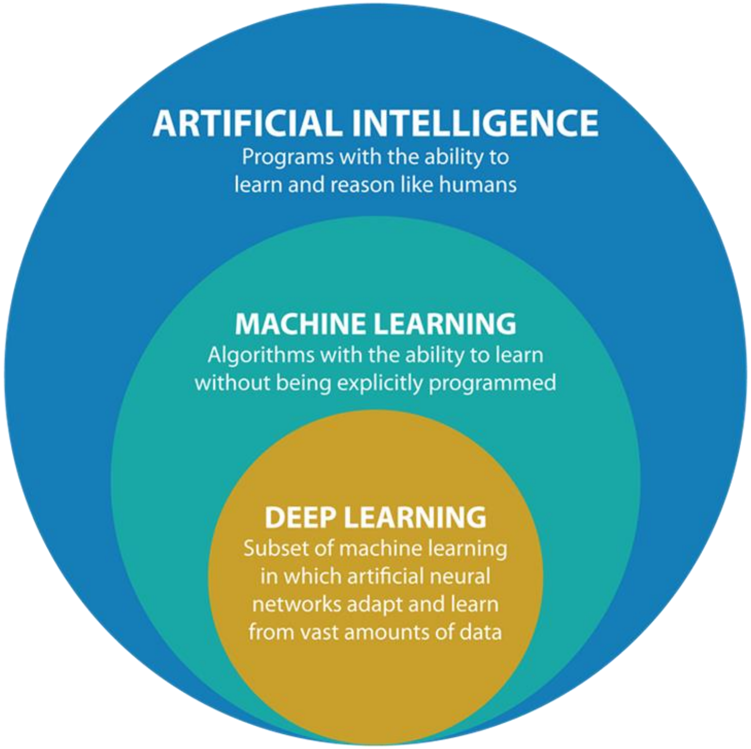
Introduction

Department of Computer Languages and Systems

What is Machine Learning?



Machine Learning (ML) is defined as the subfield of Artificial Intelligence that focuses on the development of algorithms that have the ability to **learn from data** without any human interventions or actions

but, what does it mean to learn?

Source: *https://www.argility.com/argility-ecosystem-solutions/industry-4-0/machine-learning-deep-learning/*

A definition of Machine Learning

A computer system is said to learn from some experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*

Mitchell (1997)

Machine Learning was the term first introduced by **Arthur Samuel in 1959**

Experience

experience: collection of examples (observations, instances, samples, objects) of a task

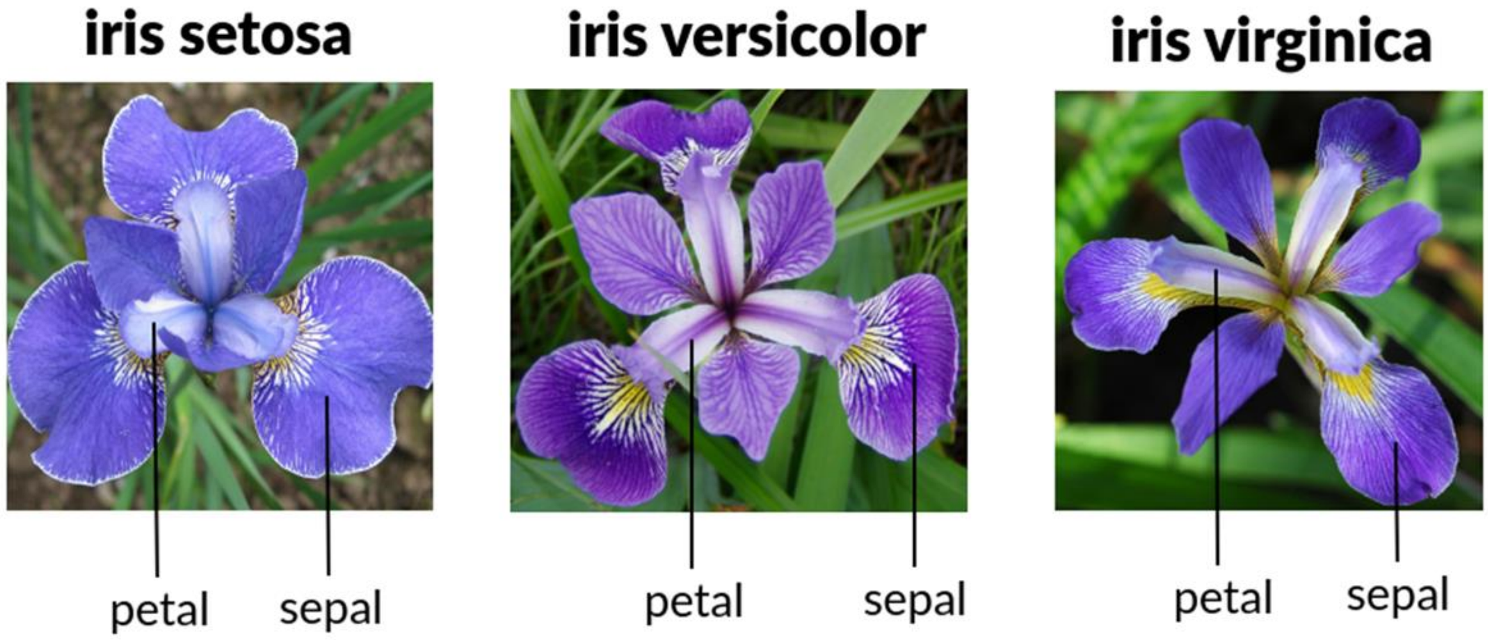
example: collection of features (attributes, variables)

