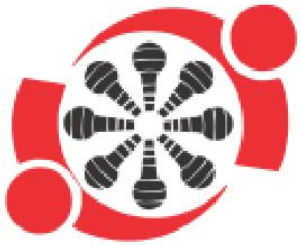
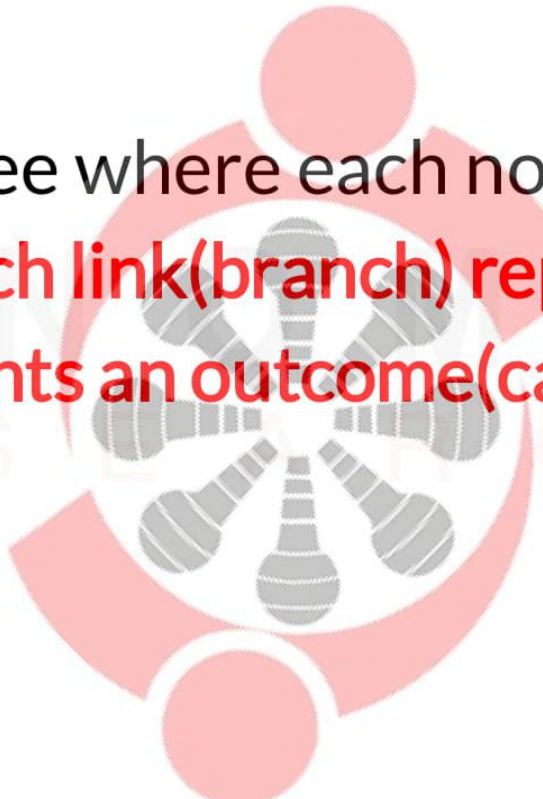


INNOMATİCS
RESEARCH LABS

Decision Tree (Rule-Based)

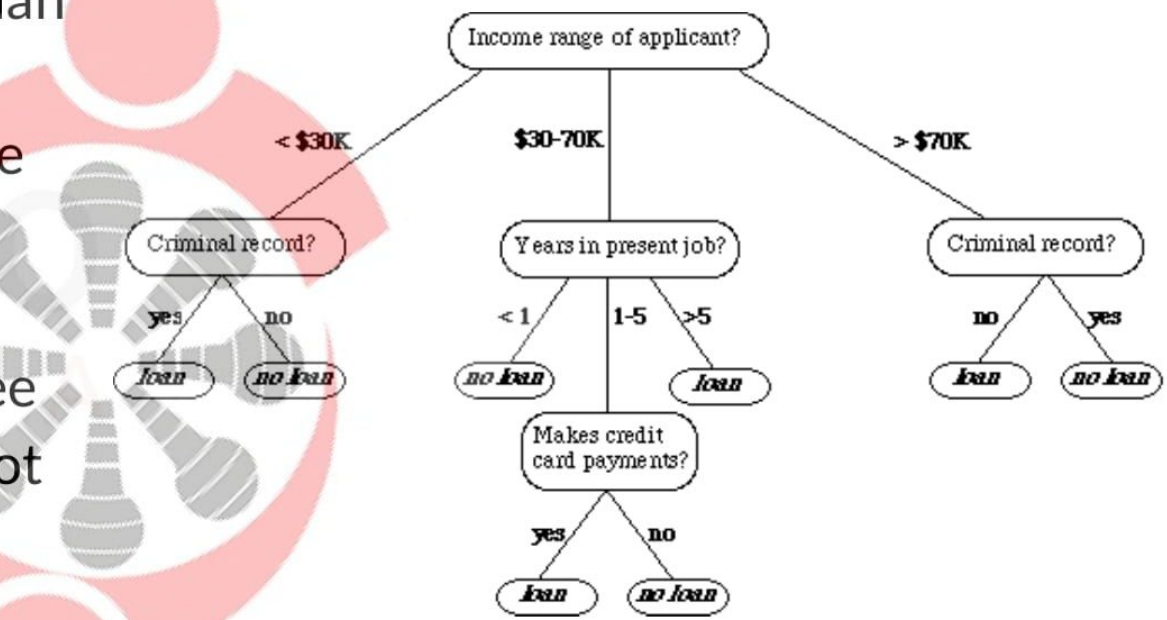
What is Decision Tree?

A decision tree is a tree where each node represents a **feature(attribute), each link(branch) represents a decision(rule) and each leaf represents an outcome(categorical or continuous value).**



Decision Tree

- Decision trees often mimic the human level thinking so its so simple to understand the data and make some good interpretations.
- Decision trees actually make you see the logic for the data to interpret(not like black box algorithms like SVM,NN,etc..)
- For example : if we are classifying bank loan application for a customer, the decision tree may look like this



INNOMATICS RESEARCH LABS

Advantages

- Simple to understand, interpret, visualize.
- Use of statistical properties of all the examples (e.g., information gain for ID3) results in a less sensitive algorithm to errors in the training examples.
- Algorithms can be easily extended to handle noisy training data by modifying its termination criterion to accept hypotheses that imperfectly fit the training data.
- Non-linear relations can be captured.
- Can be used for feature engineering as decision trees implicitly perform variable screening or feature selection.

Disadvantages

- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement (Random Forest).
- It fails to determine how many alternative decision trees are consistent with the available training data.
- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting.
- Unstable. Small variation in data - completely different tree formation. This is called variance, it reduces by using methods like bagging and boosting.

Types of Decision Trees

- ***Categorical Variable Decision Tree***: Decision Tree which has categorical target variable then it called as categorical variable decision tree.
- ***Continuous Variable Decision Tree***: Decision Tree which has continuous target variable then it is called as Continuous Variable Decision Tree.

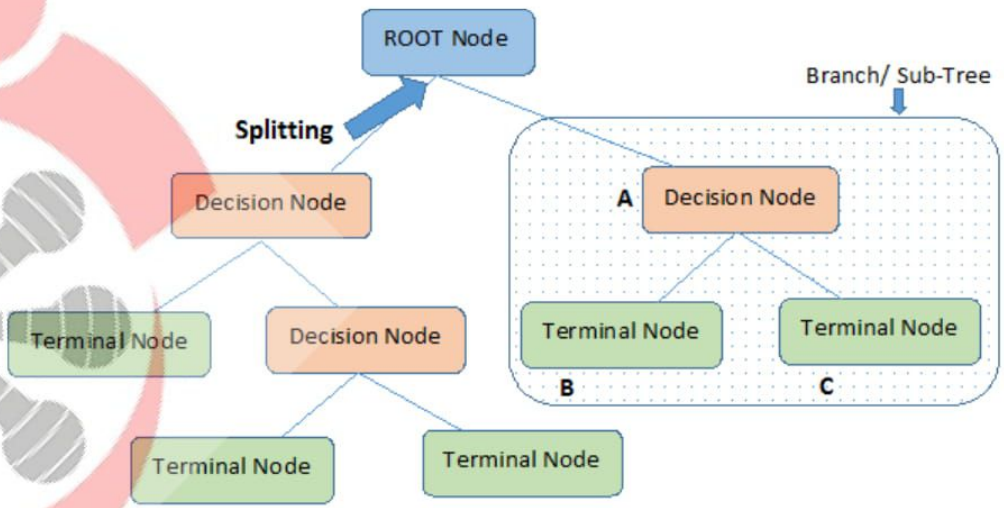
Important Terminology related to Decision Trees

1. ***Root Node***: It represents entire population or sample and this further gets divided into two or more homogeneous sets.

1. ***Splitting***: It is a process of dividing a node into two or more sub-nodes.

2. ***Decision Node***: When a sub-node splits into further sub-nodes, then it is called decision node.

3. ***Leaf/ Terminal Node***: Nodes with no children (no further split) is called Leaf or Terminal node.



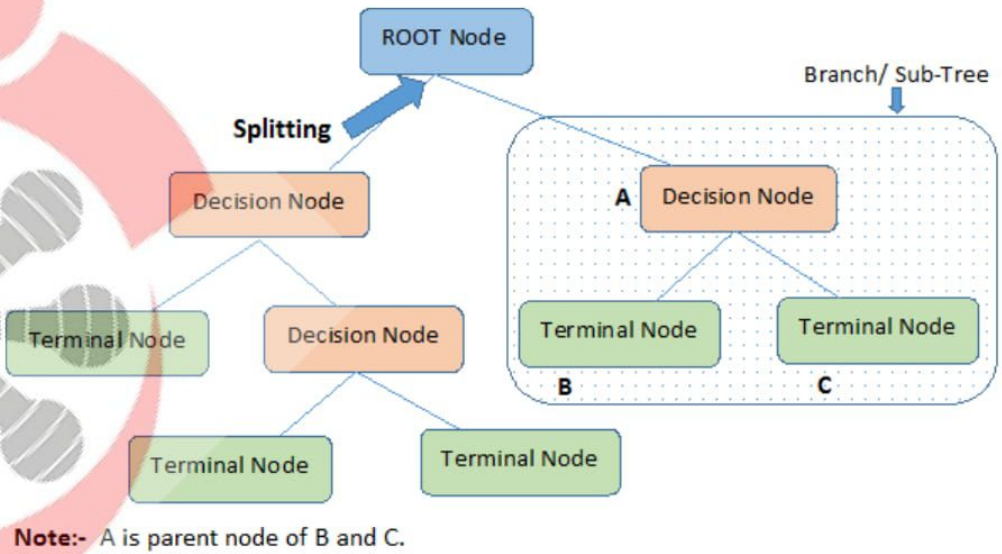
INNOMATICS RESEARCH LABS

Important Terminology related to Decision Trees

1. Pruning: When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.

2. Branch / Sub-Tree: A subsection of decision tree is called branch or sub-tree.

3. Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.



INNOMATICS RESEARCH LABS

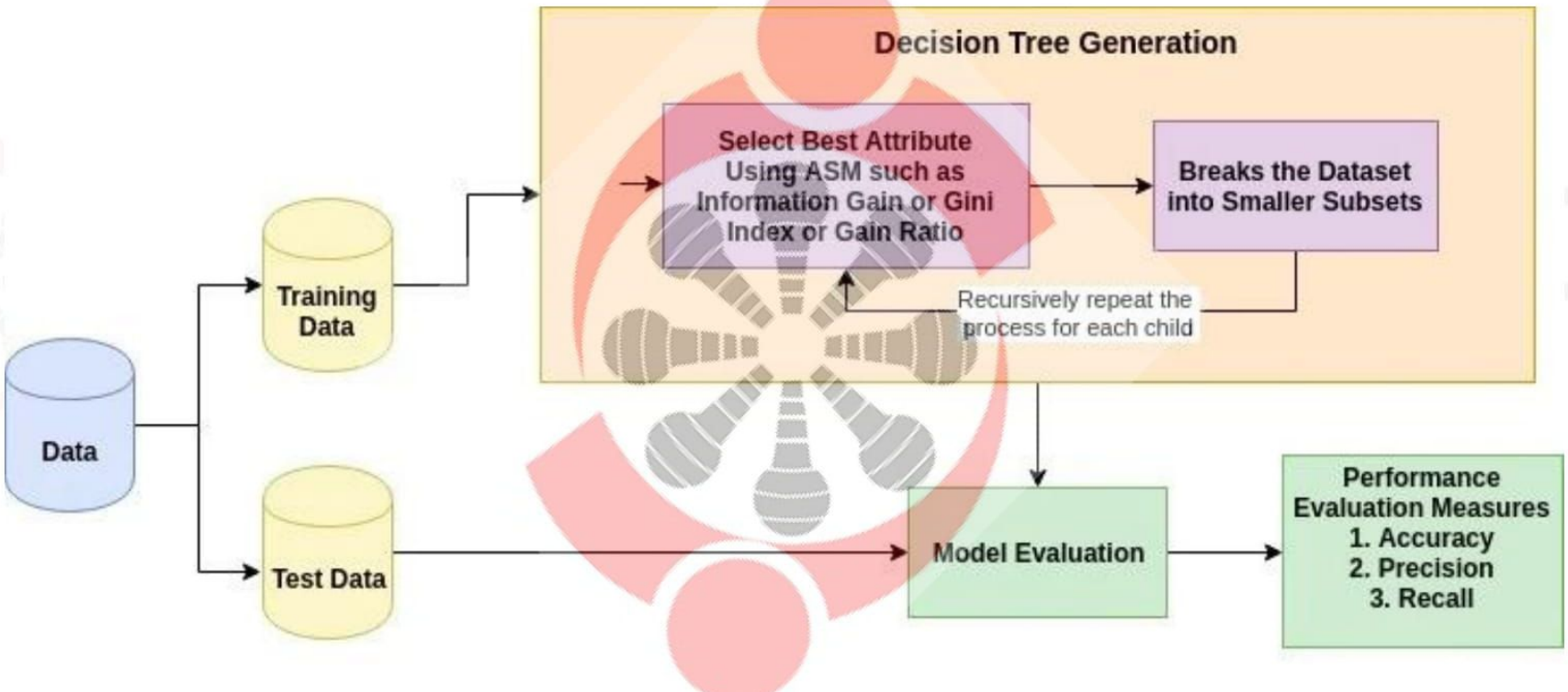
Decision Tree Algorithm Work

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:

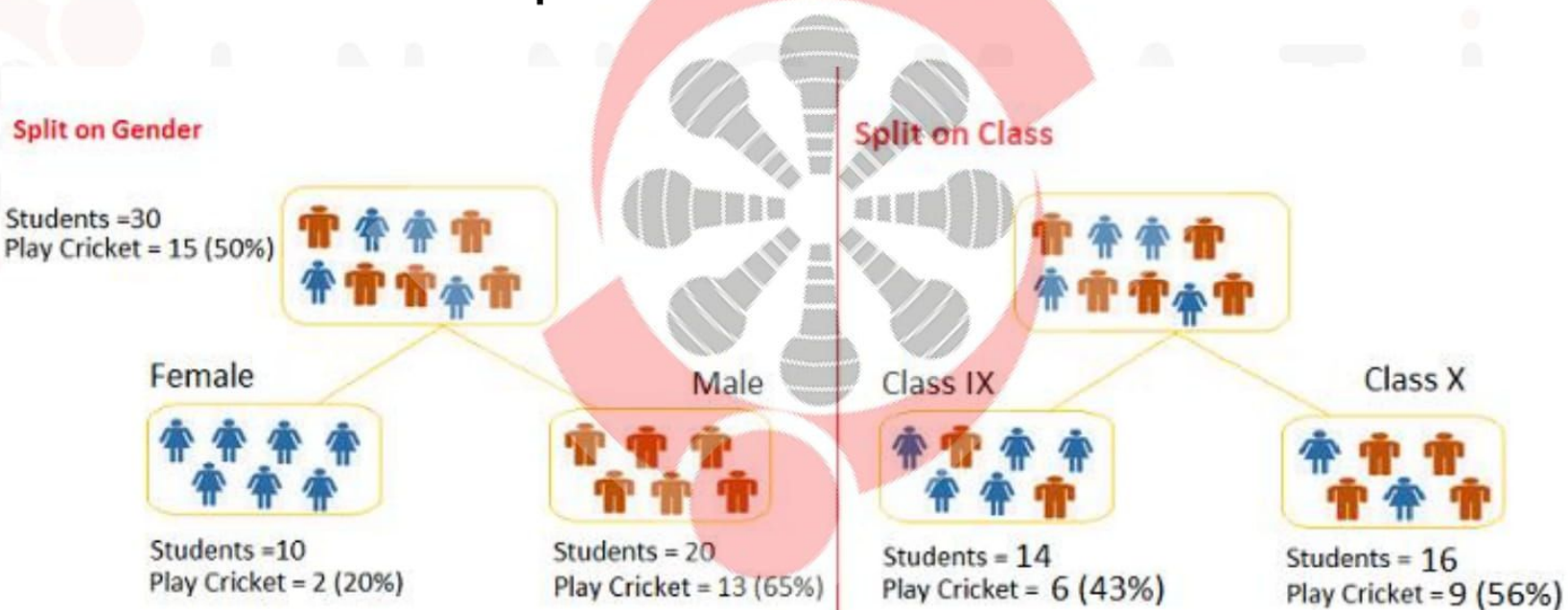
- All the tuples belong to the same attribute value.
- There are no more remaining attributes.
- There are no more instances.

Decision Tree Generation



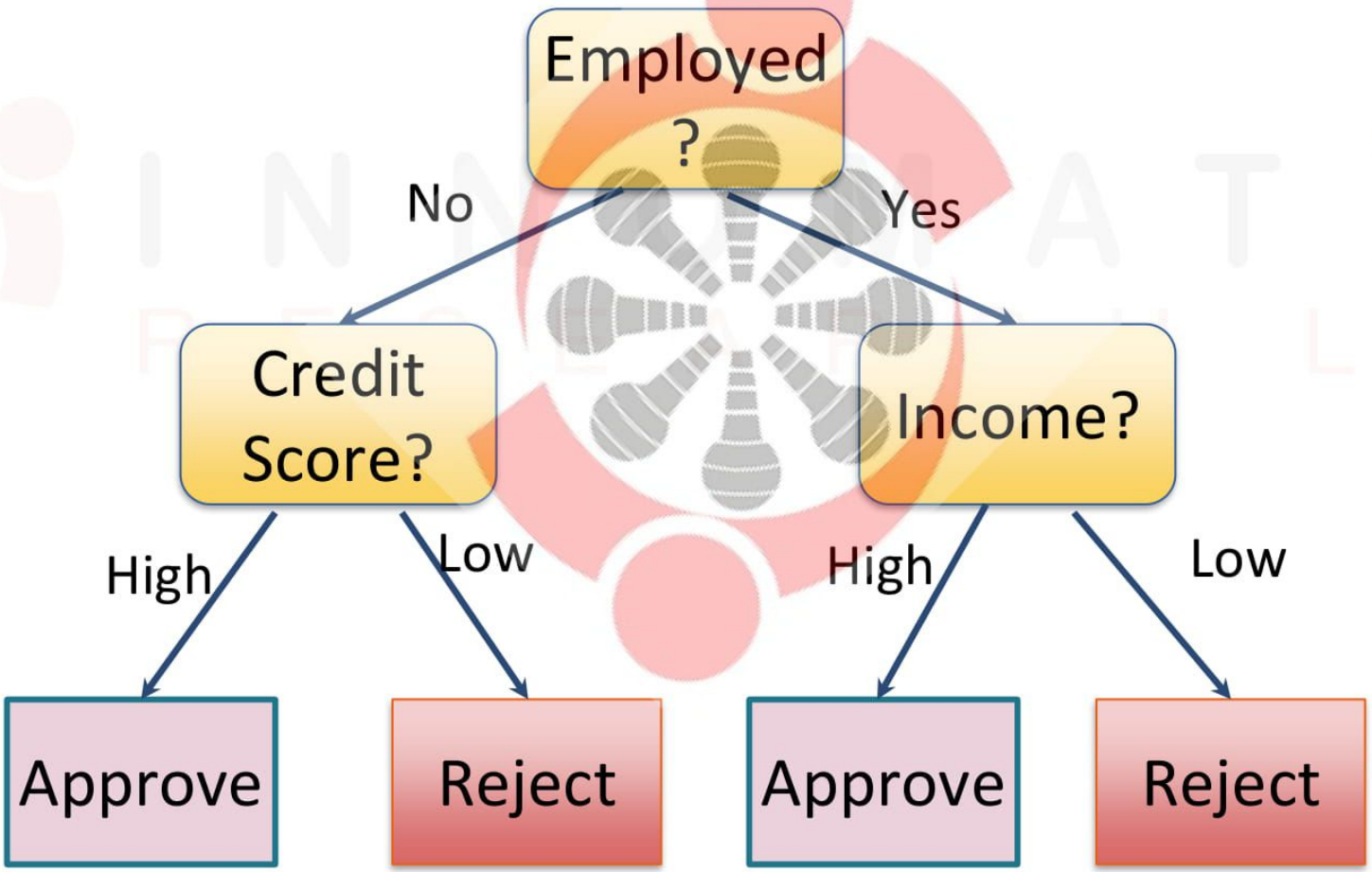
How do Decision Trees Work?

- ***Splitting*** - The process of partitioning the data set into subsets. Splits are formed on a particular variable

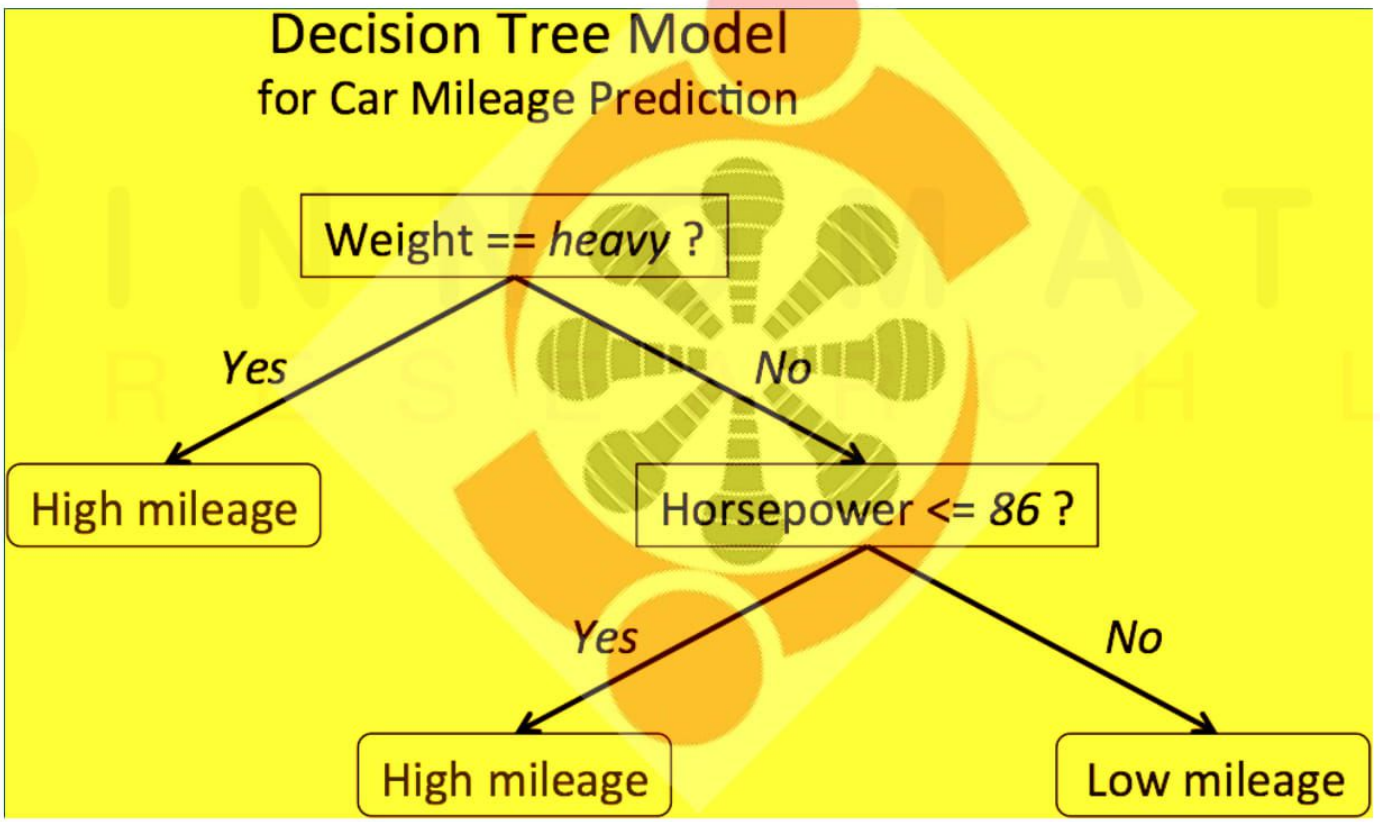


Splitting is done on various factors

Decision Tree Example 1:- Whether to approve a loan



Decision Tree Example



Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the **smallest tree** that is consistent with the data (**Bias**)
  - the tree with the least depth?
  - the tree with the fewest nodes?
- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data.

