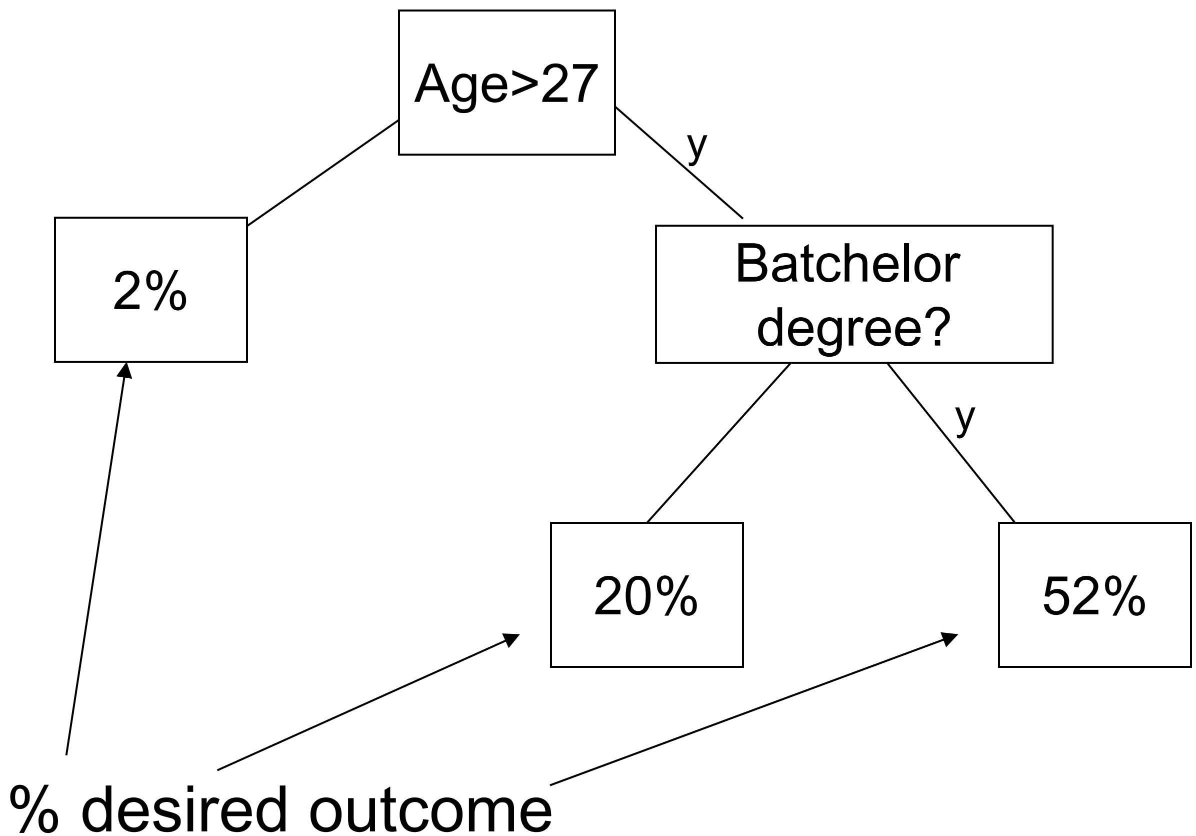
Decision Trees

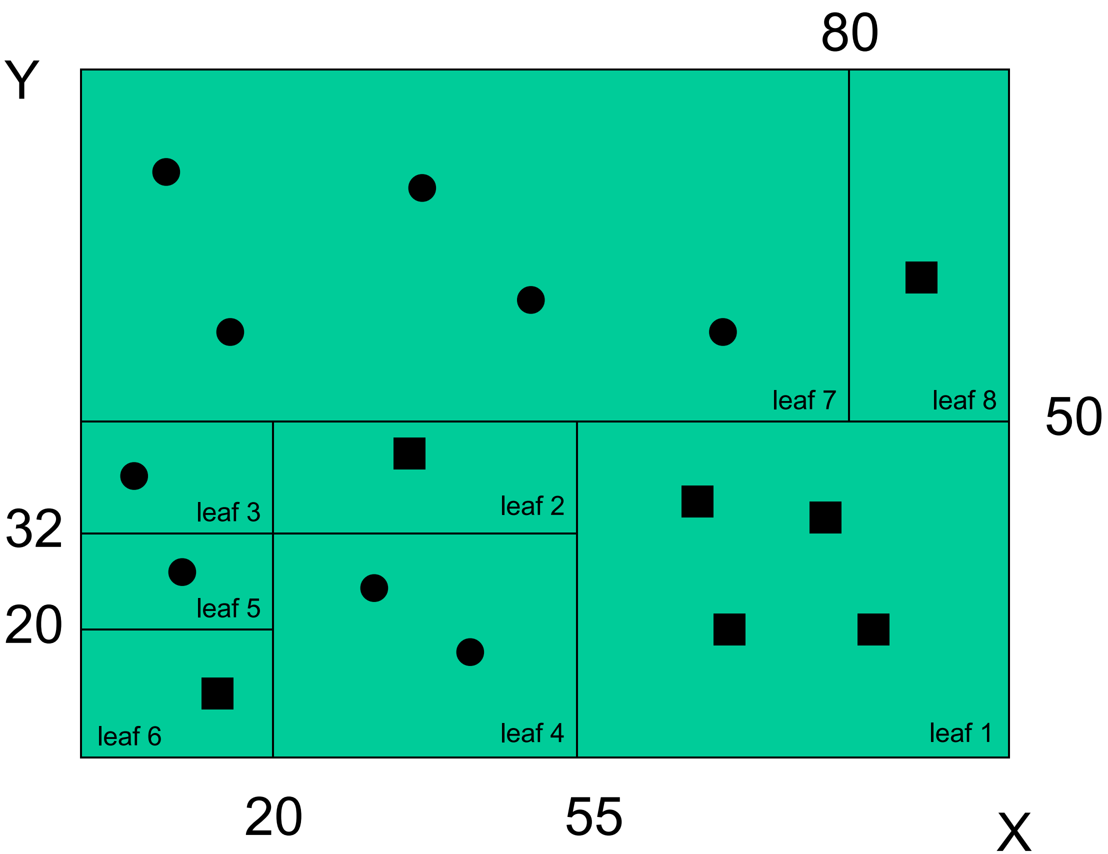


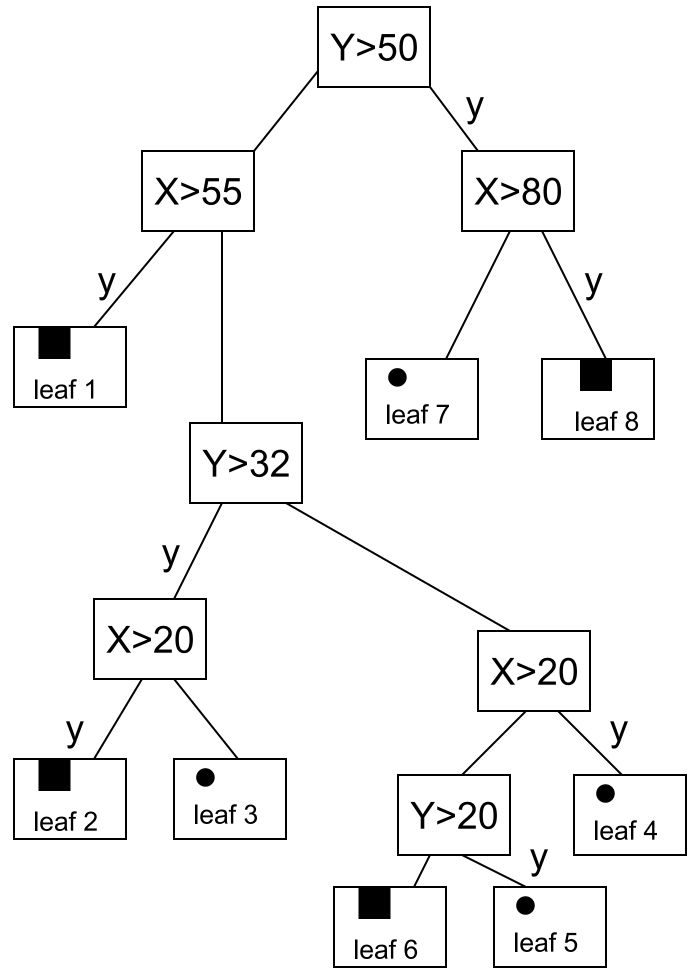
Desired outcome : high income

Decision trees

- Directed and clear box technique
- Can be employed for
  - classification (target values are discrete)
  - estimation (target values are continuous)
  - prediction (target values are in the future)
- Various algorithms are known, and tools provide the option to amend parameters of the generated tree
- Widely applied to directed mining

