Decision trees: basic concepts

Mauricio A. Álvarez, PhD

Scalable Machine Learning,
University of Sheffield


LEFT
EXIT 12
LOGISTIC REGRESSION AND DECISION TREES THAT WORK PRETTY WELL
DEEP LEARNING THAT WE BARELY UNDERSTAND WITH A 1% ACCURACY LIFT
DATA SCIENTISTS
imgflip.com

Contents

Contents

Informative features

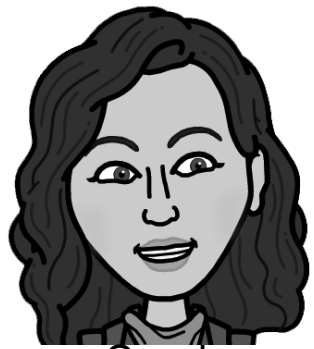
- A decision trees is a machine learning algorithm that tries to build predictive models **using the most informative features**.

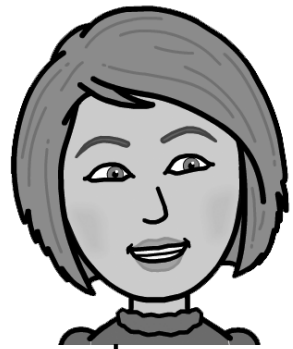
- An informative feature is a feature whose values split the instances in the dataset into **homogeneous sets** with respect to the target value.