Example Data

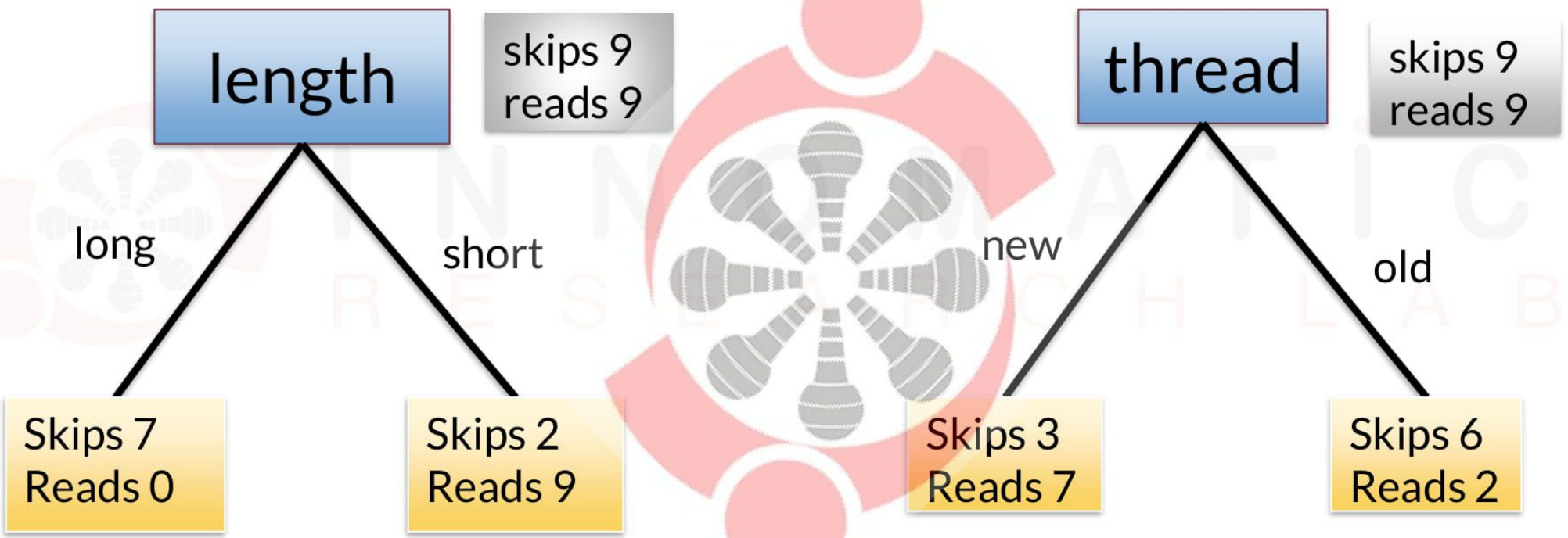
Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	Home
e2	reads	unknown	new	short	Work
e3	skips	unknown	old	long	Work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

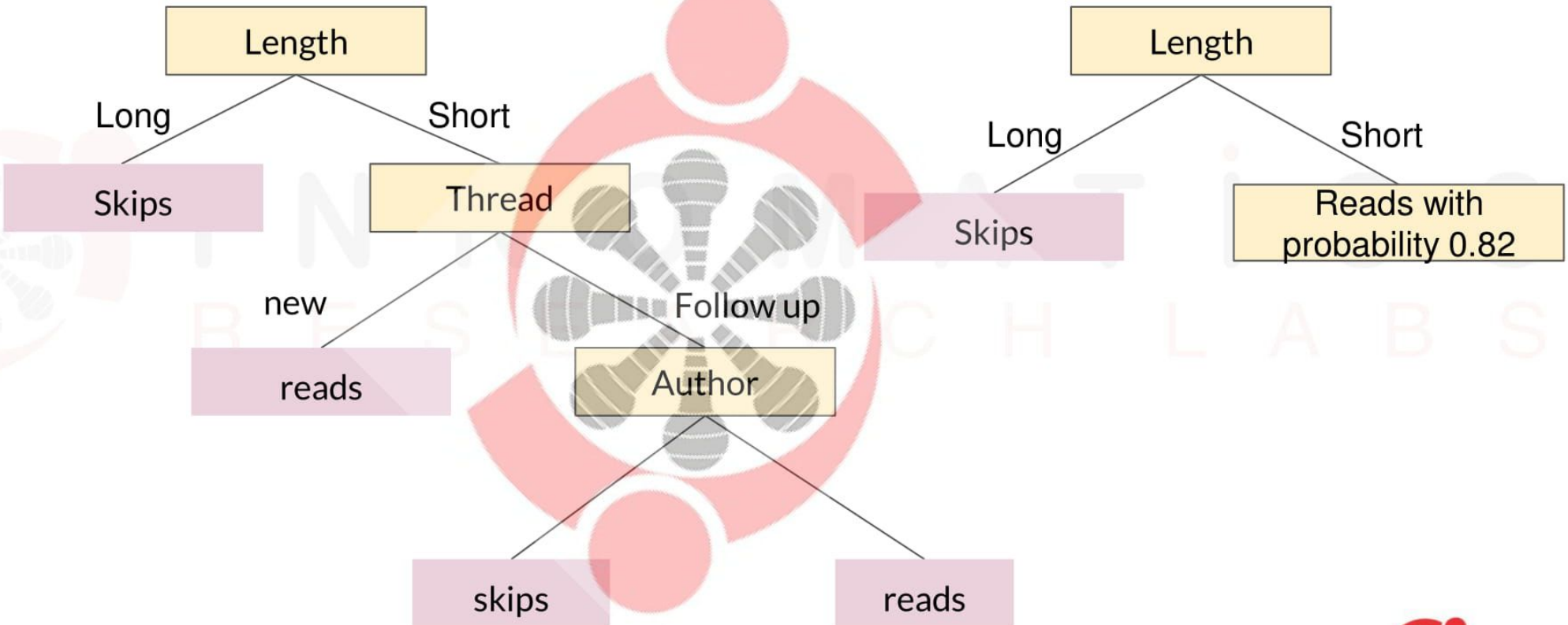
New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work

Possible Splits



Two Example DTs

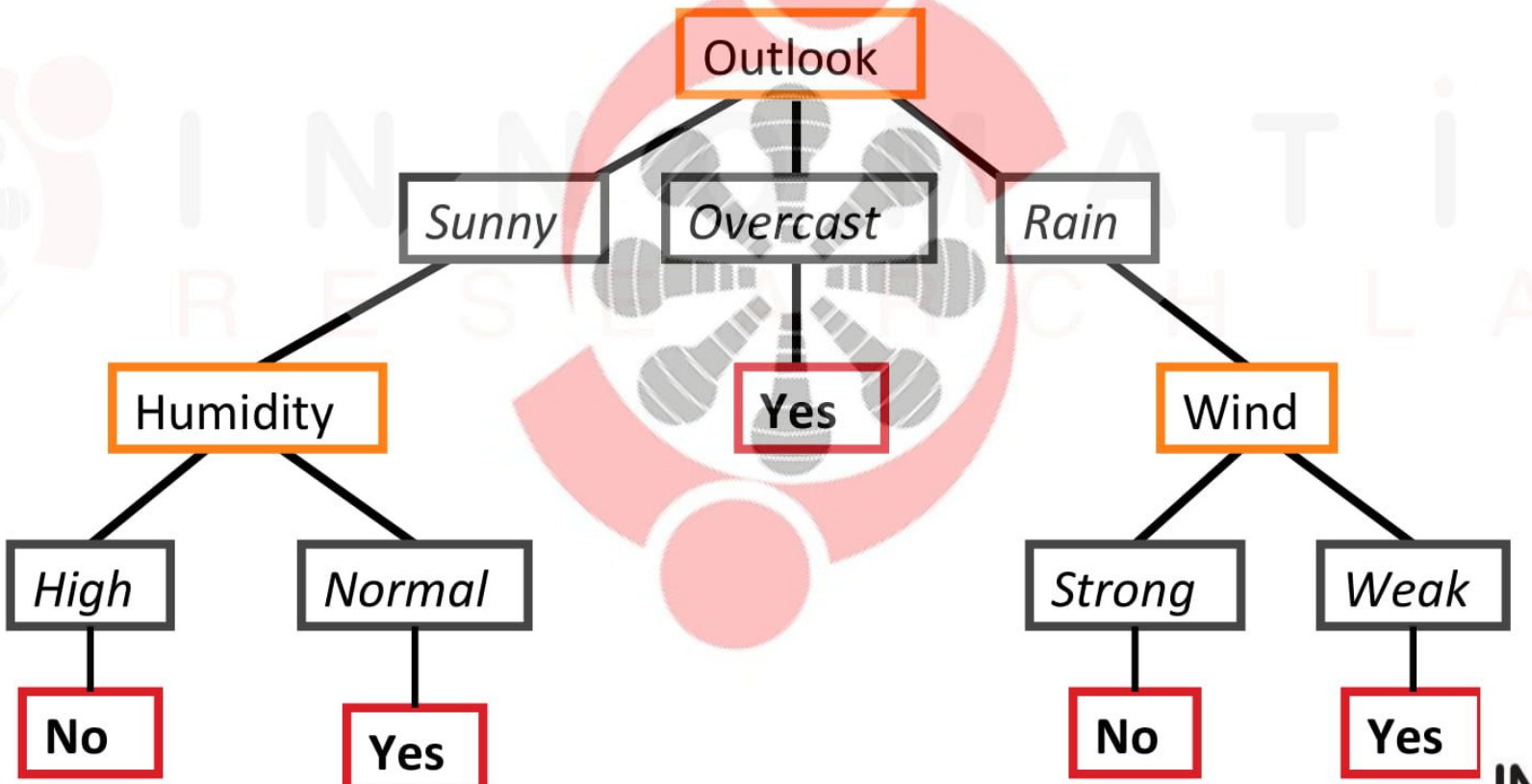


Decision Tree for Play Tennis

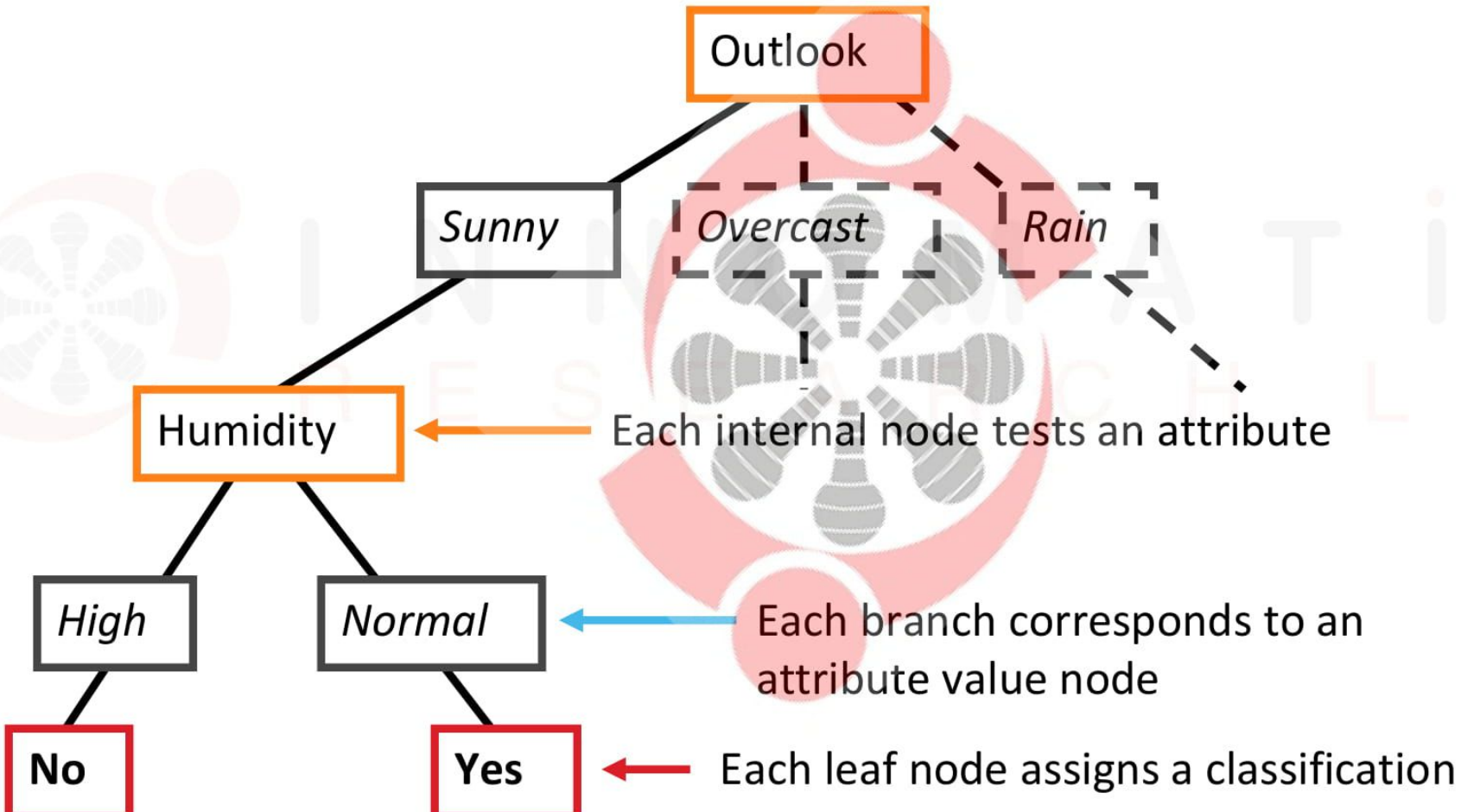
- Attributes and their values:
  - Outlook: Sunny, Overcast, Rain
  - Humidity: High, Normal
  - Wind: Strong, Weak
  - Temperature: Hot, Mild, Cool

