Week 6

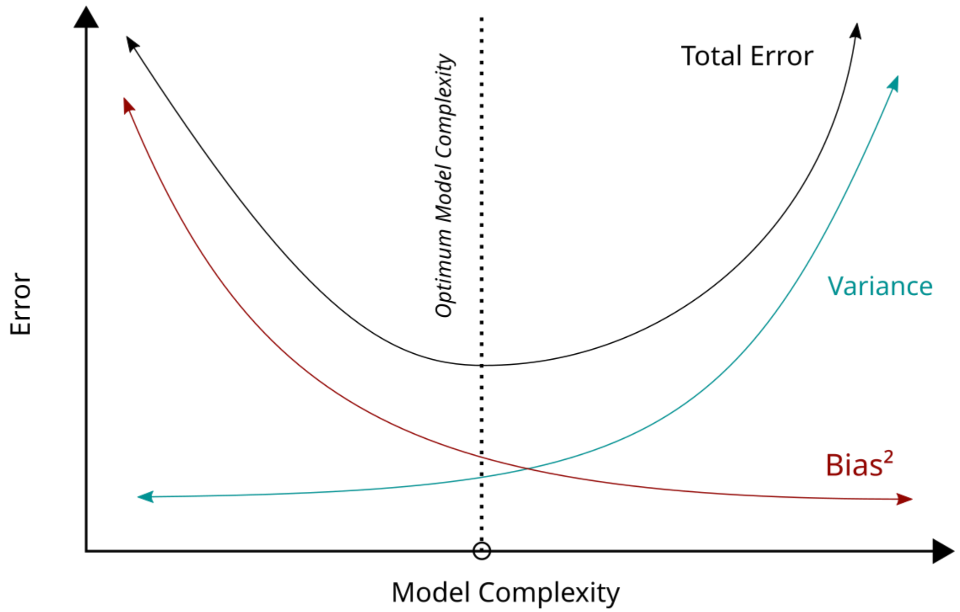
Agenda

- Bias and variance trade off
- Decision Tree
- Entropy
- Information Gain

What Is the Bias–Variance Trade-Off?

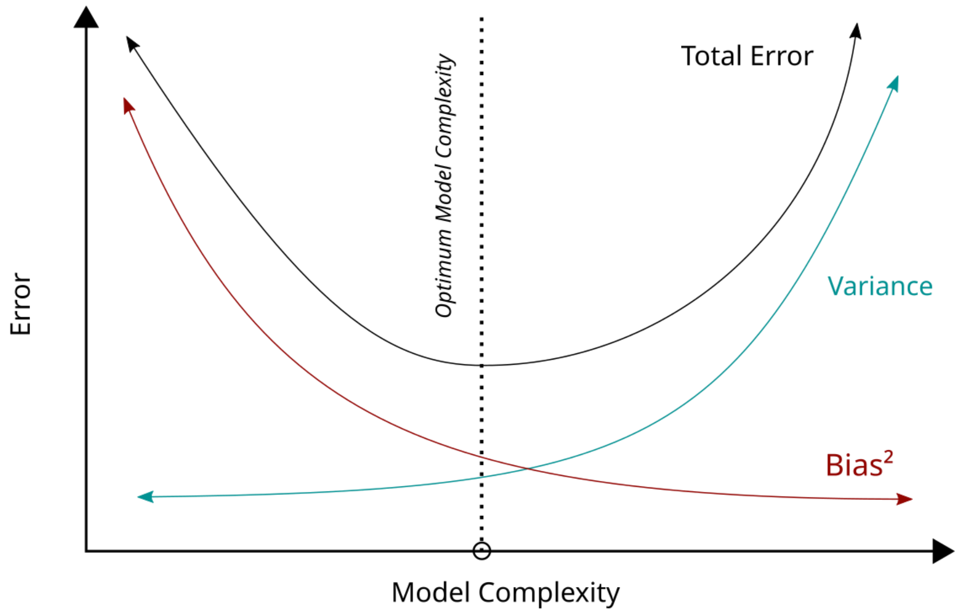
A key concept explaining why models perform poorly when they are too simple or too complex.

- **Bias**: Error due to oversimplifying the model.
- **Variance**: Error due to being too sensitive to training data.
- The trade-off: Finding the right balance for the lowest total error



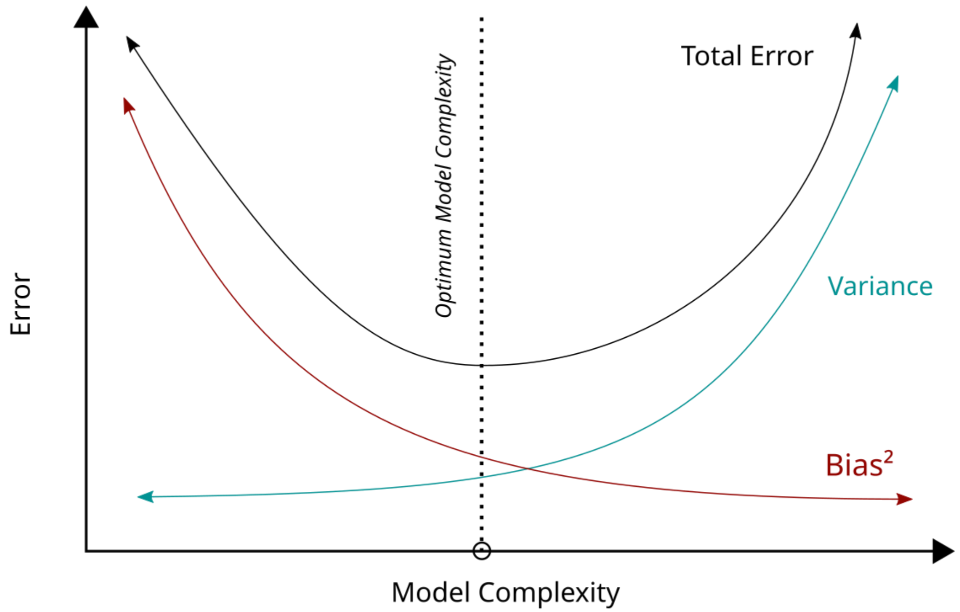
Understanding Bias

- **High Bias = Underfitting**
    - Model assumptions are too strong (e.g., linear line for nonlinear data).
    - Fails to capture important patterns in the data.
- Example: Predicting housing prices using only “number of rooms.”
- Visual: Flat or overly simple prediction line.



