VIT-AP UNIVERSITY

Decision Tree in Machine Learning

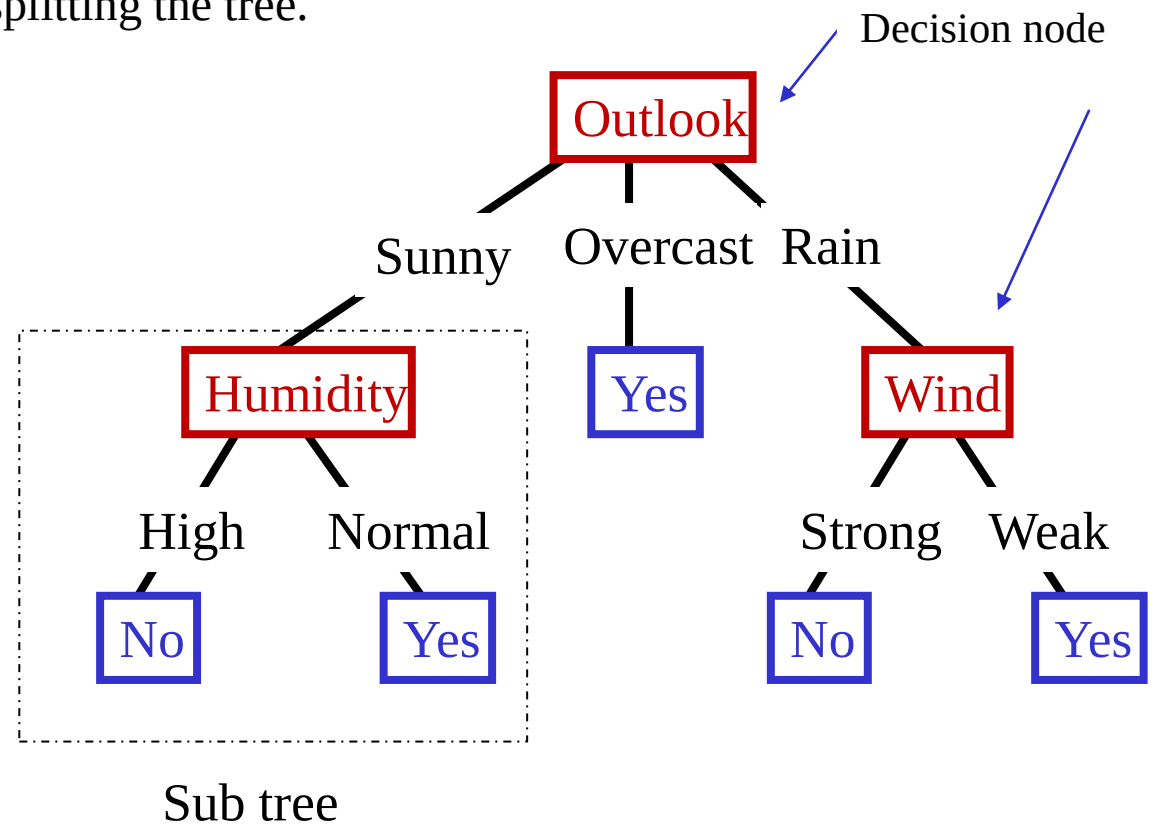
Dr. Kuppusamy .P
Associate Professor / SCOPE

Decision Tree Learning – Supervised Learning

- Decision tree is a **tree** structure that **makes decision (**classification or regression) by learning **knowledge**.
- It is a graphical representation for getting all the possible solutions/decisions based on given conditions.
- At the end of the learning process, the decisions or test are performed based on features of the given dataset.
  - The decision tree can be thought of as a set sentences (in Disjunctive Normal Form) written propositional logic.
  - It deals both categorical and numerical data.
  - Some characteristics of problems in Decision Tree Learning:
    - Attribute-value paired elements
    - Discrete target function
    - Disjunctive descriptions (of target function)
    - Works well with missing or erroneous training data

Terminologies in Decision Tree

- **Root node** initiates the decision tree that represents the entire dataset D. Then D is divided into two or more homogeneous sets.
- **Internal** (Decision) nodes represent the features of a dataset that make the decision, branches represent the decision rules to make decision, and each leaf node represents the decision (outcome).
- **Splitting** process divides the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree** is formed by splitting the tree.
- **Pruning** process removes the unwanted branches from the tree.



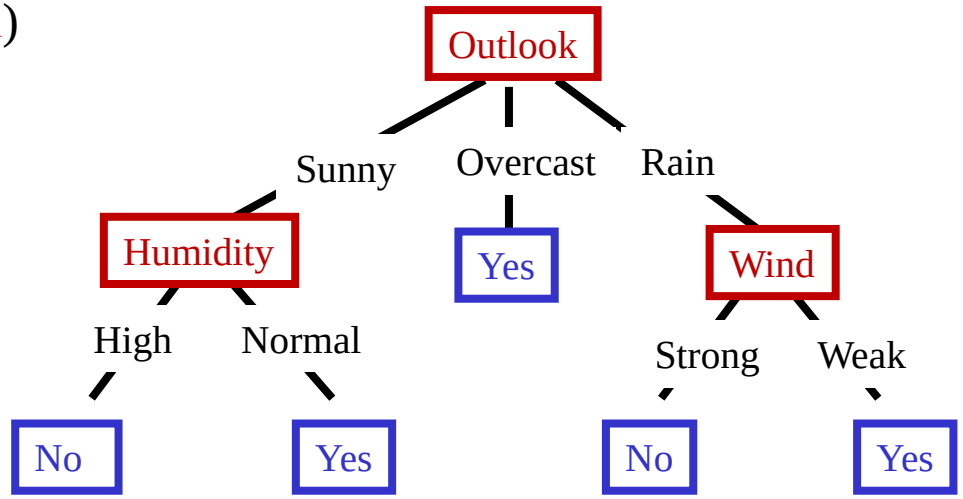
Training Examples

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

Decision Tree Representation

- Decision trees represent a **disjunction of conjunctions of constraints** on the attribute values of instances.
- Each path from the root to a leaf corresponds to a conjunction of attribute tests, and
- The tree itself is a disjunction of these conjunctions.
- Each node represents a feature, and each link represents a decision.
- Each leaf node represents an outcome (classification)

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



Building a Decision Tree

1. First test all attributes and select the **one attribute** that would function as the **best** root.
2. Break-up the training set into **subsets** based on the branches of the root node.
3. Test the **remaining attributes** to check which one fit best underneath the **branches** of the root node;
4. Continue this process for all other branches until
   a. all examples of a subset are of one type
   b. there are no examples left (return majority classification of the parent)
   c. there are **no more** attributes left (default value should be majority classification)

When to Consider Decision Trees?

- Instances are represented by attribute-value pairs.
  - Fixed set of attributes, and the attributes take a small number of disjoint possible values.
- The target function has discrete output values.
  - Decision tree learning is appropriate for a boolean classification, but it easily extends to learning functions with more than two possible output values.
- Disjunctive descriptions may be required.
  - decision trees naturally represent disjunctive expressions.
- The training data may contain errors.
  - Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
- The training data may contain missing attribute values.
  - Decision tree methods can be used even when some training examples have unknown values.
- Decision tree learning has been applied to problems such as learning to classify
  - medical patients by their disease,
  - equipment malfunctions by their cause, and
  - loan applicants by their likelihood of defaulting on payments.

Attribute (Feature) Selection Measures

1. Information Gain
2. Gini Index

Information gain:

- ***Information gain*** measures how well a given attribute separates the training examples according to their target classification.
- Information Gain refers to the **decline** (changes) in entropy after the dataset is split (**Entropy Reduction)**.
- It calculates how much information gained from a feature about a class.
- Split the node based on information gain value and build the decision tree.
- Decision tree algorithm always attempts to maximize the information gain value.
- Node/attribute contains the highest information gain is split first.

