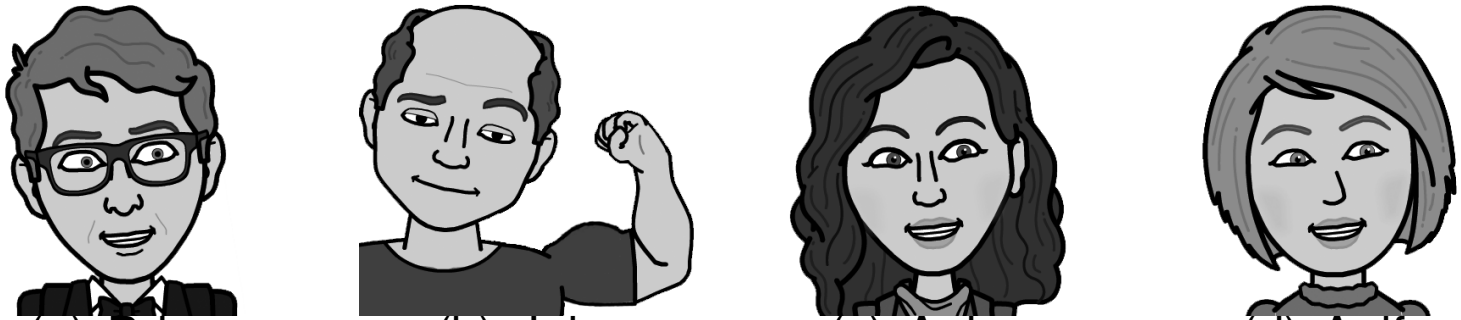
Information-based Learning
Sections 4.1, 4.2, 4.3

John D. Kelleher and Brian Mac Namee and Aoife D'Arcy

- In this chapter we are going to introduce a machine learning algorithm that tries to build predictive models **using only the most informative features**.
- In this context an informative feature is a **descriptive feature** whose values split the instances in the dataset into **homogeneous sets** with respect to the target feature value.

Big Idea

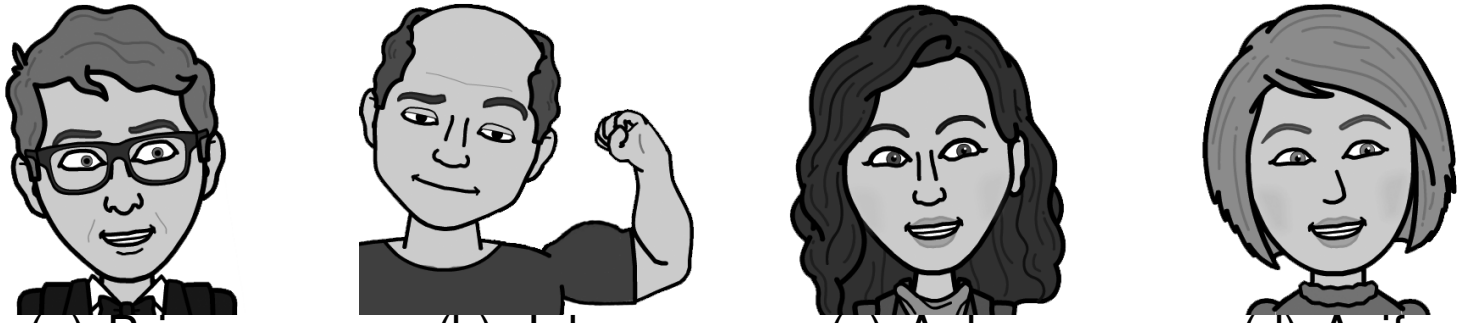


(a) Brian (b) John (c) Aphra (d) Aoife

Figure: Cards showing character faces and names for the *Guess-Who* game

.

Man	Long Hair	Glasses	Name
Yes	No	Yes	Brian
Yes	No	No	John
No	Yes	No	Aphra
No	No	No	Aoife



(a) Brian (b) John (c) Aphra (d) Aoife

Figure: Cards showing character faces and names for the *Guess-Who* game

.

Which question would you ask first:

1. Is it a man?
2. Does the person wear glasses?

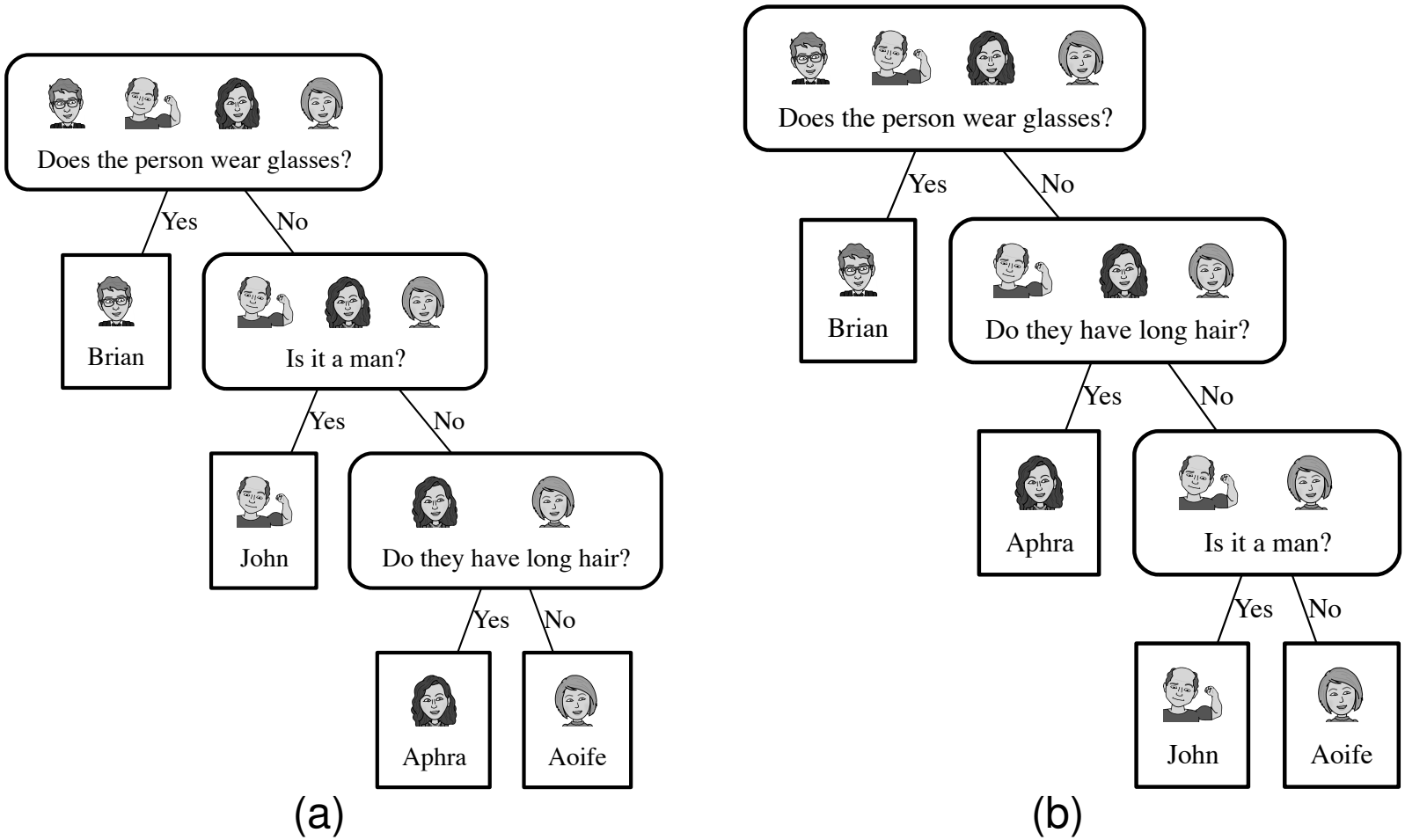


Figure: The different question sequences that can follow in a game of *Guess-Who* beginning with the question Does the person wear glasses?

- In both of the diagrams:
  - one path is 1 question long,
  - one path is 2 questions long,
  - and two paths are 3 questions long.
- Consequently, if you ask Question (2) first the average number of questions you have to ask per game is:

$$\frac{1+2+3+3}{4} = 2.25$$

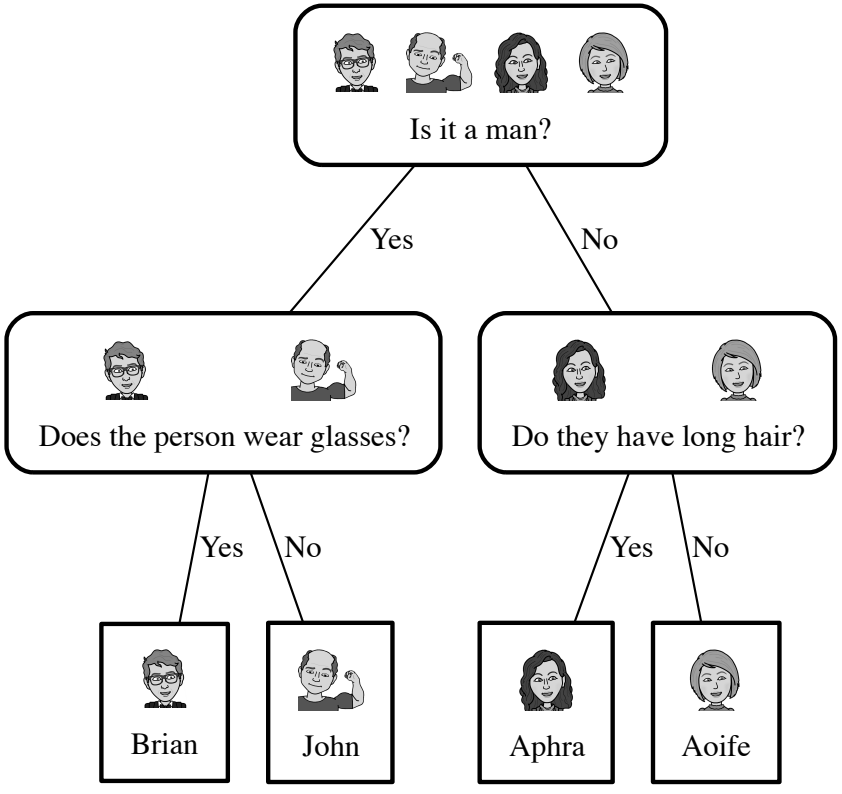


Figure: The different question sequences that can follow in a game of *Guess-Who* beginning with the question Is it a man?

- All the paths in this diagram are two questions long.
- So, on average if you ask Question (1) first the average number of questions you have to ask per game is:

$$\frac{2+2+2+2}{4} = 2$$

- On average getting an answer to Question (1) seems to give you more information than an answer to Question (2): less follow up questions.
- This is not because of the literal message of the answers: YES or NO.
- It is to do with how the answer to each questions splits the domain into different sized sets based on the value of the descriptive feature the question is asked about and the likelihood of each possible answer to the question.

Big Idea

- So the big idea here is to figure out which features are the most informative ones to ask questions about by considering the effects of the different answers to the questions, in terms of:
  1. how the domain is split up after the answer is received,
  2. and the likelihood of each of the answers.

Fundamentals

- A decision tree consists of:
  1. a **root node** (or starting node),
  2. **interior nodes**
  3. and **leaf nodes** (or terminating nodes).
- Each of the non-leaf nodes (root and interior) in the tree specifies a test to be carried out on one of the query's descriptive features.
- Each of the leaf nodes specifies a predicted classification for the query.

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

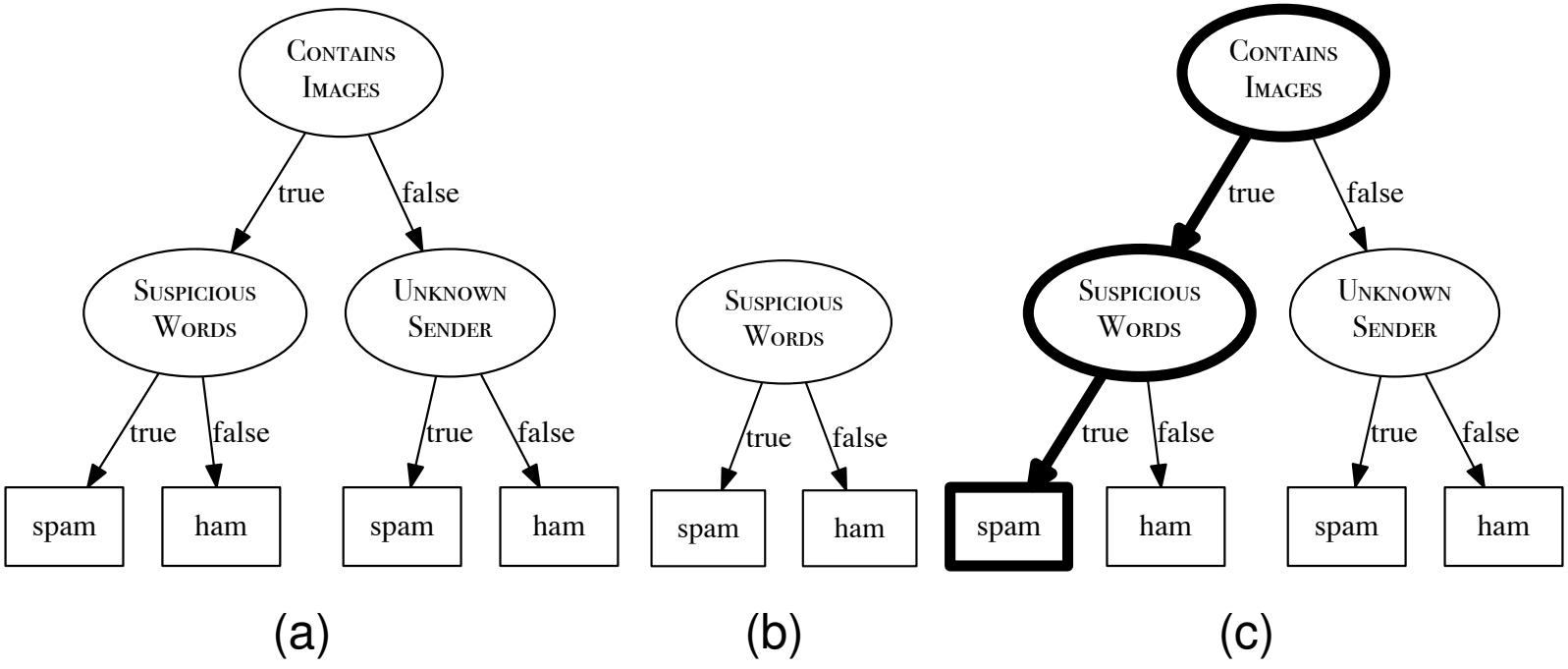


Figure: (a) and (b) show two decision trees that are consistent with the instances in the spam dataset. (c) shows the path taken through the tree shown in (a) to make a prediction for the query instance: SUSPICIOUS WORDS = *'true'*, UNKNOWN SENDER = *'true'*, CONTAINS IMAGES = *'true'*.

