



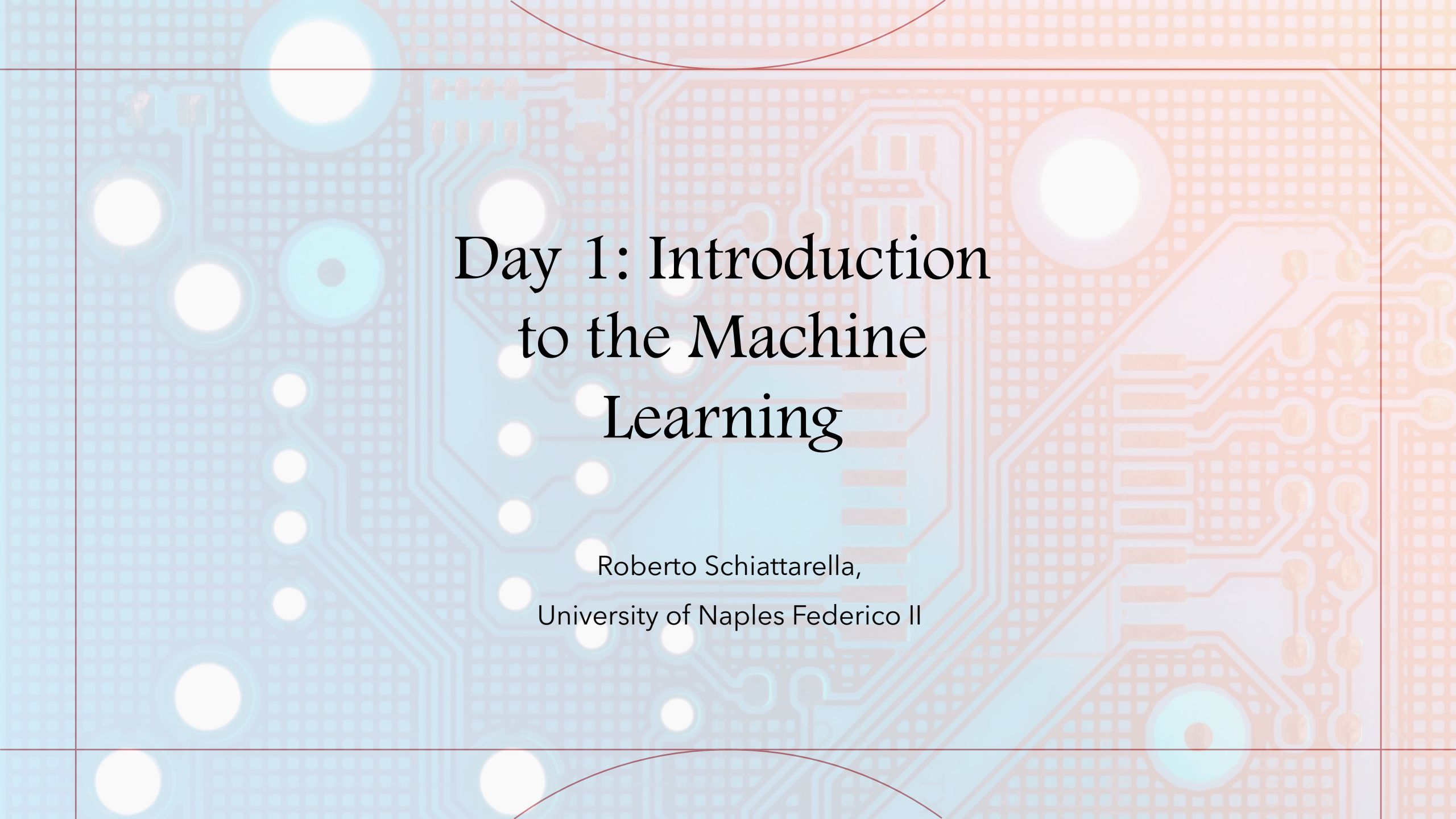
Corso di Formazione Nazionale  
INFN:  
“ Introduzioni alle Reti Neurali e  
Applicazioni sui Dispositivi  
Elettronici ”

# Module 2

- Day 1: Introduction to the Machine Learning
  - Day 2: Deep Neural Networks
  - Day 3: Convolutional Neural Network

Silvia Auricchio, Antimo Cagnotta, Francesco Carnevali, Francesco Cirotto, Roberto Schiattarella





# Day 1: Introduction to the Machine Learning

Roberto Schiattarella,  
University of Naples Federico II

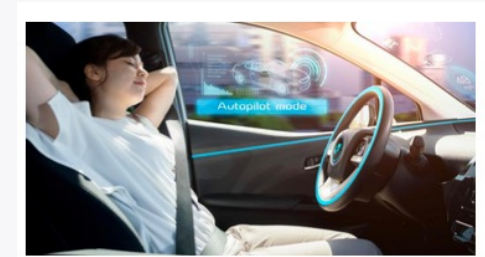
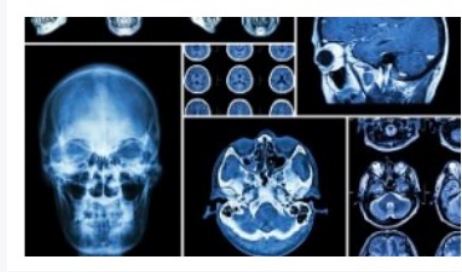
# Contents

- Introduction
- Machine Learning Tasks
- Supervised Learning
- McCulloch-Pitts Neuron
- Perceptron
- Logistic Regression
- Hands on: Data Preprocessing



# INTRODUCTION TO THE MACHINE LEARNING

- In this age of modern technology, there is one resource that we have in abundance: a **large amount** of structured and unstructured **data**.
- In the second half of the twentieth century, machine learning evolved as a **subfield of artificial intelligence** that involved the development of self-learning algorithms to gain knowledge from that data in order to make predictions.
- Several **everyday** life **applications**



# GENERAL PROBLEM

Input Data

```
graph TD; A[Input Data] --> B[Model]; B --> C[Output];
```

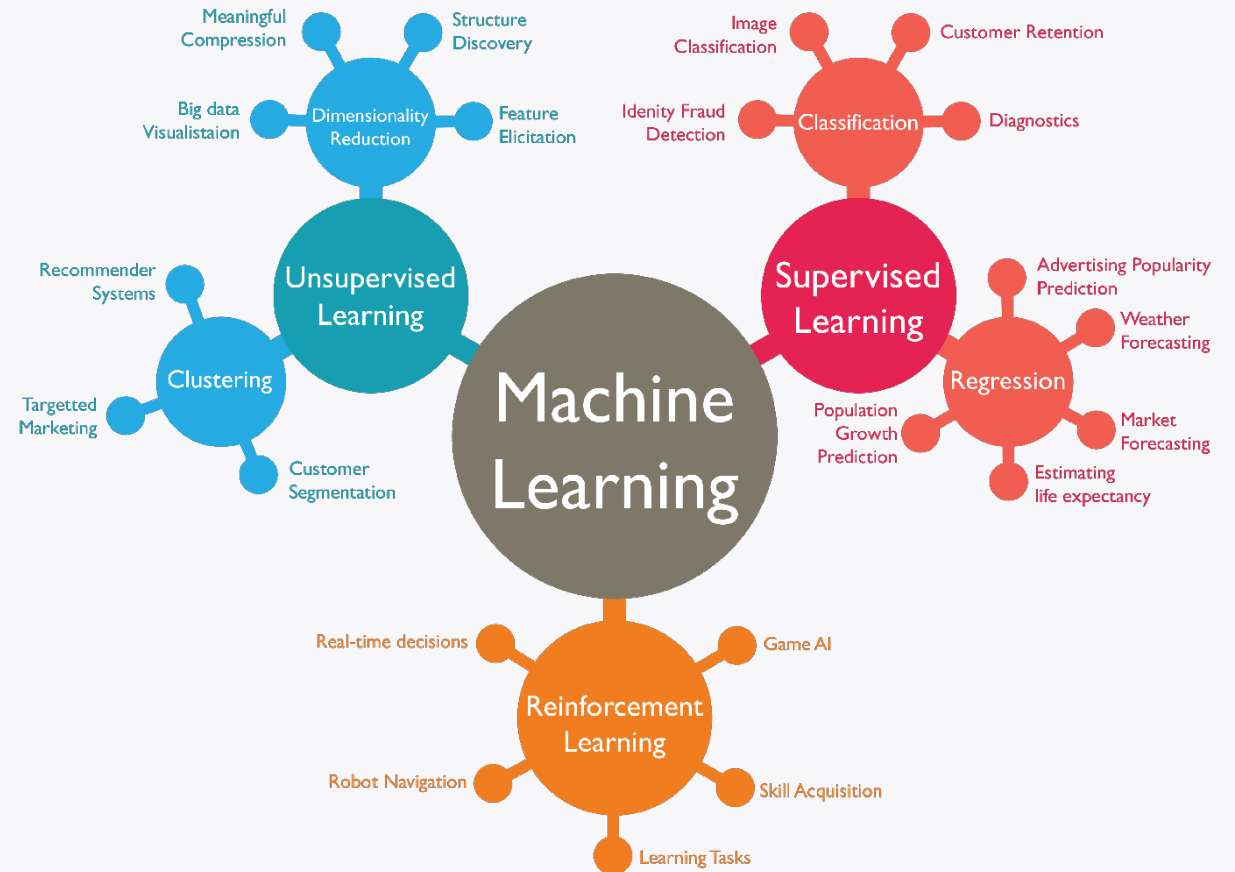
The diagram illustrates a three-step process. It begins with a box labeled 'Input Data' in a dark olive green color. A downward-pointing arrow connects this box to a second box labeled 'Model' in a medium brown color. Another downward-pointing arrow connects the 'Model' box to a final box labeled 'Output' in a reddish-brown color. The boxes are arranged vertically and are connected by arrows pointing downwards, indicating a sequential flow.

