Evaluating Data Mining Tool: WEKA

Leila Homaeian
Nasimeh Asgarian
Department of Computing Science
University of Alberta

CMPUT 695: Principles of Knowledge Discovery in Data
Assignment 2
Instructor: Dr. Osmar Zaiane

November 5, 2004

.

1 Software Information

WEKA is a data mining software/library which is written in Java. It is a collection of machine learning algorithms for data mining tasks. You can apply these algorithms directly to a dataset or use them from your own Java code. WEKA contains comprehensive set of data pre-processing tools, learning algorithms for classification, regression, clustering, association rules and evaluation methods. It also has graphical user interfaces including data visualization. WEKA has an environment for comparing learning algorithms. It is open source software, and you can use it for developing new machine learning techniques.

We evaluated WEKA 4.3, which is developed at the University of Waikato. You can find the complete information and download the software from the following link:

`http://www.cs.waikato.ac.nz/ml/weka/index.html`

2 Platforms Supported

WEKA 3.4 requires Java 1.4 or higher. You can download and install the WEKA for Windows, Mac OS X, and other platforms, like Linux.

3 Data Pre-processing Techniques

WEKA has large set of data-preprocessing tools. The data pre-processing tools in WEKA are called "filters". WEKA contains algorithms for both supervised and unsupervised filtering for attributes and instances. For supervised filtering it has *attribute selection*, *discretization*, *class order*, and *nominal to binary* for attributes and *re-sampling*, and *spread subsample* for instances. For unsupervised filtering it has many methods like *normalization*, *random projection*, and *discretization* for attributes and *randomization*, and *range removal* for instances.

Data pre-processing is very important, for instance if an algorithm cannot be applied to a specific data type, these pre-processing techniques may be used in order to manipulate data and make it suitable for that algorithm. For example, the Apriori algorithm does not handle numerical data, but we can discretize the data and then apply the algorithm.

4 Implemented Learning Algorithms

4.1 Classification

Classifiers in WEKA handle and predict both numerical and nominal data types. The learning schemes implemented in WEKA include:

- *Bayes:* is a technique for estimating probabilities of individual feature values, given a class, from training data and to then allow the use of these probabilities to classify unseen instances.

- *Decision tree classifiers:* can break down a complex decision making problem into a group of simpler decisions, so they provide a solution which is easier to interpret.
- *instance-based classifiers:* generates classification predictions using only specific instances
- and many more.

WEKA also has "meta-classifiers" like *boosting*, *stacking*, and *error-correcting output codes*. Figure 1 shows the interface WEKA provides to choose a classifier. In this example we chose J48 Tree classifier.

