



*Université de sciences et de la technologie Houari Boumediene*

*Faculté d’Informatique  
Département d’Intelligence Artificielle  
Master 2 Systèmes Informatiques intelligents*

---

# **De l’Analyse des Données à l’Extraction de Connaissances**

---

**Partie 2 : Apprentissage supervisé et non supervisé**

purplestate

## **Data Mining**



*Réalisé Par :*

FERKOUS Sarah  
KHEMSSI Maroua

*Professeur : M. BELKADI WIDAD*

*Année universitaire : 2023 / 2024*

# Table des matières

<b>Introduction</b>	<b>6</b>
<b>1 Analyse et Pretraitemet du dataset</b>	<b>7</b>
1.1 Objectifs . . . . .	7
1.2 Visualisation du contenu du dataset . . . . .	7
1.3 Traitement des valeurs manquantes : . . . . .	7
1.4 Traitement des valeurs aberrantes . . . . .	9
1.5 Réduction des données (élimination des redondances) . . . . .	9
1.6 Normalisation des donnees : . . . . .	10
1.7 Récapitulatif . . . . .	11
1.8 Conclusion . . . . .	11
<b>2 Analyse supervisée : Classification</b>	<b>12</b>
2.1 Algorithme K-NN . . . . .	12
2.1.1 Méthodes de Calcul de Distance pour K-NN . . . . .	13
2.2 Arbre de decision . . . . .	13
2.2.1 Les Fonction de cout . . . . .	14
2.2.2 Gini vs Entropy . . . . .	15
2.3 Foret Aléatoire . . . . .	15
2.4 Etude Experimentale . . . . .	17
2.4.1 Objectifs : . . . . .	17
2.4.2 Les mesures de comparaison . . . . .	17
2.4.3 Le choix de distance :Algorithme K-NN . . . . .	17
2.4.4 Matrice de confusion . . . . .	18
2.4.5 Comparaison entre les trois modeles de classification . . . . .	20
2.5 Conclusion . . . . .	20

<b>3 Analyse non supervisée : clustering</b>	<b>21</b>
3.1 Algorithme de clustering basé partitionnement . . . . .	21
3.1.1 L'Algorithme K-means . . . . .	21
3.1.2 Choisir K : le nombre de clusters : . . . . .	23
3.1.3 l'initialisation des centroïdes . . . . .	24
3.1.4 le nombre d'itérations et la convergence . . . . .	24
3.1.5 Visualisation des clusters . . . . .	26
3.1.6 Etude de la Stabilité de K-means . . . . .	28
3.2 Algorithme de clustering basé densité . . . . .	30
3.2.1 Algorithme DBSCAN . . . . .	30
3.2.2 Le choix de esp $\varepsilon$ . . . . .	32
3.2.3 Le choix de nombre minimal de voisins nécessair . . . . .	32
3.3 Comparaison des deux algorithmes K-means et DBSCAN . . . . .	34
3.4 Conclusion . . . . .	35
.1 Annexe A : Détails du Jeu de Données . . . . .	38
.2 Annexe B : Visualisation du Jeu de Données . . . . .	38
.2.1 Les symétries . . . . .	38
.2.2 Visualisation des outliers . . . . .	40
.2.3 Construction des histogrammes de . . . . .	41

# Liste des tableaux

1.1	Description de dataset-1- . . . . .	7
1.2	Description des colonnes du dataset . . . . .	8
1.3	Portion Résultat Normalisation Min-Max . . . . .	10
1.4	Portion Résultat normalisées Z-score . . . . .	11
1.5	Récapitulatif des traitements effectués. . . . .	11
2.1	Méthodes de Calcul de Distance pour K-NN . . . . .	13
2.2	Description des Mesures d'Évaluation de Classification . . . . .	17
2.3	Résultats d'Évaluation des Modèles . . . . .	20
3.1	Extrait des résultats de l'expérimentation . . . . .	25
3.2	Résultats de l'expérimentation avec différents paramètres de DBSCAN . . . . .	32
3.3	Comparaison entre K-means et DBSCAN . . . . .	34
4	Description statistique des attributs. . . . .	38

# Table des figures

1.1	Les valeurs uniques de l'attribut P . . . . .	8
1.2	Boîte à moustaches de N et K . . . . .	9
1.3	La matrice de corrélation . . . . .	9
2.1	Algorithme K-NN . . . . .	13
2.2	Arbre de decision . . . . .	14
2.3	Gini vs Entropy . . . . .	15
2.4	Arbre de decision . . . . .	16
2.5	Choix de distance :Algorithme K-NN . . . . .	18
2.6	Matrice de confusion modele KNN . . . . .	19
2.7	Matrice de confusion modele decision tree . . . . .	19
2.8	Matrice de confusion modele Random Forest . . . . .	19
3.1	Exemple illustratif de K-means . . . . .	22
3.2	Dataset visualisation . . . . .	22
3.3	La courbe du coude . . . . .	23
3.4	Aleatoirement . . . . .	24
3.5	Avec k-means++ . . . . .	24
3.6	Variation de convergence . . . . .	25
3.7	Variation d'Iteration . . . . .	25
3.8	Un seul Cluster (k=1) . . . . .	26
3.9	Deux Clustres (k=2) . . . . .	26
3.10	Trois Clusters (k=3) . . . . .	26
3.11	Cinq Clusters (k=5) . . . . .	26
3.12	Taille Clusters Pour k=3 . . . . .	26
3.13	Stabilité du cluster . . . . .	28
3.14	Stabilité de l'instance 12 . . . . .	28
3.15	Stabilité de l'instance 3 . . . . .	28
3.16	Stabilité de centroïde 1 . . . . .	29

3.17	Stabilité de centroïde 2	29
3.18	Application	30
3.19	Illustration	30
3.20	Le choix de $(\varepsilon)$	32
3.21	silhouette score avec DBSCAN	33
3.22	Clusters avec DBSCAN	33
3.23	Methode de DBSCAN	33
3.24	Clusters avec DBSCAN	34
3.25	Clusters avec k-means	34
3.26	Illustration de comparaison	34
27	Graphe Densité de K et pH	38
28	Graphe Densité de P et EC	39
29	Graphe Densité de OC et Zn	39
30	Graphe Densité de Mn et OM	39
31	Boîte à moustaches de N et K	40
32	Boîte à moustaches de pH et S	40
33	Boîte à moustaches de Fe	40
34	Histogramme de N et K	41
35	Histogramme de S et pH	41
36	Histogramme de EC	41

# Introduction Générale

Après avoir effectué la première étape préliminaire de prétraitement du dataset 1, qui regroupe les informations sur les propriétés du sol dans la partie 1, nous progressons maintenant vers une phase plus avancée de notre étude. Cette étape revêt une importance cruciale et se concentre sur deux aspects majeurs : tout d'abord, une analyse approfondie de la fertilité du sol à travers l'utilisation de techniques de classification visant à assigner des catégories spécifiques aux divers types de sols. En parallèle, nous mettons en place une approche de clustering pour regrouper les propriétés du sol présentant des similitudes, facilitant ainsi l'identification de tendances et de structures intrinsèques au sein de notre ensemble de données. Cette double approche analytique nous permettra d'obtenir une vision globale et détaillée des caractéristiques du sol examiné, offrant ainsi des conclusions éclairées et des recommandations pertinentes dans le domaine de l'agriculture.

# Chapitre 1

## Analyse et Pretraitemet du dataset

### 1.1 Objectifs

Afin de pouvoir appliquer des modèles de classification sur notre jeu de données, il est impératif d'y opérer un prétraitement afin d'obtenir des résultats corrects et précis. C'est pour cela que nous allons appliquer les différentes méthodes de prétraitements vu dans la partie 1 du projet pour obtenir un dataset en parfaite condition pour y appliquer nos algorithmes d'apprentissage automatique.

### 1.2 Visualisation du contenu du dataset

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	OM	Fertility
0	138	8.6	560	7.46	0.62	0.70	5.90	0.24	0.31	0.77	8.71	0.11	1.2040	0
1	213	7.5	338	7.62	0.75	1.06	25.40	0.30	0.86	1.54	2.89	2.29	1.8232	0
2	163	9.6	718	7.59	0.51	1.11	14.30	0.30	0.86	1.57	2.70	2.03	1.9092	0
3	157	6.8	475	7.64	0.58	0.94	26.00	0.34	0.54	1.53	2.65	1.82	1.6168	0
4	270	9.9	444	7.63	0.40	0.86	11.80	0.25	0.76	1.69	2.43	2.26	1.4792	1
5	220	8.6	444	7.43	0.65	0.72	11.70	0.37	0.66	0.90	2.19	1.82	1.2384	0

TABLE 1.1 – Description de dataset-1-

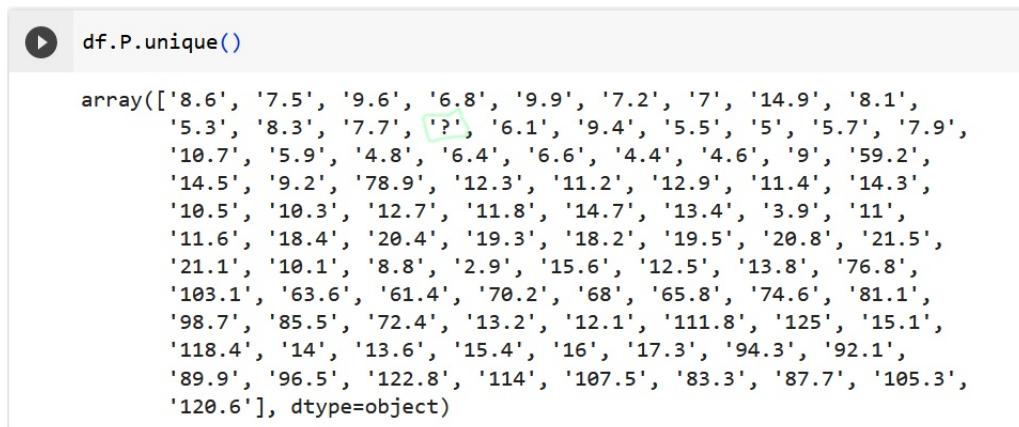
### 1.3 Traitement des valeurs manquantes :

Les valeurs manquantes sont très gênantes dans le cadre de la classification, c'est pour cela que nous nous devons d'y remédier. Notre jeu donné heureusement ne contient pas beaucoup de valeurs manquantes, Il est à noter que la colonne OC et Cu a une valeur de Non-Null Count inférieure aux autres, indiquant qu'il y a une observation manquante dans cette colonne. donc on a le supprimer. La colonne P semble contenir des objets, ce qui pourrait nécessiter une vérification pour s'assurer qu'elle est correctement interprétée.

Column	Non-Null-Count	Dtype	Unique-Value-Count
N	885 non-null	int64	61
P	885 non-null	object	93
K	885 non-null	int64	63
pH	885 non-null	float64	107
EC	885 non-null	float64	71
OC	884 non-null	float64	69
S	885 non-null	float64	153
Zn	885 non-null	float64	70
Fe	885 non-null	float64	387
Cu	884 non-null	float64	167
Mn	885 non-null	float64	429
B	885 non-null	float64	127
OM	885 non-null	float64	68
Fertility	885 non-null	int64	3

TABLE 1.2 – Description des colonnes du dataset

Après l'affichage des valeurs uniques de chaque attribut on constate qu'on doit convertir les valeurs erronées dans la colonne P par (nan)



```
df.P.unique()

array(['8.6', '7.5', '9.6', '6.8', '9.9', '7.2', '7', '14.9', '8.1',
       '5.3', '8.3', '7.7', '?', '6.1', '9.4', '5.5', '5', '5.7', '7.9',
       '10.7', '5.9', '4.8', '6.4', '6.6', '4.4', '4.6', '9', '59.2',
       '14.5', '9.2', '78.9', '12.3', '11.2', '12.9', '11.4', '14.3',
       '10.5', '10.3', '12.7', '11.8', '14.7', '13.4', '3.9', '11',
       '11.6', '18.4', '20.4', '19.3', '18.2', '19.5', '20.8', '21.5',
       '21.1', '10.1', '8.8', '2.9', '15.6', '12.5', '13.8', '76.8',
       '103.1', '63.6', '61.4', '70.2', '68', '65.8', '74.6', '81.1',
       '98.7', '85.5', '72.4', '13.2', '12.1', '111.8', '125', '15.1',
       '118.4', '14', '13.6', '15.4', '16', '17.3', '94.3', '92.1',
       '89.9', '96.5', '122.8', '114', '107.5', '83.3', '87.7', '105.3',
       '120.6'], dtype=object)
```

FIGURE 1.1 – Les valeurs uniques de l'attribut P

Il existe plusieurs méthodes pour les traiter. soit les remplacer par la moyenne ou par la mediane

## 1.4 Traitement des valeurs aberrantes

Les valeurs aberrantes, également appelées outliers, peuvent avoir plusieurs effets sur les analyses de données, Influence sur les mesures de tendance centrale, Impact sur la dispersion des données, Déformation des corrélations...

