PASED - Canadian Summer School on Practical Analyses of Software Engineering Data

Hands-on predictive models and machine learning for software

Foutse Khomh, Queen's University

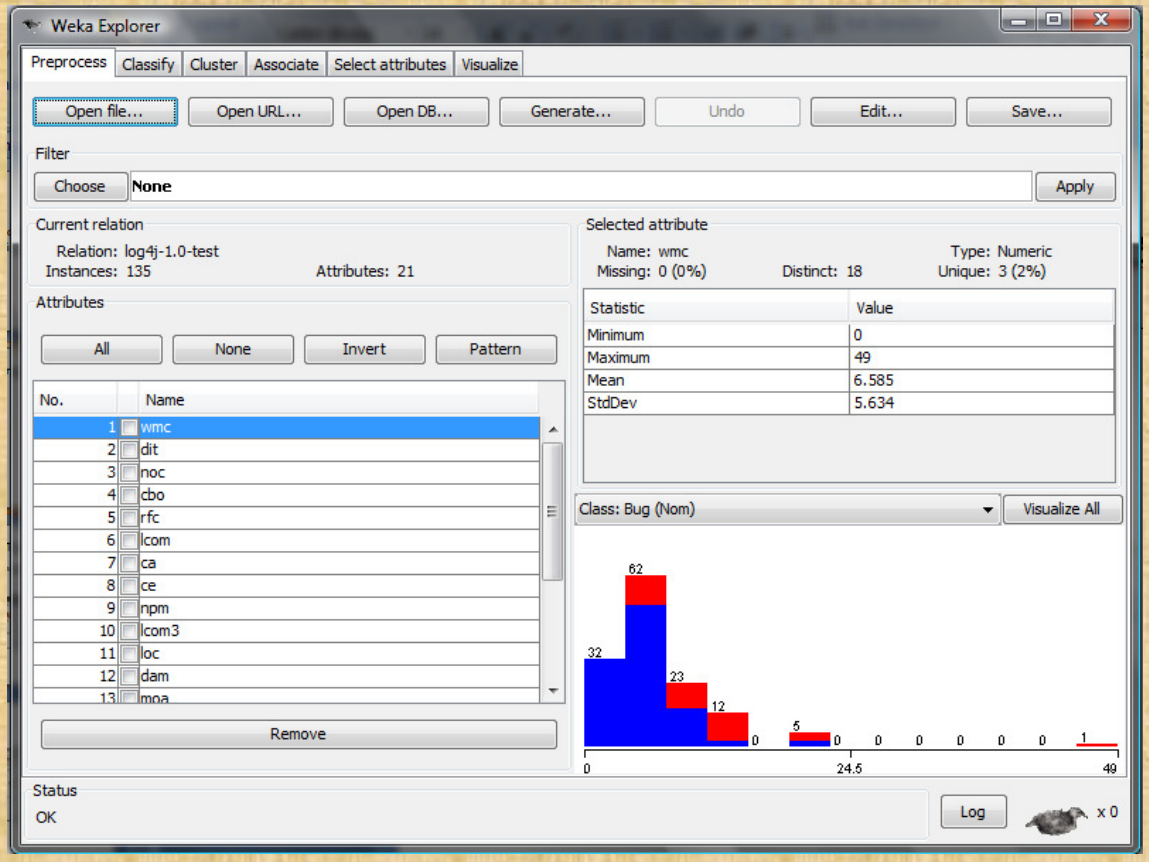
Segla Kpodjedo, École Polytechnique de Montreal

Data Carving

- We use Weka Explorer to filter the data

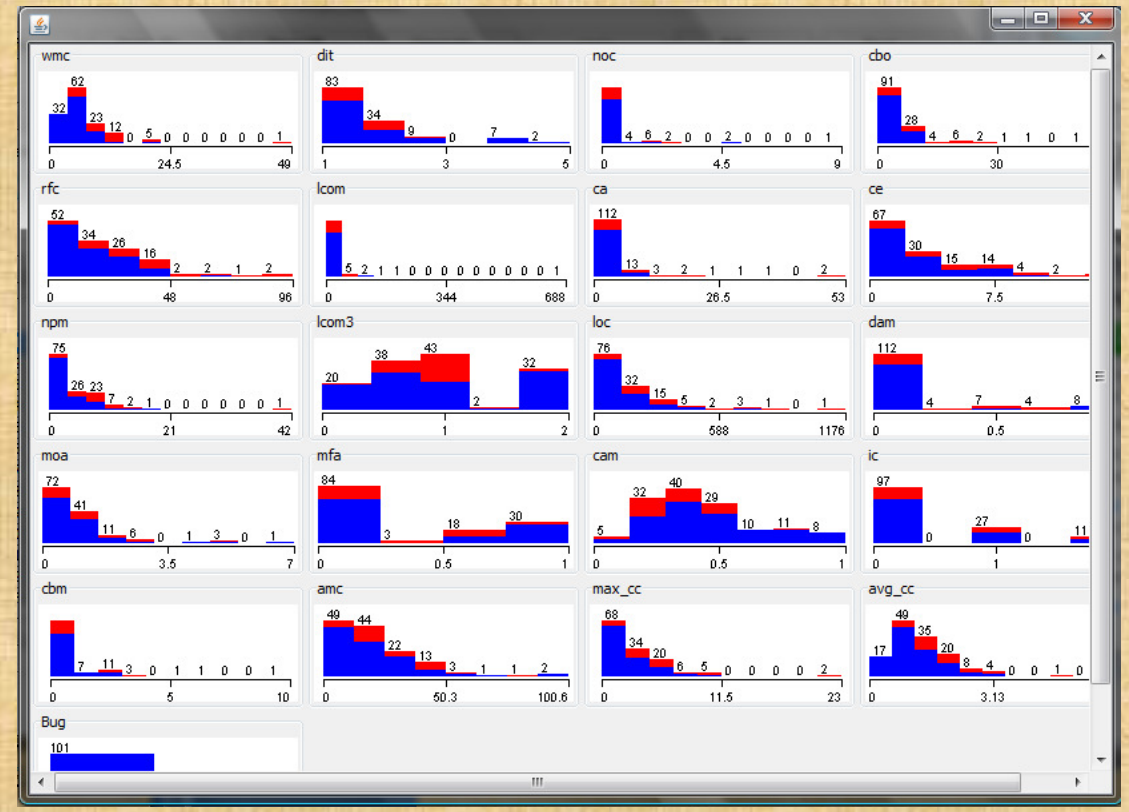
Load a file

- Open the Weka Explorer and load the data (using Open file, load the file log4j-1.0.csv)



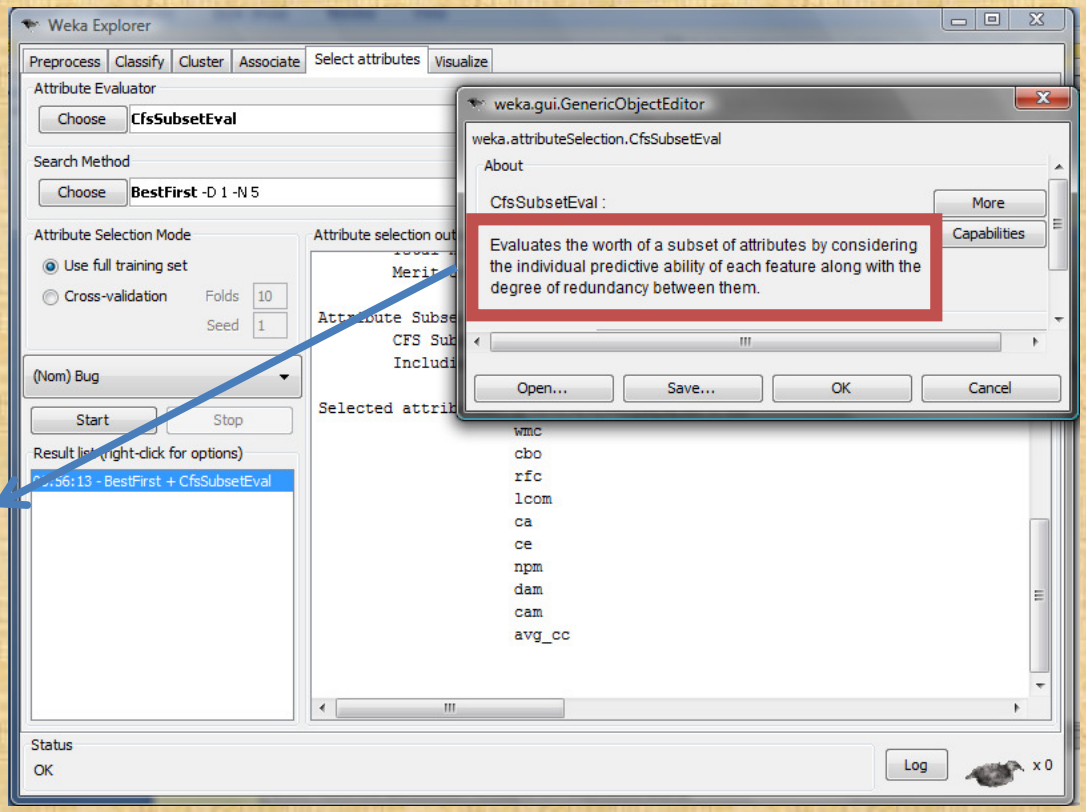
Visualize the distributions

- We can visualize the distributions of the attributes (click on **Visualize all** visible on the last screenshot)
- *Close this window*



Variable selection with Weka

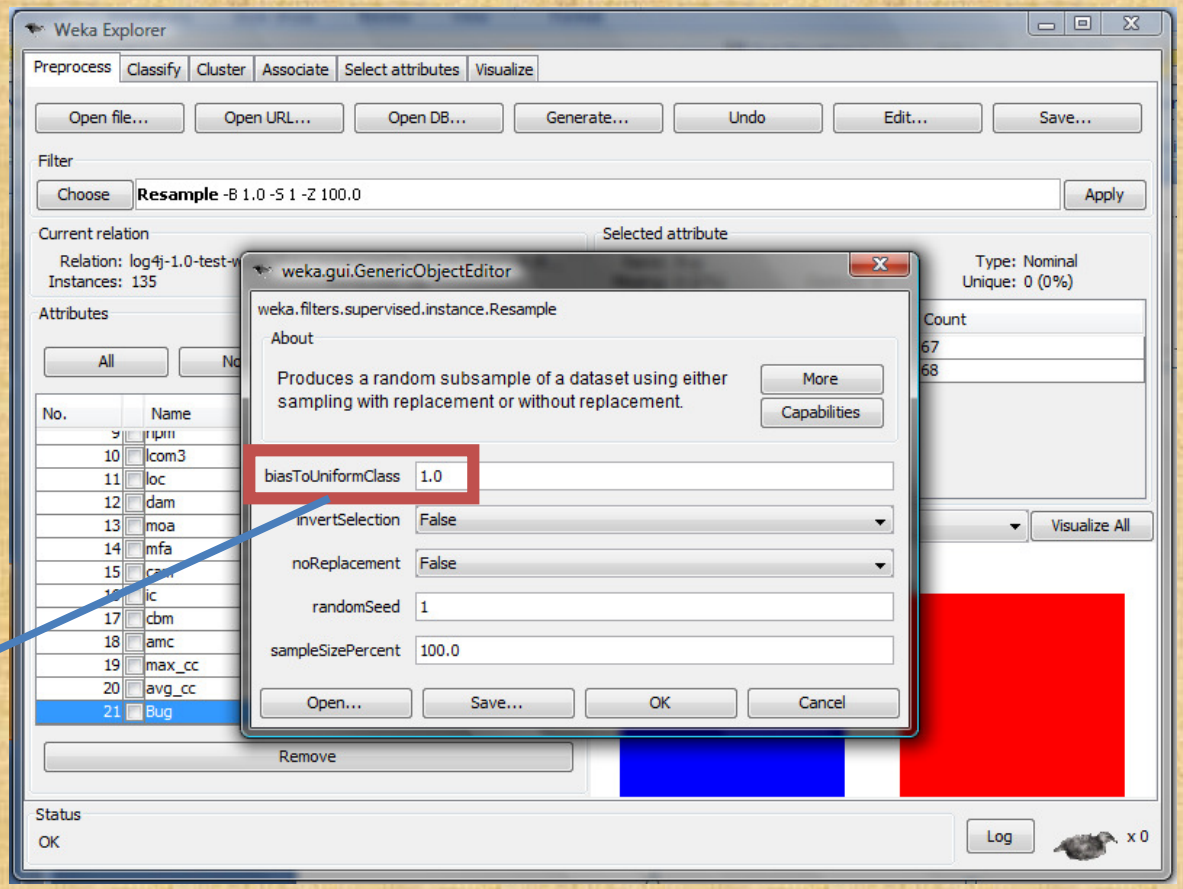
- Select attributes using CFsSubsetEval: it evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.



Resampling

We resample to balance the classes of our dependant variable

- To bias the class distribution toward a uniform distribution, we put the biasToUniformClass to “1”.
- Save the result as rlog4j-1.0.csv

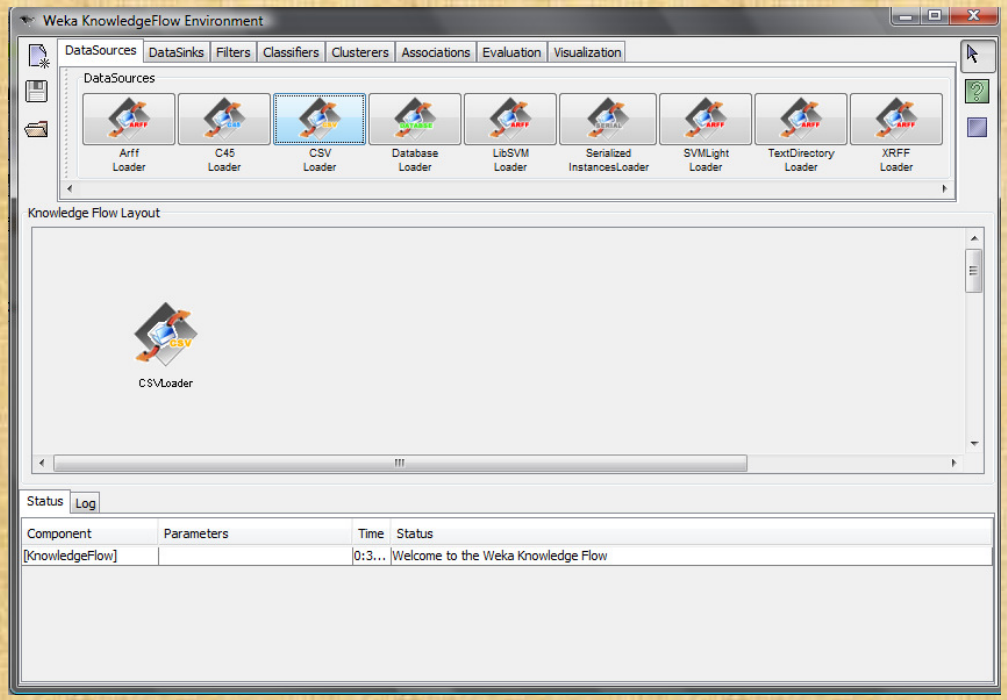


Building and Testing Models

- We use the Weka knowledge flow to train and test our models.

