Decision Trees

Release 3.5

Copyright

An Introduction to Decision Trees

A decision tree is used as a classifier for determining an appropriate action or decision (among a predetermined set of actions) for a given case. A decision tree helps you to effectively identify the factors you must consider and how each factor has historically been associated with different outcomes of the decision.

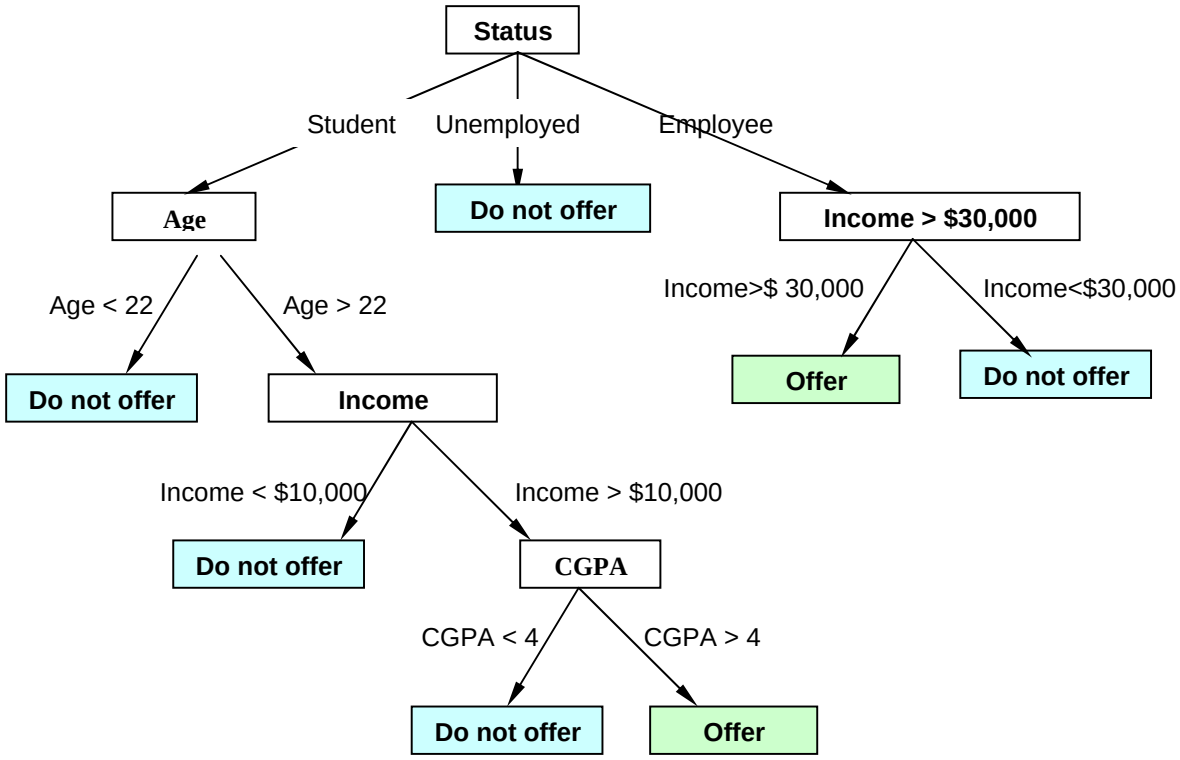
Decision trees have become one of the most popular data mining tools. Their visual presentation makes the decision trees very easy to read, understand and assimilate information from it. They are called decision trees because the resulting model is presented in the form of a tree structure. Decision trees are most commonly used for classification, that is, predicting to which group a particular case belongs.

A decision tree is constructed from a training set. A training set contains historical data, which is used to predict the possible outcomes such as aspects of customer behavior. For example, we can predict if a customer churns or remains loyal to the company.

Each record in the training set is characterized by a set of attributes and a class label. Attributes are features or variables that define the object. A class label is the decision or the set of possible outcomes. The tree consists of decision nodes and child nodes. Each decision node in the tree is labeled with a “test ”. The test is defined in terms of attributes and contains splits or branches for each possible outcome. Each branch or a split represents a possible value for the decision. A node that forms the last node in the branch and that doesn’t split further is called a Leaf node.

For example, in a credit risk case study, you have the task of targeting reliable candidates to be sent an invitation to apply for a credit card. Given the set of information about an individual, we need to determine whether or not he or she can be a reliable candidate. From the past historical data, we have information about candidates who are termed as bad or good credit risk. The information about an individual is given as a set of attributes such as sex (male or female), age, status (student, employee, or unemployed), College Grade Point Average (CGPA) and annual income.

A decision tree algorithm takes this data as input and creates a model that can be represented as either a graphical tree or a set of text rules. Based on this model, we can classify the candidates and decide whether to offer an invitation to apply for credit card. For example, a decision tree based on this data can look like the one depicted below.



The possible outcomes in this example are: **Offer** or **Do not offer** an invitation to apply for a credit card. We start at the root, which is labeled as *Status* and follow the branch labeled *Student*. In the test node *Age*, there are 2 branches, *Age < 22* and *Age > 22* . If we follow the *Age>22* branch, then we get another test node *Income*, that in turn splits into branches of *Income < $10,000* and *Income > $10,000* . The branch *Income > $10,000* leads to the test node *CGPA,* which splits into *CGPA < 4* and *CGPA > 4.* The branch *CGPA > 4* finally leads to the Leaf node labeled **Offer**, indicating that the person could be made an offer to apply for a credit card. This is the appropriate action to be taken for this case.

We can say that a decision tree not only provides a solution for a given case, but also clearly states the reason behind the choice.

Uses and Applications

Marketing

Market Segmentation
Decisions trees can be used to create segmentation of the original database, where each segment would be the set of the rules associated with the leaf node or part of tree. You can also classify your customers into distinct segments based on their behavior. In the previous example, one profile could be “Status is Student aged less than 22.”

Customer Profile
You can understand the profile of a particular group of customers, which shows a specific behavior of interest. Based on the profiles generated, you can address the needs of your customers more effectively.

Churn Management
With the help of decision trees, you can detect the trends and patterns in customer behavior, which then allow you to interpret signals of customer churn. You can minimize customer churn by analyzing the factors that are most likely to cause it.

Customer Profitability
You can obtain insights by understanding patterns of purchasing behavior of your most profitable customers and design the means to retain these customers. You can also predict how profitable a prospect can be, based on his characteristics and behavior.

Response Prediction
Decision trees enable you to predict the response rate for specific segments before launching marketing campaigns. This helps you to optimize the target groups for your marketing campaigns.

Banking & Finance

Credit Risk Prediction: You can predict credit risk for prospective customers and set their credit limit and pricing accordingly.

Fraud Detection: Decision trees help you to identify customers who are likely to engage in a fraudulent behavior

Data Mining Functions in the Analysis Pr ocess Designer (APD)

The Analysis Process Designer (APD) is the application environment for the SAP data mining solution. From SAP BW Release 3.5, data mining functions are fully integrated into the APD. You can perform the following functions in the APD:

- Creating and changing data mining models
- Training data mining models with SAP BW data (data mining model as data target in the analysis process)
- Execution of data mining methods such as prediction with decision tree, with cluster model and integration of data mining models from third parties (data mining model as a transformation in the analysis process)
- Visualization of data mining models

For more information, see SAP Library at help.sap.com under *SAP NetWeaver -> Release '04 -> Information Integration -> SAP Business Information Warehouse -> BI Platform -> Analysis Process Designer / Data Mining*

Typical Input

Following is a typical set of training data used for decision trees.

Rec No.	Region	Sales Period	Revenue	Sales Prediction
1	South	Winter	100000	Good
2	North	Spring	45000	Fair
3	West	Summer	30000	Average
4	East	Autumn	5000	Poor
5	West	Spring	5000	Poor
6	East	Spring	200000	Good
7	South	Summer	25000	Average
8	South	Spring	10000	Average
9	North	Winter	50000	Average

In the above example, each of the columns *Region, Sales Period* and *Region*, is an Attribute. The attribute *Sales Prediction* is the predicted field, that is, the output or decision. Each record in the dataset is used as a unique observation by the decision tree.


Decision trees do not support aggregations and the input data has to be pre-processed.

Similarly, for a weather forecast for a game of cricket, the input data can be in the following format.

