Unit-2

1. Supervised Learning Algorithm
   a) Linear Regression
   b) Logistic Regression
   c) Decision Tree
   d) Support Vector Machine
   e) KNN
   f) Naïve Bayes

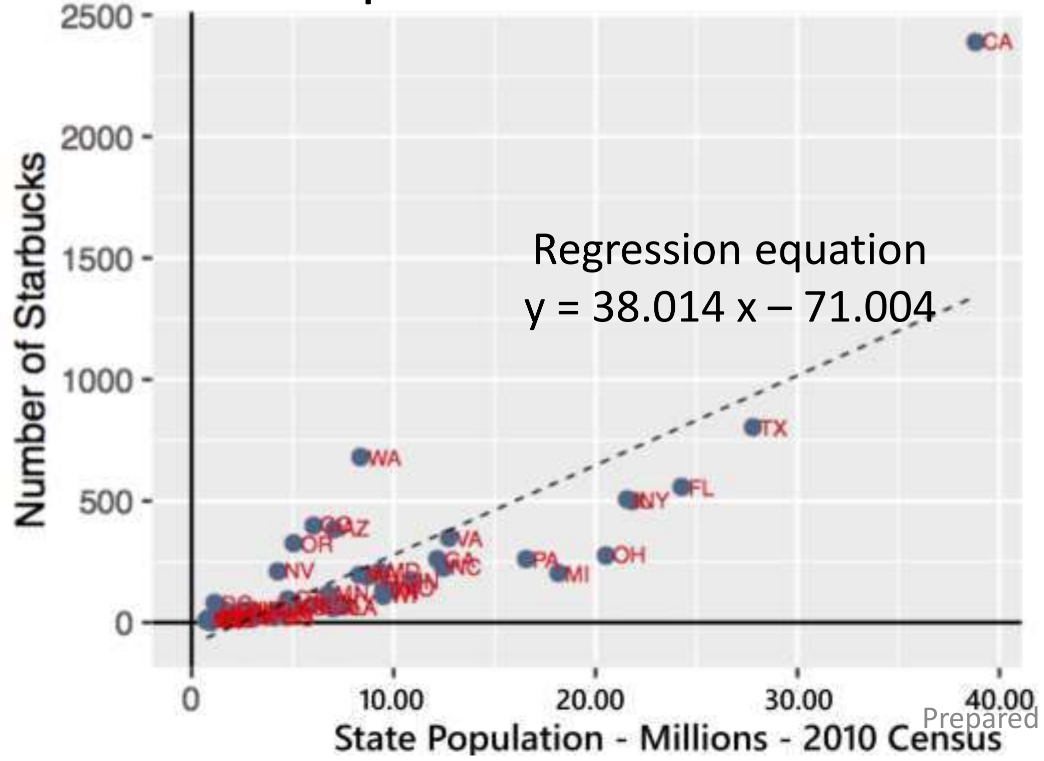
2. Validation Technique
   1. K-Fold Cross Validation
   2. Boot Strapping

Linear Regression

Linear Regression is a way of predicting an unknown variable using results that you do know. If you have a set of x and y values, you can use a regression equation to make a straight line relating the x and y.

The chart below shows an example of linear regression using real world data.

State Population Vs Number of Starbucks



It shows the relationship between the population of states within the United States, and the number of Starbucks (a coffee chain restaurant) within that state.

Linear Regression

The result of the regression equation is that I can predict the number of Starbucks within a state by taking the population (in millions), multiplying it by 38.014, and subtracting 71.004.

So, if there was a state with 10 million people, it is predicted that there would be just over 309 Starbucks within the state.

$y = 38.014 \times 10 - 71.004$

$y = 309.136 \approx 309$

Linear Regression is one of the simplest models that use a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set.

Linear Regression Model Prediction

A linear model makes a prediction by simply computing a weighted sum of the input features, plus a constant called the bias term (also called the intercept term), as shown in the Equation below

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots + \theta_n x_n$$

In this equation:
y is the predicted value
n is the number of feature
x_i is the i^th feature value
Θ_j is the j^th model parameter (including the bias term θ_0 and the feature weights θ_1, θ_2,…, θ_n)

Linear Regression Model Prediction

- Vectorized form: – This can be written more concisely using a vectorized form, as shown in Equation below

$$\hat{y} = h_\theta(x) = \theta . x$$

Instance's feature vector, containing x_0 to x_n, with x_0 always equal to 1

Hypothesis function, using the model parameter θ

model's parameter vector, containing the bias term and the feature weights

$\theta . x$ is the dot product of the vectors θ and x, which is of course equal to

$$\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \ldots + \theta_n x_n$$

Applications of Linear Regression

- Real estate: A simple linear regression analysis can be used to model residential home prices as a function of the home's living area. Such a model helps set or evaluate the list price of a home on the market. The model could be further improved by including other input variables such as number of bathrooms, number of bedrooms, lot size, school district ranking, crime statistics, and property taxes.
- Demand forecasting: Businesses and governments can use linear regression models to predict demand for goods and services. E.g., Restaurant chain
- Medical: A linear regression model can be used to analyse the effect of a proposed radiation treatment on reducing tumor sizes. Input variables might include duration of a single radiation treatment, frequency of radiation treatment, and patient attributes such as age or weight.

Logistic Regression

Introduction

- When the outcome variable is categorical in nature, logistic regression can be used to predict the likelihood of an outcome based on the input variables.

- Although logistic regression can be applied to an outcome variable that represents multiple values, the next side discussion examines the case in which the outcome variable represents two values such as true/false, pass/fail, or yes/no

Logistic Regression

For example

Logistic regression model can be built to determine if a person will or will not purchase a new automobile in the next 12 months.

- The Training set could include input variables for a person’s age, income, and gender as well as the age of an existing automobile.
- The training set would also include the outcome variable on whether the person purchased a new automobile over a 12-month period.

Logistic regression model provides the likelihood or probability of a person making a purchase in the next 12 months.

Logistic Regression

Use Cases

Medical: Develop a model to determine the likelihood of a patient's successful response to a specific medical treatment or procedure.

Input variables: age, weight, blood pressure, cholesterol levels

Finance: Using a loan applicant's credit history and the details on the loan, determine the probability that an applicant will default on the loan. Based on the prediction, the loan can be approved or denied, or the terms can be modified.

Logistic Regression

Use Cases

Marketing: Determine a wireless customer’s probability of switching carriers based on age, number of family members on the plan, months remaining on the existing contract, and social network contacts. With such insight, target the high probability customers with appropriate offers to prevent churn.

Engineering: Based on operating conditions and various diagnostic measurements, determine the probability of a mechanical part experiencing a malfunction or failure.

With this probability estimate, schedule the appropriate preventive maintenance activity.

Logistic Regression

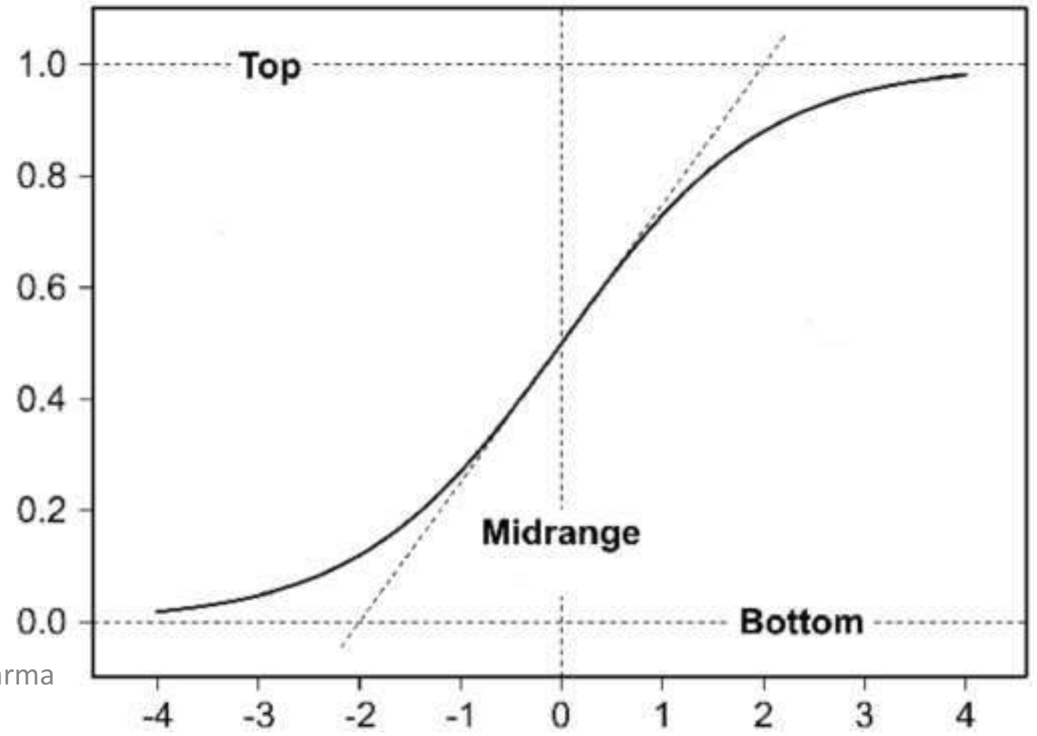
Model Description

Logistic regression is based on the logistic function $f(y)$,

$$f(y) = \frac{e^y}{1+e^y} \text{ for } -\infty < y < \infty$$

$As\ y \longrightarrow \infty$ then $f(y) \longrightarrow 1$ and as $y \longrightarrow -\infty$ then $f(y) \longrightarrow 0$

The value of the logistic function $f(y)$ varies from 0 to 1 as y increases. Because the range of $f(y)$ is (0, 1), the logistic function appears to be an appropriate function to model the probability of a particular outcome occurring.



Decision Tree

- Concept of Decision Tree
- Use of Decision Tree to classify data
- Basic algorithm to build Decision Tree
    - Some illustrations
- Concept of Entropy
    - Basic concept of entropy in information theory
    - Mathematical formulation of entropy
    - Calculation of entropy of a training set
- Decision Tree induction algorithms
    - ID3
    - CART
    - C4.5

How to Make Decision Tree

1. Determine Your Initial Question
2. Determine Your Decision and Unknown
3. Determine the Values
4. Determine the Probabilities
5. Calculate the Weighted Value
6. Calculate the Net Benefit of Each Decision

Determine Your Initial Question

- Should I relocate my ice cream shop or remain in my current location?

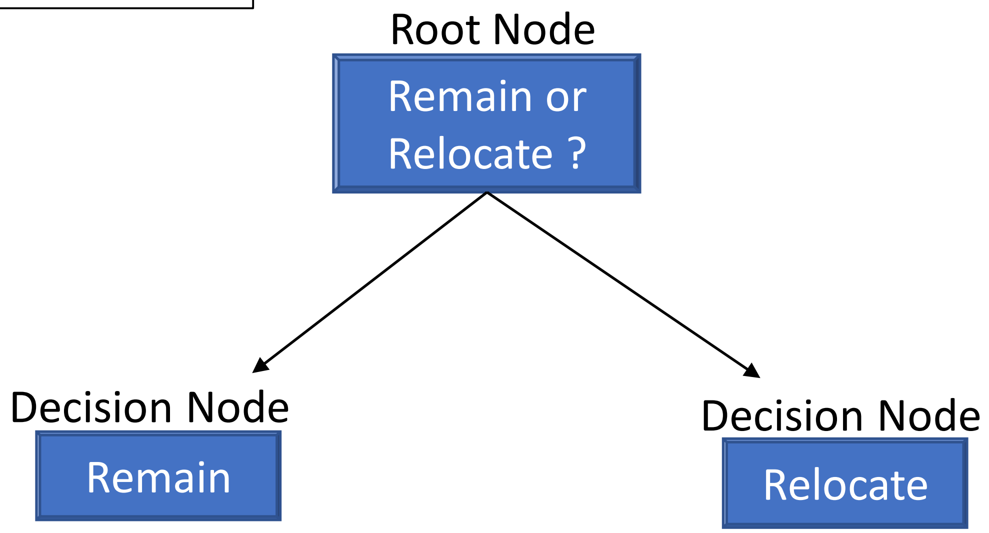
Determine Your Decision and Unknown

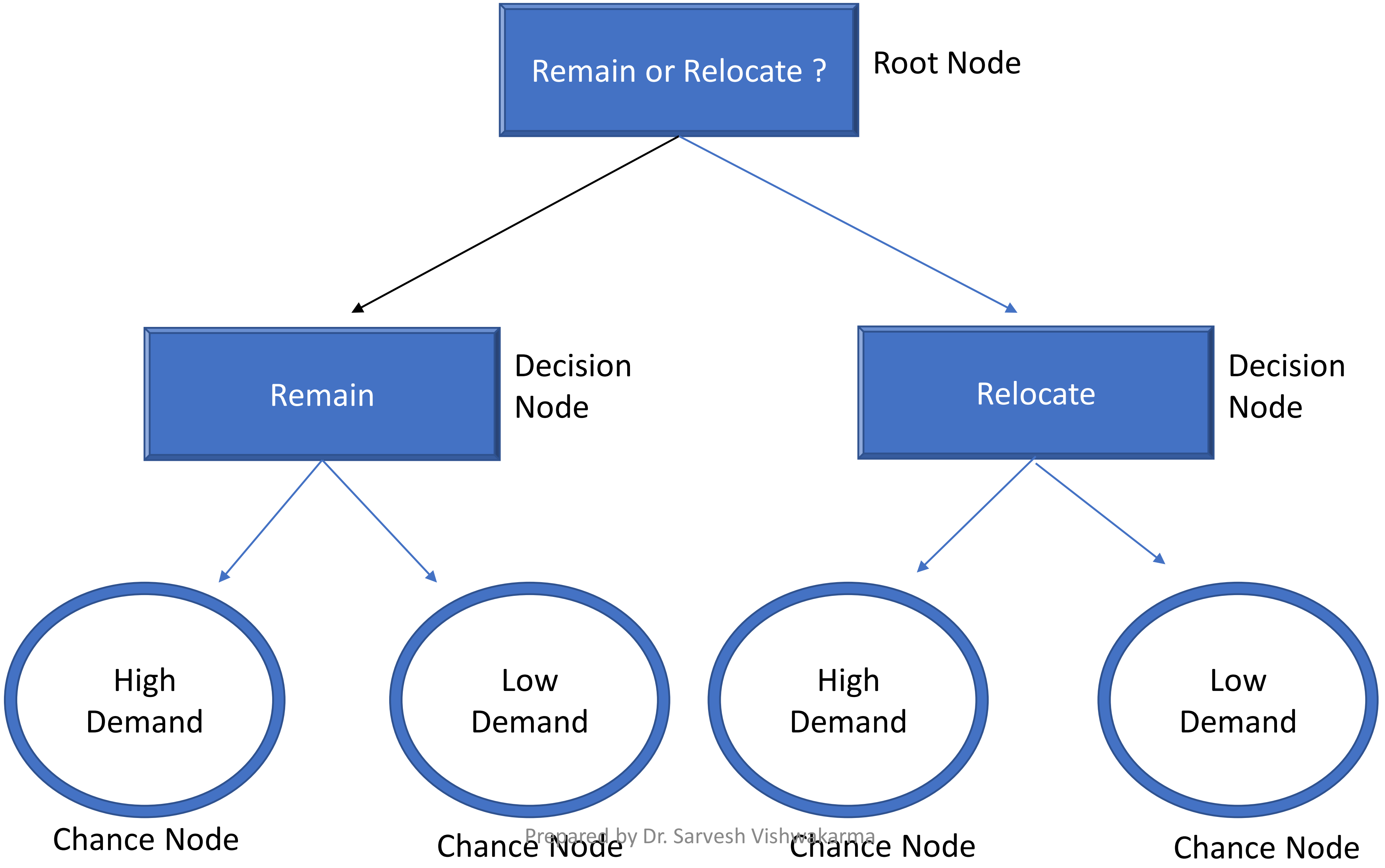
- In this scenario there is only 1 decision: to relocate your ice cream shop or remain where you are.
- Unknown(s) - In this scenario there is only 1 unknown possibility: high demand or low demand for ice cream. These are technically called chance node.
- When the decision and unknown are combined a total of 4 possible outcomes are created.

