

A decorative graphic on the right side of the page. It features three overlapping circles of varying sizes, each composed of concentric layers of different shades of blue. Two thin, light blue lines intersect at the top left and extend diagonally across the page, framing the circles.

# Mini-projet d'Apprentissage

Abres de Décision

Le but de ce MiniProjet est d'exploré l'utilisation du logiciel libre Weka dans une application JAVA développée en ECLIPSE qui construisse des arbres de décision à partir de données.

**Nguyen Thi Hong Hiep – Le Bao Anh – Promo13 - IFI  
10 Nov 2008**

## Table des Matières

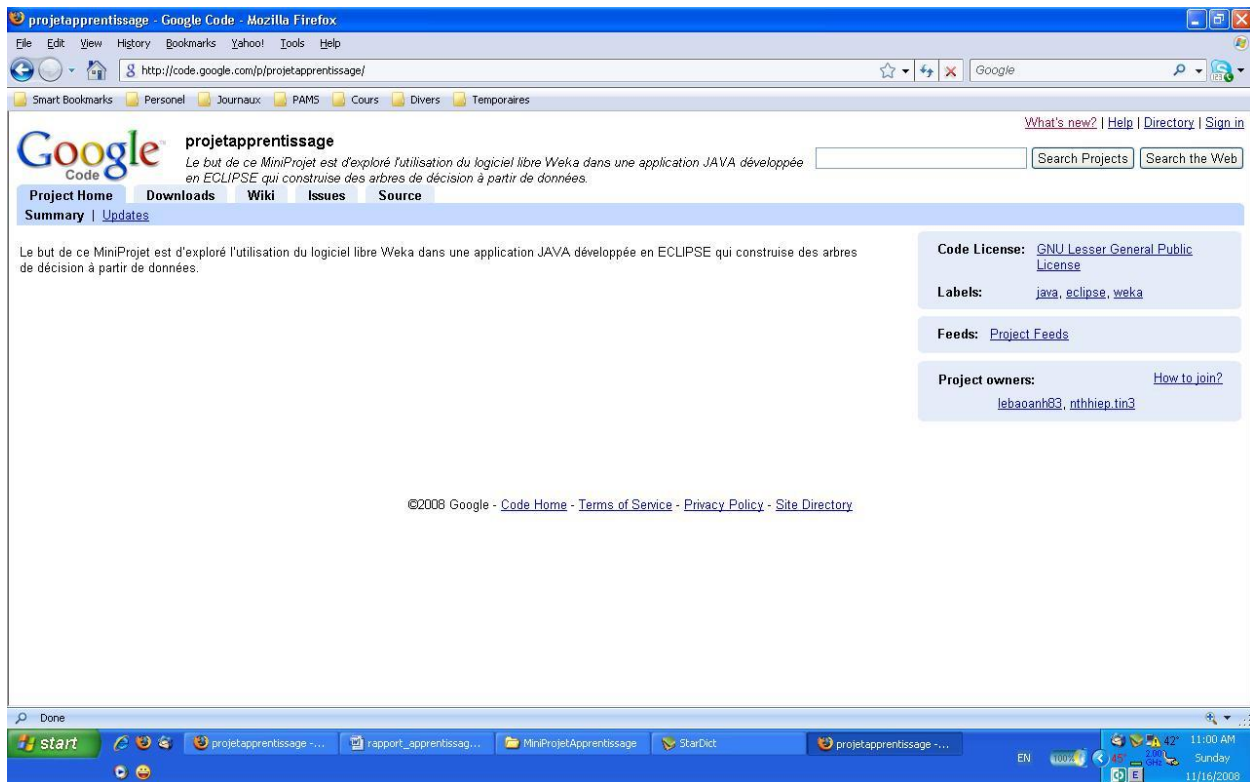
INTRODUCTION .....	3
I. LE PROJET DEVELOPPE .....	4
III. DES EXPERIMENTATIONS.....	7
III.1. La base de données de “Iris” du “Repository UCI” .....	7
La description de la base de données choisie .....	7
Cas de test.....	7
Explications des Résultats .....	7
III.2. La base de données de “zoo” .....	13
La description de la base de données choisie .....	13
II. CONCLUSION .....	16
III. ANNEXE.....	17
V.1. Guide d’installation .....	17
V.2. Guide d’utilisation .....	18

## INTRODUCTION

Dans l'espoir de mettre en oeuvre ce projet de la meilleure façon. Nous avons décidé de profiter les outils informatiques abordés comme ci-dessous :

Nous déposons nos codes sources dans un dépôt de Google sur l'internet :

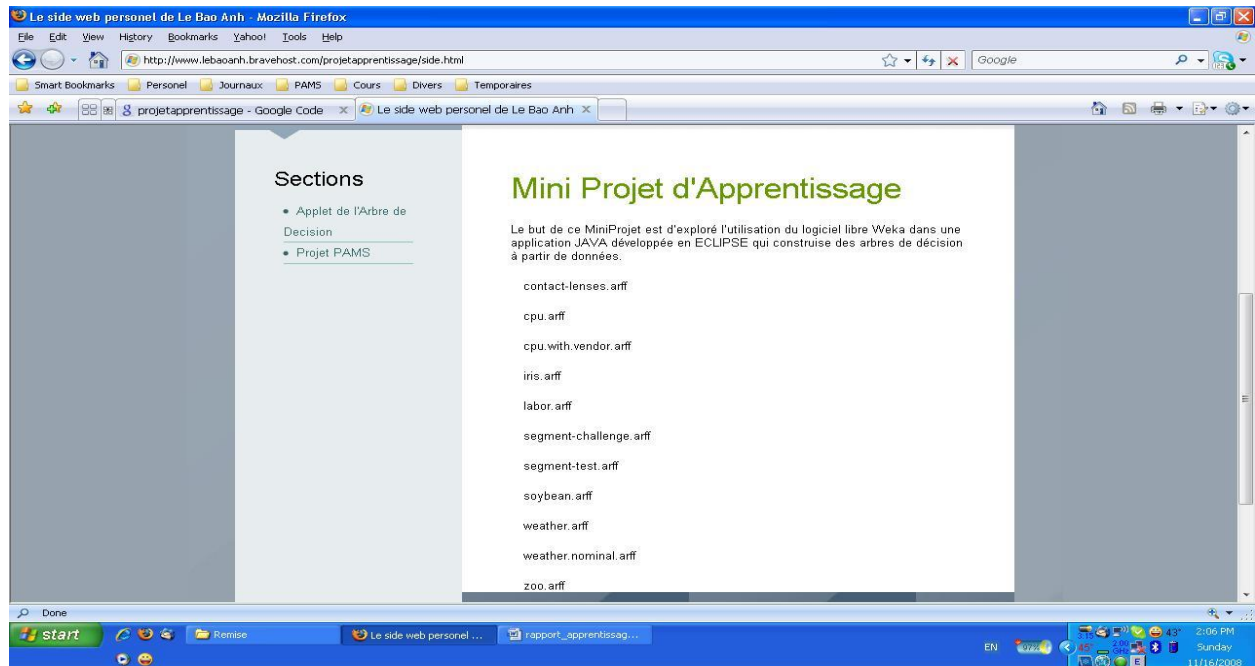
<http://code.google.com/p/projetapprentissage/>



