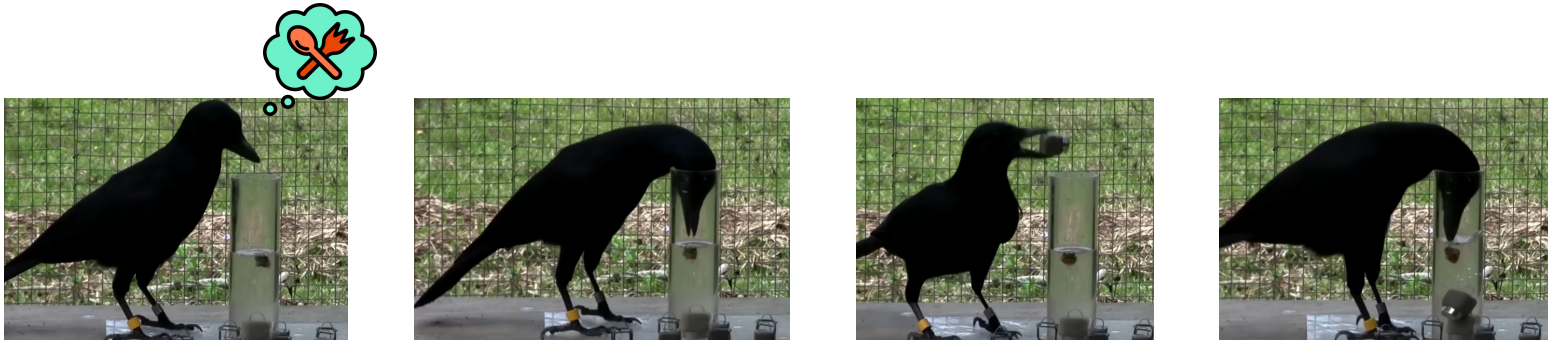
Introductory Seminar on Artificial Intelligence and Machine Learning

Emanuele Ledda, Cagliari Digital Lab 2024 - Day 2

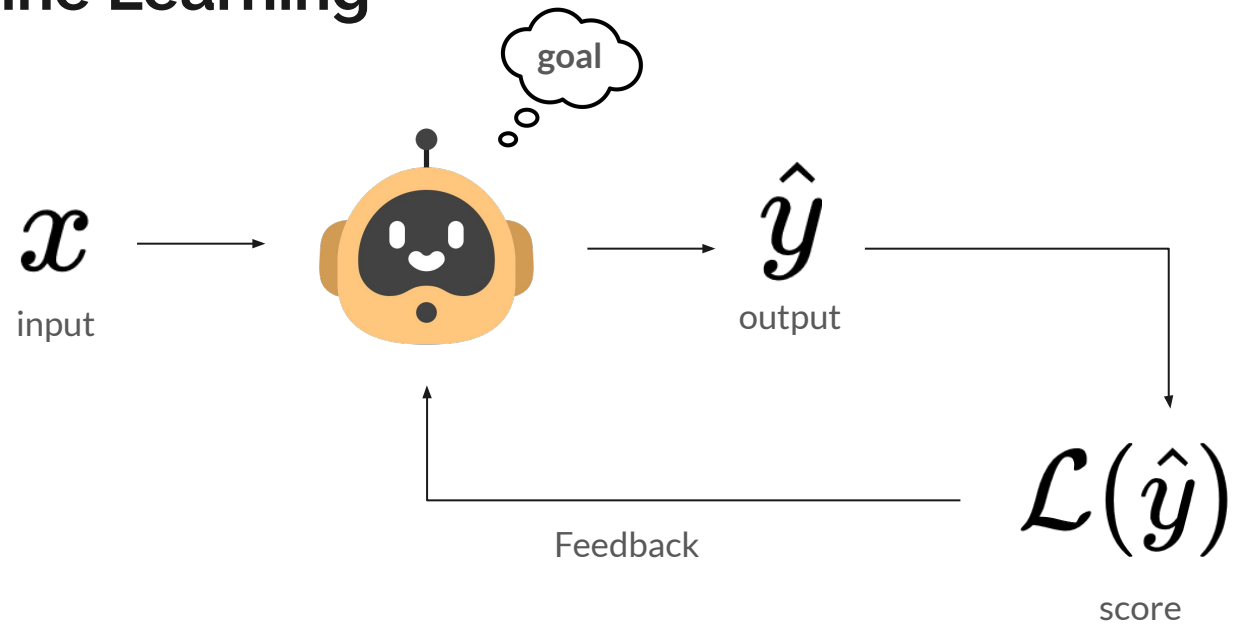
Shallow Learning

Machine Learning

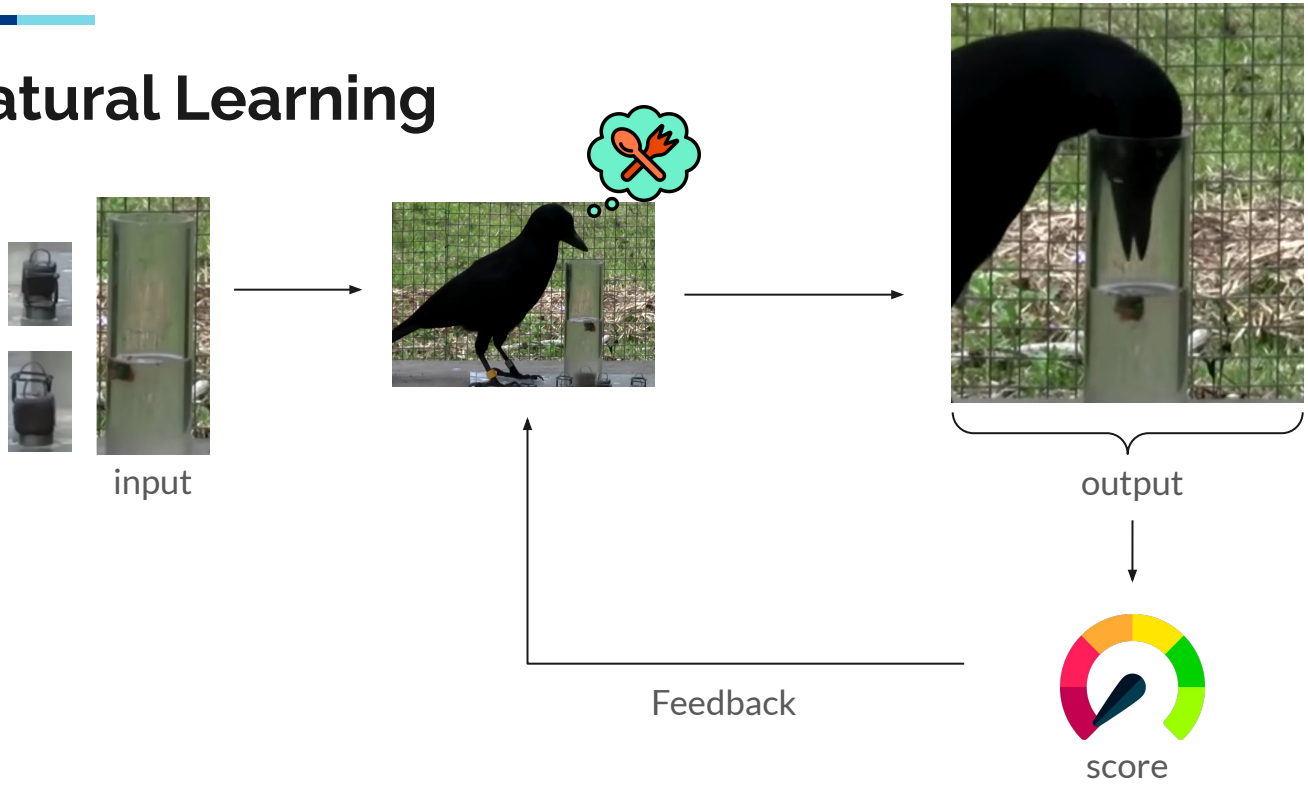
Natural Learning



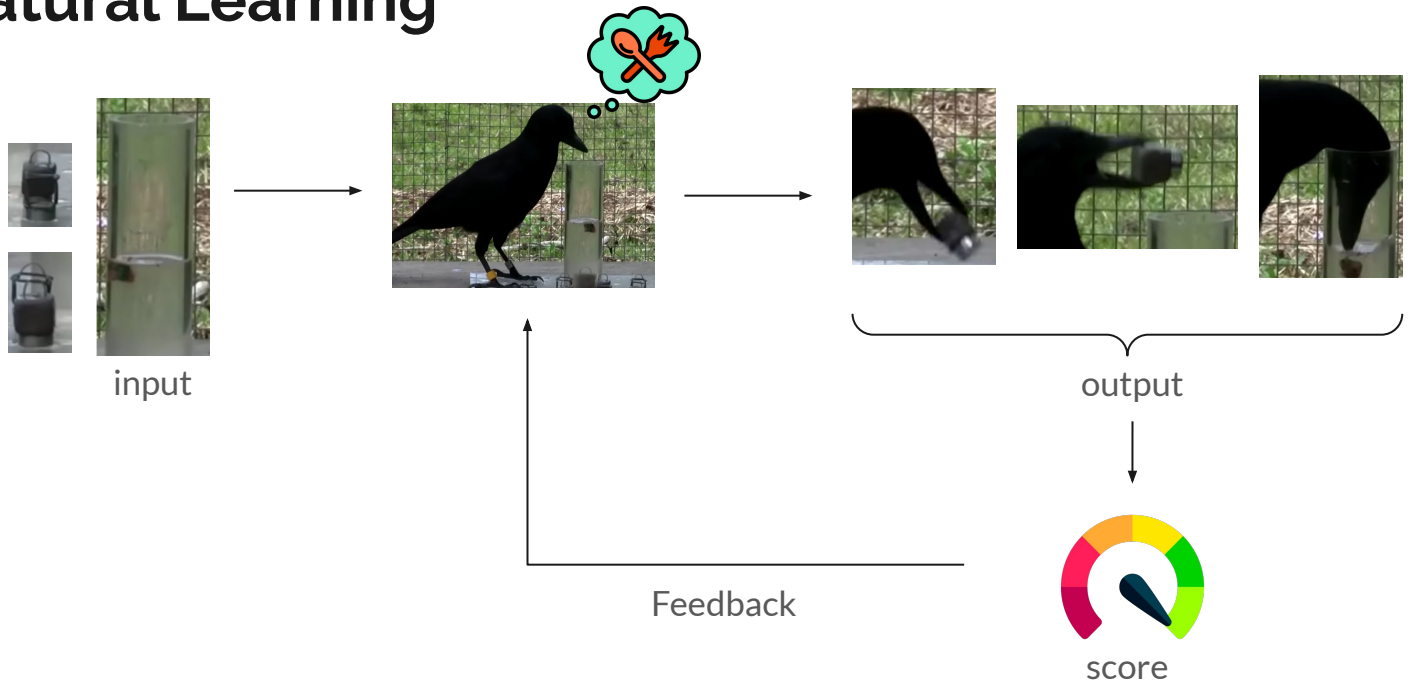
Machine Learning



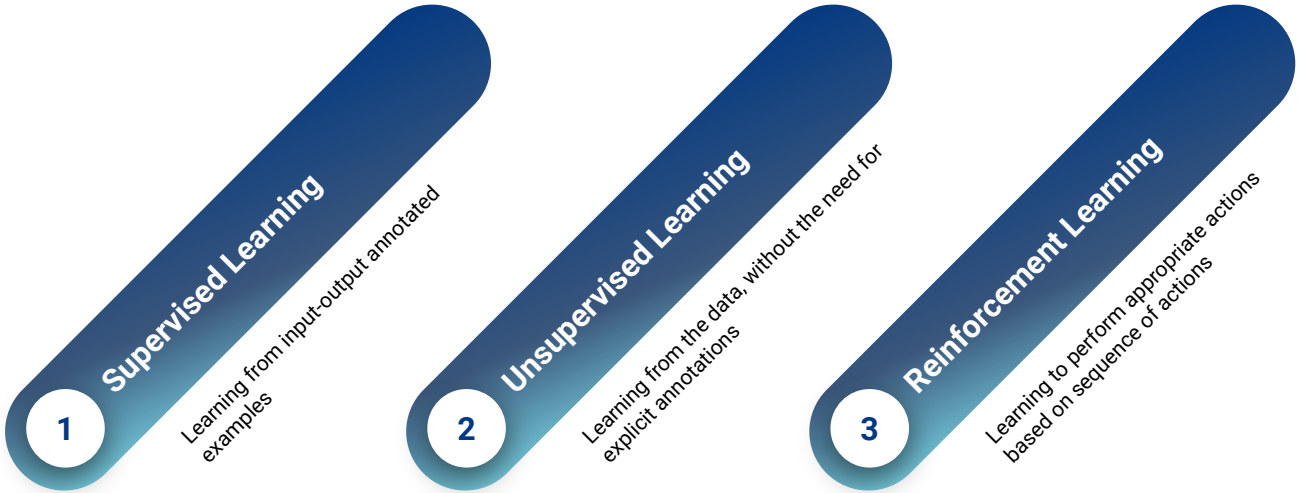
Natural Learning



Natural Learning



Learning Paradigms



Supervised Learning

The algorithm learns by using **labels**, i.e., annotations of correct outputs associated with specific inputs that the machine can use for learning correct behaviors.

Examples of supervised task:

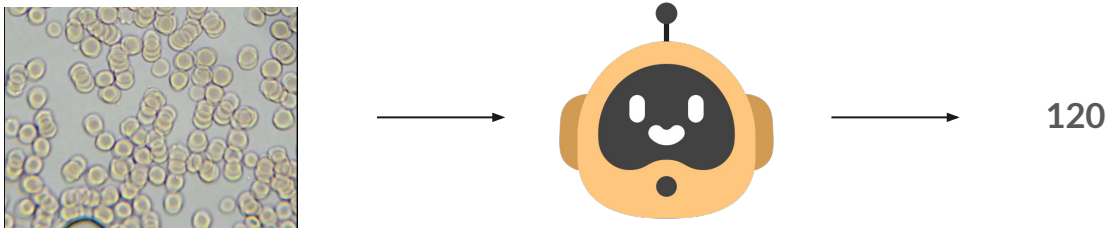
- **Classification**: assigning a **class** to an input object

Supervised Learning

The algorithm learns by using **labels**, i.e., annotations of correct outputs associated with specific inputs that the machine can use for learning correct behaviors.

Examples of supervised task:

- **Regression**: assigning a **numeric value** to an input object (e.g., counting)



Unsupervised Learning

The algorithm **cannot access labels** and learns by extracting information only from the input.

Examples of unsupervised task:

- **Clustering**: divide similar objects in clusters

input

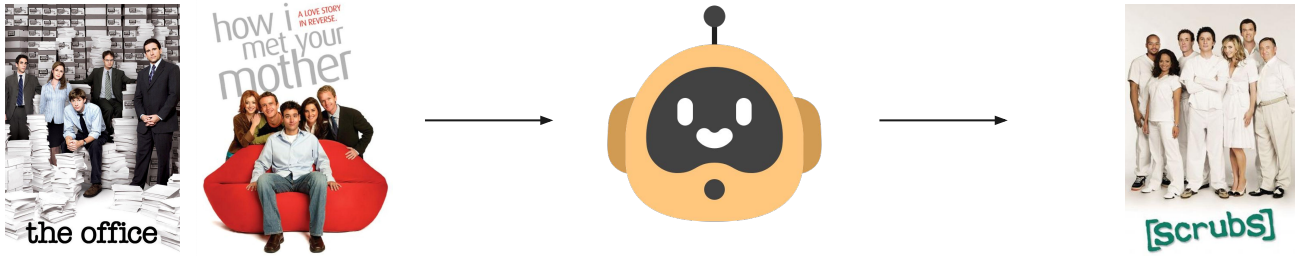
output

Unsupervised Learning

The algorithm **cannot access labels** and learns by extracting information only from the input.

Examples of unsupervised task:

- **Association Analysis**: Extract association between objects (e.g., recommending tv series)



input

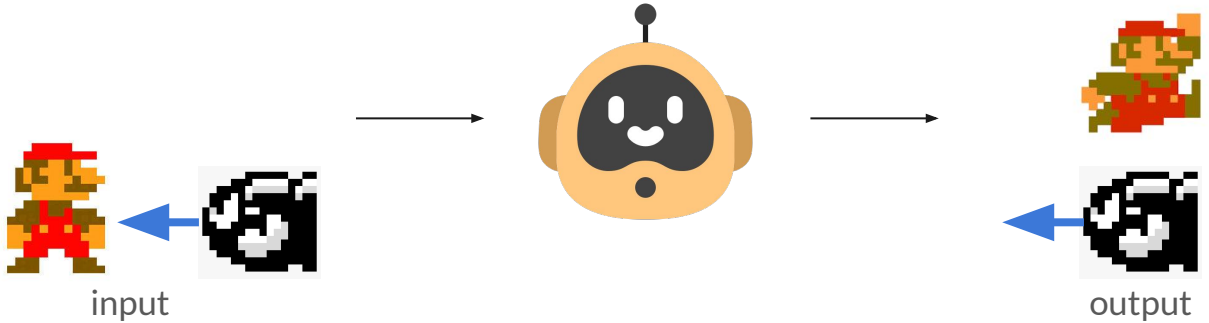
output

Reinforcement Learning

The algorithm should find an optimal action policy based on examples of the outcome of entire courses of actions.

