COMP9033
DATA ANALYTICS

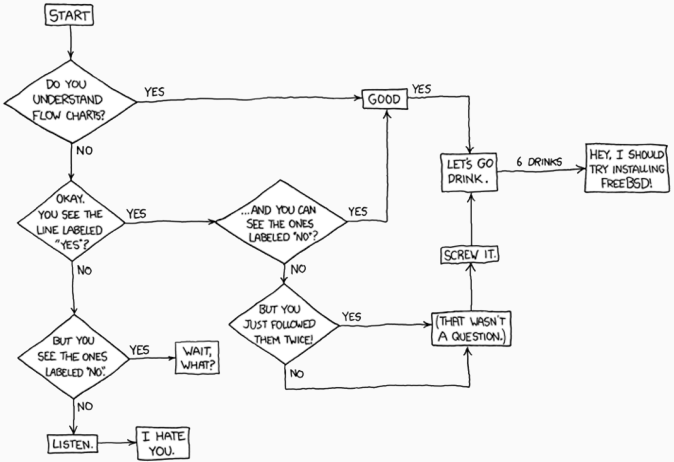
8/12

DECISION TREES

DR. DONAGH HORGAN

DEPARTMENT OF COMPUTER SCIENCE
CORK INSTITUTE OF TECHNOLOGY

2017.03.22



Credit: Randall Munroe / xkcd

Overview

1. k nearest neighbours:
   - The k nearest neighbours algorithm.
   - Weighted observations.
   - Choosing the number of neighbours.
   - Choosing a distance measure.
   - Choosing a weighting scheme.
   - Advantages and disadvantages.

2. Recommender systems:
   - The long tail phenomenon.
   - Content-based recommenders.
   - User-based collaborative filters.
   - Item-based collaborative filters.
   - Advantages and disadvantages.
   - The Netflix Prize.

1. Measuring classifier error:
   - Confusion matrices.
   - Accuracy.
   - Misclassification rate.
   - Precision.
   - Recall.
2. Decision tree classification:
   - Tree structures.
   - Classification trees.
   - Iris data set example.
   - The CART algorithm.
   - Impurity measures.
   - Stopping criteria.
3. Decision tree regression:
   - Regression trees.
   - Impurity measures.
4. Advanced techniques:
   - Decision tree pruning.
   - Random forests.

Measuring classifier error

- Classifiers produce categoric predictions, not numeric predictions.
- This means that we can't use quantitative measures (*e.g.* mean absolute error, root mean square error) to measure how accurate a classifier is.
- Instead, we must use alternative measurements of accuracy.

		predicted ⟶		
		Cat	Dog	Pig
true ↓	Cat	3	0	1
	Dog	0	6	1
	Pig	2	2	5

- One commonly used technique is the confusion matrix, shown opposite.
- In a confusion matrix, the rows represent the true classes of the data, while the columns represent the predicted classes.
- By examining a particular entry, we can check how many correct classifications were made, *e.g.*
  - Three cats were correctly identified.
  - One dog was mistaken for a pig.
  - Two pigs were mistaken for cats.

		predicted ⟶		
		Cat	Dog	Pig
true ↓	Cat	3	0	1
	Dog	0	6	1
	Pig	2	2	5

- We can condense the data in the confusion matrix to get a better picture about the accuracy of our classifier on a single category.
- For instance, if we focus on cats, then we can create the new confusion matrix on the bottom right.

	Cat	Dog	Pig
Cat	3	0	1
Dog	0	6	1
Pig	2	2	5

	Cat	Not a cat
Cat	3	1
Not a cat	2	14

- The top left entry represents the number of *true positives* (TP), *e.g.* the number of correctly identified cats.
- The top right entry represents the number of *false negatives* (FN), *e.g.* the number of cats identified as other animals.
- The bottom left entry represents the number of *false positives* (FP), *e.g.* the number of other animals identified as cats.
- The bottom right entry represents the number of *true negatives* (TN), *e.g.* the number of other animals identified as other animals.

	Cat	Not a cat
Cat	3	1
Not a cat	2	14

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- Ideally, we'd like the number of *false negatives* and the number of *false positives* to be zero, *i.e.* only the diagonal elements of the matrix would be non-zero.
- This is also true for uncondensed confusion matrices: for zero error, the only non-zero elements should lie on the diagonal running from the top left to the bottom right.