- Target concept - Play Tennis: Yes, No

Decision Tree for Play Tennis

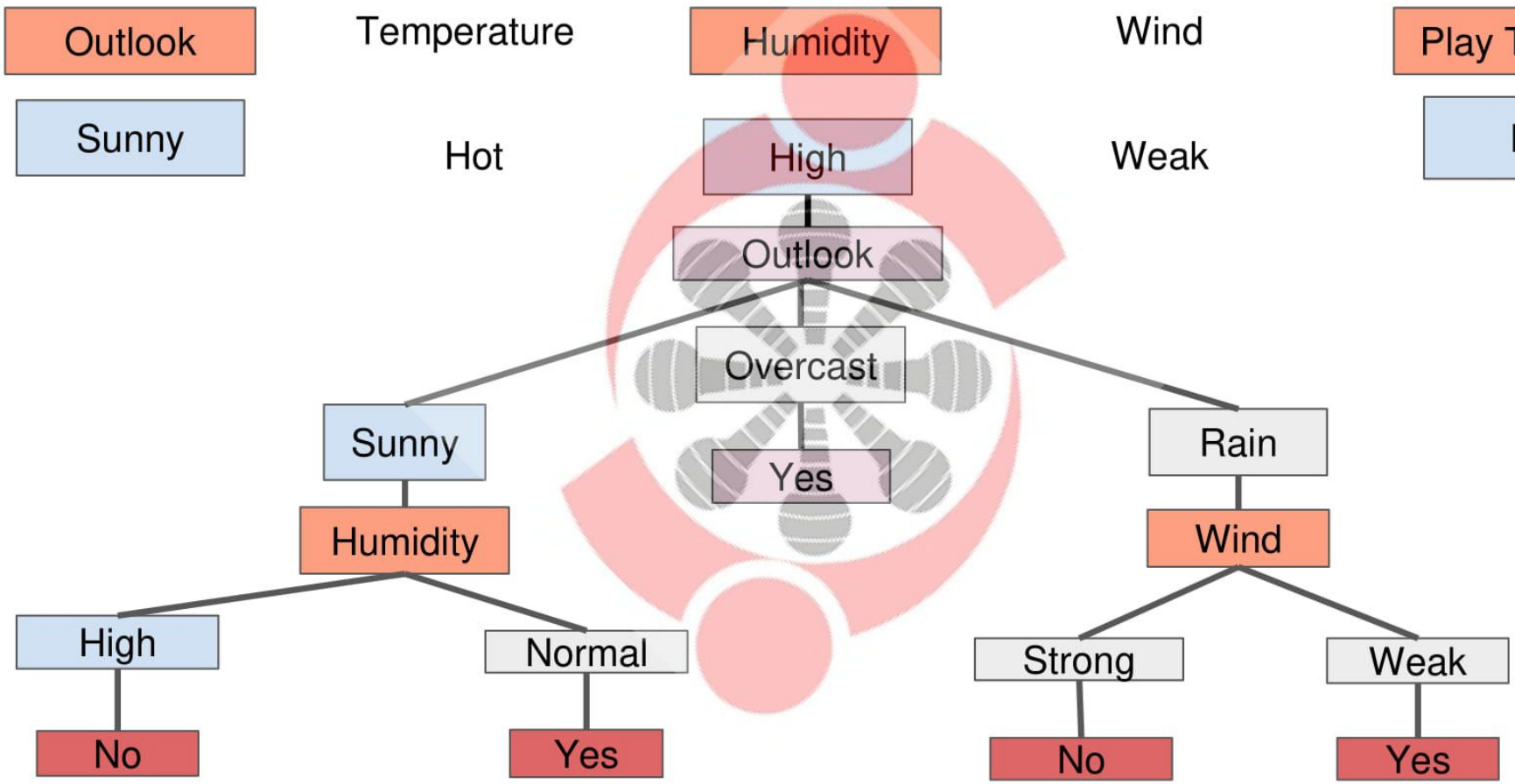


Decision Tree for Play Tennis



Decision Tree for Play Tennis

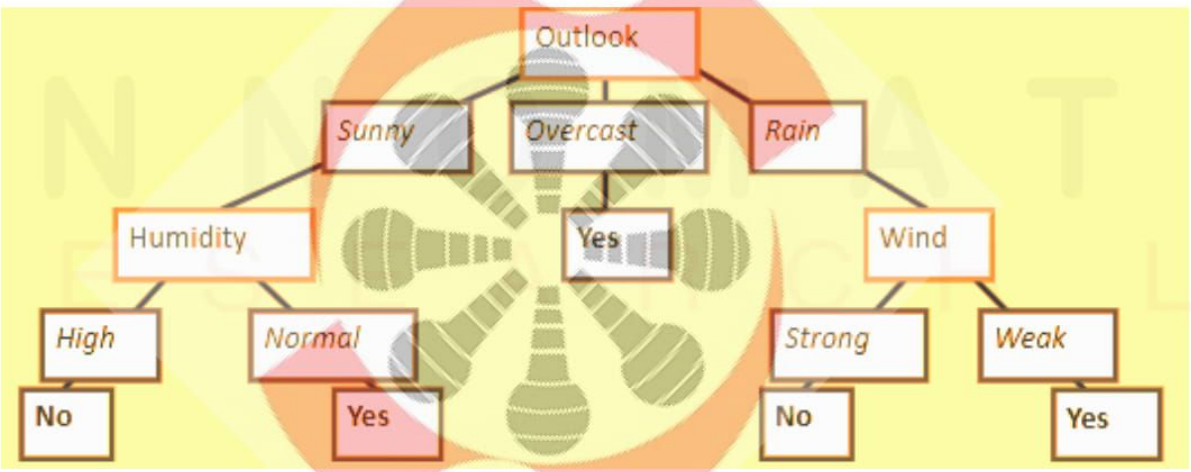
Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No



INNOMATICS
RESEARCH LABS

Decision Tree

decision trees represent disjunctions of conjunctions



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

Searching for a Good Tree

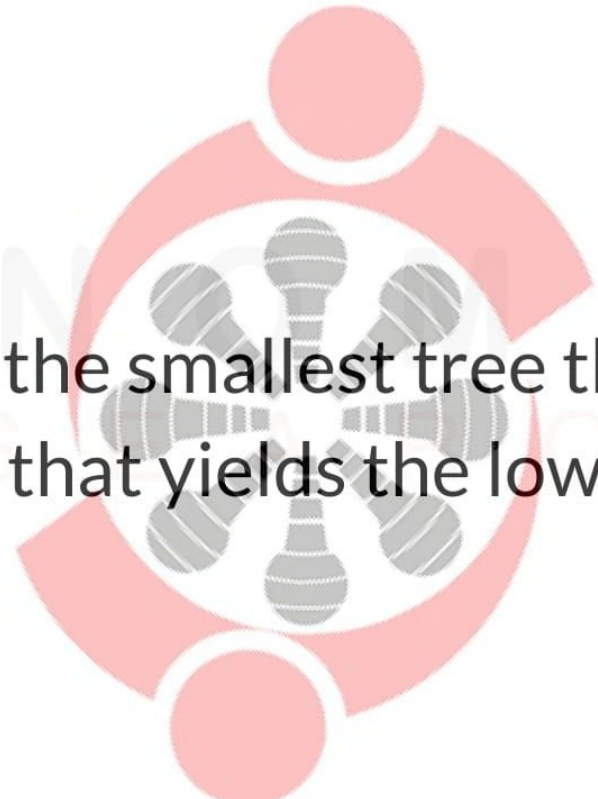
How should you go about building a decision tree?

- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values
- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

Conditions for Stopping Partitioning

- All samples for a given node belong to the same class.
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf.
- There are no samples left.

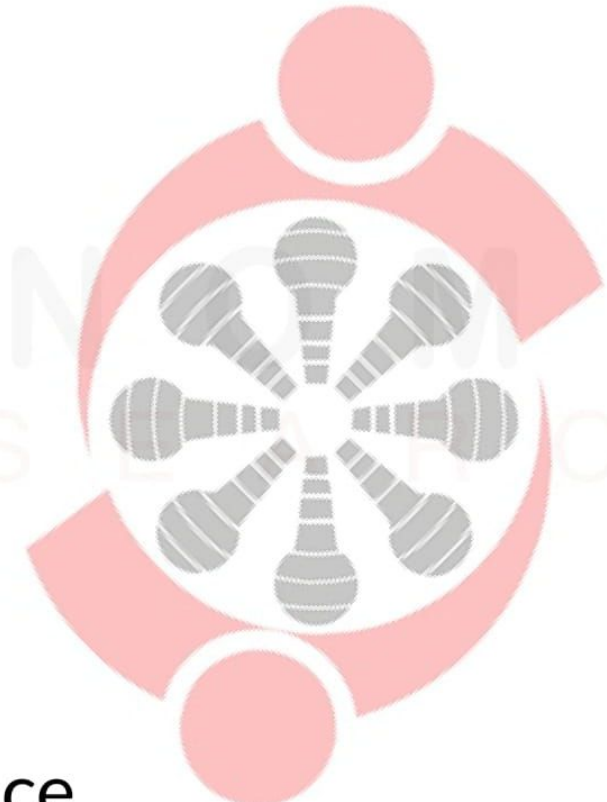
Tree Selection



The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest cross-validated error.

Algorithm Used in Decision Trees

1. ID3 and C4.5
2. Gini Index
3. Chi-Square
4. Reduction in Variance



ID3 Algorithm

1. Establish the Target Classification Attribute.
2. Compute Classification Entropy
3. For each attribute in R, calculate IG using the classification attribute.
4. Select attribute with the highest IG to be the next Node in the tree (starting from the Root node)
5. Remove the Node Attribute, creating reduced table R
6. Repeat steps 3-5 until all attributes have been used, or the same classification value remains for all rows in the reduced table.

Algorithm Used in Decision Trees:- ID3

- The core algorithm for building decision trees is called ID3. This algorithm employs a top-down, greedy search through the space of possible branches with no backtracking.

- ID3 uses Entropy and Information Gain to construct a decision tree.

Top-Down Induction of Decision Trees ID3

1. Which node to proceed with?

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

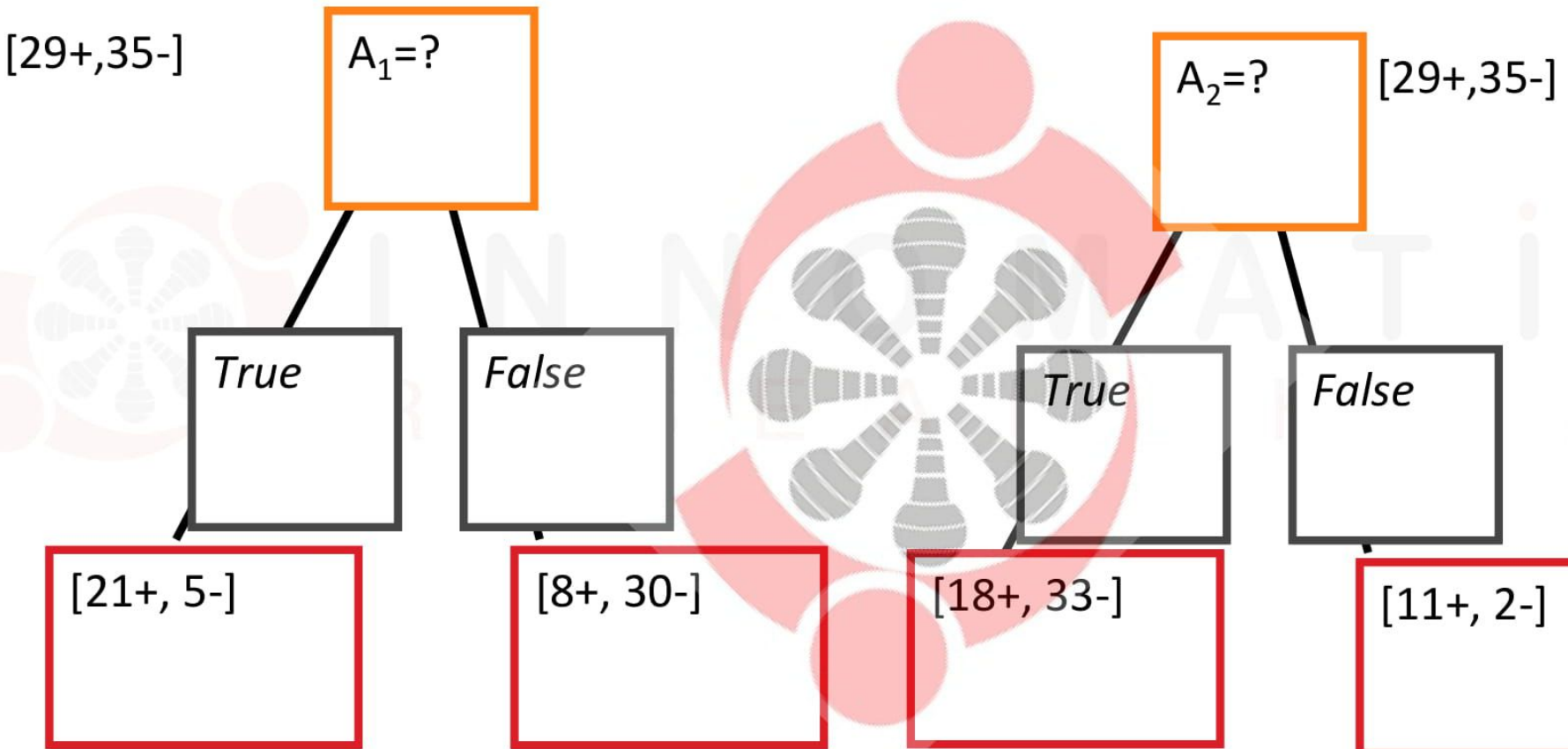
2. When to stop?

