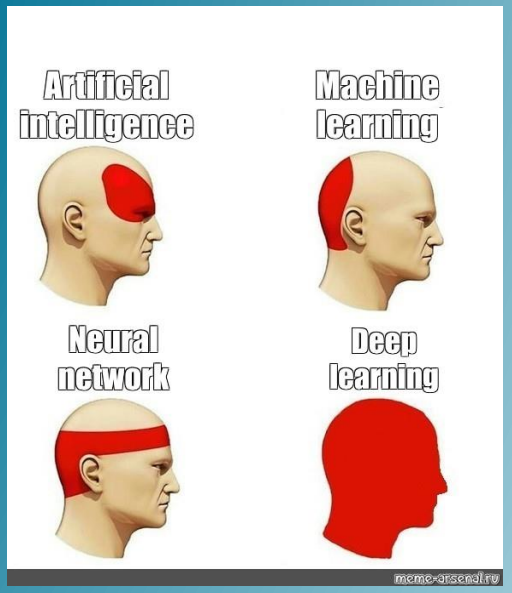
M2 Atmosphère/Climat

Méthodes Numériques et Modélisation

Fondements de l'intelligence artificielle et applications aux sciences atmosphériques



LiCIS LRC CEA DiGiT
CALCUL INTENSIF ET IMAGE POUR LA SIMULATION

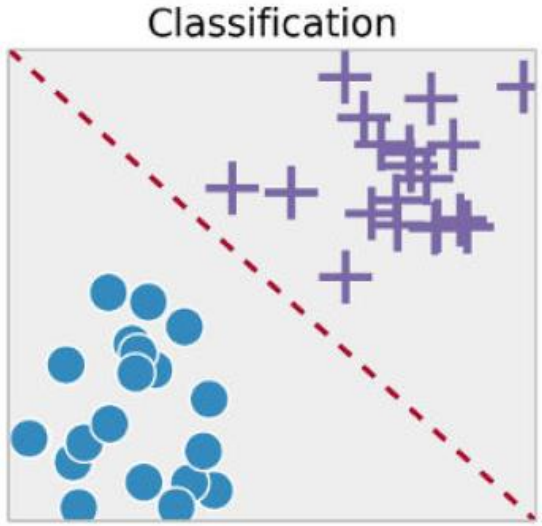
UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

UR | UNIVERSITÉ DE LA RÉUNION

CLASSIFICATION

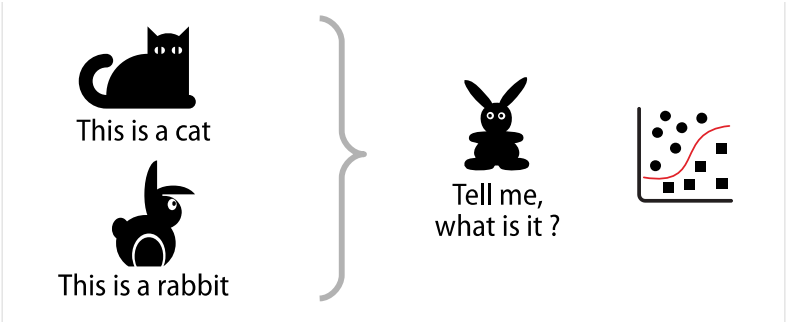
- Associer à chaque donnée un label (étiquette) parmi un ensemble de labels possibles
- Classification binaire
  - Deux classes -> on peut faire un "est-ce X" oui ou non
- Classification multiclasse
  - Plusieurs méthodes
    - Binaire pour chaque classe
    - "vrai" multiclasse -> probabilité générale

- Cas particuliers
  - Classification multilabel
    - Une donnée peut appartenir à plusieurs classes
  - Détection d'objets
    - Classer des éléments d'une image (et pas toute l'image)
  - Segmentation
    - Chaque pixel de l'image sera classé (puis associés)



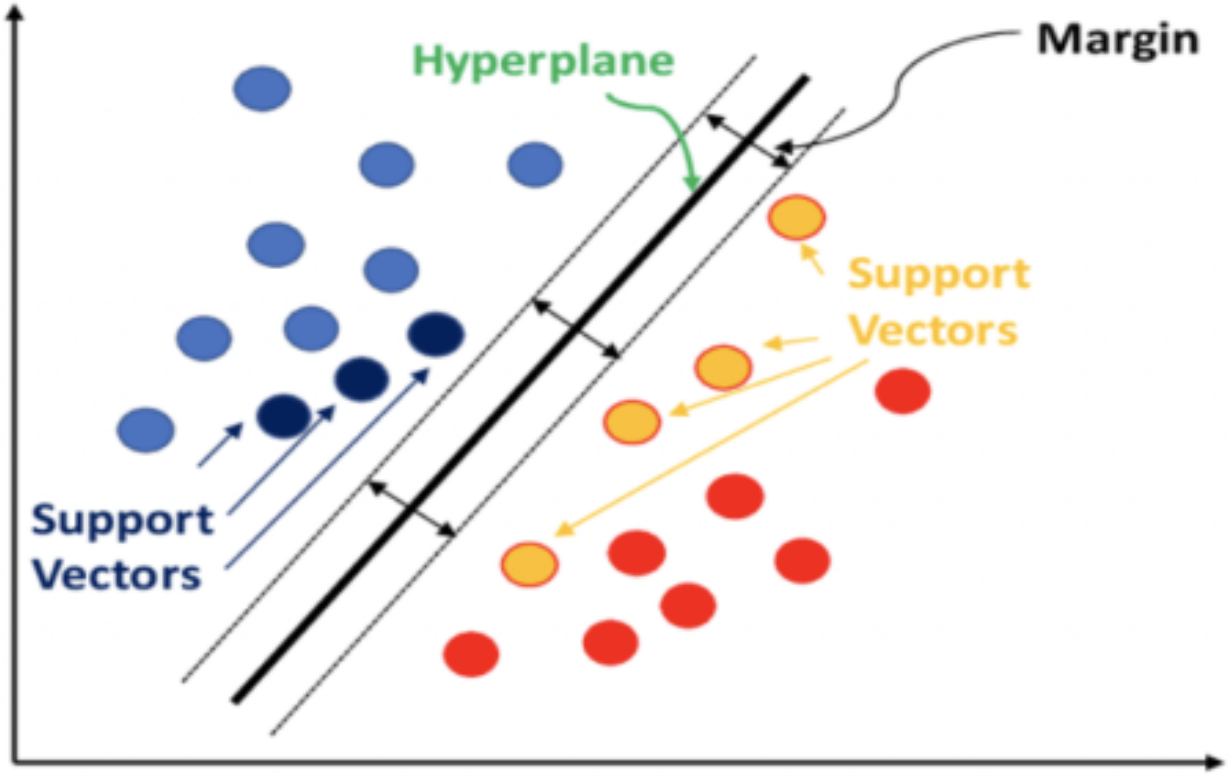
Classification

Prédire une classe (qualitative, discrète)



WHAT IS A

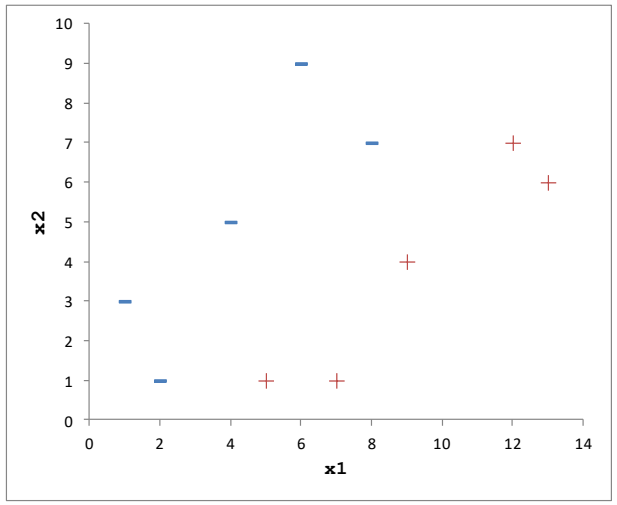
SUPPORT VECTOR MACHINE?

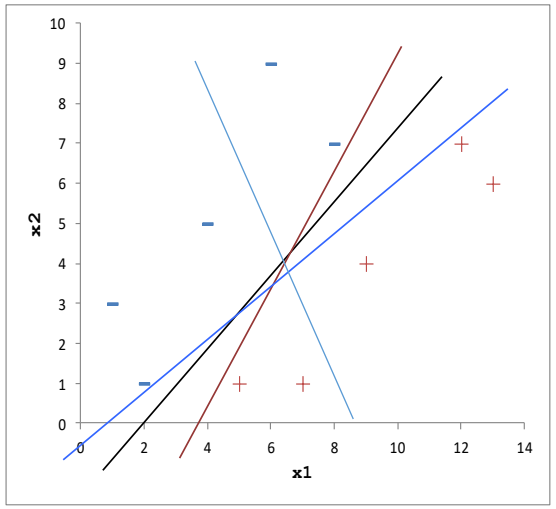


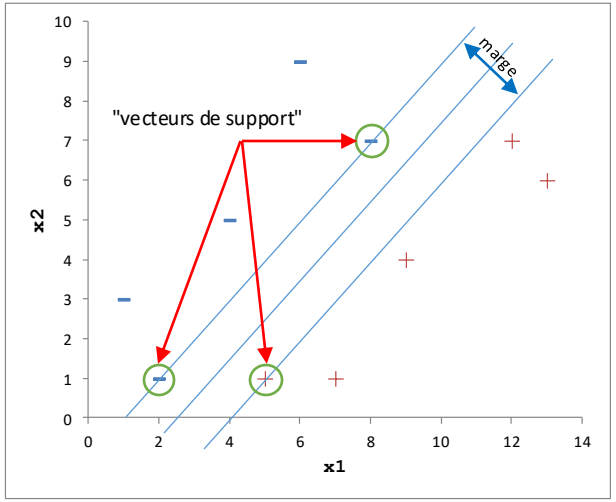
Les SVM

Les SVM

- Les "Support Vector Machines" sont des méthodes proches de l'algèbre linéaire très populaires à la fin des années 90, début 2000
- À la base, le principe est proche de celui de la régression logistique simple, mais associée à une recherche d'une marge de sécurité plus grand
  - Il ne suffit pas de trouver la fonction de séparation, il faut trouver la meilleure fonction
  - doit maximiser la distance entre la frontière de séparation et les points de chaque classe qui lui sont le plus proche
  - D'où le surnom en français "Séparateur à Vastes Marges"

