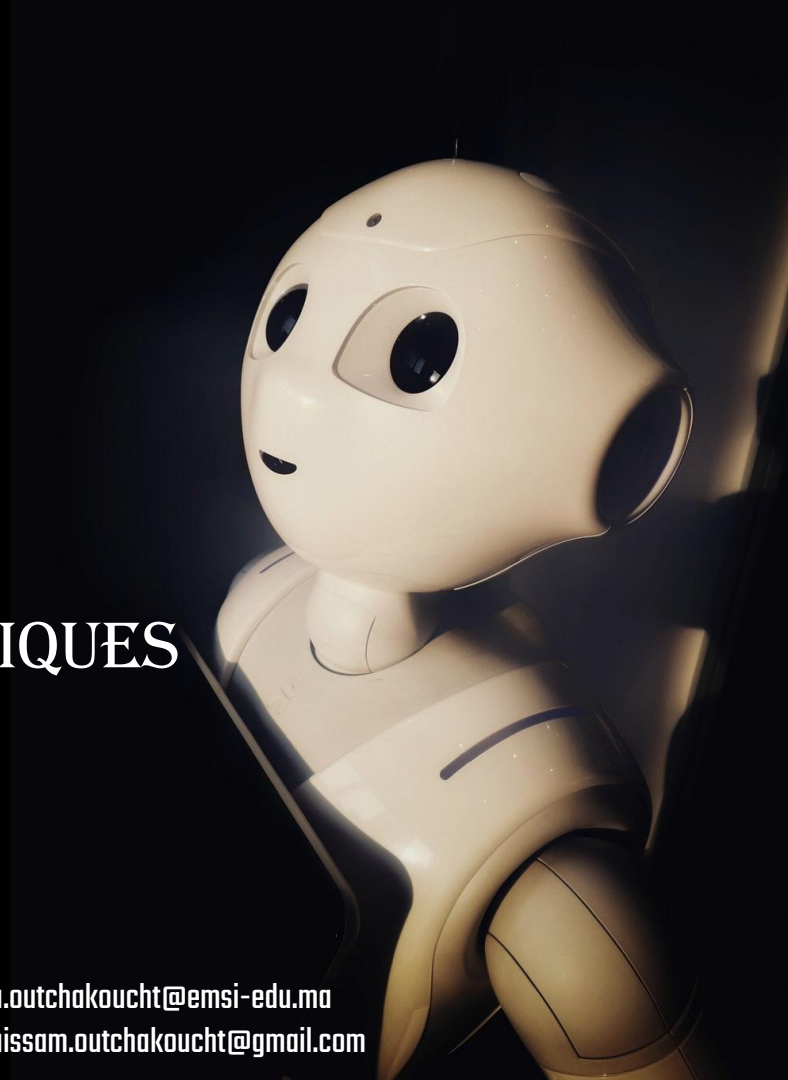
ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de
HONORIS UNITED UNIVERSITIES

ML / DL

MACHINE LEARNING: LES ALGORITHMES CLASSIQUES



Aissam Outchakoucht

a.outchakoucht@emsi-edu.ma
aissam.outchakoucht@gmail.com

RECAP

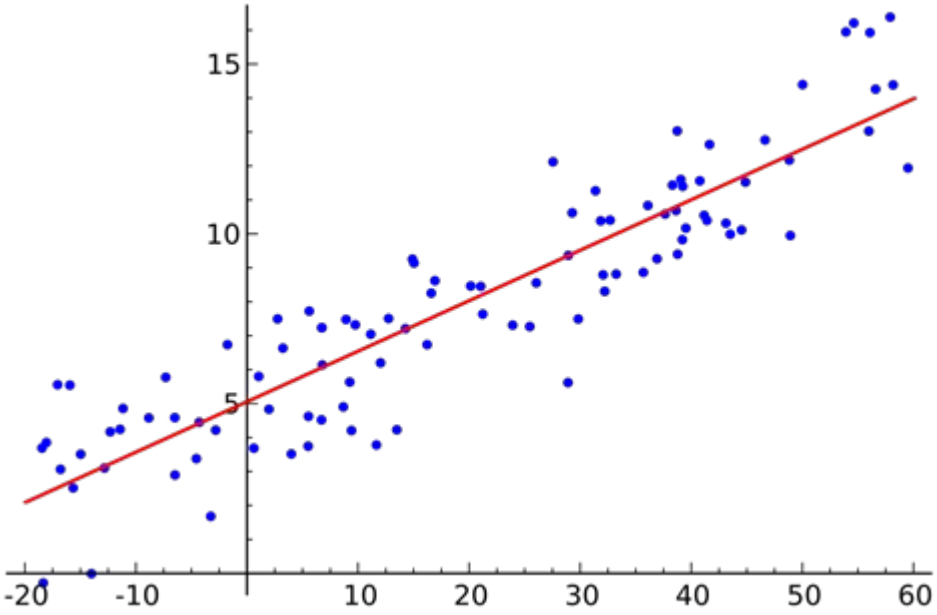
```
# Charger l'ensemble de données
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()

# Convertir en DataFrame pandas
data = pd.DataFrame(housing.data, columns=housing.feature_names)
data['MedHouseVal'] = housing.target

# Afficher les cinq premières lignes
data.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

