MLWizard Documentation

available on-line under: `https://www.madm.eu/rapidminer/mlwizard`

Matthias Reif

July 26, 2012

Abstract

MLWizard is a tool supporting non-experts in developing classification system within RapidMiner. For a given data set, it automatically recommends, constructs, and optimizes a classification process. The user is guided in three simple steps from its dataset to a ready-to-use classification system.

MLWizard implements several meta-learning methods. Based on the knowledge about many datasets, it predict the accuracy of different classification algorithms for the given dataset. Additionally, it provides an improved genetic optimization for the most important parameters of the classifiers by replacing the random start population with already promising solutions.

1 Installation

The easiest for installing MLWizard is to download it over the RapidMiner update mechanism. For this you click on `Help` ⇒ `Update RapidMiner` in the menu bar and select MLWizard for installation. After the automatic installation and a restart of RapidMiner, the functionality of MLWizard is available in RapidMiner.

Alternatively, you can also download and install MLWizard manually from the website[1]. Download the archive and extract its content to the `lib/plugins` folder of your RapidMiner installation.

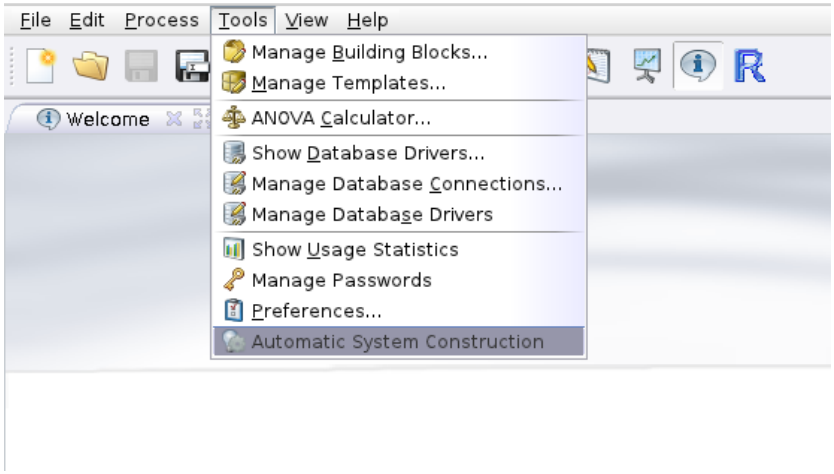
The source can be compiled using ant and the the provided build.xml Therefore, you need to have the RAPIDMINER_HOME environment variable set to your RapidMiner location or you have to adjust the build.xml file and set the rm.dir variable correctly. Then, you can create the MLWizard jar-file by:

```
1 :$ ant createJar
```

[1] `https://www.madm.eu/rapidminer/mlwizard`

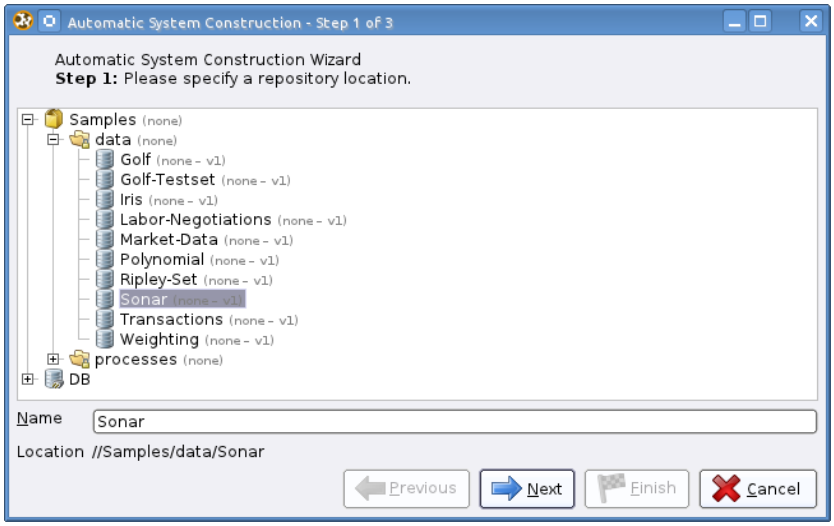
2 Usage

The wizard can be found within the Tools menu in the menu bar of RapidMiner:



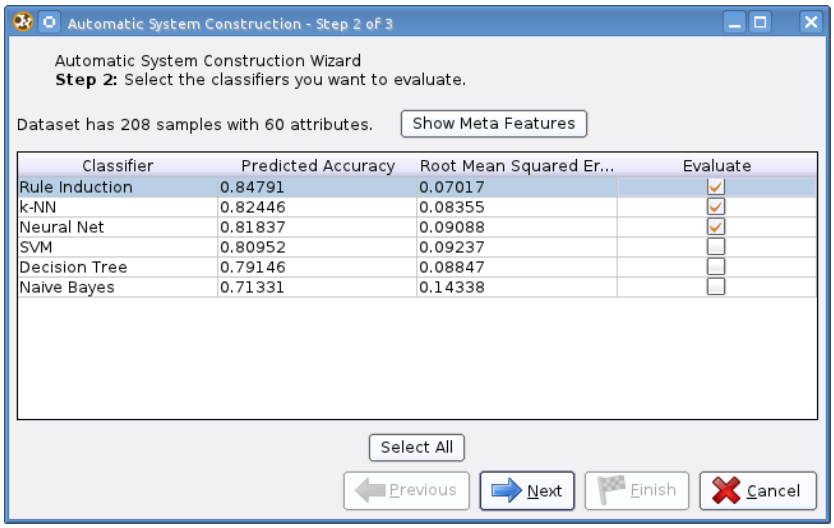
Step 1: Dataset Selection

First, a dataset has to be selected from any repository of RapidMiner:



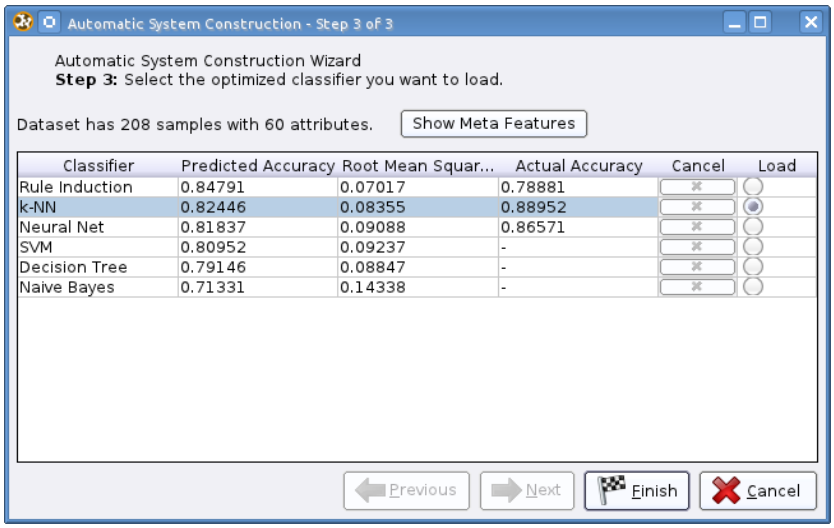
Step 2: Evaluation Selection

After the dataset has been analyzed, the predicted accuracies of the classifiers are shown. Now, the classifier that should be actually evaluated on the dataset have to be selected:



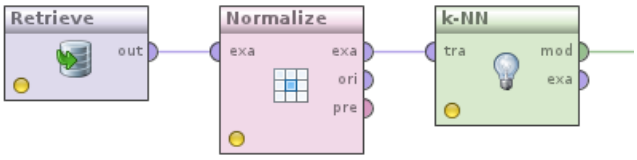
Step 3: Classifier Selection

The evaluation of the classifiers may take some time. After it is finished, the computed accuracies are shown and the classifier on which basis the classification system should be constructed has to be selected:



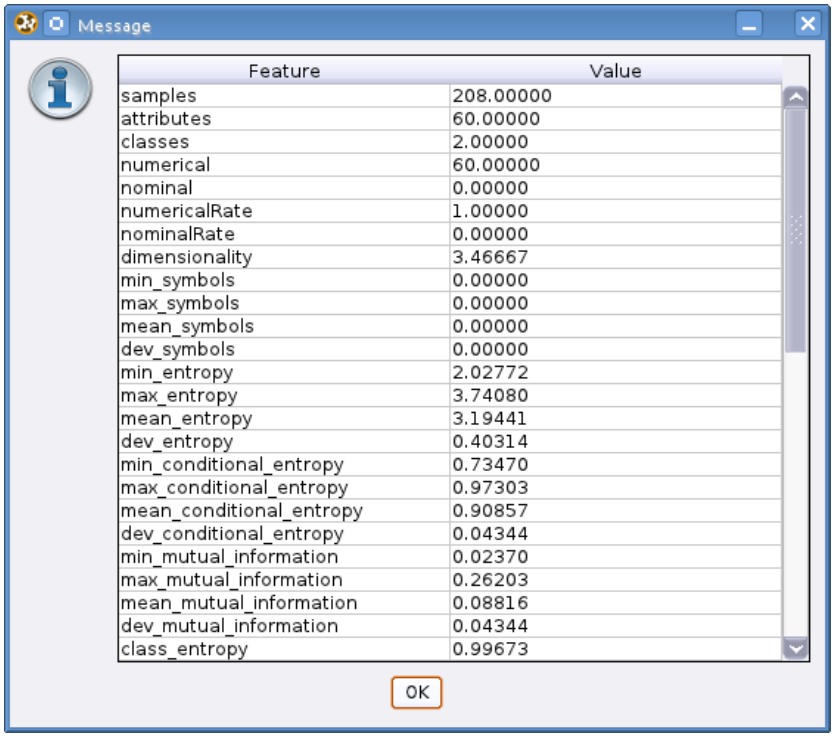
System Construction

Finally, the system is constructed automatically. If a classifier has been chosen that was also evaluated, the optimized parameter values have been already set. Otherwise, default values are used.



Meta-features

It is also possible to look at the meta-features of the datasets. These are the properties of the dataset used for the classifier recommendation and the parameter optimization.



Feature	Value
samples	208.00000
attributes	60.00000
classes	2.00000
numerical	60.00000
nominal	0.00000
numericalRate	1.00000
nominalRate	0.00000
dimensionality	3.46667
min_symbols	0.00000
max_symbols	0.00000
mean_symbols	0.00000
dev_symbols	0.00000
min_entropy	2.02772
max_entropy	3.74080
mean_entropy	3.19441
dev_entropy	0.40314
min_conditional_entropy	0.73470
max_conditional_entropy	0.97303
mean_conditional_entropy	0.90857
dev_conditional_entropy	0.04344
min_mutual_information	0.02370
max_mutual_information	0.26203
mean_mutual_information	0.08816
dev_mutual_information	0.04344
class_entropy	0.99673

3 API

The implementation of MLWizard is divided into core functionality and the GUI components. The three main functionalities of MLWizard can also be easily accessed via an API.

Sample Usage

```
ExampleSet dataset = ... // e.g. using RapidMiner file readers
KnowledgeBase knowledgeBase = KnowledgeBase.read();
// predicting the accuracies of all included classifiers
RegressionResult result = Regressioner.predict(knowledgeBase, dataset);
// evaluating a set of classifiers
Evaluator evaluator = new Evaluator();
Iterable<Classifier> classifiers = ... // e.g.
    knowledgeBase.getClassifiers()
Map<Classifier, ParameterSet> result = evaluator.evaluate(classifiers,
result.metaFeatures, knowledgeBase, dataset, nThreads);
// constructing the final system
String path = .. // path where the dataset was read from
Process process = SystemConstructor.createProcess(classifier,
    parameterSet, path);
```

The API reference documentation (Javadoc) is included into the release and can be found on-line[2].

