# Machine Learning

Bertrand Borel, Loïc Plessis, Pauline Sanchez, Victor Lannurien.

# TABLE DES MATIÈRES

# Définition du machine learning :

- Science moderne permettant de découvrir des patterns et d’effectuer des prédictions à partir de données
- Basé sur des statistiques, du forage de données, de la reconnaissances de patterns et des analyses prédictives
- Très efficace dans les situations où les insights doivent être découvertes à partir de larges ensembles de données diverses et changeantes (Big Data)
- Branche de l’intelligence artificielle englobant de nombreuses méthodes. Ces méthodes sont des algorithmes
- Un système ML ne suit pas d’instructions, mais apprend à partir de l’expérience.
- Ses performances s’améliorent au fil de son  “entraînement”

# Apprentissage supervisé

- Modèle d’apprentissage le plus populaire en ML
- Consiste à superviser l’apprentissage de la machine en lui montrant des exemples (des données) de la tâche qu’elle doit réaliser
- Fonctionne en 4 étapes :
    - Importer un Dataset (x, y) qui contient les exemples
    - Développer un **Modèle** aux paramètres aléatoires
    - Développer une **Fonction Coût** qui mesure les erreurs entre le modèle et le Dataset
    - Développer un **Algorithme d’apprentissage** pour trouver les **paramètres** du modèle qui **minimisent** la **Fonction Coût**

# Dataset (features, target)

## Les Features

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
| 2 | 1 | 0 | 3 | Braund, Mr. ( | male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 3 | 2 | 1 | 1 | Cumings, Mr: | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 4 | 3 | 1 | 3 | Heikkinen, M | female | 26 | 0 | 0 | STON/O2. 31 | 7.925 | | S |
| 5 | 4 | 1 | 1 | Futrelle, Mrs | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 6 | 5 | 0 | 3 | Allen, Mr. Wi | male | 35 | 0 | 0 | 373450 | 8.05 | | S |
| 7 | 6 | 0 | 3 | Moran, Mr. J | male | | 0 | 0 | 330877 | 8.4583 | | Q |

# Target

Prenons un exemple : on cherche à déterminer si une personne, dans la base de données d'un magasin de chaussures, va cliquer sur le lien contenu dans le mail promotionnel qui lui a été envoyé. Pour cela, nous observons dans un premier temps l'historique, dont voici un extrait :

| Premier email promotionnel | Âge | A cliqué |
|---|---|---|
| OUI | 52 | OUI |
| NON | 19 | NON |
| NON | 37 | NON |
| OUI | 25 | OUI |

# **Modèle** (régression linéaire simple, polynomiale, multiple)

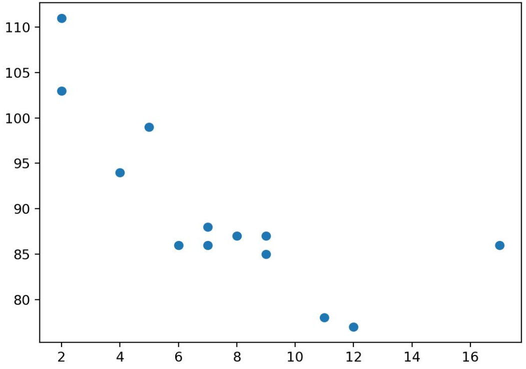
Régression linéaire:

Il existe 3 types de régression linéaire:

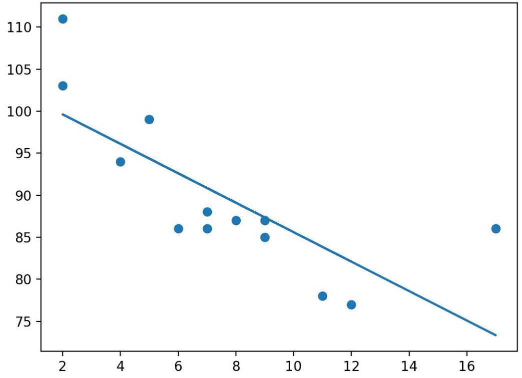
- la régression linéaire simple
- la régression linéaire polynomiale
- la régression linéaire multiple

# Modèle : régression linéaire simple

```python
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```




```python
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```



# Modèle : régression linéaire simple

```python
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

print(r)
```

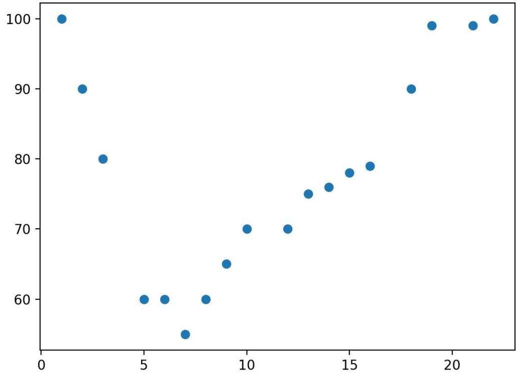
```
-0.758591524376155
```

```python
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

speed = myfunc(10)

print(speed)
```