1. Decision trees - for classification





Built using a training set of pre-classified data

e.g., target = risk, ● = low-risk, ■ = high-risk

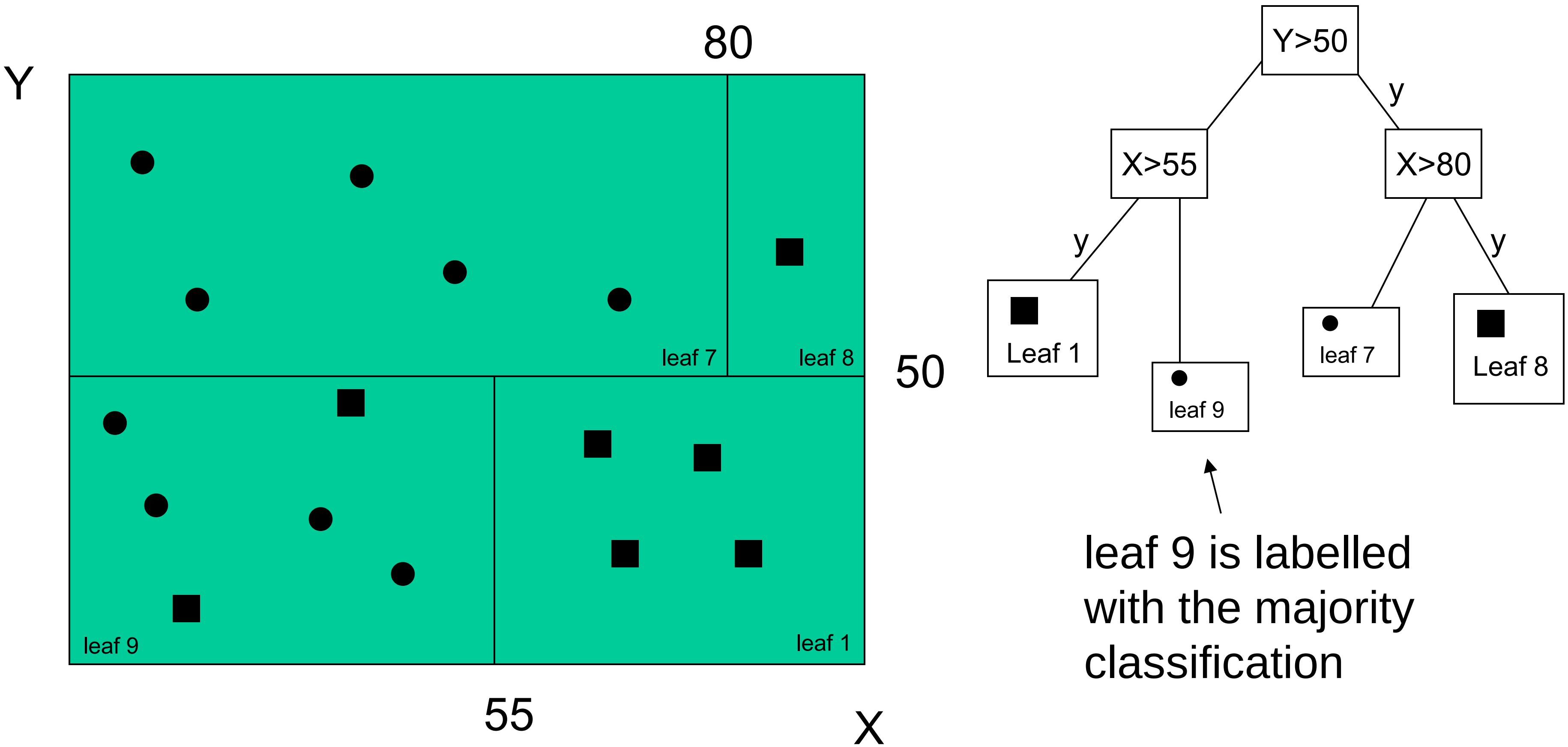
Applying the decision tree

- Having built the decision tree, we can then classify new individuals (for whom we don't know the correct classification) by using their attribute values to trace through the tree to a leaf node, e.g.,

  if $Y \leq 50$ and $X > 55$ then

  risk = ■ = high-risk

Applying a "non-full" decision tree



leaf 9 is labelled with the majority classification

if Y ≤ 50 and X ≤ 55 then risk = ● = low-risk

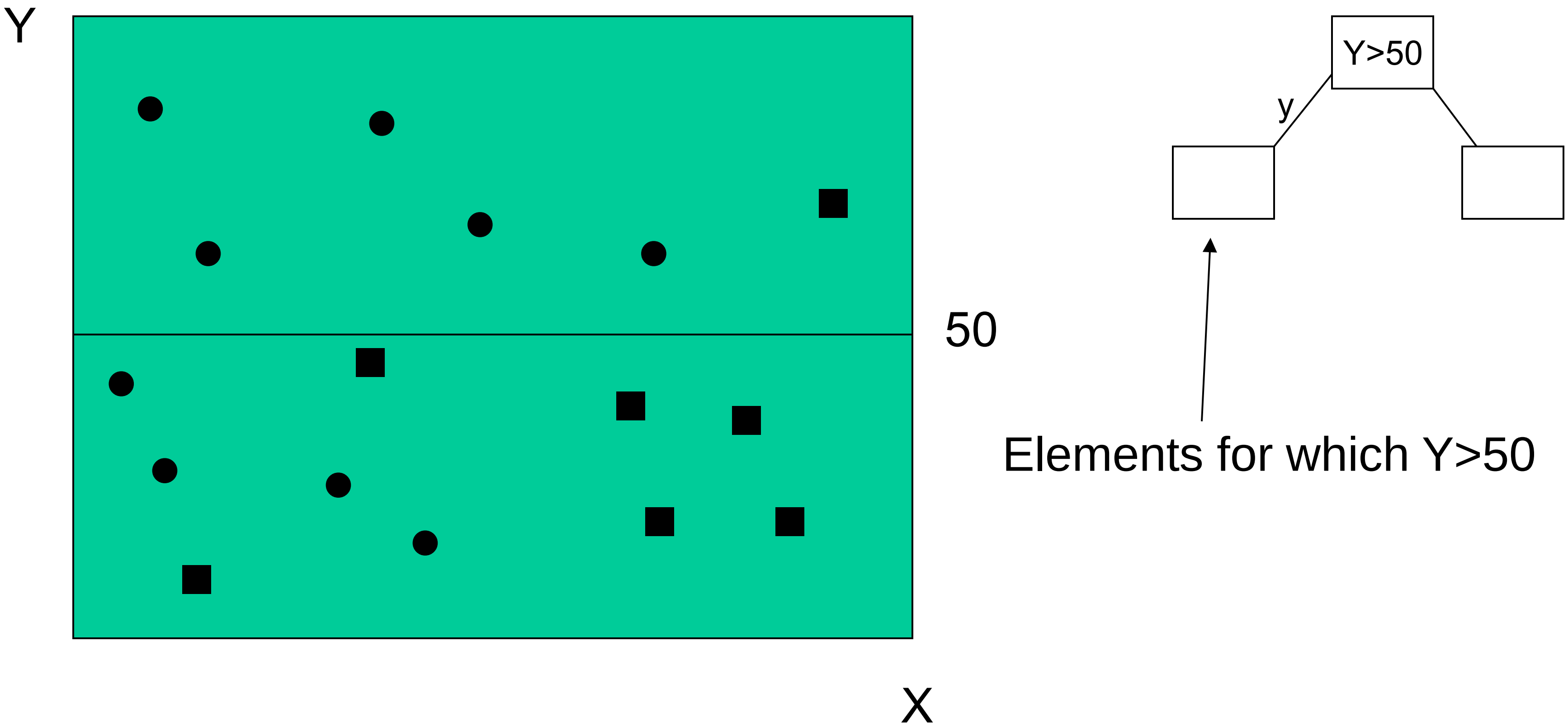
2 Building decision trees

- Each internal (non-leaf) node contains a splitting criteria
  - previous example is binary, i.e., each split results in 2 child nodes
- The previous example has two input variables (X and Y) and the classification is also binary (● and ■)
  - generally both can be >2