	Cat	Not a cat
Cat	3	1
Not a cat	2	14

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- The *accuracy* of a classifier is defined as the ratio of correct predictions to total predictions, *i.e.*

$$\begin{aligned}\text{Accuracy} &= \frac{\text{Correct predictions}}{\text{Total predictions}} \\ &= \frac{TP + TN}{TP + FN + FP + TN}.\end{aligned} \tag{8.1}$$

- By definition, accuracy lies in the range $[0, 1]$.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- Accuracy is a useful measure of the overall performance of a classifier:
  - As the proportion of our correct predictions grows larger, accuracy grows larger.
  - As the proportion of our correct predictions grows smaller, accuracy grows smaller.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- The *misclassification rate* (MR) of a classifier is defined as the ratio of incorrect predictions to total predictions, *i.e.*

$$\begin{aligned} \text{MR} &= \frac{\text{Incorrect predictions}}{\text{Total predictions}} \\ &= \frac{FN + FP}{TP + FN + FP + TN}. \end{aligned} \tag{8.2}$$

- By definition, the misclassification rate lies in the range $[0, 1]$.
- Using Equation 8.1, it can be shown that $\text{MR} = 1 - \text{Accuracy}$.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- The *precision* of a classifier is the ratio of true positives to the total positives, *i.e.*

$$\begin{aligned}\text{Precision} &= \frac{\text{True positive predictions}}{\text{Total positive predictions}} \\ &= \frac{TP}{TP + FP}. \end{aligned} \tag{8.3}$$

- By definition, precision lies in the range $[0, 1]$.
- In our earlier example, precision would measure the proportion of animals labelled as cats that were actually cats.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- Precision is useful because it gives us finer detail than accuracy about how our classifier behaves on the *positive* class:
  - As the number of true positive predictions increases, precision grows larger.
  - As the number of true positive predictions decreases, precision grows smaller.
  - As the number of false positive predictions increases, precision grows smaller.
  - As the number of false positive predictions decreases, precision grows larger.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- The *recall* of a classifier is the ratio of the number of true positive classifications to the total number of elements in the target class:

$$\begin{aligned}\text{Recall} &= \frac{\text{True positive predictions}}{\text{Target class}} \\ &= \frac{TP}{TP + FN}.\end{aligned} \tag{8.4}$$

- By definition, recall lies in the range $[0, 1]$.
- In our earlier example, recall would measure the proportion of true cats that were labelled as cats.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

- Recall is useful because it gives us finer detail than accuracy about how our classifier behaves on the *target* class:
  - As the number of true positive predictions increases, recall grows larger.
  - As the number of true positive predictions decreases, recall grows smaller.
  - As the number of false negative predictions increases, recall grows smaller.
  - As the number of false negative predictions decreases, recall grows larger.
- Recall is sometimes referred to as *sensitivity*.

	Cat	Not a cat
Cat	TP	FN
Not a cat	FP	TN

Q. Compute accuracy, precision and recall using confusion matrix opposite.

A. We can compute the accuracy using Equation 8.1 as

$$\begin{aligned}\text{Accuracy} &= \frac{TP + TN}{TP + FN + FP + TN} \\ &= \frac{3 + 14}{3 + 1 + 2 + 14} \\ &= 0.85.\end{aligned}$$

	Cat	Not a cat
Cat	3	1
Not a cat	2	14

- We can compute the precision using Equation 8.3 as

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3+2} = 0.6.$$

- Finally, we can compute the recall using Equation 8.4 as

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{3}{3+1} = 0.75.$$

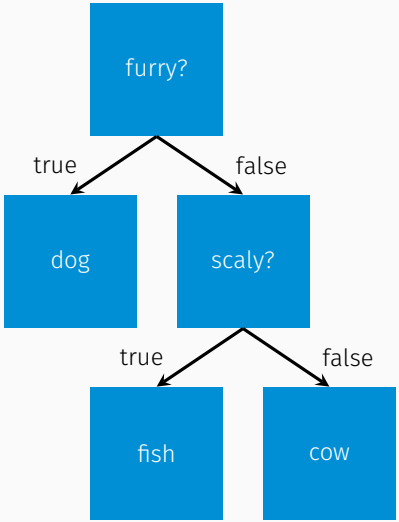
	Cat	Not a cat
Cat	3	1
Not a cat	2	14

- The precision and recall are lower than the accuracy!
  - Our accuracy is high because we're relatively good at correctly labelling cats and non-cats.
  - Our precision and recall are lower because, relative to the total number of cats, the number of mistakes we make is higher.

