

Apprentissage automatique

January 2023



Définition de Machine Learning

- L'apprentissage automatique est une branche de l'intelligence artificielle (IA) qui se concentre sur le développement de programmes informatiques capables d'apprendre et de s'adapter grâce à l'expérience.
- Il s'agit d'un sous-ensemble de l'IA qui existe depuis les années 1950, lorsque les premiers algorithmes ont été développés.
- L'objectif principal de l'apprentissage automatique est de créer des programmes informatiques capables d'apprendre à partir de données, d'identifier des modèles et de prendre des décisions avec un minimum d'interférences humaines.
- Les systèmes d'apprentissage automatique peuvent être utilisés dans une variété d'applications telles que la vision par ordinateur, le traitement du langage naturel, la robotique et l'analyse financière.

Inventeur Machine Learning

- L'invention de l'apprentissage automatique est attribuée à Arthur Samuel, qui a développé le premier programme d'auto-apprentissage en 1959.
- Samuel était un pionnier américain dans le domaine des jeux informatiques et de l'intelligence artificielle (IA).
- Il a créé un programme de jeu de dames qui pouvait apprendre de ses erreurs et ajuster sa stratégie en fonction de son expérience.



L'apprentissage automatique prend son envol

- L'apprentissage automatique a vraiment pris son envol à la fin des années 1990, lorsque les progrès de la puissance de calcul et des algorithmes ont permis d'analyser des ensembles de données plus importants et de créer des modèles plus complexes.
- Ce nouveau niveau de puissance de calcul a permis aux chercheurs de créer des algorithmes plus puissants capables de traiter et d'apprendre de grandes quantités de données.
- Au début des années 2000, le développement de bibliothèques logicielles open source telles que TensorFlow et PyTorch a permis aux chercheurs de mettre en œuvre plus facilement leurs idées et de les partager avec d'autres chercheurs.
- Cela a conduit à une explosion de la recherche dans le domaine, avec de nouveaux algorithmes et applications développés à un rythme rapide.
- Au cours de la dernière décennie, l'apprentissage automatique est devenu de plus en plus populaire et est utilisé dans une variété d'applications, y compris la vision par ordinateur, le traitement du langage naturel, la robotique et l'analyse financière.
- Au fur et à mesure que la technologie continue de s'améliorer, on s'attend à ce que l'apprentissage automatique devienne une partie encore plus intégrante de la vie moderne.
-

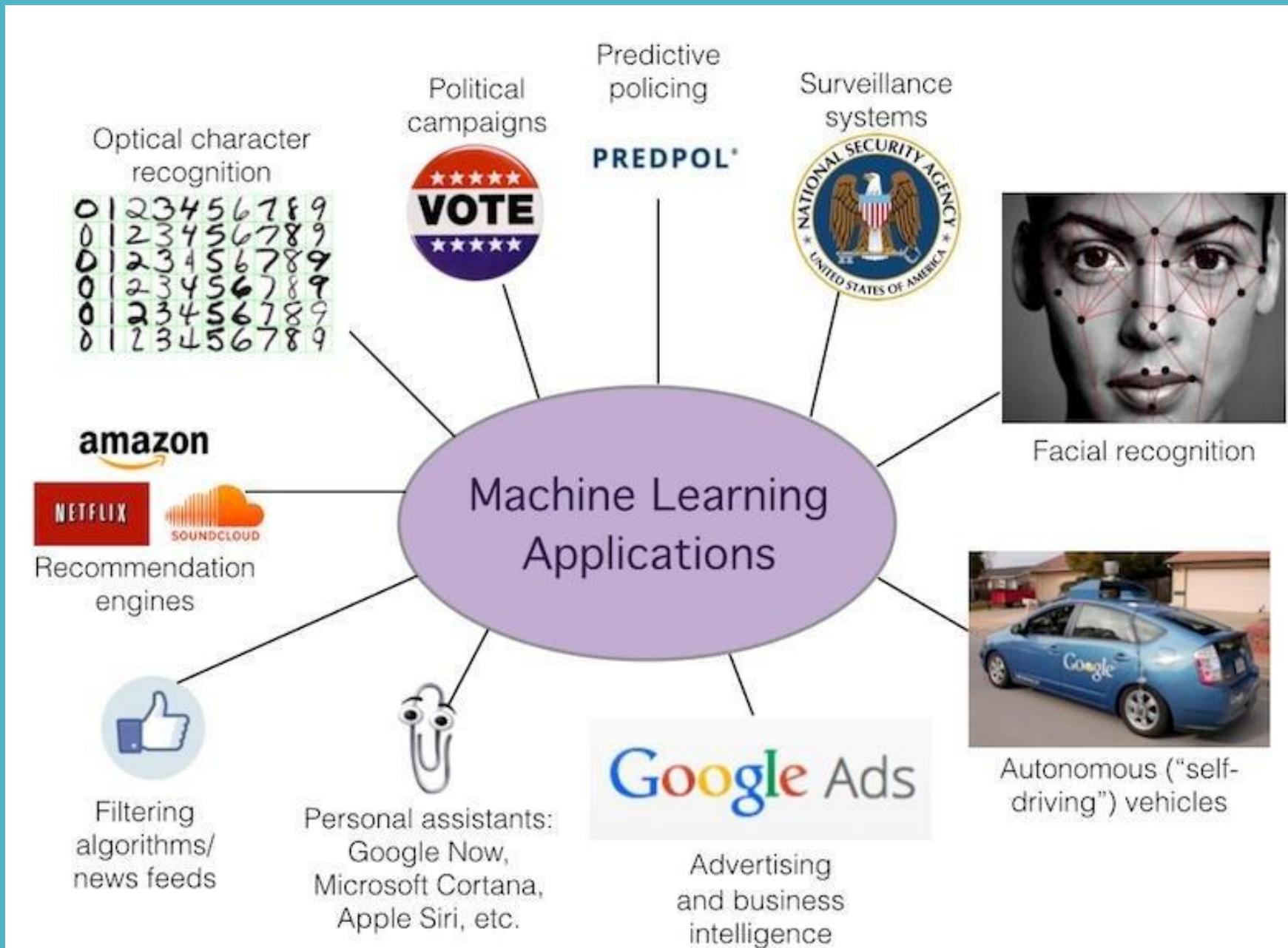


Machine Learning et informatique

- L'amélioration de l'informatique qui a rendu possible l'apprentissage automatique dans les années 1990 a été les progrès de la puissance de calcul et des algorithmes.
- Cela a permis aux chercheurs d'analyser de plus grands ensembles de données et de créer des modèles plus complexes.
- La puissance de calcul a été améliorée par l'invention de processeurs plus puissants, d'algorithmes plus efficaces et de périphériques de stockage plus rapides.
- Cela a permis aux chercheurs de créer des algorithmes d'apprentissage automatique plus puissants capables de traiter et d'apprendre à partir d'ensembles de données plus importants.
- En outre, le développement de bibliothèques de logiciels open source telles que TensorFlow et PyTorch a permis aux chercheurs de mettre en œuvre plus facilement leurs idées et de les partager avec d'autres chercheurs.
- La vitesse de traitement informatique et la capacité de mémoire spécifiques qui ont rendu possible l'apprentissage automatique étaient la capacité d'analyser des ensembles de données plus importants et de créer des modèles plus complexes.
- Cela a été rendu possible par l'invention de processeurs plus puissants, de périphériques de stockage plus rapides et d'algorithmes plus efficaces.
- La combinaison de ces avancées a permis aux chercheurs de créer des algorithmes d'apprentissage automatique plus puissants capables de traiter et d'apprendre à partir d'ensembles de données plus importants.

Applications d'apprentissage automatique

1. Vision par ordinateur: utilisé pour analyser des images et des vidéos afin de détecter des objets, des visages et d'autres caractéristiques.
2. Traitement du langage naturel: utilisé pour comprendre et répondre au langage humain.
3. Robotique: utilisé pour contrôler les robots et les véhicules autonomes.
4. Analyse financière: utilisée pour prédire les cours des actions et d'autres tendances financières.
5. Systèmes de recommandation : utilisés pour fournir des recommandations personnalisées aux utilisateurs.
6. Détection de fraude: utilisé pour détecter les activités frauduleuses dans les transactions financières.
7. Soins de santé: utilisés pour analyser des images médicales et diagnostiquer des maladies.
8. Reconnaissance vocale : utilisée pour reconnaître la parole et répondre aux commandes vocales.
9. Traitement d'image: utilisé pour reconnaître des objets dans des images et détecter des objets dans une vidéo.
10. Traduction automatique : utilisée pour traduire du texte d'une langue à une autre.



Intelligence artificielle vs Machine Learning

- L'intelligence artificielle (IA) et l'apprentissage automatique (ML) sont des domaines étroitement liés de l'informatique.
- L'IA est le concept plus large de machines capables d'effectuer des tâches d'une manière que nous considérerions comme « intelligente ».
- Le ML est un type d'IA qui permet à une machine d'apprendre à partir de données sans être explicitement programmée.
- La principale différence entre l'IA et le ML est que l'IA se concentre sur le développement de machines intelligentes capables de penser et d'agir comme des humains, tandis que le ML se concentre sur le développement de programmes informatiques capables d'apprendre et de s'adapter à l'évolution des données.
- L'IA est généralement utilisée pour décrire des machines capables d'imiter le comportement humain et de résoudre des problèmes, tandis que le ML est utilisé pour décrire des machines capables d'apprendre de grandes quantités de données et d'identifier des modèles.
- Les algorithmes ML sont utilisés pour développer des applications d'IA telles que les voitures autonomes, la reconnaissance faciale et le traitement du langage naturel.

Machine Learning vs Deep Learning

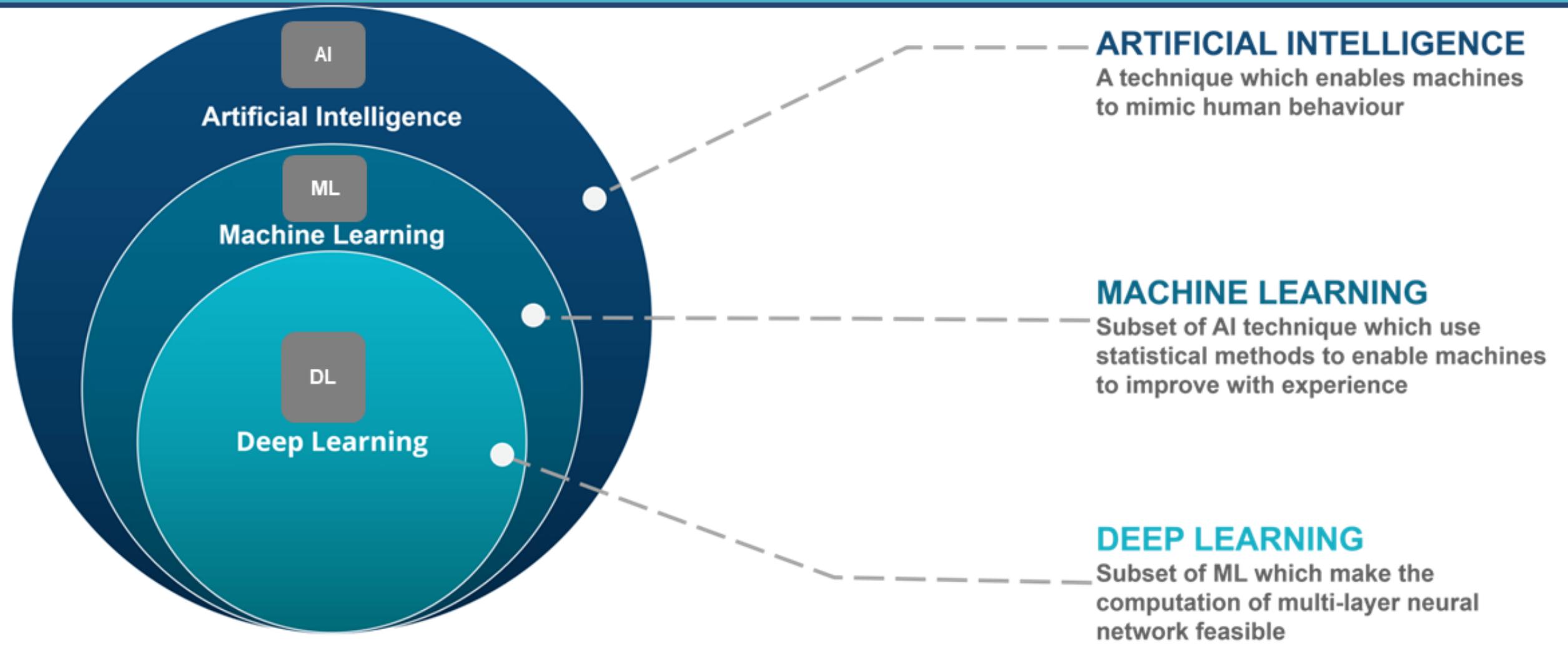
- La principale différence entre l'apprentissage automatique et l'apprentissage profond est que les algorithmes d'apprentissage automatique sont conçus pour apprendre à partir des données et identifier des modèles, tandis que les algorithmes d'apprentissage profond sont conçus pour apprendre des données et prendre des décisions.
- Les algorithmes d'apprentissage automatique sont utilisés pour développer des applications d'IA telles que la vision par ordinateur, le traitement du langage naturel, la robotique et l'analyse financière.
- Les algorithmes d'apprentissage profond sont utilisés pour développer des applications d'IA telles que les voitures autonomes, la reconnaissance faciale et le traitement du langage naturel.

Algorithmes d'apprentissage automatique

1. Arbres de décision : utilisés pour les tâches de classification et de régression.
2. Support Vector Machines (SVM) : utilisées pour les tâches de classification et de régression.
3. Naïve Bayes Classifier : utilisé pour les tâches de classification.
4. K-Nearest Neighbors (KNN) : utilisé pour les tâches de classification et de régression.
5. Forêt aléatoire : utilisée pour les tâches de classification et de régression.
6. Régression logistique : utilisée pour les tâches de classification.
7. Gradient Boosting: utilisé pour les tâches de classification et de régression.
8. Réseaux de neurones : utilisés pour les tâches de classification et de régression.
9. K-Means Clustering: utilisé pour les tâches d'apprentissage non supervisées.
10. Il existe de nombreux autres algorithmes d'apprentissage automatique tels que les modèles de Markov cachés (HMM), la maximisation des attentes (EM) et les réseaux de croyances profondes (DBN). Il existe également de nombreux autres algorithmes d'apprentissage profond tels que les machines de Boltzmann, les réseaux antagonistes génératifs (GAN) et l'apprentissage par renforcement profond.

Algorithmes d'apprentissage profond

1. Réseaux de neurones convolutifs (CNN) : utilisés pour analyser des images et des vidéos.
2. Réseaux neuronaux récurrents (RNN) : utilisés pour le traitement du langage naturel et la reconnaissance vocale.
3. Réseaux antagonistes génératifs (GAN) : utilisés pour générer des images et des vidéos.
4. Réseaux de mémoire à long terme (LSTM) : utilisés pour le traitement du langage naturel et la reconnaissance vocale.
5. Autoencodeurs : utilisés pour apprendre des représentations compressées de données.
6. Reinforcement Learning : utilisé pour l'entraînement des robots et des véhicules autonomes.



Artificial Intelligence

TRANSFORMA
INSIGHTS

Mimicking of
human
intelligence by
computers

Business rules
including
IFTTT

Machine Learning

Application of
statistical
techniques

Learn to
improve with
experience

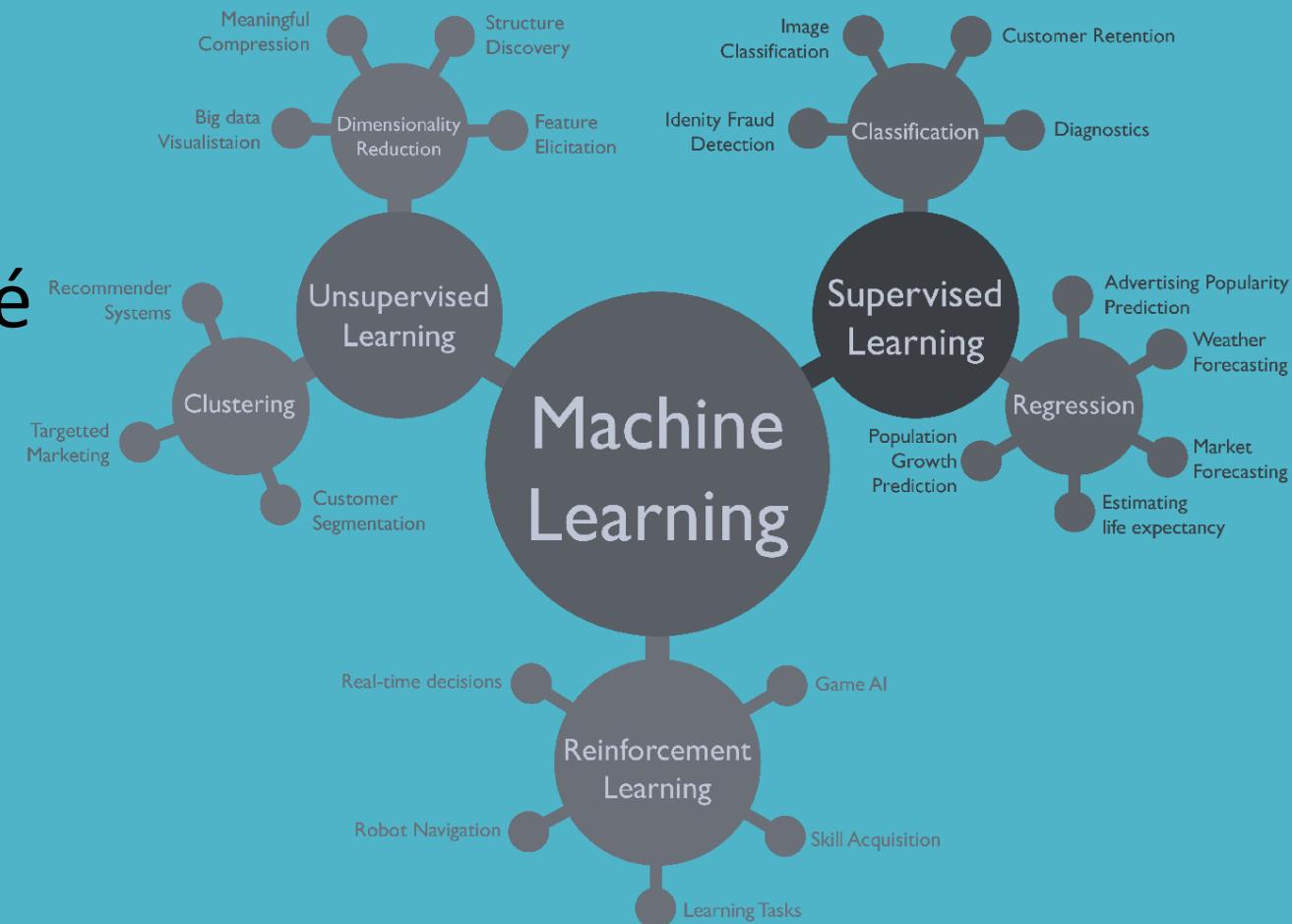
Deep Learning

Self-training software

Application of neural
networks

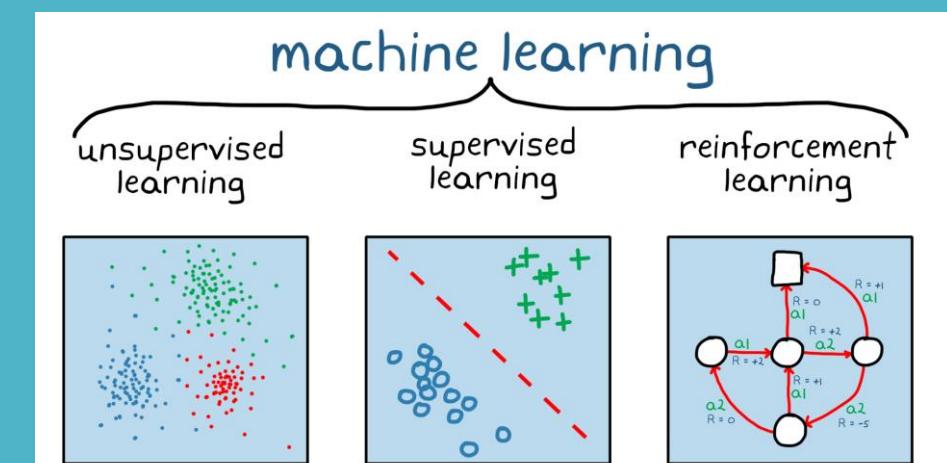
Types d'apprentissage automatique

1. Apprentissage supervisé
2. Apprentissage non supervisé
3. Apprentissage par renforcement
4. Apprentissage semi-supervisé



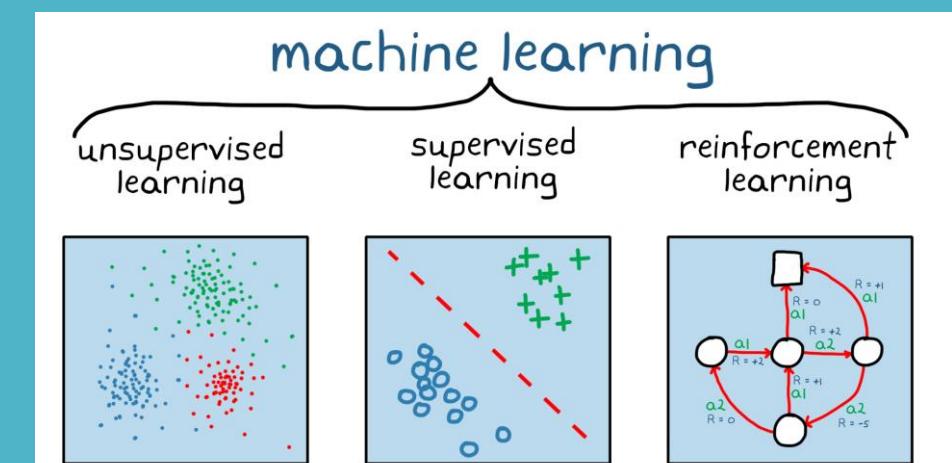
Apprentissage supervisé

- L'apprentissage supervisé implique la formation d'un algorithme d'apprentissage automatique avec des données étiquetées.
- L'algorithme est entraîné pour reconnaître les modèles dans les données et faire des prédictions basées sur ces modèles.
- Les algorithmes d'apprentissage supervisé courants incluent les arbres de décision, les machines à vecteurs de support, les classificateurs bayésiens naïfs, la régression logistique et les forêts aléatoires.



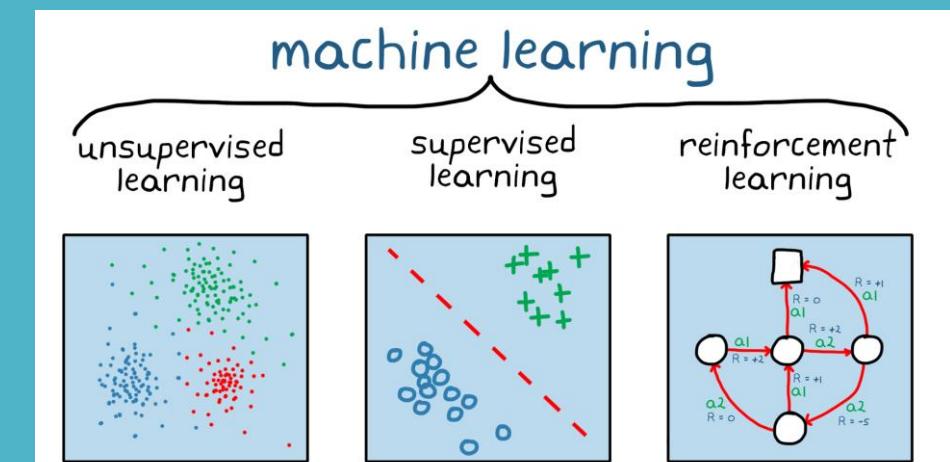
Apprentissage non supervisé

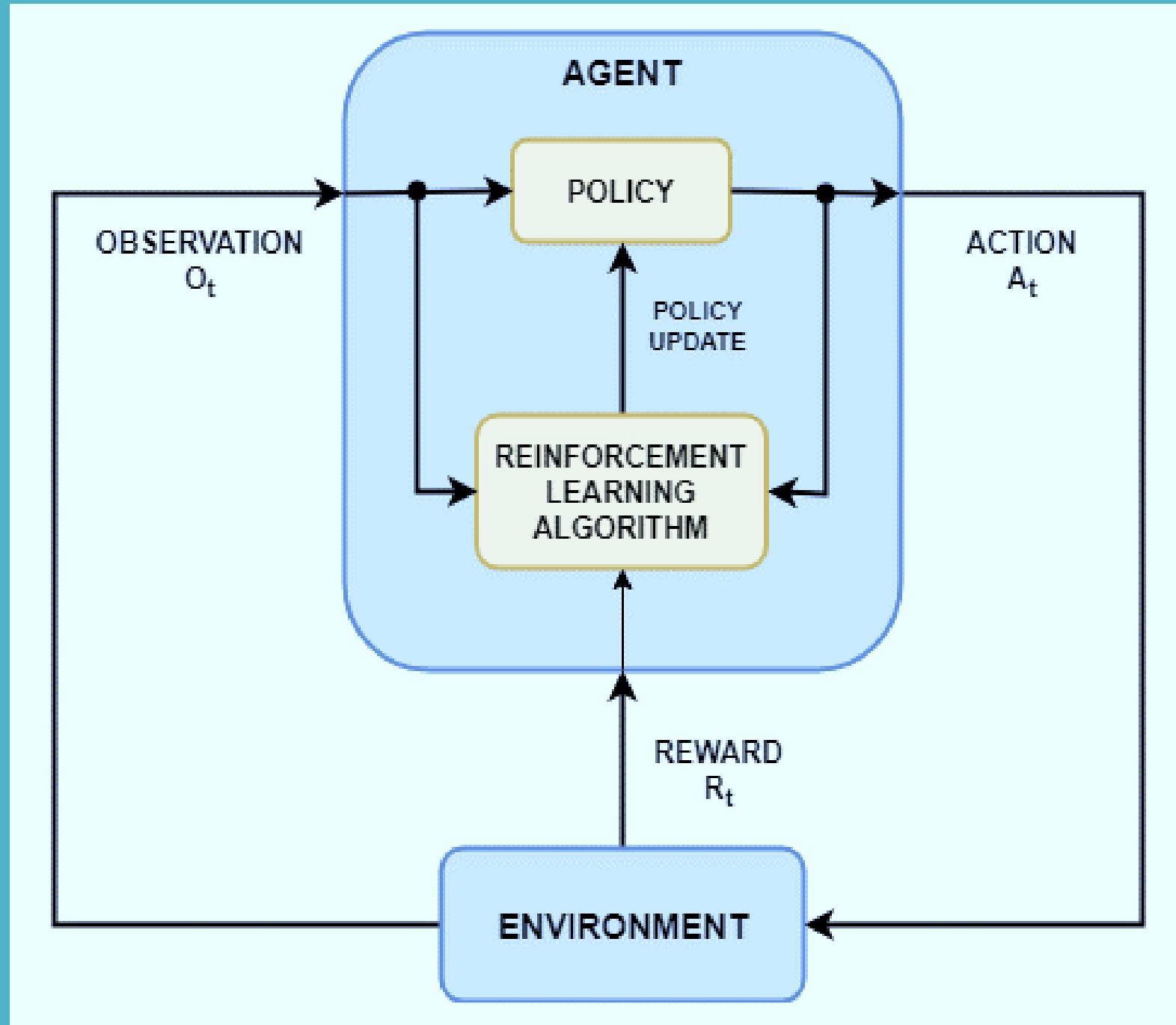
- L'apprentissage non supervisé implique l'entraînement d'un algorithme d'apprentissage automatique avec des données non étiquetées.
- L'algorithme est entraîné pour identifier des modèles dans les données sans aucune étiquette.
- Les algorithmes d'apprentissage non supervisé courants incluent le clustering K-means et le clustering hiérarchique.