```
selected_features = [
    'MedInc', 'Latitude', 'Longitude', 'RoomsPerHousehold',
    'BedroomsPerRoom', 'PopulationPerHousehold', 'HouseAge'
]
X = data[selected_features]
y = data['MedHouseVal']

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the features
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42)
```

```
selected_features = [
    'MedInc', 'Latitude', 'Longitude', 'RoomsPerHousehold',
    'BedroomsPerRoom', 'PopulationPerHousehold', 'HouseAge'
]
X = data[selected_features]
y = data['MedHouseVal']

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the features
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42)

# Train with scaling
lr_with_scaling = LinearRegression()
lr_with_scaling.fit(X_train, y_train)

# Predict on test set
y_pred_with_scaling = lr_with_scaling.predict(X_test)

rmse_with_scaling = np.sqrt(mean_squared_error(y_test, y_pred_with_scaling))
```

RÉGRESSION LINÉAIRE



Comment ça fonctionne :

- **Objectif** : Prédire une variable cible continue.
- Ajuste une droite ($y = wX + b$) sur les points de données pour minimiser l'erreur (par exemple, l'erreur quadratique moyenne).
- Suppose une relation linéaire entre les variables d'entrée et la cible.

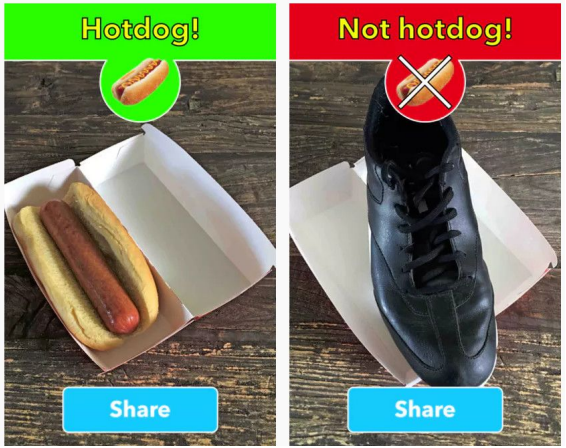
Quand l'utiliser :

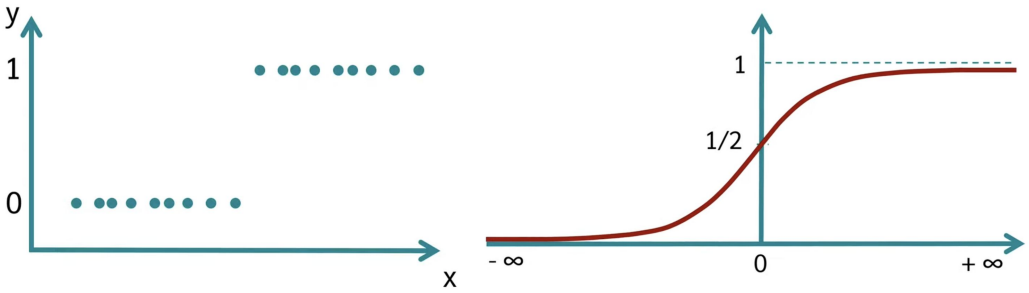
- Lorsqu'il existe une relation linéaire évidente entre les entrées et la sortie.
- Pour des problèmes de régression simples avec peu de variables.

