

CHAPTER 7

INTRODUCTION TO CLASSIFICATION

Learning Objectives:

1. *Understand basic concepts in the construction of decision trees.*
 2. *Compute impurity measures such as entropy and the Gini index.*
 3. *Use KNIME to construct workflows for decision classifier.*
-

Content

Lecture Notes	p.2	
- Introduction to classification		p.2
- Training and test sets		p.3
- Decision tree classifier		p.3
- How to grow a tree?		p.5
- Attribute selection		p.6
- Attribute selection using entropy		p.9
- More about Gini and entropy measures		p.11
- Overfitting		p.12
- Build Decision Tree in KNIME: Case study with UCI dataset		p.13
- Evaluation of Performance of Classification Models		p.15
 Tutorial 7	 p.18	
Answers	p.22	
Lab 7A	p.24	
Lab 7B	p.27	

1. Introduction to Classification

In classification, we try to predict the value of a target variable y based on the values of an attribute set $\mathbf{x} = (x_1, x_2, \dots, x_p)$. The input data for a classification task is a collection of records (\mathbf{x}, y) . As mentioned in a previous chapter, each record is also known as an *observation*, *instance*, *sample*, *example*, or *row*. Here, (\mathbf{x}, y) is a record, $\mathbf{x} = (x_1, x_2, \dots, x_p)$ is the *attribute set* and y the *target variable*.

Also recall that in a classification task, the target variable y is categorical. Other commonly used names for the target variable include *category*, *class* or *class label*.

Example 1: The example below shows a sample dataset used for classifying whether a patient has heart disease. The attributes and its description are given in the table below. Identify the response and predictor variables:

Attribute	Description
<i>fbs</i>	Fasting blood sugar > 120 mg/dl (1 = Yes, 0 = No)
<i>exang</i>	Exercise-induced angina (1 = Yes, 0 = No)
<i>cp</i>	Chest pain type (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)
<i>thalach</i>	Maximum heart rate achieved
<i>Heart</i>	Heart disease (Yes = heart disease, No = normal)

Solution:

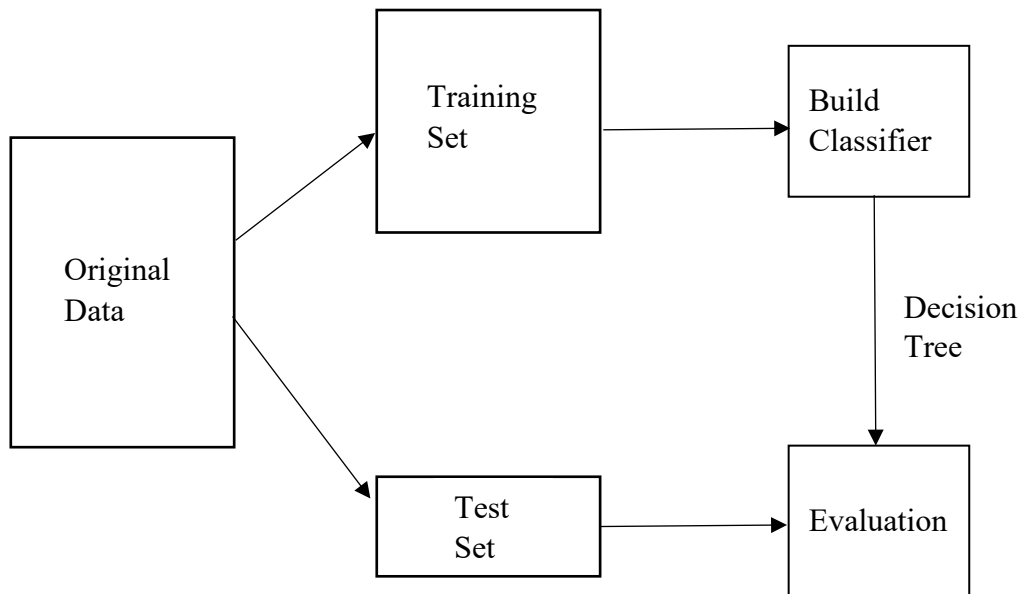
Other examples include:

- Predicting whether tumor cells are benign or malignant
- Predicting whether a student is at risk (yes/no) of failure in a module
- Classifying whether a credit card transaction is fraudulent or legitimate.

In the above, the target variable results in a binary outcome and are **binomial classification** tasks. We can also have **multinomial classification**, where the target variable involves more than two outcomes. An example is to classify a news story as politics, weather, entertainment, etc.

2. Training and Test Sets

In order to develop a model that makes fewer errors in classification, the available dataset (with *known* target variable) is usually divided into two subsets: a **training** set and a **test** set. Records from the training set are used to build (or train) a model. Records from the test set are used to evaluate the accuracy of the model. It is assumed that both the training data and test data are representative samples of the underlying problem. Note that the test set is NOT used in any way to train the model.



There are many classification techniques. In this course, we will only discuss the Decision tree classifier technique.

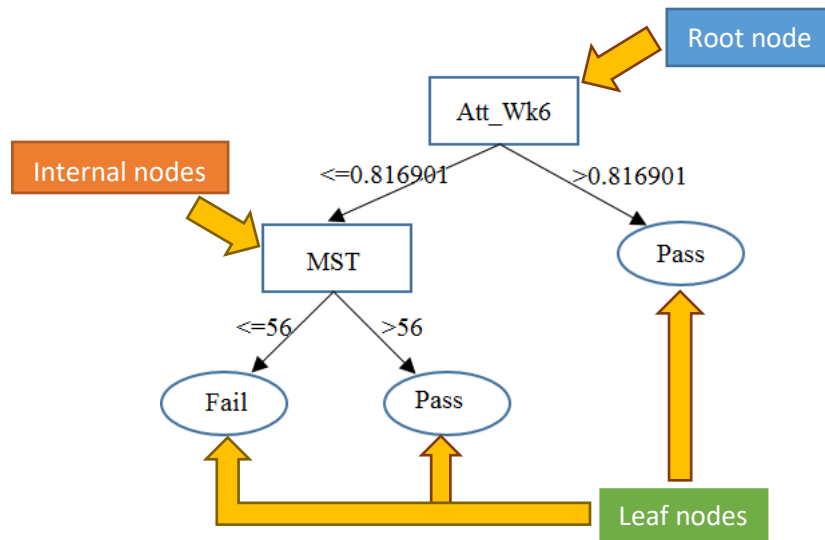
3. Decision Tree Classifier

A decision tree classifier is a widely used supervised technique which can be used to predict the *class label* (or *target variable*) y given the attribute vector \mathbf{x} of an unknown observation (\mathbf{x}, y) .

A decision tree is grown using a process called recursive splitting. The resulting tree consists of three types of nodes:

- a *root* node, with no incoming edges and zero or more outgoing edges
- *internal* nodes, with exactly one incoming edge and two or more outgoing edges
- *leaf* or *terminal* nodes, with exactly one incoming edge and no outgoing edges

An example is given below:



Decision Trees

Decision trees can be likened to a Guessing game where the player is allowed to ask the Game Host “Yes-no” questions to arrive at the hidden answer. The decision tree represents the questioning strategy a player uses to (hopefully) arrive at the correct answer.

Nodes are basically questions that you ask and the branches (edges) of this tree are the possible responses to the questions. Each response gives rise to more questions and so on, until the series of answers terminates with a guess as to the hidden value.

Root nodes represent the very first question a player asks to the Game Host.

Internal nodes represent subsequent questions you ask.

Leaf nodes represent your guess as to the hidden value. At that point, you would have collected information from the Game host about the hidden value in order to make a reasonable guess.

