Machine Learning

Lecture 1 : Course Overview, Introduction to Machine Learning, Regression

Mohamad GHASSANY

EFREI PARIS

Mohamad GHASSANY

- Associate Professor at EFREI Paris, head of Data & Artificial Intelligence Master program.
- Phd in Computer Science Université Paris 13.
- Master 2 in Applied Mathematics & Statistics from Université Grenoble Alpes.
- Personal Website: mghassany.com

Introduction

Machine Learning examples

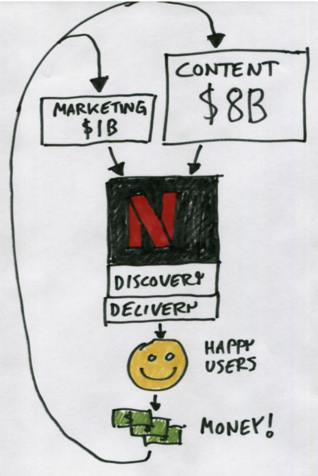
You probably use it dozens of times a day without even knowing it.

Application examples:

- Effective web search.
- Social networks recognize friends from photos or suggest friends.
- Email spam detection.
- Handwriting recognition.
- Understanding the human genome.
- Medical diagnostics.
- Predict possibility for a certain disease on basis of clinical measures.
- Fraud detection.
- Drive vehicles.
- Recommendations (eg, Amazon, Netflix).
- Natural language processing.

The aim of ML is to build computer systems that can adapt to their environments and learn form experience.

This is a high-level view of what Netflix does.

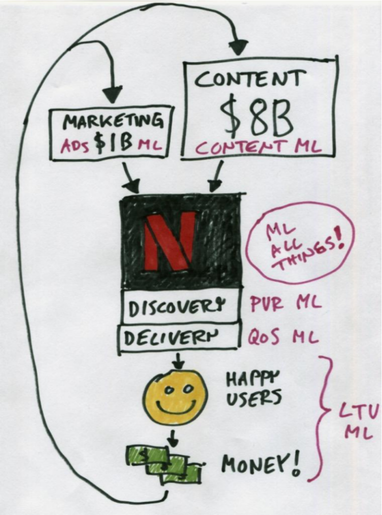


[1] Savin Goyal - useR'19

It is probably necessary to **get smarter** about everything:

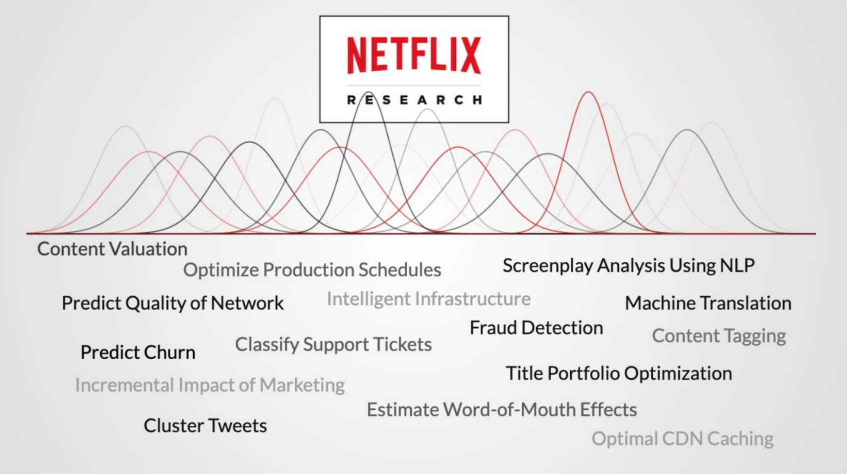
- Content acquisition
- Marketing
- Discovery
- Delivery
- and more.

ML gets applied everywhere!

