CPSC429/529: Machine Learning

Lecture 05: Decision Tree

Dongsheng Che
Computer Science Department
East Stroudsburg University

Decision Tree

The Basics

A Motivating Example: Playtennis

Example	Outlook (x_1)	Temp (x_2)	Humid (x_4)	Wind (x_5)	Play? (y)
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

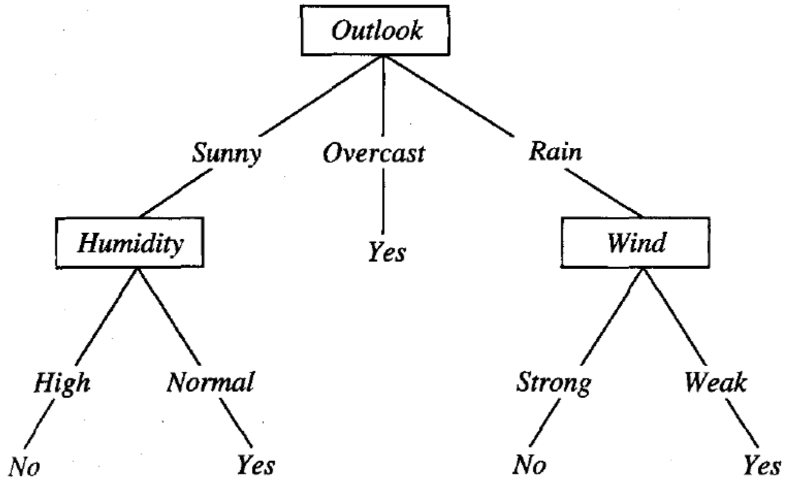
Features:

x_1: Outlook
x_2: Temp
x_3: Humidity
x_4: Wind

y: Play {0/1}

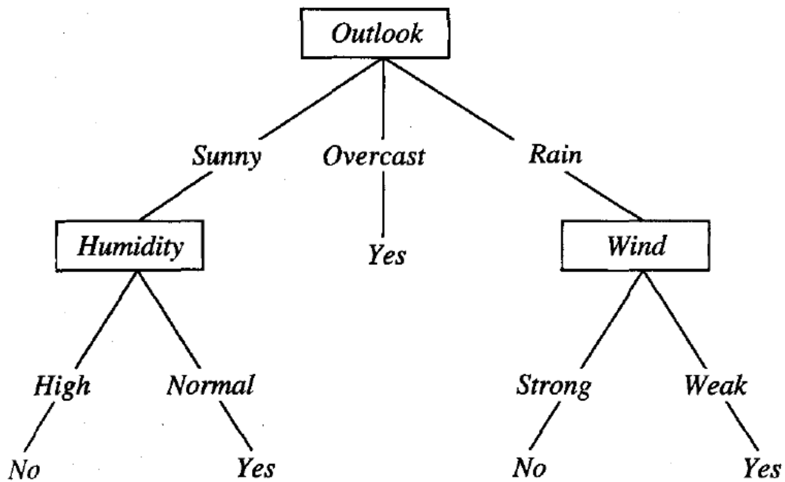
Decision Tree Learning (DTL)

- DTL is a method for approximating discrete-values target function in which the learned function is represented by a DT.



Decision Tree Representation

- Each internal node tests an attribute
- Each branch corresponds to one attribute value
- Each leaf node assigns a classification



Question

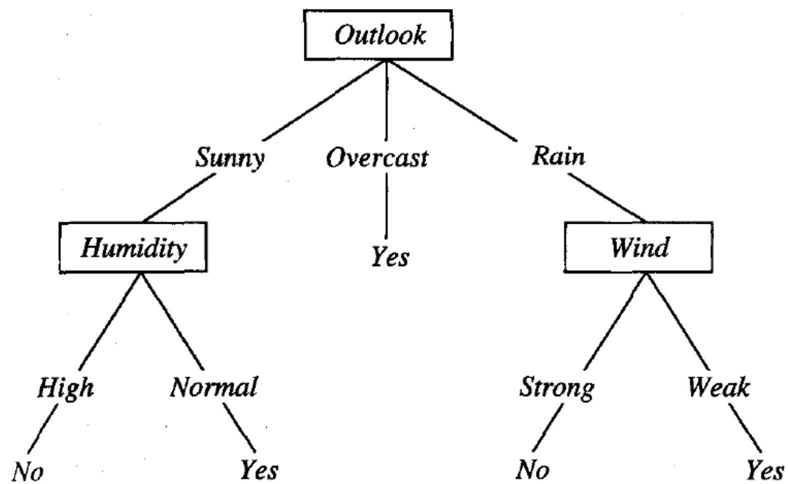
Given the decision tree below, and a new instance where x_1 =sunny, x_2 =hot, x_3 =high, x_4 =strong.

What is your predicted output value (y) ?

Features:

x_1: Outlook
x_2: Temp
x_3: Humidity
x_4: Wind

y: Play {0/1}



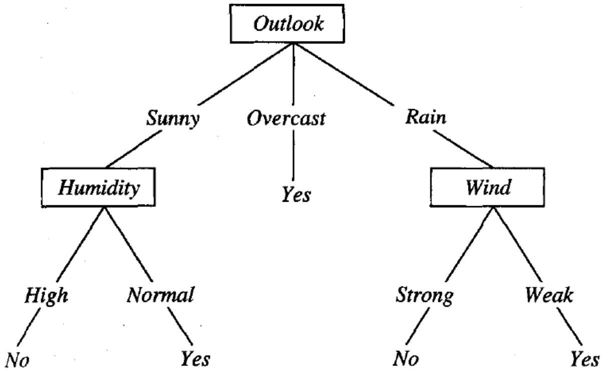
When to consider decision tree?

- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors or missing attribute values.

- Examples:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - …

Rules and Decision Trees

- Can turn the tree into a set of rules:
  - (outlook = sunny & humidity = normal) | (outlook = overcast) | (outlook = rain & wind = weak)
- How do we generate the trees?
  - Need to choose features
  - Need to choose order of features



Basic Decision Tree Learning Algorithm

- Employs a top-down, greedy search through the space of possible DT.

- ID3 (Quinlan, 1986) and its successor C4.5 (Quinlan, 1993)

Decision Tree

The Basics

Decision Tree

The ID3 Algorithm

ID3 (Quinlan)

- Learns decision trees by constructing them top-down, beginning with the question “Which attribute should tested at the root of the tree?”
- To answer this question, each attribute is evaluate to determine how well it alone classifies the training examples?

ID3 Algorithm

1. The algorithm begins by choosing the best descriptive feature to test (i.e., the best question to ask first) using information gain.
2. A root node is then added to the tree and labelled with the selected test feature.
3. The training dataset is then partitioned using the test.
4. For each partition a branch is grown from the node.
5. The process is then repeated for each of these branches using the relevant partition of the training set in place of the full training set and with the selected test feature excluded from further testing.