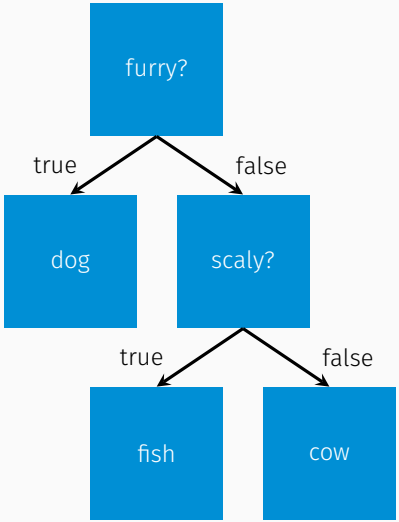
	Cat	Not a cat
Cat	3	1
Not a cat	2	14

Decision tree classification

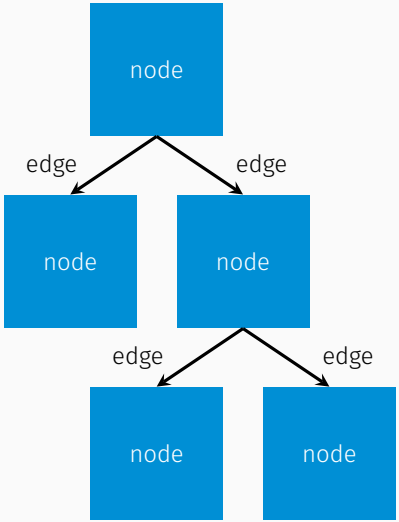
- *Decision trees* are a class of supervised machine learning algorithms, *i.e.* they learn from data using statistics and heuristics.
- They are based on nested if-else clauses and so can be represented using a hierarchical flow chart, which makes them easy to interpret.
- The tree to the right illustrates a (very) simple animal classification model.



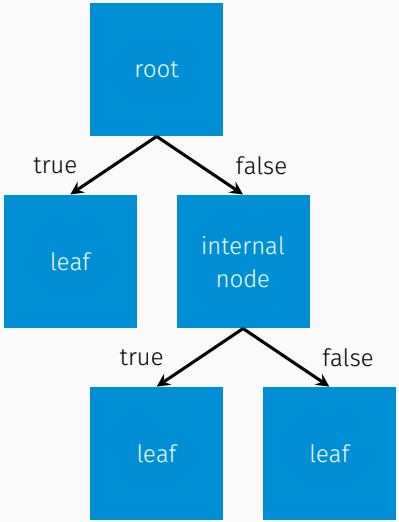
- A number of algorithms exist for creating decision trees, *e.g.*
  - ID3: classification only.
  - C4.5: classification only.
  - CART: classification and regression.
- For a more complete list, see bit.ly/2n417Sj.
- For the remainder of this lecture, we'll focus on CART, which stands for *Classification And Regression Trees*.
- The decision tree subpackage in scikit-learn uses a variant of CART to build its trees.



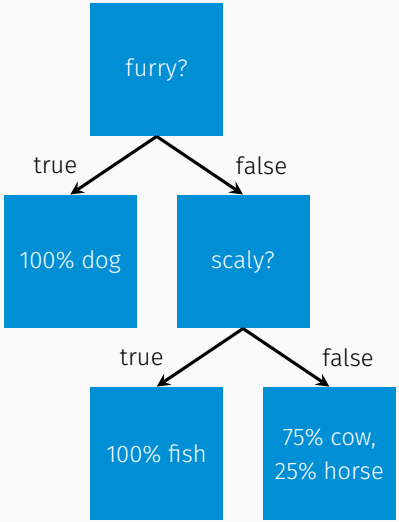
- A decision tree is essentially a graph structure, consisting of *nodes* and *edges*:
  - Nodes represent states.
  - Edges represent connections between states.
- In binary decision trees, nodes either have *two* edges (representing a true/false condition) or *none*.



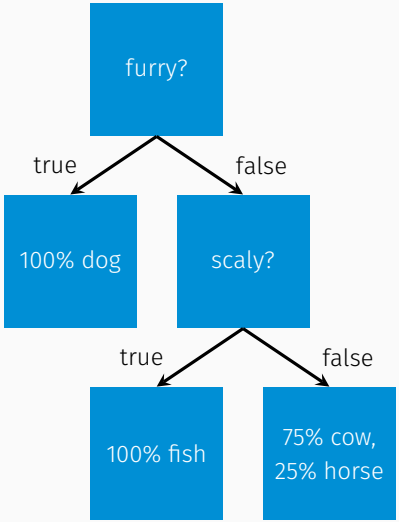
- Depending on their position within the tree, nodes have different meanings:
  - The uppermost node in the tree is known as the *root node* and is the starting state of any prediction attempt.
  - The end nodes (*i.e.* those with no edges) are known as *leaf nodes* and represent a final outcome (*e.g.* spam or ham) or set of outcomes.
  - All other nodes are called *internal nodes* and represent some intermediate state between the root and one of the leaves.