- Both of these trees will return identical predictions for all the examples in the dataset.
- So, which tree should we use?

- Apply the same approach as we used in the *Guess-Who* game: prefer decision trees that use less tests (shallower trees).
- This is an example of Occam's Razor.

How do we create shallow trees?

- The tree that tests SUSPICIOUS WORDS at the root is very shallow because the SUSPICIOUS WORDS feature perfectly splits the data into pure groups of *'spam'* and *'ham'*.
- Descriptive features that split the dataset into pure sets with respect to the target feature provide information about the target feature.
- So we can make shallow trees by testing the informative features early on in the tree.
- All we need to do that is a computational metric of the purity of a set: entropy

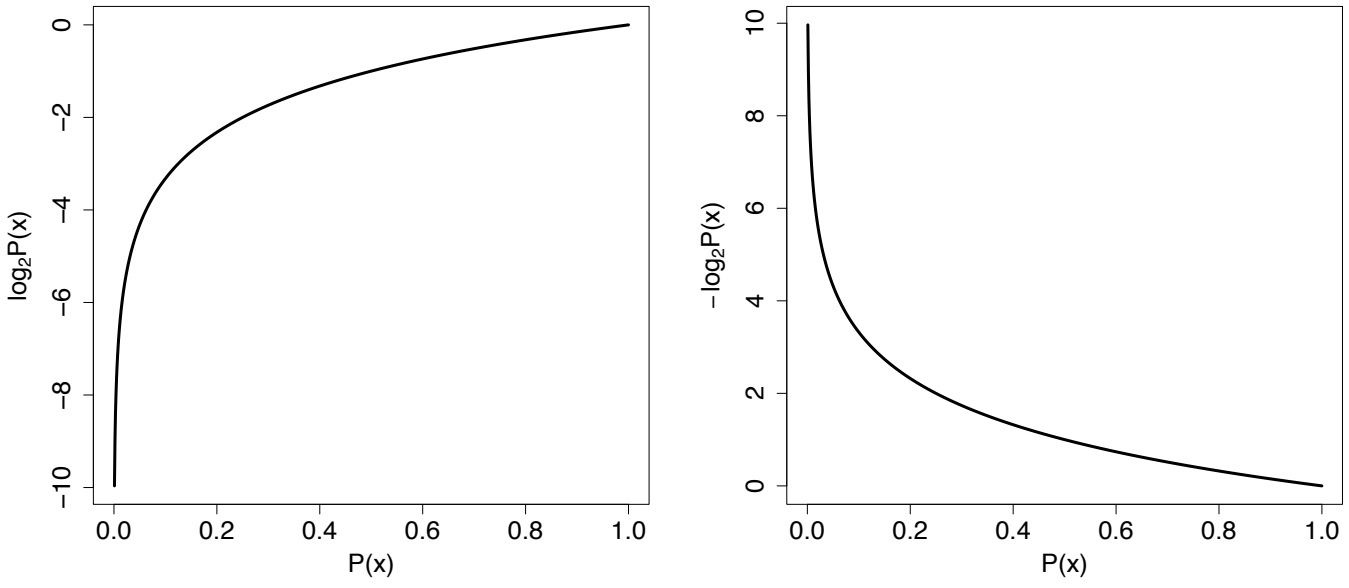
- Claude Shannon's entropy model defines a computational measure of the impurity of the elements of a set.
- An easy way to understand the entropy of a set is to think in terms of the uncertainty associated with guessing the result if you were to make a random selection from the set.

- Entropy is related to the probability of a outcome.
  - High probability $\rightarrow$ Low entropy
  - Low probability $\rightarrow$ High entropy
- If we take the log of a probability and multiply it by -1 we get this mapping!

What is a log?

Remember the *log* of *a* to the base *b* is the number to which we must raise *b* to get *a*.

- $log_2(0.5) = -1$ because $2^{-1} = 0.5$
- $log_2(1) = 0$ because $2^0 = 1$
- $log_2(8) = 3$ because $2^3 = 8$
- $log_5(25) = 2$ because $5^2 = 25$
- $log_5(32) = 2.153$ because $5^{2.153} = 32$



(a) (b)

Figure: (a) A graph illustrating how the value of a binary log (the log to the base 2) of a probability changes across the range of probability values. (b) the impact of multiplying these values by -1.

- Shannon's model of entropy is a weighted sum of the logs of the probabilities of each of the possible outcomes when we make a random selection from a set.

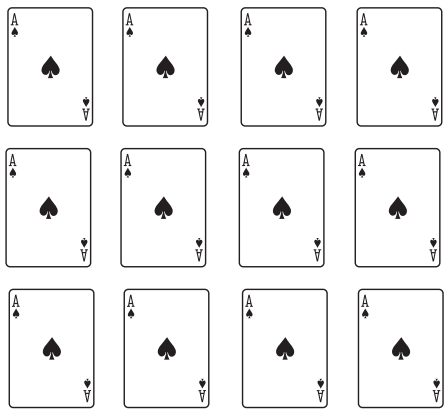
$$H(t) = -\sum_{i=1}^{I} (P(t = i) \times log_s(P(t = i))) \quad (1)$$

- What is the entropy of a set of 52 different playing cards?

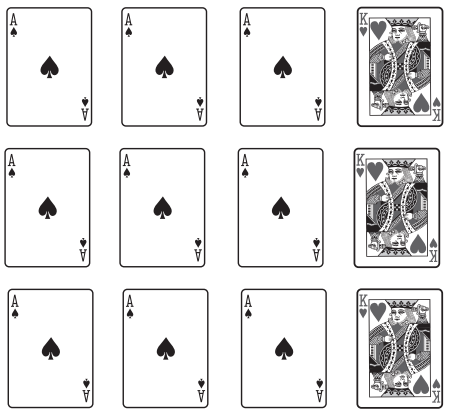
$$
\begin{aligned}
H(card) &= -\sum_{i=1}^{52} P(card = i) \times log_2(P(card = i)) \\
&= -\sum_{i=1}^{52} 0.019 \times log_2(0.019) = -\sum_{i=1}^{52} -0.1096 \\
&= 5.700 \; bits
\end{aligned}
$$

- What is the entropy of a set of 52 playing cards if we only distinguish between the cards based on their suit $\{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}$?

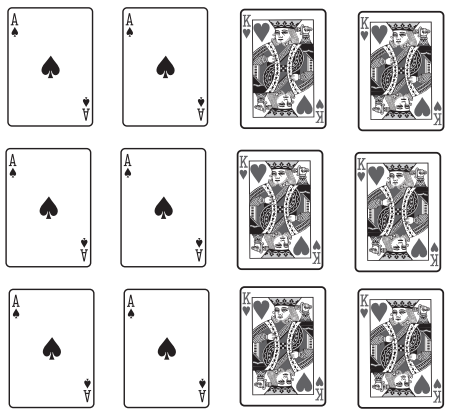
$$
\begin{aligned}
H(suit) &= -\sum_{l \in \{\heartsuit,\clubsuit,\diamondsuit,\spadesuit\}} P(suit = l) \times log_2(P(suit = l)) \\
&= -\big((P(\heartsuit) \times log_2(P(\heartsuit))) + (P(\clubsuit) \times log_2(P(\clubsuit))) \\
&\qquad + (P(\diamondsuit) \times log_2(P(\diamondsuit))) + (P(\spadesuit) \times log_2(P(\spadesuit))) \big) \\
&= -\Big(\Big({}^{13}/_{52} \times log_2({}^{13}/_{52}) \Big) + \Big({}^{13}/_{52} \times log_2({}^{13}/_{52}) \Big) \\
&\qquad + \Big({}^{13}/_{52} \times log_2({}^{13}/_{52}) \Big) + \Big({}^{13}/_{52} \times log_2({}^{13}/_{52}) \Big) \Big) \\
&= -\big((0.25 \times -2) + (0.25 \times -2) \\
&\qquad + (0.25 \times -2) + (0.25 \times -2) \big) \\
&= 2 \; bits
\end{aligned}
$$



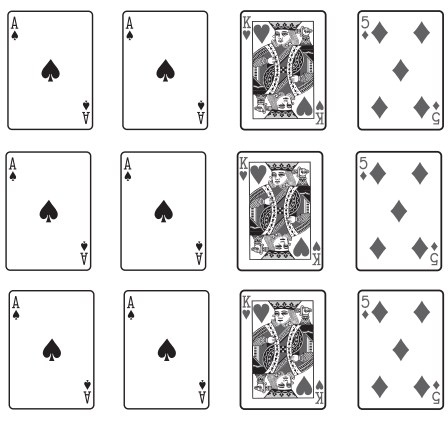
(a) $H(card) = 0.00$



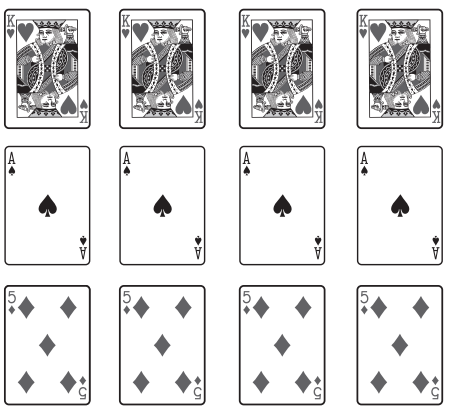
(b) $H(card) = 0.81$



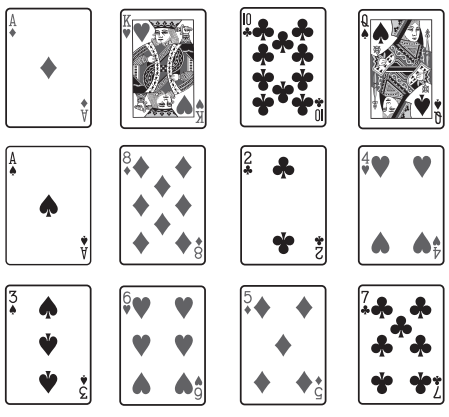
(c) $H(card) = 1.00$



(d) $H(card) = 1.50$



(e) $H(card) = 1.58$



(f) $H(card) = 3.58$

Figure: The entropy of different sets of playing cards measured in bits.

Table: The relationship between the entropy of a message and the set it was selected from.

Entropy of a Message	Properties of the Message Set
High	A large set of equally likely messages.
Medium	A large set of messages, some more likely than others.
Medium	A small set of equally likely messages.
Low	A small set of messages with one very likely message.

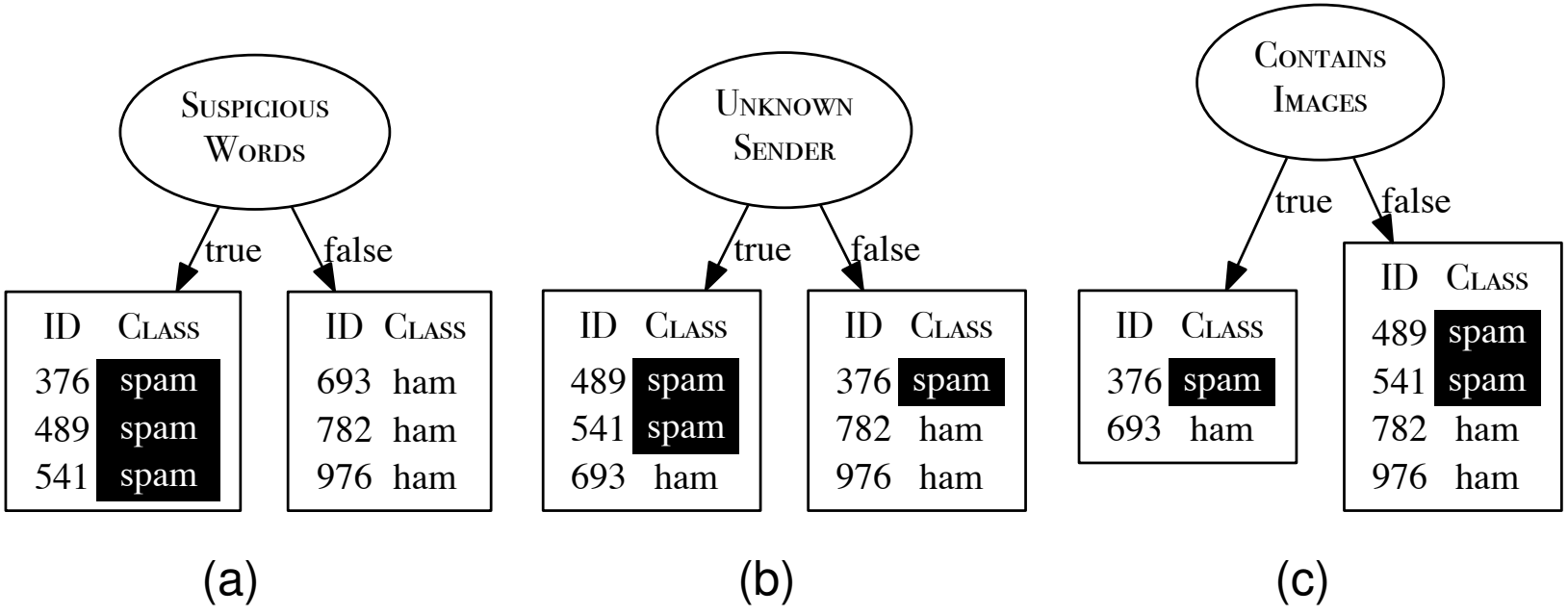


Figure: How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset in Table 1 [15]

- Our intuition is that the ideal discriminatory feature will partition the data into **pure** subsets where all the instances in each subset have the same classification.
  - SUSPICIOUS WORDS perfect split.
  - UNKNOWN SENDER mixture but some information (when *'true'* most instances are *'spam'*).
  - CONTAINS IMAGES no information.
- One way to implement this idea is to use a metric called **information gain**.

