An Exploration of Open-Source Data Mining Tools

Angel Trifonov, angelt@wpi.edu
Worcester Polytechnic Institute
CS542 – Database Management Systems, S14
Course Project Final Report, April 21st 2014

Abstract

In the previous report, we briefly examined and overviewed three data mining tools: Orange – a component-based data mining and machine learning software suite, implemented in C++/Python, Tanagra – a machine learning suite, which supports most data mining techniques, and Weka – a popular suite of machine learning software written in Java. This report will compare these data mining tools in hopes to find the best one for a particular task. We will also see how they perform on several tasks: building a ROC curve, building a decision tree, comparison of Supervised Methods, and computing association rules.

1. Introduction

In this report, we perform several data-mining tasks on each tool, and compare the results of each program. The way we'll evaluate the three tools is the following: for a particular task, each tool uses the same dataset. After completion of all three tests, we'll offer some thoughts on which tool performed best and which one is most appropriate to use for this particular job. Including screenshots for every single step of each process will be too much, so we will rely mostly on a step-by-step process to explain how to accomplish the tasks. Screenshots will only be included for the most important parts. For clarity, different components/options will be followed by the section, in brackets, where they can be found.

There were three main goals when choosing the tools to compare. First, the tools needed to have an easy enough to use user interface, so not much time would be spend trying to learn it. Second, they all needed to have a good amount of documentation and information written about them in order to provide background and in case there was an issue with the program. Finally, and most importantly, they all needed to be able to perform the same four tasks, using roughly the same methods and/or algorithms. To that end, Weka, Tanagra and Orange covered all three requirements. After an initial installation to get a feel for the programs and interfaces, they were chosen for the study.

A few things were taken into consideration when choosing the datasets to work on. First, they needed to be of sufficient size, as to get more accurate results. Second, they needed to be popular. What this means is, they needed to come from a reliable source. This way, we can be certain the data is reliable. Finally, some previous experimentation should have been performed on them. This would guarantee the datasets were good for a data-mining task.

The problems examined are: (1) computing a ROC curve from a logistic regression, (2) how to build and estimate the error rate of a decision tree using a cross-validation process, (3) how to build several prediction models with the same training set and estimate their error rate using a predefined test set, and finally, (4) how to compute association rules from various implementations of the APRIORI algorithm. These tasks were chosen because they offer a good mix of different data mining techniques and algorithms, graph creation, and data manipulation.

This report is organized as follows: sections 2 examines building a ROC curve, sections 3 examines building a decision tree, section 4 compares supervised methods, and section 5 details computing association rules. Section 6 offers a brief conclusion.

2. Building a ROC Curve

2.1 Overview

Regardless of the software we used, we have to perform the following steps when we want build a ROC curve:

1) Import the dataset in the desired tool

2) Compute descriptive statistics

3) Select target and input attributes

4) Select the "positive" value of the target attribute

5) Split the dataset into learning (e.g. 70%) and test set (30%)

6) Choose the learning algorithm. We have to be careful, because the tools can have a different implementation and present different results

7) Build the prediction model on the learning set and visualize the results

8) Build the ROC curve on the test set

We use the dataset from the Komarek's website which implements the LR-TRIRLS logistic regression library (http://komarix.org/ac/lr). The DS1-10 dataset contains 26733 examples, 10 continuous descriptors; the frequency of the positive value of the target attribute is 5%. We use ARFF file format for Weka and Tanagra, TXT for Orange (Tanagra can also handle TXT file format).

2.2 Weka

When we execute Weka, a dialog box enables to choose the execution mode. The EXPLORER is simpler in terms of UI, but KNOWLEDGE FLOW offers more control. We select the KNOWLEDGE FLOW option. The ARFF LOADER component (DATASOURCES palette) enables us to load a dataset. We add it in the workspace, then select the file with the by right-clicking on the ARFF LOADER and selecting CONFIGURE from the menu.

The ATTRIBUT SUMMARIZER component (VISUALIZATION) shows the data distribution; we can use it to quickly detect abnormalities in the dataset. We add this component and we connect ARFF LOADER to this new component (DATASET connection). The treatment will be executed when we select the START LOADING menu of the ARFF LOADER component. To see the results, we select the SHOW SUMMARIES option of ATTRIBUTES SUMMARIZER.

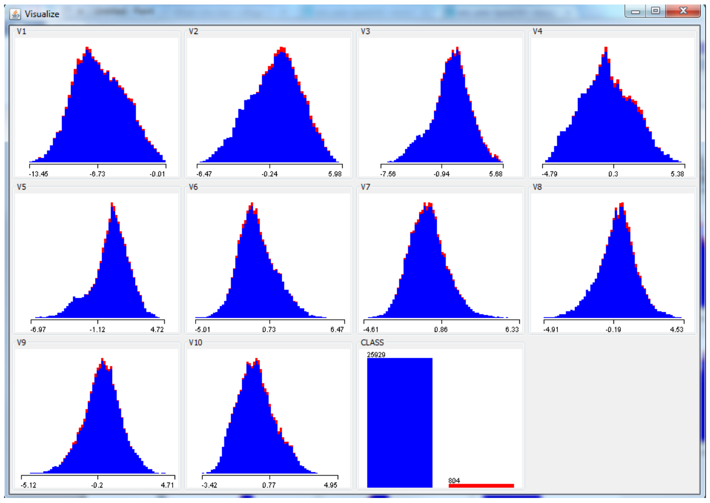
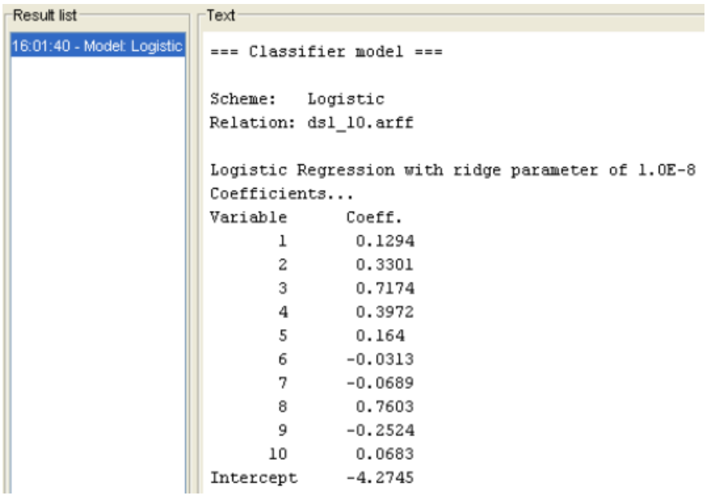


Figure 1 – Visualization of the Dataset

We see that there are no abnormalities on the descriptors; we see also that we have imbalanced dataset (804 "positive" vs. 25929 "negative"). The last column is the default class attribute for Weka, but we can also explicitly select the column of the class attribute. In this case, we use the CLASS ASSIGNER component (EVALUATION palette). We add this component in our diagram; we connect ARFF LOADER to this new component (DATASET connection). We select the CONFIGURE menu in order to select the right class attribute.

In the following step, we must specify the "positive" value of the class attribute. We use the CLASSVALUE PICKER (EVALUATION palette) component and select the "TRUE" value in the parameter dialog box. We want to build our prediction model on the 70% of the whole dataset, and compute the ROC curve on the remaining. So, we set the TRAINTEST SPLIT MAKER (EVALUATION) in the diagram and configure its parameters.