Outlook	Temperature	Humidity	Windy	Outcome
Sunny	85	85	False	Play
Rain	70	90	True	Don't Play
Overcast	83	78	False	Play
Rain	70	96	False	Don't Play
Overcast	68	80	True	Don't Play
Sunny	85	70	True	Play
…	…	…	…	…

Hence for the weather forecast, the fields and the possible values can be summarized as:

Attributes	Possible Values
Outlook	Sunny, Overcast, Rain
Temperature	Continuous
Humidity	Continuous
Windy	True, False

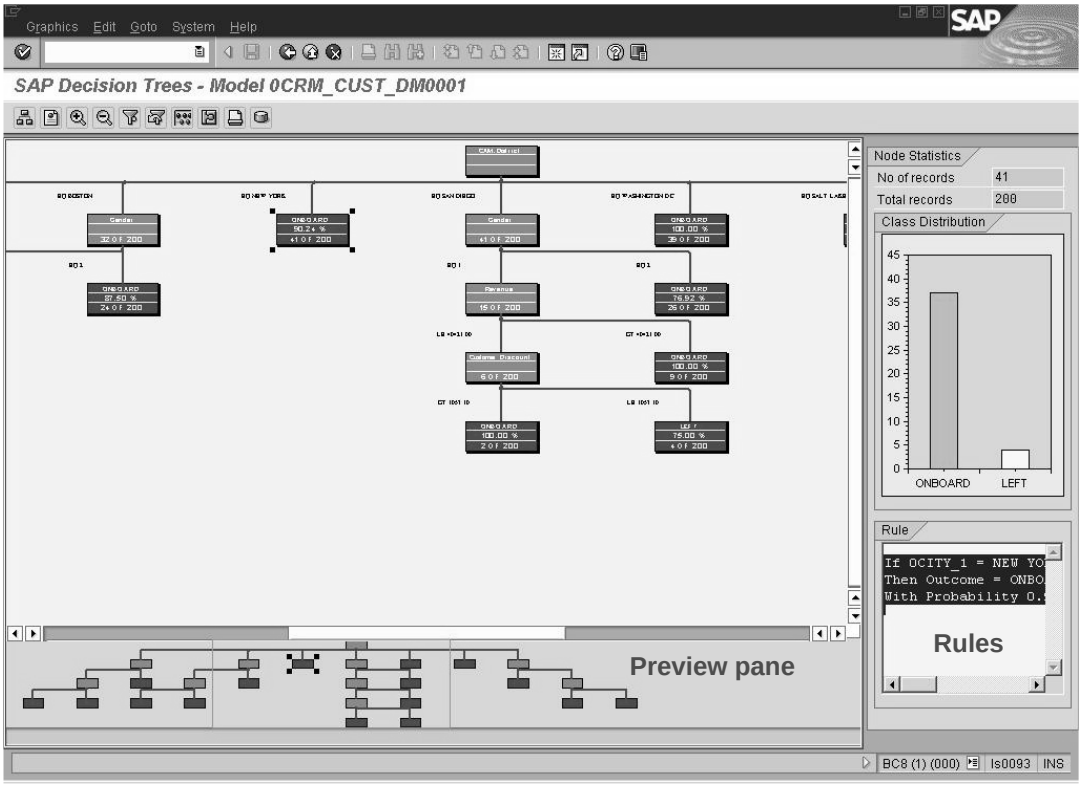
The possible outcomes or the value for the predicted field is *Play* or *Don't Play.*


Although the decision tree method determines which attributes are necessary, it is recommended that you consider only the ones that are relevant for classification purposes. This has a direct implication on the performance of the analysis.

Typically, an attribute like *Telephone Number, Social Security Number, Customer ID's* should not be considered for classification purposes since it contains too many unique values and a very rare chance of having any impact on the outcome.

Typical Output

The decision tree results can be viewed in either the *Network View* or the *Tree View*. Following is an example of how the typical output for the Decision Tree (*Network View*) would look like.

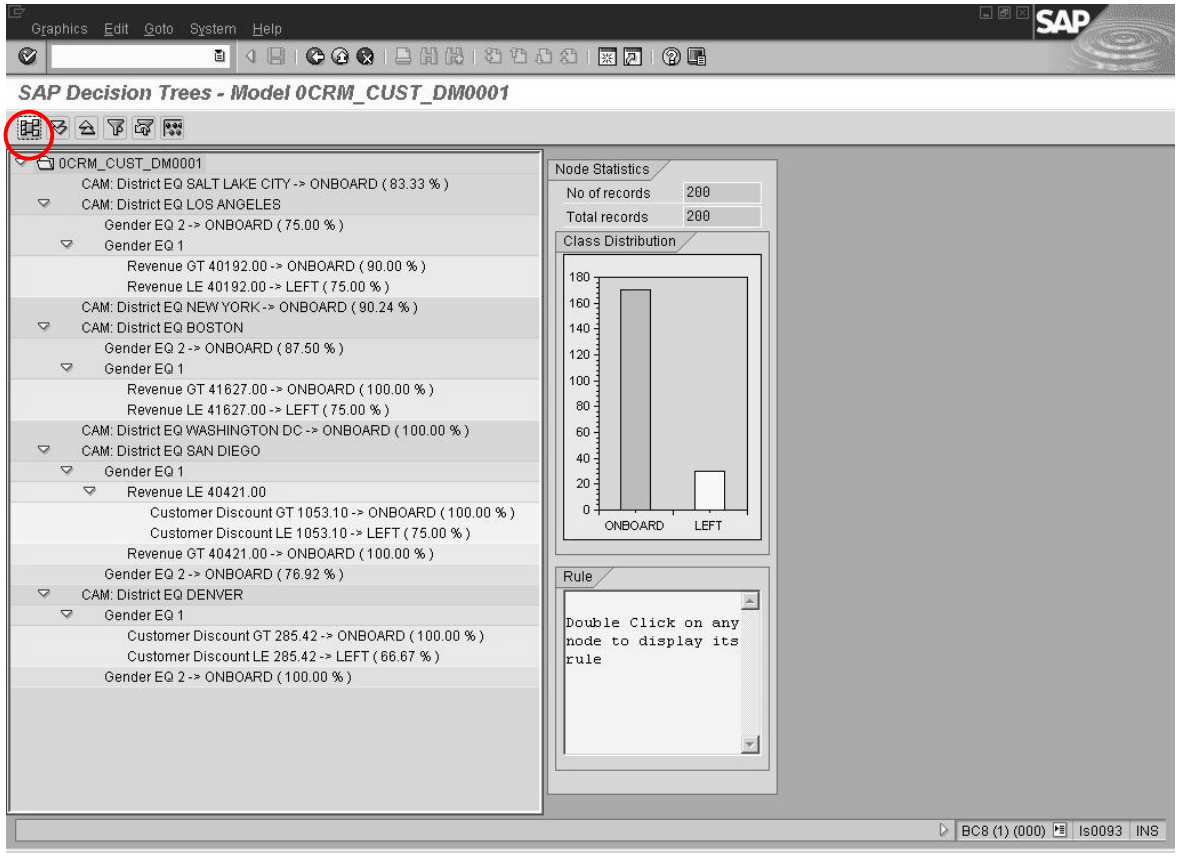


If the decision tree contains a large number of nodes, then you can use the **Preview** pane to focus on the set of the nodes that you would like to view. Decision Tree Rules for the selected node are displayed in the **Rules** pane and you can double click on the node to display the rules for that particular node.

Decision tree rules are concise and often easier to understand, particularly when the decision tree is very large. Following is an example for the rules of a particular node.