Examples of reinforcement learning tasks:

- **Game Playing**: Learn which move should the agent take for winning a game



Linear Classifiers

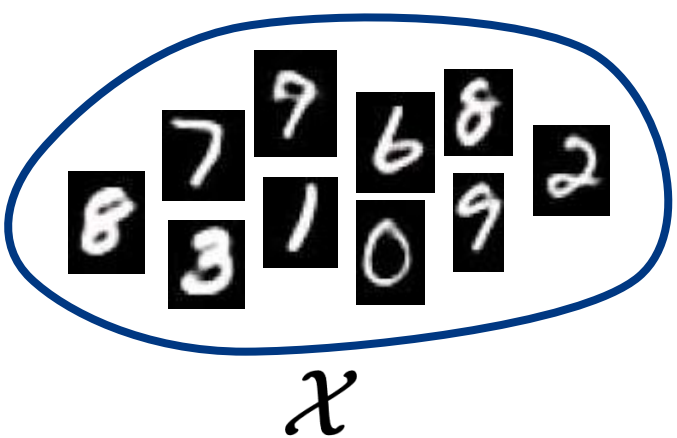
Supervised Classification

- Assigning a **class** to an input **object**

- One of the most common task:
  - Optical Character Recognition
  - Biometrics (e.g., face recognition)
  - Sentiment Analysis
  - Malware Detection
  - …

Supervised Classification

- Assigning a **class** to an input **object**
- e.g, the “objects” are images of hand written digits and the “classes” are the digits between 0 and 9

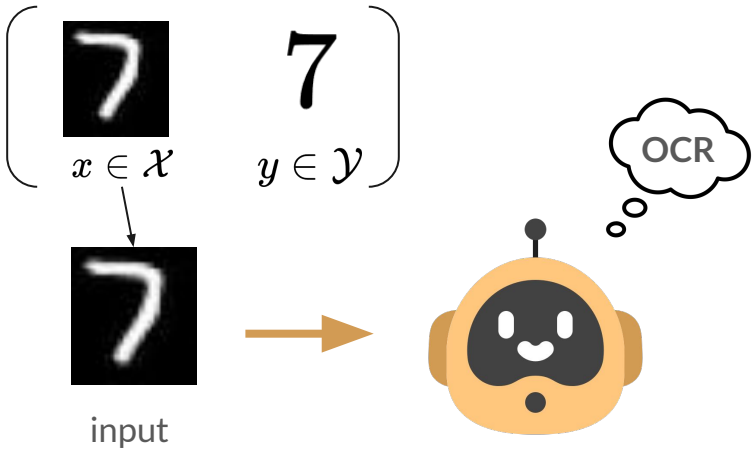


x

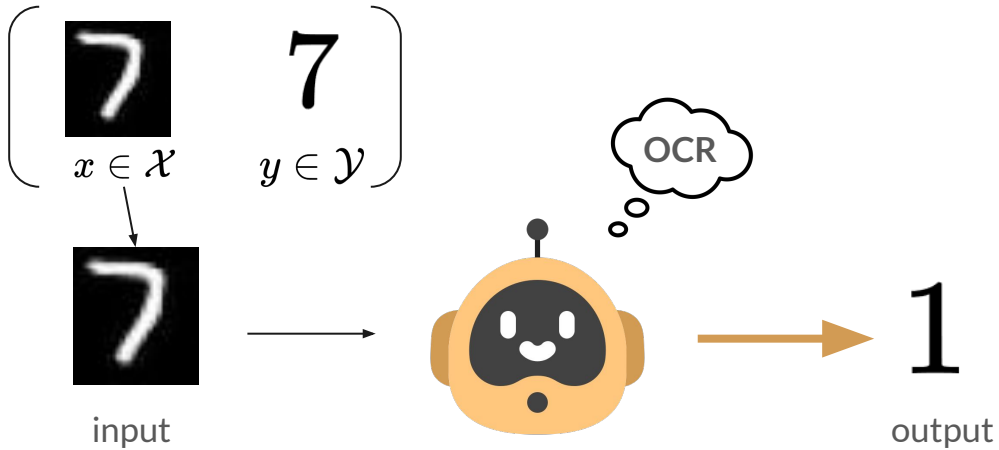
$$0, 1, 2, 3, 4,$$
$$5, 6, 7, 8, 9$$

y

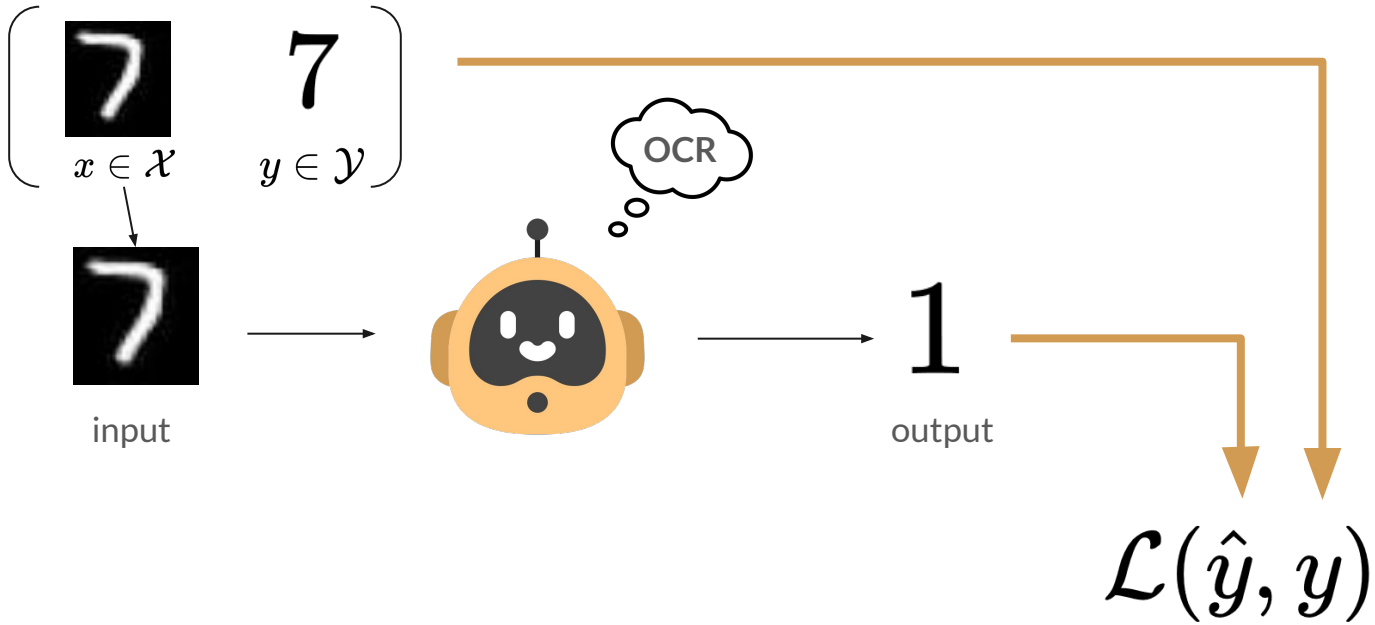
Supervised Classification



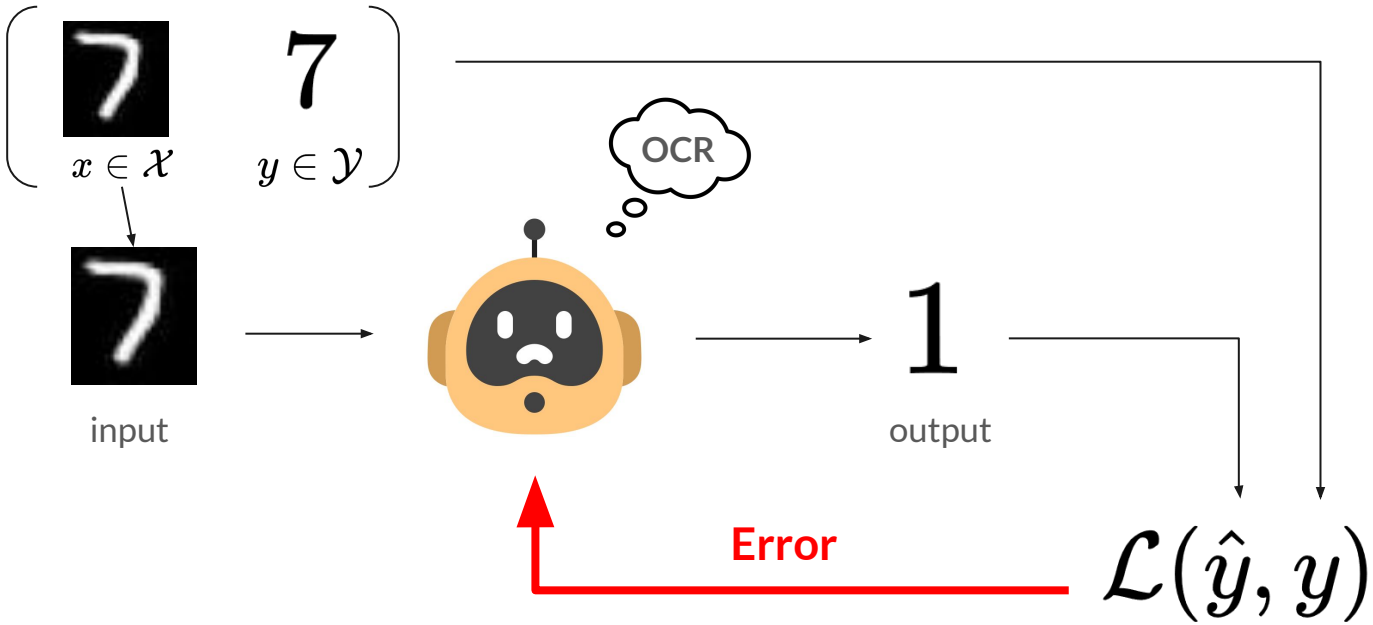
Supervised Classification



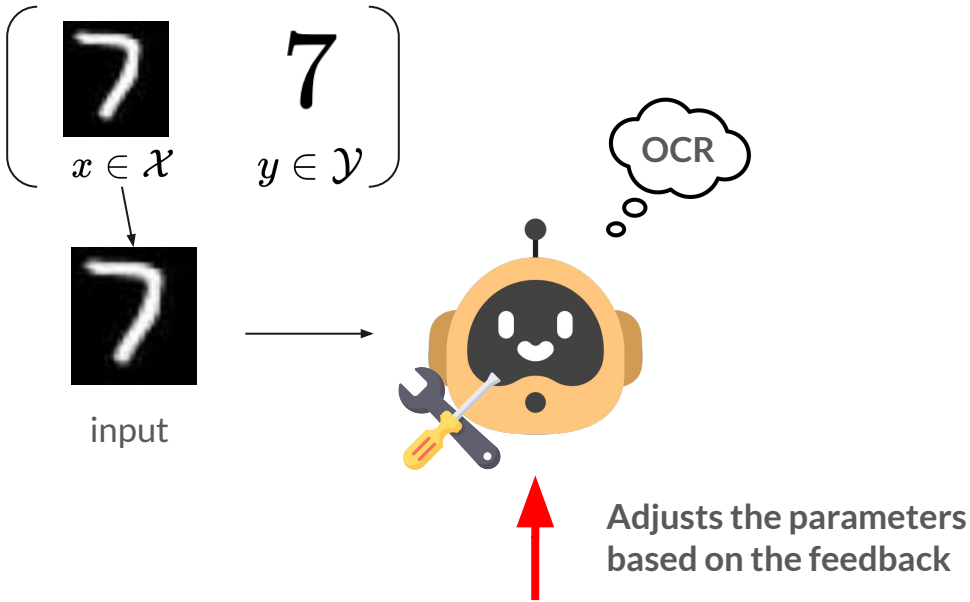
Supervised Classification



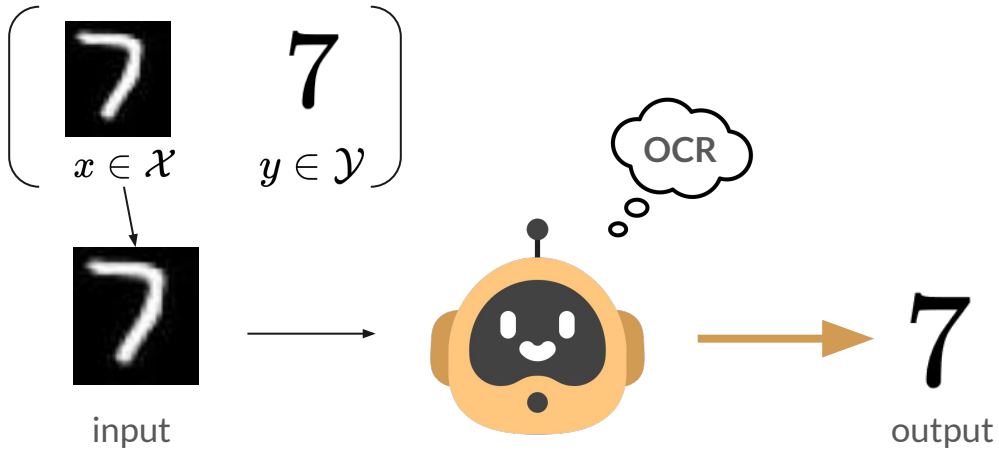
Supervised Classification



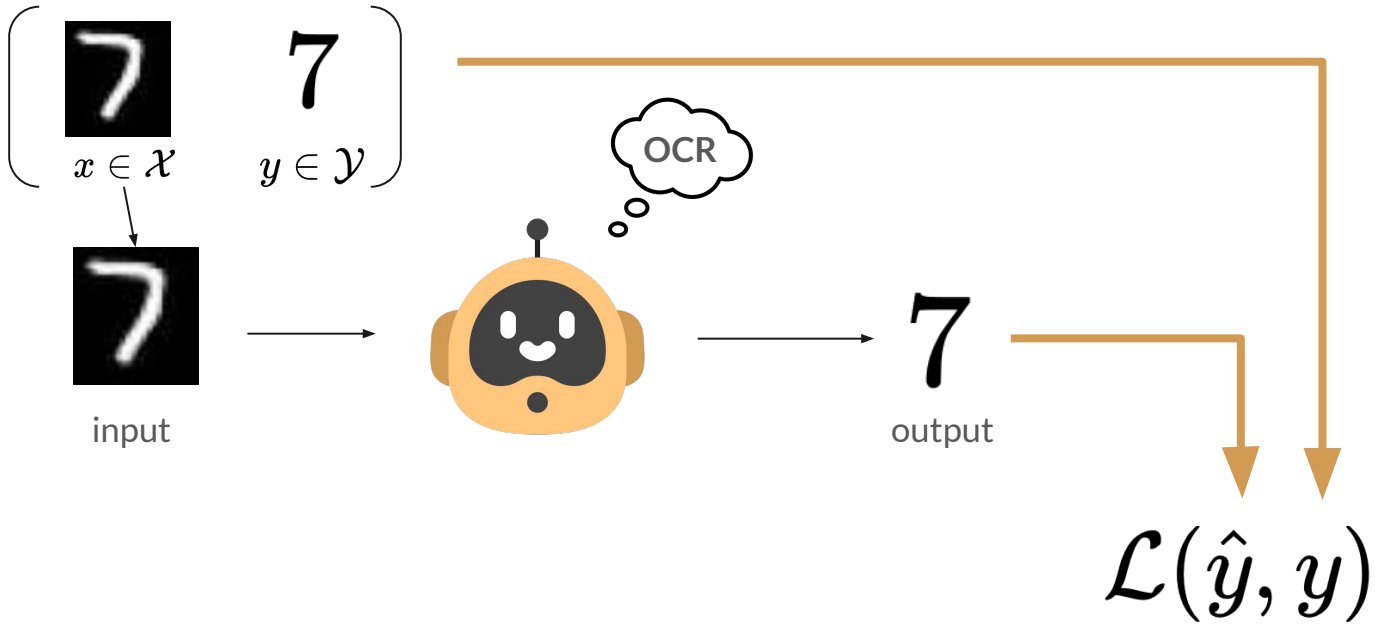
Supervised Classification



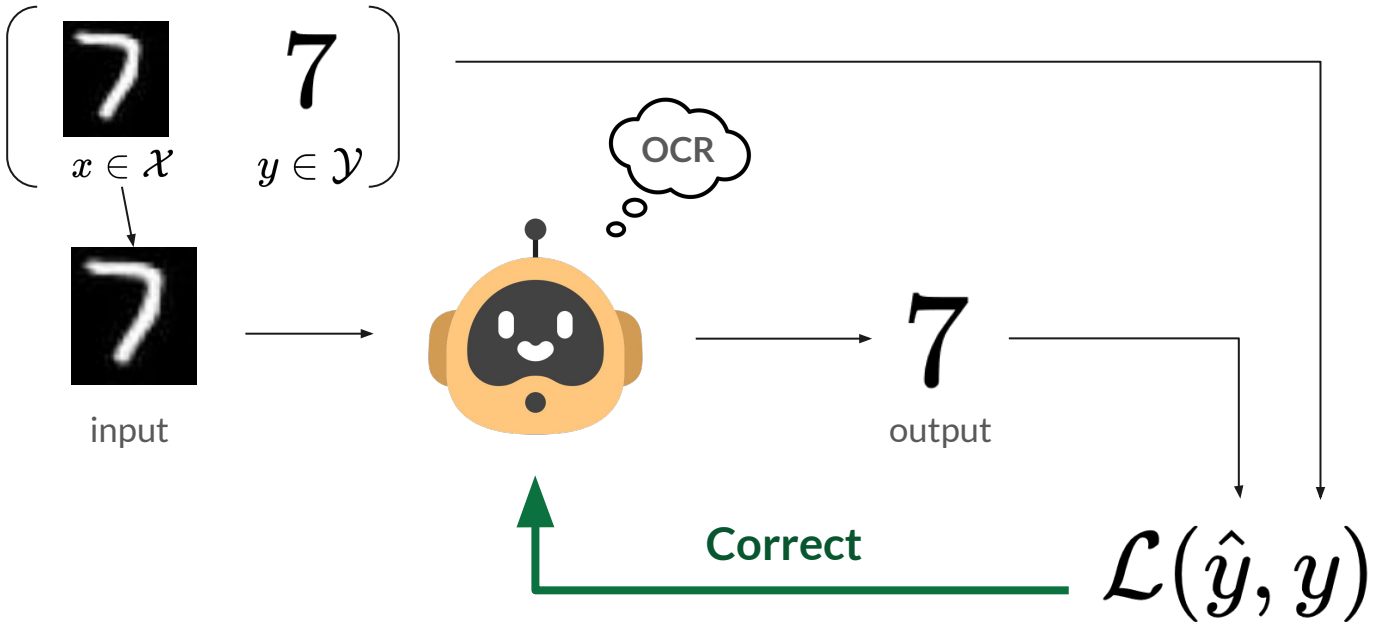
Supervised Classification



Supervised Classification



Supervised Classification



Supervised Classification

- Formally, we can see the input **x** as a **feature vector**, and the **y** as the **respective class**
- One can see a classification algorithm as a function **h** (called a **hypothesis**) mapping an input **x** to a class **y**.

