Classement ou classification

Classement

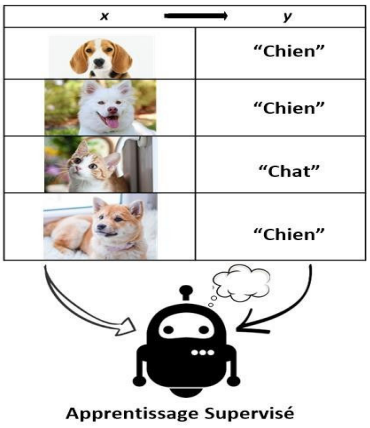
- Le classement est l'opération qui permet de placer chaque individu de la population étudiée dans une classe, parmi plusieurs classes prédéfinies, en fonctions des caractéristiques de l'individu indiquées comme variables explicatives.

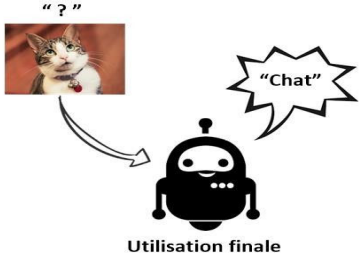
- L'affectation à une classe à partir des caractéristiques explicatives se fait généralement par une formule, un algorithme ou un ensemble de règles, qui constitue un modèle et qu'il faut découvrir.

Apprentissage supervisé : Classification

- La classification est une méthode d'apprentissage automatique supervisée dans laquelle le modèle tente de prédire l'étiquette correcte d'une donnée d'entrée.
- Dans la classification, le modèle est entièrement entrainé à l'aide des données d'apprentissage, puis il est évalué sur des données de test avant d'être utilisé pour effectuer des prédictions sur de nouvelles données inédites.

Exemple: Classification





Exemple: Classification

Le tableau présente des données de transactions avec plusieurs caractéristiques (heure, montant, longitude, latitude) et une étiquette "Fraude" qui indique si la transaction est frauduleuse (1) ou non (0). L'objectif est de classer la dernière transaction. Prédiction si une transaction est frauduleuse ou non.

	Input				Output
	Heure X1	Montant X2	Long X3	Lat X4	Fraude
Data Set	12:20	4500	-1.2	3.2	0
	01:45	1,60	0.6	4.3	1
	10:08	100	-2.45	1.3	0
	11:55	80	-1.2	3.2	1
	00:00	3200	-1	3	??????

Types de Classification

Les types de classification sont :

- **Classification binaire** (ou la classification binomiale)
- **Classification multi-classe**
- **classification multi-label**

12

Types de Classification

Classification binaire:

- La classification binaire consiste à classer les éléments d'un ensemble en deux groupes sur la base d'une règle de classification. Ou selon que l'élément possède ou non une propriété / fonctionnalité donnée.
- Elle est utilisée dans le cas où les étiquettes sont *binaires (0 ou 1)* , elles indiquent l'appartenance à une *classe*.

Par exemple:

—Identifier si un email est un spam ou non ;

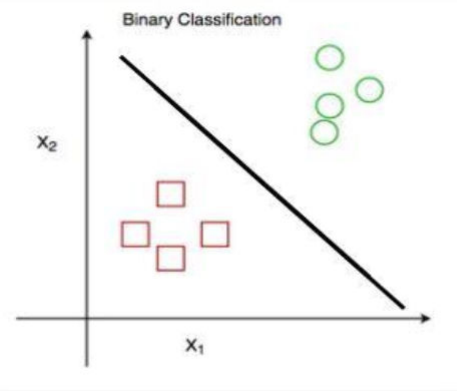
—Identifier si une image contient ou non un chat;

—Identifier si une transaction financière est frauduleuse ou non.

12

Types de Classification

Classification binaire: Exemple



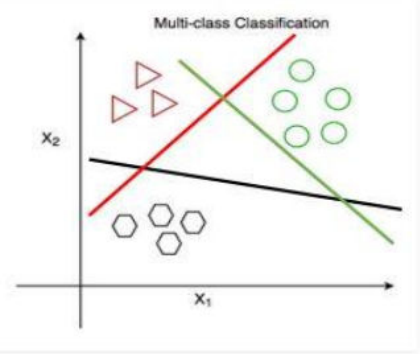
Types de Classification

Classification multi-classe

- La classification multi classes désigne une tâche de classification comportant plus de deux classes, par exemple, classer un ensemble d'images de fruits qui peuvent être des oranges, des pommes ou des poires. La classification multi classes part du principe que chaque échantillon est affecté à une et une seule étiquette : un fruit peut être soit une pomme, soit une poire, mais pas les deux en même temps
- Elle est utilisée dans le cas où les étiquettes sont *discrètes (1, 2, …C)*, et correspondent donc à plusieurs *classes*.
- Par exemple:
  - —Identifier en quelle langue un texte est écrit ;
  - —Identifier lequel des 10 chiffres arabes est un chiffre manuscrit

Types de Classification

Classification multi-classe: Exemple



Types de Classification

Classification multi-label :

Jusqu'à présent, chaque instance a toujours été affectée à une seule classe. Dans certains cas notre classificateur peut produire plusieurs classes pour chaque instance.

Par exemple, considérons un classificateur de reconnaissance de visage : que doit-il faire s'il reconnaît plusieurs personnes sur la même image ?
Bien sûr, il devrait attacher une étiquette à chaque personne qu'il reconnaît.