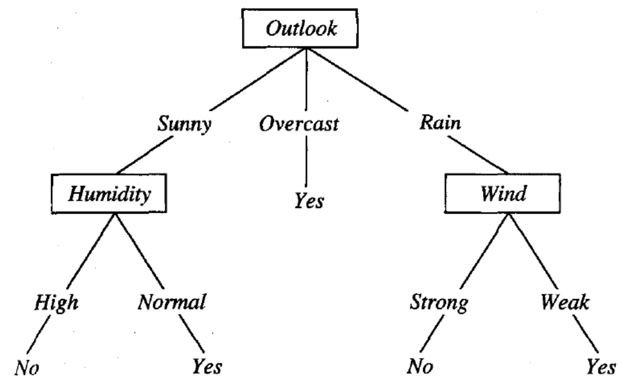
ID3 Algorithm (cont’d)

The algorithm defines three situations where the recursion stops and a leaf node is constructed:

1. All of the instances in the dataset have the same classification (target feature value) then return a leaf node tree with that classification as its label.
2. The set of features left to test is empty then return a leaf node tree with the majority class of the dataset as its classification.
3. The dataset is empty return a leaf node tree with the majority class of the dataset at the parent node that made the recursive call.

PlayTennis Dataset and Decision Tree Model

Example	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



How to Pick the Best Feature?

- Compute information gain for each feature, and then pick the feature with the highest information gain.

$$\mathrm{Gain}(S, F) = \mathrm{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \mathrm{Entropy}(S_f)$$

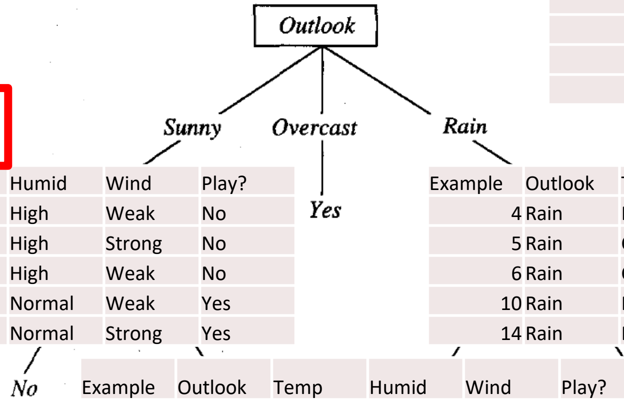
Weighted average of Entropy of subset of dataset

- S is a set of training examples
- S_f is a subset of training examples whose feature value is f for feature F

IG(S, Outlook) Example

S

Example	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



$S_{outlook=Sunny}$

Example	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

$S_{outlook=Overcast}$

Example	Outlook	Temp	Humid	Wind	Play?
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

$S_{outlook=Rain}$

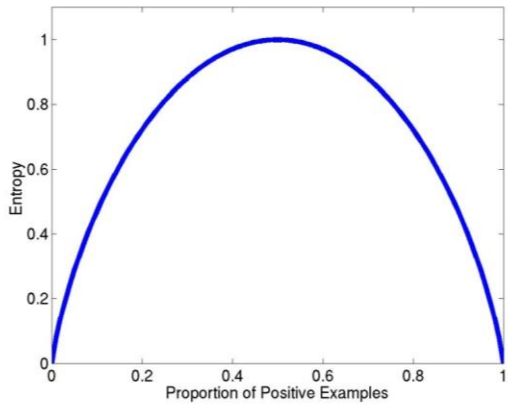
Example	Outlook	Temp	Humid	Wind	Play?
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

$$\text{Gain}(S,F) = \text{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \text{Entropy}(S_f)$$

What is Entropy?

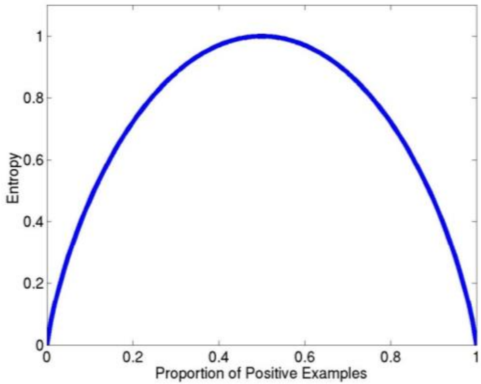
Measures the impurity of an arbitrary collection of examples

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



- *S* is a sample of training examples
- *p+* is the proportion of positive examples in *S*
- *p-* is the proportion of negative examples in *S*

Entropy



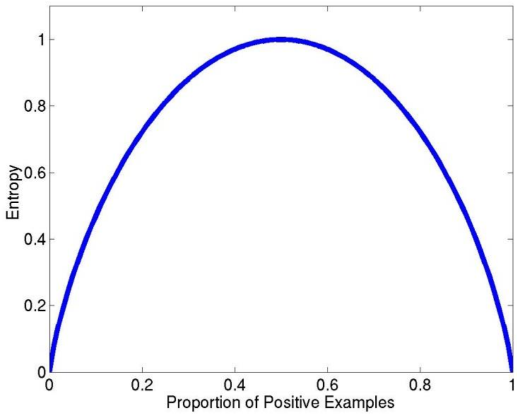
- Entropy(S)=
    - Expected number of bits needed to encode class (+ or -) of randomly drawn member of *S*

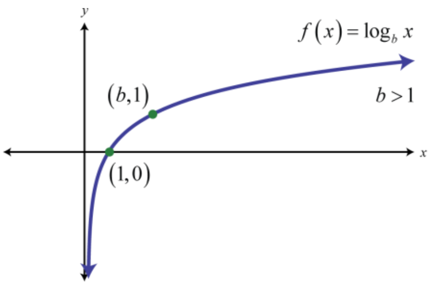
- In general, if the target attribute has *c* different values, then

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

p_i is the proportion of *S* belonging to class *i*

Entropy Calculation Example





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

Suppose there 10 instances:

- 10 instances of +, 0 instance of -
  P(+)=10/10, P(-)=0/10
  E(S) =-P(+)log(P+)-)-P(-)log(P-)
  =-1.0*log(1.0)-0*log(0)=0

- 5 instances of +, 5 instance of -
  P(+)=5/10, P(-)=5/10
  E(S) =-P(+)log(P+)-)-P(-)log(P-)
  =-0.5*log(0.5)-0.5*log(0.5)=1

The purer the dataset, the lower the entropy score!!

Example	Outlook	Temp	Humid	Wind	Play?
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
1	Sunny	Hot	High	Weak	No
8	Sunny	Mild	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

PlayTennis Example

Calculate IG(S, F=Wind)

$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \text{Entropy}(S_f)$$

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

PlayTennis Example

Example	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Step 1: calculate $E(S)$

- S = [9+, 5-]

Important:
We are counting the **target values** when computing entropy scores!!