Understanding Variance

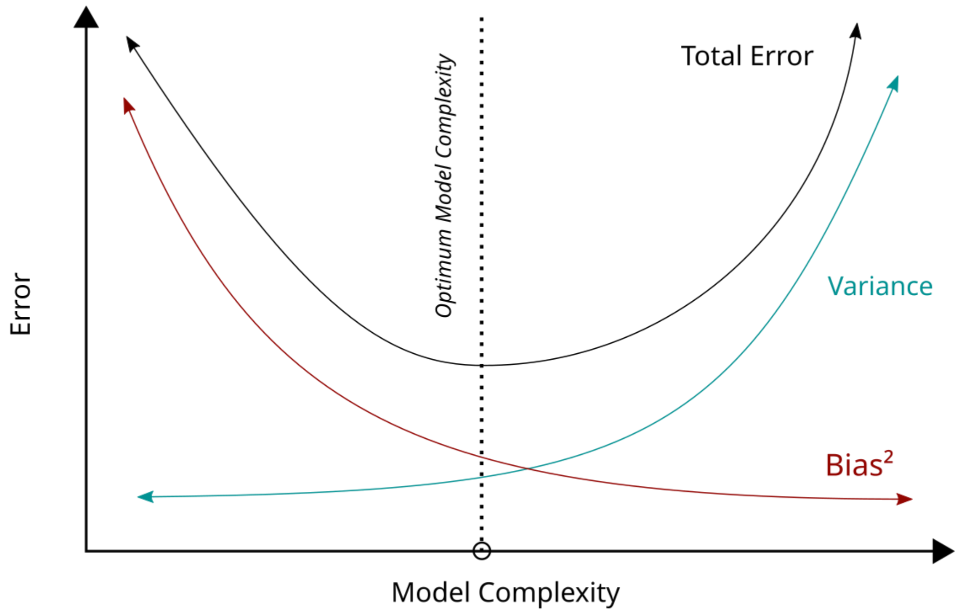
- **High Variance = Overfitting**
  - Model captures noise in the training data.
  - Performs well on training data but poorly on new data.
- Example: Deep decision tree that fits every data point perfectly.
- Visual: Extremely wiggly curve matching all training points.



The Trade-Off Curve

Total Error = $Bias^2$ + Variance + Irreducible Error.

- As model complexity increases:
  - Bias decreases.
  - Variance increases.
- Ideal model lies at the point where total error is minimized.

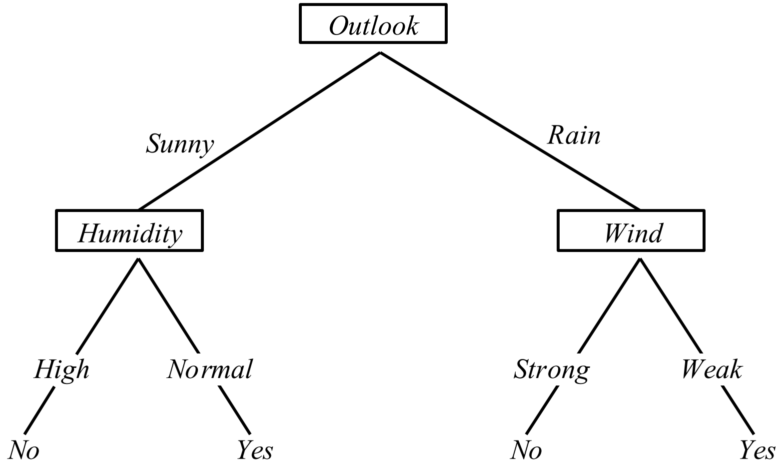


Practical Implications

- To control bias–variance:
  - Use cross-validation for performance estimation.
  - Apply regularization (L1/L2) to prevent overfitting.

- Goal: Consistent performance on unseen data.

Decision Trees Represent Rules



$$(Outlook = Sunny \ \wedge \ Humidity = Normal)$$

$$(Outlook = Rain \ \wedge \ Wind = Weak)$$

These rules make it easy to explain how predictions are made.
Because of this decision trees are known as transparent algorithms *or white box algorithms*.

Overview of Decision Trees

A decision tree is a simple transparent machine learning technique that uses the concept of a decision tree to make predictions.

The learned function is represented in the form of a tree.

Predictions are made by traversing the tree comparing feature values to tree nodes.

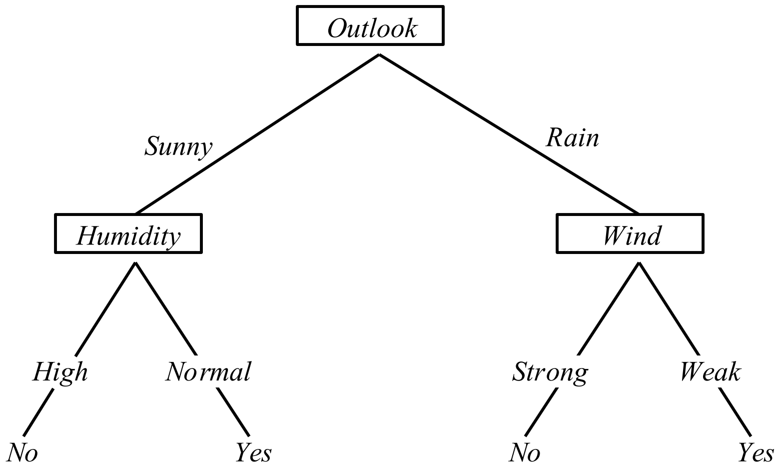
Decision trees can be used for classification (output is category like “spam”/”not spam” or regression problems (continuous real value).

Overview of Decision Trees

The tree nodes represent features, tree edges represent the direction to go based on feature values, and leaves represent the prediction.

Each node of the tree specifies a test of a feature and the branch corresponds to one of the possible values of the feature.

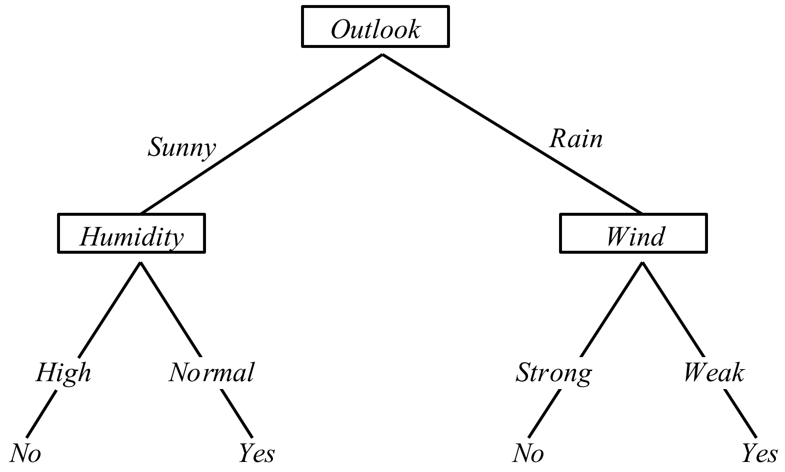
Predictions are made starting at the tree root and traversing the tree comparing feature values to tree nodes until a leaf node is encountered and a prediction is made.



