Data Mining Methods Week 4 RapidMiner Lab - Classification Using Decision Trees

Purposes:

- Practice Classification using Decision Tree
- Understand how to interpret and apply Decision Tree models.

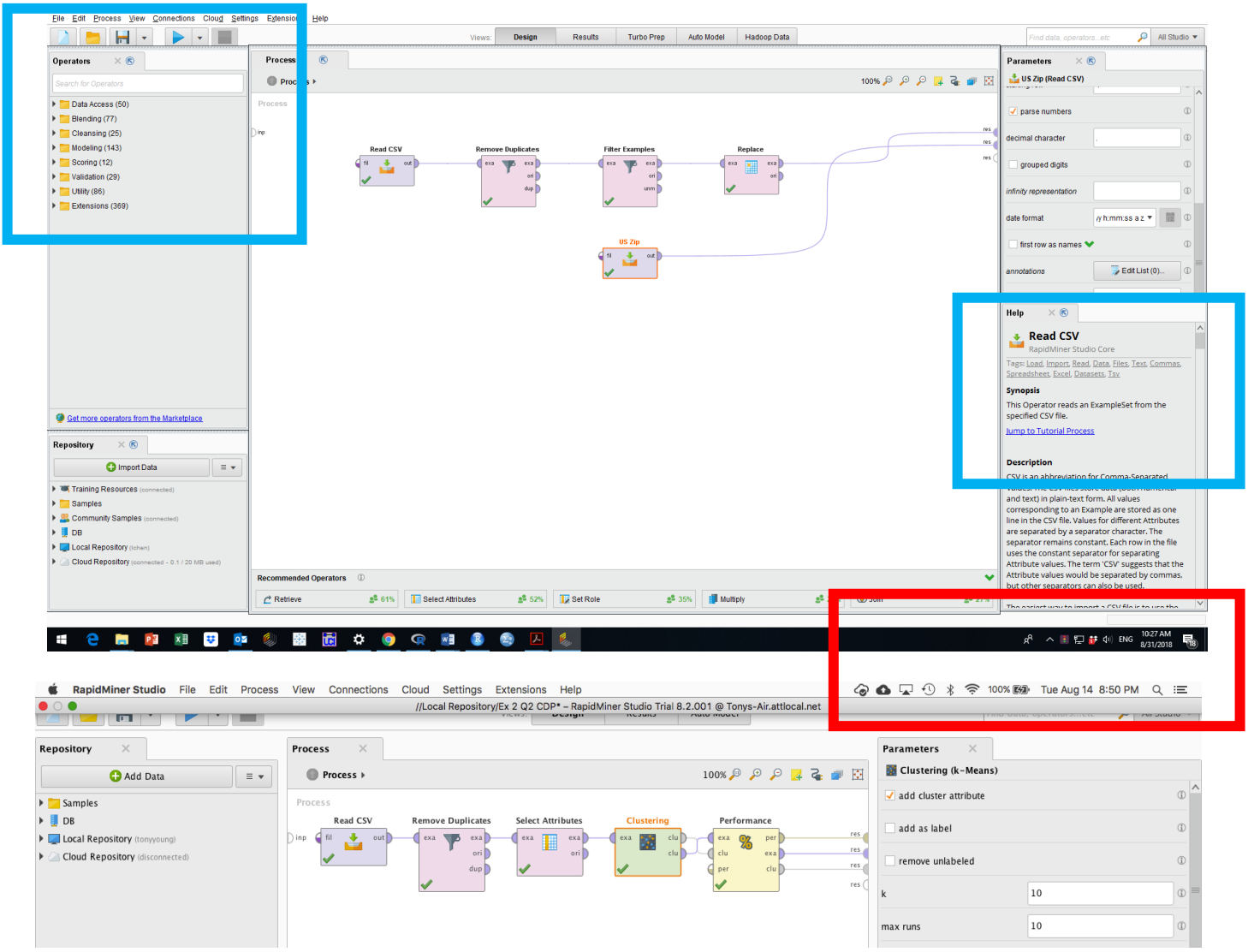
Dataset: Please download the datasets titled "iris_train.csv" and "iris_predict.csv" from the WT Class to use for this lab session. The data includes information about three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of sepal and petal, in centimeters.

Requirements: Follow the instruction, take the required screenshots with date and time (see the examples highlighted in red rectangle below), and answer all the questions. Sharing your queries, screenshots, or answers with other students is considered as cheating, which will be reported to the university authority.

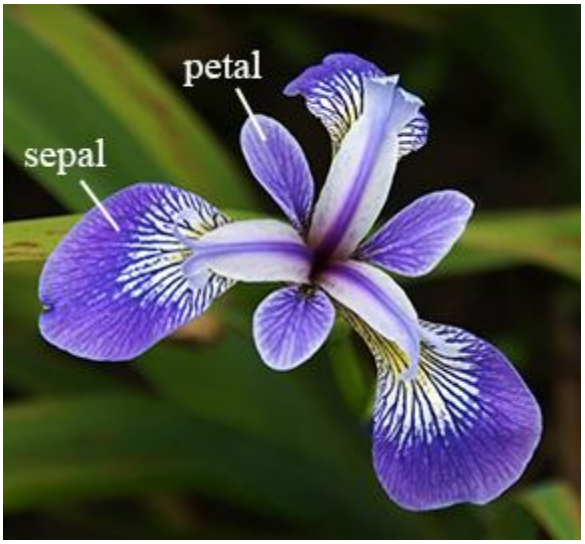
Note: 1. If you cannot find an operator, please use the search function in the operator panel.