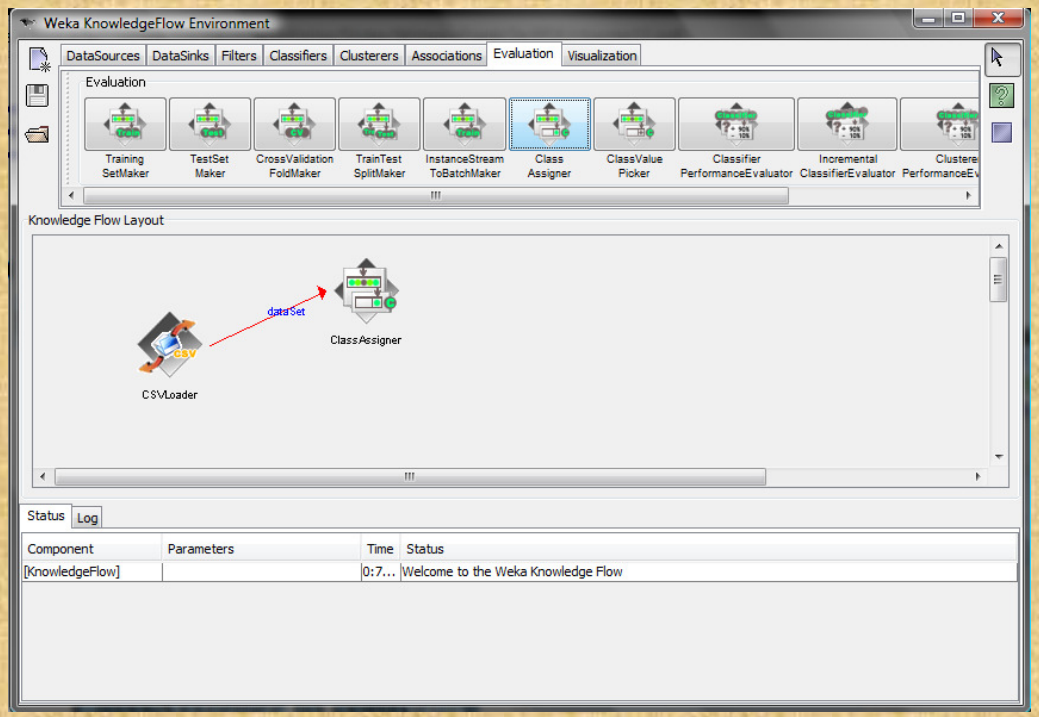
1- Specify a data source

- **DataSource panel**
- Select CSVloader (place it on the Layout)
- Right-click the CSVloader icon to configure by specifying the location of the dataset



2- Specify the class attribute

- Select ClassAssigner from the Evaluation panel right-click the CSVloader icon and select dataset to connect to ClassAssigner
- Right-click the ClassAssigner to configure the target class(by default, it is the last column of your data: here, Bug)

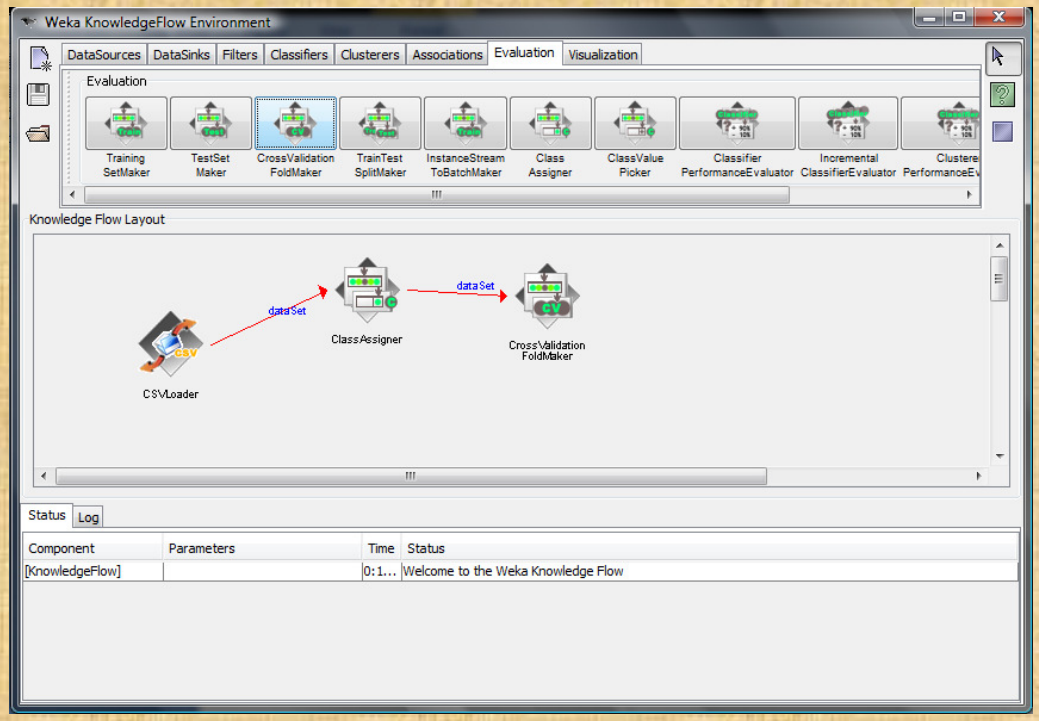


3- Specify cross-validation

- Select CrossValidation FoldMaker from the Evaluation panel

- Right-click the ClassAssigner icon and select dataset to connect to CVFoldMaker

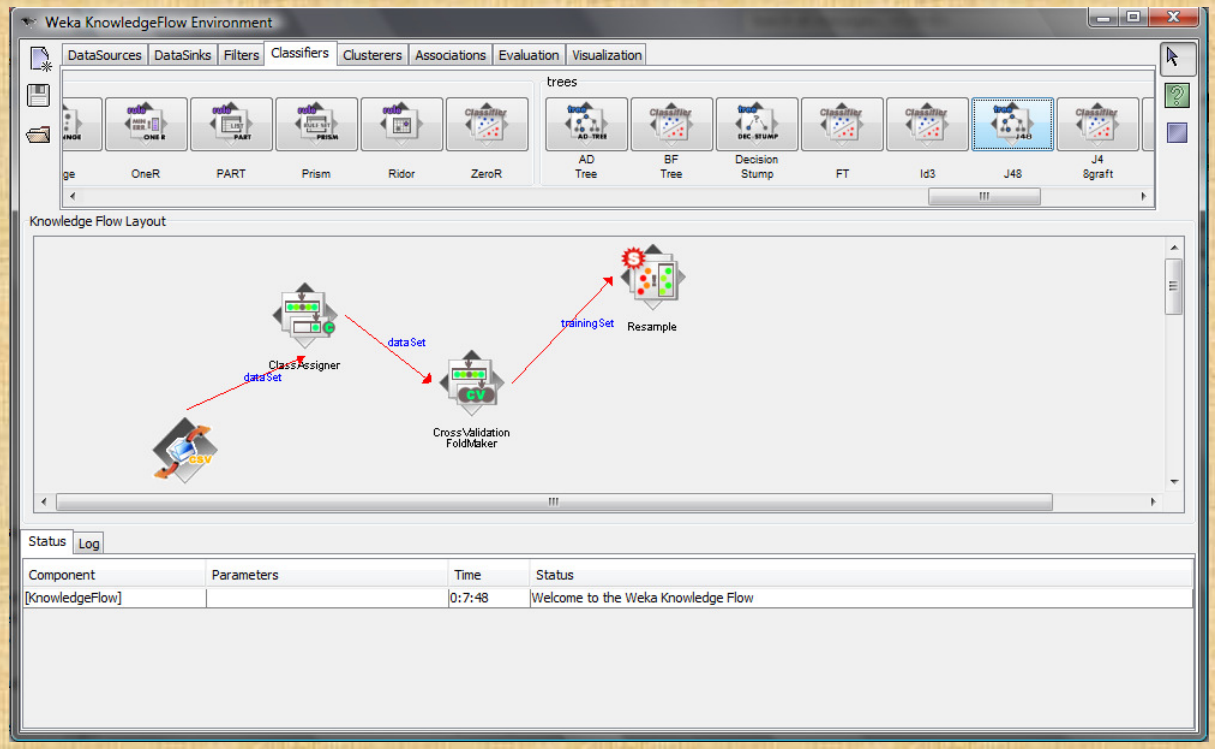
- Right-click CrossValidation FoldMaker to configure (By default, it is 10 folds)



4- Resample the training set to balance classes

- Select the Resample Filter from the Filters panel

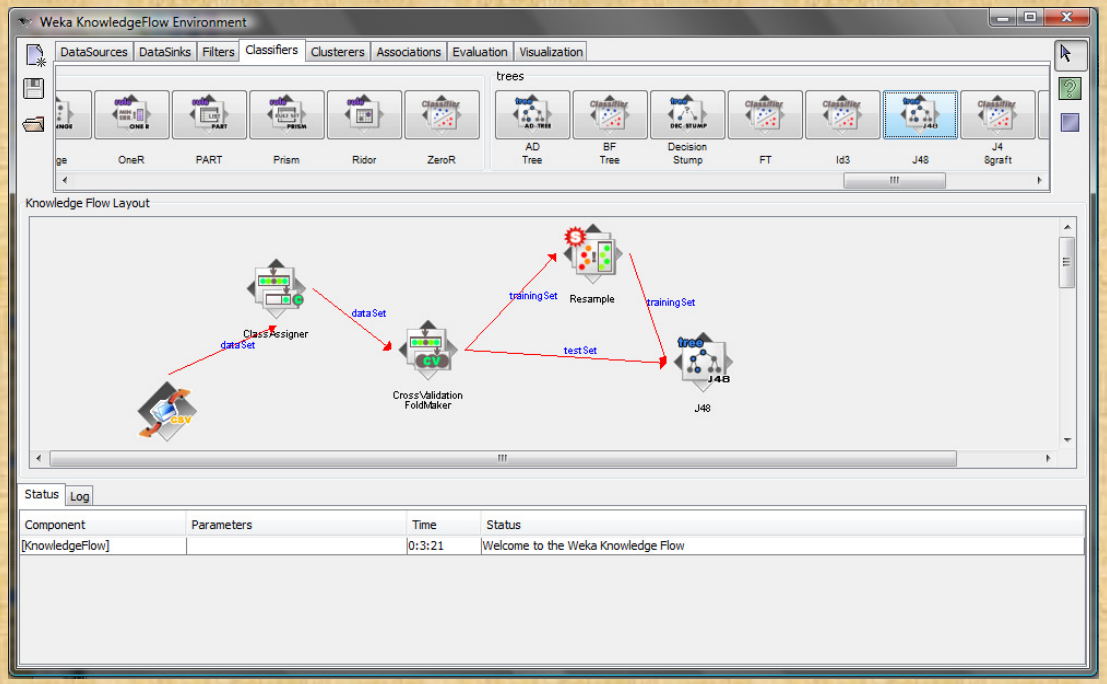
- Right-click the CVFoldMaker icon and select trainingSet to connect to Resample



5- Specify a classifier: J48,…

In this example we choose the decision tree algorithm

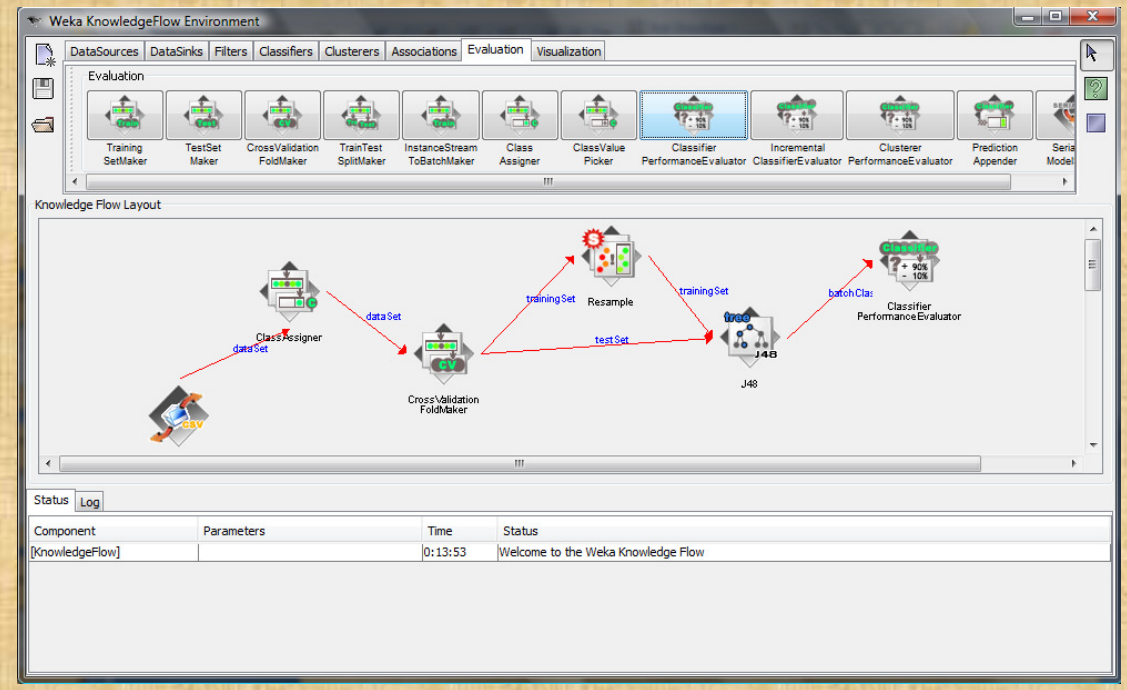
- Select J48 from the Classifiers panel
- Right-click the Resample icon and select trainingSet to connect to J48
- Right-click the CVFoldMaker icon and select testSet to connect to J48



