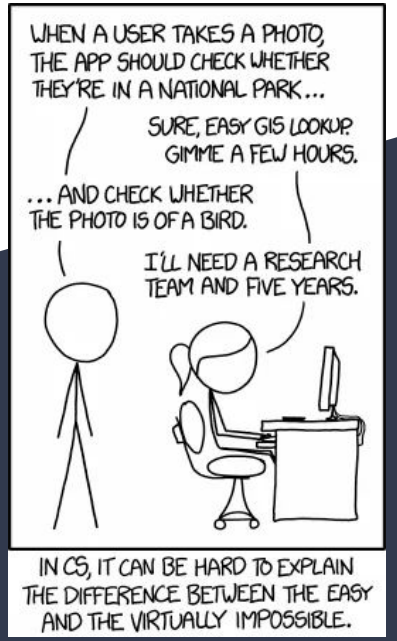
DATA 2

Apprentissage supervisé et classification

Alexandre Bruckert
2023

Nantes Université
Master Cultures Numériques
Semestre 2



alexandre.bruckert@univ-nantes.fr
https://abruckert.github.io

Apprentissage supervisé

On a :

- des données X
- des labels Y
- une fonction (ou un modèle) *paramétrique* $f_\Theta : X \to \hat{Y}$
- une méthode d'évaluation $\mathcal{L}\left(\hat{Y}, Y\right)$

On cherche :

$$\arg\min_{\Theta} \mathcal{L}\left(f_\Theta(X), Y\right)$$

Apprentissage supervisé

On a :

- des données X
- des labels Y
- une fonction (ou un modèle) *paramétrique* $f_\Theta : X \to \hat{Y}$
- une méthode d'évaluation $\mathcal{L}\left(\hat{Y}, Y\right)$

On cherche :

$$\arg\min_{\Theta} \mathcal{L}\left(f_\Theta(X), Y\right)$$

On cherche donc à **adapter un modèle** afin que ses prédictions quand on lui entre des **données annotées** **correspondent le plus possible** aux **labels associés**.

Apprentissage supervisé : classification

On a :

- des données X
- des labels Y

Les données sont représentées par des points dans un espace, où chaque axe / coordonnée représente une caractéristique.
Les labels appartiennent à un ensemble fini, représentant des **classes**, ou **catégories**

Apprentissage supervisé : classification

On a :

- des données X
- des labels Y

Les données sont représentées par des points dans un espace, où chaque axe / coordonnée représente une caractéristique.
Les labels appartiennent à un ensemble fini, représentant des **classes**, ou **catégories**

Pour chaque point de donnée de l'ensemble X, on connait sa classe

Apprentissage supervisé : classification

On a :

- des données X
- des labels Y

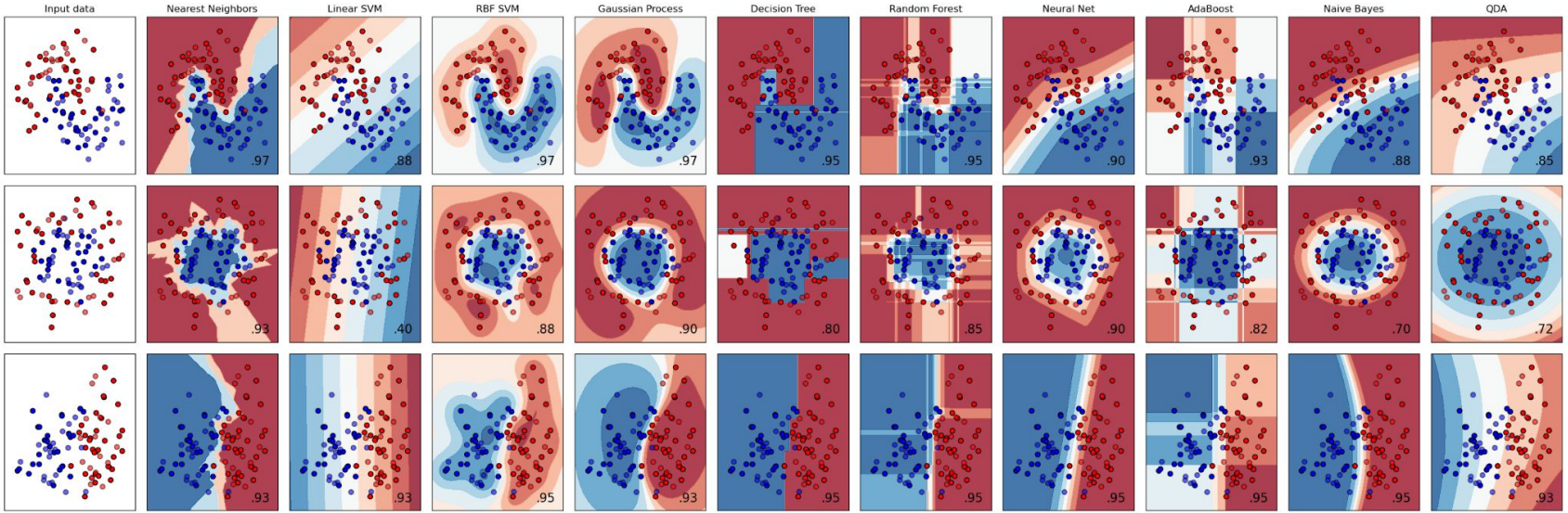
Les données sont représentées par des points dans un espace, où chaque axe / coordonnée représente une caractéristique.
Les labels appartiennent à un ensemble fini, représentant des **classes**, ou **catégories**