Apprentissage par renforcement

- L'apprentissage par renforcement est un type d'apprentissage automatique où un agent est formé pour prendre des mesures dans un environnement afin de maximiser une récompense.
- Ce type d'apprentissage automatique est utilisé pour entraîner des robots et des véhicules autonomes.
- Les algorithmes courants d'apprentissage par renforcement incluent le Q-learning et la recherche d'arbres de Monte Carlo.





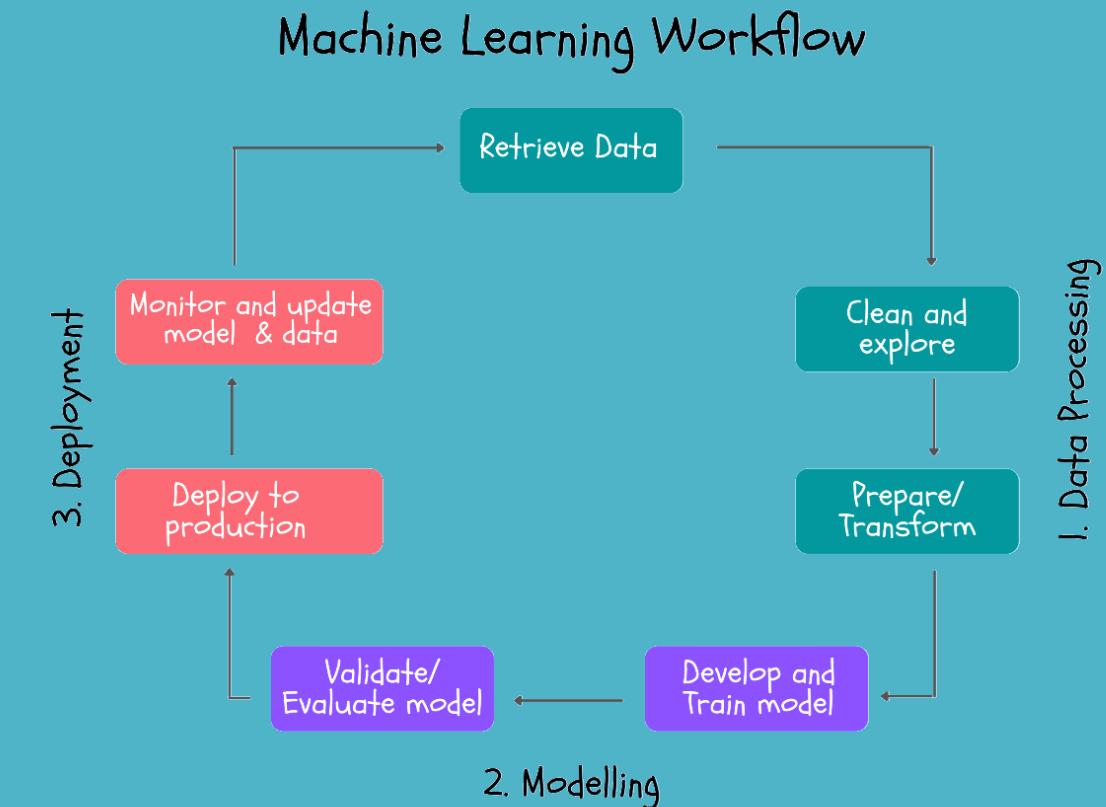
Apprentissage semi-supervisé

- L'apprentissage semi-supervisé est un type d'apprentissage automatique qui combine l'apprentissage supervisé et non supervisé.
- Dans ce type d'apprentissage automatique, l'algorithme est entraîné avec des données étiquetées et non étiquetées.
- Les données étiquetées sont utilisées pour entraîner l'algorithme à reconnaître les modèles dans les données, tandis que les données non étiquetées

Model	Learning task
Supervised Learning Algorithms	
Nearest Neighbor	Classification
Naive Bayes	Classification
Decision Trees	Classification
Classification Rule Learners	Classification
Linear Regression	Numeric prediction
Regression Trees	Numeric prediction
Model Trees	Numeric prediction
Neural Networks	Dual use
Support Vector Machines	Dual use
Unsupervised Learning Algorithms	
Association Rules	Pattern detection
k-means clustering	Clustering
Meta-Learning Algorithms	
Bagging	Dual use
Boosting	Dual use
Random Forests	Dual use

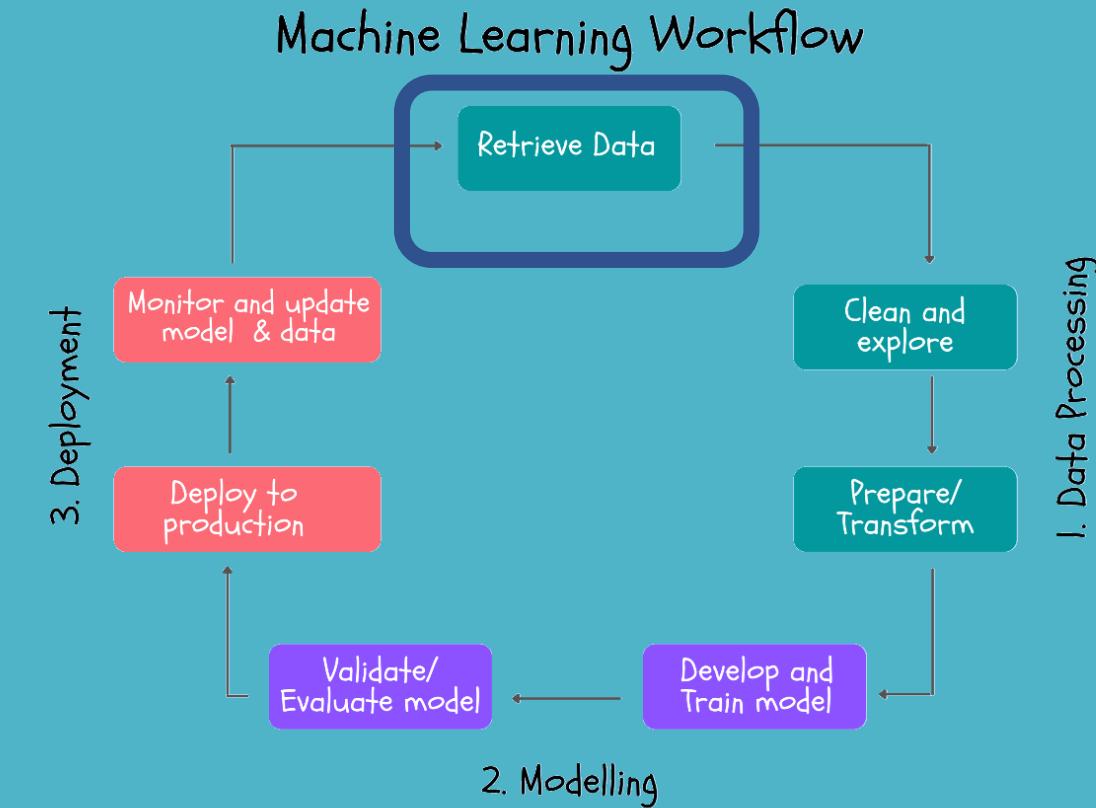
Flux de travail d'apprentissage automatique

- **Collecte de données** : collecte de données provenant de diverses sources.
- **Prétraitement des données** : nettoyage, transformation et normalisation des données.
- **Model Training** : formation du modèle sur les données traitées.
- **Évaluation du modèle** : évaluation du modèle sur les données de test.
- **Déploiement de modèle** : déploiement du modèle pour une utilisation dans un environnement de production.
- **Surveillance du modèle** : surveillance de la précision et des performances du modèle.



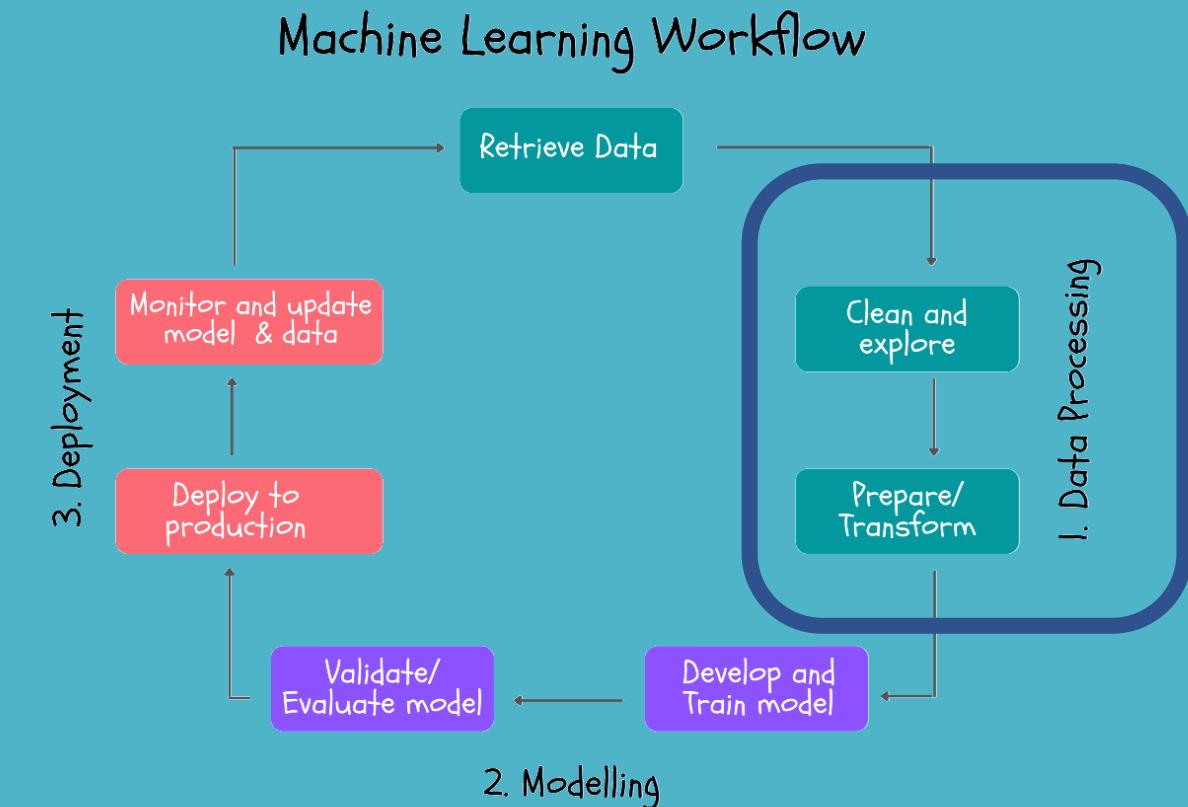
Machine learning : collecte de données

- La collecte de données à partir de diverses sources est la première étape du flux de travail d'apprentissage automatique.
- Cela implique la collecte de données à partir de bases de données, d'API Web et d'autres sources.
- Il est important de s'assurer que les données sont complètes, exactes et pertinentes pour la tâche à accomplir.



Machine learning : Prétraitement des données

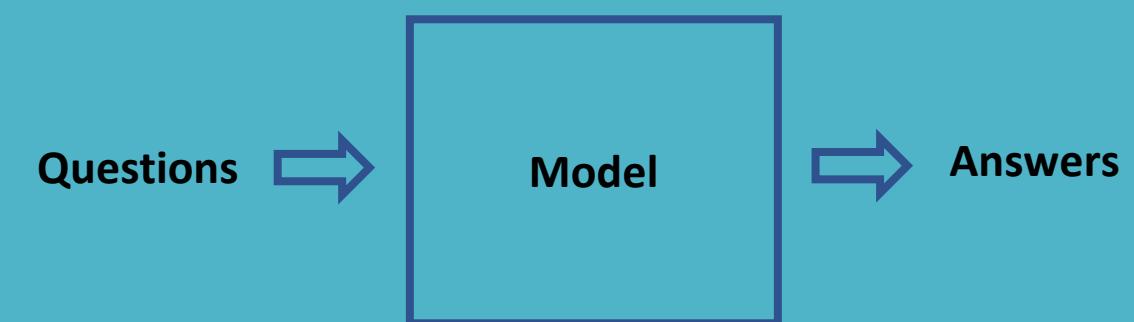
- Nettoyez, transformez et normalisez les données.
- Cela implique de supprimer les valeurs aberrantes, de transformer les variables catégorielles en variables numériques et de mettre à l'échelle les données.
- Il est également important de diviser les données en ensembles d'apprentissage, de validation et de test.



Phase 1: Training – deriving a model

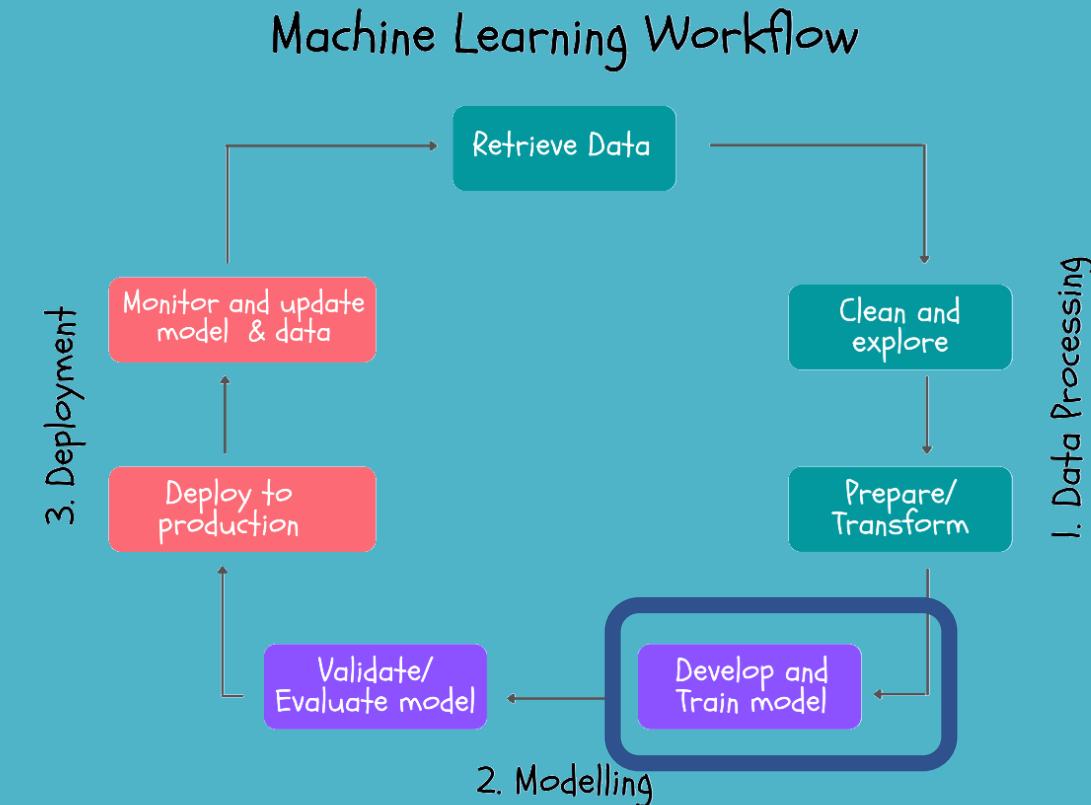


Phase 2: Inference – using the model



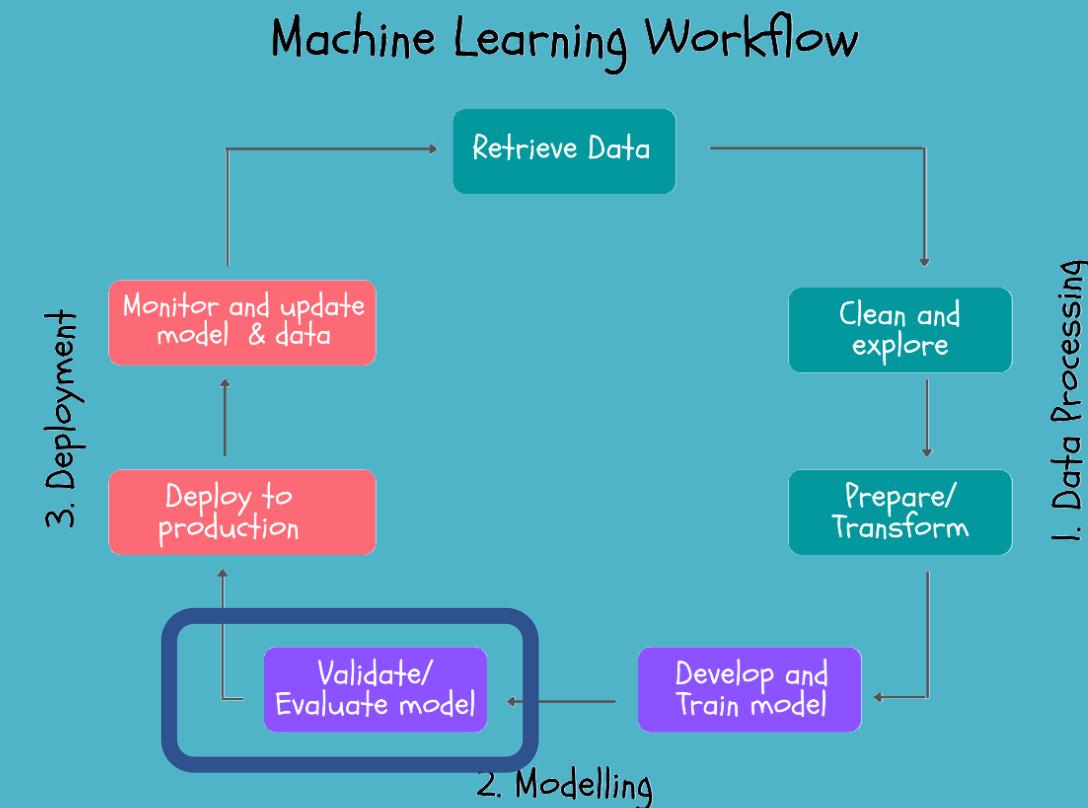
Machine learning : formation de modèles

- Après le prétraitement des données, le modèle est formé sur les données traitées.
- Cela implique d'ajuster le modèle aux données et d'ajuster les paramètres.
- Il est important de choisir le bon modèle pour la tâche à accomplir et de s'assurer qu'il ne surajuste pas ou ne sous-ajuste pas les données.



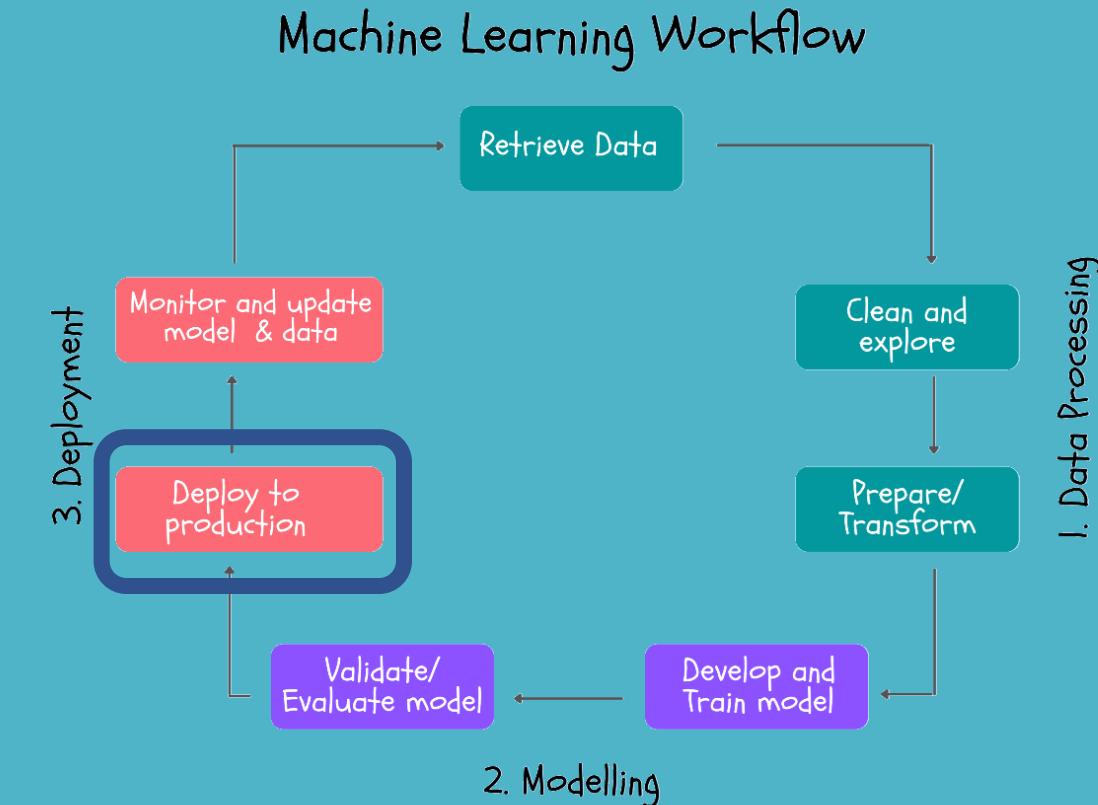
Machine learning : évaluation de modèle

- Après l'entraînement du modèle, il est évalué sur les données de test.
- Cela implique de mesurer la précision et la performance du modèle et d'apporter les ajustements nécessaires.
- Les mesures courantes pour évaluer les modèles d'apprentissage automatique incluent l'exactitude, la précision, le rappel et le score F1.
- Collecte de données : collecte de données auprès de



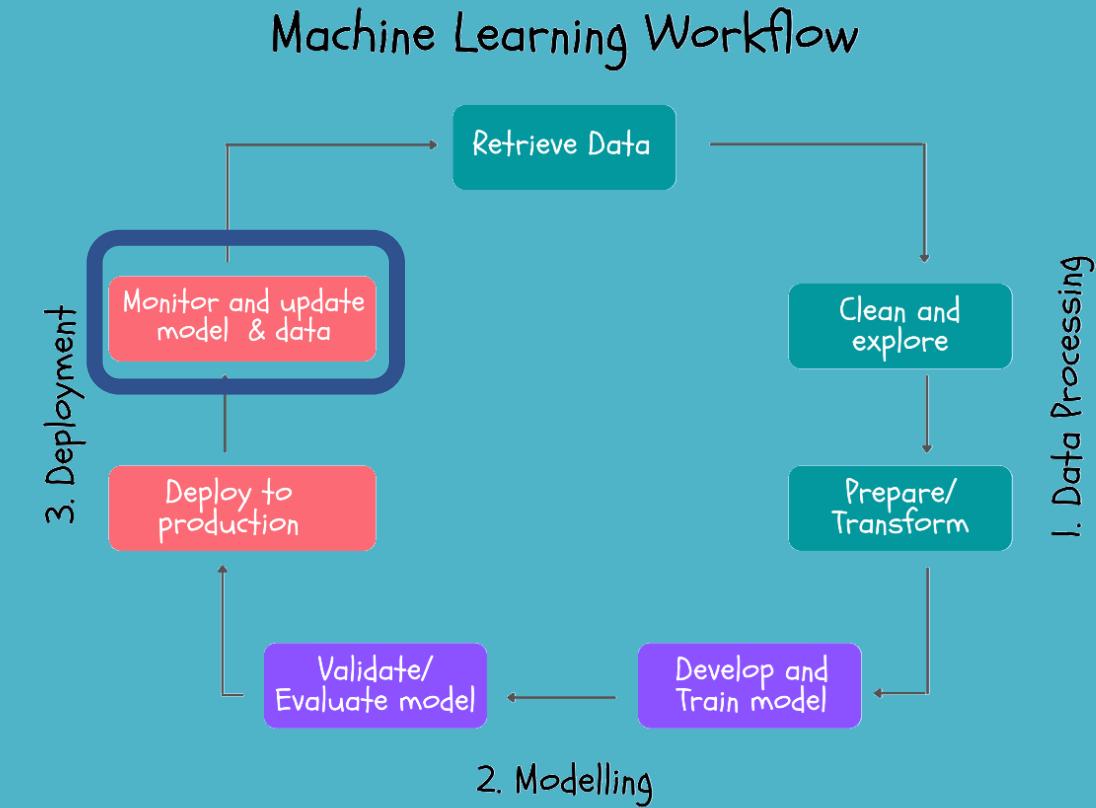
Machine learning : déploiement de modèles

- Le modèle est ensuite déployé dans un environnement de production, ce qui lui permet d'être utilisé en continu dans des applications réelles.
- Il est important de tenir compte du coût de la formation et du déploiement du modèle, ainsi que des implications en matière de confidentialité et de sécurité des données.