Dans ce site, vous pouvez voir le progrès du développement du projet. Toutes les revisions sont stockées dans la rubrique "Source". La version finale du projet, le rapport et ainsi que la bibliothèque de Weka (version 3-5-8) qui se trouve dans la rubrique "Download".

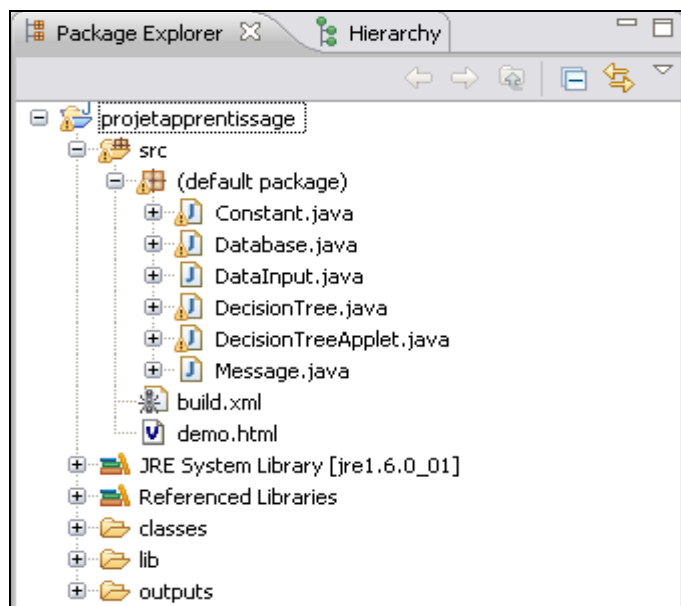
Pour but d'éviter les problèmes du réseau, nous récupérons et puis stockons toutes les bases de données utilisées nous-même sur un hébergement propre selon l'adresse :

<http://www.lebaoanh.bravehost.com/projetapprentissage/side.html>



Au fil du temps de réalisation du projet, nous avons beaucoup consulté de différents documents ainsi que les codes sources ouverts notamment sur l'internet.

## I. LE PROJET DEVELOPPE



Dans notre projet, nous avons implémenté 6 classes correspondant aux 6 fichiers java mais 3 classes principales du projet sont : Database, DecisionTree et DecisionTreeApplet.

**Database** : contient la base de donnée à exécuter et fournit quelques fonctions de traitement sur la base de données comme :

Télécharger une base de données sur Internet et l'enregistrer dans la base interne

Prendre les attributs nominaux pour choisir l'attribut à classer

Prendre tous les attributs de la base de données comme les candidats qu'on peut utiliser pour la construction de l'arbre de décision

Filtrer les attributs

Exemple : Dans la base de données originale, il y a 10 attributs en total. Mais l'utilisateur n'a choisi que 5 attributs comme des attributs d'entrées pour l'apprentissage et pour la construction de l'arbre. Donc, nous devons filtrer pour prendre seulement ces 5 attributs.

Calculer le « gain d'information » des attributs choisis à utiliser dans la construction de l'arbre de décision

Séparer une partie pour l'apprentissage et une pour le test de la base de données.

Partie d'apprentissage : il y a 3 manières d'extraire :

Extraire aléatoirement un certain nombre des instances de la base

Extraire un certain nombre de premières instances

Extraire un certain nombre de dernières instances

Partie de test : il y a 5 façons de créer l'échantillon de test :

L'échantillon de test  $\equiv$  ceux d'apprentissage

L'échantillon de test est le complément de ceux d'apprentissage

Extraire aléatoirement un certain nombre des instances de la base

Extraire un certain nombre de premières instances

Extraire un certain nombre de dernières instances

**DecisionTree** : fournit des fonctions de manipulation avec l'arbre de décision ci-dessous :

Construire un arbre de décision à partir de l'échantillon d'apprentissage contenant des attributs choisis par l'utilisateur

Calculer le taux d'erreur (cross-validation) de l'arbre de décision construit sur la partie d'apprentissage

Tester l'arbre de décision sur un échantillon de test. La partie de test est déterminé par l'utilisateur

Prédire ou bien classer les instances de la base de données par l'évaluation de l'arbre de décision

Par exemple : dans la base de données, l'instance X appartient à la classe A mais elle est classé à la classe B selon l'évaluation de l'arbre de décision.

Visualiser graphiquement l'arbre de décision : nous prenons la description de l'arbre de décision en Weka pour l'afficher graphiquement sur l'écran

**DecisionTreeApplet** : c'est une interface graphique d'interaction entre le programme et l'utilisateur. Vous pouvez consulter les tâches de cette interface dans la guide d'utilisation pour avoir plus d'informations en détail. Cette classe fournit aux utilisateurs des fonctions suivantes :

Télécharger une base de données

Filtrer les attributs nominaux et afficher tous les attributs de la base de données

Établir le type de l'arbre de décision : J48, ID3, BFTree,...