Determine Your Decision and Unknown

Four Possible Outcomes

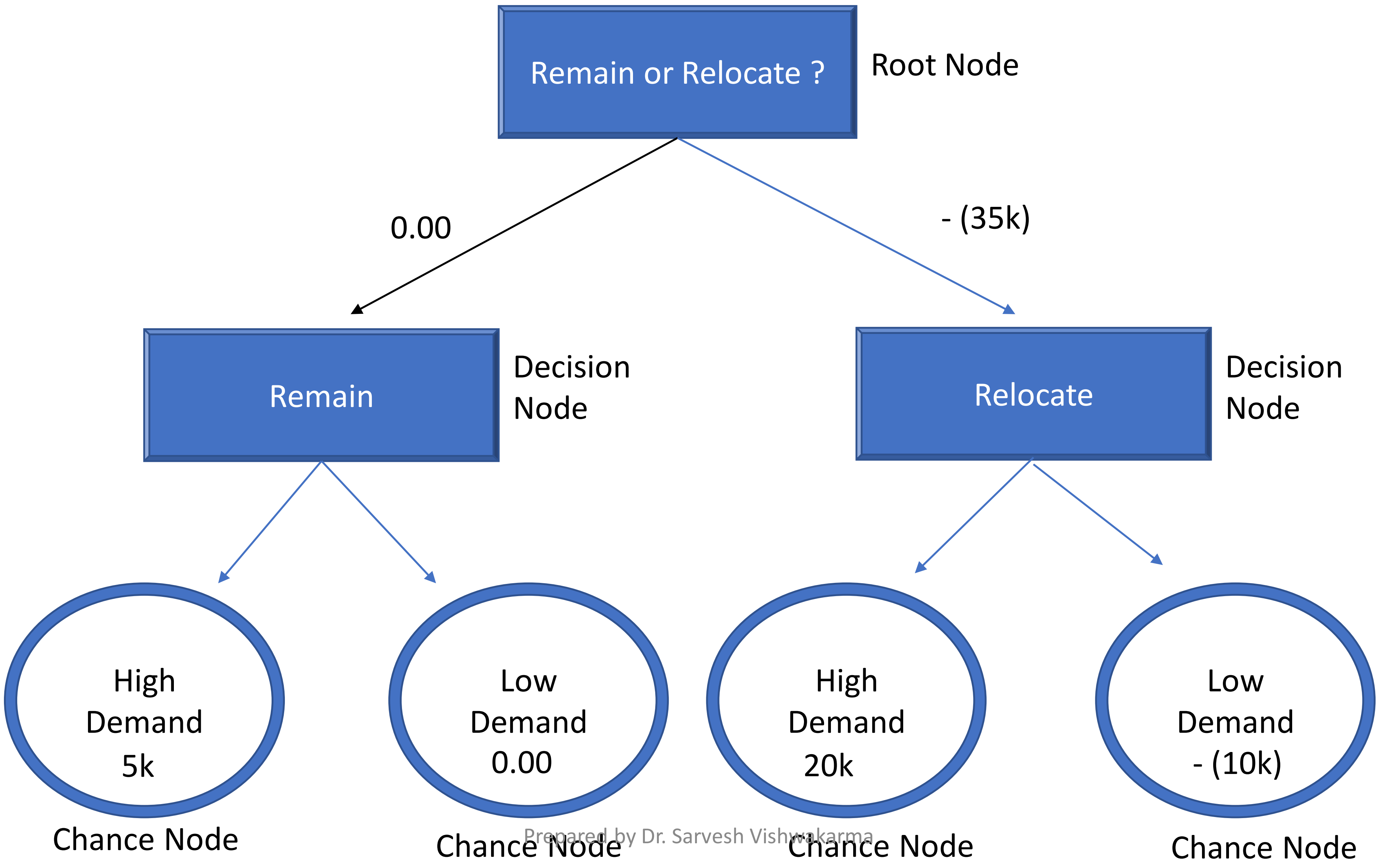
1. Relocate > High Demand
2. Relocate > Low Demand
3. Remain > High Demand
4. Remain > Low Demand




Remain or Relocate ?
Root Node
Remain
Decision Node
Relocate
Decision Node
High Demand
Low Demand
High Demand
Low Demand
Chance Node
Chance Node
Chance Node
Chance Node

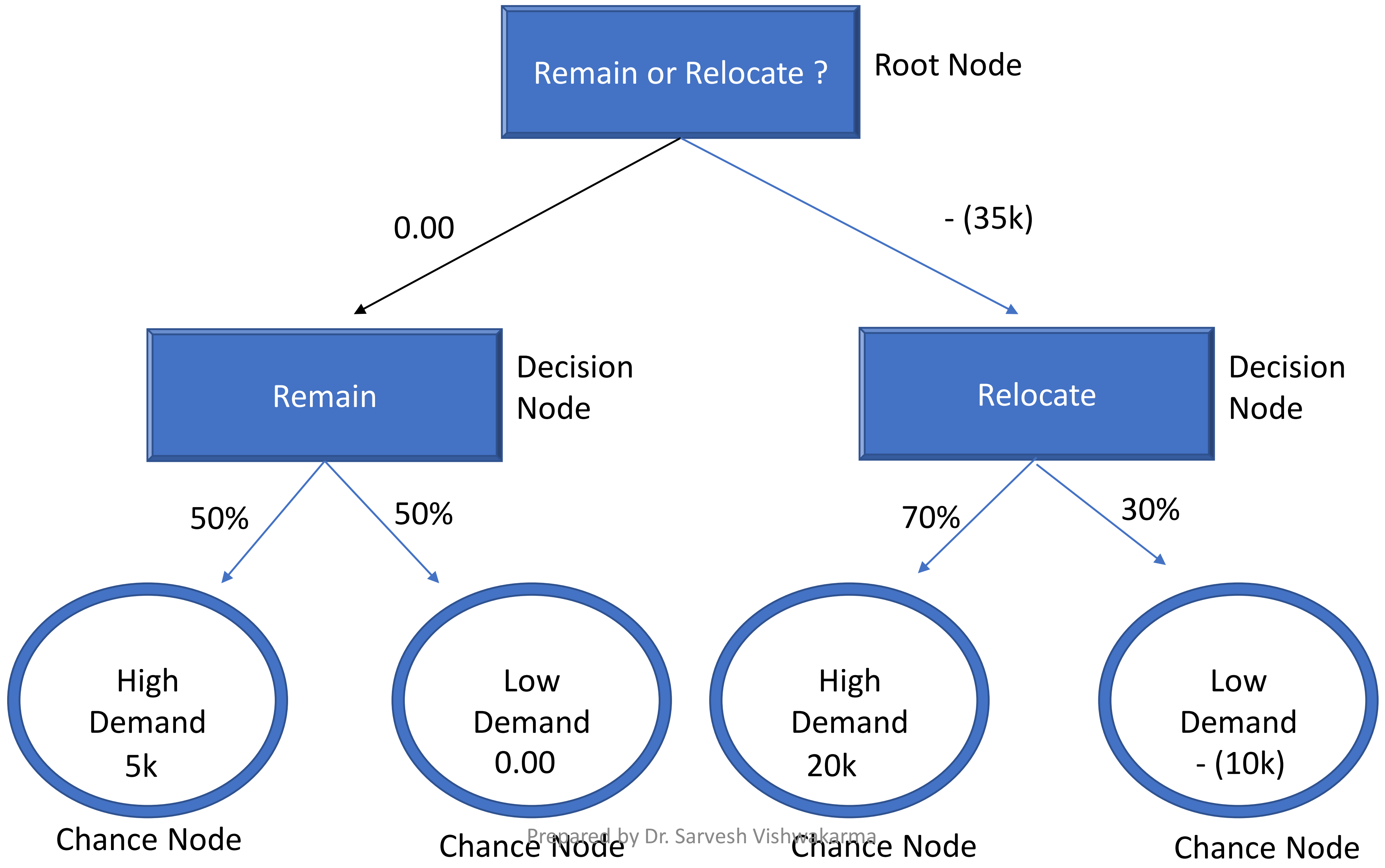
Determine the Values of Each Branch

1. Relocating your ice cream shop will roughly cost $35k
2. Remaining at your current location will cost $0.00
3. If you relocate and demand is high for ice cream, you estimate that you will see new profits up to $20k per year.
4. If you relocate and demand is low for ice cream, you estimate that it will cost you an extra $10k per year due to increased leasing costs.
5. If you remain and demand is high for ice cream, you estimate that you will see new profits up to $5k
6. If you remain and demand is low for ice cream, you estimate that you will break even and not see any extra costs or profits.


Remain or Relocate ?
Root Node
0.00
- (35k)
Remain
Decision Node
Relocate
Decision Node
High Demand 5k
Chance Node
Low Demand 0.00
Chance Node
High Demand 20k
Chance Node
Low Demand - (10k)
Chance Node

Determine the Probabilities of Each Branch

1. The probabilities of our initial decision do not need to be labelled, only the subsequent unknowns.
2. Based on your estimation, if you relocate: There is a 70% likelihood that demand will be high for ice cream.
3. Based on your estimation, if you remain: There is a 50% likelihood that demand will be high for ice cream.


Remain or Relocate ?
Root Node
0.00
- (35k)
Remain
Decision Node
Relocate
Decision Node
50%
50%
70%
30%
High Demand 5k
Low Demand 0.00
High Demand 20k
Low Demand - (10k)
Chance Node
Chance Node
Chance Node
Chance Node

Calculate the Weighted value of Each Decision

1. To calculate the weighted value of each decision, we first calculate the value of each unknown for each decision, and then add them together. We do this for both “Remain” and “Relocate”.
2. Remain: High Demand = \$5000×50% = \$2500 Low Demand = \$0 Weighted Value = \$2500 + \$0 = \$2500
3. Relocate: High Demand = \$20000×70% = \$14000 Low Demand = -\$10,000×30% = -\$3000
   Weighted Value = \$14000 + (-\$3000) = \$11000

Calculate the Net Benefit of Each Decision

1. If we choose to “Remain” the weighted value is +\$2500, and if we “Relocate”, the weighted value is +\$1100o. Final step is to calculate the net benefit of each decision.
2. Remain: Initial Cost = \$0 Weighted Value = \$2500 Net Benefit = \$2500 - \$0 = \$2500
3. Relocate: Initial Cost = \$35000 Weighted Value = \$11000 Net Benefit = \$11000 - \$35000 = -\$24000

Weaknesses of Decision Trees

1. Decision trees can change very quickly
2. Decision trees can become overly complex
3. Decision trees can cause “paralysis by analysis”

Strengths of Decision Trees

1. Decision trees force you to consider outcomes and options
2. Decision trees can help you visualize a problem
3. Decision trees can help you prioritize

Terminology

- Decision Node
  - A decision node is any node that involves making a choice between two options. Decision nodes are often represented by drawing a square or rectangular, and are sometimes called conditions or internal nodes.
- Chance Node
  - A chance node is any node that represents an uncertain result. Chance nodes are often represented by drawing a circle.
- End Node
  - An end node is a final answer and it does not split any further. End nodes are typically represented by a triangle, and are also called a terminal node or leaf node.

