Discovering Knowledge in Data

Daniel T. Larose, Ph.D.

Chapter 8
Decision Trees

Prepared by James Steck and Eric Flores

Decision Trees

- Decision Trees
  - Popular classification method in data mining.
  - Decision Tree is a collection of <u>decision nodes</u>, connected by <u>branches</u>, extending downward from <u>root node</u> to terminating <u>leaf nodes.</u>
  - Beginning with root node, attributes tested at decision nodes, and each possible outcome results in a branch.
  - Each branch leads to a decision node or a leaf node.

Decision Trees *(cont'd)*

- Example
  - *Credit Risk* is the target variable
  - Customers are classified as either "Good Risk" or "Bad Risk"
  - Predictor variables are *Savings* (Low, Med, High), *Assets* (Low, High) and *Income*

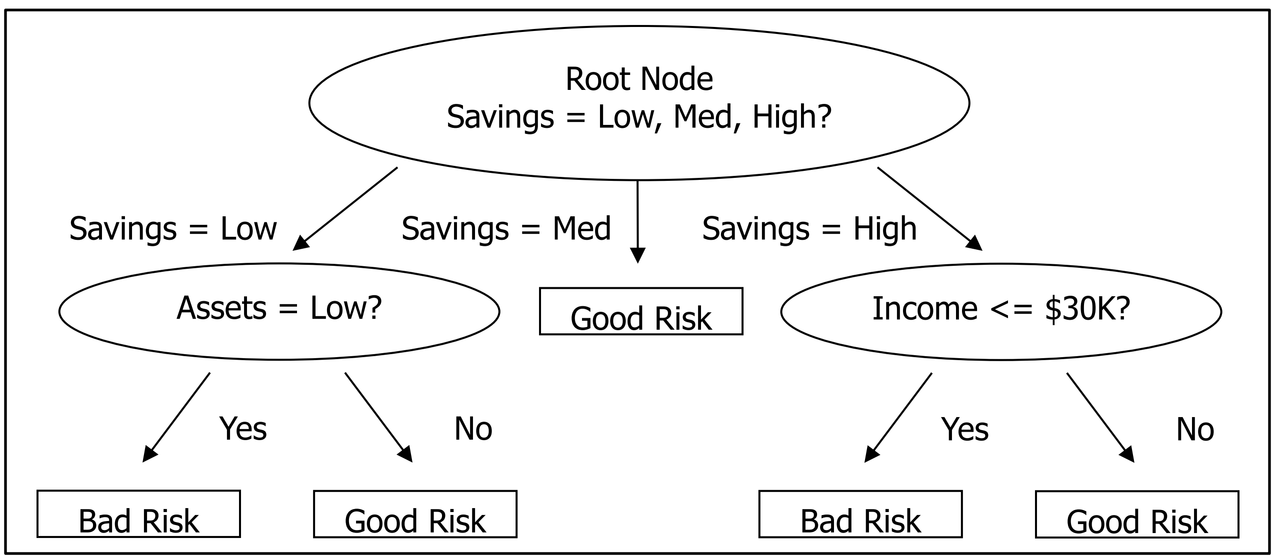


Figure 8.1

Decision Trees *(cont'd)*

- Highest-level decision node is root node and tests whether record has *Savings* = "Low", "Med", or "High"
- Records are firt split according to value of *Savings*.
- Records with *Savings* = "Low" go down leftmost branch to next decision node
- Records with *Savings* = "Med" proceed down middle branch to leaf node. This terminates branch with all records *Savings* = "Med" classified as "Good Risk". There is no need for another decision node, because our knowledge that the customer has medium savings predicts good credit.
- Records with *Savings* = "High" go down rightmost branch to another decision node

Decision Trees *(cont'd)*

- Records with *Savings* = "Low" tested at second-level decision node to determine whether *Assets* = "Low"
- Those with low assets classified "Bad Risk, while others classified "Good Risk"
- Second-level decision node in right branch tests whether customers with *Savings* = "High" have *Income* <= $30,000
- Those with *Income* less than or equal to $30,000 classified "Bad Risk". Others classified "Good Risk"
- If no further splits possible, algorithm terminates

