IS-ZC444: ARTIFICIAL INTELLIGENCE

Lecture-15: K-NN + Decision Tree + Random Forest

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems,
BITS Pilani, Pilani, Jhunjhunu-333031, Rajasthan, INDIA

November 02, 2018 (WILP @ BITS-Pilani Jul-Nov 2018)

Classification

Finding the right label

Classification

Finding the right label

Classification

Finding the right label

What feature (attributes) would you choose?

Color, texture, weight, density, hardness .....

K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

K Nearest Neighbor (KNN)

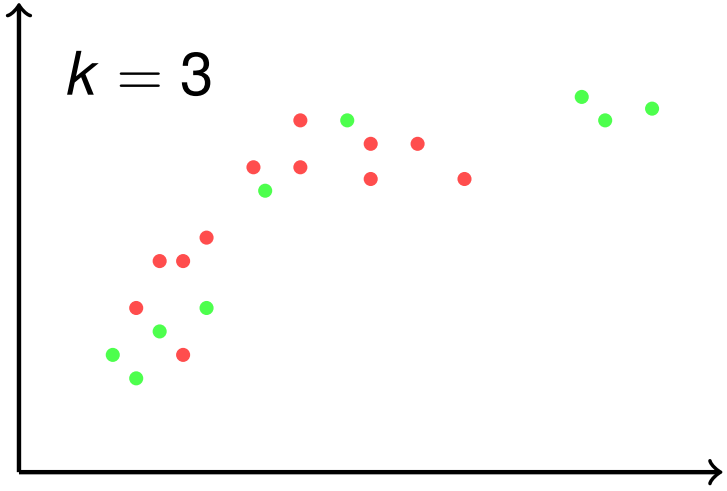
You are most likely as your friends (Bias)

- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting

K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

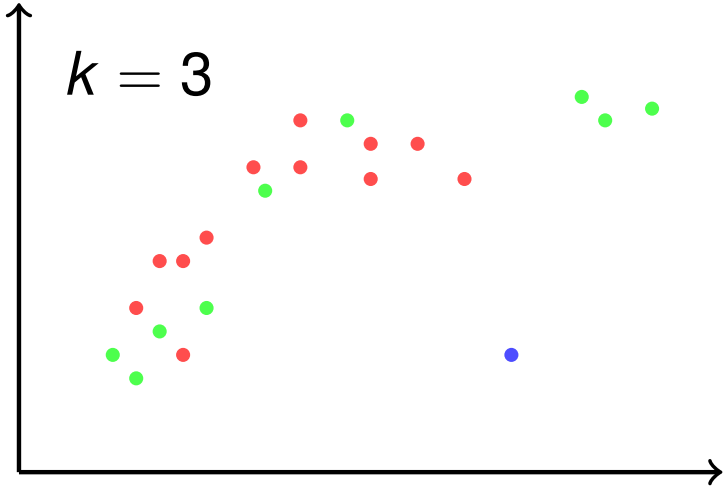
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

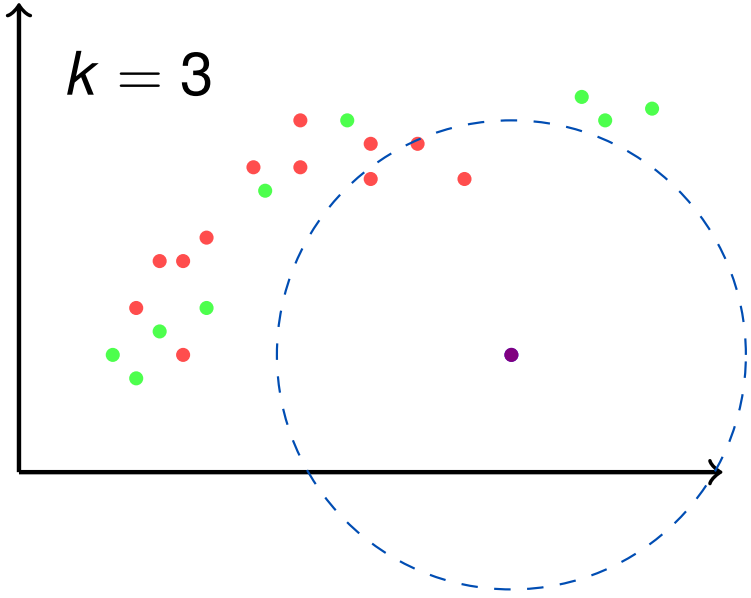
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

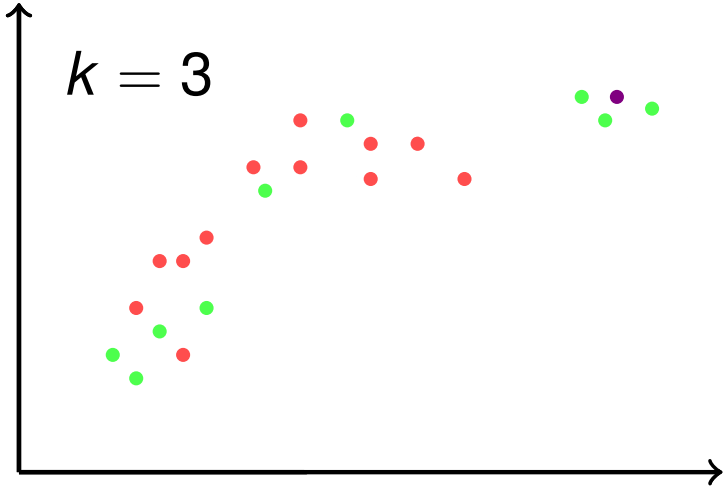
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

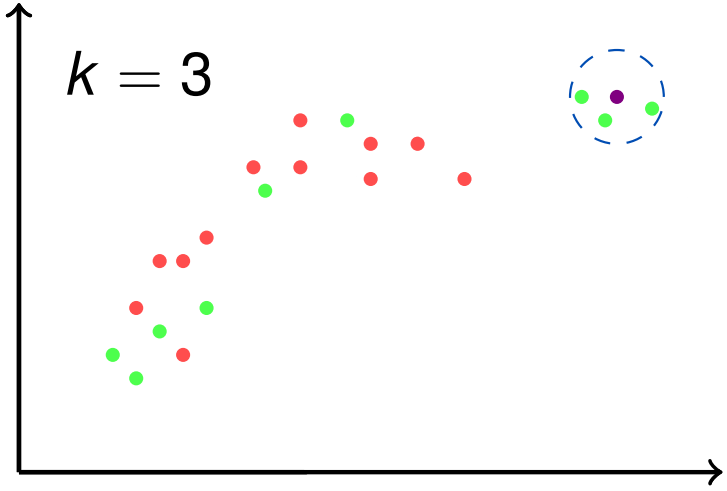
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

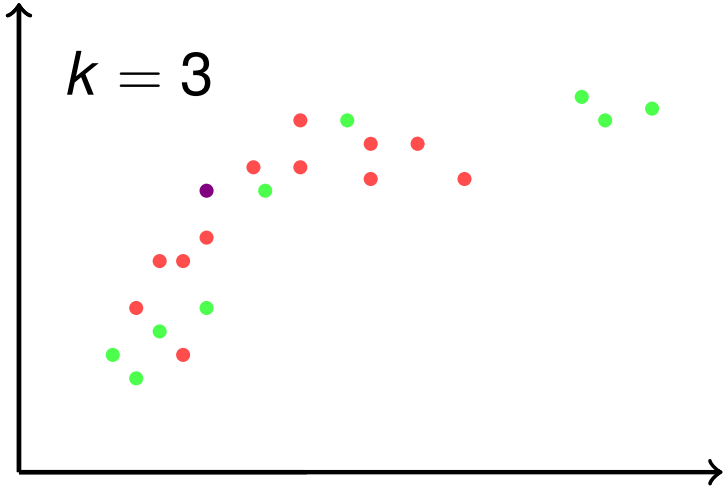
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

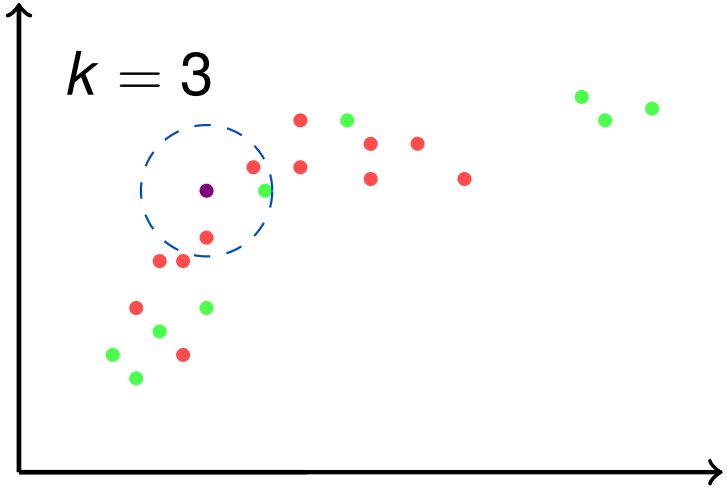
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

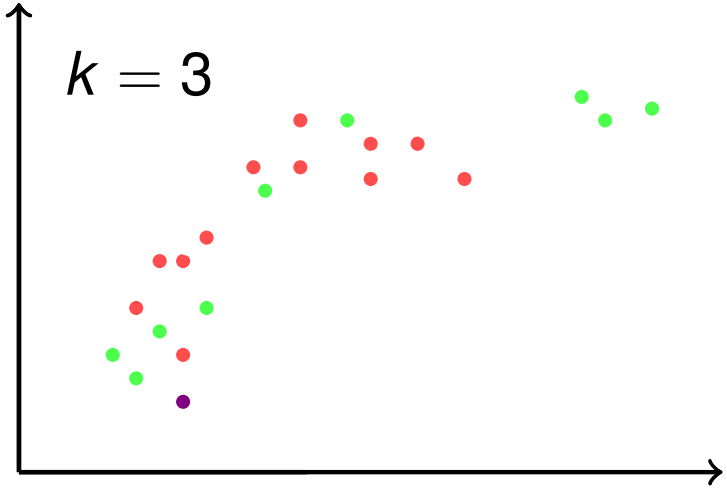
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

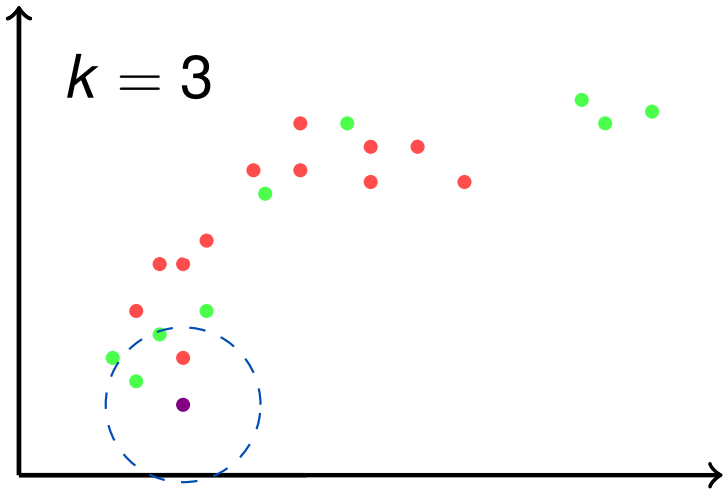
- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