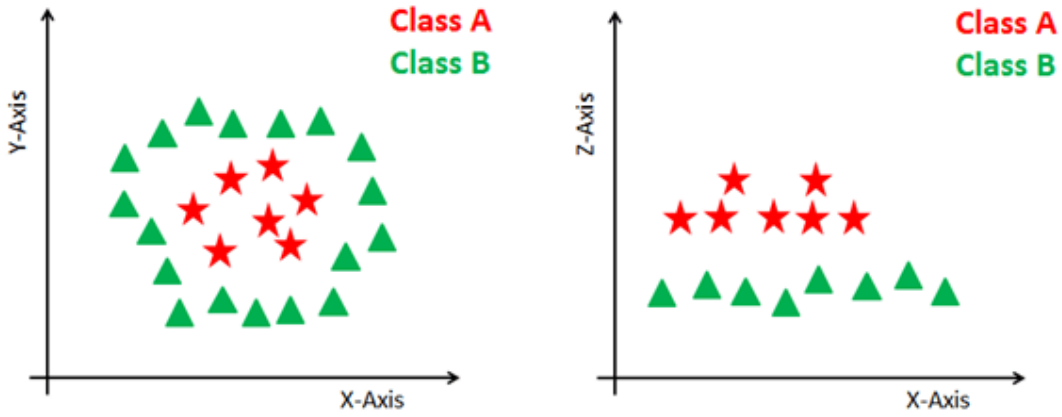






FAIRE APPEL À UNE DIMENSION SUPÉRIEUR

- L'astuce de SVM pour traiter des données qui ne sont pas séparables avec une ligne est de chercher de l'ordre dans une dimension supérieure
  - Ex : on rajoute une troisième colonne Z avec la formule $z = x^2 + y^2$
- Cette petite modification permet à SVM de trouver un hyperplan quand on regarde les données plotées sur les axes X-Z



- Différents types d'opération peuvent être utilisées pour créer ces dimensions supérieures

Les astuces des SVM (Kerne tricks)

- Dans le cas d'un SVM linéaire, le plan de coupe (hyperplan) est déterminé grâce à de l'algèbre linéaire
  - Ex : f(x) = aX + b
- un SVM linéaire peut être obtenu à partir du produit intérieur de deux observations
  - Ex : le produit intérieur de deux vecteurs [2,3] et [5,6] est 2*5+3*8=28
  - Donc on peut utiliser une fonction f(x) = somme(ai*(x,xi))+b
  - Le terme "Kernel" définit alors ce produit intérieur : K(x,xi) = somme(x * xi)
  - On obtient ainsi un **"Kernel trick**" très populaire lorsqu'on a un grand nombre de features car ce calcul est rapide

Autres Kernel Tricks

- Kernel Trick Polynomial -> **F(x, xj) = (x.xj+1)^d**
  - Moins populaire, ici d indique le degré du polynôme
- Gaussian Radial Basis Function (RBF) -> **F(x, xj) = exp(-gamma * ||x - xj||^2)**
  - L'un des plus populaires, il faut donner une valeur gamma entre 0 et 1 (souvent 0.1)
- Sigmoid -> **F(x, xj) = tanh(αxay + c)**
  - C'est la fonction utilisée dans les réseaux de neurones
- Mais aussi Gaussian, Bessel, ANOVA…

- Le kernel linéaire est préféré pour les problèmes de classification de texte car il fonctionne bien pour les grands ensembles de données
- Les kernels gaussiens donnent de bons résultats lorsqu'il n'y a pas d'informations supplémentaires concernant les données
- Le kernel RBF est comme un kernel gaussien qui projette les données à de grandes dimensions et recherche ensuite une séparation linéaire pour celles-ci
- Les kernels polynomiaux donnent de bons résultats pour les problèmes où toutes les données d'apprentissage sont normalisées

SVM avec Scikit Learn

- L'exemple de l'Iris, bien sûr ;)
  - Tout d'abord, les import, la lecture des données et la prépa pour la visualisation

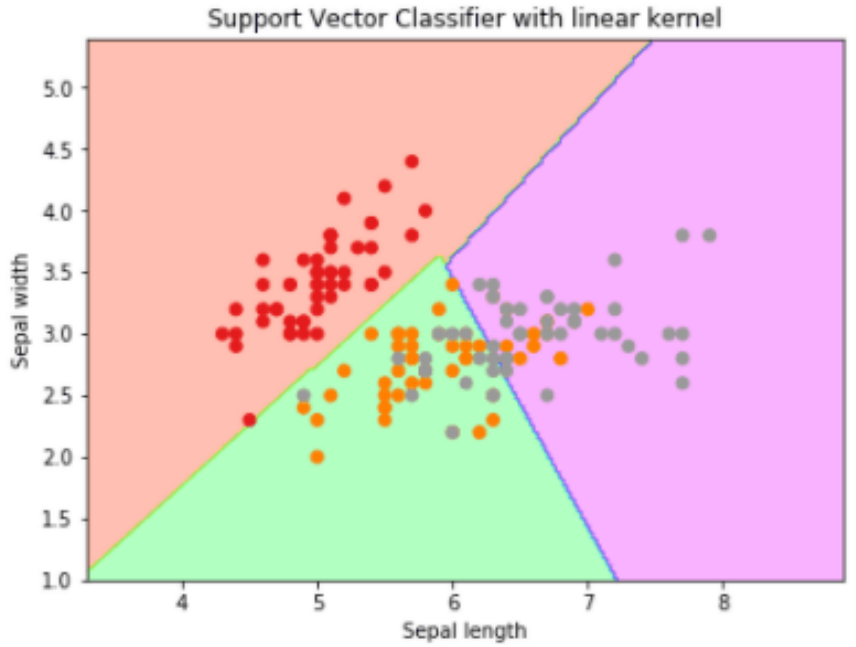
```
## Requried Python Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

## Load iris dataset
iris = datasets.load_iris()

## Create features and target data
X = iris.data[:, :2]
y = iris.target

## Plotting
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
X_plot = np.c_[xx.ravel(), yy.ravel()]
```

SVM avec Scikit Learn

- Un Linear Kernel

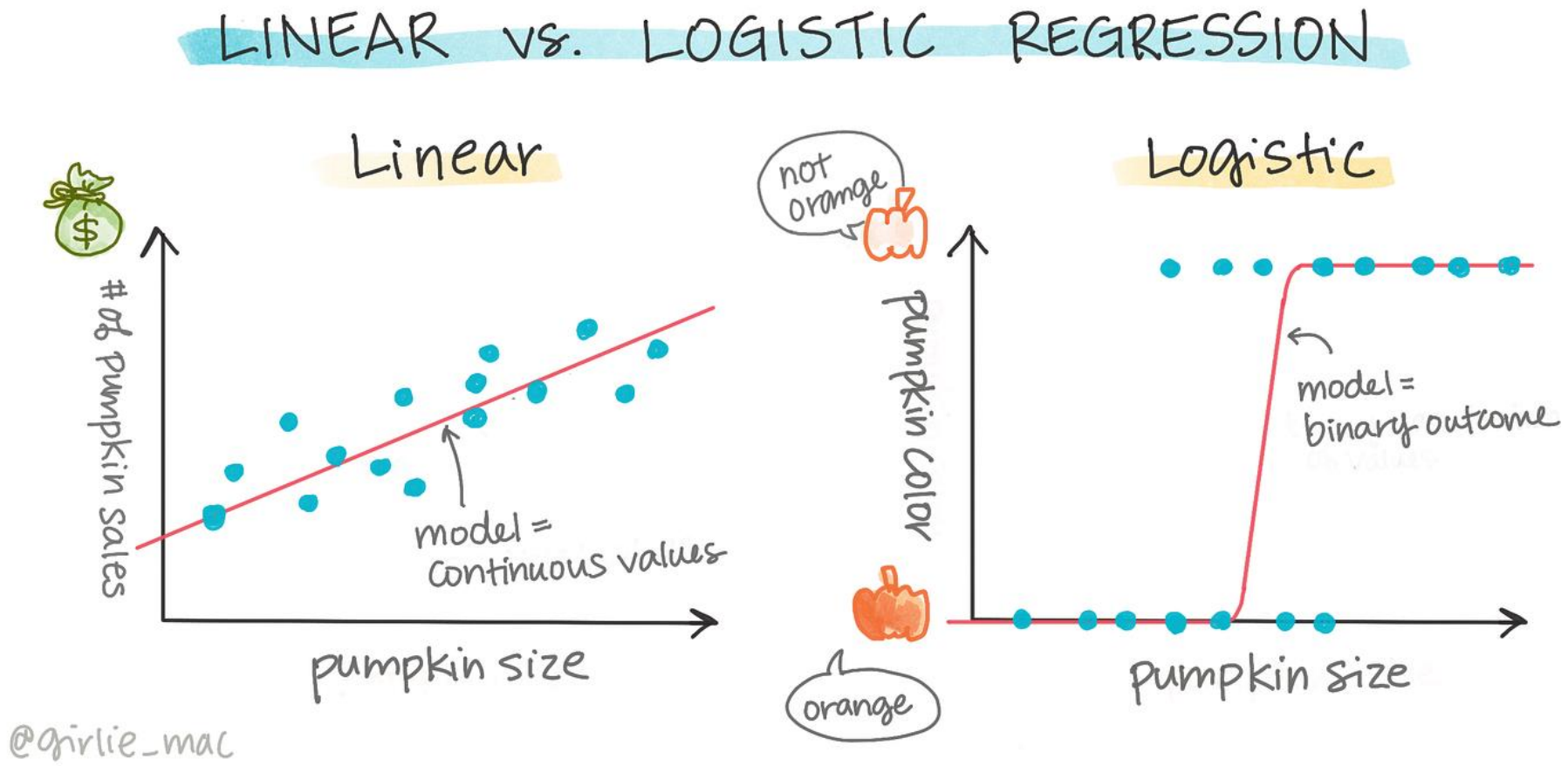
```
## Creating the linear kernel
svc_classifier = svm.SVC(kernel='linear', C=C).fit(X, y)
C = 1.0
Z = svc_classifier.predict(X_plot)
Z = Z.reshape(xx.shape)

## Code of plotting
plt.figure(figsize=(15, 5))
plt.subplot(121)
plt.contourf(xx, yy, Z, alpha=0.3)
plt.set_cmap("gist_rainbow")
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('Support Vector Classifier with linear kernel')
```

```
Text(0.5, 1.0, 'Support Vector Classifier with linear kernel')
```