Additional Terminology

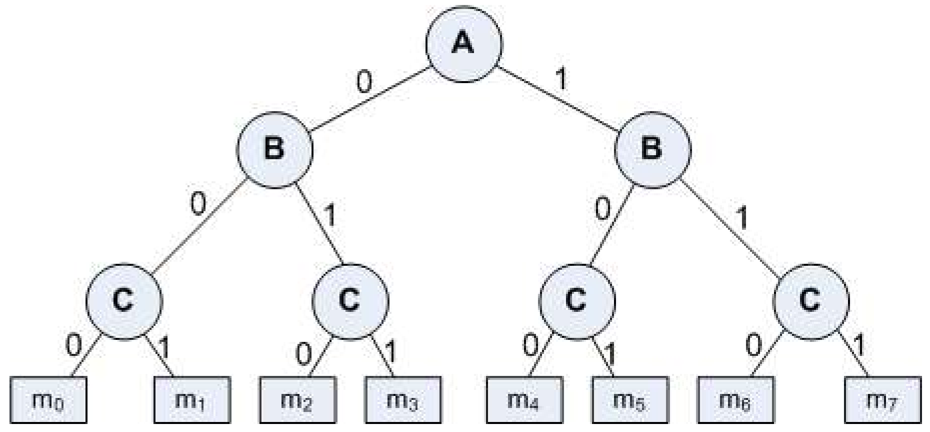
- Root Node
  - ➢ The root node is the original question/decision that starts the tree. For example, should you purchase a home or rent a home?
- Child Node/Subset
  - ➢ A child node/subset is the result of splitting a node into new nodes.
- Splitting
  - ➢ Splitting is the process of dividing a node into two or more subsets. When a node is split, it creates one or more subsets.
- Pruning
  - ➢ Pruning involves removing a subset node from a decision node. This is the opposite of splitting. It involves removing the branches that, for various reasons have low importance.
- Branch
  - ➢ A branch, also called an edge, is a subsection of an entire tree that connects nodes

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems

Example 9.1: Binary Decision Tree

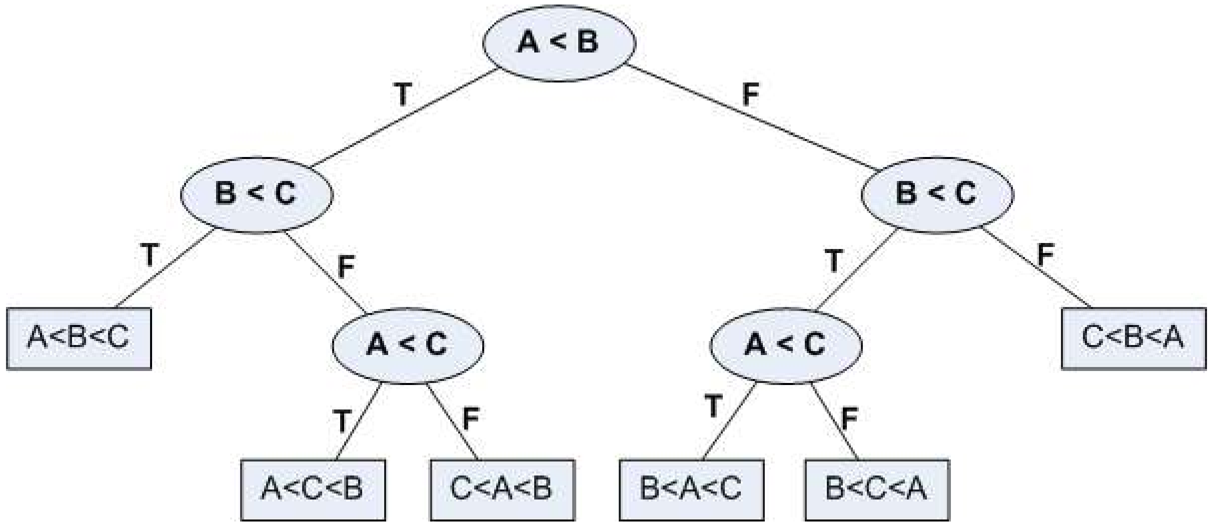
A	*B*	*C*	*f*
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



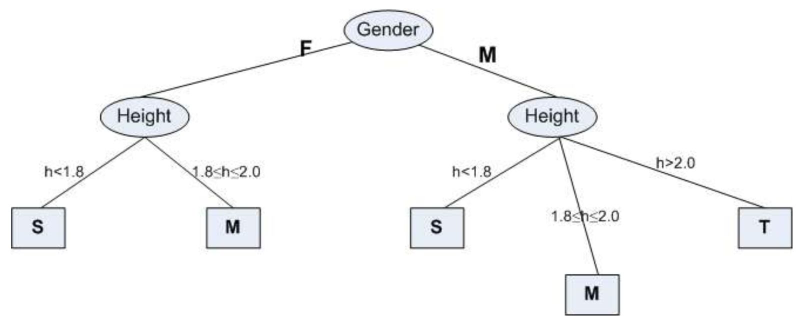
Basic Concept

- In Example 9.1, we have considered a decsion tree where values of any attribute if binary only. Decision tree is also possible where attributes are of continuous data type

Example 9.2: Decision Tree with numeric data



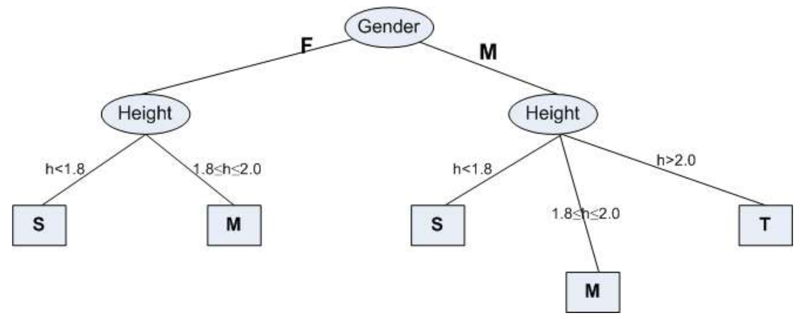
Some Characteristics



- Decision tree may be *n*-ary, $n \geq 2$.
- There is a special node called root node.
- All nodes drawn with circle (ellipse) are called internal nodes.
- All nodes drawn with rectangle boxes are called terminal nodes or leaf nodes.
- Edges of a node represent the outcome for a value of the node.
- In a path, a node with same label is never repeated.
- Decision tree is not unique, as different ordering of internal nodes can give different decision tree.

Decision Tree and Classification Task

- Decision tree helps us to classify data.
    - Internal nodes are some attribute
    - Edges are the values of attributes
    - External nodes are the outcome of classification
- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.



Decision Tree and Classification Task

Example 9.3 : Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	**Mammal**
Python	Cold	scales	no	no	no	no	yes	**Reptile**
Salmon	Cold	scales	no	yes	no	no	no	**Fish**
Whale	Warm	hair	yes	yes	no	no	no	**Mammal**
Frog	Cold	none	no	semi	no	yes	yes	**Amphibian**
Komodo	Cold	scales	no	no	no	yes	no	**Reptile**
Bat	Warm	hair	yes	no	yes	yes	yes	**Mammal**
Pigeon	Warm	feathers	no	no	yes	yes	no	**Bird**
Cat	Warm	fur	yes	no	no	yes	no	**Mammal**
Leopard	Cold	scales	yes	yes	no	no	no	**Fish**
Turtle	Cold	scales	no	semi	no	yes	no	**Reptile**
Penguin	Warm	feathers	no	semi	no	yes	no	**Bird**
Porcupine	Warm	quills	yes	no	no	yes	yes	**Mammal**
Eel	Cold	scales	no	yes	no	no	no	**Fish**
Salamander	Cold	none	no	semi	no	yes	yes	**Amphibian**

What are the class label of Dragon and Shark?

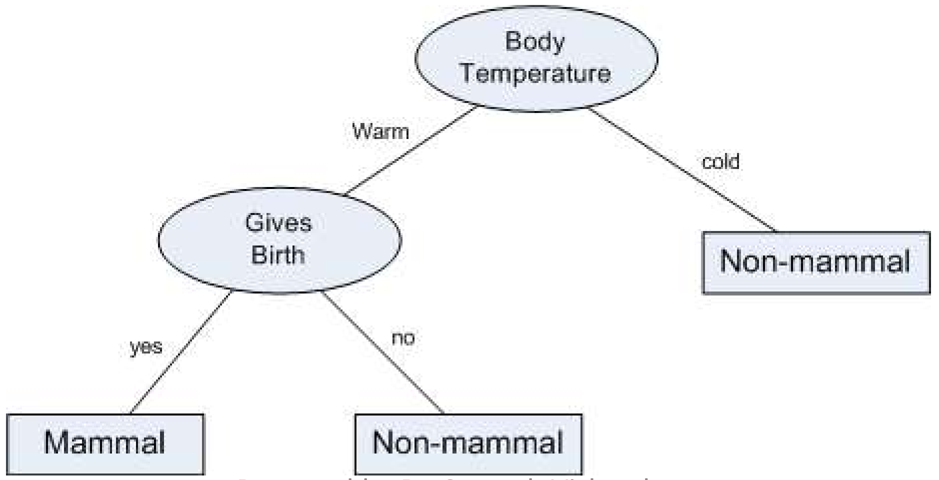
Decision Tree and Classification Task

Example 9.3 : Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	**?**

- Decision Tree that can be inducted based on the data (in Example 9.3) is as follows.



Decision Tree and Classification Task

- Example 9.3 illustrates how we can solve a classification problem by asking a series of question about the attributes.
    - Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- The series of questions and their answers can be organized in the form of a decision tree
    - As a hierarchical structure consisting of nodes and edges
- Once a decision tree is built, it is applied to any test to classify it.

Definition of Decision Tree

Definition 9.1: **Decision Tree**

Given a database $D = \{t_1, t_2, \ldots\ldots, t_n\}$, where t_i denotes a tuple, which is defined by a set of attribute $A = \{A_1, A_2, \ldots\ldots, A_m\}$. Also, given a set of classes $C = \{c_1, c_2, \ldots\ldots, c_k\}$.

A decision tree ***T*** is a tree associated with *D* that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
    - Some of the tree may not be optimum
    - Some of them may give inaccurate result
- Two approaches are known
    - **Greedy strategy**
        - A top-down recursive divide-and-conquer
    - **Modification of greedy strategy**
        - ID3
        - C4.5
        - CART, etc.

Built Decision Tree Algorithm

- **Algorithm BuiltDT**
- Input: *D* : Training data set
- Output: *T* : Decision tree

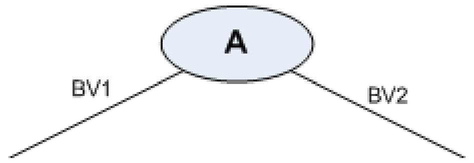
Steps

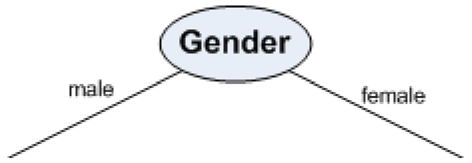
```
1. If all tuples in $D$ belongs to the same class $C_j$
        Add a leaf node labeled as $C_j$
        Return                          // Termination condition

2. Select an attribute $A_i$ (so that it is not selected twice in the same branch)

3. Partition $D = \{ D_1, D_2, \ldots, D_p\}$ based on $p$ different values of $A_i$ in $D$

4. For each $D_k \in D$
        Create a node and add an edge between $D$ and $D_k$ with label as the $A_i$'s attribute value in $D_k$

5. For each $D_k \in D$
        BuildTD($D_k$)                  // Recursive call

6. Stop
```

Node Splitting in **BuildDT** Algorithm

- BuildDT algorithm must provides a method for expressing an attribute test condition and corresponding outcome for different attribute type

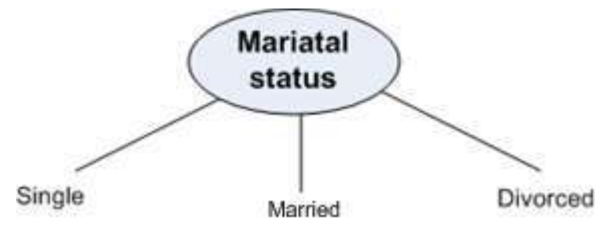
- **Case: Binary attribute**
  - This is the simplest case of node splitting
  - The test condition for a binary attribute generates only two outcomes



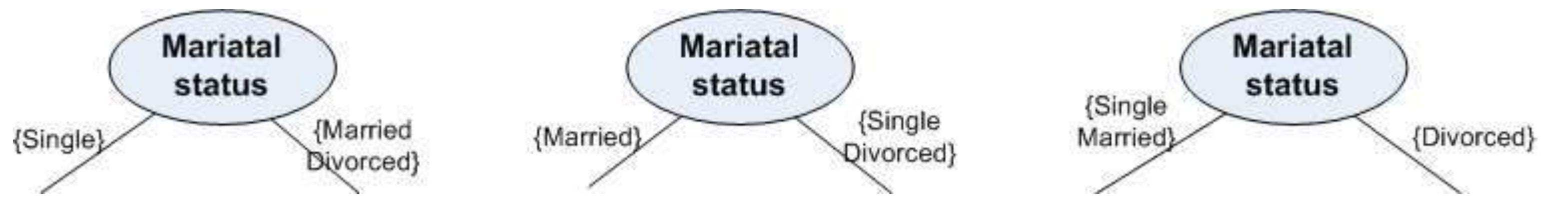


Node Splitting in **BuildDT** Algorithm

- **Case: Nominal attribute**
  - Since a nominal attribute can have many values, its test condition can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: Outcome depends on the number of distinct values for the corresponding attribute

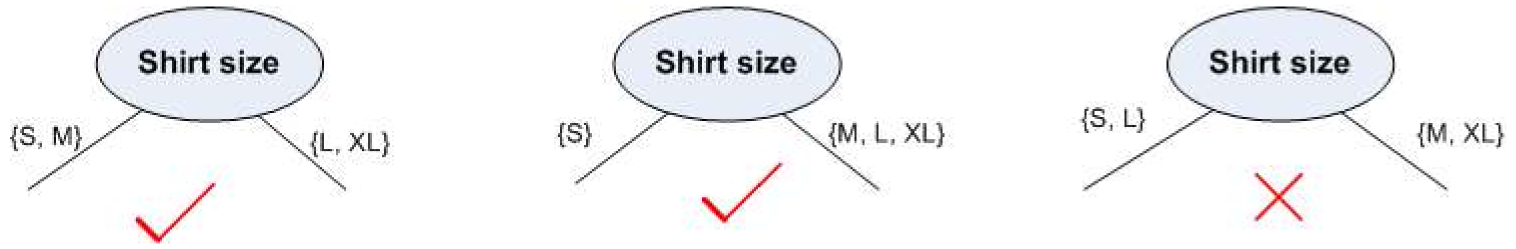


  - Binary splitting by grouping attribute values



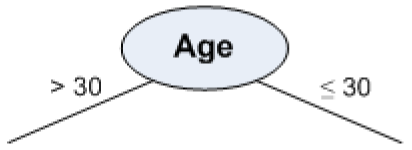
Node Splitting in **BuildDT** Algorithm

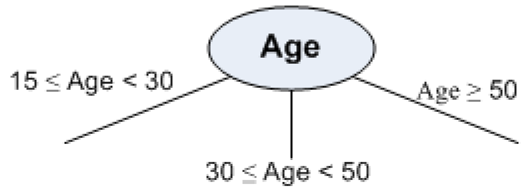
- **Case: Ordinal attribute**
  - It also can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: It is same as in the case of nominal attribute
  - Binary splitting attribute values should be grouped maintaining the order property of the attribute values



Node Splitting in **BuildDT** Algorithm

- **Case: Numerical attribute**
  - For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
    - Binary outcome: $A > v$ or $A \leq v$
      - In this case, decision tree induction must consider all possible split positions
    - Range query : $v_i \leq A < v_{i+1}$ for $i = 1, 2, \ldots, q$ (if q number of ranges are chosen)
      - Here, q should be decided a priori





- For a numeric attribute, decision tree induction is a combinatorial optimization problem

Illustration : **BuildDT** Algorithm

Example 9.4: Illustration of BuildDT Algorithm

- Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute
Height = {1.5, …, 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : **BuildDT** Algorithm