Machine learning : Surveillance des modèles

- Le modèle est ensuite surveillé pour la précision et la performance.
- Cela implique de mettre à jour le modèle au besoin et de suivre ses performances au fil du temps.
- Les mesures courantes pour la surveillance des modèles d'apprentissage automatique incluent l'exactitude, la précision, le rappel et le score F1



k-NN

Apprentissage automatique : k-NN

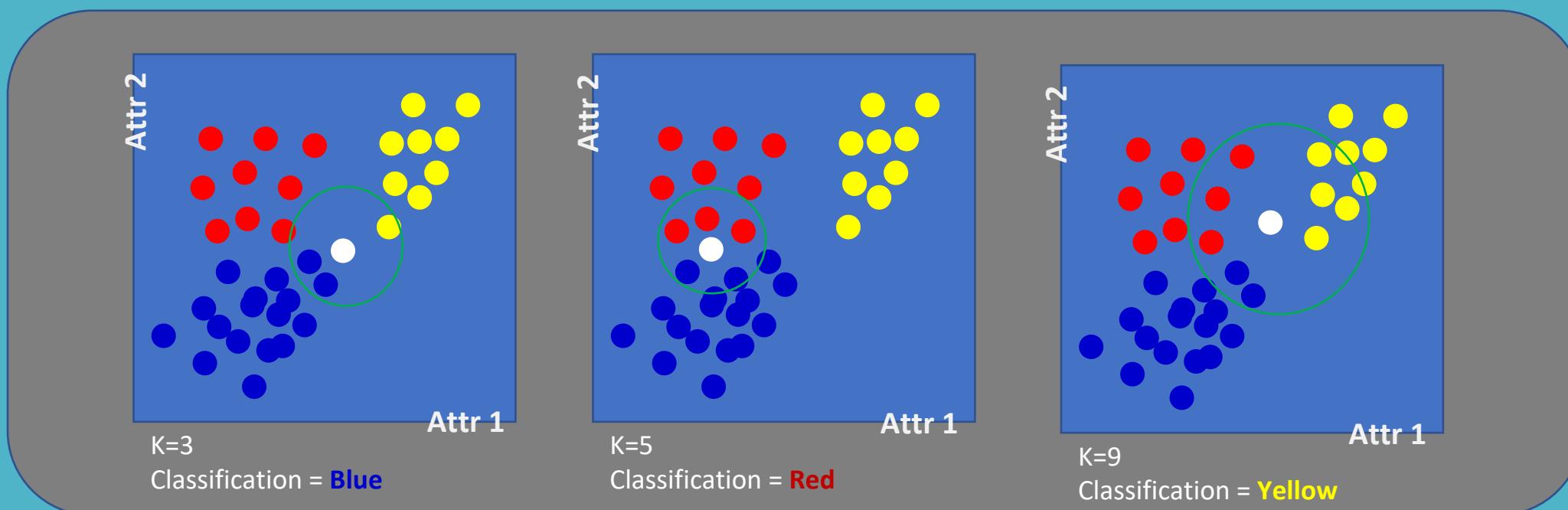
- Classer les échantillons non étiquetés en les affectant à une classe d'échantillons étiquetés similaires
- La similarité est déterminée par l'une des nombreuses mesures de distance
- Le k dans kNN est le nombre de voisins auxquels l'échantillon est comparé
- En règle générale, un vote majoritaire détermine à quelle classe appartient l'échantillon.
- Apprenant paresseux : l'algorithme kNN classe sans apprendre grand-chose sur la répartition des données dans le jeu de données. C'est le contraire des apprenants enthousiastes
- kNN est une méthode de classification basée sur les instances : le calcul est retardé jusqu'à la classification
- La classification kNN est un algorithme de classification d'apprentissage supervisé
- La régression kNN n'est pas la même chose que la classification kNN
- La régression kNN fait la moyenne des valeurs des k plus proches voisins au lieu de prendre un vote majoritaire
- Ils sont connus sous le nom de régression pondérée localement

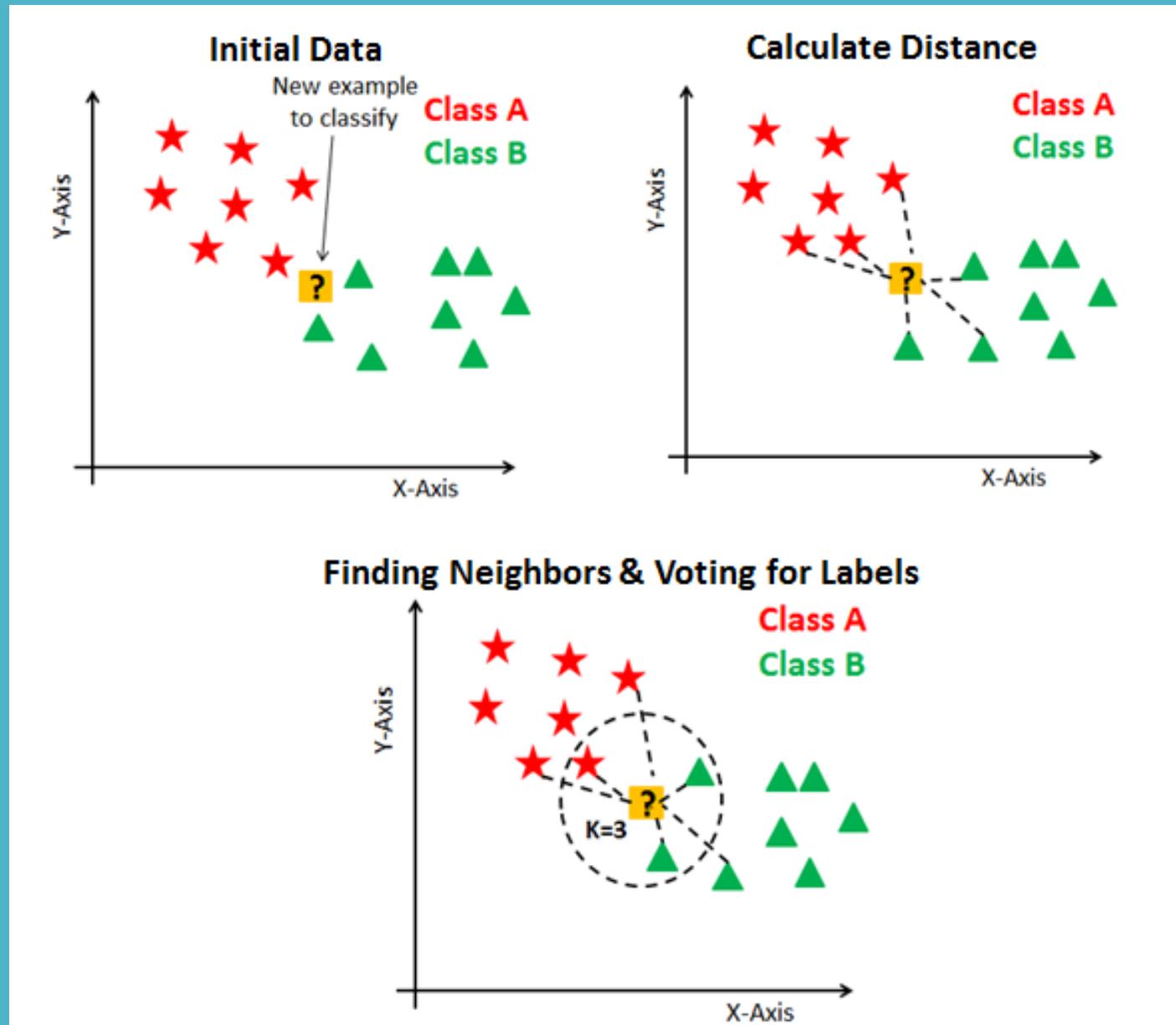
k-NN : terminologie

- k : nombre de voisins dans la classification
- Instances : nombre d'observations, échantillons utilisés dans la classification (formation et essais)
- Attributs : nombre de caractéristiques, paramètres utilisés pour comparer la distance entre les échantillons
- Mesure de la distance : méthode de calcul de la distance entre les échantillons, par exemple distance euclidienne
- Elbow Method : technique utilisée pour déterminer le k optimal pour un ensemble de données donné

k-NN : visualisation

- Dans cette illustration, le cercle vert capture les k plus proches voisins. Ensuite, la classification se fait à la majorité des voix. Quelle que soit la classe ayant la majorité au sein du cercle, remporte le vote.





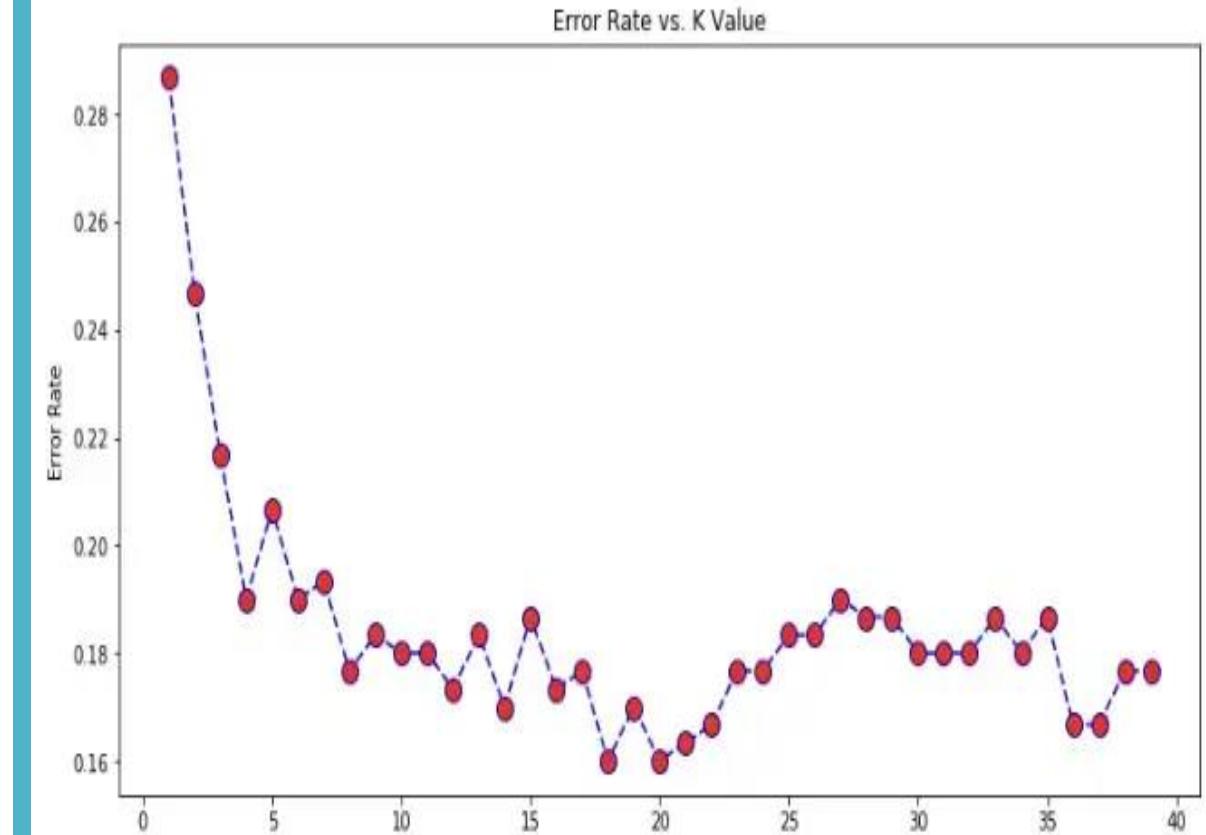
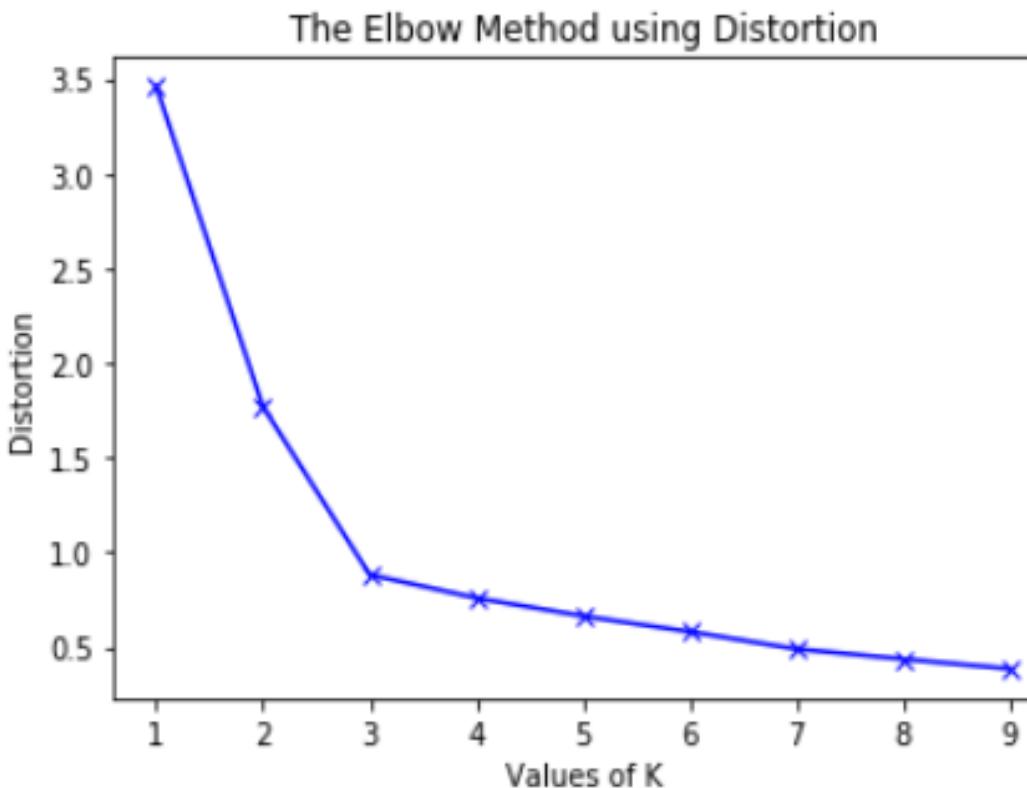
k-NN : avantages et inconvénients

- **Avantages:**
- kNN est robuste même avec des ensembles de données bruyants
- Efficacité relative avec de grands ensembles de données
- L'algorithme est exécuté en un seul passage, chaque échantillon est évalué une fois
- kNN peut modéliser facilement des modèles relativement complexes
- **Inconvénients:**
- Trouver le k optimal n'est pas systématique
- Calculs de distance par « force brute » : chaque échantillon est comparé à toutes les données du jeu de données d'apprentissage
- Les grands ensembles de données peuvent causer des problèmes de performance en ce qui concerne le coût de calcul, bien que la précision finira par être relativement élevée
- Les valeurs aberrantes peuvent entraîner une augmentation des erreurs de classification

k-NN : sélection de k

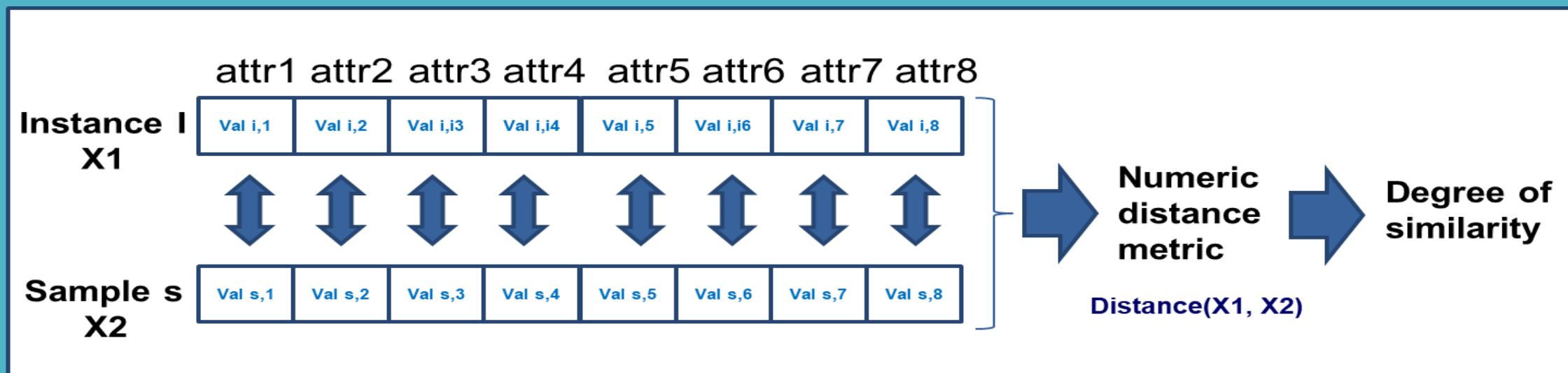
- Règle empirique : choisissez k à peu près égal à la racine carrée du nombre de classes
- k devrait être un nombre impair pour éviter l'égalité des voix dans le vote majoritaire
- k ne doit pas être un multiple du nombre de classes pour éviter les liens N-way ($N = \#$ classes)
- k ne peut pas être trop petit ou trop grand
- K Trop petit : problèmes de surajustement
- k trop grand:
- Problèmes d'ajustement dus à l'apprentissage de la distribution locale des données
- Coût de calcul élevé du calcul des distances pour chaque paire d'instances possible
- Les performances de classification diminuent à mesure que k et la taille du jeu de données augmentent

K-NN: elbow method



K-NN : concept de similitude

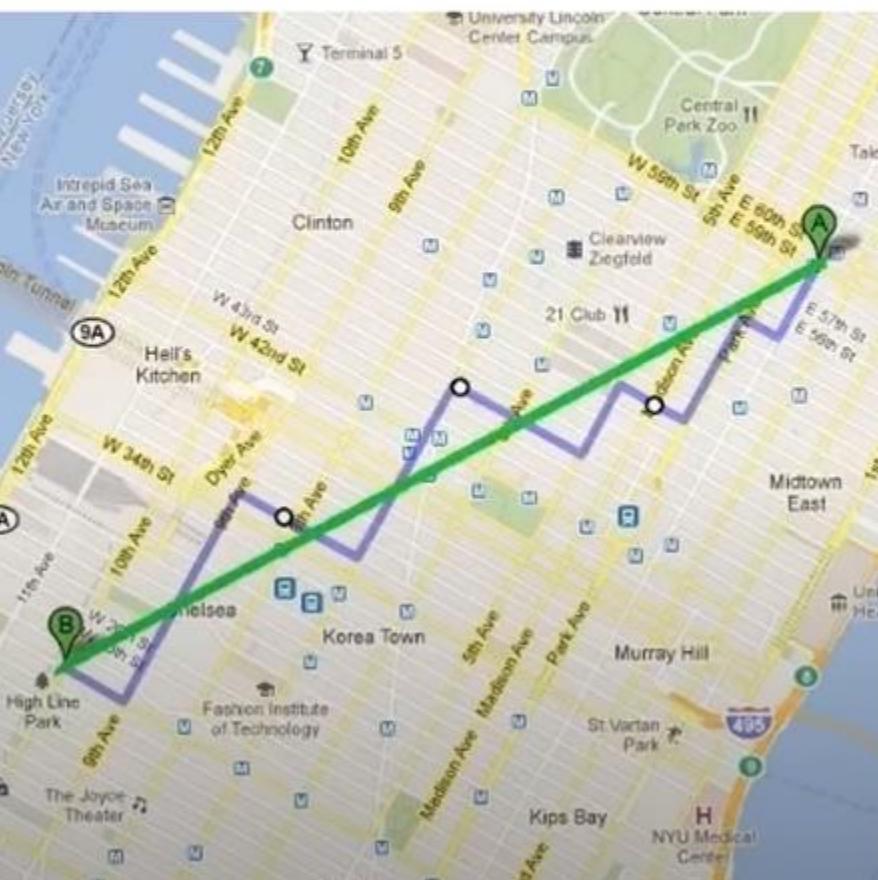
- La similitude entre les données d'échantillon et d'entraînement en kNN est calculée en appliquant une métrique de distance
- Une façon de visualiser le concept est de penser à chaque instance de l'ensemble d'apprentissage comme un vecteur de valeurs numériques val i, j de chaque attribut attrj
- Le calcul de la similarité entre l'échantillon et les instances revient à calculer les distances en N dimensions entre ces vecteurs



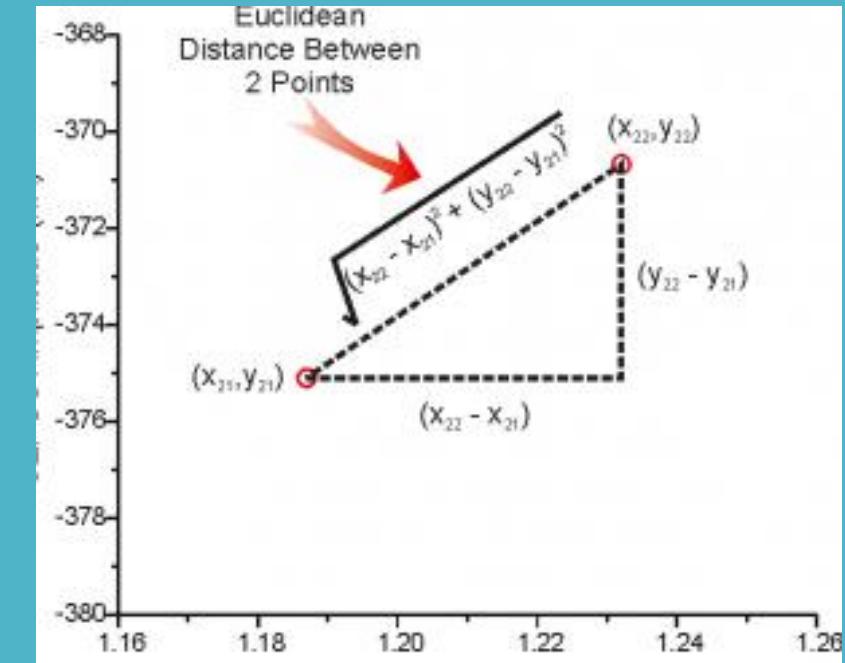
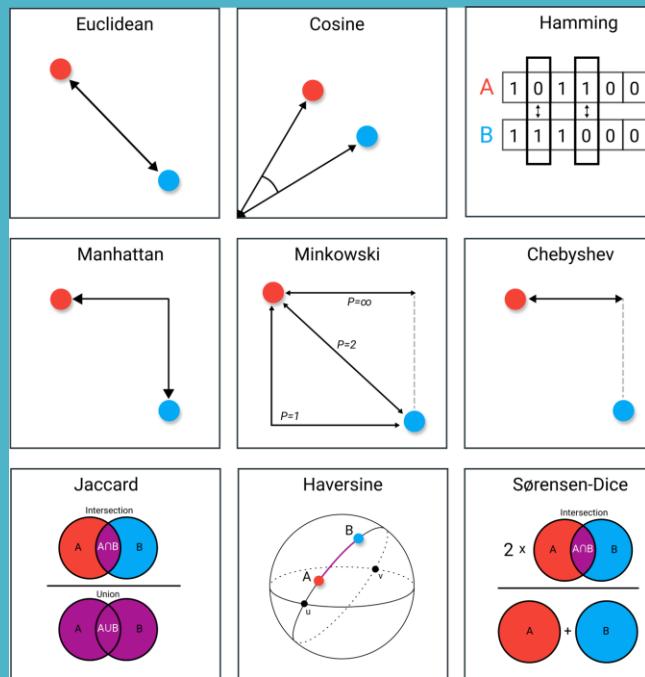
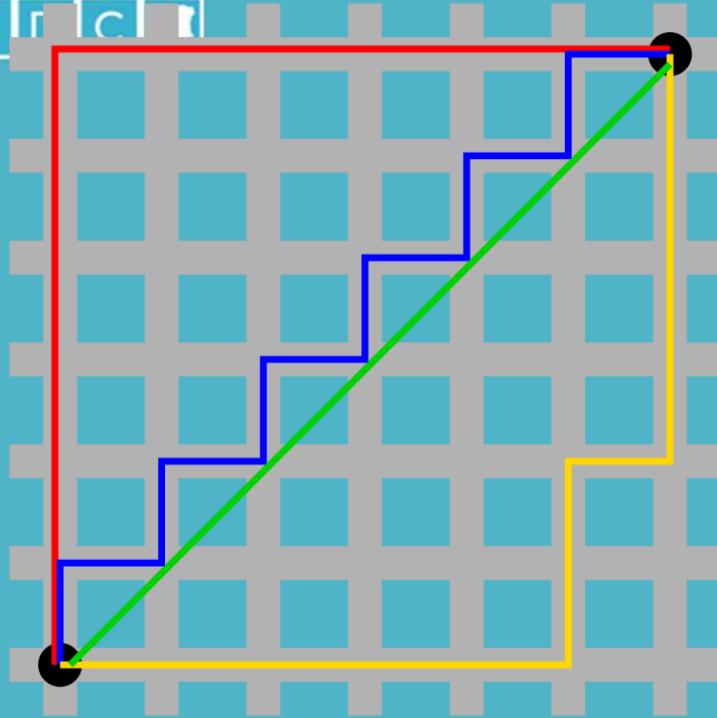
k-NN : mesures de distance

- Distance euclidienne: distance basée sur la formule de distance pythagoricienne dans un système de coordonnées cartésiennes. Couramment utilisé dans la classification kNN
- Distance de Manhattan: également connue sous le nom de distance de pâté de maisons ou géométrie de taxi
- Distance de Minkowski: formule de distance généralisée dans laquelle les cas spéciaux $p = 2$ est la distance euclidienne et $p = 1$ est la distance de Manhattan
- Distance de Hamming: nombre de positions par lesquelles deux vecteurs de longueur égale diffèrent. Très populaire en théorie des codes
- Distance cosinus : mesure la similitude entre deux vecteurs non nuls d'un espace produit interne.
- Indépendamment de leur magnitude vectorielle, la similitude cosinus est égale à
- 1 si deux vecteurs sont orientés dans la même direction
- -1 si deux vecteurs sont orientés dans des directions opposées
- 0 si deux vecteurs sont perpendiculaires

Manhattan Distance vs Euclidean Distance



- La distance euclidienne est la métrique de distance par défaut et la plus populaire utilisée dans la classification k-NN.
- D'autres métriques, comme la distance Manhattan, doivent être prises en compte pour voir si leur utilisation améliore le taux de précision de la classification (réduit le taux d'erreur)



Some frequently used distance functions.	
Camberra :	
$d(x, y) = \sum_{i=1}^m \frac{ x_i - y_i }{ x_i + y_i }$	(2)
Minkowsky :	
$d(x, y) = \left(\sum_{i=1}^m x_i - y_i ^r \right)^{\frac{1}{r}}$	(3)
Chebychev :	
$d(x, y) = \max_{i=1}^m x_i - y_i $	(4)
Euclidean :	
$d(x, y) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2}$	(5)
Manhattan / city - block :	
$d(x, y) = \sum_{i=1}^m x_i - y_i $	(6)

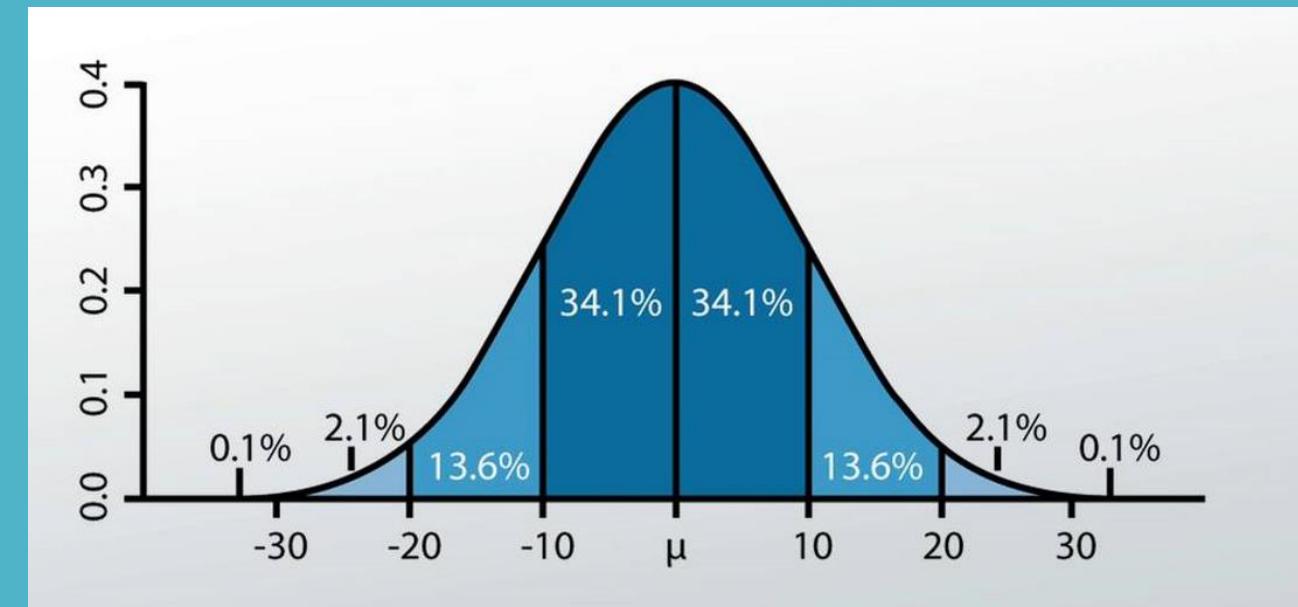
Normalisation des données

- Pourquoi normaliser les données ?
- Les classificateurs calculent la distance entre deux entités (attributs) à l'aide d'une métrique de distance.
- Si l'une des entités a une large gamme de valeurs et/ou de nombreuses valeurs aberrantes, la distance sera dominée par cette caractéristique, ce qui affectera les performances du classificateur.
- La normalisation des valeurs des caractéristiques garantit que chaque entité contribue proportionnellement aux calculs de distance
- Les méthodes de normalisation comprennent la normalisation du score z, la normalisation min-max, la normalisation moyenne et la mise à l'échelle à la longueur unitaire.