Limites :

- Mauvaise performance si la relation est non linéaire.
- Sensible aux valeurs aberrantes.
- Suppose l'indépendance entre les prédicteurs.

RÉGRESSION LOGISTIQUE





Comment ça fonctionne :

- **Objectif** : Prédire des probabilités pour des tâches de classification (binaire ou multiclasses).
- Utilise une fonction sigmoïde pour modéliser la probabilité d'un résultat et maximise la probabilité des classifications correctes.

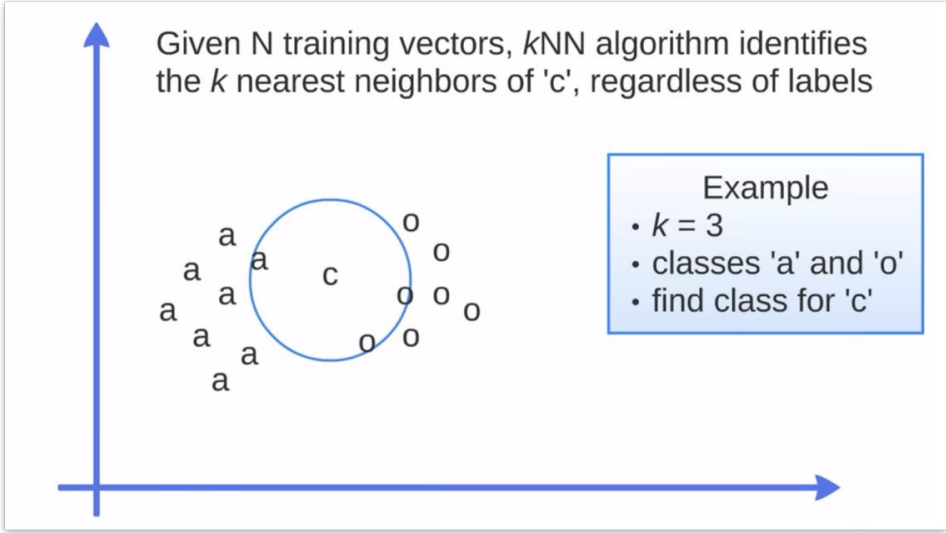
Quand l'utiliser :

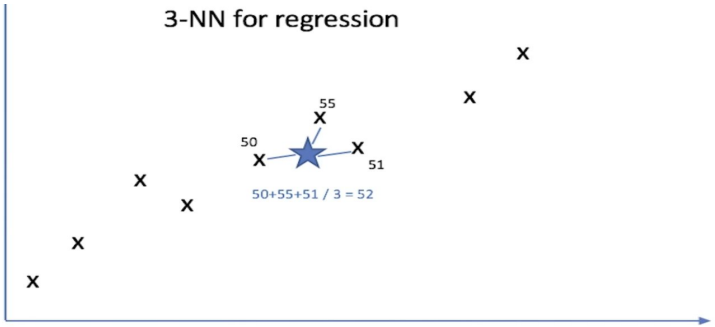
- Pour des problèmes de classification binaire ou multiclasses simples.

Limites :

- Suppose une linéarité entre les entrées et les log-odds.
- Incapable de capturer des frontières de décision complexes.

K-PLUS PROCHES VOISINS (KNN)





Comment ça fonctionne :

- Prédit l'étiquette d'un point en fonction de l'étiquette majoritaire de ses K plus proches voisins.

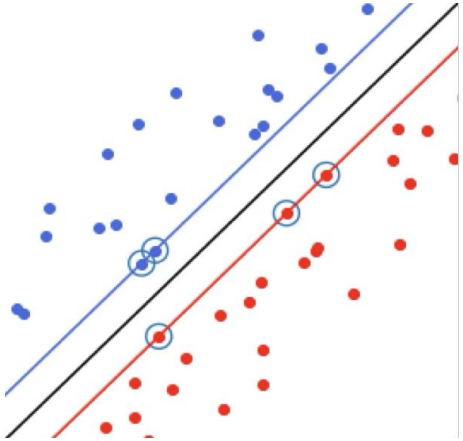
Quand l'utiliser :

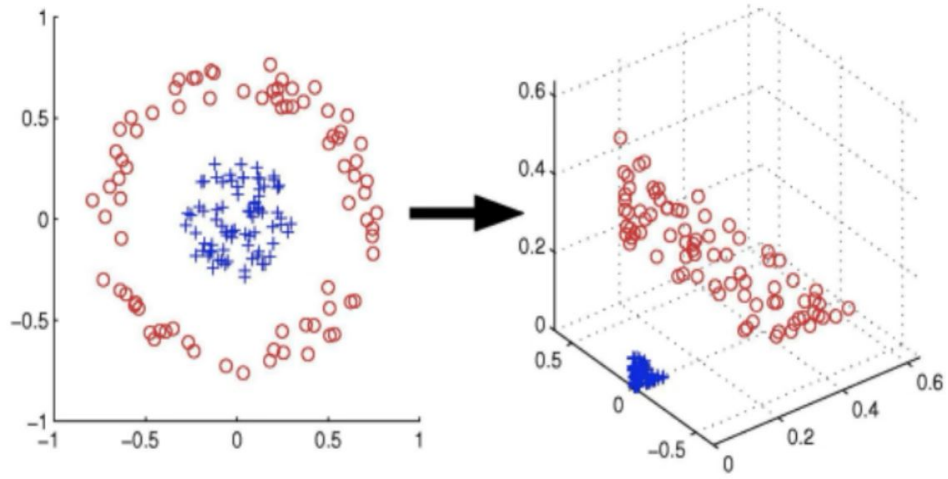
- Pour des ensembles de données de faible dimension et de petite taille.
- Lorsqu'aucune hypothèse sur la distribution des données n'est nécessaire.

Limites :

- Coûteux en temps de prédiction (calcul des distances).
- Sensible au bruit et aux variables non pertinentes.

MACHINES À VECTEURS DE SUPPORT (SVM)





Comment ça fonctionne :

- Trouve l'hyperplan optimal qui sépare les classes dans un espace de haute dimension.