6- Specify the Evaluation

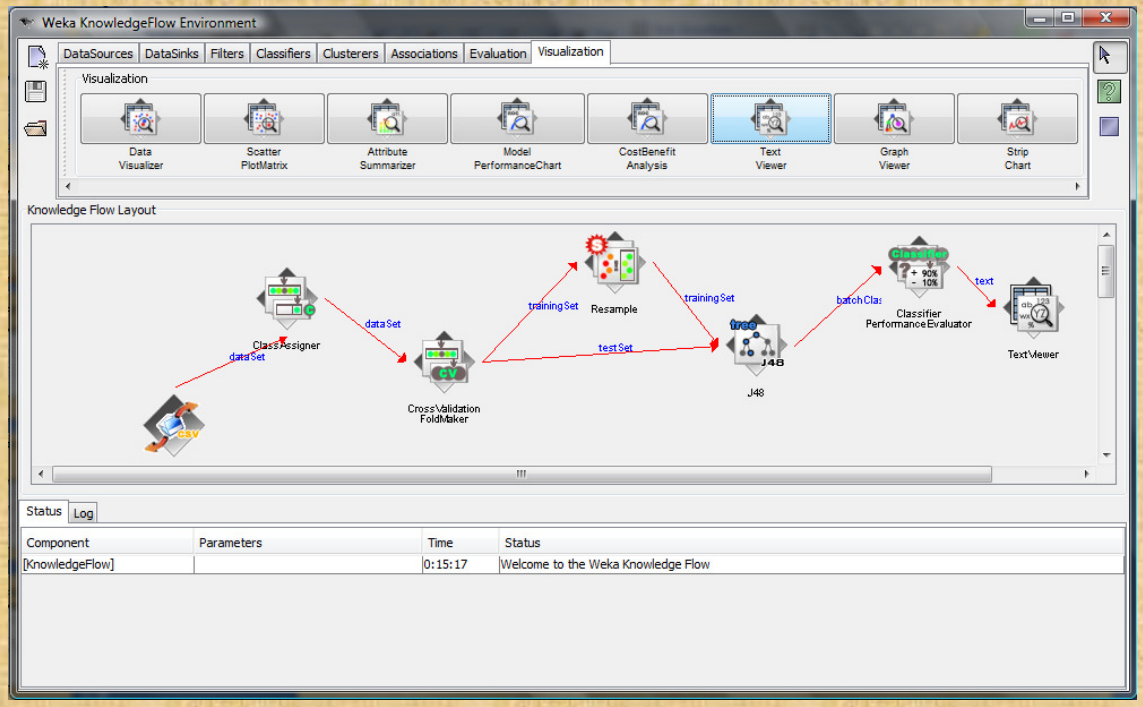
- Select ClassifierPerformance Evaluator from the Evaluation panel

- Right-click on the J48 icon and select batchClassifier to connect to the ClassifierPerformance Evaluator



7- Specify the Evaluation Output

- Select TextViewer from the Visualization panel
- Right-click the ClassifierPerformance Evaluator icon and select text to connect to TextViewer

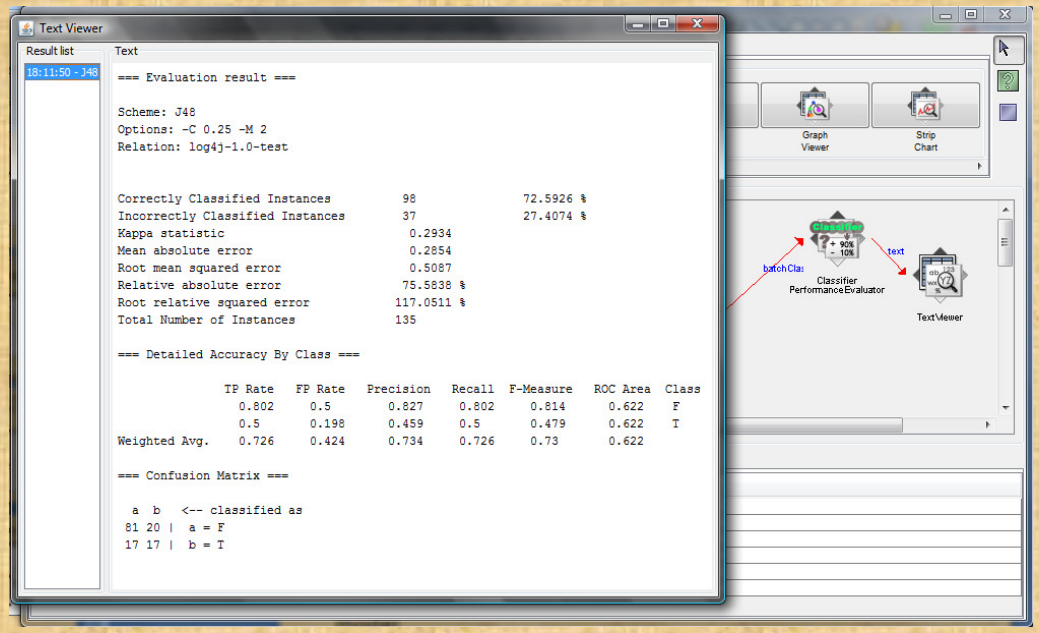


8- Running the Experiment

- Right-click on the CSVLoader and click configure to load the data

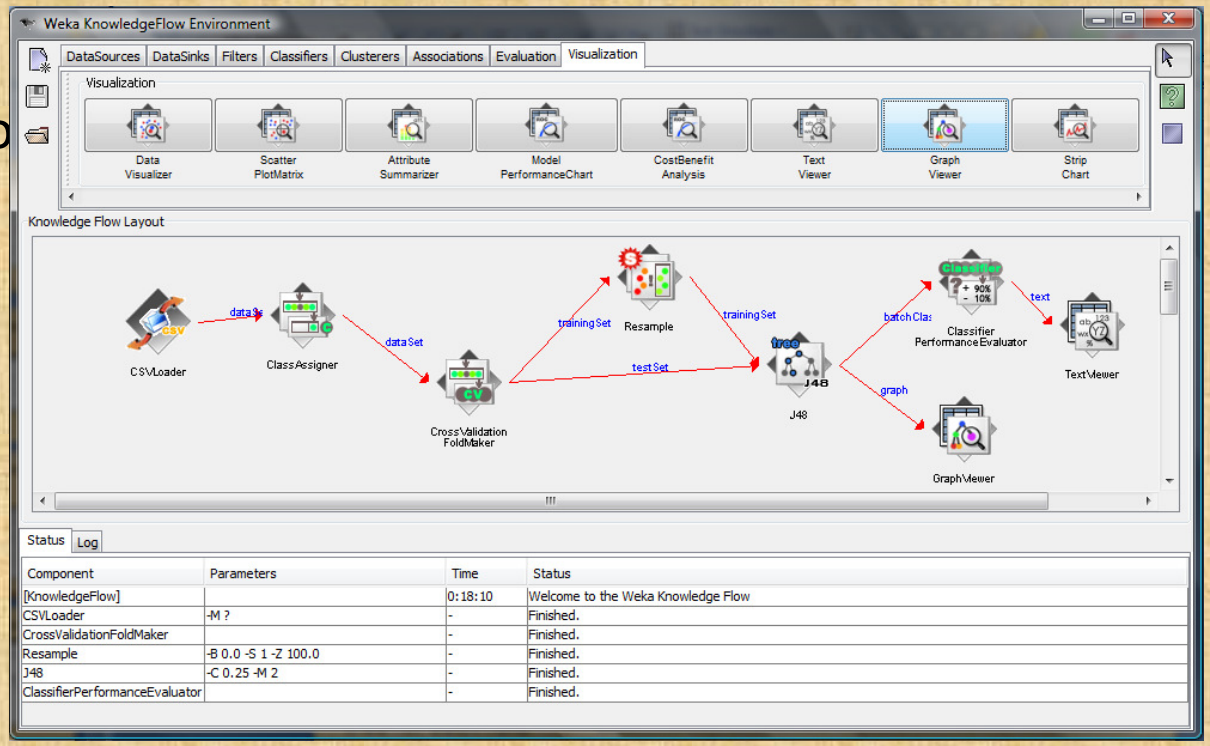
- Righ-click on the CSVLoader and select start loading

- Right-click on TextViewer and select Show results to view the results of your model

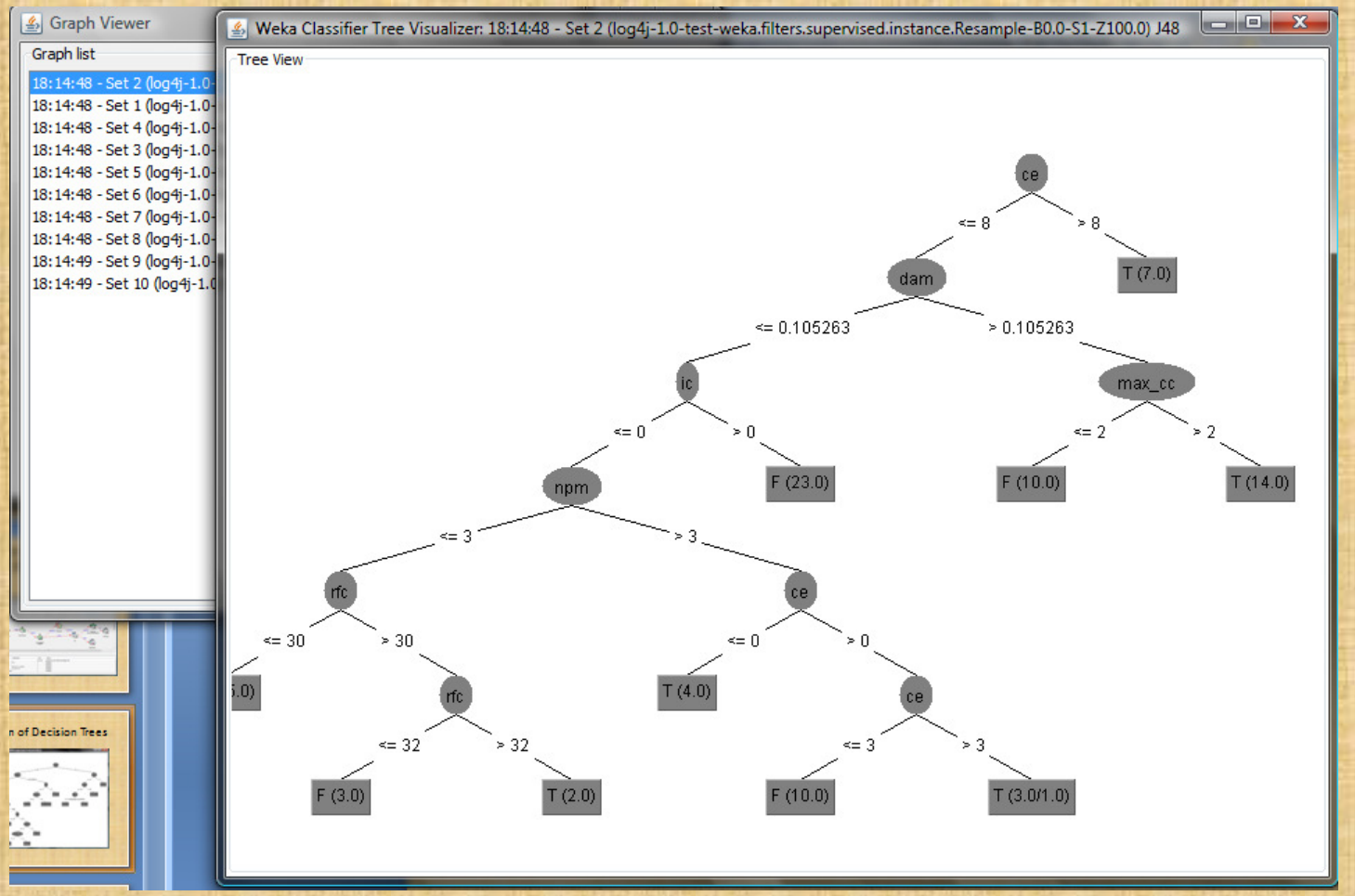


9- Allow the Viewing of Decision Trees per fold

- Select GraphViewer from the Visualization panel and
- Right-click the J48 icon and select graph to connect to the GraphViewer
- Right-click CSVloader icon and select Start Loading
- Right-click the GraphViewer icon to view decision tree for each fold



Visualization of Decision Trees



Model Saving

- All models built using the knowledge flow can be saved for future experiments.
- Now, save the model under the name treeModelCrossvalidation.kf (the save icon is at the extreme top left)

Exercise

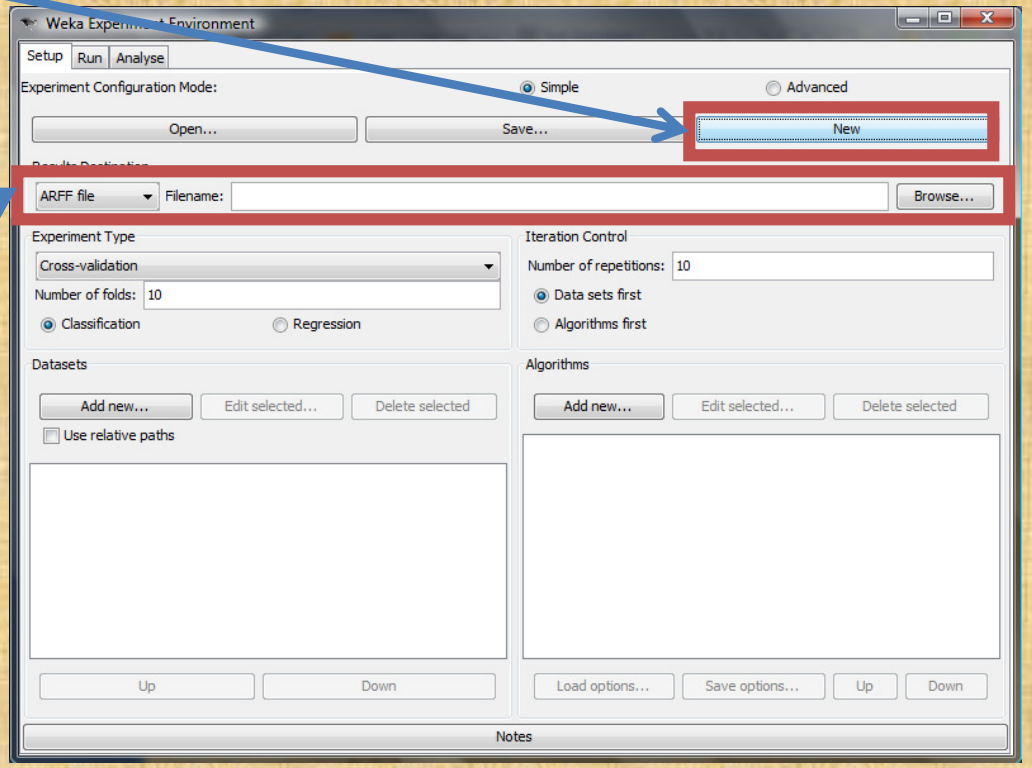
- Replace the J48 algorithm with the Random Forest algorithm and repeat the previous steps to evaluate your model.

Comparing Models

- We use Weka Experimenter to build and compare different models.