Decision Trees *(cont'd)*

- A branch may terminate at a <u>pure leaf node</u>. This means that the subset of records corresponding to that leaf node all have the <u>same target class value.</u>
- A <u>diverse leaf node</u> has records with different target class values ("Good Risk" and "Bad Risk"). Algorithm is possibly unable to split.
- For example, consider the subset of records with *Savings* = "High" and *Income* <= $30,000. Suppose there are 5 such records and all of them have *Assets* = "Low". Suppose the corresponding leaf node contains 2 "Good Risk" and 3 "Bad Risk" records.
- All these records contain the same predictor values. There is no way to split them further to obtain a pure leaf node.
- In this case, the leaf node is classified as "Bad Risk" with 60% (3/5 records) confidence.

Requirements for using Decision Trees

- Decision Tree is a supervised classification method
- The target variable must be categorical, not continuous.
- Pre-classified target variable must be included in training set
- Decision trees learn by example, so training set should be rich and contain records with varied attribute values
- If training set systematically lacks definable subsets, classification becomes problematic
- There are different measures for leaf node purity
- Classification and Regression Trees (CART) and C4.5 are two leading algorithms used in data mining for constructing decision trees (use different measures for leaf node purity)

Classification and Regression Trees

- Classification and Regression Trees (CART) developed by Breiman, 1984
- Splits at decision nodes are binary, resulting in two branches
- CART recursively partitions data into subsets with similar values for target variable
- Algorithm grows tree by evaluating all predictor variables and choosing optimal split according to "goodness" of candidate split $\Phi(s|t)$

Classification and Regression Trees (cont'd)

Let $\Phi(s \mid t)$ be a measure of the "goodness" of a candidate split s at node t, where

$$\Phi(s \mid t) = 2P_L P_R \underbrace{\sum_{j=1}^{\#classes} \left| P(j \mid t_L) - P(j \mid t_R) \right|}_{Q(s \mid t)}$$

and where,

t_L = left child node of node t

t_R = right child node of node t

$$P_L = \frac{\text{number of records at } t_L}{\text{number of records in training set of the parent}}$$

$$P_R = \frac{\text{number of records at } t_R}{\text{number of records in training set of the parent}}$$

$$P(j \mid t_L) = \frac{\text{number of class } j \text{ records at } t_L}{\text{number of records at } t_L}$$

$$P(j \mid t_R) = \frac{\text{number of class } j \text{ records at } t_R}{\text{number of records at } t_R}$$

- Optimality measure maximizes split over all possible splits at node *t*

Classification and Regression Trees *(cont'd)*

- Example
  - Predict whether customer is classified "Good" or "Bad" credit risk using three predictor fields (Savings, Assets, Income) according to data in Table 8.2
  - All records enter root node, and CART evaluates possible binary splits

Table 8.2

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	High	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

Classification and Regression Trees (*cont'd*)

- Splits evaluated for *Savings*, *Assets*, and *Income* in Table 8.3
- *Income* is numeric. CART identifies possible splits based on values it contains
  - Alternatively, the analyst could have categorized it beforehand
- Nine candidate splits, for *t* = root node

Table 8.3

Candidate Split	Left Child Node t_L	Right Child Node t_R
1	Savings = Low	Savings = {Medium or High}
2	Savings = Medium	Savings = {Low or High}
3	Savings = High	Savings = {Low or Medium}
4	Assets = Low	Assets = {Medium or High}
5	Assets = Medium	Assets = {Low or High}
6	Assets = High	Assets = {Low or Medium}
7	Income <= $25K	Income > $25K
8	Income <= $50K	Income > $50K
9	Income <= $75K	Income > $75K

Classification and Regression Trees (*cont'd*)