- quantitative measures of a certain object or event
- typically, $x_i \in \mathbb{R}^n$, where x_i is a feature

Some examples of features:

- pixels (features) of an image (example)
- grades (features) of an academic transcript (example)
- notes (features) of a song (example)

Experience (i): the IRIS data set



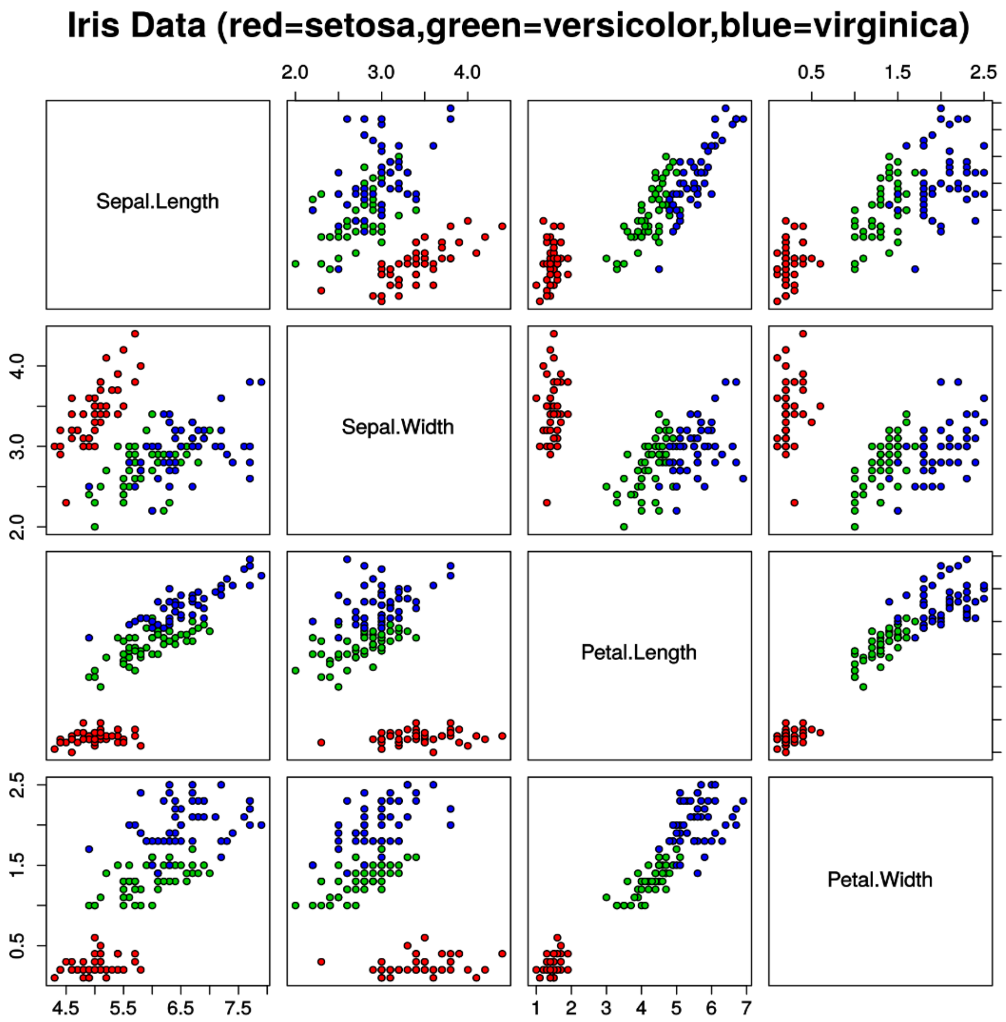
https://s3.amazonaws.com/assets.datacamp.com/blog_assets/iris-machinelearning.png

Experience (ii): the IRIS data set

- the best known database in Machine Learning
- 150 examples evenly distributed in 3 classes:
  - Versicolor
  - Setosa
  - Virginica
- examples described by 4 features:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm

Experience (iii): the IRIS data set

class distribution according to 2D subspaces



https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Iris_dataset_scatterplot.svg/2000px-Iris_dataset_scatterplot.svg.png

Experience (iv): the IRIS data set

Article in which the IRIS database was first introduced:

Fisher, R.A. (1936) "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188

See article …

Experience (v): supervised vs. unsupervised

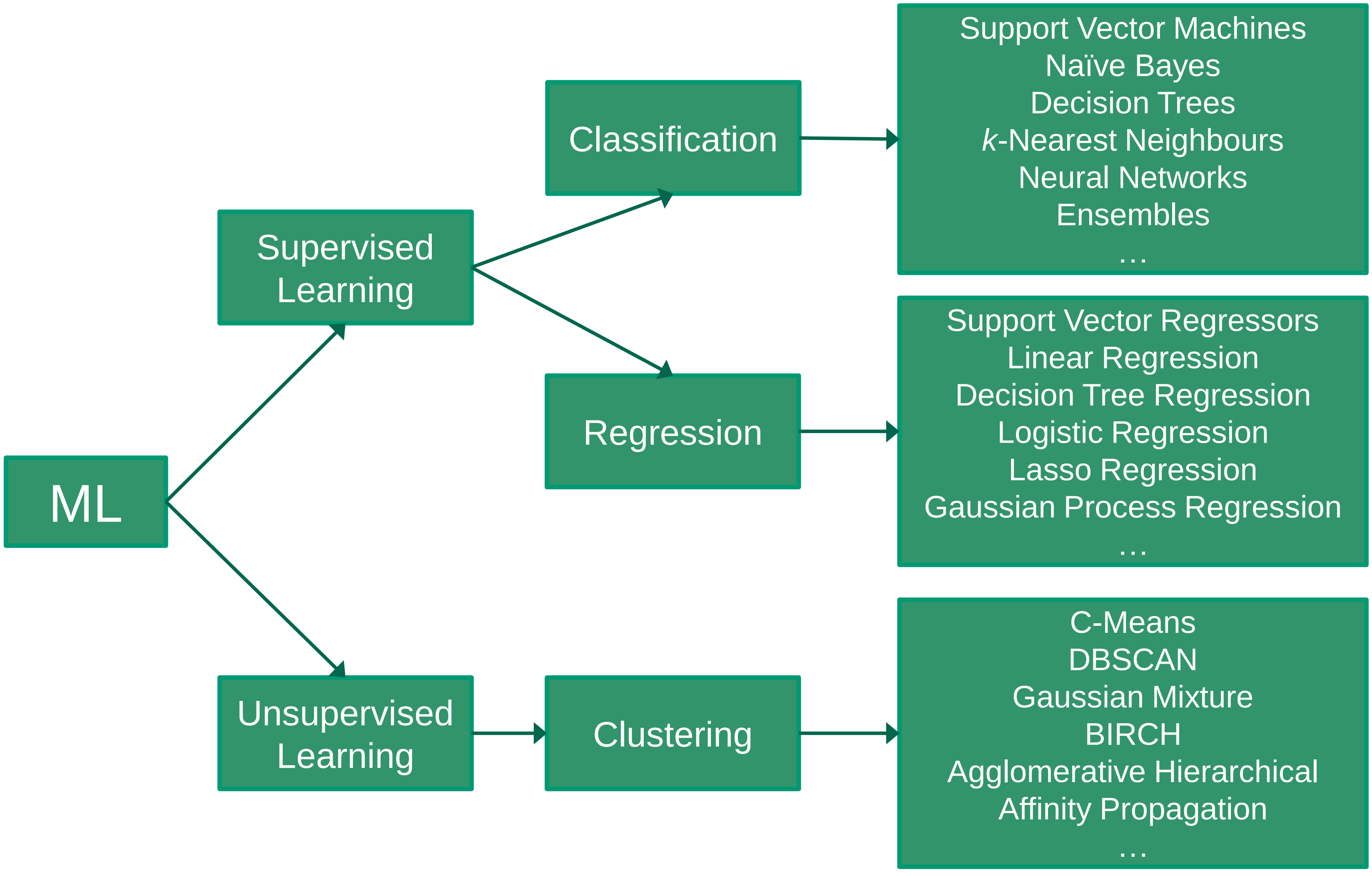
Supervised learning:

- each example x in the data set is associated with a class label y (e.g. the Iris data set)
- supervised learning: it attempts to predict y from x, usually by estimating $p(y \mid x)$

Unsupervised learning:

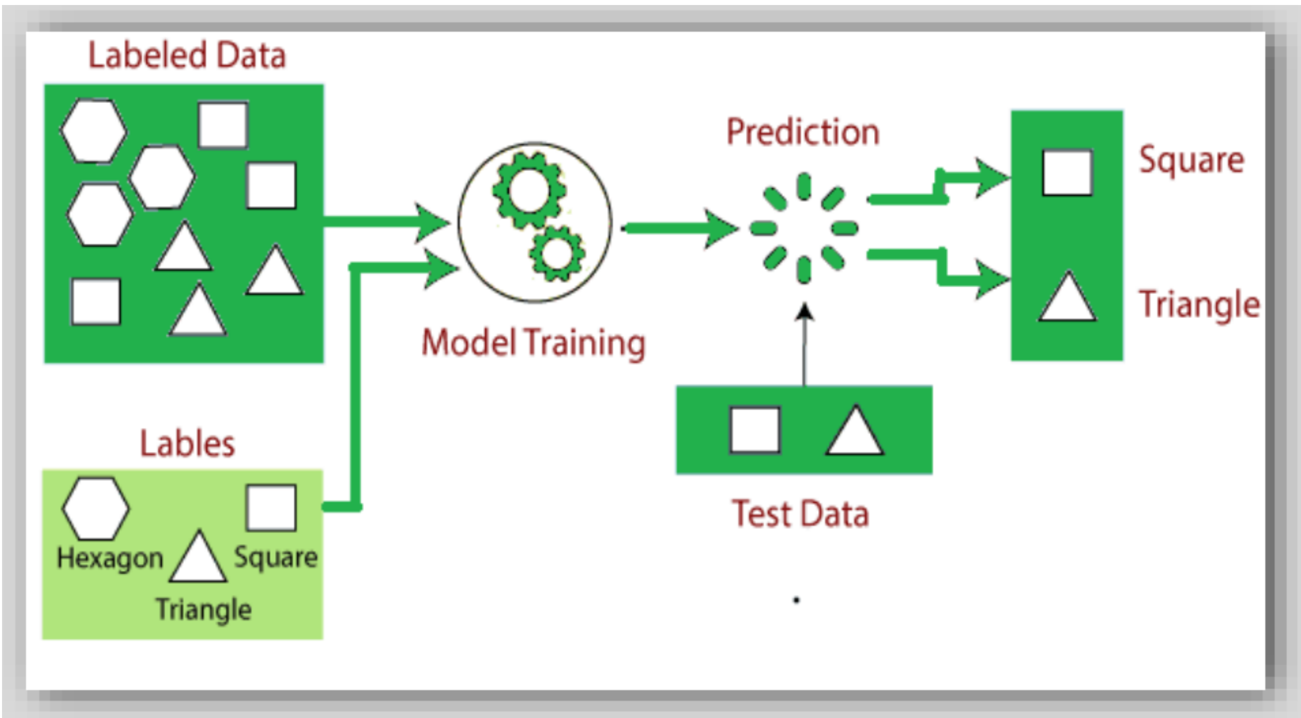
- the examples in the data set are not associated with class labels
- unsupervised learning: it attempts to learn the probability distribution $p(x)$