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

- A specific hypothesis function represents a "**classifier**"
- The set of all possible hypotheses represents a **model** (also known as **hypothesis space**).

$$h \in \mathcal{H}$$

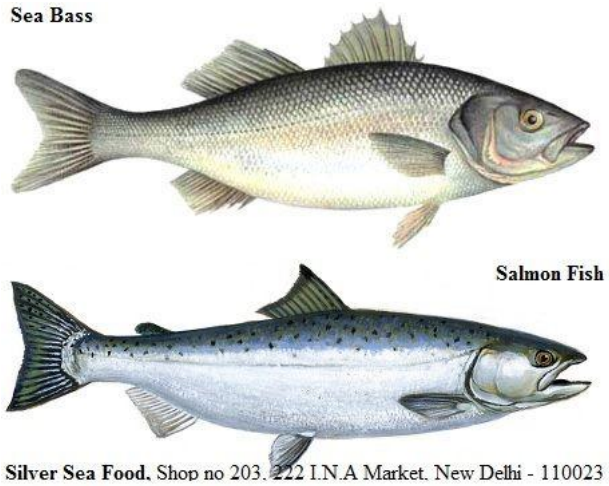
Supervised Classification

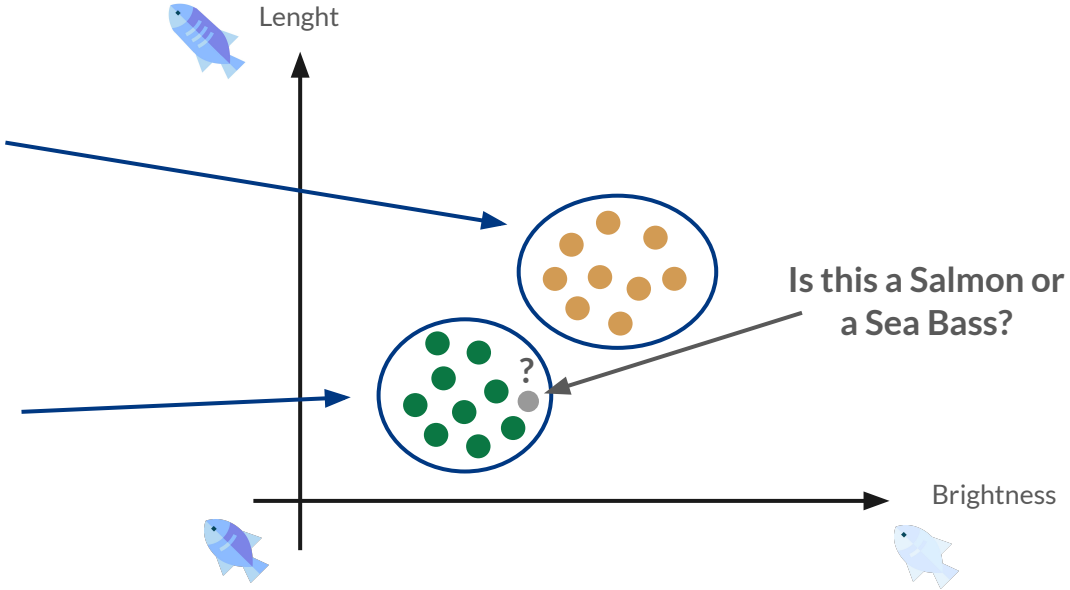
- Example of a simple model: **linear functions**

$$f(x) = \sum_{i=1}^{D} x_i \cdot w_i + b$$

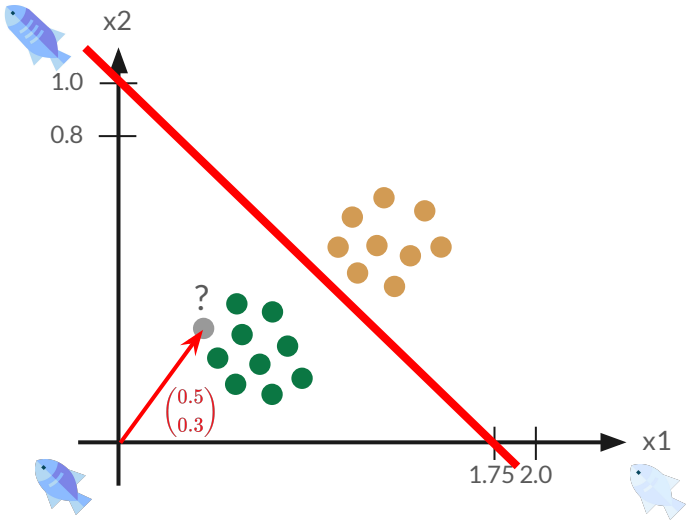
- **W** (weights) e **b** (bias) are the model parameters: a **classifier** is a specific instance of the weights and bias (**hyperplane**)

Binary Classification: Fishes Example





Linear Binary Classification



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

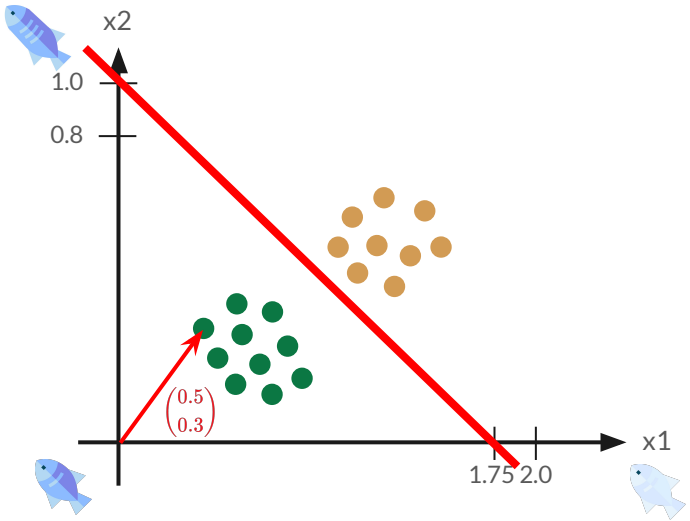
$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [0.5, 0.3]$$

Linear Binary Classification



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

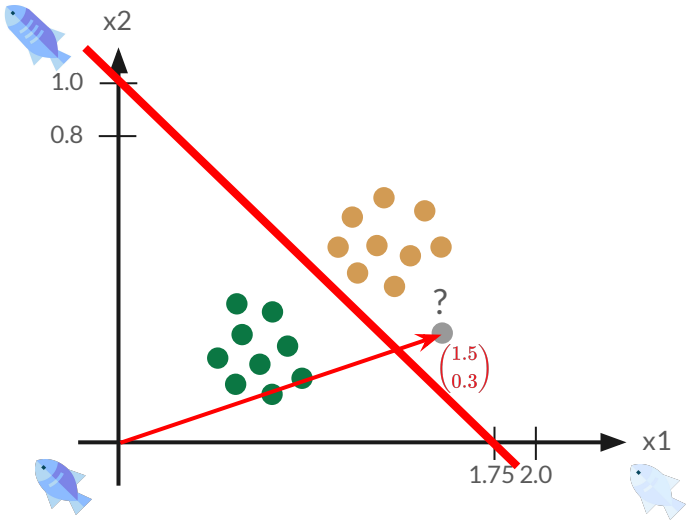
$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [0.5, 0.3]$$

$$0.5 \cdot 1.0 + 0.3 \cdot 1.75 - 1.75 = -0.725$$

Linear Binary Classification



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

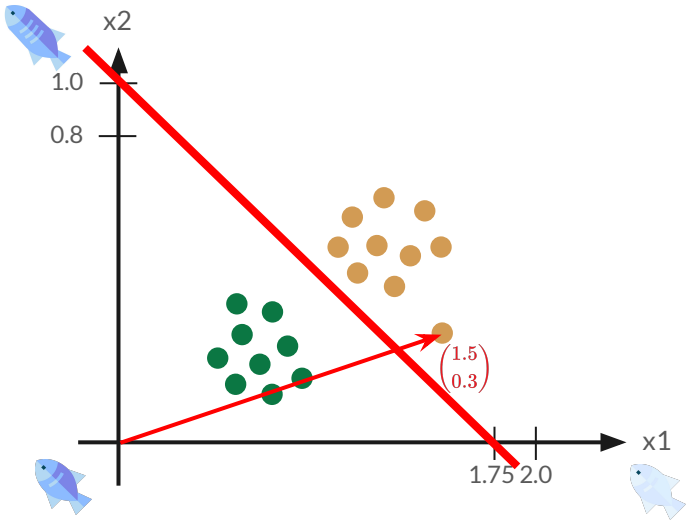
$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [1.5, 0.3]$$

Classificazione Binaria Lineare



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

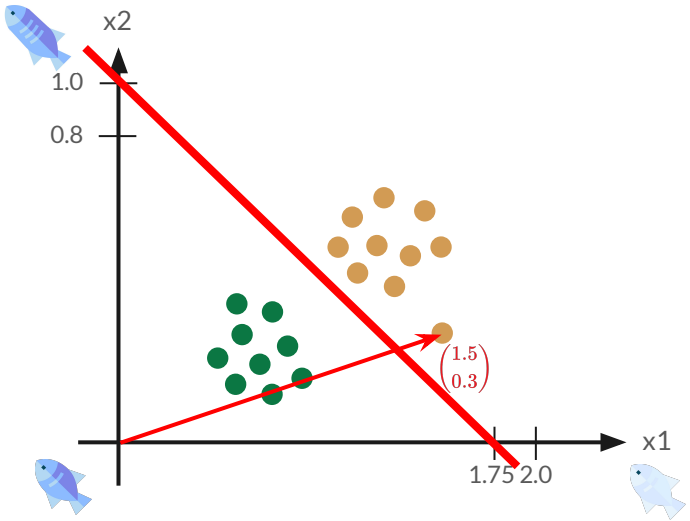
$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [1.5, 0.3]$$

$$1.5 \cdot 1.0 + 0.3 \cdot 1.75 - 1.75 = 0.275$$

Linear Binary Classification



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [1.5, 0.3]$$

$$1.5 \cdot 1.0 + 0.3 \cdot 1.75 - 1.75 = 0.275$$

Shallow Learning vs Deep Learning

Shallow Learning

- Simple tasks
- Smaller datasets
- Interpretability / Semi-Manual Feature Extraction

Deep Learning

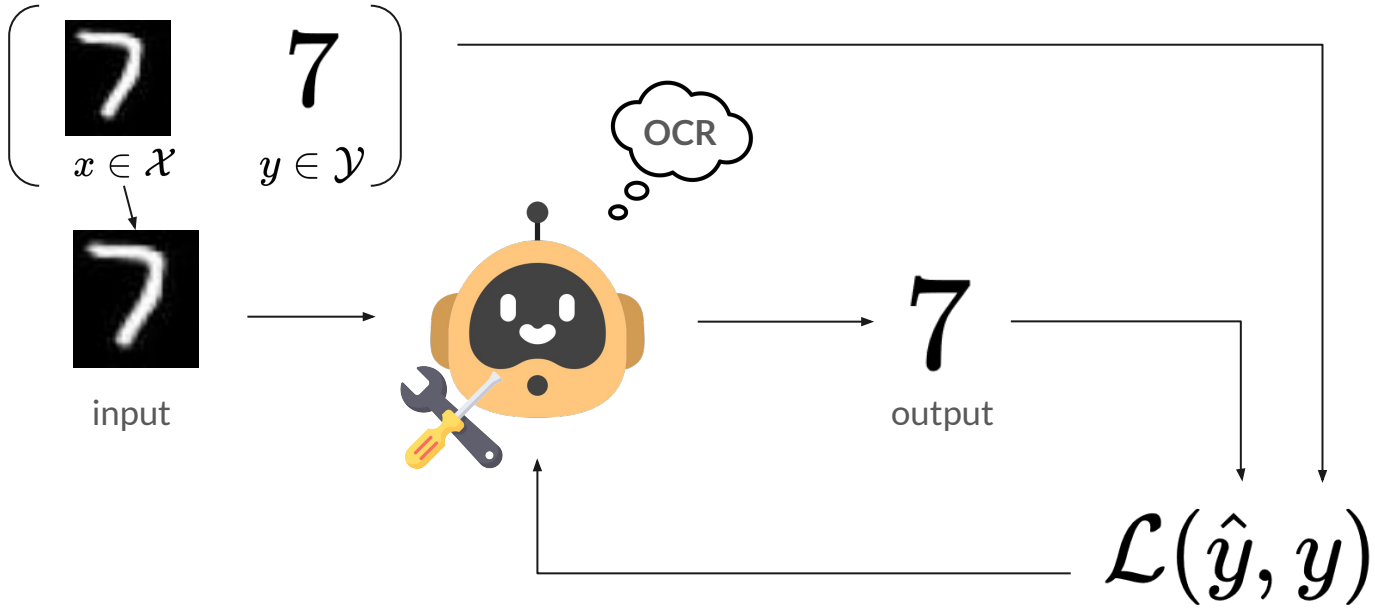
- Complex tasks
- Large datasets
- Automatic Feature Learning

Examples of Shallow Learning

Learning Algorithm

Learning Algorithm

- After selecting a proper model (e.g., linear model), we choose a proper **learning algorithm** 🛠️



Classification Models

- Many models for solving classification problems exist, each with its own learning algorithm.

- Many Shallow Learning Algorithms
  - Decision Trees
  - Naive Bayes
  - K-Nearest-Neighbor
  - Support Vector Machine

Decision Trees

Decision Trees

- One of the most known classification techniques
- They represents a set of rules by using a tree
- Often used for tabular data

Decision Trees

features | Classes

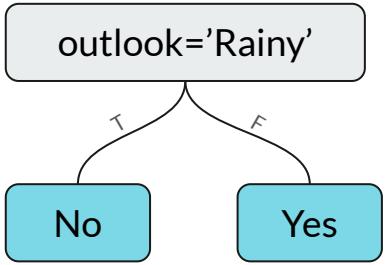
Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Decision Trees

- We perform a **test** on an attribute to discriminate between classes.
- Notably, we notice **that the outcome is often negative when the outlook is rainy**.