Diviser la base de données en partie d'apprentissage et en partie de test

Construire l'arbre de décision

Calculer le taux d'erreur (cross-validation) de l'arbre sur l'échantillon d'apprentissage

Tester l'arbre sur l'échantillon de test

visualiser le gain-info des attributs utilisés

visualiser graphiquement l'arbre de décision

**DataInput** : enregistre quelques sources des bases de données au format arff. Ces sources sont affichées dans le champ d'adresse pour que l'utilisateur choisisse une source à travailler.

**Constant** : enregistre les codes des différents arbres de décision

**Message** : c'est un dialogue pour informer l'utilisateur de quelques certaines informations, par exemple : l'échec du téléchargement de la base de données



### III. DES EXPERIMENTATIONS

#### *III.1. La base de données de “Iris” du “Repository UCI”*

##### La description de la base de données choisie

Nous testons encore notre programme avec une base de données des fleurs, téléchargée de la source <http://www.hakank.org/weka/iris.arff>. Cette base de données contient 5 attributs ci-dessous :

sepalength : la longueur du calice. C'est un attribut numérique, [4.3 – 7.9 cm]

sepalwidth : la largeur du calice. Cet attribut est numérique, [4.3 – 7.9 cm]

petallength : la longueur du pétale. C'est un attribut numérique, [4.3 – 7.9 cm]

petalwidth : la largeur du pétale. C'est un attribut numérique, [4.3 – 7.9 cm]

classe : l'espèce de la fleur. Il y a 3 classes : Setosa, Versicolour et Virginica. attribut à classer.

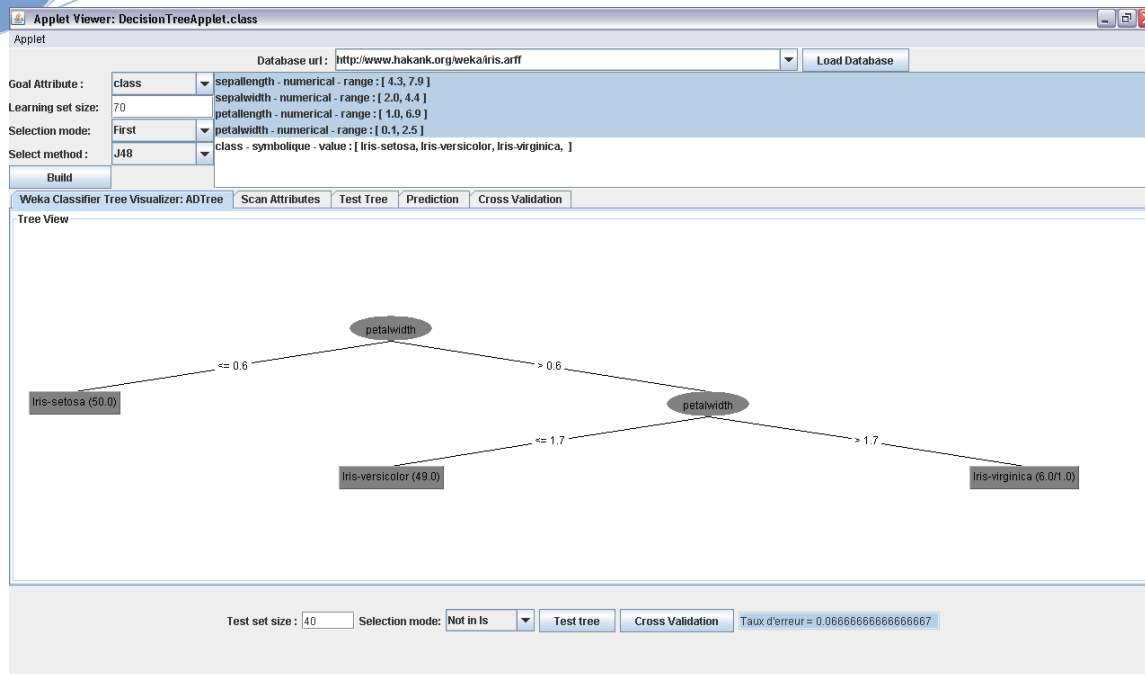
Cette base contient en total 150 instances.

##### Cas de test

- taille de la partie d'apprentissage : premier 70 % de la base
- taille de la partie de teste : 30 % le reste
- type de l'arbre de décision : J48
- attributs de base : 4 attributs numériques : sepalength , sepalwidth, petallength et petalwidth
- attribut à classer : classe

