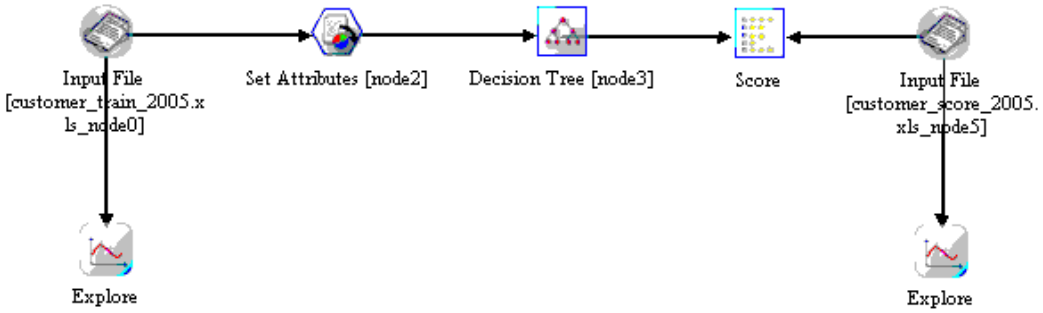
AlphaMiner Example sessions

As example of a traditional data mining program (knowledge discovering) we use the decision tree algorithm as available in the data mining program *AlphaMiner*. See the appendix attached to this exercises for more details about the implemented algorithm.The Alpha Miner software, the User Guide, and example data sets are available on the BPM-process-mining web-page. Install the software and use the short ***User Guide*** and the example in data in the directory C:\Program Files\AlphaMiner\data\telecom to get an idea about the working of the software. First click the + sign before the Telecommunication Co. map and double click the "Customer profiling 2005" case to get the following mining model:

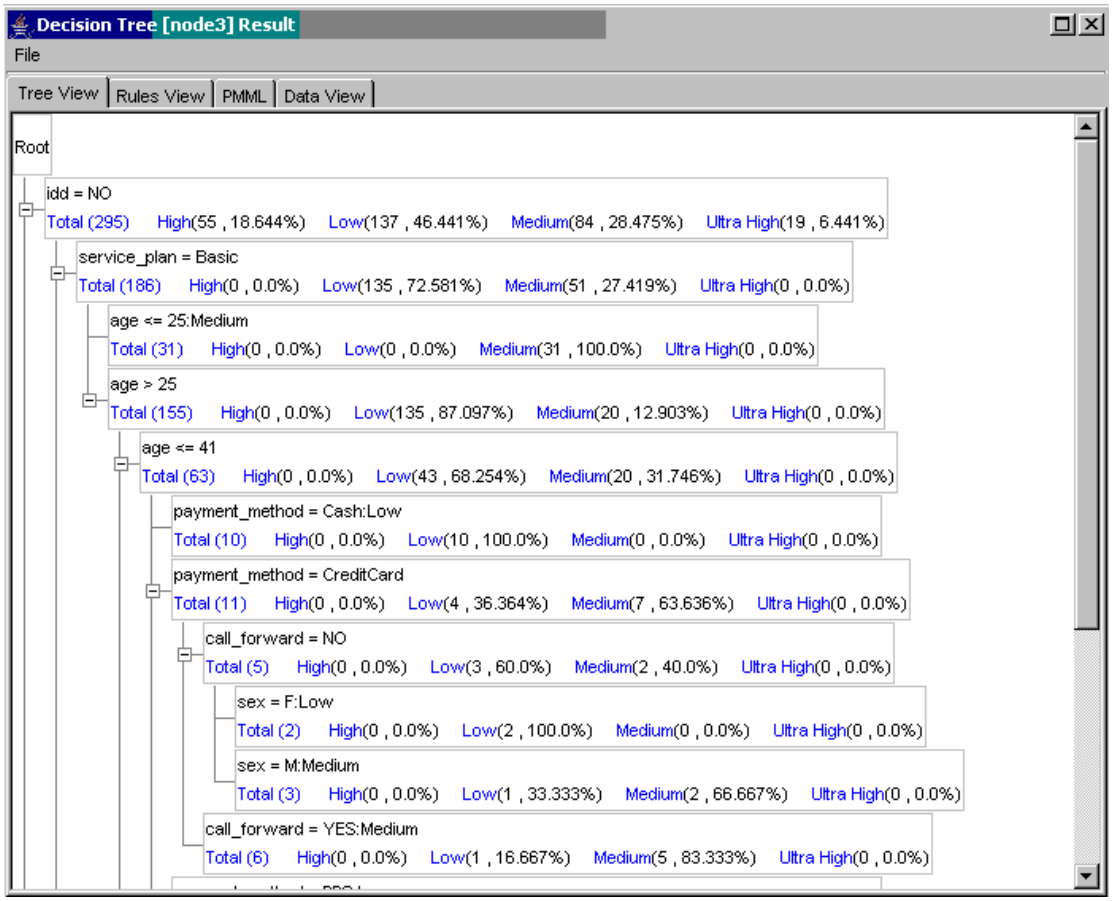


On my machine there were problems with the input files. If this is the case, please try the following. Right click the Input File node, use the RESET option, use the BROWS-option and select the data>telecom>customer_train_2005.exl file to connect this file as an input file. The last thing to do is selecting the Feb 2005 worksheet. Try the same for the other input file (data>telecom>customer_test_2005.exl).

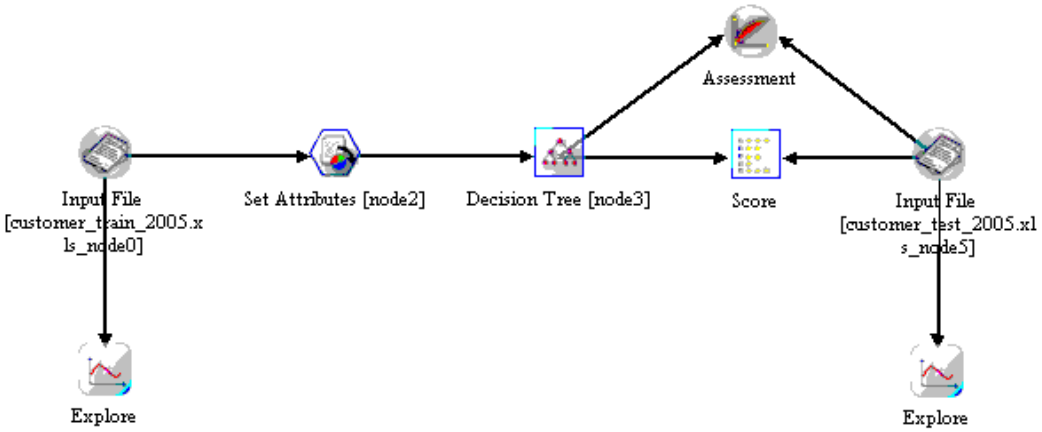
We can use an Explore node to get an idea of the properties of the customer_train_2005.xls file (right click the Explore node). In the Set Attributes node we can define the target attributes and the attributes that are used during the construction of the tree.

In this example the dataset customer_train_2005.xls is used for the construction of an Decision Tree. Use the Run option (right click the Decision Tree Node) to build the tree. If the building of the decision tree is successful the node becomes green. Use the View –option to look to the decision tree. There are four views available: (i) tree view, (ii) the rule view, (iii) a PMML view, and (iv) a data view. In the data view the classification for each element in the test file is given.