- Utilise des fonctions noyau pour gérer les relations non linéaires.

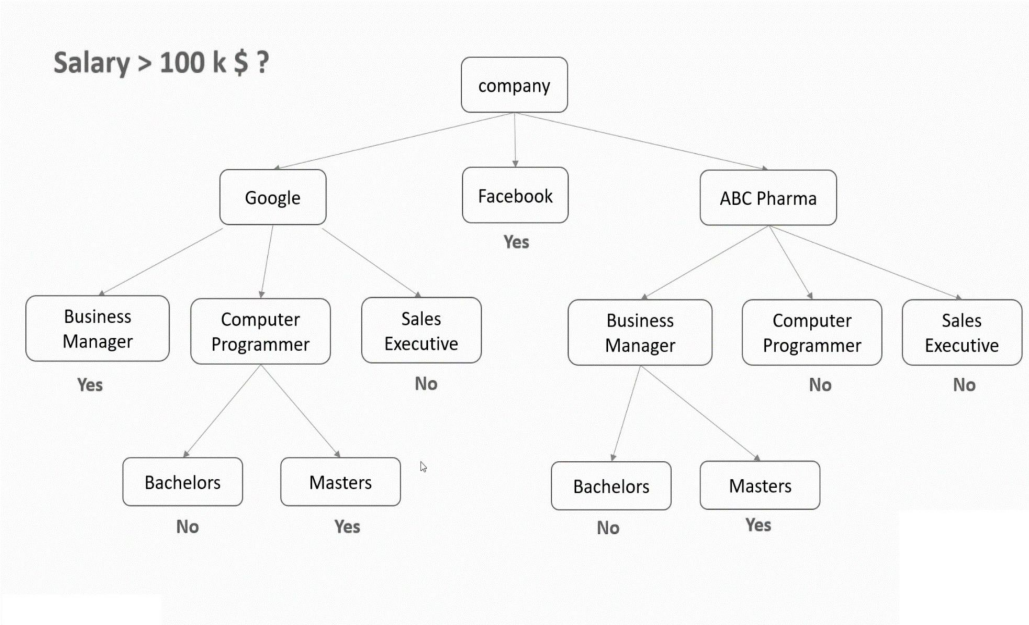
Quand l'utiliser :

- Pour des ensembles de données petits à moyens.
- Lorsque la frontière de décision est complexe.

Limites :

- Couteux en calcul pour de grands ensembles de données.
- Nécessite un ajustement précis des hyperparamètres

ARBRES DE DÉCISION



Comment ça fonctionne :

- Divise les données en branches en fonction des seuils des variables pour minimiser l'impureté
- Chaque nœud feuille représente une classe ou une valeur.

Quand l'utiliser :

- Pour des modèles interprétables avec des relations non linéaires.
- Lorsqu'il s'agit de données catégoriques ou mixtes.

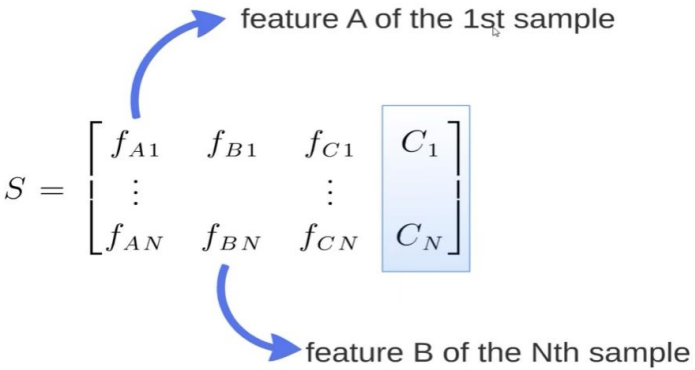
Limites :

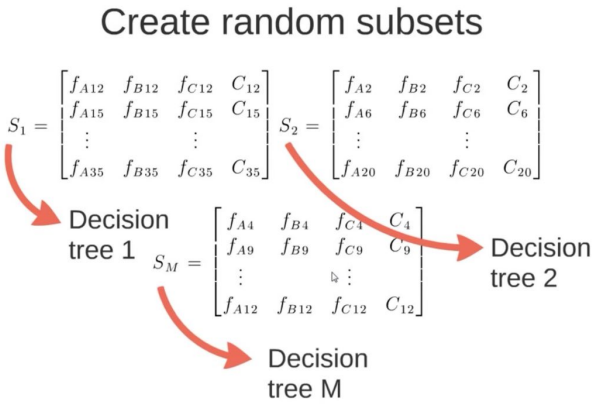
- Sujet au surapprentissage (variance élevée).
- Sensible aux petits changements dans les données.

ARBRES DE DÉCISION

Company	Job Title	Degree	Salary ($)
Google	Business Manager	Masters	120,000
Google	Computer Programmer	Masters	110,000
Google	Computer Programmer	Bachelors	90,000
Google	Sales Executive	Bachelors	80,000
Facebook	Business Manager	Masters	130,000
Facebook	Business Manager	Bachelors	95,000
Facebook	Computer Programmer	Masters	115,000
Facebook	Computer Programmer	Bachelors	85,000
ABC Pharma	Sales Executive	Masters	75,000
ABC Pharma	Business Manager	Masters	105,000

FORÊTS ALÉATOIRES (RANDOM FOREST)





Comment ça fonctionne :