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split

- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

Which Attribute is “best”



Principled Criterion

- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

Entropy

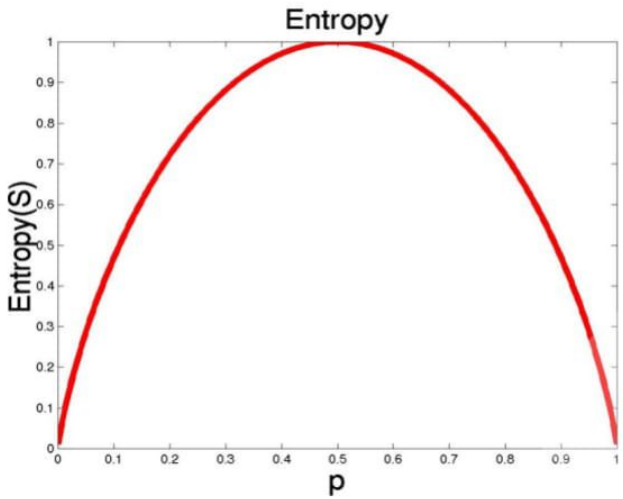
- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns (− log2p) bits to message having probability p
- S is a sample of training examples
  - p+ is the proportion of positive examples in S
  - p− is the proportion of negative examples in S

Entropy

Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

$$Entropy(S) = p+(-log2p+) + p-(-log2p-) = -p+log2p+-p-log2p$$

Entropy



- The entropy is 0 if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

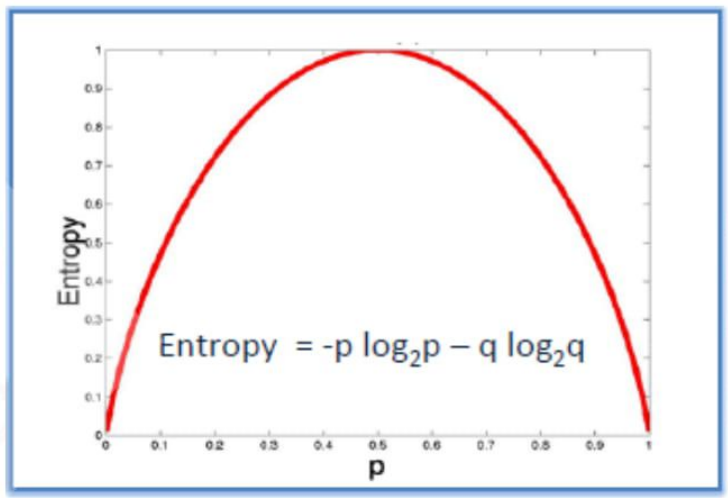
- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$Entropy(S) = -p_+ log_2 p_+ - p_- log_2 p$$

INNOMATICS
RESEARCH LABS

Algorithm Used in Decision Trees: Entropy

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous).
- ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided then it has entropy of one.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Entropy Calculation

To build a decision tree, There are two types of entropy:

a) Entropy using the frequency table of one attribute:

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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

Algorithm Used in Decision Trees:- Information Gain

The information gain is based on the decrease in entropy after a data-set is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

***Step 1*: Calculate entropy of the target:**

$$\text{Entropy(PlayGolf)} = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

- ***Step 2**: The dataset is then split on the different attributes. The entropy for each branch is calculated.*
- ***Then it is added proportionally, to get total entropy for the split.***
- ***The resulting entropy is subtracted from the entropy before the split.***
- ***The result is the Information Gain, or decrease in entropy.***

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

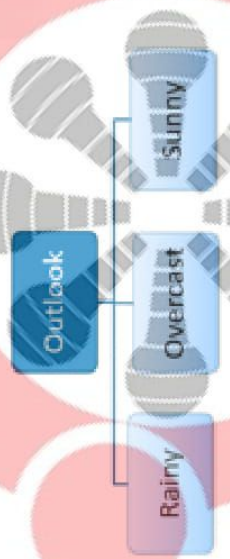
= 0.940 – 0.693 = 0.247

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

***Step 3**: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.*

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

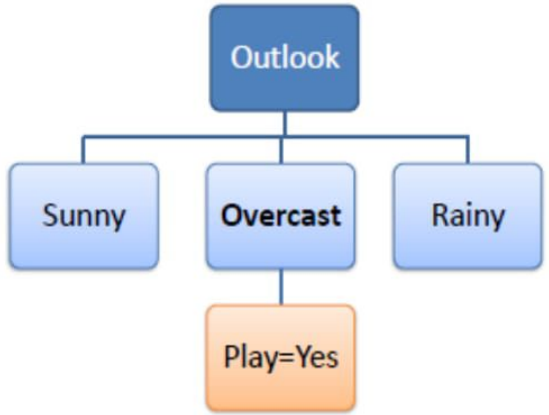
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes

Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

INNOMATICS
RESEARCH LABS

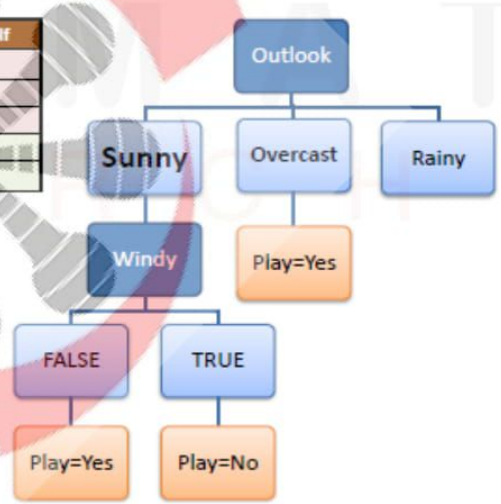
***Step 4a**: A branch with entropy of 0 is a leaf node;*

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



***Step 4b**: A branch with entropy more than 0 needs further splitting;*

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



***Step 5**: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.*

Model	Engine	SC/Turbo	Weight	Fuel Eco	Fast
Prius	small	no	average	good	no
Civic	small	no	light	average	no
WRX STI	small	yes	average	bad	yes
M3	medium	no	heavy	bad	yes
RS4	large	no	average	bad	yes
GTI	medium	no	light	bad	no
XJR	large	yes	heavy	bad	no
S500	large	no	heavy	bad	no
911	medium	yes	light	bad	yes
Corvette	large	no	average	bad	yes
Insight	small	no	light	good	no
RSX	small	no	average	average	no
IS350	medium	no	heavy	bad	no
MR2	small	yes	average	average	no
E320	medium	no	heavy	bad	no

Sample Problem

Attribute "Model" can be tossed out, since it's always unique, and it doesn't help our result.

INNOMATICS RESEARCH LABS

Sample Problem - Information Gain

1. Engine: 6 small, 5 medium, 4 large
2. 3 Values for attribute engine, so we need 3 entropy calculations

small: 5 no, 1 yes	$I_{small} = -(5/6)\log_2(5/6)-(1/6)\log_2(1/6) = \sim 0.65$
Medium: 3 no, 2 yes	$I_{medium} = -(3/5)\log_2(3/5)-(2/5)\log_2(2/5) = \sim 0.95$
Large: 2 no, 2 yes	$I_{large} = 1$ (evenly distributed subset)

- $IG_{Engine} = IE(s) - [(6/15)*I_{small} + (5/15)*I_{medium} + (4/15)*I_{la}]$
- $IG_{Engine} = 0.971-0.85 = 0.121$

Sample Problem: Information Gain

- SC/Turbo 4 yes, 11 no
- 2 Values for attributes SC/Turbo, so we need 2 entropy calculations

Yes: 2 yes, 2 no	$I_{turbo} = 1$ (evenly distributed subset)
No: 3 yes, 8 no	$I_{no\ turbo} = -(3/11)\log_2(3/11)-(8/11)\log_2(8/11) = \sim 0.84$

- $IG_{turbo} = IE(s) - [(4/15)*I_{turbo} + (11/15)*I_{noturbo}]$
- $IG_{turbo} = 0.971-0.886 = 0.085$

INNOMATICS RESEARCH LABS

Sample Problem: Information Gain

1. Weight: 6 Average, 4 Light, 5 Heavy
2. 3 Values for attribute weight, so we need 3 entropy calculations

Average: 3 no, 3 yes	$I_{average} = 1$ (evenly distributed subset)
Light: 3 no, 1 yes	$I_{light} = -(¾)\log_2(¾)-(¼) = \sim 0.81$
Heavy: 4 no, 1 yes	$I_{heavy} = -(⅘)\log_2(⅘)-(⅕)\log_2(⅕) = \sim 0.72$

$IG_{weight} = IE(S)-[(6/15)*I_{average}+(4/15)*I_{light}+(5/15)*I_{heavy}]$

$IG_{weight} = 0.971-0.856 = 0.115$

Sample Problem: Information Gain

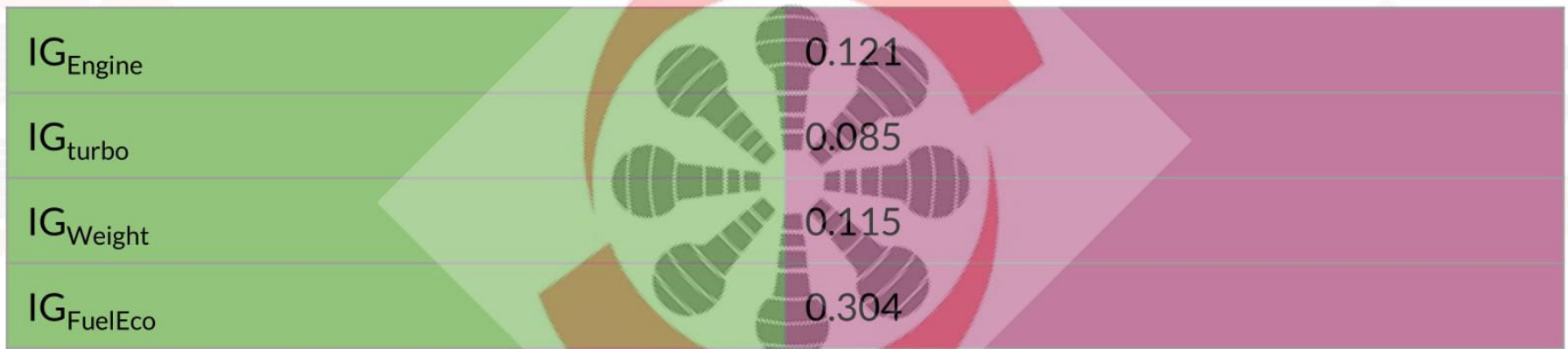
1. Fuel Economy: 2 good, 3 average, 10 bad
2. 3 values for attribute Fuel Eco, so we need 3 entropy calculations

Good: 0 yes, 2 no	I_{goode} = 1 (no variability)
Average: 0 yes, 3 no	$I_{average}$ = 0 (no variability)
Bad:5 yes, 5 no	I_{bad} = 1 (evenly distributed subset)

We can omit calculations for good and average since they always end up not fast.