Accuracy Prediction

The accuracy prediction takes a dataset and a knowledge base and returns a result object that contains the predicted accuracies for all classifiers and the computed meta-features of the dataset.

```
predict(KnowledgeBase knowledgeBase, ExampleSet dataset) :
    RegressionResult
```

Evaluation

The optimization takes mainly three arguments: a knowledge base, a list of classifiers, the dataset, and the meta-features of the dataset. The result are the optimized parameter values for all considered classifiers.

```
evaluate(Iterable<Classifier> classifiers, Example queryMetaFeatures,
    KnowledgeBase knowledgeBase, ExampleSet dataset, int nThreads,
    EvaluationListener evaluationListener) : Map<String, ParameterSet>
```

System Construction

Finally, the system construction takes a classifier, its parameter values to use, and the location of the dataset. The result is a RapidMiner process.

```
createProcess(Classifier classifier, ParameterSet parameterSet, String
    datasetLocation) : Process
```

[2] http://www.dfki.uni-kl.de/~reif/mlwizard_javadoc/

Knowledge Base

Additionally, it is possible to modify the knowledge base, that serves as a basis for all meta-learning methods. It is easily possible to add new datasets and classifiers. If a new dataset is added, all existing classifiers will be evaluated on this dataset. Analogically, if a new classifier is added, it is evaluated on all included datasets. The resulting performance values and optimized parameter values are added to the knowledge base as well in order to improve further meta-learning runs.

The knowledge base included in the jar-File can be loaded very easily:

```
1 KnowledgeBase knowledgeBase = KnowledgeBase.read();
```

Meta-Features

The computation of the meta-features for a dataset can be accessed via the according RapidMiner Operator included in the MLWizard release. The operator can be used within the GUI and programmatically.

```
1 MetaFeaturesOperator metaFeaturesOperator =
      OperatorService.createOperator(MetaFeaturesOperator.class);
2 ExampleSet metaFeatures = metaFeaturesOperator.apply(dataset);
```

4 Command Line Interface

MLWizard can be used via the command line, as well. It will be started using the provided jar-file:

```
:$ java -cp /path/to/rapidminer.jar:/path/to/MLWizard.jar
    de.dfki.madm.mlwizard.cli.CommandLineInterface <dataset> <task>
    [outfile]
```

The first argument is the path to the dataset in XRFF-format that will be used as input. The second argument is the task that should be performed and can be one of the following:

wizard runs the complete wizard

metafeatures computes the meta-features

recommend recommends classifers

evaluate evaluates all classifiers

construct constructs the classification system

The third argument is optional and defines where the results are written to. The result depends on the task and might be meta-features, a parameter set or a RapidMiner pipeline. If no file is supplied, the results are written to stdout.

Sample Session

```
:$ java -cp rapidminer.jar:MLWizard.jar
    de.dfki.madm.mlwizard.cli.CommandLineInterface iris.xrff wizard
    result.xml
[ 0] k-NN                0.87
[ 1] Rule Induction      0.87
[ 2] Neural Net          0.87
[ 3] SVM                 0.86
[ 4] Decision Tree       0.84
[ 5] Naive Bayes         0.79
Enter comma separated numbers of classifiers you want to be evaluated:
0,1,2
[ 0] k-NN                0.97
          k-NN.weighted_vote            true
[ 1] Rule Induction      0.96
          Rule Induction.criterion      information_gain
[ 2] Neural Net          0.98
          Neural Net.decay              false
[ 3] SVM                 <not evaluated>
[ 4] Decision Tree       <not evaluated>
[ 5] Naive Bayes         <not evaluated>
Enter number of classifier you want a system for:
2
```

The created RapidMiner pipeline will be stored in result.xml and can be imported into Rapidminer.