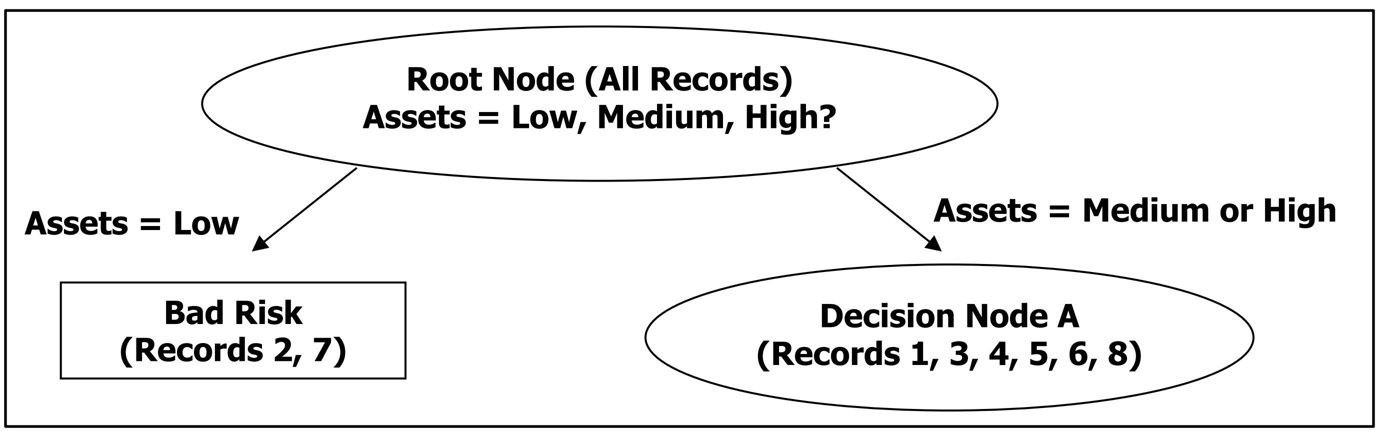
- Values for components of optimality measure candidate splits, t = root node

Table 8.4

Split	P_L	P_R	$P(j \mid t_L)$	$P(j \mid t_R)$	$2P_L P_R$	$Q(s \mid t)$	$\Phi(s \mid t)$
1	.375	.625	G: .333 B: .667	G: .8 B: .2	0.46875	0.934	0.4378
2	0.375	0.625	G: 1 B: 0	G: 0.4 B: 0.6	0.46875	1.2	0.5625
3	0.25	0.75	G: 0.5 B: 0.5	G: 0.667 B: 0.333	0.375	0.334	0.1253
4	0.25	0.75	G: 0 B: 1	G: 0.833 B: 0.167	0.375	1.667	0.6248
5	0.5	0.5	G: 0.75 B: 0.25	G: 0.5 B: 0.5	0.5	0.5	0.25
6	0.25	0.75	G: 1 B: 0	G: 0.5 B: 0.5	0.375	1	0.375
7	0.375	0.625	G: 0.333 B: 0.667	G: 0.8 B: 0.2	0.46875	0.934	0.4378
8	0.625	0.375	G: 0.4 B: 0.6	G: 1 B: 0	0.46875	1.2	0.5625
9	0.875	0.125	G: 0.571 B: 0.429	G: 1 B: 0	0.21875	0.858	0.1877

Classification and Regression Trees (*cont'd*)

- Optimality measure <u>maximized</u> to 0.6248, when *Assets* = "Low" (Left branch), *Assets* = "Medium or High" (Right branch)
- Left branch terminates to pure leaf node; both records have target value = "Bad Risk"
- Right branch diverse and calls for further partitioning



Classification and Regression Trees *(cont'd)*

- Behavior of Optimality Measure
  - Observe values of Optimality Measure components
  - Recall the component values from last example
- When is Optimality Measure large?
  - Measure $\Phi(s|t)$ is large when its main components are large.

(1) Larger values of $\Phi(s \mid t)$ tend to be associated with larger values of its main components: $2P_L P_R$ and $\sum_{j=1}^{\#classes} \left| P(j \mid t_L) - P(j \mid t_R) \right|$

Classification and Regression Trees (*cont'd*)

- When is component *Q*(s|t) large?

$$(2)\ \text{Let } Q(s \mid t) = \sum_{j=1}^{\#classes} \left| P(j \mid t_L) - P(j \mid t_R) \right|$$

$Q(s \mid t)$ is large when the distance between $P(j \mid t_L)$ and $P(j \mid t_R)$ is maximized across each class value

  - *Q*(s|t) is maximized when proportions of records in child nodes are as different as possible
  - Maximum occurs when for each class value, two child nodes are pure (one contains all the class observations and the other contains none)
  - *For Credit Risk* = "Good" or "Bad", there are two classes (i.e., k = 2). So, the maximum possible value for component *Q*(s|t) is k=2.