##### Explications des Résultats

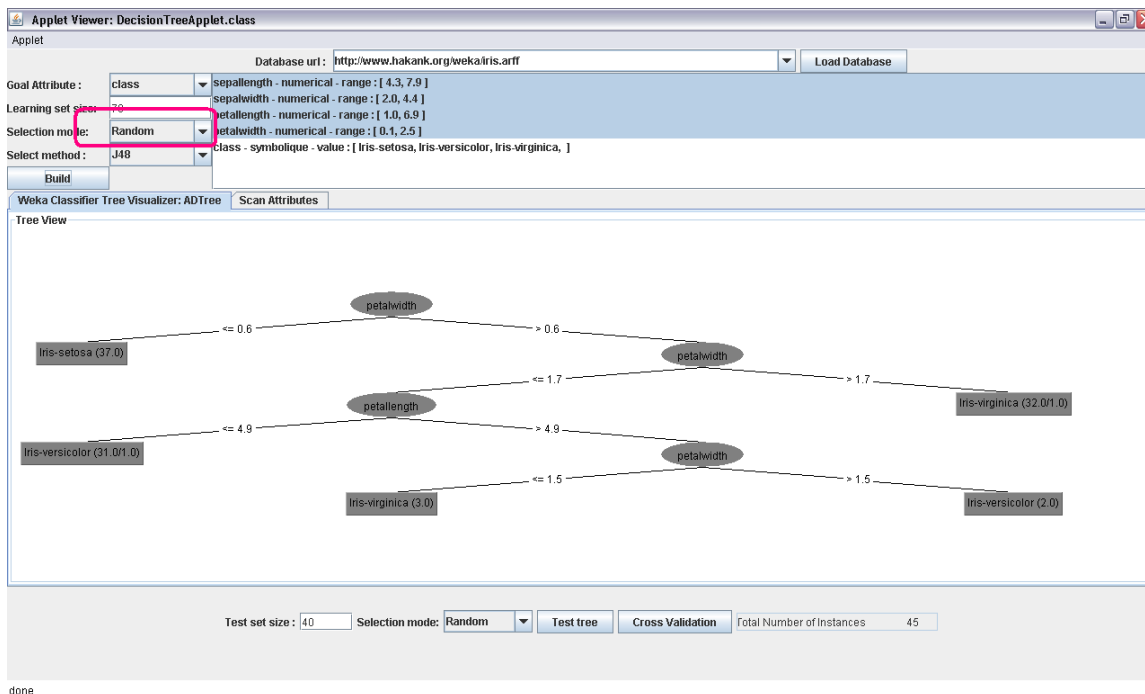
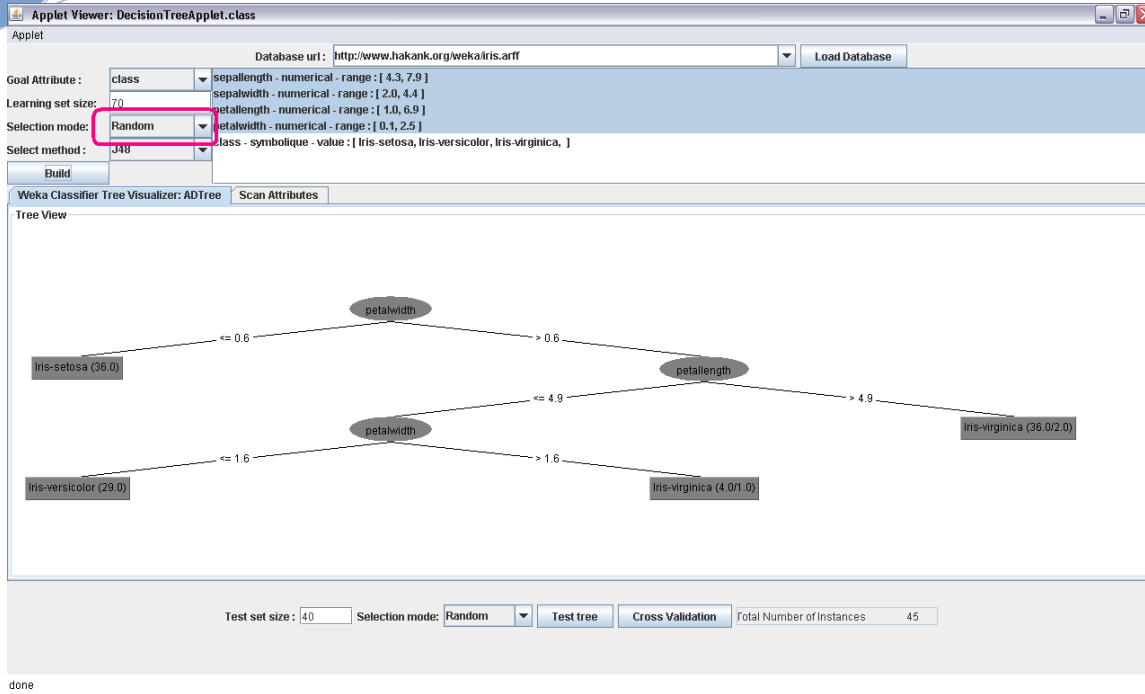
###### **a. Arbre de décision**



Donc, à partir de la partie d'apprentissage, l'arbre de décision peut classer les données en se basant sur un seul attribut « petalwidth »

- si  $\text{petalwidth} \leq 0.6$  → la fleur considérée est Iris-setosa
- si  $0.6 < \text{petalwidth} \leq 1.7$  → la fleur considérée est Iris-versicolor
- si  $1.7 \leq \text{petalwidth}$  → la fleur considérée est Iris-virginica

Ici, la partie d'apprentissage contient 70% des premières instances de la base. Donc, le résultat (arbre construit) est le même pour toutes les différentes fois d'exécution. Mais si nous changeons la manière d'extraire la partie d'apprentissage en mode aléatoire, les résultats des différentes fois d'exécutions sont aussi différents parce que la partie d'apprentissage est changée tout le temps.



## b. Gain d'information des attributs

- Gain d'information d'un attribut présente la diminution de pureté de cet attribut après la scission. L'objectif de la classification est de grouper les instances dans les groupes purs contenant un seul type (classe) de données. Les feuilles de l'arbre de

décision sont parfaitement pures. Donc, nous avons besoin de diminuer le plus possible la pureté. Par conséquent, les attributs plus purs sont choisis avant les autres dans la construction de l'arbre ou dans le processus du branchement de l'arbre.

Scan Attributes
sepalwidth - gain info : 0.5243120130108034
sepalwidth - gain info : 0.36114188784993256
petalwidth - gain info : 0.9464939241607738
petalwidth - gain info : 0.9464939241607738

Ici, le gain d'information de l'attribut petalwidth et petalwidth est pareil. Donc, un de ces 2 attributs sont choisis avant tout. Dans cet exemple, c'est « petalwidth » qui est choisis avant tout.

### c. Cross-validation

Weka Classifier Tree Visualizer: ADTree	Scan Attributes	Test Tree	Prediction	Cross Validation
Taux d'erreur = 0.0666666666666667				
Correctly Classified Instances	98	93.3333 %		
Incorrectly Classified Instances	7	6.6667 %		
Kappa statistic	0.8784			
Mean absolute error	0.0459			
Root mean squared error	0.2074			
Relative absolute error	12.5663 %			
Root relative squared error	48.6576 %			
Total Number of Instances	105			

Le taux d'erreur moyenne de 10 fois de l'arbre de décision sur la partie d'apprentissage est  $0.066667 = 6.67\%$ .