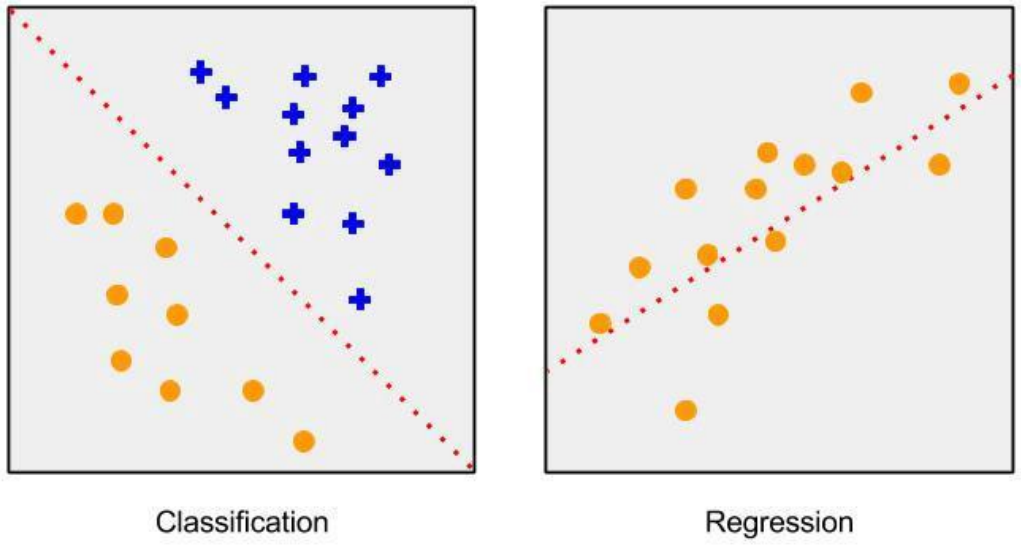

do you want to enforce social distancing?
yes, please
use SVM to maximize the margin

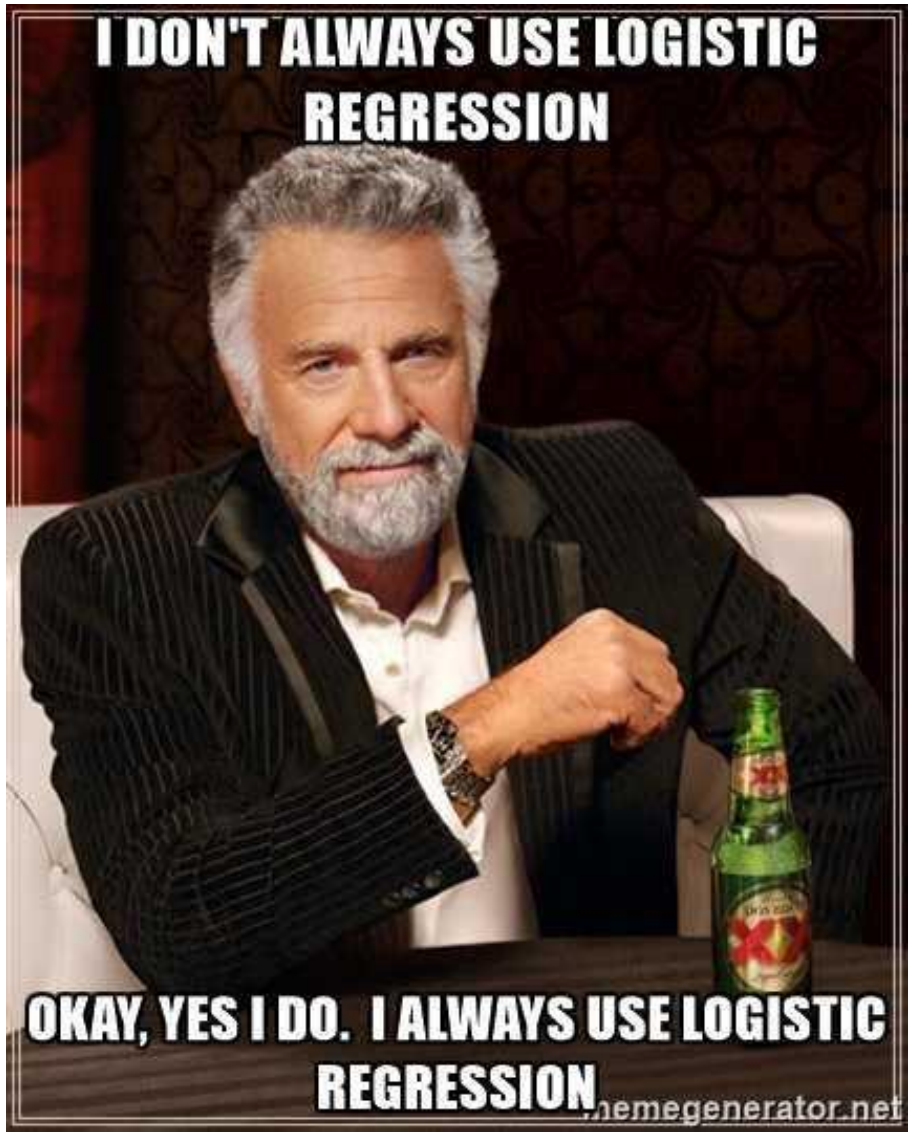
Régression Logistique



Régression Logistique

- La base de la régression logistique est de créer une fonction probabiliste permettant d'identifier les données (surtout pour la classification)





EST-CE QU'IL PLEUT DEMAIN ?

x_i = features for day i

y_i = 1, yes

y_i = 0, no

features

Cloud Cover	Humidity	Temperature	Air Pressure
0.5	80%	75	1.2
0.2	95%	83	1.3

outcome

Did it Rain
1
0

WEATHER

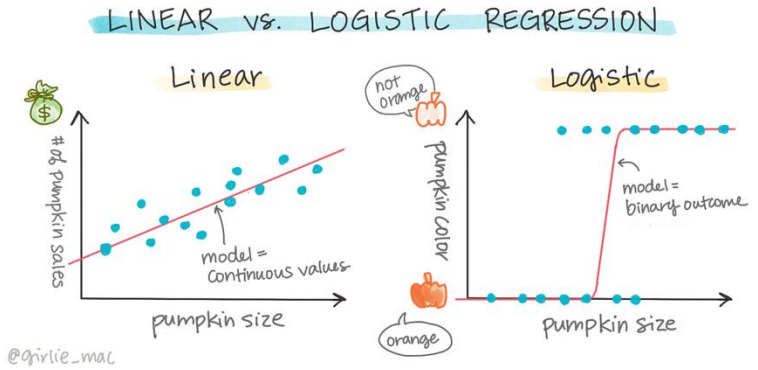
RAIN

$$z_1 = (b_1 \text{ X } 0.5) + (b_2 \text{ X } 0.8) + (b_3 \text{ X } 75) + (b_4 \text{ X } 1.2) + b_0 \quad y_1 = 1$$

$$z_2 = (b_1 \text{ X } 0.2) + (b_2 \text{ X } 0.95) + (b_3 \text{ X } 83) + (b_4 \text{ X } 1.3) + b_0 \quad y_2 = 0$$

On aurait pu faire un système d'équations linéaires, mais beaucoup de cas "moyens" serait dans l'impasse -> baisse sensibilité/spécificité

