comme vue précédemment. [24]

Nous n'allons pas tout redévelopper pour décrire C4.5 car il repose complètement sur l'algorithme ID3. Alors nous allons concentrer sur les améliorations tenues par C4.5

1.5.1.2.2 Les améliorations dans l'algorithme C4.5

Parmi les améliorations qui sont ajoutées dans l'algorithme C4.5 (extension de l'algorithme ID3(Induction of Decision Tree)) :

- Les attributs de valeur inconnue.[13]
- Les attributs des valeurs discrets. [13]
- L'élagage l'arbre de décision après sa création :
  En remplaçant un sous-arbre entier par une feuille de données, si une règle de décision établit que le taux d'erreur attendu dans le sous-arbre n'est supérieur que celui d'une feuille [13]. L'élagage l'arbre de décision a pour effet :

  - Augmenter l'erreur sur l'ensemble d'apprentissage.[13]
  - Améliorer le pouvoir de la généralisation de l'arbre de décision avec de grosses bases de données. [13]

- Gestion des attributs avec des coûts différents.[13]
- Les attributs à valeur sur intervalle continu.[13]

1.5.1.3 Algorithme C5.0

1.5.1.3.1 Définition

L'algorithme C5.0 ou See5 (C5.0 pour Unix/Linux et See5 pour Windows) a été présenté par Ross Quinlan, en 1997, c'est une version commerciale et vastement améliorée de l'algorithme C4.5, que s'applique dans les bases données volumineuses.[24] [28]

1.5.1.3.2 Les améliorations dans l'algorithme C5.0

Parmi les améliorations qui sont ajoutées dans l'algorithme C5.0 (extension de l'algorithme C4.5 et ID3) sont :

- Des nouveaux types de données qui sont ajoutés [28] [29] comme :
  - Les dates.
  - Les valeurs « sans objet ».
- Amélioration au niveau des arbres de décision.[28] [29]
- Amélioration au niveau de l'efficacité, la mémoire et la vitesse de traitement de données.[28] [29]

1.5.1.4 Algorithme CART (Classification And Regression Trees)

1.5.1.4.1 Définition

L'acronyme CART convient à deux situation bien distinctes qualitative (en anglais classification) ou quantitative (en anglais regression). Cet algorithme développé par Breiman, Friedman, Olshen et Stone (1984) et qui construisent des prédicteurs par des arbres de décision.[5]

L'objectif principal de l'algorithme CART est faire un partitionnement récursif de l'ensemble d'entrée X de façon binaire (1 ou 0) (quand le problème à deux classes), puis d'indiquer une sous partition idéal. [5]

La construction d'un arbre CART se fait en deux étapes :

- La première phase : la construction d'un arbre maximal : c'est-à-dire permet de déterminer l'ensemble de modèles dans laquelle on explorera à choisir le meilleur. [27] [5]
- La seconde phase : élagage un arbre : c'est-à-dire permet de construit une suite de sous arbre optimal élagué de l'arbre maximal. (On a déjà parlé dans l'algorithme C4.5). [5] [27]

1.5.1.4.2 Algorithme de base CART

Dans la figure-1.13 présente le fonctionnement de l'algorithme CART

Algorithme : Algorithme CART
Entrées : Commence l'arbre avec une seule racine, Q : l'index de Gini
Sortie : retourne l'arbre de décision
Début
  Répéter
    Choisir une pointe non-homogène **P** telle que $Q(\mathbf{P}) \neq 1$
    Attacher à **P** deux nœuds filles $\mathbf{P_1}$ et $\mathbf{P_2}$
    Pour toutes les variables $\mathbf{X_j}$ **Faire**
      Trouver le seuil $\mathbf{T_j}$ dans la règle $\mathbf{X_j} < \mathbf{T_j}$ qui minimise $N(\mathbf{P_1})\, Q(\mathbf{P_1}) + N(\mathbf{P_2})\, Q(\mathbf{P_2})$
    Fin Pour
    Trouver la règle $\mathbf{X_j} < \mathbf{T_j}$ qui minimise $N(\mathbf{P_1})\, Q(\mathbf{P_1}) + N(\mathbf{P_2})\, Q(\mathbf{P_2})$ en **j** et définir cette meilleure règle au nœud **P**
  Jusqu'à ce que tous les points **P** soient homogènes ($Q(\mathbf{P}) = 0$)
  Définir les étiquettes de tous les points
  Retourner l'arbre de décision
Fin

FIGURE 1.13 – Algorithme CART [5]

1.5.1.4.3 Avantages

Parmi les avantages existé dans l'algorithme CART :

- Accepte tous les types de variable (continues, discrètes, catégoriques).
- Règles explicitent :

  - Arbre de décision.
  - Règles de décision simples.
  - Modèle facile à programmer pour affecter de nouveaux individus.[5]

1.5.1.4.4 Inconvénients

Parmi les inconvénients existant dans l'algorithme CART :

- N'est pas adaptée pour supporter des données volumineuses. [5]

1.5.2 Algorithmes de classification distribuées

Dans cette partie, nous allons présenter quelques algorithmes des classifications distribuées basés sur les arbres de décision avec un exemple de déroulement

1.5.2.1 Algorithme SLIQ (Supervised Learning In Quest)

1.5.2.1.1 Définition

L'algorithme SLIQ « Supervised Learning In Quest » permet de classifier les données récursivement, elle accepte plusieurs types de variable (continues et nominatives) et elle déploie aussi le principe de longueur de description minimale (MDL) pour l'élagage de l'arbre de décision.[18]

Le principe de cet algorithme récursif se déroule en deux étapes distinctes :

- La première phase : la construction de l'arbre de décision. [13]
- La seconde phase : l'élagage de l'arbre de décision. (déjà expliqué dans algorithme de C4.5) [13]

1.5.2.1.2 Limite

SLIQ arrête la construction d'arbre de décision lorsqu'il termine la construction d'une feuille pure, c'est-à-dire, dans le cas ou les vecteurs d'apprentissage liés à une feuille appartiennent à la même classe. [6]

1.5.2.1.3 Algorithme de base SLIQ(Supervised Learning In Quest)

la figure-1.14 montre le fonctionnement de l'algorithme SLIQ, pour la construction d'arbre de décision

```
Algorithme : Algorithme SLIQ_ARBRE
Entrées : jeu de données  M
Sortie : retourne l'arbre de décision
Début
        SLIQ _ Distribution(M)
 Fin
```

FIGURE 1.14 – Algorithme SLIQ_ARBRE. [6]

La figure-1.15 montre le fonctionnement de l'algorithme SLIQ, pour la distribution des jeux de données :

```
Algorithme : Algorithme SLIQ
Entrées : jeu de données P
Début
        Si tous les points de P trouvent à la même classe Alors
            Retourner
        Fin Si
        Estimer les distributions pour chaque attribut D
        Utiliser le meilleur partitionnement de P résultat sur la création de
                                        P1 et P2
        Distribution(P1)      /* Appel récursif à gauche */
 Fin
```

FIGURE 1.15 – Algorithme SLIQ.[6]

1.5.2.2 Algorithme SPRINT (Scalable Parallelizable Induction Of Decision Trees)

1.5.2.2.1 Définition

Cet algorithme est une amélioration de l'algorithme SLIQ. [10]. Les principales structures de données utilisées dans SPRINT sont les listes d'attributs et les histogrammes de classe. L'algorithme SPRINT réalise un placement de données uniformes et équilibrages de charge en distribuant les listes d'attributs uniformément sur N processeurs.[25]

Cet algorithme avait mêmes principes que l'algorithme SLIQ, mais il y a une différence entre ces algorithmes (SPRINT et SLIQ), comme :

- SLIQ garde la liste entière en mémoire. [25]
- Mais, SPRINT ne garde en mémoire que la partie utilisée des listes triées. [25]

1.5.3 Comparaison entre les algorithmes de classification

Durant tout le chapitre, nous avons présenté de manière générale les algorithmes de classification basés sur les arbres de décision (ID3, C4.5, C5.0, etc..), mais dans cette partie en basant sur des critères pour comparer entre ces algorithmes.

1.5.3.1 Algorithmes ID3 et C4.5

on a parlé dans la première partie de ce chapitre, que l'algorithme C4.5 c'est une amélioration et couverture des limites de l'algorithme ID3(Induction of Decision Tree), comme :

- Permettre d'utiliser des données continues.[7]
- Permettre d'utiliser des valeurs inconnues.[7]
- Permettre d'élagage l'arbre de décision après avoir été créé.[7]
- Permettre de prédiction d'erreur inquiète.[7]

l'algorithme C4.5 plus précis et moins complexes par rapport à l'autre, mais l'algorithme ID3 permet de choisir le meilleur attribut fondé sur la notion d'entropie et le gain d'information pour la croissance de l'arbre de décision.[7]

Pour cette implémentation, on utilise un système d'exploitation « Windows 7 » et avec un processeur Intel de 3ème génération de vitesses 2.54 GHZ et une RAM de 4 GO, en effectuant des tests sur trois bases des données des tailles différentes. Le tableau-1.2 suivant possède le temps d'exécution,la précision et l'utilisation espace mémoire que nous allons calculer après plusieurs itérations :

Taille de jeux de données **(les nombres des enregistrements)**		18	34	65
Algorithme ID3(Second)	Le temps d'exécution	0.28	0.387	0.58
	L'utilisation espace mémoire	1.32	1.47	2.85
	La précision	92.10	82.32	58.48
Algorithme C4.5(Second)	Le temps d'exécution	0.002	0.18	0.42
	L'utilisation espace mémoire	0.5	0.8	1.2
	La précision	94.76	87.21	60.34

TABLE 1.2 – Comparaison du temps d'exécution,l'espace mémoire et la précision pour les algorithmes ID3 et C4.5

Finalement, l'algorithme C4.5 avait la meilleure performance que l'algorithme ID3.