Dans notre jeu de données, nous avons plusieurs :

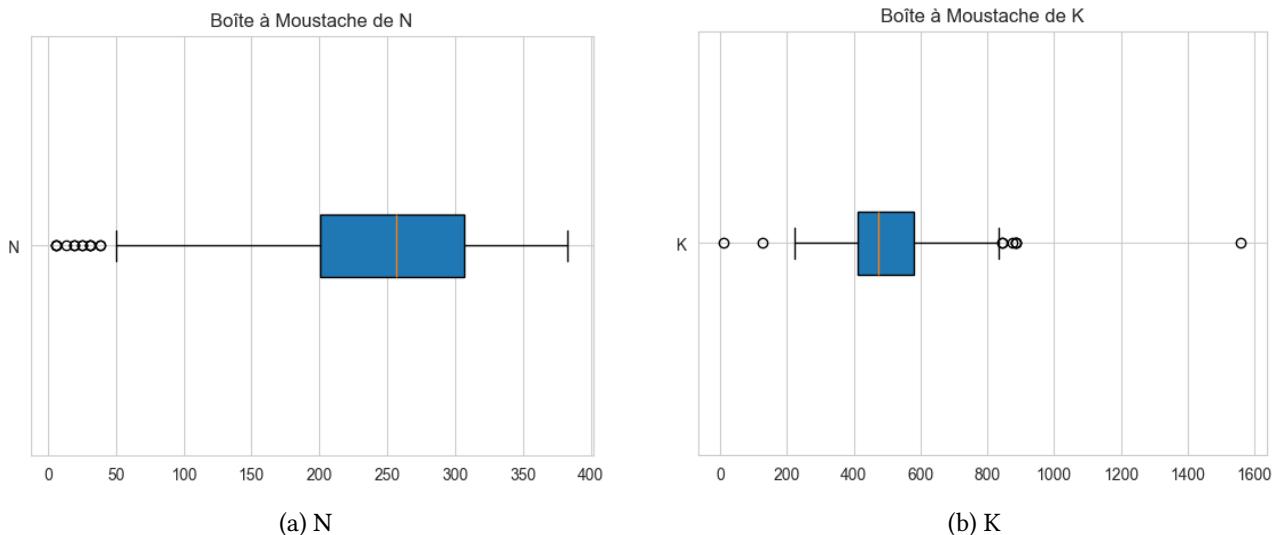


FIGURE 1.2 – Boîte à moustache de N et K

Pour les traiter, nous pouvons les remplacer soit par la moyenne ou par la médiane.

Le choix entre la moyenne et la médiane pour traiter les valeurs aberrantes dépend de la nature spécifique de nos données et des objectifs de notre analyse.

## 1.5 Réduction des données (élimination des redondances)

### Réduction verticale

Identifiez et supprimez les variables qui présentent une corrélation élevée entre elles, car elles peuvent apporter des informations similaires, on supprime donc OC et OM.

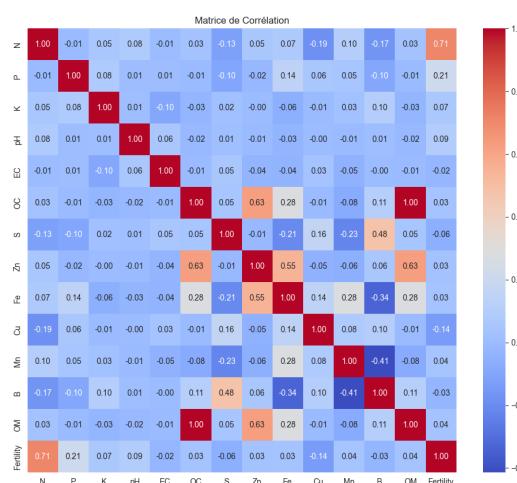


FIGURE 1.3 – La matrice de corrélation

## Réduction horizontales

Il y a 3 lignes en double que nous devons supprimer.  
nous aurons donc 880 lignes et 12 colonnes au final.

## 1.6 Normalisation des données :

La normalisation des données est un processus visant à ajuster l'échelle des variables dans un ensemble de données. L'objectif principal de la normalisation est de rendre les données comparables et de faciliter le processus d'analyse en garantissant que chaque variable contribue de manière équitable aux calculs.

### Méthode Min-Max

Cette méthode redimensionne les valeurs d'une variable pour qu'elles tombent dans une plage spécifique, généralement entre 0 et 1. La formule est la suivante :

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

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	OM
0	0.264	0.435	0.552	0.459	0.56	0.513	0.371	0.25	0.00967	0.338	0.449	0.0568	0.513
1	0.489	0.351	0.19	0.606	0.733	0.82	0.488	0.338	0.0629	0.721	0.145	0.606	0.82
2	0.339	0.511	0.81	0.578	0.413	0.863	0.965	0.338	0.0629	0.736	0.135	0.606	0.863
3	0.321	0.298	0.413	0.624	0.507	0.718	0.488	0.397	0.0319	0.716	0.133	0.606	0.718
4	0.661	0.534	0.363	0.615	0.267	0.65	0.788	0.265	0.0532	0.796	0.121	0.606	0.65

TABLE 1.3 – Portion Résultat Normalisation Min-Max

### Méthode z-score

Cette méthode transforme les données pour qu'elles aient une moyenne de 0 et un écart-type de 1. La formule est la suivante :

$$Z = \frac{X - \mu}{\sigma}$$

	<i>N</i>	<i>P</i>	<i>K</i>	pH	EC	OC	<i>S</i>	<i>Zn</i>	<i>Fe</i>	<i>Cu</i>	<i>Mn</i>
0	-1.550	-0.101	0.522	-0.101	0.539	0.365	-0.348	-1.150	-1.440	-0.395	0.044
1	-0.512	-0.474	-1.354	0.606	1.462	1.580	0.294	-0.647	-1.229	1.281	-1.406
2	-1.204	0.238	1.857	0.473	-0.242	1.749	2.936	-0.647	-1.229	1.346	-1.454
3	-1.287	-0.712	-0.196	0.694	0.255	1.175	0.294	-0.312	-1.352	1.259	-1.466
4	0.276	0.339	-0.458	0.650	-1.023	0.905	1.958	-1.067	-1.268	1.607	-1.521

TABLE 1.4 – Portion Résultat normalisées Z-score

## 1.7 Récapitulatif

Le problème	Le choix
Valeurs manquantes	Nous avons remplacé les valeurs NaN par la moyenne.
Valeurs aberrantes	Nous avons choisi la moyenne car elle élimine les valeurs aberrantes mieux que la médiane pour notre jeu de données.
Réduction verticale	Suppression des colonnes OC et OM
Réduction horizontale	3 lignes en double
Normalisation	Min-max

TABLE 1.5 – Récapitulatif des traitements effectués.

### Remarque

Tout est mis en annexe pour plus de détails sur le jeu de données et sa visualisation.

## 1.8 Conclusion

Ce premier chapitre de notre étude nous a introduit aux différentes étapes nécessaires pour nettoyer une base de données et spécifiquement avec des données statistiques.

Nous avons suivi plusieurs approches afin d'obtenir une version propre et efficace de notre jeu de données.

Ces préparations sont cruciales pour garantir la fiabilité et la validité des analyses ultérieures.

# Chapitre 2

## Analyse supervisée : Classification

La classification est une forme d'analyse de données qui extrait des modèles décrivant des classes de données importantes permettant d'étiqueter de nouvelles données en se basant sur des exemples déjà vus. De nombreuses méthodes ont été proposées afin de développer des techniques de classification et de prédiction capables de gérer de grandes quantités de données. Chacune de ces méthodes comporte des avantages et des inconvénients la rendant plus ou moins adaptée à un type de dataset. Ainsi, dans ce chapitre, on va construire un classificateur permettant de prédire si une instance représentant un échantillon du sol est faiblement (low), moyennement (medium), ou fortement (high) fertile. et on va réaliser ça à l'aide des trois algorithmes de classification .

### 2.1 Algorithme K-NN

L'intuition derrière l'algorithme des K plus proches voisins est l'une des plus simples de tous les algorithmes de Machine Learning supervisé elle consiste à trouver les K voisins les plus proches selon la distance calculée. Parmi ces K voisins, comptez le nombre de points appartenant à chaque catégorie. en fin on attribuez le nouveau point à la catégorie la plus présente parmi ces K voisins. Voici l'algorithme de K-NN :

- 
- 1: **Entrée :** Ensemble de données  $D$ , point de test  $x$ , nombre de voisins  $k$
  - 2: **Sortie :** Étiquette attribuée à  $x$   
**forall** *points p dans D do*  
3:  
    Calculer la distance entre  $x$  et  $p$
  - 4:
  - 5: Trier les distances calculées en ordre croissant
  - 6: Sélectionner les  $k$  plus proches voisins
  - 7: **Pour la classification :**  
8: Compter le nombre de voisins appartenant à chaque classe  
9: Attribuer à  $x$  la classe majoritaire parmi les voisins
  - 10: **Pour la régression :**  
11: Calculer la valeur moyenne (ou médiane) des étiquettes des  $k$  voisins  
12: Attribuer à  $x$  cette valeur moyenne comme prédiction
-

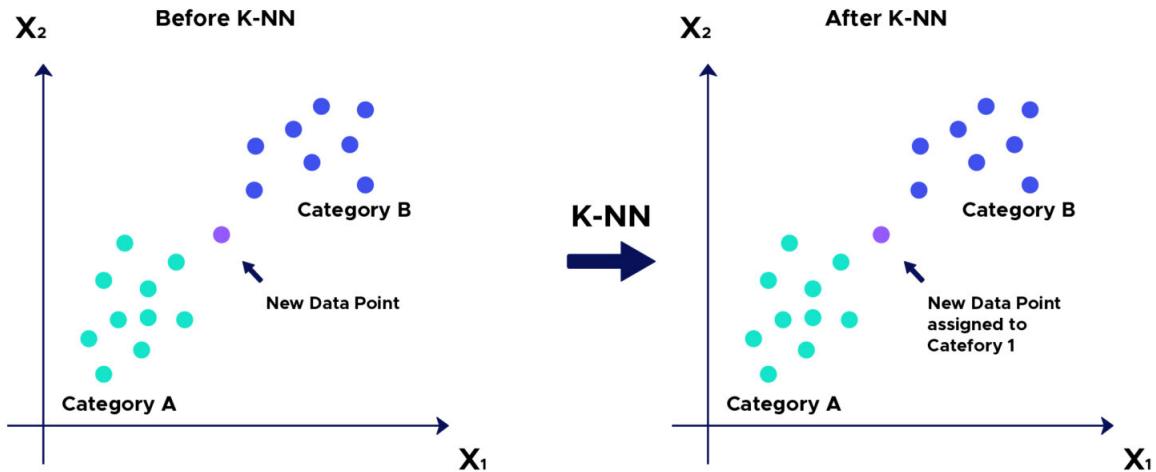


FIGURE 2.1 – Algorithme K-NN

### 2.1.1 Méthodes de Calcul de Distance pour K-NN

Méthode	Équation	Avantages	Inconvénients
Euclidienne	$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$	Simplicité, largement utilisée	Sensible à l'échelle, affectée par les dimensions
Manhattan (Cityblock)	$d(x, y) = \sum_{i=1}^n  x_i - y_i $	Robuste aux valeurs aberrantes, adaptée aux données de grande dimension	Ne prend pas en compte les relations diagonales
Minkowski	$d(x, y) = \left( \sum_{i=1}^n  x_i - y_i ^p \right)^{\frac{1}{p}}$	Flexibilité pour ajuster la sensibilité, inclut Euclidienne et Manhattan comme cas particuliers	Sensible au choix du paramètre $p$
Chebyshev	$d(x, y) = \max_i  x_i - y_i $	Robuste aux valeurs aberrantes, mesure la différence maximale	Sensible aux dimensions dominantes
Cosine	$d(x, y) = \frac{x \cdot y}{\ x\  \cdot \ y\ }$	Invariant à l'échelle, adaptée aux données textuelles ou vectorielles	Ignorance des magnitudes, sensibilité aux dimensions non informatives

TABLE 2.1 – Méthodes de Calcul de Distance pour K-NN

## 2.2 Arbre de décision

Lors de l'apprentissage d'un modèle sur les relations caractéristique-cible, un arbre est développé à partir d'un nœud racine (parent) (toutes les données contenant des relations caractéristiquecible), qui

est ensuite divisé de manière récursive en nœuds enfants (sous-ensemble de l'ensemble des données) de manière binaire.

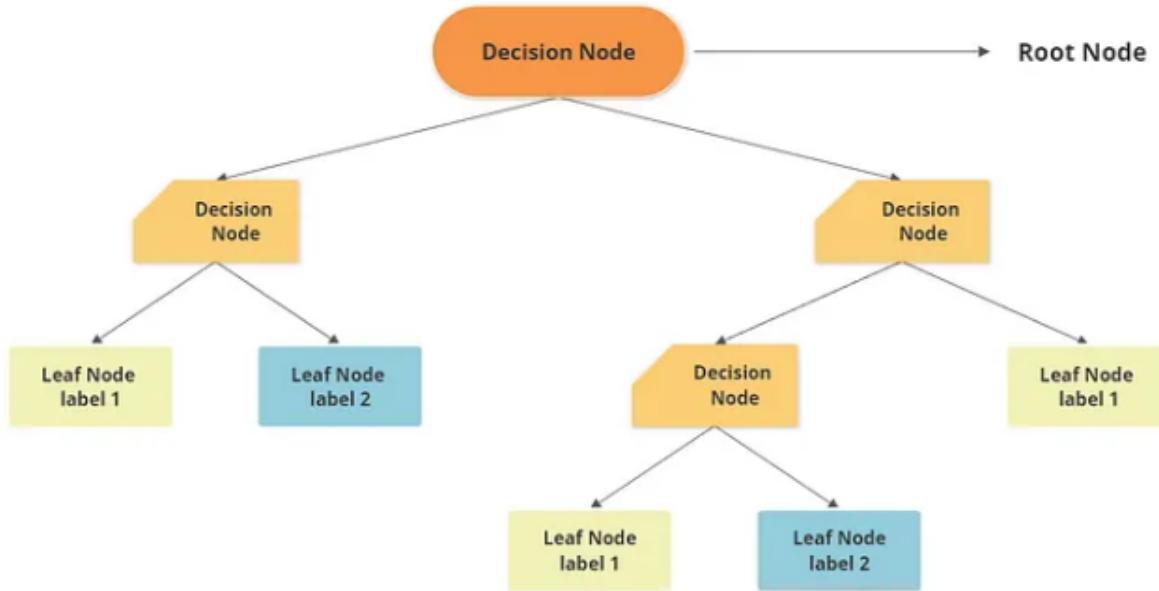


FIGURE 2.2 – Arbre de decision

Chaque division est effectuée sur une seule caractéristique du nœud parent, à une valeur seuil 1 souhaitée de la caractéristique. Par exemple, lors de chaque division du nœud parent, on passons au nœud de gauche (avec le sous-ensemble de données correspondant) si une caractéristique est inférieure au seuil, et au nœud de droite sinon. et pour la décision de division utilisons une fonction de cout.