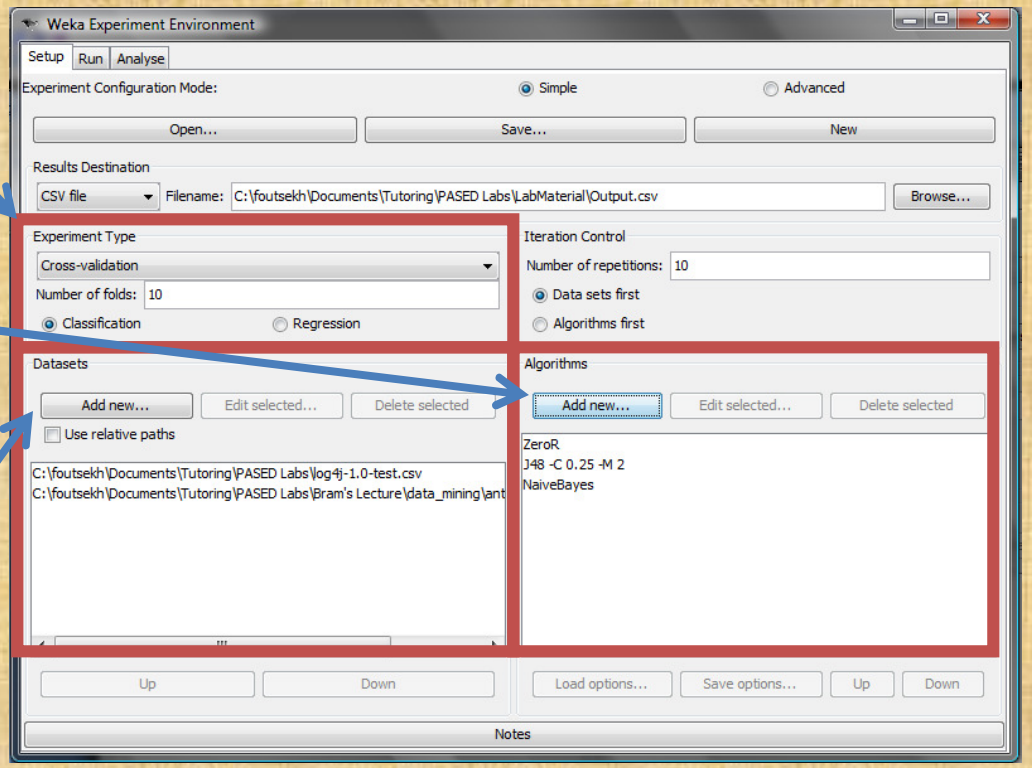
1- Defining an Experiment

- Select New to start a new experiment
- Next, specify the results destination file



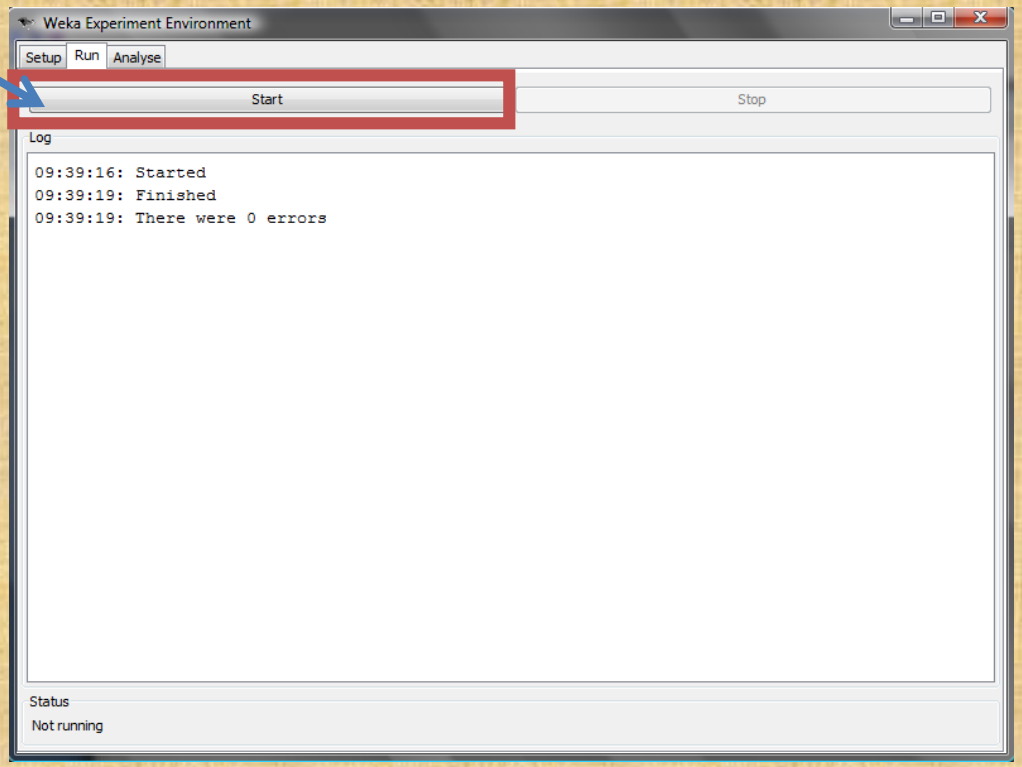
2- Setup the Experiment

- Specify the Experiment Type
- Add new classifiers to the experiment
- Add new datasets to the experiment (log4j1.0.csv, ant-1.7-pop-binary.arff)



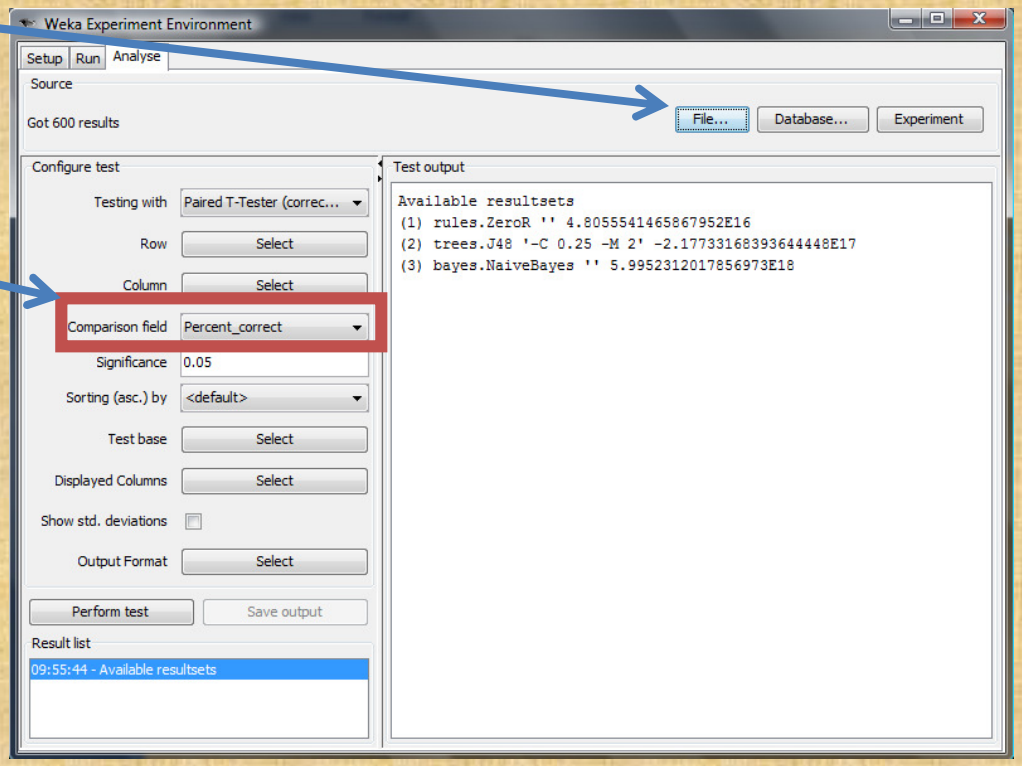
3- Run the Experiment

- Click on Start to run the experiment



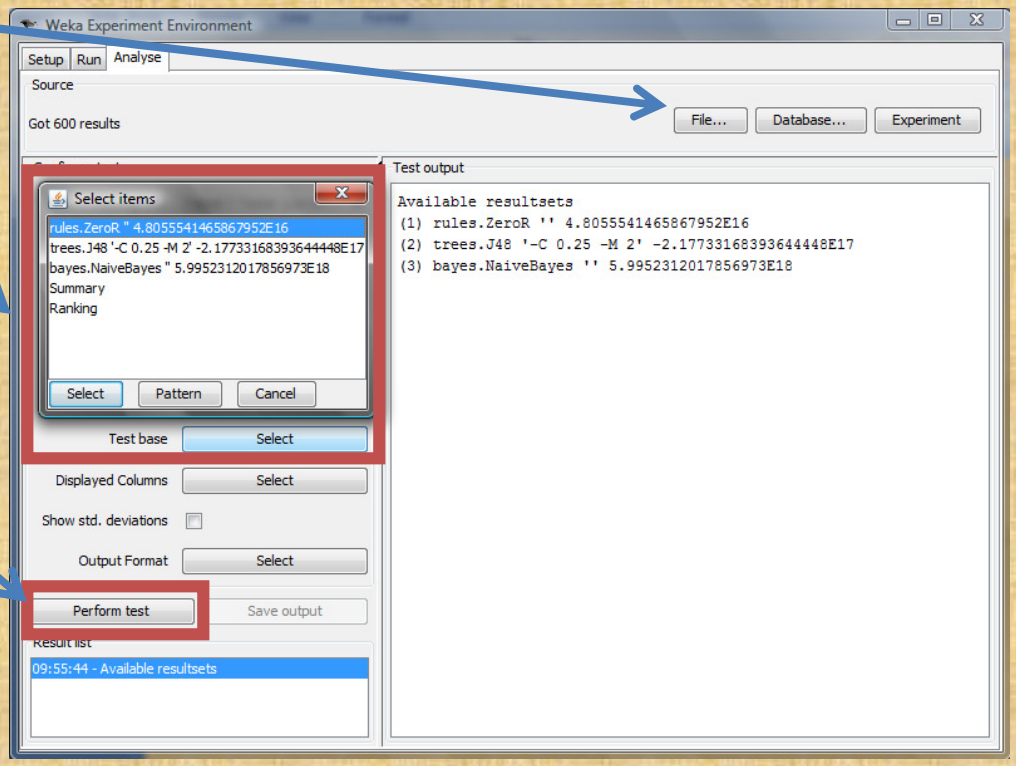
4- Analyze the Results (1)

- Load the Results file
- Select the comparison field



4- Analyze the Results (2)

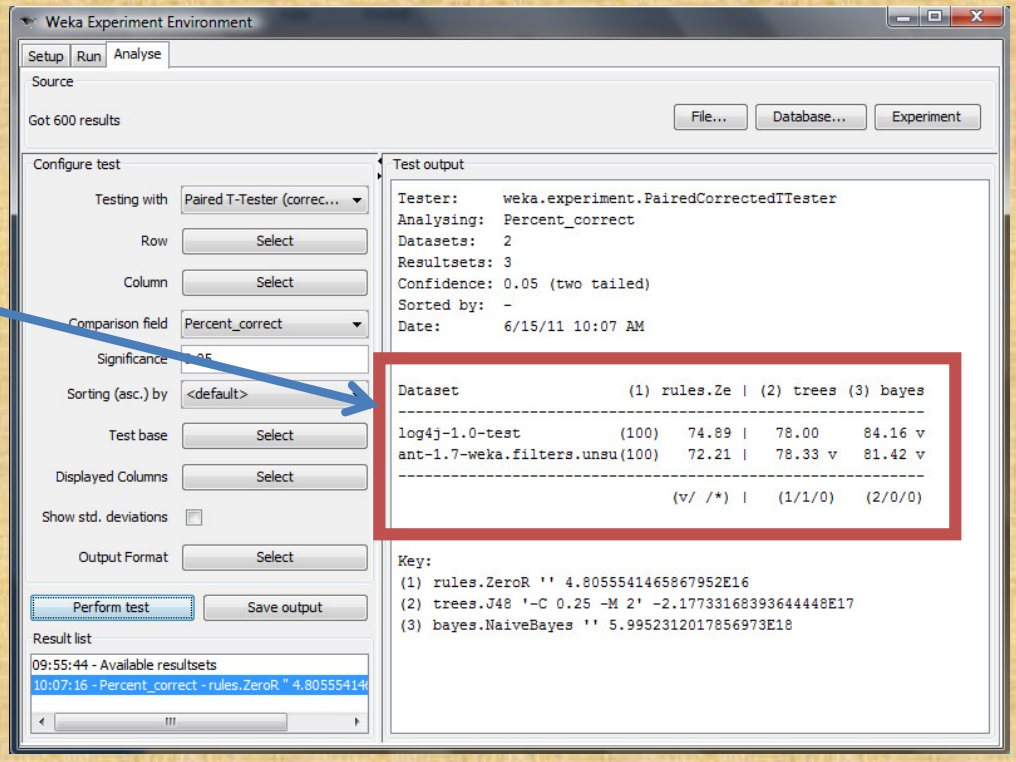
- Select the base model to test upon
- Perform the test



5- Interpretation of Results (1)

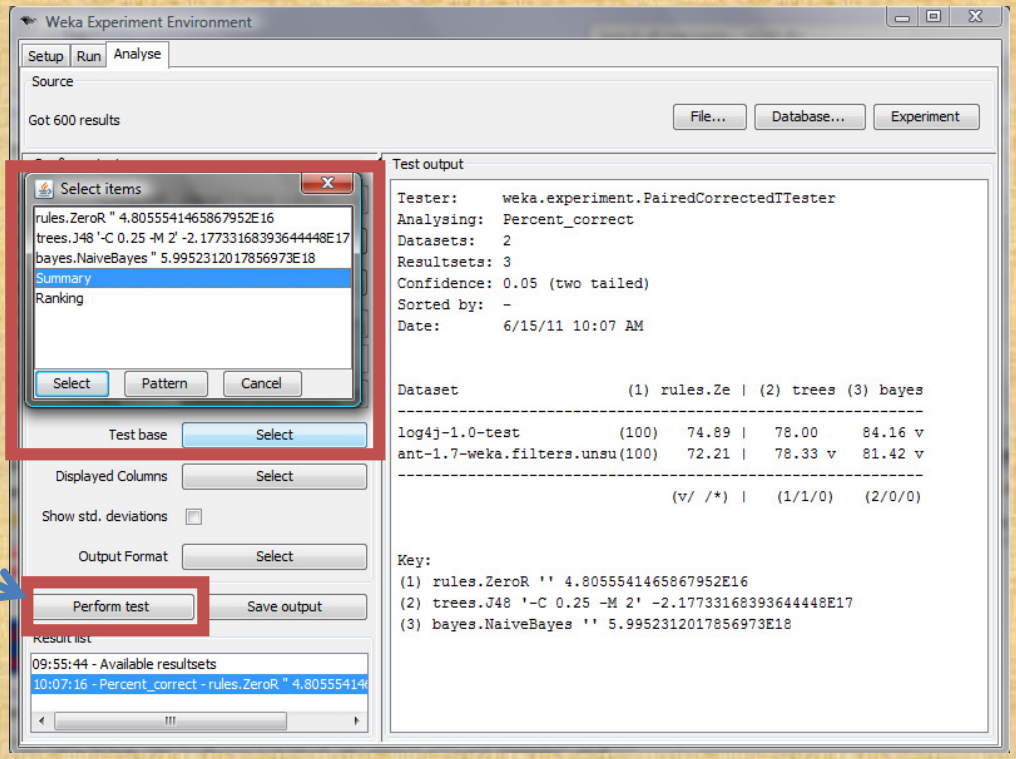
- Results show that on the log4j-1.0 dataset, Naïve Bayes is significantly better than ZeroR

- While on the Ant-1.7 dataset, both J48 and Naïve Bayes are significantly better than ZeroR



5- Interpretation of Results (2)

