

Knowledge Flow creation using WEKA

Using Weka's knowledge Flow module in a classification problem

Elisa Maria e Castro Rocha Duarte

1/29/2010

Contents

Contents	1
Table of figures	2
Introduction	3
<i>Knowledge Flow</i> interface presentation	4
<i>Knowledge Flow</i> utilization example	8
Credit data base	8
Results	11
Linear Regression:	11
BayesNet:	12
J48:	13
MultilayerPreceptron:	14
Logistic:	15
Interpreting the results	16
Incremental Learning	17
Conclusions	18
Bibliography	19

Table of figures

FIG. 1 WEKA GUI CHOOSER	4
FIG. 2 WEKA KNOWLEDGE FLOW INTERFACE	5
FIG. 3 WEKA KNOWLEDGE FLOW	9
FIG. 4 WEKA ATTRIBUTE SUMMARIZER	10
FIG. 5 TEXT VIEW LINEAR REGRESSION	11
FIG. 6 TEXT VIEW BAYES NET	12
FIG. 7 TEXT VIEW J48	13
FIG. 8 TEXT VIEW MULTILAYER	14
FIG. 9 TEXT VIEW LOGISTIC	15
FIG. 10 MODEL PERFORMANCE CHART	16
FIG. 11 INCREMENTAL STRIP CHART	17

Introduction

The Weka workbench provides the data mining practitioner with a set of machine learning algorithms, allowing data mining on data sets with different parameters. Being a data mining tool, one can use methods like attribute selection, association rules, regression, classification and clustering on a set of data with the following objectives:

- Output analysis to gather knowledge on the data.
- Prediction about new instances.
- Evaluation on knowledge methods to choose the one with the best prediction performance.

The Weka machine learning workbench has three separate interactive interfaces. The *Explorer* gives access to Weka's facilities using menu selection and form filling. The *Knowledge Flow* interface allows the design of configurations for processing data streams. With the *Experimenter*, it is possible set up automated experiments that run selected machine learning algorithms with different parameter settings.

This work assignment aims to present the *Knowledge Flow* interface, showing how it can be used utilizing filters and classifiers. Focus will be made on showing the interface capacities.

Knowledge Flow interface presentation

With the *Knowledge Flow* interface it is possible to implement most of the Explorer functionalities for those who like to think in terms of data flows. The user can design and execute the data processing with this interface.

To enter the *Knowledge Flow* interface, from the *Weka GUI Chooser*, one must press the KnowledgeFlow button as shown on figure 1.



Fig. 1 Weka Gui Chooser

This will enter the following interface:

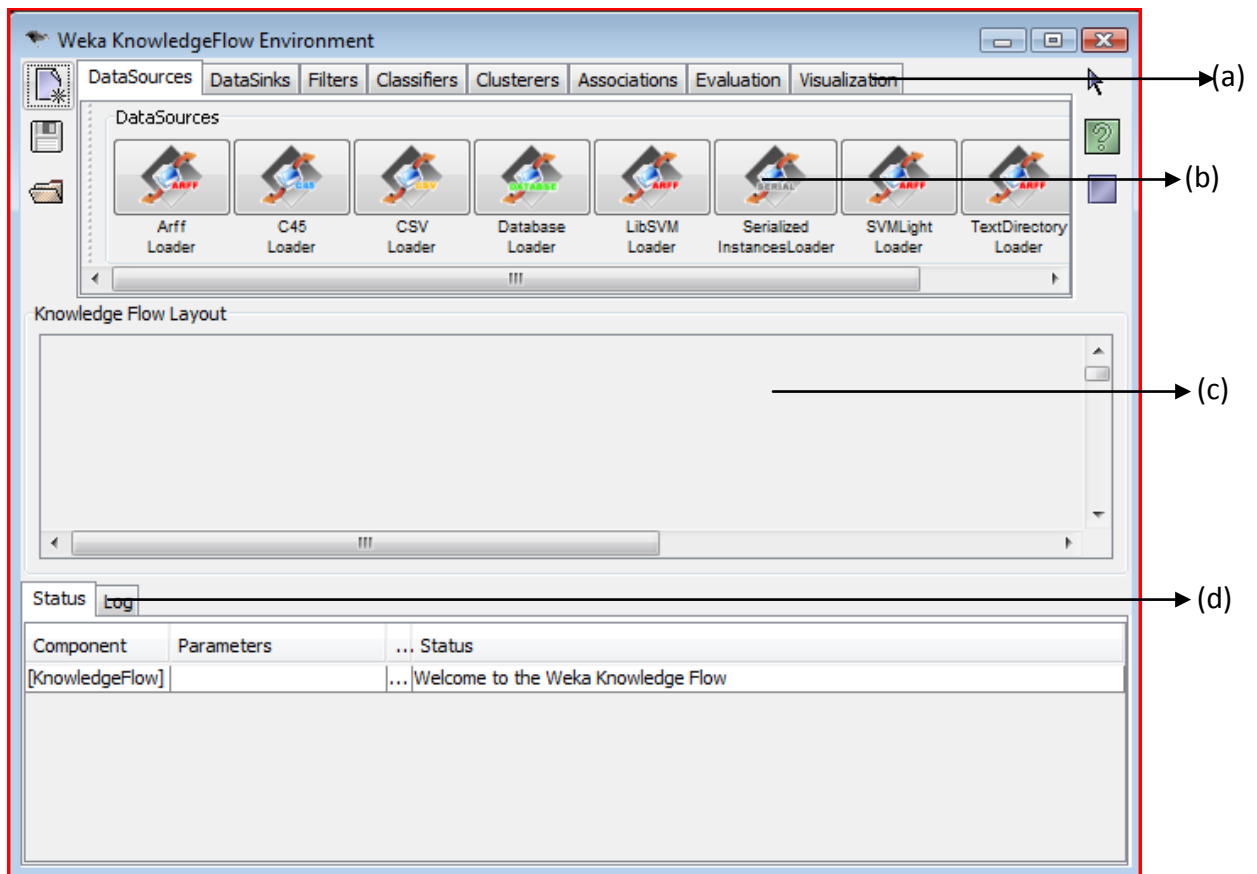


Fig. 2 Weka Knowledge Flow interface

From the toolbar (a) as shown on figure 2, it is possible to access several weka components. Clicking on each one of them, one can visualize the components of each group, on bar (b). Components are grouped according the functions they represent:

DataSources: shows all the necessary components to perform data loads from files.

DataSink: displays the necessary components to save files.

Filters: displays all Weka filters.

Classifiers: shows available Weka classifiers.

Clusterers: shows available Weka clusterers.

Evaluation: displays the available components necessary to performance analysis on each of the used classifiers. Following, a short description of these components:

TrainingSetMaker: Make a dataset into a training set.

TestSetMaker: Make a dataset into a test set.

CrossValidationFolderMaker: Split a dataset into folds.

TrainTestSplitMaker: Split a dataset into training and test sets.

ClassAssigner: Assign one of the attributes to be a class.

ClassValuePicker: Choose a value for the positive class.

ClassifierPerformanceEvaluator: Collect evaluation statistics for batch evaluation.

IncrementalClassifierEvaluator: Collect evaluation statistics for incremental evaluation.

ClustererPerformanceEvaluator: Collect evaluation statistics for clusterers.

PredictionAppender: Append classifier's predictions to a database.

Visualization: displays the necessary components for both data visualization, and the statistics of the data and the classifier performance statistics. Following, a brief description of each one of these components:

DataVisualizer: Visualize data in 2D scatter plot.

ScatterPlotMatrix: Matrix of scatter plots.

AttributeSummarizer: Set of histograms, one for each attribute.

ModelPerformance Chart: Draw ROC and other threshold curves.

TextViewer: Visualize data or models as text.

GraphViewer: Visualize tree-based models.

StripChart: Display a scrolling plot of data.

After the functionality and the correspondent component have been selected, one can click on the *Knowledge Flow* layout canvas (c) to place the component.

Right-clicking each one of these components makes a configuration menu display. For each component there is a configuration menu.

The menu in DataSource components has three different sections: Edit, Connections and Actions. Edit sections permit deleting components or open the configuration list menu, which in this case are configured opening a file. The Connections are used to link components. There are two types of connections for data sources: Dataset used for batch operations, and Instances used for stream operations. To do that, the connection type must be clicked (as long as it is active) on the out component and dropped over destination component. When the destination component is classifier, it is necessary to create, from the original dataset, a training set and a test set. To do

that, one must put an Evaluation component between the two original components, as a *TestSetMaker* or *TrainingSetMaker* or *CrossValidationEvaluator*. If the final component is Filter type, direct links (dataset or instance) can be made. Operations of type *Actions* for *DataSource* components only permit starting loading data from the data file used as data source.

With *Filters*, only two sections are available: *Edit* and *Connections*. *Edit* operations display component deletion and *Filter* configuration. This configuration is done like it is done in *Explorer*. They take input from data sources and allow *dataset* and *instances* connections.

In *Classifier*, the configuration is done the same way as it is in *Explorer*, from the *Edit* section. From a *Classifier*, graph and *text* connections can be made. That provides graphical and textual representations of classifier's learned state. These types of connections are activated only if they receive a training set input. The *batchClassifier* and *incrementalClassifier* connections are used to put data into an *Evaluation* component.