Pour chaque point de donnée de l'ensemble X, on connait sa classe

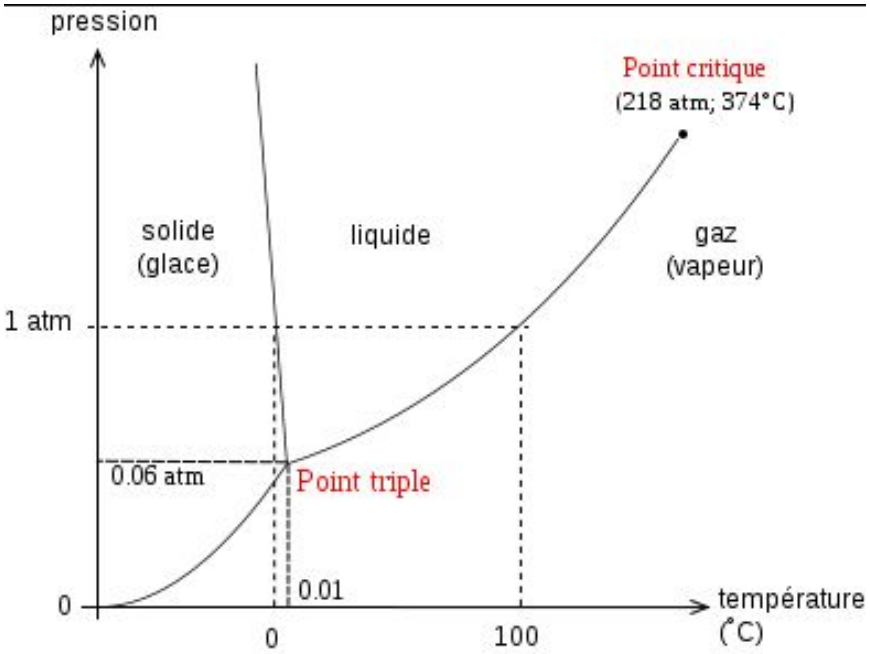
On va donc adapter les paramètres d'un modèle, appelé **classifieur**, afin qu'il prédise la bonne classe pour tous les points (ou au moins un maximum).

Apprentissage supervisé : classification

Concrètement, le modèle va séparer l'espace des données en plusieurs zones, de telle sorte que tous les points dans une même zone appartiennent à une même classe.