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

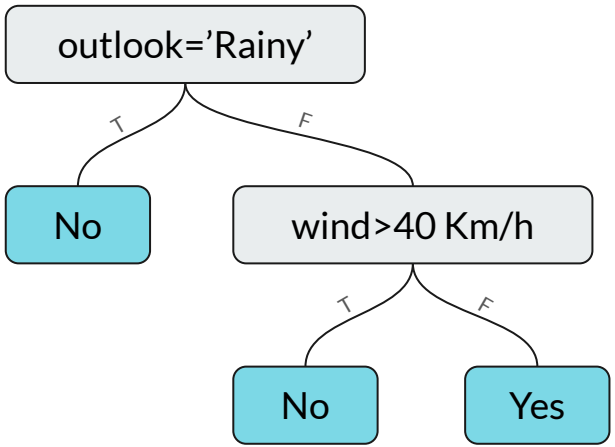
Decision Trees



- By doing so we make two errors
- We can correct them by adding more nodes to the tree

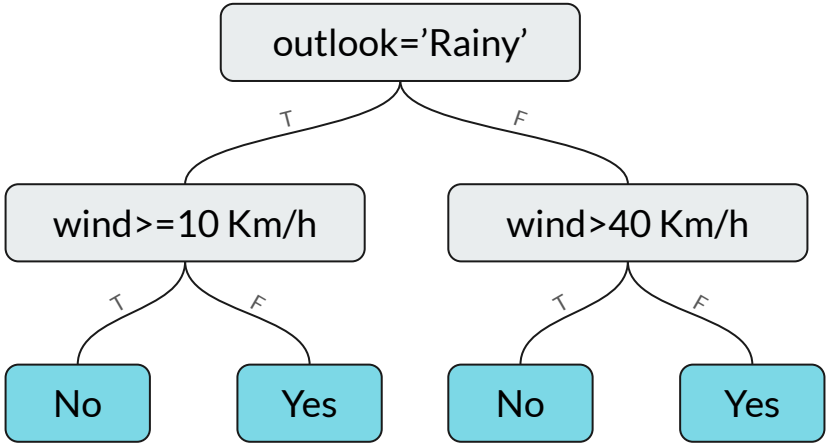
Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Decision Trees



Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

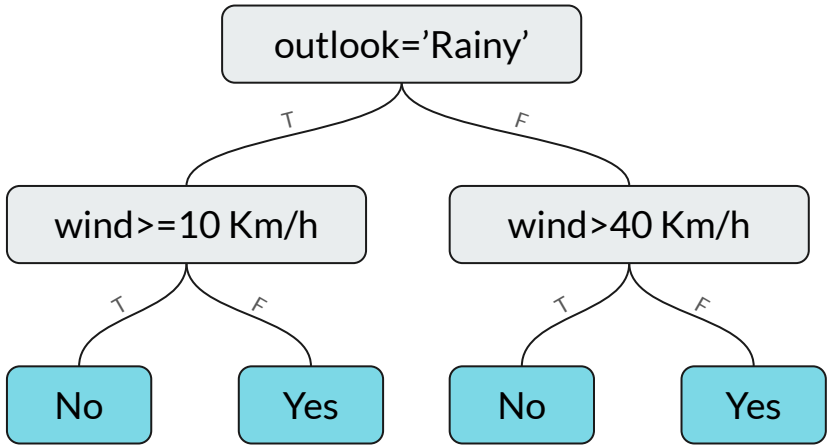
Decision Trees



Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Decision Trees

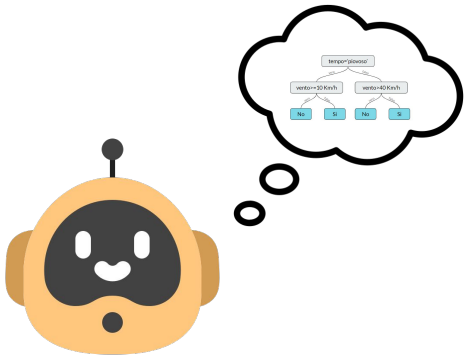
- **Inner nodes** represent **tests on the attributes**
- Every **edge** is a possible **answer to a test** (true or false)
- Each leaf represents a class
- Each **path** on the tree represent a **classification rule**



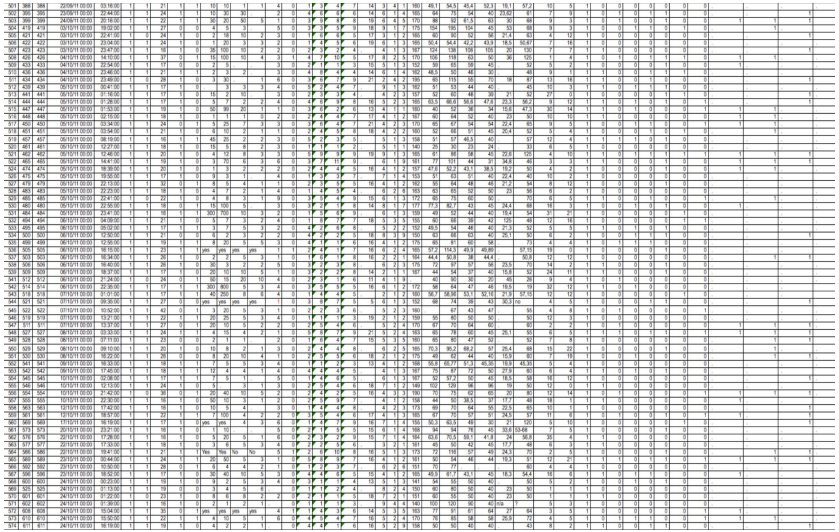
Induction of the Tree

- Finding a set of rules for constructing a tree may seem simple at first sight…

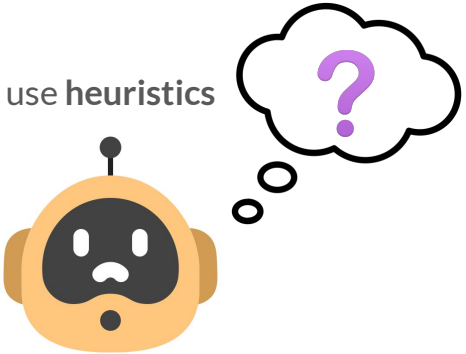
Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No



Induction of the Tree

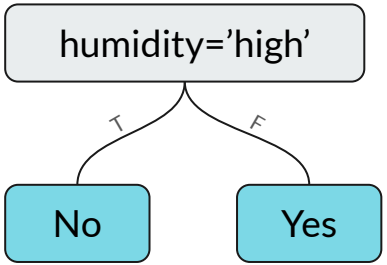


- Finding a set of rules for constructing a tree may seem simple at first sight…
- However, the set of possible trees **grows exponentially** with respect to the **number of features**
- For this reason, we need to use **heuristics**



Induction of the Tree

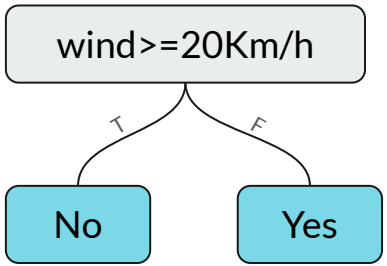
- If we discriminate between classes using the humidity attribute, we will obtain a suboptimal solution.



Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Induction of the Tree

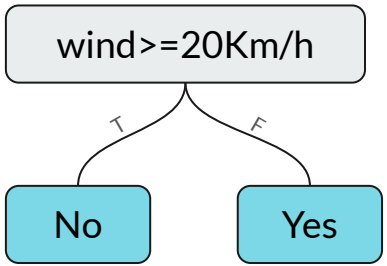
- If we discriminate between classes using the humidity attribute, we will obtain a suboptimal solution.
- But if we discriminate using the wind attribute we would obtain better results



Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Induction of the Tree

- If we discriminate between classes using the humidity attribute, we will obtain a suboptimal solution.
- But if we discriminate using the wind attribute we would obtain better results



Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

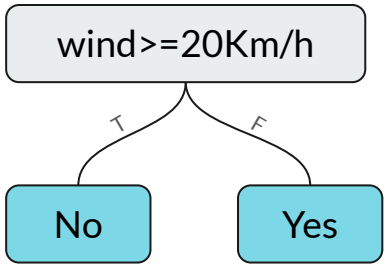
Choose as the root node the attribute that maximizes the separation between classes.

Induction Algorithms

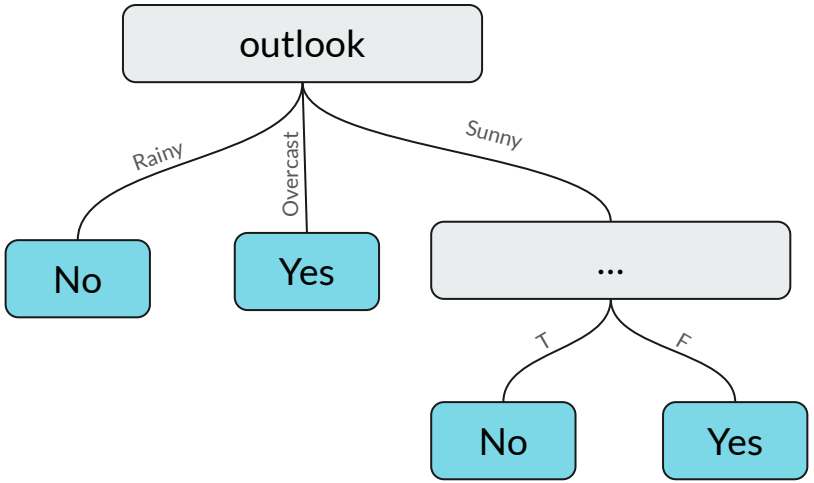
- Hunt
- Cart
- ID3 / C4.5
- SLIQ / SPRINT
- …

Types of splits

- Binary Split



- Multi-Way Split



Split Quality

Yes: 4 – No: 4

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Yes: 2 – No: 1

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes

Yes: 2 – No: 3

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Split Quality

Yes: 4 – No: 4

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No

Yes: 4 – No: 1

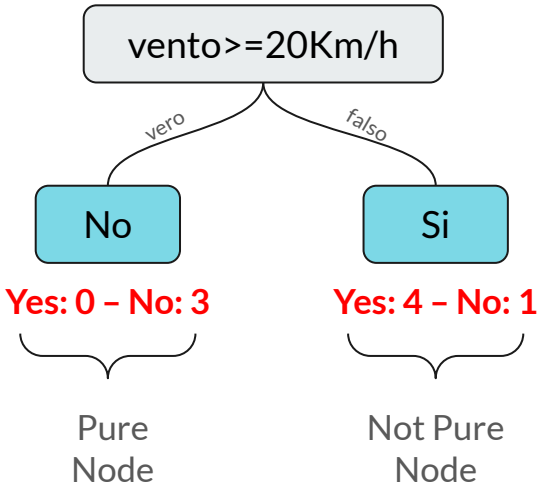
Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Sunny	23°	Normal	15 Km/h	Yes
Sunny	24°	Low	10 Km/h	Yes

Yes: 0 – No: 3

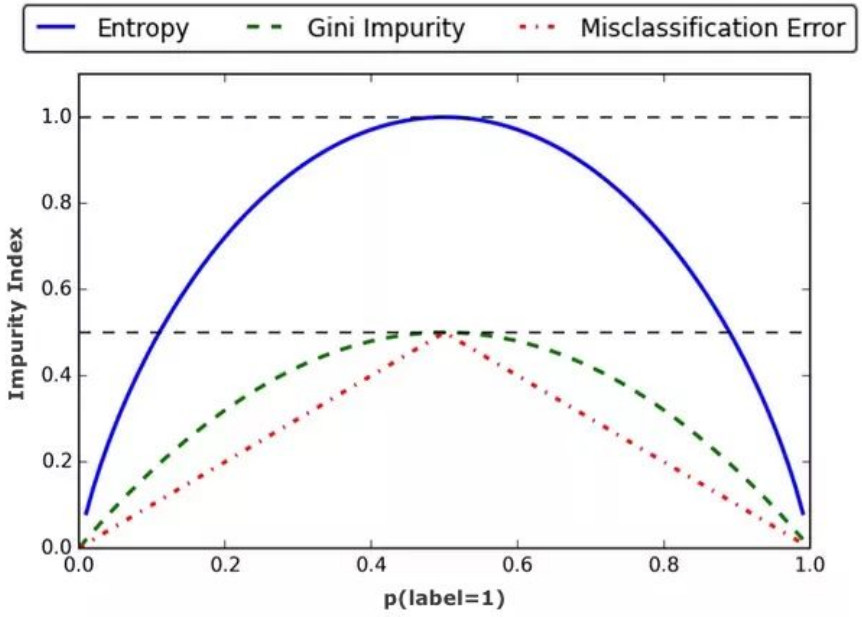
Outlook	Temperature	Humidity	Wind	Hike?
Rainy	15°	Normal	20 Km/h	No
Sunny	25°	Low	50 Km/h	No
Rainy	20°	Normal	45 Km/h	No

Purity of a Node

- **Maximum Purity**: all the records belong to the same class
- **Minimum Purity**: all the records are equally distributed
- Measures of impurity:
  - **Gini Index**
  - **Entropy**
  - **Classification error**
- To evaluate the quality of a split, one can compare the parent and the child node (using a so-called "**gain**").



Measures of Impurity

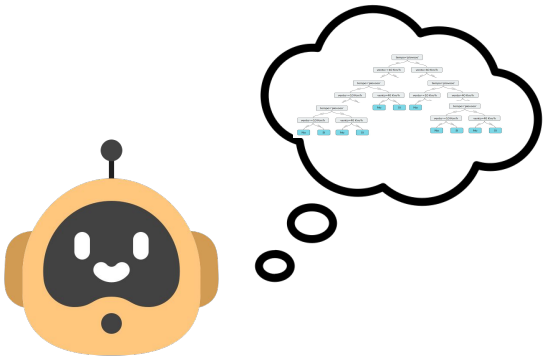


$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

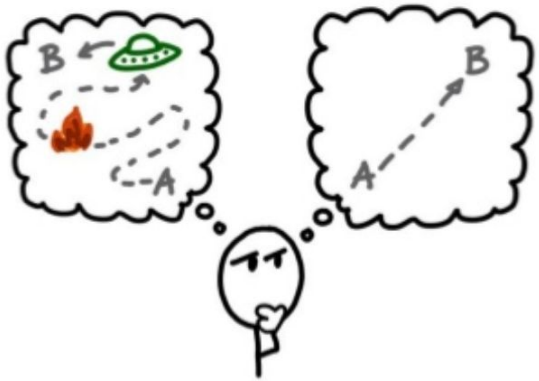
Possible Problems

- Impurity measures lead to many splits…
- Indeed, with many splits, one can eventually reach a state where every class is correctly discriminated against!



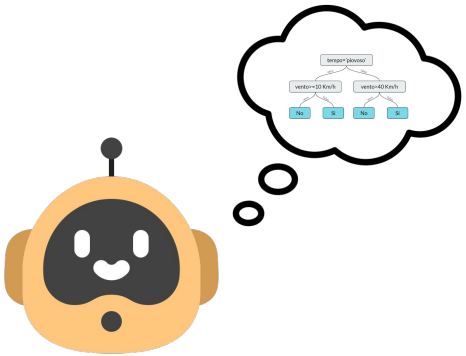
Occam's Razor

if you have two competing ideas to explain the same phenomenon, you should prefer the simpler one



Solutions for simpler trees

- Express everything in binary form
- Weight the number of splits with the **gain ratio**
- Early Termination



Bayesian Classifiers

Probability Theory

- What's the probability of obtaining a **face up 6** when rolling a die?

$P(X=6)=1/6$

- What's the probability of **drawing 1 ♦** in a deck of 52 cards?

$P(X=1\ \diamondsuit)=1/52$

Probability... Conditioned!

- What's the probability of obtaining a **face up 6** when rolling a die, **knowing the number is even**?

P(X=6 | X is even)=1/3

- What's the probability of **drawing 1 ♦** in a deck of 52 cards, **knowing the seed is red**?

P(X=1 ♦ | seed is red)=1/26

Probability - Main Idea for Machine Learning

- What's the probability of **assigning a specific class** given a set of features?

P(**Hike=Yes**|Overlook=Rainy, Temperature=18, Humidity=High, Wind=10Km/h)=???

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No

- Can we express the classes following probabilistic rules as we did for the die and the deck of cards?

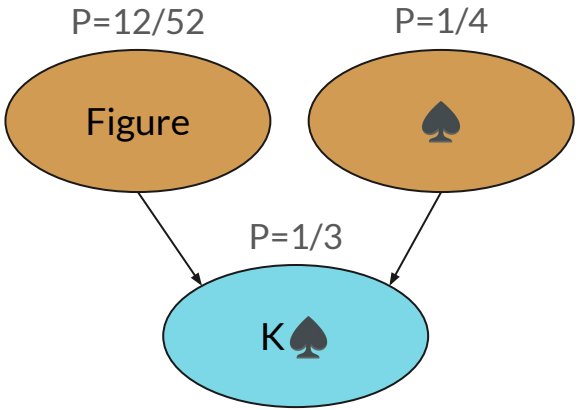
Bayesian Approach

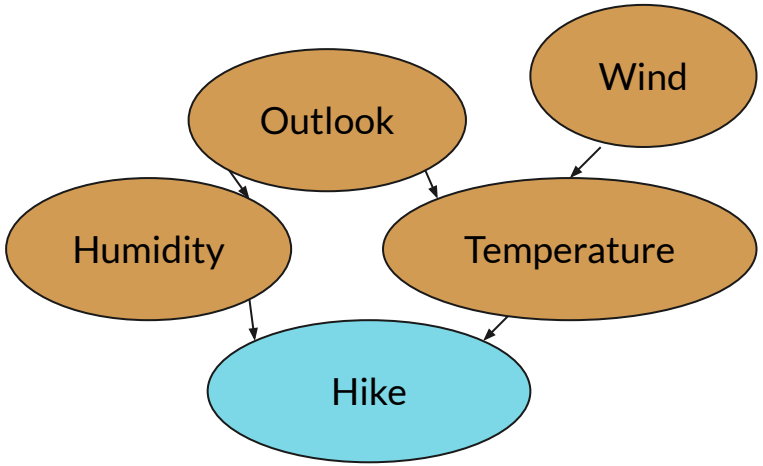
- We can define the probability of events without the need to measure them by using the rule of classic probability theory (starting from the Kolmogorov Axioms).
- Example: What is the probability of drawing a K knowing the seed is ♠?
- We do not need to repeat the experiment many times (**frequentist** approach), but we can simply use the rules of probability (**bayesian** approach).

$$P(N=K \mid S=\spadesuit) = P(N=K \text{ AND } S=\spadesuit) / P(S=\spadesuit)$$

Bayesian Networks

- What is the probability of **drawing a K♠** knowing that the **seed is ♠** and that it is a **figure**?

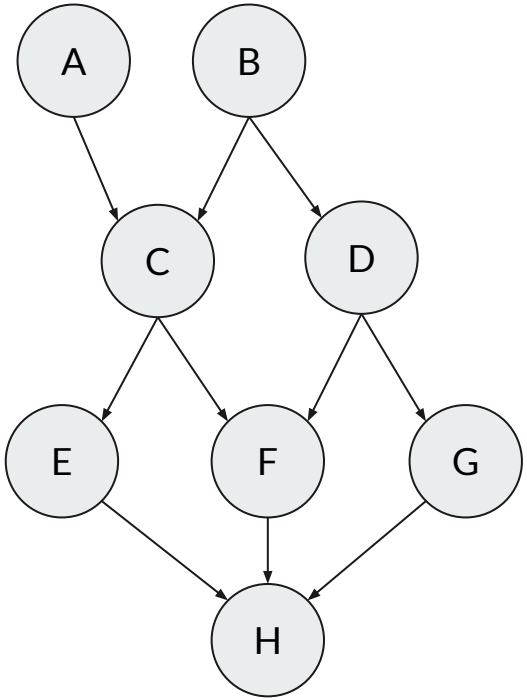




- What is the probability of a **specific class** given a **set of features**?

