Using the Weka Scoring Kettle Plugin

For the latest information, please visit our web site at www.pentaho.org

Last Modified on October 29, 2007

Contents

Introduction

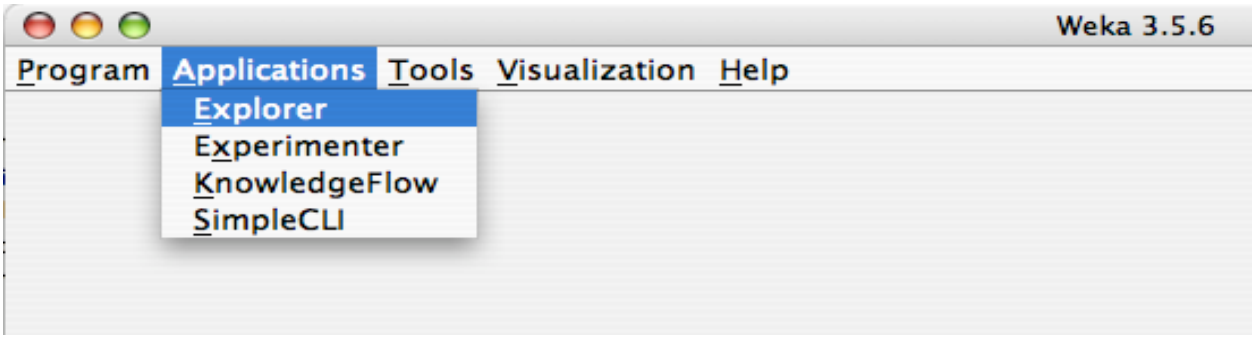
The Weka scoring plugin is a tool that allows classification and clustering models created with Weka to be used to "score" new data as part of a Kettle transform. "Scoring" simply means attaching a prediction to an incoming row of data. The Weka scoring plugin can handle all types of classifiers and clusterers that can be constructed in Weka. It provides the ability to attach a predicted label (classification/clustering), number (regression) or probability distribution (classification/clustering)[1] to a row of data.

Getting Started

In order to use the Weka scoring plugin, a model must first be created in Weka and then exported as a serialized Java object to a file. The model can then be loaded by the plugin and applied to fresh data. This section briefly describes how to create and export a model from Weka.

Starting the Weka Explorer

Assuming you have Weka 3.5 installed, launch the Weka environment by double-clicking on the weka.jar file or by selecting it from the Start menu (under Windows). Once the UI is visible, choose Explorer from the Applications menu.



Alternatively, you can start the Explorer directly from the command line by typing

```
java -cp path_to_directory_containing_weka/weka.jar weka.gui.explorer.Explorer
```

