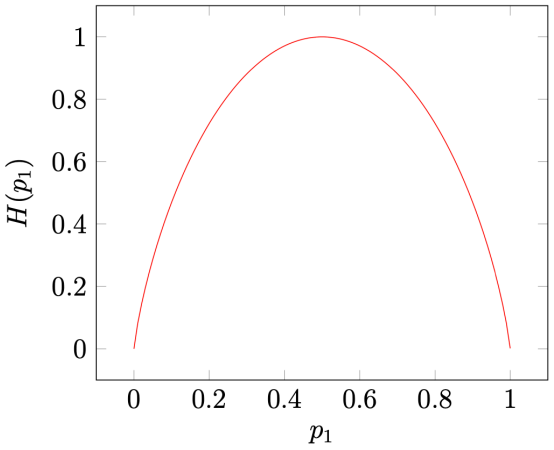
Decision Tree Learning

Measure purity

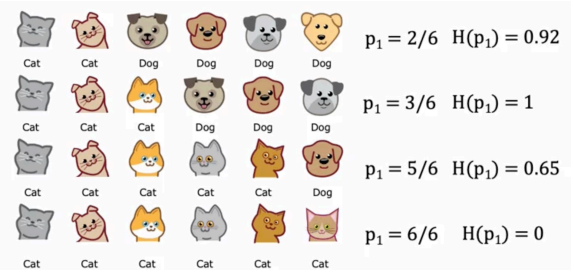
Entropy as a measure of impurity

- Entropy quantifies the impurity of a dataset; a pure set (all cats or all dogs) has an entropy of zero, while a mixed set (50% cats, 50% dogs) has the highest entropy of one.
- $p_0, \dots, p_n$ represent each fraction of n training examples for classifications



- The entropy function is defined mathematically as $H(p_1) = -p_1 log_2(p_1) - p_0 log_2(p_0)$, where
  - p_1 is the fraction of positive examples (cats).
  - p_0 is the fraction of negative examples (not cats).

- A set with three cats and three dogs $(p_1 = 0.5)$ has an entropy of 1, indicating ***maximum impurity***.
- A set with five cats and one dog $(p_1 \approx 0.83)$ has an entropy of about 0.65, showing less impurity than the 50-50 mix.



example of entropy

Then, with the expression: $p_0 = 1 - p_1$, the entropy function can be described as follows:

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

Entropy Behavior

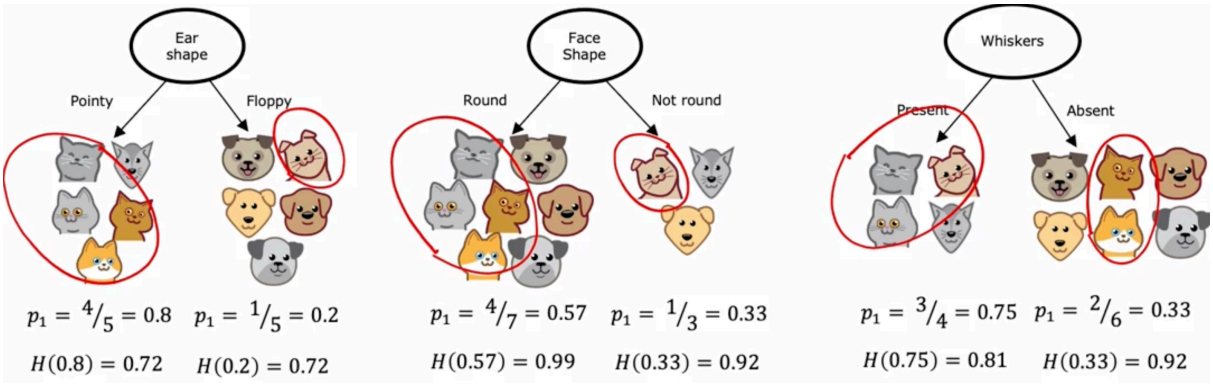
- As the proportion of one class increases (e.g., from 50% to 100% cats), entropy decreases from one to zero, indicating increasing purity.
- If p_1 or p_0 equals **zero**, the entropy is conventionally defined as **zero** to avoid undefined logarithmic expressions.

$$\text{Note: } \text{“}0\log(0)\text{”} = 0$$

Choosing a split: Information Gain

Choosing a split

Example:



❓ The **key** question we need to answer is, **given these three options of a feature to use at the root node, which one do we think works best?**

⇒ It turns out that **rather than looking at these entropy numbers and comparing them, it would be useful to take a weighted average of them.**

⇒ If there's a node with a lot of examples in it with high entropy that seems worse than if there was a node with just a few examples in it with high entropy. (example of Face Shape have all entropy values higher than Ear Shape)

- Because entropy, as a measure of **impurity**, is **worse** if we have a very large and impure dataset compared to just a few examples and a branch of the tree that is very impure.