Information Gain

- The information gain of a descriptive feature can be understood as a measure of the reduction in the overall entropy of a prediction task by testing on that feature.

Computing information gain involves the following 3 equations:

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

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

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

- As an example we will calculate the information gain for each of the descriptive features in the spam email dataset.

- Calculate the entropy for the target feature in the dataset.

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

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

$$
\begin{aligned}
H(t, \mathcal{D}) &= - \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} (P(t = l) \times log_2(P(t = l))) \\
&= - ((P(t = \text{'spam'}) \times log_2(P(t = \text{'spam'})) \\
&\quad + (P(t = \text{'ham'}) \times log_2(P(t = \text{'ham'}))) \\
&= -\left(\left({}^{3}/_{6} \times log_2({}^{3}/_{6})\right) + \left({}^{3}/_{6} \times log_2({}^{3}/_{6})\right)\right) \\
&= 1 \; bit
\end{aligned}
$$

- Calculate the remainder for the SUSPICIOUS WORDS feature in the dataset.

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

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

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

- Calculate the remainder for the UNKNOWN SENDER feature in the dataset.

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

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

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

- Calculate the remainder for the CONTAINS IMAGES feature in the dataset.

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

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

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

- Calculate the information gain for the three descriptive feature in the dataset.

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

$$IG(\text{SUSPICIOUS WORDS}, \mathcal{D}) = H(\text{CLASS}, \mathcal{D}) - rem(\text{SUSPICIOUS WORDS}, \mathcal{D})$$
$$= 1 - 0 = 1 \textit{ bit}$$

$$IG(\text{UNKNOWN SENDER}, \mathcal{D}) = H(\text{CLASS}, \mathcal{D}) - rem(\text{UNKNOWN SENDER}, \mathcal{D})$$
$$= 1 - 0.9183 = 0.0817 \textit{ bits}$$

$$IG(\text{CONTAINS IMAGES}, \mathcal{D}) = H(\text{CLASS}, \mathcal{D}) - rem(\text{CONTAINS IMAGES}, \mathcal{D})$$
$$= 1 - 1 = 0 \textit{ bits}$$

- The results of these calculations match our intuitions.

Standard Approach: The ID3 Algorithm

- ID3 Algorithm (Iterative Dichotomizer 3)
- Attempts to create the shallowest tree that is consistent with the data that it is given.
- The ID3 algorithm builds the tree in a recursive, depth-first manner, beginning at the root node and working down to the leaf nodes.

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.

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.

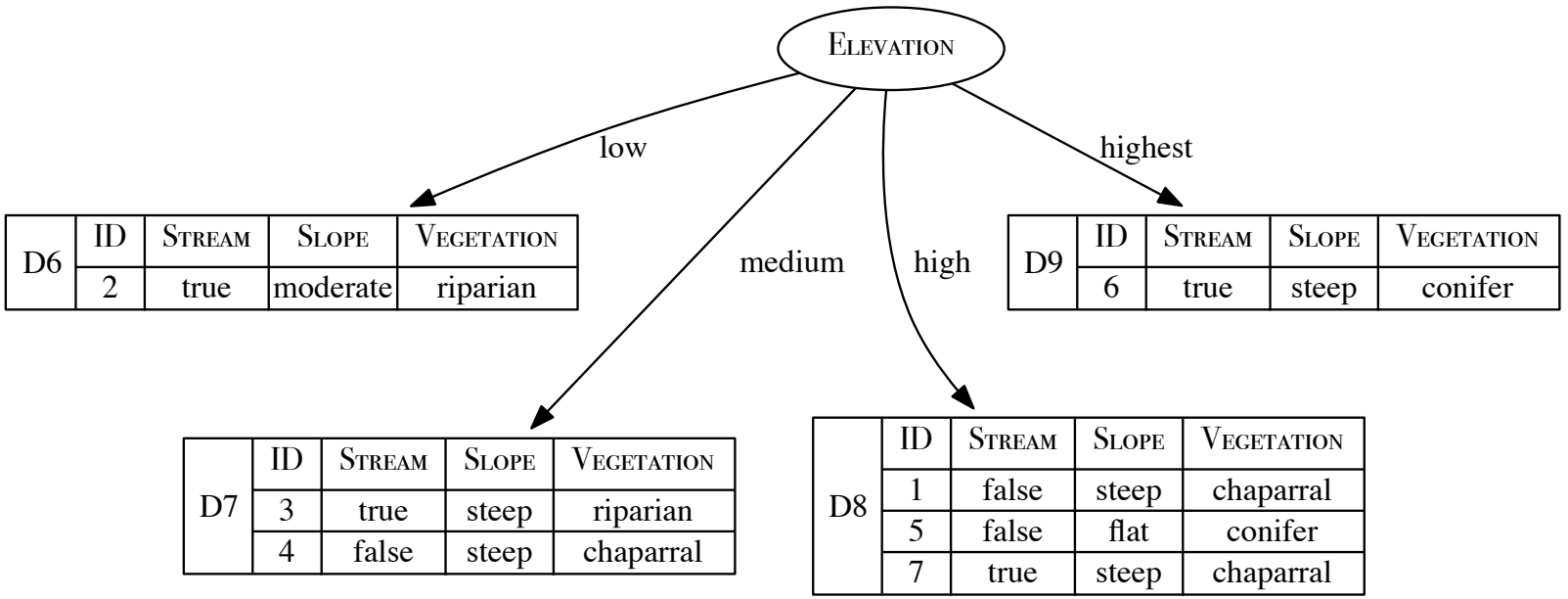
Table: The vegetation classification dataset.

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

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

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by feature for the dataset in Table 3 [49].

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		



D6

ID	STREAM	SLOPE	VEGETATION
2	true	moderate	riparian

D9

ID	STREAM	SLOPE	VEGETATION
6	true	steep	conifer

D7

ID	STREAM	SLOPE	VEGETATION
3	true	steep	riparian
4	false	steep	chaparral

D8

ID	STREAM	SLOPE	VEGETATION
1	false	steep	chaparral
5	false	flat	conifer
7	true	steep	chaparral

Figure: The decision tree after the data has been split using ELEVATION.

$$
\begin{aligned}
& H(\text{VEGETATION}, \mathcal{D}_7) \\
&= - \sum_{l \in \left\{ \begin{array}{l} \text{'chaparral'}, \\ \text{'riparian'}, \\ \text{'conifer'} \end{array} \right\}} P(\text{VEGETATION} = l) \times log_2 \left(P(\text{VEGETATION} = l) \right) \\
&= - \left(\left({}^1/_2 \times log_2({}^1/_2) \right) + \left({}^1/_2 \times log_2({}^1/_2) \right) + \left({}^0/_2 \times log_2({}^0/_2) \right) \right) \\
&= 1.0 \; \textit{bits}
\end{aligned}
$$

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by feature for the dataset $\mathcal{D}_7$ in Figure 9 [52].

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	*'true'*	$\mathcal{D}_{10}$	$\mathbf{d}_3$	0	0	1.0
	'false'	$\mathcal{D}_{11}$	$\mathbf{d}_4$	0		
SLOPE	*'flat'*	$\mathcal{D}_{12}$		0	1.0	0
	'moderate'	$\mathcal{D}_{13}$		0		
	'steep'	$\mathcal{D}_{14}$	$\mathbf{d}_3, \mathbf{d}_4$	1.0		

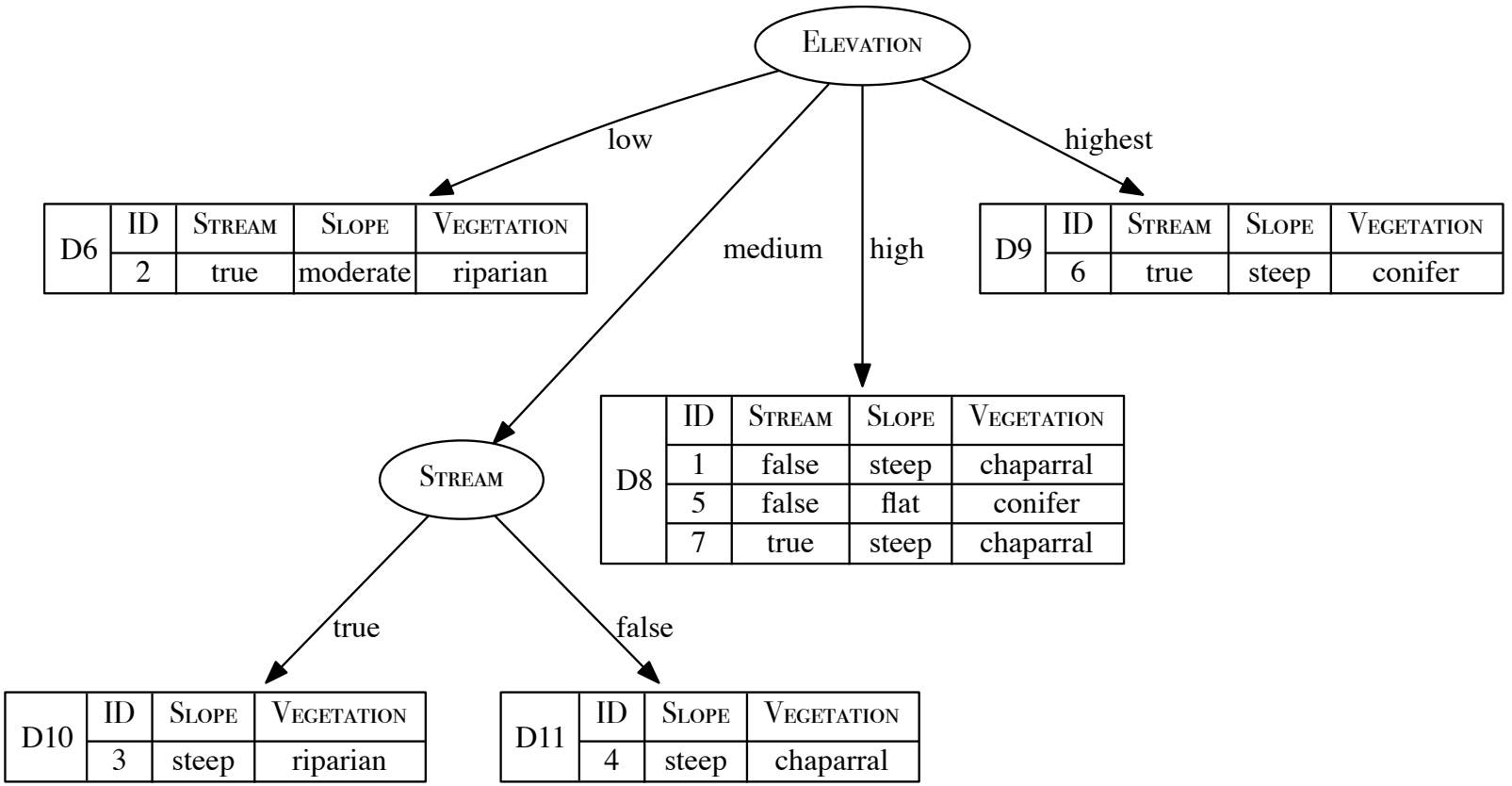


Figure: The state of the decision tree after the $\mathcal{D}_7$ partition has been split using STREAM.

$$
\begin{aligned}
& H\left(\textsc{Vegetation}, \mathcal{D}_8\right) \\
&= - \sum_{l \in \left\{ \begin{array}{l} \text{'chaparral'}, \\ \text{'riparian'}, \\ \text{'conifer'} \end{array} \right\}} P(\textsc{Vegetation} = l) \times log_2 \left(P(\textsc{Vegetation} = l) \right) \\
&= - \left(\left({}^2/_3 \times log_2({}^2/_3) \right) + \left({}^0/_3 \times log_2({}^0/_3) \right) + \left({}^1/_3 \times log_2({}^1/_3) \right) \right) \\
&= 0.9183 \; \textit{bits}
\end{aligned}
$$

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by by feature for the dataset $\mathcal{D}_8$ in Figure 10 [55].