The logistic regression component is in the CLASSIFIERS palette. We set it in the diagram, we connect twice the TRAIN TEST SPLIT MAKER to this new component: twice because we must use together the training and the test set which are produced by the same component. To see the results of the regression, we connect the LOGISTIC component to TEXT VIEWER (VISUALIZATION palette) that we set in the diagram. We execute again the diagram (START LOADING of ARFF LOADER component). The SHOW RESULTS of TEXT VIEWER opens a new window with the results of the learning process. We obtain the regression coefficients.



```
=== Classifier model ===

Scheme:   Logistic
Relation: ds1_10.arff

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
Variable      Coeff.
       1       0.1294
       2       0.3301
       3       0.7174
       4       0.3972
       5       0.164
       6      -0.0313
       7      -0.0689
       8       0.7603
       9      -0.2524
      10       0.0683
Intercept     -4.2745
```

Figure 2 – Regression Coefficients

In order to evaluate the learning process, we add the CLASSIFIER PERFORMANCE EVALUATOR component (EVALUATION). We connect the BATCH CLASSIFIER output of LOGISTIC to this new component. We set MODEL PERFORMANCE CHART (VISUALIZATION) in the diagram; we use the THRESOLD DATA output of the CLASSIFIER PERFORMANCE EVALUATOR when we connect the two components. We run again the diagram with the START LOADING menu of ARFF LOADER. We select the SHOW PLOT menu of the last component (MODEL PERFORMANCE CHART). We obtain the ROC curve.

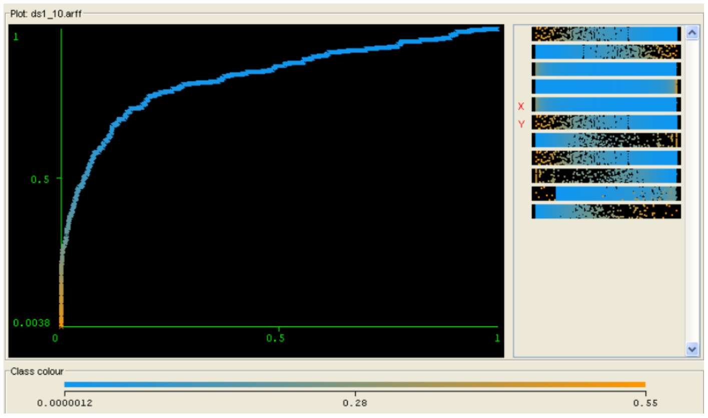
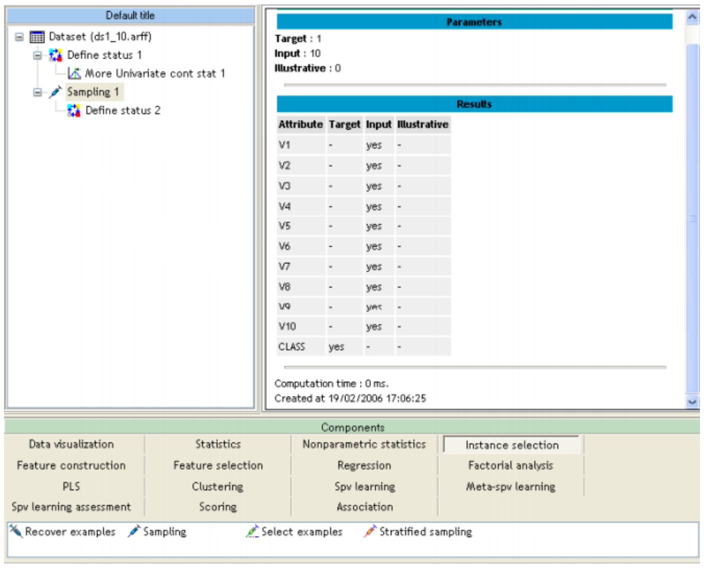


Figure 3 – The Weka ROC Curve

2.3 Tanagra

There are three parts in the main window of Tanagra. At the left, we have the stream diagram where we define our treatments; at the bottom, the data mining tools in various palettes; at the center, the window for displaying the results. To load a dataset, we select the FILE → NEW menu in order to create a new diagram. We add a DEFINE STATUS in the diagram; we set as INPUT all continuous attributes. We set the MORE UNIVARIATE CONTSTAT (STATISTICS) in the diagram, we execute the component (VIEW menu), and the detailed results are displayed.

We select again the root of the diagram and insert the SAMPLING component (INSTANCE SELECTION); we set the right parameters (PARAMETERS menu). With a new DEFINE STATUS component, we set CLASS as TARGET attribute, and the continuous attributes as INPUT. We obtain the following result when we select the VIEW menu.



Parameters

Target : 1
Input : 10
Illustrative : 0

Results

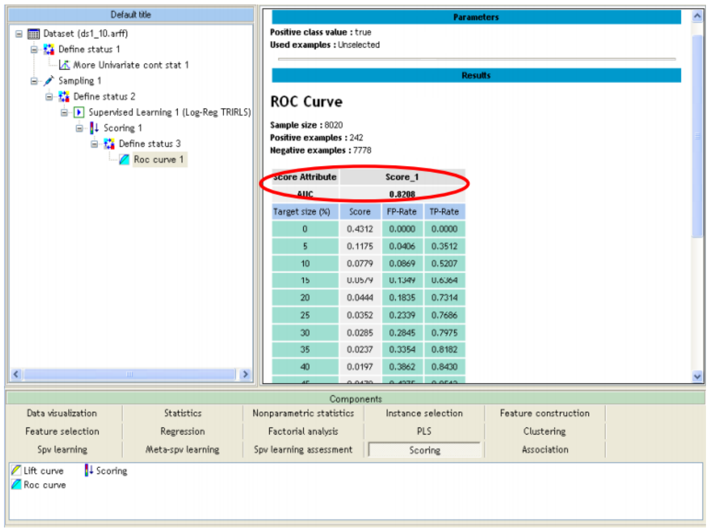
Attribute	Target	Input	Illustrative
V1	-	yes	-
V2	-	yes	-
V3	-	yes	-
V4	-	yes	-
V5	-	yes	-
V6	-	yes	-
V7	-	yes	-
V8	-	yes	-
V9	-	yes	-
V10	-	yes	-
CLASS	yes	-	-

Computation time : 0 ms.
Created at 19/02/2006 17:06:25

Figure 4 – Target and Input Attributes

We want to add the logistic regression component in the diagram. There are two steps: first, we add a “meta-supervised” component (SUPERVISED LEARNING from META-SPV LEARNING palette); second, we embed the LOG-REG TRIRLS component (SPV LEARNING palette). The results are displayed in a new window. The confusion matrix is not relevant in our context because our dataset is unbalanced. The most important is that our classifier can set the “positive” examples before the “negative” ones according a computed “score” which is proportional to the “positive” conditional probabilities.

