innovate achieve lead

Machine Learning

Dr. Monali Mavani

BITS Pilani
Pilani Campus

Machine Learning

innovate achieve lead

Disclaimer and Acknowledgement

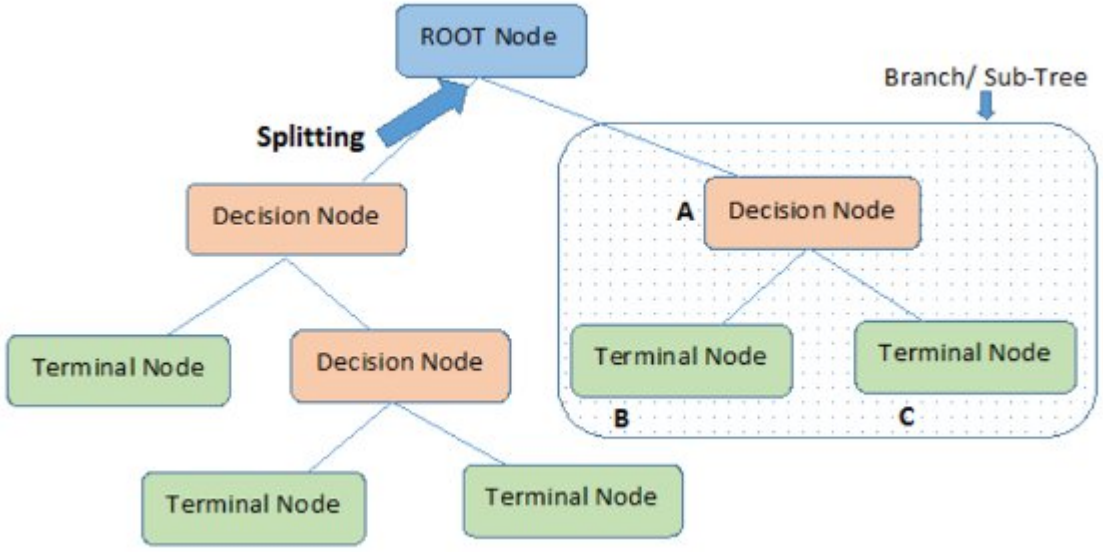
- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

BITS Pilani

Session Content

Decision Tree

Decision Tree



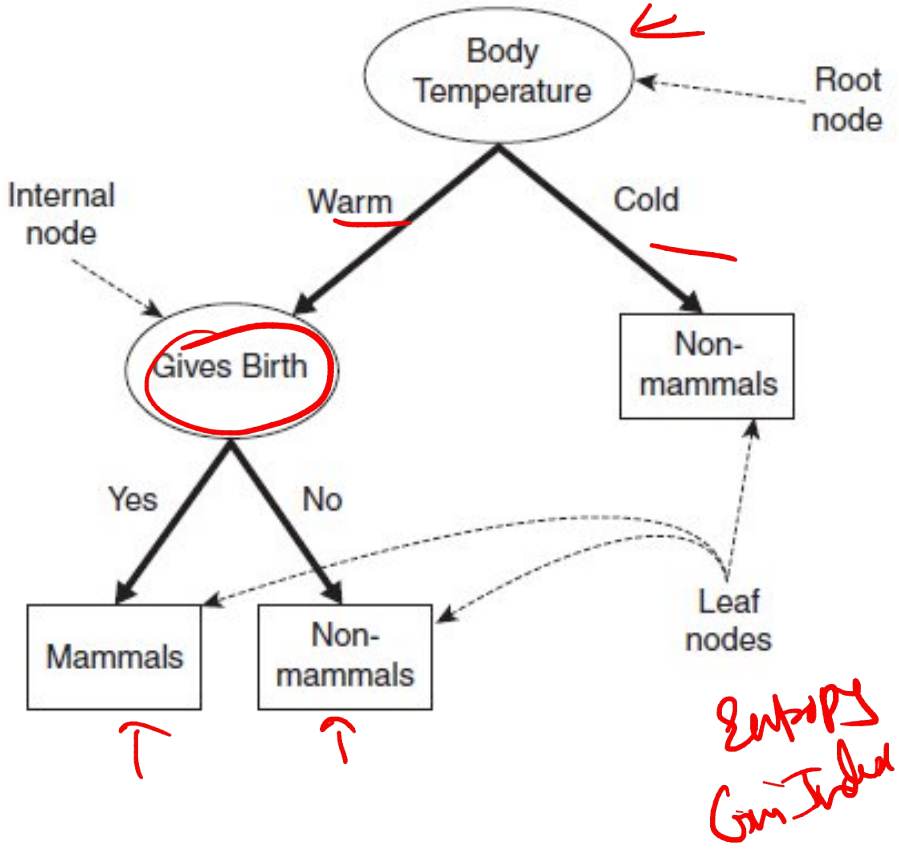
- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers

Decision Tree :Mammal Classification Problem

Vertebrate Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Decision Tree :Mammal Classification Problem



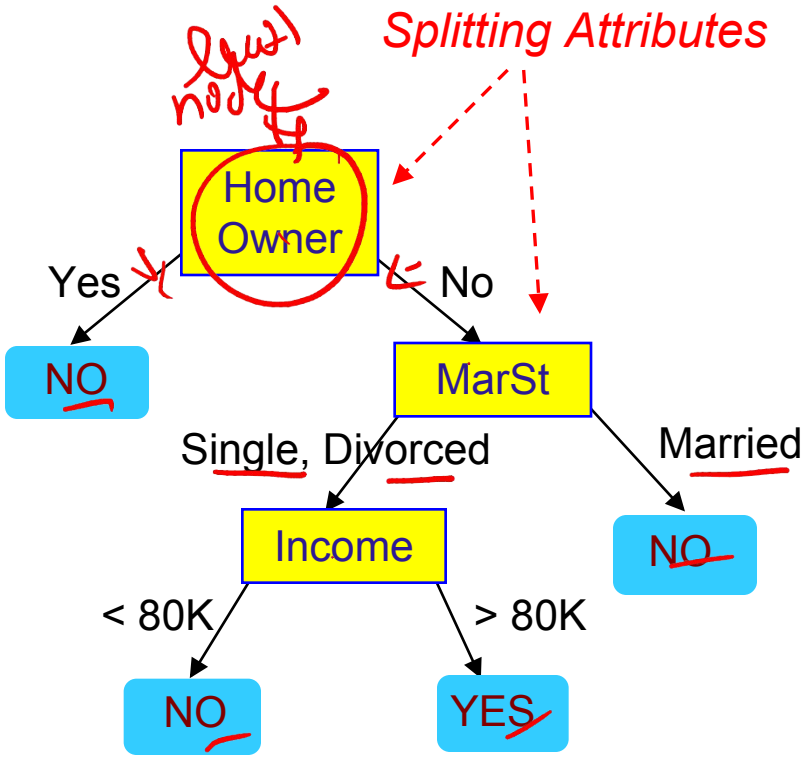
- Recursive Partitioning: each node is divided into child nodes, and this process continues until a stopping criterion is met.
- Homogeneity :homogeneous subgroups in each node, i.e samples within a node are as similar as possible regarding the target variable.
- Top-Down Greedy Approach: each split is chosen to **maximize information gain or minimize impurity** at the current node.
- May not always result in the globally optimal tree.

Decision Tree - example

- At start, all the training examples are at the root
- Partition examples recursively based on selected attributes

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

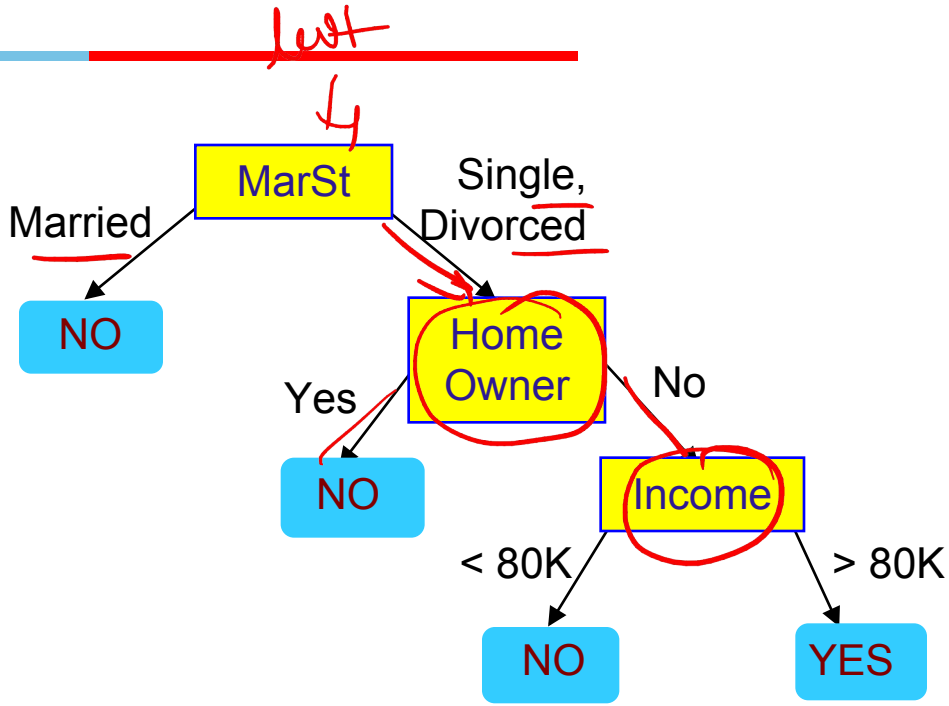
Training Data



Model: Decision Tree

Another Decision Tree same data

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



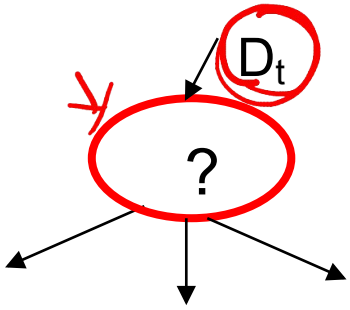
There could be more than one tree that fits the same data!

Decision Tree Induction

- Many Algorithms:
  - **Hunt's Algorithm (one of the earliest)**
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

Hunt's Algorithm

- A decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets.
  - D_t = set of training records that are associated with node t
  - $y = \{y_1, y_2, \ldots, y_c\}$ class labels

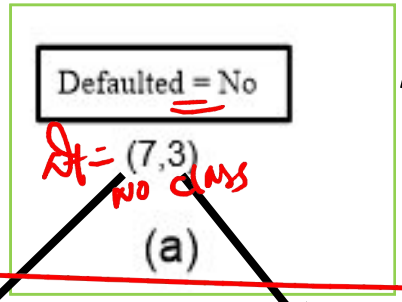


ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- General Procedure:
  - If D_t is an empty set, then t is a leaf node labeled by the default class y_t
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset

Decision tree construction example

- The initial tree for the classification problem contains a single node with class label Defaulted = No (most of the borrowers successfully repaid their loans)

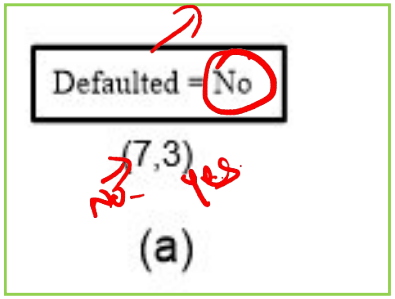


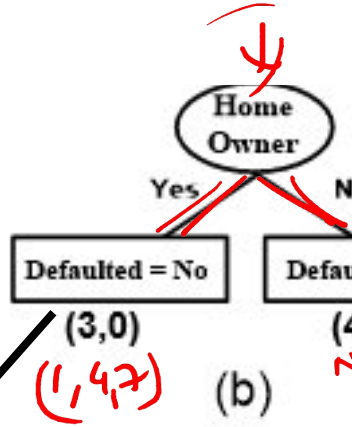
If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
6	No	Married	60K	No
7	Yes	Divorced	220K	No
9	No	Married	75K	No

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

Contd..





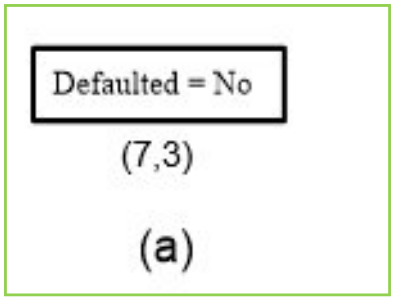
If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t

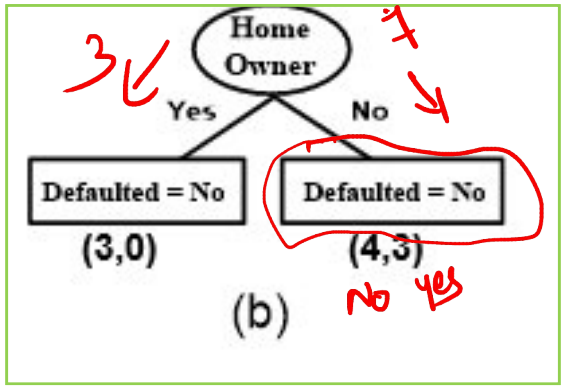
If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

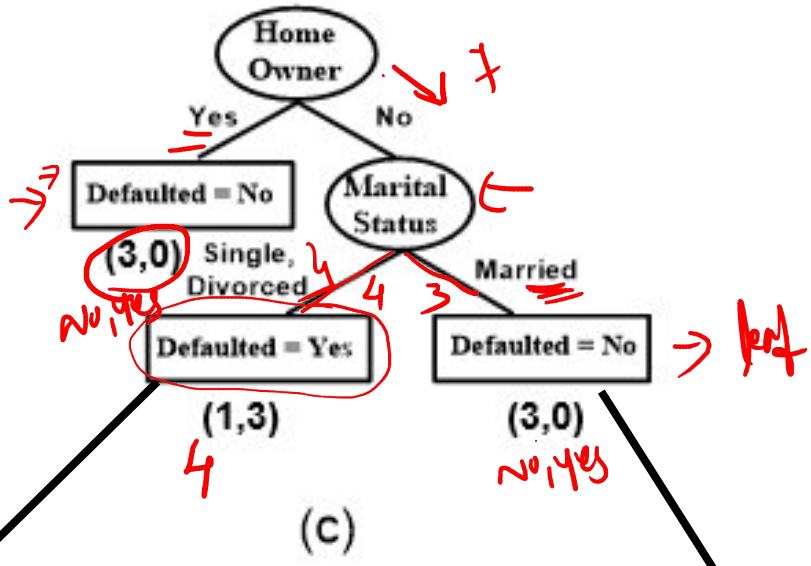
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
2	No	Married	100K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Contd..