The *Edit* section for the *Evaluation* components, only allows delete actions. The *Connections* are from type *text* or *ThresholdData*. *Visualization* components have a section *Actions* for *Show results* and *Clear results*.

In the bottom part of the working canvas, there are two tab folders: *Status* and *Log* (d). With *Status* it is possible to visualize processing flow through the various components with the configured parameters. In *Log* tab it is possible to look at the commands supporting the processing flow when the processing is activated.

To take a better look of Weka KnowledgeFlow functionalities, some examples are provided in next chapter.

Knowledge Flow utilization example

Credit data base

In financial credit business, banks are interested in information whether prospective consumers will pay back their credit or not. The aim of credit-scoring is to model or predict the probability that a consumer with certain covariates is to be considered as a potential risk. The dataset presented consists of 1000 consumer credits from a German bank. For each consumer the binary response variable "creditability" is available. In addition, 20 covariates that are assumed to influence creditability were recorded. These covariates are:

- Balance of current account,
- Duration in months ,
- Payment of previous credits,
- Amount of credit in "Deutsche Mark",
- Value of savings or stocks,
- Number of years been employed by current employer ,
- Installment in percentage of available income,
- Marital Status / Sex,
- Further debtors / Guarantors,
- Number of years living in current household ,
- Most valuable available assets,
- Age in years,
- Further running credits,
- Type of apartment,
- Number of previous credits at this bank (including the running one),
- Occupation,
- Number of persons entitled to maintenance,
- Telephone,
- Foreign worker.

Knowledge Flow creation on *credit* data base:

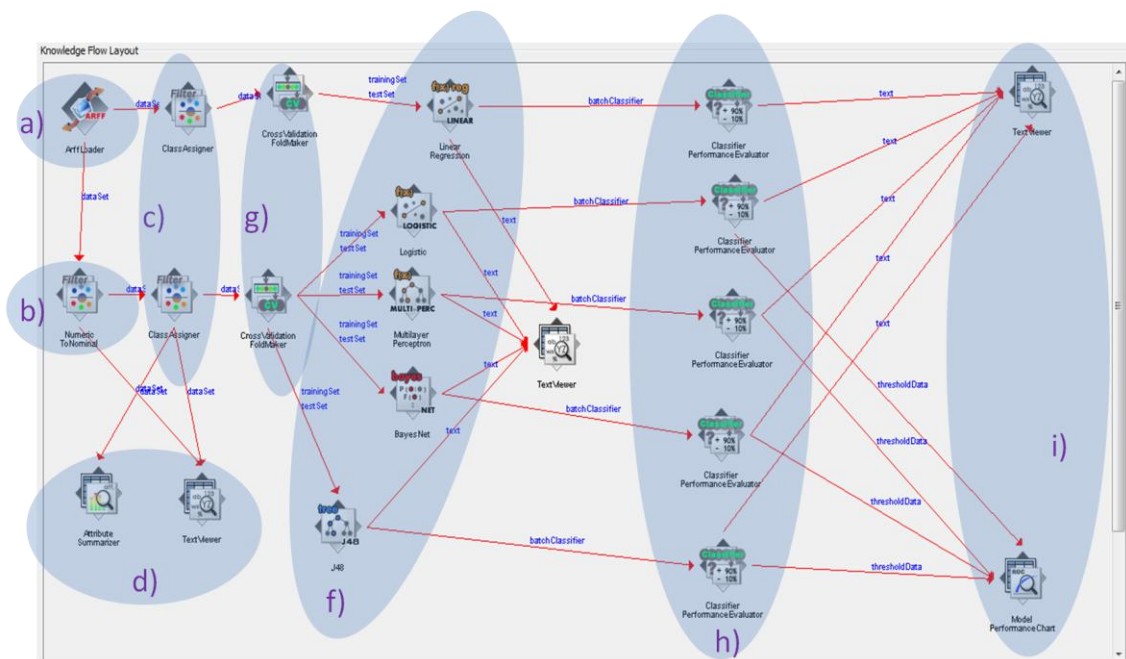


Fig. 3 Weka Knowledge Flow

All database attributes are numeric. Some variables however are categorical. This way, it is possible to apply a filter over the database in order to make a nominal transformation on the attributes.

This way an *ArffLoader* (a) component is needed to load credit data file.

A *dataset* connection is used to drop the file into a *Filter* (b) component. This component will convert all attributes selected in configuration window, into nominal ones.

Then data flow to another filter, *Class Assigner* (c), used to define the attribute credit as class. Without this previous operation, classifiers don't work.

