

# Introduction à Data science

B.E. 1 : Manipulation des méthodes d'apprentissage avec WEKA

ECL - 3A - 2024/2025

octobre 2024

## Table des matières

<b>1</b>	<b>Pour 24-25</b>	<b>3</b>
<b>2</b>	<b>Utilisation de Weka</b>	<b>3</b>
2.1	Chargement d'un fichier de données	3
2.2	Prétraitement	3
2.3	Exercice : application d'une méthode basique 1R	3
2.4	Résultats de la méthode appliquée	3
2.5	Exercice	5
<b>3</b>	<b>Arbre de décision</b>	<b>6</b>
3.1	Application d'une meilleure méthode (arbre de décision par ID3)	6
3.2	Application d'une autre méthode (arbres de décision C4.5)	6
<b>4</b>	<b>A propos des choix de la méthode du test</b>	<b>8</b>
<b>5</b>	<b>Arbre de classification par la méthode CART</b>	<b>8</b>
5.1	Exercice	8
<b>6</b>	<b>Autres remarques sur Weka</b>	<b>8</b>
6.1	Autres formats de fichiers de données	9
<b>7</b>	<b>Exercices : la pratique de Weka sur l'exemple Banque</b>	<b>9</b>
7.1	Banque : choix d'attribut	9
7.2	Exercice : trouver le meilleur attribut à la main	9
7.3	Exercice : arbre de décision de l'exemple Banque	9
7.4	Exercice (optionnel) : règles sur exemple météo	10
7.5	Autres Exemples de manipulation	10
<b>8</b>	<b>Travail 1</b>	<b>10</b>
<b>9</b>	<b>Travail 2 : Méthode Bayésienne</b>	<b>11</b>
9.1	Exemple Haggis	11
<b>10</b>	<b>La méthode Bayésienne sous WEKA (Spams)</b>	<b>15</b>
10.1	Travail à rendre (3) : Bayésienne sur la BD Spam	15
10.2	Rappel : paramètres de Bayésienne Naïve	16
<b>11</b>	<b>Remarque sur les réseaux de Bayes</b>	<b>16</b>
<b>12</b>	<b>Évaluation des modèles : quelques éléments</b>	<b>18</b>
12.1	A propos des mesures Kappa	18
12.2	Matrice de Confusion de classification non binaire	20
12.3	Comparaison des méthodes	20
<b>13</b>	<b>Annexes-1 : Rappels du cours sur les Arbres de décision</b>	<b>23</b>
13.1	Définition	23
13.2	Induction d'arbres de décision et Entropie	23
13.2.1	Critères de l'algorithme	23
13.3	Discrimination d'attributs et la fonction Entropie	24
13.4	Exercice de construction d'un arbre de décision	25
13.5	Exercice optionnel : créez l'arbre!	25
13.6	Rappel Méthode ID3 et le calcul de l'Entropie	25
13.7	L'algorithme C4.5	26
13.8	Conclusions sur les arbres de décision	26
<b>14</b>	<b>Annexes-2 : exploitation de l'extracteur WEKA</b>	<b>27</b>
14.1	Format de fichiers	27
14.2	Préparation des données sous Weka	27
14.3	Suppression des instances par leur indice	28
14.4	Anonymisation de données sous Weka	28
14.5	Discretisation dans Weka	28
14.5.1	Un exemple de discrétisation	28
14.6	Conversion des données	29
14.7	Conversions de données transactionnelles	29
14.8	Conversions numériques vers nominales	30
14.9	Autres méthodes de préparation / conversion de données	30
14.10	Échantillonnage des données	31
14.11	Échantillonnage de x%	31
14.12	Échantillonnage selon valeur d'attribut	31
14.13	Filtre suivant les attributs	32
14.14	supprimer les instances avec "missing values"	32

<b>15 Annexes-3 : Compléments</b>	<b>34</b>
15.1 A propos de C4.5	34
15.2 Classification Bayésienne (compléments)	35
15.2.1 Classe majoritaire $C_{maj}$	35
15.2.2 Maximum de vraisemblance $C_{mv}$	35
15.2.3 Classification bayésienne $C_{Bayes}$	35
15.2.4 Qualité de l'approximation	35
15.3 Élagage : en savoir plus	36
15.3.1 A propos d'élagage dans C4.5	36

# 1 Pour 24-25

Utiliser la biblio python "chefboost" qui implante les AD (bien).

## 2 Utilisation de Weka

Voir la section 14 en page 27 pour le téléchargement et le lancement de weka.

Voir également ci-dessous (exercice suivant) pour le téléchargement des deux packages nécessaires.

☞ Pour un rappel sur les arbres de décision, voir cours et la section 13 en page 23 de ce document.

Le principe d'utilisation basique de Weka est le suivant :

1. Charger un fichier de données en mémoire (les données préparées, mises au format accepté par *weka*).
2. Répéter Jusqu'au chargement d'un autre fichier de données
  - (a) Éventuellement procéder à un pré-traitement (par exemple, discrétisation)
  - (b) Appliquer une méthode d'Extraction de Connaissances
  - (c) Visualiser (sous différentes formes) les résultats, les sauvegarder éventuellement ...

### 2.1 Chargement d'un fichier de données

Après le lancement de Weka, dans la première fenêtre de WEKA, cliquez sur **Explorer** (ou *Application* → *Explorer*) : cela fait apparaître une interface composée d'onglets.

Pour ouvrir (et charger en mémoire) un fichier de données, aller sur l'onglet **Preprocess** et cliquez sur **Open file...**, et choisissez votre fichier de données "météo" **weather.nominal.arff** (situé éventuellement dans le répertoire **data**). Le contenu de cette base de données est rappelé dans la section 7.1 en page 9.

### 2.2 Prétraitement

Rien dans ce 1er exemple.

### 2.3 Exercice : application d'une méthode basique 1R

☞ A faire ! :

La version de développement de Weka n'intègre pas toutes les méthodes (comme le fait la version stable mais antérieure). Pour la suite, il faut charger deux *packages* sous Weka.

Dans la toute première fenêtre ouverte par Weka ("*WekaGuiChooser*"), choisir le menu "*Tools*" puis "*Package Manager*" et dans la fenêtre qui s'affiche après quelques instants, sélectionner "*simpleEducationalLearningSchemes*" et "*simpleCART*" qui contiennent les méthodes utilisées dans ce BE.

Commençons par l'application d'une méthode basique de classification (la méthode 1R vue en cours).

- Cliquer sur l'onglet **Classify** puis sur le bouton **Choose** ;
- Dans l'onglet **rules** qui se déroule, choisissez **OneR** ;
- Prenez note que le bouton **Cross-validation** avec **Folds = 10** est par défaut activé. Cela désigne la méthode d'évaluation des résultats (voir cours).
- Prenez également note que sous le bouton **More Options ...**, soit affiché (**Nom**) **play**. Cela signifie que l'attribut **play** de type nominal sera la classe (la conclusion=la décision) des règles produites.
- Appuyer à présent sur le bouton **Start** qui appliquera la méthode choisie (1R) aux données en mémoire.
- Vous constaterez **OK** en bas de la fenêtre (sous Status) signalant la fin du traitement avec l'affichage des résultats dans la partie droite de la fenêtre (fenêtre *Classifier Output*).

### 2.4 Résultats de la méthode appliquée

Avant de voir les résultats, quelques mots sur la matrice de confusion.

Pour la plupart des méthodes, Weka produit une **matrice de confusion** (ici pour 2 classes) où les cases contiennent des entiers : nombre d'exemples concernés.

**TP** (true positive) : *Les exemples positifs bien classés* : on connaît la classe = oui et l'outil a bien classé ces exemples.

**FN** (false negative) : *Les exemples positifs mal classés* : on connaît la classe = oui et l'outil a mal classé ces exemples.

**TN** (true negative) : *Les exemples négatifs bien classés* : on connaît la classe = non et l'outil a bien classé ces exemples.

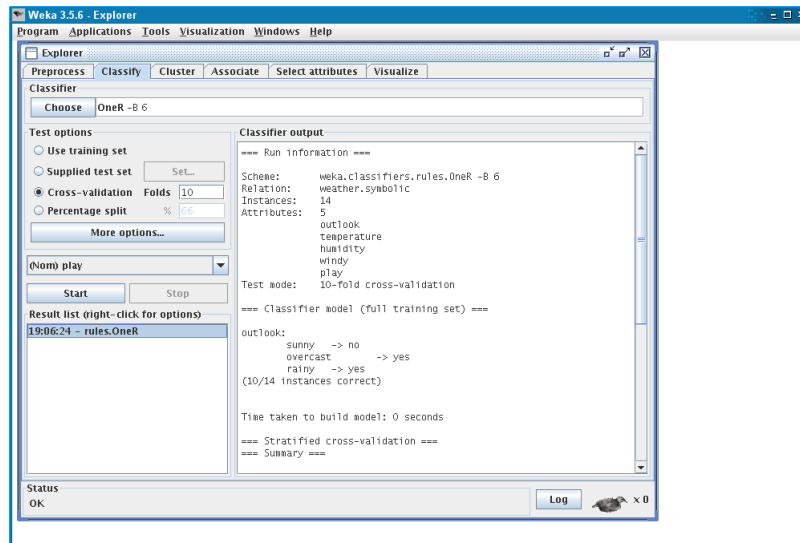
○ **FP** (false positive) : *Les exemples négatifs mal classés* : on connaît la classe = non et l'outil a mal classé ces exemples.

	classé positif	classé négatif
vraiment positif	<i>TP</i>	<i>FN</i>
vraiment négatif	<i>FP</i>	<i>TN</i>
totaux	<i>P</i>	<i>N</i>

→ La diagonale  $TP + TN$  donne le nombre d'instances bien classées.

Ce qui donnera pour notre exemple les résultats ci-dessous.

La fenêtre des résultats résume les valeurs des paramètres puis donne les détails et conclusions ainsi que quelques statistiques avant de fournir une matrice de confusion.



=== Summary ===

Correctly Classified Instances	6	42.8571 %
Incorrectly Classified Instances	8	57.1429 %
Kappa statistic	-0.1429	
Mean absolute error	0.5714	
Root mean squared error	0.7559	
Relative absolute error	120 %	
Root relative squared error	153.2194 %	
Total Number of Instances	14	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.444	0.6	0.571	0.444	0.5	0.422	yes
0.4	0.556	0.286	0.4	0.333	0.422	no

=== Confusion Matrix ===

```
a b  <-- classified as
4 5 | a = yes
3 2 | b = no
```

Dans la fenêtre des résultats, noter le taux (élevé) d'erreur ( > 57% ) ! Mieux vaut deviner les classes au pile-ou-face ; voire, prendre ces règles à contre pieds !

Différents mesures d'évaluation sont données. Voir cours pour ces valeurs. Pour le moment, on se contente du taux de classement. Notez la matrice de confusion tout en bas des résultats.

**Résumons cette étape :** la méthode 1R a été appliquée aux données "météo" (données de type nominal). Les résultats non probants : l'exploration par 1R ne suffit pas à caractériser cette BD (et à l'apprendre). La relation entre les attributs de cette BD n'est donc pas triviale.

Dans la partie **Test options** de la fenêtre Weka, si vous activez le bouton **Use training set**, vous aurez de meilleurs résultats (taux d'erreur de  $< 29\%$ ) mais vous risquez la sur-adaptation du modèle (overfitting, *appris par-cœur*) car le modèle aura été construit ET testé sur le même ensemble de données ; elle n'a pas été testée sur des données indépendantes de l'apprentissage. Les résultats sont alors très peu fiables. Voir aussi la section 4, page 8

Le principe de *la plus simple d'abord* (simplest-first) n'a pas payé sur ces données !

## 2.5 Exercice

Appliquez les méthodes 0R et 1R à la base de données "banque2-app.arff" et consignez les résultats. Testez vos modèles à la base de données "banque2-test.arff" et consignez les résultats (voir la section 4, page 8).

## 3 Arbre de décision

### 3.1 Application d'une meilleure méthode (arbre de décision par ID3)

Conservons les mêmes données mais appliquons la méthode de production d'arbre de décision **ID3**.

Dans l'onglet **Classify**, cliquez sur **Choose** et dans l'onglet **trees**, choisissez **id3**<sup>1</sup>

Réactivez le bouton **Cross-validation** avec **Folds = 10** et cliquez sur le bouton **Start**.

**Observez les résultats affichés** dans la partie droite de la fenêtre. Vous pouvez y lire les règles de classifications qui ont été produites à partir de l'arbre de décision crée. Le taux d'erreur est  $< 15\%$ .

Rappelons que ID3 utilise l'entropie (de *Shanon*) pour trouver les attributs discriminants .



**Weka ne permet pas** de visualiser l'arbre de décision de la méthode **ID3**.

Prenez note que vous pouvez re-consulter les résultats des méthodes précédentes (fenêtre *Result List*), voire ré-appliquer une méthode si de nouvelles données de test (par le bouton "supplied test set" pour charger un fichier de test, expliqué en section 4) ont été chargées en mémoire.

**N.B. :** ID3 peut donner des instances non classées (*Unclassified instances*). L'exemple Banque (précédent) en produira. Ce sont les instances dont ID3 n'a pas décidé de la classe.

Cela arrive par exemple quand on épuise les attributs mais qu'il reste encore des instances (dans une feuille) dont la classe ne peut pas être décidée.

Typiquement, cela arrive dans un cas bi-classes avec un nombre pair d'instances, moitié dans chaque classe (dans les noeuds de l'arbre). Mais le cas peut arriver dans d'autres situations (multi-classes, nombre quelconque d'instances concernées, ...). L'attribution d'une classe majoritaire pourrait être dans certains cas une solution.

### 3.2 Application d'une autre méthode (arbres de décision C4.5)

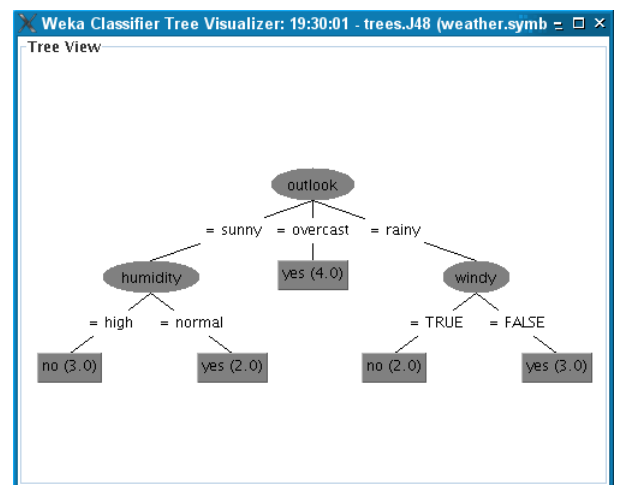
Choisissez la méthode **j48** dans l'onglet **trees**, au dessus d'ID3.

Il s'agit d'une amélioration d'ID3 (voir cours Chap 4) et en plus, on peut visualiser l'arbre de décision crée.

Après l'application de la méthode **j48**, dans la petite fenêtre à gauche (zone au dessous **Start**, dans la petite fenêtre *Result list*), cliquez avec le bouton droit sur la ligne de la méthode et choisissez **visualize tree**.

Dans la fenêtre de l'arbre, les boutons de la souris ont un sens. Vous disposez ainsi de l'arbre de décision qui donne l'origine des règles qui en ont été déduites.

Rappel : J48 est l'implantation de la méthode C4.5 (en Java, d'où le J ; et la version est passée du 4.5 d'origine à 4.8). J48 utilise le *ratio de gain* qui est le ratio entre l'entropie et le degré de dispersion dans les noeuds (SplitInfo) d'un attribut (section 13.7).



