

CHAPITRE 4

CATEGORISATION DES TEXTES PAR ARBRES DE DECISION

4.1. Introduction :

L'apprentissage inductif [29] consiste à extraire un modèle à partir d'un ensemble d'exemples de cas résolus par des experts d'un domaine. À partir de cet ensemble, appelé *ensemble d'apprentissage* (voir tableau 4.2), on génère un modèle qui sert à étudier de nouveaux exemples dans le même domaine. Chaque exemple (objet) de cet ensemble est représenté par un vecteur d'attributs et chaque attribut prend un ensemble de valeurs.

Si la classe de chaque vecteur est donnée, on se situe dans le cadre de l'*apprentissage supervisé*, dont les méthodes sont les arbres de décision, les règles d'associations, les réseaux bayésien, etc. Dans ce cadre, on cherche à utiliser les exemples fournis déjà classés pour apprendre un modèle qui permet ensuite de déterminer la classe de tout nouvel exemple rencontré.

L'induction avec des arbres de décision est l'une des formes d'algorithme d'apprentissage les plus simples et pourtant les plus efficaces [30].

Les *arbres de décision* sont une des techniques les plus populaires de l'apprentissage automatique et de la fouille de données. L'apprentissage par arbre de décision se situe dans le cadre de l'*apprentissage supervisé*, où la classe de chaque objet dans la base est donnée. Le but est de construire un modèle à partir d'un ensemble d'exemples associés aux classes pour trouver une description pour chaque classe à partir des propriétés communes entre les exemples. Une fois ce modèle construit, on peut extraire un ensemble de règles de classement. Ce modèle ou les règles extraites sont ensuite utilisés pour classer de nouveaux objets dont la classe est inconnue.

Le classement se fait en parcourant un chemin depuis la racine jusqu'à une feuille. La classe renvoyée est celle qui est la plus fréquente parmi les exemples de la feuille [31].

4.2. Définition :

Un arbre de décision est une structure qui est souvent utilisée pour représenter des connaissances. Il permet de remplacer ou d'assister un expert humain dans la détermination des propriétés d'un objet, c'est l'opération de *classement* (en anglais : *classification*). Un arbre de décision est une représentation d'une procédure de décision pour déterminer la classe d'un objet donné.

En général, à chaque nœud interne de l'arbre, il y a un test (question) qui correspond à un attribut dans la base d'apprentissage, et une branche correspondant à chacune des valeurs possibles de l'attribut. À chaque nœud de feuille, il y a une valeur de classe. Les arbres de décision sont construits à partir d'un ensemble d'apprentissage. Un chemin de la racine à un nœud correspond à une série d'attributs (questions) avec leurs valeurs (réponses).

Par exemple, prenons l'arbre de décision donné dans la figure 4.1 et construit à partir de la base de météo (tableau 4.2) : les nœuds internes sont *Temps*, *Humidité* et *Vent*. Pour classer un objet dont le *temps* est *ensoleillé*, l'*humidité* est *haute* et le *vent* est *faux*, on part de la racine de cet arbre et on pose la question sur *Temps*. La réponse qui est la valeur *ensoleillé* nous aide à déterminer quelle branche on doit choisir pour descendre dans l'arbre. On continue le parcours et on pose une question sur l'*humidité*. La réponse à cette question est *haute*, ce qui nous conduit à arriver à une feuille ayant la valeur A, qui est la classe de cet objet [31].

4.3. Les types de données:

Dans ce paragraphe, nous décrivons les types de données dans une base d'apprentissage.

Nous pouvons diviser le type d'un attribut en deux grandes catégories :

4.3.1. Quantitative (Numérique) : Si l'ensemble des valeurs qu'il peut prendre est un ensemble de nombres, fini ou infini, ou un intervalle de valeurs réelles. Un attribut X numérique peut être *discret* ou *continu* selon sa nature :

- **Continu :**

Si l'ensemble des valeurs qu'il peut prendre est réel ou un intervalle réel. Il s'agit donc d'un ensemble infini non dénombrable : on ne peut pas énumérer systématiquement l'ensemble de tous les points d'un intervalle réel. Par exemple, X peut être l'âge d'une personne prise au hasard, sa taille, son poids, etc [31].

- **Discret :**

Si l'ensemble des valeurs qu'il peut prendre est un ensemble numérique fini (comprenant un nombre fini d'éléments) ou un ensemble infini dénombrable (comprenant une infinité de nombres que l'on peut énumérer) [31].

