ANALYTICS TRAINING INSTITUTE

Decision Trees

Classification

- Refers to “grouping or arranging into categories”
- More specifically, it means grouping of similar objects into categories
- It is a form of supervised learning

- Eg. If we have a customer base of 100,000. We would like to find out which of our customers falls in the low usage, medium usage and high usage groups.
- We can choose the parameters by which we will group the customers, like frequency of usage
- At the end of the classification exercise, each group would have homogeneous observations, with the maximum possible difference between groups

Decision Trees

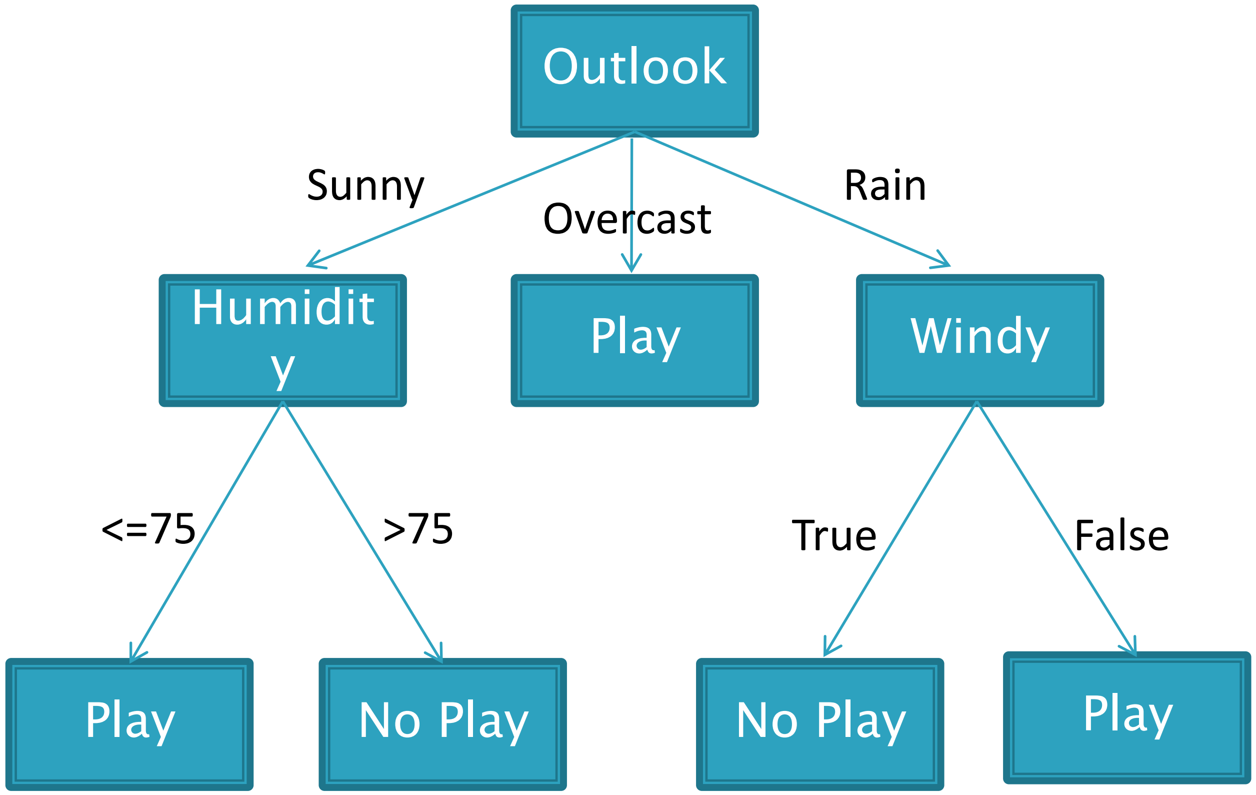
- A decision tree is a decision support tool that uses a tree-like graph to represent decisions and their possible outcomes
- We can identify the strategy that is most likely to reach a goal
- It’s a mapping of observations about an item to conclusions about it’s target value (class).

Example of a decision tree

Outlook	Temp (F)	Humidity (%)	Windy	Class
Sunny	79	90	TRUE	No Play
Sunny	56	70	FALSE	Play
Sunny	79	75	TRUE	Play
Sunny	60	90	TRUE	No Play
Overcast	88	88	FALSE	Play
Overcast	63	75	TRUE	Play
Overcast	88	95	FALSE	Play
Rain	78	60	FALSE	Play
Rain	66	70	FALSE	Play
Rain	68	60	TRUE	No Play

Tree Structure

- Rule 1: If it is sunny and humidity is not above 75%, then **play**
- Rule 2: If it is sunny and humidity is above 75%, then **don't play**
- Rule 3: If it is overcast then **play**
- Rule 4: If it is rainy and not windy, **then play**
- Rule 5: If it is rainy and windy, then **don't play**