Manipuler les différents paramètres de la méthode **j48** : la confiance est en rapport avec l'élagage de l'arbre de décision : on élague un sous arbre (en affectant une classe majoritaire aux instances concernées) si le taux d'erreur résultant de l'élagage ne dépasse pas  $1 - \text{valeur\_de\_confiance}$ .

**A propos de la méthode de test** (cadre "Test options" à gauche) :

pour un apprentissage plus fiable, on devrait partager les données en 3 ensembles d'apprentissage (la majeure partie des données), de validation et de test. Plus fréquemment (ou si peu de données), on ne crée que les ensembles *Apprentissage* et *Test*. On apprend (un modèle) avec l'ensemble d'apprentissage et pour valider le modèle, on *teste* le modèle appris sur l'ensemble de *test*.

Dans la méthode X-V avec 10 plis (folds), on fait 10 itérations et à chaque itération, on apprend sur 90% et on teste sur les 10% ; un tour complet des données nécessitera ainsi de répéter 10 fois ces découpages.

Il y a différentes façons de procéder à ce découpage. Voir aussi plus loin.

1. Si vous travaillez avec la version 3.8 ou plus de Weka et que vous avez déjà installé des packages dans une version antérieure de Weka, vous risquez une erreur de la part du **PackageManager** et pour résoudre ce problème : supprimez le fichier **installedPackageCache.ser** dans **Votre\_rép\_Home/wekafiles/packages/** et vérifiez que vous avez les packages *"simpleEducationalLearningSchemes"* et *"simpleCART"* pour ce BE.

## Les résultats de J48 :

```
J48 pruned tree
-----
outlook = sunny                ← A noter : pas d'erreur dans cet arbre.
| humidity = high: no (3.0)
| humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
| windy = TRUE: no (2.0)
| windy = FALSE: yes (3.0)

Number of Leaves : 5

Size of the tree : 8

Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 7 50                ← Pourtant : 50% d'erreur.
Incorrectly Classified Instances 7 50
```

Les résultats de l'application de la méthode j48 avec une méthode d'évaluation 10-XV (10-Folds-Cross-Validation) sont rappelés ci-dessus. On constate que les règles (= traduction de l'arbre de décision de la page précédente) permettent de décider SANS erreur de la classe. Or, le taux d'erreur est de 50%. **Pourquoi ?**

La raison est que le taux d'erreur est celui obtenu pendant la validation (donc a posteriori) et non pendant l'apprentissage de l'arbre. Cela veut dire que pendant les itérations de XV, il y a eu 1 cas sur 2 où l'instance de test (en admettant que 10% des 14 instances=1 instance) a été mal classée.

Il faut se rappeler que l'arbre de décision est construit A LA FIN des validations en faisant en quelque sorte la moyenne des validations croisées (X-V). Il se trouve ici que les instances "tombent bien" dans l'arbre final; *il n'empêche que Weka nous dit ce qui s'est passé pendant la création du modèle.*

☞ Weka nous met donc en garde en indiquant que le modèle obtenu final risque de produire des erreurs dans un cas sur 2, nonobstant sa parfaite adaptation à la BD en main.

Pour s'en convaincre, il faudrait pouvoir étudier chacune des XV pour voir à quel moment l'erreur a été visible.

Si l'on essaie une méthode de validation différente, le taux d'erreur peut changer radicalement. Avec *training-set* le taux tombe à 0 mais **gare à l'apprentissage par-coeur!** Et avec *Percentage Split* (66%) ?

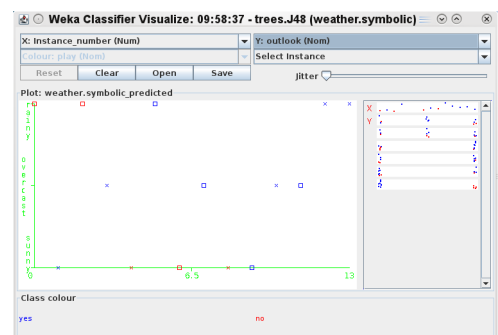
### Remarque :

Dans les mêmes conditions, après avoir appliqué J48, visualisez les erreurs (figure ci-contre) avec une clique droit sur votre *trees.J48* dans le cadre *Result list*.

Ici, les carrés en bleu sont les exemples positifs mal classés (false negative) et en rouge les false positives.

Quand on clique sur l'un de ces carrés, on voit s'afficher "classe prédite".

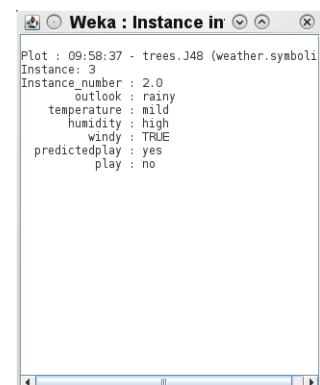
Cette dénomination fait référence à la prédiction pendant les XV et non d'après l'arbre de décision finale.



Notons également que le n° de l'instance affiché ne correspond pas à ceux de la BD originale mais à une numérotation interne à WEKA pendant les XV.

→ Dans la figure ci-contre, on a cliqué sur un des (petits) carrés rouges (correspond à la classe "No"). Pour Weka, l'instance est numérotée 3 mais ce n'est pas forcément la 3e instance de notre BD.

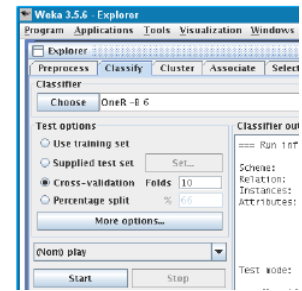
☞ Avec Weka version 3.8 ou 3.9, si la méthode de test a été X-V, le No. des instance s'obtient avec  $X : Prediction\ Margin(Num)$ . Pour une méthode de test différente, si vous ne voyez pas les numéros d'instance ( $Num$ ), choisissez alors en X et en Y d'autres attributs : par ex.  $X : predicted\ play$  et  $Y : play$ . etc.



## 4 A propos des choix de la méthode du test

L'option **Supplied test set** de la zone *Test options* permet de fournir un ensemble de test (dans un fichier arff).

- A chaque fois que cela est possible, il vaut mieux apprendre sur un ensemble et tester le modèle obtenu sur un ensemble de test, faute de quoi on peut obtenir un modèle qui convient mais ne se comporte pas très bien en phase d'application.
- Si un ensemble de test n'est pas disponible, on peut retirer de manière aléatoire une partie des données (10% .. 33%) pour les tests.
- Si l'on répète 10 fois le retrait des 10%, on aura la méthode d'évaluation *10 folds cross-validation*. Avec 33%, on aura une méthode d'évaluation 3-folds.
- L'option **Percentage split** permet de fixer la partie (2/3 par défaut) des données utilisée pour l'apprentissage et le reste pour le test.



## 5 Arbre de classification par la méthode CART

Rappel : ID3 utilise l'entropie; C4.5 utilise entropie et *splitInfo* pour évaluer la qualité d'un attribut.

Pour créer un arbre (dit de *classification* = arbre de décision) binaire, CART utilise la fonction *Gini* définie par :

$$Gini(p) = 1 - \sum_{k=1}^c Pr(k|p)^2 = 2 \sum_{k < k'} Pr(k|p)Pr(k'|p) \quad (1)$$

- $p$  est une position dans l'arbre
- $c$  est le nombre de classes à caractériser.

☞ **simpleCart** ne fait pas d'arbre de régression. Elle produit un arbre binaire avec une classe binaire NON numérique ; elle ne prend pas une BD avec classe numérique.

- Sous Weka, la méthode CART se trouve dans l'onglet "*classify/trees/simpleCART*". Elle n'accepte pas de données numériques ; ce qui la différencie de la (vraie méthode CART). Pour un arbre de régression, voir *RepTree* dans les BEs suivants.

*SimpleCART* utilise l'indice d'impureté *Gini* et le critère de *split* (voir cours). De plus, l'implantation de cette méthode sous Weka utilise le principe de *minimal cost-complexity* pour l'élagage de l'arbre binaire produit.

### 5.1 Exercice

**Appliquez cette méthode** à la base de données "météo.arff" (non numérique : "weather-nominal.arff") en prenant la précaution de fixer le paramètre *usePrune* à False (sans quoi tout l'arbre sera élagué!). Pour changer un paramètre, on clique dans la zone où le nom de la méthode apparaît, en face de "choose" sous "classifier".

- Comparer le taux de succès (*accuracy*) des ces 3 méthodes (ID3, J48 et SimpleCart).
- Quelle est la meilleure méthode?
- Nous verrons plus loin comment mieux valider et comparer les modèles obtenus.

**Petit Bilan provisoire** : 1R insuffisante. On a obtenu des arbres de décision par les méthodes ID3, C4.5, CART (ces arbres peuvent aisément être transformés en règles de classification). Les taux d'erreurs et autres mesures d'évaluation permettent de quantifier / qualifier ces résultats.

## 6 Autres remarques sur Weka

- Dans la fenêtre principale, dans l'onglet "classify", en bas à gauche (sous "start"), un clique droit vous propose un menu de sauvegarde des résultats. On obtient le même effet avec "shift-Alt-clique gauche" dans la fenêtre des résultats.
- Un clique droit sur la méthode exécutée (zone **Result List**) permet de voir les options de visualisation.
- Sous Weka, dans la session *Preprocess*, il est possible d'éditer un fichier de données (bouton **edit**), le modifier et sauvegarder (bouton **save**). La sauvegarde (sous un nom différent) vous permet par exemple de conserver les données discrétisées (voir la section 14.5).
- Lorsque vous faites afficher la fenêtre des paramètres d'une commande sous Weka (en cliquant dans la ligne de la commande en question), une aide vous est proposée par le bouton *More*. Le bouton *Capabilities* vous en dit plus sur les capacités de la commande.

## 6.1 Autres formats de fichiers de données

Sous Weka, on peut charger une BD dans différents formats (p.ex, le format C4.5 constitué de *fichier.data* et *fichier.names*) et la sauvegarder dans une autre (par exemple, *fichier.arff*).

Outre le format arff, Weka (depuis les versions récentes) sait lire des données au format **csv** (données sorties des tableurs tels que Excel / OpenOffice / Numbers).

Pour le format csv (colonnes séparées par une virgule), veuillez à placer en première ligne les intitulés des colonnes. Ces deux formats (arff et csv) nous intéressent particulièrement. Pour les autres formats traités, consulter la documentation de Weka.

## 7 Exercices : la pratique de Weka sur l'exemple Banque

### 7.1 Banque : choix d'attribut

Sous Weka,

1. Charger la BD Banque *banque-app.arff* (*banque2-app.arff* éventuellement).
2. Choisir l'onglet "Select attributes" (outils de sélection d'attributs)
3. Dans la partie supérieure de la fenêtre, choisir "Attribute Evaluator"
4. Dans la liste de méthodes qui s'affiche, choisir "InfoGainAttributeEval" (*Entropie*); valider
  - Un message nous indique la méthode de recherche adéquate (*Ranker*).
  - On choisira dans un second temps la méthode *Ratio de Gain* ("GainRatioAttributeEval")
5. Dans la partie supérieure de la fenêtre, cliquer sur le bouton "Search Method"
  - On peut laisser le choix par défaut "Ranker ..." et de laisser les paramètres par défaut de cette méthode. Weka corrige les conflits entre le choix de "Attribute Evaluator" et de "Search Method".
6. Cliquer sur "Start" pour obtenir la liste des attributs ordonnée selon le critère (Entropie ou Ratio de Gain)
7. Comparer à vos résultats de la section 7.3
8. Refaire avec "GainRatioAttributeEval" (utilisé dans C4.5)
9. Tester d'autres méthodes de choix d'attributs.

### 7.2 Exercice : trouver le meilleur attribut à la main

L'application de l'entropie est rappelée en section 13.6 en page 25.

Sur la même base de données (*banque*), trouver manuellement le meilleur attribut et comparez avec les résultats ci-dessus. Un fichier Python de calcul vous est fourni (*Code-calcul-entropy.py*). Vous pouvez vous en servir pour ce calcul.

### 7.3 Exercice : arbre de décision de l'exemple Banque

Une banque dispose des informations suivantes sur un ensemble de clients. Une partie des données (*banque2-app.arff*) est présentée ci-dessous.

Client	Montant	Classe-Age	Résidence	Etudes	Internet
1	moyen	moyen	village	oui	oui
2	élevé	moyen	bourg	non	non
3	faible	âgé	banlieue	non	non
4	faible	moyen	bourg	oui	oui
5	moyen	jeune	ville	oui	oui
6	élevé	âgé	ville	oui	non
7	moyen	âgé	banlieue	oui	oui
...	...	...			

L'attribut ternaire *Montant* décrit la moyenne des montants sur le compte client. Le second attribut ternaire *Classe-Age* donne la tranche d'âge du client. Le troisième attribut quaternaire *Résidence* décrit la localité de résidence du client. Le dernier attribut binaire *Etudes* a la valeur oui si le client a un niveau d'études supérieures.

La **classe associée** à chacun de ces clients correspond au contenu de la colonne *Internet*. La classe oui correspond à un client qui effectue des opérations sur ses comptes bancaires en utilisant l'Internet.

**A faire à l'aide d'une calculatrice ou d'un tableur ou du simple code (fourni) :** trouver le meilleur attribut (à la racine) en utilisant la fonction gain basée sur l'entropie (cf. méthode ID3).

→ Avec l'enseignant, vérifier la justesse de vos calculs.

☞ Si vous voulez aller plus loin, vous pouvez construire l'arbre de décision puis calculer les performances sur l'ensemble test suivant.

Comme nous l'avons vu, vous aurez la possibilité de demander à Weka de vous construire cet arbre et tester ses performances (section 7.1, page 9).

Client	Montant	Classe-Age	Résidence	Etudes	Internet (bonne réponse)	Votre Réponse
21	moyen	âgé	village	oui	oui	?
22	élevé	jeune	ville	non	oui	?
23	faible	âgé	banlieue	non	non	?
24	moyen	moyen	bourg	oui	non	?

## 7.4 Exercice (optionnel) : règles sur exemple météo

Sur le jeu de données "météo" (*weather-nominal.arff*), lancer l'extraction des règles d'association (méthode *A Priori* qui se trouve dans l'onglet "Associate" de Weka).

Essayer diverses méthodes de test.

Modifier les paramètres de la méthode en choisissant "True" pour l'option "CAR" (CAR : *classification association rules*).

Comparez les règles obtenues avec les résultats de la méthode ID3/J48. Consignez les résultats.

Num	Temps(S)	Temperature(T)	Humidité(H)	Vent(V)	Classe
1	ensoleillé	Elevée	Elevée	non	N
2	ensoleillé	Elevée	Elevée	oui	N
3	nuageux	Elevée	Elevée	non	P
4	pluvieux	Moyenne	Elevée	non	P
5	pluvieux	Faible	Normale	non	P
6	pluvieux	Faible	Normale	oui	N
7	nuageux	Faible	Normale	oui	P
8	ensoleillé	Moyenne	Elevée	non	N
9	ensoleillé	Faible	Normale	non	P
10	pluvieux	Moyenne	Normale	non	P
11	ensoleillé	Moyenne	Normale	oui	P
12	nuageux	Moyenne	Elevée	oui	P
13	nuageux	Elevée	Normale	non	P
14	pluvieux	Moyenne	Elevée	oui	N

La même base d'instances avec deux attributs numériques est donné dans le fichier *weather-numeric.arff* :

## 7.5 Autres Exemples de manipulation

Si vous avez envie d'aller plus loin :

- Exercez-vous sur les autres exemples fournis pour ce BE (et dans le répertoire **data** de Weka). L'exemple des **champignons** est une bonne base de données (quelque peu complexe) pour la prise en main de WEKA. On l'utilisera dans le BE suivant (et dans le bonus ci-dessous).

