Data Mining Algorithms: Classification

—

Department of Computer Science and Engineering
Kathmandu University

Classification

A form of data analysis that extracts models, called **classifiers**, describing important **data classes.**

Classifiers predict categorical class labels.

Data classification is a two-step process.

1. **(Supervised) Learning**, where a classification model is constructed describing a predetermined set of data classes
2. **Classification**, where the model is used to predict class labels for given data

Supervised learning

- Building a model from past experience (training data)
- The classifier is told to which class each training tuple belongs

Independent variables, $X = (x_1, x_2, ..., x_n)$

Class variable, y
(Dependent variable/
Target variable)

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes

Supervised learning

Learning of **a mapping or function, y = f(X)**, that can predict the associated class label y of a given tuple

Typically this mapping is represented in the form of **classification rules, decision trees**, or **mathematical formulae**

Inferring rudimentary rules: 1R algorithm

1R algorithm

- The simplest rule-based classification learning algorithm for discrete attributes.
- Learns rules that all test one particular attribute (basically a 1-level decision tree)
- Choose an attribute with the lowest error rate (proportion of instances that don’t belong to the majority class)

1R algorithm

Given a table T of labelled instances, and a class attribute C, the 1R algorithms returns a rule that predicts C on the basis of a single predictive attribute A in T; i.e., it returns a rule of the form

```
If A is:
    V1, then C is W1
    V2, then C is W2
    …
    Vk, then C is Wk
```

where V_1 ... V_k range over the possible values of A, and W_1 ... W_k are possible values of C. The W_i need not all be different and need not cover all possible values of C.

1R Algorithm

1. For each attribute,
    a. For each value of the attribute, make a rule as follows:
        i. count how often each class appears
        ii. find the most frequent class
        iii. make the rule assign that class to this attribute-value
    b. Calculate the error rate of the rules
2. Choose the rules with the smallest error rate

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Class: Play?

Set of independent variables:

{Outlook, Temperature, Humidity, Windy}

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Outlook = Sunny and Play? = Yes

2 instances

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Outlook = Sunny and Play? = Yes

2 instances

Outlook = Sunny and Play = No

3 instances

$\therefore$ Majority class is No.

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

For the rule

Outlook:

Sunny → No

Error rate is 2/5

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Similarly, for the rule

Outlook:

Overcast → Yes

Error rate is 0/4

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

For the rule

Outlook:

Rain → Yes

Error rate is 2/5

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

For the rule

Outlook:

Sunny → No

Overcast → Yes

Rain → Yes

Total error rate is

$(2 + 0 + 2)/ (5 + 4 + 5) = 4/14$

We continue in the same manner for other attributes.

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Attribute	Rules	Error rate	Total error rate
Outlook	Sunny → No Overcast → Yes Rain → Yes	2/5 0/4 2/5	4/14
Temperature	Hot → Mild → Cool →	?	?

1R Algorithm: An example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Attribute	Rules	Error rate	Total error rate
Outlook	Sunny → No Overcast → Yes Rain → Yes	2/5 0/4 2/5	4/14
Temperature	Hot → No Mild → Yes Cool → Yes	2/4 2/6 1/4	5/14
Humidity	High → No Normal → Yes	3/7 1/7	4/14
Windy	False → Yes True → No	2/8 3/6	5/14

1R Algorithm: An example

There are two solutions.

- First solution

  If Outlook is:
  - Sunny, then play is no
  - Overcast, then play is yes
  - Rainy, then play is yes

- Second solution

  If Humidity is:
  - High, then play is no
  - Normal, then play is yes

The final solution can be arbitrarily selected from one of them.