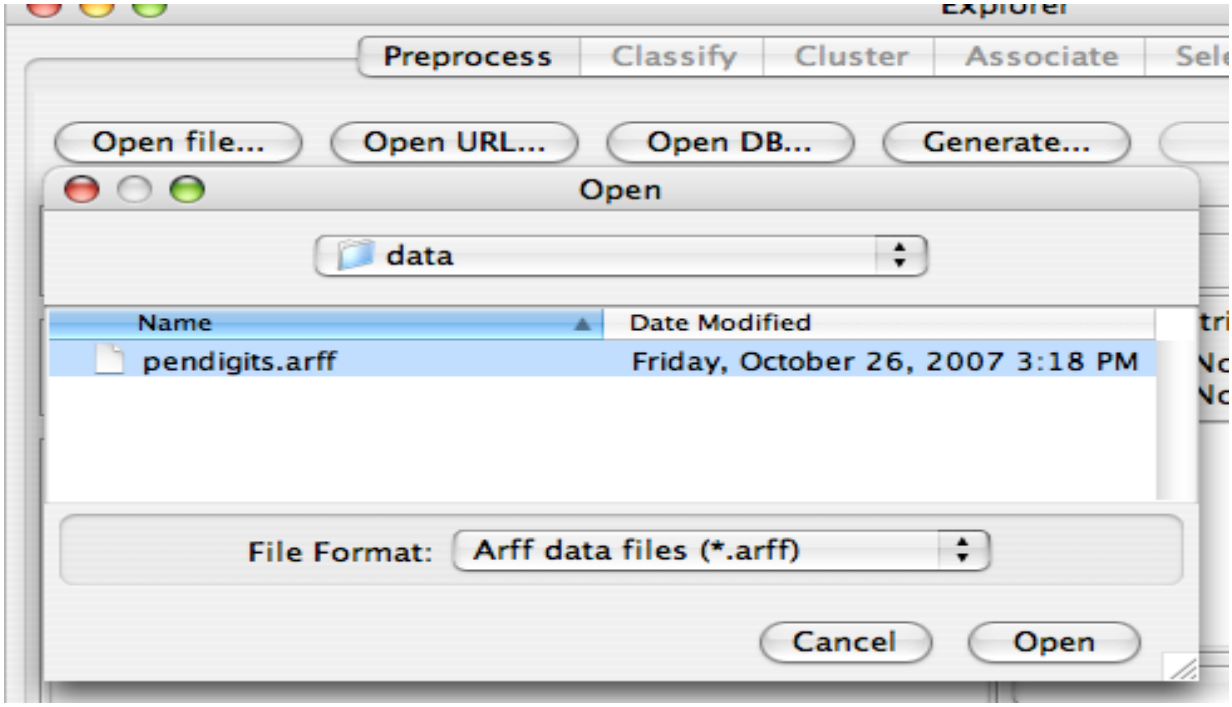
The latter approach allows you to control how much memory is made available to the Java virtual machine through the use of the "-Xmx" flag.

Loading Data into the Explorer

Data can be imported into the explorer from files (arff, csv or c4.5 format) or from databases. In this example we will load data from a file in Weka's native arff (Attribute Relation File Format) format. Click on "Open File" and select the "pendigits.arff" file. The file will be loaded and summary statistics for the attributes shown in the Preprocess panel.

