



Machine Learning 10











Kihyun Shin
DMSE, HBNU

TheMAQ

Decision tree



Cat classification example

	Ear shape (x_1)	Face shape (x_2)	Whiskers (x_3)	Cat (y)
	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

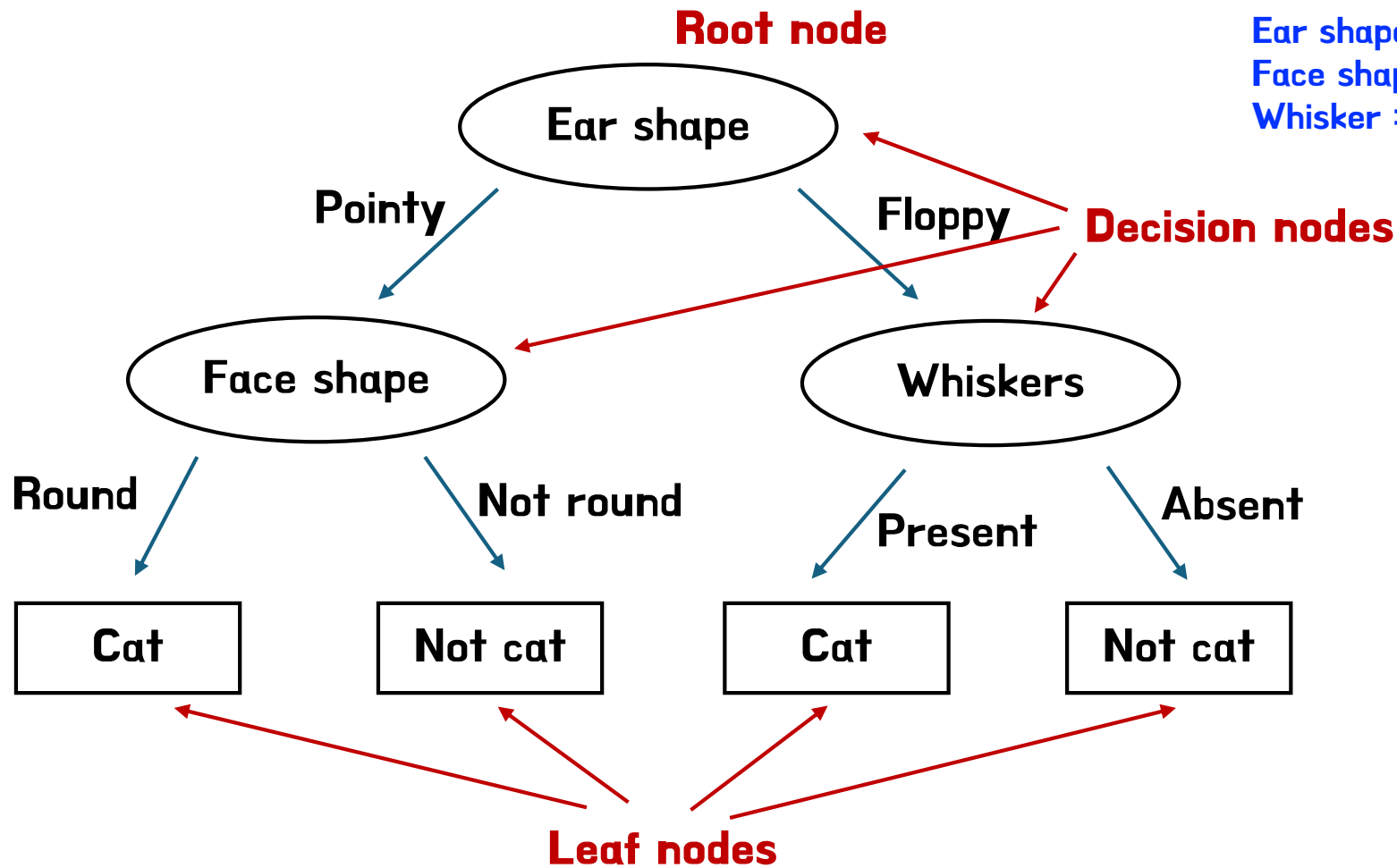
Currently, Features are binary

Decision tree

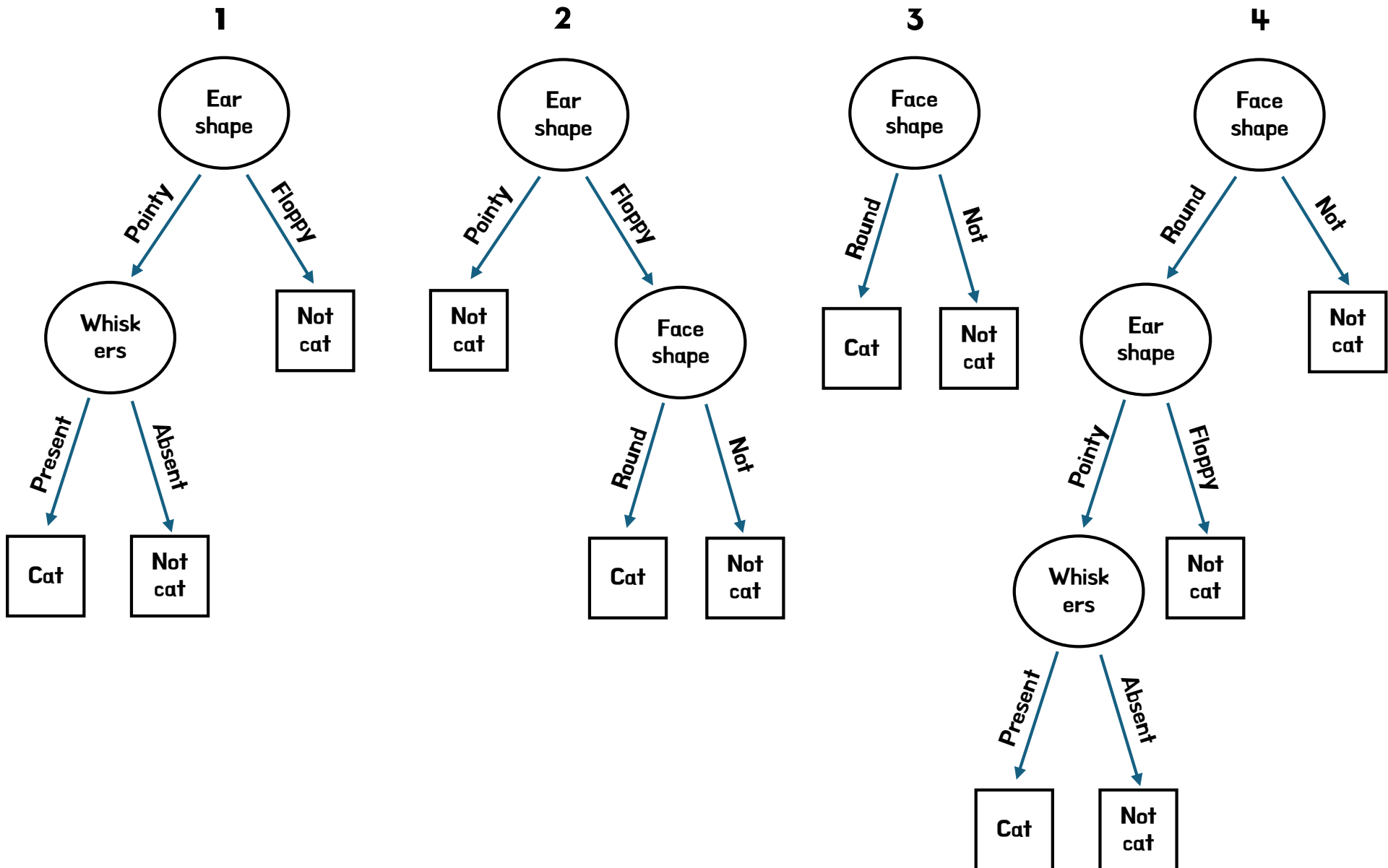
New test examples



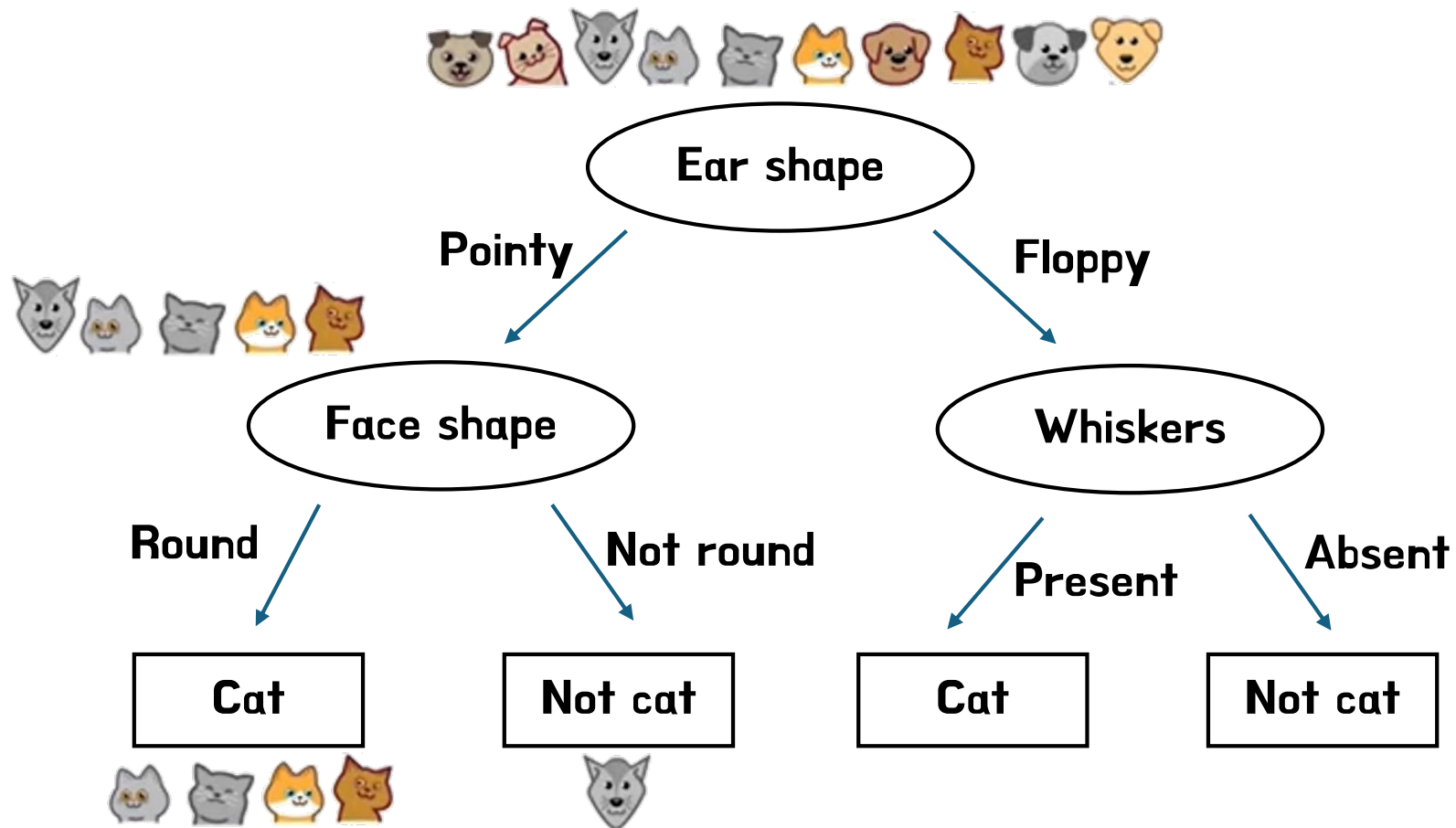
Ear shape : Pointy
Face shape : Round
Whisker : Present