4.3.2. Qualitative (symbolique) : Si l'ensemble des valeurs qu'il peut prendre est non numérique.

X peut être par exemple la couleur des yeux d'une personne prise au hasard, sa région de naissance, son sexe, etc. D'autre part, une donnée numérique ou symbolique peut être **ordinaire** ($<$, $>$) si ses

valeurs sont ordonnées. Par exemple, l'attribut dont les valeurs sont {bien, très-bien, excellente} est un attribut ordinal symbolique ; l'attribut dont les valeurs sont {1, 2, 3, 4, 5} est un attribut ordinal numérique (discret).

De plus, si les valeurs d'un attribut discret ou symbolique sont **binaires**, on parle d'un attribut binaire, par exemple l'attribut symbolique *sexe* qui prend les valeurs {masculin, féminin} ou un attribut discret qui prend les valeurs {0,1} [31].

Un attribut symbolique est dit **nominal** si l'ordre n'est pas important, comme le *groupe sanguin* {A, B, AB, O} ou l'*état civil* {marié, célibataire, divorcé, veuf}.

Les types de données sont résumés dans le tableau 4.1.

Un attribut quantitatif discret peut être traité comme une variable qualitative en considérant chaque valeur de l'attribut comme une modalité.

Dans notre travail, nous traitons les attributs symboliques et les attributs discrets. Si les attributs dans la base d'apprentissage sont continus, on applique des méthodes de discrétisation pour les rendre discrets. La discrétisation des attributs continus sera expliquée dans la troisième partie de ce mémoire[31].

Quantitative (numérique)			Qualitative (symbolique)			
Continue	Discrete		Nominale		Ordinale	
Taille, âge	Binaire	\neg Binaire	Binaire	\neg Binaire	Binaire	\neg Binaire
	{0,1}	{1,2,3,4}	{yes,no}	{A,B,AB,O}	{low, upper}	{+,++,+++}

Table 4.1 : Les types de données [31].

4.4. Construction d'un arbre de décision :

L'idée centrale qui préside à la construction d'un arbre de décision consiste à diviser récursivement les objets de l'ensemble d'apprentissage en utilisant des tests définis à l'aide des attributs jusqu'à ce que l'on obtienne des feuilles ne contenant (idéalement) que des objets appartenant tous à la même classe. Pour diviser l'ensemble d'apprentissage, on choisit des attributs qui vont minimiser l'impureté dans les sous-arbres ; autrement dit, qui maximisent l'information apportée par les réponses. C'est-à-dire que pour chaque attribut qui n'a pas encore été utilisé, on calcule l'impureté qui reste après son utilisation. Celui qui laisse le moins de désordre est choisi comme étant le prochain nœud de l'arbre de décision ; on répète le processus sur chaque nouveau

nœud. Le processus s'arrête quand les feuilles de l'arbre ainsi obtenu contiennent des exemples d'un seul concept (classe) ou quand aucun test n'apporte plus d'amélioration [31].

Dans toutes les méthodes de construction d'un arbre de décision, on trouve les trois opérateurs suivants (**divide-and-conquer strategy**) :

- Décider si un nœud est terminal : tous les exemples (un ou plus) appartiennent à la même classe (il y a moins d'un certain nombre d'erreurs).
- Sélectionner un test à associer à un nœud.
- Affecter une classe à une feuille. On attribue la classe majoritaire à une feuille [31].

La construction d'un arbre de décision se base sur le principe de **Top-Down Induction** [32] : On commence par construire la racine de l'arbre en continuant jusqu'à la feuille. A chaque étape un test sur le nœud courant est choisi, le choix d'un test se fait en mesurant l'impureté (l'incertitude). Le but est de trouver l'arbre de décision le plus petit possible. Pour choisir un attribut test, on a besoin de calculer le désordre. Une des fonctions utilisée à ce fin est l'index de Gini qui est expliqué plus tard dans CART [33]. Une autre fonction possible est la formule classique de l'entropie, qui est :

$$H(C) = \sum_{i=1}^{i=k} P_i \log_2 P_i \quad (1)$$

Où P_i est la probabilité de la classe C_i dans l'ensemble d'apprentissage. Cette formule explique la quantité d'information. S'il n'y a qu'une seule classe, l'entropie est nulle. Si les k classes sont équiprobables, elle vaut $\log_2(k)$ où k est le nombre de classes. Pour partitionner l'ensemble d'apprentissage T , on va calculer l'information gagnante par attribut en utilisant la formule de l'entropie conditionnelle :

$$H(C|A) = \sum_{i=1}^{i=n} P(A = a_i) H(C|a_i) \quad (2)$$

Où n est le nombre des valeurs possibles pour l'attribut A . Le gain d'information (l'information mutuelle) est :

$$gain(A, C) = IM(A, C) = H(C) - H(C|A) \quad (3)$$

On choisit l'attribut qui maximise le gain d'information, c'est-à-dire qui minimise l'entropie.