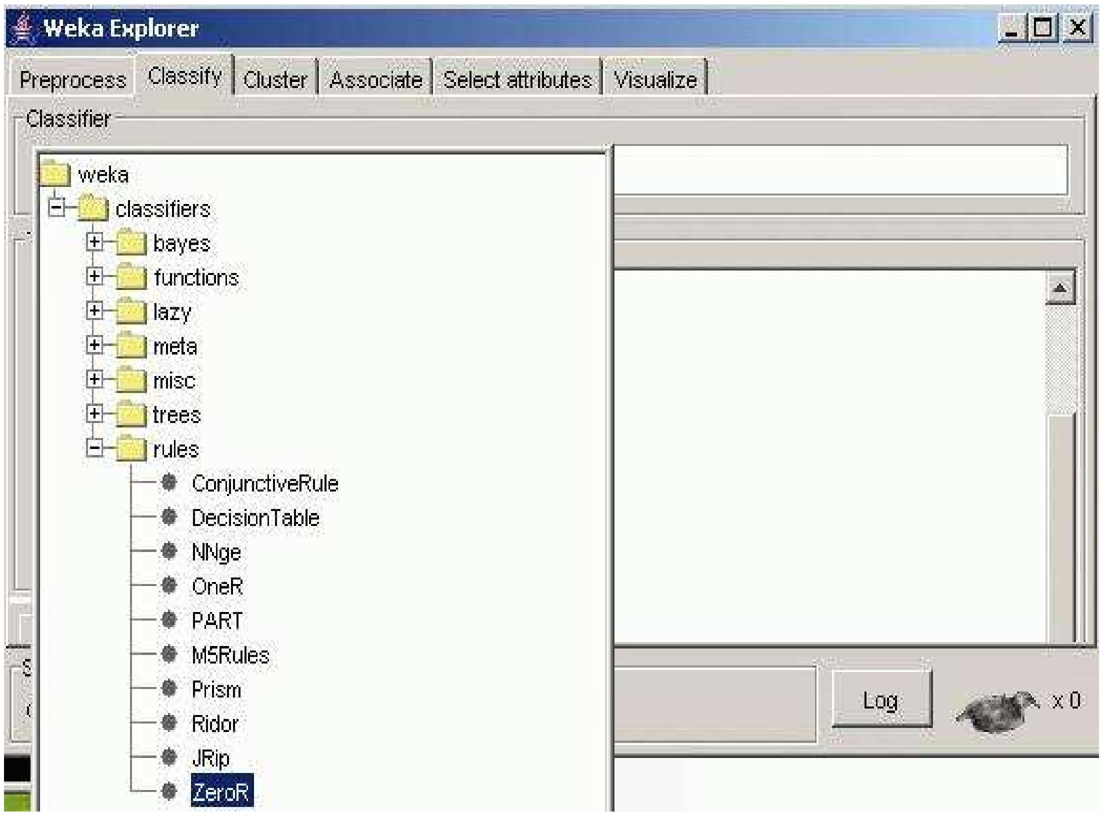
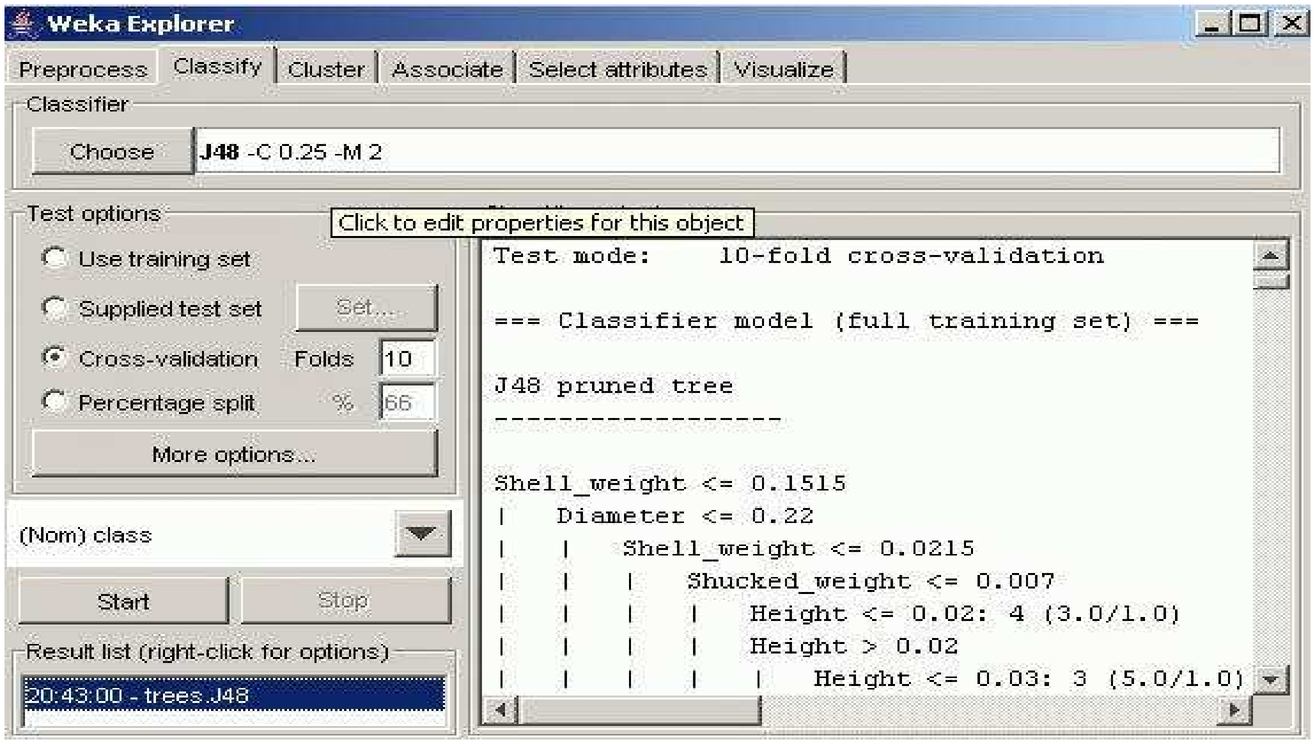


Figure 1: Classifier Interface

Figure 2, 3, and 4 show the resulting tree, Detailed Accuracy by Class and the Confusion Matrix WEKA computed, respectively.