On peut encourager les décisions avec une fonction non linéaire

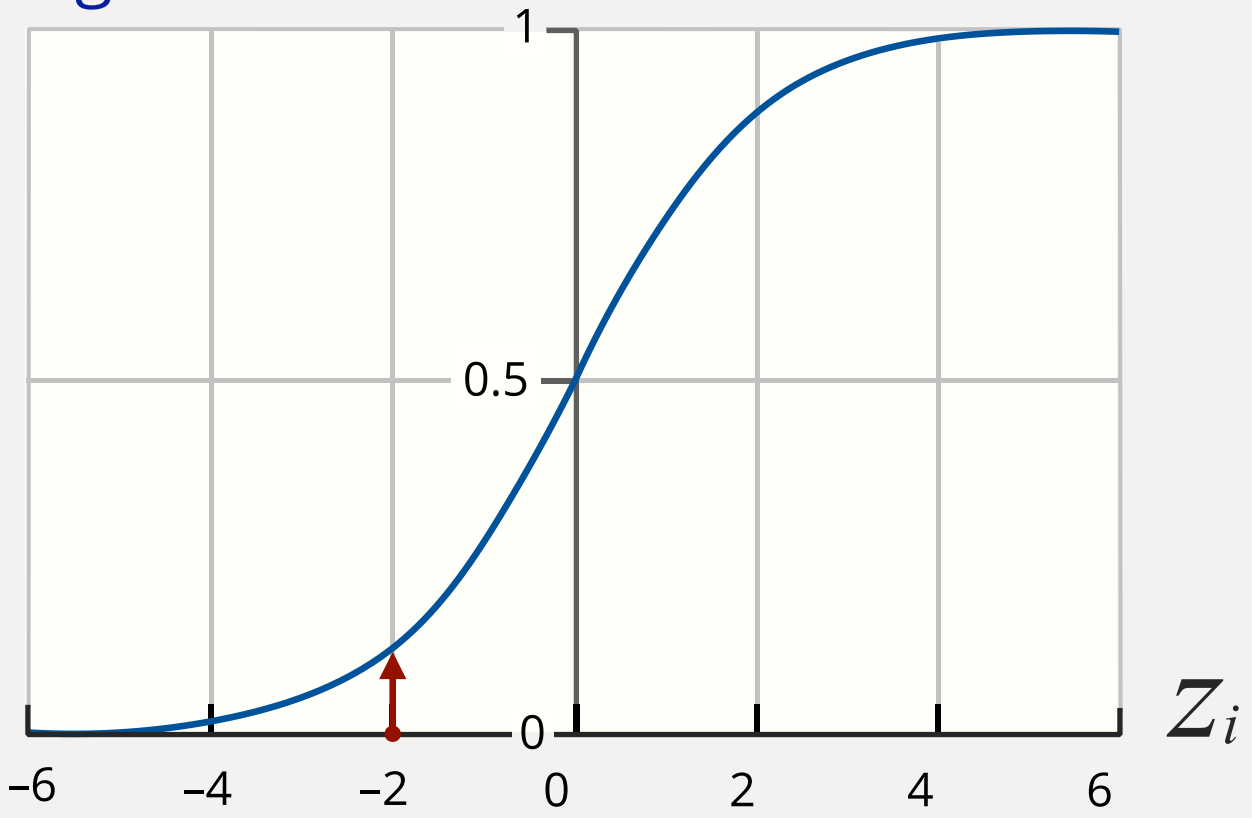


Si on transforme tout ça en probabilités

- La fonction sigmoide - **σ(*zi*)** - est une manière de représenter les prédictions selon un point de vue probabiliste
  - Plus grand et positif est *zi*, plus probable sera la valeur *yi*
  - Plus petit et négatif est *zi*, moins probable sera la valeur *yi*

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) + b_0$$

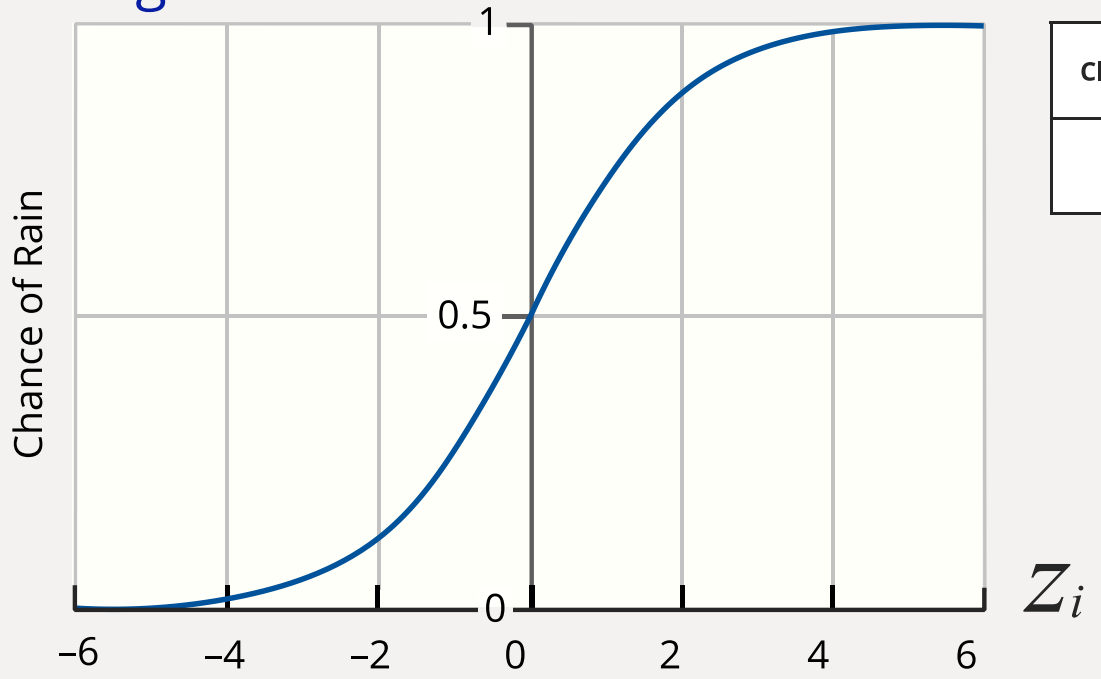
Sigmoid Function $p(y_i = 1|x_i) = \sigma(z_i)$



EXEMPLE

$$z_i = (b_1 \times 0.5) + (b_2 \times 0.8) + (b_3 \times 75) + (b_4 \times 1.2) + b_0$$

Sigmoid Function $p(y_i = 1|x_i) = \sigma(z_i)$



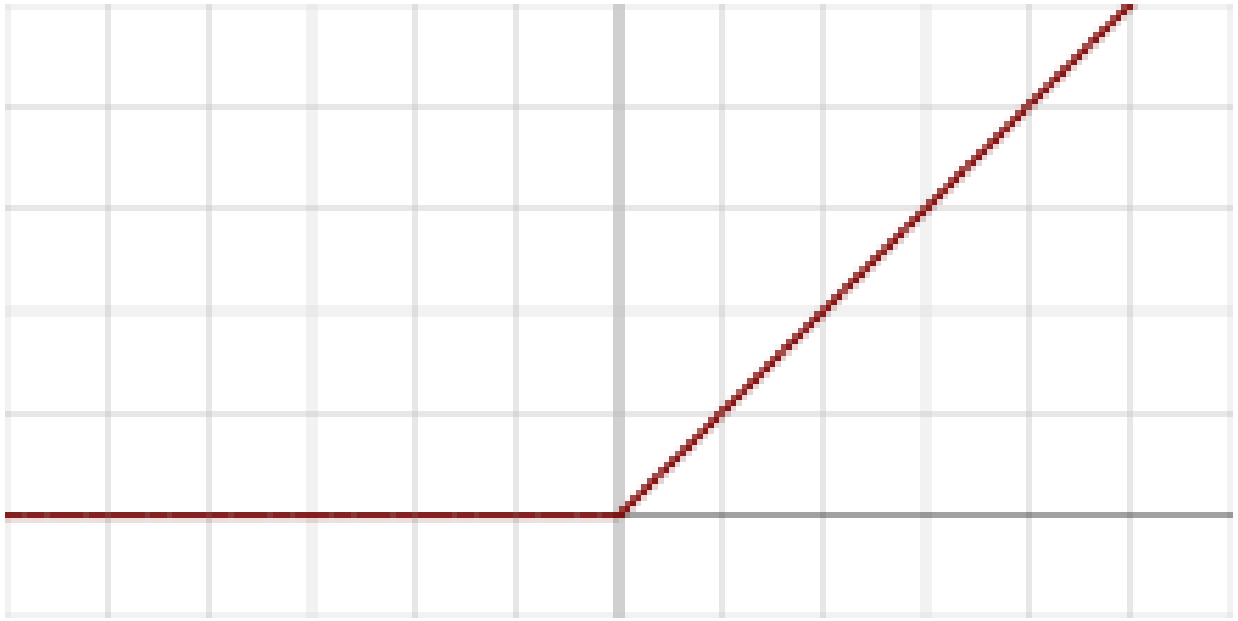
features

Cloud Cover	Humidity	Temperature	Air Pressure
0.5	80%	75	1.2

- Les paramètres *b* indiquent combien les variables sont importantes pour la prédiction

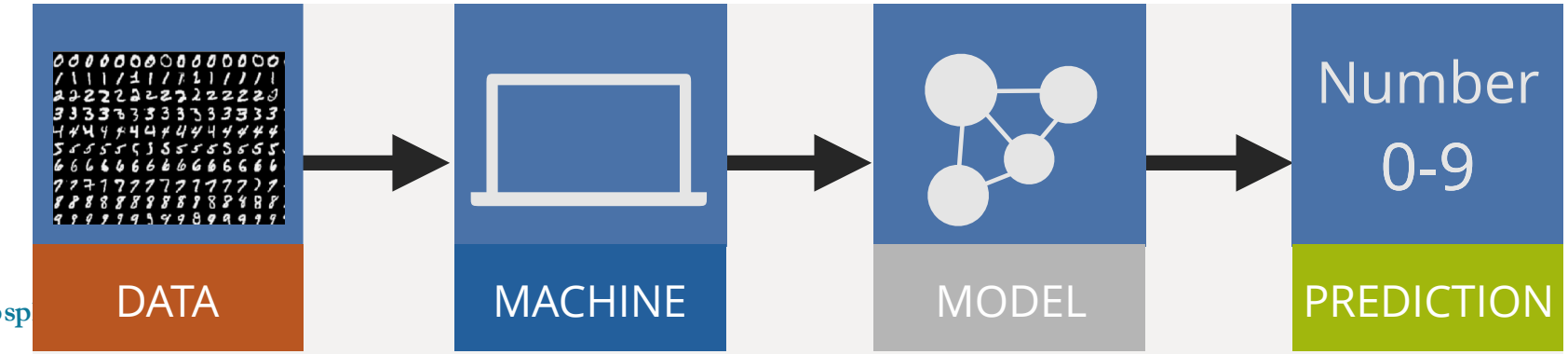
ReLU (Rectified Linear Unit)