1.5.3.2 Algorithme C4.5 et C5.0

Dans ce chapitre, nous avons parler que l'algorithme C5.0 est vastement amélioré et qui profite des ordinateurs avec plusieurs processeurs par rapport à l'algorithme C4.5.[29]
En général, les points forts de cet algorithme sont :

- La vitesse de traitement.[29]
- L'optimisation d'utilisation de la mémoire.[29]
- L'optimisation pour des bases de données de très grand taille.[29]
- La pondération des cas et erreurs de classement. [29]

Finalement, l'algorithme C5.0 avait la meilleure performance que l'algorithme C4.5 et ID3, mais étant implémenté dans un logiciel commercial, il n'est pas possible d'en avoir le détail. [28] [29]

1.5.3.3 Algorithme C5.0 et CART

Dans ce chapitre, nous avons déjà parlé sur les avantages et les inconvénients de l'algorithme CART.
En général, les différences entre l'algorithme CART et l'algorithme C5.0 sont :

- L'algorithme CART génère un arbre de décision non optimal, au contraire de l'algorithme C5.0.[5] [28]
- L'algorithme CART n'est pas adapté pour exécuter un grand volume d'enregistrements, alors que l'algorithme C5.0 s'exécute dans les bases données volumineuses.[5] [28]
- L'algorithme CART élague les arbres à l'aide d'un modèle composé, alors que l'algorithme C5.0 utilise un algorithme de passage unique.[5]
  [28]

Finalement, l'algorithme C4.5 ont les meilleures performances que l'algorithme CART,

1.5.3.4 Algorithme SLIQ et SPRINT

Nous avons déjà cité dans la première partie de ce chapitre, que l'algorithme SPRINT(Scalable Parallelizable Induction Of Decision Trees) c'est une amélioration de l'algorithme SLIQ(Supervised Learning In Quest) comme :

- Amélioration dans l'utilisation de mémoire
- Facile à rendre parallèle.[25]

Alors l'algorithme SPRINT a meilleure performance que l'algorithme SLIQ.

1.5.3.5 Table comparative des algorithmes des classifications

Le tableau-1.3 suivant possédait la différence entre les algorithmes des classifications des données :

Algorithme	Type d'arbre génère	Type de variable	Critère de segmentation des données
Algorithme ID3	Arbre n-aire	Nominatives	Entropie de Shannon
Algorithme C4.5	Arbre n-aire	Nominatives et Continues	Entropie de Shannon
Algorithme C5.0	Arbre n-aire	Nominatives et Continues	Entropie de Shannon
Algorithme CART	Arbre n-aire	Nominatives et Continues	Indice de Gini
Algorithme SLIQ	Arbre n-aire	Nominatives et Continues	Indice de Gini
Algorithme SPRINT	Arbre n-aire	Nominatives et Continues	Indice de Gini

TABLE 1.3 – Table comparative des algorithmes des classifications[10]

1.6 Conclusion

Dans ce chapitre, nous avons présenté le Data Mining de manière générale, ensuite nous avons détaillé et comparé entre les différents algorithmes de classification séquentielle et distribuée basée sur les arbres de décision.

Les algorithmes C4.5/C5.0 restent les meilleurs algorithmes de classification par rapport aux autres d'algorithmes que nous avons étudiés dans ce chapitre, ils prennent aussi la 1ere place dans le top 10 des algorithmes d'exploration de données. [30] [29]

Néanmoins, l'algorithme C5.0 est implémentée dans un logiciel commercial, il n'est pas possible d'avoir le détail sur cet algorithme.

Dans le prochain chapitre, nous allons voir en détail l'algorithme C4.5 ensuite nous proposons une amélioration de cet algorithme.

Chapitre II

Chapitre 2

Les améliorations apportées par l'algorithme C4.5

2.1 Introduction

Dans ce chapitre, nous allons présenter en détail l'algorithme C4.5, ainsi que ces améliorations à savoir :

- Les attributs avec des valeurs continues.
- L'adaptation de la fonction de gain.

Nous allons définir le fonctionnement de cet algorithme par deux phases essentielles :

- La phase d'expansion.
- La phase d'élagage.

Finalement, pour pouvoir améliorer cet algorithme, nous allons entamer l'algorithme de base de C4.5 et ses limites.

2.2 Algorithme C4.5

2.2.1 Les améliorations :

Parmi les améliorations qui sont ajoutées dans l'algorithme C4.5 sont :

- L'adaptation de la fonction de gain.[13]
- Les attributs avec des valeurs continues.[13]
- Les attributs avec des valeurs discrètes.[13]
- Les attributs avec des valeurs inconnues. [13]

2.2.2 La fonction de gain

Cette fonction se nomme ratio de gain (ou gain ratio), elle est utilisée afin d'éviter de privilégier des attributs par rapport aux autres, Par exemple, si nous avons un attribut "S" qui a une valeur différente pour chaque enregistrement,alors Info(S, A) vaut 0, donc Gain(S, A) sera maximal. Pour compenser cet état de fait,Quinlan suggère d'utiliser le calcul pondéré suivant plutot que le Gain :

$Ratiodegain(S, A) = (Gain(S, A))/(Splitd'Entropie(S, A))$ [15]

avec A1, ..., Am la partition de A induite par la valeur de S.

2.2.2.0.1 La fonction ratio de gain La figure-2.1 suivante illustre la fonction de gain ou ratio de gain :

```
DEF FN Ratio_gain (nom_attribut : entier)
   split ← Split_entropie(nom_attribut)
   gain ← gain_entropie(nom_attribut)
   RETOURNER gain/split Si split ! = 0
FIN Ratio_gain
```

FIGURE 2.1 – La fonction ratio de gain. [1]

2.2.2.0.2 fonction split d'entropie

La figure-2.2 suivante illustre la fonction de split d'entropie :

```
DEF FN Split_entropie(nom_attribut : entier)
   ret ← 0
Pour valeur de 0 à valeurs_possibles_attribut(nom_attribut) Faire
      sous_ensemble ← sous_ensemble_attribut(nom_attribut, valeur)
      ret ← longueur(sous_ensemble) * log2(longueur(sous_ensemble))
   RETOURNER log2 (longueur(nom_attribut)) - ret/ longueur(nom_attribut)
FIN Split_entropie
```

FIGURE 2.2 – La fonction split d'entropie. [1]

2.2.3 Les attributs avec des valeurs discrètes

Pour les attributs discrets possédant un grand nombre de valeurs, nous avons vu que la fonction GainRatio permettait d'éviter de privilégier ces attributs. Il existe, de plus, une option de C4.5 qui permet le regroupement des valeurs. Par exemple, si on dispose d'un attribut A prenant les valeurs a, b, c et d, en standard le test considéré serait 4-aire. Si on active l'option regroupement, seront également considéré des tests de la forme :

- Le test binaire $Aa, betAc, d$. [14] [13]
- Le test ternaire $A = a, A = cetAb, d$. [14] [13]

•Exemple
Le tableau-2.1 suivant qui illustre le résultat de la fonction XOR des valeurs A et B :

A	B	A XOR B
Vrai	Vrai	Faux
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai

TABLE 2.1 – la fonction XOR des valeurs A et B.

•L'arbre de décision
La figure-2.3 suivante illustre l'arbre de décision du tableau 4.1 :

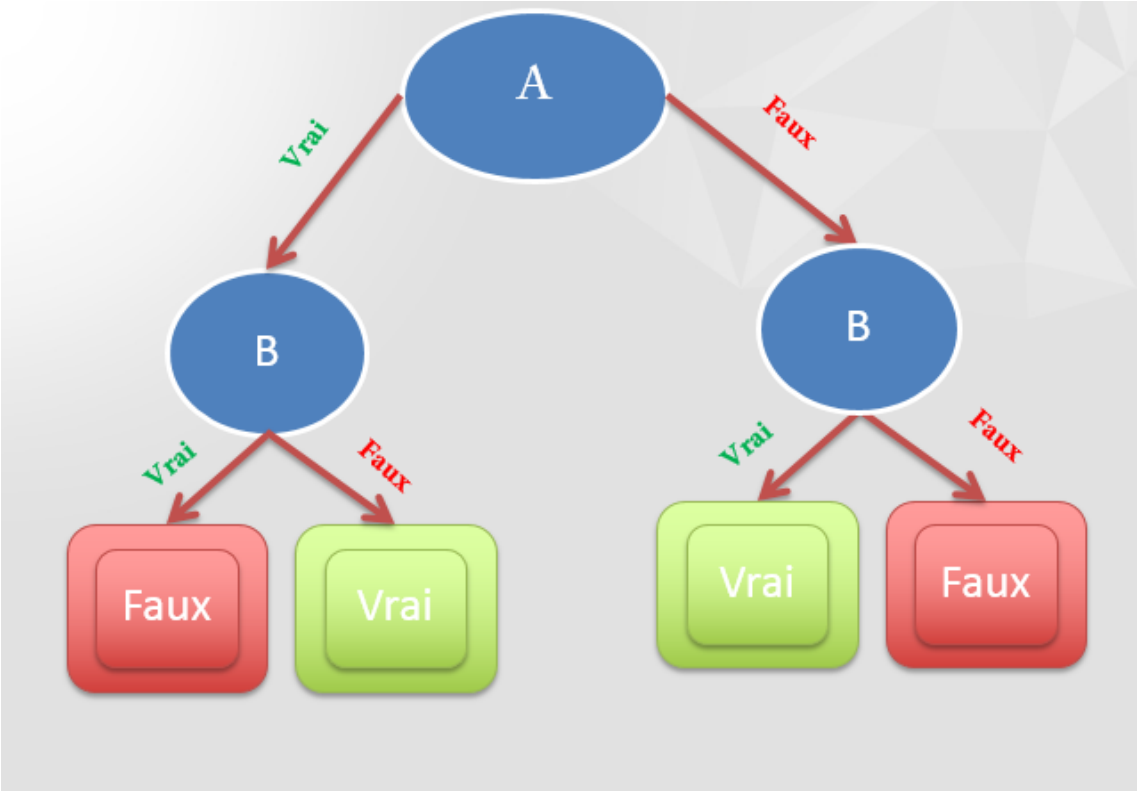


FIGURE 2.3 – Arbre de décision de la fonction XOR

Les lignes du tableau-2.1 précédent sont considérées comme les chemins vers les feuilles dans l'arbre de décision.