Disons que le classificateur a été entraîné à reconnaître trois visages, Alice, Bob et Charlie ; alors lorsqu'on lui montre une image d'Alice et de Charlie.
Il devrait produire [1, 0, 1] (ce qui signifie "Alice oui, Bob non, Charlie oui").

Un tel système de classification qui produit plusieurs binaires multiples est appelé un système de classification multilabel

Exemple: Prédire la liste des objets qui se trouvent dans une photo ou un texte.

La différence entre la classification multiclasses et la classification multi-label

- la classification multi-classes part du principe que chaque échantillon est affecté à une et une seule étiquette : un fruit peut être soit une pomme, soit une poire, mais pas les deux en même temps.
- En revanche, un exemple de classification multi-label peut être qu'un texte peut porter sur la religion, la politique, la finance ou l'éducation en même temps ou sur aucun de ces sujets

Algorithmes de classification

Pour traiter des problèmes de de classification il existe plusieurs algorithmes au niveau de ML. Ces algorithmes peut être classés en deux deux catégories il y a ce qu'on appelle :

- **Les apprenants (**ou Les algorithmes**) enthousiastes (Enthusiastic learners)**
- **Les apprenants paresseux (Lazy learners)**

Algorithmes de classification

- **Les apprenants enthousiastes (Enthusiastic learners)**
  - sont des algorithmes d'apprentissage automatique qui construisent d'abord un modèle à partir de l'ensemble de données d'apprentissage, puis faire des prédictions sur données les futurs.
  - Ils passent plus de temps au cours du processus d'entrainement en raison de leur volonté d'obtenir une meilleure généralisation, mais ils ont besoin de moins de temps pour faire des prédictions.
  - La plupart des algorithmes d'apprentissage automatique sont des apprenants enthousiastes, dont voici quelques exemples :
    - **Régression logistique.**
    - **Machine à vecteur de support.**
    - **Arbres de décision.**
    - **Réseaux neuronaux artificiels.**

Algorithmes de classification

- **Les apprenants paresseux (Lazy learners)**
  - Les **apprenants paresseux** ou les apprenants basés sur les instances, en revanche, ne créent pas de modèle immédiatement à partir des données d'apprentissage, et c'est de là que vient l'aspect paresseux.
  - Ils se contentent de mémoriser les données d'apprentissage (Rapide)
  - Puis à chaque fois qu'il est nécessaire de faire une prédiction, ils recherchent le plus proche voisin à partir de l'ensemble des données d'apprentissage, ce qui les rend **très lents lors de la prédiction**.
  - Exemple :
    - **KNN (K-Nearest Neighbors) : K Voisins les plus proches**

Régression logistique

Définition

- La régression logistique est une méthode statistique utilisée pour modéliser la relation entre une variable dépendante binaire (ayant deux valeurs possibles, par exemple 0 ou 1, oui ou non) et un ensemble de variables indépendantes. Elle est particulièrement utile pour les problèmes de classification, où l'objectif est de prédire à quelle classe appartient une observation.
- Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

Définition

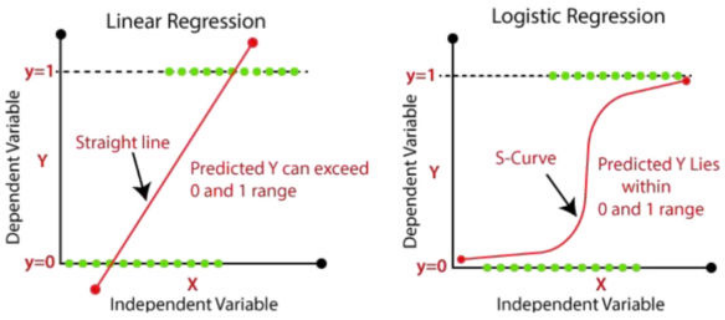
- La régression logistique est un algorithme d'apprentissage supervisé utilisé pour les problèmes de classification binaire, c'est-à-dire lorsque la variable dépendante est catégorielle binaire
- Dans la régression logistique, nous utilisons la **fonction sigmoïde** pour **calculer la probabilité de la variable dépendante**.

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Exemples :
  - Prédire si un email est spam ou non
  - Prédire si un patient est malade ou non
  - Prédire si une transaction bancaire est frauduleuse ou pas

Définition

- Au lieu d'une équation linéaire directe, la régression logistique utilise une fonction logistique (ou sigmoïde) qui contraint les valeurs de prédiction entre 0 et 1, ce qui correspond à des probabilités.
- Cependant, dans la régression linéaire on cherche une droite, alors les valeurs de prédiction ne sont pas contraintes à l'intervalle [0,1]



Régression logistique

Fondement Mathématique:

Le cœur de la régression logistique est la **fonction logistique** (ou fonction sigmoïde). Cette fonction prend en entrée n'importe quelle valeur réelle et renvoie une valeur comprise entre 0 et 1, ce qui est idéal pour représenter une probabilité.

La fonction logistique est définie comme suit :

```
σ(z) = 1 / (1 + e^(-z))
```

Où :

- z est une combinaison linéaire des variables explicatives : $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$
- β_0, β_1, ... β_p sont les coefficients de régression à estimer
- $\sigma(z)$ est la probabilité estimée

Régression logistique

- **Estimation des coefficients:** Les coefficients de la régression logistique sont estimés en maximisant la vraisemblance, ce qui signifie qu'on cherche les paramètres qui rendent les données observées les plus probables.