- To built a decision tree, we can select an attribute in two different orderings: <Gender, Height> or <Height, Gender>
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : <Gender, Height>**

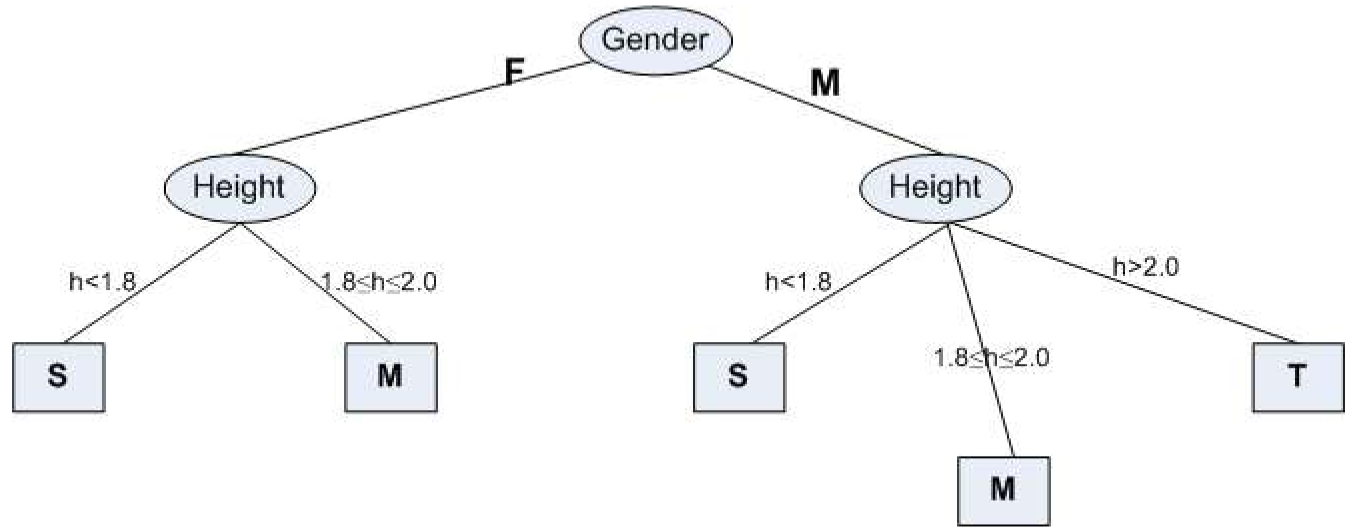
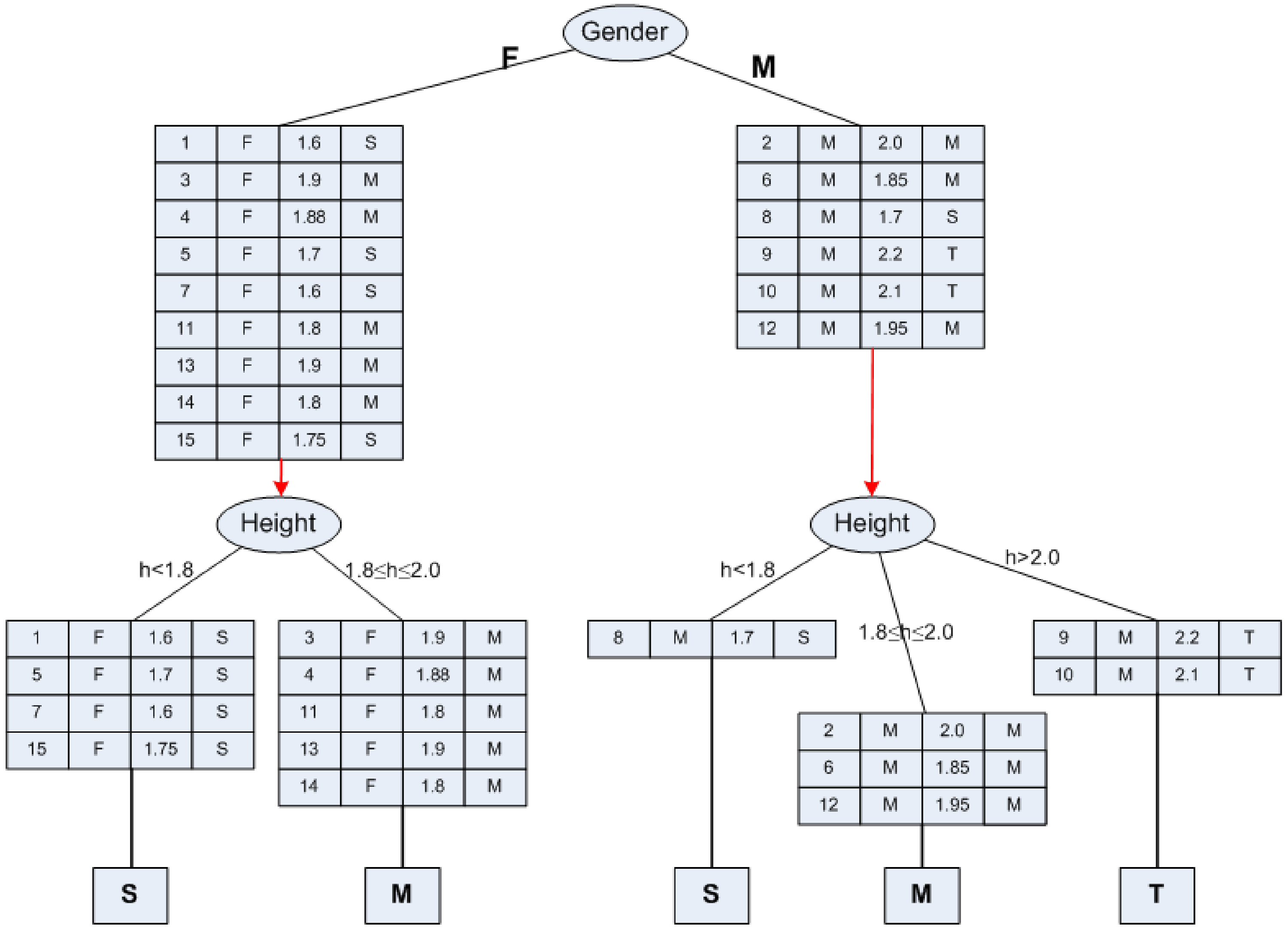


Illustration : **BuildDT** Algorithm



Gender = F:

1	F	1.6	S
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
7	F	1.6	S
11	F	1.8	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Gender = M:

2	M	2.0	M
6	M	1.85	M
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
12	M	1.95	M

F, h<1.8:

1	F	1.6	S
5	F	1.7	S
7	F	1.6	S
15	F	1.75	S

F, 1.8≤h≤2.0:

3	F	1.9	M
4	F	1.88	M
11	F	1.8	M
13	F	1.9	M
14	F	1.8	M

M, h<1.8:

8	M	1.7	S

M, 1.8≤h≤2.0:

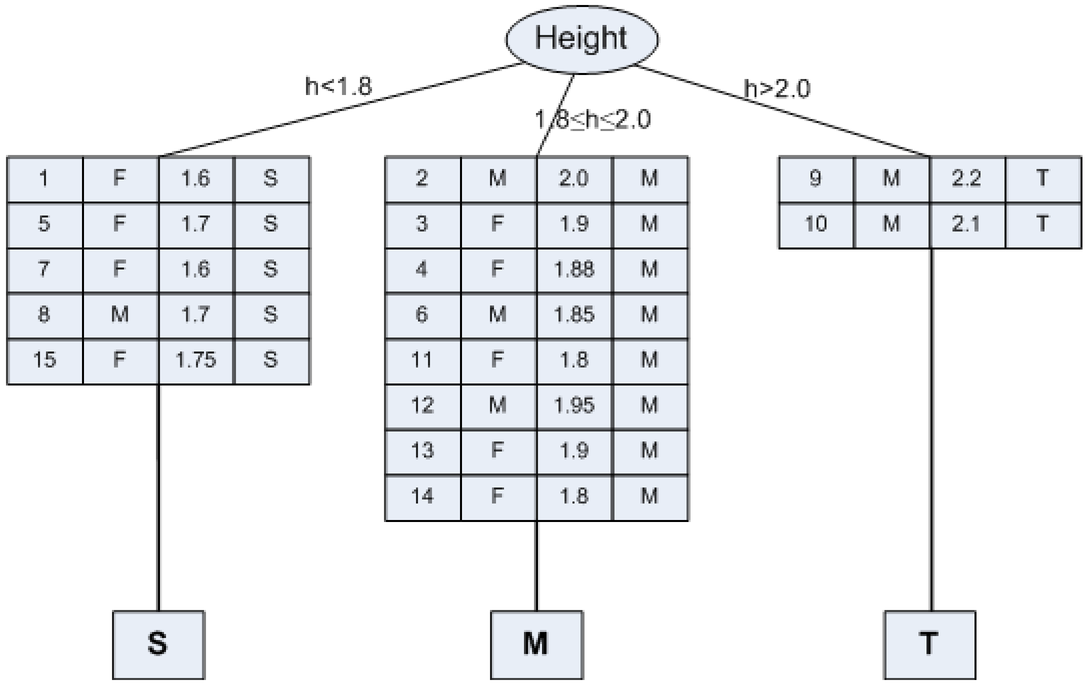
2	M	2.0	M
6	M	1.85	M
12	M	1.95	M

M, h>2.0:

9	M	2.2	T
10	M	2.1	T

Illustration : **BuildDT** Algorithm

- **Approach 2 : <Height, Gender>**



1	F	1.6	S
5	F	1.7	S
7	F	1.6	S
8	M	1.7	S
15	F	1.75	S

2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
6	M	1.85	M
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M

9	M	2.2	T
10	M	2.1	T

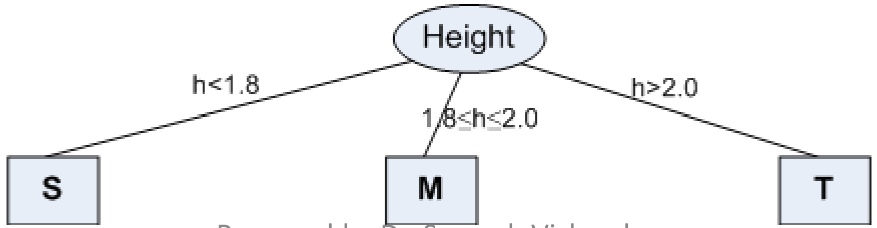


Illustration : **BuildDT** Algorithm

Example 9.5: Illustration of BuildDT Algorithm

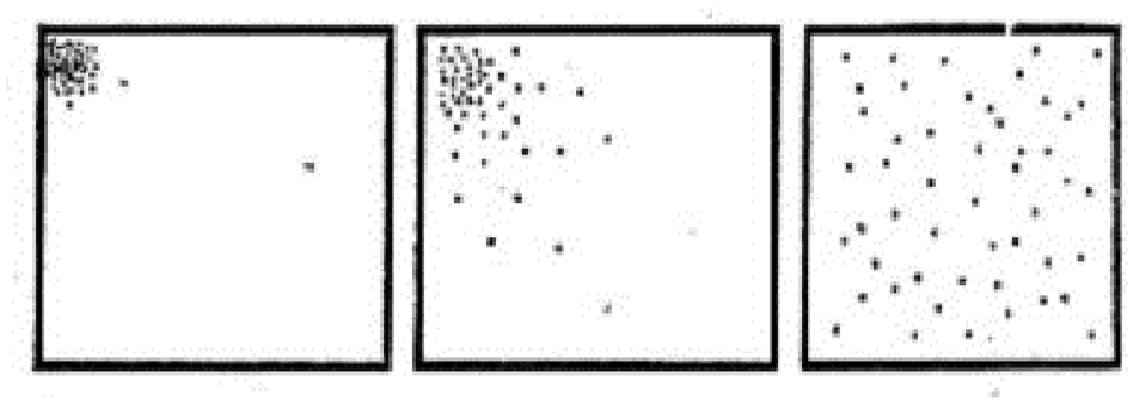
- Consider an anonymous database as shown.

A1	A2	A3	A4	Class
a11	a21	a31	a41	C1
a12	a21	a31	a42	C1
a11	a21	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a31	a42	C1
a11	a21	a32	a42	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a12	a22	a31	a42	C1

- Is there any “clue” that enables to select the “best” attribute first?
- Suppose, following are two attempts:
  - A1→A2→A3→A4 [naïve]
  - A3→A2→A4→A1 [Random]
- Draw the decision trees in the above-mentioned two cases.
- Are the trees different to classify any test data?
- If any other sample data is added into the database, is that likely to alter the decision tree already obtained?

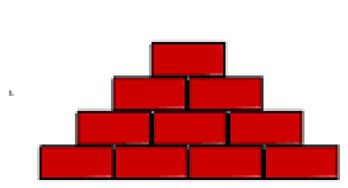
Concept of Entropy

Concept of Entropy

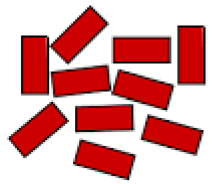


If a point represents a gas molecule, then which system has the more entropy?

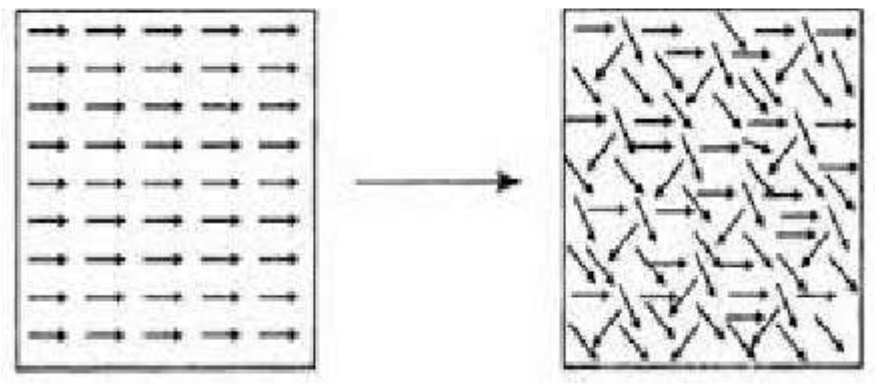
How to measure? $\Delta S = \frac{\Delta Q}{T}$?



More **ordered** less **entropy**



Less ordered **higher** entropy



More organized or **ordered** (less **probable**)

Less organized or disordered (**more** probable)

Concept of Entropy

Universe!
What was its entropy value at its starting point?

An Open Challenge!