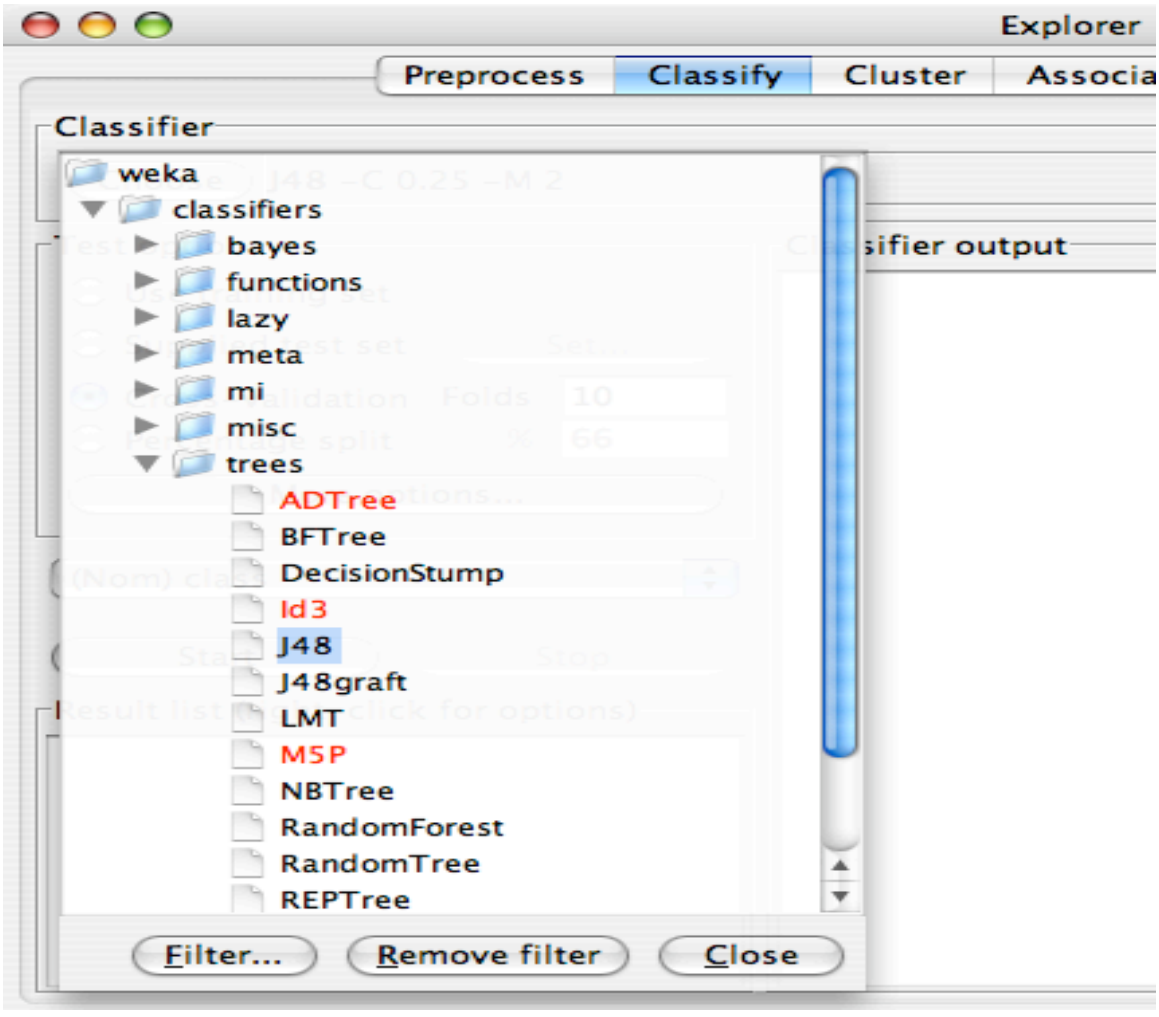
---

[1] Only certain classifiers and clusterers are capable of providing probability estimates.



Building a Classifier

In the Classifier panel of the Explorer first choose a learning scheme to apply to the training data. In this example you will use a decision tree learner (J48).

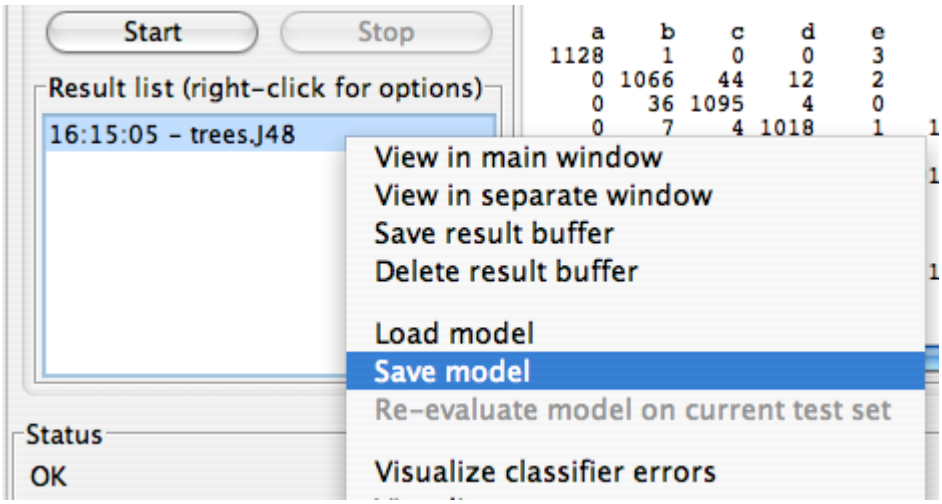


Clicking on the scheme summary will bring up a dialog window that allows you to configure the parameters of J48. The defaults work well in most cases.