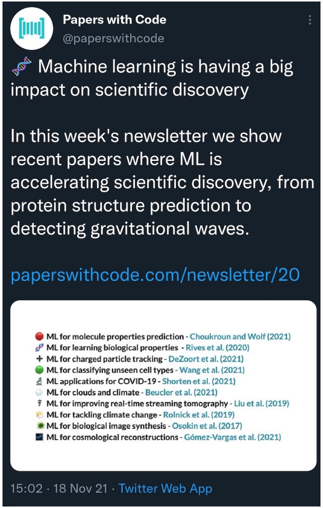


[2] Savin Goyal - useR'19

Machine Learning examples[3]

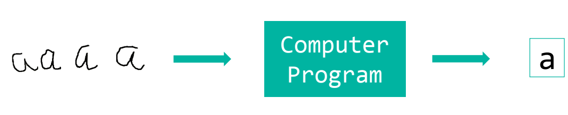


[3] Savin Goyal - useR'19


Papers with Code
@paperswithcode
Machine learning is having a big impact on scientific discovery
In this week's newsletter we show recent papers where ML is accelerating scientific discovery, from protein structure prediction to detecting gravitational waves.
paperswithcode.com/newsletter/20
ML for molecule properties prediction - Choukroun and Wolf (2021)
ML for learning biological properties - Rives et al. (2020)
ML for charged particle tracking - DeZoort et al. (2021)
ML for classifying unseen cell types - Wang et al. (2021)
ML applications for COVID-19 - Shorten et al. (2021)
ML for clouds and climate - Beucler et al. (2021)
ML for improving real-time streaming tomography - Liu et al. (2019)
ML for tackling climate change - Rolnick et al. (2019)
ML for biological image synthesis - Osokin et al. (2017)
ML for cosmological reconstructions - Gómez-Vargas et al. (2021)
15:02 · 18 Nov 21 · Twitter Web App

What is Machine Learning?

- A science of getting computers to learn without being explicitly programmed[4].
- Study of algorithms that improve their performance P at some task T with experience E[5].



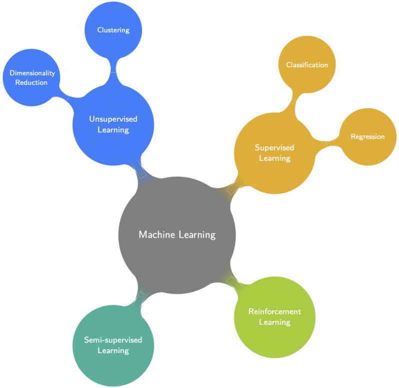
T: recognition of a handwritten letter “a” from its image.

E: images of a handwritten “a”.

P: recognition rate.

[4] Arthur Samuel.
[5] Tom Mitchell.

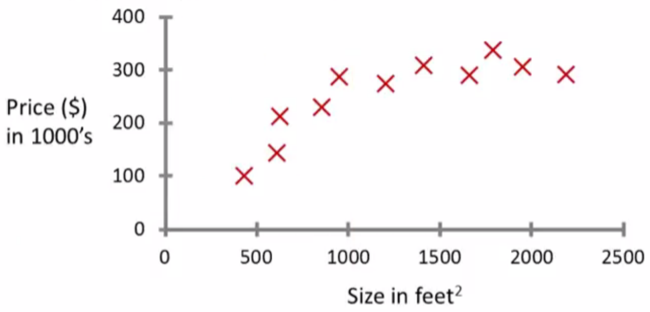
In general, any machine learning problem can be assigned to one of two broad types:



Supervised Learning

Example: House price prediction[6]

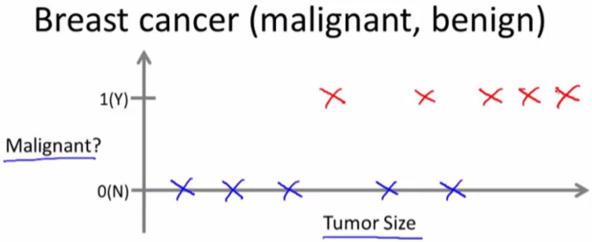
Let's say we want to predict housing prices. We plot a data set and it looks like this:



Let's say we own a house that is, say 750 square feet and hoping to sell the house and we want to know how much we can get for the house.

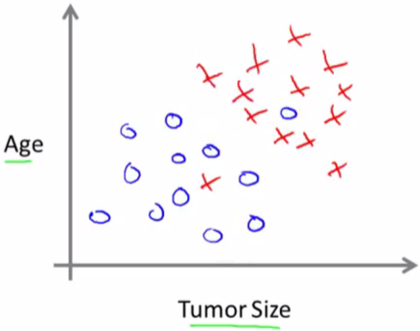
[6] Examples from Andrew Ng's MOOC.