•Exemple de la génération d'un arbre de décision sous Weka :

Nous allons présenter un exemple de la génération de l'arbre de décision par l'algorithme J48 (extension de l'algorithme C4.5) sous Weka avec une base des données nommée « contact-lenses.arff » et contient des attributs comme : Outlook, Temperature, Humidity, Windy, Play. Le tableau-2.2 suivant présente les données de contact-lenses :

Outlook	Temperature	Humidity	Windy	Play
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

TABLE 2.2 – La base des données "contact-lenses" sous Weka

La figure-2.4 suivante présente l'arbre de décision de la base « contact-lenses » générer par l'algorithme J48 sous Weka.

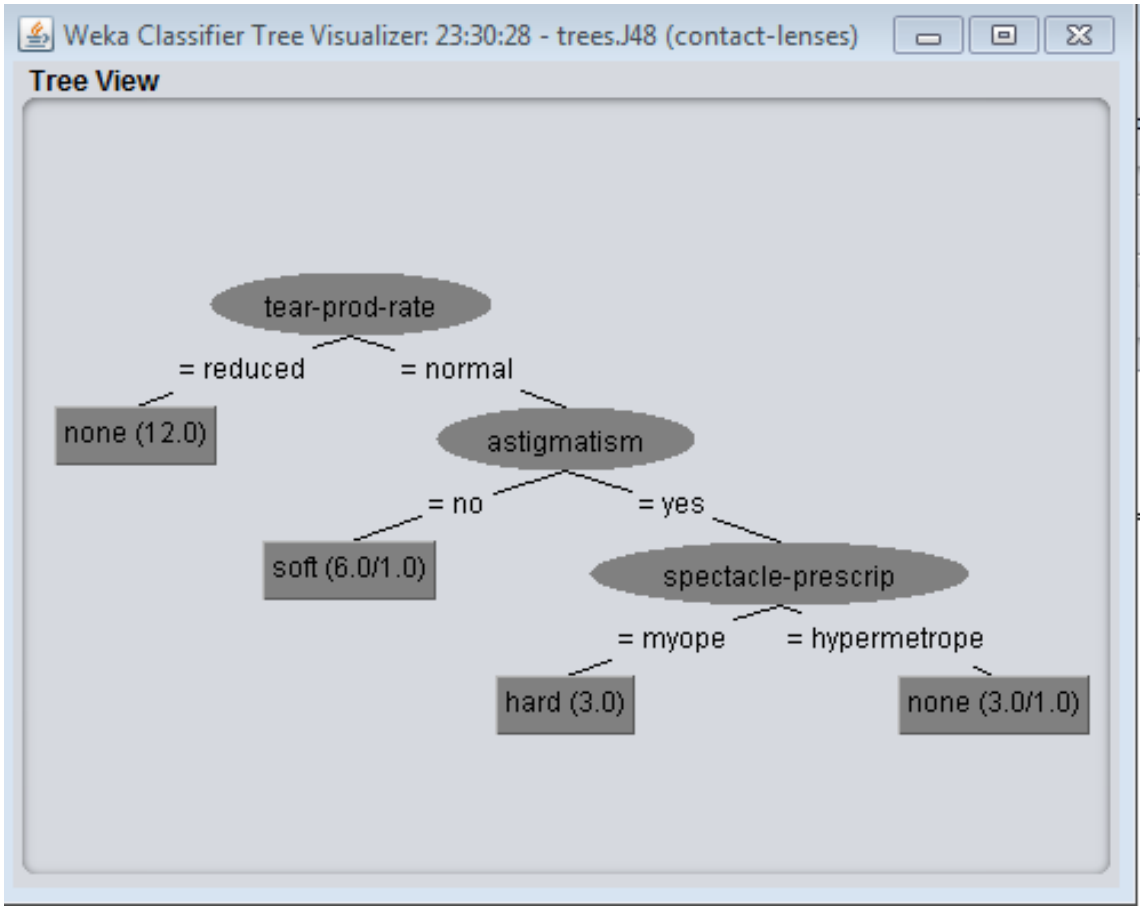


FIGURE 2.4 – Arbre de décision sous Weka

2.2.4 Les attributs avec des valeurs inconnues ou manquantes

il existe des attributs avec les valeurs qui ne sont pas informées comme :

- Si on range la description de produit, il est très possible que toutes les unités ne soient pas disponibles. [14] [13]

Lorsqu'on trouve un test et que la valeur de l'attribut est manquante ou inconnue on considère la branche marginale.[14] [13]

•Exemple :

Le tableau-2.3 suivant c'est la récupération du tableau-2.1 que nous avons donné comme exemple dans « les attributs des valeurs discrètes » dans ce chapitre, mais cette fois-ci, nous allons supprimer volontairement des valeurs dans le tableau de façon que la fonction XOR serait illisible :

A	B	A XOR B
Vrai	Vrai	Faux
Faux	?	?
Faux	vrai	Vrai
Vrai	?	?

TABLE 2.3 – Ensemble adapté aux données manquantes de la fonction XOR des valeurs A et B

Pour résoudre le problème, il existe deux méthodes possibles :

- La première méthode : soit nous supprimons la valeur manquante de cet exemple.[14] [13]
- La seconde méthode : soit nous essayons d'assigner une valeur à la valeur manquante.[14] [13]

La première méthode est la plus sûre dans le sens où on ne tient pas le risque de créer un exemple qui n'existe pas, mais l'algorithme C4.5 nous offre la possibilité de décider une valeur à cet attribut manquant de cet exemple. Ici nous n'avons que 2 valeurs possibles par attribut (en ne considérant que les attributs discrets « Faux ou Vrai »).
[14] [13]

- **Arbre de décision :**

  Si on supprime les valeurs manquantes du tableau-2.3, la génération de l'arbre de décision est suivante :

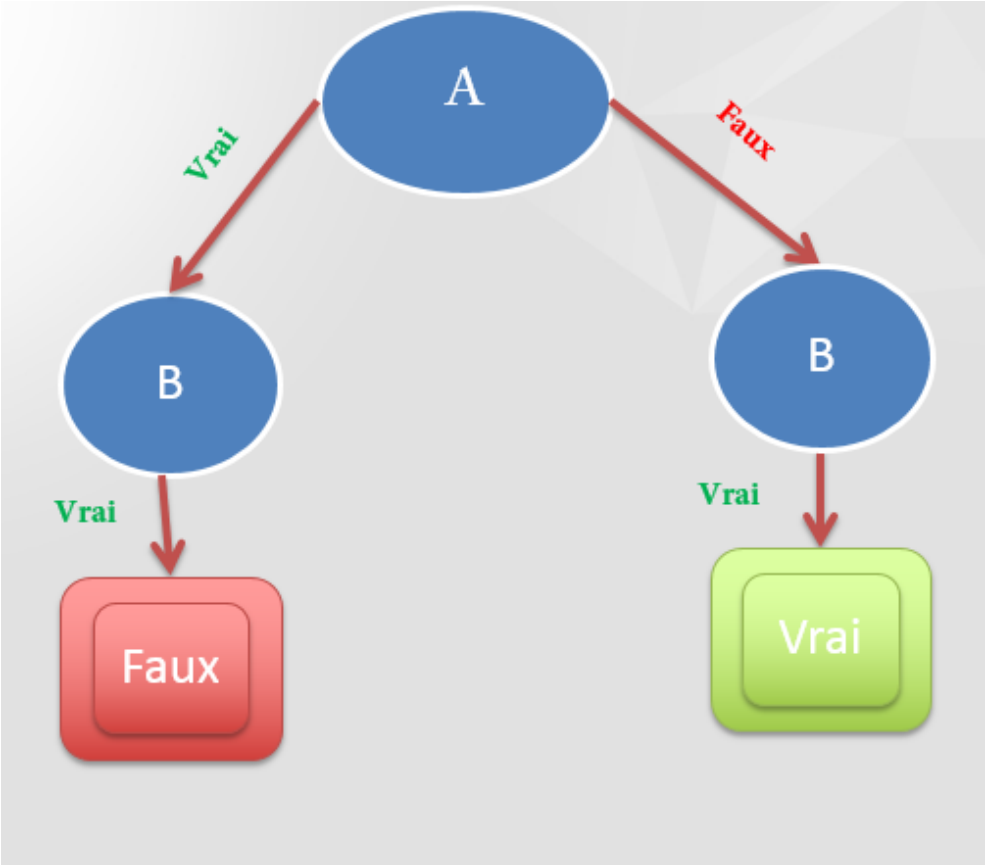


FIGURE 2.5 – Arbre de décision de la fonction XOR

•Exemple avec Weka :

Nous allons présenter un exemple de génération de l'arbre de décision par l'algorithme J48(extension de l'algorithme C4.5) sous Weka avec une base des données nommée « contact-lenses.arff »et contient des attributs comme : Outlook, Temperature, Humidity, Windy, Play,cette base contient des valeurs manquantes. Le tableau-2.4 suivant présente les données « contact-lenses » avec des valeurs manquantes :

Outlook	Temperature	Humidity	Windy	Play
young	?	no	reduced	none
young	?	no	normal	soft
young	myope	yes	reduced	none
young	myope	?	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

TABLE 2.4 – La base des données contact-lenses

Si on supprime les valeurs manquantes du tableau-2.4, la génération de l'arbre de décision sous Weka est suivante :

