

# Decision Tree

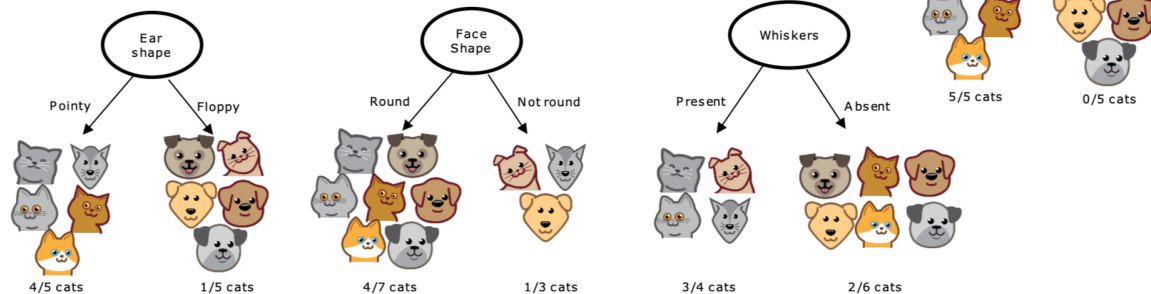
A decision tree is a model that looks like a tree, with each oval or rectangle being called a node. The model starts with a test example at the topmost node (the root node) and based on the value of the feature written inside, it will either go left or right. This process continues until it reaches a leaf node, which makes a prediction.

The process of building a decision tree involves several steps. Given a training set, the first step is to decide which feature to use at the root node. This is done via an algorithm that splits the training examples based on the value of the chosen feature. The second step is to decide what feature to use next, focusing on one branch of the decision tree at a time. This process continues until all examples in a node are of the same class (all cats or all dogs), at which point a leaf node is created that makes a prediction.

There are two key decisions to make when building a decision tree.

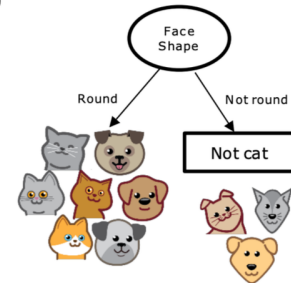
**Decision 1:** How to choose what feature to split on at each node?

Maximize purity (or minimize impurity)

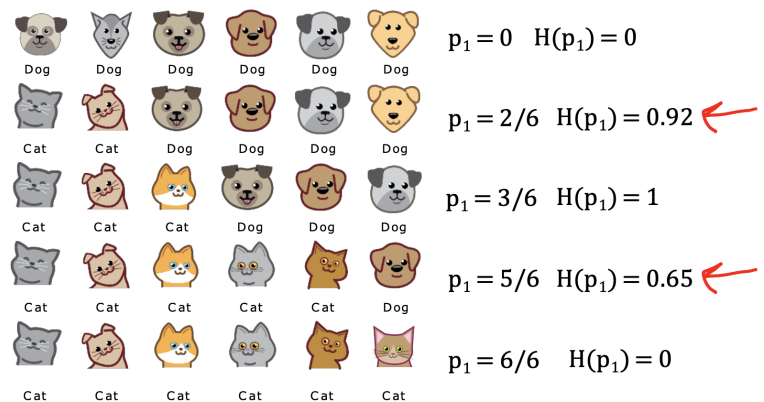
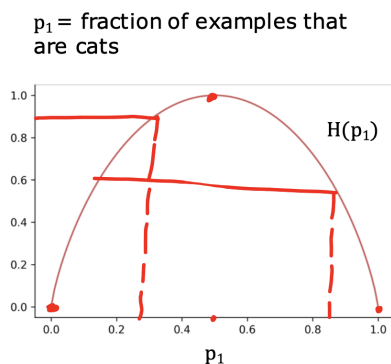


## Decision 2: When do you stop splitting?

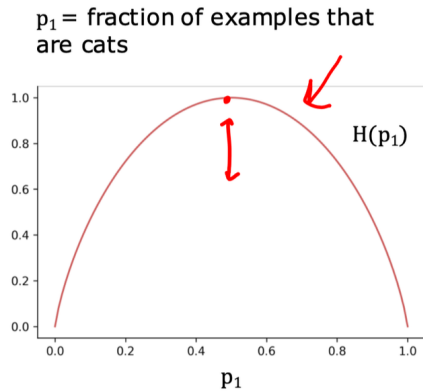
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