Let's say a person has a breast tumor, and her breast tumor size is known.



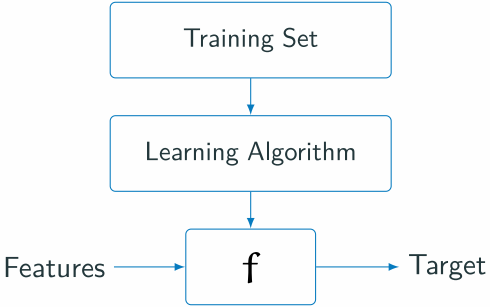
- The machine learning question here is, can you estimate what is the **probability** that a tumor is malignant versus benign?

Let's say that we know both the age of the patients and the tumor size. In that case maybe the data set will look like this.



The term supervised learning refers to the fact that we gave the algorithm a data set in which the **“right answers”** (known as **labels**) were given.

The term supervised learning refers to the fact that we gave the algorithm a data set in which the **"right answers"** (known as **labels**) were given.



- Supervised Learning refers to a set of approaches for estimating f.
- f is also called ***hypothesis*** in Machine Learning.

Regression

- The example of the house price prediction is also called a regression problem.
- A regression problem is when we try to predict a quantitative (continuous) value output. Namely the price in the example.

Classification

- The process for predicting qualitative (categorical, discrete) responses is known as classification.
- Methods: Logistic regression, Support Vector Machines, etc..

Notations:

- The size of the house in the first example, tumor size and age in the second example, are the **input** variables. Typically denoted by X.
- The inputs go by different names, such as *predictors*, *independent variables*, *features*, *predictor* or sometimes just *variables*.

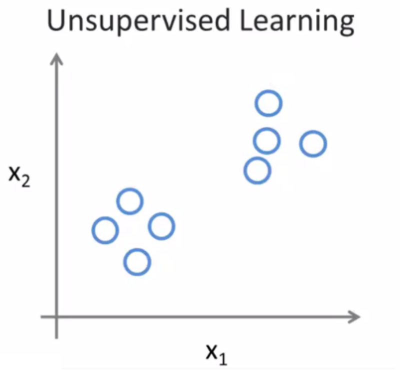
Notations:

- The size of the house in the first example, tumor size and age in the second example, are the **input** variables. Typically denoted by X.
- The inputs go by different names, such as *predictors*, *independent variables*, *features*, *predictor* or sometimes just *variables*.
- The house price in the first example and the diagnosis in the second example are the **output** variables, and are typically denoted using the symbol Y.
- The output variable is often called the *response*, *dependent variable* or *target*.

Unsupervised Learning

In Unsupervised Learning, we’re given data that doesn’t have any **labels**.

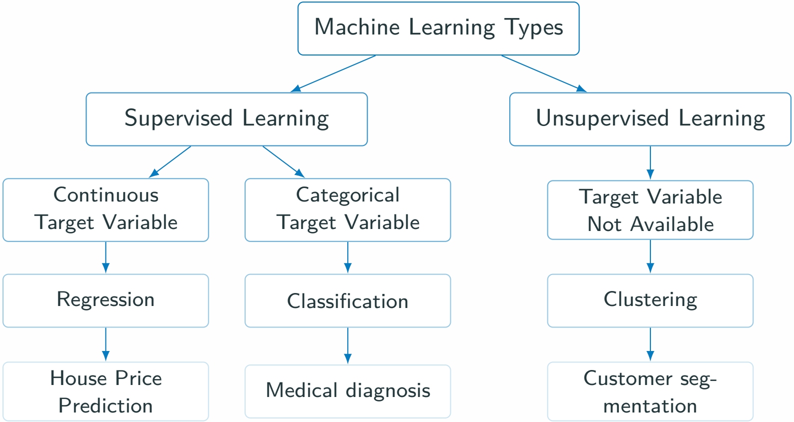
For example:



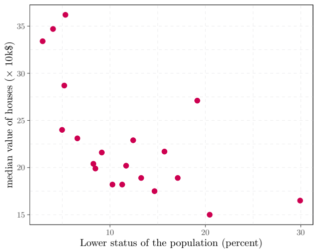
Question: Can you find some structure in the data?