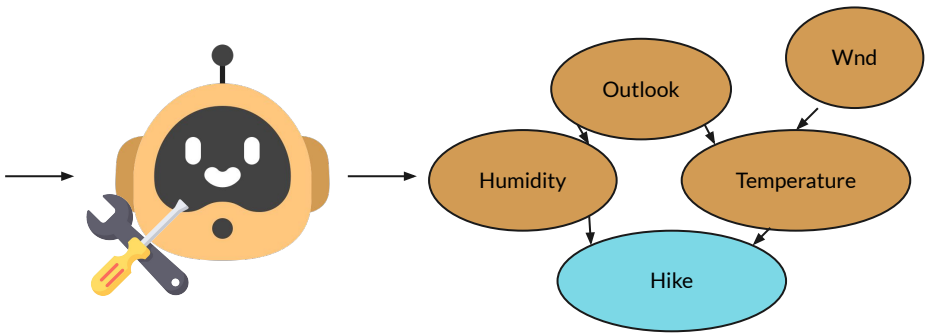
Bayesian Networks

- In the case of complex relationships, it is challenging to compute the joint probability
- From the rules of probability, we know we can simplify some dependencies: each node depends only on its parent nodes (**Local Markov Property**)



Building a Classifier with Bayesian Networks

Outlook	Temperature	Humidity	Wind	Hike?
Rainy	18°	High	10 Km/h	No
Rainy	20°	High	5 Km/h	Yes
Overcast	22°	High	12 Km/h	Yes
Rainy	15°	Normal	20 Km/h	No
Sunny	23°	Normal	15 Km/h	Yes
Sunny	25°	Low	50 Km/h	No
Sunny	24°	Low	10 Km/h	Yes
Rainy	20°	Normal	45 Km/h	No



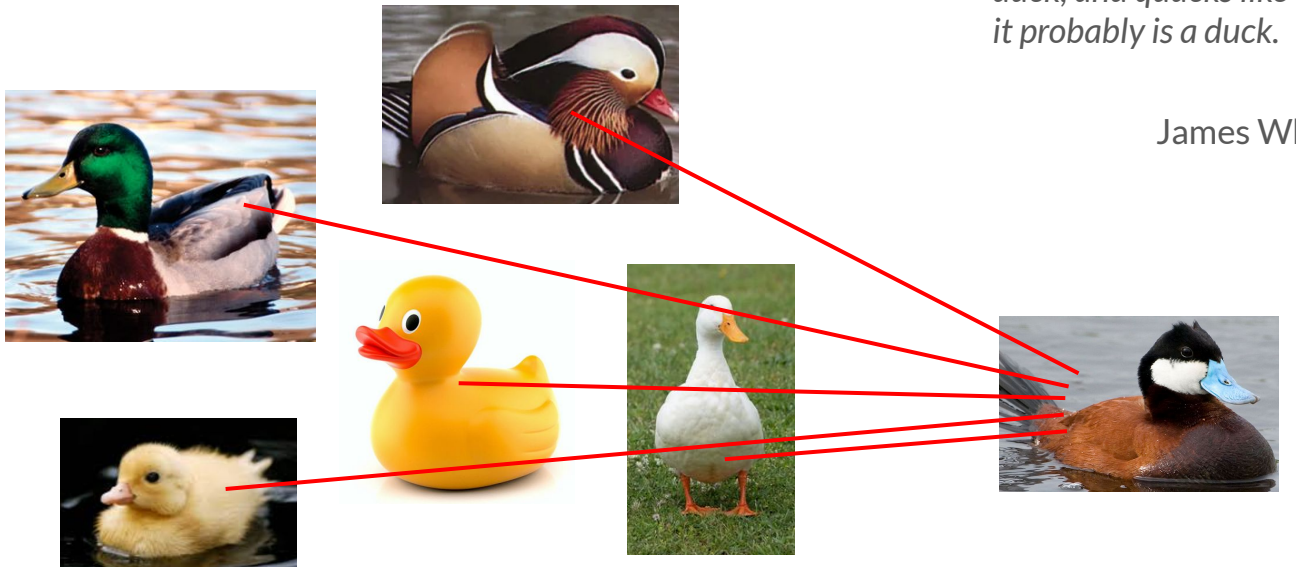
Naive Bayes

- During training we should:
  - Define a structure
  - Estimate the values of the probability table for each node
- We can simplify this problem by using a naive method: **Naive Bayes**
- **Assumption: All the attributes are conditionally independent given the class**
- We need to define the rule for estimating the probability on the attributes (based on their type) and on the classes, estimating them from the training set
- We need to apply the Bayes rule during inference, taking advantage of the attribute independence:

$$P(\text{class} \mid f1, f2, \ldots, fn) = P(f1, f2, \ldots, fn \mid \text{class})\, P(\text{class})$$

K-Nearest-Neighbor (KNN)

K-Nearest-Neighbor

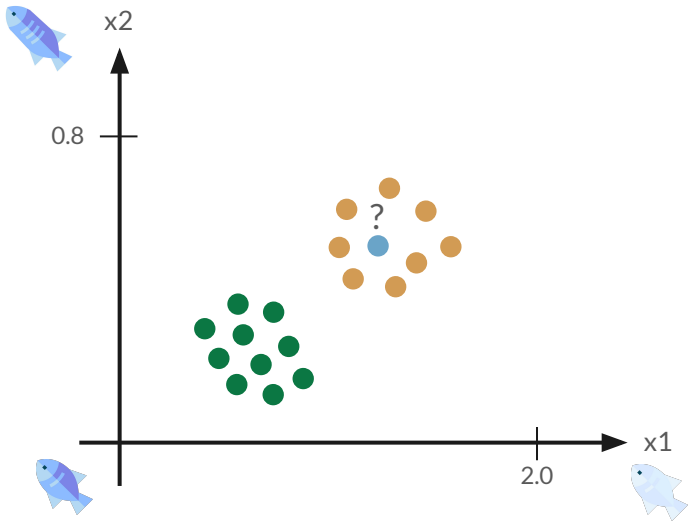


If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.

James Whitcomb Riley (1849–1916)

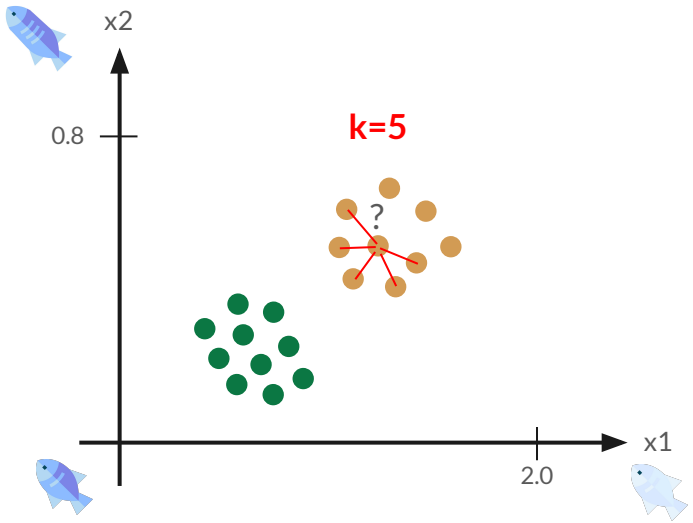
K-Nearest-Neighbor

- Il K-Nearest-Neighbor (KNN) is a classification algorithm built upon the instances
- It exploits the concept of “proximity” (“nearest neighbor”)



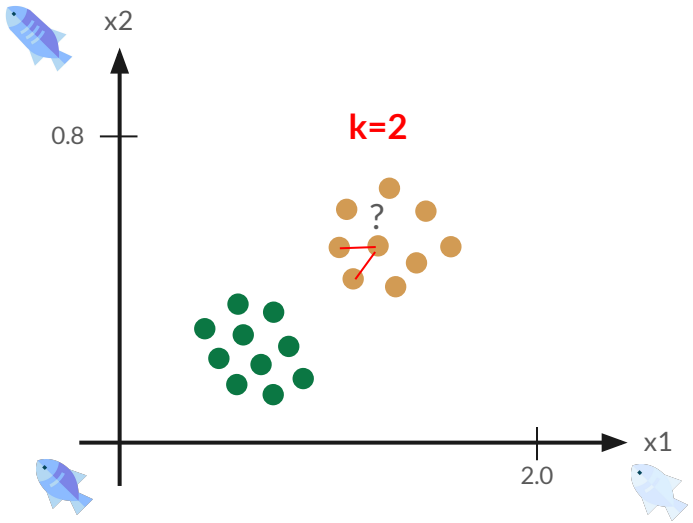
K-Nearest-Neighbor

- Il K-Nearest-Neighbor (KNN) is a classification algorithm built upon the instances
- It exploits the concept of “proximity” (“nearest neighbor”)
- Performs the classification on the base of the **k** nearest similar instances



K-Nearest-Neighbor

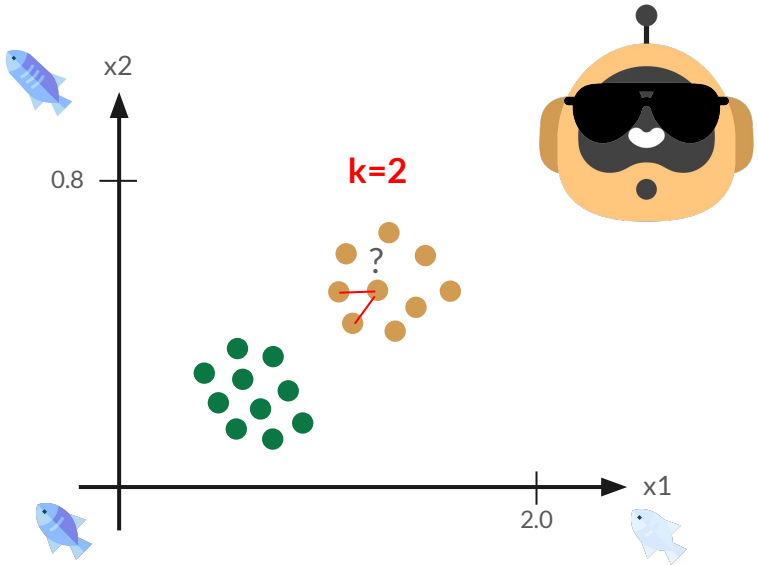
- Il K-Nearest-Neighbor (KNN) is a classification algorithm built upon the instances
- It exploits the concept of “proximity” (“nearest neighbor”)
- Performs the classification on the base of the **k** nearest similar instances



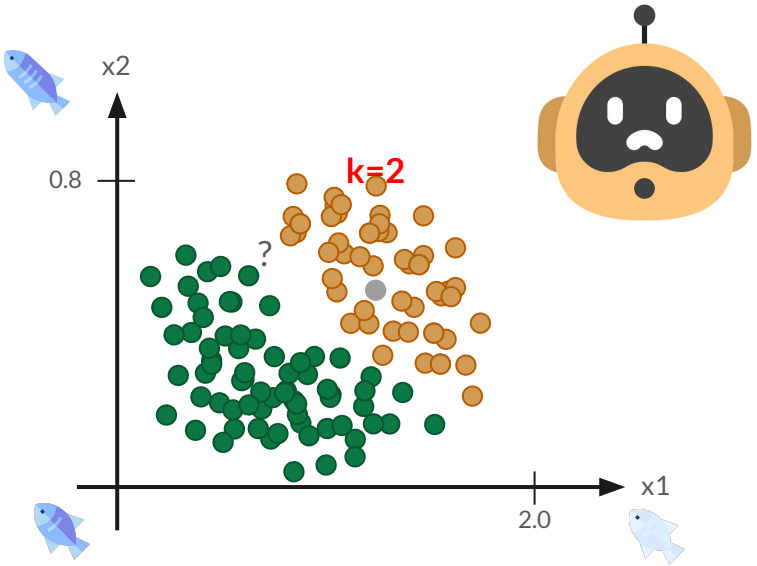
Lazy Learner 

- KNN **does not perform actual "learning,"** but it simply **memorizes the data**, which will later be used during the inference phase. For this reason, we call it a **"lazy" learner**.
- It is a non parametric model
- Even though the learning phase is fast (just the time for memorizing the information), **the inference phase increases when the training data increases**.

Lazy Learner



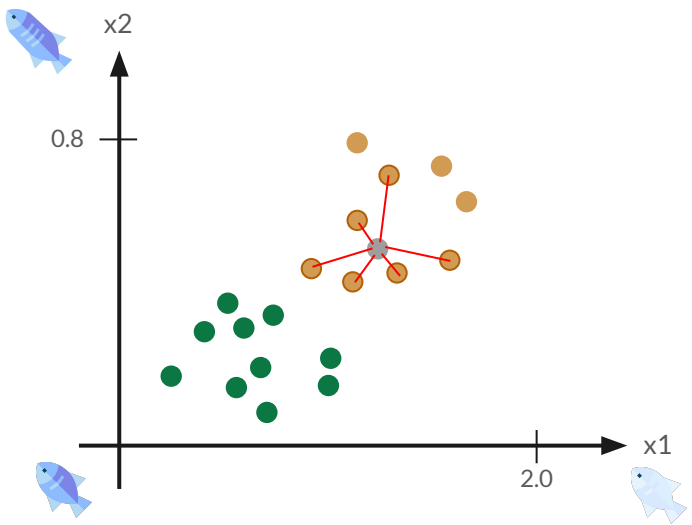
Number of controls before finding the k instances: **~19**



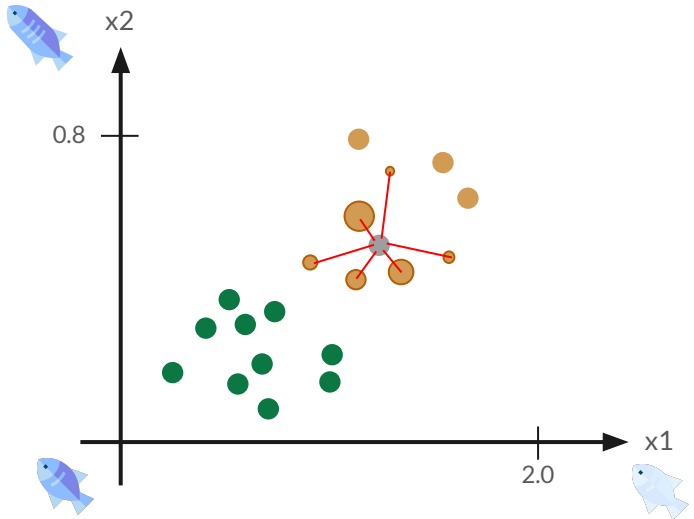
Number of controls before finding the k instances: **~200**

Weighting the Samples Contributions

Majority Voting



Distance Weight Voting



Support Vector Machines

Support Vector Machines

- Invented in the '60s by Vladimir Vapnik, and extended in '90s)
- Very powerful for handling **non linear** problems
- **Convex Optimization / Quadratic Programming**
- They represent a crucial state-of-the-art tool that exploits a lot of **strong mathematical interpretations**.

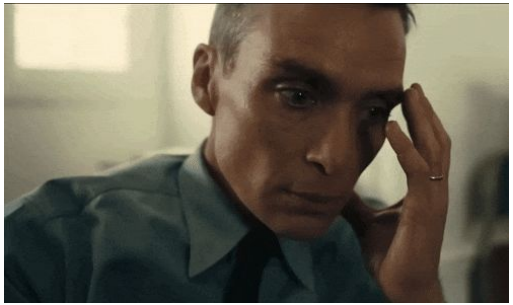
A lot of "strong" mathematical interpretations…

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i$$

$$\sum_{i=1}^{\ell} y_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}) + b = 0$$

$$\frac{\partial L_P}{\partial w_\nu} = w_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0 \qquad \frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \qquad \frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$\min_{\mathbf{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t. } y_i f(\mathbf{x}_i) \geq +1, \ \forall i$$



$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i \left[y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right]$$

$$\max_{\alpha_1,\ldots,\alpha_\ell} \min_{\mathbf{w},b} L(\alpha_1,\ldots,\alpha_\ell,\mathbf{w},b)$$
$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1,\ldots,N$$

$$\text{minimise} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad y_i f(\mathbf{x}_i) \geq 1, \quad i = 1,\ldots,N$$

$$\max_{\boldsymbol{\alpha}} \ \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$
$$\text{s.t.} \ \ 0 \leq \alpha_i \leq C, \ \forall i$$
$$\sum_i \alpha_i y_i = 0, \forall i$$

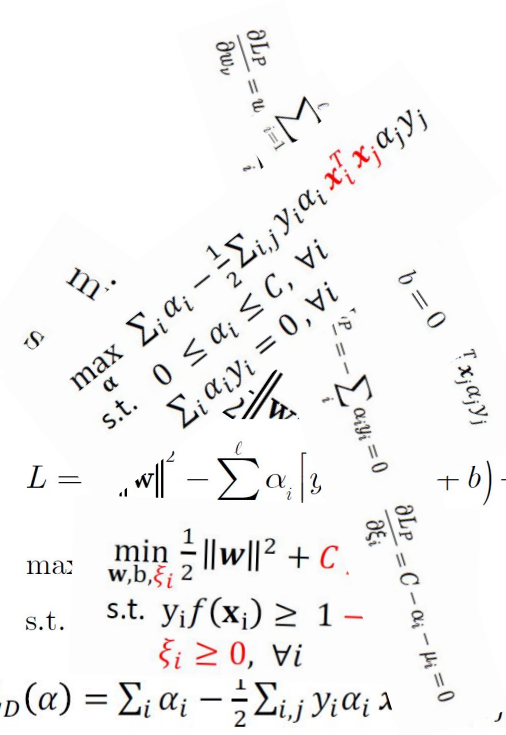
$$L_D(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{s.t. } y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \ \forall i$$
$$\xi_i \geq 0, \ \forall i$$

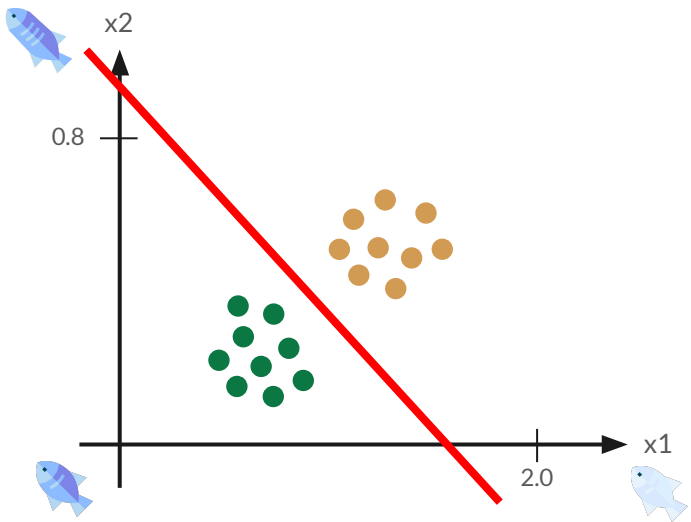
$$\max_{\boldsymbol{\alpha}} \ \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$
$$\text{s.t.} \ \ \alpha_i \geq 0, \ \forall i$$
$$\sum_i \alpha_i y_i = 0, \forall i$$

But “intuitive” as well!