K Nearest Neighbor (KNN)

You are most likely as your friends (Bias)

- Two step algorithm
  1. Search k other datum points (most difficult part)
  2. Apply majority voting
- A lazy learner
- To avoid ties, k should NOT be a multiple of number of classes
- Small k is sensitive to noise and large one has high bias



Decision Tree

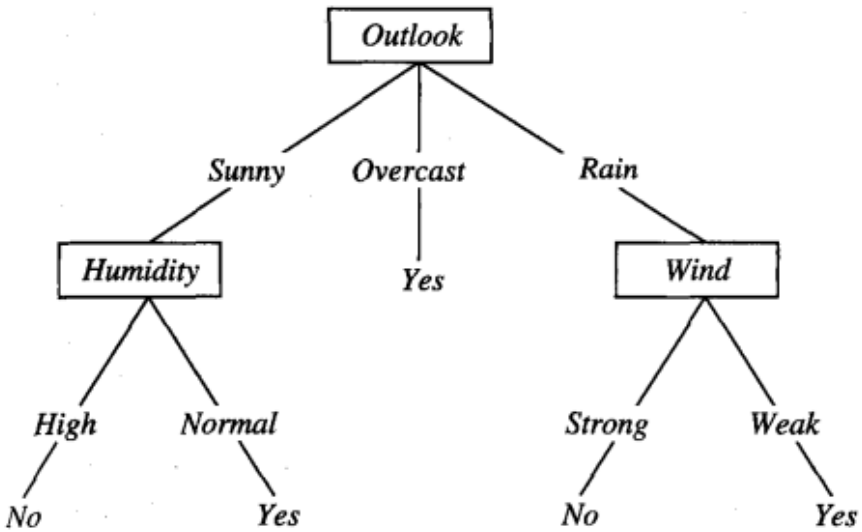
Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

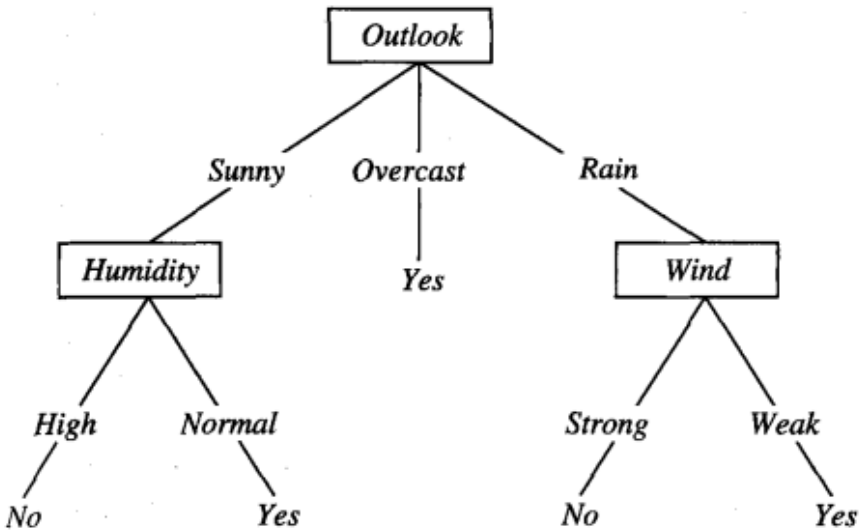


- Each node in the tree specifies a test for some attribute

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

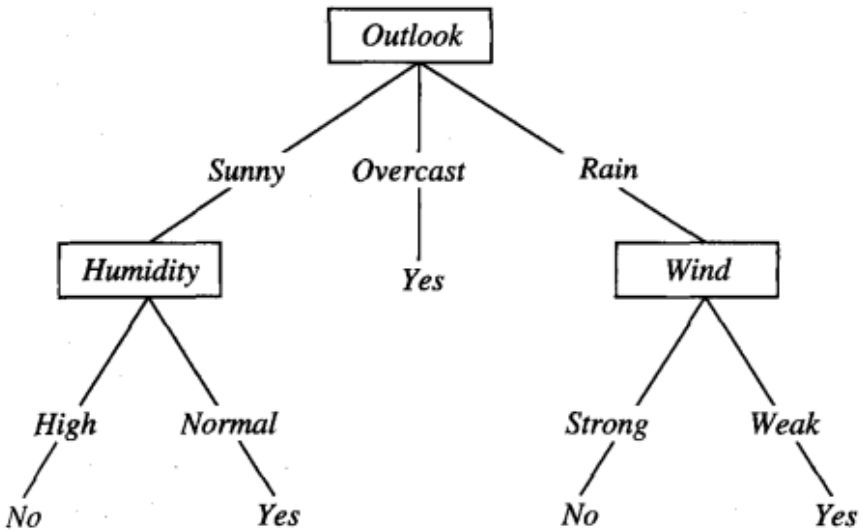


- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.

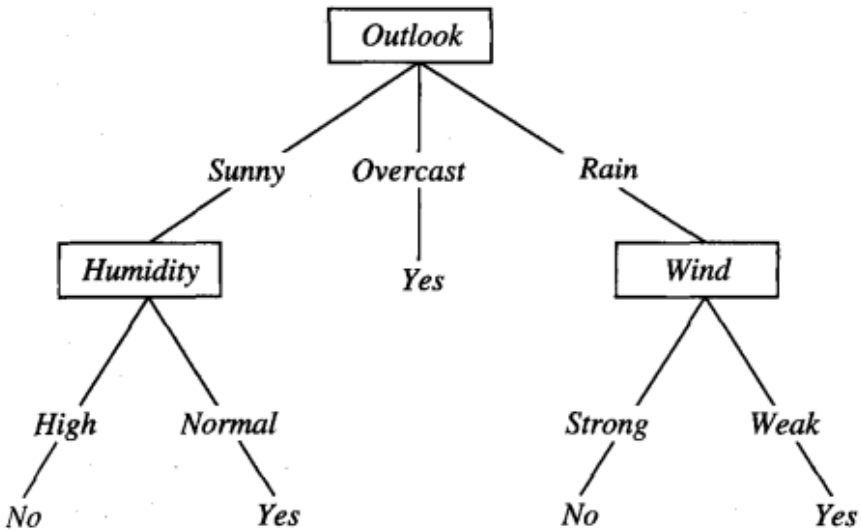


- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value
- Decision trees represent a disjunction of conjunctions

Decision Tree

Decision Tree

is a method for approximating discrete-valued functions. It is robust to noisy data and capable of learning disjunctive expressions. Primarily useful for classification.



- Each node in the tree specifies a test for some attribute
- Each branch descending from the node corresponds to one of the possible value
- Decision trees represent a disjunction of conjunctions

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$$

DT is Appropriate when

- Instances are represented by attribute-value pairs

DT is Appropriate when

- Instances are represented by attribute-value pairs
- The target function has discrete output values

DT is Appropriate when

- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required

DT is Appropriate when

- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors

DT is Appropriate when

- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors
- The training data may contain missing attribute values

Entropy

Characterizes the impurity of an arbitrary collection of examples

Entropy

Characterizes the impurity of an arbitrary collection of examples

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

Entropy

Characterizes the impurity of an arbitrary collection of examples

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

Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

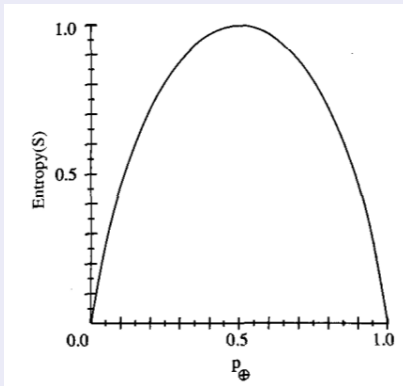
0 – when all members are of same class.

1 – if equal number of positive and negative

Entropy

Characterizes the impurity of an arbitrary collection of examples

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



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

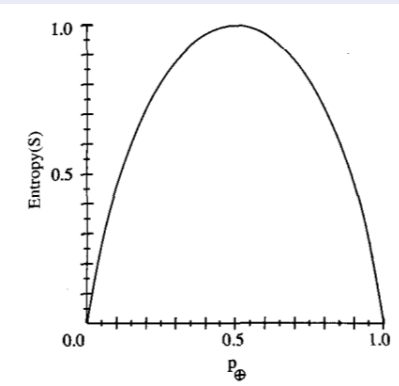
0 – when all members are of same class.

1 – if equal number of positive and negative

Entropy

Characterizes the impurity of an arbitrary collection of examples

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



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

0 – when all members are of same class.

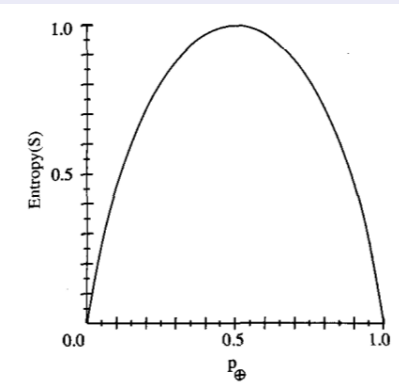
1 – if equal number of positive and negative

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

Entropy

Characterizes the impurity of an arbitrary collection of examples

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



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

0 – when all members are of same class.

1 – if equal number of positive and negative

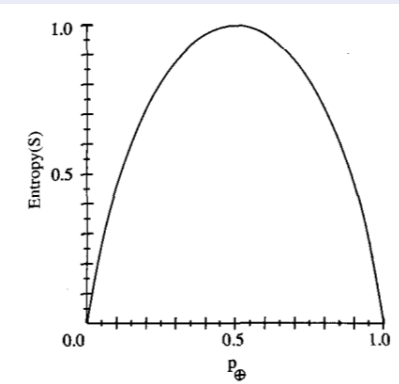
Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak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

$Entropy([9+, 5-])$

Entropy

Characterizes the impurity of an arbitrary collection of examples

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



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

0 – when all members are of same class.

1 – if equal number of positive and negative

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

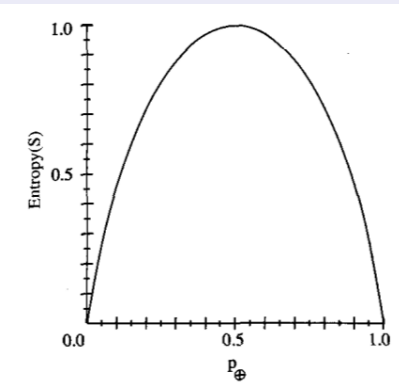
$$Entropy([9+, 5-])$$

$$= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

Entropy

Characterizes the impurity of an arbitrary collection of examples

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



Range is 0 to 1, *i.e.* $0 \leq Entropy(S) \leq 1$

0 – when all members are of same class.