One example where clustering is used is in Google News (news.google.com)

Types of Machine Learning Problems

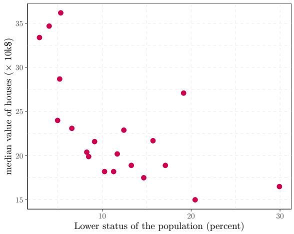


Linear Regression



Let:

- n: sample size
- x: features
- y: target variable
- $(x^{(i)}, y^{(i)})$: one sample, a training example



- Hypothesis: $f(x) = f_\omega(x) = \omega_0 + \omega_1 x$
- Choose ω_0 and ω_1 so that $f_\omega(x)$ is close to y
- Cost function $J(\omega) =$
- How to calculate ω?
  - GD
  - OLS

Cost function intuition

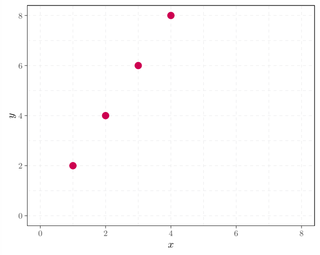
Simple linear regression

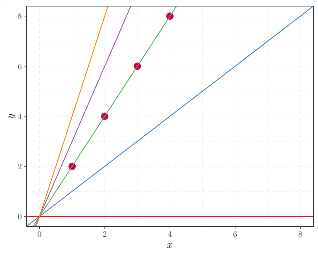
- Model: $f_\omega(x) = \omega_0 + \omega_1 x = \omega' x$
- Parameters: ω_0 and ω_1
- Cost function: $J(\omega_0, \omega_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$
- Goal: $\min_{\omega_0, \omega_1} J(\omega_0, \omega_1)$

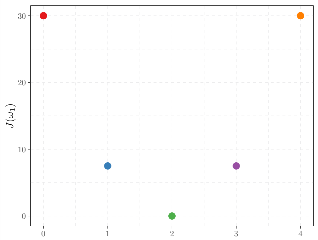
Suppose a simplified hypothesis (with 1 parameter):

- Model: Let $f_\omega(x) = \omega_1 x = \omega' x$
- Parameter: ω_1
- Cost function: $J(\omega_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$

Let the following example:

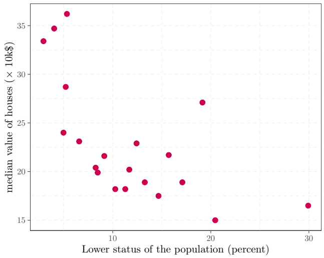


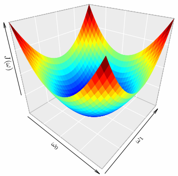


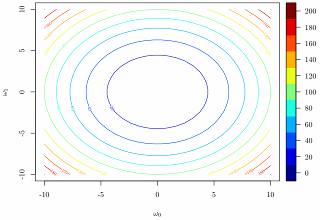


Simple linear regression

- Model: $f_\omega(x) = \omega_0 + \omega_1 x = \omega' x$
- Cost function: $J(\omega_0, \omega_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$

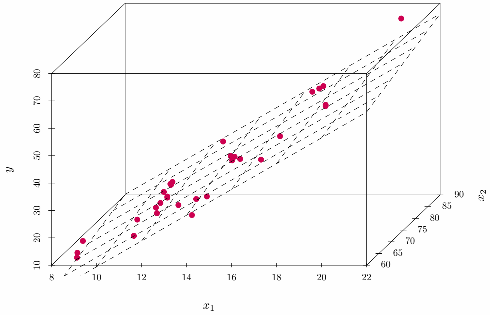






- Let p features: $x_1, x_2, \ldots, x_p$
- Multiple linear regression: $f(x) = f_\omega(x) = \omega_0 + \omega_1 x_1 + \ldots + \omega_p x_p$

Linear Regression with 2 features



Multiple Linear Regression

- Let p variables: $x_1, x_2, \ldots, x_p$
- Multiple linear regression: $f(x) = f_\omega(x) = \omega_0 + \omega_1 x_1 + \ldots + \omega_p x_p$
- Define $x_0 = 1$, and

$$\omega = \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_p \end{pmatrix} \quad x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$$

- Using matrices: $f_\omega(x) = \omega' x$
- Methods to estimate ω:
  - OLS
  - GD
- Cost function $J(\omega) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega \left(x^{(i)} \right) - y^{(i)} \right)^2$