- Aux BEs suivants, on utilisera l'apprentissage à base d'instances (IBL) :

**Classify/lazy/IB1 et IBK** et la création d'un arbre de régression avec une classe numérique. (IBL avec 1-NN .. K-NN).

## 8 Travail 1

Réalisez **au moins deux cas** parmi les propositions ci-dessous. Aux besoins de la méthode choisie, vous aurez recours à la discrétisation des attributs numériques.

☞ Tenez compte dans vos comptes rendus de l'indice Kapa (expliqué section 12.1, page 18) , "mean absolute error", "TP rate" et les autres mesures affichées par Weka à la fin d'un apprentissage (voir cours chap4-Part1-2).

**1-** Appliquer la méthode **ID3, J48** et **CART** à l'exemple **Titanic** afin d'obtenir les arbres de décision et les règles qui en découlent. Les données sont fournies sous format *.arff* et *.csv*.

**2-** Idem pour les BDs "Cars.arff" et "Nursery.arff".

**3-** Appliquer la méthode (Attention à Id3 : attribut branchant "animal" mais bug dans la matrice de confusion si 10-XV!) qui vous donnera les meilleurs résultats à l'exemple **Zoo** et présenter les résultats.

**4-** (En avance sur BE2) Appliquer une méthode d'obtention d'arbre de décision *RepTree* (une méthode d'obtention d'arbre – Tree –) à l'exemple **weather\_bigger.arff** (présent dans le répertoire du BE1) et présenter les résultats. On reprendra cet exemple en BE2.

☞ Sur cette BD, une discrétisation peut être nécessaire si on applique une méthode telle que *ID3*.

☞ Aux BEs suivants, on utilisera l'outil Weka de comparaison statistique des résultats de différentes méthodes sur une même BD.

## 9 Travail 2 : Méthode Bayésienne

On utilise ici la méthode Bayésienne sous WEKA.

**Paramètres de la méthode bayésienne Naïve :** une fois la méthode "classify/bayes/NaiveBayes" choisie, si on clique sur la ligne "NaiveBayes" à droite de *choose*, on peut fixer les valeurs :

- *Debug* : plus de détails.
- *"Display old format ..."* : affichage des détails pour chaque classe (moins évident pour les prédictions, laisser à "false").
- *"Use Kernel Estimator"* : utilisant de méthode à base de noyau pour l'estimation de la distribution des valeurs réels (KDE : *Kernel Density Estimator*)
- *"UseSupervisedDiscretization"* : discrétiser (automatiquement) les valeurs des attributs réels.

### 9.1 Exemple Haggis

Support du cours relatif à cette section : chapitre 4, partie 1.

(I) Lancer Weka, puis choisir onglet *Explorer*.

→ NB : si problèmes de mémoire, lancer Weka plutôt avec l'option

`java -Xmx2048m -jar weka.jar`

On demande 2048m (2048 mega octets=2GB). Fixer cette valeur par exemple à 256MB.

• On utilise la BD "Haggis" (Haggis sauvage : une espèce d'oiseau légendaire Écossais ressemblant à l'autruche)<sup>2</sup>.

• Charger le fichier *"haggis\_data.arff"*. Il contient 7 instances.

→ Les attributs explicatifs sont :

*skin, color, size, flesh, eats shortbread, length.*

L'attribut *classe* : *Haggis* et *No\_Haggis*.

• Dans l'onglet "preprocess", cliquer sur un des attributs (*skin, color ...*) pour voir quelques statistiques sur ces attributs (distribution des valeurs, ...).

(II) Application de la méthode Bayes :

Pour pouvoir prédire la classe (*Haggis/No\_Haggis*) pour une nouvelle instance, sélectionner "classifiers / Bayes / NaiveBayes".

→ Sous l'onglet "Classify" sous l'option "Test", choisir "Percentage split" puis fixer la valeur à 60%.

→ Cela veut dire que Weka divisera les données en 60% pour apprendre et 40% pour tester le modèle obtenu. On utilisera le modèle obtenu pour faire de la prédiction sur de nouvelles instances.

• Appuyer sur "Start" pour construire le modèle.

• Les résultats de la classification sont affichés dans la fenêtre à droite.

→ Dans cette fenêtre, on obtient quelques statistiques (fréquences).

(III) Question 1 :

Que voyez-vous en face de l'attribut *skin*? Interpréter les valeurs des ses 3 modalités.

→ **Indice** : penser à *Laplace* (Cours Chap 4-I)!

Par exemple, on a :

Attribute	Class	
	0	1
	(0.44)	(0.56)
=====		
skin		
hairy	1.0	5.0
smooth	3.0	1.0
[total]	4.0	6.0

Or, dans la BD, on a :

skin		
hairy	0	4
smooth	2	0
[total]	2	4

2. En fait, "Haggis" n'a jamais existé et son origine viendrait de ceci : "Haggis" est un plat (très apprécié des) Écossais, composé d'une farce à base d'abats hachée et entourée (comme une saucisse) d'une panse d'agneau. Pour ne pas répondre à ceux qui leur demandent la recette, les Écossais répondent que c'est un oiseau (à 4 pattes mais délicieux)!

(IV) En bas de cette fenêtre, on a une matrice de confusion.

La matrice de confusion est une table de contingences. Les valeurs actuelles (BD) sont celles qui sont dans la BD (qu'on aimerait voir prédites) et les valeurs "modèle" sont les conclusions (prédictions) du modèle appris.

Valeurs Prédites		Valeurs actuelles (BD)		Total
		Positifs	Négatifs	
	Positifs →	Vrais positifs (TP)	Faux Positifs (FP)	les positifs (modèle)
	Négatifs →	Faux Négatifs (FN)	Vrais Négatifs (TN)	les négatifs (modèle)
	Total	les positifs (BD)	les négatifs (BD)	

**Question 2 :** Commenter la matrice de confusion de notre exemple.

☞ Remarquez bien que l'apprentissage est fait sur l'ensemble de la BD (malgré le 60-40%). La raison est simple : les fréquence et les paramètre d'une loi Normale se calculent avec tout. Ensuite, le verdict dépendra des 40% de test.

→ Remarquez également le traitement des données manquantes (P.ex : seul 6 valeurs pour *skin*).

(V) **Questions 3 :** dans l'onglet "Preprocess", éditer au besoin la BD.

1. Dans la BD chargée (sans passer par Weka), quel type de données manquante est désigné par '?' ?
2. Comment Weka traite ces valeurs (revoir les statistiques, cf. Question 1) ?
3. L'attribut "length" est continue (réel). Comment Weka traite les attributs numériques par opp. aux données catégorielles (de valeurs énumérées). Quelle hypothèse de distribution pour les numériques ?
4. On veut classer une nouvelle instance (dont la classe est inconnue) :

$x = (skin=smooth, color=red, size=large, flesh=hard, eats\_shortbread=yes, length=3.25)$

(VI) **Question 4 :** classer  $x$  à la main (sans passer par Weka). Quelle serait sa classe ?

**Indices :** Oui empiriquement !.

→ Visualisation des données et les données absentes (onglet "Edit" de "Preprocess").

→ Observer également les statistiques sur ces mêmes données (disponibles dès le chargement de la BD).

No.	1: skin Nominal	2: colour Nominal	3: size Nominal	4: flesh Nominal	5: eats_shortbread Nominal	6: length Numeric	7: is_haggis Nominal
1	hairy	brown	large	hard	1	3.25	1
2	hairy	green	large	hard	1	4.22	1
3	red	small	soft	0	1.27	0	0
4	hairy	green	large	hard	1	3.55	1
5	smooth	red	small	soft	0	2.13	0
6	smooth	green	large	soft	1	2.67	0
7	hairy	large	soft	0	3.77	1	1

(VII) **Voyons comment** Weka classe cette instance avec le modèle obtenu ci-dessus ?

**Indices :**  $x = (skin=smooth, color=red, size=large, flesh=hard, eats\_shortbread=yes, length=3.25)$

☞ **Pour les données de test, il n'y a pas de Lissage.**

**Rappel** de la formule de Bayes appliquée à notre cas :

$$P(Haggis|x) = \frac{P(smooth|Haggis) * P(red|Haggis) * P(large|Haggis) * P(hard|Haggis) * P(eats\_shortbread|Haggis) * P(3.25|Haggis) * P(Haggis)}{P(x)} \leftarrow \text{la somme des 2 numérateurs}$$

☞ N.B. : la notation  $P(Haggis|x)$  veut dire  $P(Classe = Haggis|X = x)$ .

## Questions 5

- Calculer l'attribut *length* dans l'hypothèse "Haggis" puis "No\_Haggis" via la fonction de densité de probabilité (PDF pour les variables continues) et obtenez :

$$P(\text{length} = 3.25 | \text{Haggis})$$

et  $P(\text{length} = 3.25 | \text{No\_Haggis})$

- Puis calculer la probabilité d'appartenance de  $x$  à chacune de ces 2 classes.

→ Rappel : la marginal  $P(x)$  au dénominateur est :

$$P(x | \text{Haggis}) \cdot P(\text{Haggis}) + P(x | \overline{\text{Haggis}}) \cdot p(\overline{\text{Haggis}})$$

→ C-à-d : la somme des numérateurs dans les calculs de  $= P(\text{Haggis} | x)$  et  $P(\overline{\text{Haggis}} | x)$

Pourquoi : ?

On sait que  $\sum_y P(x, y) = p(x)$  si "les"  $y$  sont disjoints, et on a :  $p(x, y) = p(x | y) \cdot p(y)$ .

- Les valeurs dont vous aurez besoin pour ces calculs sont ci-contre :

☞ Vous devriez trouver, par vos calculs, que  $x$  appartient nettement à la classe **Haggis**. Notez les probabilités obtenues.

☞ Pour un cas bi-classes comme ici, le calcul de  $P(x)$  n'est pas nécessaire : il suffit de comparer les deux *PDF* (*Fonction de densité de probabilité*, avant la normalisation par  $P(x)$ ).

### Naive Bayes Classifier

Attribute	Class	
	0 (0.44)	1 (0.56)
=====		
skin		
hairy	1.0	5.0
smooth	3.0	1.0
[total]	4.0	6.0
colour		
brown	1.0	2.0
green	2.0	3.0
red	3.0	1.0
[total]	6.0	6.0
size		
small	3.0	1.0
large	2.0	4.0
[total]	5.0	5.0
flesh		
soft	4.0	2.0
hard	1.0	4.0
[total]	5.0	6.0
eats_shortbread		
0	3.0	2.0
1	2.0	4.0
[total]	5.0	6.0
length		
mean	1.9667	3.8104
std. dev.	0.4014	0.4077
weight sum	3	4
precision	0.4917	0.4917

(VIII) On enregistre l'instance  $x$  ci-dessus ainsi que 3 autres (voir ci-bas) dans le fichier "*haggis\_test.arff*" (fichier fourni) :

@RELATION Haggis

```
@ATTRIBUTE skin          hairy,smooth
@ATTRIBUTE colour        brown,green,red
@ATTRIBUTE size           small,large
@ATTRIBUTE flesh          soft,hard
@ATTRIBUTE eats_shortbread 0,1
@ATTRIBUTE length         NUMERIC
@ATTRIBUTE is_haggis      0,1
```

@DATA

```
smooth,red,large,hard,1,3.25,1
hairy,brown,small,hard,1,2.56,0
smooth,green,small,hard,1,3.05,1
hairy,red,large,soft,0,2.05,1
```

☞ Les classes sont précisées mais, **pendant le test du modèle, Weka ignore cette information** et tentera de la prédire : on peut obtenir ou pas ces mêmes classes ; ce qui permet d'évaluer le taux d'erreur du modèle.

- Choisir l'onglet *Classify* dans Weka puis dans la zone *Test Options*, choisir *Supplied test set*, ouvrir le fichier "*haggis\_test.arff*", puis cliquer sur "Close". Appuyer maintenant sur Start pour lancer la méthode Bayes Naïve où l'apprentissage aura lieu sur "*haggis\_data.arff*" et le test sur "*haggis\_test.arff*" (sur  $x$ ).

→ Dans la fenêtre qui s'affiche à droite, vous trouverez les mêmes statistiques que dans le cas précédent mais la matrice de confusion est différente.

## Question 6 :

- a) A l'aide du clique droit et "visualize classifier errors" dans "Result List", **trouver** si la classe de l'instance  $x$  par vos calculs ci-dessus correspond à celle de Weka.
- b) Pour nos 4 exemples de test, quel est le taux du succès ? Commenter la matrice de confusion obtenue.

## 10 Travail 3 : La méthode Bayésienne sous WEKA

Nous avons utilisé la méthode Bayésienne sur la BD *Haggis* en BE1. Étant donné l'importance de cette méthode, nous la réutilisons ici sur une BD de Spam.

### 10.1 Travail à rendre (3) : Bayésienne sur la BD Spam

On s'intéresse à la BD *Spam*. Il est nécessaire de faire quelques préparations de données.

**N.B.** : les manipulations suivantes vous permettent de mieux maîtriser Weka. Cependant, votre travail est facilité et les fichiers résultants de cette préparation vous sont fournis (les fichiers *spambase\_app\_bool\_sans\_3\_att.arff* et *spambase\_test\_bool\_sans\_3\_att.arff*).  
Si vous ne faites pas ces préparations de données, vous pouvez donc aller directement à la première question ci-dessous.

Dans l'onglet "Preprocess", ouvrir la BD "spambase.arff".

Supprimer les attributs "capital\_run\_length\_average", "capital\_run\_length\_longest" et "capital\_run\_length\_total" en cliquant sur la box à leur gauche, puis "remove".

Les attributs restants représentent les fréquences relatives des mots / caractères significatifs dans un mail.

#### Binéarisation :

On souhaite convertir ces valeurs en booléens : 1 si le mot / caractère est présent, 0 sinon.

Pour réaliser cela, appuyer sur "Choose" dans la zone "Filter" et sélectionner "filters / unsupervised / attribute / NumericToBinary".

→ Cliquer sur "Apply" tout à droite de la même zone.

- Les attributs sont maintenant en booléens.

→ Chaque mail est représenté maintenant par un vecteur de 55 booléens permettant de savoir si tel mot ou caractère existe dans un mail.

→ On appelle cela une représentation par *sac de mots* (*bag of words*).

☞ Remarquons qu'ici, on ne tient pas compte de l'ordre des mots dans un mail, ce qui peut être une information importante ("*Data Mining Blablabla..*" et "*Data Blablabla Mining..*" n'ont pas le même sens ; "*Jean Pierre mange*" vs "*Jean mange Pierre*" non plus ! ).

Enregistrez ces résultats en cliquant sur "Save" dans le fichier *spambase\_app\_bool\_sans\_3\_att.arff*.

- On souhaite appliquer la méthode Bayésienne Naïve à la BD *spambase\_app\_bool\_sans\_3\_att.arff* et distinguer un pourriel (*spam*) d'un "vrai" mail (*ham*) en calculant une distribution du nombre d'occurrences de chaque mot dans un *spam* ou *ham* (non-spam).

Choisir "Classify" puis Bayes/ NaiveBayes. Laisser les options par défaut et construire le modèle.

**Question 1** : Etudier les résultats en particulier le pourcentage des instances correctement classées. On constate que le modèle se comporte assez bien, malgré les hypothèses faites (que ces données représentent bien les mails, l'indépendance des attributs, sac de mots, etc).