To compute this score, we set the SCORING component (SCORING) in the diagram; we select the PARAMETERS menu in order to define the “TRUE” value of the class attribute as “positive”. To produce the ROC curve, we add a new DEFINE COMPONENT in the diagram; we set as TARGET the class attribute, and the generated attribute SCORE_1 as INPUT. We can select several INPUT attributes here and compare the performances of various learning algorithms. Last, we add a ROC CURVE component in the diagram. We select “TRUE” as the “positive” label; the computation must be realized on unselected examples (test set). We obtain the results (VIEW menu). We do not build the curve but a table with all necessary values (TRUE POSITIVE rate, FALSE POSITIVE rate).



Parameters

Positive class value : true
Used examples : Unselected

Results

ROC Curve

Sample size : 8020
Positive examples : 242
Negative examples : 7778

Score Attribute	Score_1
AUC	0.8208

Target size (%)	Score	FP-Rate	TP-Rate
0	0.4312	0.0000	0.0000
5	0.1175	0.0406	0.3512
10	0.0779	0.0869	0.5207
15	0.0579	0.1349	0.6364
20	0.0444	0.1835	0.7314
25	0.0352	0.2339	0.7686
30	0.0285	0.2845	0.7975
35	0.0237	0.3354	0.8182
40	0.0197	0.3862	0.8430

Table 5 – The Tanagra ROC Table

2.4 Orange

The main window of Orange is similar to Weka interface: tools are in the top of the window; there is a workspace where we can define the stream diagram. Orange can handle TXT file format. We select the FILE tool, and it is automatically added in the workspace. We can select the dataset to load with the OPEN menu. Orange offers nice graphical tools for descriptive statistics. We select the ATTRIBUTE STATISTICS component. We connect FILE to this new component; the execution is automatically started. We can see the result with the OPEN menu. As with Weka, the last column is the class attribute. We can modify the selection with the SELECT ATTRIBUTES component (DATA). For this dataset, the default selection is right.

Next we perform logistic regression. First, we add the LOGISTIC REGRESSION from the CLASSIFY palette; second, we add the TEST LEARNER component (EVALUATE). We set as “train – test” the evaluation process in this last component, the learning set size is 70% (OPEN menu). We connect the LOGISTIC component to TEST LEARNERS, and SELECT ATTRIBUTES to TEST LEARNERS. In this last connection, a dialog box appears which enables us to select the right dataset. When we confirm this connection, the computation is automatically started. We can display the results with the OPEN menu of TEST LEARNERS.

Next we produce the ROC curve. Orange has an interactive graphical tool for ROC curve computation. We set this component (ROC ANALYSIS from EVALUATE) in the diagram. We connect the TEST LEARNERS to this new component, so our final network looks like this:

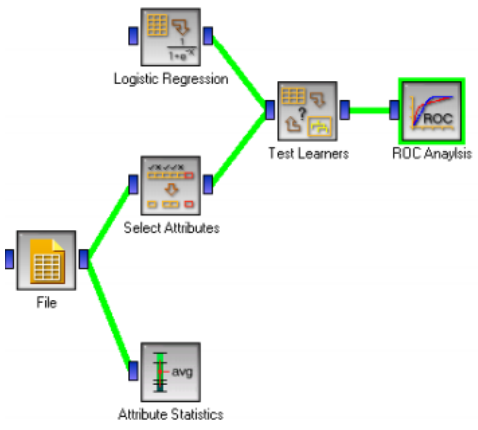


Figure 6 – ROC Diagram

When we activate the OPEN menu of ROC ANALYSIS, we obtain the following chart.

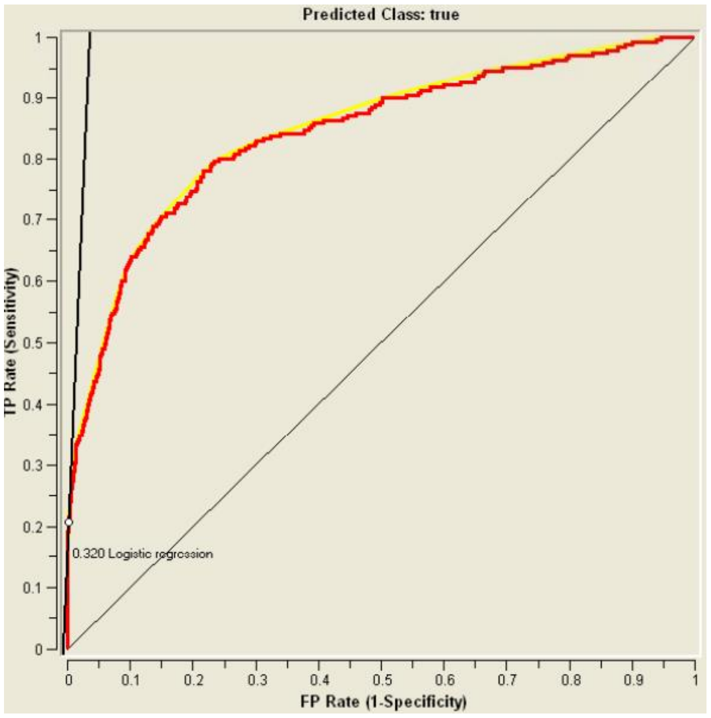


Figure 7 – The Orange ROC Curve

2.5 Task Observations

So that the results are really comparable, it’s necessary to subdivide the file in training set and test set, then run the tools on the same datasets. Orange and Weka use the same strategy. The dataset is subdivided into two files. Because they produce graphs, their diagrams can handle several data sources. Both of them were good for producing a ROC curve.

Tanagra adopts another strategy. We must add a new column in the dataset; it defines the role of each example. Then we replace the SAMPLING component with the SELECT EXAMPLES component in the previous diagram, the other components of the diagram are the same one. Tanagra is not really good for producing a ROC curve.

Personal recommendation: either Weka or Orange. Weka offers more options, while Orange allows an existing ROC curve to be quickly updated. Final decision is up to the experimenter.

3. Building a Decision Tree

3.1 Overview

When we build a decision tree from a dataset, we much follow the following steps:

1) Import the dataset

2) Select the class attribute (TARGET) and the descriptors (INPUT)

3) Choose the induction algorithm

4) Viewing the decision tree

5) Use cross-validation to obtain an error rate estimate

We use the HEART.TXT (UCI Irvine) – there are 270 examples in the dataset.

3.2 Weka

We can build a decision tree in two ways: either through the Explorer or Knowledge Flow. We will use the Knowledge Flow window for two reasons: to keep things consistent (since we already used it to build a ROC curve) and because the other method is quicker and easier, so it doesn't require explanation.

The CSV LOADER enables handling the text file format. We select the HEART.TXT dataset with the CONFIGURE contextual menu. By default, the target attribute is the last column; the others are the input attributes. We have the right configuration in our dataset. On the other hand, we must explicitly select the learning set in the Weka diagram. We add the TRAINING SET MAKER (EVALUATION tab) in the diagram; all examples are used for the construction of the decision tree. We choose the DATASET connection when we connect the LOADER component to this last component.

We add the J48 component (decision tree algorithm, CLASSIFIERS tab). We set the connection between TRAINING SET MAKER and J48 (training set connection). We have two visual representations in Weka: a textual representation (for when we have a lot of nodes in the tree) and a graphical representation (if we want it to look pretty). We select the latter one (GRAPH VIEWER – VISUALIZATION tab) and use the GRAPH connection.

