Data Mining with Decision Trees: An Introduction to CART®

Dan Steinberg

8880 Rio San Diego Dr., Suite 1045

San Diego, CA 92108

619.543.8880 - 619.543.8888 fax

http://www.salford-systems.com

dstein@.salford-systems.com

Goals of Tutorial

- Provide Overview of Methodology
  - Historical background
  - Motivation for research
- Explain Key Concepts
  - Tree growing & splitting
  - Testing & pruning
  - Selection of “optimal” tree
  - Refinement of analysis
- Guide to Reading the Output
  - How to interpret statistics and reports
  - What to ask for in the way of diagnostic reports
- Introduce Software
  - New look and feel of CART 3.6®
  - Hints, tricks, traps
  - Troubleshooting
- New Developments
  - “Committee of Experts” -- bagging and boosting (ARCing)
  - Probabilistic splits
  - Split revisitation
  - Survival analysis

In 1984 Berkeley and Stanford statisticians announced a remarkable new classification tool

- **A computer intensive technique that could automatically analyze data**
- **Method could sift through any number of variables**
  - **Could separate relevant from irrelevant predictors**
  - **Did not require any kind of variable transforms (logs, square roots)**
  - **Impervious to outliers and missing values**
  - **Could yield relatively simple and easy to comprehend models**
  - **Required little to no supervision by the analyst**
  - **Was frequently more accurate than traditional logistic regression or discriminant analysis, and other parametric tools**

Is this too good to be true?

- **Why are so many of us hearing about CART for the first time 15 years after its introduction?**
- **Haven't all Artificial Intelligence approaches to data analysis failed to live up to expectations?**
- **Does it work but only in the hands of experts?**
- **Is using it a black art like training neural networks?**

Yes, it is too good to be true!

- CART cannot quite deliver these extravagant results
  - Will not replace statistician or data analyst
- Yet CART comes remarkably close and is an extraordinary tool
- A radically new method for the analysis of data
  - Developed by statisticians but has a non-statistical feel
  - Akin to exploratory data techniques communicating via pictures
  - Computer intensive — links to AI and machine learning
  - Remarkable promises
    - Automatic analysis of data
    - No need to specify a model

Why didn't we learn about CART in school?

- **CART was slow to gain widespread recognition for several reasons**
  - **Monograph introducing CART is challenging to read**
    - **brilliant book overflowing with insights into tree growing methodology but fairly technical and breif**
  - **Method was not expounded in any textbooks**
  - **Originally taught only in advanced graduate statistics classes at a handful of leading Universities**
  - **Original standalone software came with slender documentation, and output was not self-explanatory**
  - **Method was such a radical departure from conventional statistics**

CART Biomedical Applications

- In spite of lack of marketing CART has had an enormous impact in biomedical studies
- A large number of cancer and coronary heart disease studies published
- Over 700 publications in last decade
- Pfizer used CART in its selection of subject for Viagra clinical trials
- Recently, a growing number of applications in other fields
  - Market research: direct mail response, market segmentation
  - Financial services: predicting bankruptcy, predicting refinance of home mortgages
  - Pharmaceuticals: identification of promising drugs

Journals Covering CART Analyses

- American Journal of Agriculture Economics
- American Criminal Law Review
- American Journal of Botany
- American Journal of Epidemiology
- American Statistical Association Journal
- Analytical and Quantitative Cytology and Histology
- Analytical Cellular Pathology
- Annals of Mathematics and Artificial Intelligence
- Annals of Statistics
- Annals of the New York Academy of Sciences
- Applied Artificial Intelligence
- Applied Statistics
- Archives of Psychiatric Nursing
- Arthritis and Rheumatism
- Artificial Intelligence in Medicine
- Automatica
- Biometrics
- Bone Joint Surgery
- Brain Topography
- British Journal of Hematology
- Cancer
- Cancer Treatment Reports
- Cardiology
- Circulation
- Cognitive Psychology
- Computer Journal
- Computer Speech and Language
- Computers and Biology and Medicine
- Computers and Biomedical Research
- Controlled Clinical Trials
- Death Studies
- Economics Letter
- Europhysics Letters
- Family Practice
- Financial Analysts Journal
- Financial Management
- Health Physic
- IEEE Transactions on Information Theory
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- International Journal of Forecasting
- International Journal of Man-Machine Studies
- International Neural Network Conference
- Investigative Radiology
- Journal of Clinical Oncology
- Journal of Biomedical Engineering
- Journal of Cancer Research and Clinical Oncology

Journals Covering CART Analyses

- Journal of Clinical Oncology
- Journal of Dental Research
- Journal of Direct Marketing
- Journal of Forensic Sciences
- Journal of Mental Health Administration
- Journal of Neurosurgery
- Journal of Psychiatric Research
- Journal of Public Health Dentistry
- Journal of Studies on Alcohol
- Journal of the American Academy of Dermatology
- Journal of the American Statistical Association
- Journal of the American Veterinary Medical Association
- Journal of the Operational Research Society
- Journal of Toxicology and Environmental Health
- Knowledge Acquisition
- Machine Learning
- Management Science
- Marketing Research: A Magazine of Management & Applications
- Medical Care
- Medical Decision Making
- Medical Information
- Methods of Information in Medicine
- Psychiatry
- Quarterly Journal of Business & Economics
- Revista Medica de Chile
- Scandinavian Journal of Gastroenterology
- Science
- Statistics in Medicine
- Statistical Science
- Statistics and Computing
- Statistics in Medicine
- Technometrics
- The American Statistician
- The Annals of Statistics
- The Journal of Pediatrics
- The Journal of Behavioral Economics
- The Journal of Finance
- The Journal of Industrial Economic
- The Journal of Rheumatology
- The Journal of Risk and Insurance
- The Journal of Wildlife Management
- The New England Journal of Medicine
- Therapie
- Ultrasonic Imaging
- World Journal of Surgery

Why is CART finally receiving more attention?

- **Rising interest in data mining**
  - **Availability of huge data sets requiring analysis**
  - **Need to automate or accelerate and improve analysis process Comparative performance studies**
- **Advantages of CART over other tree methods**
  - **handling of missing values**
  - **assistance in interpretation of results (surrogates)**
  - **performance advantages: speed, accuracy**
- **New software and documentation make techniques accessible to end users**
- **Word of mouth generated by early adopters**

CART is not alone in the field of tree structured analysis

- **First precursor was AID, Automatic Interaction Detection (1963)**
  - **Developed by economists at University of Michigan (Morgan & Sonquist)**
  - **Idea stimulated by the challenge of large survey data sets**
  - **AID fell into disrepute because of its capability of giving misleading results**
- **Followed by CHAID, as an enhancement of AID (Kass, 1980)**
  - **CHAID gained a foothold in commercial market research but not academic journals**
  - **CHAID and KnowledgeSEEKER™ inherit some serious shortcomings of AID**

AID and its progeny (such as CHAID) stimulated the research leading to CART

- **Used in critical applications such as medicine, and radar imaging**
  - **How to diagnose a patient**
    - **Is this patient at risk of dying?**
    - **Is this lump breast cancer?**
  - **Does this molecule contain chlorine?**
  - **What type of ship is our radar picking up?**
- **AID and CHAID were yielding unacceptable results; no place to hide when errors are made**
  - **Unlike market research & social science, where consequences of ERRORS are not so severe, or more difficult to detect**

Primary Research Leading to CART Conducted 1974-1984

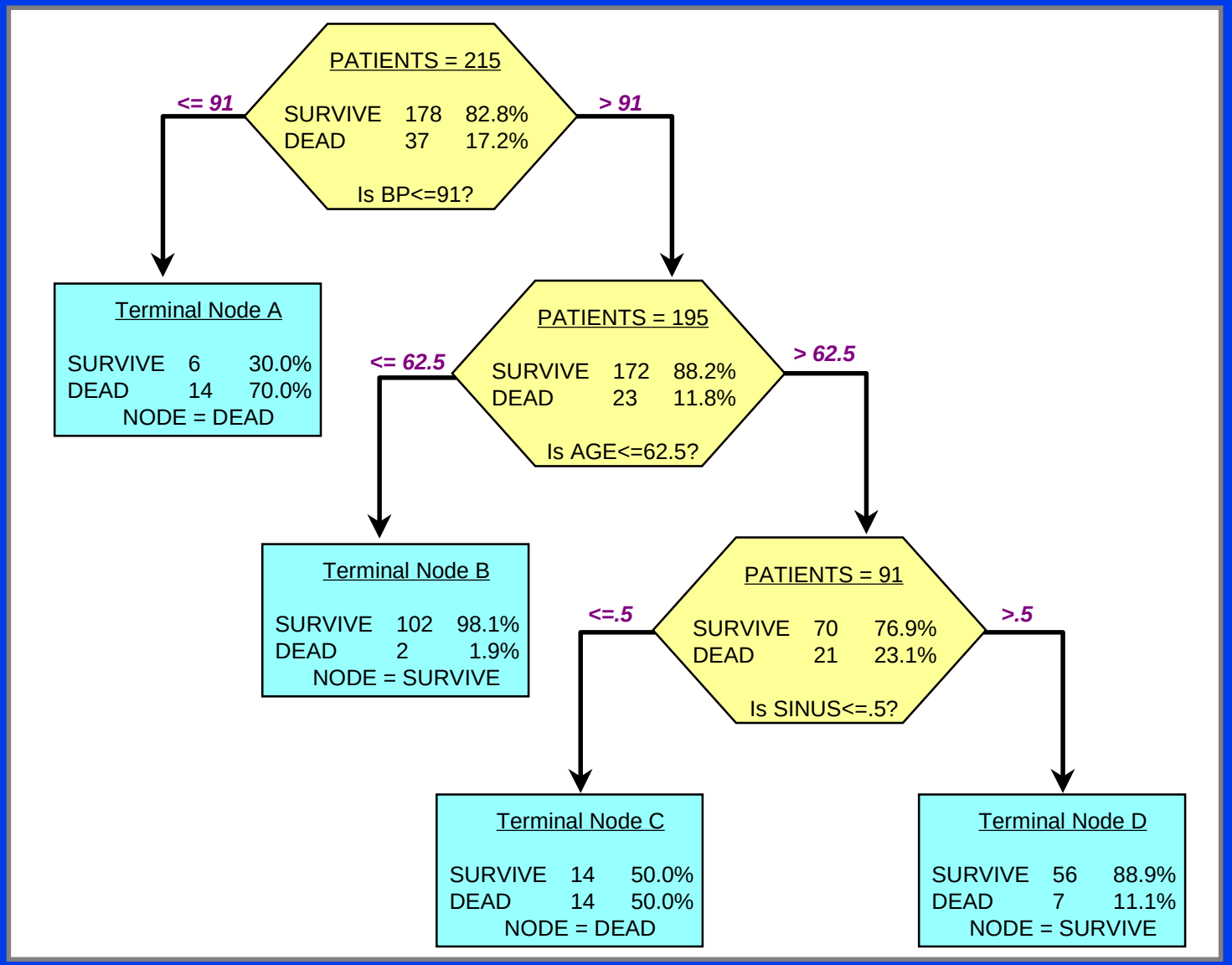
- **California Statistical Software, Inc.**
  - **Leo Breiman, University of California, Berkeley**
  - **Jerry Friedman, Stanford University**
  - **Charles J. Stone, University of California, Berkeley**
  - **Richard Olshen, Stanford University**
- **Released the original stand-alone CART distributed primarily to universities, research hospitals, and sophisticated industrial laboratories**
- **Software now incorporated in new versions co-developed with Salford Systems, the exclusive worldwide distributors of CART® software**

So what is CART?

- **Best illustrated with a famous example-the UCSD Heart Disease study**
- **Given the diagnosis of a heart attack based on**
  - **Chest pain, Indicative EKGs, Elevation of enzymes typically released by damaged heart muscle**
- **Predict who is at risk of a 2nd heart attack and early death within 30 days**
  - **Prediction will determine treatment program (intensive care or not)**
- **For each patient about 100 variables were available, including:**
  - **Demographics, medical history, lab results**
  - **19 noninvasive variables were used in the analysis**

2nd Heart Attack Risk Tree

- Example of a CLASSIFICATION tree
- Dependent variable is categorical (SURVIVE, DIE)
- Want to predict class membership



What's new in this report?

- Entire tree represents a complete analysis or model
- Has the form of a decision tree
- Root of inverted tree contains all data
- Root gives rise to child nodes
- Child nodes can in turn give rise to their own children
- At some point a given path ends in a terminal node
- Terminal node classifies object
- Path through the tree governed by the answers to QUESTIONS or RULES

Question is of the form: Is statement TRUE?

- **Is continuous variable $X \leq c$?**
- **Does categorical variable D take on levels i, j, or k?**
  - **e.g. Is geographic region 1, 2, 4, or 7?**
- **Standard split:**
  - **If answer to question is YES a case goes left; otherwise it goes right**
  - **This is the form of all primary splits**
- **Question is formulated so that only <u>two</u> answers possible**
  - **Called binary partitioning**
  - **In CART the YES answer always goes left**

Continuous Variable Splits

- **Is FITNESS-SCORE <= 3.1?**
  - **CART reports a split value which is the mid-point between two actual data values**
  - **E.g., actual data values might include 3.0 and 3.2 and the split is set at 3.1**
  - **best guess for dealing with future unseen data**
- **Is HOME-OWNER <= .5?**
  - **mid-point between the observed values of 0 and 1**

Classification is determined by following a case's path down the tree

- **Terminal nodes are associated with a single class**
  - **Any case arriving at a terminal node is assigned to that class**
- **In standard classification, tree assignment is not probabilistic**
  - **Another type of CART tree, the *class probability* tree does report distributions of class membership in nodes (discussed later)**
- **With large data sets we can take the empirical distribution in a terminal node to represent the distribution of classes**

Accuracy of a Tree

- For classification trees
  - All objects reaching a terminal node are classified in the same way
    - e.g. All heart attack victims with BP>91 and AGE less than 62.5 are classified as SURVIVORS
  - Classification is same regardless of other medical history and lab results
- Some cases may be misclassified:
  - Simplest measure:
    - R(T)=percent of learning sample misclassified in tree T
    - R(*t*)=percent of learning sample in node *t* misclassified
      - T identifies a TREE
      - *t* identifies a NODE

Prediction Success Table

	Classified As			
TRUE	Class 1 "Survivors"	Class 2 "Early Deaths"	Total	% Correct
Class 1 "Survivors"	**158**	**20**	**178**	**88%**
Class 2 "Early Deaths"	**9**	**28**	**37**	**75%**
Total	**167**	**48**	**215**	**86.51%**

- **In CART terminology the performance is described by the error rate R(T), where T indexes a specific tree**
  - **In the example R(T)=1-.8651=.1349**

Example Tree Exhibits Characteristic CART Features

- **Tree is relatively simple: focus is on just 3 variables**
  - **CART trees are often simple relative to the problem**
  - **but trees can be very large if necessary (50, 200, 500+ nodes)**
- **Accuracy is often high — about as good as any logistic regression on more variables**
  - **Not easy to develop a parametric model significantly better than a CART tree, unless data exhibits linear structure**
  - **logistic regression will require intensive modeling effort**
- **Results can be easily explained to non-technical audience**
  - **More enthusiasm from decision makers for CART trees than for black box neural nets**

Interpretation and use of the CART Tree

- **Practical decision tool**
  - **Trees like this are used for real world decision making**
    - **Rules for physicians**
    - **Decision tool for a nationwide team of salesmen needing to classify potential customers**
- **Selection of the most important prognostic variables**
  - **Variable screen for parametric model building**
  - **Example data set had 100 variables**
  - **Use results to find variables to include in a logistic regression**
- **Insight — in our example AGE is not relevant if BP is not high**
  - **Suggests which interactions will be important**

Core Questions

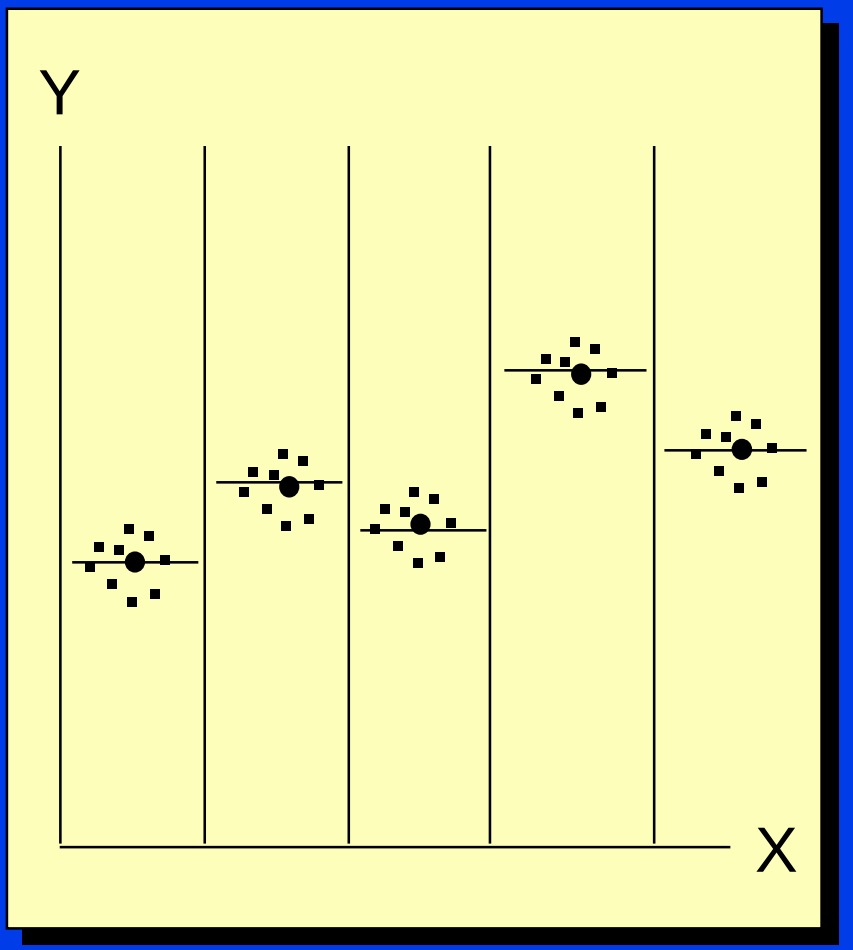
- **How is a CART tree generated?**
- **What assurances do we have that it will give correct answers?**
- **Is CART better than other decision tree tools?**

CART is a form of Binary Recursive Partitioning

- **Data is split into two partitions**
  - **thus “binary” partition**
- **Partitions can also be split into sub-partitions**
  - **hence procedure is recursive**
- **CART tree is generated by repeated partitioning of data set**

CART is like other Non-parametric procedures

- Consider Y as a function of X
- Partition X into regions
  - ■ = data point
  - ● = summary
- Use median or mean of the function of Y values to relate Y to X
- No attempt to fit a function
- Each region gets its own fitted Y value, a constant
- Straightforward in two or three dimensions



Problems with Non-parametric Approaches

- **Curse of dimensionality**
  - **Consider 100 variables each with only 2 possible values**
  - **Have 2^{100} regions=1.2677E+30 regions**
  - **Even $2^{35} \approx$ 34 billion regions**
  - **Problem vastly larger still with continuous variables**
- **How to determine size of intervals for continuous variables**
- **How to handle unordered categorical variables**
- **Results sensitive to choice of metric**
- **Provide no insight into data structure**
  - **Often such techniques bog down in computational difficulties for social science**
- **No guidance on what variables to include — how to decide what matters**

CART is also Non-parametric—No Predetermined Functional Form

- **Approximates the data pattern via local summaries**
- ***BUT***
  - **Determines which variables to use dynamically**
  - **Determines which regions to focus on dynamically**
  - **Focuses on only a moderate and possibly very small number of variables**
  - **All automatically determined by the data**
- **We saw in the CHD study CART focused on just 3 of 19 variables**

Tracing a CART analysis - Three types of flower

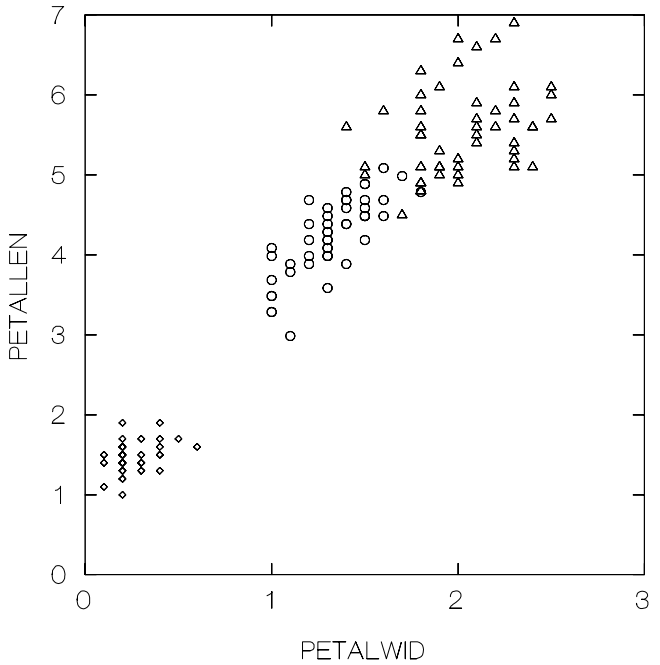
Iris Setosa

Iris Versicolor

Iris Virginica

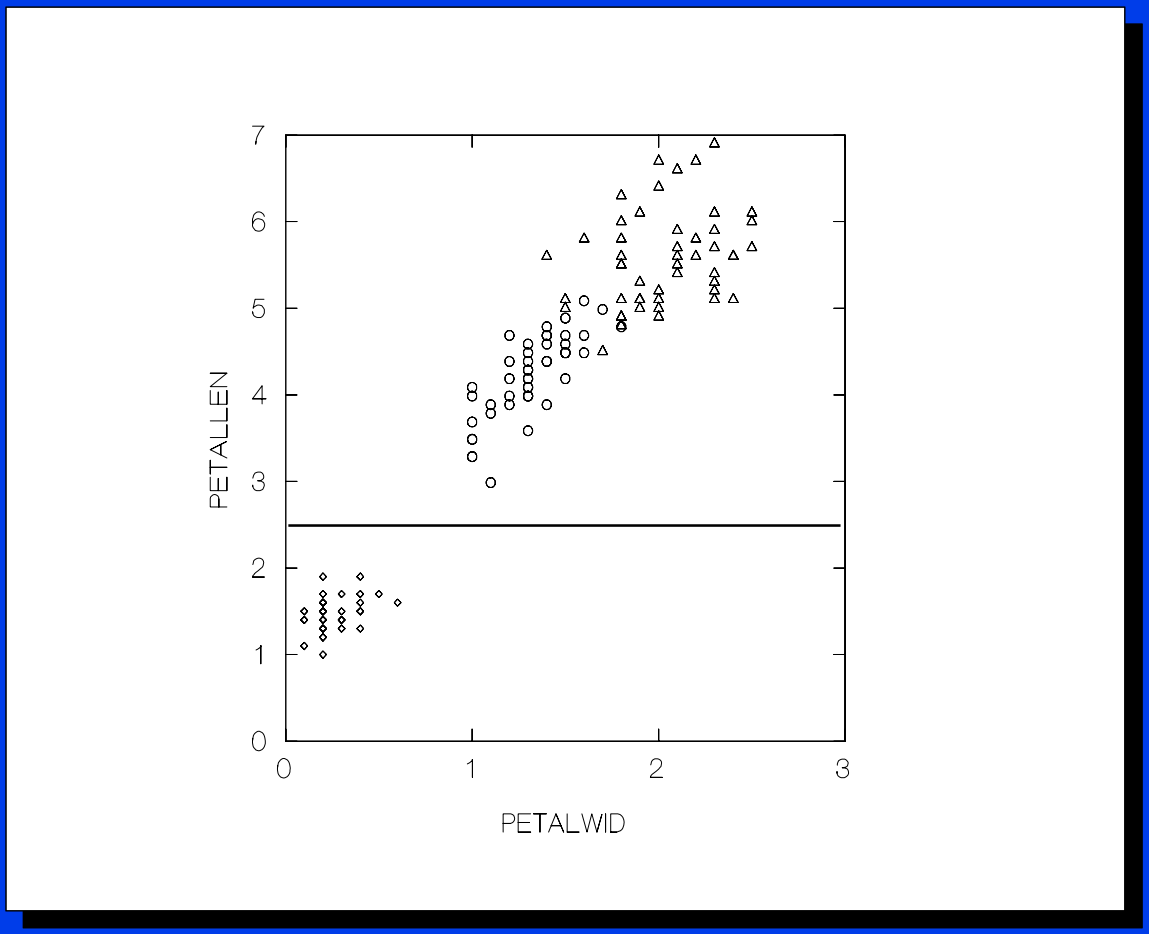
Tracing a CART Analysis

- **IRIS data set**
- **3 classes of species**
- **We can use the PETALLEN & PETALWID variables to produce a tree**
- **Key**
  - **O = Species 1**
  - **Δ = Species 2**
  - **◊ = Species 3**