Après avoir choisi un attribut test, on doit partitionner l'ensemble en sous-ensembles selon les valeurs possibles de cet attribut. On crée les branches correspondant à chaque valeur de l'attribut, on crée des nœuds pour chaque sous-ensemble non vide et on répète le processus pour chaque nouveau nœud jusqu'à ce qu'on arrive à un critère d'arrêt. Dans ce cas, on crée une feuille et on associe à cette feuille la classe la plus probable [31].

Exemple :

Considérons l'ensemble d'apprentissage donné dans le tableau 4.2 extrait de [34]:

Pour construire l'arbre de décision à partir de l'ensemble d'apprentissage donné dans le tableau 4.2, on va chercher l'attribut le plus pertinent qui maximise le gain d'information (équation (3)):

$$IM(Décision, Temps) = H(Décision) - H(Décision/Temps) = 0.940 - 0.694 = 0.246$$

id	Temps	Température	Humidité	Vent	Décision
1	Ensoleillé	Elevée	Haute	Faux	A
2	Ensoleillé	Elevée	Haute	Vrai	A
3	Couvert	Elevée	Haute	Faux	B
4	Pluvieux	Moyenne	Haute	Faux	B
5	Pluvieux	Basse	Normale	Faux	B
6	Pluvieux	Basse	Normale	Vrai	A
7	Couvert	Basse	Normale	Vrai	B
8	Ensoleillé	Moyenne	Haute	Faux	A
9	Ensoleillé	Basse	Normale	Faux	B
10	Pluvieux	Moyenne	Normale	Faux	B
11	Ensoleillé	Moyenne	Normale	Vrai	B
12	Couvert	Moyenne	Haute	Vrai	B
13	Couvert	Elevée	Normale	Faux	B
14	Pluvieux	Moyenne	Haute	Vrai	A

Table 4.2 : La base Météo [34].

$$IM(Décision, Température) = H(Décision) - H(Décision/Température) = 0.940 - 0.910 = 0.030$$

$$IM(Décision, Humidité) = H(Décision) - H(Décision/Humidité) = 0.940 - 0.786 = 0.154$$

$$IM(Décision, Vent) = H(Décision) - H(Décision/Vent) = 0.940 - 0.891 = 0.049$$

L'attribut qui maximise le gain d'information est *Temps*. Donc, *Temps* est la racine de l'arbre de décision. L'ensemble d'apprentissage sera partitionné en trois sous-ensembles selon les trois valeurs de *Temps*, qui sont *ensoleillé*, *pluvieux* et *couvert*. Les trois sous-ensembles sont donnés dans les tableaux 4.3, 4.4, 4.5.

Pour continuer à construire l'arbre, on recommence le processus en choisissant d'autres attributs pertinents pour chaque sous-ensemble d'apprentissage (tableaux 4.3, 4.4, 4.5). Ce processus s'arrête quand les feuilles de l'arbre ainsi obtenu contiennent des exemples d'un seul concept ou quand aucun test ne donne d'améliorations.

Dans le sous-ensemble correspondant à la valeur *ensoleillé* de l'attribut *Temps* (tableau 4.3), l'attribut le plus pertinent est *Humidité*. Donc, le sous-ensemble est partitionné en deux sous ensembles selon les valeurs *haute* et *normale* de l'*humidité*.

Dans le tableau 4.4, nous remarquons que tous les objets appartiennent à la classe *B*; dans ce cas, le processus s'arrête dans cette branche et on crée une feuille avec la classe *B*.

Dans le sous-ensemble correspondant à la valeur *pluvieux* de l'attribut *Temps* (tableau 4.5), c'est l'attribut *Vent* qui maximise le gain et le sous-ensemble est partitionné en deux sous-ensembles selon les valeurs *vrai* et *faux* du *Vent* [31].