Parmi 105 instances de la partie de test, 98 sont bien classé tandis que 7 sont mal classés

### d. Test

Weka Classifier Tree Visualizer: ADTree		Scan Attributes	Test Tree
Taux d'erreur = 0.1111111111111111			
Correctly Classified Instances	40	88.8889 %	
Incorrectly Classified Instances	5	11.1111 %	
Kappa statistic	0		
Mean absolute error	0.1728		
Root mean squared error	0.3009		
Relative absolute error	622.2222 %		
Root relative squared error	1021.2556 %		
Total Number of Instances	45		

Parmi 45 instances de la partie de test, 41 sont bien classé tandis que 4 sont mal classés. Le taux d'erreur est  $0.1111 = 11.11\%$ , plus grand que ceux sur la partie d'apprentissage.

De même manière pour la construction de l'arbre de décision, si l'extraction de la partie d'apprentissage est aléatoire, les taux d'erreur des différentes fois d'exécution sont différents. Il faut reconstruire l'arbre de décision avant chaque test pour voir ce fait.

Goal Attribute : class

Learning set size: 70

Selection mode: **First**

Select method: J48

Build

Weka Classifier Tree Visualizer: ADTree		Scan Attributes	Test Tree	Prediction
Taux d'erreur = 0.0266666666666667				
Correctly Classified Instances	<b>73</b>	97.3333 %		
Incorrectly Classified Instances	<b>2</b>	2.6667 %		
Kappa statistic	0.9598			
Mean absolute error	0.0262			
Root mean squared error	0.1203			
Relative absolute error	5.9232 %			
Root relative squared error	25.5643 %			
Total Number of Instances	75			

Test set size : 50 Selection mode: **Random** Test tree Cross Validation

Applet Viewer: DecisionTreeApplet.class

Applet

Database url:

Goal Attribute:

Learning set size:

Selection mode:

Select method:

Weka Classifier Tree Visualizer: ADTree

Taux d'erreur = 0.013333333333333334

Correctly Classified Instances	74	98.6667 %
Incorrectly Classified Instances	1	1.3333 %
Kappa statistic	0.9799	
Mean absolute error	0.0233	
Root mean squared error	0.09	
Relative absolute error	5.2603 %	
Root relative squared error	19.1175 %	
Total Number of Instances	75	

Test set size:  Selection mode:

### e. Prédiction et évaluation de l'arbre de décision

The screenshot shows the Weka Classifier interface with the following configuration:

- Database url: <http://www.hakank.org/weka/iris.arff>
- Goal Attribute: class
- Learning set size: 70
- Selection mode: First
- Select method: J48

The list of instances is as follows:

```

7.2,3.2,6.1,8,Iris-virginica -> [Iris-virginica]
6.2,2.8,4.8,1.8,Iris-virginica -> [Iris-virginica]
6.1,3.4,9.1,8,Iris-virginica -> [Iris-virginica]
6.4,2.8,5.6,2.1,Iris-virginica -> [Iris-virginica]
7.2,3.5,8.1,6,Iris-versicolor -> [Iris-versicolor]
7.4,2.8,6.1,1.9,Iris-virginica -> [Iris-virginica]
7.9,3.8,6.4,2,Iris-virginica -> [Iris-virginica]
6.4,2.8,5.6,2.2,Iris-virginica -> [Iris-virginica]
6.3,2.8,5.1,1.5,Iris-versicolor -> [Iris-versicolor]
6.1,2.6,5.6,1.4,Iris-versicolor -> [Iris-versicolor]
7.7,3.6,1.2,3,Iris-virginica -> [Iris-virginica]
6.3,3.4,5.6,2.4,Iris-virginica -> [Iris-virginica]
6.4,3.1,5.5,1.8,Iris-virginica -> [Iris-virginica]
6.3,4.8,1.8,Iris-virginica -> [Iris-virginica]
6.9,3.1,5.4,2.1,Iris-virginica -> [Iris-virginica]
6.7,3.1,5.6,2.4,Iris-virginica -> [Iris-virginica]
6.9,3.1,5.1,2.3,Iris-virginica -> [Iris-virginica]
5.8,2.7,5.1,1.9,Iris-virginica -> [Iris-virginica]
6.8,3.2,5.9,2.3,Iris-virginica -> [Iris-virginica]
6.7,3.3,5.7,2.5,Iris-virginica -> [Iris-virginica]
6.7,3.5,2.2,3,Iris-virginica -> [Iris-virginica]
6.3,2.5,1.9,Iris-virginica -> [Iris-virginica]
6.5,3.5,2.2,Iris-virginica -> [Iris-virginica]

```

At the bottom, the Test set size is 40 and Selection mode is Not in Is.

Dans la tab prediction apparaissant après avoir cliqué sur le bouton « Test tree », nous présentons l'évaluation de la classification de l'arbre de décision. La plupart d'instances sont bien classé. Mais dans certains cas, l'évaluation de l'arbre et la valeur actuelle dans la base sont différents

### III.2. La base de données de "zoo"

#### La description de la base de données choisie

Une simple base de données contenant 17 Booléen-valeur des attributs. L'attribut «type» est l'attribut de classe.

Nombre des instances : 101

Nombre des attributes : 18 (le nom de l'animal, 15 attributs booléen, 2 numériques).

Attribute Information: (name of attribute and type of value domain)

1. animal : Unique for each instance
2. hair : Booléen
3. feathers : Booléen
4. eggs : Booléen
5. milk : Booléen

- 6. airborne : Booléen
- 7. aquatic : Booléen
- 8. predator : Booléen
- 9. toothed : Booléen
- 10. backbone: Booléen
- 11. breathes : Booléen
- 12. venomous : Booléen
- 13. fins : Booléen
- 14. legs : Numeric (set of values: {0,2,4,5,6,8})
- 15. tail : Booléen
- 16. domestic : Booléen
- 17. catsize Booléen
- 18. type Numeric (integer values in range [1,7])

Ici, nous choisissons les 17èmes attributes comme les attributes de candidat, le dernier attribute “type” sera “goal Attribute”.