Building decision trees

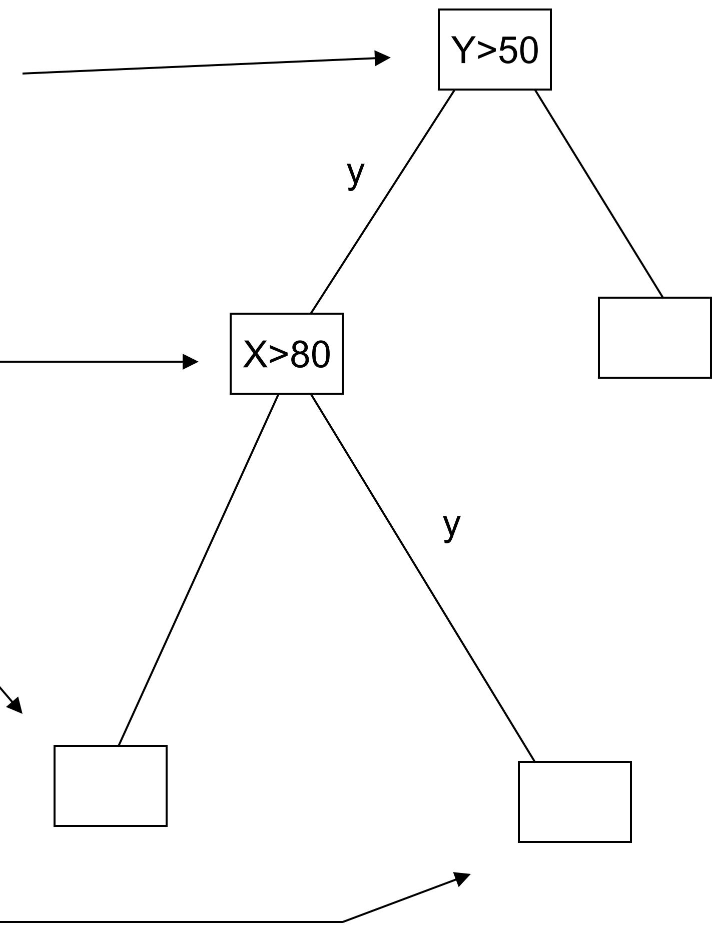
- Using a training set of pre-classified data (i.e., target value is known) & recursive partitioning to create subsets that are (more) homogeneous with respect to the target value
- Each node represents a subset of the training set
  - the root represents the entire training set

Each node represents a subset of the training set



Building decision trees

- Root - entire training set
- Elements for which Y > 50
- Elements for which Y>50 and X≤80
- Elements for which Y>50 and X>80

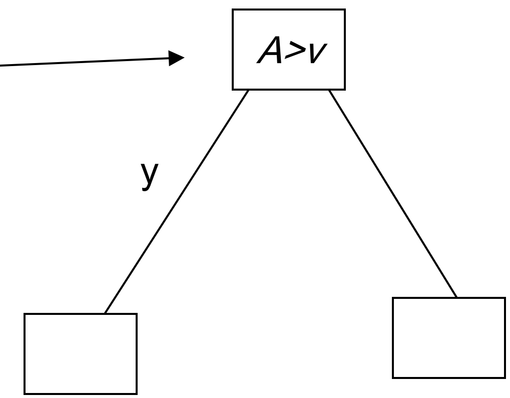


Building decision trees

- Suppose we have node ***N*** representing the subset ***S*** (of the training set)
- If ***N*** is to be split, we pick the input variable *A* and value *v* for *A* (which occurs in the training set) such that the resulting split produces the greatest reduction in diversity
- ***N*** is then labelled with the splitting rule
  - i.e., $A > v$

Building decision trees

- ***N*** representing ***S***
- Suppose we split with $A > v$ then the child nodes represent:
- S1 = elements in ***S*** for which $A > v$
- S2 = elements in ***S*** for which $A \leq v$



Diversity – the Gini Index

- Suppose we have a population P of size p (written |P| = p) in which each individual is either in class1 or class2 (but not both)
- Let p1 = # elements of P in class1, and

  p2 = # elements of P in class2

then the **Gini index** is given by

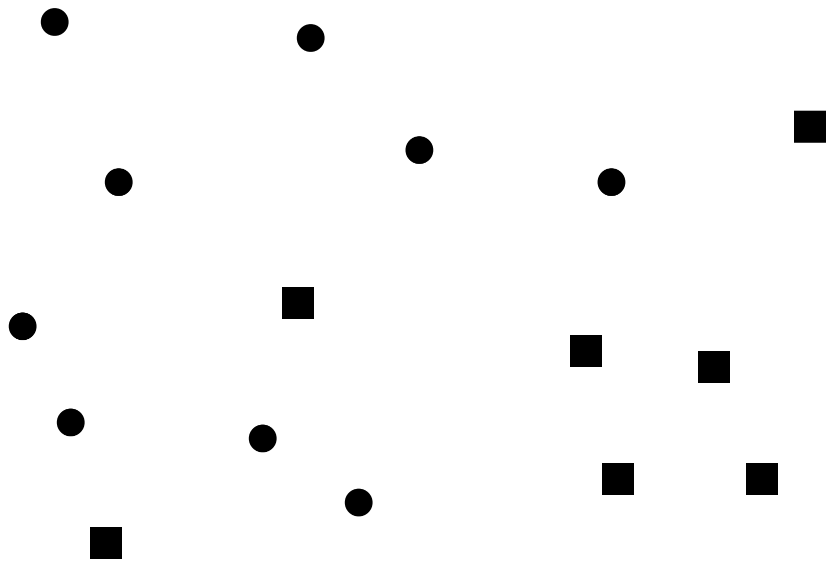
$$2*(p1/p)*(p2/p)$$

Gini Index

- When P is homogenous p1 or p2 equal 0 and hence so does the Gini index
  - homogeneous = zero diversity
- When P is entirely diverse, both p1 and p2 equal p/2 (where p = |P|) in which case the Gini index =

$$2*(1/2)*(1/2) = 1/2$$

Gini index - example



Squares = 7

Circles = 9

Gini = 2*(7/16)*(9/16)

= 0.49

--- almost completely diverse

The Gini Index - intuition

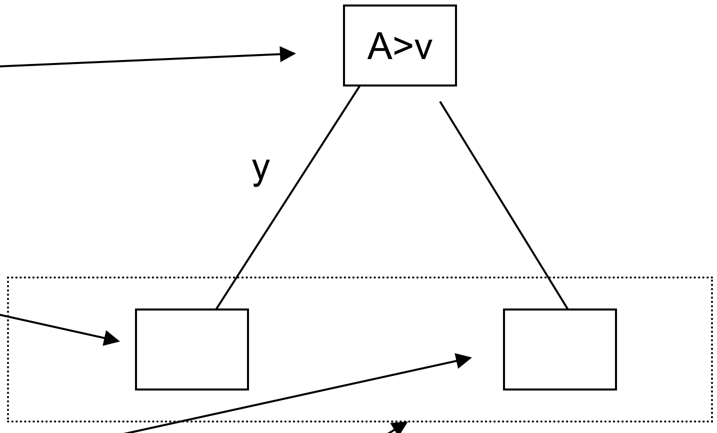
A S I D E

- Suppose we point (at random) to an individual from P; then point to a second individual.

  The Gini index is the probability that the second individual will be in a different class to the first
  - the probability we point to a member of class1, then a member of class2 is (p1/p)*(p2/p)
  - ditto for the converse order
  - then add the two probabilities
- The higher the probability, the higher the diversity

For mutually exclusive events A and B, prob(A or B) = prob(A) + prob(B)