Guess-who game


Brian


John

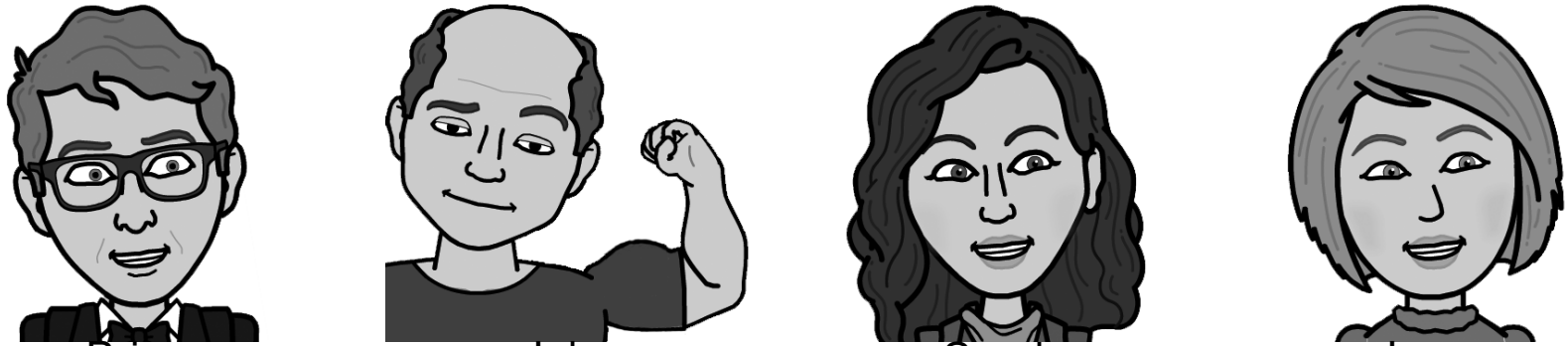

Sarah


Lucy

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	Sarah
No	No	No	Lucy

Guess-who game

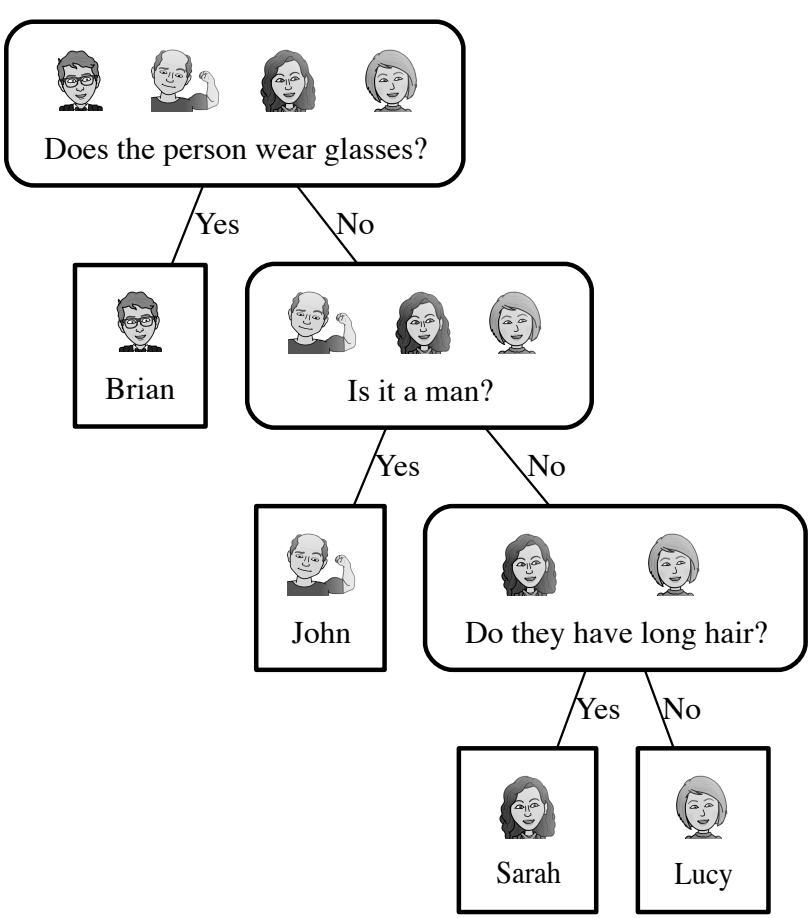


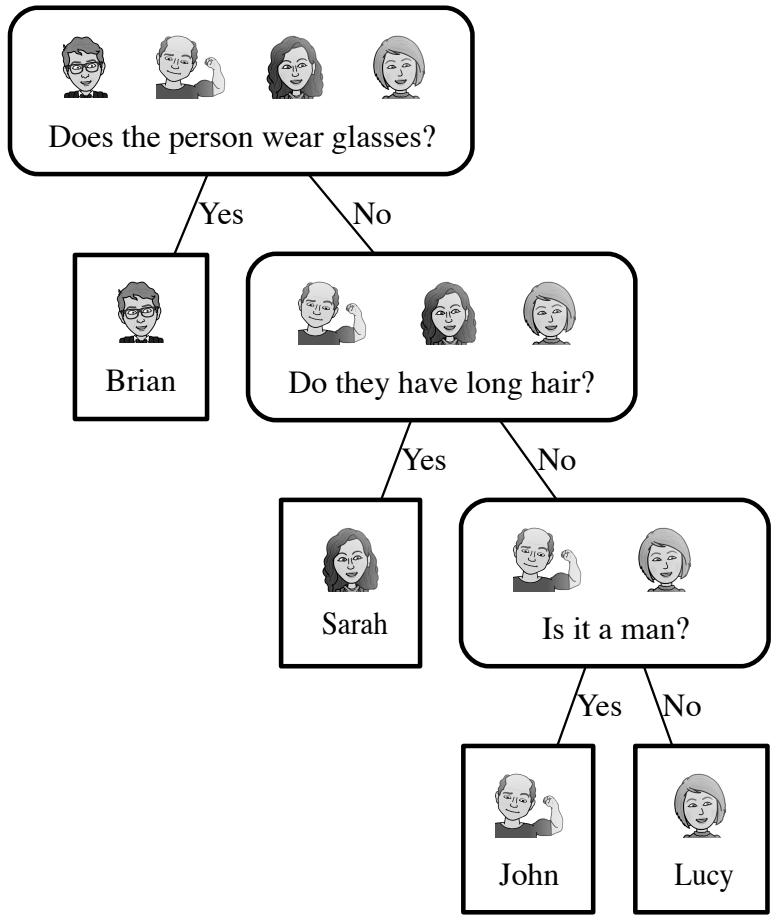
Cards showing character faces and names for the *Guess-Who* game.

Which question would you ask first:

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

Does the person wear glasses?