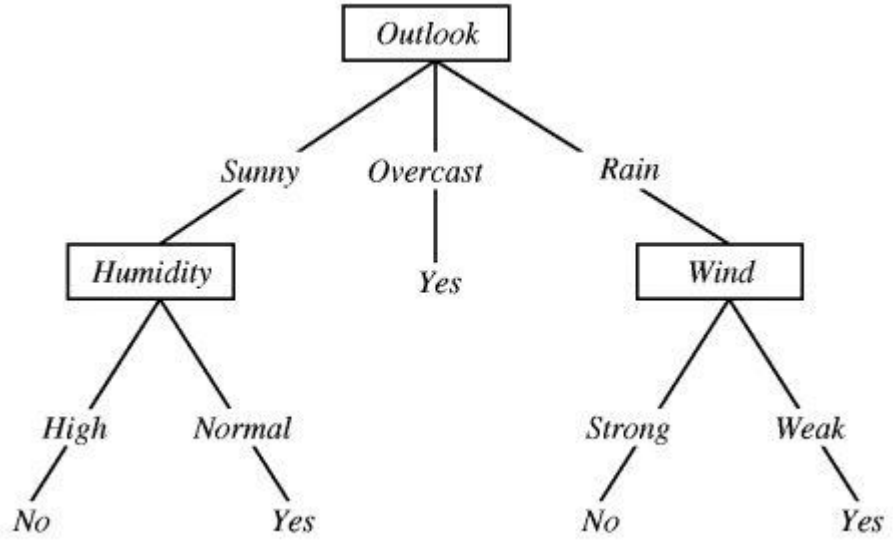
Decision trees

Decision tree induction

Learning a decision trees from class-labeled training tuples

Decision tree:

- Each internal node(non-leaf node) denotes a test on an attribute,
- Each branch represents an outcome of the test, and
- Each leaf node holds a class label

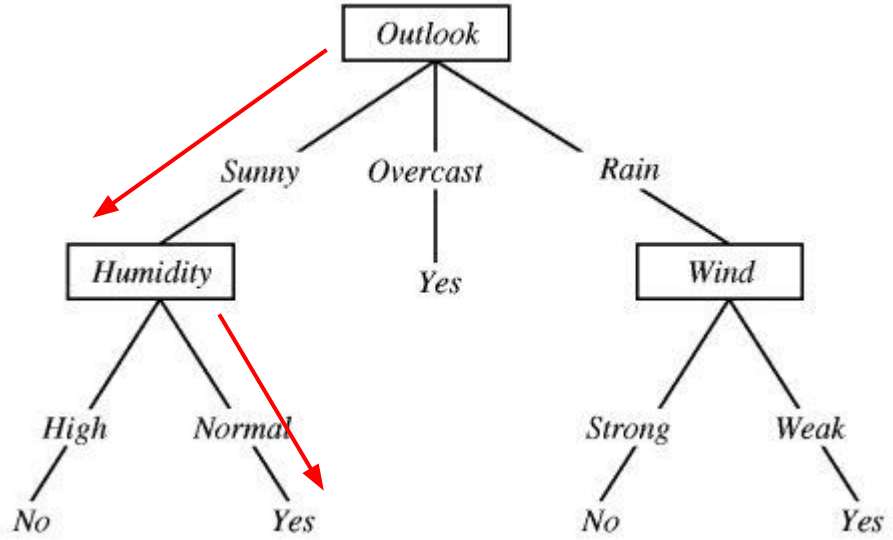


Decision trees for prediction

Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree.

A path is traced from the root to a leaf node, which holds the class prediction for that tuple.

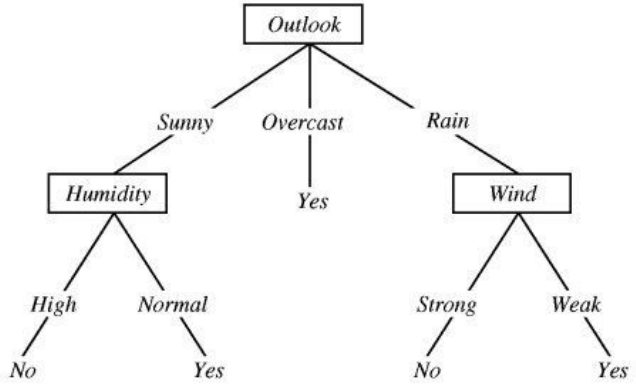
Example:
If outlook = Sunny and Humidity = Normal, then the decision would be play tennis.



Rule induction systems

From each decision tree, a set of rules can be inferred, rules that can replace the decision tree.

Example: This decision tree can be replaced by the following set of rules:



- Outlook = Overcase → Play = Yes
- Outlook = Sunny and Humidity = High → Play = No
- Outlook = Sunny and Humidity = Normal → Play = Yes
- Outlook = Rain and Wind = Strong → Play = No
- Outlook = Rain and Wind = Weak → Play = Yes

Learning decision trees

Greedy approach (adopted by ID3, C4.5, CART):

Construct the decision tree in a top-down manner choosing at each node the “best” attribute for branching

Basic algorithm:

1. First a root attribute is chosen, building a separate branch for each different value of the attribute
2. The training set is also divided, each branch inheriting the examples matching the attribute value of the branch

Learning decision trees

Basic algorithm (continued)

3. Process repeats for each descendant until all examples have the same class (in that case the node becomes a leaf labeled with that class) or all attributes have been used (the node also become a leaf labeled with the *mode* value – the majority class)

   Note that an attribute cannot be chosen twice on the same path; from the moment it was chosen for a node it will never be tested again for the descendants of that node.