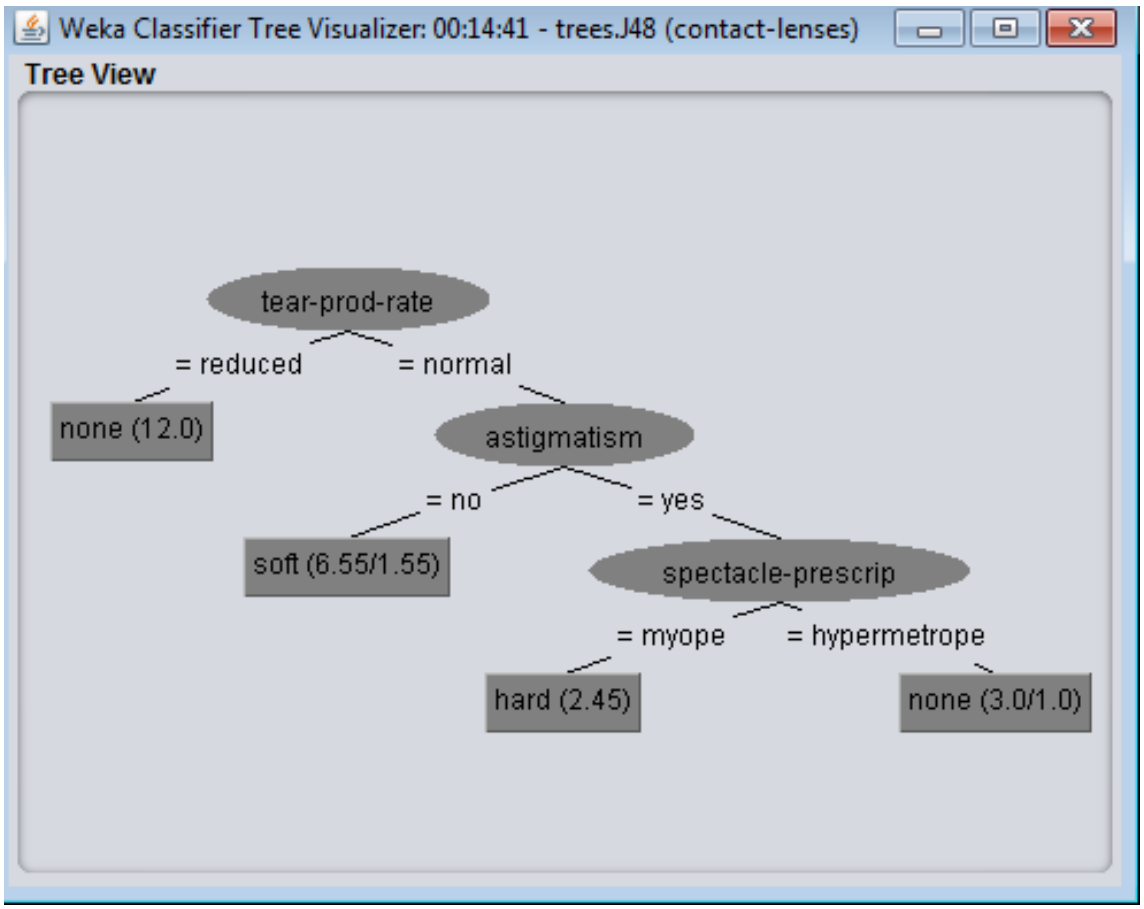


FIGURE 2.6 – Arbre de décision sous Weka

2.2.5 La fonction pour compléter attribut manquant

La figure-2.7 suivante illustre la fonction pour compléter les attributs manquants dans l'arbre de décision par l'algorithme C4.5, on change tout « ? » dans les attributs par la valeur de l'attribut la plus fréquente dans notre exemple.
[13] [14]

```
FONCTION Compléter_attribut_manq (Var T : Tab, nom_attribut : Entier)

Pour i de 0 à longueur(T[i]) Faire
    Pour nom_attribut de 0 à longueur(T[i]).dict_attributs Faire
     Si (T[i].dict_attributs[nom_attribut] == '?' ) Alors
    sous_ensemble = sous_ensemble_etiquette(T[i].etiquette)
    T[i].dict_attributs[nom_attribut] = sous_ensemble.valeur_plus_frequente_attribut(nom_attribut)
     FIN SI
  FIN POUR
     FIN POUR
FIN Compléter_ar
```

FIGURE 2.7 – La fonction pour compléter attribue manquante [14]

Avec T c'est une table d'exemples étiquetés.

2.2.6 Les attributs avec des valeurs continues

Pour chacune de ces partitions (les intervalles continus) on calcule le gain et en choisissant la partition qui maximise le gain.
[13]

•Exemple :

Le tableau-2.5 suivant qui illustre un exemple de l'état de la machine à l'instant t avec des valeurs continues de l'attribut température :

Température	Alimentation externe
59	Oui
73	Non
61	Non
45	Oui
79	Non
55	Oui
47	Oui

TABLE 2.5 – état de la machine

Pour une valeur continue, les branches ne nécessitent pas de représenter une valeur définir, néanmoins un ensemble délimité par une borne inférieure et une borne supérieure.

Essayons de discrétiser l'attribut température pour cet exemple. Il nécessite de commencer par trier les attributs de tableau-2.5 dans l'ordre croissant ou décroissant.

Le tableau-2.6 suivante, c'est le résultat de l'ordre croissant des valeurs de l'attribut température dans le tableau-2.5 :

Température	Alimentation externe
45	Oui
47	Oui
55	Oui
59	Oui
61	Non
73	Non
79	Non

TABLE 2.6 – état de la machine dans l'ordre croisant

On commence par le premier attribut « 45, Oui », si on voit le deuxième attribut, on a (47, Oui). Alors en change la valeur de classification entre (Température, 45) et (Température ,47). On fait donc la moyenne des deux attributs premiers :

Température (attribut (attribut_1, attribut_2)) =] 45,46[Cette procédure répété plusieurs fois jusqu'au bout et vous devriez obtenir :

- Température attribut_1=]45,46[∪[46,47[∪ [47,51[∪[51,55[∪[55,57[∪[57,59]
- Température attribut_2=]59,60[∪]60,61[∪ [61,67[∪[67,73[∪[73,76[∪[76,79]

2.2.7 Le fonctionnement de l'algorithme C4.5

Pour définir le fonctionnement de cet algorithme, nous allons présenter deux phases nécessaires utilisées par cet algorithme :

2.2.7.1 La phase d'expansion

Cette phase permettre de calculer un arbre de décision et peuvent être fournies par le recours des trois méthodes suivantes :

- Décider si un noeud est terminal.[11]
- Choisir un test à associer à un noeud.[11]
- Affecter une classe à une feuille. [11]

2.2.7.1.1 Décider si un noeud est terminal

Un nœud p est terminal, si tous les éléments joints à ce nœud sont dans une même classe ou si aucun test n'a pu être choisi.[11]

2.2.7.1.2 Choisir un test à associer à un noeud

À chaque étape, dans l'unité des tests disponibles, ne peuvent être découverts que les tests, il existe au moins deux branches ayant au moins deux éléments. Si aucun test ne satisfait alors le nœud est terminal.[11]

2.2.7.1.3 Affecter une classe à une feuille

Cette étape permettre d'attribuer la classe majoritaire d'arbre de décision. [11]

2.2.7.2 La phase d'élagage

Cette étape permettre de supprime les parties terminales, pour garder de bonnes performances prédictives. Cet algorithme utilise un ensemble d'apprentissage pour élaguer l'arbre génèré. Le critère d'élagage est basé sur une heuristique permettant d'estimer l'erreur réelle sur un sous arbre donné.[23]

il existe deux types d'élagage de l'arbre de décision :

- Pré-élagage :Si arrête le développement de l'arbre avant que tous les exemples de l'ensemble d'apprentissage soient bien classés.[23]
- Post-élagage :Si construit le modèle complet, puis élague les éléments qui contribuent au sur-apprentissage.[23]

2.2.8 L'algorithme de base

La figure-2.8 suivante illustre l'algorithme de base C4.5 :

Algorithme : Algorithme C4.5
Entrées : B : une base de données d'attributs, Tr= {}
Sortie : retourne un arbre de décision
Début
Si B est vide Ou autres critères d'arrêt rencontrés **Alors**
Retourner
Fin Si
Pour tout attribut a∈B **Faire**
Calculer des critères théoriques de l'information si nous divisons sur a
Fin Pour
A_{top}= meilleur attribut selon les critères calculés ci-dessus
Tr= créer un noeud de décision qui test A_{top} dans la racine
Bs= Sous-jeux de données induits à partir de B basé sur Atop
Pour tout attribut Bs **Faire**
Trs=C4.5(Bs)
Attacher Trs à la branche correspondante de Tr
Fin Pour
Retourner Tr
Fin

FIGURE 2.8 – Algorithme C4.5 [2]

2.2.9 Les limites

- La vitesse de traitement de données de l'algorithme C4.5 est lente par rapport au l'algorithme C5.0 (l'algorithme C5.0 nous avons déjà traite dans le chapitre 1).[2]
- Utilisation de la mémoire n'est pas optimale.[2]
- Algorithme C4.5 n'est pas optimale pour des bases des données des très grandes tailles.[2]
- Algorithme C4.5 ne permet pas la pondération dans les cas des erreurs de classement.[2]

2.3 Conclusion

Dans ce chapitre nous avons présenté l'algorithme de classification supervisée C4.5 en deux phases la phase d'expansion et la phase d'élagage, ensuite nous avons détaillé les améliorations proposées par cet algorithme.

Nous avons illustré cet algorithme par des exemples de génération de l'arbre de décision sous le logiciel « Weka ».

Dans le chapitre suivant, nous allons proposer l'utilisation de l'algorithme C4.5 afin de générer les arbres de décision de chaque partition de données.

Chapitre III

Chapitre 3

Technique proposé

3.1 Introduction

Le Data Mining parallèle et distribué et un axe de recherche qui a attiré beaucoup d'intérêt dans ces dernières années. [16] il a été utilisé dans plusieurs perspectives différentes à savoir :

- Amélioration du taux de précision de données.
- Diminution du temps de calcul de données.
- Amélioration de la vitesse de traitement de données.

L'apprentissage par arbre de décision est une méthode classique en Data Mining. Son but est de créer un modèle qui prédit la valeur d'une variable-cible depuis la valeur de plusieurs variables d'entrée. Cependant cette technique pose un certain nombre de problèmes, comme le fort coût de calcul.[16]

L'algorithme C4.5 reste parmi les meilleurs algorithmes de classification supervisée basés sur les arbres des décisions, mais il souffre comme la plus part des algorithmes séquentiels du fort coût de calcul surtout pour les grands volumes de données. [29]

Dans ce chapitre, nous allons proposer un algorithme qui est une version parallèle de construction de l'arbre de décision en utiliser la stratégie de partitionnement horizontal des données pour améliore les performances et simplifie la maintenance. Ensuite nous allons analyser les performances de cet algorithme, ainsi que les gains apportés.