Gradient descent

Gradient descent: algorithm

- Let a function $J(\theta)$
- **Goal**: Find θ that minimizes $J(\theta)$, e.g. $\theta = \text{argmin}_\theta J(\theta)$
- **Algorithm**:
  - `initialize` θ `randomly`
  - `repeat until convergence`$\{$

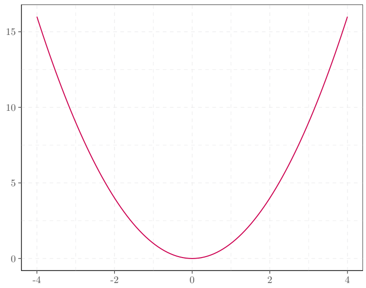
$$\theta^{new} = \theta^{old} - \alpha J'(\theta)$$

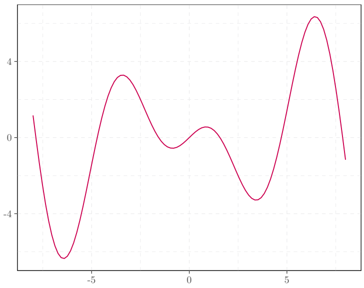
$\}$

- α is the learning rate

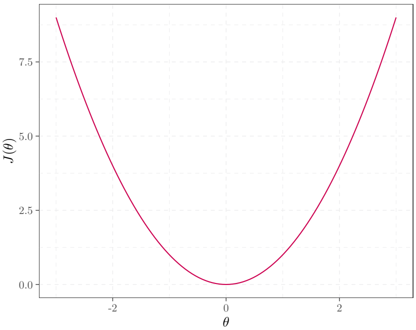
Convex function

- f is convex if $f(\lambda x_1 + (1-\lambda)x_2) \leqslant \lambda f(x_1) + (1-\lambda) f(x_2)$, $\forall x_1$ and $x_2 \in d_f, \lambda \in (0,1)$.
- f is convex iff $f'' \geqslant 0$
- A convex funtion has a global minimum

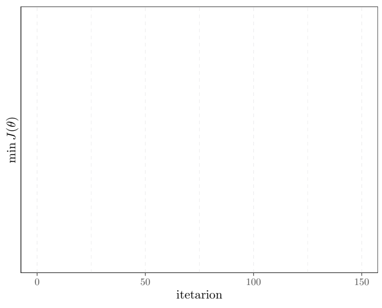


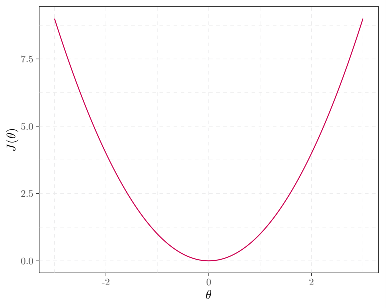


- Let $J(\theta) = \theta^2$
- So $J'(\theta) = 2\theta$
- Let $\alpha = 0.1$



- $J(\theta)$ must decrease after each iteration
- Define the convergence





- If α is too small, slow convergence
- If α is too large, convergence is not guaranteed

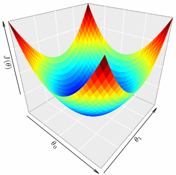
- Let a function $J(\theta_0, \theta_1)$
- **Goal**: find (θ_0, θ_1) that minimize $J(\theta_0, \theta_1)$, e.g. $\text{argmin}_{(\theta_0,\theta_1)} J(\theta_0, \theta_1)$
- **Algorithm**:
  - `initialize` (θ_0, θ_1) `randomly`
  - `repeat until convergence`$\{$

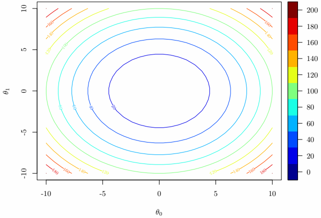
$$\theta_0^{new} = \theta_0^{old} - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

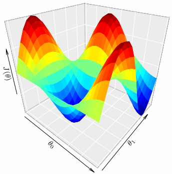
$$\theta_1^{new} = \theta_1^{old} - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$\}$