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	*'true'*	$\mathcal{D}_{15}$	$\mathbf{d}_7$	0	0.6666	0.2517
	'false'	$\mathcal{D}_{16}$	$\mathbf{d}_1, \mathbf{d}_5$	1.0		
SLOPE	*'flat'*	$\mathcal{D}_{17}$	$\mathbf{d}_5$	0	0	0.9183
	'moderate'	$\mathcal{D}_{18}$		0		
	'steep'	$\mathcal{D}_{19}$	$\mathbf{d}_1, \mathbf{d}_7$	0		

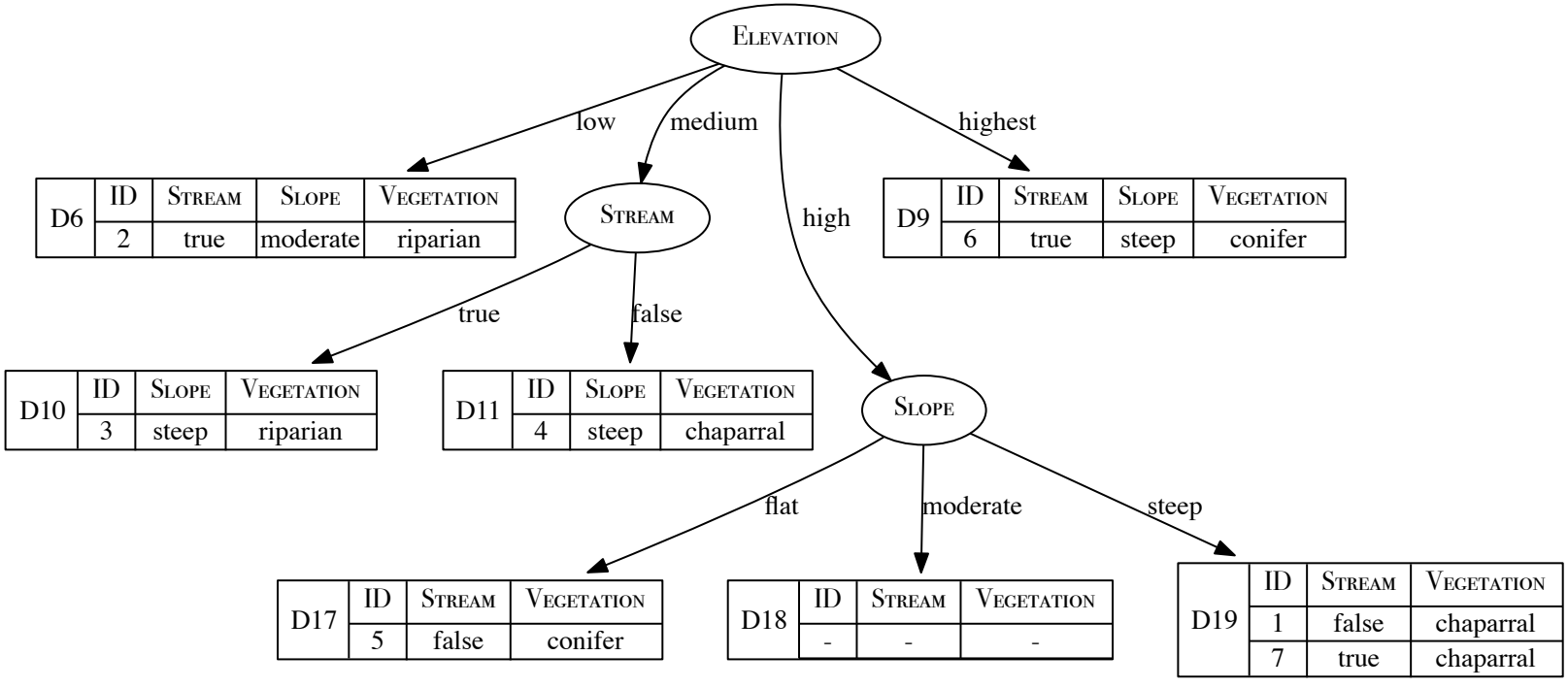


Figure: The state of the decision tree after the $\mathcal{D}_8$ partition has been split using SLOPE.

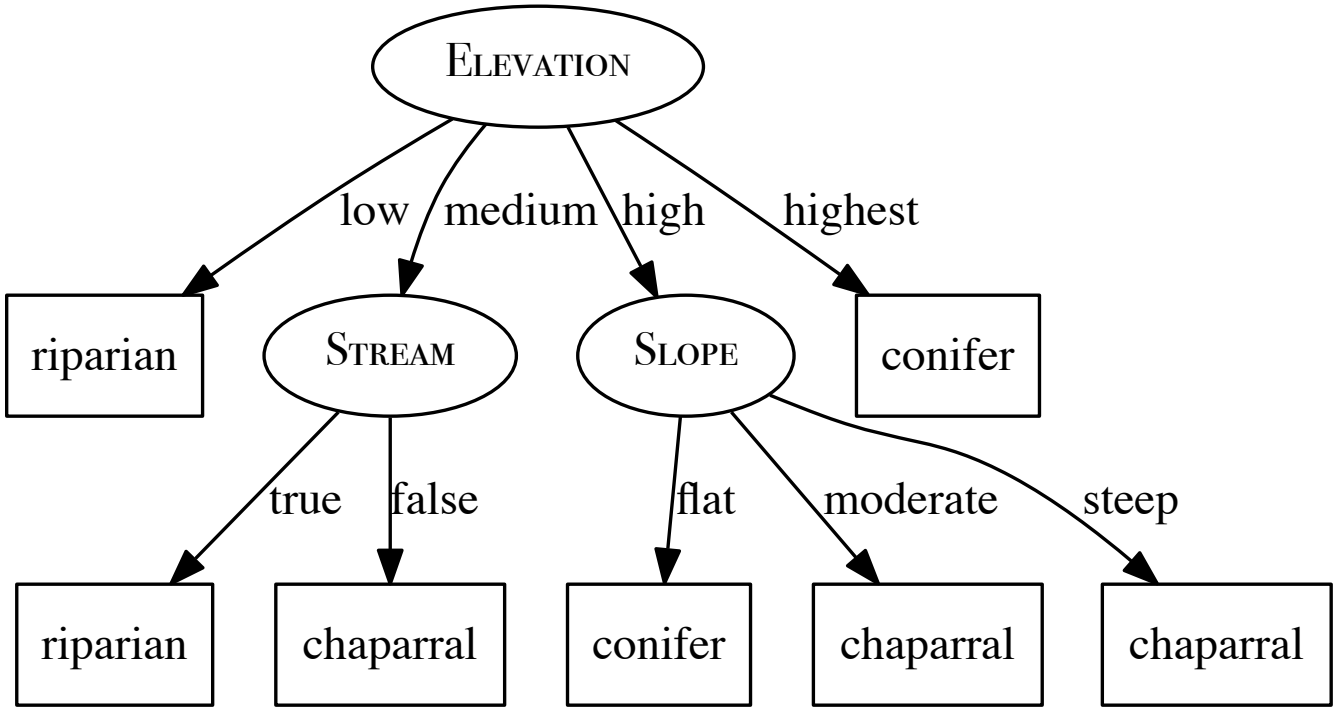
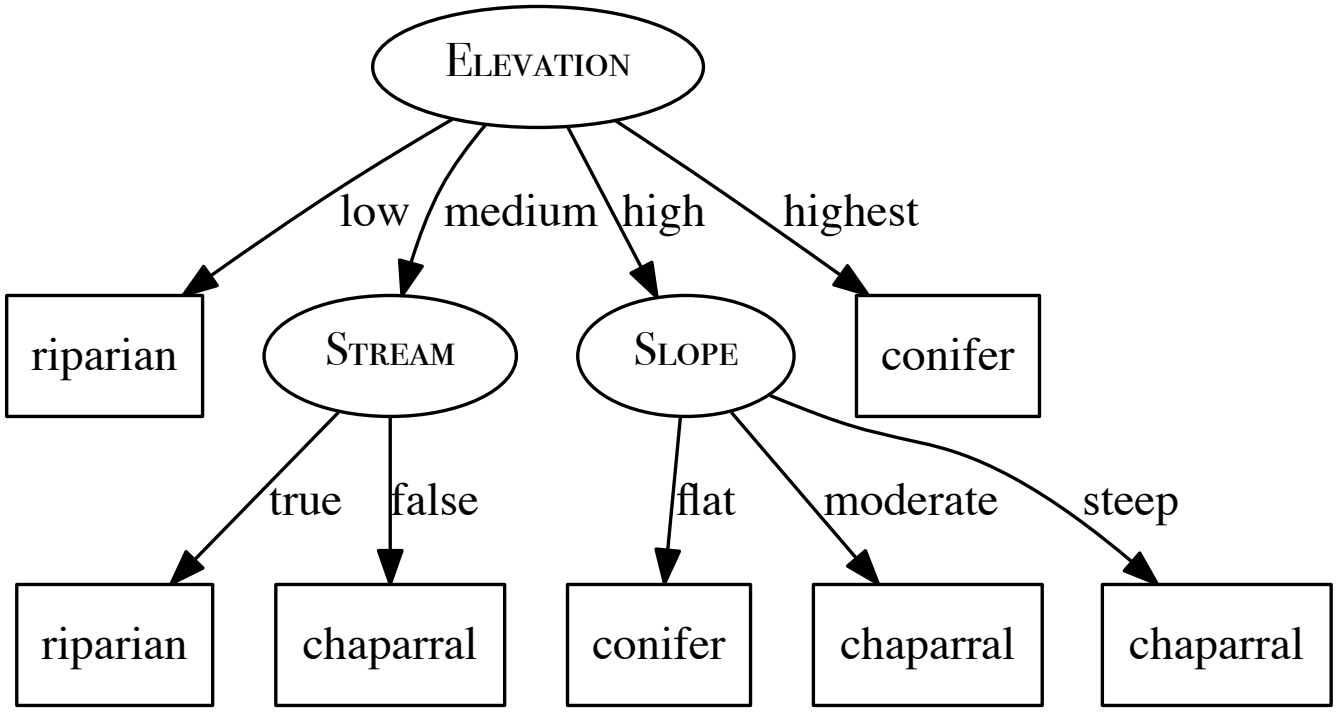
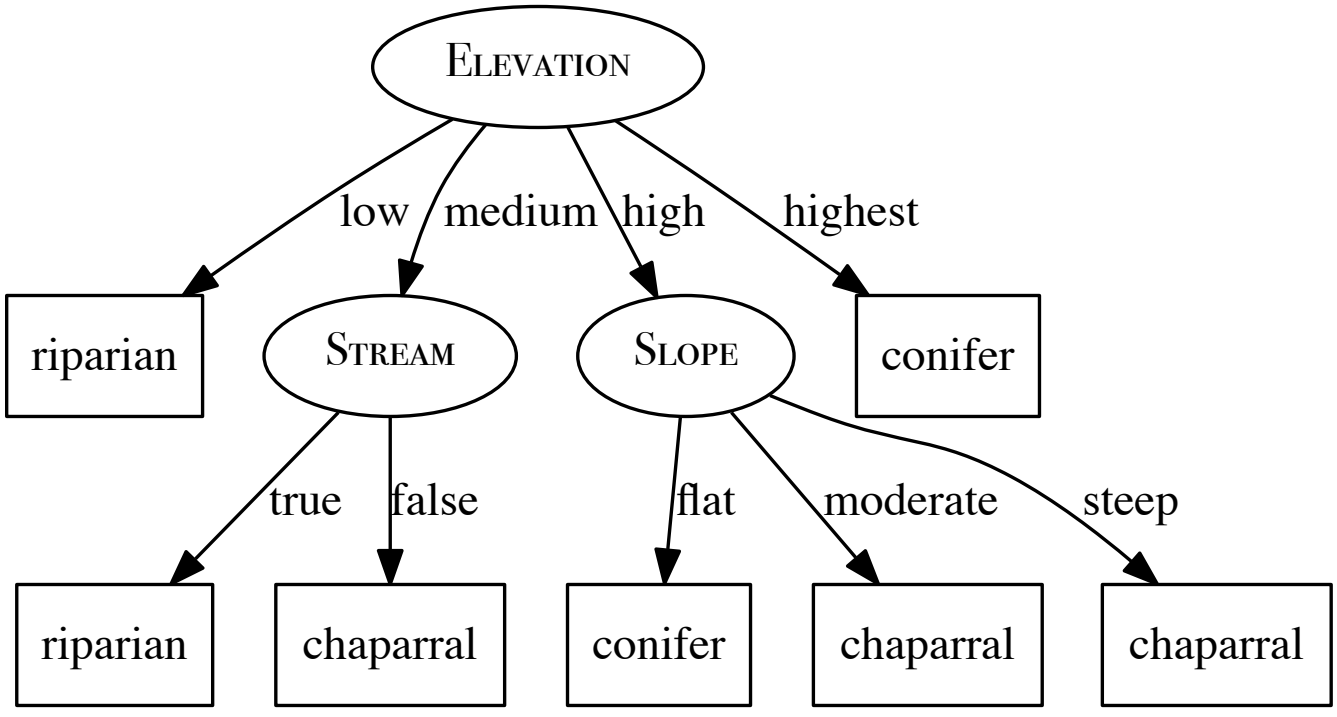


Figure: The final vegetation classification decision tree.



- What prediction will this decision tree model return for the following query?

STREAM = *'true'*, SLOPE=*'Moderate'*, ELEVATION=*'High'*



- What prediction will this decision tree model return for the following query?

STREAM = *'true'*, SLOPE=*'Moderate'*, ELEVATION=*'High'*

VEGETATION = *'Chaparral'*

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

STREAM = *'true'*, SLOPE=*'Moderate'*, ELEVATION=*'High'*

VEGETATION = *'Chaparral'*

- This is an example where the model is attempting to generalize beyond the dataset.
- Whether or not the generalization is correct depends on whether the assumptions used in generating the model (i.e. the inductive bias) were appropriate.