3.2 Le principe général de notre technique

Dans ce travail, nous allons se baser sur la stratégie de partitionnement des données en proposant une version parallèle de construction des arbres de décision. En effet, le partitionnement des données est nécessaire dans beaucoup d'applications. Le traitement des données dans un seul site central peut engendrer plusieurs problèmes comme la capacité de stockage et le fort coût de calcul.

La parallélisation de l'algorithme C4.5 est motivée par plusieurs raisons :

- Malgré ses performances relatives, C4.5 souffre comme la plupart des algorithmes séquentiels du problème de scalabilité en présence de grand volume de données.
- La structure d'arbre utilisée permet de représenter les données sous une forme compacte, mais elle peut ne pas tenir en mémoire centrale, d'où la nécessité de construire des arbres locaux de taille plus réduite.

Notre application opère en trois étapes successives :

- Le chargement d'un fichier de données.
- Le partitionnement d'un fichier de données.
- La génération de l'arbre de décision.

3.2.1 La première phase : le chargement d'un fichier textuel

L'accès aux données c'est la première étape du processus de Data Mining, alors nous allons charger un fichier textuel (« Txt » ou « Csv ») dans un tableau de notre application. La figure-3.1 suivante illustre le chargement d'un fichier textuel dans une table :

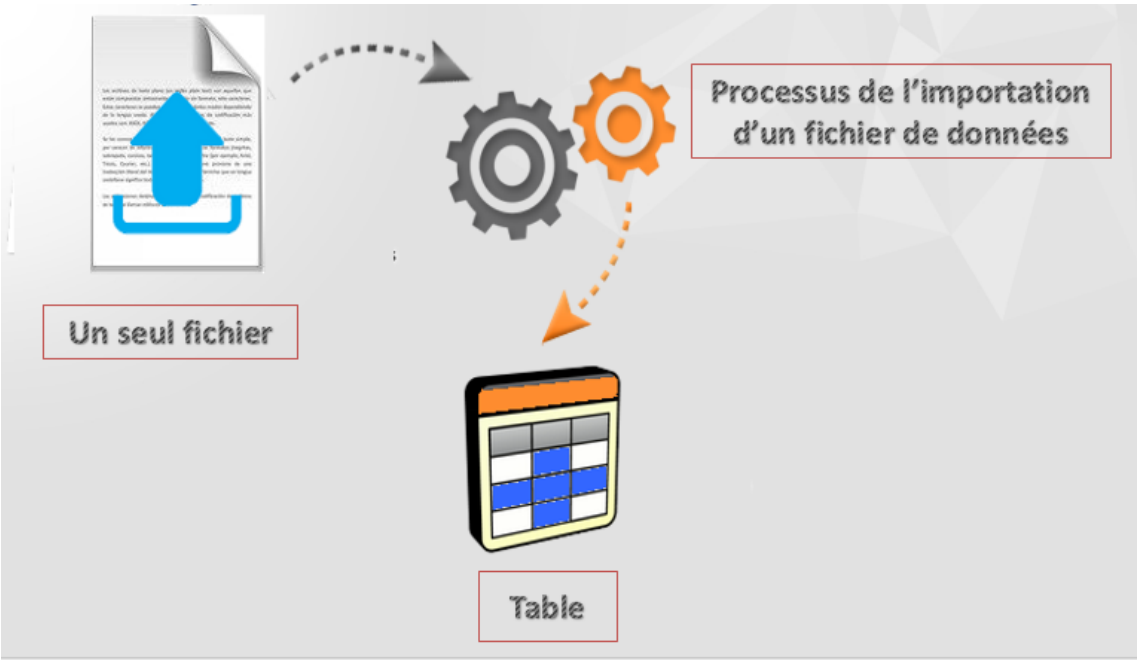


FIGURE 3.1 – Le chargement d'un fichier textuel

3.2.1.1 Les différents formats des fichiers textuels

On va actuellement s'intéresser à la représentation de données tabulaires sous forme de fichiers texte. Des données tabulaires sont des données que l'on peut représenter sous la forme d'un tableau [30], comme illustré dans le « tableau 8 » dans le chapitre précédent.

La figure-3.2 suivante illustre les différents formats des fichiers textuels :

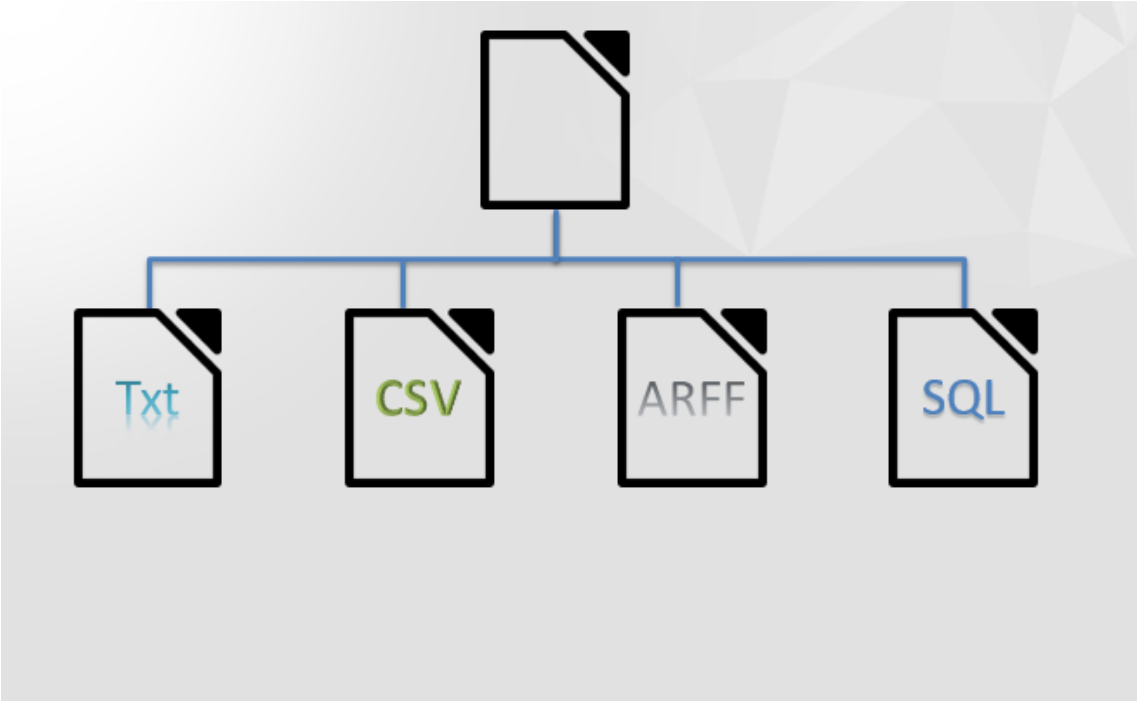


FIGURE 3.2 – les formats des fichiers textuels

D'après la figure-3.2, Il existe plusieurs formats de fichiers textuels, par exemple :

- Un fichier de format « CSV » ou « Comma-Separated Values » :

Un fichier CSV est un fichier tableur, contenant des données sur chaque ligne séparés par un caractère de séparation (généralement une virgule, un point-virgule ou une tabulation).[26]

Dans notre cas on utilise deux formats de fichier qui sont :

- Un fichier « Txt ». Et/Ou
- Un fichier « Csv ».

3.2.2 La seconde phase : le partitionnement d'un fichier de données

Dans une base de données, une partition est une division logique d'une table stockée en plusieurs parties indépendantes. Le partitionnement de tables est généralement effectué pour améliorer la gestion, la performance ou la disponibilité. Cela permet également d'effectuer des requêtes en parallèle sur plusieurs partitions (construction des arbres en parallèle).

La figure-3.3 suivante illustre la seconde phase du partitionnement d'un fichier textuel :

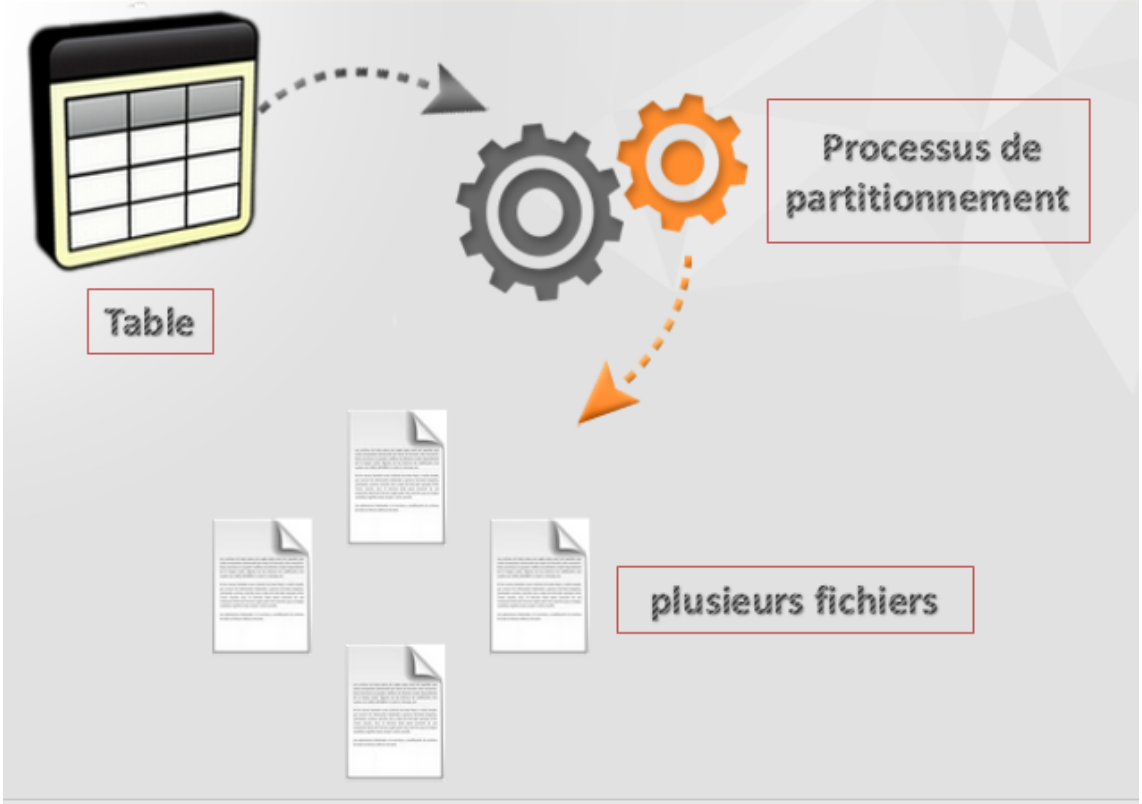


FIGURE 3.3 – Le partitionnement d'un fichier textuel

En général, Il existe plusieurs types de partitionnement des données, par exemple :

- Le partitionnement horizontal.
- Le partitionnement vertical.