L'arbre de décision final obtenu est donné dans la figure 4.1.

id	Temps	Température	Humidité	Vent	Décision
1	Ensoleillé	Elevée	Haute	Faux	A
2	Ensoleillé	Elevée	Haute	Vrai	A
8	Ensoleillé	Moyenne	Haute	Faux	A
9	Ensoleillé	Basse	Normale	Faux	B
11	Ensoleillé	Moyenne	Normale	Vrai	B

Table 4.3 : Le sous-ensemble associé à la valeur ensoleillé de Temps [31].

id	Temps	Température	Humidité	Vent	Décision
3	Couvert	Elevée	Haute	Faux	B
7	Couvert	Basse	Normale	Vrai	B
12	Couvert	Moyenne	Haute	Vrai	B
13	Couvert	Elevée	Normale	Faux	B

Table 4.4 : Le sous-ensemble associé à la valeur couvert de Temps [31].

id	Temps	Température	Humidité	Vent	Décision
4	Pluvieux	Moyenne	Haute	Faux	B
5	Pluvieux	Basse	Normale	Faux	B
6	Pluvieux	Basse	Normale	Vrai	A
10	Pluvieux	Moyenne	Normale	Faux	B
14	Pluvieux	Moyenne	Haute	Vrai	A

Table 4.5 : Le sous-ensemble associé à la valeur pluvieux de Temps [31].

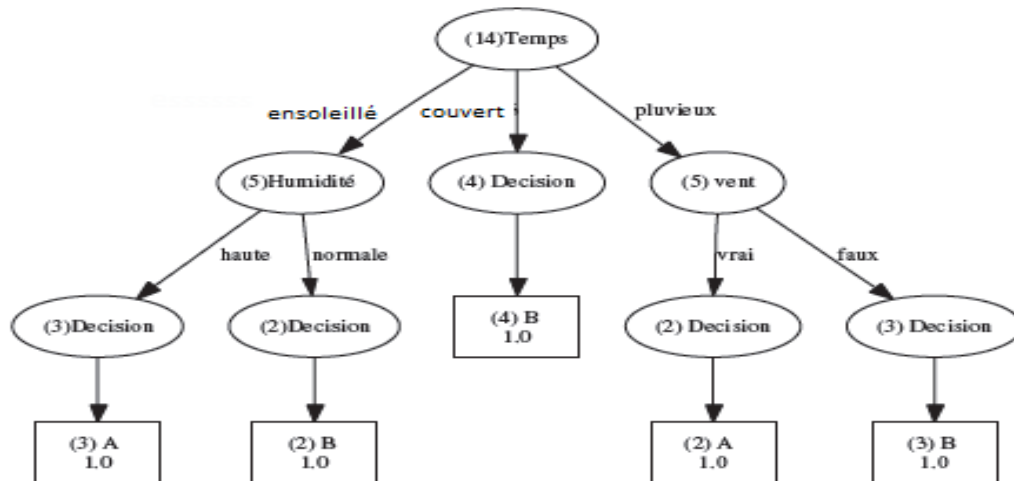


Figure 4.1 : L'arbre de décision final pour la base météo [31].

4.5. Les méthodes de construction d'un arbre de décision :

En apprentissage automatique, il existe de nombreux systèmes d'induction qui construisent des arbres de décision. Hunt, Martin et Stone [36] ont été les premiers à étudier l'apprentissage automatique à partir des exemples. Leur CLS (Concept Learning System framework) construit un arbre de décision qui essaie de minimiser le coût de classement d'un objet. Il y a deux types de coûts : 1) le coût de la détermination de la valeur d'une propriété de l'objet ; 2) le coût du mauvais classement. CLS utilise une stratégie appelée *lookahead* qui consiste à explorer l'espace de tous les arbres de décision possibles à une profondeur fixe et à choisir une action pour minimiser le coût dans cet espace limité puis avancer d'un seul niveau vers le bas dans l'arbre.

Dans le reste de ce chapitre, nous allons expliquer les méthodes les plus populaires : ID3, C4.5 et CART.

4.5.1. ID3 :

Dans cette méthode [36], on génère tous les arbres de décision possibles pour classer correctement l'ensemble d'apprentissage et pour choisir l'arbre le plus simple. Le nombre de ces arbres est très grand, mais fini. La structure de base pour cette méthode est itérative. Un sous-ensemble de l'ensemble d'apprentissage est choisi aléatoirement ; ce sous-ensemble est appelé fenêtre (window) et un arbre de décision est construit à partir de cette fenêtre. Cet arbre classe correctement tous les objets dans la fenêtre. On va classer le reste des objets en utilisant cet arbre. Si l'arbre donne des bonnes réponses pour tous les objets restants (en dehors de la fenêtre) alors l'arbre est correct pour l'ensemble d'apprentissage entier et le processus est terminé ; sinon les objets qui

sont mal classés sont ajoutés à la fenêtre et le processus continue, c'est-à-dire qu'un nouvel arbre de décision doit être construit.