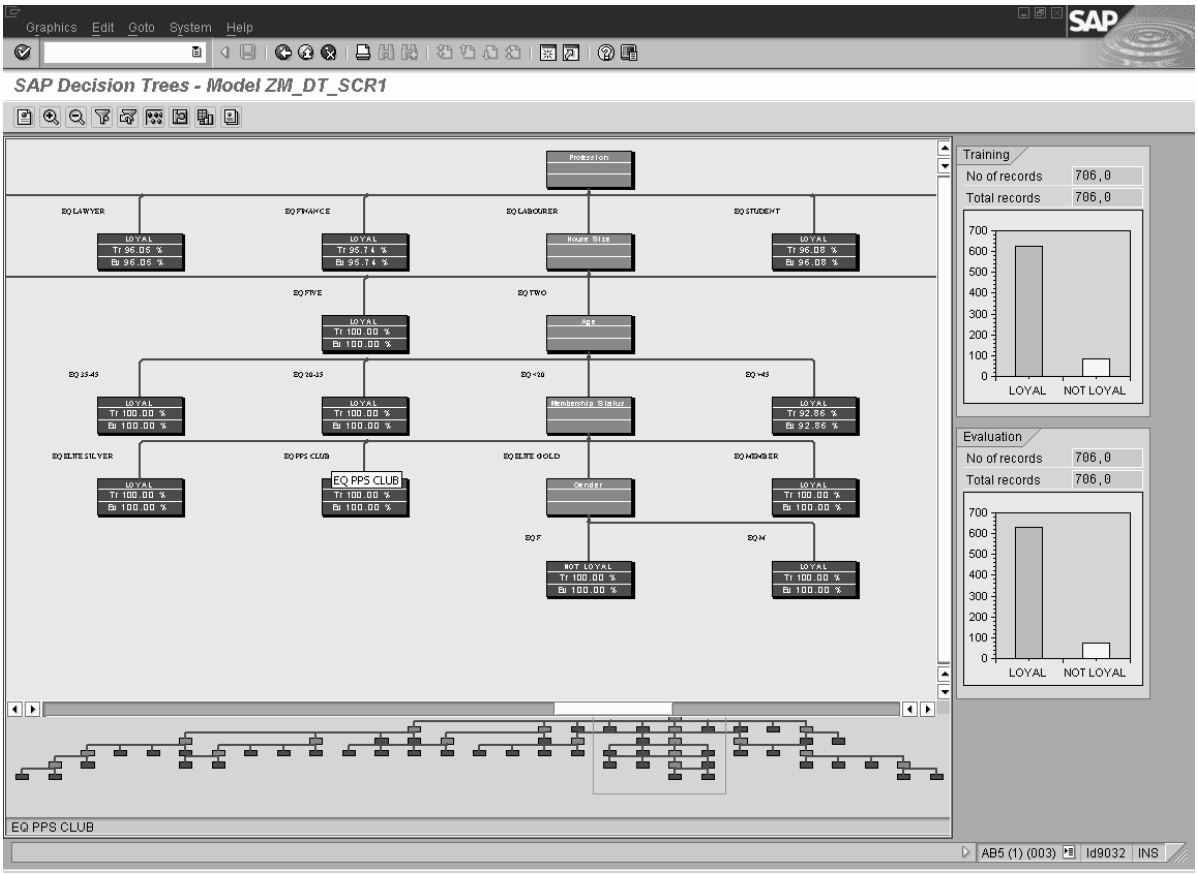
```
If 0CITY_1 = DENVER
And 0GENDER = 1
And 0CUST_DSCNT > 285.42
Then Outcome = ONBOARD
With Probability 1.00
```

Another view for the decision tree is the *Tree View*. Following is an example for the output in Tree form.

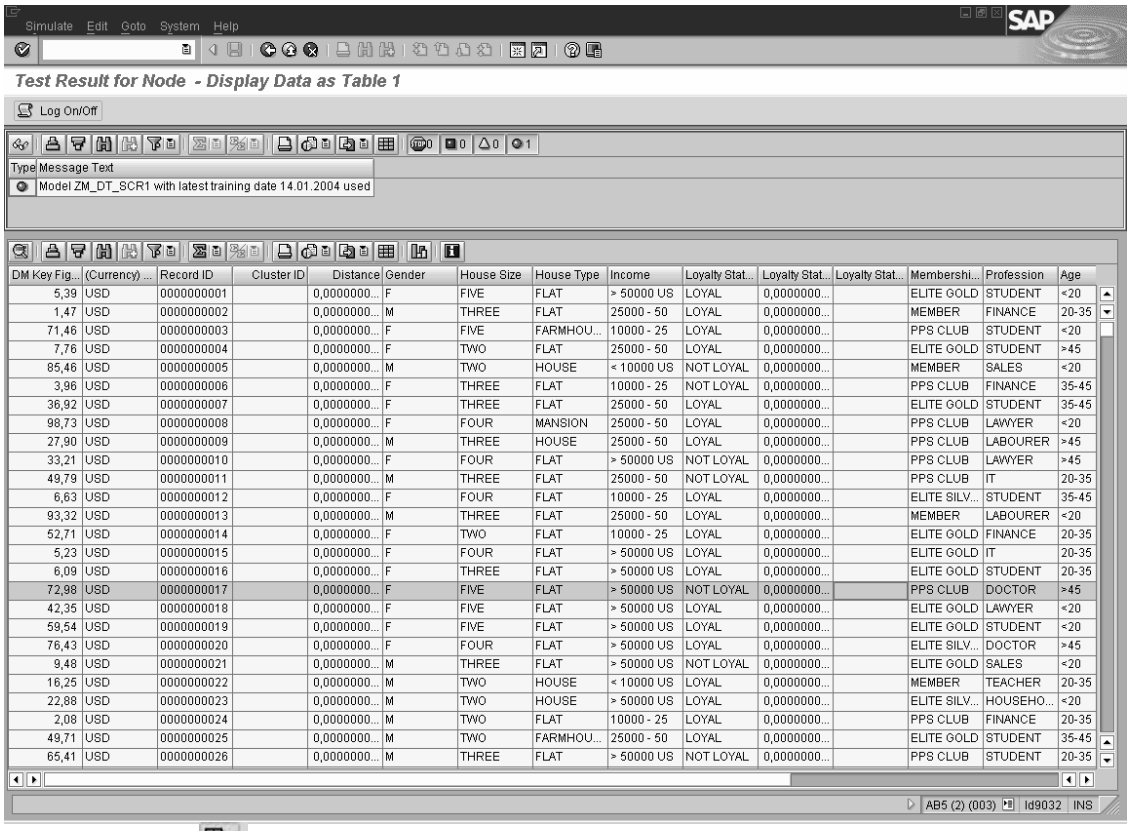


You can use the  button to switch between the *Network View* and the *Tree View*.

Following is an example of *Evaluation* results for a decision tree. You can display the evaluation results through the APD application environment, by choosing Calculation Summary -> Display for an evaluation model.

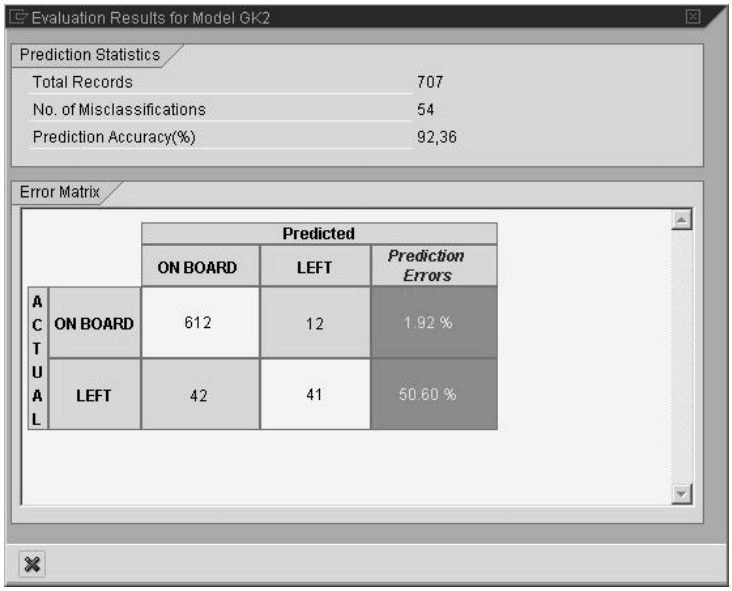


To view the evaluation results data, then you can use the *Display Data* function in APD.



