Decision Trees

Nipun Batra
Jan 9, 2019

Need For Interpretability

How to Maintain Trust in AI

Beyond developing initial trust, however, creators of AI also must work to maintain that trust. Siau and Wang suggest seven ways of "developing continuous trust" beyond the initial phases of product development:

- **Usability and reliability.** AI "should be designed to operate easily and intuitively," Siau and Wang write. "There should be no unexpected downtime or crashes."
- **Collaboration and communication.** AI developers want to create systems that perform autonomously, without human involvement. Developers must focus on creating AI applications that smoothly and easily collaborate and communicate with humans.
- **Sociability and bonding.** Building social activities into AI applications is one way to strengthen trust. A robotic dog that can recognize its owner and show affection is one example, Siau and Wang write.
- **Security and privacy protection.** AI applications rely on large data sets, so ensuring privacy and security will be crucial to establishing trust in the applications.
- **Interpretability.** Just as transparency is instrumental in building initial trust, interpretability – or the ability for a machine to explain its conclusions or actions – will help sustain trust.

Training Data

Day	Outlook	Temp	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	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

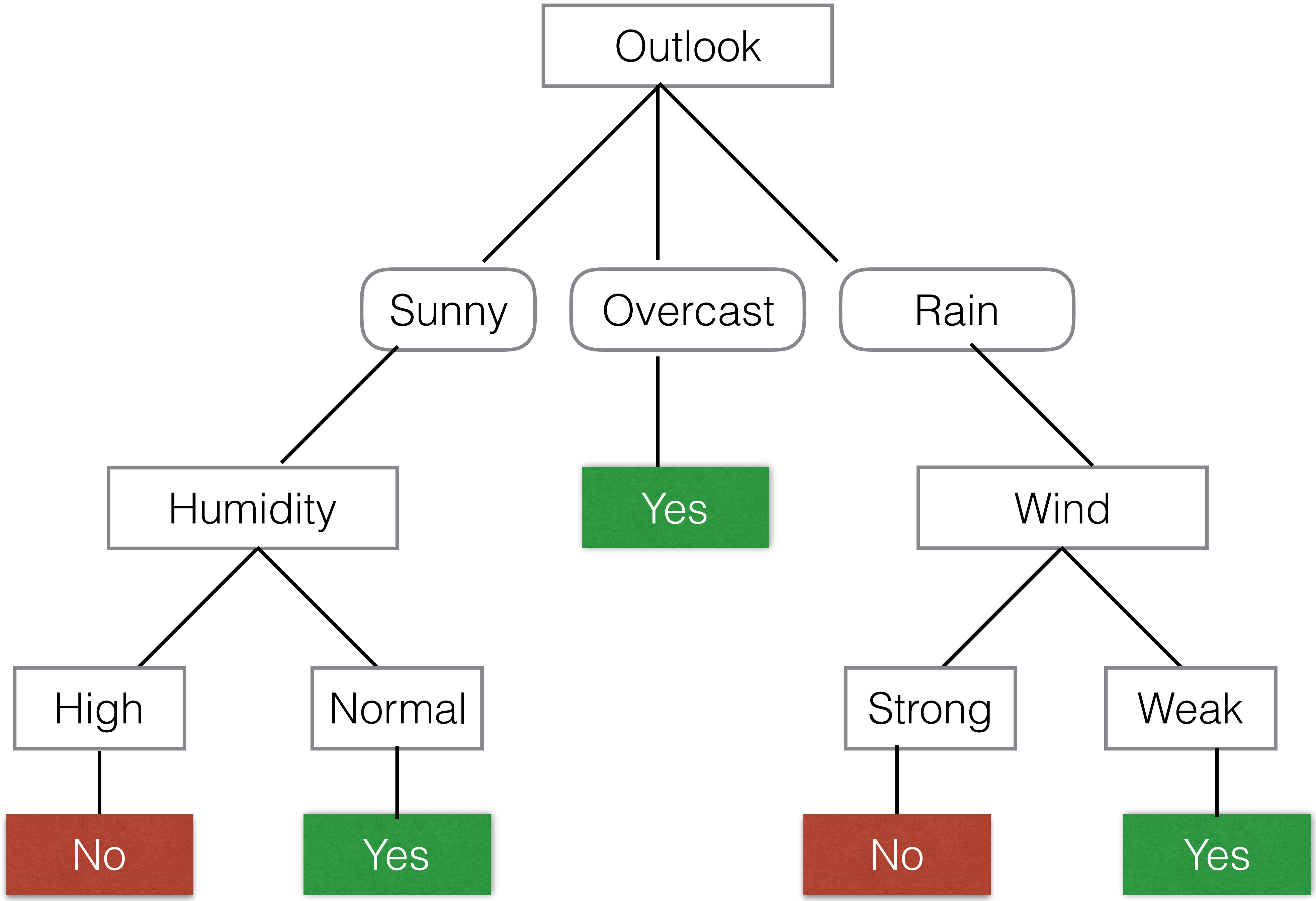
Example borrowed from Tom Mitchell's text book

Training Data

Day	Outlook	Temp	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	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example borrowed from Tom Mitchell's text book

Decisions - Will I Play Tennis?



Example borrowed from Tom Mitchell's text book

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
    1. A <- attribute from Attributes which **best** classifies Examples

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
   1. A <- attribute from Attributes which **best** classifies Examples
   2. Root <- A

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
    1. A <- attribute from Attributes which **best** classifies Examples
    2. Root <- A
    3. For each value (v) of A

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
    1. A <- attribute from Attributes which **best** classifies Examples
    2. Root <- A
    3. For each value (v) of A
        1. Add new tree branch : A = v

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
   1. A <- attribute from Attributes which **best** classifies Examples
   2. Root <- A
   3. For each value (v) of A
      1. Add new tree branch : A = v
      2. Examples_v : subset of examples that A = v

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
    1. A <- attribute from Attributes which **best** classifies Examples
    2. Root <- A
    3. For each value (v) of A
        1. Add new tree branch : A = v
        2. Examples_v : subset of examples that A = v
        3. If Examples_v is empty: add leaf with label = most common value of Target_Attribute

Greedy Algorithm

ID3 (Examples, Target_Attribute, Attributes)

