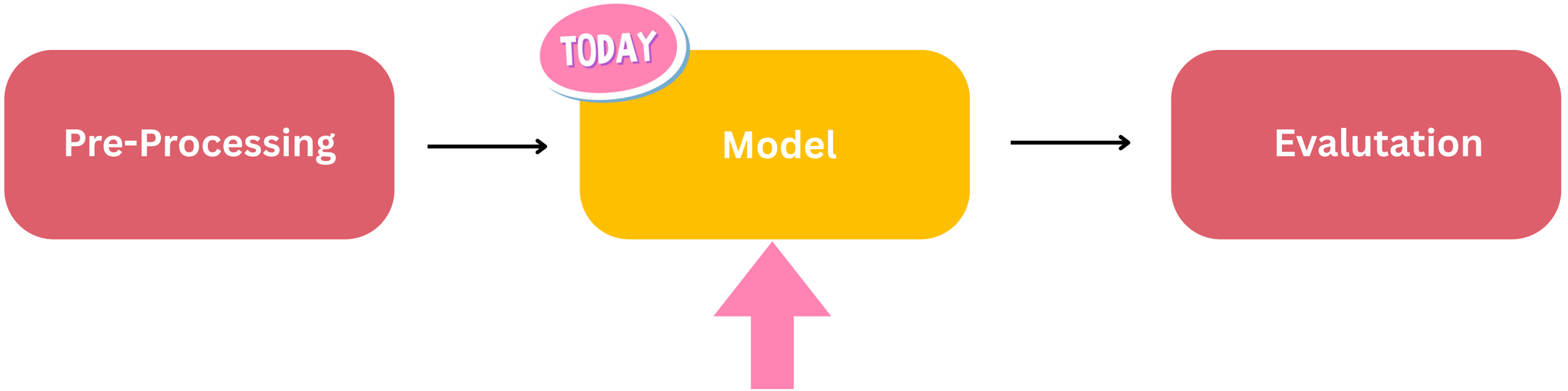
Data Science and Machine Learning

For New Enthusiasts

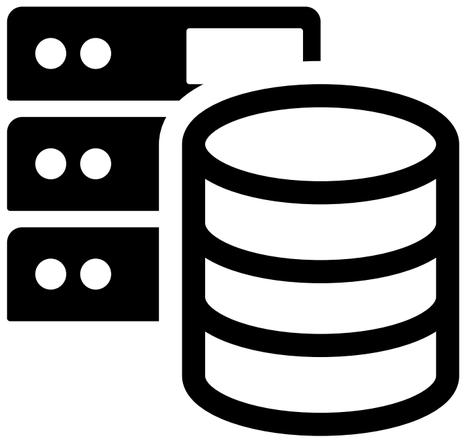
Rafael M Mamede (rafael.c.maia@inesctec.pt)

REPO LINK



- K-Nearest Neighbours
- Naive Bayes
- Decision Trees
- Artificial Neural Networks

What is Machine Learning?



MODEL REPRESENTATION

f(x;Θ): Predictors → Target

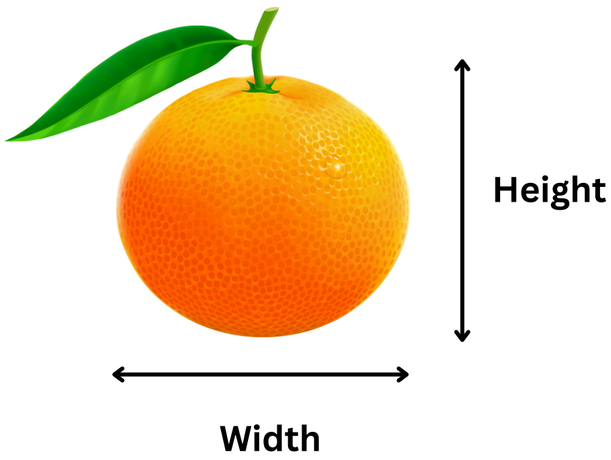
Goal

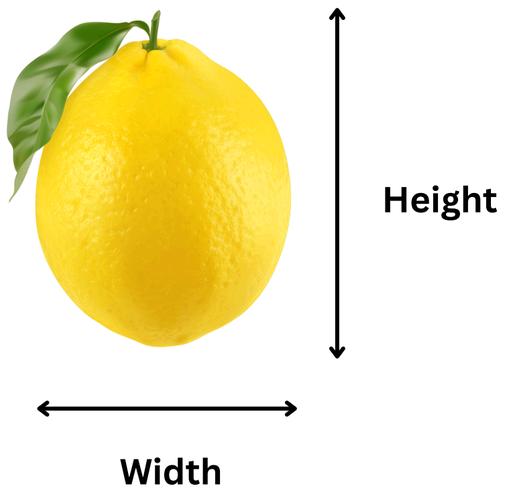
Find Best Θ

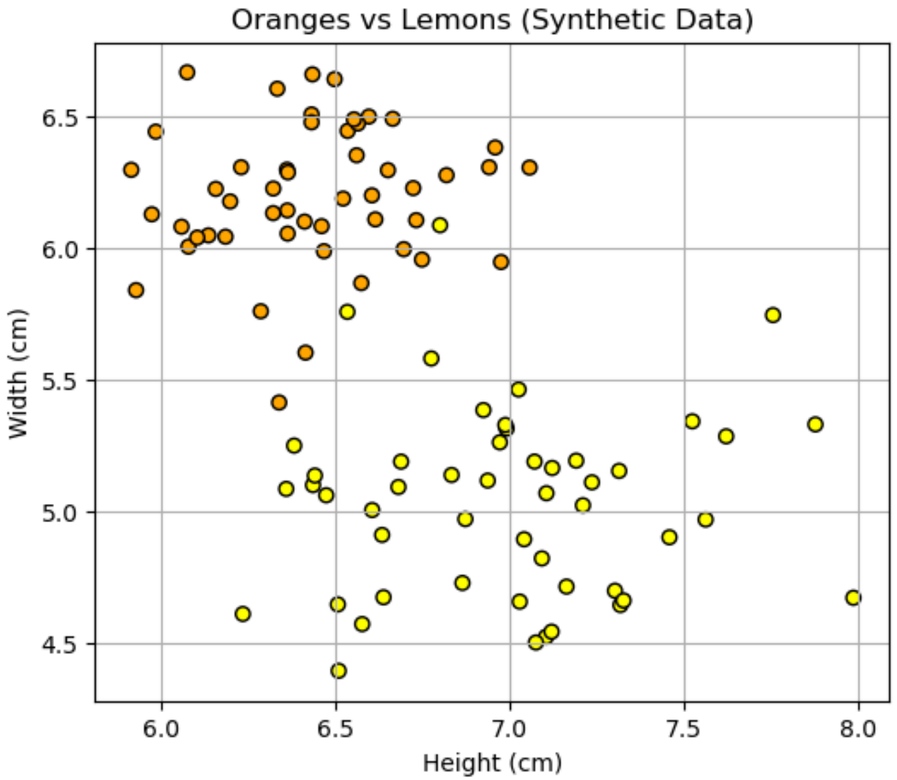
→ **Define optimizing function (either a Loss or a Goal)**

