Introductory Notes on RapidMiner

Introduction

RapidMiner is a tool for applying knowledge discovery techniques to a dataset including statistical analysis, visualisation, data pre-processing and data mining algorithms.

Rapid Miner implements over 600 techniques which can be applied to a dataset, each of which is called an operator. When analysing a dataset, you generally need to apply a sequence of operators to the dataset. A typical sequence could include: an operator to read in the dataset, a few operators to pre-process the data, operators to mine the data, and operators to evaluate the results. This sequence of operators is called a process. Processes can be saved to a repository and reused.

Each operator is implemented as a Java program. It has required inputs, may have additional optional inputs, and will have one or more outputs.

The outputs from one operator will be the inputs to the next operator in the process. This is represented visual in RapidMiner by lines connecting output ports in one operator with input ports in the next operator. Outputs from the final operator in the process must be connected to the output ports in the process window to view the process outputs. Note: automatic wiring on operators (i.e. connecting operators) can be enabled/disabled from an icon on the top left hand corner of the operator window.

Probably the most frustrating part of learning RapidMiner is understanding the required inputs for each operator, the available outputs from each operator, and how to provide the inputs needed to add an operator to a process.

This document aims to clarify the inputs and outputs of the more commonly use operators.

For an introduction to the Rapidminer user-interface, see user-interface-and-first-process video.

Reading in a dataset

Typically the first step of any process is to read in a dataset. The dataset can come from two sources, a rapidminer repository, or directly from source. Its easier to load the dataset into a repository in advance of using it in a process. This is because meta data is stored with the dataset which will include information such as attribute data types, and attribute roles (e.g. which attribute is the class label). This saves you the effort of defining this information each time you read the dataset into a new process. For an overview of how to load data into the repository, see data-import-and-repositories video

Attribute datatypes:

Data types for numeric attributes:

Number: Any numeric attribute. This can be further refined as:

Integer: Whole numbers

Real: Decimal values

Data types for non-numeric attributes:

Nominal: Any non-numeric attribute. This can be further refined as:

BiNominal: A nominal attribute with only two possible values

PolyNominal: A nominal attribute with more than two possible values

Retrieve operator

Inputs	Outputs	Required Parameters
None	**Out**: A dataset and it's meta data	Reference to a data source in a repository

Dragging a repository data source onto the process window adds the retrieve operator to the process, and automatically fills in the required parameter value.

Data Input Operators

Rapidminer provides a range of data import operator (see **import/data** in the operator window), each for a specific data source such as a database table, a flat file, an excel spreadsheet, a csv file, and many more.

Inputs	Outputs	Required Parameters
None	**Out:** A dataset	Varies depending on the operator used, but at a minimum you would need the location of the data source.

For an overview of how to read data from source, see import-flat-files video.

Inner operators

In addition to inputs and outputs, a selection of operators in RapidMiner require INNER OPERATORS, which means they need to have other operators nested within them. When such operators are added to the process window, they appear in a different colour (yellowish rather than purplish) and have a process symbol on the bottom right hand corner. To embed operators, click on the parent operator to open the internal process window(s) for the operator.

Test and Training dataset

Performance of a classifier is best assessed if the model is trained on a training dataset, and tested on a separate test dataset. This means the source dataset needs to be divided into a training and test dataset. Typically this is done using the X-Validation operator which implements cross validation.

X-Validation and Split validation

The following describes the X-Validation operator, but can also be applied to Split Validation. These operators have TWO inner sub processes. The first sub process works on the training dataset. The second sub process works on the test dataset.

Inputs	Outputs	Required Parameters
X-Validation – parent operator		
tra: Dataset	**tra:** the dataset including the predicted class label. **Mod:** The classification model **Ave:** The average performance of the model. There can be more than one ave port depending on the number of performance measures (operators) used to evaluate the model.	Number of validations: number of splits, and so also number of models generated. Default value is 10.
Training sub process		**nested operators**
tra: Training dataset.	**Mod:** The classification/regression model learned from the training dataset	Must include an operator from classification and regression which will generate a model from the training dataset.
Testing sub process		
mod: model learned from the dataset. **tes**: Test	**per**: Performance measure showing the average performance of models generated by each iteration of X-validation. Note: more than one vector of measures	Must include **two** operators: a Model Applier to apply the model (input parameter from training subprocess) to a test dataset **AND** a performance operator which will calculate the accuracy of the model based on its

dataset **thr**: see note below	can be outputted, donated by the **ave** output ports on the process window. **exa**: the dataset	performance on the test dataset. There are a number of performance operators under **Evaluation/Performance Measures**, depending on the type of classification task, and the data type of the class label.

Note: The sub process windows also include an optional **THR** port, which is to allow any object to be passed through these sub process to be used be an operator later in the process.

For examples of using an X-validation block, see decision-trees-basic video or neuralmarkettrends video

Classification Models

Rapid Miner implements a large variety of learners. Learners are restricted by the type of data they can model, and the type of class label they can predict. Table 1 summarises which operators can be used for each combination of attribute type and class label.