In order to start the execution, we select the first node of the diagram and click on the START LOADING contextual menu. When the computation is achieved, we can select the last component (GRAPH VIEWER) and click on the SHOW GRAPH menu. The decision tree has 18 leaves.

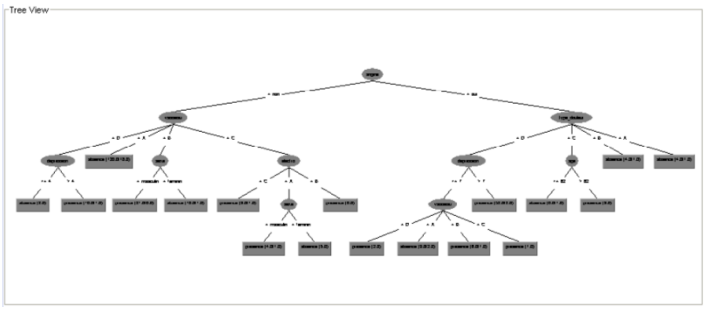
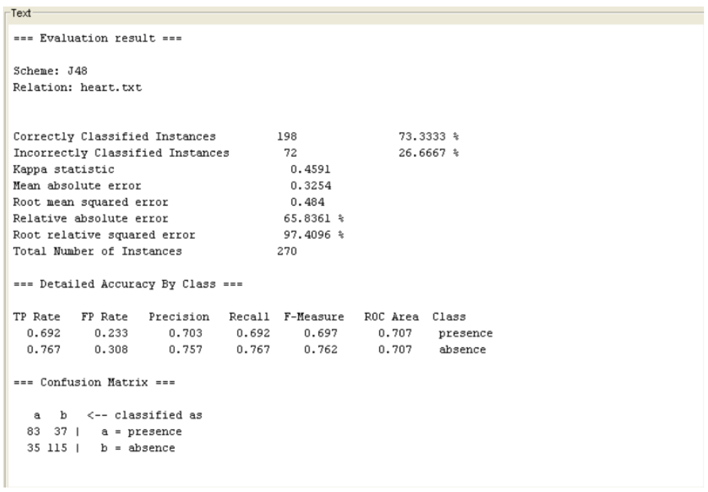


Figure 8 – Weka Decision Tree

Weka has sophisticated error rate estimation, but needs to create a new sequence of components to do that. We need to add the following components:

- CROSS VALIDATION FOLD MAKER (EVALUATION), which builds folds (DATASET connection)
- Decision tree J48 component (CLASSIFY); we have to set the same parameters as the precedent J48 component. We must connect twice the CROSS VALIDATION FOLD MAKER to this component, for the training and the test sets.
- CLASSIFIER PERFORMANCE EVALUATION (EVALUATION) computes the error rate in each fold. We use the BATCH CLASSIFIER output of J48.
- The TEXT VIEWER component displays the results

One again, we select the START LOADING of the CSV LOADER component in order to start the execution. The SHOW RESULTS menu of TEXT VIEWER displays the following results.



```
=== Evaluation result ===

Scheme: J48
Relation: heart.txt


Correctly Classified Instances         198               73.3333 %
Incorrectly Classified Instances        72               26.6667 %
Kappa statistic                          0.4591
Mean absolute error                      0.3254
Root mean squared error                  0.484
Relative absolute error                 65.8361 %
Root relative squared error             97.4096 %
Total Number of Instances              270

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.692     0.233      0.703     0.692     0.697      0.707    presence
  0.767     0.308      0.757     0.767     0.762      0.707    absence

=== Confusion Matrix ===

   a   b   <-- classified as
  83  37 |   a = presence
  35 115 |   b = absence
```

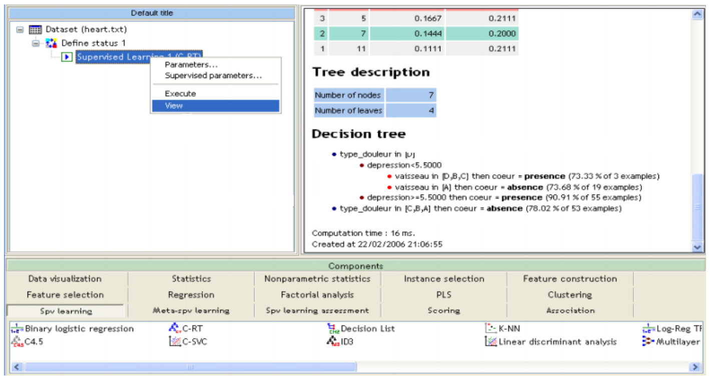
Figure 9 – Weka Estimated Error Rate

The classification accuracy is 73.33%, and the error rate is 26.67%.

3.3 Tanagra

We create a new diagram and import the dataset with the FILE/NEW menu. We need to define the role of attributes. We add the DEFINE STATUS component (we use the button in the toolbar) in the diagram. We set COEUR as TARGET, the other attributes as INPUT. We want to use the Classification and Regression Tree algorithm. There are two steps when we want to define a supervised learning process in Tanagra: we insert a meta-supervised learning component from the META SPV LEARNING tab or embed the learning algorithm, C-RT, from the SPV learning tab.

In order to view the decision tree, we click on the VIEW menu of the last component. The output is a textual representation of tree. The resubstitution error rate is 19.63% and the tree has 4 leaves (4 rules).



3	5	0.1667	0.2111
2	7	0.1444	0.2000
1	11	0.1111	0.2111

Tree description

Number of nodes	7
Number of leaves	4

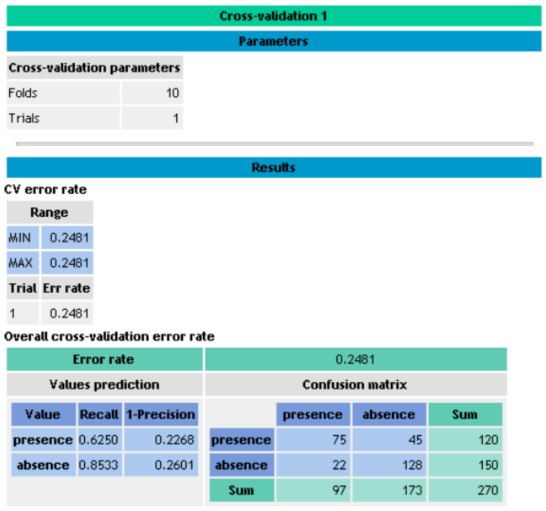
Decision tree

- type_douleur in [D]
  - depression<5.5000
    - vaisseau in [D,B,C] then coeur = **presence** (73.33 % of 3 examples)
    - vaisseau in [A] then coeur = **absence** (73.68 % of 19 examples)
  - depression>=5.5000 then coeur = **presence** (90.91 % of 55 examples)
- type_douleur in [C,B,A] then coeur = **absence** (78.02 % of 53 examples)

Computation time : 16 ms.
Created at 22/02/2006 21:06:55

Figure 10 – Tanagra's Version of a Decision Tree

We want to compute the error rate with a cross-validation resampling method. We add the CROSS-VALIDATION component from the SPV LEARNING ASSESMENT tab. We set the number of folds to 10, and the number of repetition to 1. From Figure XX, we see that the approximate percentage of correctly classified instances is 75.19%, and the error rate is 24.81%.