→ **Optimization Procedure**

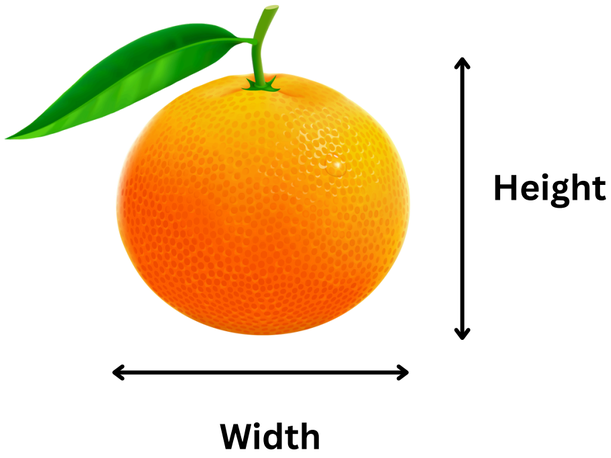
Practical use-case

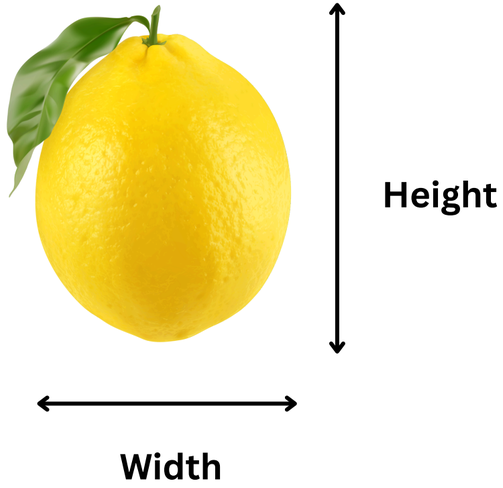


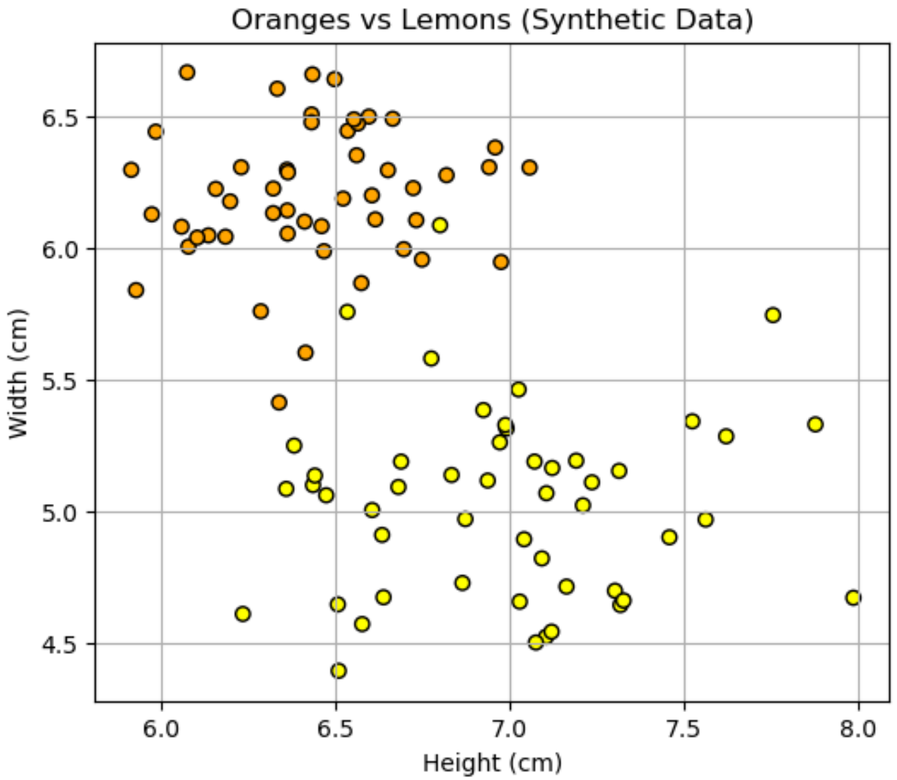




Practical use-case



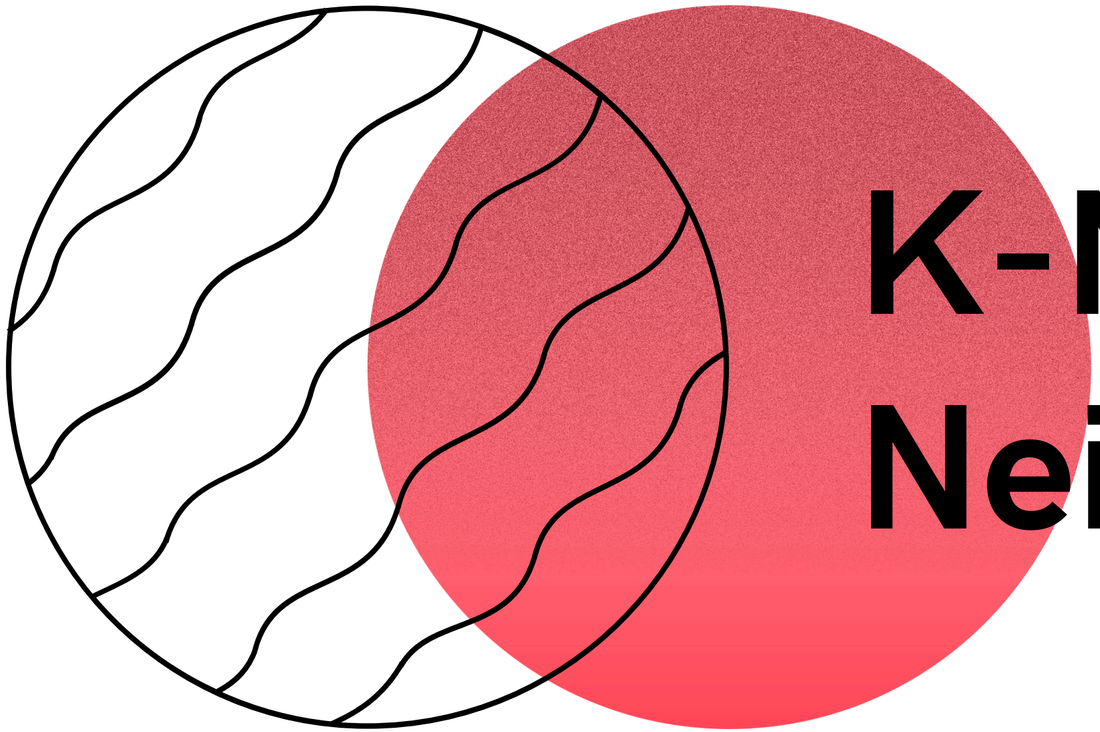




f(width,height): R^2 → {0,1},

0 - Orange

1 - Lemon



K-Nearest Neighbours

K-NN

- **Non-parametric: makes no assumptions about data distribution**
- **Lazy learner: defers computation to prediction time**
- **Distance-based: sensitive to feature scaling and irrelevant features**

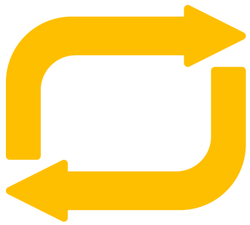
One of the simplest of all machine learning classifiers

K-NN

Idea:

–**Similar examples have similar label.**

–Classify new examples like similar training examples.



Algorithm:

–Given some new example x for which we need to predict its class y

–Find the k most similar training examples

–Classify x “like” these most similar examples (i.e. mode of target variable)

K-NN

k most similar instances:

K Closest according to some distance measure

Distance Functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

FROM: https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/

K-NN

k most similar instances:

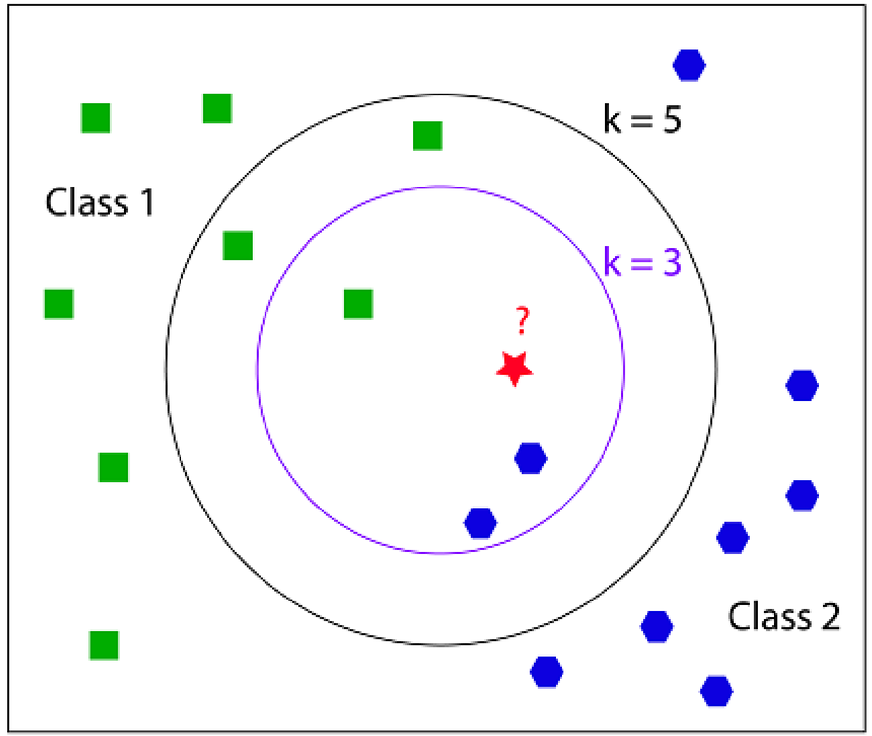
K Closest according to some distance measure

Distance Functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

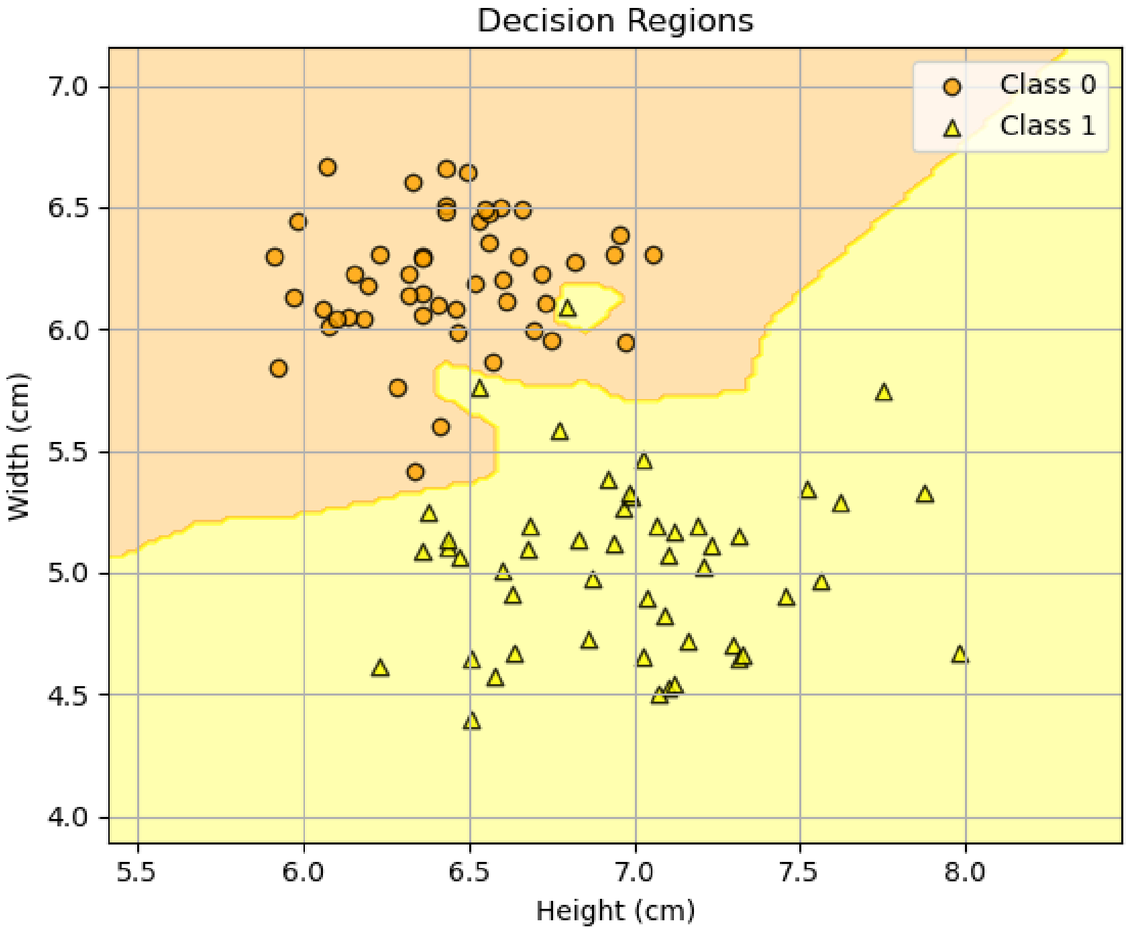


FROM: https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/

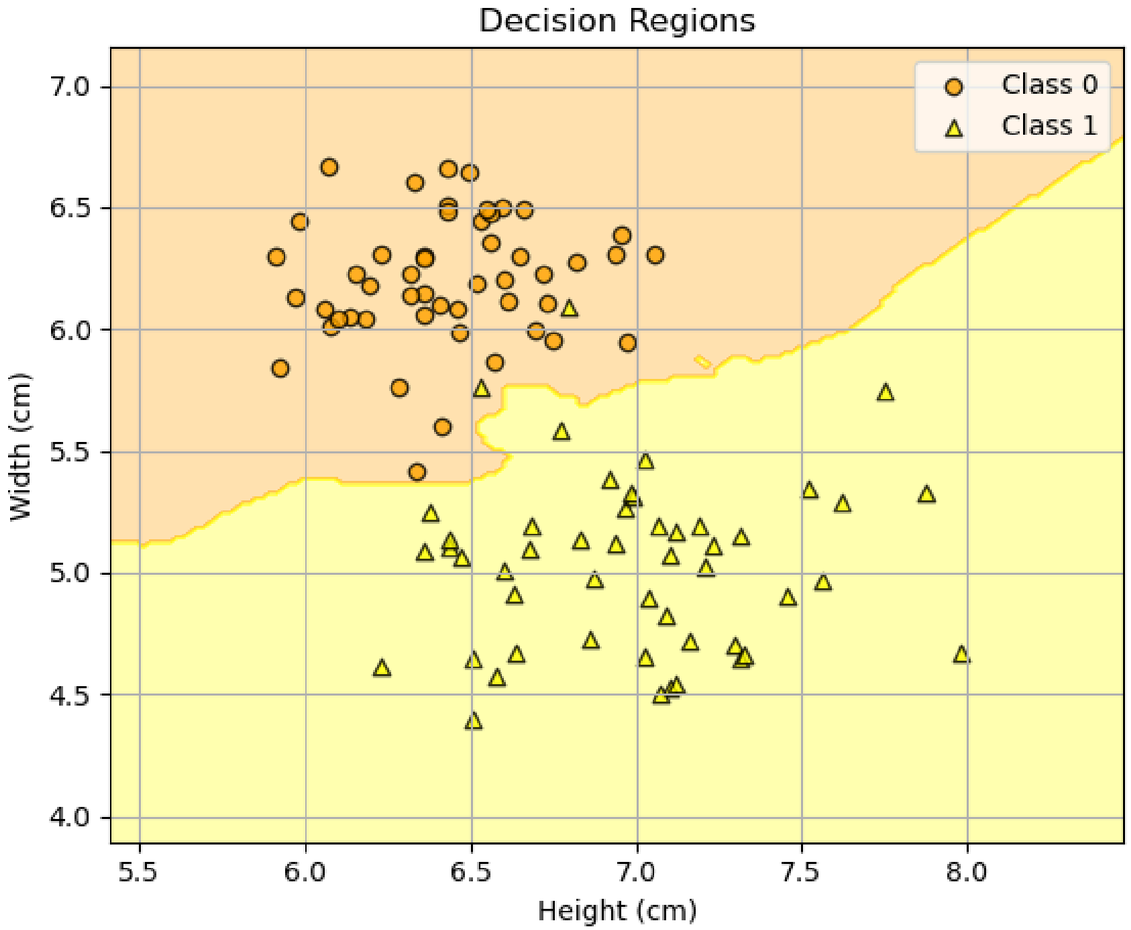
FROM: https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/