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Gain(S,A) = **Entropy(S)- `I(attribute)`**

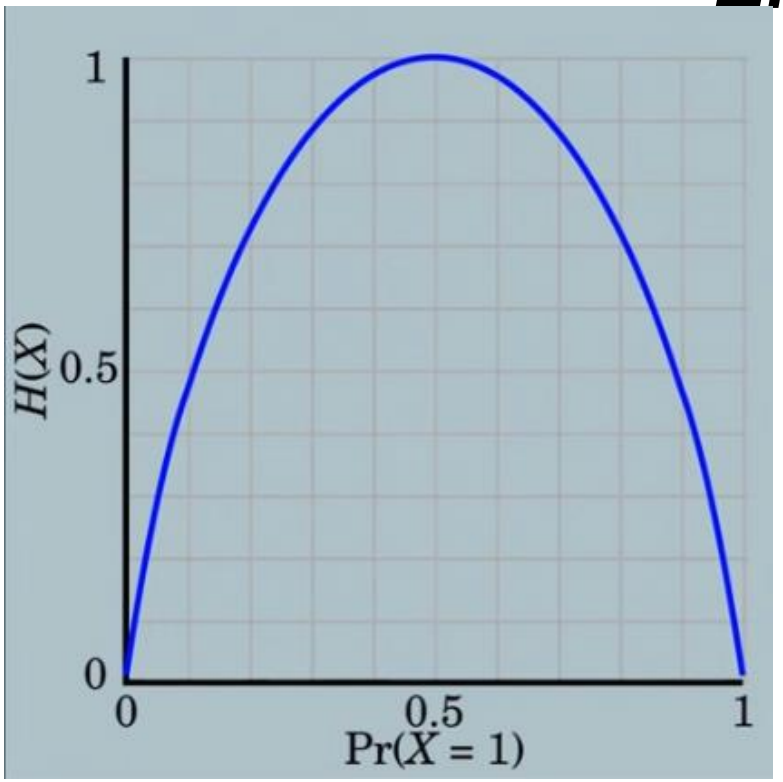
Entropy

- ***Entropy*** characterizes (measures) the impurity in a given attribute of arbitrary training examples.
- It specifies degree of randomness (uncertainty) in data.
- Entropy values used in splitting i.e., which node is to be split first.
- Given a collection S, containing positive and negative examples of some target concept, the ***entropy of S*** relative to this Boolean classification is:

$$\textbf{Entropy(S)} = -\frac{p}{(p+q)} * \log_2\left(\frac{p}{(p+q)}\right) - \frac{q}{(p+q)} * \log_2\left(\frac{q}{(p+q)}\right)$$

- **S** is a total number of training samples
- **p** is the proportion of positive classes
- **q** is the proportion of negative classes.

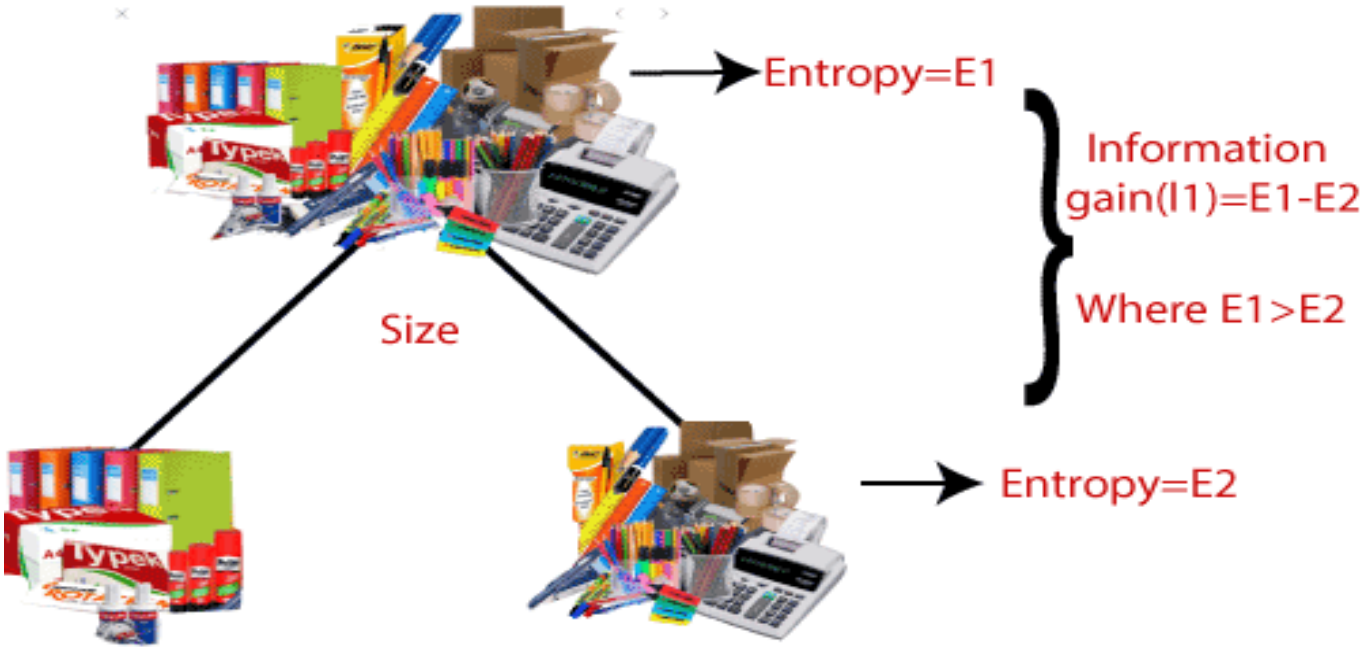
Entropy



- If data is completely (highly) pure / (highly) impure, randomness is 0.
- If impurity is 0.5, randomness (entropy) is 1.

Decision tree - ID3 algorithm

- **Iterative Dichotomiser 3 (**ID3) algorithm uses this *information gain* measure to select among the candidate attributes at each step that return the highest data gain while growing the tree.



Training Examples

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

- 9 positive instances and 5 negative instance

Examples: Entropy calculation

- **Entropy(S) = -p/(p+q)*log2(p/(p+q)) - q/(p+q)*log2 (q/(p+q))**
- 9 positive instances and 5 negative instances.

- Entropy value range is from 0 to 1.
- Leaf nodes with greater entropy value is considered for further splitting.
- Entropy(S)= Entropy([9+,5-]) = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$

  (94% impure or non-homogeneous)
- In given dataset, If **50% is positive** and **50% is negative** after the splitting, the entropy value is 1 (worst case).

  Entropy([8+,8-] = $-(8/16)\log_2(8/16) - (8/16)\log_2(8/16) = 1.0$

Examples: Entropy calculation

- In a given dataset, **All examples are positive.**

  Entropy([8+,0-] = $-(8/8)\log_2(8/8) - (0/8)\log_2(0/8) = 0.0$

- In a given dataset, **All examples are Negative.**

  Entropy([0+,8-] = $-(0/8)\log_2(0/8) - (8/8)\log_2(8/8) = 0.0$

Calculate Average Information Entropy of Attribute:

I(attribute) = $(p_i + q_i)/(p+q)$ Entropy(A)

- p_i, q_i - +ve, -ve values of corresponding attribute (A) possibility value
- p, q – total +ve, -ve values of dataset

ID3 - Algorithm

ID3(*Examples, TargetAttribute, Attributes*)

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = -
- If *Attributes* is empty, Return the single-node tree Root, with label = most common value of *TargetAttribute* in *Examples*
- Otherwise Begin
  - A = The attribute from list of *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* ←A
  - For each possible value, v_i, of attribute A
    - Add a new tree branch below *Root* corresponding to the test A = v_i
    - Let $Examples_{vi}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{vi}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *TargetAttribute* in *Examples*
      - else below this new branch add the subtree

        ID3($Examples_{vi}$, *TargetAttribute, Attributes* – {A})
- end
- return *Root*

ID3 - Training Examples

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

- 9 positive instances and 5 negative instance

Calculate the Entropy for the entire data set

Create a *Root* node for the tree

- Entropy([9+,5-] = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$

- Select root node from 4 features outlook, temperature, humidity and windy.

Select best decision attribute:

- Let first select feature "Outlook". Outlook contains three possibilities such as sunny, rainy, overcast.
- Hence, calculate entropy for each possibility value of outlook.