The tree above is used to predict whether a student will pass or fail a particular module at the end of the semester. The attribute vector is $\mathbf{x} = (x_1, x_2)$, where

$x_1 = Att_Wk6$ = cumulative attendance of the student up to Week 6, and

$x_2 = MST$ = MST score of student in module

The root node uses the attribute *Att_Wk6* to separate the students with an attendance cutoff at 81.69%. The right child node consists of students with attendance higher than 81.69% who subsequently ended at the leaf node “*Pass*”. If a student has attendance less than 81.69%, a subsequent attribute, *MST* score, is used to separate the students who subsequently either pass or fail the module.

The above can be summarized as a set of classification rules of the form **IF..THEN**. For the tree above, the following rules can be derived:

IF $Att_Wk6 > 0.816901$ THEN class=Pass;
 IF $Att_Wk6 \leq 0.816901$ AND $MST \leq 56$ THEN class=Fail;
 IF $Att_Wk6 \leq 0.816901$ AND $MST > 56$ THEN class=Pass;

Example 2: The records of two students are given below. Use the decision tree above to predict whether the student will pass or fail the module at the end of the semester.

<i>Stud_ID</i>	<i>Att_Wk6</i>	<i>MST</i>	<i>Pass/Fail?</i>
P000001	0.9650	75.1	
P000002	0.8019	53.7	

4. How to Grow a Tree?

In order to perform classification, we need to produce a decision tree which is derived from our training set. In a decision tree classifier, the assigned label is a result of a sequence of “Yes..No” answers to questions (nodes in a decision tree). Naturally, at each stage of questioning, we want the answers to convey information to us. That means that at each stage of questioning, we hope to divide our dataset in such a way to contain only one value of the response variable.

A (data)set which contains only a single (response) value is known as a *pure* set. The algorithm which grows a decision tree recursively splits the training data into purer and purer sets by using “Is $X \leq a$?” type questions on attributes. In other words, we hope to arrive at a correct labelling by reducing labelling possibilities in response to a sequence of “Yes..No” answers.

Below is the pseudocode for growing a decision tree.

1. Put the “best” attribute of the dataset at the root of the tree.
2. Split the training records into subsets. Subsets should be generated in such a way that they are as “pure” as possible according to some chosen measures.
3. Repeat steps 1 and 2 above on each subset generated until we reach a leaf node in all branches of the tree.

5. Attribute Selection

In a dataset which many features, we have to select an attribute to effect the partitioning. The decision tree algorithm uses a *greedy search* algorithm. What this means is that at each stage of the algorithm, **as far as it is possible**, we always choose an attribute which divides the dataset into two partitions which contain only one value of the response variable.

To measure the degree in which a partition contains only one response value, we use measures of *impurity*. The two best known are the **Gini index** and the **entropy**. Definitions are as follows:

Let $p(i|t)$ denote the proportion of training records belonging to class i at node t . Then:

$$Gini(t) = 1 - \sum_{i=1}^c [p(i|t)]^2$$

$$\text{Entropy}(t) = - \sum_{i=1}^c p(i|t) \log_2 p(i|t)$$

Here c is the number of classes and $\log_2 0 = 0$ in entropy calculations.

The best attribute to split the records on at any stage of training is always the one which **maximizes** the average purity of the partitions. This corresponds to asking a very discriminating question, where a response in either direction tells you as much information as you can hope to gain at that stage.

The **gain** Δ is used to determine the goodness of a split at every.

$$\Delta = I(\text{parent}) - \sum_{i=1}^k \frac{N(v_i)}{N} I(v_i),$$

Here $I(\square)$ = impurity measure of a given node;

N = total number of records at the parent node;

k = number of attribute values, and

$N(v_i)$ = number of records in the child node v_i .

The attribute (and test condition) which gives the maximum gain is then chosen to split the training records at the corresponding node.

When entropy is used as the impurity measure, the difference in entropy between the parent node and the child nodes is also known as the **information gain**.

The process above is repeated further down the tree until some threshold conditions are reached.

Example 3: Refer to Example 1 above. Recall that the objective is to classify whether a patient has heart disease. The attributes and attribute values are:

Attribute	Description
<i>fbs</i>	Fasting blood sugar > 120 mg/dl (1 = Yes, 0 = No)
<i>exang</i>	Exercise-induced angina (1 = Yes, 0 = No)
<i>cp</i>	Chest pain type (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)
<i>thalach</i>	Maximum heart rate achieved
<i>Heart</i>	Heart disease (Yes = heart disease, No = normal)

- Compute the Gini index for the overall collection of training records before splitting.
- Compute the Gini index for the *fbs* attribute.
- Compute the Gini index for the *exang* attribute.
- Compute the Gini index for the *cp* attribute using a multiway split.
- Which attribute is better: *fbs*, *exang*, or *cp*?

Solution:

In this example the target variable is *Heart* (Yes/No), which is binary. The first column is the *Patient ID*, which has no bearing on the target variable and hence ignored. The remaining attributes, *fbs*, *exang*, and *cp*, which are categorical, are used to construct the decision tree.

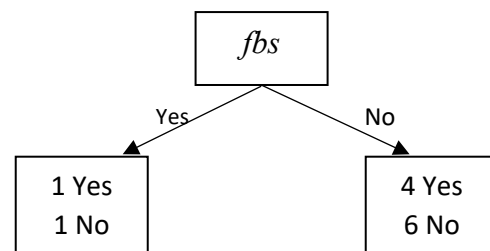
a) Before splitting

<i>Patient ID</i>	<i>fbs</i>	<i>exang</i>	<i>cp</i>	<i>Heart</i>
1	Yes	No	1	No
2	No	Yes	4	Yes
3	No	Yes	4	Yes
4	No	No	3	No
5	No	No	2	No
6	No	No	2	No
7	No	No	4	Yes
8	No	Yes	4	No
9	No	No	4	Yes
10	Yes	Yes	4	Yes
11	No	No	4	No
12	No	No	2	No

$$\text{Gini} = 1 - \left(\frac{5}{12} \right)^2 - \left(\frac{7}{12} \right)^2 = 0.4861$$

b) Split on Attribute *fbs*

<i>Patient ID</i>	<i>fbs</i>	<i>exang</i>	<i>cp</i>	<i>Heart</i>
1	Yes	No	1	No
2	No	Yes	4	Yes
3	No	Yes	4	Yes
4	No	No	3	No
5	No	No	2	No
6	No	No	2	No
7	No	No	4	Yes
8	No	Yes	4	No
9	No	No	4	Yes
10	Yes	Yes	4	Yes
11	No	No	4	No
12	No	No	2	No



$$\text{Gini} = 1 - \left(\frac{1}{2} \right)^2 - \left(\frac{1}{2} \right)^2 = 0.5 \quad \text{Gini} = 1 - \left(\frac{4}{10} \right)^2 - \left(\frac{6}{10} \right)^2 = 0.48$$

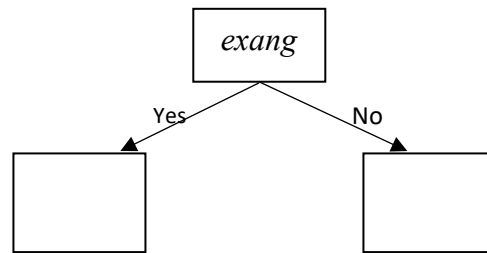
Weighted average Gini index for *fbs* =

$$\frac{10}{12} \times 0.48 + \frac{2}{12} \times 0.5 = 0.4833$$

$$\text{gain} = 0.4861 - 0.4833 = 0.0028$$

c) Split on attribute *exang*

Patient ID	fbs	exang	cp	Heart
1	Yes	No	1	No
2	No	Yes	4	Yes
3	No	Yes	4	Yes
4	No	No	3	No
5	No	No	2	No
6	No	No	2	No
7	No	No	4	Yes
8	No	Yes	4	No
9	No	No	4	Yes
10	Yes	Yes	4	Yes
11	No	No	4	No
12	No	No	2	No