## Entropy as a measure of impurity



Entropy is a measure of the impurity of a set of data. It quantifies how mixed a set of examples is. If all examples are of the same class (all cats or all dogs), the entropy is 0, indicating a pure set. If the examples are evenly split between the classes, the entropy is 1, indicating maximum impurity.



$$p_0 = 1 - p_1$$

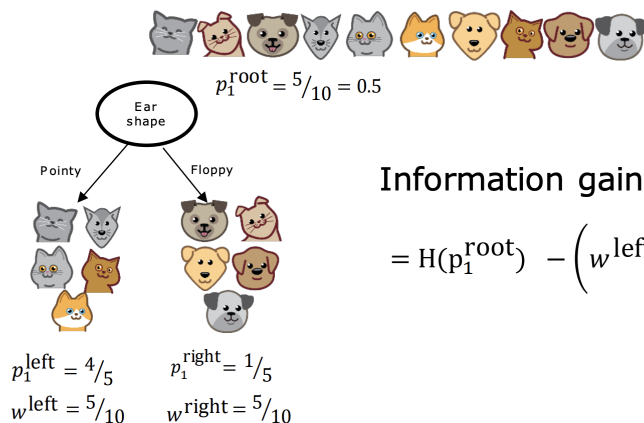
$$H(p_1) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

$$= -p_1 \log_2(p_1) - (1 - p_1) \log_2(1 - p_1)$$

Note: " $0 \log(0)$ " = 0

The entropy function is used in decision tree learning to decide which feature to split on at each node. The goal is to choose the feature that results in the greatest decrease in entropy, i.e., the greatest increase in purity. Other functions similar to entropy, such as the Gini criteria, can also be used for this purpose.

## Information Gain

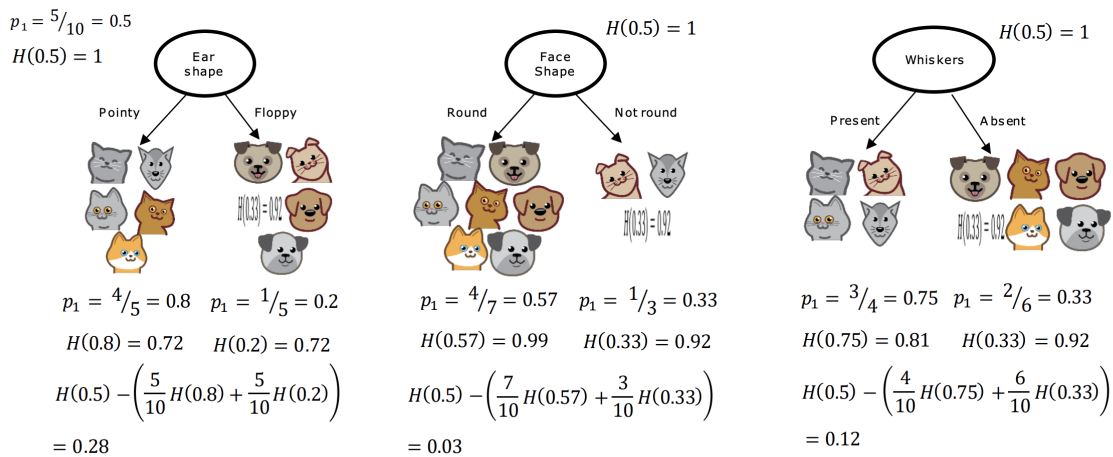


Information gain

$$= H(p_1^{\text{root}}) - \left( w^{\text{left}} H(p_1^{\text{left}}) + w^{\text{right}} H(p_1^{\text{right}}) \right)$$

Decision trees are built by choosing features to split on at each node. The chosen feature is the one that results in the greatest reduction in entropy, or the greatest increase in purity. This **reduction in entropy** is called **information gain**.

# Choosing a split



## Decision Tree Learning














- Start with all examples at the root node
- Calculate information gain for all possible features, and pick the one with the highest information gain
- Split dataset according to selected feature, and create left and right branches of the tree
- Keep repeating splitting process until stopping criteria is met:
  - When a node is 100% one class
  - When splitting a node will result in the tree exceeding a maximum depth
  - Information gain from additional splits is less than threshold
  - When number of examples in a node is below a threshold

# One hot encoding

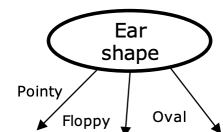
If a categorical feature can take on  $k$  values, create  $k$  binary features (0 or 1 valued).