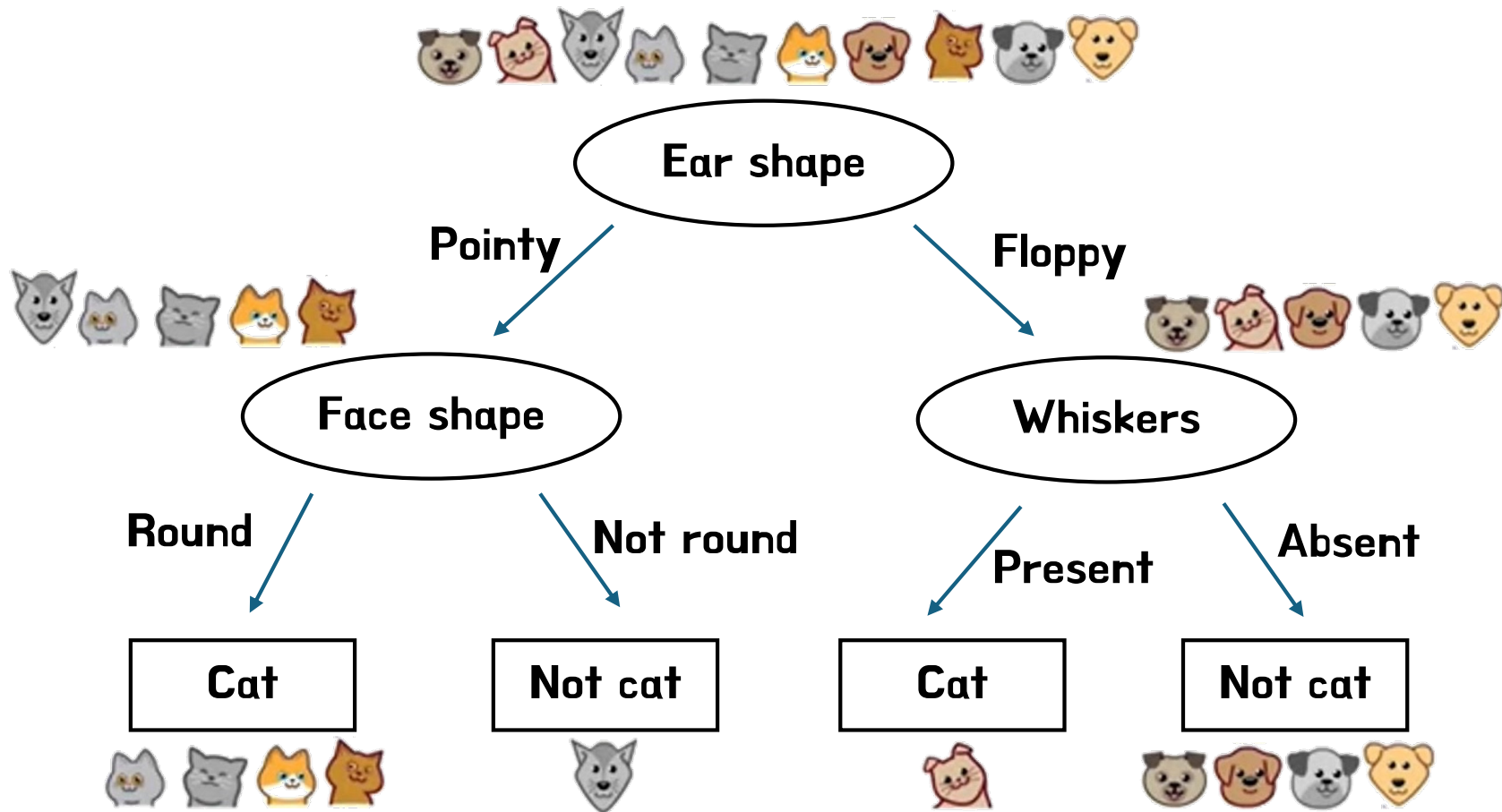
Decision tree



Decision tree learning



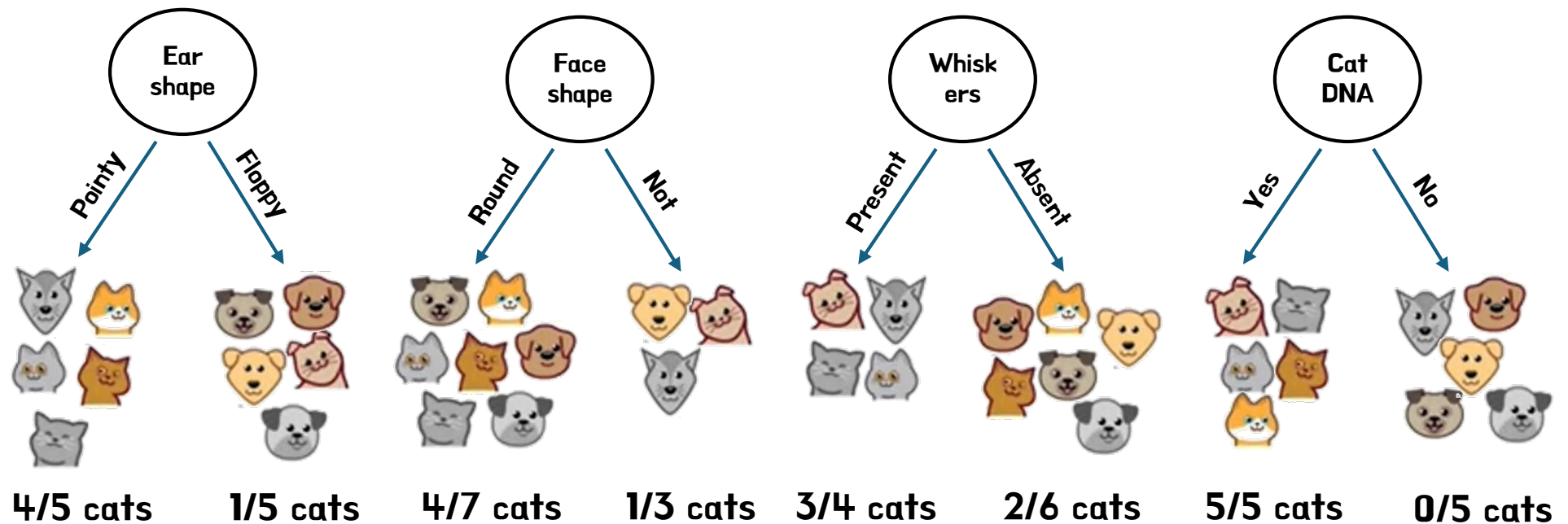
Decision tree learning



Decision tree learning

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

- Maximize purity (or minimize impurity)



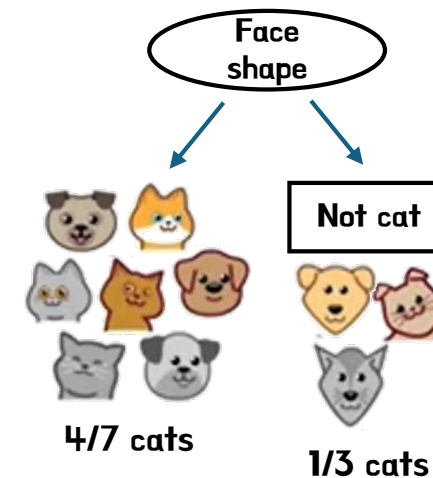
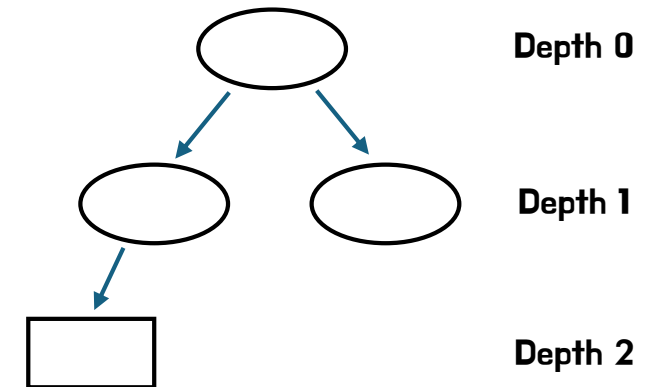
Decision tree learning

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

We can chose the depth

- Too deep: overfitting
- When do we have to stop ?