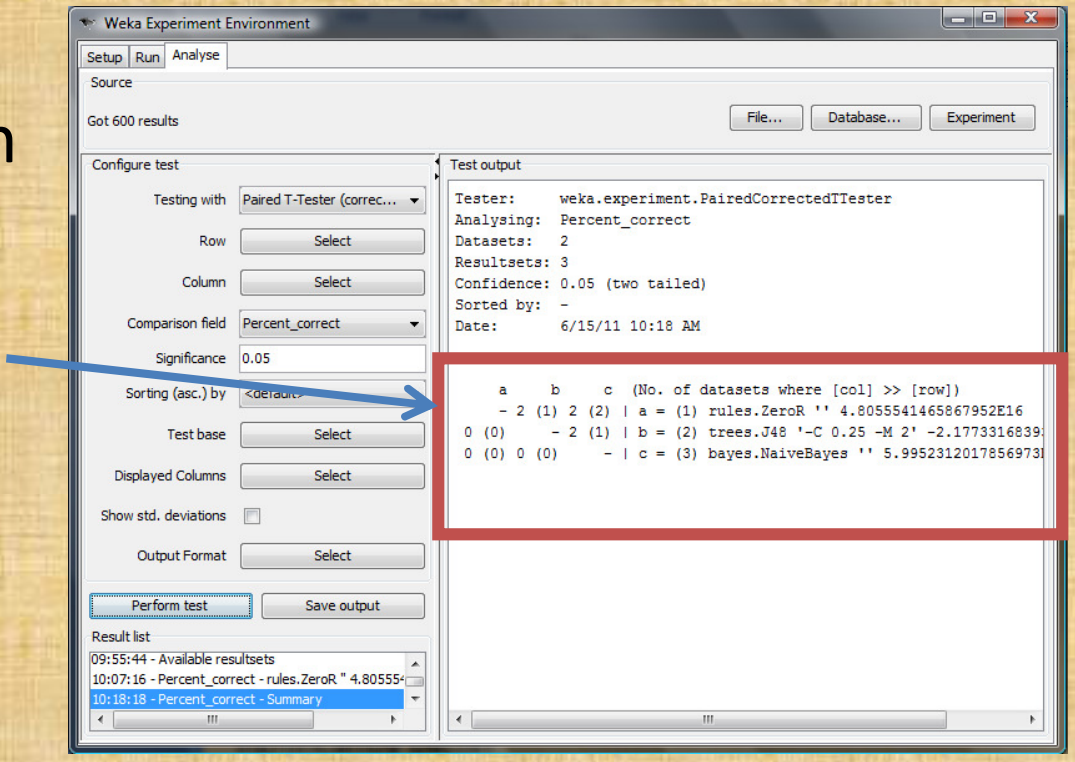
- Lets select Summary in the Test base menu
- Perform the test



5- Interpretation of Results (3)

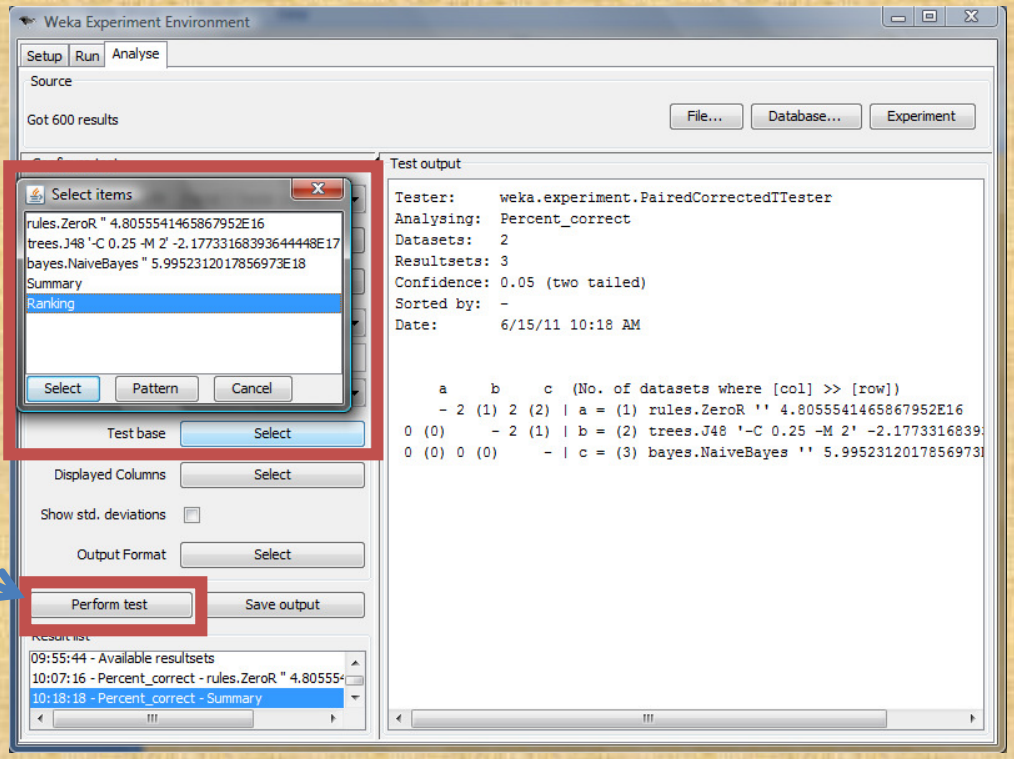
Results show that:

- J48 is significantly better than ZeroR on one dataset, while
- Naïve Bayes is significantly better than ZeroR on two datasets



5- Interpretation of Results (4)

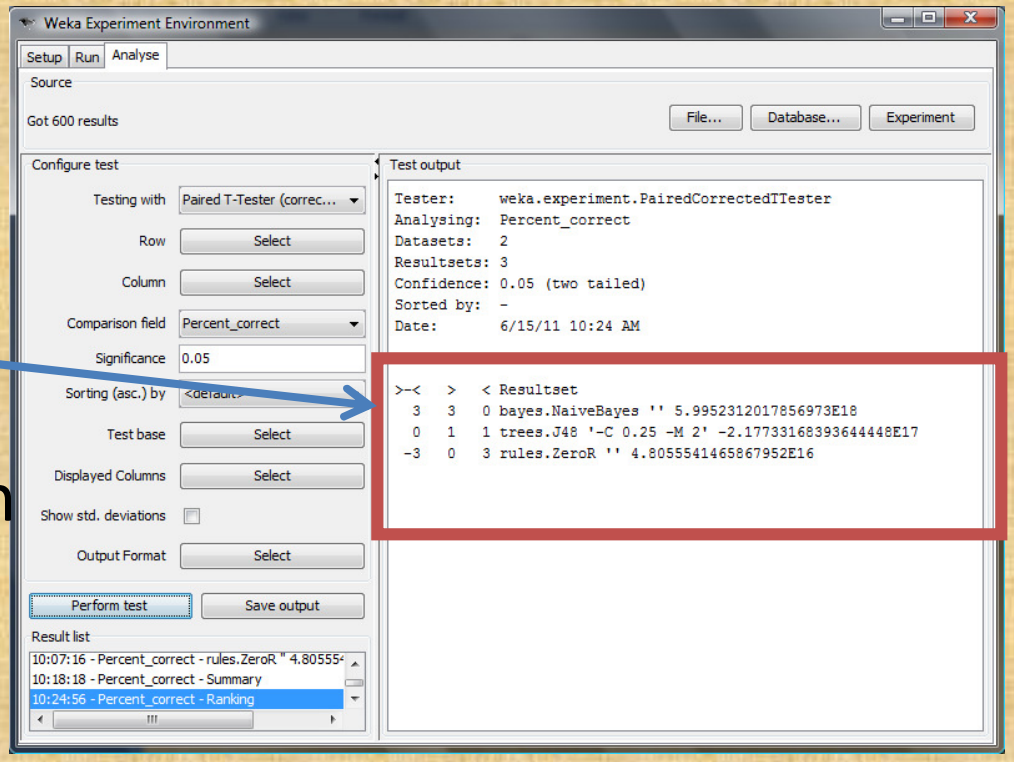
- Let's select Ranking in the Test base menu
- Perform the test



5- Interpretation of Results (5)

- Models are ranked based on their numbers of total wins (“>”) and total losses (“<“)

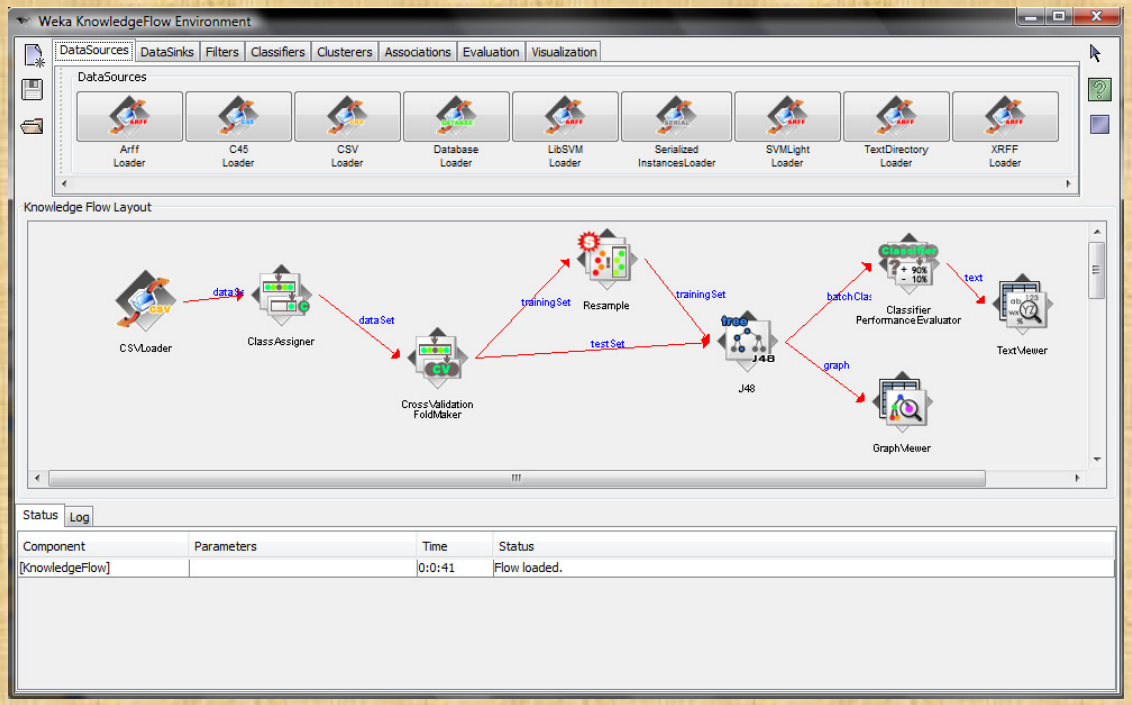
- The column (“>-<”) is the difference between the number of wins and the number of losses.



Multinomial classification Models

1- Loading a Model

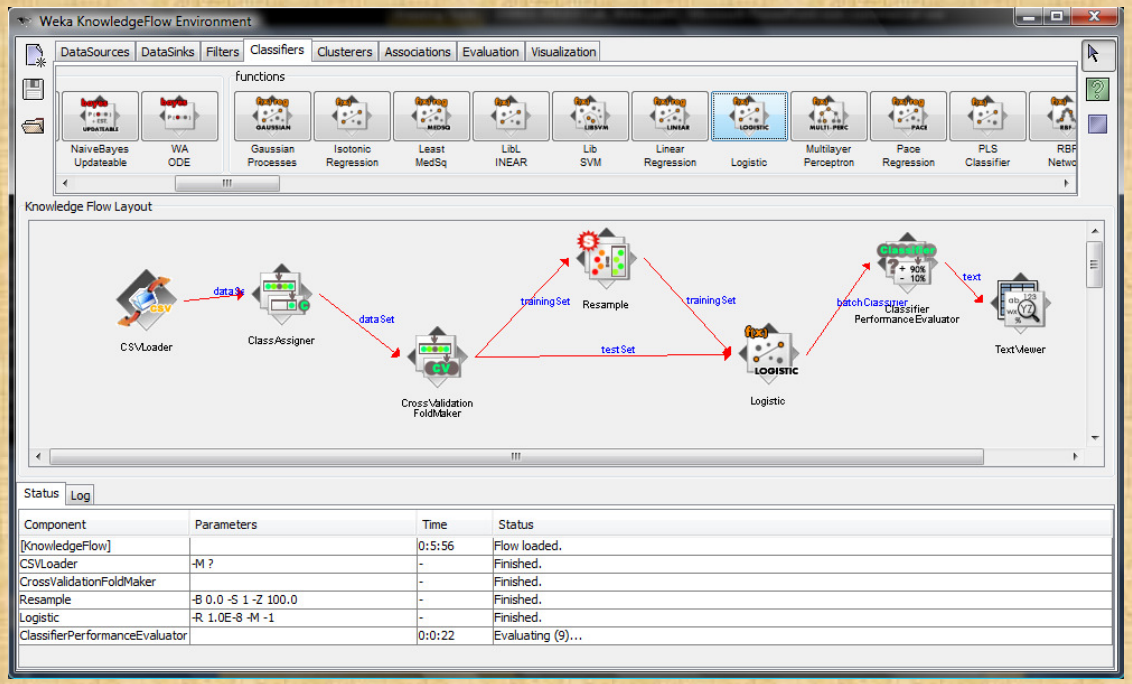
- Open the Knowledge Flow and load the treeModelCrossvalidation.kf saved on slide 18.