Régression logistique

Le Modèle de Régression Logistique:

Le modèle de régression logistique relie la probabilité d'un événement à un ensemble de variables prédictives. On peut l'exprimer ainsi :

$$P(Y=1 \mid X) = \sigma(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p)$$

Où :

- $P(Y=1 \mid X)$ est la probabilité que la variable dépendante Y soit égale à 1 (succès) étant donné les valeurs des variables explicatives X.

Estimation des Coefficients

Pour estimer les coefficients β, on utilise généralement la **méthode du maximum de vraisemblance**. Cette méthode consiste à trouver les valeurs des coefficients qui maximisent la probabilité d'observer les données réelles, étant donné le modèle.

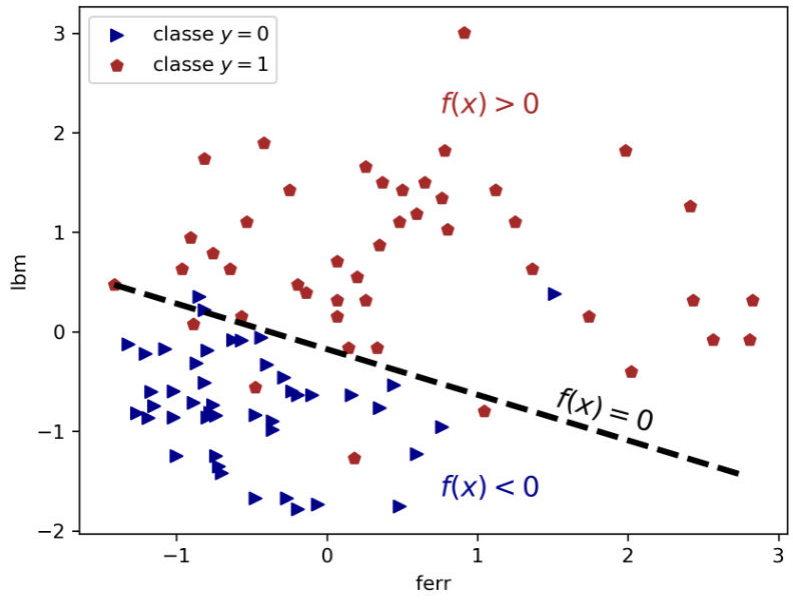
En bref

- La régression logistique est un **cas particulier d'analyse de régression** et est utilisée lorsque la **variable dépendante est nominalement échelonnée (qualitative nominale)**. C'est le cas, par exemple, de la variable décision d'achat avec les deux valeurs "achète un produit" et "n'achète pas de produit".
- Un modèle de régression logistique permet de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1.
- Lorsque la valeur prédite est supérieure à 0,5, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure à 0,5, il ne l'est pas.

En bref

Dans la régression logistique, on cherche a trouver une fonction ***f(x).*** Tel que:

- Si **f(X)> seuil** alors **Y=1**
- Si **f(X)< seuil** alors **Y=0**



Régression logistique

Utilisation de Régression Logistique:

Quand utiliser la régression logistique ?

- **Classification binaire:** Prédire si un individu aura une maladie ou non, si un client va acheter un produit ou non, etc.
- **Modélisation de la probabilité:** Estimer la probabilité qu'un événement se produise.
- **Variables indépendantes continues ou catégorielles:** La régression logistique peut gérer à la fois des variables indépendantes continues et catégorielles.

Exemple : Prédiction de l'admission à l'université

Imaginez que vous souhaitez prédire si un étudiant sera admis à l'université en fonction de ses notes au baccalauréat et de son score au test d'admission. La variable dépendante (admission) est binaire (0 : non admis, 1 : admis). La régression logistique vous permettra de modéliser la probabilité d'admission en fonction des notes et du score.

RÉGRESSION LOGISTIQUE

Le Dataset contiendra surement les colonnes suivantes:

- **Admission:** Variable cible (0 ou 1).
- **Notes au baccalauréat:** Note obtenue au baccalauréat.
- **Score au test d'admission:** Score obtenu à un test standardisé (ex : SAT, GRE).
- **Rang de classe:** Position de l'étudiant dans sa classe.
- **Nombre d'activités extra-scolaires:** Nombre d'activités auxquelles l'étudiant a participé.
- **Recommandation des enseignants:** Note attribuée par les enseignants (sur une échelle)

Équation: La régression logistique utilise une équation de la forme :

$$P(\text{Admission} = 1) = 1 / (1 + \exp(-(\beta_0 + \beta_1 * \text{Notes} + \beta_2 * \text{Score} + \ldots)))$$

RÉGRESSION LOGISTIQUE

)ù :

$P(\text{Admission} = 1)$ est la probabilité d'admission.

β_0, β_1, β_2, ... sont les coefficients du modèle à estimer.

Notes, Score, ... sont les variables prédictives.

nterprétation des coefficients:

Un coefficient positif indique qu'une augmentation de la variable correspondante augmente la probabilité d'admission.

Un coefficient négatif indique qu'une augmentation de la variable correspondante diminue la probabilité d'admission.

euil de décision: On fixe généralement un seuil de 0.5. Si la probabilité prédite est upérieure à 0.5, on considère que l'étudiant sera admis.