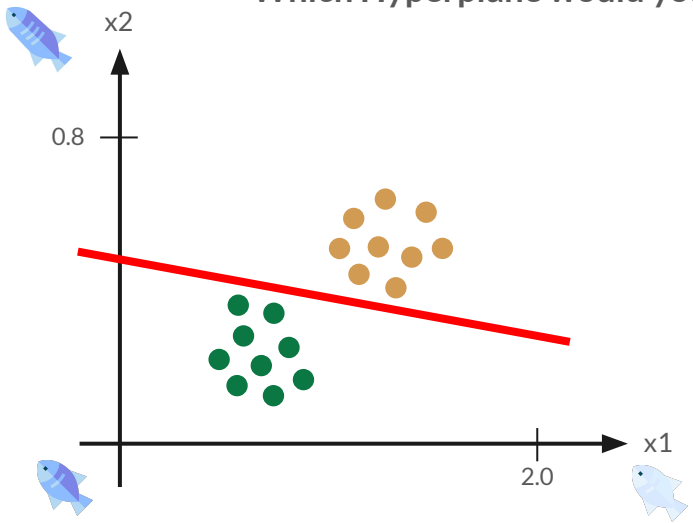
- In this seminar, we limit ourselves to understanding the main idea behind this elegant technique.
- Today, we can easily use already-implemented solutions! (i.e., Sci-Kit Learn)



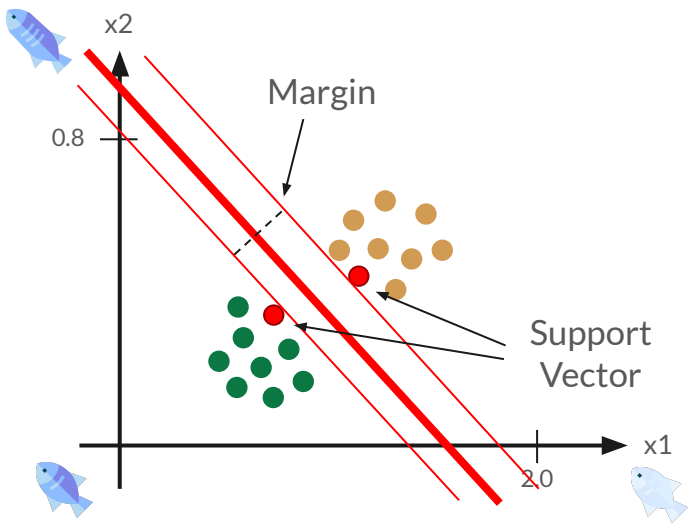
Support Vector Machines

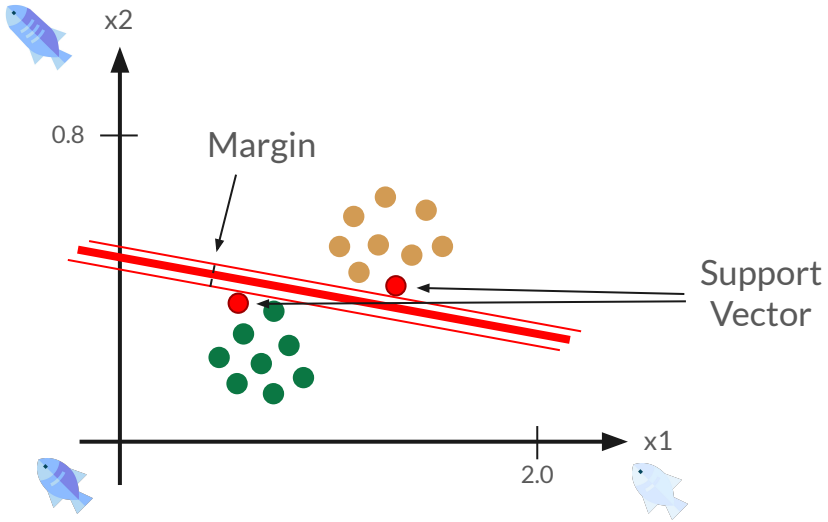


Which Hyperplane would you pick?



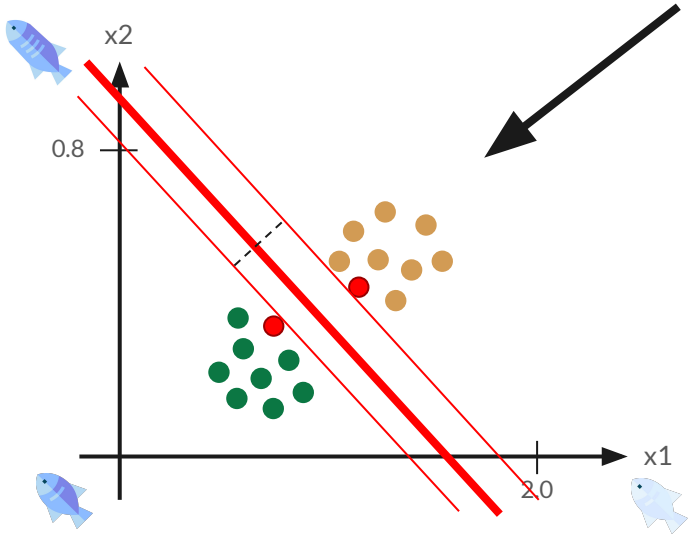
Support Vector Machines

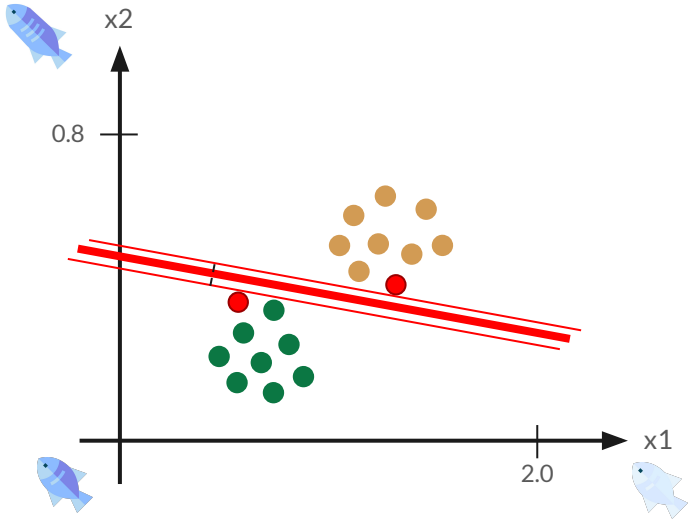




Support Vector Machines

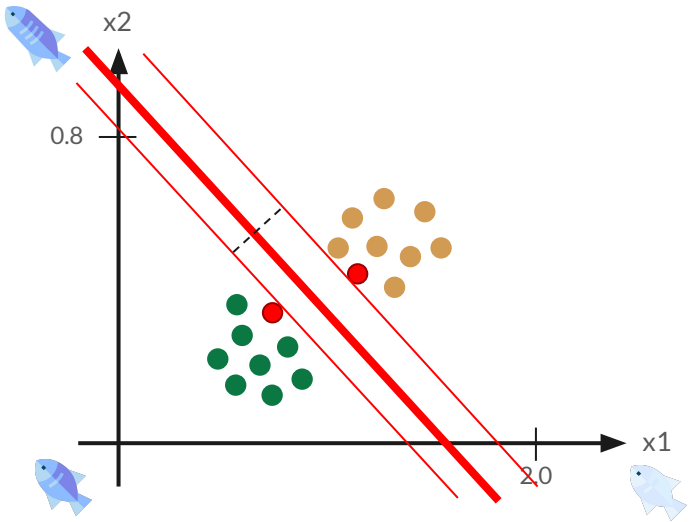
It is convenient to choose the hyperplane which maximizes the margin



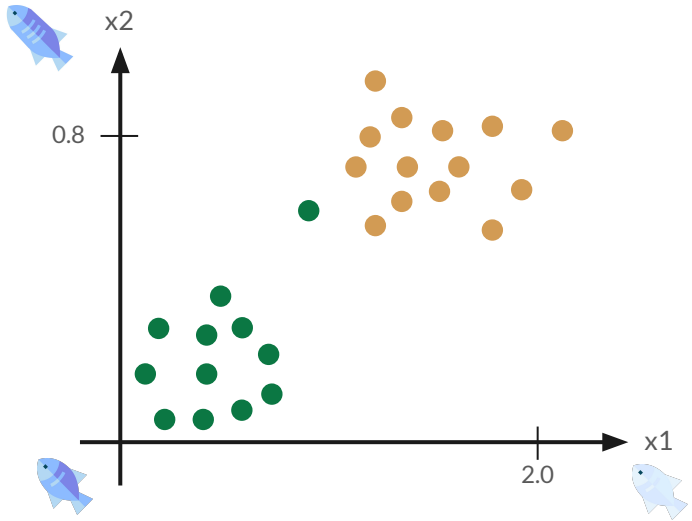


Support Vector Machines

- SVMs find the equations for the best hyperplane separating the two classes, which coincides with **solving a problem of convex optimization**
- Today, many automatic solvers can do that

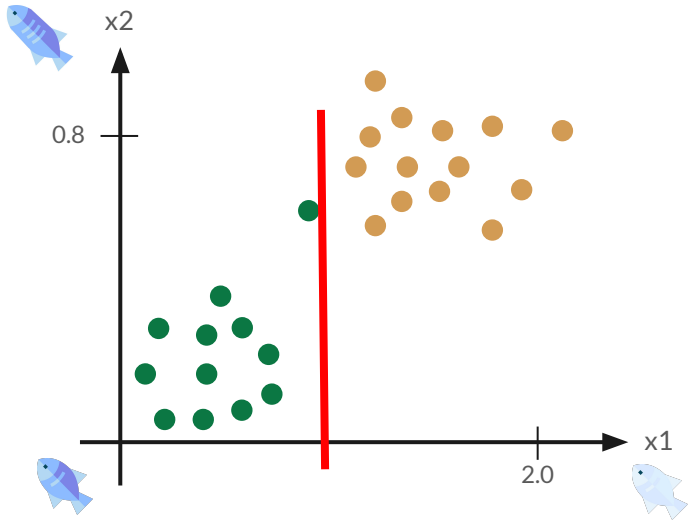


Support Vector Machines - Noisy Data



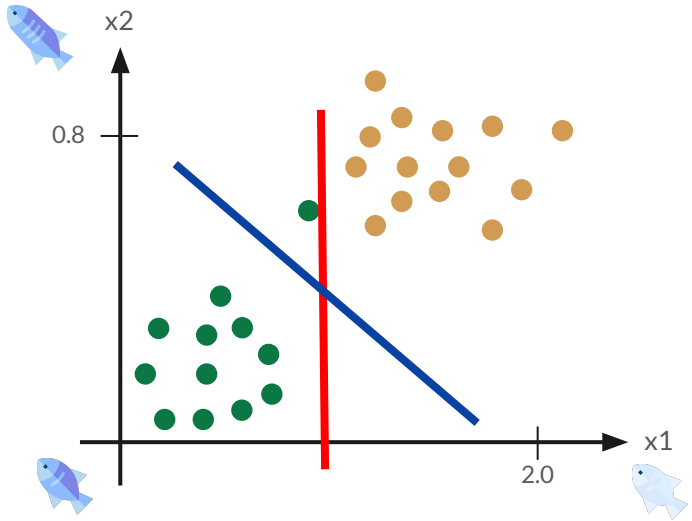
- Unfortunately, we have described a method that does not work very well with noisy data.

Support Vector Machines - Noisy Data



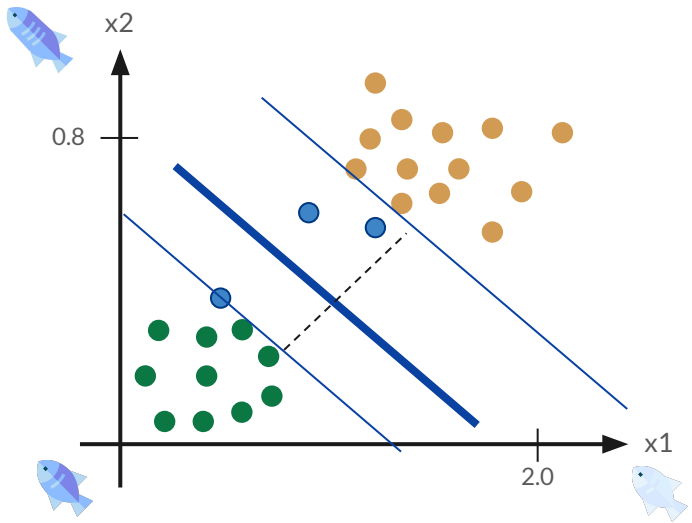
- Unfortunately, we have described a method that does not work very well with noisy data.
- In a similar situation, the algorithm would pick the **red hyperplane**…

Support Vector Machines - Noisy Data



- Unfortunately, we have described a method that does not work very well with noisy data.
- In a similar situation, the algorithm would pick the **red hyperplane**…
- … However, it is clearly a wrong solution caused by **noisy data**, and it would be better to select the **blue hyperplane**.

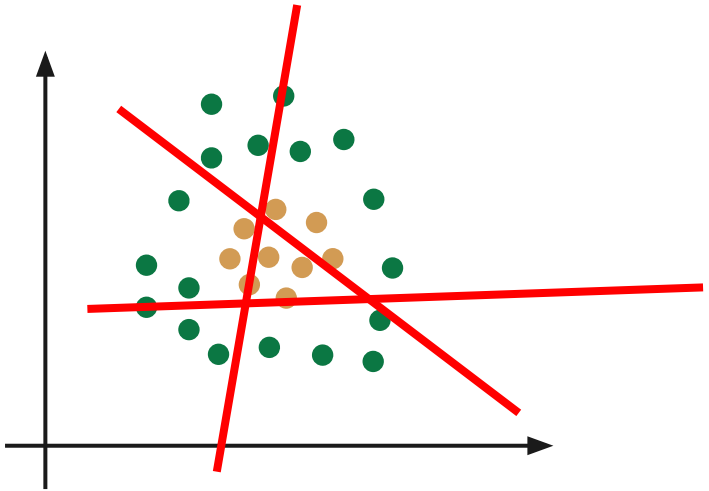
Support Vector Machines - Noisy Data



- Unfortunately, we have described a method that does not work very well with noisy data.
- In a similar situation, the algorithm would pick the **red hyperplane**….
- … However, it is clearly a wrong solution caused by **noisy data**, and it would be better to select the **blue hyperplane**.
- In this case, we can soften the condition by using a **margin of tolerance against a small portion of error (slack variables)**.

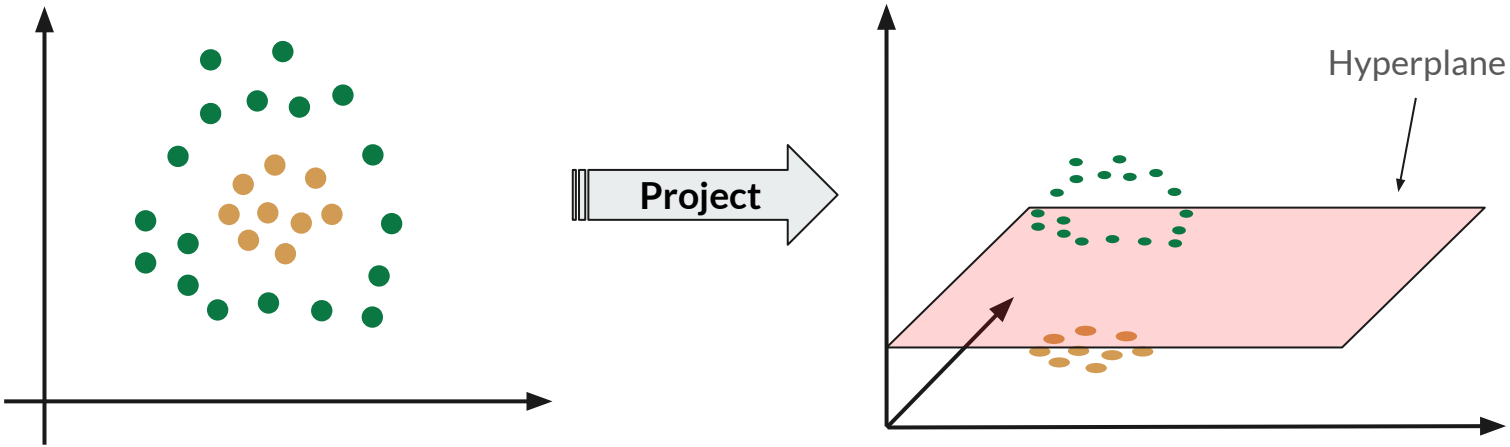
Support Vector Machines - Non Linearity

- We said SVMs are good for non linear problems
- But how can an hyperplane be useful?