Model

Output

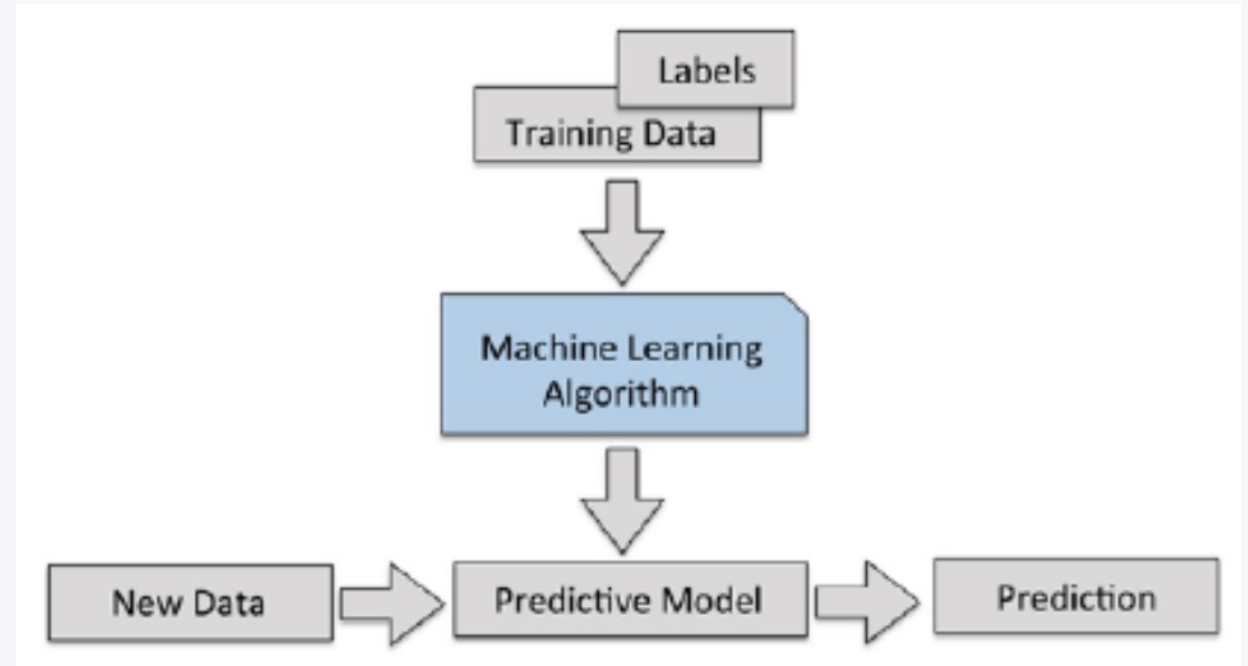
# Machine Learning Tasks

- Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.



# Supervised Learning

- The main goal in supervised learning is to learn a model from **labeled** training data that allows us to make predictions about unseen or future data.



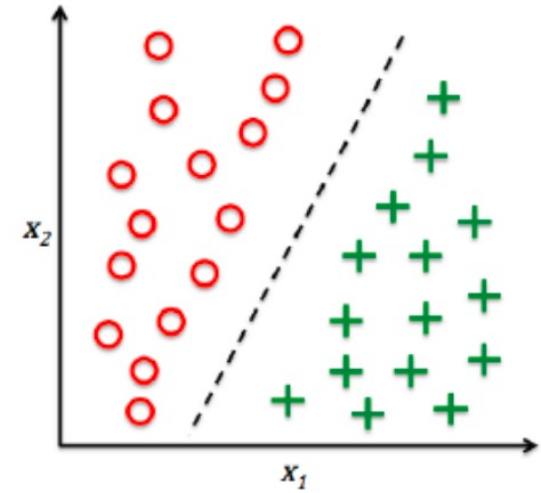


# Supervised Learning: Classification

The goal of the classification is to predict the **categorical class labels** of new instances based on past observations.

Email Spam:  
**binary**  
classification

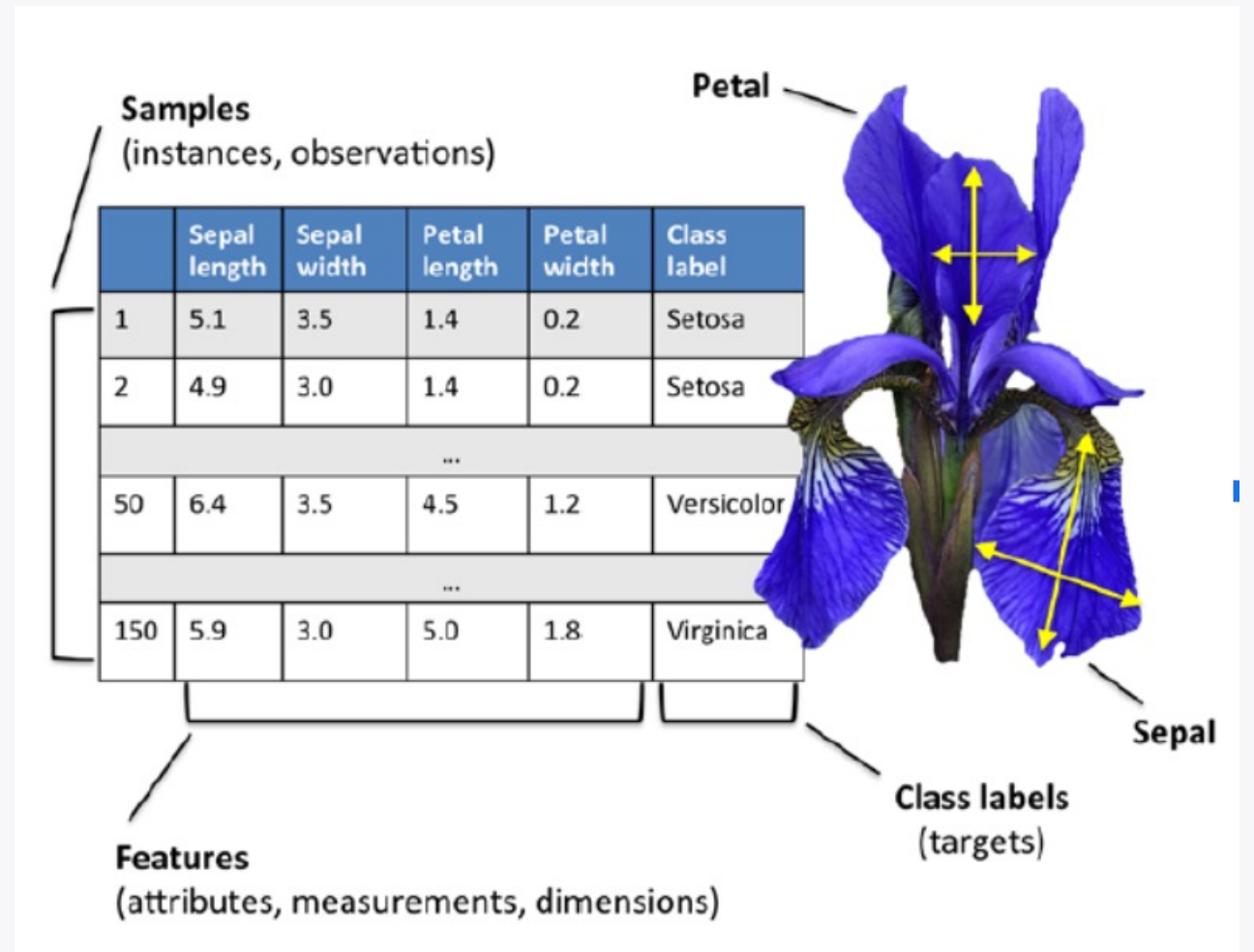
Handwritten  
character  
recognition: **multi-**  
**class** classification



# DATA

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

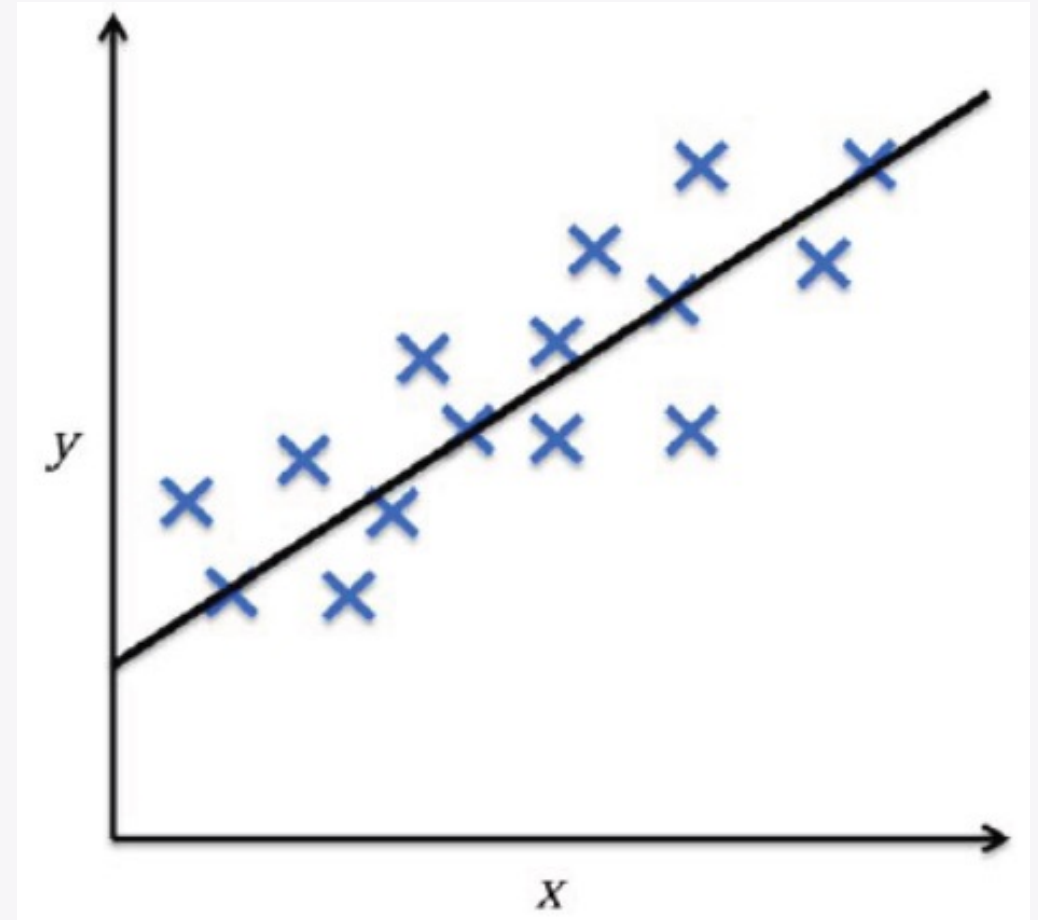
$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} (y \in \{\text{Setosa, Versicolor, Virginica}\})$$



# Supervised Learning: Regression

The goal of the regression is the prediction of **continuous** outcomes.