1 – if equal number of positive and negative

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

$$\begin{aligned} Entropy([9+, 5-]) \\ &= -(9/14)\log_2(9/14) \\ &\quad -(5/14)\log_2(5/14) \\ &= 0.94 \end{aligned}$$

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

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

[1] Outlook, Temperature, Humidity, Wind

Information Gain

Information Gain of an attribute A[1] is the expected reduction in entropy caused by partitioning the dataset S according to that attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

here S_v contains that data items of S where the value of attribute A is v

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

For example

$$S_{Sunny} = \{D1, D2, D8, D9, D11\}$$
$$S_{Overcast} = \{D3, D7, D12, D13\}$$
$$S_{Cool} = \{D5, D6, D7, D9\}$$
$$S_{Hot} = \{D1, D2, D3, D13\}$$
$$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$$
$$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$$

And so on....

[1] Outlook, Temperature, Humidity, Wind

Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$

$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$

$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$

$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$

Information Gain

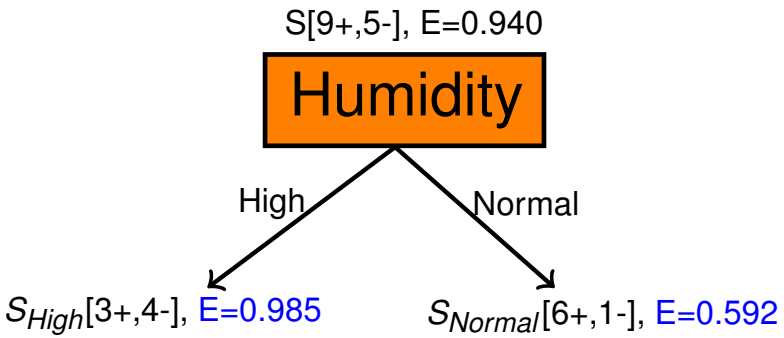
$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$

$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$

$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$

$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$



Information Gain

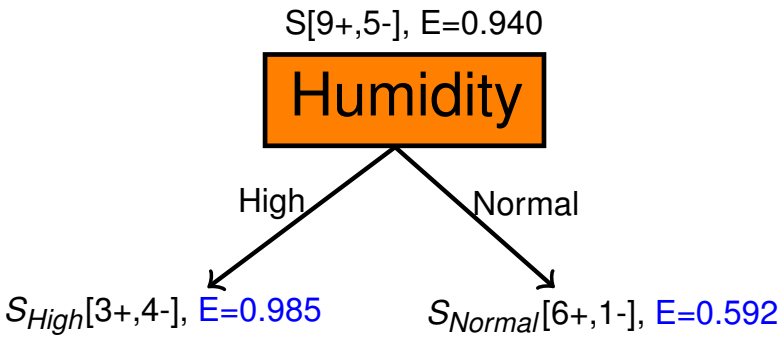
$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$

$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$

$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$

$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$



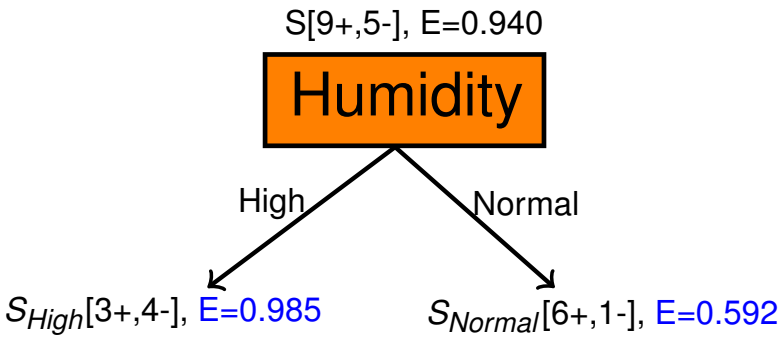
$$\text{Gain(S,Humidity)} = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$

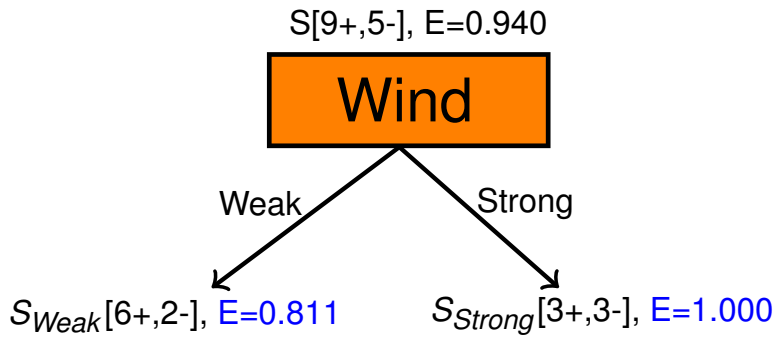
Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$S_{High} = \{D1, D2, D3, D4, D8, D12, D14\}$
$S_{Normal} = \{D5, D6, D7, D9, D10, D11, D13\}$

$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\}$
$S_{Strong} = \{D2, D6, D7, D11, D12, D14\}$





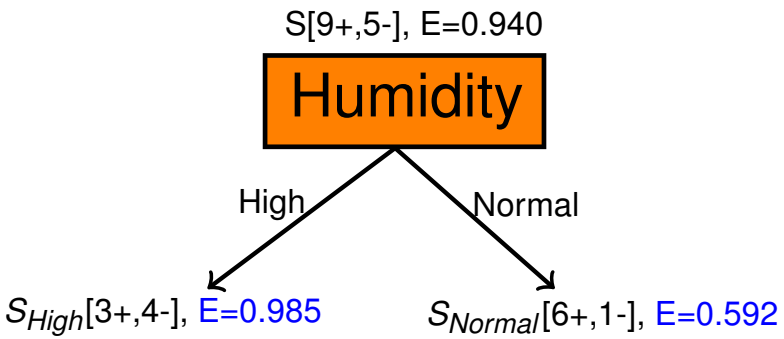
$$\text{Gain(S,Humidity)} = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$

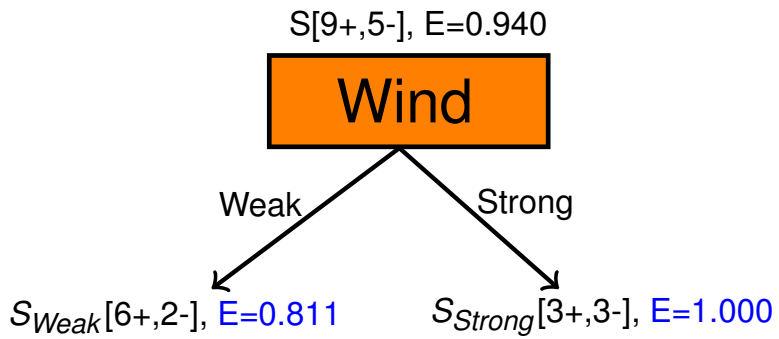
Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

S_{High} = {D1, D2, D3, D4, D8, D12, D14}
S_{Normal} = {D5, D6, D7, D9, D10, D11, D13}

S_{Weak} = {D1, D3, D4, D5, D8, D9, D10, D13}
S_{Strong} = {D2, D6, D7, D11, D12, D14}





Gain(S,Humidity)$= 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$

Gain(S,Wind) $= 0.940 - (8/14)0.811 - (6/14)1.000 = 0.048$

Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

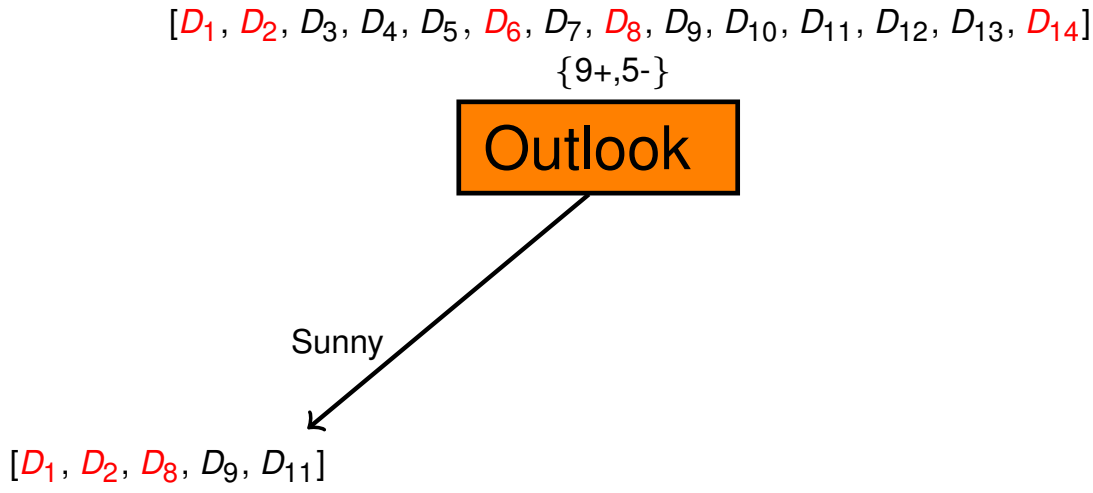
$[D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, D_{13}, D_{14}]$

{9+,5-}

Outlook

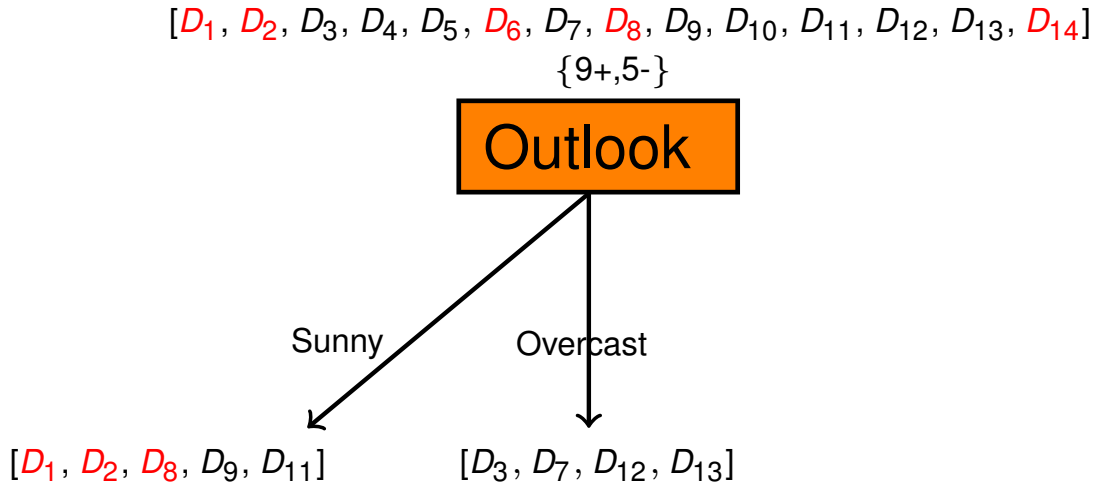
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