2. Use the help information at the right bottom. By default, RapidMiner shows the help information. If you don't see it, go to View->Show Panel->Help.

3. In order to be consistent across all students, please never change random seed in RapidMiner.



1. Creating A Decision Tree Using The Training Dataset:



First of all, please remember that when doing classification analysis, our goal is to predict the target variable values that are missing in a dataset. Let's call this dataset the prediction dataset. To accomplish this, we will use an existing dataset with complete information (i.e. target variable values are already given). Let's call this dataset the training dataset (since we are "training" a model). We are going to practice this in a classic classification case.

1.1. Open both datasets using a text editor (or Excel) and notice that iris_predict.csv is missing "Species_name" information. Our goal is to predict these names.

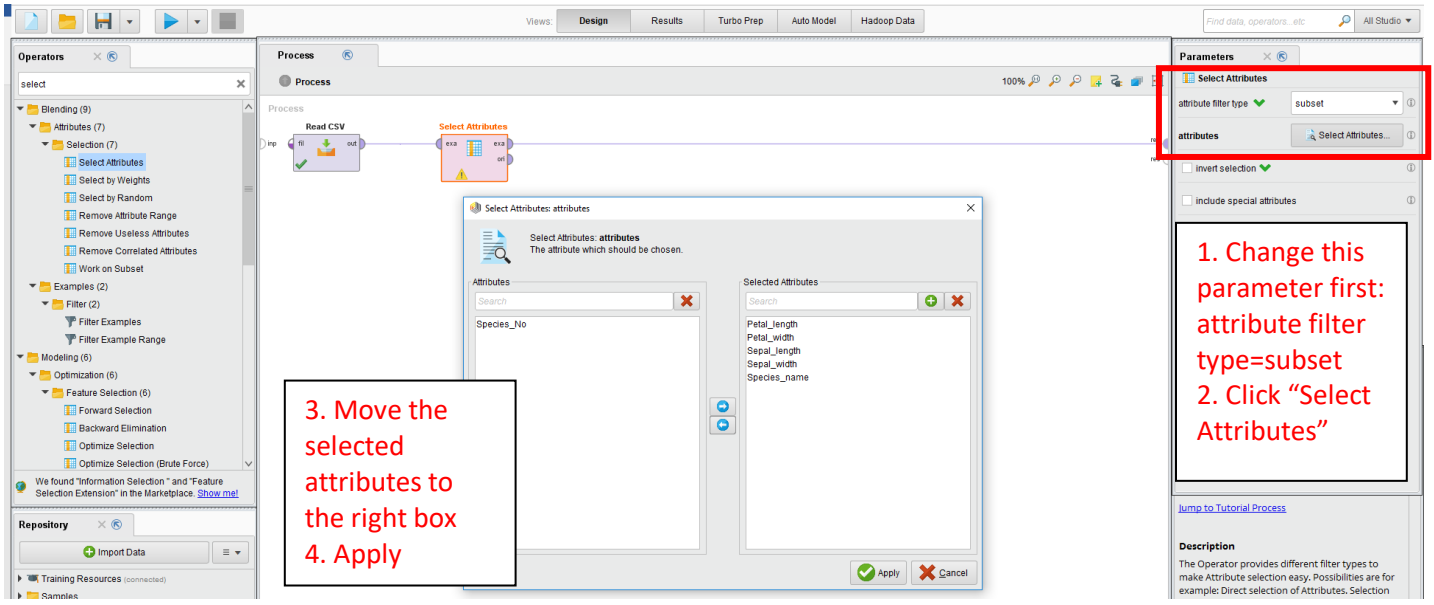
1.2. First, import the Iris_train dataset into RapidMiner. Use the Import Configuration Wizard.

- Step 1: Select the data location
- Step 2: Specify your data format [Use the default settings if no problems are identified by RM]
- Step 3: Format your Columns [Make sure that Attribute types in Step 3 make sense to you; especially, Species_No is identified as integer by RM, is it right? If not, please change it via Change Type]

1.3. Run the process and observe the results in the results pane. There are two attributes that provide the same information. What are they?