K-NN

1-NN


Decision Regions
Class 0
Class 1
Width (cm)
Height (cm)
7.0
6.5
6.0
5.5
5.0
4.5
4.0
5.5
6.0
6.5
7.0
7.5
8.0

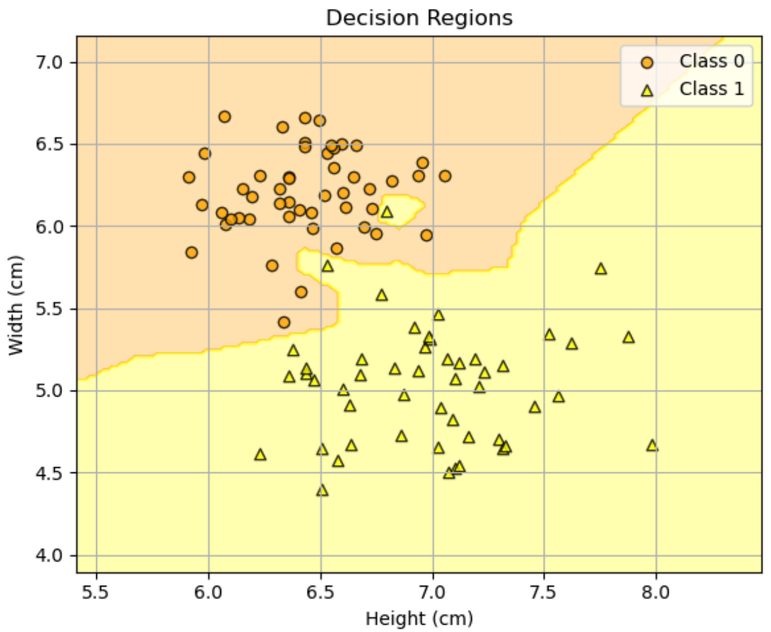
3-NN

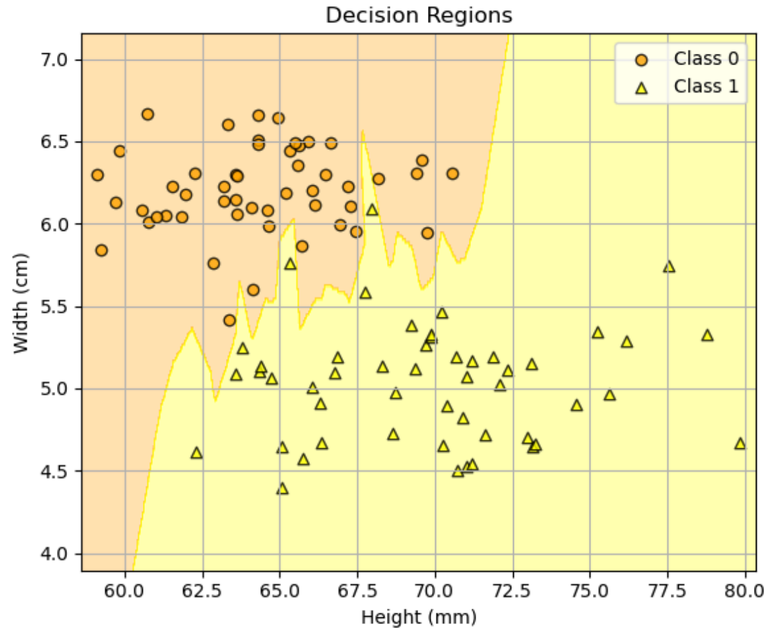

Decision Regions
Class 0
Class 1
Width (cm)
Height (cm)
7.0
6.5
6.0
5.5
5.0
4.5
4.0
5.5
6.0
6.5
7.0
7.5
8.0

K-NN

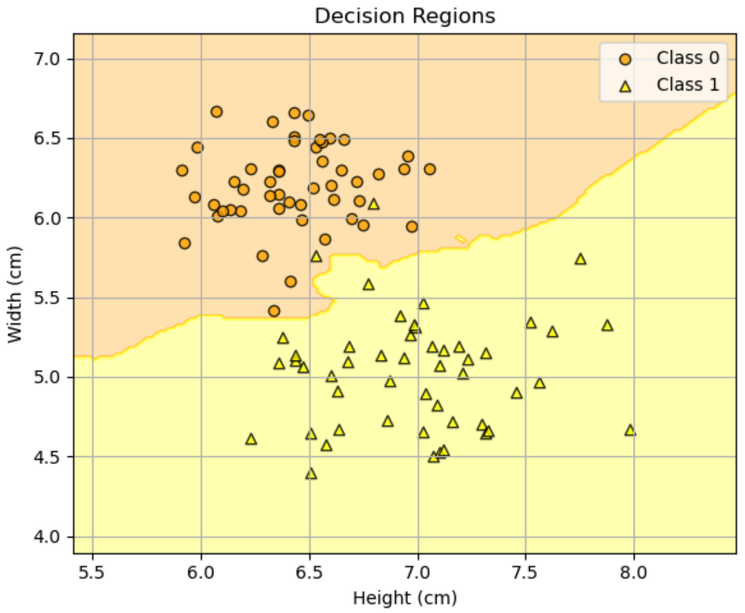
Importance of Scale

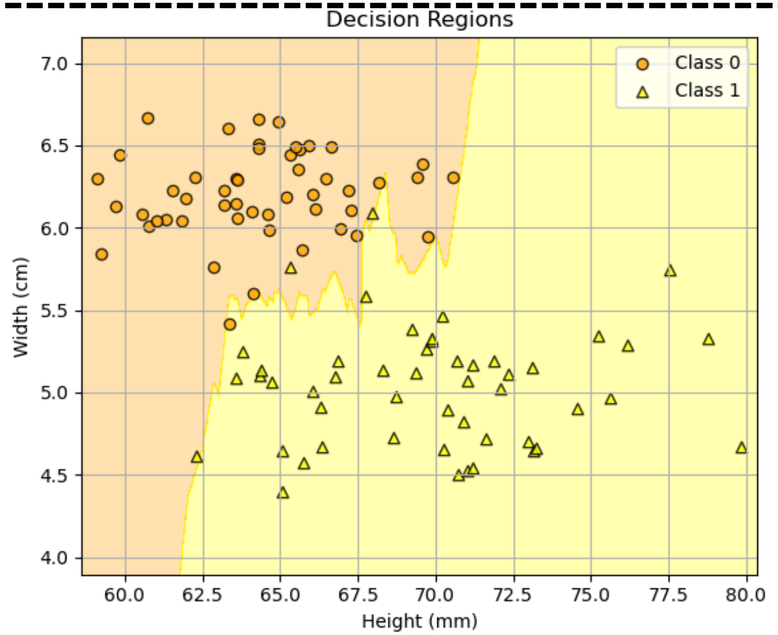
1-NN

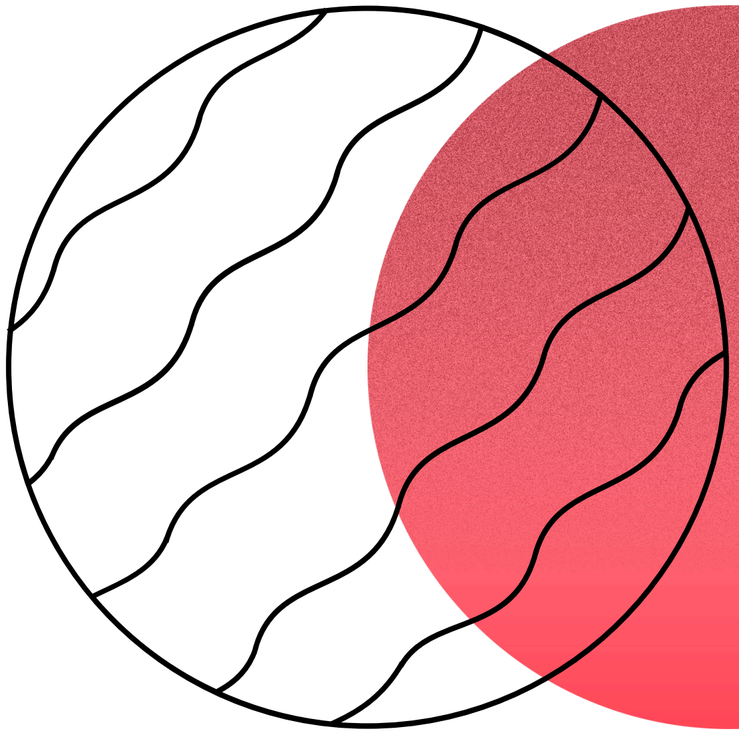




3-NN







Naive Bayes

Naive Bayes

- **Probabilistic**

  For a pair of (height,width) what is more likely?
  Lemon or Orange?

$$P\left(y=1|\left(\text{height, width}\right)\right) \quad (> \text{ or } < ?) \quad P\left(y=0|\left(\text{height, width}\right)\right)$$

- **Naive: Assumes predictor independance**

Naive Bayes

Idea:

Likelihood

Class Prior Probability

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Posterior Probability

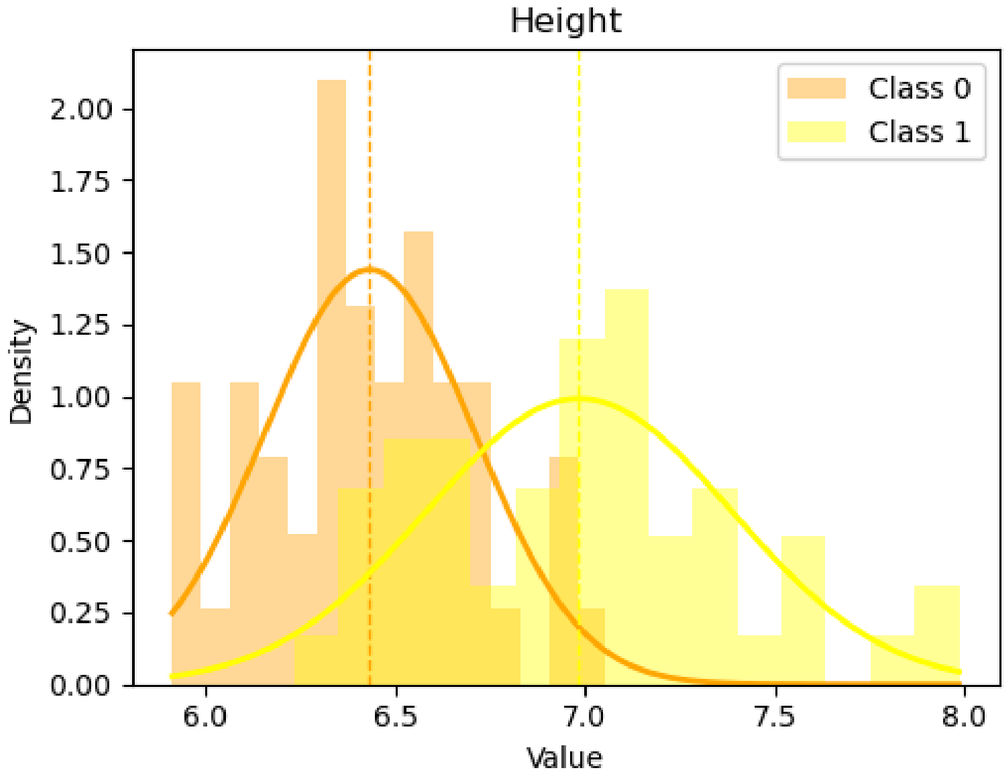
Predictor Prior Probability

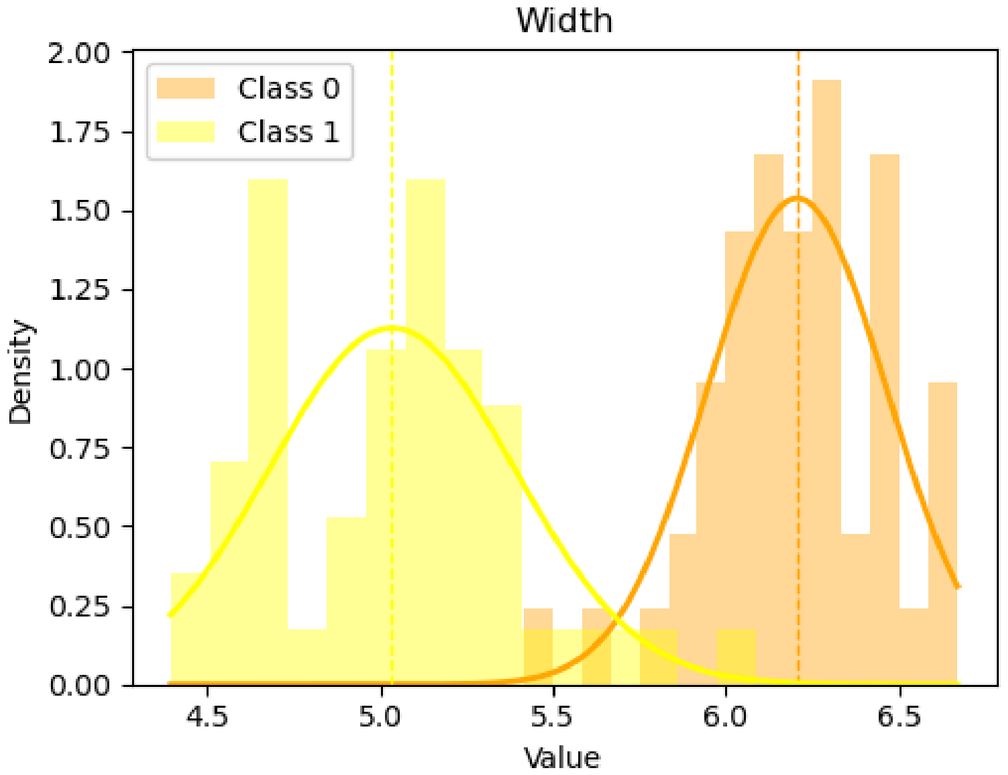
$$P(c|x) \propto P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Assume a distribution for P(x|c), and use parametric statistics to find distribution parameters

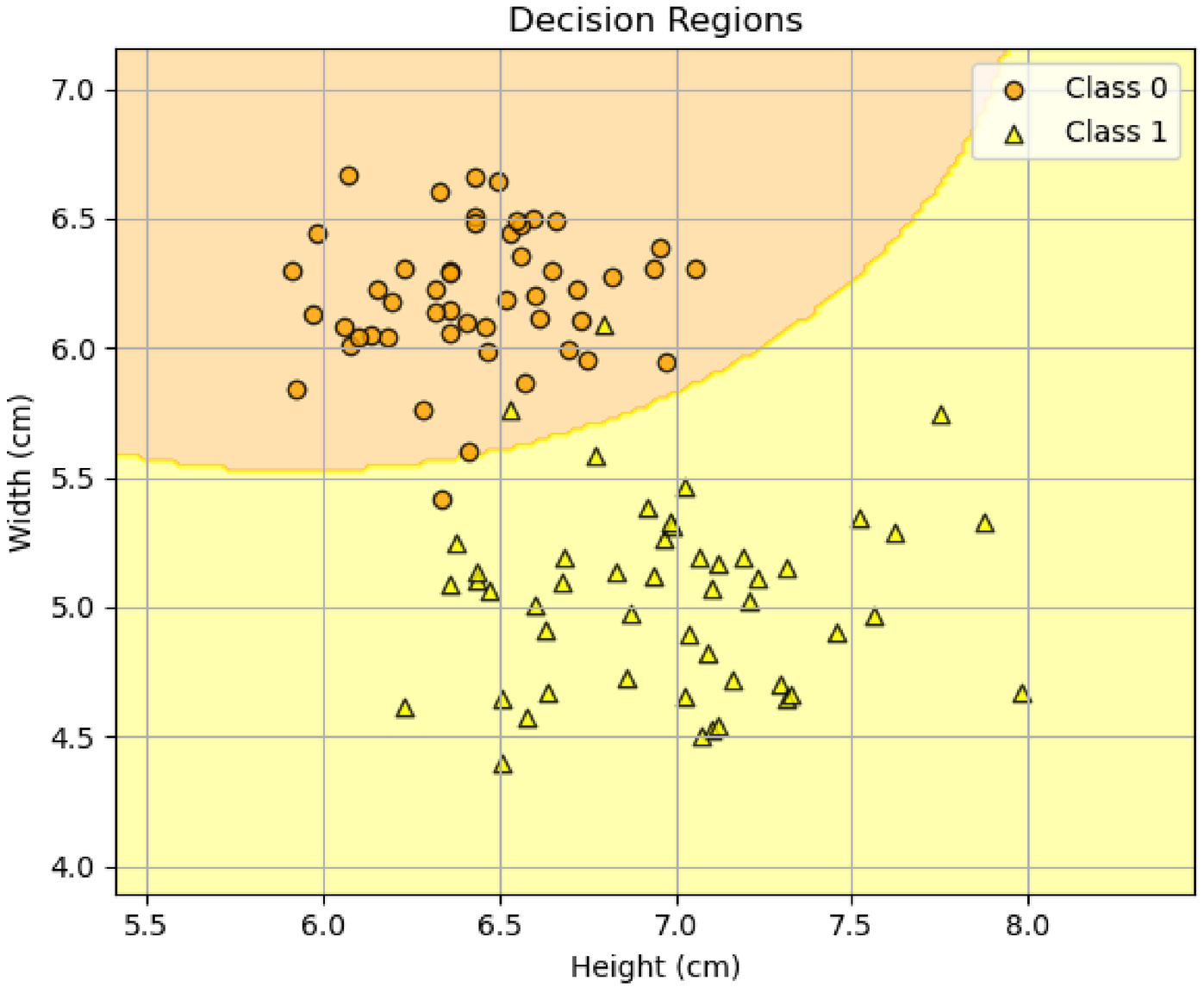
Naive Bayes

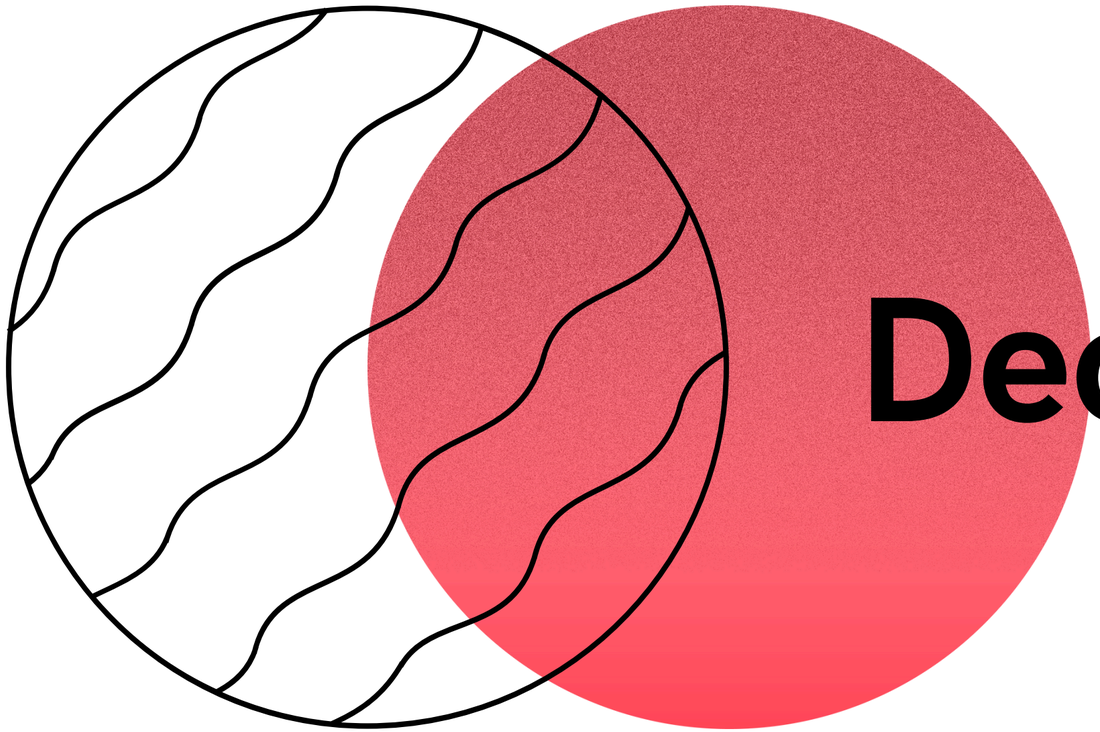
$$\mathcal{N}\left(\mu, \sigma^2\right) \qquad \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$





Naive Bayes


Decision Regions
Class 0
Class 1
Width (cm)
Height (cm)
7.0
6.5
6.0
5.5
5.0
4.5
4.0
5.5
6.0
6.5
7.0
7.5
8.0



Decision Trees

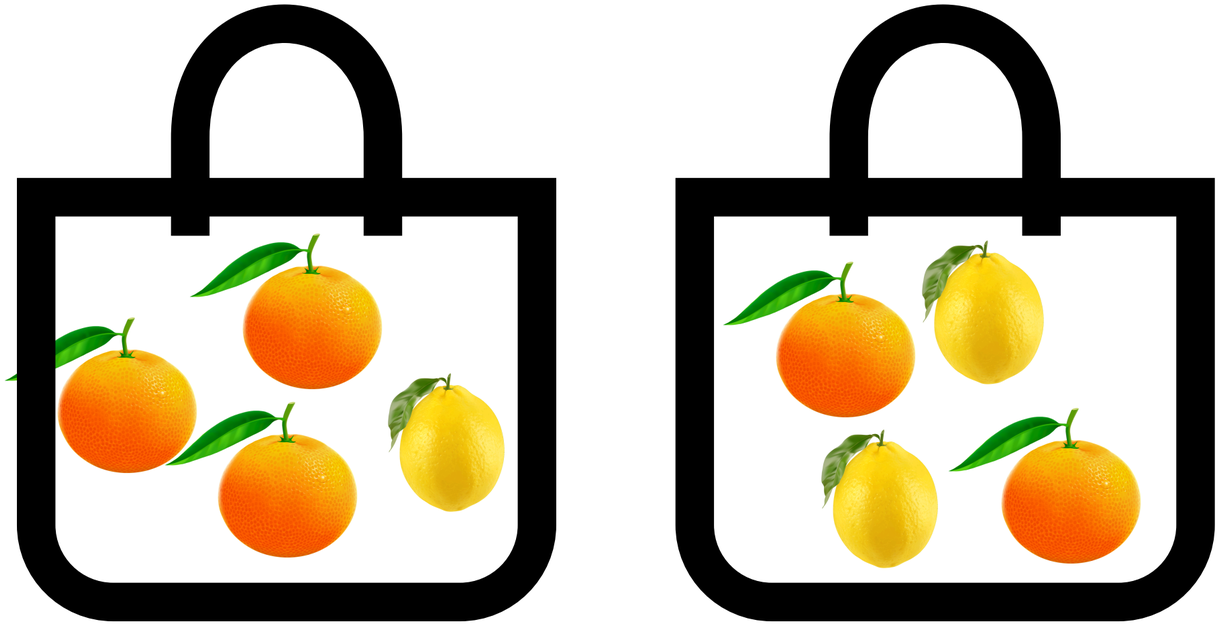
Decision Trees

- Mimics human decision-making by learning simple if-then rules from the data

- Builds a tree-like model of decisions:
  - Internal nodes: feature-based decisions
  - Branches: outcomes of the decision
  - Leaves: final predictions
- Greedy optimization strategy