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	89	99	56	91
10IM30013	95	98	55	93
12CE31005	98	96	58	97
12EC35015	93	95	54	99
12GG2005	90	91	53	98
12MI33006	91	93	57	97
13AG36001	96	94	58	95
13EE10009	92	96	56	96
13MA20012	88	98	59	96
14CS30017	94	90	60	94
14ME10067	90	92	58	95
14MT10038	99	89	55	93

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	19	59	16	71
10IM30013	37	38	25	83
12CE31005	38	16	48	97
12EC35015	23	95	54	19
12GG2005	40	71	43	28
12MI33006	61	93	47	97
13AG36001	26	64	48	75
13EE10009	92	46	56	56
13MA20012	88	58	59	66
14CS30017	74	20	60	44
14ME10067	50	42	38	35
14MT10038	29	69	25	33

Two sheets showing the tabulation of marks obtained in a course are shown.

Which tabulation of marks shows the “good” performance of the class?
How you can measure the same?

Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.
- At a later stage, with the growth of Information Technology, entropy becomes an important concept in Information Theory.
- To deal with the classification job, entropy is an important concept, which is considered as
  - an information-theoretic measure of the "uncertainty" contained in a training data
    - due to the presence of more than one classes.

Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).

- The first time it was used to measure the “information content” in messages.

.

- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

Measure of Information Content

- People, in general, are information hungry!
- Everybody wants to acquire information (from newspaper, library, nature, fellows, etc.)
  - Think how a crime detector do it to know about the crime from crime spot and criminal(s).
  - Kids annoyed their parents asking questions.
- In fact, fundamental thing is that we gather information asking questions (and decision tree induction is no exception).
- We may note that information gathering may be with certainty or uncertainty.

Measure of Information Content

Example 9.6

a) Guessing a birthday of your classmate

It is with uncertainty ~ $\frac{1}{365}$

Whereas guessing the day of his/her birthday is $\frac{1}{7}$.

This uncertainty, we may say varies between 0 to 1, both inclusive.

b) As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

- Does sun rises in the East? (answer is with 0 uncertainty)
- Will mother give birth to male baby? (answer is with ½ uncertainty)
- Is there a planet like earth in the galaxy? (answer is with an extreme uncertainty)

Definition of Entropy

Suppose there are m distinct objects, which we want to identify by asking a series of **Yes/No** questions. Further, we assume that m is an exact power of 2, say $m = 2^n$, where $n \geq 1$.

Definition 9.2: **Entropy**

The entropy of a set of m distinct values is the minimum number of yes/no questions needed to determine an unknown values from these m possibilities.

Entropy Calculation

- **How can we calculate the minimum number of questions, that is, entropy?**
  - There are two approaches:
    - Brute –force approach
    - Clever approach.

Example 9.7: City quiz

Suppose, Thee is a quiz relating to guess a city out of 8 cities, which are as follows:

Bangalore, Bhopal, Bhubaneshwar, Delhi, Hyderabad, Kolkata, Madras, Mumbai

The question is, "Which city is called **city of joy**"**?**

Approach 1: Brute-force search

- Brute force approach
  - We can ask “Is it city *X*?”,
  - if *yes* stop, else ask next …

In this approach, we can ask such questions randomly choosing one city at a time. As a matter of randomness, let us ask the questions, not necessarily in the order, as they are in the list.

Q.1:	Is the city Bangalore?	No
Q.2:	Is the city Bhubaneswar?	No
Q.3:	Is the city Bhopal?	No
Q.4:	Is the city Delhi?	No
Q.5:	Is the city Hyderabad?	No
Q.6:	Is the city Madras?	No
Q.7:	Is the city Mumbai?	No

No need to ask further question! Answer is already out by the Q.7. If asked randomly, each of these possibilities is equally likely with probability $\frac{1}{8}$. Hence on the average, we need

$$\frac{(1+2+3+4+5+6+7+7)}{8} = 4.375 \text{ questions.}$$

Approach 2: Clever approach

- Clever approach (binary search)
  - In this approach, we divide the list into two halves, pose a question for a half
  - Repeat the same recursively until we get *yes* answer for the unknown.

Q.1:	Is it Bangalore, Bhopal, Bhubaneswar or Delhi?	No
Q.2:	Is it Madras or Mumbai?	No
Q.3:	Is it Hyderabad?	No

So after fixing 3 questions, we are able to crack the answer.

Note:

Approach 2 is considered to be the best strategy because it will invariably find the answer and will do so with a minimum number of questions on the average than any other strategy.

Approach 1 occasionally do better (when you are lucky enough!)

- It is no coincidence that $8 = 2^3$, and the minimum number of yes/no questions needed is 3.
- If m = 16, then $16 = 2^4$, and we can argue that we need 4 questions to solve the problem. If m = 32, then 5 questions, m = 256, then 8 questions and so on.

Entropy Calculation

Lemma 9.1: **Entropy calculation**

The minimum number of *yes/no* questions needed to identify an unknown object from $m = 2^n$ equally likely possible object is n.

If m is not a power of 2, then the entropy of a set of m distinct objects that are equally likely is $\log_2 m$

Entropy in Messages

- We know that the most conventional way to code information is using binary bits, that is, using 0s and 1s.
- The answer to a question that can only be answered *yes/no* (with equal probability) can be considered as containing one **unit of information**, that is, one bit.
- In other words, the unit of information can also be looked at as the amount of information that can be **coded** using only 0s and 1s.

Entropy in Messages

Example 9.7: Information coding

- If we have **two** possible objects say **male** and **female,** then we use the coding
  0 = female
  1 = male
  $m = 2(= 2^n, n = 1)$

- We can encode **four** possible objects say **East**, **West**, **North**, **South** using two bits, for example
  00 : North
  01 : East
  10 : West
  11 : South
  $m = 4(= 2^n, n = 2)$

- We can encode **eight** values say eight different colours, we need to use **three** bits, such as
  000 : Violet
  001 : Indigo
  010 : Blue
  011 : Green
  100 : Yellow
  101 : Orange
  110 : Red
  111 : White
  $m = 8(= 2^n, n = 3)$

Thus, in general, to code *m* values, each in a distinct manner, we need *n* bits such that $m = 2^n$.

Entropy in Messages

- In this point, we can note that to identify an object, if it is encoded with bits, then we have to ask questions in an alternative way. For example
  - Is the first bit 0?
  - Is the second bit 0?
  - Is the third bit 0? and so on
- Thus, we need n questions, if m objects are there such that $m = 2^n$.
- The above leads to (an alternative) and equivalent definition of entropy

Definition 9.3: **Entropy**

The entropy of a set of m distinct values is the number of bits needed to encode all the values in the most efficient way.

Messages when $(m \neq 2^n)$

- In the previous discussion, we have assumed that m, the number of distinct objects is exactly a power of 2, that is $m = 2^n$ for some $n \geq 1$ and all m objects are equally likely.

- This is mere an assumption to make the discussion simplistic.

- In the following we try to redefine the entropy calculation in more general case, that is, when m $\neq 2^n$ and not necessarily m objects are equally probable. Let us consider a different instance of *yes/no* question game, which is as follows.

Example 9.8: Name game

- There are seven days: Sun, Mon, Tue, Wed, Thu, Fri, Sat.

- We are to identify a sequence of $k \geq 1$ such values (each one chosen independently of the others, that is, repetitions are allowed). Note that if k = 1, it is the type of game, we have already dealt with.

- We denote the minimum number of yes/no questions needed to identify a sequence of k unknown values drawn independently from m possibilities as E_k^m, the entropy in this case.

- In other words, E_k^m is the number of questions required to discriminate amongst m^k distinct possibilities.

Messages when $(m \neq 2^n)$

- Here, $m = 7$ (as stated in the game of sequence of days) and $k = 6$ (say).
- An arbitrary sequence may be {Tue, Thu, Tue, Mon, Sun, Tue}, etc. There are $7^6 = 117649$ possible sequences of six days.
- From our previous understanding, we can say that the minimum number of yes/no questions that is required to identify such a sequence is $\log_2 11769 = 16.8443$.
- Since, this is a non integer number, and the number of question should be an integer, we can say *17* questions are required. Thus,

$$E_6^7 = \lceil \log_2 7^6 \rceil$$

- In general,

$$E_k^m = \lceil \log_2 m^k \rceil$$

- Alternatively, the above can be written as,

$$\lceil \log_2 m^k \rceil \leq E_k^m \leq \lceil \log_2 m^k \rceil + 1$$

- Or

$$\lceil \log_2 m \rceil \leq \frac{E_k^m}{k} \leq \lceil \log_2 m \rceil + \frac{1}{k}$$

Entropy of Messages when $(m \neq 2^n)$

Note that here $\frac{E_k^m}{k}$ is the average number of questions needed to determine each of the values in a sequence of k values. By choosing a large enough value of k, that is, a long enough sequence, the value of $\frac{1}{k}$ can be made as small as we wish. Thus, the average number of questions required to determine each value can be made arbitrarily close to $\log_2 m$. This is evident from our earlier workout, for example, tabulated below, for $m = 7$.

$$E_k^m = \lceil \log_2 m^k \rceil$$

k	m^k	$\log_2 m^k\rceil$	No. Q	$\frac{No.Q}{k}$
6	117649	16.84413	17	2.8333
21		58.95445	59	2.8095
1000		2807.3549	2808	2.8080
…..	…..	…..	…..	…..

No. Q = Number of questions

Note that $\log_2 7 \approx 2.8074$ and $\frac{No.Q}{k} \approx \log_2 7$. Further, $\frac{No.Q}{k} = \frac{E_k^7}{k}$ *i.e.* $\frac{E_k^7}{k} = \log_2 7$ (is independent of k and is a constant!)

Entropy of Messages when $(m \neq 2^n)$

Lemma 9.4: **Entropy Calculation**

The entropy of a set of m distinct objects is $\log_2 m$ even when m is not exactly a power of 2.

- We have arrived at a conclusion that $E = \log_2 m$ for any value of m, irrespective of weather it is a power of 2 or not.

Note: E is not necessarily be an integer always.

- Next, we are to have our observation, if all m objects are not equally probable.
- Suppose, p_i denotes the frequency with which the i^{th} of the m objects occurs, where $0 \leq p_i \leq 1$ for all p_i such that

$$\sum_{i=1}^{m} p_i = 1$$

Discriminating amongst m values ($m \neq 2^n$)

Example 9.8: Discriminating among objects

- Suppose four objects $A, B, C\ and\ D$ which occur with frequencies $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ and $\frac{1}{8}$, respectively.
- Thus, in this example, $m = 4$ and $p_1 = \frac{1}{2}, p_2 = \frac{1}{4}, p_3 = \frac{1}{8}$ and $p_4 = \frac{1}{8}$.
- Using standard 2-bit encoding, we can represent them as $A = 00, B = 01, C = 10, D = 11.$
- Also, we can follow variable length coding (also called Huffman coding) as an improved way of representing them.
- The Huffman coding of $A, B, C\ and\ D$ with their frequencies $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ and $\frac{1}{8}$ are shown below.

A = 1
B = 01
C = 001
D = 000

Discriminating amongst m values ($m \neq 2^n$)

- With the above representation say, if A is to be identified, then we need to examine only one question, for B it is 2 and for C and D both, it is 3.
- Thus, on the average, we need

$$\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75\ bits$$

- This is the number of yes/no questions to identify any one of the four objects, whose frequency of occurrences are not uniform.
- This is simply in contrast to 2-bit encoding, where we need 2-bits (questions) on the average.

Discriminating amongst m values ($m \neq 2^n$)

- It may be interesting to note that even with variable length encoding, there are several ways of encoding. Few of them are given below.

1)	2)	3)
$A = 0$	$A = 01$	$A = 101$
$B = 11$	$B = 1$	$B = 001$
$C = 100$	$C = 001$	$C = 10011$
$D = 101$	$D = 000$	$D = 100001$

- The calculation of entropy in the observed cases can be obtained as:

  1) 1.75 2) 2 3) 3.875

- Anyway, key to finding the most efficient way of encoding is to **assign a smallest number of bits to the object with highest frequency** and so on.
- The above observation is also significant in the sense that it provides a **systematic way of finding a sequence of well-chosen question in order to identify an object at a faster rate.**

Information Content

Based on the previous discussion we can easily prove the following lemma.

Lemma 9.3: **Information content**

If an object occurs with frequency p, then the most efficient way to represent it with $\log_2(^1/_p)$ bits.

Example 9.9: Information content

- A which occurs with frequency $\frac{1}{2}$ is represented by 1-bit, B which occurs with frequency $\frac{1}{4}$ represented by 2-bits and both C and D which occurs with frequency $\frac{1}{8}$ are represented by 3 bits each.

Entropy Calculation

We can generalize the above understanding as follows.

- If there are *m* objects with frequencies $p_1, p_2 \ldots\ldots, p_m$, then the average number of bits (i.e. questions) that need to be examined a value, that is, entropy is the frequency of occurrence of the i^{th} value multiplied by the number of bits that need to be determined, summed up values of *i* from *1* to *m*.

Theorem 9.4: Entropy calculation

If p_i denotes the frequencies of occurrences of *m* distinct objects, then the entropy *E* is

$$E = \sum_{i=1}^{m} p_i \log({}^{1}/{p_i}) \;\; and \sum_{i=1}^{m} p_i = 1$$

Note:

- If all are equally likely, then $p_i = \frac{1}{m}$ and $E = \log_2 m$; it is the special case.

Entropy of a Training Set

- If there are k classes $c_1, c_2\ldots\ldots, c_k$ and p_i for $i = 1\ to\ k$ denotes the number of occurrences of classes c_i divided by the total number of instances (i.e., the frequency of occurrence of c_i) in the training set, then entropy of the training set is denoted by

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Here, *E* is measured in “bits” of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which $p_i \neq 0$
- E is always a positive quantity
- E takes it’s minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only **one** non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among k possible classes. In this case, the maximum value of *E* is $log_2 k$.

Entropy of a Training Set

Example 9.10: OPTH dataset

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

A coded forms for all values of attributes are used to avoid the cluttering in the table.

Entropy of a training set

Specification of the attributes are as follows.

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Class: 1: Contact Lens 2:Normal glass 3: Nothing

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy *E* of the database is:

$$E = -\frac{4}{24}\log_2\frac{4}{24} - \frac{5}{24}\log_2\frac{5}{24} - \frac{15}{24}\log_2\frac{15}{24} = 1.3261$$

Note:

- The entropy of a training set implies the number of yes/no questions, on the average, needed to determine an unknown test to be classified.
- It is very crucial to decide the series of questions about the value of a set of attribute, which collectively determine the classification. Sometimes it may take one question, sometimes many more.
- Decision tree induction helps us to ask such a series of questions. In other words, we can utilize entropy concept to build a better decision tree.