- Entropy(S)=$-p^{+}\log_2 p^{+} - p^{-}\log_2 p^{(-)}$

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

  $= 0.940$

Example	Outlook	Temp	Humid	Wind	Play?
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
1	Sunny	Hot	High	Weak	No
8	Sunny	Mild	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

PlayTennis Example

Step 2: calculate each $E(S_f)$

- $S_{Wind=Weak} = [6+, 2-]$
- $S_{Wind=Strong} = [3+, 3-]$

- $Entropy(S_{Wind=Weak}) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8)$

  $= 0.811$
- $Entropy(S_{Wind=Strong}) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$

  $= 1$

Example	Outlook	Temp	Humid	Wind	Play?
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
1	Sunny	Hot	High	Weak	No
8	Sunny	Mild	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

PlayTennis Example

Step 3: put together

IG(S, Wind)

$= E(S) - |S_{Weak}|/|S| * E(S_{Weak}) - |S_{Strong}|/|S| * E(S_{Strong})$

$= 0.94 - 8/14*0.811-6/14*1$

$= 0.048$

PlayTennis Example

To find the best feature for the root node, you should calculate

1. IG(S, Outlook)=0.246
2. IG(S, Temp)=0.029
3. IG(S, Humidity)=0.151
4. IG(S, Wind)=0.048

Choose the feature with the highest IG, which is Outlook

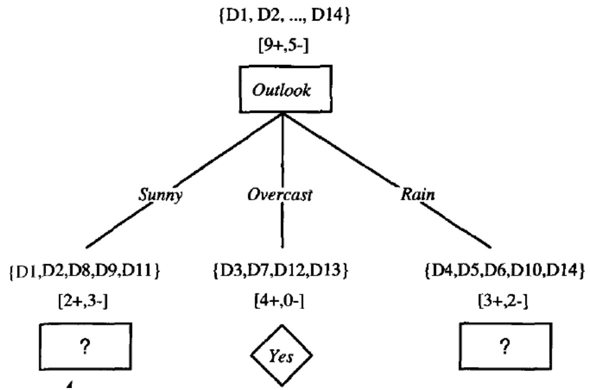
Features:

x_1: Outlook
x_2: Temp
x_3: Humidity
x_4: Wind

y: Play {0/1}

Example	Outlook	Temp	Humid	Wind	Play?
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
1	Sunny	Hot	High	Weak	No
8	Sunny	Mild	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

DT after finding the first best feature



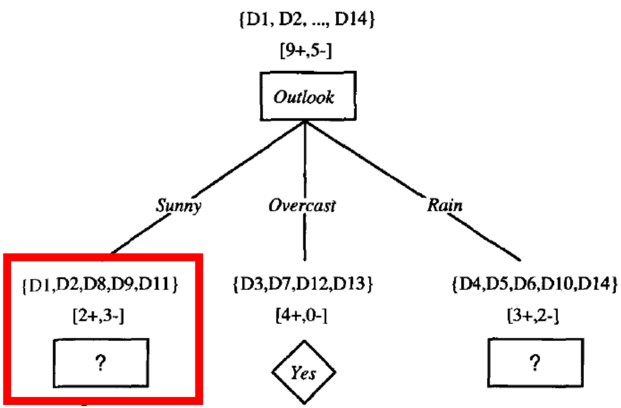
Example	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Example	Outlook	Temp	Humid	Wind	Play?
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Example	Outlook	Temp	Humid	Wind	Play?
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

PlayTennis Example: What Next?

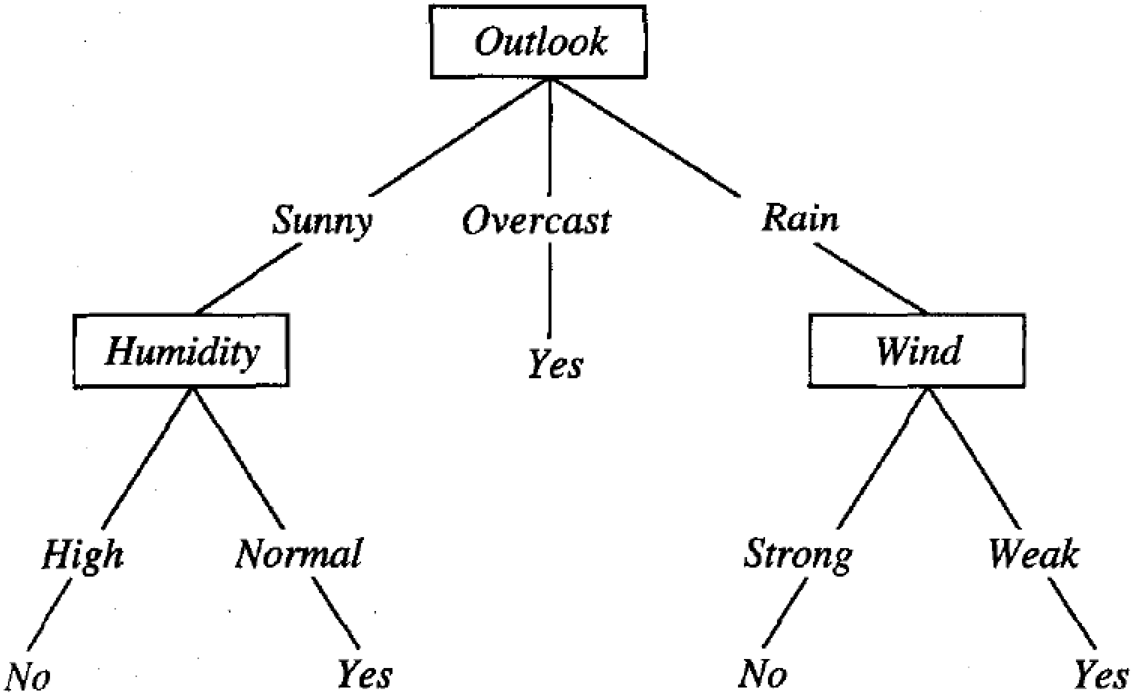
Example	Temp	Humid	Wind	Play?
2	Hot	High	Strong	No
11	Mild	Normal	Strong	Yes
1	Hot	High	Weak	No
8	Mild	High	Weak	No
9	Cool	Normal	Weak	Yes



To find the best feature for this node, you should calculate

1. IG(S_{sunny}, Temp)=0.57
2. IG(S_{sunny}, Humidity)=0.97
3. IG(S_{sunny}, Wind)=0.019
4. Choose Maximum IG

PlayTennis Example: Final Tree



Example 2: Email Spam Dataset

Table: An email spam prediction dataset.

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

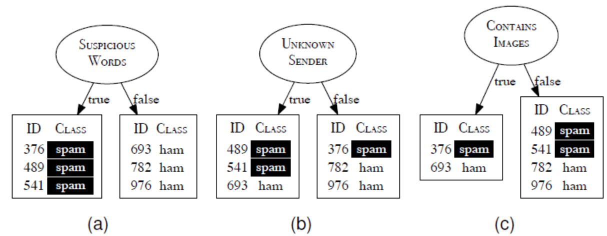


Figure: How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset in Table 1 [15]

$$H(t, \mathcal{D}) = -\sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l))) \tag{2}$$

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\text{entropy of partition } \mathcal{D}_{d=l}} \tag{3}$$

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D}) \tag{4}$$

Calculate Entropy

Table: An email spam prediction dataset.