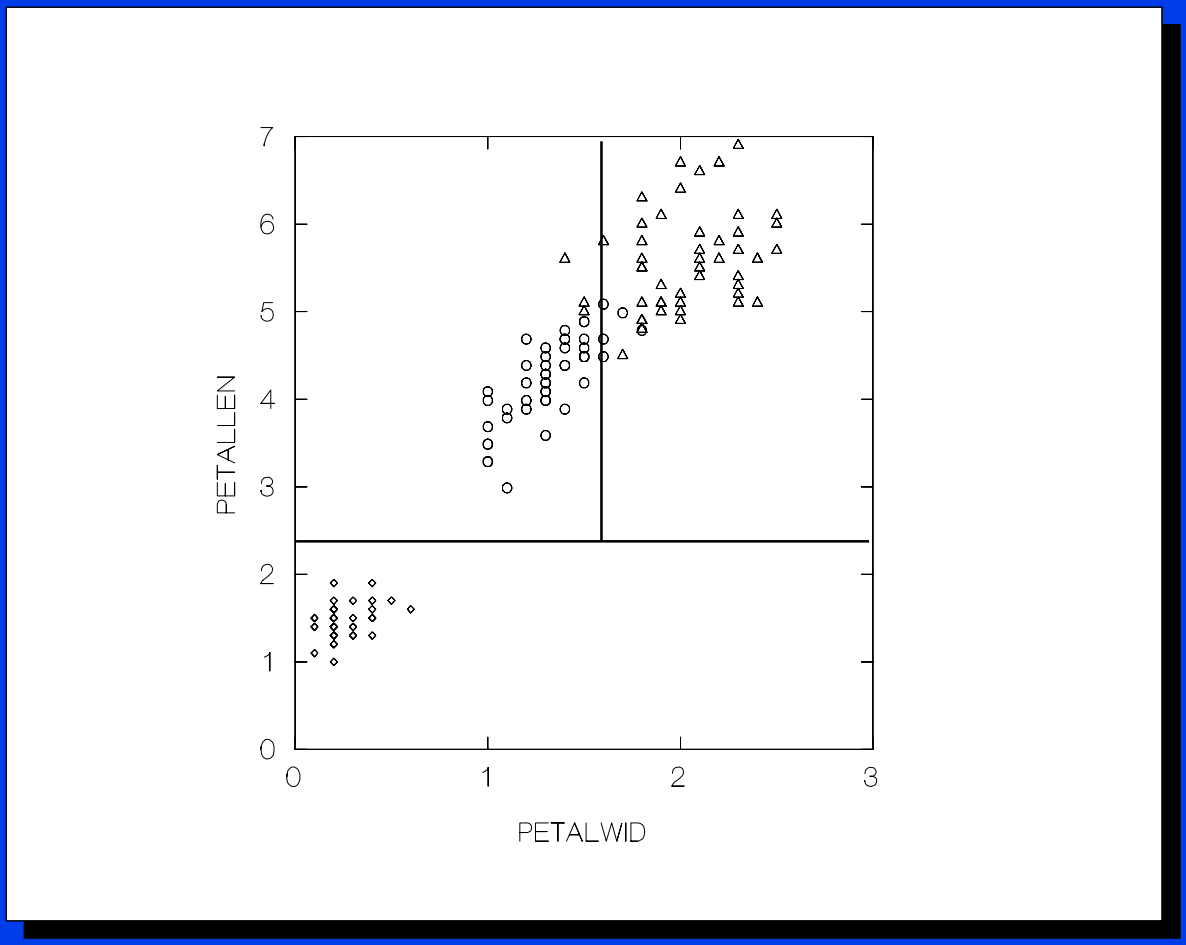
First Split: Partition Data Into Two Segments

- Partitioning line parallel to an axis
- Root node split first
  - ≤ 2.450
  - Isolates all the type 1 species from rest of the sample
- This gives us two child nodes
  - One is a Terminal Node with only type 1 species
  - The other contains only type 2 and 3
- Note: entire data set divided into two separate parts



Second Split: Partitions Only Portion of the Data

- Again, partition with line parallel to one of the two axes
- CART selects PETALWID to split this NODE
  - Split it at ≤ 1.75
  - Gives a tree with a misclassification rate of 4%
- Split applies only to a single partition of the data
- Each partition is analyzed separately



Key Components of Tree Structured Data Analysis

- **Tree growing**
  - **Splitting Rules**
  - **Stopping Criteria**
- **Tree Pruning**
- **Optimal Tree Selection**
  - **We will first conduct a brief tour of the entire process**

Splitting Rules

- One key to generating a CART tree is asking the right question
- Question always in the form
  - Is CONDITION ≤ VALUE
- In the simplest case CONDITION is a measured variable and VALUE is a constant
- We saw examples earlier:
  - Is AGE ≤ 62.5
  - Is BP ≤ 102
- More complex conditions
  - Boolean combinations, Linear combinations

Searching all Possible Splits

- **For any node CART will examine ALL possible splits**
  - **Computationally intensive but there are only a finite number of splits**
- **Consider first the variable AGE--in our data set has minimum value 40**
  - **The rule**
    - **Is AGE ≤ 40? will separate out these cases to the left — the youngest persons**
- **Next increase the AGE threshold to the next youngest person**
    - **Is AGE ≤ 43**
    - **This will direct six cases to the left**
- **Continue increasing the splitting threshold value by value**

Split Tables

Sorted by Age

AGE	BP	SINUST	SURVIVE
40	91	0	SURVIVE
40	110	0	SURVIVE
40	83	1	DEAD
43	99	0	SURVIVE
43	78	1	DEAD
43	135	0	SURVIVE
45	120	0	SURVIVE
48	119	1	DEAD
48	122	0	SURVIVE
49	150	0	DEAD
49	110	1	SURVIVE

Sorted by Blood Pressure

AGE	BP	SINUST	SURVIVE
43	78	1	DEAD
40	83	1	DEAD
40	91	0	SURVIVE
43	99	0	SURVIVE
40	110	0	SURVIVE
49	110	1	SURVIVE
48	119	1	DEAD
45	120	0	SURVIVE
48	122	0	SURVIVE
43	135	0	SURVIVE
49	150	0	DEAD

For any Variable at most N Different Splits for Sample Size N

- If there are 215 cases in the data set and 100 variables
- There will be at most 215*100=21,500 possible splits
- Often many fewer
  - Not all values are unique
  - Many variables are dummies or categorical with just a few values
- Rapidly processed on a PC or minicomputer
- As an example, on a DEC Alpha server we search a tree with 35,000 cases and 296 variables to a tree with 71 nodes in under 5 minutes
- Similar results with a high-speed Pentium

Categorical Predictors

- **CART considers all possible splits based on a categorical predictor**
- **Example: four regions - A, B, C, D**

	Left	Right
1	A	B, C, D
2	B	A, C, B
3	C	A, B, D
4	D	A, B, C
5	A, B	C, D
6	A, C	B, D
7	A, D	B, C

- **Each decision is a possible split of the node and each is evaluated for improvement of impurity**

Categorical Predictors

- When a predictor variable is categorical CART considers all possible subsets
  - e.g. If RACE has 5 levels then there are 2^{5-1} =16 race groupings; all searched as if they were separate variables
- When number of categorical levels is high compute burden can be overwhelming
- CART allows 15 levels by default and allows setting to any number
  - WARNING: for each three levels above 15, compute time goes up by a factor of 10
  - So if a 15 level variable in a problem takes 3 minutes
  - A 21 level variable in the same problem will take 300 minutes
  - A 27 level variable in the same problem will take 30,000 minutes or 500 hrs
- Problem can come up with SIC code or occupational category
  - You may need to create subgroups first

Exception to this Rule: Binary DEPENDENT Variable Problems

- **Here special algorithms reduce compute time to linear in number of levels**
- **30 levels takes twice as long as 15 not 10,000+ times as long**
- **When you have high-level categorical variables in a multi-class problem**
  - **Create a binary classification problem**
  - **Try different definitions of DPV (which groups to combine)**
  - **Explore predictor groupings produced by CART**
  - **From a study of all results decide which are most informative**
  - **Create new grouped predicted variable for multi-class problem**

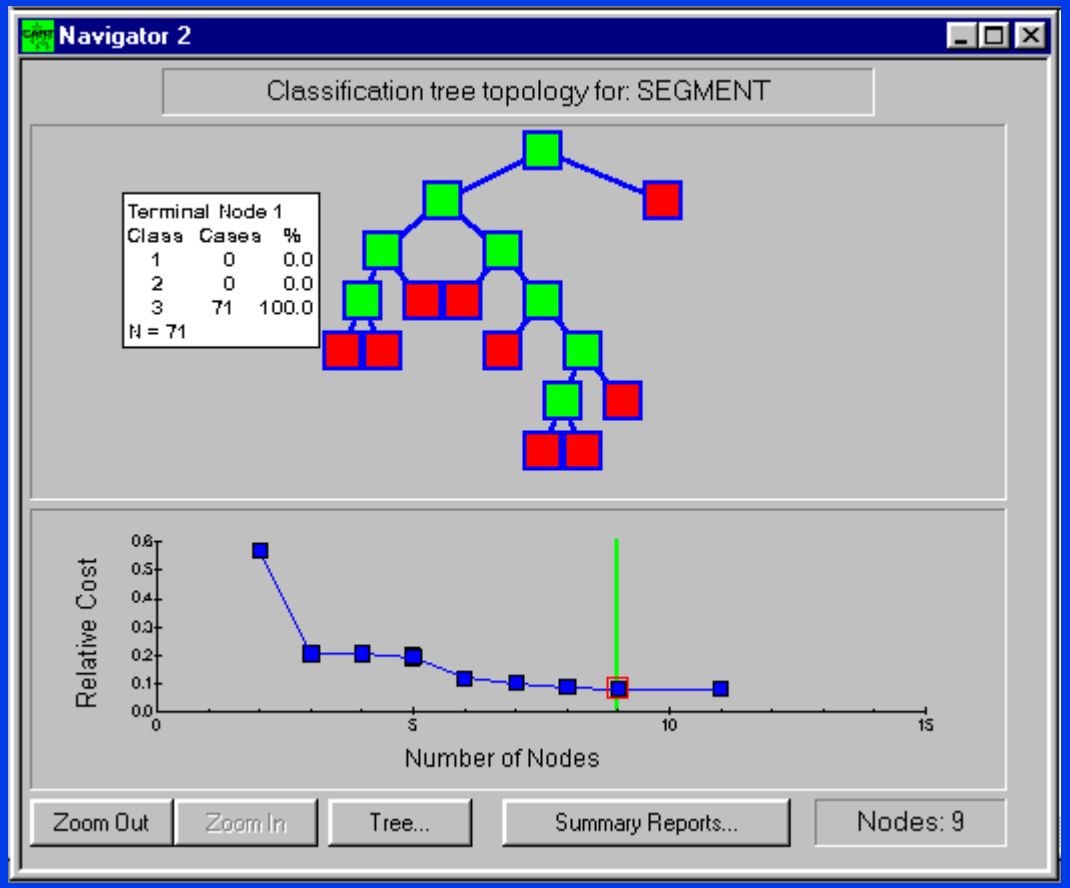
Detailed Example: GYM Cluster Model

- **Problem: needed to understand a market research clustering scheme**
- **Clusters were created using 18 variables and conventional clustering software**
- **Wanted simple rules to describe cluster membership**
- **Experimented with CART tree to see if there was an intuitive story**

Gym Example - Variable Definitions

ID	**Identification # of member**
CLUSTER	**Cluster assigned from clustering scheme (10 level categorical coded 1-10)**
ANYRAQT	**Racquet ball usage (binary indicator coded 0, 1)**
ONRCT	**Number of on-peak racquet ball uses**
ANYPOOL	**Pool usage (binary indicator coded 0, 1)**
ONPOOL	**Number of on-peak pool uses**
PLRQTPCT	**Percent of pool and racquet ball usage**
TPLRCT	**Total number of pool and racquet ball uses**
ONAER	**Number of on-peak aerobics classes attended**
OFFAER	**Number of off-peak aerobics classes attended**
SAERDIF	**Difference between number of on- and off-peak aerobics visits**
TANNING	**Number of visits to tanning salon**
PERSTRN	**Personal trainer (binary indicator coded 0, 1)**
CLASSES	**Number of classes taken**
NSUPPS	**Number of supplements/vitamins/frozen dinners purchased**
SMALLBUS	**Small business discount (binary indicator coded 0, 1)**
OFFER	**Terms of offer**
IPAKPRIC	**Index variable for package price**
MONFEE	**Monthly fee paid**
FIT	**Fitness score**
NFAMMEN	**Number of family members**
HOME	**Home ownership (binary indicator coded 0, 1)**

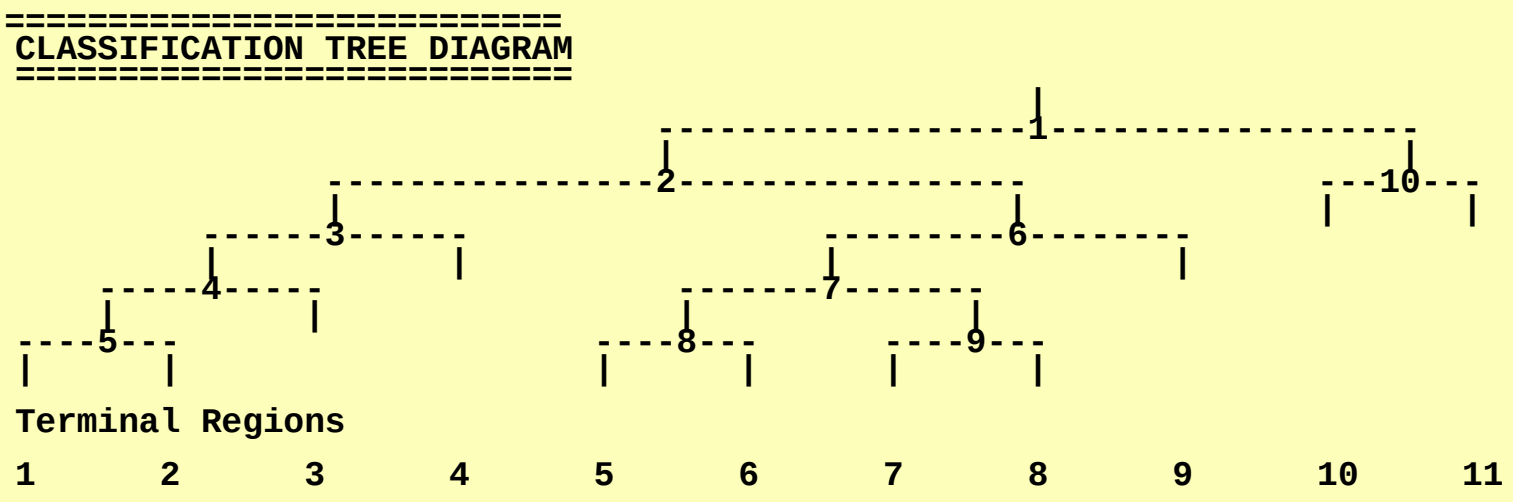
GUI CART Output:
Tree Navigator for GYM Example



Classic CART output: Tree Sequence

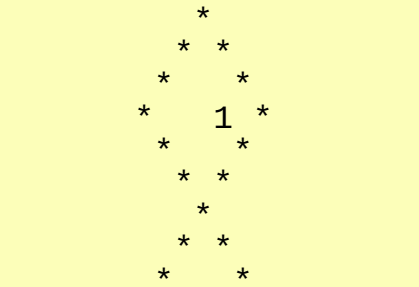
```
=============
TREE SEQUENCE
=============

Dependent variable: CLUSTER

  Terminal            Test Set          Resubstitution   Complexity
Tree Nodes          Relative Cost        Relative Cost    Parameter
-------------------------------------------------------------------
   1**  11     0.168493 +/- 0.005304         0.195980     0.012555
   2    10     0.208200 +/- 0.005482         0.214243     0.014769
   3     9     0.249918 +/- 0.004828         0.245512     0.025262
   4     7     0.340175 +/- 0.004018         0.323120     0.031346
   5     6     0.413026 +/- 0.003861         0.388242     0.052599
   6     5     0.474169 +/- 0.003117         0.461266     0.058981
   7     4     0.558584 +/- 0.003051         0.543542     0.066451
   8     3     0.686229 +/- 0.002973         0.671612     0.103433
   9     2     0.835007 +/- 0.001592         0.833190     0.130491
  10     1     1.000000 +/- 0.000069         1.000000     0.134717

Initial misclassification cost = 0.807546
Initial class assignment =  3
```

Classic CART Output
Skeleton Tree Diagram:



Classic CART Output: Node 1 Detail

```
================
NODE INFORMATION
================
```



Node 1 was split on variable HOME
A case goes left if variable HOME <= 0.500000
Improvement = 0.136091 C. T. = 0.134707

Node	Cases	Class	Cost
1	42557	3	0.807546
2	27702	1	0.786652
10	14855	3	0.452744

27702 14855

	Number Of Cases			Within Node Prob.		
Class	Top	Left	Right	Top	Left	Right
1	6009	5964	45	0.141689	0.213348	0.003113
2	2916	2858	58	0.064625	0.096093	0.003771
3	8418	259	8159	0.192454	0.008983	0.547256
4	3849	3765	84	0.089112	0.132242	0.005706
5	3149	3103	46	0.069156	0.103384	0.002964
6	3717	3677	40	0.085714	0.128638	0.002706
7	2621	2526	95	0.066911	0.097832	0.007115
8	6318	0	6318	0.145417	0.000000	0.426628
9	4166	4156	10	0.105093	0.159055	0.000740
10	1394	1394	0	0.039829	0.060426	0.000000

Classic CART Output:
Competitors & Surrogates for Node 1

	Surrogate		Split	Assoc.	Improve.
1	PERSTRN	s	0.500000	0.914703	0.127490
2	FIT	s	6.353424	0.523818	0.110693
3	NFAMMEM	s	4.500000	0.206078	0.030952
4	TANNING	s	2.500000	0.029640	0.007039
5	NSUPPS	s	30.500000	0.029236	0.003991
6	CLASSES	s	1.500000	0.011499	0.006856

	Competitor	Split	Improve.
1	PERSTRN	0.500000	0.127490
2	FIT	3.107304	0.111190
3	ANYRAQT	0.500000	0.102227
4	PLRQTPCT	0.031280	0.090114
5	TPLRCT	0.500000	0.090085
6	ONRCT	0.500000	0.077620
7	NFAMMEM	2.500000	0.055136
8	CLASSES	0.500000	0.045368
9	IPAKPRIC	7.632159	0.033419
10	OFFAER	0.500000	0.022993

Classic CART Output: Terminal Node Report

```
=========================
TERMINAL NODE INFORMATION
=========================

(Test Set)
                                                |--LEARNING SET--||----TEST SET----|
Node    N        Prob Class      Cost Class     N        Prob     N        Prob
-------------------------------------------------------------------------------
  1  2157    0.057568  10    0.341883     1   501    0.205206     9    0.034556
  (   547    0.042367        0.119094)    2     0    0.000000     0    0.000000
         Parent C.T. =      0.033113     3     0    0.000000     0    0.000000
                                         4    19    0.007641     0    0.000000
                                         5     0    0.000000     0    0.000000
                                         6   207    0.082917     2    0.009093
                                         7   104    0.046119    30    0.075445
                                         8     0    0.000000     0    0.000000
                                         9     0    0.000000     0    0.000000
                                        10  1326    0.658117   506    0.880906

  2  2363    0.055145   6    0.205059     1   307    0.131271     8    0.031321
  (   233    0.041550        0.077476)    2     0    0.000000     0    0.000000
         Parent C.T. =      0.033113     3     0    0.000000     0    0.000000
                                         4    22    0.009236     0    0.000000
                                         5     0    0.000000     0    0.000000
                                         6  1901    0.794941   199    0.922524
                                         7    79    0.036572     0    0.000000
                                         8     0    0.000000     0    0.000000
                                         9     0    0.000000     0    0.000000
                                        10    54    0.027979    26    0.046155
```

