UNIVERSITÄT MANNHEIM

Data and Web Science Group
Prof. Dr. Christian Bizer
B6 – B1.15
68159 Mannheim

Data Mining – FSS 2016

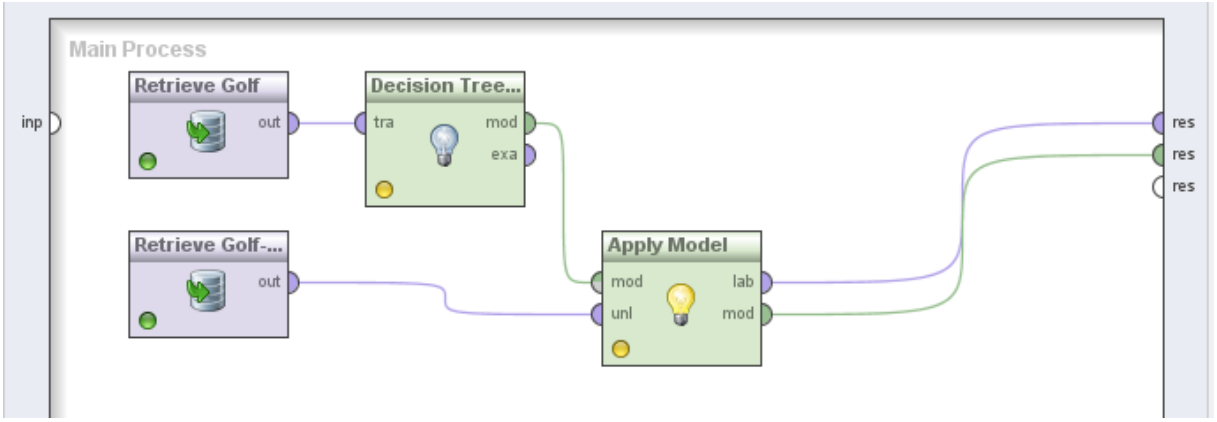
Exercise 3: Classifcation

3.1. Should we play golf?

The *Golf data* set is one of the examples that is delivered together with RapidMiner. The data set models different aspects of the weather (outlook, temperature, humidity, forecast) that are relevant for deciding whether one should play golf or not.

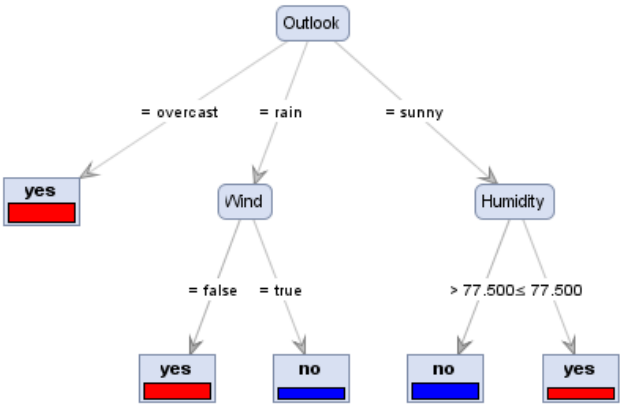
1. Learn a decision tree from the Golf data set (Operator: *Decision Tree*). Use this tree to classify the examples in the Golf-Testset, which is also delivered together with RapidMiner (Operator: *Apply Model*).

   Solution: Add a retrieve operator for each data set (training and test). Link the training data to the operator decision tree. Link the output of the decision tree to the apply model operator. Link the output of the retrieved test data set also to the apply model operator. Compare the prediction (Prediction (Play)) with the column Play.



   Conclusion: The test set is classified by the learned model with 5 wrong classifications and 9 correct classifications.

   The created decision tree looks like:



2. Evaluate the performance of your model by adding a *Performance (Classification)* operator to your process. Examine the confusion matrix. What is the accuracy of your classifier?

   Solution: Link the performance (classification) operator between the output from 3-1-1 and the apply model operator. Accuracy can be found at the Table / Plot View of the Performance Vector

accuracy: 64.29%

	true no	true yes	class precision
pred. no	3	3	50.00%
pred. yes	2	6	75.00%
class recall	60.00%	66.67%	

   Conclusion: The accuracy is about 64.29%

3. Does a k-nearest-neighbor classifier work better for this task? To find out, replace the *Decision Tree* operator with a *K-NN* operator and check how the accuracy of your classifier changes. Do different values of k improve the performance?