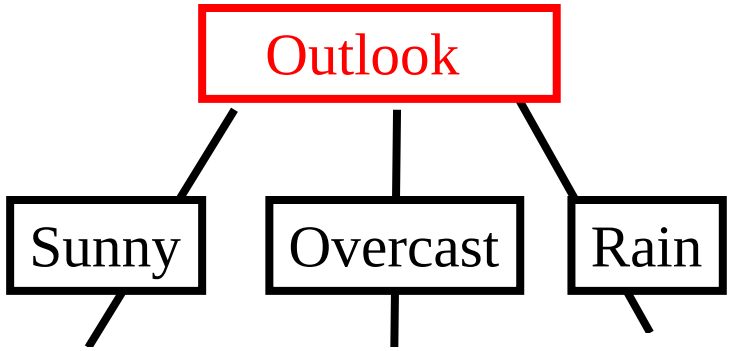
Calculate the Entropy for Attribute

Outlook	PlayTennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

Outlook	PlayTennis
Rainy	Yes
Rainy	Yes
Rainy	No
Rainy	Yes
Rainy	No

Outlook	PlayTennis
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

S=[9+,5-]
E=0.940



[2+, 3-] [4+, 0] [3+, 2-]

E=0.971 E=0.0 E=0.971

Entropy of Outlook = Sunny

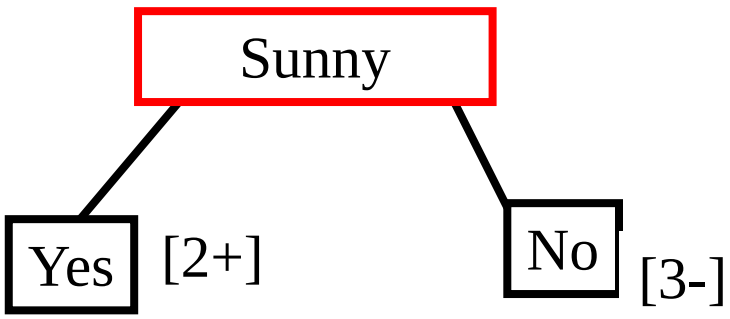
Entropy of Outlook = Overcast

Entropy of Outlook = Rain

Gain(S,Outlook) =
0.940- [(5/14)*0.971 +(4/14)*0.0 +(5/14)*0.971]
= **0.247**

Calculate the Entropy for Attribute

S=[2+,3-]
E=0.971

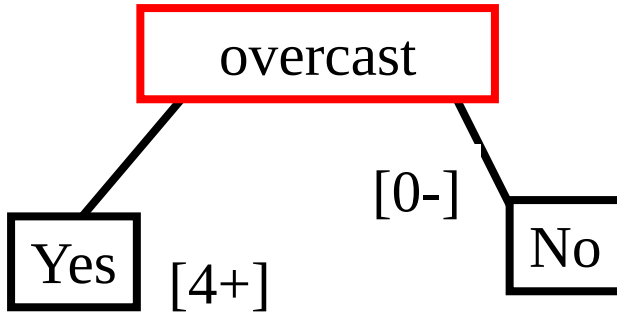


Outlook	PlayTennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

- Entropy([outlook = sunny] = $-(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.971$

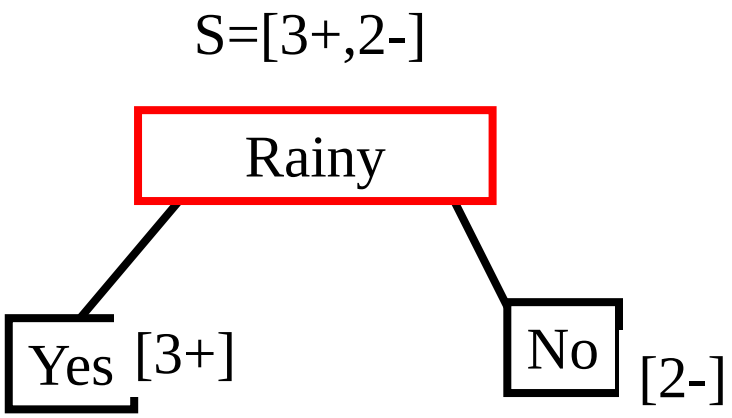
Outlook	PlayTennis
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

S=[4+,0-]



- Entropy([outlook = overcast] = $-(4/4)\log_2(4/4) - (0/4)\log_2(0/4) = 0$

Calculate the Entropy for Attribute



Outlook	PlayTennis
Rainy	Yes
Rainy	Yes
Rainy	No
Rainy	Yes
Rainy	No

- Entropy([outlook = sunny] = $-(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.971$

Calculate Average Information Entropy

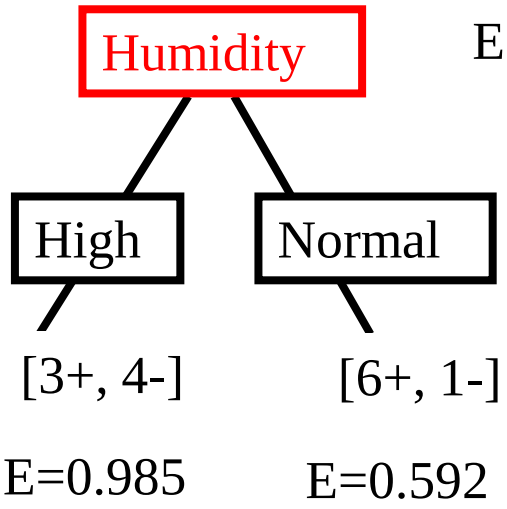
- $I(outlook) = (P_{sunny} + N_{sunny})/(p+n)$ (Entropy (outlook = sunny)) +
  $(P_{overcast} + N_{overcast})/(p+n)$ (Entropy (outlook = overcast))+
  $(P_{rainy} + N_{rainy})/(p+n)$ (Entropy (outlook = rainy))

- I(outlook) = (2+3)/(9+5) * 0.971 + (4+0)/((9+5) *0 +
  (3+2)/(9+5) * 0.971 = **0.693**
- **Info Gain(**S,Outlook) = Entropy (S) – I(Outlook)
- **Info Gain(**S,Outlook) = 0.940 - 0.693
  = **0.247**

Entropy for Humidity

$$E(S) = E([9+,5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

S=[9+,5-]
E=0.940



$$E(\text{Humidity=}\mathbf{High}) = -(3/7)\log_2(3/7) - (4/7)\log_2(4/7) = 0.985$$

$$E(\text{Humidity=}\mathbf{Normal}) = -(6/7)\log_2(6/7) - (1/7)\log_2(1/7) = 0.592$$

Info. Gain(S, **Humidity)** = E(S) – I(Humidity)
= 0.940- [(7/14)*0.985+(7/14)*0.592]
= **0.151**

Entropy for wind

$E(S) = E([9+,5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$

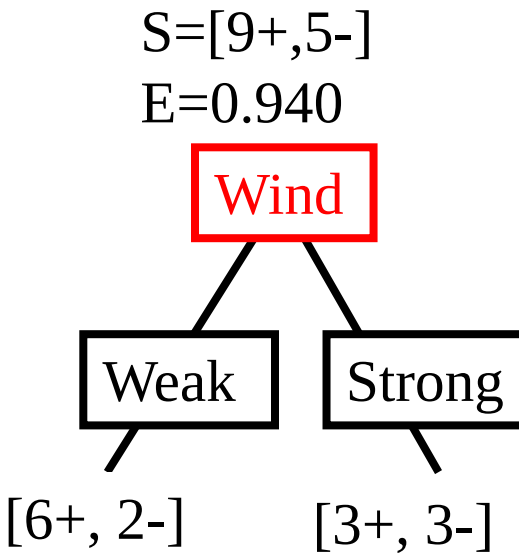
$E(\text{Wind}=\textbf{Weak}) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8) = 0.811$

$E(\text{Wind}=\textbf{Strong}) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$

