GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

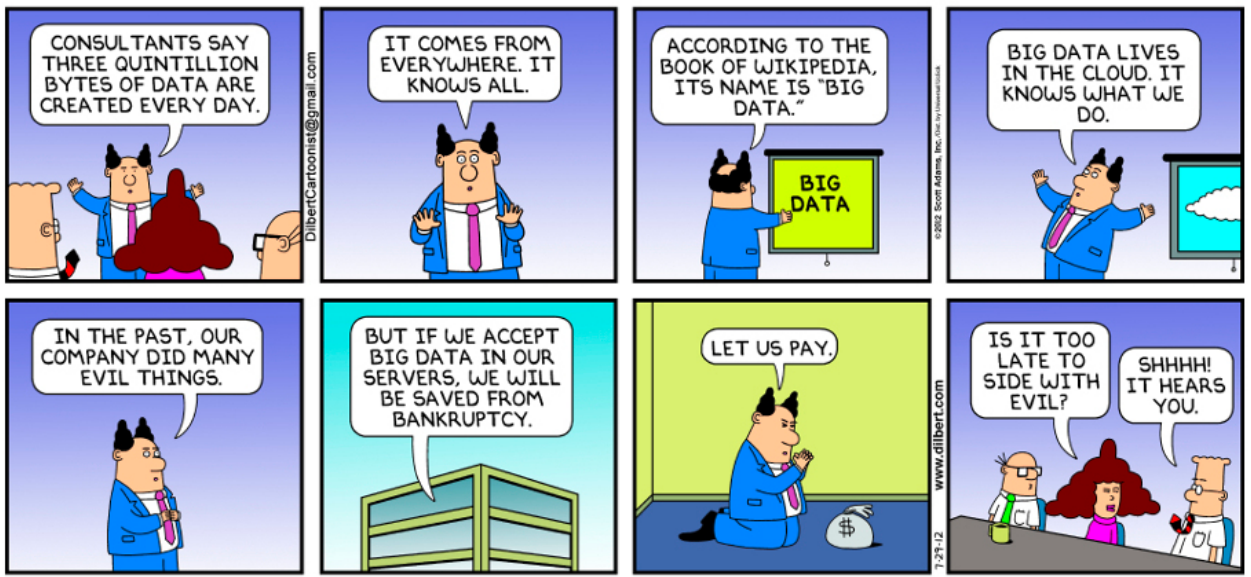
DECISION TREES

DATA SCIENCE IN THE NEWS

DATA SCIENCE IN THE NEWS

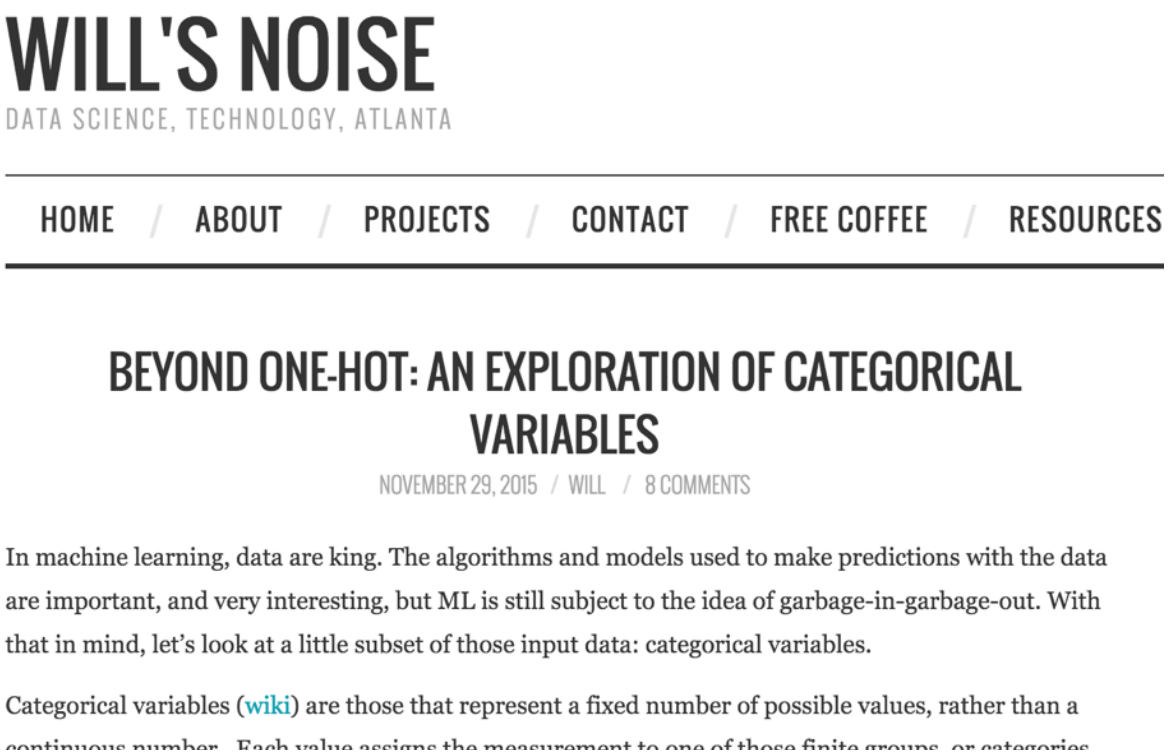
Problem definition is hard

There are many reasons why problem definition can be hard. It is sometimes due to stakeholders who don't know what they want, and expect data scientists to solve all their data problems (either real or imagined). This type of situation is summarised by the following Dilbert strip. It is best handled by cleverly managing stakeholder expectations, while stirring them towards better-defined problems.



