# Rapport de la SAE 2.02
# - Exploration algorithmique d'un problème -

Dans le cadre de cette SAE, nous avons mis en œuvre et analysé trois algorithmes d'apprentissage automatique : l'algoDelimiteur, K Nearest Neighbors (KNN) et KMeans. L'objectif est de comprendre leur fonctionnement, leurs applications et leurs limitations.

## I - Explication brève des objectifs des algorithmes :

**AlgoDélimiteur**

- **Utilisation :** L'algorithme AlgoDelimiteur est utilisé pour **segmenter un texte en parties distinctes** en fonction de délimiteurs spécifiques. Il sert à **diviser un texte en phrases**, en mots ou en sections selon des **séparateurs définis** comme des espaces, des virgules ou des points.
- **Pourquoi l'utiliser :**
  - **Simplicité :** AlgoDelimiteur est facile à comprendre et à mettre en place. Il se base uniquement sur la reconnaissance de délimiteurs sans nécessiter de traitement complexe.
  - **Pas de paramètre d'entraînement complexe :** Il fonctionne directement sans phase d'apprentissage, ce qui permet une exécution rapide et efficace.
  - **Efficacité avec de grands textes :** AlgoDélimiteur est adapté aux grands volumes de texte, car il permet une segmentation rapide avec un faible coût en calcul.

**KNN** (K Nearest Neighbors)

- **Utilisation** : L'algorithme KNN est utilisé dans des **problèmes de classification** et de **régression**. En classification, il sert à prédire l'étiquette d'une donnée en fonction de celles des K plus proches voisins. En régression, il prédit une valeur numérique en prenant la moyenne (ou médiane) des valeurs des voisins les plus proches.
- **Pourquoi l'utiliser** :
  - **Simplicité** : KNN est facile à comprendre et à implémenter, ne nécessitant aucune phase d'apprentissage explicite. L'algorithme est basé sur les données elles-mêmes, et donc, il est flexible.
  - **Pas de paramètre d'entraînement complexe** : Il ne nécessite pas un modèle complexe ou une longue phase d'entraînement, ce qui le rend adapté aux situations où il est important de prédire rapidement sans un long processus d'apprentissage.
  - **Efficacité avec de petits jeux de données** : KNN est efficace pour des ensembles de données de petite taille où la recherche des voisins les plus proches n'est pas trop coûteuse en termes de temps de calcul.

**KMeans**

- **Utilisation** : KMeans est un algorithme de **clustering (regroupement)** non supervisé, qui permet de segmenter un ensemble de données en K groupes (clusters) en fonction de la similarité des données. Il est souvent utilisé pour **l'exploration des données**, la **réduction de la dimensionnalité** (technique qui consiste à transformer des données en haute dimension en un espace de plus faible dimension tout en conservant un maximum d'informations essentielles, afin d'améliorer l'efficacité des algorithmes et faciliter l'analyse) ou **la détection de structures cachées** dans les données.
- **Pourquoi l'utiliser** :
  - **Clustering rapide** : KMeans est particulièrement utilisé pour sa simplicité et sa rapidité dans des contextes où il est nécessaire de segmenter rapidement un grand nombre de données.

- **Adapté aux données numériques** : Il fonctionne bien avec des données numériques et est efficace quand les clusters sont bien définis par des formes sphériques et de taille plus ou moins homogène.
- **Exploration de données** : Il est très utile dans les tâches où l'on veut découvrir des structures sous-jacentes dans les données sans avoir d'étiquettes ou de connaissances préalables sur la distribution des données.

## II - Décrire les algorithmes, le type de données à traiter, la structure de ces données :

**AlgoDélimiteur** :

### Description de notre algorithme :

**- Lecture de la chaîne :** L'algorithme analyse une chaîne de caractères contenant des délimiteurs.
- **Utilisation d'une pile :** Une pile est utilisée pour gérer l'appariement des délimiteurs gauche et droit.
**- Identification des délimiteurs :** Il détecte les caractères délimiteurs gauche (ouvrants) et les stocke avec un identifiant unique.
**- Vérification des correspondances :** Lorsqu'un délimiteur droit (fermant) est rencontré, il est comparé au dernier délimiteur gauche empilé.
**- Retour du résultat :** Si un délimiteur droit n'a pas de correspondant ou si les délimiteurs ne correspondent pas, l'algorithme retourne une erreur. Si tous les délimiteurs sont correctement appariés, l'algorithme retourne une liste d'identifiants correspondant aux appariements. Sinon, il retourne une chaîne vide.

### Type de données :

L'algorithme traite des chaînes de caractères contenant divers types de délimiteurs (parenthèses, crochets, accolades, etc.). Il attribue un identifiant numérique à chaque ouverture et fermeture valide.