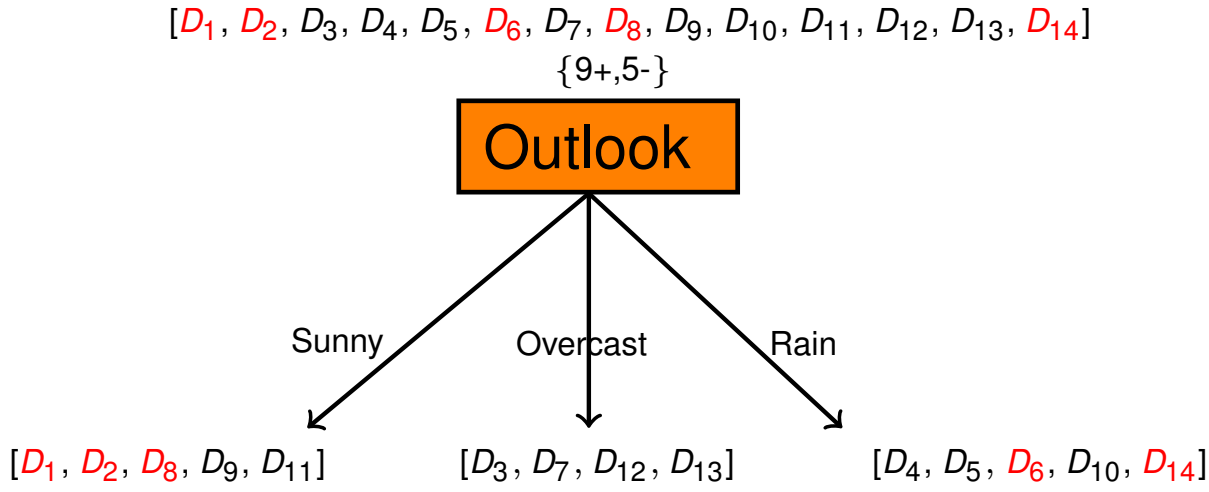
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



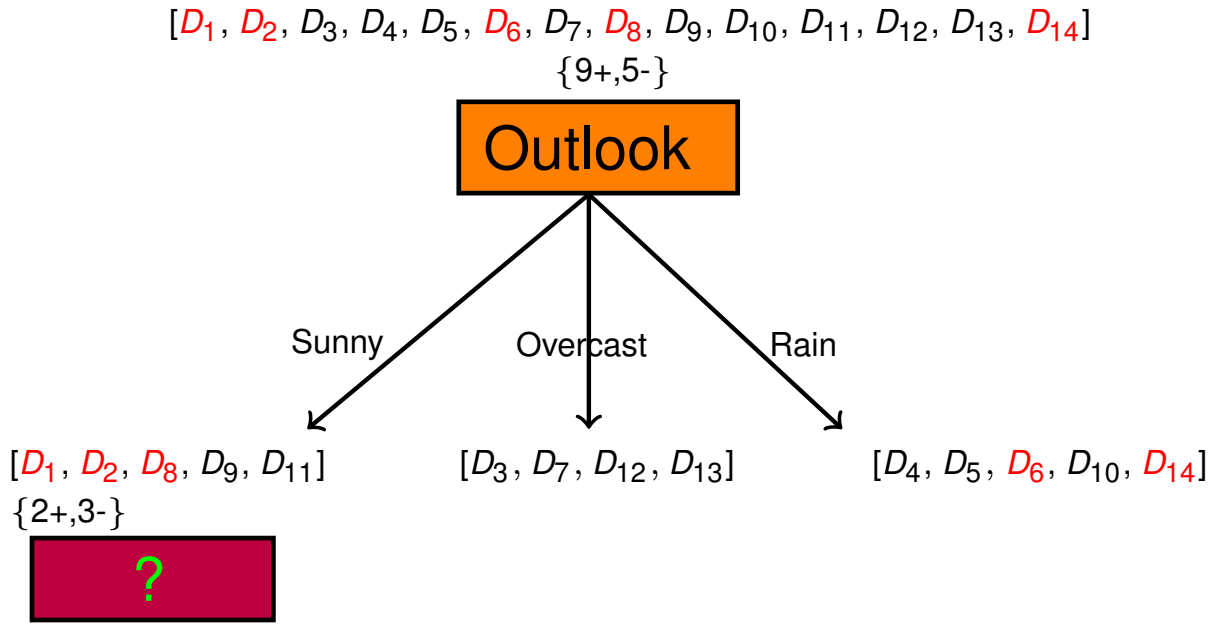
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



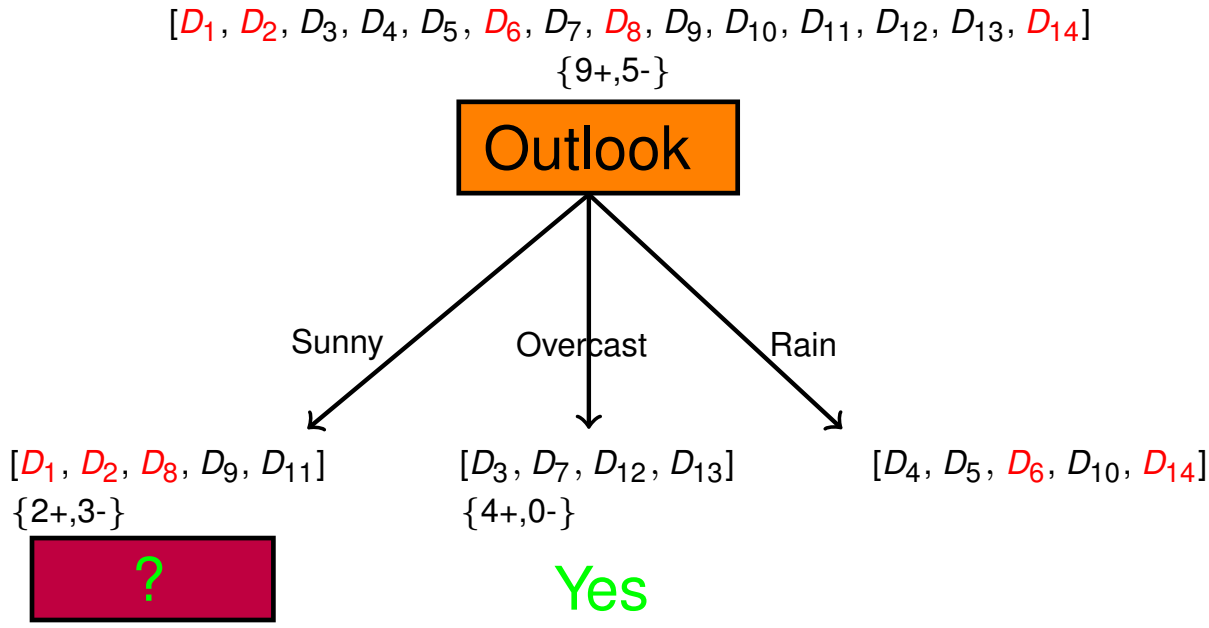
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