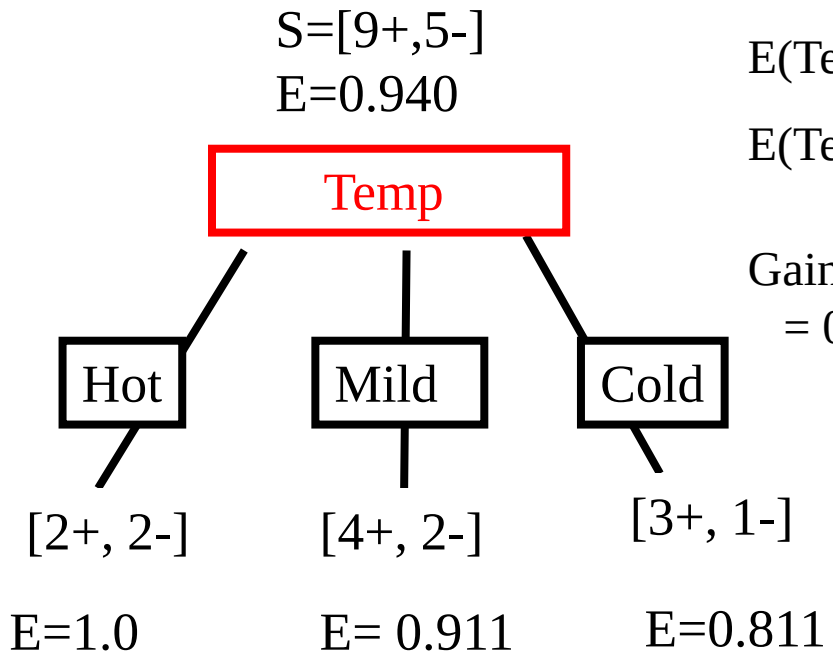
$\text{Gain}(S,\text{Wind}) = E(S) - I(\text{Wind})$

$= 0.940 - [(8/14)*0.811 + (6/14)*1.0]$

$= \mathbf{0.048}$



Entropy for Temperature



E(Temp=**Hot**) = – (2/4) $\log_2$(2/4) – (2/4) $\log_2$(2/4) = 1

E(Temp=**Mild**) = – (4/6) $\log_2$(4/6) – (4/6) $\log_2$(4/6) = 0.911

E(Temp=**Cold**) = – (3/4) $\log_2$(3/4) – (1/4) $\log_2$(1/4) = 0.811

Gain(S,Temperature) = E(S) – I (Temp)
= 0.940- [(4/14)*1.0 + (6/14)*0.911+ (4/14)*0.811]
= **0.029**

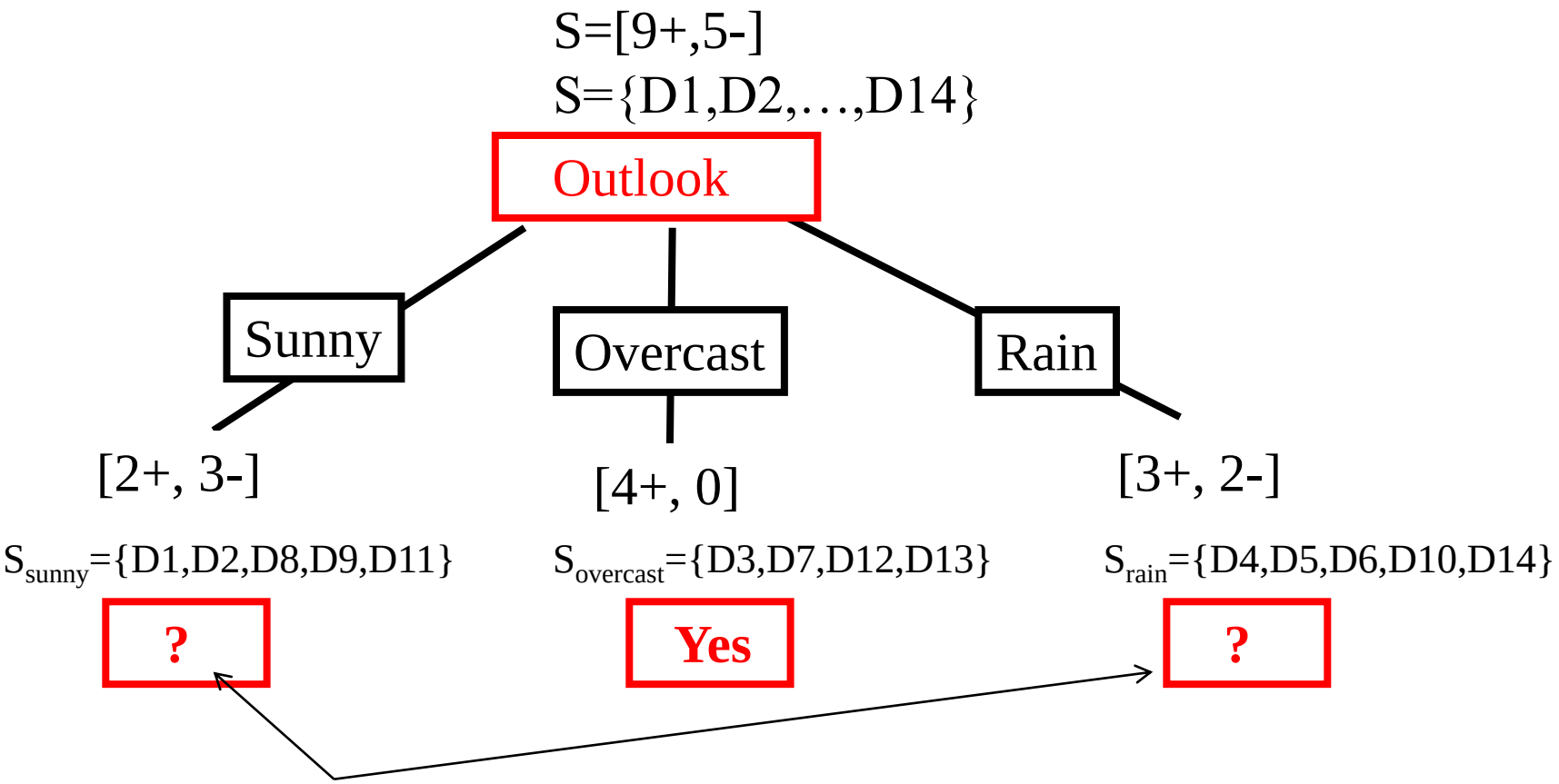
Information Gain of 4 Attributes:

- Info Gain(S,Outlook) = 0.247
- Info. Gain(S, Humidity) = 0.151
- Info. Gain(S,Wind) = 0.048
- Info. Gain(S,Temperature) = 0.029

Select the attribute which contains Max Information Gain i.e. Info Gain(S,**Outlook**) = 0.247

Best Attribute - Outlook

Algorithm:
1. First test all attributes and select the **one attribute** that would function as the **best** root.
2. Break-up the training set into **subsets** based on the branches of the root node.

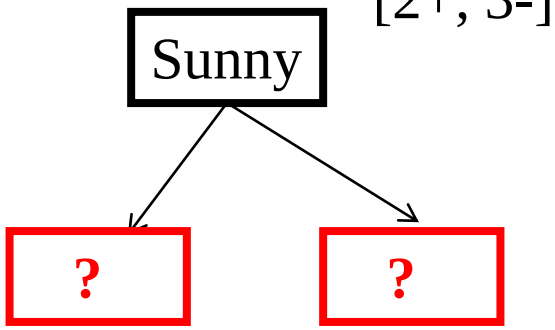


Repeat the same process for the sub-trees till get the tree

Outlook	Temp	Humidity	Windy	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

S_{sunny}={D1,D2,D8,D9,D11}

[2+, 3-]



If Outlook = **Sunny**

p= 2 and n=3

Entropy(Sunny) = **-**$2/(2+3)\log_2(2/(2+3)) - 3/(2+3)\log_2(3/(2+3))$

= 0.970

Calculate the entropy value for Humidity

Outlook	Humidity	Play Tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

Humidity	p	n	Entropy
High	0	3	0
Normal	2	0	0

E(Humidity=**High**) = $-(0/3)\log_2(0/3) - (3/3)\log_2(3/3) = 0$

E(Humidity=**Normal**) = $-(2/2)\log_2(2/2) - (0/2)\log_2(0/2) = 0$

Average Info. Entropy (Humidity) = (3/5) * 0.0 + 2/5 *(0.0) = 0

Gain(S_{sunny} , Humidity) = 0.970 - 0 = **0.97**

Calculate the entropy value for each Windy

Outlook	Windy	Play Tennis
Sunny	Weak	No
Sunny	Strong	No
Sunny	Weak	No
Sunny	Weak	Yes
Sunny	Strong	Yes

Windy	p	n	Entropy
Strong	**1**	**1**	**1**
Weak	**1**	**2**	**0.8962**

E(Windy=**Strong**) = – (1/2) $\log_2$(1/2) – (1/2) $\log_2$(1/2) = 1

E(Windy=**Weak**) = – (1/3) $\log_2$(1/3) – (2/3) $\log_2$(2/3) = - 0.3 (-1.585) – 0.6 * (-0.5851)