The rest of the default settings for evaluation are good for most situations, so you can simply press the "Start" button to launch the training and evaluation of the learning scheme. The default is to perform a 10-fold cross-validation of the learning scheme on the training data, so the statistics on performance that are reported are fairly reliable estimates of what can be expected on future data.

Exporting the Trained Classifier

You can save export any classifier that you have trained in the Classifier panel by right clicking on its entry in the Results History. Trained models are stored on disk a serialized Java objects. Save this model to a file called "J48" (a ".model" extension will be added for you).



Using the Weka Classifier in Kettle

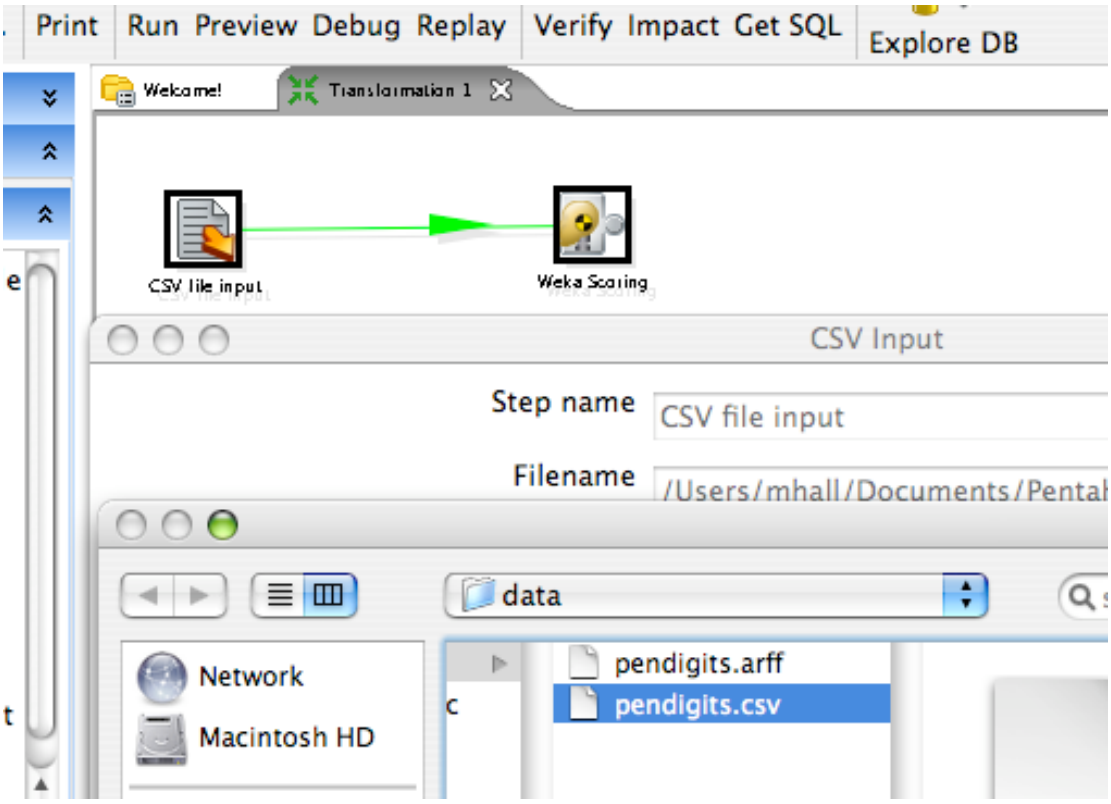
Using the trained model in Kettle to score new data is simply a matter of configuring the Weka scoring plugin to load and apply the model file you created in the previous section.

Preliminaries

First make sure that the Weka scoring plugin is installed correctly in Kettle—unpack the plugin archive and copy all files in the WekaScoringDeploy directory to a sub-directory in the plugins/steps directory of your Kettle installation. Now start Spoon (you may want to give more memory to the Java virtual machine if your Weka model is large).