4.2 Clustering

WEKA contains clustering algorithms for finding groups of similar instances in a dataset. The clusters can be visualized and compared to the given "true" clusters. The schemes implemented in WEKA include:



J48 -C 0.25 -M 2

Cross-validation Folds 10

Percentage split % 66

(Nom) class

20:43:00 - trees.J48

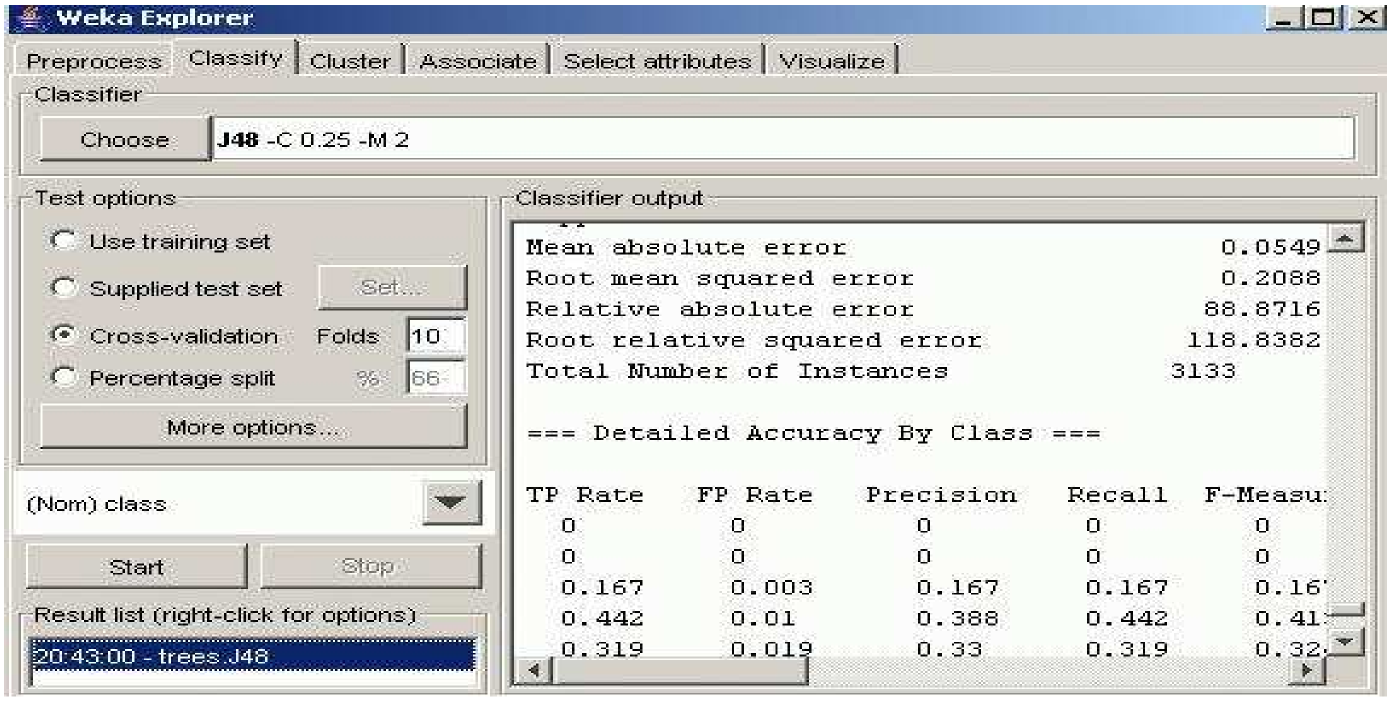
```
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

Shell_weight <= 0.1515
|   Diameter <= 0.22
|   |   Shell_weight <= 0.0215
|   |   |   Shucked_weight <= 0.007
|   |   |   |   Height <= 0.02: 4 (3.0/1.0)
|   |   |   |   Height > 0.02
|   |   |   |   |   Height <= 0.03: 3 (5.0/1.0)
```