Summary

- The ID3 algorithm works the same way for larger more complicated datasets.
- There have been many extensions and variations proposed for the ID3 algorithm:
  - using different impurity measures (Gini, Gain Ratio)
  - handling continuous descriptive features
  - to handle continuous targets
  - pruning to guard against over-fitting
  - using decision trees as part of an ensemble (Random Forests)
- We cover these extensions in Section 4.4.

Information-based Learning

Sections 4.4, 4.5

John D. Kelleher and Brian Mac Namee and Aoife D'Arcy

Alternative Feature Selection Metrics

- 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:

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

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

$$
\begin{aligned}
&H(\textsc{Stream}, \mathcal{D}) \\
&= - \sum_{l \in \left\{ \substack{true, \\ false} \right\}} P(\textsc{Stream} = l) \times log_2 \left(P(\textsc{Stream} = l) \right) \\
&= - \left(\left({}^4/_7 \times log_2({}^4/_7) \right) + \left({}^3/_7 \times log_2({}^3/_7) \right) \right) \\
&= 0.9852 \; bits
\end{aligned}
$$

$$
\begin{aligned}
&H(\textsc{Slope}, \mathcal{D}) \\
&= - \sum_{l \in \left\{ \substack{flat, \\ moderate, \\ steep} \right\}} P(\textsc{Slope} = l) \times log_2 \left(P(\textsc{Slope} = l) \right) \\
&= - \left(\left({}^1/_7 \times log_2({}^1/_7) \right) + \left({}^1/_7 \times log_2({}^1/_7) \right) + \left({}^5/_7 \times log_2({}^5/_7) \right) \right) \\
&= 1.1488 \; bits
\end{aligned}
$$

$$
\begin{aligned}
&H(\textsc{Elevation}, \mathcal{D}) \\
&= - \sum_{l \in \left\{ \substack{low, \\ medium, \\ high, \\ highest} \right\}} P(\textsc{Elevation} = l) \times log_2 \left(P(\textsc{Elevation} = l) \right) \\
&= - \left(\left({}^1/_7 \times log_2({}^1/_7) \right) + \left({}^2/_7 \times log_2({}^2/_7) \right) + \left({}^3/_7 \times log_2({}^3/_7) \right) + \left({}^1/_7 \times log_2({}^1/_7) \right) \right) \\
&= 1.8424 \; bits
\end{aligned}
$$

$$GR(\text{STREAM}, \mathcal{D}) = \frac{0.3060}{0.9852} = 0.3106$$

$$GR(\text{SLOPE}, \mathcal{D}) = \frac{0.5774}{1.1488} = 0.5026$$

$$GR(\text{ELEVATION}, \mathcal{D}) = \frac{0.8774}{1.8424} = 0.4762$$

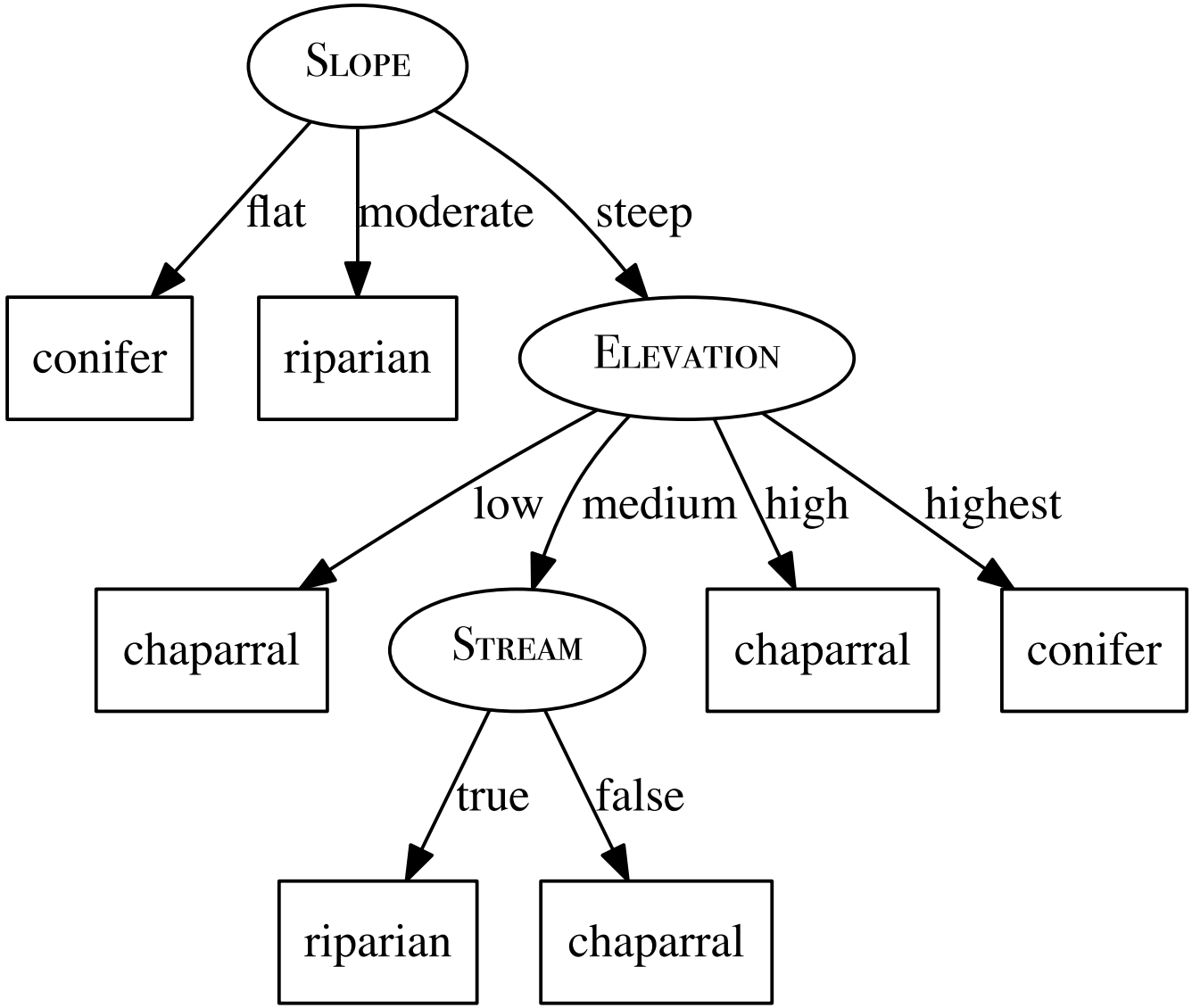


Figure 1: The vegetation classification decision tree generated using information gain ratio

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

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

- The Gini index can be thought of as calculating how often you would misclassify an instance in the dataset if you classified it based on the distribution of classifications in the dataset.
- Information gain can be calculated using the Gini index by replacing the entropy measure with the Gini index.

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

Table 1: Partition sets (Part.), entropy, Gini index, remainder (Rem.), and information gain (Info. Gain) by feature

Split by Feature	Level	Part.	Instances	Partition Gini Index	Rem.	Info. Gain
STREAM	*true*	$\mathcal{D}_1$	$\mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_6, \mathbf{d}_7$	0.625	0.5476	0.1054
	false	$\mathcal{D}_2$	$\mathbf{d}_1, \mathbf{d}_4, \mathbf{d}_5$	0.4444		
SLOPE	*flat*	$\mathcal{D}_3$	$\mathbf{d}_5$	0	0.4	0.2531
	moderate	$\mathcal{D}_4$	$\mathbf{d}_2$	0		
	steep	$\mathcal{D}_5$	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6, \mathbf{d}_7$	0.56		
ELEVATION	*low*	$\mathcal{D}_6$	$\mathbf{d}_2$	0	0.3333	0.3198
	medium	$\mathcal{D}_7$	$\mathbf{d}_3, \mathbf{d}_4$	0.5		
	high	$\mathcal{D}_8$	$\mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_7$	0.4444		
	highest	$\mathcal{D}_9$	$\mathbf{d}_6$	0		

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.
- How do we set the threshold?

1. The instances in the dataset are sorted according to the continuous feature values.
2. The adjacent instances in the ordering that have different classifications are then selected as possible threshold points.
3. 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.

- Once a threshold has been set the dynamically created new boolean feature can compete with the other categorical features for selection as the splitting feature at that node.
- This process can be repeated at each node as the tree grows.

Table 2: Dataset for predicting the vegetation in an area with a continuous ELEVATION feature (measured in feet).

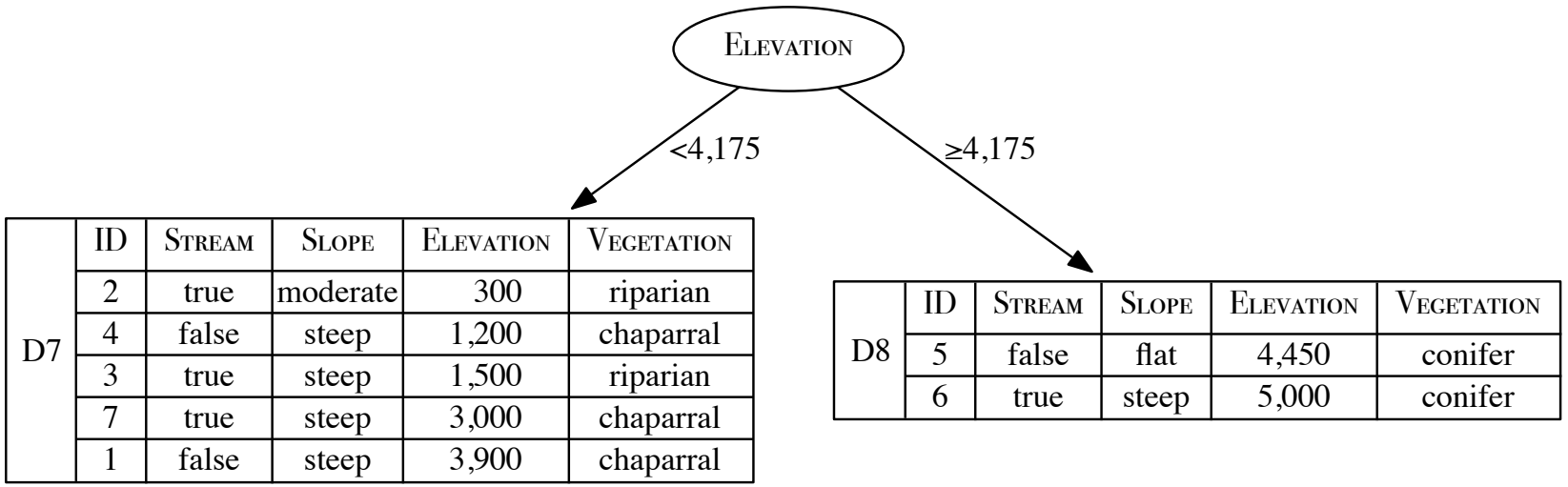
ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	3 900	chapparal
2	true	moderate	300	riparian
3	true	steep	1 500	riparian
4	false	steep	1 200	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer
7	true	steep	3 000	chapparal

Table 3: Dataset for predicting the vegetation in an area sorted by the continuous ELEVATION feature.

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

Table 4: 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		



D7

ID	STREAM	SLOPE	ELEVATION	VEGETATION
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

D8

ID	STREAM	SLOPE	ELEVATION	VEGETATION
5	false	flat	4,450	conifer
6	true	steep	5,000	conifer

Figure 2: The vegetation classification decision tree after the dataset has been split using ELEVATION $\geq$ 4 175.

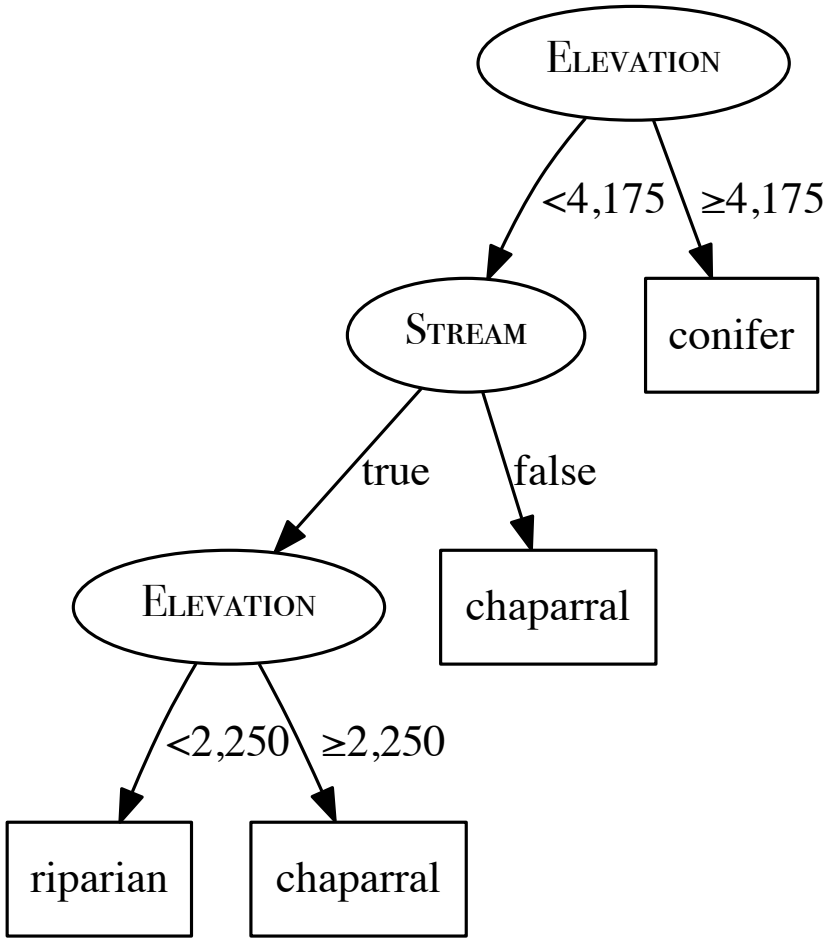


Figure 3: The decision tree that would be generated for the vegetation classification dataset listed in Table 3[17] using information gain.