2- Specify the Logistic Classifier

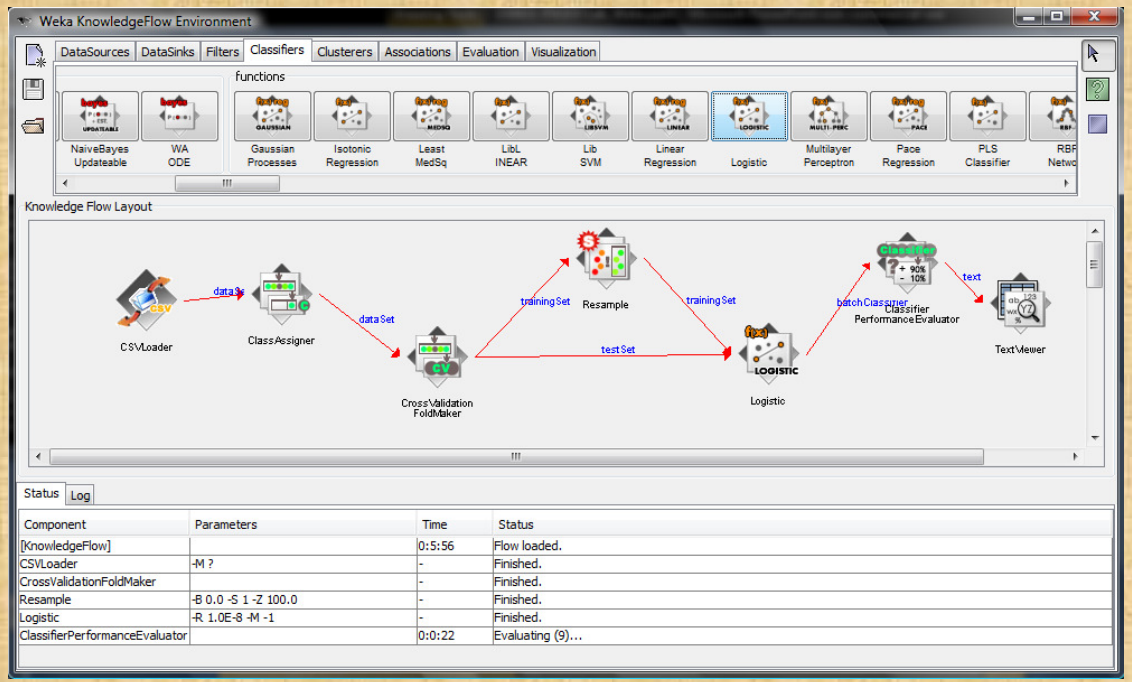
- Replace the J48 classifier by the Logistic classifier

[delete J48, add Logistic, rigth-click on Resample and connect it (trainingSet) to Logistic, right-click on CrossValidationFoldMaker and connect it (testSet) to Logistic, right-click on Logistic and connect it (batchClassifier) to ClassifierPerformanceEvaluator, and delete GraphViewer]

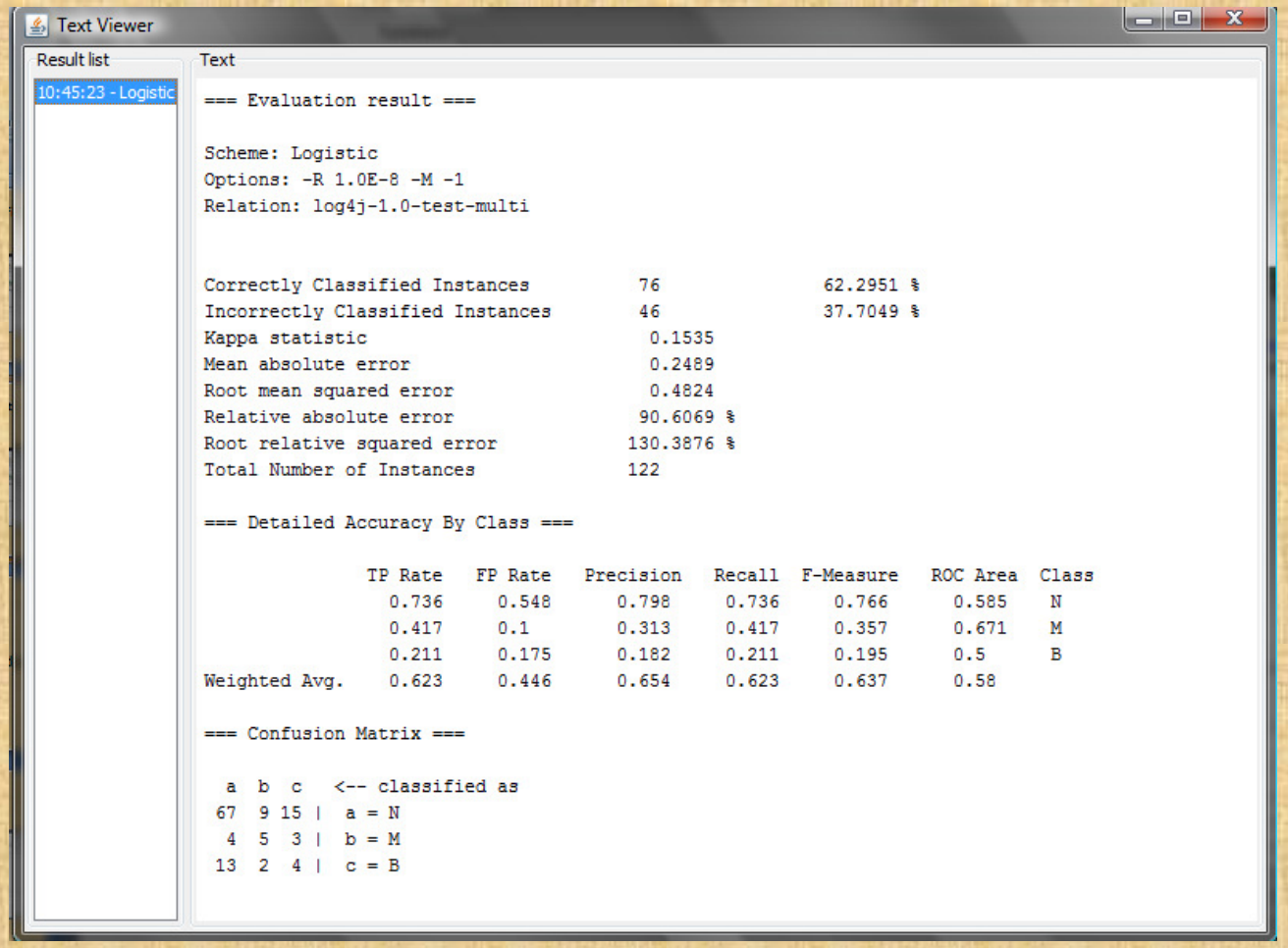


3- Running the Experiment

- Right-click on the CSVLoader and click configure to load the file log4j-1.0-test-multi
- Righ-click on the CSVLoader and select start loading
- Right-click on TextViewer and select Show results to view the results of your model



Results


Text Viewer
Result list
10:45:23 - Logistic
Text
=== Evaluation result ===
Scheme: Logistic
Options: -R 1.0E-8 -M -1
Relation: log4j-1.0-test-multi
Correctly Classified Instances 76 62.2951 %
Incorrectly Classified Instances 46 37.7049 %
Kappa statistic 0.1535
Mean absolute error 0.2489
Root mean squared error 0.4824
Relative absolute error 90.6069 %
Root relative squared error 130.3876 %
Total Number of Instances 122
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.736 0.548 0.798 0.736 0.766 0.585 N
0.417 0.1 0.313 0.417 0.357 0.671 M
0.211 0.175 0.182 0.211 0.195 0.5 B
Weighted Avg. 0.623 0.446 0.654 0.623 0.637 0.58
=== Confusion Matrix ===
a b c <-- classified as
67 9 15 | a = N
4 5 3 | b = M
13 2 4 | c = B

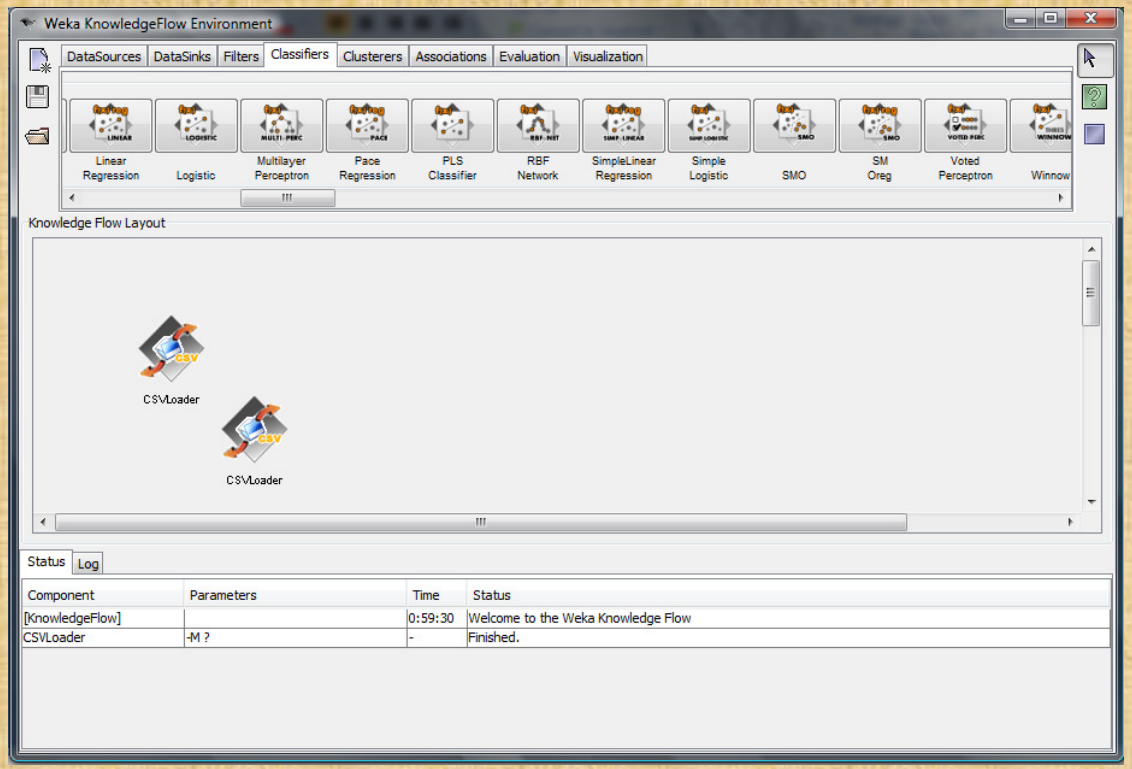
Training and Testing Models on different sets

1- Specify Data Sources

- Select two CSVloader

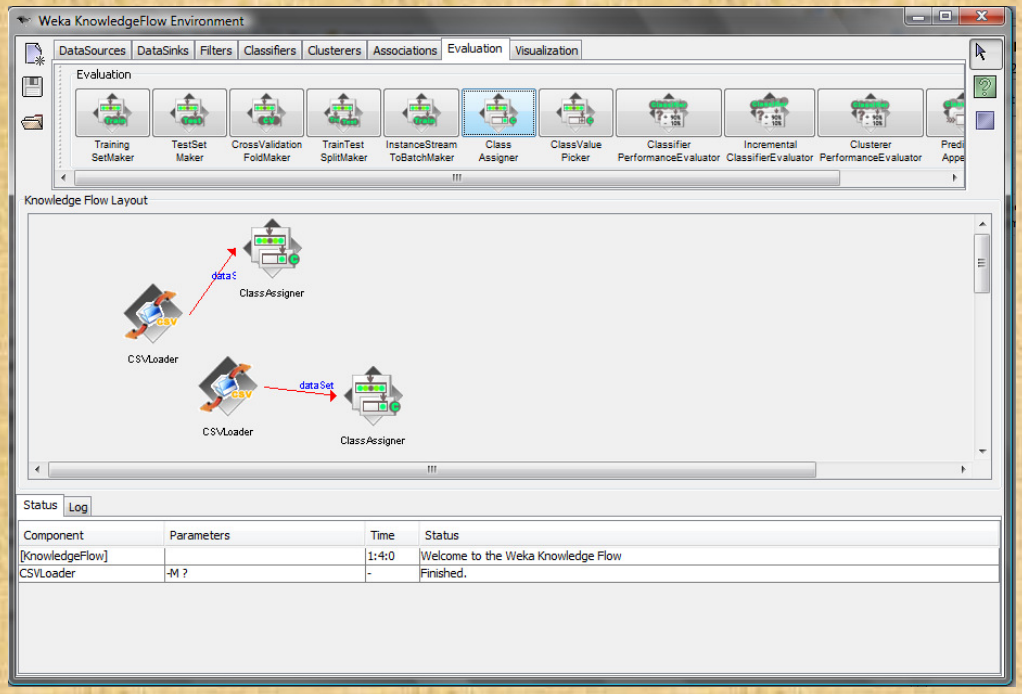
- Right-click the first CSVloader icon to configure by specifying the location of the training dataset

- Right-click the second CSVloader icon to configure by specifying the location of the testing dataset



2- Specify Class Attribute

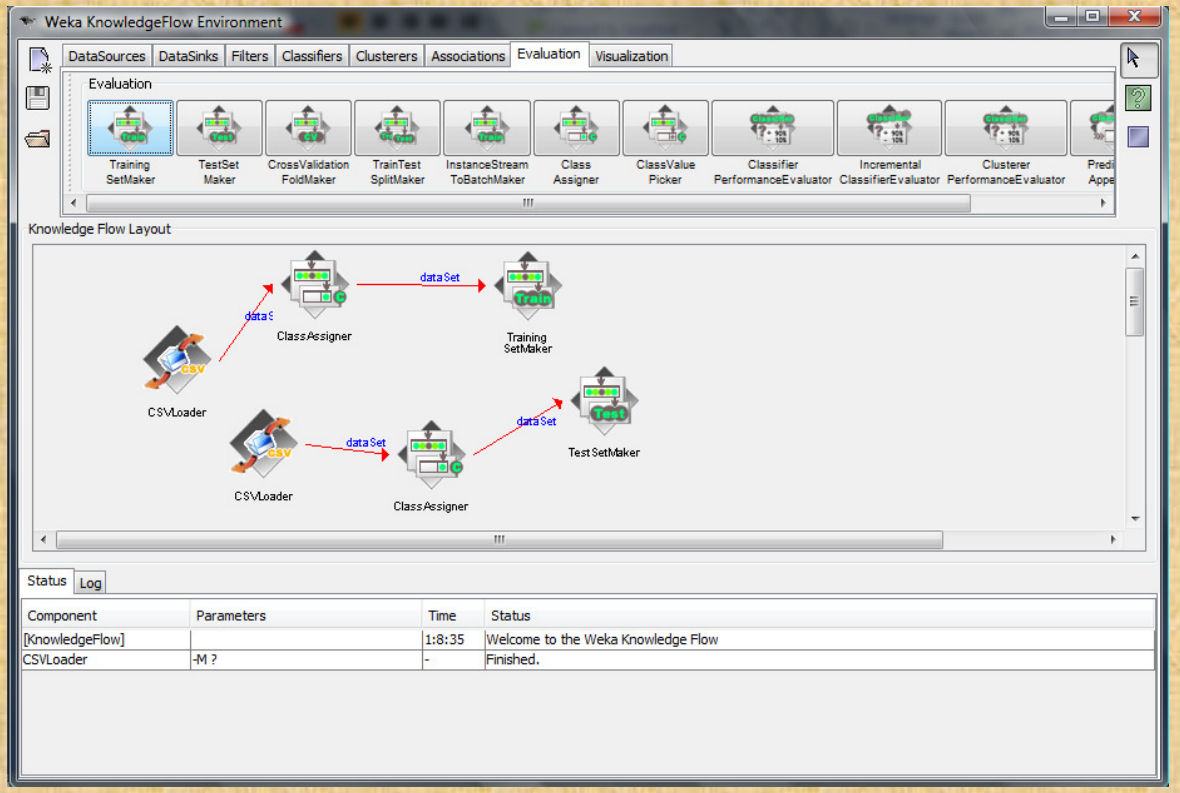
- Select two ClassAssigners from the Evaluation panel
- Right-click on each of the CSVloader icon and select dataset to connect to the ClassAssigners
- Right-click on each ClassAssigner to configure the target class