Normalisation du Z-score

- Normalisation du score Z: Pour chaque caractéristique, la moyenne de distribution et l'écart-type sont calculés. La moyenne est soustraite de chaque caractéristique. Ensuite, le résultat est divisé par son écart-type
- Cette transformation conduit à des valeurs centrées autour de chaque caractéristique moyenne
- Les valeurs peuvent, en théorie, aller de $-\infty$ à $+\infty$. Cependant, pour les données normalement distribuées, près de 100 % des valeurs se situeront à moins de 3 écarts-types de la moyenne

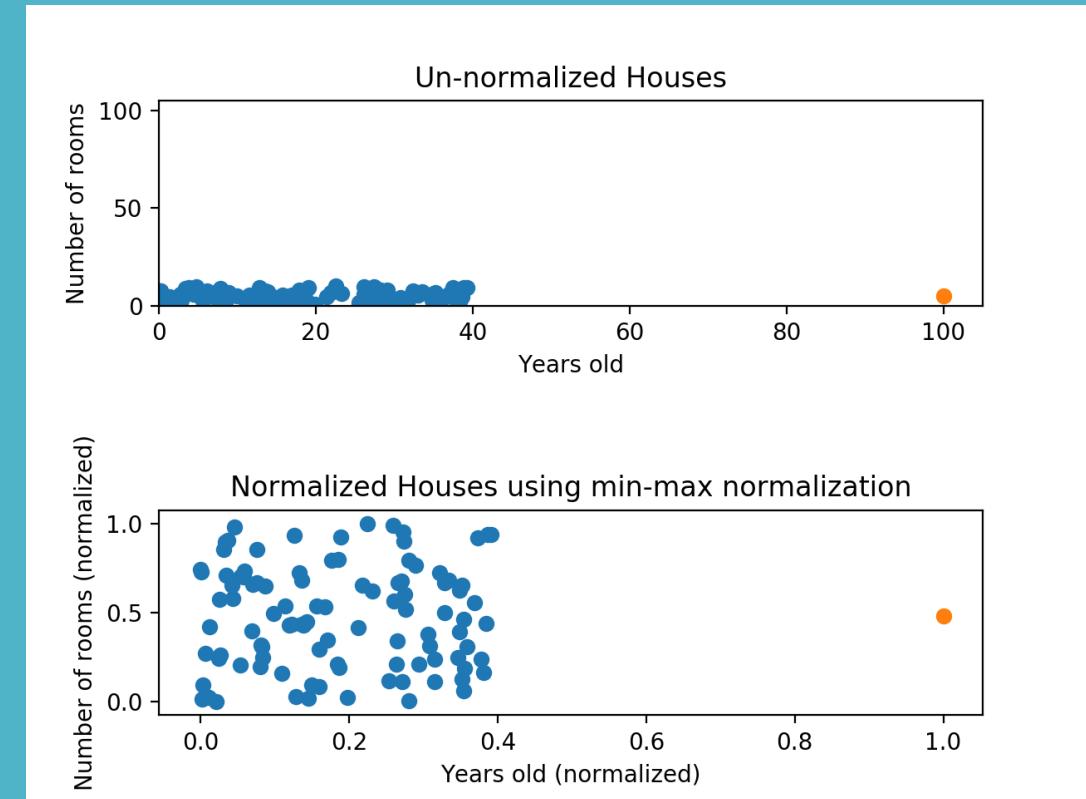
$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$



Normalisation Min-Max

- Normalisation Min-Max : pour chaque entité (attribut), la valeur minimale de cette caractéristique est transformée en 0, la valeur maximale en 1 et toutes les autres valeurs sont transformées en décimales comprises entre 0 et 1.

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$



Naive Bayes

Naïve Bayes

- Naive Bayes est un algorithme de classification probabiliste basé sur le théorème de Bayes, qui stipule que la probabilité d'une hypothèse (dans ce cas, une étiquette de classe) étant donné certaines observations (dans ce cas, les valeurs de caractéristique) est égale à la probabilité des observations étant donné l'hypothèse, multipliée par la probabilité préalable de l'hypothèse, divisée par la probabilité des observations. En d'autres termes,
- $P(\text{classe} | \text{données}) = (P(\text{données} | \text{classe}) * P(\text{classe})) / P(\text{données})$
- Naive Bayes est naïf parce qu'il fait une hypothèse forte et irréaliste sur l'indépendance des caractéristiques, ce qui signifie que la présence ou l'absence d'une caractéristique particulière dans les données n'est pas liée à la présence ou à l'absence de toute autre caractéristique.
- Malgré cette hypothèse, l'algorithme fonctionne souvent bien dans la pratique, en particulier lorsque le nombre de fonctionnalités est important.

Naïve Bayes Types

- Il existe plusieurs variantes de Bayes naïfs, notamment les bayes naïfs gaussiens, les bayésiens naïfs multinomiaux et les bayésiens naïfs de Bernoulli.
- **Bayes naïf gaussien:** suppose que les données de chaque entité sont normalement distribuées. Il est utilisé pour les données continues.
- **Bayes naïf multinomial:** suppose que les données de chaque entité sont distribuées de manière multinomiale. Il est utilisé pour les données discrètes telles que la classification de texte.
- **Bernoulli Naive Bayes:** suppose que les données de chaque fonctionnalité sont distribuées par Bernoulli. Il est utilisé pour les données binaires.

Naïve Bayes entraînement

- Le processus d'entraînement pour Naive Bayes implique l'estimation des paramètres des distributions de probabilité pour chaque caractéristique et classe, généralement en utilisant l'estimation du maximum de vraisemblance.
- Une fois le modèle formé, faire des prédictions pour les nouvelles instances est relativement efficace, car il s'agit simplement de calculer les probabilités conditionnelles pour chaque classe en fonction des valeurs de fonctionnalité de la nouvelle instance et de choisir la classe avec la probabilité la plus élevée.

Naive Bayes

	Spam	No spam
Total	25	75
Buy	20	5
Cheap	15	10
Buy & Cheap	12	$\frac{2}{3}$

Naive Bayes

$P(\text{"Buy"} \ \& \ \text{"Cheap"}) = P(\text{"Buy"}) \ P(\text{"Cheap"})$

S: Spam

H: Ham (not spam)

B: 'Buy'

Bayes Theorem

$$P(S|B) = \frac{P(B|S) P(S)}{P(B|S) P(S) + P(B|H) P(H)}$$

S: Spam
H: Ham (not spam)
B: 'Buy'
C: 'Cheap'

Naive Bayes

$$P(S | B \cap C) = \frac{P(B|S)P(C|S) P(S)}{P(B|S)P(C|S) P(S) + P(B|H)P(C|H) P(H)}$$

S: Spam

H: Ham (not spam)

B: 'Buy'

Bayes Theorem

$$P(S|B) = \frac{P(B|S)P(S)}{P(B|S)P(S) + P(B|H)P(H)}$$

$$\frac{20}{25} \quad \frac{25}{100}$$

$$P(\text{spam if "Buy"}) = \frac{\frac{20}{25} \quad \frac{25}{100}}{\frac{20}{25} \quad \frac{25}{100} \quad + \quad \frac{5}{75} \quad \frac{75}{100}} = 80\%$$

S: Spam
H: Ham (not spam)
B: 'Buy'
C: 'Cheap'

Naive Bayes

$$P(S|B \cap C) = \frac{P(B|S)P(C|S)P(S)}{P(B|S)P(C|S)P(S) + P(B|H)P(C|H)P(H)}$$

$$\frac{20}{25} \frac{15}{25} \frac{25}{100}$$

$$P(\text{spam if "Buy" \& "Cheap"}) = \frac{\frac{20}{25} \frac{15}{25} \frac{25}{100}}{\frac{20}{25} \frac{15}{25} \frac{25}{100} + \frac{5}{75} \frac{10}{75} \frac{75}{100}} = 94.737\%$$

Avec trois

Naive Bayes

	Spam	No Spam		
Total	25		75	
Buy	20	4/5	5	1/15
Cheap	15	3/5	10	2/15
Work	5	1/5	30	6/15
Buy, Cheap, & Work		12/125		

Naive Bayes

	Spam	No Spam	
Total	25	75	
Buy	20	4/5	5
Cheap	15	3/5	10
Work	5	1/5	30
Buy, Cheap, & Work	12/5	12/125	

Naive Bayes

	Spam		No Spam	
Total	25		75	
Buy	20	4/5	5	1/15
Cheap	15	3/5	10	2/15
Work	5	1/5	30	6/15
Buy, Cheap, & Work	12/5	12/125	4/15	12/3375

Naive Bayes

	Spam		No Spam	
Total	25		75	
Buy	20	4/5	5	1/15
Cheap	15	3/5	10	2/15
Work	5	1/5	30	6/15
Buy, Cheap, & Work	12/5	12/125	4/15	12/3375

$$\frac{12/5}{12/5 + 4/15} = \frac{36}{40} = 90\%$$

S: Spam
H: Ham (not spam)
B: ‘Buy’
C: ‘Cheap’

Naive Bayes

$$P(S|B \cap C) = \frac{P(B|S)P(C|S)P(S)}{P(B|S)P(C|S)P(S) + P(B|H)P(C|H)P(H)}$$

$$\frac{20}{25} \frac{15}{25} \frac{25}{100}$$

$$P(\text{spam if } \text{“Buy” \& “Cheap”}) = \frac{\frac{20}{25} \frac{15}{25} \frac{25}{100}}{\frac{20}{25} \frac{15}{25} \frac{25}{100} + \frac{5}{75} \frac{10}{75} \frac{75}{100}}$$

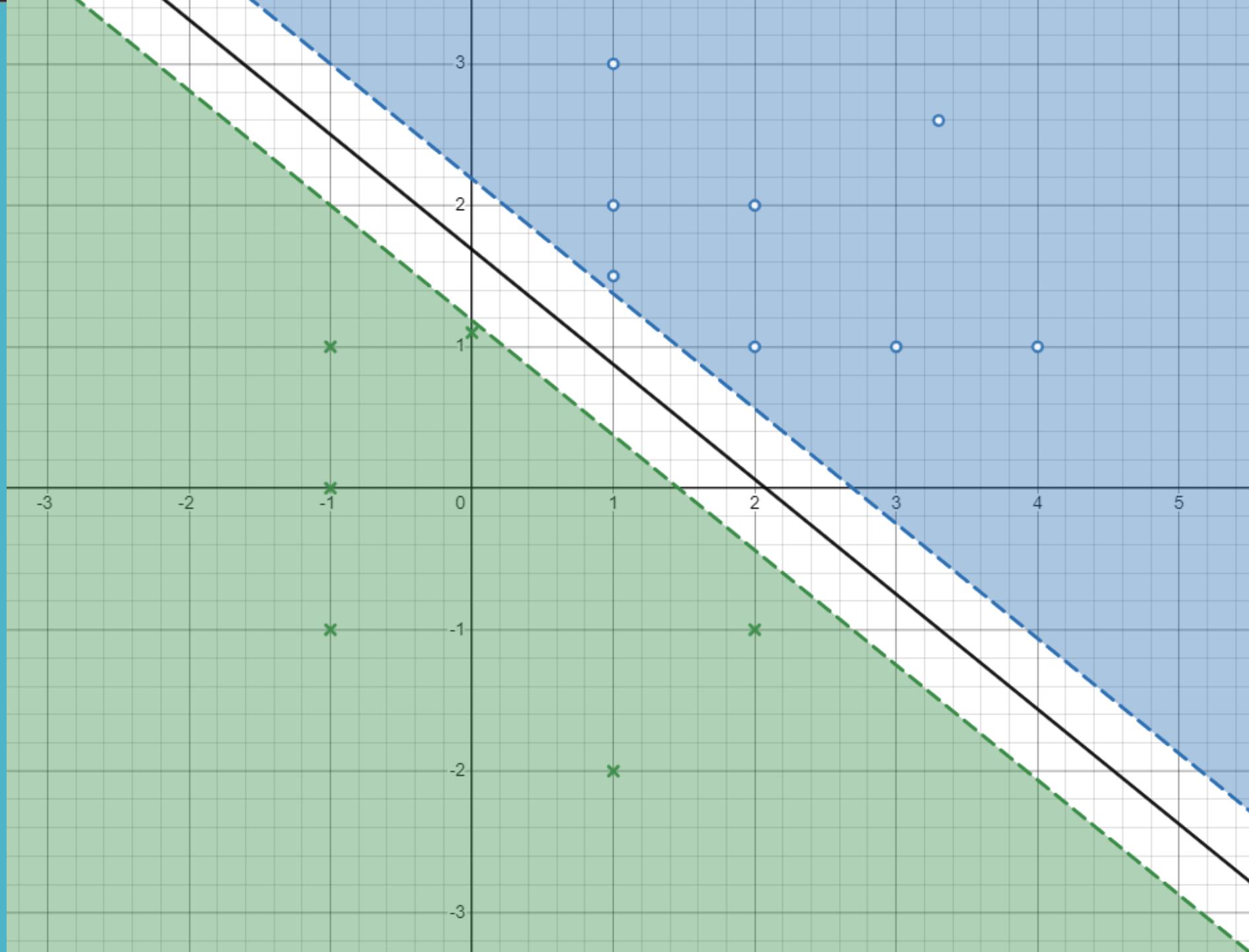
Support Vector Machines

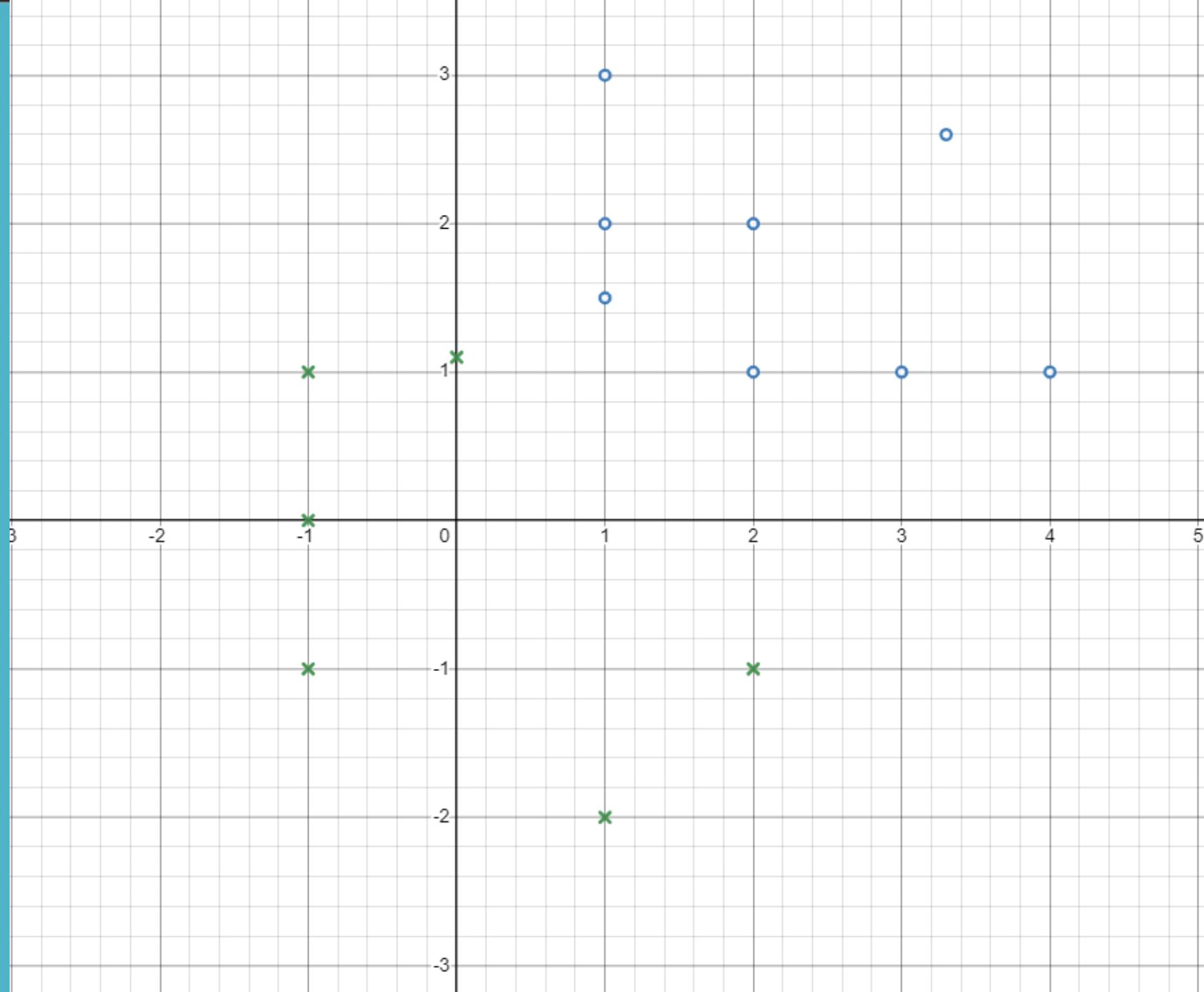
SVM Definition

- Les machines à vecteurs de support (SVM) sont un type d'algorithme d'apprentissage supervisé qui peut être utilisé pour des tâches de classification ou de régression.
- L'idée de base derrière les SVM est de trouver un hyperplan qui sépare au maximum les différentes classes dans les données.

SVM linéaire

- La première étape de la création d'une SVM linéaire consiste à choisir une fonction du noyau.
- Le noyau le plus couramment utilisé est le noyau linéaire, qui calcule simplement le produit ponctuel entre les vecteurs d'entrée.
- D'autres fonctions du noyau, telles que les noyaux de fonctions de base polynomiale et radiale (RBF), peuvent également être utilisées.





Formation du modèle

- Once the kernel is selected, the SVM algorithm will find the hyperplane that maximally separates the different classes in the data.
- The hyperplane is chosen so that it has the largest margin, which is the distance between the hyperplane and the closest data points from each class.
- These closest points are called the support vectors.

Limites de la décision

- La limite de décision d'une SVM linéaire est un hyperplan qui sépare les différentes classes dans les données.
- Les points d'un côté de la frontière appartiennent à une classe et les points de l'autre côté appartiennent à l'autre classe.

SVM non linéaire

- Les SVM non linéaires utilisent une astuce du noyau pour transformer les données d'entrée en un espace de dimension supérieure, où une limite linéaire peut séparer les classes.
- Certains des noyaux populaires sont:
- Noyau polynomial
- Noyau de la fonction de base radiale (RBF)
- Noyau sigmoïde

Astuce du noyau SVM (1)

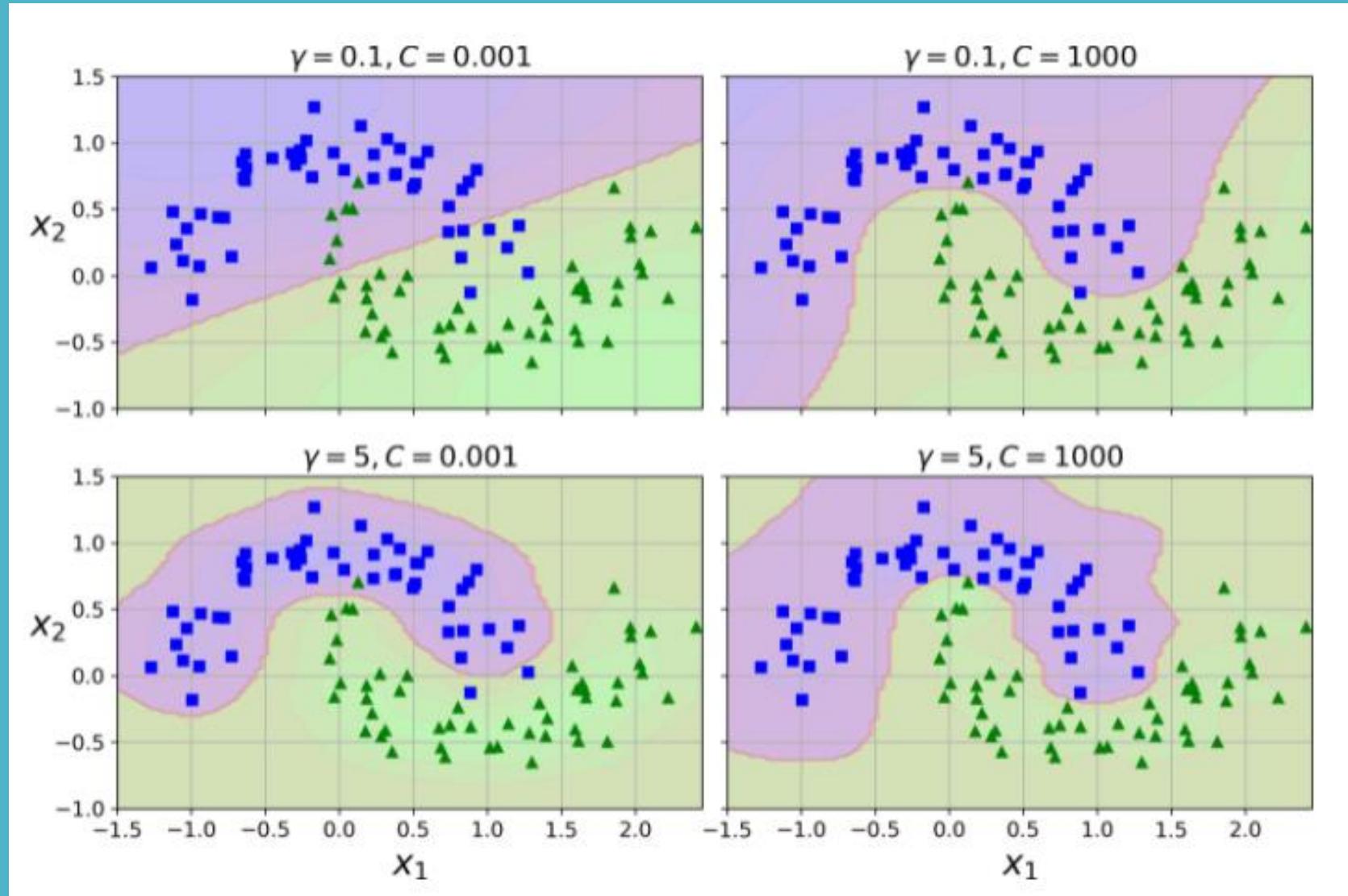
- L'astuce du noyau est une technique mathématique utilisée dans les machines à vecteurs de support (SVM) pour transformer les données d'entrée en un espace de dimension supérieure, où une frontière linéaire peut séparer les classes.
- L'idée principale derrière l'astuce du noyau est de mapper les données d'entrée dans un espace d'entités de dimension supérieure, où les données deviennent linéairement séparables.
- L'astuce du noyau permet aux SVM d'apprendre une limite de décision non linéaire sans calculer explicitement les coordonnées des données dans l'espace de dimension supérieure.
- Plusieurs fonctions de noyau peuvent être utilisées dans les SVM, telles que les noyaux linéaires, polynomiaux et de fonction de base radiale (RBF).
- Le noyau linéaire calcule simplement le produit ponctuel entre les vecteurs d'entrée et est utilisé pour les problèmes de classification linéaire et de régression.
- Le noyau polynomial élève le produit ponctuel à une puissance, qui peut être utilisée pour apprendre les limites de décision non linéaires.
- Le noyau RBF est un type de fonction de base radiale qui calcule la distance entre les vecteurs d'entrée, qui peut être utilisée pour apprendre les limites de décision non linéaires.

Astuce du noyau SVM (2)

- L'astuce du noyau est particulièrement utile lorsque les données ne sont pas séparables linéairement dans leur espace d'entités d'origine, car elles permettent aux SVM d'apprendre une limite de décision non linéaire sans avoir besoin de calculer explicitement les coordonnées des données dans un espace de dimension supérieure.
- L'astuce du noyau rend les SVM efficaces sur le plan informatique, car elle évite d'avoir à effectuer le mappage explicite des données dans un espace de dimension supérieure.
- Il est important de noter que le choix de la fonction noyau peut grandement affecter les performances du modèle SVM, et la meilleure fonction du noyau dépendra du problème spécifique et des caractéristiques des données.
- Les méthodes de réglage des hyperparamètres telles que la validation croisée peuvent être utilisées pour sélectionner la fonction optimale du noyau et ses paramètres.

Réglage des hyperparamètres

- Une chose importante à noter est que les SVM ont plusieurs hyperparamètres qui peuvent être réglés pour améliorer les performances du modèle, tels que le paramètre de régularisation et les paramètres du noyau.
- Ces hyperparamètres peuvent être sélectionnés à l'aide de techniques telles que la validation croisée.