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

$$\begin{aligned}
H(t, \mathcal{D}) &= -\sum_{l \in \{\text{'spam'}, \text{'ham'}\}} (P(t = l) \times log_2(P(t = l))) \\
&= -\quad ((P(t = \text{'spam'}) \times log_2(P(t = \text{'spam'})) \\
&\qquad + (P(t = \text{'ham'}) \times log_2(P(t = \text{'ham'}))) \\
&= -\left(\left({}^3/_6 \times log_2({}^3/_6)\right) + \left({}^3/_6 \times log_2({}^3/_6)\right)\right) \\
&= 1\ bit
\end{aligned}$$

Calculate Remainder

$$
\begin{aligned}
&rem\,(\text{WORDS}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{WORDS}=T}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{WORDS}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{WORDS}=F}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{WORDS}=F}) \right) \\
&= \left({}^{3}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \left(\left({}^{3}/_{3} \times log_2({}^{3}/_{3}) \right) + \left({}^{0}/_{3} \times log_2({}^{0}/_{3}) \right) \right) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \left(\left({}^{0}/_{3} \times log_2({}^{0}/_{3}) \right) + \left({}^{3}/_{3} \times log_2({}^{3}/_{3}) \right) \right) \right) \right) = 0 \textit{ bits}
\end{aligned}
$$

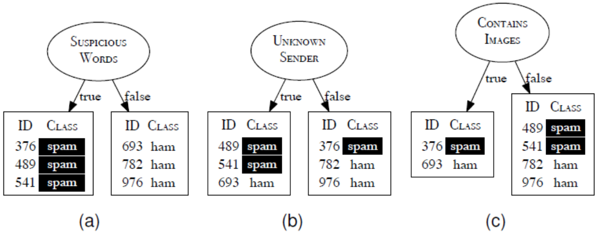


Figure: How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset in Table 1 [15]

$$
\begin{aligned}
&rem\,(\text{SENDER}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{SENDER}=T}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{SENDER}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{SENDER}=F}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{SENDER}=F}) \right) \\
&= \left({}^{3}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \left(\left({}^{2}/_{3} \times log_2({}^{2}/_{3}) \right) + \left({}^{1}/_{3} \times log_2({}^{1}/_{3}) \right) \right) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \left(\left({}^{1}/_{3} \times log_2({}^{1}/_{3}) \right) + \left({}^{2}/_{3} \times log_2({}^{2}/_{3}) \right) \right) \right) \right) = 0.9183 \textit{ bits}
\end{aligned}
$$

$$
\begin{aligned}
&rem\,(\text{IMAGES}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{IMAGES}=T}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{IMAGES}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{IMAGES}=F}|}{|\mathcal{D}|} \times H\,(t, \mathcal{D}_{\text{IMAGES}=F}) \right) \\
&= \left({}^{2}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&\quad + \left({}^{4}/_{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t=l) \times log_2(P(t=l)) \right) \right) \\
&= \left({}^{2}/_{6} \times \left(- \left(\left({}^{1}/_{2} \times log_2({}^{1}/_{2}) \right) + \left({}^{1}/_{2} \times log_2({}^{1}/_{2}) \right) \right) \right) \right) \\
&\quad + \left({}^{4}/_{6} \times \left(- \left(\left({}^{2}/_{4} \times log_2({}^{2}/_{4}) \right) + \left({}^{2}/_{4} \times log_2({}^{2}/_{4}) \right) \right) \right) \right) = 1 \textit{ bit}
\end{aligned}
$$

Calculate Information Gain

$$\begin{aligned} IG(\textsc{Suspicious Words}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Suspicious Words}, \mathcal{D}) \\ &= 1 - 0 = 1 \; bit \end{aligned}$$

$$\begin{aligned} IG(\textsc{Unknown Sender}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Unknown Sender}, \mathcal{D}) \\ &= 1 - 0.9183 = 0.0817 \; bits \end{aligned}$$

$$\begin{aligned} IG(\textsc{Contains Images}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Contains Images}, \mathcal{D}) \\ &= 1 - 1 = 0 \; bits \end{aligned}$$

The feature of “suspicious words” has the highest IG, so use this feature as the root node

Decision Tree

The ID3 Algorithm

Decision Tree

Alternative Feature Selection

Metrics

Information Gain Ratio

- Entropy based information gain, preferences features with many values.
- One way of addressing this issue is to use **information gain ratio** which is computed by dividing the information gain of a feature by the amount of information used to determine the value of the feature:

$$GR(d, \mathcal{D}) = \frac{IG(d, \mathcal{D})}{\boxed{-\sum_{l \in levels(d)} (P(d = l) \times log_2(P(d = l)))}} \quad (1)$$

The denominator is the entropy based on the feature value, not the entropy of the target when calculating information gain

Information Gain

Table: The vegetation classification dataset.

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	high	chaparral
2	true	moderate	low	riparian
3	true	steep	medium	riparian
4	false	steep	medium	chaparral
5	false	flat	high	conifer
6	true	steep	highest	conifer
7	true	steep	high	chaparral

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	'true'	$\mathcal{D}_1$	$\mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_6, \mathbf{d}_7$	1.5	1.2507	0.3060
	'false'	$\mathcal{D}_2$	$\mathbf{d}_1, \mathbf{d}_4, \mathbf{d}_5$	0.9183		
SLOPE	'flat'	$\mathcal{D}_3$	$\mathbf{d}_5$	0	0.9793	0.5774
	'moderate'	$\mathcal{D}_4$	$\mathbf{d}_2$	0		
	'steep'	$\mathcal{D}_5$	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6, \mathbf{d}_7$	1.3710		
ELEVATION	'low'	$\mathcal{D}_6$	$\mathbf{d}_2$	0	0.6793	0.8774
	'medium'	$\mathcal{D}_7$	$\mathbf{d}_3, \mathbf{d}_4$	1.0		
	'high'	$\mathcal{D}_8$	$\mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_7$	0.9183		
	'highest'	$\mathcal{D}_9$	$\mathbf{d}_6$	0		

Information Gain Ratio

$$
\begin{aligned}
&H(\textsc{Stream}, \mathcal{D}) \\
&= -\sum_{l \in \left\{\substack{\text{'true'}, \\ \text{'false'}}\right\}} P(\textsc{Stream} = l) \times log_2\left(P(\textsc{Stream} = l)\right) \\
&= -\left(\left({}^{4}/{}_{7} \times log_2({}^{4}/{}_{7})\right) + \left({}^{3}/{}_{7} \times log_2({}^{3}/{}_{7})\right)\right) \\
&= 0.9852 \text{ bits}
\end{aligned}
$$

$$
\begin{aligned}
&H(\textsc{Slope}, \mathcal{D}) \\
&= -\sum_{l \in \left\{\substack{\text{'flat'}, \\ \text{'moderate'}, \\ \text{'steep'}}\right\}} P(\textsc{Slope} = l) \times log_2\left(P(\textsc{Slope} = l)\right) \\
&= -\left(\left({}^{1}/{}_{7} \times log_2({}^{1}/{}_{7})\right) + \left({}^{1}/{}_{7} \times log_2({}^{1}/{}_{7})\right) + \left({}^{5}/{}_{7} \times log_2({}^{5}/{}_{7})\right)\right) \\
&= 1.1488 \text{ bits}
\end{aligned}
$$