How entropy can be used to build a decision tree is our next topic of discussion.

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.

- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.

- Different algorithms have been proposed to take a good control over

  1. Choosing the best attribute to be splitted, and

  2. Splitting criteria

- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into three important of them
  - **ID3**
  - **C 4.5**
  - **CART**

Algorithm ID3

ID3: Decision Tree Induction Algorithms

- Quinlan [1986] introduced the ID3, a popular short form of **I**terative **D**ichotomizer 3 for decision trees from a set of training data.

- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.

- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

Algorithm ID3

- In ID3, entropy is used to measure how informative a node is.
  - It is observed that splitting on any attribute has the property that average entropy of the resulting training subsets will be less than or equal to that of the previous training set.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
  - The attribute with the largest value of information gain is chosen as the splitting attribute and
  - it partitions into a number of smaller training sets based on the distinct values of attribute under split.

Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α.
- $D \equiv$ denotes the training set at any instant
- $|D| \equiv$ denotes the size of the training set D
- $E(D) \equiv$ denotes the entropy of the training set D
- The entropy of the training set D

$$E(D) = -\sum_{i=1}^{k} p_i \, log_2(p_i$$

  - where the training set D has $c_1, c_2, \ldots, c_k$, the k number of distinct classes and
  - p_i, $0 < p_i \leq 1$ is the probability that an arbitrary tuple in D belongs to class c_i ($i = 1, 2, \ldots, k$).

Defining Information Gain

- p_i can be calculated as

$$p_i = \frac{|C_{i,D}|}{|D|}$$

- where $C_{i,D}$ is the set of tuples of class c_i in *D*.

- Suppose, we want to partition *D* on some attribute *A* having *m* distinct values $\{a_1, a_2, \ldots, a_m\}$.

- Attribute *A* can be considered to split *D* into *m* partitions $\{D_1, D_2, \ldots, D_m\}$, where D_j $(j = 1,2, \ldots ,m)$ contains those tuples in *D* that have outcome a_j of *A*.

Defining Information Gain

Definition 9.4: **Weighted Entropy**

The weighted entropy denoted as $E_A(D)$ for all partitions of D with respect to A is given by:

$$E_A(D) = \sum_{j=1}^{m} \frac{|D_j|}{|D|} E(D_j)$$

Here, the term $\frac{|D_j|}{|D|}$ denotes the weight of the j-th training set.

More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from D based on the splitting of A.

Defining Information Gain

- Our objective is to take A on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.

- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.

- In that sense, $E_A(D)$ is a measure of impurities (or purity). A lesser value of $E_A(D)$ implying more power the partitions are.

Definition 9.5: **Information Gain**

Information gain, $\alpha(A, D)$ of the training set D splitting on the attribute A is given by

$$\alpha(A, D) = E(D) - E_A(D)$$

In other words, $\alpha(A, D)$ gives us an estimation how much would be gained by splitting on A. The attribute A with the highest value of α should be chosen as the splitting attribute for D.

Information Gain Calculation

Example 9.11 : Information gain on splitting OPTH

- Let us refer to the OPTH database discussed in Slide #48.
- Splitting on **Age** at the root level, it would give three subsets D_1, D_2 and D_3 as shown in the tables in the following three slides.
- The entropy $E(D_1), E(D_2)$ and $E(D_3)$ of training sets D_1, D_2 and D_3 and corresponding weighted entropy $E_{Age}(D_1)$, $E_{Age}(D_2)$ and $E_{Age}(D_3)$ are also shown alongside.
- The Information gain $\alpha\ (Age, OPTH)$ is then can be calculated as **0.0394**.
- Recall that entropy of OPTH data set, we have calculated as *E(OPTH)* = **1.3261**

  (see Slide #49)

Information Gain Calculation

Example 9.11 : Information gain on splitting OPTH

Training set: D_1(Age = 1)

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

$$E(D_1) = -\frac{2}{8} log_2(\frac{2}{8}) - \frac{2}{8} log_2(\frac{2}{8}) - \frac{4}{8} log_2(\frac{4}{8}) = \mathbf{1.5}$$

$$E_{Age}(D_1) = \frac{8}{24} \times 1.5 = \mathbf{0.5000}$$

Calculating Information Gain

Training set: D_2(Age = 2)

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

$$E(D_2) = -\frac{1}{8}log_2(\frac{1}{8}) - \frac{2}{8}log_2(\frac{2}{8}) - \frac{5}{8}log_2(\frac{5}{8})$$
$$= \mathbf{1.2988}$$

$$E_{Age}(D_2) = \frac{8}{24} \times 1.2988 = \mathbf{0.4329}$$

Calculating Information Gain

Training set: D_3(Age = 3)

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

$$E(D_3) = -\frac{1}{8}log_2(\frac{1}{8}) - \frac{1}{8}log_2(\frac{1}{8}) - \frac{6}{8}log_2(\frac{6}{8}) = \mathbf{1.0613}$$

$$E_{Age}(D_3) = \frac{8}{24} \times 1.0613 = \mathbf{0.3504}$$

$\alpha\ (Age, D)$ = 1.3261 – (0.5000 + 0.4329 + 0.3504) = **0.0394**

Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on Eye-sight, Astigmatic and Use Type. The results are summarized below.
- Splitting attribute: Age

$$\alpha(Age, OPTH) = 0.0394$$

- Splitting attribute: Eye-sight

$$\alpha(Eye - sight, OPTH) = 0.0395$$

- Splitting attribute: Astigmatic

$$\alpha(\text{Astigmatic}\,, OPTH) = 0.3770$$

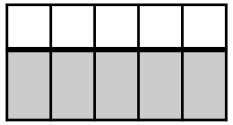
- Splitting attribute: Use Type

$$\alpha(\text{Use Type}\,, OPTH) = 0.5488$$

Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
  - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is $\alpha(\text{Use Type}\,, OPTH) = 0.5488$
  - Hence, the attribute should be chosen for splitting is "Use Type".
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

Decision Tree Induction : ID3 Way



$\boldsymbol{E(OPTH) = 1.3261}$

$OPTH$

Age ✗ $\alpha = 0.0394$

Eye-sight ✗ $\alpha = 0.395$

Use Type ✓ $\alpha = 0.0394$

Astigmatic ✗ $\alpha = 0.3770$

$D1$ $Frequent(1)$

$E(D_1) = ?$

Age	Eye	Ast	Use	Class
1	1	1	1	3
1	1	2	1	3
1	2	1	1	3
1	2	2	1	3
2	1	1	1	3
2	2	1	1	3
2	2	2	1	3
3	1	1	1	3
3	1	2	1	3
3	2	1	1	3
3	2	2	1	3

? Age $\alpha = ?$

? Eye-sight $\alpha = ?$

? Astigmatic $\alpha = ?$

$D2$ $Less(2)$

$E(D_2) = ?$

Age	Eye	Ast	Use	Class
1	1	1	2	2
1	1	2	2	1
1	2	1	2	2
1	2	2	2	1
2	1	1	2	2
2	1	2	2	1
2	2	1	2	2
3	1	1	2	3
3	1	2	2	3
3	2	1	2	2
3	2	2	2	3

? Age $\alpha = ?$

? Eye-sight $\alpha = ?$

? Astigmatic $\alpha = ?$

Frequency Table : Calculating α

- Calculation of entropy for each table and hence information gain for a particular split appears tedious (at least manually)!
- As an alternative, we discuss **a short-cut method** of doing the same using a special data structure called **Frequency Table.**
- **Frequency Table**: Suppose, $X = \{x_1, x_2, \ldots, x_n\}$ denotes an attribute with $n - different$ attribute values in it. For a given database D, there are a set of $k\ classes$ say $C = \{c_1, c_2, \ldots c_k\}$. Given this, a frequency table will look like as follows.

Frequency Table : Calculating α

		X					
		x_1	x_2	…………	x_i	…………	x_n
Class	c_1			…………		…………	
	c_2			…………		…………	
	$\vdots$	$\vdots$	$\vdots$	…………	$\vdots$	…………	$\vdots$
	c_j			…………	f_{ij}	…………	
	$\vdots$	$\vdots$	$\vdots$	…………	$\vdots$	…………	$\vdots$
	c_k			…………		…………	

- Number of rows = Number of classes
- Number of columns = Number of attribute values
- f_{ij} = Frequency of x_i for class c_j

Assume that $|D| = N$, the number of total instances of D.

Calculation of α using Frequency Table

Example 9.12 : OTPH Dataset

With reference to OPTH dataset, and for the attribute Age, the frequency table would look like

	Age=1	**Age=2**	**Age=3**	**Row Sum**
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column Sum	8	8	8	24

N=24

Column Sums

Calculation of α using Frequency Table

- The weighted average entropy $E_X(D)$ then can be calculated from the frequency table following the
  - Calculate $V = f_{ij} \log_2 f_{ij}$ for all $i = 1,2,\dots,k$
    (Entry Sum) $j = 1,2,\dots,n$ and $v_{ij} \neq 0$
  - Calculate $S = s_i \log_2 s_i$ for all $i = 1,2,\dots,n$
    (Column Sum) in the row of column sum
  - Calculate $E_X(D) = (-V + S)/N$

Example 9.13: OTPH Dataset
For the frequency table in **Example 9.12**, we have

$$V = 2\log 2 + 1\log 1 + 1\log 1 + 2\log 2 + 2\log 2 + 1\log 1 + 4\log 4 + 5\log 5 + 6\log 6$$

$$S = 8\log 8 + 8\log 8 + 8\log 8$$

$$\boldsymbol{E_{Age}(OPTH) = 1.2867}$$

Proof of Equivalence

- In the following, we prove the equivalence of the short-cut of entropy calculation using Frequency Table.
- Splitting on an attribute A with n values produces n subsets of the training dataset D (of size $|D| = N$). The $j-th$ subset ($j = 1,2,\ldots,n$) contains all the instances for which the attribute takes its $j-th$ value. Let N_j denotes the number of instances in the $j-th$ subset. Then

$$\sum_{j=1}^{n} N_j = N$$

- Let f_{ij} denotes the number of instances for which the classification is c_i and attribute A takes its $j-th$ value. Then

$$\sum_{i=1}^{k} f_{ij} = N_j$$

Proof of Equivalence

Denoting E_j as the entropy of the $j-th$ subset, we have

$$E_j = -\sum_{i=1}^{k} \frac{f_{ij}}{N_j} \log_2 \frac{f_{ij}}{N_j}$$

Therefore, the weighted average entropy of the splitting attribute A is given by

$$E_A(D) = \sum_{j=1}^{n} \frac{N_j}{N} . E_j$$

$$= -\sum_{j=1}^{n} \sum_{i=1}^{k} \frac{N_j}{N} . \frac{f_{ij}}{N_j} . \log_2 \frac{f_{ij}}{N_j}$$

$$= -\sum_{j=1}^{n} \sum_{i=1}^{k} \frac{f_{ij}}{N_j} . \log_2 \frac{f_{ij}}{N_j}$$

Proof of Equivalence

$$= -\sum_{j=1}^{n}\sum_{i=1}^{k}\frac{f_{ij}}{N_j}.\log_2 f_{ij} + \sum_{j=1}^{n}\sum_{i=1}^{k}\frac{f_{ij}}{N_j}.\log_2 N_j$$

$$= -\sum_{j=1}^{n}\sum_{i=1}^{k}\frac{f_{ij}}{N_j}.\log_2 f_{ij} + \sum_{j=1}^{1}\frac{N_j}{N}.\log_2 N_j$$

$$\because \sum_{i=1}^{k} f_{ij} = N_j$$

$$= \left(-\sum_{j=1}^{n}\sum_{i=1}^{k} f_{ij}.\log_2 f_{ij} + \sum_{j=1}^{n} N_j \log_2 N_j\right)\Big/N$$

$$= (-V + S)/N$$

$$\text{where } V = \sum_{j=1}^{n}\sum_{i=1}^{k} f_{ij}.\log_2 f_{ij} \qquad \text{(Entries sum)}$$

$$and\ S = \sum_{j=1}^{n} N_j \log_2 N_j \qquad (Column\ Sum)$$

Hence, the equivalence is proved.

Limiting Values of Information Gain

- The Information gain metric used in ID3 always should be positive or zero.
- It is always positive value because information is always gained (i.e., purity is improved) by splitting on an attribute.
- On the other hand, when a training set is such that if there are k classes, and the entropy of training set takes the largest value i.e., $\log_2 k$ (this occurs when the classes are balanced), then the information gain will be zero.

Limiting Values of Information Gain

Example 9.14: Limiting values of Information gain

Consider a training set shown below.

Data set *Table A*

X	Y	Class
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

X *Table X*

	1	2	3	4
A	1	1	1	1
B	1	1	1	1
$C.Sum$	2	2	2	2

Frequency table of X

Y *Table Y*

	1	2
A	2	2
B	2	2
$C.Sum$	4	4

Frequency table of Y

Limiting values of Information Gain

- Entropy of Table A is

$$E = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = \log 2 = 1$$ (The maximum entropy).

- In this example, whichever attribute is chosen for splitting, each of the branches will also be balanced thus each with maximum entropy.
- In other words, information gain in both cases (i.e., splitting on X as well as Y) will be zero.

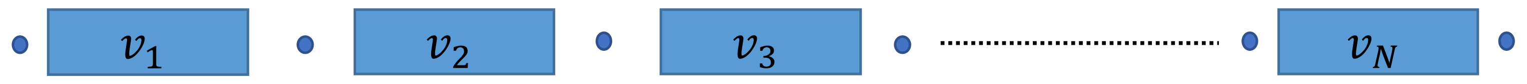
Note:

- The absence of information gain does not imply that there is no profit for splitting on the attribute.
- Even if it is chosen for splitting, ultimately it will lead to a final decision tree with the branches terminated by a leaf node and thus having an entropy of zero.
- Information gain can never be a negative value.

Splitting of Continuous Attribute Values

- In the foregoing discussion, we assumed that an attribute to be splitted is with a finite number of discrete values. Now, there is a great deal if the attribute is not so, rather it is a continuous-valued attribute.
- There are two approaches mainly to deal with such a case.