3.2.2.1 Le partitionnement vertical à base d'attributs

Le partitionnement vertical d'une table consiste à fractionner une table en plusieurs tables contenant chacune un sous-ensemble des colonnes et le même nombre des lignes que la table partitionnée. [3]

• **Exemple :**
La figure-3.4 suivante, illustre le partitionnement vertical d'une table donnée :

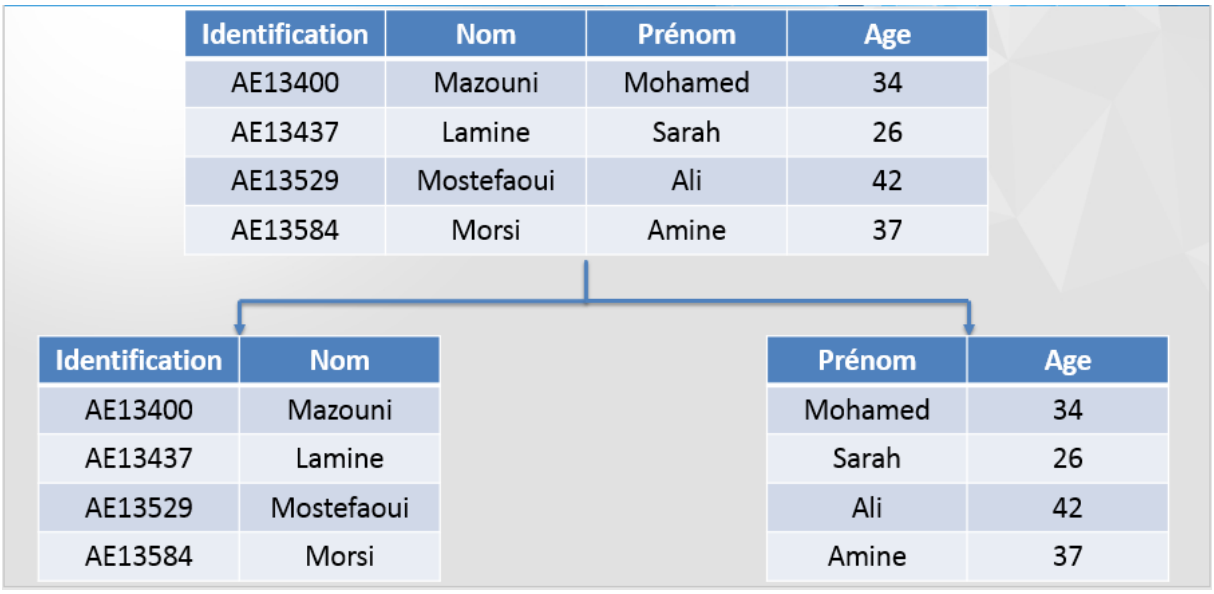


FIGURE 3.4 – Le partitionnement vertical d'une table.

3.2.2.2 Le partitionnement horizontal à base d'enregistrements

Le partitionnement horizontal d'une table, consiste à fractionner une table en plusieurs tables contenant chacune un sous-ensemble des lignes et les mêmes colonnes que la table partitionnée. Le partitionnement horizontal d'une table permet de réduire l'interrogation des données en réduisant le nombre de lignes d'une table.[26]

•**Exemple :**
La figure-3.5 suivante illustre le partitionnement horizontal d'une table donnée :

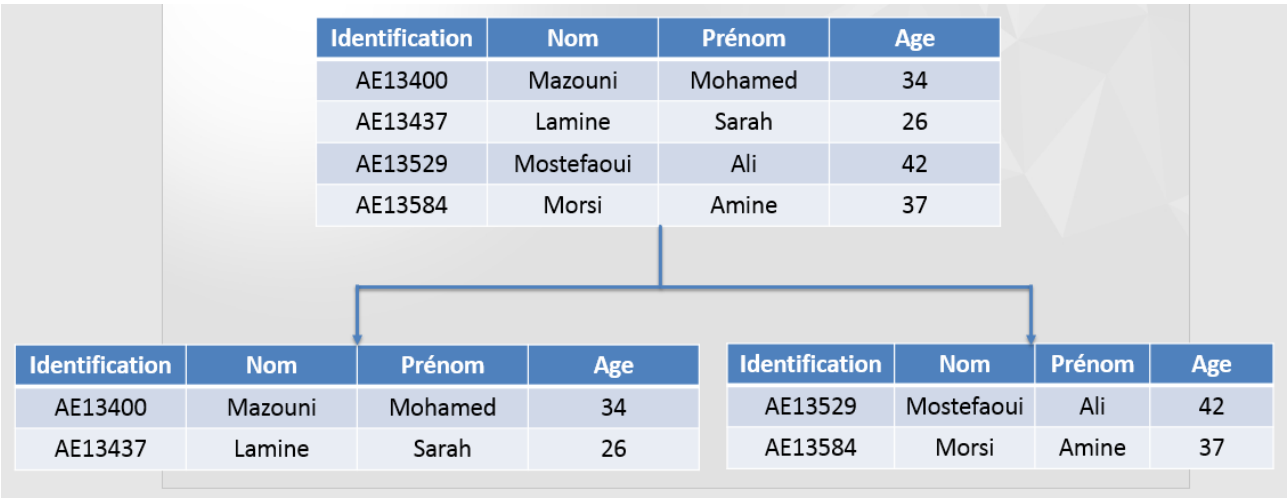


FIGURE 3.5 – Le partitionnement horizontal d'une table.

Dans notre technique on utilise seulement le partitionnement horizontal de la table des données, parce que si on fractionne une grande table en tables individuelles plus petites, les requêtes qui qu'à une fraction des données peuvent être exécutées plus rapidement, car la qualité de données à analyser est moindre.

3.2.3 La génération des arbres de décision

Notre algorithme prend en entrée la base de données partitionnée horizontalement sur Plusieurs Thread, ensuite chaque Thread construit son arbre de décision par l'utilisation de l'algorithme C4.5.

Pour gérer chacun des threads, il faut utiliser un mécanisme très avance, par exemple :

- MPI (Message Passing Interface) :
  C'est un mécanisme développe en 1994 pour obtenir de très bonnes performances

aussi bien sur des machines à mémoire distribuée ou/et parallèle.[26]

La figure-3.6 suivante illustre le processus de chargement des données dans les threads d'une machine :

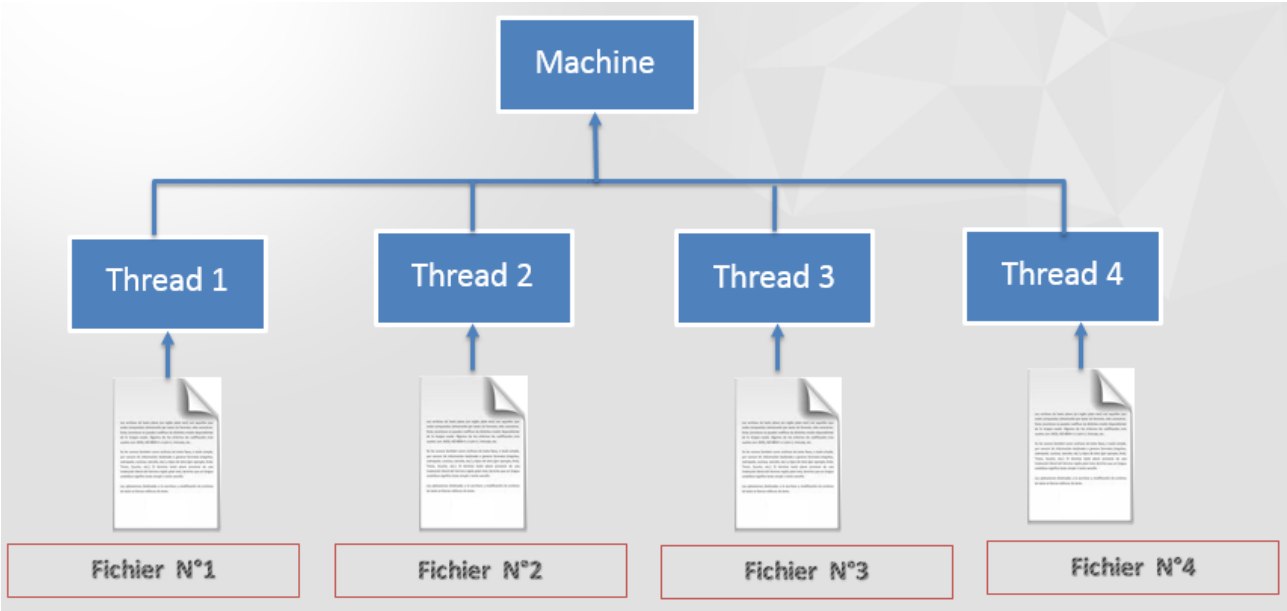


FIGURE 3.6 – Le chargement des données.