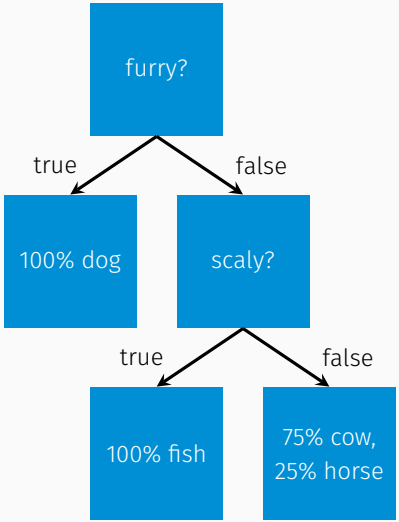
- Once a tree has been created, we can use it to predict the category of an unlabelled sample as follows:
  1. Start at the root node.
  2. Check which edge matches our criteria (one always will) and then follow it to the next node.
  3. Repeat Step 2 until we reach a leaf node.



- In decision tree classification, leaf nodes represent a category or set of categories, depending on how the decisions in the tree have split the training data:
  - If a leaf node represents a single outcome, then that outcome is predicted.
  - If a leaf node represents more than one outcome, then we select the most likely outcome (*i.e.* the weighted mode) as the prediction.



- For instance, in the tree to the right:
  - If an animal is furry, then we would predict that it is a dog.
  - If an animal is not furry, but is scaly, then we would predict that it is a fish.
  - If an animal is not furry and is not scaly, then we would predict that it is a cow.



Iris setosa

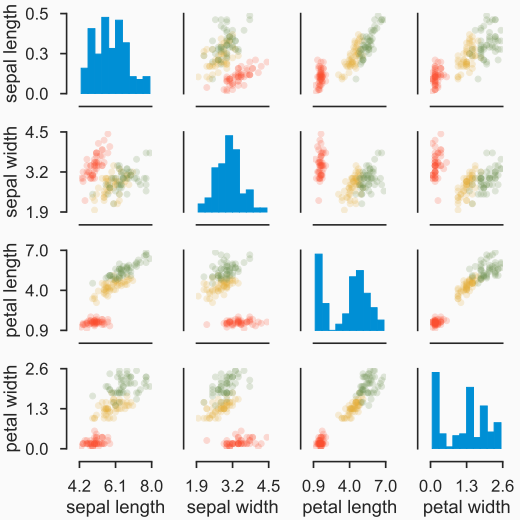
Iris virginica

Iris versicolor

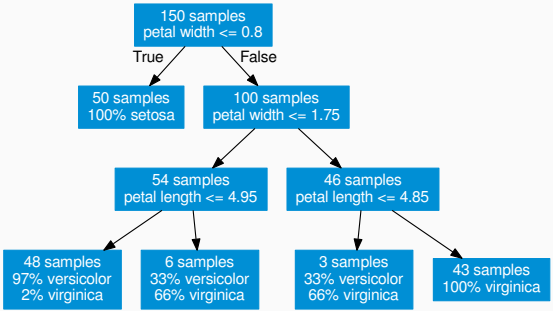
- Let's use decision trees to analyse the Iris data set:
  - The data set consists of 150 evenly distributed categoric samples, *i.e.* each category has the same number of samples.
  - There are three categories: *Iris setosa*, *Iris virginica*, *Iris versicolor*.
  - There are four explanatory variables: sepal length, sepal width, petal length and petal width.

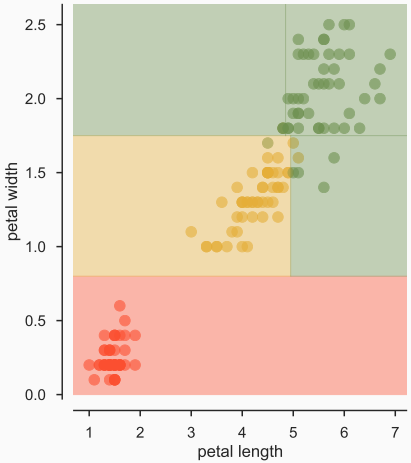
Credit: Wikipedia

- *Iris setosa* (red) is easily distinguished as its characteristics are very different to the other two species.
- *Iris versicolor* (yellow) and *Iris virginica* (green) are more difficult to separate, as their properties are more similar.



2.10 / EXAMPLE: IRIS DATA SET





- The CART algorithm works as follows:
  1. Select an explanatory variable to split on.
  2. Choose one of the explanatory variable values to be the threshold value.
  3. Using an *impurity measure*, calculate the reduction in impurity that results from the splitting the data according to the chosen threshold.
  4. Repeat Steps 2-3 for various threshold values and select the threshold that results in the best impurity reduction.
  5. Repeat Steps 1-4 for each explanatory variable and select the variable that gives the best reduction in impurity.
  6. Split the data in two, according to the selected variable threshold.
  7. Repeat Steps 5 and 6 until some *stopping criterion* is met.
- CART can be run as a brute force algorithm (*i.e.* all possible threshold values are evaluated for each explanatory variable) or using heuristics to narrow the number of evaluations.