Cross-validation 1

Parameters

Cross-validation parameters	
Folds	10
Trials	1

Results

CV error rate

Range	
MIN	0.2481
MAX	0.2481
Trial	**Err rate**
1	0.2481

Overall cross-validation error rate

Error rate	0.2481

Values prediction

Value	Recall	1-Precision
presence	0.6250	0.2268
absence	0.8533	0.2601

Confusion matrix

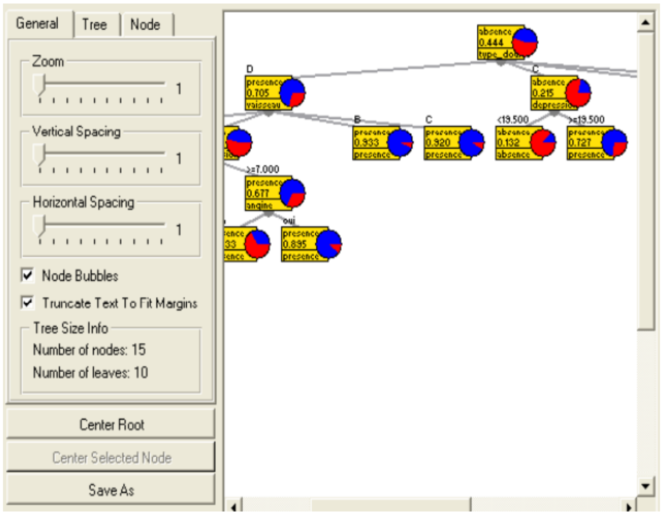
	presence	absence	Sum
presence	75	45	120
absence	22	128	150
Sum	97	173	270

Figure 11 – Tanagra Estimated Error Rate

3.4 Orange

As mentioned before, Orange can handle text file format using tab to separate. When we select the tool, a new component is inserted in the diagram. We can select the file with the OPEN contextual menu. By default, the target attribute is the last column; the others are the input attributes. In the case of this dataset, the default assumptions are correct. We can add the classification tree component (CLASSIFY tab) in our diagram. We connect this component with the dataset component.

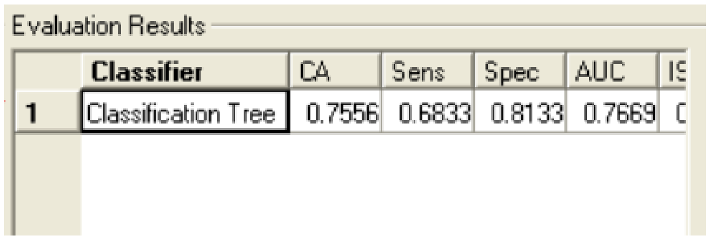
We can display the tree in a text viewer. This is recommended if we have numerous nodes in the tree. There is also a graphical viewer that is more pleasant (CLASSIFICATION TREE VIEWER 2D – CLASSIFY tab). We connect the CLASSIFICATION TREE component to this last one. We click on the OPEN menu in order to display the tree. There are 10 rules (leaves) in our tree.



Zoom 1
Vertical Spacing 1
Horizontal Spacing 1
☑ Node Bubbles
☑ Truncate Text To Fit Margins
Tree Size Info
Number of nodes: 15
Number of leaves: 10

Figure 12 – Orange Decision Tree

The TEST LEARNERS component (EVALUATE tab) enables to compute the cross-validation error rate estimate. We connect to this new component the classification tree. This component becomes operational when we will have specified the data source and the training method -- it is possible to connect simultaneously several learning methods, which makes it possible to realize, very easily, the comparison of performances. We thus carry out the right connections, and then we display the results using the OPEN menu.



Evaluation Results

	Classifier	CA	Sens	Spec	AUC	IS
1	Classification Tree	0.7556	0.6833	0.8133	0.7669	C

Figure 13 – Orange Estimated Error Rate

The classification accuracy is 75.56%, so the error rate is 24.44%. We can also interactively choose another resampling method, such as Leave-One-Out or Random Sampling.

3.5 Task Obsevations

Cross-validation is a very popular method of error rate estimation, especially when we have few examples in our dataset. Our tools can easily handle this process.

Personal recommendation: Weka. Even though Weka scored about 2% lower in correct classifications, its interface is just too good not to use. The Explorer especially makes it quite simple to build and experiment with decision trees. The output it produces is really compact and easy to understand. Classification error is included in the output when going through the Explorer, so the experimenter doesn’t need to perform additional steps.

4. Comparison of Supervised Methods

4.1 Overview

In this section, we will perform a comparison of learning algorithms using a predefined learning and test set. Very often, we use the accuracy to compare the performances of the algorithms. We then select the method that is the most accurate. So that the comparison is rigorous, it is necessary that we use the same dataset in training and test phase. We chose to compare the performances of a SVM (linear kernel), a logistic regression and a decision tree.

We use the BREAST dataset (UCI Irvine). We have a binary class attribute (benign or malignant tumor), 9 continuous descriptors, and 699 examples. We have selected 499 examples for the training phase, 200 examples for the test. We use the same subdivision for our three packages.

4.2 Weka

We again open up the Knowledge Flow interface. We have to use two separate dataset with Weka. We set two ARFF LOADER components in the diagram and select the datasets. We must now specify the role of these datasets in the diagram. We use the TRAINING SET MAKER and TEST SET MAKER components (EVALUATE tab). We set the necessary connections.

We set three learning algorithms components (CLASSIFIERS tab) in the diagram. About SMO, we check that we really use a linear kernel (exponent = 1, not RBF kernel). We connect the three TRAINING SET MAKERs, and the three TEST SET MAKERs.

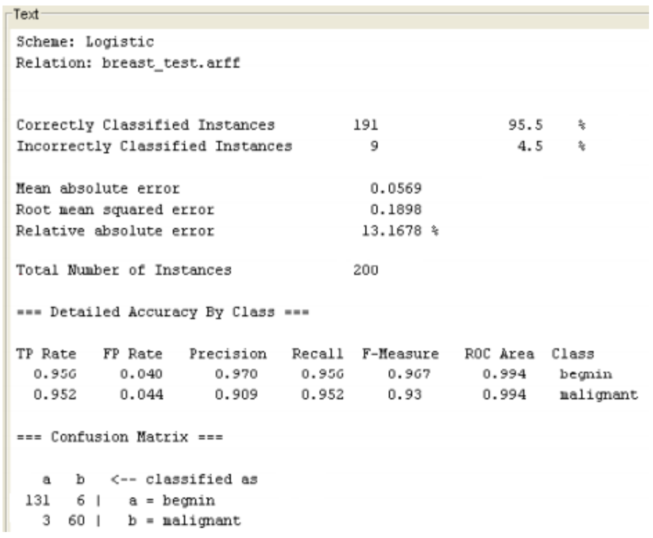
To compute the accuracy of the classifiers, we set CLASSIFIER PERFORMANCE EVALUATOR component (EVALUATION), one for each learning method. The type of the connection must be BATCH CLASSIFIER. We set the TEXT VIEWER (VISUALIZATION) component in order to display the results. Since only one component is necessary, it makes it possible to join together the results in the same window. We use the TEXT connection.

The execution of the diagram is done into two steps:

1) We select the START LOADING menu of the first data source (learning set), and the prediction models are computed;

2) We select the START LOADING menu of the second data source, the test set, and the accuracy of the models is computed.

