RÉGRESSION LOGISTIQUE

Introduction didactique

DÉMARRER PRESENTATION

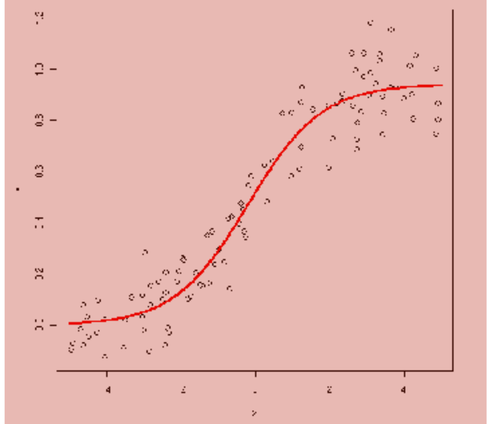
LES MODÈLES DE MACHINE LEARNING

LES ALGORITHMES NE SONT PAS TOUS DESTINÉS AUX MÊMES USAGES. ON LES CLASSE USUELLEMENT SELON PLUSIEURS COMPOSANTES.

supervisé

Régression

données numériques
[x_i] –> [y]



Prévision de marché
Prévision météorologiques
Estimation de durée de vie
Prédiction de popularité (pubs)

Classification

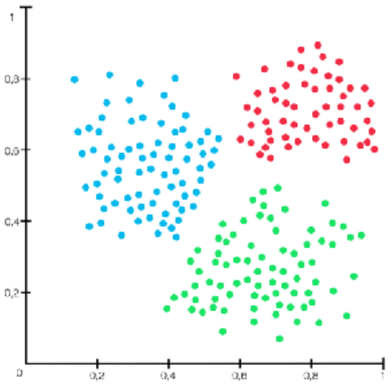
données labellisées
[pixels] –> [label]

Classification d'images
Fidélisation de clients
Détection de fraudes
Diagnostiques

non supervisé

Clustering

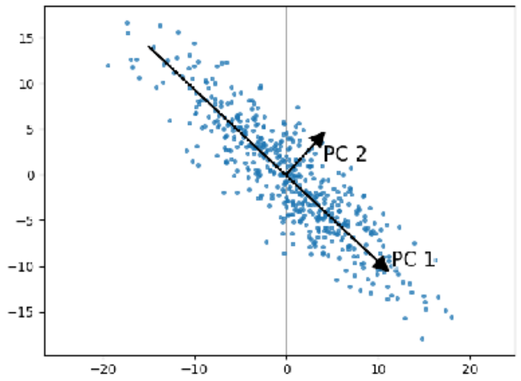
données non labellisées
[x, y, z, …]



Système de recommandation
Marketing ciblé
Segmentation de clientèle

Réduction dimension

données non labellisées
[x, y, z, …]

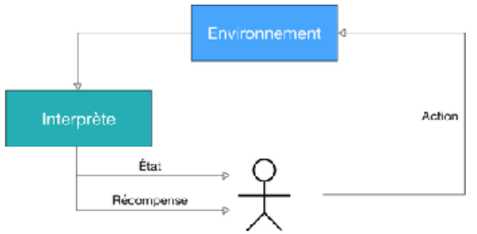


Visualisation de Big Data
Compression des données
Découverte de structures

renforcement

Ajustement - Action

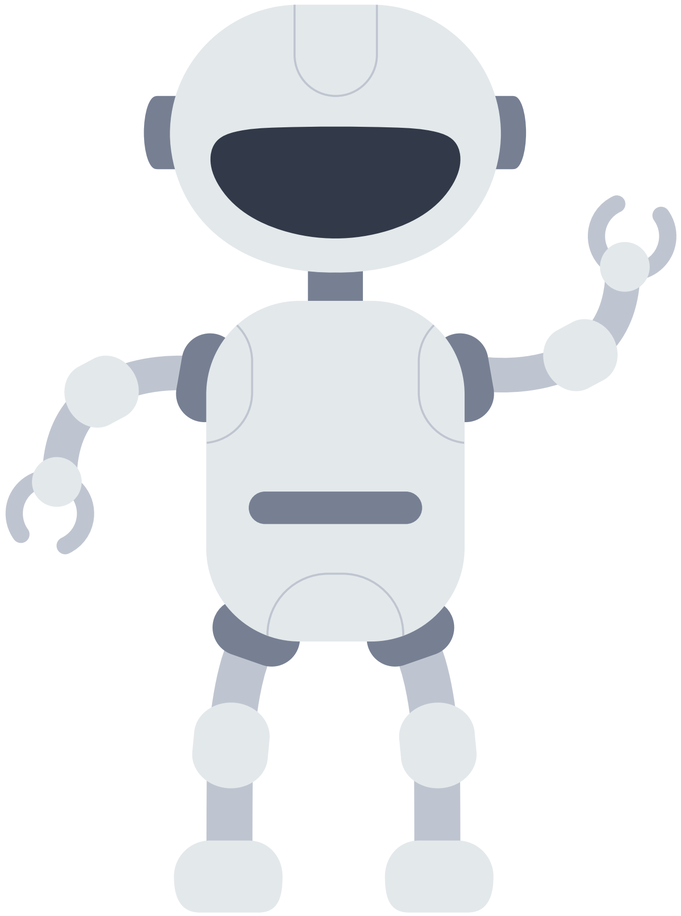
toutes données
[env.] –> [action]



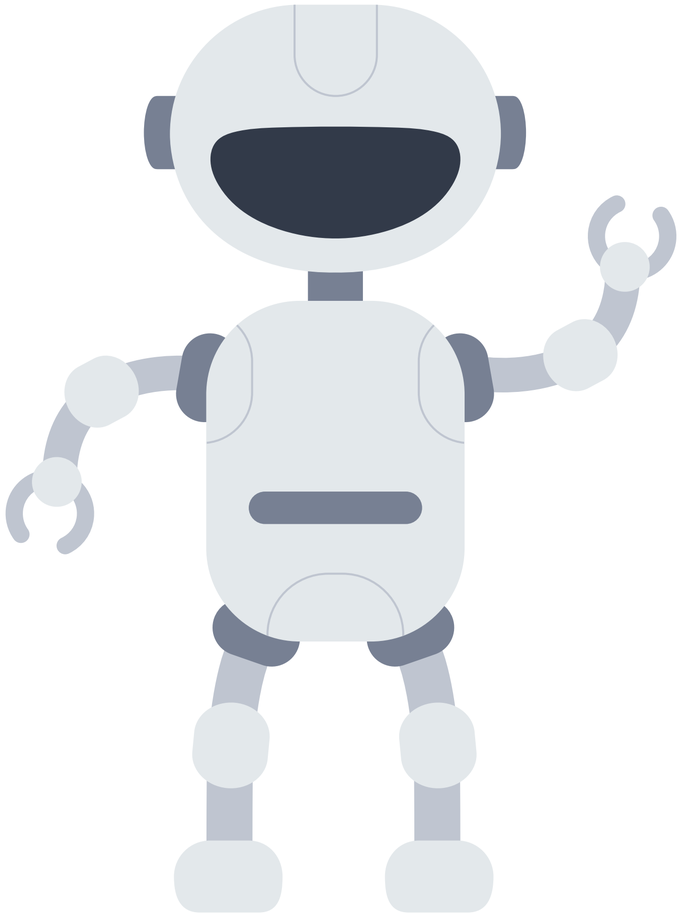
Décisions en temps réel
Navigation robotique
Apprentissage de tâches
Acquisition de compétences

Inspiré de : Les usages de l'intelligence artificielle 2018

SOMMAIRE



SOMMAIRE

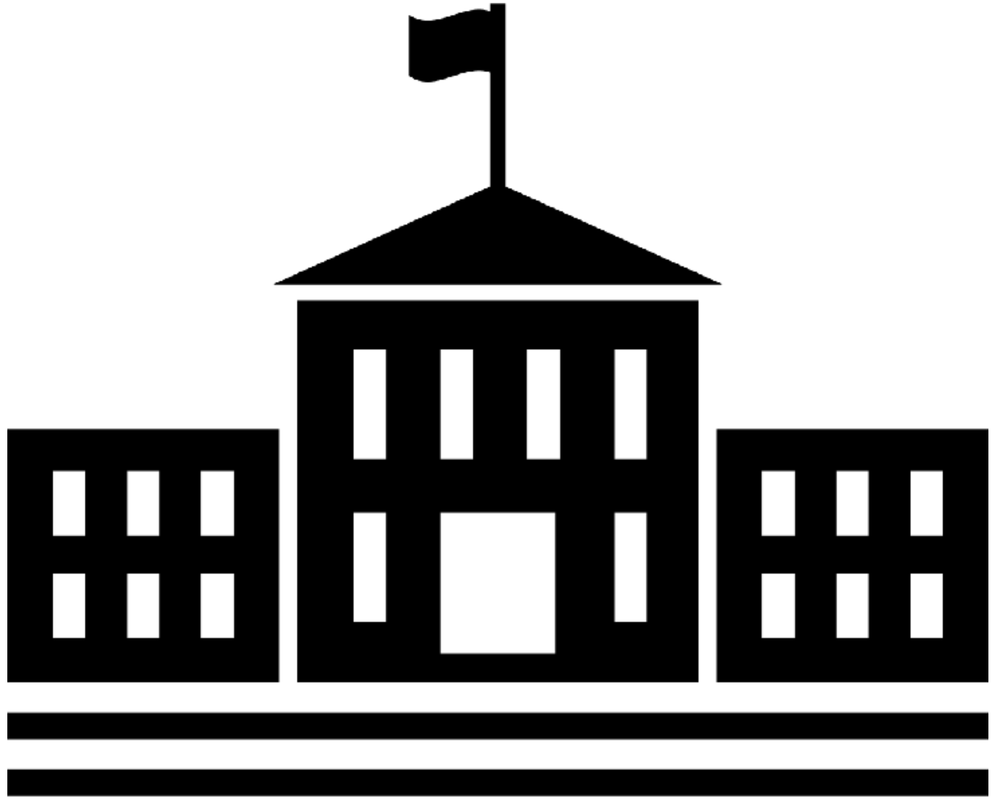


ACCEPTATION À L'UNIVERSITÉ

EN FONCTION DES TESTS D'ENTRÉE ET DES NOTES DU LYCÉE

Description

- On veut prédire si un **étudiant** va être **accepté à l'université** en fonction des **deux critères** suivants :
  - Les notes obtenues aux **contrôles d'entrée**.
  - Les notes obtenues au **lycée**.



Notes contrôle

Notes lycée

ACCEPTATION À L'UNIVERSITÉ

EN FONCTION DES TESTS D'ENTRÉE ET DES NOTES DU LYCÉE

Notes contrôle

Notes lycée

Étudiant 1

- Notes contrôle : 9/10
- Notes lycée : 8/10

Étudiant 2

- Notes contrôle : 3/10
- Notes lycée : 4/10

Étudiant 3

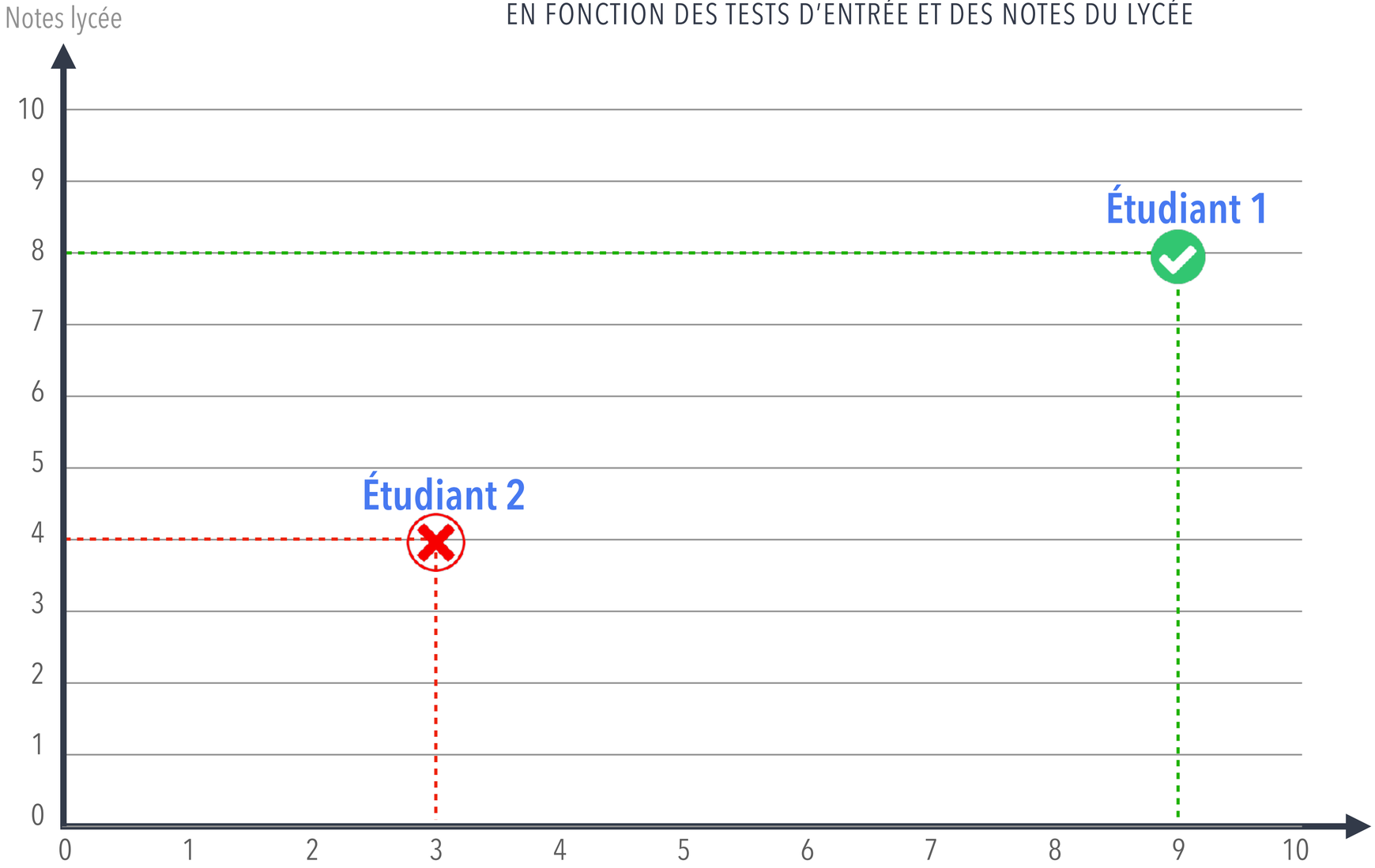
- Notes contrôle : 7/10
- Notes lycée : 6/10

Prenons trois étudiants.

Nous connaissons les **résultats des deux premiers** et non voulons déterminer si le **troisième** a des chances d'être accepté ou pas

ACCEPTATION À L'UNIVERSITÉ

EN FONCTION DES TESTS D'ENTRÉE ET DES NOTES DU LYCÉE



Étudiant 3

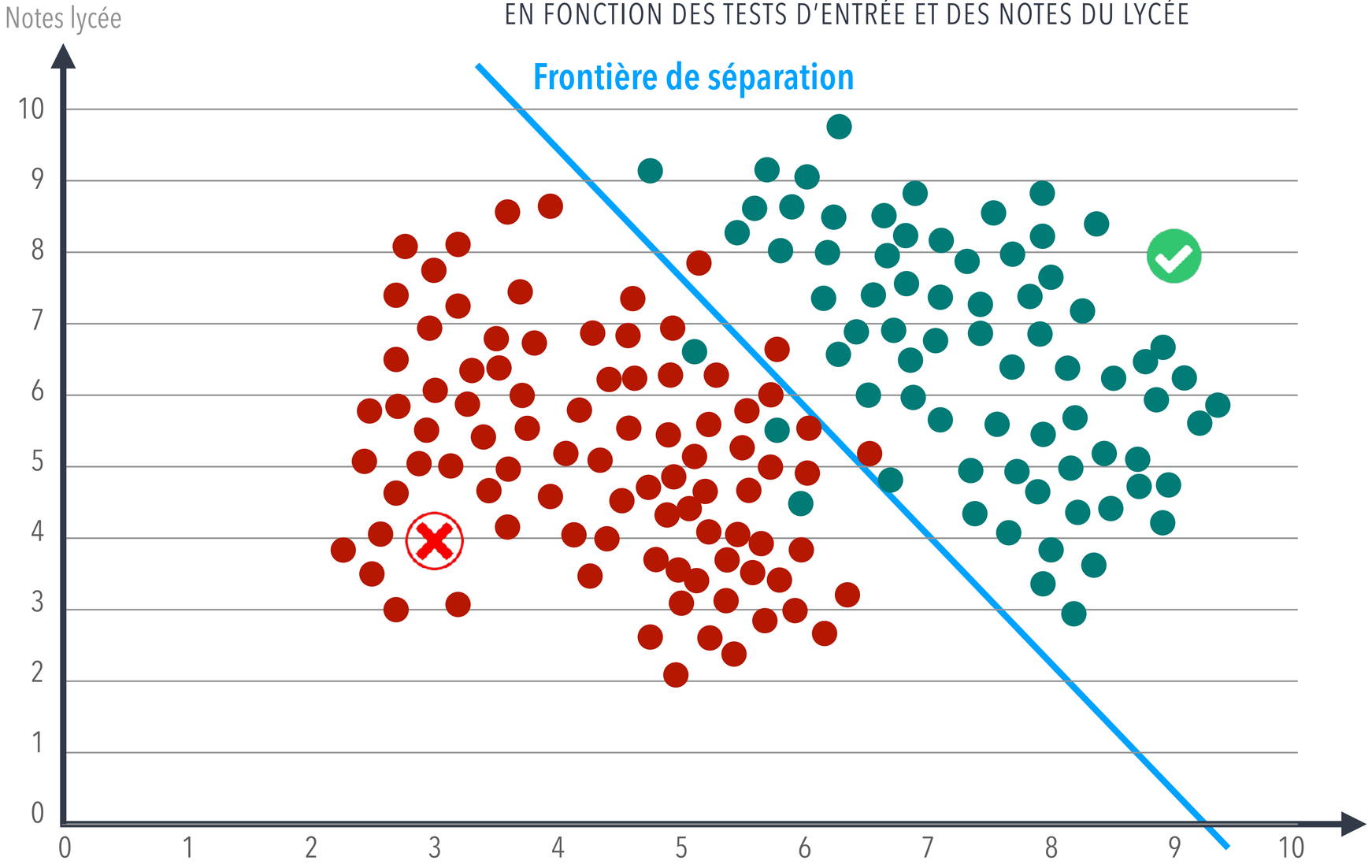
- Notes contrôle : 7/10
- Notes lycée : 6/10

?

On reporte dans un plan les deux premiers étudiants en fonction de leurs notes au contrôle et au lycée

ACCEPTATION À L'UNIVERSITÉ

EN FONCTION DES TESTS D'ENTRÉE ET DES NOTES DU LYCÉE



Étudiant 3

- Notes contrôle : 7/10
- Notes lycée : 6/10

?

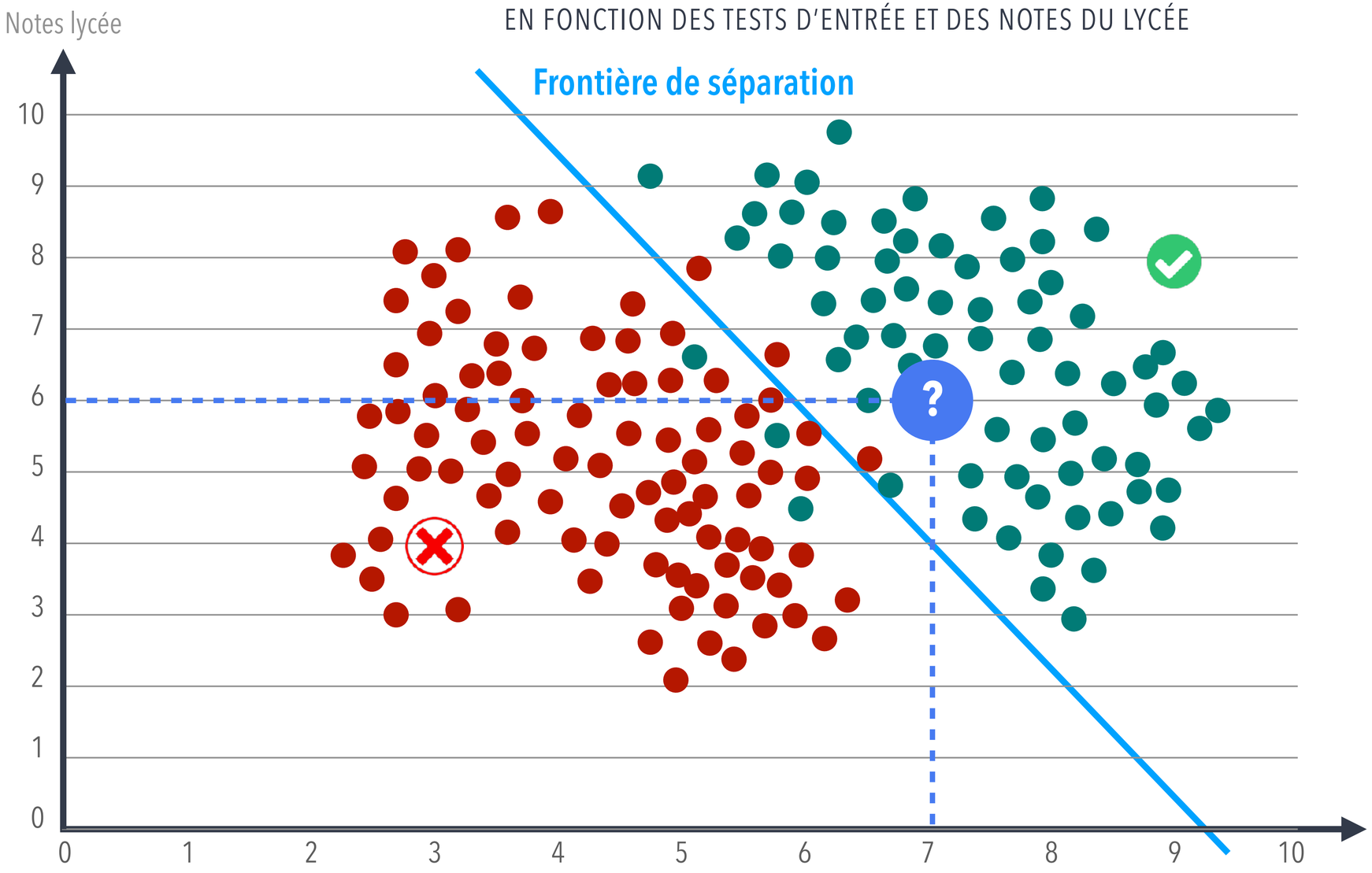
Question

- L'étudiant est-il accepté ?
  - Oui
  - Non

Nous représentons plusieurs dizaines d'étudiants dont le résultat d'admission est connu. Ceux de couleur rouge ne sont pas admis et ceux en vert sont admis
Nous indiquons la frontière de séparation qui sera être obtenue par apprentissage comme nous allons l'expliquer.