- An *impurity measure* is a mathematical function that quantifies how good a candidate split is.
- If we split a parent node (P) into a left child (L) and a right child (R), then the reduction in impurity is given by

$$\text{Impurity reduction} = I_P - \left(\frac{N_L}{N_P} I_L + \frac{N_R}{N_P} I_R \right), \tag{8.5}$$

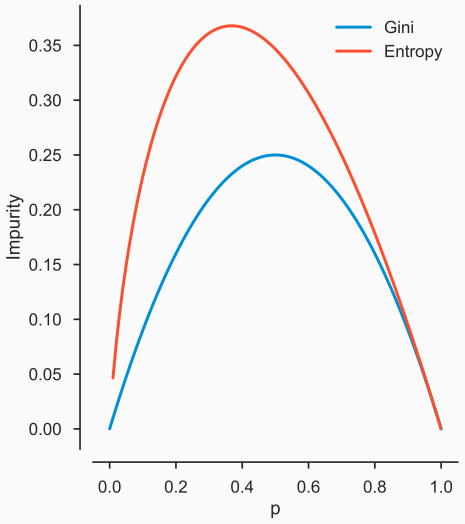
  where I_X represents the impurity of node X and N_X represents the number of samples at node X.
- There are many ways to measure the impurity at a node, but two common measures are Gini impurity and entropy.

- For a classification problem with k classes, the Gini impurity at node X is defined as

$$I_X = \sum_{i=1}^{k} p_i(1 - p_i), \qquad (8.6)$$

  where p_i is the proportion of samples at node X belonging to class i.

- The chart opposite shows the Gini impurity for a problem with $k = 2$ (*i.e.* $p_1 = 1 - p_2$).

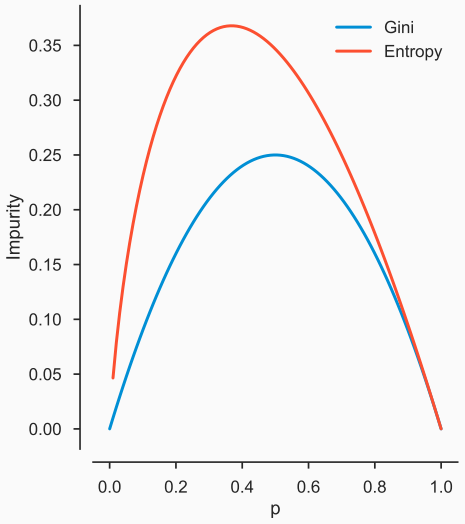


- For a classification problem with k classes, the entropy impurity at node X is defined as

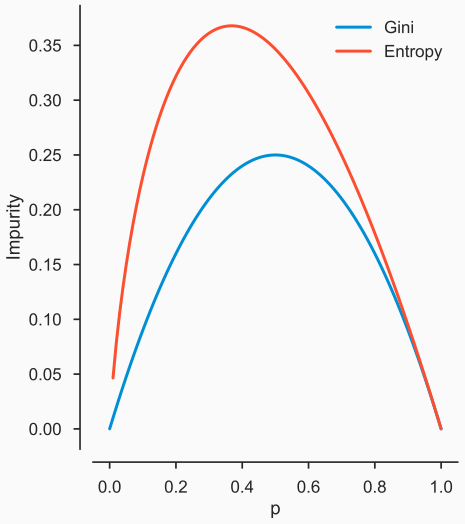
$$I_X = -\sum_{i=1}^{k} p_i \log(p_i), \qquad (8.7)$$

  where p_i is the proportion of samples at node X belonging to class i.

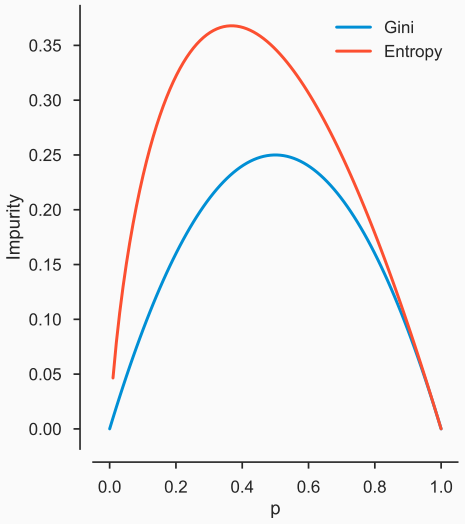
- The chart opposite shows the entropy impurity for a problem with $k = 2$ (*i.e.* $p_1 = 1 - p_2$).