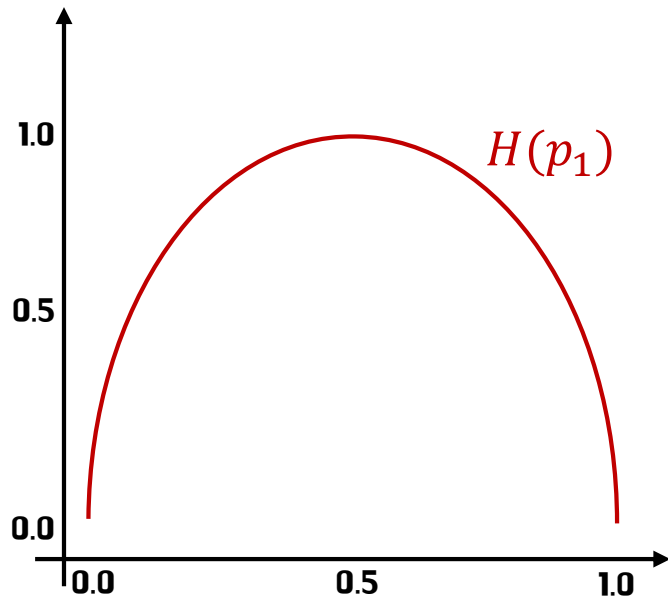
































Decision tree Learning



Entropy as a measure of impurity

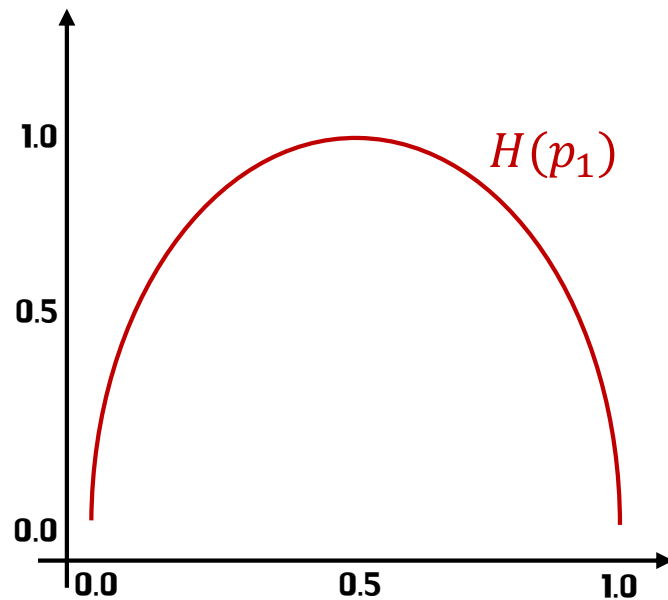
p_1 = fraction of examples that are cats



						$p_1 = 0$	$H(p_1) = 0$
Dog	Dog	Dog	Dog	Dog	Dog		
						$p_1 = 2/6$	$H(p_1) = 0.92$
Cat	Cat	Dog	Dog	Dog	Dog		
						$p_1 = 3/6$	$H(p_1) = 1$
Cat	Cat	Cat	Dog	Dog	Dog		
						$p_1 = 5/6$	$H(p_1) = 0.65$
Cat	Cat	Cat	Cat	Cat	Dog		
						$p_1 = 6/6$	$H(p_1) = 0$
Cat	Cat	Cat	Cat	Cat	Cat		

Entropy as a measure of impurity

p_1 = fraction of examples
that are cats



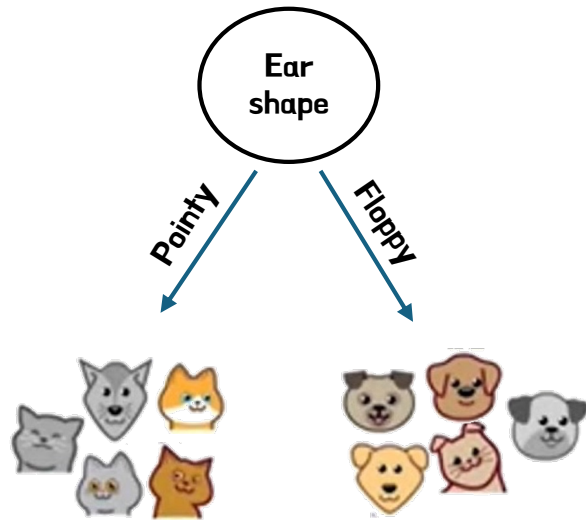
$p_0 = 1 - p_1$
(fraction of example that are not cats)

$$\begin{aligned} H(p_1) &= -p_1 \log_2(p_1) - p_0 \log_2(p_0) \\ &= -p_1 \log_2(p_1) - (1 - p_1) \log_2(1 - p_1) \end{aligned}$$

***Note:** $0 \log_2(0) = 0$

Choosing a split

Root node: $p_1 = \frac{5}{10} = 0.5$ $H(0.5) = 1$



$$p_1 = 4/5 = 0.8$$

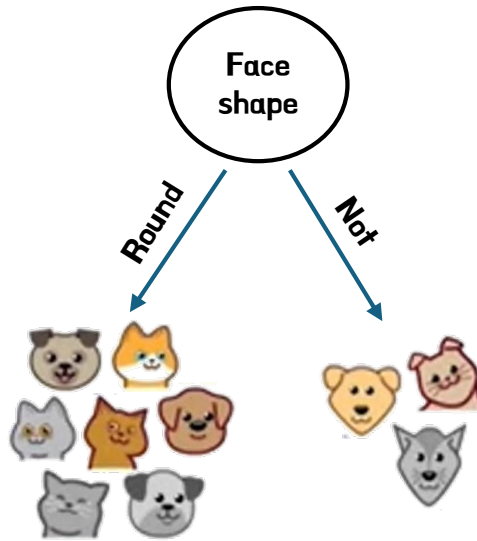
$$p_1 = 1/5 = 0.2$$

$$H(0.8) = 0.72$$

$$H(0.2) = 0.72$$

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

$$= 0.28$$



$$p_1 = 4/7 = 0.57$$

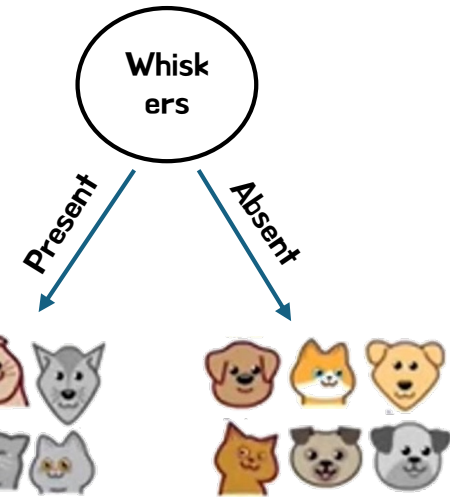
$$p_1 = 1/3 = 0.33$$

$$H(0.57) = 0.99$$

$$H(0.33) = 0.92$$

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

$$= 0.03$$



$$p_1 = 3/4 = 0.75$$

$$p_1 = 2/6 = 0.33$$

$$H(0.75) = 0.81$$

$$H(0.33) = 0.92$$

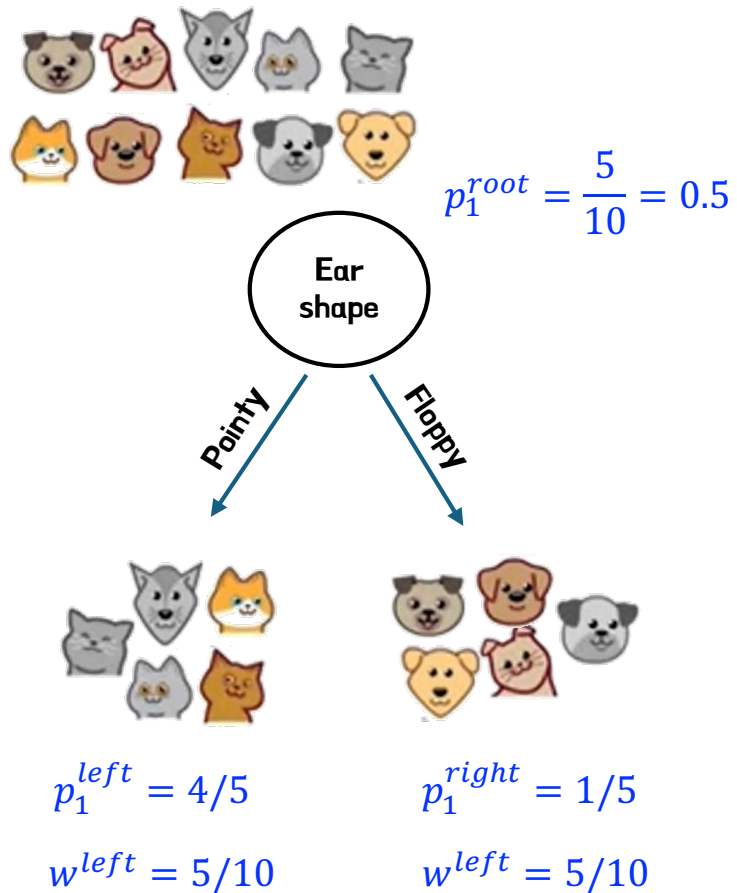
$$H(0.5) - \left(\frac{4}{10} H(0.75) + \frac{6}{10} H(0.33) \right)$$

$$= 0.12$$

Entropy reduction (weighted average) = Information gain



Information gain



**Information gain
(or entropy reduction)**











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

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**













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

Features are not binary (three possible values)