### 2.2.1 Les Fonction de cout

Il y'a deux criteres de cout utiliser dans l'arbre de decision , on commence par expliquer les criteres puis on présent un exemple pratique qui compare les deux critères.

#### Gini impurity

L'impureté gini est calculée à l'aide de la formule suivante :  $GiniIndex = 1 - \sum_{j=1}^k p_j^2$  où  $p_j$  est la probabilité de la classe  $j$ . L'impureté de Gini mesure la fréquence à laquelle un élément de l'ensemble de données sera mal étiqueté lorsqu'il est étiqueté de manière aléatoire. La valeur minimale de l'indice de Gini est de 0. Cela se produit lorsque le nœud est pur, ce qui signifie que tous les éléments contenus dans le nœud sont d'une classe unique. Par conséquent, ce nœud ne sera pas divisé à nouveau. Ainsi, la division optimale est choisie par les caractéristiques ayant le plus faible indice de Gini. De plus, il obtient la valeur maximale lorsque la probabilité des deux classes est la même.  $Gini_{min} = 1 - (0.5^2 + 0.5^2) = 0$   $Gini_{max} = 1 - (0.5^2 + 0.5^2) = 0.5$

## Entropy

L'entropie est calculée à l'aide de la formule suivante :  $\text{Entropy} = -\sum_{j=1}^J p_j \cdot \log_2 p_j$ . Où, comme précédemment,  $p_j$  est la probabilité de la classe  $j$ . L'entropie est une mesure d'information qui indique le désordre des caractéristiques par rapport à la cible. Comme pour l'indice de Gini, la répartition optimale est choisie par la caractéristique ayant la plus faible entropie. Elle obtient sa valeur maximale lorsque la probabilité des deux classes est la même et un nœud est pur lorsque l'entropie a sa valeur minimale, qui est 0 :  $\text{Entropy}_{\min} = -1 \cdot \log_2(1) = 0$ .  $\text{Entropy}_{\max} = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5) = 1$

### 2.2.2 Gini vs Entropy

L'indice de Gini et l'entropie présentent deux différences principales : – L'indice de Gini a des valeurs comprises dans l'intervalle  $[0, 0.5]$  alors que l'intervalle de l'entropie est  $[0, 1]$ . Dans la figure suivante, les deux sont représentés. L'indice de Gini a également été représenté multiplié par deux pour voir concrètement les différences entre eux.

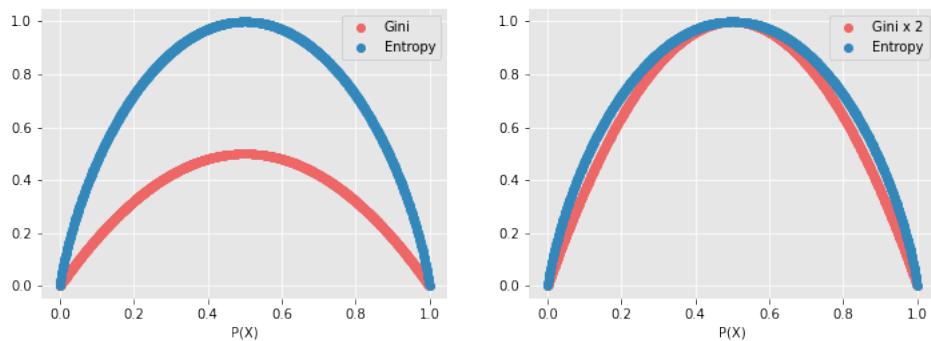


FIGURE 2.3 – Gini vs Entropy

–Du point de vue computationnel, l'entropie est plus complexe puisqu'elle fait appel à des logarithmes et, par conséquent, le calcul de l'indice de Gini sera plus rapide.

## 2.3 Forêt Aléatoire

La Forêt Aléatoire, est une collection d'arbres décisionnels. L'algorithme Random Forest est construit sur l'idée du vote par des apprenants "faibles" (arbres décisionnels), ce qui donne l'analogie avec les arbres qui composent une forêt.

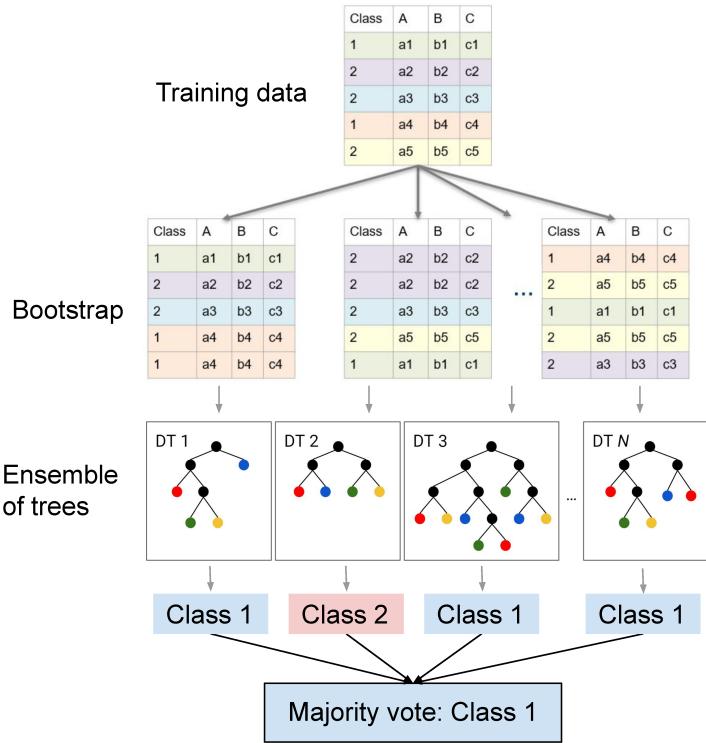


FIGURE 2.4 – Arbre de décision

Voici l'algorithme de construction de random forest :

---

**Algorithm 1** Random Forest

---

- 1: **Entrée** : Ensemble de données  $D$ , nombre d'arbres  $T$ , nombre de caractéristiques à considérer à chaque division  $m$
- 2: **Sortie** : Modèle de Random Forest
- 3: **procédure** CONSTRUIRERANDOMFOREST( $D, T, m$ ) **for**  $t = 1$  to  $T$  **do**
- 4:      $D_t \leftarrow$  Sous-ensemble aléatoire de  $D$  par échantillonnage avec remplacement
- 5:      $T_t \leftarrow$  Arbre de décision construit sur  $D_t$  en considérant  $m$  caractéristiques à chaque division
- 6:
- 7:     **return** Ensemble des arbres  $\{T_1, T_2, \dots, T_T\}$
- 8:
- 9: **procédure** RANDOMFORESTPREDICT( $\{T_1, T_2, \dots, T_T\}, x$ )
- 10:     **Pour la classification :**
- 11:         Compter les votes de chaque arbre pour la classe de  $x$
- 12:         Attribuer à  $x$  la classe majoritaire parmi les votes des arbres
- 13:     **Pour la régression :**
- 14:         Calculer la valeur prédictée par chaque arbre pour  $x$
- 15:         Attribuer à  $x$  la moyenne (ou médiane) des valeurs prédictées par les arbres
- 16: **end procédure**

---

## 2.4 Etude Experimentale

### 2.4.1 Objectifs :

Dans cette section, nous allons procéder à l'évaluation de nos modèles avec différentes métriques, une comparaison entre les hyperparamètres et en fin, une comparaison avec les modèles de Sklearn.

### 2.4.2 Les mesures de comparaison

L'évaluation des modèles de classification est cruciale pour comprendre leur performance et leur capacité à effectuer des prédictions précises. Différentes métriques d'évaluation sont utilisées pour mesurer différents aspects du comportement d'un modèle sur un ensemble de données. Dans ce contexte, le tableau 2.2 présente plusieurs mesures couramment utilisées pour évaluer la performance des modèles de classification.

Mesure	Symbol	Description et Rôle
Exactitude	$ACC$	Mesure la proportion des prédictions correctes parmi toutes les prédictions. Utile pour évaluer globalement la performance d'un modèle.
Spécificité	$SPEC$	Mesure la capacité du modèle à identifier les vrais négatifs parmi tous les exemples réels négatifs. Particulièrement important dans les cas où la classe négative est critique.
Précision	$PREC$	Indique la proportion des vrais positifs parmi toutes les instances prédictes comme positives. Utile pour évaluer la pertinence des prédictions positives.
Rappel	$REC$	Mesure la proportion des vrais positifs parmi toutes les instances réellement positives. Fournit des informations sur la capacité du modèle à identifier toutes les instances positives.
F-Score	$F1$	La moyenne harmonique de la précision et du rappel. Donne un équilibre entre ces deux mesures. Utile lorsque l'on cherche un compromis entre la précision et le rappel.

TABLE 2.2 – Description des Mesures d'Évaluation de Classification

### 2.4.3 Le choix de distance :Algorithme K-NN

Nous avons mis en œuvre différentes méthodes de calcul de distance, puis avons pris une instance pour tester le modèle K-NN avec ces différentes distances. Les résultats graphiques montrent que les voisins sont représentés en bleu.

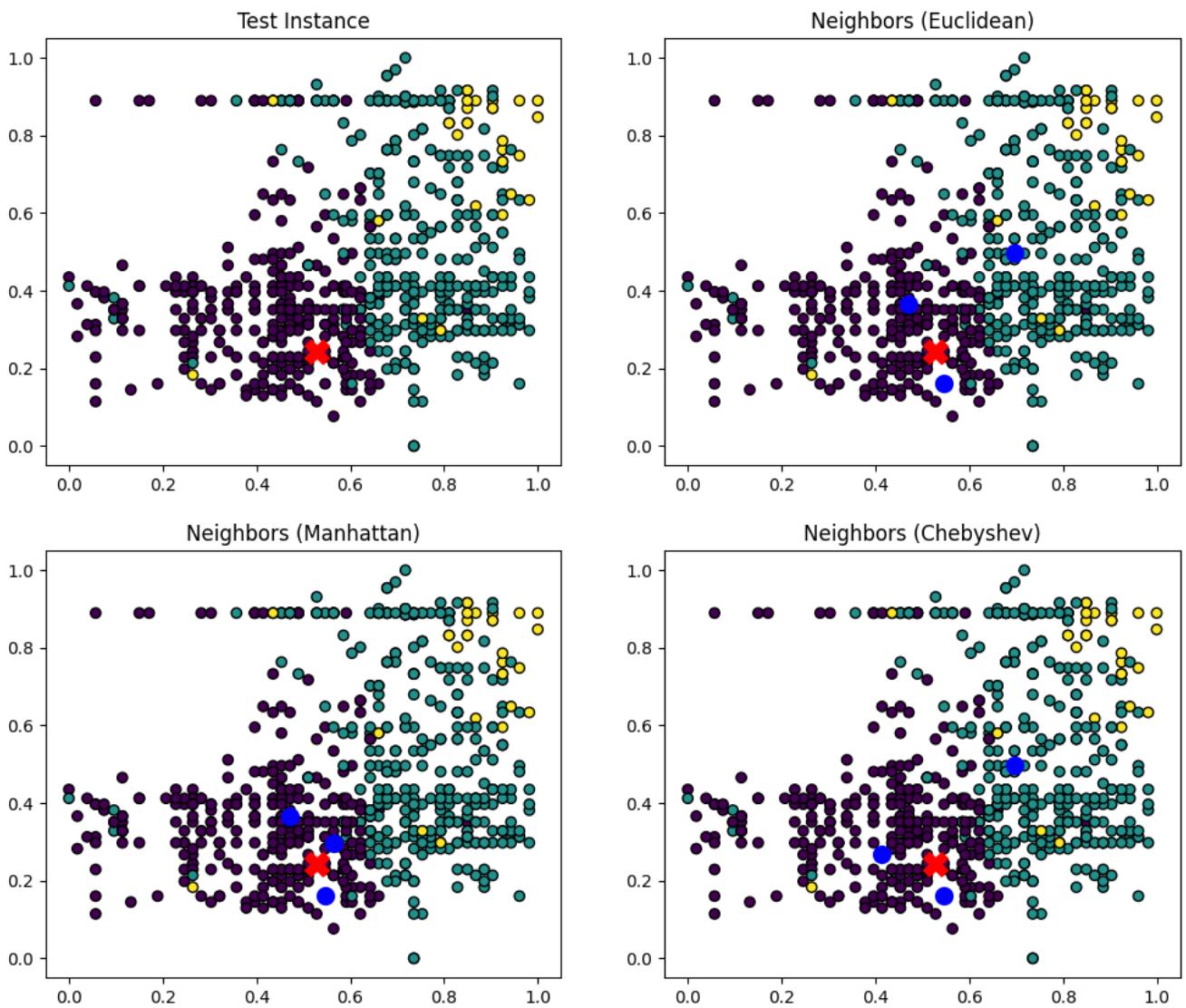


FIGURE 2.5 – Choix de distance :Algorithme K-NN

D'après les graphiques, il est observé que la distance de Manhattan a produit des résultats plus précis, avec les trois voisins appartenant à la véritable classe.