Exemples de classification

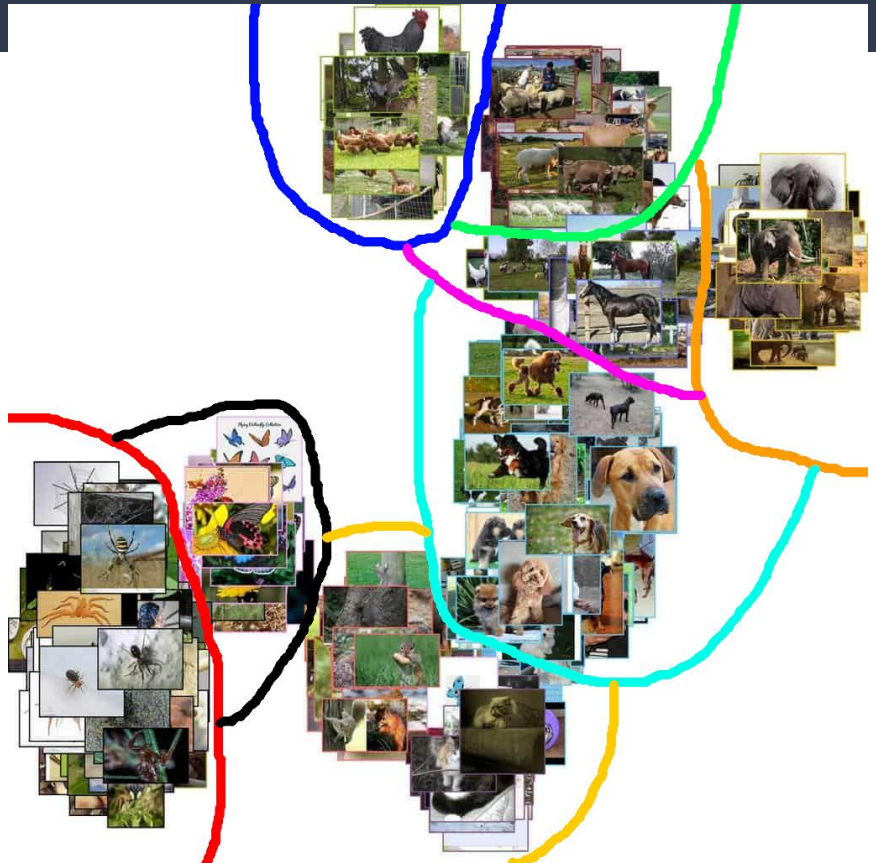


Phase de l'eau en fonction de la température et de la pression

Exemples de classification

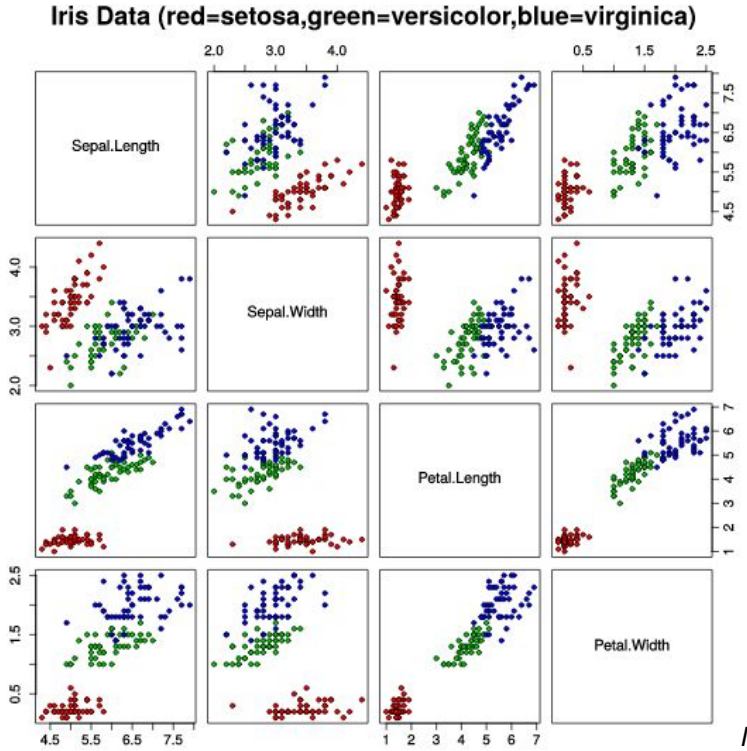
Représentation t-sne d'images d'animaux dans un espace d'images

Exemples de classification



Représentation t-sne d'images d'animaux dans un espace d'images

Exemples de classification



Base IRIS :
3 classes
4 caractéristiques

Nicoguaro, CC BY 4.0, via Wikimedia Commons

Exemples de classification

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Base Titanic :

- 2 classes (survivant ou non) : **classification binaire**
- 10 caractéristiques

https://www.kaggle.com/code/startupsci/titanic-data-science-solutions

Complexité vs généralisation

Apprentissage supervisé : on utilise un ensemble d'exemples pour créer notre modèle.

Problème : **cet ensemble d'exemples est fini !**

Complexité vs généralisation

Apprentissage supervisé : on utilise un ensemble d'exemples pour créer notre modèle.

Problème : **cet ensemble d'exemples est fini !**

De plus, la **réalité est complexe**, et cet ensemble d'exemples est **imparfait** ! (e.g. caractéristiques insuffisantes pour la tâche, bruit dans les données, etc)

Dès lors : veut-on un modèle très proche des données d'entraînement (i.e. très complexe), quitte à apprendre les défauts de ces données, ou un modèle qui généralise bien sur de nouvelles données (i.e. qui soit robuste aux défauts, quitte à perdre en précision) ?

Complexité vs généralisation

Apprentissage supervisé : on utilise un ensemble d'exemples pour créer notre modèle.

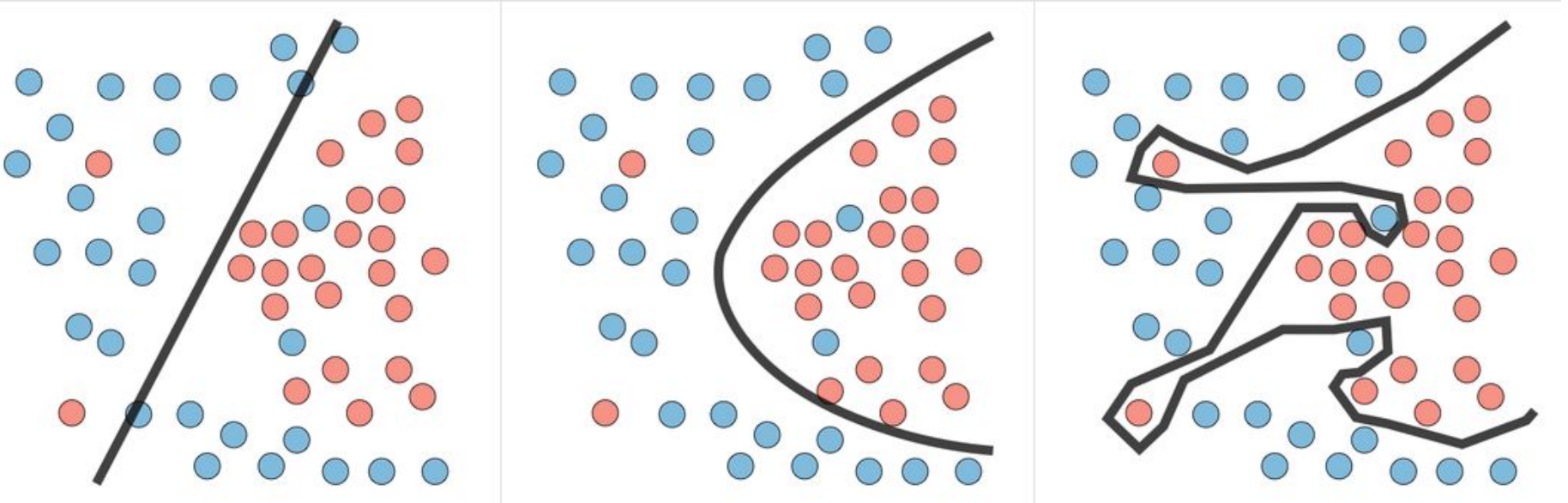
Problème : **cet ensemble d'exemples est fini !**

De plus, la **réalité est complexe**, et cet ensemble d'exemples est **imparfait** ! (e.g. caractéristiques insuffisantes pour la tâche, bruit dans les données, etc)

Dès lors : veut-on un modèle très proche des données d'entraînement (i.e. très complexe), quitte à apprendre les défauts de ces données, ou un modèle qui généralise bien sur de nouvelles données (i.e. qui soit robuste aux défauts, quitte à perdre en précision) ?