http://yanirseroussi.com/2015/11/23/the-hardest-parts-of-data-science/

DATA SCIENCE IN THE NEWS



WILL'S NOISE

DATA SCIENCE, TECHNOLOGY, ATLANTA

HOME / ABOUT / PROJECTS / CONTACT / FREE COFFEE / RESOURCES

BEYOND ONE-HOT: AN EXPLORATION OF CATEGORICAL VARIABLES

NOVEMBER 29, 2015 / WILL / 8 COMMENTS

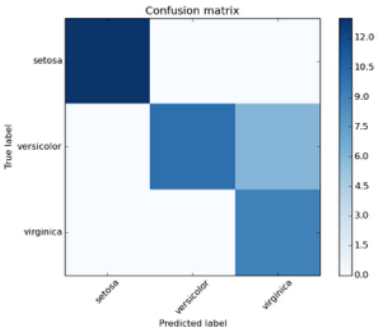
In machine learning, data are king. The algorithms and models used to make predictions with the data are important, and very interesting, but ML is still subject to the idea of garbage-in-garbage-out. With that in mind, let's look at a little subset of those input data: categorical variables.

Categorical variables (wiki) are those that represent a fixed number of possible values, rather than a continuous number. Each value assigns the measurement to one of those finite groups, or categories

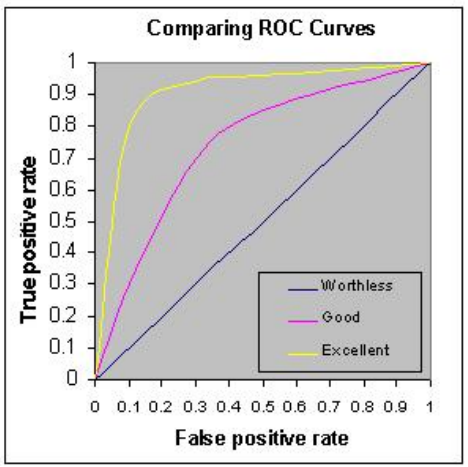
http://willmcginnis.com/2015/11/29/beyond-one-hot-an-exploration-of-categorical-variables/

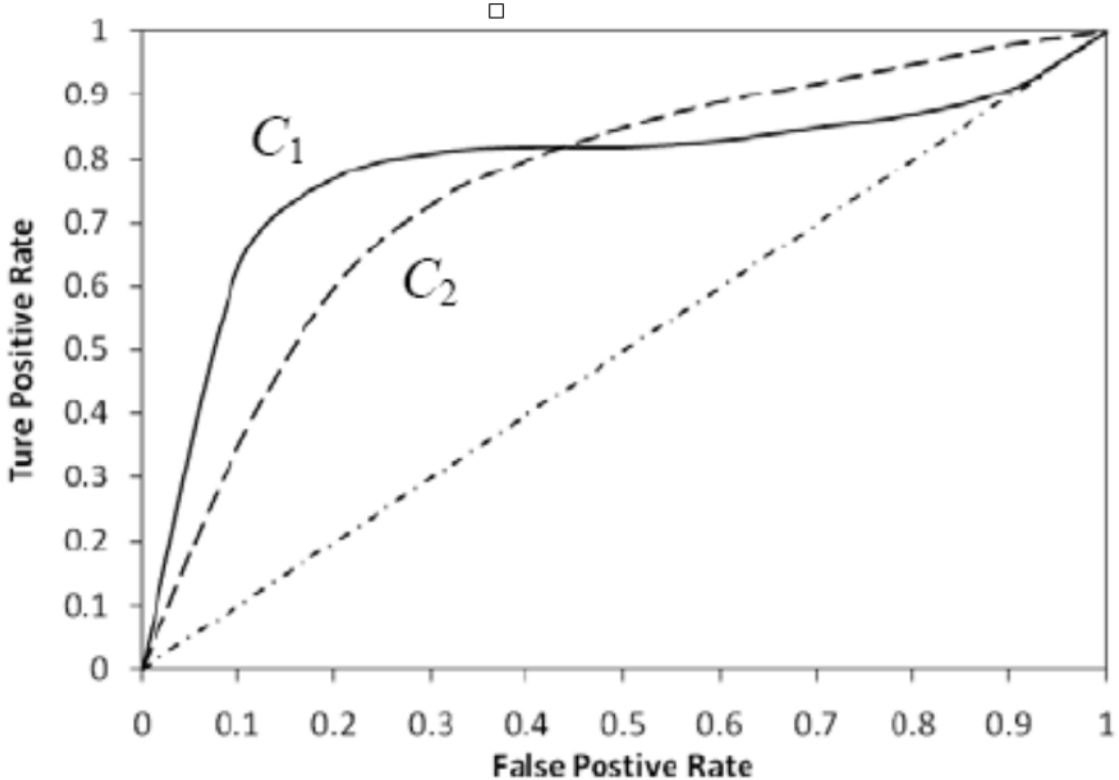
LAST TIME:

I. ERROR RATES & CONFUSION MATRIX
II. ROC CURVES
III. AUC AS EVALUATION METRIC



	Condition Positive	Condition Negative
Test Positive	TRUE POSITIVE	FALSE POSITIVE (Type I error)
Test Negative	FALSE NEGATIVE (Type II error)	TRUE NEGATIVE





Pop Quiz!

Which of the two classifiers shown (on the same data), C_1 or C_2, is better and why?

QUESTIONS?

WHAT WAS THE MOST INTERESTING THING YOU LEARNED?

WHAT WAS THE HARDEST TO GRASP?

AGENDA

I. DECISION TREES INTRO
II. BUILDING DECISION TREES
III. OPTIMIZATION FUNCTIONS
IV. PREVENTING OVERFITTING
V. LAB: DECISION TREES IN PYTHON

DECISION TREES INTRO

Decision trees

non-parametric hierarchical classification technique.

- **non-parametric**: no parameters, no distribution assumptions

- **hierarchical**: consists of a sequence of questions which yield a class label when applied to any record

Decision trees

non-parametric hierarchical classification technique.

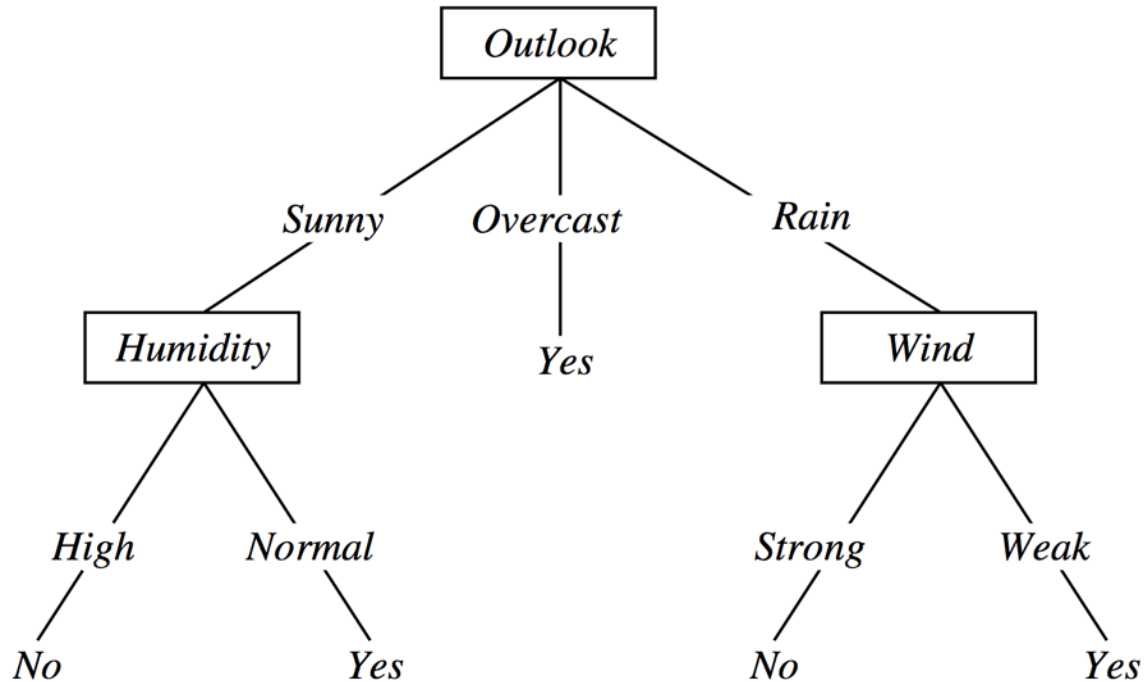
- **non-parametric**: no parameters, no distribution assumptions

  from Wikipedia:
  "Non-parametric models differ from parametric models in that the model structure is not specified a priori but is instead determined from data. The term non-parametric is not meant to imply that such models completely lack parameters but that the number and nature of the parameters are flexible and not fixed in advance."

represented as a **multiway tree**, which is a type of (directed acyclic) **graph**