ACCEPTATION À L'UNIVERSITÉ

EN FONCTION DES TESTS D'ENTRÉE ET DES NOTES DU LYCÉE



Étudiant 3

- Notes contrôle : 7/10
- Notes lycée : 6/10

Question

- L'étudiant est-il accepté ?
  - Oui
  - Non

En reportant les notes de l'étudiant 3, on constate qu'il a de forte chances d'être accepté

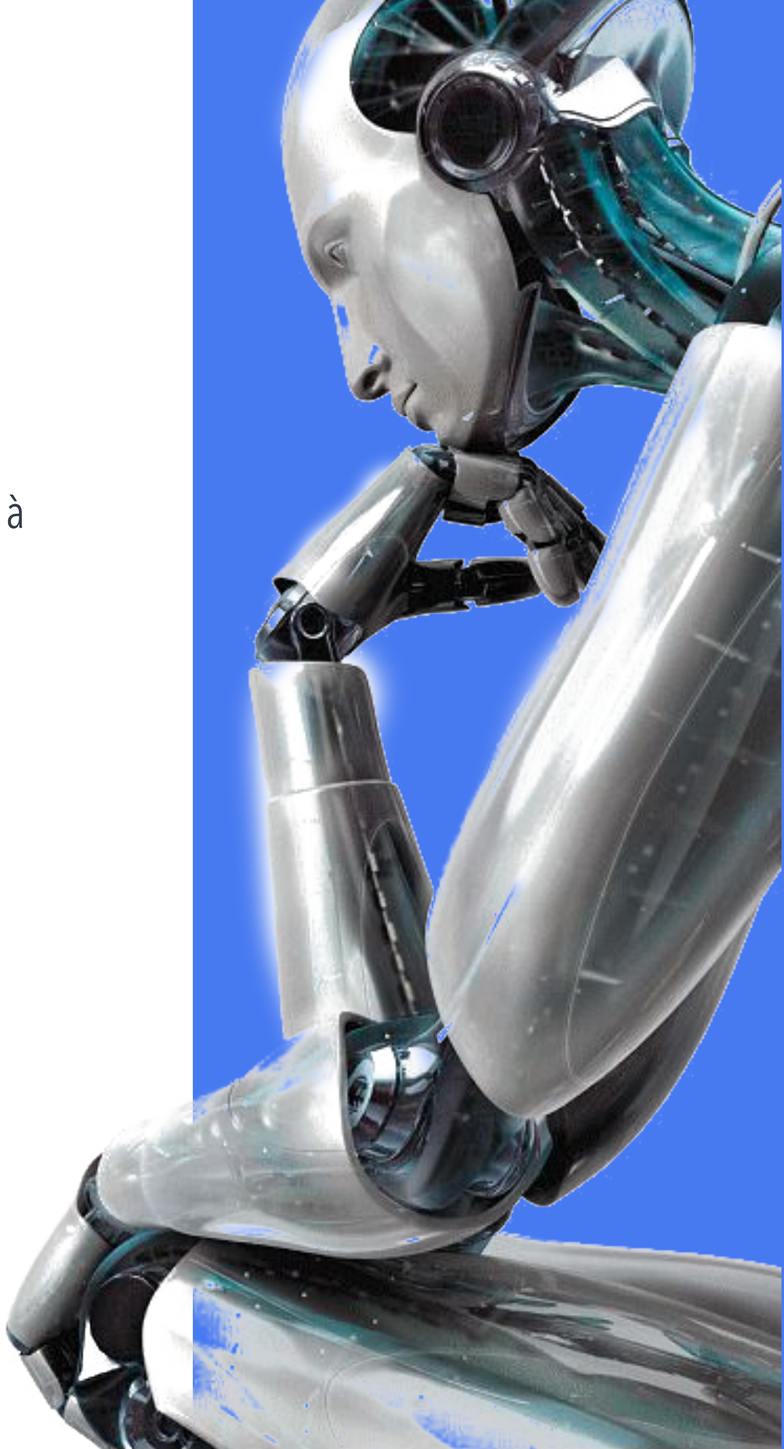
RÉGRESSION LOGISTIQUE

LA MÉTHODE DE CLASSIFICATION QUE NOUS AVONS DÉCRITE REPOSE EST UNE RÉGRESSION LOGISTIQUE

La question à résoudre

- Comme pour la régression linéaire pour laquelle on devait déterminer la droite optimale, la question à laquelle on doit s'atteler maintenant est la suivante :

Comment obtient-on la **frontière optimale** de séparation des données ?



RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES

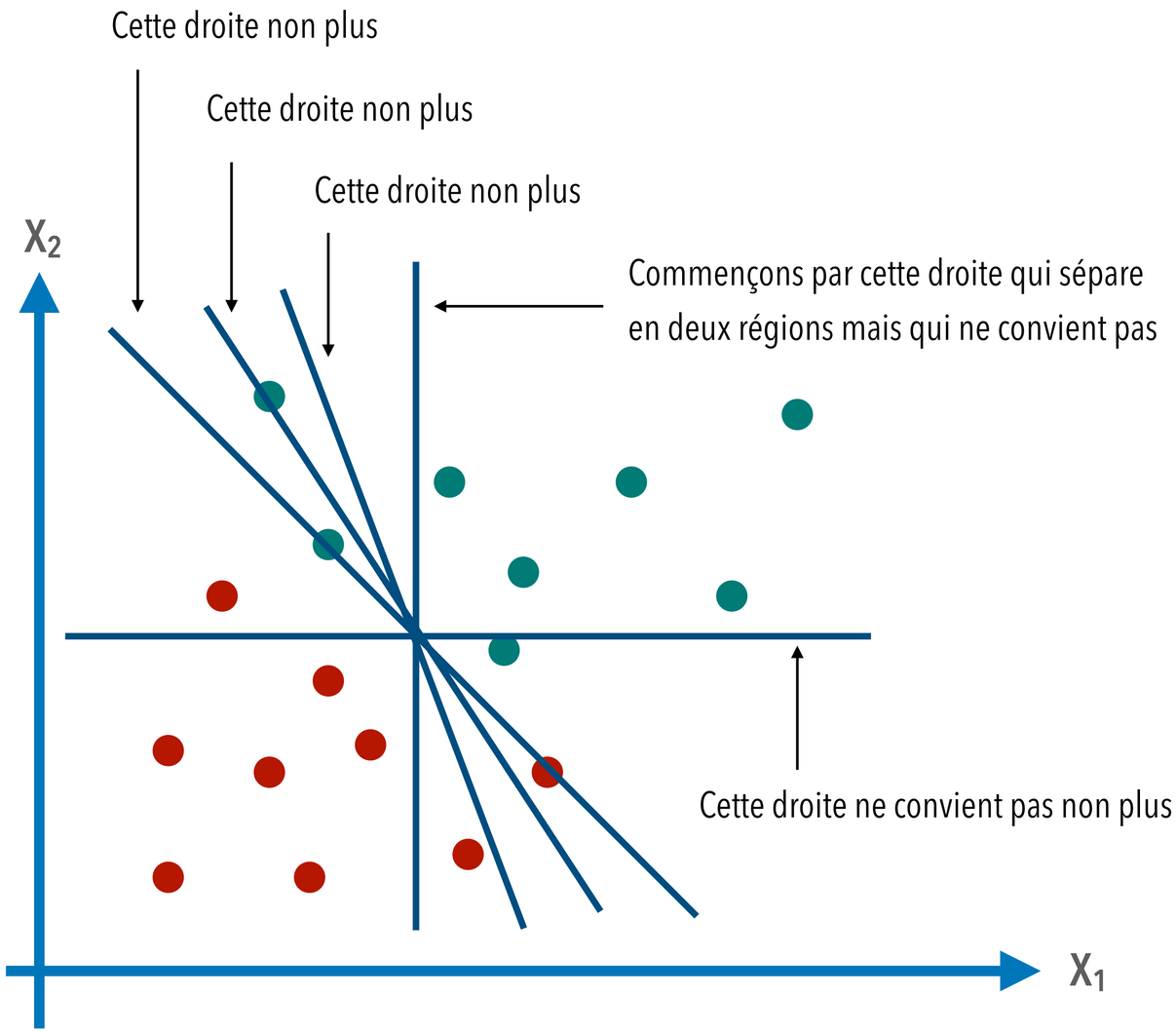
La question est :

Quelle est la droite qui sépare les deux régions de points ?

Une droite de séparation choisie, la procédure consiste à :

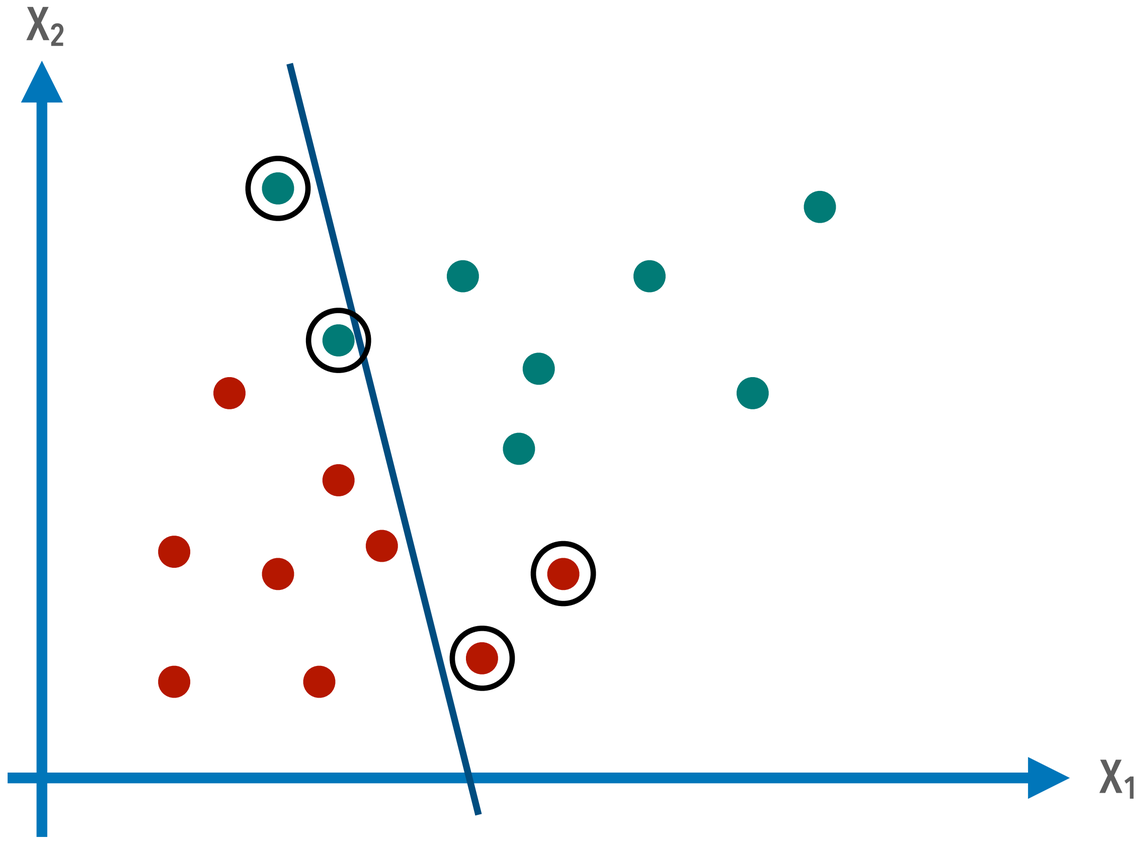
- calculer le nombre de points qui ne sont pas dans la bonne zone et,
- à diminuer ce nombre d'erreurs à chaque nouvelle droite de séparation.

Examinons ensemble cette procédure.



RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES



Prenons la droite aléatoire suivante.

Combien avons-nous d'erreurs ?

Erreurs : 4

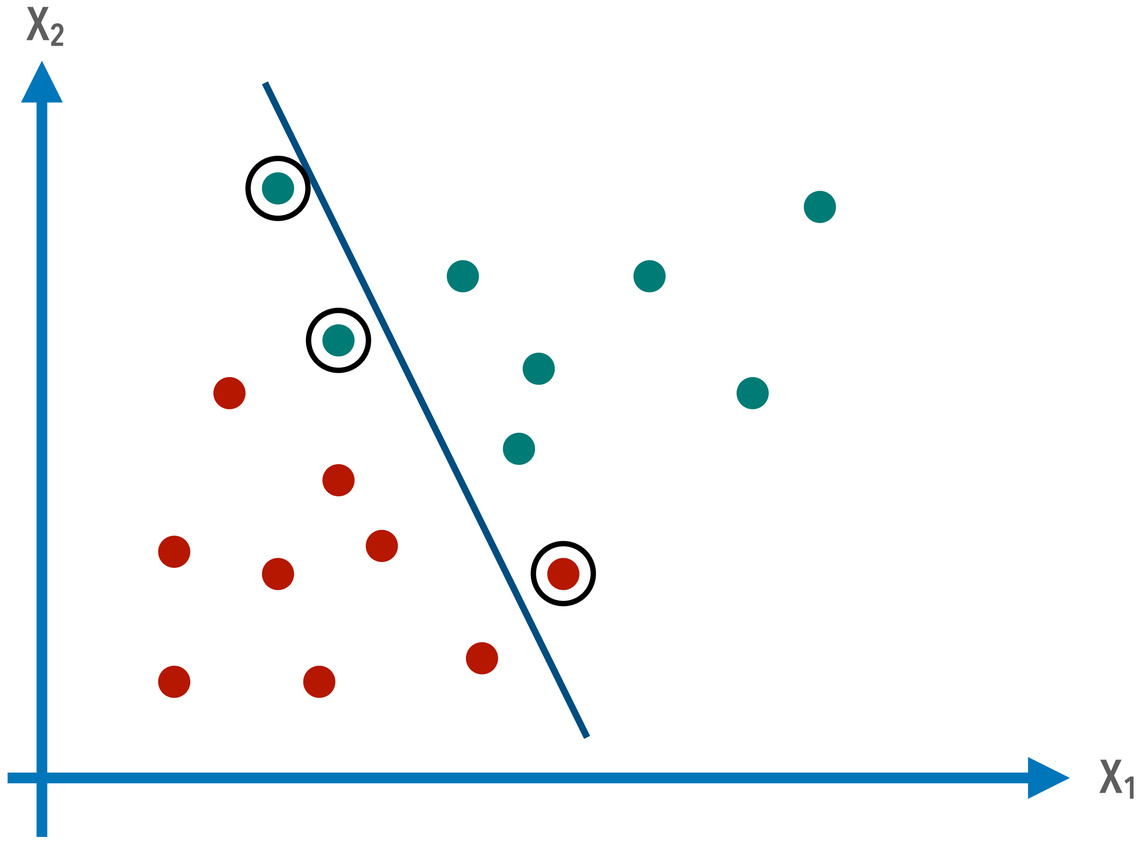
Descente de gradient*

Comme pour la régression linéaire, pour diminuer le nombre d'erreurs à chaque tentative, on va utiliser la descente de gradient

* En réalité, on utilise la méthode « Log-Loss Function » qui ne calcule pas le nombre d'erreurs, mais un taux d'erreur en associant des valeurs élevées aux points mal placés et des valeurs faibles aux points bien positionnés.

RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES



La descente de gradient donne la droite suivante.

Combien avons-nous d'erreurs ?

Erreurs : 3

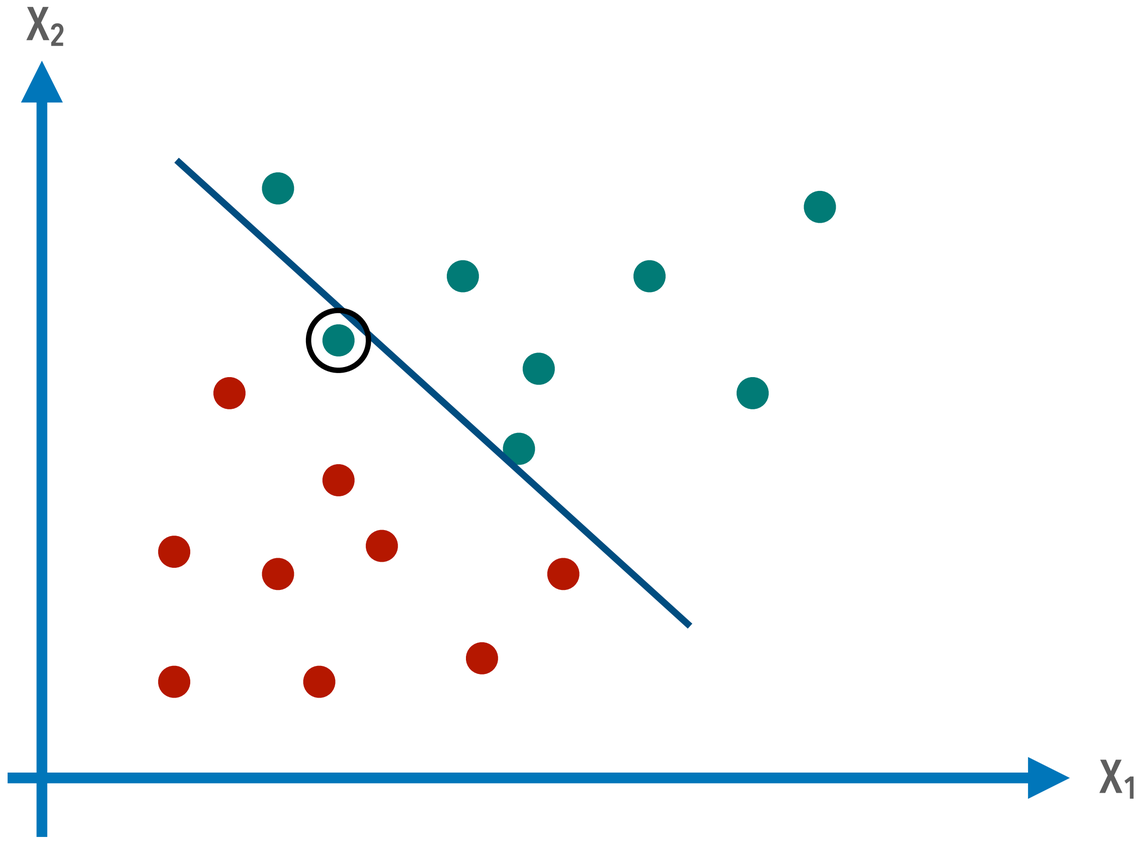
Descente de gradient*

Comme pour la régression linéaire, pour diminuer le nombre d'erreurs à chaque tentative, on va utiliser la descente de gradient

* En réalité, on utilise la méthode « Log-Loss Function » qui ne calcule pas le nombre d'erreurs, mais un taux d'erreur en associant des valeurs élevées aux points mal placés et des valeurs faibles aux points bien positionnés.

RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES



La descente de gradient donne la droite suivante.

Combien avons-nous d'erreurs ?

Erreurs : 1

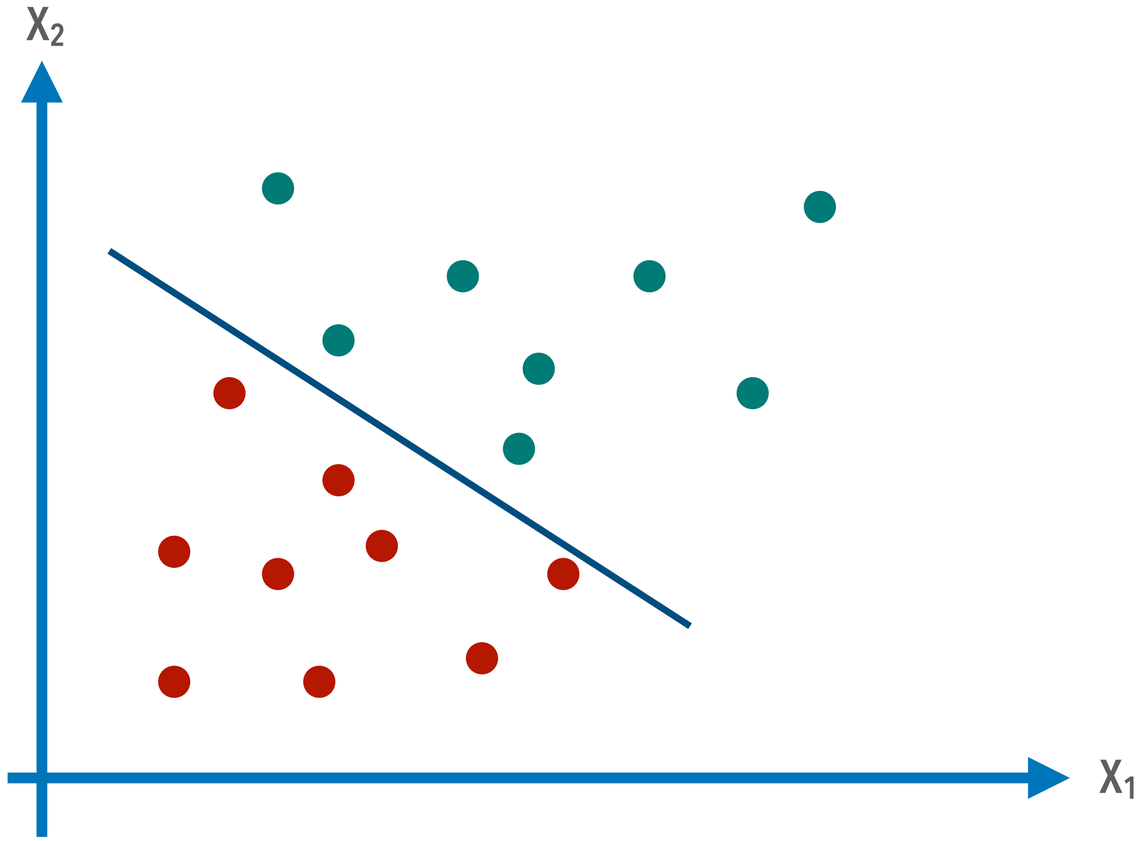
Descente de gradient*

Comme pour la régression linéaire, pour diminuer le nombre d'erreurs à chaque tentative, on va utiliser la descente de gradient

* En réalité, on utilise la méthode « Log-Loss Function » qui ne calcule pas le nombre d'erreurs, mais un taux d'erreur en associant des valeurs élevées aux points mal placés et des valeurs faibles aux points bien positionnés.

RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES



La descente de gradient donne la droite suivante.

Combien avons-nous d'erreurs ?

Erreurs : 0

Descente de gradient*

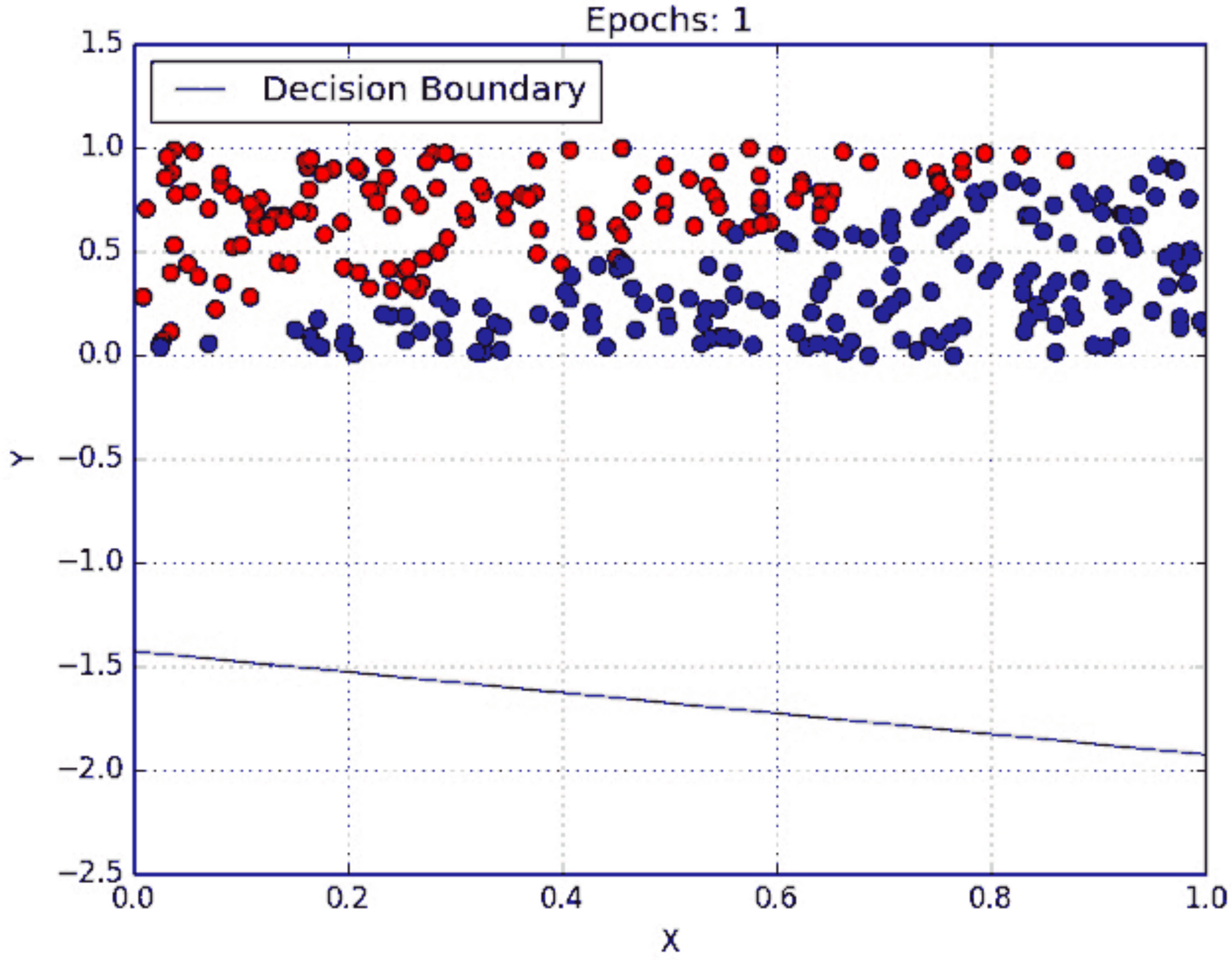
Comme pour la régression linéaire, pour diminuer le nombre d'erreurs à chaque tentative, on va utiliser la descente de gradient

Il n'y a plus d'erreurs, la droite de séparation en deux régions a été trouvée. Le processus est interrompu.

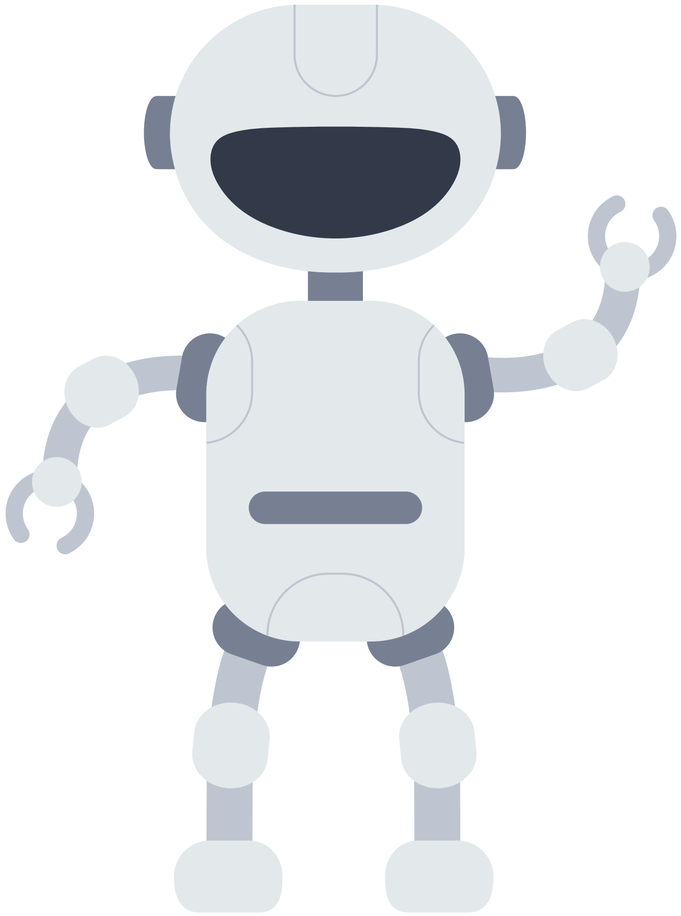
* En réalité, on utilise la méthode « Log-Loss Function » qui ne calcule pas le nombre d'erreurs, mais un taux d'erreur en associant des valeurs élevées aux points mal placés et des valeurs faibles aux points bien positionnés.

RÉGRESSION LOGISTIQUE

LA FRONTIÈRE OPTIMALE DE SÉPARATION DES DONNÉES



SOMMAIRE



INTRODUCTION

Classification binaire

- La régression logistique est un grand **classique de la classification**. C'est le modèle certainement le plus **répandu** dans le monde des **banques** et des **assureurs**.

- Le modèle offre un bon compromis entre **performance** et pouvoir **explicatif**.

- Par **souci didactique**, nous décrivons les principes de la régression logistique sur un problème de classification binaire ; sorties sont **{0,1}**.

- Nous **généraliserons** ensuite à des problèmes de classification **multiclasses**, c'est-à-dire lorsque les valeurs de sortie ne sont plus binaires.

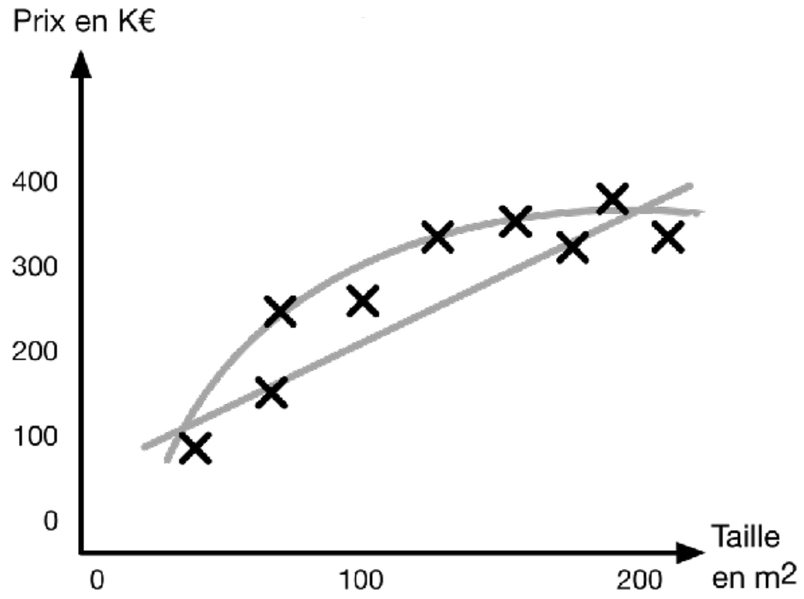
Source : Data science : fondamentaux et études de cas: Machine Learning avec Python et R

RETOUR SUR LA RÉGRESSION LINÉAIRE

Modèle de classification

- Désormais, on passe d'un modèle de régression à un modèle de classification.

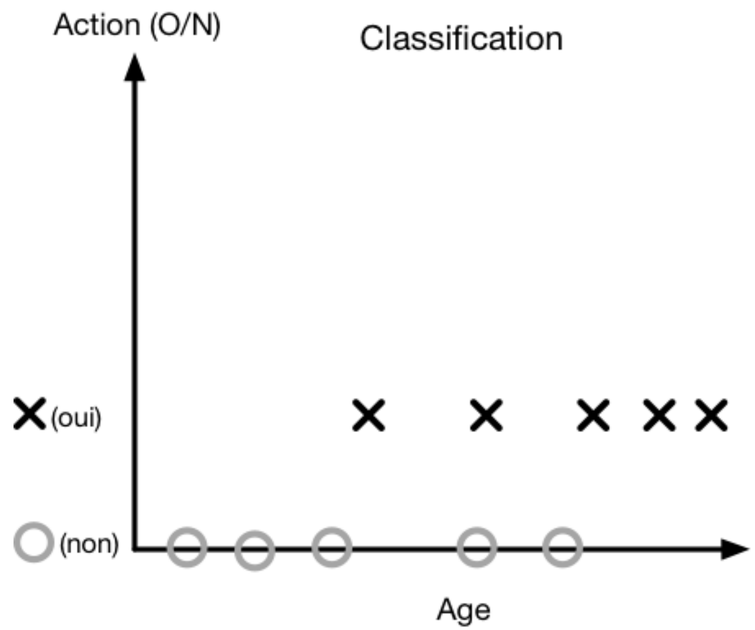
Algorithmes de régression



Il y a une certaine corrélation entre la superficie et le prix du logement.

C'EST UN PROBLÈME DE RÉGRESSION

Algorithmes de classification



Il y a une certaine corrélation entre l'age et l'action d'achat[1].

C'EST UN PROBLÈME DE CLASSIFICATION

1. Clic sur l'offre d'achat dans un mail ou sur une publicité dans un réseau social par exemple.

RETOUR SUR LA RÉGRESSION LINÉAIRE

Régressions linéaire univariées

La recherche du **meilleur modèle** de régression linéaire (univarié ou pas) ou polynomiale repose sur **trois étapes** essentielles :

01 *La définition d'une **fonction hypothèse***

02 *La construction d'une **fonction de coût***

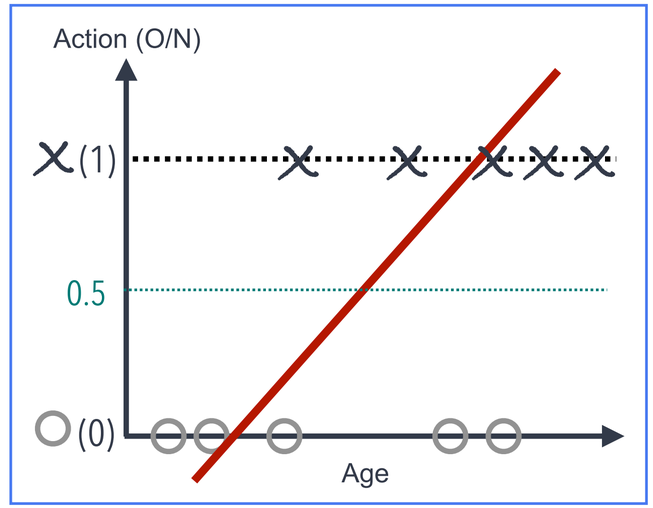
03 *La **minimisation** de la fonction de coût*

LA FONCTION HYPOTHÈSE

Régression Logistique

- En théorie on pourrait reprendre la même **fonction hypothèse** que pour les régressions linéaires. Il suffit d'écrire :

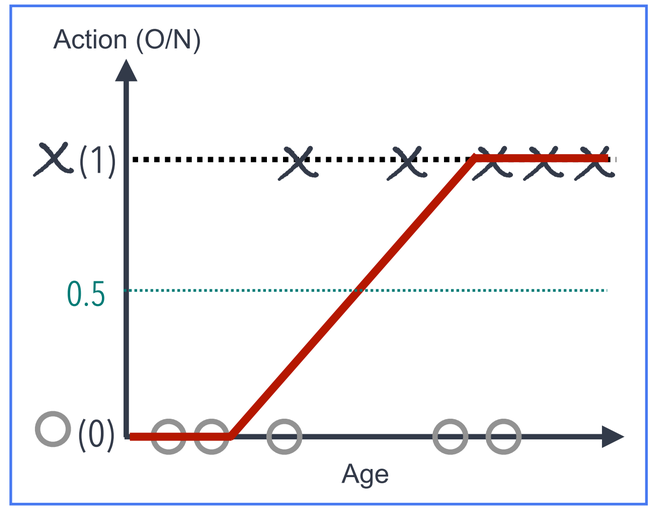
$$logistic\_regression(x) = \begin{cases} 0, & h(x) < 0.5 \\ 1, & h(x) \geq 0.5 \end{cases}$$



- Mais ce traitement est loin d'être **optimal pour la classification**. En particulier, on préférerait que h représente une **probabilité** alors que, dans le cas de la régression linéaire, h représente directement des valeurs continues.
- La fonction h pour la classification doit respecter la condition suivante :

$$0 \leq h(x) \leq 1$$

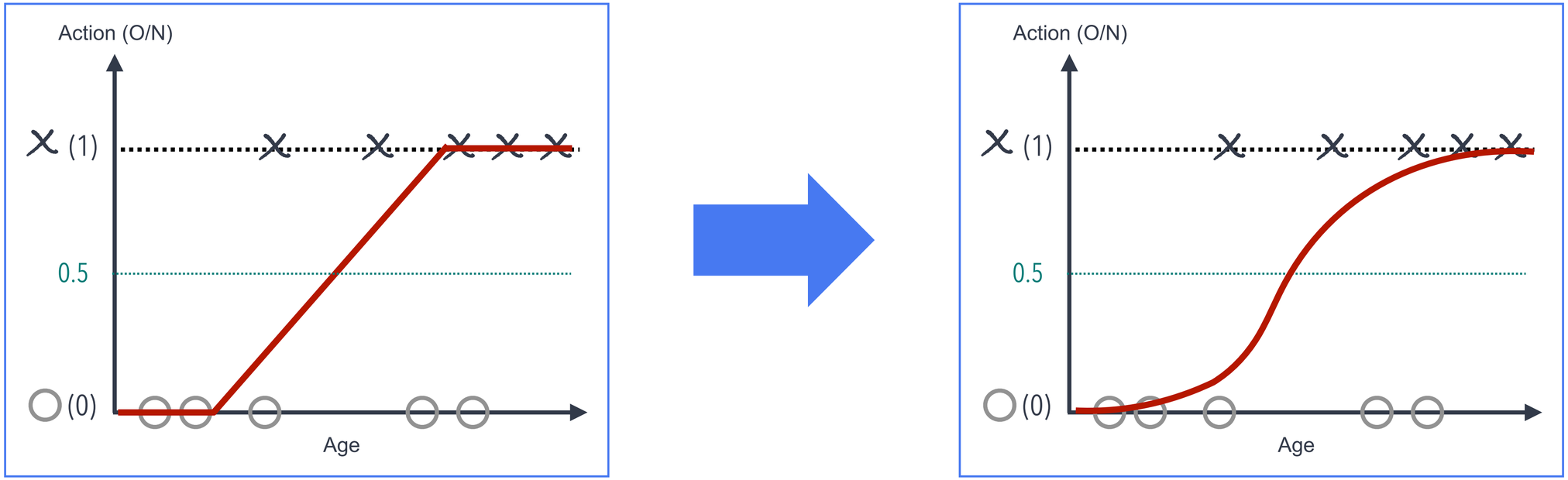
$$h(x) = P(y = 1 \mid x, w)$$



LA FONCTION HYPOTHÈSE

Régression Logistique

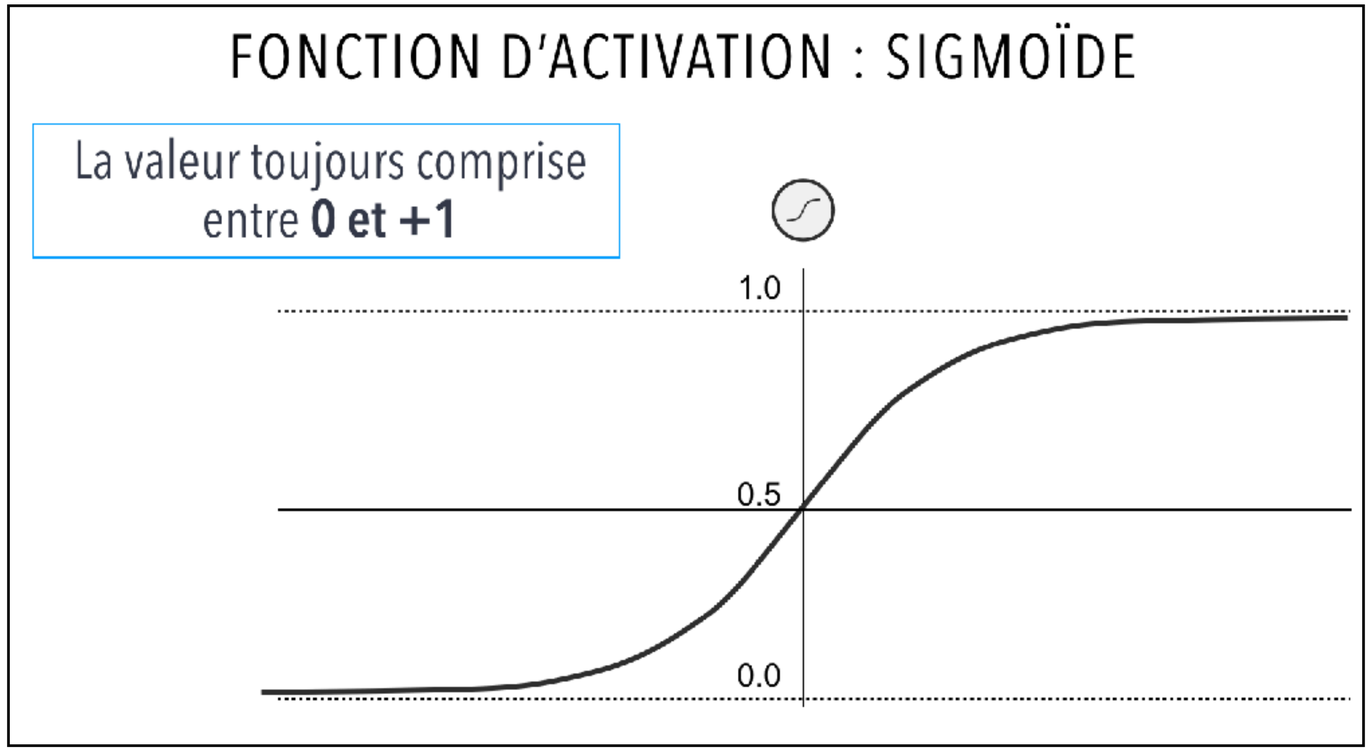
- On ne prend pas l'équation linéaire pour **déterminer les probabilités**.
- On utilise la fonction « **Sigmoïde** ».



LA FONCTION HYPOTHÈSE

Régression Logistique

- Les points de la fonction « **Sigmoïde** » se rapproche beaucoup mieux des **points d'observation** que les points de la fonction linéaire.
- On l'appelle **fonction sigmoïde** de la régression logistique.