Training Data

Outlook	Humidity	Wind	Play Tennis
Sunny	High	Weak	No
Rain	High	Strong	No
Sunny	Normal	Strong	Yes
Rain	High	Weak	Yes

Decision trees are a supervised learning algorithm.
They are built using training data.



Once the tree is built, use it to make predictions.

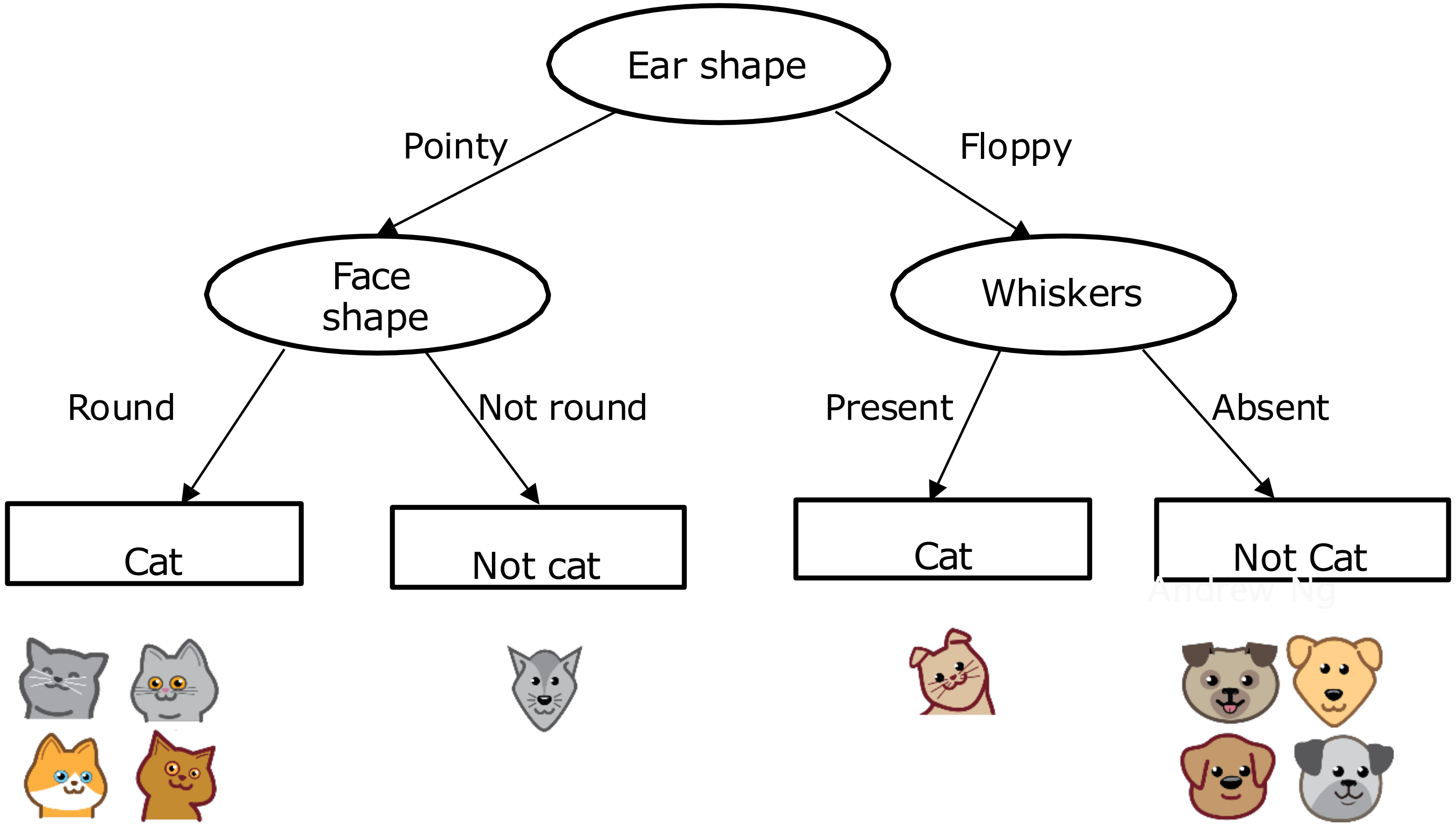
What would the tree predict on a rainy day with high humidity and weak wind?