*Voilà les résultats :*

The screenshot shows the Weka Classifier Tree Visualizer interface. The top menu bar includes 'Weka Classifier Tree Visualizer: ADTree', 'Scan Attributes', 'Test Tree', 'Prediction', and 'Cross Validation'. The configuration area shows the following settings:

- Database url: <http://www.lebaonh.bravehost.com/projetapprentissage/data/zoo.arff>
- Goal Attribute: type
- Learning set size: 60
- Selection mode: Random
- Select method: J48

The performance metrics table is as follows:

Metric	Value	Percentage
Taux d'erreur	0.05	
Correctly Classified Instances	57	95 %
Incorrectly Classified Instances	3	5 %
Kappa statistic	0.9335	
Mean absolute error	0.0185	
Root mean squared error	0.1217	
Relative absolute error	8.388 %	
Root relative squared error	36.8538 %	
Total Number of Instances	60	

At the bottom of the interface, there are buttons for 'Test set size: 40', 'Selection mode: First', 'Test tree', and 'Cross Validation'. The Windows taskbar at the bottom shows the system tray with the date and time: Sunday, 11/16/2008, 8:51 PM.

Premièrement, nous obtenons un arbre de décision de la méthode J48 dans notre cas.

Weka Classifier Tree Visualizer: ADTree

Scan Attributes Test Tree Prediction Cross Validation

```

animal - gain info : 0.38652670559531127
hair - gain info : 0.7461876113297465
feathers - gain info : 1.0
eggs - gain info : 0.7771712115043878
milk - gain info : 1.0
airborne - gain info : 0.6423938137980608
aquatic - gain info : 0.30178866501763124
predator - gain info : 0.11542250647719716
toothed - gain info : 0.8987447145391185
backbone - gain info : 1.0
breathes - gain info : 0.7790536461742945
venomous - gain info : 0.4947049428901417
fins - gain info : 0.6836592422489939
legs - gain info : 0.6908382338243251
tail - gain info : 0.6963396282851518
domestic - gain info : 0.13378542649158853
catsize - gain info : 0.49045826278903826

```

Test set size : 40 Selection mode: First Test tree Cross Validation

Le deuxième résultat très important est le “gain infos” de chaque “candidat attribue”.

Weka Classifier Tree Visualizer: ADTree

Scan Attributes Test Tree Prediction Cross Validation

Taux d'erreur = 0.025

Correctly Classified Instances	39	97.5 %
Incorrectly Classified Instances	1	2.5 %
Kappa statistic	0.968	
Mean absolute error	0.0111	
Root mean squared error	0.0786	
Relative absolute error	4.9068 %	
Root relative squared error	23.4925 %	
Total Number of Instances	40	

Test set size : 40 Selection mode: First Test tree Cross Validation

Le taux d'erreur de l'arbre de décision sur la partie de teste et les autres résultats analysés.



### III. ANNEXE

#### V.1. Guide d'installation

Au début, notre but est de réaliser un applet intégré avec un site web sur l'internet mais nous avons rencontré quelques problèmes techniques. Ici, nous présentons la façon de l'installation de l'applet dans Eclipse.

- Tout d'abord, on crée un projet Java.
- En suite, après avoir supprimé le répertoire "src" on peut :

+ Télécharger la version finale à partir de

<http://code.google.com/p/projetapprentissage/downloads/list>

+ Faire check-out à partir du code source du server SVN :

```
svn checkout http://projetapprentissage.googlecode.com/svn/trunk/src
projetapprentissage-read-only
```

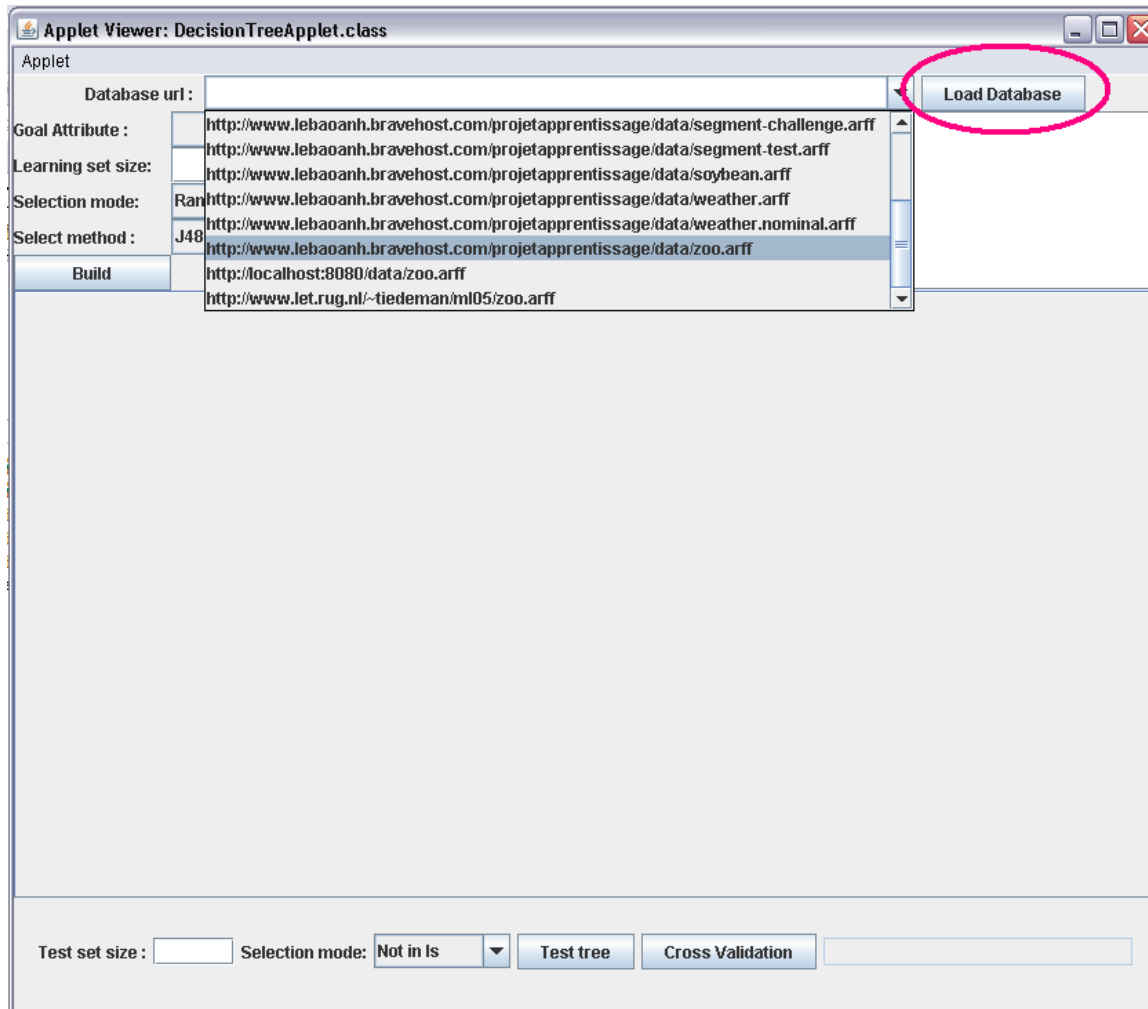
- Créer un répertoire de bibliothèque qui s'appelle "lib" de meme niveau avec "src".