Advantage of Decision tree Classification

- Easy to understand – the tree is self-explanatory
- Can handle both numerical & categorical attributes
- Provides clear indication of which field is important for prediction and classification

Shortcomings of Decision tree Classification

- Some decision trees can deal only with binary-valued outcomes
- The process of growing tree is computationally expensive
- There might be a certain loss of accuracy by grouping continuous variables into categories

Decision Tree Fundamentals

Phases in Tree Growth

- Construction Phase:

  Initial Decision tree is Constructed in this Phase

- Pruning Phase:

  In this stage lower branches are removed to improve the performance

Construction Phase

Recursively,

- Decide the final categories into which continuous and categorical variables will be “split” (Chi–Square Test for Independence)

- Select the split that gives the maximum information gain or reduction in impurity (Entropy and Gini Impurity Index)

Chi Square Test For Independence

- Consider a random sample of 10,000 voters in a town
- We want to see if "Gender and Voting are independent"
- Observed:

	Men	Women	Total
Voted	2792	3591	6383
Didn't vote	1486	2131	3617
Total	4278	5722	10000

Solution:

- H0 : Gender and voting are independent
- H1 : Gender and voting are dependent for this town
- Compute expected table under assumption of the Null Hypothesis:
- Expected counts in Each cell are

  Chance(row i, col j)*N

Expected Table:

	Men	Women	Total
Voted	2731	3652	6383
Didn't Vote	1547	2070	3617
Total	4278	5722	10000

- For each (i,j),
  C(i,j) = (Observed(i,j) – Expected(i,j))^2 / Expected(i,j)
- X2 = Sum(C(i,j))
- Here, X2 = 6.58 → p-value between 1% and 5% → Null Hypothesis is FALSE

Determining the best split

- Best split is the one that best discriminates between records in the different classes
  - Goal is to have a single class predominate in each resulting node
  - Split that maximizes the reduction in **impurity** of node is chosen
    - CART algorithm (Classification And Regression Trees) evaluates all possible binary splits
    - C4.5 algorithm splits categorical predictor with K categories into K children (creates bushiness)

Determining the best split

- Two impurity measures are **entropy** and **Gini index**
  - Impurity of split = weighted average of impurities of children

Splitting INDICES

- ENTROPY
- INFORMATION FOR PARTITION
- GAIN
- GINI INDEX

Entropy Measure

Entropy of a node =

$$-\sum_{i=1}^{K} p_i \log_2(p_i)$$

K = number of classes

p_i = the percentage of records in class i

Entropy

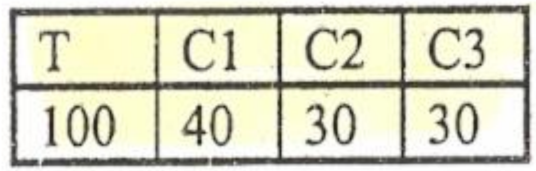
$$0 \leq \text{Entropy} \leq \log_2(K)$$

Good; all records in same class.
PURE

Bad; the *K* classes are equally likely.
AS **IMPURE** AS POSSIBLE

Example Entropy

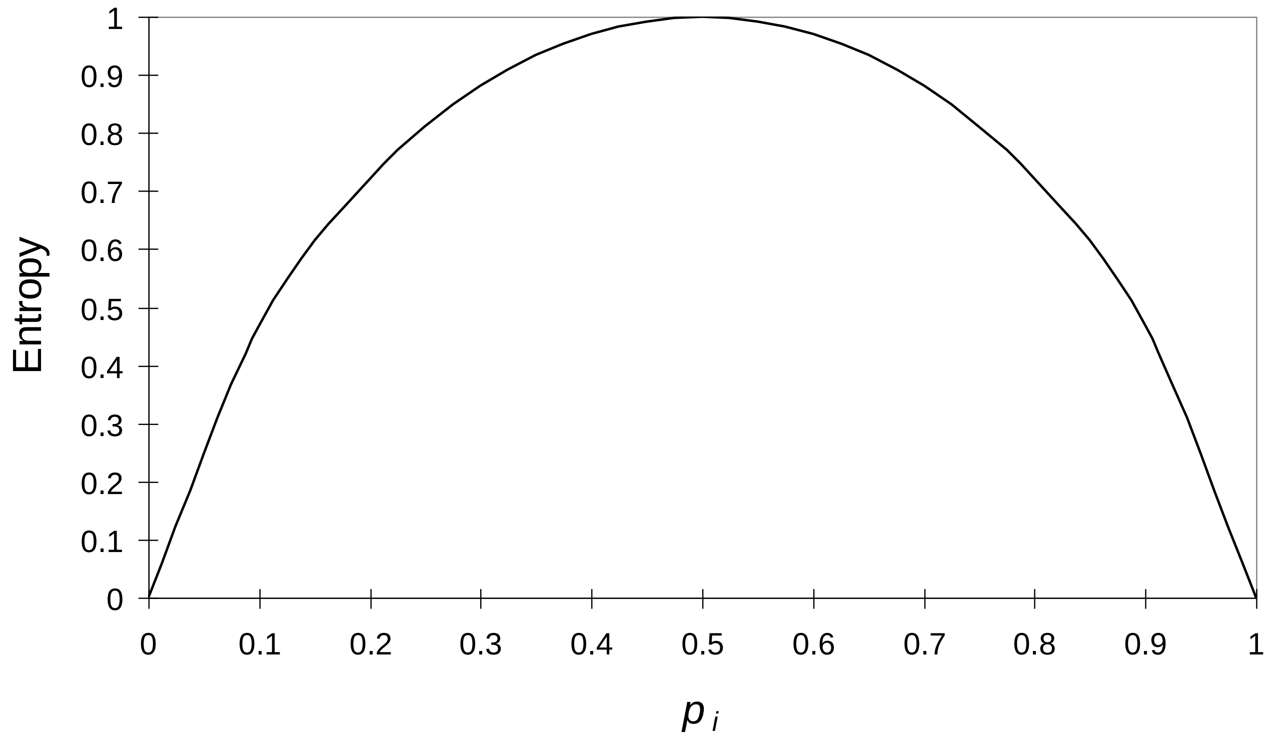
- T → Target Variable
- C1, C2, C3 → Classes of the Target Variable