Nodes => questions

Edges => answers



Classify an instance: <outlook=Sunny, temp = Hot, humidity=High, wind = Strong>

Top node => root node
0 incoming edges, and 2+ outgoing edges

Internal node => test condition
1 incoming edge, and 2+ outgoing edges

Leaf node => class label
has 1 incoming edge and, 0 outgoing edges

Table 4.1. The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

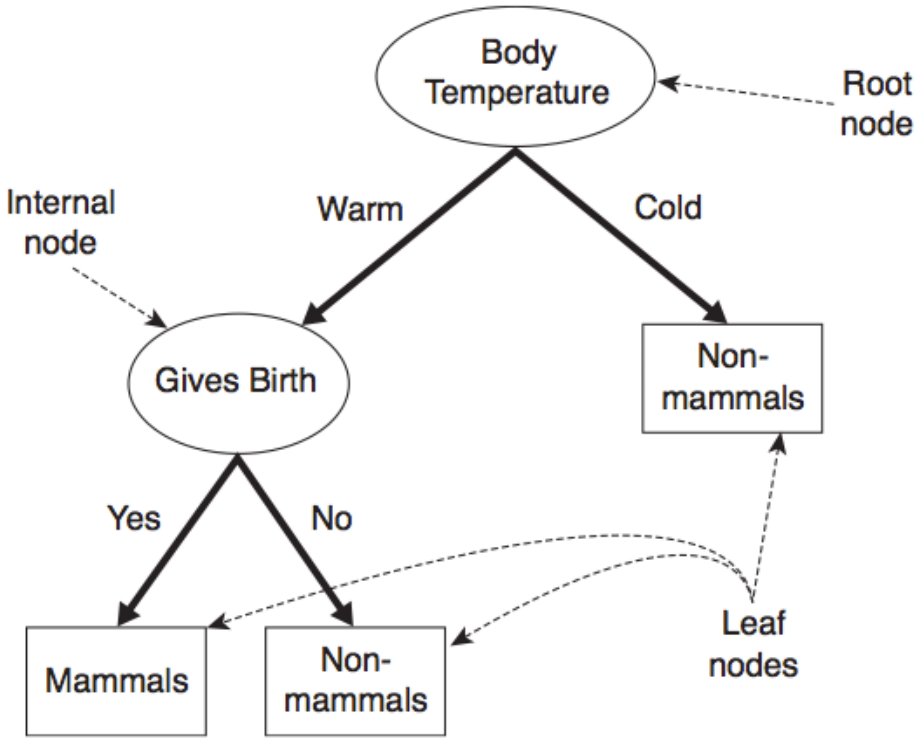


Figure 4.4. A decision tree for the mammal classification problem.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

Quick exercise:

what's a decision tree? (3 words)

BUILDING DECISION TREES

How would you build a decision tree?

Hunt's algorithm

greedy recursive algorithm that leads to a **local optimum**

greedy

algorithm makes locally optimal decision at each step

recursive

splits task into subtasks, solves each the same way

local optimum

solution for a given neighborhood of points

Recursively partition records into smaller & smaller subsets.

The partitioning decision is made at each node according to a **metric** called **purity**

100% pure when all of its records belong to a single class

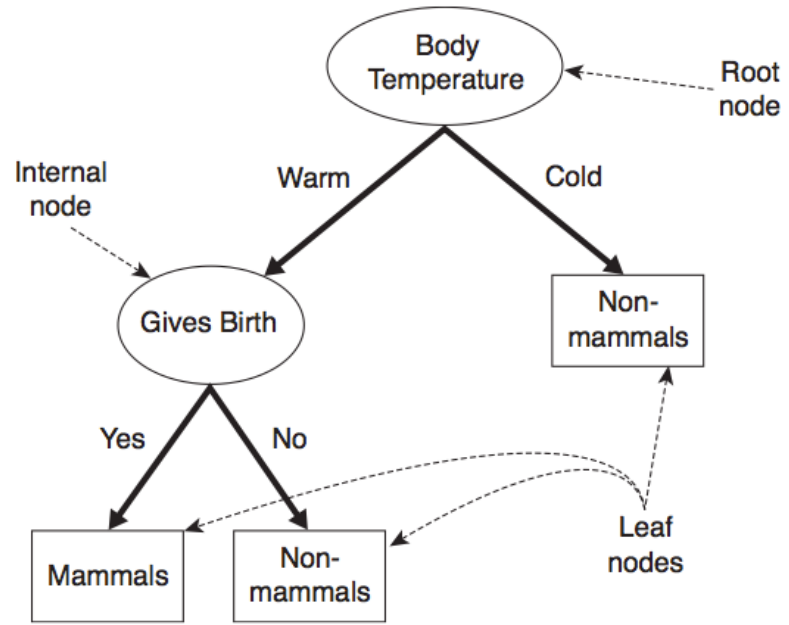


Figure 4.4. A decision tree for the mammal classification problem.

Example: binary classification with classes A, B. Given a set of records D_t at node t:

Example: binary classification with classes A, B. Given a set of records D_t at node t:

I) If all records in D_t belong to class A, then t is a **leaf node** corresponding to class *(Base case)*

Example: binary classification with classes A, B. Given a set of records D_t at node t:

1) If all records in D_t belong to class A, then t is a **leaf node** corresponding to class *(Base case)*

2) If D_t contains records from both A and B do the following:
   i. create test condition to partition the records further
   ii. t is an internal node, with outgoing edges to child nodes
   iii. partition records in D_t to children according to test

Example: binary classification with classes A, B. Given a set of records D_t at node t:

1) If all records in D_t belong to class A, then t is a **leaf node** corresponding to class *(Base case)*

2) If D_t contains records from both A and B do the following:
   i. create test condition to partition the records further
   ii. t is an internal node, with outgoing edges to child nodes
   iii. partition records in D_t to children according to test

3) These steps are then recursively applied to each child node.