❓ The **key** decision is, **of these three possible choices of features to use at the root node, which one do we want to use?**

⇒ In this example, splitting on **Ear Shape** resulted in the highest information gain, making it the best choice for the root node of the decision tree.

- When a dataset is split at a node based on a particular feature, it results in two new entropy values—one for the left branch and one for the right branch.
- After computing the entropy for each possible split (one per class per split), it's more **helpful** to **calculate a weighted average** of the entropy values rather than assessing them individually. With $p_1, \ldots, p_n$ refer to the nodes resulting from the split and is the fraction of split nodes from pre-split nodes

$$w^{left} H(p_1^{left}) + w^{right} H(p_2^{tight})$$

Ear shape	Face shape	Whiskers
$\frac{5}{10} \cdot H(0.8) + \frac{5}{10} \cdot H(0.2)$	$\frac{7}{10} \cdot H(0.57) + \frac{3}{10} \cdot H(0.33)$	$\frac{4}{10} \cdot H(0.75) + \frac{6}{10} \cdot H(0.33)$

- w^{left}, w^{right} can be seen as: $P(alt = T), P(alt = F)$

Information Gain Algorithm

$$\text{Imformation Gain (IG)} = H(p_1^{\text{root}}) - \left(w^{\text{left}} H(p_1^{\text{left}}) + w^{\text{right}} H(p_2^{\text{right}})\right)$$

- The second term is a ***weighted*** **average** impurity after you split; subtracting it from the parent's impurity tells you how much you gained. A higher IG means a better question for the tree.
- Information gain measures the reduction in entropy when a dataset is split based on a feature.
- The goal is to choose the feature that maximizes information gain, leading to purer subsets of data

More general from:

$$\text{Gain}(S, a) = H(S) - \sum_{v \in \text{Values}(a)} \frac{|S_v|}{S} H(S_v)$$

Putting it together - Build decision tree

Recursive Splitting

- 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 recursively **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

Using one-hot encoding of categorical features

❓ **How to handle categorical features that can take on more than two discrete values in machine learning?**

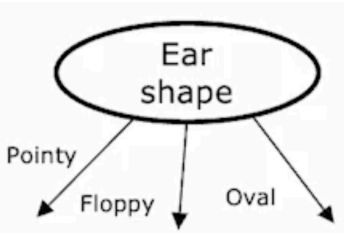
⇒ One-hot encoding

Feature with Three possible values

- For example, an ear shape feature with values like pointy, floppy, and oval is converted into three binary features: **pointy** ears, **floppy** ears, and **oval** ears.

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

Becomes:



One-hot encoding

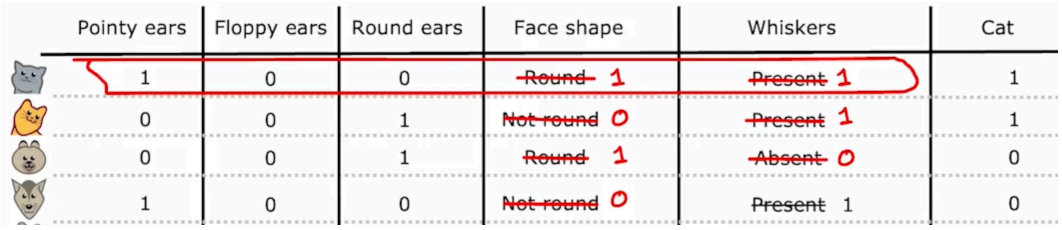
- One-hot encoding **transforms a categorical feature** with multiple values into **multiple binary features**.

~~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

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

❓
- Each new binary feature can only take values of 0 or 1, allowing decision tree algorithms to function effectively.
- This method ensures that only one of the binary features is "hot" (equal to 1) at any time, which is the basis for the term "one-hot encoding."