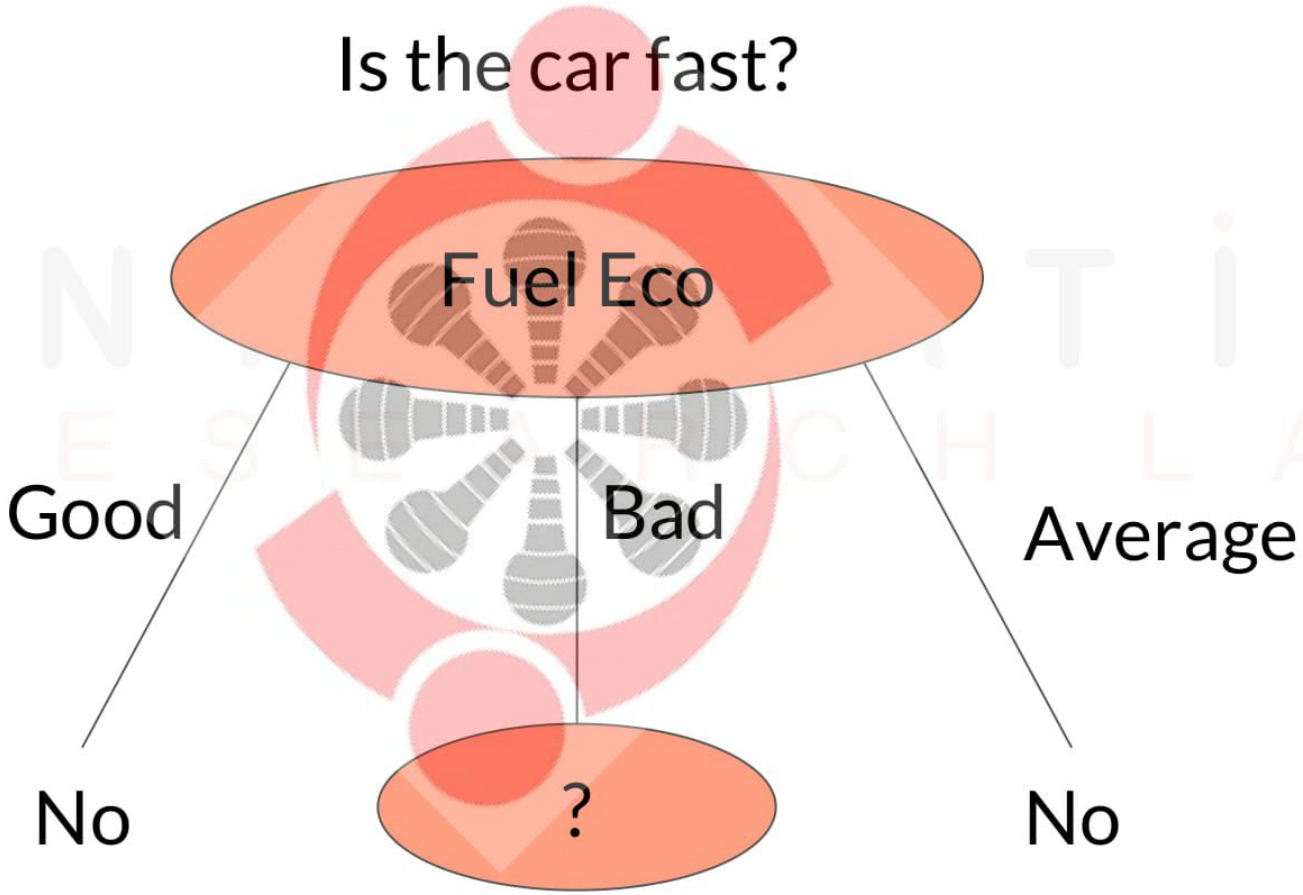
- $IG_{FuelEco} = IE(S)-[(10/15)*I_{bad}]$
- $IG_{FuelEco} = 0.971 - 0.667 = 0.304$

Sample Problem: Choosing the Root Node



Our best pick is Fuel Eco, and we can immediately predict the car is not fast when fuel economy is good or average.

Root of Decision Tree



ID3: Shortcomings

- ID3 attempts to learn the shortest decision tree from the learning data which may not be always the best.
- Requires learning to have completely consistent patterns with no uncertainty or missing values allowed
- IG for unique keys such as ”Employee ID” will be very high
  - Need to fix IG to avoid such trivial splits

Handling Missing Values

- Missing value denoted "?" in C4.X
- Simple Idea: Treat missing as a separate value
- *Q: When this not appropriate?*
- A: When values are missing due to different reasons

C4.5

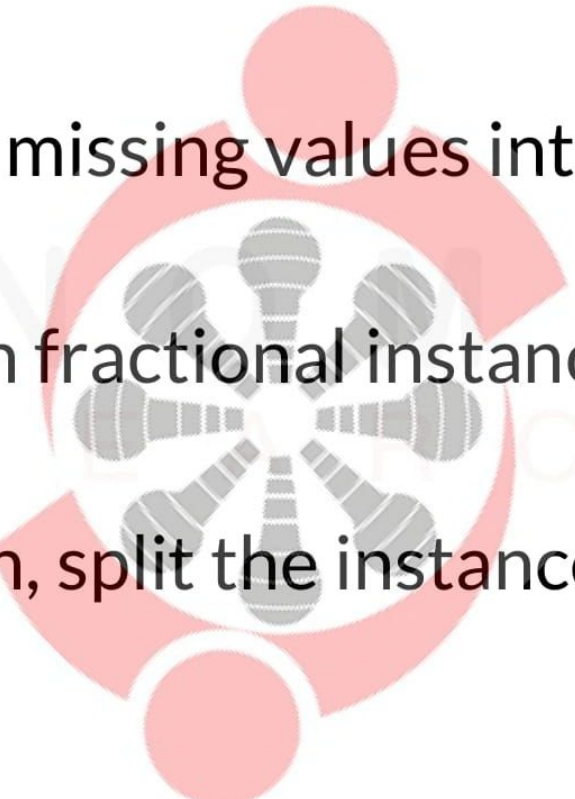
- C4.5's method
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Standard Bernoulli-process-based method
  - Shaky statistical assumptions (based on training data)

Handling Missing Values

- Example 1: Gene expression could be missing when it is very high or very low
- Example 2: Field IsPregnant = missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

Missing Values: Advanced

- Split instances with missing values into pieces
- Info gain works with fractional instances
- During classification, split the instance into pieces in the same way



Missing Values: Example

Attribute 1	Attribue 2	Attribute 3	Class
A	70	TRUE	CLASS 1
A	90	TRUE	CLASS 2
A	85	FALSE	CLASS 2
A	95	FALSE	CLASS 2
A	70	FALSE	CLASS 2
?	90	TRUE	CLASS 1
B	78	FALSE	CLASS 1
B	65	TRUE	CLASS 1
B	75	FALSE	CLASS 1
C	80	TRUE	CLASS 2
C	70	TRUE	CLASS 2
C	80	FALSE	CLASS 1
C	80	FALSE	CLASS 1
C	96	FALSE	CLASS 1

Missing Values: Example

T1: (attribute1 = A)

Att. 2	Att.3	Class	W
70	True	C1	1
90	True	C2	1
85	False	C2	1
95	False	C2	1
70	False	C1	1
90	True	C1	5/13

T1: (attribute1 = B)

Att. 2	Att. 3	Class	W
90	True	C1	3/13
78	False	C1	1
65	True	C1	1
75	False	C1	1

T1: (attribute1 = C)

Att. 2	Att. 3	Class	W
80	True	C2	1
70	True	C2	1
80	False	C1	1
80	False	C1	1
96	Flase	C1	1
90	True	C1	5/13

Missing Values: Example

If Attribute1 = A Then
  If Attribute2 <= 70 Then
   Classification = CLASS1 (2.0/0);
else

  Classification = CLASS (3.4/0.4);
elseif Attribute1 = B Then
  Classification = CLASS1 (3.2/0);

elseif Attribute1 = C Then
  If Attribute3 = true Then
   Classification = CLASS2 (2.4/0)
else
   Classification = CLASS1 (3.0/0)

Splitting based on GainRATIO

Gain Ratio:

$$GainRATIO_{split} = GAIN_{split}/SplitINFC \quad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

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.

- Designed to overcome the disadvantage of Information Gain.

Handling Numeric Attributes

Split on Temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

- E.g temperature<71.5:yes/4, no/2
- temperature≤71.5:yes/5, no/3
    - Info([4,2],[5,3])
    - =6/14 info([4,2]) + 8/14 info([5,3])
    - =0.939 bits

- Place split points halfway between values
- Can evaluate all split points in one pass.

Continuous Value

When you have continuous value unlike this example, Suppose attribute A has continuous value so choose best –split for A , where the split-point is a threshold on A.

First sort the A in increasing order, so the midpoint between each pair of adjacent value is considered as possible split-point.

Avoid Repeated Sorting

- Sort instances by the values of the numeric attribute
  - ➢ Time complexity for sorting: $O(n \log n)$
- *Q. Does this have to be repeated at each node of the tree?*
  - ➢ A: No! Sort order for children can be derived from sort order for parent
- Time complexity of derivation: O(n)
- Drawback: need to create and store an array of sorted indices for each numeric attribute.

More Speed Up

- Entropy only needs to be evaluated between points of different classes(Fayyad & Irani, 1992)

Value	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Class	yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal.

Binary vs. Multi-Way Splits

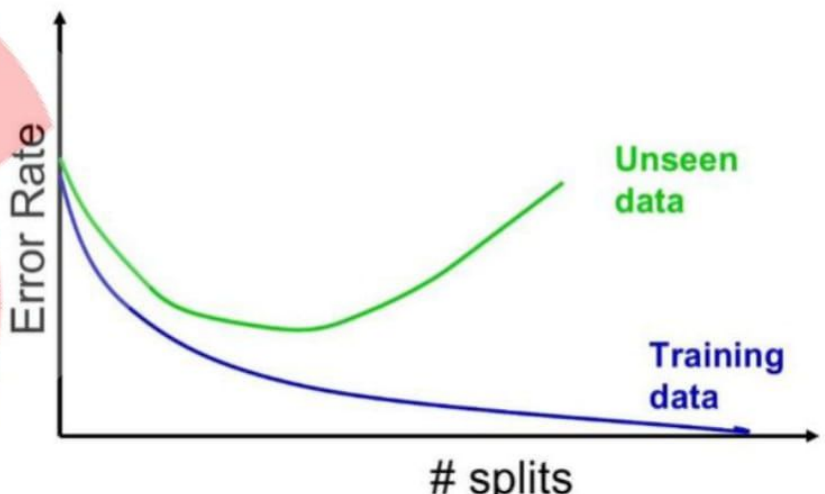
- Splitting (multi-way) on a nominal attribute exhausted all information in that attribute.
  - ➢ Nominal attributes is tested (at most) once on any path in the tree
- Not so for binary splits on numeric attributes.
  - ➢ Numeric attributes may be tested several times along a path in the tree

Binary vs. Multi-Way Splits

- Disadvantage: tree is hard to read
- Remedy:
  - ➢ Pre-discretize numeric attributes, or
  - ➢ Use multi-way splits instead of binary ones

Pruning

- Goal: Prevent Overfitting to noise in the Data.
- Two Types of Pruning
- Post-Pruning: Taking a fully grown decision unreliable parts.
- Pre-pruning: Stope growing a branch when information becomes unreliable.



Pre-Pruning

- Based on Statistical significant test.
- Stop growing the tree when there is no statistically significant association between anty attribute and the class at particular node.
- Use Chi-squared test (CHAID)

Post-Pruning

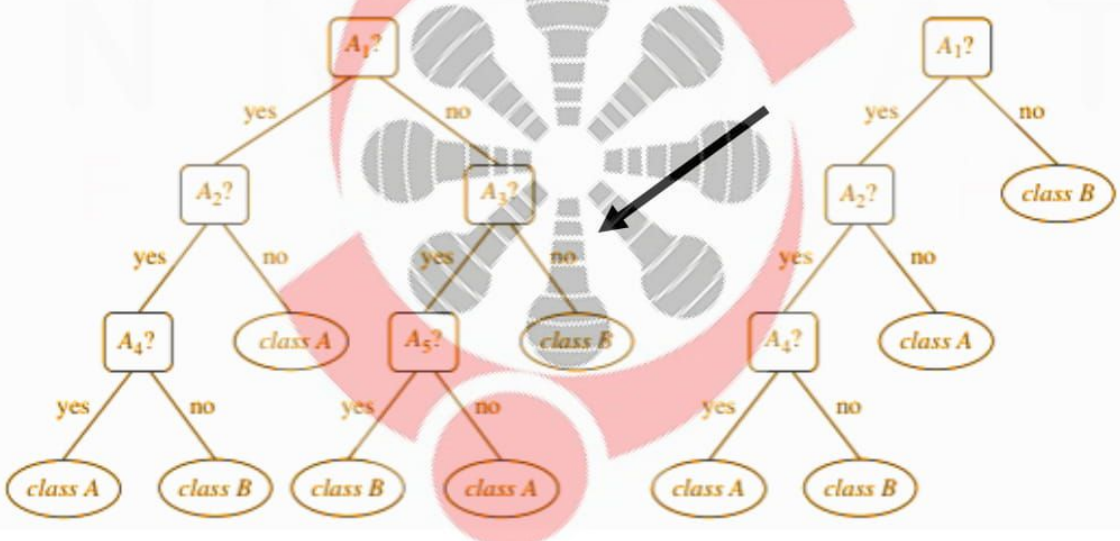
- Taking a fully grown decision tree.
- Remove unreliable parts.

Problem: Some subtree might be due to chance effects.

- Two Types: Subtree replacement.
  Subtree Raising.