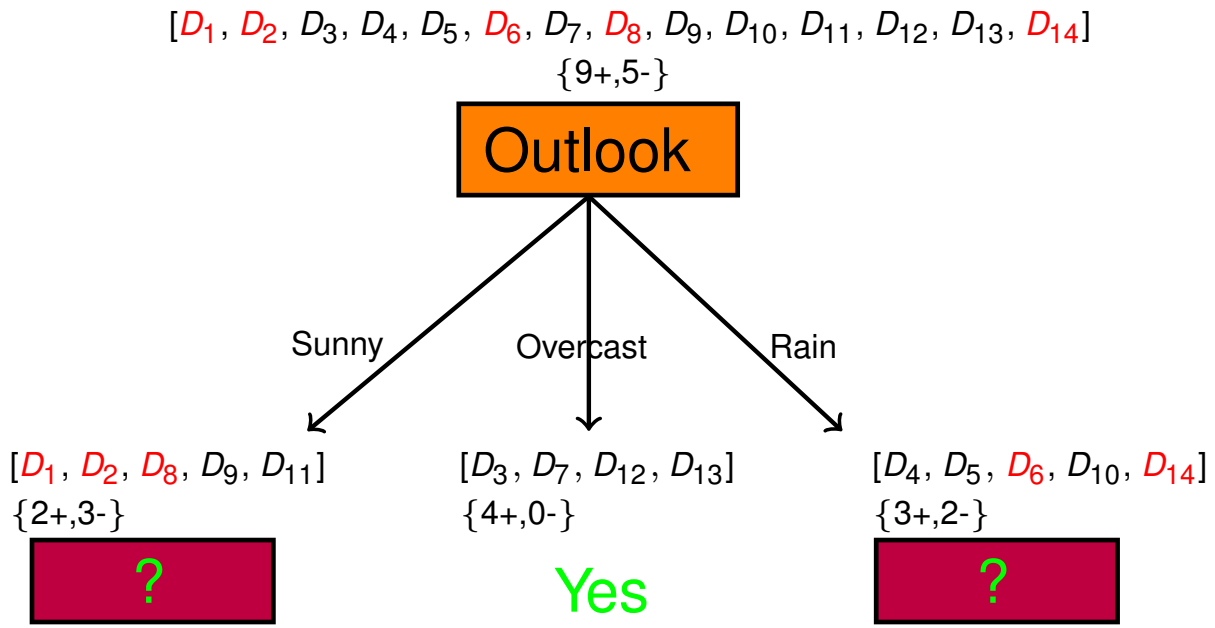
Information Gain and Decision Tree

$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

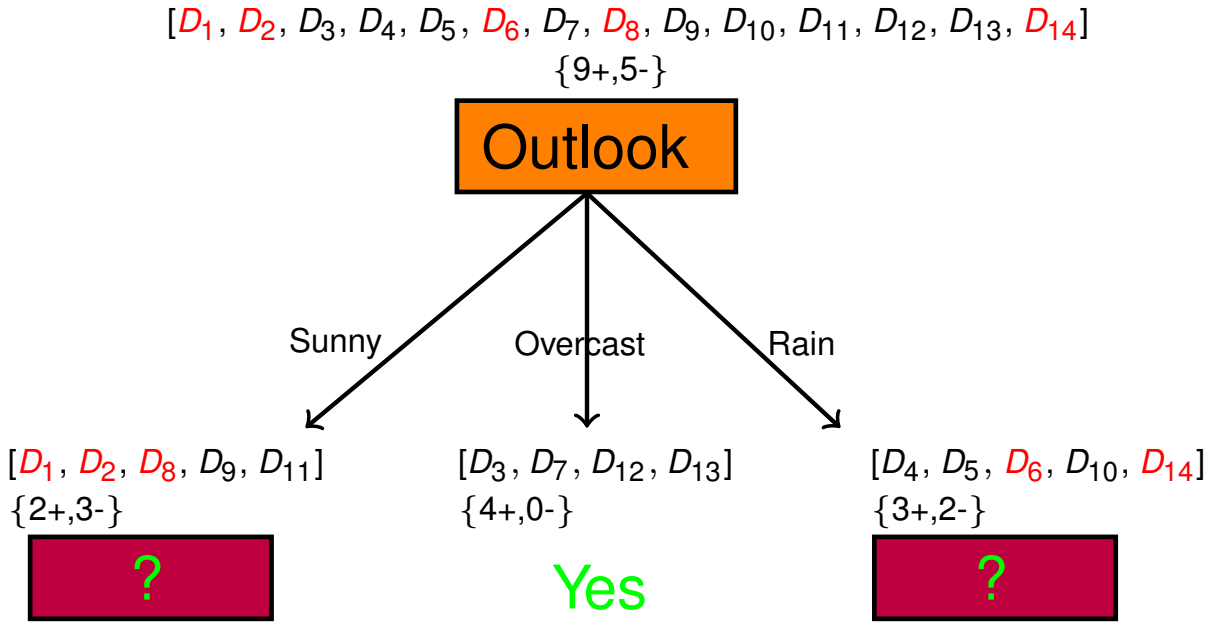


Information Gain and Decision Tree

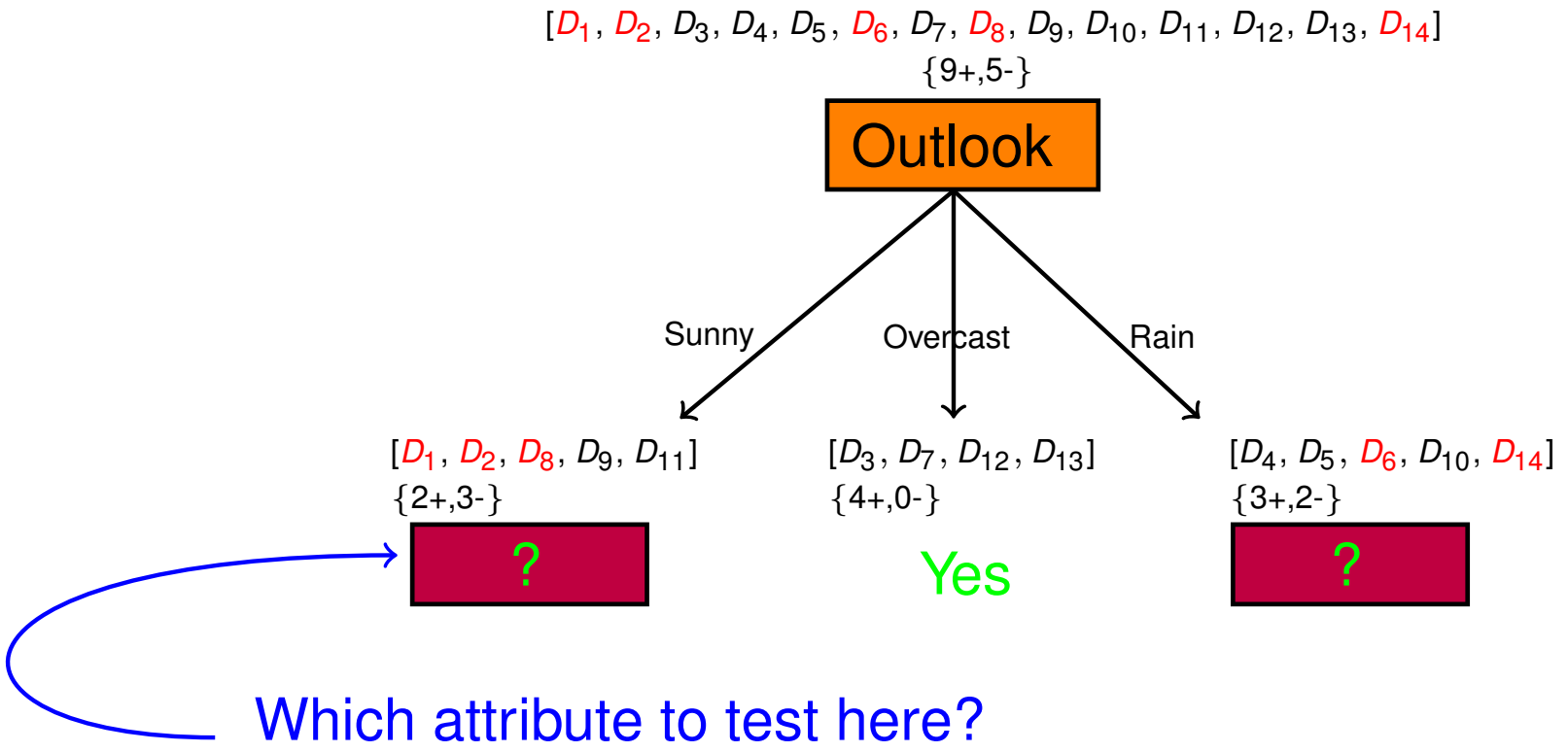
$$
\begin{aligned}
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Outlook) &= 0.246 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$