T	C1	C2	C3
100	40	30	30

$$Info(T) = -\frac{40}{100}\log\frac{40}{100} - \frac{30}{100}\log\frac{30}{100} - \frac{30}{100}\log\frac{30}{100} = 1.09$$

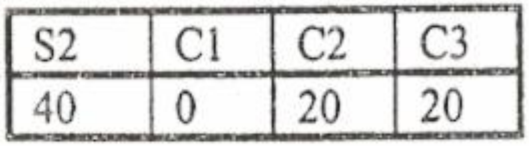
Entropy For 2 classes



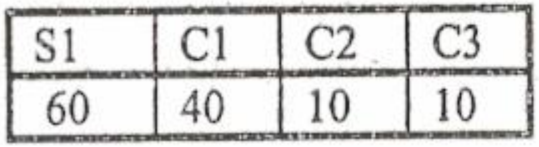
INFORMATION FOR PARTITION

- If T is partitioned based on the value of non-class attribute X, into set T1,T2,…,Tn then the information needed to identify the class of an element T become the weight average of the information to identify the class of the element of Ti, i.e. the Weighted Average

$$Info(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} Info(T_i) \cdot$$



S2	C1	C2	C3
40	0	20	20



S1	C1	C2	C3
60	40	10	10

$$\text{og}\frac{20}{40} - \frac{20}{40}\log\frac{20}{40}\Big) + \frac{60}{100}\Big(-\frac{40}{60}\log\frac{40}{60} - \frac{10}{60}\log\frac{10}{60} - \frac{10}{60}\log\frac{10}{60}\Big) = 0.80.$$

GAIN

- The information Gain represents the difference between the information needed to identify an element of T and the information needed to identify an element of T after the value of attribute X is obtained.

$$\text{GAIN}(X,T) = \text{Info}(T) - \text{Info}(X,T)$$

GAIN RATIO

Gain Ratio (X,T) = Gain (X,T) / Info (X,T)

THE GINI IMPURITY INDEX

K = number of classes

p_i = the percentage of records in class i

$$\text{gini}\ (T) = 1 - \sum^{n} \left(p_i^2\right)$$

The Gini Impurity Index

$$0 \leq GI \leq (K-1)\,/\,K$$

Good; all records in same class.
PURE

Bad; the *K* classes are equally likely.
AS **IMPURE** AS POSSIBLE

Algorithms to grow Decision Trees

Rain Forest

- Rainforest, like other algorithms, is a top-down decision tree induction scheme. It is similar to SPRINT, but instead of attribute lists it uses a different structure called the A VC-set. The attribute list has one tuple corresponding to each record of the training data set. But the A VC-set has one entry for each distinct value of an attribute. Thus, the size of the A VC-set for any attribute is much smaller,

BOAT (Bootstrap Optimistic Algorithm for Tree Construction)

- BOAT (Bootstrap Optimistic Algorithm for Tree Construction) is another approximate algorithm based on sampling.
- As the name suggests, it is based on a well-known' statistical principle called *boot strapping.*
- We take a very large sample *D* from the training data set *T,* so that *D* fits into the main memory.
- Now, using the boot strapping technique, we take many small samples with replacement of *D* as *D1, D2,* ·.· , *Dk* and construct, by any of the known methods, decision trees *DT1, DT2,* ·.· , *DTk,* respectively, for each of these samples.