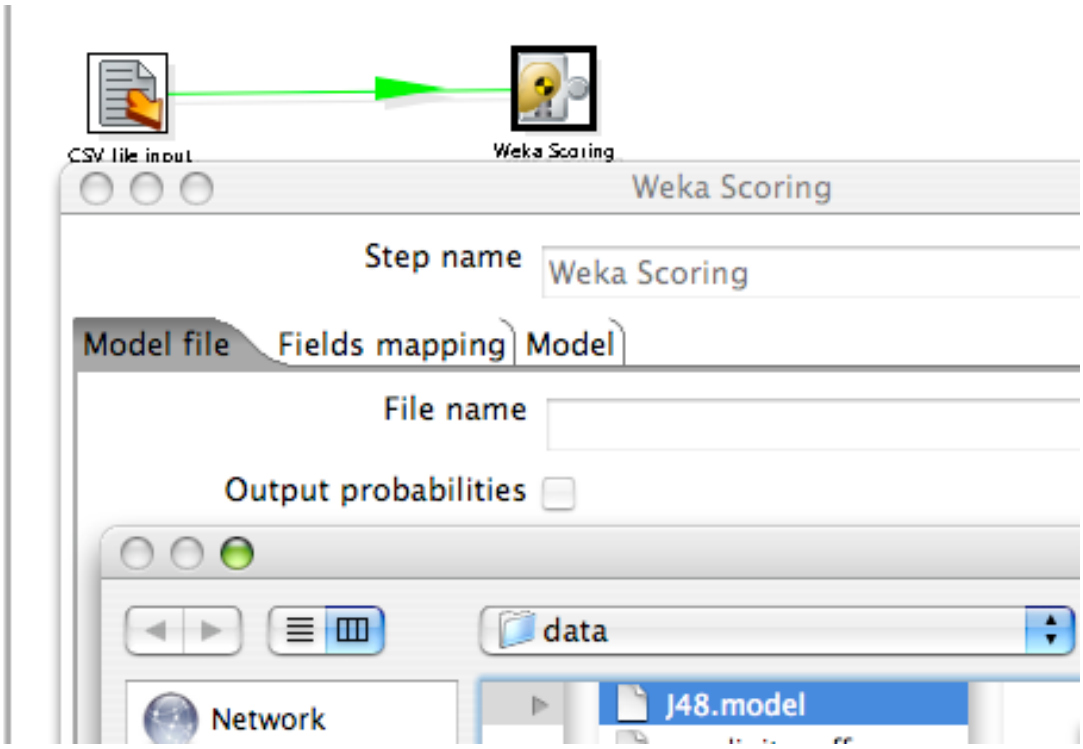
A Simple Example

As a simple demonstration of how to use the scoring plugin, you will use the model you created in Weka to score the same data that it was trained on. First start Spoon, and then construct a simple transform that links a CSV input step to the Weka scoring step.