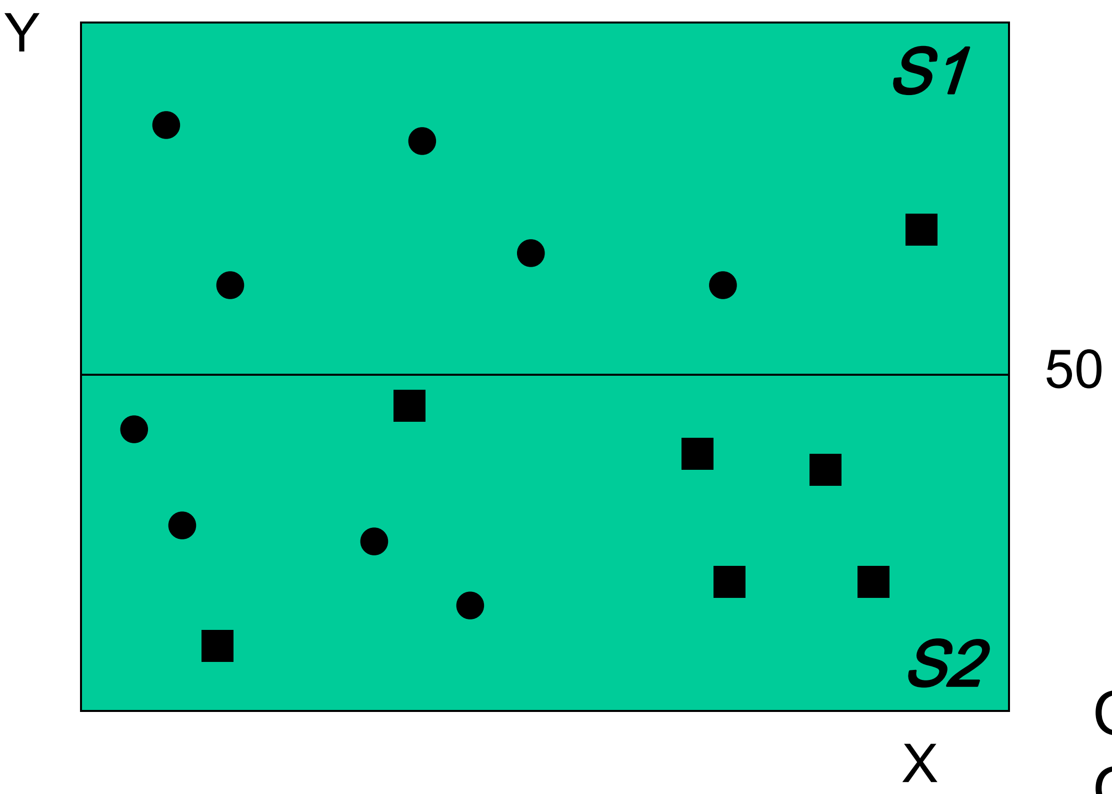
Reduction in diversity

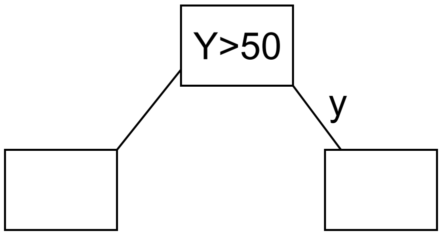


- ***N*** representing ***S***
- S1 = elements in ***S*** for which $A > v$
- S2 = elements in ***S*** for which $A \leq v$
- The diversity after the split is given by