#### 2.4.4 Matrice de confusion

Une matrice de confusion est une table utilisée pour évaluer les performances d'un modèle de classification, en particulier dans le domaine de l'apprentissage automatique et de la statistique. Elle compare les prédictions du modèle avec les valeurs réelles (étiquettes de classe) d'un ensemble de données. En analysant la matrice de confusion, on peut calculer diverses métriques d'évaluation du modèle, telles que la précision, le rappel, la spécificité, le taux de faux positifs, le taux de faux négatifs, etc.

Voici les matrices de confusion de nos trois modèles :

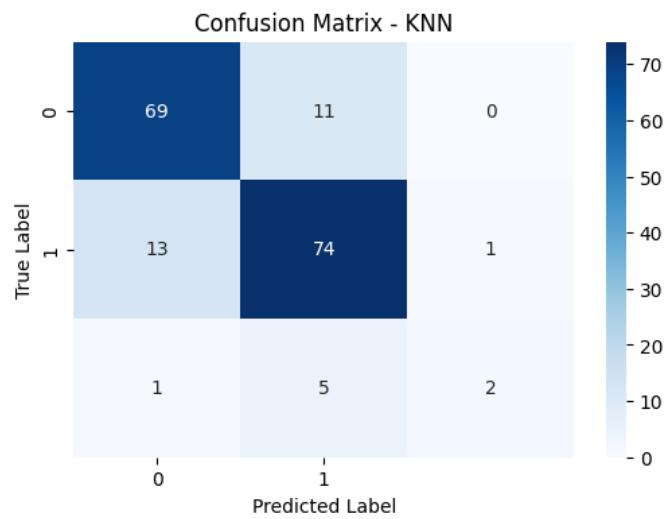


FIGURE 2.6 – Matrice de confusion modele KNN

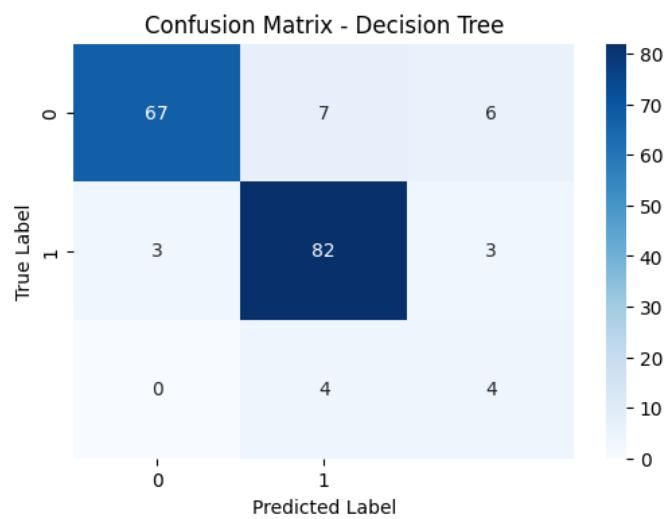


FIGURE 2.7 – Matrice de confusion modele decision tree

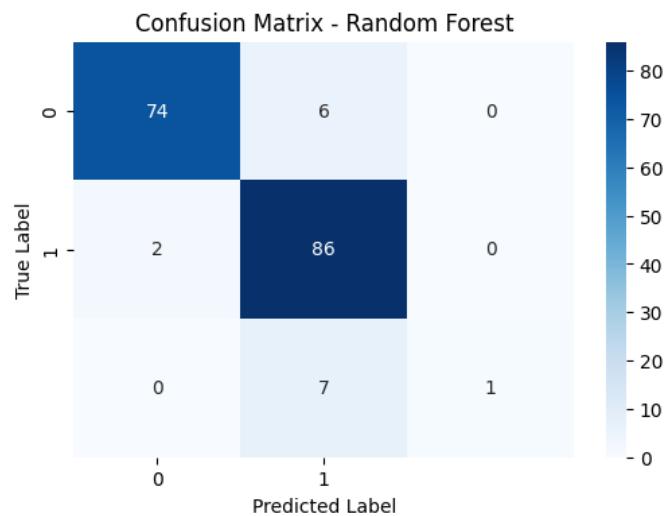


FIGURE 2.8 – Matrice de confusion modele Random Forest

#### 2.4.5 Comparaison entre les trois modeles de classification

Nous avons intégré toutes les mesures spécifiées dans notre programme. Pour l'algorithme K-NN, nous avons opté pour la distance de Manhattan, car elle a démontré une précision supérieure lors du calcul des voisins. En ce qui concerne l'arbre de décision, nous avons choisi l'indice de pureté de Gini, car il a montré de meilleurs résultats lors de l'exécution. Voici les résultats obtenus :

Modèle	Exactitude	Précision	Rappel	F-Score
KNN	0.8239	0.8193	0.8239	0.8216
Decision Tree	0.8693	0.8899	0.8693	0.8795
Random Forest	0.9148	0.9204	0.9148	0.9176

TABLE 2.3 – Résultats d'Évaluation des Modèles

**Comparaison :** Les résultats montrent que le modèle Random Forest a la meilleure performance en termes d'exactitude, de précision, de rappel et de F-Score parmi les trois modèles évalués. Cependant, la différence de performance entre les modèles KNN, Decision Tree et Random Forest semble être relativement faible.

## 2.5 Conclusion

Dans cette partie du projet, nous nous sommes familiarisés avec le concept de classification, son importance dans le domaine du datamining, et le fonctionnement de certain modèle de classification. Tel que K-NN, les arbres de décisions (decision trees) et forêt d'arbres décisionnels (Random forest). Nous avons aussi pu constater l'importance du prétraitement des données vu lors de la partie deux, car sans elle notre classification n'aurait pas été des plus performantes. Cependant, l'apport de la classification au datamining restant indéniable, elle reste limitée par le fait que les classes doivent être prédéfinies à l'avance. Classes qui ne peuvent pas toujours être fournies. C'est pourquoi il existe un autre sous domaine du machine learning appelé apprentissage non supervisé qui peut remédier à ce problème et que nous verrons plus en détail dans le prochain chapitre.

# Chapitre 3

## Analyse non supervisée : clustering

Dans cette partie, nous nous intéressons au clustering, on n'essaie pas d'apprendre une relation de corrélation entre un ensemble de features X d'une observation et une valeur à prédire Y, comme c'est le cas pour l'apprentissage supervisé. L'apprentissage non supervisé va plutôt trouver des patterns dans les données. Notamment, en regroupant les choses qui se ressemblent.

Notre objectif est de découvrir la similarité entre ces instances afin de les regrouper dans leur cluster approprié. La structure du jeu de données que nous utilisons est bien détaillée dans la première partie. Sachant que nous avons l'attribut "Fertility", nous allons l'éliminer et tenter de trouver la classe appropriée pour chaque instance du dataset. Nous allons tester et comparer différents algorithmes de clustering (K-means et DBSCAN) afin de trouver des partitionnements intéressants à partir du dataset 1.

### 3.1 Algorithme de clustering basé partitionnement

#### 3.1.1 L'Algorithme K-means

**K-means** est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du dataset. Ainsi les données similaires se retrouveront dans un même cluster.

l'algorithme K-Means a besoin d'un moyen de comparer le degré de similarité entre les différentes observations, le plus connue pour les cas de clustering, **la distance Euclidienne** :

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

La complexité de cet algorithme est  $O(t \cdot n \cdot k \cdot d)$

tel que t est le nombre d'itération, n est la taille des données, k est le nombre de clusters et d est la dimension des données.

voici son pseudo code :

---

**Algorithm 2** K-means Algorithm

---

- 1: **Input** : Dataset  $X$ , Number of clusters  $k$
- 2: **Output** : Cluster centroids  $\{c_1, c_2, \dots, c_k\}$ , Assignments  $\{a_1, a_2, \dots, a_n\}$
- 3: Randomly initialize cluster centroids  $\{c_1, c_2, \dots, c_k\}$  **repeat**
- | each data point  $x_i$
- | **until for do**
- | ;
- 4: Compute distances to centroids :  $d_{ij} = \text{distance}(x_i, c_j)$  for  $j = 1$  to  $k$
- 5: Assign  $x_i$  to the cluster with the closest centroid :  $a_i = \arg \min_j d_{ij}$
- 6: **for each cluster  $j$  do**
- | **repeat**
- |   | Update centroid  $c_j$  as the mean of all data points assigned to cluster  $j$
- 8: **Convergence** =0

---

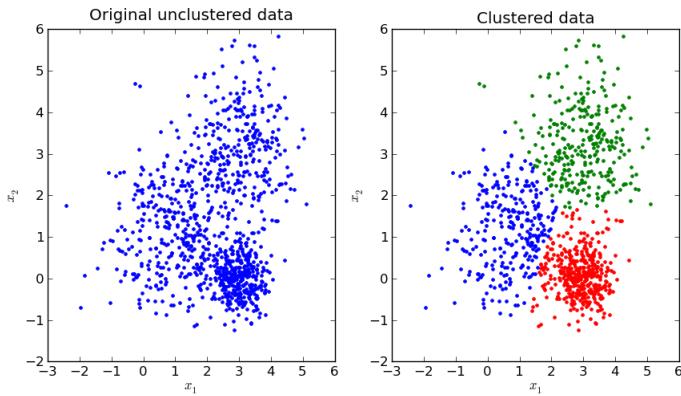


FIGURE 3.1 – Exemple illustratif de K-means

Les problèmes rencontrés avec l'algorithme K-means concerne plusieurs aspects :

1. le choix du nombre de clusters ( $k$ )
2. l'initialisation des centroides
3. le nombre d'itérations
4. Le choix d'autres paramètres pour améliorer le résultat

Donc, pour définir les paramètres adéquats pour notre jeu de données figure 3.2 lors de l'application de k-means, il faut mener des expérimentations pour chaque paramètre.

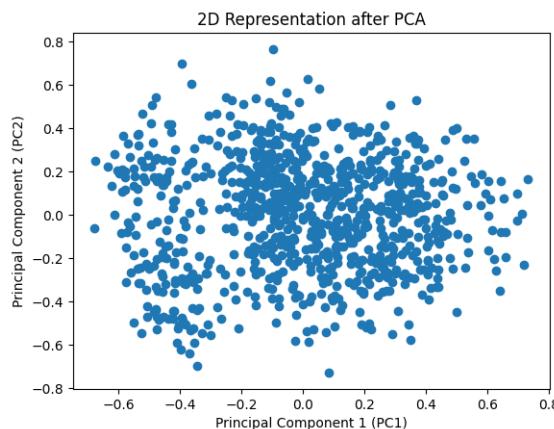


FIGURE 3.2 – Dataset visualisation

En examinant la distribution de notre ensemble de données d'après la figure 3.2, il est évident que les clusters ne sont plus visibles et que les points sont rapprochés. Ainsi, l'extraction de clusters à partir de cet ensemble de données est quelque peu difficile en raison de sa taille importante (840 lignes en 3 clusters seulement) et de sa distribution complexe.

### 3.1.2 Choisir K : le nombre de clusters :

Choisir un nombre de cluster K n'est pas forcément intuitif. La méthode la plus usuelle pour choisir le nombre de clusters est de lancer K-Means avec différentes valeurs de K et de calculer la variance des différents clusters. La variance est la somme des distances entre chaque centroid d'un cluster et les différentes observations incluses dans le même cluster. La variance des clusters se calcule comme suit :

$$V = \sum_j \sum_{x_i \rightarrow c_j} D(c_j, x_i)^2$$

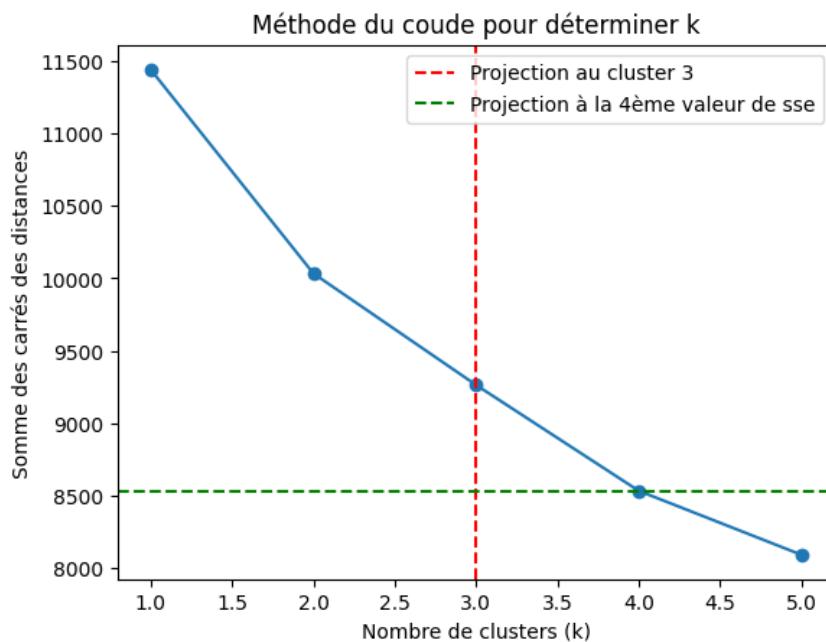


FIGURE 3.3 – La courbe du coude

#### Analyse

On remarque sur ce graphique de la figure 3.2, la forme d'un bras où le point le plus haut représente l'épaule et le point où K vaut 5 représente l'autre extrémité. Le nombre optimal de clusters est le point représentant le coude. Ici le coude peut être représenté par K valant 3 ou 4. C'est le nombre optimal de clusters.

On prend k=3 dans ce cas.

#### Remarque :

A noter que le domaine d'application consiste à considérer la signification pratique des clusters dans le contexte de notre application. Parfois, un nombre spécifique de clusters peut avoir plus de sens en fonction de la nature des données et des objectifs de l'analyse.

### 3.1.3 l'initialisation des centroïdes

L'initialisation des centroides dans l'algorithme K-means peut se faire de différentes manières, et le choix de la méthode peut influencer la convergence et la performance de l'algorithme.