Cat classification example

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

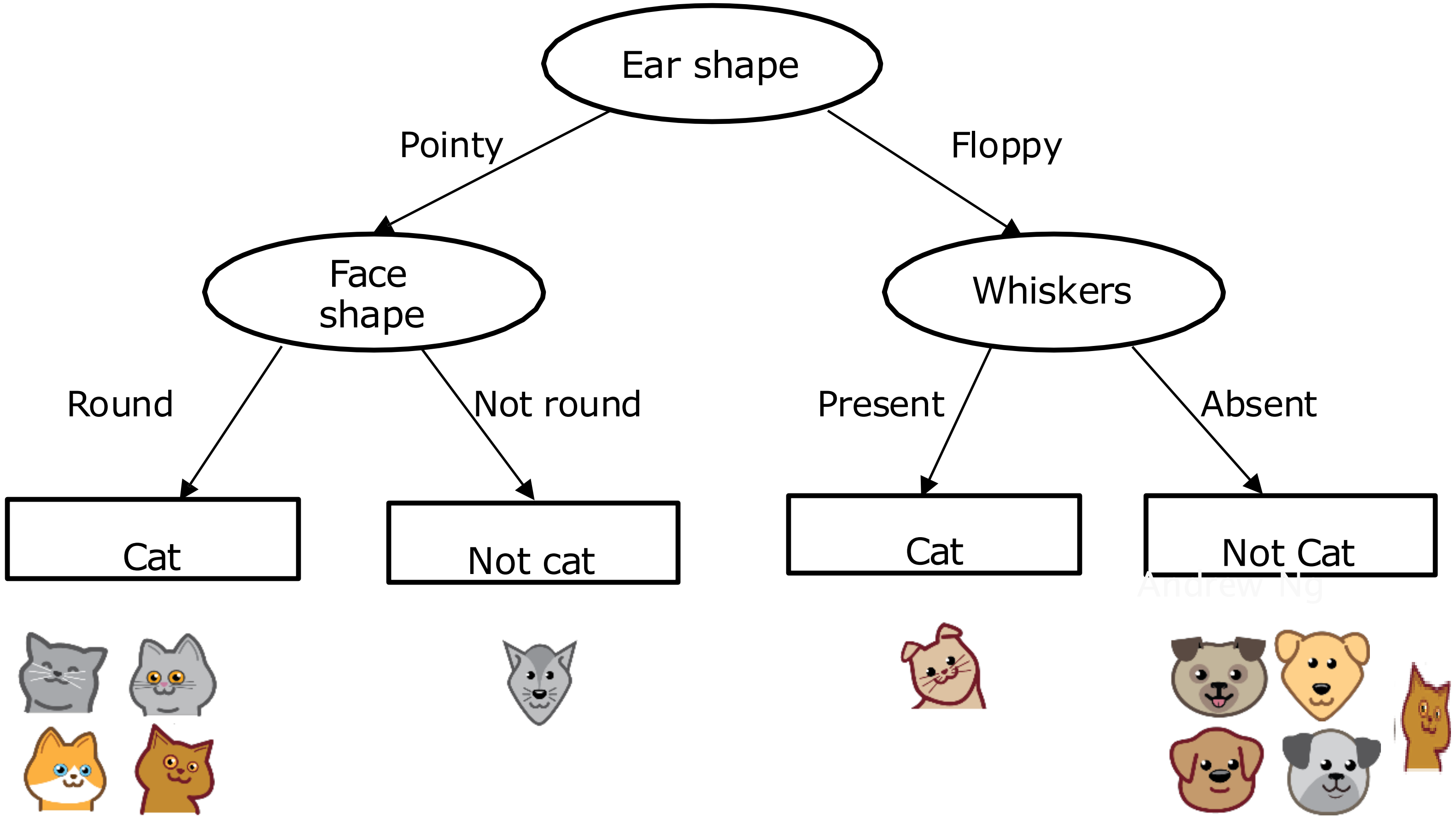
Categorical (discrete values)

Leaf Nodes



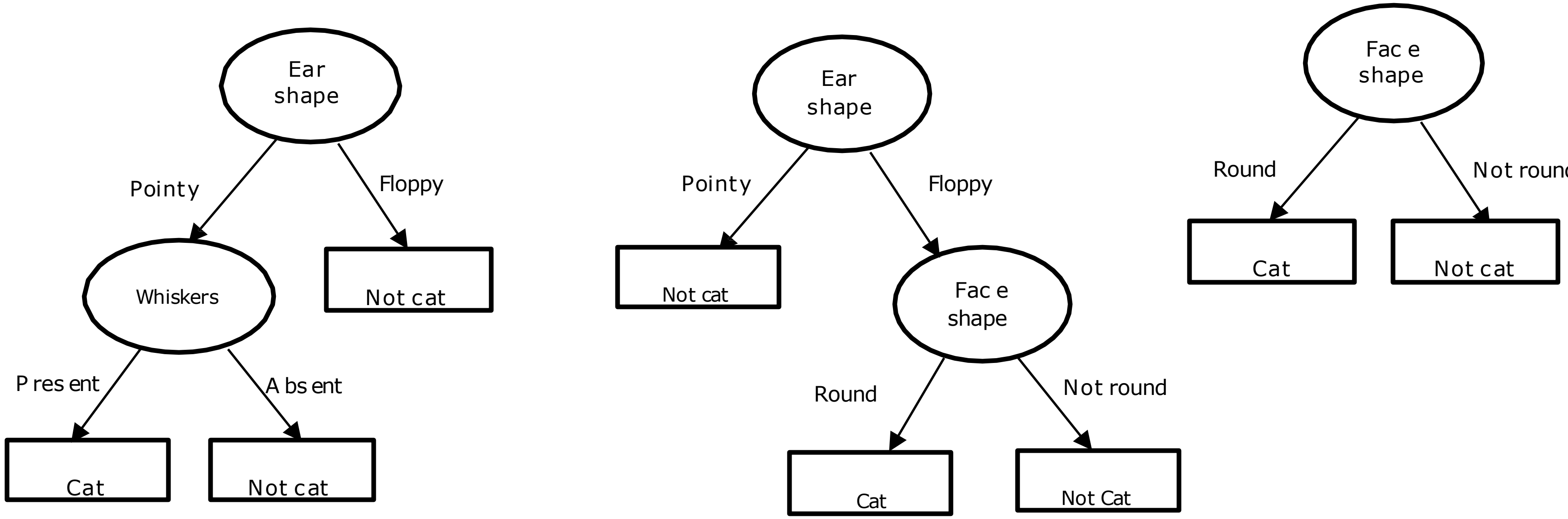
Leaf nodes hold the Prediction

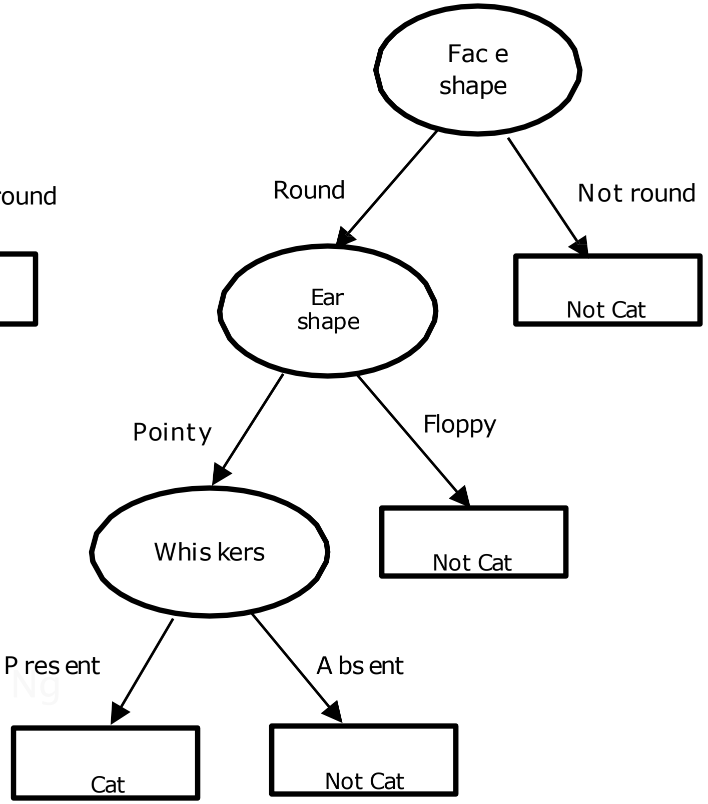
Majority Vote



Notice a cat is in the Not Cat prediction. Prediction is the majority class.