3- Specify Training and Testing sets

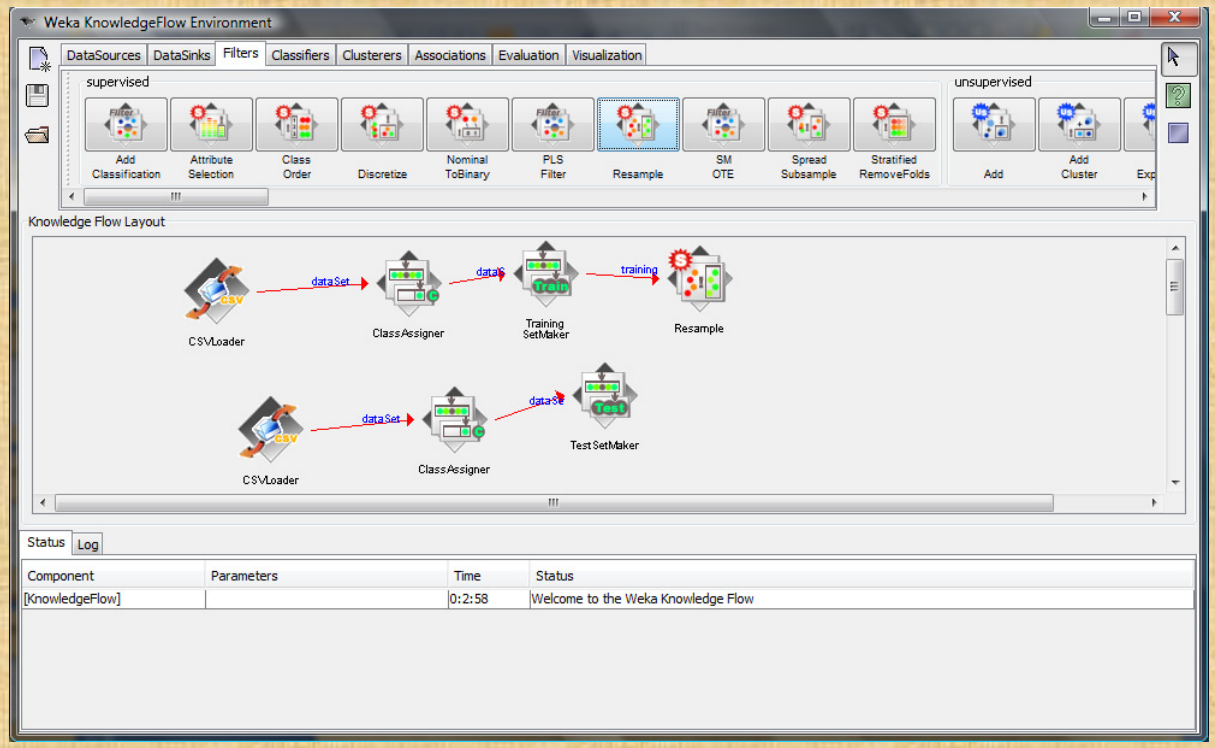
- Select Training SetMaker and Test SetMaker from the Evaluation panel

- Right-click on each of the ClassAssigner icon and select dataset to connect respectively to the Training SetMaker and the TestSetMaker



4- Resample the training set to balance classes

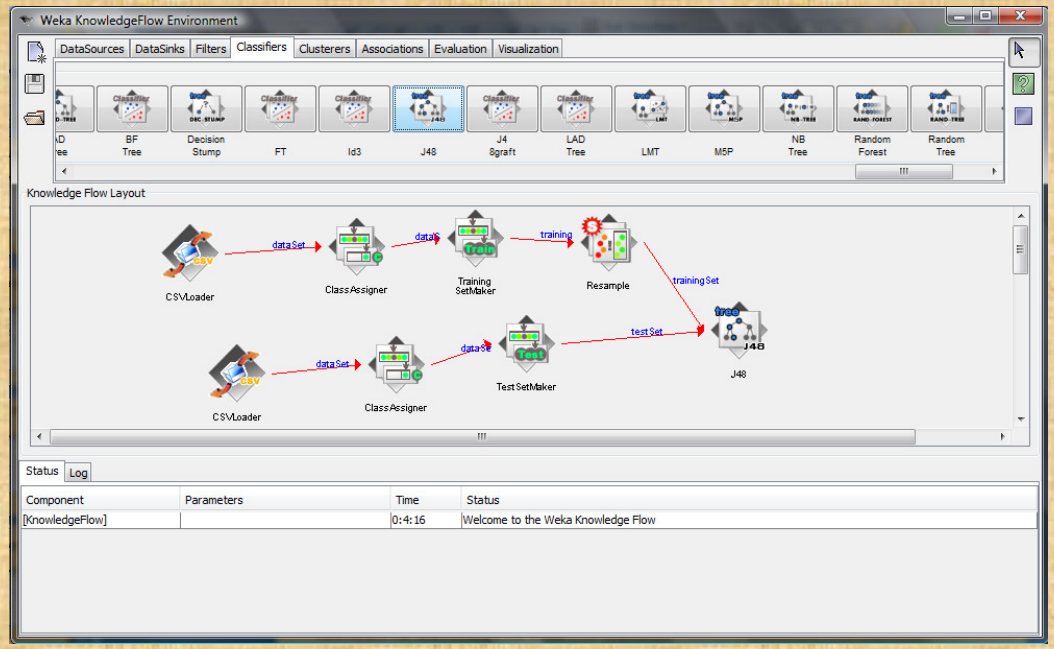
- Select the Resample Filter from the Filters panel
- Right-click the Training SetMaker icon and select trainingSet to connect to Resample



5- Specify a Classifier

In this example we choose the decision tree algorithm again

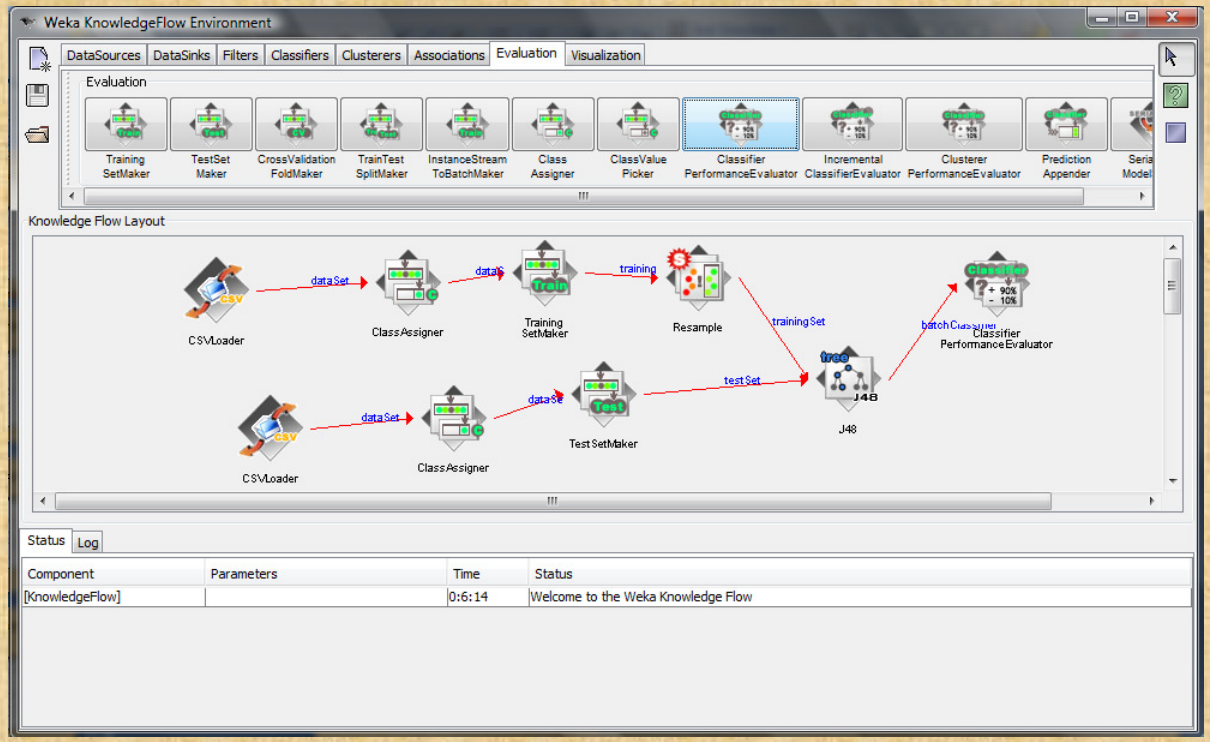
- Select J48 from the Classifiers panel
- Right-click the Resample icon and select trainingSet to connect to J48
- Right-click the Test SetMaker icon and select testSet to connect to J48



6- Specify the Evaluation

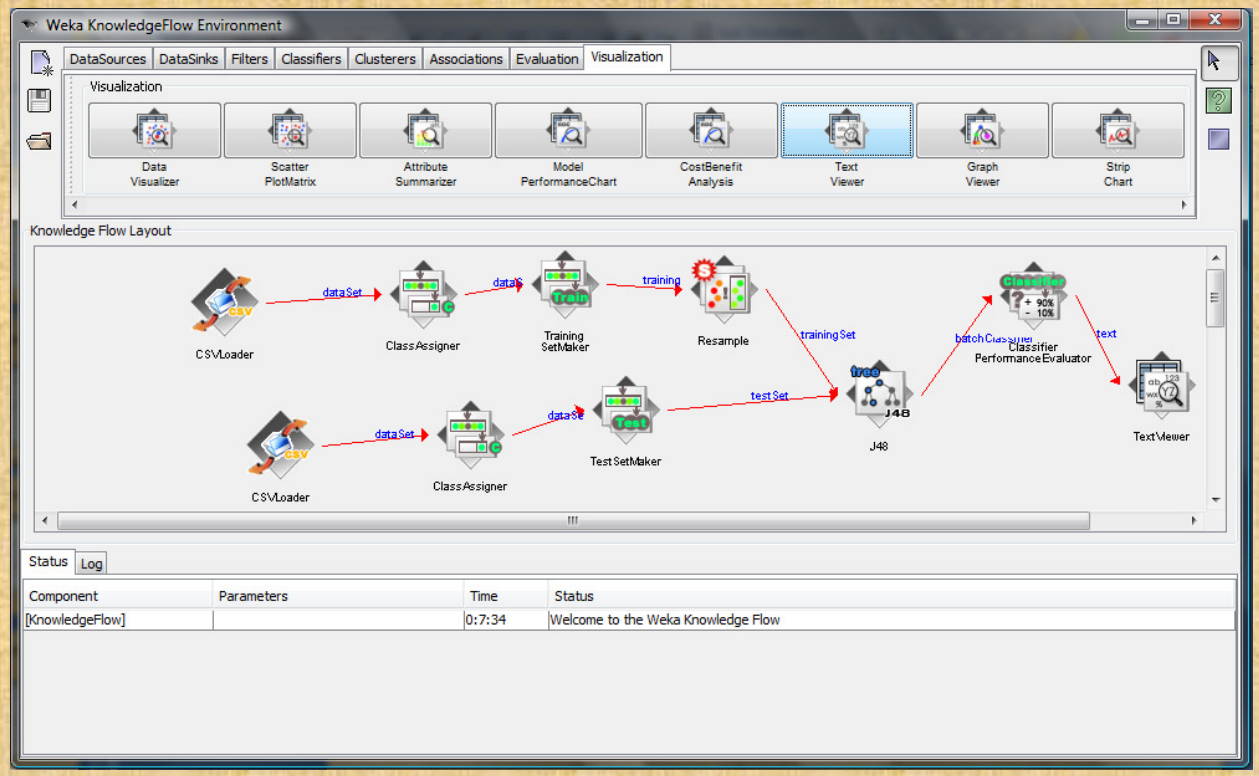
- Select ClassifierPerformance Evaluator from the Evaluation panel

- Right-click on the J48 icon and select batchClassifier to connect to the ClassifierPerformance Evaluator



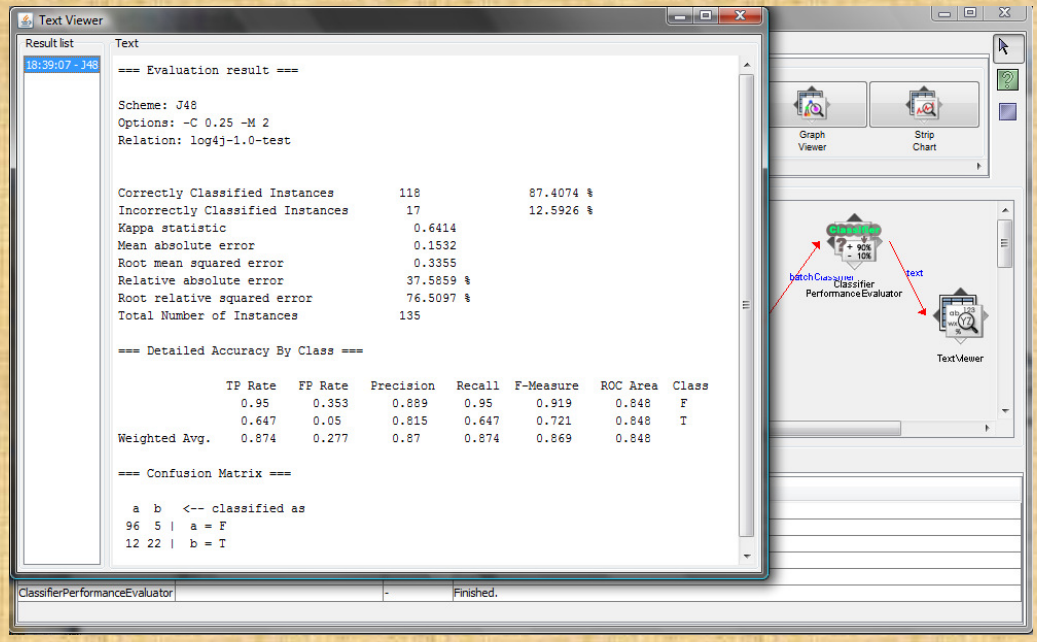
7- Specify the Evaluation Output

- Select TextViewer from the Visualization panel
- Right-click the ClassifierPerformance Evaluator icon and select text to connect to TextViewer