1. Create a root node for tree
2. If all examples are +/-, return root with label = +/-
3. If attributes = empty, return root with most common value of Target_Attribute in Examples
4. Begin
    1. A <- attribute from Attributes which **best** classifies Examples
    2. Root <- A
    3. For each value (v) of A
        1. Add new tree branch : A = v
        2. Examples_v : subset of examples that A = v
        3. If Examples_v is empty: add leaf with label = most common value of Target_Attribute
        4. Else: ID3 (Examples_v, Target_attribute, Attributes - {A})

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

PlayTennis
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

PlayTennis
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

5 No, 9 Yes

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

PlayTennis
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

Entropy

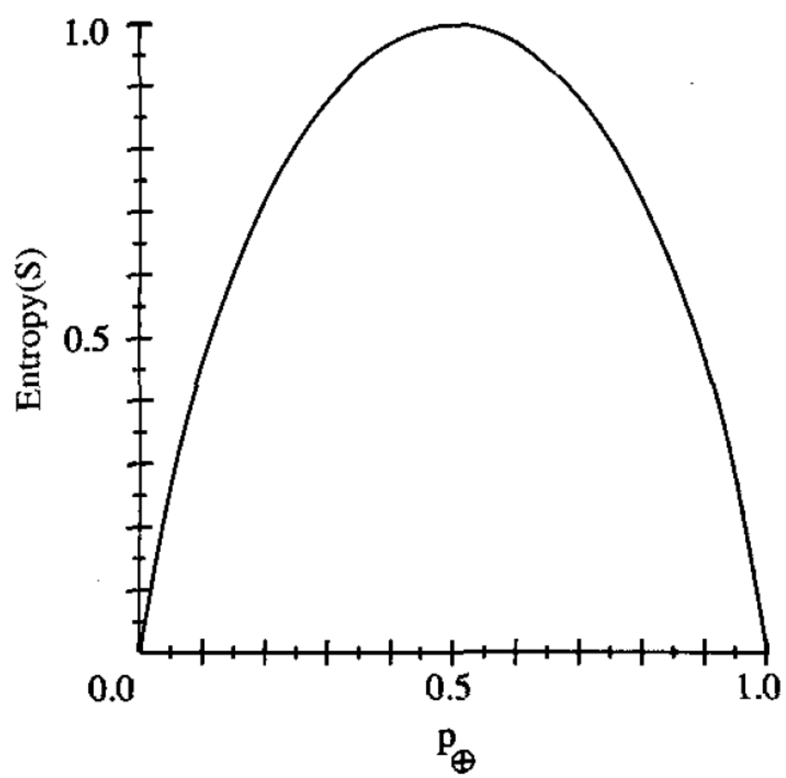
Entropy: Statistical measure to characterize the (im)purity of examples

$$
\begin{aligned}
\text{Entropy} &= - p_{No} \log_2 p_{No} - p_{Yes} \log_2 p_{Yes} \\
&= -(5/14) \log_2(5/14) - (9/14)\log_2(9/14) \\
&= 0.94
\end{aligned}
$$

Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

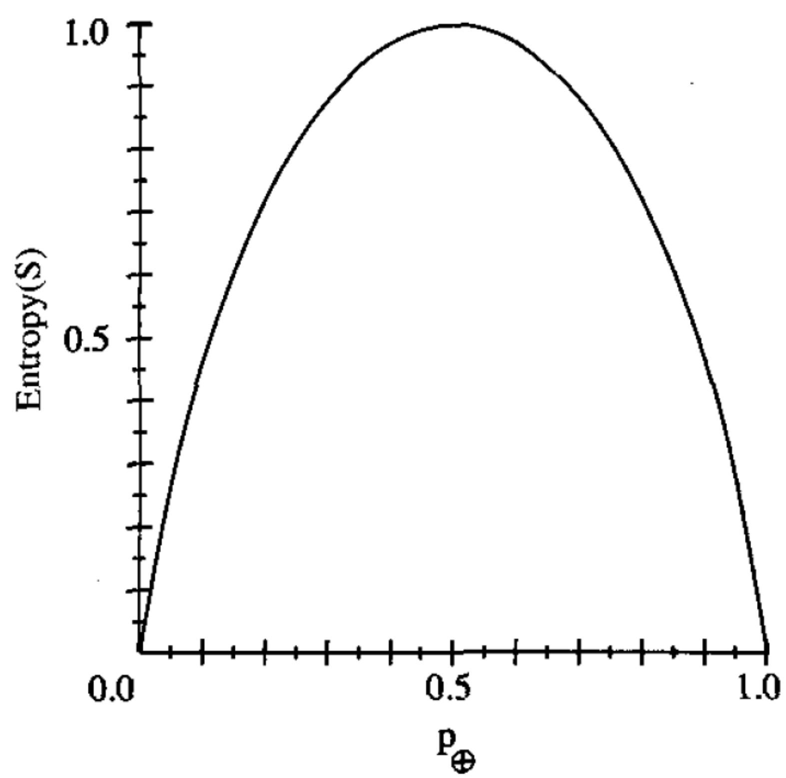
$$\begin{aligned}\text{Entropy} &= - p_{No} \log_2 p_{No} - p_{Yes} \log_2 p_{Yes} \\ &= -(5/14) \log_2(5/14) - (9/14)\log_2(9/14) \\ &= 0.94\end{aligned}$$



Entropy

Entropy: Statistical measure to characterize the (im)purity of examples

$$\text{Entropy} = - p_{No} \log_2 p_{No} - p_{Yes} \log_2 p_{Yes}$$
$$= -(5/14) \log_2(5/14) - (9/14)\log_2(9/14)$$
$$= 0.94$$



Avg. # of bits to transmit

Information Gain

Information Gain: Reduction in entropy

Information Gain

Information Gain: Reduction in entropy

By partitioning examples (S) on attribute A

Information Gain

By partitioning examples (S) on attribute A

Information Gain

Information Gain

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

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong
- S = [9+, 5-]

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong
- S = [9+, 5-]
- S_{Weak} = [6+, 2-]

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong
- S = [9+, 5-]
- S_{Weak} = [6+, 2-]
- S_{Strong} = [3+, 3-]

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong
- S = [9+, 5-]
- S_{Weak} = [6+, 2-]
- S_{Strong} = [3+, 3-]
- Gain (S, Wind) = Entropy (S) - (8/14)*Entropy (S_{Weak}) - (6/14)*Entropy(S_{Strong})

