Data Mining :: Unit-3

(Classification – Decision Trees)

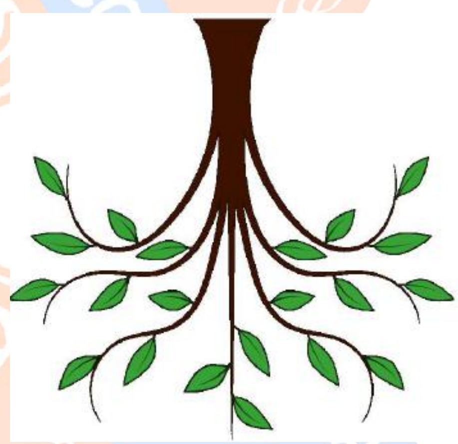
Er. Dinesh Baniya Kshatri
(Lecturer)

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus

Decision Trees

- **A flow-chart-like inverted tree structure**

- **Consists of the following:**
  - Root node
  - Internal nodes
  - Branches
  - Leaf nodes



Anatomy of a Decision Tree

- **The root node is the beginning of the decision tree**
- **Each internal node has an associated splitting predicate**
  - Internal nodes denote a test on an attribute
- **Branches represent the outcome of a test**
- **Leaf nodes represent class labels**
  - A node in a decision tree without children is called a leaf node

Decision Tree Generation

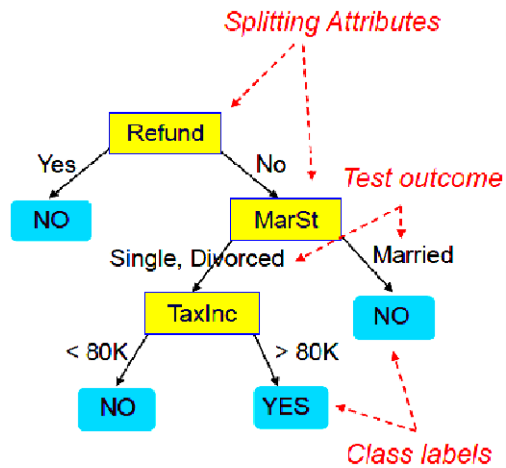
- **Tree Construction**
  - Follows the top-down construction schema
  - Examine training data and find best splitting predicate for the root node
  - Partition training data
  - Recursively partition on each child node based on selected attributes
- **Tree Pruning**
  - Identify and remove branches that reflect noise or outliers

Decision Tree Construction
(Root node = Refund)

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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



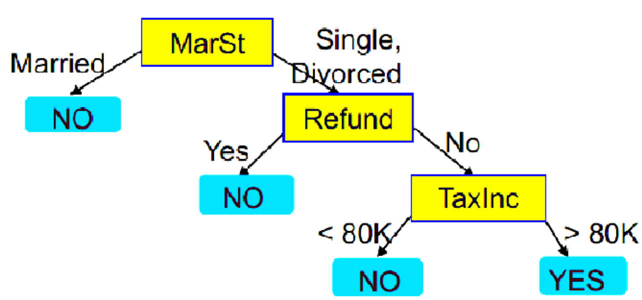
Model: Decision Tree

Decision Tree Construction
(Root node = Marital Status)

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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



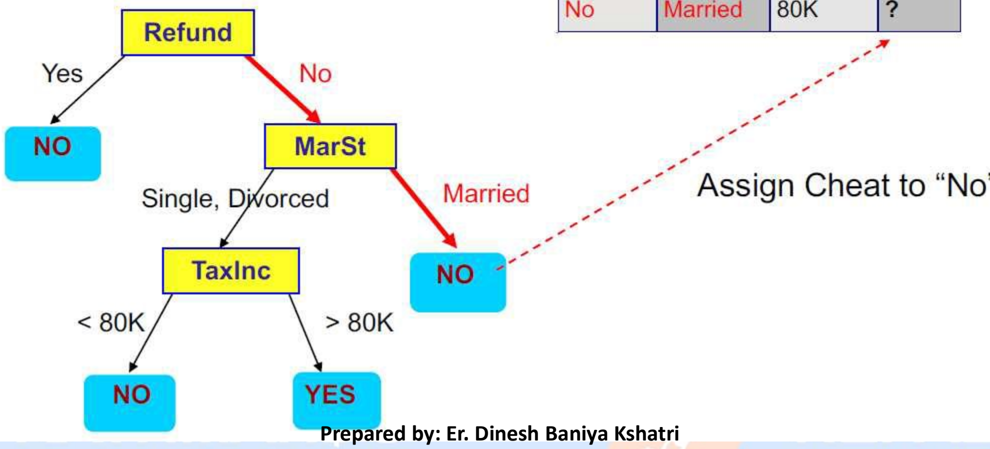
Model: Decision Tree

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

Apply Decision Tree to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decision Tree Induction

- **Goal: Find the tree that has the lowest classification error in the training data (training error)**
  – Finding the best decision tree (lowest training error) is NP-hard
- **In practice: Use Greedy Algorithms**
  – Grow a decision tree by making a series of locally optimum decisions on which attributes to use for partitioning the data
  – Hunt's Algorithm (earliest)
  – ID3 (***Iterative Dichotomiser 3***), CART (***Classification and Regression Tree***), C4.5, SLIQ (***Supervised Learning In Quest***), SPRINT (***S****calable* ***P****a****R****allelizable* ***IN****duction of decision* ***T****rees*)

General Construction Process – [1] (Decision Trees)

- The basic algorithm for **decision tree** construction is a greedy algorithm that constructs **decision trees** in a top-down **recursive** divide-and-conquer manner
- Given a **training set D** of classification data, i.e. a data table with a **distinguished class attribute**
- This **training set** is **recursively partitioned** into smaller subsets (data tables) as the **tree is being built**

General Construction Process – [2] (Decision Trees)

- **Tree** STARTS as a single node (**root**) representing all **training dataset D** (samples)
- We **choose** a **root attribute** from **D**
  It is called a **SPLIT attribute**
- **A branch** is created for **each value as defined in D** of the **node attribute** and **is labeled** by its values and the samples (it means the data table) are **partitioned** accordingly
- The **algorithm** uses the same process **recursively** to form a **decision tree** at **each partition**
- Once an attribute has occurred at a node, it **need not be** considered in any other of the node's descendants