- α is the learning rate
- Same principle if J is a function of more variables


$J(\theta)$
θ_0
θ_1


θ_1
θ_0
10
5
0
-5
-10
-10
-5
0
5
10
200
180
160
140
120
100
80
60
40
20
0


$J(\theta)$
θ_0
θ_1

Simple linear regression

- Model: $f_\omega(x) = \omega_0 + \omega_1 x = \omega' x$
- Parameters: ω_0 and ω_1
- Cost function: $J(\omega_0, \omega_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$
- Goal: $\min_{\omega_0, \omega_1} J(\omega_0, \omega_1)$

Algorithm

- `initialize` (ω_0, ω_1) `randomly`
- `repeat until convergence`$\{$

$$\omega_i^{new} = \omega_i^{old} - \alpha \frac{\partial}{\partial \omega_i} J(\omega_0, \omega_1)$$

for $i = 0$ and $i = 1$

Algorithm

- `initialize` (ω_0, ω_1) `randomly`
- `repeat until convergence`$\{$

$$\omega_0^{new} = \omega_0^{old} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\omega_1^{new} = \omega_1^{old} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right) . x^{(i)}$$

$\}$

Multiple linear regression

- Model: $f_\omega(x) = \omega_0 + \omega_1 x_1 + \ldots + \omega_p x_p = \omega' x$
- Parameters: $\omega_0, \omega_1, \ldots, \omega_p$
- Cost function: $J(\omega) = \frac{1}{2n} \sum_{i=1}^{n} \left(f_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$

Algorithm

- `initialize the` ω_i `randomly`
- `repeat until convergence{`

$$\omega_i^{new} = \omega_i^{old} - \alpha \frac{\partial}{\partial \omega_i} J(\omega) \quad \text{simultaneously for every } i = 0, \ldots, p$$

}

Algorithm

- `initialize the` ω_i `randomly`
- `repeat until convergence{`

$$\omega_0^{new} = \omega_0^{old} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(f_\omega \left(x^{(i)} \right) - y^{(i)} \right) . x_0^{(i)}$$

$$\omega_1^{new} = \omega_1^{old} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(f_\omega \left(x^{(i)} \right) - y^{(i)} \right) . x_1^{(i)}$$

$$\vdots$$

$$\omega_p^{new} = \omega_p^{old} - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(f_\omega \left(x^{(i)} \right) - y^{(i)} \right) . x_p^{(i)}$$

}

Remarks

Gradient descent

- Gradient descent ("Batch" version): each step uses all the training examples
- Features must be scaled
- We must choose α
- There is more advanced gradient based algorithms

Normal equation

- OLS leads to an analytical solution
- $\theta = (X'X)^{-1}X'y$
- No need to choose α neither to iterate
- Need to compute $(X'X)^{-1}$
- Slow if p is large
- What if $(X'X)^{-1}$ is non-invertible?

Regression: Some important questions

When we perform multiple linear regression, we usually are interested in answering a few important questions.

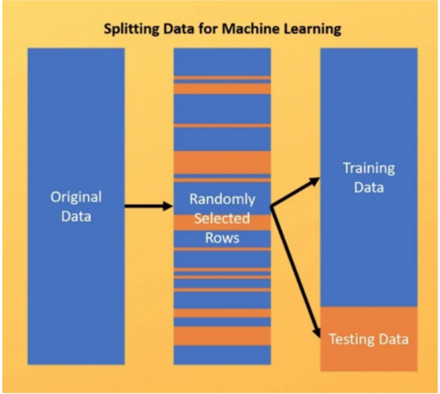
1. Is at least one of the predictors $X_1, X_2, \ldots, X_p$ useful in predicting the response?
2. Do all the predictors help to explain y, or is only a subset of the predictors useful?
3. How well does the model fit the data?
4. Given a set of predictor values, what response value should we predict, and how accurate is our prediction?

	Coefficient	Std. error	t-statistic	p-value
Constant	2.939	0.3119	9.42	<0.0001
X_1	0.046	0.0014	32.81	<0.0001
X_2	0.189	0.0086	21.89	<0.0001
X_3	-0.001	0.0059	-0.18	0.8599

In this table we have the following model

$$Y = 2.939 + 0.046X_1 + 0.189X_2 - 0.001X_3$$

Assessing model accuracy & Bias/Variance Trade-off



Comment peut-on mesurer la performance d'un modèle sur des données connues ?