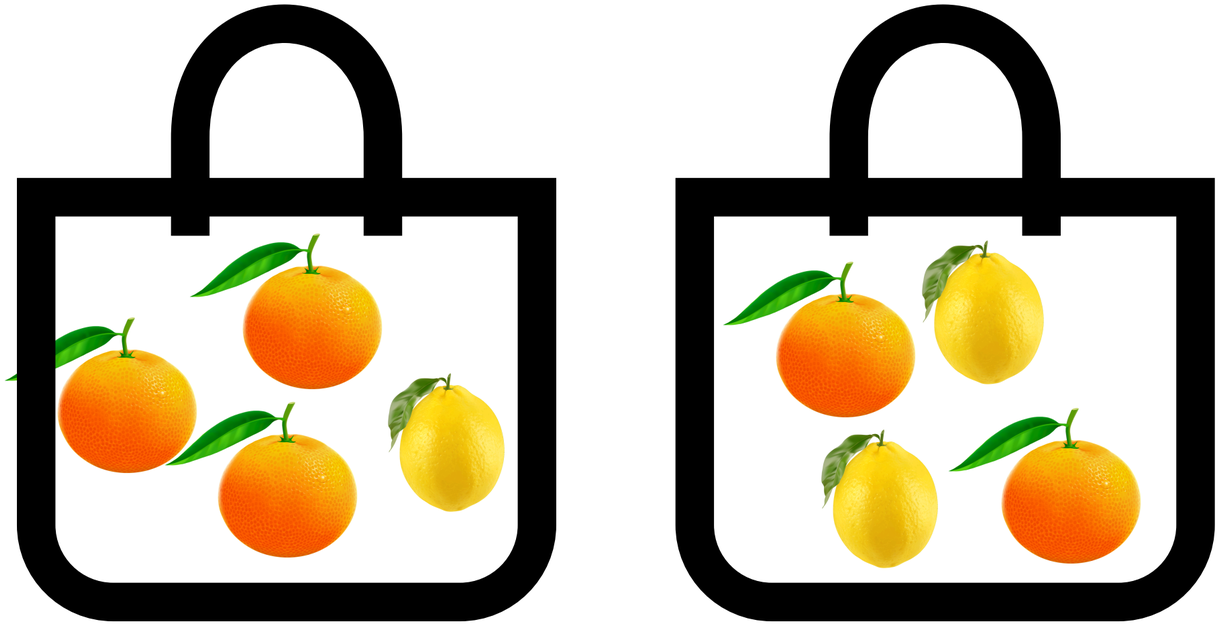
Decision Trees

Idea: **In which bag can u make a more confident decision of what fruit you'll randomly take?**



Decision Trees

Idea: **In which bag can u make a more confident decision of what fruit you'll randomly take?**



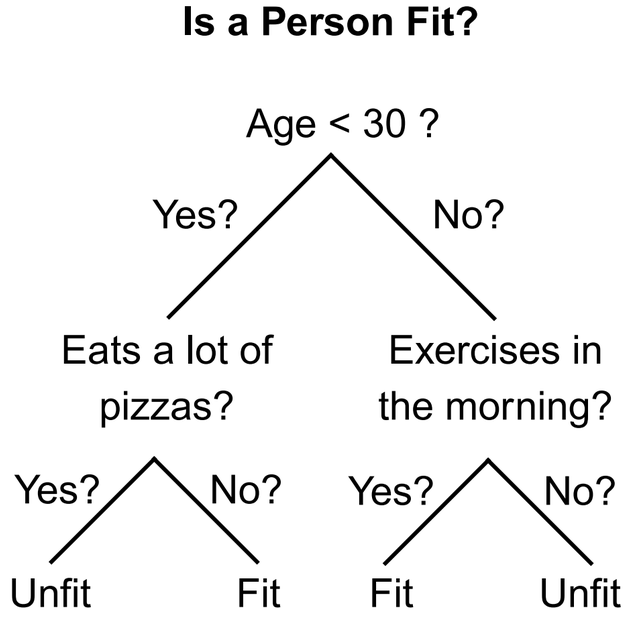
The right bag is more 'mixed'

Decision Trees

Idea:

–Create disjoint regions in the space, where data is less ‘mixed’ so decisions locally can be more confident.

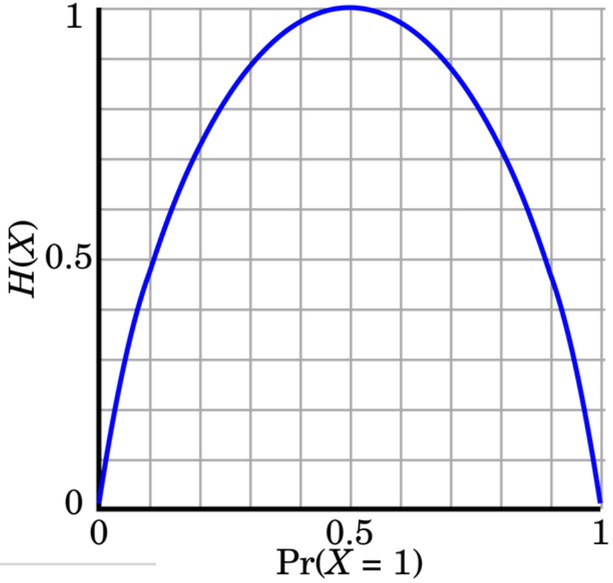
–Done by creating if-then questions!



Decision Trees

How can we measure how mixed our bags are?

$$\text{Entropy} = -p\log_2 p - (1-p)\log_2(1-p)$$



Basket Contents	Entropy	How Mixed?
100% lemons	0.0	Not mixed – only lemons
50% lemons / 50% oranges	1.0 (max)	Very mixed – max uncertainty
80% lemons / 20% oranges	~0.72	A bit mixed

Decision Trees

$$\text{Information Gain} = \text{Entropy(before)} - \text{Weighted Entropy(after)}$$

$$\text{Weighted Entropy(after)} = \frac{n_1}{n} \cdot \text{Entropy}_1 + \frac{n_2}{n} \cdot \text{Entropy}_2 + \dots$$

Suppose we have:

- 10 fruits: 5 🍋 lemons, 5 🍊 oranges →

$$\text{Entropy(before)} = 1.0$$

Now we split them:

- Basket A: 4 lemons, 1 orange → Entropy = 0.72
- Basket B: 1 lemon, 4 oranges → Entropy = 0.72

$$\text{Weighted Entropy} = 0.5 \cdot 0.72 + 0.5 \cdot 0.72$$

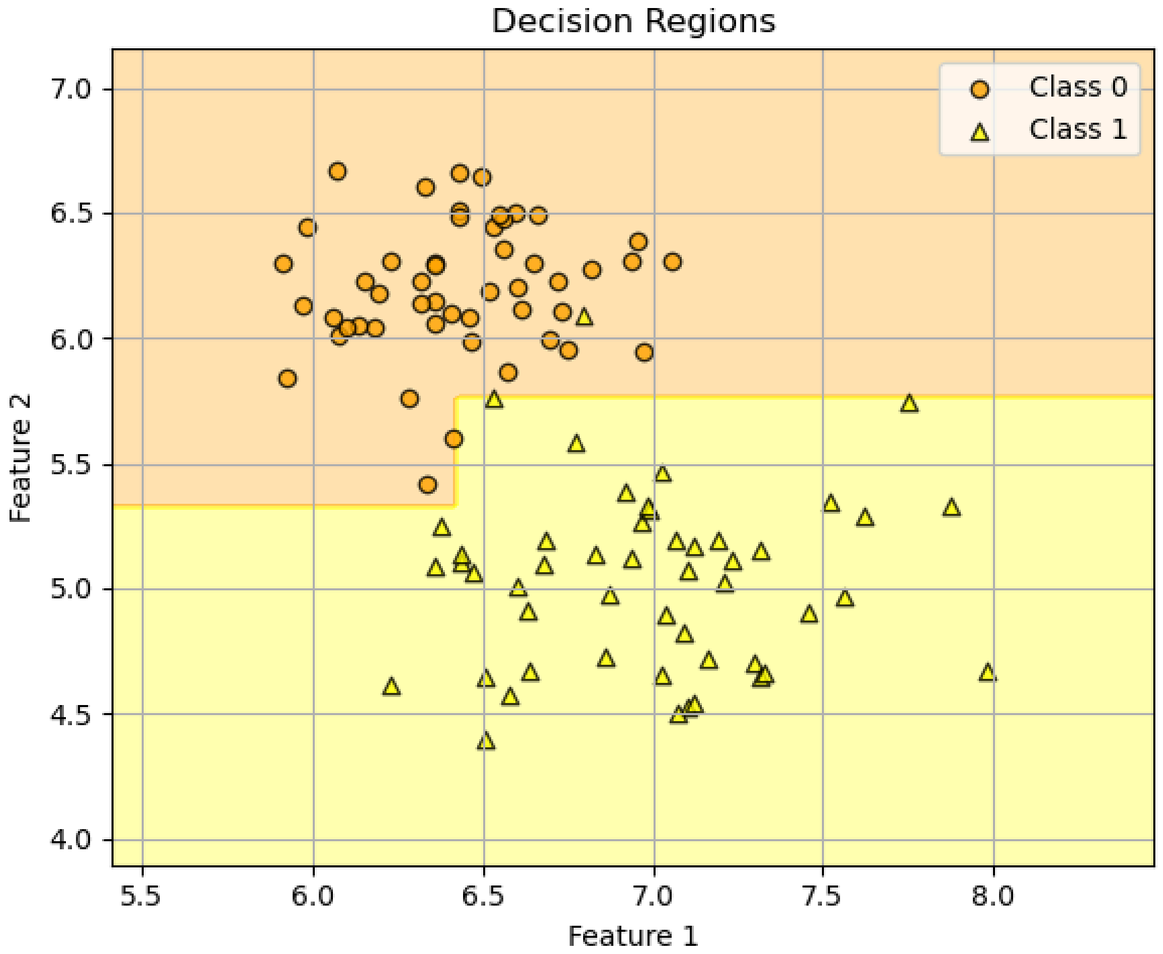
$$\text{Information Gain} = 1.0 - 0.72 = 0.28$$