RÉGRESSION LOGISTIQUE

Processus:

1. **Collecte des données:** Les données peuvent être obtenues auprès de l'université ou d'autres sources.
2. **Prétraitement des données:** Nettoyer les données, gérer les valeurs manquantes, encoder les variables catégorielles (si nécessaire).
3. **Séparation des données:** Diviser les données en un ensemble d'entraînement et un ensemble de test.
4. **Entraînement du modèle:** Utiliser l'ensemble d'entraînement pour estimer les coefficients du modèle.
5. **Évaluation du modèle:** Évaluer la performance du modèle sur l'ensemble de test à l'aide de métriques comme l'accuracy, la précision, le rappel, la F1-score, la matrice de confusion, etc.
6. **Interprétation des résultats:** Analyser les coefficients du modèle pour comprendre l'importance de chaque variable prédictive et faire des prédictions.

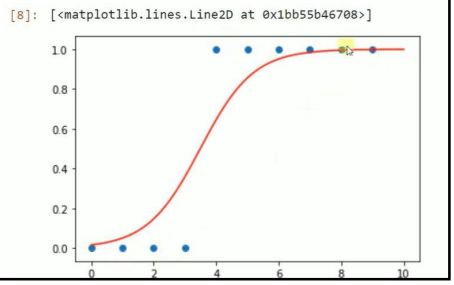
RÉGRESSION LOGISTIQUE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


# Charger les données
data = pd.read_csv("data_admission.csv")

# Séparer les variables
X = data[['notes', 'score_test']]
y = data['admission']

# Division en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# Créer
 et entraîner le modèle
model = LogisticRegression()
model.fit(X_train, y_train)

# Faire des prédictions
y_pred = model.predict(X_test)

# Évaluer le modèle
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

RÉGRESSION LOGISTIQUE

Interprétation des résultats :

- **Accuracy:** Indique le pourcentage de prédictions correctes.
- **Coefficients du modèle:** Les coefficients associés à chaque variable prédictive indiquent l'impact de cette variable sur la probabilité d'admission. Un coefficient positif signifie qu'une augmentation de cette variable augmente la probabilité d'admission.



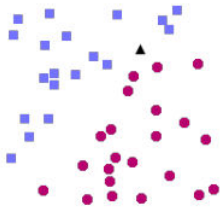
SVM

SVM

- SVM est l'acronyme de *Support Vector Machines*, soit *machines à vecteurs support* en français, parfois traduit par *séparateur à vaste marge*
- SVM est un algorithme d'apprentissage supervisé.

Problème de classification, SVM : C'est quoi?

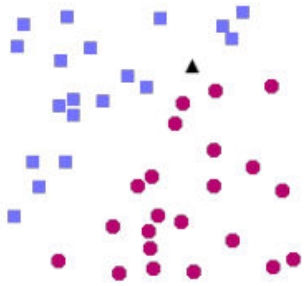
Considérons l'exemple suivant. On se place dans le plan, et l'on dispose de deux catégories: les ronds rouges et les carrés bleus, chacune occupant une région différente du plan. Cependant, la frontière entre ces deux régions n'est pas connue. Ce que l'on veut, c'est que quand on lui présentera un nouveau point dont on ne connaît que la position dans le plan, l'algorithme de classification sera capable de prédire si ce nouveau point est un carré rouge ou un rond bleu.



Problème de classification, SVM : C'est quoi?

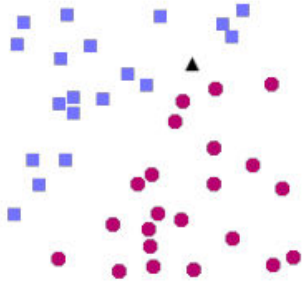
Voici notre **problème de classification**: pour chaque nouvelle entrée, être capable de déterminer à quelle catégorie cette entrée appartient.

Autrement dit, il faut être capable de trouver la frontière entre les différentes catégories. Si on connaît la frontière, savoir de quel côté de la frontière appartient le point, et donc à quelle catégorie il appartient.



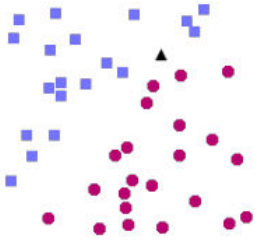
Problème de classification, SVM : C'est quoi?

Le SVM est une solution à ce problème de classification. Le SVM appartient à la catégorie des *classificateurs linéaires* (qui utilisent une séparation linéaire des données), et qui dispose de sa méthode à lui pour trouver la frontière entre les catégories.



Problème de classification, SVM : C'est quoi?

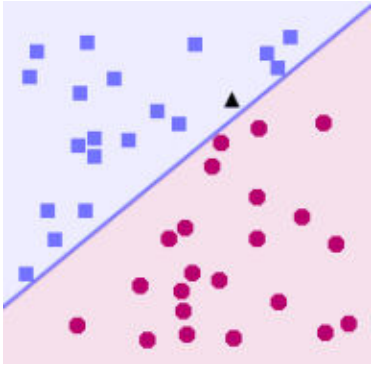
Pour que le SVM puisse trouver cette frontière, il est nécessaire de lui donner des **données d'entraînement**. En l'occurrence, on donne au SVM un ensemble de points, dont on sait déjà si ce sont des carrés rouges ou des ronds bleus, comme dans la Figure ci-dessous. A partir de ces données, le SVM va estimer l'emplacement le plus plausible de la frontière



Problème de classification, SVM : C'est quoi?

