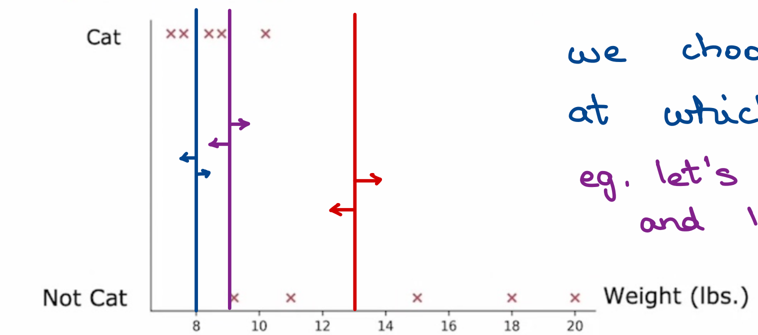
one-hot encoding works for classification, but for tasks where features can take any value continuous value functions are much better.

This is a dataset where we are classifying cats and dogs.

Ear shape	Face shape	Whiskers	Weight (lbs.)	Cat
Pointy	Round	Present	7.2	1
Floppy	Not round	Present	8.8	1
Floppy	Round	Absent	15	0
Pointy	Not round	Present	9.2	0
Pointy	Round	Present	8.4	1
Pointy	Round	Absent	7.6	1
Floppy	Not round	Absent	11	0
Pointy	Round	Absent	10.2	1
Floppy	Round	Absent	18	0
Floppy	Round	Absent	20	0

Splitting on a continuous variable



we choose a threshold at which we split it.

eg. let's take 8lbs, 9lbs and 13 lbs as the threshold.

When splitting on 8, we see how many cats and dogs are there and calculate the information gain.

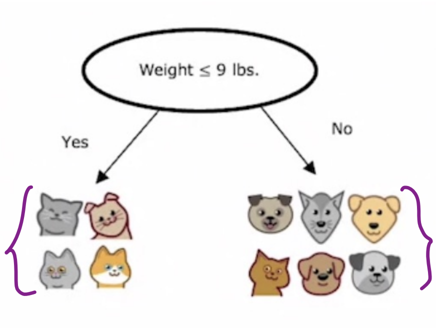
$$I.G = H(0.5) - \left[\frac{2}{10} H\left(\frac{2}{2}\right) + \frac{8}{10} H\left(\frac{3}{8}\right)\right] = 0.24$$

Repeat for all,

$$I.G = H(0.5) - \left[\frac{4}{10} H\left(\frac{4}{4}\right) + \frac{6}{10} H\left(\frac{1}{6}\right)\right] = 0.61$$

$$I.G = H(0.5) - \left[\frac{7}{10} H\left(\frac{5}{7}\right) + \frac{3}{10} H\left(\frac{0}{3}\right)\right] = 0.40$$

In this example, we get to know that 9 is the most adequate threshold.



can even build more recursively using the same technique.

In general, we take many different thresholds and then decide which is most suitable.

To create candidate threshold values, you look at the midpoints of consecutive values in a sorted list of features (in this case weights). Use the midpoint as the threshold.

eg. If you have 10 examples, there will be atleast 9 potential threshold values.

[2.3, 3.1, 4.0, 4.5, 5.0, 5.2, 6.0, 7.1, 8.0, 9.2]

$\frac{(2.3 + 3.1)}{2}$] midpoint

$= 2.7$

↑

threshold value.