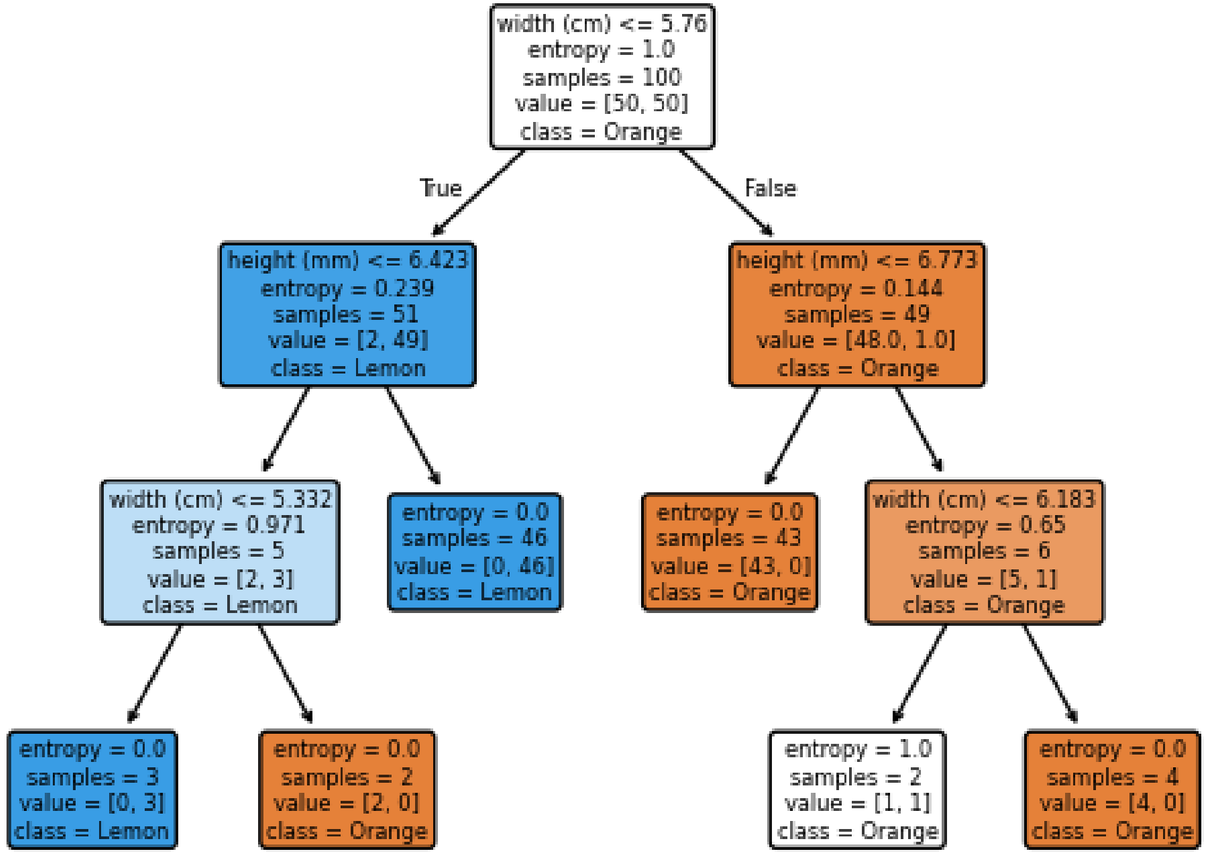
We **reduced uncertainty**, so this is a **useful split**!

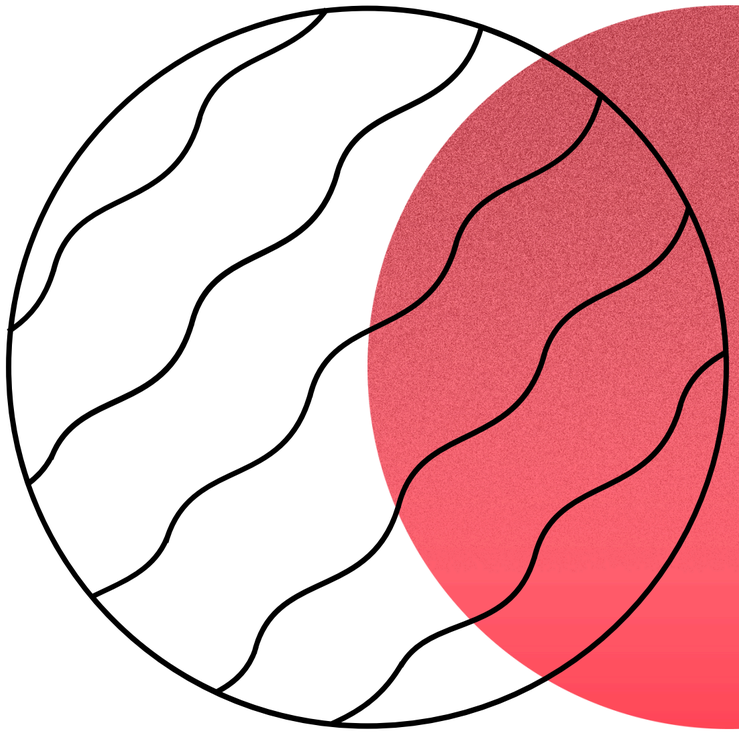
Decision Trees

```
function BuildTree(data, depth):
    if stopping_condition(data, depth):
        return Leaf(predict_class(data))

    best_split = None
    best_score = ∞

    for feature in data.features:
        for threshold in possible_splits(data, feature):
            left, right = split(data, feature, threshold)

            if left is empty or right is empty:
                continue

            score = weighted_impurity(left, right)

            if score < best_score:
                best_score = score
                best_split = (feature, threshold, left, right)

    if best_split is None:
        return Leaf(predict_class(data))

    feature, threshold, left, right = best_split

    left_subtree = BuildTree(left, depth + 1)
    right_subtree = BuildTree(right, depth + 1)

    return Node(feature, threshold, left_subtree, right_subtree)
```

Decision Trees







Artificial Neural Networks

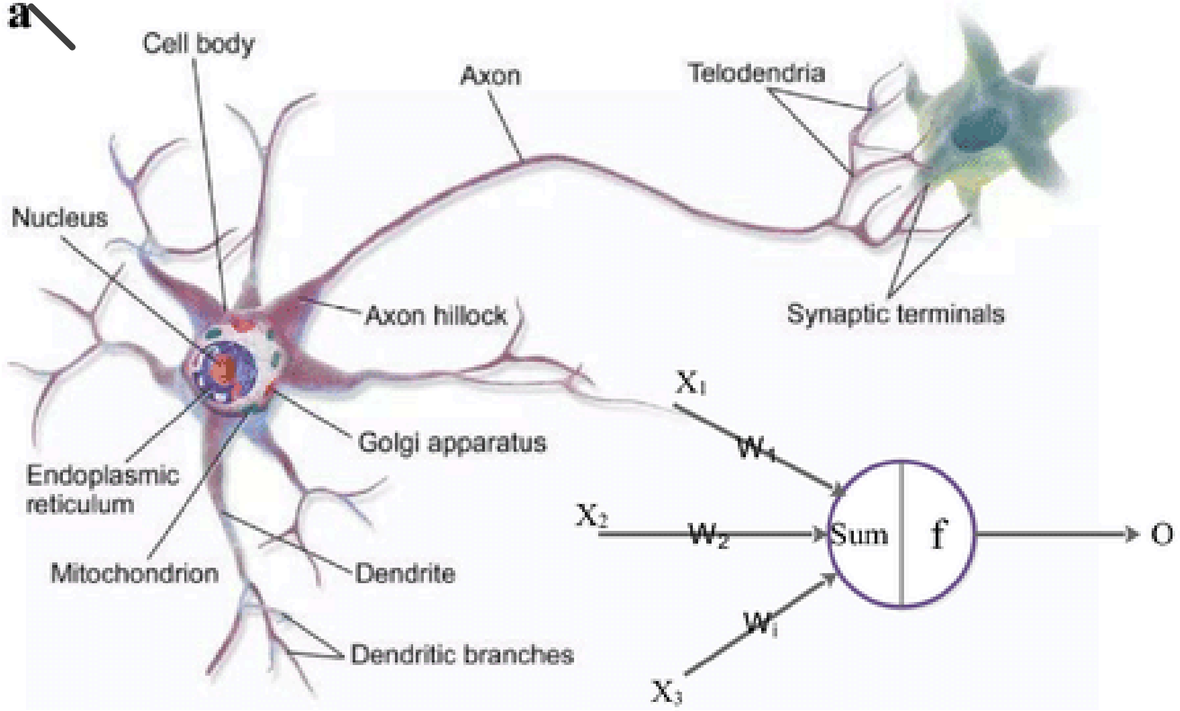
Artificial Neural Networks

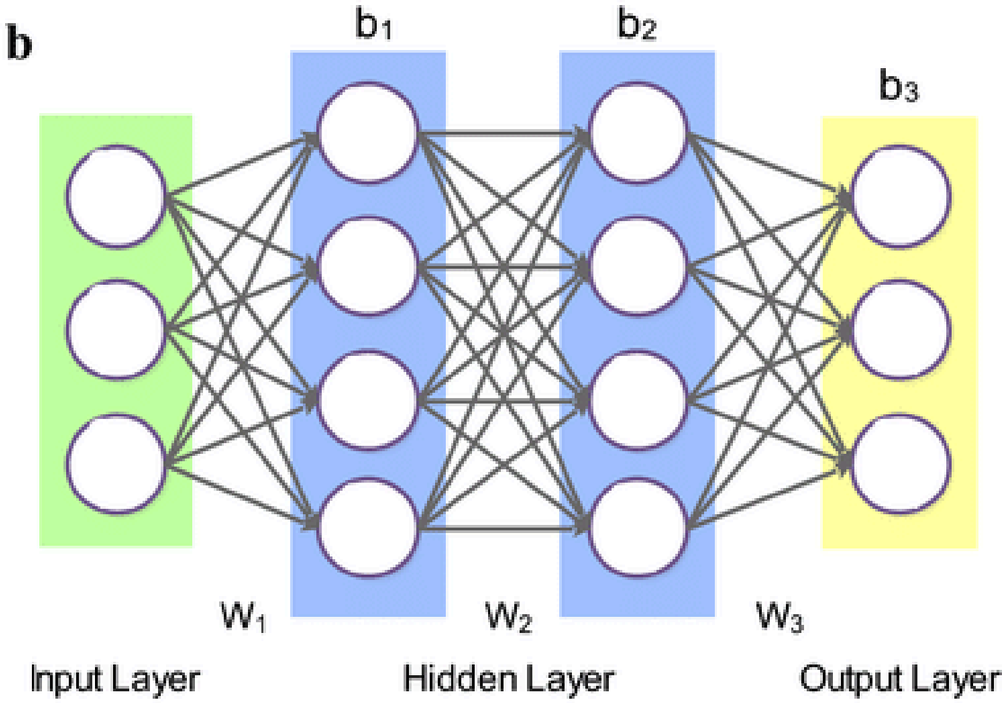
- A computational model ‘inspired’ by how the human brain processes information
- Composed of layers of interconnected “neurons” that transform inputs into outputs

- The ‘Legos’ of Machine Learning

Artificial Neural Networks

Idea:

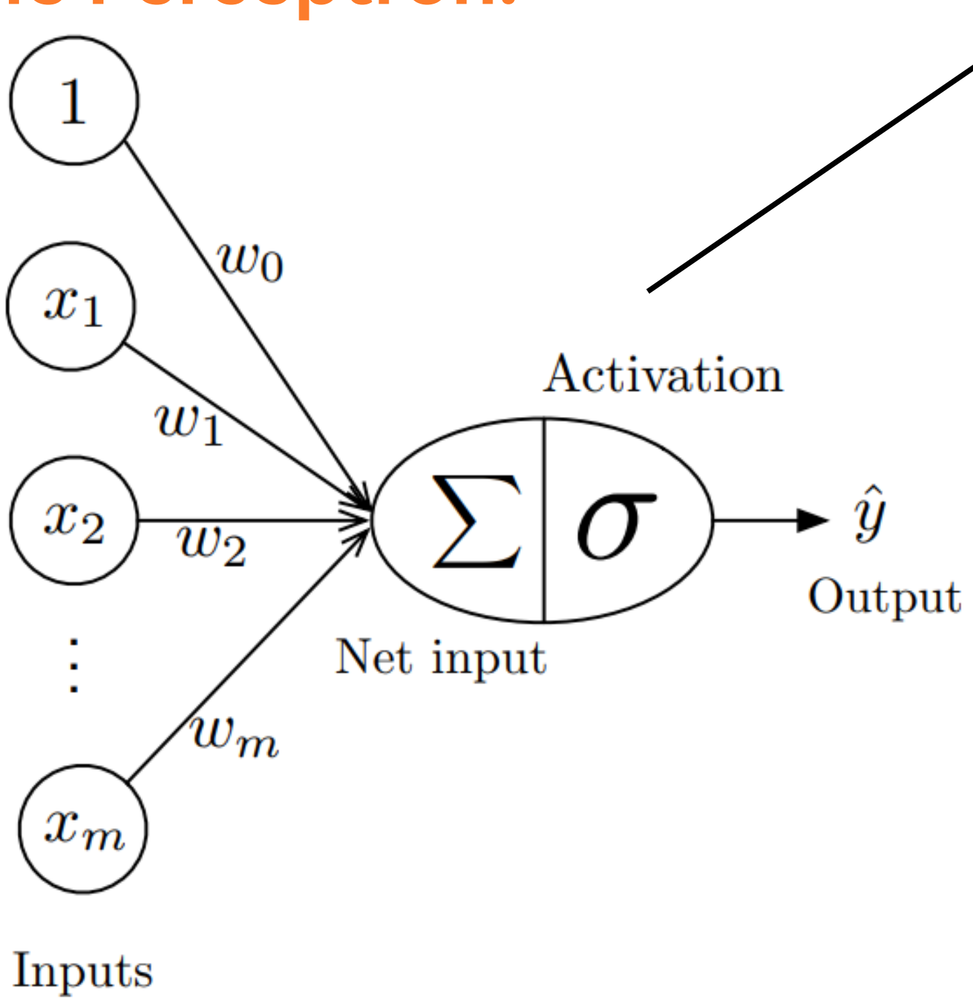




FROM: An intelligent recommendation approach for online advertising based on hybrid deep neural network and parallel computing - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/a-Biological-neuron-structure-and-neuron-model-b-Deep-neural-network-structure_fig1_334637084 [accessed 19 May 2025]

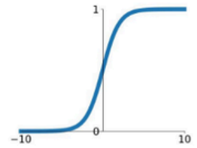
Artificial Neural Networks

The Perceptron:

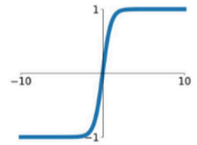


Activation Functions

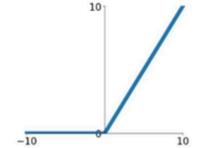
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



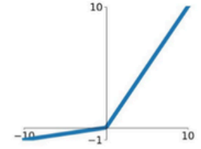
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

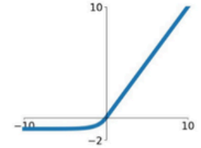


Leaky ReLU
$\max(0.1x, x)$



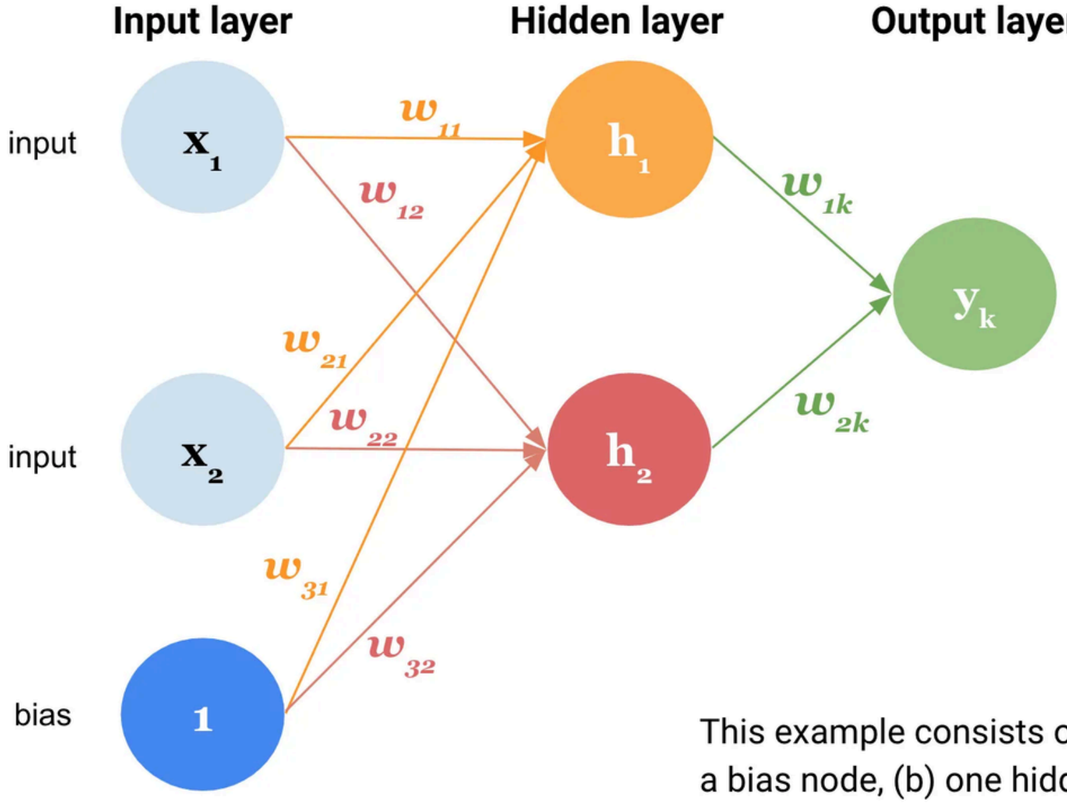
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation introduces non-linearities

Artificial Neural Networks



FROM: An intelligent recommendation approach for online advertising based on hybrid deep neural network and parallel computing - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/a-Biological-neuron-structure-and-neuron-model-b-Deep-neural-network-structure_fig1_334637084 [accessed 19 May 2025]

Artificial Neural Networks

Finding the weights:

1 - Define how bad current predictions are using a Loss Function:

$$\text{Loss} = -\left[y \cdot \log\left(\hat{y}\right) + (1-y) \cdot \log\left(1-\hat{y}\right)\right]$$

$y \in \{0, 1\}$ is the **true label**

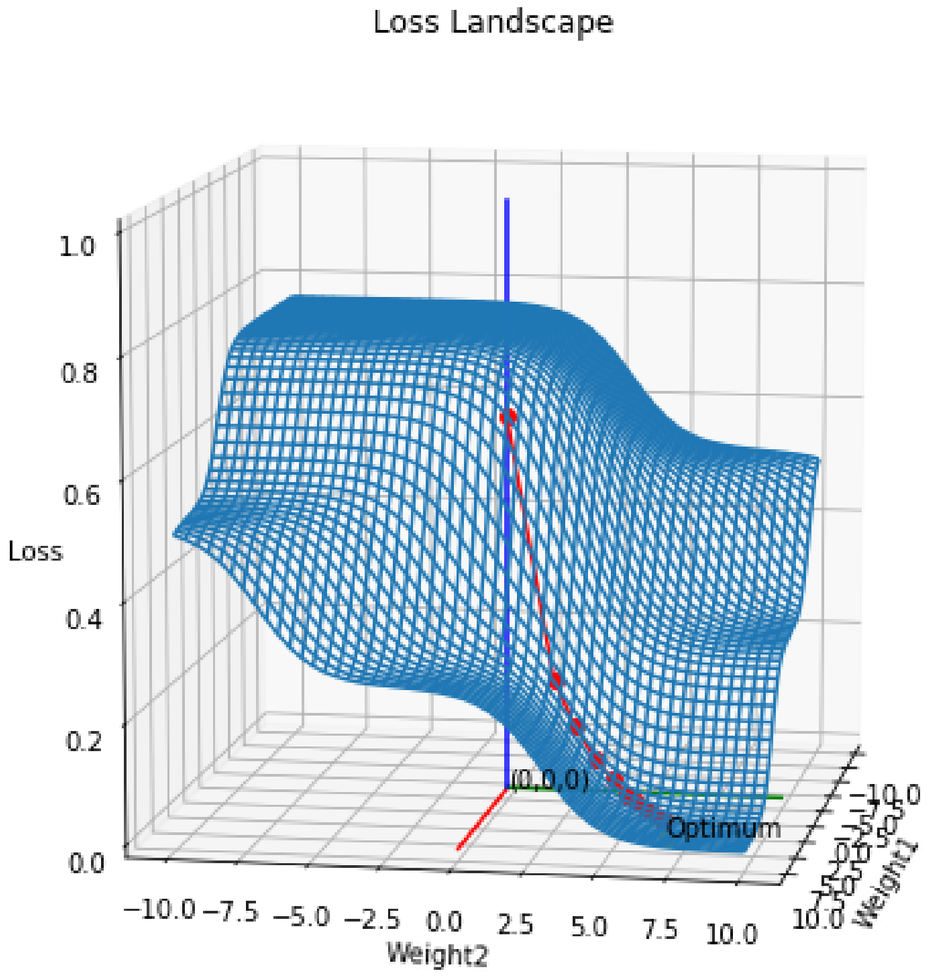
$\hat{y} \in (0, 1)$ is the **predicted probability**

2 - Find weights that minimize this Loss:

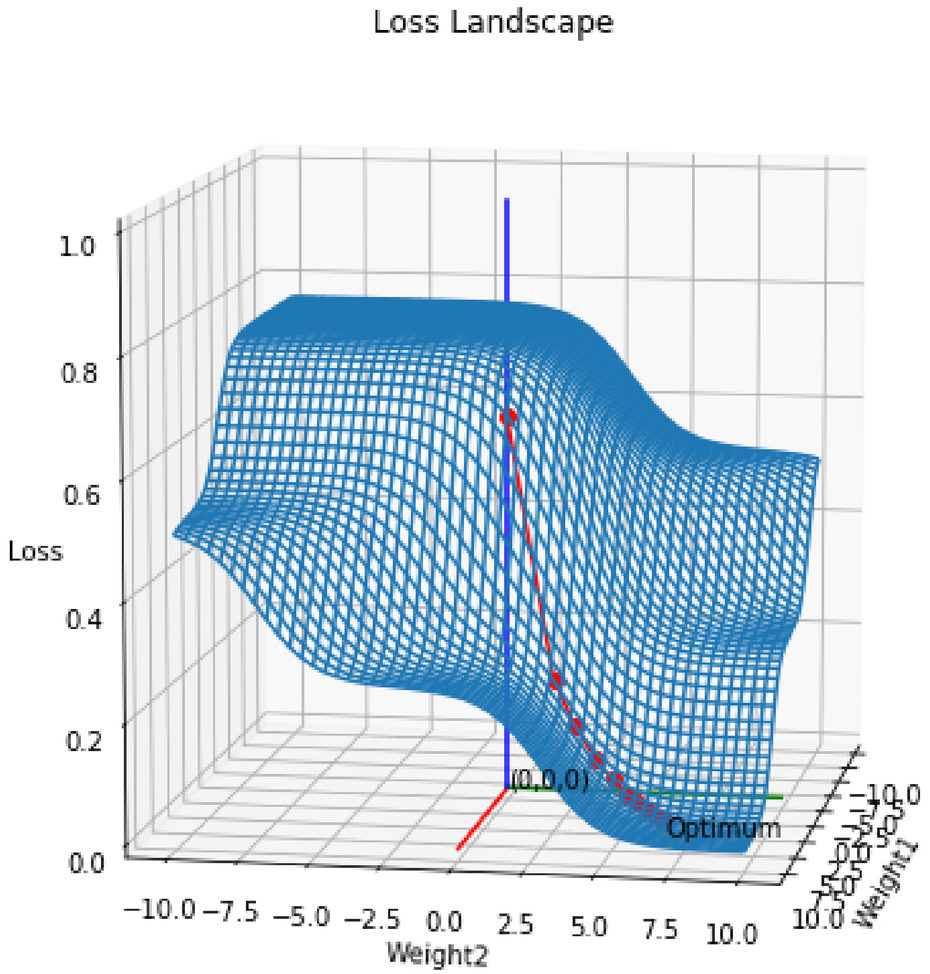
$$\min_{\mathbf{w}} \mathcal{L}\left(\mathbf{w}\right) = \frac{1}{n}\sum_{i=1}^{n}\left[-y^{(i)}\log\left(\hat{y}^{(i)}\right) - \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right]$$