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

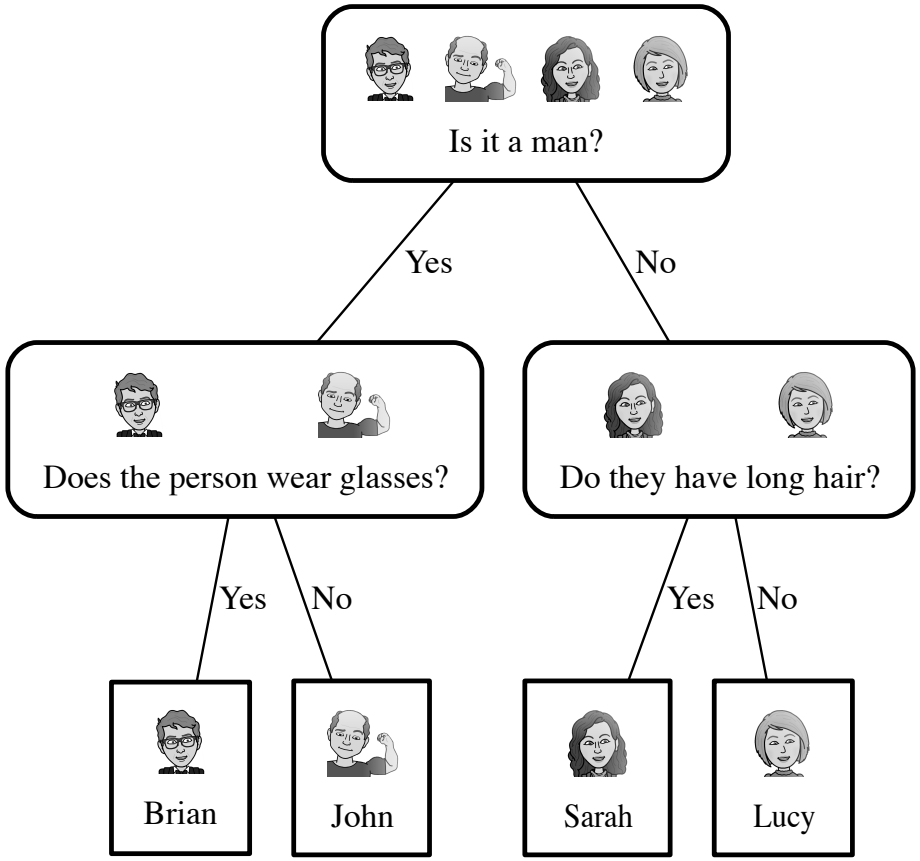
Number of questions you ask

- 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:

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

Is it a man?



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

Number of questions you ask

- 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.

Big Idea

- So the big idea here is to figure out which features are the most informative ones to ask questions about.

- We do that 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.

Contents

Nodes in a decision tree

- 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.

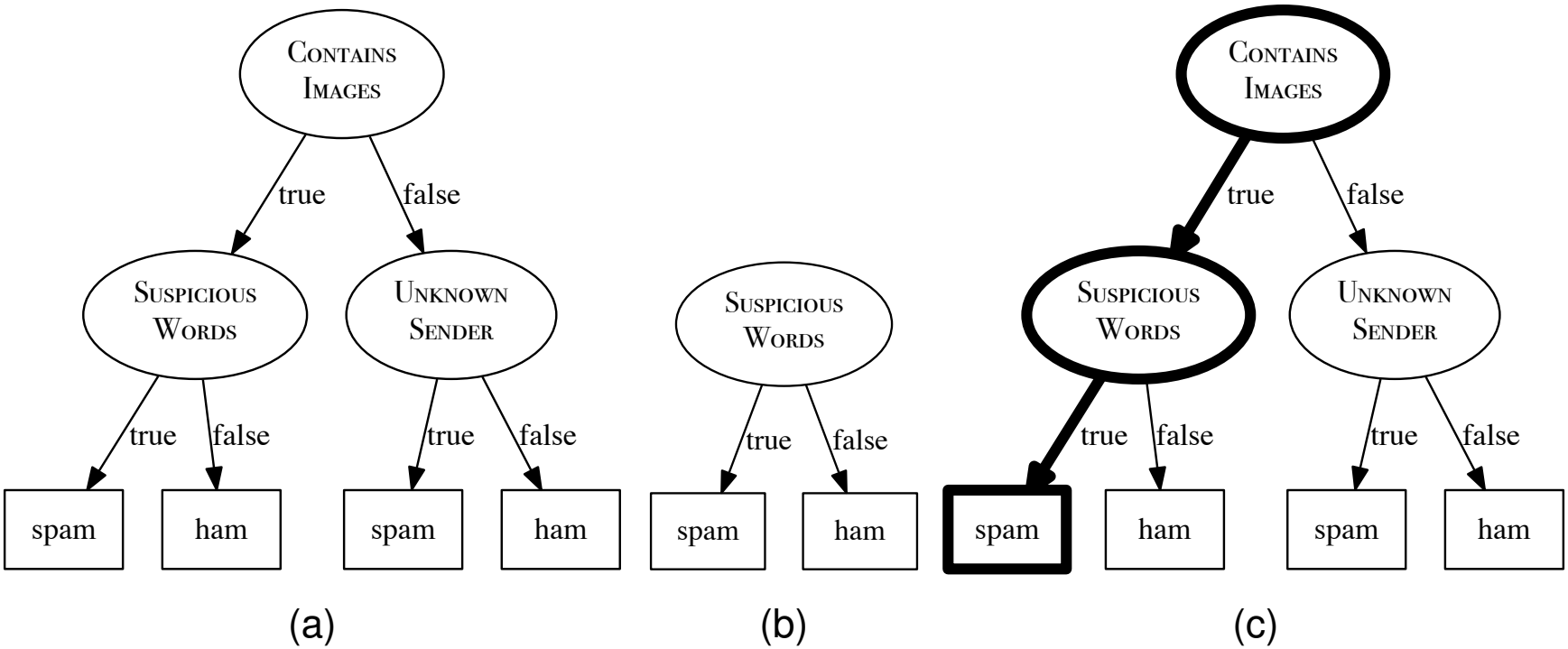
- Each of the leaf nodes specifies a predicted classification for the query.

An email spam prediction dataset

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

Two decision trees and a query instance



(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'*.

Which tree to use?

- Both of these trees will return identical predictions for all the examples in the dataset.