BOAT

- From these trees, called *bootstrap trees,* we construct a single decision tree for the sample data set *D.* We call this tree the sample tree.

ID3

- Quinlan introduced the ID3, Iterative Dichotomizer 3, for constructing the decision trees from data.
- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.
- At each node the splitting attribute is selected to be the most informative among the attributes not yet considered in the path from the root.

ID3

- Entropy is used to measure how informative a node is.
- This algorithm uses the criterion of information gain to determine the goodness of a split.
- The attribute with the greatest information gain is taken as the splitting attribute, and the data set is split for all distinct values of the attribute.

```
function ID3 Input: (R: a set of non-target attributes, C: the target attribute, S: a training set) returns a decision tree;
begin
  - If S is empty, return a single node with value Failure;
  - If S consists of records all with the same value for the target attribute, return a single leaf node with that value;
  - If R is empty, then return a single node with the value of the most frequent of the values of the target attribute that are found in records of S; [in that case there may be errors, examples that will be improperly classified];
  - Let A be the attribute with largest Gain(A,S) among attributes in R;
  - Let {aj| j=1,2, .., m} be the values of attribute A; Let {Sj| j=1,2, .., m} be the subsets of S consisting respectively of records with value aj for A;
  - Return a tree with root labeled A and arcs labeled a1, a2, .., am going respectively to the trees (ID3(R-{A}, C, S1), ID3(R-{A}, C, S2), .....,ID3(R-{A}, C, Sm);
  - Recursively apply ID3 to subsets {Sj| j=1,2, .., m} until they are empty
end
```

C4.5

- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees and rule derivation.
- In building a decision tree, we can deal with training sets that have records with unknown attribute values by evaluating the gain, or the gain ratio, for an attribute by considering only those records where those attribute values are available.

OVERFIT

- A tree is set to over fit the training data if there exist some other tree T’ which is a simplification of T, such as T has smaller error then T’ over the training set but T’ has a smaller error than T over the entire distribution of instances.

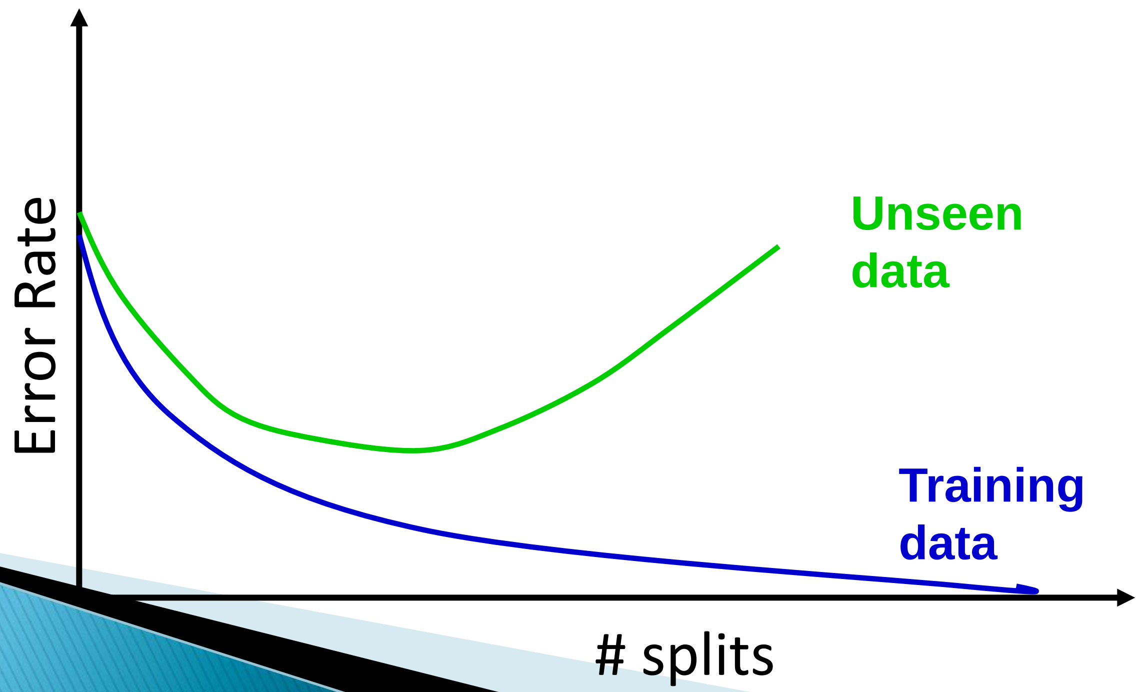
OVERFIT Leads to

- OVERFIT lead to difficulties when there is a noise in the training data, or when the number of training example is too small.