Broader Use in Machine Learning



	Pointy ears	Floppy ears	Round ears	Face shape	Whiskers	Cat
	1	0	0	~~Round~~ 1	~~Present~~ 1	1
	0	0	1	~~Not round~~ 0	~~Present~~ 1	1
	0	0	1	~~Round~~ 1	~~Absent~~ 0	0
	1	0	0	~~Not round~~ 0	~~Present~~ 1	0

- One-hot encoding is not limited to decision trees; it can also be applied in **training neural networks** and **logistic regression**.
- The technique allows for the encoding of categorical features into a numerical format suitable for various machine learning models.

Continuous valued features

Continuous Features

- Traditional decision trees split data based on features with discrete values (like “ear shape” or “whiskers”).
- But real-world data often includes **continuous features** (like “weight in pounds”), which can take any value.

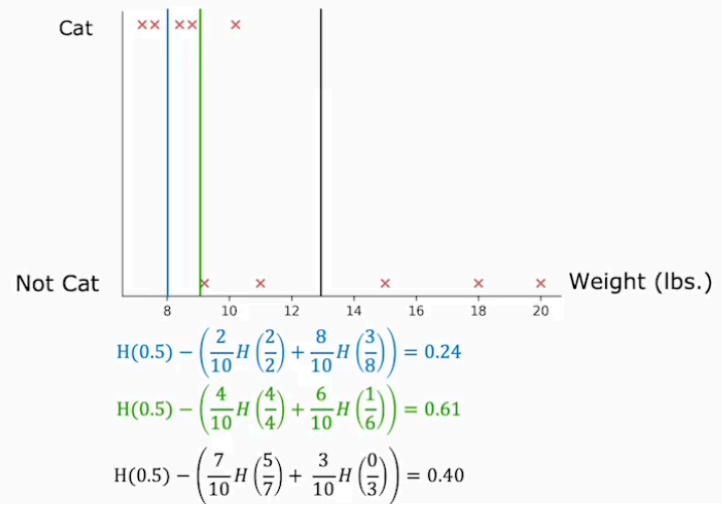
Example Dataset

	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

- The dataset now includes a **Weight (lbs.)** column.
- **Cats** and **dogs overlap** in **weight**, but generally, **cats** are **lighter**.

Spliting on a continuous variable

- **Weight ≤ 8 lbs:**
  - Left: 2 cats
  - Right: 3 cats, 5 dogs
  - Information gain: **0.24**
- **Weight ≤ 9 lbs:**
  - Left: 4 cats
  - Right: 1 cat, 5 dogs
  - Information gain: **0.61**
- **Weight ≤ 13 lbs:**
  - Left: 5 cats, 2 dogs
  - Right: 3 dogs
  - Information gain: **0.40**



- **Best split: Weight ≤ 9 lbs** (**highest** info gain)

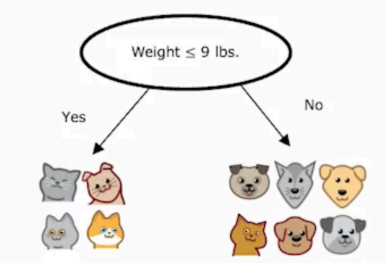
How Thresholds Are Chosen

- The algorithm sorts all the weights and tests splits at every point between two consecutive weights.

- For 10 examples, it tries 9 possible thresholds.

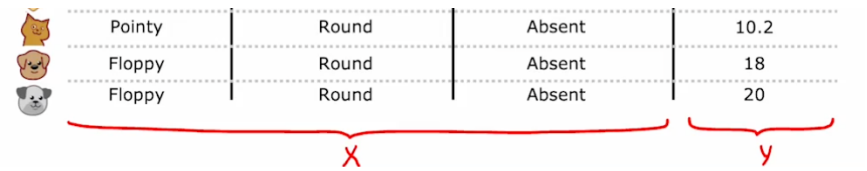
Decision Tree Construction

- If splitting on weight gives the highest information gain (compared to other features), the tree splits on weight at the best threshold.
- The process repeats recursively on each subset.