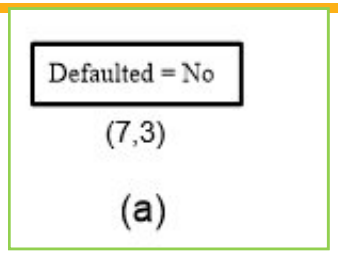


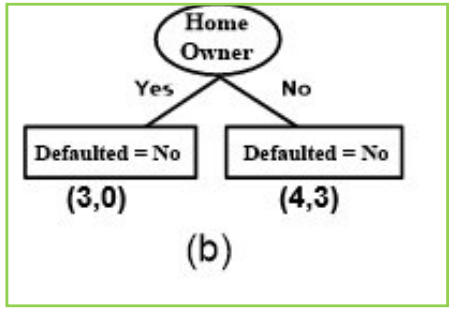


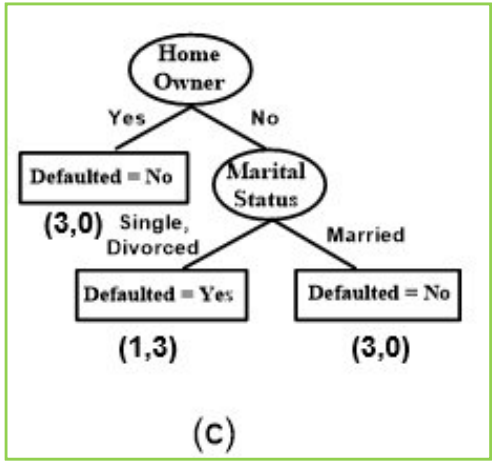
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
3	No	Single	70K	No
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

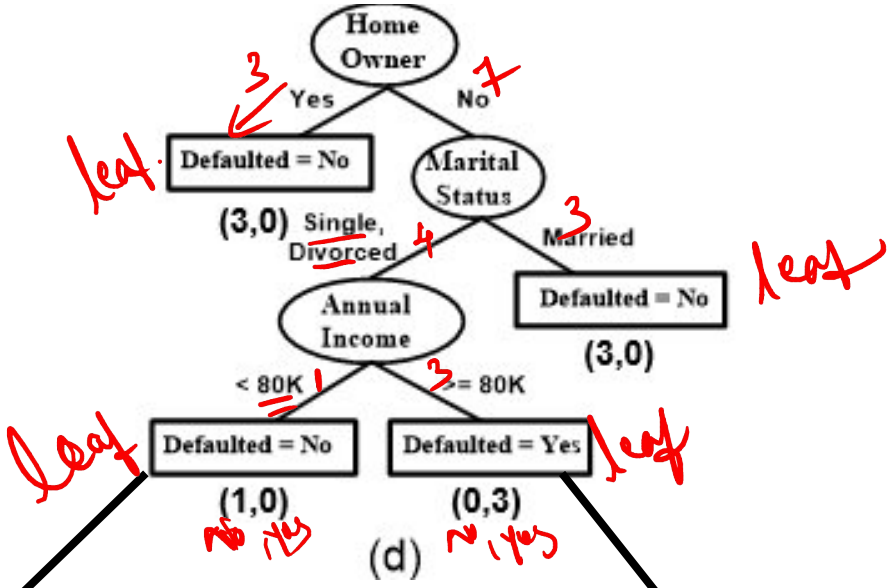
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
2	No	Married	100K	No
6	No	Married	60K	No
9	No	Married	75K	No

Contd..







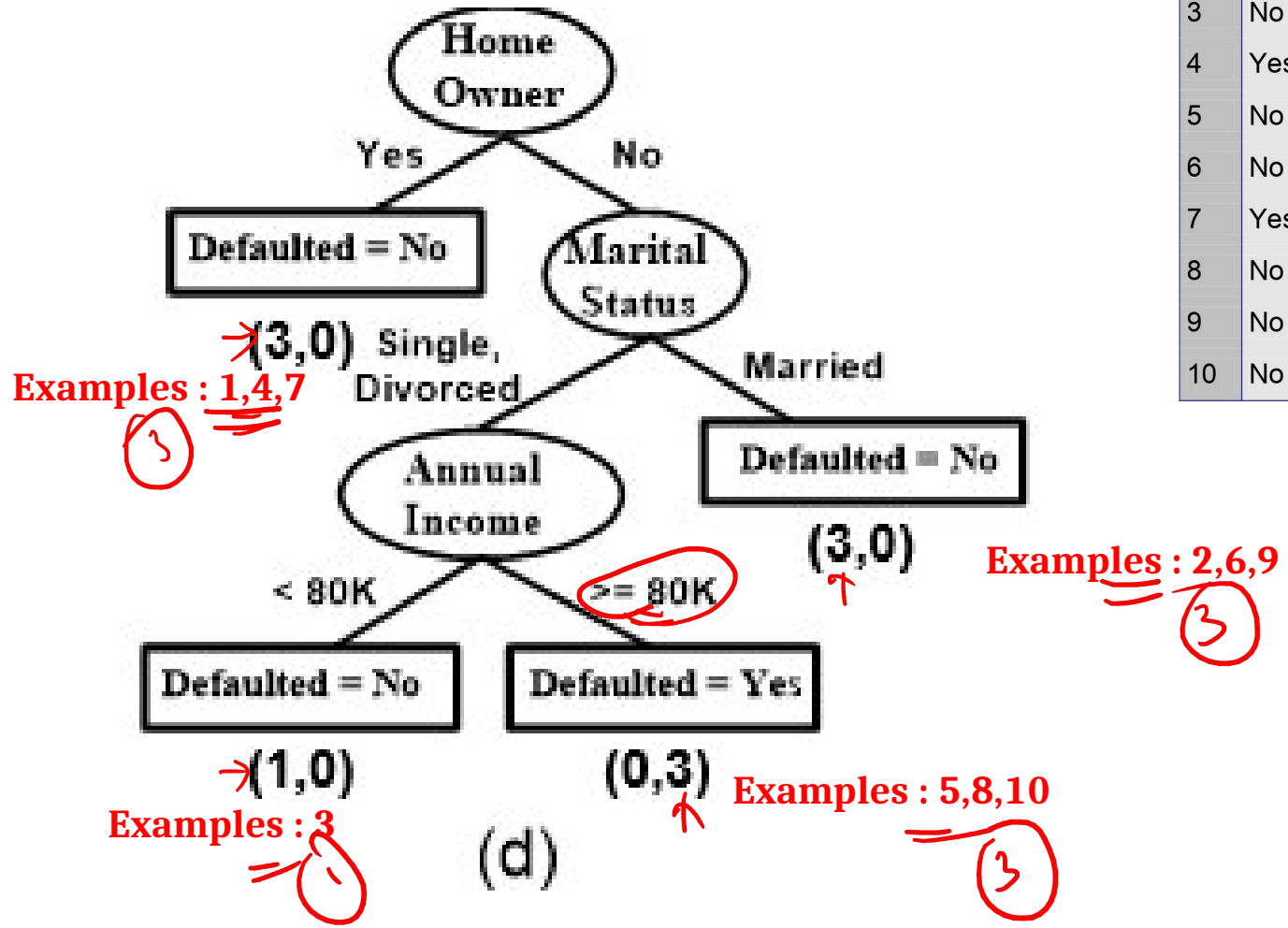


ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
3	No	Single	70K	No

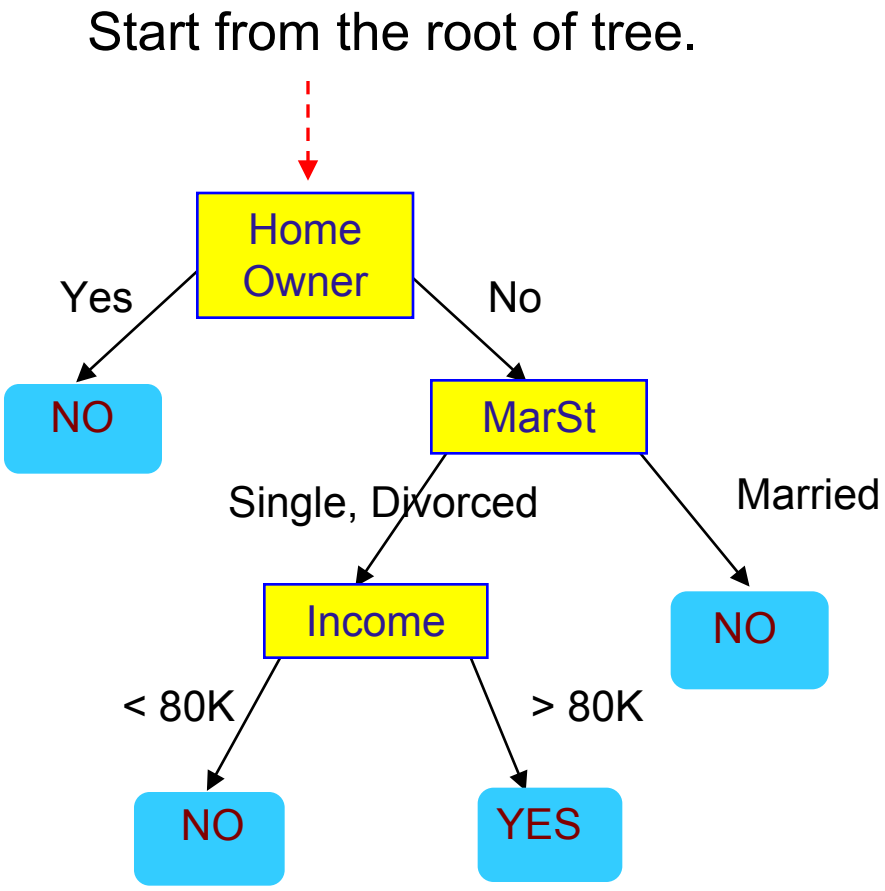
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

Final Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Apply Model to Test Data



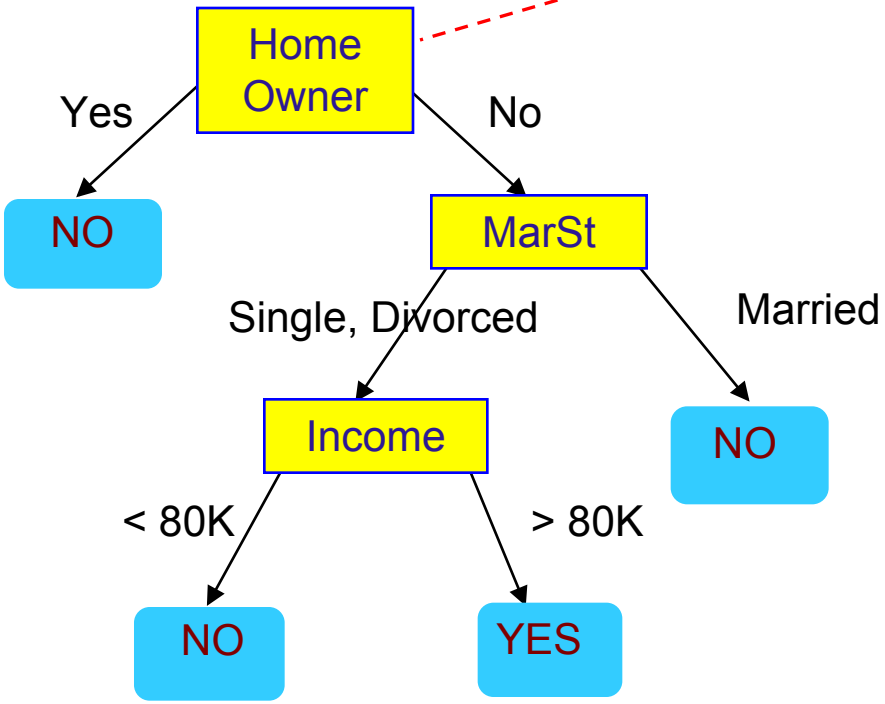
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Apply Model to Test Data

Test Data

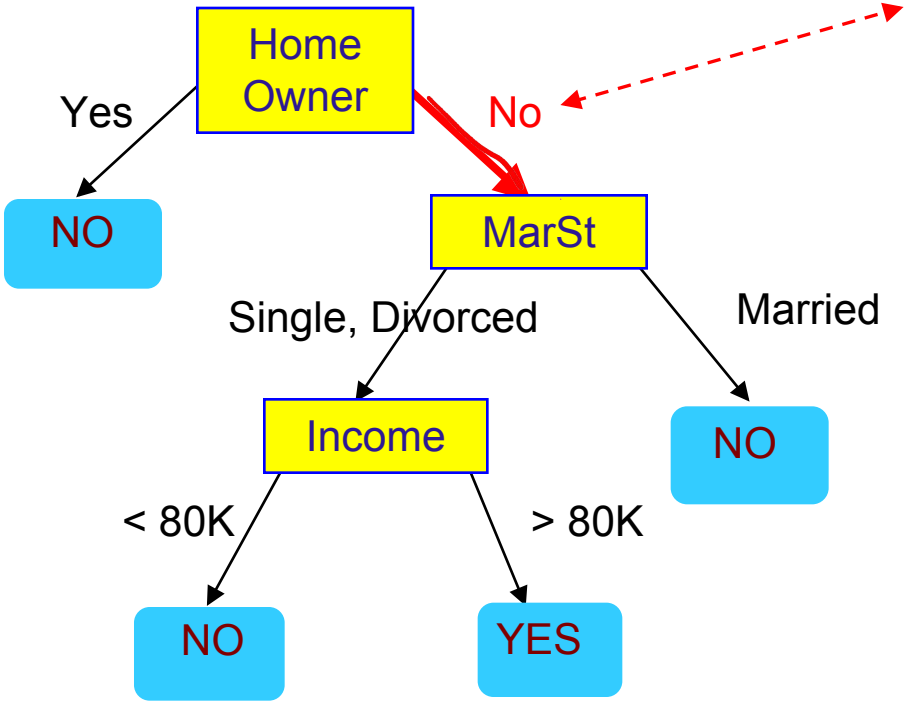
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

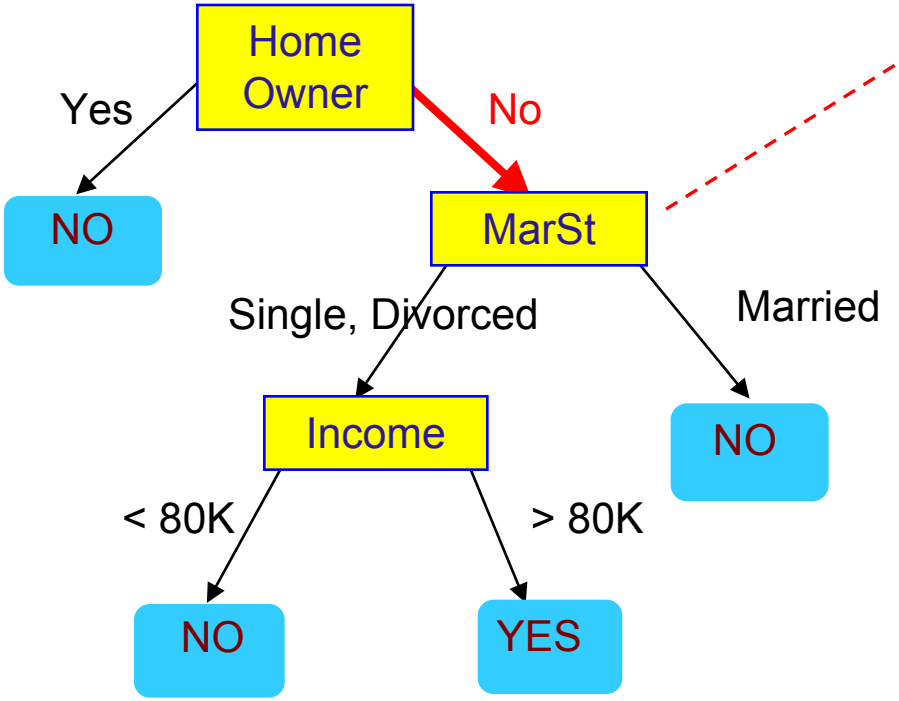
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