Figure 2: Tree Representation



J48 -C 0.25 -M 2

Cross-validation Folds 10

Percentage split % 66

(Nom) class

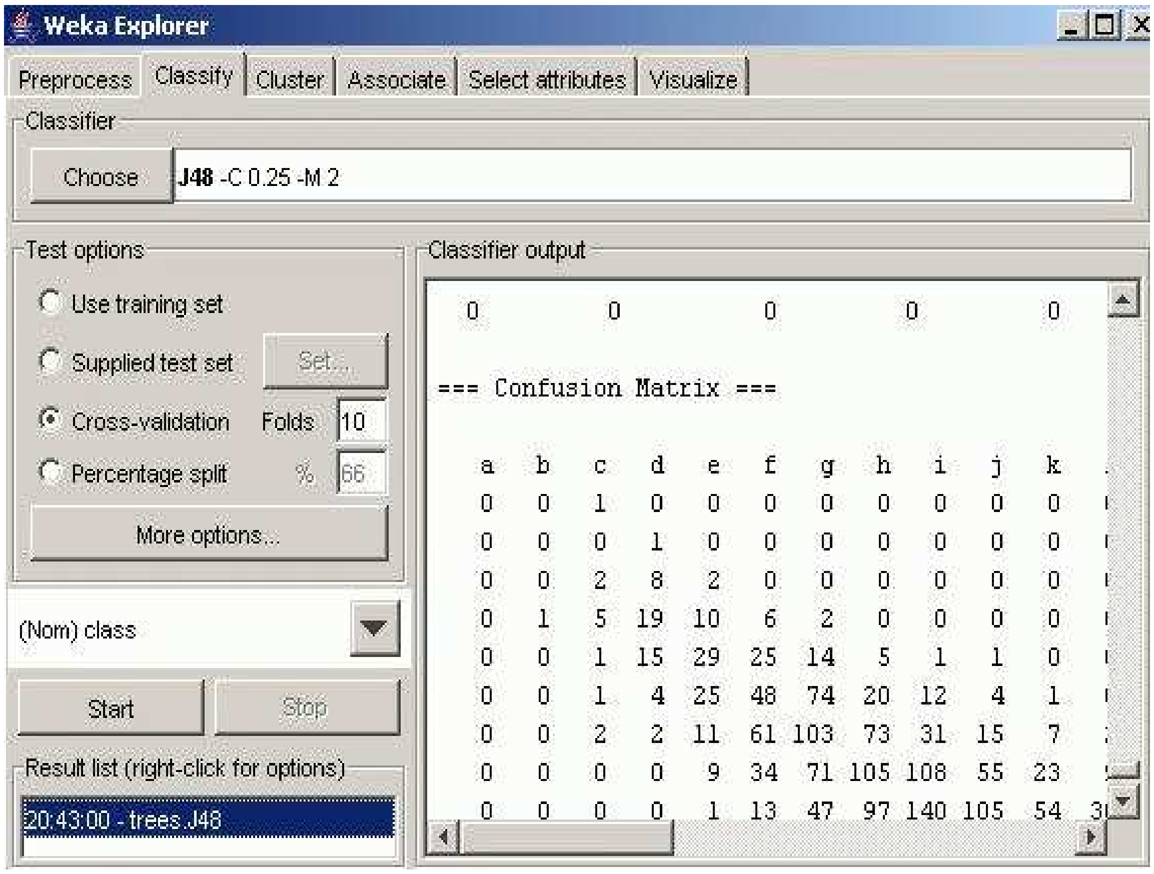
20:43:00 - trees.J48

```
Mean absolute error                          0.0549
Root mean squared error                      0.2088
Relative absolute error                     88.8716
Root relative squared error                118.8382
Total Number of Instances                  3133

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measu
  0          0          0            0          0
  0          0          0            0          0
  0.167      0.003      0.167        0.167      0.16
  0.442      0.01       0.388        0.442      0.41
  0.319      0.019      0.33         0.319      0.32
```

Figure 3: Detailed Accuracy by Class



J48 -C 0.25 -M 2

Cross-validation Folds 10

% 66

(Nom) class

20:43:00 - trees.J48

```
  0          0           0          0          0

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k
   0   0   1   0   0   0   0   0   0   0   0
   0   0   0   1   0   0   0   0   0   0   0
   0   0   2   8   2   0   0   0   0   0   0
   0   1   5  19  10   6   2   0   0   0   0
   0   0   1  15  29  25  14   5   1   1   0
   0   0   1   4  25  48  74  20  12   4   1
   0   0   2   2  11  61 103  73  31  15   7
   0   0   0   0   9  34  71 105 108  55  23
   0   0   0   0   1  13  47  97 140 105  54
```

Figure 4: Confusion Matrix

- *Simple KMeans*
- *Farthest first*
- *Density based clusterer*
- and few more.