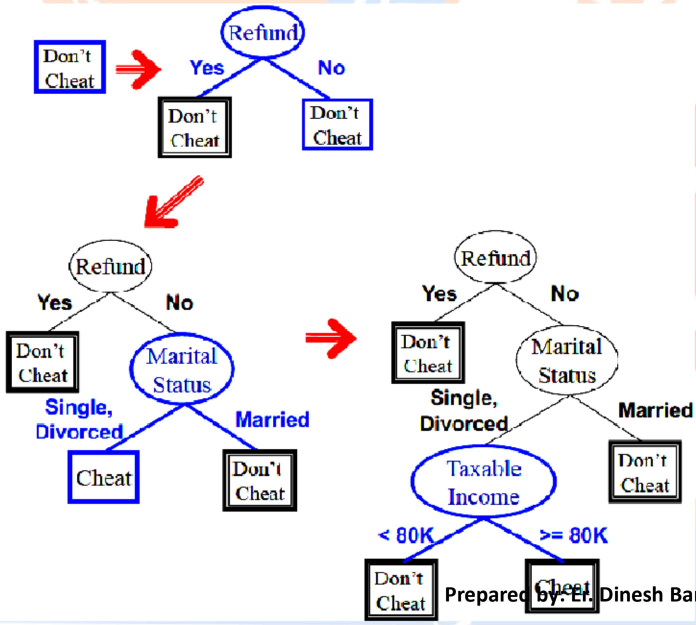
Constructing Decision Trees
(Hunt's Algorithm) – [1]

- X_t: Set of training records that reach a node (t)
- Y = {y_1, …. y_c}: Class labels
- **Step 1:** If all records in (X_t) belong to the same class (y_t), then (t) is a leaf node labeled as (y_t)
- **Step 2:** If (X_t) contains records with the same attribute values, then (t) is a leaf node labeled with the majority class (y_t)

Constructing Decision Trees
(Hunt's Algorithm) – [2]

- **Step 3:** If (X_t) is an empty set, then (t) is a leaf node labeled by the default class (y_d)
- **Step 4:** If (X_t) contains records that belong to more than one class:
  - Select attribute test condition to partition the records into smaller subsets
  - Create a child node for each outcome of test condition
  - Apply algorithm recursively for each child

Hunt's Algorithm in Action



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No
2	No	Married	100K	No
6	No	Married	60K	No
9	No	Married	75K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

Design Issues

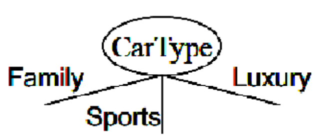
- **Determine how to classify a leaf node:**
  - Assign the majority class
  - If leaf is empty, assign the default class – the class that has the highest popularity (overall or in the parent node)
- **Determine how to split the records:**
  - How to specify the attribute test condition?
  - How to determine the best spilt?
- **Determine when to stop splitting**

How to Specify Attribute Test Condition?

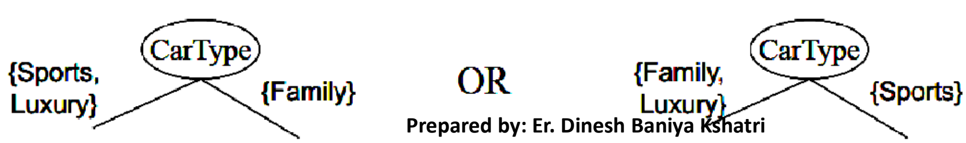
- **Depends on attribute types**
  - Categorical vs. Numeric
    - Categorical attribute (Nominal, Ordinal)
    - Numeric attribute: (Interval, Ratio)
  - Discrete vs. Continuous
- **Depends on number of ways to split**
  - Two-way split
  - Multi-way split

Splitting Based on Nominal Attributes

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

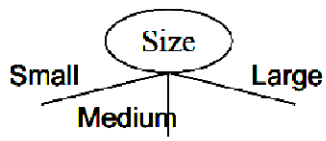


- Binary split: Divides values into two subsets. Need to find optimal partitioning.

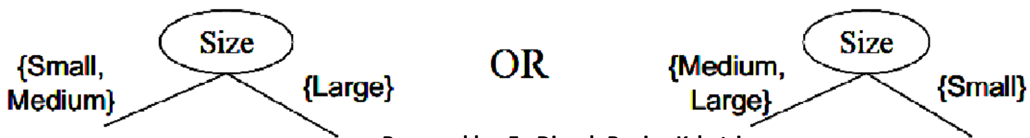


Splitting Based on Ordinal Attributes

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



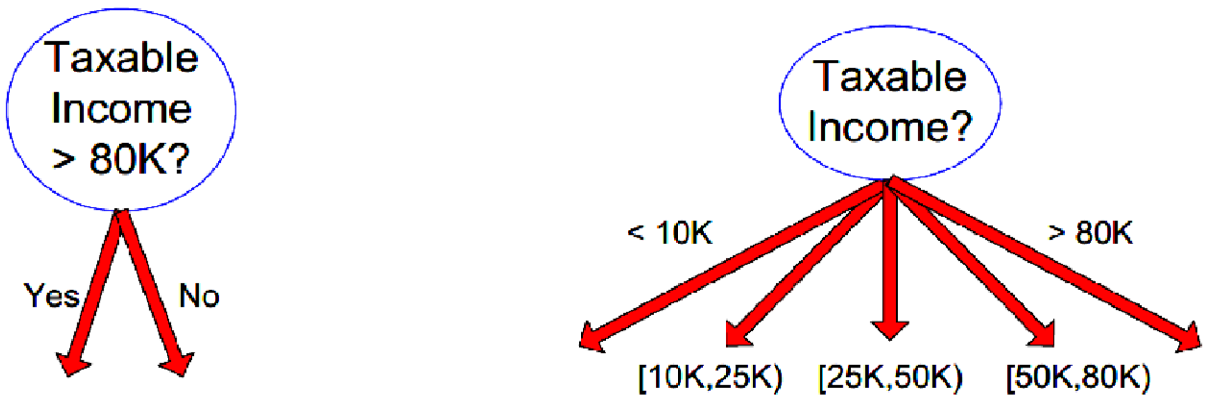
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more computationally intensive

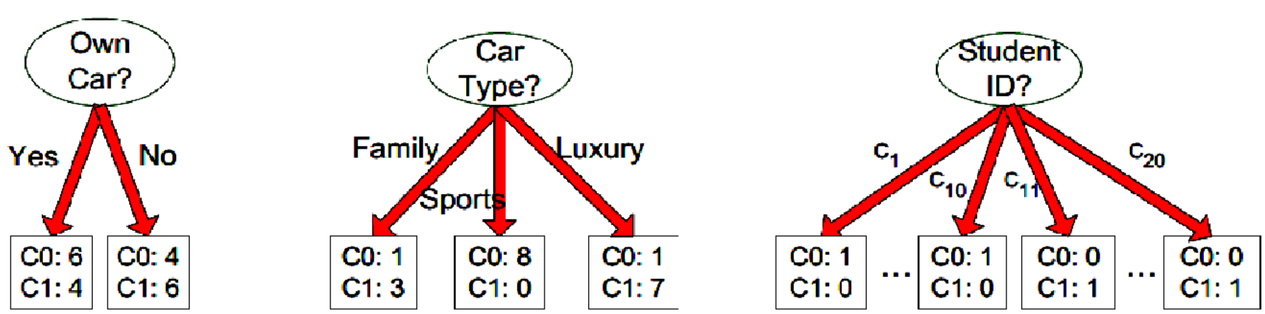
Splitting Based on Continuous Attributes



(i) Binary split (ii) Multi-way split

How to determine the Best Split?