**Question 2** : Quelle serait la raison de ce bon comportement ?

**Question 3** : Quelle seraient les problèmes principaux si on ne faisait pas ces hypothèses sur cette BD ?

**Question 4** : Examiner le modèle obtenu par Weka et trouver les probabilités *a priori* de chaque classe. Comment Weka les calcule ?

→ Vous l'avez sous vos yeux ! Vous pouvez également les observer dans l'onglet "Preprocess".

**Question 5** : Comme dans cette méthode dans le BE1, calculer la probabilité conditionnelle d'observer le mot "3d" sachant que le mail est un spam ( $P(3d|spam)$ ), de même pour non-spam ( $P(3d|non - spam)$ ). Pour ce faire, on utilise les nombres (les comptes) affichés par Weka dans la fenêtre des résultats. Le format est (pour le mot "3d") :

		Class	
Attribute		1	0
		(0.39)	(0.61)
=====			
word_freq_make_binarized			
...			
word_freq_3d_binarized			
0		1775.0	2781.0
1		40.0	9.0
[total]		1815.0	2790.0

→ NB : il y a lissage de Laplace :  $40+9=49$  dans la table ci-dessus. Or, si on affiche ce même attribut dans "Preprocess", on a 47.

- On veut tester notre modèle pour savoir si un mail est *spam* ou *ham*. L'apprentissage a eu lieu avec la BD *spambase\_app\_bool\_sans\_3\_att.arff* et pour les tests, on utilise *spambase\_test\_bool\_sans\_3\_att.arff* qui est au même format (binaire) que la BD d'apprentissage.

N.B. : Pour ce découpage, un exemple est donnée en section ?? page ?? . Également, un document sur la prise en main de Weka (fichier "Data-Preparation-INTRO-WEKA.pdf", voir BE1) aborde ce point en détails.

Si le fichier *spambase\_test\_bool\_sans\_3\_att.arff* contient plus de 4 instances, tronquez-le et ne conservez qu'au maximum 4 instances pour les tests.  
 Pour ce faire, servez-vous de Weka et de son filtre "Resample" appliqué à *spambase\_test\_bool\_sans\_3\_att.arff* et demandez à conserver un pourcentage qui correspond à 3 ou 4 instances.  
 Une autre façon de faire est de vous servir d'un éditeur texte, éditer *spambase\_test\_bool\_sans\_3\_att.arff* pour ne conserver qu'au maximum 4 instances et supprimez les autres.

- Dans l'onglet "Classify", choisir "Supplied test set" et charger le fichier *spambase\_test\_bool\_sans\_3\_att.arff* qui contient un *spam*. Choisir "close" pour revenir à la classification et cliquer sur "Start" pour lancer la méthode Bayes.

#### Questions 6 :

- 1- Est-ce que le modèle classe bien cette nouvelle instance.
  - 2- Éditer ce fichier test, identifier des mots non spam et ajouter les à ce test. Laisser la classe finale.
- ☞ Pendant le test du modèle, Weka ignore cette classe pour le test et utilisera le modèle pour la prédire.
- 3- Relancer le classifieur sur cet exemple modifié. La prédiction a-t-elle changé ?

**Remarques :** en insérant des mots "non spam" à l'exemple, on a ajouté des "évidences" supplémentaires (dites "votes") pour rendre l'exemple un *ham* et s'appuyer sur l'indépendance des attributs supposée par Bayes naïve. Chaque mot contribue de manière indépendante des autres à la classe finale.

☞ C'est la raison pour laquelle beaucoup de spams contiennent des mots ou des passages de livres pour ajouter des mots "bons" et espérer échapper aux filtres que l'on met en place sur nos serveurs. Pour palier ces difficultés dans la pratique, beaucoup de logiciels commerciaux anti-spam (e.g. *gmail*, *Spam-Assassin*, etc) utilisent des méthodes de détection de spam plus sophistiquées.

## 10.2 Rappel : paramètres de Bayésienne Naïve

Une fois la méthode "classify/bayes/NaiveBayes" choisie, si on clique sur la ligne "NaiveBayes" à droite de *choose*, on peut fixer les valeurs :

- Debug : plus de détails.
- "Display old format ..." : affichage des détails pour chaque classe (moins évident pour les prédictions, laisser à "false").
- "Use Kernel Estimator" : utilisant de méthode à base de noyau pour l'estimation de la distribution des valeurs réels (KDE : *Kernel Density Estimator*)
- "UseSupervisedDiscretization" : discrétiser (automatiquement) les valeurs des attributs réels.

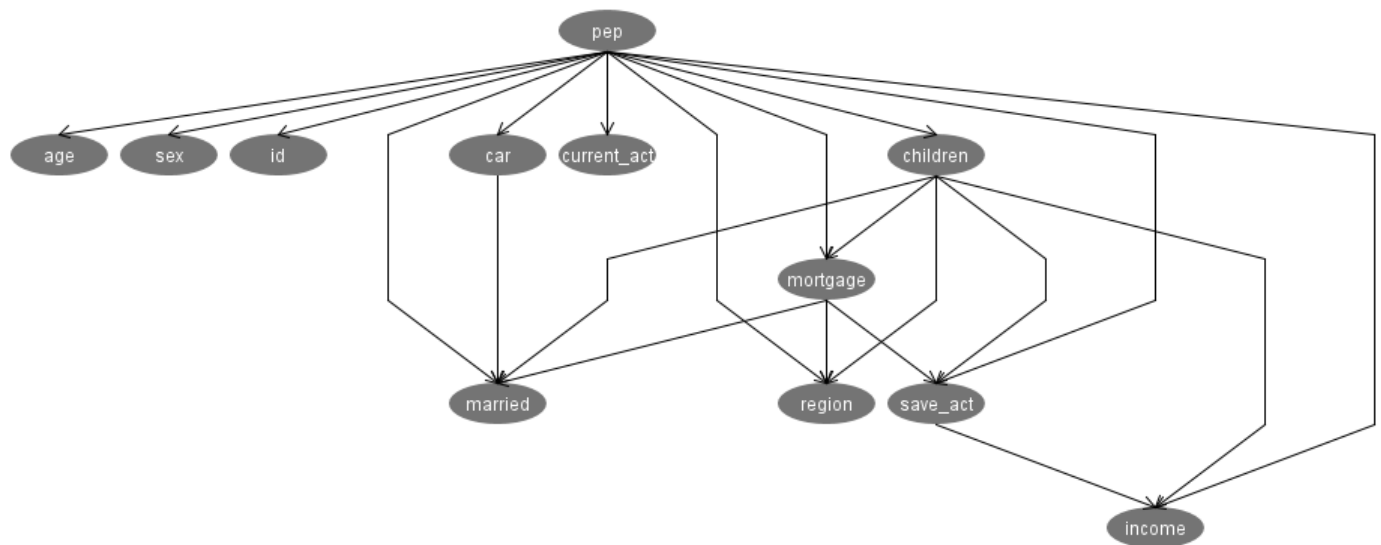
## 11 Remarque sur les réseaux de Bayes

Weka peut produire un réseau Bayésien (onglet *Classify/classifiers/Bayes/Bayse Net*). Par contre, il ne permet pas de prédiction sur le réseau obtenu, à moins de procéder aux calculs manuels.  
 Pour les prédictions, il faudra alors sauvegarder le réseau obtenu, puis utiliser un logiciel (libre) tel que *JavaBayes*.

A titre indicatif, le réseaux obtenu pour la BD "Bank-data-all.arff" sera :

Scheme: weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.global.TabuSearch -- -L 5 -U 10 -P 4 -S  
 L00-CV -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5  
 Correctly Classified Instances 487 81.1667 % Incorrectly Classified Instances 113 18.8333 %  
 a b <-- classified as  
 211 63 | a = YES  
 50 276 | b = NO

On clique sous Weka sur un noeud pour obtenir sa table de proba.  
 Avec un Kappa de 0,62 et un ROC area de ~84% avec la validation 10-XV



Où on peut constater les effets des attributs explicatifs sur la classe PeP (parcourir les flèches dans le sens inverse). Pour en connaître les détails, sous Weka, on peut cliquer sur chaque noeud pour disposer de la table de probabilités sur ce noeud.

## 12 Évaluation des modèles : quelques éléments

☞ Voir ci-dessous comment (empiriquement) comparer plusieurs modèles obtenus pour une BD. Nous verrons une méthode de comparaison statistique au BE3.

La comparaison passe toujours par celle de différents indicateurs. Nous ferons des comparaisons simples ci-après. Les méthodes plus élaborées (voir BE3) utiliseront des tests statistiques.

La mesure kappa ci-dessous est l'un de ces indicateurs.

### 12.1 A propos des mesures Kappa

**Kappa (Chance corrected agreement)** : voir également cours chapitre 4 part 2.

- Est comme un coefficient de corrélation ; on l'utilise pour la similarité et la fiabilité des résultats.
- Plusieurs références (où hypothèses) sont possibles dans ces calculs dont les résultats dépendent de la prévalence et du biais (v. + loin)

☞ **Un  $kappa > 0$  veut dire** : le modèle appris fait mieux que la chance (pile ou face),  $kappa > 0.6$  est préférable

☞  $Kappa \leq 0$  : désaccord (ou accord seulement au hasard)

☞  $Kappa = 1$  : max d'accord

☞ Observez toujours kappa avec e.g. l'aire sous ROC. Préférer les mesures pondérées

**Kappa** pour la table de contingence suivante :

$$K = \frac{(O_{ag} - E_{ag})}{(1 - E_{ag})}$$

$O_{ag}$  : observed agreement (diag / total) = **Accord**

$E_{ag}$  : expected agreement (expected in diag / total) = **Hasard**  
= la probabilité d'un accord aléatoire

$$O_{ag} = 57/91 = .63$$

$$E_{ag} = 31/91 = .34$$

$$K = (.63 - .34)/(1 - .34) = 0.43$$

	ref std A	ref std B	ref std C	total
system A	13(6.6)	4	6	23
system B	8	23 (13.1)	2	33
system C	5	9	21 (11.3)	35
Total	26	36	29	91

13 + 23 + 21 = 57 les valeurs sur la diagonale

6.6 + 13.1 + 11.3 = 31 les valeurs entre parenthèse (attendus = expected) sur la diagonale

☞ Les détails de calculs de cet exemple sont donnés dans l'exemple suivant.

**Un autre exemple** : ici, les valeurs attendues ne sont pas données → on se réfère au calcul du *Hasard*.

	Ref. Std +	Ref. Std -
Système +	25	0
Système -	50	25

Pour trouver Kappa, on calcule :

1- **Accord** : proportion des données sur laquelle les deux sont d'accord  
= (TP+TN)/total BD      c-à-d.,  $\frac{\text{la diagonale}}{\text{taille BD}}$

2- **Hasard** :  $(\sum_i (\text{total ligne } i * \text{total col } i)) / \text{total BD au carre}$

Ici, on a :

$$\text{pour 1 : } \frac{25 + 25}{100} = 0.5$$

$$\text{pour 2 : } \frac{(25 * 75(\text{lig/col 1})) + (75 * 25(\text{lig/col 2}))}{100 * 100} = 0.375$$

$$\rightarrow \text{Ce qui donne : } kappa = \frac{0.5 - 0.375}{1 - 0.375} = 0.2$$

**Pour la première table précédente :**

Pour calculer  $E_{ag}$  (le Hasard), les valeurs entre parenthèses étaient données pour chaque  $i$  mais on peut les retrouver :

$$\text{Hasard : } (\sum_i (\text{total ligne } i * \text{total col } i)) / \text{total BD au carre}$$

$$\text{Ici : } E_{ag} = \frac{(23 * 26) + (33 * 36) + (35 * 29)}{(91 * 91)} = 0.338 \sim 0.34$$

## Notes sur la mesure Kappa pour 2 avis (Kappa de Cohen)

$$\frac{P(\text{accord relatif des deux}) - P(\text{accord par hasard})}{1 - P(\text{accord par hasard})}$$

→  $1 - P(\text{accord par hasard})$  représente le maximum d'accord possible.

Kappa mesure l'accord entre deux modèles (2 codeurs)

☞ Dans le cas des méthodes d'apprentissage, les 2 avis sont ceux exprimés dans la matrice de confusion où l'accord relatif des 2 modèles =  $TP + TN$

- Kappa = (la différence entre l'avis d'un modèle et le hasard) divisée par (1 - hasard).
- $Kappa = 1$  : max d'accord
- $Kappa \leq 0$  : désaccord (ou accord seulement au hasard)
- $Kappa \geq 0$  : le modèle fait mieux que le hasard (pile ou face).
- Moins il y a des classes, plus Kappa sera grand
- Si plus de 2 avis : prendre Kappa de Fleiss (cf. cas général ci-dessus)

## Kappa et les règles d'association (nous sera utile au BE2) :

Pour une règle (ou l'itemset = {A,B})  $A \Rightarrow B$ , on a :  $Kappa = \frac{P(AB) + P(\overline{A}\overline{B}) - P(A)P(B) - P(\overline{A})P(\overline{B})}{P(A) + P(B) - P(A)P(B) - P(\overline{A})P(\overline{B})}$

Dans une autre formulation plus simple (sans tenir compte de  $\overline{A}$  et  $\overline{B}$ ), on a :

$$Kappa = \frac{P(AB) - P(A)P(B)}{P(A) + P(B) - 2P(A)P(B)}$$

Où  $P(A)P(B)$  est la probabilité théorique de  $A$  et de  $B$  en l'absence de toute hypothèse (de dépendance ou d'indépendance) inspirée d'un processus général de Bernoulli, et qui représente donc  $E_{ag}$  dans  $K = \frac{(O_{ag} - E_{ag})}{(1 - E_{ag})}$

☞ Un cas particulier (où l'absence de  $\overline{A}$  et  $\overline{B}$  est justifié) est une BD. où on a forcément  $A$  ou  $B$  dans chaque instance auquel cas  $P(A) + P(B) - P(A)P(B) = 1$  et nous pouvons retrouver la forme originelle de Kappa.

## Kappa et la matrice de confusion : soient les données d'un modèle :

Kappa statistic      0.3108

ROC :      0.709

### • Et la matrice de Confusion :

a	b	< -- classifié comme
59	2	— a = 0
27	12	— b = 1

• Ici, pour 100 instances, on a  $TP + TN = 59 + 12 = 71$ ,  $FP + FN = 27 + 2 = 29$ .

• Le pourcentage de correctement classés = justesse simple (accuracy).

→ Son inconvénient est que la valeur n'est pas pondérée par le hasard (*chance corrected*) et n'est pas **sensible à la distribution** des classes.

→ Dans ce cas, l'aire ROC est un bon complément (ici, 0.709 pour les 2 classes).

• Kappa est pondérée et mesure l'accord entre le modèle et la BD (d'apprentissage).

☞  $Kappa \geq 0$  : le modèle fait mieux que le hasard (pile ou face). En statistiques, certains considèrent que Kappa est exploitable seulement à partir de 0.6 (ou 0.7) ; en deçà, Kappa dit peu de chose.

## Autres remarques :

Les taux d'erreurs numériques (données par Weka) comme *Mean absolute error*, *Root mean squared error*, *Relative absolute error*, *Root relative squared error* sont plus utiles à la prédiction numérique qu'à la classification.

→ Les prédictions numériques ne sont pas justes ou erronées mais leur erreur a une certaine magnitude reflétée par ces valeurs d'erreur.

## 12.2 Matrice de Confusion de classification non binaire

Nous avons utilisé ci-dessus la matrice de confusion d'une classification binaire. Voyons comment faire pour les autres cas.