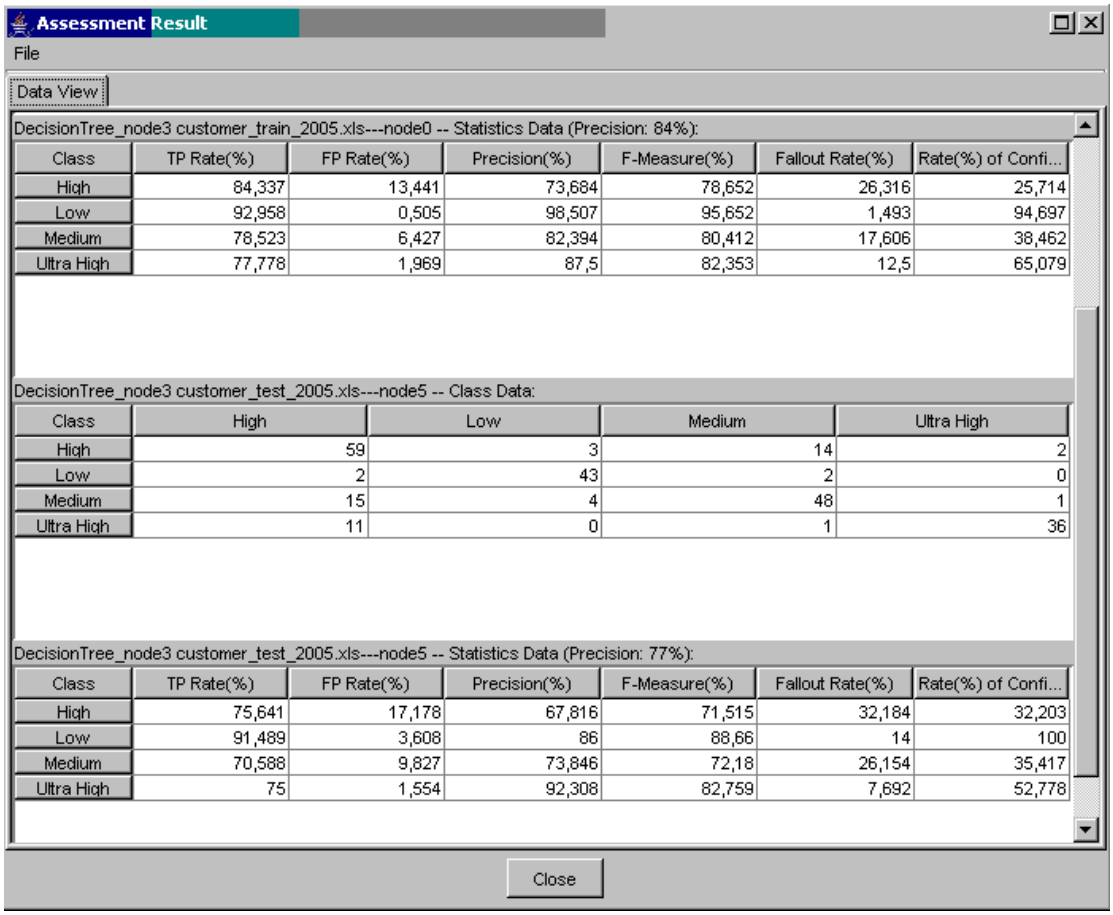
Below, the first nodes in the Tree view are expanded by clicking the + boxes. Some of the leaves are based on only 2 cases. That means that there is the dangerous for overfitting of the data. We can check this by looking to the performance on new data (not used for the building of the decision tree).



To get this information we will extend our model will an Assessment Node. Fist save and close the current model an use the  option to create a copy of the first model. Add an Assessment node and connections to get the following model:



The customer_test_2005.xls file is used to test the quality of the mined model on new data. Normally the performance of the mined model op the train data is a bit better than the performance on test data. In this case 84% of the training material is correct classified (precision), and 77% of the test material. Run the assessment node and use the View-option to get this information.



Try to update the Decision Tree parameters to get a better performance on the test material (a performance of 80% or more is possible). What is the optimal setting and the performance on train and learning material for this parameter setting. Explain the better results.

The HOSPITPL data set contains information about patients (diagnoses, gender, age, insurance information) and an indication of the time the have spend in the hospital. The goal is to make a model that can be used to estimate the number of hospital days for new patients.