Overfitting vs Underfitting

Complexité vs généralisation



https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Evaluer un classifieur

Qu’est-ce qu’un bon classifieur ?

Evaluer un classifieur

Qu’est-ce qu’un bon classifieur ?

- Un classifieur qui fait la synthèse **biais / variance**

Evaluer un classifieur

Qu’est-ce qu’un bon classifieur ?

- Un classifieur qui fait la synthèse **biais / variance**
- Un classifieur qui associe le **label correct à de nouvelles données**

Evaluer un classifieur

Qu’est-ce qu’un bon classifieur ?

- Un classifieur qui fait la synthèse **biais / variance**
- Un classifieur qui associe le **label correct à de nouvelles données**
- Avant tout, dépendant du problème ! Toutes les classes prédites n’ont pas nécessairement la même importance

Evaluer un classifieur

Qu'est-ce qu'un bon classifieur ?

- Un classifieur qui fait la synthèse **biais / variance**
- Un classifieur qui associe le **label correct à de nouvelles données**
- Avant tout, dépendant du problème ! Toutes les classes prédites n'ont pas nécessairement la même importance

Sur quelles données évaluer mon classifieur ?

- Séparation : ensembles **d'entraînement**, de **validation** et de **test**
- Pour définir ces ensembles : holdout, K-fold cross-validation, …

Evaluer un classifieur : matrice de confusion

		Predicted class: +	**Predicted** class: -
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Métriques : classification binaire

- **Exactitude (accuracy)**: le taux d'échantillons correctement prédits par le modèle

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Précision (precision)**: le taux d'échantillons correctement prédits comme positifs par le modèle parmi tous les échantillons prédits comme positifs (i.e. "à quel point une prédiction positive est fiable")

$$Precision = \frac{TP}{TP + FP}$$

- **Taux de rappel (recall / sensitivity)**: le taux d'échantillons positifs correctement prédits comme positifs par le modèle (i.e. "à quel point un point positif est classifié comme positif")

$$Recall = \frac{TP}{TP + FN}$$

Métriques : classification binaire

- **Spécificité (specificity)**: le taux d'échantillons négatifs correctement prédits comme négatifs par le modèle (i.e. "à quel point un point négatif est classifié comme négatif")

$$Specificity = \frac{TN}{TN + FP}$$

- **Score F1** : une synthèse entre précision et taux de rappel

$$F_1 = 2\frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

Et pour plusieurs classes ?

		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Grandini et al., Metrics for multi-class classification : an overview, 2023

Métriques : classification multi-classe

- **Exactitude (accuracy)**: le taux d'échantillons correctement prédits par le modèle

$$Accuracy = \frac{aa + bb + cc + dd}{aa + ab + \ldots + dc + dd}$$

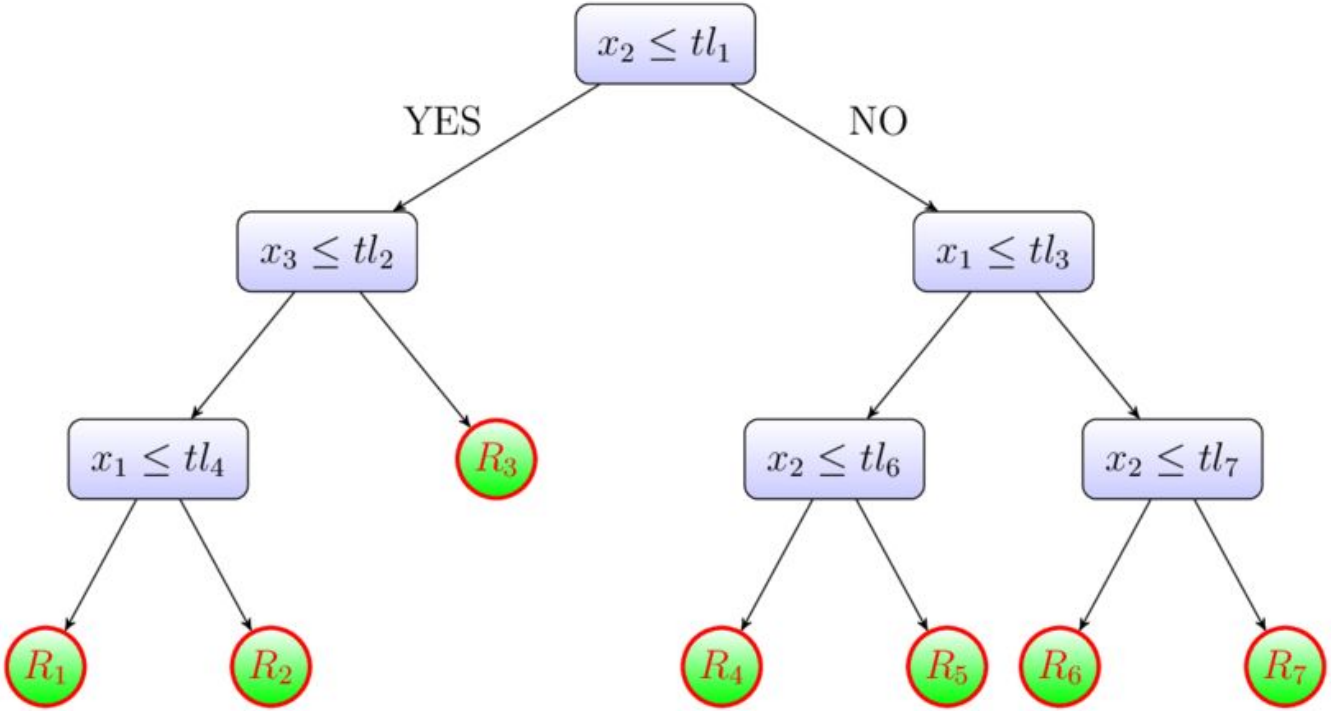
- **Précision (precision)**: le taux d'échantillons correctement prédits comme appartenant à la classe k par le modèle parmi tous les échantillons prédits comme k (i.e. "à quel point une prédiction de classe k est fiable")