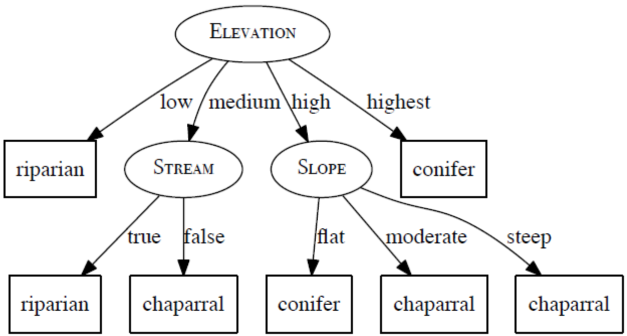
$$
\begin{aligned}
&H(\textsc{Elevation}, \mathcal{D}) \\
&= -\sum_{l \in \left\{\substack{\text{'low'}, \\ \text{'medium'}, \\ \text{'high'}, \\ \text{'highest'}}\right\}} P(\textsc{Elevation} = l) \times log_2\left(P(\textsc{Elevation} = l)\right) \\
&= -\left(\left({}^{1}/{}_{7} \times log_2({}^{1}/{}_{7})\right) + \left({}^{2}/{}_{7} \times log_2({}^{2}/{}_{7})\right) + \left({}^{3}/{}_{7} \times log_2({}^{3}/{}_{7})\right) + \left({}^{1}/{}_{7} \times log_2({}^{1}/{}_{7})\right)\right) \\
&= 1.8424 \text{ bits}
\end{aligned}
$$

$$
\begin{aligned}
IG(\textsc{Stream}, \mathcal{D}) &= 0.3060 \\
IG(\textsc{Slope}, \mathcal{D}) &= 0.5774 \\
IG(\textsc{Elevation}, \mathcal{D}) &= 0.8774
\end{aligned}
$$

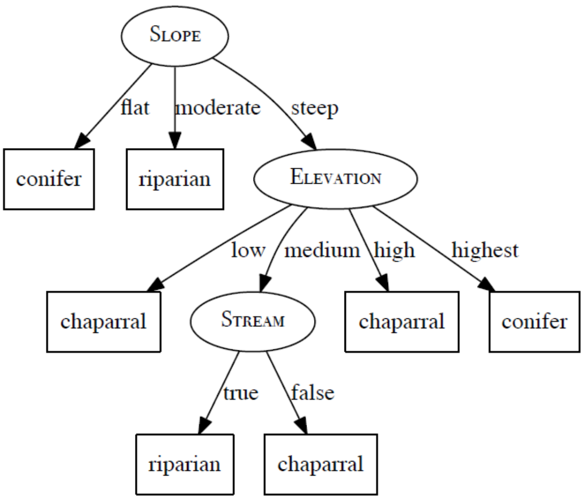
$$
\begin{aligned}
GR(\textsc{Stream}, \mathcal{D}) &= \frac{0.3060}{0.9852} = 0.3106 \\
GR(\textsc{Slope}, \mathcal{D}) &= \frac{0.5774}{1.1488} = 0.5026 \\
GR(\textsc{Elevation}, \mathcal{D}) &= \frac{0.8774}{1.8424} = 0.4762
\end{aligned}
$$

Decision Trees by Two Selection Metrics

Using Information Gain



Using Information Gain Ratio



Gini Index

- Another commonly used measure of impurity is the Gini index:

$$Gini\,(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2$$

$$\begin{aligned}
&Gini\,(\textsc{Vegetation}, \mathcal{D})\\
&= 1 - \sum_{l \in \left\{\substack{\text{'chapparal',}\\ \text{'riparian',}\\ \text{'conifer'}}\right\}} P(\textsc{Vegetation} = l)^2\\
&= 1 - \left((3/7)^2 + \left({}^2/_7\right)^2 + \left({}^2/_7\right)^2 \right)\\
&= 0.6531
\end{aligned}$$

Entropy-based

$$\begin{aligned}
&H\,(\textsc{Vegetation}, \mathcal{D})\\
&= - \sum_{l \in \left\{\substack{\text{'chaparral',}\\ \text{'riparian',}\\ \text{'conifer'}}\right\}} P(\textsc{Vegetation} = l) \times log_2\,(P(\textsc{Vegetation} = l))\\
&= - \left(\left({}^3/_7 \times log_2({}^3/_7)\right) + \left({}^2/_7 \times log_2({}^2/_7)\right) + \left({}^2/_7 \times log_2({}^2/_7)\right) \right)\\
&= 1.5567\ bits
\end{aligned}$$

Information Gain using Gini Index

- Information gain can be calculated using the Gini index by replacing the entropy measure with the Gini index

$$Gini(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2$$

$$\cancel{H(t, \mathcal{D}) = - \sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l)))} \quad \cancel{(2)}$$

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{\cancel{H(t, \mathcal{D}_{d=l})}}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}} \quad (3)$$

$$IG(d, \mathcal{D}) = \cancel{H(t, \mathcal{D})} - rem(d, \mathcal{D}) \quad (4)$$

Decision Tree

Alternative Feature Selection

Metrics

Decision Tree

Handling Continuous Descriptive

Features

Handling Continuous Descriptive Features

- The easiest way to handle continuous valued descriptive features is to turn them into boolean features by defining a threshold and using this threshold to partition the instances based their value of the continuous descriptive feature ((i.e., < threshold, >= threshold).
- How do we set the threshold?
  - The instances in the dataset are sorted according to the continuous feature values.
  - The adjacent instances in the ordering that have different classifications are then selected as possible threshold points.
  - The optimal threshold is found by computing the information gain for each of these classification transition boundaries and selecting the boundary with the highest information gain as the threshold.

Sort by Elevation (Continuous Feature)

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	3 900	chapparal
2	true	moderate	300	riparian
3	true	steep	1 500	riparian
4	false	steep	1 200	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer
7	true	steep	3 000	chapparal

ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

Find Thresholds

ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

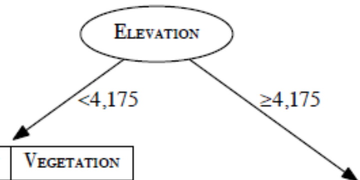
- We look for adjacent pairs that have different target levels:
  - (d_2,d_4), $(300+1200)/2 = 750$
  - (d_4,d_3), $(1200+1500)/2 = 1350$
  - (d_3,d_7), $(1500+3000)/2 = 2250$
  - (d_1,d_5), $(3900+4450)/2 = 4175$

Calculate Information Gain

Table: Partition sets (Part.), entropy, remainder (Rem.), and information gain (Info. Gain) for the candidate ELEVATION thresholds: $\geq$750, $\geq$1 350, $\geq$2 250 and $\geq$4 175.