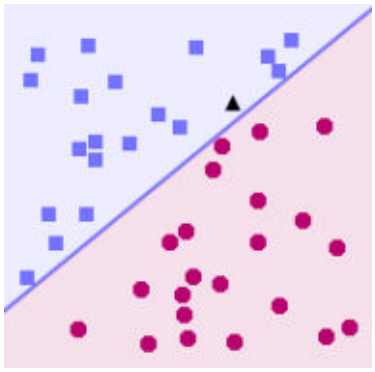
- Une fois la phase d'entraînement terminée, le SVM a ainsi trouvé, à partir de données d'entraînement, l'emplacement supposé de la frontière. En quelque sorte, il a «appris» l'emplacement de la frontière grâce aux données d'entraînement.

- le SVM est maintenant capable de prédire à quelle catégorie appartient une entrée qu'il n'avait jamais vue avant, et sans intervention humaine (comme c'est le cas avec le triangle noir dans la Figure ci-dessous): c'est là tout l'intérêt de l'apprentissage automatique.



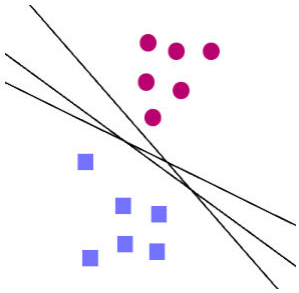
Problème de classification, SVM : C'est quoi?

- Comme vous pouvez le constater dans la figure ci-dessus, pour notre problème le SVM a choisi une ligne droite comme frontière. C'est parce que, comme on l'a dit, le SVM est un classificateur *linéaire*.



Les SVM, dans les grandes lignes

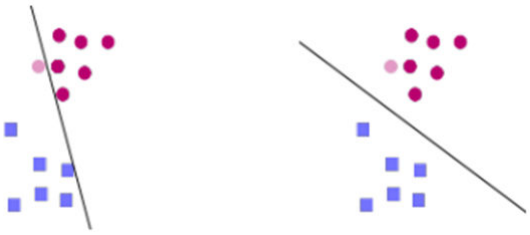
Nous savons donc que le but, pour un SVM, est d'apprendre à bien placer la frontière entre deux catégories. Mais comment faire? Quand on a un ensemble de points d'entraînement, il existe plusieurs lignes droites qui peuvent séparer nos catégories. La plupart du temps, il y en a une infinité… Alors, laquelle choisir?



Les SVM, dans les grandes lignes

Intuitivement, on se dit que si on nous donne un nouveau point, très proche des ronds rouges, alors ce point a de fortes chances d'être un rond rouge lui aussi.

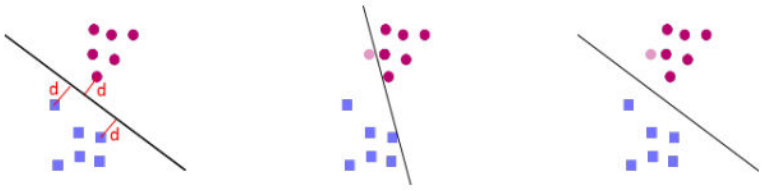
Inversement, plus un point est près des carrés bleus, plus il a de chances d'être lui-même un carré bleu.



Les SVM, dans les grandes lignes

Pour cette raison, un SVM va placer la frontière aussi loin que possible des carrés bleus, mais également aussi loin que possible des ronds rouges.

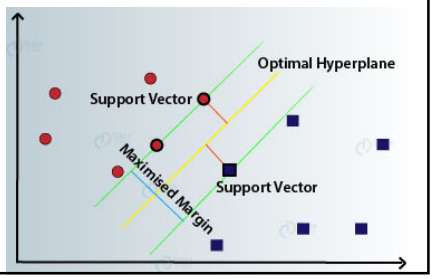
Comme on le voit dans la figure à gauche, c'est bien la frontière la plus éloignée de tous les points d'entraînement qui est optimale, on dit qu'elle a la meilleure capacité de généralisation. Ainsi, le but d'un SVM est de trouver cette frontière optimale, en maximisant la distance entre les points d'entraînement et la frontière.



Les SVM, dans les grandes lignes

- Les points d'entraînement les plus proches de la frontière sont appelés **vecteurs support**.
- Le modèle SVM tente d'élargir la distance entre les deux classes en créant une frontière de décision (hyperplan) bien définie.

Dans le cas ci-contre, notre hyperplan a divisé les données. Alors que nos données étaient en 2 dimensions, l'hyperplan était de 1 dimension.

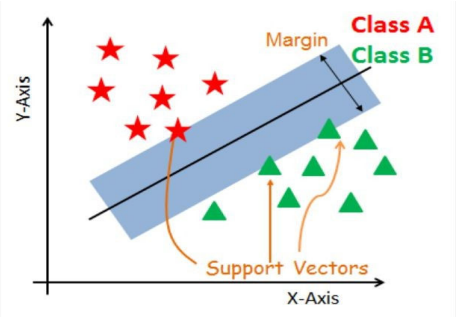


À RETENIR

- **Machine à Vecteurs de Support,** est un algorithme d'apprentissage supervisé. Son but principal est de classer des données en différentes catégories. Imaginez que vous ayez des points de différentes couleurs sur un graphique. Une SVM va essayer de trouver la meilleure ligne (ou hyperplan en dimension supérieure) pour séparer ces points de couleurs différentes.
- **Hyperplan:** C'est la ligne (ou le plan) qui sépare au mieux les données en différentes classes.
- **Marge:** C'est la distance entre l'hyperplan et les points de données les plus proches, appelés **vecteurs de support**.
- **Optimisation:** L'algorithme SVM cherche à trouver l'hyperplan qui maximise cette marge. Plus la marge est grande, meilleure est la généralisation du modèle.

