Decision Trees

Khasha Dehnad

Choosing good apples

Apple	Color	Weight	Price	Outcome
1	Yellow	High	0.75	Good
2	Red	Low	0.5	Bad
3	Green	Medium	0.25	Bad
4	Yellow	Medium	0.5	Good
5	Red	Medium	0.75	Bad
6	Green	High	0.25	Bad
7	Red	Low	0.25	Bad
8	Yellow	Medium	0.75	Good
9	Yellow	High	0.75	Good
10	Red	Low	0.5	Bad

Goodness of a candidate split

$$Q(s/t) = 2P_L * P_R \sum | p(j/t_L) - P(j/t_R) |$$

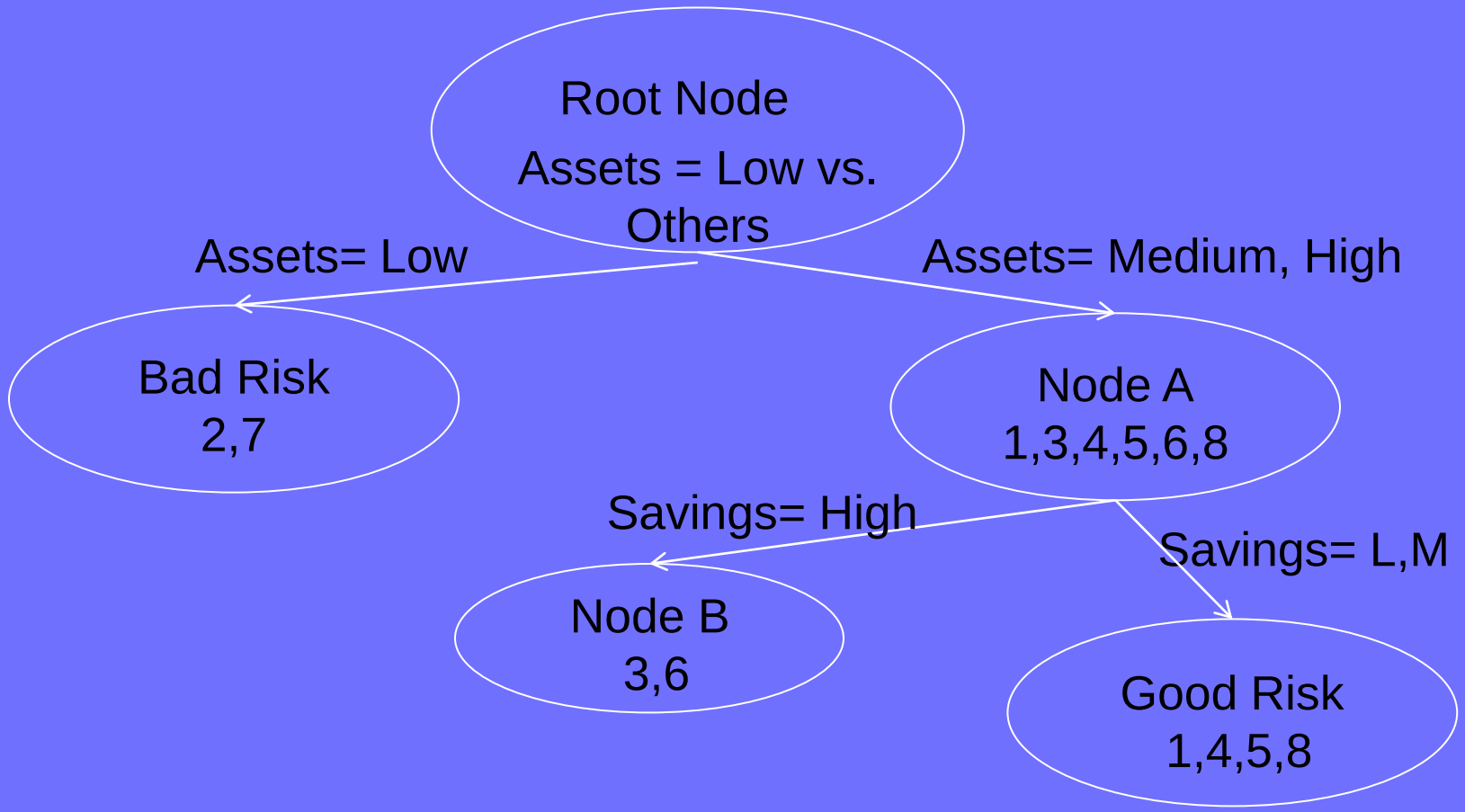
Example From the Book

Customer	Savings	Assets	Income	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

Example From the Book

Split			p(j/tl)		p(j/tr)				
	PL	PR	Good	Bad	Good	Bad	2Pl * PR	Q(s/t)	Over all
Savings =Low	3/8	5/8	1/3	2/3	4/5	1/5	2*3/8*5/8	|1/3-4/5|+|2/3-1/5|	**2Pl * PR * Q(s/t)**
	0.375	0.625	0.333	0.667	0.800	0.200	0.469	0.933	0.438
Savings =Med	0.375	0.625	1.000	0.000	0.400	0.600	0.469	1.200	0.563
Savings=High	0.250	0.750	0.500	0.500	0.667	0.333	0.375	0.333	0.125

Goodness of a candidate split



Probability, Information Theory and Entropy
Measuring Disorder (Impurity)

Entropy:

Entropy generalizes the information (disorder) of particular categories to the information (disorder) of a cluster with a number of possible categories.

If pi is the probability of the occurrence of the ith category in a particular cluster, then the Entropy of that cluster is defined as:

$$H = \sum_i p_i \log_2(1/p_i) = -\sum_i p_i \log_2(p_i) bits$$

***H* is the measure of disorder or the information content of a cluster.**

The higher the entropy (disorder) of a cluster, the more information you need to describe (code) the cluster, the more information you need to describe a cluster, the less certain you can be of what the cluster contains.

As the probability of a single category in a cluster gets closer and closer to 1, the entropy of the cluster (*H)* gets closer and closer to 0, the disorder of the cluster gets closer and closer to 0, and the cluster becomes most valuable in the sense that it contains a single category.

Deriving Rules From Data

Machine Learning Algorithms (ML):

- **derive rules from the data,**
- **create rules and rule-trees by searching through the data for statistical patterns and relationships,**
- **uncover useful relationships among variables, relationships that you may not have known even existed**
- **use the information in the data and cluster recorded items into specific classes/categories,**
- **build prediction and classification models and explicit rules from the data,**
- **derived models and rules help explaining the process that generated the data.**

- **Database Queries** **answer the question: "What are the data that match this pattern?"**
- **ML Algorithms** **answer the question: "What is the pattern that matches these data?"**
- **Neural Nets** **learn from the data and drive implicit relationships between variables,**
- **Rule Based Systems** **encode knowledge in explicit rules - but rules are elicited from experts,**
- **ML Algorithms** **learn from the data and drive explicit rule-like relationship between variables,**

C4.5 Algorithm

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

- What is Entropy?
  - Event with probability = p, average amount of information, in bits, required to transmit result $-\log_2(p)$
  - For example, toss fair coin with $p = 0.5$
  - Result of toss transmitted using $-\log_2(0.5) = 1$ bit of information
  - 1 bit of information represents result toss as 0 or 1
  - Corresponds to either "HEAD" or "TAIL"

C4.5 Algorithm (*cont'd*)

- Variable *X* has *k* values with probabilities p_1, p_2, ... , p_k
- For variables with several outcomes, use weighted sum of $-\log_2(p)$'s to transmit result
- Entropy of *X* defined using formula:

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

- Represents smallest number of bits, on average per symbol, needed to transmit stream of symbols corresponding to observed values of *X*

C4.5 Algorithm (*cont'd*)

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

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

  - Represents increase in information by partitioning training data *T* according to candidate split *S*
  - For each candidate split, C4.5 chooses split that has maximum information gain, gain(S)
- Example
  - C4.5 illustrated with example
  - Recall the data set where customer classified "Good" or "Bad" credit risk using three predictor fields

C4.5 Algorithm (*cont'd*)

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**

- Consider all candidate splits for root node shown below

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*	

C4.5 Algorithm (*cont'd*)

		- (Pj* log(Pj)		- (Pj* log(Pj)	Row Total	Percent	Pct * Row total
Savings	Good	Good	Bad	Bad			
Low	0.333	0.528	0.667	0.390	0.918	0.375	0.344
Medium	1.000	0.000	0.000	0.000	0.000	0.375	0.000
High	0.500	0.500	0.500	0.500	1.000	0.250	0.250
Total						1.000	0.594
Net Gain			0.954 - .594 =	0.360			