Split by Threshold	Part.	Instances	Partition Entropy	Rem.	Info. Gain
$\geq$750	$\mathcal{D}_1$	$\mathbf{d}_2$	0.0	1.2507	0.3060
	$\mathcal{D}_2$	$\mathbf{d}_4, \mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.4591		
$\geq$1 350	$\mathcal{D}_3$	$\mathbf{d}_2, \mathbf{d}_4$	1.0	1.3728	0.1839
	$\mathcal{D}_4$	$\mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.5219		
$\geq$2 250	$\mathcal{D}_5$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_3$	0.9183	0.9650	0.5917
	$\mathcal{D}_6$	$\mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.0		
$\geq$4 175	$\mathcal{D}_7$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1$	0.9710	0.6935	0.8631
	$\mathcal{D}_8$	$\mathbf{d}_5, \mathbf{d}_6$	0.0		

Decision Tree



	ID	Stream	Slope	Elevation	Vegetation
D7	2	true	moderate	300	riparian
	4	false	steep	1,200	chaparral
	3	true	steep	1,500	riparian
	7	true	steep	3,000	chaparral
	1	false	steep	3,900	chaparral

	ID	Stream	Slope	Elevation	Vegetation
D8	5	false	flat	4,450	conifer
	6	true	steep	5,000	conifer

Figure: The vegetation classification decision tree after the dataset has been split using Elevation $\geq$ 4 175.

Important: Unlike discrete features, we don’t strike out the continuous feature columns after splitting the dataset. We may still this feature down the tree!

Decision Tree

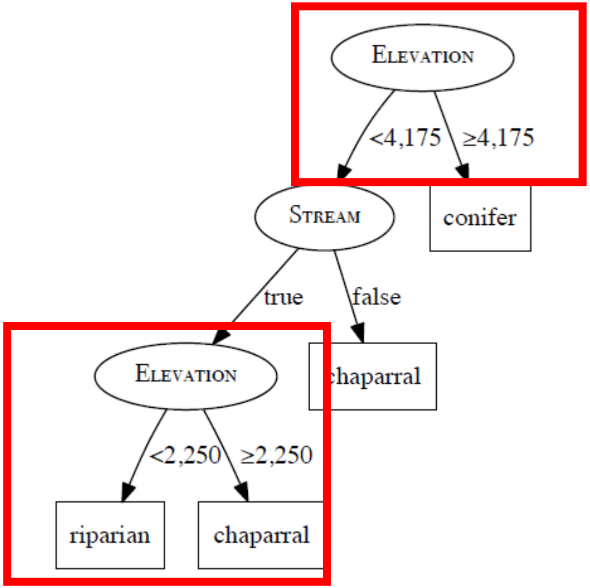


Figure: The decision tree that would be generated for the vegetation classification dataset listed in Table 3 [17] using information gain.

Decision Tree

Handling Continuous Descriptive

Features

Decision Tree

Predicting Continuous Targets (Regression)

Regression Tree

- Regression trees are constructed by adapting the ID3 algorithm to use a measure of variance rather than a measure of impurity (entropy) when selecting the best attribute

  - The impurity (variance) at a node can be calculated using the following equation:

$$var(t, \mathcal{D}) = \frac{\sum_{i=1}^{n} \left(t_i - \bar{t}\right)^2}{n-1} \tag{3}$$

  - We select the feature to split on at a node by selecting the feature that minimizes the weighted variance across the resulting partitions:

$$\mathbf{d}[best] = \operatorname*{argmin}_{d \in \mathbf{d}} \sum_{l \in levels(d)} \frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|} \times var(t, \mathcal{D}_{d=l}) \tag{4}$$

Classification and Regression

Classification

$$H(t, \mathcal{D}) = - \sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l))) \quad (2)$$

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}} \quad (3)$$

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D}) \quad (4)$$

Regression

$$var(t, \mathcal{D}) = \frac{\sum_{i=1}^{n} (t_i - \bar{t})^2}{n - 1} \quad (3)$$

$$\mathbf{d}[best] = \underset{d \in \mathbf{d}}{\operatorname{argmin}} \sum_{l \in levels(d)} \frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|} \times var(t, \mathcal{D}_{d=l}) \quad (4)$$

Important: We are using the **target values** for both calculations!

Weighted Variance

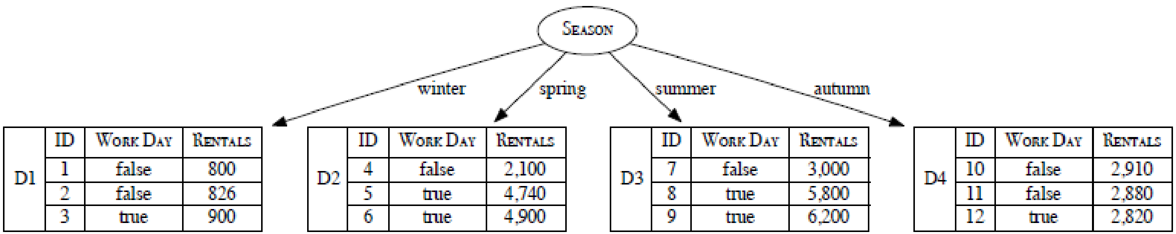
ID	Season	Work Day	Rentals
1	winter	false	800
2	winter	false	826
3	winter	true	900
4	spring	false	2 100
5	spring	true	4 740
6	spring	true	4 900

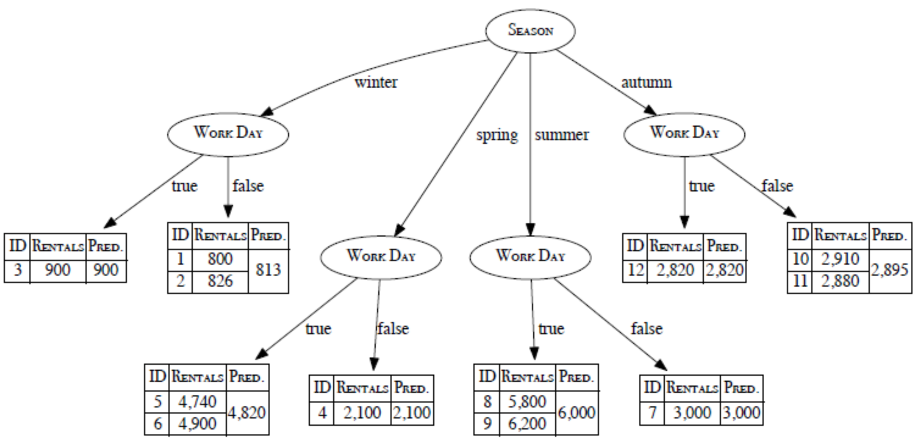
ID	Season	Work Day	Rentals
7	summer	false	3 000
8	summer	true	5 800
9	summer	true	6 200
10	autumn	false	2 910
11	autumn	false	2 880
12	autumn	true	2 820

Split by Feature	Level	Part.	Instances	$\frac{\lvert\mathcal{D}_{d=l}\rvert}{\lvert\mathcal{D}\rvert}$	$var(t, \mathcal{D})$	Weighted Variance
Season	'winter'	$\mathcal{D}_1$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.25	2 692	$1\,379\,331\frac{1}{3}$
	'spring'	$\mathcal{D}_2$	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.25	$2\,472\,533\frac{1}{3}$	
	'summer'	$\mathcal{D}_3$	$\mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9$	0.25	3 040 000	
	'autumn'	$\mathcal{D}_4$	$\mathbf{d}_{10}, \mathbf{d}_{11}, \mathbf{d}_{12}$	0.25	2 100	
Work Day	'true'	$\mathcal{D}_5$	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{12}$	0.50	$4\,026\,346\frac{1}{3}$	$2\,551\,813\frac{1}{3}$
	'false'	$\mathcal{D}_6$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_7, \mathbf{d}_{10}, \mathbf{d}_{11}$	0.50	1 077 280	

The best feature should be the one with lowest weighted variance!