La figure-3.7 suivante illustre la génération de l'arbre de décision de chaque partition des données :

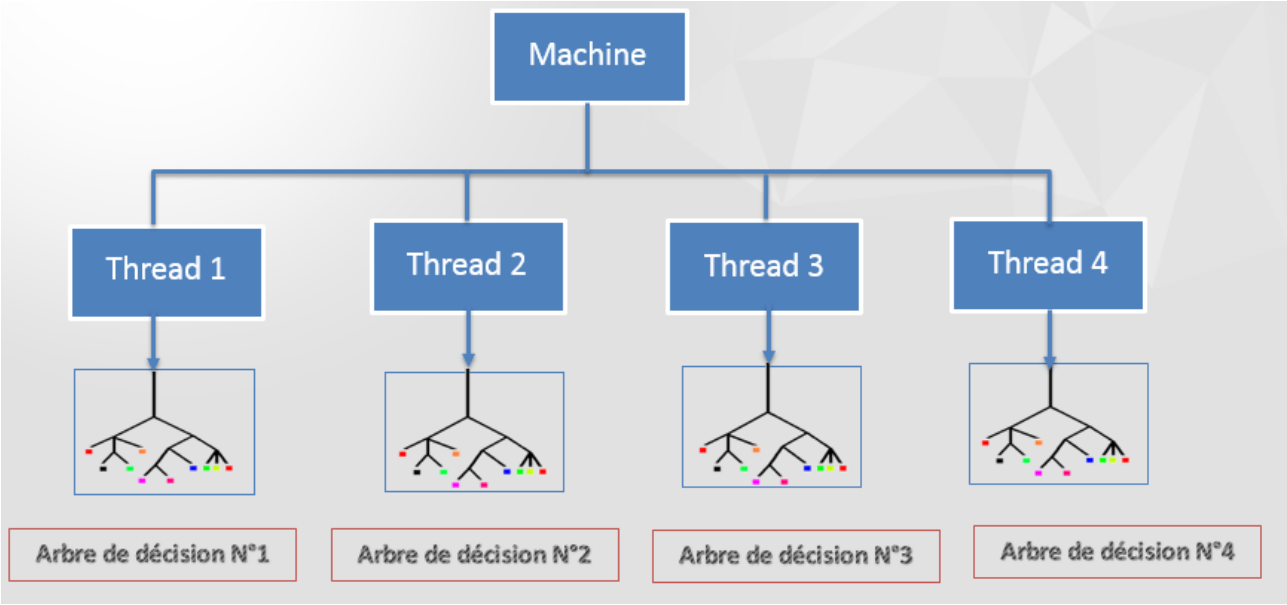


FIGURE 3.7 – La génération des arbres des décisions.

3.3 Évaluation des performances de l'algorithme proposé

Dans cette partie, nous présentons les résultats des différentes expérimentations que nous avons réalisées pour valider notre technique. Nous étudierons plus particulièrement les points suivants :

- Le temps d'exécution.
- L'utilisation espace mémoire.

Le tableau-3.1 suivant possède les temps d'exécution et l'occupation mémoire par l'algorithme proposé et C4.5 que nous allons calculer après plusieurs itérations :

Taille de jeux de données (les nombres des enregistrements)		6000	4000	9000
Algorithme C4.5(Second)	Le temps d'exécution	6.77	5.32	10.002
	L'utilisation espace mémoire	34.67	29.61	48.318
Algorithme proposé(Second)	Le temps d'exécution	4.12	2.95	8.713
	L'utilisation espace mémoire	32.005	28.372	45.9

TABLE 3.1 – Comparaison du temps d'exécution et l'espace mémoire pour les algorithmes proposé et C4.5

pour calculer le temps d'exécution de l'algorithme proposé dans le tableau-3.1, il faut calculer le temps d'exécution d'une partition des données.

pour calculer l'utilisation de l'espace mémoire total de l'algorithme proposé dans le tableau-3.2, il faut calculer utilisation de l'espace mémoire des différentes partitions des données.

Donc le but est de comparer l'amélioration apportée par notre technique.

3.4 Exemples d'implémentation de notre application

On a utilisé un langage java seulement (sans utiliser aucune bibliothèque) pour pouvoir développer Notre application, la figure-3.8 suivante présente l'interface principale de notre application avec les différents outils :

FIGURE 3.8 – Interface principale de notre application

La figure-3.9 suivante présente l'interface de chargement des données de notre application :

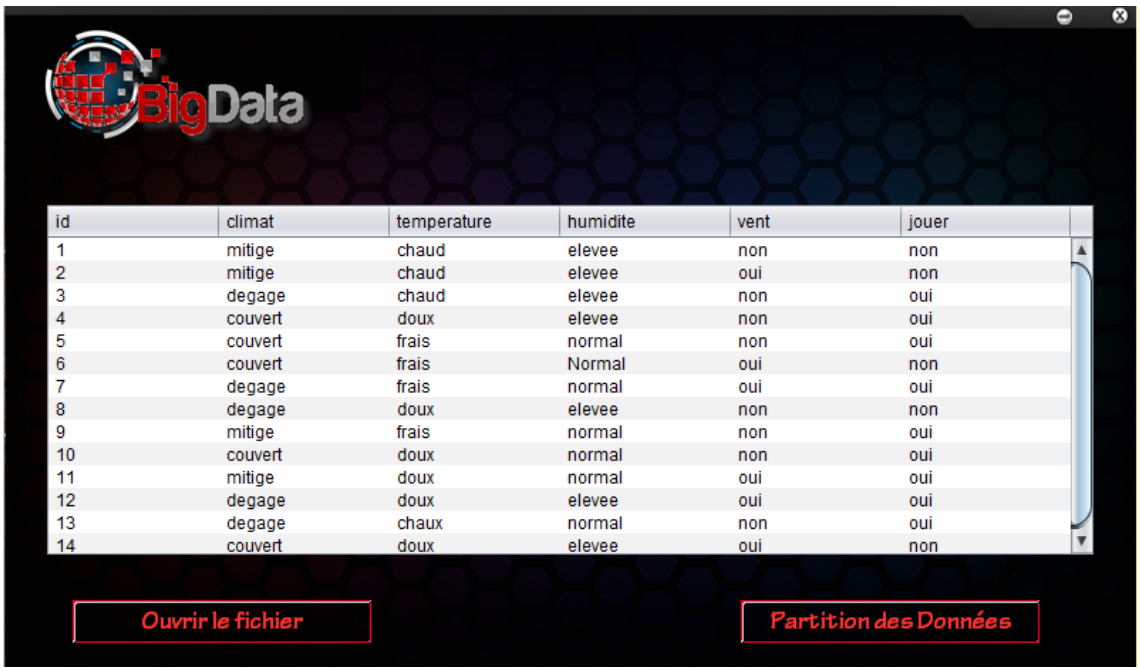


FIGURE 3.9 – Interface de la fenêtre de chargement des données

Donc, la figure-3.10 suivante présente l'interface de la fenêtre de partitionnement horizontal des données :

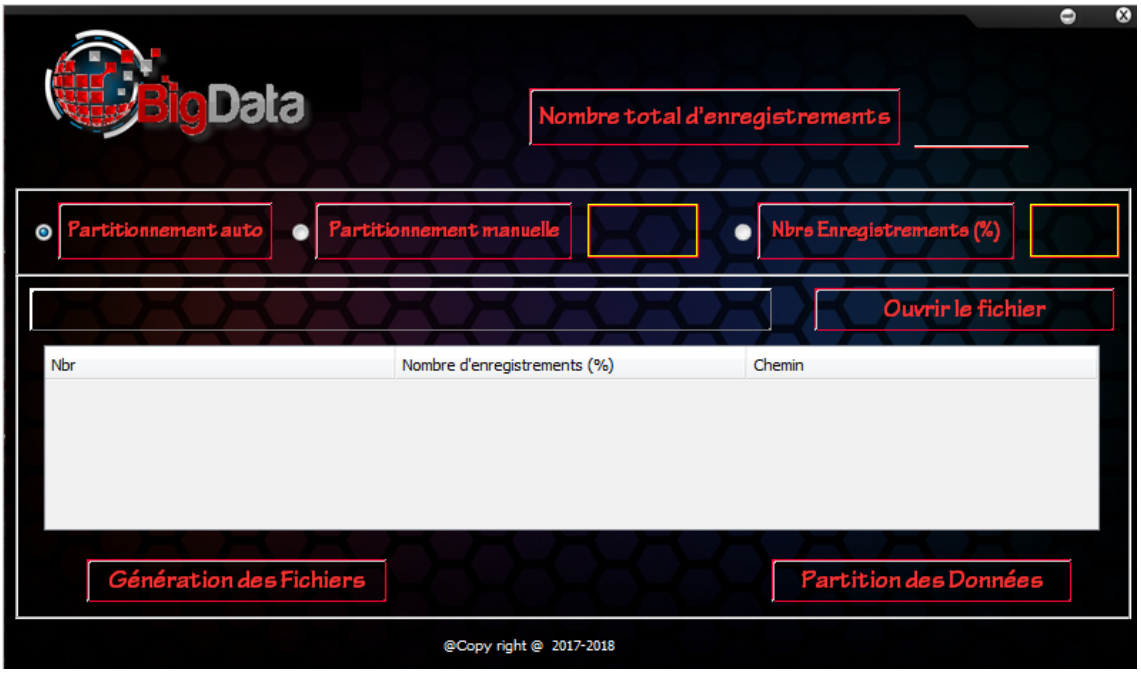


FIGURE 3.10 – Interface de la fenêtre de Partitionnement des données

Donc, la figure-3.11 suivante présente l'interface de la génération de l'arbre de décision d'une partition des données "weather.csv" :