With *Visualization* components, *Attribute Summarizer* and *Text Viewer* (d), it's possible to verify these transformations.

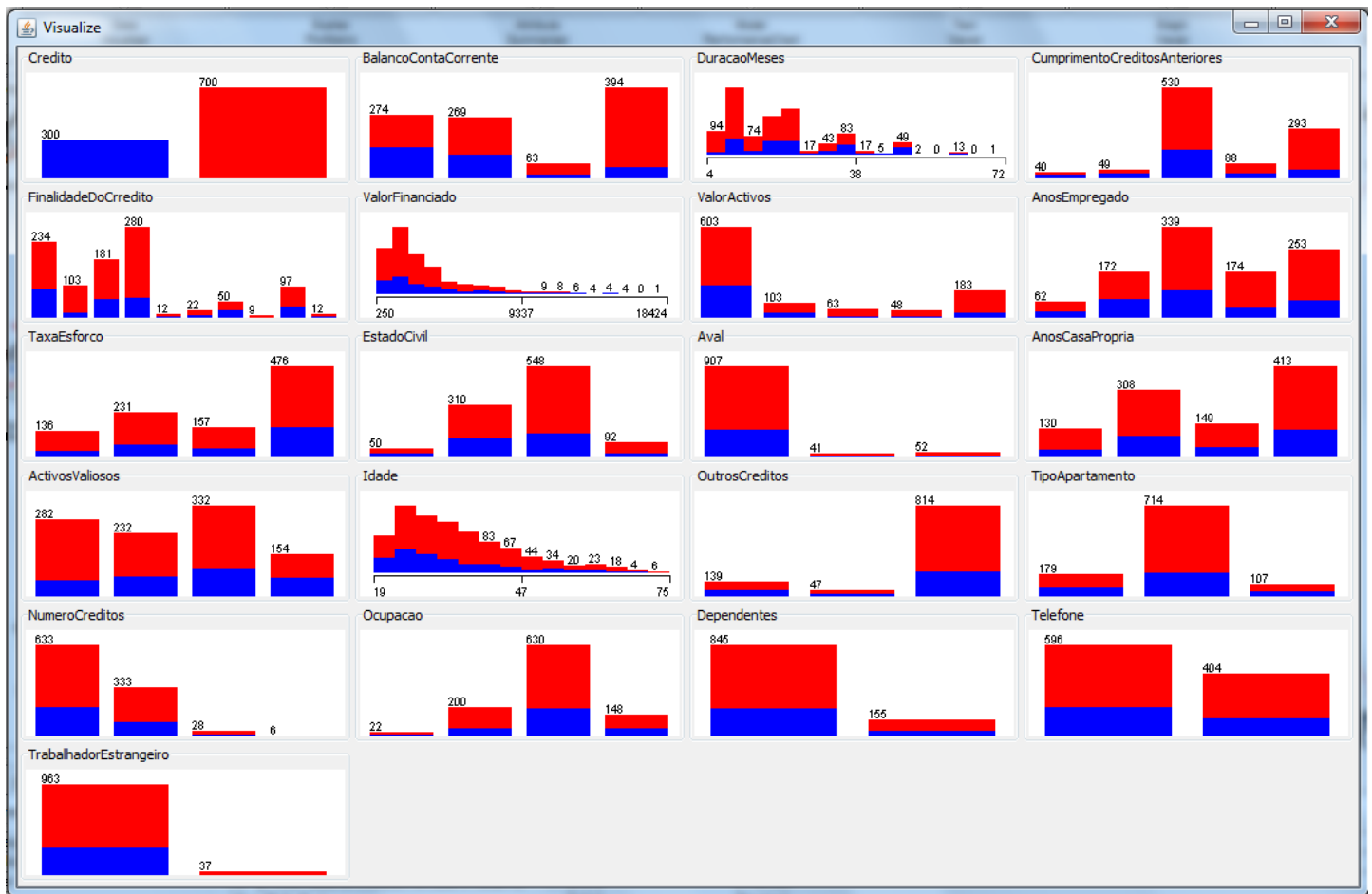


Fig. 4 Weka attribute summarizer

Fig. 4 shows the *Attribute Summarizer* viewer. With this view it's possible to understand that *Credit* is the class attribute and all attributes are nominal except *Balance of Current Account*, *Duration* and *Age*. For each nominal attribute there is a bar for each category and each bar has two colors: red and blue. The red one indicates value 1 for class attribute *credit* and the blue one, indicates value 0. So for each bar a proportion of observation with each value for *credit* is shown.

Since linear regression does not accept nominal attributes as class, it was created a *Class Assigner* filter connected with *Arff Loader*, for test linear regression classifier.