$$Precision_k = \frac{TP_k}{TP_k + FP_k}$$

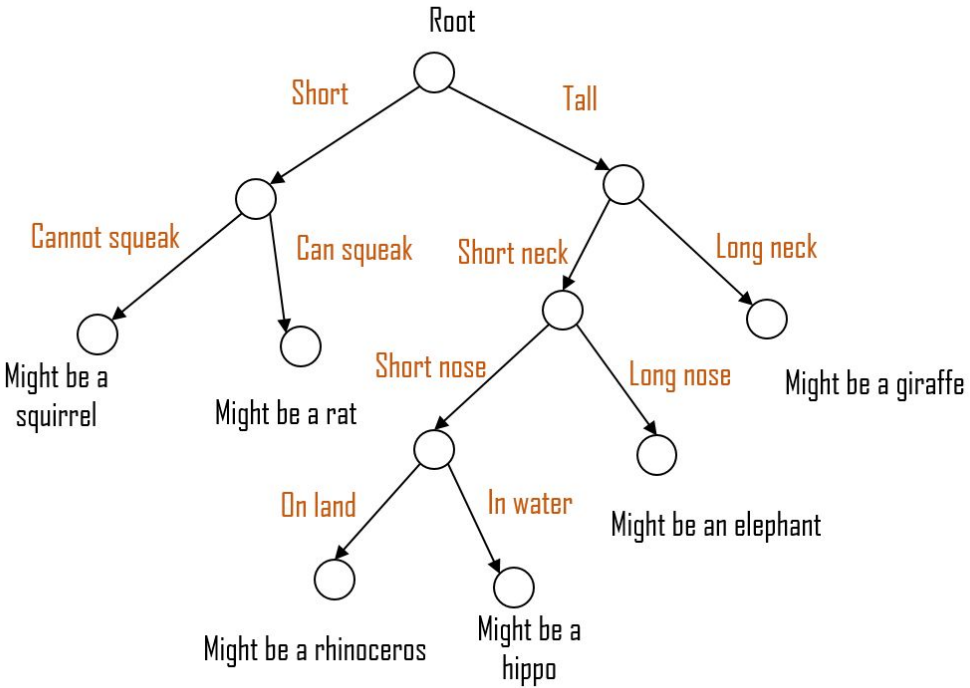
- **Taux de rappel (recall / sensitivity)**: le taux d'échantillons k correctement prédits comme k par le modèle (i.e. "à quel point un point k est classifié comme k")

$$Recall_k = \frac{TP_k}{TP_k + FN_k}$$

Un premier classifieur : l'arbre de décision



Un premier classifieur : l’arbre de décision



Un premier classifieur : l’arbre de décision

- **Principe de l’apprentissage :** on va venir séparer une partie de l’espace en deux, de telle sorte à maximiser l’information gagnée, puis on recommence jusqu’à la profondeur souhaitée.

Un premier classifieur : l'arbre de décision

- **Principe de l'apprentissage :** on va venir séparer une partie de l'espace en deux, de telle sorte à maximiser l'information gagnée, puis on recommence jusqu'à la profondeur souhaitée.

- Avantages :
  - Fonctionne avec tout type de données (continues / catégorielles)
  - Facile à interpréter
  - Très peu de pré-processing nécessaire
  - Insensible aux relations inter-caractéristiques (i.e. gère bien les variables corrélées)

Un premier classifieur : l'arbre de décision

- **Principe de l'apprentissage :** on va venir séparer une partie de l'espace en deux, de telle sorte à maximiser l'information gagnée, puis on recommence jusqu'à la profondeur souhaitée.

- Avantages :
  - Fonctionne avec tout type de données (continues / catégorielles)
  - Facile à interpréter
  - Très peu de pré-processing nécessaire
  - Insensible aux relations inter-caractéristiques (i.e. gère bien les variables corrélées)
- Inconvénients :
  - Très sensibles à l'overfitting, ne sont pas très robustes à de nouvelles données
  - Variance très élevée (i.e. de petites variations dans les données peuvent résulter en des arbres très différents)
  - Très coûteux à entraîner

Machines à vecteurs de support

- **Support Vector Machines (SVM) :** classe d'algorithmes de machine learning, utilisée à la fois pour de la régression (SVR) ou de la classification (SVC).

Machines à vecteurs de support

- **Support Vector Machines (SVM) :** classe d'algorithmes de machine learning, utilisée à la fois pour de la régression (SVR) ou de la classification (SVC).

- **Principe** : On suppose qu'il existe une frontière simple (i.e. **linéaire**) entre deux classes. On va donc chercher **l'hyperplan** qui sépare **le mieux les deux classes**, c'est à dire l'hyperplan avec **la plus grande marge**

Machines à vecteurs de support

- **Support Vector Machines (SVM) :** classe d'algorithmes de machine learning, utilisée à la fois pour de la régression (SVR) ou de la classification (SVC).