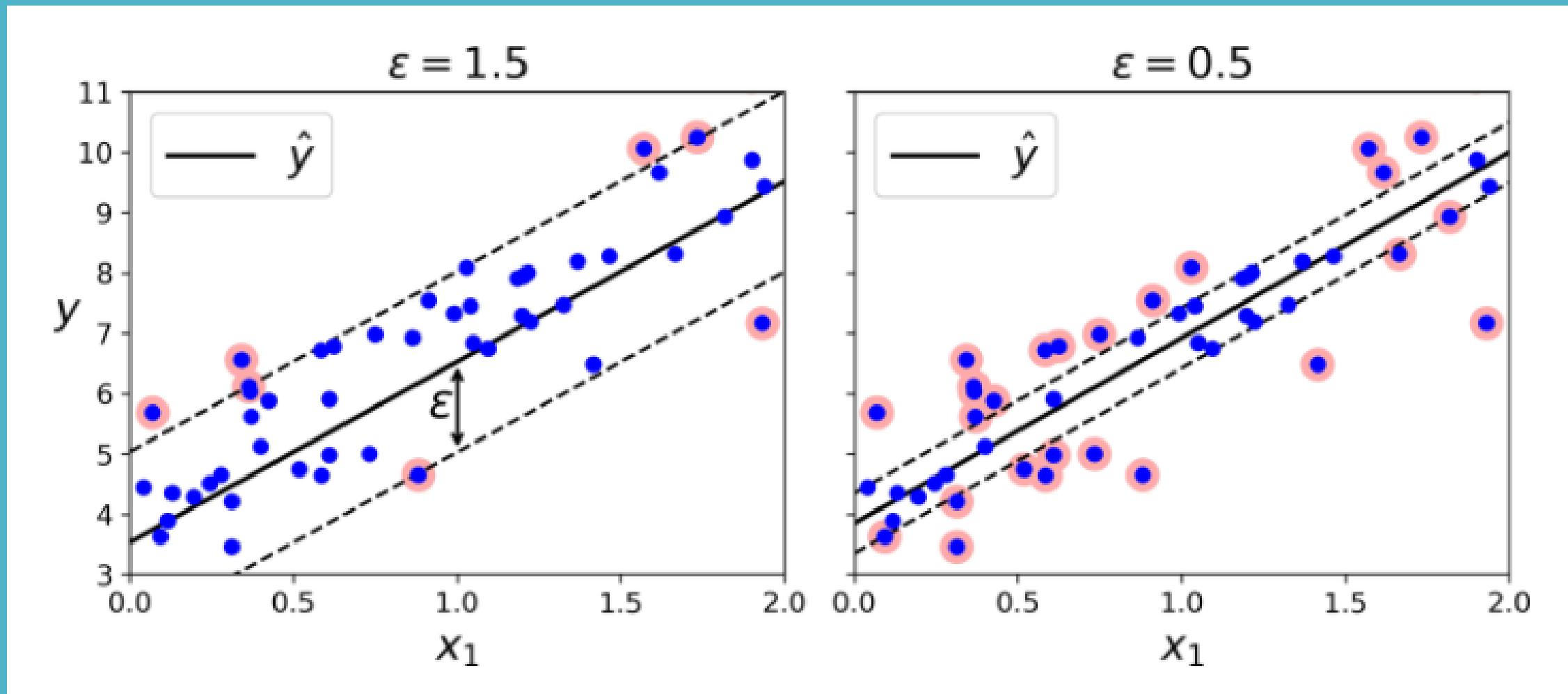


Évaluation de la SVM

- L'évaluation des modèles SVM peut être effectuée en utilisant des mesures d'évaluation courantes telles que l'exactitude, la précision, le rappel et le score F1 pour les problèmes de classification et l'erreur quadratique moyenne (MSE) pour les problèmes de régression.

Régression SVM

- La régression SVM (Support Vector Machine) est une extension de l'algorithme SVM utilisée pour les problèmes de régression, plutôt que pour les problèmes de classification.
- L'idée principale derrière la régression SVM est de trouver une fonction qui se rapproche le mieux des données en minimisant l'erreur entre les valeurs prédites et les valeurs réelles
- La fonction objective de régression SVM est similaire à celle de la classification SVM, mais avec une légère modification :
- Dans la régression SVM, l'objectif est de minimiser l'erreur entre les valeurs prédites et les valeurs réelles, tout en s'assurant que les erreurs sont inférieures à un seuil spécifié, appelé epsilon.



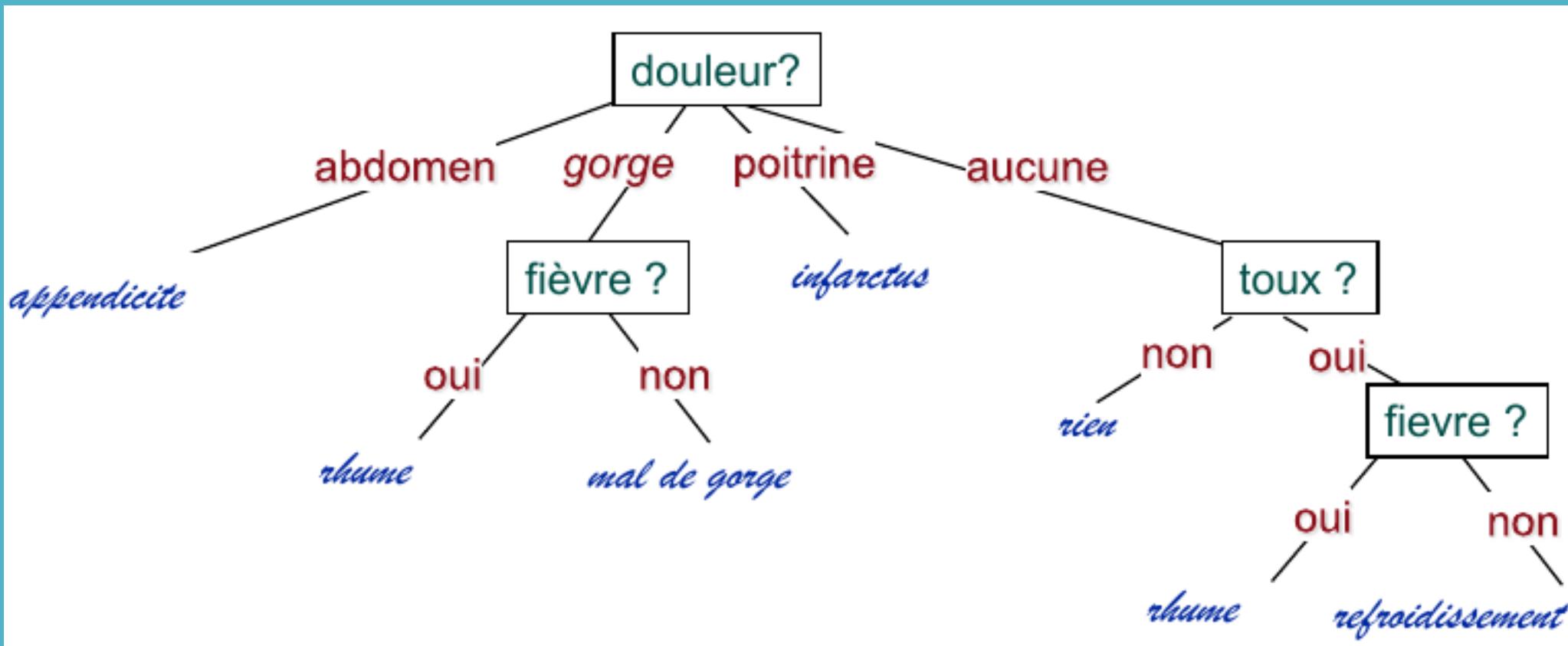
Résumé de la SVM

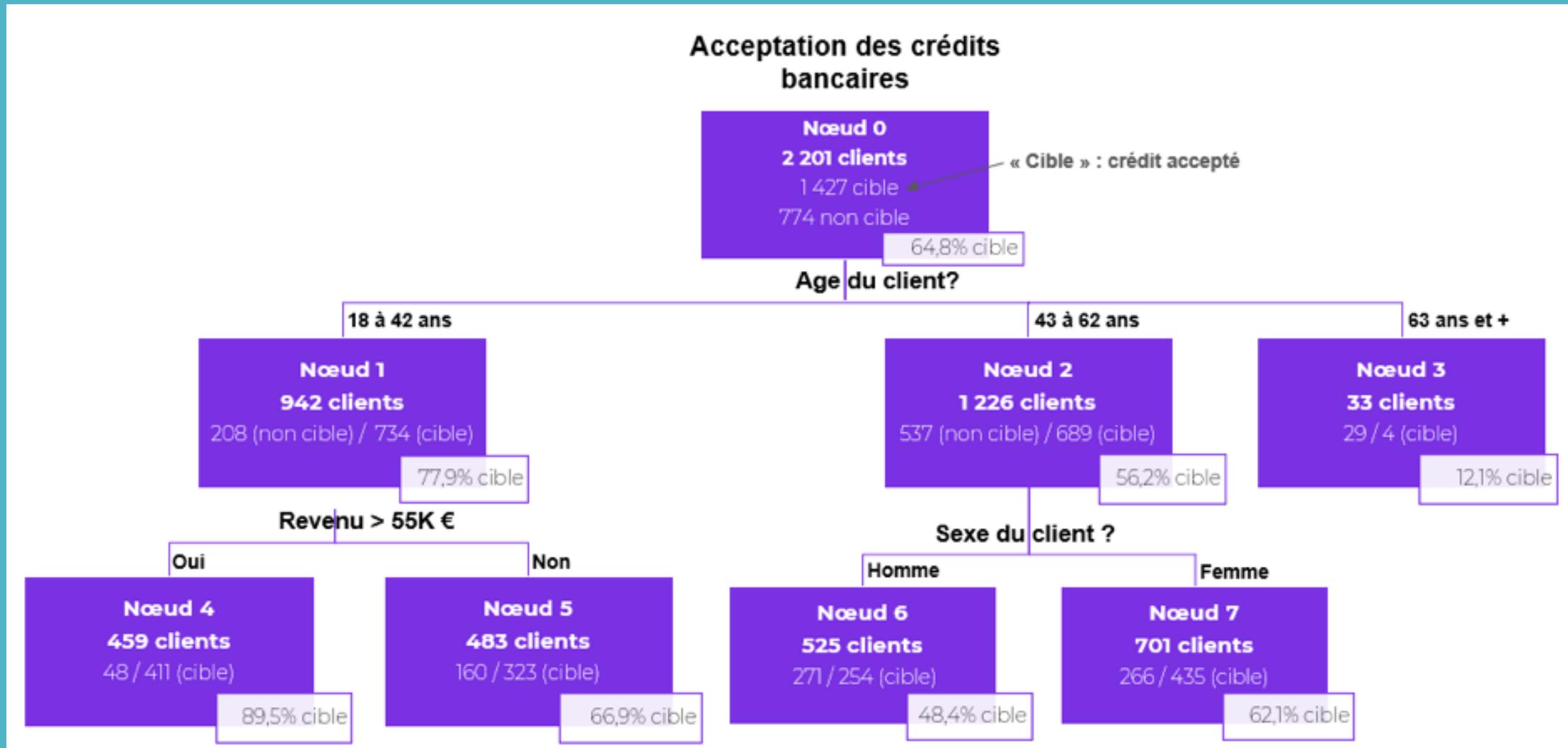
- Les SVM sont des algorithmes puissants pour les données linéaires et non linéaires, mais ce n'est pas toujours la meilleure méthode pour chaque problème, il est important d'évaluer les performances du modèle sur un ensemble de test retenu et de le comparer à d'autres modèles.

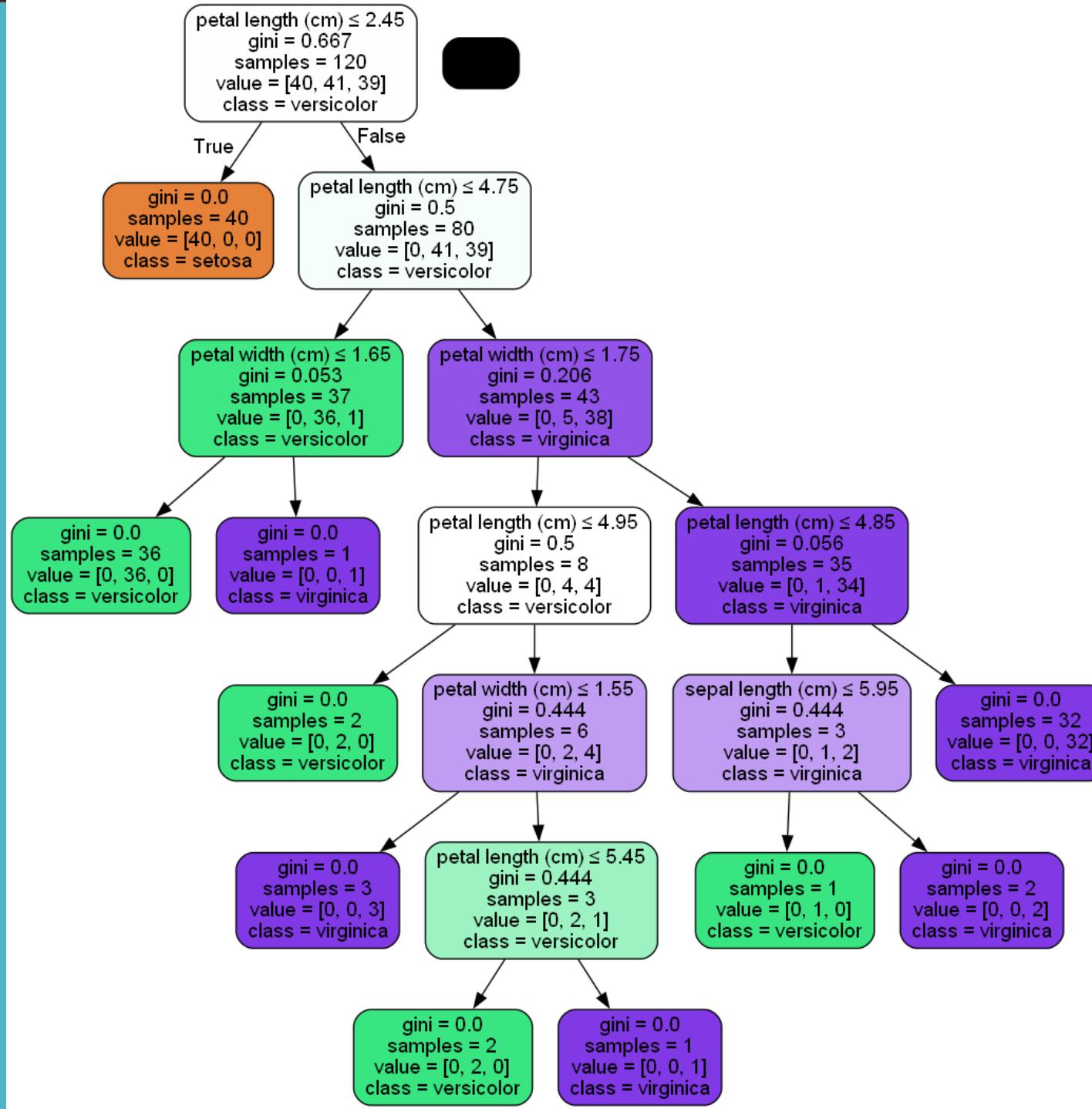
Arbres de décision

Arbre de décision - Définition

- Un arbre de décision est un modèle arborescent qui représente un ensemble de décisions et leurs conséquences possibles.
- Il s'agit d'un algorithme d'apprentissage automatique supervisé qui peut être utilisé à la fois pour les problèmes de classification et de régression.
- Dans un arbre de décision, chaque nœud interne représente un test sur un attribut, chaque branche représente le résultat du test et chaque nœud feuille représente une étiquette de classe ou une valeur numérique.
-







Applications des arbres de décision

- Les arbres de décision sont applicables dans un large éventail de domaines tels que la finance, la santé, le marketing et bien d'autres.
- Quelques exemples incluent la notation de crédit, la détection de la fraude, le diagnostic de maladies, la segmentation de la clientèle et la prévision du cours des actions.
-

Arbres de décision dans la classification et la régression

- Les arbres de décision sont utilisés à la fois dans l'apprentissage automatique de classification et dans les régressions

Classification de l'arbre de décision

- Pour la classification, un arbre de décision est construit en partitionnant récursivement les données en sous-ensembles aussi homogènes que possible en termes de variable cible.
- L'algorithme le plus couramment utilisé pour cela s'appelle l'algorithme ID3, qui utilise le critère de gain d'information pour déterminer le meilleur attribut pour diviser les données.

Régression de l'arbre de décision

- Les arbres de décision peuvent également être utilisés pour les problèmes de régression, où l'objectif est de prédire une variable de sortie continue. Dans les arbres de régression, la valeur de sortie est prédite par la moyenne de la variable cible pour les échantillons dans le nœud feuille.
- L'idée de base de l'algorithme est la même que pour les arbres de classification, mais au lieu de diviser les données en fonction de la mesure des impuretés, l'algorithme divise les données en fonction de la réduction de la variance. La réduction de la variance est mesurée par l'erreur quadratique moyenne (EMS), qui est la somme des différences au carré entre les valeurs prédites et les valeurs réelles.
- L'algorithme CART peut également être utilisé pour les arbres de régression, avec quelques modifications à la mesure des impuretés et au critère de fractionnement. Dans les arbres de régression, la mesure des impuretés est la variance de la variable cible, qui est la somme des différences au carré entre la variable cible et sa moyenne. L'indice de Gini et l'entropie ne sont pas utilisés dans les arbres de régression.
- Le critère de division dans les arbres de régression est la réduction de la variance, qui est la différence entre la variance du nœud parent et la somme pondérée des variances des nœuds enfants. Les poids sont les proportions des échantillons dans chaque nœud enfant.
-

Algorithme des arbres de classification et de régression (CART)

- L'algorithme le plus couramment utilisé pour cela s'appelle l'algorithme CART, qui utilise l'indice de Gini pour déterminer le meilleur attribut pour diviser les données.

L'algorithme CART fonctionne dans les étapes suivantes

1. Choisissez le meilleur attribut pour fractionner les données. Cela se fait en calculant l'indice de Gini pour chaque attribut et en choisissant l'attribut avec le plus petit indice de Gini. L'indice de Gini mesure l'impureté d'un ensemble de données, qui est définie comme la probabilité de mal classer un élément choisi au hasard dans l'ensemble de données s'il est étiqueté au hasard en fonction de la distribution des étiquettes dans le sous-ensemble.
2. Divisez les données en deux sous-ensembles en fonction de la valeur de l'attribut choisi. Le sous-ensemble qui contient les valeurs de l'attribut est considéré comme le sous-ensemble « gauche », et le sous-ensemble qui ne contient pas les valeurs de l'attribut est considéré comme le sous-ensemble « droit ».
3. Appliquez de manière récursive les étapes 1 et 2 à chaque sous-ensemble jusqu'à ce qu'un critère d'arrêt soit rempli. Le critère d'arrêt pourrait être, par exemple, une profondeur maximale de l'arbre, un nombre minimum d'échantillons dans un nœud foliaire ou une réduction minimale des impuretés.
4. Taillez l'arbre pour éviter de trop l'ajustement. Pour ce faire, supprimez les nœuds qui n'améliorent pas les performances de l'arborescence sur un jeu de validation. Les performances de l'arbre sont généralement mesurées par une mesure telle que l'exactitude, la précision, le rappel ou l'erreur quadratique moyenne.

Algorithme CART avec attributs catégoriels

- Pour les attributs catégoriels, l'indice de Gini est calculé comme suit:

$$Gini(D) = 1 - \sum_{i=1}^k p_i^2$$

où D est l'ensemble de données, k est le nombre de classes et p_i est la probabilité de classe i.

Algorithme CART avec attributs numériques

- Pour les attributs numériques, l'algorithme CART fonctionne en triant les valeurs de l'attribut dans l'ordre croissant et en calculant l'indice de Gini pour tous les points de division possibles.
- Le point de division qui donne le plus petit indice de Gini est choisi comme le meilleur split. L'indice de Gini est calculé comme suit:

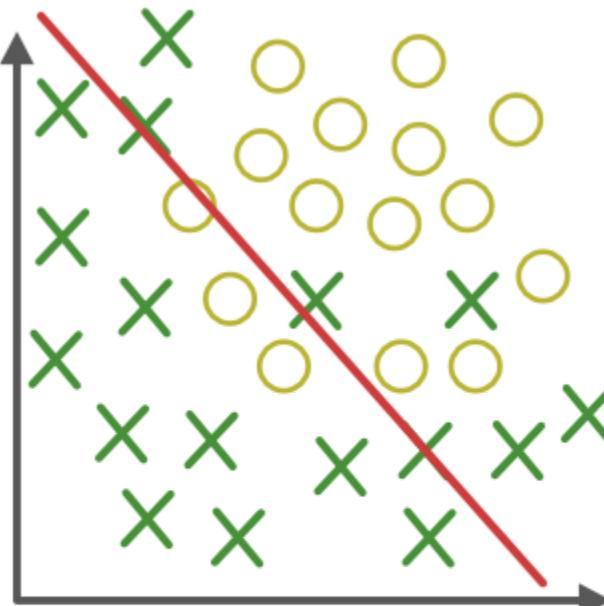
$$Gini(D, a, t) = (|D_1|/|D|) * Gini(D_1) + (|D_2|/|D|) * Gini(D_2)$$

où D est l'ensemble de données, k est le nombre de classes et p_i est la probabilité de classe i.

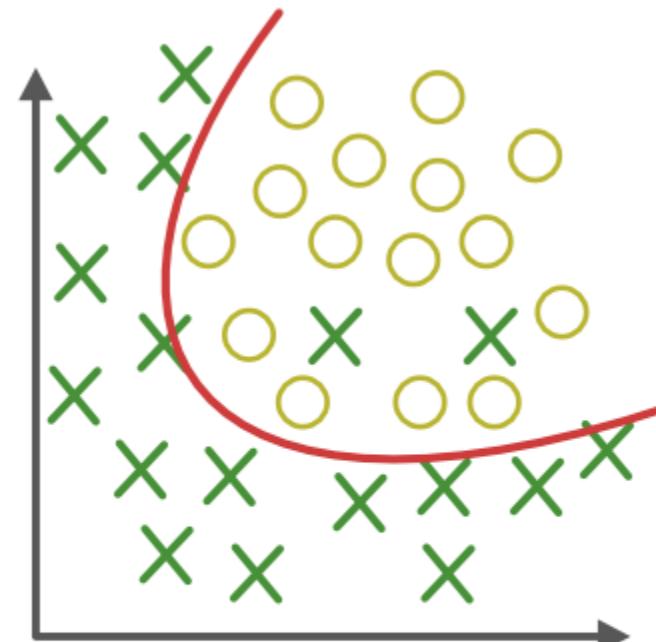
Gestion des valeurs manquantes de l'algorithme CART

- L'algorithme CART peut gérer les valeurs manquantes à l'aide de fractionnements de substitution.
- Une division de substitution est une division utilisée comme sauvegarde lorsque la division principale ne peut pas être utilisée en raison de valeurs manquantes.

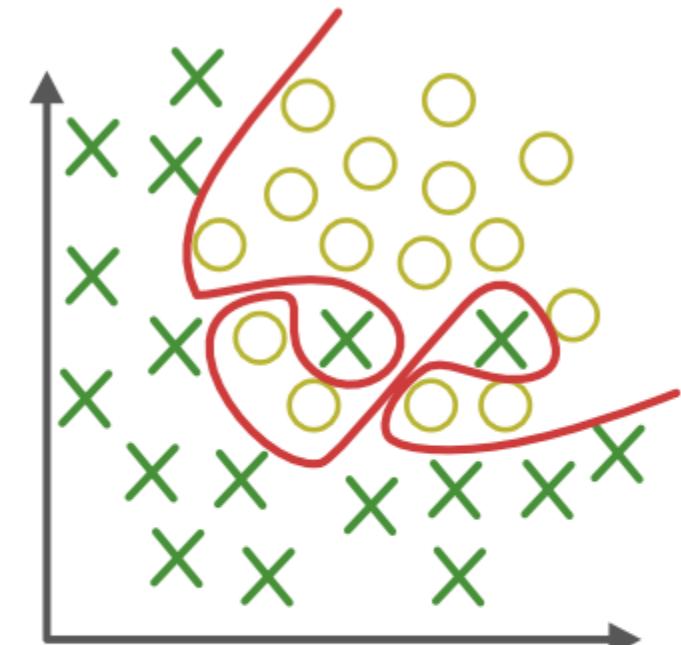
Sur-ajustement vs sous-ajustement



Under-fitting
(too simple to explain the variance)



Appropriate-fitting



Over-fitting
(forcefitting--too good to be true) DG

Sur-ajustement vs sous-ajustement

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			
Classification illustration			
Deep learning illustration			

Sous-ajustement

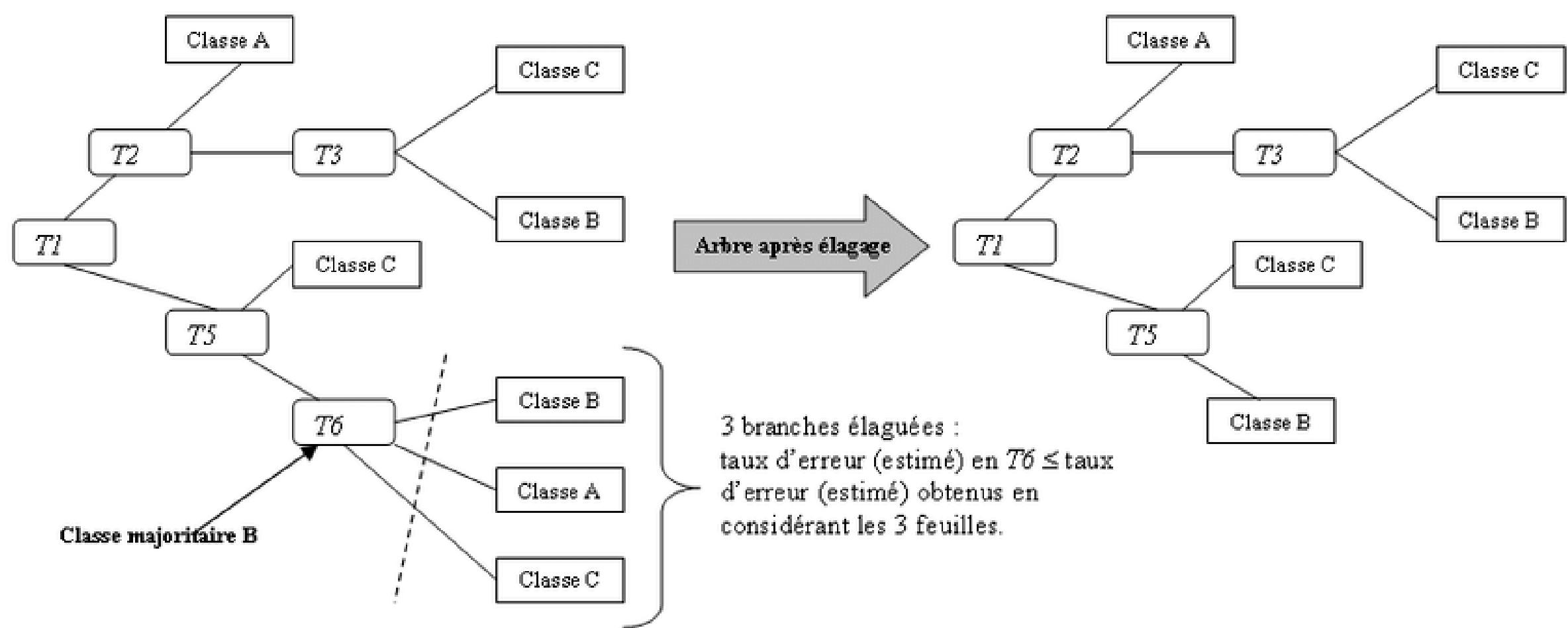
- Le sous-ajustement est un problème courant dans les algorithmes d'arbre de décision, y compris l'algorithme CART. Le sous-ajustement se produit lorsque l'arborescence est trop simple et ne parvient pas à capturer les modèles sous-jacents dans les données. Cela peut entraîner un manque de précision à la fois sur les données d'entraînement et sur les nouvelles données invisibles.
- Le sous-ajustement est souvent causé par des méthodes de pré-taille qui empêchent l'arbre de pousser trop tôt ou par l'utilisation d'un critère d'arrêt trop strict. Par exemple, la définition d'une faible profondeur maximale pour l'arbre ou d'un nombre minimum élevé d'échantillons dans un nœud feuille peut entraîner un arbre trop simple.
- Pour éviter le sous-ajustement, il est important de permettre à l'arbre de grandir à sa taille optimale et d'utiliser un critère d'arrêt plus détendu. Cela peut être fait en utilisant des méthodes post-élagage, telles que l'élagage réduit les erreurs ou l'élagage de complexité des coûts, qui suppriment les nœuds de l'arbre une fois qu'il a atteint sa taille maximale. Ces méthodes peuvent supprimer les nœuds qui n'améliorent pas la précision de l'arborescence, tout en conservant les nœuds qui capturent les modèles sous-jacents dans les données.
- Une autre façon d'éviter le sous-ajustement consiste à utiliser des algorithmes d'arbre de décision plus complexes, tels que les forêts aléatoires ou l'amplification de gradient, qui peuvent capturer des modèles plus complexes dans les données. Ces algorithmes utilisent plusieurs arbres de décision, chacun étant formé sur un sous-ensemble de données, et combinent leurs prédictions pour améliorer la précision du modèle.
-

Sur-ajustement

- Le surajustement est un problème courant dans les algorithmes d'arbre de décision, y compris l'algorithme CART.
- Le surajustement se produit lorsque l'arborescence est trop complexe et capture le bruit dans les données, plutôt que les modèles sous-jacents.
- Cela peut entraîner de mauvaises performances de généralisation et une faible précision sur les nouvelles données invisibles.