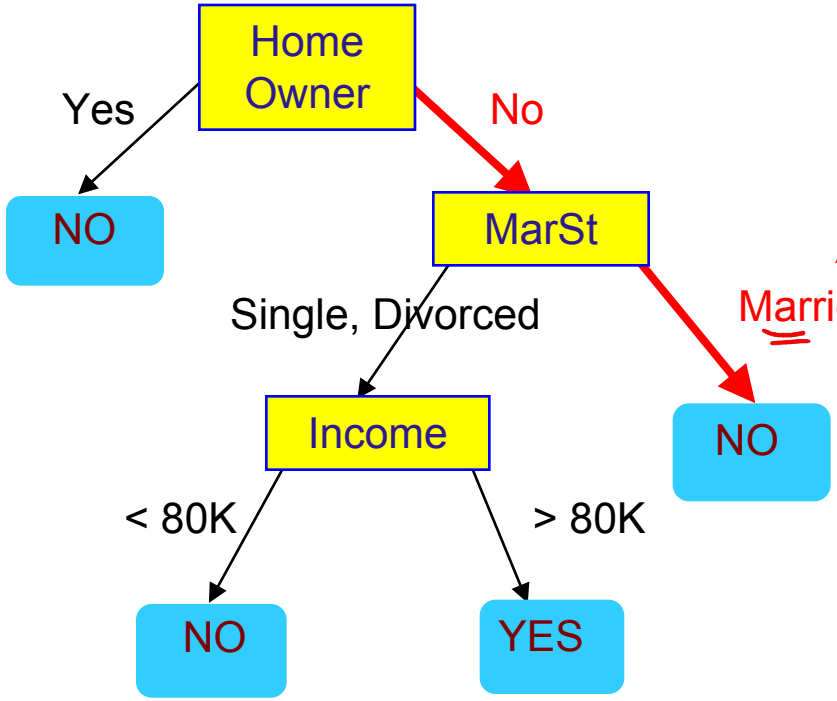
Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data



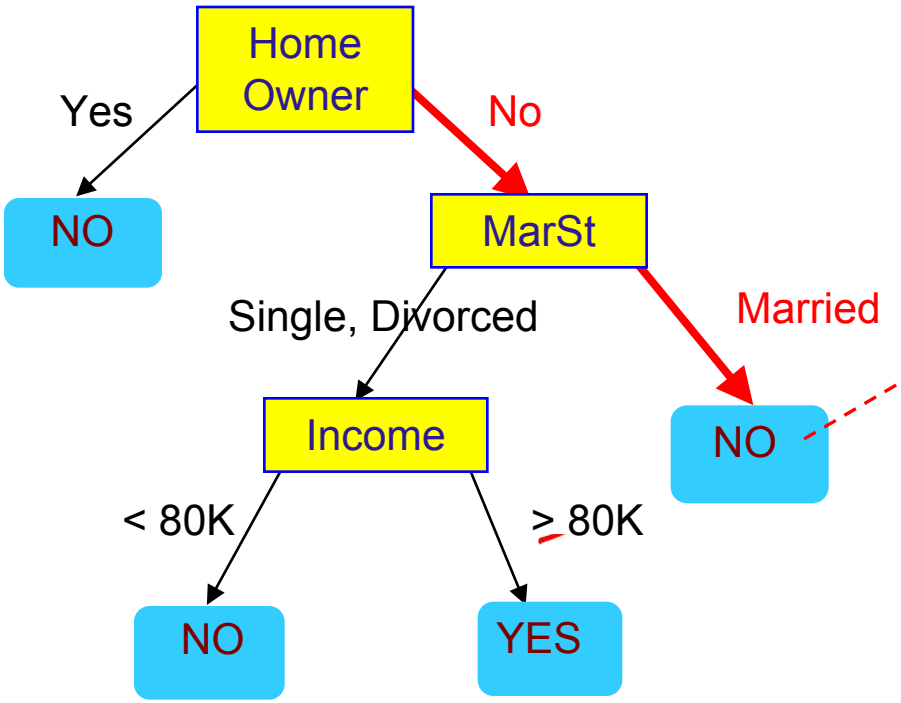
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	? NO

Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Assign Defaulted to "No"

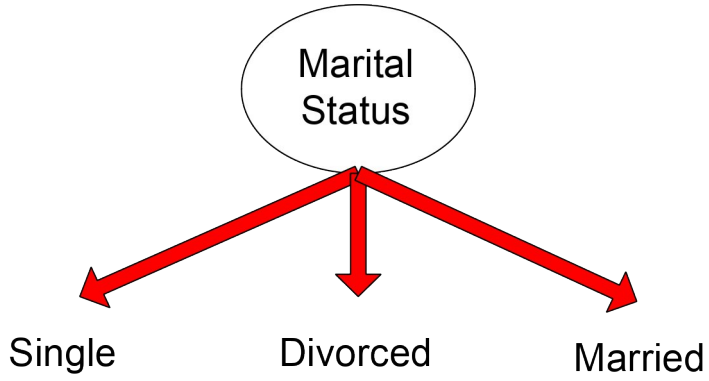
Design Issues of Decision Tree Induction

- **How should training records be split?**
  - Method for expressing test condition
  - depending on attribute types
    - **Binary, Nominal, Ordinal, Continuous**
  - Measure for evaluating the goodness of a test condition

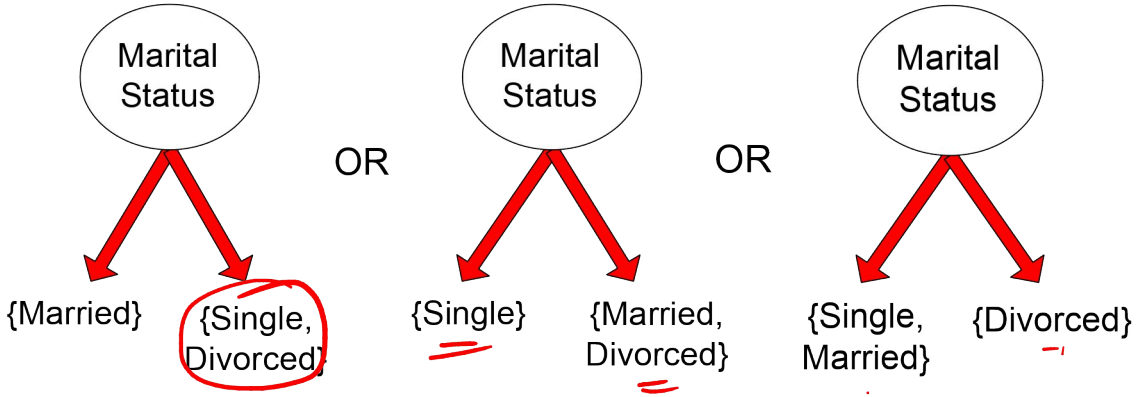
- **How should the splitting procedure stop?**
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Test Condition for Nominal Attributes

- Multi-way split:
  - Use as many partitions as distinct values.

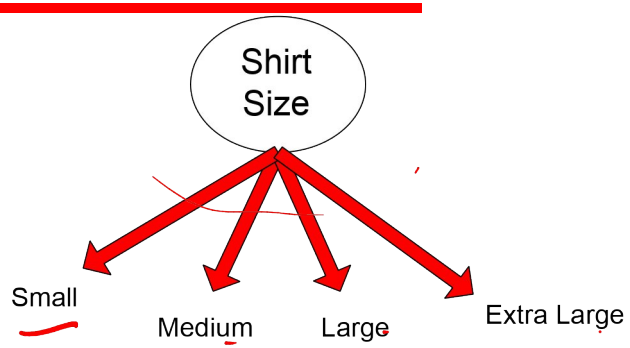


- Binary split:
  - Divides values into two subsets

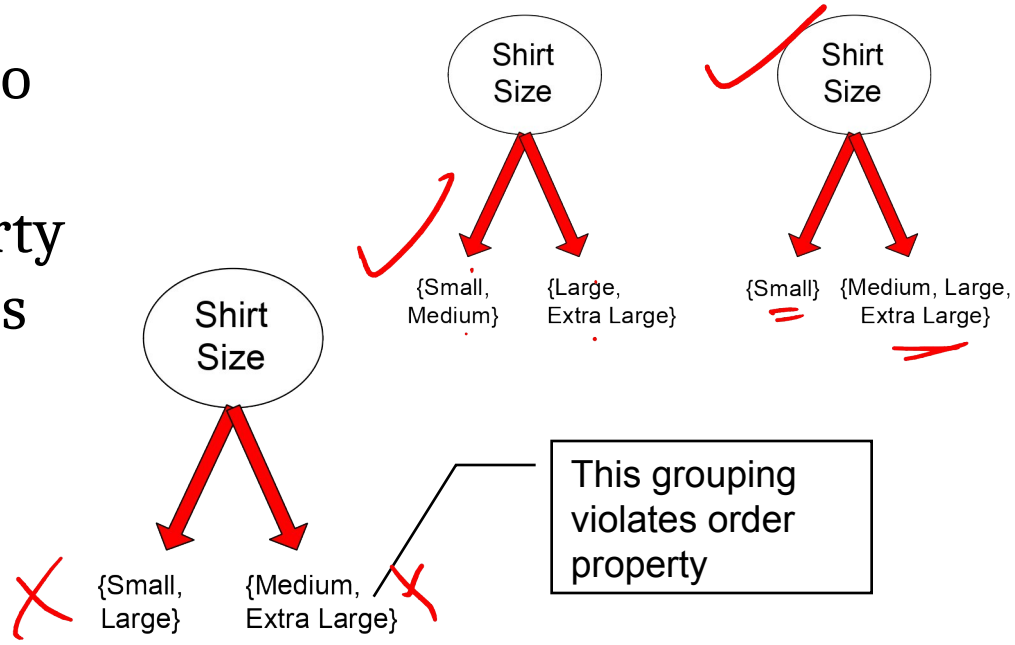


Test Condition for Ordinal Attributes

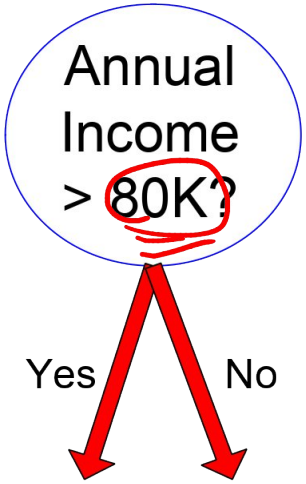
- Multi-way split:
  - Use as many partitions as distinct values.



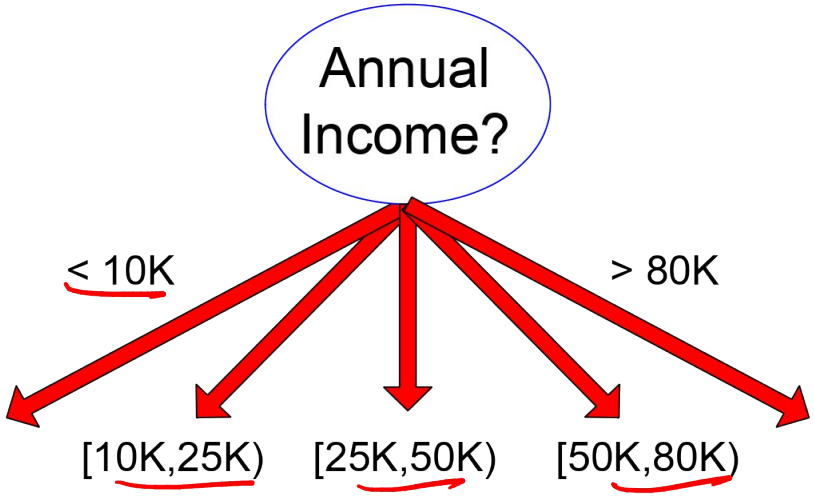
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

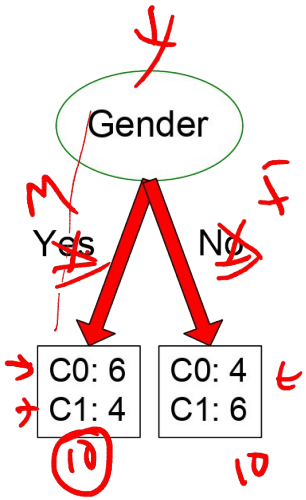
Binary Decision : consider all possible splits and finds the best cut can be more compute intensive

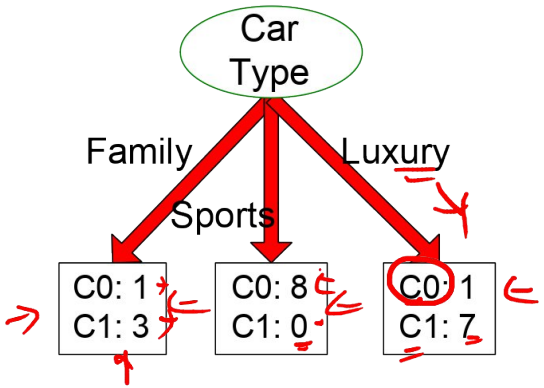
Discretization : to form an ordinal categorical attribute

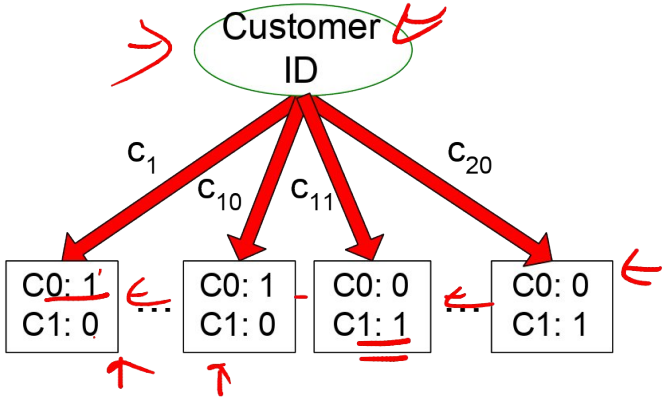
How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Before Splitting: 10 records of class 0,
10 records of class 1



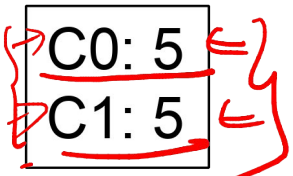




Which test condition is the best?

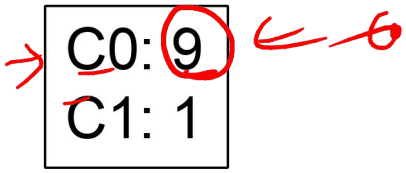
How to determine the Best Split

- Impurity is a measure of homogeneity of the labels at the node at hand
- **Greedy approach:**
  - Nodes with purer class distribution are preferred
- **Need a measure of node impurity:**



High degree of impurity/ High Uncertainty

Relating it to information theory: Low information content e.g It rained heavily in Shillong yesterday



Low degree of impurity / Low Uncertainty

Relating it to information theory: High information content e.g There was a heavy rainfall in Rajasthan last night.