- **Principe** : On suppose qu'il existe une frontière simple (i.e. **linéaire**) entre deux classes. On va donc chercher **l'hyperplan** qui sépare **le mieux les deux classes**, c'est à dire l'hyperplan avec **la plus grande marge**

- **Hyperplan :** $h(x) = \sum_i w_i \cdot x_i + b$

- Pour un problème de classification binaire**,** si $h(x) > 0$, alors x est prédit comme étant de classe 1, et 0 dans le cas contraire.

Machines à vecteurs de support

- **Support Vector Machines (SVM) :** classe d'algorithmes de machine learning, utilisée à la fois pour de la régression (SVR) ou de la classification (SVC).

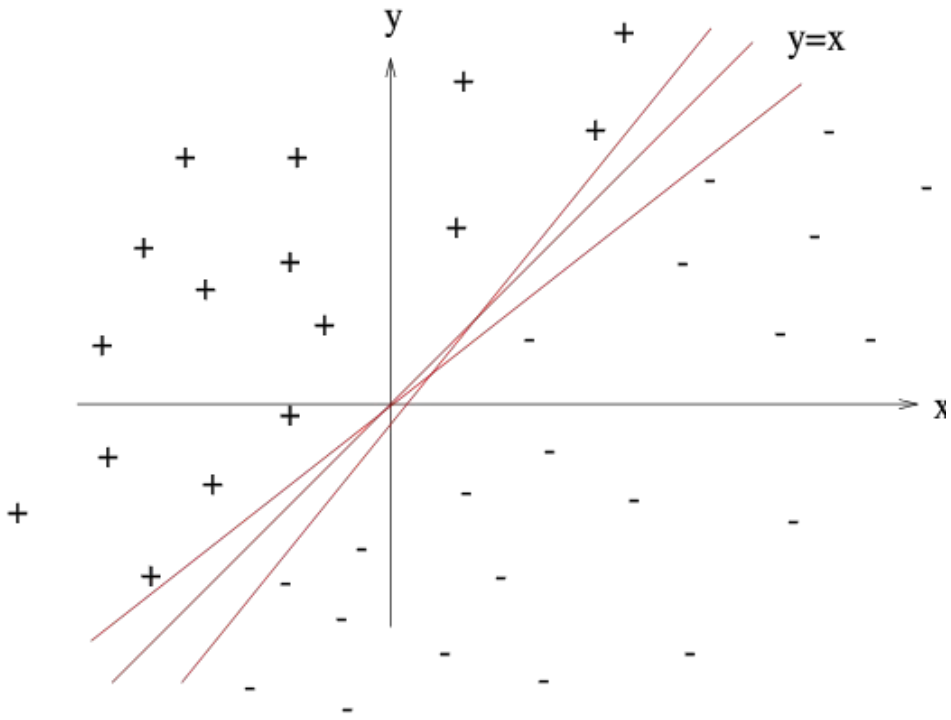
- **Principe** : On suppose qu'il existe une frontière simple (i.e. **linéaire**) entre deux classes. On va donc chercher **l'hyperplan** qui sépare **le mieux les deux classes**, c'est à dire l'hyperplan avec **la plus grande marge**

- **Hyperplan :** $h(x) = \sum_i w_i \cdot x_i + b$

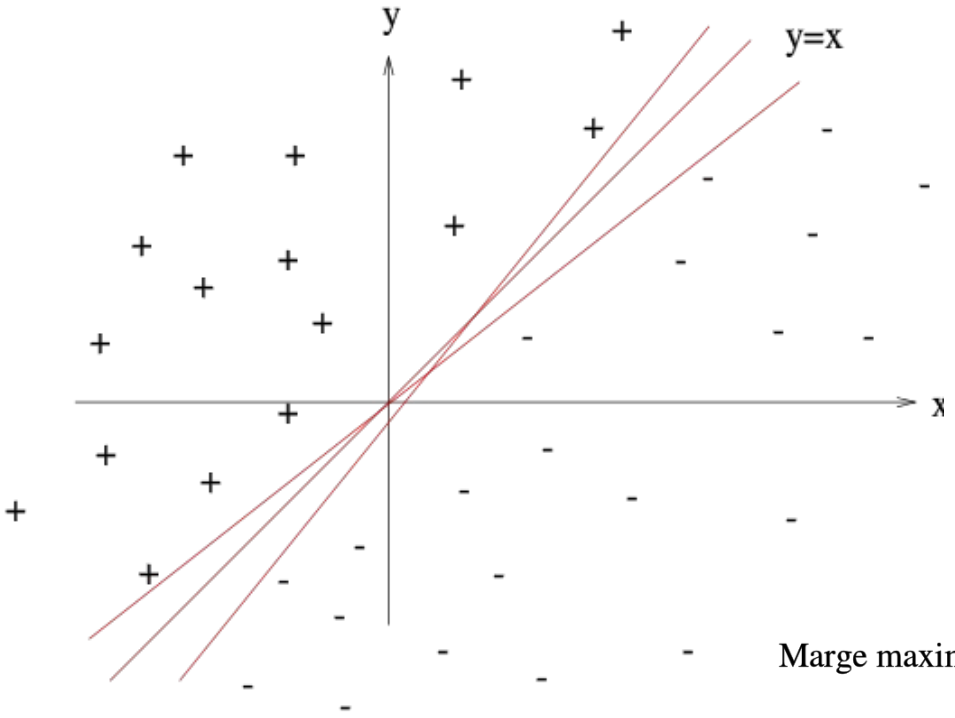
- Pour un problème de classification binaire**,** si $h(x) > 0$, alors x est prédit comme étant de classe 1, et 0 dans le cas contraire.