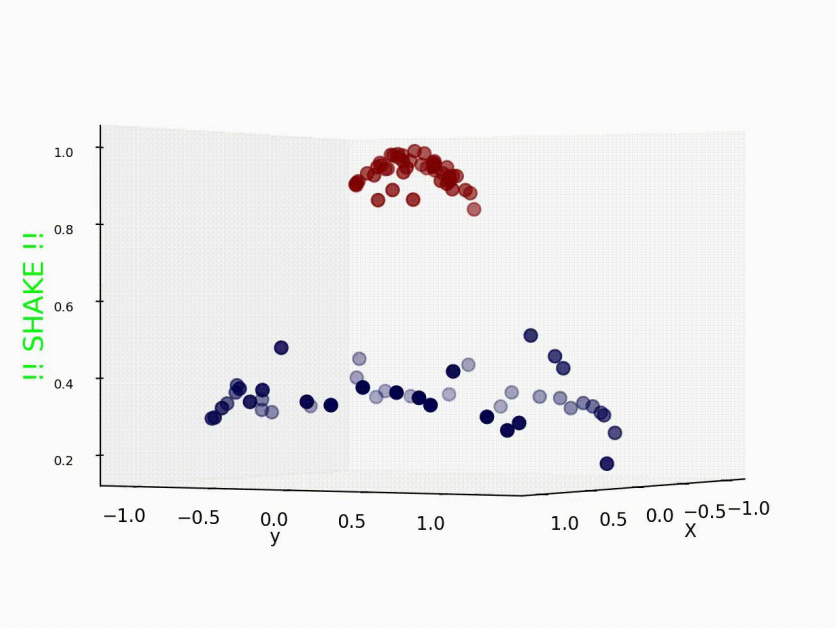


Kernel Trick

- Instead of discriminating the data on their "original space," we can **project** the data onto a space with a higher dimension **on which a discriminative hyperplane does exist.**



Kernel Trick

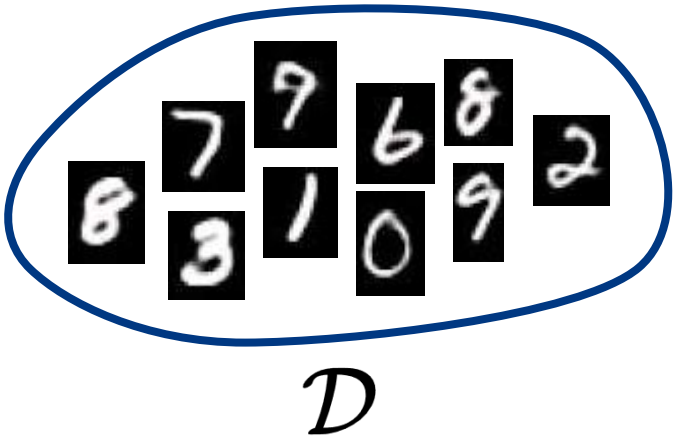


- In the original dimension, we could not separate the data linearly.
- But in a higher dimension, we can!
- There exist different types of **kernels** which let us project the data:
  - Polynomial
  - Radial Basis Function (RBF)

Validation e Overfitting

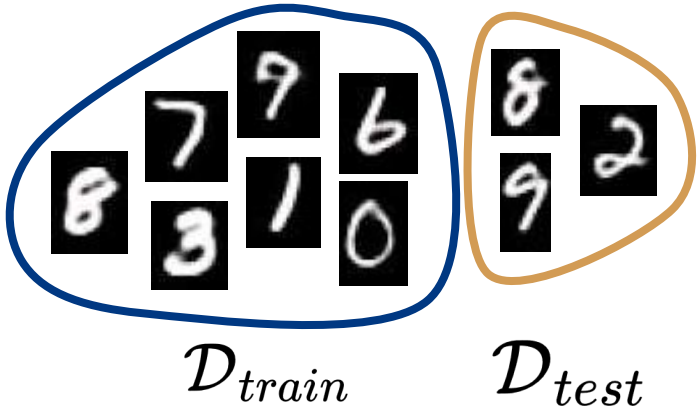
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"



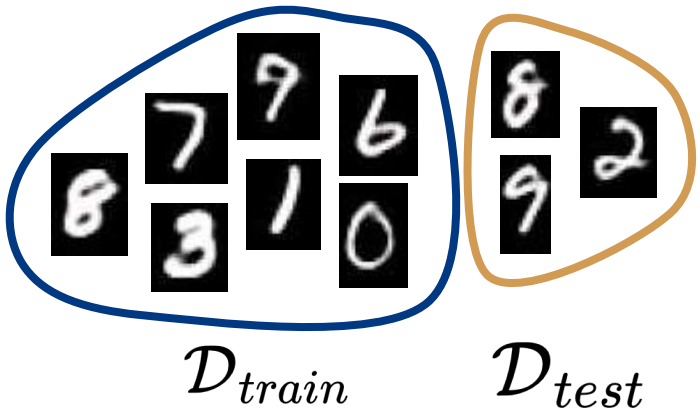
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"
- Accordingly, we divide the **data set into two partitions**, one dedicated to the training phase (**training set**) and one for evaluating the performances after training (**test set**).



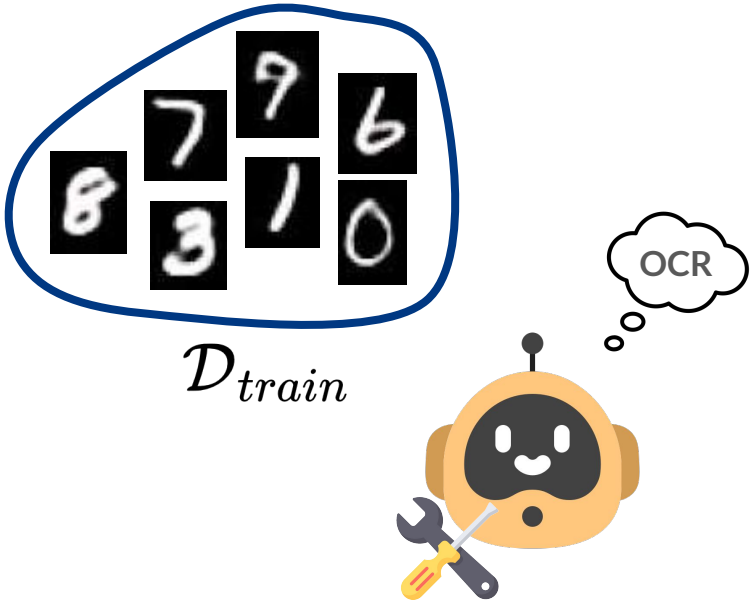
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"
- Accordingly, we divide the **data set into two partitions**, one dedicated to the training phase (**training set**) and one for evaluating the performances after training (**test set**).
- By doing so, we can obtain a classifier from a specific data set and evaluate it using separate data.



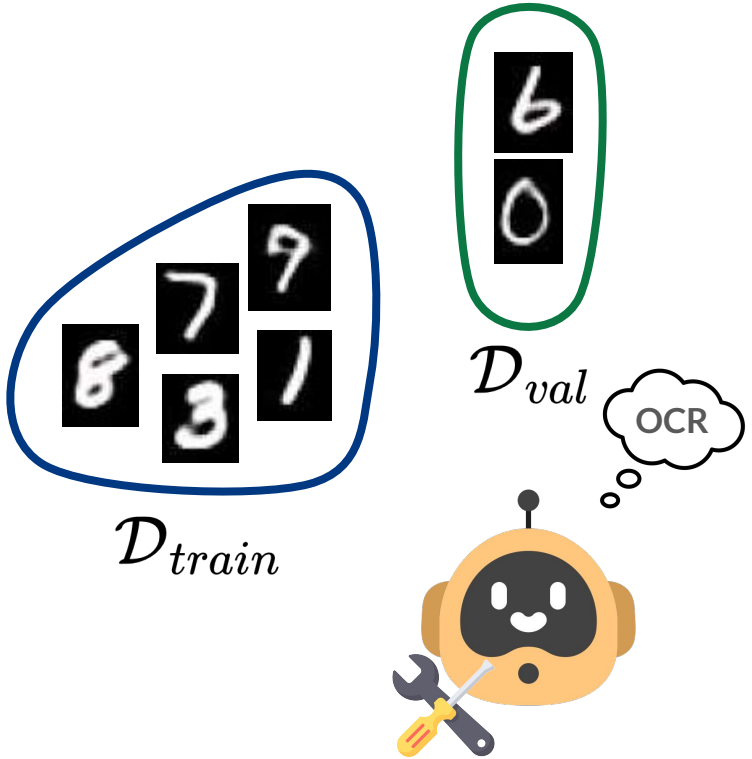
Learning Algorithm

- The learning algorithm can use only the data dedicated to the training phase

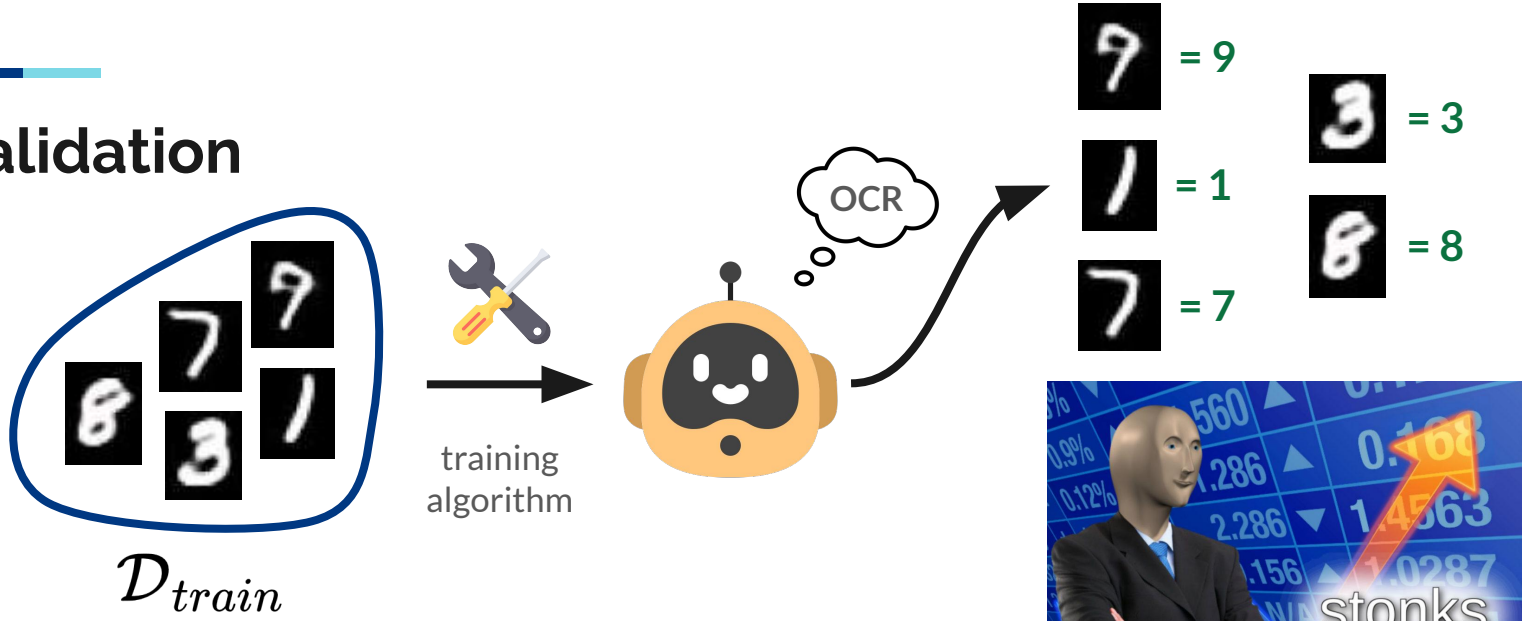


Validation

- The learning algorithm can use only the data dedicated to the training phase
- We can use a dedicated portion of the training set, called the "**validation set**," to **validate** the performances during the training.

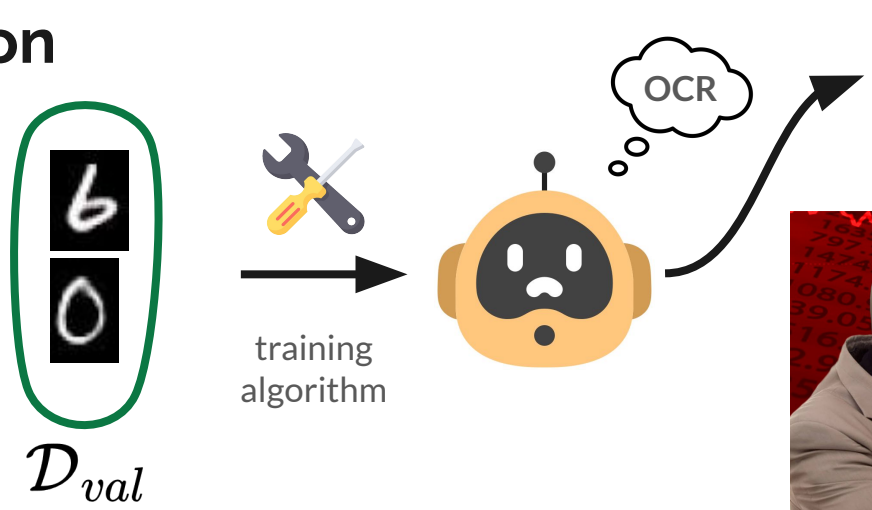


Validation



- Many classifiers can be pretty accurate on the training data
- But not so accurate with “new data”

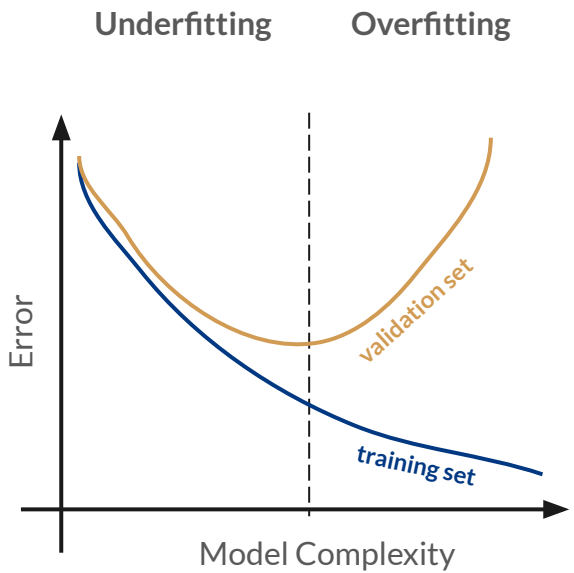
Validation



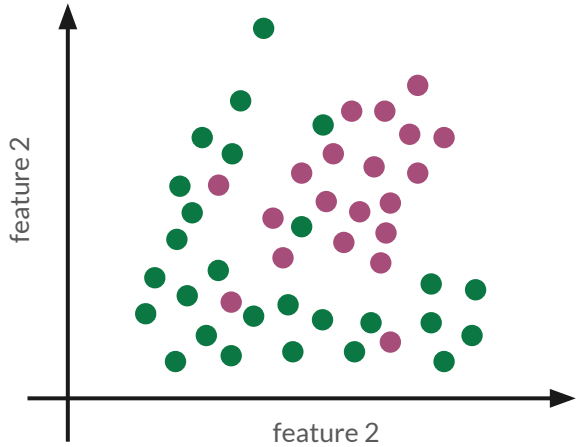
- Many classifiers can be pretty accurate on the training data
- But not so accurate with “new data”

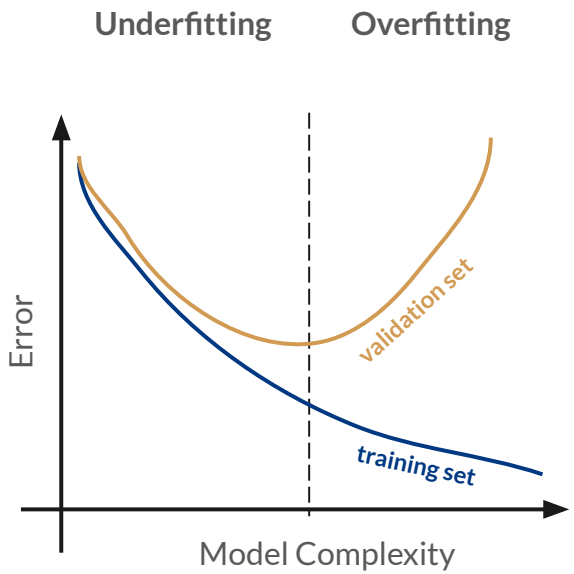
Overfitting

- A good model should be able to express the data (avoiding **underfitting**)
- At the same time, it should not be excessively complex to avoid **overfitting**
- Overfitting is one of the principal problems when using modern machine learning algorithms (we will see this in detail in day 3).