One hot encoding

	Ear shape (x_1)	Pointy ears	Floppy ears	Oval ears	Face shape (x_2)	Whiskers (x_3)	Cat (y)
	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



Features are not binary (three possible values)

One hot encoding











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













One hot encoding and neural networks

	Ear-shape (x_1)	Pointy ears	Floppy ears	Oval ears	Face shape (x_2)	Whiskers (x_3)	Cat (y)
	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

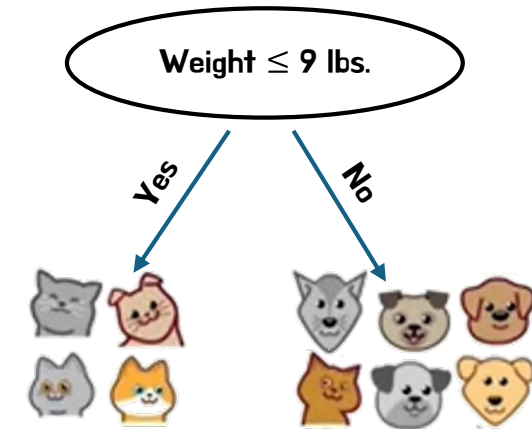
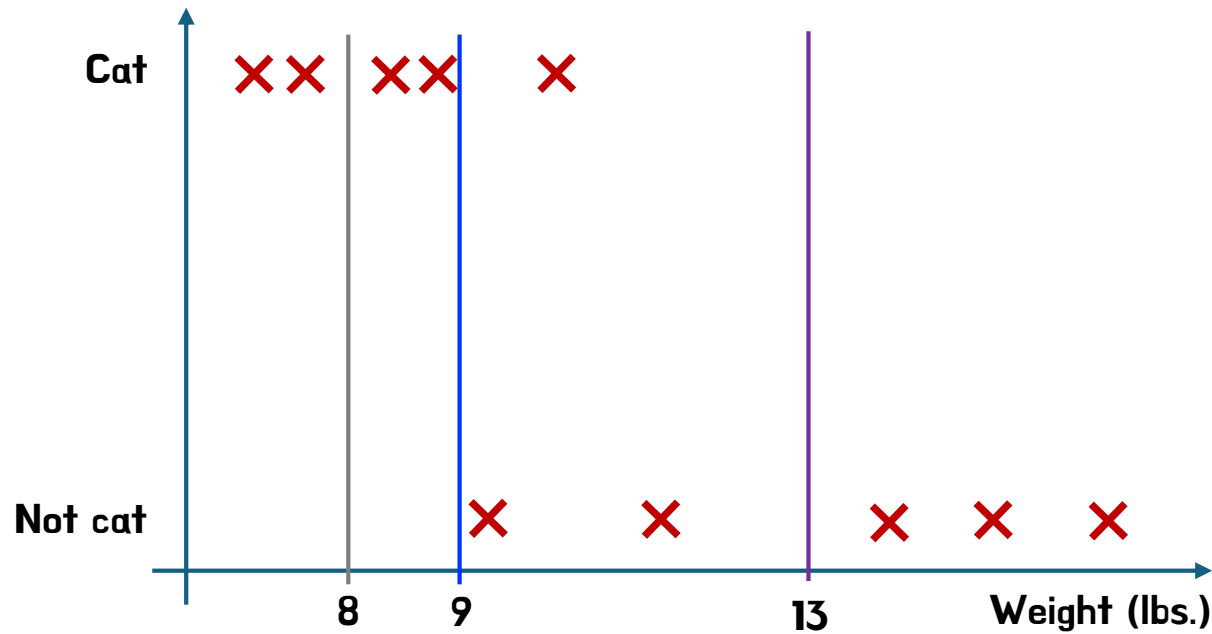
One hot encoding and neural networks

	Pointy ears	Floppy ears	Oval ears	Face shape (x_2)	Whiskers (x_3)	Cat (y)
	1	0	0	1	1	1
	0	0	1	0	1	1
	0	0	1	1	0	0
	1	0	0	0	1	0
	0	0	1	1	1	1
	1	0	0	1	0	1
	0	1	0	0	0	0
	0	0	1	1	0	1
	0	1	0	1	0	0
	0	1	0	1	0	0

Continuous features

	Ear shape (x_1)	Face shape (x_2)	Whiskers (x_3)	Weight (lbs.)	Cat (y)
	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













$$H(0.5) - \left(\frac{2}{10} H\left(\frac{2}{2}\right) + \frac{8}{10} H\left(\frac{3}{8}\right) \right) = 0.24$$

$$H(0.5) - \left(\frac{4}{10} H\left(\frac{4}{4}\right) + \frac{6}{10} H\left(\frac{1}{6}\right) \right) = 0.61$$