Experience (vi): supervised vs. unsupervised



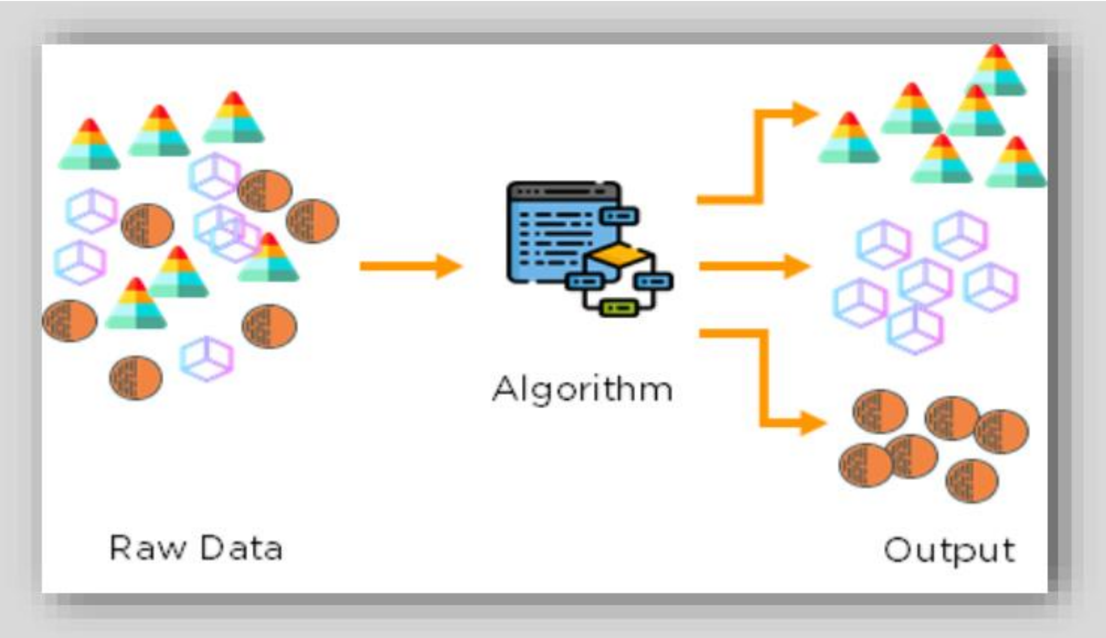
Experience (vii): supervised vs. unsupervised