Information Gain

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

Wind	PlayTennis
Weak	No
Strong	No
Weak	Yes
Weak	Yes
Weak	Yes
Strong	No
Strong	Yes
Weak	No
Weak	Yes
Weak	Yes
Strong	Yes
Strong	Yes
Weak	Yes
Strong	No

- A = Wind
- Values (Wind) = Weak, Strong
- S = [9+, 5-]
- S_{Weak} = [6+, 2-]
- S_{Strong} = [3+, 3-]
- Gain (S, Wind) = Entropy (S) - (8/14)*Entropy (S_{Weak}) - (6/14)*Entropy(S_{Strong})

= 0.048

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]
- S_{Sunny} = [2+, 3-]

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- $S = [9+, 5-]$
- $S_{Sunny} = [2+, 3-]$
- $S_{Overcast} = [4+, 0-]$

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]
- S_{Sunny} = [2+, 3-]
- $S_{Overcast}$ = [4+, 0-]
- S_{Rain} = [3+, 2-]

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]
- S_{Sunny} = [2+, 3-]
- $S_{Overcast}$ = [4+, 0-]
- S_{Rain} = [3+, 2-]
- Gain (S, Outlook) = Entropy (S) - (5/14)*Entropy (S_{Sunny}) - (4/14)*Entropy($S_{Overcast}$)-

Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]
- S_{Sunny} = [2+, 3-]
- $S_{Overcast}$ = [4+, 0-]
- S_{Rain} = [3+, 2-]
- Gain (S, Outlook) = Entropy (S) - (5/14)*Entropy (S_{Sunny}) - (4/14)*Entropy($S_{Overcast}$)- (5/14)*Entropy(S_{Rain})