One-hot encoding is a process by which categorical variables are converted into a form that could be provided to machine learning algorithms to improve prediction. When a categorical feature can take on  $k$  possible values, we replace it by creating  $k$  binary features that can only take on the values 0 or 1. For example, if we have a feature "ear shape" that can take on three possible values: "pointy", "floppy", and "oval", we would create three new binary features: "has pointy ears", "has floppy ears", and "has oval ears". Each of these new features can only take on a value of 0 or 1.











## Features with three possible values

	Ear shape ( $x_1$ )	Face shape ( $x_2$ )	Whiskers ( $x_3$ )	Cat ( $y$ )
	Pointy 	Round	Present	1
	Oval	Not round	Present	1
	Oval 	Round	Absent	0
	Pointy	Not round	Present	0
	Oval	Round	Present	1
	Pointy	Round	Absent	1
	Floppy 	Not round	Absent	0
	Oval	Round	Absent	1
	Floppy	Round	Absent	0
	Floppy	Round	Absent	0

3 possible values













# One hot encoding

Ear-shape	Pointy ears	Floppy ears	Oval ears	Face shape	Whiskers	Cat
 Pointy	1	0	0	Round	Present	1
 Oval	0	0	1	Not round	Present	1
 Oval	0	0	1	Round	Absent	0
 Pointy	1	0	0	Not round	Present	0
 Oval	0	0	1	Round	Present	1
 Pointy	1	0	0	Round	Absent	1
 Floppy	0	1	0	Not round	Absent	0
 Oval	0	0	1	Round	Absent	1
 Floppy	0	1	0	Round	Absent	0
 Floppy	0	1	0	Round	Absent	0

This method is called one-hot encoding because only one of these k binary features can take on the value 1 (hot) for each sample.











Note :- One-hot encoding can be used not only for decision tree learning but also for neural networks, linear regression, or logistic regression training.

## One hot encoding and neural networks

	Pointy ears	Floppy ears	Round ears	Face shape	Whiskers	Cat
 1	1	0	0	<del>Round</del> 1	<del>Present</del> 1	1
 0	0	0	1	<del>Not round</del> 0	<del>Present</del> 1	1
 0	0	0	1	<del>Round</del> 1	<del>Absent</del> 0	0
 1	1	0	0	<del>Not round</del> 0	Present 1	0
 0	0	0	1	Round 1	Present 1	1
 1	1	0	0	Round 1	Absent 0	1
 0	0	1	0	Not round 0	Absent 0	1
 0	0	0	1	Round 1	Absent 0	1
 0	0	1	0	Round 1	Absent 0	1
 0	0	1	0	Round 1	Absent 0	1

## Continuous valued features

# Continuous features

	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

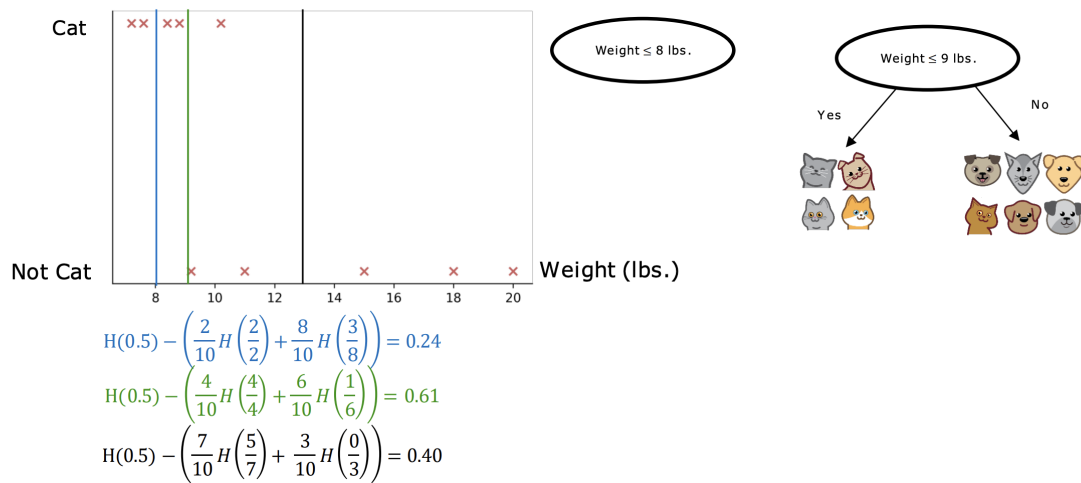
The decision tree algorithm can be modified to work with continuous values by considering different thresholds for the continuous feature and performing the usual information gain calculation.