Binary splits

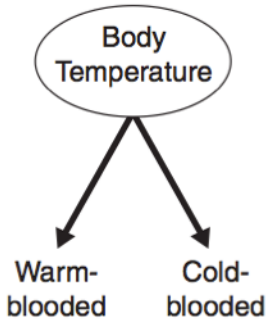
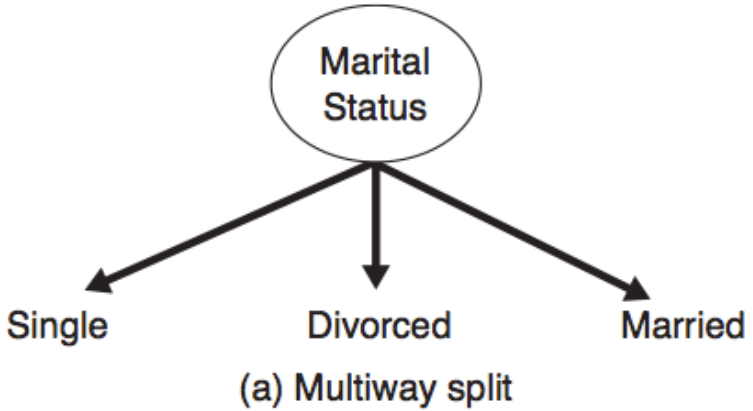


Figure 4.8. Test condition for binary attributes.

Multiway splits



(a) Multiway split

Continous features

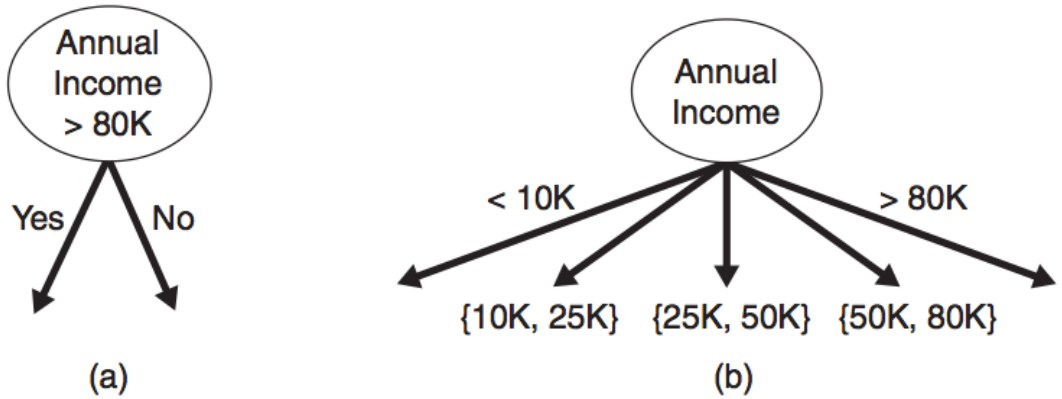


Figure 4.11. Test condition for continuous attributes.

Quick pair exercise: find rules to split the data below, then share with the class

Completed	*Time*	*Result*
3	*30*	*Pass*
2	*25*	*Pass*
4	*49*	*Pass*
3	*47*	*Fail*
2	*50*	*Fail*
1	*32*	*Fail*
3	*26*	*Pass*

OPTIMIZATION FUNCTIONS – FINDING THE BEST SPLIT

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the ***highest possible purity.***

We need an objective function to optimize!

We want our objective function to measure the gain in purity from a particular split.

Table 4.1. The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

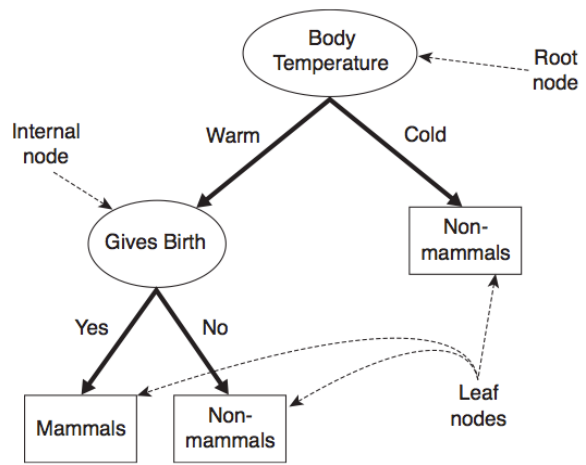


Figure 4.4. A decision tree for the mammal classification problem.