4.3 Association Rules

WEKA has implementation of *Apriori*, *predictive Apriori*, and *Tertius* algorithms for finding association rules. It can identify statistical dependencies between groups of attributes (humidity=normal 7 ⇒ play=yes 6   conf: (0.86)). The user can set the minimum support and confidence for finding the rules. But it works only with discrete data.

4.4 Attribute Selection

WEKA has a panel to let the user find out which attributes (or subset of attributes) are the most predictive ones. The attribute selection panel has two parts, *evaluation methods* and *search methods.* Each part has different algorithms and the user can use most of the combination of algorithms for both parts. For evaluation methods, we can name:

- *Principal component:* performing principal components analysis/transformation.
- *Gain ratio attribute evaluation:* Evaluating attributes individually by measuring gain ratio with respect to the class.
- *Information gain attribute evaluation* Evaluating attributes individually by measuring information gain with respect to the class.
- etc.

For search method, WEKA has *best-first*, *forward selection*, *random*, *ranking*, and *genetic algorithm.*

5 Data Input and Model Output Options

Data can be imported to WEKA from a file in various formats: ARFF, CSV, C4.5, and binary. But it handles only flat files. Data can also be read from a URL or from an SQL database. "Filters" are the data pre-processing tools in WEKA. WEKA has filters for discretization, normalization, re-sampling, attribute selection, transforming, and combining algorithms.

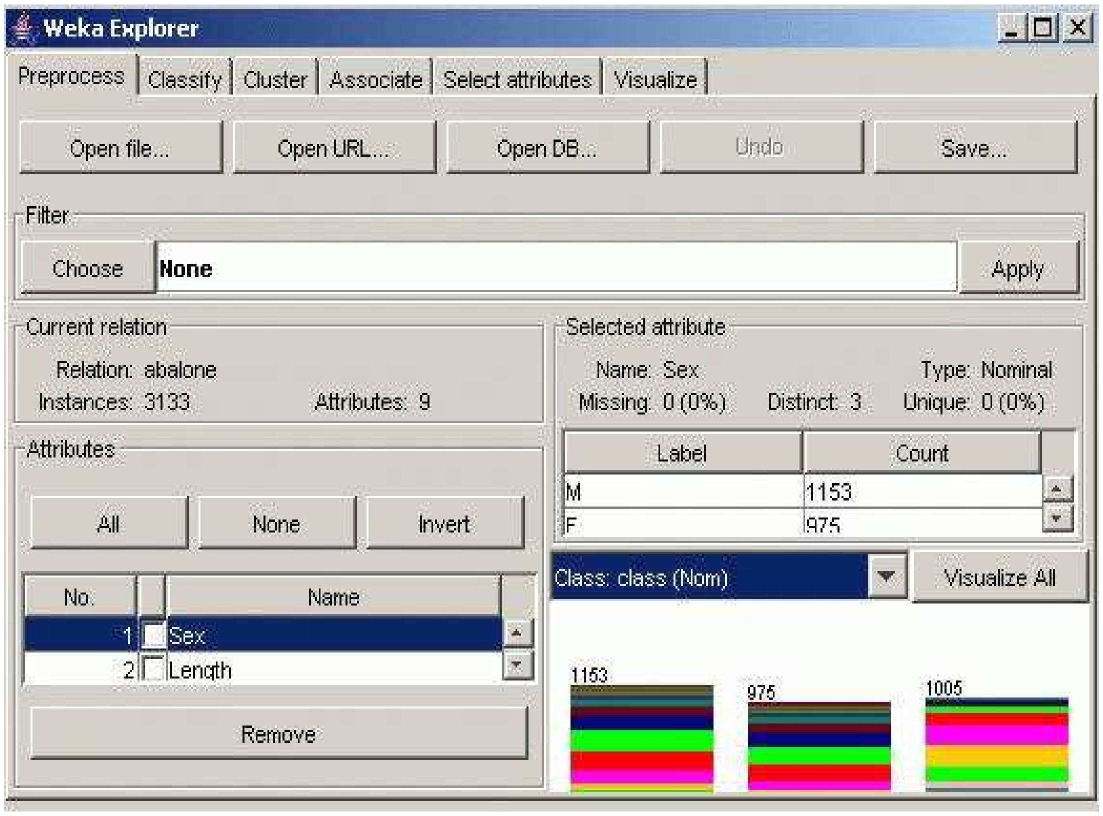
Experimenter can compare different learning schemes easily. The result of the experiments can be written to a file or database. WEKA also has different evaluation options like learning curves, cross-validation, and confusion matrix.