Measure of Information

- The amount of information (surprise element) conveyed by a message is inversely proportional to its probability of occurrence. That is

$$I_k \alpha \frac{1}{p_k}$$

- The mathematical operator satisfies above properties is the logarithmic operator.

$$I_k = \log_r \frac{1}{p_k} \; units$$

Measures of Node Impurity

- **Gini Index**

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

- **Entropy**

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

Where $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes

- As uncertainty increases, entropy increases

- Measures seek to determine which variable would split the data to lead to the underlying child nodes being most homogenous or pure. **In other words splitting on attribute should reduce impurity**

Impurity Measure for a Single Node

Node N_1	Count
Class=0	0
Class=1	6

$P(C_1) = 0/6 = 0$ and $P(C_2) = 6/6 = 1$

Gini $= 1 - (0)^2 - (1)^2 = 0$

Entropy $= -(0)\ \log_2(0) - (1)\ \log_2(1) = 0$

Low entropy and GINI index preferred – low impurity

Node N_2	Count
Class=0	1
Class=1	5

$P(C_1) = 1/6$ and $P(C_2) = 5/6$

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$

Entropy $= -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$

Node N_3	Count
Class=0	3
Class=1	3

$P(C_1) = 2/6$ $P(C_2) = 4/6$

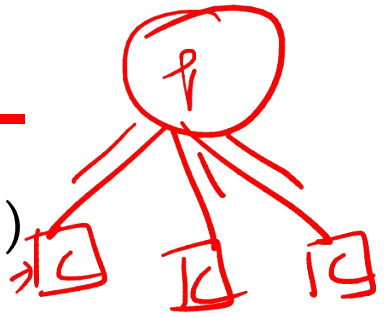
Gini $= 1 - (3/6)^2 - (3/6)^2 = 0.5$

Entropy $= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

$$Entropy = -\sum_{i=0}^{c-1} p_i(t)\log_2 p_i(t)$$

Computing impurity for a Collection of Nodes



Collective Impurity of Child Nodes

- When a node p is split into k partitions (children)

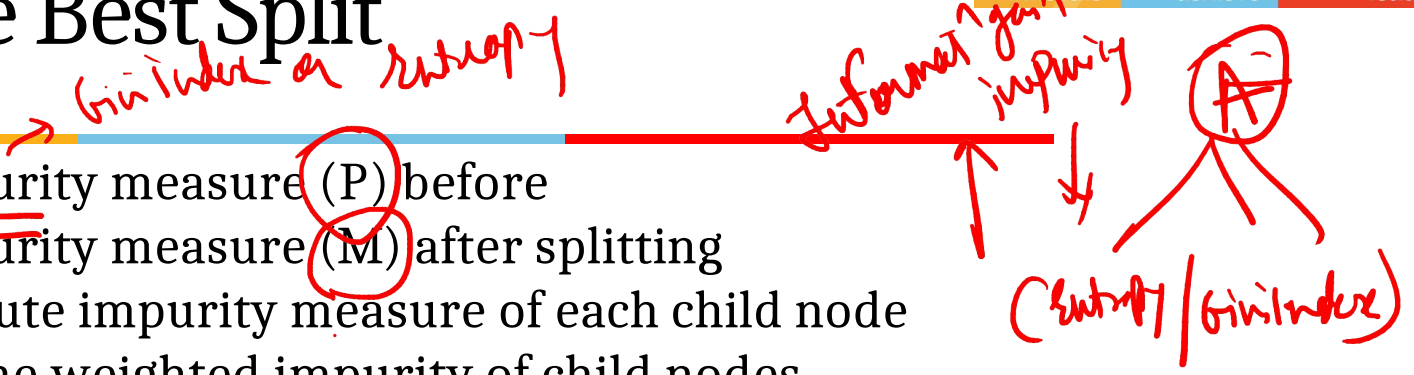
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

$$Entropysplit = \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

where, n_i = number of records at child i,

n = number of records at parent node p

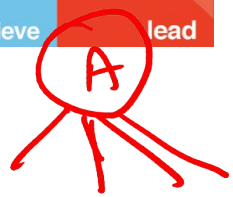
Finding the Best Split



- Compute impurity measure (P) before
- Compute impurity measure (M) after splitting
  - Compute impurity measure of each child node
  - M is the weighted impurity of child nodes
- Choose the attribute test condition that produces the **highest gain**

  Gain = P – M
- **or equivalently, lowest impurity measure after splitting (M)**
- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- **Maximizing the gain == minimizing the weighted average impurity measure of children nodes**

Computing Information Gain After Splitting



- Expected reduction in entropy knowing A / *after splitting the node with attribute A*

$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

Where , Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

Notations as per Tom Mithell book

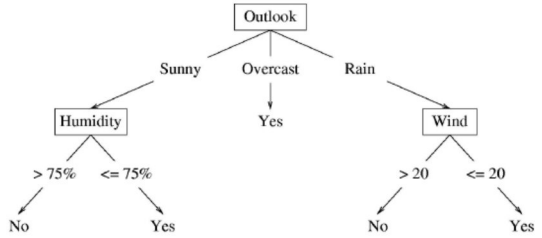
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms
- Information gain is the mutual information between the class variable and the splitting variable

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Problem Setting



Set of possible instances X
- each instance x in X is a feature vector
- e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>

- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y
- *Construct a DT -> Find the attribute that returns the highest information gain*

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- information gain due to sorting the original ***14*** examples by the attribute ***Wind***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-]$ = 14

$S_{Weak} = [6+, 2-]$ = 8

$S_{Strong} = [3+, 3-]$ = 6

$P_+ \Rightarrow 9/14$

$P_- \Rightarrow 5/14$

$$Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$Entropy([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- information gain due to sorting the original ***14*** examples by the attribute ***Wind***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-]$

$S_{Weak} = [6+, 2-]$

$S_{Strong} = [3+, 3-]$

$Entropy\ (S_{strong}[3+, 3-]) = 1$

$Entropy\ (S_{weak}[6+, 2-])$

$= -6/8\ log_2\ (6/8) - 2/8\ log_2\ (2/8)$

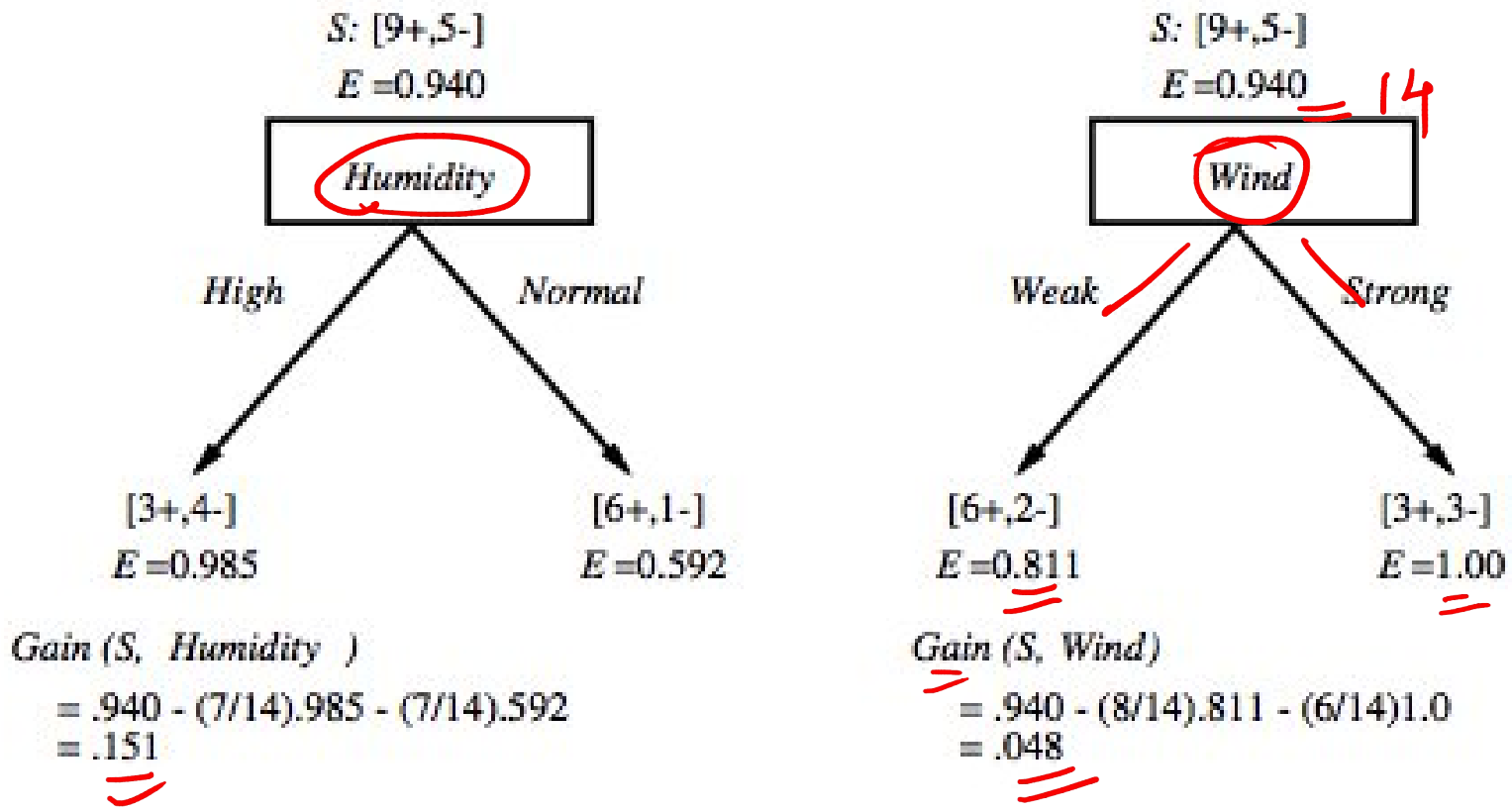
$= 0.811$

$Gain(S, A_{Wind}) = Entropy(S) - 8/14\ Entropy(S_{Weak}) - 6/14\ Entropy(S_{Strong})$

$= 0.94 - 8/14 \boxtimes 0.811 - 6/14 \boxtimes 1.00$

$= 0.048$

Which attribute is the best classifier?



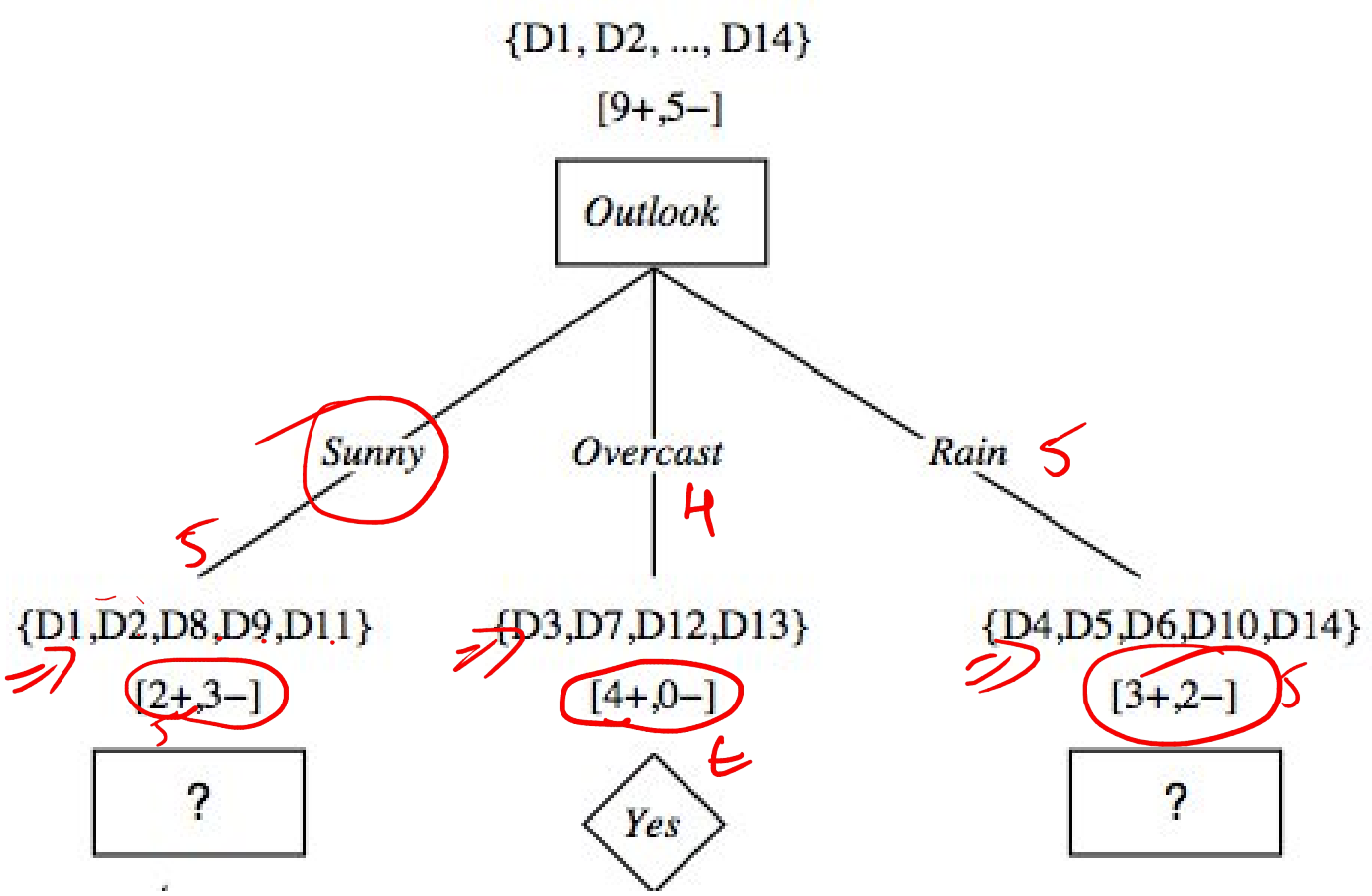
First step: which attribute to test at the root?