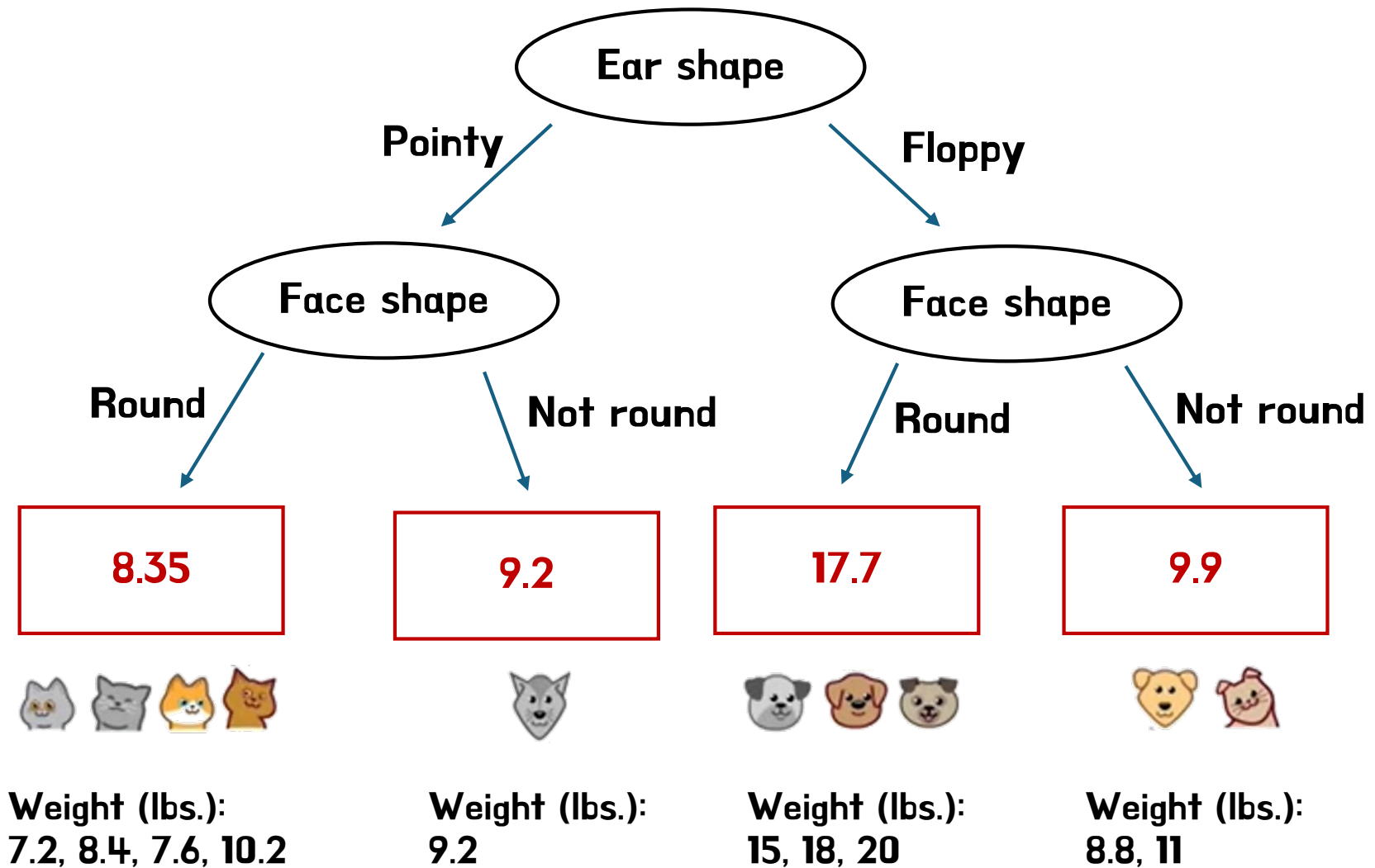
$$H(0.5) - \left(\frac{7}{10} H\left(\frac{5}{7}\right) + \frac{3}{10} H\left(\frac{0}{3}\right) \right) = 0.40$$

Regression with decision trees

	Ear shape (x_1)	Face shape (x_2)	Whiskers (x_3)	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