Table 1. Matching learners with attribute and label data types

Attributes	Class label	Learners
Numeric attributes only	Numeric class label	Regression Neural Network Support Vector Machine K-Nearest Neighbour Gaussian Process
Numeric attributes only	Binominal class label	Decision Tree Linear or Logistic regression Naïve Bayes Neural Network Rule Induction Support Vector Machine K-Nearest Neighbour Gaussian Process
Numeric attributes only	Polynominal class label	Decision Tree Naïve Bayes Neural Network Rule Induction Support Vector Machine K-Nearest Neighbour
Nominal attributes	Numeric class label	K-Nearest Neighbour
Nominal attributes	Binominal class label	Decision trees (including CHAID) Naïve Bayes Rule induction K-Nearest Neighbour
Nominal attributes	Polynominal class label	Decision trees (including CHAID) Naïve Bayes Rules induction K-Nearest Neighbour

Note: Rapidminer provides a handy dialog box to help users find the right operator for a specific scenario under **File / New Operator.**

Learner inputs and outputs:

Inputs	Outputs
All learners:	
Tra: Training dataset – this can be the full dataset, or a subset generated from X-Validation or Simple Validation	**Mod:** Model – the trained model which can predict the class variable **Exa:** Example set with the predicted class label added. Can also include a confidence level for each prediction.
Additional outputs for SVM's and Regression algorithms	
	Wei: a weighting for each attributed based on how significant it was in predicting the class label **Est:** estimated accuracy of the model

Performance operators

Performance operators evaluate the performance of a learned model by comparing the predicted class label with the actual class label in a test data set.

There are a variety of measures one can use to evaluate the performance of a model, depending on the data type of the class label and learner used. Some of these are summarised in Table 2.

Table 2. Performance measures

Measure	Description	Type of class label supported
Accuracy	Percentage of correct predictions.	Binominal and polynomial class label
Precision	Number of correct positive predictions divided by the total number of positive predictions, i.e. what percentage of positive predictions were correct? (TP / TP+FP)	Binominal class label
Recall	The number of correct positive predictions divided by the total number of positive examples, i.e. what percentage of the positive class was predicted correctly. (TP / TP+FN)	Binominal class label
AUC	Area under the ROC curve. The confidence measure of each prediction is needed to plot the curve (not all learners provide this).	Binominal class label
F measure	Combines precision and recall, giving a weighted (harmonic) mean of precision	Binominal class label

	and recall.	
Kappa statistic	Compares the predicted values with what would be expected if predictions were done by random guessing.	Polynomial class label
Mean squared error.	Calculated based on the average difference between predicted and actual values.	Numeric class label
Root mean squared error.	Square root of the mean squared error above	Numeric class label
Correlation	Measures the dependency between actual and predicted values, i.e. the extent to which one can be predicted from the other, and so is a similarity measure between the two.	Numeric class label
Absolute error	The magnitude of the difference between the actual and predicted values.	Numeric class label
Relative error	As above, but divided by the magnitude of the actual value, and so is scaled.	Numeric class label
Spearman rho	A measure of the linear relationship between the actual and predicted values	Numeric class label
Kendall tau	A measure of the extent to which two lists are in the same order, i.e. if the actual value in row **n** is less than the value in row **n-1**, can the same be said of the predicted value.	Numeric class label

The generic performance operator located under **evaluation / performance measure** will select the appropriate measures to use based on the data type of the class label in the dataset.

Alternatively, performance operators specialised to a particular type of class label, or learning algorithm, can be found under **evaluation / performance measure / classification and Regression.**

Performance inputs and outputs:

Inputs	**Outputs**
All performance operators	
Lab: A dataset which includes a class label and a predicted class label.	**per:** A performance vector, which is a list of performance measures. This output should be connected to an **avg** output port on the process window. **Exa:** The data set that was passed in to the operator.
Additional inputs for some of the performance operators	
Per: performance – this is an optional input which accepts a performance vector (i.e. a list of performance measures) which can have been generated by selected learners, or another performance operator.	

Performance inputs and outputs:

Compare ROCs, an operator found under **evaluation / visual evaluation** is useful for comparing the models generated by a number of learners. The learners are added is inner operators to the **compare ROC** operator. It will generate an ROC curve for any learner that outputs confidence levels with the predicted label (e.g. nueral net and naïve bayes)

Inputs	Outputs	Inner operators
exa: a dataset with a binominal class label	**exa:** The data set that was passed in to the operator. **roc:** a graph showing the roc curve for each inner operator	Any learner that can predict a binominal class label, and outputs confidence levels for each predicted value.

Clustering Models

Rapid miner implements a large variety of clustering algorithms. Algorithms are restricted by the type of data they can model, and the type of class label they can predict. Some of the learners, and their restrictions, are summarised in Table 3.

Table 3. Matching clustering algorithms with data types

Attributes	Learners
Numeric attributes only	DBScan K-Means, K-Medoids Support Vector Clustering
Nominal and numeric attributes	DBScan K-Medoids
Nominal attributes only	DBScan K-Medoids

Note: The operator '**Nominal to Numeric'** can convert nominal attributes to numeric attributes. However, be careful how you interpret the results, as the algorithm may assume an ordering to these numeric values which is not accurate.