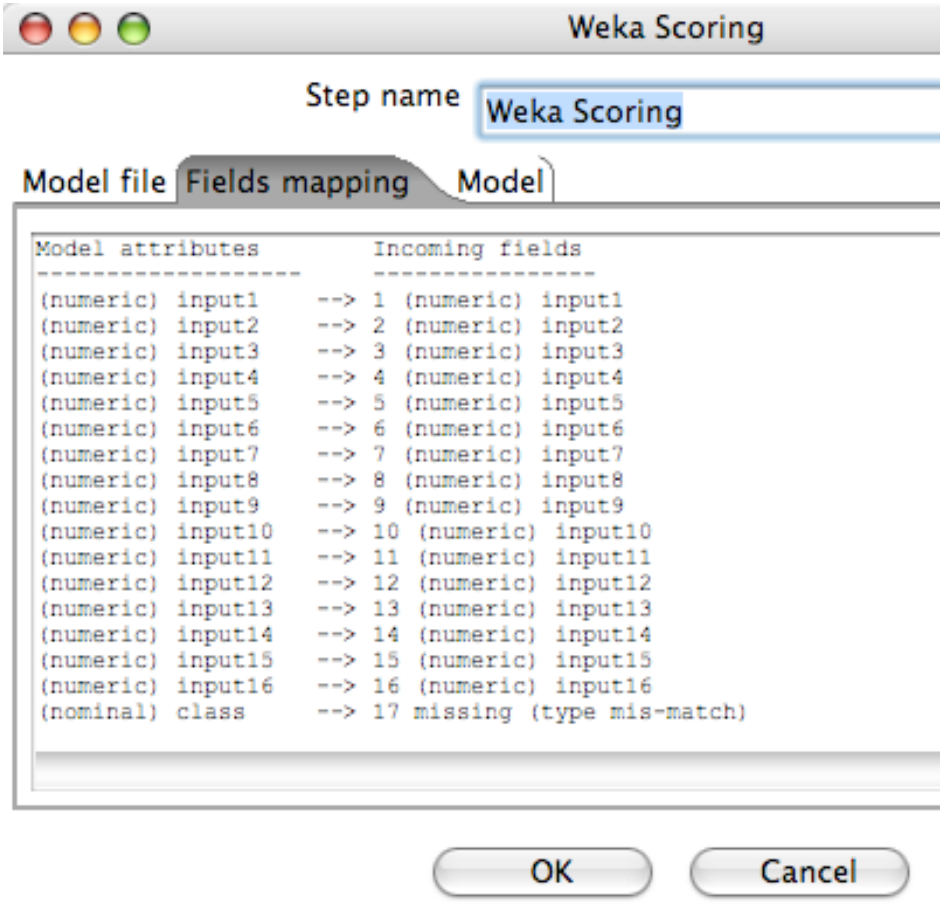


Next, configure the CSV input step to load the "pendigits.csv" file (this is the same data as pendigits.arff, but in csv format). Make sure that the Delimeter text box contains a "," and then click on "Get Fields" to make the CSV input step analyze a few lines of the file and determine the types of the fields.

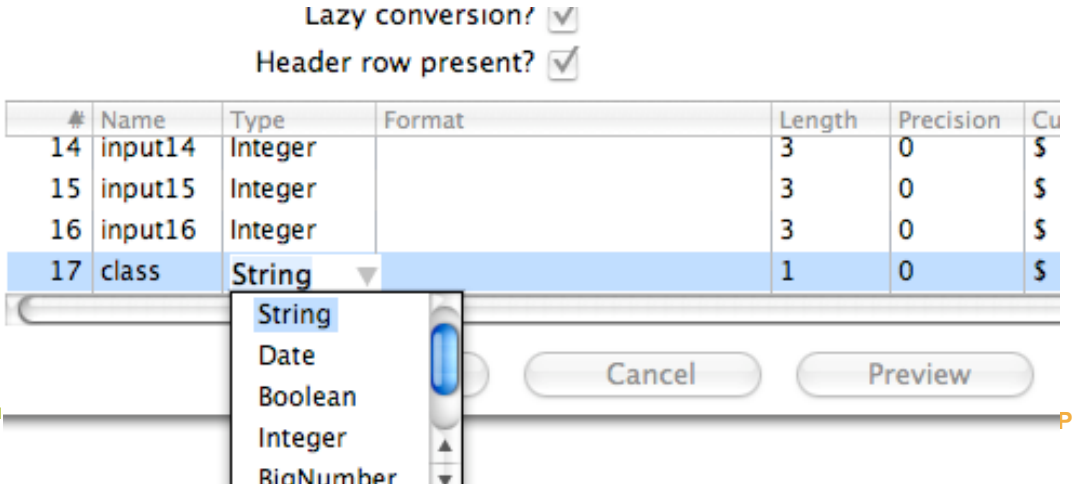
Now configure the Weka scoring step. First load your J48 model.