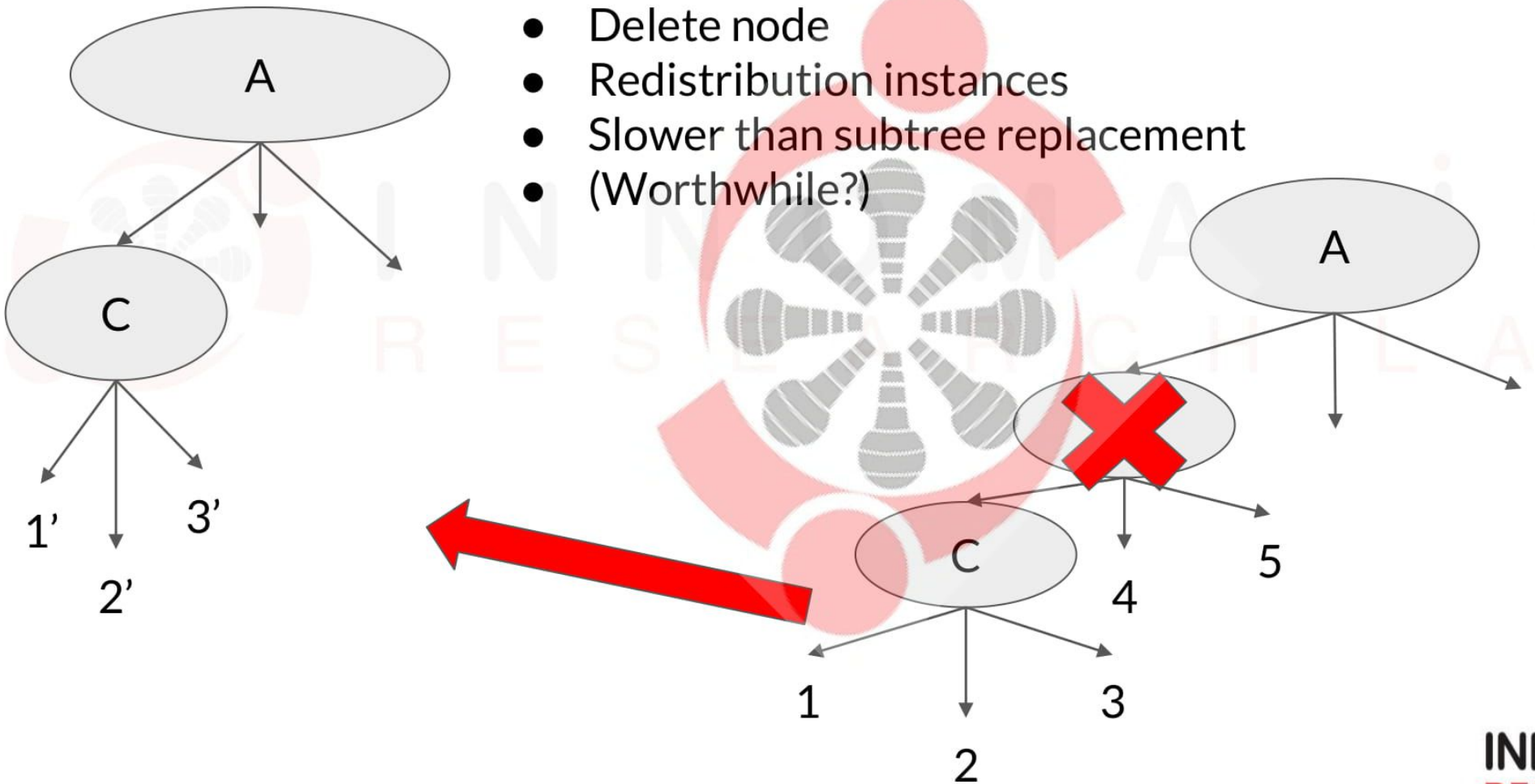
Subtree Replacement: Example

1. Bottom-up approach
2. Consider replacing only after considering all the subtree.



Subtree Raising

- Delete node
- Redistribution instances
- Slower than subtree replacement
- (Worthwhile?)



Estimating Error Rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
- *Q: Why it would result in very little pruning?*
- Use hold-out set for pruning
- ("reduced-error pruning")

Chi Square

- It is an algorithm to find out the statistical significance between the differences between sub nodes and parent node.
- We measure it by sum of squares of standardised differences between observed and expected frequencies of target variable.

Chi Square

1. It works with categorical target variable “Success” or “Failure”.
2. It can perform two or more splits.
3. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.
4. Chi-Square of each node is calculated using formula,
5. Chi-square = $((\text{Actual} - \text{Expected})^2 / \text{Expected})^1/2$
6. It generates tree called CHAID
   (Chi-square Automatic Interaction Detector).

Steps to calculate Chi-Square for a Split

1. Calculate Chi-square for individual node by calculating the deviation for Success and Failure both

1. Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

Steps to calculate Chi-Square for a Split

Example: Let's work with above example that we have used to calculate Gini.
Split on Gender:

1. First we are populating for node Female, Populate the actual value for "Play Cricket" and "Not Play Cricket", here these are 2 and 8 respectively.

2. Calculate expected value for "Play Cricket" and "Not Play Cricket", here it would be 5 for both because parent node has probability of 50% and we have applied same probability on Female count(10).

3. Calculate deviations by using formula, Actual — Expected. It is for "Play Cricket" (2–5 = -3) and for "Not play cricket" (8–5 = 3).

5. Calculate Chi-square of node for “Play Cricket” and “Not Play Cricket” using formula with formula, = ((Actual — Expected)^2 / Expected)^{1}/2. You can refer below table for calculation.

6. Follow similar steps for calculating Chi-square value for Male node.

7. Now add all Chi-square values to calculate Chi-square for split Gender.

Node	Play Cricket	Not Play Cricket	Total	Expected Play Cricket	Expected Not Play Cricket	Deviation Play Cricket	Deviation Not Play Cricket	Chi-Square	
								Play Cricket	Not Play Cricket
Female	2	8	10	5	5	-3	3	1.34	1.34
Male	13	7	20	10	10	3	-3	0.95	0.95
							Total Chi-Square	4.58	

Split on Class:

Perform similar steps of calculation for split on Class and you will come up with below table.

Node	Play Cricket	Not Play Cricket	Total	Expected Play Cricket	Expected Not Play Cricket	Deviation Play Cricket	Deviation Not Play Cricket	Chi-Square	
								Play Cricket	Not Play Cricket
IX	6	8	14	7	7	-1	1	0.38	0.38
X	9	7	16	8	8	1	-1	0.35	0.35
							Total Chi-Square	1.46	

Now, We can see that Chi-square also identify the Gender split is more significant compare to Class.

Reduction in Variance

1. Reduction in variance is an algorithm used for continuous target variables (regression problems).

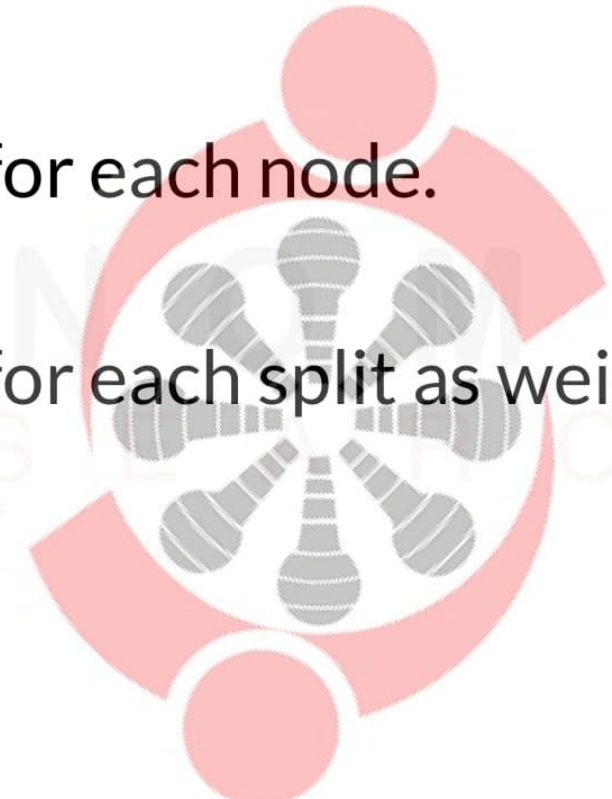
1. This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

    - ***Above X-bar is mean of the values, X is actual and n is number of values.***

Steps to calculate Variance

1. Calculate variance for each node.

1. Calculate variance for each split as weighted average of each node variance.



Example

Example:- Let's assign numerical value 1 for play cricket and 0 for not playing cricket. Now follow the steps to identify the right split:

1. Variance for Root node, here mean value is (15*1 + 15*0)/30 = 0.5 and we have 15 one and 15 zero. Now variance would be $((1-0.5)^2+(1-0.5)^2+....15 \text{ times}+(0-0.5)^2+(0-0.5)^2+...15 \text{ times}) / 30$, this can be written as $(15*(1-0.5)^2+15*(0-0.5)^2) / 30 = 0.25$

1. Mean of Female node = $(2*1+8*0)/10=0.2$ and Variance = $(2*(1-0.2)^2+8*(0-0.2)^2) / 10 = 0.16$

1. Mean of Male Node = $(13*1+7*0)/20=0.65$ and Variance = $(13*(1-0.65)^2+7*(0-0.65)^2) / 20 = 0.23$

Example

4. Variance for Split Gender = Weighted Variance of Sub-nodes = (10/30)*0.16 + (20/30) *0.23 = 0.21

5. Mean of Class IX node = (6*1+8*0)/14=0.43 and Variance = (6*(1–0.43)²+8*(0–0.43)²) / 14= 0.24

6. Mean of Class X node = (9*1+7*0)/16=0.56 and Variance = (9*(1–0.56)²+7*(0–0.56)²) / 16 = 0.25

7. Variance for Split Gender = (14/30)*0.24 + (16/30) *0.25 = 0.25

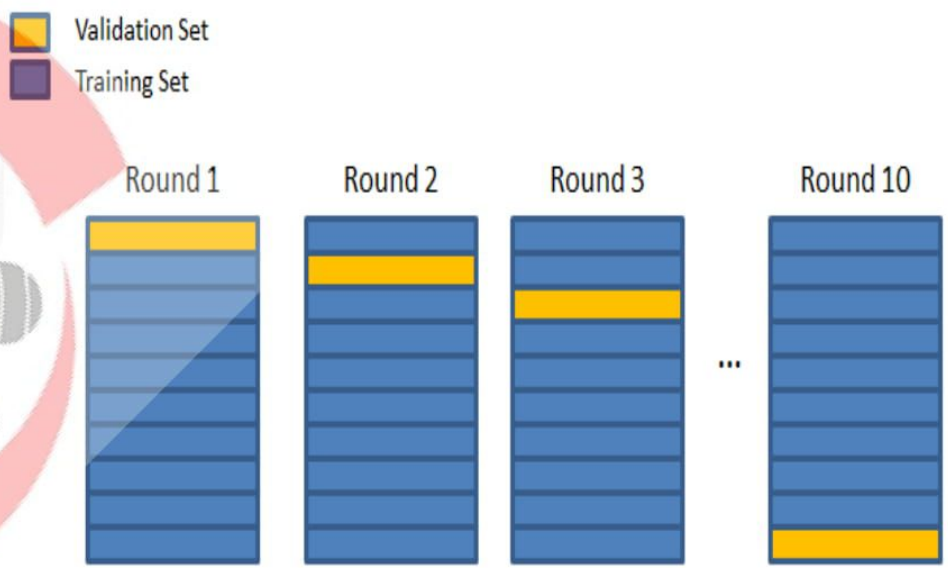
We can see that Gender split has lower variance compare to parent node, so the split would take place on Gender variable.

Tackle Problem of Overfitting

Through Cross Validation Overfitting Problem can be solved:

In general, we don't know how well our model will perform on new data until we actually test it.

To address this, we can split our initial dataset into separate training and test subsets.



- To partition the data into a number of subsets.
- Hold out a set at a time and train the model on remaining set.
- Test model on hold out set.

Repeat the process for each subset of the dataset.