Regression Tree





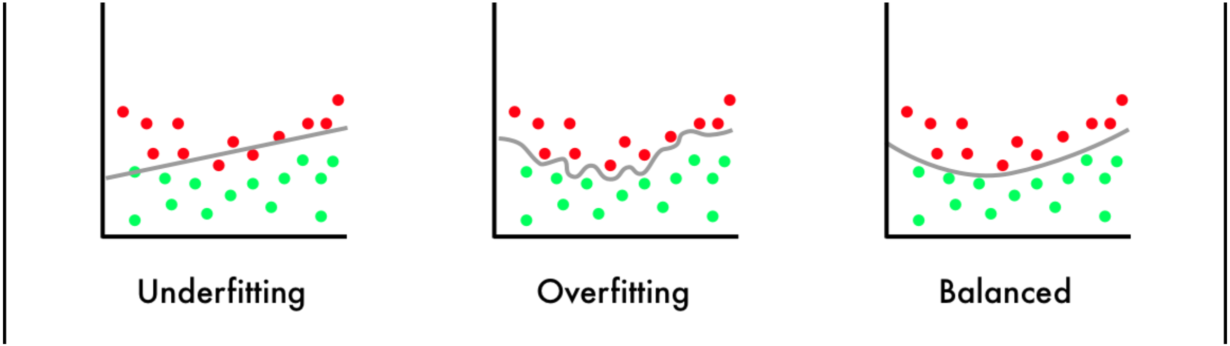
Decision Tree

Predicting Continuous Targets (Regression)

Decision Tree

Overfitting and Tree Pruning

Overfitting v.s. Underfitting



Overfitting occurs when the model performs well on training data but generalizes poorly to unseen data.

https://laptrinhx.com/8-simple-techniques-to-prevent-overfitting-3288224346/

Overfitting Problem in Decision Tree

- In the case of a decision tree, over-fitting involves splitting the data on an irrelevant feature.

The likelihood of over-fitting occurring increases as a tree gets deeper because the resulting classifications are based on smaller and smaller subsets as the dataset is partitioned after each feature test in the path.

Pruning

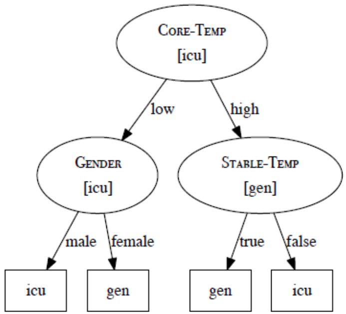
- Pre-pruning: stop the recursive partitioning early. Pre-pruning is also known as forward pruning. Common pre-pruning approaches:
  - Early stopping: tree depth, minimum instance numbers of leaf node
  - Chi-square pruning
- Post-pruning: allow the algorithm to grow the tree as much as it likes and then prune the tree of the branches that cause over-fitting.

post-pruning Approach

Using the validation set evaluate the prediction accuracy achieved by both the fully grown tree and the pruned copy of the tree. If the pruned copy of the tree performs no worse than the fully grown tree the node is a candidate for pruning.

Post-Pruning Example

Tree built from training dataset



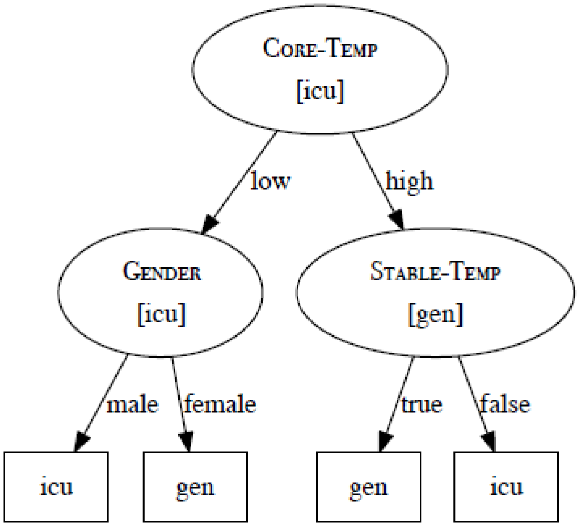
Validation dataset used for pruning

ID	CORE-TEMP	STABLE-TEMP	GENDER	DECISION
1	high	true	male	gen
2	low	true	female	icu
3	high	false	female	icu
4	high	false	male	icu
5	low	false	female	icu
6	low	true	male	icu

Pruning strategy

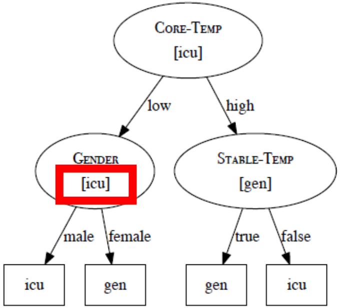
Check non-leaf node from left to right, from bottom to up:

1. Check **Gender** node to see whether we can prune it or not
2. Check **Stable-Temp** node to see whether we can prune it or not
3. Check **Core-Temp** node to see whether we can prune it or not

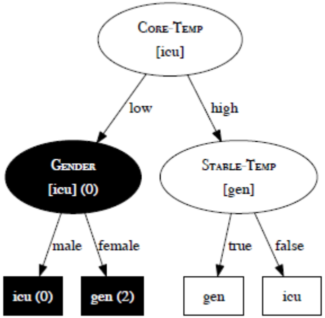


Check Gender Node

ID	CORE-TEMP	STABLE-TEMP	GENDER	DECISION
1	high	true	male	gen
2	low	true	female	icu
3	high	false	female	icu
4	high	false	male	icu
5	low	false	female	icu
6	low	true	male	icu

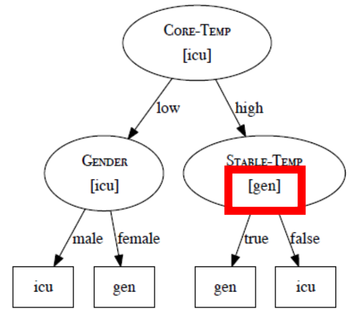


- If pruned, all three instances (2,5,6) are predicted as "icu", and the validation set shows all three are "icu", error rate=0/3=0
- If not pruned:
  - If gender='male', then predicts 'icu', correct
  - If gender='female', then predicts 'gen', the actual labels are 'icu', so incorrect for two predictions here
  - error rate = (0+2)/3
- Conclusion: Prune this **subtree rooted this node**