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



Which test condition is the best?

Determining Best Split

- **Crucial Point of Decision Tree Creation**
  - Good choice of the root attribute and internal nodes attributes is vital
  - Bad choice may result, in the worst case, in just another knowledge representation:
    - A relational table rewritten as a tree with class attributes as the leaves

Measures of Node Impurity
(Determining Best Split)

- Gini Index
- Entropy
- Misclassification error

Measure of Impurity: Gini Index

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Splitting Based on Gini Index

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

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

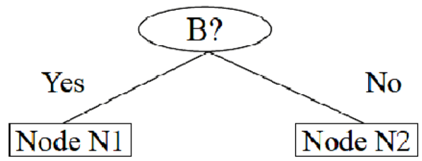
where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes
(Computing Gini Index)

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.

	Parent
C1	6
C2	6
Gini = 0.500	



Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.408$

Gini(N2)
$= 1 - (1/5)^2 - (4/5)^2$
$= 0.32$

	N1	N2
C1	5	1
C2	2	4
Gini=0.371		

Gini(Children)
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

Prepared by: Er. Dinesh Baniya Kshatri 25

Categorical Attributes
(Computing Gini Index)

- For binary values split in two
- For multivalued attributes, for each distinct value, gather counts for each class in the dataset
  - Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

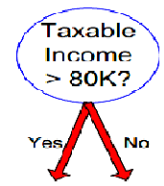
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Prepared by: Er. Dinesh Baniya Kshatri 26

Continuous Attributes – [1]
(Computing Gini Index)

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
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



Continuous Attributes – [2]
(Computing Gini Index)

- 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

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		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	

Alternative Splitting Criteria
(Based on Information Gain) – [1]

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

– Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

– Entropy based computations are similar to the GINI index computations

Why is $(0)(\log_2 0) = 0$?
(Side Note)

- **Making use of L'Hospital's Rule:**

$$\lim_{x\to 0} x\, log_2(x) = \lim_{x\to 0} \frac{\frac{ln(x)}{ln(2)}}{x^{-1}} = \lim_{x\to 0} \frac{\frac{x^{-1}}{ln(2)}}{-x^{-2}} = \lim_{x\to 0} \frac{-x}{ln(2)} = 0$$

Alternative Splitting Criteria – [1]
(Based on Information Gain) – [2]

- Information Gain:

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

Parent Node, p is split into k partitions;
n_i is number of records in partition i

– Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
– Used in ID3 and C4.5
– Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Alternative Splitting Criteria
(Based on Information Gain) – [3]

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions
n_i is the number of records in partition i

– Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
– Used in C4.5
– Designed to overcome the disadvantage of Information Gain

Alternative Splitting Criteria
(Based on Classification Error)

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misclassification error made by a node.
  - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Prepared by: Er. Dinesh Baniya Kshatri 33

Impurity Measures
(Common Ground)

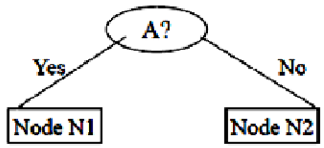
- **All the impurity measures take value zero (minimum)**
  - For the case of a pure node where a single value has probability one

- **All the impurity measures take maximum value**
  - When the class distribution in a node is uniform

Prepared by: Er. Dinesh Baniya Kshatri 34

Misclassification Error vs Gini

	Parent
C1	7
C2	3
Gini = 0.42	
Error = 0.3	



Class	N1	N2
C1	3	4
C2	0	3
Gini = 0.342		
Error = 0.3		

Error(N1)
= 1 – (3/3) = 0

Error(N2)
= 1 – (4/7) = 0.428

Error(Children)
= 3/10 * 0
+ 7/10 * 0.428
= 0.3

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2$
= 0

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2$
= 0.489

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves !!

Prepared by: Er. Dinesh Baniya Kshatri 35

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
  - What to do? majority voting
- Early termination, e.g., when the information gain is below a threshold.

Prepared by: Er. Dinesh Baniya Kshatri 36

Decision Tree Creation

(Training Data)

rec	Age	Income	Student	Credit_rating	Buys_computer(CLASS)
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

Building The Tree – [1]

(Choose "age" as the Root)

age

<=30

income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

>40

income	student	credit	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

31…40

income	student	credit	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Building The Tree – [2]
(Assign Class on 31…40 Age Branch)

age

<=30

income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

>40

income	student	credit	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

31…40

class=yes

Building The Tree – [3]
(Choose “student” on <=30 Age Branch)