		- (Pj* log(Pj)		- (Pj* log(Pj)	Row Total	Percent	Pct * Row total
Assets	Good	Good	Bad	Bad			
Low	0.000	#NUM!	1.000	0.000	#NUM!	0.250	0.000
Medium	0.750	0.311	0.250	0.500	0.811	0.500	0.406
High	1.000	0.000	0.000	#NUM!	#NUM!	0.250	0.000
Total						1.000	0.406
Net Gain			0.954 - .406 =	0.549			

C4.5 Algorithm (*cont'd*)

- Calculate Entropy of training set before splitting, 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 maximizes information gain
- Candidate 1
- Split on *Savings*, "High" = 2 records, "Medium" = 3 records, and "Low" = 3 records:

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

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:

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

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

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

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 is measure interpreted?
- Before split, H(T) = 0.9544 bits, on average. Takes 0.9544 bits of information to transmit credit risk associated with 8 records

C4.5 Algorithm (*cont'd*)

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

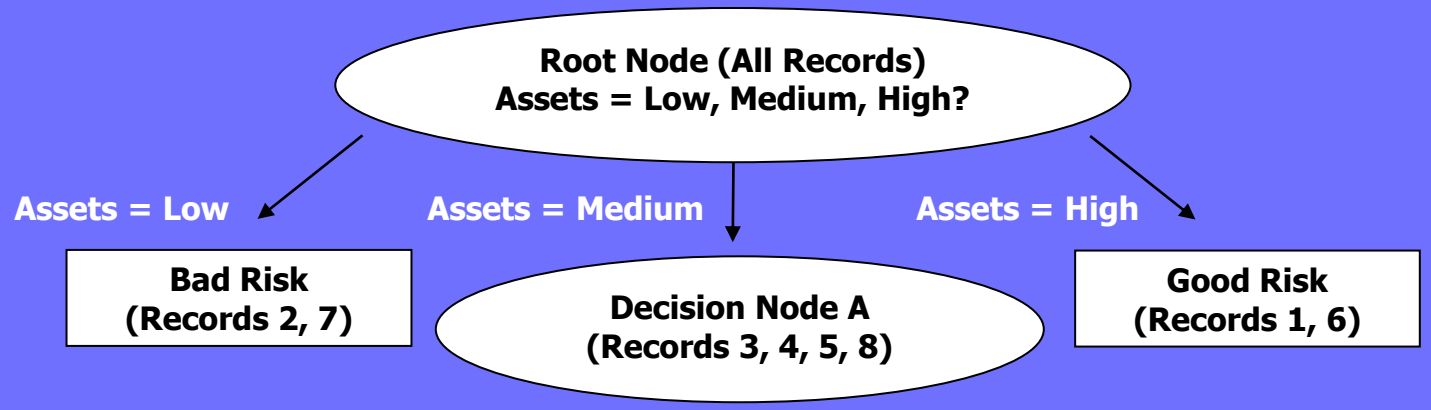
C4.5 Algorithm (*cont'd*)

- Information Gain for 5 candidate splits occurring at root node summarized below
- Candidate split 2 has highest Information Gain = 0.5487 bits, and chosen for initial split

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

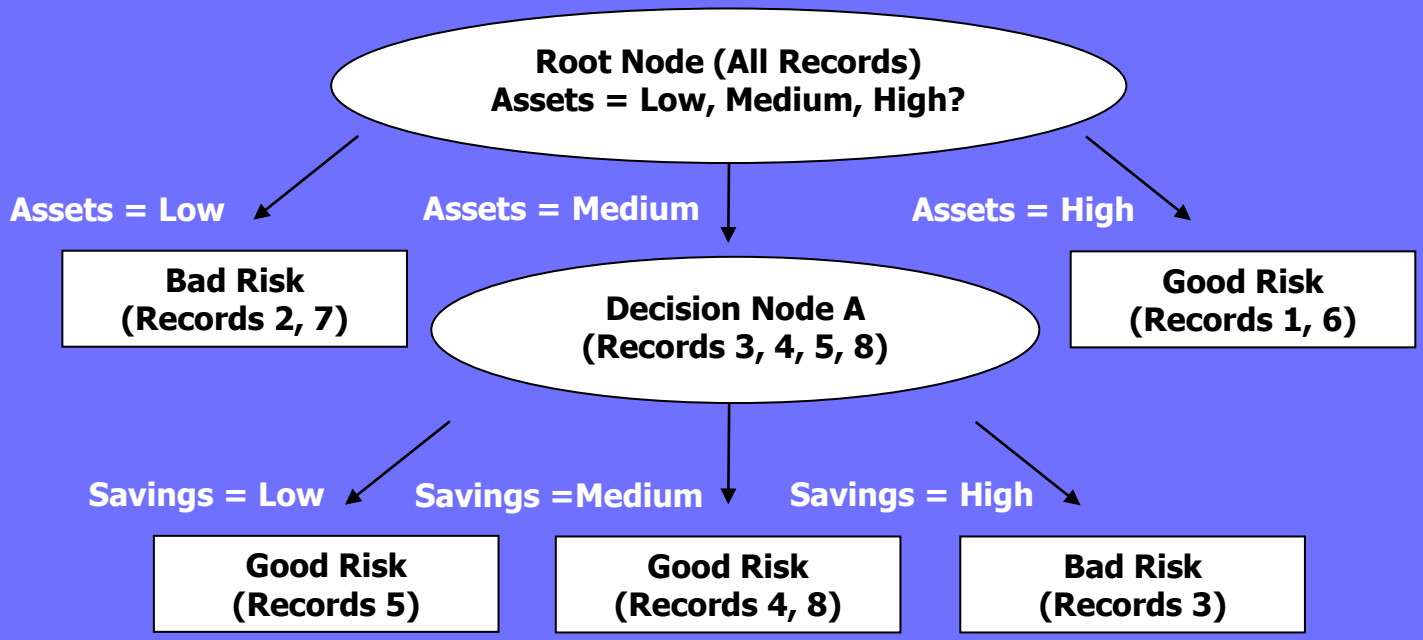
C4.5 Algorithm (*cont'd*)

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



- C4.5 iterates at Decision Node A, choosing optimal split from list of four possible candidate splits

C4.5 Algorithm (*cont'd*)



- After growing tree, C4.5 performs pessimistic postpruning, if necessary. Increases generality of tree
- Diagram shows fully-grown C4.5 tree. "bushier" and one level shallower, compared to tree build by CART
- Both algorithms concur on importance of *Assets* (root level) and *Savings* (second level)

Decision Rules

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

Decision Rules (*cont'd*)