1. Initialisation aléatoire : Les centroides sont choisis de manière aléatoire parmi les points du jeu de données.
2. Initialisation k-means++ : Cette méthode améliore l'initialisation aléatoire en s'assurant que les centroides initiaux sont bien répartis dans l'espace des données.

Approche	Temps d'exécution de k-means	Tentatives Pour un bon résultat
aléatoire	0.10 s	9
k-means++	0.06 s	5

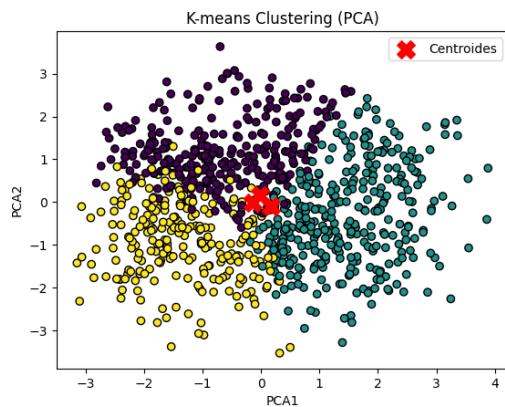


FIGURE 3.4 – Aleatoirement

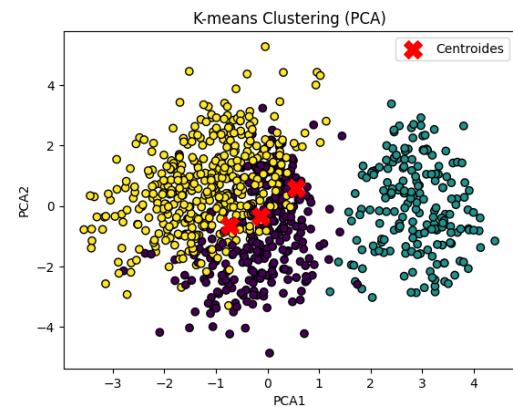


FIGURE 3.5 – Avec k-means++

#### Analyse

Nous avons appliqué les deux approches et avons constaté que les deux donnent des résultats plus ou moins satisfaisants. Cependant, il y a quelques avantages dans l'initialisation k-means++. Elle produit des résultats plus rapides en termes de temps d'exécution par rapport à la méthode aléatoire. De plus, elle converge vers de meilleurs résultats après un nombre d'itérations inférieur par rapport à l'autre méthode. En outre, les centroides sont bien distribués dans le dataset, tandis que pour l'approche aléatoire, ils sont très proches.

Donc on a choisi l'initialisation k-means++.

### 3.1.4 le nombre d'itérations et la convergence

Nous cherchons à identifier les meilleures combinaisons d'itérations et de critères de convergence pour l'algorithme K-means en termes **d'homogénéité**, de **complétude**, de **Vmesure** et de **silhouette**. Cela est réalisé en effectuant plusieurs exécutions et en ne conservant que le résultat présentant les valeurs les plus élevées pour chaque combinaison.

Iterations	Convergence	Homogeneity	Completeness	V_measure	Silhouette
50	0.1	0.1603	0.1235	0.1395	0.3929
50	0.01	0.1851	0.1467	0.1637	0.4116
50	0.0001	0.0113	0.0090	0.0100	0.4023
50	1e-05	0.0316	0.0249	0.0279	0.3907
100	0.1	0.2116	0.1726	0.1901	0.3863
100	0.01	0.1646	0.1276	0.1438	0.3901
100	0.001	0.1531	0.1195	0.1342	0.3935
100	0.0001	0.2094	0.1623	0.1828	0.3927
150	0.1	0.1627	0.1255	0.1417	0.3898
150	0.01	0.1507	0.1184	0.1326	0.4112
150	0.0001	0.1753	0.1365	0.1535	0.3924
200	0.1	0.1014	0.0805	0.0898	0.3811
200	0.01	0.1529	0.1235	0.1366	0.3937
200	1e-05	0.1888	0.1455	0.1644	0.3918
300	0.01	0.1698	0.1309	0.1478	0.3833

TABLE 3.1 – Extrait des résultats de l’expérimentation

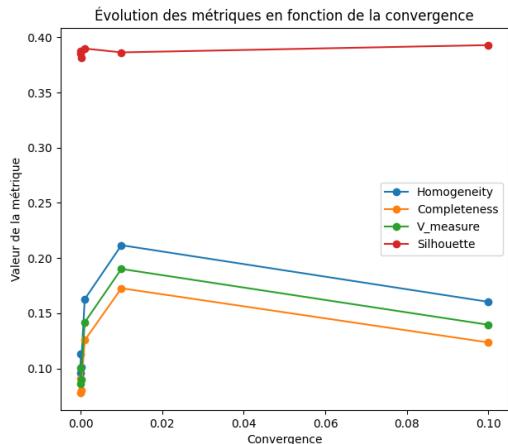


FIGURE 3.6 – Variation de convergence

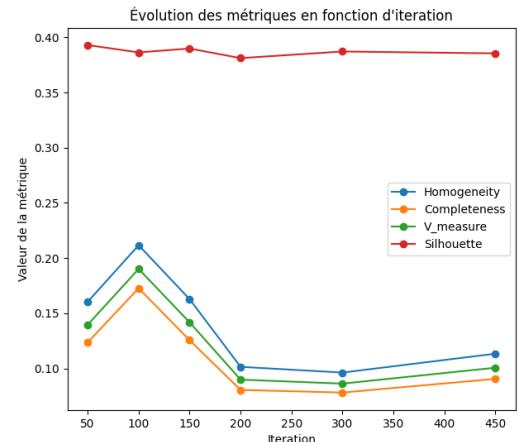


FIGURE 3.7 – Variation d’Iteration

## Analyse

D’après les graphes de convergence et d’itération présentés dans les figures 3.4 et 3.5, ainsi que le tableau 3.1, on observe clairement que l’algorithme K-means produit des résultats prometteurs lorsque la convergence est fixée à 0.01 et le nombre d’itérations à 100. En effet, ces paramètres marquent le sommet de notre progression, justifiant ainsi le choix de ces métriques pour notre jeu de données.

### 3.1.5 Visualisation des clusters

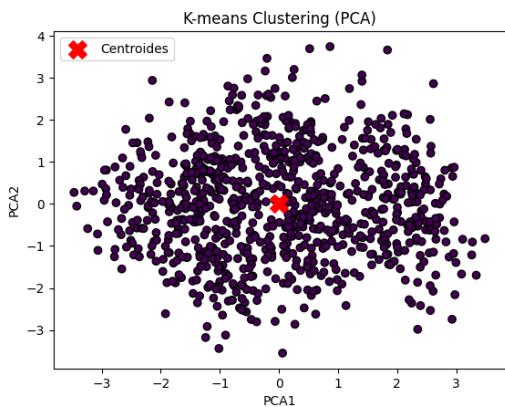


FIGURE 3.8 – Un seul Cluster (k=1)

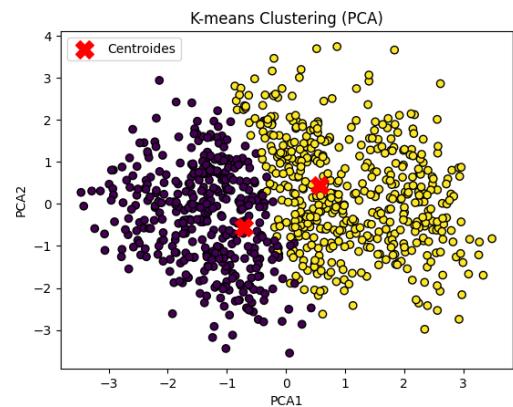


FIGURE 3.9 – Deux Clustres (k=2)

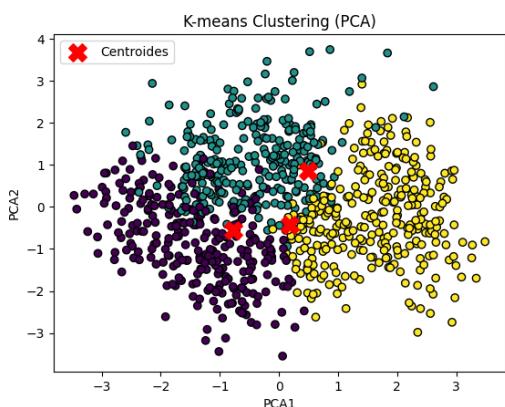


FIGURE 3.10 – Trois Clusters (k=3)

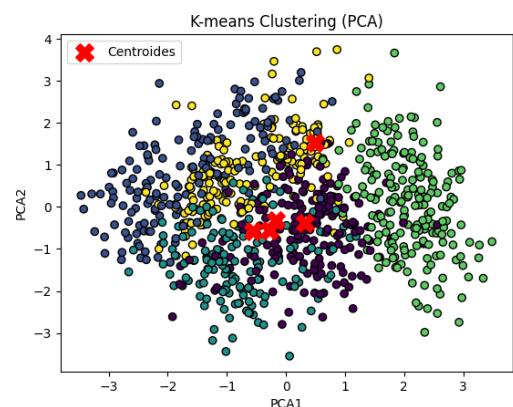


FIGURE 3.11 – Cinq Clusters (k=5)

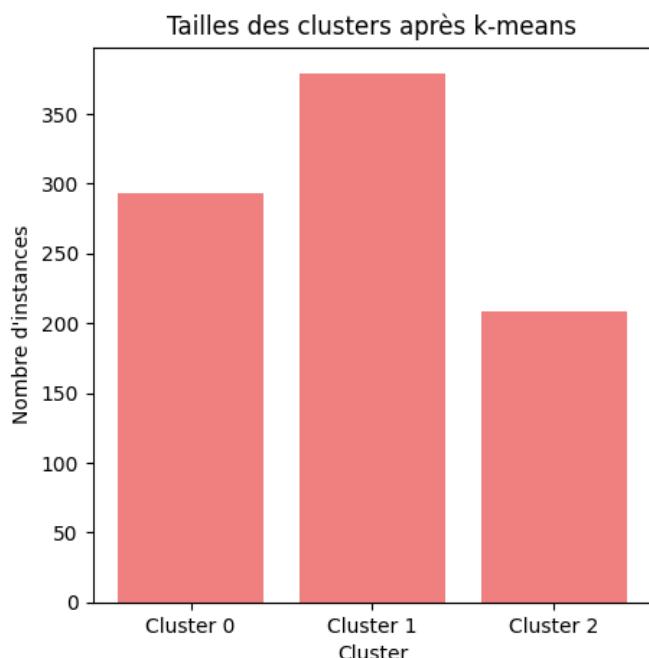


FIGURE 3.12 – Taille Clusters Pour k=3

## Analyse

En se référant à la figure 3.12 qui offre une représentation graphique de la distribution de notre ensemble de données dans le contexte de 3 clusters, obtenue grâce à la méthode k-means, il est observé que cette distribution présente une symétrie presque parfaite. Cette observation suggère une répartition équilibrée des données entre les différents clusters, ce qui pourrait indiquer une certaine homogénéité dans la composition de chaque groupe identifié par l'algorithme k-means

## Récapitulatif

K	centroïdes	iteration	convergence	Homogeneity	Completeness	Vmeasure	silhouette
3	k-means++	100	0.01	0.27	0.21	0.24	0.40

## Analyse

Une homogénéité élevée signifie que les clusters sont constitués principalement d'échantillons d'une seule classe. Une complétude élevée signifie que tous les membres d'une classe donnée sont attribués au même cluster. La V-measure est une mesure qui combine l'homogénéité et la complétude. Enfin, la silhouette mesure à quel point chaque point d'un cluster est similaire aux points du même cluster par rapport à d'autres clusters.

### 3.1.6 Etude de la Stabilité de K-means

L'algorithme K-means est sensible aux positions initiales des centroïdes, et des variations dans ces positions peuvent entraîner des résultats différents. L'idée derrière l'approche de la stabilité de K-means est d'évaluer la robustesse des résultats de l'algorithme en effectuant plusieurs exécutions avec des initialisations aléatoires différentes et en observant dans quelle mesure les résultats sont cohérents.

#### Stabilité du cluster

Un algorithme de clustering est considéré comme stable si de petites variations dans les données d'entrée ou dans les conditions initiales ne conduisent pas à des changements significatifs dans la structure des clusters résultants.

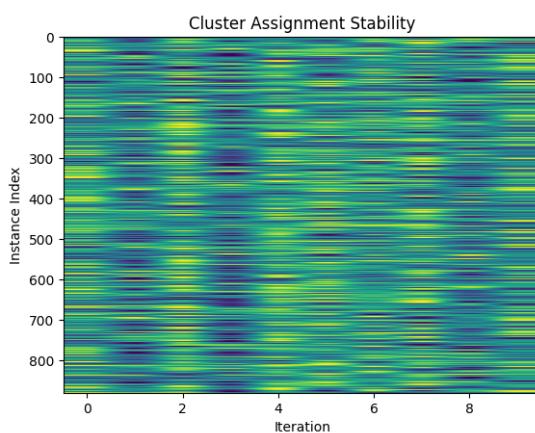


FIGURE 3.13 – Stabilité du cluster

#### stabilité des instances

Elle mesure dans quelle mesure les mêmes instances sont regroupées ensemble lorsque l'algorithme est exécuté plusieurs fois avec des conditions initiales légèrement différentes ou avec des sous-ensembles différents des données.