**Score**  
Prediction

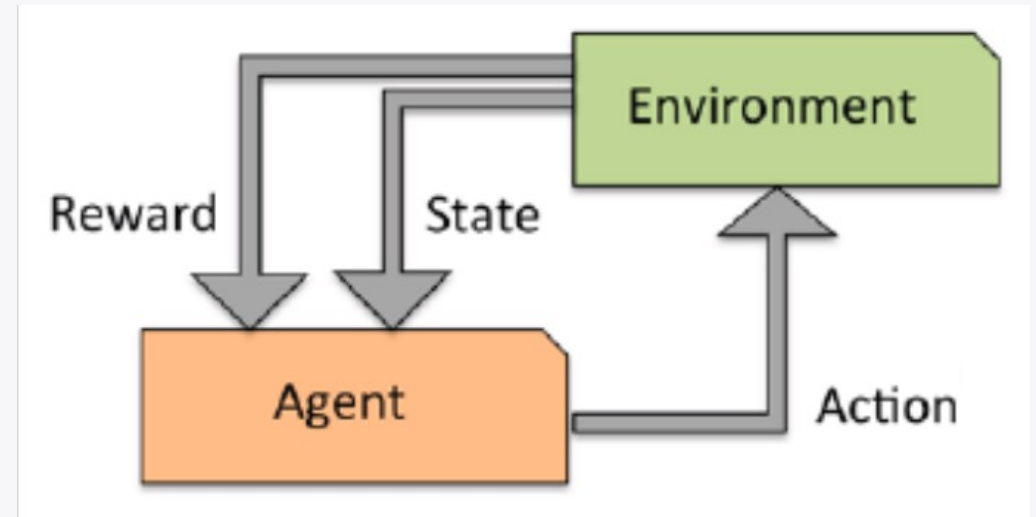




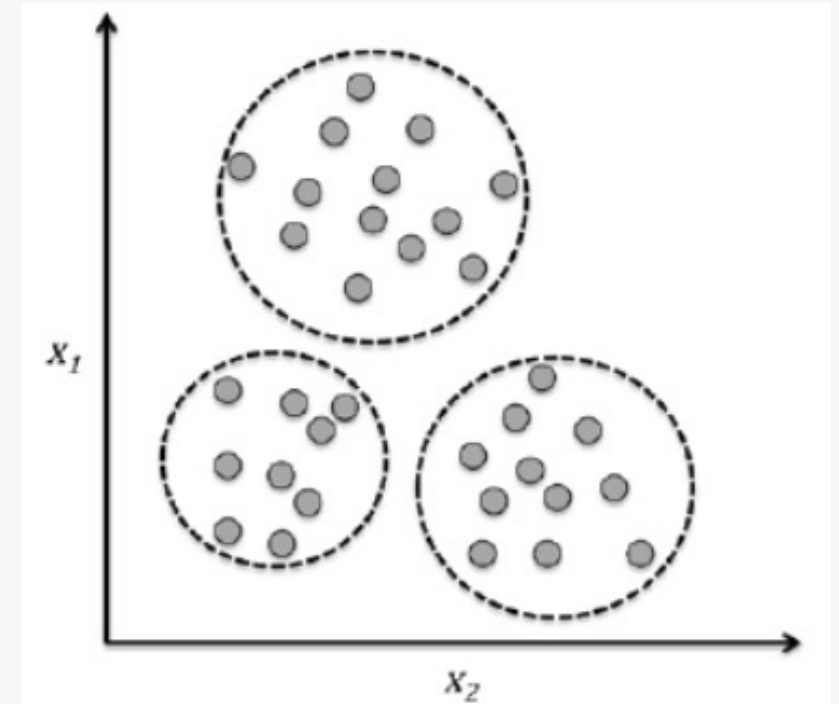
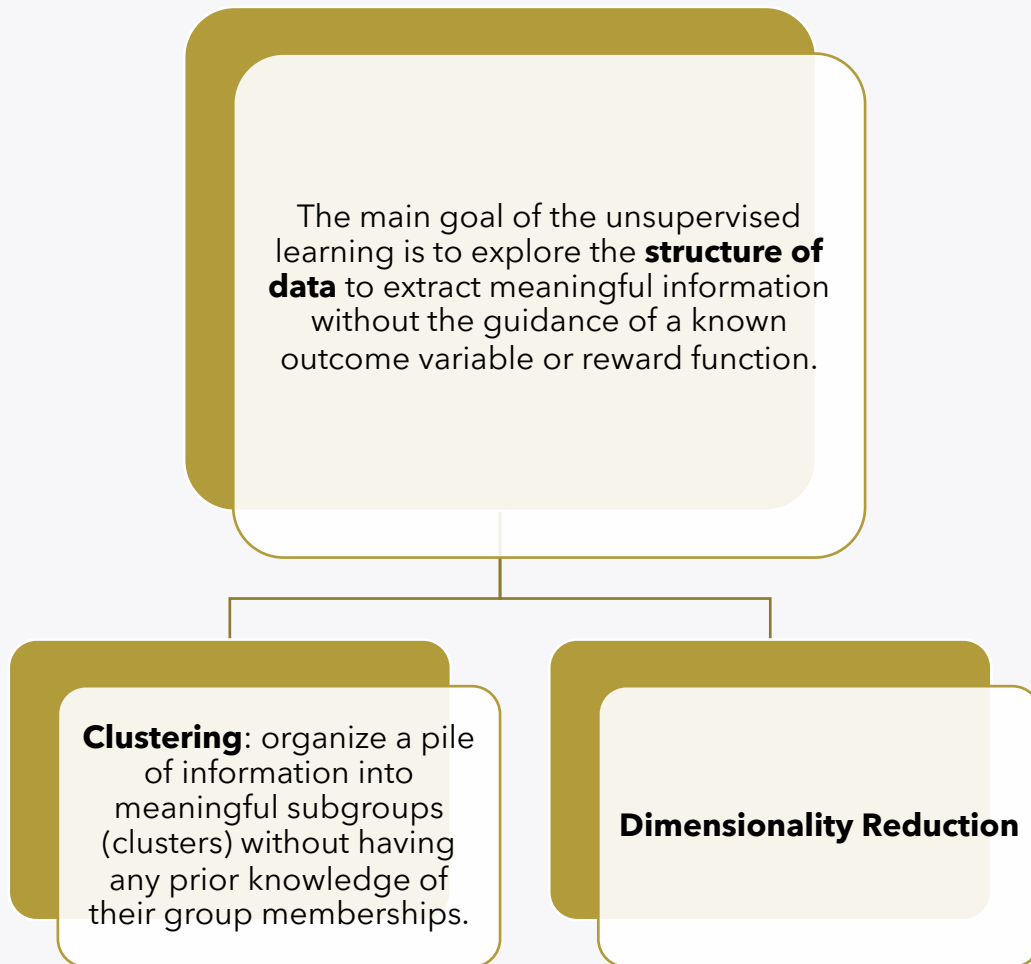
# Reinforcement Learning

The main goal of the reinforcement learning is to develop a system that improves its performance based on **interactions** with the **environment**.