De cette façon, un arbre de décision correct est trouvé après quelques itérations. Le principe de construction d'un arbre de décision est celui donné au début de ce chapitre, et le choix d'un attribut se base sur l'idée de maximisation du gain d'information. Dans le cas spécial où on a un ensemble C d'objets qui ne contient aucun objet pour une valeur a_i d'un attribut A , le sous-ensemble correspondant à cette valeur est vide. ID3 étiquette cette feuille comme nulle et le classement de chaque objet qui arrive à cette feuille échoue. Une solution est alors générée à partir de l'ensemble C consistant à associer la classe la plus fréquente dans C à cette feuille. ASSISTANT [37] arrête la construction sur un nœud lorsque tous ses attributs apportent un gain d'information inférieur à un seuil fixé.

4.5.2 .C4.5 :

C4.5 est une extension de l'algorithme de base d'ID3. Dans cette méthode, il y a deux phases : La première consiste à construire l'arbre de décision jusqu'au bout en divisant récursivement l'ensemble d'apprentissage. Une fois l'arbre construit, une phase d'élagage est appliquée. L'idée est d'élaguer les branches qui augmentent le taux d'erreurs de classement dans l'arbre de décision. Le processus d'élagage sera expliqué dans la suite. La fonction d'impureté utilisée par C4.5 pour construire un arbre de décision est le ratio du gain (équation (4)), qui consiste à diviser l'information mutuelle par l'entropie de l'attribut testé.

$$gainRatio(A, C) = \frac{IM(A, C)}{H(A)}$$

Contrairement à ID3, C4.5 traite le problème des attributs continus, le problème des valeurs manquantes ainsi que l'élagage d'un arbre de décision [31].

4.5.3. CART :

Cette méthode [38] permet d'inférer les arbres de décision binaires, c'est-à-dire que tous les tests étiquetant les nœuds de décision sont binaires. Les attributs peuvent être

- _ Binaires.
- _ Continus à valeurs réelles.
- _ Discrets ou qualitatifs, à valeurs dans un ensemble fini de modalités.

Un Split est un test (question à poser) sur un attribut choisi pour diviser l'ensemble d'apprentissage ; chaque split dépend des valeurs d'un seul attribut.

Le problème de construction d'un arbre de décision binaire est de savoir comment utiliser la base d'apprentissage pour déterminer les tests (splits) binaires. L'idée principale est de sélectionner chaque split de manière à ce que les données dans chaque sous-ensemble descendant soient plus homogènes que les données dans le nœud père.

_ Pour les attributs continus : toutes les questions possibles (splits) sont de la forme :

$$\{Is x_m \leq c \text{ ?} \} \quad c \in [-\infty, \infty]$$

_ Pour les attributs discrets et qualitatifs : toutes les questions possibles prennent la forme suivante :

$$\{Is x_m \in c \text{ ?} \} \quad \text{où } c \text{ est un sous-ensemble de } \{b_1, b_2, \dots, b_L\} \text{ par exemple.}$$

Ces questions binaires forment tous les splits possibles. Par exemple, si on a 4 attribut x_1 ; x_2 ; x_3 qui sont des attributs continus et $x_4 \in \{b_1, b_2, b_3\}$ alors toutes les questions possibles sont de la forme :

$$\begin{aligned} &Is x_1 \leq 3.2 \text{ ?} \\ &Is x_3 \leq -6.8 \text{ ?} \\ &Is x_4 \in \{b_1, b_2\} \text{ ? , etc.} \end{aligned}$$

Sur un nœud t , les instances qui répondent *oui* à une question posée sur ce nœud sont associées à la partie gauche de l'arbre et les instances qui répondent *non* à une question posée sur un nœud t sont associées à la partie droite de l'arbre. Le nombre de tests (splits) à explorer va dépendre de la nature des attributs :

_ À un attribut binaire correspond un test binaire.

_ À un attribut discret ou qualitatif ayant n modalités, on peut associer $2^n - 1$ tests binaires possibles parce que $\{x_m \in S\}$ et $\{x_m \notin S\}$ génèrent la même partition. Par exemple : si un attribut x_m prend ses valeurs dans l'ensemble $\{b_1; b_2; b_3\}$, alors on a 3 tests binaires possibles qui sont : $x_m \in \{b_1, b_2\}$, $x_m \in \{b_1, b_3\}$ et $x_m \in \{b_2, b_3\}$.

Les ensembles compléments sont respectivement : $x_m \in \{b_3\}, x_m \in \{b_2\}, x_m \in \{b_1\}$ Pour un attribut continu, il y a une infinité de tests envisageables. On découpe l'ensemble des valeurs possibles en segments ; ce découpage peut être fait par un expert ou de façon automatique.