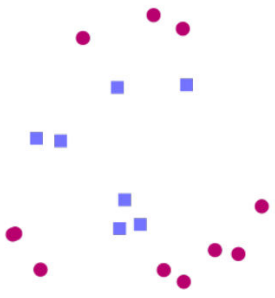
À RETENIR

"Dans le cas le plus simple, imaginons que nous avons des données qui peuvent être séparées par une ligne droite.
Cette ligne est notre hyperplan. L'objectif est de trouver la ligne qui est le plus éloignée des points de données des deux classes. Cette distance, c'est ce qu'on appelle la marge. Plus la marge est grande, meilleure est la généralisation de notre modèle."



Systèmes non linéaires: astuce du noyau

Abordons maintenant le problème des données non linéairement séparables. Pour rappel, des données sont non linéairement séparables quand il n'existe pas d'hyperplan capable de séparer correctement les deux catégories. Ce qui, d'ailleurs, arrive quasiment tout le temps en pratique.



Plongée en dimension supérieure

Pour contourner le problème, l'idée est donc la suivante: il est impossible de séparer linéairement les données dans notre espace vectoriel? Essayons dans un autre espace.

Si l'on arrivait à transposer les données dans un espace de plus grande dimension, on arriverait peut-être à trouver un hyperplan séparateur.

Plongée en dimension supérieure

Considérons l'exemple suivant, on souhaite construire un SVM qui, à partir de la taille d'un individu, nous dit si cet individu est un jeune adolescent (entre 12 et 16 ans, carrés bleus) ou non (ronds rouges).

- Entre 10 et 12 ans, on mesure entre 120 et 140 cm.
- Entre 12 et 16 ans, entre 140 et 165 cm.
- Entre 16 et 18 ans on mesure plus de 165 cm.

Peut-on trouver un hyperplan qui sépare les jeunes de 12-16 ans des autres ?

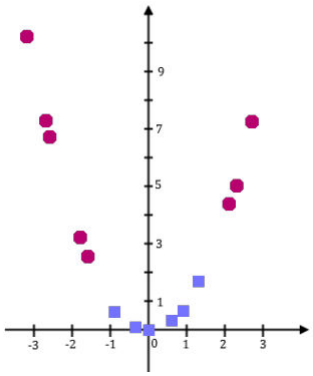
Plongée en dimension supérieure

- Entre 10 et 12 ans, on mesure entre 120 et 140 cm.
- Entre 12 et 16 ans, entre 140 et 165 cm.
- Entre 16 et 18 ans on mesure plus de 165 cm.

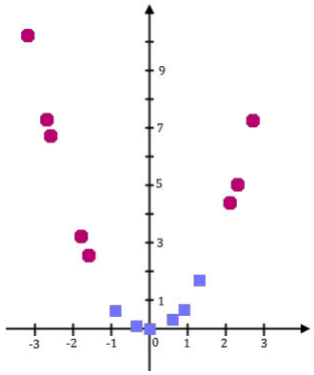
- On constate que l'on ne peut pas trouver d'hyperplan séparateur (ici, on est en dimension 1, un hyperplan séparateur est donc un simple point).
- Si on ne peut pas trouver d'hyperplan séparateur, notre SVM ne sera pas capable de s'entraîner, et encore moins de classer de nouvelles entrées…
- On essaie donc de trouver un nouvel espace, généralement de dimension supérieure, dans lequel on peut projeter nos valeurs d'entraînement, et dans lequel on pourra trouver un séparateur linéaire.

Plongée en dimension supérieure

Dans cet exemple, On passe donc de l'espace vectoriel R, de dimension 1, à l'espace vectoriel R2, de dimension 2. On se retrouve donc avec les données d'entraînement suivantes:



Plongée en dimension supérieure



On voit que les données sont alors linéairement séparables: notre SVM va pouvoir fonctionner!

La fonction Noyau (Kernel)

Le noyau est un concept fondamental dans les Machines à Vecteurs de Support (SVM). Il joue un rôle crucial en permettant aux SVM de gérer des problèmes de classification non linéaires. C'est à dire:

- Les SVM de base sont conçues pour trouver un hyperplan qui sépare linéairement les données. Or, de nombreux problèmes réels ne sont pas linéairement séparables
- Alors, le noyau va permettre de projeter les données dans un espace de plus haute dimension où elles peuvent devenir linéairement séparables. Cette projection est souvent implicite et calculée efficacement grâce à une astuce mathématique.

le noyau transforme l'espace initial en un espace où les données peuvent être séparées linéairement, même si elles ne l'étaient pas dans l'espace original.

Types de noyaux

Types de noyaux couramment utilisés :

- **Noyau linéaire:** Le plus simple, il correspond à un produit scalaire standard. Il est utilisé lorsque les données sont linéairement séparables.
- **Noyau polynomial:** Permet de capturer des relations polynomiales entre les données.
- **Noyau RBF (Radial Basis Function):** Très populaire, il permet de capturer des relations non linéaires complexes.
- **Autres noyaux:** Sigmoïde, etc.

Types de noyaux

Types de noyaux couramment utilisés :

- Noyau linéaire
- Noyau polynomial
- Noyau RBF (Radial Basis Function)
- Autres noyaux.