- Ensemble d'arbres de décision, chaque arbre étant formé sur un sous-ensemble aléatoire des données.
- Agrège les prédictions de tous les arbres (vote majoritaire pour la classification ou moyenne pour la régression).

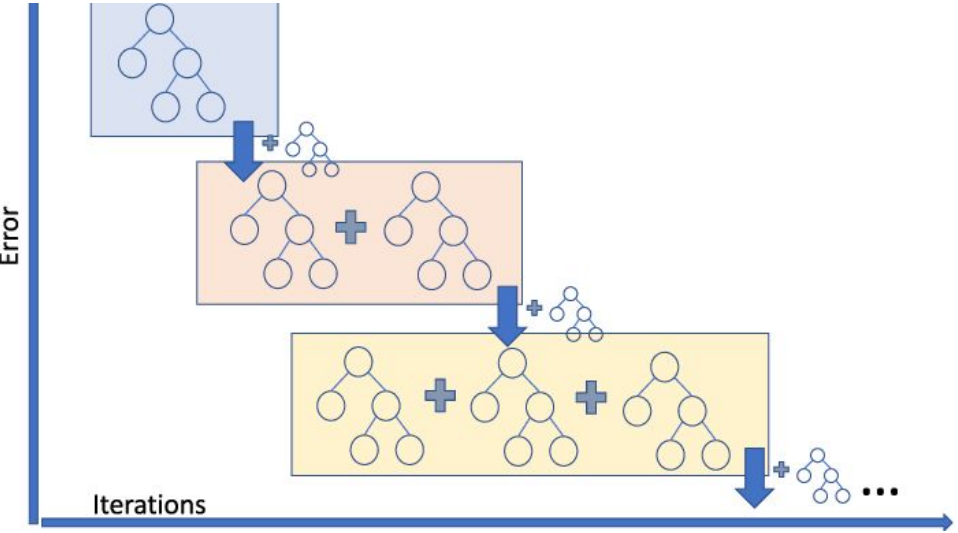
Quand l'utiliser :

- Lorsque l'interprétabilité n'est pas une priorité, mais que la précision l'est.
- Pour des ensembles de données avec une grande dimensionnalité ou des données manquantes.

Limites :

- Peut être coûteux en calcul.
- Pas intrinsèquement interprétable.

BOOSTING



Comment ça fonctionne :

- Construit des modèles séquentiellement, chaque modèle corrigeant les erreurs du précédent.
- Minimise une fonction de perte via la descente de gradient.

Quand l'utiliser :

- Lorsque vous avez besoin d'une grande précision sur des données structurées ou tabulaires.
- Pour des concours de machine learning compétitifs.

Limites :

- Sensible aux hyperparamètres.
- Couteux en calcul.

Classical Machine Learning

Task Driven

Data Driven

Supervised Learning

(Pre Categorized Data)

Classification

(Divide the socks by Color)

Eg. Identity Fraud Detection

Regression

(Divide the Ties by Length)

Eg. Market Forecasting

Unsupervised Learning

(Unlabelled Data)

Clustering

(Divide by Similarity)

Eg. Targeted Marketing

Association

(Identify Sequences)

Eg. Customer Recommendation

Dimensionality Reduction

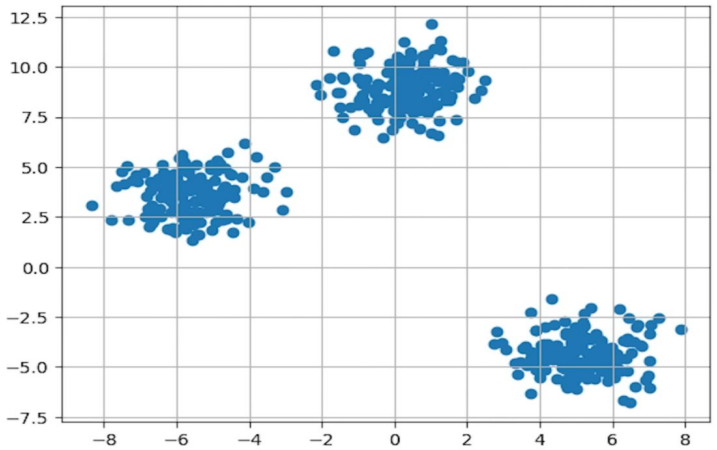
(Wider Dependencies)

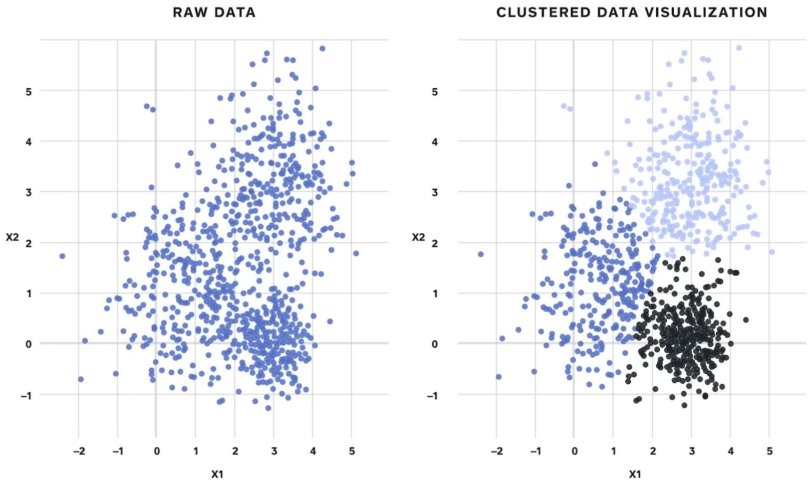
Eg. Big Data Visualization