Télécharger le fichier "external library Weka – weka-lib.jar" dans le répertoire "lib" :

<http://code.google.com/p/projetapprentissage/downloads/list>

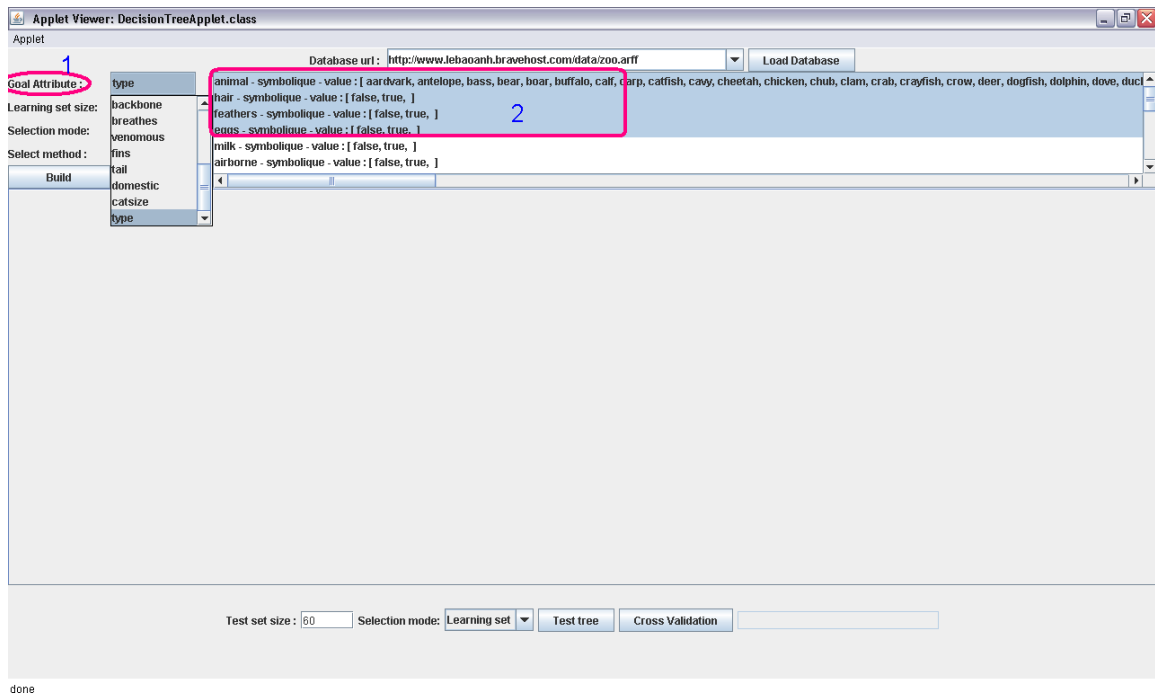
- Dans Eclipse choisir le projet -> souris à droite -> Properties -> Java Build Path -> Libraries -> Add External Jars... -> choisir : lib/weka-lib.jar téléchargé tout à l'heure -> ok.
- Maintenant on peut lancer DecisionTreeApplet.java comme un applet à partir du code source.

## V.2. Guide d'utilisation

**Étape 1 : Télécharger une base de données**

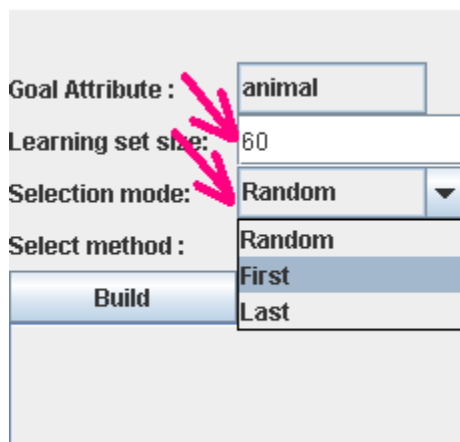
Choisissez une des adresses des bases de données existantes dans le combo-box ou tapez une nouvelle adresse. Ensuite, vous appuyez sur le bouton « load database » pour télécharger les données de la base.

**Étape 2 : Sélectionner des attributs à étudier et un attribut de classe**



La liste droite « 2 » contient tous les attributs de la base de données tandis que la liste gauche « 1 » ne contient que les attributs nominaux, pas numériques. Vous pouvez sélectionner quelques ou tous les attributs à apprendre comme des attributs d'entrée et un seul attribut nominal à classer.

### Étape 3 : Séparer une partie à apprendre de la base de données



Nous ne prenons qu'une partie de la base de données pour l'apprentissage. On doit déterminer la taille (%) de cette partie. Dans cet exemple, l'échantillon d'apprentissage

contient 60% des instances de la base de données. Il y a 3 façons de choisir la partie d'apprentissage:

Random : on prend aléatoirement les instances de la base de données (60% par défaut).