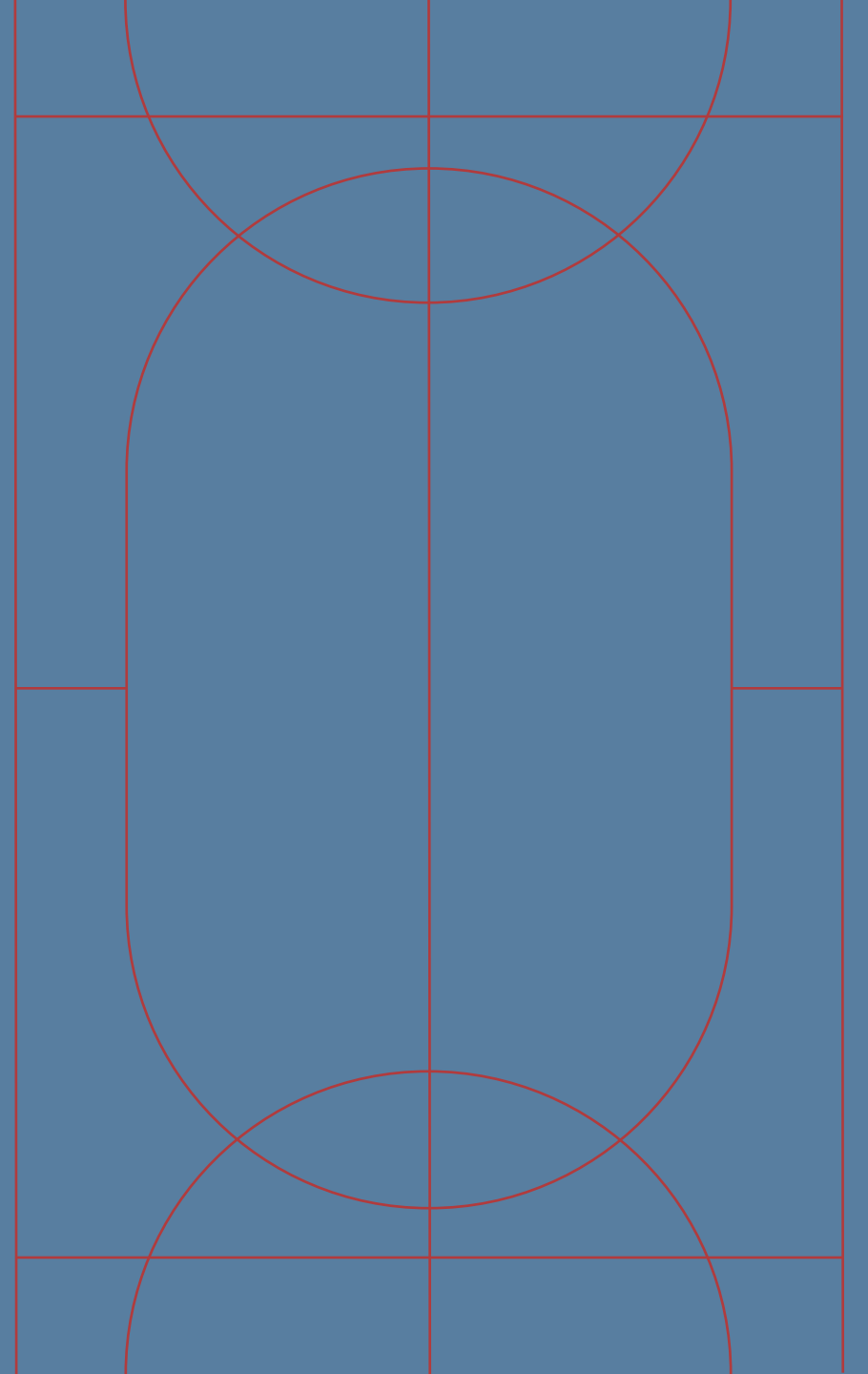
**PacMan** game



# Unsupervised Learning

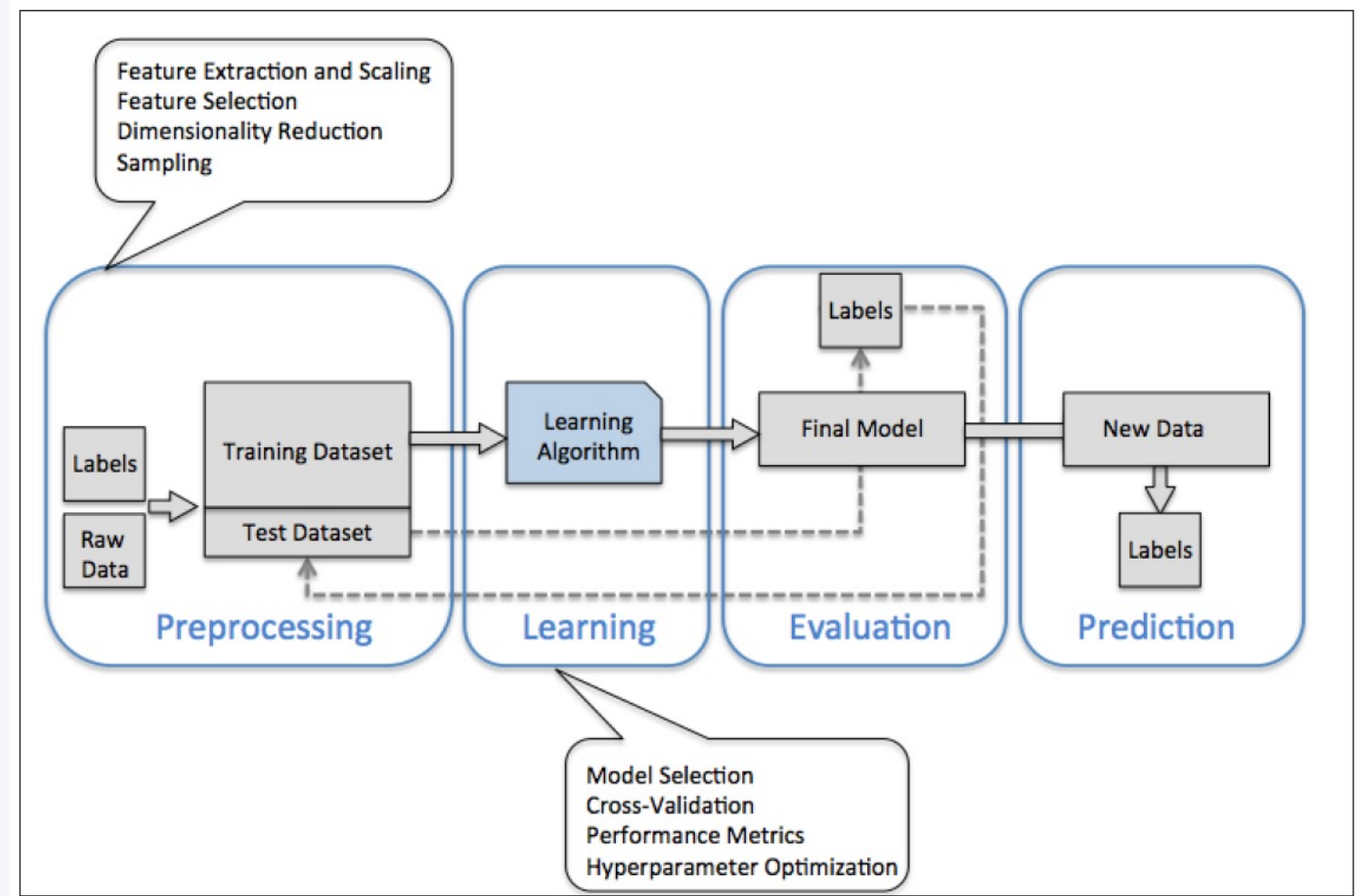


# Supervised Learning



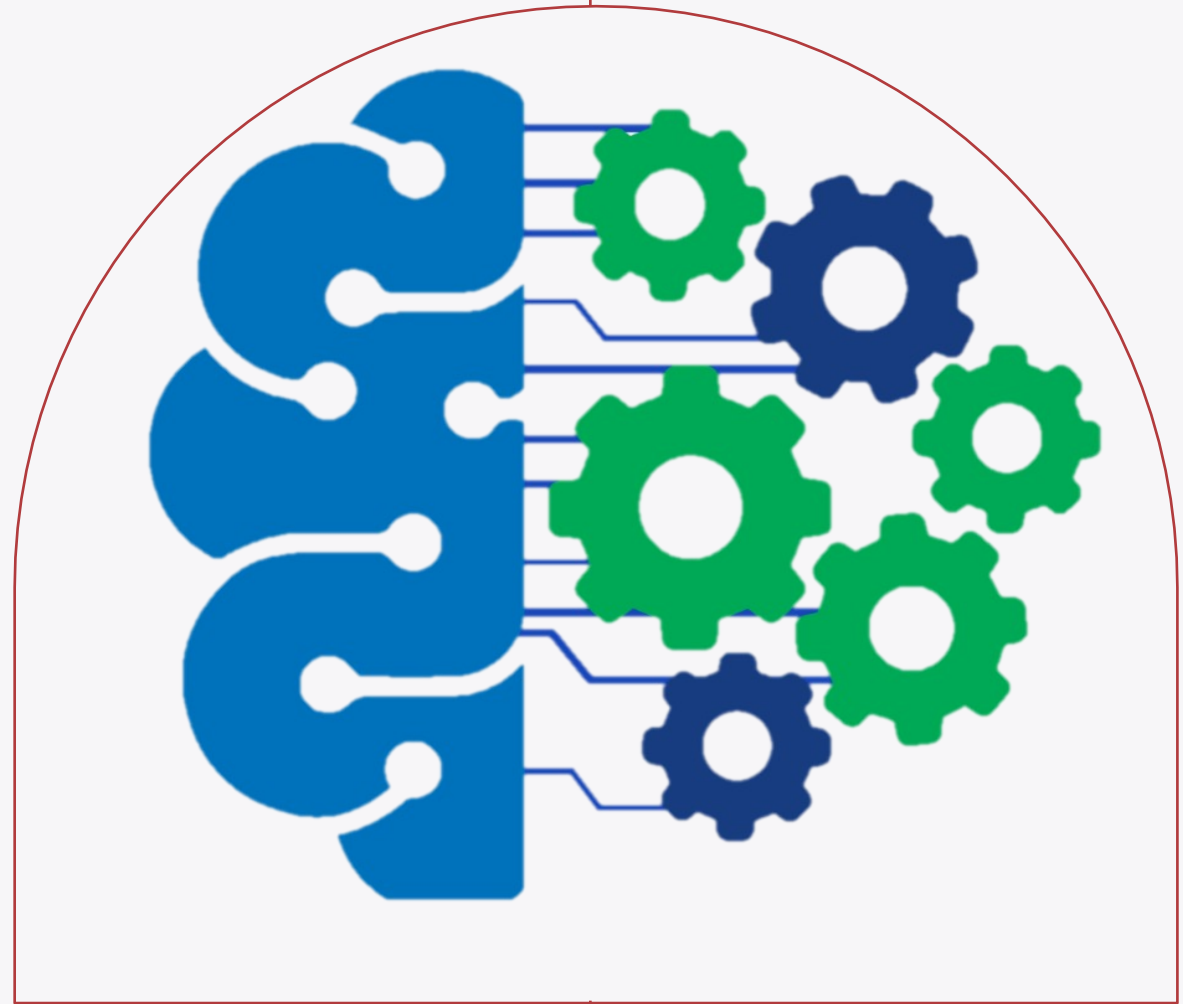


# Supervised Machine Learning Systems: a Roadmap



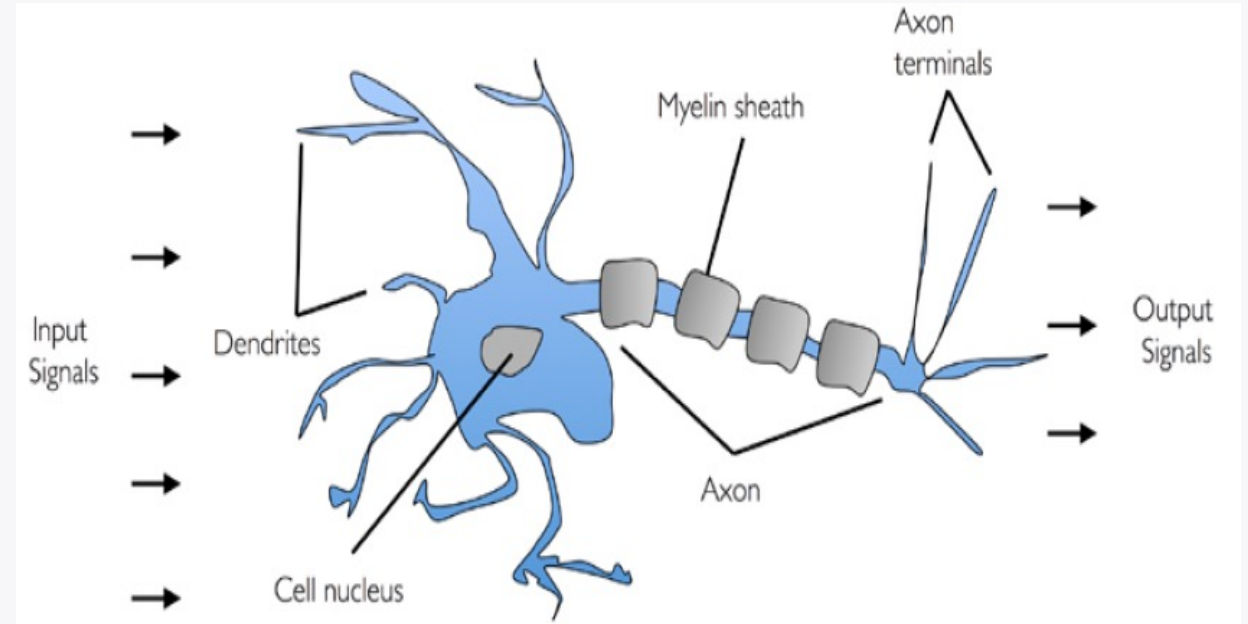
# Supervised Learning Techniques

- Perceptron
- ADALINE
- Logistic Regression
- Artificial neural network
- Support Vector Machine
- Decision Trees
- K-nearest neighbors



# Natural Neurons

- In order to show how artificial neurons works, it is necessary to figure out how natural neurons works.





# McCulloch-Pitts (MCP) Neuron



The first artificial representation of a brain cell is the so-called McCulloch-Pitts (MCP) neuron introduced in 1943.