Analogue to the previous example use the excel dataset HOSPITPL750.EXL and the default parameter setting to build a decision tree for hospital days prediction (target value). **Try to answer the following questions:**

a. **How many examples of HOSPITPL750.EXL (the learning material) are correct classified?**
b. **How many examples of HOSPITPL250.EXL (the test material) are correct classified?**
c. **How many rules contains the decision rule set?**
d. **Why is the insurance variable hardly or not used in the rules (and tree)?**
e. **Which decision rules has the highest confidence and which rule the highest support? Translate this two rules to a decision rule in normal English.**
f. **Update the Decision Tree parameters to get a better performance on the test material (a performance of 80% or more is possible). How many rules are there now in the decision rule set?**
g. **Use the assessment information to say something about the type of misclassifications. For instance, are there patient with 10 or more hospital days (10_) classified as patients with only one or two hospital days?**
h. **Make an excel file with, for each new patient, the predicted number of hospital days.**

Appendix Weka information about the decision algorithm implemented in AlphaMiner

3.3 Decision Trees for Supervised Learning

Decision trees represent a supervised approach to classification. A decision tree is a simple structure where non-terminal nodes represent tests on one or more attributes and terminal nodes reflect decision outcomes. J.R. Quinlan has popularized the decision tree approach with his research spanning more than 15 years. The latest public domain implementation of Quinlan's model is C4.5. The Weka classifier package has its own version of C4.5 known as J48.

Although we will not present the specific decision tree algorithm here, we can summarize the general approach, specifically:

1. Choose an attribute that best differentiates the output attribute values.
2. Create a separate tree branch for each value of the chosen attribute.
3. Divide the instances into subgroups so as to reflect the attribute values of the chosen node.
4. For each subgroup, terminate the attribute selection process if:
    a. All members of a subgroup have the same value for the output attribute, terminate the attribute selection process for the current path and label the branch on the current path with the specified value.
    b. The subgroup contains a single node or no further distinguishing attributes can be determined. As in (a), label the branch with the output value seen by the majority of remaining instances.
5. For each subgroup created in (3) that has not been labeled as terminal, repeat the above process.

The algorithm is applied to the training data. The created decision tree is tested on a test data set, provided on is available. If test data is not available, J48 performs a cross-validation using the training data, as described in section 3.3.2. The created decision tree is then output in the Model section of the Learner Model Output.

3.3.1 ToolShed Output for the Weka Decision Tree Model

Here is the output of J48 when presented with the attributes Log T90, Log HR321, Log Fluence, and MFBMFR Class derived from the MFBMFR 3B catalog.

```
J48 pruned tree
---------------

log_T90 <= 0.40824
| log_T90 <= 0.019947: short (161.0)
| log_T90 > 0.019947
| | log_fluence <= -5.823021
| | | log_hr321 <= 0.756906: short (27.0/3.0)
| | | log_hr321 > 0.756906: inter (4.0)
| | log_fluence > -5.823021
```

```
| | | log_fluence <= -5.266626: inter (8.0)
| | | log_fluence > -5.266626: short (3.0/1.0)
log_T90 > 0.40824
| log_T90 <= 1.116873
| | log_fluence <= -5.935778: inter (60.0/1.0)
| | log_fluence > -5.935778
| | | log_hr321 <= 0.076742:
| | | | log_T90 <= 0.91677: inter (12.0)
| | | | log_T90 > 0.91677: long (5.0/1.0)
| | | log_hr321 > 0.076742
| | | | log_fluence <= -5.440835
| | | | | log_T90 <= 0.843606
| | | | | | log_hr321 <= 0.463924: inter (8.0)
| | | | | | log_hr321 > 0.463924: short (4.0/1.0)
| | | | | log_T90 > 0.843606: long (16.0)
| | | | log_fluence > -5.440835: long (31.0)
| log_T90 > 1.116873
| | log_fluence <= -6.100924
| | | log_T90 <= 1.544167
| | | | log_hr321 <= -0.268603: long (4.0)
| | | | log_hr321 > -0.268603
| | | | | log_hr321 <= 0.367237: inter (11.0/2.0)
| | | | | log_hr321 > 0.367237: long (2.0)
| | | log_T90 > 1.544167: long (8.0)
| | log_fluence > -6.100924: long (414.0)

Number of Leaves :  17

Size of the tree :  33
```

Explanation

Let us take a second look at the first three lines of the output to see how the tree structure is represented:

```
log_T90 <= 0.40824
| log_T90 <= 0.019947: short (161.0)
| log_T90 > 0.019947
```