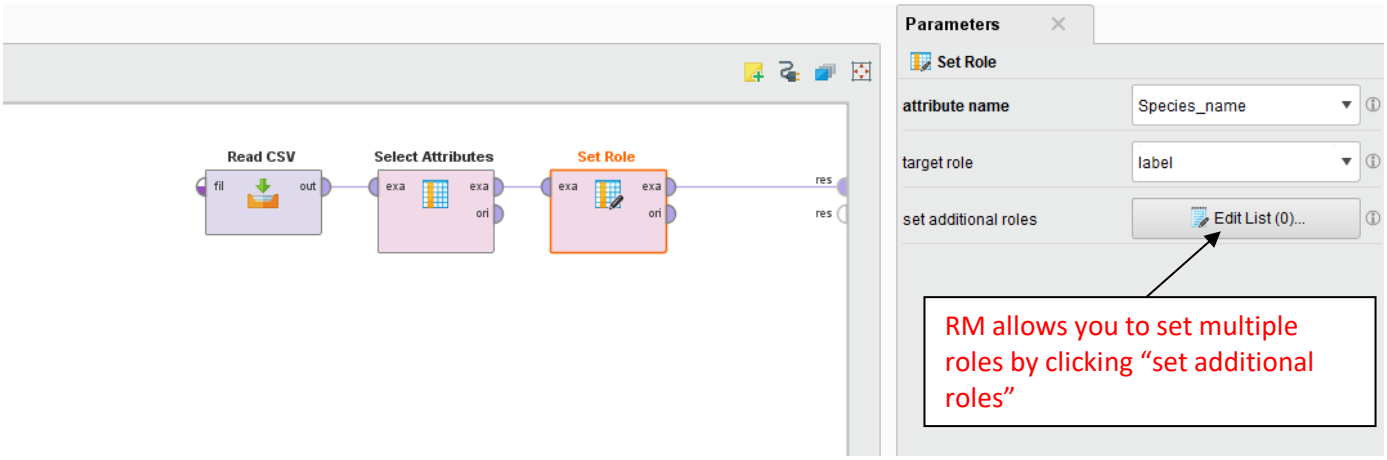
1.4. Consider the fact that we are trying to build a model to predict "Species_name" and we have an attribute ("Species_no") that is perfectly correlated with the target attribute. Think about what can be a potential problem here. Open the "iris_predict" dataset and notice that "Species_no" does not exist. Think about the potential problem here and how you can overcome it.

1.5. Therefore, we need to remove the "Species_no" attribute from the training dataset by using the "Select Attributes" operator.

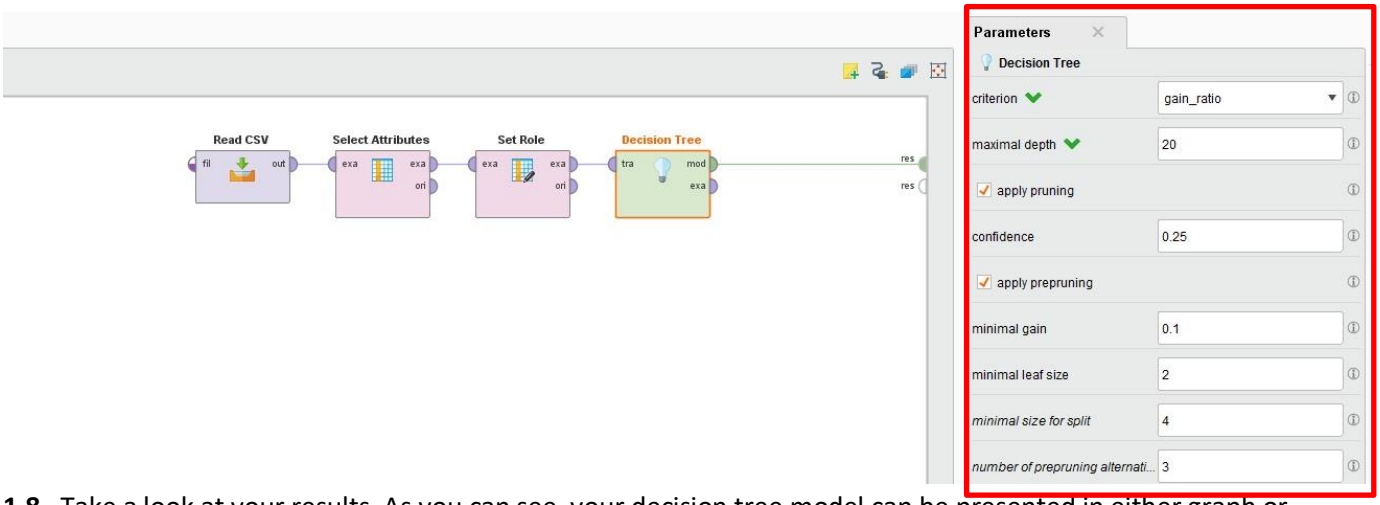


1.6. Then, add a "Set Role" operator to define the target class as "Species_name". The screenshot is given below.

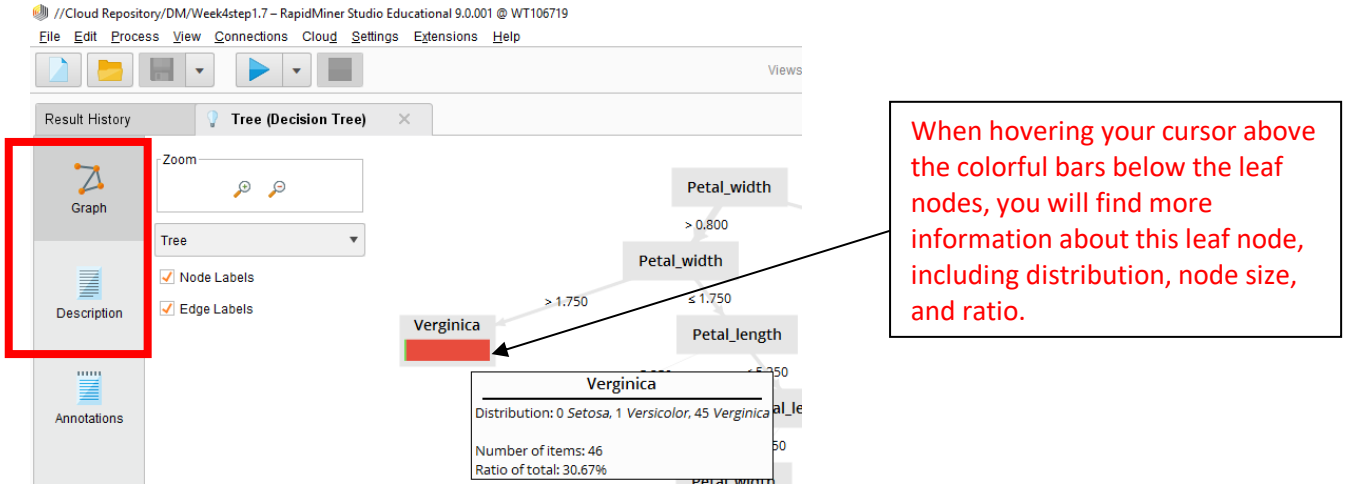
Note that doing this makes sure that RapidMiner understands the class definitions correctly when making the classification. Run this model to make sure that your process is working. Use the "help information" to understand why we use "Set role".