- Which attribute should be tested at the root?
  - *Gain*(*S*, *Outlook*) = 0.246
  - *Gain*(*S*, *Humidity*) = 0.151
  - *Gain*(*S*, *Wind*) = 0.048
  - *Gain*(*S*, *Temperature*) = 0.029
- *Outlook* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Outlook*
  - partition the training samples according to the value of *Outlook*

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

After first step



$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Outlook = Sunny branch and Attribute wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$S_{(outlook==sunny)} = [2+, 3-]$$
$$S_{(sunny \,\&\&\, week)} = [1+, 2-]$$
$$S_{(sunny\&\&srong)} = [1+, 1-]$$

$$Entropy\,(S_{sunny}) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$Entropy\,(S_{sunny}) = Entropy\,([2+, 3-]) = -2/5 \log_2 (2/5) - 3/5 \log_2 (3/5) = 0.97$$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

innovate achieve lead

Attribute wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$S_{(outlook==sunny)} = [2+, 3-]$

$S_{(sunny \&\& week)} = [1+, 2-]$

$S_{(sunny \&\& srong)} = [1+, 1-]$

$E(S_{(week \&\& sunny)}) = E([1+,2-]$

$= -1/3 \log_2(1/3) - 2/3 \log_2(2/3)$

$= 0.918$

$E(S_{(sunny \&\& rong)}) = Entropy([1+,1-] = 1$

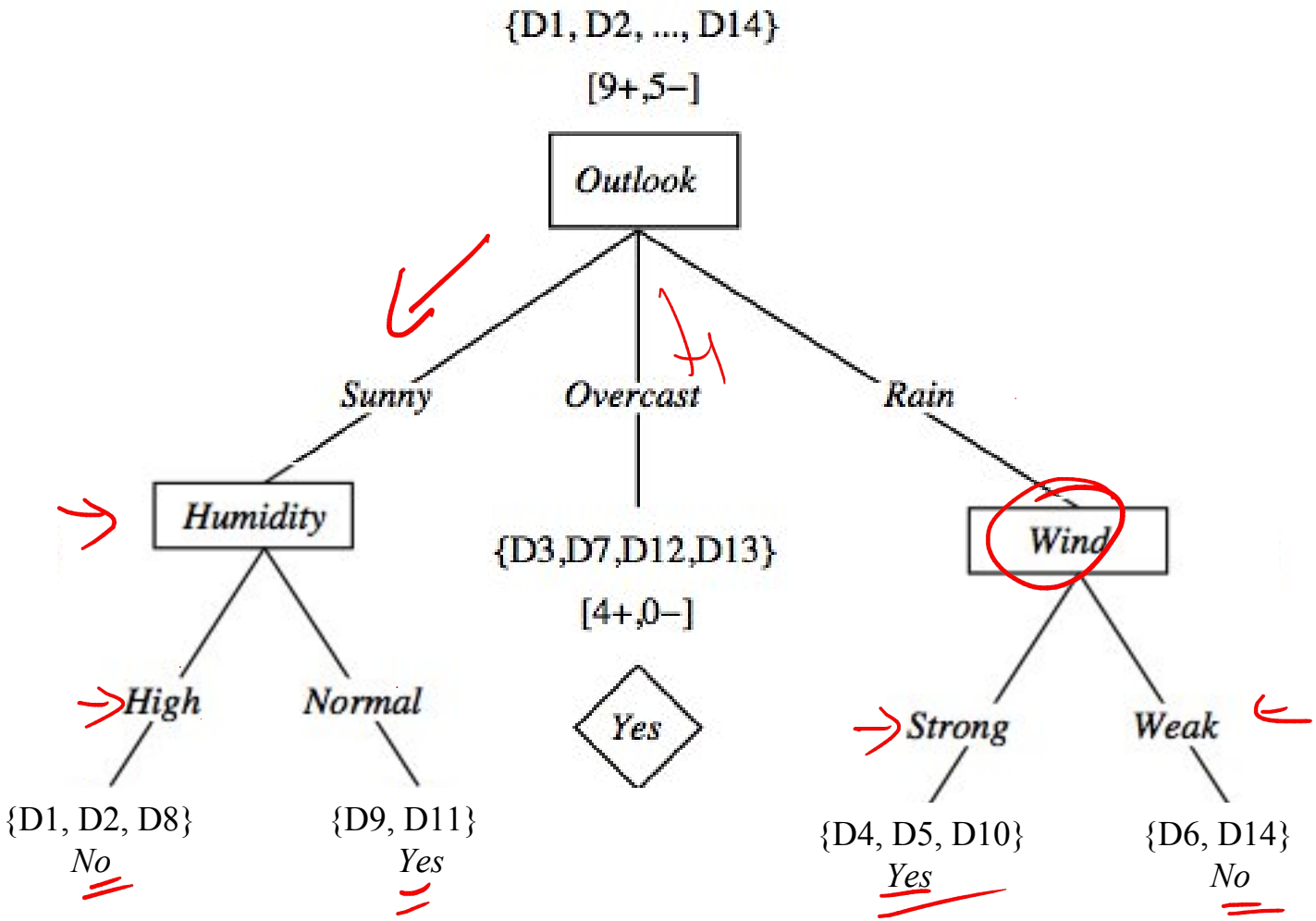
$Gain(S_{sunny}, A_{Wind}) = Entropy(S_{sunny}) - 3/5.\ Entropy(S_{sunny \&\& Weak}) - 2/5\ Entropy(S_{sunny \&\& Strong})$

$= 0.97 - 3/5 \times 0.918 - 2/5 \times 1 = 0.019$

Second step

- Working on *Outlook=Sunny* node:
  $Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \cdot 0.0 - 2/5 \cdot 0.0 = 0.970$
  $Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \cdot 1.0 - 3/5 \cdot 0.918 = 0.019$
  $Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \cdot 0.0 - 2/5 \cdot 1.0 - 1/5 \cdot 0.0 = 0.570$
- *Humidity* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Humidity*
  - partition the training samples according to the value of *Humidity*

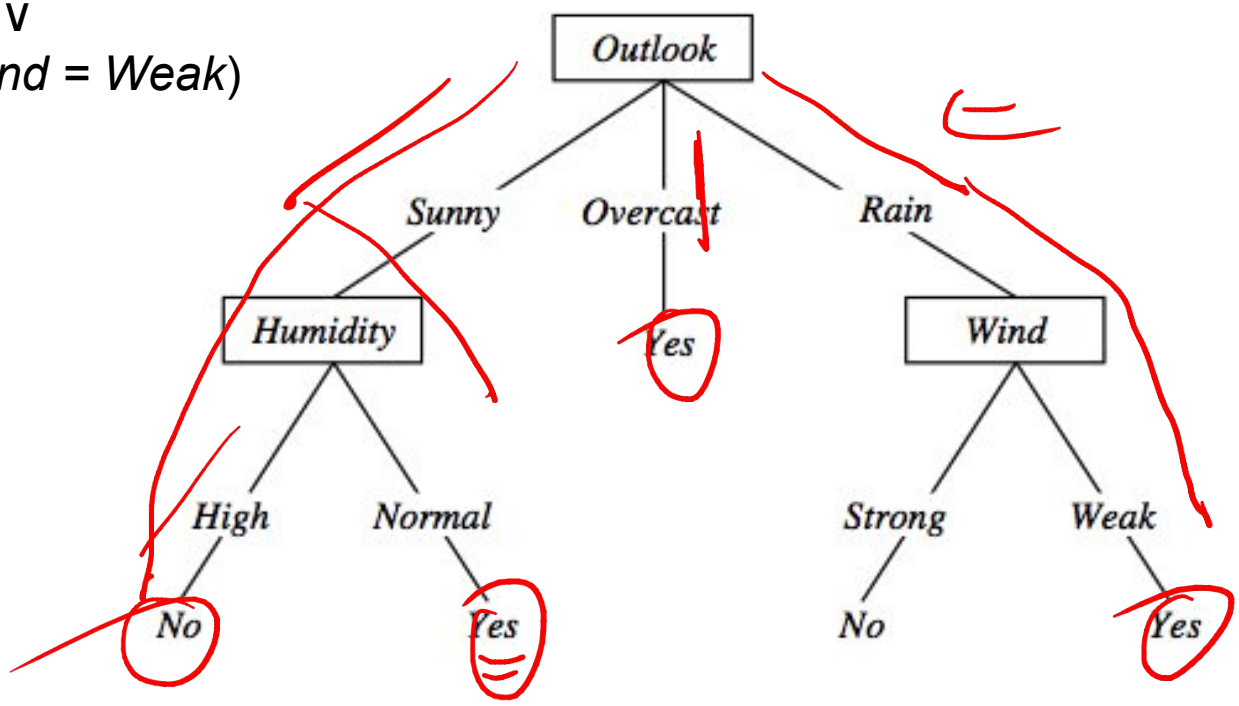
Second and third steps



Decision Trees – Representation & Expressiveness

Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

(*Outlook* = *Sunny* ∧ *Humidity* = *Normal*) ∨
(*Outlook* = *Overcast*) ∨
(*Outlook* = *Rain* ∧ *Wind* = *Weak*)

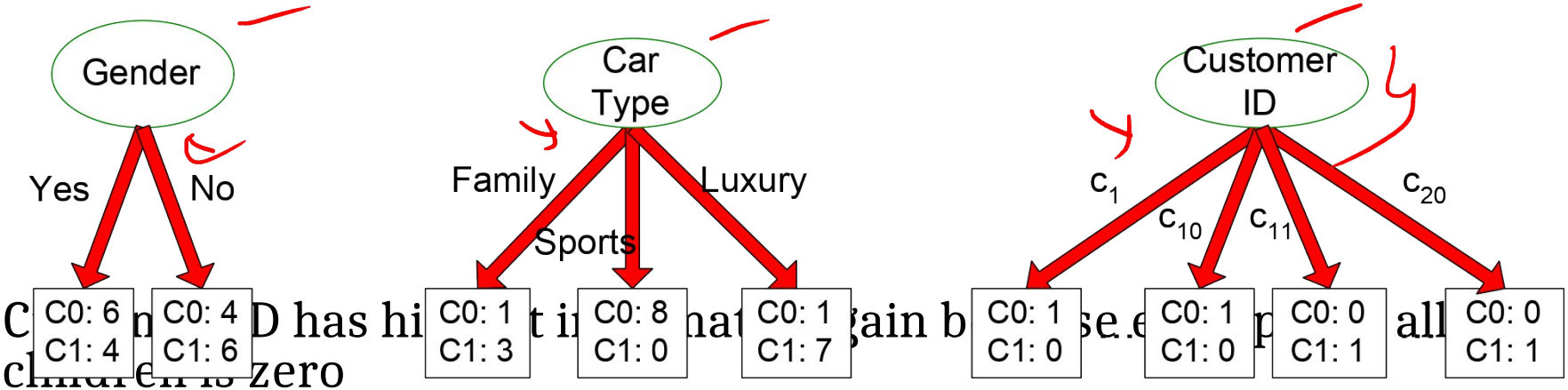


⟨*Outlook*=*Sunny*, *Temp*=*Hot*, *Humidity*=*High*, *Wind*=*Strong*⟩

No

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



- C[illegible]n ID has hi[illegible]t i[illegible]ha[illegible]gain b[illegible]se.e[illegible]p all children is zero
- Consider the attribute *Date* in the *PlayTennis* example.
  - *Date* would have the highest information gain since it perfectly separates the training data.
- Very good on the training, this tree would perform poorly in predicting unknown instances. Overfitting.

An alternative measure: Gain ratio

- **Recall…** Given a collection S, the entropy of S relative to target variable is

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

- *SplitInformation* measures the entropy of *S* with respect to the values of *A*. The more uniformly dispersed the data the higher it is.

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_1 through S_c, are the ***c*** subsets of examples resulting from partitioning S by the c-valued attribute A

- SplitInfomation term discourages the selection of attributes with many uniformly distributed values such as *Date*

An alternative measure: Gain ratio

- Let $|S| = n$, *Date* splits examples in n classes
  - $SplitInformation(S, Date) = -[(1/n \log_2 1/n) + \ldots + (1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$
- Compare with A, which splits data in two even classes:
  - $SplitInformation(S, A) = -[(1/2 \log_2 1/2) + (1/2 \log_2 1/2)] = -[-1/2 - 1/2] = 1$
- The Gain Ratio measure is defined in terms of the earlier Gain measure, as well as this SplitInfomation

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

Limitation of Information Gain:

Ways to tackle

$$\text{Entropy(parent)} = -\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20} = 1.$$

If `Gender` is used as attribute test condition:

$$\text{Entropy(children)} = \frac{10}{20}\left[-\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10}\right] \times 2 = 0.971$$

$$\text{Gain Ratio} = \frac{1-0.971}{-\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20}} = \frac{0.029}{1} = 0.029$$

If `Car Type` is used as attribute test condition:

$$\text{Entropy(children)} = \frac{4}{20}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right] + \frac{8}{20} \times 0 + \frac{8}{20}\left[-\frac{1}{8}\log_2\frac{1}{8} - \frac{7}{8}\log_2\frac{7}{8}\right] = 0.380$$

$$\text{Gain Ratio} = \frac{1-0.380}{-\frac{4}{20}\log_2\frac{4}{20} - \frac{8}{20}\log_2\frac{8}{20} - \frac{8}{20}\log_2\frac{8}{20}} = \frac{0.620}{1.52} = 0.41$$

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Limitation of Information Gain:

Ways to tackle

Finally, if `Customer ID` is used as attribute test condition:

$$\text{Entropy(children)} = \frac{1}{20}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right] \times 20 = 0$$

$$\text{Gain Ratio} = \frac{1-0}{-\frac{1}{20}\log_2\frac{1}{20} \times 20} = \frac{1}{4.32} = 0.23$$

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Inference : Thus, even though Customer ID has the highest information gain, its gain ratio is lower than Car Type since it produces a larger number of splits

ID3

- Uses **Hunt's algorithm**
- Uses **information gain** as the measure to select the best attribute at each step in the growing tree
- Does not have the ability to determine how many alternative decision trees are consistent with the available training data
- **Shorter trees are preferred over longer trees.**
- Trees that place high information gain attributes close to the root are preferred over those that do not - Inductive bias of ID3
- Refer: Mitchell

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

ID3: algorithm

ID3(X, T, $Attrs$) X: training examples:
T: target attribute (e.g. *PlayTennis*),
$Attrs$: other attributes, initially all attributes

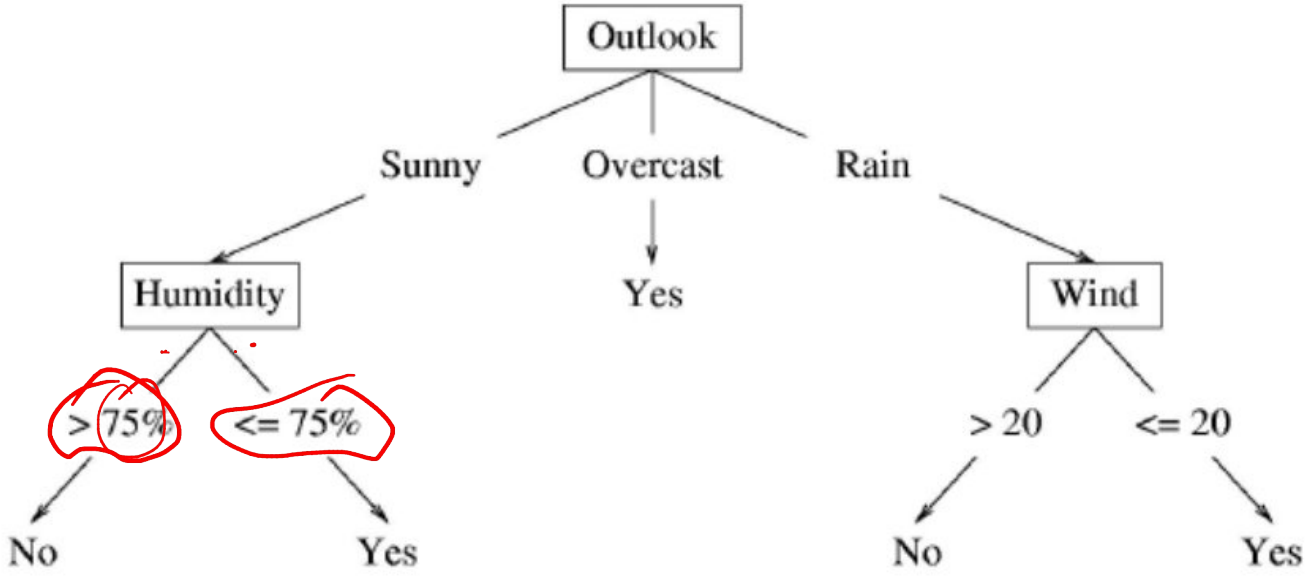
Create *Root* node
If all X's are +, *return* *Root* with class +
If all X's are –, *return* *Root* with class –
If *Attrs* is empty *return* *Root* with class most common value of T in X
else
  A ← best attribute; decision attribute for Root ← A
  For each possible value v_i of A:
  - add a new branch below *Root*, for test $A = v_i$
  - X_i ← subset of X with $A = v_i$
  - *If* X_i is empty *then* add a new leaf with class the most common value of T in X
    else add the subtree generated by ID3(X_i, T, $Attrs - \{A\}$)
return *Root*

refer shorter hypotheses: Occam's razor

- Why prefer shorter hypotheses?
- Arguments in favor:
  - There are fewer short hypotheses than long ones
  - If a short hypothesis fits data unlikely to be a coincidence
  - Elegance and aesthetics
- Arguments against:
  - Not every short hypothesis is a reasonable one.
- Occam's razor says that when presented with competing hypotheses that make the **same** predictions, one should select the solution which is simple"