Attribute selection measures

An **attribute selection measure** is a **heuristic** for selecting the **splitting criterion** that “best” separates a given data partition, D, of class-labeled training tuples into individual classes.

If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be **pure** (i.e. all the tuples in it belong to the same class).

Conceptually, the **“best” splitting criterion** is the one that most closely results in such a scenario.

Attribute selection measures

The attribute selection measure provides a ranking for each attribute describing the given training tuples.

The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples.

3 popular attribute selection measures:

1. Information gain
2. Gain ratio
3. Gini index

Information gain

ID3 (**I**terative **D**ichotomiser) algorithm uses information gain as its attribute selection measure.

This measure is based on information theory.

The attribute with the highest gain is the one that minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.

Information gain

The expected information needed to classify a tuple in D (aka entropy) is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2 (p_i)$$

Where $p_i = |C_{i,D}|/|D|$ is the nonzero probability that a tuple belongs to class C_i,

m is the number of classes

$C_{i,D}$ is the set of tuples of class C_i in D

$|C_{i,D}|$ and |D| are the number of tuples in D and $C_{i,D}$ respectively.

Information gain

Now, suppose we were to partition the tuples in D on some attribute A having v distinct values $\{a_1, a_2, .., a_v\}$, *as observed from the training data*, such that D is partitioned into v subsets $\{D_1, D_2, ... D_v\}$, where D_j contains those tuples in D that have outcome a_j of A.

Ideally, we would like each partition to be pure. However, it is quite likely that the partitions will be impure.

How much more information would we still need (after the partitioning) to arrive at exact classification?

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Information gain

$Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A.

The smaller the expected information (still) required, the greater the purity of the partitions.

Information gain, $Gain(A) = Info(D) - Info_A(D)$

Gain(A) is the expected reduction in the information requirement caused by knowing the value of A.

The attribute with the highest Gain(A) is chosen as the splitting attribute at node N.

Example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Play?	Count	p_i
Y	9	9/14
N	5	5/14

No. of classes, $m = 2$

$$\begin{aligned} \text{Info}(D) &= -\sum_{i=1}^{m} p_i \log_2(p_i) \\ &= -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} \\ &= 0.94 \end{aligned}$$

Example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

		Humidity	
		High	Normal
Play?	Y	3	6
	N	4	1
Number of instances in partition j	$\|D_j\|$	7	7

$$\text{Info}(D_1) = -\frac{3}{7}log_2\frac{3}{7} - \frac{4}{7}log_2\frac{4}{7} = 0.985$$

$$\text{Info}(D_2) = -\frac{6}{7}log_2\frac{6}{7} - \frac{1}{7}log_2\frac{1}{7} = 0.592$$

$$\text{Info}_{Humidity}(D) = \sum_{j=1}^{v}\frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

$$= \frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592$$

$$= 0.79$$

$$\text{Gain(Humidity)} = 0.94 - 0.79 = 0.15$$

Example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Similarly,

$$\text{Info}_{\text{Wind}}(\text{D}) = 0.89$$

$$\text{Info}_{\text{Temperature}}(\text{D}) = 0.91$$

$$\text{Info}_{\text{Outlook}}(\text{D}) = 0.69$$

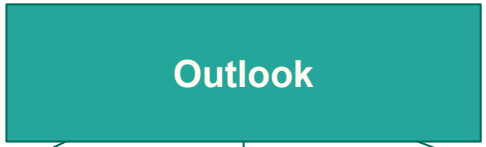
Example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Attribute	**$Info_A(D)$**	**Gain(A)**
Humidity	0.79	0.15
Wind	0.89	0.05
Temperature	0.91	0.03
Outlook	**0.69**	**0.25**