Classification and Regression Trees *(cont'd)*

- What is the maximum value for $2P_LP_R$?
  - Component is maximized when proportion of records in P_L and P_R branches are equal
  - Optimality measure favors balanced splits of records between left and right branches
  - Theoretical maximum is 2(0.5)(0.5) = 0.5, which happens at k=5.

Classification and Regression Trees *(cont'd)*

- Additional Data Partitions Using CART
  - Repeat for every decision node that is not a leaf node:
    - Find the training records associated with the decision node.
    - Recompile a table of available candidate splits at the decision node based on its associated training records.

Classification and Regression Trees *(cont'd)*

- Additional Data Partitions Using CART
  - In our example, CART grows full tree by iterating twice more. Creates candidate splits, and splits records at decision node according to optimality measure (Table 8.5 Page 180)

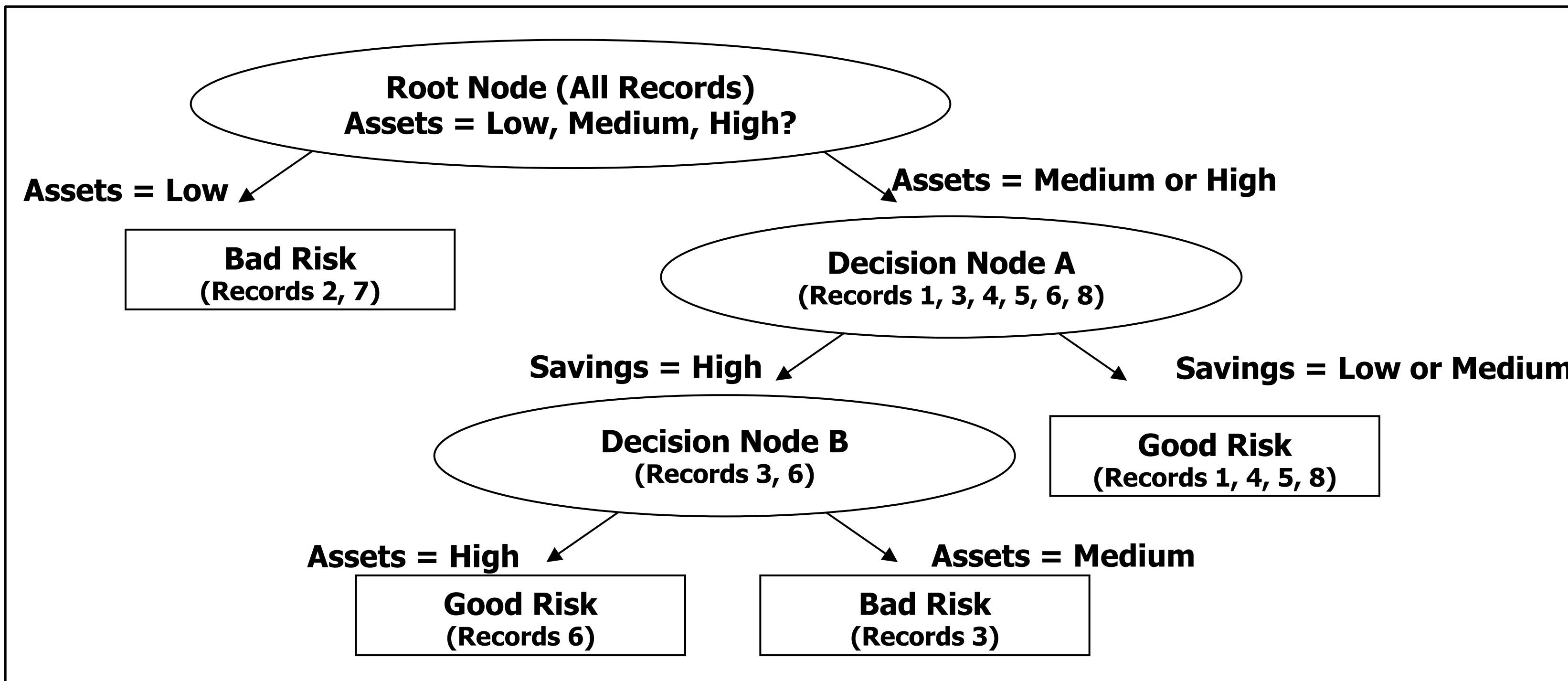


Figure 8.4

Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition, by Daniel Larose and Chantal Larose, John Wiley and Sons, Inc., 2014.