Dealing with continuous-valued attributes

- If features are continuous, internal nodes can test the value of a feature against a threshold



Dealing with continuous-valued attributes

- Given a continuous-valued attribute A, dynamically create a new attribute Ac

  Ac = True if A < c, False otherwise

- How to determine threshold value c ?
- Example. Temperature in the PlayTennis example
  - Sort the examples according to Temperature
  - Temperature 40 48 | 60 72 80 | 90
  - PlayTennis No No 54 Yes Yes Yes 85 No
  - Determine candidate thresholds by averaging consecutive values where there is a change in classification: (48+60)/2=54 and (80+90)/2=85

Binary Splitting of continuous Attributes-Gini Index

ID	Home Owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes
6	No	Single	60000	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes
9	No	Married	75000	No
10	No	Single	90000	Yes

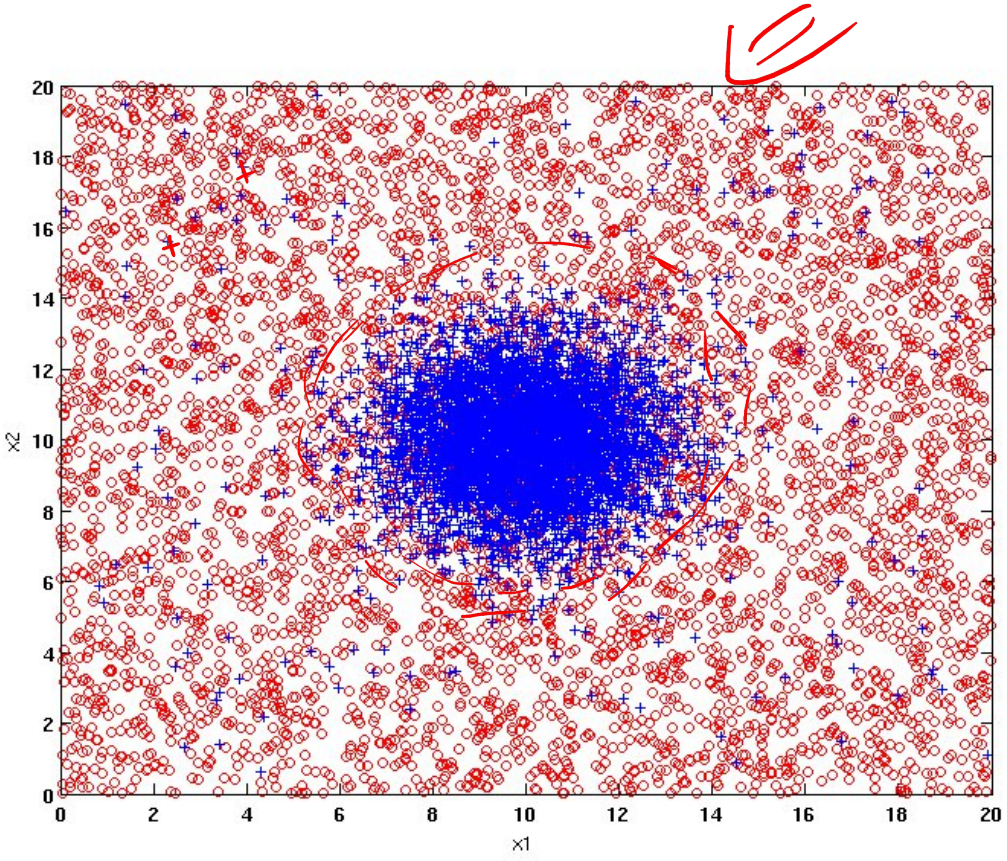
Binary Splitting of continuous Attributes-Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least Gini index (**the most purer subset**)
  - Splitpoint – 97.5 in below example

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (in '000s)										
Sorted Values	60	70	75	85	90	95	100	120	125	220
Split Positions	55	65	72.5	80	87.5	92.5	97.5	110	122.5	172.5	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

Example Data Set



Two class problem:

+ : 5200 instances

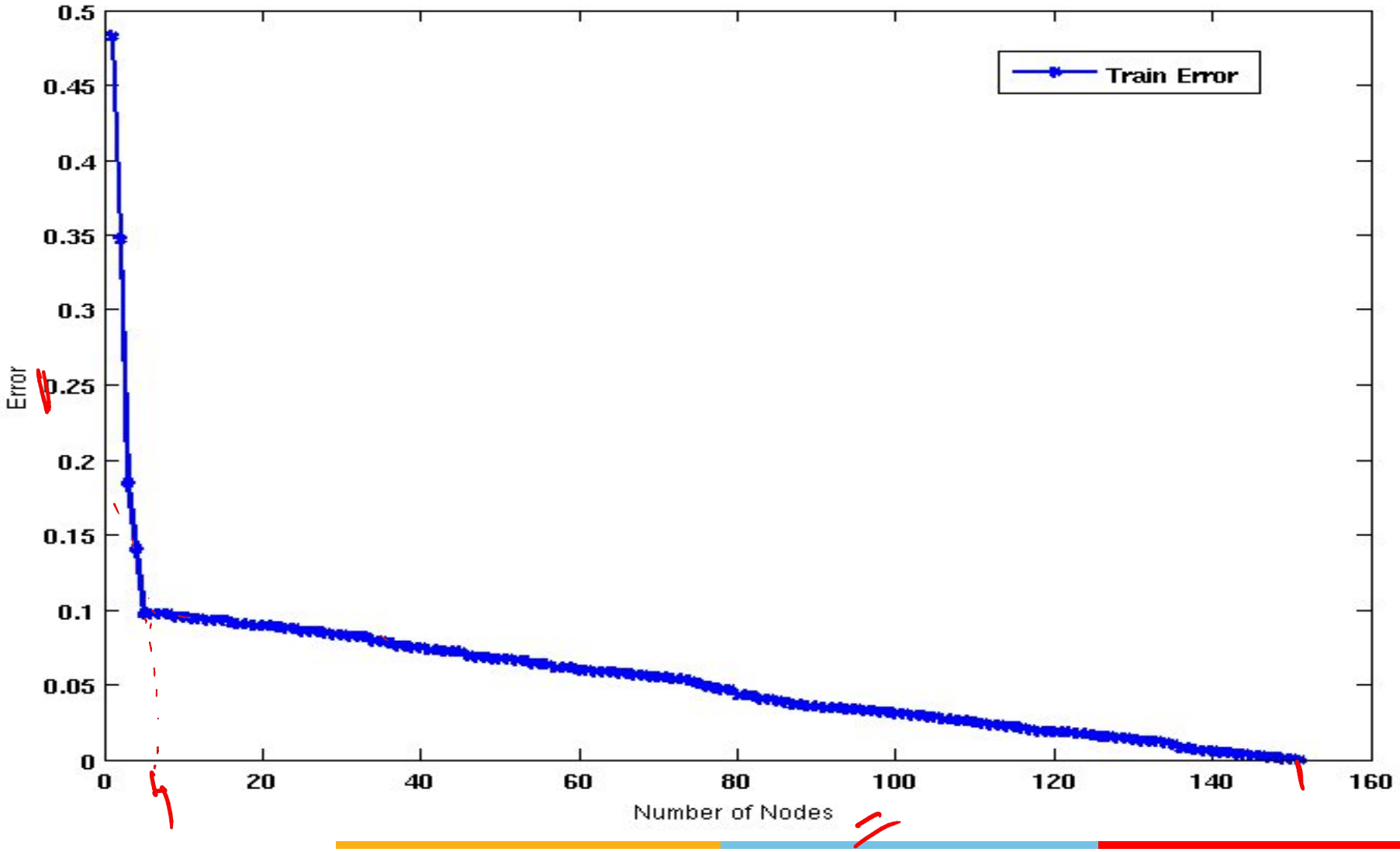
- **5000 instances generated from a Gaussian centered at (10,10)**
- **200 noisy instances added**

o : 5200 instances

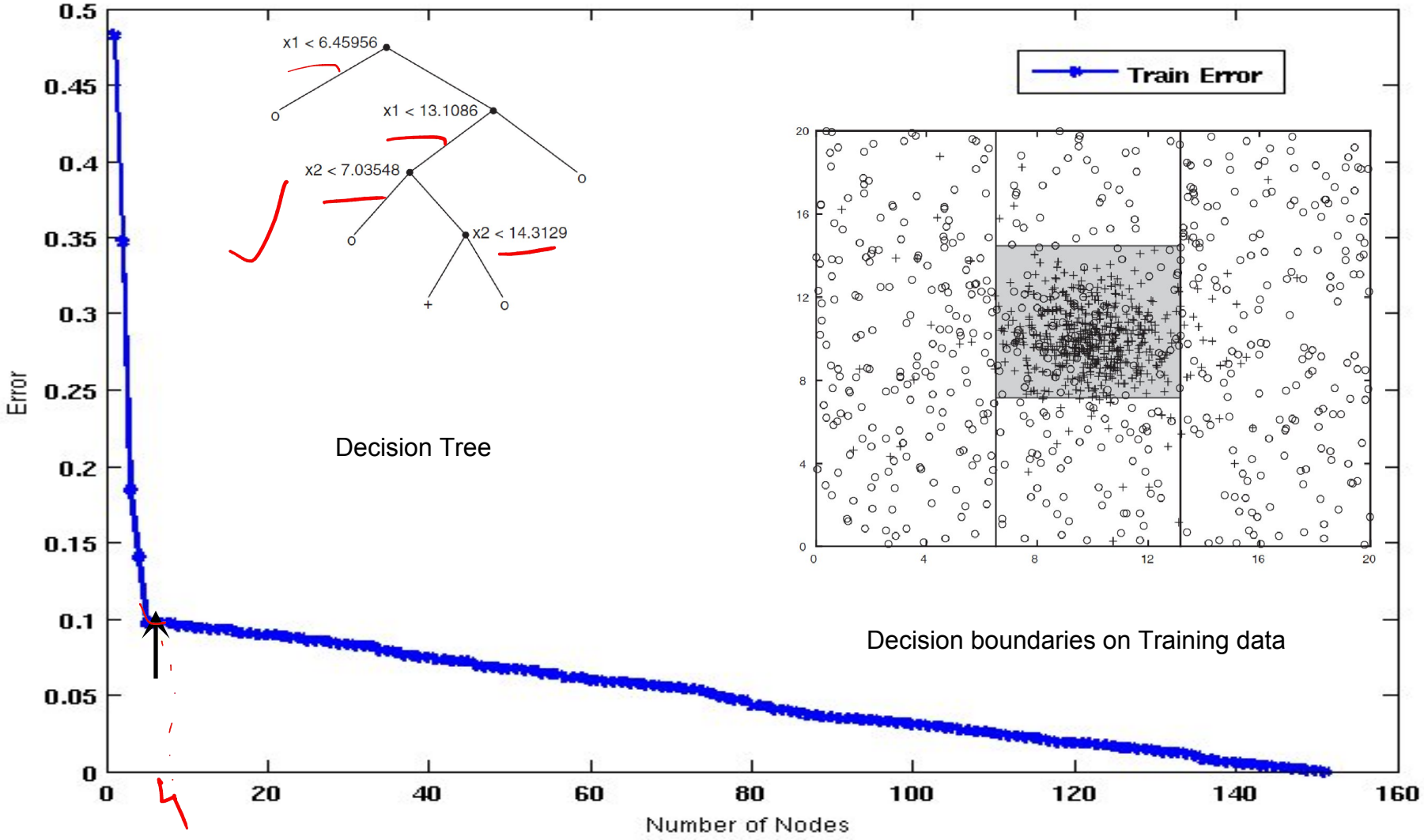
- **Generated from a uniform distribution**