- So, which tree should we use?

Occam's Razor

- 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.
- Occam's Razor: *"Among competing hypotheses, the one with the fewest assumptions should be selected"*.

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

Contents

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.

Probability and entropy

- Entropy is related to the probability of an 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!

Mathematical definition of entropy

- 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.

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

Entropy in poker cards

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

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

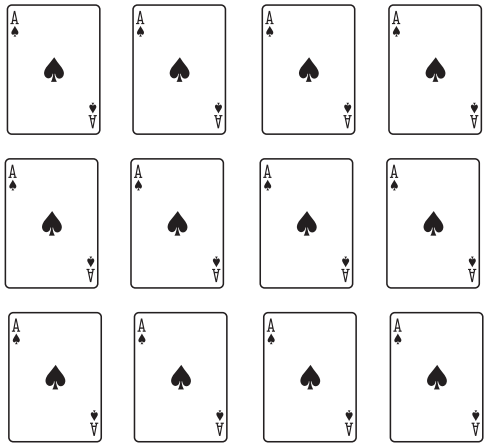
Entropy in poker cards

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

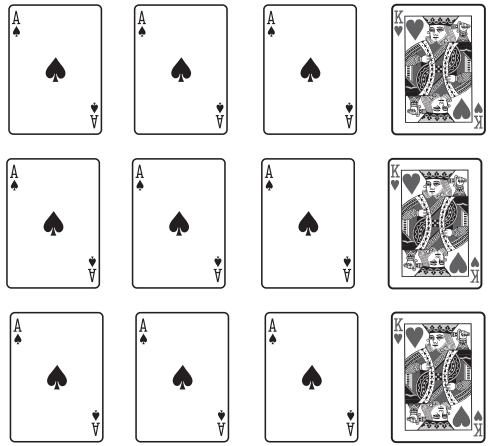
Entropy in poker cards

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

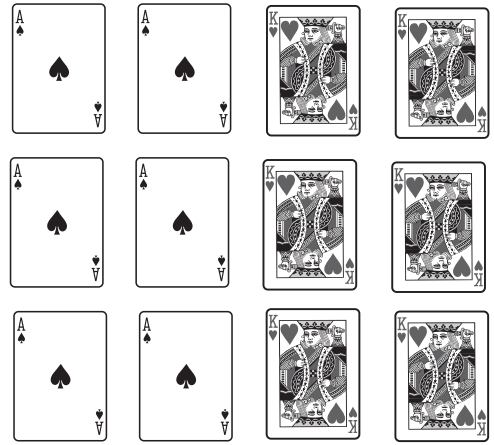
Different entropies



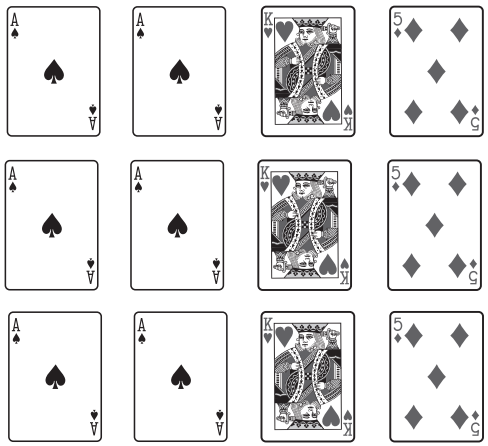
(a) $H(card) = 0.00$



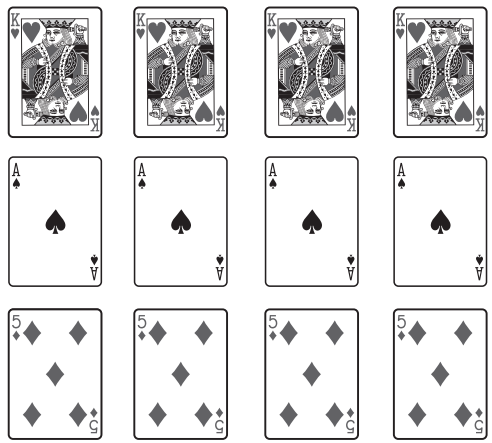
(b) $H(card) = 0.81$



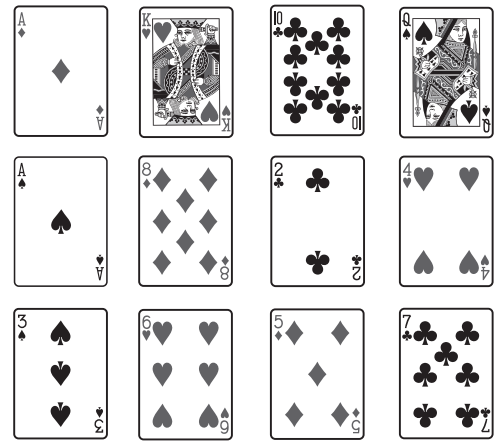
(c) $H(card) = 1.00$



(d) $H(card) = 1.50$



(e) $H(card) = 1.58$



(f) $H(card) = 3.58$

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

Entropy of a message

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.