6 Usability

There are good tutorials and documentations for WEKA. Having a friendly user interface, WEKA is easy to get familiar with and use. It is easy to navigate through WEKA since it has different tabs for each data mining task, each of them having specific related operations. The models and reports that WEKA generates are understandable. Not only WEKA supports a variety of platforms, but also the algorithms may be called from command line[1]. One can use WEKA's Java class library to build his/her own data mining/machine learning algorithms.

7 Visualization

WEKA has fairly good visualization abilities. After loading a dataset, the user may choose one attribute then WEKA will visualize the value distribution of it and also it has attribute-based dataset visualization (Figure 5 and 6). Also it is possible to see values of every pair of attributes in 2D space (figure 7). After applying a classifier it provides visualization of the classifier errors, the tree (if applicable), margin curve, threshold curve, and cost curve. It should be mentioned that in case of tree classifiers, if the tree is too big, the visualization is not clear, so the user has to use the generated text representation of the tree (figure 2). In case of clustering algorithms, it visualizes cluster assignments. After applying an attribute selection method, WEKA offers to visualize the reduced dataset.



Filter: None

Current relation
Relation: abalone
Instances: 3133 Attributes: 9

Selected attribute
Name: Sex Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

Label	Count
M	1153
F	975

Attributes

No.		Name
1	☐	Sex
2	☐	Length

Class: class (Nom)

Figure 5: Attribute Value Distribution

8 Scalability

Some of WEKA's algorithms require the entire data to fit in the main memory. So in this case it depends on the machine WEKA runs on. Moreover, some algorithms need enough main memory to run. It failed to apply Linear Regression to a dataset with 5800 instances and 505 numeric attributes (on machines with 512 MB and even 1GB memory) due to lack of memory. (It should be mentioned that WEKA did manage to read and visualize this dataset). Same failure happened when applying J48 Tree classifier to a dataset with 67557 instances and 43 attributes, except that this time it did not manage to visualize the dataset.

In general we do not recommend WEKA for large datasets.