Supervised Learning



Experience (viii): supervised vs. unsupervised

Unsupervised Learning



Experience (ix): supervised vs. unsupervised

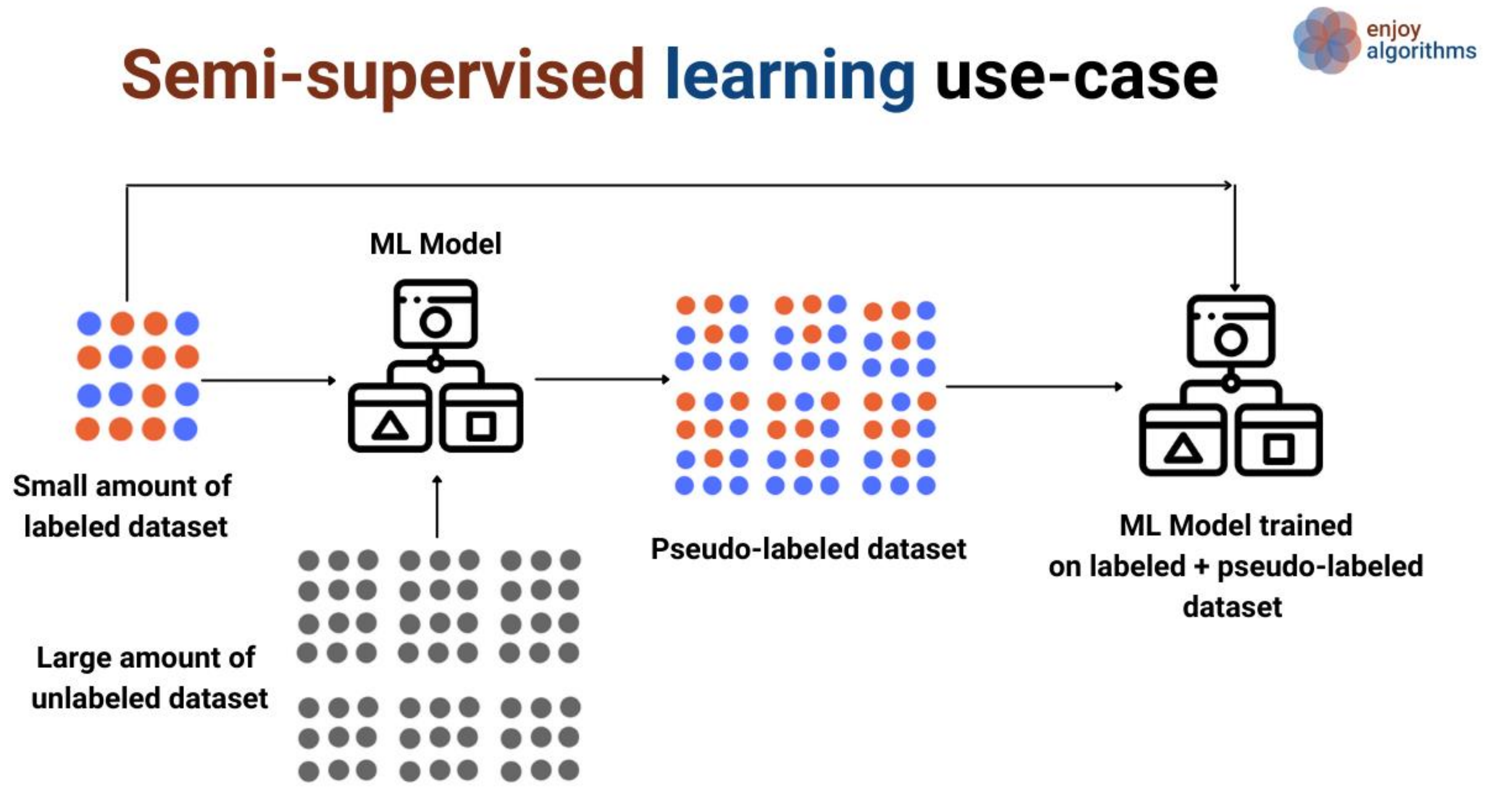
Semi-supervised (or partially supervised) learning:

- in general, to obtain labelled examples is a difficult and costly process
- on the contrary, it is relatively easy and fast to have unlabeled samples
- the solution is to use a small set of labeled examples and a possibly large set of unlabeled samples

We can use the unsupervised techniques to predict labels and then feed these labels to supervised techniques

Mostly applicable in the case of image data sets where usually all images are not labeled

Experience (x): supervised vs. unsupervised



Task

Classification: given an example x, predict its class or category ω_k

$$f: \mathbb{R}^n \rightarrow \{\omega_1, \omega_2, \dots, \omega_c\}$$

Regression: given an example x, predict a single numerical value y. The output value (the one to predict) is called the dependent variable, while the inputs (those used to predict) are the independent variable.

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

Task (ii): parametric model vs. non-parametric model

Parametric model

it assumes the functional form of the mathematical function (f), that is, it assumes a specific form (linear, quadratic, etc.) of the probability distribution with some parameters

Non-parametric model

it does not make any assumptions about the functional form

Task (iii): parametric model vs. non-parametric model

Parametric models

- poor performance if the assumptions are not met
- less training data are required
- computationally faster
- good at interpretability

- Some algorithms: naïve Bayes, linear regression, logistic regression, linear SVM, neural networks, etc.

Non-parametric models

- more accurate predictions since they offer a better fit to data
- easier since they do not need to make any assumption
- risk of overfitting

- Some algorithms: decision tree, *k* nearest neighbours, non-linear SVM, Gaussian processes, etc.

Performance measure

Performance measure *P* …

- quantitative measure of performance (effectiveness) to evaluate the abilities of a machine learning algorithm
- in general, it depends on the task to be evaluated
- commonly, it is calculated on a test sample not seen before, that is, in the training stage (generalization)

Performance measure (ii): examples

- Classification:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ of\ predictions}$$

$$error\ rate = 1\ - accuracy$$

Error rate is often referred to as the expected 0-1 loss: the 0-1 loss on an example is 0 if it is correctly classified and 1 if it is not

- Regression:

$$mean\ squared\ error\ (MSE) = \frac{1}{N}\sum_{i=1}^{N}(f(xi) - yi)2$$

Generalization

Learning (estimation of the parameters of a model)

- training set: the data set used in this stage
- training error: the error the model makes on the training set
- objective: to minimize the error on the training set

Evaluation (to measure the performance of the model on new data)

- test set: the data set used to evaluate the model (new data, different from those used in learning)
- test error: the error the model makes on the test set
- expectation: to achieve a test error close to the training error

Generalization (ii)

Generalization

- performance on new examples not used in training/learning
- expected value of the error on a new sample/input
- it is estimated by measuring the performance of the model on the test set

$$\text{generalization error} \approx \text{test error}$$

Generalization (iii)

Assumptions (known as the i.i.d. assumptions)

- the training examples are independent of the test examples
- the training and test sets are identically distributed
- i.e. the same (unknown, implicit) distribution function generated both training and test examples