$$\frac{\text{Gini}(S1)*|S1| + \text{Gini}(S2)*|S2|}{|S1| + |S2|}$$

Computing the reduction in diversity



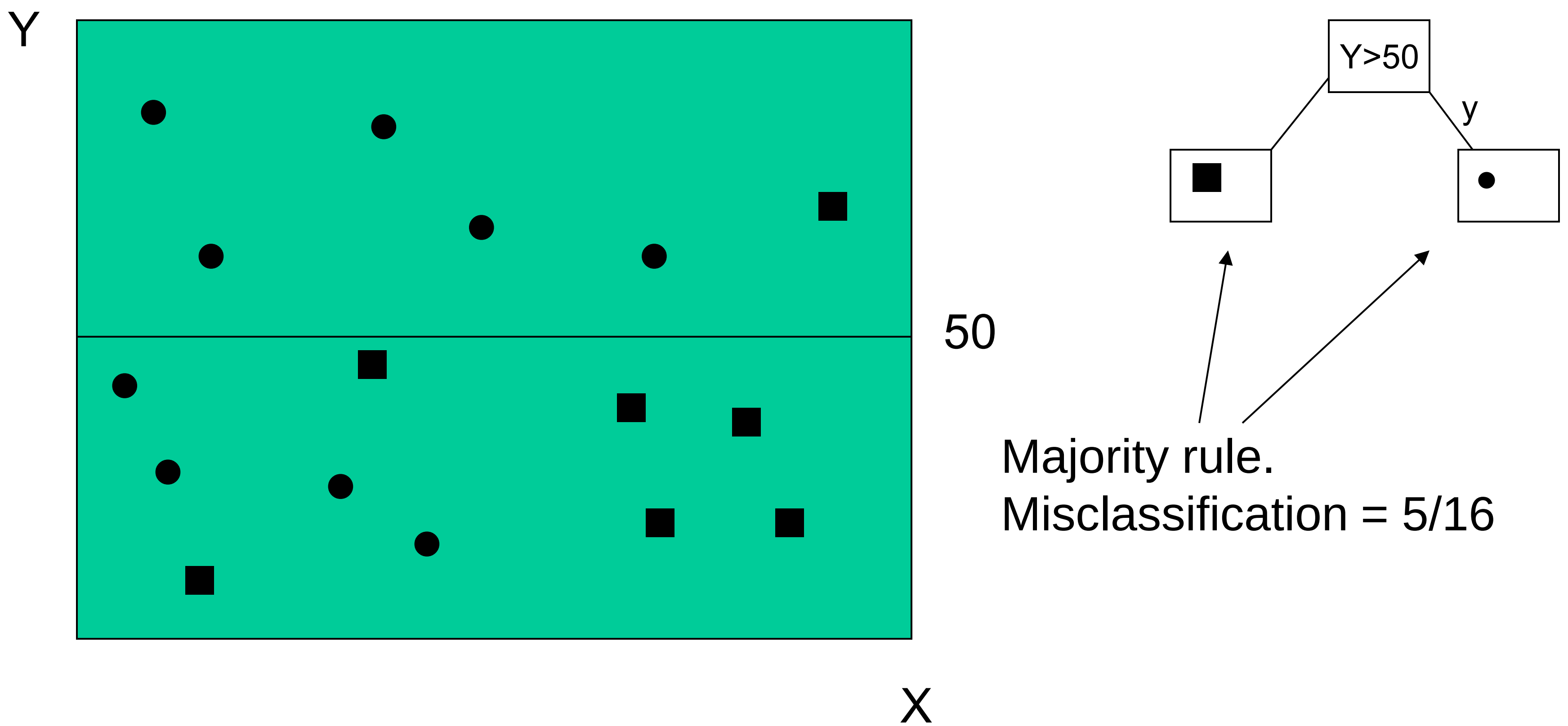


Gini(S1) = $2*(5/6)*(1/6)$

Gini(S2) = $2*(6/10)*(4/10)$

$$\text{Gini after split} = \frac{(10/36)*6 + (48/100)*10}{16} = 0.404$$

Other metrics for driving the partitioning: Misclassification

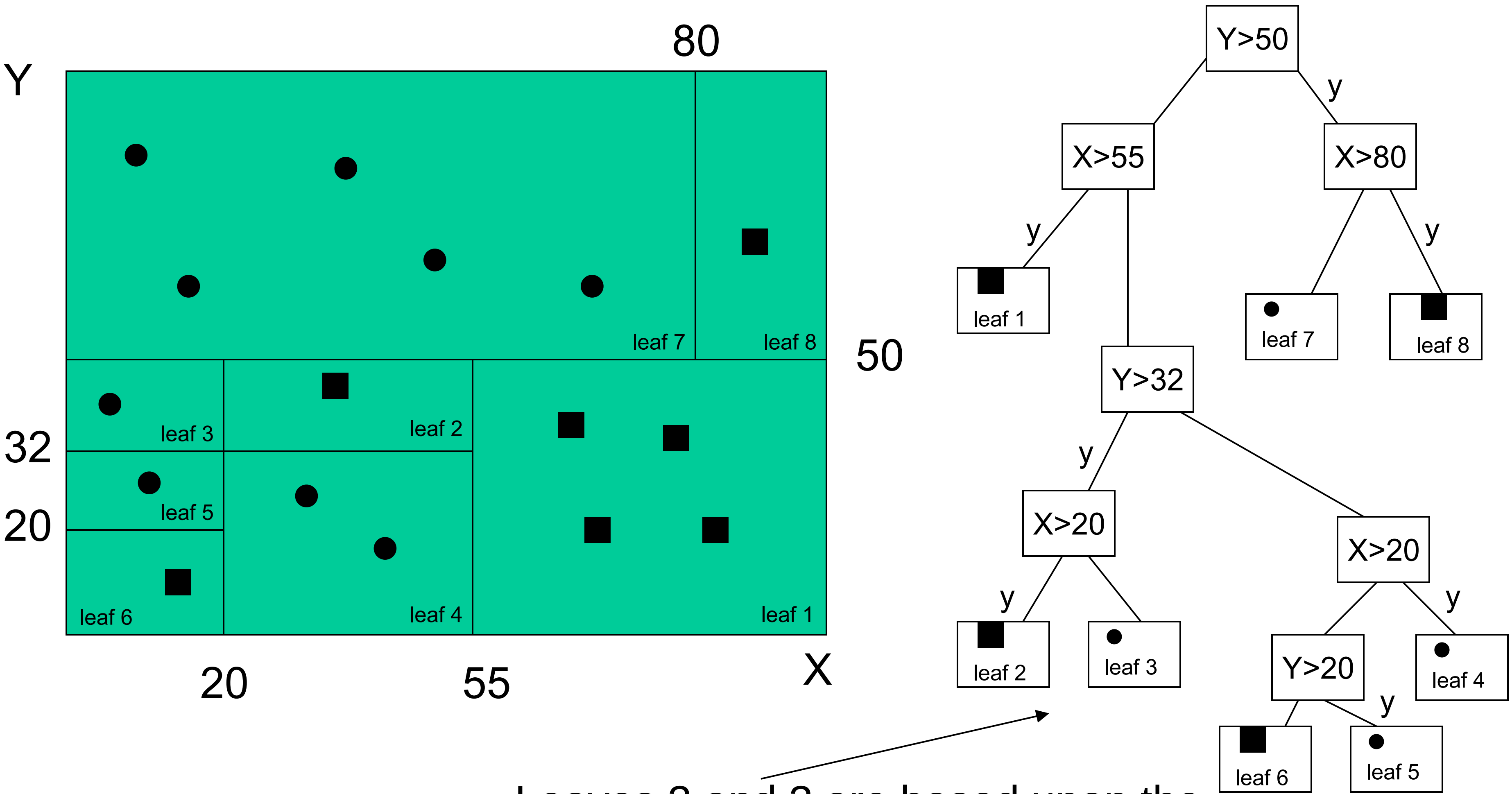


Majority rule.
Misclassification = 5/16

3 Overfitting

- The "full tree" is constructed as above by "recursive partitioning" - until we get purity
- The full tree is likely to **overfit** the training set since
  - splits near the top of the tree tend to reflect fundamental properties of the population
  - splits near the bottom tend to pick up the idiosyncrasies of the small number of particular individuals in the training set at that node -- and so do not generalise – giving poor performance when applied to the wider population

The lower levels of the tree pick up idiosyncracies



Leaves 2 and 3 are based upon the differences between 2 individuals

Bonsai techniques

Stunt the growth of the tree, e.g., by

- limiting the number of leaves or levels
- insisting that each node represents > X individuals . . . and is therefore (to some extent) representative of the underlying population
- imposing a minimum threshold on the required reduction in diversity

Pruning using a loss function

Construct the full-tree & then prune progressively

& evaluate performance against a "loss function"

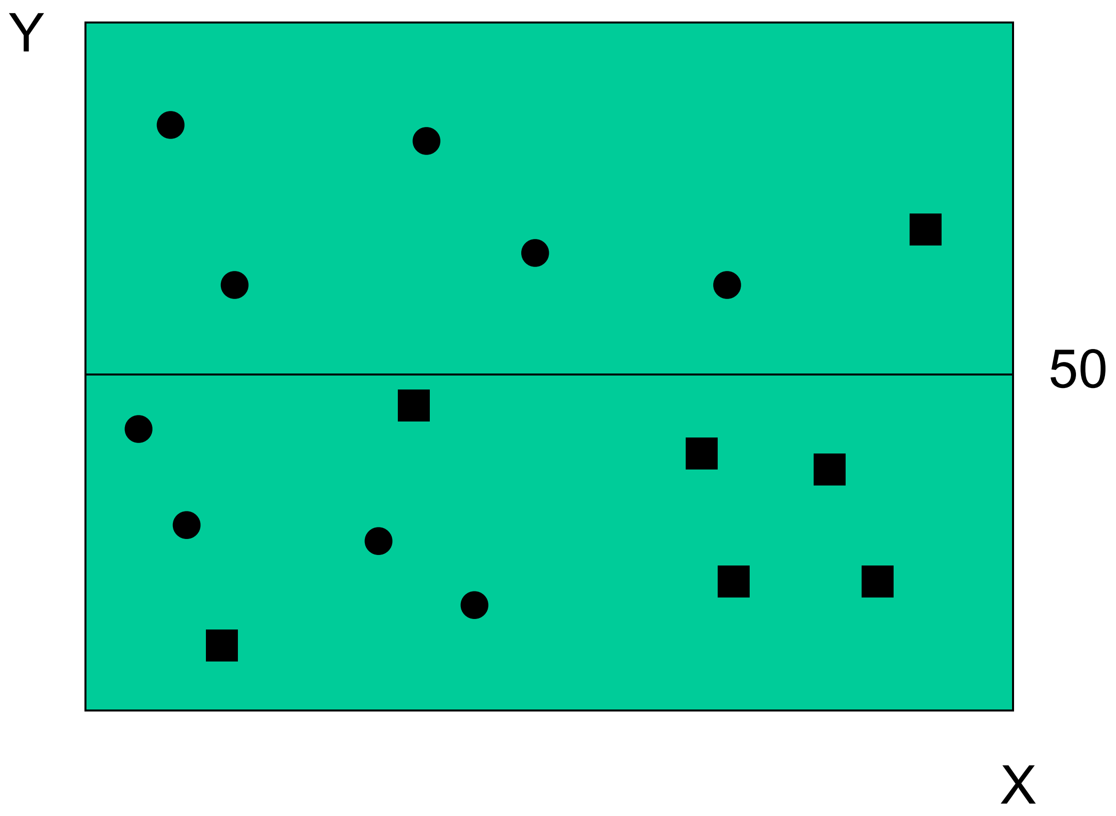
$$\text{Error}(T) + \alpha \times N(T)$$

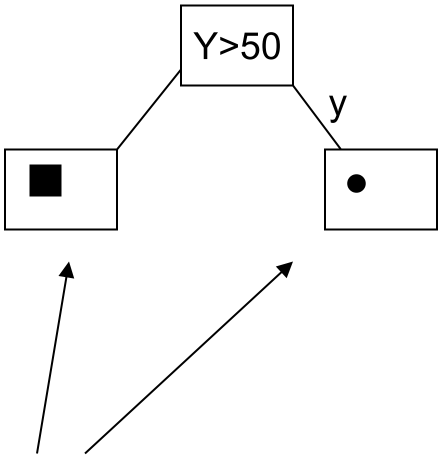
where

- Error(T) is the error (mis-classification) rate of T when applied to separate "test set" of data
- N(T) is the number of leaf nodes of T, and
- α is a parameter to be set by the miner