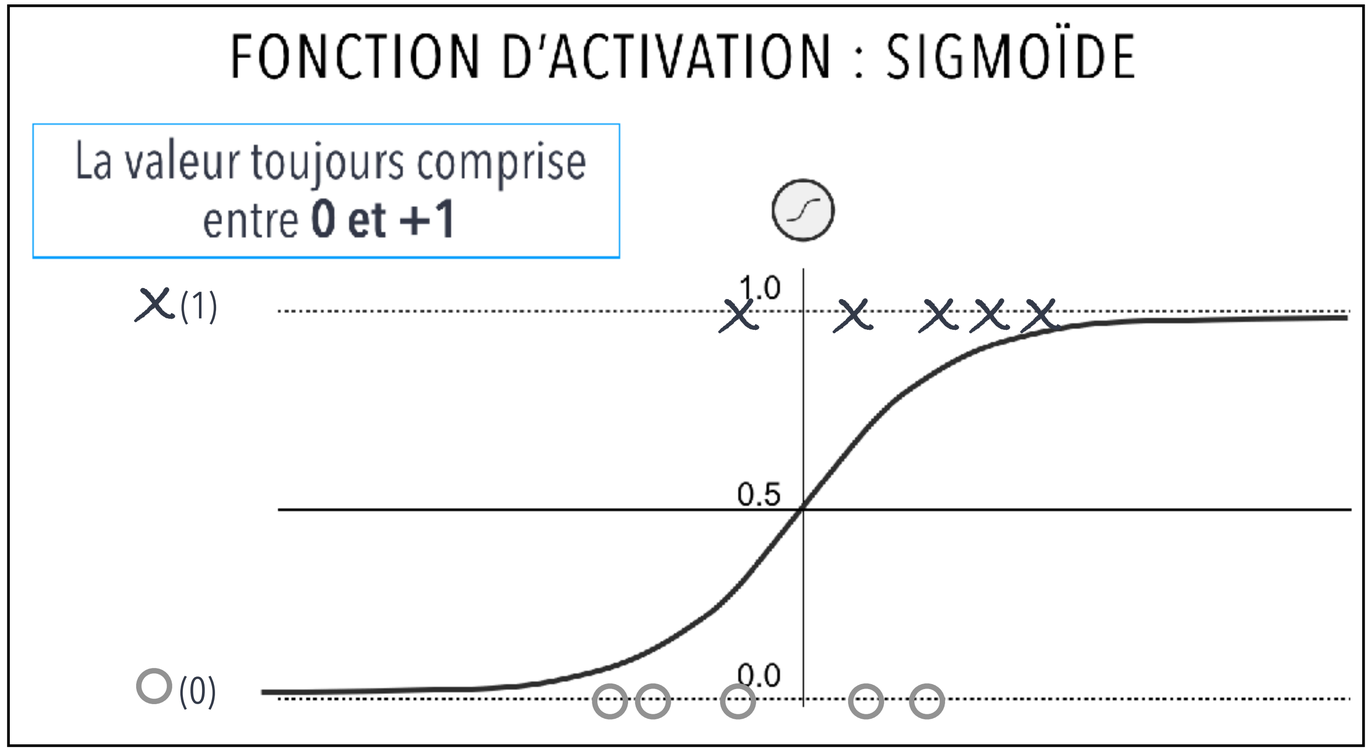
$$y = \frac{1}{1 + e^{-x}}$$

$$y = \frac{e^{x}}{1 + e^{x}}$$

LA FONCTION HYPOTHÈSE

Régression Logistique

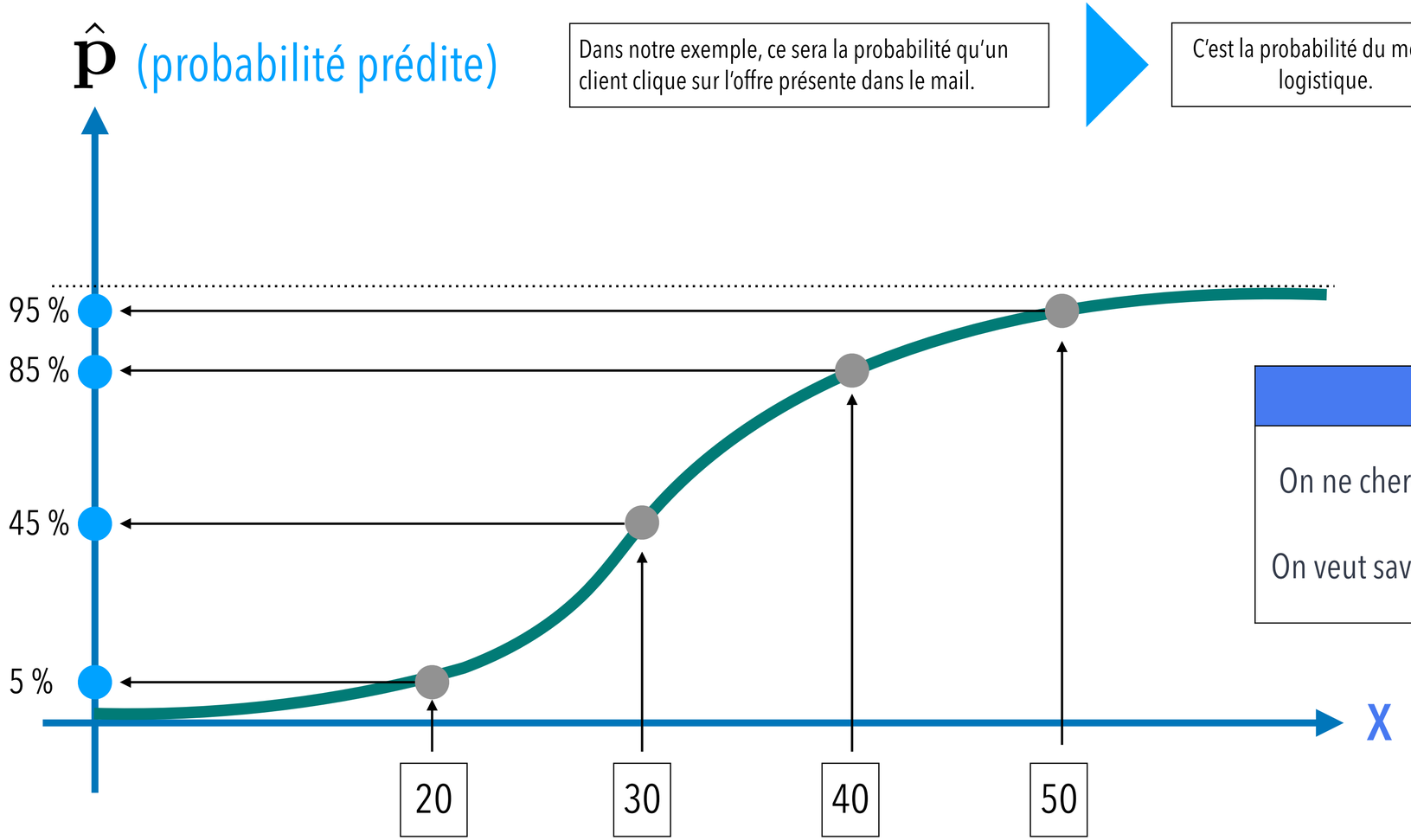
- Les points de la fonction « **Sigmoïde** » se rapprochent beaucoup mieux des **points d'observation** que les points de la fonction linéaire.
- On l'appelle **fonction sigmoïde** de la régression logistique.



$$y = \frac{1}{1 + e^{-x}}$$

$$y = \frac{e^{x}}{1 + e^{x}}$$

LA FONCTION HYPOTHÈSE : ILLUSTRATION

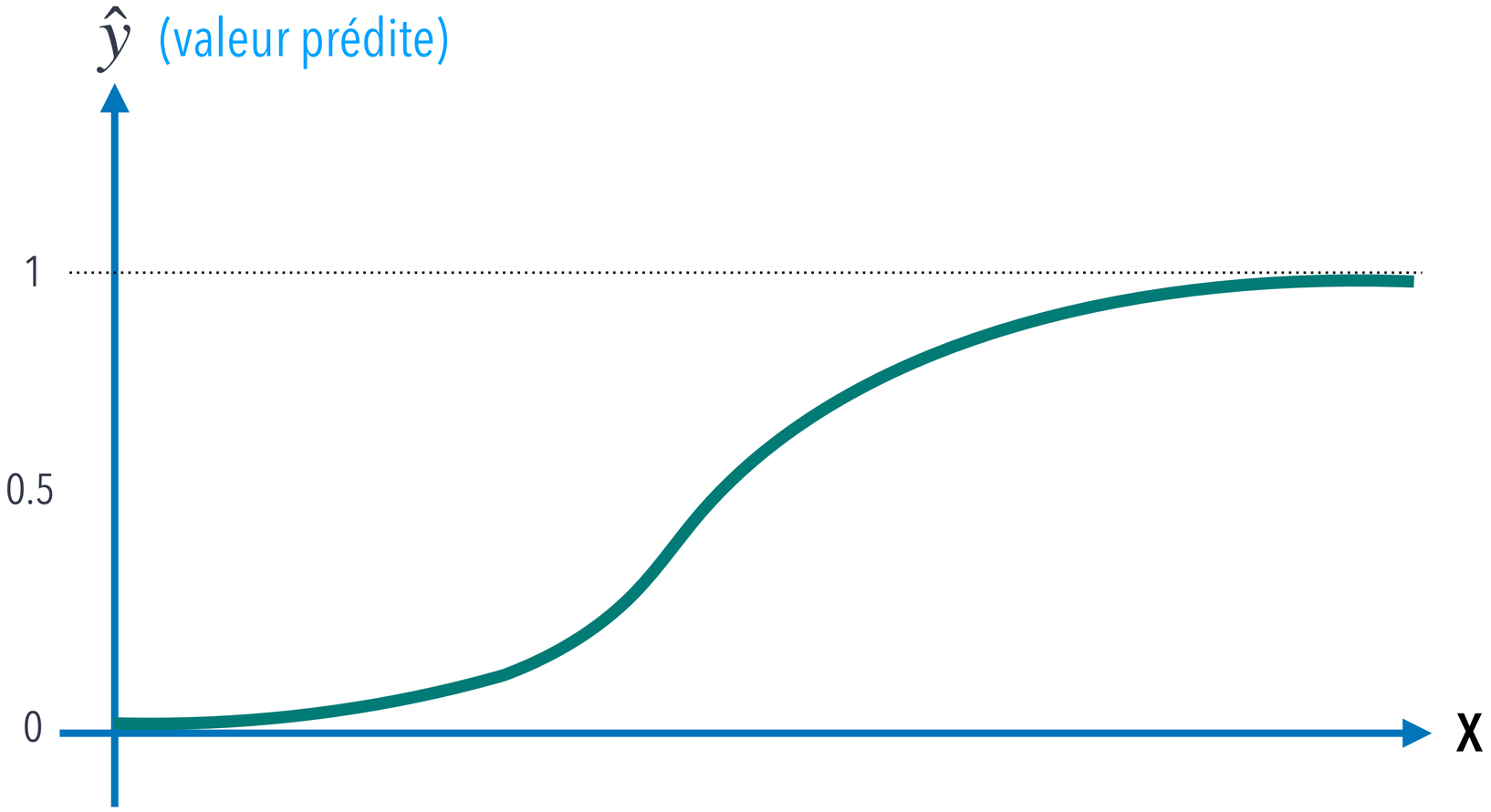


Remarque

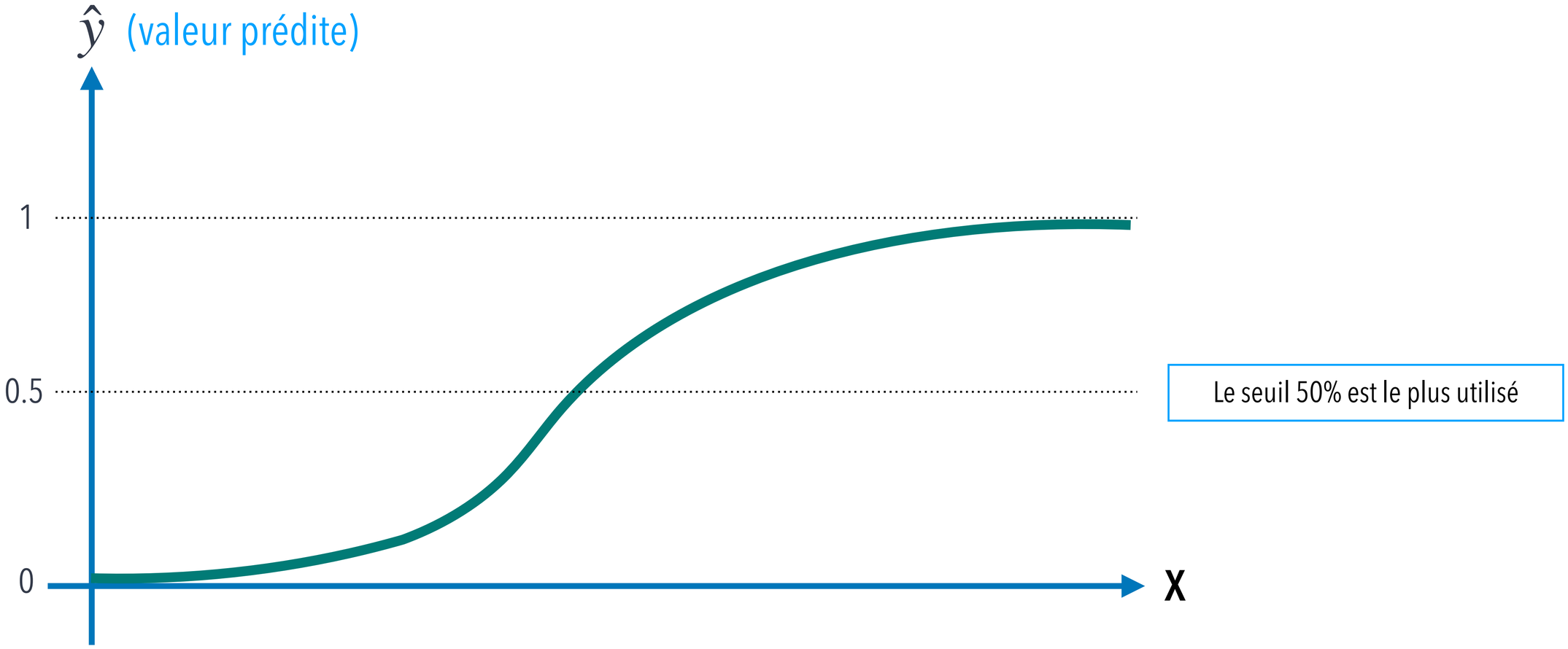
On ne cherche pas la **probabilité** qu'un client clique sur le lien.
On veut savoir s'il va **cliquer ou pas** sur le lien.

Ce modèle est très utilisé pour les classements suivant les probabilités prédites (images, médecine, …).
Par exemple, suivant les probabilités d'une tumeur, on classera les patients par ordre de priorité.

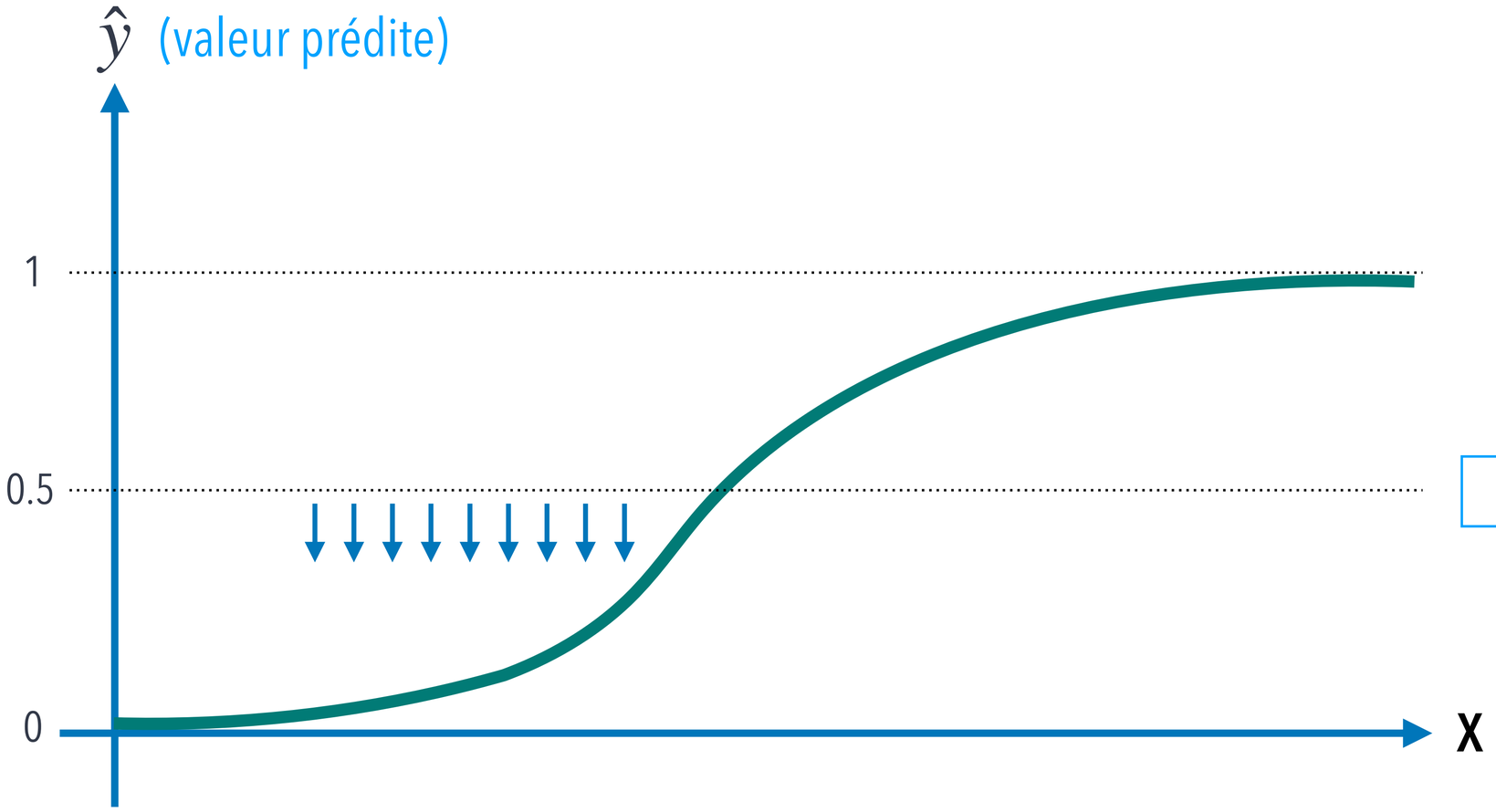
LA FONCTION HYPOTHÈSE : ILLUSTRATION



LA FONCTION HYPOTHÈSE : ILLUSTRATION



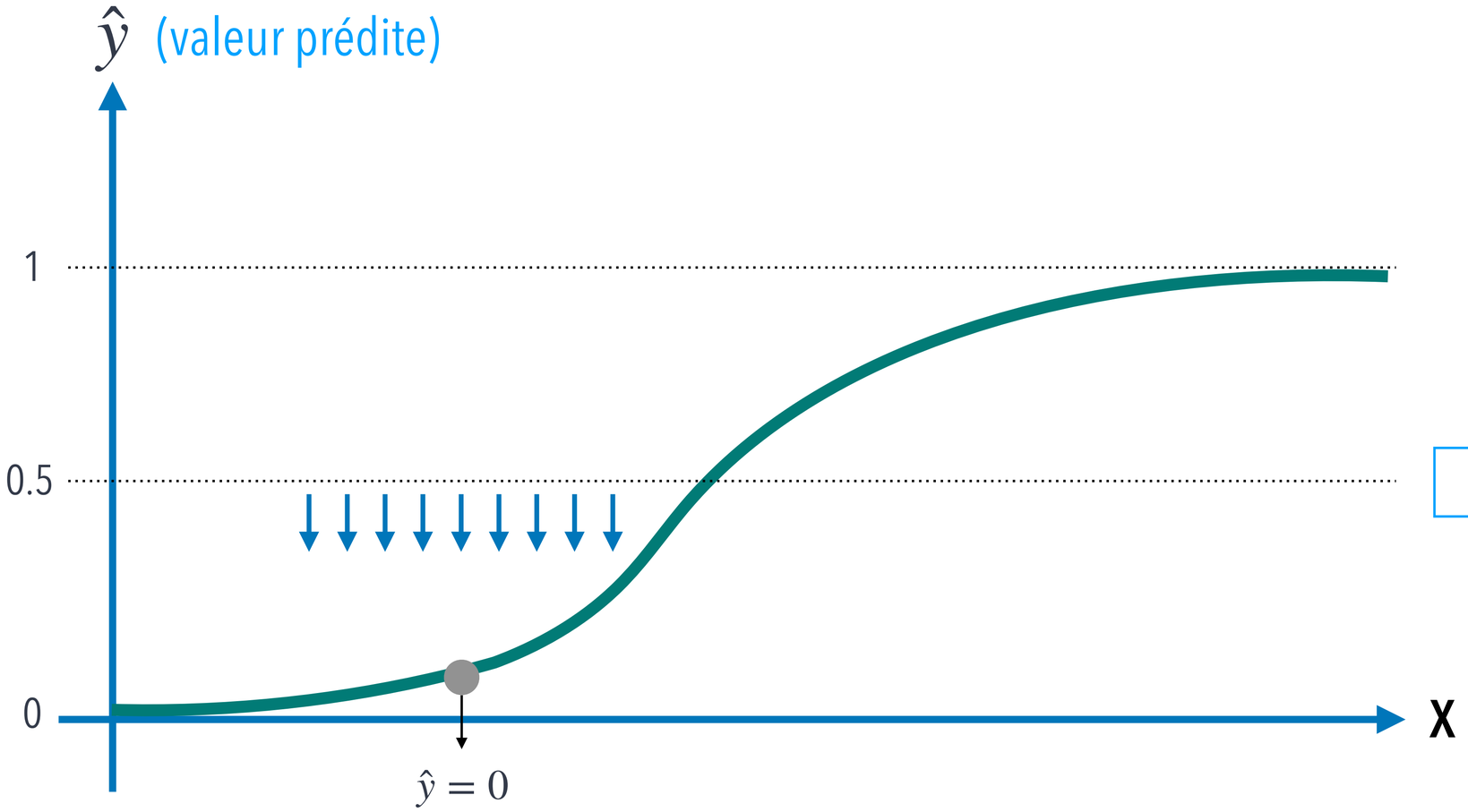
LA FONCTION HYPOTHÈSE : ILLUSTRATION



Le seuil 50% est le plus utilisé

Si la probabilité prédite est inférieure au seuil alors la valeur prédite est égale à 0.