  - Incorrect model
  - Requirement of space be increased and more computational resource
  - Require unnecessary collection of features
  - Difficult to comprehend

Avoiding Over-fitting

- How will the tree perform on new data?
- The error rate of a tree = misclassifications

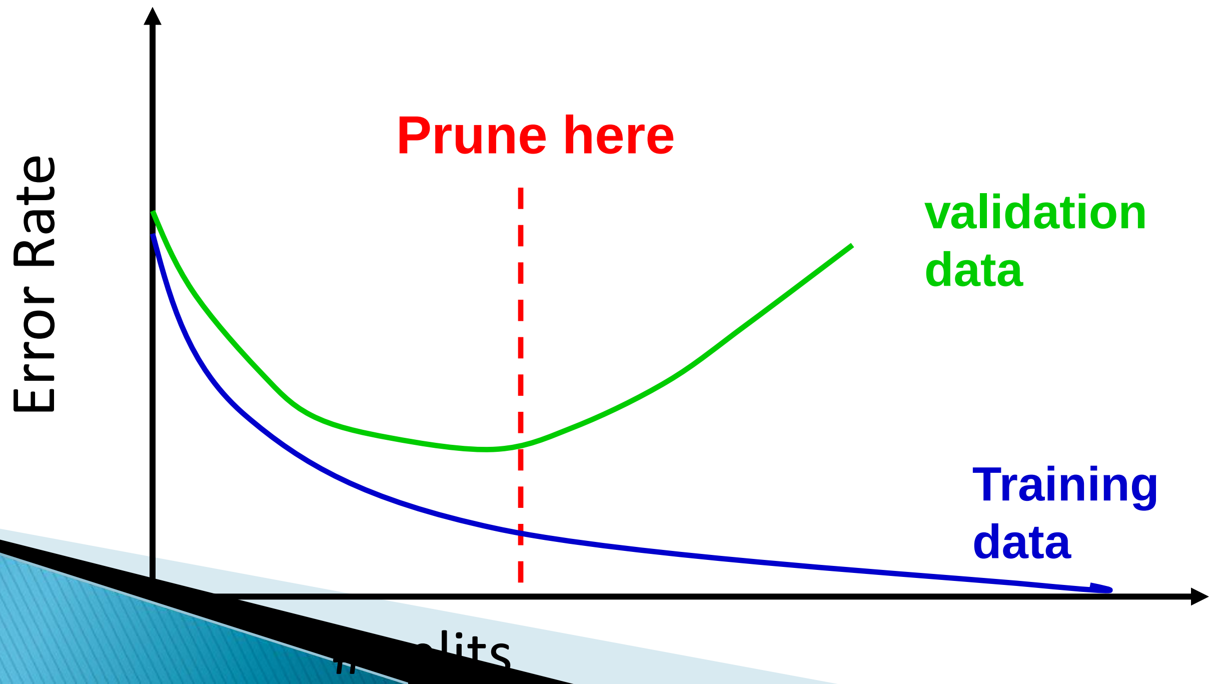


Solution 1: Stopping Tree Growth

- Rules/Criteria used to stop tree growth:
  - Tree depth
  - Minimum # of records (cases) in node
  - Minimum reduction in impurity measure
- Problem: hard to know when to stop…

Solution 2: Pruning (CART, C4.5)

- We would like to maximize prediction accuracy for unseen data
- Use validation set to prune the tree



Pruning

- CART lets tree grow to full extent, then prunes it back
- Idea is to find that point at which the validation error begins to rise
- Generate successively smaller trees by pruning leaves
- At each pruning stage, multiple trees are possible
- Use *cost complexity* to choose the best tree at that stage

α

Cost Complexity

$$CC(T) = Err(T) + \alpha L(T)$$

$CC(T)$ = cost complexity of a tree
$Err(T)$ = proportion of misclassified records
α = penalty factor attached to tree size (set by user)

- Among trees of given size, choose the one with lowest CC
- Do this for each size of tree