age

<=30

student

no

in	cr	cl
h	f	n
h	e	n
m	f	n

yes

in	cr	cl
l	f	y
m	e	y

>40

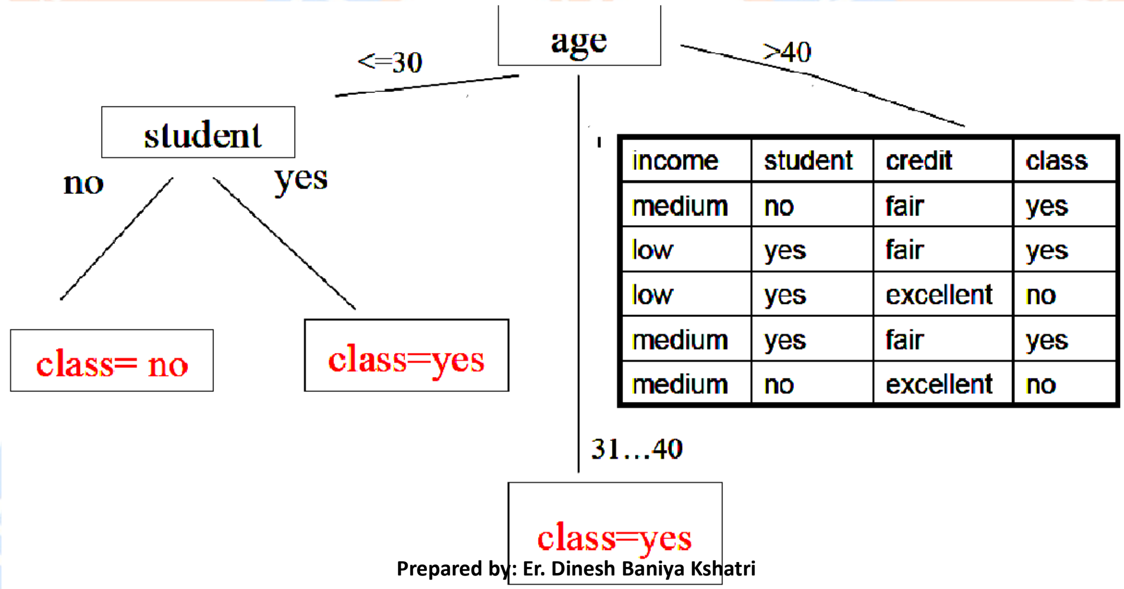
income	student	credit	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

31…40

class=yes

Building The Tree – [4]

(Assign Class to Student Node on <= 30 Age Branch)

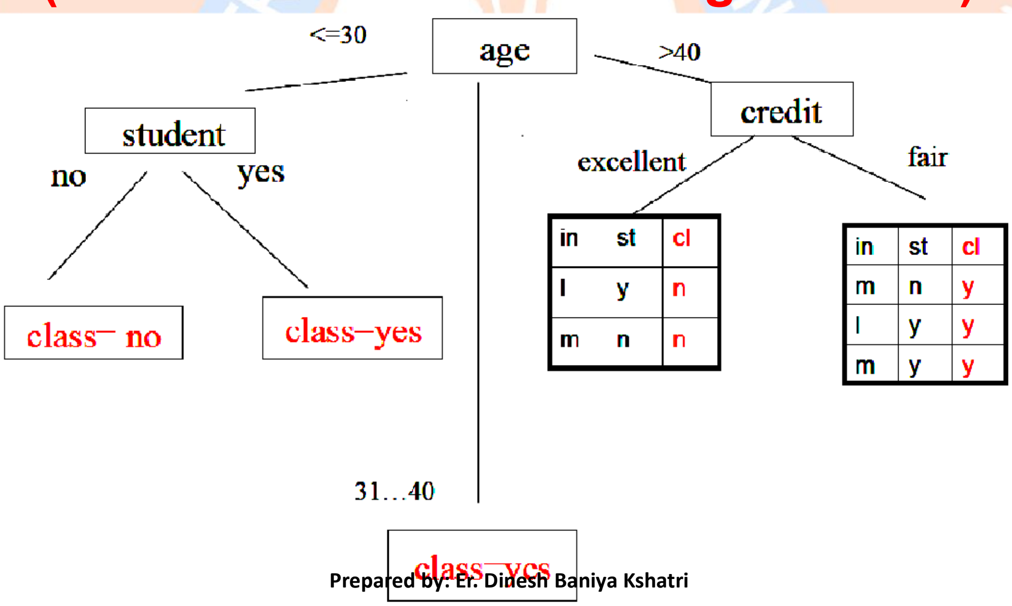


income	student	credit	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

Prepared by: Er. Dinesh Baniya Kshatri 41

Building The Tree – [5]

(Choose "credit" on >40 Age branch)



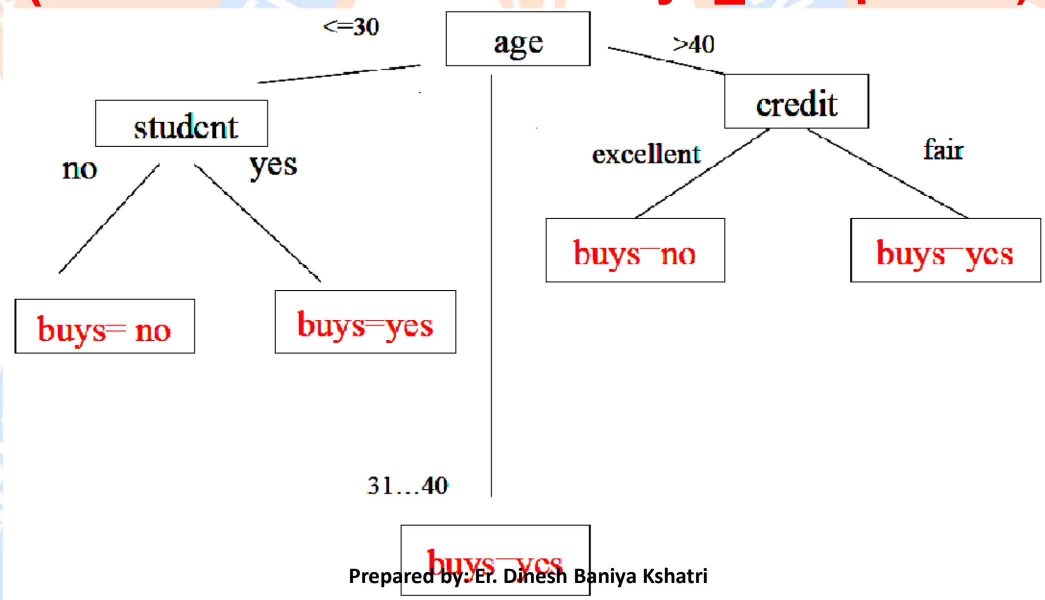
in	st	cl
l	y	n
m	n	n

in	st	cl
m	n	y
l	y	y
m	y	y

Prepared by: Er. Dinesh Baniya Kshatri 42

Building The Tree – [6]

(Finial Tree for Class "buys_computer")



Classification Rule Extraction from Trees

- **Goal: Represent the knowledge in the form of IF-THEN rules**
- **One rule is created for each path from the root to a leaf**
- **The leaf node holds the class prediction**

Classification Rule Extraction – Example

IF *age* = "<=30" AND *student* = "*no*" THEN
buys_computer = "*no*"

IF *age* = "<=30" AND *student* = "*yes*" THEN
buys_computer = "*yes*"

IF *age* = "31…40" THEN
buys_computer = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN
buys_computer = "no"

IF *age* = ">40" AND *credit_rating* = "*fair*" THEN
buys_computer = "*yes*"

Attribute Selection Measures

- **Construction of the tree depends on the order in which root attributes are selected**
  - Different choices produce different trees; some better, some worse
- **Shallower trees are better; they are the ones in which classification is reached in fewer levels**
  - These trees are said to be more efficient and hence termination is reached quickly

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.027$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

The attribute "age" becomes the root.