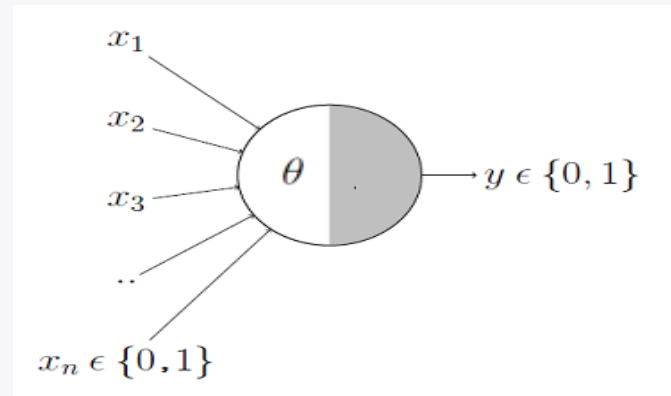
McCulloch and Pitts described a brain cell as a logic gate with **binary inputs and output**;



This gate collects and integrated the set of signals arriving at cell body;



If the value of the integrated signals exceeds a **given threshold**, an output signal is generated and passed to other artificial cells.



*AND function*

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

# McCulloch~ Pitts Neuron: Limitations



What about non-boolean  
(say, real) inputs?



Are all inputs equal? What if  
we want to assign more  
importance to some inputs?



Do we always need to hand  
code the threshold?

# Rosenblatt's Perceptron

- Overcoming the limitations of the MCP neuron, Frank Rosenblatt, an American psychologist, proposed the classical **perception model**, the mighty *artificial neuron*, in 1958. It is more generalized computational model than the McCulloch-Pitts neuron where **weights and thresholds can be learnt over time**.



# Rosenblatt's Perceptron

- It implements a decision function  $\phi(z)$ ;
- $z$  is a linear combination of input values  $x$  and a weight vector  $w$ :

$$z = w_1x_1 + \dots + w_mx_m$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$\phi(z) = \begin{cases} 1 & z \geq \theta \\ -1 & \textit{otherwise} \end{cases}$$

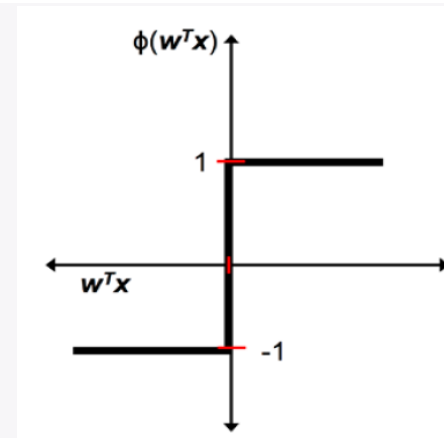
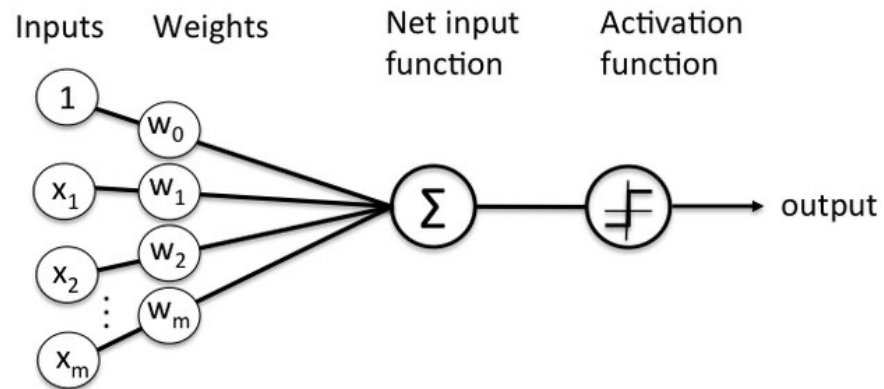
# Rosenblatt's Perceptron

$\theta$  can be moved to the left side of the equation and define a weight-zero as  $w_0 = -\theta$  and  $x_0 = 1$  in order to write  $z$  in a more compact way:

$$z = w_0x_0 + w_1x_1 + \dots + w_mx_m = \mathbf{w}^T \mathbf{x}$$

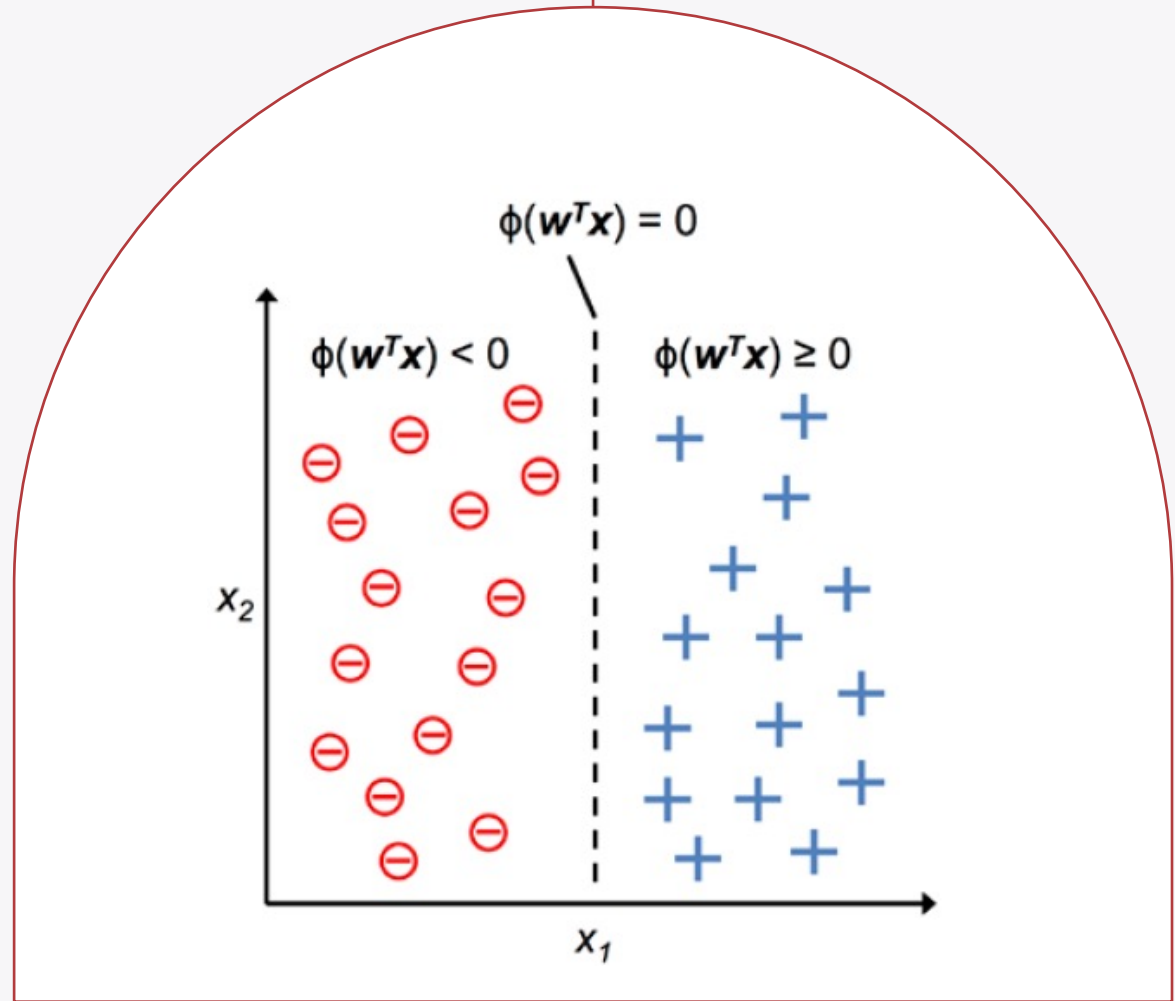
$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

In machine learning the value  $w_0 = -\theta$  is called **bias unit**.



# Rosenblatt's Perceptron as Binary Classifier

- A classifier corresponds to a decision boundary, or a hyperplane such that the positive examples lie on one side, and negative examples lie on the other side.



# Rosenblatt's Perceptron: Learning rule

- Rosenblatt's Perceptron extends MCP neuron with learning capabilities by means an appropriate **training algorithm**

- 1 Initialise the weights to 0 or small random numbers.
- 2 For each training sample  $x^{(i)}$ :
  - 1 Compute the output value  $\hat{y}$ .
  - 2 Update weights,

$$w_j = w_j + \Delta w_j$$

How to compute  $\Delta w_j$ :

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

- $\eta$  is named learning rate (a constant between 0.0 and 1.0);
- $y^{(i)}$  is the true class label of the  $i$ -th training sample;
- $\hat{y}^{(i)}$  is the predict class label.

# Train a Rosenblatt's Perceptron: Example

Example for a two-dimensional dataset:

$$\Delta w_0 = \eta(y^{(i)} - \text{output}^{(i)})$$

$$\Delta w_1 = \eta(y^{(i)} - \text{output}^{(i)})x_1^{(i)}$$

$$\Delta w_2 = \eta(y^{(i)} - \text{output}^{(i)})x_2^{(i)}$$

In the case that the perceptron predicts the class label correctly:

$$\Delta w_j = \eta(-1 - (-1))x_j^{(i)} = 0$$

$$\Delta w_j = \eta(1 - 1)x_j^{(i)} = 0$$

In the case of a wrong prediction:

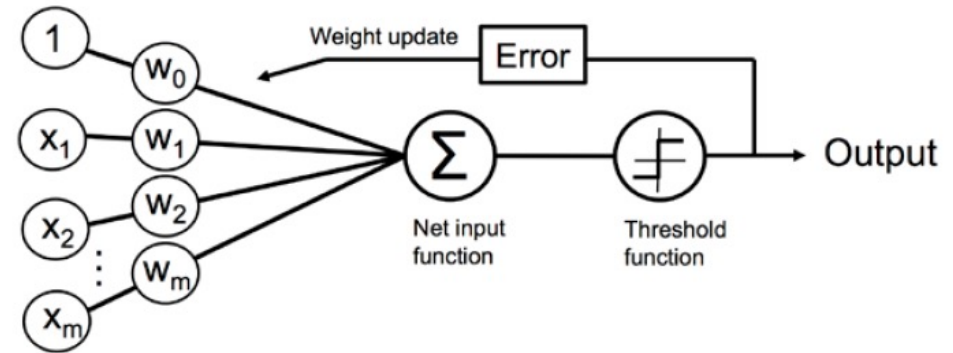
$$\Delta w_j = \eta(1 - (-1))x_j^{(i)} = 2\eta x_j^{(i)}$$

$$\Delta w_j = \eta(-1 - 1)x_j^{(i)} = -2\eta x_j^{(i)}$$



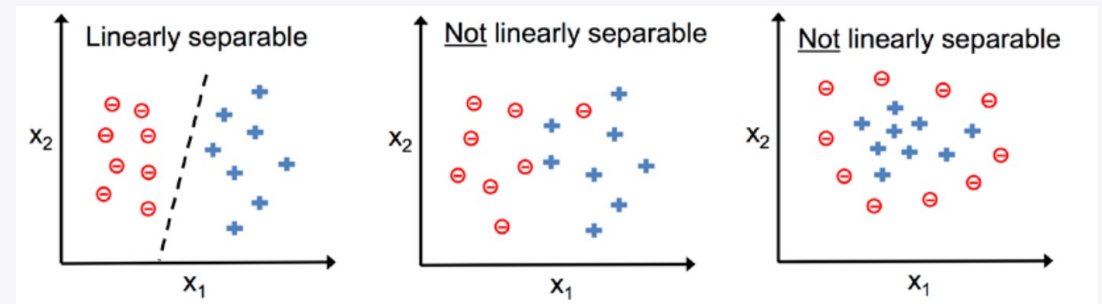
# Rosenblatt's Perceptron: Final scheme

- The learning algorithm passes over the training dataset until all the input vectors are classified correctly (until it achieves convergence)