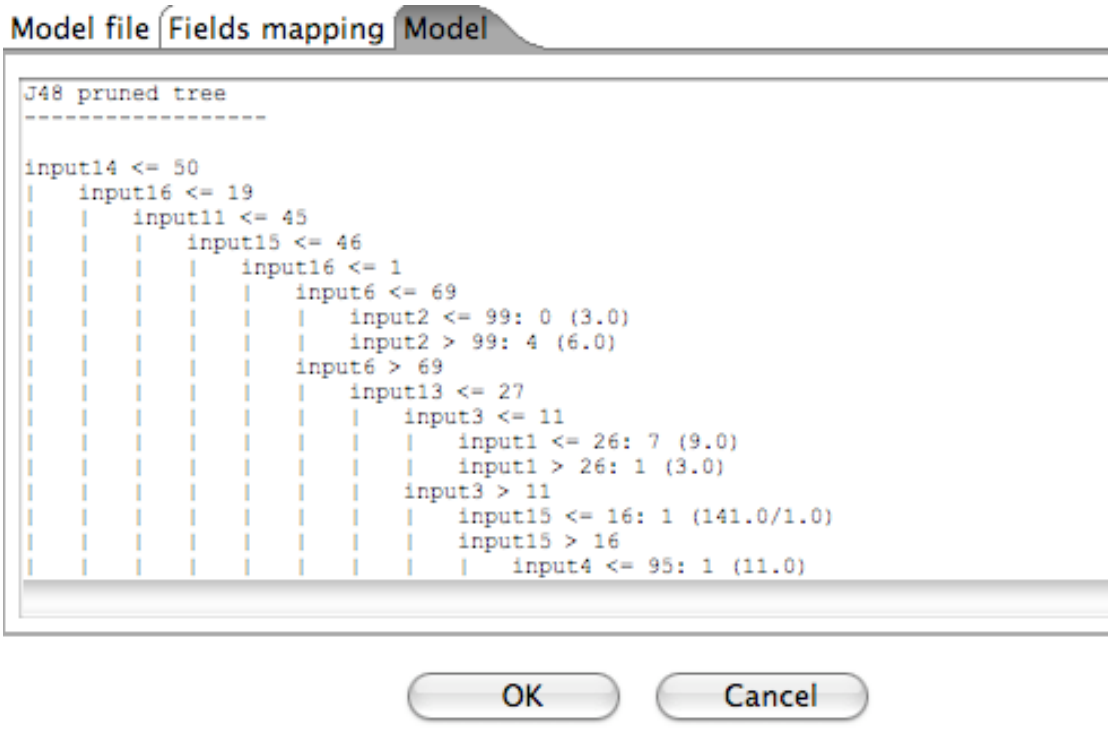
The fields mapping tab of the scoring dialog shows how the fields in the incoming data from the previous step in the transform—the CSV input step—have been matched with the attributes in the data that the model was trained from. Any attributes that don't have a counterpart in the incoming data are indicated by an entry labeled "missing". If there is a difference in type between a model attribute and an incoming field, then this will be indicated by the label "type-mismatch". In both cases, the classifier will receive missing value as input for the attribute in question for all incoming data rows. In the figure below, there is a type mismatch between the attribute "class" and the incoming field with the same name.



Returning to the CSV input step's configuration dialog, it can be seen that the type of the "class" field has been identified as "Integer". While this is correct as far as the raw data goes, this attribute has been encoded as nominal in the arff file that the classifier was trained from. You can resolve this type mismatch by changing the type of this field to "String" in the CVS input step's dialog.



The "Model" tab of the Weka scoring step's dialog shows the textual description of the classifier (exactly as it was shown in the Classifier panel of the Weka Explorer).



Click on the "Preview" button in Spoon to preview the result of using the classifier to score the pendigits data. Predictions for the "class" field have been appended as a new field at the end of the data.