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

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

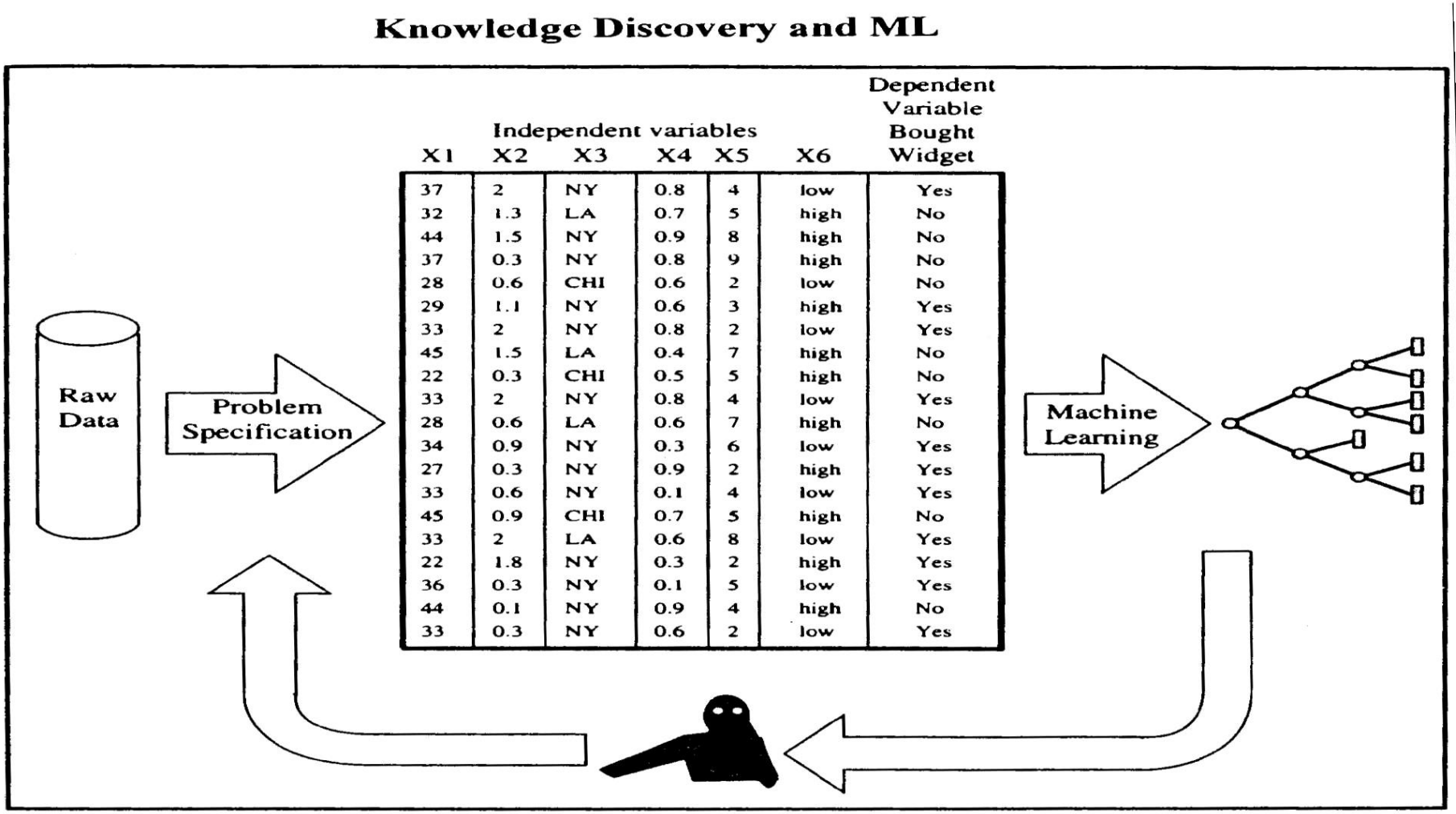
- Support of decision rule shows proportion of records in training set resting in leaf node
- Confidence is percentage of records in leaf node, for which decision rule is true
- Although confidence levels reported = 100%, results not typical of real-world examples

Deriving Rules From Data

Machine Learning Algorithms (ML):

- **derive rules from the data,**
- **create rules and rule-trees by searching through the data for statistical patterns and relationships,**
- **uncover useful relationships among variables, relationships that you may not have known even existed**
- **use the information in the data and cluster recorded items into specific classes/categories,**
- **build prediction and classification models and explicit rules from the data,**
- **derived models and rules help explaining the process that generated the data.**

- **Database Queries** **answer the question: "What are the data that match this pattern?"**
- **ML Algorithms** **answer the question: "What is the pattern that matches these data?"**
- **Neural Nets** **learn from the data and drive implicit relationships between variables,**
- **Rule Based Systems** **encode knowledge in explicit rules - but rules are elicited from experts,**
- **ML Algorithms** **learn from the data and drive explicit rule-like relationship between variables,**



X1	X2	X3	X4	X5	X6	Widget
37	2	NY	0.8	4	low	Yes
32	1.3	LA	0.7	5	high	No
44	1.5	NY	0.9	8	high	No
37	0.3	NY	0.8	9	high	No
28	0.6	CHI	0.6	2	low	No
29	1.1	NY	0.6	3	high	Yes
33	2	NY	0.8	2	low	Yes
45	1.5	LA	0.4	7	high	No
22	0.3	CHI	0.5	5	high	No
33	2	NY	0.8	4	low	Yes
28	0.6	LA	0.6	7	high	No
34	0.9	NY	0.3	6	low	Yes
27	0.3	NY	0.9	2	high	Yes
33	0.6	NY	0.1	4	low	Yes
45	0.9	CHI	0.7	5	high	No
33	2	LA	0.6	8	low	Yes
22	1.8	NY	0.3	2	high	Yes
36	0.3	NY	0.1	5	low	Yes
44	0.1	NY	0.9	4	high	No
33	0.3	NY	0.6	2	low	Yes

FIGURE 10.1 The tree toward the right provides a c

Deriving Rules From Data
Knowledge Discovery and ML

Steps:

- **specify the problem, a profound question,**
- **specify relevant variables - independent and dependent variables,**
- **access, prepare, and clean the data,**
- **apply machine learning algorithms to derive rules, patterns, relationships, models, decision trees.**

ML Algorithm:

- **takes the selected data and finds the relationship between dependent and independent variables,**
- **the relationship can be described as a decision tree, a rule, a chart, or an equation,**
- **the challenge is to focus the statistical analysis so that hidden relationships bubble up to the surface.**

ML Techniques:

- **Genetic Algorithms,**
- **Bayesian Probabilities,**
- **Recursive Partitioning Algorithms.**
- **Neural Networks,**

Deriving Rules From Data
Recursive Partitioning Algorithms:

– split the original data into finer and finer subsets (recursively) resulting in a decision tree,
– a decision tree is a compact explanation of the data,
– the tree will provide the profile of a "typical case" leading to a specific value of the dependent variable,
– example: most "widget" buyers live in New York and are less than 35 years old.

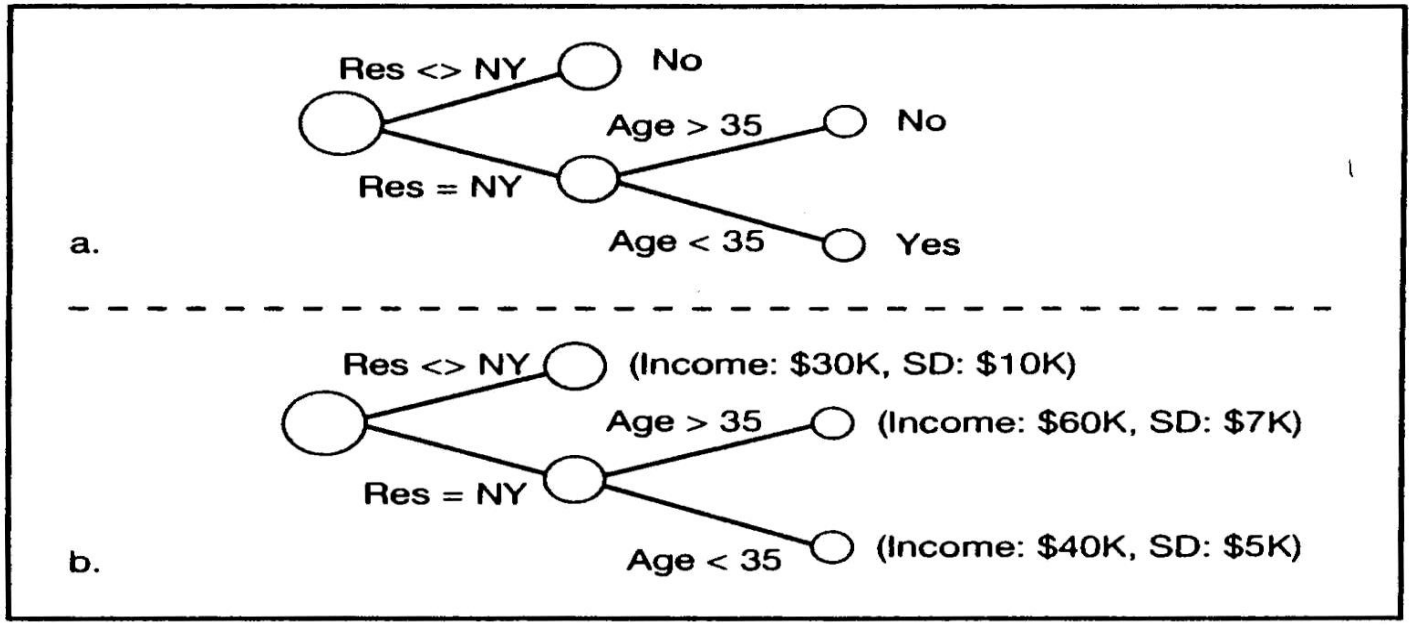


FIGURE 10.2a A Classification Tree **b** A Regression Tree

Deriving Rules From Data
Decision Trees:

Classification Trees:

- **classification trees are used when dependent variable is a categorical/qualitative "YES" or "NO" type of variable,**
- **the goal is to categorize (classify) cases (observations) and derive rules,**
- **cases are fed into the decision tree ,**
- **each case is classified at each node of the tree into branches (categories),**
- **the size of nodes (circles) decreases towards the right indicating that number of cases that fall into each successive cluster is decreasing,**
- **the end result is classification rules such as " IF Age is less than 35 and Residence = New York, THEN *widget buyer* ",**
- **this tree is called a classification tree.**