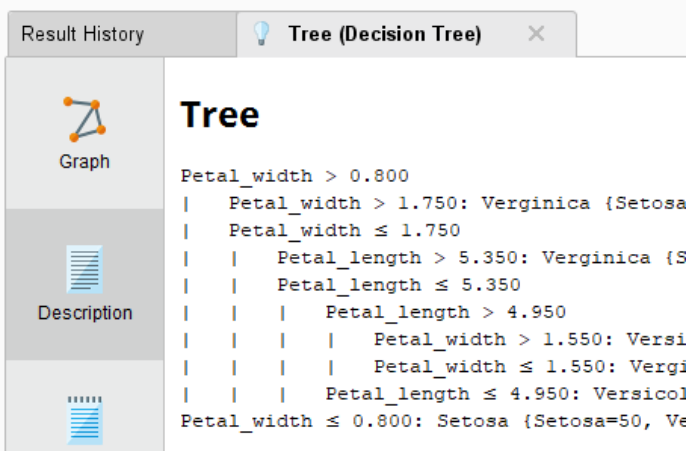


1.7. Now, we'll create a decision tree from this dataset using the "Decision Tree" operator, which uses C4.5 algorithm. Run it first time with this operator with the following parameters.



1.8. Take a look at your results. As you can see, your decision tree model can be presented in either graph or description. Please answer the following questions. Take a screenshot of the decision tree graph with date and time (Screenshot 1).





The second presentation of your decision tree model

For example, if Petal_width is not greater than 0.800, the flower is classified as Setosa.

1.8.1. Observe the decision tree graph, identify the root, interior and leaf nodes. In addition to the root node, how many other split nodes does this decision tree have? How many leaf nodes?

1.8.2. How many leaf nodes are **completely** homogeneous (the ones with a single color)? Is the leaf node with the biggest size (the highest number of items) completely homogeneous?

1.8.3. Please compute the Gini index for the leaf node with 46 flowers (this node: Verginica {Setosa=0, Versicolor=1, Verginica=45}). Round your answers to the third decimal place (e.g., 0.234). You may refer to Slides 33-36 in our PPT for the computation.

1.8.4. Based on the decision tree model, which one plays a determining role in classifying a flower, petal or sepal?

1.8.5. Given the following two flowers, use the decision tree to predict their type manually.

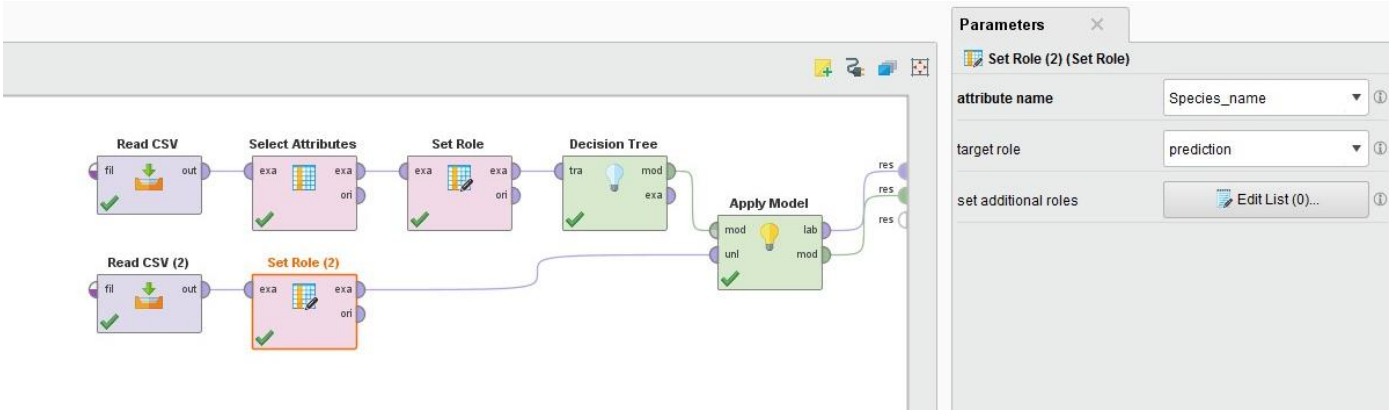
Flower number	Petal Width	Petal Length	Sepal width	Sepal Length	Species Name
A	1.3	4.6	3.4	4.8	
B	0.6	2.5	2.4	5.7	

1.9. An interesting observation: Run the same process again without removing the "Species_no" attribute. You can do this by just deleting the "Select Attributes" operator. Observe the results and confirm your answer to Question 1.4. Then, add back the "Select Attributes" operator to your process.