DM Key Fig...	(Currency) ...	Record ID	Cluster ID	Distance	Gender	House Size	House Type	Income	Loyalty Stat...	Loyalty Stat...	Loyalty Stat...	Membershi...	Profession	Age
5,39	USD	0000000001		0,0000000...	F	FIVE	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE GOLD	STUDENT	<20
1,47	USD	0000000002		0,0000000...	M	THREE	FLAT	25000 - 50	LOYAL	0,0000000...		MEMBER	FINANCE	20-35
71,46	USD	0000000003		0,0000000...	F	FIVE	FARMHOU...	10000 - 25	LOYAL	0,0000000...		PPS CLUB	STUDENT	<20
7,76	USD	0000000004		0,0000000...	F	TWO	FLAT	25000 - 50	LOYAL	0,0000000...		ELITE GOLD	STUDENT	>45
85,46	USD	0000000005		0,0000000...	M	TWO	HOUSE	< 10000 US	NOT LOYAL	0,0000000...		MEMBER	SALES	<20
3,96	USD	0000000006		0,0000000...	F	THREE	FLAT	10000 - 25	NOT LOYAL	0,0000000...		PPS CLUB	FINANCE	35-45
36,92	USD	0000000007		0,0000000...	F	THREE	FLAT	25000 - 50	LOYAL	0,0000000...		ELITE GOLD	STUDENT	35-45
98,73	USD	0000000008		0,0000000...	F	FOUR	MANSION	25000 - 50	LOYAL	0,0000000...		PPS CLUB	LAWYER	<20
27,90	USD	0000000009		0,0000000...	M	THREE	HOUSE	25000 - 50	LOYAL	0,0000000...		PPS CLUB	LABOURER	>45
33,21	USD	0000000010		0,0000000...	F	FOUR	FLAT	> 50000 US	NOT LOYAL	0,0000000...		PPS CLUB	LAWYER	>45
49,79	USD	0000000011		0,0000000...	M	THREE	FLAT	25000 - 50	NOT LOYAL	0,0000000...		PPS CLUB	IT	20-35
6,63	USD	0000000012		0,0000000...	F	FOUR	FLAT	10000 - 25	LOYAL	0,0000000...		ELITE SILV...	STUDENT	35-45
93,32	USD	0000000013		0,0000000...	M	THREE	FLAT	25000 - 50	LOYAL	0,0000000...		MEMBER	LABOURER	<20
52,71	USD	0000000014		0,0000000...	F	TWO	FLAT	10000 - 25	LOYAL	0,0000000...		ELITE GOLD	FINANCE	20-35
5,23	USD	0000000015		0,0000000...	F	FOUR	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE GOLD	IT	20-35
6,09	USD	0000000016		0,0000000...	F	THREE	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE GOLD	STUDENT	20-35
72,98	USD	0000000017		0,0000000...	F	FIVE	FLAT	> 50000 US	NOT LOYAL	0,0000000...		PPS CLUB	DOCTOR	>45
42,35	USD	0000000018		0,0000000...	F	FIVE	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE GOLD	LAWYER	<20
59,54	USD	0000000019		0,0000000...	F	FIVE	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE GOLD	STUDENT	<20
76,43	USD	0000000020		0,0000000...	F	FOUR	FLAT	> 50000 US	LOYAL	0,0000000...		ELITE SILV...	DOCTOR	>45
9,48	USD	0000000021		0,0000000...	M	THREE	FLAT	> 50000 US	NOT LOYAL	0,0000000...		ELITE GOLD	SALES	<20
16,25	USD	0000000022		0,0000000...	M	TWO	HOUSE	< 10000 US	LOYAL	0,0000000...		MEMBER	TEACHER	20-35
22,88	USD	0000000023		0,0000000...	M	TWO	HOUSE	> 50000 US	LOYAL	0,0000000...		ELITE SILV...	HOUSEHO...	<20
2,08	USD	0000000024		0,0000000...	M	TWO	FLAT	10000 - 25	LOYAL	0,0000000...		PPS CLUB	FINANCE	20-35
49,71	USD	0000000025		0,0000000...	M	TWO	FARMHOU...	25000 - 50	LOYAL	0,0000000...		ELITE GOLD	STUDENT	35-45
65,41	USD	0000000026		0,0000000...	M	THREE	FLAT	> 50000 US	NOT LOYAL	0,0000000...		PPS CLUB	STUDENT	20-35

You can use the  *View Error Matrix* to view the *Error* or *Confusion Matrix* for the given decision tree. Following is an example of the *Error Matrix*, which displays the prediction accuracy, and prediction errors.

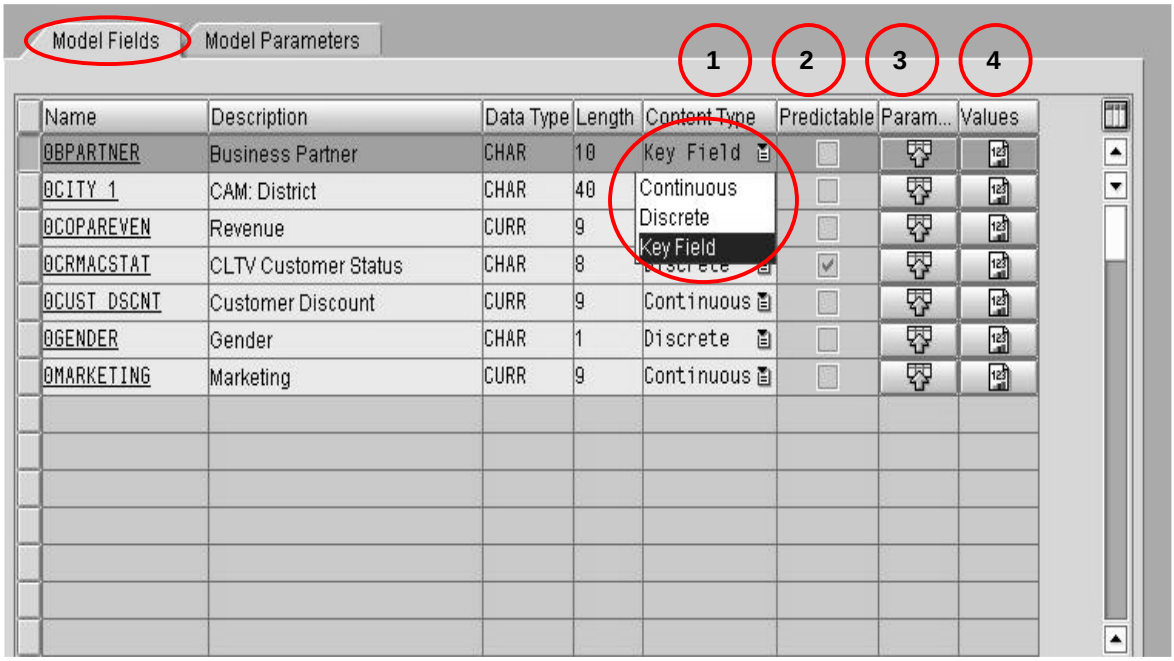


Settings for Decision Tree Classification

The input data for SAP's Decision Tree Classification is divided into two parts:

- Model Fields
- Model Parameters

Model Fields for Decision Tree Classification



Model fields are the attributes that define the object and the predictable field is the class label. In *Model Fields* screen, you can add the fields that are required for creating decision trees. You must define the content type for each model field.

(1) Content Type

It defines the data in a model field. There are 4 content types for model fields used in decision tree classification.

Key field: The key field acts as the record identifier. This field does not have any influence on the outcome.

Discrete: Also referred to as categorical, the data in the model field for this content type contains a finite set of values. For example, a model field 'Gender' has two values - Male and Female. Attributes like Color, Gender, and Status are examples of Discrete attributes.

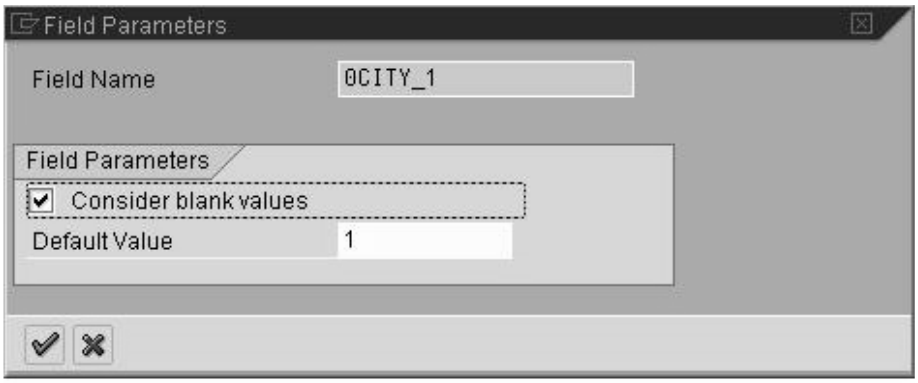
Continuous: Continuous data can have any value in an interval of real numbers. This implies that the value does not have to be an integer. Attributes having infinite set of possible real values are called Continuous. Typically, they have a Minimum and Maximum value and attribute values could be anything within this interval. Attributes like Salary, Sales Revenue, Quantity sold etc are

examples of Continuous attributes. You can discretize a Continuous attribute by defining fixed intervals. For example, if the salary ranges from $100 to $20000, then we can form intervals like $0 – 2000, $2000 – $4000, $4000 – $6000…. $18000 – $20000. An attribute value will fall into any one of these intervals.

(2) Predictable

You must select the *Predictable* checkbox for the field that needs to be predicted. You can have only one field as predictable. The predictable field is dependent on other model fields. A predictable field must be of content type *Discrete.* If the predictable field is of content type *Continuous*, then it has to be binned manually by giving range of values in the *Values* screen. A *Key field* cannot be defined as predictable.

(3) Parameter Values



You can specify the parameter values for each model field, except the *Key Field*. The field parameters are:

Consider blank values
You can use this field to specify whether the blank values in the model field should be considered. If you check this field, then the blank values are considered as valid values. If you do not select this field, then all the blank values are ignored.