Classification and Regression Trees (*cont'd*)

- Classification Error Rate
  - CART recursively builds tree by following the procedure outlined previously
  - After tree is "fully grown", not all leaf nodes are necessarily homogenous. Some might be diverse leaf nodes
  - Leads to certain level of classification error
  - Consider the records in table below. They cannot be further partitioned, and are classified as "Bad"

Customer	Savings	Assets	Income	Credit Risk
004	High	Low	<=$30K	Good
009	High	Low	<=$30K	Good
027	High	Low	<=$30K	Bad
031	High	Low	<=$30K	Bad
104	High	Low	<=$30K	Bad

Classification and Regression Trees *(cont'd)*

- Probability that a record in the given leaf node is classified correctly (as "Bad") is 3/5 = 0.60 = 60%
- Classification error rate for leaf node is 0.40 = 40%. 2/5 "Good" records classified incorrectly ("Bad")
- CART calculates classification error rate for tree as the weighted average of the individual leaf error rates
- The weight of each leaf equals to the proportion of records in that leaf

Classification and Regression Trees (*cont'd*)

- Pruning
  - As tree grows, each subset of records to partition becomes smaller and less representative
  - Fully grown tree has lowest classification error rate
  - However, the resulting model may be too complex, resulting in overfitting the training set
  - CART avoids overfitting (memorizing) the training set by <u>pruning</u> nodes and branches that reduce generalizability
  - (Self Study) CART algorithm finds adjusted classification error rate that penalizes tree for having too many leaf nodes (too much complexity)
    - Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone, Classification and Regression Trees, Chapman & Hall/CRC Press, Boca Raton, FL, 1984.

C4.5 Algorithm

- C4.5 is extension of ID3 developed by Quinlan in 1992.
- Similar to CART, C4.5 builds a tree by recursively visiting decision nodes and choosing an optimal split, until no further splits are possible

- Key Differences Between CART and C4.5
  - Unlike CART, C4.5 is not limited to binary splits and produces a tree with variable shape (not necessarily a binary tree)
  - C4.5 produces a separate branch for each categorical value. This may result in more "bushiness" than desired, since some values may have low frequency or may naturally be associated with other values.
  - C4.5 uses a different method to measure the homogeneity occurring at leaf nodes

C4.5 Algorithm *(cont'd)*

- C4.5 uses information gain or entropy reduction to select optimal split at each decision node
- In Engineering, information is analogous to signal, and entropy is analogous to noise

- What is Entropy?
  - For an event with probability *p*, the average amount of information in bits required to transmit the result is $-log_2(p)$
  - For example, toss a fair coin with *p* = 0.5. Result of toss transmitted using $-log_2(0.5) = 1$ bit of information. This 1 bit of information represents the result of a toss as 0 or 1 that corresponds to either "HEAD" or "TAIL".
  - Another example, toss a fair dice. Consider the event "5" with probability 1/6. We need $-log_2(1/6) = 2.6$ bit of information. So, we need at least 3 bits to represent the result "5" as 101

C4.5 Algorithm *(cont'd)*

- Consider a variable *X* which can have *k* values with probabilities $p_1, p_2, \ldots, p_k$
- The smallest number of bits, on average per symbol, needed to transmit a stream of symbols representing the values of X observed is called the Entropy of *X* defined as:

$$H(X) = -\sum_j p_j \log_2(p_j)$$

Expected minimum number of bits required to represent X

- In other words, for variables with several outcomes, we use a weighted sum of $-\log_2(p)$'s to transmit the result, where weights are equal to the outcome probabilities.