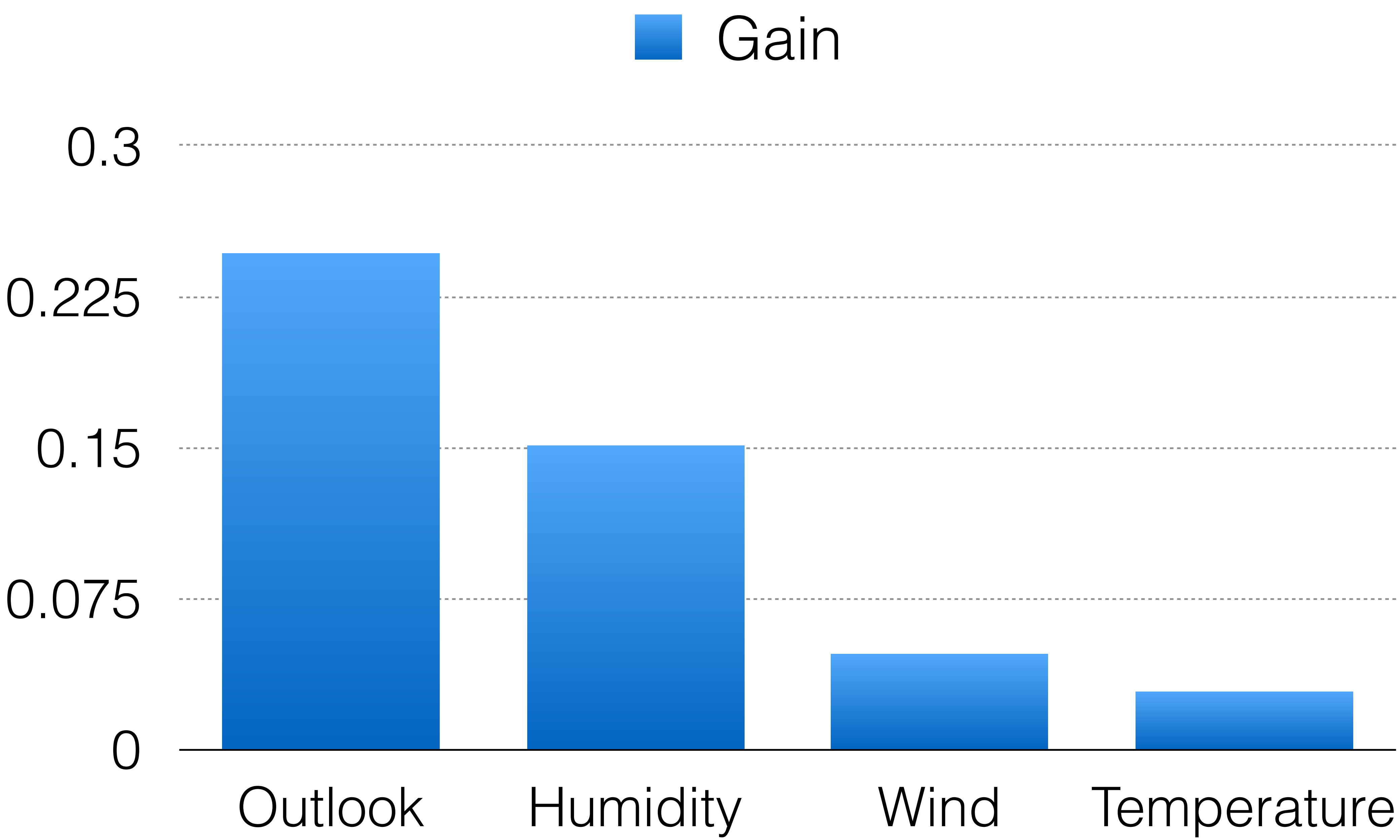
Information Gain

Outlook	PlayTennis
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

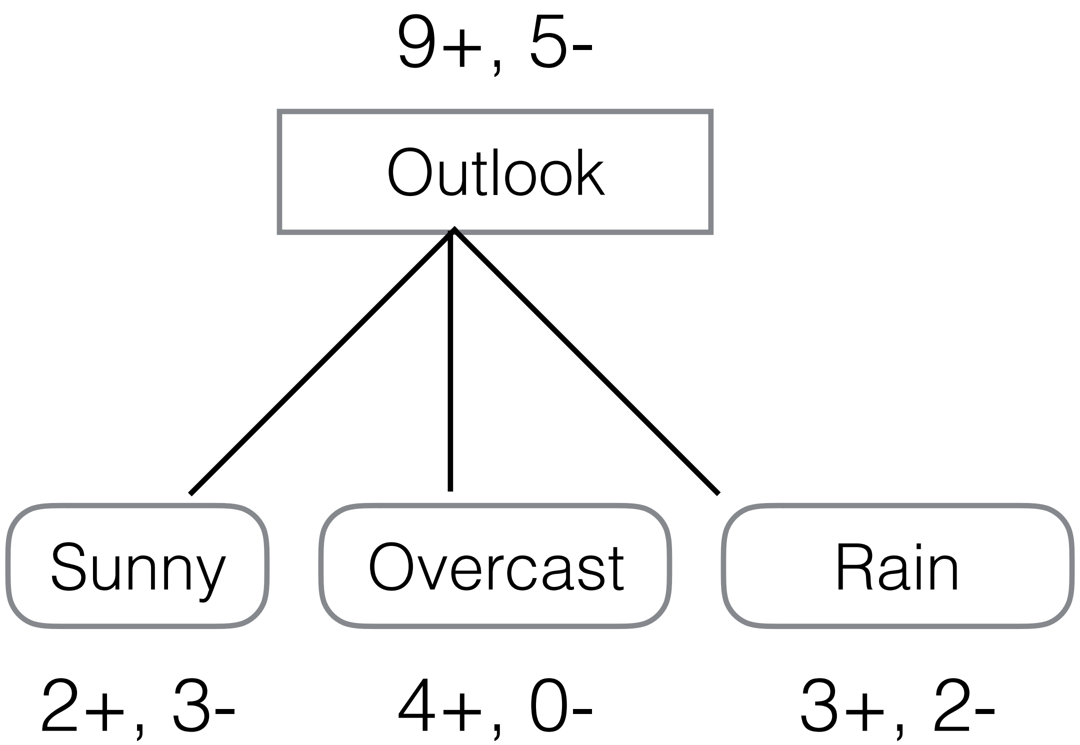
- A = Outlook
- Values (Outlook) = Sunny, Overcast, Rain
- S = [9+, 5-]
- S_{Sunny} = [2+, 3-]
- $S_{Overcast}$ = [4+, 0-]
- S_{Rain} = [3+, 2-]
- Gain (S, Outlook) = Entropy (S) - (5/14)*Entropy (S_{Sunny}) - (4/14)*Entropy($S_{Overcast}$)- (5/14)*Entropy(S_{Rain})