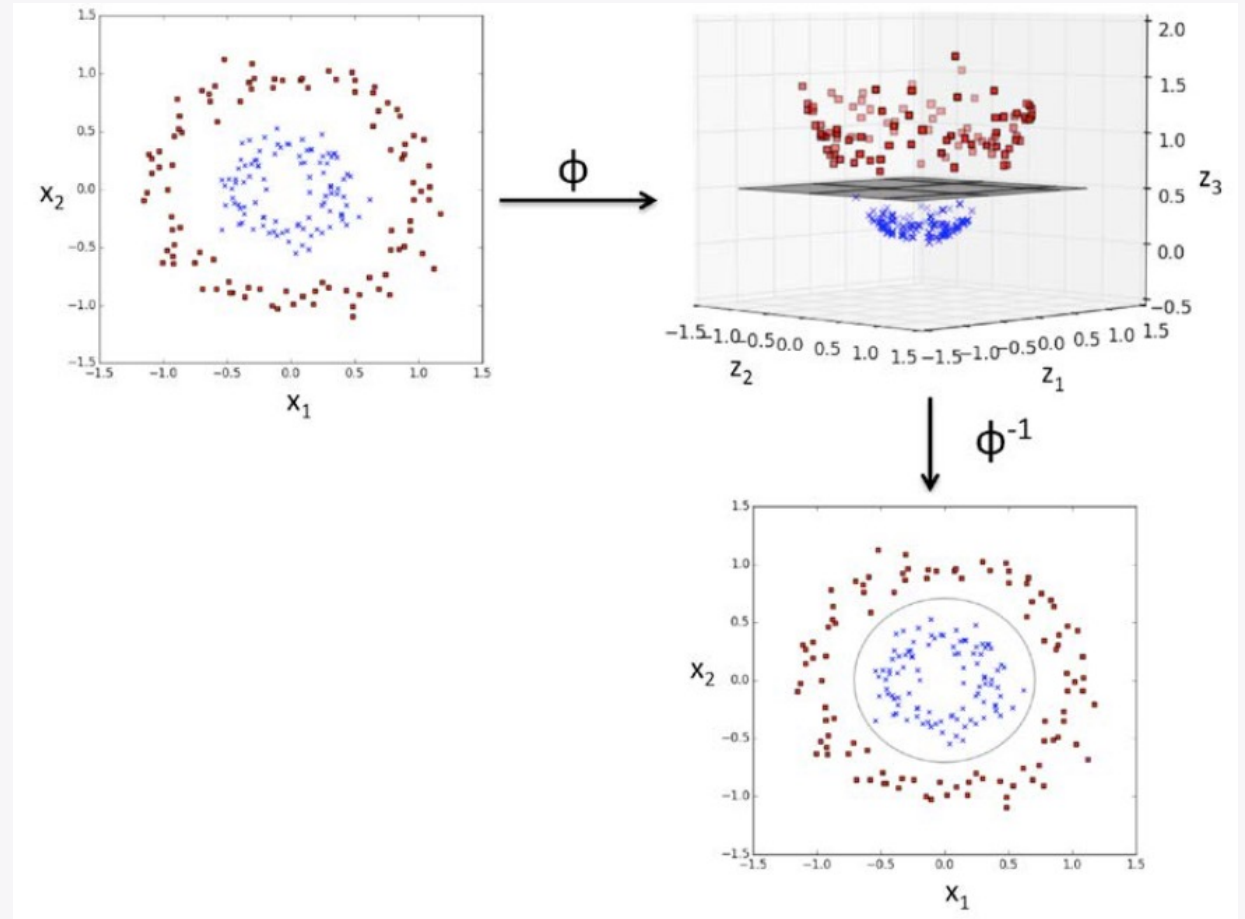
# Rosenblatt's Perceptron: Convergence

- The perceptron will never get to the state with all the input vectors classified correctly (**it will never converge**) if the training set is not linearly separable, i.e. if the positive examples cannot be separated from the negative examples by a hyperplane.
- If the training set is linearly separable, then the perceptron is guaranteed to converge.
- If the two classes can't be separated by a linear decision boundary, we can set a maximum number of passes over the training dataset (**epochs**) and/or a **threshold for the number of tolerated misclassifications**—the perceptron would never stop updating the weights otherwise.



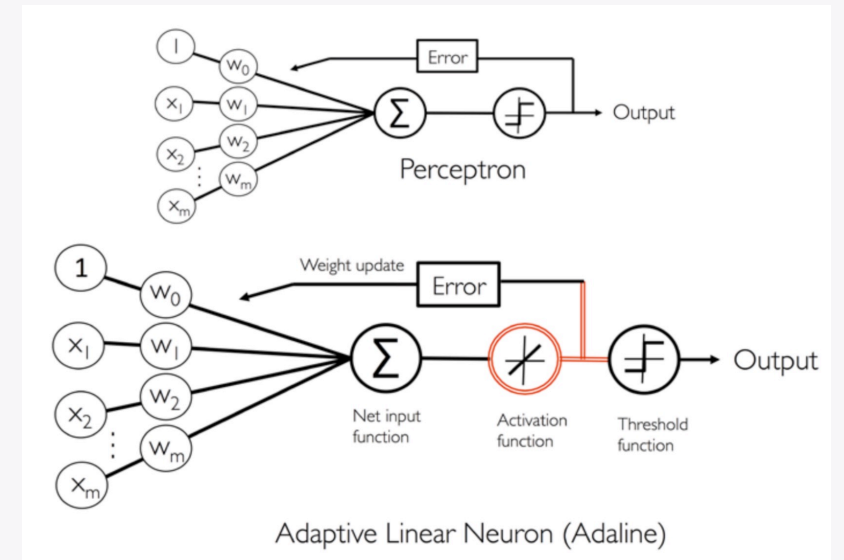
# Not Linearly Separable Dataset

- Kernel Function
- Deep Neural Networks



# Adaptive Linear Neuron: Adaline

- Published by Bernard Widrow and Tedd Ho few years after Frank Rosenblatt's perceptron algorithm (1960);
- It can be considered as an improvement of the perceptron:
  - Adaline uses **continuous predicted values** to learn the model weights, which is more “powerful” since it tells us by “how much” we were right or wrong (activation function is the identity function for Adaline)
  - Adaline uses the **gradient descent** to find the most suitable weights that minimize the error to classify the training data samples.

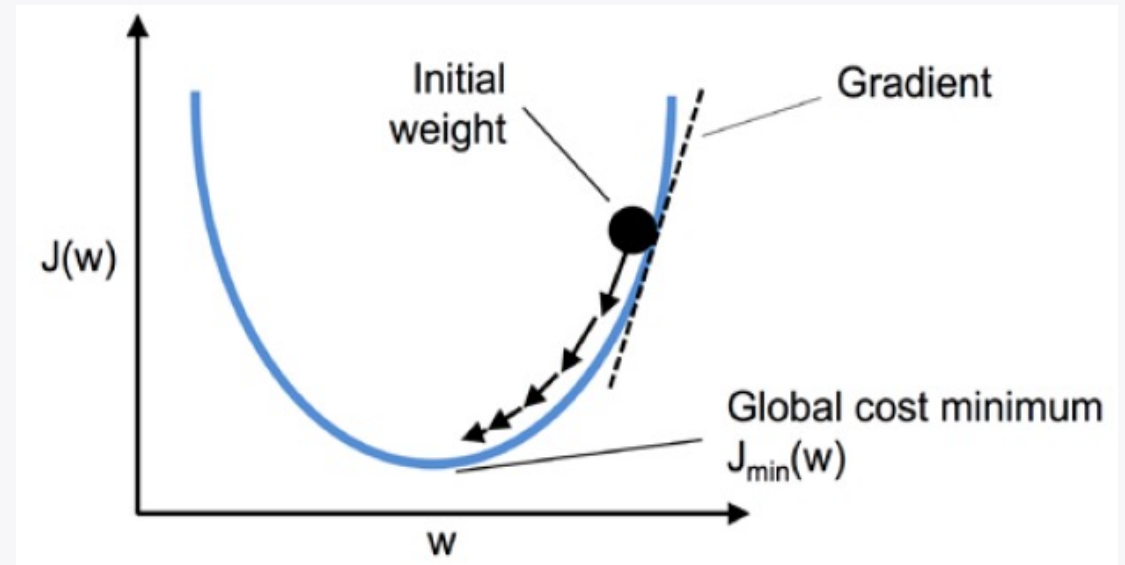


$$\phi(\mathbf{w}^T \mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

# Minimizing Cost Functions with Gradient Descent

- One of the key ingredients of supervised machine learning algorithms is a defined **objective function** that is to be optimized during the learning process.
- In the case of Adaline, we can define the cost function,  $J$ , to learn the weights as the **sum of squared errors (SSE)** between the calculated outcome and the true class label:

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (y^{(i)} - \phi(z^{(i)}))^2$$

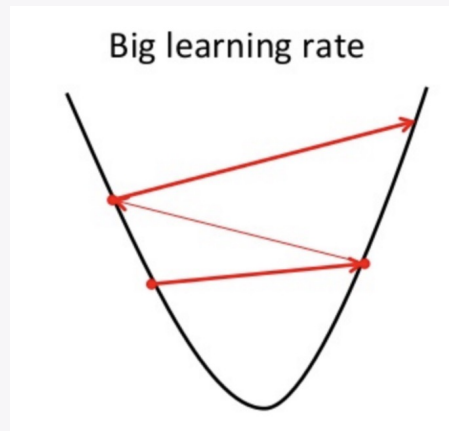


$$\mathbf{w} := \mathbf{w} + \Delta \mathbf{w}$$
$$\Delta \mathbf{w} = -\eta \nabla J(\mathbf{w})$$

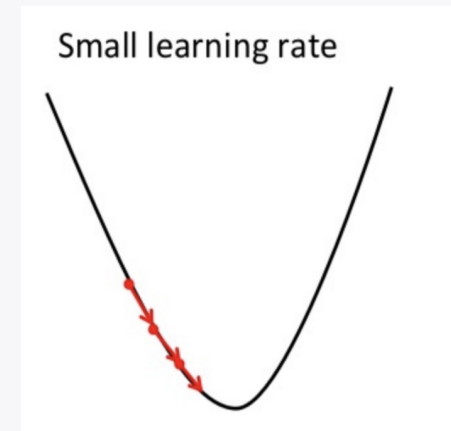


# Gradient Descent: Learning Rate

- $\eta$  is the learning rate constant that determines the size of the steps



With a **high learning rate** we can cover more ground each step (so it **learns faster**), but we **risk overshooting** the lowest point since the slope of the hill is constantly changing.