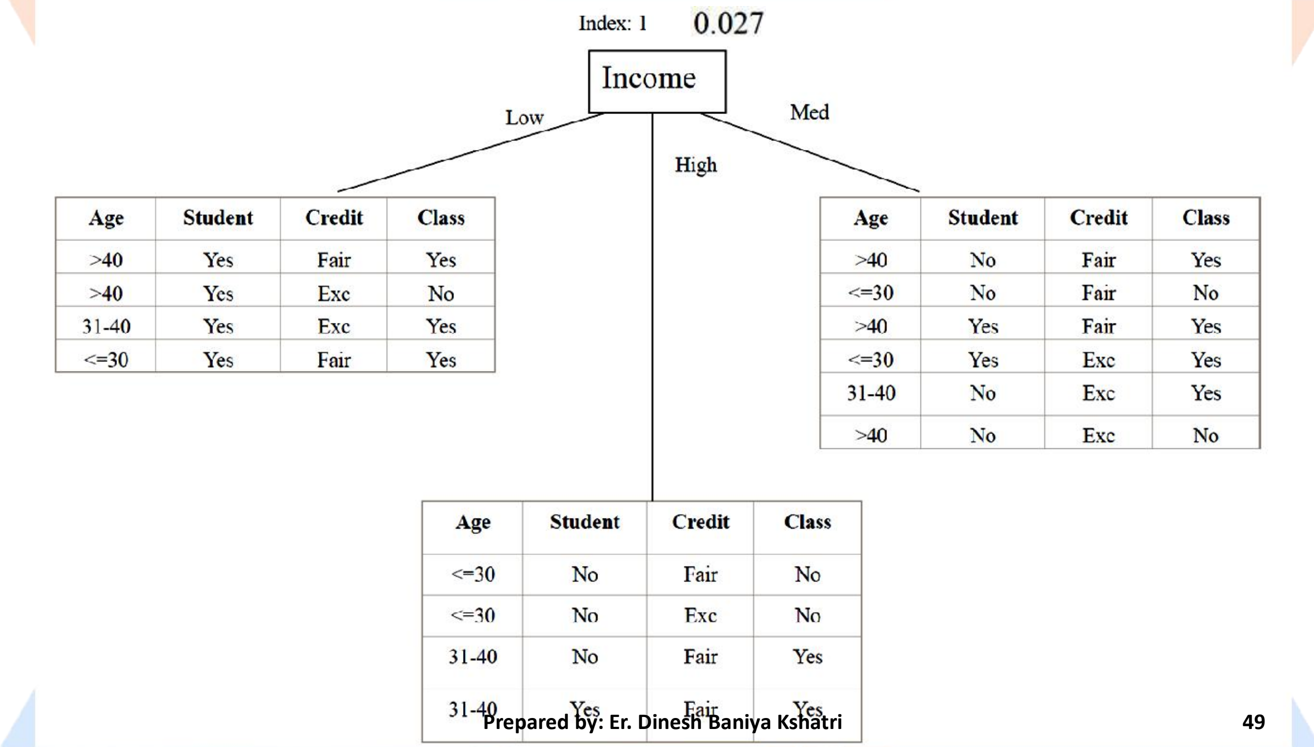
Prepared by: Er. Dinesh Baniya Kshatri 47

Decision Tree Construction (Class Work)

- Choose the feature "buys_computer" as the class attribute
- Perform DT algorithm "by hand" using "Income" as the root attribute
- Use the ID3 algorithm (i.e. use entropy and information gain as the attribute selector)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Prepared by: Er. Dinesh Baniya Kshatri 48



Income = Low:

Age	Student	Credit	Class
>40	Yes	Fair	Yes
>40	Yes	Exc	No
31-40	Yes	Exc	Yes
<=30	Yes	Fair	Yes

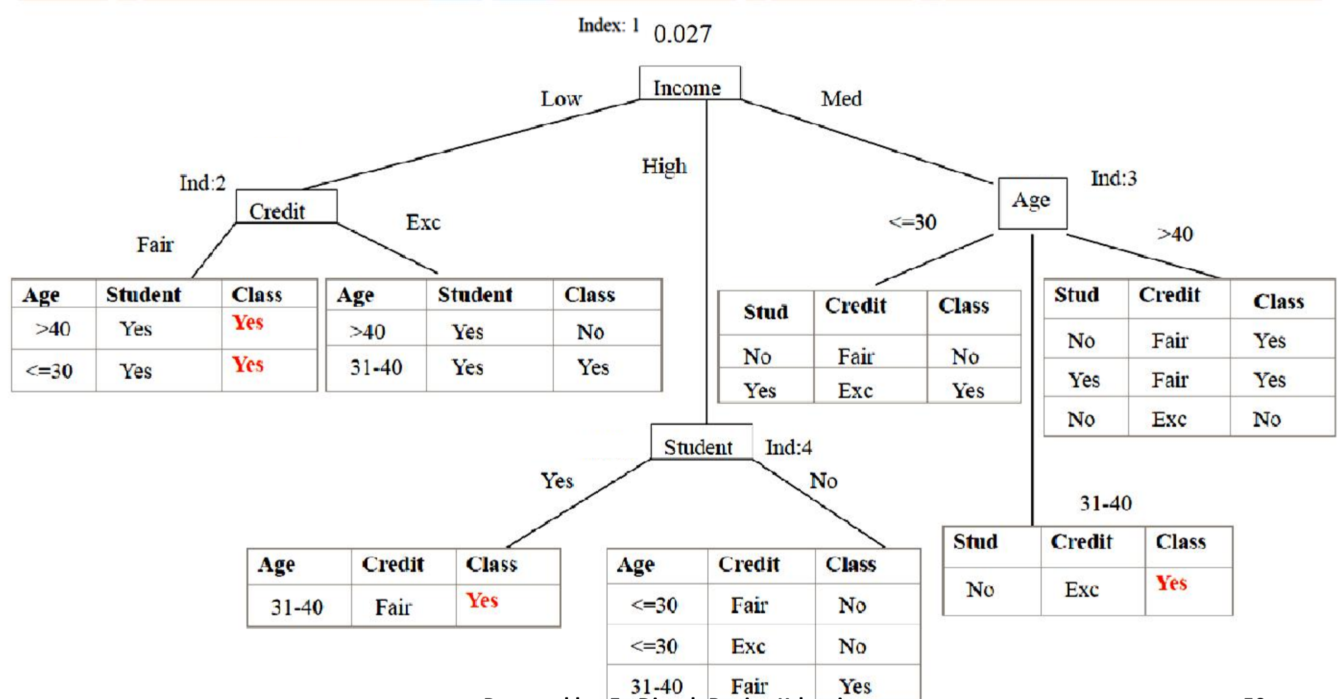
Income = Med:

Age	Student	Credit	Class
>40	No	Fair	Yes
<=30	No	Fair	No
>40	Yes	Fair	Yes
<=30	Yes	Exc	Yes
31-40	No	Exc	Yes
>40	No	Exc	No

Income = High:

Age	Student	Credit	Class
<=30	No	Fair	No
<=30	No	Exc	No
31-40	No	Fair	Yes
31-40	Yes	Fair	Yes

Prepared by: Er. Dinesh Baniya Kshatri 49



Credit = Fair:

Age	Student	Class
>40	Yes	Yes
<=30	Yes	Yes

Credit = Exc:

Age	Student	Class
>40	Yes	No
31-40	Yes	Yes

Age <=30:

Stud	Credit	Class
No	Fair	No
Yes	Exc	Yes

Age >40:

Stud	Credit	Class
No	Fair	Yes
Yes	Fair	Yes
No	Exc	No

Age 31-40:

Stud	Credit	Class
No	Exc	Yes

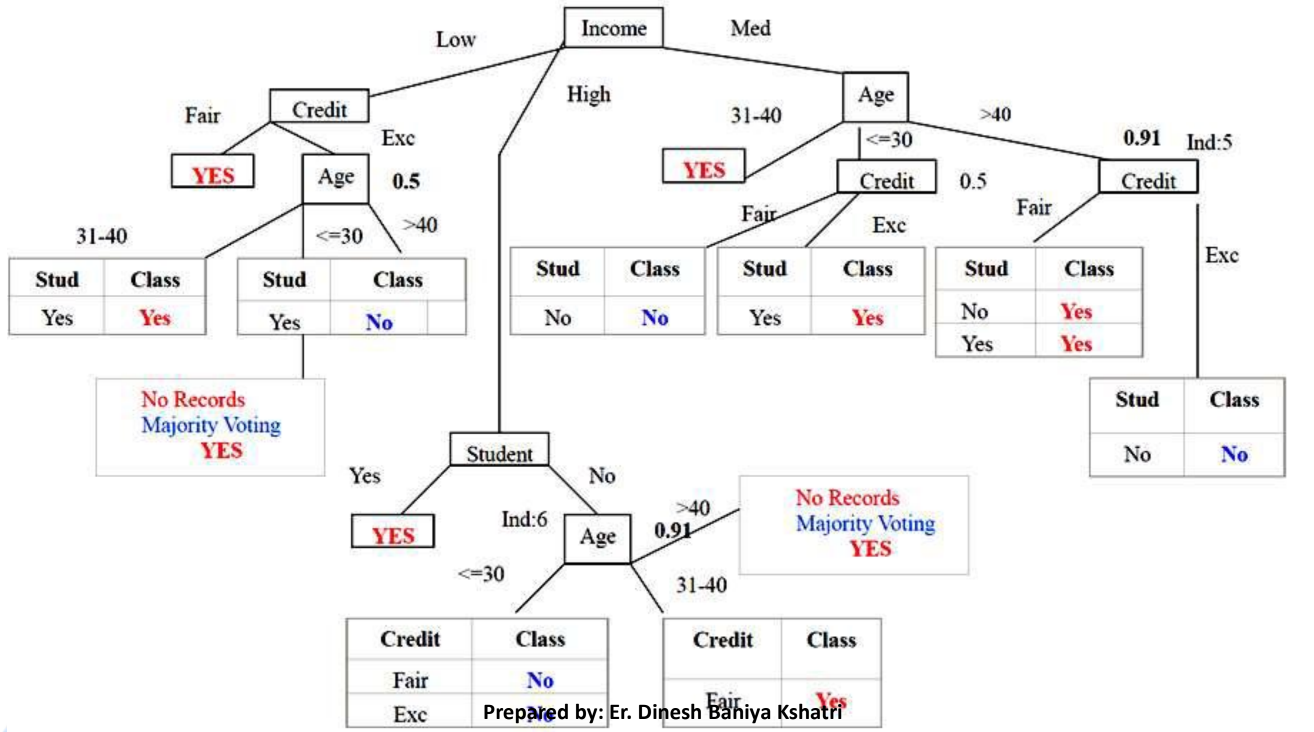
Student = Yes:

Age	Credit	Class
31-40	Fair	Yes

Student = No:

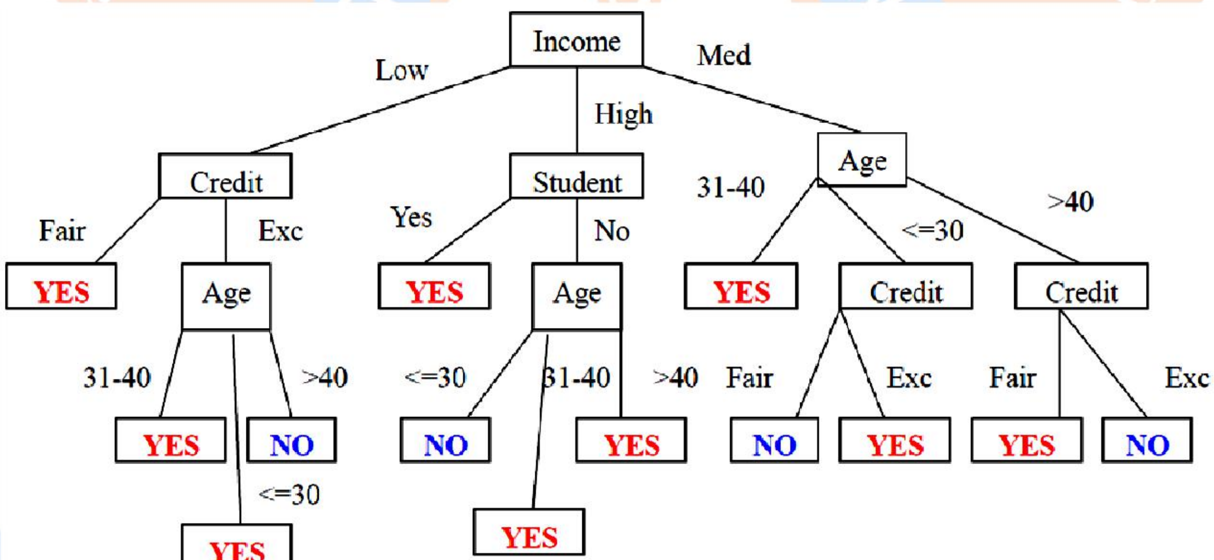
Age	Credit	Class
<=30	Fair	No
<=30	Exc	No
31-40	Fair	Yes