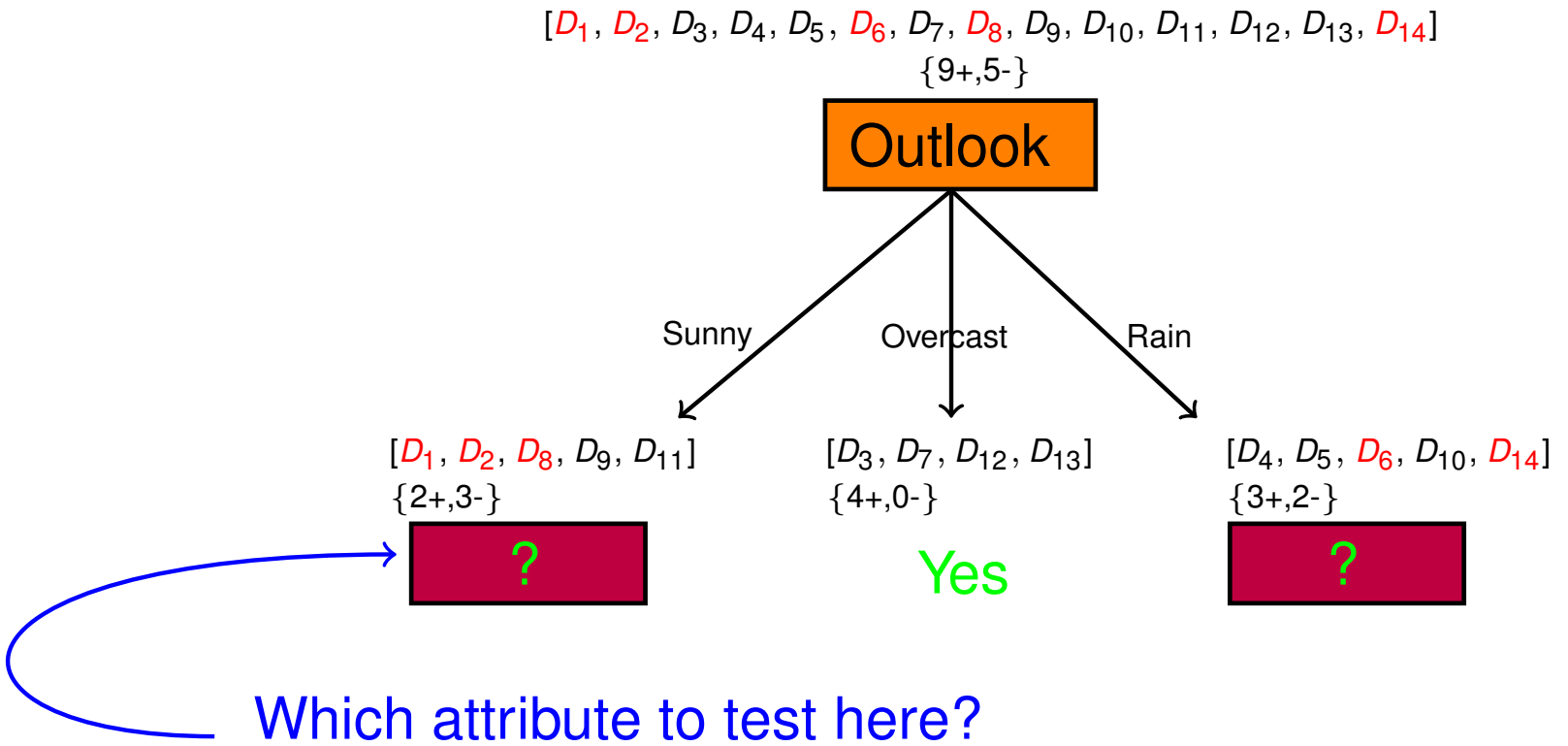
Recursively apply the same



Recursively apply the same

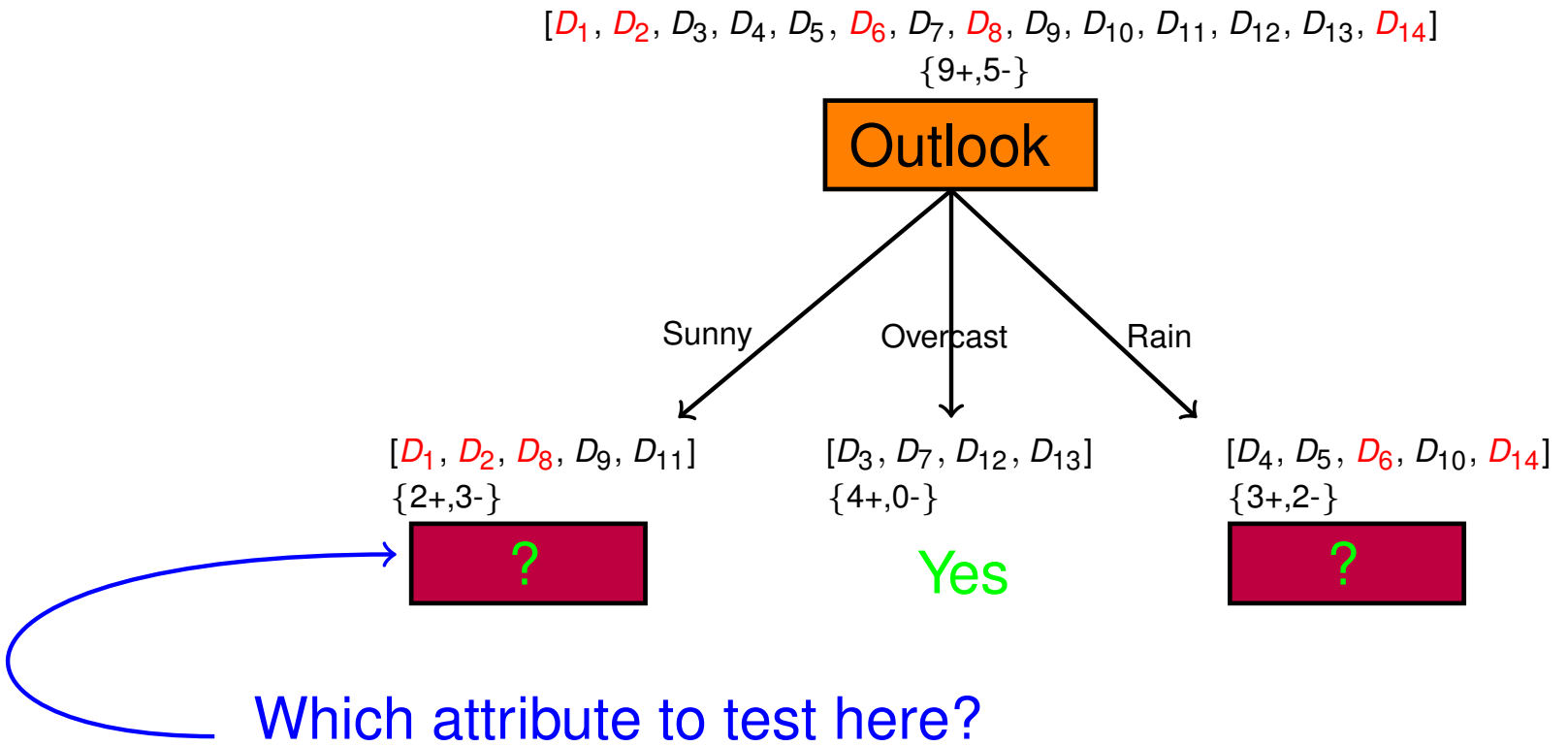


Recursively apply the same



S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

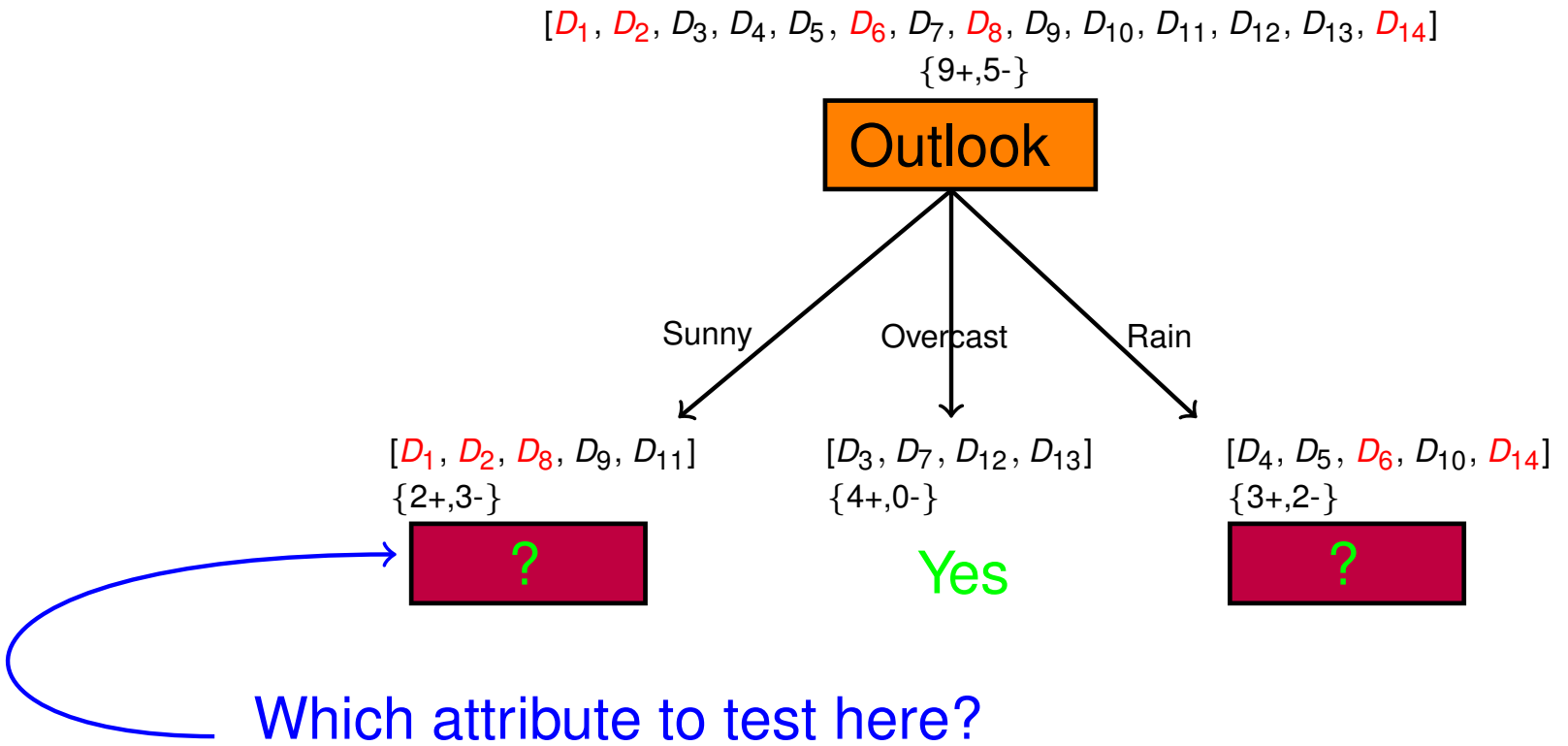
Recursively apply the same



S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Recursively apply the same

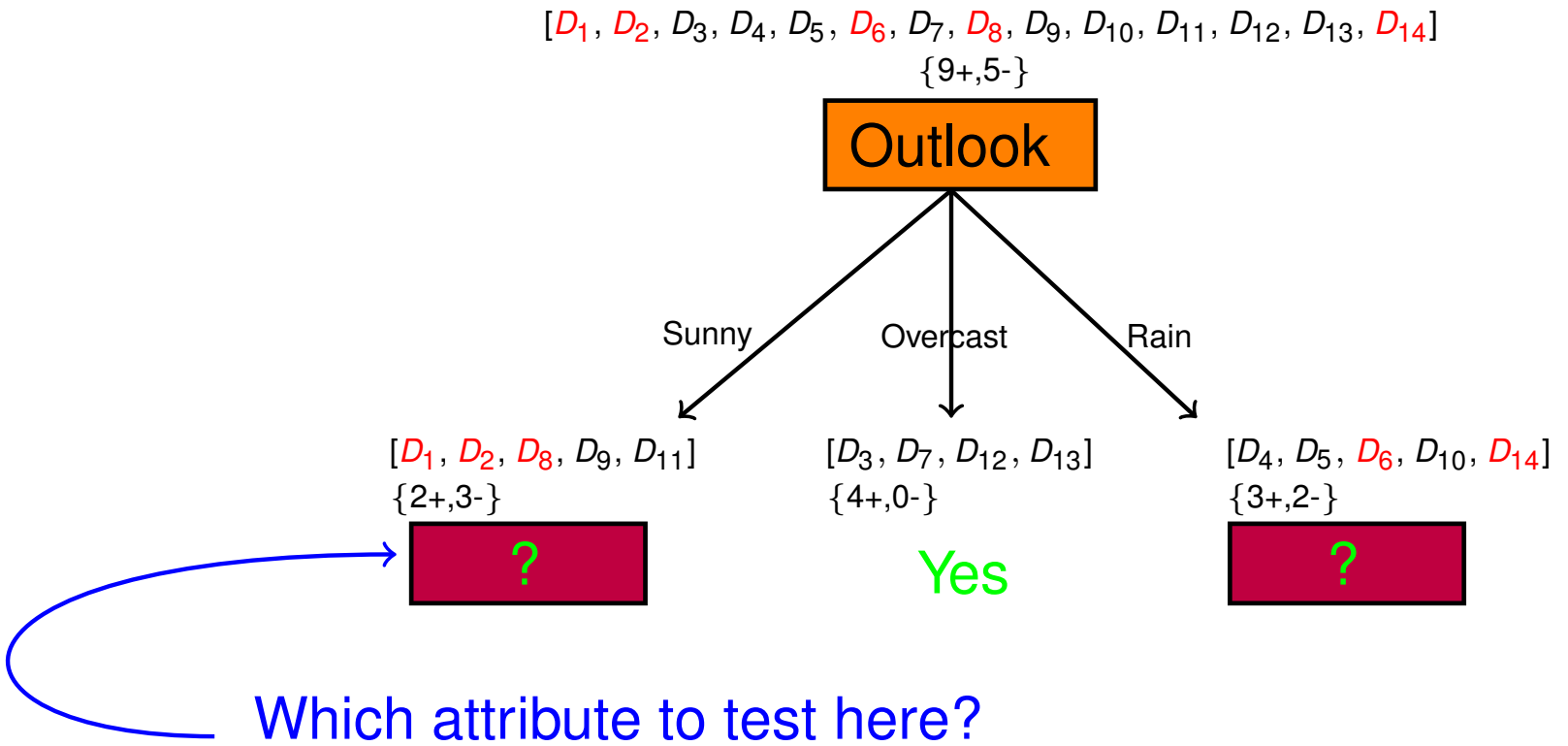


S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Recursively apply the same



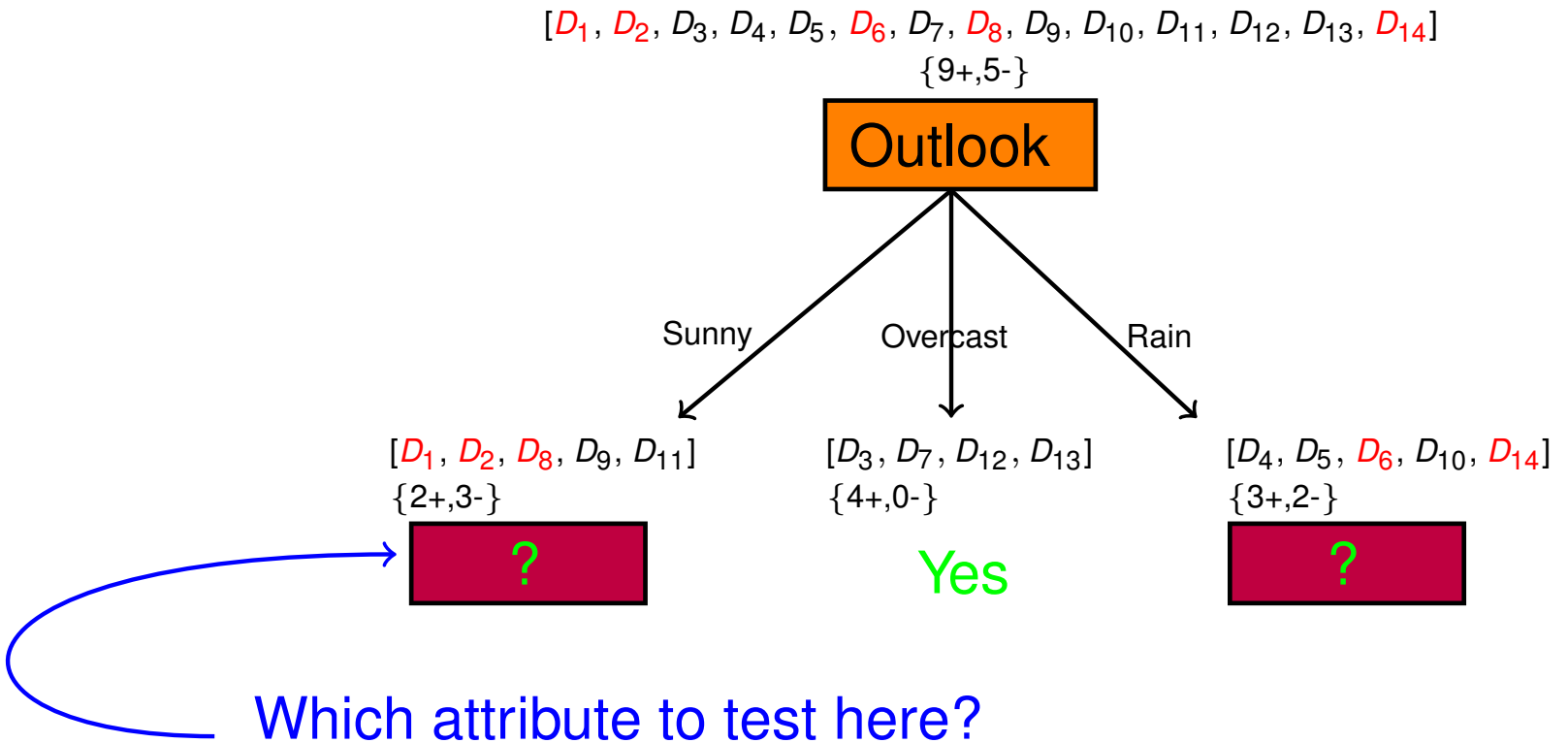
S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Gain(S_{sunny},Wind)$= 0.970 - (2/5)1.0 - (3/5)1.0 = 0.019$

Recursively apply the same



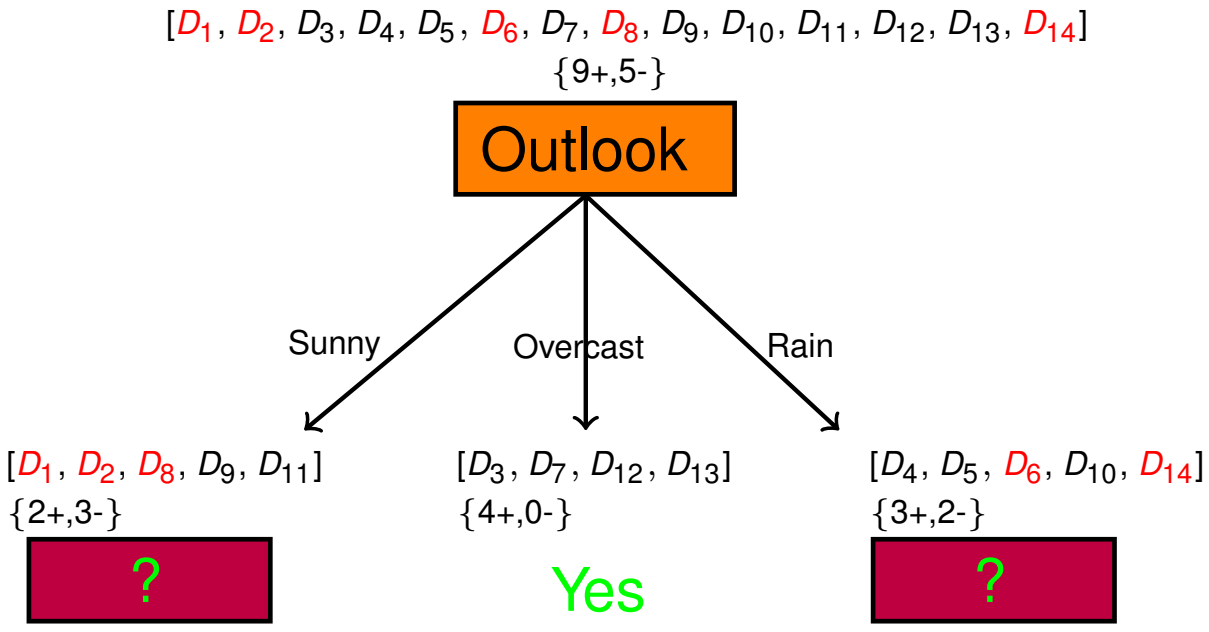
S_{sunny}=[D_1, D_2, D_8, D_9, D_{11}]

Gain(S_{sunny},Humidity) $= 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$

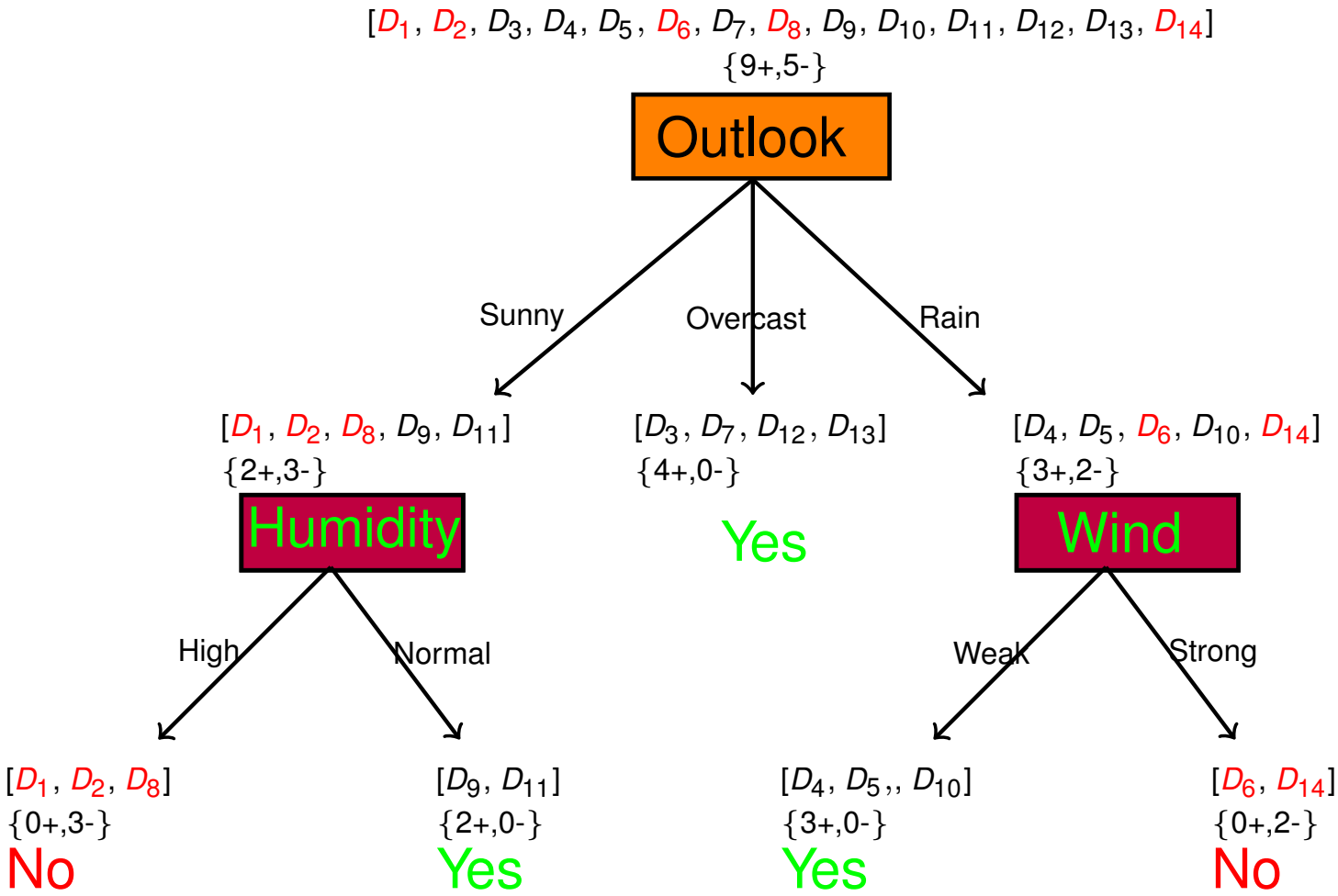