   Solution: Accuracy can be found at the Table / Plot View of the Performance Vector

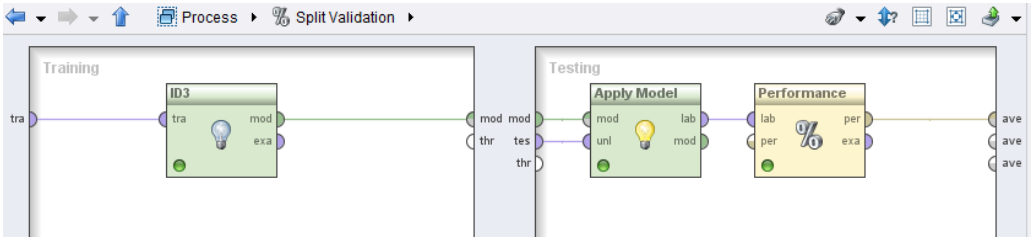
   Conclusion: Different k values change the accuracy of the classifier.

k	accuracy	comment
1	71.43 %	Equally distributed classification errors
2	35.71 %	Classification errors mainly for “yes” - class
3	64.29 %	Classification errors mainly for “no” - class
4	57.14 %	Equally distributed classification errors
5	64.29 %	Almost equally distributed classification errors

3.2. Learning a classifier for the Iris Data Set

1. Let’s try the ID3 tree building algorithm first. Build a process that (1) discretizes all attributes of the Iris data set by frequency into three bins. (2) Afterwards, the process should use the *Split Validation* operator (split ratio=0.7, stratified sampling) to generate a training and test data set. (3) As inner operator of the split validation, the process should use the *ID3* operator to learn a decision tree and the *Performance (Classification)* operator to evaluate the accuracy of the learned model.

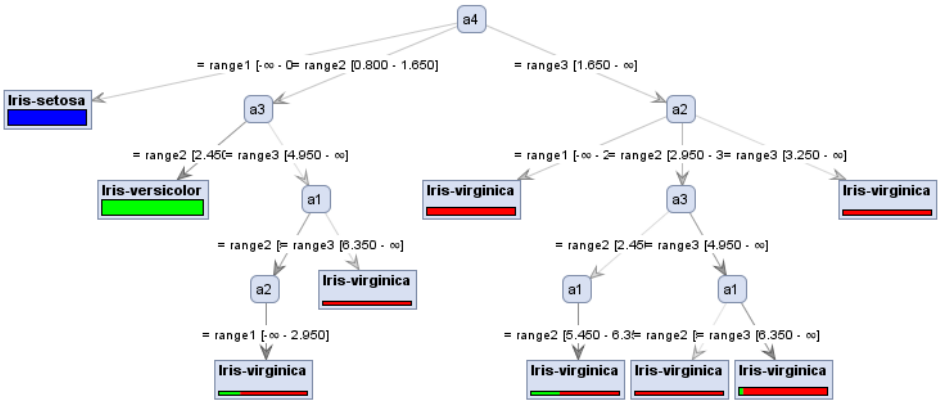
Solution: Use the *Discretize by Frequency* Operator. Add the *Split Validation* Operator. To edit the inner operators select the blue button on the lower right corner of the *Split Validation* Operator. Link the "ave" port to a result port.



Conclusion: Performance reached is about 97.78% where the classification errors are made in the group of versicolor. Note that the performance is calculated on a model learned from the splitted training data, but the tree that is shown is based on the full data set (that's why you get varying performance values without a local random seed but always the same tree).

Note: the performance numbers are obtained using "use local random seed" option, local random seed=1992, compatibility level=6.3.000. Without or with different local seed, and with different compatibility level (in case you haven't installed the recent updates) you might get different performance results.

The created decision tree looks like:

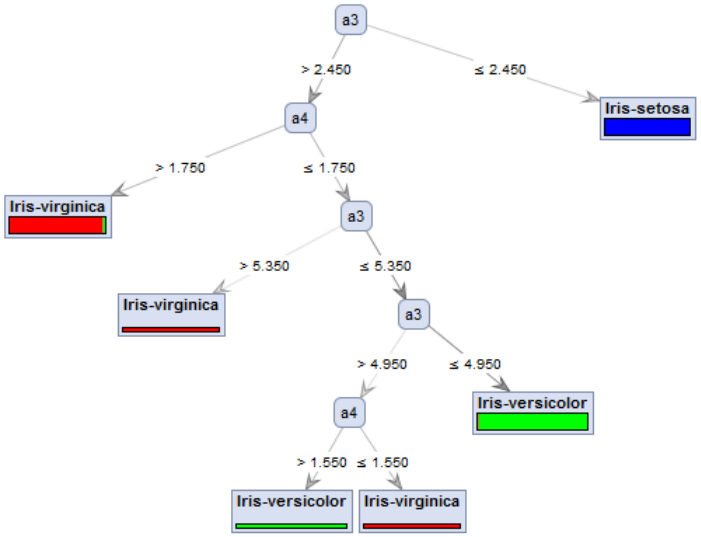


2. Remove the discretization operator and change the ID3 operator into RapidMiner's standard *DecisionTree* building operator. Run the process again. Does the accuracy change? Compare the complexity of the two models. Which model should be preferred according to Occam's razor?

   Solution: Occam's Razor: Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.

   Conclusion: The accuracy is 95.56% of this process where most of the classification errors are made in the group of versicolor.

   The created decision tree looks like:



Applying the rule of Occam's razor it would be better to use the simple decision tree instead of ID3.