With a very **low learning rate**, we can **confidently move** in the direction of the negative gradient since we are recalculating it frequently, but calculating the gradient is **time-consuming**, so it will take us a very long time to get to the bottom.

# Scaling Data

- Feature data can have different scales and ranges.
- This can be a problem for gradient descent:
  - The **weights updated is proportional to feature value**, so, with features being on different scales, certain weights may update faster than others
  - It is **difficult to select the most suitable learning** rate value
    - If we choice the value based on the input value having the smallest range, small learning rate it takes ages for the large range to converge.
    - if we choice high value for learning rate, the gradient descent might not converge for small ranges.

“ *Having features on a similar scale can help the gradient descent converge more quickly towards the minima.* ”

- **Feature scaling** is a method used to normalize the range of features of data.

# Scaling Data: Normalization

- Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.
- The general formula for a **min-max** of [0, 1] is given as the top equation
- Another form of normalization is called **mean-normalization**. The formula is the equation in the bottom.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$x' = \frac{x - \text{avg}(x)}{\max(x) - \min(x)}$$

# Scaling Data: Standardization

- Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance.
- The general method of calculation is to determine the distribution mean and standard deviation for each feature. Next we subtract the mean from each feature. Then we divide the values (mean is already subtracted) of each feature by its standard deviation.

$$x' = \frac{x - \mu}{\sigma}$$

# Logistic Regression

- Logistic regression is a binary classifier
  - The output variable  $Y$  has two possible values 0 and 1
- Logistic regression is a probabilistic model
  - its goal is to model the probability of the positive class (i.e., the class that we want to predict), typically class 1.
- Classification
  - To compute the conditional probability of the response  $Y$ , given the input variables  $X$ ,  $Pr(Y|X)$
  - Consider a single input observation  $\mathbf{x}$ , which we will represent by a vector of features  $[x_1, x_2, \dots, x_n]$ , we want to know the probability that this observation  $\mathbf{x}$  belongs to the positive class 1,  $P(Y=1|\mathbf{x})$ .



# Logistic Regression

- To explain the idea behind logistic regression as probabilistic model for binary classification let's first introduce the **odds** in favor of a particular event:

$$\frac{p}{1-p}$$

- The **logit** function is the logarithm of the odds
- Logit function takes input values in range 0 and 1 and transforms them to values over the entire real-number range, which we can use to express a linear relationship between feature values and the log-odds

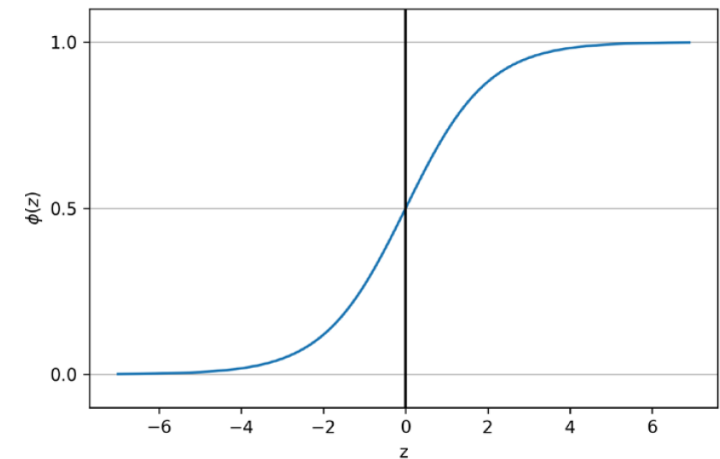
$$\text{logit}(p) = \log \frac{p}{1-p}$$

$$\begin{aligned} \text{logit}(p(y = 1|\mathbf{x})) &= w_0x_0 + \dots + w_mx_m = \\ &= \sum_{i=0}^m w_ix_i = \mathbf{w}^T \mathbf{x} = z \end{aligned}$$

# Logistic Regression: Sigmoid Function

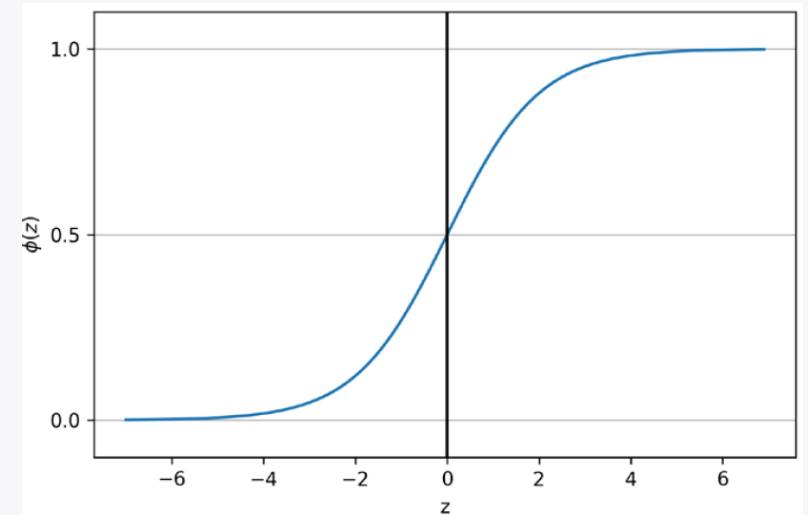
- We are actually interested in predicting the probability that a certain example belongs to a particular class, which is the inverse form of the logit function.
- It is also called the **logistic sigmoid function**, which is sometimes simply abbreviated to **sigmoid function** due to its characteristic S-shape.

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



# Logistic Regression: Decision Boundary

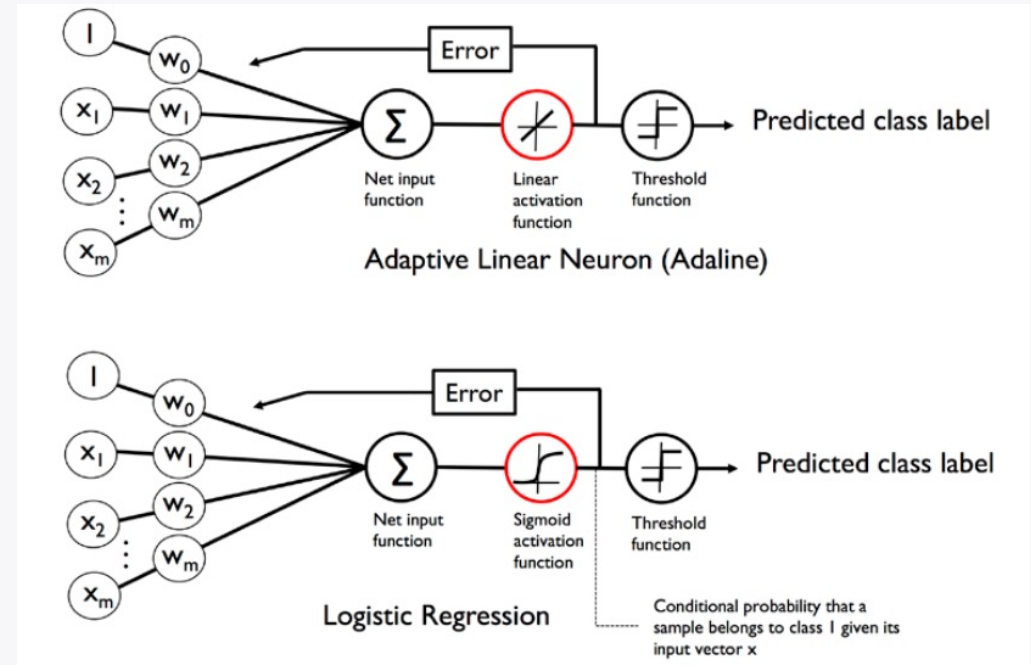
- The predicted probability can then simply be converted into a binary outcome via a quantizer (unit step function):



$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

# Logistic Regression: Scheme

- In Adaline, we used the identity function  $\phi(z) = z$  as activation function. In logistic regression, this activation function simply becomes the **sigmoid** function
- There are many applications where we are not only interested in the predicted class labels, but where the estimation of the class-membership probability is particularly useful

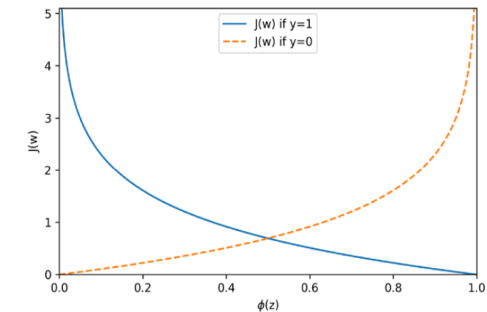


# Logistic Regression: Learning Rule

- Using Logistic Regression we should maximize the likelihood  $L$ .

$$L(\mathbf{w}) = P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \prod_{i=1}^n P(y^{(i)}|x^{(i)};\mathbf{w}) = \prod_{i=1}^n (\phi(z^{(i)})^{y^{(i)}} (1-\phi(z^{(i)}))^{1-y^{(i)}})$$

$$J(\mathbf{w}) = \sum_{i=1}^n \left[ -y^{(i)} \log(\phi(z^{(i)})) - (1-y^{(i)}) \log(1-\phi(z^{(i)})) \right]$$



- Logistic regression uses the gradient descent after converting the log-likelihood function in the cost function  $J$