Pruning Results

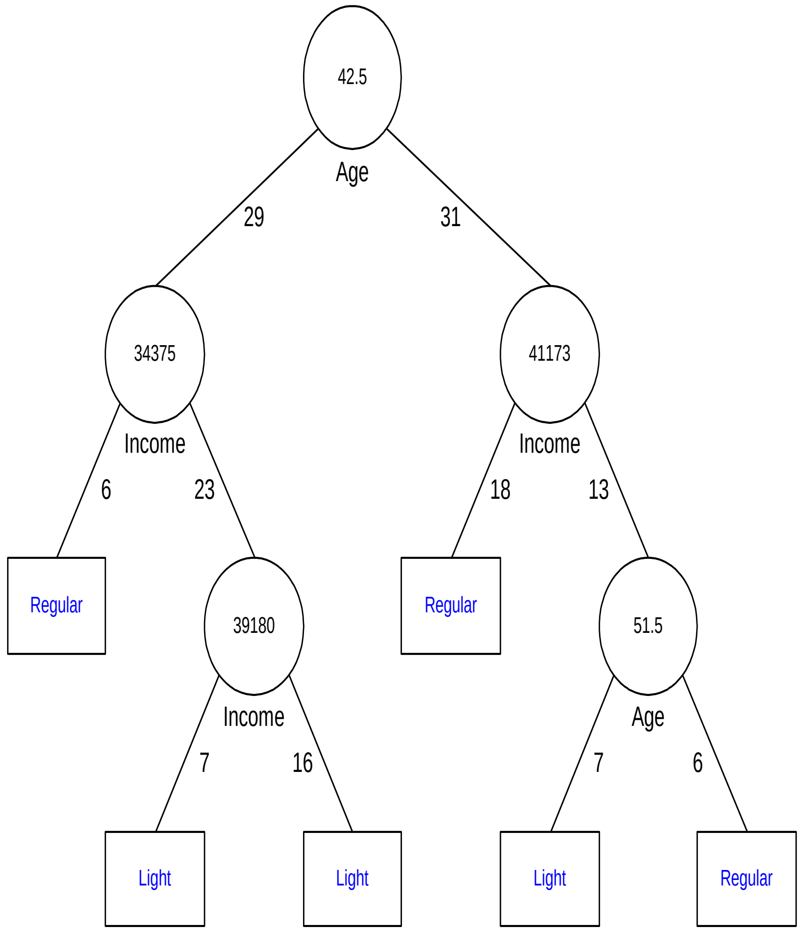
- This process yields a set of trees of different sizes and associated error rates

Two trees of interest:

- Minimum error tree
  - Has lowest error rate on validation data
- Best pruned tree
  - Smallest tree within one std. error of min. error
  - This adds a bonus for simplicity/parsimony

Converting a Tree into Rules

- Convert tree to IF-ELSE-AND rules
  - IF age>42.5 AND income< $41,173 THEN prefers regular beer
  - IF age<42.5 AND income < $34,375 then prefers regular beer
- Can condense some rules
  - If income < $34,375 then regular beer