Weighted average Gini index for *exang* =

gain =

d) Split on Attribute *cp*

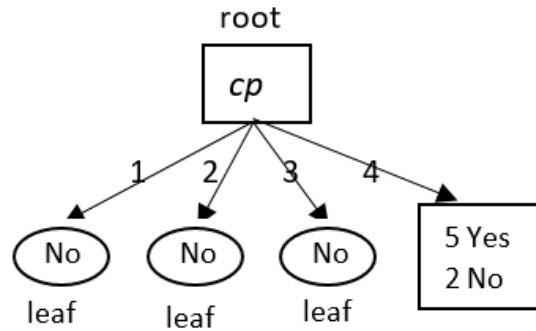
Patient ID	fbs	exang	cp	Heart
1	Yes	No	1	No
2	No	Yes	4	Yes
3	No	Yes	4	Yes
4	No	No	3	No
5	No	No	2	No
6	No	No	2	No
7	No	No	4	Yes
8	No	Yes	4	No
9	No	No	4	Yes
10	Yes	Yes	4	Yes
11	No	No	4	No
12	No	No	2	No

Weighted average Gini index for *cp* =

gain =

- e) The attribute *cp* (chest pain) is better since it has the lowest impurity $Gini = 0.2381$ and the highest gain $\Delta = 0.2479$. Therefore attribute *cp* will be chosen as the root node.

The decision tree with *cp* as the root node is as follows:



Since the values of the Gini index for the three nodes on the left are zero, no further splitting is necessary and they can be converted into leaf nodes.

6. Attribute Selection Using Entropy

We repeat the above *Example 3* but this time using entropy calculations. Recall that entropy at a node t is defined to be:

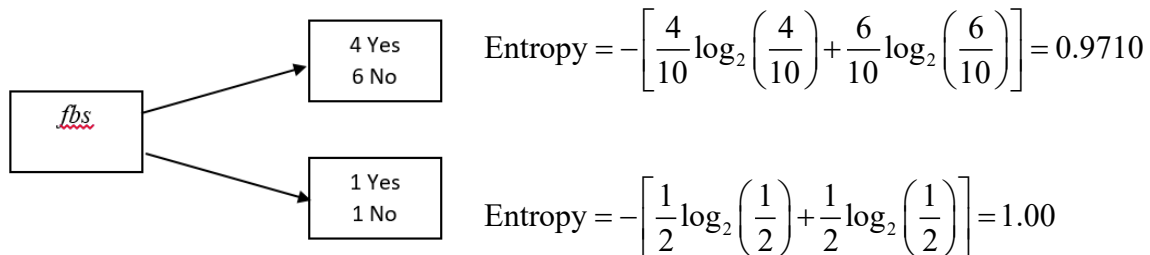
$$\text{Entropy}(t) = -\sum_{i=1}^c p(i|t) \log_2 p(i|t)$$

Example 4:

Parent Node:

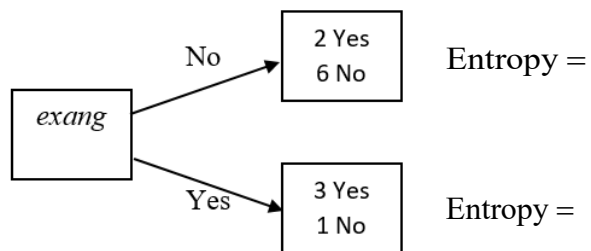
$$\text{Entropy} = -\left[\frac{5}{12} \log_2 \left(\frac{5}{12} \right) + \frac{7}{12} \log_2 \left(\frac{7}{12} \right) \right] = 0.9799$$

Attribute *fbs*:



$$\begin{aligned} \text{average entropy for } fbs &= \frac{10}{12} \times 0.9710 + \frac{2}{12} \times 1.00 \\ &= 0.9758 \end{aligned}$$

Attribute *exang*:



average entropy for *exang* =

Attribute *cp*:

The attribute _____ has the lowest entropy and the largest information gain of _____.

Hence _____ wins the first test, and will be chosen as the root node.

7. More about Gini and Entropy Measures

Recall the definition of the Gini and entropy measures:

$$Gini(t) = 1 - \sum_{i=1}^c [p(i|t)]^2$$

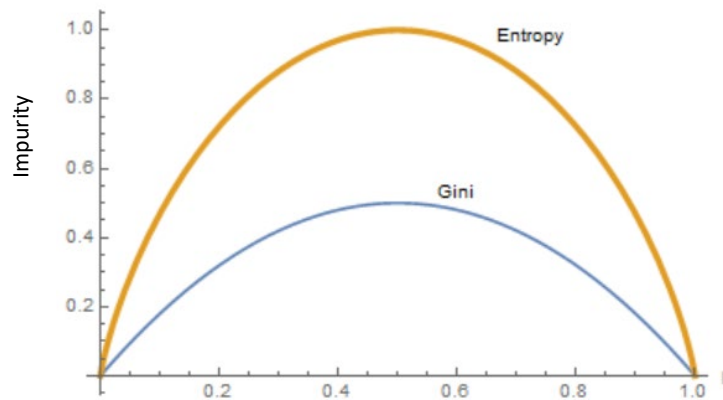
$$Entropy(t) = - \sum_{i=1}^c p(i|t) \log_2 p(i|t)$$

For a binary classification problem, they can be rewritten as:

$$Gini = 1 - [p^2 + (1-p)^2]$$

$$Entropy = -[p \log_2 p + (1-p) \log_2 (1-p)]$$

The graphs of Gini and entropy for the binary classification problem are given below:



Notice that:

- Both the Gini and entropy attain their maximum when $p = 0.5$, that is, when the class distribution is uniform.
- Both measures are zero when $p = 0$ or $p = 1$, that is, when all records belong to the same class.

Notice that the attributes used in building the tree in Example 3 above are all categorical. What if we have an attribute A which is *continuous*-valued? For such a scenario, one way is to implement a binary split, by selecting the best threshold c so that the training records satisfy either $A \leq c$ or $A > c$. The next step is to select the best value for the threshold c which produces the maximum gain or information gain. The computations involved are tedious and are best implemented using a software package.

8. Overfitting

The algorithm for growing decision trees have a tendency to produce very dense and deep trees. These kinds of decision tree do not generalize well to predict data which was not used in its training. Thus, although the resulting decision tree may be able predict almost every label in the **training set** correctly, it performs poorly when used in real life. This is because the resulting complicated decision tree is not generalizing the genuine features in the data, but **memorizing** the training dataset. This is called overfitting.