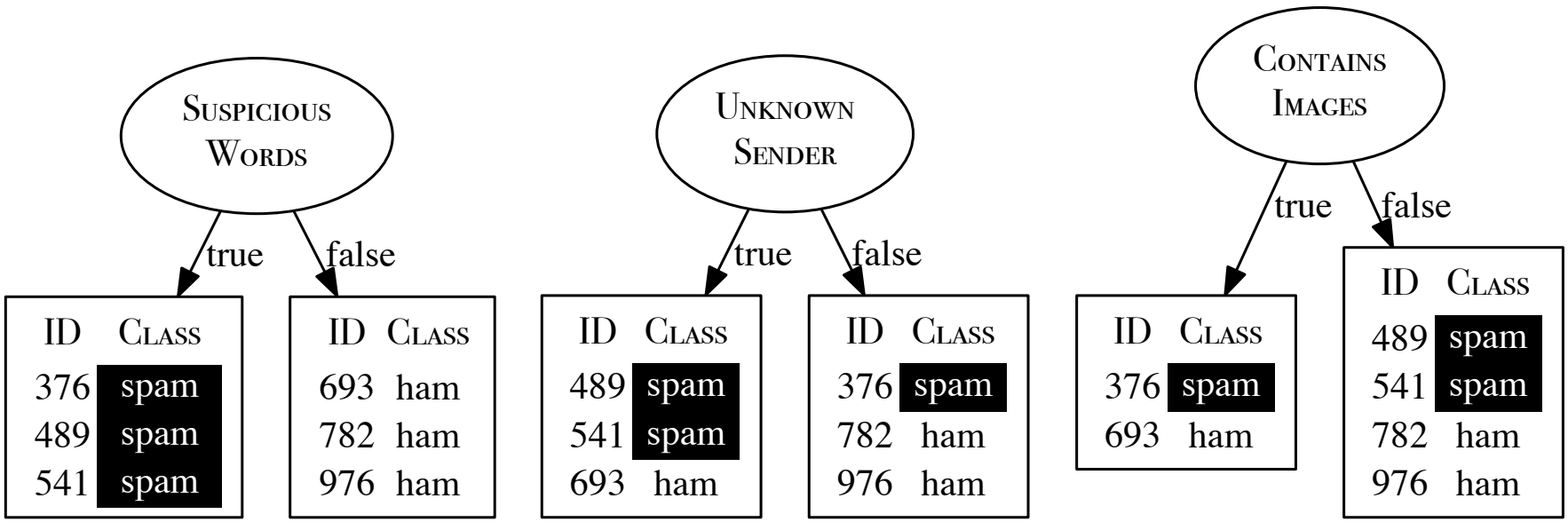
Contents

The spam dataset again

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

Partitions in the spam dataset



How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset table.

Pure subsets

- 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 the information gain

1. Compute the entropy of the original dataset with respect to the target feature. This gives us an measure of how much information is required in order to organize the dataset into pure sets.

2. For each descriptive feature, create the sets that result by partitioning the instances in the dataset using their feature values, and then sum the entropy scores of each of these sets. This gives a measure of the information that remains required to organize the instances into pure sets after we have split them using the descriptive feature.

3. Subtract the remaining entropy value (computed in step 2) from the original entropy value (computed in step 1) to give the information gain.

Computing the information gain

Computing information gain involves the following three equations:

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

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

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

Example with the spam dataset

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

Step 1: Entropy for the target feature

- 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

Step 1: Entropy for the target feature

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

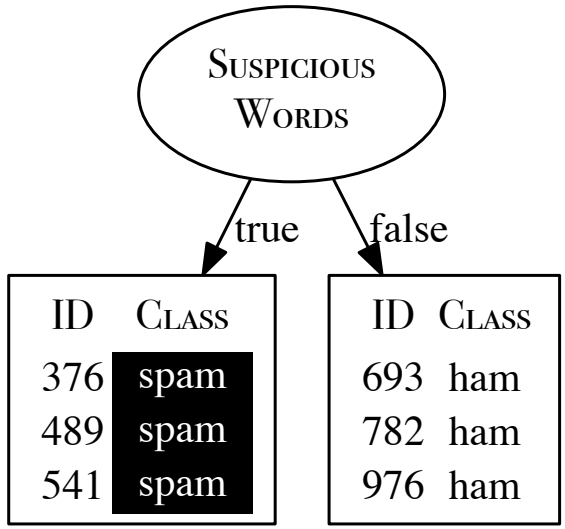
Step 2: Remainder for the SUSPICIOUS W. feature