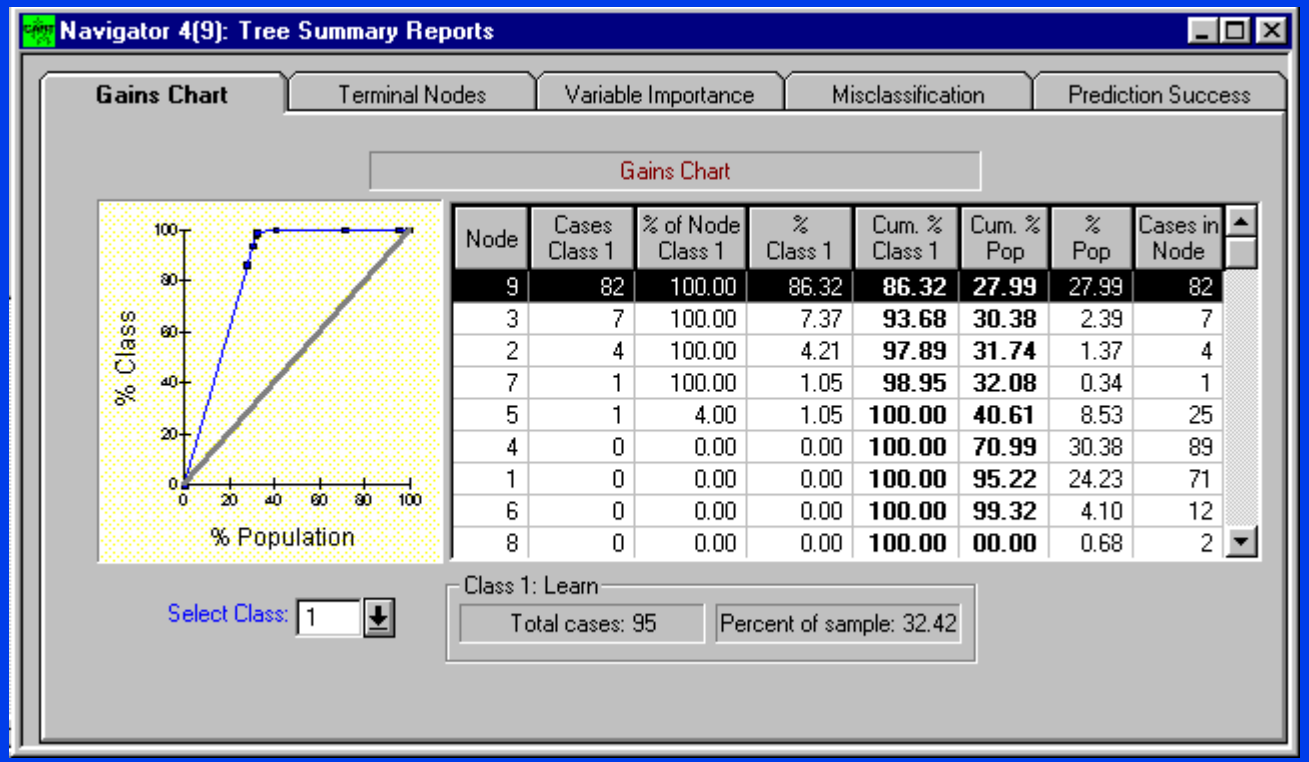
Classic CART Output: Misclassification by Class

```
==========================
MISCLASSIFICATION BY CLASS
==========================

               |--------TEST SAMPLE---------||------LEARNING SAMPLE-------|
           Prior          N Mis-                          N Mis-
Class      Prob.     N  Classified         Cost     N  Classified         Cost
-------------------------------------------------------------------------------
  1     0.141689   871         36     0.041332  6009        960     0.159760
  2     0.064625   222         52     0.234234  2916        254     0.087106
  3     0.192454   927         38     0.040992  8418        274     0.032549
  4     0.089112   478        103     0.215481  3849       1871     0.486100
  5     0.069156   209          2     0.009569  3149         71     0.022547
  6     0.085714   445        246     0.552809  3717       1816     0.488566
  7     0.066911   628         98     0.156051  2621        705     0.268981
  8     0.145417   743        127     0.170929  6318        634     0.100348
  9     0.105093   937         19     0.020277  4166         97     0.023284
 10     0.039829   540         34     0.062963  1394         68     0.048780
-------------------------------------------------------------------------------
 Tot    1.000000  6000        755              42557       6750

                    Overall 12.58%            Overall     15.86%
```

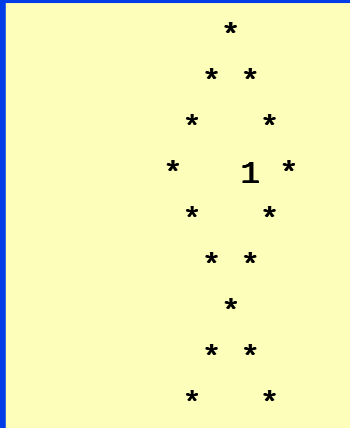
Classic CART Output: Variable Importance

```
====================
VARIABLE IMPORTANCE
====================

                 Relative     Number Of   Minimum
               Importance    Categories  Category
-------------------------------------------------
FIT            100.000000
ANYRAQT         76.030205
PLRQTPCT        73.177528
TPLRCT          73.155396
ONRCT           69.397400
NFAMMEM         52.537178
HOME            49.884525
PERSTRN         46.731873
CLASSES         46.349724
ONAER           43.151215
OFFAER          32.945679
IPAKPRIC        29.908035
NSUPPS          29.102987
SAERDIF         23.325085
TANNING         22.936609
ONPOOL          13.036715
ANYPOOL         10.677163
SMALLBUS         0.244279
```

GUI CART Output: Example Summary Reports



GYMEXAM using Twoing-Root Node Split

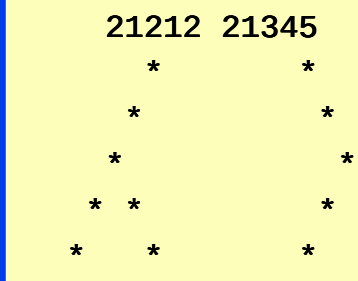


Node 1 was split on variable FIT

A case goes left if variable FIT <= 3.107304

Improvement = 0.903798 C. T. = 0.135800

Node	Cases	Class	Cost
1	42557	3	0.807546
2	21212	1	0.727528
8	21345	3	0.612371

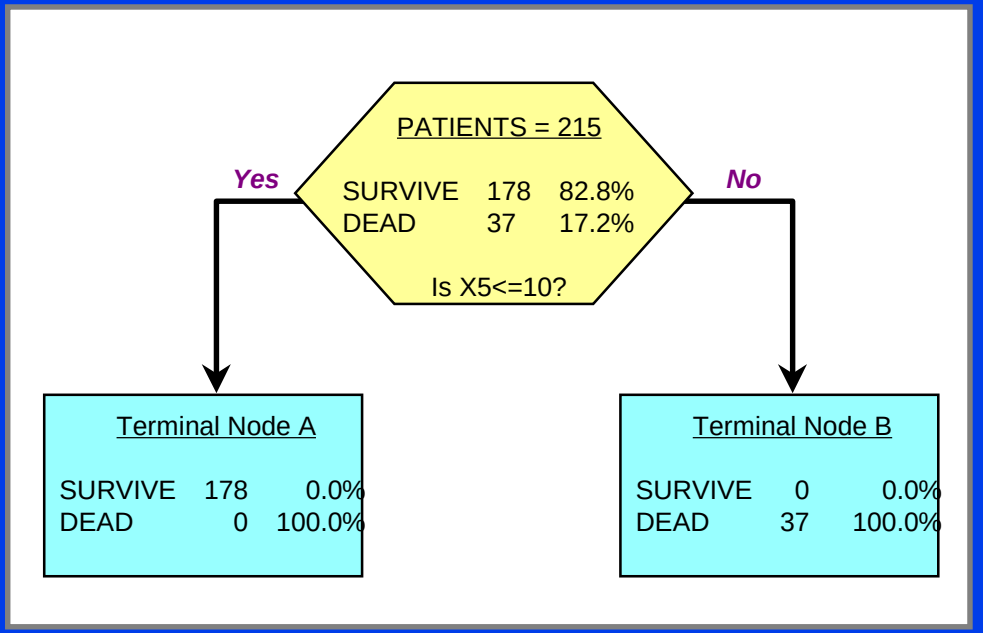


	Number Of Cases			Within Node Prob.		
Class	Top	Left	Right	Top	Left	Right
1	6009	5958	51	0.141689	0.272472	0.002483
2	2916	2	2914	0.064625	0.000086	0.133321
3	8418	205	8213	0.192454	0.009090	0.387629
4	3849	3803	46	0.089112	0.170765	0.002199
5	3149	0	3149	0.069156	0.000000	0.142766
6	3717	3067	650	0.085714	0.137170	0.030943
7	2621	2571	50	0.066911	0.127297	0.002635
8	6318	48	6270	0.145417	0.002143	0.297919
9	4166	4164	2	0.105093	0.203729	0.000104
10	1394	1394	0	0.039829	0.077249	0.000000

- Note that this twoing split yields a result MUCH more balanced split both in size of children and in the number of classes going left and right.

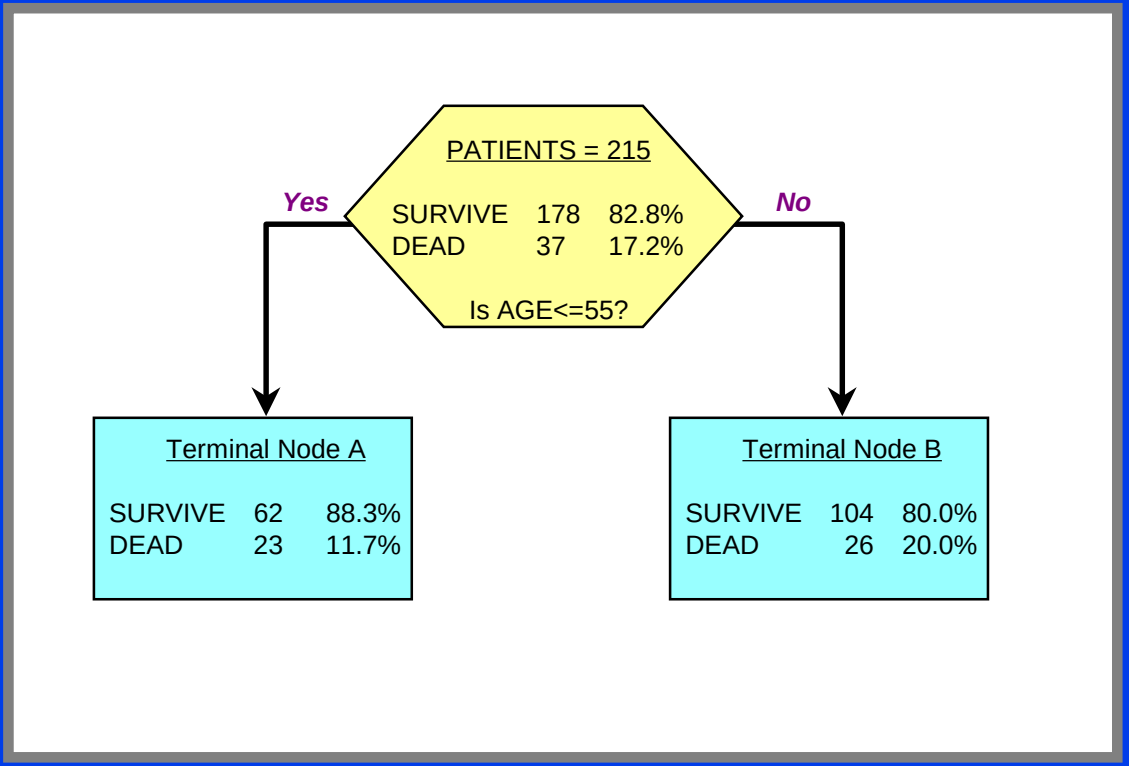
What makes one splitting rule better than another?

- Sometimes easy to tell we have the right question,
- e.g., in the San Diego Heart Attack study, if we had a hypothetical variable X5 such that:
  - 37 DEAD 178 SURVIVE
  - Is X5 ≤ 10 ?
  - Yes (0 DEAD 178 SURVIVE)
    No (37 DEAD , 0 SURVIVE)
- We have a perfect classification and we are done.
  - Usually we don't find such perfect splitters



Suppose instead we were looking at the split:

- **Is AGE ≤ 55?**
  - **Yes: 23 DEAD**
    62 SURVIVE
  - **No: 26 DEAD**
    104 SURVIVE
- **To assess, we may want to compare this to another split**



Which split is better?

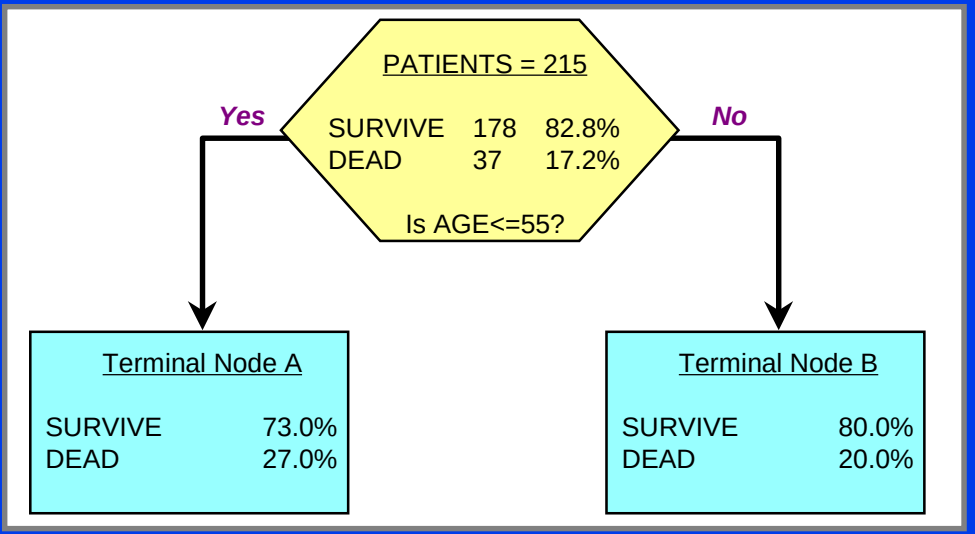
- Easier to assess if we use proportions

.17 DEAD .83 SURVIVE

Is AGE ≤ 55

40% Yes
.27 DEAD
.73 SURVIVE

60% No
.20 DEAD
.80 SURVIVE



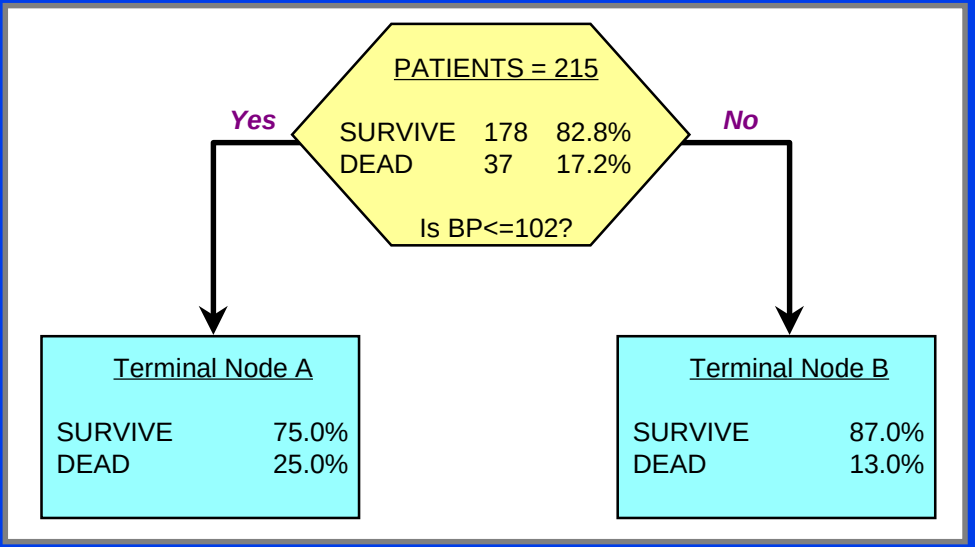
- Consider another split

.17 DEAD .83 SURVIVE

Is BP ≤ 102

32% Yes
.25 DEAD
.75 SURVIVE

68% No
.13 DEAD
.87 SURVIVE



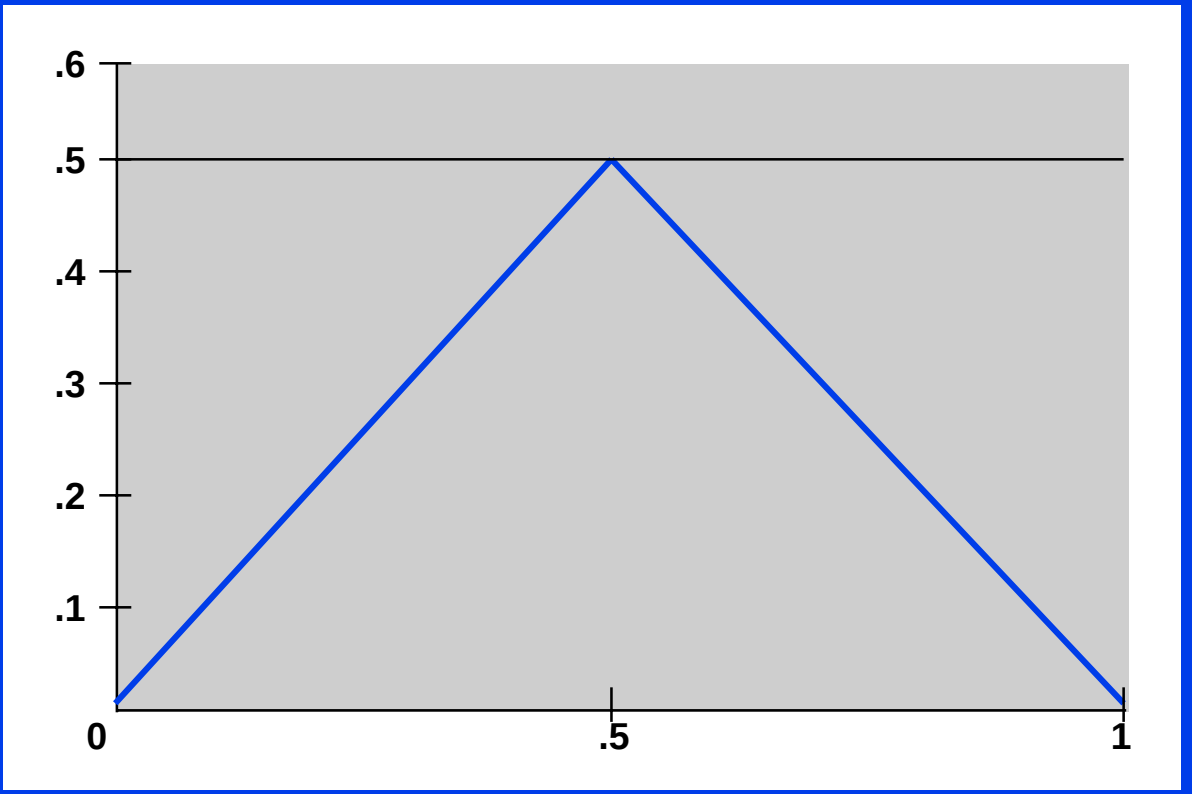
- This latter split seems to be a little better
- Relatively less class DEAD in the left node
- More of class SURVIVE in the right node

Evaluation of Splits with the Impurity Function

- **CART evaluates the goodness of any candidate split using an impurity function**
- **A node which contains members of only one class is perfectly pure**
  - **Obviously such a node is clearly classifiable**
  - **No ambiguity**
  - **Want impurity measure = 0**
- **A node which contains an equal proportion of every class is least pure**
  - **No easy way to classify — any class is as good as any other**
  - **Highly ambiguous**
  - **Want impurity measure to be maximal (scaling sets max to 1) when all proportions equal**

Many Impurity Functions to Choose From

- i(t)=impurity measure of the node t
- Consider i(t)=1-max p(j|t)
- e.g. In two class problem
  - Function min (p, 1-p)
    p ranges 0,1
- This is a linear impurity function

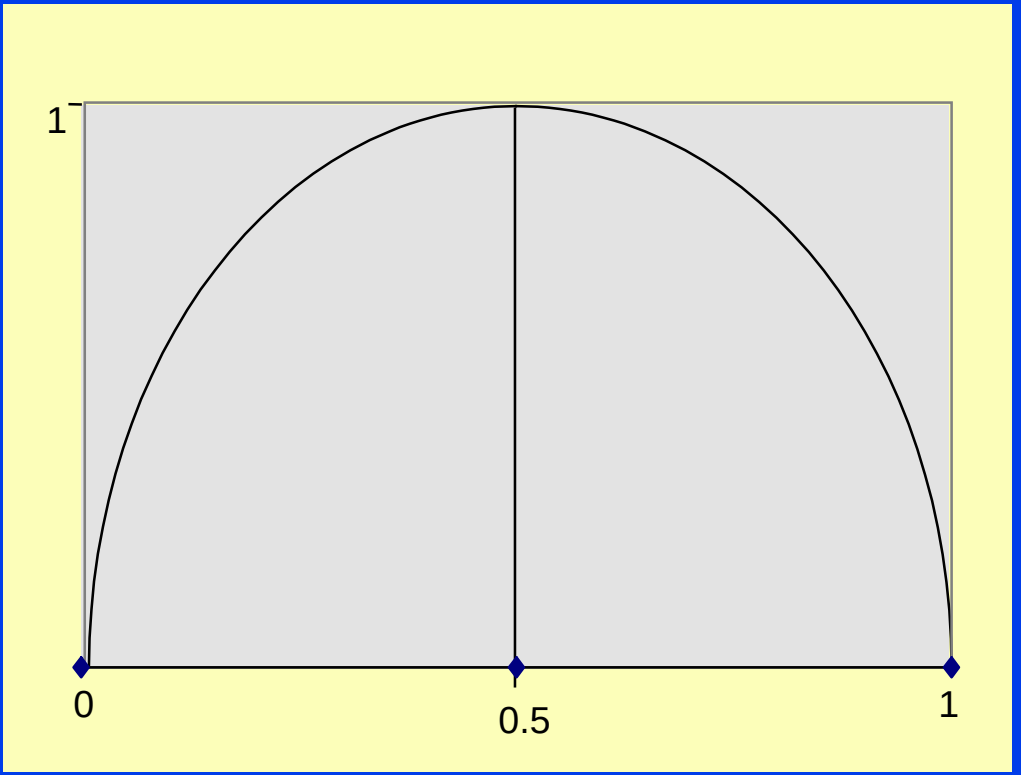


Linear Impurity Function

- **Has desirable properties:**
  - **Maximum value occurs with uniform distribution of classes**
  - **Minimum value occurs when all node members are in same class**
  - **Is symmetric — doesn't matter which class dominates node**
- ***Problems* with Linear Impurity:**
  - **It does NOT adequately reward purer nodes**
  - **Need the impurity function to accelerate towards 0 as node becomes pure**

A Number of Good Impurity Functions are Available

- You can experiment with several in CART
- A simple example of a better impurity function
  - ⇨ $i(t) = 4p\,(1-p)$
- Scaling makes i(t)=1 when p=0.5
- 4 * 0.5 *0.5 = 1

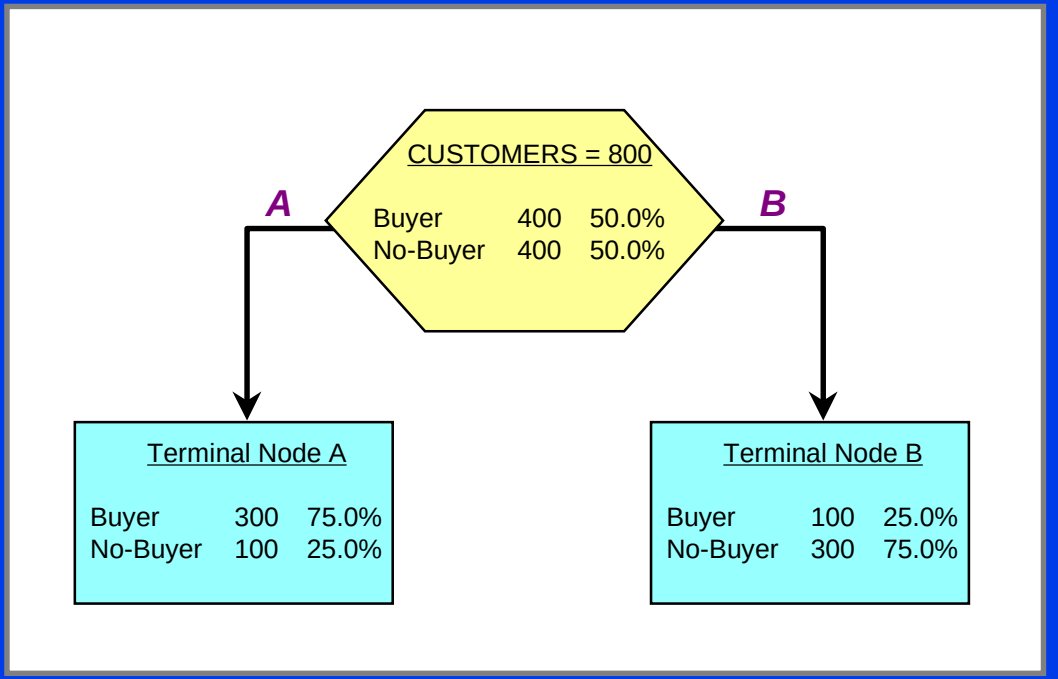


Why impurity?
Why not predictive accuracy?