Le choix du noyau dépend du problème à résoudre et des caractéristiques des données. Il n'existe pas de règle universelle. On utilise souvent une approche empirique en testant différents noyaux et en sélectionnant celui qui donne les meilleurs résultats.

Implémentation de SVM en Python

1. **Importation des modules:** On importe les modules nécessaires de scikit-learn pour travailler avec les SVM, diviser les données, charger les données et évaluer les performances.
2. **Chargement des données:** On charge le jeu de données Iris, qui contient des informations sur différentes espèces d'iris.
3. **Division des données:** On divise les données en deux ensembles : un ensemble d'entraînement pour entraîner le modèle et un ensemble de test pour évaluer ses performances.
4. **Création du modèle:** On crée un modèle SVM avec un noyau RBF, qui est un choix courant pour les problèmes non linéaires.
5. **Entraînement du modèle:** On entraîne le modèle sur l'ensemble d'entraînement.
6. **Prédictions:** On utilise le modèle entraîné pour faire des prédictions sur l'ensemble de test.
7. **Évaluation:** On calcule la précision du modèle en comparant les prédictions aux vraies étiquettes.

Implémentation de SVM en Python: Code

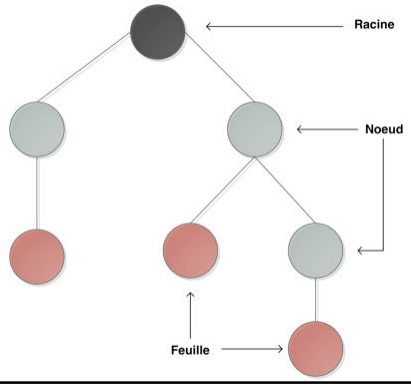
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

# Charger le jeu de données Iris
iris = load_iris()
X = iris.data
y = iris.target

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# Créer un modèle SVM avec un noyau RBF (Radial Basis Function)
model = SVC(kernel='rbf', gamma='scale')

# Entraîner le modèle
model.fit(X_train, y_train)

# Faire des prédictions sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluer les performances du modèle
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Arbres de décision

Arbres de décision

Un arbre de décision est une structure arborescente, comme illustré par la Figure ci-contre, dans laquelle :

- chaque nœud interne représente un test sur un attribut,
- chaque branche représente le résultat du test
- et chaque nœud feuille est étiqueté avec une classe (décision prise après le calcul de tous les attributs).
- Un chemin de la racine à la feuille représente les règles de classification.



Arbres de décision

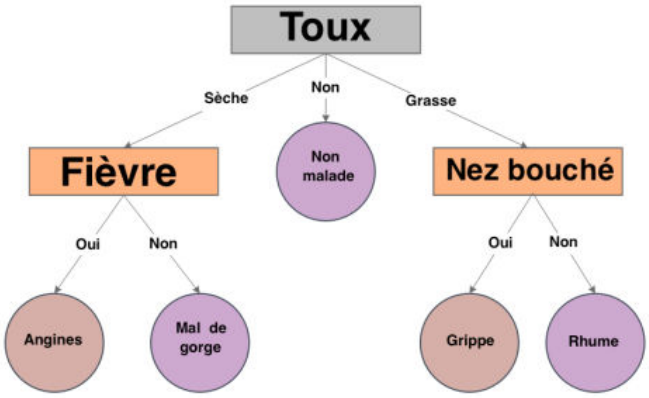
- Un arbre de décision est une méthode que nous pouvons utiliser pour faire les bons choix, en particulier les décisions qui impliquent des coûts et des risques élevés.
- Les arbres de décision utilisent une approche graphique pour comparer des alternatives concurrentes et assigner des valeurs à ces alternatives en combinant les incertitudes, les coûts et les gains en valeurs numériques spécifiques.
- Les arbres de décision trouvent une utilisation dans un large éventail d'applications, ils sont utilisés dans de nombreuses disciplines différentes, y compris le diagnostic médical, les sciences cognitives, l'intelligence artificielle, la théorie des jeux, l'ingénierie et l'exploration de données.

Arbres de décision

- Un arbre peut être appris en divisant l'ensemble source en sous-ensembles basés sur un test de valeur d'attribut.
- Ce processus est répété sur chaque sous-ensemble dérivé d'une manière récursive appelée partitionnement récursif.
- La récursion est terminée lorsque les sous-ensembles de chaque nœud ont tous la même valeur de la variable cible (on l'appelle aussi feuille pure), ou lorsque la division n'ajoute plus de valeur aux prédictions.

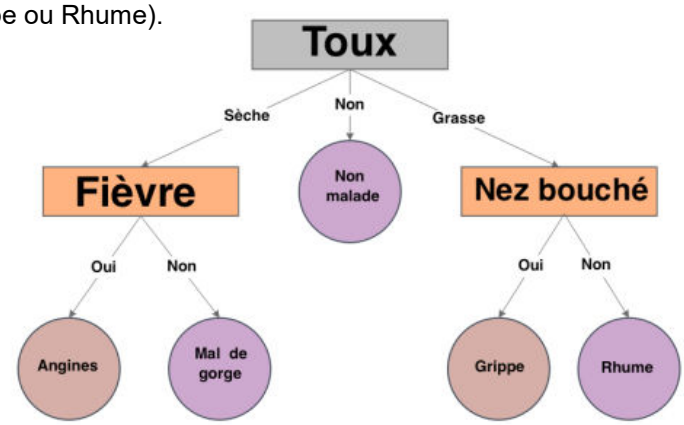
Exemple

La figure cj-dessous fournit un exemple d'arbre de décision pour une tâche de prédiction.