- La **marge** est la distance entre la frontière de séparation et les points les plus proches. On va donc résoudre un problème de **maximisation**.

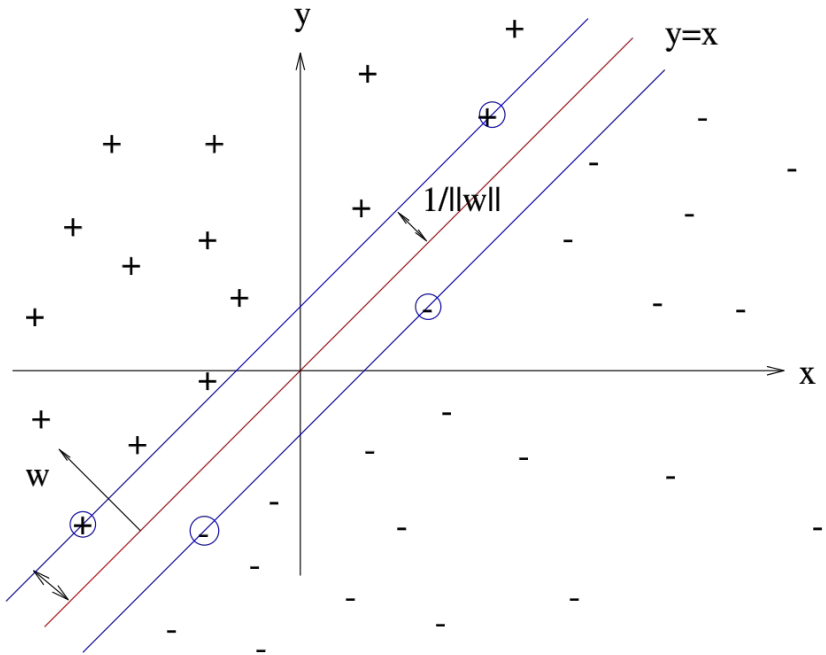
Machines à vecteurs de support



Machines à vecteurs de support

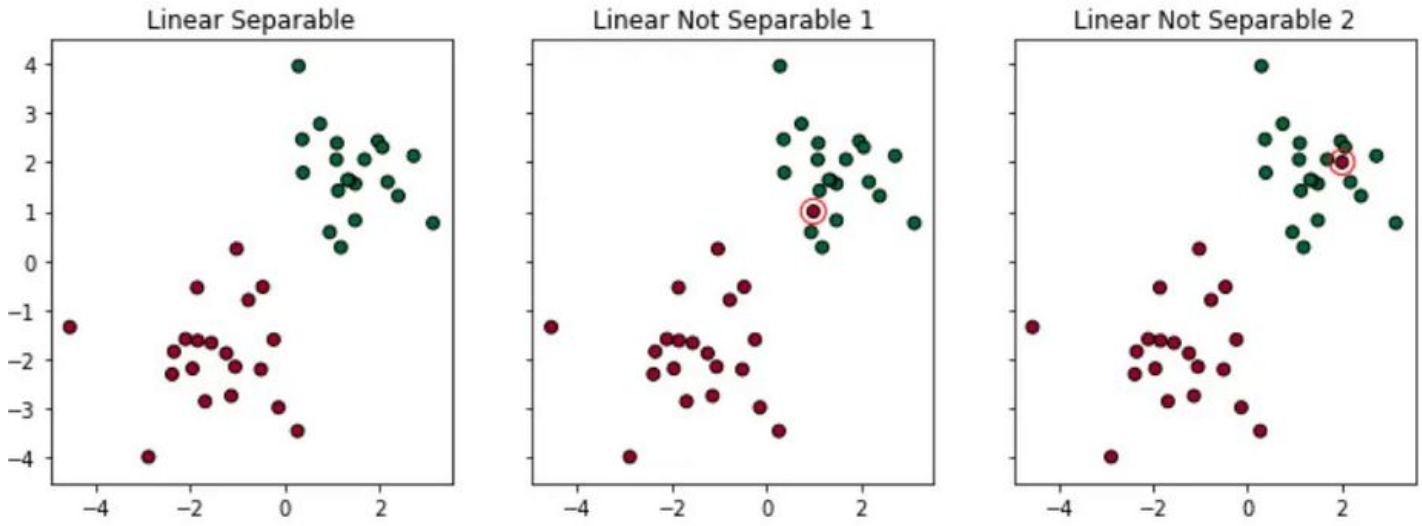


Marge maximale



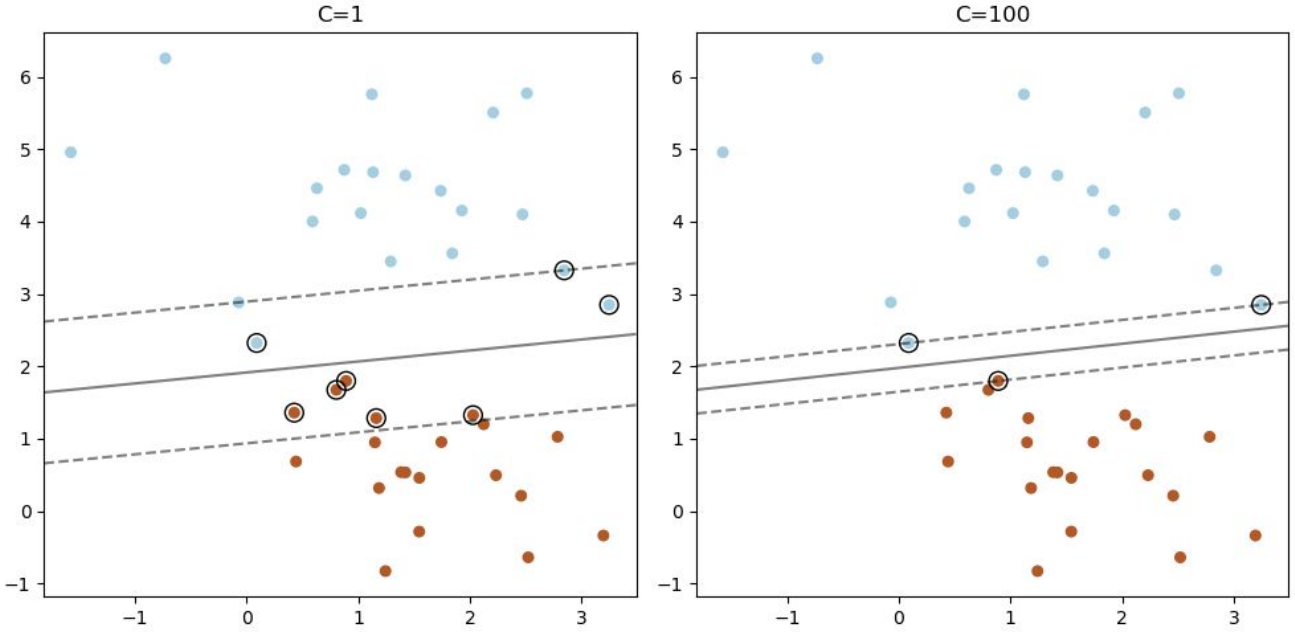
Non-séparabilité linéaire

- **Problème :** en général, les données ne sont pas séparables linéairement !



Soft margin SVM : hyperparamètre C

- **Solution 1 :** on ajoute une tolérance dans la marge (i.e. quelquespoints peuvent être mal classifiés ou dans la marge). Cette tolérance est décidée par l'hyperparamètre C.

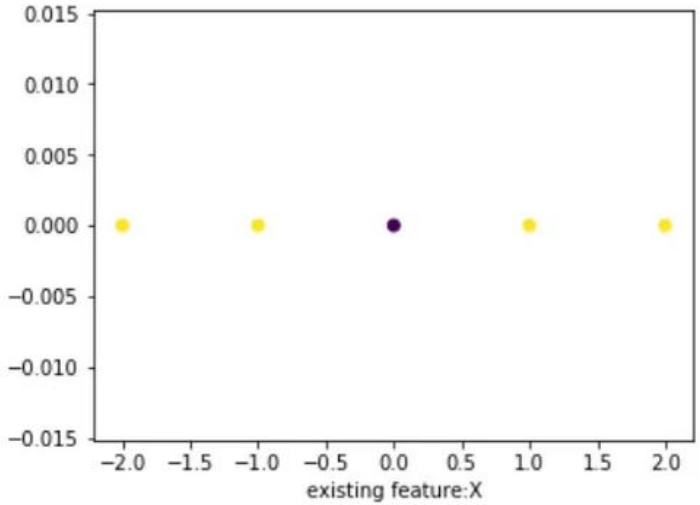


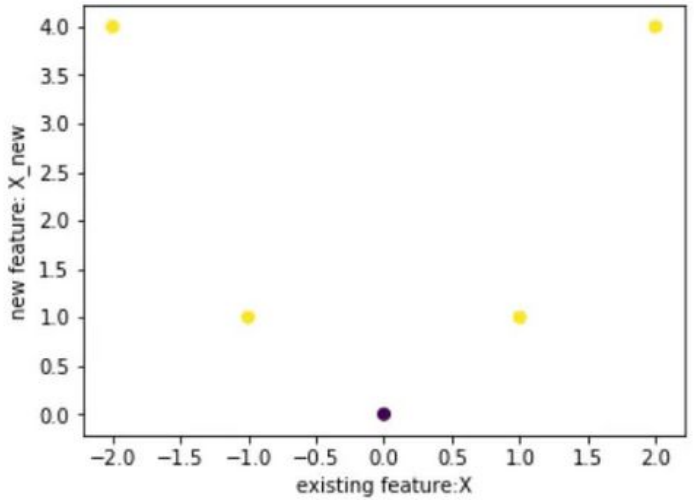
Kernel polynomial

- **Solution 2 :** on vient déformer les données en les projetant dans un espace de dimension supérieure, en créant des dimensions. Dans ces espaces, on peut (potentiellement) trouver une séparation linéaire.

Kernel polynomial

- **Solution 2 :** on vient déformer les données en les projetant dans un espace de dimension supérieure, en créant des dimensions. Dans ces espaces, on peut (potentiellement) trouver une séparation linéaire.





Kernel polynomial : hyperparamètre d

- **Solution 2 :** on vient déformer les données en les projetant dans un espace de dimension supérieure, en créant des dimensions. Dans ces espaces, on peut (potentiellement) trouver une séparation linéaire.

- Pour un **kernel polynomial**, on va créer une coordonnée artificielle $x_d = x_0^d$

- Par exemple, pour des points dans un espace à 1 dimension et un kernel de degré 2, on va associer à chaque point $x = (x_0)$ un point $x' = (x_0, x_0^2)$

Cas compliqués : kernel RBF

- **Solution 3 :** on vient déformer les données en les projetant dans un espace de dimension infinie. Dans ces espaces, on peut (potentiellement) trouver une séparation linéaire.

- Le fonctionnement est similaire à celui du kernel polynomial, et est guidé par un hyperparamètre γ