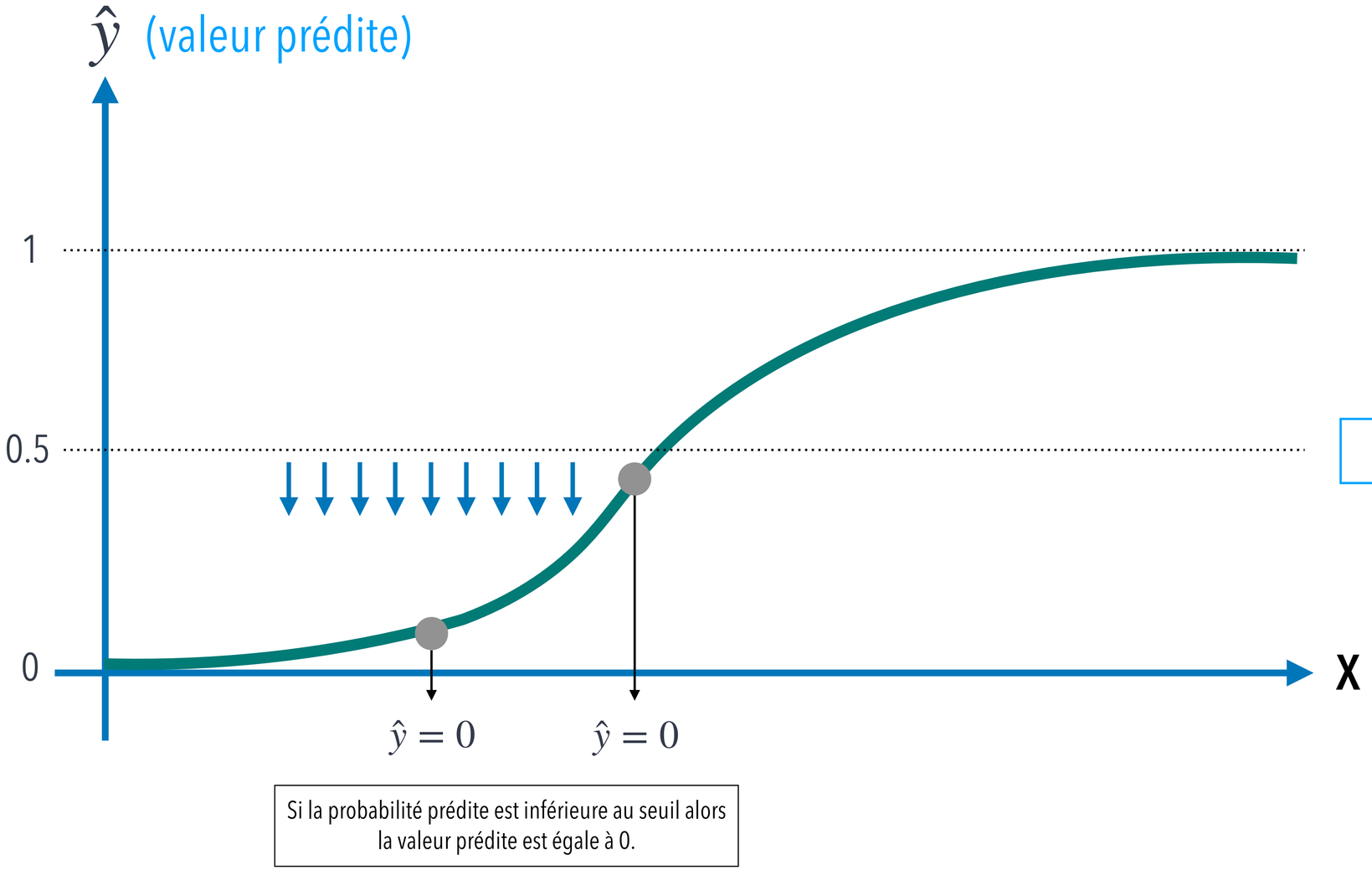
LA FONCTION HYPOTHÈSE : ILLUSTRATION



Le seuil 50% est le plus utilisé

Si la probabilité prédite est inférieure au seuil alors la valeur prédite est égale à 0.

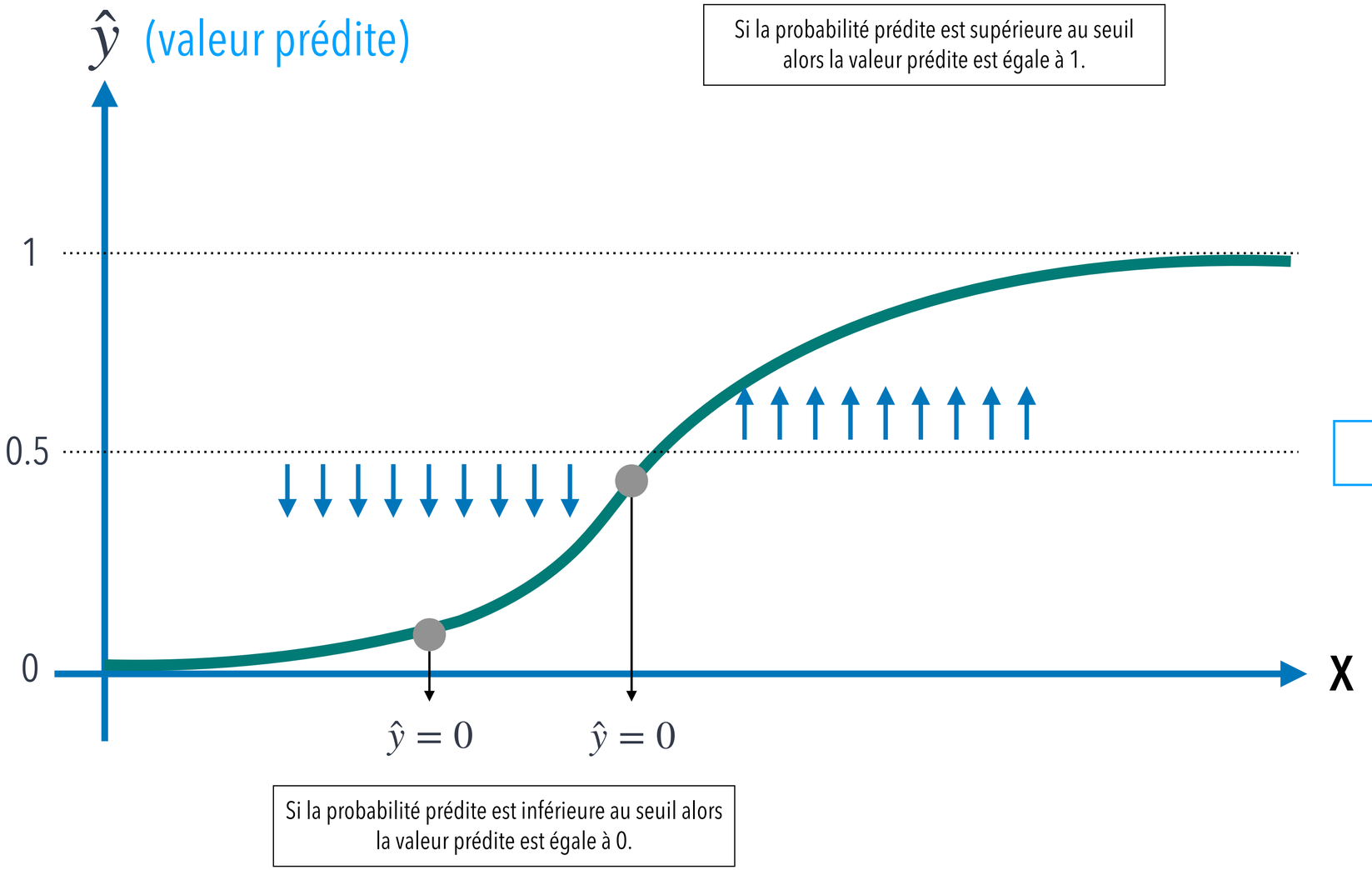
LA FONCTION HYPOTHÈSE : ILLUSTRATION



Le seuil 50% est le plus utilisé

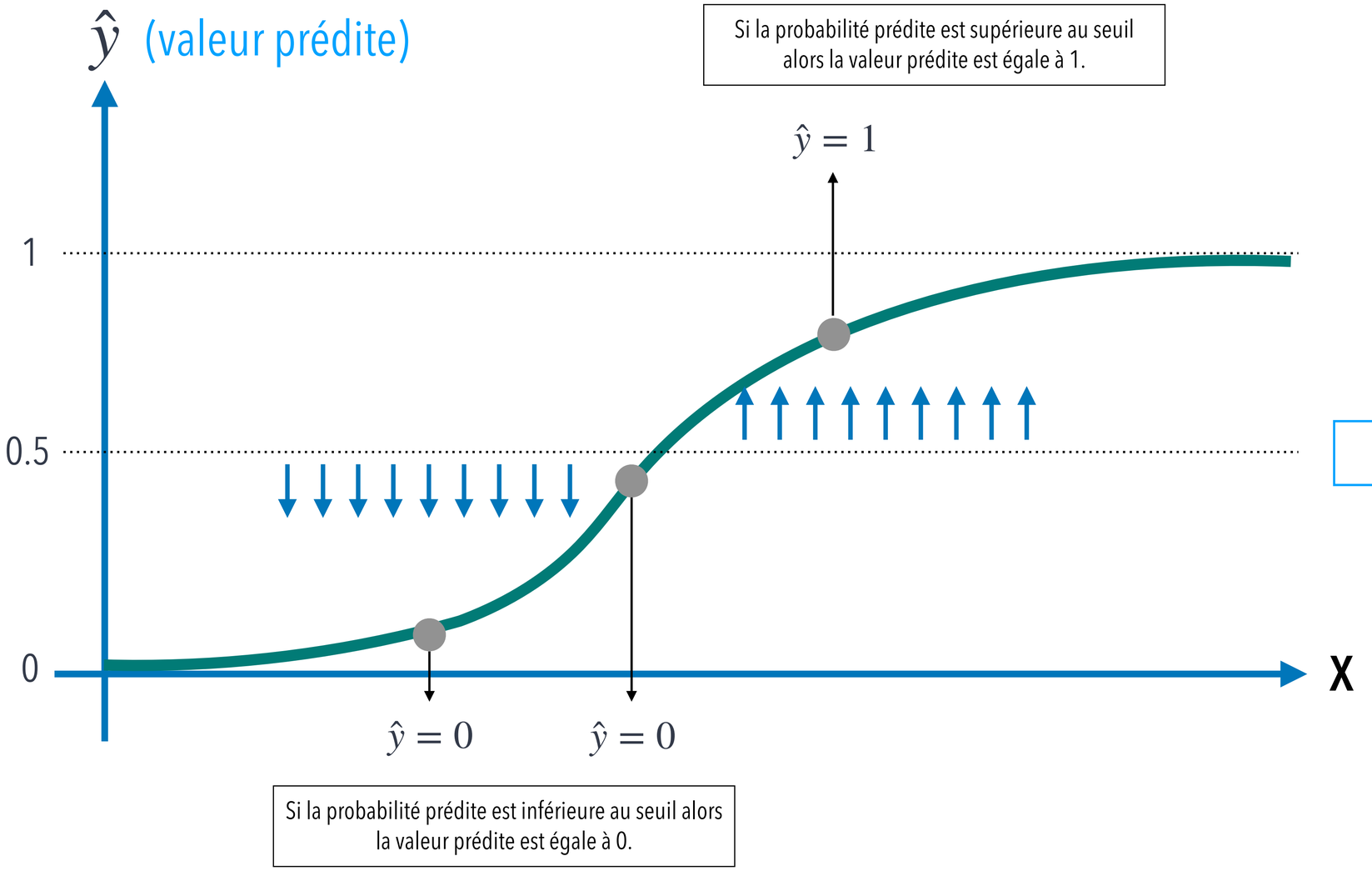
Si la probabilité prédite est inférieure au seuil alors la valeur prédite est égale à 0.