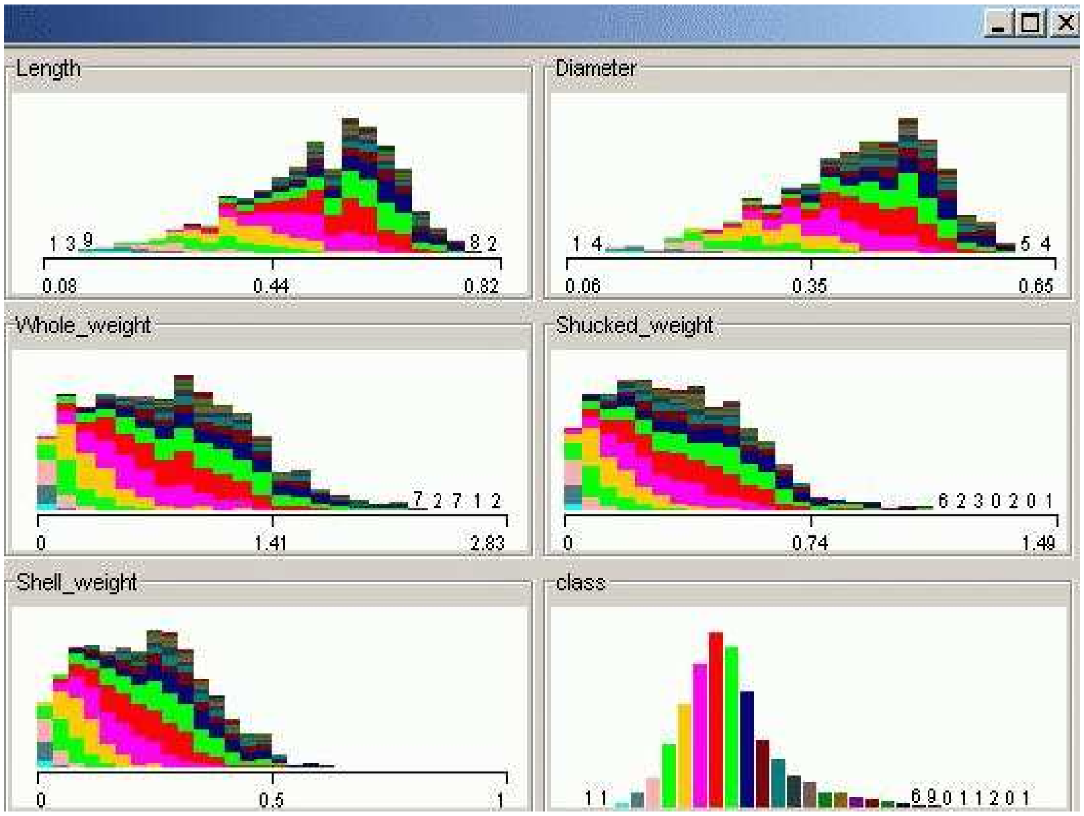


Figure 6: Attribute-based Dataset Visualization

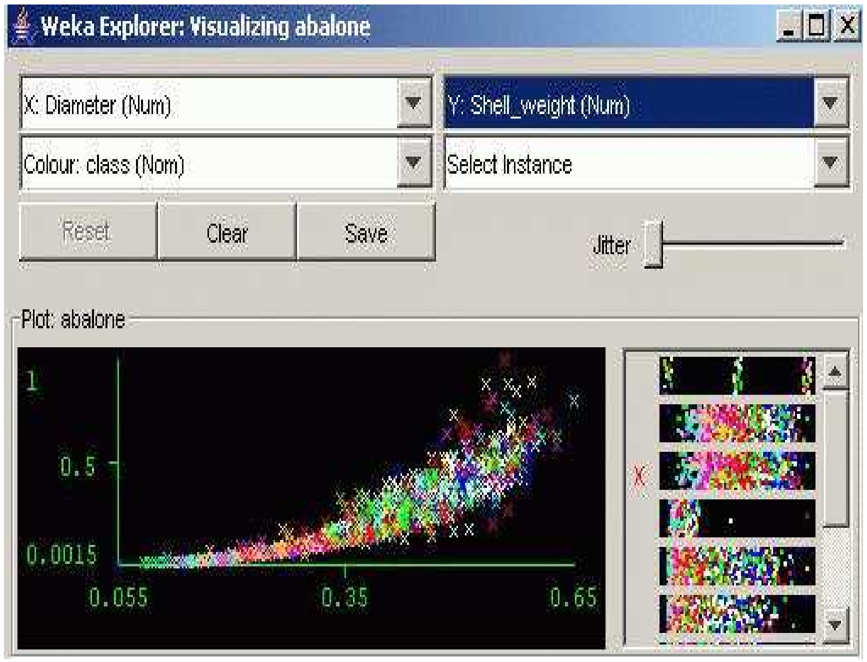


Figure 7: 2D Visualization of two attribute values

9 Knowledge Flow Environment

There is a new graphical user interface for WEKA, called knowledge flow GUI. It is a Java-beans-based interface for setting up an running data mining experiments[2]. Data sources, Classifiers, Visualizers, Evaluator and etc are beans (nodes of a graph), and can be connected by different kind of edges. This feature is specially useful when the user wants a set of tasks to be performed automatically. Figure 8 shows an example of data flows through components. These graphs can be saved and loaded for future use.