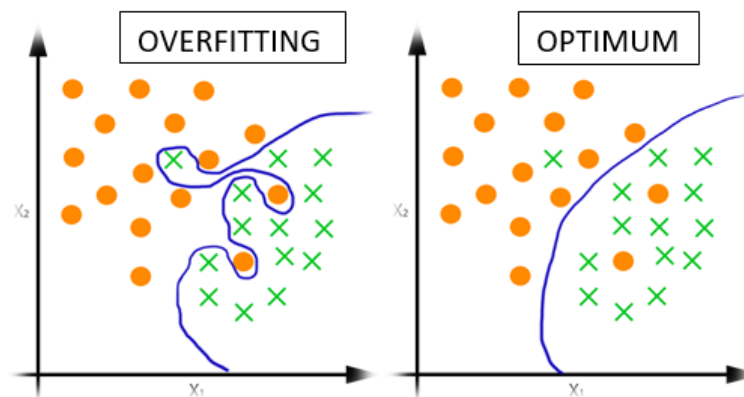
Here are some ways to prevent overfitting or ending up with an overly complicated decision tree.

- Stop when the number of records in a leaf node is less than some user-specified limit
- Stop when the number of records in a node prior to splitting is less than some user-specified limit
- Stop when the attribute values for all training records are identical – in which case no rule can be generated to split them.

An alternative approach is to allow a tree to grow to its full size and then **prune** it back to an optimum size by removing branches in a manner which improves the overall accuracy of the model.

Overfitting

Overfitting happens when the decision tree tries to produce rules to predict outliers and anomalous labels in the training dataset.



(source : <https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19>)

When this happens, the decision tree will tend to give wrong predictions on test data since testing data is supposed to follow the same general distribution of points as the training data and has its own set of outliers.

The complicated decision tree will simply be making lots of wrong predictions on the test data because its rules are too specialised to the training set.

9. Build Decision Tree in KNIME: Case study with UCI dataset

Download the dataset *Heart.csv* from Blackboard. This is adapted from the *Heart* dataset in the UCL Machine Learning Repository (<http://archive.ics.uci.edu/ml/datasets/heart+Disease>). The goal is to classify whether a patient has heart disease. In this case, we consider the “Positive class: Patient **with** heart disease” and the “Negative class: Healthy patient **without** heart disease”. The attributes and their description are:

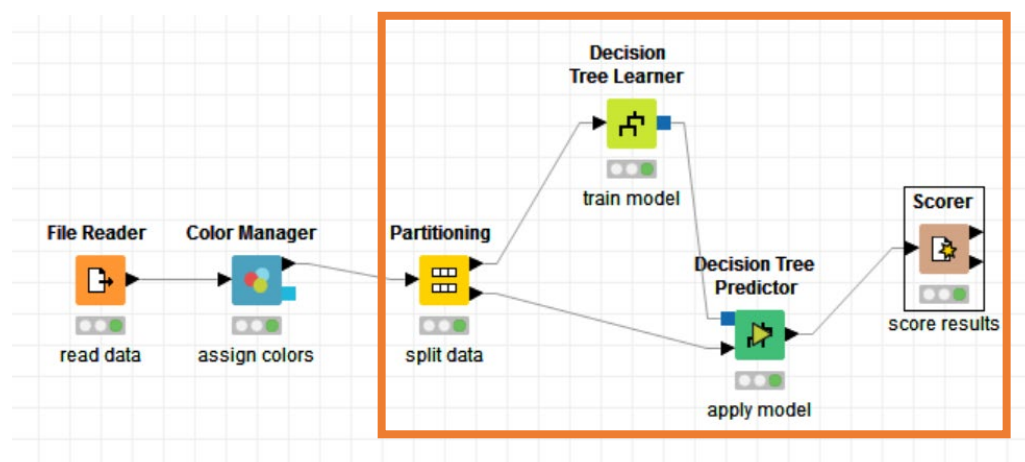
Attribute	Description
<i>fbs</i>	Fasting blood sugar > 120 mg/dl (1 = Yes, 0 = No)
<i>exang</i>	Exercise-induced angina (1 = Yes, 0 = No)
<i>cp</i>	Chest pain type (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)
<i>thalach</i>	Maximum heart rate achieved
<i>Heart</i>	Heart disease (Yes = heart disease, No = normal)

Construct a decision tree model to predict whether a patient is at risk of getting heart disease.

Classification model in KNIME use a Learner-predictor motif. The process overview for generating a classification model in KNIME works as follows:

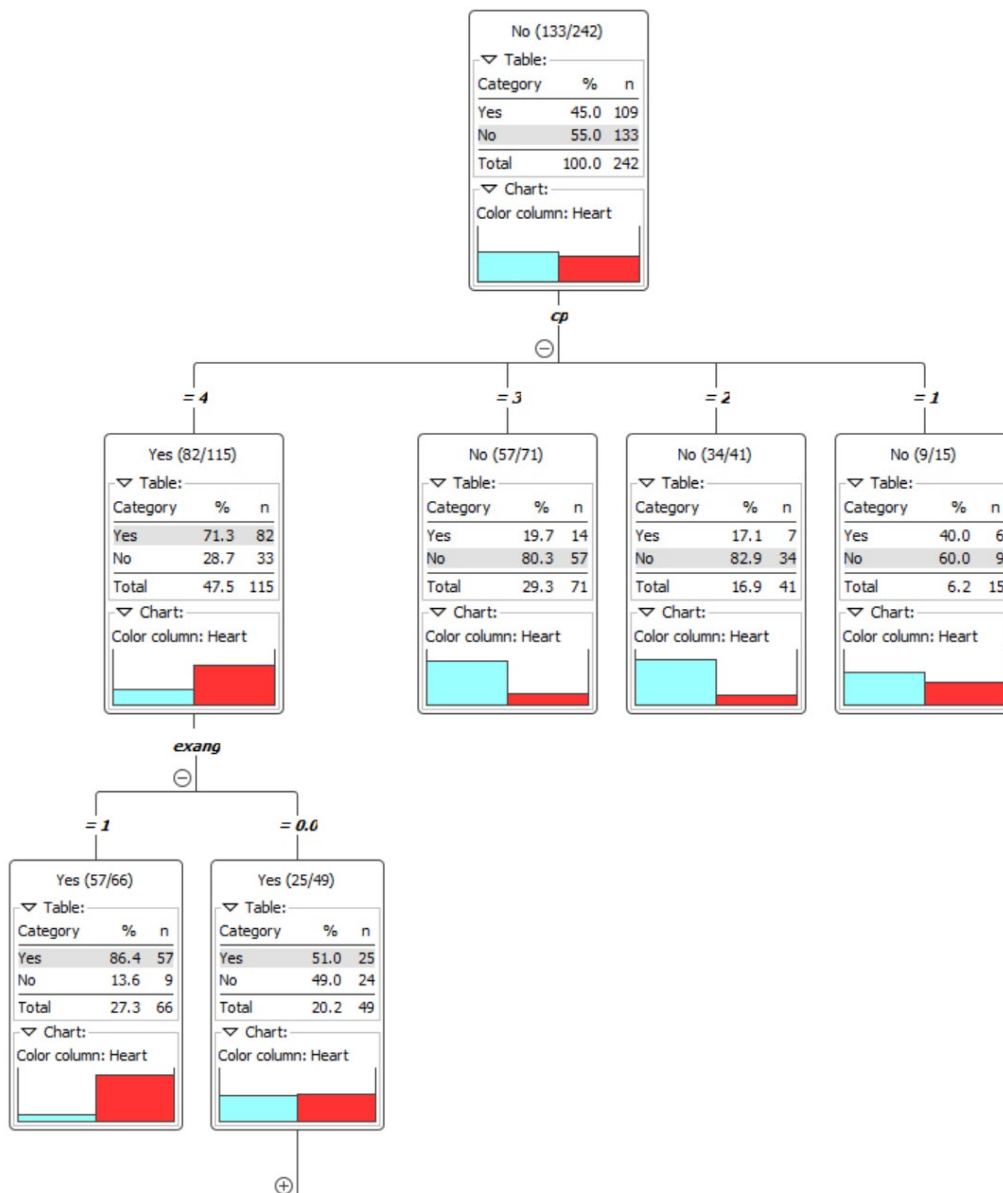
1. Partition the original data to the training set and test set.
2. The Learner node trains the model with its input data: The training set (first partition) is used to build (or train) a model.
3. The Predictor node applies the model to a different subset of data: Takes the decision tree model from Step 2 and applies it to the test set (second partition).
4. The Scorer node compare predicted results to known truth (target variable) in order to evaluate model quality:
 - Confusion matrix shows the distribution of model errors
 - An accuracy statistics table provides a detailed analysis of model quality.

