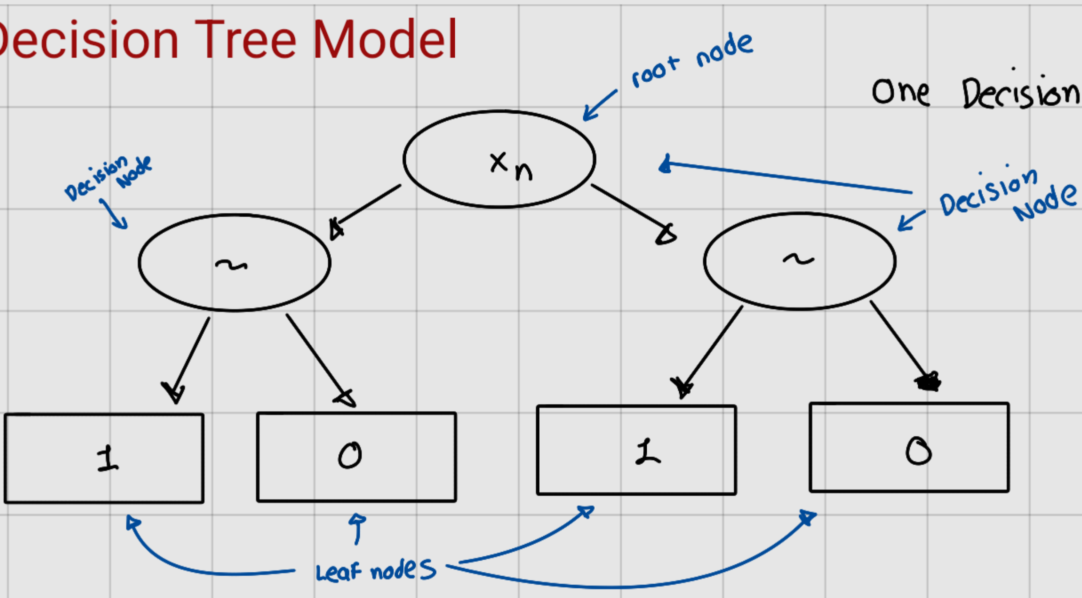
# Decision Tree Model

One Decision tree for each feature that we choose




## Decision tree learning

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

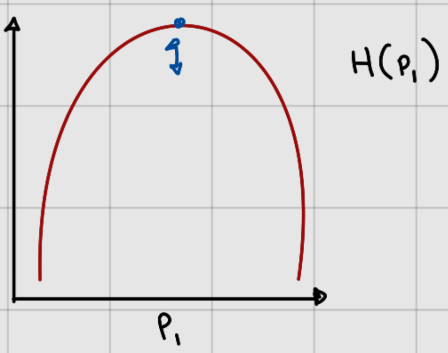
- Maximize purity (or minimize impurity)

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 mesaure of impurity

$p_1$ = fractions of examples that are cats




$$p_0 = 1 - p_1$$

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

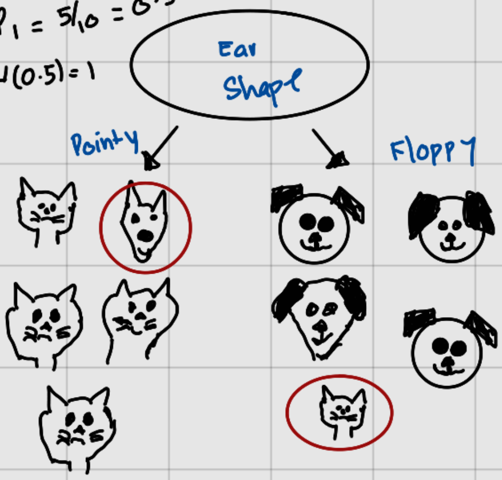
$$= -p_1 \log_2(p_1) - (1-p_1) \log_2(1-p_1)$$

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

# Choosing a split

$p_1 = 5/10 = 0.5$

$H(0.5) = 1$




$p_1 = 4/5 = 0.8$ $\qquad$ $p_1 = 1/5 = 0.2$

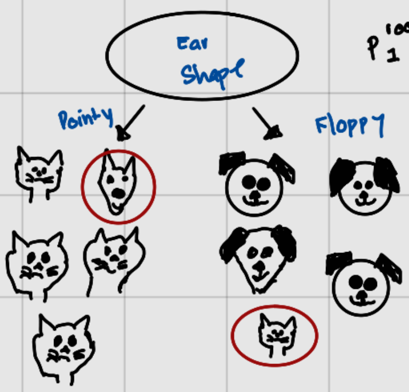
$H(0.8) = 0.72$ $\qquad$ $H(0.2) = 0.72$

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

$$= 0.28$$

# Information Gain

$p_1^{root} = 5/10 = 0.5$




$p_1^{left} = 4/5$ $\qquad$ $p_1^{right} = 1/5$

$w^{left} = 5/10$ $\qquad$ $w^{right} = 5/10$

Information gain formula

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

# Random Forest Algorithm

Given training set of size m

For $b = 1$ to $B$ :

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

Train a decision tree on the new dataset

### Randomizing the 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

$$k = \sqrt{n}$$

# XGBoost

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 previously trained trees.

Train a decision tree on the new dataset

### XGBoost (eXtreme Gradient Boosting)

- Open source implementation of boosted trees
- Fast efficient implementation
- Good choice of default splitting criteria for when to stop splitting
- Built in regularization to prevent overfitting
- Highly competitive algorith for ML competitions

# XGBoost implementation

## Classification

```python
from xgboost import XGBClassifier

model = XGBClassifier()

model.fit (x_train, y_train)

y_pred = model.predict(x_test)
```

## Regression

```python
from xgboost import XGBRegressor

model = XGBRegressor()

model.fit (x_train, y_train)

y_pred = model.predict(x_test)
```