Deriving Rules From Data
Decision Trees:

Regression Trees:

- **regression trees are used for both categorical and continuous dependent variable,**
- **the goal is to predict or estimate the value of dependent variable(s),**
- **each branch of the tree partitions off a subset of the data,**
- **the last node of the tree shows the value of the dependent variable (income) for that subset of the data,**
- **the end result is a prediction/estimation rule such as: " IF Residence = New York and Age is less than 35 then average Income is $40K with a standard deviation of $5K",**
- **this tree is called a regression tree.**

Deriving Rules From Data
Recursive Partitioning Algorithms:

The two best known RP algorithms are: C4.5 and CART

- **both algorithms try to create homogeneous clusters,**
- **the goal is to make clusters at the nodes "purer" and "purer" by progressively reducing the "disorder" in the original data,**
- **C4.5 developed by computer scientists (Ross Quinlan, C4.5 Programs for ML) reduces the information disorder using *entropy,***
- **CART developed by statisticians (Sanquist and Morgan,1964, Leo Breiman etl, 1984) reduces the statistical disorder using *variance*.**

C4.5 Algorithm: Example

Problem: Widgitco Inc. wants to predict types of customers who are more likely to buy Widget. In other words, the firm wants to find the pattern of customers who buy the product.

Deriving Rules From Data
C4.5 Algorithm: Example (continued)

Solution:

- **divide up the data along the independent variables so that cases form clusters where each cluster contains predominantly "YES" or "NO",**
- **build the decision tree by first splitting the data with respect to the independent variable X3 (residence) and then X1 (age),**
- **this strategic splitting leads to a quick and simple clustering of the data useful for marketing decisions:**

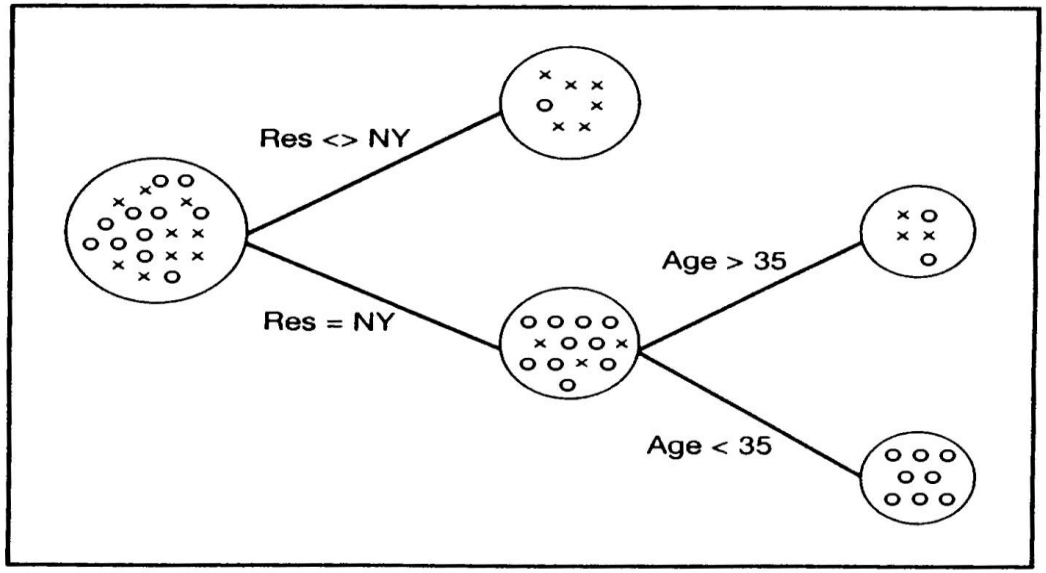


FIGURE 10.3 ML algorithm attempts to split up the data (examples) into more homogeneous subsets (cross = yes, circle = no).

Deriving Rules From Data
C4.5 Algorithm:

How does the algorithm determine which independent variable to split first?
How does the algorithm determine the boundaries for the splits?
How does the algorithm arrive at the most useful discrimination of the data?

The C4.5 algorithm uses Entropy to calculate the impurity of a cluster.

The higher the entropy (disorder) of a cluster, the more information you need to describe the cluster, the more information you need to describe a cluster, the less certain you can be of what the cluster contains.

As the probability of a single category in a cluster gets closer and closer to 1, the entropy of the cluster (H) gets closer and closer to 0, the disorder of the cluster gets closer and closer to 0, and the cluster becomes most valuable in the sense that it contains a single category.

Deriving Rules From Data
C4.5 Algorithm: Example

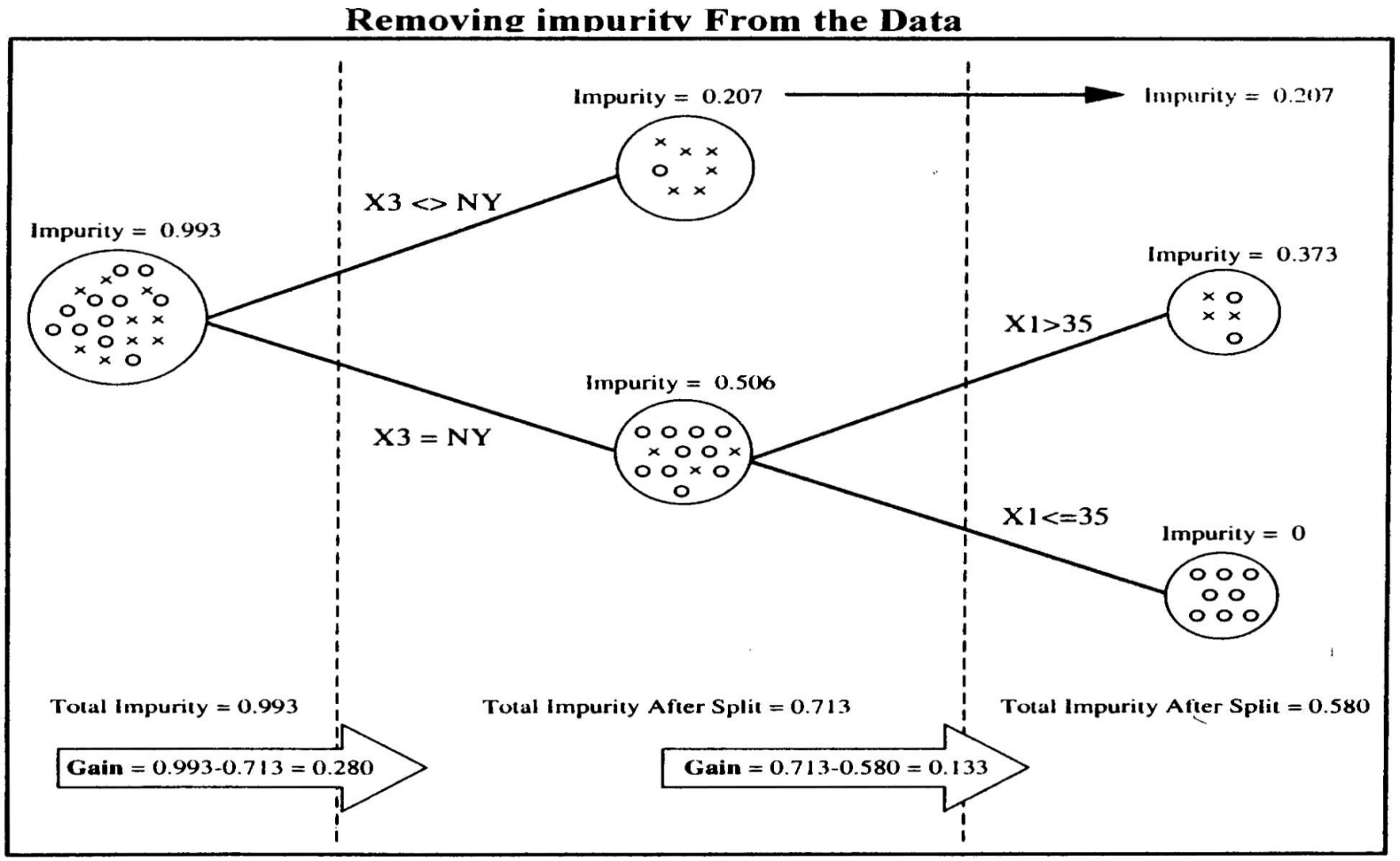


FIGURE 10.5 How "Impurity" is Removed from the Data. The objective is to partition some independent variable that maximizes information gain.