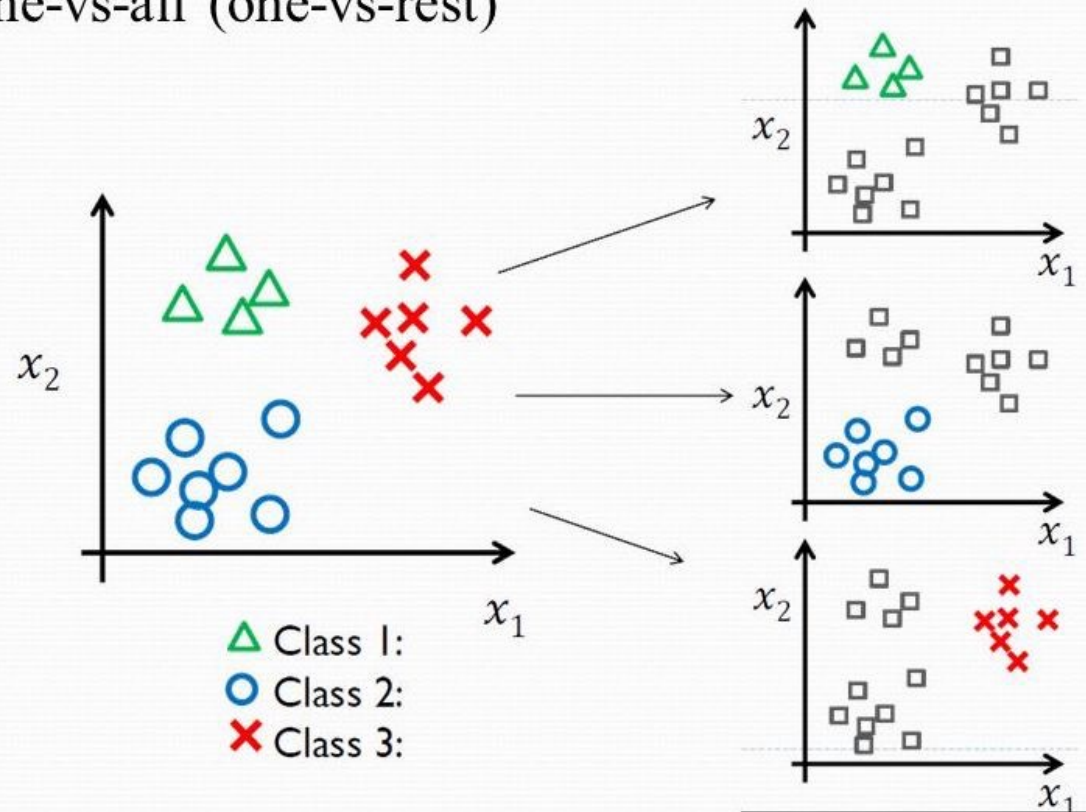
# Multi-Class Classification



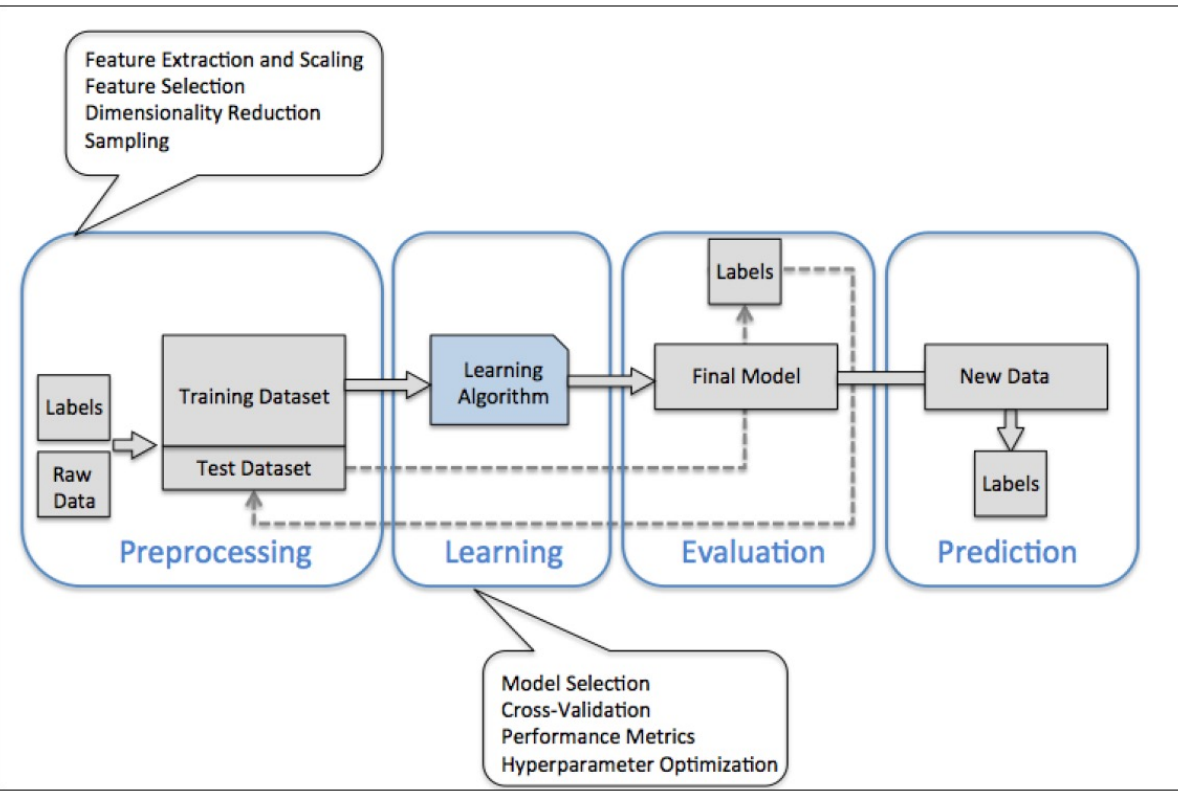
# One-vs-Rest

- The One-vs-Rest strategy splits a multi-class classification into one binary classification problem per class.

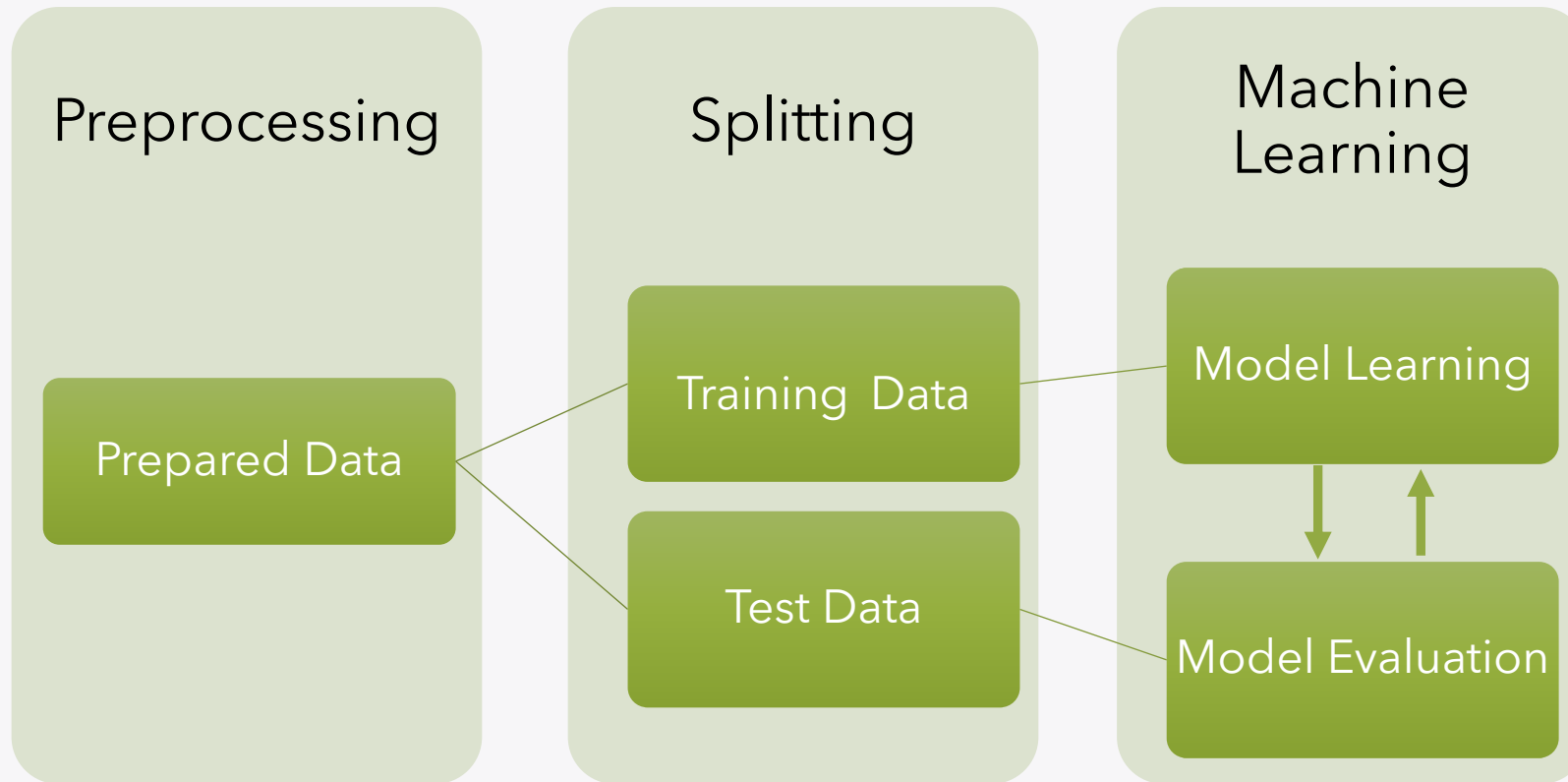
One-vs-all (one-vs-rest)



# A Supervised Machine Learning System



# Train~Test Splitting



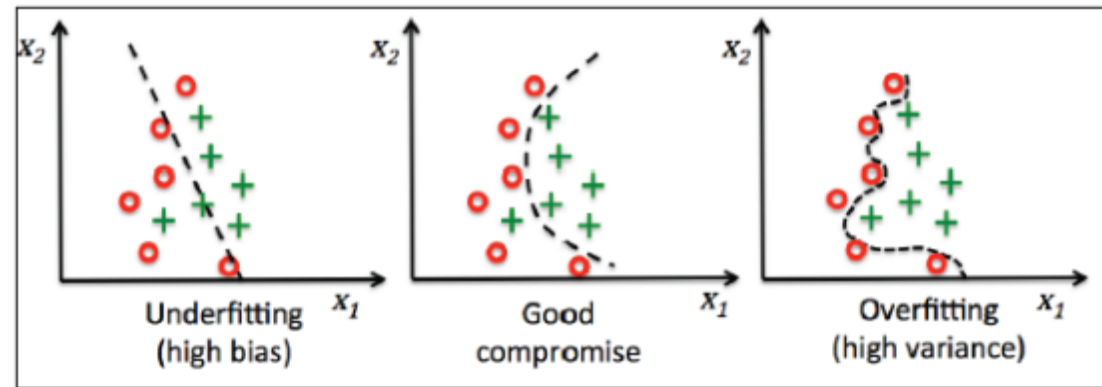
# Metrics

		PREDICTED VALUE	
		Positive	Negative
ACTUAL VALUE	Positive	TP	FN
	Negative	FP	TN

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

# Over/Under-Fitting

- **Overfitting** is a common problem in machine learning, where a model performs well on training data but does not generalize well to unseen data (test data).
- **Underfitting** means that our model is not complex enough to capture the pattern in the training data well and therefore also suffers from low performance on unseen data.



# What's Next?