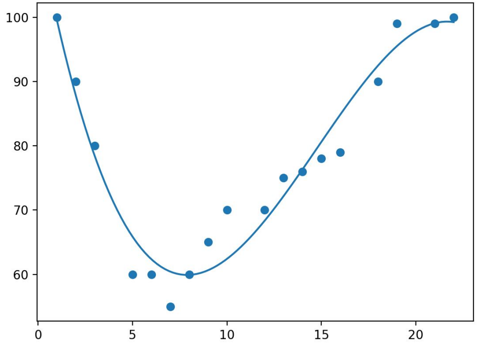
```
85.59308314937454
```

# Modèle : régression linéaire polynomiale

```python
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

plt.scatter(x, y)
plt.show()
```




```python
import numpy
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(1, 22, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```



# Modèle : régression linéaire polynomiale

```python
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

print(r2_score(y, mymodel(x)))
```

```
0.9432150416451027
```

```python
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

speed = mymodel(17)
print(speed)
```

```
88.87331269697987
```

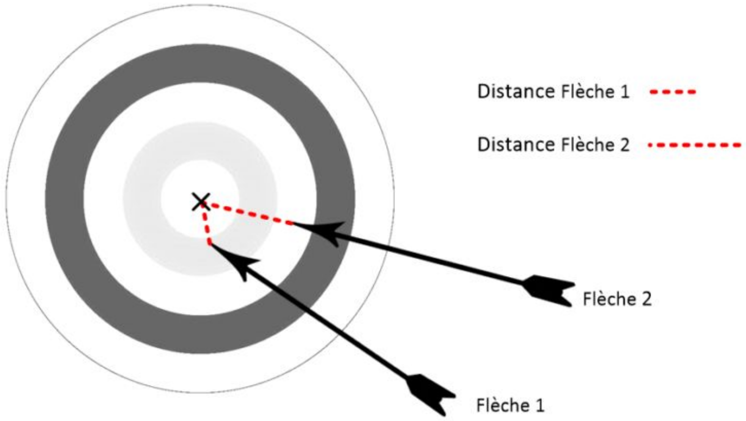
# Modèle : régression linéaire multiple

Exactement la même chose que pour les deux autres régressions précédentes mais basée sur 2 variables ou plus.

# Fonction de coût (l'erreur quadratique moyenne)

Pour savoir quel modèle est le meilleur parmi 2 candidats, il faut les évaluer. Pour cela, on mesure l'erreur entre un modèle et le Dataset, et on appelle ça la **Fonction Coût.**

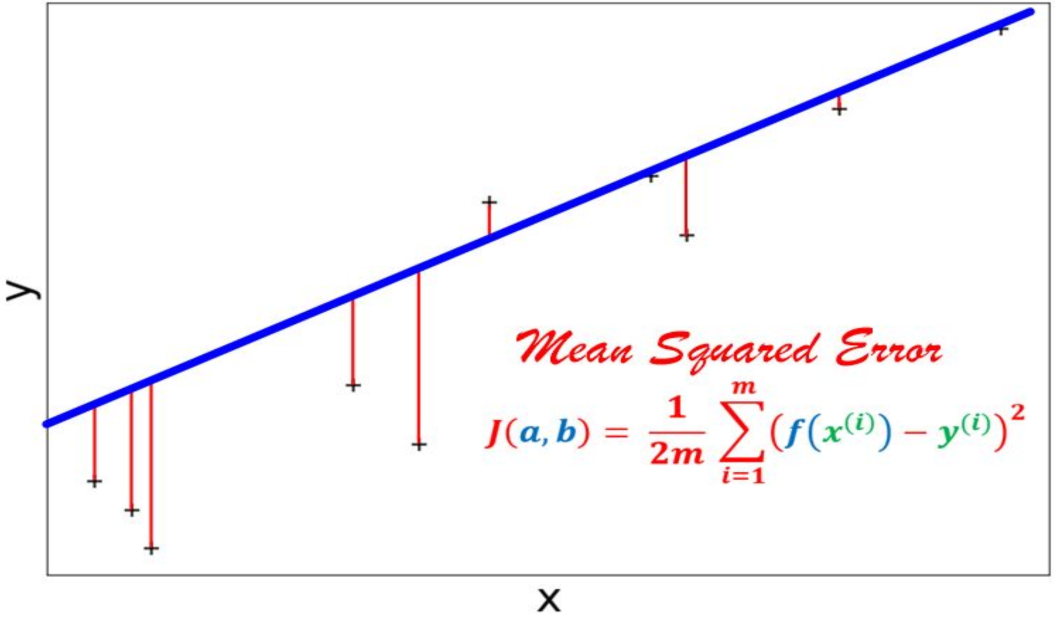
Dans le cas d'une régression, on peut par exemple mesurer l'erreur entre la prédiction du modèle et la valeur qui est associée à ce dans notre Dataset. C'est similaire à l'idée de mesurer la distance entre votre flèche () et le centre de la cible, qui n'est autre que le point () qu'elle est sensée atteindre.



# L’erreur quadratique moyenne

$$\mathrm{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} (\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}})^2$$