LA FONCTION HYPOTHÈSE : ILLUSTRATION



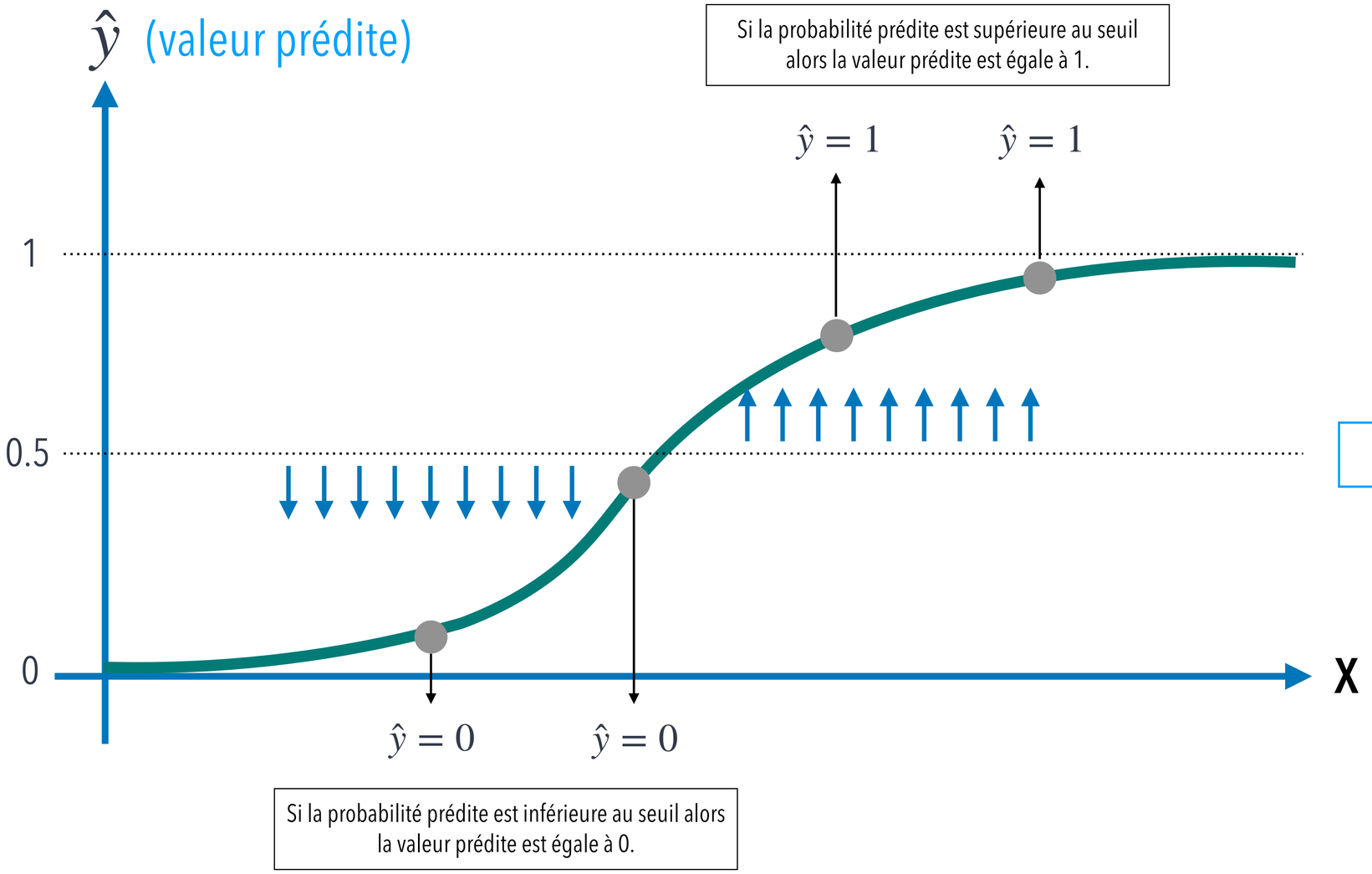
Le seuil 50% est le plus utilisé

LA FONCTION HYPOTHÈSE : ILLUSTRATION



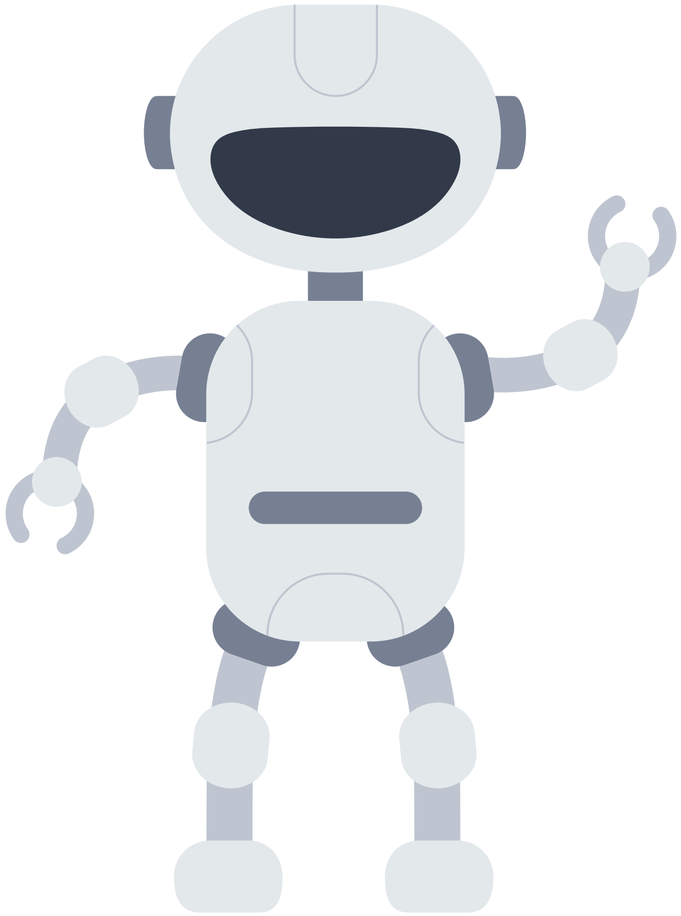
Le seuil 50% est le plus utilisé

LA FONCTION HYPOTHÈSE : ILLUSTRATION



Le seuil 50% est le plus utilisé

SOMMAIRE

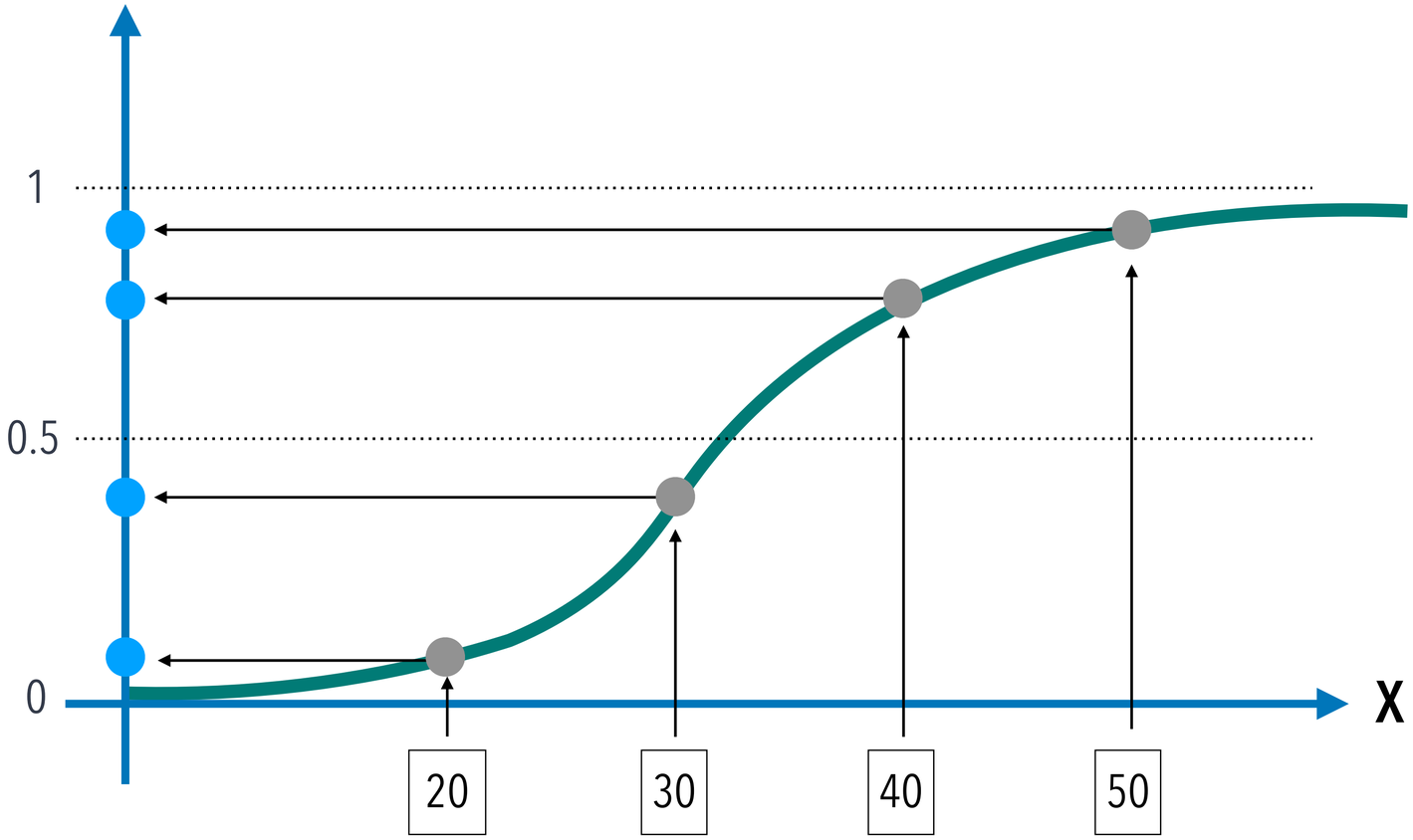


ANALYSE DU MODÈLE

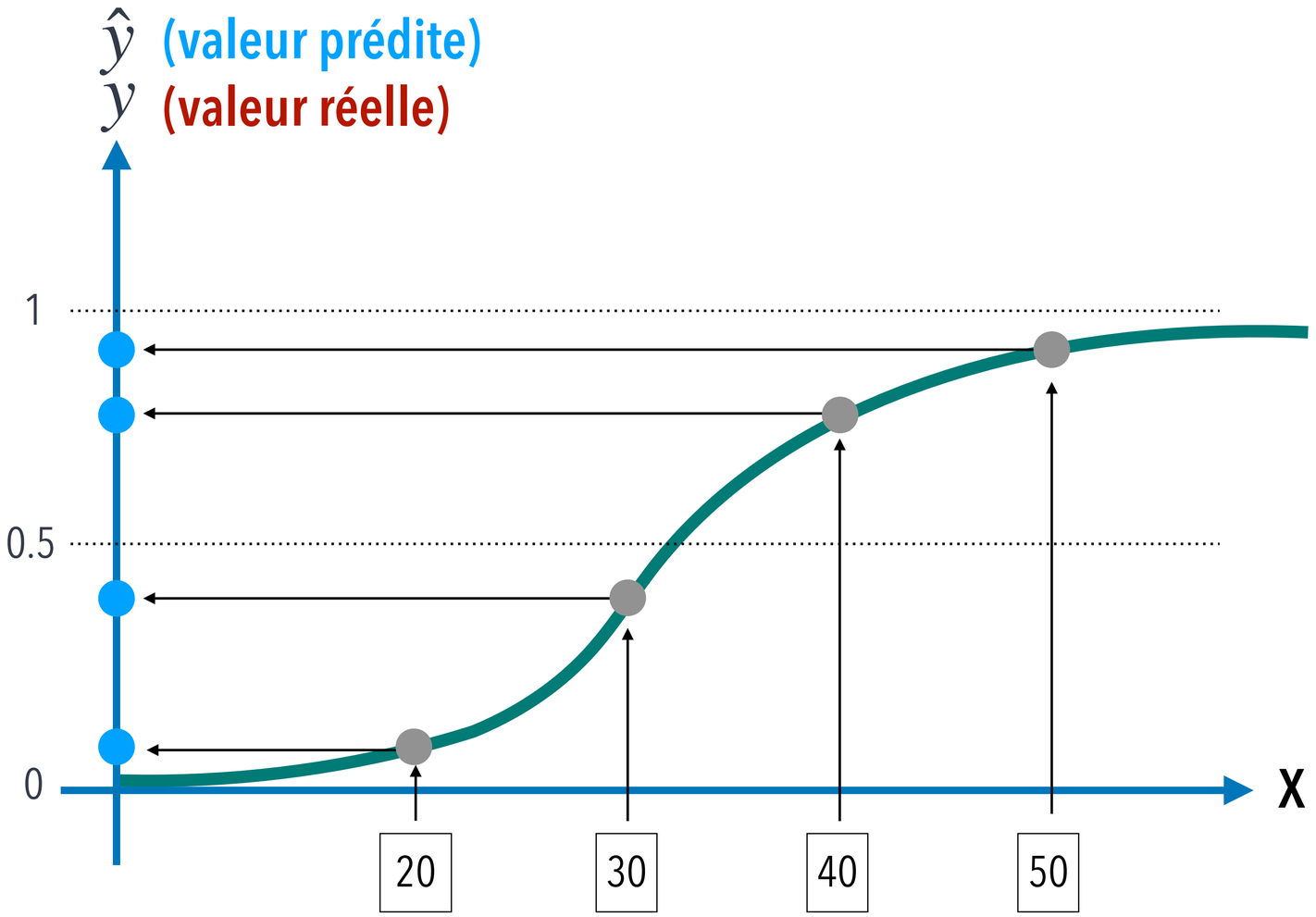
False Positives • False Negatives

FALSE POSITIVE • FALSE NEGATIVE

$\hat{y}$ **(valeur prédite)**



FALSE POSITIVE • FALSE NEGATIVE

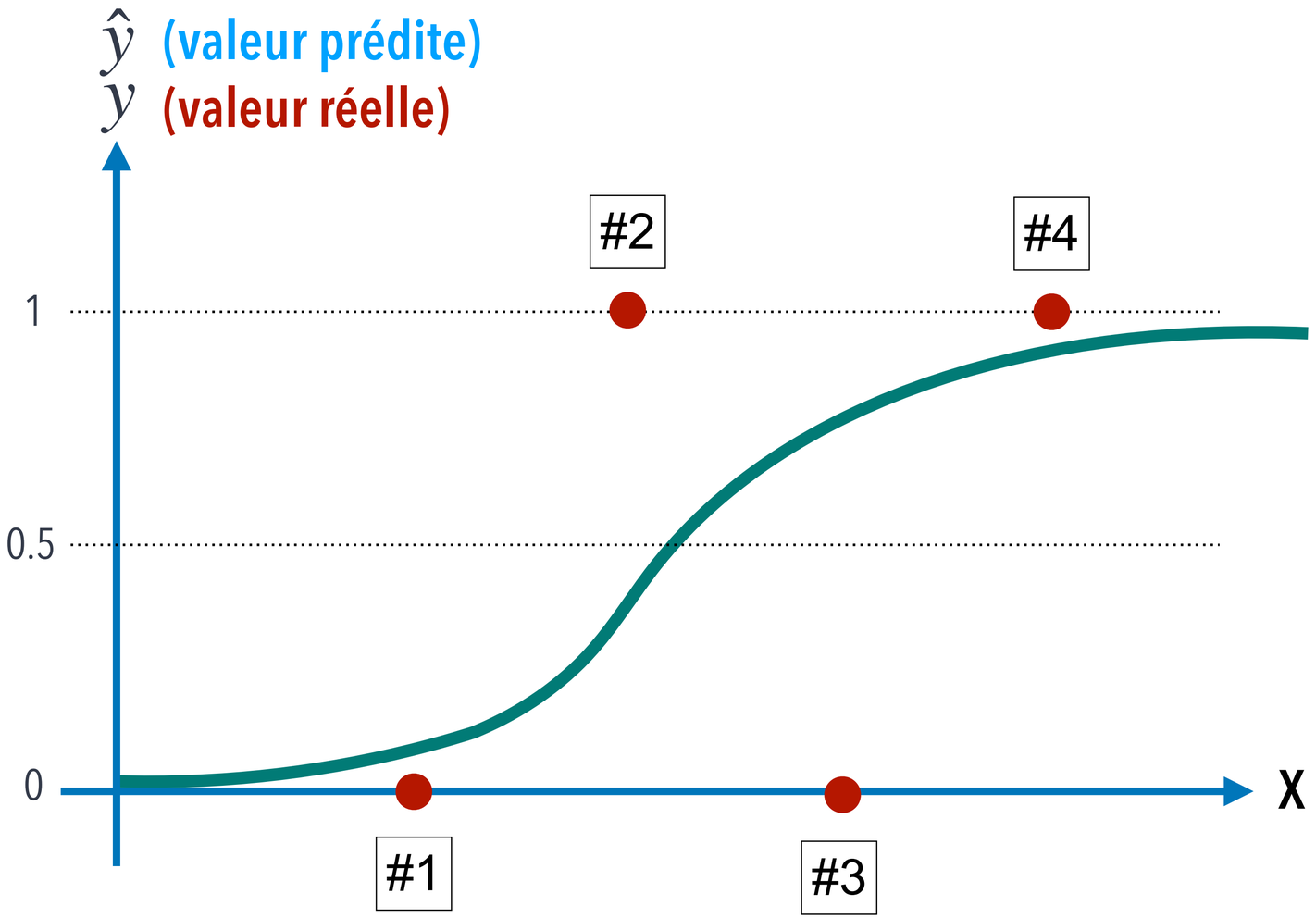


On va maintenant comparer les valeurs **prédites** et les valeurs **réelles** des quatre points d'observation.

Pour **chaque observation**, on indique :

- La **valeur prédite** : $\hat{y}$
- La **valeur réelle** : y

FALSE POSITIVE • FALSE NEGATIVE

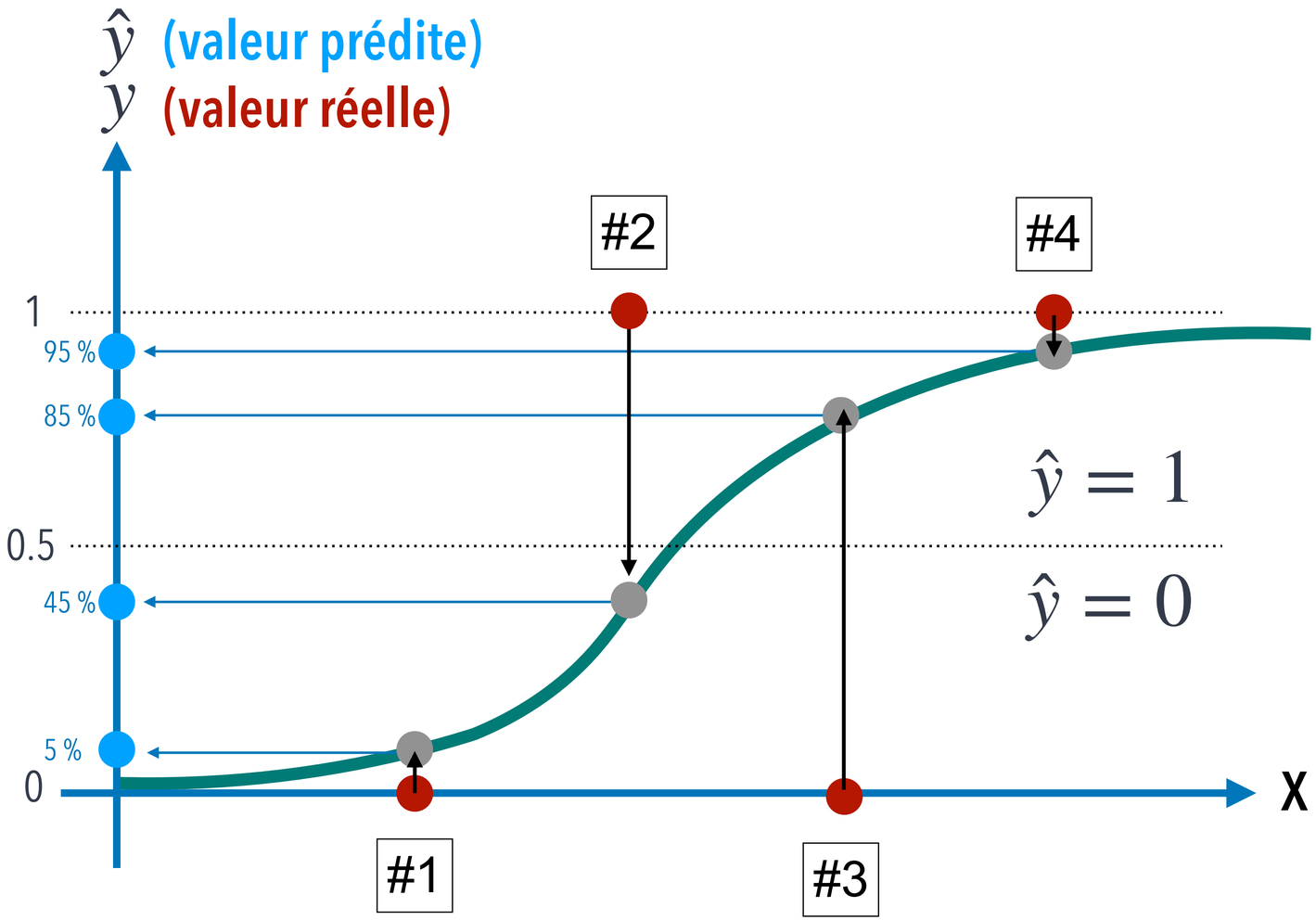


On va maintenant comparer les valeurs **prédites** et les valeurs **réelles** des quatre points d'observation.

Pour **chaque observation**, on indique :

- La **valeur prédite** : $\hat{y}$
- La **valeur réelle** : y

FALSE POSITIVE • FALSE NEGATIVE

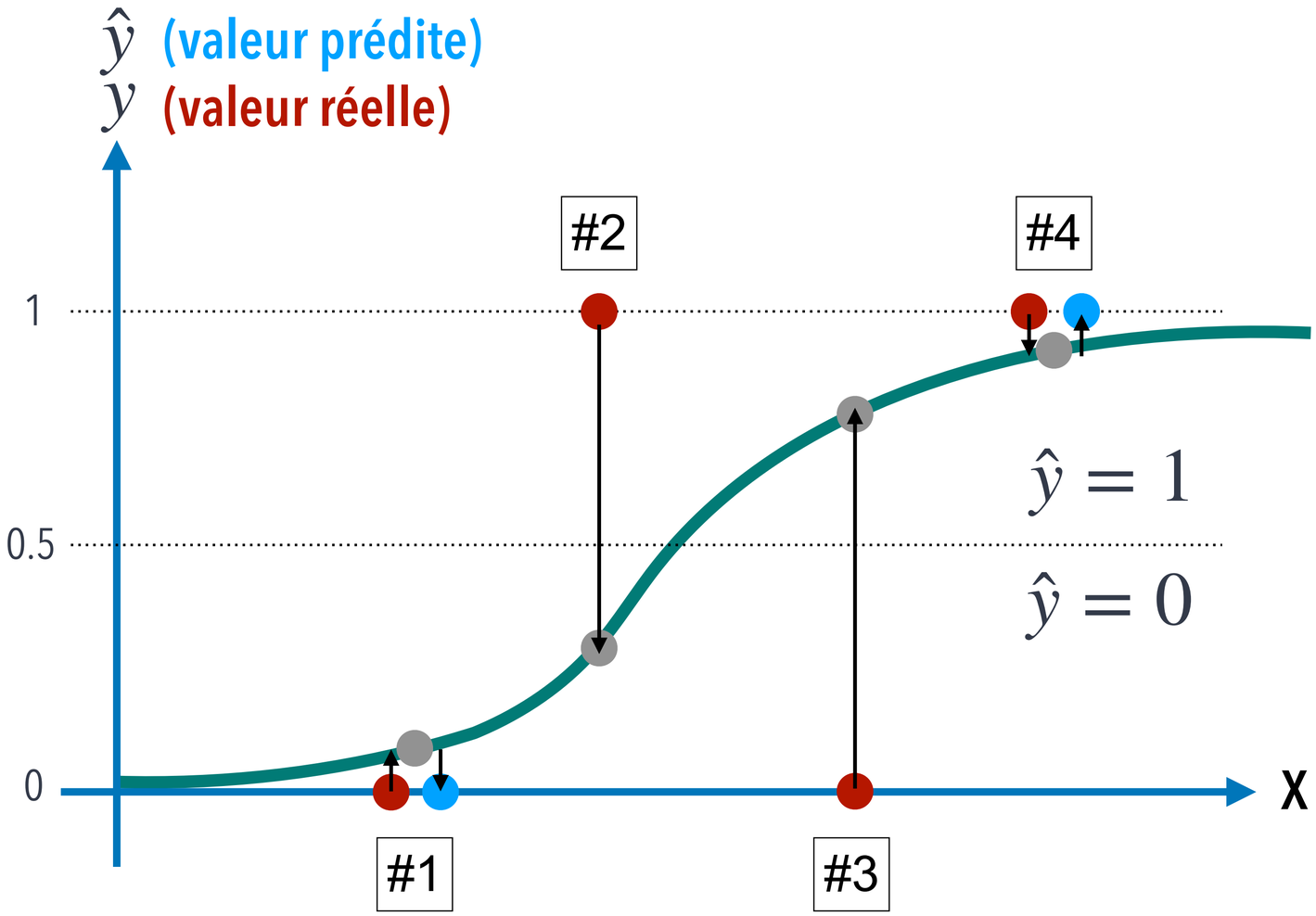


On va maintenant comparer les valeurs **prédites** et les valeurs **réelles** des quatre points d'observation.

➡ Pour cela, on comptabilise ce qu'on appelle :

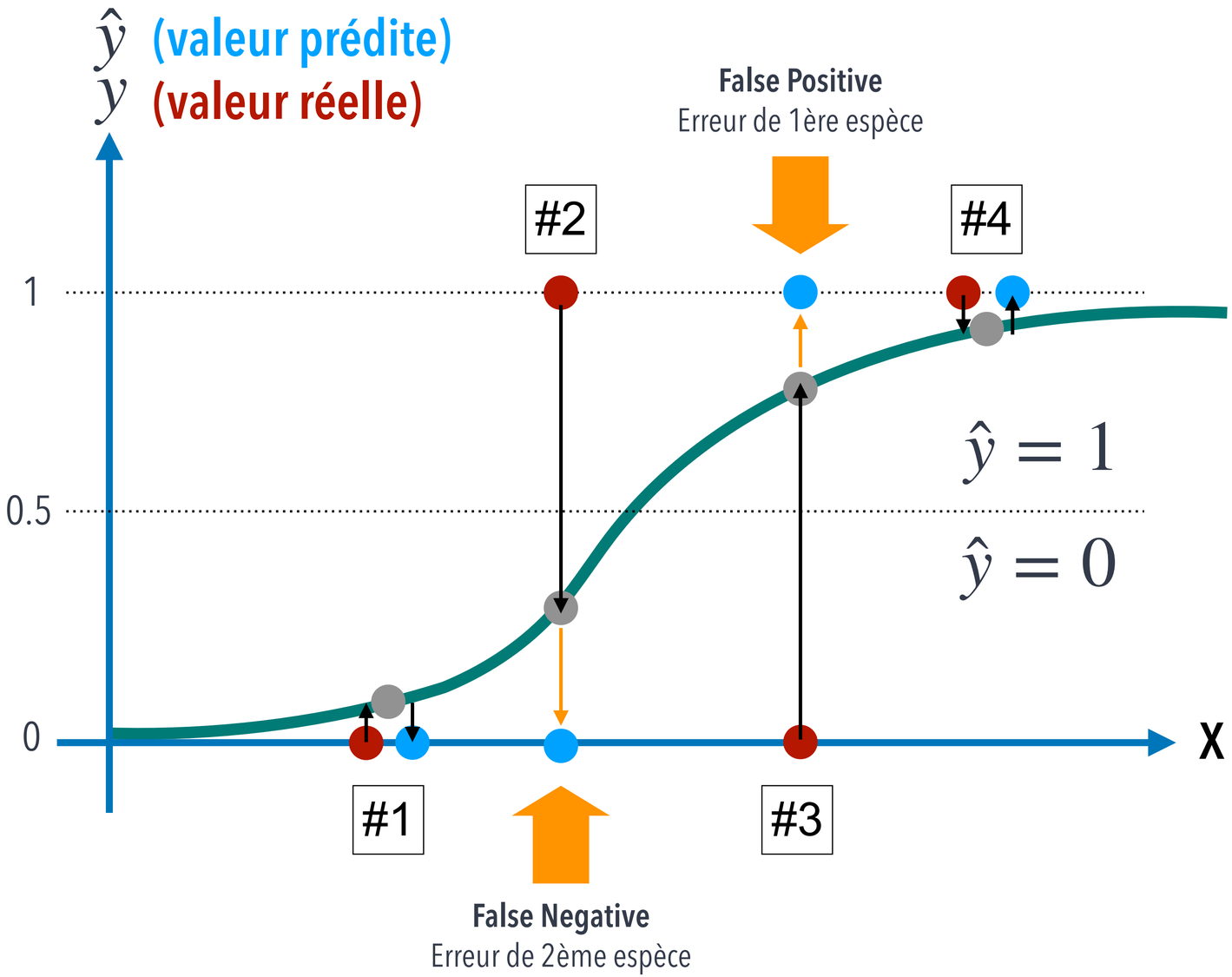
- Les **False Positives** : Erreurs Positives
- Les **False Negatives** : Erreurs Négatives

FALSE POSITIVE • FALSE NEGATIVE



Les prédictions des points #1 et #4 sont en **accord avec les observations**.