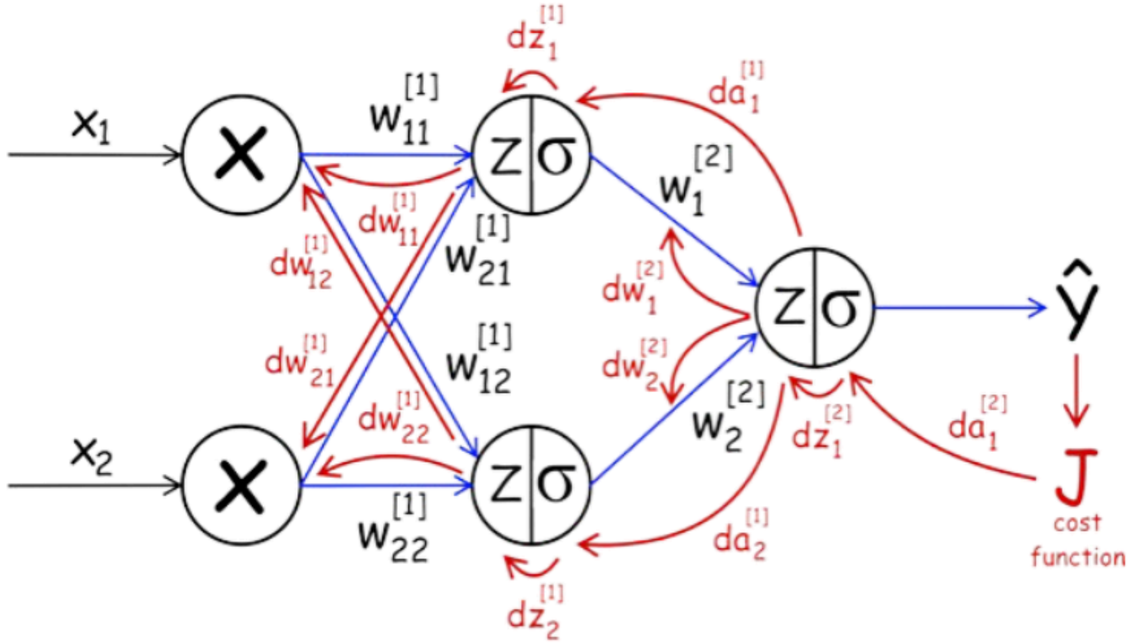
Solved using gradient descent

Artificial Neural Networks



Artificial Neural Networks

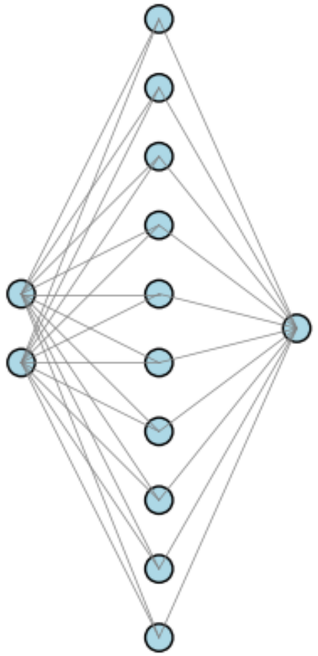


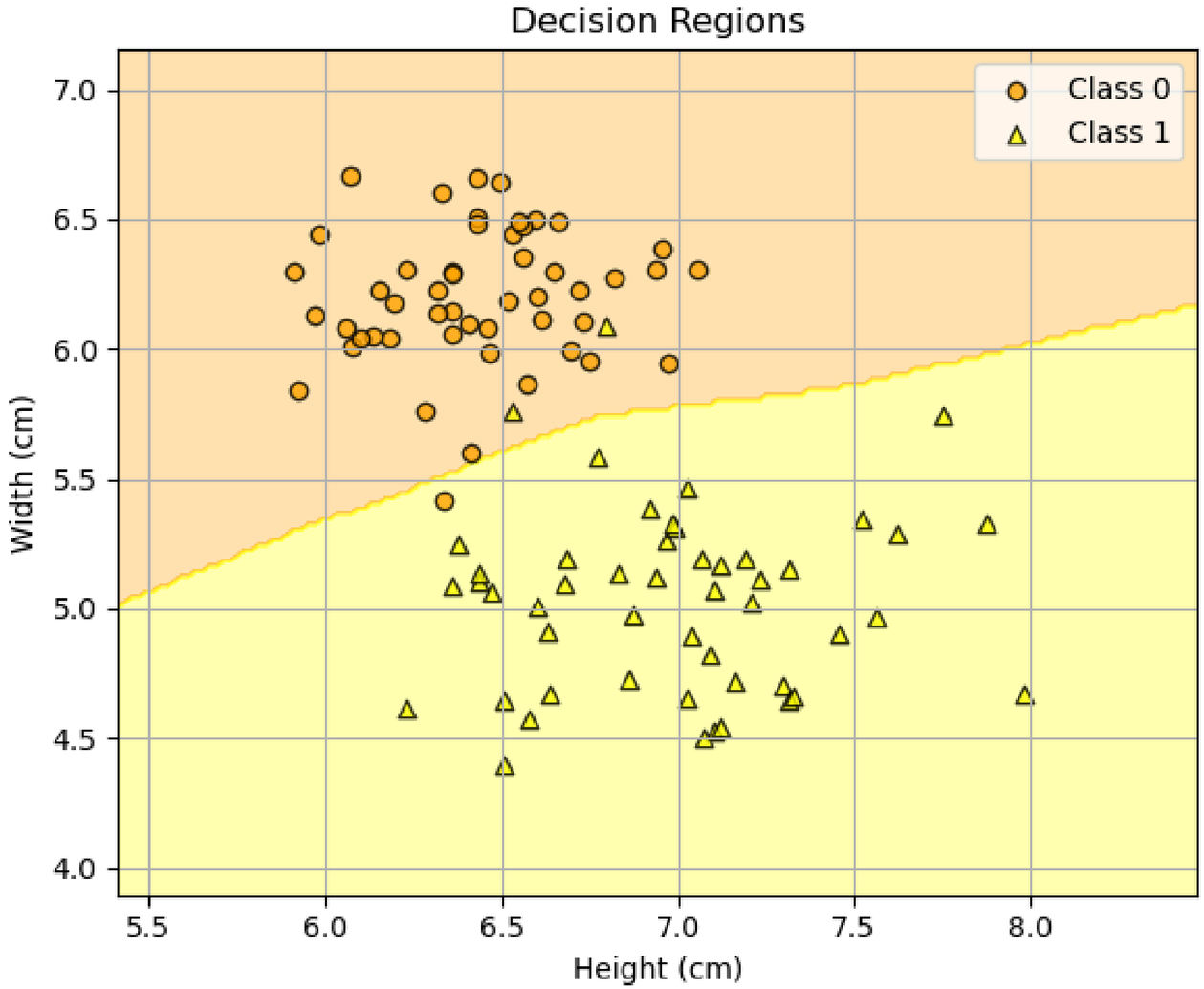


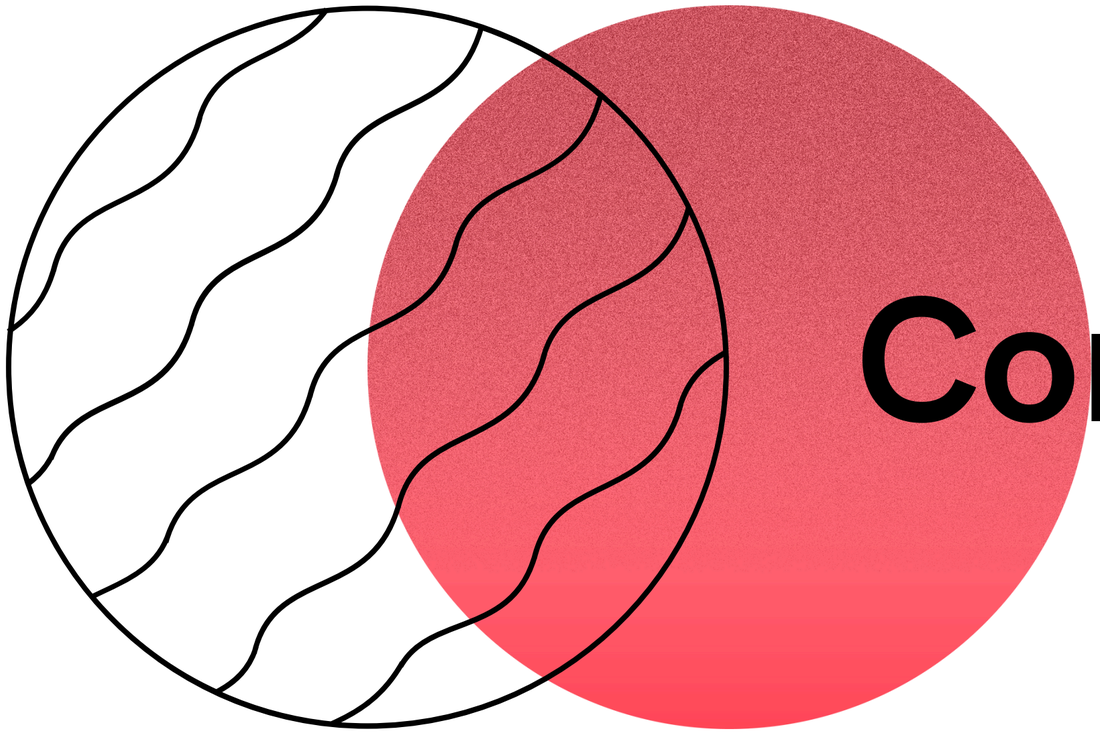
Backpropagation

Artificial Neural Networks

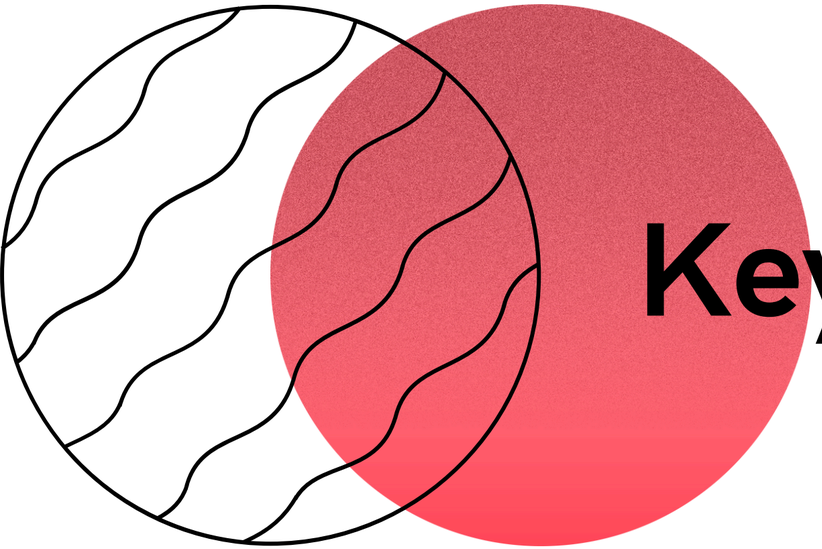
NN Arquitecture







Conclusion



Key Takeaways

Feedback here

- K-Nearest Neighbours
- Naive Bayes
- Decision Trees
- Artificial Neural Networks

Evaluating Machine Learning Models