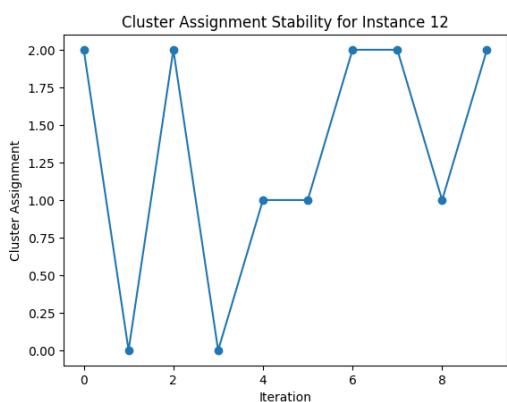


FIGURE 3.14 – Stabilité de l'instance 12

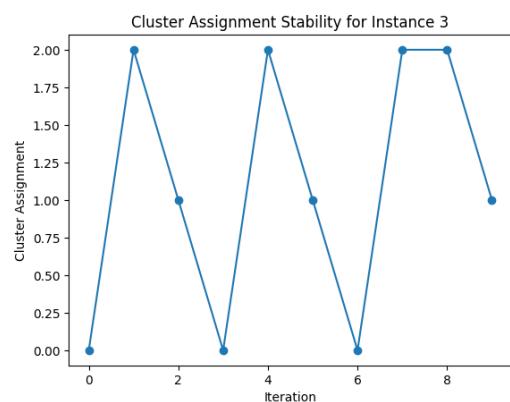


FIGURE 3.15 – Stabilité de l'instance 3

## stabilité du centroïde

La stabilité du centroïde dans le contexte de l'algorithme K-means fait référence à la cohérence ou à la constance des positions des centroïdes lors de l'exécution de l'algorithme sur différentes itérations ou sur des échantillons différents du même jeu de données

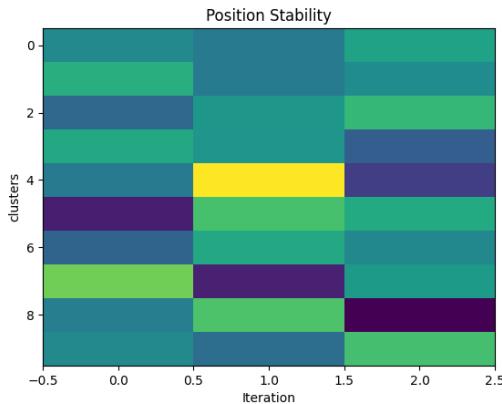


FIGURE 3.16 – Stabilité de centroïde 1

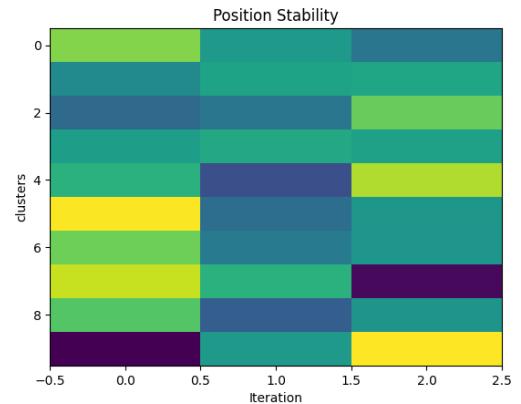


FIGURE 3.17 – Stabilité de centroïde 2

## Analyse

D'après les trois graphes ci-dessus, on voit bien que l'algorithme k-means n'est pas vraiment stable ; il présente des changements dans les résultats. En ce qui concerne la stabilité au niveau des clusters, on constate une stabilité intégrée avec de petites instabilités, mais en général, il a conservé la forme globale des clusters. Pour les instances, on a pris l'instance numéro 12, et on voit qu'elle a pris la valeur 2 cinq fois, la valeur 1 trois fois et la valeur 0 deux fois. Bien que la distribution soit différente, on peut prendre en considération le nombre d'affectations, qui est 5, plus grand que 3 et 2. La stabilité des centroïdes est vraiment différente ; il choisit des centroïdes différents pour chaque itération, mais tant que les couleurs sont proches, les positions des clusters ne sont pas vraiment éloignées.

## 3.2 Algorithme de clustering basé densité

### 3.2.1 Algorithme DBSCAN

Le DBSCAN est un algorithme simple qui définit des clusters en utilisant l'estimation de la densité locale. On peut le diviser en 4 étapes :

1. Pour chaque observation on regarde le nombre de points à au plus une distance ( $\varepsilon$ ) de celle-ci. On appelle cette zone le ( $\varepsilon$ )-voisinage de l'observation.
2. Si une observation compte au moins un certain nombre de voisins y compris elle-même, elle est considérée comme une observation cœur. On a alors décelé une observation à haute densité.
3. Toutes les observations au voisinage d'une observation cœur appartiennent au même cluster. Il peut y avoir des observations cœur proche les unes des autres. Par conséquent de proche en proche on obtient une longue séquence d'observations cœur qui constitue un unique cluster.
4. Toute observation qui n'est pas une observation cœur et qui ne comporte pas d'observation cœur dans son voisinage est considérée comme une anomalie.

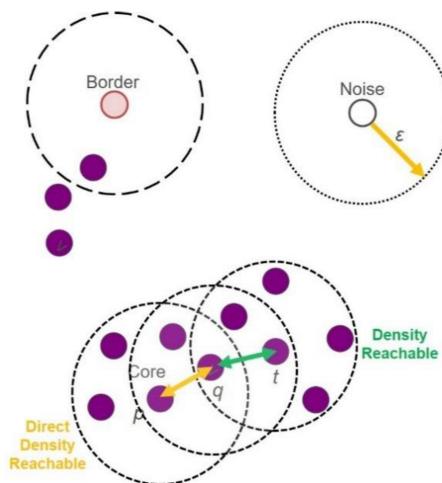


FIGURE 3.18 – Application

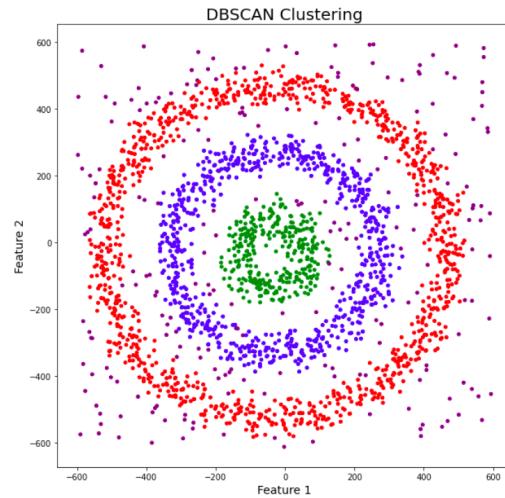


FIGURE 3.19 – Illustration

On doit donc définir deux informations avant d'utiliser le DBSCAN : la distance ( $\varepsilon$ ) et le nombre minimal de voisins nécessaire.

L'algorithme DBSCAN a une complexité en temps généralement considérée comme  $\mathcal{O}(n \log n)$  dans le meilleur des cas et  $\mathcal{O}(n^2)$  dans le pire des cas, où  $n$  est le nombre de points de données.

Voici son pseudo code :

---

**Algorithm 3** DBSCAN Algorithm

---

```
1: procedure DBSCAN(data, eps, min_samples)
2:   clusters  $\leftarrow$  []
3:   visited_points  $\leftarrow$  set()
4:   for each point  $P$  in data do
5:     P is visited
6:     mark P as visited
7:     neighbors  $\leftarrow$  REGION_QUERY( $P$ , eps, data)
8:     if  $\text{len}(\text{neighbors}) < \text{min\_samples}$  then
9:       mark P as noise else
10:      cluster  $\leftarrow$  EXPAND_CLUSTER( $P$ , neighbors, eps, min_samples, data)
11:      clusters.append(cluster)
12:   return clusters
13: end procedure

14: procedure REGION_QUERY( $P$ , eps, data)
15:   neighbors  $\leftarrow$  [] for each point  $Q$  in data do
16:     DISTANCE( $P$ ,  $Q$ )  $<$  eps
17:     add  $Q$  to neighbors
18:
19:   return neighbors
20: end procedure

21: procedure EXPAND_CLUSTER( $P$ , neighbors, eps, min_samples, data)
22:   cluster  $\leftarrow$  new cluster
23:   add  $P$  to cluster
24:   for each neighbor  $Q$  in neighbors do
25:     if  $Q$  is not visited
26:       mark  $Q$  as visited
27:       new_neighbors  $\leftarrow$  REGION_QUERY( $Q$ , eps, data)
28:       if  $\text{len}(\text{new\_neighbors}) \geq \text{min\_samples}$  then
29:         add new_neighbors to neighbors
30:
31:       if  $Q$  is not yet member of any cluster then
32:         add  $Q$  to cluster
33:   return cluster
end procedure
```

---

### 3.2.2 Le choix de $\epsilon$

Nous allons choisir un  $\epsilon$  de tel sorte que 90% des observations aient une distance au proche voisin inférieure à  $\epsilon$ . Dans notre exemple (la figure 3.16), c'est à partir de 0.55 et au maximum 0.66. semble convenir.

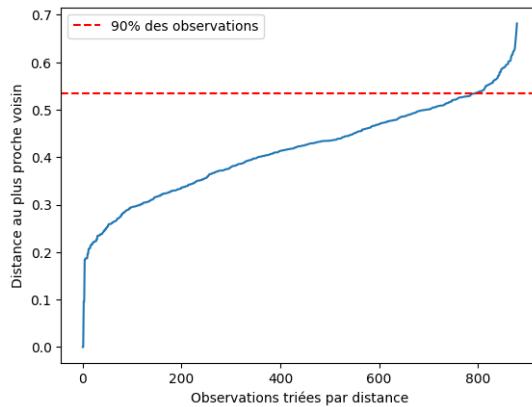


FIGURE 3.20 – Le choix de ( $\epsilon$ )

### 3.2.3 Le choix de nombre minimal de voisins nécessaire

eps	min_samples	silhouette_score	num_clusters
0.3	4	-0.21	6
0.5	5	0.11	2
0.5	7	0.09	2
0.5	8	0.09	2
0.5	4	0.12	2
0.5	6	0.11	2
0.54	2	0.15	2
0.54	6	0.14	2
0.55	6	0.16	2
0.58	6	0.20	2
0.6	6	0.20	2
0.6	4	0.27	2

TABLE 3.2 – Résultats de l'expérimentation avec différents paramètres de DBSCAN

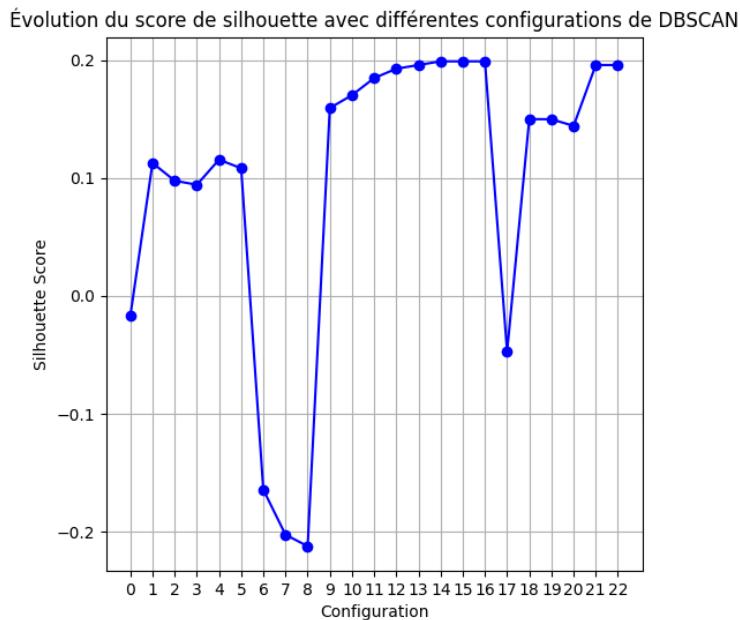


FIGURE 3.21 – silhouette score avec DBSCAN

## Analyse

Selon le graphique présenté dans la Figure 3.20, on observe une variation significative du score de silhouette en fonction des valeurs d'eps et de minsamples. Les scores varient entre des valeurs positives et négatives, indiquant la qualité de la séparation des clusters. La configuration optimale semble être celle avec  $\text{eps}=0.6$  et  $\text{minsamples}=4$ , offrant le meilleur score de silhouette (0.27) avec seulement 2 clusters. Cela suggère une séparation efficace des données par rapport aux autres configurations.

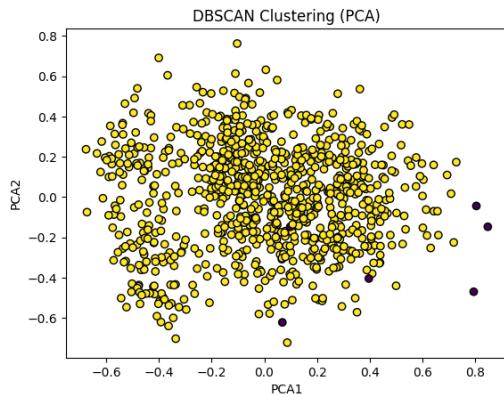


FIGURE 3.22 – Clusters avec DBSCAN

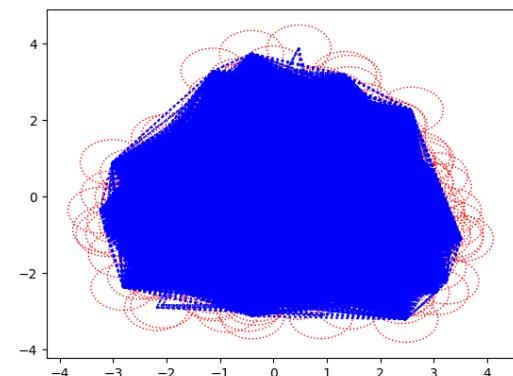


FIGURE 3.23 – Methode de DBSCAN

## Récapitulatif

eps	minsamples	clusters	Détection des valeurs aberrantes
0.6	4	1	Oui

On constate que DBSCAN a divisé notre ensemble de données en seulement deux clusters ; l'un des deux est détecté comme un groupe d'outliers, tandis que le reste du jeu de données est considéré comme un seul cluster.