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

For example, let $p(i|t)$ *be the probability of class* i *at node* t *(eg, the fraction of records labeled* i *at node* t*).*

Then for a binary (0/1) classification problem,

Then for a binary (0/1) classification problem,

The minimum purity partition is given by the distribution:

$$p(0|t) = p(1|t) = 0.5$$

Then for a binary (0/1) classification problem,

The minimum purity partition is given by the distribution:

$$p(0|t) = p(1|t) = 0.5$$

The maximum purity partition is given (eg) by the distribution:

$$p(0|t) = 1 - p(1|t) = 1$$

how to measure the value of information?

*Some measures of **impurity** include:*

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)],$$

Example impurity calculations for three different nodes (each with two classes)

Node N_1	Count
Class=0	0
Class=1	6

$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$
$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
$\text{Error} = 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$
$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$
$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

$$\text{Entropy}(t) \quad = \quad -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

Use entropy as a measure of impurity or disorder of the data set

$$\text{Entropy}(t) \quad = \quad -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

1. The data set D has 50% positive examples (Pr(*positive*) = 0.5) and 50% negative examples (Pr(*negative*) = 0.5).

2. The data set D has 20% positive examples (Pr(*positive*) = 0.2) and 80% negative examples (Pr(*negative*) = 0.8).

3. The data set D has 100% positive examples (Pr(*positive*) = 1) and no negative examples, (Pr(*negative*) = 0).

source: http://www.cs.uic.edu/~liub/teach/cs583-spring-12/CS583-supervised-learning.ppt

1. The data set D has 50% positive examples (Pr(*positive*) = 0.5) and 50% negative examples (Pr(*negative*) = 0.5).

$$entropy(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples (Pr(*positive*) = 0.2) and 80% negative examples (Pr(*negative*) = 0.8).

$$entropy(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples (Pr(*positive*) = 1) and no negative examples, (Pr(*negative*) = 0).

$$entropy(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

As the data become purer and purer, the entropy value becomes smaller and smaller.

Note that each measure achieves its max at 0.5, min at 0 & 1.

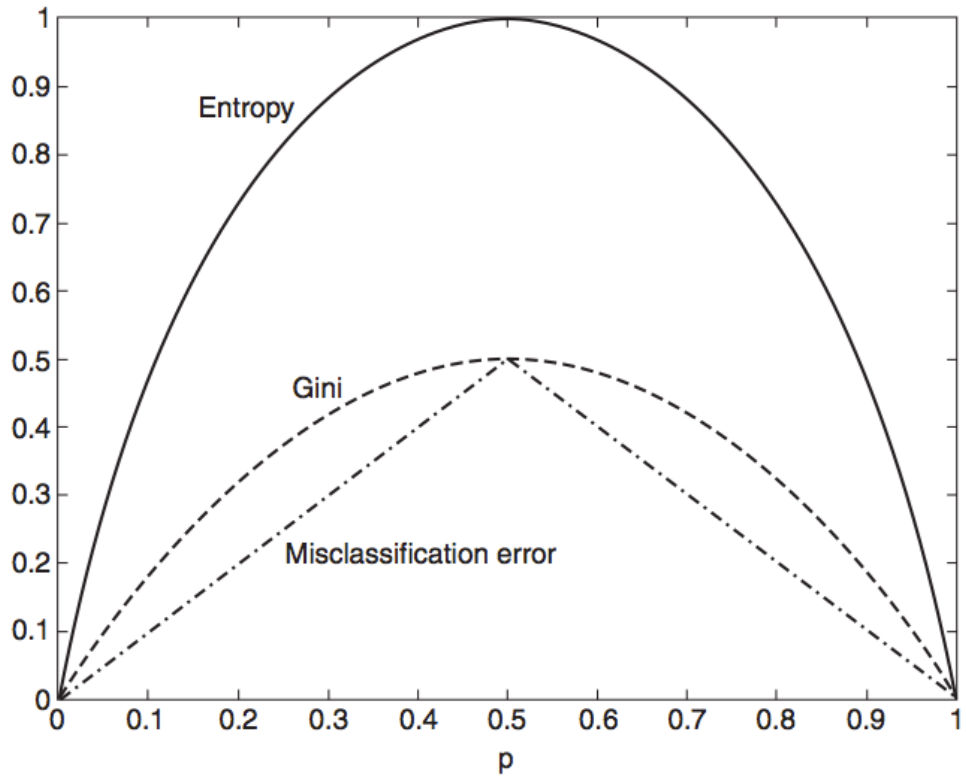


Figure 4.13. Comparison among the impurity measures for binary classification problems.

Note that each measure achieves its max at 0.5, min at 0 & 1.

NOTE

Despite consistency, different measures may create different splits.