Decision Tree





There are many potential decision trees one can build with the training data.
Which one is the best?

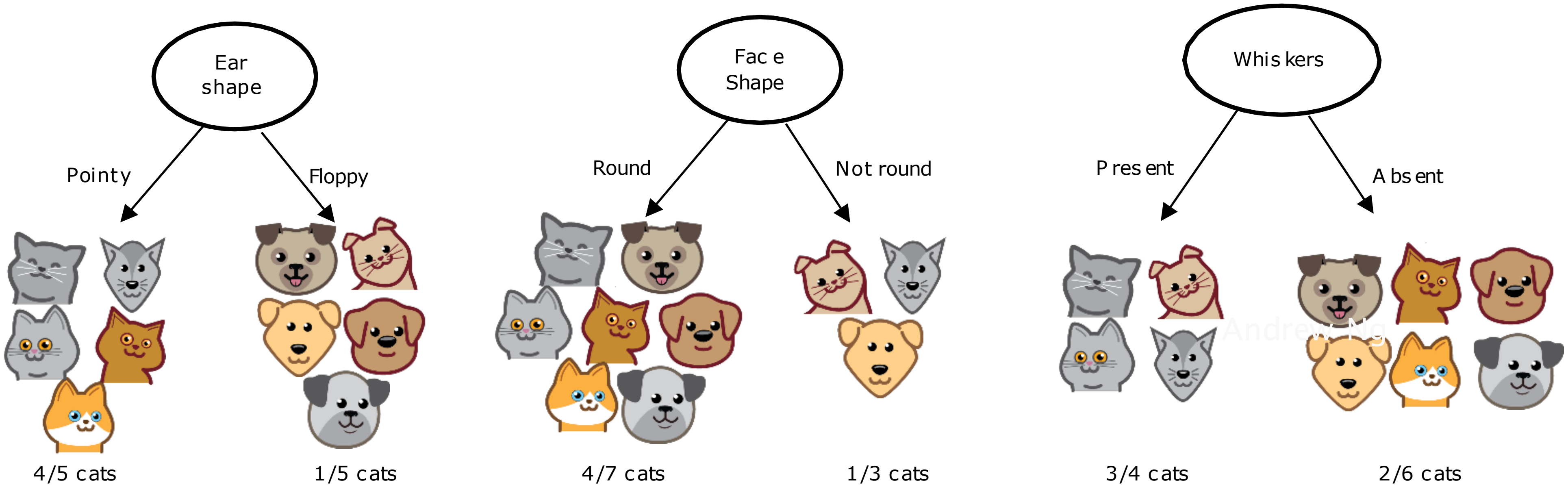
Decision Tree Learning

How to choose a feature for each node?
At the root node, start with all samples in the training set.

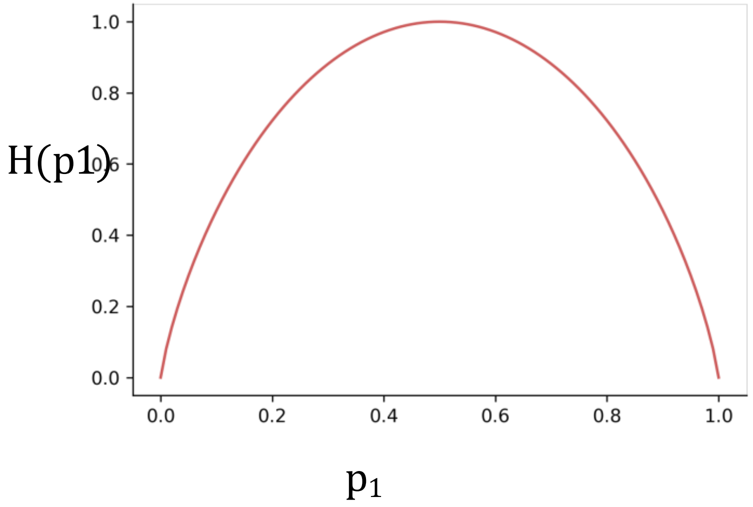
Decision Tree Learning

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

Choose the node that maximizes purity (or minimizes impurity). A pure node is a node where all samples are of the same class.



Use entropy to measure the impurity of the samples at a node – $h(p_1)$



p_1 = fraction of samples that are the positive class
p_0 = fraction of samples that are the negative class $(1 - p_1)$

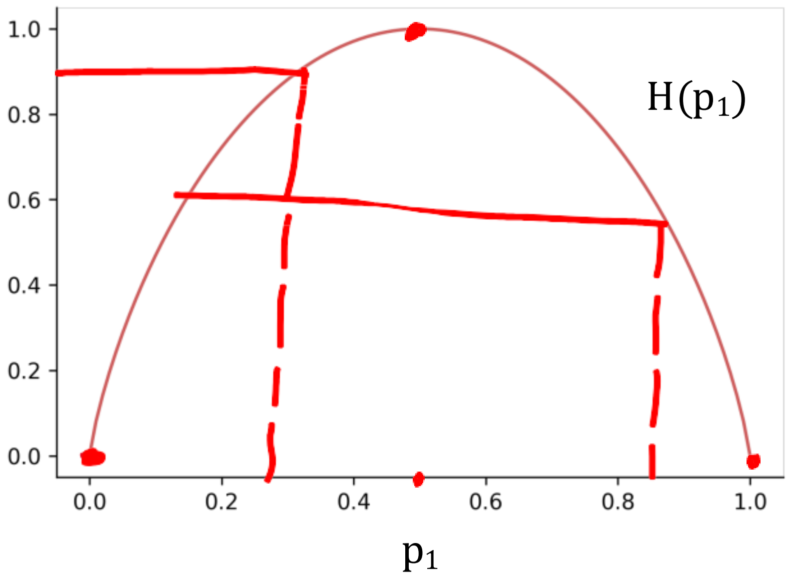
$$h(p_1) = -p_1 \log_2(p_1) - (1 - p_1)\log_2(1 - p_1)$$

Entropy is highest when the node is **most uncertain (p_1 = 0.5)**
and lowest (zero) when all samples belong to one class (pure node).

You can use entropy or the gini index. We will use entropy.