- Entropy tends to select splits on attributes with large numbers of values.
- This isn't ideal in some cases, can lead to overfitting.
- Gini impurity tends to select splits based on the largest class.
- Again, this isn't ideal in some cases.
- Can use model selection to determine the best measure in a given situation.



- Generally, impurity measures are based on the proportion of samples at a node.
- Consequently, splits are typically unaffected by the presence of outlying data.

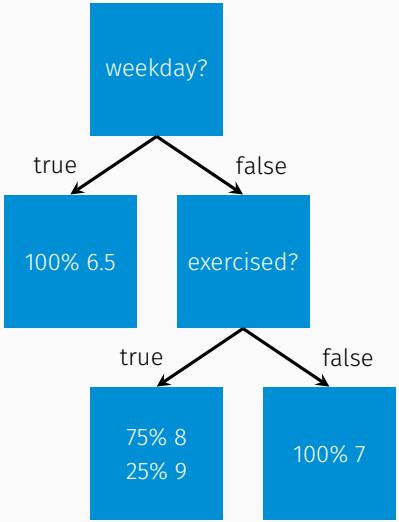


- A *stopping criterion* is a condition that, when met, halts tree building.
- If we don't specify a stopping criterion, then the input data will be split until we reach a state where each leaf node represents just one outcome.
- This can lead to large and complex trees, *i.e.* overfitted models.
- On the other hand, if specify a harsh stopping criterion, we may stop splitting too soon and produce an underfitted model.
- Again, we can use model selection to choose the best model.
- Stopping criteria can be used individually or in combination.

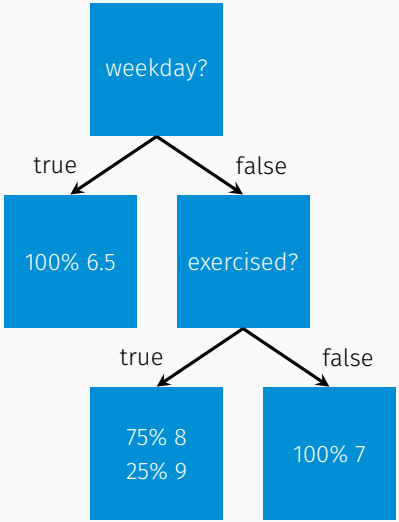
- Minimum samples for a split:
  - If the number of samples available at a node does not reach some threshold value, then do not split it any further.
- Minimum samples to be a leaf:
  - If the number of samples available at a node is larger than some threshold value, then split it; otherwise, make the node a leaf.
  - Guarantees a minimum number of samples in each leaf.
- Impurity reduction threshold:
  - Only split when the impurity reduction is greater than some threshold value.
  - Can lead us to halt at a weak split, when the following split would have added significant value.
- Maximum depth:
  - Only split while the number of levels in the tree is less than some threshold value.
  - Can lead us to miss valuable splits beyond the maximum depth.

Decision tree regression

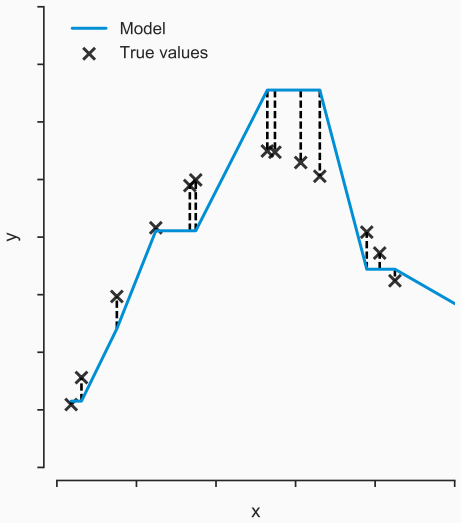
- Decision trees can also be used for regression:
  - If a leaf node represents a single quantity value, then that value is predicted.
  - More generally, leaf nodes represent many different quantity values, in which case a weighted average value is predicted.
- In the case where a weighted average is used, the outcomes are weighted according to the proportions of samples at the leaf node.



- For instance, the tree to the right illustrates a model that predicts the number of hours of sleep a person gets every night:
  - If it's a weekday, then we would predict 6.5 hours of sleep.
  - If it's not a weekday, and the person didn't exercise, then we would predict 7 hours of sleep.
  - If it's not a weekday, and the person did exercise, then we would predict $0.75 \times 8 + 0.25 \times 9 = 8.25$ hours of sleep.