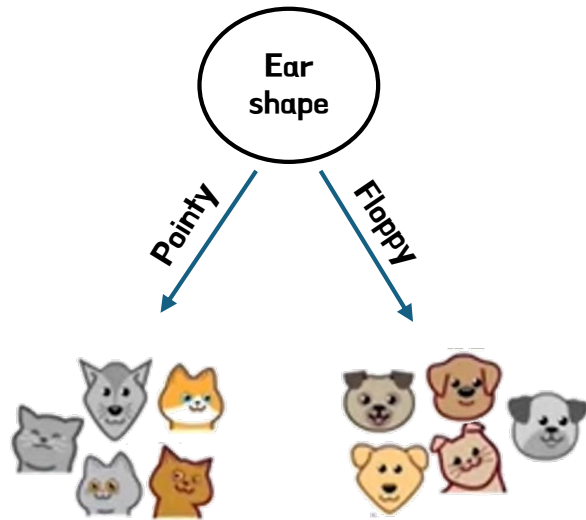
x
 y

Regression with decision trees



Choosing a split

Variance at root node: 20.51



Weights: 7.2, 9.2, 8.4, 7.6, 10.2

Weights: 8.8, 15, 11, 18, 20

Variance: 1.47

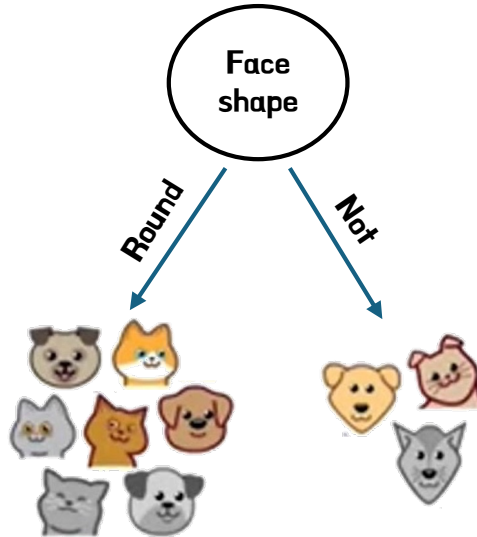
Variance: 21.87

$$w^{left} = 5/10$$

$$w^{right} = 5/10$$

$$20.51 - \left(\frac{5}{10} * 1.47 + \frac{5}{10} * 21.87 \right)$$

$$= 8.84$$



Weights: 7.2, 15, 8.4, 7.6, 10.2, 18, 20

Weights: 8.8, 9.2, 11

Variance: 27.80

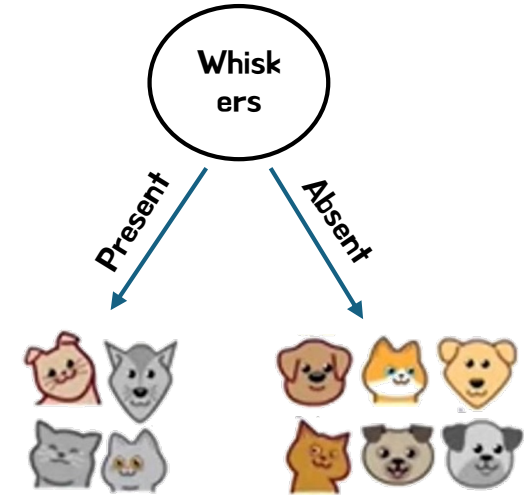
Variance: 1.37

$$w^{left} = 7/10$$

$$w^{right} = 3/10$$

$$20.51 - \left(\frac{7}{10} * 27.80 + \frac{3}{10} * 1.37 \right)$$

$$= 0.64$$



Weights: 7.2, 8.8, 9.2, 8.4

Weights: 15, 7.6, 11, 10.2, 18, 20

Variance: 0.75

Variance: 23.32

$$w^{left} = 4/10$$

$$w^{right} = 6/10$$

$$20.51 - \left(\frac{4}{10} * 0.75 + \frac{6}{10} * 23.32 \right)$$

$$= 6.22$$

Variance reduction = Information gain

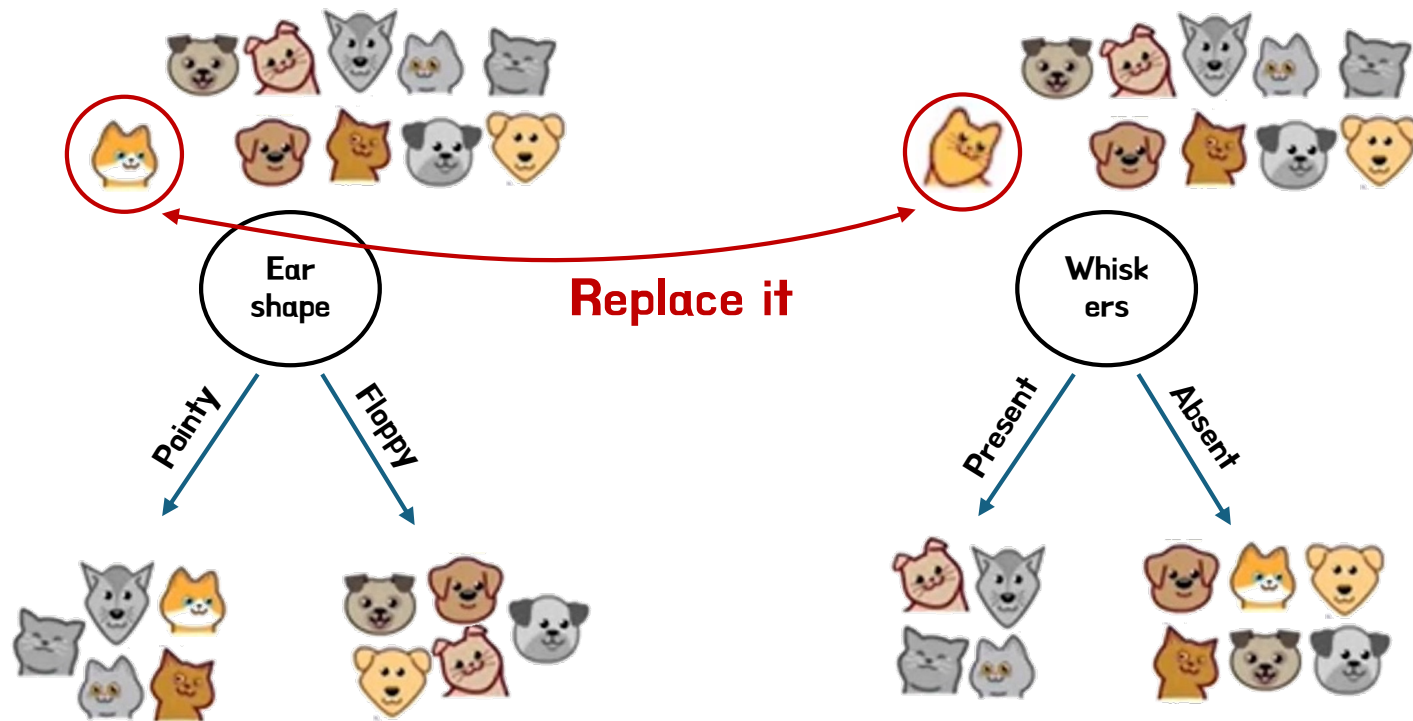


Tree ensembles



Tree ensemble

Highly sensitive to small changes of the data



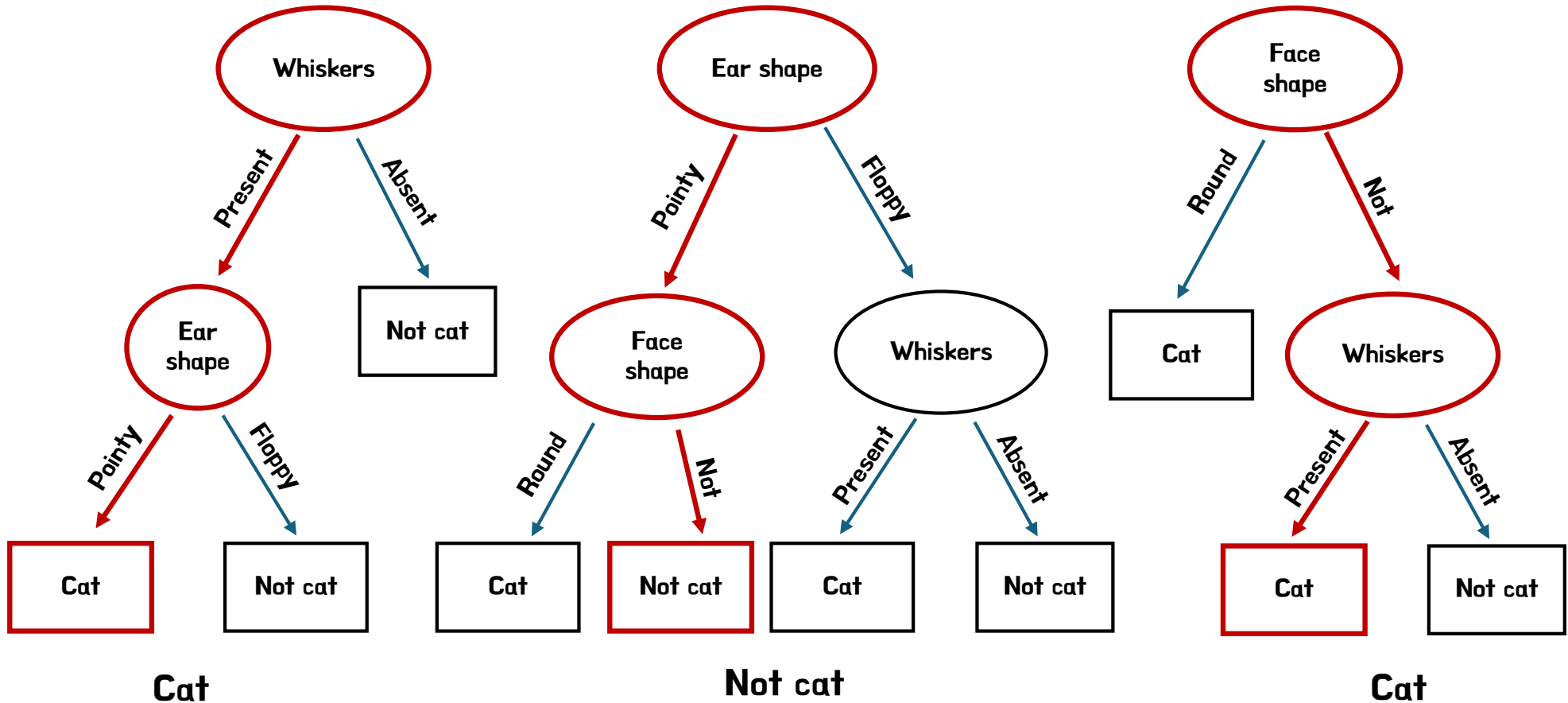
Single data change \rightarrow change of optimal tree