- if we could set the parameters in advance, $training\ error\ \cong test\ error$
- however, the parameters are optimized from the training set and therefore, $expected\ training\ error\ \leq expected\ test\ error$

Generalization (iv)

The factors that determine how well a learning algorithm will perform are its ability to:

- make the training error small
- make the difference between training and test errors small

These two factors correspond to the two central challenges in machine learning: underfitting and overfitting

Underfitting and overfitting

underfitting

if the model is too simple, it cannot represent the data well enough and cannot obtain a sufficiently low error rate on the training set

overfitting

the difference between the errors obtained on the training and test sets is too large. If the model is too complex, it can predict the training data set labels very well, but it does not perform so well with the test set

Model capacity

model capacity

- is the ability of a model to fit different data distributions
- allows you to control the probability that a model falls into *underfitting* or *overfitting*

- low capacity model will be unable to achieve a good fit to the training data (*underfitting*)
- too high capacity model will memorize training data, but it will not generalize (*overfitting*)

Model capacity (ii)

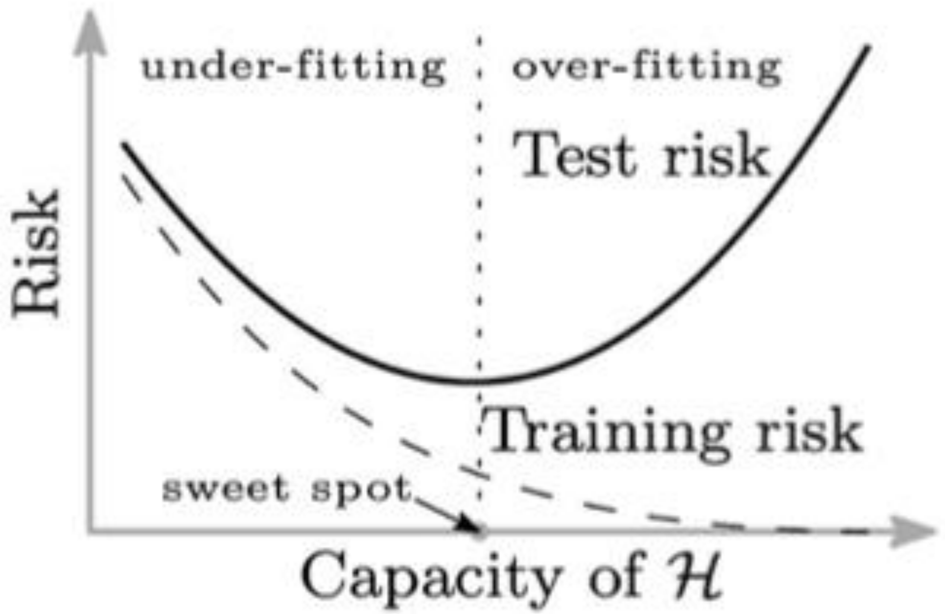


Figure from Belkin et al. (2019)

A case study: shopping cart

A supermarket chain…

- sells millions of products to millions of customers
- for each transaction stores: date, customer, products and quantities, cost, …
- raw data impossible to manage
- then, why store so much data?

Objective:

to learn from historical data how to predict future events

A case study: shopping cart

What can be learned?

- what customer profiles exist (pensioners, students, families*, …*)
- correlations between customer profiles, products, days of the week, dates of the year, weather, etc.
- correlations between products (beer + chips, cava + chocolates, diapers + wipes*, etc.)*

What can be predicted?

- how much of a certain product will be sold on a given day

A case study: shopping cart

Given

a future date f,

we want to

predict how much of a product p will be sold on day f

A case study: shopping cart

Stages, steps:

- to collect historical data
- to set assumptions
- to define the representation of the elements of the problem
- to estimate or predict the value of the target variable
- to evaluate estimates (and the method chosen)
- to refine the solution method

A case study: shopping cart
historical data collection

Data: what past experiences could be useful to us?

- example: given the same date f of some previous year …
  - how much of product p was sold on day f?
  - how much of product p was sold on day f-1?
  - what day of the week (Monday, Tuesday) was f?
  - what were the weather conditions (temperature, general condition, humidity, wind, ...)?
  - what was the retail price of product p?
  - has the retail price of p changed from previous days? How did it change?
  - what was the economic situation (unemployment rate, Euribor, ...)?

A case study: shopping cart
to set assumptions

Assumptions: what can we assume about customers or motivations?

- customers act independently
- the decision to buy one product is independent of the decision to buy any other
- social behaviors are similar between different years

A case study: shopping cart
representation of the elements

Representation: how do we “summarize” past experiences?

```
description(f) = (sale_f-1,            // numerical data
                  week_day,             // Mo,Tu,We,Th,Fr,Sa,Su
                  temperature,          // numerical data in ºC
                  general_conditions,   // sunny, cloudy
                  price,                // numerical data in €
                  price_change,         // less, equal, greater
                  unemployment_rate,    // numerical data in %
                  Euribor )             // numerical data in %

sale(p) = quantity_sold                 // target variable
```

Training set: {[description(f_i),sale(p_i)]}

A case study: shopping cart
estimation, prediction

Estimation: how to predict **sale(p)** at a future date f?