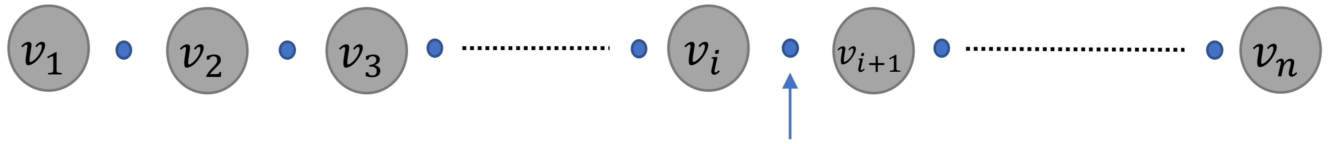
1. **Data Discretization**: All values of the attribute can be discretized into a finite number of group values and then split point can be decided at each boundary point of the groups.



So, if there are $n - groups$ of discrete values, then we have $(n + 1)$ split points.

Splitting of Continuous attribute values

2. **Mid-point splitting:** Another approach to avoid the data discretization.
    - It sorts the values of the attribute and take the distinct values only in it.
    - Then, the mid-point between each pair of adjacent values is considered as a split-point.



- Here, if n-distinct values are there for the attribute A, then we choose $n-1$ split points as shown above.
- For example, there is a split point $s = \frac{v_i + v_{(i+1)}}{2}$ in between v_i and $v_{(i+1)}$
- For each split-point, we have two partitions: $A \leq s \; and \; A > s$, and finally the point with maximum information gain is the desired split point for that attribute.

Algorithm CART

CART Algorithm

- It is observed that information gain measure used in ID3 is biased towards test with many outcomes, that is, it prefers to select attributes having a large number of values.

- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
    - CART stands for **Classification and Regression Tree**
    - In fact, invented independently at the same time as ID3 (1984).
    - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.

- CART is a technique that generates a **binary decision tree;** That is, unlike ID3, in CART, for each node only two children is created.

- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index** . It is also known as **Gini Index of Diversity** and is denote as γ.

Gini Index of Diversity

Definition 9.6: **Gini Index**

Suppose, D is a training set with size $|D|$ and $C = \{c_1, c_2, \ldots, c_k\}$ be the set of k classifications and $A = \{a_1, a_2, \ldots, a_m\}$ be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D. It can be defined as follows.

$$G(D) = 1 - \sum_{i=1}^{k} p_i^2$$

where p_i is the probability that a tuple in D belongs to class c_i and p_i can be estimated as

$$p_i = \frac{|C_{i,D}|}{D}$$

where $|C_{i,D}|$ denotes the number of tuples in D with class c_i.

Gini Index of Diversity

Note

- $G(D)$ measures the "impurity" of data set D.
- The smallest value of $G(D)$ is zero
  - which it takes when all the classifications are same.
- It takes its largest value $= 1 - \frac{1}{k}$
  - when the classes are evenly distributed between the tuples, that is the frequency of each class is $\frac{1}{k}$.

Gini Index of Diversity

Definition 9.7: **Gini Index of Diversity**

Suppose, a binary partition on A splits D into D_1 and D_2, then the weighted average Gini Index of splitting denoted by $G_A(D)$ is given by

$$G_A(D) = \frac{|D_1|}{D}.G(D_1) + \frac{|D_2|}{D}.G(D_2)$$

This binary partition of D reduces the impurity and the reduction in impurity is measured by

$$\gamma(A, D) = G(D) - G_A(D)$$

Gini Index of Diversity and CART

- This $\gamma(A, D)$ is called the Gini Index of diversity.
- It is also called as “impurity reduction”.
- The attribute that maximizes the reduction in impurity (or equivalently, has the minimum value of $G_A(D)$) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.

- We shall discuss how the same is possible for attribute with more than two values.

- **Case 1: Discrete valued attributes**

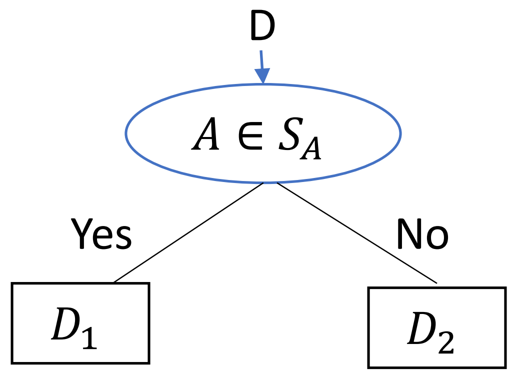
- Let us consider the case where A is a discrete-valued attribute having m discrete values $a_1, a_2, \dots, a_m$.

- To determine the best binary split on A, we examine all of the possible subsets say 2^A of A that can be formed using the values of A.

- Each subset $S_A \in 2^A$ can be considered as a binary test for attribute A of the form "$A \in S_A$?“.

n-ary Attribute Values to Binary Splitting

- Thus, given a data set *D*, we have to perform a test for an attribute value *A* like



- This test is satisfied if the value of *A* for the tuples is among the values listed in S_A.

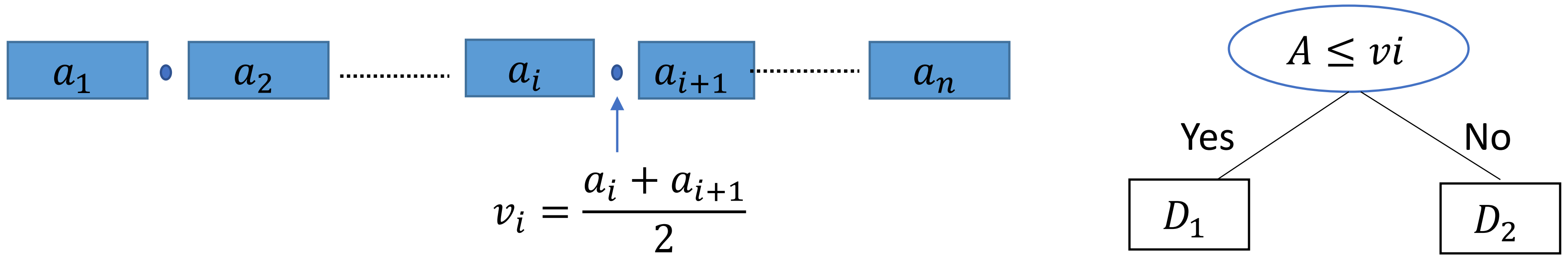
- If *A* has *m* distinct values in *D*, then there are 2^m possible subsets, out of which the empty subset $\{\}$ and the power set $\{a_1, a_2, \dots, a_n\}$ should be excluded (as they really do not represent a split).

- Thus, there are $2^m - 2$ possible ways to form two partitions of the dataset *D*, based on the binary split of *A*.

n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

- For a continuous-valued attribute, each possible split point must be taken into account.
- The strategy is similar to that followed in ID3 to calculate information gain for the continuous –valued attributes.
- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i, then



n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say v_i.
- D_1 is the set of tuples in D satisfying $A \leq v_i$ and D_2 in the set of tuples in D satisfying $A > v_i$.
- The point giving the minimum Gini Index $G_A(D)$ is taken as the split-point of the attribute A.

Note

- The attribute A and either its splitting subset S_A (for discrete-valued splitting attribute) or split-point v_i (for continuous valued splitting attribute) together form the splitting criteria.

CART Algorithm : Illustration

Example 9.15 : CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

Age
Y : young
M : middle-aged
O : old

Salary
L : low
M : medium
H : high

Job
G : government
P : private

Performance
A : Average
E : Excellent

Class : Select
Y : yes
N : no

Tuple#	Age	Salary	Job	Performance	Select
1	Y	H	P	A	N
2	Y	H	P	E	N
3	M	H	P	A	Y
4	O	M	P	A	Y
5	O	L	G	A	Y
6	O	L	G	E	N
7	M	L	G	E	Y
8	Y	M	P	A	N
9	Y	L	G	A	Y
10	O	M	G	A	Y
11	Y	M	G	E	Y
12	M	M	P	E	Y
13	M	H	G	A	Y
14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

$$G(EMP) = 1 - \sum_{i=1}^{2} p_i^2$$

$$= 1 - \left[\left(\frac{9}{14}\right)^2 + \left(\frac{5}{14}\right)^2\right]$$

$$= \mathbf{0.4592}$$

Now let us consider the calculation of $G_A(EMP)$ for **Age**, **Salary**, **Job** and **Performance**.

CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

$$\underset{age_1}{\{Y\}},\quad \underset{age_2}{\{M\}},\quad \underset{age_3}{\{O\}},\quad \underset{age_4}{\{Y,M\}},\quad \underset{age_5}{\{Y,O\}},\quad \underset{age_6}{\{M,O\}}$$

$$G_{age_1}(D) = \frac{5}{14} * \left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \frac{9}{14}\left(1 - \left(\frac{6}{14}\right)^2 - \left(\frac{8}{14}\right)^2\right) = \mathbf{0.4862}$$

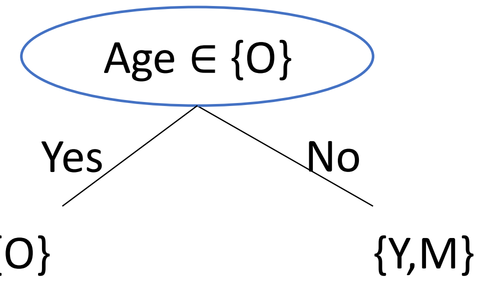
$G_{age_2}(D) = \mathbf{?}$

$G_{age_3}(D) = \mathbf{?}$

$G_{age_4}(D) = G_{age_3}(D)$

$G_{age_5}(D) = G_{age_2}(D)$

$G_{age_6}(D) = G_{age_1}(D)$



The best value of Gini Index while splitting attribute Age is $\gamma(Age_3, D) = \mathbf{0.3750}$

CART Algorithm : Illustration

Attribute of Splitting: Salary

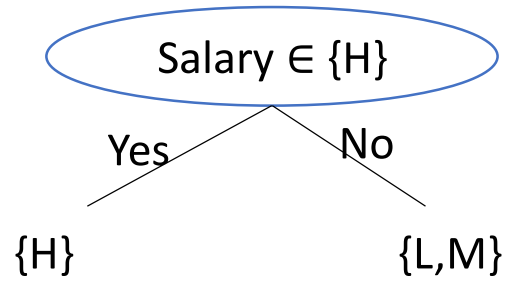
The attribute salary has three values namely L, M and H. So, there are 6 subsets, that should be considered for splitting as:

$$\underset{sal_1}{\{L\}}, \quad \underset{sal_2}{\{M,H\}}, \quad \underset{sal_3}{\{M\}}, \quad \underset{sal_4}{\{L,H\}}, \quad \underset{sal_5}{\{H\}}, \quad \underset{sal_6}{\{L,M\}}$$

$$G_{sal_1}(D) = G_{sal_2}(D) = 0.3000$$

$$G_{sal_3}(D) = G_{sal_4}(D) = 0.3150$$

$$G_{sal_5}(D) = G_{sal_6}(D) = 0.4508$$



$$\gamma(salary_{(5,6)}, D) = 0.4592 - 0.4508 = 0.0084$$

CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

$$G_{job}(D) = \frac{7}{14}G(D_1) + \frac{7}{14}G(D_2)$$

$$= \frac{7}{14}\left[1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2\right] + \frac{7}{14}\left[1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2\right] = \textbf{?}$$

$$\gamma(job, D) = \textbf{?}$$

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

$G_{Performance}(D) =$ **?**

$\gamma(performance, D) =$ **?**

Out of these $\gamma(salary, D)$ gives the maximum value and hence, the attribute **Salary** would be chosen for splitting subset $\{M, H\}$ or $\{L\}$.

Note:

It can be noted that the procedure following "information gain" calculation (i.e. $\propto (A, D)$) and that of "impurity reduction" calculation (i.e. $\gamma(A, D)$) are near about.

Calculating γ using Frequency Table

- We have learnt that splitting on an attribute gives a reduction in the average Gini Index of the resulting subsets (as it does for entropy).

- Thus, in the same way the average weighted Gini Index can be calculated using the same frequency table used to calculate information gain $\alpha(A, D)$, which is as follows.

The G(D_j) for the j^{th} subset D_j

$$G(D_j) = 1 - \sum_{i=1}^{k} \left(\frac{f_{ij}}{|D_j|}\right)^2$$

Calculating γ using Frequency Table

The average weighted Gini Index, $G_A(D_j)$ (assuming that attribute has *m* distinct values is)

$$G_A(D_j) = \sum_{j=1}^{k} \frac{|D_j|}{|D_1|}.G(D_j)$$

$$= \sum_{j=1}^{m} \frac{|D_j|}{|D|} - \sum_{j=1}^{m} \sum_{i=1}^{k} \frac{|D_j|}{|D|}.\left(\frac{f_{ij}}{|D_j|}\right)^2$$

$$= 1 - \frac{1}{|D|} \sum_{j=1}^{m} \frac{1}{D_j}.\sum_{i=1}^{k} {f_{ij}}^2$$

The above gives a formula for *m*-attribute values; however, it an be fine tuned to subset of attributes also.

Illustration: Calculating γ using Frequency Table

Example 9.16 : Calculating γ using frequency table of OPTH

Let us consider the frequency table for OPTH database considered earlier. Also consider the attribute A_1 with three values 1, 2 and 3. The frequency table is shown below.

	1	**2**	**3**
Class 1	2	1	1
Class 2	2	2	1
Class 3	4	5	6
Column sum	8	8	8

Illustration: Calculating γ using Frequency Table

Now we can calculate the value of Gini Index with the following steps:

1. For each non-empty column, form the sum of the squares of the values in the body of the table and divide by the column sum.

2. Add the values obtained for all columns and divided by $|D|$, the size of the database.

3. Subtract the total from 1.

As an example, with reference to the frequency table as mentioned just.

$$A_1 = 1 = \frac{(2^2+2^2+4^2)}{24} = 3.0 \qquad A_1 = 2 = \frac{(1^2+2^2+5^2)}{24} = 3.75$$

$$A_1 = 3 = \frac{(1^2+1^2+6^2)}{24} = 4.75 \qquad \text{So, } G_{A1}(D) = 1 - \frac{1+3.75+4.75}{24} = 0.5208$$

Illustration: Calculating γ using Frequency Table

Thus, the reduction in the value of Gini Index on splitting attribute A_1 is

$\gamma(A_1, D) = 0.5382 - 0.5208 = 0.0174$

where $G(D) = 0.5382$

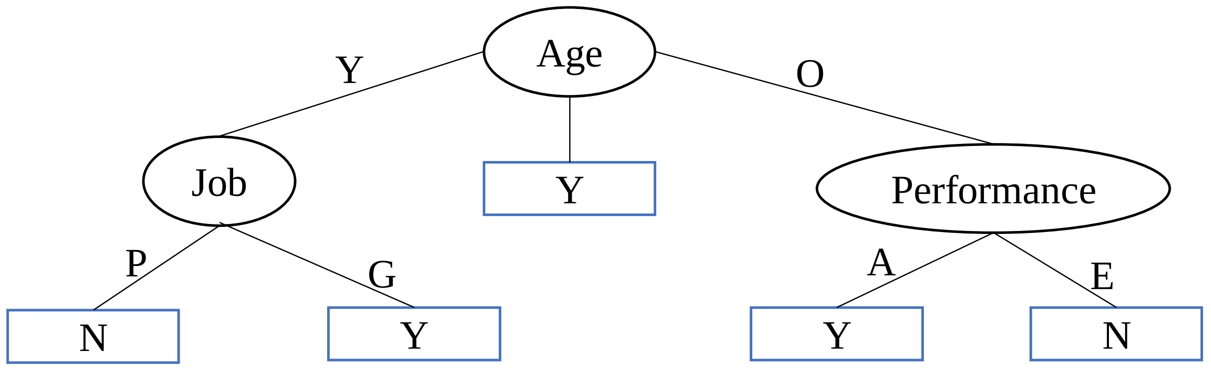
The calculation can be extended to other attributes in the OTPH database and is left as an exercise.