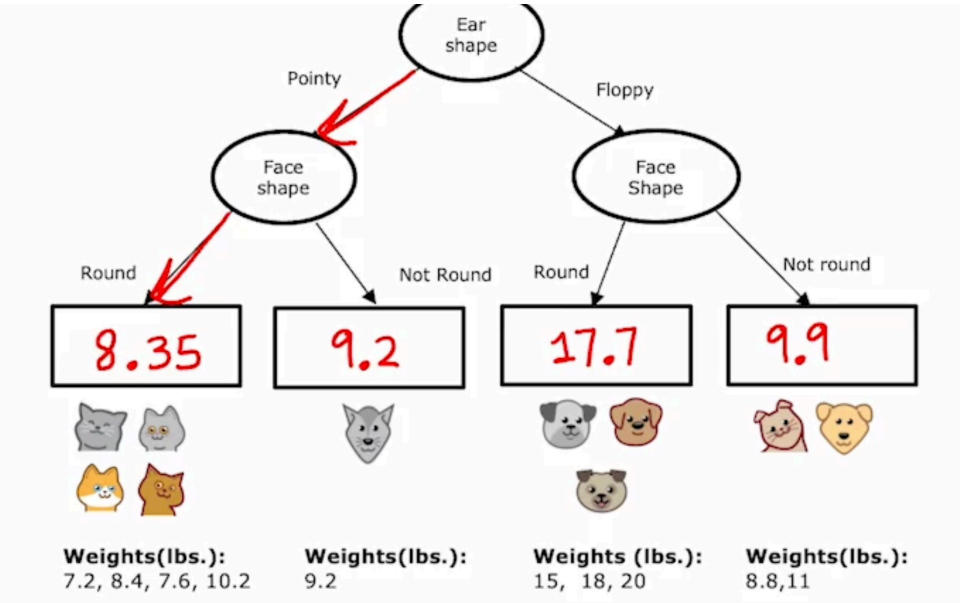


Regression with Decision Trees: Predicting a number

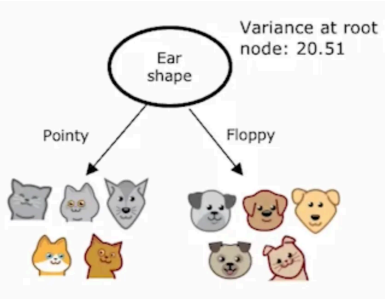
- Regression trees predict a continuous output Y, such as the weight of an animal, based on input features X.



- The tree structure involves nodes that split based on features, with predictions made at leaf nodes by averaging the target values of training examples.



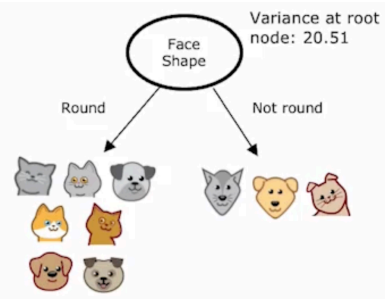
Choosing a split



- **Left**: *Pointy* (weights: $7.2, 9.2, 8.4, 7.6, 10.2$)
- **Right**: *Floppy* (weights: $8.8, 15, 11, 18, 20$)

A. Split on Ear Shape

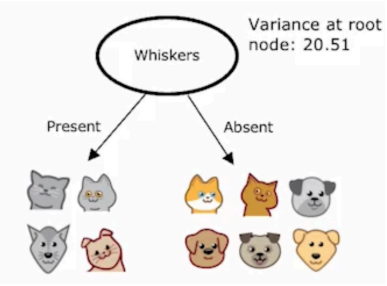
- **Left Variance**: 1.47 and $w^{left} = \frac{5}{10}$
- **Right Variance**: 21.87 and $w^{right} = \frac{5}{10}$
- **Weighted variance:** $w^{left} \times 1.47 + w^{right} \times 21.87 = \frac{5}{10} \times 1.47 + \frac{5}{10} \times 21.87 = 11.67$
- Reduction: $20.51 - 11.67 = 8.84$



B. Split on Face Shape

- **Left Variance**: 27.80 and $w^{left} = \frac{7}{10}$
- **Right Variance**: 1.37 and $w^{right} = \frac{3}{10}$

- **Left**: *Round*
  (weights:
  $7.2, 15, 8.4, 7.6, 10.2, 18, 20$)
- **Right**: *Not round*
  (weights: $8.8, 9.2, 11$)



- **Left:** *Present*
  (weights: $7.2, 8.8, 9.2, 8.4$)
- **Right:** *Absent*
  (weights: $15, 7.6, 11, 10.2, 18, 20$
)

- **Weighted variance:**
  $w^{left} \times 27.80 + w^{right} \times 1.37 = \frac{7}{10} \times 27.80 + \frac{3}{10} \times 1.37 = 19.87$
- Reduction: $20.51 - 19.87 = 0.64$

C. Split on Whiskers

- **Left Variance:** 0.75 and $w^{left} = \frac{4}{10}$
- **Right Variance**: 23.32 and $w^{left} = \frac{6}{10}$
- **Weighted variance:**
  $w^{left} + 0.75 + w^{right} \times 23.32 = \frac{4}{10} \times 0.75 + \frac{6}{10} \times 23.32 = 14.29$
- Reduction: $20.51 - 14.29 = 6.22$

Which Split is Best?