= 0.246

Information Gain

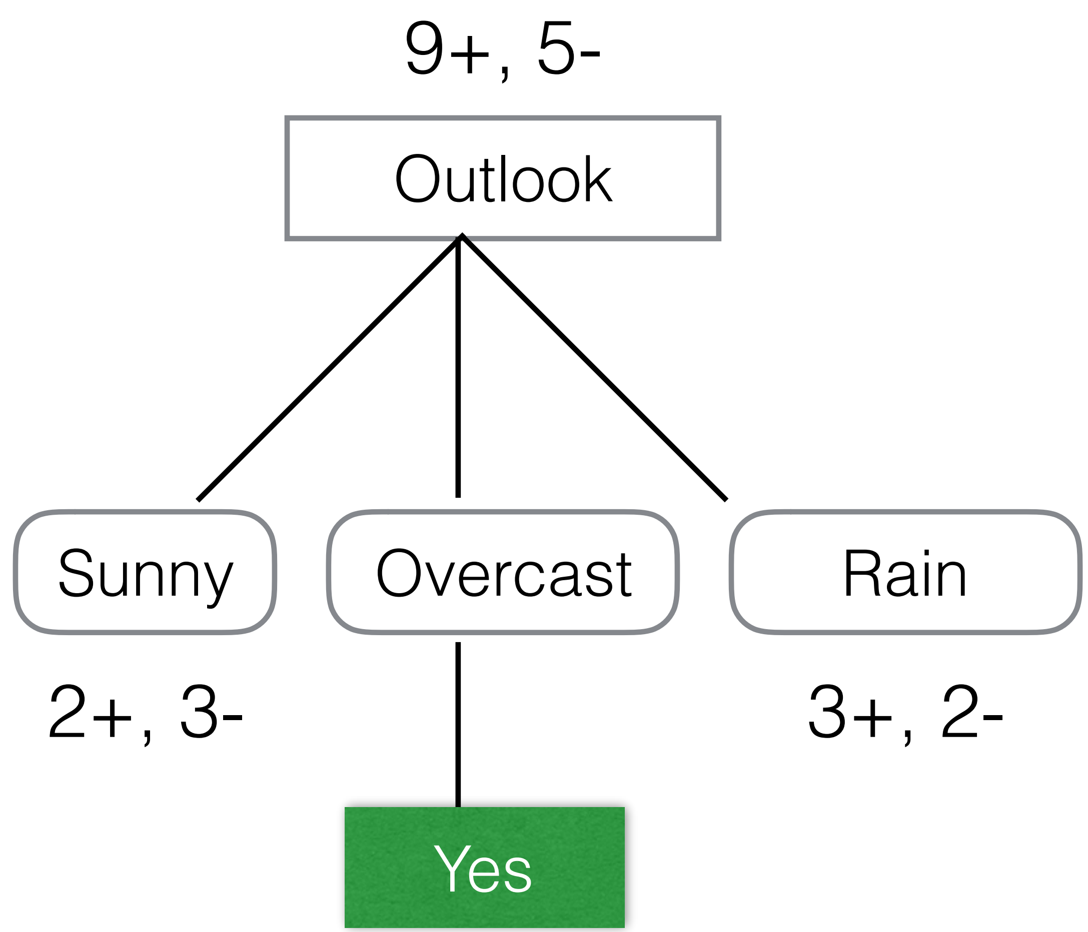


Worked Out Example



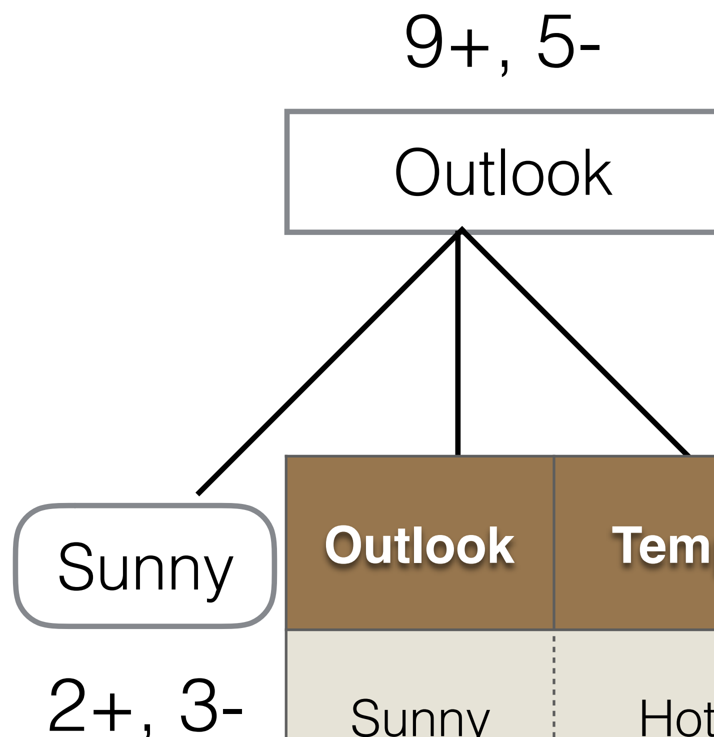
Example borrowed from Tom Mitchell's text book