Example

Example



Sunny

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

Overcast

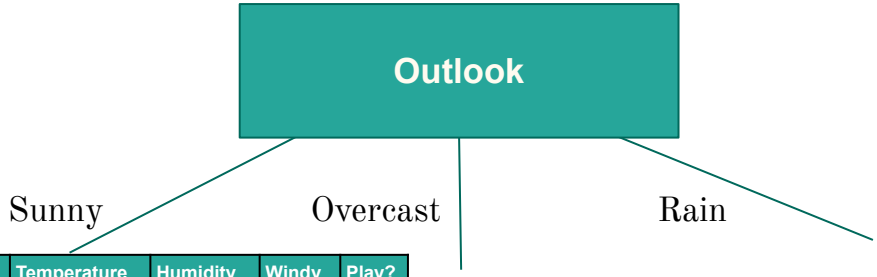
Outlook	Temperature	Humidity	Windy	Play?
overcast	hot	high	false	Yes
overcast	cool	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes

Rain

Outlook	Temperature	Humidity	Windy	Play?
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
rain	mild	normal	false	Yes
rain	mild	high	true	No

For each of these three branches, we continue searching for the best attribute for further splitting.

Example



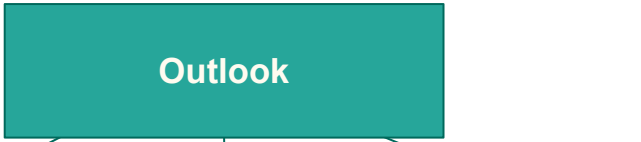
Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

For Outlook = Sunny branch

Play?	Count	p_i
Y	2	2/5
N	3	3/5

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$
$$= -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5}$$
$$= 0.971$$

Example



Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

For Outlook = Sunny branch

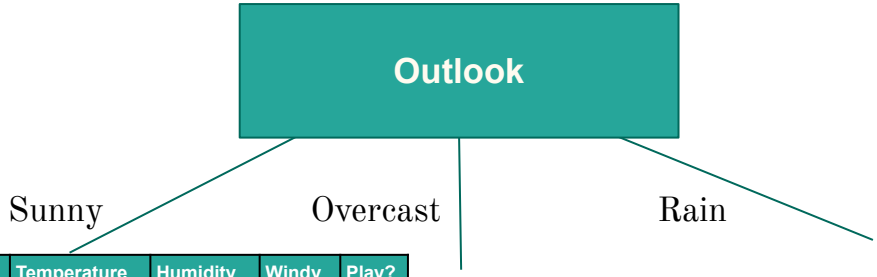
	Humidity: High	Humidity: Normal
Play? Y	0	2
Play? N	3	0
$\|D_j\|$	3	2

$$\text{Info}_{Humidity}(D) = \sum_{j=1}^{2} \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

$$= \frac{3}{5} \times \left(-\frac{0}{3} log_2 \frac{0}{3} - \frac{3}{3} log_2 \frac{3}{3} \right) + \frac{2}{5} \times \left(-\frac{2}{3} log_2 \frac{2}{3} - \frac{0}{3} log_2 \frac{0}{3} \right)$$

$$= 0$$

Example



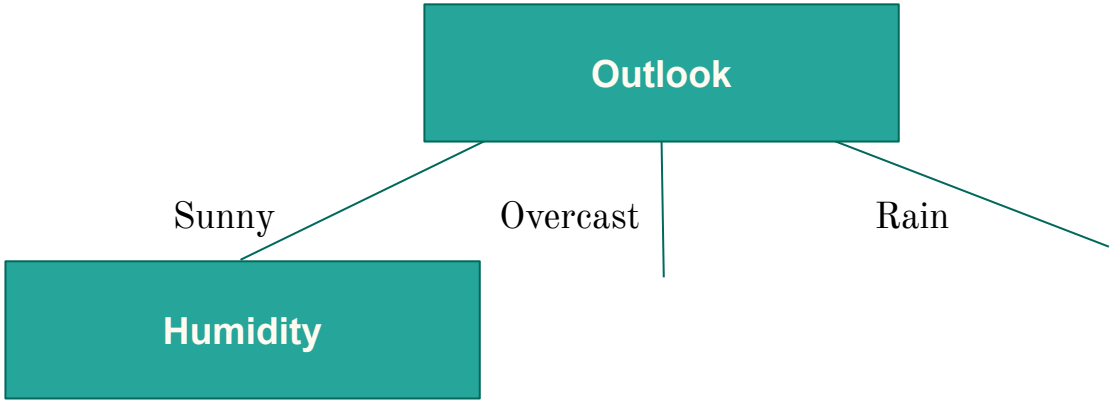
Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