When we select the SHOW RESULTS menu of the TEXT VIEWER component, we can see the detailed results for each learning algorithm. Below is the output for Logistic Regression.



```
Scheme: Logistic
Relation: breast_test.arff


Correctly Classified Instances        191               95.5    %
Incorrectly Classified Instances        9                4.5    %

Mean absolute error                     0.0569
Root mean squared error                 0.1898
Relative absolute error                13.1678 %

Total Number of Instances             200

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.956     0.040      0.970     0.956     0.967      0.994    begnin
  0.952     0.044      0.909     0.952     0.93       0.994    malignant

=== Confusion Matrix ===

   a   b   <-- classified as
 131   6 |   a = begnin
   3  60 |   b = malignant
```

Figure 14 – Weka Logistic Regression Results

We obtain the following accuracy rates:

- Decision tree: 93.5% (error rate 6.5%)
- Logistic regression: 95.5% (error 4.5%)
- Linear SVM: 95.5% (error rate 4.5%)

We note that SVM and Logistic regression have the same accuracy rate but not the same confusion matrix; the structure of the error is not the same one.

4.3 Tanagra

Compared to the two other packages, Tanagra uses a tree to represent the treatments. That simplifies its structure, but induced a strong constraint; it is not possible to specify two data sources. It is consequently necessary to prepare the data differently.

We use BREAST_ALL.XLS. All the examples are gathered in the same sheet; we add a new column, which enables us to distinguish training set and test set (STATUS).

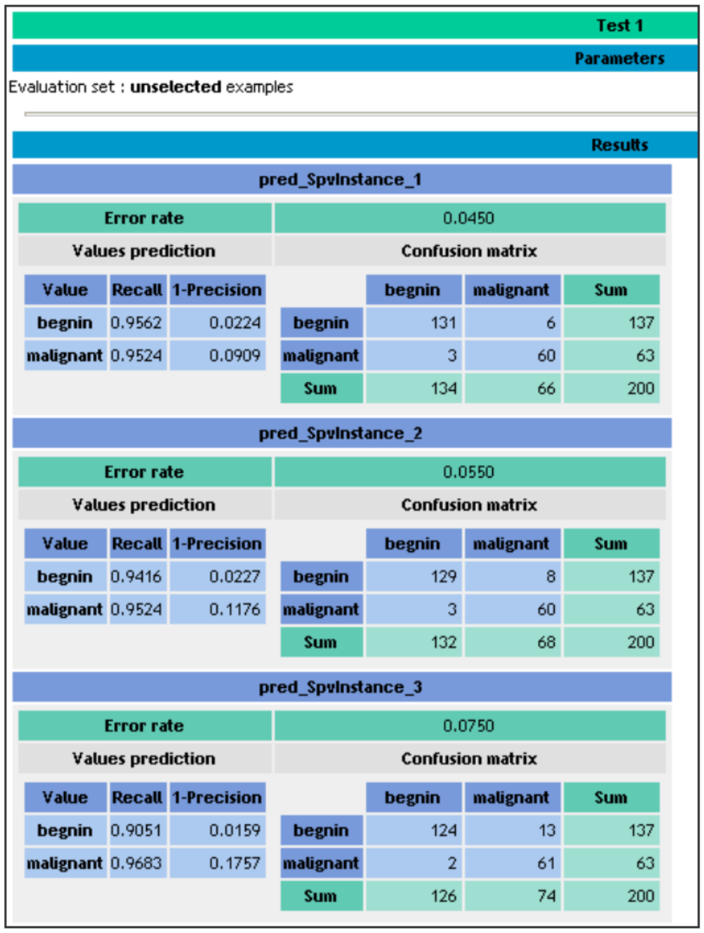
Next, we execute Tanagra. We create a new diagram and import BREAST_ALL.XLS. We use the SELECT EXAMPLES component (INSTANCE SELECTION) in order to select the active (training) examples. When we execute the component (VIEW menu), we see that we have 499 selected examples for the computations. We add a DEFINE STATUS component in order to select the TARGET attribute (CLASS); the continuous attributes are INPUT. We do not need use the STATUS attribute here.

We must insert the three learning methods in the diagram. We present the detailed operation for the logistic regression. There are two steps when we want to add a supervised algorithm in the diagram: first, we insert a meta-supervised component that defines the aggregation strategy (SUPERVISED LEARNING – META SPV LEARNING tab). Second, we embed in this component the learning strategy BINARY LOGISTIC REGRESSION (SPV LEARNING tab). This implementation of logistic regression is slightly slower than the others, but it has the advantage of providing a series of additional statistics.

We insert the other learning methods in the diagram: SVM (C-SVC) and the decision tree (C-RT). We start the execution of the whole diagram by selecting the VIEW menu of the last component.

To compare the performances, we must insert again the DEFINE STATUS component in the diagram by clicking on the shortcut into the toolbar. We set the CLASS attribute as TARGET; the predictions of each method are the INPUT attributes. We note that these predictions are computed for the whole dataset, including the unselected examples. We add the TEST (SPV LEARNING ASSESMENT tab) in the diagram. We must not forget to specify that the confusion matrix computation must be done on the unselected examples, which represents the test set.

The view menu displays the following results:



Test 1

Parameters

Evaluation set : **unselected** examples

Results

pred_SpvInstance_1

Error rate	0.0450

Values prediction

Value	Recall	1-Precision
begnin	0.9562	0.0224
malignant	0.9524	0.0909

Confusion matrix

	begnin	malignant	Sum
begnin	131	6	137
malignant	3	60	63
Sum	134	66	200

pred_SpvInstance_2

Error rate	0.0550

Values prediction

Value	Recall	1-Precision
begnin	0.9416	0.0227
malignant	0.9524	0.1176

Confusion matrix

	begnin	malignant	Sum
begnin	129	8	137
malignant	3	60	63
Sum	132	68	200

pred_SpvInstance_3

Error rate	0.0750

Values prediction

Value	Recall	1-Precision
begnin	0.9051	0.0159
malignant	0.9683	0.1757

Confusion matrix

	begnin	malignant	Sum
begnin	124	13	137
malignant	2	61	63
Sum	126	74	200

Figure 15 – Tanagra Algorithm Comparison

The classification accuracy rates are:

- Decision tree: 92.5% (error rate 7.5%)
- Logistic regression: 95.5% (error rate 4.5%)
- Linear SVM: 94.5% (error rate 5.5%)

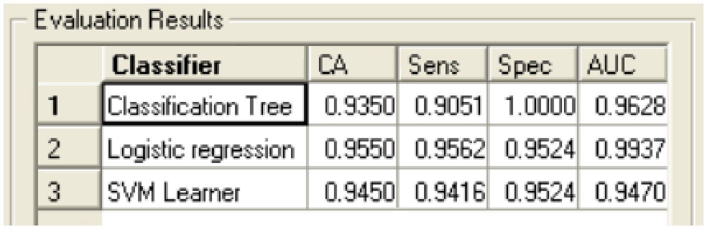
4.4 Orange

We divide the whole dataset into two files: BREAST_TRAIN.TXT for training, BREAST_TEST.TXT for testing. We set two data access components in the diagram; we parameterize them by activating the OPEN menu. We want to compare three learning methods from the CLASSIFY tab, so we set them in the diagram. The comparison can be gathered in only one component, the TEST LEARNERS component, from the EVALUATE tab. We connect the three learning method to this new component.