C4.5 Algorithm *(cont'd)*

- How Does C4.5 use Entropy?
  - Consider a candidate split *S* that partitions the training data set *T* into subsets $T_1, T_2, \ldots, T_k$
  - The <u>Mean information requirement</u> can be calculated as the <u>**weighted**</u> sum of <u>**entropies associated with each subset Ti**</u>

$$H_S(T) = \sum_{i=1}^{k} p_i H(T_i),$$

  - where weights *Pi* represents proportion of records in subset T_i
  - $H(T_i)$ is calculated for each subset considering the target variable (classification label) as a random variable with probability of each class equal to proportion of records with that label in subset T_i

C4.5 Algorithm *(cont'd)*

- Information Gain
  - C4.5's uses Information Gain

    $$gain(S) = H(T) - H_S(T)$$

  - It represents the increase in information produced by partitioning training data *T* according to the candidate split *S*
  - Among all candidate splits at a decision node, C4.5 chooses the split that has the maximum information gain, i.e., gain(S)
- Example
  - C4.5 is illustrated with the example from Table 8.2 where a customer is classified either "Good" or "Bad" credit risk using three predictor fields

C4.5 Algorithm *(cont'd)*

Table 8.2

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	High	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

- As with CART, consider all candidate splits for root node shown below

Table 8.6

Candidate Split	Child Nodes		
1	Savings = Low	Savings = Medium	Savings = High
2	Assets = Low	Assets = Medium	Assets = High
3	Income <= $25,000		Income > $25,000
4	Income <= $50,000		Income > $50,000
5	Income <= $75,000		Income > $75,000

Alternative split for income:
income<=25k, 25k< income <=50k, 50k<income<=75k, income>75k

C4.5 Algorithm *(cont'd)*

- Calculate Entropy of training set <u>before splitting</u>, where 5/8 records classified "Good" and 3/8 "Bad"

$$H(T) = -\sum_j p_j \log_2(p_j) = -\frac{5}{8}\log_2(\frac{5}{8}) - \frac{3}{8}\log_2(\frac{3}{8}) = 0.9544 \text{ bits}$$

- Compare each candidate split against H(T) = 0.9544 to determine which split <u>maximizes the information gain</u>
- Candidate 1

Split on *Savings*, "High" = 2 records, "Medium" = 3 records, and "Low" = 3 records:

$$P_{High} = \frac{2}{8},\ P_{Medium} = \frac{3}{8},\ P_{Low} = \frac{3}{8}$$

C4.5 Algorithm *(cont'd)*

- Of records *Savings* = "High", 1 is "Good" and 1 is "Bad". *P* = 0.5 of choosing "Good" record
- Where *Savings* = "Medium", 3 are "Good", so *P* = 1.0 choosing "Good"
- Of records *Savings* = "Low", 1 is "Good" and 2 are "Bad". This results *P* = 0.33 choosing "Good"
- Entropy of 3 branches, "High", "Medium", and "Low", are:

$$H(High) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$$

Absolute 0 * -infinity =0

$$H(Medium) = -\frac{3}{3}\log_2(\frac{3}{3}) - \frac{0}{3}\log_2(\frac{0}{3}) = 0$$

$$H(Low) = -\frac{1}{3}\log_2(\frac{1}{3}) - \frac{2}{3}\log_2(\frac{2}{3}) = 0.9183$$

C4.5 Algorithm *(cont'd)*

- Combining Entropy for three branches, along with corresponding proportion P_i:

$$H_S(T) = \sum_{i=1}^{k} P_i H_S(T_i) = \frac{2}{8}(1) + \frac{3}{8}(0) + \frac{3}{8}(0.9183) = 0.5944 \text{ bits}$$

- Information Gain represented by split on *Savings* attribute is $H(T) – H_{Savings}(T) = 0.9544 – 0.5944 = 0.36$ bits

- How are these measures interpreted?
  - <u>Before split</u>, H(T) = 0.9544 bits. It takes 0.9544 bits on average to transmit the credit risk associated with the 8 records in the data set
  - After split, 0.36 bits less required to transmit the credit risk associated with the 8 records in the data set