Prepared by: Er. Dinesh Baniya Kshatri 50



Complete Tree

(Root Attribute = Income)



Information Gain at Each Tree Level
(Root Attribute = Income)

1. Original Table:

Class P: *buys_computer* = yes; Class N: *buys_computer* = No

I(P,N) = -P/P+N log ₂ (P/P+N) – N/P+N log₂ N/P+N-------(equation 1)

I(P,N) = I(9,5) = (-9/9+5) log ₂(9/9+5) – (5/9+5) log₂ (5/9+5)

= 0.940

2. Index:1

Income	Pi	Ni	I(Pi,Ni)
Low	3	1	0.8111
Med	4	2	0.9234
High	2	2	1

Substituting the values in eq.2 we get,
E(Income) = 0.2317 + 0.3957 + 0.2857 = 0.9131
Gain (Income) = I(P,N) – E(Income)
= 0.940 – 0.9131 = 0.027

E(Income) = 4/14 I(3,1) + 6/14 I(4,2) + 4/14 I(2,2)------------(eq.2)
I(3,1) = 0.8111 (Using equation 1)
I(4,2) = 0.9234 (Using equation 1)
I(2,2) = 1

Similarly we can calculate Information gain of tables at each stage.

Example – 2
(Problem Description) – [1]

- **Taste, Temperature and Texture are exploratory variables and Eat (Yes/No) is the target variable**
- **Need to construct a top-down decision tree that splits the dataset and finally forms a pure group**
- **Use the ID3 algorithm to find the decision tree**

	Taste	Temperature	Texture	Eat
0	Salty	Hot	Soft	No
1	Spicy	Hot	Soft	No
2	Spicy	Hot	Hard	Yes
3	Spicy	Cold	Hard	No
4	Spicy	Hot	Hard	Yes
5	Sweet	Cold	Soft	Yes
6	Salty	Cold	Soft	No
7	Sweet	Hot	Soft	Yes
8	Spicy	Cold	Soft	Yes
9	Salty	Hot	Hard	Yes

Example – 2
(Calculating Parent Entropy)

$$E_o = \sum_{i=1}^{2}\left[-P_i log_2(P_i)\right]$$
$$= \frac{-4}{10}log_2(\frac{4}{10}) - \frac{-6}{10}log_2(\frac{6}{10})$$
$$= 0.971$$

$No.\ of\ 'NO' \rightarrow 4$

$No.\ of\ 'YES' \rightarrow 6$

$No.\ of\ objects \rightarrow 10$

Example – 2
(Calculating Entropy & IG due to Taste)

$$E_{Salty} = -\frac{N_1}{N}S_1$$
$$= -\frac{3}{10}\left[\underbrace{\frac{1}{3}log_2(\frac{1}{3})}_{Yes} + \underbrace{\frac{2}{3}log_2(\frac{2}{3})}_{NO}\right]$$
$$= 0.2754$$

$$E_{Spicy} = -\frac{N_2}{N}S_2$$
$$= -\frac{5}{10}\left[\frac{3}{5}log_2(\frac{3}{5}) + \frac{2}{5}log_2(\frac{2}{5})\right]$$
$$= 0.4854$$

$$E_{Sweet} = -\frac{N_3}{N}S_3$$
$$= -\frac{2}{10}\left[\frac{2}{2}log_2(\frac{2}{2})\right]$$
$$= 0$$

$$E_{Taste} = E_{Salty} + E_{Spicy} + E_{Sweet}$$
$$= 0.7608$$

$$IG_{Taste} = E_o - E_{Taste}$$
$$= 0.971 - 0.7608$$
$$= 0.21$$

Example – 2
(Calculating Entropy & IG due to Temperature)

$$E_{Hot} = -\frac{N_1}{N}S_1$$

$$= -\frac{6}{10}\left[\frac{4}{6}log_2\left(\frac{4}{6}\right) + \frac{2}{6}log_2\left(\frac{2}{6}\right)\right]$$

$$= 0.5509$$

$$E_{Cold} = -\frac{N_2}{N}S_2$$

$$= -\frac{4}{10}\left[\frac{2}{4}log_2\left(\frac{2}{4}\right) + \frac{2}{4}log_2\left(\frac{2}{4}\right)\right]$$

$$= 0.4$$

$$E_{Temp.} = E_{Hot} + E_{Cold}$$

$$= 0.9509$$

$$IG_{Temp.} = E_o - E_{Temp.}$$

$$= 0.971 - 0.9509$$

$$= 0.02$$

Example – 2
(Calculating Entropy & IG due to Texture)

$$E_{Soft} = -\frac{N_1}{N}S_1$$

$$= -\frac{6}{10}\left[\frac{3}{6}log_2\left(\frac{3}{6}\right) + \frac{3}{6}log_2\left(\frac{3}{6}\right)\right]$$

$$= 0.6$$

$$E_{Hard} = -\frac{N_2}{N}S_2$$

$$= -\frac{4}{10}\left[\frac{1}{4}log_2\left(\frac{1}{4}\right) + \frac{3}{4}log_2\left(\frac{3}{4}\right)\right]$$

$$= 0.3245$$

$$E_{Temp.} = E_{Soft} + E_{Hard}$$

$$= 0.9245$$

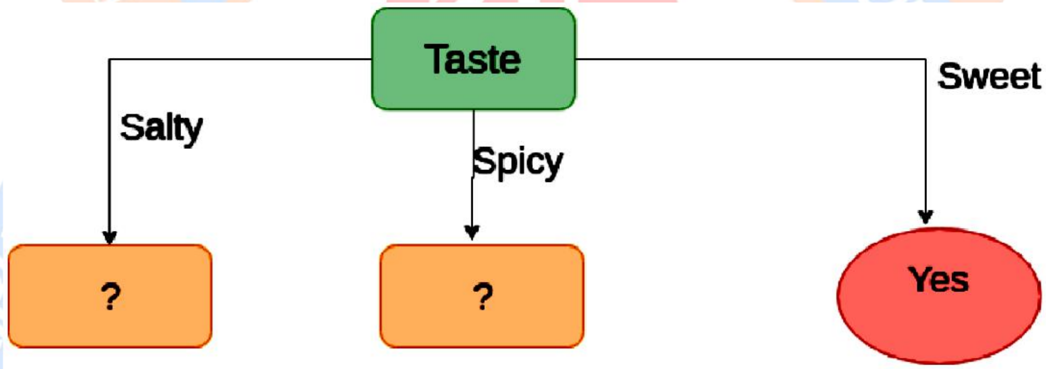
$$IG_{Temp.} = E_o - E_{Text.}$$

$$= 0.971 - 0.9245$$

$$= 0.05$$

Example – 2
(1st Level of Decision Tree)