- 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

Step 2: Partitions for the SUSPICIOUS W. feature



Partitions for the SUSPICIOUS W. feature

Step 2: Remainder for the SUSPICIOUS W. feature

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

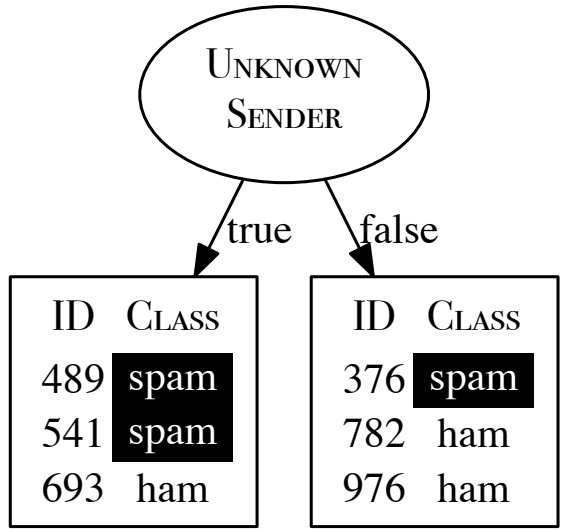
Step 2: Remainder for the UNKNOWN SENDER feature

- 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

Step 2: Partitions for the UNKNOWN SENDER feature



Partitions for the UNKNOWN SENDER feature

Step 2: Remainder for the UNKNOWN SENDER feature

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

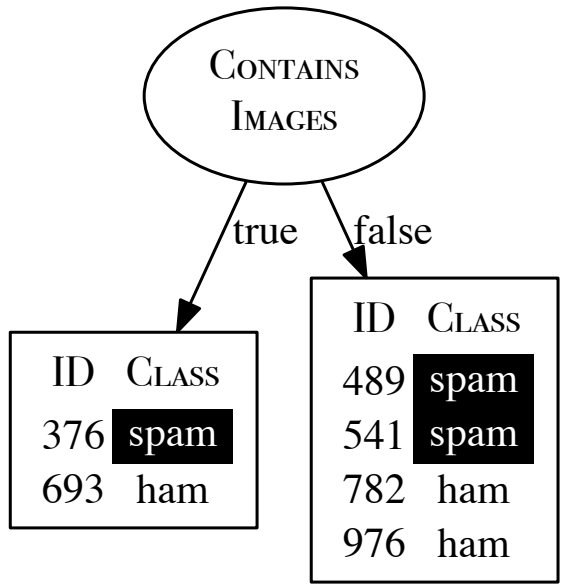
Step 2: Remainder for the CONTAINS IMAGES feature

- 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

Step 2: Partitions for the CONTAINS IMAGES feature



Partitions for the CONTAINS IMAGES feature

Step 2: Remainder for the CONTAINS IMAGES feature

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

Step 3: Compute the Information Gain

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

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

Step 3: Compute the Information Gain

$$
\begin{aligned}
IG(\textsc{Suspicious Words}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Suspicious Words}, \mathcal{D}) \\
&= 1 - 0 = 1 \textit{ bit} \\
IG(\textsc{Unknown Sender}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Unknown Sender}, \mathcal{D}) \\
&= 1 - 0.9183 = 0.0817 \textit{ bits} \\
IG(\textsc{Contains Images}, \mathcal{D}) &= H(\textsc{Class}, \mathcal{D}) - rem(\textsc{Contains Images}, \mathcal{D}) \\
&= 1 - 1 = 0 \textit{ bits}
\end{aligned}
$$

- The results of these calculations match our intuitions.

Contents

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.

How the different nodes are treated?

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.

Contents

Turn continuous features into boolean features

- The easiest way to handle continuous valued descriptive features is to turn them into boolean features by defining a threshold.
- The threshold is used to partition the instances based on their value of the continuous descriptive feature.
- How do we set the threshold?

Sorting the instances

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.

Treat the feature as a categorial feature

- 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.

Vegetation classification dataset

(a) chaparral veg.

(b) riparian veg.

(c) conifer veg.

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

Sorted instances

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

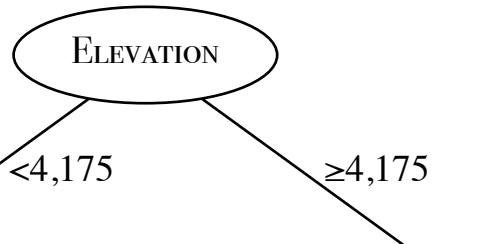
ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