C4.5 Algorithm *(cont'd)*

- Splitting based on *Savings* results in $H_{Savings}(T) = 0.5944$. Now, on average, fewer bits of information required to transmit the credit risk associated with 8 records
- Reduction in Entropy is Information Gain, i.e., 0.9544 – 0.5944 = 0.36 bits of information gained by splitting based on *Savings*
- C4.5 chooses the candidate split with the highest Information Gain as an optimal split at the root node
- Information Gain for other 4 candidate splits calculated similarly

C4.5 Algorithm *(cont'd)*

- Information Gain for 5 candidate splits occurring at root node are summarized below (pages 175-178 of Larose textbook)
- Candidate split 2 has the highest Information Gain = 0.5487 bits, and is chosen for initial split

Table 8.7

Candidate Split	Child Nodes	Information Gain (Entropy Reduction)
1	Savings = Low Savings = Medium Savings = High	0.36 bits
2	Assets = Low Assets = Medium Assets = High	0.5487 bits
3	Income <= \$25,000 Income > \$25,000	0.1588 bits
4	Income <= \$50,000 Income > \$50,000	0.3475 bits
5	Income <= \$75,000 Income > \$75,000	0.0923 bits

C4.5 Algorithm *(cont'd)*

- Initial split results in two terminal leaf nodes and one second-level decision node
- Records with *Assets* = "Low" and *Assets* = "High" have the same target class value, "Bad" and "Good", respectively

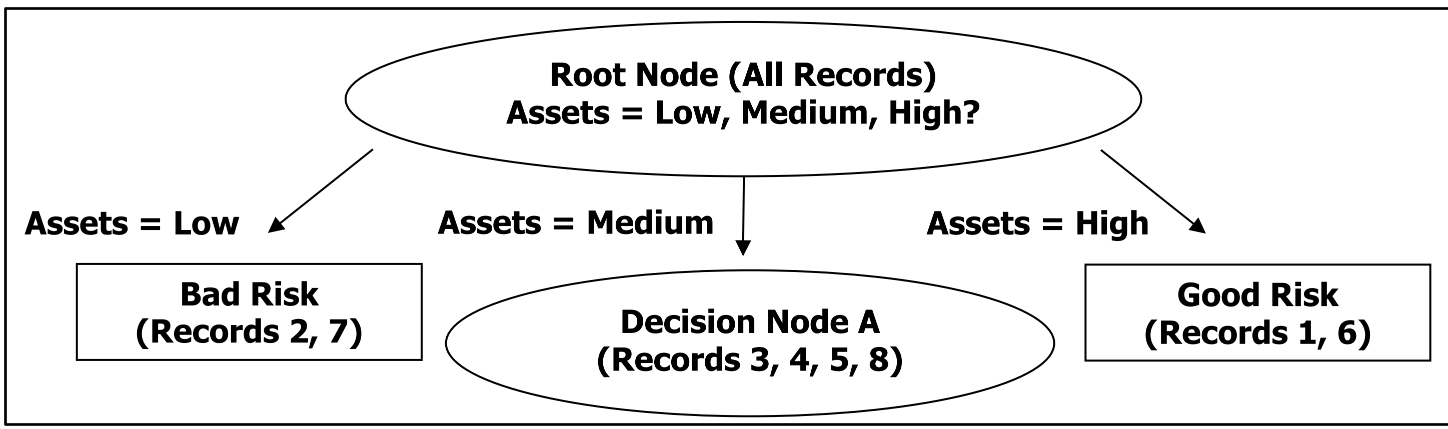


Figure 8.6

- C4.5 iterates at Decision Node A, choosing optimal split from list of four possible candidate splits (pages 178-179 of Larose textbook)

C4.5 Algorithm *(cont'd)*

- Diagram below shows fully-grown C4.5 tree. It is "bushier" and one level shallower, compared to tree build by CART
- Both algorithms concur on importance of *Assets* (root level) and *Savings* (second level)