- Devenue très populaire, ReLU remplace sigmoid
  - introduit de la non-linéarité dans le modèle
  - convergence plus rapide qu'avec sigmoïde
  - le calcul de ReLU est simples à effectuer et ne nécessite pas d'exponentielles coûteuses



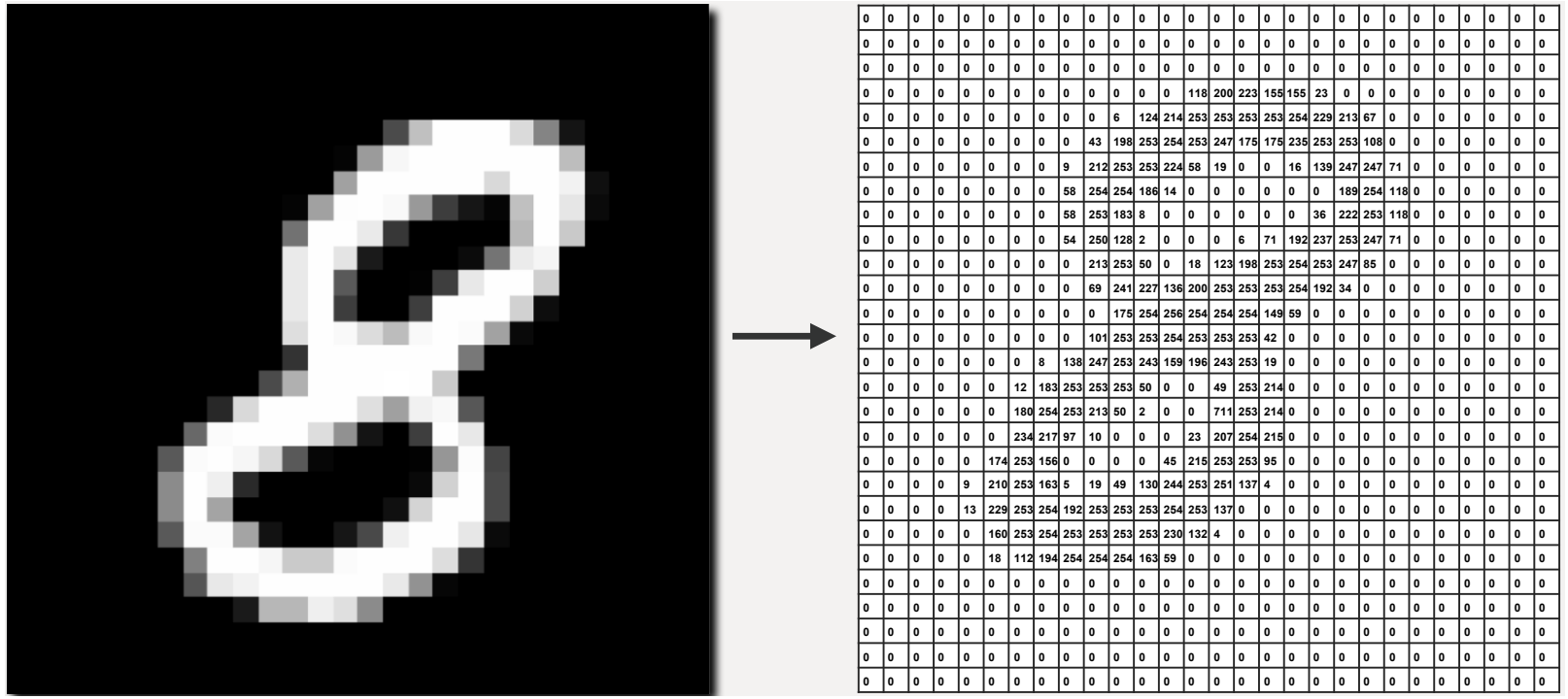
Comment ça marche dans la pratique ?

- Exemple : le dataset MNIST
  - Ensemble de chiffres écrits à la main
- Objectif : identifier le chiffre 1



Mise en route

- Les chiffres sont des images, chaque pixel a une valeur numérique



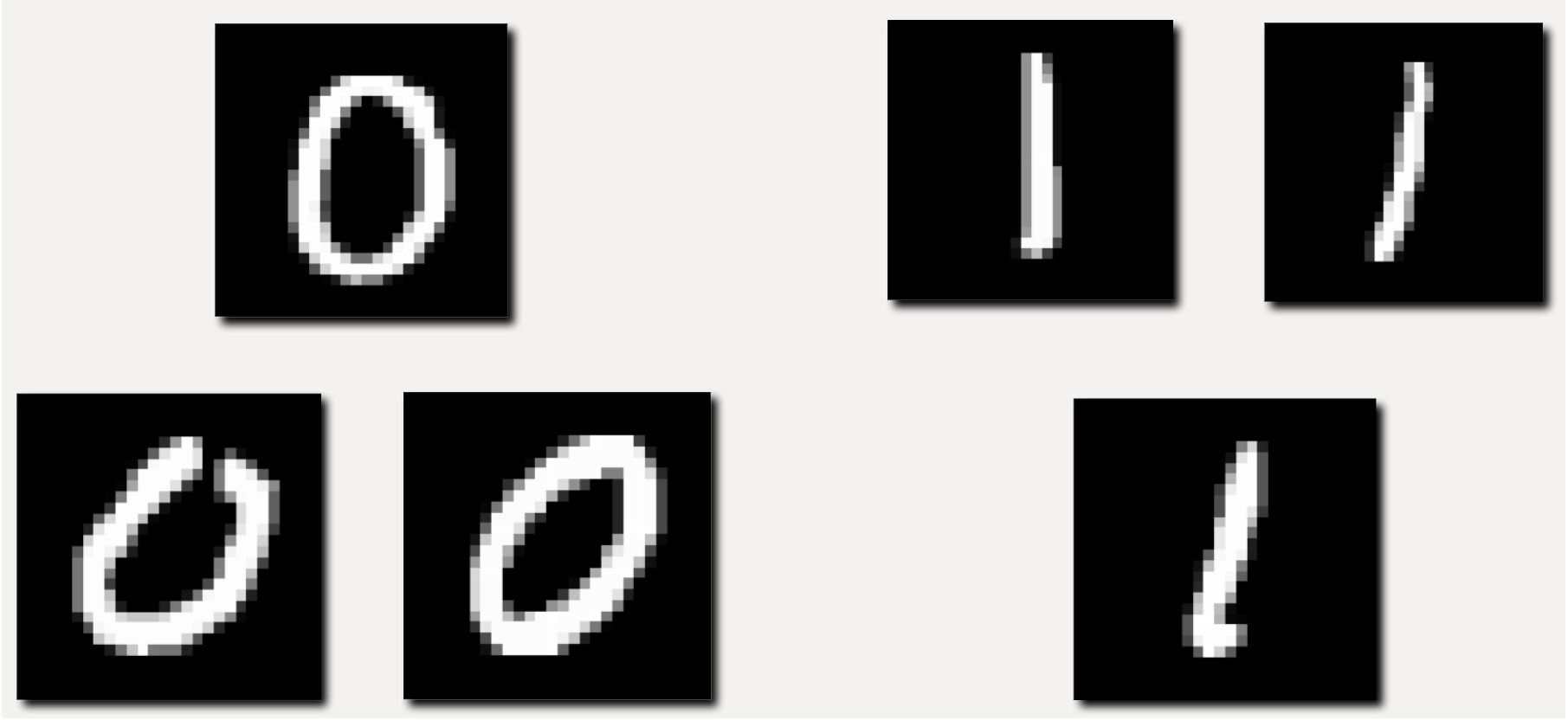
= X

MNIST Dataset of Handwritten Digits (Images)