The classifiers that will be testes are: Logist and Linear regression, Multilayer Perceptron, Bayes Net, J48 (M5 – tree) Fig.3 (f).

To connect the classifiers, *Cross Validation Fold Maker* Fig.3 (g) was used to create the training sets and test sets that will be operated by the classifiers. This way, this component is connected to each one of the classifiers.

In turn, these are connected using a *Classifier Performance Evaluator* Fig.3 (h) and the results are viewed through a *TextViewer and Model Performance Chart* Fig.3 (i).

The results from linear regression were not connected to *Model Performance Chart* because the output from this regression is continuous and so, there is no sense in ROC curve analysis.

Results

Linear Regression:

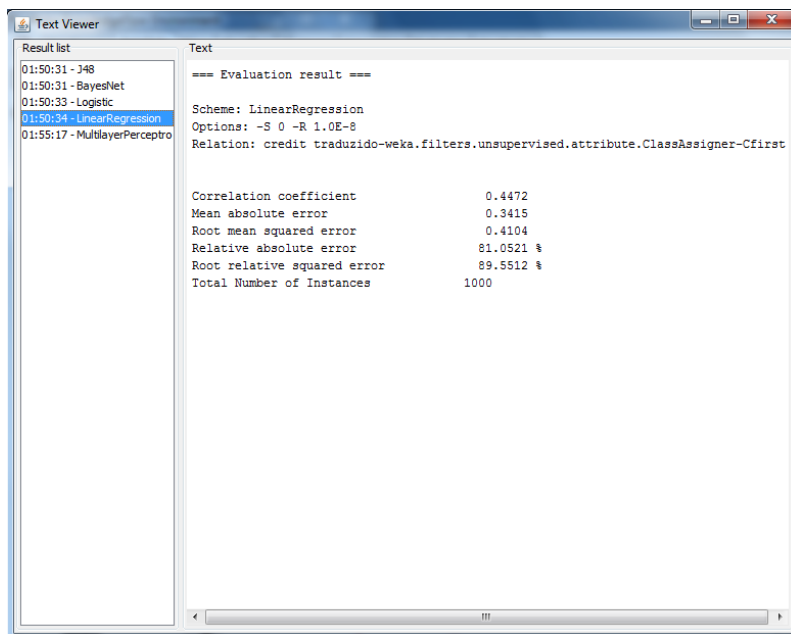


Fig. 5 Text view linear regression

Correlation coefficient too low = 0.4472

Relative absolute error 81% is a high value

This does not reveal a good model. In fact that should be waited because the output variable *credit* is binary discrete (0 or 1).

BayesNet:

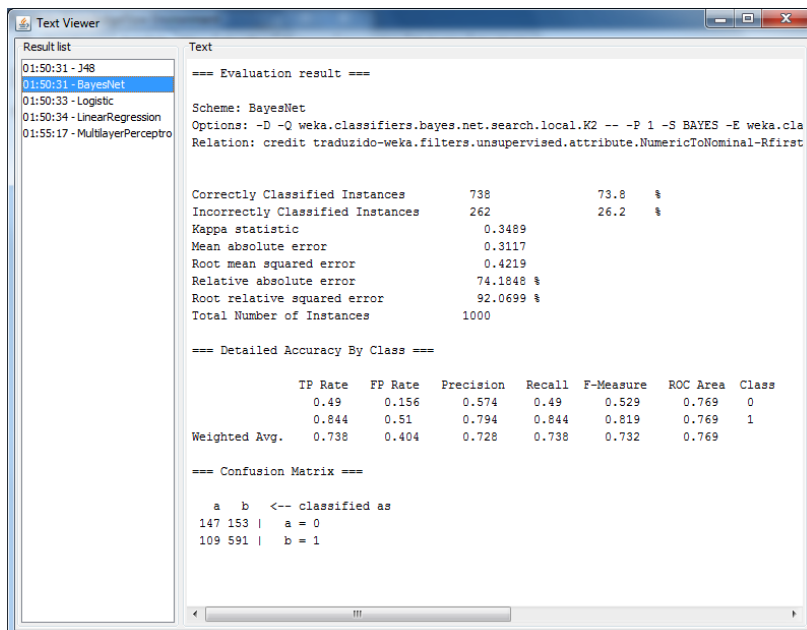


Fig. 6 Text view Bayes net

Kappa statistic: 0.3489

True positive Rate:

Class 0: 0.49

Class 1: 0.844

Roc area: 0.769

Correctly Classified Instances: 73.8%

J48:

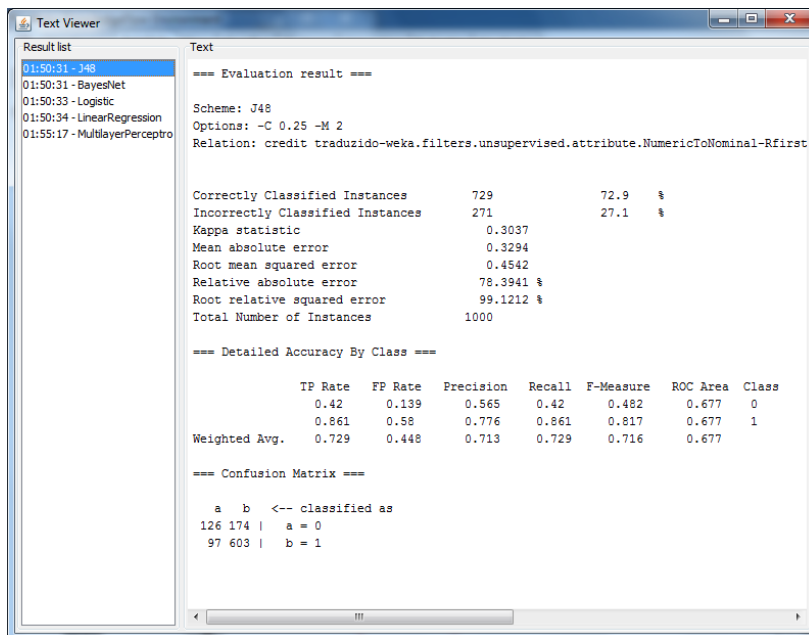


Fig. 7 Text view J48

Kappa statistic: 0.3037

True positive Rate:

Class 0: 0.42

Class 1: 0.861

Roc area: 0.677

Correctly Classified Instances: 72.9%

MultilayerPreceptron:

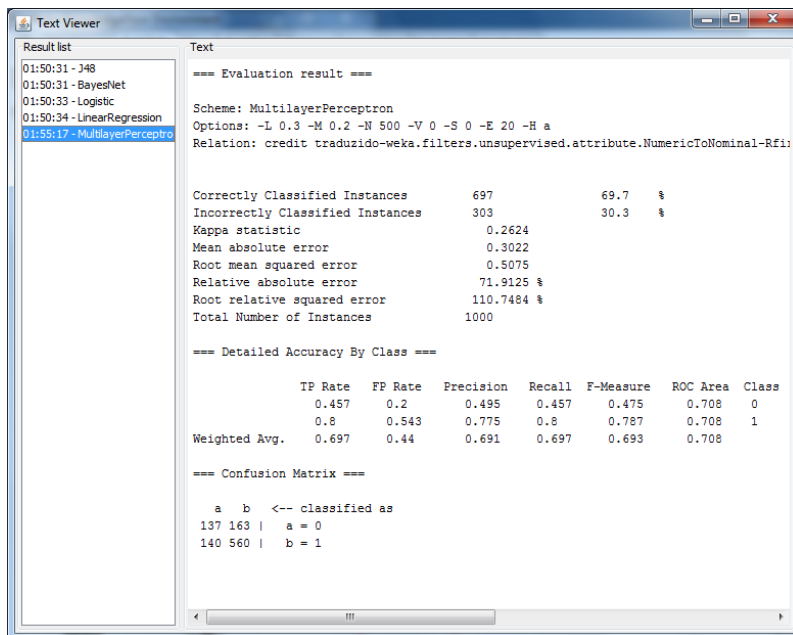


Fig. 8 Text view Multilayer

Kappa statistic: 0.2624

True positive Rate:

Class 0: 0.457

Class 1: 0.8

Roc area: 0.708

Correctly Classified Instances: 67.9%

Logistic:

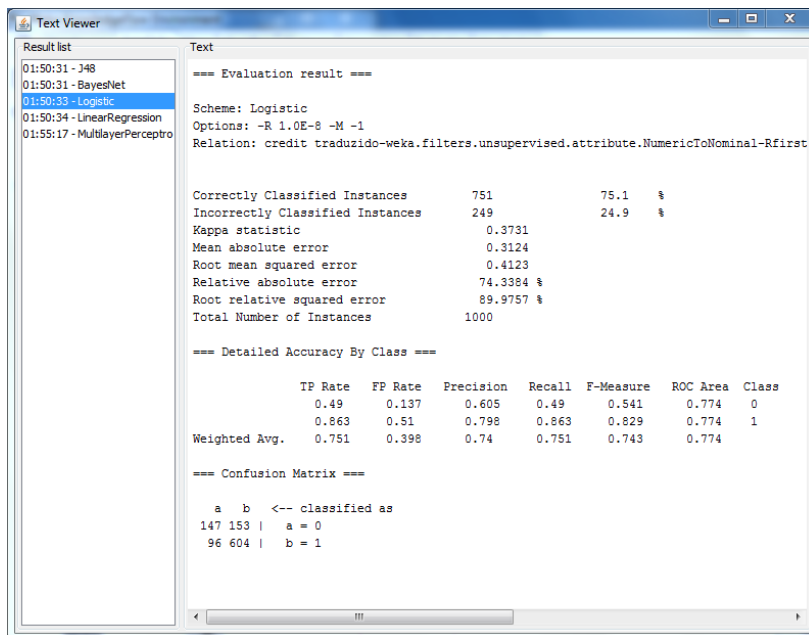


Fig. 9 Text view Logistic

Kappa statistic: 0.3731

True positive Rate:

Class 0: 0.49

Class 1: 0.863

Roc area: 0.774

Correctly Classified Instances: 75.1%

Interpreting the results

Analyzing the results on the shown classifiers, it's possible to observe that Logistic Regression it's the one that shows higher Kappa statistic, greater rate of correctly classified instances, and a also a better True Positive rate on both classes.

Also, observing bellow ROC curve, Logistic Regression it's the one with the highest value. This way, one can conclude that this is the best classifier for this problem.

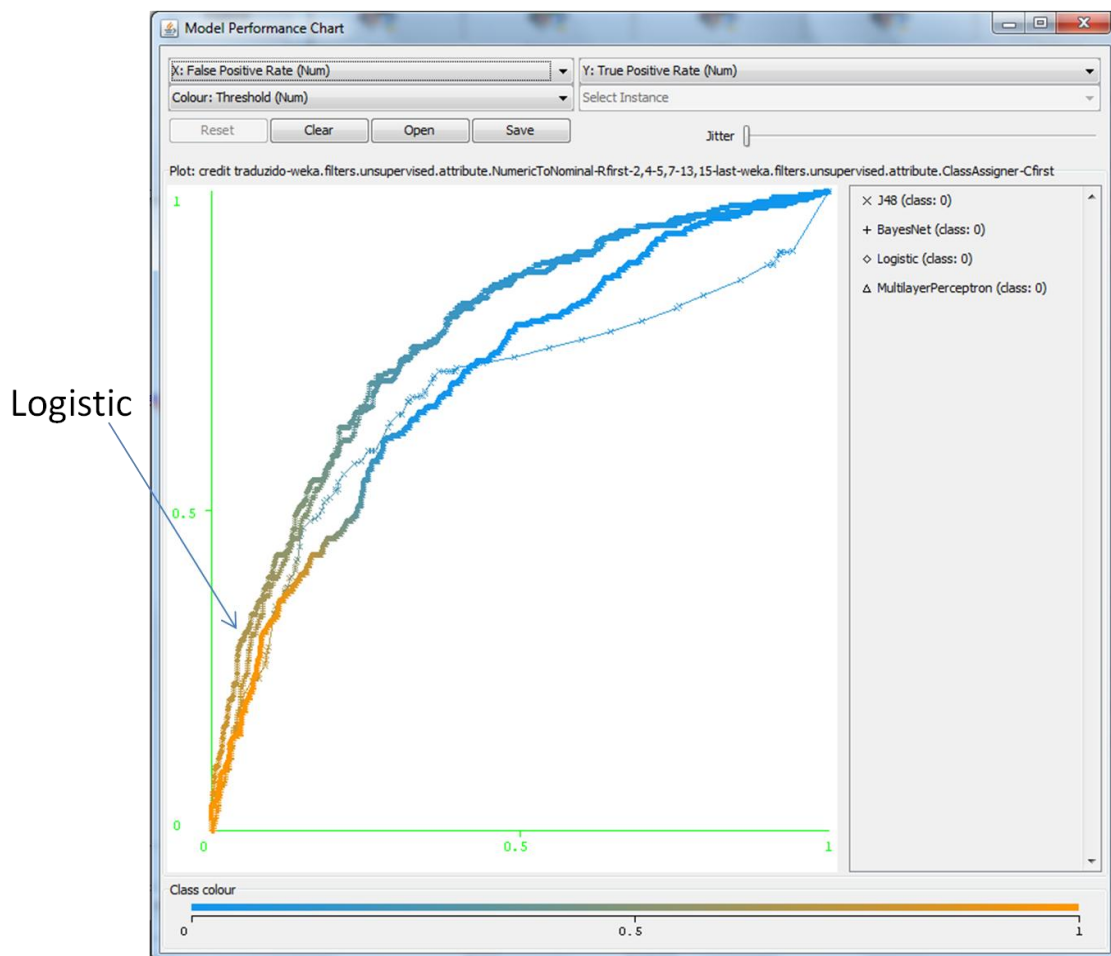


Fig. 10 Model performance chart

The Model Performance chart shown in Fig.10, confirms Logistic Regression as the best classifier.