For Outlook = Sunny branch

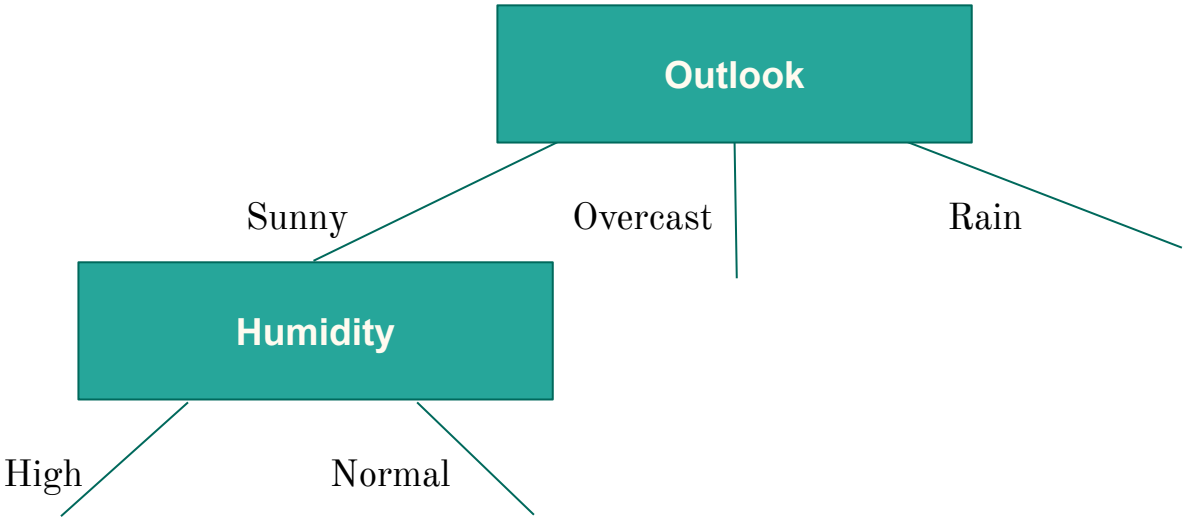
$$\text{Info}_{Humidity}(D) = 0$$
$$\text{Info}_{Windy}(D) = 0.62$$
$$\text{Info}_{Temperature}(D) = 0.4$$

$$\text{Gain(Humidity)} = 0.971 - 0 = \mathbf{0.971}$$
$$\text{Gain(Windy)} = 0.971 - 0.62 = 0.351$$
$$\text{Gain(Temperature)} = 0.971 - 0.4 = 0.571$$