Entropy as a measure of impurity

$p_1 =$ fraction of examples that are cats



$p_1 = 0 \quad H(p_1) = 0$

$p_1 = 2/6 \quad H(p_1) = 0.92$

$p_1 = 3/6 \quad H(p_1) = 1$

$p_1 = 5/6 \quad H(p_1) = 0.65$

$p_1 = 6/6 \quad H(p_1) = 0$

Information Gain

We want to select a feature that will minimize entropy in the left and right nodes after the split.

How do we know which feature will do that?

Use the formula for information gain.

Information Gain

Information gain tells us how much a feature will reduce entropy after the split. We want the feature with the greatest information gain.

$$\text{Information Gain} = H(\text{parent}) - \left(\frac{N_{\text{left}}}{N_{\text{parent}}} H(\text{left}) + \frac{N_{\text{right}}}{N_{\text{parent}}} H(\text{right}) \right)$$

$H(\text{parent})$: Entropy of the parent node before splitting

$H(\text{left})$: Entropy of the left child node after splitting

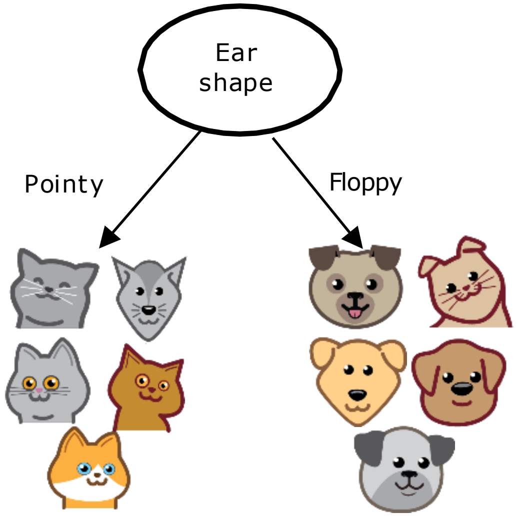
$H(\text{right})$: Entropy of the right child node after splitting

N_{left}: Number of samples in the left child node

N_{right}: Number of samples in the right child node

N_{parent}: Total number of samples before the split

Information Gain



Information gain

$$= \mathrm{h}(\mathrm{p}_1^{\mathrm{root}}) - \left(w^{\mathrm{left}} H\left(p_1^{\mathrm{left}}\right) + w^{\mathrm{right}} H\left(p_1^{\mathrm{right}}\right) \right)$$

$$\text{Information Gain} = H(\text{parent}) - \left(\frac{N_{\text{left}}}{N_{\text{parent}}} H(\text{left}) + \frac{N_{\text{right}}}{N_{\text{parent}}} H(\text{right}) \right)$$

$H(\text{parent})$: Entropy of the parent node before splitting

$H(\text{left})$: Entropy of the left child node after splitting

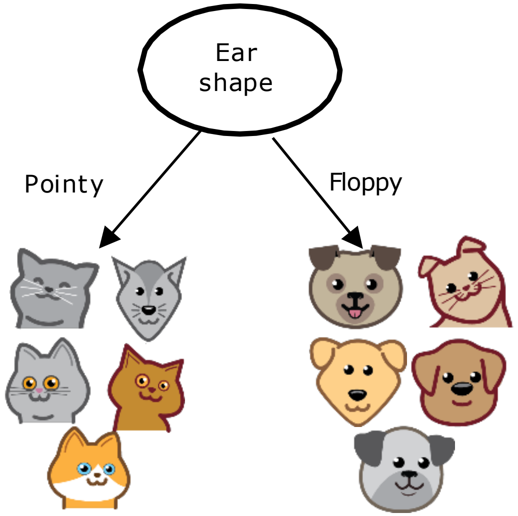
$H(\text{right})$: Entropy of the right child node after splitting

N_{left}: Number of samples in the left child node

N_{right}: Number of samples in the right child node

N_{parent}: Total number of samples before the split

Information Gain



Information gain

$$= \mathrm{h}(\mathrm{p}_1^{\mathrm{root}}) - \left(w^{\mathrm{left}} H\left(p_1^{\mathrm{left}}\right) + w^{\mathrm{right}} H\left(p_1^{\mathrm{right}}\right)\right)$$

$h(p^{root})$ = 5 of 10 samples are cats, h(.5) = 1

w^{left} = percentage of samples in left tree = 5/10
$h(p^{left})$ – entropy of left tree – 4 out of 5 are cats, h(.8)

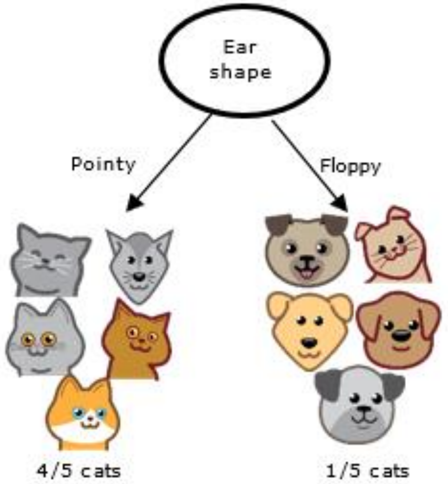
w^{right} = percentage of samples in right tree = 5/10
$h(p^{right})$ – entropy of right tree – 1 out of 5 are cats, h(.2)