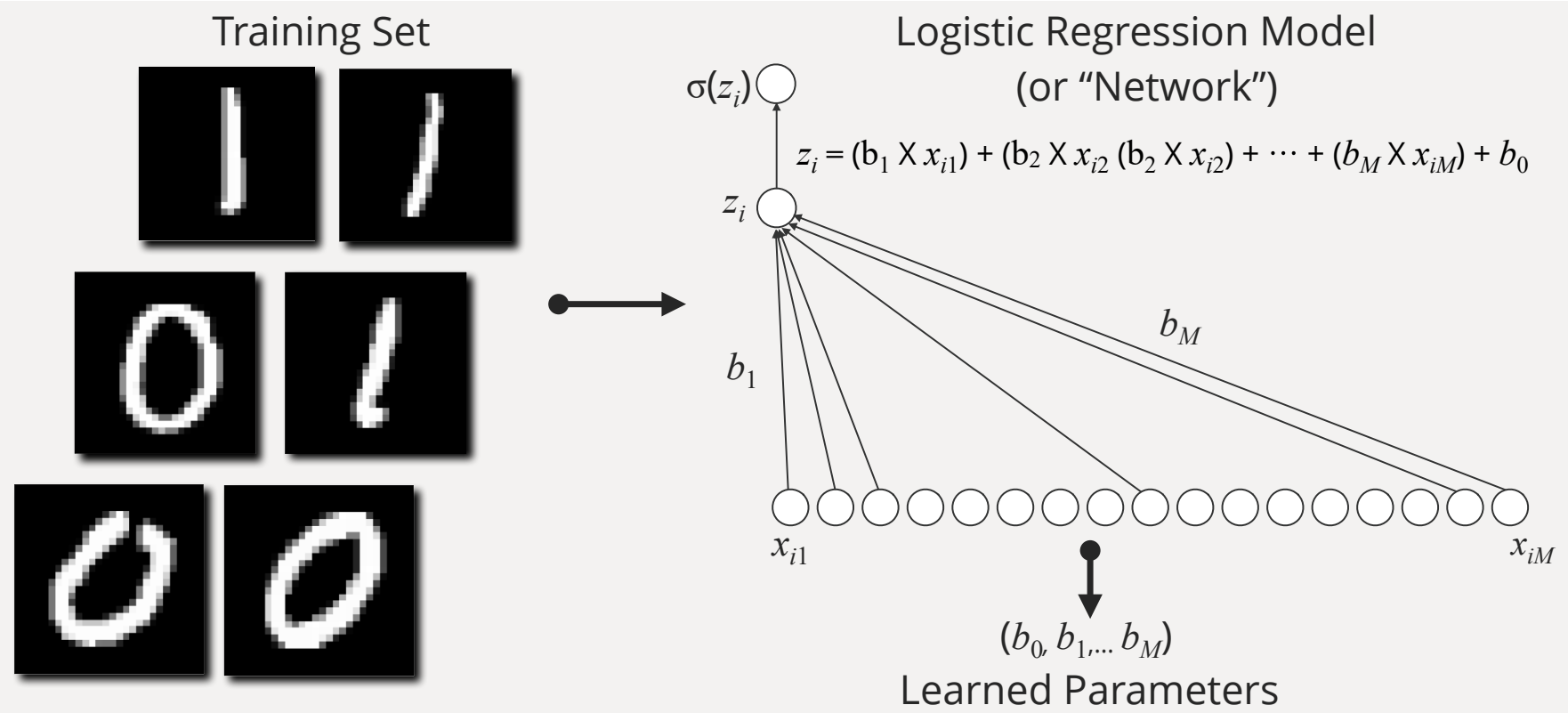
Yann LeCun (Courant Institue, NYU) and Corinna Cortes (Google Labs, New York) CC-by-SA 3.0

http://yann.lecun.com/exdb/mnist/

LE CAS SIMPLE : SORTIE BINAIRE

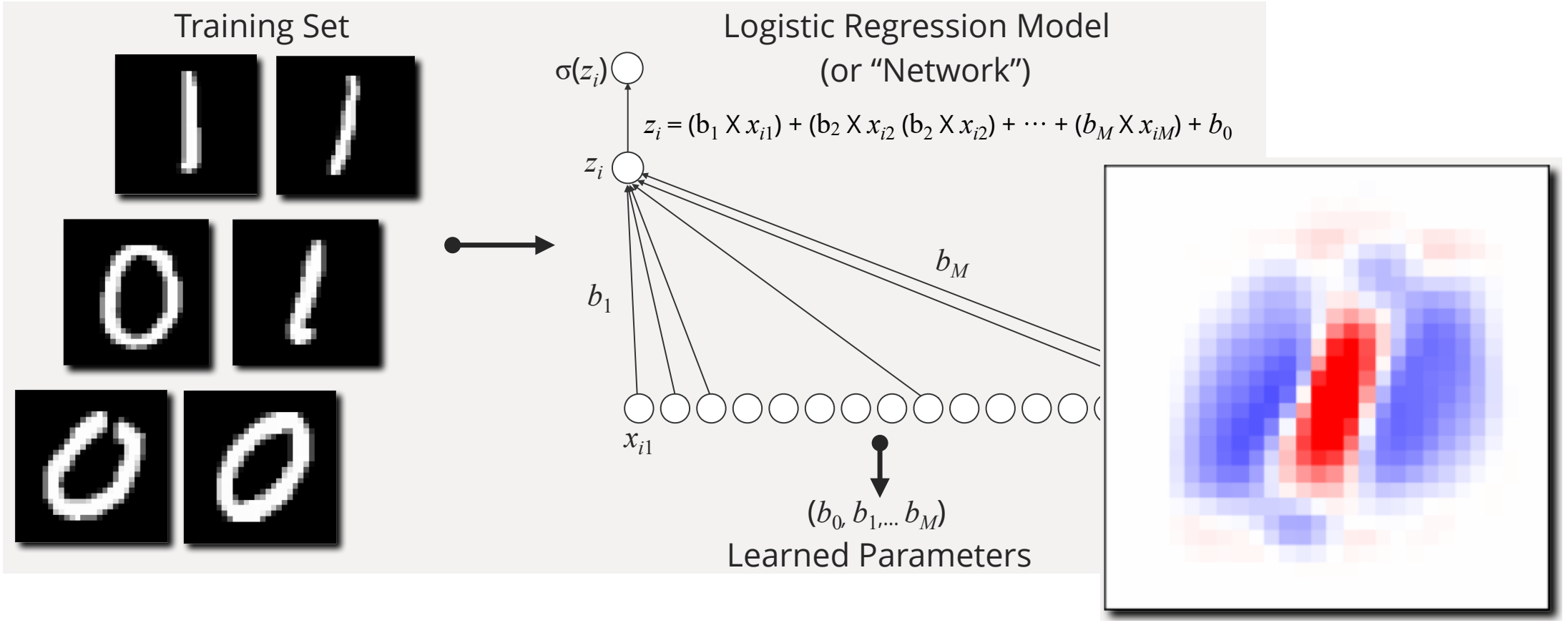


APPRENTISSAGE DU MNIST



Chaque position de la matrice est étiquetée avec une valeur négative (0), positive (1) ou rien. Plus on a des 0 ou 1, plus la "case" est fortement marquée (poids de *b*)

APPRENTISSAGE DU MNIST

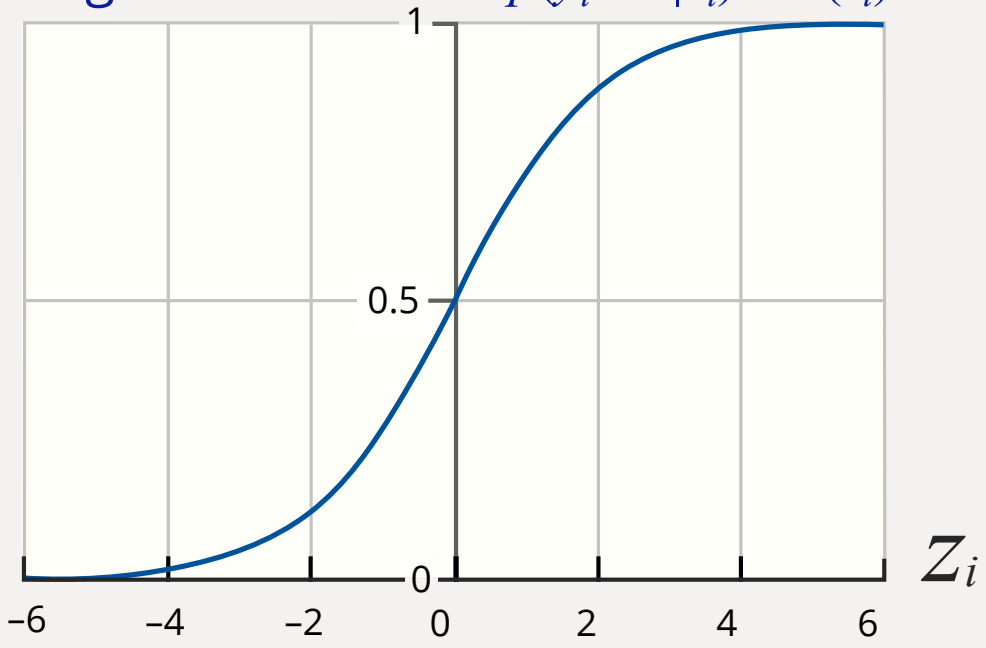


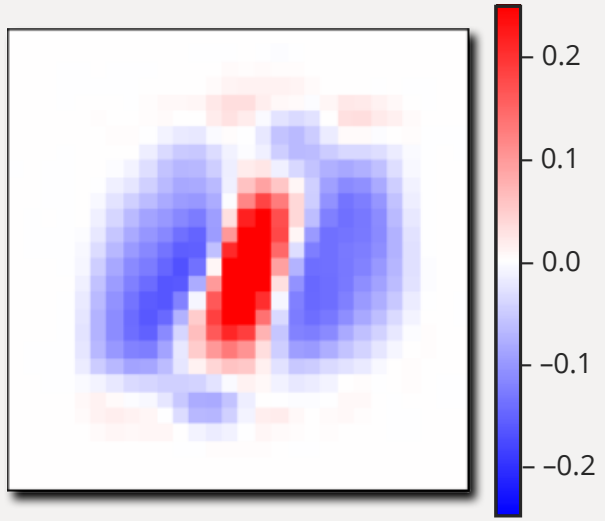
Chaque position de la matrice est étiquetée avec une valeur négative (0), positive (1) ou rien. Plus on a des 0 ou 1, plus la "case" est fortement marquée (poids de *b*)