- (1) Can almost always improve purity
  - Often unable to improve accuracy for a specific node
  - we will see later: if parent and both children are all assigned to same class then split does nothing for accuracy
- (2) Predictive accuracy is the long run objective of the tree
  - This goal is not served well by maximizing it at every node!
- Instead we need splitting rules that encourage good tree evolution
  - Since splitting is myopic (looks only at current node — the next step)
  - Need rules that are in some sense geared towards the long run outcome

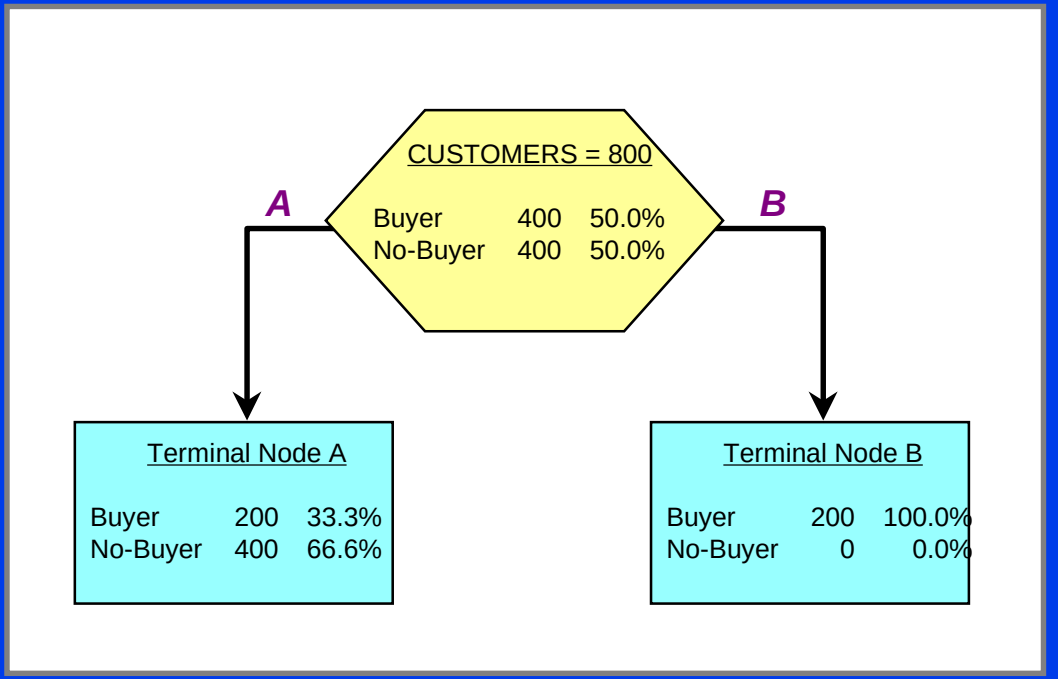
Example of Predictive Accuracy Splitting Rule

- Split takes error rate of 400 in root down to 200 overall
  - 100 incorrect in left child node
  - 100 incorrect in right child node
  - A node classifies all its members into one class



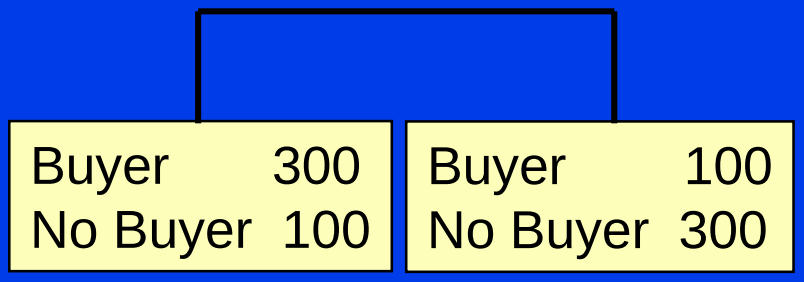
Now Consider an Alternative Split with Same Error Rate

- This also splits the root node down to 200 incorrect overall
  - 200 incorrect in left child node
  - 0 incorrect in right child node

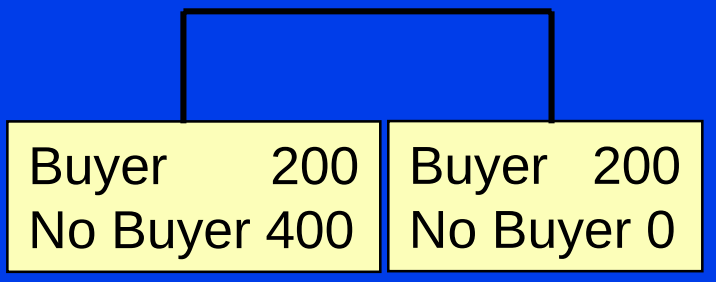


This Latter Split may be Thought of as "Better"

- **One child node is perfectly pure**
- **No more work needed for subsequent splits — this path is done**
- **A linear impurity criterion would rate both splits equally**
- **The nonlinear impurity criterion would favor the 2nd split**



Split #1

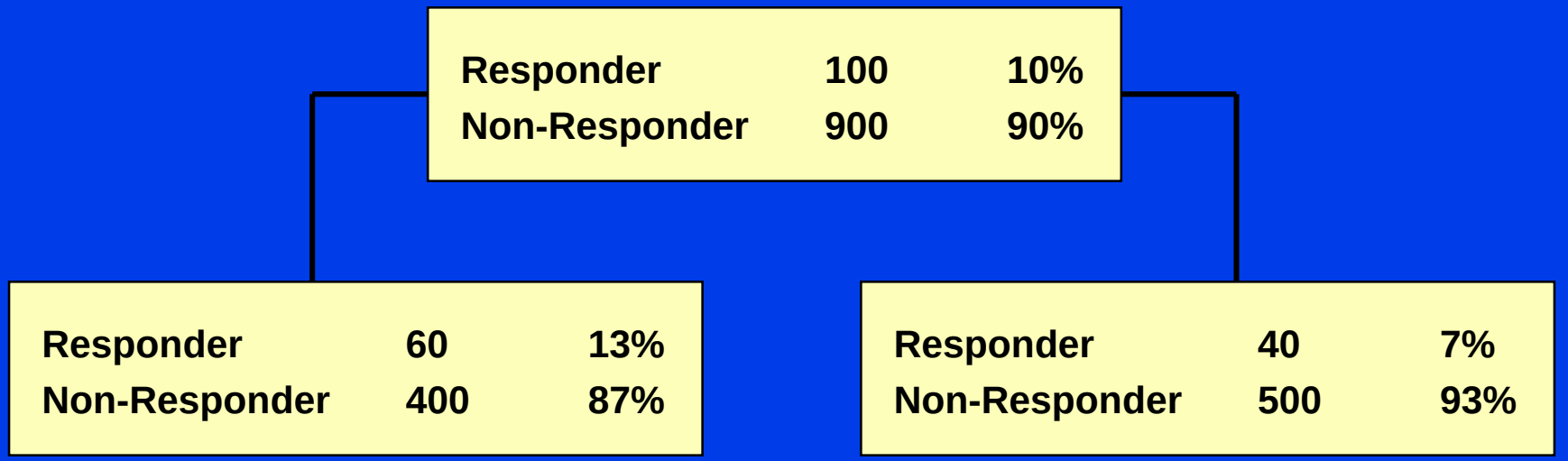


Split #2

Classification Trees Classify - They don't assign probabilities

- **Classification Trees do not assign probabilities to outcomes - although we the analysts look at the probabilities**
- **In a classification tree node, all cases in that node are classified in the same way**
- **For example, in direct mail targeting a node is either a responder node or a non-responder node**
- **Node class assignment applies to all cases in a node**

Some splits assign <u>both</u> children to the same class



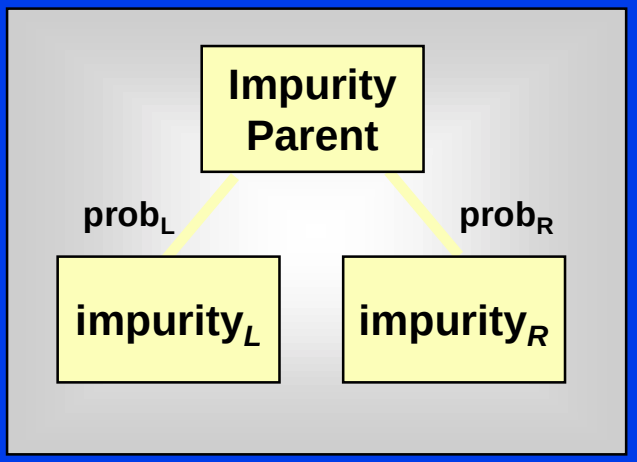
- From a direct mail perspective, any response rate above 1% is good
- So both children classified as responder nodes
- Eventually, though further splits we may find a non-responder node

The Improvement Measure is Decrease in Impurity

$$\Delta(t,s) = i(t) - p_L\, i(t_L) - p_R\ \ i(t_R)$$

Parent node impurity minus weighted average of the impurities in each child node

- p_L = probability of case going left (fraction of node going left)
- p_R = probability of case going right (fraction of node going right)
- t = node
- s = splitting rule

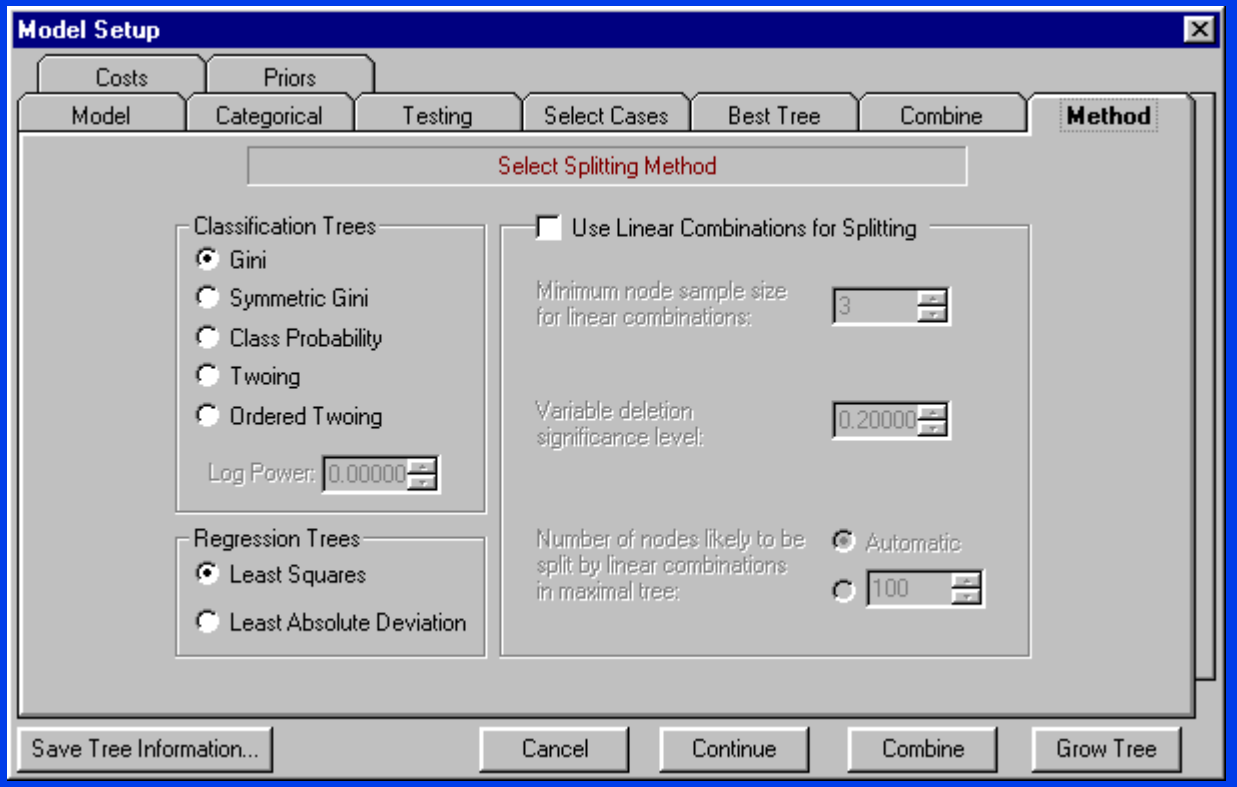


CART Always Grows Trees by Reducing Impurity

- **You get to choose how impurity is measured**
- **CART offers measures that have exhibited excellent performance characteristics**
- **Choice of impurity measure is a choice of splitting rule**

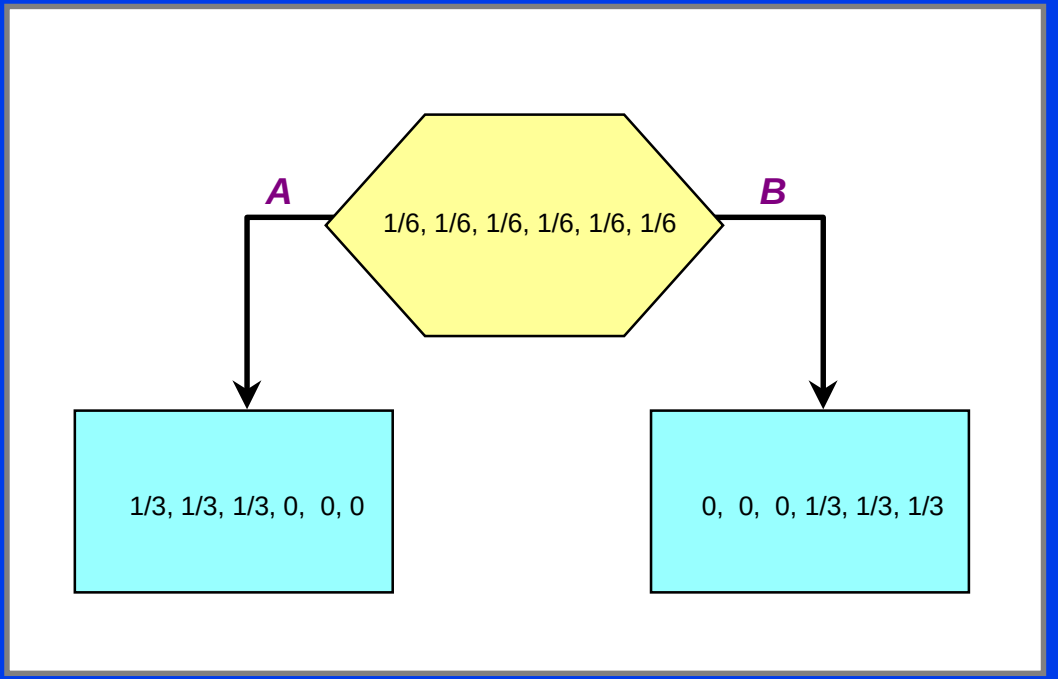
CART Splitting Rules for Binary and Multi-class Problems

- **Primary splitting rules for classification: Gini, Twoing, Power Modified Twoing**
- **Other rules available: Ordered Twoing, Symmetric Gini**



Example: Six Segments in a Market, all Equally Likely

- Here we have a very satisfactory split
- 3 segments go left in their entirety
- Remainder go right
- What has happened to impurity?



Gini Measure: $i(t)=1-\sum_i p_i^2$

- **Parent impurity is:** 5/6

  $i(t)= 1 - (1/6)^2 -(1/6)^2 - (1/6)^2 -(1/6)^2 - (1/6)^2 -(1/6)^2$

  $= 1- 6(1/36) = 5/6$

- **Left child node impurity=** 4/6

  $i(t) = 1 - (1/3)^2 - (1/3)^2 -(1/3)^2 - 0 - 0 - 0$

  $= 1 - 3(1/9) =2/3 =4/6$

- **Right child node has same impurity of** 4/6
- **Weighted average of the two is 4/6**

  $\Delta (t, s) = i(t) - pL\ i(tL) - pR\ i(tR)$
- **The improvement 5/6 - 4/6 = 1/6 = .16** =1/6
- **Note the numeric scale of improvements—often rather small numbers**

Twoing Criterion for Multiclass Problem

- Classes are numbered $\{1,2,...,J\}$
- At each node group the classes into two subsets
- e.g. In the 10 class gym example the two subsets identified by the twoing root node splitter were
  - $C1 = \{1,4,6,7,9,10\}$ $C2 = \{2,3,5,8\}$
- Then the best split for separating these two groups is found
- The process can be repeated for all possible groupings
  - best overall is the selected splitter
- Same as GINI for a binary dependent variable

The Twoing Measure

The twoing criteron function *Ø(s, t) is:*

$$\phi(s,t) = \frac{p_L p_R}{4}\left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2.$$

The best twoing split maximizes *Ø(s, t)*

C^*_1 is given by:

$$C^*_1 = \left\{j : p(j|t^*_L) \geq p(j|t^*_R)\right\},$$

C^*_1 is the set of classes with higher probabilities of going left

Twoing Interpreted

$$\frac{p_L p_R}{4}\left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

- Maximizes sum of differences between the fraction of a class going left & fraction going right.
- Note lead multiplying factor $p_L p_R$
- $p_L p_R$ is greatest when $p_L = p_R = 1/2$
- Sum is scaled downwards for very uneven splits
- So first find split which maximizes this sum of probability differences
  - Group C1 is defined as those classes more likely to go left
  - Group C2 is defined as those classes more likely to go right
  - CART authors think of these two groups as containing strategic information
- Exhibit class similarities

Center Cutting Exponent

- One variant of twoing criterion

$$\frac{(p_L p_R)^k}{4}\left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

- The $p_L p_R$ component is raised to power *k*
- Called *"center cutting exponent"*
- Used to emphasize or de-emphasize even splits
- Control with METHOD POWER = number
- Increases penalty on uneven splits
- METHOD TWOING, POWER=1
- Suggested values: 0, .5, 1, 2, and 10
- POWER=10 generates near median splits

Use Center Cutting Exponent for Difficult Trees

- **When trees are especially difficult to grow**
  - **e.g. Best tree reduces initial misclassification cost by less than 10%**
- **Center cutting exponent of .5 or 1 can improve final tree results**
  - **Often doubling improvement**
  - **Strictly an empirical finding**
- **Can set POWER for both Gini and Twoing**

GINI or Twoing: Which splitting criterion to use?