Each line represents a node in the tree. The second two lines, those that start with a '|', are child nodes of the first line. In the general case, a node with one or more '|' characters before the rule is a child node of the node that the right-most line of '|' characters terminates at, if you follow it up the page. The next part of the line declares the rule. If the expression is true for a given instance, you either classify it if the rule is followed by a semicolon and a class designation--that designation becomes the classification of the rule--or, if it isn't followed by a semicolon, you continue to the next node in the tree (i.e. the first child node of the node you just evaluated the instance on). If the expression is instead false, you continue to the ``sister'' node of the node you just evaluated; that is, the node that has the same number of '|' characters before it and the same parent node.

Nodes that generate a classification, such as

```
| log_T90 <= 0.019947: short (161.0)
```

are followed by a number (sometimes two) in parentheses. The first number tells how many instances in the training set are correctly classified by this node, in this case 161 are. The second number, if it exists (if not, it is taken to be 0.0), represents the number of instances incorrectly classified by the node.

Following the output which describes the tree is some error measurement data on the model, which is described in depth in section 3.2.

3.3.2 Special Options

Before invoking the J48 model, you are presented with the following five options:

1. Which attribute is your *output* attribute
2. Confidence Factor
3. Minimum Number of Instances per Leaf
4. Number of folds for Cross-validation
5. Test Data Set

The first option allows you to choose from a list which attribute you want to be the output attribute. You will only be able to choose from categorical output attributes, as J48 only allows categorical attributes for output.

The second option determines the confidence value to be used when pruning the tree (removing branches that are deemed to provide little or no gain in statistical accuracy of the model). Since the default of 25% works reasonably well in most cases, you likely will not have to modify it. However, if the actual error rate on real data (or the error rate on cross-validation) is significantly higher than the error rate on the training data, you may want to decrease the confidence factor to cause more drastic pruning, and a more general model of the data. If you wnat a more specific modeling based on the training data, you can increase the confidence factor, which will decrease the amount of pruning that occurs.

The third option determines the minimum number of instances that must be present in the training data for a new leaf to be created in the decision tree to handle those particular instances. This too can create a more generalized or specialized tree; a higher number will create a more generalized tree and a lower number will create a more specialized tree.

The fourth option determines how to construct and then test the model in the absence of test data (which is generally the case with the ToolShed). If the number of folds for cross-validation is n, then $\frac{n-1}{n}$ of the training data is used to construct the model and $\frac{1}{n}$ of the training data is used to test the model. This process is then repeated n times so that all the training data is used exactly once in the test data. The n different error estimates are then averaged to yield an overall error estimate--the stratified cross-validation statistics you see. While extensive tests on numerous datasets have shown that ten-fold cross-validation is one of the best numbers for getting the most accurate error estimate, other values can be used. Decreasing the number of folds from the

default of 10 will likely decrease the amount of time it takes for the decision tree to be generated, and increasing the number of folds will likely increase the amount of time it takes. Of course, increasing the number of folds will create a larger dataset for the training data, which may increase accuracy of the decision tree; similarly, decreasing the number of folds will create a smaller dataset for the training data, which may decrease the accuracy of the decision tree.

The fifth option allows you to test the data against a set of test data rather than by testing the model through cross-validation, as described in the previous paragraph. The number of options you select is dependent on the number of saved extractions you have saved, with only those extractions that use the same attributes (and possibly more) being listed. There is also the option to use ``none" as the test data set. This option is always available, and choosing it will cause the model to use cross-validation rather than a test set for testing the model.

3.3.3 Advantages and Limitations

- Decision trees are easy to understand
- Decision trees are easily converted to a set of production rules
- Decision trees can classify both categorical and numerical data, but the output attribute must be categorical
- There are no a priori assumptions about the nature of the data.
- Multiple output attributes are not allowed
- Decision tree algorithms are unstable. Slight variations in the training data can result it different attribute selections at each choice point within the tree. The effect can be significant since attribute choices affect all descendent subtrees.
- Trees created from numeric data sets can be quite complex since attribute splits for numeric data are binary