Fonction Logistique

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) + b_0$$

Sigmoid Function $p(y_i = 1|x_i) = \sigma(z_i)$

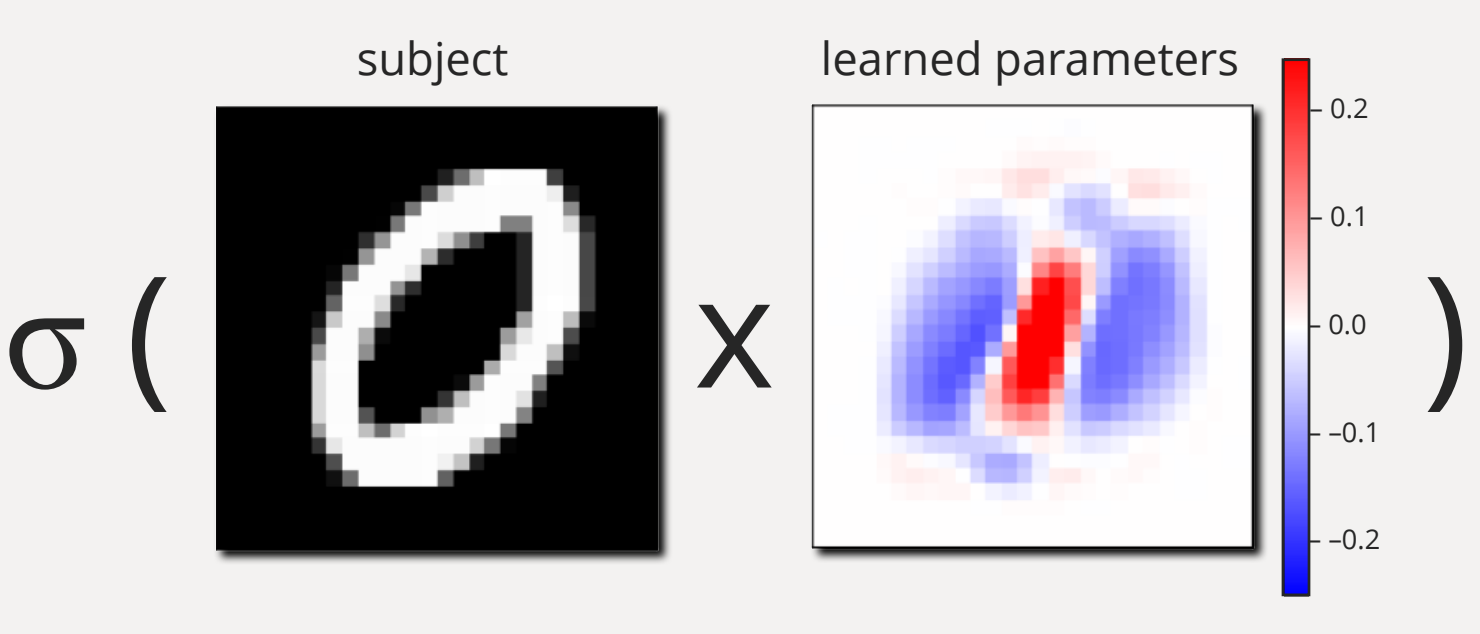




filter b

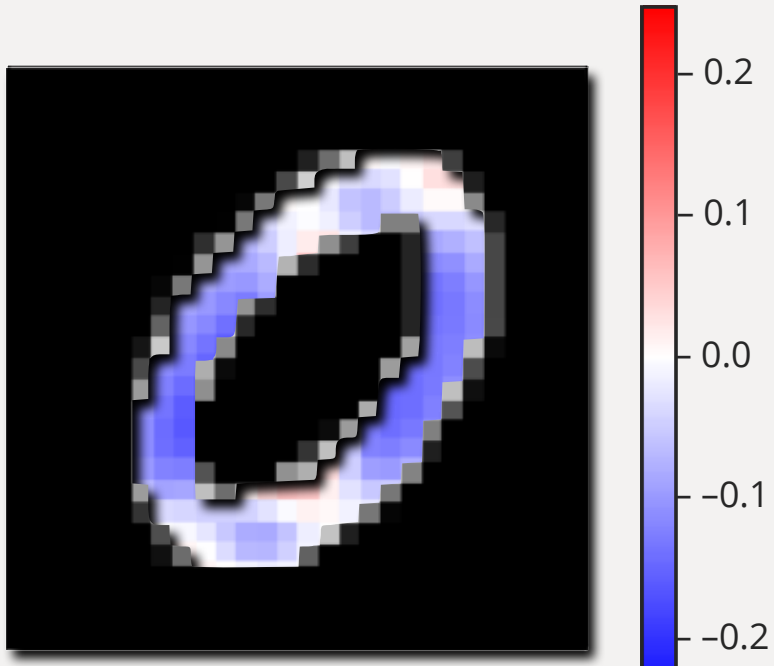
Quelle est la probabilité d'être un Zéro ?

- Pour un chiffre donné, on compare sa matrice avec la matrice entraînée :