INNOMATICS
RESEARCH LABS

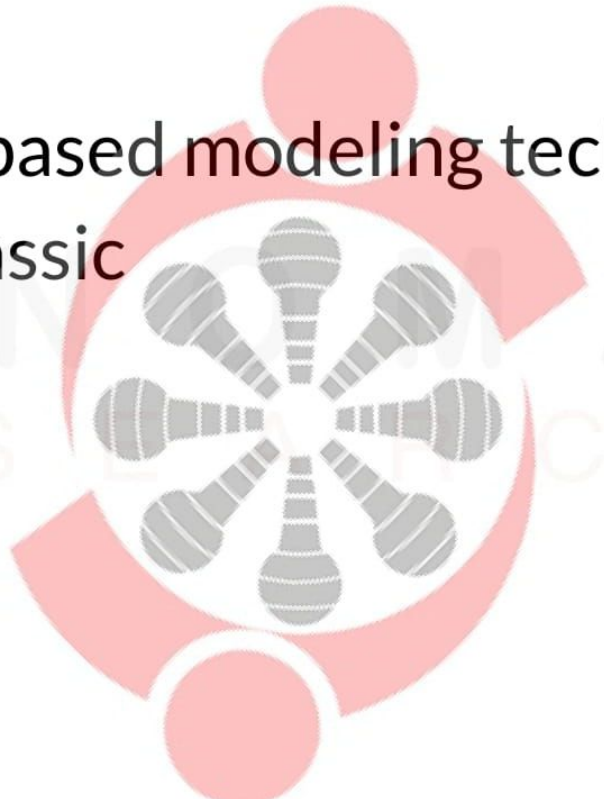
CART

- Classification And Regression Trees

- Developed by Breiman, Friedman, Olshen, Stone in early 80's.
  - Introduces tree-based modeling into the statistical
  - Mainstream.
  - Rigorous approach involving cross-validation to select the optimal tree

CART

- One of many tree-based modeling techniques:
  - CART → the classic
  - CHAID
  - C5.0



Impurity Measure of CART: GINI

Gini Index for a given node t:

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

(NOTE: p(j|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 belongs 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	

Example for Computing: GINI

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

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1
Gini = 1 - $P(C2)^2$ - $P(C2)^2$ = 1-0-1=0

C1	1
C2	5

P(C1) = ⅙ P(C2) = ⅚
Gini = 1 - $(⅙)^2$ - $(⅚)^2$ = 0.278

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6
Gini = 1 - $(2/6)^2$ - $(4'6)^2$ = 0.444

Splitting Based on GINI

- When a node p is split into k partitions (children), the quality of split is computed as;

$$GINI_{split} = \sum_{i-j}^{k} n_i/n \; GINI(i)$$

Where, n_i = number of records at child i,

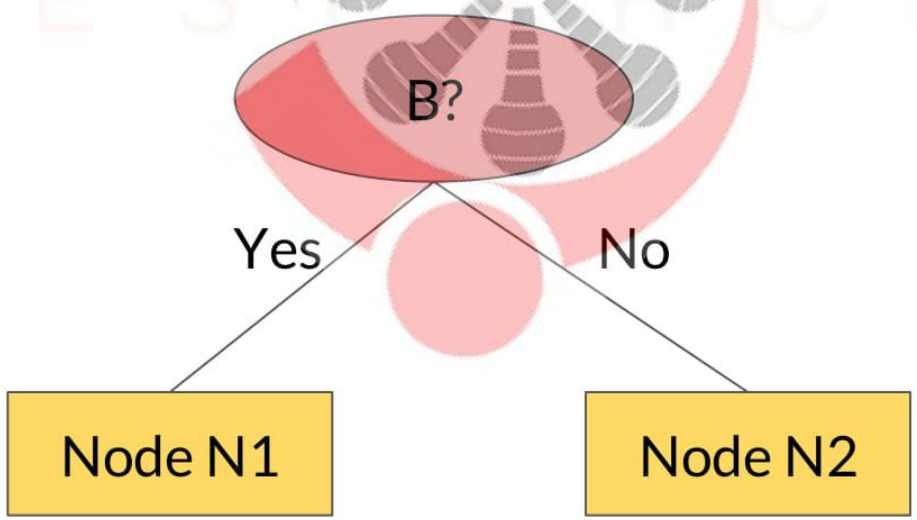
n = number of records at node p.

The "twoing" rule strikes a balance between purity and creating roughly equal sized nodes

NOTE: "towing" is available in Salford Systems 'CART' but not in the "rpart" package in R.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effects of Weighting partitions;
  - Larger and Purer Partitions are sought for.



	Parent
C1	6
C2	6
Gini=0.500	

Binary Attributes: Computing GINI Index

- Gini(N1)

$= 1-(5/7)^2 - (2/7)^2$

$=0.4082$

- Gini(N2)

$= 1-(1/5)^2 - (4/5)^2$

$= 0.32$

	N1	N2
C1	5	1
C2	2	4
	Gini=0.371	

- Gini(Children)

$= 7/12*0.408+5/12*0.32$

$= 0.371$

Gini Index

Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

1. It works with categorical target variable “Success” or “Failure”.
2. It performs only Binary splits
3. Higher the value of Gini higher the homogeneity.
4. CART (Classification and Regression Tree) uses Gini method to create binary splits.

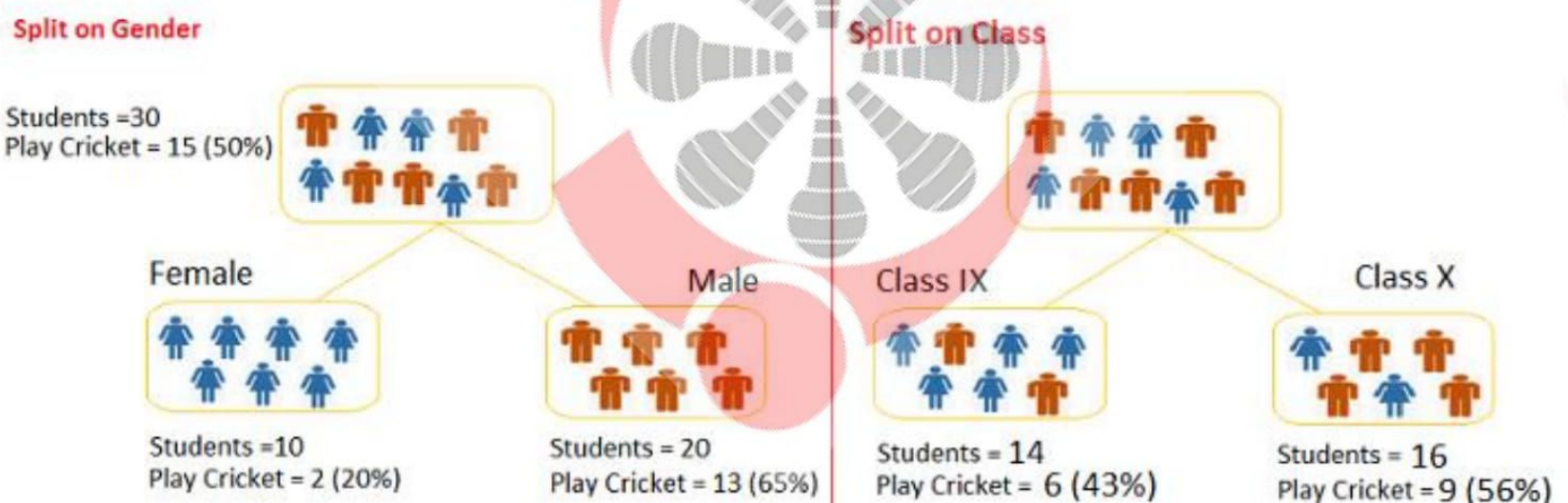
Steps to calculate Gini for a split

1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).

1. Calculate Gini for split using weighted Gini score of each node of that split.

Steps to calculate Gini for a split

Example: — Referring to example where we want to segregate the students based on target variable (playing cricket or not). In the snapshot below, we split the population using two input variables Gender and Class. Now, Identify which split is producing more homogeneous sub-nodes using Gini index.



Split on Gender:

1. Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68

2. Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55

3. Weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = 0.59

Similar for Split on Class:

1. Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51

1. Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51

2. Weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = 0.51

We can see Gini score for Split on Gender is higher than Split on Class, hence, the node split will take place on Gender.

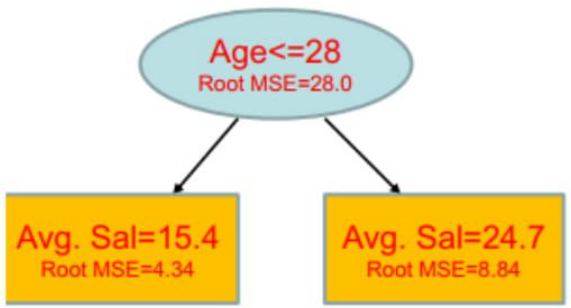
Regression Tree:

- Tree-based modeling for ***continuous* target variable**
- Most intuitively appropriate method for loss ratio analysis
- Find split that produces greatest separation in

$$\sum[y - E(y)]^2$$

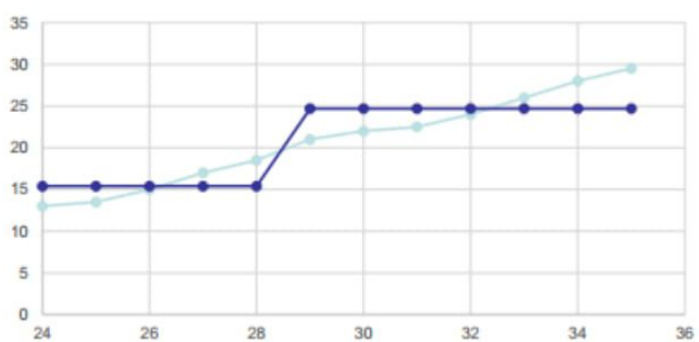
- Find nodes with minimal *within variance*
- Therefore greatest *between variance*
- Every record in a node is assigned the same y
- Model is a *step function*

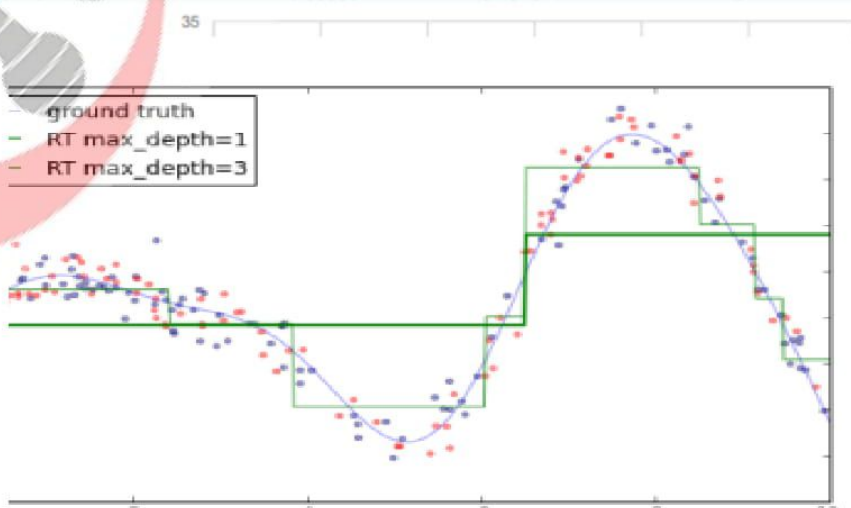
Example:



Age	Avg. Sal (in Ks)
24	13
25	13.5
26	15
27	17
28	18.5
29	21
30	22
31	22.5
32	24
33	26
34	28

Split Cond	Root Node MSE	L MSE	R MSE	Gain
Age<=24	28.05556	0	24.52066	3.534894
Age<=25	28.05556	0.0625	19.8525	8.140556
Age<=26	28.05556	0.722222	15.38889	11.94444
Age<=27	28.05556	2.421875	11.96484	13.66884
Age<=28	**28.05556**	**4.34**	**8.846939**	**14.86862**
Age<=29	28.05556	7.972222	7.638889	12.44444
Age<=30	28.05556	10.76531	6.5	10.79025
Age<=31	28.05556	12.55859	4.296875	11.20009
Age<=32	28.05556	14.94444	2.055556	11.05556
Age<=33	28.05556	18.5125	0.5625	8.980556
Age<=34	28.05556	23.15702	0	4.898531





Classification Tree

- Splitting Criteria:
  – Gini, Entropy, Twoing
- • Goodness of fit measure:
  – misclassification rates
- Prior probabilities and misclassification costs
  – available as model "tuning parameters"

Regression Tree

- Splitting Criterion:
  – sum of squared errors
- Goodness of fit:
  – same measure!
  – sum of squared errors
- No priors or misclassification costs…
  – … just let it run

Reference

References

• Mitchell, Tom M. *Machine Learning*. McGraw-Hill, 1997.
pp. 55–58.

• https://www.geeksforgeeks.org/decision-tree/

• https://artint.info/html/ArtInt_177.html

• https://www.youtube.com/watch?v=DCZ3tsQIoGU

• Jiawei Han And Micheline Kamber, *Data Mining Concept and Techniques*

• Andrew Moore, *Decision Trees Tutorial*, Auton Lab, http://www.autonlab.org/tutorials/dtree.html

• T. Hastie, R. Tibshirani and J. Friedman. *Elements of Statistical Learning*, Springer, 2009.

• L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

• J.R. Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.