FALSE POSITIVE • FALSE NEGATIVE

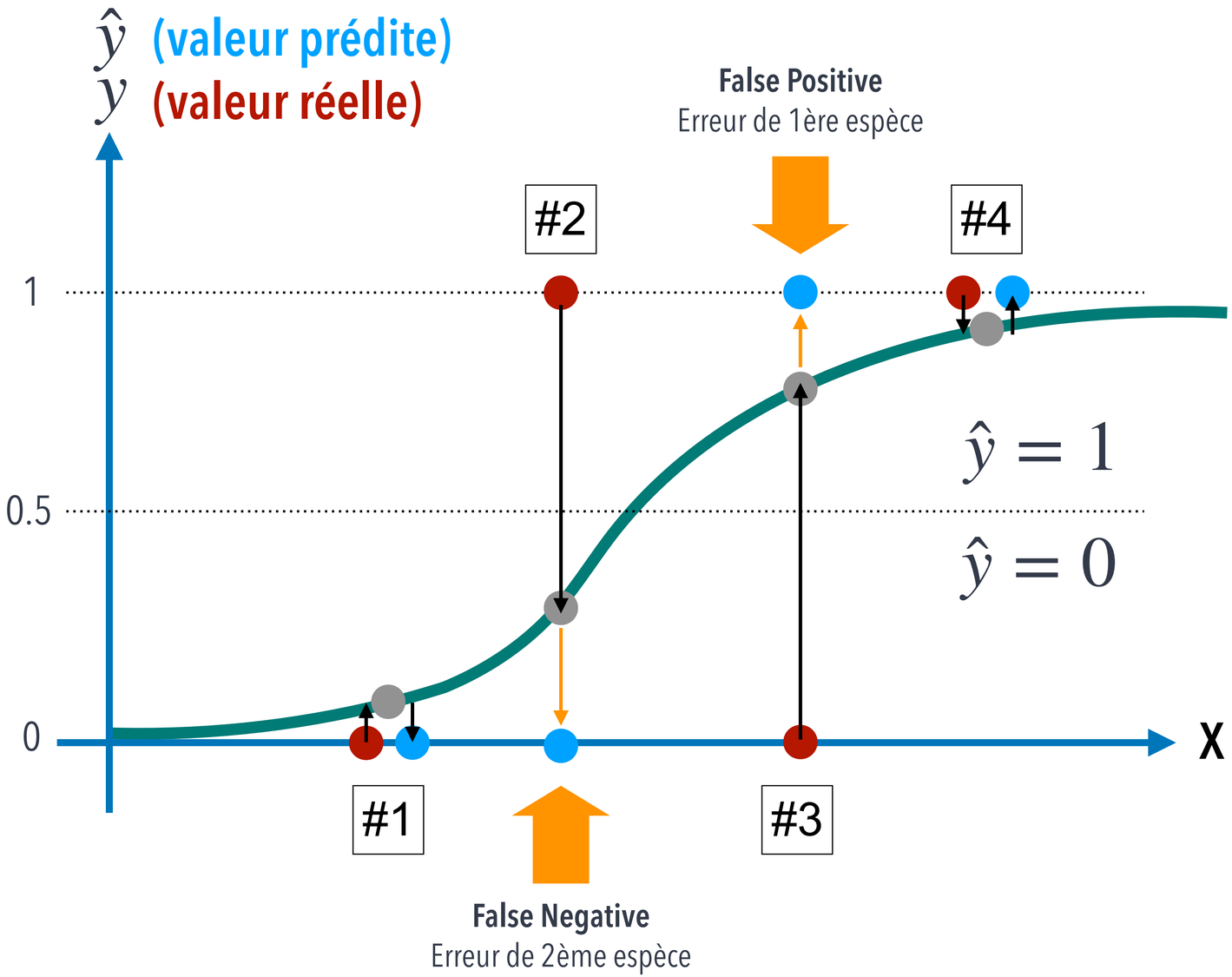


Les prédictions des points #1 et #4 sont en **accord avec les observations**.

Les prédictions des points #2 et #3 sont en **désaccord avec les observations**.

- Le modèle a prévu que #2 ne clique pas ($\hat{y} = 0$) alors qu'il a cliqué (`y=1`). On dit qu'il a fait une erreur (`False`) et que l'erreur est négative (en dessous du seuil).
- Le modèle a prévu que #3 clique ($\hat{y} = 1$) alors qu'il n'a pas cliqué (`y=0`). On dit qu'il a fait une erreur (`False`) et que l'erreur est positive (au dessus du seuil).

FALSE POSITIVE • FALSE NEGATIVE



Différences entre les deux types d'erreurs

False Positive

False Positive est de nature « **Avertissement** »

- On prévoit un évènement qui n'a pas lieu (ici un clic d'achat).
- On prévoit qu'il va y avoir un tremblement de terre qui n'aura pas lieu.

False Negative

False Positive est de nature « **plus grave** »

- On ne prévoit pas un évènement qui a lieu (clic d'achat).
- Pas grave dans le cas d'un clic, mais ne pas prévoir un tremblement de terre qui va avoir lieu, c'est très grave.

FALSE POSITIVE • FALSE NEGATIVE

FALSE POSITIVES INDÉSIRABLES

L'accident mortel de la voiture autonome d'Uber[1]

- Les faits remontent à mars 2018 où une voiture autonome de la firme Uber a **heurté mortellement** une femme dans la ville de Tempe dans l'Arizona alors qu'elle traversait en dehors d'un passage piéton. Bien qu'un conducteur humain soit toujours au volant de la voiture, celle-ci était en **mode autonome** au moment de l'impact.
- Pendant les six secondes qui ont précédé l'impact, le système de conduite autonome a classé le piéton en tant qu'**objet inconnu**, puis en tant que **véhicule**, puis en tant que **bicyclette**. Bien que le système ait identifié la nécessité d'une manœuvre de **freinage d'urgence** pour atténuer une collision, le système a été configuré pour ne pas activer **systématiquement** (dans tous les cas) le freinage d'urgence lorsqu'il est piloté par l'ordinateur.
- En fait, le système de la voiture est conçu de manière à ce que les manœuvres d'urgence soient plutôt (toujours) à la **charge de l'opérateur humain**. De ce fait, quel est le véritable degré d'autonomie d'une voiture incapable de prendre des décisions d'une telle importance en temps réel ?
- Mettre en place une voiture autonome sur laquelle le freinage d'urgence n'est pas **systématiquement** activé (apparemment parce que cela crée trop de **false positives**[2]) est un risque injustifiable. Le freinage d'urgence devrait être la première des nécessités dans l'autonomie des véhicules, d'où la possible responsabilité de Uber.
- La responsabilité pénale d'Uber n'a **pas été engagée**, seule la conductrice a été mise en cause.

1. Source : La responsabilité pénale d'Uber.
2. On détecte un objet comme étant une personne (détection positive de l'obstacle humain) alors que ce n'est pas le cas (erreur). On a donc une erreur positive (on détecte un évènement qui n'a pas lieu).

MATRICE DE CONFUSION

y (valeur réelle) \ $\hat{y}$ (valeur prédite)	0	1
0	True Negative **35**	False Positive (Erreur de 1ère espèce) **5**
1	False Negative (Erreur de 2ème espèce) **10**	True Positive **50**

Résultats avec

- Nombre total de prédictions : 35 + 5 +10 + 50 = 100.
- Nombre total de prédictions correctes : 35 + 50 = 85.
- Nombre total de prédictions incorrectes : 5 + 10 = 15.

Calcul de deux TAUX

1. **Accuracy Rate** (taux de réussite)
   AR = Correct / Total
   AR = 851/100 = 85 %
2. **Error Rate** (taux d'erreur)
   ER = Incorrect / Total
   AR = 152/100 = 15 %

ANALYSE DU MODÈLE

Les trois étapes

RETOUR SUR LA RÉGRESSION LINÉAIRE

Régressions linéaire univariées

La recherche du **meilleur modèle** de régression linéaire (univarié ou pas) ou polynomiale repose sur **trois étapes** essentielles :

01 *La définition d'une **fonction hypothèse***

02 *La construction d'une **fonction de coût***

03 *La **minimisation** de la fonction de coût*

LA FONCTION HYPOTHÈSE

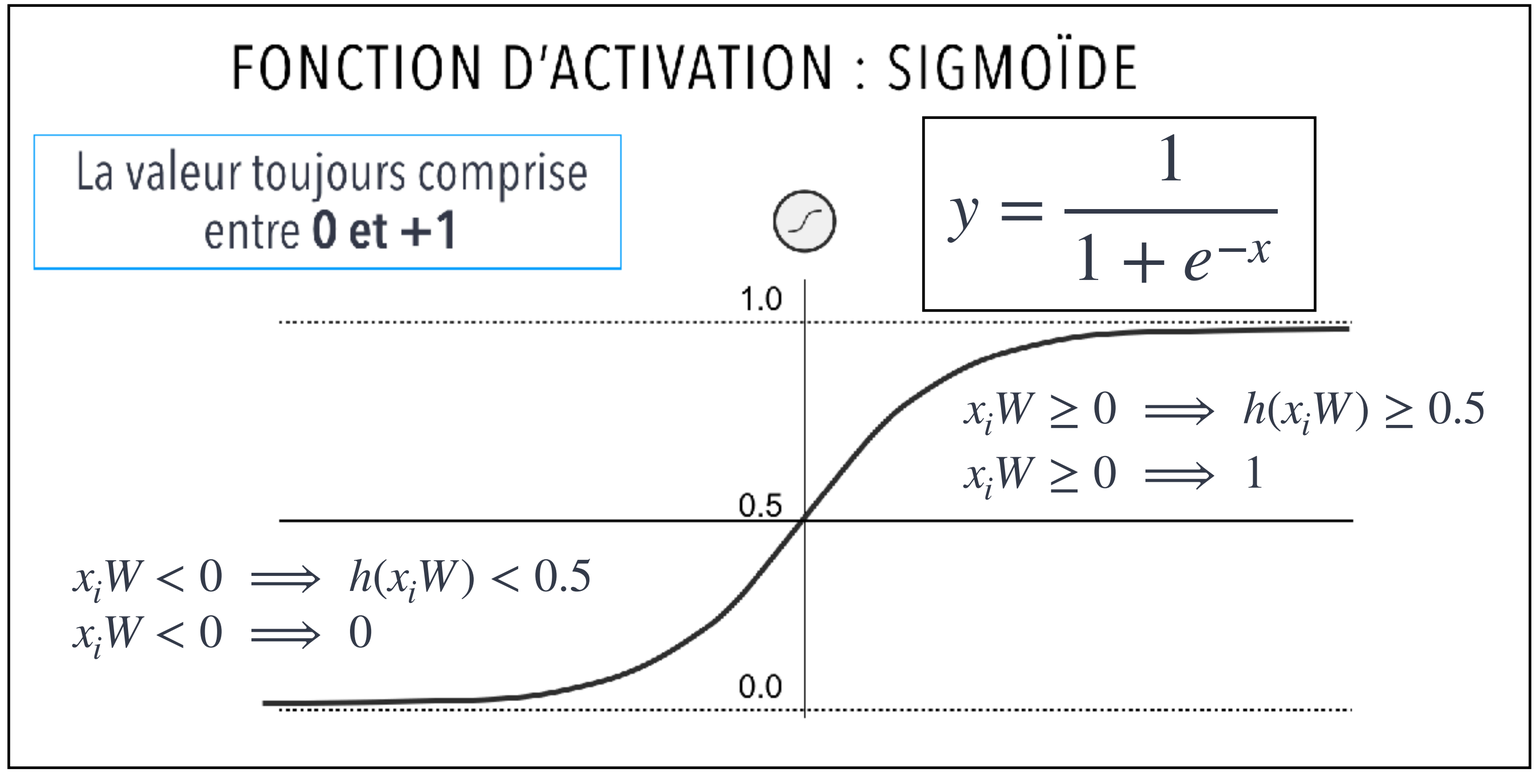
01 *La définition d'une **fonction hypothèse***

- **Régression linéaire** : pour l'observation x_i, la prédiction de la valeur continue étant donnée x_i et W est donnée par :

$$h_{li}(x_i W) = x_i W = w_0 + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in}$$

- **Régression logistique** : pour l'observation x_i, la probabilité de prédire un 1 étant donnée x_i et W est donnée par :

$$h_{lo}(x_i W) = \text{Sigmoid}(w_0 + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in})$$



- Le résultat, d'après h_{lo}, est :

$$\begin{cases} 0, & sigmoid(x_i W) < 0.5 \\ 1, & sigmoid(x_i W) \geq 0.5 \end{cases}$$

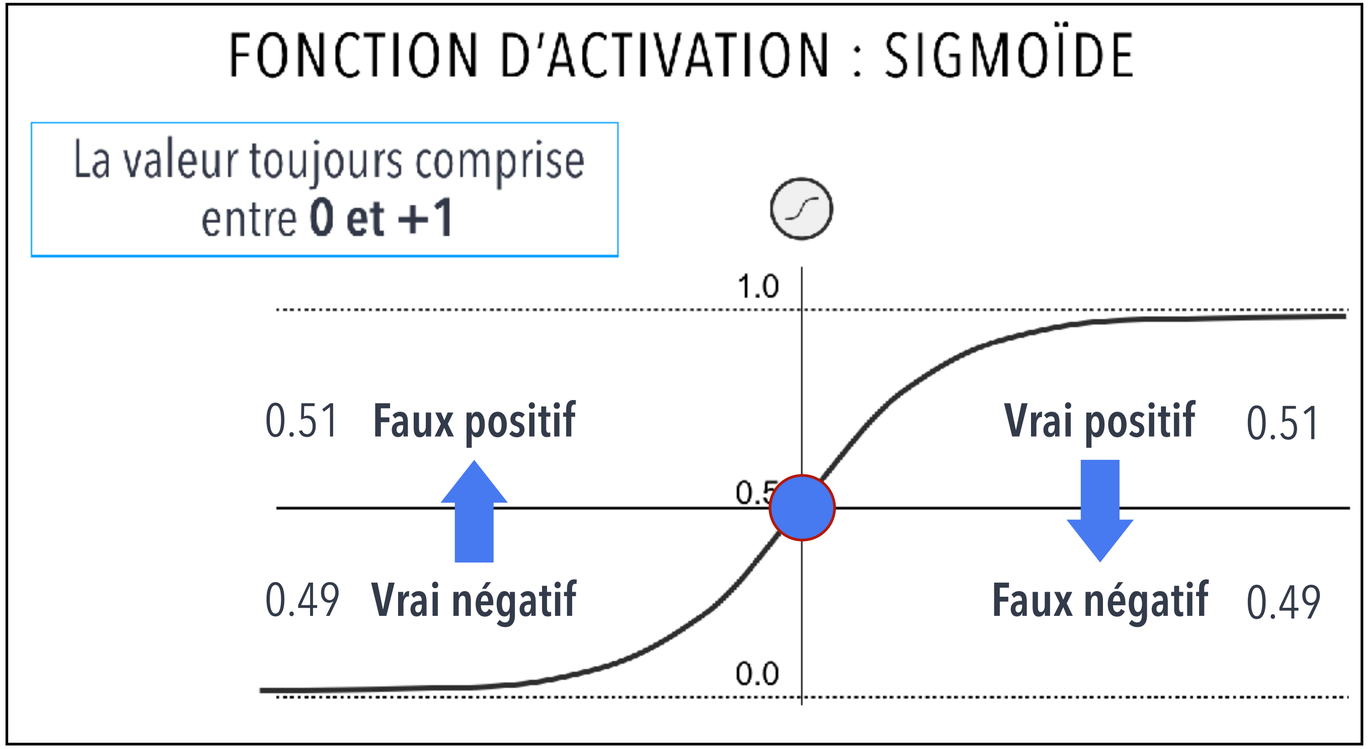
- Ce qui est équivalent à :

$$\begin{cases} 0, & x_i W < 0 \\ 1, & x_i W \geq 0 \end{cases}$$

LA FONCTION HYPOTHÈSE

01 *La définition d'une **fonction hypothèse***

- Les prédictions **proches de la frontière** de décision sont **imprécises**.
- Il suffit de modifier quelque peu les **hyper-paramètres** ou les données d'**entrainement** pour passer d'une probabilité de **0.49 à 0.51**.
- Ou à l'inverse, passer d'une probabilité de **0.51 à 0.49**.

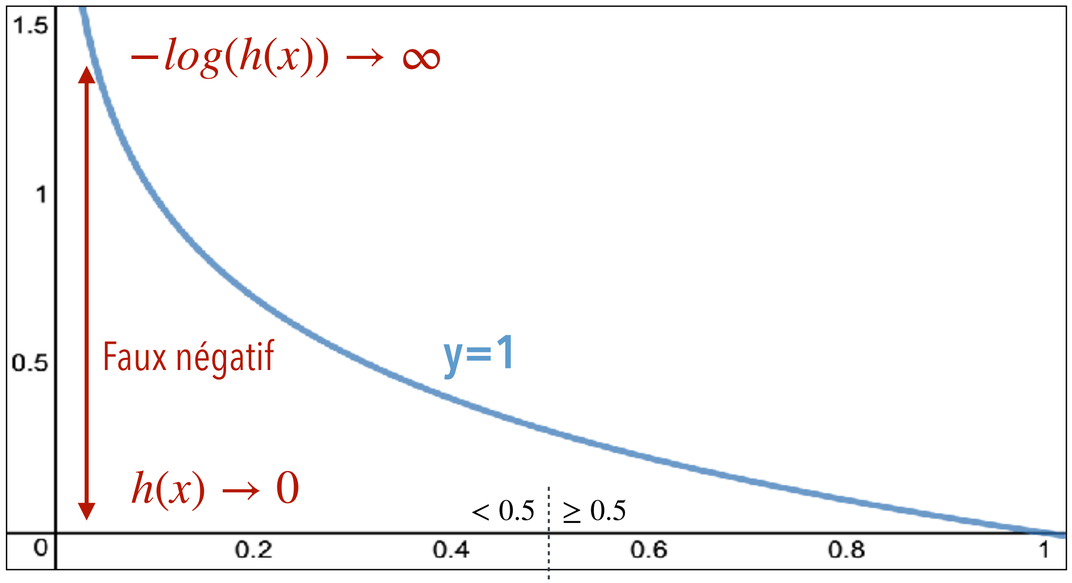


LA FONCTION DE COÛT

02 *La construction d'une fonction de coût*

- Pour la classification, on souhaite une fonction de coût qui **pénalise très fortement** les **faux positifs** et les **faux négatifs**.
- C'est le cas de la fonction par morceau suivante :

$$C(h(x), y) = \begin{cases} -log(h(x)), & y = 1 \\ -log(1 - h(x)), & y = 0 \end{cases}$$



Faux négatif

- **Faux négatif** : $\hat{y}$=0 alors que y=1.
- Valeur réelle y=1 la fonction de coût est : `C=-log(h(x))`
- Prédiction $\hat{y}$=0 implique `h(x)<0.5` donc possiblement proche de 0
- La fonction de coût `C` remplit son rôle puisqu'elle tend vers l'infini[1] : `log(0)`[2]

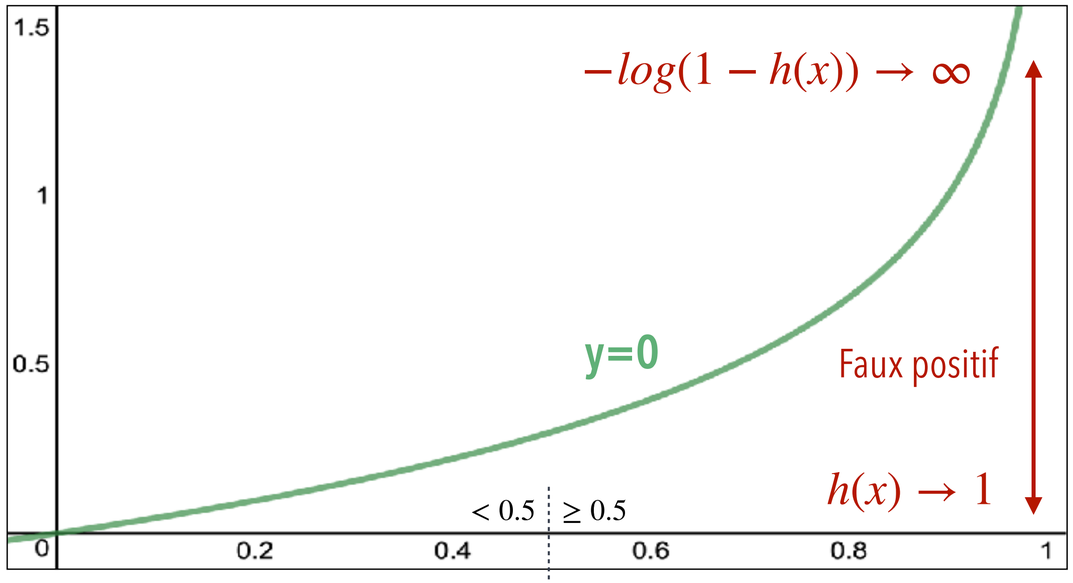
1. Les prédictions sont bornées pour éviter les trop fortes pénalités sur les faux positifs/négatifs.
2. Puisque pour `y=1`, la fonction de coût `C=-log(h(x))`. Avec `h(x)` qui tend vers `0`, `C` tend vers `log(0)`.

LA FONCTION DE COÛT

02 *La construction d'une fonction de coût*

- Pour la classification, on souhaite une fonction de coût qui **pénalise très fortement** les **faux positifs** et les **faux négatifs**.
- C'est le cas de la fonction par morceau suivante :

$$C(h(x), y) = \begin{cases} -log(h(x)), & y = 1 \\ -log(1 - h(x)), & y = 0 \end{cases}$$



Faux négatif

- **Faux négatif** : $\hat{y}$=0 alors que y=1.
- Valeur réelle y=1 la fonction de coût est : `C=-log(h(x))`
- Prédiction $\hat{y}$=0 implique `h(x)<0.5` donc possiblement proche de 0
- La fonction de coût `C` remplit son rôle puisqu'elle tend vers l'infini[1] : `log(0)`[2]

1. Les prédictions sont bornées pour éviter les trop fortes pénalités sur les faux positifs/négatifs.
2. Puisque pour `y=1`, la fonction de coût `C=-log(h(x))`. Avec `h(x)` qui tend vers `0`, `C` tend vers `log(0)`.