Suivant la méthode appliquée, on doit adapter certaines mesures au modèle obtenu. Par exemple, dans le cas de 3 classes, la valeur **TN** (*True Negative*) ne s'obtient pas directement.

Pour un cas bi-classes, les valeurs TP, TN, FP et FN sont directement accessibles. Par contre, pour une classification avec plus de 2 classes, comment procéder ?

- Exemple des sorties d'un modèle avec les codes couleurs :
  - noire : Bien classés, **orange** : Chats pris pour autre chose, **magenta** : Chiens pris pour autre chose, **bleu** : Loups pris pour autre chose.

		Actuels (B.D.)		
		Chat	Chien	Loup
Prédictions	Chat	5	2	0
	Chien	3	3	2
	Loup	0	1	11

- Constat : sur 8 Chats, 5 ont été bien classés (TP) et 3 ont été pris pour des Chiens.
- Parmi les 6 Chiens, 3 ont été bien classés (TP) et 2+1=3 autres non.
- Parmi les 13 loups, 11 ont été bien classés (TP) et 2 autres non.

Calcul de TP, TN, FP et FN pour le **Chat** (suivre les couleurs) :

- Rappel de la table de confusion des 3 classes :

		Actuels (B.D.)		
		Chat	Chien	Loup
Prédictions	Chat	5	2	0
	Chien	3	3	2
	Loup	0	1	11

- Considérez  $Chat \times Chien$  mais on a besoin des non-Chats (= Chien + Loup)


	Chat	Chien	
Chat	TP=5 Chiens	FP= 2 + 0	← Positifs (Modèle)
Chien	FN= 3 + 0	TN= 3+11 + 1+2 =17	← Négatifs (Modèle)

- Il y a dans cette table :
  - 5 Chats bien classés (TP) et 3 des (vrais) Chats sont pris pour des Chiens (FN)
  - 2 Chiens Pris pour Chats (FP)
  - Les TN pour notre matrice (on prend Chats comme repère) :
    - 11 Loups et 3 Chiens sont bien classés (donc TN pour Chats),
    - 2 Loups sont pris pour des Chiens (TN pour Chats) car ils ne sont pas pris pour des Chats ; 1 Chien est pris Loup (TN pour la même raison.)
- De même pour le *Chien* et le *Loup* : on crée les matrices de confusion 2 à 2.

## 12.3 Comparaison des méthodes

Pour comparer des modèles obtenus via différentes méthodes, les choix sont :

- Weka propose un outil de comparaisons (dans "experimentation", cf. test d'*étudiant*) avec quelques limitations. Voir BE3 (ou demander à l'enseignant si vous êtes impatient!).
- On peut procéder soit-même à ces comparaisons. C'est ce que nous ferons pour ce BE où nous comparons simplement (linéairement) les indicateurs produits par Weka.

 Si vous ne voyez pas ces mesures s'afficher dans les sorties, cliquez sur "**More Options**" (au dessus du bouton "**Start**") puis dans la fenêtre qui s'affiche, cliquez sur "**Evaluation metrics...**" et cochez les mesures que vous voulez voir afficher.

Pour ce faire, supposons avoir appliqué 4 méthodes à une BD. et avons obtenu 4 modèles. Établir un tableau comme ci-dessous (voir après ce tableau pour quelques explications) :

Mesures	Modèles →	$M_1$	$M_2$	$M_3$	$M_4$	Remarques
TP % (True Positive)		$TP_1$	$TP_2$	$TP_3$	$TP_4$	simple % de TP sur $ BD $
FP % (False Positive)		$FP_1$	$FP_2$	$FP_3$	$FP_4$	simple % de FP sur $ BD $
Precision = $\frac{TP}{TP+FP}$		$Pr_1$	$Pr_2$	$Pr_3$	$Pr_4$	le nombre de TP, pas son %
Recall = $\frac{TP}{TP+FN}$		$Rc_1$	$Rc_2$	$Rc_3$	$Rc_4$	équivalent à "TP rate" = "sensitivity"
Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$		$Acc_1$	$Acc_2$	$Acc_3$	$Acc_4$	TN=True Negative, FN=False Negative
Specificity = $\frac{TN+FP}{TN+FP+TP+FN}$		$SP_1$	$SP_2$	$SP_3$	$SP_4$	
F-Measure = $\frac{2*Recall*Precision}{Recall+Precision}$		$F_1$	$F_2$	$F_3$	$F_4$	
FP rate = $\frac{FP}{TN+FP}$		$Fr_1$	$Fr_2$	$Fr_3$	$Fr_4$	Cas idéal : FP rate = 0, ici 2 ex aequo
Roc			best			
Kappa				best		
Coverage of cases (0.95 level)			best			
Mean rel. region size (0.95 level)				best		
Mean absolute error				best		à partir d'ici, le best est le minimum
Relative absolute error					best	le best est le minimum
Root relative squared error			best			le best est le minimum
Root mean squared error		best		best		le best est le minimum
Comptes		4	5	6	3	← $M_3$ semble la meilleure.

La quasi toutes ces mesures sont affichées par Weka, dans la fenêtre des résultats à droite.

- **TP %** (ou *TP rate*) affiché par Weka est une moyenne qui correspond aux *instances correctement classées* (ou *l'accuracy = TP + TN*) dans différentes classes. Remarquez que *TN* pour une classe est *TP* pour les autres. Par contre, pour *FP%*, observez les indications de Weka.

→ Si vous voulez une comparaison plus complète, appliquez la méthode de construction (donnée ci-dessus) de la matrice de confusion pour un cas non binaire et reportez FN et FP dans le tableau de comparaison.

- **PRC area** : l'aire sous Precision-Recall-Curve (voir ci-dessous).

- La *surface Roc* (voir cours) est une valeur dans l'intervalle [0,1]. Plus elle est proche de 1, mieux c'est. Le hasard (la droite  $f(x)=x$ ) représente une surface de 50% pour un cas bi-classe.

- Signaler ensuite la meilleure (**best**, en blue ci-dessus) de chaque mesure puis choisir le modèle qui maximise le nombre de "best".

- Nous étudierons toutes ces mesures dans les cours suivants.

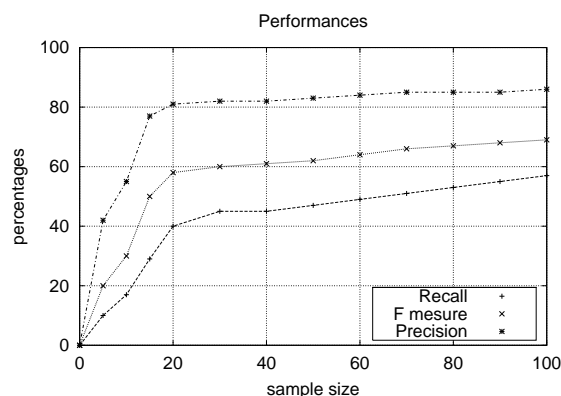
**Rappel (voir cours) de quelques unes de ces mesures et un ex. d'aire sous les courbes Precision, Recall et F-mesure :**

$$\text{Recall} = \frac{\# \text{ relevants extraits (TP)}}{\text{total relevants (TP + FN)}}$$

$$\text{Precision} = \frac{\# \text{ relevants extraits (TP)}}{\text{total extraits (TP + FP)}}$$

$$\text{F-mesure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

→ L'aire sous le courbe Precision ou celle de Recall est à droite. Pour la PRC, la courbe est produite dans un système de coordonnées avec en Axe des Xs la *Precision* et en Ys le *Recall*. Le repère inverse est également possible.



**Sous Weka**, cliquer droit sur le modèle appliqué (cadre gauche) pour visualiser différentes courbes (ROC, Precision, Recall, Lift, etc...). Pour la courbe ROC, cliquer droit et choisissez *threshold curve*. Dans la fenêtre proposée, dans la zone "Couleur", choisissez différentes courbes (ROC, Precision, recall, Lift, ...).

Vous pouvez également visualiser le *cost/benefit curve* de la même manière.

**A propos des mesures "Coverage" et "Mean rel. region size"** (du tableau ci-dessus) :

- Pour estimer un intervalle de confiance, Weka produit également les mesures **coverage** et **Mean rel. region size** (la moyenne relative de la largeur de l'intervalle de confiance à un niveau de 95%).

Pour obtenir ces mesures, la méthode appliquée doit disposer (sous Weka) de l'estimation de l'intervalle (cf. tout schéma général de *régression*).

- **Mean rel. region size** (ou, le nom sous Weka "sizeOfPredictedRegions") ou la largeur relative est une normalisation des largeurs d'intervalles. Cette normalisation est faite par l'intervalle des valeurs de la classe (attribut cible, voir ci-dessus) dans l'ensemble d'apprentissage (toute valeur relative  $\geq 100\%$  est écartée).

- La **couverture** (*coverage*) correspond à  $\frac{N_{out}}{N_{tot}}$  où  $N_{out}$  est le nombre d'instances bien classées (faire aussi attention aux instances non classées, s'il y en a) et  $N_{tot}$  est le total des instances. Dans une méthode de clustering,  $N_{out}$  est le nombre d'instances dans un cluster (voir cours).

Le (*coverage*) empirique d'un estimateur d'intervalle doit être d'un niveau de confiance  $\geq 95\%$ . Un estimateur raisonnable est celui qui affiche une couverture  $\geq 95\%$  tout en produisant un intervalle "étroit".

- Ces deux valeurs sont calculées également pour une classe nominale si la méthode utilise une prédiction probabiliste. Dans ce cas, un "intervalle" est le plus petit ensemble de valeurs de classe (attribut cible) tel que la probabilité cumulative pour ces valeurs dépasse 95%. Dans ce même cas, la largeur relative d'intervalle est le nombre d'occurrences de la valeur cible (la classe) divisé par toutes les différentes valeurs cibles.

- Pour les *règles d'association* (voir BE2), on pourra ajouter à ce tableau d'autres mesures (Lift, Conviction, etc. voir cours Chapitre 4, Part 1.2)

- Pour les particularités de ces mesures adaptées en "extraction d'informations" (IR, cf. Google), voir le complément au chapitre 4, Part1.2.
- 

☞ La partie suivante est un supplément qui vous servira dans tous les BEs.

## 13 Annexes-1 : Rappels du cours sur les Arbres de décision

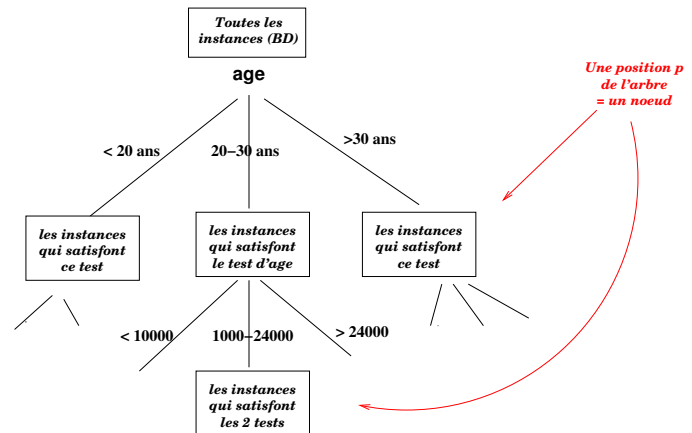
### Rappels sur l'apprentissage d'arbres de décision

La réalisation de ce BE nécessite la connaissance des points du cours rappelés brièvement dans cette section.

#### 13.1 Définition

Un arbre de décision associé à une BD d'individus (instances) est un arbre dont :

- Les **noeuds** internes sont étiquetés par un **attribut** de la description d'un individu.
- Pour un noeud donné, les **labels des arcs** issus de ce noeud correspondent aux réponses à un **test** sur cet attribut.
- Les feuilles (noeuds sans descendant) sont étiquetées par des **classes**.
- A chaque position (noeud)  $p$  d'un arbre de décision  $T$  correspond un sous-ensemble de l'échantillon d'entrée  $S$ , noté  $S_p$  qui est le sous-ensemble des éléments de  $S$  qui satisfont les tests depuis la racine jusqu'à cette position  $p$ .
- Rappelons que les éléments de  $S$  sont les descriptions des individus de la population.  $S_p$  peut contenir par exemple les individus d'âge 20-30 avec un revenu annuel de 10000-24000. (section 13.4).



Un exemple d'arbre de décision est donné plus loin

#### 13.2 Induction d'arbres de décision et Entropie

Un arbre de décision étant construit sur un jeu de données, on parle d'induction d'arbres de décision. L'algorithme général d'induction **DTBuild** est :

```
Données :  $D$  l'ensemble d'apprentissage
Résultat :  $T$  l'arbre de décision
 $T = \emptyset$ ;
Déterminer l'attribut  $A$  offrant la meilleure discrimination;
 $T =$  Créer un noeud et l'étiqueter avec cet attribut;
 $T =$  Ajouter à  $T$  un arc au noeud créé pour chaque valeur de cet attribut et étiqueter les arcs;
pour chaque arc faire
     $D' =$  sous-ensemble de  $D$  obtenu en appliquant la discrimination par un test sur  $A$  pour  $arc$  ;
    si le point d'arrêt est atteint (sur  $D'$ ) alors
        |  $T' =$  créer une feuille et l'étiqueter avec la classe appropriée
    sinon
        |  $T' = DTBuild(D')$ ;
    fin si
     $T =$  ajouter  $T'$  à  $T$ ;
fin pour chaque
retourner  $T$ ;
```

**Algorithme 1 :** L'algorithme DTBuild

##### 13.2.1 Critères de l'algorithme

Les critères importants de l'algorithme ci-dessus sont :

- Choix de l'attribut discriminant.
- Ordre des attributs discriminants.
- Structure de l'arbre.
- Critères d'arrêt.
- Élagage de l'arbre.

Le critère clé de cet algorithme est la méthode de choix de l'attribut de discrimination (ou de *coupure*). Ce choix judicieux ainsi que l'ordre de discrimination doit permettre de réduire le nombre de comparaisons avant de classer un objet.

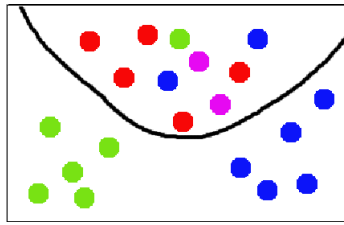


FIGURE 1 – Discrimination des instances en fonction de leurs attributs

### 13.3 Discrimination d'attributs et la fonction Entropie

Le caractère discriminant d'un attribut peut être quantifié par une fonction qui, après la discrimination, prend son minimum quand tous les exemples sont dans la même classe et qui prend son maximum s'ils sont équi-répartis sur plusieurs classes.

Ainsi, l'induction des arbres de décision est souvent basée sur des mesures issues de la théorie de l'information.

L'**entropie** est l'une de ces mesures. Elle mesure *l'incertitude* (d'attribution d'une classe à une instance) et permet de quantifier le caractère aléatoire d'une distribution de probabilités. Elle reflète en quelque sorte l'**hétérogénéité** des classes des instances arrivées jusqu'au au noeud  $p$ .

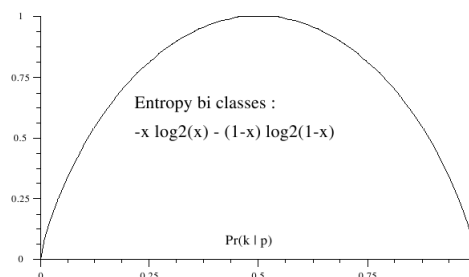
Lors de l'induction de l'arbre de décision, on utilisera l'Entropie pour mesurer l'incertitude relative à l'appartenance des objets aux différentes classes. Lorsque tous les objets appartiennent à une seule classe, l'incertitude est nulle (valeur minimum). Par contre, elle est maximum si les instances sont équi-réparties dans divers classes. (voir cours, chapitre 4).