First : on prend un certain pourcentage de premières instances de la base (60% par défaut)

Last : on prend un certain pourcentage de dernières instances de la base (60% par défaut).

#### Étape 4 : Construire l'arbre de décision

Cliquez sur le bouton « build » pour construire l'arbre de décision à partir de l'échantillon d'apprentissage. Vous pouvez voir l'image de cet arbre dans le tab « Weka Classifier Tree Visualiser ». Pour ajuster la taille graphique de l'arbre, vous appuyez sur le bouton droit du souris et choisissez une des options fournies.

#### Étape 5 : Voir les résultats et les évaluer

Vous pouvez voir tous les attributs utilisés pour la construction de l'arbre de décision et ainsi que leur « *gain d'information* » dans le tab « Scan Attributes »

Pour évaluer l'exactitude du résultat obtenu, ou bien le taux d'erreur de l'arbre de décision sur la partie d'apprentissage elle-même, vous cliquez sur le bouton «Cross Validation». Le champ de texte à côté présente le taux d'erreur.

The screenshot shows the Weka Classifier Tree Visualizer interface. The window title is "Applet Viewer: DecisionTreeApplet.class". The interface includes a "Database url" field with the value "http://www.lebaoanh.bravehost.com/projetapprentissage/data/zoo.arff" and a "Load Database" button. The "Goal Attribute" is set to "type". The "Learning set size" is 60. The "Selection mode" is "Random" and the "Select method" is "J48". A "Build" button is visible. Below these settings are three tabs: "Weka Classifier Tree Visualizer: ADTree", "Scan Attributes", and "Cross Validation", with the latter being selected and circled in red. The main area displays the following statistics:

Taux d'erreur = 0.25		
Correctly Classified Instances	45	75 %
Incorrectly Classified Instances	15	25 %
Kappa statistic	0.676	
Mean absolute error	0.0946	
Root mean squared error	0.2299	
Relative absolute error	41.4201 %	
Root relative squared error	68.1747 %	
Total Number of Instances	60	

At the bottom, there is a "Test set size" field with the value 40, a "Selection mode" dropdown set to "Learning set", a "Test tree" button, and a "Cross Validation" button, which is also circled in red. The status bar at the bottom left shows "done".

## Étape 6 : Tester les résultats

Pour tester l'arbre de décision construit sur un échantillon de test, il faut créer un échantillon de test par une de 5 manières suivantes :

« Not is in » : la partie de test est le complément de l'échantillon d'apprentissage

« Learning set » : la partie de test = celle d'apprentissage

« Random » : on prend aléatoirement un certain nombre des instances de la base de données pour créer l'échantillon de test. Vous tapez la taille voulue en pourcentage (%) de la partie de test dans le champ « Test set size », par exemple 40 (%)

« First » : de la même manière que pour l'option « random » mais on prend un certain nombre des premières instances de la base de données pour tester.

« Last » : on prend un certain nombre de premières instances de la base de données pour tester.

Vous cliquez sur le bouton « Selection mode » pour choisir une des 5 manières ci-dessus.

Maintenant, appuyez sur le bouton « Test tree » pour recevoir l'évaluation présenté dans le tab Test

Applet Viewer: DecisionTreeApplet.class

Applet

Database url :

Goal Attribute :  animal - symbolique - value : [ aardvark, antelope, bass, bear, boar, buffalo, ]

Learning set size:  hair - symbolique - value : [ false, true, ]

Selection mode:  eggs - symbolique - value : [ false, true, ]

Select method :  milk - symbolique - value : [ false, true, ]

airborne - symbolique - value : [ false, true, ]

Build

Weka Classifier Tree Visualizer: ADTree Scan Attributes **Test Tree** Prediction

Taux d'erreur = 0.2833333333333333

Correctly Classified Instances	43	71.6667 %
Incorrectly Classified Instances	17	28.3333 %
Kappa statistic	0.645	
Mean absolute error	0.0939	
Root mean squared error	0.2167	
Relative absolute error	40.3819 %	
Root relative squared error	63.7237 %	
Total Number of Instances	60	

Test set size :  Selection mode:  **Test tree** Cross Validation

Applet started.

Applet Viewer: DecisionTreeApplet.class

Applet

Database url : <http://www.lebaoanh.bravehost.com/projetapprentissage/data/zoo.arff>

Goal Attribute : type

Learning set size: 60

Selection mode: Random

Select method: J48

Build

Weka Classifier Tree Visualizer: ADTree Scan Attributes Test Tree Prediction

```

seahorse,false,false,true,fish -> [fish]
kiwi,false,true,true,bird -> [bird]
stingray,false,false,true,fish -> [fish]
bass,false,false,true,fish -> [fish]
pitviper,false,false,true,reptile -> [fish]
gorilla,true,false,false,mammal -> [mammal]
termite,false,false,true,insect -> [fish]
seasnake,false,false,false,reptile -> [mammal]
chicken,false,true,true,bird -> [bird]
wasp,true,false,true,insect -> [insect]
sealion,true,false,false,mammal -> [mammal]
polecat,true,false,false,mammal -> [mammal]
wallaby,true,false,false,mammal -> [mammal]
=====
TEST SET
animal hair feathers eggs type
=====
frog,false,false,true,amphibian -> [fish]
starfish,false,false,true,invertebrate -> [fish]
dove,false,true,true,bird -> [bird]
tuatara,false,false,true,reptile -> [fish]
octopus,false,false,true,invertebrate -> [fish]
nike,false,false,true,fish -> [fish]

```

Test set size: 40 Selection mode: Learning set Test tree Cross Validation

Applet started.

Par ailleurs, vous pouvez voir l'évaluation de l'arbre de décision sur l'attribut à classer, ou bien la prédiction de l'arbre, dans la table « Prediction ». Nous présentons l'évaluation de l'arbre pour la partie d'apprentissage et aussi pour celle de test.