### 3.3 Comparaison des deux algorithmes K-means et DBSCAN

Il est souvent recommandé de tester plusieurs algorithmes sur nos données et d'évaluer leur performance en fonction de mesures telles que la silhouette, inter-cluster et intra-cluster pour déterminer lequel fonctionne le mieux pour votre cas particulier.

	k-means	DBSCAN
<b>Nombre de clusters</b>	3	2
<b>Silhouette</b>	0.40	0.27
<b>Stabilité</b>	Non	Oui
<b>Inter-cluster</b>	6.98	0.020
<b>Intra-cluster</b>	2604.93	35642.0
<b>Detecter les valeurs anomalies</b>	Non	Oui

TABLE 3.3 – Comparaison entre K-means et DBSCAN

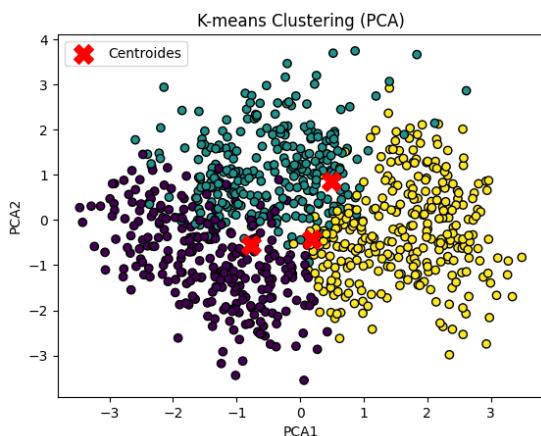


FIGURE 3.24 – Clusters avec DBSCAN

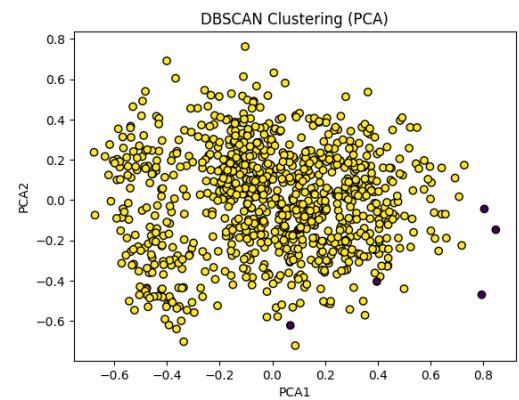


FIGURE 3.25 – Clusters avec k-means

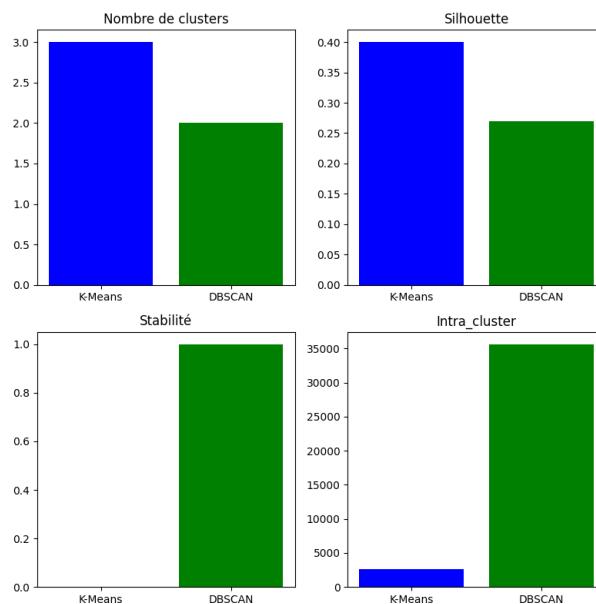


FIGURE 3.26 – Illustration de comparaison

## Analyse

- K-means a été configuré avec 3 clusters, tandis que DBSCAN a identifié un seul cluster.
- DBSCAN a pu détecter les outliers
- k-means obtient un score de 0.40, ce qui suggère une séparation raisonnable entre les clusters. DBSCAN obtient un score de 0.27, ce qui indique une séparation, mais peut-être moins nette.
- Il est indiqué que k-means n'est pas stable, tandis que DBSCAN est stable
- K-means a une inter-cluster de 6.98 et une intra-cluster de 2604.93. Une faible inter-cluster et une haute intra-cluster sont généralement souhaitables, indiquant des clusters bien définis. DBSCAN a une inter-cluster de 0.020 et une intra-cluster de 35642.0

## 3.4 Conclusion

DBSCAN et K-means sont deux algorithmes pour le clustering non supervisé, et chacun d'eux présente des avantages et des inconvénients. Le choix entre les deux dépend vraiment de nos besoins et de nos données, car chacun a ses conditions pour donner de bons résultats. Dans notre cas, K-means est préférable, car il parvient à détecter le nombre exact de clusters avec une répartition uniforme de leurs tailles, et même le score de silhouette est une valeur acceptable et plus élevée que celui de DBSCAN.

# Conclusion Générale

Dans le cadre de ce projet, nous nous sommes plongés dans les intrications de la classification et du clustering, deux approches majeures du data mining. La classification trouve son utilité dans la prédiction des étiquettes de classe associées aux objets de données, tandis que le clustering s'attache à regrouper les données en fonction de leurs similitudes intrinsèques, sans préjuger des étiquettes de groupes.

Pour la classification, nous avons mis en œuvre des algorithmes tels que le "KNN" (k plus proches voisins), les "Decision Trees" (arbres de décision), et le "Random Forest" à partir de zéro, c'est-à-dire sans utiliser de bibliothèques existantes. Concernant le clustering, nous avons opté pour l'utilisation de K-means et de DBSCAN.

Au cours de cette implémentation, nous avons minutieusement examiné chacune des étapes de ces algorithmes, comprenant leur mécanisme interne, le processus de production des résultats, ainsi que les nuances de leur fonctionnement. En parallèle, nous avons évalué ces algorithmes afin de sélectionner le plus performant, en menant diverses expérimentations visant à affiner notre compréhension des forces et des limites de chaque approche. Cette démarche d'évaluation a été essentielle pour déterminer quel algorithme répondait le mieux aux exigences spécifiques de notre projet, contribuant ainsi à orienter nos choix vers des solutions optimales.

# Références

- [1] Cours Data Mining - Prof DERIAS.H .
- [2] TPs Data Mining - DR Belkadi Widad Hassina ,DR Khelfa Celia.
- [3] <https://datascientest.com/machine-learning-clustering-dbscan>.
- [4] <https://mrmint.fr/algorithme-k-means>.

## 1.1 Annexe A : Détails du Jeu de Données

	Moyenne	Médiane	Mode	Max	Min	q0	q1	q2	q3	q4
N	247.0	257	207	383	6	6	201.0	257	307.0	307.0
P	14.56	7.5	8.3	125.0	2.9	2.9	12.4	7.5	10.6	10.7
K	501.34	475	444	1560	11	11	412.0	475	581.0	581.0
pH	7.51	7.5	7.5	11.15	0.9	0.9	7.35	7.5	7.63	7.63
EC	0.54	0.55	0.62	0.95	0.1	0.1	0.43	0.55	0.64	0.64
OC	0.62	0.68	0.88	24.0	0.1	0.1	0.39	0.68	1.07	1.07
S	7.55	6.64	5.13	31.0	0.64	0.64	4.7	6.64	8.75	8.75
Zn	0.47	0.36	0.28	42.0	0.07	0.07	0.28	0.36	0.47	0.47
Fe	4.13	3.56	6.32	44.0	0.21	0.21	2.035	3.56	6.31	6.32
Cu	0.95	0.93	1.25	3.02	0.09	0.09	0.63	0.93	1.25	1.25
Mn	8.65	8.34	7.54	31.0	0.11	0.11	6.21	8.34	11.45	11.48
B	0.59	0.41	0.34	2.82	0.06	0.06	0.27	0.41	0.61	0.61
OM	1.06	1.01	1.51	41.28	0.172	0.172	0.6536	1.0148	1.34	1.34
Fertility	0.59	1	1	2	0	0	0.0	1	1.0	1.0

TABLE 4 – Description statistique des attributs.

## 1.2 Annexe B : Visualisation du Jeu de Données

### 1.2.1 Les symétries

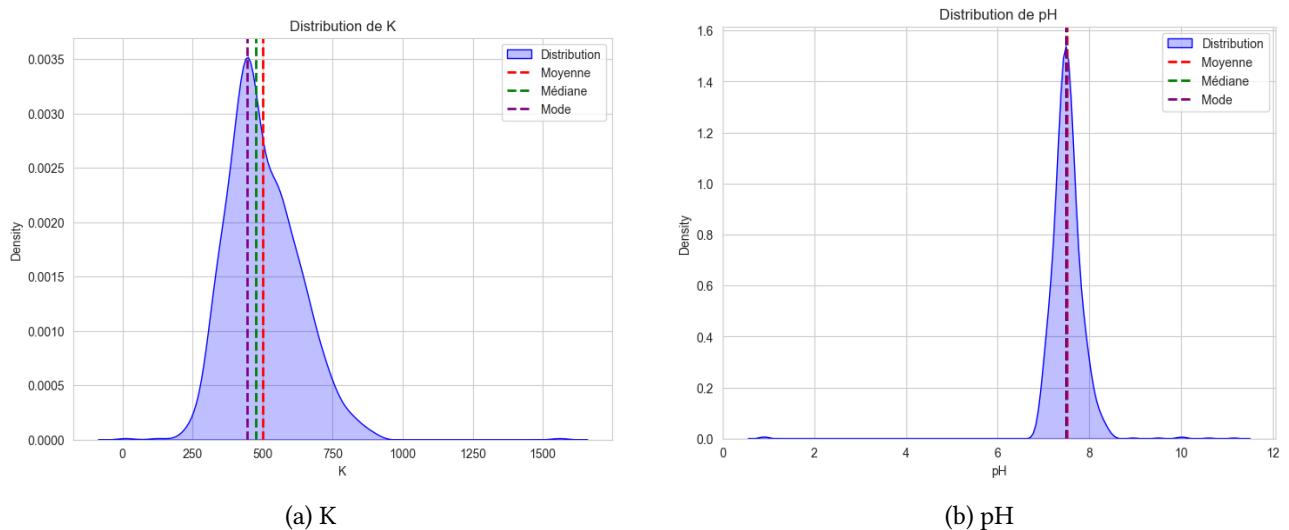
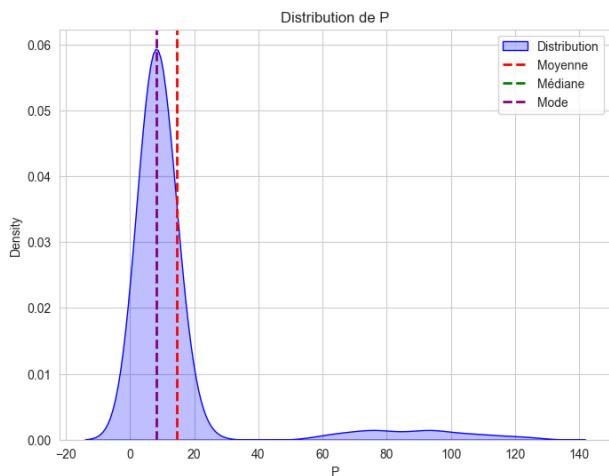
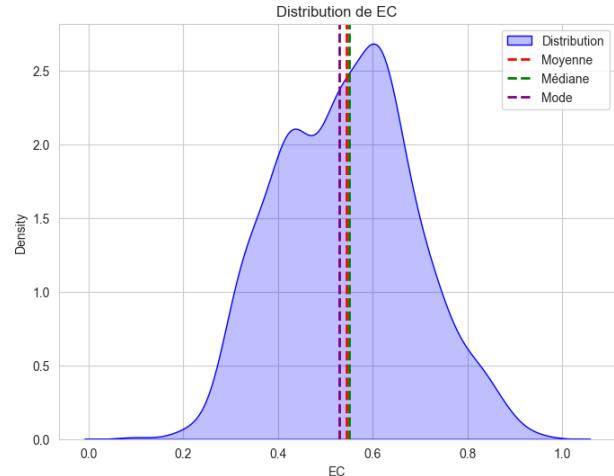