= 0.4755 + 0.3516 = 0.8271

Average Info. Entropy (Windy) = (2/5)1.0 + 3/5(0.8271) = 0.4+0.4962

= 0.8962

Gain(S_{sunny} , Windy) = 0.970 - 0.8962 = **0.0738**

Calculate the entropy value for Temperature

Outlook	Temp	Play Tennis
Sunny	Hot	No
Sunny	Hot	No
Sunny	Mild	No
Sunny	Cool	Yes
Sunny	Mild	Yes

Temp	p	n	Entropy
Hot	**0**	**2**	**0**
Mild	**1**	**1**	**1**
Cool	**1**	**0**	**0**

Average Info. Entropy (Temp) = (2/5)0.0 +2/5(1.0) + (1/5)0.0 = 0.4

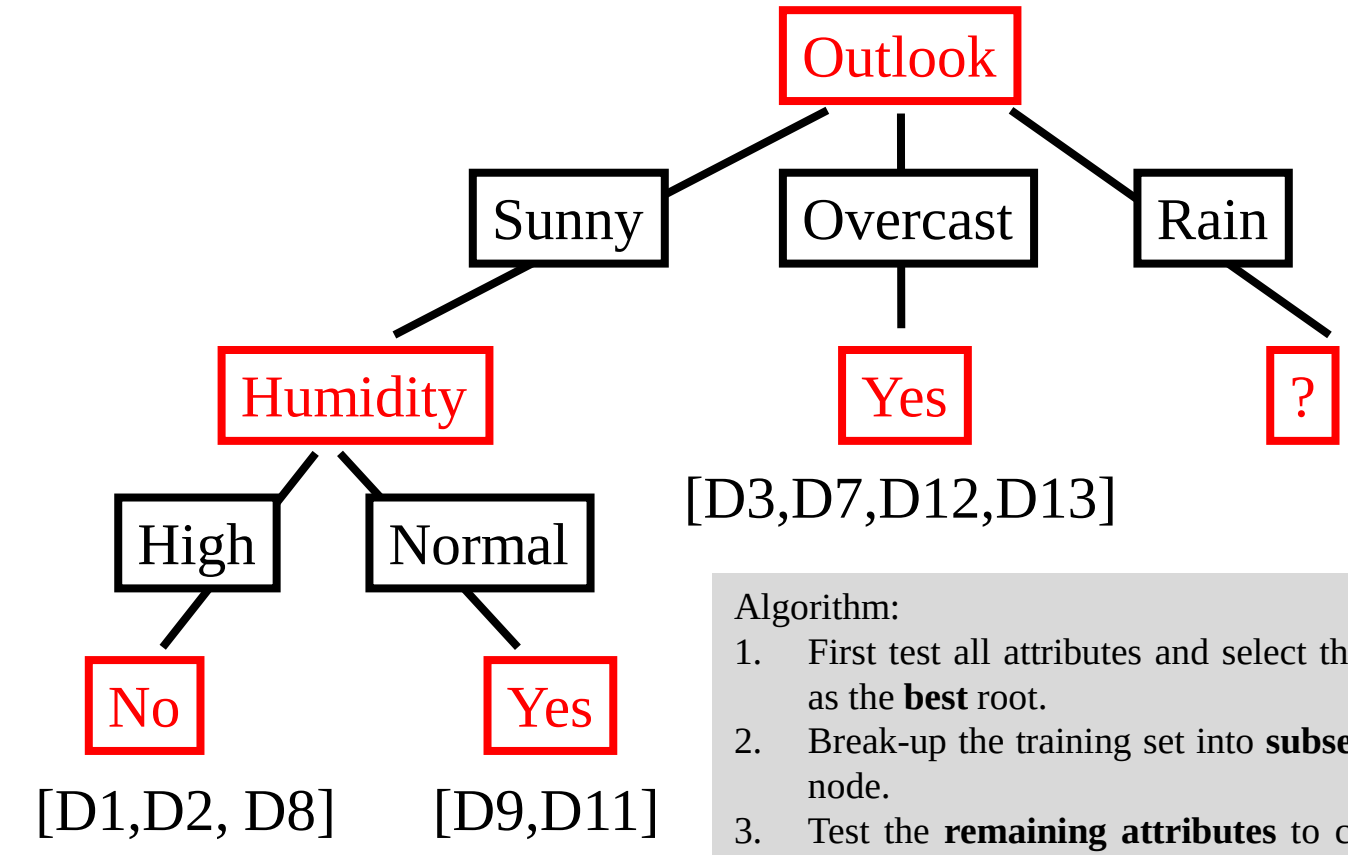
Gain(S_{sunny} , Temp) = 0.970 - 0.4 = **0.571**

Best Attribute:

- Gain(S_{sunny} , Humidity) = 0.970 - 0 = **0.970**
- Gain(S_{sunny} , Wind) = 0.970 - 0.951 = **0.0738**
- Gain(S_{sunny} , Temp) = 0.970 - 0.4 = **0.571**

So, **Humidity** is selected as best attribute.

ID3 - Result



Algorithm:

1. First test all attributes and select the **one attribute** that would function as the **best** root.
2. Break-up the training set into **subsets** based on the branches of the root node.
3. Test the **remaining attributes** to check which one fit best underneath the **branches** of the root node;
4. Continue this process for all other branches until
   a. all examples of a subset are of one type
   b. there are no examples left (return majority classification of the parent)
   c. there are **no more** attributes left (default value should be majority classification)

Repeat the same process for the sub-trees till get the tree

Outlook	Temp	Humidity	Windy	Play Tennis
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Rain	Mild	Normal	Weak	Yes
Rain	Mild	High	Strong	No

If Outlook = **Rain**

p= 3 and n=2

Entropy(Rain) = $-3/(3+2)\log_2(3/(3+2)) - 2/(3+2)\log_2(2/(3+2))$

= - 0.6 * (-0.737) – 0.4 * (-1.322)

= 0.4422 + 0.5288 = 0.971

Calculate the entropy value for Humidity

Outlook	Humidity	Play Tennis
Rain	High	Yes
Rain	High	No
Rain	Normal	Yes
Rain	Normal	No
Rain	Normal	Yes

Humidity	p	n	Entropy
High	1	1	1
Normal	2	1	0.8271

E(Humidity=**High**) = – (1/2) $\log_2(1/2)$ – (1/2) $\log_2(1/2)$ = 1

E(Humidity=**Normal**) = – (2/3) $\log_2(2/3)$ – (1/3) $\log_2(1/3)$ = – 0.6 * (-0.5851) - 0.3 (-1.585)

= 0.3516 + 0.4755 = 0.8271

Average Info. Entropy (Humidity) = 2/5(1) + (3/5) * 0.8271 = 0.4 + 0.6 (0.8271)

= 0.4+0.496 = 0.896

Gain(S_{rain} , Humidity) = Entropy(S_{rain}) – ($I_{humidity}$)

= 0.971- 0.896= 0.075

Calculate the entropy value for Temperature

Outlook	Temp	Play Tennis
Rain	Mild	Yes
Rain	Cool	Yes
Rain	Cool	No
Rain	Mild	Yes
Rain	Mild	No

Temp	p	n	Entropy
Hot	**0**	**0**	**0**
Mild	**2**	**1**	**0.8271**
Cool	**1**	**1**	**1**

E(Temp=**Hot**) = 0

E(Temp=Mild) = $-(2/3)\log_2(2/3) - (1/3)\log_2(1/3) = -0.6 * (-0.5851) - 0.3 (-1.585)$

$= 0.3516 + 0.4755 = 0.8271$

E(Temp=**Cool**) = $-(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$

Average Info. Entropy (Temp) = (0/5) * 0 + (3/5) * 0.8271 +(2/5) * 1

=0+ 0.496+0.4 = 0.896

Gain(S_{rain} , Humidity) = Entropy(S_{rain}) – ($I_{humidity}$)

= 0.971- 0.896= 0.075

Calculate the entropy value for Windy

Outlook	Windy	Play Tennis
Rain	Weak	Yes
Rain	Weak	Yes
Rain	Strong	No
Rain	Weak	Yes
Rain	Strong	No