Obj: Predications & Predictive Models

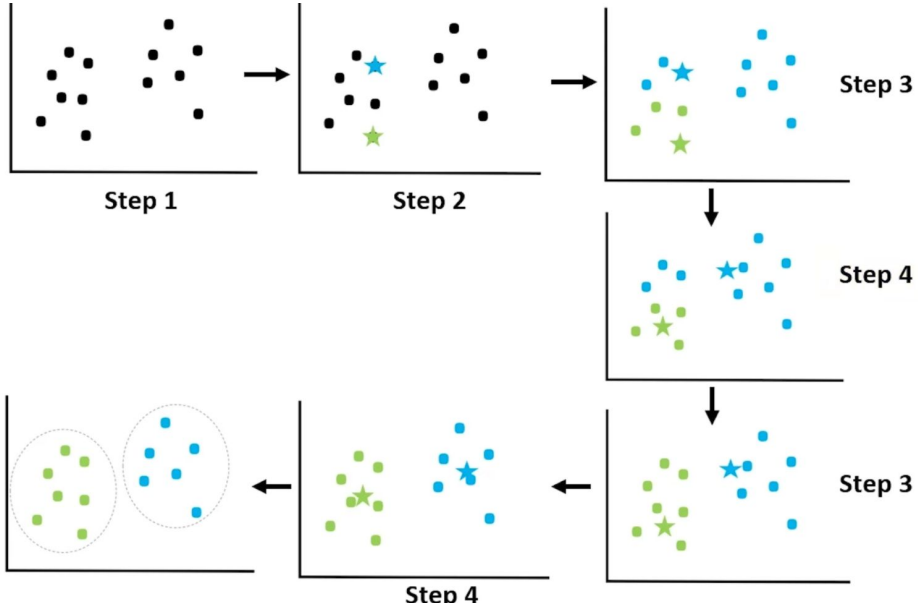
Pattern/ Structure Recognition

UL: K-MEANS CLUSTERING





UL: K-MEANS CLUSTERING



Comment ça fonctionne :

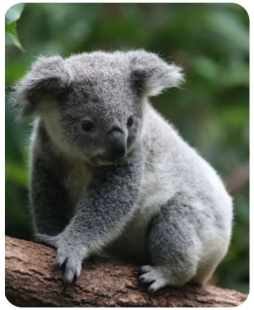
- **Objectif** : Regrouper les données en K clusters en minimisant la distance intra-cluster (les points sont proches de leur centre de cluster).
- **Étapes principales** :
  1. Initialiser K centres de clusters aléatoirement.
  2. Attribuer chaque point de données au cluster dont le centre est le plus proche.
  3. Recalculer les centres des clusters en prenant la moyenne des points attribués.
  4. Répéter jusqu'à convergence (lorsque les centres de clusters ne bougent plus).

Quand l'utiliser :

- Pour explorer des structures ou groupes naturels dans des données non étiquetées (non supervisées).
- Pour les cas où les données peuvent être regroupées en clusters bien séparés.

UL: RÉDUCTION DE DIMENSIONNALITÉ

JPEG | 20KB



JPEG-XR | 13KB

WebP | 10KB

Objectif :

Réduire le nombre de variables ou caractéristiques dans un jeu de données tout en préservant un maximum d'information importante. Cela est utile pour :

- Combattre la malédiction de la dimensionnalité.
- Réduire le bruit dans les données.
- Visualiser des données en haute dimension (souvent dans un espace 2D ou 3D).

Avantages :

- **Réduction de bruit** : La PCA peut éliminer les caractéristiques redondantes ou peu informatives.