Rows of step: Weka Scoring

#	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11	input12	input13	input14	input15	input16	class	class(predicted)
1	47	100	27	81	57	37	26	0	0	23	56	53	100	90	40	98	8	8
2	0	89	27	100	42	75	29	45	15	15	37	0	69	2	100	6	2	2
3	0	57	31	68	72	90	100	100	76	75	50	51	28	25	16	0	1	1
4	0	100	7	92	5	68	19	45	86	34	100	45	74	23	67	0	4	4
5	0	67	49	83	100	100	81	80	60	60	40	40	33	20	47	0	1	1
6	100	100	88	99	49	74	17	47	0	16	37	0	73	16	20	20	6	6
7	0	100	3	72	26	35	85	35	100	71	73	97	65	49	66	0	4	4
8	0	39	2	62	11	5	63	0	100	43	89	99	36	100	0	57	0	0
9	13	89	12	50	72	38	56	0	4	17	0	61	32	94	100	100	5	5
10	57	100	22	72	0	31	25	0	75	13	100	50	75	87	26	85	0	0
11	74	87	31	100	0	69	62	64	100	79	100	38	84	0	18	1	9	9
12	48	96	62	65	88	27	21	0	21	33	79	67	100	100	0	85	8	8
13	100	100	72	99	36	78	34	54	79	47	64	13	19	0	0	2	5	5

Enabling the "Output probabilities" checkbox on the Weka scoring dialog's "Model file" tab will result in a predicted probability distribution being appended to each incoming data row. In this case there will be one new field containing a predicted probability for each possible class label (Note: this option is only available for models that have been trained on a discrete class problem).

Rows of step: Weka Scoring

ut14	input15	input16	class	class:0(predicted prob)	class:1(predicted prob)	class:2(predicted prob)	class:3(predicted prob)	class:4(predicted prob)	class:5(pr…icted prob)	class:6(p…ted prob
	40	98	8	0.001976285	0.0	0.0	0.0	0.0	0.001976285	0.001976285
	100	6	2	0.0	0.009615385	0.990384615	0.0	0.0	0.0	0.0
	16	0	1	0.0	0.998050682	0.0	0.0	0.0	0.0	0.0
	67	0	4	0.0	0.0	0.0	0.0	0.99713467	0.0	0.00191022
	47	0	1	0.0	0.941176471	0.058823529	0.0	0.0	0.0	0.0
	20	20	6	0.0	0.0	0.0	0.0	0.0	0.0	1.0
	66	0	4	0.0	0.0	0.0	0.0	0.99713467	0.0	0.00191022
0	0	57	0	0.996105161	0.0	0.0	0.0	0.0	0.00097371	0.0
	100	100	5	0.0	0.0	0.0	0.0	0.0	0.998324958	0.001675042
	26	85	0	0.996105161	0.0	0.0	0.0	0.0	0.00097371	0.0
	18	1	9	0.0	0.0	0.0	0.006535948	0.0	0.0	0.0
0	0	85	8	0.001976285	0.0	0.0	0.0	0.0	0.001976285	0.001976285

Tips and Tricks

Maximizing Throughput

In order to maximize the data throughput in the Weka scoring step, it is necessary to understand how Weka represents attribute values internally. All values encapsulated in Weka's Instance class are stored in primative Java double floating point format. This is the case for integer, real and nominal (discrete) values. In the case of the latter, the value represents the index of the discrete value, stored in double floating point format. Maximum speed, when the scoring step converts Kettle rows to Weka Instances, will be achieved when the incoming Kettle rows contain values that are Numbers (Doubles) for all numeric fields and Strings for all discrete fields. Integers or Booleans will trigger a conversion to Double. When there are a lot of fields, this conversion will affect performance.

A Note About the CSV Input Step

Kettle's CSV input step has an option called "Lazy conversion". By default, lazy conversion is turned on. This means that fields are read from the csv file into byte arrays. Construction of objects that represent the correct type for a particular field only occurs **when a Kettle step requests a particular value** from a row of data. This makes reading rows of data from the csv file extremely fast, and little overhead is imposed as long as downstream steps are not requesting a lot of values from the data rows. In the case of the Weka scoring step, most, if not all, values in a row will need to be accessed in order to construct an Instance to pass to the classifier. When there are many attributes/fields this can result in a performance hit. Better performance can be achieved by turning off "Lazy conversion" in the CSV input step. This has the effect of making the CSV input step construct the correct type of object for a field as data is read in.