Élagage

- L'élagage est une technique courante utilisée pour éviter le surajustement dans les arbres de décision.
- L'élagage consiste à supprimer de l'arborescence les nœuds qui n'améliorent pas ses performances de généralisation.
- Il existe plusieurs types de méthodes d'élagage, y compris le pré-élagage et le post-taille.



Pré-élagage

- La pré-élagage est une méthode d'élagage qui empêche l'arbre de pousser avant qu'il ne devienne trop complexe.
- Cela se fait en établissant un critère d'arrêt basé sur une mesure d'impureté, telle que l'indice de Gini ou l'entropie.
- Le critère d'arrêt pourrait être, par exemple, une profondeur maximale de l'arbre, un nombre minimum d'échantillons dans un nœud foliaire ou une réduction minimale des impuretés.
- La pré-élagage est rapide et simple, mais peut entraîner un sous-ajustement si l'arbre n'est pas autorisé à atteindre sa taille optimale.

Post-élagage

- Le post-élagage est une méthode d'élagage qui enlève les nœuds de l'arbre après qu'il ait atteint sa taille maximale.
- Cela se fait en cultivant d'abord l'arbre sur un ensemble d'entraînement, puis en l'élaguant sur un ensemble de validation.
- Le jeu de validation est un sous-ensemble de l'ensemble d'apprentissage qui n'est pas utilisé pour l'apprentissage, mais qui est utilisé pour évaluer les performances de l'arborescence.
- Les performances de l'arbre sont généralement mesurées par une mesure telle que l'exactitude, la précision, le rappel ou l'erreur quadratique moyenne.
- Il existe plusieurs types de méthodes post-élagage, notamment l'élagage à erreur réduite, l'élagage de complexité des coûts et l'élagage de la longueur minimale de description.

Types de méthodes post-élagage (1)

- Réduction de l'élagage des erreurs:
- L'élagage à erreurs réduites est une méthode d'élagage qui supprime de l'arborescence les nœuds qui n'améliorent pas ses performances de généralisation.
- Pour ce faire, supprimez récursivement chaque nœud de l'arborescence et évaluez les performances du sous-arbre résultant sur le jeu de validation.
- Si les performances de la sous-arborescence sont meilleures que celles de l'arborescence d'origine, le nœud est supprimé. Ce processus est répété jusqu'à ce qu'aucun autre nœud ne puisse être supprimé.
-

Types de méthodes post-élagage (2)

- **Complexité des coûts d'élagage**

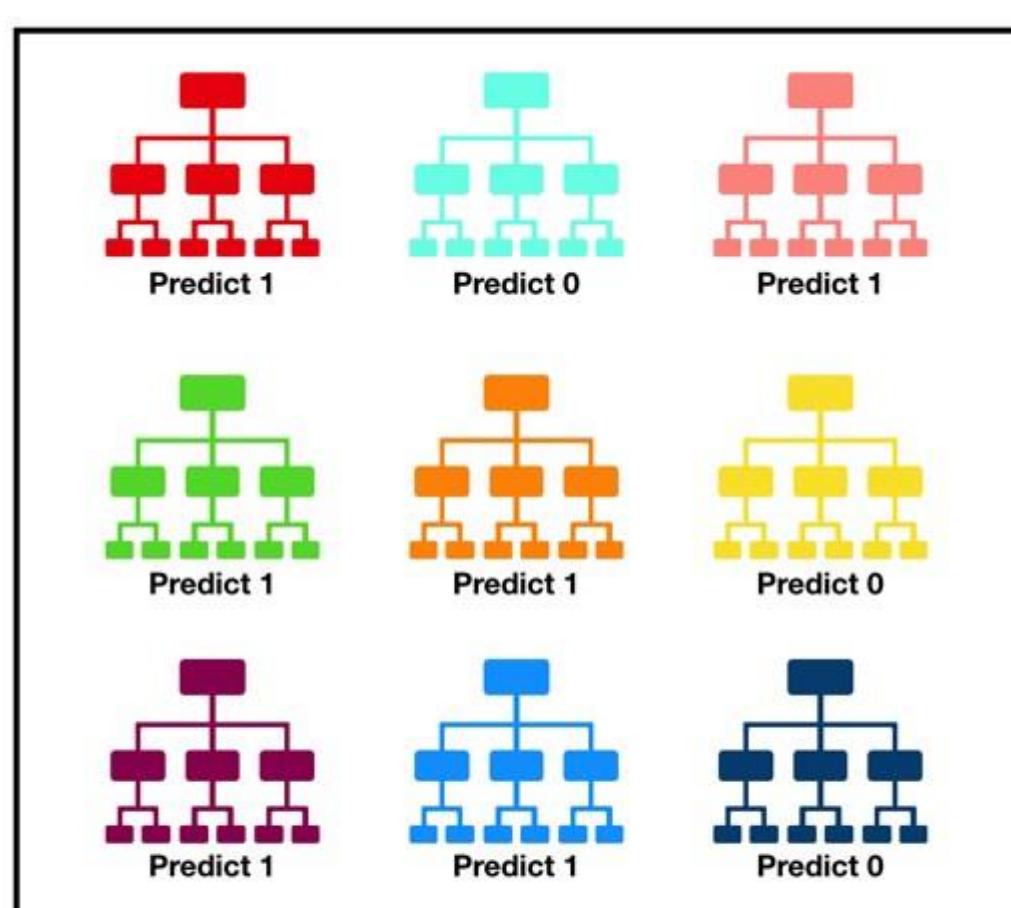
- L'élagage de la complexité des coûts, également connu sous le nom d'élagage de la complexité des coûts minimes, est une méthode d'élagage qui équilibre la complexité et la précision de l'arbre.
- Cela se fait en introduisant un terme de pénalité qui pénalise la complexité de l'arbre, en plus du terme d'erreur qui mesure la précision de l'arbre.
- Le terme de pénalité est contrôlé par un hyperparamètre appelé paramètre de complexité, qui détermine le compromis entre précision et complexité.

Types de méthodes post-élagage (3)

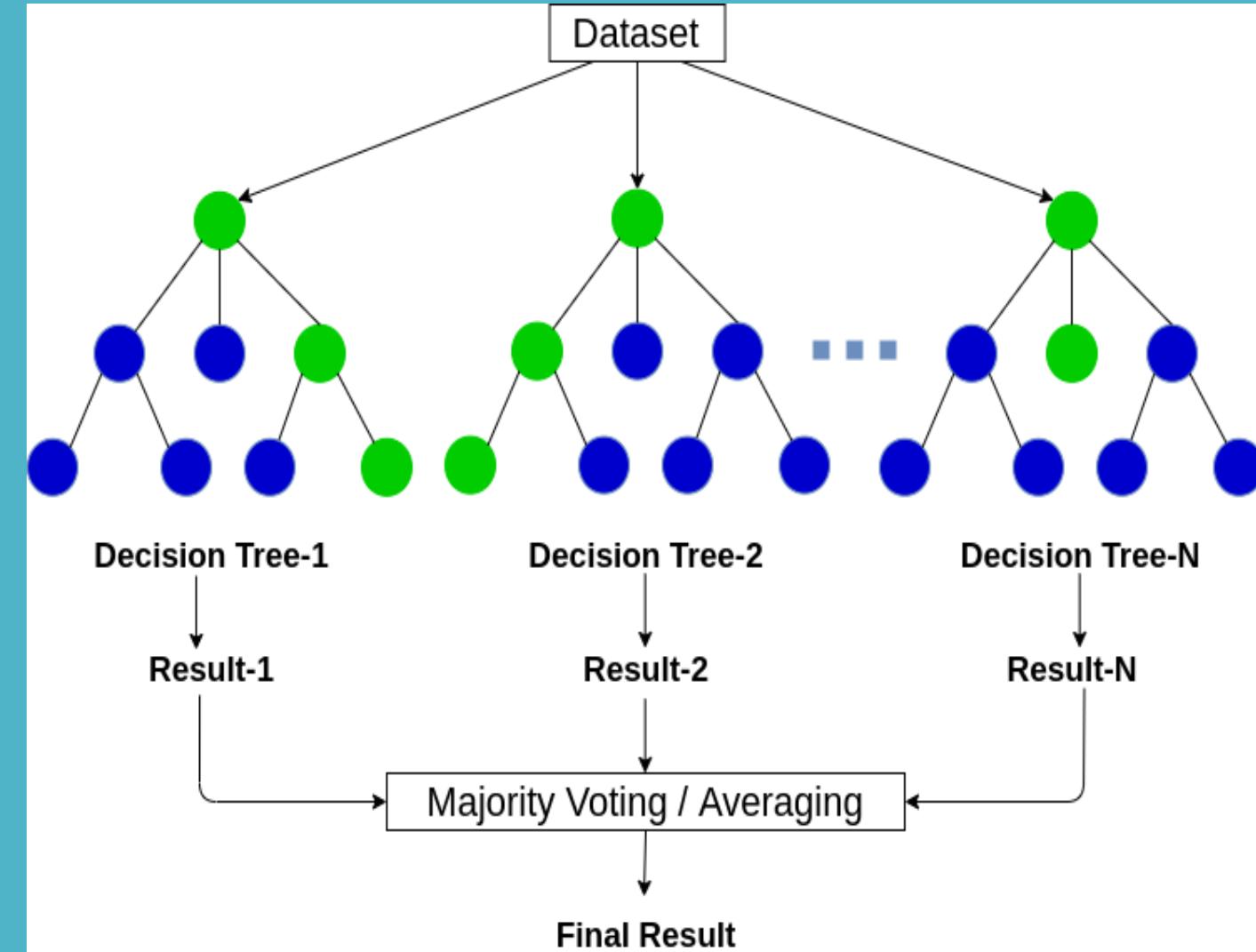
- **Longueur minimale de description élagage**
 - L'élagage de la longueur minimale de description est une méthode d'élagage qui minimise la longueur de description de l'arbre et des données, en partant du principe que l'explication la plus simple est la meilleure.
 - Cela se fait en codant l'arbre et les données à l'aide d'un schéma de codage, tel que le codage de Huffman ou le codage arithmétique, puis en calculant la longueur totale du code.
 - L'arbre est élagué pour minimiser la longueur totale du code, sous réserve d'une contrainte sur la précision de l'arbre.

Méthode de la forêt aléatoire

- Une forêt aléatoire est une collection d'arbres de décision qui sont formés sur des sous-ensembles aléatoires des données d'apprentissage et des sous-ensembles aléatoires des entités. L'idée derrière les forêts aléatoires est d'introduire le caractère aléatoire dans l'algorithme de l'arbre de décision, ce qui peut réduire le surajustement et améliorer les performances de généralisation du modèle.
- L'algorithme de forêt aléatoire fonctionne comme suit :
- Sélectionnez au hasard un sous-ensemble des données d'entraînement, avec remplacement. C'est ce qu'on appelle l'agrégation bootstrap ou l'ensachage.
- Sélectionnez au hasard un sous-ensemble des entités, généralement la racine carrée du nombre total d'entités. C'est ce qu'on appelle l'ensachage de caractéristiques ou la méthode du sous-espace aléatoire.
- Entraînez un arbre de décision sur les données et fonctionnalités sélectionnées.
- Répétez les étapes 1 à 3 pour un nombre fixe de fois, généralement 100 ou plus.
- Faire des prédictions en combinant les prédictions de tous les arbres, soit par vote majoritaire (pour la classification), soit par moyenne (pour la régression).



Tally: Six 1s and Three 0s
Prediction: 1



Avantages de la méthode de la forêt aléatoire

- L'algorithme de forêt aléatoire présente plusieurs avantages par rapport à un seul arbre de décision:
- Il peut gérer un grand nombre de fonctionnalités et de données bruyantes, ce qui peut être problématique pour un seul arbre de décision.
- Il peut capturer des modèles plus complexes dans les données, en combinant les prédictions de plusieurs arbres de décision.
- Il peut réduire le surajustement et améliorer les performances de généralisation du modèle, en introduisant le caractère aléatoire dans l'algorithme de l'arbre de décision.

Applications de la méthode de la forêt aléatoire

- Les forêts aléatoires sont devenues l'un des algorithmes d'apprentissage automatique les plus populaires et sont largement utilisées dans de nombreuses applications, telles que la classification d'images, la bioinformatique et la finance.
- Le package Python sklearn fournit une implémentation simple et facile à utiliser de l'algorithme de forêt aléatoire, qui peut être personnalisé à l'aide d'une gamme d'hyperparamètres, tels que le nombre d'arbres, la profondeur maximale des arbres et le nombre minimum d'échantillons requis pour diviser un nœud.

Régressions

Définition de régression

- En modélisation statistique, la régression est un ensemble de processus statistiques permettant d'estimer les relations entre une variable dépendante (souvent appelée variable « résultat » ou « réponse », ou « étiquette » dans le langage de l'apprentissage automatique) et une ou plusieurs variables indépendantes (souvent appelées « prédicteurs », « covariables », « variables explicatives » ou « caractéristiques »).
- La forme la plus courante d'analyse de régression est la régression linéaire, dans laquelle on trouve la droite (ou une combinaison linéaire plus complexe) qui correspond le mieux aux données selon un critère mathématique spécifique.
- Par exemple, la méthode des moindres carrés ordinaires calcule la ligne unique (ou hyperplan) qui minimise la somme des différences au carré entre les données vraies et cette droite (ou hyperplan).

Pourquoi l'appelle-t-on « régression » ?

- Le terme « régression » a été inventé par Francis Galton au 19ème siècle pour décrire un phénomène biologique.
- Le phénomène était que la taille des descendants des grands ancêtres avait tendance à régresser vers une moyenne normale (un phénomène également connu sous le nom de régression vers la moyenne).
- Pour Galton, la régression n'avait que cette signification biologique, mais son travail a ensuite été étendu par Udny Yule et Karl Pearson à un contexte statistique plus général

Applications de régression

- Prévision des ventes
- Analyse des campagnes marketing
- Prédiction de l'attrition des clients
- Évaluation du risque
- Prévision de prix
- Prévision de la demande
- Prédiction de la rétention des employés

Types de régression

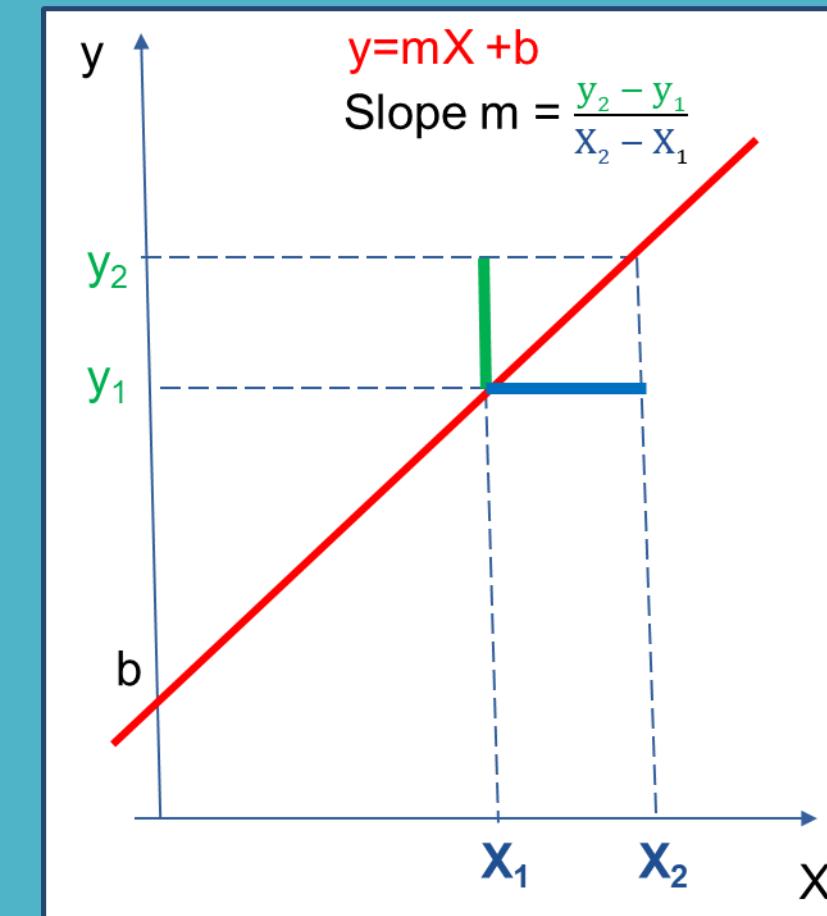
- Régression linéaire simple : La régression linéaire simple est utilisée lorsqu'il n'y a qu'une seule variable prédictive (ou variable indépendante) et une variable de réponse (ou variable dépendante). L'objectif est de trouver une relation linéaire entre la variable prédictive et la variable de réponse.
- Régression linéaire multiple : La régression linéaire multiple est utilisée lorsqu'il y a deux variables prédictives ou plus (ou variables indépendantes) et une variable de réponse (ou variable dépendante). L'objectif est de trouver une relation linéaire entre les variables prédictives et la variable de réponse.
- Régression polynomiale : La régression polynomiale est utilisée lorsqu'il existe une relation non linéaire entre la ou les variables prédictives et la variable de réponse. C'est un type d'analyse de régression dans lequel la relation entre la variable indépendante et la variable dépendante est modélisée comme un polynôme du nième degré.

Types de régressions linéaires

- Régression de crête : La régression de crête est un type de régression linéaire qui ajoute un terme de pénalité à la fonction de coût pour réduire le surajustement. Ce terme de pénalité (également connu sous le nom de régularisation L2) réduit les coefficients vers zéro, réduisant ainsi l'impact de certaines variables prédictives sur le modèle.
- Régression Lasso : La régression Lasso est un autre type de régression linéaire qui ajoute un terme de pénalité à la fonction de coût pour réduire le surajustement. Ce terme de pénalité (également connu sous le nom de régularisation L1) force certains coefficients à être nuls, ce qui élimine efficacement certaines variables prédictives du modèle.
- Régression nette élastique : La régression nette élastique est une combinaison de régression de crête et de lasso qui ajoute des pénalités L1 et L2 à la fonction de coût, réduisant efficacement le nombre de variables prédictives et réduisant l'impact de certaines variables prédictives sur le modèle.
- Régression bayésienne : La régression bayésienne est un type de régression linéaire qui utilise l'inférence bayésienne pour estimer les paramètres du modèle. Il est particulièrement utile lorsque la quantité de données disponibles est limitée et qu'il est nécessaire d'intégrer des connaissances ou des croyances préalables sur les paramètres.

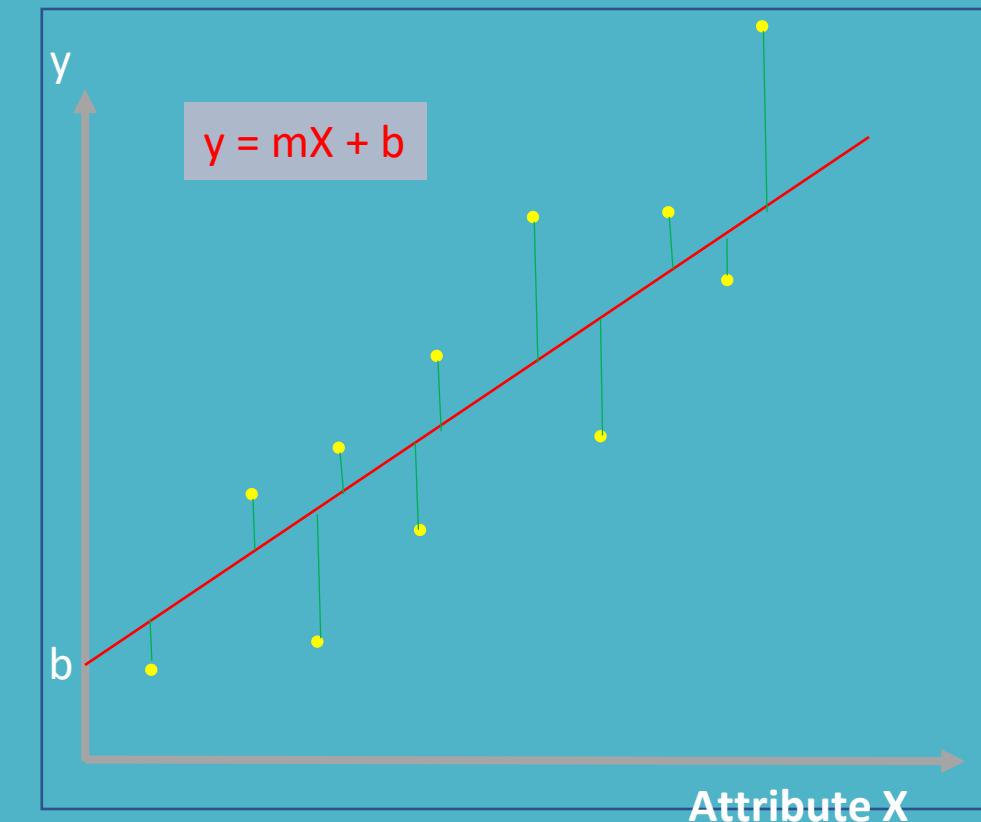
Modèle de régression linéaire simple

- Modèle de régression linéaire simple la relation entre deux variables. Il existe une variable dépendante y et une variable indépendante X telle que $y = \beta_0 + \beta_1 X$ ou plus communément $y = mX + b$
- $y = mX + b$ peut être représenté graphiquement par un tracé linéaire dans l'espace 2D avec X , sur l'axe horizontal et y sur l'axe vertical
- m , la pente de la droite, représente l'inclinaison de la droite, tandis que b , l'intersection y , représente la valeur de y lorsque $x = 0$
- En apprentissage automatique, une régression linéaire simple modélisera une fonction linéaire avec la sortie y en fonction de l'attribut (caractéristique) X



Évaluation de la performance des modèles de régression linéaire

- Les points jaunes représentent les données de l'attribut du jeu de données – paires (x_i, y_i)
- La ligne du meilleur ajustement (représentée en rouge) représente les paires de valeurs estimées (x_i, \hat{y}_i)
- Sont appelés estimations ou prédictions \hat{y}_i
- Les résidus ou les erreurs sont calculés comme $e_i = \hat{y}_i - y_i$
- Nous avons besoin d'une méthode pour évaluer globalement à quel point la ligne est éloignée des données réelles. Un nombre unique qui reflète la qualité de l'ajustement du modèle linéaire
- Voici quelques-unes des méthodes utilisées :
- Erreur absolue moyenne (MAE)
- Erreur quadratique moyenne (MSE)
- Erreur quadratique moyenne (RMSE)
- Pourcentage d'erreur absolu moyen
- Pourcentage moyen d'erreur
- R-carré: R2 aka Coefficient de détermination
- R2 ajusté
-



Erreur absolue moyenne

- L'erreur absolue moyenne (MAE) évalue la qualité de l'ajustement en faisant la moyenne de la valeur absolue des résidus
- La valeur absolue est utilisée pour éviter d'annuler les erreurs positives et négatives conduisant à une erreur artificiellement faible
- L'EMA est directement proportionnelle à l'amplitude des erreurs
- Un problème potentiel : MAE n'est pas une fonction analytique qui peut être utilisée dans des techniques avancées d'apprentissage automatique où la différentiation de l'erreur est nécessaire

$$\text{MAE} = \frac{1}{N} \sum |\hat{y}_i - y_i|$$

Pourcentage d'erreur absolu moyen et pourcentage moyen d'erreur

- Le MAPE modifie la formule MAE afin que l'erreur soit exprimée en pourcentage.
- L'EMT modifie la formule MAPE en supprimant les valeurs absolues. Les résidus positifs et négatifs se compenseront
- L'EMT donne une bonne indication sur la question de savoir si les estimations \hat{y} surestiment ou sous-estiment les valeurs réelles de y
- Remarque : MAPE et MAP ne peuvent pas être utilisés si y_i est égal à zéro.

$$\text{MPE} = \frac{100}{N} \sum (y_i - \hat{y}_i) / y_i$$

in %

$$\text{MAPE} = \frac{100}{N} \sum |(y_i - \hat{y}_i) / y_i|$$

in %

Erreur quadratique moyenne et erreur quadratique moyenne (RMSE)