Étant données une position  $p$  dans l'arbre et  $c$  classes différentes que l'on cherche à prédire, l'Entropie associée à  $p$  est donnée par :

$$H(p) = - \sum_{k=1}^c Pr[k|p] \times \log(Pr[k|p]) \quad (2)$$

Dans le cas simple d'une distribution normale,  $Pr[k|p]$  est une fréquence (une proportion).

La courbe de la fonction Entropie dans le cas d'un problème à deux classes est :



Lorsque l'entropie est nulle, toutes les instances de la position (noeud)  $p$  sont dans la même classe (ici, dans l'une des 2 classes).

#### Interprétation empirique de l'entropie :

L'entropie = incertitude = degré d'hétérogénéité = erreur *de désignation de la classe d'une instance*.

Par exemple, dans un cas **bi-classe** avec  $N$  instances :

- si la répartition est  $[N/2, N/2]$  (équi-répartition), on a une chance sur 2 de se tromper lorsque l'on donne la classe d'une instance (entropie maximale) : la classe d'une instance  $\in \{C_1, C_2\}$  (2 infos dans le cas binaire)
- si la répartition est  $[N, 0]$ , on a aucune chance de se tromper (entropie minimale) : la classe d'une instance  $\in C_1$  (ou  $C_2$  mais une seule des 2)
- Les autres répartitions se situent entre ces deux cas :  
Si la répartition est  $[N/3, 2N/3]$ , on a moins d'une (env. 0.9, calculé par  $H(p)$ ) chance sur 2 de se tromper lorsque l'on donne la classe d'une instance

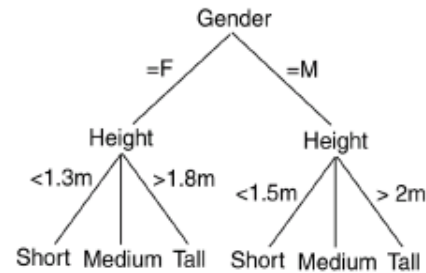
On constate bien que l'entropie n'est pas (directement) la quantité d'information nécessaire mais est en relation directe avec cette quantité.

En terme d'hétérogénéité, quand toutes les instances sont dans la même classe, cette mesure d'hétérogénéité=0. Pour  $[N/2, N/2]$ , elle est maximale et pour  $[N/3, 2N/3]$ , elle est entre les deux. L'entropie permet d'en mesurer la valeur.

### 13.4 Exercice de construction d'un arbre de décision

Soit la base (partielle) d'instances suivante.

Name	Gender	Height	Size
Kristina	F	1.6m	Medium
Jim	M	2m	Medium
Maggie	F	1.9m	Tall
Martha	F	1.88m	Tall
Stephanie	F	1.7m	Medium
Bob	M	1.85m	Medium
Kathy	F	1.3m	Short
Dave	M	1.7m	Medium
Sara	F	1.20m	Short
Worth	M	2.2m	Tall
Steven	M	2.1m	Tall
Debbie	F	1.8m	Medium
Todd	M	1.95m	Medium
Kim	F	1.9m	Tall
Amy	F	1.8m	Medium
Wynette	F	1.75m	Medium
Mike	M	1.25m	Short
...	...	...	...



Un arbre de décision pour cet exemple est donné à droite. Voir aussi la sous section 7.1) où la variable *Height* (taille) a été discrétisée.

### 13.5 Exercice optionnel : créez l'arbre !

Les arbres de décision sont produits par les méthodes **ID3** et **C4.5**. Les méthodes correspondantes dans WEKA sont **id3** et **j48**<sup>3</sup>.

### 13.6 Rappel Méthode ID3 et le calcul de l'Entropie

ID3 est une instance de l'algorithme générique DTBuild précédent. ID3 utilise une fonction de **gain** pour mesurer la qualité de la discrimination de l'ensemble  $D$  par un attribut  $A$  pouvant prendre  $s$  valeurs.

$$Gain(D, A) = H(D) - \sum_{i=1}^s P_i H(D_i) \quad (3)$$

où :

- $D_i$  est le sous-ensemble des exemples de  $D$  pour lesquels l'attribut  $A$  prend la valeur  $i$ .
- $P_i$  est la proportion d'éléments de  $D$  pour lesquels l'attribut  $A$  vaut  $i$ .

Enfin, ID3 utilise ces autres critères pour l'induction d'arbre de décision :

- Un noeud est terminal si tous les éléments associés à ce noeud sont de même classe ou si aucun test n'a pu être sélectionné.
- Pour déterminer l'attribut de classe d'une feuille (un noeud que l'on ne peut plus diviser), on prend la classe majoritaire des instances dans cette feuille.

3. par rapport à **C4.5** où C désigne le langage C, la version 4.8 de la méthode a été développée en Java

**Un exemple :** supposons un cas à 2 classes, un attribut  $A$  a trois valeurs  $V_1, V_2, V_3$  ( $s = 3$ ).  
L'ensemble  $D$  avant le partitionnement sur l'attribut  $A$  contient 14 instances dont 9 sont dans la 1e classe et 5 dans la 2e.

Ce qui veut dire :  $H(D) = H([9, 5]) = -9/14 \times \log(9/14) - 5/14 \times \log(5/14) = 0.94$

Ensuite, la discrimination de ces 14 instances selon les valeurs de l'attribut  $A$  est faite de la manière suivante :

- pour la valeur  $V_1$ , la répartition est de 2 instances pour la 1e classe et 3 pour la 2e ;
- pour la valeur  $V_2$ , la répartition est de 4 instances pour la 1e classe et 0 pour la 2e ;
- pour la valeur  $V_3$ , la répartition est de 3 instances pour la 1e classe et 2 pour la 2e ;

Ce qui donne :  $\sum_{i=1}^3 P_i H(D_i) = 5/14 \times H([2, 3]) + 4/14 \times H([4, 0]) + 5/14 \times H([3, 2])$

avec  $H([2, 3]) = -\frac{2}{5} \log \frac{2}{5} - \frac{3}{5} \log \frac{3}{5} = 0.971$ ,  $H([4, 0]) = 0$  et  $H([3, 2]) = H([2, 3]) = 0.971$

On a donc :  $\sum_{i=1}^3 P_i H(D_i) = 5/14 \times 0.971 + 4/14 \times 0 + 5/14 \times 0.971 = 0.693$

Enfin,  $Gain(D, A) = H(D) - \sum_{i=1}^s P_i H(D_i) = 0.94 - 0.693 = 0.247$

Voir également le cours (chapitre 4, Part 1.1) pour plus de détails et exemples.

### 13.7 L'algorithme C4.5

Le problème de la fonction *gain* est de privilégier les attributs ayant un grand nombre de valeurs possibles (attributs très "branchants", e.g. un ID-Code unique par transaction).

L'algorithme C4.5 corrige ce problème en pondérant le gain par une fonction *SplitInfo* qui pénalise les tests répartissant les éléments en un trop grand nombre groupes. Cette fonction est également appelée le *gain intrinsèque* de l'information sur un attribut.

$$SplitInfo(D, A) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|} \quad (4)$$

où  $D_i$  est le sous-ensemble des éléments de  $D$  de taille  $|D_i|$  valant  $i$  pour l'attribut  $A$ .

C4.5 utilise ainsi une fonction **GainRatio** pour le critère de sélection, définie comme suit :

$$GainRatio(D, A) = \frac{Gain(D, A)}{SplitInfo(D, A)} \quad (5)$$

Voir en Annexes pour les apports de C4.5 par rapport à ID3.

N.B. : par rapport à l'entropie qui s'intéresse aux proportions et les classes des instances "descendues" dans un noeud, *SplitInfo* ne regarde pas les classes des instances placées dans un noeud mais seulement leur nombre. *SplitInfo* mesure en fait la capacité de dispersion d'un attribut alors que l'entropie mesure la capacité de répartition des instances dans différentes classes.

### 13.8 Conclusions sur les arbres de décision

#### — Avantages :

- Compréhensible pour l'utilisateur.
- Permet de générer très facilement des règles (il suffit de "lire" les tests sur les branches).

#### — Inconvénients :

- Peut conduire à un "overfitting" des données.
- Difficultés à traiter les attributs numériques (la discrétisation provoque toujours une perte d'information).
- L'influence de l'heuristique de l'élagage (*pruning*).
- Les faiblesses (théoriques) de la mesure Entropie (GINI, &cet.).

## 14 Annexes-2 : exploitation de l'extracteur WEKA

WEKA est un utilitaire Open Source d'apprentissage automatique écrit en Java et développé à l'université de Waikato en Nouvelle Zélande (Site Web : <http://www.cs.waikato.ac.nz/ml/weka/>).

Sur la page <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>, téléchargez la version de développement (*Developer version*) pour votre machine.

Weka est réalisé en Java (archive jar) et existe pour les plate-formes Windows, Linux et Mac.

Pour utiliser Weka, vous aurez besoin de la **machine abstraite Java**. Sous Windows ou MacOS, le mieux serait de récupérer la version contenant la machine virtuelle Java qui convient.

☞ Les essais et tests de ce documents sont réalisés avec la version 3.5.6 de Weka.

WEKA implante un grand nombre d'algorithmes d'apprentissage automatique.

### Pour lancer Weka

- **Sous Linux / MacOS** : placer vous dans le répertoire d'installation puis tapez :

```
java -jar weka.jar
```

ou bien cliquer sur l'archive **weka.jar**.

☞ Pour demander plus de mémoire (BD volumineuse) : `java -Xmx1000M -jar weka.jar`

- **Sous Windows** : récupérer sur le même site la version AVEC la machine virtuelle Java.

Installer puis lancer l'exécutable Weka dans le répertoire d'installation (ou sur *Weka* dans les "programmes")

☞ Il se peut que la façon de lancer Weka se modifie d'une version à une autre. Vérifier la présence d'un fichier exécutable qui vous dispensera de lancer java à la main (voir avec l'enseignant).

### 14.1 Format de fichiers

Le format de fichier utilisé par Weka (**.arff**) est constitué comme suit :

- Une entête (*header*) qui contient :
  - le nom de la relation (au sens "Base de Données" relationnelle) : `@RELATION <relation_name>`
  - une liste d'attributs et leurs types : `@ATTRIBUTE <attribute_name> <datatype>` où *datatype* peut prendre les valeurs **numeric**, **string** où un ensemble énuméré de valeurs de la forme {*val1*, *val2*, *val3*}.

→ Un exemple de header pourrait être :

```
% Commentaire : ici, on écrit du blabla
@RELATION ventes
@ATTRIBUTE num_vente numeric
@ATTRIBUTE nom_vendeur string
@ATTRIBUTE detail_facture numeric
@ATTRIBUTE total numeric
@ATTRIBUTE paiement {especes, CB, cheque}
```

- Un corps qui contient les données. Le début du corps est marqué par le mot clé **@DATA**. Les différentes valeurs d'un tuple (une instance) sont séparés par une virgule. Les valeurs inconnues sont spécifiées par un "?".

Un exemple d'une section "DATA" pourrait être celui-ci :

```
@DATA
1, 'Corine', 120, 25.34, especes
2, 'Jean-Luc', 124, 32, CB
3, 'Michel', 126, 58.25, CB
....
```

### 14.2 Préparation des données sous Weka

Weka dispose d'un certain nombre d'outils d'exploration, filtrage et préparation de données. Voir onglets "*preprocess*" et "*select attributes*". Quelques filtres seront présentés ci-dessous.

**Nota Bene** : Après l'application de tout filtre expliqué dans cette partie (discrétisation, échantillonnage, anonymisation, normalisation, etc.), les données sont modifiées en mémoire et peuvent être utilisées par exemple pour une classification. Les données d'origine ne sont plus en mémoire.

Si vous voulez par exemple essayer une autre discrétisation, **il faudra recharger** le fichier initial.

## 14.3 Suppression des instances par leur indice

Sous WEKA, on peut supprimer des instances de la DB suivant leur indice (numéro d'ordre). Dans l'exemple ci-dessous appliqué à *banque-app.arff*, on se donne l'objectif d'éliminer les instances allant de 3 à 16.

Charger le fichier de données *banque-app.arff*.

- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **RemoveRange**.
- Comme pour toute commande, dans la fenêtre qui s'affiche, le bouton **More** nous dit davantage sur cette commande.

En particulier, ici, on remarquera :

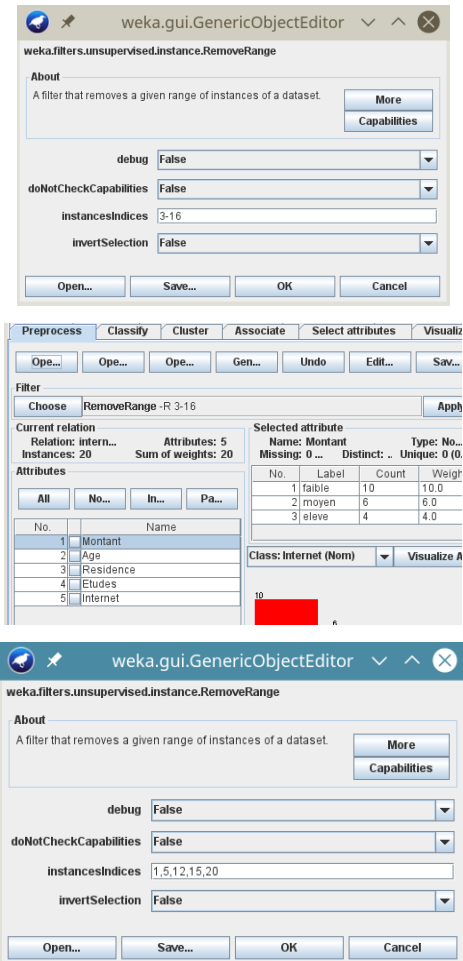
- **instancesIndices** : l'intervalle d'indice ou les numéros des instances à supprimer. On donnera l'attribut qui servira de filtre. Pour nous, ce sera **1** (premier attribut = **Montant**). Cette information est visible dans l'onglet *Preprocess* (où on est actuellement), dans la zone des attributs de la base de données *banque-app.arff* chargée. Voir figure ci-contre.
- **invertSelection** : inverser le sens de la sélection : au lieu de supprimer les instances désignées, on les garde et on supprime les autres.

Les 3 figures ci-contre :

- Les 2 premières montrent les options pour supprimer les instances allant de 3 à 16 (séparés par un trait).

→ Cliquer sur **Apply** pour que sur les 20 instances de la BD, il nous reste 6.

- La 3e figure montre les options pour supprimer les instances dont on précise les numéros, séparés par des virgules.



## 14.4 Anonymisation de données sous Weka

Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance*. Puis choisir :

**Datafly** : anonymisation des données (pour ne pas pouvoir identifier des informations éventuellement sensibles telle qu'un nom, etc...)

## 14.5 Discretisation dans Weka

Dans WEKA, certaines méthodes comme ID3 (mais pas C4.5) n'acceptent que des données de type nominal. Dans ce cas, lorsque l'on charge un fichier ".arff" avec des données numériques (ou mixtes = mélange numérique-nominal), on peut procéder à une discrétisation des attributs numériques.

**Manipulation** : Dans l'onglet **Preprocess**, charger les données de l'exemple "météo" (fichier *weather.arff*) qui contient des attributs mixtes (cf. 2e tableau ci-dessous).

Cliquez ensuite sur *Filter/Choose* (sous *Open file*) puis choisir *filters/unsupervised/attribute/Discretize*.