?

Decision Trees with ID3 and CART Algorithms

Example 9.17 : Comparing Decision Trees of EMP Data set

Compare two decision trees obtained using ID3 and CART for the EMP dataset. The decision tree according to ID3 is given for your ready reference (subject to the verification)



Decision Tree using ID3

?

Decision Tree using CART

Algorithm C4.5

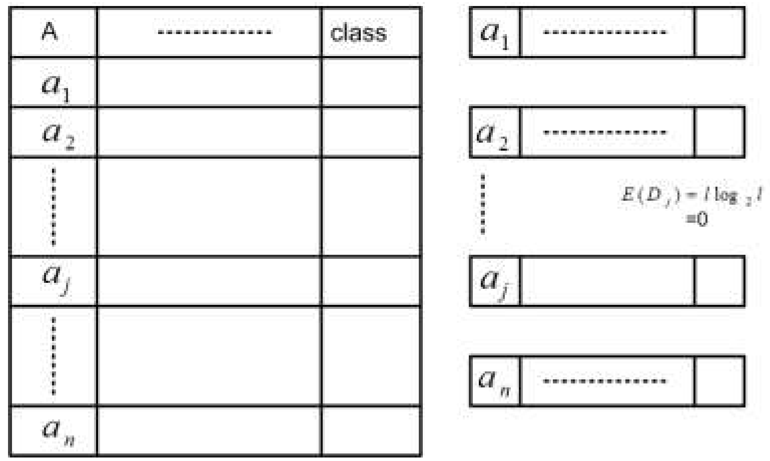
Algorithm C 4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotometer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact biased towards splitting attribute having a large number of outcomes.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
    - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is $E_A(D) = 0$

Algorithm C4.5 : Introduction

Example 9.18 : Limitation of ID3

In the following, each tuple belongs to a unique class. The splitting on A is shown.



$$E_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} . E(D_j) = \sum_{j=1}^{n} \frac{1}{|D|} . 0 = 0$$

Thus, $\alpha(A, D) = E(D) - E_A(D)$ is maximum in such a situation.

Algorithm: C 4.5 : Introduction

- Although, the previous situation is an extreme case, intuitively, we can infer that ID3 favours splitting attributes having a large number of values
  - compared to other attributes, which have a less variations in their values.

- Such a partition appears to be useless for classification.

- This type of problem is called **overfitting problem**.

Note:

Decision Tree Induction Algorithm ID3 may suffer from overfitting problem.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as β.
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C 4.5 : Gain Ratio

Definition 9.8: **Gain Ratio**

The gain ratio can be defined as follows. We first define split information $E_A^*(D)$ as

$$E_A^*(D) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} . \log \frac{|D_j|}{|D|}$$

Here, *m* is the number of distinct values in *A*.

The gain ratio is then defined as $\beta(A, D) = \frac{\alpha(A,D)}{E_A^*(D)}$, where $\alpha(A, D)$ denotes the information gain on splitting the attribute *A* in the dataset *D*.

Physical Interpretation of E_A^*(D)

Split information E_A^*(D)

- The value of split information depends on
    - the number of (distinct) values an attribute has and
    - how uniformly those values are distributed.

- In other words, it represents the potential information generated by splitting a data set *D* into *m* partitions, corresponding to the *m* outcomes of on attribute *A*.

- Note that for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in *D*.

Physical Interpretation of E_A^*(D)

Example 9.18 : Split information E_A^*(D)

- To illustrate E_A^*(D), let us examine the case where there are 32 instances and splitting an attribute A which has a_1, a_2, a_3 and a_4 sets of distinct values.

  - Distribution 1 : Highly non-uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	32	0	0	0

$$E_A^*(\text{D}) = -\frac{32}{32} log_2(\frac{32}{32}) = -log_2 1 = 0$$

  - Distribution 2

	a_1	a_2	a_3	a_4
Frequency	16	16	0	0

$$E_A^*(\text{D}) = -\frac{16}{32} log_2(\frac{16}{32}) - \frac{16}{32} log_2(\frac{16}{32}) = log_2 2 = 1$$

Physical Interpretation of $E_A^*(\mathrm{D})$

- Distribution 3

	a_1	a_2	a_3	a_4
Frequency	16	8	8	0

$$E_A^*(\mathrm{D}) = -\frac{16}{32}\,log_2(\frac{16}{32}) - \frac{8}{32}\,log_2(\frac{8}{32}) - \frac{8}{32}\,log_2(\frac{8}{32}) = 1.5$$

- Distribution 4

	a_1	a_2	a_3	a_4
Frequency	16	8	4	4

$$E_A^*(\mathrm{D}) = 1.75$$

- Distribution 5: Uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	8	8	8	8

$$E_A^*(\mathrm{D}) = (-\frac{8}{32}\,log_2(\frac{8}{32}))*4 = -log_2(\frac{1}{4}) = 2.0$$

Physical Interpretation of E_A^*(D)

- In general, if there are m attribute values, each occurring equally frequently, then the split information is $log_2 m$.

- Based on the Example 9.18, we can summarize our observation on split information as under:

  - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.

  - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.

  - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $log_2 m$ (both inclusive)

Physical Interpretation of $\beta(A, B)$

- Information gain signifies how much information will be gained on partitioning the values of attribute A
    - Higher information gain means splitting of A is more desirable.

- On the other hand, split information forms the denominator in the gain ratio formula.
    - This implies that higher the value of split information is, lower the gain ratio.
    - In turns, it decreases the information gain.

- Further, information gain is large when there are many distinct attribute values.
    - When many distinct values, split information is also a large value.
    - This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.

- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.

Calculation of β using Frequency Table

- The frequency table can be used to calculate the gain ratio for a given data set and an attribute.

.

- We have already learned the calculation of information gain using Frequency Table.
- To calculate gain ratio, in addition to information gain, we are also to calculate split information.
- This split information can be calculated from frequency table as follows.
- For each non-zero column sum say s_j contribute $|D_j|$ for the *j*-th column (i.e., the *j*-th value of the attribute). Thus the split information is

$$E_A^*(D) = -\sum_{j=1}^{m} \frac{s_j}{|D|} log_2 \frac{s_j}{|D|}$$

If there are *m*-columns in the frequency table.

Practice:

Using Gain ratio as the measurement of splitting attributes, draw the decision trees for OPTH and EMP data sets. Give calculation of gain ratio at each node.

Summary of Decision Tree Induction Algorithms

- We have learned the building of a decision tree given a training data.
  - The decision tree is then used to classify a test data.
- For a given training data *D*, the important task is to build the decision tree so that:
  - All test data can be classified accurately
  - The tree is balanced and with as minimum depth as possible, thus the classification can be done at a faster rate.
- In order to build a decision tree, several algorithms have been proposed. These algorithms differ from the chosen splitting criteria, so that they satisfy the above mentioned objectives as well as the decision tree can be induced with minimum time complexity. We have studied three decision tree induction algorithms namely ID3, CART and C4.5. A summary of these three algorithms is presented in the following table.

Table 11.6

Algorithm	Splitting Criteria	Remark
ID3	**Information Gain** $\alpha(A, D) = E(D) - E_A(\mathrm{D})$ Where $E(D)$ = Entropy of D (a measure of uncertainty) = $-\sum_{i=1}^{k} p_i \log_2 p_i$ where *D* is with set of *k* classes $c_1, c_2, \ldots, c_k$ and $p_i = \frac{\lvert C_{i,D}\rvert}{\lvert D\rvert}$; Here, $C_{i,D}$ is the set of tuples with class c_i in *D*. $E_A(\mathrm{D})$ = Weighted average entropy when *D* is partitioned on the values of attribute A = $\sum_{j=1}^{m} \frac{\lvert D_j\rvert}{\lvert D\rvert} E(D_j)$ Here, *m* denotes the distinct values of attribute *A*.	• The algorithm calculates $\alpha(A_i,D)$ for all A_i *in* D and choose that attribute which has **maximum** $\alpha(A_i,D)$. • The algorithm can handle both categorical and numerical attributes. • It favors splitting those attributes, which has a large number of distinct values.

Algorithm	Splitting Criteria	Remark
CART	**Gini Index** $\gamma(A, D) = G(D) - G_A(\mathrm{D})$ where $G(D)$ = Gini index (a measure of impurity) $= 1 - \sum_{i=1}^{k} {p_i}^2$ Here, $p_i = \frac{\lvert C_{i,D} \rvert}{\lvert D \rvert}$ and D is with k number of classes and $G_A(D) = \frac{\lvert D_1 \rvert}{\lvert D \rvert} G(D_1) + \frac{\lvert D_2 \rvert}{\lvert D \rvert} G(D_2)$, when D is partitioned into two data sets D_1 and D_2 based on some values of attribute A.	• The algorithm calculates all binary partitions for all possible values of attribute A and choose that binary partition which has the **maximum** $\gamma(A, D)$. • The algorithm is computationally very expensive when the attribute A has a large number of values.

Algorithm	**Splitting Criteria**	**Remark**
C4.5	**Gain Ratio** $$\beta(A,D) = \frac{\alpha(A,D)}{E_A^*(\text{D})}$$ where $\alpha(A,D)$ = Information gain of D (same as in ID3, and $E_A^*(\text{D})$ = splitting information $= -\sum_{j=1}^{m} \frac{\|D_j\|}{\|D\|} log_2 \frac{\|D_j\|}{\|D\|}$ when D is partitioned into D_1, D_2, … , D_m partitions corresponding to m distinct attribute values of A.	• The attribute A with **maximum** value of $\beta(A,D)$ is selected for splitting. • Splitting information is a kind of normalization, so that it can check the biasness of information gain towards the choosing attributes with a large number of distinct values.

In addition to this, we also highlight few important characteristics of decision tree induction algorithms in the following.

Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.

2. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.

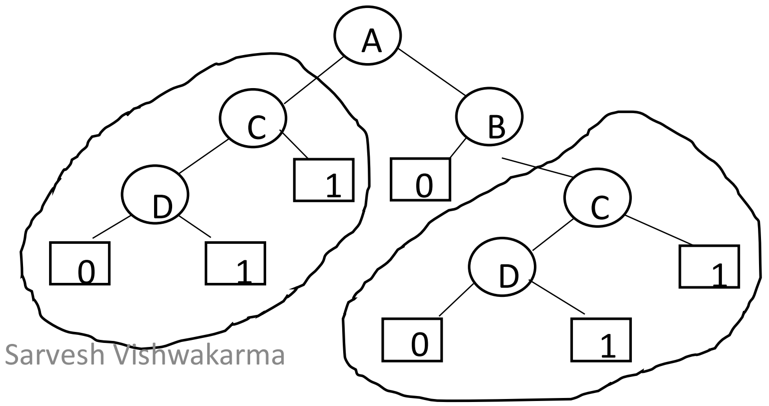
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

4. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of $O(d)$, where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

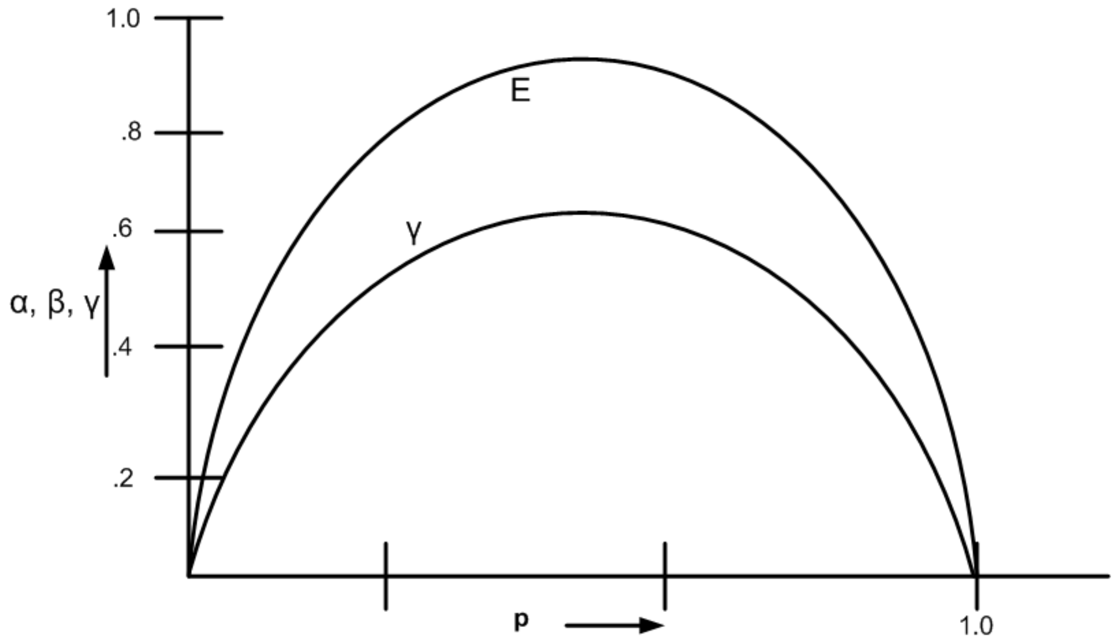
5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Notes on Decision Tree Induction algorithms

7. **Decision tree equivalence:** The different splitting criteria followed in different decision tree induction algorithms have little effect on the performance of the algorithms. This is because the different heuristic measures (such as information gain (α), Gini index (γ) and Gain ratio (β) are quite consistent with each other); also see the figure below.


1.0
.8
.6
.4
.2
α, β, γ
E
γ
p
1.0