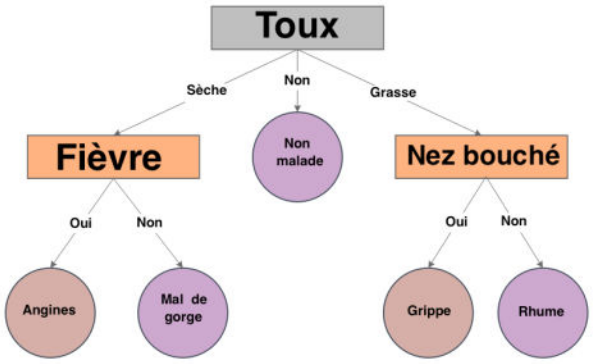
Exemple

- Dans ce cas, les nœuds de test sont décorés de questions sur les maladies.
- tandis que les feuilles sont associées à des informations sur la variable cible de sortie Malade (ici par exemple : Angines, Mal de gorge, Grippe ou Rhume).

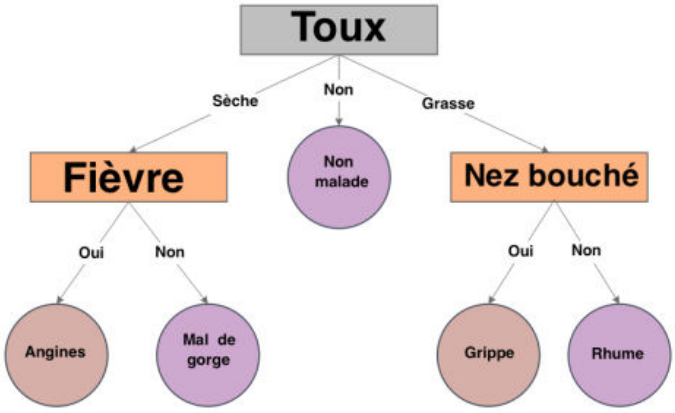


Exemple

- Chaque nœud représente un test à faire.
- Chaque branche représente une valeur possible résultant du test.
- Une feuille correspond à une prise de décision.



Exemple



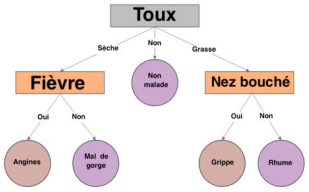
Cette représentation graphique est souvent facile à comprendre et à interpréter par des experts humains

Arbres de décision

Un arbre de décision est un modèle de machine learning qui imite le processus de décision humaine en se basant sur une série de règles. Il est particulièrement utile pour les problèmes de classification et de régression.

Structure d'un arbre de décision :

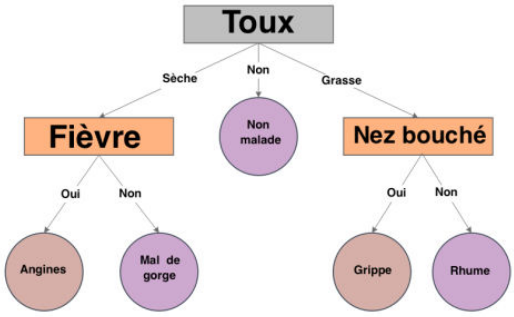
- **Nœuds:** Représentent les attributs (ou features) sur lesquels la décision est prise.
- **Branches:** Représentent les issues possibles d'un test sur un nœud.
- **Feuilles:** Représentent les classes (pour la classification) ou les valeurs (pour la régression).



Arbres de décision

Principe de fonctionnement:

L'algorithme parcourt l'arbre à partir de la racine jusqu'à une feuille, en prenant des décisions à chaque nœud en fonction de la valeur de l'attribut. La feuille atteinte correspond à la prédiction finale.



Types des arbres de décision

On distingue deux types des arbres de décision:

- **Arbres de classification**: où la variable cible est catégorique et que l'arbre est utilisé pour identifier la "classe" dans laquelle une variable cible tomberait probablement
- **Arbres de régression**: où la variable cible est continue et que l'arbre est utilisé pour prédire sa valeur.

Algorithmes d'arbre de décision

Il existe de nombreux algorithmes spécifiques d'arbre de décision, les plus connus étant :

- L'arbre de classification et de régression (CART) qui a été introduit par Breiman en 1984 (les arbres utilisés pour la régression et les arbres utilisés pour la classification ont des similitudes mais aussi des différences, comme la procédure de division).
- ID3 (Iterative Dichotomiser 3)
- C4.5 (successeur de ID3).

Les algorithmes de construction des arbres de décision fonctionnent généralement de haut en bas, en choisissant une variable à chaque étape qui scinde le mieux l'ensemble des éléments.

Avantages et inconvénients

- **Avantages:**
  - Interprétation facile
  - Traitement de données hétérogènes
  - Rapide à entraîner
- **Inconvénients:**
  - Peu performant sur des données très bruitées
  - Peut créer des arbres complexes difficiles à visualiser

Implémentation en Python

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Charger
 les données
X = ...  # Variables explicatives
y = ...  # Variable cible

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y,
 test_size=0.2, random_state=42)


# Créer et entraîner l'arbre de décision
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Faire des prédictions
y_pred = model.predict(X_test)

# Évaluer le modèle
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```