Cliquer ensuite dans la ligne de commande de discrétisation qui s'est affichée (*Discretize -unset ...*) pour faire afficher la fenêtre des paramètres.

Par défaut, Weka discrétise TOUS les attributs (paramètre *attributeindices*). Si vous voulez discrétiser un attribut de votre choix, ou bien appliquer des discrétisations différentes, modifier le paramètre *attributeindices*. Par exemple, la valeur *2* ne discrétise que le 2e attribut et *2,4* applique une discrétisation aux 2e et 4e attributs. Les valeurs *first-* et *-last* vous permettent de désigner le 1er et le dernier attributs.

Le paramètre *bins* désigne le nombre d'intervalles souhaités. La valeur de *bins* est fixée à 10 par défaut.

Pour les attributs numériques de l'exemple "météo" chargé, tester 3 intervalles (*bins*) puis faites afficher les partitions obtenues en cliquant sur le nom de l'attribut.

Pour l'application de la discrétisation, appuyer sur **Apply** à droite de la ligne de commande.

### 14.5.1 Un exemple de discrétisation

Chargez le fichier *weather.arff* (météo). Il y a deux attributs numériques *temperature* et *humidity*.

Dans la même fenêtre, devant *Filter* (dans la partie supérieure de la fenêtre), cliquez sur *Choose*. Sélectionner *filters/unsupervised/attribute/Discretize*. Vous verrez devant le bouton *Choose* s'afficher :

**Discretize ...**

**-B 10** : 10 partitions

**-M -1.0** : poids des instances par intervalle (ici rien, cliquer sur more pour en savoir plus))

**-R first-last** : tous les attributs.

Cliquez dans cette zone des paramètres, fixer le nombre de partions (bins) à 3, changez *attributeindices* en 2,3 (seuls les 2 attributs numériques concernés). Fermer cette petite fenêtre des paramètres puis cliquez sur *apply* tout à droite de cette zone.

Maintenant, dans la zone où sont affichés la liste des attributs, cliquez sur *temerature* ou *humidity* pour voir les partitions.

## 14.6 Conversion des données

### 14.7 Conversions de données transactionnelles

Les données transactionnelles (par exemple un ticket de caisse pour des achats ) sont souvent stockés dans des BDs avec un identifiant (ID). Par exemple un numéro d'ordre de client (ou un nom). Cette information joue le rôle d'une clef discriminante : une clé qui désigne un ensemble d'informations de manière unique.

☞ **La plupart des méthodes** d'apprentissage échouent (et produisent un modèle inutilisable ; voir cours) lorsque les données contiennent ce type d'attribut.

Les transactions (concernant par exemple un numéro de client) peuvent être scindées sur plusieurs lignes de la table (chacune avec le même ID). Une BD. dans ce format doit être convertie en une ligne par transaction avant de pouvoir être utilisée par un classificateur ou d'en extraire des règles d'association, clustering, etc sous Weka .

Le filtre de dé-normalisation (*denormalize*) de Weka peut effectuer ce genre de processus d'aplatissement. Ce filtre nécessite que :

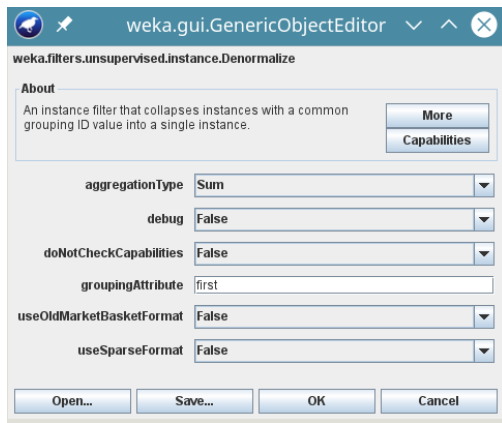
- 1- les données contiennent un champ ID unique qui identifie chaque transaction distincte, et que
- 2- les données soient déjà triées dans l'ordre de cet ID.

**Exemple** : la base de données (à gauche) et le fichier de données de Weka (*exemple\_numerique.arff*) au milieu et le fichier (*exemple\_nominal.arff*) avec les attributs numériques transformés en catégoriels (à droite) :

User	ItemID	Sequence	TimeSpent	@relation exemple_numerique	@relation exemple_nominal
1	1	1	5	@attribute User numeric	@attribute User numeric
1	2	2	1	@attribute ItemID numeric	@attribute ItemID {1,2,3,4,5,6,8 }
1	5	3	8	@attribute Sequence numeric	@attribute Sequence {1,2,3,4,5 }
1	6	4	12	@attribute TimeSpent numeric	@attribute TimeSpent {1,2,3,5,7,8,12 }
1	8	5	2	@data	@data
				1, 1, 1, 5	1,1,1,5
				1, 2, 2, 1	1,2,2,1
				1, 5, 3, 8	1,5,3,8
				1, 6, 4, 12	1,6,4,12
2	1	1	7	1, 8, 5, 2	1,8,5,2
2	2	2	3	2, 1, 1, 7	2,1,1,7
2	3	3	3	2, 2, 2, 3	2,2,2,3
2	4	4	2	2, 3, 3, 3	2,3,3,3
2	5	5	7	2, 4, 4, 2	2,4,4,2
				2, 5, 5, 7	2,5,5,7

On applique un premier filtre pour transformer ces données numériques en nominales (cf. ci-dessus). Ce qui a donné le fichier *exemple\_nominal.arff* (ci-dessus, à droite).

Dans un second temps, il faudra veiller à ce que **les données contenu dans ce fichier (la section @data) sont triées selon ID**, . Puis on applique le filtre *denormalize* (à gauche de la figure ci-dessous) et on obtient le fichier *exemple\_final.arff* (à droite).



```
@relation exemple.final
@attribute User numeric
@attribute ItemID.1 {f,t}
@attribute ItemID.2 {f,t}
@attribute ItemID.3 {f,t}
@attribute ItemID.4 {f,t}
@attribute ItemID.5 {f,t}
@attribute ItemID.6 {f,t}
@attribute ItemID.8 {f,t}
@attribute Sequence.1 {f,t}
@attribute Sequence.2 {f,t}
@attribute Sequence.3 {f,t}
@attribute Sequence.4 {f,t}
@attribute Sequence.5 {f,t}
@attribute TimeSpent.1 {f,t}
@attribute TimeSpent.2 {f,t}
@attribute TimeSpent.3 {f,t}
@attribute TimeSpent.5 {f,t}
@attribute TimeSpent.7 {f,t}
@attribute TimeSpent.8 {f,t}
@attribute TimeSpent.12 {f,t}

@data

1,t,t,f,f,t,t,t,t,t,t,t,t,t,f,t,f,t,t
2,t,t,t,t,t,f,f,t,t,t,t,t,f,t,f,t,f,f
```

Pour pouvoir appliquer une méthode telle que *A Priori* (règles d'association), il faudra se "débarrasser" du premier attribut qui est numérique (l'ID). Il suffit pour cela de sélectionner, dans l'onglet *Preprocess*, l'attribut *User* (cocher la case) puis faire "remove". Maintenant, on peut appliquer *A Priori* et obtenir les règles d'association.

## 14.8 Conversions numériques vers nominales

- Charger le fichier de données *credit-g.arff*.
- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute* et dans la liste proposée, choisir **NumericToNominal**.
- Utiliser le bouton **More** pour en savoir davantage sur cette commande.

En particulier, ici, on remarquera :

- *attributIndices* : le numéro (d'ordre) de l'attribut concerné.

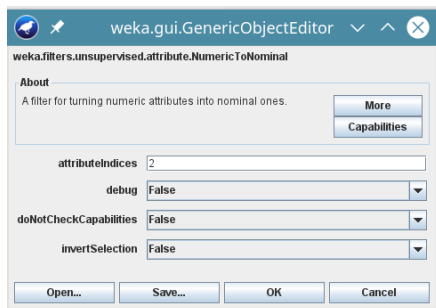
On choisira pour cet exemple le 2e attribut *duration* qui est numérique (v. fig ci-contre).

On peut également spécifier une plage de numéros d'attributs. Si vous voulez transformer tous les attributs numériques en nominaux, conserver la valeur par défaut *first-last* proposée par Weka qui appliquera la méthode à tous les attributs numériques.

- *invertSelection* : inverser le sens de la sélection : au lieu de transformer les attributs sélectionnés, transformer les autres.

Statistic		Value
Minimum		4
Maximum		72
Mean		20.903
StdDev		12.059

Ci-dessous, les options de la méthode pour notre exemple (à gauche) et le résultat de l'application de la méthode (à droite) : l'attribut numéro 2 (*duration*) est devenu nominal.



No.	Label	Count	Weight
1	4	6	6.0
2	5	1	1.0
3	6	75	75.0
4	7	5	5.0
5	8	7	7.0

La commande sous Linux, dans le répertoire où il y a *weka.jar* :

```
java -cp ./weka.jar weka.filters.unsupervised.attribute.NumericToNominal -R 2-last -i exemple_num.arff > exemple_nom.arff
```

## 14.9 Autres méthodes de préparation / conversion de données

- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute*.
  - **Normalize** : normalise les données numériques (sauf la classe, si on le précise dans la commande). Le résultat sera des valeurs réelles dans l'intervalle  $[0, 1]$ .
    - ☞ Si vous choisissez dans les options de ce filtre *scale* = 2.0 et *translation* = -1.0, vous obtiendrez des valeurs normalisées réelles dans l'intervalle  $[-1, +1]$ .
  - **Standardize**, etc. Voir les autres filtres ...

## 14.10 Échantillonnage des données

Parfois, la BD est trop large et l'application d'une méthode peut prendre beaucoup de temps (cf. BD **adults**). Dans ce cas, on peut procéder à un re-échantillonnage correct (stratifié). Suivez les étapes suivantes :

- On va échantillonner 10%
  - Dans l'onglet "preprocess", choisir "filter/supervised/instances/Resample"
  - Cliquer sur apply
  - Choisir 10%, les autres options par défaut
  - Sauvegarder éventuellement le résultat
- Vous pouvez choisir l'échantillonnage avec ou sans remise.
- L'option "biasToUniformClass" : Si 0, on considère la *classe* telle qu'elle est. Si 1, Weka s'arrange pour que la distribution de la valeur de la *classe* soit uniforme dans l'échantillon.

On peut obtenir les mêmes résultats via "filter/unsupervised/instances/StratifiedRemoveFolds" pour avoir 10% des données.

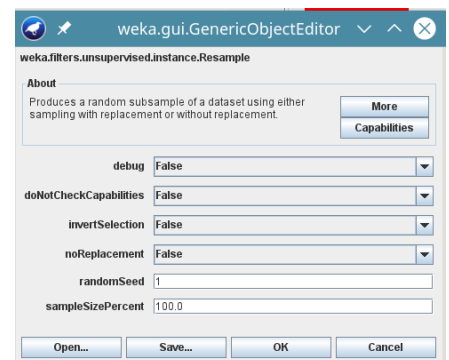
## 14.11 Échantillonnage de x%

Pour procéder à l'échantillonnage de  $x\%$  des données :

- Charger le fichier de données *banque-app.arff*.
- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **Resample**.
- Utiliser le bouton **More** pour en savoir davantage sur cette commande.

En particulier, ici, on remarquera :

- **noReplacement** : échantillonnage avec **remise** (par défaut). Mettre cette valeur à True pour un échantillonnage sans remise.
- **sampleSizePercent** : le pourcentage à échantillonner. La valeur par défaut de 100% permet simplement de brasser les instances (si échantillonnage sans remise). Par contre 100% avec remise produira une nouvelle base de données en mémoire où certaines instances seront répétées. Ce qui peut être néfaste dans certains cas.
- **invertSelection** : inverser le sens de la sélection : les  $x\%$  échantillonnés seront écartés et on conserve le reste :  $(1 - x)\%$ .
- **randomSeed** : par défaut =1. Le *seed* est la *graine* utilisée pour initialiser une séquence aléatoire. La variation de cette valeur permet de diversifier les  $x\%$  échantillonnés. En théorie, deux tirages dans les mêmes conditions avec la même graine produisent la même séquence de  $x\%$ .



Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance*, on peut disposer également de :

- **SpreadSubsample** : échantillonnage aléatoire. Au besoin, charger le package correspondant.  
On peut spécifier le maximum de "spread" (répartition de la classe) entre la valeur la plus commune de la classe et celle la plus rare. On peut par exemple demander un "spread" de 2 :1 pour avoir deux fois plus souvent la classe la plus commune.  
→ Ce paramètre ("distributionSpread") sera donc : 0 = rien changer ; 1 = distribution uniforme des valeurs de la classe dans l'échantillon ; 10 = on veut voir la classe la plus commune de se répéter au maximum dix fois plus que la plus la plus rare. L'option "maxCount" permet de contrôler le nombre d'occurrence maximum d'une classe (0 = sans limite.)
- Voir les autres méthodes concernant les instances.

## 14.12 Échantillonnage selon valeur d'attribut

Nous utilisons ici la BD *banque-app.arff* pour illustrer ces exemples.

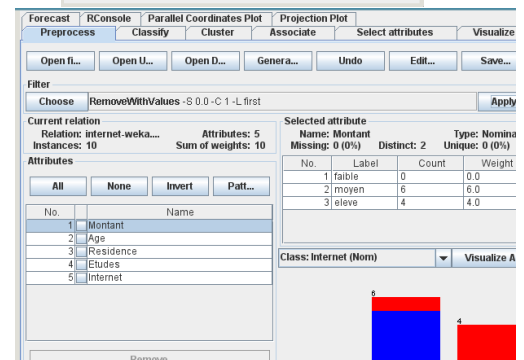
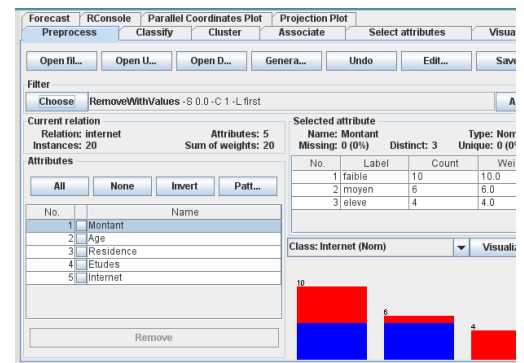
Sous WEKA, on peut supprimer des instances de la DB suivant la valeur d'un attribut. Dans l'exemple ci-dessous appliqué à *banque-app.arff*, on se donne l'objectif d'éliminer les instances dont la valeur de l'attribut **Montant** est égale à **Faible**.

- Charger le fichier de données *banque-app.arff*.
- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **RemoveWithValues**.
- Comme pour toute commande, dans la fenêtre qui s'affiche, le bouton **More** nous dit davantage sur cette commande.

En particulier, ici, on remarquera :

- **attributeIndex** : l'attribut qui servira de filtre. Pour nous, ce sera **1** (premier attribut = **Montant**). Cette information est visible dans l'onglet *Preprocess* (où on est actuellement), dans la zone des attributs de la base de données *banque-app.arff* chargée. Voir figure ci-contre.
- **nominalIndices** : la rang de la valeur utilisée pour la sélection sur les attributs nominaux (catégoriel, non numérique). Cette valeur va de *first* à *last* et dans notre cas, ce sera *first* car **Faible** est la première valeur de l'attribut *Montant* (ce rang est visible dans l'onglet *Preprocess*, si vous sélectionnez l'attribut *Montant*). Voir figure ci-contre.
- **splitPoint** : un seuil numérique à utiliser pour les attributs numériques. Les instances dans lesquelles la valeur de cet attribut est inférieure à ce seuil seront sélectionnées.
- **invertSelection** : inverser le sens de la sélection : au lieu de supprimer les instances qui satisferont nos choix, on les garde et on supprime les autres.
- **matchMissingValues** : si True, les valeurs manquantes seront considérées comme satisfaisant le filtre. Cette option est indépendante du choix **invertSelection** ci-dessus.
- **modifyHeader** : si True, et si l'attribut en question est nominal (*Montant* dans notre exemple), alors enlever toute trace de la valeur **Faible** de cet attribut dans l'entête de la base de données (cf. le format des fichiers arff de WEKA). Si tel est notre choix, la BD. sera modifiée comme si la valeur *Faible* de l'attribut *Montant* n'avait jamais existé.

