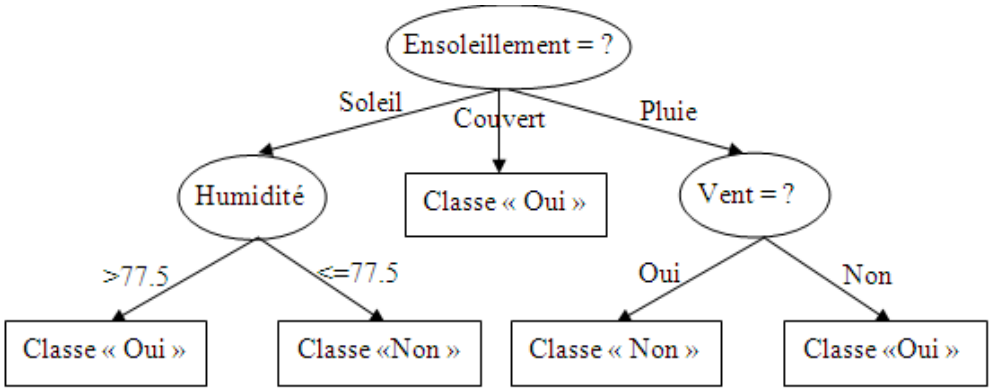
3.4 Arbres de décision

Les arbres de décision représentent une méthode très efficace d'apprentissage supervisé. Il s'agit de partitionner un ensemble de données en des groupes les plus homogènes possible du point de vue de la variable à prédire. On prend en entrée un ensemble de données classées, et on fournit en sortie un arbre qui ressemble beaucoup à un diagramme d'orientation où chaque nœud final (feuille) représente une décision (une classe) et chaque nœud non final (interne) représente un test. Chaque feuille représente la décision d'appartenance à une classe des données vérifiant tous les tests du chemin menant de la racine à cette feuille. L'exemple suivant montre un ensemble de données avec quatre attributs : Ensoleillement, Température, Humidité, Vent et l'attribut à prédire Jouer.

N°	Ensoleillement	Température	Humidité	Vent	Jouer
1	Soleil	75	70	Oui	Oui
2	Soleil	80	90	Oui	Non
3	Soleil	85	85	Non	Non
4	Soleil	72	95	Non	Non
5	Soleil	69	70	Non	Oui
6	Couvert	72	90	Oui	Oui
7	Couvert	83	78	Non	Oui
8	Couvert	64	65	Oui	Oui
9	Couvert	81	75	Non	Oui
10	Pluie	71	80	Oui	Non
11	Pluie	65	70	Oui	Non
12	Pluie	75	80	Non	Oui
13	Pluie	68	80	Non	Oui
14	Pluie	70	96	Non	Oui

L'arbre appris à partir de cet ensemble de donnée est le suivant :



En effet, toutes les données ayant l'attribut Ensoleillement="Soleil" et l'attribut Humidité>77.5 appartiennent à la classe 1 ("oui"). Toute nouvelle donnée peut être classée en testant ses valeurs d'attributs l'un après l'autre en commençant de la racine jusqu'à atteindre une feuille c'est-à-dire une décision. Pour construire un tel arbre, plusieurs algorithmes existent : ID3, CART, C4.5,...etc. On commence généralement par le choix d'un attribut puis le choix d'un nombre de critères pour son nœud. On crée pour chaque critère un nœud concernant les données vérifiant ce critère. L'algorithme continue d'une façon récursive jusqu'à obtenir des nœuds concernant les données de chaque même classe.

Algorithme 3 CONSTRUIRE-ARBRE(D : ensemble de données)

1: Créer nœud N
2: **Si** tous les exemples de D sont de la même classe C alors
3: Retourner N comme une feuille étiquetée par C ;
4: **Si** la liste des attributs est vide alors
5: Retourner N Comme une feuille étiquetée de la classe de la majorité dans D ;
6: Sélectionner l'attribut A du meilleur Gain dans D ;
7: Etiqueter N par l'attribut sélectionné ;
8: Liste d'attributs ← Liste d'attributs - A ;
9: **Pour** chaque valeur V_i de A
10: Soit D_i l'ensemble d'exemples de D ayant la valeur de $A = V_i$;
11: Attacher à N le sous arbre généré par l'ensemble Di et la liste d'attributs
12: **FinPour ;**
13: **Fin ;**

En réalité ce n'est pas si simple, plusieurs problèmes doivent être résolus :

– Comment choisir l'attribut qui sépare le mieux l'ensemble de données ? On parle souvent de la variable de segmentation.