Worked Out Example



Example borrowed from Tom Mitchell's text book

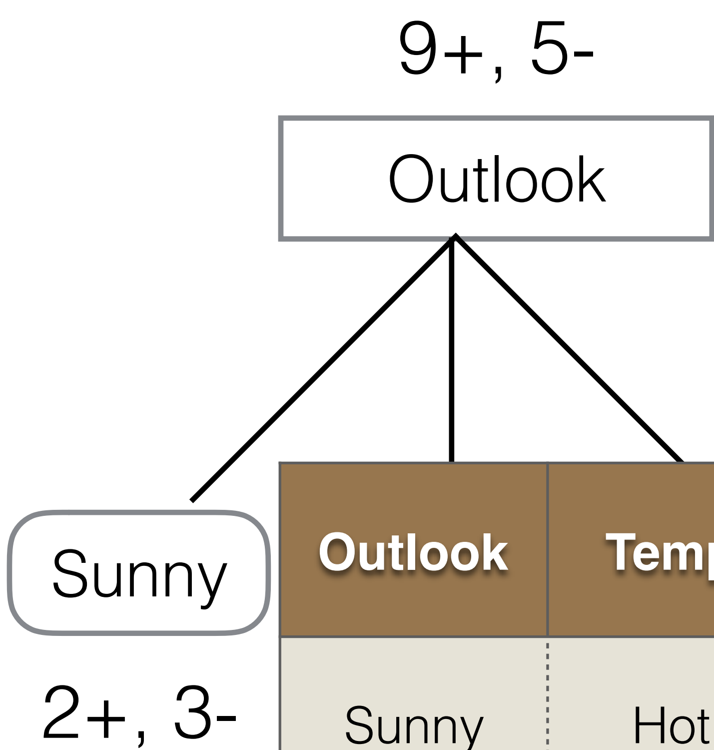
Worked Out Example



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

Example borrowed from Tom Mitchell's text book

Worked Out Example



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

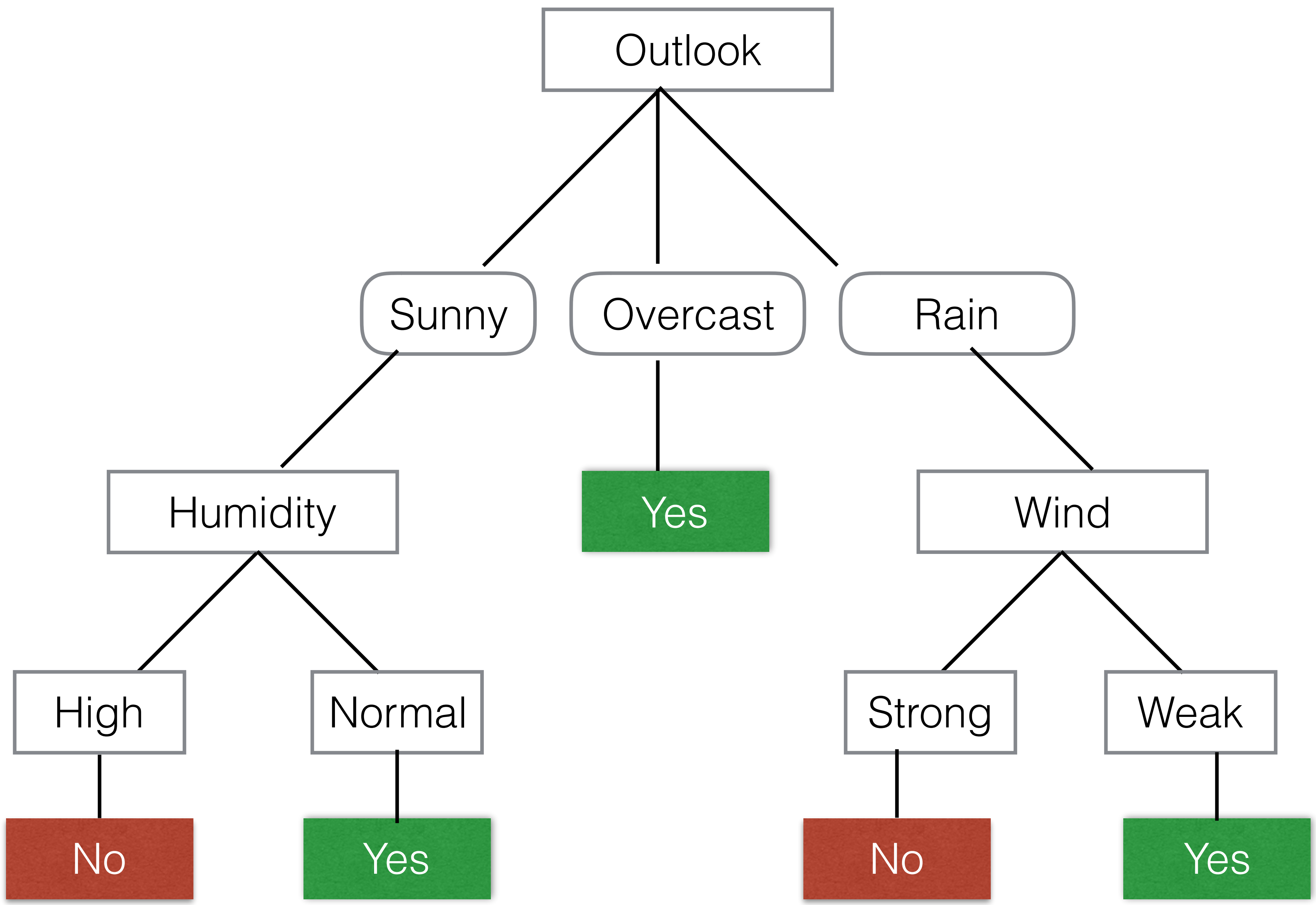
S' = S_outlook is sunny

Entropy (S') =

$-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)$

Example borrowed from Tom Mitchell's text book

Worked Out Example



Example borrowed from Tom Mitchell's text book

Case II: Regression Task on Discrete Features (using decrease in stdev)

Day	Outlook	Temp	Humidity	Wind	Minutes Played
D1	Sunny	Hot	High	Weak	20
D2	Sunny	Hot	High	Strong	24
D3	Overcast	Hot	High	Weak	40
D4	Rain	Mild	High	Weak	50
D5	Rain	Cool	Normal	Weak	60
D6	Rain	Cool	Normal	Strong	10
D7	Overcast	Cool	Normal	Strong	4
D8	Sunny	Mild	High	Weak	10
D9	Sunny	Cool	Normal	Weak	60
D10	Rain	Mild	Normal	Weak	40
D11	Sunny	Mild	High	Strong	45
D12	Overcast	Mild	High	Strong	40
D13	Overcast	Hot	Normal	Weak	35
D14	Rain	Mild	High	Strong	20

Material heavily inspired from: https://www.saedsayad.com/decision_tree_reg.htm

Case II: Regression Task on Discrete Features (using decrease in stdev)

Standard Deviation: Statistical measure to characterize the (im)purity/“variation” of examples

Minutes Played
20
24
40
50
60
10
4
10
60
40
45
40
35
20

Case II: Regression Task on Discrete Features (using decrease in stdev)

Standard Deviation: Statistical measure to characterize the (im)purity/“variation” of examples

Minutes Played
20
24
40
50
60
10
4
10
60
40
45
40
35
20

$$\text{Mean} = (20 + 24 + \ldots + 20)/14 = 32.7$$

Case II: Regression Task on Discrete Features (using decrease in stdev)

Standard Deviation: Statistical measure to characterize the (im)purity/“variation” of examples

Minutes Played
20
24
40
50
60
10
4
10
60
40
45
40
35
20

Mean = (20 + 24 +…+20)/14
= 32.7

STDEV(S) = 18.3

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Day	Wind	Minutes
D2	Strong	24
D6	Strong	10
D7	Strong	4
D11	Strong	45
D12	Strong	40
D14	Strong	20

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Day	Wind	Minutes
D2	Strong	24
D6	Strong	10
D7	Strong	4
D11	Strong	45
D12	Strong	40
D14	Strong	20

STDEV= 16.2

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Day	Wind	Minutes
D2	Strong	24
D6	Strong	10
D7	Strong	4
D11	Strong	45
D12	Strong	40
D14	Strong	20

STDEV= 16.2

Samples = 6

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Day	Wind	Minutes
D2	Strong	24
D6	Strong	10
D7	Strong	4
D11	Strong	45
D12	Strong	40
D14	Strong	20

STDEV= 16.2

Samples = 6

Weighted STDEV = (6/14)*16.2 = 6.9

Case II: Regression Task on Discrete Features (using decrease in stdev)

STDEV(S) = 18.3

Day	Wind	Minutes
D1	Weak	20
D2	Strong	24
D3	Weak	40
D4	Weak	50
D5	Weak	60
D6	Strong	10
D7	Strong	4
D8	Weak	10
D9	Weak	60
D10	Weak	40
D11	Strong	45
D12	Strong	40
D13	Weak	35
D14	Strong	20

Day	Wind	Minutes
D1	Weak	20
D3	Weak	40
D4	Weak	50
D5	Weak	60
D8	Weak	10
D9	Weak	60
D10	Weak	40
D13	Weak	35

STDEV= 17.8

Samples = 8

Weighted STDEV = (8/14)*17.8 = 10.2

Day	Wind	Minutes
D2	Strong	24
D6	Strong	10
D7	Strong	4
D11	Strong	45
D12	Strong	40
D14	Strong	20

STDEV= 16.2

Samples = 6

Weighted STDEV = (6/14)*16.2 = 6.9

GAIN(S, Wind) = 18.3 - (10.2 + 6.9) =1.2

Case II: Regression Task on Discrete Features (using decrease in stdev)

GAIN(S, Wind) = 18.3 - (10.2 + 6.9) =1.2

GAIN(S, Temp) = 18.3 - 18.1 = 0.2

GAIN(S, Humidity) = 18.3 - 18.5 = -0.4

GAIN(S, Outlook) = 18.3 - 19.6 = -1.3

Case II: Regression Task on Discrete Features (using decrease in stdev)