We must now specify which data to use for the training. We connect the first data source [FILE] to the TEST LEARNERS. A dialog box appears which is important, because it enables us to check that we are transmitting the training set (DATA). The training phase is automatically started.

In the next step, we connect the second data source [FILE 2] to the TEST LEARNERS component. Orange considers that this second data source is the test set (SEPARATE TEST DATA).

To display the results, we select the OPEN menu of the TEST LEARNERS component.


Evaluation Results

	Classifier	CA	Sens	Spec	AUC
1	Classification Tree	0.9350	0.9051	1.0000	0.9628
2	Logistic regression	0.9550	0.9562	0.9524	0.9937
3	SVM Learner	0.9450	0.9416	0.9524	0.9470

Figure 16 – Orange Algorithm Comparison

We check the "Test on test data" option. Various statistics are available; we are interested primarily in these accuracies:

- Classification tree: 93.5% (error rate 6.5%)
- Logistic regression: 95.5% (error rate 4.5%)
- Linear SVM: 94.5% (error rate 5.5%)

4.5 Task Observations

It is easy to perform a comparison of algorithms using a predefined test set with Orange, Weka and Tanagra. The results can be slightly different between the packages. This is not surprising because of the heuristic nature of learning algorithms. The effect of the implementation choices also is not negligible. Nevertheless, very large differences would have been alarming.

Personal recommendation: Orange. In Tanagra, we cannot specify two data sources, which makes an analysis such as this one rather inconvenient. Weka produces different outputs for each algorithm, which makes the results less coherent. Orange's method is the fastest and simplest. While its results may not be as detailed as Tanagra's, the simplicity of the process is too tempting to ignore.

5. Computing Association Rules

5.1 Overview

We must perform the following steps if we want to compute association rules from a dataset:

1) Import the dataset

2) Select the descriptors

3) Set the parameters of the association rule algorithm i.e. the minimal support and the minimal confidence

4) Execute the algorithm and visualize the rules

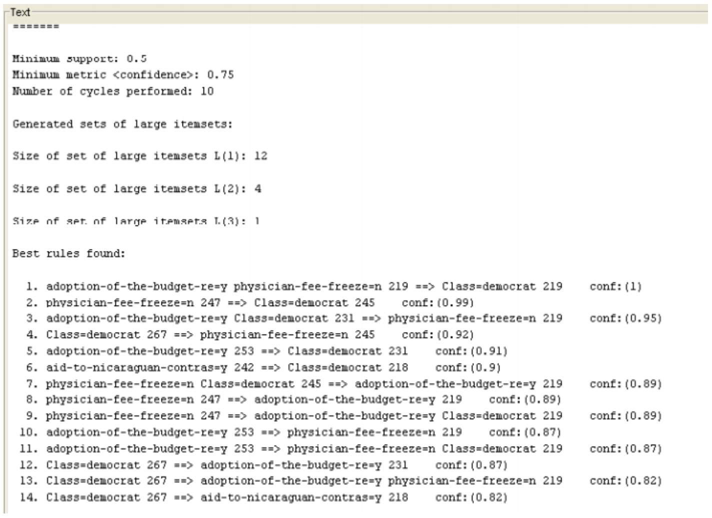
Our three tools use attribute-based datasets. Each attribute-value couple becomes an item which can be used for generating rules. We use the VOTE.TXT dataset from the UCI Irvine repository.

5.2 Weka

We'll use the Knowledge Flow once again. The CSV LOADER enables to handle text file format. We select the VOTE.TXT dataset with the CONFIGURE contextual menu. The CSV LOADER enables to handle text file format. We select the VOTE.TXT dataset with the CONFIGURE contextual menu. The default selections are all instances and all attributes, so we must add only the APRIORI component from the ASSOCIATION tab in the diagram.

We use the DATASET connection type. The CONFIGURE contextual menu allows to set the parameters values. LOWERBOUNDMINSUPPORT is the minimal support of rules (set to 0.5); MINMETRIC (set to 0.75) is the minimal confidence, if we set CONFIDENCE as METRIC TYPE; NUMRULES set the maximal number of rules that we can generate (set to 100).

In order to visualize the rules, we add the TEXT VIEWER component in the diagram; we use the TEXT connection. To execute the computation, we click on the START LOADING of the first component (CSV LOADER). We can see the rules by clicking the SHOW RESULTS menu of the TEXT VIEWER component.


Text

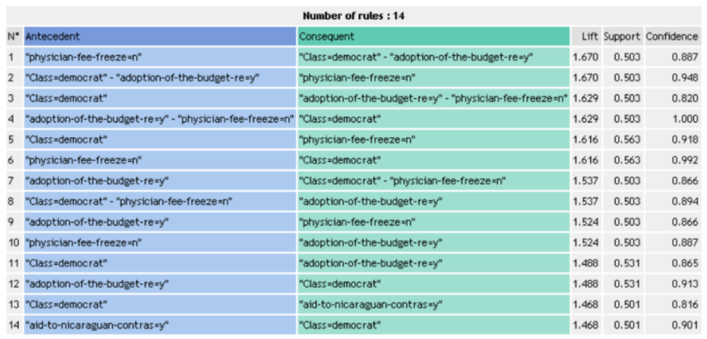
```
=======

Minimum support: 0.5
Minimum metric <confidence>: 0.75
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 4

Size of set of large itemsets L(3): 1

Best rules found:

  1. adoption-of-the-budget-re=y physician-fee-freeze=n 219 ==> Class=democrat 219    conf:(1)
  2. physician-fee-freeze=n 247 ==> Class=democrat 245    conf:(0.99)
  3. adoption-of-the-budget-re=y Class=democrat 231 ==> physician-fee-freeze=n 219    conf:(0.95)
  4. Class=democrat 267 ==> physician-fee-freeze=n 245    conf:(0.92)
  5. adoption-of-the-budget-re=y 253 ==> Class=democrat 231    conf:(0.91)
  6. aid-to-nicaraguan-contras=y 242 ==> Class=democrat 218    conf:(0.9)
  7. physician-fee-freeze=n Class=democrat 245 ==> adoption-of-the-budget-re=y 219    conf:(0.89)
  8. physician-fee-freeze=n 247 ==> adoption-of-the-budget-re=y 219    conf:(0.89)
  9. physician-fee-freeze=n 247 ==> adoption-of-the-budget-re=y Class=democrat 219    conf:(0.89)
 10. adoption-of-the-budget-re=y 253 ==> physician-fee-freeze=n 219    conf:(0.87)
 11. adoption-of-the-budget-re=y 253 ==> physician-fee-freeze=n Class=democrat 219    conf:(0.87)
 12. Class=democrat 267 ==> adoption-of-the-budget-re=y 231    conf:(0.87)
 13. Class=democrat 267 ==> adoption-of-the-budget-re=y physician-fee-freeze=n 219    conf:(0.82)
 14. Class=democrat 267 ==> aid-to-nicaraguan-contras=y 218    conf:(0.82)
```

Figure 17 – The Weka Generated Rules

5.3 Tanagra

First, we must import the dataset. We create a new diagram and import the dataset with the FILE/NEW menu. We select the VOTE.TXT dataset. We add a DEFINE STATUS component in the diagram; we set all attributes as INPUT. There are various algorithms in Tanagra; some of them come from external libraries. For this experiment, we'll use the standard APRIORI algorithm. We click on the PARAMETERS contextual menu of the component. The minimal support is set to 0.5; the minimal confidence to 0.75; we use only frequent itemsets of cardinal lower or equal to 4; the rules with a LIFT lower than 1 are removed.