Windy	p	n	Entropy
Strong	**0**	**2**	**0**
Weak	**3**	**0**	**0**

$$Gain(S_{rain}, Wind) = Entropy(S_{rain}) - (I_{windy})$$

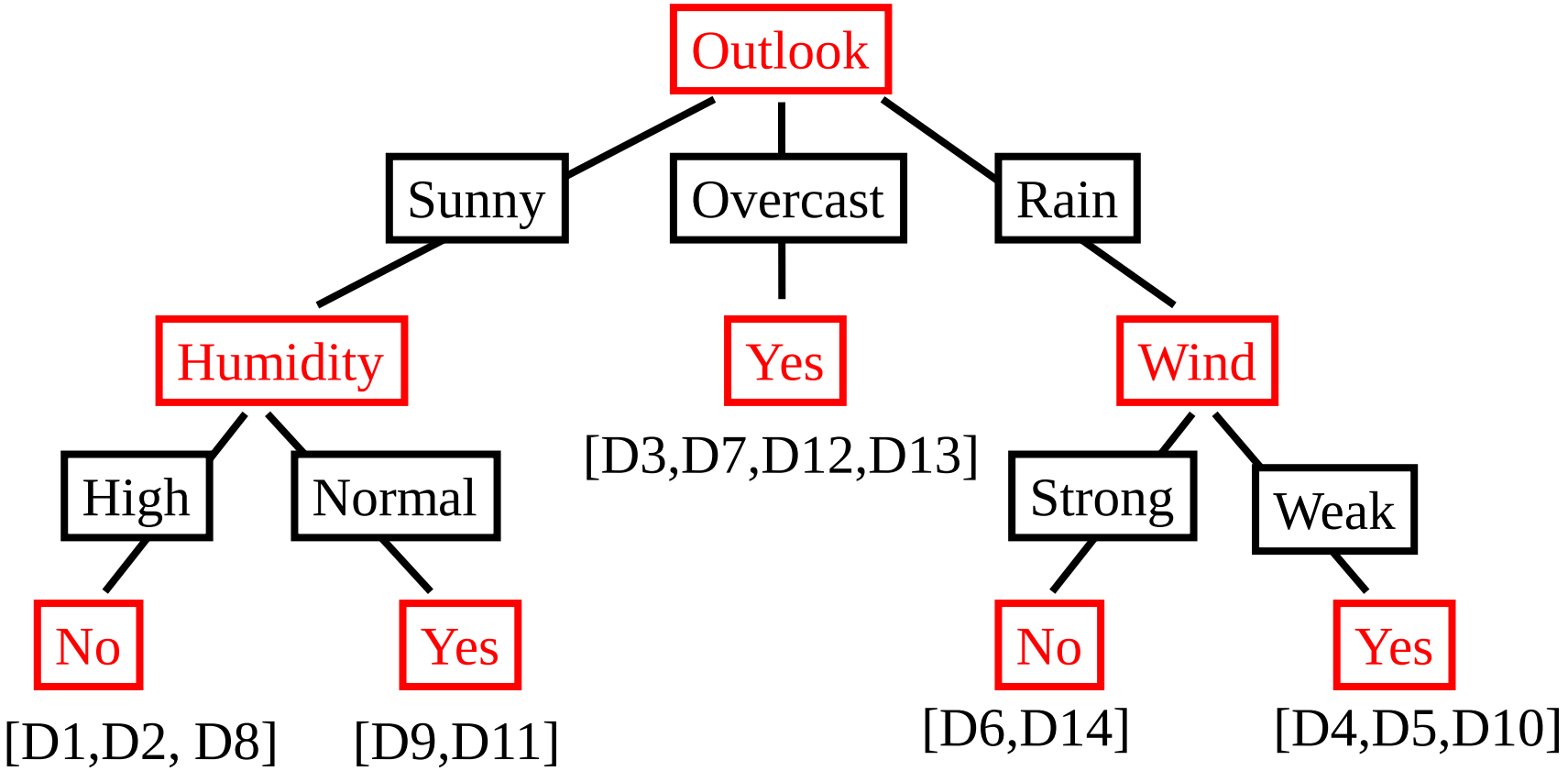
$$= 0.971 - 0 = \mathbf{0.971}$$

Best Attribute

- $Gain(S_{rain}, Humidity) = 0.075$
- $Gain(S_{rain}, Temp.) = 0.075$
- $Gain(S_{rain}, Wind) = \mathbf{0.971}$

So, **Wind** will be selected

ID3 - Result

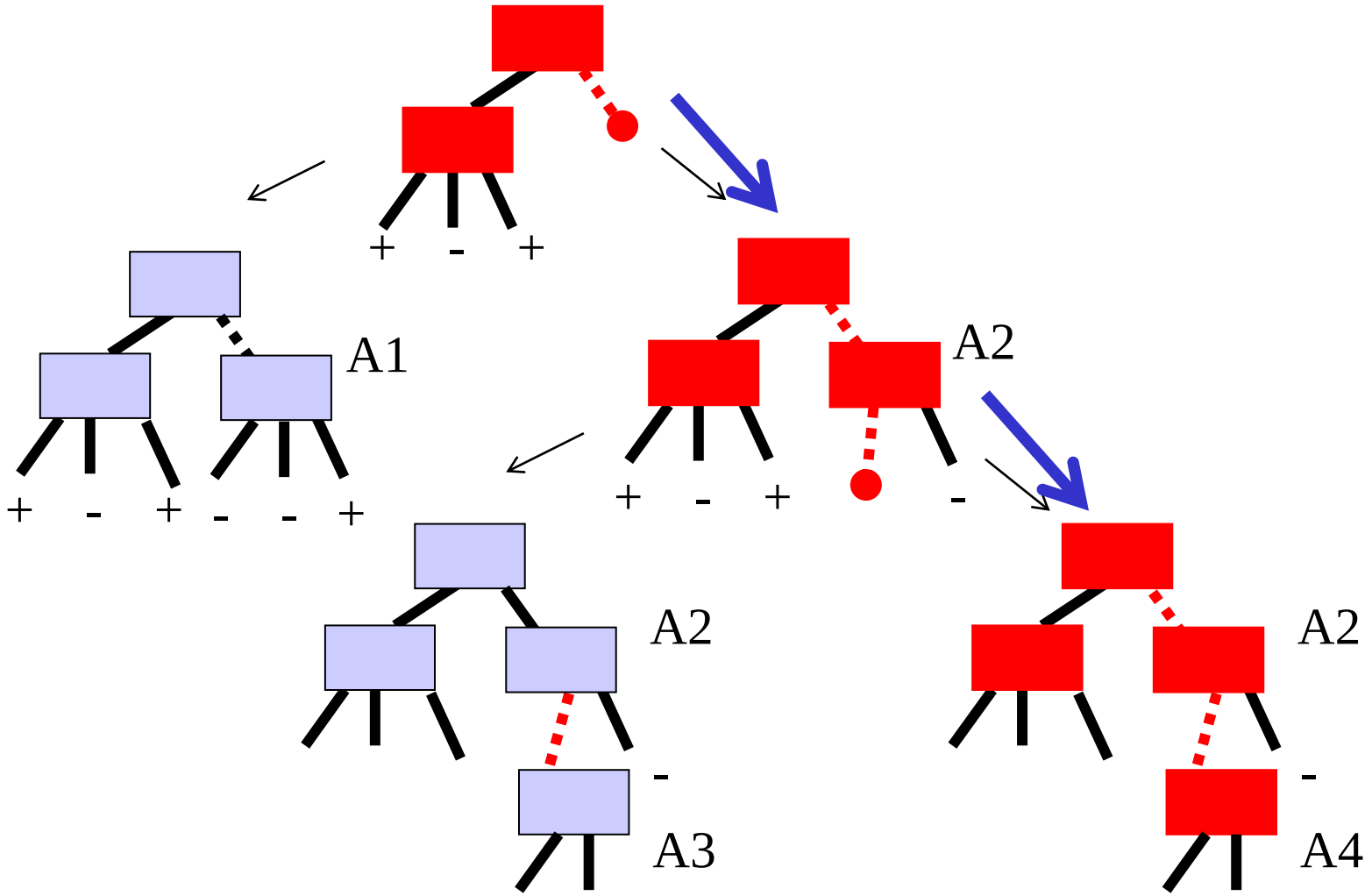


34

Hypothesis Space Search in Decision Tree Learning (ID3)

- The hypothesis space searched by ID3 is the set of possible decision trees.
- ID3 performs a simple-to complex, hill-climbing search through hypothesis space
- Begins with the empty tree, then considers progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.
- The information gain measure guides the hill-climbing search.