- Split the data based on Taste, as it has highest information gain



Example – 2
(Attributes to Split at Second Level)
(Under the Salty Branch)

- Need to find the attributes to split at second level nodes for the Salty branch

	Temperature	Texture	Eat
0	Hot	Soft	No
1	Cold	Soft	No
2	Hot	Hard	Yes

Example – 2
(Calculating Entropy & IG due to Temperature)
(Under the Salty Branch)

$$E_{1a} = \frac{2}{3}log_2\left(\frac{2}{3}\right) + \frac{1}{3}log_2\left(\frac{1}{3}\right)$$
$$= 0.9182$$

$$E_{Hot} = -\frac{2}{3}\left[\frac{1}{2}log_2\left(\frac{1}{2}\right) + \frac{1}{2}log_2\left(\frac{1}{2}\right)\right]$$
$$= 0.67$$

$$E_{Cold} = -\frac{1}{3}\left[\frac{1}{1}log_2\left(\frac{1}{1}\right)\right]$$
$$= 0$$

$$E_{Temp.} = 0.67$$

$$IG_{Temp.} = E_{1a} - E_{Temp.}$$
$$= 0.9182 - 0.67$$
$$= 0.2482$$

Prepared by: Er. Dinesh Baniya Kshatri

Example – 2
(Calculating Entropy & IG due to Texture)
(Under the Salty Branch)

$$E_{Soft} = -\frac{2}{3}\left[\frac{2}{2}log_2\left(\frac{2}{2}\right)\right]$$
$$= 0$$

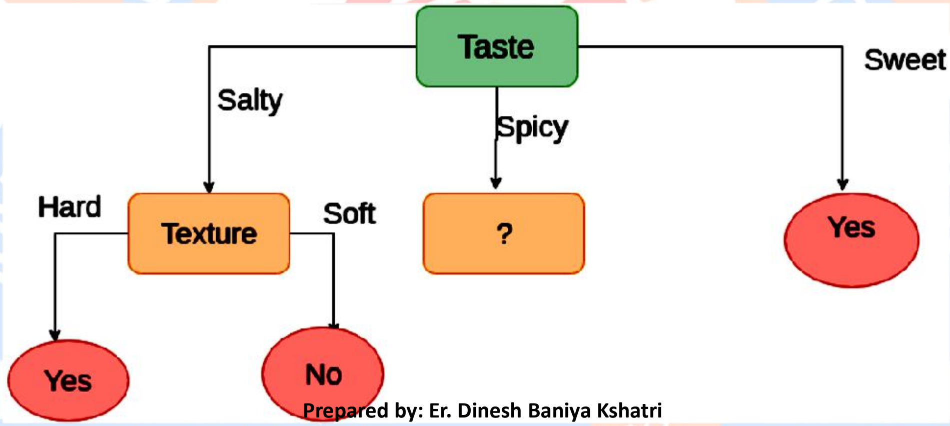
$$E_{Hard} = -\frac{1}{3}\left[\frac{1}{1}log_2\left(\frac{1}{1}\right)\right]$$
$$= 0$$

$$IG_{Text.} = E_{1a} - E_{Text.}$$
$$= 0.9182 - 0$$
$$= 0.9182$$

Prepared by: Er. Dinesh Baniya Kshatri 62

Example – 2
(Partial 2nd Level of Decision Tree)

- **Splitting based on texture sounds a good option, as it has higher information gain.**



Example – 2
(Attributes to Split at Second Level)
(Under the Spicy Branch)

- **Need to find the attributes to split at second level nodes for the Spicy branch**

	Temperature	Texture	Eat
0	Hot	Soft	No
1	Hot	Hard	Yes
2	Cold	Hard	No
3	Hot	Hard	Yes
4	Cold	Soft	Yes

Example – 2
(Calculating Entropy & IG due to Temperature)
(Under the Spicy Branch)

$$E_{1b} = -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)$$
$$= 0.9709$$

$$E_{Hot} = -\frac{3}{5}\left[\frac{1}{3}log_2\left(\frac{1}{3}\right) + \frac{2}{3}log_2\left(\frac{2}{3}\right)\right]$$
$$= 0.5509$$

$$E_{Cold} = -\frac{2}{5}\left[\frac{1}{2}log_2\left(\frac{1}{2}\right) + \frac{1}{2}log_2\left(\frac{1}{2}\right)\right]$$
$$= 0.4$$

$$E_{Temp.} = 0.9509$$

$$IG_{Temp.} = E_{1b} - E_{Temp.}$$
$$= 0.9709 - 0.9509$$
$$= 0.02$$

Example – 2
(Calculating Entropy & IG due to Texture)
(Under the Spicy Branch)

$$E_{Soft} = -\frac{2}{5}\left[\frac{1}{2}log_2\left(\frac{1}{2}\right) + \frac{1}{2}log_2\left(\frac{1}{2}\right)\right]$$
$$= 0.4$$

$$E_{Hard} = -\frac{3}{5}\left[\frac{1}{3}log_2\left(\frac{1}{3}\right) + \frac{2}{3}log_2\left(\frac{2}{3}\right)\right]$$
$$= 0.5509$$

$$E_{Text.} = 0.9509$$

$$IG_{Text.} = E_{1b} - E_{Text.}$$
$$= 0.9709 - 0.9509$$
$$= 0.02$$

Example – 2
(Splitting Decision under Spicy Branch)

- **Both the attributes (Temperature and Texture) generated same Information Gain**
  – So, can split with any attribute
  – Temperature has been chosen as the splitting parameter
- **Tables left after Temperature split, for both branches are:**

	Texture	Eat
0	Soft	No
1	Hard	Yes
2	Hard	Yes

Table : Spicy-Temperature-Hot path

	Texture	Eat
0	Hard	No
1	Soft	Yes

Table : Spicy-Temperature-Cold path

Example – 2
(Final Decision Tree)