Deriving Rules From Data
C4.5 Algorithm: Example

- **the first split is on variable X3 (residence),**
- **it creates two much purer clusters: a cluster with 6 out of 7 buyers, and a cluster with 10 out of 13 non-buyers,**
- **the first split has reduced the impurity from 0.993 to 0.713 a gain of 0.280 bits,**
- **the second split is on X1 (age),**
- **the second split further reduces the impurity from 0.713 to 0.580 a gain of 0.133,**

Patterns:

Non-New Yorkers have a pattern of not buying widget,

New Yorkers over the age of 35 have a pattern of not buying widgets,

Younger New Yorkers have a pattern of buying widgets.

Deriving Rules From Data
C4.5 Algorithm

Splitting:

- **test all possible independent variables at all their possible split points,**
- **compute resulting gains in purity for each variable and each of its possible splitting points,**
- **pick the split (the variable and its splitting point) that maximizes the gain and form the resulting cluster,**
- **do this over and over again on each resulting cluster (recursively) until the decision tree is built.**

Example:

- **for the widget sales data (Page 3), the independent variable X3 (residence) at the splitting point NY resulted in the highest gain in purity and clusters " X3 = NY" and "X3 = NY" were formed,**
- **once the data were split at X3, the new cluster formed by all the NY records could be further purified,**
- **the algorithm tested all variables again, but only used the data in the new cluster "X3 = NY",**
- **the variable X1 (age) at the splitting point X1 <= 35 resulted in the highest gain in purity and became the most discriminator variable at its value of 35.**

Deriving Rules From Data
C4.5 Algorithm:

Overfitting :

- **how much splitting is appropriate?**
- **figure below shows two extremes: "One split" (underfit) VS "Seven splits" (overfit):**

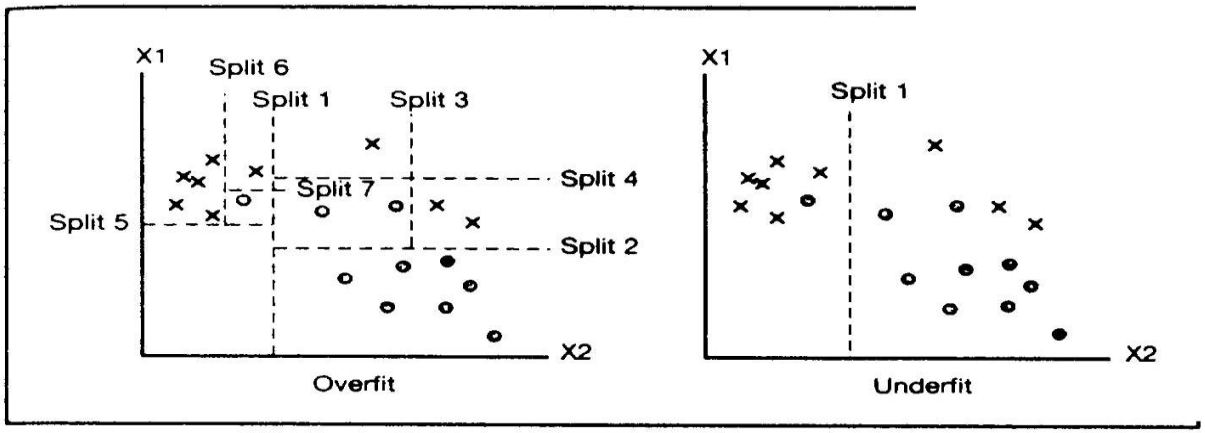
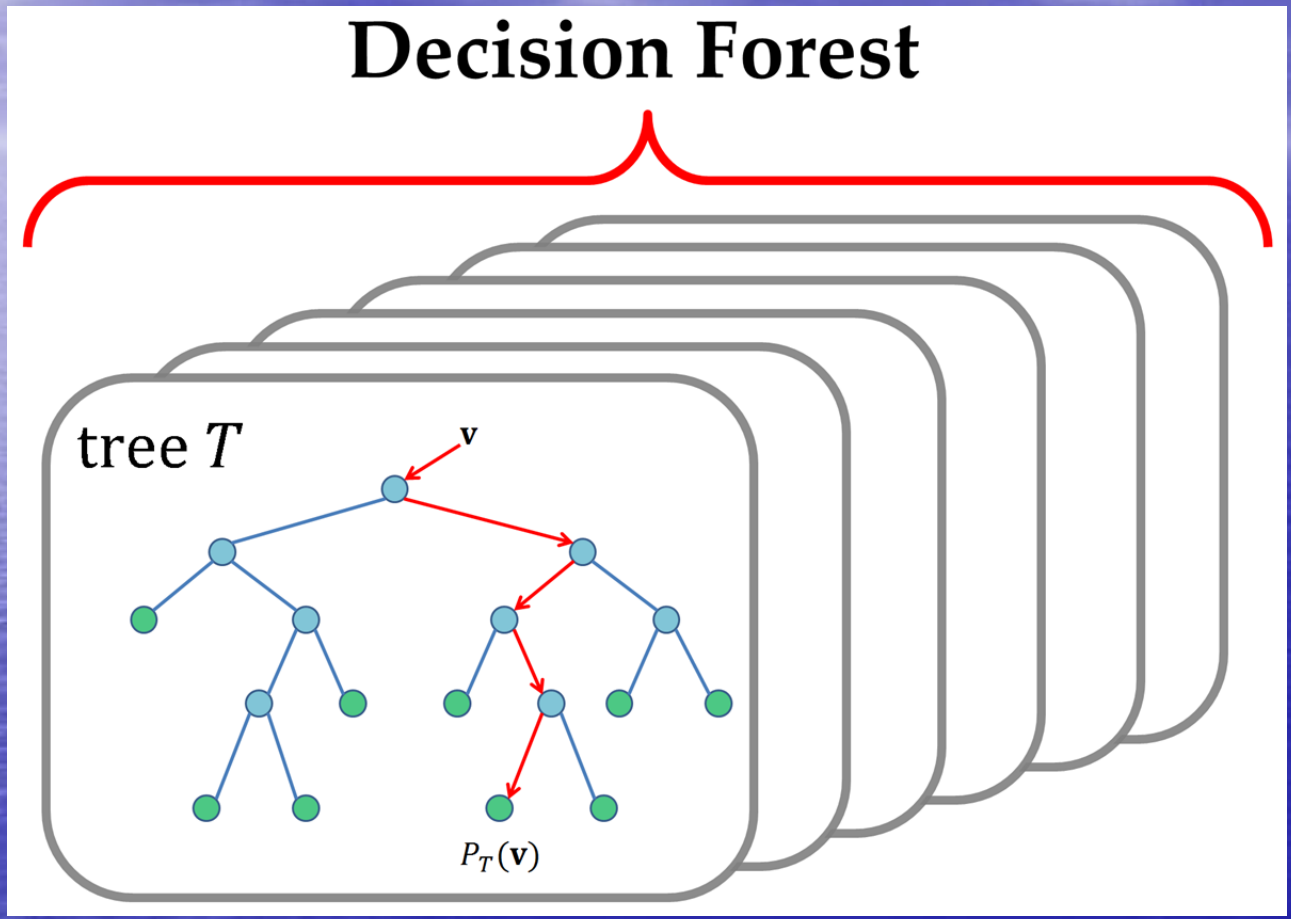
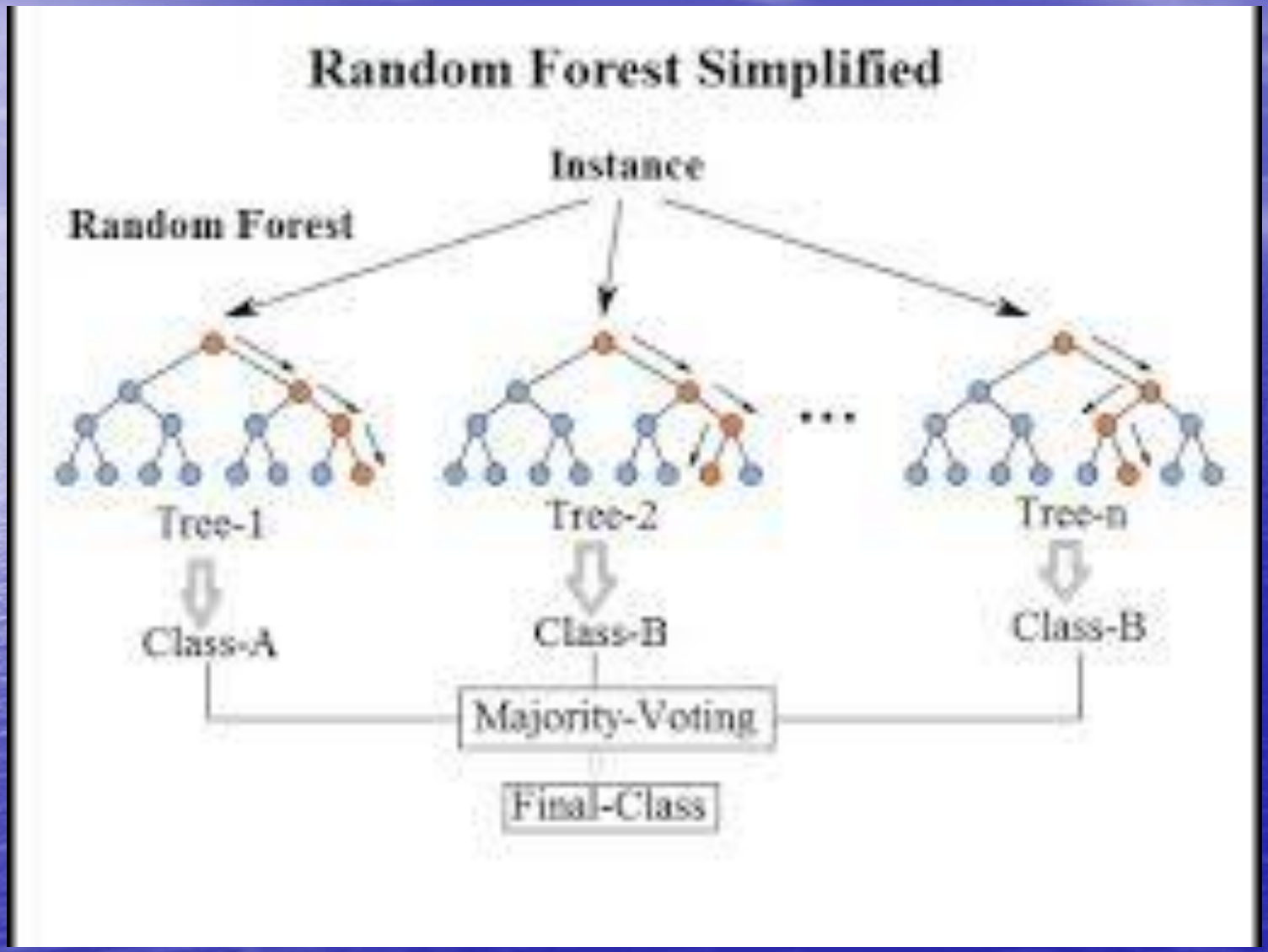