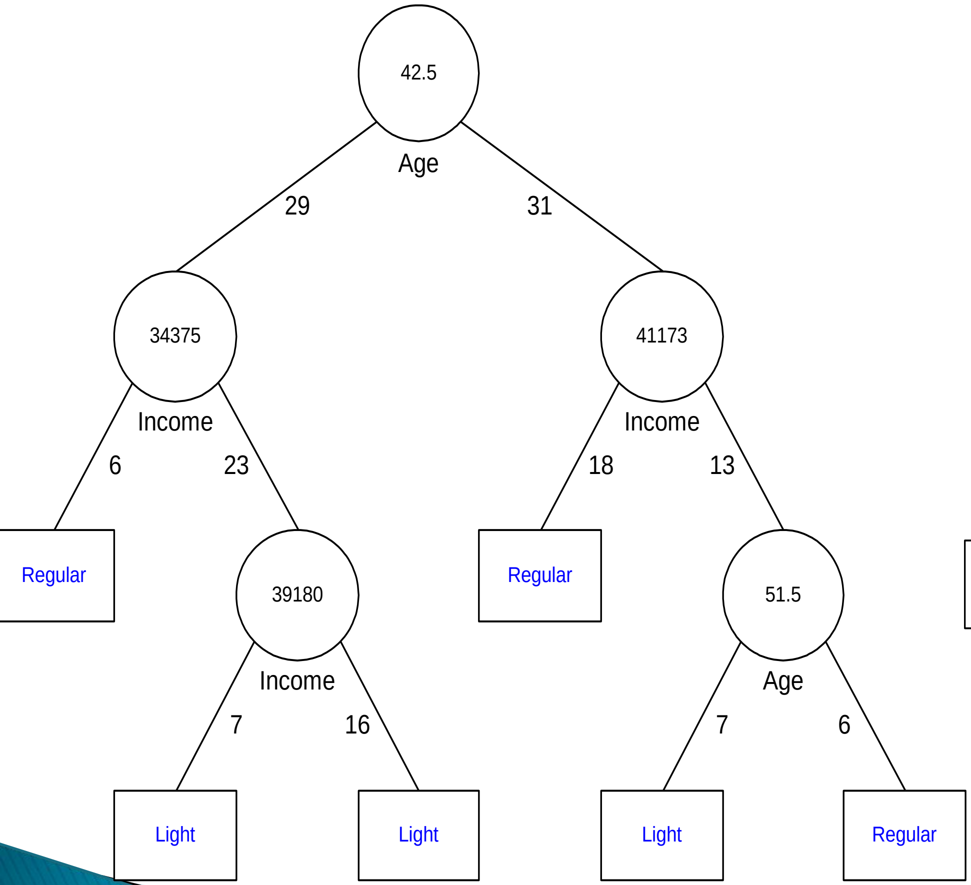


Labeling Leaf Nodes

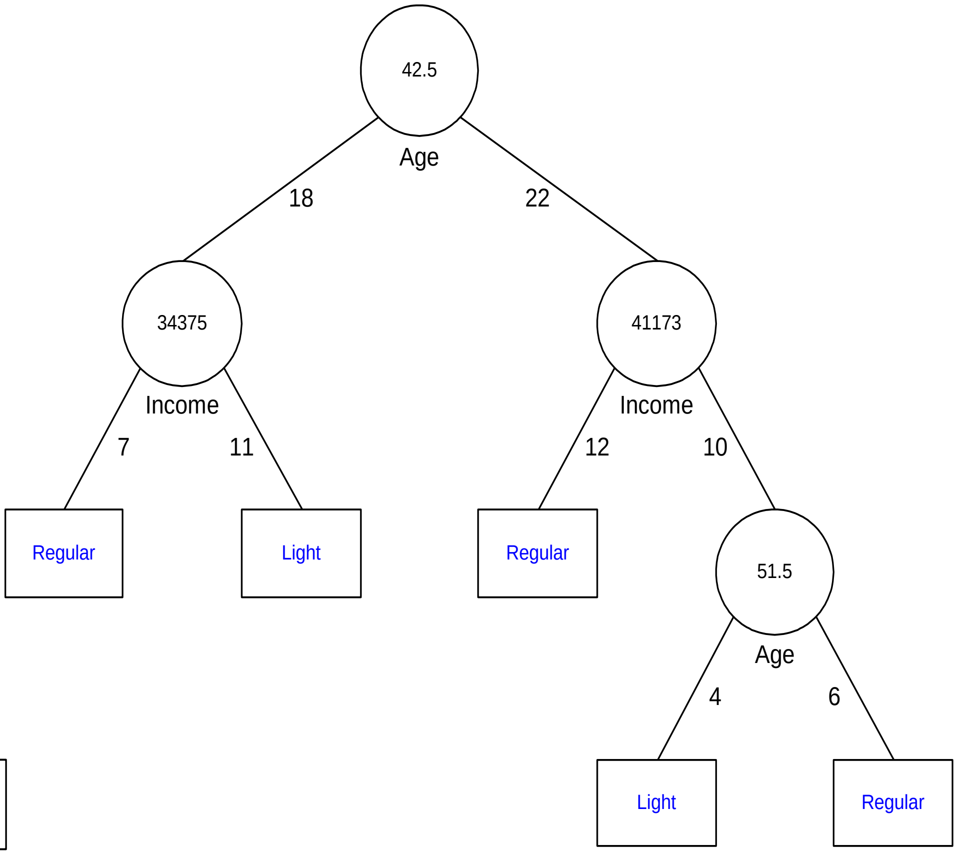
- Default: majority vote
- In 2-class case, majority vote = cutoff of 0.5
- Changing the cutoff will change the labeling
- Example:
  - success class=buyers
  - 0.75 means that to label the node as "buyer", at least 75% of the training set observations with that predictor combination should be buyers.

Beer Preference Example

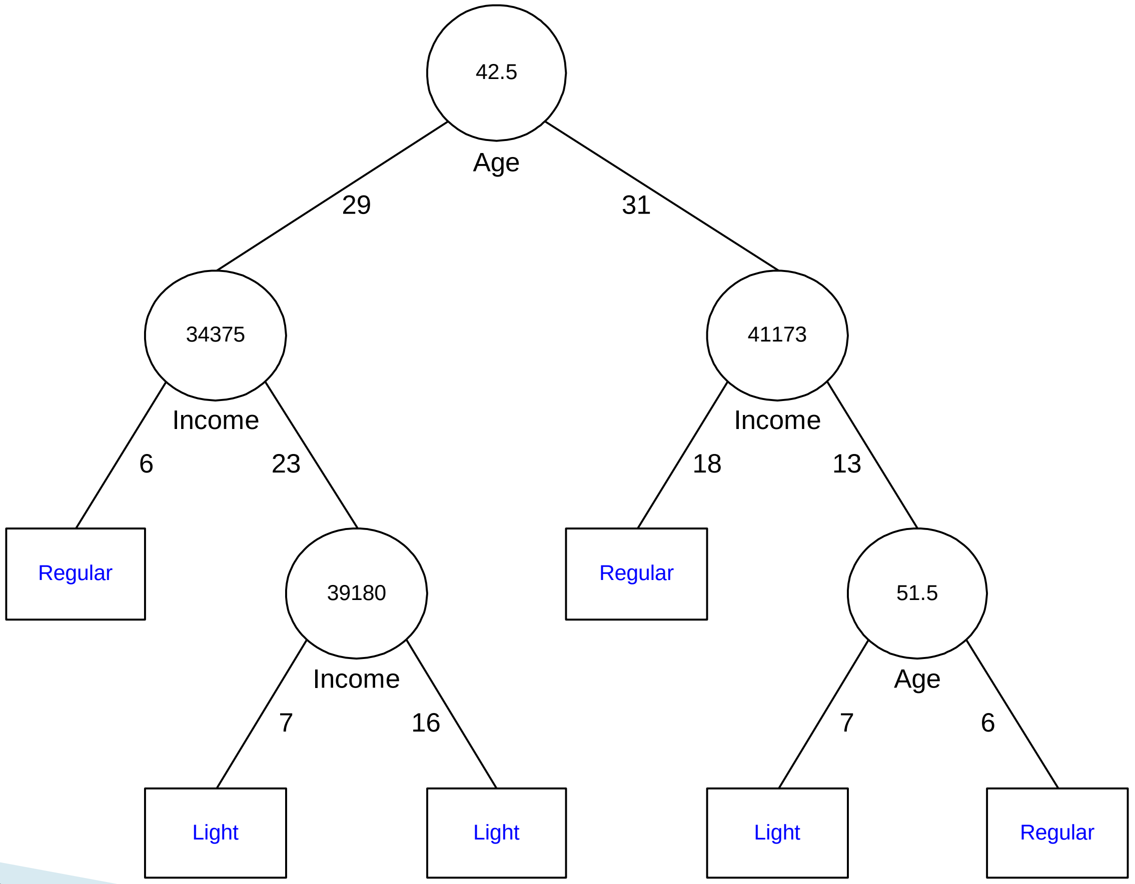
Full tree (training)