Hypothesis Space Search in Decision Tree Learning (ID3)



ID3 - Capabilities and Limitations

- ID3's hypothesis space of all decision trees is a **complete** Hypothesis space of finite discrete-valued functions.
  - Every finite discrete-valued function can be represented by some decision tree.
  - Target function is surely in the hypothesis space.
- ID3 maintains only a single current hypothesis, and outputs only a single hypothesis.
  - ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.
  - ID3 cannot determine how many alternative decision trees are consistent with the available training data.
- No backtracking on selected attributes (greedy search)
  - Local minimal (suboptimal splits)
- Statistically-based search choices
  - Robust to noisy data

Inductive Bias in ID3

- ID3 search strategy
  - Selects in favor of shorter trees over longer ones,
  - Selects trees that place the attributes with highest information gain closest to the root using information gain heuristic and a hill climbing strategy.
  - ID3 **does not always find the shortest consistent tree**, and it is biased to favor trees that place attributes with high information gain closest to the root.

Inductive Bias of ID3:

- Shorter trees are preferred over longer trees.
- Trees that place high information gain attributes close to the root are preferred over less information gain attributes .

Inductive Bias in ID3 – Restriction Bias and Preference Bias

- ID3 searches a complete hypothesis space
  - It searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met
  - Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy.
  - Its hypothesis space dose not introduce any additional bias.
- Candidate Elimanation searches an incomplete hypothesis space
  - It searches this space completely, finding every hypothesis consistent with the training data.
  - Its inductive bias is solely a consequence of the expressive power of its hypothesis representation.
  - Its search strategy introduces no additional bias.

Inductive Bias in ID3 – Restriction Bias and Preference Bias

- **Preference bias** (***search bias***)
  - The inductive bias of ID3 is a preference for certain hypotheses over others, with no hard restriction on the hypotheses that can be eventually enumerated.
- **Restriction bias** (***language bias***)
  - The inductive bias of Candidate Elimination is in the form of a categorical restriction on the set of hypotheses considered.
- A *preference bias* is more desirable than a *restriction bias*, because it allows the learner to work within a **complete hypothesis space** that is assured to contain the unknown target function.

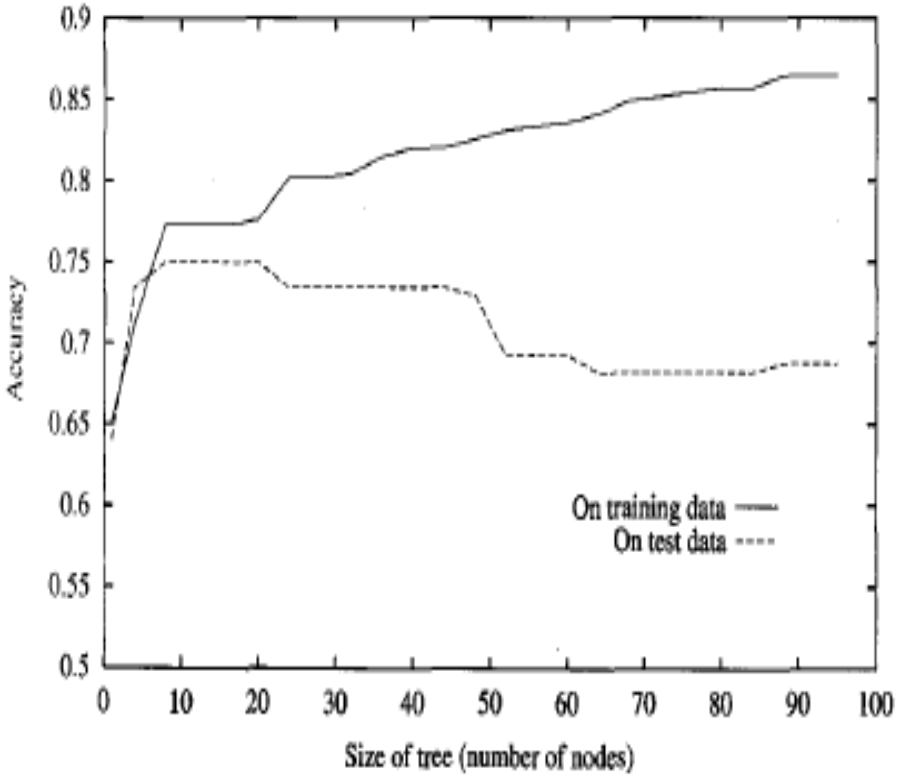
Overfitting

- Overfitting happens when **a model learns the detail and noise in the training data.**
- It leads to **negatively impacts** the **performance** of the model on **new data** i.e., noise or random fluctuations in the training data also learned as concepts by the model.
- Given a hypothesis space H, a hypothesis h∈H is said to ***OVERFIT*** the training data if there exists some alternative hypothesis h'∈H, such that h has smaller error than h' over the training examples.
- But h' has a smaller error than h over the entire distribution of instances.

Reasons for overfitting:

- Errors and noise in training examples
- Coincidental regularities (especially small number of examples are associated with leaf nodes).

- As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically.
- However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases.



Hyperparameter

- Hyperparameters values control the learning process and determine the values of learner (model) parameters.
- Learner completes the learning with these parameters.
- ‘hyper’ denotes ‘top-level’ parameters that control the learning process, and the model utilizes these parameters for better result.

E.g., Hyperparametrs in Decision tree algorithm

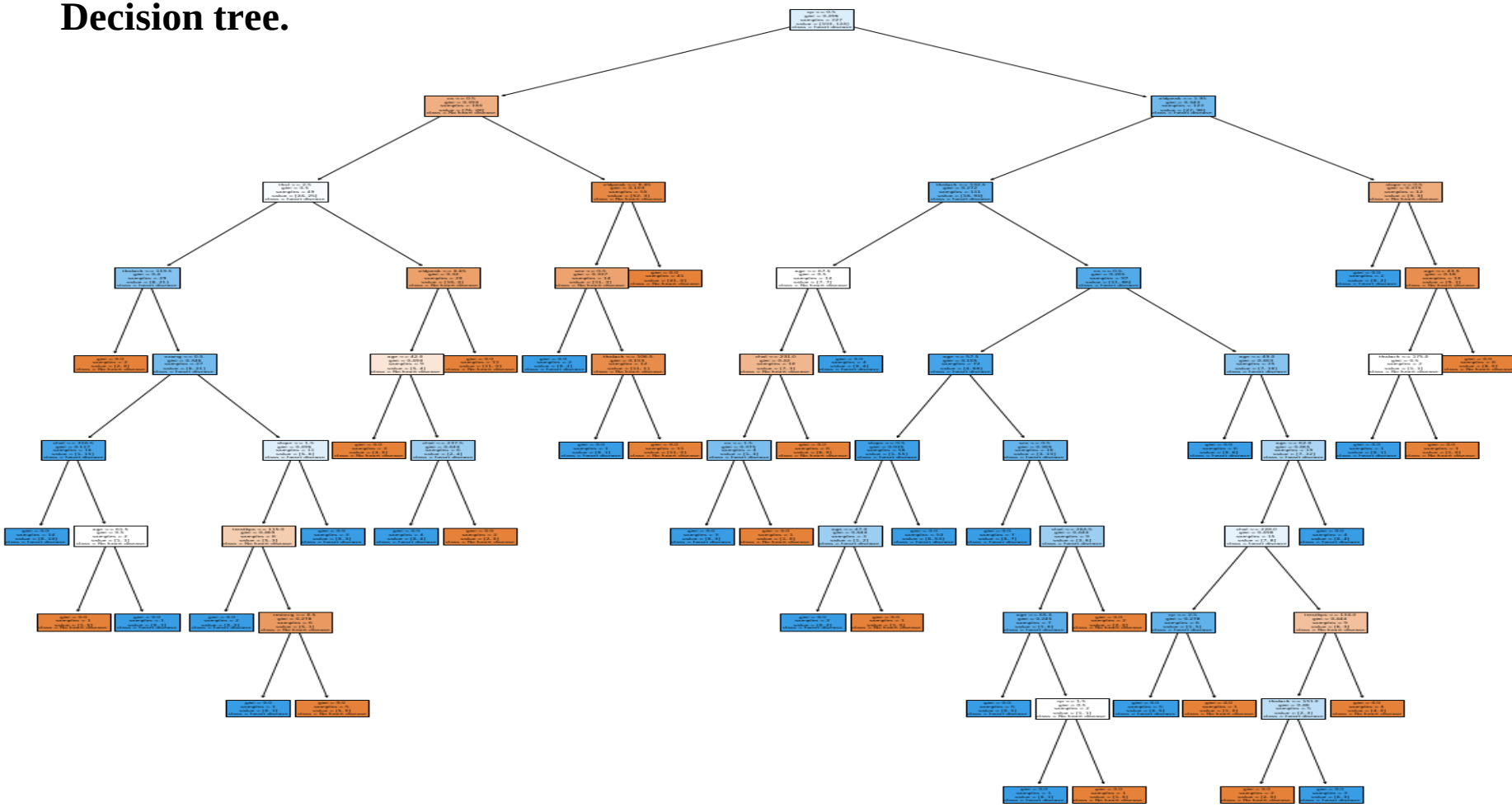
- max_depth, max_features, max_leaf_nodes, max_samples, min_impurity_decrease, min_impurity_split, min_samples_leaf, min_samples_split, min_weight_fraction_leaf, n_estimators, random_state