Incremental Learning

One of the strengths Knowledge Flow represents facing Explorer it's his capacity to process incrementally. Sometimes the input file is so large that it cannot be loaded into computer's memory. If when generating the model, the learning scheme processes data incrementally, one instance at a time, it should be no problem. One instance is read from the input file, the model is updated, and then the next read can be made and so on. However not all models are yet developed to perform incrementally in some cases, and other methods like instance-based schemes and locally weighted regression, need access to all instances before a prediction can be made, on the other cases.

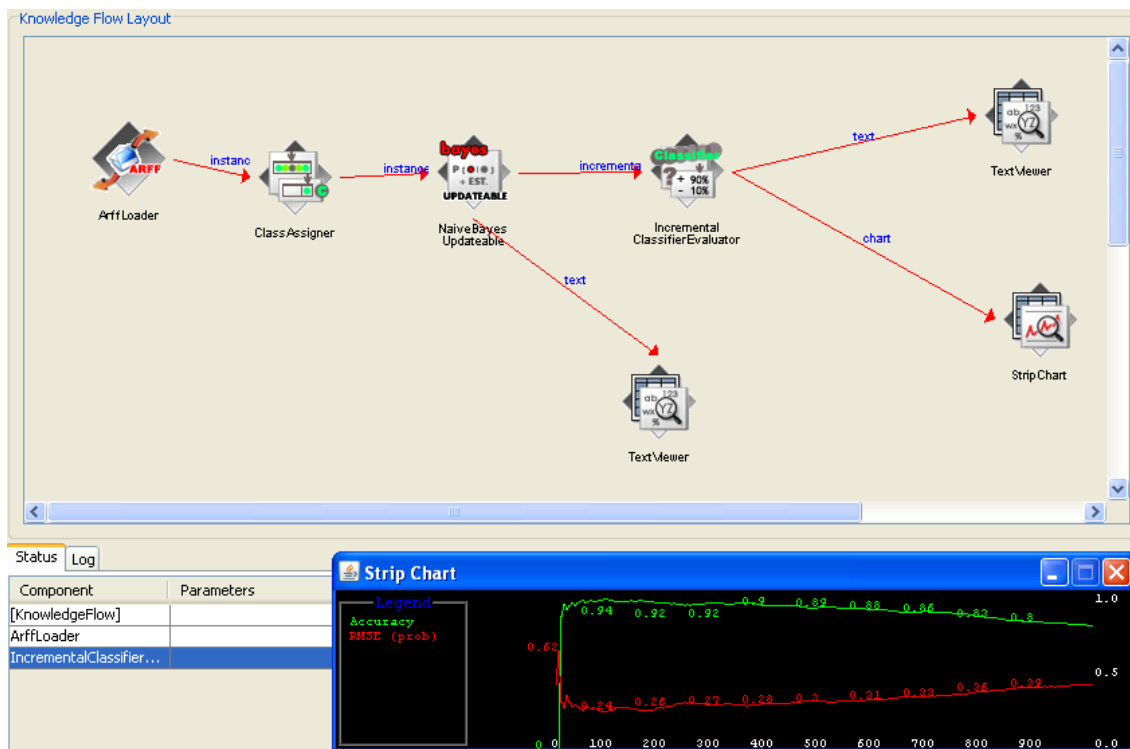


Fig. 11 Incremental strip chart

Fig. 11 presents NaiveBayesUpdatable classification method which runs incrementally on input data. The strip chart shows how the classifier behaves instance by instance with the evolution of Accuracy and RMSE. In this specific example, we can see that Accuracy decays and RMSE rises as instances are processed, which shows this is not a good method.

Conclusions

With WEKA's KnowledgeFlow module, it is possible to access a working environment that provides the combination of filters and classifiers on a set of data. It is of high value for several reasons. It allows the validation of the outcomes by evaluating the performance on the used classifiers. It gives a lifecycle perspective, from the starting data file, to the various outputs that can be configured. This way, the whole process flow can be configured to run a sequence of predefined tasks and transformations repeatedly, or for use with different input data set's. Is also makes possible the use of an incremental approach which is valuable for instance, lowering processing power demands.

Bibliography

Kirby, R. and Frank, E. and Reutemann, P. (2008) WEKA Explorer User Guide for version 3-5-8, University of Waikato

Witten, I.H. and Frank, E. (2005) Data Mining: Practical machine learning tools and techniques. 2nd edition Morgan Kaufmann, San Francisco.