GAIN(S, Wind) = 18.3 - (10.2 + 6.9) =1.2

GAIN(S, Temp) = 18.3 - 18.1 = 0.2

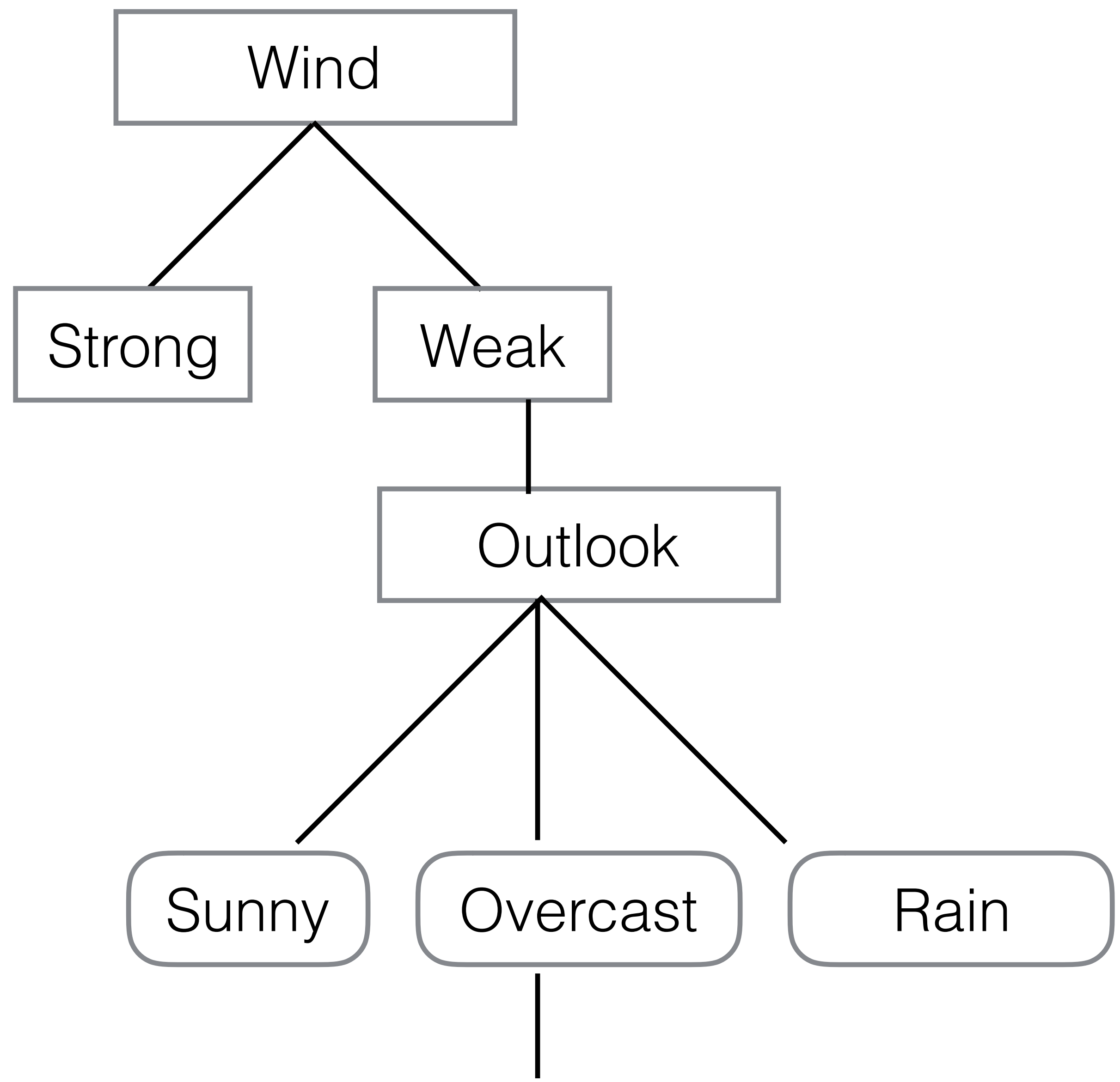
GAIN(S, Humidity) = 18.3 - 18.5 = -0.4

GAIN(S, Outlook) = 18.3 - 19.6 = -1.3

- Wind is the root node
- Recursively use the same procedure to find the tree …

Case II: Regression Task on Discrete Features (using decrease in stdev)