Predicting Continuous Targets

- Regression trees are constructed so as to reduce the variance in the set of training examples at each of the leaf nodes in the tree
- We can do this by adapting the ID3 algorithm to use a measure of variance rather than a measure of classification impurity (entropy) when selecting the best attribute

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

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

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

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

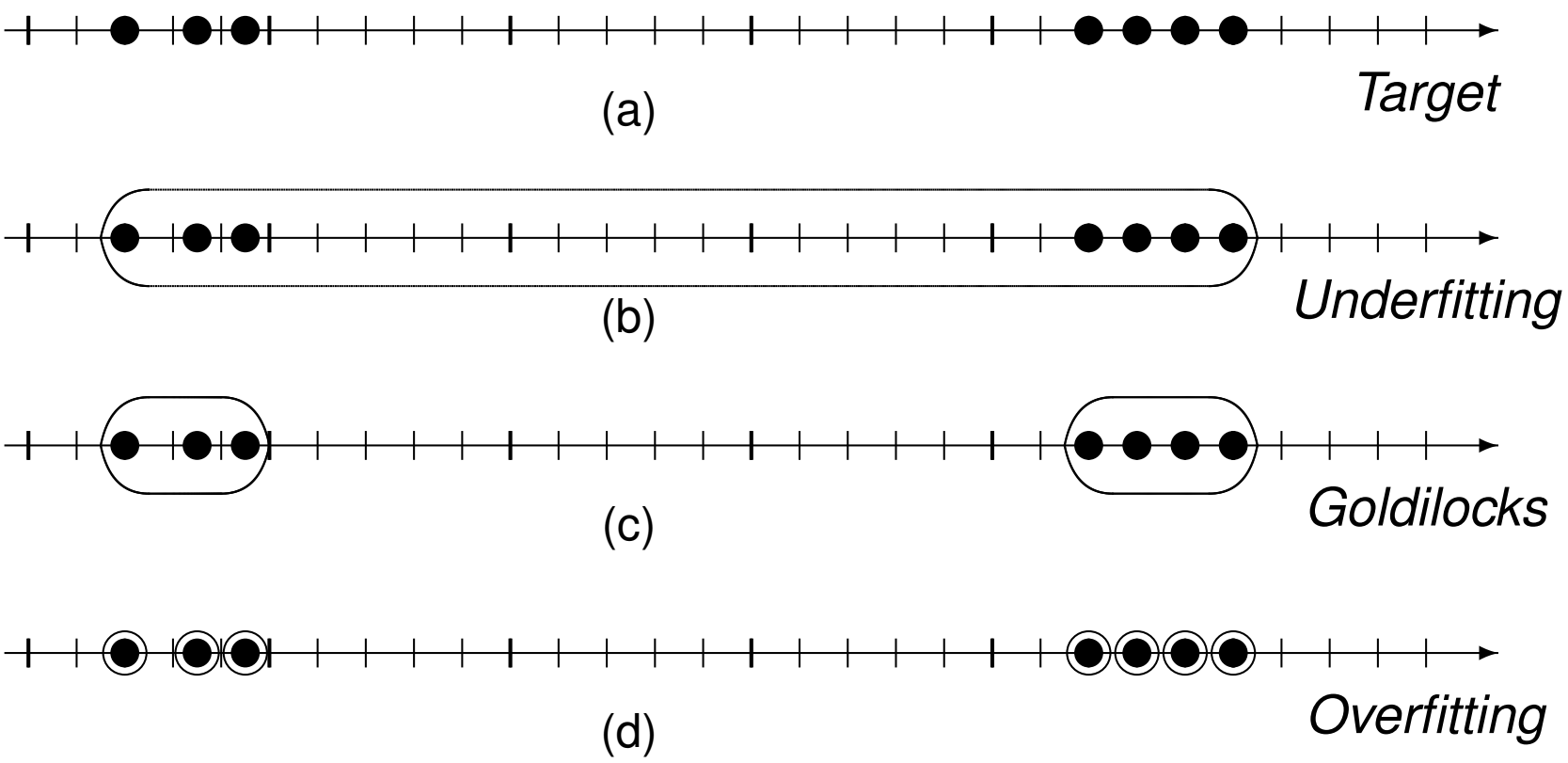


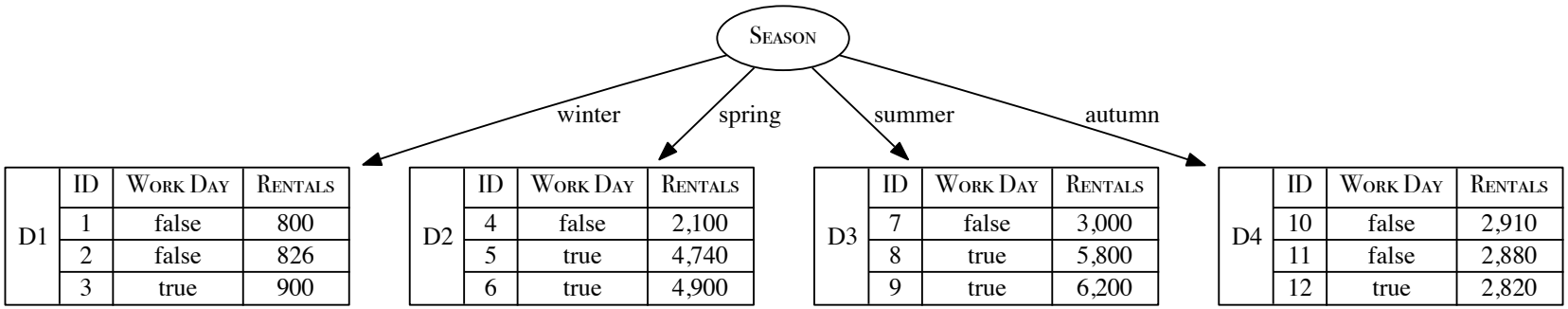
Figure 4: (a) A set of instances on a continuous number line; (b), (c), and (d) depict some of the potential groupings that could be applied to these instances.

Table 5: A dataset listing the number of bike rentals per day.

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
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

Table 6: The partitioning of the dataset in Table 5[25] based on SEASON and WORK DAY features and the computation of the weighted variance for each partitioning.

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	



D1

ID	WORK DAY	RENTALS
1	false	800
2	false	826
3	true	900

D2

ID	WORK DAY	RENTALS
4	false	2,100
5	true	4,740
6	true	4,900

D3

ID	WORK DAY	RENTALS
7	false	3,000
8	true	5,800
9	true	6,200

D4

ID	WORK DAY	RENTALS
10	false	2,910
11	false	2,880
12	true	2,820

Figure 5: The decision tree resulting from splitting the data in Table 5[25] using the feature SEASON.

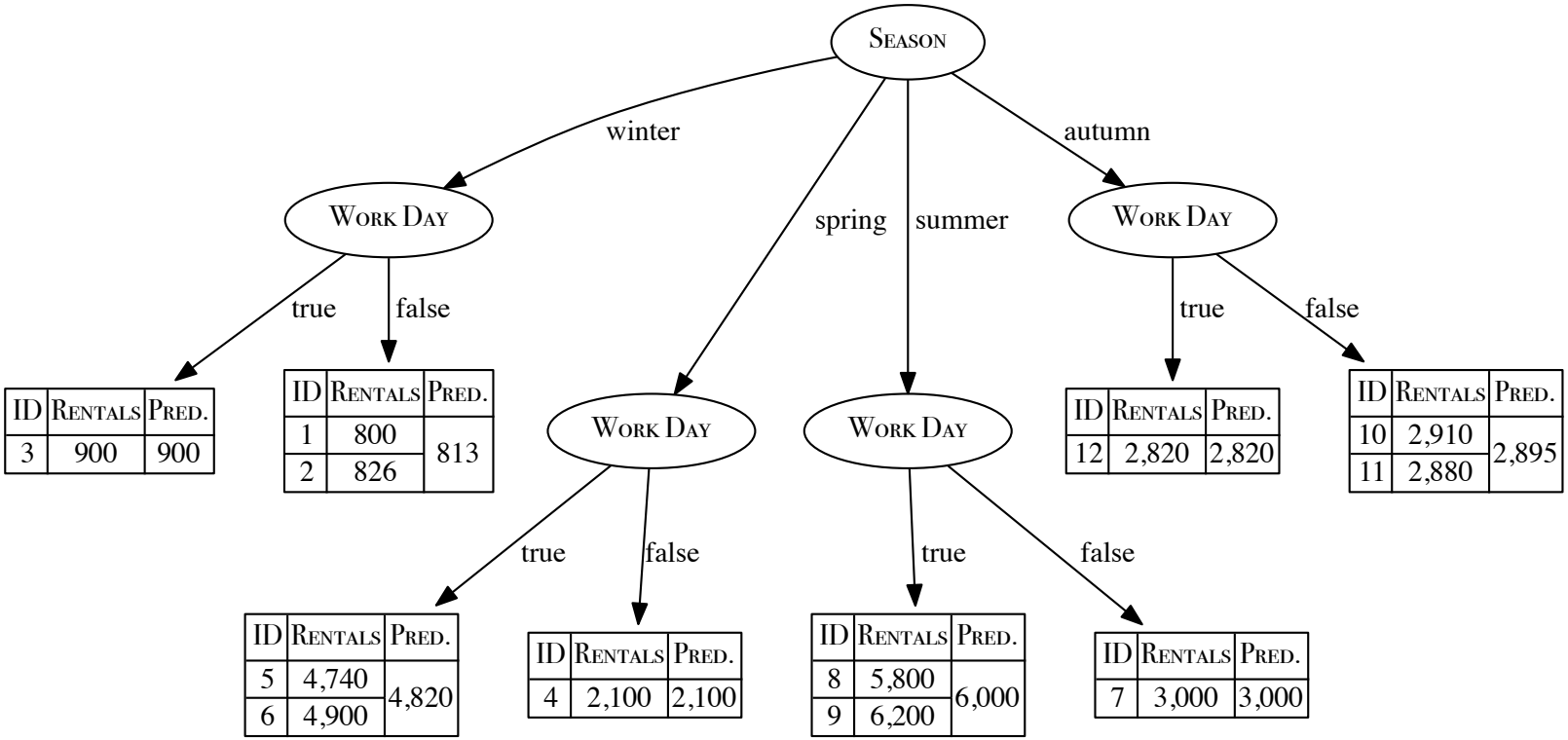


Figure 6: The final decision tree induced from the dataset in Table 5[25]. To illustrate how the tree generates predictions, this tree lists the instances that ended up at each leaf node and the prediction (PRED.) made by each leaf node.

Noisy Data, Overfitting and Tree Pruning

- 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.

- **Pre-pruning**: stop the recursive partitioning early. Pre-pruning is also known as **forward pruning**.

Common Pre-pruning Approaches

1. **early stopping**
2. χ^2 **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.

Common 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.

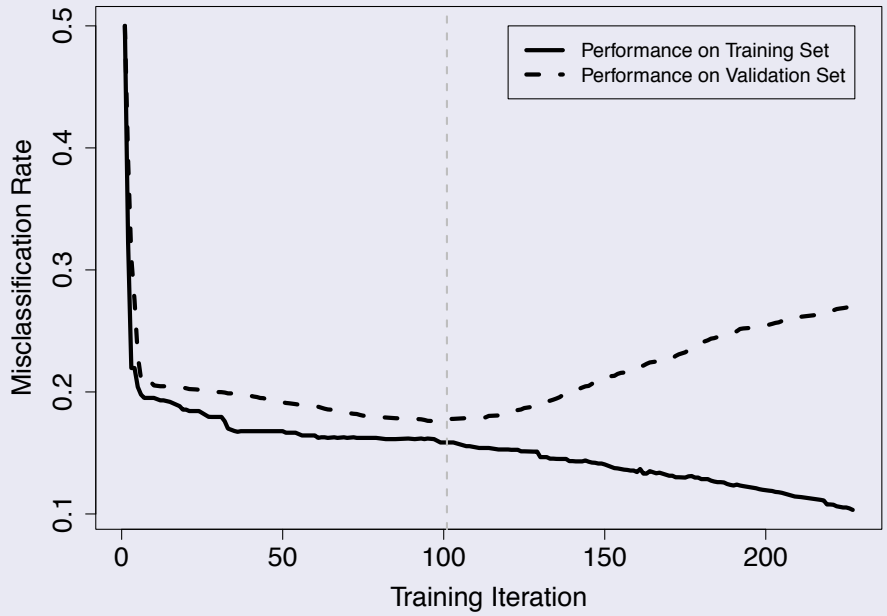


Table 7: An example validation set for the post-operative patient routing task.

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

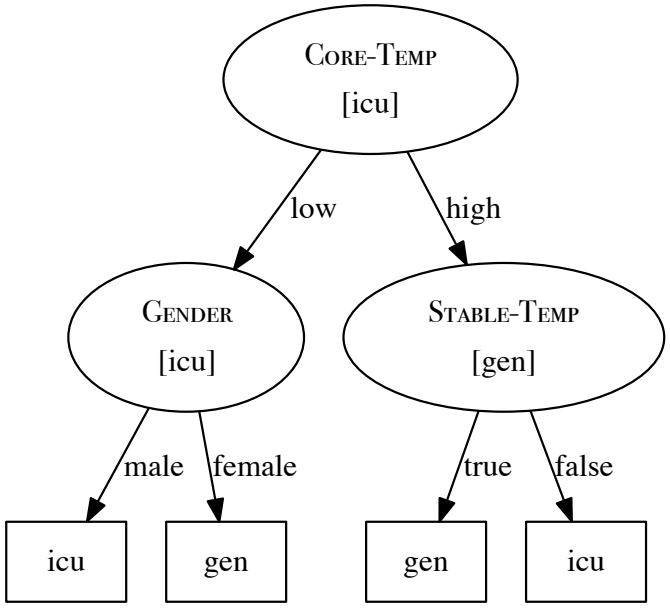


Figure 7: The decision tree for the post-operative patient routing task.