FIGURE 27 – Graphe Densité de K et pH

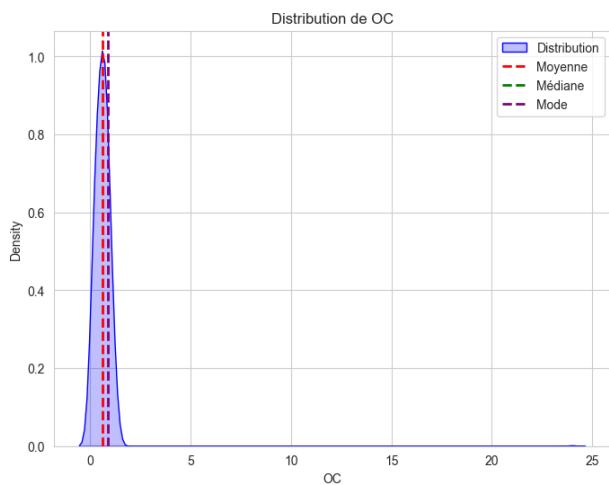


(a) P

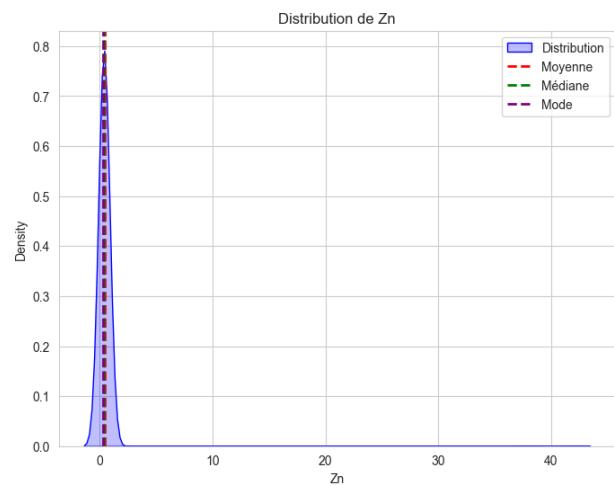


(b) EC

FIGURE 28 – Graphe Densité de P et EC

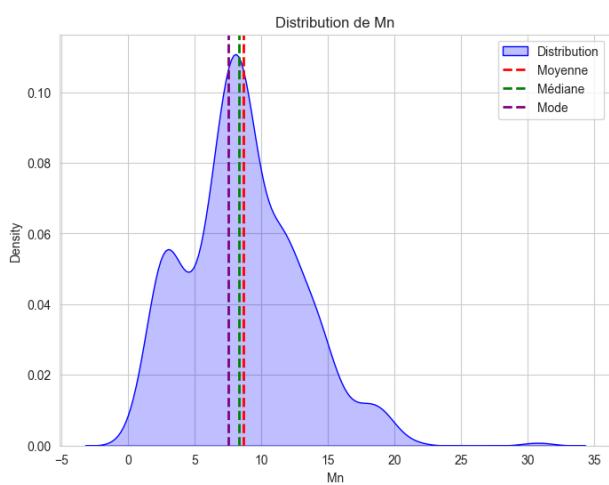


(a) OC

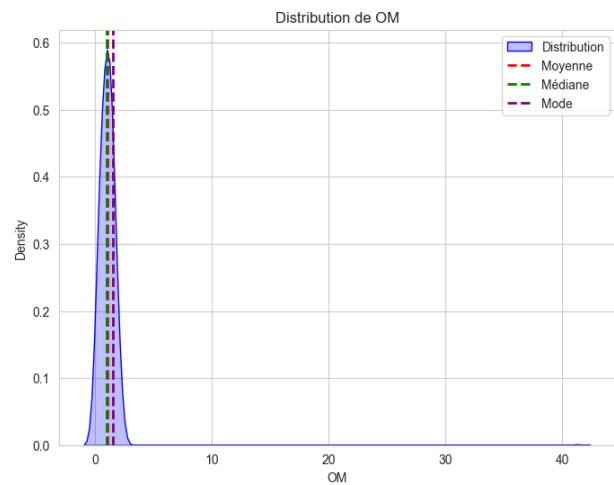


(b) Zn

FIGURE 29 – Graphe Densité de OC et Zn



(a) Mn



(b) OM

FIGURE 30 – Graphe Densité de Mn et OM

## 2.2 Visualisation des outliers

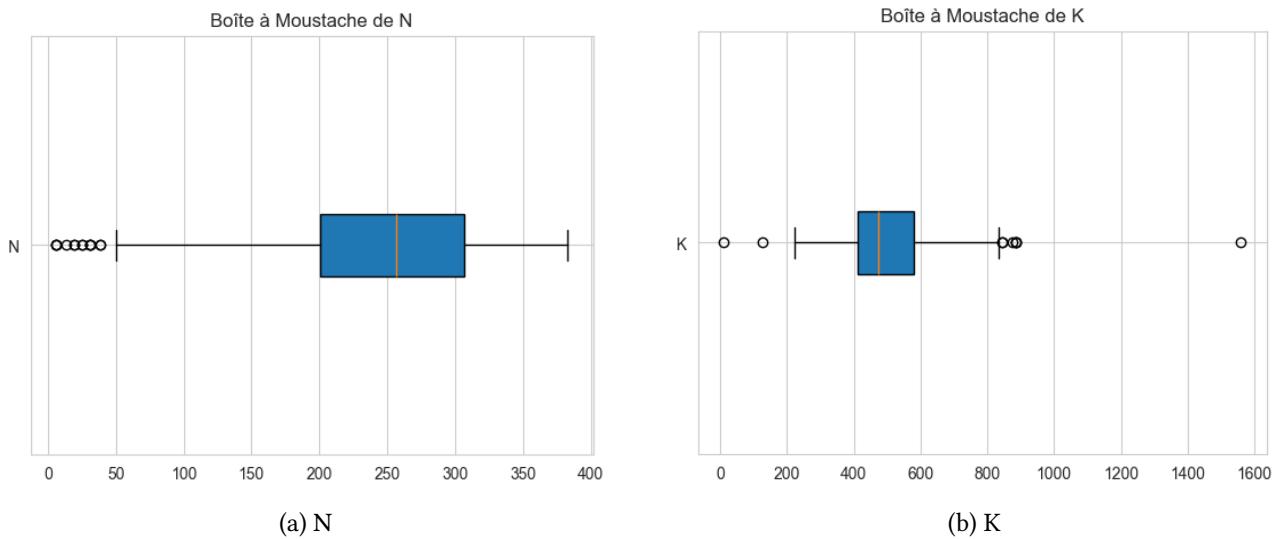


FIGURE 31 – Boîte à moustaches de N et K

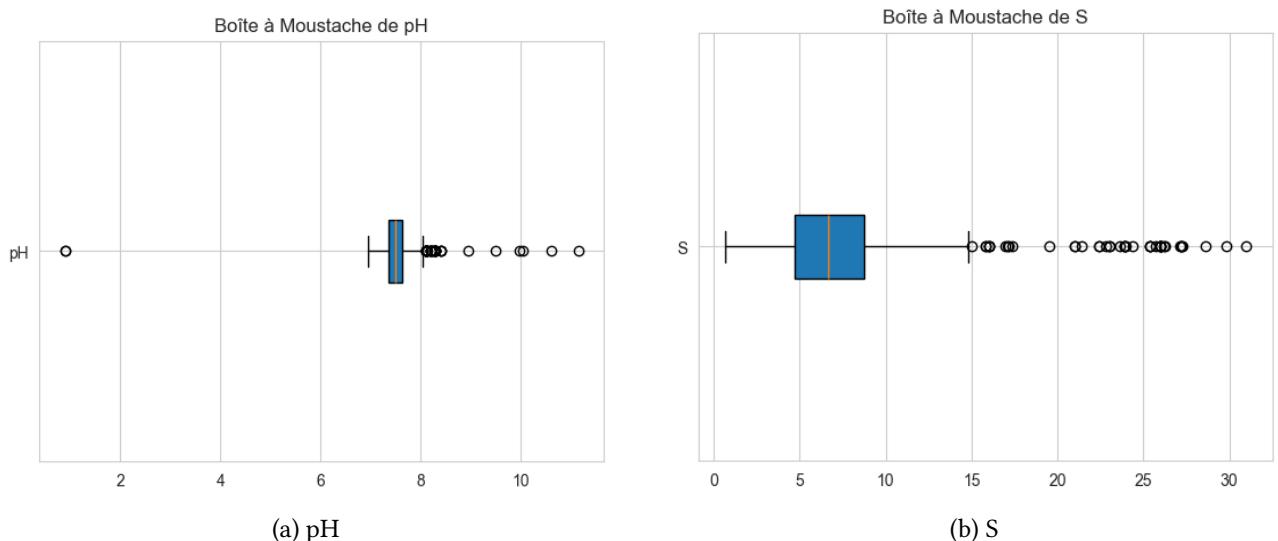


FIGURE 32 – Boîte à moustaches de pH et S

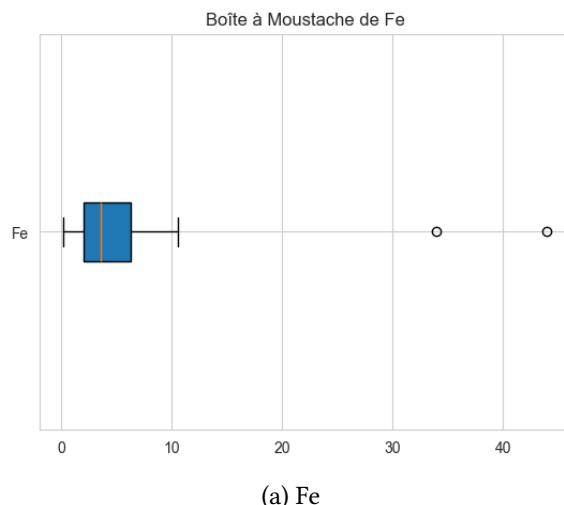


FIGURE 33 – Boîte à moustaches de Fe

### 2.3 Construction des histogrammes de

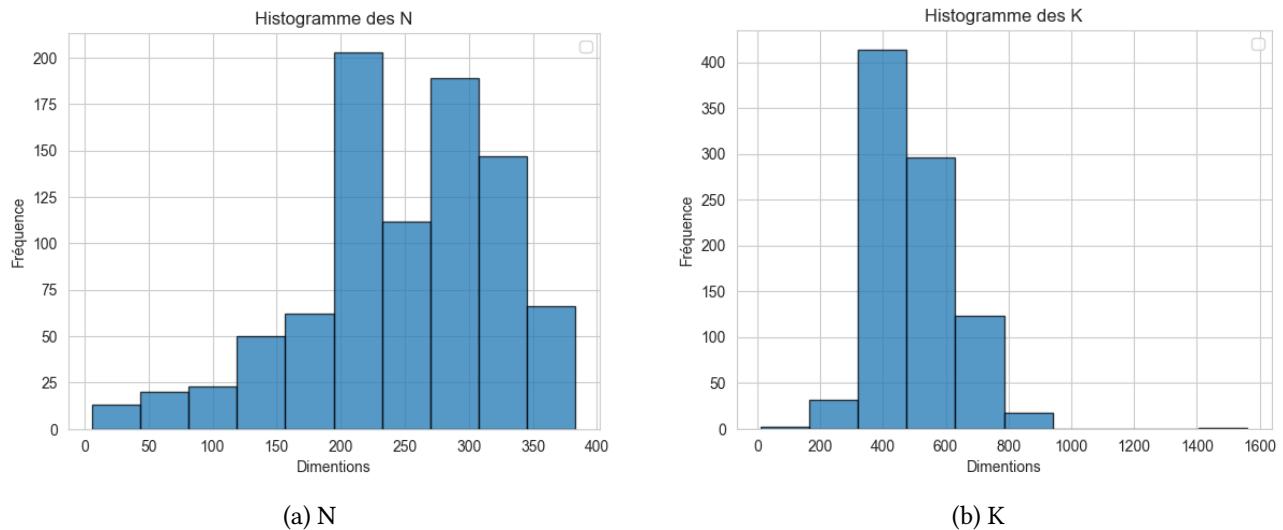


FIGURE 34 – Histogramme de N et K

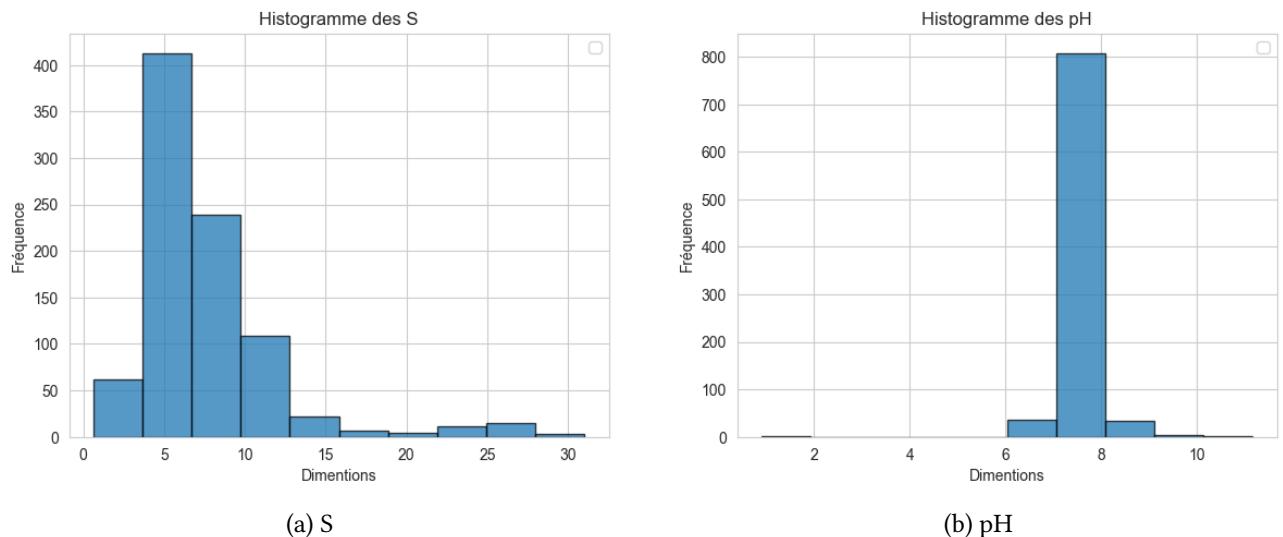


FIGURE 35 – Histogramme de S et pH

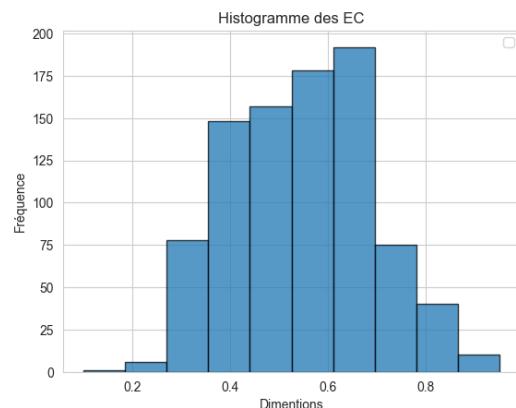


FIGURE 36 – Histogramme de EC