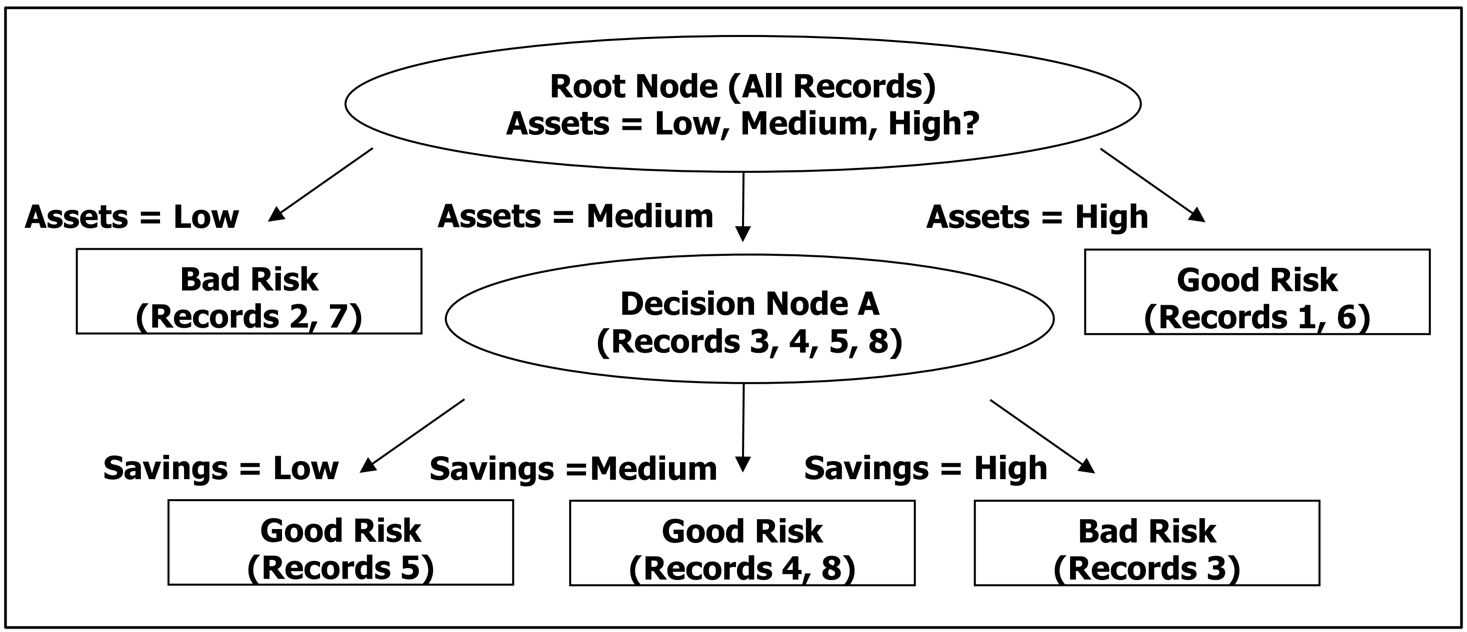


Figure 8.7

- (Self-Study) After fully growing the tree, C4.5 performs <u>pessimistic postpruning</u>, if necessary. Increases the generality of the tree
  - Mehmed Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms, 2nd edn, Wiley-Interscience, Hoboken, NJ, 2011.

Decision Rules

- Decision Trees produce interpretable output in human-readable form
- Decision Rules are constructed directly from a Decision Tree output by traversing the unique path from the root node to a given leaf node
- Decision Rules have the form of IF antecedent THEN consequent
- Antecedent consists of the attributes' values from branches of a given path
- Consequent is the classification of the records contained in a particular leaf node corresponding to a path

Decision Rules *(cont'd)*

- Recall the full decision tree produced by C4.5. Table below shows Decision Rules associated with the tree

Table 8.10

Antecedent	Consequent	Support	Confidence
If assets = low	then bad credit risk.	2/8	1.00
If assets = high	then good credit risk.	2/8	1.00
If assets = medium and savings = low	then good credit risk.	1/8	1.00
If assets = medium and savings = medium	then good credit risk.	2/8	1.00
If assets = medium and savings = high	then bad credit risk.	1/8	1.00

- <u>Support</u> of decision rule shows proportion of records in training set resting in leaf node
- <u>Confidence</u> is percentage of records in leaf node, for which decision rule is true
- Although confidence levels reported for this example are 100%, but this is not typical in real-world examples