- Other loss functions can be used

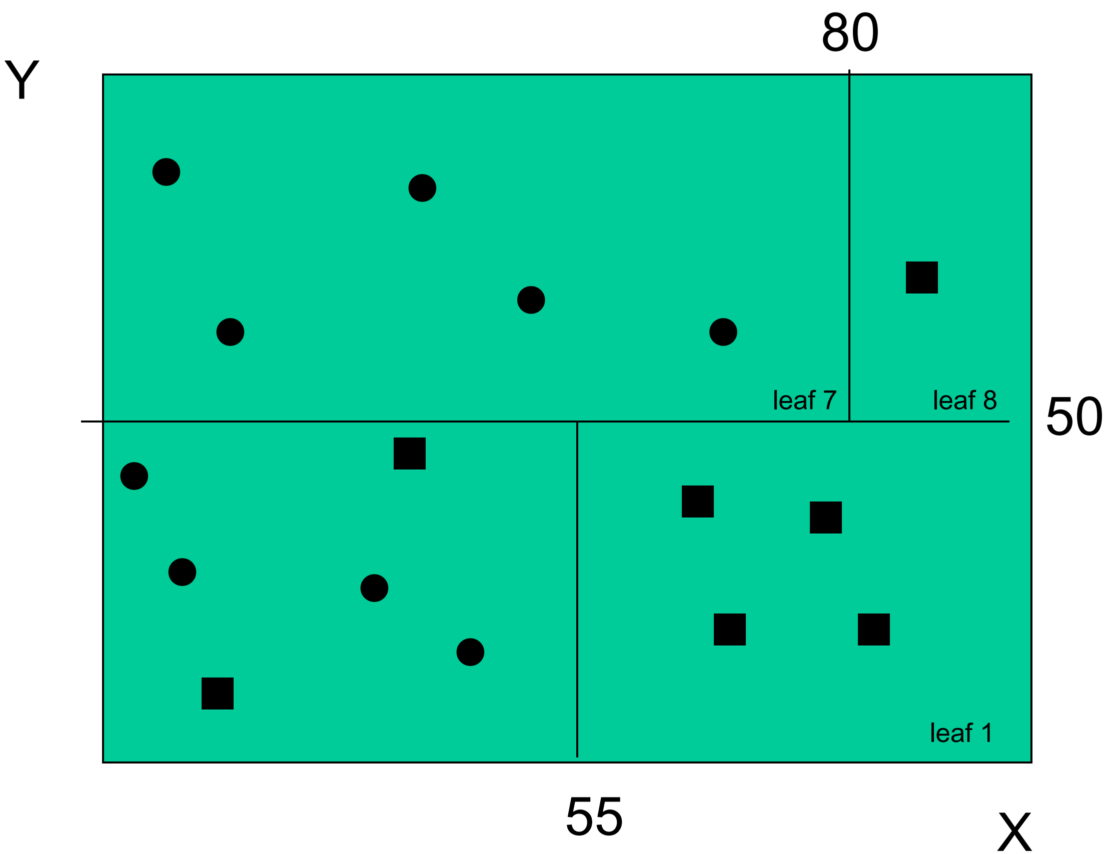
Error rate on original training set

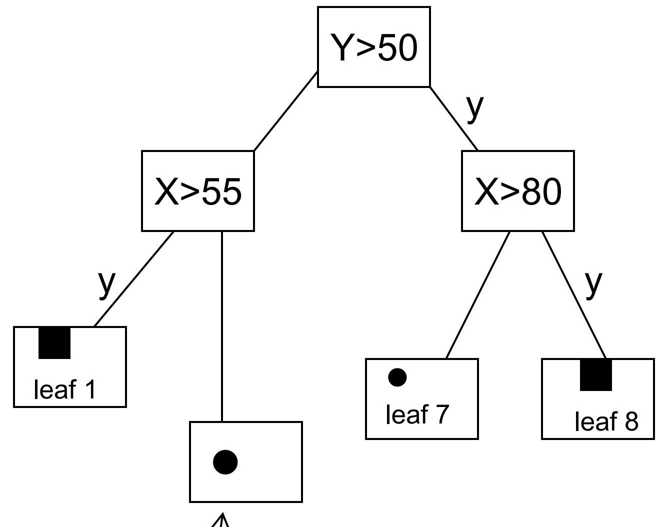




Majority rule.
Misclassification = 5/16

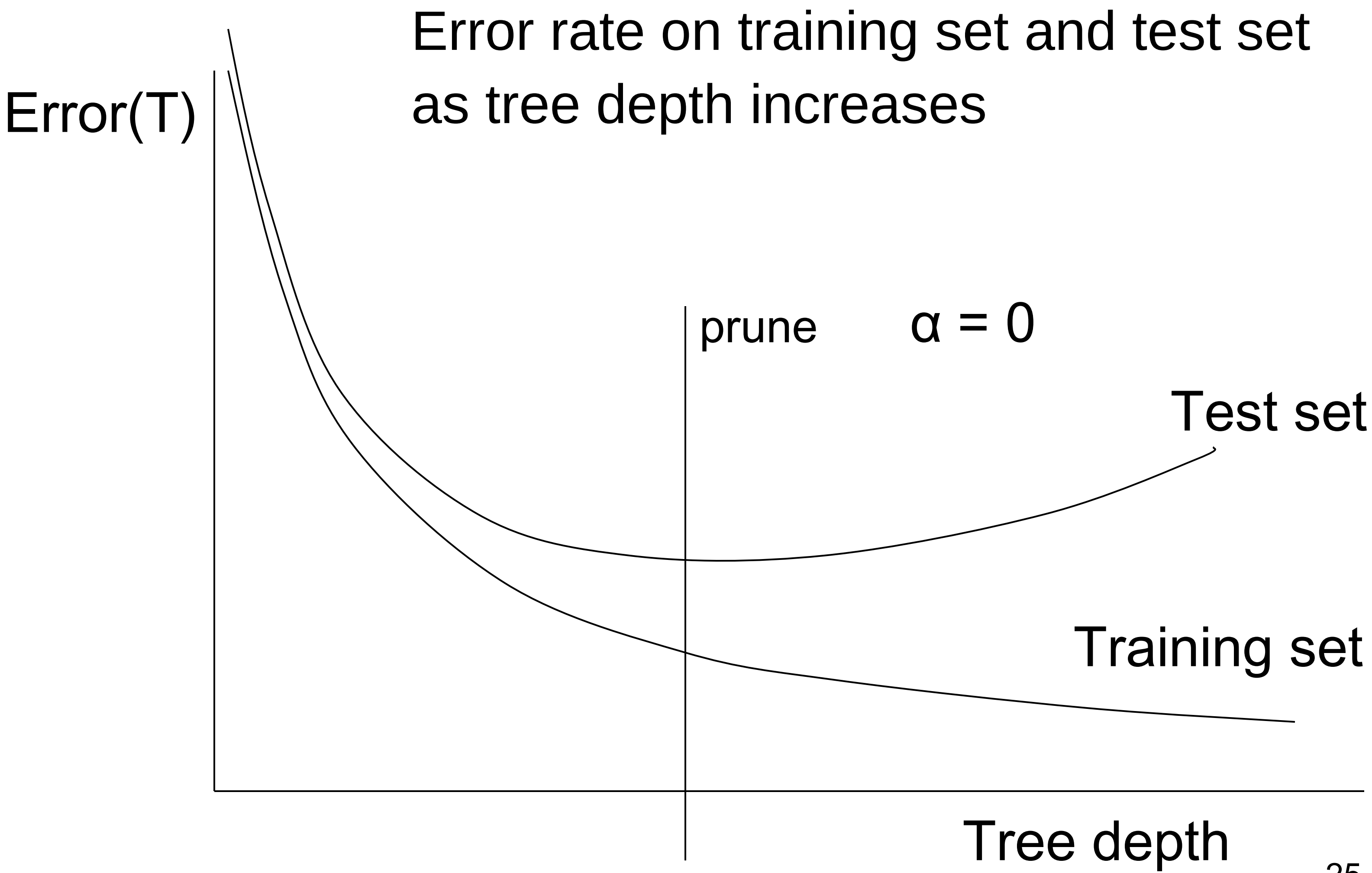
Error rate on original training set



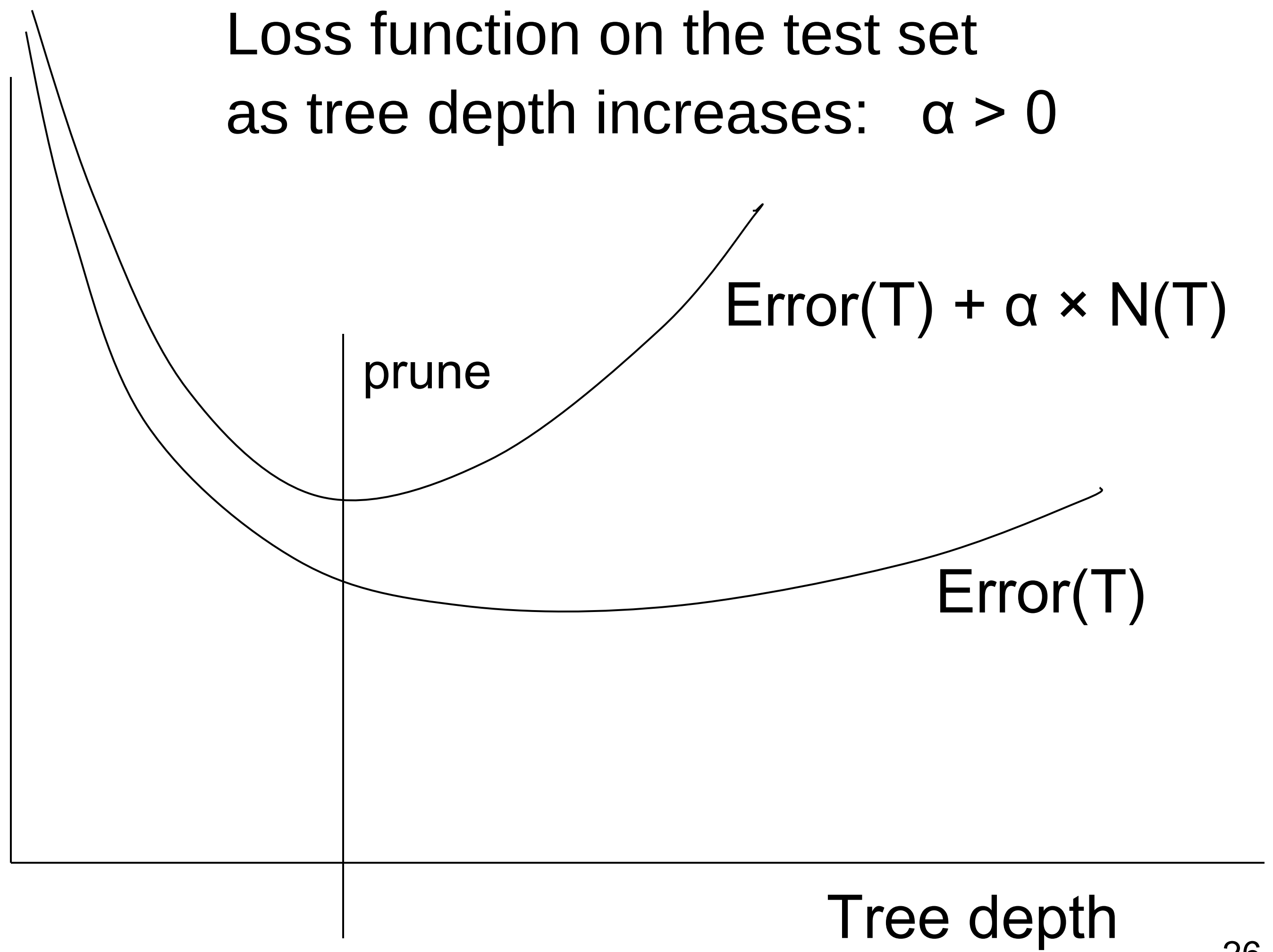


Majority rule.
Misclassification = 2/16

Pruning using a separate test set



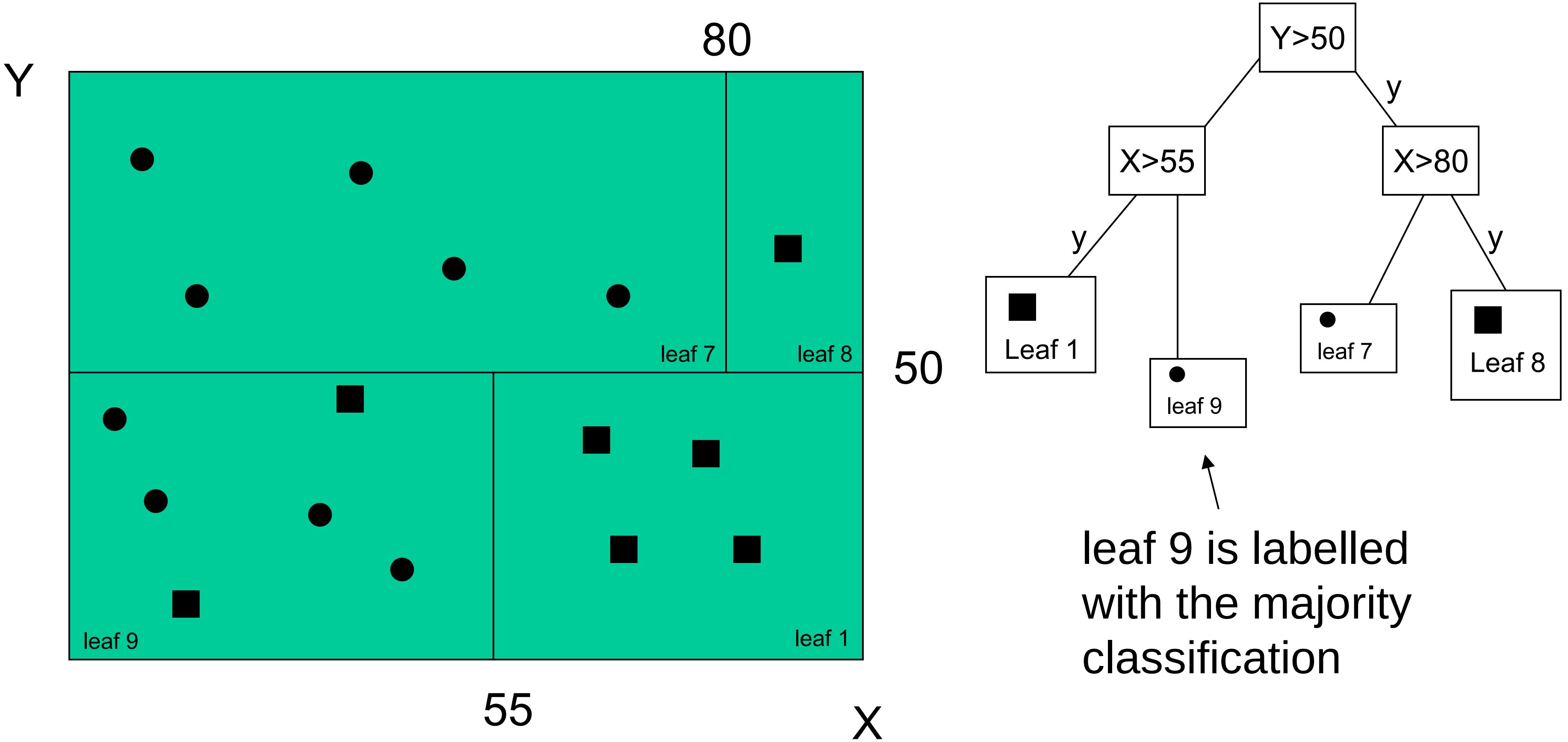
Pruning using a separate test set



The model set

- "Training set" - used to build first cut model