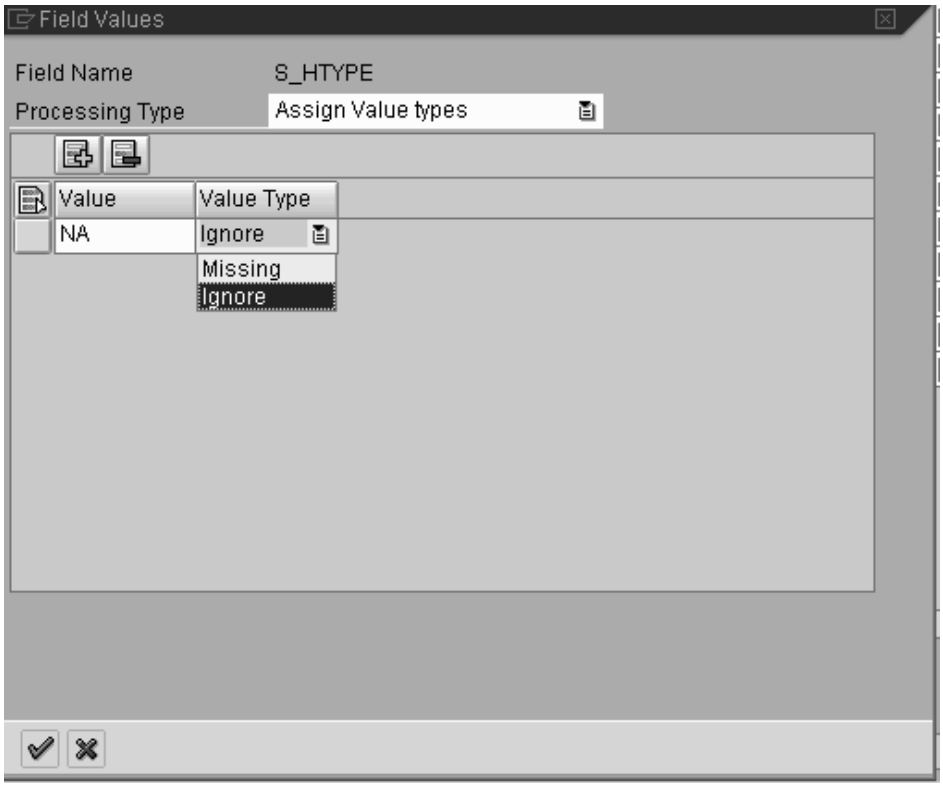
For model fields of type *Continuous*, by default, **0**s' and **blanks** are considered valid. In such fields, all non-numeric values are ignored. If you enter a default value, then this value will be considered instead of blank values and 0s'. The same approach applies to model fields of content type *Discrete.*

Default value
You can specify the default value for any value in the model field that is initial or specified as missing. Such values are replaced by the default value and will be considered valid for the purpose of decision tree classification. For example, you have a value like ‘XYZ’ and you have specified this value as missing and specified the default value as 100, then 100 will replace ‘XYZ’.

(3) Values for Model Fields

Discrete Values

This is valid for the content type *Discrete.*



You use this screen to specify which values are to be treated as missing or need to be ignored.

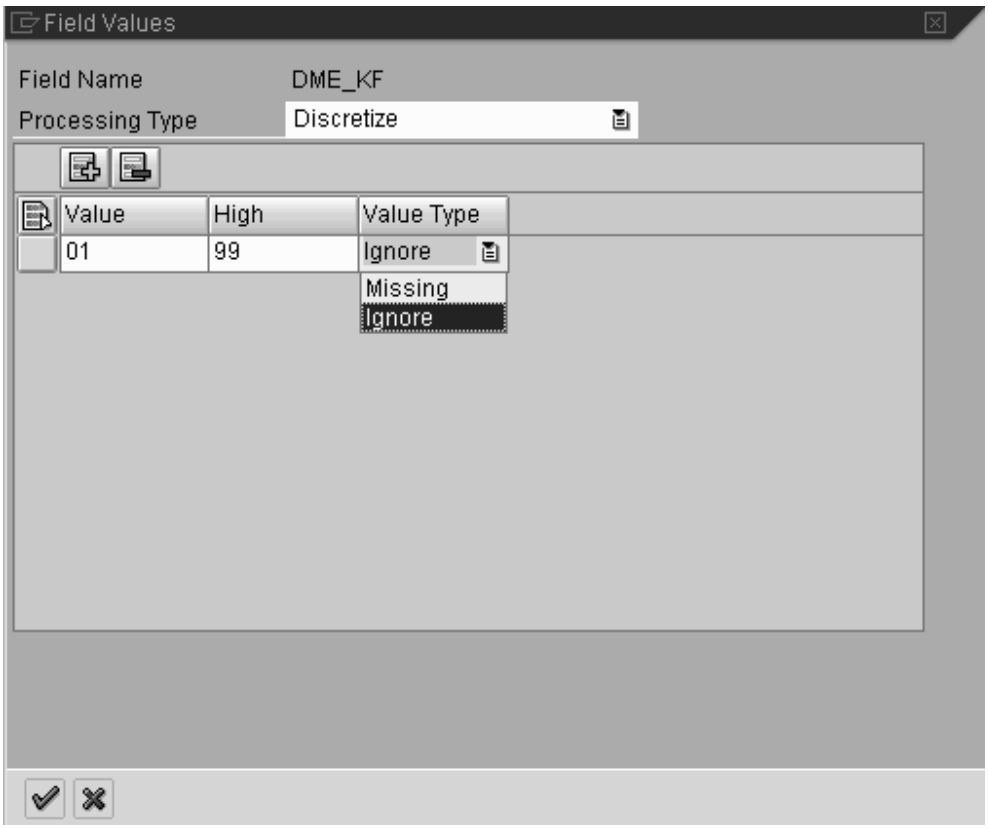
Processing Type: Assign Value Types

Value Type: You can define how the model field value must be treated. For example, if you assign a value 'NA' to the model field and choose *Ignore*, then this value will not be considered for Clustering and the weight of this value will be 0.

If you choose the *Missing* option, and if you have specified a default value for the model field, then the default value replaces this value wherever it occurs. If no default value is specified for this field, then this value will not be considered for Clustering and the we ight of this value will be 0.

Continuous Values

This option is only applicable for model fields of content type *Continuous*. This enables you to discretize the values in the continuous fields.



Value: This is the starting or the minimum value for the binning interval that you define. This value is included in the interval.

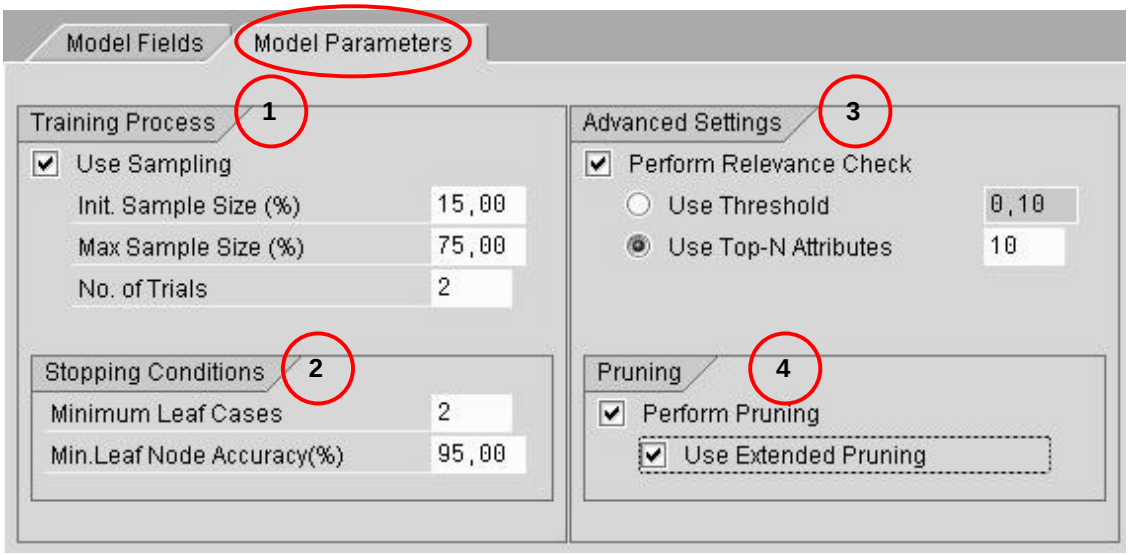
High: This indicates the maximum value for the binning interval or range that you define. This value is not included in the interval.

For example, if you have ranges from 1-10 and 10- 20, then the value 10 will fall under the second range, that is, 10-20.

The *High* value of the next range must be equal to that of the *Value* of the previous range. If this is not the case, then the system creates a range internally which satisfies the above criteria and ignores the values that fall in this range. For example, if you have the first range as 1-10 and the second range as 20-30, then the will create a range from 10-20 and all the values which fall under 10-20 will be ignored.

Value Type is the same as defined in the *Discrete Values* screen.