L'algorithme de construction d'un arbre de décision binaire en utilisant CART parcourt, pour chaque nœud, les M attributs (x_1, x_2, \dots, x_m) un par un, en commençant par x_1 , et en continuant jusqu'à x_m . Pour chaque attribut, il explore tous les tests possibles (splits) et il choisit le meilleur split (dichotomie) qui maximise la réduction en impureté. Ensuite, il compare les M meilleurs splits pour choisir le meilleur d'entre eux. La fonction qui mesure l'impureté devra atteindre son maximum lorsque les instances sont équitablement réparties entre les différentes classes et son minimum lorsqu'une classe contient tous les exemples (le nœud est pur). Il existe différentes fonctions qui

satisfont ces propriétés ; la fonction utilisée par CART est la fonction de Gini (indice d'impureté de Gini). La fonction de Gini sur un nœud t avec une distribution de probabilités des classes sur ce nœud $P(j|t), j=1,...,J$ est :

$$G(t) = i(t) = \phi(p(1/t), p(2/t), ..., p(J/t)) = 1 - \sum_j P(j|t)^2 \quad (4)$$

Si un split s sur un nœud t partitionne le sous-ensemble associé à ce nœud en deux sous ensembles gauche t_G avec une proportion p_G et droite t_D avec une proportion p_D , on peut définir la mesure de réduction d'impureté comme suit :

$$\Delta i(s, t) = i(t) - p_G \times i(t_G) - p_D \times i(t_D)$$

Par conséquent, sur chaque nœud, si l'ensemble de splits candidats est S , l'algorithme cherche, le meilleur splits tel que :

$$\Delta i(s^*, t) = \max_{s \in S} \Delta i(s, t) \quad (5)$$

Supposons que nous avons obtenu quelques splits et que nous sommes arrivés à un ensemble de nœuds terminaux \tilde{T} . L'ensemble des splits, utilisés dans le même ordre, détermine l'arbre binaire T .

Nous avons $I(t) = i(t) p(t)$. Donc, la fonction d'impureté sur l'arbre est :

$$I(T) = \sum_{t \in \tilde{T}} I(t) = \sum_{t \in \tilde{T}} i(t) p(t)$$

$i(t)$ est la mesure d'impureté sur le nœud t et $p(t)$ est la probabilité qu'une instance appartienne au nœud t .

Il est facile de voir que la sélection des splits qui maximisent $\Delta i(s, t)$ est équivalente à la sélection des splits qui minimisent l'impureté $I(T)$ dans tout l'arbre. Si nous prenons n'importe quel nœud

$t \in \tilde{T}$ et nous utilisons un split s qui partitionne le nœud en deux parties t_D et t_G . Le nouvel arbre \hat{T} possède l'impureté suivante :

$$I(\hat{T}) = \sum_{\tilde{T}-\{t\}} I(t) + I(t_D) + I(t_G)$$

Parce que nous avons partitionné le sous-ensemble arrivé à t en t_D et t_G . Donc, la réduction de l'impureté de l'arbre est :

$$\begin{aligned} I(T) - I(\hat{T}) &= \sum_{t \in \tilde{T}} I(t) - \sum_{\tilde{T}-\{t\}} I(t) - I(t_D) - I(t_G) \\ &= I(t) - I(t_D) - I(t_G) \end{aligned}$$

Cela dépend seulement du nœud t et du split s . Donc, pour maximiser la réduction d'impureté dans l'arbre sur un nœud t , on maximise :

$$\Delta I(s, t) = I(t) - I(t_G) - I(t_D) \quad (6)$$

Les proportions t_D et t_G sont définies comme suit : $p_D = p(t_D)/p(t)$, $p_G = p(t_G)/p(t)$ et $p_G + p_D =$

1. Donc, (6) peut être écrite comme suit :

$$\Delta I(s, t) = [i(t) - p_G \times i(t_G) - p_D \times i(t_D)]p(t) = \Delta i(s, t)p(t)$$

Puisque $p(t)$ est la seule différence entre $\Delta i(s, t)$ et $\Delta I(s, t)$ le même split s^* maximise les deux expressions.

Le critère d'arrêt initial (stop splitting) utilisé par CART était très simple :

Pour un seuil $\beta > 0$, un nœud est déclaré terminal (feuille) si $\Delta I(s, t) \leq \beta$. L'algorithme associe à chaque nœud terminal la classe la plus probable.

4.5.4. Les critères d'arrêt de construction d'un arbre de décision :

Il y a plusieurs critères d'arrêt possibles lors de la construction d'un arbre de décision.