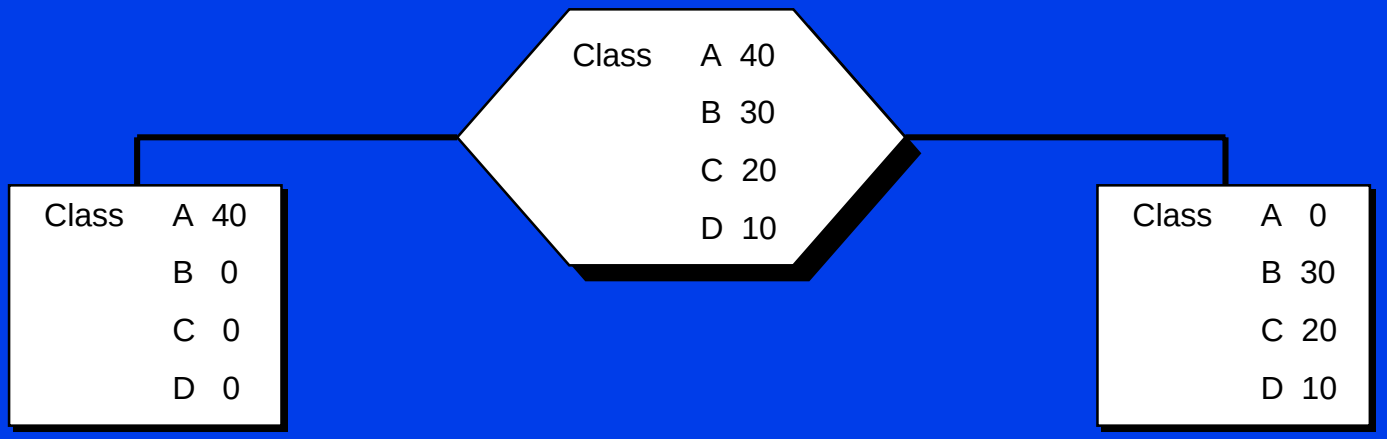
- **Monograph suggests Gini is usually better**
- **Will be problem dependent**
- **When end-cut splits (uneven sizes) need to be avoided use twoing**
  - **but can also just set POWER>0**
- **If the target variable has many levels, consider twoing**
- **Always experiment - try both**

Gini vs. Twoing in the multi-class problem

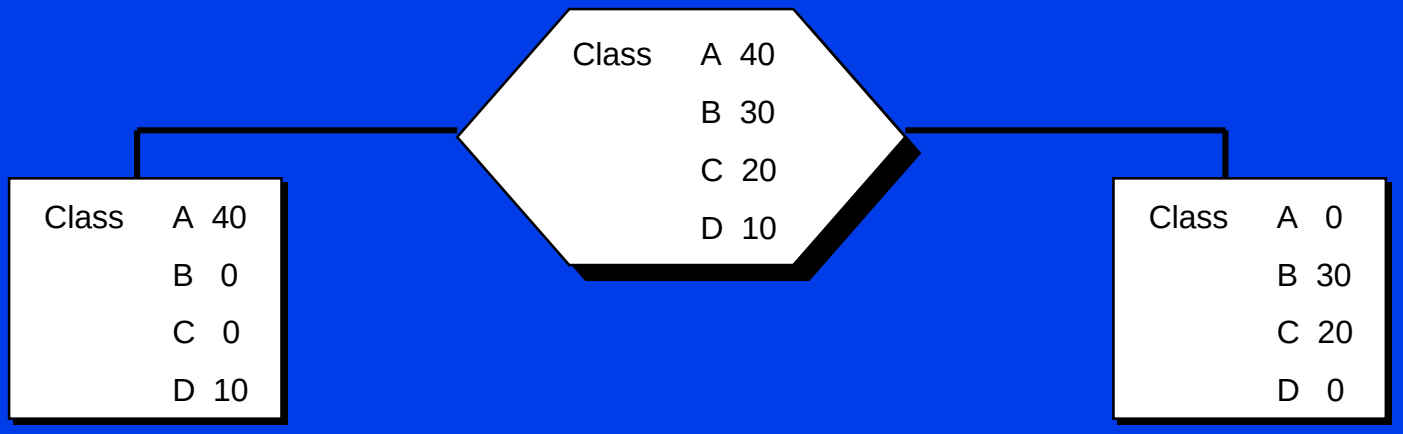
- **Gini always favors splits which separate the largest class from all the others**
- **Gini will isolate one class if it can**
- **Twoing tries to make children equally sized**

Gini vs. Twoing Example

Gini Best Split



Twoing Best Split



Prior Probabilities

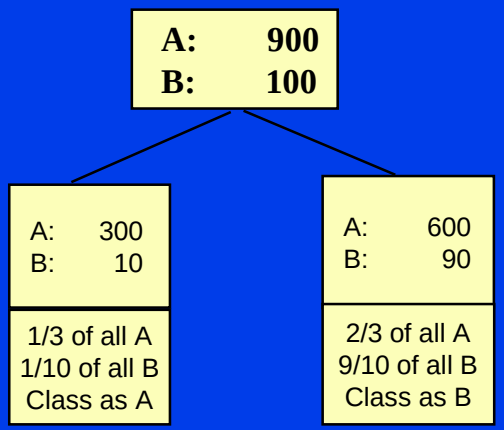
- An essential component of any CART analysis
- A prior probability distribution for the classes is needed to do proper splitting since prior probabilities are used to calculate the p_i in the Gini formula
- Example: Suppose data contains

  99% type A (nonresponders),

  1% type B (responders),
- CART might focus on not missing any class A objects
  - one "solution" classify all objects nonresponders
- Advantages: Only 1% error rate
- Very simple predictive rules, although not informative
- But realize this could be an optimal rule

Terminology: Re-weight Classes with "PRIOR PROBABILITIES"

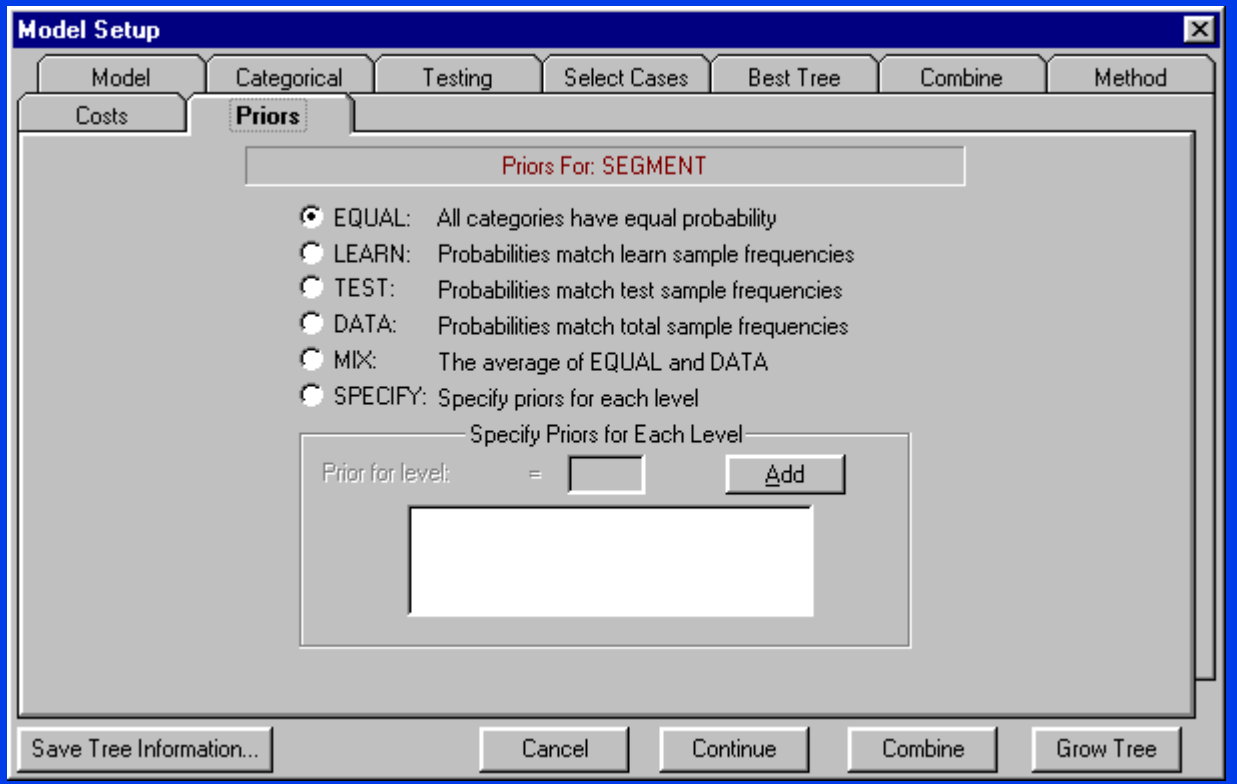
- Most common priors: PRIORS EQUAL
- This gives each class equal weight regardless of its frequency in the data
- Prevents CART from favoring more prevalent classes
- If we have 900 class A and 100 class B and equal priors



- With equal priors prevalence measured as percent of OWN class size
- Measurements relative to own class not entire learning set
- If PRIORS DATA both child nodes would be class A
- Equal priors puts both classes on an equal footing

Specifying Priors in GUI: Priors Model Setup Dialog

- **Priors settings: equal, learn, test, data, mix, user-specified**



Quick formula for Binary Dependent Variable (Version 1)

- **For equal priors class node is class 1 if**

$$\frac{N_1(t)}{N_2(t)} > \frac{N_1}{N_2}$$

N_i = **number of cases in class i at the root node**

$N_i(t)$ = **number of cases in class i at the node t**

- **Classify any node by "count ratio in node" relative to "count ratio in root"**
- **Classify by which level has greatest relative richness**

Quick formula for Binary Dependent Variable (Version 2)

- **When priors are not equal:** $$\frac{\pi(1)N1(t)}{\pi(2)N2(t)} > \frac{N1}{N2}$$

  where π_ι is the prior for that class
- **Since $\pi(1)/\pi(j)$ always appears as ratio just think in terms of boosting amount $Ni(t)$ by a factor, e.g. 9:1, 100:1, 2:1**
- **If priors are EQUAL, π terms cancel out**
- **If priors are DATA, then** $$\frac{\pi(1)}{\pi(2)} = \frac{N1}{N2}$$

  which simplifies formula to plurality rule (simple counting)

  classify as class 1 if $$N1(t) > N2(t)$$
- **If priors are neither EQUAL nor DATA then formula will provide boost to up-weighted classes**

Recalling our Example 900 Class A and 100 Class B in the Root Node

A: 900
B: 100

A: 300
B: 10

$\frac{300}{10} > \frac{900}{100}$ **so class as A**

A: 600
B: 90

$\frac{90}{600} > \frac{100}{900}$ **so class as B**

In general the tests would be weighted by the appropriate priors

$$\frac{\pi(B)\ 90}{\pi(A)\ 600} > \frac{100}{900} \qquad \frac{\pi(A)\ 300}{\pi(B)\ 10} > \frac{900}{100}$$

tests can be written as $\frac{A}{B}$ **or** $\frac{B}{A}$ **as convenient**

Priors for the Multi-class Problem (costs not specified)

- Math is identical to binary dependent variable problem
- For PRIORS EQUAL root node will be classified as CLASS 1
- Classification at start is arbitrary for EQUAL PRIORS
  - Only one class will be correctly classified in root
  - All classes equally important
  - So choose any class assignment to start
- Example: 4 segments
  - Root node will get 1/4 correct , 3/4 wrong
  - Below root, classification will be accomplished by weighting by priors

Root Node Classification for Multi-class 4-Level Problem

- Dependent variable: 1, 2, 3, 4
- Root is classified as first level of DPV (1)
- Initial cost = .75 (75% of sample misclassified in root

Terminal Tree	Nodes	Cross-Validated Relative Cost	Resubstitution Relative Cost	Complexity Parameter
1	15	0.959 +/- 0.063	0.414	0.000
2	13	0.973 +/- 0.062	0.433	0.007
3	12	0.969 +/- 0.062	0.446	0.010
4	10	0.950 +/- 0.063	0.474	0.011
5	8	0.950 +/- 0.063	0.525	0.019
6**	7	0.893 +/- 0.066	0.553	0.021
7	5	0.934 +/- 0.064	0.620	0.025
8	4	0.896 +/- 0.066	0.659	0.030
9	3	0.919 +/- 0.064	0.757	0.073
10	1	1.000 +/- 0.000	1.000	0.091

Initial misclassification cost = 0.750

Initial class assignment = 1

Equal Priors Tend to Equalize Misclassification Rates

- **EQUAL PRIORS is the default setting for all CART analyses**
- **Gives each class an equal chance at being correctly classified**

- **Example of PRIORS**

– **PRIORS EQUAL**	**Default - should use this to start with**
– **PRIORS DATA**	**Empirical frequency, whatever is found in data**
– **PRIORS MIX**	**Average of EQUAL and DATA -- shades toward empirical**
– **PRIORS = n1,n2**	**Explicit priors**
– **PRIORS = 2,1**	**Makes class 1 prior twice class 2 prior**
– **PRIORS = .67, .33**	**Same as PRIORS 2,1**

Priors Incorporated Into Splitting Criterion

Gini = $1 - \sum p_i^2$

- $p_i(t)$ = **Proportion of class *i* in node t**

$$p(t) = \frac{\pi(i)\frac{Ni(t)}{Ni}}{\sum \pi(j)\frac{Nj(t)}{Nj}}$$

- **If priors DATA then**

$$\pi(i) = \frac{Ni}{N}$$

- **Proportions of class *i* in node t with data priors**

$$p(t) = \frac{N_i(t)}{\sum N_i(t)} = \frac{N_i(t)}{N(t)}$$

- **Otherwise proportions are always calculated as weighted shares using priors adjusted p_i**

Using Priors to Reflect Importance

- Putting a larger prior on a class will tend to decrease its misclassification rate
- Hence can use priors to shape an analysis
- Example:

MISCLASSIFICATION RATES BY CLASS FOR ALTERNATIVE PRIORS: Number Misclassified (N) and Probability of Misclassification (%)

	Sample	Relative Prior on Class 2 and N misclassified							
	Size	1.0		1.2		1.4		1.6	
Class		N	%	N	%	N	%	N	%
1	**12**	3	25.0	5	41.7	5	41.7	6	50.0
2	**49**	47	95.9	30	61.2	26	53.1	15	30.6
3	**139**	46	33.1	74	53.2	77	55.4	94	67.6
Total	**200**	**96**		**109**		**108**		**115**	

Note: As priors change, absolute # misclassified also changes

Shortcomings of Using Priors to Reflect Costs

- Blunt instrument-emphasizes getting entire classes right vs. wrong
  - All misclassifications of a given class equally bad
- Consider segmentation scheme with 5 classes of customers
  - High Volume Buyers
  - Medium Volume Buyers
  - Low Volume Buyers
  - Not Likely to Buy
  - Would Never Buy
- May not be so bad to occasionally mix up HIGH and MEDIUM VOLUME BUYERS or to mix WOULD NEVER BUY and NOT LIKELY TO BUY
- Very bad to mix WOULD NEVER BUY with HIGH VOLUME BUYERS
- One solution — combine segments
  - Loses important detail
- Another solution in CART: variable misclassification costs

Costs of Misclassification

- For most classification schemes...
  - "AN ERROR IS AN ERROR IS AN ERROR"
- No distinction made between different types of error
  - All count equally -- are equally bad
- In practical applications different errors are quite different in importance
  - In medical application: classifying a breast tumor as malignant or benign
  - Misclassify malignant as benign — possible death of patient
  - Misclassify benign as malignant — unnecessary open breast biopsy
- Want both classes classified correctly — either mistake is serious
- May want to up-weight the misclassification of malignant tumors error

Market Segmentation Examples

- **Martin & Wright (1974): segmenting population into buyers vs. non-buyers**
- **Purpose: allocate marketing effort to buyers**
  - **Marketing to a non-buyer loses the company $1**
  - **Not marketing to a buyer forgoes $3 of profit**
- **Want to include this in the analysis**
- **Consider group of customers to be classified on demographics**
  - **Actually contain 40% buyers 60% non-buyers**
  - **Would want to classify this group as ALL buyers!**
  - **Even though we would be misclassifying MOST people in the group**
  - **.40 · $3 + .60 · (-$1) = $1.20 - $0.60 = $0.60 per person profit**

Explicit Cost Matrix

Default matrix is:

		Classified as *non-buyer*	Classified as *buyer*
Truth	*non-buyer*	0	1
	buyer	1	0

Want to specify explicit costs as:

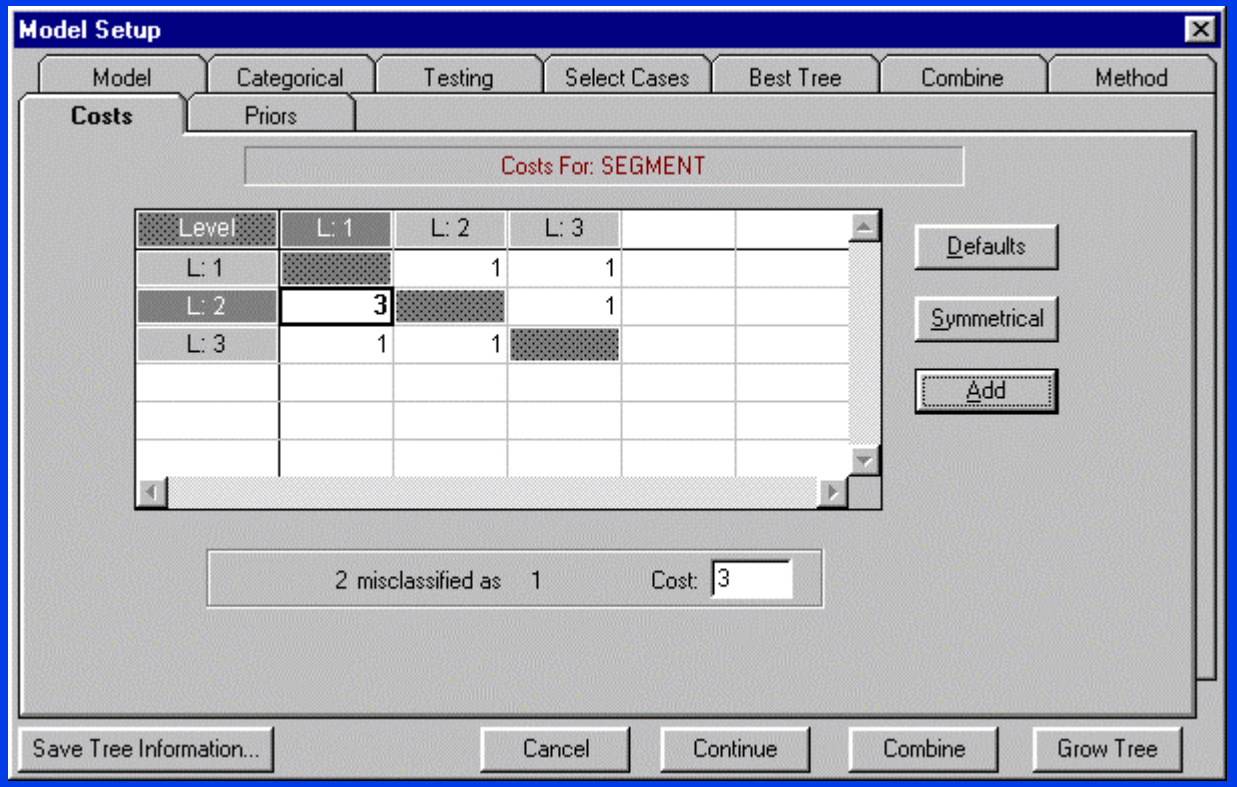
		Classified as *non-buyer*	Classified as *buyer*
Truth	*non-buyer*	0	1
	buyer	3	0

Misclassifying buyers is three times as bad as misclassifying non-buyers

Specifying Costs

- **EXPLICIT COST COMMANDS**
  - **MISCLASS COST=3 CLASS 2 AS 1** **Implements scheme on last slide**

- **Specifying in GUI:**



Cost Matrix

- **Can specify ANY cost matrix by specifying each cell separately**
- **Short hand forms of command:**
  - **MISCLASS COST=number CLASS group AS class_number**
  - **MISCLASS COST=number CLASS class_number AS group**
- **Only one side of the AS keyword can contain a shorthand group notation**
- **So some misclassification costs may require several commands to implement**

Misclassification Costs vs. Priors

- Can increase the *"importance"* of a class by increasing prior
- Prior weights equivalent to raising all costs for a given class
- Misclassification costs can vary by specific error
  - Misclassify a class B as a class C high cost
  - Misclassify a class B as a class A low cost
- This level of control not available by manipulating priors

Sample cost matrix

	A	B	C
A	•	3	1
B	1	•	4
C	2	2	•

Quick Formula for Binary Dependent Variable

- C(2|1) = Cost of classifying as 2 when it is really a class 1
- Classify a node as class 1 rather than class 2 if:

$$\frac{C(2|1)\pi(1)N_1(t)}{C(1|2)\pi(2)N_2(t)} > \frac{N_1}{N_2}$$

- Note the adjustments are (a) reweight by priors and (b) reweight by costs
- Generalize by comparing any two classes i, j with:

$$\frac{C(j|i)\pi(i)N_i(t)}{C(i|j)\pi(j)N_j(t)} > \frac{N_i}{N_j}$$

- node classified as class i if the inequality held for all j $(j \neq i)$

Example: Classifying Segments Without Costs

Table Without Costs (MDL5A.DAT)

ACTUAL CLASS	PREDICTED CLASS 1	2	3	4	5	6	ACTUAL TOTAL
1	45.00	1.00	1.00	3.00	0.00	1.00	***51.00***
2	4.00	32.00	1.00	7.00	8.00	10.00	***62.00***
3	9.00	20.00	3.00	10.00	6.00	6.00	***54.00***
4	4.00	11.00	1.00	2.00	2.00	2.00	***22.00***
5	6.00	3.00	0.00	0.00	29.00	1.00	***39.00***
6	2.00	1.00	0.00	2.00	3.00	15.00	***23.00***
PRED. TOT.	70.00	68.00	6.00	24.00	48.00	35.00	***251.00***
CORRECT	0.88	0.52	0.06	0.09	0.74	0.65	
SUCCESS IND.	0.68	0.27	-0.16	0.00	0.59	0.56	
TOT. CORRECT	0.50						

Example: Classifying Segments With Costs

Table With Costs (MDL5EO.DAT)

ACTUAL CLASS	PREDICTED CLASS 1	2	3	4	5	6	ACTUAL TOTAL
1	45.00	4.00	2.00	0.00	0.00	0.00	***51.00***
2	3.00	33.00	10.00	3.00	6.00	7.00	***62.00***
3	0.00	15.00	27.00	3.00	5.00	4.00	***54.00***
4	4.00	10.00	3.00	3.00	2.00	0.00	***22.00***
5	7.00	12.00	0.00	0.00	19.00	1.00	***39.00***
6	1.00	9.00	4.00	1.00	1.00	7.00	***23.00***
PRED. TOT.	60.00	83.00	46.00	10.00	33.00	19.00	***251.00***
CORRECT	0.88	0.53	0.50	0.14	0.49	0.30	
SUCCESS IND.	0.68	0.29	0.29	0.05	0.33	0.21	
TOT. CORRECT	0.53						

Comparison: Classification With and Without Costs

	PREDICTED CLASS						
	1	2	3	4	5	6	
% CORRECT	0.88	0.52	0.06	0.09	0.74	0.65	*Without Costs*
% CORRECT	0.88	0.53	0.50	0.14	0.49	0.30	*With Costs*

Then Costs Incorporated Into GINI Splitting Rule

$$COSTS\text{-}GINI = \sum_i \sum_j C(i \mid j)\, p(i \mid t)\, p(j \mid t)$$

- **Formula simplifies only when UNIT costs are used**

$$= 1 - \sum_i p(i \mid t)^2$$

- **This is an entirely new splitting rule**

Note:

$$C(i|j)\,p(i|t)\,p(j|t) + C(j|i)\,p(j|t)\,p(i|t) =$$

$$\left[C(i|j) + C(j|i)\right] p(i|t)\,p(j|t)$$

- **Rule uses symmetrized costs**

Handling Missing Values in Tree Growing

- **Allow cases with missing split variable to follow majority**
- **Assign cases with missing split variable to go left or right probabilistically, using PL and PR as probabilities**
- **Allow missing to be a value of variable**

Missing Values on Primary Splitter

- One option: send all cases with missings with the majority of that node
- Some machine learning programs use this procedure
- CHAID treats missing as a categorical value; all missings go the same way
- CART uses a more refined method —a surrogate is used as a stand in for a missing primary field
- Consider variable like INCOME — could often be missing
- Other variables like Father's or Mother's Education or Mean Income of Occup. might work as good surrogates
- Using surrogate means that missing on primary not all treated same way
- Whether go left or right depends on surrogate value

Surrogates — Mimicking Alternatives to Primary Splitters

- A primary splitter is the best splitter of a node
- A surrogate is a splitter that splits in a fashion similar to the primary
- Surrogate — A Variable with possibly equivalent information
- Why Useful
  - Reveals structure of the information in the variables
  - If the primary is expensive or difficult to gather and the surrogate is not
    - Then consider using the surrogate instead
    - Loss in predictive accuracy might be slight
  - If primary splitter is MISSING then CART will use a surrogate