- L'erreur quadratique moyenne (MSE) évalue la qualité de l'ajustement en faisant la moyenne du carré des résidus ou des erreurs $e = \hat{y}_i - y_i$
- La quadrature des résidus entraîne une pénalité plus importante sur les erreurs plus importantes, ce qui oblige l'algorithme à travailler dur pour minimiser le MSE (idéalement très faible – proche de zéro)
- L'erreur quadratique moyenne racine (RMSE) est essentiellement la même mesure, mais pour éviter de traiter de très grandes valeurs de MSE, nous prenons la racine carrée pour la ramener au même ordre de grandeur que les données originales
- Le RMSE est mathématiquement l'écart-type des résidus – en supposant que la moyenne résiduelle est nulle. RMSE est une mesure de la dispersion des résidus
- Le MSE et le RMSE sont tous deux des fonctions analytiques. Leurs dérivés peuvent facilement être calculés et nous les utilisons dans des algorithmes avancés d'apprentissage automatique

$$\text{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{N}$$

$$\text{RMSE} = \sqrt{\frac{\sum(\hat{y}_i - y_i)^2}{N}}$$

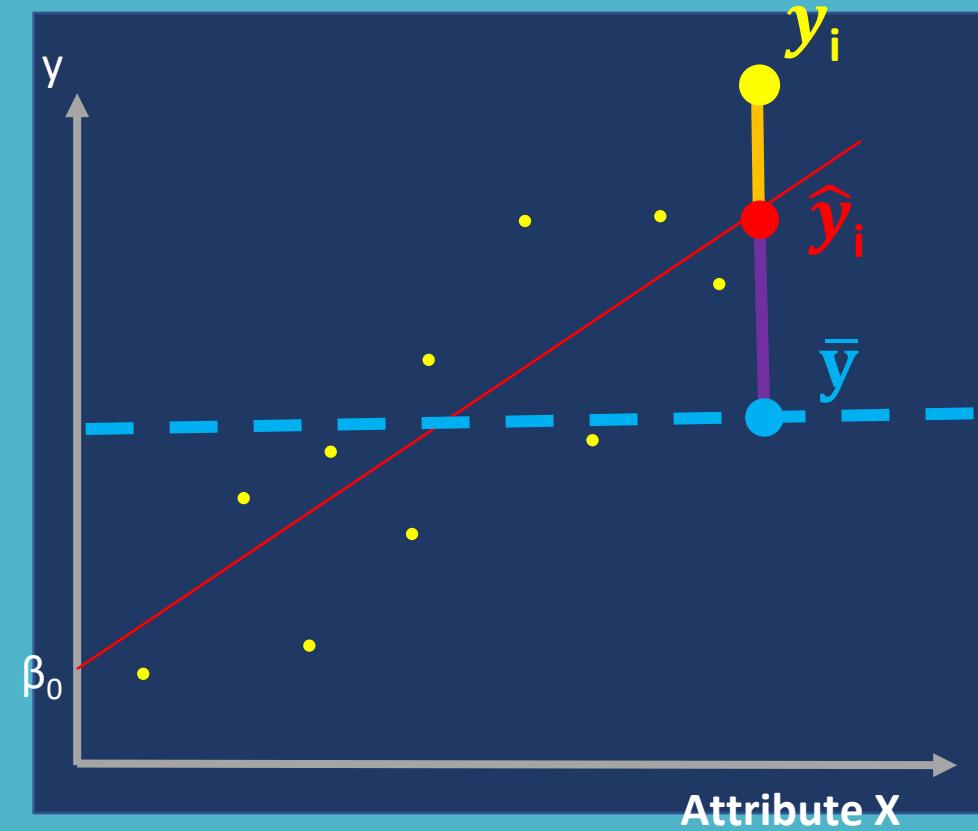
Coefficient de détermination: R-carré (R²)

- Le coefficient de détermination R² fournit une interprétation de la qualité de l'ajustement dans les modèles de régression
- Pour ce faire, il indique le pourcentage de la variance de sortie qui est expliqué par les variables indépendantes X dans $y = \sum \beta_i X_i + \beta_0$
- C'est un moyen très simple et compréhensible de résumer dans quelle mesure le modèle s'adapte aux données.
- Par exemple, si les données tracent le chiffre d'affaires par rapport aux dépenses publicitaires commerciales, un R² de 85% signifie que 85% de la variance des ventes s'explique par l'influence des annonces sur les habitudes d'achat des clients. Cela signifie également que 15% ne peut pas être expliqué par les dépenses publicitaires et est dû au hasard
- R² est le carré du facteur de corrélation R entre y et le X_i
- R² varie entre 0 et 1 (0 % et 100 %)

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

Explication de R²

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} = \frac{SSR}{SST}$$

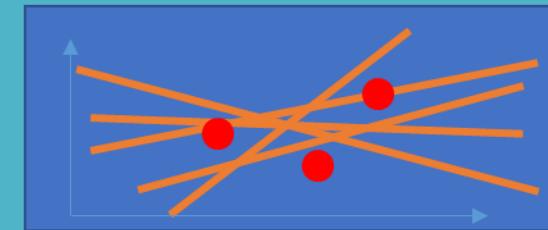


Degrees of freedom (1) Degrés de liberté (1)

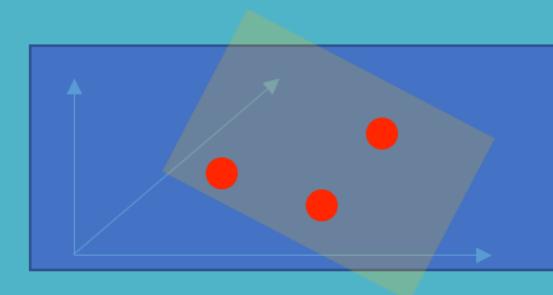
- Lors de l'élaboration d'un modèle de régression ou de tout modèle, il est important de s'assurer que la performance du modèle n'est pas limitée par le nombre d'attributs sélectionnés.
- Si nous commençons avec un jeu de données avec deux points de données (instances) et un seul attribut (caractéristique), nous pouvons voir que R^2 sera toujours égal à 1, car il n'y a qu'une seule droite de régression possible, et elle s'adapte parfaitement.
- Si nous ajoutons un point de données pour un total de 3 points de données et conservons un seul attribut, le modèle a plus de liberté dans les résultats dans peut produire et R^2 tombe en dessous de 1
- Cependant, si nous ajoutons un attribut pour un total de 2 attributs et 3 points, le modèle de régression est à nouveau contraint. Cette fois, un seul plan est possible à travers les trois points et R^2 à 1



#data points = 2
#attributes = 1
Zero degrees of freedom
 $R^2 = 1$



#data points = 3
#attributes = 1
One degrees of freedom
 $R^2 < 1$



#data points = 3
#attributes = 2
Zero degrees of freedom
 $R^2 = 1$

Degrees of freedom (2)

- Generalizing the observations made on the previous slide, we see that the performance of the regression model (as measured by R^2) is impacted by the number of data points and number of attributes used in the model
- The concept of degrees of freedom, df tries to capture the regression model flexibility in modeling the data based on the number points (N) and attributes (k)
- The degrees of freedom formula is $df = N - k - 1$
- For example:
 - a dataset with $N = 100$ points and $k = 25$ attributes has $df = N-k-1 = 100-25-1 = 74$ degrees of freedom
 - If we increase the number of attributes to 60, df drops to $df = 100-60-1 = 39$

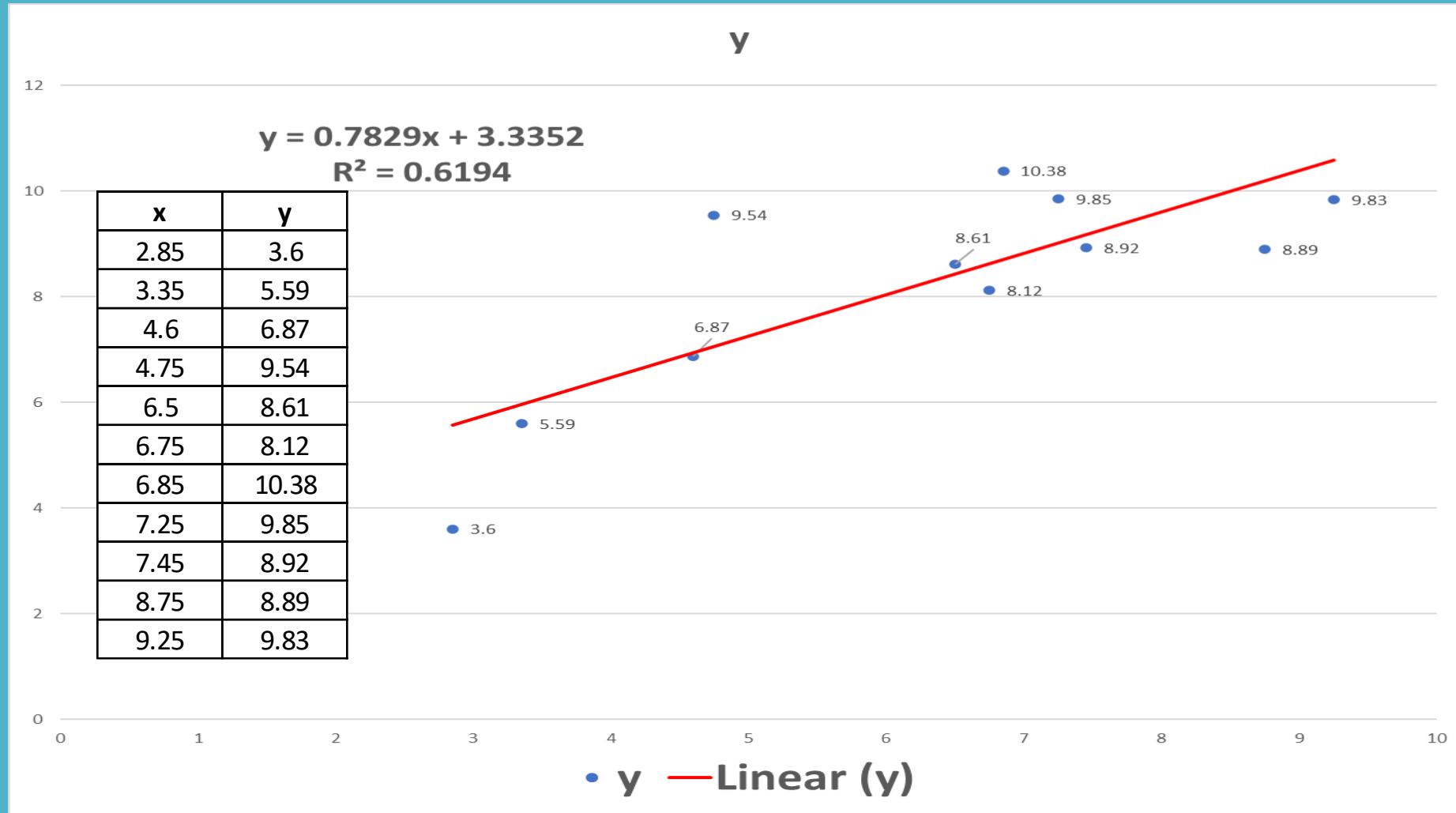
$$df = N - k - 1$$

R-carré ajusté (R²)

- Une observation faite à propos de R² est que l'ajout d'attributs plus indépendants à l'ensemble de données donne des valeurs plus élevées de R², quelle que soit la pertinence des attributs par rapport au problème en question.
- Par exemple, dans l'exemple précédent avec les ventes par rapport aux dépenses publicitaires, si nous ajoutons des mesures de température à l'ensemble de données, le R² passera de 85%. Cela est dû à l'expression mathématique de R² et à la nature statistique de la régression
- Afin de compenser ce problème, nous utilisons la valeur R² ajustée au lieu de la valeur R² en tenant compte du nombre de prédicteurs
- N = nombre d'échantillons (instances)
- k = nombre d'attributs indépendants (entités)
- Note: N - k - 1 = degrés de liberté df
- Dans la formule R² ajustée ci-dessous, k sur le dénominateur agit pour réduire R² à mesure que k croît

$$R^2_{\text{adjusted}} = 1 - \frac{(1-R^2)(N-1)}{N-k-1}$$

Exemple de régression linéaire sur des données



Calcul de MAE, MSE, RMSE pour l'exemple de données

x	y	$\hat{y} = 0.7829x + 3.3352$	Error (residual)	MAE	MSE	RMSE
2.85	3.6	5.566465	1.966465	0.94398	1.481988	1.217369
3.35	5.59	5.957915	0.367915			
4.6	6.87	6.93654	0.06654			
4.75	9.54	7.053975	-2.486025			
6.5	8.61	8.42405	-0.18595			
6.75	8.12	8.619775	0.499775			
6.85	10.38	8.698065	-1.681935			
7.25	9.85	9.011225	-0.838775			
7.45	8.92	9.167805	0.247805			
8.75	8.89	10.185575	1.295575			
9.25	9.83	10.577025	0.747025			

$$\text{MAE} = \frac{1}{N} \sum |\hat{y}_i - y_i|$$

$$\text{MSE} = \frac{\sum (\hat{y}_i - y_i)^2}{N}$$

$$\text{RMSE} = \sqrt{\frac{\sum (\hat{y}_i - y_i)^2}{N}}$$

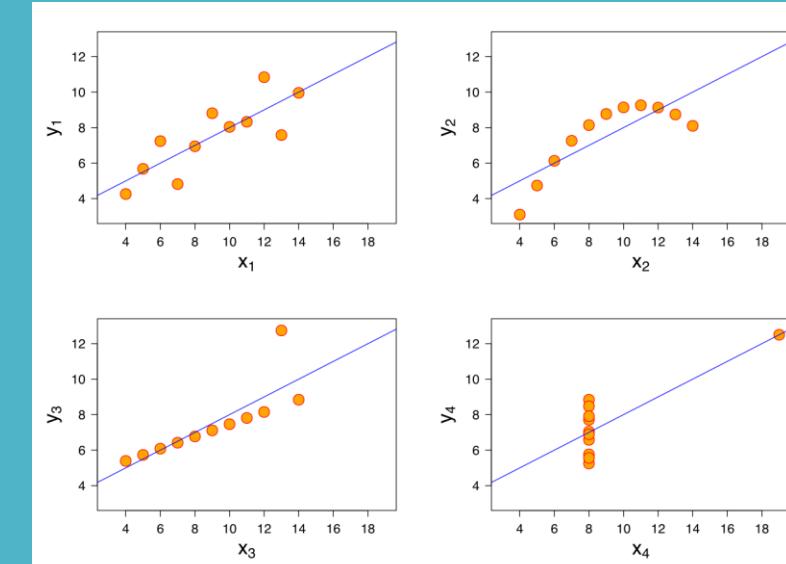
Traditional Ordinary Least Square

- Ordinary Least Square (OLS) est une méthode utilisée pour trouver la droite de régression
- Le MCO utilise le calcul pour minimiser la somme des erreurs au carré (résidus)
- C'est une méthode simple pour trouver la pente et l'intersection y de la droite de régression
- Par exemple : Solution OLS pour $y = mX + b$
- - Slope: $m = \frac{\sum(X_i - \bar{X})(y_i - \bar{y})}{\sum(X_i - \bar{X})^2}$ or $m = \frac{Covariance(X,y)}{Variance(X)}$
 - y-intercept: $b = \bar{y} - m\bar{X}$

Les ensembles de données du Quatuor Anscombe

- Des ensembles de données très différents peuvent produire un diagramme de sortie du modèle de régression similaire et même des mesures de performance (R^2 , moyenne, variance, etc.)
- Référence : Anscombe, F. (1973). Graphiques en analyse statistique. American Statistician, 27:17--21
- À retenir: tracez toujours vos données !

Anscombe's quartet									
I		II		III		IV			
x	y	x	y	x	y	x	y		
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58		
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76		
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71		
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84		
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47		
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04		
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25		
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50		
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56		
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91		
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89		



Machine Learning vs Analyse

- Les MCO (moindres carrés ordinaires) sont une méthode classique utilisée pour résoudre des problèmes de régression linéaire.
- Il s'agit de trouver les valeurs des paramètres du modèle (y compris la pente et le biais) qui minimisent la somme des erreurs au carré entre les valeurs prédites et réelles de la variable de réponse.
- Bien que le SLO soit une méthode puissante et largement utilisée, il présente plusieurs limites :
- L'une des limites est qu'il suppose que la relation entre les variables prédictives et la variable de réponse est linéaire, ce qui n'est pas toujours le cas dans la pratique.
- Une autre limite est qu'elle exige une connaissance des propriétés statistiques des données, telles que la variance et la covariance des variables, qui peuvent être difficiles à estimer avec précision dans certains cas.
- Les algorithmes d'apprentissage automatique, en revanche, peuvent être utilisés pour modéliser des relations plus complexes et non linéaires entre les variables prédictives et la variable de réponse, sans nécessiter de connaissance explicite des propriétés statistiques des données.
- Les algorithmes d'apprentissage automatique peuvent également être utilisés pour gérer des données manquantes ou incomplètes, des valeurs aberrantes et d'autres défis pouvant être présents dans les ensembles de données du monde réel.
- En outre, les algorithmes d'apprentissage automatique peuvent apprendre automatiquement les valeurs optimales des paramètres du modèle en minimisant une fonction de coût, sans nécessiter la sélection manuelle des paramètres du modèle ou des hypothèses sur la distribution des données. Cela permet d'économiser du temps et des efforts, et peut conduire à de meilleures performances de modèle et à des prévisions plus précises.

Régression linéaire

- La régression linéaire est un type d'approche statistique qui vise à établir la relation linéaire entre les variables dépendantes et indépendantes.
- En d'autres termes, il est utilisé pour prédire une variable de résultat continu (Y) basée sur une ou plusieurs variables prédictives (X).
- La régression linéaire simple est également connue sous le nom de régression par les moindres carrés ordinaires ou MCO.

$$y = \beta_0 + \beta_1 x + \epsilon$$

Code de régression linéaire

```
from sklearn.linear_model import LinearRegression

# Initialize the linear regression model
lr = LinearRegression()

# Fit the model to the training data
lr.fit(X_train, y_train)

# Make predictions on the test data
predictions = lr.predict(X_test)
```

Multiple Regression

- La régression multiple est une technique statistique utilisée pour modéliser la relation entre une variable dépendante (aussi appelée variable de réponse) et deux variables indépendantes ou plus (aussi appelées variables prédictives).
- Dans la régression multiple, l'objectif est de trouver une relation linéaire entre la variable dépendante et les variables indépendantes.
-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

- où Y est la variable dépendante, β_0 est l'interception, $\beta_1, \beta_2, \dots, \beta_n$ sont les coefficients (pentes) des variables indépendantes X_1, X_2, \dots, X_n , respectivement, et ϵ est le terme d'erreur.
- Les coefficients des variables indépendantes peuvent être estimés à l'aide de diverses méthodes, y compris la régression des moindres carrés, qui minimise la somme des erreurs au carré entre les valeurs prédites et réelles de la variable dépendante.
- La régression multiple est largement utilisée dans de nombreux domaines, notamment l'économie, la finance, le marketing et les sciences sociales, pour modéliser des relations complexes entre plusieurs variables et faire des prédictions basées sur ces relations. Il peut également être utilisé pour la sélection des variables, pour identifier les prédicteurs les plus importants de la variable dépendante, et pour l'interprétation du modèle, pour comprendre les effets des variables indépendantes sur la variable dépendante.

Régression polynomiale

- La régression polynomiale consiste à modéliser la relation entre la variable indépendante X et la variable dépendante Y à l'aide d'une fonction polynomiale de degré n.
- La formule de régression polynomiale de degré n est donnée par:

$$Y = \beta_0 + \beta_1X + \beta_2X^2 + \cdots + \beta_nX^n + \epsilon$$

- où Y est la variable dépendante, X est la variable indépendante, les bêtas sont les coefficients (pentes) de la fonction polynomiale et epsilon est le terme d'erreur.
- Le degré de la fonction polynomiale est déterminé par la valeur de n.

Code de régression polynomiale

```
# Import the required libraries
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Set the degree of the polynomial
degree = 2

# Create polynomial features and fit the model
poly_features = PolynomialFeatures(degree=degree)

# Train the model
x_poly = poly_features.fit_transform(Xtrain)
poly_reg = LinearRegression()
poly_reg.fit(x_poly, y)

# Predict the response variable for new data
predictions = poly_reg.predict(Xtest)
```

Régression Lasso

- La régression Lasso est un type de régression linéaire utilisé pour éviter le surajustement en ajoutant un terme de pénalité (L1) à la fonction de coût.
- Ce terme de pénalité force certains coefficients à être nuls, ce qui élimine efficacement certaines variables prédictives du modèle.

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \alpha \|\mathbf{w}\|_1$$

Code de régression Lasso

```
from sklearn.linear_model import Ridge

# Initialize the Ridge model
ridge = Ridge(alpha=0.1)

# Fit the model to the training data
ridge.fit(X_train, y_train)

# Make predictions on the test data
predictions = ridge.predict(X_test)
```

Régression de crête

- La régression de crête est un type de régression linéaire utilisé pour éviter le surajustement en ajoutant un terme de pénalité (L2) à la fonction de coût.
- Ce terme de pénalité réduit l'ampleur des coefficients, réduisant ainsi l'impact de certaines variables prédictives sur le modèle.

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \alpha \|\mathbf{w}\|_2^2$$

Code de régression de crête

```
from sklearn.linear_model import Ridge

# Initialize the Ridge model
ridge = Ridge(alpha=0.1)

# Fit the model to the training data
ridge.fit(X_train, y_train)

# Make predictions on the test data
predictions = ridge.predict(X_test)
```

Régression nette élastique

- La régression nette élastique est une combinaison de régression de Lasso et de Ridge
- Il ajoute des pénalités L1 et L2 à la fonction de coût, réduisant efficacement le nombre de variables prédictives et réduisant l'impact de certaines variables prédictives sur le modèle.

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \alpha \left((1 - \lambda) \|\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 \right)$$

Régression logistique

- La régression nette élastique est une combinaison de régression de Lasso et de Ridge
- Il ajoute des pénalités L1 et L2 à la fonction de coût, réduisant efficacement le nombre de variables prédictives et réduisant l'impact de certaines variables prédictives sur le modèle.

$$\text{logit}(p) = \ln \left(\frac{p}{1 - p} \right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

Régression logistique

- La régression logistique est une approche statistique utilisée pour prédire la probabilité qu'un événement se produise.
- Il est utilisé pour prédire le résultat binaire (0 ou 1) en fonction d'une ou plusieurs variables prédictives (X).
- Dans cette formule, p représente la probabilité d'un résultat binaire et z représente la combinaison linéaire de variables prédictives dans un modèle de régression logistique.
- La fonction sigmoïde transforme la combinaison linéaire z en une probabilité p comprise entre 0 et 1.

$$p = \frac{1}{1 + e^{-z}}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}}$$

Code de régression logistique

```
from sklearn.linear_model import LogisticRegression

# Initialize the logistic regression model
lr = LogisticRegression()

# Fit the model to the training data
lr.fit(X_train, y_train)

# Make predictions on the test data
predictions = lr.predict(X_test)
```

Exemple de régression logistique

- Disons que nous avons un ensemble de données sur les transactions des clients et que nous voulons prédire si un client achètera un produit ou non. Le jeu de données contient les variables suivantes :
- Acheté (variable dépendante) : 1 si le client a acheté le produit, 0 si ce n'est pas le cas
- Âge (variable indépendante) : l'âge du client en années
- Sexe (variable indépendante) : le sexe du client (0 pour un homme, 1 pour une femme)
- Revenu (variable indépendante) : revenu annuel du client en milliers de dollars
- Pour effectuer une régression logistique sur ce jeu de données, nous devons d'abord diviser le jeu de données en un ensemble d'apprentissage et un ensemble de tests. Ensuite, nous ajusterions un modèle de régression logistique aux données d'apprentissage en utilisant les variables indépendantes Âge, Sexe et Revenu pour prédire la variable dépendante Acheté.

$$\text{logit}(Purchased) = \beta_0 + \beta_1 \times \text{Age} + \beta_2 \times \text{Gender} + \beta_3 \times \text{Income}$$

où logit(Acheté) est le logarithme naturel des cotes d'un client achetant le produit, et β_0 , β_1 , β_2 et β_3 sont des coefficients qui déterminent l'effet de chaque variable indépendante sur la variable dépendante.

Exemple de régression logistique - cont

- Pour estimer les coefficients, l'algorithme de régression logistique utilise l'estimation du maximum de vraisemblance, qui cherche à trouver l'ensemble des coefficients qui maximisent la probabilité des données observées.
- Une fois le modèle ajusté aux données d'apprentissage, nous pouvons l'utiliser pour prédire la probabilité qu'un client achète le produit dans les données de test. Par exemple, si un client est une femme de 35 ans avec un revenu de 60 000 \$, le modèle de régression logistique peut prédire une probabilité de 0,75 qu'elle achète le produit.
- Nous pouvons ensuite utiliser une valeur seuil pour convertir les probabilités prédites en prédictions binaires (1 pour l'achat, 0 pour aucun achat). Par exemple, si nous fixons le seuil à 0,5, le modèle de régression logistique prédirait un achat pour la cliente de 35 ans dans l'exemple ci-dessus, puisque sa probabilité d'achat prédite (0,75) est supérieure au seuil.

Applications de régression

1. Prévisions des ventes: La régression peut être utilisée pour prédire les ventes futures en fonction des données de ventes passées et d'autres variables telles que les dépenses marketing, les indicateurs économiques et les tendances saisonnières. En analysant les données de ventes passées et en identifiant les modèles et les relations avec d'autres variables, les modèles de régression peuvent aider les entreprises à prévoir les ventes futures et à planifier en conséquence.
2. Analyse des campagnes marketing: La régression peut être utilisée pour mesurer l'efficacité des campagnes marketing en analysant la relation entre les dépenses marketing et les ventes ou l'acquisition de clients qui en résultent. Les modèles de régression peuvent aider les entreprises à identifier les canaux marketing les plus efficaces, à optimiser les dépenses marketing et à prendre des décisions basées sur les données pour améliorer le retour sur investissement des campagnes.
3. Prédiction de désabonnement client : la régression peut être utilisée pour prédire quels clients sont les plus susceptibles de se désabonner (annuler leur abonnement ou quitter le service) en fonction de divers attributs et comportements des clients. En analysant les données des clients passés et en identifiant les modèles et les relations avec le taux de désabonnement, les modèles de régression peuvent aider les entreprises à identifier les clients à risque et à prendre des mesures proactives pour prévenir le taux de désabonnement.
4. Évaluation des risques : La régression peut être utilisée dans l'évaluation des risques pour prédire la probabilité de certains événements ou résultats en fonction de divers facteurs de risque. Par exemple, dans la souscription d'assurance, les modèles de régression peuvent être utilisés pour prédire la probabilité qu'un titulaire de police fasse une réclamation en fonction de son âge, de son emplacement, de ses antécédents de conduite et d'autres facteurs.
5. Prévision des prix: La régression peut être utilisée pour prédire le prix optimal d'un produit ou d'un service en fonction de divers facteurs tels que les coûts de production, la demande, la concurrence et le comportement des consommateurs. En analysant les données de tarification passées et en identifiant les modèles et les relations avec d'autres variables, les modèles de régression peuvent aider les entreprises à fixer des prix qui maximisent les revenus et les bénéfices.
6. Prévision de la demande : La régression peut être utilisée pour prévoir la demande future d'un produit ou d'un service en fonction des données de ventes passées et d'autres variables telles que les indicateurs économiques, la saisonnalité et les promotions. En analysant les données de la demande passée et en identifiant les modèles et les relations avec d'autres variables, les modèles de régression peuvent aider les entreprises à optimiser les niveaux de stock, les calendriers de production et les stratégies de tarification.
7. Prédiction de la rétention des employés: La régression peut être utilisée pour prédire quels employés sont les plus susceptibles de quitter l'entreprise en fonction de divers facteurs tels que la satisfaction au travail, le rendement, la durée d'emploi et le salaire. En analysant les données des employés passés et en identifiant les tendances et les relations avec le roulement du personnel, les modèles de régression peuvent aider les entreprises à identifier les employés à risque et à prendre des mesures proactives pour améliorer la rétention.