Faux positif

- **Faux positif** : $\hat{y}$=1 alors que y=0.
- Valeur réelle y=0 la fonction de coût est : `C=-log(1-h(x))`
- Prédiction $\hat{y}$=1 implique `h(x)>=0.5` donc possiblement proche de 1
- La fonction de coût `C` remplit son rôle puisqu'elle tend vers l'infini : `log(0)`[3]

3. Puisque pour `y=0`, la fonction de coût `C=-log(1-h(x))`. Avec `h(x)` qui tend vers `1`, `C` tend vers `log(0)`.

LA FONCTION DE COÛT

03 *La minimisation de la fonction de coût*

- La **fonction de coût** par morceau :

$$C(h(x), y) = \begin{cases} -log(h(x)), & y = 1 \\ -log(1 - h(x)), & y = 0 \end{cases}$$

- Puisque y est toujours égal à 0 ou 1, on peut plus simplement **réécrire la fonction de coût** pour l'apprentissage :

$$\begin{aligned} C(h(x), y) &= -ylog(h(x)) - (1 - y)log(1 - h(x)) \\ &= \begin{cases} -1log(h(x)) - (1 - 1)log(1 - h(x)) = -log(h(x)), & y = 1 \\ -0log(h(x)) - (1 - 0)log(1 - h(x)) = -log(1 - h(x)), & y = 0 \end{cases} \end{aligned}$$

- L'estimation d'erreur globale est, comme précédemment, la **somme des erreurs unitaires** :

$$C(W) = \frac{1}{m} \sum_{i=1}^{m} C(h(y_i), y_i)$$

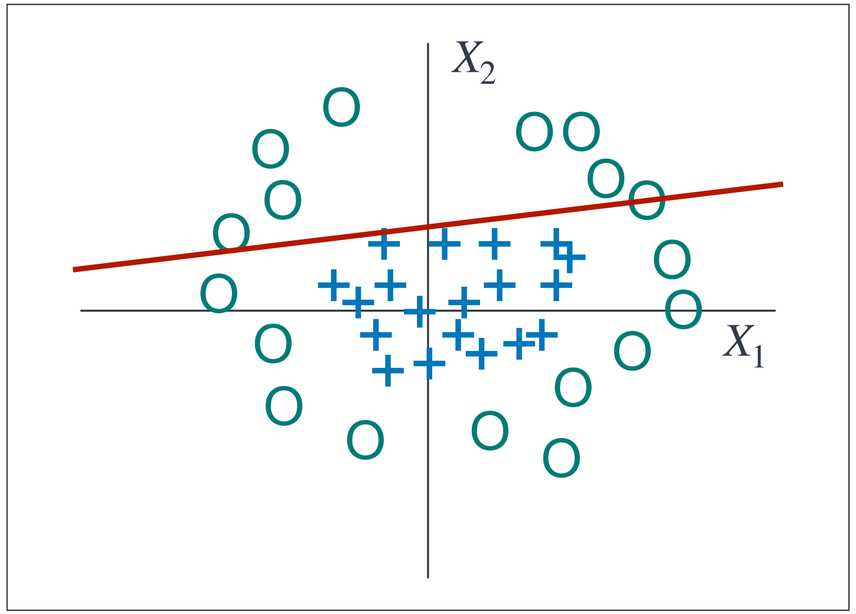
- La **minimisation** de la fonction de coût, par la **descente de gradient**, permet de trouver le meilleur vecteur de paramètres.

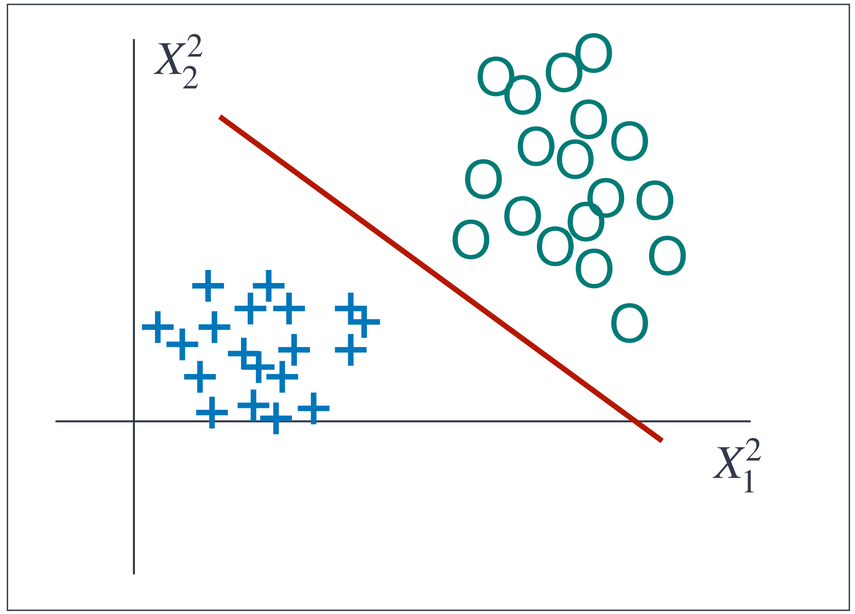
$$W_{t+1} = W_t - \alpha \nabla C_t$$

RÉGRESSION LOGISTIQUE ET LINÉARITÉ

Qu'appelle-t-on linéarité en Machine Learning ?

- On voit bien qu'**aucune droite** (plus généralement hyperplan) ne permet de **séparer correctement** les cas positifs (**+**) et les cas négatifs (**0**).
- Mais rien ne nous interdit de nous positionner dans un **autre espace de variables**. Un choix judicieux ici est d'élever au carré les variables X_1 et X_2. Dans ce nouvel espace, les observations **deviendraient linéairement séparables** !
- Même si la transformation appliquée aux variables est **non linéaire**, le modèle reste linéaire en ce sens que ce sont toujours les **mêmes poids** de la régression logistique qui s'appliqueront sur X_1^2 et X_2^2 .



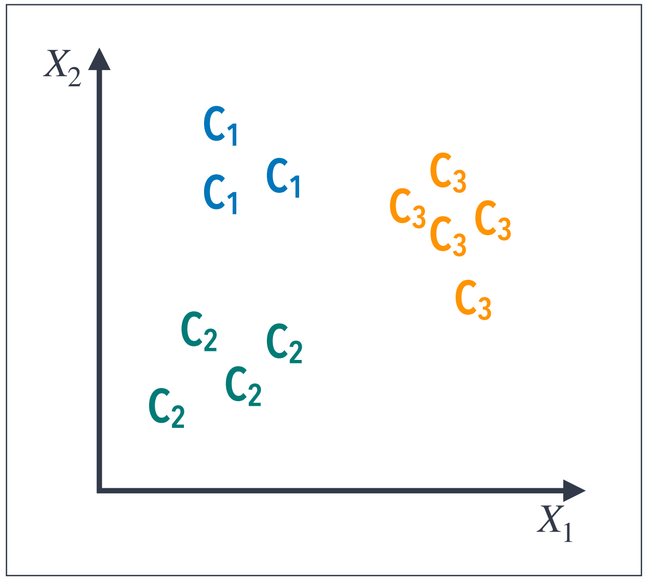


Source : Data science : fondamentaux et études de cas. Eyrolles, 2015.

RÉGRESSION LOGISTIQUE MULTICLASSES

Classification multiclasses

- La régression logistique pose une **contrainte forte** : la classification doit être supervisée par un vecteur contenant des **valeurs binaires** : oui/non, sain/malade, blanc/noir, etc.
- Cette contrainte peut **être levée**, puisqu'il existe une variante de la régression logistique, dite **multiclasses**, qui permet d'entraîner un modèle à partir d'un vecteur Y à i modalités (ou classes), avec $i > 2$.
- Dès lors, la variante de la régression logistique doit permettre de détecter les **trois classes** de l'exemple ci-après.



Classification multiclasses ($i = 3$)

- Pour cela, l'algorithme le plus utilisé en machine learning[1] est l'algorithme **one-versus-all** (ou one-versus-rest).
- Son principe est simple : il consiste à transformer le problème multiclasses en plusieurs **problèmes binaires**.
- Pour cela, il attribue un **index de classe** allant de 1 à i à chaque modalité pouvant être prise par Y.
- Prenons l'exemple de la figure ci-contre : C_1 va correspondre à la classe 1, C_2 à la classe 2 et C_3 à la classe 3.

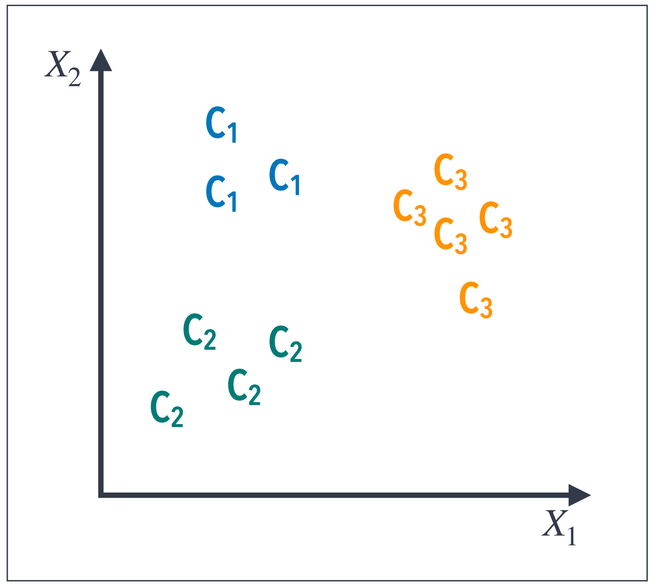
1. En statistique classique, on parlera plutôt de régression logistique polytomique et on emploiera des méthodes de modélisation un peu différentes.

Source : Data science : fondamentaux et études de cas. Eyrolles, 2015.

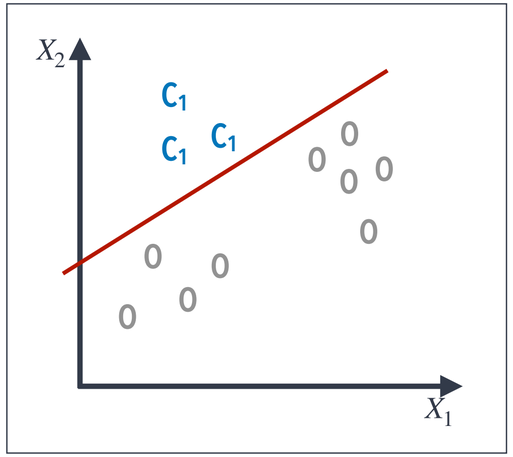
RÉGRESSION LOGISTIQUE MULTICLASSES

Classification multiclasses

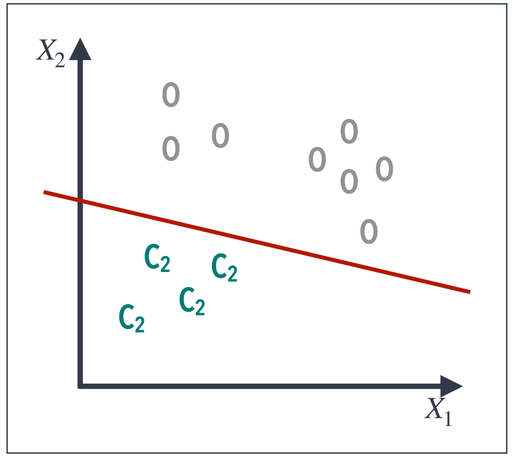
- **Étape 1** : on considère que les **C_1** représentent des cas positifs et **C_2** et **C_3** des cas négatifs. On entraîne un classifieur $h^1(x)$ pour cette représentation des données, qui saura reconnaître tous les **C_1** de tout ce qui n'est pas **C_1**.
- **Étape 2** : les **C_2** deviennent les cas positifs, **C_1** et **C_3** des cas négatifs. On entraîne un classifieur $h^2(x)$ qui sera spécialisé dans la reconnaissance des **C_2** par rapport à tout ce qui n'est pas **C_2**.
- **Étape 3** : même principe avec **C_3** pour cas positifs, et **C_1** et **C_2** pour cas négatifs, afin d'entraîner $h^3(x)$, expert en reconnaissance des **C_3**.



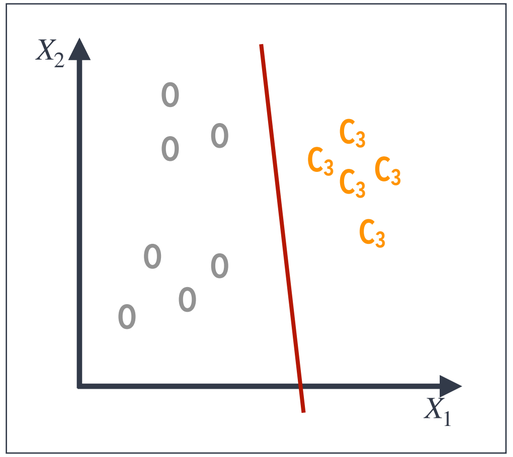
Classification multiclasses ($i = 3$)



$h^1(x)$



$h^2(x)$



$h^3(x)$

Source : Data science : fondamentaux et études de cas. Eyrolles, 2015.

RÉGRESSION LOGISTIQUE MULTICLASSES

Classification multiclasses

- Dans le cas général, on va donc **entraîner un classifieur classique** de régression logistique $h^i(x)$ pour chacune des i valeurs pouvant être prise par le vecteur Y, afin de prédire la probabilité que $Y = i$.

- On peut écrire :

$$0 \leq h^i(x) \leq 1$$
$$h^i(x) = P(y = i \,|\, x, w)(i = 1, \ \ldots, I)$$

- Ensuite, pour utiliser ce modèle afin d'effectuer des prédictions, on applique chacun des i **classifieurs aux nouvelles observations**.
- On obtiendra ainsi i **probabilités de classification**.
- La **probabilité la plus élevée** indiquera la classe à laquelle la nouvelle observation est la plus susceptible d'appartenir.

- Autrement dit, on cherche :

$$\max_a h^i(x)$$

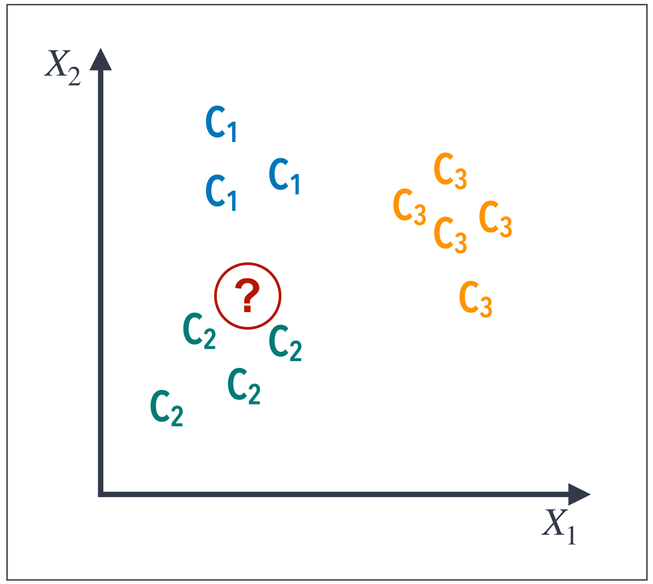
Source : Data science : fondamentaux et études de cas. Eyrolles, 2015.

RÉGRESSION LOGISTIQUE MULTICLASSES

Classification multiclasses

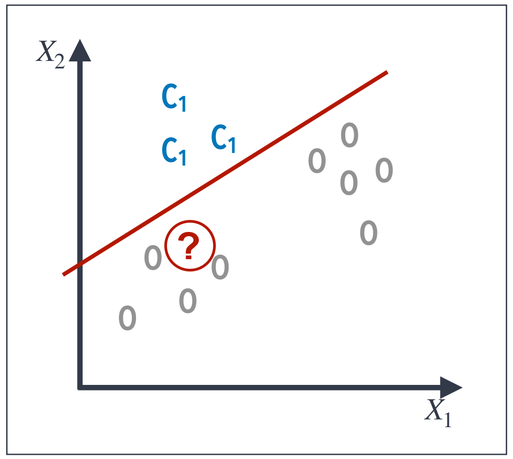
- Reprenons notre exemple constitué de **trois classes** afin d'effectuer une prédiction pour une **nouvelle observation** ; ici représentée par « ? ».
- $h^1(x)$ indique que la probabilité que l'observation soit un **C_1** est de 0.20[1].
- $h^2(x)$ indique que la probabilité que l'observation soit un **C_2** est de 0.75.
- $h^3(x)$ indique que la probabilité que l'observation soit un **C_3** est de 0.05.

La nouvelle observation est vraisemblablement[2] un **C_2**.

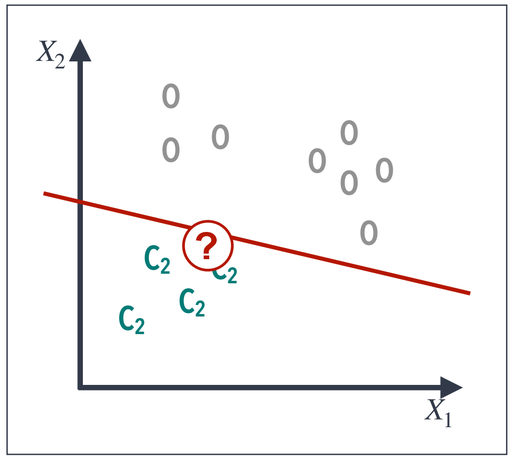


Classification multiclasses ($i = 3$)

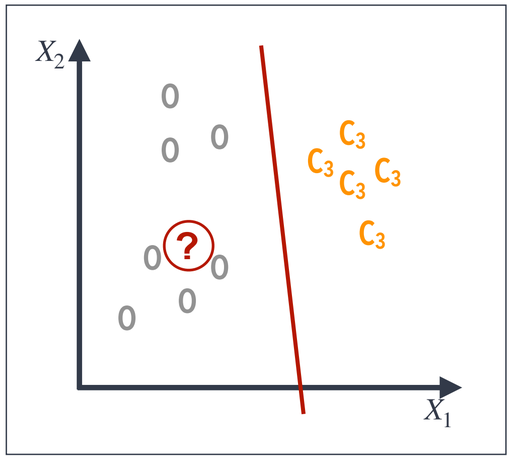
$h^1(x) = 0.20$



$h^2(x) = 0.75$



$h^3(x) = 0.05$



1. On ne se préoccupe pas de savoir si c'est un C2 ou un C3

2. La somme des probabilités des différents classements doit être égale à 1 pour que la sortie de la régression logistique soit bien une loi de probabilité.

Source : Data science : fondamentaux et études de cas. Eyrolles, 2015.

RÉGRESSION LOGISTIQUE MULTICLASSES

Alternative au classificateur one-versus-all : classificateur multinomial

- Le classificateur multinomial est une alternative au classificateur **one-versus-all** (ou one-versus-rest) que nous venons de présenter.
- Le classificateur multinomial ne classe pas chaque classe séparément, mais il utilise la fonction **softmax** pour prédire si un point de données unique appartient à l'une des classes N.
- La plupart du temps, on ne voit pas de **différence significative** dans les résultats, mais l'un des **avantages** du classificateur multinomial est qu'il modélise une **distribution entière** plutôt que les distributions individuelles $P(y = i \mid x, w)$.

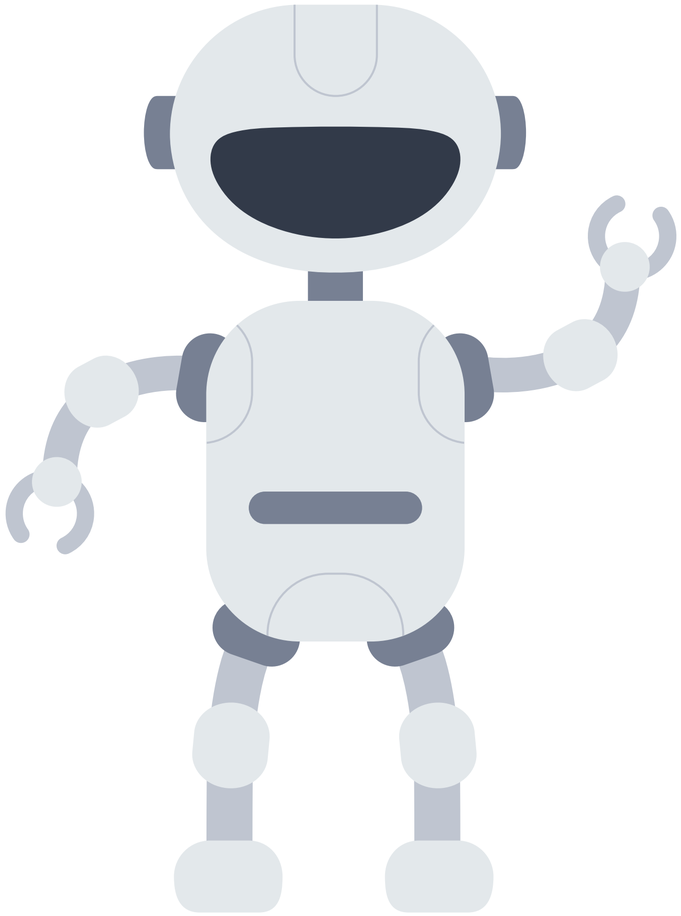
$$0 \leq h^i(x) \leq 1$$
$$h^i(x) = P(y = i \mid x, w)(i = 1, \ \ldots, I)$$

one-versus-all

$$h(x) = P(y = i \mid x, w) = softmax(x, w)$$

Multinomial

SOMMAIRE



RÉGRESSION LOGISTIQUE • ÉTAPES

ÉTAPES DE CONSTRUCTION D'UN MODÈLE

01 ***Construction du modèle***

Création d'une instance de la classe de définition du modèle.

$$estimateur = ClasseEstimateur(\overbrace{param_1, param_2, \ldots}^{\text{hyperparamètres}})$$

02 ***Entrainement du modèle***

Ajuste les variables du modèle à partir des données d'entrainement

$$estimateur\,.fit(X\_train, y\_train)$$

Modification des hyperparamètes

03 ***Prédiction du modèle***

Réalise les prédictions sur les données d'entrainement

$$estimateur\,.predict(X\_train)$$

04 ***Précision du modèle***

Détermine la précision du modèle à partir des données de test

$$estimateur\,.score(X\_test, y\_test)$$

RÉGRESSION LOGISTIQUE • MISE EN ŒUVRE

PRINCIPAUX HYPERPARAMÈTRES

- **penalty** : la méthode de régularisation. Les valeurs possibles sont : [l1, l2, elasticnet, none].
  Attention, les solveurs ne supportent pas les mêmes régularisations.

- **C** : module la force de la régularisation. Plus la valeur est petite et plus la régularisation est importante. C'est une valeur positive. L'intervalle des valeurs suit généralement une progression logarithmique : [.001, .01, .1, 1, 10, 100, 1000].

- **multi_class** : pour les classifications qui ne sont pas simplement binaires. Les valeurs possibles sont : [ovr, multinomial]

- **max_iter** : le nombre d'itérations.
  Par exemple : [100, 1000, 2500, 5000]

- **n_jobs** : précise le nombre de processeurs (disponibles) pour une exécution en parallèle.

```
LogisticRegression(
      C=1.0,
      max_iter=100,
      multi_class='auto',
      n_jobs=None,
      penalty='l2'
)
```

MERCI DE VOTRE ATTENTION

Q & R