Association

- How well can another variable mimic the primary splitter?
- Predictive association between primary and surrogate splitter
- Consider another variable and allow reverse splits
  - A reverse split sends cases to the RIGHT if the condition is met
  - Standard split (primary) always sends cases to the LEFT
- Consider a default splitter to mimic primary
  - Send ALL cases to the node that the primary splitter favors
  - Default mismatch rate is $\min(P_L, P_R)$
    - e.g. if primary sends 400 cases left and 100 cases right with PRIORS DATA
    - $P_L = .8$ $P_R = .2$
    - If default sends all cases left then 400 cases are matched and 100 mismatched
    - Default mismatch rate is .2
    - A credible surrogate must yield a mismatch rate LESS THAN the default

Evaluating Surrogates

- **Matches are evaluated on a case by case basis**
- **If surrogate matches on 85% of cases then error rate is 15%**

$$\text{Association} = \frac{\text{default mismatch-surrogate mismatch}}{\text{default mismatch}}$$

- **In this example: Association= (.2 - .15)/.2 = .05/.2 =.25**
- **Note that surrogate is quite good by *a priori* standards yet measure seems low**
- **Also note that association could be negative and will be for most variables**
- **If match is perfect then surrogate mismatch=0 and association=1**

Competitors vs. Surrogates

		Class A 100 Class B 100 Class C 100	
		Left	Right
Primary Split	Class A	90	10
	Class B	80	20
	Class C	15	85
Competitor Split	Class A	80	20
	Class B	25	75
	Class C	14	86
Surrogate Split	Class A	78	22
	Class B	74	26
	Class C	21	79

Classic CART Output: Node 1
Gym Example
Competitors & Surrogates

	Surrogate		Split	Assoc.	Improve.
1	PERSTRN	s	0.500000	0.914703	0.127490
2	FIT	s	6.353424	0.523818	0.110693
3	NFAMMEM	s	4.500000	0.206078	0.030952
4	TANNING	s	2.500000	0.029640	0.007039
5	NSUPPS	s	30.500000	0.029236	0.003991
6	CLASSES	s	1.500000	0.011499	0.006856

	Competitor	Split	Improve.
1	PERSTRN	0.500000	0.127490
2	FIT	3.107304	0.111190
3	ANYRAQT	0.500000	0.102227
4	PLRQTPCT	0.031280	0.090114
5	TPLRCT	0.500000	0.090085
6	ONRCT	0.500000	0.077620
7	NFAMMEM	2.500000	0.055136
8	CLASSES	0.500000	0.045368
9	IPAKPRIC	7.632159	0.033419
10	OFFAER	0.500000	0.022993

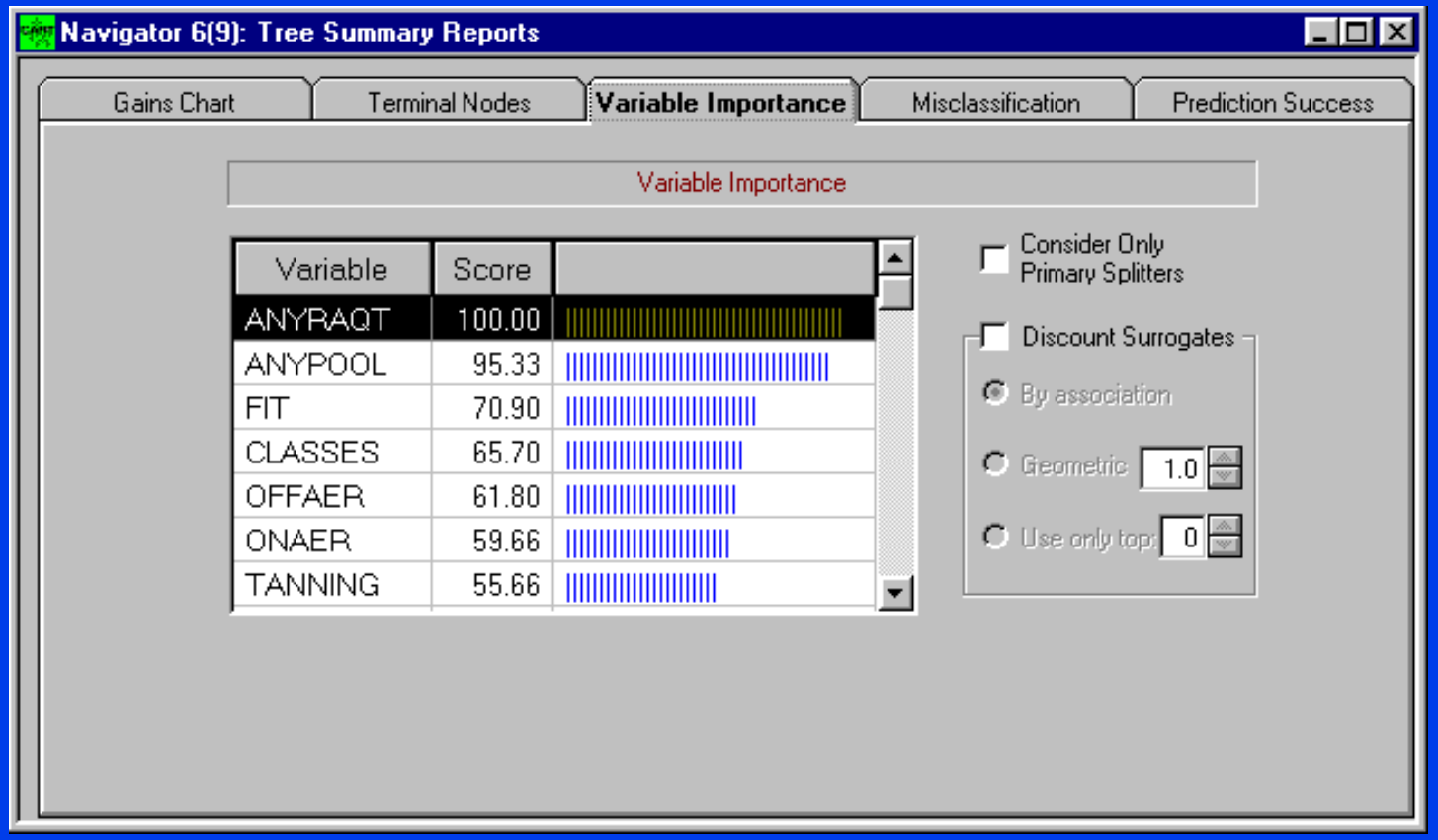
Classification Is Accomplished by Entire Tree Not Just One Node

- **Even if a case goes wrong way on a node (say surrogate is imperfect) it is not necessarily a problem**
- **Case may be get correctly classified on next node or further down tree**
- **CART trees will frequently contain self-correcting information**

Surrogates — How Many Can You Get

- Depends on the data — you might not get any!
- Often will see fewer than the top 5 that would be printed by default
- A splitter qualifies as a surrogate ONLY if association value >0
- Note surrogates are ranked in order of association
- A relatively weak surrogate might have better IMPROVEMENT than best surrogate
- A variable can be both a good competitor and a good surrogate
  - Competitor split values might be different than surrogate split value

GUI CART Output: Variable Importance



Variable Importance

- How can we assess the relative importance of the variables in our data
- Issue is a measure of the splitting potential of all variables
- CART includes a general measure with very interesting characteristics
  - The most important variables might not ever be used to split a node!
  - Importance is related to both potential and actual splitting behavior
  - If one variable say EDUCATION is masked by another say INCOME then...
    - Both variables have SIMILAR splitting information
    - In any one node only one variable will be best but the other may be a close 2nd best
  - If INCOME appears just once as primary splitter and EDUCATION is never a primary splitter
    - But EDUCATION appears as a strong surrogate in several different nodes
    - Then EDUCATION may have more potential splitting power
  - Means that if you had to live with just one of the variables
    - EDUCATION would most likely to be the better performer

Variable Importance Measured by Impurity Improvement

- **Only look at primary splitters and SURROGATES *not* COMPETITORS**
- **Reason: competitors are trying to make a particular split**
  - **If prevented from making split at one node will try again at next node**
  - **Next node attempt could be SAME split conceptually leading to double count**
- **Focus is on improvement among surrogates; non-surrogates not counted**
- **Hence IMPROVEMENT is a measure relative to a given tree structure**
- **Changing the tree structure could yield different importance measures**

Variable Importance Caution

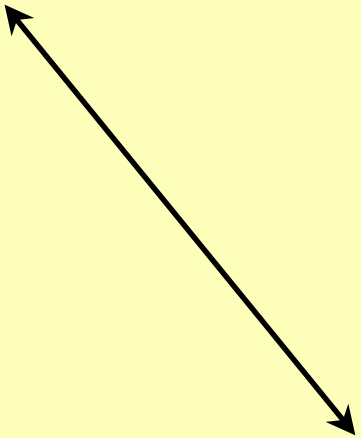
- **Importance is a function of the OVERALL tree including deepest nodes**
- **Suppose you grow a large exploratory tree — review importances**
- **Then find an optimal tree via test set or CV yielding smaller tree**
- **Optimal tree SAME as exploratory tree in the top nodes**
- **YET importances might be quite different.**
- **WHY? Because larger tree uses more nodes to compute the importance**
- **When comparing results be sure to compare similar or same sized trees**

Variable Importance & Number of Surrogates Used

- **Importance determined by number of surrogates tracked**
- **Tables below are derived from SAME tree**
- **WAVE data set example**

Allowing up to 5 Surrogates

V07	100.000
V15	93.804
V08	78.773
V11	68.529
V14	66.770
V06	65.695
V16	61.355
V09	37.319
V12	32.853
V05	32.767



Allowing Only 1 Surrogate

V15	100.000
V11	73.055
V14	71.180
V09	32.635
V08	28.914
V10	18.787
V12	12.268
V17	5.699
V04	0.000
V07	0.000

Modifying Importance--Down Weighting Weak Surrogates

- Each surrogate gets full credit for any improvement it might yield on a split
- Credit accrues in proportion to improvement irrespective of degree of association
- May want to down weight surrogates
- BOPTIONS IMPROVEMENT=q where $q < 1$ down weights best surrogate
  - Credits surrogate with proportion q of the improvement
  - Credits second surrogate with proportion q^2 of the improvement
  - Credits third surrogate with proportion q^3 of the improvement
- CART also allows weight to be proportional to association

Competitor Splits

- CART searches for the BEST splitter at every node
- To accomplish this it first finds the best split for a specific variable
- Then it repeats this search over all other variables
- Results in a split quality measure for EVERY variable
- The variable with the highest score is the PRIMARY SPLITTER
- The other variables are the COMPETITORS
- Can see as many COMPETITORS as you want — they have all been computed
- BOPTIONS COMPETITORS=5 is the default for number to PRINT
- To see scores for every variable try BOPTIONS COMPETITORS=250

Printing Many Competitors

Node 1 was split on variable SI

Competitor		Split	Improve.
1	G1	8.903	0.040
2	SU	19.081	0.040
3	AG6	-13.758	0.037
4	AG5	-35.594	0.035
5	SBHI7	25.871	0.035
6	AH	19.161	0.034
7	AI	16.180	0.033
8	SBHI8	25.980	0.031
9	HISB	25.614	0.031
10	N5	0.496	0.031
11	G12	6.583	0.029
12	SO	15.605	0.029
13	SBHI4	23.097	0.029
14	SBHI5	23.645	0.029
15	G7	50.006	0.027
16	MIRA8	6.467	0.027
17	ABHI8	22.103	0.026
18	G4	13.736	0.026
19	SBLO5	12.597	0.025
20	AC	18.755	0.025
21	N1	0.033	0.025
22	BBDA7	1.674	0.025
23	SBHI6	22.196	0.025
24	AL	16.136	0.025
25	SC	39.654	0.025
26	ABHI5	18.206	0.024

Linear Combination Splits

- Decision tree methods are notoriously awkward in tracking linear structure
- If functional relationship is $y = X\beta + \text{error}$, CART will generate a sequence of splits of the form
  - Is X < c1
  - Is X < c2
  - Is X < c3 etc.
- Awkward, crude, and not always easy to recognize when multiple variables are involved
- Can instead allow CART to search for linear combinations of predictors
  - So decision rule could be: If .85*X1 + .52*X2 < -5 then go left
- If the best linear combination found beats all standard splits it becomes the primary splitter
- The linear combination will be reported as a splitter only, never as a surrogate or a competitor

Linear Combination Caveats

- Linear combination involves only numeric not categorical variables
- Scale not unique; coefficients normalized to sum of squared coefficients =1
- Linear combinations searched using a stepwise algorithm — global best may not be found
- Backward deletion used; start with all numerics and back down
- May want to limit search to larger nodes
- LINEAR N=500 prevents search when number of cases in node < 500
- Benefit to favoring small number of variables in combinations
  - Easier to interpret and assess
  - Does combination make any empirical sense

Linear Combination Caveats, Cont'd.

- **LINEAR N=500 DELETE=.40 permits deletion if variable γ-value <.4**
  - **Would generate linear combinations with few variables**
- **Linear combinations not invariant to variable transformations**
  - **Quality of result influenced by log or square root transforms**
- **Default DELETE = .20**
  - **Linear combination — get best improvement Δ I**
  - **Drop one variable and re-optimize coefficients**
  - **Repeat for each variable**
  - **Rank loss in Δ I**

Searching for Ratios

- **Standard CART invariant to monotone transforms of variables**
- **Linear combination splits and quality of splits not invariant to transforms**
- **Might want to create log transforms of selected predictor variables and search linear combinations**
- **Allows CART to split on $\alpha_1 1nx_1 - \alpha_2 1nx_2$**
- **Would suggest using the x_1/x_2 ratio**

Restricting Linear Combination Split to Root Node

- **LINEAR N = sample-size, e.g.**
  - **LINEAR N = 12347**

    when the learn sample is 12347 will prevent linear combination splits below the root

- **LINEAR N = 12347 / EXHAUSTIVE**
  - **lengthy search for global maximum**
  - **not efficient but can find superior splits to default method**
  - **could take 100 times longer to compute; use judiciously**

Linear Combinations Can Give Frustratingly Poor CV Results

- **The more variables appearing in a linear combination the less likely it is to do well in cross validation**
- **Cross-validation results could be quite pessimistic**
- **If a linear combination you like is found via exploratory tree, you may want to CREATE that variable**
- **CV results could be much improved**
- **Do this only if you have independent reason to trust new variable**

Key Components of Tree Structured Data Analysis

- **Tree growing**
  - **Splitting Rules**
  - **Stopping Criteria**
- **Tree Pruning**
- **Optimal Tree Selection**

We Now Know How to Grow a Tree

- **Binary recursive partitioning**
  - **Each node can yield two child nodes**
- **Can re-use variables as often as needed**
- **Can specify priors to weight classes in any way we want**
- **Can specify costs to shape development**

Problems With Early Tree Growing Methods

- **Early tree methods often gave "dishonest" results**
- **Would not hold up on new data**
- **Key Issues:**
  - **The more you split the better you think you are doing!**
    - **Like R-squared in regression fit always improves with more variables (splits)**
    - **Error measures based on learning data are unrealistically low**
  - **CHAID and brethren use stopping rules -- will always have problems going too far along some branches and not going far enough on others**

A Core CART Innovation
Don't Stop Tree Growth

- You will never know when to stop
- Key to finding the right tree is in the pruning
  - CART introduced cost-complexity pruning
  - Develops a nested sequence of candidate trees for evaluation
  - Pruning will often yield highly asymmetric (unbalanced) trees
- Testing critical to finding honest trees
  - External test sets when large data volumes available
  - Internal cross-validation when data not so plentiful
- No way to tell which tree is believable without these tests
- Trees without test and proper pruning are like regressions without standard errors

Tree Growing — Stopping Criteria

- CART differs from earlier methods such as AID or CHAID on stopping
- STOPPING not an essential component of CART
- IF time and data were available CART would NOT stop!
- Grow tree until further growth is not possible
  - Terminal nodes only have one case
  - Terminal nodes with more than one case are identical on predictor variables
- Result is called the MAXIMAL tree
- In practice certain limits can be imposed
  - Do not attempt to split smaller nodes
    - 100 cases in binary dependent variable market research problems

Tree Pruning

- Take some large tree such as the maximal tree
- Tree may be radically overfit
  - Tracks all the idiosyncrasies of THIS data set
  - Tracks patterns that may not be found in other data sets
  - Analogous to a regression with very large number of variables
- PRUNE branches from the large tree
- CHALLENGE is HOW TO PRUNE
  - Which branch to cut first?
  - What sequence — There are hundreds, if not thousands of pruning sequences
- If you have 200 terminal nodes, then…
  - 200 ways to prune away 1st. deleted node
  - 199 ways to prune away 2nd. deleted node, etc.

Cost-Complexity Pruning

- **Begin with a large enough tree — one that is larger than the truth**
  - **With no processing constraints grow until 1 case in each terminal node**
- **PRUNING AT A NODE means making that node terminal by deleting its descendants**
- **Idea**
  - **Suppose we have a 100 node tree and we want to prune to a 90 node tree**
  - **Should prune to 90 node sub-tree that has the smallest error**
  - **Similarly for all other sub-trees with 89, 88,...5, 4, 3, 2 nodes**
  - **Prune to sub-tree with smallest misclassification cost for that tree size**

Cost-Complexity Pruning

$R_\alpha(T)$	= cost - complexity measure of the tree T
	= R(T) + α * | T |
	= misclassification cost + α * (# of terminal nodes in T)
	= misclassification cost + α * (complexity of T)
α	= penalty placed on complexity

- Select ANY non-terminal node in a tree — it has an error rate $R(t)$
- Also consider the sub-tree below it
- It has an error rate $R(T_t)$
  - The sum of the error rates of the terminal nodes in T_t

Resubstitution Cost vs. True Cost

- Compare error rates measured by
  - learn data
  - large test set
- Learn R(T) always decreases as tree grows
- Test R(T) first declines then increases
- Much previous disenchantment with tree methods due to reliance on learn R(T)
- Can lead to disasters when applied to new data

No. Terminal Nodes	R(T)	$R^{ts}(T)$
71	.00	.42
63	.00	.40
58	.03	.39
40	.10	.32
34	.12	.32
19	.20	.31
*10	.29	.30
9	.32	.34
7	.41	.47
6	.46	.54
5	.53	.61
2	.75	.82
1	.86	.91

Why look at resubstitution error rates (or cost) at all?

- First, provides a rough guide of how you are doing
  - Truth will typically be WORSE than resubstitution measure
  - If tree performing poorly on resubstitution error may not want to pursue further
  - Resubstitution error more accurate for smaller trees
    - So better guide for small nodes
    - Should be poor guide for many nodes
  - Resubstitution rate useful for comparing SIMILAR SIZED trees
  - Even though true error rate is not measured correctly relative ranking of different trees probably is correct
    - different trees because of different splitting rules
    - different trees because of different variables allowed in search

How to Prune: The Macro View

- **Let's examine the tree sequence below:**

Dependent variable: CLUSTER

Terminal Tree	Nodes	Test Set Relative Cost	Resubstitution Relative Cost	Complexity Parameter
1**	11	0.168493 +/- 0.005304	0.195980	0.012555
2	10	0.208200 +/- 0.005482	0.214243	0.014769
3	9	0.249918 +/- 0.004828	0.245512	0.025262
4	7	0.340175 +/- 0.004018	0.323120	0.031346
5	6	0.413026 +/- 0.003861	0.388242	0.052599
6	5	0.474169 +/- 0.003117	0.461266	0.058981
7	4	0.558584 +/- 0.003051	0.543542	0.066451
8	3	0.686229 +/- 0.002973	0.671612	0.103433
9	2	0.835007 +/- 0.001592	0.833190	0.130491
10	1	1.000000 +/- 0.000069	1.000000	0.134717

Initial misclassification cost = 0.807546
Initial class assignment = 3

Pruning

- **Start with complexity penalty set to 0**
- **Best resubstitution cost tree is maximal tree (biggest tree) has smallest error**
- **Start increasing complexity penalty**
- **At some point penalty is big enough to warrant choosing a smaller tree since penalty on nodes exceeds gain in accuracy**
- **Example: Pruning our CLUSTER tree from 6 to 5 terminal nodes**

Initial misclassification cost = 0.807546

- Complexity needed to prune
- Absolute Cost = Relative Cost * Initial cost

Tree	Terminal Nodes	Resubstitution Cost - Complexity Measure ABSOLUTE Cost	ABSOLUTE Cost	+Nodes*		
5	6	0.388242 *0.807546	0.388242 *0.807546	+	6	* α
6	5	0.461266 *0.807546	0.461266 *0.807546	+	5	* α
Tree 15-Tree 14			0.073024 *0.807546	-	α	= 0, when
			.058981	=	α	= complexity penalty

Pruning — The Micro View

Node 6 was split on variable CLASSES
A case goes left if variable CLASSES <= 0.500000
Improvement = 0.060257 C. T. = 0.058971

- **Complexity Threshold = penalty large enough to prune child nodes away**
- **Cost decrease=(.668535-p_L*.581630-p_R*.102462)*$p(t)$**
- **p(t)=prob(case enters non-terminal node 6**

Node	Cases	Class	Cost
6	13161	9	0.668535
7	10204	9	0.581630
-9	2957	2	0.102462

Order of Pruning

- **Prune away *"weakest link"* — the nodes that add least to overall accuracy**
- **If several nodes have same complexity threshold they all prune away simultaneously**
- **Hence more than two terminal nodes could be cut off in one pruning**
- **Often happens in the larger trees; less likely as tree gets smaller**

Terminal Node Information Report

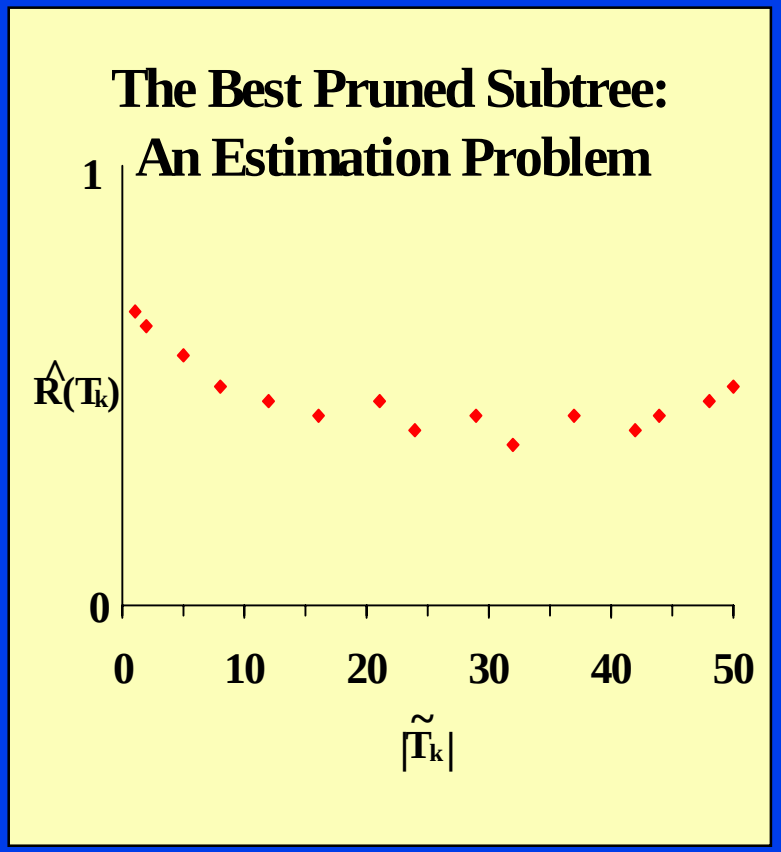
- **Lists complexity threshold for each terminal node**
- **Tells which nodes would be dropped next, and order of pruning**
- **The higher the complexity threshold (C.T.) the more valuable the node**

Terminal Node Report for Terminal Node 9, Child of Node 6

```
=========================
TERMINAL NODE INFORMATION
=========================

(Test Set)
                                                |--LEARNING SET--||----TEST SET----|
Node    N        Prob Class      Cost Class     N        Prob     N        Prob
-------------------------------------------------------------------------------
  9  2957    0.065731   2    0.102462     1     1    0.000359     0    0.000000
  (   210    0.057633        0.141332)    2  2662    0.897537   170    0.858668
         Parent C.T. =      0.058971     3   258    0.089737    36    0.129682
                                         4     7    0.002466     3    0.009704
                                         5    25    0.008353     0    0.000000
                                         6     0    0.000000     0    0.000000
                                         7     3    0.001165     0    0.000000
                                         8     0    0.000000     0    0.000000
                                         9     1    0.000384     1    0.001946
                                        10     0    0.000000     0    0.000000
```

The Optimal Tree

- Within a single CART run which tree is best?
- The process of pruning the maximal tree can yield many sub-trees
- The test data set or cross- validation measures the error rate of each tree
- Current wisdom — select the tree with smallest error rate
- Only drawback — minimum may not be precisely estimated
- Running CV with a new random seed could yield a different sized tree
- Typical error rate as a function of tree size has flat region
- Minimum could be anywhere in this region



One SE Rule -- The One Standard Error Rule

- Original CART monograph recommends NOT choosing minimum error tree because of possible instability of results from run to run
- Instead suggest SMALLEST TREE within 1 SE of the minimum error tree
- 1 SE tree is smaller than minimum error tree
- Lies one standard error away from minimum error tree
- Tends to provide very stable results from run to run
- Is possibly as accurate as minimum cost tree yet simpler
  - BOPTIONS SERULE=0
  - BOPTIONS SERULE=1
    these are all options for the analyst
  - BOPTIONS SERULE=.5
- Current learning — one SERULE is generally too conservative, will prune up to too small a tree

In what sense is the optimal tree best?