- **Ear shape** gives the largest reduction in variance (8.84), so the tree splits on ear shape at the root.

Example:

Dataset

After using one-hot encoding

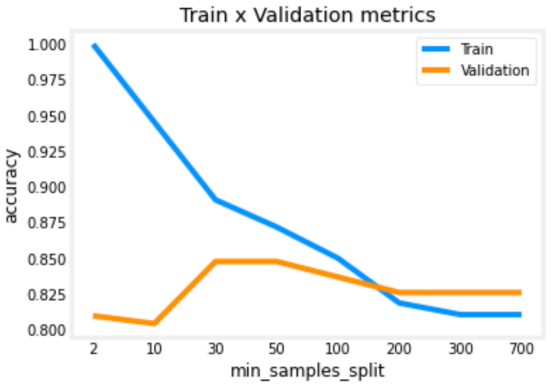
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_F	Sex_M	ChestPainType_ASY	...	ChestPainType_NAP	ChestPainType_TA	F
0	40	140	289	0	172	0.0	0	0	1	0	...	0	0	
1	49	160	180	0	156	1.0	1	1	0	0	...	1	0	
2	37	130	283	0	98	0.0	0	0	1	0	...	0	0	
3	48	138	214	0	108	1.5	1	1	0	1	...	0	0	
4	54	150	195	0	122	0.0	0	0	1	0	...	1	0	

Decision Tree

The hyperparameter `min_samples_split` controls **the minimum number of samples required to split an internal node**.

Note how increasing the the number of `min_samples_split` reduces overfitting.

- Increasing `min_samples_split` from 10 to 30, and from 30 to 50, even though it does not improve the validation accuracy, it brings the training accuracy closer to it, showing a reduction in overfitting.



Explain

Here's the **situation**:

- Each **row = one patient**
- The **columns = patient features**
- You're training a DecisionTreeClassifier to predict HeartDisease (target)

Suppose you use min_samples_split = 10

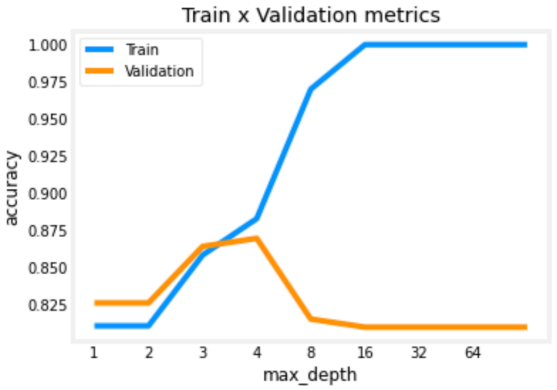
Let's say the decision tree is building a node with 8 patient rows during training:

- Since 8 < 10 → this node **cannot split further**, becomes a **leaf**
- No matter how diverse the samples are, it won't split unless it has **at least 10 patients**

Let's do the same experiment with `max_depth`.

We can see that in general, reducing `max_depth` can help to reduce overfitting.

- Reducing `max_depth` from 8 to 4 increases validation accuracy closer to training accuracy, while significantly reducing training accuracy.
- The validation accuracy reaches the highest at `tree_depth=4`.
- When the `max_depth` is smaller than 3, both training and validation accuracy decreases. The tree cannot make enough splits to distinguish positives from negatives (the model is underfitting the training set).



- When the `max_depth` is too high (>= 5), validation accuracy decreases while training accuracy increases, indicating that the model is overfitting to the training set.

So we can choose the best values for these two hyper-parameters for our model to be:

- `max_depth = 4`
- `min_samples_split = 50`

```
Metrics train:
    Accuracy score: 0.8583
Metrics validation:
    Accuracy score: 0.8641
```