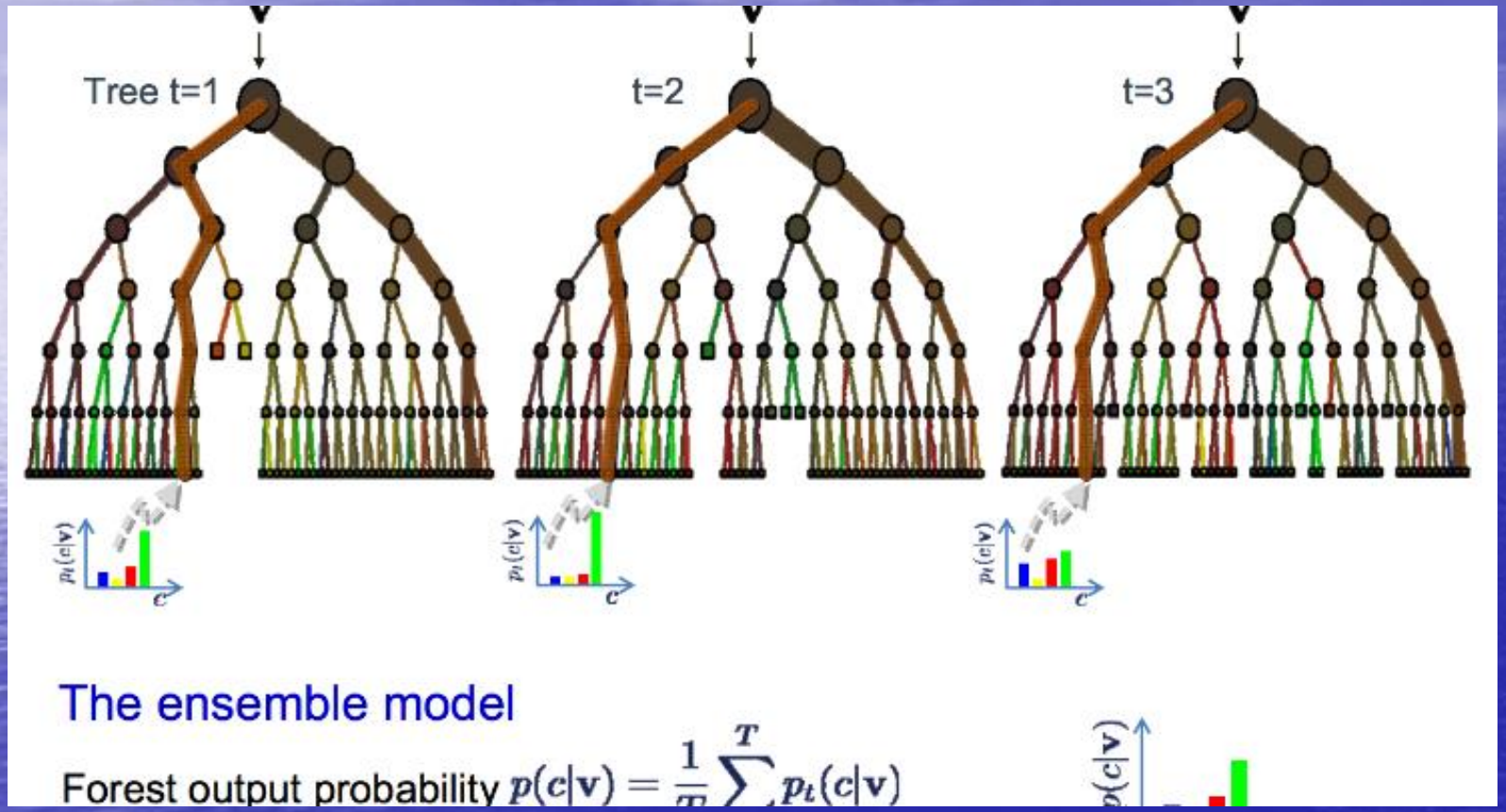
FIGURE 10.7 The extreme case of overfitting is one where each data point has its own terminal node. Underfitting means that the algorithm has not become meaningfully specific.

- **"One split" results two sub-samples (clusters) that are still quite impure (underfit),**
- **"Seven splits" results completely purified clusters, but the model has become very complicated,**
- **very complicated models are hard to understand and have the danger of overfitting the data.**

Decision Forest




Random Forest Simplified
Instance
Random Forest
Tree-1
Tree-2
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class



The ensemble model

Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t} p_t(c|\mathbf{v})$

Gini index

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Deriving Rules From Data
C4.5 Algorithm:

Overfitting:

- overfitting means that the algorithm finds model that fits the "training data" (or the "sample") and performs well on the trained data, but performs poorly on real world new data it has not seen before (out of "sample" data),
- the ML algorithm may pick up details in the data that are characteristics of the training sample, but not the actual problem being modeled,
- the neural nets and ML algorithms easily "overtrain": perform well on the training data but badly on real world data they have not seen before.

Cross - Validation:

- to avoid ovefitting one approach adopted by statisticians is "cross-validation",
- in cross-validation the training data is divided into several parts, say 10,
- all parts but one is used to train the model, and use the leftover part to test the model,
- this is done several times each time using different training parts and leftover testing part,
- in effect splits are tested on several different combinations of training and testing data,
- for instance, with 10 parts there is a total 10 combinations of training and testing (9 parts for training and 1 part for testing) .