- Tree has lowest or near lowest cost as determined by a test procedure
- Tree should exhibit very similar accuracy when applied to new data
- This is key to the CART selection procedure — applicability to external data
- BUT Tree is NOT necessarily unique — other tree structures may be as good
- Other tree structures might be found by
  - Using other learning data
  - Using other growing rules
- Running CV with a different random number seed
- Some variability of results needs to be expected
- Overall story should be similar for good size data sets

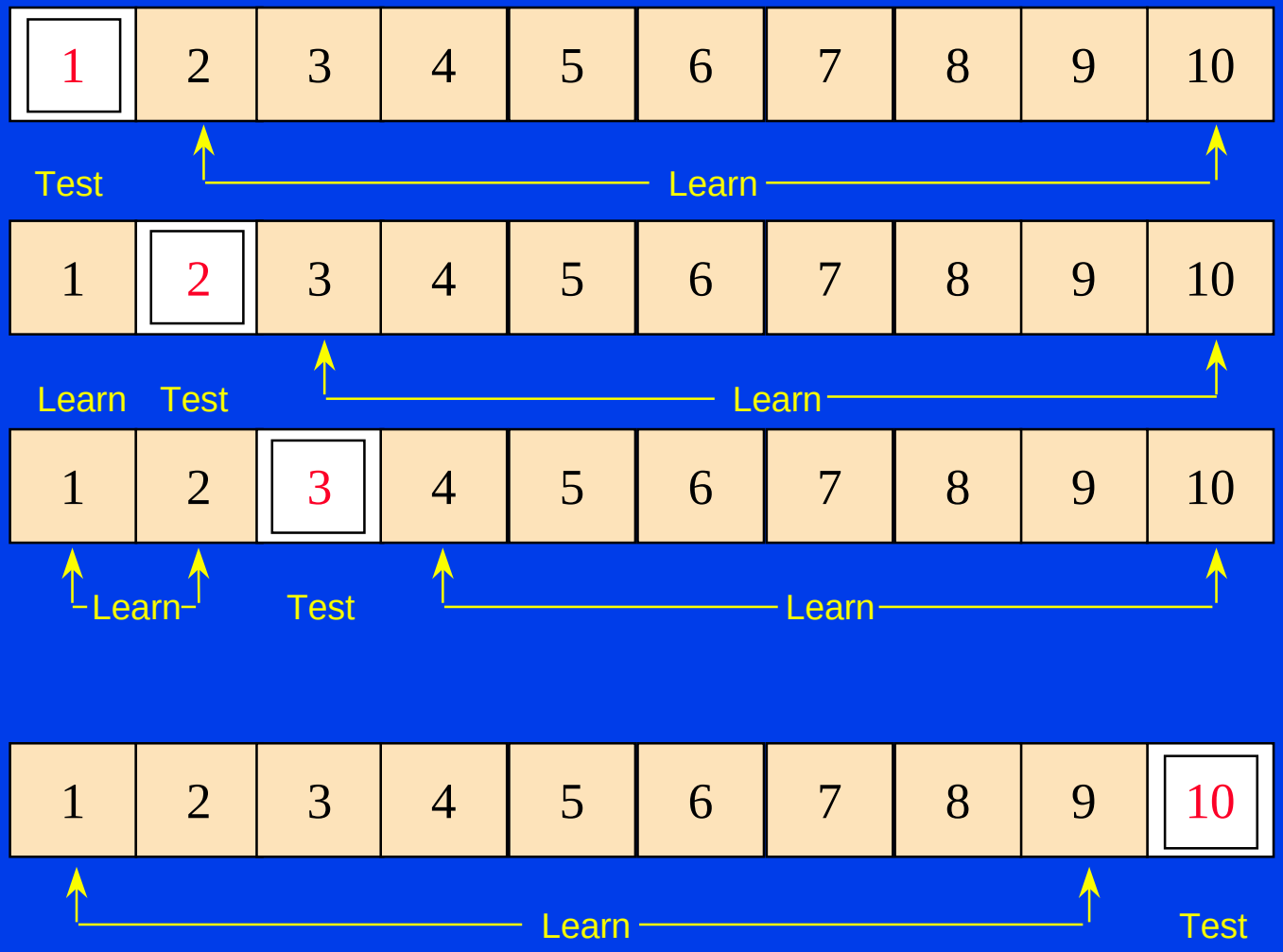
Cross-Validation

- Cross-validation is a recent computer intensive development in statistics
- Purpose is to protect oneself from over fitting errors
  - Don't want to capitalize on chance — track idiosyncrasies of this data
  - Idiosyncrasies which will NOT be observed on fresh data
- Ideally would like to use large test data sets to evaluate trees, N>5000
- Practically, some studies don't have sufficient data to spare for testing
- Cross-validation will use SAME data for learning and for testing

10-Fold Cross-Validation: The Industry Standard

- Begin by growing maximal tree on ALL data; put results aside
- Divide data into 10 portions stratified on dependent variable levels
- Reserve first portion for test
- Grow new tree on remaining 9 portions
- Use the 1/10 test set to measure error rate for this 9/10 data tree
  - Error rate is measured for the maximal tree and for all subtrees
  - Error rate available for 2, 3, 4, 5,....etc nodes of 9/10 data sub-trees
- Now rotate out new 1/10 test data set
- Grow new tree on remaining 9 portions
  - Compute error rates as before
- Repeat until all 10 portions of the data have been used as test sets

Cross-Validation Procedure



Cross-Validation Details

- **Every observation is used as test case exactly once**
- **In 10 fold CV each observation is used as a learning case 9 times**
- **In 10 fold CV 10 auxiliary CART trees are grown in addition to initial tree**
- **When all 10 CV trees are done the error rates are cumulated (summed)**
- **Summing of error rates is done by TREE complexity**
- **Summed Error (cost) rates are then attributed to INITIAL tree**
- **Observe that no two of these trees need be the same**
- **Even the primary splitter of the root node may be different across CV trees**
- **Results are subject to random fluctuations — sometimes severe**

Cross-Validation Replication & Tree Instability — Macro View

- Macro checking — Replicated CV
- Rerun CV analysis several times to compare broad features of results
  - Error rate of best tree
  - Size of tree
- Initial tree will always be the same because it is grown on all the data
  - Separate runs with new random number seed may yield different error rates
  - Different amount of pruning
- Best estimate of true error rate will be an average of the replicated CV runs
- Apply the average error rate to one of the selected trees judgmentally

Cross-Validation Replication & Tree Instability — Micro View

- **Look at the separate CV tree results of a single CV analysis**
- **CART produces a summary of the results for each tree at end of output**
- **Table titled "CV TREE COMPETITOR LISTINGS" reports results for each CV tree Root (TOP), Left Child of Root (LEFT), and Right Child of Root (RIGHT)**
- **Want to know how stable results are -- do different variables split root node in different trees?**
- **Only difference between different CV trees is random elimination of 1/10 data**

Cross-Validation Results

- **Since CV can produce so much output it is necessary to summarize and select**
- **Regardless of what CART produces for a final tree, CV runs generate one full tree for each CV**
- **10-fold CV produces**
  - **10 CV maximal trees**
  - **Initial maximal tree**
  - **Final pruned tree — IF IT SURVIVES**
- **Default node detail output only for FINAL tree**
- **Remainder of results generally not reported**
- **Can lead to one not uncommon frustration**

Cross-Validation Getting Selected Output

- **May want to see all intermediate tree results or only some of it**
- **Least amount of tree detail — tree pictures and Terminal node summary only**
- **Obtained with two commands working together**
  - **LOPTIONS NOPRINT suppresses node detail**
  - **BOPTIONS COPIOUS produces information for each CV tree**
- **Most voluminous output available is complete node detail for each CV tree**
- **BOPTIONS COPIOUS with full printing will print final and all CV trees**

CV-TREE Variable Scaled Importance Measures

Variable

Cross-Validation Tree Number

	V01	V02	V03	V04	V05
INIT	0.001	0.000	0.002	0.000	0.001
CV 1	0.044	0.072	0.203	0.199	0.574
CV 2	0.039	0.056	0.074	0.149	0.439
CV 3	0.050	0.051	0.114	0.203	0.286
CV 4	0.038	0.082	0.130	0.065	0.682
CV 5	0.068	0.122	0.211	0.177	0.816
CV 6	0.123	0.048	0.269	0.116	0.370
CV 7	0.114	0.123	0.193	0.190	0.346
CV 8	0.045	0.000	0.202	0.195	0.685
CV 9	0.137	0.067	0.127	0.256	0.405
CV 10	0.048	0.044	0.135	0.223	1.000
FINAL	0.000	0.000	0.079	0.000	0.328

- **Note: Initial tree might be huge and variable importance measures based on it are not useful**
- **V05 is important in many CV trees**
- **V01 is never important**

CV-TREE Competitor Listings
Root Node Splitters

- **CV-TREE Competitor Listings (Split type, Variable, Split Value, Improvement)**
- **Note V15 strong in all trees**

TOP SPLIT	COMPETITORS--Rank				
	MAIN	1	2	3	4
CV 1	N	N	N	N	N
	V15	V07	V09	V13	V11
	2.396	2.450	2.984	3.057	3.454
	0.134	0.131	0.127	0.124	0.124
CV 2	N	N	N	N	N
	V07	V11	*V15*	V09	V13
	2.450	3.454	2.396	3.045	2.797
	0.140	0.136	0.132	0.125	0.123
[etc...]					
CV 10	N	N	N	N	N
	V09	V07	*V15*	V14	V13
	3.035	2.450	2.284	2.106	3.057
	0.152	0.145	0.140	0.123	0.123
FINAL	N	N	N	N	N
	V15	V07	V09	V11	V13
	2.396	2.450	3.035	3.454	3.057
	0.138	0.137	0.131	0.129	0.122

CV-TREE Competitor Listings
Left Child of Root Node Splitters

LEFT SPLIT		COMPETITORS				
	MAIN	1	2	3	4
CV 1	N	N	N	N	N
	V09	V11	V10	V17	V12
	3.032	3.024	2.661	0.253	3.214
	0.054	0.051	0.049	0.030	0.029
CV 2	N	N	N	N	N
	V11	V13	V09	V05	V17
	2.842	2.254	4.136	0.237	4.347
	0.051	0.045	0.031	0.029	0.027
[etc...]					
CV 10	N	N	N	N	N
	V11	V05	V07	V13	V06
	3.454	0.567	2.460	2.867	2.172
	0.105	0.098	0.089	0.086	0.078
FINAL	N	N	N	N	N
	V09	V11	V10	V12	V17
	3.032	3.024	2.661	3.214	0.157
	0.054	0.053	0.047	0.030	0.028

CV-TREE Competitor Listings
Right Child of Root Node Splitters

RIGHT SPLIT		COMPETITORS			
	MAIN	1	2	3	4
CV 1	N	N	N	N	N
	V11	V05	V13	V12	V07
	C3.454	0.725	2.253	2.589	2.951
	0.089	0.067	0.065	0.056	0.055
CV 2	N	N	N	N	N
	V10	V11	V09	V17	V12
	3.610	3.343	2.973	0.157	3.221
	0.098	0.089	0.088	0.060	0.054
[etc...]					
CV 10	N	N	N	N	N
	V07	V16	V08	V15	V13
	1.852	2.143	2.575	2.284	3.087
	0.041	0.038	0.036	0.033	0.031
FINAL	N	N	N	N	N
	V11	V05	V13	V06	V07
	3.454	0.725	2.701	2.172	2.436
	0.094	0.071	0.060	0.058	0.051

Node Specific Error Rates

- **Although pruning is done node by node, CART trees are evaluated as *trees***
- **Error rates reported based on the test set or CV refer to the tree overall**
- **Why? Tree is a statistical object and error rate is expected for the structure**
- **In practice you WILL want to look at node error rates AND sample size**
- **May want to reduce confidence in nodes that look inaccurate or small size**
- **Is there any help here?**

Approaches to Measuring Node Specific Error Rates — Breiman's Adjustment

- **Breiman introduced an adjustment scheme incorporated into CART**
- **Printed on TERMINAL NODE INFORMATION report**
- **Here is a fragment of one such report**
- **Breiman estimates printed in brackets and can be far higher than raw reports**
- **Terminal nodes 1 and 2 have adjusted node errors 2 to three times raw rate**

Terminal Node Information

Node	N	Prob	Class	Cost	Class	N		Prob	Complexity Threshold
1	26	0.092	1	0.104		1	23	0.896	0.016
				[	0.326]	2	3	0.104	
						3	0	0.000	
2	9	0.029	2	0.447		1	0	0.000	0.016
				[	0.890]	2	5	0.553	
						3	4	0.447	
3	108	0.352	2	0.226		1	15	0.153	0.057
				[	0.298]	2	85	0.774	
						3	8	0.074	
4	28	0.093	3	0.270		1	7	0.270	0.043
				[	0.489]	2	0	0.000	
						3	21	0.730	
5	53	0.187	1	0.121		1	46	0.879	0.043
				[	0.247]	2	1	0.017	
						3	6	0.104	
6	76	0.246	3	0.159		1	2	0.029	0.103
				[	0.259]	2	10	0.130	
						3	64	0.841	

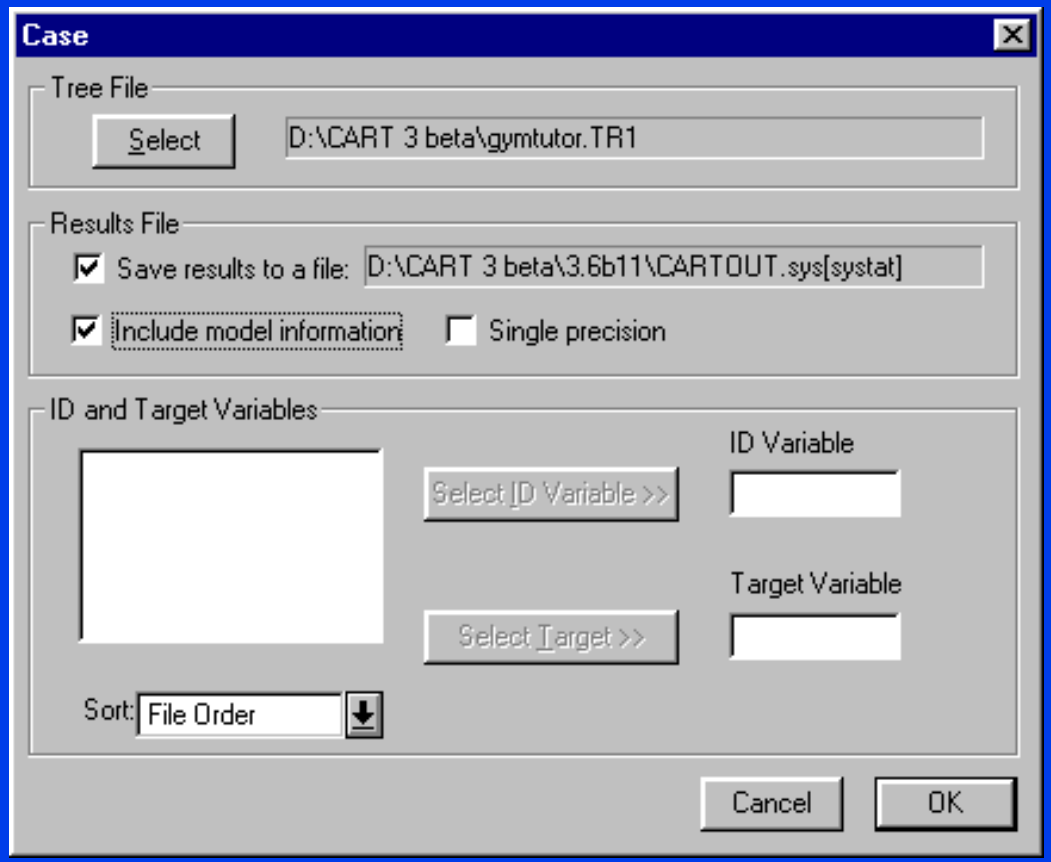
Approaches to Measuring Node Specific Error Rates — Olshen's Approach

- Olshen's approach is based on massively repeated Cross-Validation
- Not currently implemented in CART
  - Conduct a CV=10 analysis
    - During this analysis each observation was a test case exactly once
    - Record whether it was classified correctly or incorrectly as test case
  - Repeat CV=10 run with new random number seed 100 to 500 times
    - Each CV cycle record outcomes (correct or not) for test cases
  - Result is 100 to 500 measures for each case correct or not
  - Trace each case to its terminal node in the initial tree
- Apply recorded accuracy statistics of each case node by node

Classifying New Data

- In field application, CART tree used as predictive engine
- Tree structure saved via TREE command
- TREE CELLPHON in the analysis phase saves a file called CELLPHON.TR1
- To drop new data down this tree use these commands
  - USE NEWDATA
  - SAVE FORECAST
  - TREE CELLPHONE
  - CASE
- TREE command will also access previously created tree for prediction

Classifying New Data cont: Case Dialog in GUI



Applying the Tree to New Data Called Dropping Data Down Tree

- Remember that missing values are not a problem
  - CART will use surrogates (if available)
- Core results go to SAVE file
  - classification for each case
  - specific terminal node this case reached
  - complete path down tree
  - whether class was correct or not
- Can also save other variables
  - up to 50 user specified variables such as ID
  - will want ID variables for tracking and merging
  - optionally, all variables used in original model
  - splitters, surrogates

Variables Saved By CASE

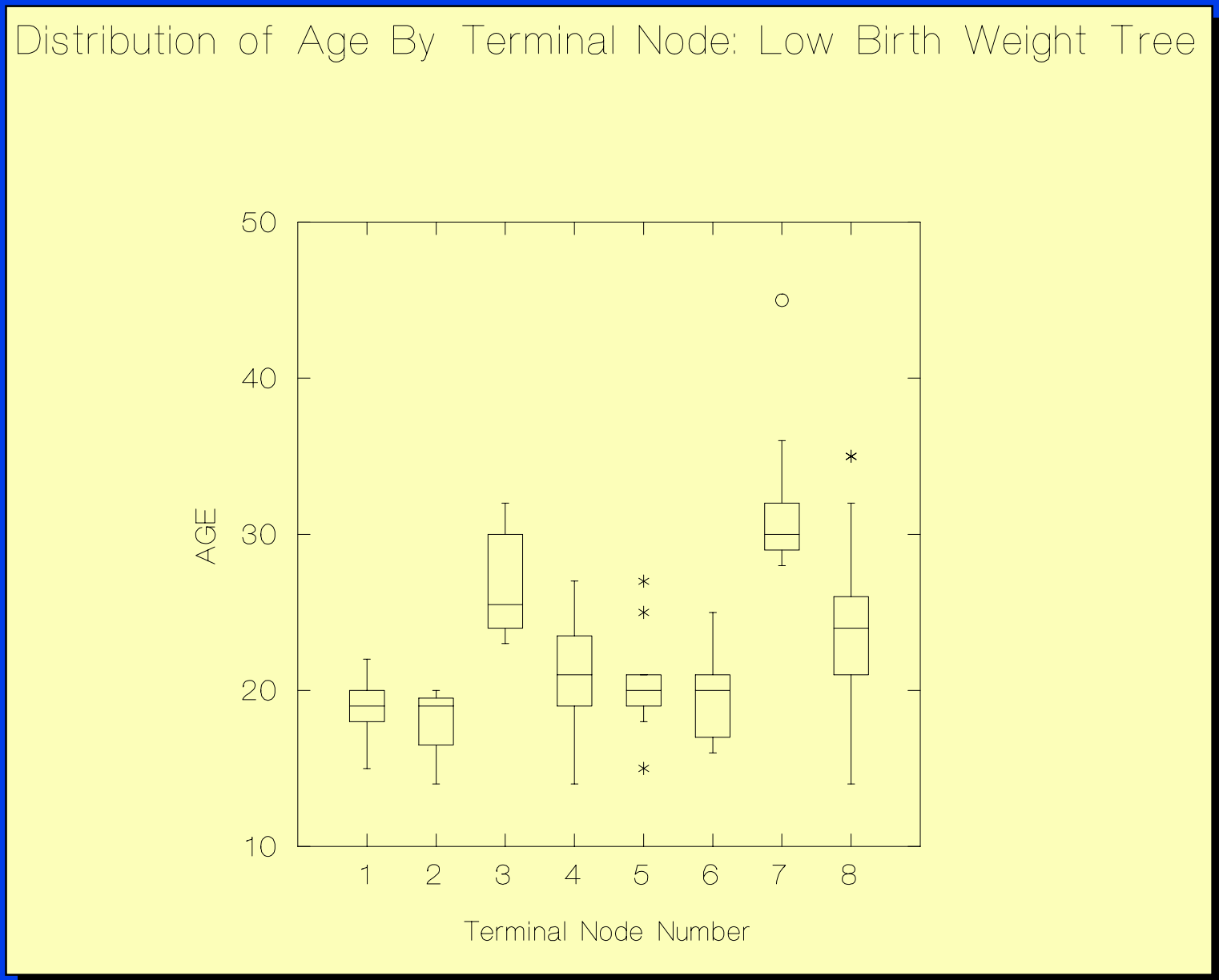
- **One record saved for each case**
- **RESPONSE: classification assigned by CART**
- **NODE: terminal node number**
- **DEPTH: depth level of the terminal node**
- **PATH(n): non-terminal node number at each depth**