- Refined using "test set" - overfitting
- Performance then evaluated using "evaluation set"
- Training, test & evaluation sets are disjoint
- Model set = training set + test set + evaluation set

Applying a “non-full” decision tree



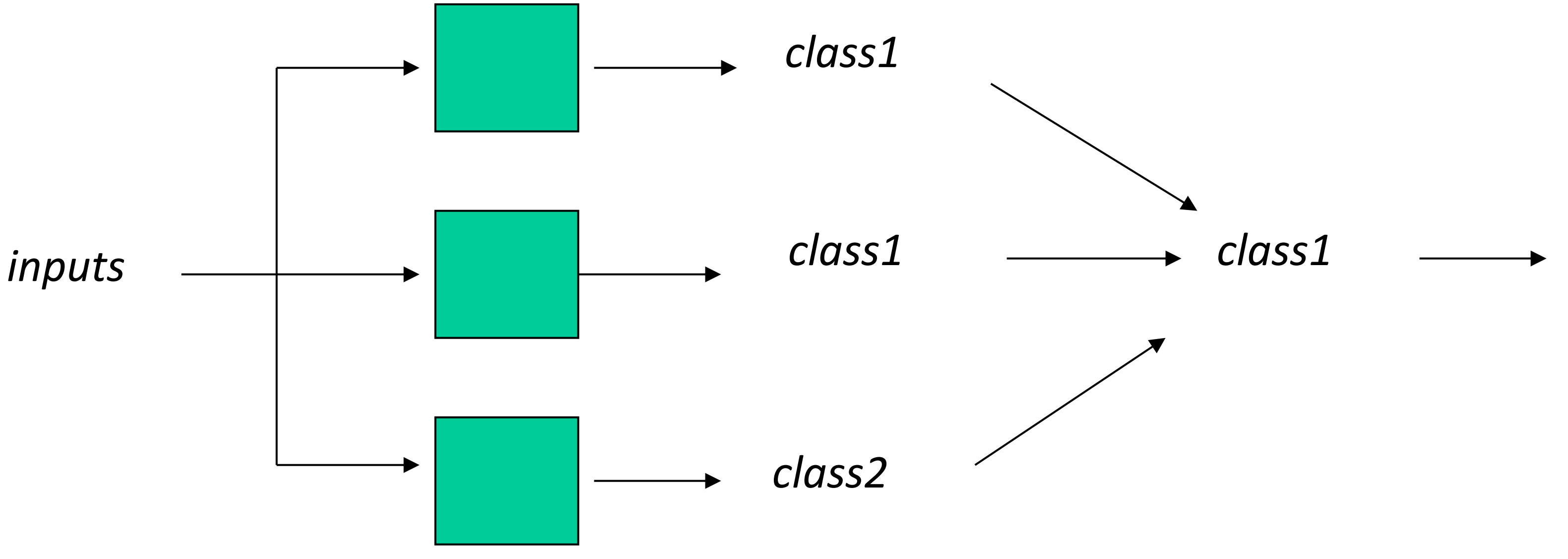
leaf 9 is labelled with the majority classification

if Y ≤ 50 and X ≤ 55 then risk = ● = low-risk

ASIDE

Enhancing performance

We may produce a number of models with similar performance … we can try to improve overall performance by merging them using voting