- _ Tous les objets appartiennent à la même classe ou il n'y a plus d'attribut à utiliser.
- _ La profondeur de l'arbre atteint une limite fixée (=nombre d'attributs utilisés).
- _ Le nombre de feuilles atteint un maximum fixé.
- _ Le nombre d'instances par nœud est inférieur à un seuil fixé.
- _ Le gain d'information maximum obtenu est inférieur à un seuil fixé.
- _ La qualité de l'arbre n'augmente plus de façon sensible. Par exemple, aucun attribut n'améliore la qualité de l'arbre.

Le fait de construire l'arbre jusqu'au bout selon le premier critère ci-dessus, jusqu'à ce qu'aucun attribut ne reste à utiliser ou tous les objets appartiennent à la même classe, favorise les feuilles ayant peu d'objets et diminue la fiabilité de l'arbre lors du classement d'un nouvel exemple.

Par conséquent, le taux d'erreurs de classement augmente. Le premier critère n'aide donc pas à construire l'arbre de décision le plus simple. Les autres critères aident à stopper la construction lorsqu'un seuil est atteint, comme le nombre d'instances par feuille, la profondeur de l'arbre, le nombre de feuilles dans l'arbre, etc. Ces critères se situent dans le cadre de l'*élégance* et plus précisément dans son premier type appelé *pré-élégance*. Les méthodes de l'*élégance* sont détaillées dans le paragraphe suivant [31].

4.6. Élagage :

Un des problèmes connus lors des phases de construction et de classement est que la taille de l'arbre grandit de manière linéaire avec la taille de la base d'apprentissage. De plus, les arbres de décision complexes peuvent avoir des taux d'erreur très élevés à cause du sur-ajustement (*overfitting*) qui peut se produire lorsque l'ensemble d'apprentissage contient des données bruitées ou qu'il ne contient pas certains exemples importants, ou encore lorsque les exemples sont trop spécifiques. L'élagage est une des solutions pour réduire ces taux d'erreurs en simplifiant l'arbre par suppression de quelques branches. Plusieurs techniques d'élagage [38] ont été proposées pour éviter le sur-ajustement. On distingue deux approches principales:

4.6.1. Pré-élagage :

Le pré-élagage a pour but d'arrêter la construction de l'arbre de décision à l'avance même si les feuilles ne sont pas pures ; c'est-à-dire qu'on décide ou non de continuer à développer un certain nœud. Cette étape, nommée également *stopping* [39], cherche à trouver la meilleure division pour un sous-ensemble d'apprentissage et à l'évaluer en utilisant par exemple le gain d'information ou d'autres mesures d'impureté ; si cette évaluation ne dépasse pas un certain seuil, alors cette division est rejetée et l'arbre pour ce sous-ensemble est la feuille la plus convenable en choisissant la classe la plus fréquente dans ce sous-ensemble. Mais un seuil suffisamment élevé pour supprimer les attributs non pertinents risque de supprimer aussi les attributs pertinents. Il est donc possible d'utiliser d'autres critères d'arrêt.

4.6.2. Post-élagage :

Dans cette approche, l'arbre de décision est simplifié en supprimant un ou plusieurs de ses sous-arbres et en les remplaçant par des feuilles. On construit l'arbre de décision jusqu'au bout et ensuite on l'élague. On estime les erreurs de classification à chaque nœud. Le sous-ensemble est remplacé par une feuille (classe) ou par la branche la plus fréquente. On commence par le fond de l'arbre et on examine chacun des sous-arbres (non-feuille) ; si le remplacement de ce sous-arbre par une feuille ou par sa branche la plus fréquente conduit à prévoir un taux d'erreur plus bas, dans ce cas, on élague le sous-arbre [31].

4.7. Classement :

Généralement, le processus du classement commence par la racine de l'arbre de décision, on descend dans l'arbre selon les valeurs des attributs nœuds dans l'objet à classer jusqu'à ce qu'on arrive à une feuille, et la classe associée à cette feuille est la classe de l'objet en question.

4.8. Applications courantes des analyses basées sur des arbres :

Voici quelques-unes des applications courantes des analyses basées sur des arbres :

- **Opérations de mailing.** Permet de déterminer quels groupes démographiques ont les taux de réponse les plus élevés. Ces informations sont utilisées pour maximiser les réponses aux futurs mailings.
- **Analyse de crédit.** Utilise l'historique de crédit d'un individu pour prendre des décisions relatives à l'accord d'un prêt.
- **Ressources humaines.** Analyse les pratiques de recrutement passées pour créer des règles de décision permettant d'affiner le processus de recrutement.
- **Recherche médicale.** Crée des règles de décision permettant de suggérer des procédures appropriées en se basant sur des preuves médicales.
- **Etude de marché.** Détermine les variables pouvant être mises en relation avec les ventes telles que la géographie, le prix et les caractéristiques de la clientèle.
- **Contrôle qualité.** Analyse les données issues de la fabrication des produits et identifie les variables déterminant les défauts de ceux-ci.
- **Etudes de stratégies.** Utilise les données issues de sondages pour élaborer une stratégie en utilisant des règles de décision pour sélectionner les variables les plus importantes.
- **Santé.** Combine les données issues de questionnaires et d'études cliniques pour découvrir les variables favorisant la santé [31].