Records of the Output Dataset

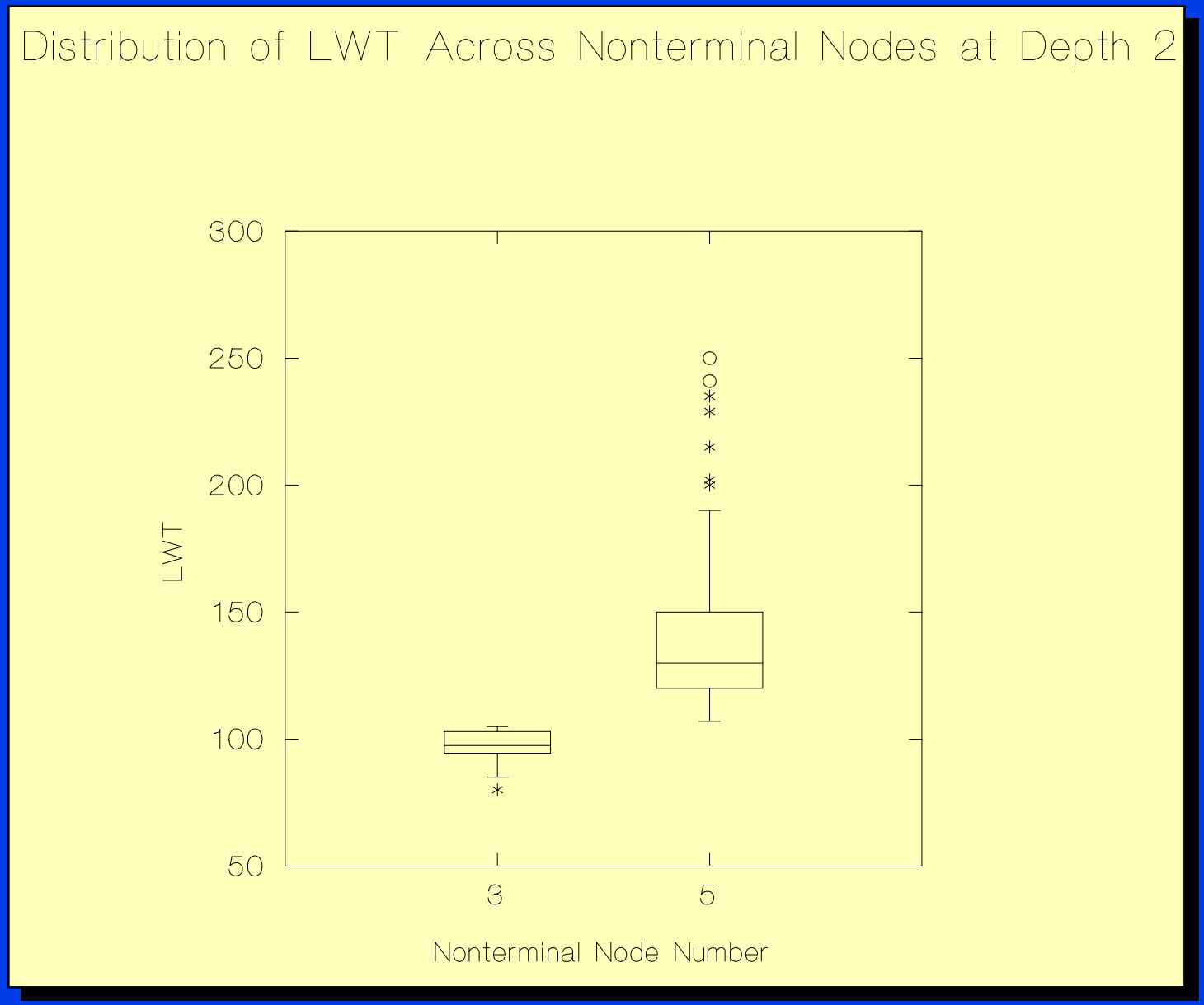
ID	NODE	RESPONSE	DEPTH	PATH1	PATH2	PATH3	PATH4
1	5	1	5	2	5	6	7
2	*7*	*0*	*3*	*2*	*5*	*-7*	*0*
3	1	0	4	2	3	4	-1
4	5	1	5	2	5	6	7
5	5	1	5	2	5	6	7
6	4	0	5	2	5	6	7
7	4	0	5	2	5	6	7

- CASE 2: drops down into NODE = 7, predicted RESPONSE = 0, final DEPTH = 3
- CASE 2: From Root node 1, splits to node 2. Splits again to node 5. Splits again, reaching terminal node 7

Plotting Saved CASE Output



Examining Non-Terminal Nodes



Practical Advice — First Runs

- **Start with ERROR EXPLORE**
- **Set FORMAT=9 to get more decimal places printed in output**
- **Fast runs since only one tree grown**
- **If you have a trivial model (perfect classifiers) you'll learn quickly, and won’t waste time testing**
- **If your resubstitution error rate is very high, test based results will only be worse**
- **Look for some good predictor variables while still exploring**

Practical Advice — Set Complexity to a Non-Zero Value

- Review your TREE SEQUENCE
- How large a tree is likely to be needed to give accurate results?
- Gives you an idea of where to set complexity to limit tree growth
- Consider sample output —Maximal tree grown had 354 nodes many more than credibly needed
- So choose a complexity that will limit maximal tree to say 100 nodes
  - May have to guess at a complexity value
  - Just use something other than 0 for large problems
- Each cross-validation tree will grow a tree until target complexity reached
  - Limiting trees to 100 instead of 300 nodes could substantially reduce your run times without risking error

Practical Advice — Set the SE Rule to Suit Your Data

- **Default of SERULE=1 could trim trees back too far**
- **New wisdom SERULE=0 could give you trees that are too large to absorb**
- **Could try something in between (SERULE=.5 or SERULE=.25)**
- **Once you have settled on a model and test results don't need SE rule at all**
  - **go back to ERROR EXPLORE and use complexity to select tree**
  - **within an interactive analysis choose another tree from the TREE SEQUENCE with the PICK command**
  - **PICK NODES=10**

Practical Advice — Set Up a Battery of Runs

- Prepare batches of CART runs with varying control parameters
- Experienced CART users realize many runs may be necessary to really understand data
- Experiment with splitting rules
  - GINI is default and should always be used
  - TWOING will differ for multi-class problems
  - TWOING POWER=1 will develop different trees
- POWER option can break an impasse; sometimes allows very difficult problems to make headway

Practical Advice — Experiment with PRIORS

- PRIORS EQUAL is default — best chance of good results
- OIshen favors PRIORS MIX — bow in direction of data
- PRIORS DATA least likely to give satisfactory results
- Try a grid search. For a binary classification tree…
  - march their priors from (.3, .7) to (.7, .3)
  - see if a shift in priors grows a more interesting tree
- Remember, priors shift CART's emphasis from one class to another
- A small shift in priors can make an unbalanced tree more balanced and cause CART to prune a tree a little differently
- The testing and pruning process will keep trees honest

Practical Advice — Experiment with TEST Methods

- CV=10 should be used for small data sets
- Rerun CV with new random number seed
  - e.g. SEED 100,200,500
- Check for general agreement of estimated error rate
- Splits and initial tree will not change
  - ONLY ERROR RATE estimate and possibly optimal size tree will change
- Try ERROR P=p
  - Where p=.1 or .25 or .5, etc.
  - Very fast compared to CV
  - Problematic when sample is very small

Practical Advice — Prepare a Summary of all Your Runs

- **Print out just the TREE SEQUENCES**
- **In a separate report print out just the primary split variables**
- **In a separate report print out the top of the VARIABLE IMPORTANCE list**

Practical Advice for CART Analyses — Variable Selection

- In theory CART can find the important variables for you
- Theory holds when you have massive data sets (say 50,000 per level)
- In practice you need to be judicious and help
- Eliminate nonsense variables such as ID, Account numbers
  - They probably track other information
- Eliminate variations of the dependent variable
  - You might have Y and LOGY in the data set
  - You might have a close but imperfect variant of DPV
- Can control these nuisance variables with EXCLUDE command

Practical Advice for CART Analyses — Variable Selection II

- Worthwhile to think about variables that should matter
- Begin with excluding those that should not — TO GET STARTED
- CART's performance on model with few variables could be much better than with a larger search set
- Reason is potential for mistracking in INITIAL tree
  - Say a chance variation makes X50 a primary splitter
  - Testing finds that this does not hold up
  - Error rate on tree is high and results disappointing
- If fluke variable not even in search set then bad split doesn't happen
  - Instead of fluky X50 a less improving but solid variable is used
  - Testing supports it and final tree performs better

Two CART Runs Using Different Number of Variables - 1

- **CART Run with 168 Independent Variables**
  - **Dependent variable: MALIGN**

Terminal Tree Nodes		Cross-Validated Relative Cost	Resubstitution Relative Cost	Complexity Parameter
1	21	1.077 +/- 0.071	0.031	0.000
2	19	1.031 +/- 0.073	0.046	0.004
3	18	1.023 +/- 0.075	0.062	0.008
4	13	1.046 +/- 0.075	0.177	0.012
5	12	1.046 +/- 0.075	0.208	0.015
6**	10	0.992 +/- 0.075	0.277	0.017
7	7	1.008 +/- 0.076	0.400	0.021
8	5	1.008 +/- 0.076	0.492	0.023
9	4	1.015 +/- 0.075	0.577	0.042
10	3	1.046 +/- 0.075	0.677	0.050
11	1	1.000 +/- 0.000	1.000	0.081

Initial misclassification cost = 0.500
Initial class assignment = 0

Two CART Runs Using Different Number of Variables -2

- **CART Run with subset of 20 Independent Variables (Same Dataset, Same Dependent Variable all variables from original set of 168)**
  - **Dependent variable: MALIGN**

Terminal Tree Nodes		Cross-Validated Relative Cost	Resubstitution Relative Cost	Complexity Parameter
1	20	0.831 +/- 0.073	0.146	0.000
2	17	0.846 +/- 0.074	0.169	0.004
3	16	0.877 +/- 0.074	0.185	0.008
4	14	0.862 +/- 0.074	0.223	0.010
5	12	0.892 +/- 0.075	0.269	0.012
6**	7	0.854 +/- 0.075	0.423	0.015
7	5	0.815 +/- 0.075	0.492	0.017
8	4	0.854 +/- 0.075	0.577	0.042
9	3	0.869 +/- 0.075	0.677	0.050
10	1	1.000 +/- 0.000	1.000	0.081

  - *Initial misclassification cost = 0.500*
  - *Initial class assignment = 0*

Misclassification Tables from the Small & Large Variable List Runs

- **CART Run with 168 Independent Variables**

Class	Prior Prob.	CROSS VALIDATION N	CROSS VALIDATION N Mis-Classified	CROSS VALIDATION Cost	LEARNING SAMPLE N	LEARNING SAMPLE N Mis-Classified	LEARNING SAMPLE Cost
0	0.500	130	57	0.438	130	21	0.169
1	0.500	65	36	0.554	65	7	0.108
Total	**1.000**	**195**	**93**		**195**	**29**	

- **CART Run with subset of 20 Independent Variables (Same Dataset, Same Dependent Variable)**

Class	Prior Prob.	CROSS VALIDATION N	CROSS VALIDATION N Mis-Classified	CROSS VALIDATION Cost	LEARNING SAMPLE N	LEARNING SAMPLE N Mis-Classified	LEARNING SAMPLE Cost
0	0.500	130	54	0.415	130	36	0.277
1	0.500	65	26	0.400	65	14	0.215
Total	**1.000**	**195**	**80**		**195**	**50**	

Troubleshooting Problem: Cross-Validation Breaks Down

- **CART requires at least one case in each level of DPV**
- **If only a few cases for a given level in data set, random test set during cross-validation might get no observations in that level**
- **May need to aggregate levels OR**
- **Delete levels with very low representation from problem**

Troubleshooting — NO TREE CREATED

- **The suggestions made in the PRACTICAL ADVICE section should get around this**
- **If not request COPIOUS OUTPUT for CV runs**
- **First, set complexity high enough to limit tree growth**
  - **Allow maybe 6 terminal nodes to get started**
- **Then review all tree output**
  - **Do the splitters make sense**
  - **Do you have very unbalanced splitting**

Tree Sequence

- **Dependent variable: CLT**

Terminal Tree Nodes	Cross-Validated Relative Error	Resubstitution Relative Error	Complexity Parameter	Relative Complexity
1	30	1.4237 +/- 0.0848	0.5150	0.0000	0.0000
10	18	1.3095 +/- 0.0735	0.6153	0.7625	0.0110
11	17	1.2745 +/- 0.0683	0.6275	0.8401	0.0121
12	15	1.2711 +/- 0.0682	0.6523	0.8603	0.0124
13	14	1.2631 +/- 0.0678	0.6648	0.8646	0.0125
14	13	1.1898 +/- 0.0530	0.6784	0.9372	0.0135
15	11	1.1891 +/- 0.0515	0.7116	1.1508	0.0166
16	7	1.0782 +/- 0.0446	0.7814	1.2076	0.0174
17	6	1.0591 +/- 0.0322	0.8050	1.6343	0.0236
18	4	1.0377 +/- 0.0261	0.8618	1.9639	0.0284
19**	1	1.0000 +/- 0.0003	1.0000	3.1890	0.0461

Initial mean = 1.4676
Initial variance = 0.1851
No tree created
60000 PREPROCESSOR WORKSPACE ELEMENTS
500000 TREE BUILDING WORKSPACE ELEMENTS

NO TREE GENERATED Message

- CART will produce a TREE SEQUENCE and nothing else when test error rate for tree of any size is higher than for root node
- In this case CART prunes ALL branches away leaving no tree
- Could also be caused by SERULE=1 and large SE
- Trees actually generated - including initial and all CV trees
- CART maintains nothing worthwhile to print
- Need some output to diagnose
- Try an exploratory tree with ERROR EXPLORE
- Grows maximal tree but does not test
- Maximum size of any initial tree or CV tree controlled by COMPLEXITY parameter

Memory Management: The Issues

- CART is a highly memory intensive program
  - Stores all data in RAM
  - All tree information maintained in RAM
- Not a problem for modest data sets or Unix workstations
- Can be a problem for PCs
- Windows versions available for 16, 32, 64 and 128mb RAM
  - Custom versions are available
- If your problem appears to be too big for your current CART, there are things you can do

Memory Management: Commands and Reports

- **Issuing the MEMORY command after problem set up & before BUILD**
- **Example comes from a small PC version**
- **Windows and minicomputer versions will have much larger workspaces**
  - **from 100,000 to 32,000,000 workspace elements**
- **But any CART analysis can run out of memory if the data set is large enough**

Memory Management

- **Current Memory Requirements**
  - **TOTAL: 90969 DATA: 37464 ANALYSIS: 53505**
  - **AVAILABLE: 55502**
  - **DEFICIT: 35467**

Adjustable Parameters	*Current Value*	*Solution Value*	*Solution Surplus*
Learning Set	1338	700	295
Depth	32	NONE	(19788)
Nodes	257	NONE	(19581)
Atom	10	NONE	(20693)
Subsampling	1000	NONE	(5190)
Continuous Ind Var	27	12	353
Dependent Var Classes	5	NONE	(34248)

() Denotes Deficit at Extreme Allowable Parameter Value

Reading the Memory Management Report

- Report displays current demanders of memory
- Also displays possible ways to free up memory
- Could reduce learning sample size to just 700 cases
- Alternatively, could reduce variables searched to 12
- These are the only definite solutions displayed
- However, a combination of other parameter values might help
  - By playing with DEPTH and NODES together
- MAXIMAL TREE probably does not need to get to 257 terminal nodes
- DEPTH of maximal tree does not need to reach 32
- Can learn more by reducing learn set size a little and reissuing MEMORY command

Memory Management — LIMIT Experiments

- **Set some LIMITs to let tree get started**
- **After results obtained refine with BOPTIONS COMPLEXITY, etc.**
- **Try LIMIT LEARN=1000, SUBSAMPLE=500 in above situation and get some further feedback from the memory analyzer**

New Developments for CART

- CART methods are continuing to be developed and advanced by Breiman, Friedman, Olshen, and Stone
- Also, an active R&D program at Salford Systems
- Methods designed to improve accuracy of classifiers
- Introduce fuzzy logic like concepts for split points
- Extend methods to time series data (financial market applications)
- Adaptation to censored data (survival analysis)
- Enhancements will be incorporated into new versions of CART software

Accuracy Improvement with a Committee of Experts

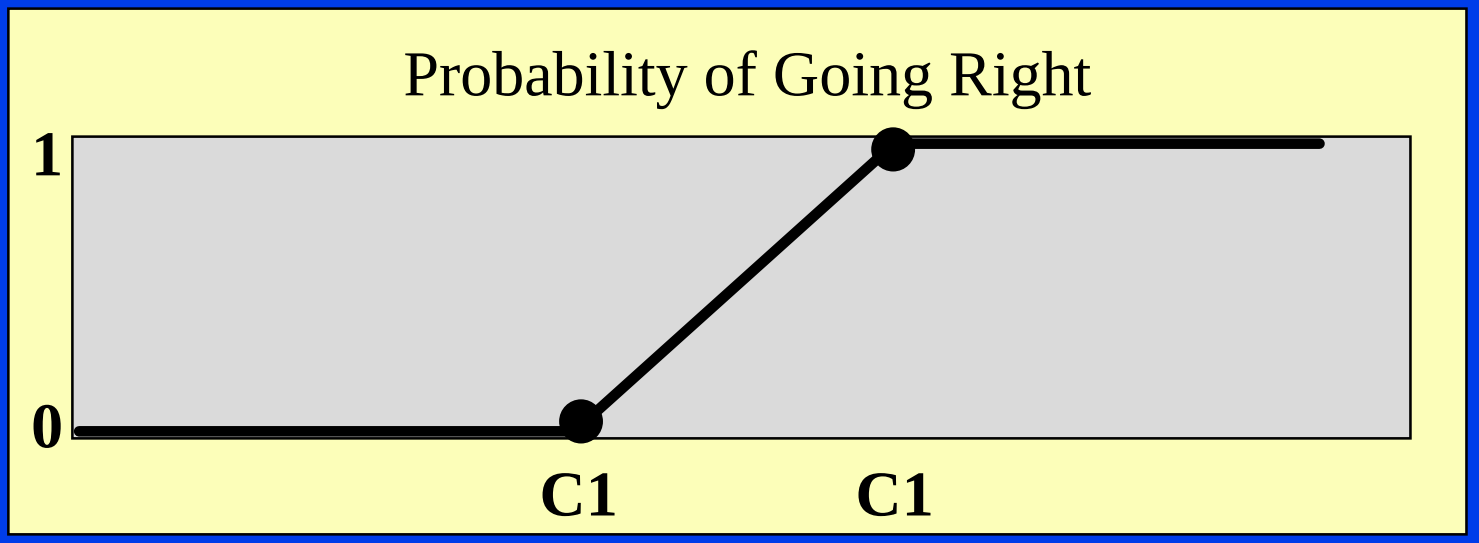
- Tree structures have proven to be high variance objects
- Want to find ways to reduce this variance -- say by averaging
- Breiman has introduced method of bootstrap aggregation
- Works by drawing large number of bootstrap samples
  - Sampling with replacement to replicate original sample
  - On average will only include 2/3 of original learn data
  - Sample brought to full size by repeated observations

Method is Bootstrap Aggregation (Bagger)

- Each bootstrap sample used as basis for growing trees
- Test method optionally used to select "optimal" tree
- Another bootstrap sample then drawn and process repeated
- Each bootstrap sample generates a tree
- Often these trees will have different personalities
  - Focus on different aspects of the data
- Moderate number grown, say 20-50
- A consensus prediction is then obtained for each new case
- Every bootstrap sample tree consulted using plurality rules
- In many (NOT all) examples BAGGER reduced error rate 20%-40%

Probabilistic Splits

- **Conventional CART uses a "hard" split**
- **Is X <=c? — has a yes/no answer**
- **Means that a small change in the value of variable can have large change on outcome**
- **Change from X=c - e to X=c + e will change path down tree**
- **Breiman has introduced concept of probabilistic split**



Probabilistic Splits

- For $X<c1$ case always goes left and for $X>c2$ case always goes right
- For $c1<X<c2$ case goes left with probability proportional to how close it is to $c1$
- Means that a small change in X has a small impact on prediction
- Cases end up in terminal nodes with probability not certainty
- Tree produces a distribution over terminal nodes for each case

Split Revisitation

- **Conventional CART is myopic; looks only at own child nodes**
- **Cannot consider entire tree or grandchildren**
- **Split revisitation, developed by Breiman, re-examines trees after growth and initial pruning**
- **Begins with a tree from original tree sequence**
  - **Notes SIZE (K terminal nodes) and impurity of this tree**
- **Starting at root node, attempt is made to revise root split alone holding rest of tree constant**
- **If an improvement in the impurity of the tree can be made revise root split**

Split Revisitation

- Go down to leaves of tree and expand tree from the leaves
  - Grow via normal splitting
- This is now meaningful because different composition of cases in the leaves
- Prune back to a tree of same size as tree started with having minimum impurity for this tree size
  - Pruning uses a dynamic programming algorithm not cost-complexity
- Establishes a candidate tree
- Now move to left child of root and repeat process revisiting this node
  - Re-optimize left child
  - Regrow tree out from the leaves
  - Prune back via dynamic programming
- Repeat process for all non-terminal nodes of the tree being revised

Split Revisitation

- **Result of this process is a re-optimized K-terminal tree**
- **Search procedure is repeated for K-1 terminal node tree from original tree sequence**
- **Entire process repeated for this tree leading to a re-optimized K-1 terminal node tree**
- **Now have a sequence of re-optimized trees**
- **These trees may have quite different splitters at root & other nodes**
- **Best found via test sample**
- **Tests show that accuracy can be improved in many cases**
- **Where accuracy is not improved new predictive tree can be much smaller**