Using collections of tree will be better

Tree ensemble



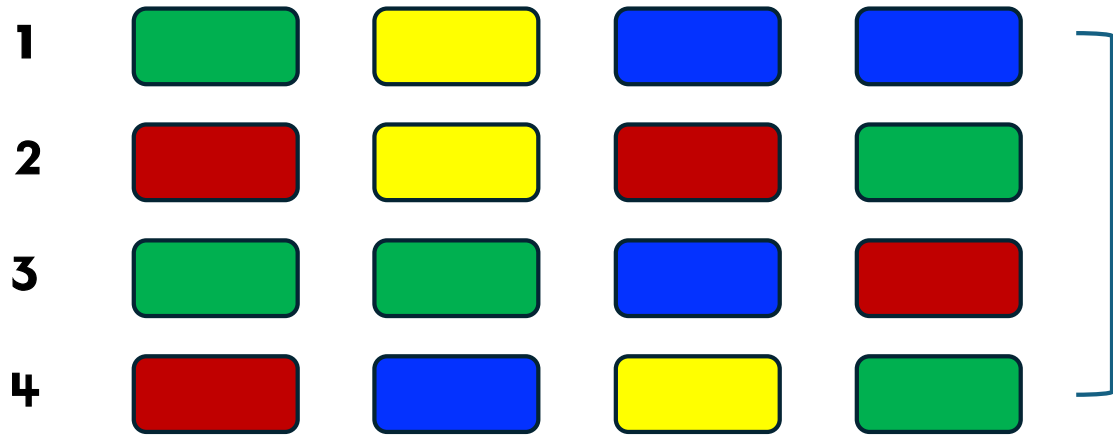
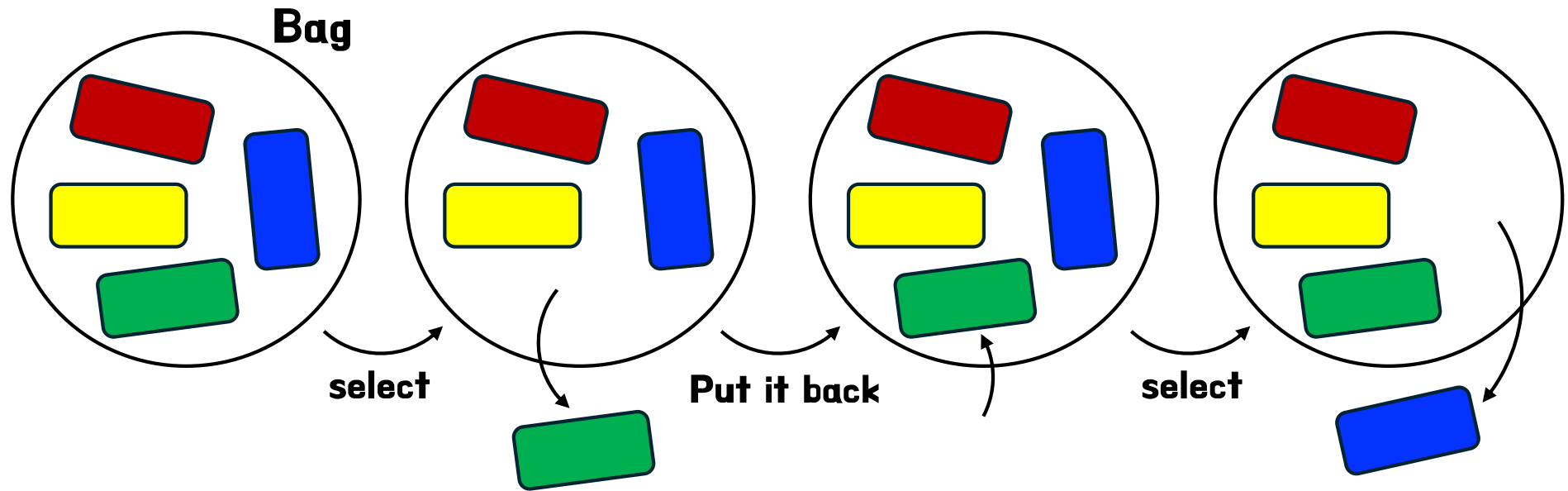
New test example
Ear shape: Pointy
Face shape : not round
Whisker : present



Prediction based on voting



Sampling with replacement













Generation of different training set

Sampling with replacement

Bag



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

Random forest: generating a tree sample

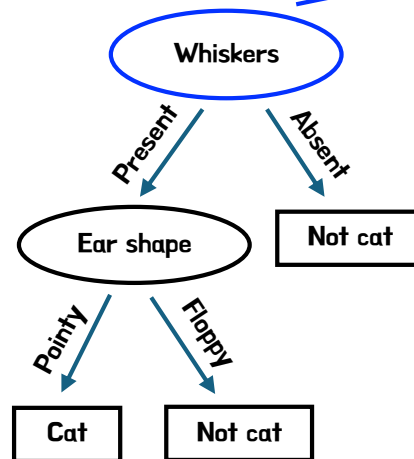
Given training set of size m

For $b = 1$ to B (*usually 100*)

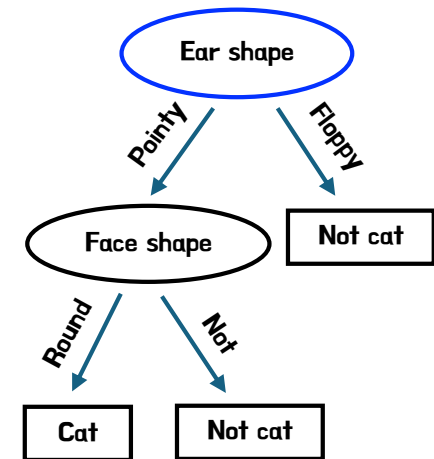
- Use sampling with replacement to create a new training set of size m
- Train a decision tree on the new dataset

Usually, root node is identical

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



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



Random forest: random feature choice

At each node, when choosing a feature to use to split, if n features are available, pick a random subset of $k < n$ features and allow the algorithm to only choose from that subset of features.

$$k = \sqrt{n}$$

Boosted trees intuition

Given training set of size m

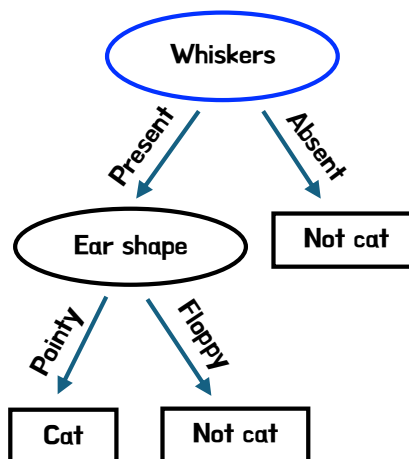
For $b = 1$ to B

- Use sampling with replacement to create a new training set of size m

But instead of picking from all examples with equal ($1/m$) probability, make it more likely to pick misclassified examples from previous trained trees

- Train a decision tree on the new dataset

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



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



XGBoost (eXtreme Gradient Boosting)

- **Open source implementation of boosted trees**
- **Fast efficient implementation**
- **Good choice of default splitting criteria and criteria for when to stop splitting**
- **Built in regularization to prevent overfitting**
- **Highly competitive algorithm for machine learning competitions (e. g. Kaggle competitions)**

Deep learning as well



Using XGBoost

Classification

```
from xgboost import XGBClassifier
```

```
model = XGBClassifier()
```

```
model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)
```

Regression

```
from xgboost import XGBRegressor
```

```
model = XGBRegressor()
```

```
model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)
```

Simple to use



Decision trees vs Neural networks

Decision trees and tree ensembles

- Works well on tabular (structured) data
- Not recommended for unstructured data (images, audio, text)
- Fast
- Small decision trees may be human interpretable

Neural networks

- Works well on all types of data, including tabular (structured) and unstructured data
- May be slower than a decision tree
- Works with transfer learning
- When building a system of multiple models working together, it might be easier to string multiple neural networks