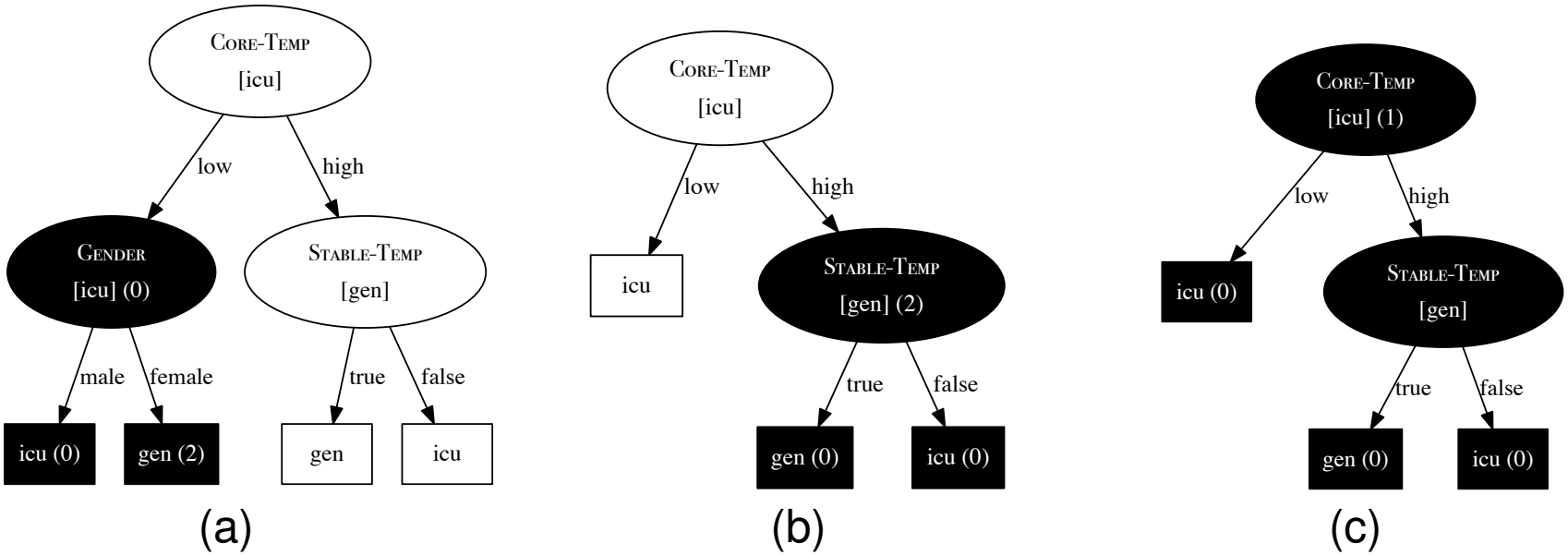


Figure 8: The iterations of reduced error pruning for the decision tree in Figure 7[34] using the validation set in Table 7[33]. The subtree that is being considered for pruning in each iteration is highlighted in black. The prediction returned by each non-leaf node is listed in square brackets. The error rate for each node is given in round brackets.

Advantages of pruning:

- Smaller trees are easier to interpret
- Increased generalization accuracy when there is noise in the training data (**noise dampening**).

Model Ensembles

- Rather than creating a single model they generate a set of models and then make predictions by aggregating the outputs of these models.
- A prediction model that is composed of a set of models is called a **model ensemble**.
- In order for this approach to work the models that are in the ensemble must be different from each other.

- There are two standard approaches to creating ensembles:
  1. **bagging**.
  2. **boosting**

- When we use **bagging** (or **bootstrap aggregating**) each model in the ensemble is trained on a random sample of the dataset known as **bootstrap samples**.
- Each random sample is the same size as the dataset and **sampling with replacement** is used.
- Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different

- When bagging is used with decision trees each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This is known as **subspace sampling**.
- The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model.

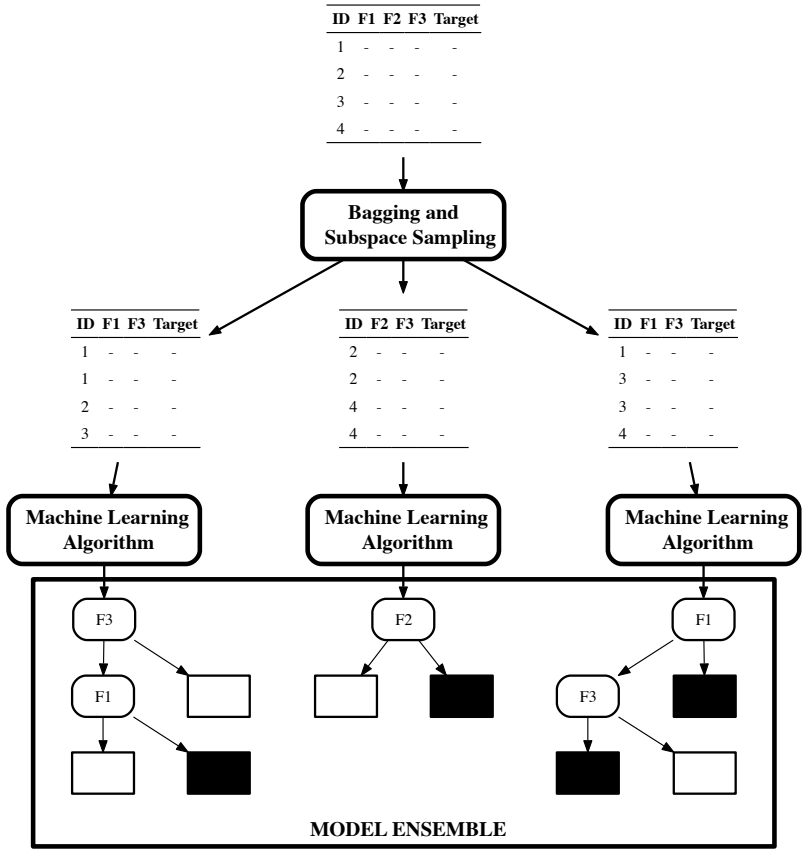


Figure 9: The process of creating a model ensemble using bagging and subspace sampling.

- Boosting works by iteratively creating models and adding them to the ensemble.
- The iteration stops when a predefined number of models have been added.
- When we use **boosting** each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified.
- This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset**

Weighted Dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $\frac{1}{n}$ where n is the number of instances in the dataset.
- After each model is added to the ensemble it is tested on the training data and the weights of the instances the model gets correct are decreased and the weights of the instances the model gets incorrect are increased.
- These weights are used as a distribution over which the dataset is sampled to created a replicated training set, where the replication of an instance is proportional to its weight.

During each training iteration the algorithm:

1. Induces a model and calculates the total error, ϵ, by summing the weights of the training instances for which the predictions made by the model are incorrect.
2. Increases the weights for the instances misclassified using:

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times \epsilon}\right) \tag{5}$$

3. Decreases the weights for the instances correctly classified:

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times (1 - \epsilon)}\right) \tag{6}$$

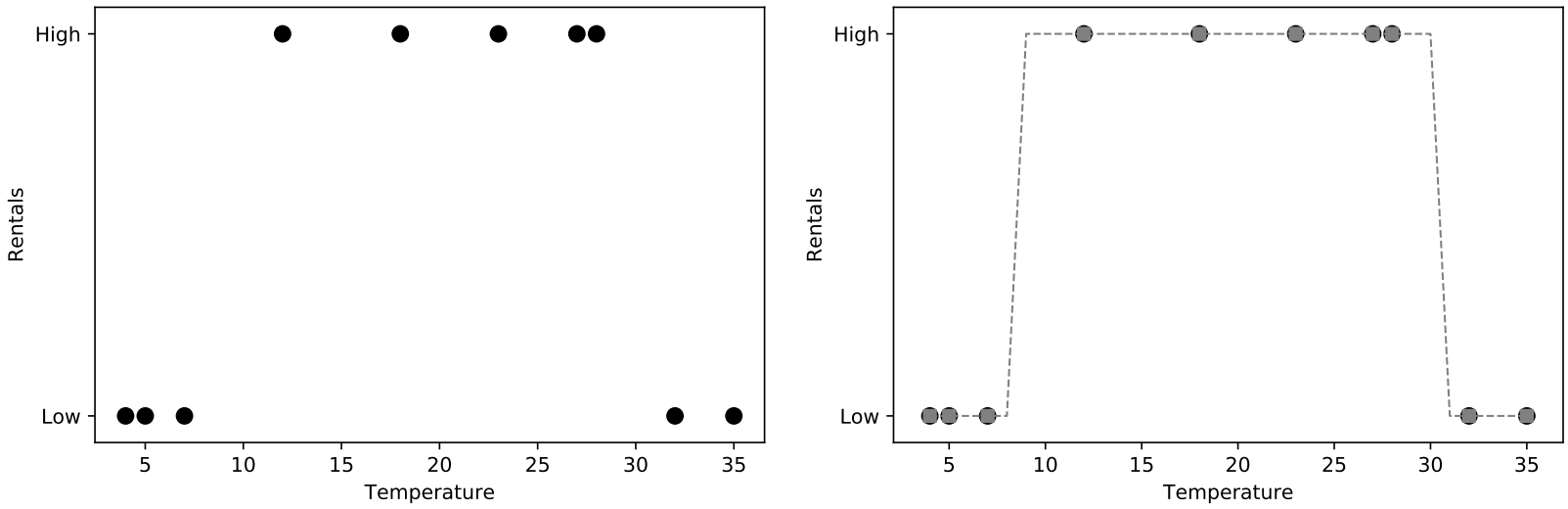
4. Calculate a **confidence factor**, α, for the model such that α increases as ϵ decreases:

$$\alpha = \frac{1}{2} \times log_e \left(\frac{1 - \epsilon}{\epsilon}\right) \tag{7}$$

- Once the set of models have been created the ensemble makes predictions using a weighted aggregate of the predictions made by the individual models.
- The weights used in this aggregation are simply the confidence factors associated with each model.

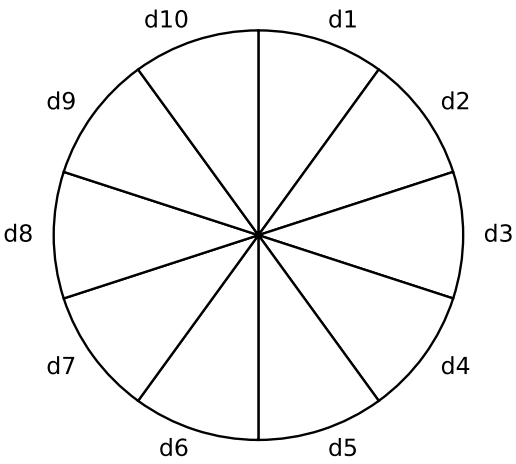
Table 8: A simple bicycle demand predictions dataset and the workings of the first three iterations of training an ensemble model using boosting to predict RENTALS given TEMP.

			Iteration 0			Iteration 1			Iteration 2		
ID	TEMP	RENTALS	Dist.	Freq.	$\mathbb{M}_0(\mathbf{d})$	Dist.	Freq.	$\mathbb{M}_1(\mathbf{d})$	Dist.	Freq.	$\mathbb{M}_2(\mathbf{d})$
1	4	*Low*	0.100	2	*Low*	0.062	0	*High*	0.167	2	*Low*
2	5	*Low*	0.100	1	*Low*	0.062	1	*High*	0.167	1	*Low*
3	7	*Low*	0.100	0	*Low*	0.062	1	*High*	0.167	3	*Low*
4	12	*High*	0.100	1	*High*	0.062	2	*High*	0.038	0	*Low*
5	18	*High*	0.100	1	*High*	0.062	0	*High*	0.038	0	*Low*
6	23	*High*	0.100	1	*High*	0.062	0	*High*	0.038	0	*Low*
7	27	*High*	0.100	1	*High*	0.062	1	*High*	0.038	0	*Low*
8	28	*High*	0.100	1	*High*	0.062	1	*High*	0.038	1	*Low*
9	32	*Low*	0.100	2	*High*	0.250	3	*Low*	0.154	1	*Low*
10	35	*Low*	0.100	0	*High*	0.250	1	*Low*	0.154	2	*Low*

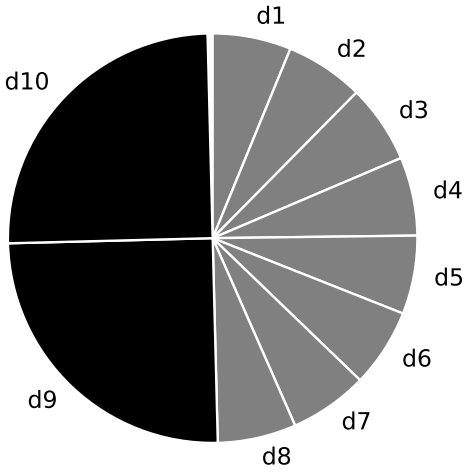


(a) Training data (b) The final ensemble model, $\mathbb{M}$

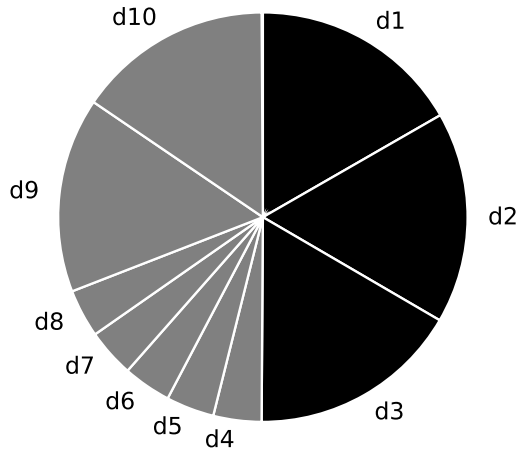
Figure 10: (a) A plot of the bike rental dataset from Table 8[47]. (b) An illustration of the final ensemble model trained using the boosting algorithm. (c)–(e) A representation of the changing weights used to generate sample datasets for the first iterations of the boosting process.