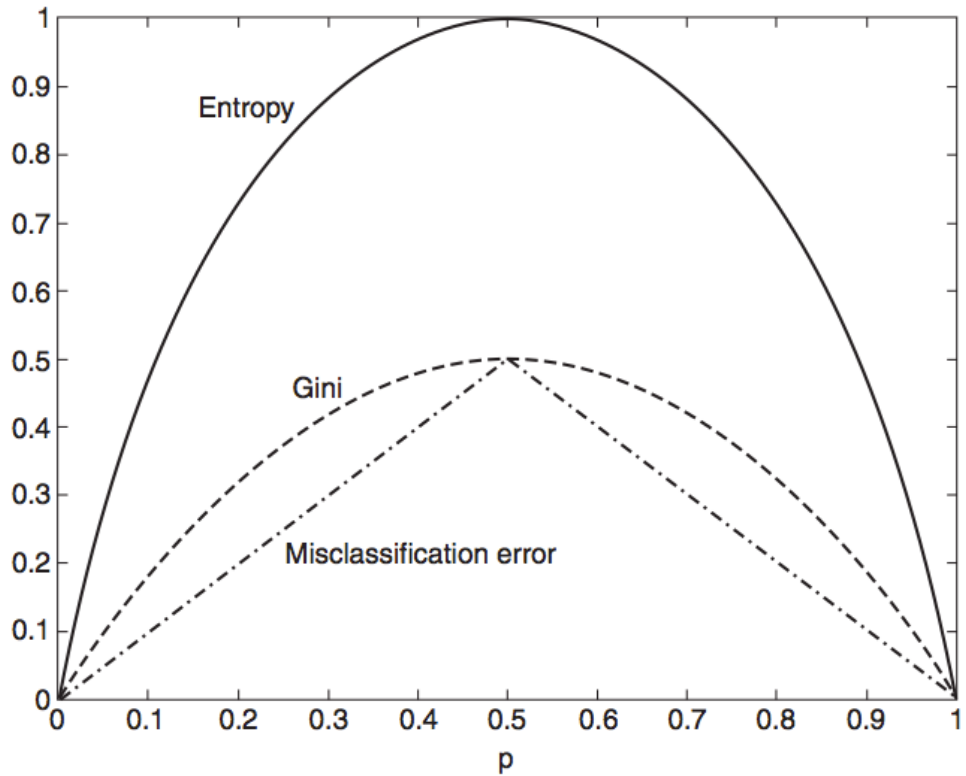
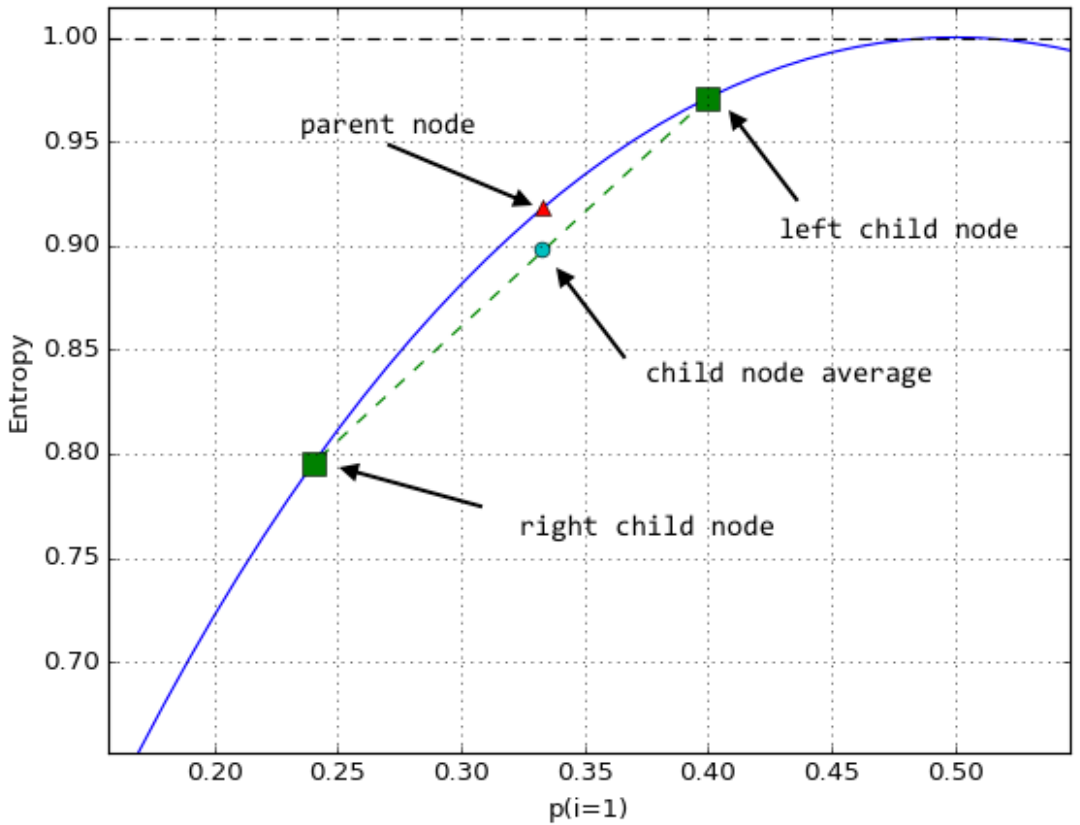


Figure 4.13. Comparison among the impurity measures for binary classification problems.