Model Parameters for Decision Tree Classification



(1) Training Process

Use Sampling
You can specify whether you want use the sampling method to perform training. You may use sample data for the classification or the entire records in the dataset. If you choose the *Use Sampling* option, an initial tree is built using sample data (equal to the specified initial sample size). The sample data contains an equal proportion of all class values. After training, the decision tree rules are applied to the remaining data. Any misclassified data (up to a maximum of 20% of data size) is added to the sample data and training is performed again. This continues till one of the following conditions is met:

- There are no misclassifications
- Tree accuracy is equal to or more than the specified accuracy (under stopping conditions)
- Training data size exceeds the specified maximum size

This process is repeated as many times as the number of trials specified.
Based on tree accuracy, the best tree is retained and all the other trees are discarded

To use the sampling technique in training, you must specify three parameters:

Initial Sampling Size
This indicates the initial sample size of data to be considered for training purposes.

Maximum Sample Size (in %)
This denotes the percentage of input data to be used for training. For example, if the percentage specified for the training data size is 50, then the decision tree is built with maximum training data size of 50%. Training stops as soon as this percentage of training data is achieved. This value must be greater than the initial sample size.

 The size of the training data, at any given point, cannot exceed the specified value for the Maximum Sample Size.

No of Trials
In this field, you can specify the number of trials for the decision tree classification.

 It is recommended to use the sampling method, if the data is suspected to have many duplicate records. Do not use sampling otherwise since it could take too much time. For example, consider a dataset with 2 fields *Gender* and *Marital Status* and an *Outcome/Predictable* field. *Gender* has 2 values and *Marital Status* has 3 values. This data can have a maximum of 6 unique records. However, if the dataset contains 20 records for this, it implies duplicate records and you can use sampling with initial size of 10 records.

It is recommended not to use the sampling when the data volume is very high, that is, more than 20,000 or so, and contains a number of duplicates.

(2) Stopping Conditions

Minimum Leaf Node Cases
You can specify the minimum number of cases that can be present in a node before that node can be split further.

Minimum Leaf Node Accuracy (in %)
This is the point at which the leaf node will not be split further. The node accuracy refers to error (is calculated as follows:

*(No. of cases with majority class) * 100 / Total number of cases at the node.*

When this accuracy or the *Minimum Leaf Node Cases* is reached, the tree splitting stops.

(3) Advanced Settings

Perform Relevance Check on Model Fields

With this field, you can specify whether a relevance check is to be performed. You can perform a relevance check to include just the more informative model fields in training. This will reduce the overall training time considerably.

The check uses the information gain of these model fields to remove irrelevant model fields based on the indicator you select:

Use threshold
You can enter a threshold between 0 (least informative) and 1 (most informative). Those model fields with an information gain higher than the specified threshold are considered for training. The higher the information gain, the more informative the model field is for training.

Use TOPN Attributes
The relevance check selects the specified number of top N most informative model fields.

(4) Perform Pruning

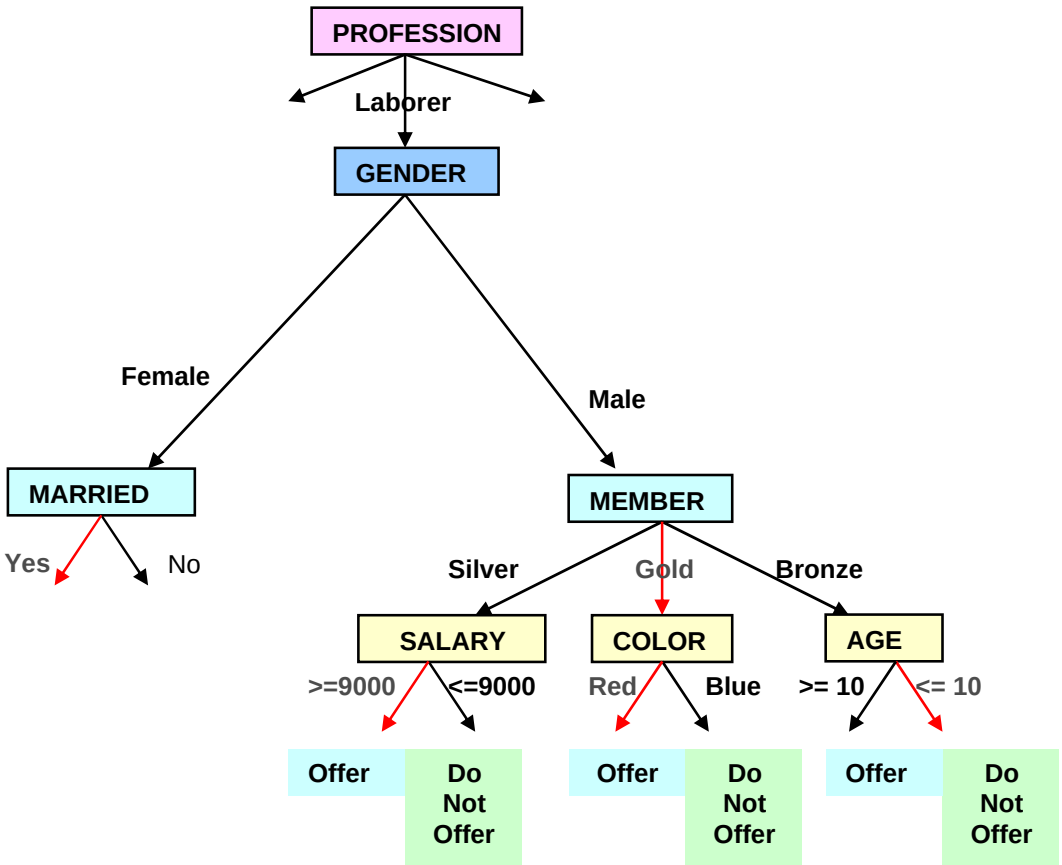
Pruning is a technique used to remove splits and sub trees, which are undesirable or superfluous. Pruning is also used to avoid over-fitting of data. Over-fitting is what happens when the decision tree algorithm finds meaningless “regularity” in the data, which is caused by irrelevant attributes. After Pruning, the resulting decision tree is smaller and comparatively more accurate in its predictions.

Extended Pruning is a subsequent extended reduction of a decision tree down to the essentials. This means that everything that can be removed without having a significant negative effect on the accuracy of the result is removed.

By performing extended pruning, the system checks if it is possible to achieve greater accuracy by replacing the current node with its major branch. If so, it retains this major branch and removes the current node and all its lower-level nodes.

 This is a very intensive process and requires an exponential scan of the whole data. Large datasets may cause extended pruning to be very time-consuming.

Consider a tree with the following information:

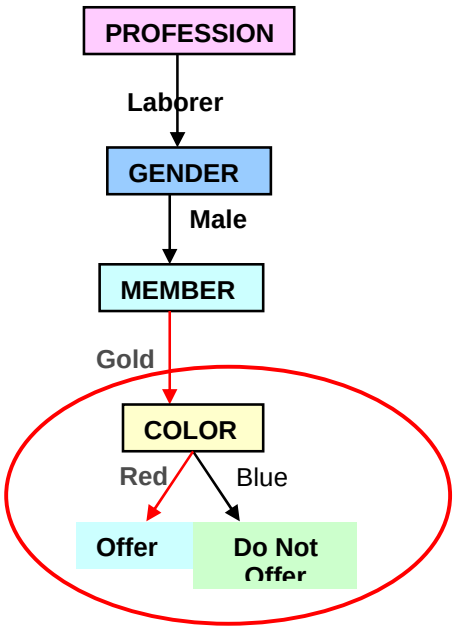


The nodes shown with red (**`Color = Red, Age <= 10, Salary > 9000, Member = Gold`** and **`Married = Yes`**) represent the splits that had a greater number of records/cases in the training set than the other corresponding child nodes.

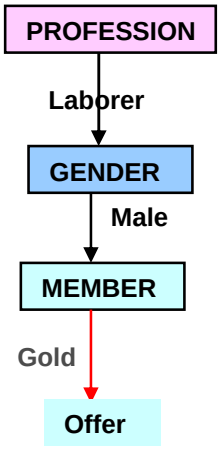
The purpose of pruning is to reduce the depth of trees and make it shallower without compromising on the prediction accuracy. Prediction accuracy may be defined as the number of cases that are predicted correctly as compared to the total number of cases predicted.

In case of normal pruning, we try to find if the prediction accuracy at a node is better without its children or sub-tree. In the example below, the sub tree under the node MEMBER is removed and converted into a leaf.