We construct a KNIME workflow as follows:



We are now ready to view the decision tree and analyse the tree generated.

Decision Tree View from KNIME:



A few observations are as follows:

- The target variable is *Heart*, which has two classes (*Heart = Yes* or *Heart = No*). The root node contains 242 records in the training data. Distribution of *Heart* is 45% *Yes* and 55% *No*.
- The first split occurs with the attribute *cp* (type of chest pain).
- 17.1% of patients in $cp=2$ (atypical chest pain) and 19.7% of patients in $cp=3$ (non-anginal pain) developed heart disease.
- Patients in $cp=4$ (asymptomatic) and $exang=1$ (exercise-induced angina) end up in a leaf node with a high risk (86.4%) of developing heart disease.
- The overall accuracy of the model when scored using the test data is 72.1%, with a sensitivity of 60.0% and precision of 78.3% (to be discussed below).

Confusion Matrix and Accuracy Statistics from KNIME:

The confusion matrix and accuracy statistics are given below. These will be discussed in the next Section.

Row ID	No	Yes
No	26	5
Yes	12	18

Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy
No	26	12	18	5	0.839	0.684	0.839	0.6	0.754	?
Yes	18	5	26	12	0.6	0.783	0.6	0.839	0.679	?
Overall	?	?	?	?	?	?	?	?	?	0.721

10. Evaluation of Performance of Classification Models

Recall that in building a decision tree model, a training dataset consisting of records whose class labels are known is used to build the model, which is subsequently applied to the test dataset. One way to evaluate the performance of the model is to consider the count of test records correctly and incorrectly predicted by the model. These counts are summarized in a table called the **confusion matrix**. The confusion matrix for a binary classification problem is given below:

		<i>Predicted</i>	
		Yes	No
<i>Actual</i>	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

In binary classification, a single prediction has four different possible outcomes:

- **True Positive (TP):** Outcome is correctly predicted as positive when it is actually positive;
- **True Negative (TN):** Outcome is correctly predicted as negative when it is actually negative;
- **False Positive (FP):** Outcome is incorrectly predicted as positive when it is actually negative;
- **False Negative (FN):** Outcome is incorrectly predicted as negative when it is actually positive;

Some performance measures used to determine the performance of a classification model are as follows:

- $\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + FP + TN + FN}$
- $\text{Error Rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{FP + FN}{TP + FP + TN + FN} = 1 - \text{Accuracy}$

- Sensitivity = $\frac{TP}{TP + FN}$, the proportion of positive outcomes correctly predicted by the model;
- Specificity = $\frac{TN}{FP + TN}$, the proportion of negative outcomes correctly predicted by the model.

Two other widely used measures are Precision and Recall, where:

- Precision = $\frac{TP}{TP + FP}$, proportion of outcomes labelled as positive are genuinely so.
- Recall = $\frac{TP}{FN + TP}$, proportion of positive outcomes which are successfully identified (labelled as such).

Note that *recall*, is the same as the *sensitivity* of the model.

Precision and recall can sometimes be summarized into another measure called the *F*-measure, defined as the harmonic mean between recall and precision, that is,

$$F = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two numbers. Hence, a high value of *F*-measure suggests that both precision and recall are reasonably high.

Example 5: The following confusion matrix gives the results of a model used to predict whether a bank customer is likely to default on loans. Calculate the accuracy, error rate, sensitivity, specificity, precision and recall.

		<i>Predicted</i>	
		No Default	Default
<i>Actual</i>	No Default	40	10
	Default	20	30

Solution:

		<i>Predicted</i>		
		No Default	Default	Total
<i>Actual</i>	No Default	40	10	50
	Default	20	30	50
	Total	60	40	100

accuracy =

error rate =

sensitivity =

precision =

recall = sensitivity =

Example 6: The confusion matrix of the model used to predict whether a patient is at risk of getting heart disease is reproduced below. Calculate the accuracy, error rate, sensitivity, specificity, precision and recall. Show your calculation step clearly. Are your results consistent with those given in the accuracy statistics table in KNIME?

Row ID	↓ No	↓ Yes
No	26	5
Yes	12	18

KNIME accuracy statistics output:

Row ID	↓ TruePo...	↓ FalsePo...	↓ TrueNe...	↓ FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy
No	26	12	18	5	0.839	0.684	0.839	0.6	0.754	?
Yes	18	5	26	12	0.6	0.783	0.6	0.839	0.679	?
Overall	?	?	?	?	?	?	?	?	?	0.721

Solution:

TUTORIAL 7

1. The table below gives the training data on bank customers who have taken out a loan. The attributes are *Age* (≤ 30 , 31-40, > 40), *Income* (low, medium, high), *Own-Home* (own, rent), and *Approved* (Yes, No). A classification tree will be built using *Approved* as the class variable.

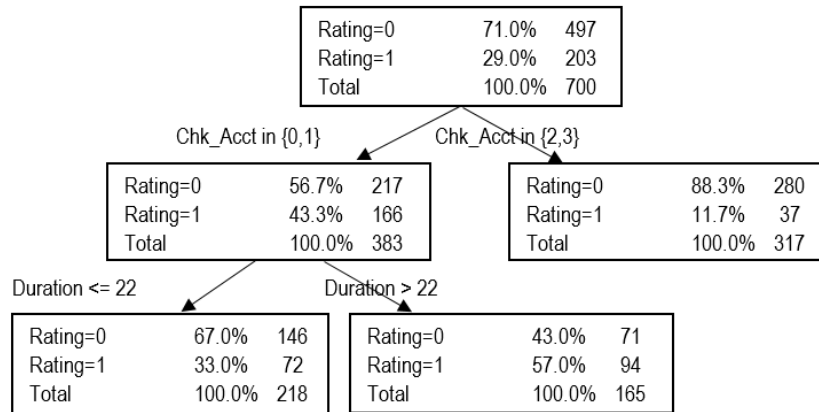
ID	Age	Income	Own_Home	Approved
1	≤ 30	high	own	Yes
2	≤ 30	high	rent	Yes
3	31-40	high	own	Yes
4	> 40	medium	rent	No
5	> 40	low	rent	No
6	31-40	low	rent	No
7	≤ 30	low	rent	No
8	≤ 30	medium	own	Yes
9	> 40	low	own	Yes
10	≤ 30	medium	own	Yes
11	31-40	medium	rent	Yes
12	31-40	high	rent	Yes

- a) Calculate the Gini index for the data before splitting.
 - b) Calculate the Gini index for the *Age* attribute.
 - c) Calculate the Gini index for the *Income* attribute.
 - d) Calculate the Gini index for the *Own-Home* attribute.
 - e) Select the best attribute for the root node.
 - f) Construct a decision tree.
2. Consider the training set below with attributes x_1 (Yes, No), x_2 (Yes, No) and *Class* (Pos, Neg). A classification tree is to be built using *Class* as the target variable.