Congratulation! You've built a decision tree model from the existing dataset. Now, we'll use this model to predict the values in the other dataset (prediction dataset).

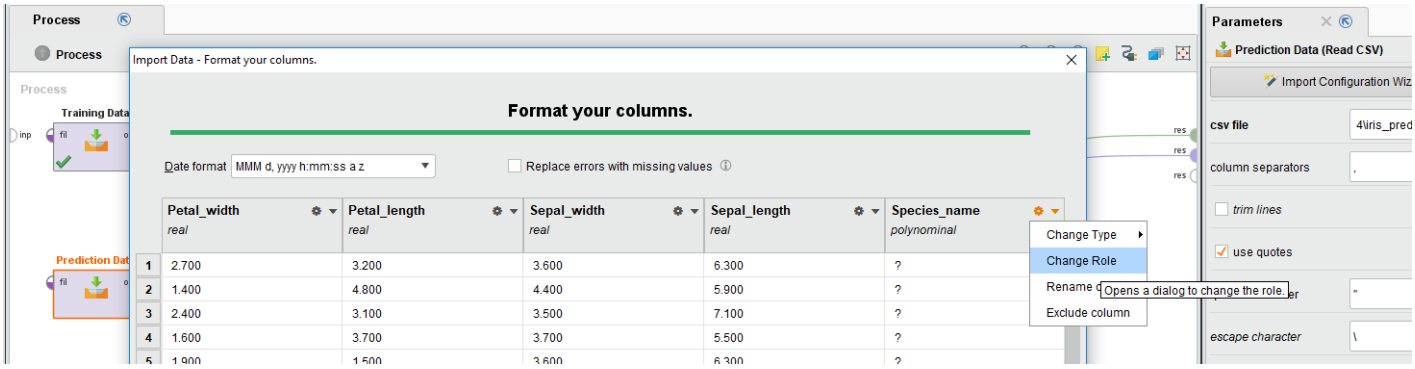
2. Prediction with Decision Trees

2.1. Create the RapidMiner process shown in the screenshot below.

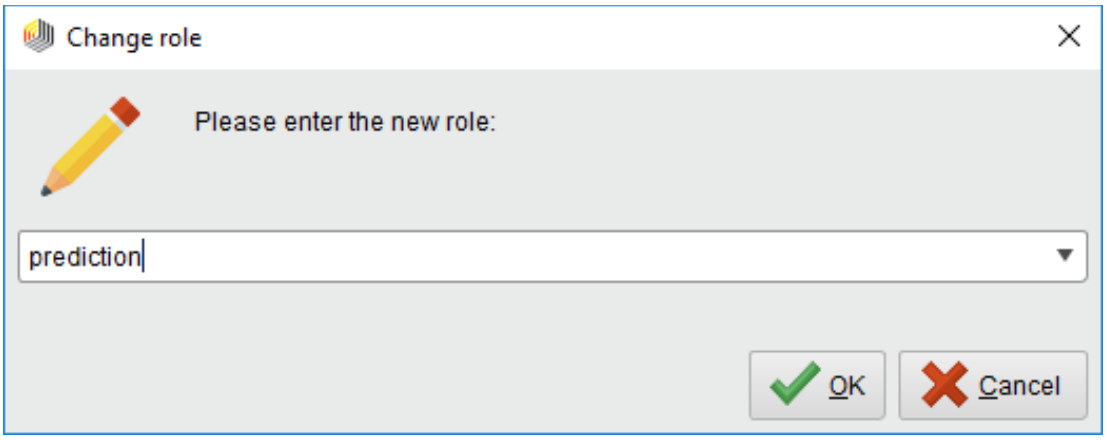


The top part of the model is almost the same from the previous question (you also need to add an Apply Model operator). However, notice that we are now reading both training and prediction datasets. Read CSV (2) should import "iris_predict.csv" and Set Role (2) should have been configured as name: Species_name and target role: prediction (remember that we are trying to predict the missing species_name values in the iris_predict.csv file). Prediction dataset includes 19 observation with missing species names. Notice that "lab" port in the "Apply Model" should be connected to the "res" port.