For estimation we might take the average

4 Decision trees for estimation

- Here we have a target variable f whose value we wish to estimate based upon input variables $x_1, x_2, \ldots, x_n$
- Training set: value of f already known
- Diversity of a (sub) population P is defined by the variance

$$\text{variance } \{f(i) : i \text{ in } P\}$$

Variance

A
S
I
D
E

If we have a set of values $\{a_1, a_2, \ldots, a_m\}$ with average

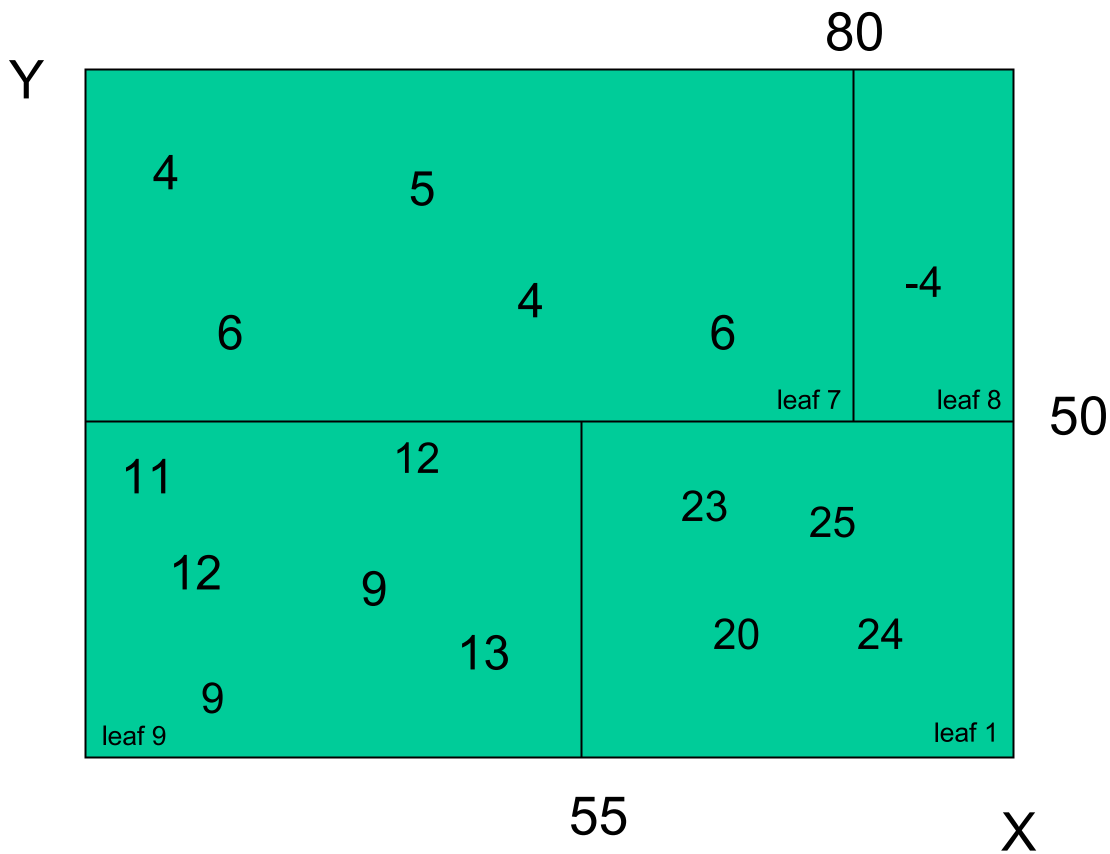
$$a = (a_1 + a_2 + \ldots + a_m)/m$$

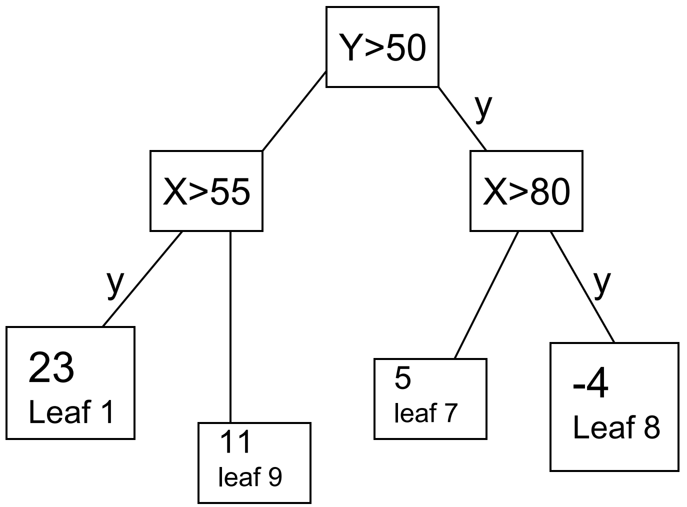
then the variance is given by

$$\frac{(a_1 - a)^2 + (a_2 - a)^2 + \ldots + (a_m - a)^2}{m}$$

and is a measure of the *spread* of the values

Applying a decision tree for estimation





each leaf is labelled with the average f value

if Y ≤ 50 and X ≤ 55 then estimated score = 11

Applying the decision tree for estimation

- Each leaf node ***N*** is labelled with the average f - value of those elements of the training set associated with ***N***
- Each new individual can be associated with a leaf node as previously
  - The label of the leaf node is then the estimated f - value for the new individual

5 Building decision trees

- Needs sufficient data in the training set: 10000+ records (preferably)
  - overfitting is easier on smaller training sets
  - bigger model sets take longer to work with
  - the tool may impose limitations
  - model parameters need tuning based upon the size of the model set
    - e.g., minimum node size in Bonsai

Derived variables

- Every split is based upon a single variable
  - decision trees do not discover relationships *between* variables
  - to facilitate this we need to add *derived variables*
    - e.g., outstanding_debt / initial_loan
  - Derived variables need to be based upon a knowledge of the business and intended business aims

The input data

- Input data must be ordered but the approach is not sensitive to outliers, skewed distributions, differences in scale between different inputs
  - decision trees are generally applicable
  - severely skewed distributions however may be problematic
- But: decision trees are not necessarily stable to changes in the input (training) data
  - trees do not generalise
  - business confidence

Handling categorical data

- Values are categories, e.g. post-code
- Categorical variables need special handling since they generally have no natural ordering. Must be careful not to employ spurious ordering:
  - Such an ordering might be assumed by the miner: e.g., …, Cornwall, Cumbria, …
  - Tools may represent categoricals as an integer: must ensure that we don’t then inadvertently employ the ordering of the integers

Handling categorical variables

- Can flatten: employ flag variable for each possible value - but maybe lots of them!
- Split on every value: *A = v1* & *A = v2* & *A = v3* & …
  - Clearly overfitting is possible
- Employ binary split: *A = v* and *A ≠ v*
  - Clearly overfitting is a possibility
- For a binary target classification – say responders and non-responders
  - Employ a binary split of the form "*A in S*" and "*A not in S*" – where S contains those values of A that are correlated with a high proportion of responders

Unary column data

- Columns with only one value may be deleted … obviously
- Columns with almost only one value
  - overfitting again
  - before we ignore
    - why is the data so skewed?
    - does aggregation help?
    - is the subset containing the non-unary value of interest?

Handling null data

- Nulls may occur because data was not entered or provided; particularly a problem when using external source
- Delete such records with care
- Can treat as a categorical value
  - split via: A = null and A ≠ null
  - but if a column has mostly nulls then you can get overfitting

Binning the input data

- Binning (e.g., salary → salary range)
  - <15K, 15K-30K, 30K-100K, 100K-250K, >250K
  - some algorithms/tools bin (automatically)
    - leading to more efficient processing but a potential loss of information
    - useful for handling outliers and skewed distributions
    - Could also use quartiles/deciles/percentiles
- Salary range is an example of a *rank* variable
  - categorical with a natural ordering
- Cannot do arithmetic

6 The results of a decision tree

- Classification trees can also yield probabilities
  - Suppose a leaf node contains 20 responders and 30 non-responders: the node is classified as a "non-responder" but the fact that the proportion of responders is high might certainly be of interest

Decision trees yield rules and explanations

- Clear box
- Decision trees can in theory yield rules
  - if X > 50 and Y< 60 then class = C
  - a form of "lookup"
  - but many long rules are not easily understandable … suggesting greater pruning

Decision trees yield local information

- For example the first split divides the data in two and then builds a sub-tree (a model) for each of the two halves separately (enabling it to capture differing behaviours in differing parts of the dataset)
- Different rules can yield the same classification reflecting the realities of the situation: e.g., there can be many ways that a customer is profitable
    - Another example of local information

The results of a decision tree

- Identifies variables that are the best determiners of a classification
  - This information can be useful generally and also to other data mining techniques
- Brings to light trivial/known dependencies
  - E.g., “people under 27 don’t have high incomes”
- May find existing business rules
  - Trees are usually impure … but when existing rules are present/found this may not be the case

Decision trees for preliminary analysis

- We can easily build quick and dirty decision trees early on to get a feel for the data
  - Small pruned trees where we are not overly concerned about predictive accuracy
  - Efficiency provided by binning (at the cost of predictive accuracy)
  - Easy to handle many data types so no heavy demands on pre-processing
  - Insight provided by rules and most significant determiners
  - Brings to light local dependencies