3. Try a k-nearest-neighbor classifier on the problem. Does it perform better?

   Solution: Replace the *Decision Tree* Operator by a *k-NN* Operator. Accuracy can be found at the Table / Plot View of the Performance Vector

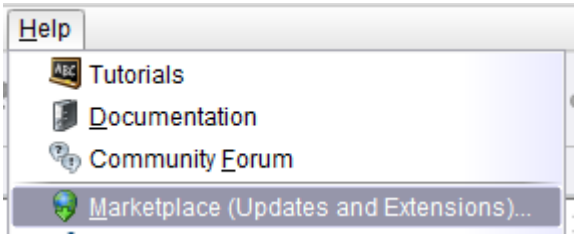
   Conclusion: Testing different k values the accuracy improves up to 97.78%.

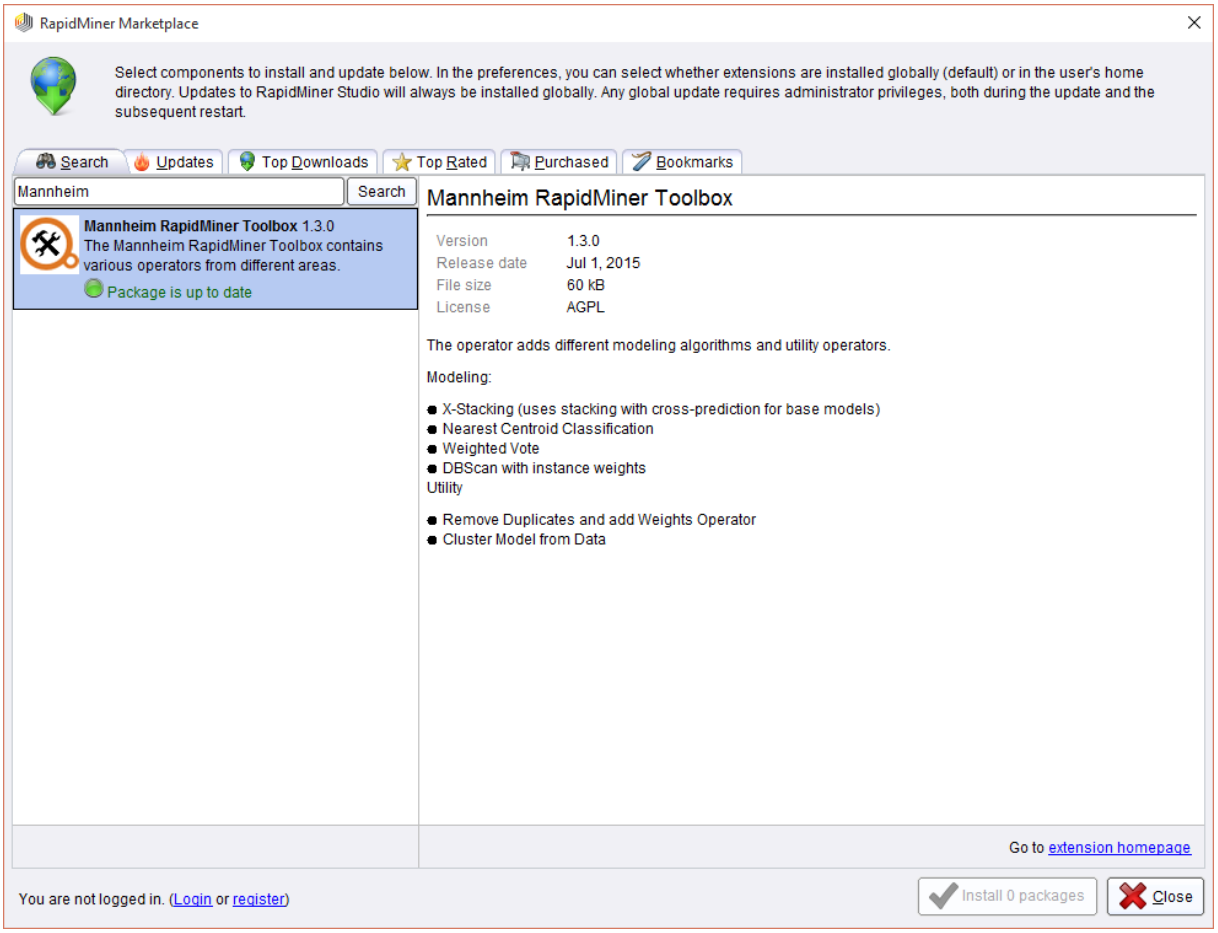
3.3 More Classification

In the lecture, you learned about the Nearest Centroid Classifier. For this classifier, RapidMiner does not provide you with an operator. So you have to install the "Mannheim RapidMiner Toolbox" in order to use this classifier.

1. Install the"Mannheim RapidMinerToolbox" from the Marketplace ("Help" -> "Marketplace")

   Solution: Install the Extension in RapidMiner





2. Compare kNN and Nearest Centroid Classification using the "Weighting" dataset from the RapidMiner Samples Repository.

   Solution: Build a Process that uses the "k-NN" and "Nearest Centroid Classification" operators and compare their results

   Conclusion: The performance of the Nearest Centroid Classification is better than that for k-NN for this particular dataset.