Descente de gradient (Gradient Descent)

- La descente de gradient est un algorithme d'optimisation itératif utilisé pour minimiser la fonction de coût du modèle.
- Dans la régression linéaire, l'algorithme de descente de gradient est utilisé pour minimiser la fonction de coût de l'erreur quadratique moyenne (MSE).

Descente de gradient (Gradient Descent)

- Gradient descent is an iterative optimization algorithm commonly used to minimize the cost function in machine learning models. The goal of gradient descent is to find the optimal set of parameters (e.g., coefficients in linear regression) that minimize the cost function.
- Here's how gradient descent is calculated:
- Initialize the parameters: The first step in gradient descent is to initialize the parameters to some arbitrary values.
- Calculate the gradient: The next step is to calculate the gradient of the cost function with respect to the parameters. The gradient is a vector that points in the direction of steepest ascent, which is the direction of increasing cost.
- Update the parameters: Once the gradient has been calculated, the next step is to update the parameters by subtracting a fraction of the gradient from the current parameter values. This fraction is called the learning rate, and it determines the size of the steps taken in the parameter space.
- Repeat: Steps 2 and 3 are repeated until the cost function reaches a minimum (or a stopping criterion is met).

Descente de gradient: adjustments de paramètres

- Plusieurs paramètres peuvent être ajustés en descente de gradient pour contrôler le processus d'optimisation:
 - **Taux d'apprentissage:** le taux d'apprentissage détermine la taille des pas effectués dans l'espace des paramètres. Un taux d'apprentissage plus faible entraîne une convergence plus lente mais plus précise, tandis qu'un taux d'apprentissage plus élevé entraîne une convergence plus rapide mais potentiellement instable. **[Learning Rate]**
 - **Nombre d'itérations:** le nombre d'itérations détermine le nombre de fois que l'algorithme mettra à jour les paramètres avant de s'arrêter. L'augmentation du nombre d'itérations peut améliorer la précision de la solution, mais aussi augmenter le coût de calcul. **[epochs]**
 - **Taille du lot:** dans la descente de gradient par lots, le gradient est calculé sur l'ensemble de l'ensemble d'apprentissage à chaque itération.
 - Dans la **descente de gradient stochastique**, le gradient est calculé sur un seul exemple d'apprentissage à chaque itération. **[Stochastic Gradient Descent (SGD)]**
 - La **descente de gradient par mini-lots** est un compromis entre les deux, où le gradient est calculé sur un petit sous-ensemble de l'ensemble d'apprentissage à chaque itération. La taille du lot détermine la taille du sous-ensemble utilisé dans la descente de gradient de mini-lots. **[mini-batch]**
 - **Régularisation:** La régularisation est une technique utilisée pour prévenir le surajustement en ajoutant un terme de pénalité à la fonction coût. La régularisation L1 (Lasso) et la régularisation L2 (Ridge) sont couramment utilisées en régression linéaire.
- L'ajustement de ces paramètres peut affecter la vitesse de convergence, la précision et la stabilité de l'algorithme. Trouver les valeurs optimales pour ces paramètres est souvent un processus itératif qui implique des essais et des erreurs ou une recherche par grille.

Mises à jour des paramètres du modèle (régression linéaire)

- Cette formule met à jour chaque paramètre du modèle θ_j en soustrayant le taux d'apprentissage multiplié par la dérivée partielle de la fonction de coût par rapport à θ_j , mise à l'échelle par la moyenne des différences entre les valeurs cibles prédites et réelles pour tous les exemples d'apprentissage.
- L'algorithme continue de mettre à jour les paramètres du modèle à l'aide de cette formule jusqu'à ce que la convergence ou un nombre maximal spécifié d'itérations soit atteint.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}, \theta) - y^{(i)})^2$$

Fonction de coût, MSE

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}, \theta) - y^{(i)}) x_j^{(i)}$$

Mise à jour des paramètres à l'aide du gradient de fonction de coût et du taux d'apprentissage (alpha)

Mises à jour des paramètres du modèle (régression logistique)

- Cette formule met à jour chaque paramètre du modèle θ_j en soustrayant le taux d'apprentissage multiplié par la dérivée partielle de la fonction de coût par rapport à θ_j , mise à l'échelle par la moyenne des différences entre les valeurs cibles prédites et réelles pour tous les exemples d'apprentissage.
- L'algorithme continue de mettre à jour les paramètres du modèle à l'aide de cette formule jusqu'à ce que la convergence ou un nombre maximal spécifié d'itérations soit atteint.

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h(x^{(i)}, \theta)) + (1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))]$$

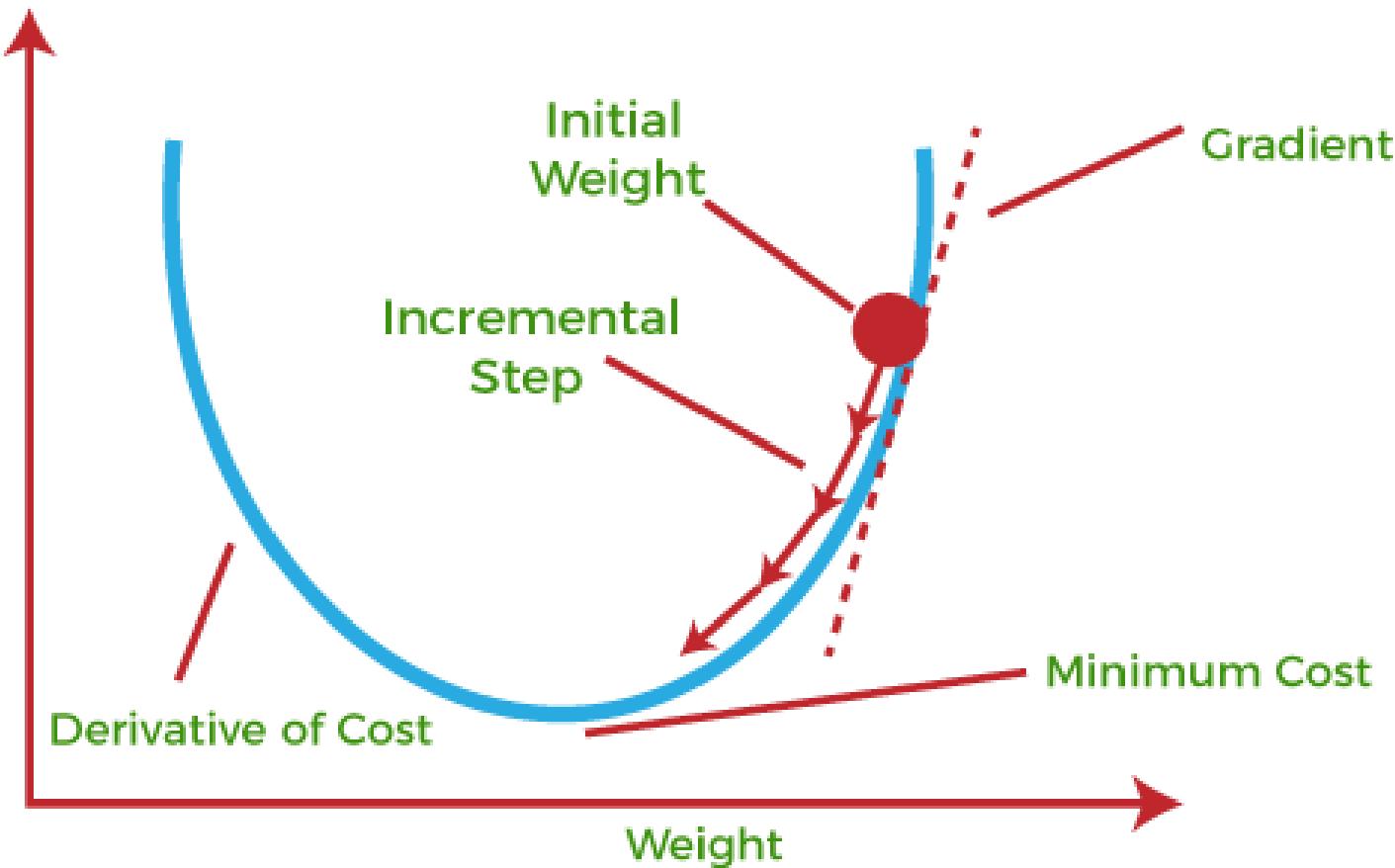
Fonction de coût, cross-entropy

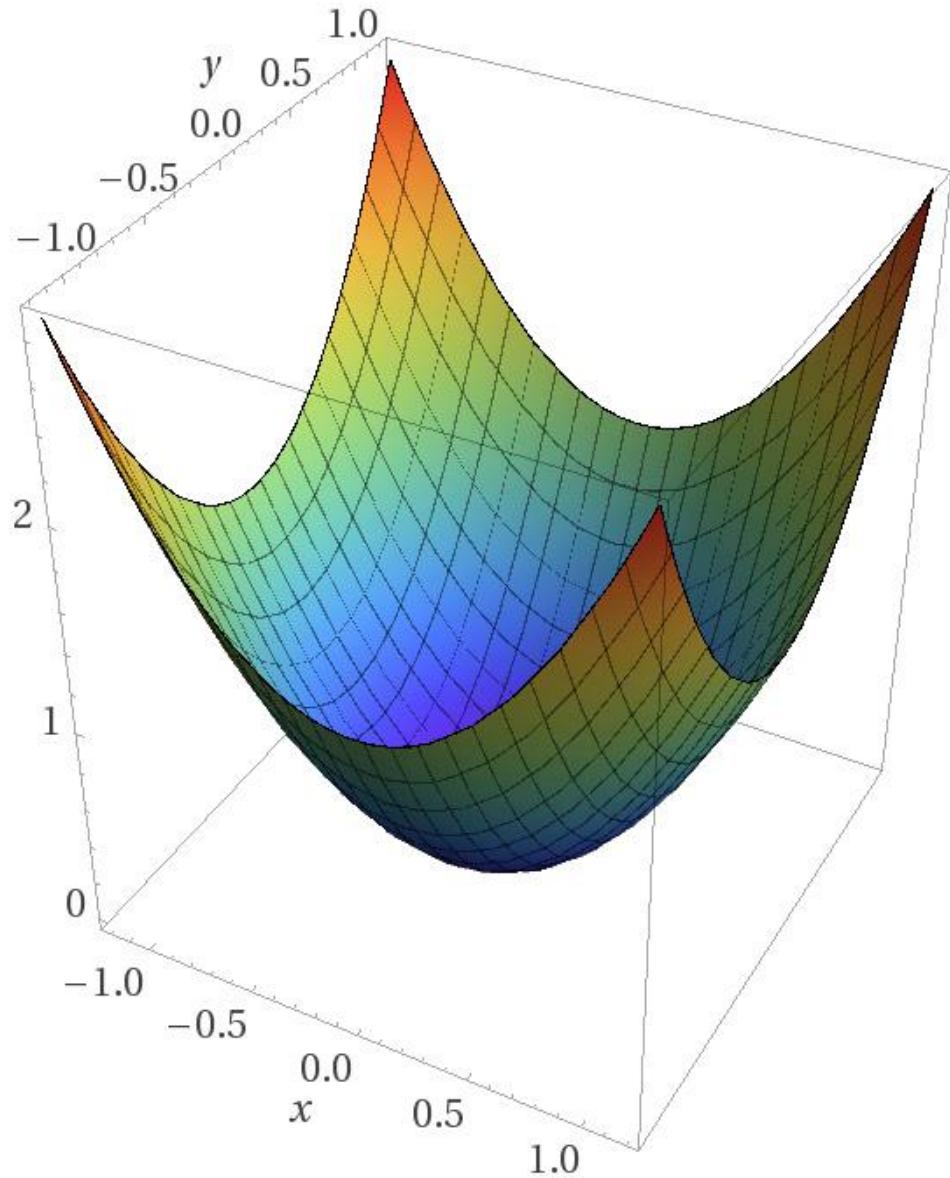
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}, \theta) - y^{(i)}) x_j^{(i)}$$

Mise à jour des paramètres à l'aide du gradient de fonction de coût et du taux d'apprentissage (alpha)

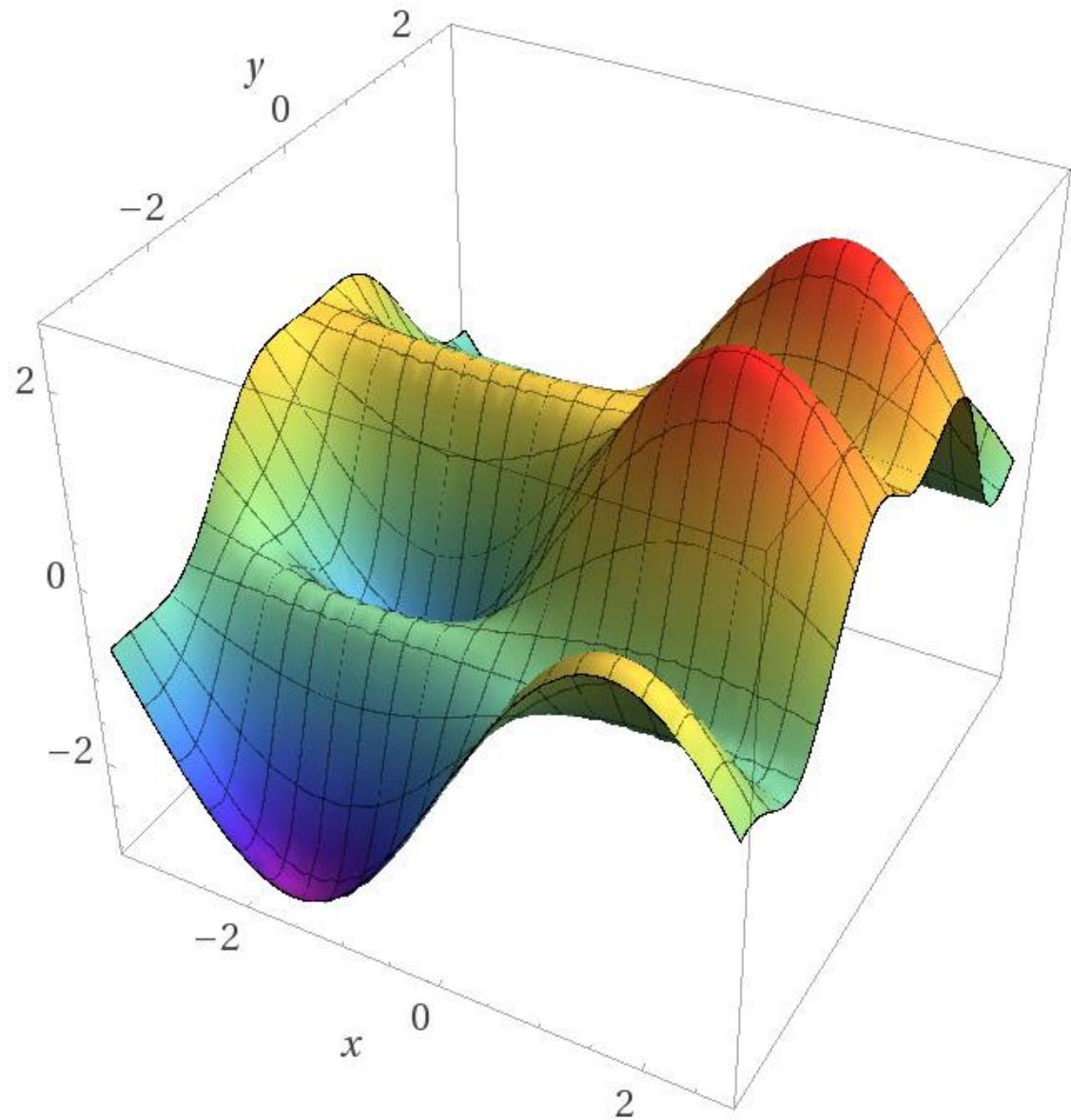
Direction of the parameter update

- En descente de gradient, la direction de la mise à jour des paramètres est déterminée par le signe du gradient. Le gradient d'une fonction en un point spécifique est un vecteur qui pointe dans la direction de l'ascension la plus raide de la fonction en ce point.
- Pour minimiser une fonction de coût, nous voulons aller dans la direction de la descente la plus raide, qui est la direction opposée de la pente.
- Lors de chaque itération de l'algorithme, le gradient est calculé par rapport aux paramètres actuels du modèle, et son signe est utilisé pour mettre à jour les paramètres dans la direction qui réduit la fonction de coût. Plus précisément, si le gradient est positif, cela signifie que l'augmentation de la valeur du paramètre augmentera la fonction de coût, nous devons donc aller dans la direction opposée en diminuant la valeur du paramètre. Si le gradient est négatif, cela signifie que la diminution de la valeur du paramètre augmentera la fonction de coût, nous devons donc aller dans la direction opposée en augmentant la valeur du paramètre.
- Le taux d'apprentissage détermine la taille des étapes de la mise à jour des paramètres, et sa valeur doit être soigneusement choisie pour équilibrer entre vitesse de convergence et stabilité. Si le taux d'apprentissage est trop faible, l'algorithme peut prendre beaucoup de temps à converger. Si le taux d'apprentissage est trop important, l'algorithme peut dépasser le minimum et osciller autour de celui-ci, voire diverger.
- Dans l'ensemble, l'algorithme de descente de gradient met à jour les paramètres du modèle de manière itérative en suivant la direction de descente la plus raide de la fonction de coût, jusqu'à ce que la convergence ou un critère d'arrêt soit atteint. En ajustant le taux d'apprentissage et d'autres hyperparamètres, nous pouvons contrôler la vitesse et la qualité du processus d'optimisation.





Computed by Wolfram|Alpha



Computed by Wolfram|Alpha

Fonctions de coût

- Les fonctions de coût sont utilisées dans l'apprentissage automatique pour mesurer l'erreur ou l'écart entre la sortie prévue d'un modèle et la sortie réelle (c'est-à-dire la réalité sur le terrain) sur un ensemble de données donné.
- L'objectif de la formation d'un modèle d'apprentissage automatique est de minimiser la fonction de coût, ce qui peut être réalisé en ajustant les paramètres du modèle pendant le processus de formation.
- Deux fonctions de coût couramment utilisées dans l'apprentissage automatique sont l'erreur quadratique moyenne (MSE) et l'entropie croisée.
- Dans les deux cas, l'objectif est de minimiser la fonction de coût en ajustant les paramètres du modèle pendant le processus de formation.
- Cela se fait généralement à l'aide d'un algorithme d'optimisation, tel que la descente de gradient, qui met à jour de manière itérative les paramètres pour minimiser la fonction de coût.

Fonctions de coût : erreur quadratique moyenne (MSE)

- L'erreur quadratique moyenne (MSE) est une fonction de coût qui mesure la différence quadratique moyenne entre les valeurs de production prévues et réelles.
- Il est couramment utilisé dans les problèmes de régression, où le but est de prédire une variable continue.
- La formule pour MSE est la suivante : La descente de gradient est un algorithme d'optimisation itératif utilisé pour minimiser la fonction de coût du modèle.
- Dans la régression linéaire, l'algorithme de descente de gradient est utilisé pour minimiser la fonction de coût de l'erreur quadratique moyenne (MSE).

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

Fonctions de coût: entropie croisée

- L'entropie croisée est une fonction de coût couramment utilisée dans les problèmes de classification, où le but est de prédire une variable discrète (par exemple, la classification binaire ou multiclasse).
- Il mesure la différence entre la distribution de probabilité prédite et la distribution de probabilité réelle. La formule de l'entropie croisée binaire est:

$$CE = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

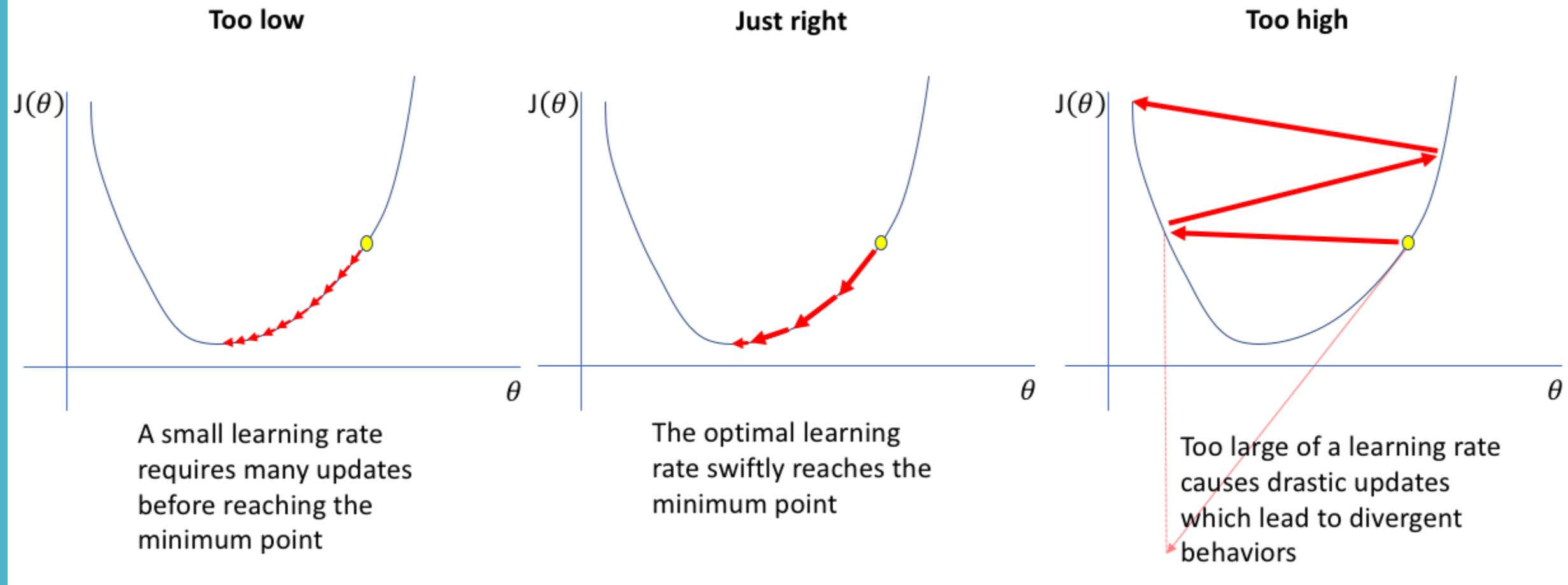
entropie croisée (cross entropy), con't

- L'entropie croisée est une mesure de la différence entre deux distributions de probabilité.
- Dans le contexte de l'apprentissage automatique, il est couramment utilisé comme fonction de coût dans les problèmes de classification, en particulier pour la régression logistique et les réseaux neuronaux.
- La fonction de perte d'entropie croisée mesure la dissimilarité entre la distribution de probabilité prédite et la distribution de probabilité réelle de la variable cible.
- L'idée de base de l'entropie croisée est de mesurer la quantité d'informations nécessaires pour coder les données d'une distribution de probabilité en utilisant une autre distribution de probabilité.
- Si deux distributions de probabilité sont identiques, l'entropie croisée entre elles est nulle. **Plus la différence entre les deux distributions est grande, plus l'entropie croisée est élevée.**
- La fonction de coût d'entropie croisée pénalise davantage le modèle pour avoir fait des prédictions incorrectes et confiantes, par rapport à des prédictions incertaines ou légèrement incorrectes.
- En effet, la fonction logarithmique amplifie la différence entre la probabilité prédite et l'étiquette binaire vraie. L'objectif de l'algorithme d'apprentissage est de minimiser la perte d'entropie croisée en ajustant les paramètres du modèle à l'aide de la descente de gradient ou d'autres méthodes d'optimisation.

Taux d'apprentissage

- Le taux d'apprentissage est un hyperparamètre utilisé dans les algorithmes d'optimisation, tels que la descente de gradient, pour déterminer la taille du pas ou la vitesse à laquelle les paramètres d'un modèle d'apprentissage automatique sont mis à jour pendant l'apprentissage.
- Il contrôle la quantité de modification des paramètres à chaque mise à jour et a donc un impact significatif sur la convergence et les performances du modèle.
- Un taux d'apprentissage trop faible peut entraîner une convergence lente et bloquer le modèle dans les optima locaux.
- D'autre part, un taux d'apprentissage trop élevé peut amener le modèle à dépasser les paramètres optimaux et à ne pas converger. Par conséquent, le choix d'un taux d'apprentissage approprié est crucial pour obtenir de bonnes performances et une convergence dans les modèles d'apprentissage automatique.
- Le choix de la stratégie de taux d'apprentissage dépendra du problème spécifique et du modèle à former.
- En général, il est important de surveiller la performance du modèle pendant la formation et d'ajuster le taux d'apprentissage en conséquence pour assurer la convergence et une bonne performance.

Choix du taux d'apprentissage



Stratégies de taux d'apprentissage

- Il existe plusieurs stratégies pour sélectionner le taux d'apprentissage, notamment :
- Recherche par grille : essayer une gamme de taux d'apprentissage différents et sélectionner celui qui donne les meilleures performances sur un ensemble de validation.
- Méthodes de taux d'apprentissage adaptatif: algorithmes qui ajustent le taux d'apprentissage pendant l'entraînement en fonction des performances du modèle. Les exemples incluent AdaGrad, RMSProp et Adam.
- Calendriers de taux d'apprentissage : algorithmes qui réduisent le taux d'apprentissage au fil du temps à mesure que le modèle converge. Les exemples incluent la décroissance par étapes, la décroissance exponentielle et les calendriers de taux d'apprentissage cycliques.