– Comment choisir les critères de séparation d'un ensemble selon l'attribut choisi, et comment ces critères varient selon que l'attribut soit numérique ou symbolique ?
– Quel est le nombre optimal du nombre de critères qui minimise la taille de l'arbre et maximise la précision ?
– Quels sont les critères d'arrêt de ce partitionnement, sachant que souvent l'arbre et d'une taille gigantesque ?

3.4.1 Choix de la variable de segmentation

Il s'agit de choisir parmi les attributs des données, celui qui les sépare le mieux du point de vue de leurs classes déjà connues. Pour choisir le meilleur attribut, on calcule pour chacun une valeur appelée "Gain" qui dépend des différentes valeurs prises par cet attribut. Cette mesure est basée sur les recherches en théorie d'informations menées par C.Shannon.

Par exemple, l'algorithme ID3 utilise le concept d'entropie introduite initialement par Shannon en 1948.

Soit un ensemble X d'exemples dont une proportion p_+ sont positifs et une proportion p_- sont négatifs. (Bien entendu, $p_+ + p_- = 1$) L'entropie de X est :

$$H(X) = \text{-}p_+ log_2(p_+) \text{-} p_- log_2(p_-)$$

Biensur

$$0 \leq H(X) \leq 1$$

Si $p_+ = 0$ ou $p_- = 0$, alors $H(X) = 0$. Ainsi, si tous exemples sont soit tous positifs, soit tous négatifs, l'entropie de la population est nulle. Si $p_+ = p_- = 0.5$, alors $H(X) = 1$. Ainsi, s'il y a autant de positifs que de négatifs, l'entropie est maximale.

$$Gain(X, a_j) = H(X) \text{-} \sum_{v \in valeurs(a_j)} \frac{|X_{a_j=v}|}{|X|} H(X_{a_j=v})$$

Où $X_{a_j=v}$, est l'ensemble des exemples dont l'attribut considéré a_j prend la valeur v, et la notation $|X|$ indique le cardinal de l'ensemble X.

Exemple

Le Gain du champs "Vent" de la table précédente est calculé comme suit :

$$
\begin{aligned}
Gain(X, vent) &= H(X) - \tfrac{9}{14}H(X_{a=oui}) - \tfrac{5}{14}H(X_{a=non}) \\
On\ a : & \\
H(X) &= -\tfrac{5}{14}ln_2\tfrac{5}{14} - \tfrac{9}{14}ln_2\tfrac{9}{14} = 0.940 \\
H(X_{a=non}) &= -(\tfrac{6}{8}ln_2\tfrac{6}{8} + \tfrac{2}{8}ln_2\tfrac{2}{8}) = 0.811 \\
Et & \\
H(X_{a=oui}) &= -(\tfrac{3}{6}ln_2\tfrac{3}{6} + \tfrac{3}{6}ln_2\tfrac{3}{6}) = 1.0 \\
D'où : & \\
Gain(X, vent) &= 0.940 - \tfrac{9}{14} * 0.811 - \tfrac{5}{14} * 1.0 \\
&= 0.048
\end{aligned}
$$

Le principe de l'algorithme ID3 pour déterminer la variable de segmentation est de prendre la variable du gain d'information maximum.

3.4.2 Choix de la bonne taille de l'arbre

Une fois l'arbre de décision construit, il peut contenir plusieurs anomalies qui peuvent être dues au bruit ou aux valeurs extrêmes, et qui peuvent conduire au problème de sur-apprentissage (overfitting). Ce problème est la déduction d'informations plus que supporte l'ensemble de données d'apprentissage. L'arbre peut être aussi d'une taille très importante qui peut épuiser les ressources de calcul et de stockage. Pour surmonter ce problème, on effectue des opérations d'élagage qui consistent à éliminer de l'arbre les branches les moins significatives (qui déduisent d'un nombre réduit d'enregistrements ou de ceux qui appartiennent à diverses classes). L'élagage peut être effectué avant ou après l'apprentissage, on parle souvent de pré et post-élagage :

- Pré-élagage : effectué lors de la construction de l'arbre, lorsqu'on calcule les caractéristiques statistiques d'une partie des données tel que le gain, on peut décider de l'importance ou non de sa subdivision, et ainsi on coupe complètement des branches qui peuvent être générée.
- Post-élagage : effectué après la construction de l'arbre en coupant des sous arbres entiers et en les remplaçant par des feuilles représentant la classe la plus fréquente dans l'ensemble des données de cet arbre. On commence de la racine et on descend, pour chaque nœud interne (non feuille), on mesure sa complexité avant et après sa coupure (son remplacement par une feuille), si la différence est peu importante, on coupe le sous arbre et on le remplace par une feuille.