ID	x_1	x_2	Class
1	Yes	Yes	Pos
2	Yes	Yes	Pos
3	Yes	No	Neg
4	No	No	Pos
5	No	Yes	Neg
6	No	Yes	Neg

- a) Calculate the entropy of this dataset.
- b) Calculate the information gain for the attributes x_1 and x_2 .
- c) Which attribute, x_1 or x_2 , should be selected as the root node of a decision tree?

3. The diagram below gives the results of applying a classification tree algorithm to a training set ($n = 700$) of bank customers who have taken out a loan. The attributes are *Chk_Acct* (checking account status), *Duration* (duration of credit in months), and *Rating* (0 = *bad credit rating*, 1 = *good credit rating*).



- a) Convert the tree into a set of rules for classifying loan applicants (there is one rule for each leaf node). Each rule should include a probability.
- b) Use the decision tree to classify (using a threshold of 0.5) the unseen records in the table below:

Chk_Acct	Duration	Rating
1	24	
3	36	

- c) Suppose a test set is given as follows. Use the decision tree in part (a) above to classify the records.

Chk_Acct	Duration	Rating	Predicted Rating
2	27	1	
3	24	0	
0	42	1	
0	6	1	
0	24	0	
1	24	1	
3	12	1	
3	12	1	
2	10	1	
0	24	0	

- d) Use your results in part (c) above to construct a confusion matrix and estimate the overall accuracy of the decision tree.

4. [📄] This Ecoli dataset (Resource: UCI Machine Learning Repository) contains protein localization sites. The meaning of the attributes are given as follows:

Attribute	Meaning
SeqName	Accession number for the SWISS-PROT database
mcg	McGeoch's method for signal sequence recognition
gvh	von Heijne's method for signal sequence recognition
lip	von Heijne's Signal Peptidase II consensus sequence score
chg	Presence of charge on N-terminus of predicted lipoproteins
aac	Score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins
alm1	Score of the ALOM membrane spanning region prediction program
alm2	Score of ALOM program after excluding putative cleavable signal regions from the sequence
Class	Localization site ('cp' (cytoplasm), 'im' (inner membrane), 'pp' (periplasm), 'om' (outer membrane))

Build a classification model to predict the localization sites using the decision tree method with 'Gain ratio' (no pruning). Reduced error pruning is checked to cut the tree in post-processing by the algorithm in KNIME. In the Partitioning node, choose the size of first partition as relative 80% stratified sampled with the 'Class' attribute using the random seed of '1234'. Restrict the minimum number of records per node to be 15.

- Which is the target variable?
 - Explain why a regression model is not selected here.
 - Make use of the confusion matrix generated by the classification workflow to compute the overall accuracy of the decision tree. Compare that with the accuracy statistics generated by KNIME.
 - From the confusion matrix generated by the classification workflow, which is/are the localization site(s) with no False Negatives produced? Interpret your answer in context.
 - Based on the decision tree produced above, summarize the tree as a set of classification rules of the form "If... Then..." for each localization site.
 - Use the decision tree above to predict the localization site of a record with the following attribute values: SeqName = AAT_ECOLI, mcg = 0.5, gvh = 0.6, lip = 0.48, chg = 0.5, acc = 0.6, alm1 = 0.5, alm2 = 0.35.
 - Name one way to avoid overfitting when building a decision tree.
 - Build a classification model to predict the localization site using the decision tree method with 'Gini Index', keeping all other parameter settings similar as above.
 - Observe the model and compare its performance in terms of accuracy of classification with the decision tree above.
 - Compare and discuss on the first split and second split of the two decision trees generated.
5. [📄] Some researchers in the USA Forensic Science Service conducted a study to classify the type of glass in terms of their oxide content. The data is contained in the *GlassType.csv* dataset. It contains 214 records with 10 attributes:

Variable	Description
ID	Glass Identification
RI	Refractive index
Na	Sodium
Mg	Magnesium
Al	Aluminium
Si	Silicon
K	Potassium
Ca	Calcium
Ba	Barium
Fe	Iron
Type	<i>Window</i> (building windows float processed, building windows non-float processed, vehicle window float processed, vehicle window non-float processed) <i>Non-window</i> (containers, tableware, headlamps)

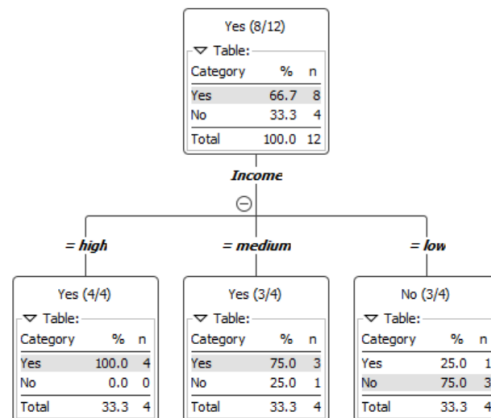
Download *GlassType.csv* from BB and load it into KNIME.

Build a classification model to predict *Type* (*Window* or *Non-window*) using the decision tree method (unpruned). In the **Partitioning** node choose **relative 80%** for the first partition by **Draw randomly** using **random seed 12345**. In the **Decision Tree Learner** node, use the settings: **Quality measure: Gini index**, **Pruning method: No pruning**, enable **Reduced Error Pruning**, **Min number of records per node: 10**. Keep the default settings in others.

- Which is the target variable?
- How many records are there in the training data and test data?
- Which variable is the first split in the tree made?
- What is the number of leaf nodes in the tree (fully expanded)?
- Use the decision tree generated by KNIME to predict the type of glass: $RI = 1.52$, $Na = 13$, $Mg = 3.1$, $Al = 1.36$, $Si = 73$, $K = 0.5$, $Ca = 9$, $Ba = 0.11$, $Fe = 0$.
- Suppose we define Positive to mean a glass is '*Window*' type. From the confusion matrix produced, what is the number of false positives? What does the answer mean in the context of the problem?
- What is the sensitivity of the model in predicting a glass type is '*Non-Window*'?

ANSWERS

1.
 - a) Gini ≈ 0.444
 - b) Gini (*Age*) ≈ 0.369
 - c) Gini (*Income*) ≈ 0.25
 - d) Gini (*Own-Home*) ≈ 0.286
 - e) Best attribute is *Income*, with largest gain 0.194
 - f)



2.
 - a) Entropy = 1
 - b) Entropy(x_1) = 0.9183; Info gain(x_1) = 0.0817
Entropy(x_2) = 1; Info gain(x_2) = 0
 - c) x_1 is the better attribute
3.
 - a)

If *Chk_Acct* in {0,1} and *Duration* ≤ 22 then *Rating* = 0 with probability 0.670

If *Chk_Acct* in {0,1} and *Duration* > 22 then *Rating* = 1 with probability 0.570

If *Chk_Acct* in {2,3} then *Rating* = 0 with probability 0.883

b)

Chk_Acct	Duration	Rating
1	24	1
3	36	0

c)

Chk_Acct	Duration	Rating	Predicted Rating
2	27	1	0
3	24	0	0
0	42	1	1
0	6	1	0
0	24	0	1
1	24	1	1
3	12	1	0
3	12	1	0
2	10	1	0
0	24	0	1

d)

Confusion Matrix		Predicted	
		0=bad	1=good
Actual	0=bad	1	2
	1=good	5	2

Overall accuracy = 30.0%

4. a) Class
b) Target is categorical
c) Accuracy statistics – 93.2%

Row ID	cp	im	om	pp
cp	26	0	1	1
im	0	16	0	0
om	0	0	4	0
pp	1	1	0	9

Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specificity	F-meas...	Accuracy	Cohen'...
cp	26	1	30	2	0.929	0.963	0.929	0.968	0.945	?	?
im	16	1	42	0	1	0.941	1	0.977	0.97	?	?
om	4	1	54	0	1	0.8	1	0.982	0.889	?	?
pp	9	1	47	2	0.818	0.9	0.818	0.979	0.857	?	?
Overall	?	?	?	?	?	?	?	?	?	0.932	0.898

- d) im and om have no FN.
e) If $alm1 > 0.6$ then class = im with prob 0.951
If $alm1 \leq 0.6$ and $aac > 0.64$ then class = om with prob 0.875
If $alm1 \leq 0.6$ and $aac \leq 0.64$ and $gvh > 0.58$ then class = pp with prob 0.941
If $alm1 \leq 0.6$ and $aac \leq 0.64$ and $gvh \leq 0.58$ then class = cp with prob 0.934
f) Classified as pp
g) Pruning
h) i. Accuracy improves to 94.9%
ii. Both first split at alm1 with different criteria. Gini splits at alm1 (0.575).
Second split for Gini is gvh instead of aac.