50 % of the data used for training and 50% of the data used for testing

Increasing number of nodes in Decision Trees

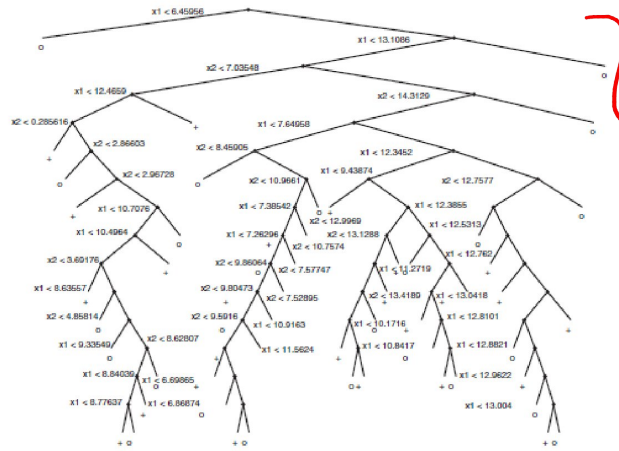


Decision Tree with 4 nodes

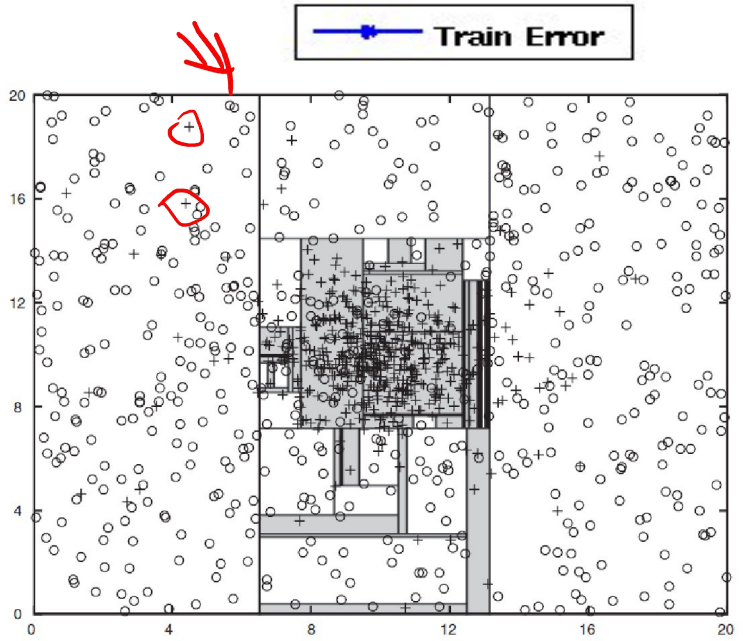


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Decision Tree with 50 nodes



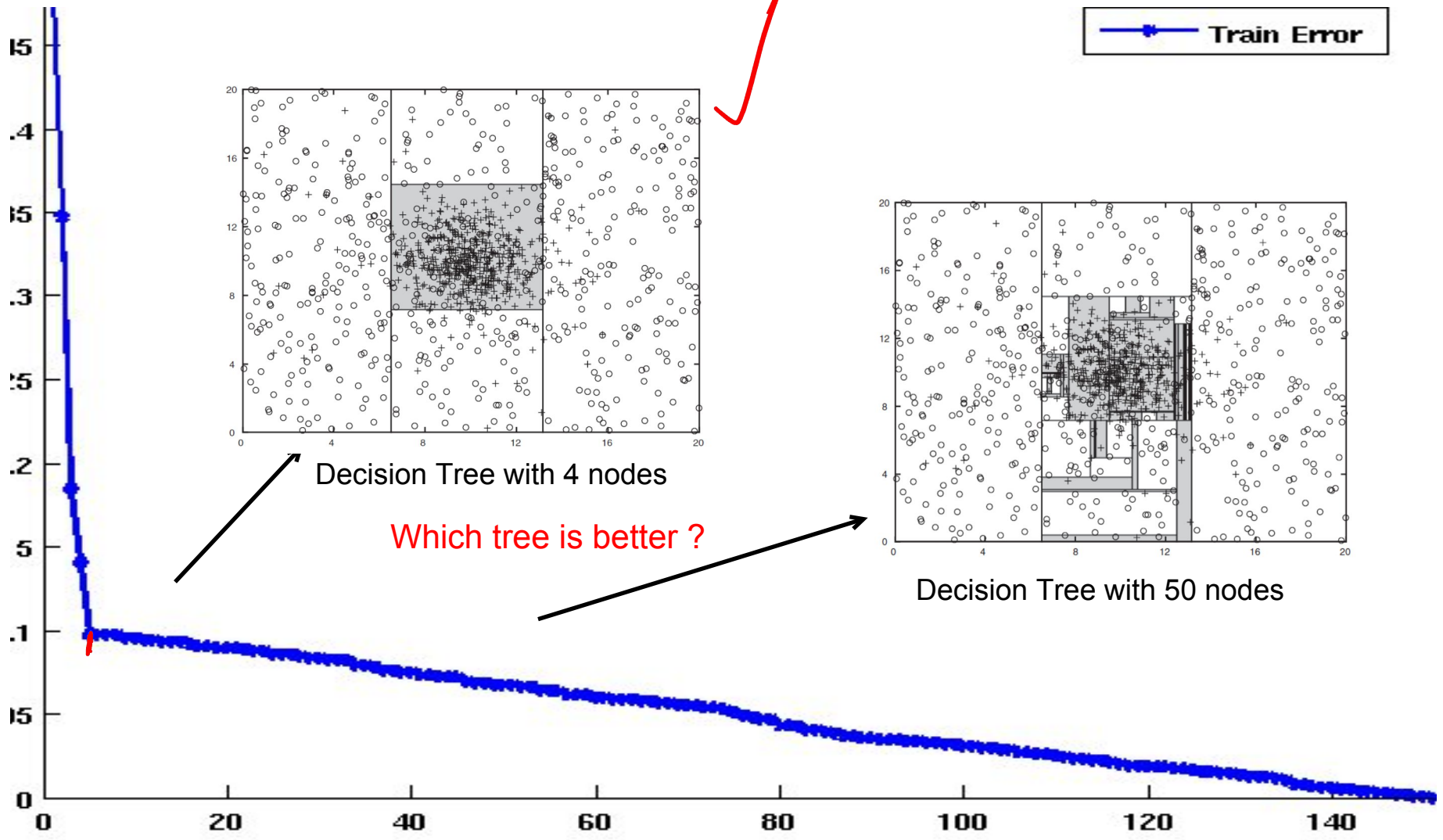
Decision Tree



Decision boundaries on Training data

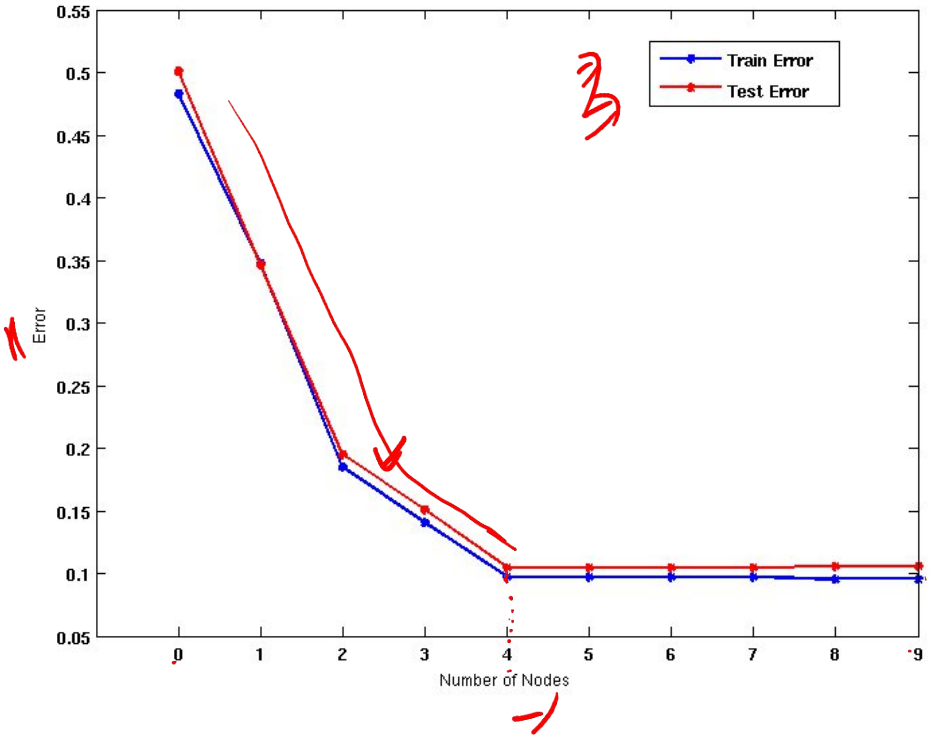
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

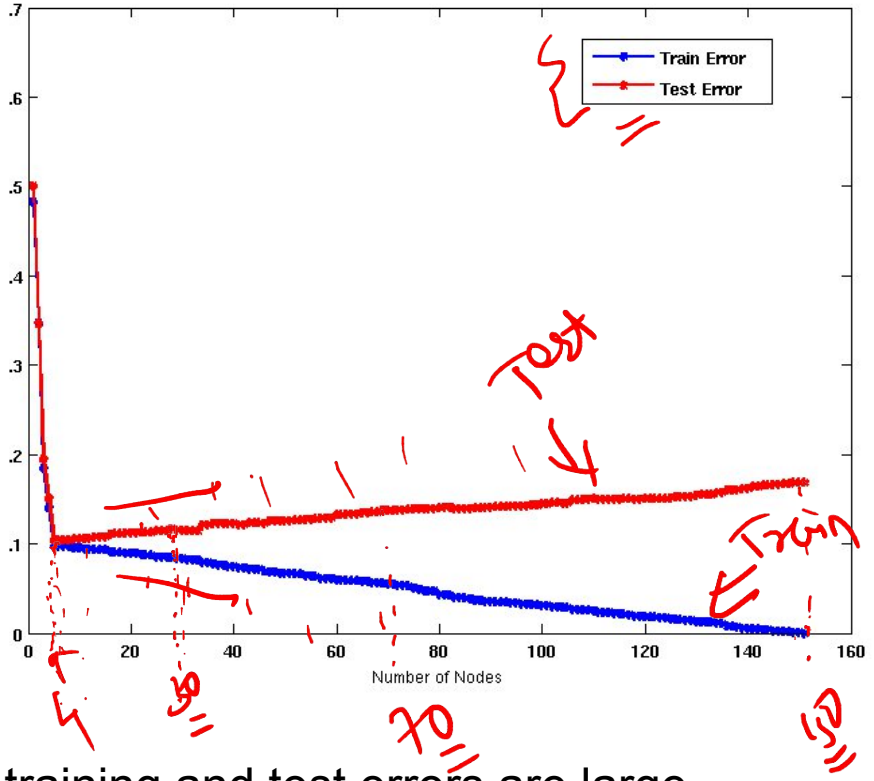
Which tree is better?



BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Model Overfitting





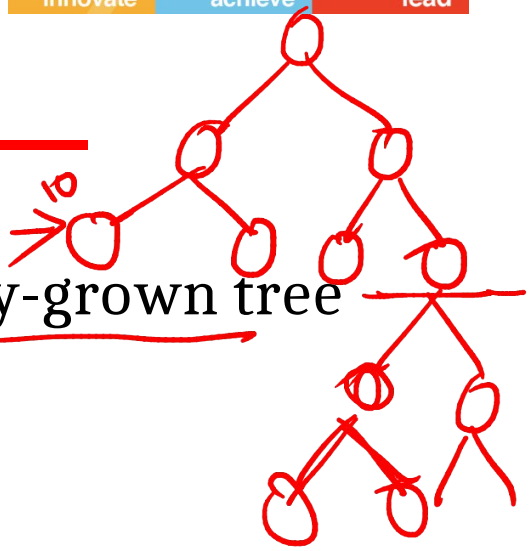
Underfitting: when model is too simple, both training and test errors are large
Overfitting: when model is too complex, training error is small but test error is large

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Issues in Decision Tree Learning - Handling Overfitting

- Two approaches
- Pre pruning (early stopping rule)
  - Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
  - Difficult to estimate when to stop growing the tree
- Post-pruning
  - Allow the tree to overfit the data, and then post-prune the tree

Model Selection for Decision Trees



- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
    - Stop if estimated generalization error falls below certain threshold

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Pre-pruning

Predict if the object being manufactured is likely to be "Reject"ed or "Allow"ed to be packed during the automated product inspection based on object features like : Shape, colour, size

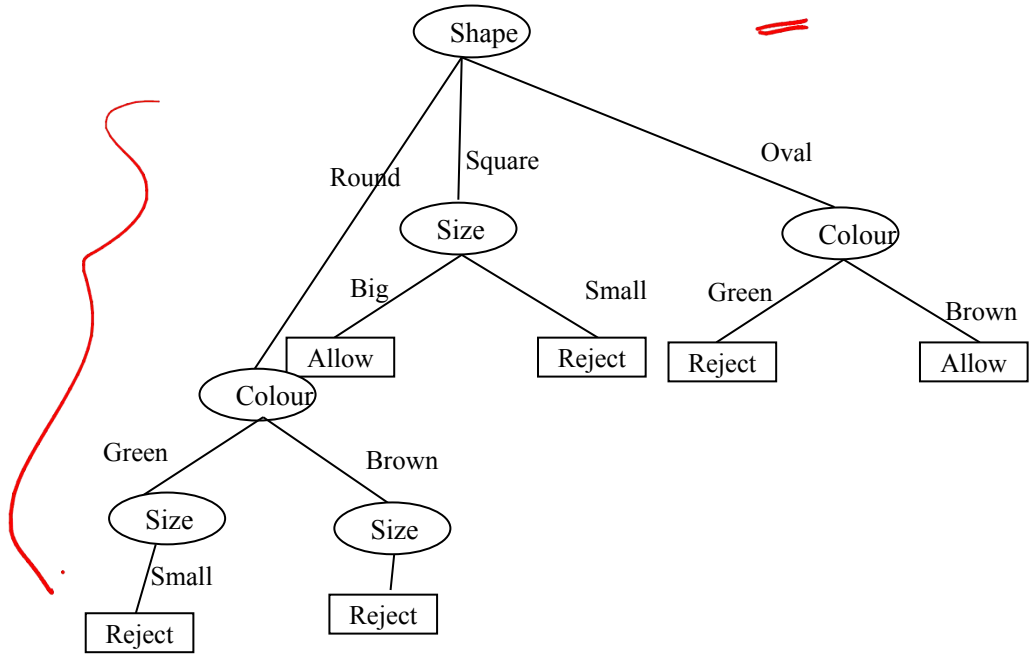
Assume this the trained model which is an over fit. Let try to experiment on Pre-pruning & Post-pruning technique.

Use validation set/test set for this experiments

Train set

Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Assumed over fit model



Test set

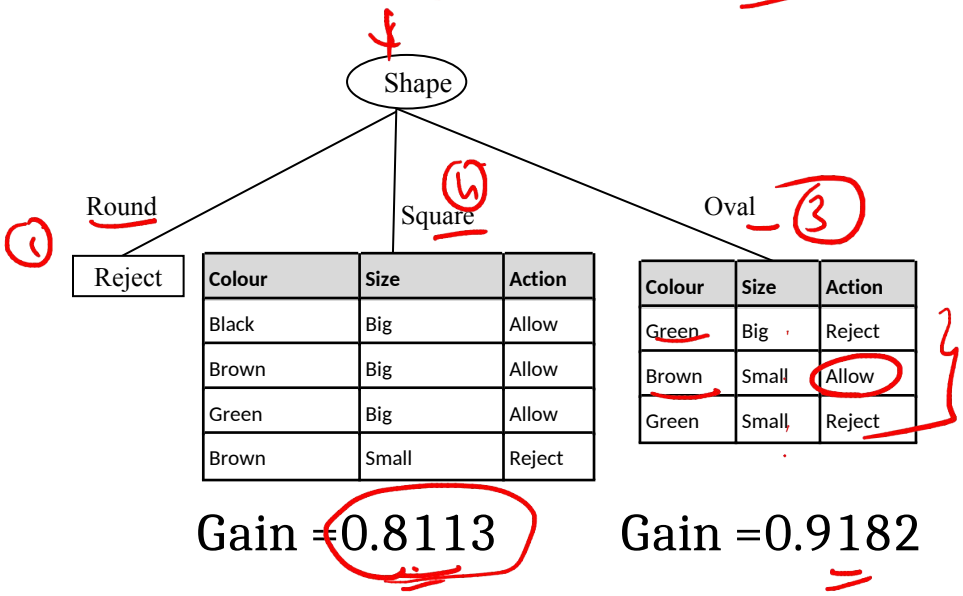
Shape	Colour	Size	Actio
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

Use maximum gain to stop the tree growth as threshold for pre-pruning

Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Prune if threshold Gain <=0.85

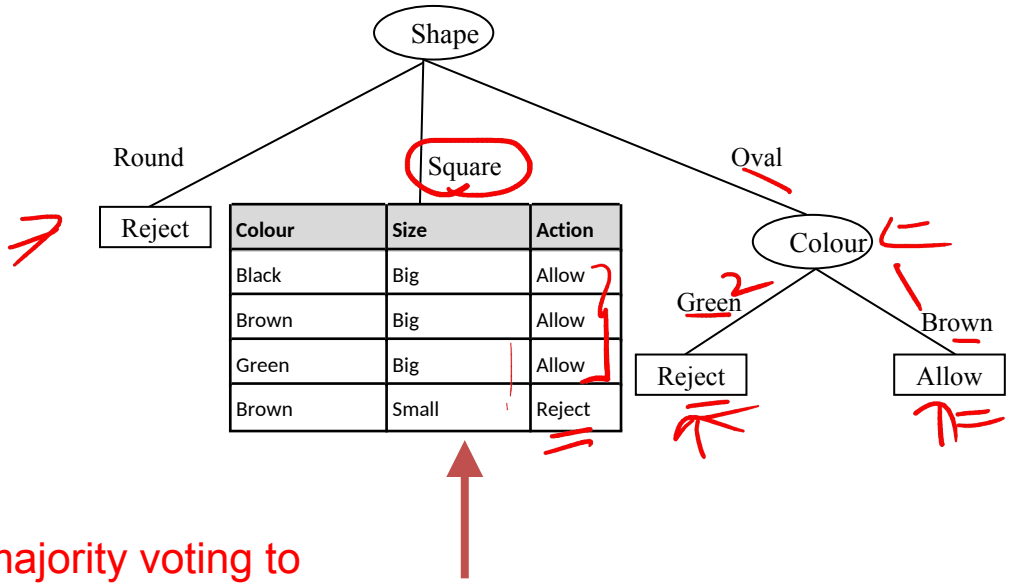


Shape	Colour	Size	Actio
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Prune if threshold Gain <=0.85