Pruned tree (validation)



Predict the preference of a 50-year old female with 50K income



CHAID

- Chi-Square Automatic Interaction Detector
- The CHAID algorithm tests a hypothesis regarding dependence between the split variable and the categorical response (using the chi-squared test for independence).
- Using a pre-specified significance level, if the test shows that the split variable and the response are independent, the algorithm stops the tree growth. Otherwise the split is created, and the next best split is searched
- No pruning is used. The algorithm stops on it's own

CHAID Algorithm

- **Merging:**

  –Merge the categories that have a p–value greater than alpha–to–merge using the Chi–Square Test For Independence

- **Splitting:**

  –Split the variable that has the most significant information gain at each stage of the process

CHAID Algorithm(continued)

- **Stopping**

  –If a node becomes pure; that is, all cases in a node have identical values of the dependent variable, the node will not be split.

  –If all cases in a node have identical values for each predictor, the node will not be split.

  –If the current tree depth reaches the user specified maximum tree depth limit value, the tree growing process will stop.

CHAID Algorithm(continued)

- If the size of a node is less than the user-specified minimum node size value, the node will not be split.

- If the split of a node results in a child node whose node size is less than the user specified minimum child node size value, child nodes that have too few cases (as compared with this minimum) will merge with the most similar child node as measured by the largest of the p-values. However, if the resulting number of child nodes is 1, the node will not be split.

Uses of CHAID

- Should be used primarily when the goal is to describe or understand the relationships between the response and predictor variables
- Not great for use in accurate prediction of response variable values
- Since there are multi-way splits, it makes CHAID an ideal candidate for Market Segmentation studies

Bonferroni Adjustment

- “Nothing repeats itself precisely”
- This natural fluctuation is the result of the interplay of several small variables and is called chance fluctuation, i.e., not traceable to any specific cause
- When multiple statistical tests are conducted in a particular study, the alpha level must be adjusted downward to take care of chance capitalization

- The alpha level is basically the probability of making a Type 1 error – accepting a false NULL hypothesis as true

- As the number of tests increases, this probability also increases

- We might now wrongly find 2 variables to be significant

- Adjusted p-value = Bonferroni Number * Original p-value

CART Classification And Regression Tree

- CART (Classification And Regression Tree) is one of the popular methods of building decision trees in the machine learning community
- CART builds a binary decision tree by splitting the records at each node, according to a function of a single attribute
- CART uses the gini index for determining the best split
- The initial split produces two nodes, each of which we now attempt to split in the same manner as the root node

CART

- At the end of the tree-growing process, every record of the training set has been assigned to some leaf of the full decision tree
- Each leaf can now be assigned a class and an error rate
- The error rate of a leaf node is the percentage of incorrect classification at that node
- The error rate (If an entire decision tree is a weighted sum of the error rates of all the leaves.
- Each leaf ‘s contribution to the total is the error rate at that leaf multiplied by the probability that a record will end up in there.

- Unlike CHAID, CART decides the best splits based on homogeneity within the split categories
- It checks for over-fitting only during the pruning phase
- We need a separate validation set in order to prune the tree
- Pruning continues till the tree behavior in the training and validation sets are similar
- It is ideal in cases where accurate prediction is required