### Structure des données :

Les données sont stockées sous deux structures principales : **une liste** contenant les identifiants des délimiteurs correctement appariés et **une pile** utilisée pour stocker temporairement les délimiteurs ouverts en attente de correspondance.

**KNN :**

### Description de notre algorithme :

- **Chargement des données :** Il lit les fichiers CSV contenant les ensembles d'entraînement et de test.
**- Conversion des données** : Les valeurs sont converties en nombres flottants pour faciliter les calculs.
**- Choix de la distance** : L'utilisateur sélectionne une métrique de distance entre Euclidienne, Manhattan ou Chebyshev.
**- Calcul des distances** : Pour chaque point de test, il calcule la distance avec tous les points d'entraînement.
**- Sélection des voisins** : Il trie les distances et sélectionne les k plus proches voisins.
**- Détermination de la classe** : La classe majoritaire parmi les voisins est attribuée au point de test.
**- Affichage des résultats** : La classe prédite, le nombre de votes et la métrique de distance sont affichés.

### Type de données :

L'algorithme KNN traite des données numériques représentant des mesures de fleurs, comme la longueur et la largeur des pétales et des sépales. Chaque échantillon est associé à une classe indiquant son espèce. L'algorithme utilise ces informations pour classer de nouveaux échantillons en fonction des plus proches voisins.

### Structure des données :

Les données sont organisées sous forme de tableau, où chaque ligne représente un échantillon et chaque colonne contient une information spécifique : 4 colonnes numériques → Mesures des fleurs ; 1 colonne catégorielle → Classe de la fleur

**KMeans :**

### Description de notre algorithme :

**- Chargement des données :** Il lit le fichier CSV et extrait les colonnes utiles (revenu annuel et score de dépense des clients).
**- Initialisation des centroïdes :** Il choisit k points initiaux de manière intelligente (K-Means++).
**- Assignation des clusters :** Chaque point est affecté au cluster dont le centroïde est le plus proche (en calculant la distance euclidienne).
**- Mise à jour des centroïdes :** Les centroïdes sont recalculés en prenant la moyenne des points assignés à chaque cluster.
**- Vérification de convergence :** Si les centroïdes ne changent plus ou si le nombre maximal d'itérations est atteint, l'algorithme s'arrête.
**- Détermination du meilleur k (méthode du coude)** : Il teste plusieurs valeurs de k et calcule l'inertie pour choisir le nombre optimal de clusters.
**- Affichage des résultats** : Il regroupe les clients en clusters et les affiche sous forme de graphique avec leurs centroïdes.

### Type de données :

L'algorithme **K-Means** traite des **données numériques continues**. Le fichier **Mall_Customers.csv** contient des informations sur les clients d'un centre commercial. L'algorithme extrait **deux colonnes numériques** : **Revenu annuel, Score de dépense**
Ces **deux variables** sont utilisées pour regrouper les clients en fonction de leur comportement d'achat.

### Structure des données :

Les données sont sous forme d'un tableau au format csv, où chaque ligne représente un exemple et chaque colonne une caractéristique.

## III. Les limitations ou problèmes rencontrés et les recherches effectuées pour résoudre ces problèmes

1. ### Comprendre les consignes

   Nous nous sommes vite retrouvés perdus à cause d'instructions parfois trop vagues vis-à-vis des algorithmes. Nous avons trouvé que l'organisation sur Moodle manquait de clarté, et il n'y avait pas de consigne globale expliquant précisément tout ce que nous devions faire et rendre pour la SAE, bien qu'un mail récapitulatif ait été envoyé.
   Pour nous en sortir, nous avons cherché plusieurs solutions. Nous avons regardé des vidéos explicatives et consulté des articles et tutoriels en ligne pour mieux comprendre les attentes.

Mais ce qui nous a le plus aidés, c'est le travail en groupe : en discutant ensemble, nous avons pu éclaircir certains points et avancer plus efficacement.

### 2. Programmer les algorithmes

Même en comprenant le fonctionnement théorique des algorithmes, les coder en Python a été un vrai défi. Certains concepts étaient difficiles à mettre en application, et nous avons souvent bloqué sur des erreurs ou des problèmes de syntaxe.
Pour progresser, nous avons cherché des explications sur des forums et utilisé des ressources en ligne pour mieux comprendre certains aspects du langage Python. Encore une fois, le travail en groupe a été un atout.

### 3. Tester les algorithmes

Une fois nos algorithmes codés, nous avons rencontré une autre difficulté : comment vérifier s'ils fonctionnaient correctement ? Pour résoudre ce problème, nous avons d'abord cherché des bases de tests en ligne. Enfin, nous avons mis en place un environnement de test fiable pour effectuer nos vérifications de manière plus efficace.