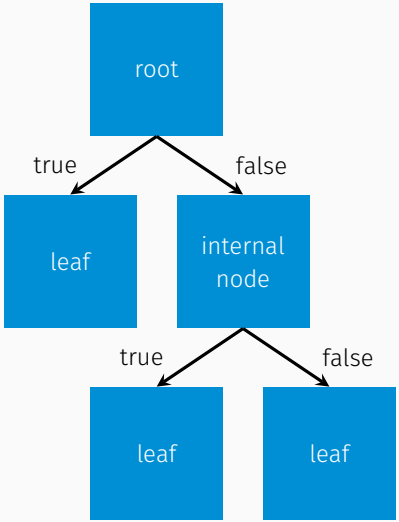
- CART builds regression trees in the same way as classification trees, but uses quantitative impurity measures as the target variable is not categoric.
- Typically, the mean absolute error (MAE) and the root mean square error (RMSE) are used to measure the impurity of nodes and Equation 8.5 is used to calculate the reduction.
- Splits can be influenced by outliers, depending on the measure used.

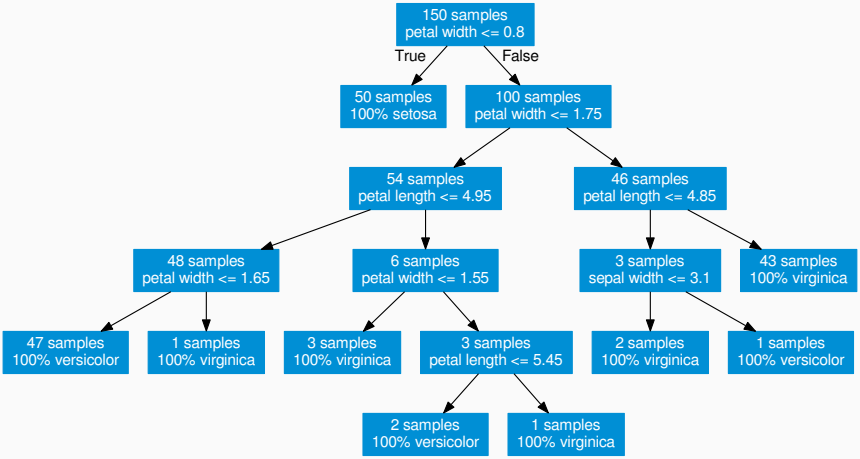


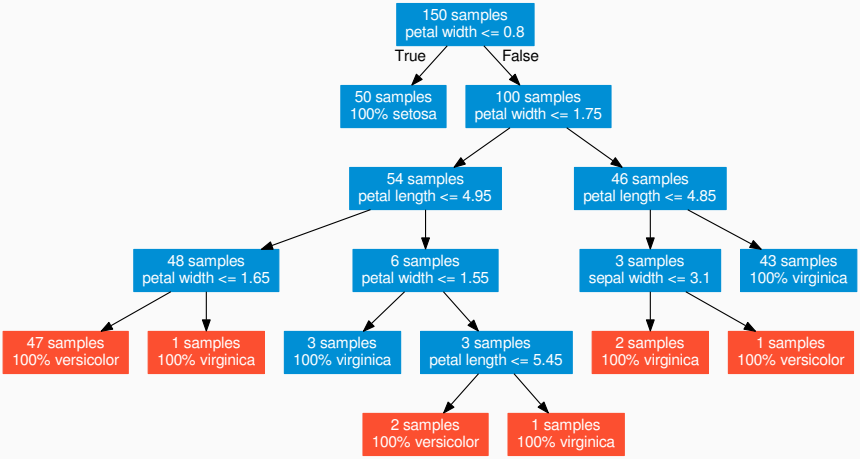
Advanced techniques

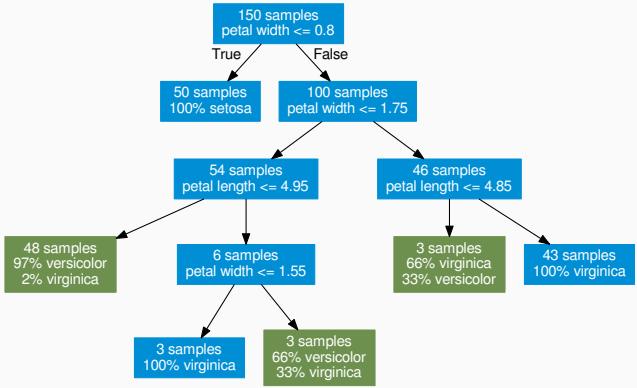
- *Decision tree pruning* is an alternative to using stopping criteria.
- Weak splits are eliminated after the *full* tree is built:
  1. Starting at the tree leaves, each pair of splits in a branch is analysed.
  2. If the splits do not reduce the impurity of their parent by more than some pre-defined threshold, then they are eliminated and their parent is made a leaf node.
  3. The process continues until all leaf nodes are those which reduce the impurity of their parent by more than the threshold amount.
- The idea is similar to using an impurity reduction threshold as a stopping criterion, but ensures that we never miss a valuable split because its parent was weak (*i.e.* it is a bottom up rather than a top down approach).
- Pruning is currently *unsupported* in scikit-learn!