(c) Distribution 0



(d) Distribution 1



(e) Distribution 2

Figure 11: (a) A plot of the bike rental dataset from Table 8[47]. (b) An illustration of the final ensemble model trained using the boosting algorithm. (c)–(e) A representation of the changing weights used to generate sample datasets for the first iterations of the boosting process.

$$\mathbf{w}[1] \leftarrow 0.100 \times \left(\frac{1}{2 \times (1 - 0.200)} \right) \leftarrow 0.0625$$

$$\mathbf{w}[9] \leftarrow 0.100 \times \left(\frac{1}{2 \times 0.200} \right) \leftarrow 0.250$$

$$\mathbb{M}_0(\mathbf{d}) = \frac{1}{n}\sum_i t_i \tag{8}$$

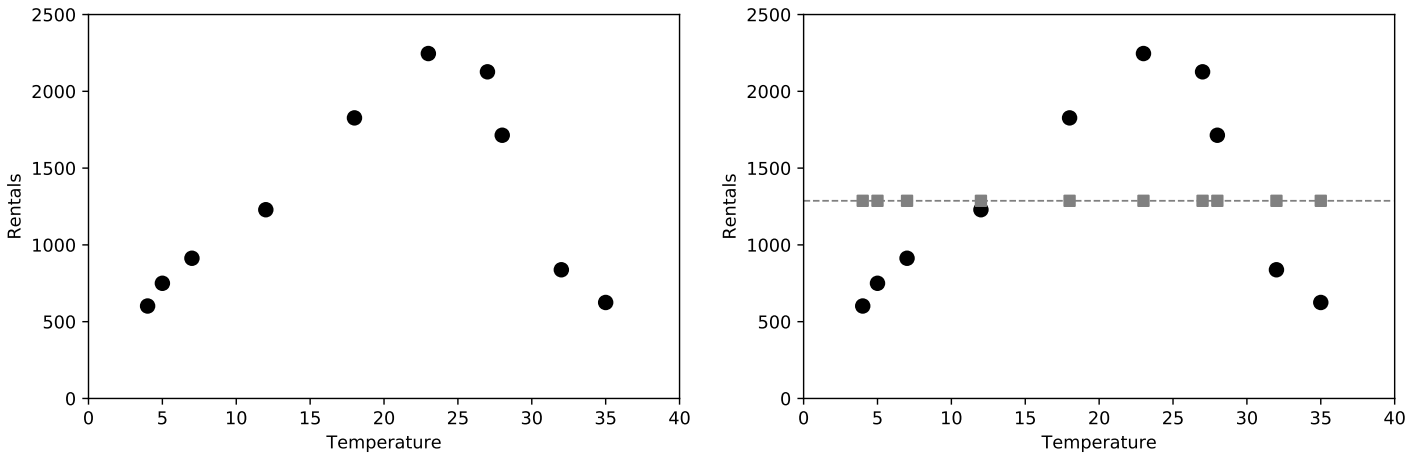
$$\mathbb{M}_1(\mathbf{d}) = \mathbb{M}_0(\mathbf{d}) + \mathbb{M}_{\Delta 1}(\mathbf{d}) \tag{9}$$

$$\mathbb{M}_i(\mathbf{d}) = \mathbb{M}_{i-1}(\mathbf{d}) + \mathbb{M}_{\Delta i}(\mathbf{d}) \tag{10}$$

$$
\begin{aligned}
\mathbb{M}_4(\mathbf{d}) &= \mathbb{M}_3(\mathbf{d}) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\
&= (\mathbb{M}_2(\mathbf{d}) + \mathbb{M}_{\Delta 3}(\mathbf{d})) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\
&= ((\mathbb{M}_1 + \mathbb{M}_{\Delta 2}(\mathbf{d})) + \mathbb{M}_{\Delta 3}(\mathbf{d})) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\
&= (((\mathbb{M}_0(\mathbf{d}) + \mathbb{M}_{\Delta 1}(\mathbf{d})) + \mathbb{M}_{\Delta 2}(\mathbf{d})) + \mathbb{M}_{\Delta 3}(\mathbf{d})) + \mathbb{M}_{\Delta 4}(\mathbf{d}) \\
&= \mathbb{M}_0(\mathbf{d}) + \mathbb{M}_{\Delta 1}(\mathbf{d}) + \mathbb{M}_{\Delta 2}(\mathbf{d}) + \mathbb{M}_{\Delta 3}(\mathbf{d}) + \mathbb{M}_{\Delta 4}(\mathbf{d})
\end{aligned}
\tag{11}
$$

Table 9: A simple bicycle demand predictions dataset and the workings of the first iterations of training a gradient boosting model.

ID	TEMP	RENTALS	$\mathbb{M}_0(\mathbf{d})$	$t - \mathbb{M}_0(\mathbf{d})$	$\mathbb{M}_{\Delta 1}(\mathbf{d})$	$\mathbb{M}_1(\mathbf{d})$	$t - \mathbb{M}_1(\mathbf{d})$	$\mathbb{M}_{\Delta 2}(\mathbf{d})$	$\mathbb{M}_2(\mathbf{d})$	$t - \mathbb{M}_2(\mathbf{d})$	$\mathbb{M}_{\Delta 3}(\mathbf{d})$	$\mathbb{M}_3(\mathbf{d})$
1	4	602	1 287.1	-685.1	-460.9	826.2	-224.2	-167.2	659.0	-57.0	-34.1	624.9
2	5	750	1 287.1	-537.1	-460.9	826.2	-76.2	-167.2	659.0	91.0	-34.1	624.9
3	7	913	1 287.1	-374.1	-460.9	826.2	86.8	71.6	897.8	15.2	-34.1	863.7
4	12	1229	1 287.1	-58.1	-460.9	826.2	402.8	71.6	897.8	331.2	-34.1	863.7
5	18	1827	1 287.1	539.9	691.4	1 978.5	-151.5	71.6	2 050.1	-223.1	-34.1	2 016.1
6	23	2246	1 287.1	958.9	691.4	1 978.5	267.5	71.6	2 050.1	195.9	136.4	2 186.5
7	27	2127	1 287.1	839.9	691.4	1 978.5	148.5	71.6	2 050.1	76.9	136.4	2 186.5
8	28	1714	1 287.1	426.9	691.4	1 978.5	-264.5	71.6	2 050.1	-336.1	-34.1	2 016.1
9	32	838	1 287.1	-449.1	-460.9	826.2	11.8	71.6	897.8	-59.8	-34.1	863.7
10	35	625	1 287.1	-662.1	-460.9	826.2	-201.2	-167.2	659.0	-34.0	-34.1	624.9



(a) Training data (b) $\mathbb{M}_0$

Figure 12: (a) A plot of the bike rental dataset from Table 9[53]. (b)–(e) Visualizations of the prediction models trained in the early iterations of the gradient boosting process. (f) The final ensemble model trained after 20 iterations of gradient boosting.

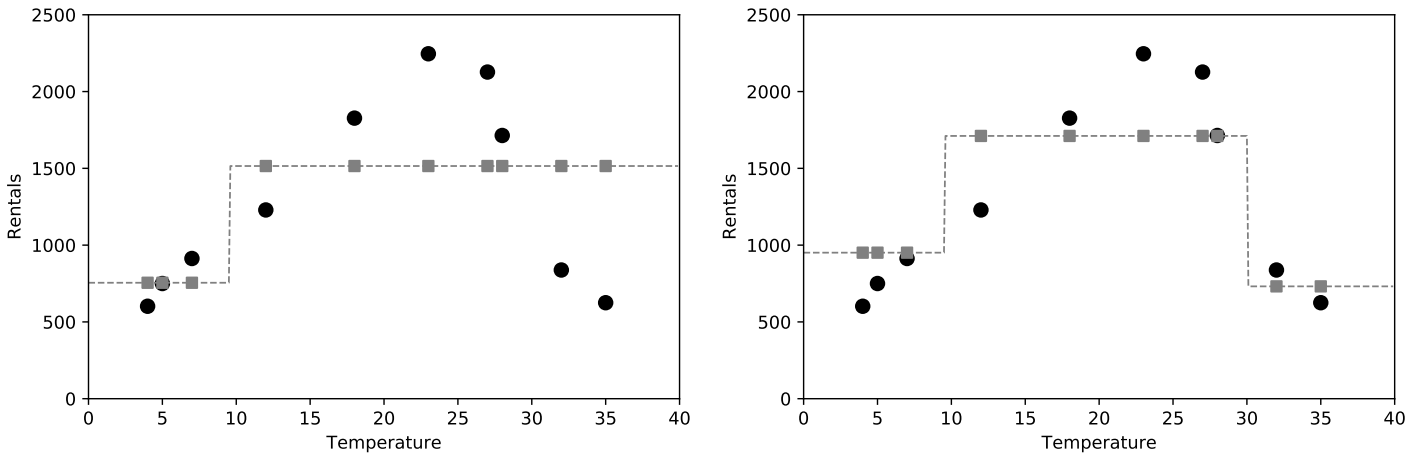
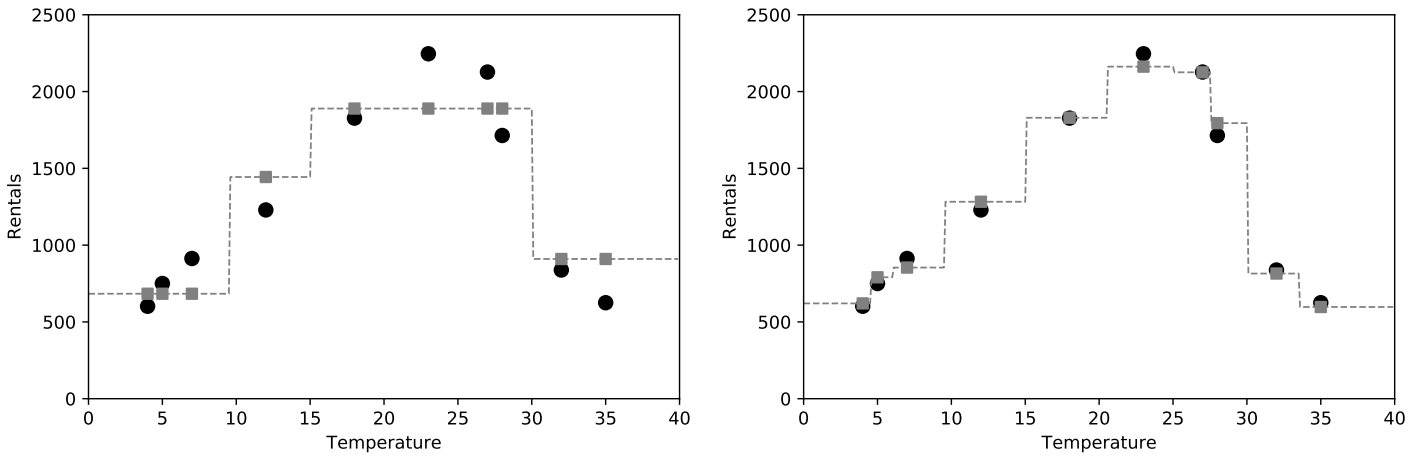


Figure 13: (a) A plot of the bike rental dataset from Table 9[53]. (b)–(e) Visualizations of the prediction models trained in the early iterations of the gradient boosting process. (f) The final ensemble model trained after 20 iterations of gradient boosting.



(e) $\mathbb{M}_3$ (f) $\mathbb{M}_{20}$

Figure 14: (a) A plot of the bike rental dataset from Table 9[53]. (b)–(e) Visualizations of the prediction models trained in the early iterations of the gradient boosting process. (f) The final ensemble model trained after 20 iterations of gradient boosting.

Gradient Boosting

$$\mathbb{M}_i(\mathbf{d}) = \mathbb{M}_{i-1}(\mathbf{d}) + \alpha \times \mathbb{M}_{\Delta i}(\mathbf{d}) \tag{12}$$

- Which approach should we use? Bagging is simpler to implement and parallelize than boosting and, so, may be better with respect to ease of use and training time.
- Empirical results indicate:
  - boosted decision tree ensembles were the best performing model of those tested for datasets containing up to 4,000 descriptive features.
  - random forest ensembles (based on bagging) performed better for datasets containing more that 4,000 features.

Summary

- The **decision tree** model makes predictions based on sequences of tests on the descriptive feature values of a query
- The **ID3** algorithm as a standard algorithm for inducing decision trees from a dataset.

Decision Trees: Advantages

- interpretable.
- handle both categorical and continuous descriptive features.
- has the ability to model the interactions between descriptive features (diminished if **pre-pruning** is employed)
- relatively, robust to the **curse of dimensionality**.
- relatively, robust to noise in the dataset if **pruning** is used.

Decision Tress: Potential Disadvantages

- trees become large when dealing with continuous features.
- decision trees are very expressive and sensitive to the dataset, as a result they can overfit the data if there are a lot of features (curse of dimensionality)
- eager learner (concept drift).