- **Amélioration de la performance** : Avec moins de dimensions, les algorithmes ML peuvent fonctionner plus rapidement et éviter le surapprentissage.
- **Visualisation** : Permet de projeter des données complexes en 2D ou 3D pour une meilleure compréhension.

LIMITES DES ALGORITHMES DE ML

- **Ingénierie des caractéristiques** : De nombreux algorithmes dépendent de caractéristiques bien conçues.
- **Évolutivité** : Des algorithmes comme SVM et KNN ont du mal avec de grands ensembles de données.
- **Relations complexes** : Les relations non linéaires sont difficiles pour des modèles comme la Régression Linéaire/Logistique.
- **Haute dimensionnalité** : Le surapprentissage devient un problème lorsque la dimensionnalité augmente.
- **Sparsité des données** : De nombreux algorithmes sous-performent avec des données comme les textes ou les images.

POURQUOI AVONS-NOUS BESOIN DU DL?

1. **Extraction des caractéristiques** : Le DL extrait automatiquement des caractéristiques, éliminant le besoin d'ingénierie manuelle.
2. **Relations non linéaires** : Les réseaux neuronaux capturent des motifs complexes et non linéaires dans les données.
3. **Évolutivité** : Le DL s'adapte bien aux ensembles de données massifs grâce au calcul distribué.
4. **Données non structurées** : Gère efficacement les images, l'audio, le texte et les vidéos.
5. **Apprentissage par transfert** : Les modèles pré-entraînés s'adaptent à de nouvelles tâches avec peu de données supplémentaires.