- given:
  - a training set associated to dates f_i: **{[description(f_i),sale(p_i)]}**
  - the description of a future date f: **description(f)**
- estimate:
  - **sale(p)**

Method: find the relationship between descriptions (inputs) and sales (outputs)

A case study: shopping cart
estimation, prediction

Simple and intuitive method: nearest neighbour classifier

- given a future date f and its **`description(`f`)`**
- find, in the training set, the closest (most similar) `description(`f_i`)` to **`description(`f`)`**
- estimation: **`sale(`p`)`** $\leftarrow$ `sale(`p_i`)`

A case study: shopping cart
performance evaluation

Evaluation: how to measure the quality of predictions?

- wait for date f and compare prediction and reality… but then the decisions would have already been made
- we could use part of the data set; method:
  - split the data set into two subsets, one for training and one for test,
  - build the classifier from the training set,
  - evaluate the classifier using the test set,
  - calculate the precision of the estimate: MSE (regression), accuracy/error rate (classification), etc.

A case study: shopping cart
refine the method

Refine: can we improve the quality of the prediction?

- improvement areas:
  - assumptions; e.g. perhaps one product depends on another
  - representation; e.g. we might summarize other information, or the same information differently (`price_change` could be numeric)
  - estimation method; e.g. we could use a different classifier, or change the parameters of a classifier
- how to validate an improvement? how to choose an alternative?
  - depending on the method used to assess the quality of the prediction

When is ML necessary?

Machine Learning is necessary when …

- there is no human expert to tackle a task
- we are not able to explain our expertise (e.g. how to convert speech into written text, how to recognize faces, …)
- a task cannot be performed by a human (e.g. manage large volumes of data, very short response times, etc.)

Current applications

- Healthcare: to make data-driven decisions that can prevent diseases, help in better patient diagnosis, speed-up root cause detection, etc.
- Manufacturing: to optimize resource planning, cut short the time to market, quality control, etc.
- Energy: power consumption and requirements prediction, dynamic per unit cost maintenance, etc.
- Banking and finance: to safeguard from frauds, illegal financial detection, identify valuable customers, credit risk prediction, stock market prediction, etc.
- Governance and surveillance: real-time image detection, drone surveillance, biometric (re)identification, automated social network monitoring, etc.

Current applications (ii)

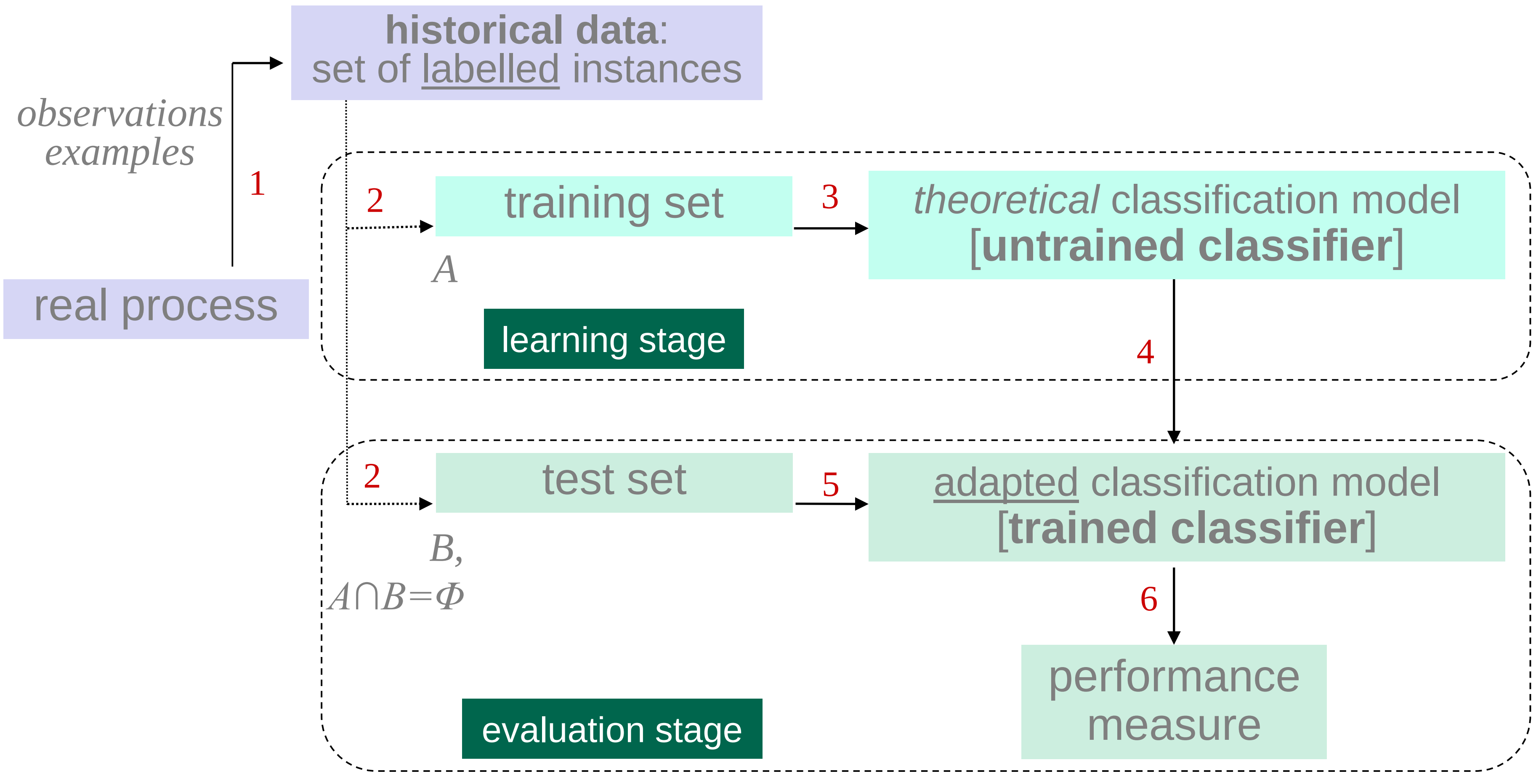
- E-commerce: to make personalized recommendations, fashion and brand design, etc.
- Marketing: to show relevant Ads to customers, identify target customers, churn analysis, etc.
- Digital media and entertainment: user behavior analysis, spam filtering, social media analysis and monitoring, etc.
- Automobile: to optimize fuel consumption, breakdown prediction, self-driving, etc.
- Robotics: assistive robots, humanoid robots, underwater robots, surgery, etc.