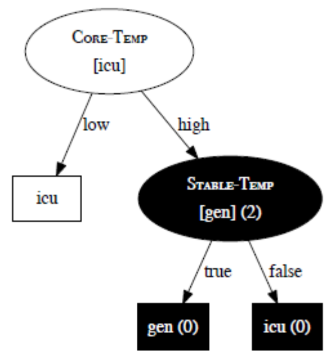


Check Stable-Temp Node

ID	Core-Temp	Stable-Temp	Gender	Decision
1	high	true	male	gen
2	low	true	female	icu
3	high	false	female	icu
4	high	false	male	icu
5	low	false	female	icu
6	low	true	male	icu

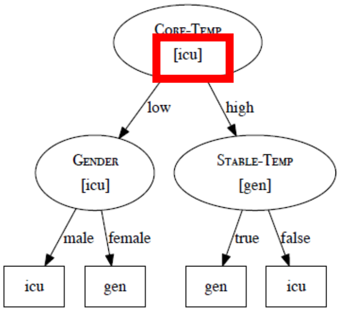


- If pruned, all three instances (1,3,4) are predicted as "gen", and the validation set shows one 'gen', two ''icu", error rate=(0+2)/3=2/3
- If not pruned:
  - If stable-temp='true', then predicts 'gen', correct
  - If stable-temp='false', , then predicts 'icu', correct
  - error rate = (0+0)/3=0
- Conclusion: Don't prune this **Subtree rooted this node**

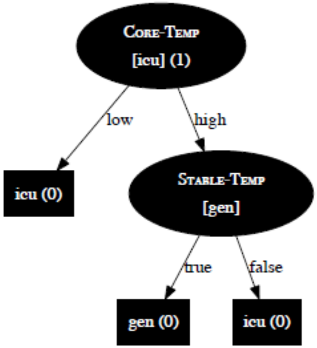


Check Core-Temp Node

ID	CORE-TEMP	STABLE-TEMP	GENDER	DECISION
1	high	true	male	gen
2	low	true	female	icu
3	high	false	female	icu
4	high	false	male	icu
5	low	false	female	icu
6	low	true	male	icu



- If pruned, all six instances are predicted as “icu”, and the validation set shows one ‘gen’, five ‘’icu”, error rate=(1+0)/6=1/5
- If not pruned:
  - If core-temp=‘low’, then predicts ‘icu’, correct
  - If core-temp=‘high’,
    - If stable-temp=‘true’, then predicts ‘gen’, correct
    - If stable-temp=‘false’, then predicts ‘icu’, correct
  - error rate = (0+0+0)/6=0
- Conclusion: Don’t prune this **Subtree rooted this node**



Decision Tree

Overfitting and Tree Pruning

Decision Tree

DecisonTreeClassifier in Scikit-Learn

DecisionTreeClassifier on Vegetation data

In [2]:
```
#load data and extract data
df = pd.read_csv('vegetation.data', sep=",")
df
```

Out[2]:

	STREAM	SLOPE	ELEVATION	VEGETATION
0	False	steep	high	chaparral
1	True	moderate	low	riparian
2	True	steep	medium	riparian
3	False	steep	medium	chaparral
4	False	flat	high	conifer
5	True	steep	highest	conifer
6	True	steep	high	chaparral

Data Preprocessing for Classifier

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
STREAM        7 non-null bool
SLOPE         7 non-null object
ELEVATION     7 non-null object
VEGETATION    7 non-null object
dtypes: bool(1), object(3)
memory usage: 303.0+ bytes
```

```
In [4]: from sklearn import preprocessing

        # it is required that all feature/target values be numerical
        # Systematically convert all string (labeled as object) type into labels(1,2,3,...)
        label_encoding = preprocessing.LabelEncoder()
        for column_name in df.columns:
            if df[column_name].dtype == object:
                df[column_name] = label_encoding.fit_transform(df[column_name])
            else:
                pass
```

```
In [5]: # # extract X, y
        y = df.iloc[:, -1]   # all columns except the last column
        X = df.iloc[:, :-1]  # last column
```

Model training and visualization (1)

In [6]:
```
from sklearn.tree import DecisionTreeClassifier
dt_clf=DecisionTreeClassifier(criterion='gini')
dt_clf.fit(X,y)
```

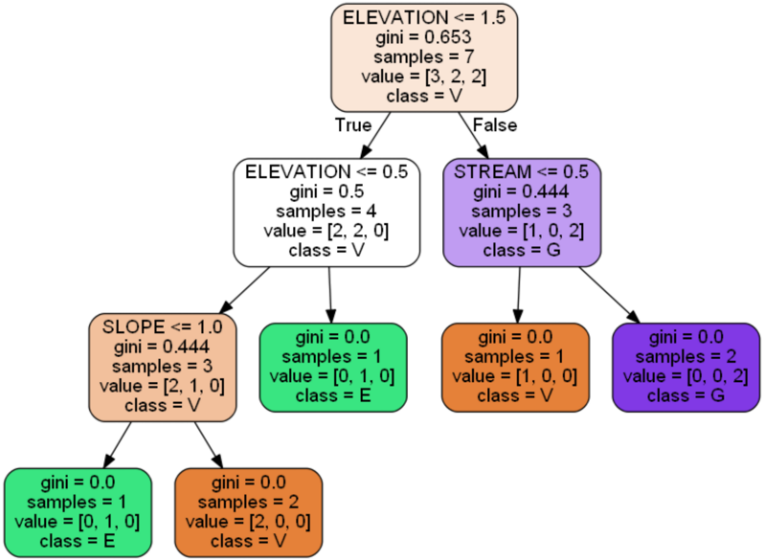
Out[6]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [7]:
```
from sklearn.tree import export_graphviz

export_graphviz(dt_clf,
                out_file='vegetation_tree_gini.dot',
                feature_names=df.columns[:-1],
                class_names=df.columns[-1],
                rounded=True,
                filled=True)
```

Model training and visualization (2)

In [8]:
```
! dot -Tpng vegetation_tree_gini.dot -o vegetation_tree_gini.png

from IPython.display import Image
Image("vegetation_tree_gini.png", width=500)
```

Out[8]:



The CART Training Algorithm in SkLearn

Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train decision trees (also called "growing" trees). The algorithm works by first splitting the training set by feature k and threshold t_k.

It chooses k and t_k by searching for the (k, t_k) that produce the purest subsets weighted by their size.

The following figure gives the loss function that CART tries to minimize:

$$\boxed{J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}}$$

Where:

- $G_{left/right}$ measures the resulted impurity in the left/right subsets.
- $m_{left/right}$ correspond to the number of instances in the left/right subsets.

Regularization Hyperparameters

- **Criterion:** The impurity metrics to be used {“gini”, “entropy”}, default=”gini”
- **min_samples_split**: The minimum number of samples a node must have for it to split.
- **min_samples_leaf**: The minimum number of samples a leaf must have.
- **min_weight_fraction_leaf**: mean_samples_leaf as a fraction.
- **max_leaf_nodes**: the maximum number of leaf nodes.
- **max_features**: The maximum number of features that are evaluated for any split.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Decision Tree

DecisonTreeClassifier in
Scikit-Learn