3.4.3 Algorithmes de construction d'arbres de décision

3.4.3.1 Algorithme ID3

ID3 construit l'arbre de décision récursivement. A chaque étape de la récursion, il calcule parmi les attributs restant pour la branche en cours, celui qui maximisera le gain d'information. C'est-à-dire l'attribut qui permettra le plus facilement de classer les exemples à ce niveau de cette branche de l'arbre. Le calcul ce fait à base de l'entropie de Shanon déjà présentée. L'algorithme suppose que tous les attributs sont catégoriels ; si des attributs sont numériques, ils doivent être descritisés pour pouvoir l'appliquer. ID3 utilise l'algorithmes *Construire_ arbre* précédent.

3.4.3.2 Algorithme C4.5 (J48)

C'est une amélioration de l'algorithme ID3, il prend en compte les attributs numérique ainsi que les valeurs manquantes. L'algorithme utilise la fonction du gain d'entropie combiné avec une fonction *SplitInfo* pour évaluer les attributs à chaque itération.

3.4.3.2.1 Attributs discrets 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 A$\in${a,b} et A$\in${c,d} ; le test ternaire A=a , A=c et A$\in${b,d} ; ...

3.4.3.2.2 Attributs continus Pour les attributs continus, la discrétisation peut être laissée à un expert du domaine d'application. Par exemple, en médecine, l'expérience du domaine peut avoir permis la mise en évidence l'existence de valeurs seuil pour un attribut correspond à une mesure médicale. Sinon, l'algorithme gère les attributs continus de la façon suivante : les exemples sont triés dans l'ordre croissant pour l'attribut continu A considéré, on considère alors tous les tests de la forme A$>a_i + a_{i+1}/2$ où a_i et a_{i+1} sont deux valeurs consécutives de l'attribut A. Par exemple, supposons que A prenne les valeurs 1 ; 3 ; 6 ; 10 ; 12, alors on considère les tests A$>$2 ; A$>$4,5 ; A$>$8 et A$>$11, ces tests participent alors à la compétition dans la recherche du test apportant le meilleur gain (fonction Gain ou GainRatio, selon l'option choisie).

3.4.3.2.3 Attributs à valeurs manquantes Dans de nombreux problèmes concrets, il existe certains attributs dont les valeurs ne sont pas renseignées. Par exemple, si on dispose du descriptif de patients, il est très probable que toutes les mesures ne soient pas disponibles car elles n'ont pas pu être faites pour tous les patients. Pour classifier un exemple possédant des valeurs manquantes à l'aide d'arbres de décision, on procède comme dans le cas standard, lorsque l'on rencontre un test et que la valeur de l'attribut est manquante, on considère la branche majoritaire. Pour la phase d'apprentissage, on suppose que la valeur de cet attribut suit la distribution des valeurs connues.

3.4.3.3 Algorithme CART

L'lgorithme CART dont l'acronyme signifie "Classification And Regression Trees", construit un arbre de décision d'une manière analogue à l'algorithme ID3. Contrairement à ce dernier, l'arbre de décision généré par CART est binaire et le critère de segmentation est l'indice de Gini.

À un attribut binaire correspond un test binaire. À un attribut qualitatif ayant n modalités, on peut associer autant de tests qu'il y a de partitions en deux classes, soit 2n-1 tests binaires possibles. Enfin, dans le cas d'attributs continus, il y a une infinité de tests envisageables. Dans ce cas, on découpe l'ensemble des valeurs possibles en segments, ce découpage peut être fait par un expert ou fait de façon automatique.

3.4.3.4 Forêts aléatoires

Les forêts aléatoires ont été inventées par Breiman en 2001 ([Bre01]). Elles sont en général plus efficaces que les simples arbres de décision mais possède l'inconvénient d'être plus difficilement interprétables. Leur construction se base sur le bootstrap (ou le bagging). On subdivise l'ensemble de données en plusieurs parties par le bootstrap puis on apprend un arbre de décision à partir de chaque partie. Un nouvel exemple est testé par tous les arbres construits et sa classe est la classe majoritaire.