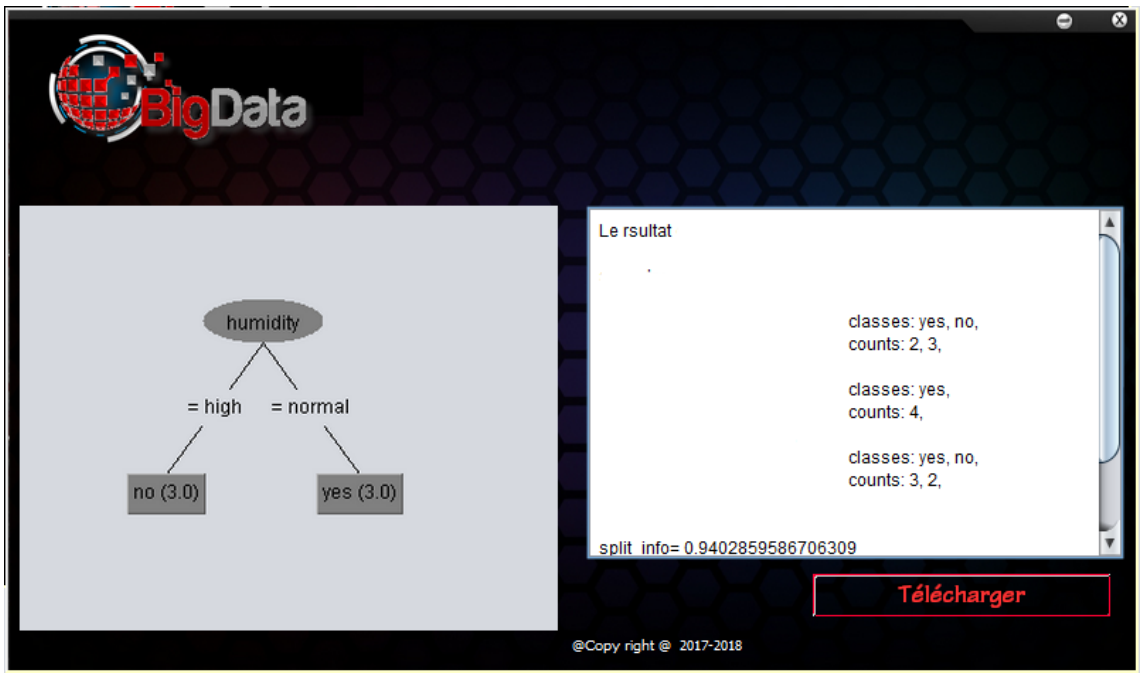


FIGURE 3.11 – Interface de la génération des arbres des décisions par notre application de Data Mining

3.5 Conclusion :

Dans ce dernier chapitre, nous avons proposé une technique parallèle de construction des arbres de décision en utilisant la stratégie de partitionnement horizontal des données, afin de profiter des avantages de partitionnement à savoir l'espace mémoire et le temps d'exécution.

Notre algorithme est basé sur l'un des meilleurs algorithmes séquentiels de classification qui est le C4.5.

Les résultats obtenus montrent que notre algorithme est plus performant que l'algorithme C4.5 en matière du temps d'exécution et d'espace mémoire, ceci est justifié par la division horizontale de la base donnée et le travail parallèle sur plusieurs threads.

Conclusion Générale

Conclusion Générale

Dans ce mémoire,nous intéresser au Data Mining pour résoudre le problème de l'un des meilleurs algorithmes séquentiels de classification C4.5, qui malgré ses performances relatives, il risque une saturation de l'espace mémoire et un coût de calcul élevé surtout pour les grosses bases de données.[19]

Les algorithmes basés sur l'arbre de décision fournissent des méthodes effectives qui emporter de bons résultats dans la pratique. les algorithmes possèdent l'avantage d'être compréhensibles par l'utilisateur si la taille de l'arbre produit est raisonnable. [22]

Après une comparaison entre les algorithmes de classification basés sur l'arbre de décision (ID3, C4.5, C5.0, CART , SLIQ, SPRINT), nous avons proposé une version parallèle de construction des arbres de décision en utilisant la stratégie de partitionnement horizontal des données. Partons de ce partitionnement, le traitement parallèle des partitions permet d'obtenir une diminution significative de la taille de l'espace de recherche.

Les résultats expérimentaux montrent que l'algorithme proposé permet de générer les arbres de décision de chaque partition, il permet de diminuer le temps de génération grâce à l'exploitation du parallélisme. Ils montrent également que notre algorithme permet de réduire considérablement l'espace mémoire nécessaire pour l'extraction des données, ceci grâce à la division de l'espace de recherche.

Dans nos futurs travaux, nous essayerons de compléter notre algorithme par l'évaluation des performances des arbres des décisions et l'utilisation des bases de test, et la représentation du synthèse de calcul par une matrice de confusion. Comme autre perspective, nous envisageons aussi d'expérimenter notre algorithme sur d'autres types de plateformes, notamment des multiprocesseurs et des machines massivement parallèles comme les grilles.

Résumé

Résumé

Les algorithmes de Data Mining nécessitent généralement de grands capacités de traitement qui peuvent être fournies par le recours à des traitements parallèles et distribués. Plusieurs solutions se présentent entre autres : la stratégie « diviser et régner » facilite le découpage du problème en un ensemble de sous problèmes plus simples à traiter.

Le but de ce projet est de développer une solution parallèle pour des problèmes de Data Mining (tel que la classification), fournissant ainsi plusieurs sorte de gains pour le temps d'exécution et d'espace mémoire.

Nous proposons une technique parallèle de construction des arbres de décision en utilisant la stratégie de partitionnement horizontal des données et en se basant sur l'algorithme C4.5.

Notre technique s'opère en trois étapes successives comme le chargement d'un fichier de données, le partitionnement d'un fichier de données et la génération de l'arbre de décision.

MOTS CLÈS :

Data Mining, Classification, Arbre de décision, Traitement Parallèle et Distribué.

Abstract

Data Mining aims to explore data in order to extract useful knowledge. A major challenge is the ability of existing algorithms to scale. Indeed with databases of the order of terabytes, it is important to do the task of datamining in a minimum of computation time.

Data Mining algorithms usually require large processing capabilities that can be provided through the use of parallel and distributed processing.
Several solutions are presented among others : the "divide and rule" strategy facilitates the division of the problem into a set of sub-problems that are easier to deal with.

The goal of this project is to develop a parallel or distributed solution for Data Mining problems (such as classification), thus providing several kinds of gains for execution time and memory space.

We propose a parallel technique of construction of decision trees using the strategy of horizontal partitioning of the data and based on algorithm C4.5.

Our technique takes place in three successive steps such as loading a data file, partitioning a data file and generating the decision tree.

KEYWORDS :

Data Mining, Classification, Decision tree, Parallel and Distributed Processing.

Bibliographie

[1] A. B. Les arbres de décisions tutoriels zeste de savoir. 2015. `https://zestedesavoir.com/tutoriels/962/les-arbres-de-decisions/`.

[2] J. Baltié, S. Specialisation, and R. M. Adjaoute. Datamining : Id3 et c4. 5. *Epita SCIA*, 2002.

[3] L. Bellatreche. Techniques d'optimisation des requetes dans les data warehouses. pages 81–98, 2003.

[4] S. Bhattacharyya, S. De, I. Pan, and P. Dutta. Intelligent multidimensional data clustering and analysis. 2016.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees (chapman y hall, eds.). *Monterey, CA, EE. UU. : Wadsworth International Group*, 1984.

[6] G. CALAS and H.-F. CHADEISSON. Les algorithmes sliq et ssdm. 2013.

[7] P. Charoenporn. Reservoir inflow forecasting using id3 and c4. 5 decision tree model. pages 698–701, 2017.

[8] B. Christophe. Enjeux et usages du big data : Technologies, méthodes et mise en oeuvre. 2013.

[9] B. Devéze and M. Fouquin. Datamining c4. 5–dbscan. *PROMOTION, SCIA Ecole pour. I'informatique et techniques avancees*, 2005.

[10] O. Francois and P. Leray. Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens. *Rencontres des Jeunes Chercheurs en IA*, 2003.

[11] A. Girard. Exploration d'un algorithme génétique et d'un arbre de décision à des fins de catégorisation. 2007.

[12] J. Han, M. Kamber, and J. Pei. Data mining : concepts and techniques (the morgan kaufmann series in data management systems). *Morgan Kaufmann*, 2000.

[13] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2), 2014.

[14] S. Jami, T.-Y. Jen, D. Laurent, G. Loizou, and O. Sy. Extraction de règles d'association pour la prédiction de valeurs manquantes. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 3 :103–124, 2005.

[15] S. Lallich, P. Lenca, and B. Vaillant. Construction d'une entropie décentrée pour l'apprentissage supervisé. *Qualité des Données et des Connaissances*, 2007.

[16] K. Liu, H. Kargupta, and J. Ryan. Distributed data mining bibliography : Release 1.7. *Baltimore : University of Maryland, Computer Science Department*, 2006.

[17] C. Marsala. Apprentissage inductif en présence de données imprécises : Construction et utilisation d'arbres de décision flous. 1998.

[18] M. Mehta, R. Agrawal, and J. Rissanen. Sliq : A fast scalable classifier for data mining. pages 18–32, 1996.

[19] A. MOUNA. Datamining distribue dans les grilles : Approche regles d'association. 2013.

[20] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. 10(2010), 2010.

[21] F. Poulet and P. Kuntz. *Visualisation en extraction de connaissances*. PhD thesis, 2006.

[22] S. RAHMOUN. Méthodes d'apprentissage pour améliorer laqos d'une flotte de logiciels embarqués. 2015.

[23] R. Rakotomalala. Arbres de décision. *Revue Modulad*, 33 :163–187, 2005.

[24] S. L. Salzberg. C4. 5 : Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3) :235–240, 1994.

[25] J. Shafer. Lvdb, sprint : A scalable parallel classifier for data mining. 1996.

[26] Y. Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005.

[27] T. Stéphane. Modélisation prédictive et apprentissage statistique avec r. 2015.

[28] A. Upadhayay, S. Shukla, and S. Kumar. Empirical comparison by data mining classification algorithms (c 4.5 & c 5.0) for thyroid cancer data set. *International Journal of Computer Science & Communication Networks*, 3(1) :64, 2013.

[29] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1) :1–37, 2008.

[30] K. Zeitouni. Analyse et extraction de connaissances des bases de données spatio-temporelles. 2006.