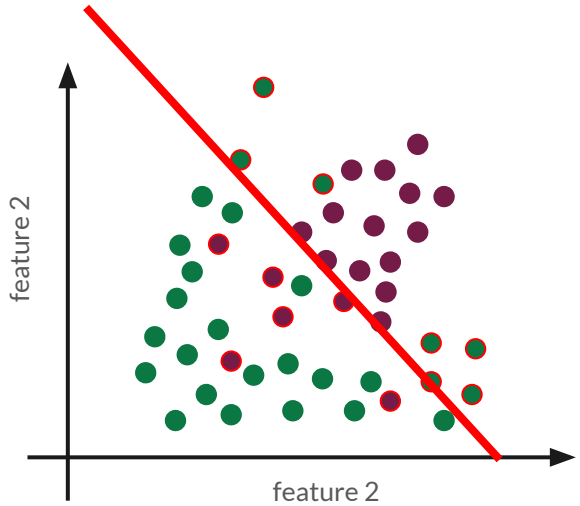


Overfitting

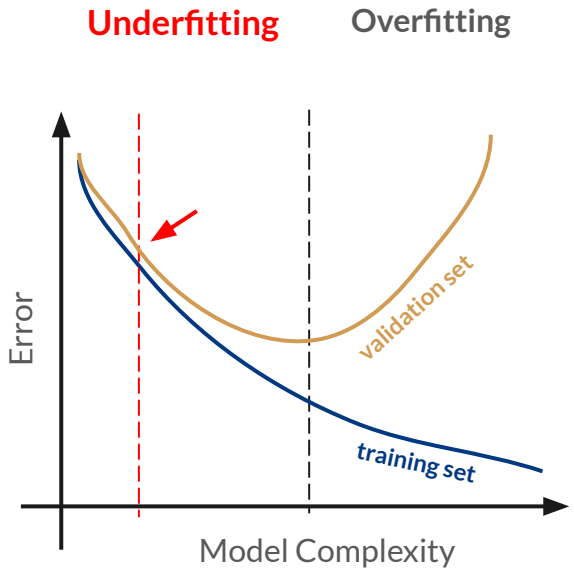




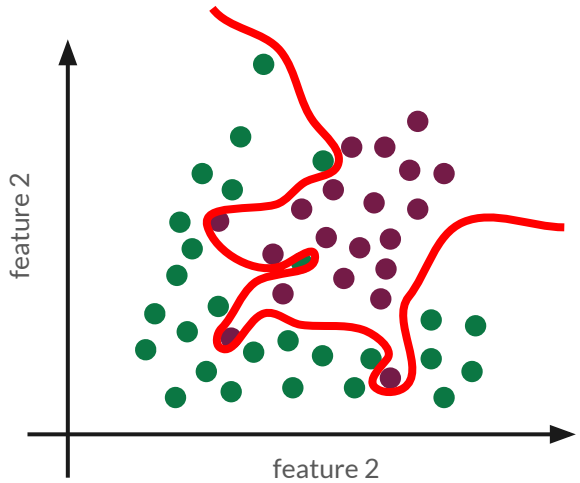
Overfitting



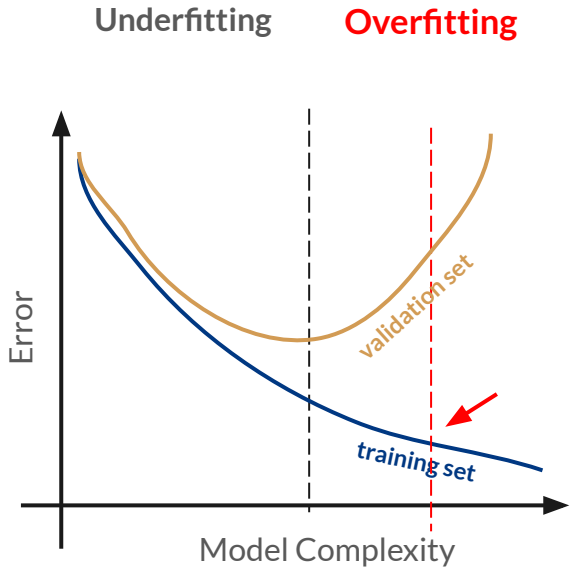
Errors on training set: 13



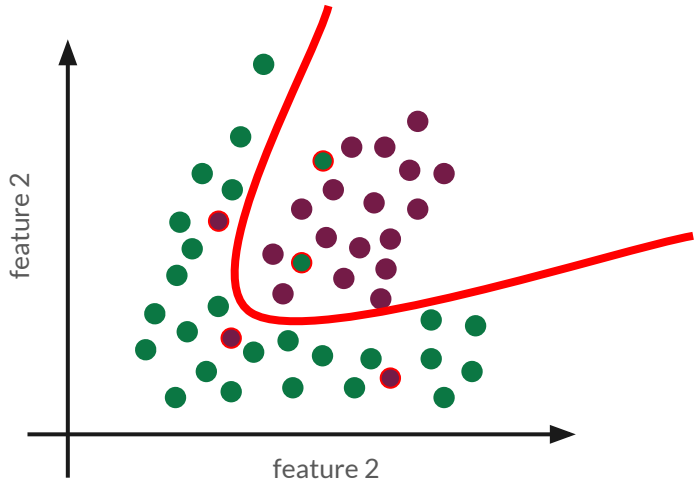
Overfitting



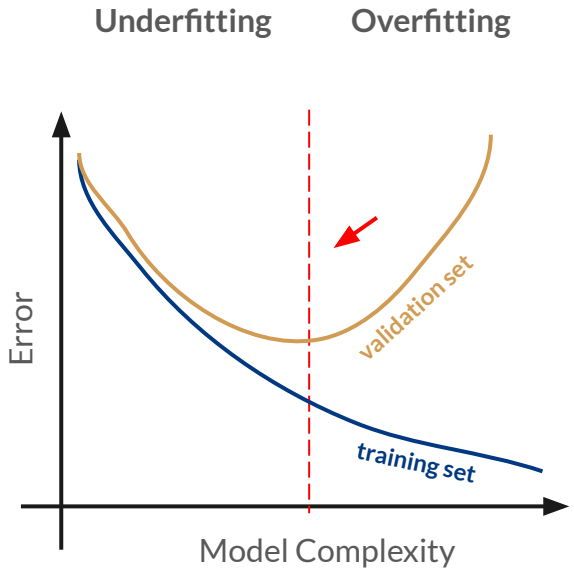
Errors on training set: 0



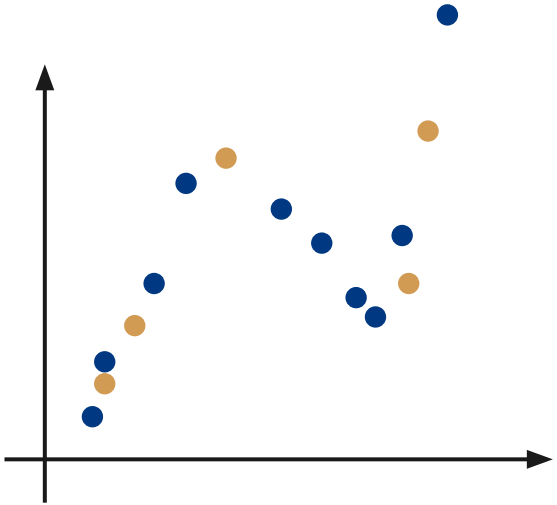
Overfitting

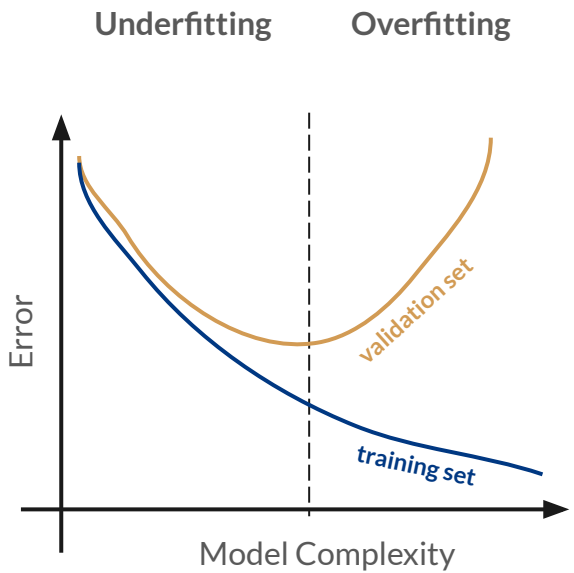


Errors on training set: 5

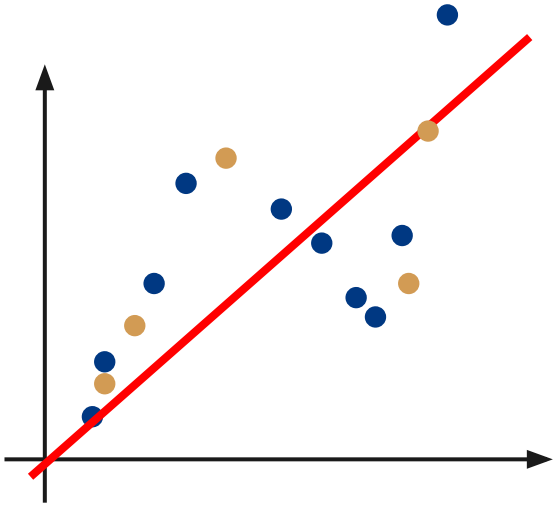


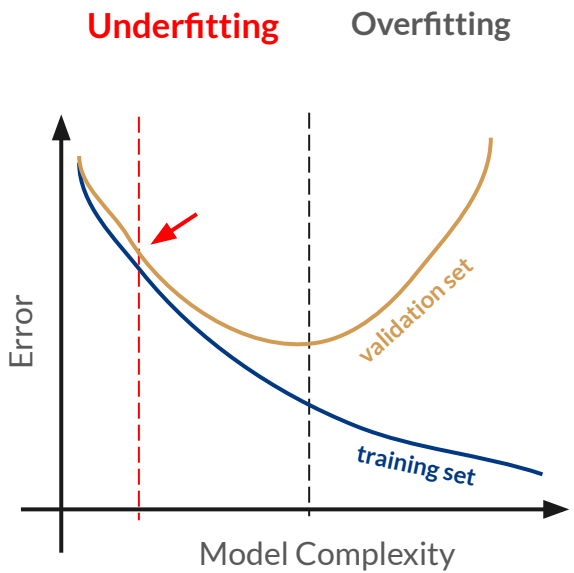
Overfitting (regression)



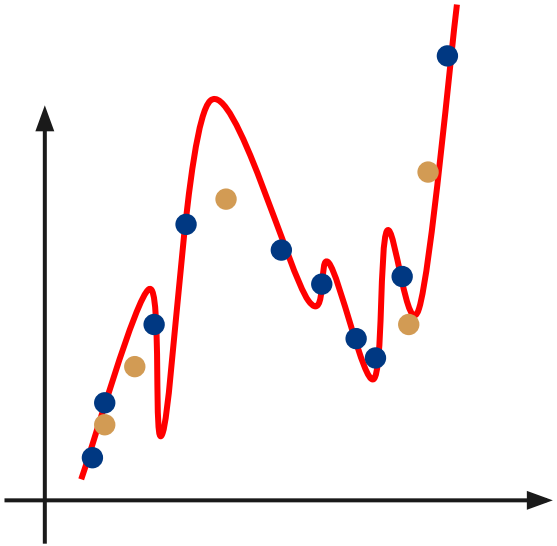


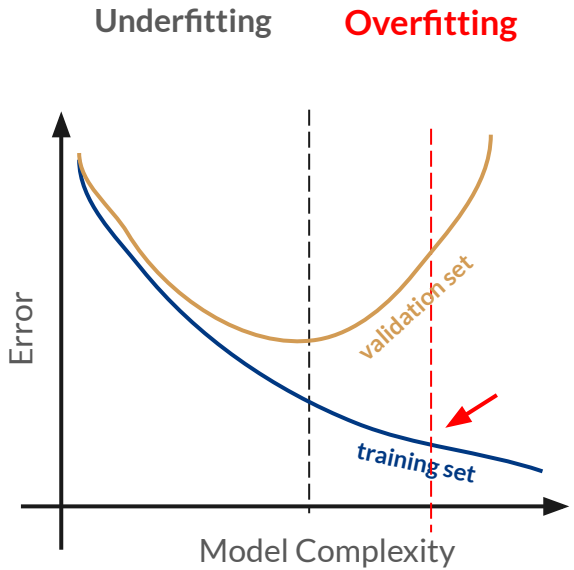
Overfitting (regression)



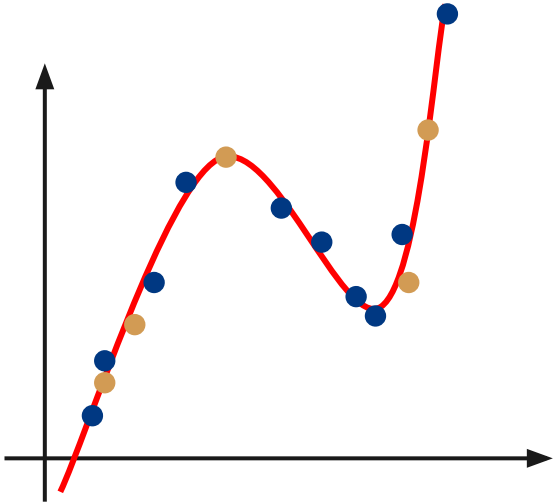


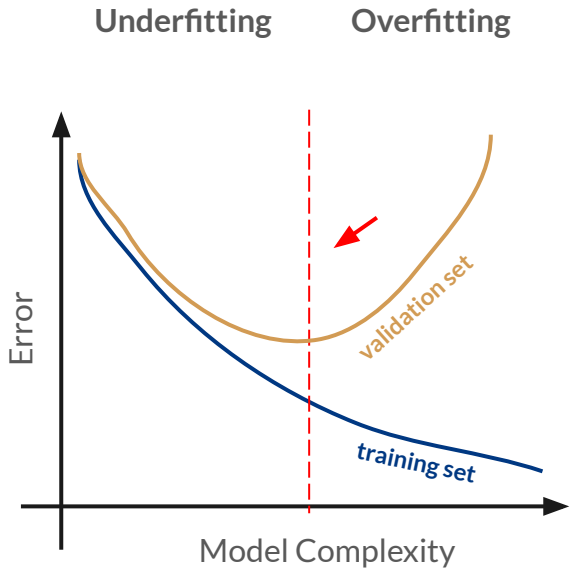
Overfitting (regression)



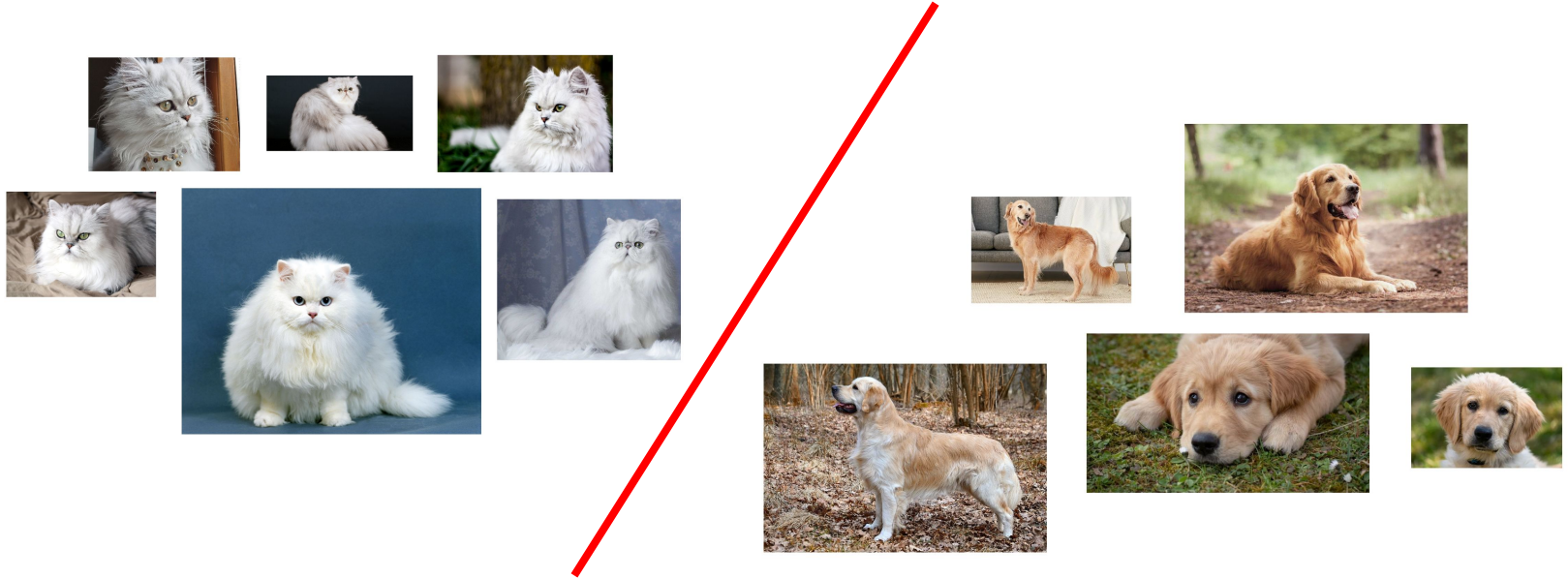


Overfitting (regression)

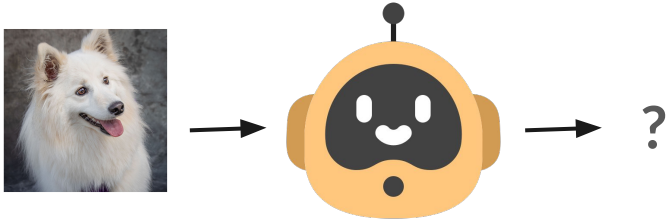




Generalization Capability



Generalization Capability



Generalization Capability