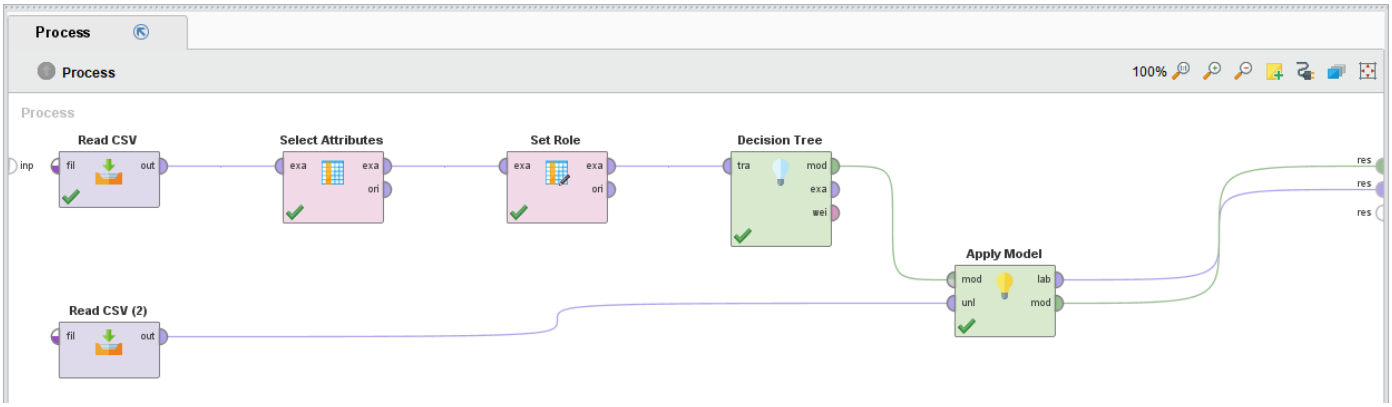
An alternative method to set the role of your target attribute is to change the role of species_name as below at the third step "Format your columns" in the Import Configuration Wizard". Doing so, you do not need the Set Role operator.



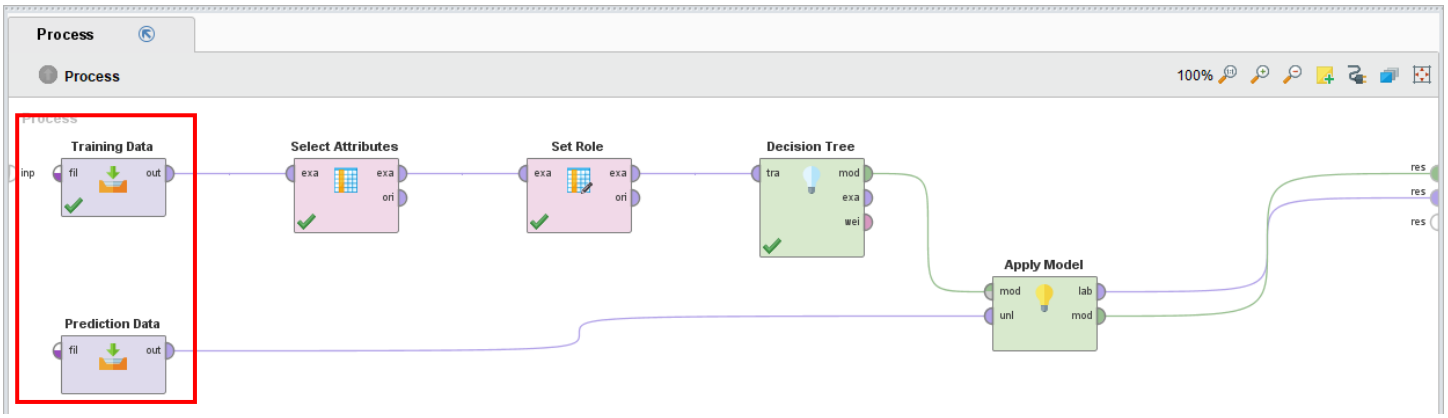
Type prediction



Doing so, your new DM process no longer needs the Set Role operator.



If you do not like two operators with the same name Read CSV, you can change their names as below to distinguish between the training and prediction datasets.

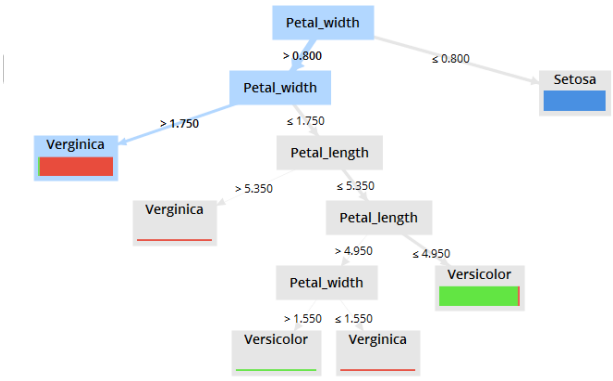


2.2. Run this process. In the results, switch to Data View and check out the Prediction column. Please take a screenshot of prediction results for the 19 observations with date and time (Screenshot 2).

2.2.1. In the prediction result page, the missing value in Species_name is replaced by the prediction result

2.2.2. In addition, three more columns are generated: confidence(Setosa), confidence(Versicolor), and confidence (Verginica). What do they mean? They indicate how confident you are to say that a particular flower is predicted as any one species. For example, the first flower is described as below.

Petal_width	Petal_length	Sepal_width	Sepal_length	Species_name
2.7	3.2	3.6	6.3	



Based on the decision tree, it will be labelled as Verginica, but the Verginica is not a pure class.