Before Pruning

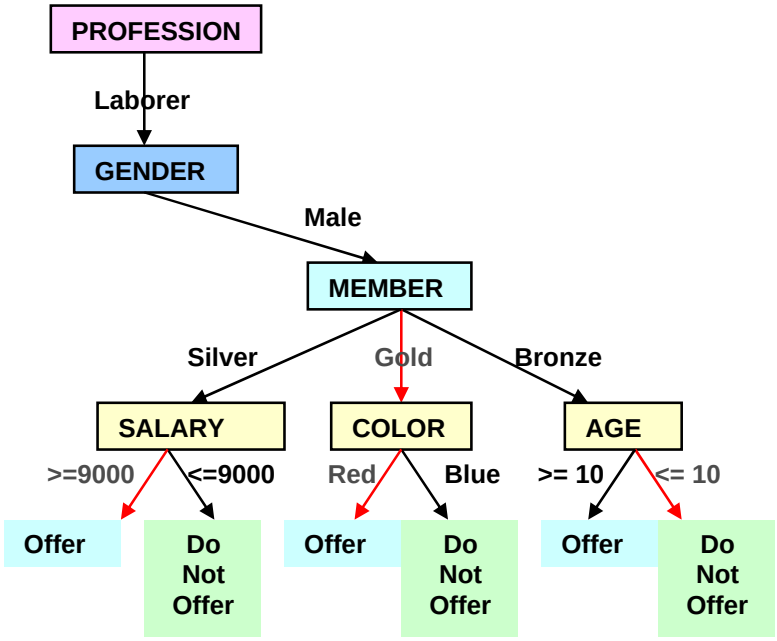


After Pruning

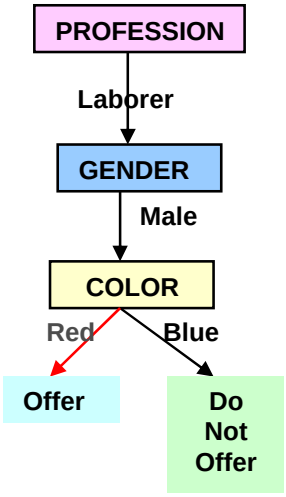


In case of *Extended Pruning*, the system checks if a node can be replaced by one of its child node itself. To optimize the performance, the system only checks for the child nodes that consist of majority of the cases. In the example below, the system checks if the node COLOR can replace the node MEMBER.

Before Extended Pruning



After Extended Pruning:



This process involves extensive calculations and hence would be very performance intensive.

Decision Tree ID3, C4.5 Algorithm

The decision tree data mining method is based on Quinlan's C4.5 Algorithm. The C4.5 decision tree algorithm uses concept learning as process of building trees.

Information Theory
Information Theory creates shallower trees by streamlining the order in which to select the attributes for splitting. This process involves calculating:

- An *Entropy* measure for all attributes and its values in a node
- Splitting the node using an attribute with the lowest entropy measure into as many child nodes as there are distinct values for the chosen attribute.

Entropy is a statistical property in information theory, which is a measure of information conveyed by a probability distribution. Detailed explanation for entropy follows in the latter part of this section.

In a classification algorithm where the attributes are picked randomly, different trees with different attributes are generated. Some attributes may give a good split whereas others may give deep and inefficient splits. Choosing the attribute, which gives a best possible split, would greatly reduce the time and computing process while generating decision trees.

An Example to illustrate the C4.5 Decision tree algorithm

Consider the following data showing observations made, over the course of two weeks, to decide whether the weather is amenable to play.

Day	Outlook	Temperature (Centigrade)	Humidity	Wind	Play
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

The attributes *Temperature* and *Humidity* are *Continuous* attributes. All the remaining attributes are *Discrete*.

The attribute *Play* is the *Discrete* attribute with the predicted outcomes as *Yes* as *No*. Of course, this works as well for attributes with more than two values.

Step I: Get the Attribute for Splitting

The next step is to locate the attribute that is to be used for the first split.

Gain Ration is used to determine the attribute that should be used to make the split. This property measures how well a given attribute separates the training set in to targeted classes.

To define Gain Ratio better, it is important to understand the Information Theory called *Entropy*. Entropy, in statistical terms, is defined as follows:

Given any discrete probability distribution $\mathbf{P=(P_1, P_2, P_3 \ldots P_n)}$, (thus $\Sigma_{i\le n} p_i=1$) the *Entropy* or *The Information Conveyed by this distribution* is:

$$\mathbf{ent(P) = -[p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + \ldots + p_n * \log_2(p_n)]}$$

where we set $\mathbf{p_i * \log_i(p_i):=0 \text{ if } p_i=0}$.

For a decision tree, we use the following natural distributions:

Let **T** be a set of records and A be an attribute of those records. For any value **a** of A let

$T_{=a}$:= the set of records of T with value **a** of the attribute A.

Moreover, if there is an ordering < on the values of A, then let

$T_{<a}$:= the set of records of T with value <**a**.

Analogous $T_{\le a}$, $T_{\ge a}$ or $T_{>a}$.

If for a discrete attribute A, $\{a_1,a_2,\ldots,a_n\}$ is the set of all values of A, then $(T_{=a1},T_{=a2}, \ldots ,T_{=an})$ forms a partition of T. This partition gives the distribution $(|T_{=a1}|/|T|, |T_{=a2}|/|T|, \ldots ,|T_{=an}|/|T|)$, where $|T_{=ai}|$ and $|T|$ are the number of records in $T_{=ai}$ and T. Now we define:

$$ent_{A,=}(T) := ent(|T_{=a1}|/|T|, |T_{=a2}|/|T|, \ldots ,|T_{=an}|/|T|)$$
$$= -[|T_{=a1}|/|T| * \log_2(|T_{=a1}|/|T|) + \ldots + |T_{=an}|/|T| * \log_2(|T_{=an}|/|T|)]$$

If the values of A are totally ordered by <, then for any value **a** we have the partition $(T_{\le a},T_{>a})$ and define

$$ent_{A,<,a}(T) := ent(|T_{\le a}|/|T|, |T_{>a}|/|T|)$$
$$= -[|T_{\le a}|/|T| * \log_2(|T_{\le a}|/|T|) + |T_{>a}|/|T| * \log_2(|T_{>a}|/|T|)]$$

To simplify the notation, we write in both cases just

$ent_A(T)$.

Let us now calculate the *Entropy* of the distribution for *Play* in the above example.

Let **T** be the set of records of the table. The values for *Play* in the records of **T** form the sequence {No, No, Yes, Yes, Yes, No, Yes, No, Yes, Yes, Yes, Yes, Yes, No}

P = (5/14, 9/14), since
$|T_{=No}|$ = 5 (attribute value *No* occurs 5 times),
$|T_{=Yes}|$ = 9 (attribute value *Yes* occurs 9 times) and
| T | = 14 (total 14 occurrences)

$ent_{Play}(T) = -[(9/14) \text{ Log } (9/14) + (5/14) \text{ Log } (5/14)]$

If a partition consists of only one part, then *Entropy* = 0 and we can say that the data is perfectly classified. For example, if:

- P is (1), then ent(P) = 0
- P is (0.5, 0.5) then ent(P) = 1
- P is (0.67, 0.33), then ent(P) = 0.91
- P is (1/n, ... ,1/n), then ent(P) = $\log_2(n)$

 The more uniform the probability distribution, the greater is its information.

If we partition T on the basis of all values $x_1,x_2,...x_k$ of a second attribute X into sets $T_{=x1}$, $T_{=x2}$, ... , $T_{=xn}$, then we define $Info_A(X,T)$ as weighted average of $ent_A(T_{=xi})$:

$$Info_A(X_{=},T) := \Sigma_{i \le k} |T_{=xi}|/|T| * ent_A(T_{=xi}) = |T_{=x1}|/|T| * ent_A(T_{=x1}) + ... + |T_{=xk}|/|T| * ent_A(T_{=xk})$$

For an ordered attribute X and some value x we define analogically

$$Info_A(X_{<,x},T) := |T_{\le x}|/|T| * ent_A(T_{\le x}) + |T_{>x}|/|T| * ent_A(T_{>x})$$

$Info_A(X,T)$ represents the information needed to identify an element of T, after the value of attribute **X** has been obtained.

In the given example, let us take the attribute *Outlook*. We now have **X** = *Outlook*. Since the attribute *Outlook* has three possible values *Sunny, Rainy* and *Overcast*, there will be a partition of T into three sets, that is $T_{=Sunny}$, $T_{=Rainy}$, $T_{=Overcast}$.

Now, we can define the term *Information Gain*. It is the difference between the information needed to identify an element of **T** and the information needed to identify an element of **T** after the value of attribute **X** has been obtained, that is, this is the gain in information due to attribute **X**. Hence,

$Gain_A(X, T) := ent_A(T) - Info_A(X, T)$

For the attribute *Outlook*, the

$Gain_{Play}$ (Outlook, T) = ent_{Play} (T) – $Info_{Play}$ (Outlook, T) = 0.94 - 0.694 = 0.246

Following is the Information Gain for the *Discrete* attributes Outlook and Wind.