Hyperparameter tuning

- Hyperparameter tuning is the process of tuning the parameters to build machine learning models for good performance.
- Hyperparameter tuning provides better model with less error rate.
- **RandomSearchCV** is used during search space is large.
- **GridSearchCV** creates a grid over the search space and evaluates the model for all possible hyperparameters in the space.

Pruning – Solves Overfitting

- Pruning is a process of deleting the unnecessary nodes from a tree to build an **Optimal Decision tree.**



Why is Pruning needed?:

- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset.
- But pruning decreases the learning tree depth without reducing accuracy.

Two Types of Pruning

1. Post (backward) Pruning is used after construction of decision tree.

- Post pruning control the branches of decision tree i.e., depth of the tree and attributes splitting.
  - **Cost Complexity Pruning approach**.
  - It provides two values: ccp_alphas and impurities.
  - ccp_alphas gives minimum leaf value of decision tree.
  - Each ccp_aphas creates different classifiers (learners) and choose best classifier out of it.

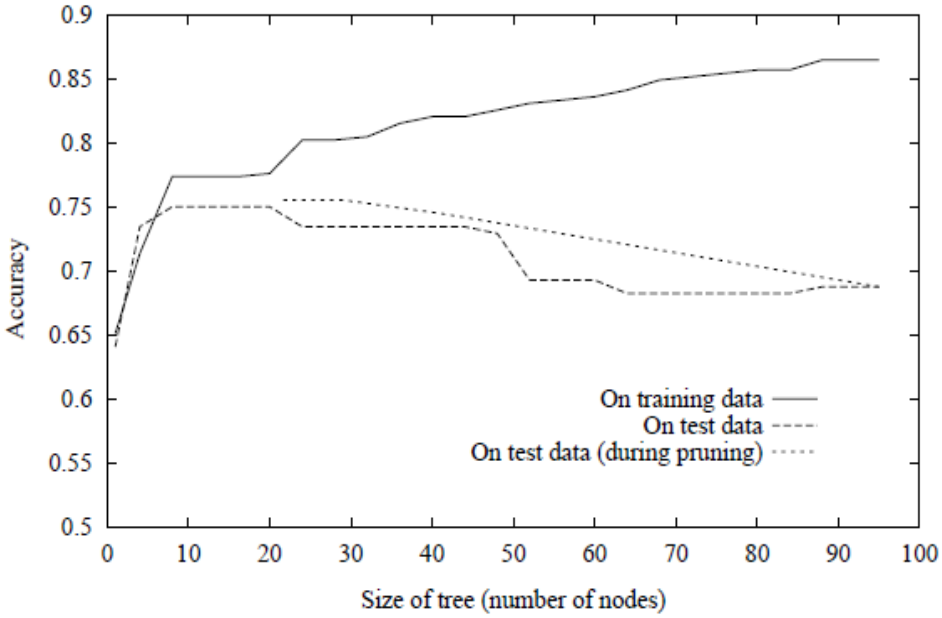
2. Pre-Pruning is used before construction of decision tree.

- Pre-Pruning is implemented using **Hyperparameter tuning (**GridSearchCV or RandomizedSearchCV) approaches.
  - **Reduced Error Pruning approach.**

Reduced-Error Pruning

- Split data into *training* and *validation* set
- Do until further pruning is harmful:
  - Evaluate impact on *validation* set by pruning each possible node (and those below it)
  - Greedily remove the one that improves the *validation* set accuracy
- Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the *validation* set).
- Using a separate set of data to guide pruning is an effective approach provided a large amount of data is available.
  - The major drawback is that when data is limited, withholding part of it for the validation set reduces even further the number of examples available for training.

- The accuracy increases over the test set as nodes are pruned from the tree.
- The validation set used for pruning is distinct from both the training and test sets.



Why Convert The Decision Tree To Rules Before Pruning?

- Converting to rules improves readability.
  - Rules are often easier for to understand.
- Distinguishing different contexts in which a node is used
  - separate pruning decision for each path
- No difference for root/inner
  - no bookkeeping on how to reorganize tree if root node is pruned

Continuous-Valued Attributes

- ID3 is restricted to attributes that take on a discrete set of values.
- Define new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals
- For a continuous-valued attribute A that is, create a new boolean attribute Ac, that is true if A < c and false otherwise.
  - Select c using information gain
  - Sort examples according to the continuous attribute A,
  - Then identify adjacent examples that differ in their target classification
  - Generate candidate thresholds midway between corresponding values of A.
  - The value of c that maximizes information gain must always lie at a boundary.
  - These candidate thresholds can then be evaluated by computing the information gain associated with each attribute.

Continuous-Valued Attributes - Example

Temperature:	40	48	60	72	80	90
PlayTennis :	No	No	Yes	Yes	Yes	No

Two candidate thresholds: (48+60)/2=54 (80+90)/2=85

Check the information gain for new boolean attributes:

$\text{Temperature}_{>54}$ $\text{Temperature}_{>85}$

Use these new new boolean attributes same as other discrete valued attributes.

Alternative Selection Measures

- Information gain measure favors attributes with many values
  - separates data into small subsets
  - high gain, poor prediction
- Ex. Date attribute has many values, and may separate training examples into very small subsets (even singleton sets – perfect partitions)
  - Information gain will be very high for Date attribute.
  - Perfect partition ➔maximum gain : Gain(S,Date) = Entropy(S) – 0 = Entropy(S) because $\log_2 1$ is 0.
  - It has high information gain, but very poor predictor for unseen data.
- There are alternative selection measures such as *GainRatio* measure based on *SplitInformation*

Split Information

- The *gain ratio* measure penalizes attributes with many values (such as Date), called *split informution*

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

- Split information for boolean attributes is 1 (= $\log_2 2$),
- Split information fot attributes for n values is $\log_2 n$

$$\text{GainRatio}(S,A) = \text{Gain}(S,A) / \text{SplitInformation}(S,A)$$

- *SplitInformation* term discourages the selection of attributes with many uniformly distributed values.

Practical Issues on Split Information

- Some value ‘rules’
  - $|S_i|$ close to $|S|$
  - SplitInformation 0 or very small
  - GainRatio undefined or very large
- Apply heuristics to select attributes
  - compute Gain first
  - compute GainRatio only when Gain large enough (above average Gain)

Missing Attribute Values

- The available data may be missing values for some attributes.
- It is common to estimate the missing attribute value based on other examples for which this attribute has a known value.
- Assume that an example (with classification c) in S has a missing value for attribute A.
  - Assign the most common value of A in S.
  - Assign the most common value of in the examples having c classification in S.
  - Or, use probabilty value for each possibşe attribute value.

Attributes with Differing Costs

- Measuring attribute costs something
  - prefer cheap ones if possible
  - use costly ones only if good gain
  - introduce cost term in selection measure
  - no guarantee in finding optimum, but give bias towards cheapest
- Example applications
  - robot & sonar: time required to position
  - medical diagnosis: cost of a laboratory test

VIT-AP UNIVERSITY

References

1. Tom M. Mitchell, Machine Learning, McGraw Hill , 2017.
2. EthemAlpaydin, Introduction to Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2017.
3. Wikipedia