Choosing a split

Entropy of root node is 1

$p_1 = 5/10$

$h(.5) = 1$



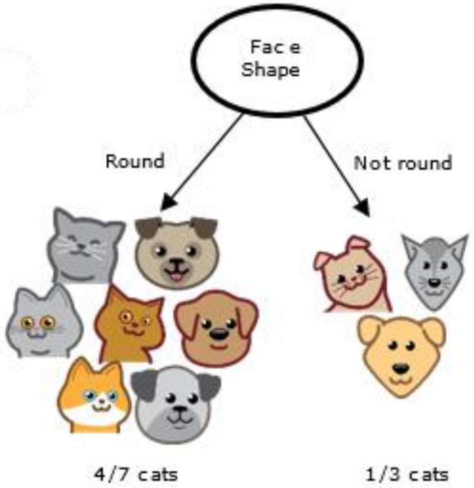
$p_1 = 0.8$ $\quad$ $p_1 = 0.2$

$h(0.8) = 0.72$ $\quad$ $h(0.2) = 0.72$

$$h(0.5) - \left(\frac{5}{10} h(0.8) + \frac{5}{10} h(0.2)\right)$$

$= 0.28$

Select ear shape with the largest information gain value

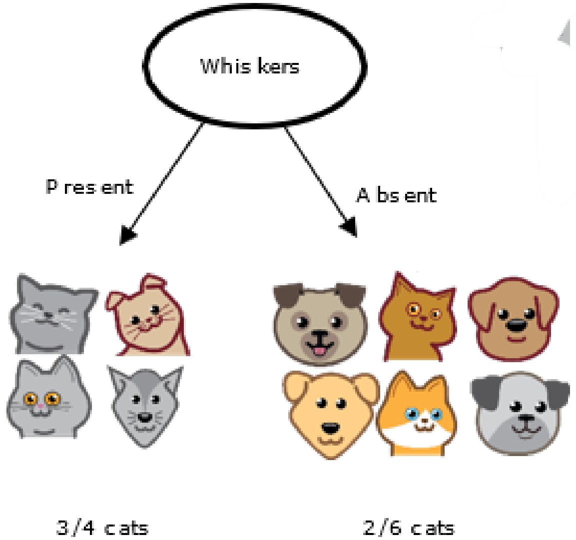


$p_1 = 0.57$ $\quad$ $p_1 = 0.33$

$h(0.57) = 0.99$ $\quad$ $h(0.33) = 0.92$

$$h(0.5) - \left(\frac{7}{10} h(0.57) + \frac{3}{10} h(0.33)\right)$$

$= 0.03$



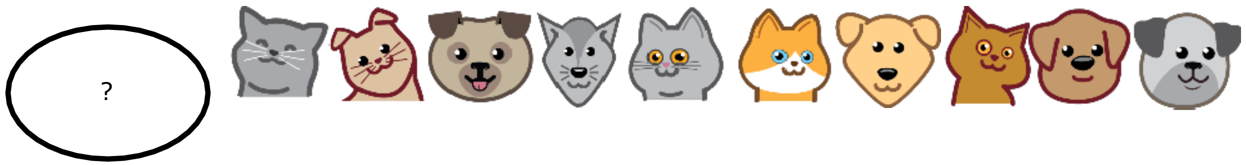
$p_1 = 0.75$ $\quad$ $p_1 = 0.33$

$h(0.75) = 0.81$ $\quad$ $h(0.33) = 0.92$

$$H(0.5) - \left(\frac{4}{10} h(0.75) + \frac{6}{10} h(0.33)\right)$$

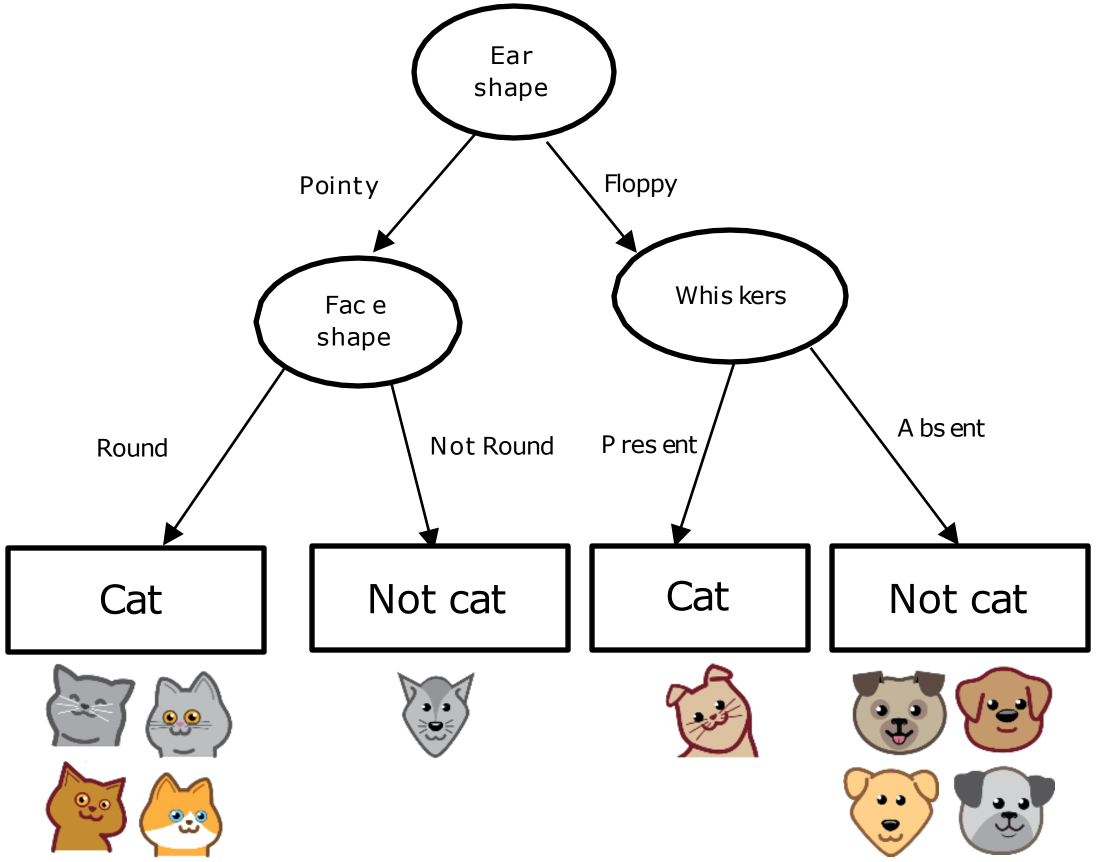
$= 0.12$

Decision Trees are built using a recursive algorithm



Calculate information gain and decide on feature to split on

Recursive splitting



Recursive algorithm

Continue to split until all leaf nodes are pure or other stopping criteria.

Decision Tree Learning

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

Decision Trees Have High Variance

If you allow a decision to continue to split and grow, they can fit the training data perfectly.

Use regularization techniques to prevent overfitting/high variance.

Decision Tree Pruning/Regularization

Stop splitting early to prevent overfitting. This is known as ***regularization*** for decision trees.