For example, if you have a feature like the weight of an animal, you would consider different weight thresholds to split your data. For each threshold, you calculate the information gain, which is the entropy at the root node minus the weighted sum of the entropies of the left and right splits.

The threshold that gives the highest information gain is selected. If this information gain is higher than that of any other feature, you decide to split the node at that feature using the selected threshold.

This process is repeated recursively to build out the rest of the decision tree. This way, the decision tree algorithm can handle continuous features as well as discrete ones.

# Splitting on a continuous variable













In a more general case, you would sort all the examples according to the value of the continuous feature and consider all the midpoints between the sorted list of training examples as possible thresholds. This allows you to test multiple possible values for the threshold and pick the one that gives the highest information gain.

## Regression Trees

**Goal :-** To predict a numerical value rather than a class label.

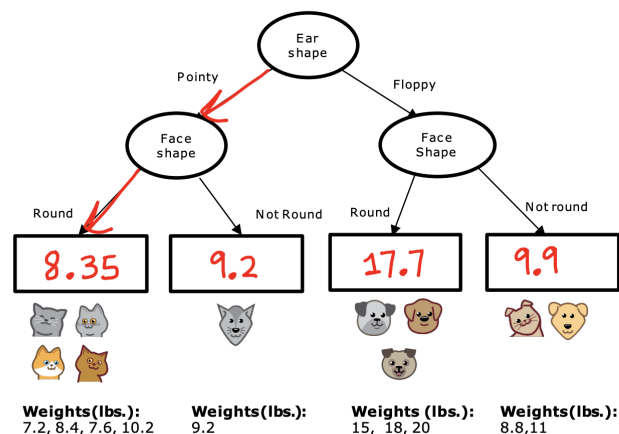


	Ear shape	Face shape	Whiskers	Weight (lbs.)
	Pointy	Round	Present	7.2
	Floppy	Not round	Present	8.8
	Floppy	Round	Absent	15
	Pointy	Not round	Present	9.2
	Pointy	Round	Present	8.4
	Pointy	Round	Absent	7.6
	Floppy	Not round	Absent	11
	Pointy	Round	Absent	10.2
	Floppy	Round	Absent	18
	Floppy	Round	Absent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

X
y

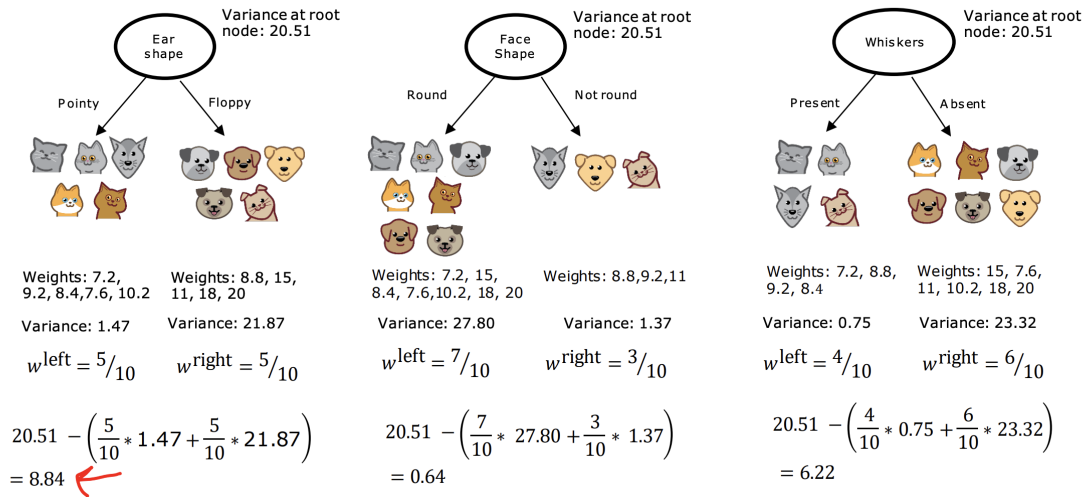
A regression tree is constructed similarly to a classification tree, but instead of making a prediction based on the majority class in the leaf node, it predicts the average value of the target variable for the training examples in that node.

## Regression with Decision Trees



When deciding which feature to split on, the goal is to reduce the variance of the target variable in each subset of the data. This is done by computing the weighted average variance for each possible split and choosing the one that results in the largest reduction in variance.

## Choosing a split



The process is repeated recursively on the subsets of data in each branch until a stopping criterion is met.