Assume a tree like this is learnt …

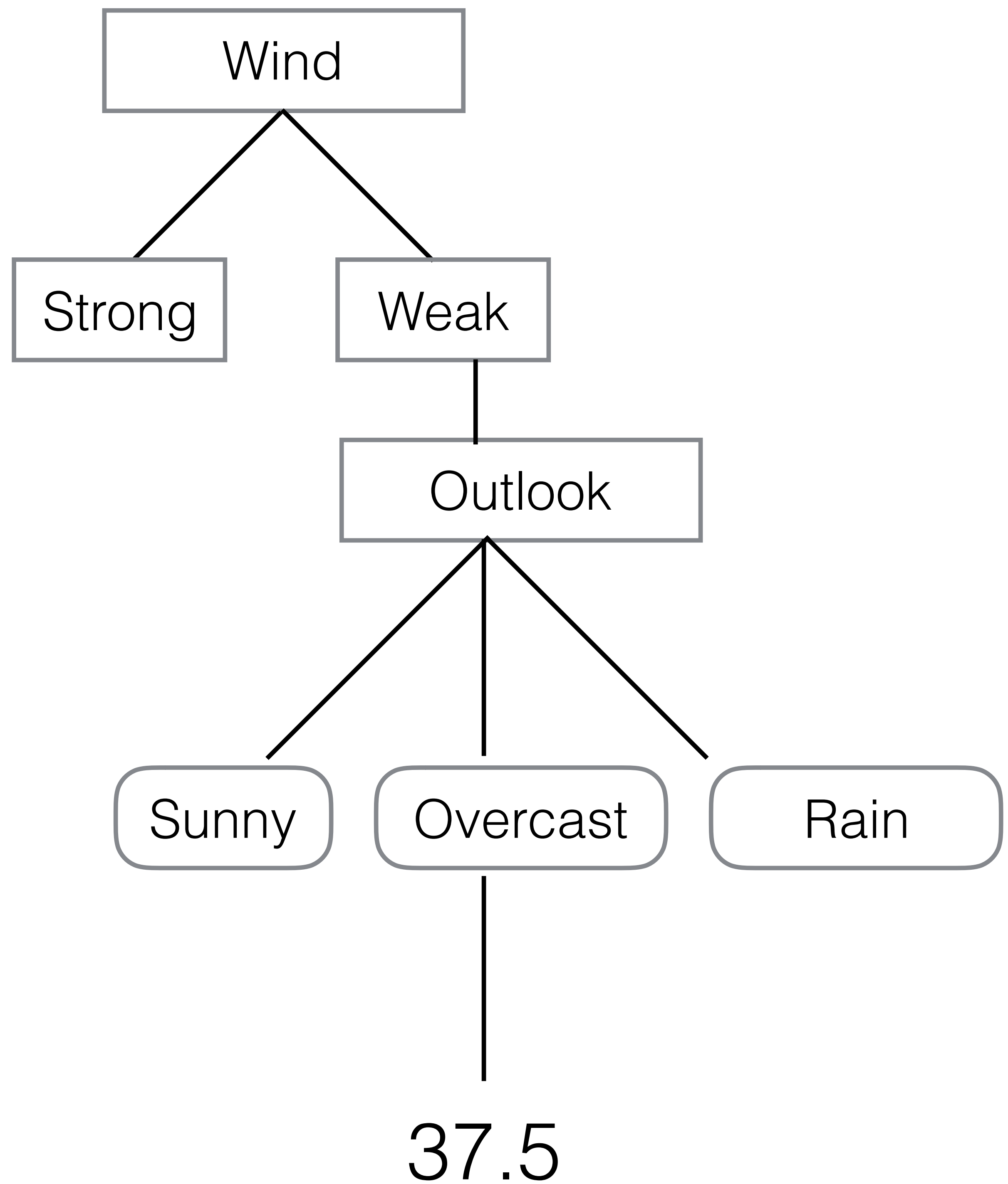


	Day	Outlook	Temp	Humidity	Wind	Minutes Played
2	D3	Overcast	Hot	High	Weak	40
12	D13	Overcast	Hot	Normal	Weak	35

Case II: Regression Task on Discrete Features (using decrease in stdev)

Method 1

Mins Played=(40+35)/2



Case III: Classification Task on Continuous Features

Day	Temperature	PlayTennis
D1	40	No
D2	48	No
D3	60	Yes
D4	72	Yes
D5	80	Yes
D6	90	No

Case III: Classification Task on Continuous Features

Day	Temperature	PlayTennis
D1	40	No
D2	48	No
D3	60	Yes
D4	72	Yes
D5	80	Yes
D6	90	No

Case III: Classification Task on Continuous Features

Day	Temperature	PlayTennis
D1	40	No
D2	48	No
D3	60	Yes
D4	72	Yes
D5	80	Yes
D6	90	No

Temperature > (48+60)/2

Case III: Classification Task on Continuous Features

Day	Temperature	PlayTennis
D1	40	No
D2	48	No
D3	60	Yes
D4	72	Yes
D5	80	Yes
D6	90	No

Temperature > (48+60)/2

Case III: Classification Task on Continuous Features

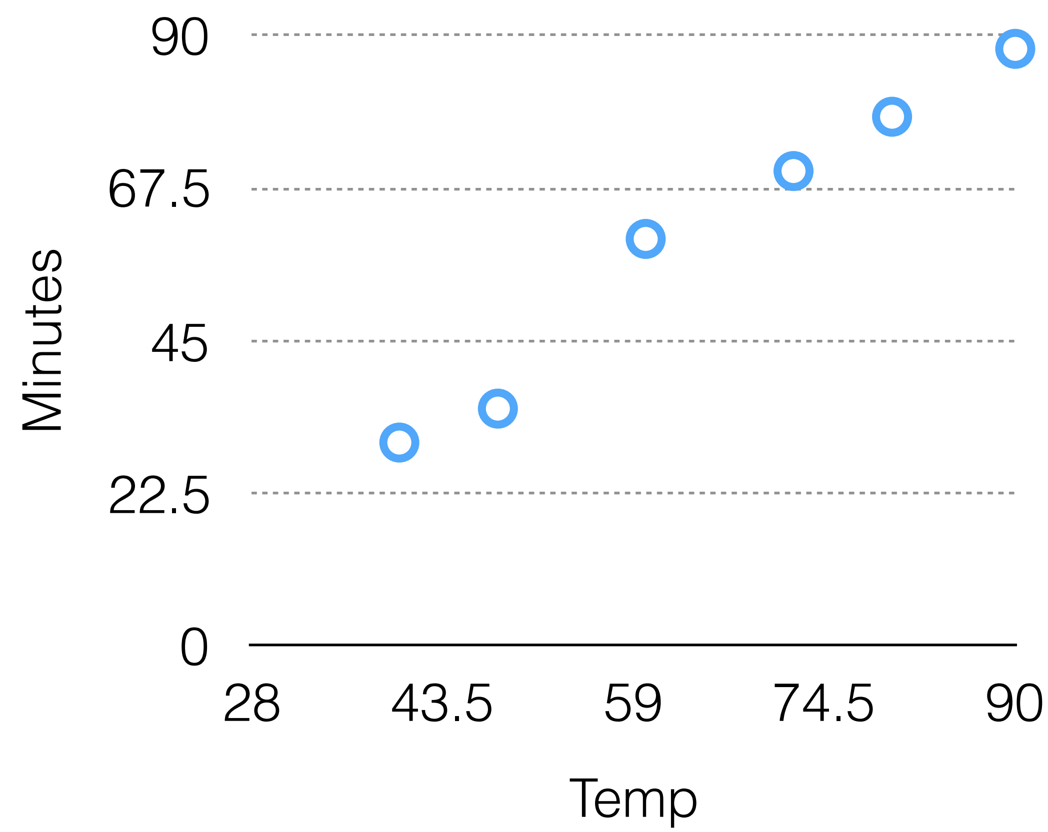
Day	Temperature	PlayTennis
D1	40	No
D2	48	No
D3	60	Yes
D4	72	Yes
D5	80	Yes
D6	90	No

Temperature > (48+60)/2

Temperature > (80+90)/2

Case IV: Regression Task on Continuous Features

Day	Temperature	Minutes
D1	40	30
D2	48	35
D3	60	60
D4	72	70
D5	80	78
D6	90	88



Jupyter Notebook

Advantages

- Interpretability
- Mixing discrete and continuous variables
- Easy to implement, even in resource constrained settings

Summary

Summary

- Machine learning is ubiquitous

Summary

- Machine learning is ubiquitous
- Interpretability an important goal

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard
  - Greedy approach:

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard
  - Greedy approach:
    - Recursively split to maximize "performance gain"

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard
  - Greedy approach:
    - Recursively split to maximize “performance gain”
  - Issues:

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard
  - Greedy approach:
    - Recursively split to maximize “performance gain”
  - Issues:
    - Can overfit easily!

Summary

- Machine learning is ubiquitous
- Interpretability an important goal
- Decision trees: well known interpretable models
  - Learning optimal tree is hard
  - Greedy approach:
    - Recursively split to maximize “performance gain”
  - Issues:
    - Can overfit easily!
    - Empirically not as powerful as other methods

Next Class

- Ensemble Learning