- When splitting a node will result in the tree exceeding a maximum depth (set a maximum depth)
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold (set a minimum samples per node)

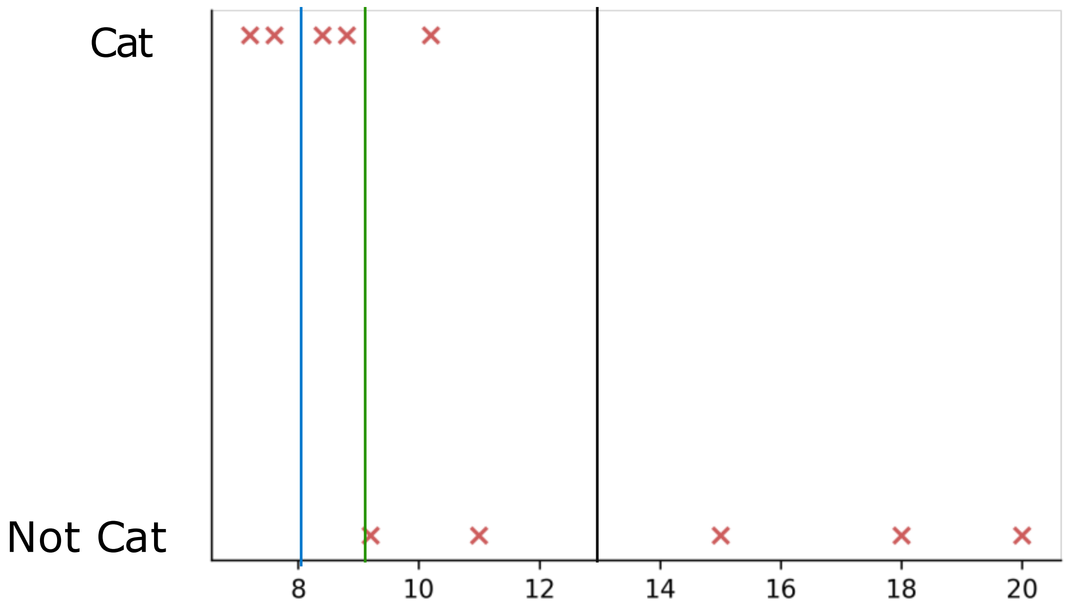
Decision Tree Learning

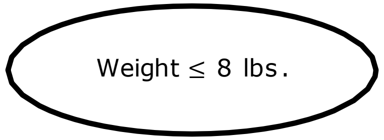
- Start with all examples at the root node
- Calculate information gain for all possible features, and pick the one with the highest information gain
- Split dataset according to selected feature, and create left and right branches of the tree
- Keep repeating splitting process until stopping criteria 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

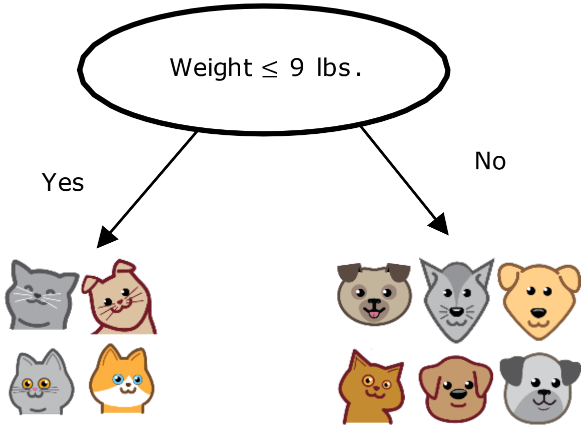
Continuous features

	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

Splitting on a continuous variable





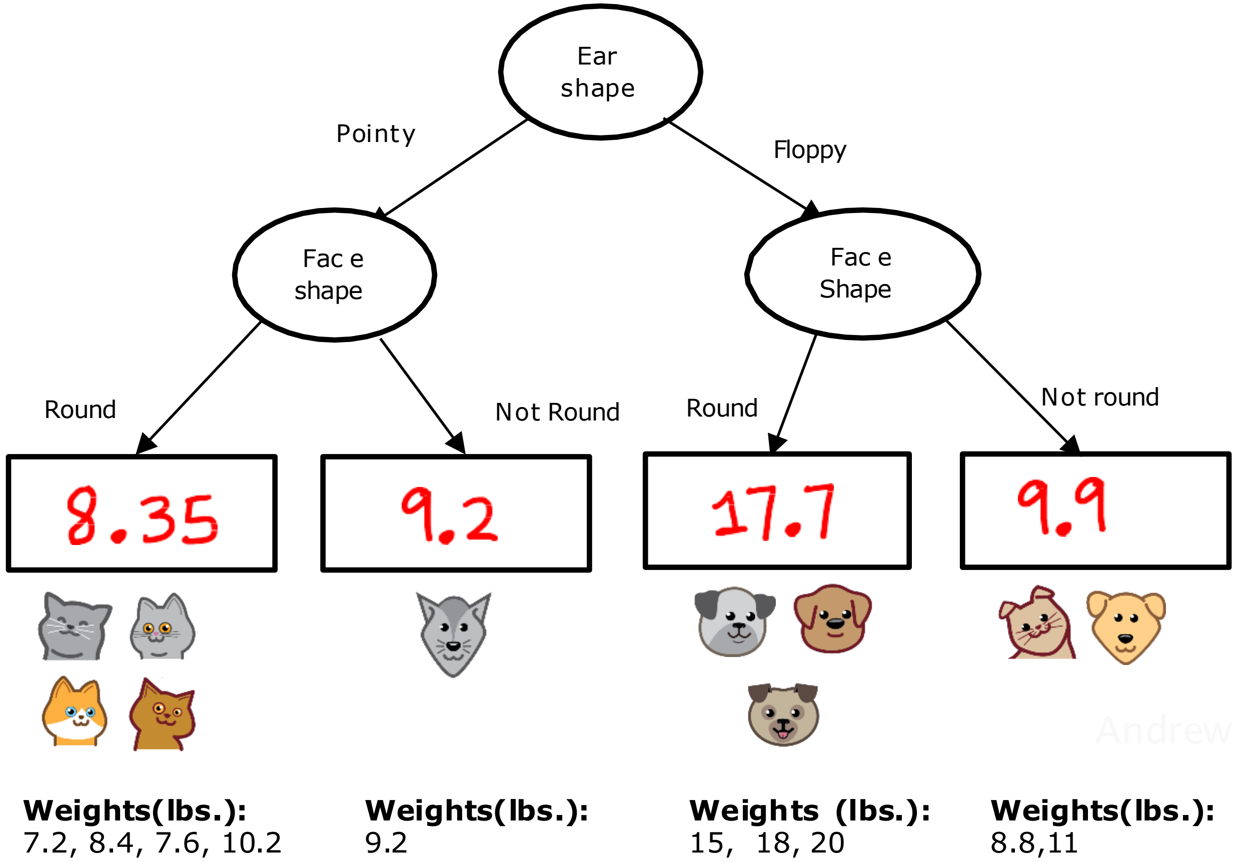


Split on mean or median of feature

Regression with Decision Trees: Predicting a number

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

Regression with Decision Trees



Prediction is the average weight of samples in leaf node