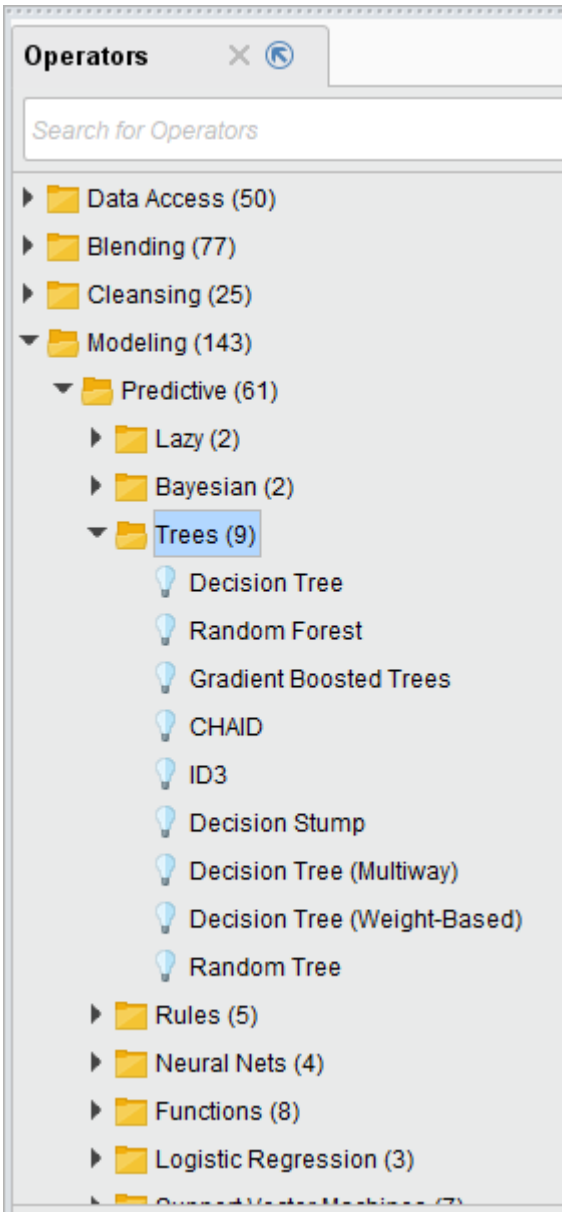
Verginica {Setosa=0, Versicolor=1, Verginica=45}

45 out of 46 flowers are actually Verginica, so the conditional probability that a flower with Petal_width>1.750 will be classified as a Verginica is 45/46=0.978. This probability is the confidence(Verginica) in RapidMiner. RapidMiner will assign the class with the highest confidence for a new flower. Therefore, the first flower is assigned as Verginica because confidence(Verginica) is the highest among all the three confidence values.

2.2.3. Among the 19 new followers, how many of them are classified as Setosa, Versicolor, and Verginica, respectively?

Advanced Materials

When you take a look at trees under Modeling ->Predictive, you will find nine operators about decision tree models. Five of them are introduce as below. Please try them in RM and compare the results.



Decision Tree (C4.5)

C4.5 is the default algorithm of Decision Tree in RM. It is an algorithm used to generate a decision tree developed by Ross Quinlan (1994). C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. C5.0 is a successor of C4.5 algorithm.

Random Forest

The Random Forest Operator creates several random trees on different Example subsets. The resulting model is based on voting of all these trees. Due to this difference, it is less prone to overtraining (i.e., overfitting).

CHAID

The CHAID Operator provides a pruned decision tree that uses chi-squared based criterion instead of information gain or gain ratio criteria. This Operator cannot be applied on ExampleSets with numerical Attributes but only nominal Attributes.

ID3

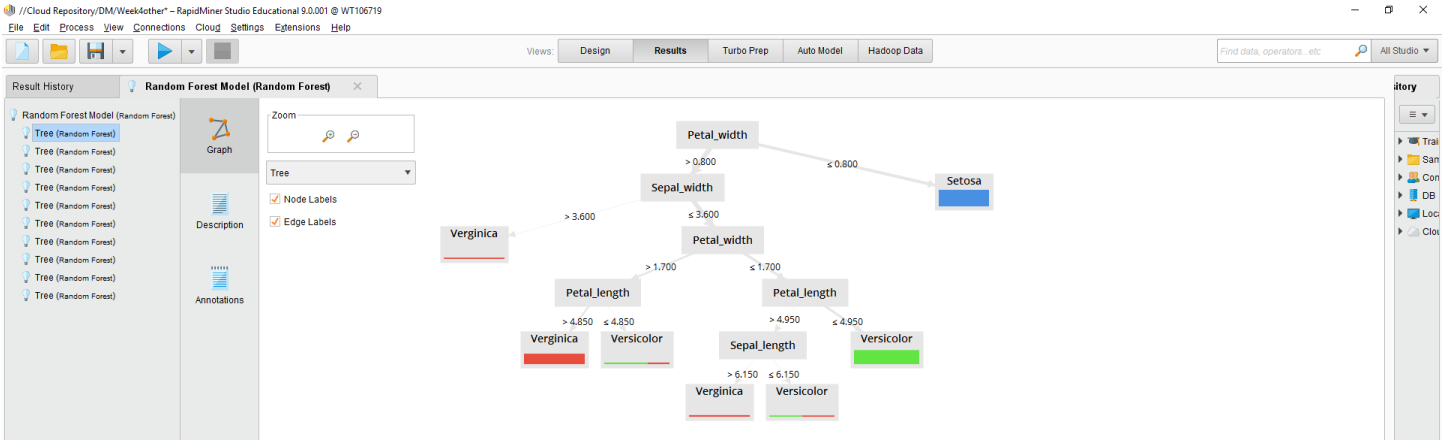
The ID3 (Iterative Dichotomiser) Operator provides a basic implementation of unpruned decision tree. It only works with ExampleSets with **nominal Attributes.** ID3 (Iterative Dichotomiser 3) is an algorithm used to generate a decision tree invented by Ross Quinlan. ID3 is the precursor to the C4.5 algorithm. Very simply, ID3 builds a decision tree from a fixed set of examples. The resulting tree is used to classify future samples. ID3 uses feature selection heuristic to help it decide which attribute goes into a decision node. The required heuristic can be selected by the criterion parameter. The ID3 algorithm can be summarized as follows:

- Take all unused attributes and calculate their selection criterion (e.g. information gain);
- Choose the attribute for which the selection criterion has the best value (e.g. minimum entropy or maximum information gain);
- Make node containing that attribute

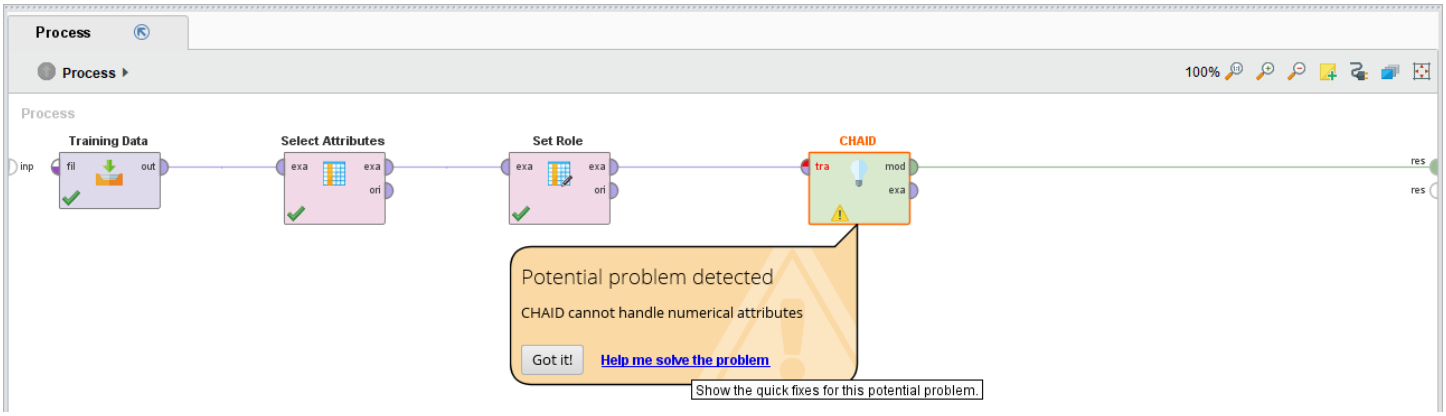
Decision Stump

The Decision Stump operator is used for generating a decision tree with only one single split. The resulting tree can be used for classifying unseen examples. This operator can be very efficient when boosted with operators like the AdaBoost operator. The examples of the given ExampleSet have several attributes and every example belongs to a class (like yes or no). The leaf nodes of a decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a possible value of the attribute. For more information about decision trees, please study the Decision Tree operator.

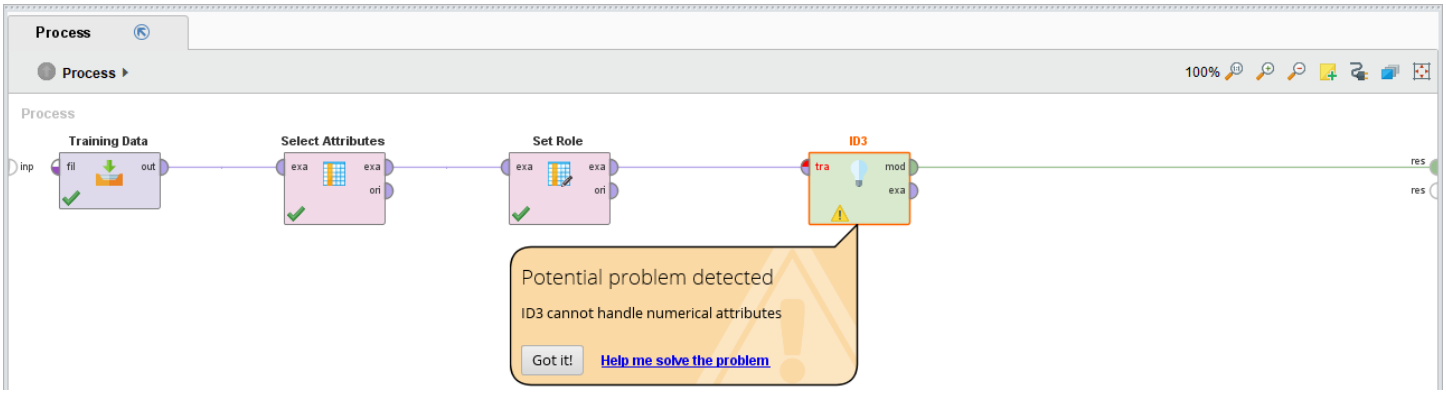
Random forest generates ten different decision models by using the default setting (but you can change the number of trees)



CHAID provides you with an error because we have numerical attributes (all the four predicting attributes are numerical)



ID3 has the same problem.



Decision Stump generates a decision tree with only one single split node.