Deriving Rules From Data
C4.5 Algorithm:

Tree Size - Optimal number of splits:

- the accuracy of the model on "out of sample" data depends on the tree size (number of splits),
- the error rate of the model initially decreases as the size of the the tree increases, but after some optimal point the error rate stars to increase (overfit),
- the accuracy is low for both very simple and very complex trees

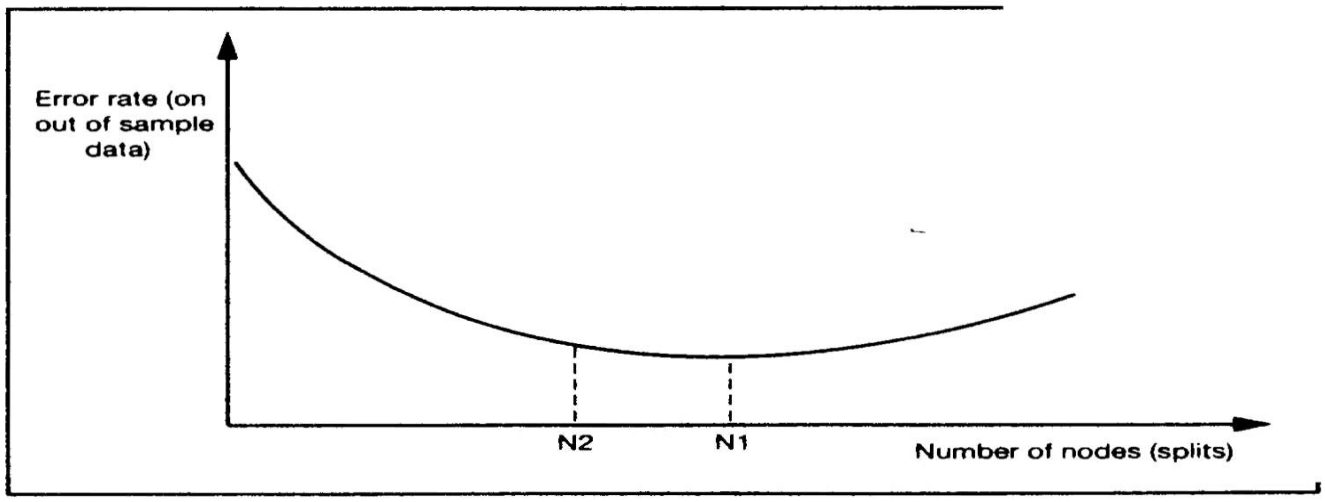


FIGURE 10.8 The Relationship between Tree Complexity and Error Rate

Deriving Rules From Data
C4.5 Algorithm:

Pruning: **there are two approaches to get to "right sized" trees: "Stopping" and "Pruning"**

- **in the stopping approach the ML algorithm must decide when to stop while going forward constructing the tree - it turns out that this approach is very hard,**
- **in the pruning approach the ML algorithm tests all possible splits and grows the complete (overfitted) tree, and then "prunes back" the useless branches - the ones that increase the error rate on "out of sample" data,**
- **the tree is iteratively pruned, and with each prune, the error of the pruned tree is computed, util the best overall tree is constructed,**
- **research shows that pruning approach results in much more robust trees.**

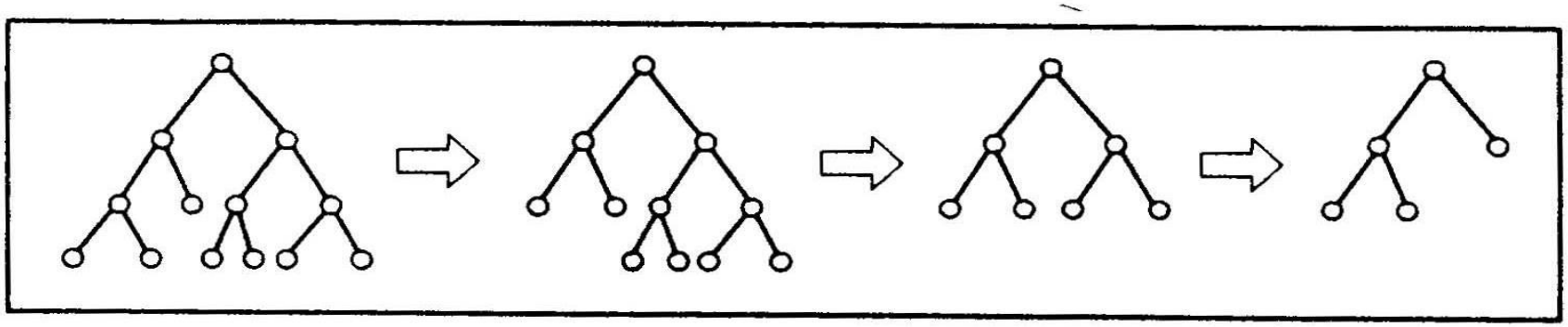


FIGURE 10.9 A large tree is first grown that overfits the data and then pruned back to a "right-sized" tree.

Deriving Rules From Data
Classification And Regression Trees (CART)

- if the dependent variable is continuous, CART algorithm is used to construct decision trees,
- for continuous variables there is no notion of "pure" partition - there are infinite values for a dependent variable, classification and clustering is done through the notion of "dispersion",
- disorder is measured by variance (a measure of "dispersion" around the " center "),
- homogeneous clusters are formed from items that their dependent variable values are all close to each other - i.e., their standard deviation would be quite small,
- the criteria is to form clusters with minimum " within cluster " variance and maximum "between clusters" means - impurity and gain are measured by the pair (mean, SD)

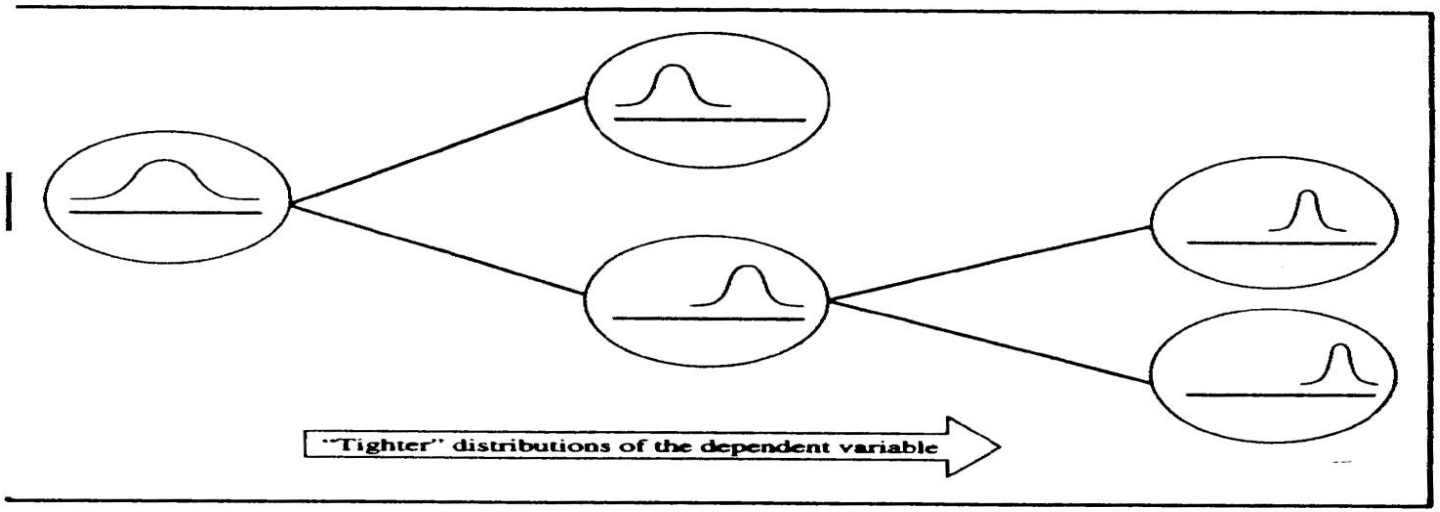


FIGURE 10.10 How Partitioning Works with a Continuous-valued Dependent Variable. When the dependent variable is continuous, the algorithm attempts to split up the data (examples) into more homogeneous subsets that are far apart from each other.

Deriving Rules From Data
CART: Example

Problem: a software company that sells packaged PC software wants to derive decision rules that determine type of customers generating highest revenue,

Data: the company has a database containing records of approximately 2000 customers - each record consists of values of 31 variables,

Dependent Variable(s): "total annual sales" made to the customer (a continuous variable),

Independent Variables: there were 30 independent variables recorded for each customer.

Solution:

_ since the dependent variable (sales) is continuous a CART Algorithm has been used,

– clusters were formed by computing mean and standard deviation of customers total annual sales,

– homogenous clusters were identified based on minimum 'within cluster SD" and maximum between clusters means,

– the partitioning algorithm found only three of the 30 independent variables to be relevant for classifying customers:

  – the number of users in the customer's organization,

  – the number of inquiries they made about the product they purchased,

  – and the number of product licenses they purchased,

– figure below shows the tree that was constructed using actual data and the CART Algorithm.