**l’Erreur Quadratique Moyenne** ou autremment appelé anglais la fonction **Mean Squared Error.**



# Gradient en ML

Repose sur la notion de dérivée

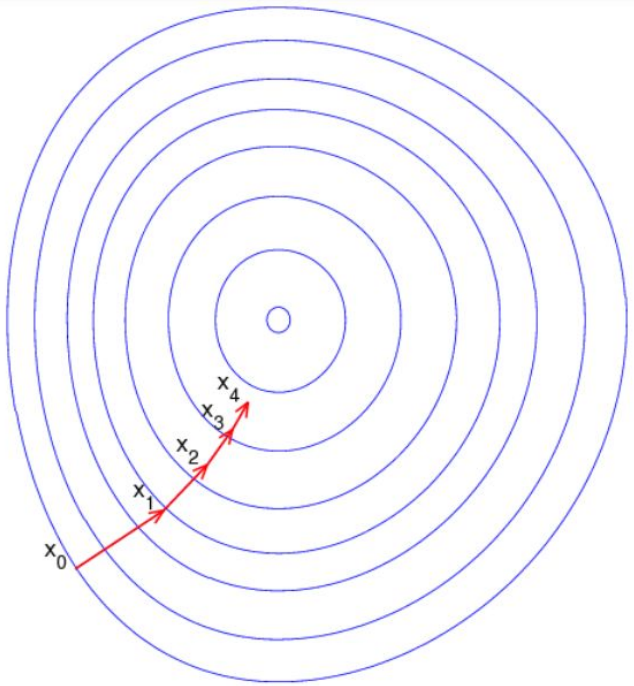
Un gradient est un dérivé d'une fonction qui

a plus d'une variable d'entrée.

En lien avec le calcul vectoriel.

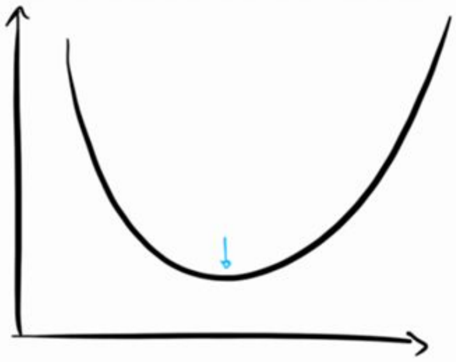
Permet de calculer la pente locale de la fonction

= progression “pas-à-pas”

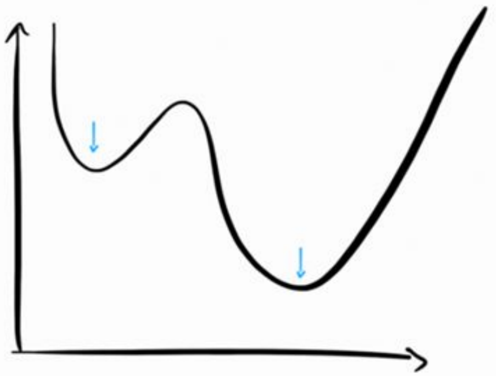


# Algorithme de minimisation (Descente de gradient)

Algorithme d’optimisation permettant de trouver le minimum d’une fonction convexe.



Fonction Convexe



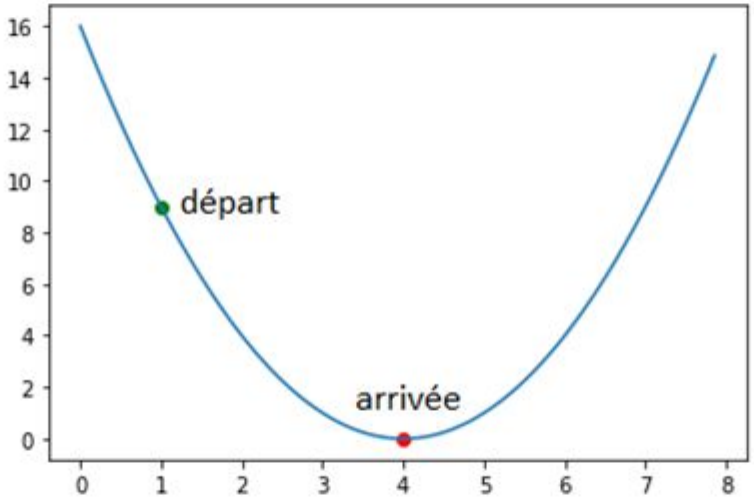
Fonction Non-Convexe

Fonction convexe = un minimum global

Fonction non-convexe = plusieurs minimums locaux

Sert pour  la minimisation de la fonction coût : la machine apprend et trouve le meilleur modèle

## L'algorithme permet de trouver la valeur idéale entre le départ et l'arrivée.




Calcul d'une dérivée (pour la courbe)

valeur négative en pente / valeur positive en montée

Importance de l'hyper-paramètre Alpha

# **Coefficient de détermination** (linéaire de Pearson)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

- Noté $R^2$, c'est une mesure de la qualité de la prédiction d'une régression linéaire
- C'est le carré de la corrélation de Pearson entre les vraies valeurs et les valeurs prédites
- Permet d'indiquer à quel point les valeurs prédites sont corrélées aux vraies valeurs

# Merci de votre attention