Après avoir appliqué ce filtre, on se retrouve avec seulement 10 instances (contre 20 dans la BD initiale, voir figure) : les instances dont Montant=Foible ont été supprimées.



### 14.13 Filtre suivant les attributs

Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute*, on trouve les filtres (importants) suivants

- **AttributeSelection** : on y précisera l'évaluateur (comment évaluer les sous-ensembles d'attributs) ainsi que la méthode de recherche de ces sous-ensembles.
- **ClassOrder** : modifier l'ordre d'apparition des classes dans la BD.
- **MergeNominalValues** : pour fusionner certains attributs nominaux (catégoriels).
- **NominalToBinary** : transformer les attributs nominaux en binaire. Si un attribut a par exemple trois valeurs énumérées (soit *v1*, *v2* et *v3*), pour ces 3 valeurs, 3 nouveaux attributs seront créés dont les noms seront "*=v1*", "*=v2*" et "*=v3*". Pour toute instance, un seul de ces trois attributs sera vrai ("*1*"), les deux autres seront *=0*.
- **PLSFilter** (packages à installer éventuellement) : exécute la méthode *Partial Least Square Regression* sur les instances et produit une matrice de prédiction appelée *matrice beta*. Par défaut, les numériques seront centrés.

### 14.14 supprimer les instances avec "missing values"

Comment supprimer les instances avec missing values ?

Un exemple :

1. Charger la BD. *weather-nominal.arff* et dans l'onglet "Preprocessing", noter le nombre d'instances (14) puis choisir le bouton "edit" pour éditer la base de données,
2. Remplacer dans une des colonnes (sauf la colonne de décision "play") 3 valeurs prises au hasard par rien (missing value). Pour insérer "rien", cliquer sur une des cases par exemple dans la colonne "Temperature" et choisir aucune valeur !,
3. Cliquer sur "OK" pour sauvegarder vos changements,
  - ➔ La BD. sur le disque n'est pas touchée, seule la BD. en mémoire est modifiée,
4. Toujours dans "PreProcessing", dans la partie "filtre", cliquer sur "choose" et choisir "Multifilter" qui est disponible immédiatement (sans rentrer dans aucun dossier),

5. Une fois "multifilter" choisi, cliquer dans la zone de la commande et aller en bas (la 3e zone) : "filtres" (qui contient par défaut "2 weka.filters.Filter") et cliquer dessus,
6. Dans la petite fenêtre qui apparaît, par défaut, "AllFilter" est inséré,
7. Cliquer dans cette petite fenêtre sur "Choose" au dessus pour spécifier notre filtre,
8. On arrive dans la fenêtre des paramètres de ce filtre qui est la fenêtre de choix de filtres de Weka. Choisir le filtre "Unsupervised/instances/RemoveWithValue" puis faire "Add".
9. A partir de maintenant, à chaque fois qu'on clique sur "Add", le même filtre sera inséré dans la petite fenêtre,
  - Pourquoi refaire "Add" ? Parce que chaque filtre va concerner une colonne et donc si on veut enlever toutes les instances avec des valeurs manquantes dans une colonne différente on doit faire "Add" autant de fois. Par exemple, si vous avez créé des valeurs manquantes dans 3 colonnes différentes, vous devez faire "Add" 3 fois.
10. On édite le 1er filtre en cliquant sur "RemoveWithValue" et dans la fenêtre des paramètres du filtre qui s'affiche, faire les modifications suivantes :
  - attributeIndx : mettre l'indice de la colonne concernée : p. ex. on met 3 car on a introduit des missing values dans la colonne 3,
  - debug True si on veut
  - inversSelection : True (**important**)
  - matchMissingValue : True (**important**)
  - laisser le reste
11. Cliquer sur OK puis fermer la petite fenêtre puis OK dans la dernière fenêtre encore ouverte.
12. Maintenant, appuyer sur "Apply" en face du "Filter",
13. Le filtre s'applique, les instances avec des valeurs manquantes sont supprimées, ce qui peut être contrôlé en suivant le nombre d'instances dans la BD relevé au début.
14. Il faut répéter l'étape 10... 12 pour chaque colonne concernée. Weka n'accepte pas qu'on donne plusieurs valeurs dans le champ "attributeIndex" Et donc on doit répéter autant de fois ces actions que de colonnes concernées.
15. A la fin, il nous reste seulement les instances sans missing value.

## 15 Annexes-3 : Compléments

### 15.1 A propos de C4.5

**C4.5 apporte** d'autres améliorations à ID3 :

- il peut gérer des attributs à valeurs continues, en ajoutant dynamiquement un attribut à valeurs discrètes qui partitionne les valeurs de l'attribut continu. Ainsi, pour avoir des tests binaires sur un attribut  $A$  qui prend les valeurs  $a_1, \dots, a_k$ , C4.5 considèrera les tests  $A < \frac{1}{2}(a_i + a_{i+1}), \forall i \in \{1, \dots, k-1\}$ . Ce qui place les frontières des partitions entre deux valeurs successives.
- il peut traiter des ensembles d'exemples dont certains attributs ont une valeur indéterminée. Pour la phase d'induction, on considèrera que la valeur de cet attribut suit la même distribution que celle des valeurs connues ; le calcul du gain se fera alors uniquement sur les exemples ayant une valeur définie pour l'attribut considéré. Pour la phase d'utilisation de l'arbre, si on arrive sur un test auquel on ne peut pas répondre, alors on choisira la classe majoritaire à cette position.
- Pour éviter d'avoir des arbres de trop grande taille, on peut **élaguer** l'arbre de décision, c'est à dire remplacer un sous-arbre entier par un noeud terminal qui a pour label la valeur la plus commune de l'attribut de classe dans le sous-arbre. L'élagage ne peut se faire que dans le cas où l'erreur commise dans le sous-arbre est plus grande que l'erreur commise en le remplaçant par une feuille. Une méthode pour évaluer cette erreur est d'évaluer la précision de l'arbre sur un jeu d'exemples (jeu de test) différents de celui qui a permis de construire l'arbre. Néanmoins, cette méthode est difficile à mettre en oeuvre dans le cas où l'on dispose d'un jeu d'exemples réduit. Dans ce cas, on peut utiliser une méthode d'élagage à base de règles :
  - Construction de l'arbre à partir de l'ensemble des exemples.
  - Conversion de l'arbre en un ensemble de règles (une règle pour chaque chemin de la racine à une feuille).
  - Élagage des règles en supprimant (une partie) des préconditions.
  - Tri des règles selon l'estimation de leur précision.

Par exemple, si grâce à l'arbre de décision sur l'exemple "météo", on obtient la règle :

Si ( $Aspect = ensoleillé$ )  $\cap$  ( $Temperature = faible$ ) Alors ( $jouer = Oui$ )

Cette règle peut être élaguée en enlevant une des préconditions ( $Aspect = ensoleillé$ ) ou bien ( $Temperature = faible$ ). Pour décider de l'élagage d'une règle, C4.5 évalue les performances des règles non pas sur un ensemble test, mais sur l'ensemble d'apprentissage lui-même.

On note que dans ce cas extrême, les résultats ne sont pas à l'abri d'un *Overfitting*.

## 15.2 Classification Bayésienne (compléments)

On suppose une distribution de probabilités  $P$  sur  $\Pi$ . On suppose que  $D$  est discret.  $Pr(d)$  est la probabilité qu'un individu ait la description  $d$ .

$Pr(k)$  est la probabilité qu'un individu appartienne à la classe  $k$ .

$Pr(d|k)$  : probabilité qu'un individu appartenant à la classe  $k$  ait  $d$  pour description.

$Pr(k|d)$  : probabilité qu'un individu qui a pour description  $d$  appartienne à la classe  $k$ .

### 15.2.1 Classe majoritaire $C_{maj}$

Pour toutes les descriptions  $d$ , on choisit la classe majoritaire, c'est à dire la classe  $k$  telle que  $Pr(k)$  soit maximum.

### 15.2.2 Maximum de vraisemblance $C_{mv}$

A chaque description  $d$ , on choisit la classe  $k$  pour laquelle cette description est la plus probable, i.e. celle telle que  $Pr(d|k)$  est maximum.

### 15.2.3 Classification bayésienne $C_{Bayes}$

Pour chaque description  $d$ , la méthode de classification Bayésienne consiste à choisir la classe  $k$  qui maximise  $Pr(k|d)$ .

#### Exemple :

Notre population  $\Pi$  sera ici un ensemble de patients. Ces patients doivent être répartis en deux classes  $S$  (Sain) et  $M$  (Malade). Les individus sont décrits par des attributs :  $T$  qui est vrai si l'individu a une tension anormale et  $C$  si l'individu a un taux de cholestérol anormal. On supposera les attributs  $T$  et  $C$  indépendants. On a les données suivantes :

classe $k$	$S$	$M$
$Pr(k)$	0.7	0.3
$Pr(T k)$	0.25	0.7
$Pr(C k)$	0.4	0.7

$$P(M|C) = \frac{P(C|M)P(M)}{P(M)P(C|M) + P(S)P(C|S)} = 0.43$$

De même,  $P(S|C) = 0.57$ ,  $P(M|T) = 0.55$ ,  $P(S|T) = 0.45$ . La méthode de classification Bayésienne associera donc la classe Malade aux individus ayant une tension anormale et la classe Sain aux individus ayant un cholestérol trop important.

### 15.2.4 Qualité de l'approximation

Soit  $C$  une fonction de classement, l'erreur  $E(d)$  pour une description  $d$  est la probabilité qu'un élément de la population  $\Pi$  de description  $d$  soit mal classé.

$$E(d) = P(Y \neq C(X)|X = d) \quad (6)$$

L'erreur de classification  $E(C)$  d'une fonction de classement est la moyenne pondérée sur les descriptions  $d$ , i.e. :

$$E(C) = \sum_{d \in D} E(d)P(X = d) \quad (7)$$

**Question.** Calculer les probabilités d'erreur de chacune des règles  $C_{maj}$ ,  $C_{mv}$  et  $C_{Bayes}$  sur l'exemple précédent.

**Théorème** L'ensemble  $\Pi$  étant probabilisé, la règle de décision de Bayes est celle dont l'erreur est la plus faible.

#### Preuve

$$E(d) = P(Y \neq C(d)|X = d) = 1 - P(Y = C(d)|X = d) \quad (8)$$

Or,  $C_{Bayes}$  associe à  $d$  la classe  $k$  qui maximise  $P(X = k|Y = d)$  et donc minimise  $E(d)$ . Donc,  $\forall C$ , on a  $E(C_{Bayes}) \leq E(C)$ .

## 15.3 Élagage : en savoir plus

N.B. : Le chapitre 6 du cours traite de cette question en détail.

### 15.3.1 A propos d'élagage dans C4.5

Une méthode d'élagage peu orthodoxe!!

La méthode classique d'apprentissage par arbres de décision consiste à construire un arbre à l'aide d'un échantillon d'apprentissage puis à l'élaguer à l'aide d'un échantillon test. L'idée sous-jacente est que la phase d'élagage doit permettre d'améliorer l'erreur réelle et que seul l'échantillon test permet de l'estimer de manière à peu près fiable.

Mais cette idée ne vaut que si l'on dispose de suffisamment d'exemples pour la première phase : un arbre dont l'erreur est catastrophique ne donnera jamais de bons résultats, quelque soit l'élagage qu'on lui fera subir!

Une idée récurrente consiste à apprendre avec tous les exemples disponibles et à élaguer avec ces mêmes exemples. Mais cette idée ne peut fonctionner qu'à condition de ne pas se baser sur l'erreur apparente calculée sur l'ensemble d'apprentissage pour estimer l'erreur réelle.

Quinlan propose la méthode suivante dans C4.5 :

On introduit un paramètre de confiance CF (par défaut, ce paramètre vaut 25). Pour chaque feuille de l'arbre, notons N le nombre d'exemples qu'elle couvre et E le nombre d'erreurs de classification qu'elle induit dans l'échantillon. Soit p la probabilité pour qu'un nouvel exemple soit mal classé par cette feuille. La quantité E/N est donc un estimateur de p. Pour tenir compte du fait que l'arbre construit n'est pas indépendant des données, nous allons supposer que cet estimateur est très optimiste. **Mean rel. region size** (la moyenne relative de la largeur de l'intervalle de confiance à un niveau de 95%). Plus précisément, soit  $E_p$  une variable aléatoire de loi binomiale de paramètres (N,p). C'est-à-dire que  $Pr(E_p = k) = C_N^k p^k (1-p)^{N-k}$  pour  $0 \leq k \leq N$

Nous posons  $p(E, N) = \max\{p | Pr(E_p \leq E) \geq CF\}$ . Nous prendrons  $p(E, N)$  comme valeur estimée de l'erreur réelle pour cette feuille.

**Exemple :** supposons qu'une feuille couvre  $N = 4$  exemples et supposons qu'elle induise une erreur ( $E = 1$ ). On prend  $CF = 25$ .

On a :  $p(E, N) = \max\{p | Pr(E_p \leq 1) \geq 0,25\}$   
 $= \max\{p | Pr(E_p = 0) + Pr(E_p = 1) \geq 0,25\} = \max\{p | (1-p)^4 + 4p(1-p)^3 \geq 0,25\}$

Pour cet exemple, on trouve  $p \simeq 0,54$ . Autrement dit, on estime par ce procédé l'erreur réelle pour cette feuille à 54 (au lieu des 25 fournis par l'erreur apparente).

Le reste est plus classique : on calcule l'erreur réelle estimée d'un arbre en faisant une somme pondérée des erreurs réelles estimées de ses fils.

Par exemple, si un noeud A a trois fils A1, A2 et A3, si le nombre d'exemples couverts par chacun de ces fils est respectivement de N1, N2 et N3, et si les erreurs réelles estimées pour chacun de ces fils sont e1, e2 et e3 alors l'erreur réelle estimée de A sera de  $(N1.e1 + N2.e2 + N3.e3)/(N1 + N2 + N3)$ .

Pour élaguer un arbre, on applique la démarche suivante :

*Tant qu'il existe un sous-arbre que l'on peut remplacer par une feuille sans faire croître l'erreur réelle estimée : élaguer ce sous-arbre.*

**Application :** On considère un espace de description comprenant deux attributs **adoption** et **éducation** pouvant prendre chacun trois valeurs : y, n et u.

On suppose que l'attribut cible est binaire et que ses valeurs sont A et B.

On considère l'arbre suivant :

```
adoption = y : A (0;151)
adoption = u : A (0;1)
adoption = n :
education = n : A (0;6)
education = y : A (0;9)
education = u : B (0;1)
```

Chacun des couples (0; 151), ..., est de la forme (E;N). On donne :

```
p(0;6)=0,206 ;
p(0;9)=0,143 ;
p(0;1)=0,750 ;
p(1;6)=0,159 ;
p(0;151)=0,009 ;
p(1;168)=0,016.
```

Procédez à la décision d'élagage.