Risque empirique du modèle

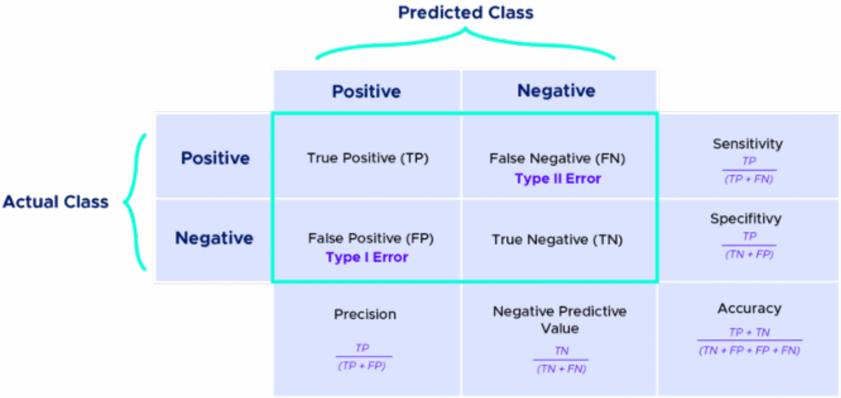
Problème de **classification** : La proportion de points que le modèle a mal étiqueté.

Problème de **régression** : La moyenne des erreurs quadratiques.

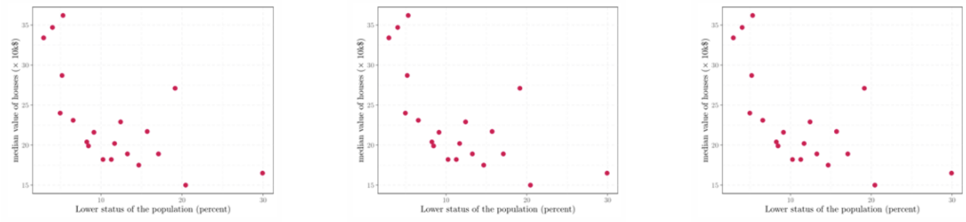
Regression

MSE (Mean Squared Error) $= \frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)}) - y^{(i)})^2$

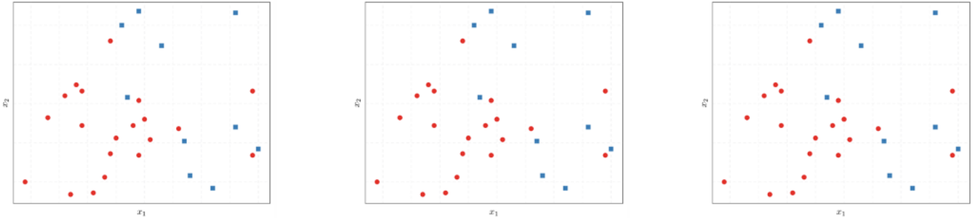
Model accuracy: Classification



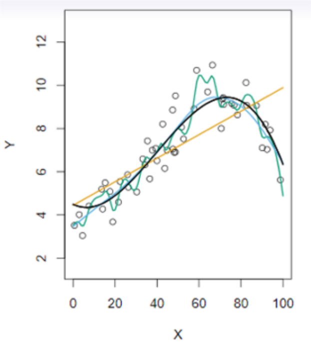
Régression



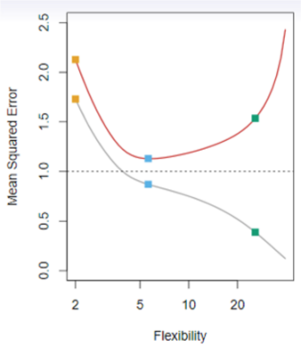
Classification



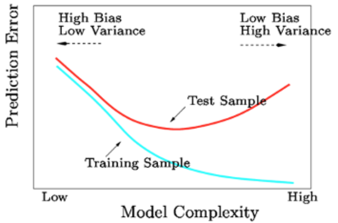
Bias/Variance Trade-off (Underfitting & Overfitting)



Black: Truth
Orange: Linear Estimate
Blue: smoothing spline
Green: smoothing spline (more flexible)



RED: Test MES
Grey: Training MSE
Dashed: Minimum possible test MSE (irreducible error)



We must always keep this picture in mind when choosing a learning method. More flexible/complicated is not always better!