Probabilité d'un Zéro

- La "superposition" donne une note pour les parties communes

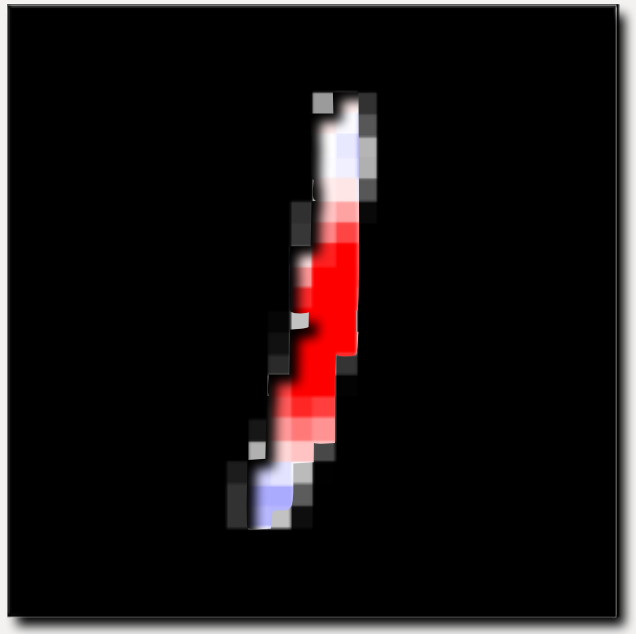


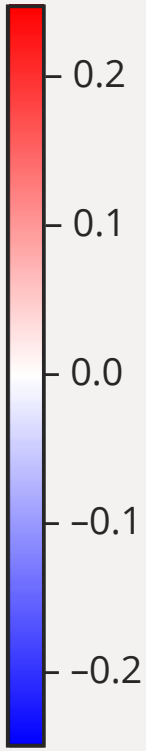
= 0.6% de chance d'être un 1

Probabilité d'un Un

- La "superposition" donne une note pour les parties communes

$$\sigma\left(\right) = 0.991$$



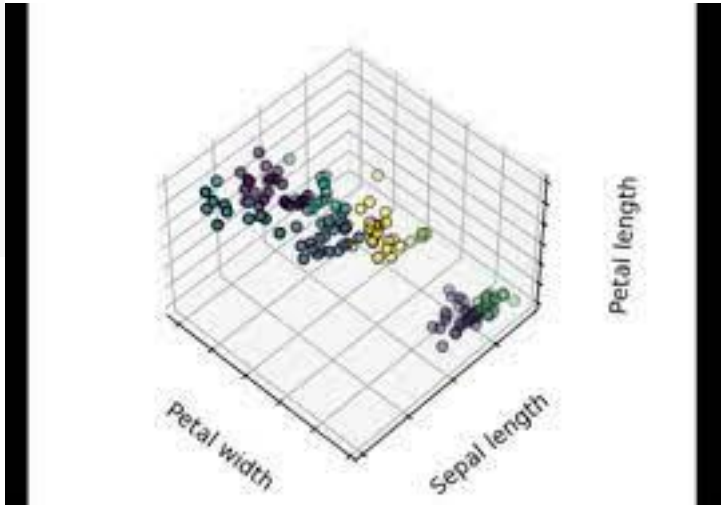


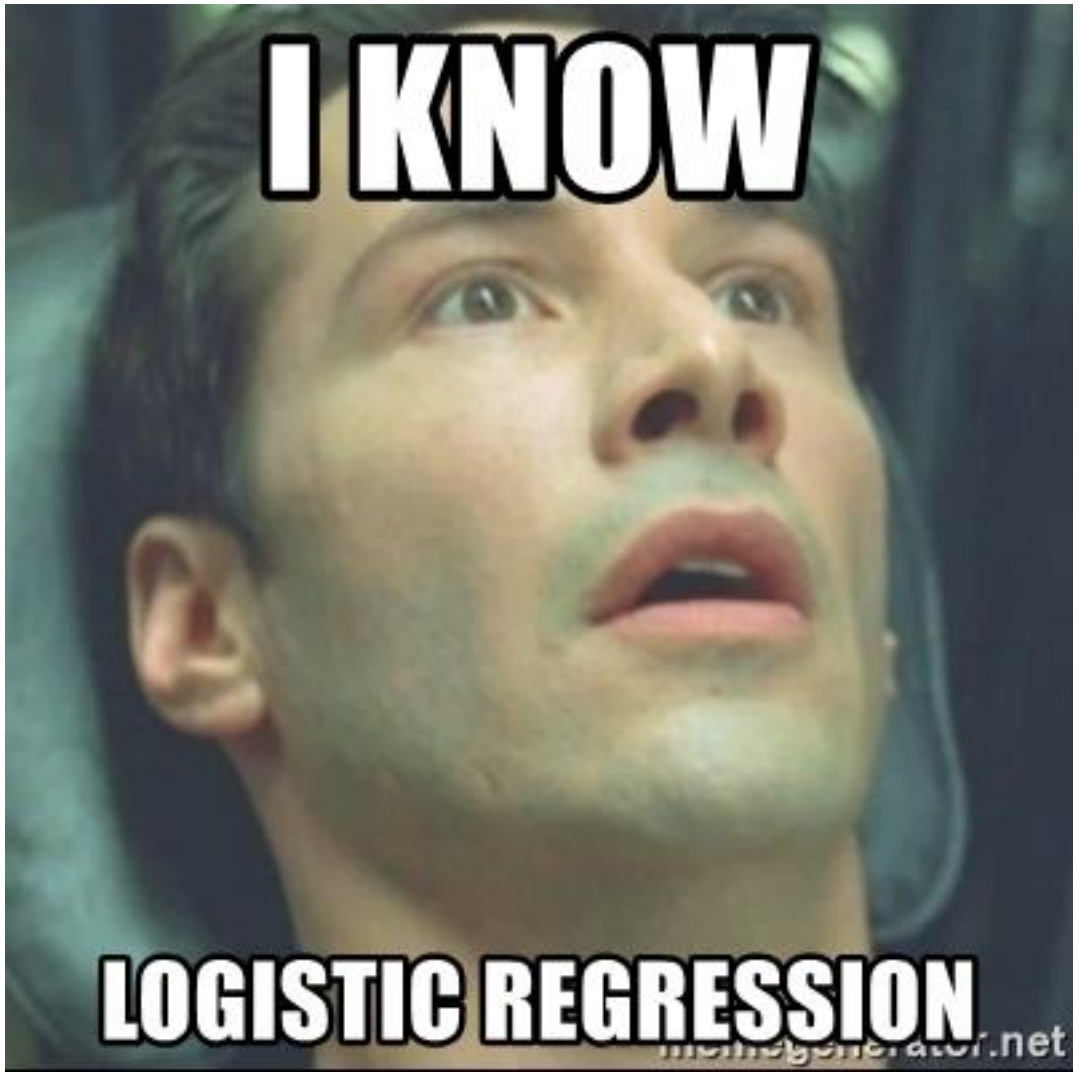
= 99,1% de chance d'être un 1

Regression Logistique dans Sklearn

- L'exemple de l'Iris

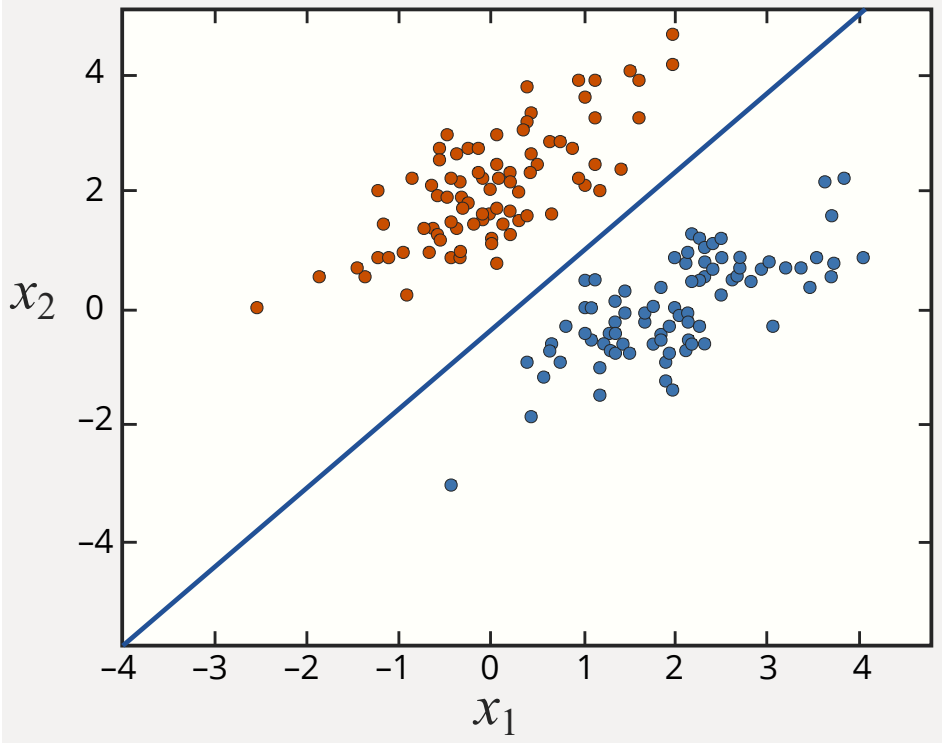
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target

# Create an instance of Logistic Regression Classifier and fit the data.
logreg = LogisticRegression(C=1e5)
logreg.fit(X, Y)
```

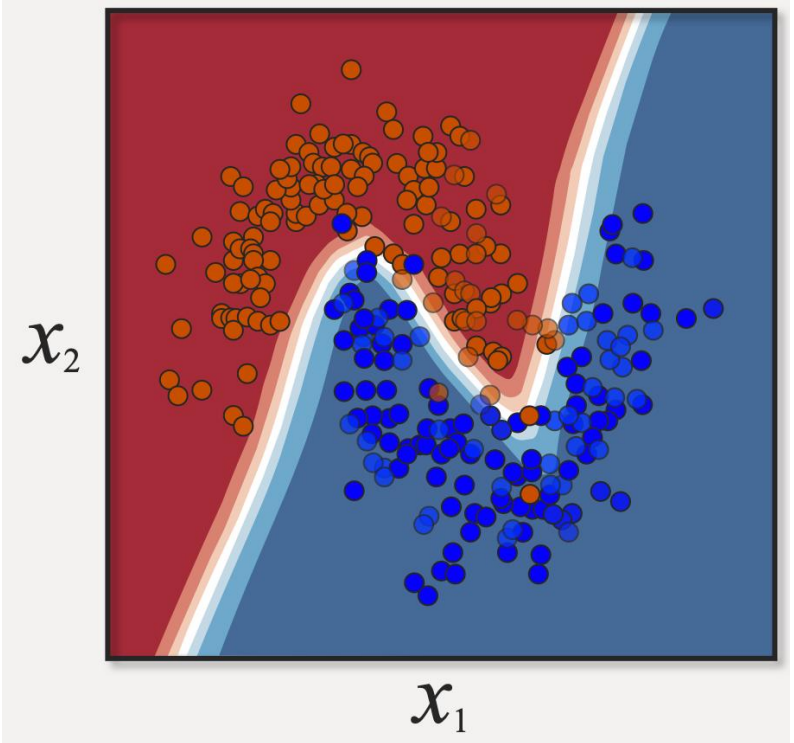



I KNOW
LOGISTIC REGRESSION

Limitations de la Régression Logistique

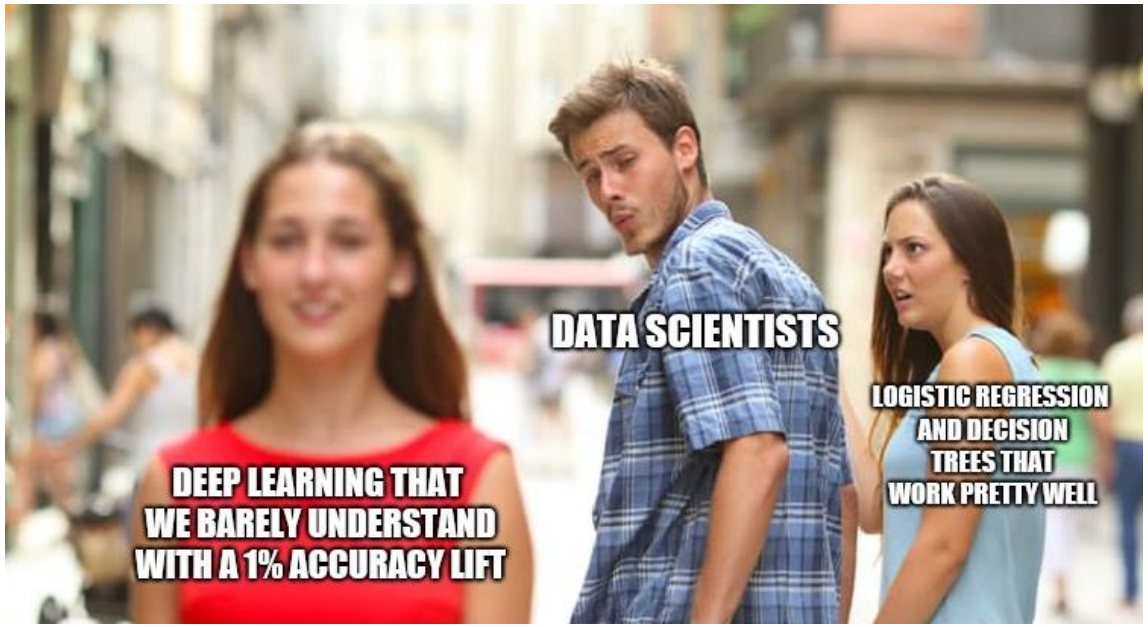
- Les classifieurs linéaires sont limités dans leurs possibilités
- Souvent on a des données non-linéaires





En Résumé...

- On a vu des des algorithmes encore plus avancés pour la **classification**
- Il n'y a pas un algorithme meilleur que l'autre
  - Ça dépend des jeux de données
  - Ça dépend des métriques qu'on veut favoriser
- Dans le prochain cours nous allons regarder plus en détail
  - La régression linéaire
  - Les réseaux de neurones
  - Les séries temporelles



Quelques exercices

- Ouvrir lien suivant et suivre les liens des exercices
  - https://tinyurl.com/yrcpunv8
    - ou
    - https://github.com/lsteffenel/M2Atmo_et_Climat/blob/main/README.md
    - Cela vous amènera dans un environnement Google Colab