- Earlier, we built a decision tree for the Iris flower data set that was limited to a maximum depth of three levels.
- Instead of stopping early like this, let's build the full tree and prune it.
- Normally, we would prune nodes that don't meet a certain impurity reduction threshold but for simplicity let's prune nodes that have fewer than three samples.

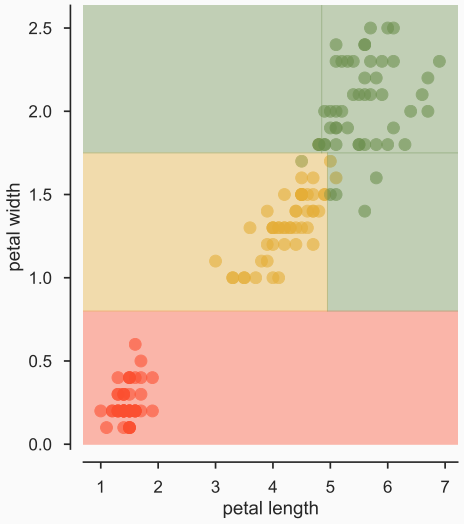



150 samples
petal width <= 0.8
True
False
50 samples
100% setosa
100 samples
petal width <= 1.75
54 samples
petal length <= 4.95
46 samples
petal length <= 4.85
48 samples
petal width <= 1.65
6 samples
petal width <= 1.55
3 samples
sepal width <= 3.1
43 samples
100% virginica
47 samples
100% versicolor
1 samples
100% virginica
3 samples
100% virginica
3 samples
petal length <= 5.45
2 samples
100% virginica
1 samples
100% versicolor
2 samples
100% versicolor
1 samples
100% virginica


150 samples
petal width <= 0.8
True
False
50 samples
100% setosa
100 samples
petal width <= 1.75
54 samples
petal length <= 4.95
46 samples
petal length <= 4.85
48 samples
petal width <= 1.65
6 samples
petal width <= 1.55
3 samples
sepal width <= 3.1
43 samples
100% virginica
47 samples
100% versicolor
1 samples
100% virginica
3 samples
100% virginica
3 samples
petal length <= 5.45
2 samples
100% virginica
1 samples
100% versicolor
2 samples
100% versicolor
1 samples
100% virginica


150 samples
petal width <= 0.8
True
False
50 samples
100% setosa
100 samples
petal width <= 1.75
54 samples
petal length <= 4.95
46 samples
petal length <= 4.85
48 samples
97% versicolor
2% virginica
6 samples
petal width <= 1.55
3 samples
66% virginica
33% versicolor
43 samples
100% virginica
3 samples
100% virginica
3 samples
66% versicolor
33% virginica

- Decision trees rely on threshold-based comparisons:
  - This makes them easy to understand and visualise.
  - However, it can make their predictions unstable.
- For instance, in the chart to the right, if we encounter a set of flowers whose petal widths are all close to 0.8, we will classify some as setosa and others as virginica, even if they are all the same species.



- One solution to this problem is the use of *random forests*:
  - The key idea is to train multiple decision tree models on random subsets of the input data.
  - The probability of each tree outputting a certain class can be measured.
  - When new data is evaluated, each tree is used to make a decision about the data and these decisions are combined using a weighted mode/average with the class probabilities of the trees as weights.
  - Weighting the outcomes of each tree can avoid overfitting and generally gives better results.
- Random forests are an example of an *ensemble method*, *i.e.* the combination of one or more weaker models to create a stronger overall model.

Summary

- Decision trees:
  - Used for classification and regression.
  - Flow chart representation, easy to understand.
  - Hyperparameters: impurity measure and stopping criteria.
  - Pruning: alternative to using stopping criteria, ensures that no valuable splits are missed.
  - Random forests: combine trees to reduce sensitivity around boundaries.
- Lab work:
  - Build a decision tree/random forest classification model for SMS spam data.
  - Build a decision tree/random forest regression model for server load data.
- Next week: unsupervised learning and clustering algorithms!

1. Hastie et al. *The Elements of Statistical Learning: Data mining, Inference and Prediction.* 2nd edition, February 2009. (stanford.io/1dLkiAv)
2. Rokach and Maimon. *The Data Mining and Knowledge Discovery Handbook.* Springer, 2010. (bit.ly/2n417Sj)
3. Stamm-Wilbrandt, Hermann. *GraphvizFiddle.* (bit.ly/2nKd8JL)