4.9. Avantages et inconvénients

L'induction par arbres de décision est une technique arrivée à maturité ; ses caractéristiques, ses points forts et ses points faibles sont maintenant bien connus ; il est possible de la situer précisément sur l'échiquier des très nombreuses méthodes d'apprentissage (Hastie *et al.*, 2001).

Les arbres présentent des performances comparables aux autres méthodes supervisées ; les nombreuses comparaisons empiriques l'ont suffisamment montré (Zighed et Rakotomalala, 2000 ; Lim *et al.*, 2000). La méthode est non paramétrique ; elle ne postule aucune hypothèse a priori sur

la distribution des données ; elle est résistante aux données atypiques ; le modèle de prédiction est non linéaire. Lorsque la base d'apprentissage est de taille importante, elle présente des propriétés similaires aux algorithmes des plus proches voisins (Breiman *et al.*, 1984).

Il faut néanmoins tempérer ce constat. Le premier reproche qu'on peut lui adresser est son incapacité, avec les algorithmes classiques (C4.5, CART, CHAID, etc.), à détecter les combinaisons de variables ; ceci est dû au principe de construction pas à pas de l'arbre, entraînant une certaine « myopie ». Le second reproche est dans la nécessité de disposer d'un échantillon d'apprentissage de grande taille. L'arbre certes peut reproduire approximativement toutes formes de frontières, mais au prix d'une fragmentation rapide des données, avec le danger de produire des feuilles avec très peu d'individus. Corollaire à cela, les arbres sont en général instables ; les bornes de discrétisation notamment dans les parties basses de l'arbre sont entachées d'une forte variabilité. Ainsi, certains chercheurs préconisent de procéder à la discrétisation préalable des variables avant la construction de l'arbre (Dougherty *et al.*, 1995) [31].

L'induction par arbre de décision est capable de traiter de manière indifférenciée les données Continues et discrètes. Elle dispose de plus d'un mécanisme naturel de sélection de variables. Elle doit être privilégiée lorsque l'on travaille dans des domaines où le nombre de descripteurs est élevé, dont certains, en grand nombre, sont non-pertinents. Nous devons également relativiser cette affirmation. En effet, non sans surprise, des travaux dans le domaine de la sélection de variables ont montré que la réduction préalable des descripteurs dans des domaines fortement bruités améliorerait considérablement les performances des arbres de décision (Yu et Liu, 2003). Il y a principalement deux causes à cela : à force de multiplier les tests, l'algorithme multiplie également le risque d'introduire des variables non significatives dans l'arbre. Ce risque est d'autant plus élevé que les méthodes comme C4.5, très utilisées dans la communauté de l'apprentissage automatique, adoptent la construction « hurdling » (introduire une variable même si elle induit un gain nul) en misant, parfois à tort, sur le post-élagage pour éliminer les branches non-pertinentes de l'arbre.

Enfin, dernier point de différenciation, qui assure en grande partie la popularité des arbres auprès des praticiens : leur capacité à produire une connaissance simple et directement utilisable, à la portée des non-initiés. Un arbre de décision peut être lu et interprété directement ; il est possible de le traduire en base de règles sans perte d'information. A la fin des années 80, on considérait que cette méthode assurait le renouveau des systèmes experts en éliminant le goulot d'étranglement que constitue le recueil des règles (Kononenko, 1993). Cette qualité est renforcée par la possibilité qu'a l'expert d'intervenir directement dans le processus de création du modèle de prédiction.

L'appropriation de l'outil par les experts du domaine assure dans le même temps une meilleure interprétation et compréhensibilité des résultats [31].

4.10. Conclusion

L'induction des arbres de décision a été la coqueluche des chercheurs dans les années 90 : les références citées dans ce didacticiel sont assez édifiantes. Ses propriétés sont maintenant bien connues et, si les tentatives pour faire évoluer la méthode sont moins nombreuses aujourd'hui, elle se positionne surtout comme une méthode de référence. Les articles proposant de nouvelles techniques d'apprentissage l'utilisent souvent dans leurs comparatifs pour situer leurs travaux. La méthode préférée en apprentissage automatique est certainement C4.5 ; la disponibilité du code source sur Internet n'est pas étrangère à ce succès.