To compute the rules, we select the contextual VIEW menu in order to see the rules. We obtain the same 14 rules as Weka.



Number of rules : 14

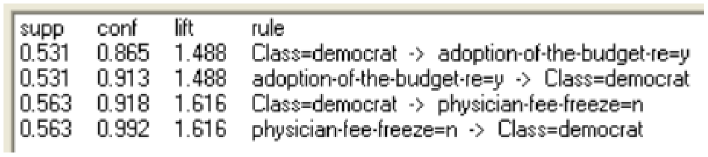
N°	Antecedent	Consequent	Lift	Support	Confidence
1	"physician-fee-freeze=n"	"Class=democrat" - "adoption-of-the-budget-re=y"	1.670	0.503	0.887
2	"Class=democrat" - "adoption-of-the-budget-re=y"	"physician-fee-freeze=n"	1.670	0.503	0.948
3	"Class=democrat"	"adoption-of-the-budget-re=y" - "physician-fee-freeze=n"	1.629	0.503	0.820
4	"adoption-of-the-budget-re=y" - "physician-fee-freeze=n"	"Class=democrat"	1.629	0.503	1.000
5	"Class=democrat"	"physician-fee-freeze=n"	1.616	0.563	0.918
6	"physician-fee-freeze=n"	"Class=democrat"	1.616	0.563	0.992
7	"adoption-of-the-budget-re=y"	"Class=democrat" - "physician-fee-freeze=n"	1.537	0.503	0.866
8	"Class=democrat" - "physician-fee-freeze=n"	"adoption-of-the-budget-re=y"	1.537	0.503	0.894
9	"adoption-of-the-budget-re=y"	"physician-fee-freeze=n"	1.524	0.503	0.866
10	"physician-fee-freeze=n"	"adoption-of-the-budget-re=y"	1.524	0.503	0.887
11	"Class=democrat"	"adoption-of-the-budget-re=y"	1.488	0.531	0.865
12	"adoption-of-the-budget-re=y"	"Class=democrat"	1.488	0.531	0.913
13	"Class=democrat"	"aid-to-nicaraguan-contras=y"	1.468	0.501	0.816
14	"aid-to-nicaraguan-contras=y"	"Class=democrat"	1.468	0.501	0.901

Figure 18 – The Tanagra Generated Rules

5.4 Orange

To import the dataset, we select the FILE tool and a new component is inserted in the diagram. We can select the file with the OPEN contextual menu. In order to compute the rules, we add ASSOCIATION RULES component. All examples and attributes are used. We click on the OPEN menu for parameters settings.

The rules are automatically computed when we connect the FILE component to ASSOCIATION RULE. We add the ASSOCIATION RULE PRINT component for rules visualization. We click on the OPEN menu in order to view the rules.



```
supp   conf   lift   rule
0.531  0.865  1.488  Class=democrat -> adoption-of-the-budget-re=y
0.531  0.913  1.488  adoption-of-the-budget-re=y -> Class=democrat
0.563  0.918  1.616  Class=democrat -> physician-fee-freeze=n
0.563  0.992  1.616  physician-fee-freeze=n -> Class=democrat
```

Figure 19 – The Orange Generated Rules

We obtain only 4 rules in Orange, compared to 14 with Weka and Tanagra. From the results, Orange seems to prefer shorter rules.

5.5 Task Observations

The three tools are all very simple to use for generating association rules. They are largely sufficient for the majority of the analyses. The situation is a little more difficult if we wish to treat big databases with thousands of items. The number of generated rules can become very high and the performance of the machine may suffer due to the large amount of computations.

Personal recommendation: Tanagra. It has the simplest process and its input is easiest to understand. Orange doesn't produce enough rules. Weka is a good alternative, but the process is a bit more involving and its output is a little harder to read. However, the process in Weka is simpler if we go through the Explorer, but Tanagra is still a favorite.

6. Conclusion

In this report, we evaluated three different open-source data mining tools. We experimented with different tasks and compared their performance. We also offered thoughts on which tool is most appropriate for a particular job. For three of the tasks, only one tool is most appropriate, while for the fourth, two tools are appropriate. Of the three tools evaluated, I'd say Weka is the best-rounded one. Its interface is easy to use and understand. It offers much more preprocessing techniques than the other two tools. The produced output is easy to read, and can be copied as plain text and pasted into a document. Finally, it offers multiple ways to accomplish tasks, as demonstrated by using Knowledge Flow to perform the described jobs, while also being possible to accomplish them in the Explorer. As a final recommendation, I'd say Weka is one of the top open-source data mining tools available on the Internet.

7. References

1) Abdullah H. Wahbeh, Qasem A. Al-Radaideh, Mohammed N. Al-Kabi, Emad M. Al-Shawakfa. *A Comparison Study between Data Mining Tools over some Classification Methods.* IJACSA 2008.

2) Ralf Mikut, Markus Reischl. *Data mining tools*. WIREs Data Mining Knowl Discov 2011 00 1–13.

3) Ricco Rakotomalala. *Tanagra: a free software for research and academic purposes*. In Proceedings of EGC'2005, RNTI-E-3, vol. 2, pp.697-702, 2005 (in French).

4) John F. Elder IV, Dean W. Abbott. *A Comparison of Leading Data Mining Tools.* In Fourth International Conference on Knowledge Discovery & Data Mining, Friday, August 28, 1998,New York, New York.

5) J. Alcalá-Fdez , L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, F. Herrera. *KEEL: a software tool to assess evolutionary algorithms for data mining problems.* Published online: 22 May 2008, Springer-Verlag.

6) Janez Demsar, Tomaz Curk, Ales Erjavec, Crt Gorup, Tomaz Hocevar, Mitar Milutinovic, Martin Mozina, Matija Polajnar, Marko Toplak, Anze Staric, Miha Stajdohar, Lan Umek, Lan Zagar, Jure Zbontar, Marinka Zitnik, Blaz Zupan. *Orange: Data Mining Toolbox in Python*. Journal of Machine Learning Research 14 (2013) 2349-2353, August 2013.

7) Janez Demsar, Blaz Zupan, Gregor Leban, Tomaz Curk. *Orange: From Experimental Machine Learning to Interactive Data Mining*. J.-F. Boulicaut et al. (Eds.): PKDD 2004, LNAI 3202, pp. 537–539, 2004.

8) Ian H. Witten, Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. ISBN: 978-0120884070. Published June 10, 2005.

9) Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, Sally Jo Cunningham. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations.* Working Paper Series ISSN 1170-487X, August 1999.

10) Ricco Rakotomalala. *Comparison of data mining tools*. ERIC Laboratory, Lumière University Lyon 2, France, 2006.

11) B. Carey, C. Marjaniemi, D. Sautter. *A Methodology for Evaluating and Selecting Data Mining Software.* Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 6, January 05-08, 1999.