5. a) Type (*Window/Non-Window*)
b) 171, 43
c) Mg
d) 3
e) Window
f) 1 FP. There is 1 record which is incorrectly predicted as '*Window*' type when it is actually '*Non-Window*' type.
g) 92.9%

Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specificity	F-meas...	Accuracy	Cohen'...
Window	29	1	13	0	1	0.967	1	0.929	0.983	?	?
Non-Window	13	0	29	1	0.929	1	0.929	1	0.963	?	?
Overall	?	?	?	?	?	?	?	?	?	0.977	0.946

LAB 7A : Classification by Decision Tree using KNIME (Heart dataset)

Learning Objectives:

1. Build a classification model by Decision Tree using KNIME.
2. Interpret decision tree result through the use of visualization plots in KNIME.

Task: The Heart Dataset

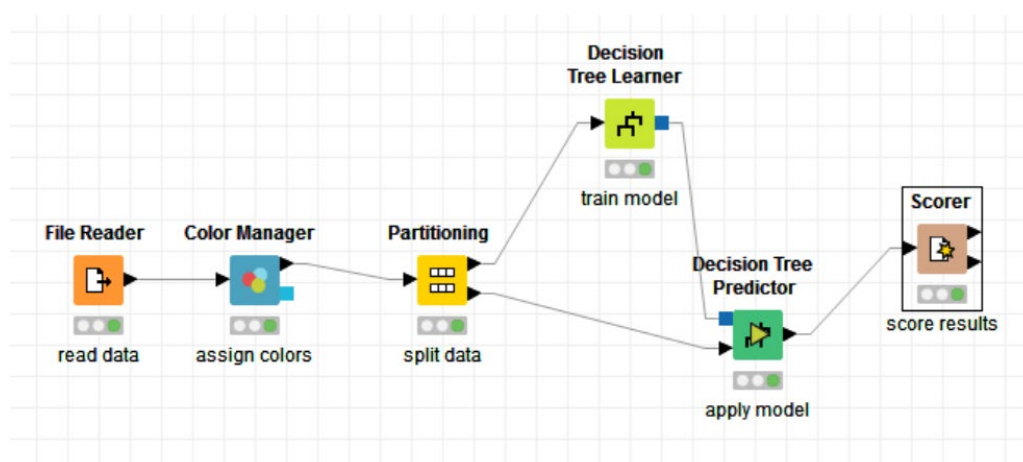
The heart dataset *heart.csv* is adapted from the *Heart* dataset in the UCL Machine Learning Repository (<http://archive.ics.uci.edu/ml/datasets/heart+Disease>).

The goal is to classify whether a patient has heart disease. In this case, we consider the “Positive class: Patient **with** heart disease” and the “Negative class: Healthy patient **without** heart disease”. The attributes and their description are:

Attribute	Description
<i>fbs</i>	Fasting blood sugar > 120 mg/dl (1 = Yes, 0 = No)
<i>exang</i>	Exercise-induced angina (1 = Yes, 0 = No)
<i>cp</i>	Chest pain type (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)
<i>thalach</i>	Maximum heart rate achieved
<i>Heart</i>	Heart disease (Yes = heart disease, No = normal)

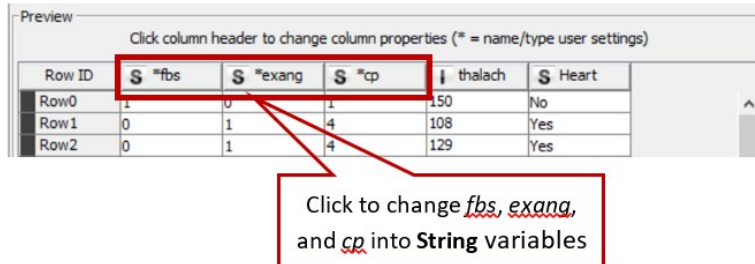
Problem: We want to construct a decision tree model to predict whether a patient is at risk of getting heart disease.

Your completed workflow might appear as follows:

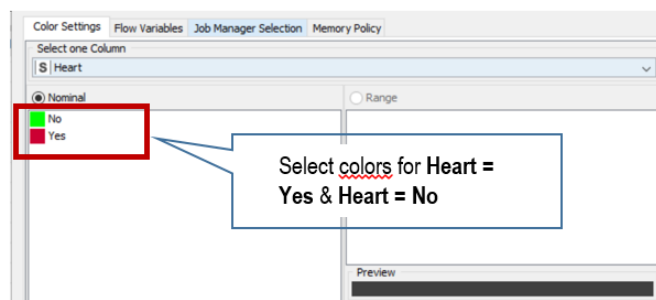


Create a decision tree model using KNIME (The heart dataset)

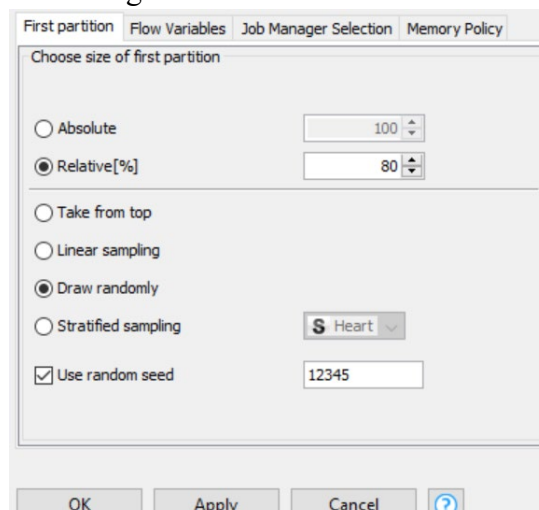
1. Start a new workflow in your local workspace.
2. Use a **File Reader** node to import the *Heart.csv* dataset. Use the configuration dialog box to change the variables *fbs*, *exang* and *cp* into **String** variables, as follows:



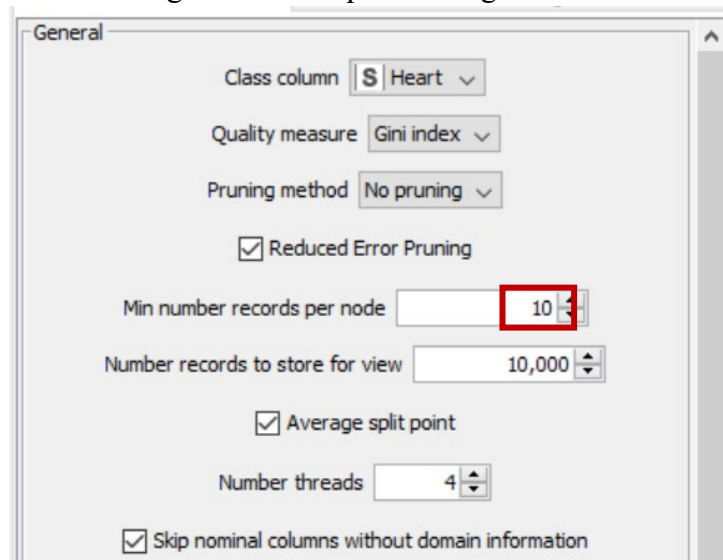
3. Connect a **Color Manager** node to the **File Reader** node. This will color-code the target variable *Heart* so that it is easier for us to visualize the results later on in the workflow. Use the configuration dialog box so that those with *Heart=Yes* are colored red and those with *Heart=No* are colored green, as follows:



4. Connect a **Partitioning** node to the **Color Manager** node. The Partitioning node is used to split the data into *training* and *testing* data. Training data is used to build the decision tree, and test data is used to evaluate the model on new data. In the Configure Dialog of the Partitioning node, choose **Relative[%]80**, **Draw randomly**, and **Use random seed 12345**. This will randomly select 80% of the data as the training data and the remaining 20% as the test data. The random seed is chosen so that everyone will get the same training and test data sets to train and test the decision tree classifier.




5. Connect a **Decision Tree Learner** node to the first output of the **Partitioning** node. This will generate the decision tree classifier using the training data. In the Configure Dialog, change the **Min number of record per node** to **10**. This is to tell the algorithm that a node with this number of records can no longer split. This serves as a stopping criterion for the algorithm. Complete dialog box as follows.



6. Connect a **Decision Tree Predictor** to the second output of the **Partitioning** node and to the output from the **Decision Tree Learner** node. This node will apply the trained model to the test data.
7. Execute the workflow. Right-click on the **Decision Tree Predictor** node and select “View: Decision Tree View” to see the generated decision tree (given below).
8. Connect a **Scorer** node to the output of the **Decision Tree Predictor** to generate the Confusion Matrix and other evaluation measures.
9. Save your workflow.

CHECKPOINT

Decision Tree Learner node configuration:

- #1. What is the class column for?
- #2. What are the possible quality measures available to build a decision tree?
- #3. Find out from the help button  in the configuration setting dialog box, on the following dialog options:

Pruning method, Reduced Error Pruning, Min number records per node.

Explore the output ports to answer the following:

- #4. What observations can we make from the decision tree (View: Decision Tree View)?
- #5. How can we evaluate the performance of the decision tree?

LAB 7B : Classification by Decision Tree using KNIME (pima dataset)

Learning Objectives:

1. Build a classification model by Decision Tree using KNIME.
2. Interpret decision tree result through the use of visualization plots in KNIME.

Task: The Pima Dataset

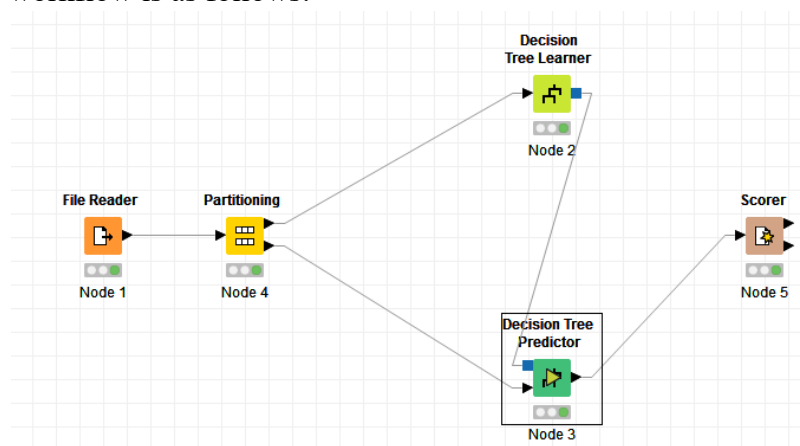
The pima dataset *Pima.csv* is adapted from the Pima Indians diabetes dataset available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.names>).

Patients in this dataset consist of Pima Indian women at least 21 years of age and living near Phoenix, Arizona, USA. There are 768 rows with two classes (0 = normal, 1 = at risk of developing diabetes). The attributes are:

- *pregnancy* = number of times pregnant
- *glucose* = Plasma glucose concentration
- *BP* = Diastolic blood pressure (mmHg)
- *triceps* = Tricep skin fold thickness (mm)
- *Insulin* = 2-hour serum insulin (μ U/ml)
- *BMI* = Body mass index (kg / m^2)
- *DPF* = Diabetes pedigree function
- *age* = Age (years)

Problem: We want to construct a decision tree model to predict whether a patient is at risk of developing diabetes.

The completed workflow is as follows:

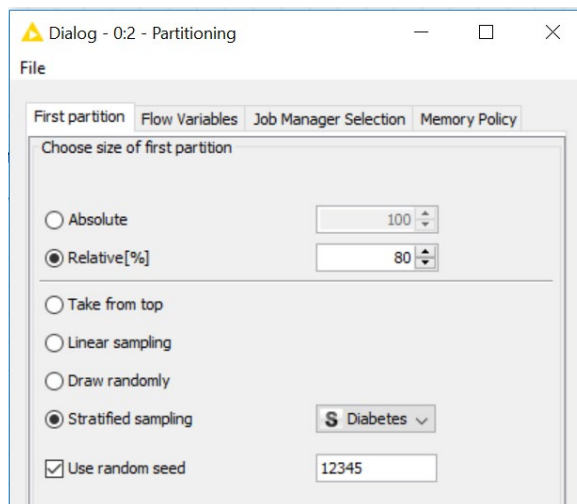


Create a decision tree model using KNIME (The pima dataset)

1. Start a new workflow in your local workspace.
2. Use a **File Reader** node to import the *Pima.csv* dataset. Use the configuration dialog box to change the variable *Diabetes* into a **String** variable:

Row ID	pregna...	glucose	BP	triceps	insulin	BMI	DPF	Age	*Diabetes
Row0	6	148	72	35	0	33.6	0.627	50	1
Row1	1	85	66	29	0	26.6	0.351	31	0
Row2	8	183	64	0	0	23.3	0.672	32	1
Row3	1	89	66	23	94	28.1	0.167	21	0
Row4	0	137	40	35	168	43.1	2.288	33	1
Row5	5	116	74	0	0	25.6	0.201	30	0
Row6	3	78	50	32	88	31	0.248	26	1

3. Connect a **Partitioning** node. In the Configure Dialog of the Partitioning node, choose **Relative[%]**, **Stratified sampling**, and **Use random seed 12345**. The random seed is chosen so that everyone will get the same training and test data sets to train and test the decision tree classifier.



4. Connect a **Decision tree Learner** node to the first output of the **Partitioning** node. This will generate the decision tree classifier using the training data.
5. Connect a **Decision tree Predictor** to the second output of the **Partitioning** node and to the output from the **Decision tree Learner** node. This node will apply the trained model to the test data.
6. Connect a **Scorer** node to the output of the **Decision tree Predictor** to generate the confusion matrix and accuracy statistics.
7. Execute the workflow.
8. Save your workflow.

CHECKPOINT

- #1. Why do we need to change the data type of '*Diabetes*' to String?
- #2. What observations can we make from the decision tree (View: Decision Tree View)?
- #3. What is the performance of the decision tree?