Gain(S_{sunny},Temperature)$= 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.57$

Gain(S_{sunny},Wind)$= 0.970 - (2/5)1.0 - (3/5)1.0 = 0.019$

Recursively apply the same



Recursively apply the same



Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

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

Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

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

What is classification for

$(Outlook = Rain, Humidity = High, Wind = Weak)$

Decision Tree

A method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions

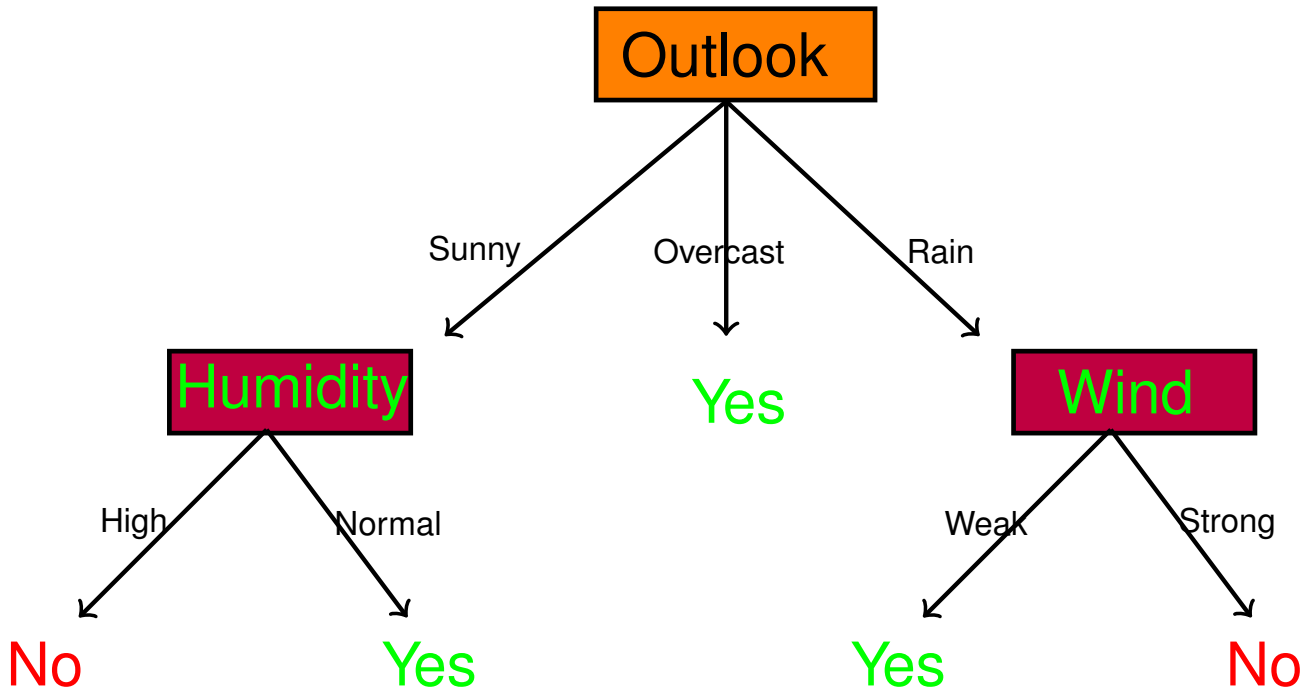
Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak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

What is classification for

$(Outlook = Rain, Humidity = High, Wind = Weak)$

ALERT: (missing value) what is Temperature?