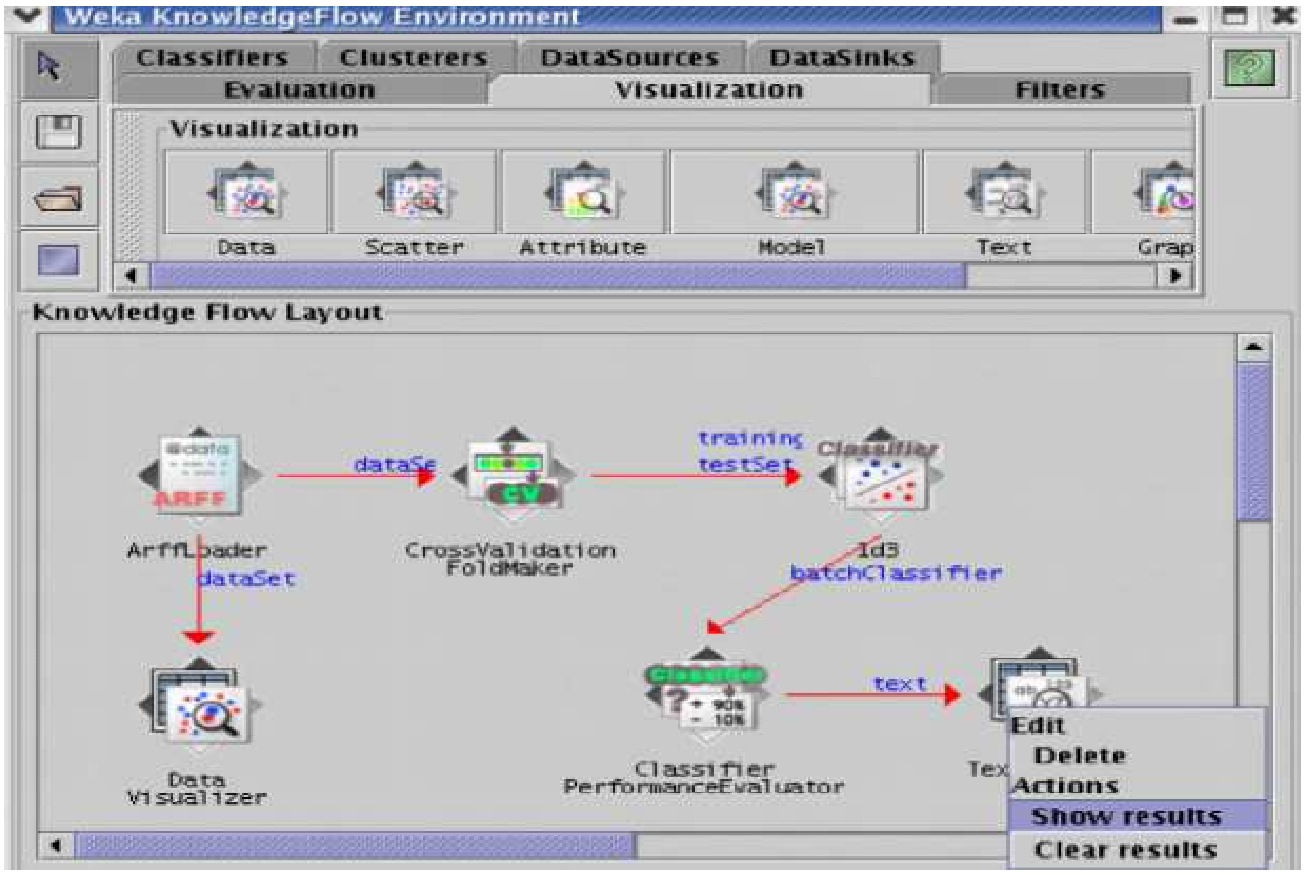


Figure 8: Data Flows through Component

10 Conclusion

10.1 Is the tool worth acquiring

Of course! The extensive number of algorithms WEKA provides is the first feature makes it worth acquiring, next its visualization capabilities. Along with aforementioned features, its ease of use makes WEKA even more useful. In addition, WEKA is an open source software, so it is easily extendable.

We noticed some limitations when evaluating WEKA. First, it only deals with flat files. For example we could not generate association rules from a transactional database. Second, it is not scalable for large datasets.

10.2 Ideal features and characteristics of a data mining tool

Besides basic features like accuracy, and robustness, in our opinion, a data mining tool should be well documented and have a friendly graphical user interface, so that it can address a broad range of users.

Next, it should include a fair amount of data mining algorithms. Also a model that a data mining tool generates, must be easily understandable, and preferably comparable to models generated from other algorithms. The latter feature is useful specially when the user is not certain about one particular algorithm. It is also important that a data mining tool have visualization capabilities which help to understand the result better.

Since most of real datasets are fairly large, a data mining tool should be scalable,too.

Some data mining algorithms cannot be applied to all data types. So it is useful that a data mining tool have data pre-processing steps.

References

[1] "Data Mining: Practical machine learning tools with Java implementations," by Ian H. Witten and Eibe Frank, Morgan Kaufmann, San Francisco, 2000.

[2] Machine Learning with WEKA, presentation by Eibe Frank.