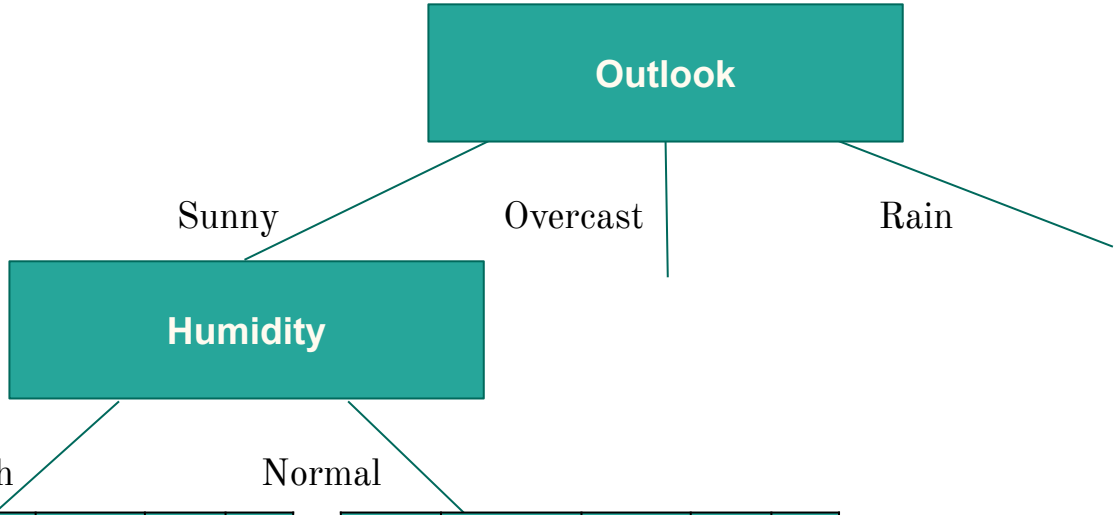
Example



Example



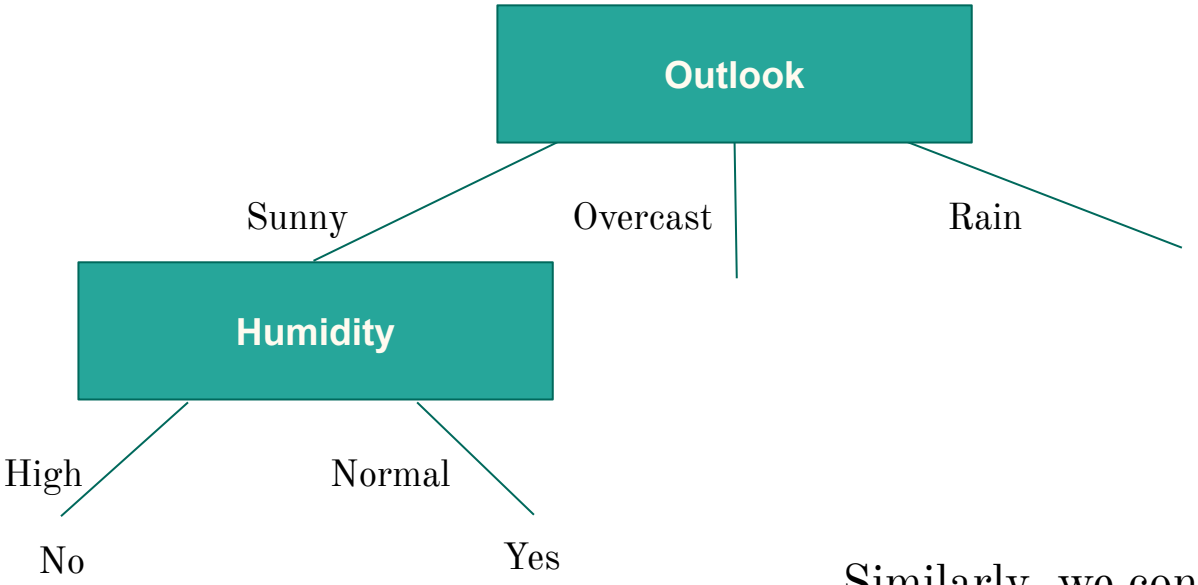
Example



Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	**high**	false	No
sunny	hot	**high**	true	No
sunny	mild	**high**	false	No

Outlook	Temperature	Humidity	Windy	Play?
sunny	cool	**normal**	false	Yes
sunny	mild	**normal**	true	Yes

Example



Similarly, we continue with the other attributes to grow the tree.

Gain ratio

Information gain prefers to select attributes having a large number of values.

Gain ratio attempts to overcome this bias.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

where,

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

C4.5, a successor of ID3, uses gain ratio.

Assignment: Learn a decision tree from the previous dataset using gain ratio.

Gini index

CART (**C**lassification **A**nd **R**egression **T**rees) algorithm uses Gini index.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

The Gini index considers a binary split for each attribute A.

If A is a discrete-valued attribute having v distinct values, $\{a_1, a_2, .. a_v\}$, we examine all the possible subsets that an be formed, i.e. $\{a_1, a_2\}$, $\{a_3, .. a_v\}$ and so on.

Gini index

If a binary split on A partitions D into D1 and D2, the Gini index of D given that partitioning is

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

The reduction in impurity that would be incurred by a binary split on A is

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

The attribute with the minimum Gini index is selected as the splitting attribute.

Tree pruning

Sometimes (when only few examples are associated with leaves) the tree **overfits** the training data and does not work well on test examples.

Overfitting is the condition when the model completely fits the training data but fails to generalize the testing unseen data.

To avoid overfitting, the tree may be simplified by pruning (i.e., removing the parts of the decision tree to prevent growing to its full depth).

- Pre-pruning
- Post-pruning

Tree pruning

Pre-pruning:

- Growing is stopped before normal end.
- The hyperparameters of the decision tree including max_depth, min_samples_leaf, min_samples_split can be tuned to early stop the growth of the tree
- The leaves are not 100% pure and are labeled with the majority class (the mode value).

Tree pruning

Post-pruning:

- Allow the decision tree model to grow to its full depth, then remove the tree branches.
- After running the algorithm some sub-trees are replaced by leaves.
- Also in this case the labels are mode values (majority class) for the matching training examples.
- Post-pruning is better because in pre-pruning it is hard to estimate when to stop