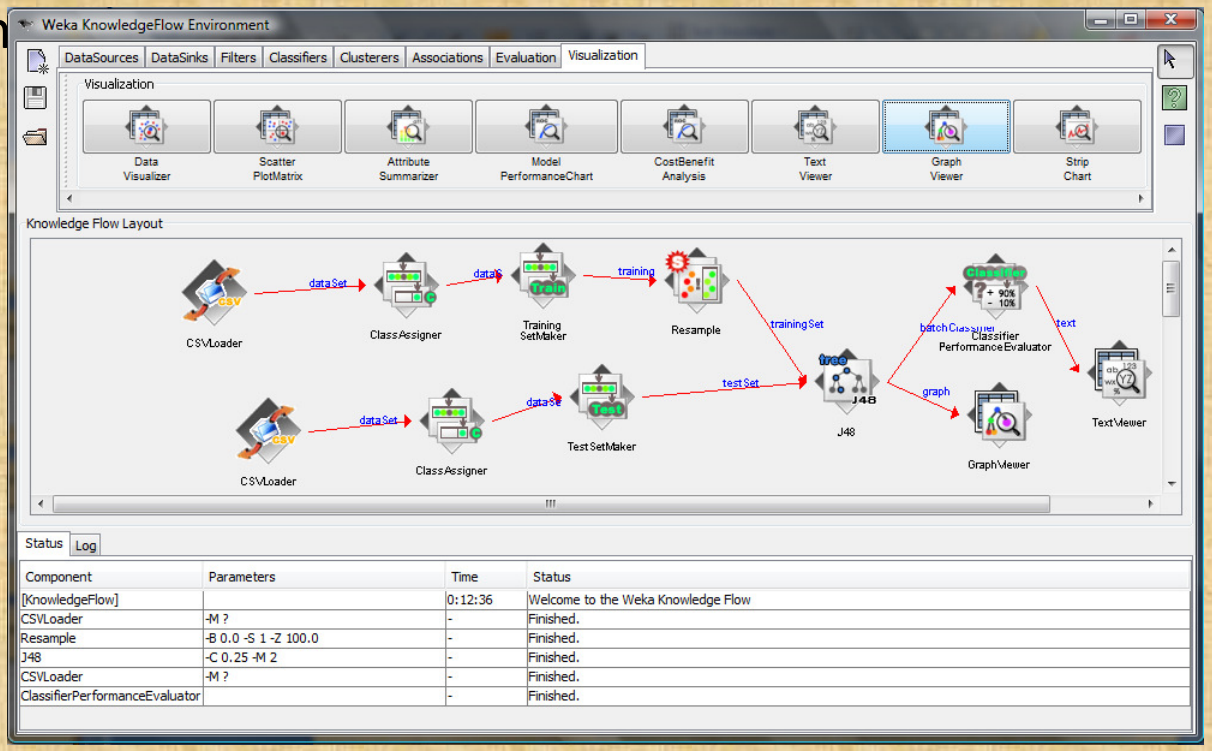
8- Running the Experiment

- Right-click on the training CSVLoader and click configure to load the training data
- Right-click on the testing CSVLoader and click configure to load the testing data
- Right-click on each CSVLoader and select start loading
- Right-click on TextViewer and select Show results to view the results of your model

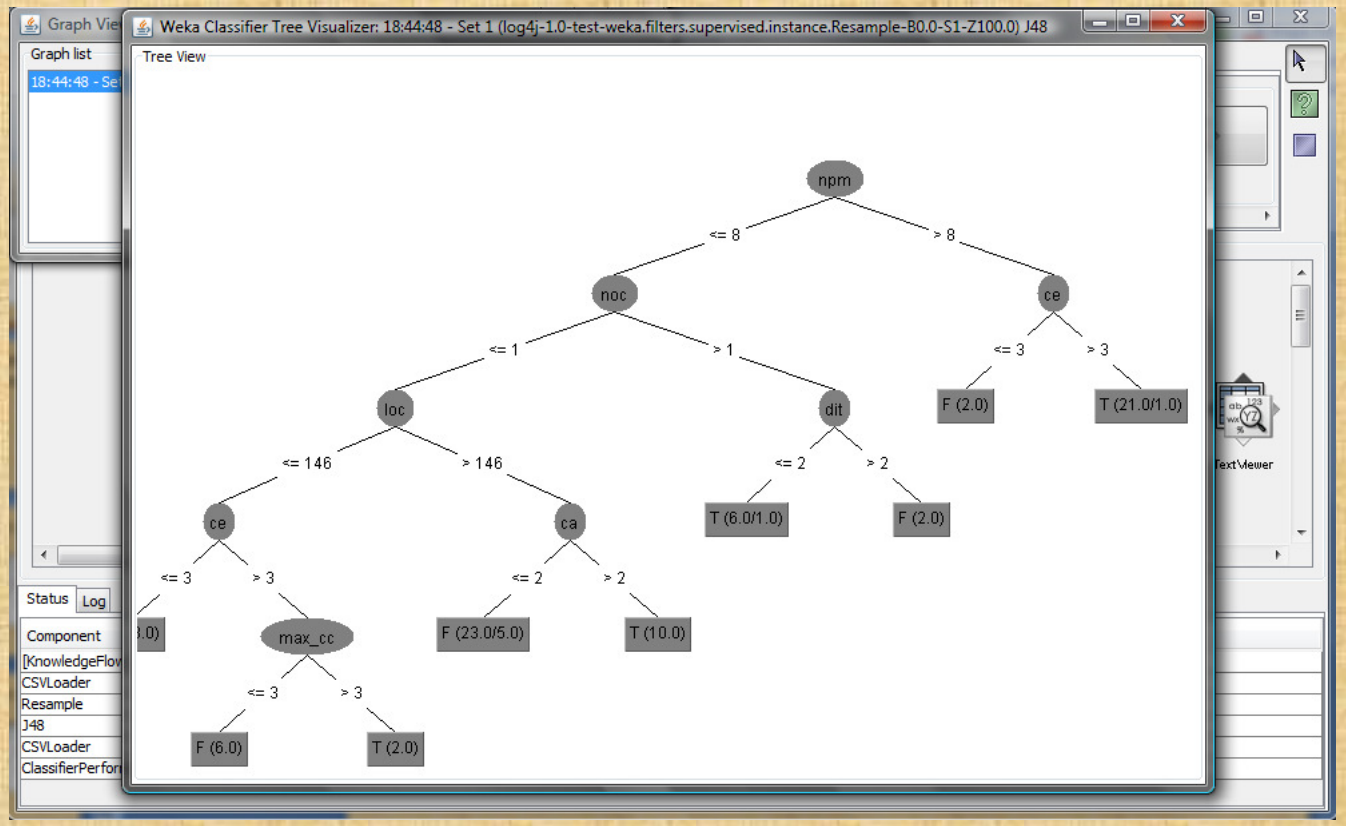


9- Viewing of Decision Tree

- Select GraphViewer from the Visualization panel and
- Right-click the J48 icon and select graph to connect to the GraphViewer
- Right-click on each CSVloader icon and select Start Loading
- Right-click the GraphViewer icon to view the decision tree



Visualization of Tree



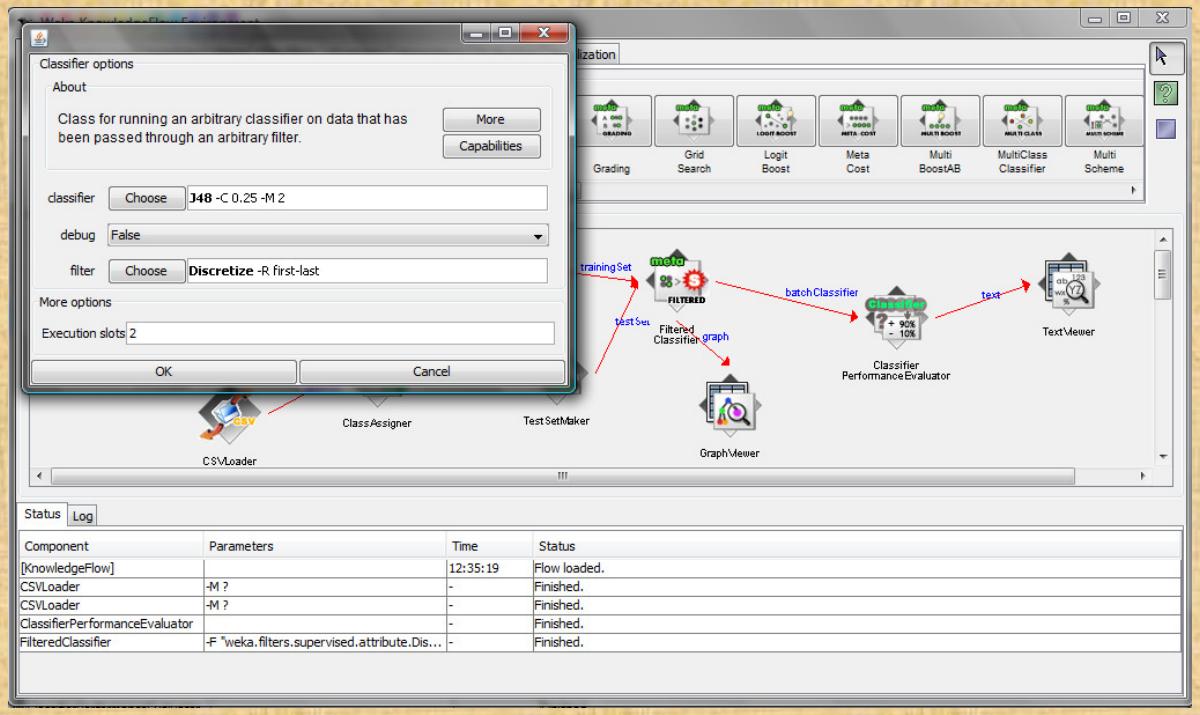
Saving the model

- Save the model under the name treeModeltesting.kf

10- Discretization (1)

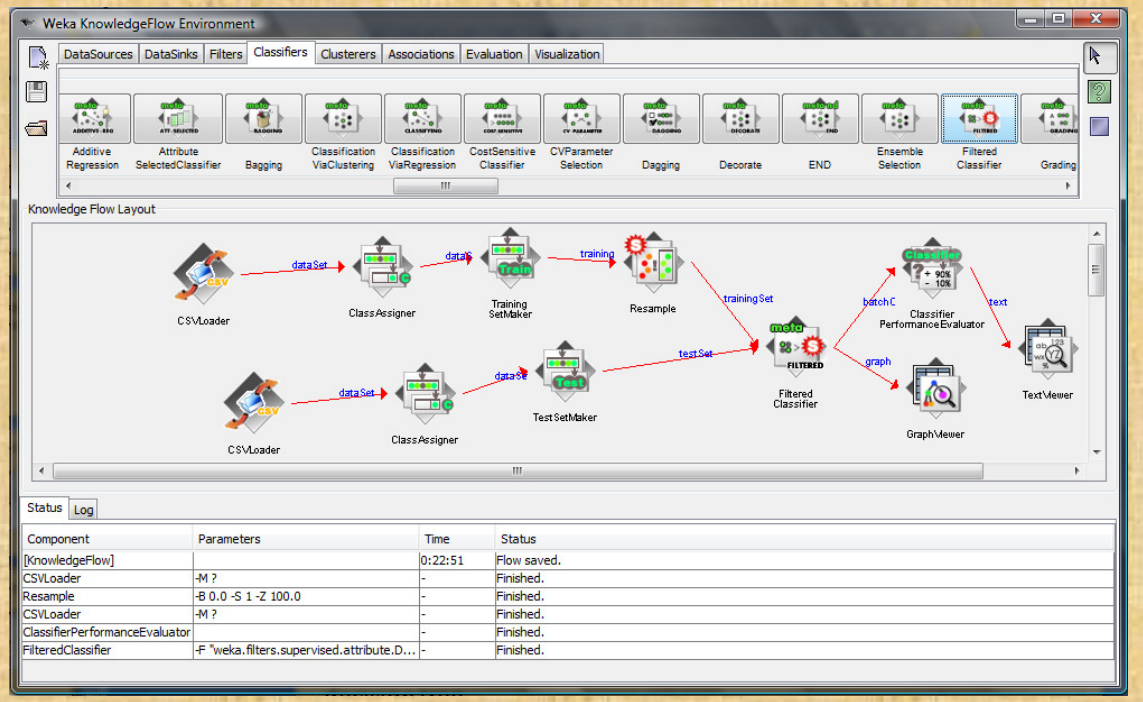
Discretization can help build more effective models in terms of performance and size

- Replace the J48 classifier by a FilteredClassifier that encapsulates discretization and the J48 classifier

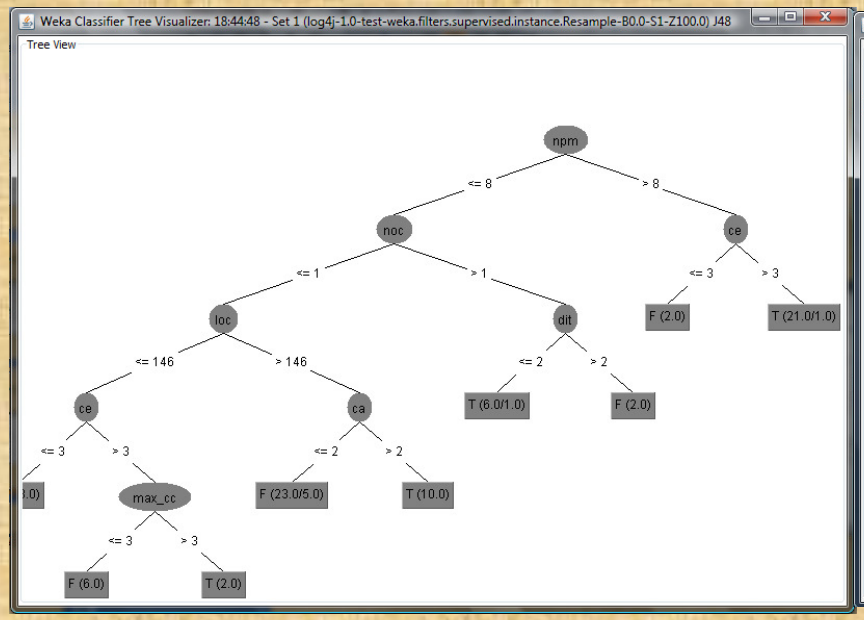


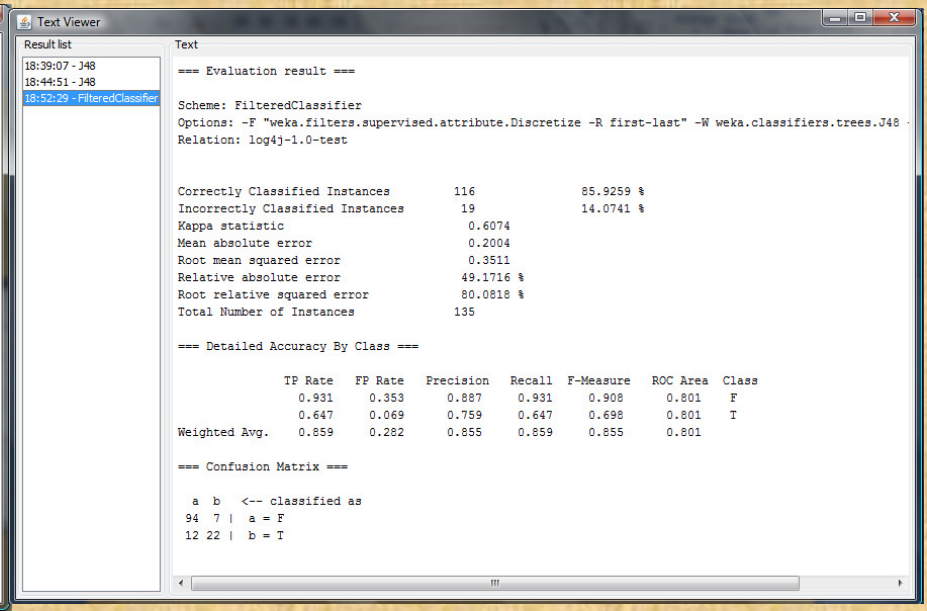
10- Discretization (2)

- Right-click on each CSVloader icon and select Start Loading
- Right-click on TextViewer and select Show results to view the results of the model
- Right-click the GraphViewer icon to view the decision tree



10- Discretization (3)





- Save the new model under the name treeModeltesting_discretized.kf

Exercise

- Select the Naïve Bayes classifier to build and test a model on data from the Lab package.