Thresholds and partitions

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		

First split

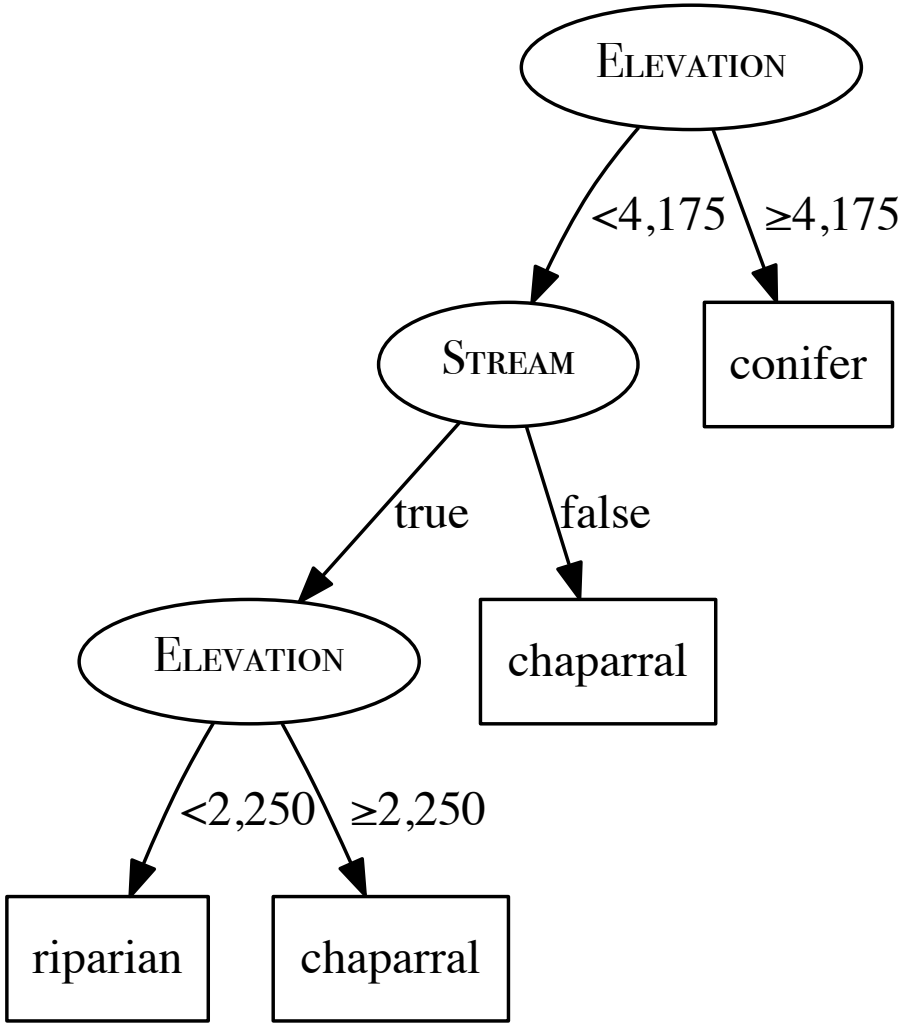


	ID	Stream	Slope	Elevation	Vegetation
D7	2	true	moderate	300	riparian
	4	false	steep	1,200	chaparral
	3	true	steep	1,500	riparian
	7	true	steep	3,000	chaparral
	1	false	steep	3,900	chaparral

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

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

Second split



The decision tree that would be generated for the vegetation classification dataset using information gain.

Contents

The `DecisionTreeClassifier` class

class `sklearn.tree.` **`DecisionTreeClassifier`** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*)

[source]

`DecisionTreeClassifier` class in scikit-learn

class `pyspark.ml.classification.`**`DecisionTreeClassifier`**(*featuresCol='features', labelCol='label', predictionCol='prediction', probabilityCol='probability', rawPredictionCol='rawPrediction', maxDepth=5, maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, impurity='gini', seed=None*)

[source]