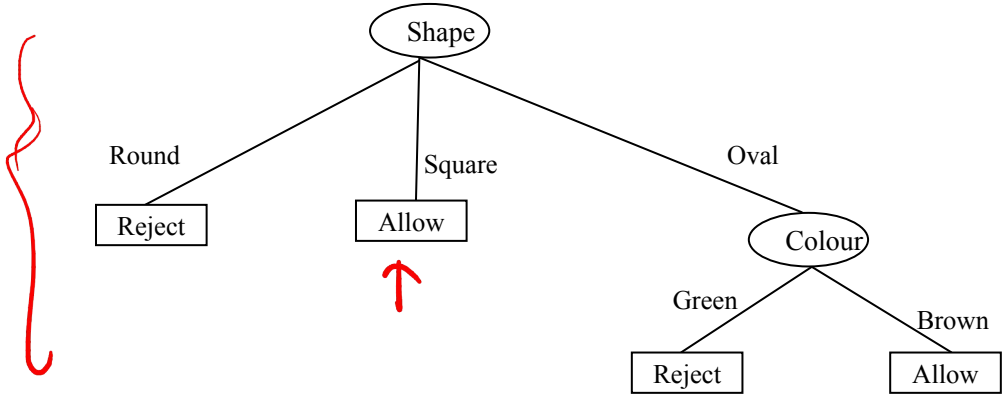
Colour	Size	Action
Black	Big	Allow
Brown	Big	Allow
Green	Big	Allow
Brown	Small	Reject

Use majority voting to label the nodes post pruning

Shape	Colour	Size	Actio
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning

Prune if threshold Gain <=0.85



Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Shape	Colour	Size	Actio
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Post-pruning

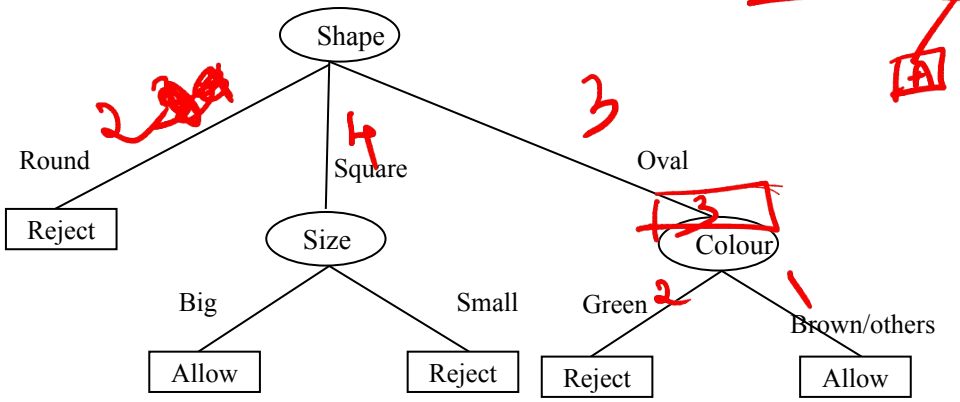
Idea :

1. Post construction, scan the tree bottom-up

2. At every decision node
   Retain the attribute node & evaluate it against the prune set (validation set)
   Remove the attribute node & reevaluate it with the same prune set
   If there is a reduction in error, prune the node else retain the node

3. Repeat this in other branches of the tree

Post Pruning

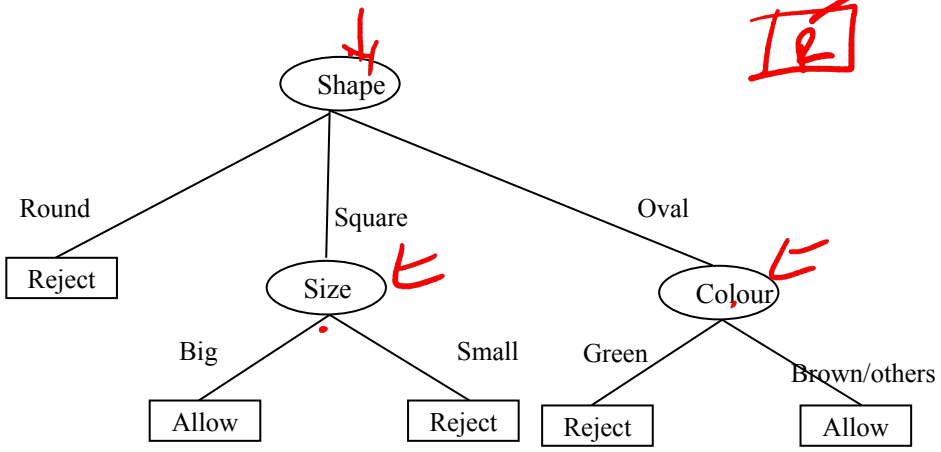


Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

- Error rate is the percentage of tuples misclassified
- Prune set is used to estimate the cost

Shape	Colour	Size	Actio
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Post Pruning



Shape	Colour	Size	Actio
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

After test prune on "Size" and "Color" attributes, its observed that "Colour" is best pruned to get better accuracy!

Prune Size &Predict	Prune Colour & Predict	Above Tree's Prediction	Shape	Colour	Size	Action
Allow	Reject	Allow	Oval	Black	Small	Reject
Reject	Reject	Reject	Round	Brown	Big	Allow
Allow	Allow	Allow	Square	Brown	Big	Allow
Reject	Reject	Reject	Oval	Green	Small	Allow

Practice Exercises (for Students)

1. For this similar problem discussed in class build a decision tree classifier using ID3 algorithm ie., Information gain and entropy measures. Grow the complete decision tree.
2. Can "Object ID" be used as one of key feature to construct the tree? Justify your answer on the results of entropy calculated.
3. Use the test data to construct the confusion matrix and find the precision , recall, F-score.
4. Compare the training accuracy vs test accuracy for the model built in part 1)

	Training Data			
Object ID	Shape	Colour	Size	Action
OB101	Round	Green	Small	Reject
OB102	Square	Black	Big	Allow
OB103	Square	Brown	Big	Allow
OB104	Round	Brown	Small	Reject
OB105	Square	Green	Big	Allow
OB105	Square	Brown	Small	Reject
OB106	Oval	Green	Big	Reject
OB107	Oval	Brown	Small	Allow
OB108	Oval	Green	Small	Reject

	Test Data / Prune Set/ Validation Set			
Object ID	Shape	Colour	Size	Action
OB109	Oval	Black	Small	Reject
OB110	Round	Brown	Big	Allow
OB111	Square	Brown	Big	Allow
OB112	Oval	Green	Small	Allow

Revision for Mid Term Examination

Review of uploaded Question Paper

Important Note to students:

- Check the canvas announcement for any exam related details/ question pattern/ sample questions.
- Handout mapped prescribed book sections, class discussions , practice exercises shared in all the slides must be revised for the exam preparation
- Since the mid term exam is CLOSED BOOK mode, no reference material is allowed and hence formula needs to be learnt by the students
- Queries may not be answered once the exam starts.
- **Kindly Plan ahead to get your preparation queries resolved before the EXAM START DATE of your batch**.

References

- **Chapter 3 - Tom Mitchell**
- Chapter 4 - Introduction to Data Mining by Pang-Ning Tan, Michael Steinbach, Vipin Kumar

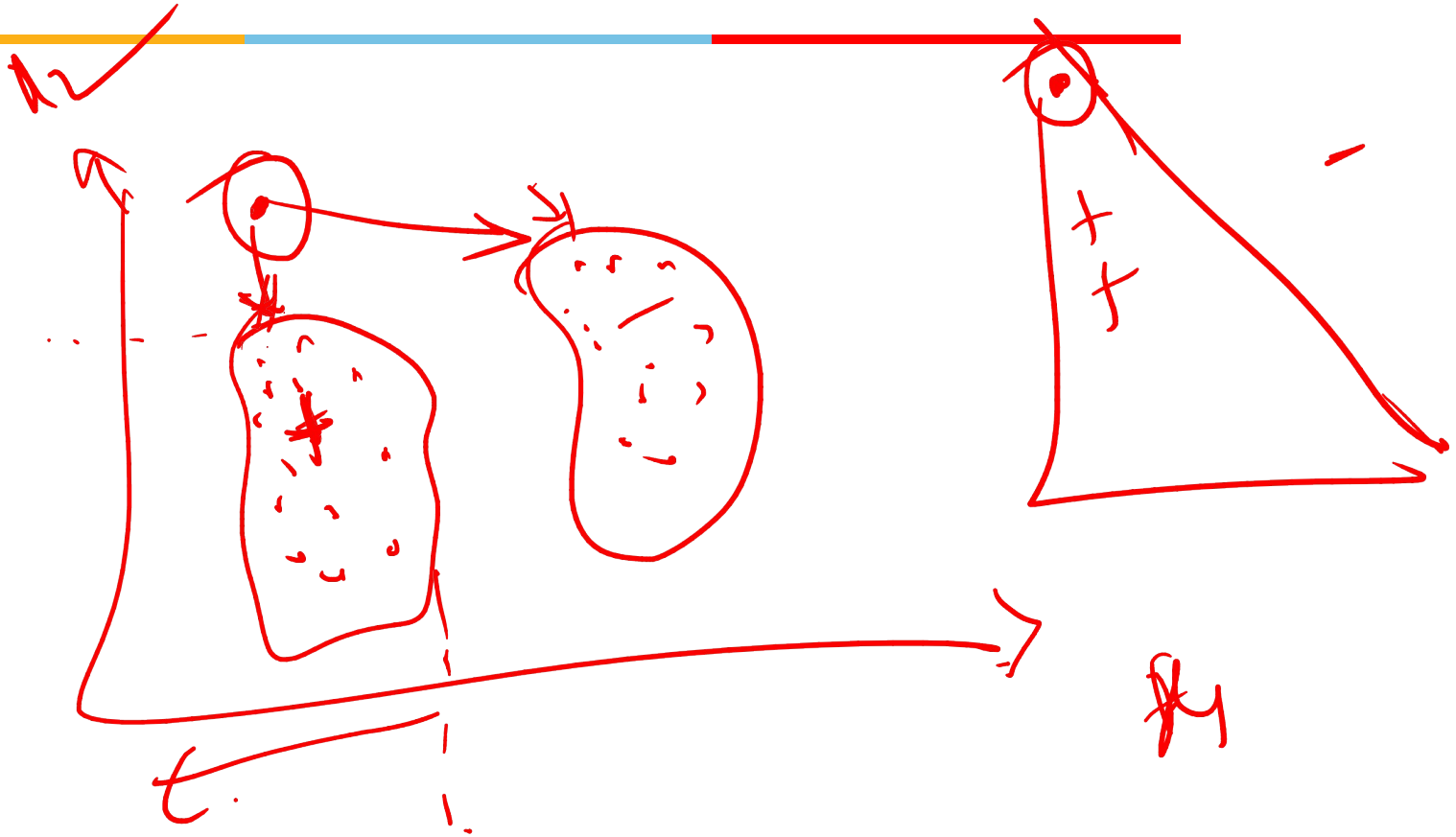
Good References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

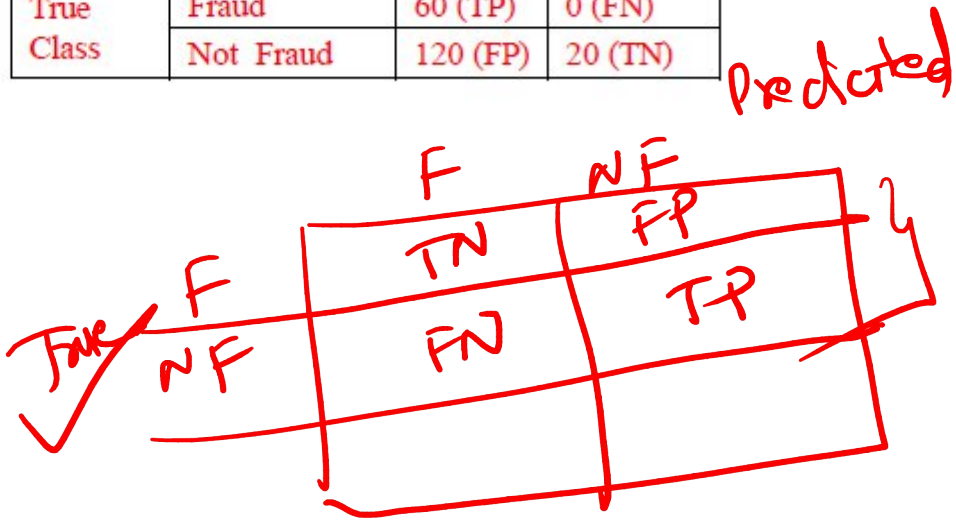
Overfitting

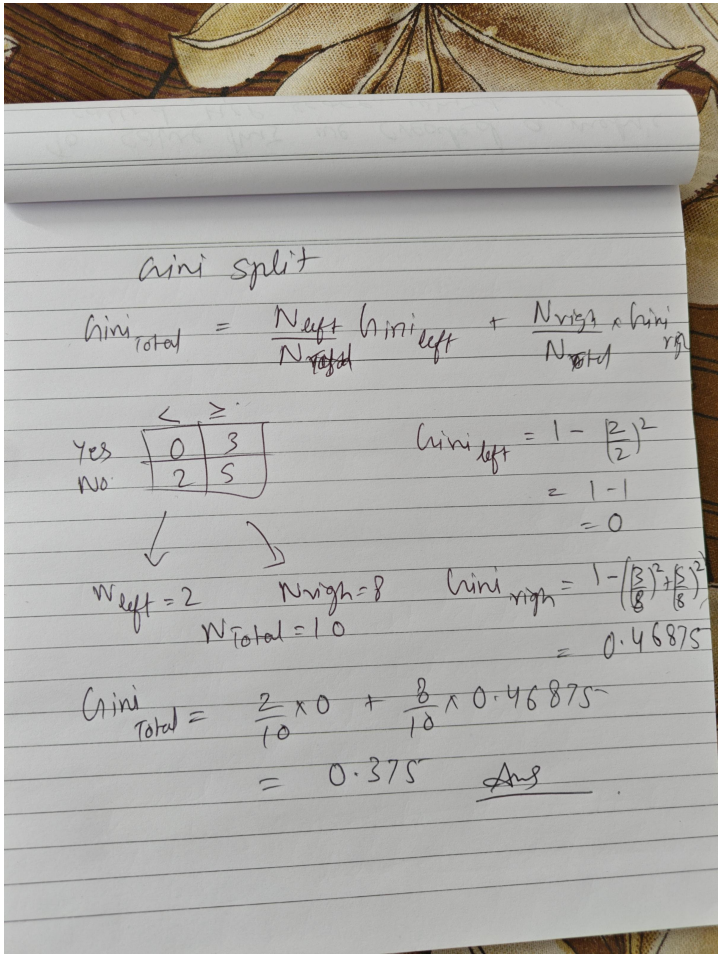
- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- https://www.youtube.com/watch?v=y6SpA2Wuyt8



Metrics with respect to "Not Fraud" Class:

		Predicted Class	
		Fraud	Not Fraud
True Class	Fraud	60 (TP)	0 (FN)
	Not Fraud	120 (FP)	20 (TN)




Gini split
Gini total = N left / N total Gini left + N right / N total x Gini right
< ≥
Yes 0 3
No 2 5
Gini left = 1 - (2/2)^2
= 1 - 1
= 0
N left = 2
N right = 8
N Total = 10
Gini right = 1 - ((3/8)^2 + (5/8)^2)
= 0.46875
Gini Total = 2/10 x 0 + 8/10 x 0.46875
= 0.375
Ans