Example



Classification for $(Outlook = Rain, Humidity = High, Wind = Weak)$ is

YES

Iterative-Dichotomiser-3 (ID3) Algorithm By: John Ross Quinlan

Algorithm 1: ID3(Examples,Target_attribute,Attributes)

1 *Examples* are the training data, *Target_attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree. Algorithm returns a decision tree that correctly classify the given example.

2 Create a single-node tree *Root*

3 **IF** *Examples* are all +ve **THEN return** *Root* with label +ve

4 **IF** *Examples* are all -ve **THEN return** *Root* with label -ve

5 **IF** *Attributes* $= \phi$ **THEN return** *Root* with most common *Target_attribute*

6 A $\leftarrow$ attribute from *Attributes* that best classifies *Examples*

7 Decision attribute for *Root* $\leftarrow$ A

8 **foreach** *value* v_i *of A* **do**

9 Add a new tree branch below Root, to test A=v_i

10 Examples$_{v_i}$ $\leftarrow$ subset of *Examples* having value v_i for A

11 **IF** Examples$_{v_i} = \phi$ **THEN** below this branch add a leaf with label = most common value of *Target_attribute* in *Examples*

12 **ELSE** below this branch add subtree ID3(Examples$_{v_i}$, *Target_attribute*, *Attributes*-$\{A\}$)

13 **return** *Root*

Issues Decision Tree

Given a collection of training examples, there could be many decision trees consistent with the examples

- ID3 search strategy
  - selects in favor of shorter trees over longer ones, and
  - selects trees that place the attributes with highest information gain closest to the root

Issues Decision Tree

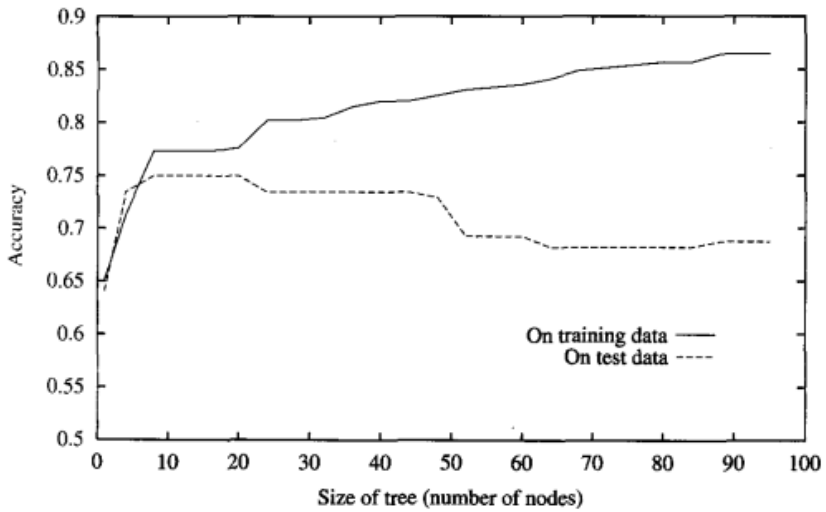
Given a collection of training examples, there could be many decision trees consistent with the examples

- ID3 search strategy
  - selects in favor of shorter trees over longer ones, and
  - selects trees that place the attributes with highest information gain closest to the root
- Issues in decision trees include
  1. how deeply to grow
  2. handling continuous attributes
  3. choosing an appropriate attribute selection measure
  4. missing attribute values
  5. attributes with differing costs, and
  6. improving computational efficiency

Issues in Decision Tree

Overfitting

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, 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.



- This can occur when training examples contain random errors or noise.

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test (such as chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.

Approaches to avoid overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- Allow the tree to overfit the data, and then post-prune the tree

Criterion to determine the correct final tree size include:

- Use a separate set of examples (called validation), distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test (such as chi-square test) to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set.
- Use an explicit measure of the complexity for encoding the training examples (such as Minimum Description Length) and the decision tree, halting growth of the tree when this encoding size is minimized.

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

[2] Leo Breiman, “Random Forests”, ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)

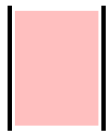
[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



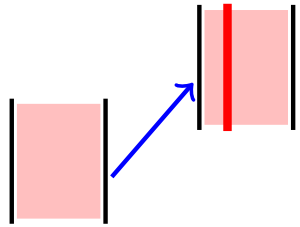
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



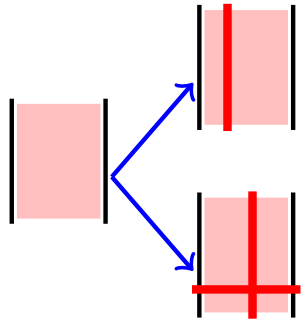
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



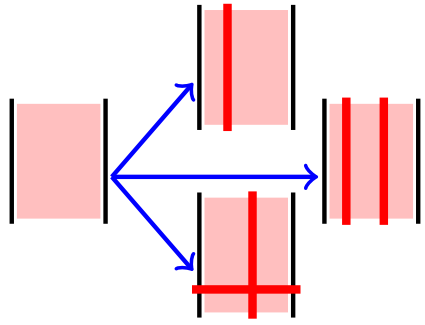
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



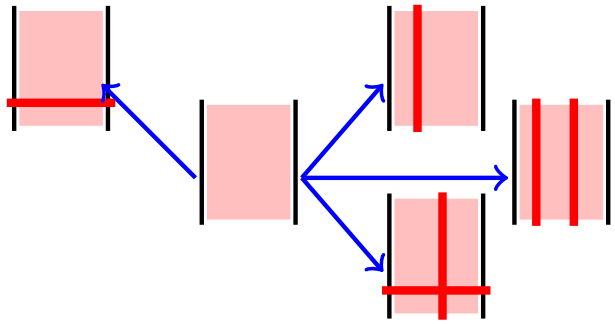
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



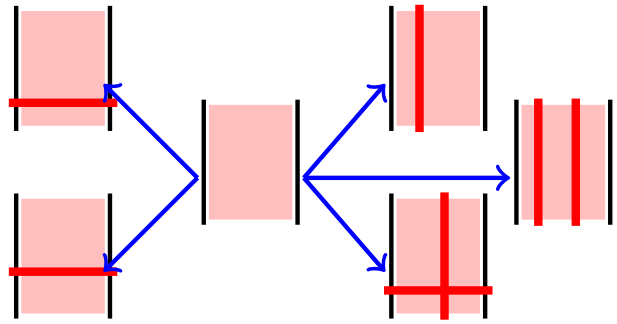
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



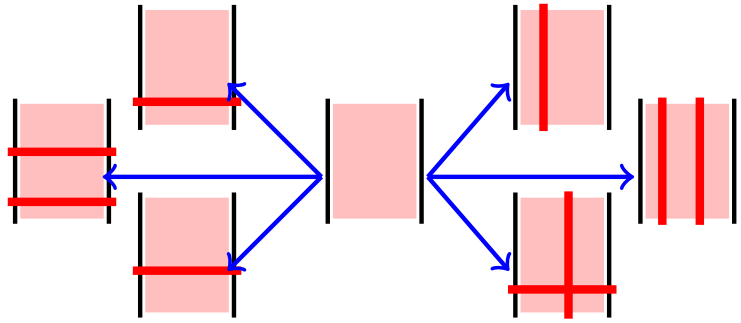
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



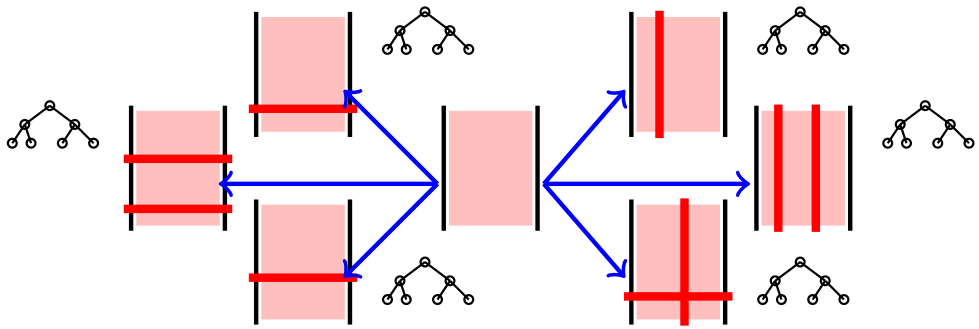
- For every subset, built a decision tree
- To classify new element: **apply voting**

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



- For every subset, built a decision tree

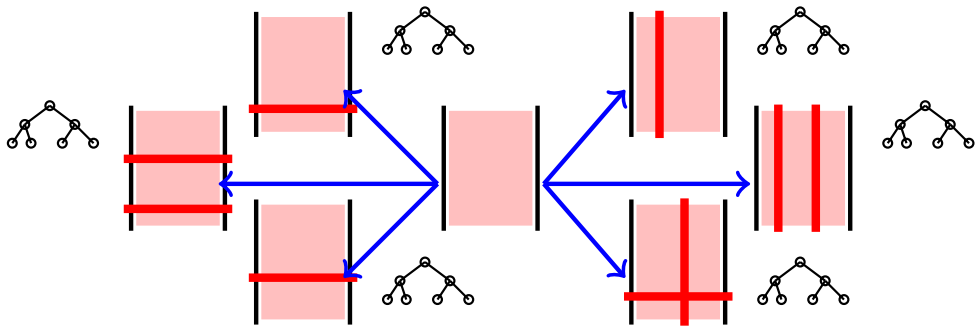
[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Random Forest

Combination of learning models (ensemble of classifiers) increases classification accuracy. Averaging compensates noise. Resulting model has low variance

Random Forest[2] is a large collection of decorrelated decision trees

- Training data is divided into a number of sub-sets (delete row or columns) that may have overlap (or subset of attributes)



- For every subset, built a decision tree
- To classify new element: **apply voting**

[2] Leo Breiman, "Random Forests", ML 45, pp 5-32, 2001

Thank You!

Thank you very much for your attention!

Queries ?

(Reference[3])

[3] 1) Book - *AIMA*, ch-14, Russell and Norvig. 2) Book - *Bayeian Reasoning and Machine Learning*, ch-04, David Barber.