Deriving Rules From Data
CART

– **it is hard to interpret the tree,**
– **the top branch has not been expanded.**

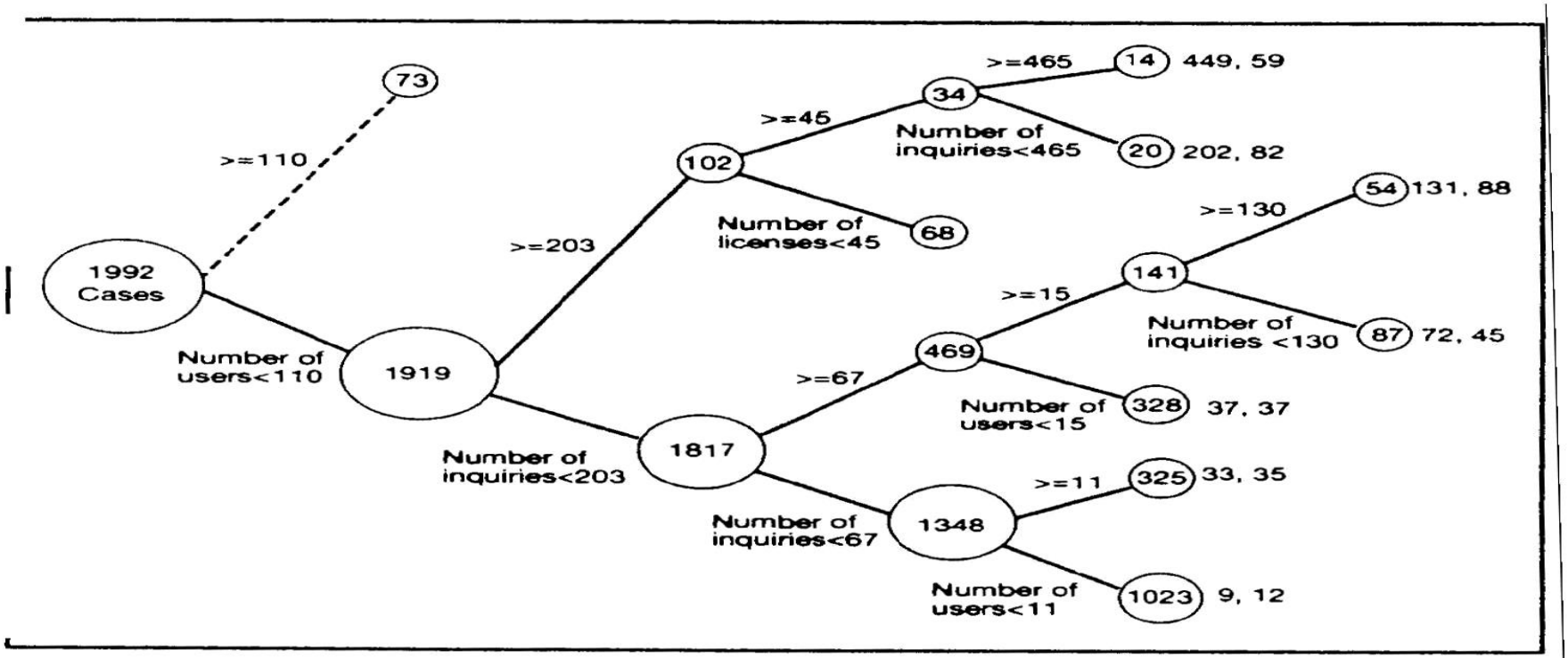


FIGURE 10.11 Part of an "Optimal Tree." Numbers in circles denote number of cases; at leaf nodes, pair is mean and SD.

Deriving Rules From Data
CART:

The simplified tree is easier to interpret, it splits the data into four types that can be classified into a 2 by 2 matrix:

- **large customers who make large number of inquiries,**
- **large customers who make a few number of inquiries,**
- **small customers who make large number of inquiries,**
- **small customers who make small number of inquiries.**

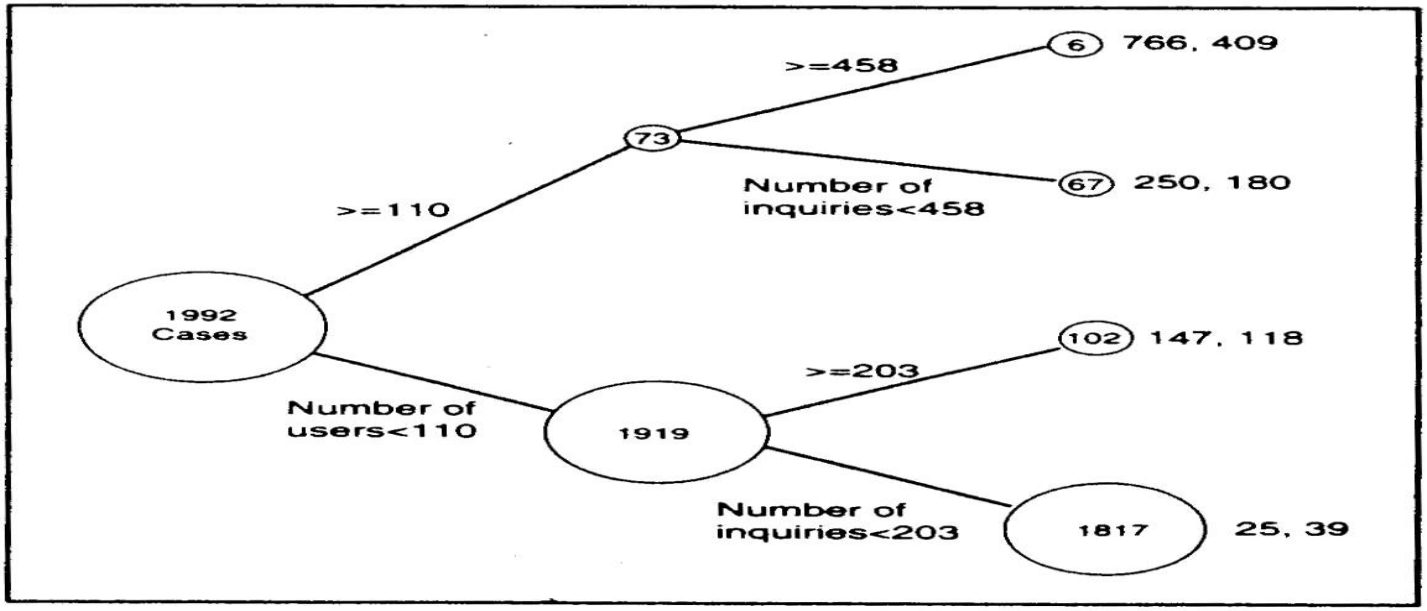


FIGURE 10.12 A Simplified Decision Tree

Machine Learning Algorithms (ML)
Summary and ID Dimensions

Dimension	ML Algorithm	But……..
Accuracy	Moderate to high	Depends on parameter settings to adjust for trade off between tree size and accuracy; over fitting must be controlled
Explainability	Moderate to high	Depends on parameter settings to adjust for trade off between tree size and accuracy;
Response Speed	High	----------
Scalability	Moderate to High	Time to build tree increases as data set get large
Compactness	Moderate	----------
Flexibility	High	Depends on availability of Data representing changes in the process
Embeddability	Moderate to High	----------
Tolerance to Complexity	Moderate	Some Complexity can be captured but only through increasingly complicated trees.
Tolerance for noise in data	Moderate	----------
Tolerance for Sparse data	Low	----------
Development Speed	Moderate	Depends on understanding of process and computer hardware
Tolerance for independence from experts	Moderate	User may wish to have expert audit developed rules
Computing resources	Moderate	Scale with respect to amount of data

Machine Learning Algorithms (ML)
Summary and ID Dimensions

Dimension	ML Algorithm	But……..
Accuracy	Moderate to high	Depends on parameter settings to adjust for trade off between tree size and accuracy; over fitting must be controlled
Explainability	Moderate to high	Depends on parameter settings to adjust for trade off between tree size and accuracy;
Response Speed	High	----------
Scalability	Moderate to High	Time to build tree increases as data set get large
Compactness	Moderate	----------
Flexibility	High	Depends on availability of Data representing changes in the process
Embeddability	Moderate to High	----------
Tolerance to Complexity	Moderate	Some Complexity can be captured but only through increasingly complicated trees.
Tolerance for noise in data	Moderate	----------
Tolerance for Sparse data	Low	----------
Development Speed	Moderate	Depends on understanding of process and computer hardware
Tolerance for independence from experts	Moderate	User may wish to have expert audit developed rules
Computing resources	Moderate	Scale with respect to amount of data