Entropy vs. classification error

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Q: Why is this true?

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Q: Why is this true?
A: We still need to look at impurity before & after the split.

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I *is the impurity measure,* N_j *denotes the number of records at child node* j*, and* N *denotes the number of records at the parent node.)*

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I *is the impurity measure,* N_j *denotes the number of records at child node* j*, and* N *denotes the number of records at the parent node.)*

When I is the entropy, this quantity is called the ***information gain****.*

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only (CART).

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only (CART).

Another way is to use a splitting criterion which explicitly penalizes the number of outcomes (C4.5)

We can use a function of the information gain called the **gain ratio** *to explicitly penalize high numbers of outcomes:*

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) log_2 p(v_i)}$$

(Where $\mathrm{p}(\mathrm{v_i})$ *refers to the probability of label* i *at node* v*)*

We can use a function of the information gain called the **gain ratio** *to explicitly penalize high numbers of outcomes:*

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) log_2 p(v_i)}$$

(Where $\mathrm{p}(\mathrm{v_i})$ *refers to the probability of label* i *at node* v*)*

NOTE

This is a form of regularization!

Quick check: what is information gain?

PREVENTING OVERFITTING

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

For example, we can stop when all records belong to the same class, or when all records have the same attributes.

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

For example, we can stop when all records belong to the same class, or when all records have the same attributes.

This is correct in principle, but would likely lead to overfitting.

One possibility is **pre-pruning**, *which involves setting a minimum threshold on the gain, and stopping when no split achieves a gain above this threshold.*

One possibility is **pre-pruning**, *which involves setting a minimum threshold on the gain, and stopping when no split achieves a gain above this threshold.*

This prevents overfitting, but is difficult to calibrate in practice (may preserve bias!)

Alternatively we could build the full tree, and then perform **pruning** *as a post-processing step.*

Alternatively we could build the full tree, and then perform **pruning** *as a post-processing step.*

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

The first approach is called **subtree replacement**, *and the second is* **subtree raising**.

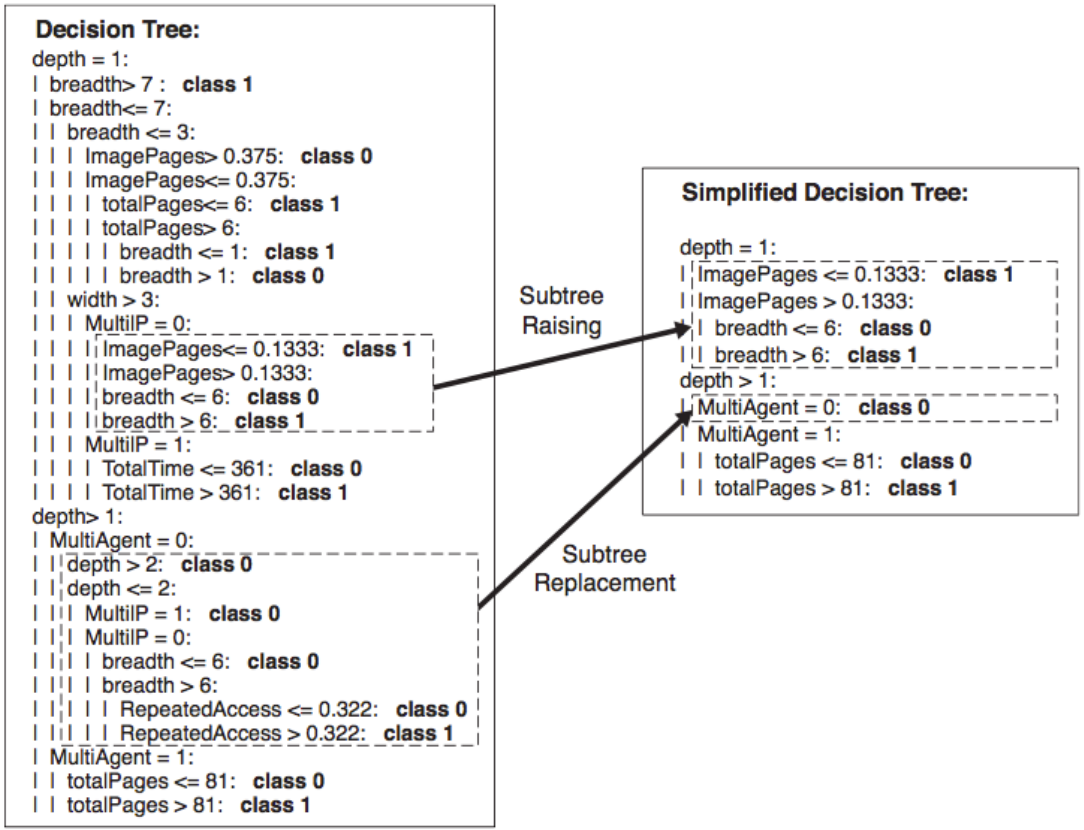


Figure 4.29. Post-pruning of the decision tree for Web robot detection.

LAB: DECISION TREES