Example applications

- Self-driving vehicles
- Handwritten character recognition
- Email spam filtering
- Face recognition
- Robotics

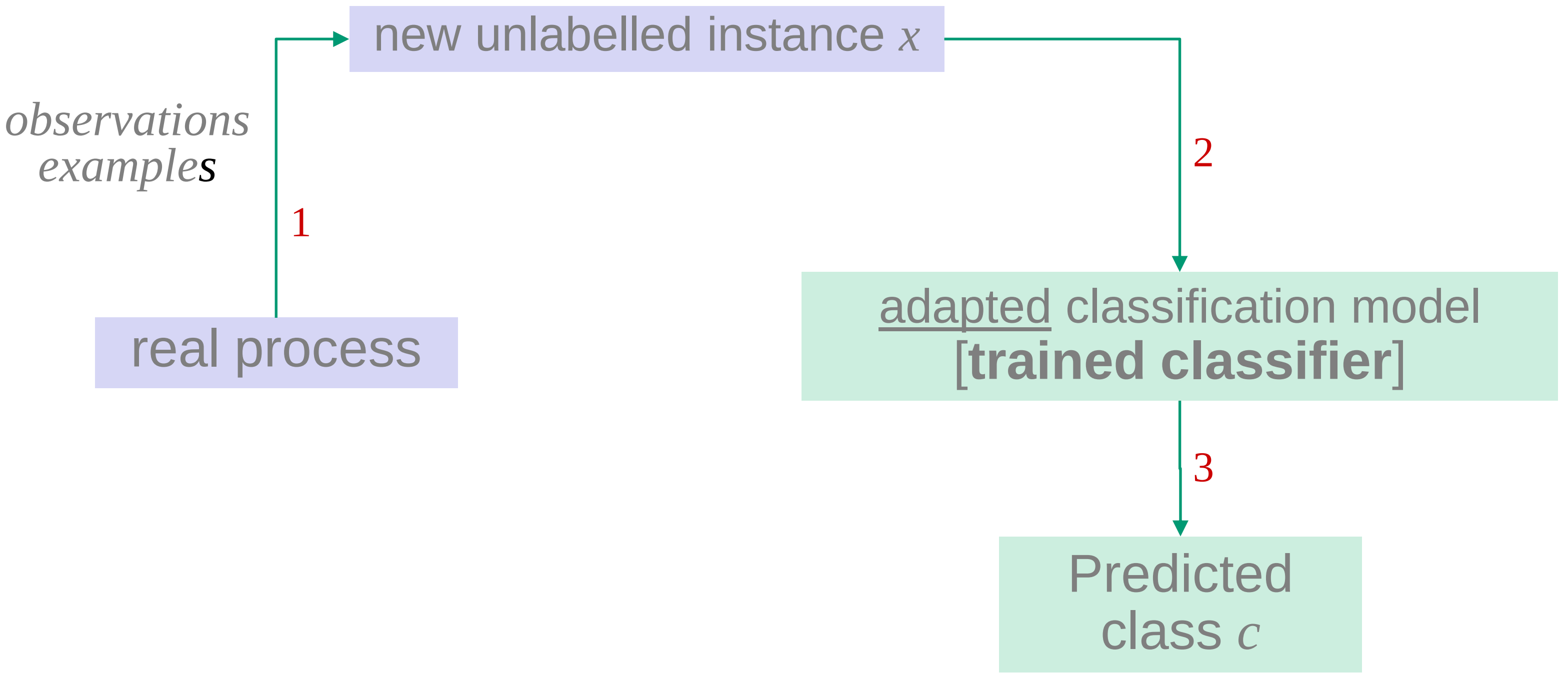
Classification

general skeleton of a classifier



Classification

use of a classifier



Classification
notation

- $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_d) \in X$, an instance
- d, dimensionality (number of features or attributes to measure)
- $X \subseteq \mathbb{R}^d$, d-dimensional space of instances
- $C = \{\omega_1, \ldots, \omega_c\}$, set of labelled classes
- c, number of classes
- $T_{tra} = \{(x_i , c_i)\}$, training set
- $T_{tst} = \{(x_i , c_i)\}$, test set
- n, number of instances
- $g_\theta(\mathrm{x})$, theoretical untrained classifier
- $g_{\theta',tra}(\mathrm{x})$, trained classifier