Attribute Name	Information Gain
Outlook	0.246
Wind	0.048

We can use *Information Gain* to rank attributes and to build decision trees in such a way that at each node is located the attribute with greatest gain. But *Information Gain* tends to favor

attributes that have a large number of values. For example, if there is an attribute X having a distinct value for each record, then Info (X, T) = 0, and thus Gain (X, T) is maximal.

To overcome this problem, *Gain Ratio* is used.

For any ordered attribute X and any attribute value x, $Gain_A(X_{<,x}, T) = ent_A(T) – Info_A(X_{<,x}, T)$ and $ent_{X,<,x}(T)$ depend on the attribute value x, which defines the partition $(T_{\le x},T_{>x})$. Therefore we define the *Gain Ratio* as supremum of those quotients:

$$GainRatio_A(X, T) := \sup \{Gain_A(X_{<,x}, T) / ent_{X,<,x}(T) \mid x \text{ is value of } X\}$$
if X is ordered.

The *Gain Ratio* doesn't depend on the base of the logarithm, which is used for the computation of the involved terms.

In the example, we first need to calculate the $ent_X(T)$ for X=*Outlook* to compute the value of *Gain Ratio*. Hence,

$ent_{Outlook}(T)$ **= -** 5/14 * Log (5/14) - 4/14 * Log (4/14) - 5/14 * Log (5/14) = **1.577**

$GainRatio_{Play}$(Outlook,T) = 0.246/1.577 = 0.156

Similarly we can calculate the Gain Ratio for the attribute *Wind*.

$ent_{Wind}(T) = - 6/14 * \log_2 (6/14) - 8/14 * \log_2 (8/14) = 0.985$

$GainRatio_{Play}$(Wind,T) = 0.048/0.985 = 0.049

Now we are required to calculate the *Gain Ratio* for the continuous attributes *Humidity* and *Temperature*.

For *Humidity*, you have to find the value x with the partition $(T_{\le x},T_{>x})$, that gives the highest quotient $Gain_{Play}(Humidity_{<,x},T) / ent_{Humidity,<,x}(T)$. In the above example, $(T_{\le 75},T_{>75})$ is the best one. This method involves a substantial number of computations.

After the *Gain Ratio* is calculated for each of the attributes, the attribute having maximum *Gain Ratio* is chosen for the first split. The number of branches at the split is equal to the number of possible attribute values in case of *Discrete* attributes whereas a *Continuous* attribute will have two splits.

To take care of missing values and bias of particular distribution, the algorithm logic is further fine tuned in the actual implementation using some thresholds in case of *Continuous* fields.

With the determination of an attribute X for the first split, we have also fixed some partition $(T_{=x1}, T_{=x2}, \dots, T_{=xn})$ or $(T_{\le x},T_{>x})$ of the training data. Then for every set $T_{=xi}$ (or $T_{\le x},T_{>x}$) of the partition, an attribute for the next splitting must be determined in the same way, but using the set $T_{=xi}$ instead of the training set T. Repeating this procedure, one gets from one level to the next a refinement of partitions. When the splitting is stopped, all partitions together form a finite tree, which is defined by the ordering $P \le Q :\Leftrightarrow \forall p \in P\ \exists q \in Q\ p \subseteq q$ for any partitions P and Q. The union U of all $\le$-minimal partitions form a partition of T. Every element u of U is a subset of T. It gets assigned with that value of the target attribute (in our example *Play*), which occurs most frequently in the records of u. E.g. if $u=(T_{Outlook=Rain})_{Humidity>=75}$, then we have the values Yes, Yes, Yes, No for the attribute *Play*. Therefore u gets assigned with the value Yes.

Step II: Obtaining the right sized trees

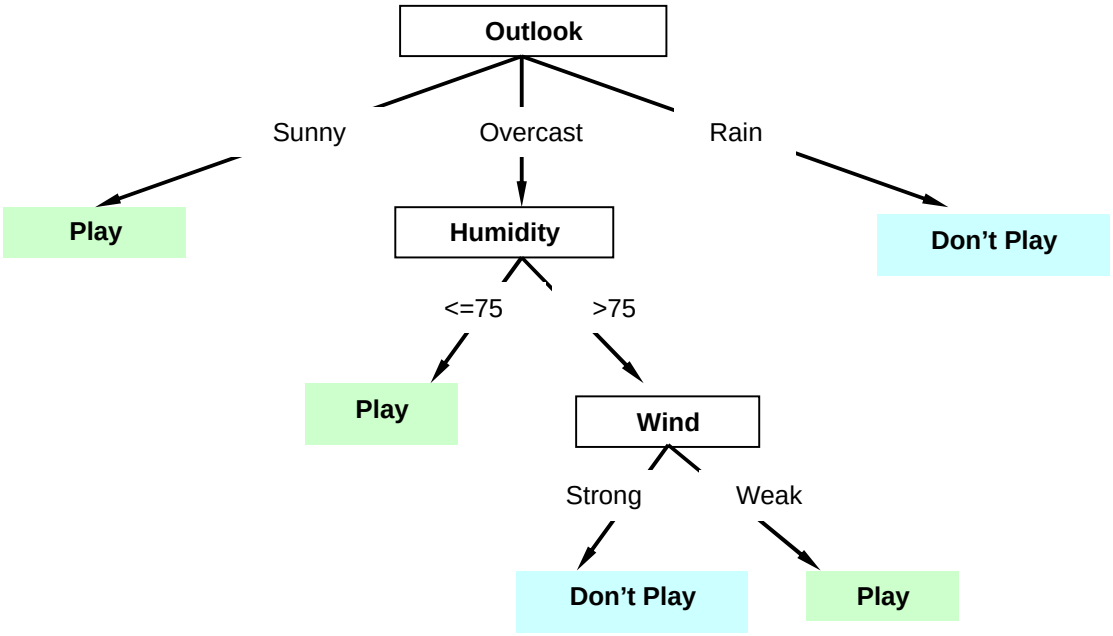
After constructing the tree, the last step is to check whether the tree:

- Is reasonably sized
- Properly generalizes the data and
- Fits the data properly

In recursive partitioning, it is desirable to know when to stop further splitting of data. The ideal classification would be a shallow tree with good generalization accuracy, honest error rate estimation and good predictive and descriptive nature. For this purpose, we use stopping rules and *Pruning.*

While using *Pruning,* the aim is to build a complete tree and to remove the sub-trees, which do not contribute significantly towards the generalization accuracy. This method is useful if there is increasing skew in the class distribution and increasing sample size.

Following is part of the decision tree that would be constructed from the example after applying the algorithm.



Evaluation

Once the tree is obtained, you need to determine how accurate or how valid this decision tree is. You can do this by using the tree to classify a separate set of data whose outcomes are already known. If you compare the predicted outcome with the known outcome, you can easily discover the number of correct predictions and ones that were not correctly predicted. This information can then be displayed in the form of a matrix, usually called the *Misclassification Matrix* or *Error Matrix* or *Confusion Matrix*. You use this matrix to know which outcome values the tree predicts well and the values that the tree doesn't predict properly.

For example, take a decision class with values GOOD, BAD and FAIR as illustrated below.

Predicted -> **Actual**	GOOD	FAIR	BAD	**%Accuracy of Predicted**
GOOD	70	20	10	70%
FAIR	10	50	40	50%
BAD	5	5	90	90%

From the misclassification matrix, it is clear that the prediction for GOOD and BAD cases are more accurate as compared to FAIR, whose accuracy is just 50%. If the Evaluation results are not satisfactory, then you could consider one of the following options:

- Try evaluating with another evaluation dataset, if any, to compare the various results.
- Change the model conditions, for example, the stopping conditions or relevance analysis, and repeat the evaluation process.
- Enrich the training dataset with further observations and retrain the model.

Prediction

Prediction is the main application of a decision tree. Using the knowledge from historical cases, we can now forecast the possible outcomes for new cases. After the prediction, each case is given the possible outcome and the confidence of this outcome.

The logic of prediction is to begin from the root and to continue applying the tests until a leaf node is reached. The prediction outcome is the *Predicted Value* and the *Confidence* associated to the leaf node. *Confidence* is expressed in terms of percentage or probability. *Confidence* signifies the support of the predicted value, among other cases in the training set, which have the same attribute values. This process is repeated for each case in the decision tree.

During the tree traversal, for any node test, if a particular attribute has a missing value or a value that is not found in the training set, then the cumulative outcome from all the child nodes below this node is assigned as the predicted value.

Prediction can be applied on a single customer or a set of customers. *Prediction* for a single customer record is useful in a call center based or web-based application to provide personalized service on the basis of the predicted value.