`DecisionTreeClassifier` class in pyspark

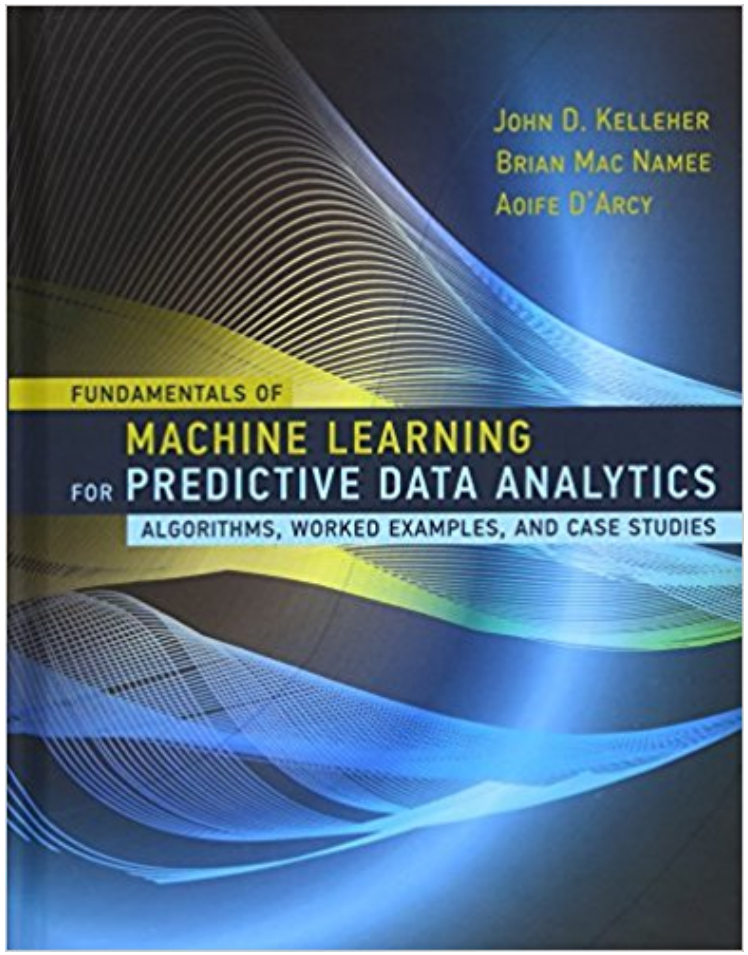
Parameters to adjust

- **`maxDepth`**: Maximum depth of a tree.
- **`maxBins`**: Max number of bins for discretizing continuous features. Must be $>=$ 2 and $>=$ number of categories for any categorical feature.
- **`impurity`**: Criterion used for information gain calculation (case-insensitive). Supported options: `entropy` or `gini`.

Contents

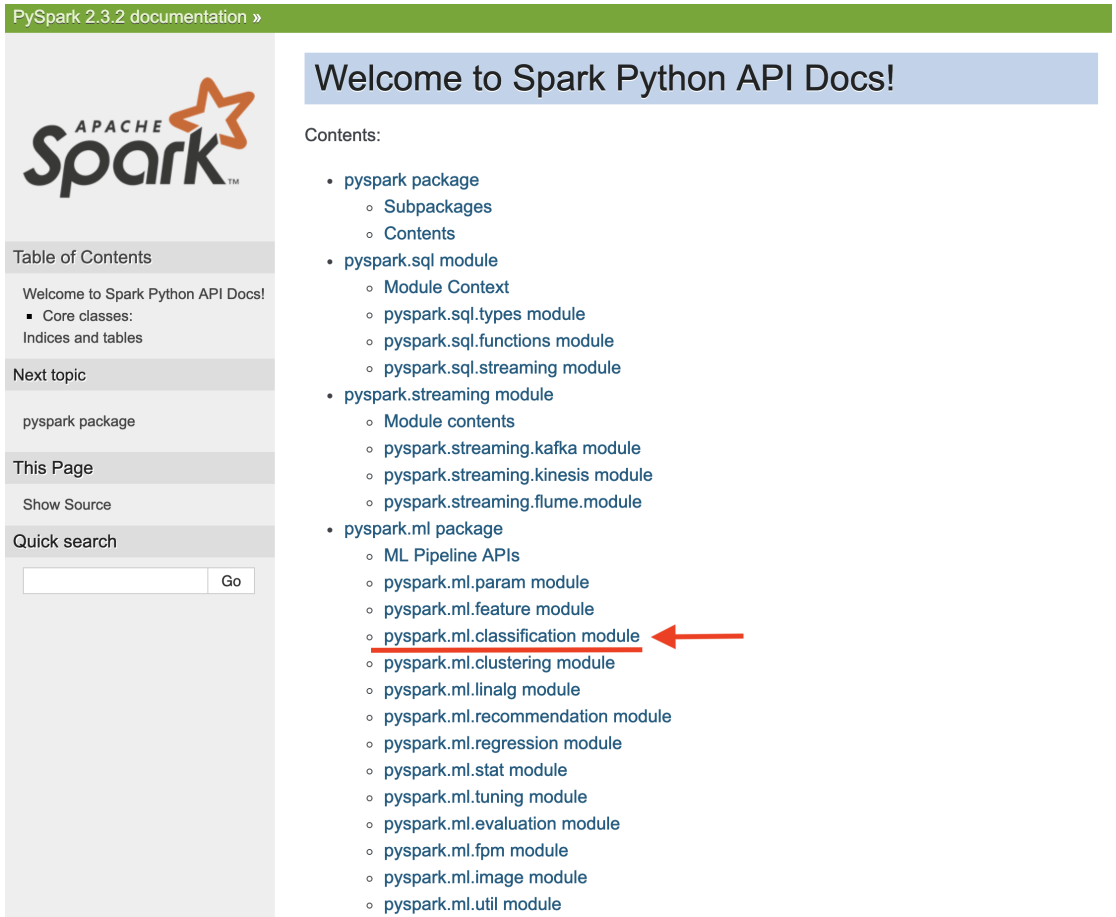
References used in this lecture

Book: *Fundamentals of Machine Learning for Predictive Analytics* by Kelleher, Mac Namee and D'Arcy, 2015.



References used in this lecture

Website: *Spark Python API Documentation*.



`https://spark.apache.org/docs/2.3.2/api/python/index.html`