***** REPUBLIQUE DU BÉNIN *****
UNIVERSITE D'ABOMEY-CALAVI

ECOLE NATIONALE D'ÉCONOMIE APPLIQUÉE ET DE MANAGEMENT (ENEAM)

Page de garde

Filière : INFORMATIQUE DE GESTION (Analyse Informatique et Programmation)

Discipline:
INTRODUCTION A L'INTELLIGENCE ARTIFICIELLE

TP1:
SEGMENTATION ~ MACHINE LEARNING ~ PYTHON

Membres du Groupe 1

- BAWA SACCA Hamid
- COCOUVI Alexandro
- HOUNKPATIN Dèwanou Hugues-Marie
- OUSSA Chadrac Espoir
- PATINDE Nolan

Professeur
Dr DAGBA Théophile Komlan

ANNÉE ACADÉMIQUE

2025 – 2026

Sommaire

Table des matières

Introduction

Ce rapport présente une étude de segmentation du marché des céréales réalisée via le langage de programmation **Python**. L'objectif principal de ce travail est d'appliquer l'algorithme de clustering **K-Means** sur un dataset de 77 céréales afin d'identifier des groupes homogènes basés sur leurs caractéristiques nutritionnelles.

Le processus analytique s'appuie sur la bibliothèque scientifique **Scikit-Learn**, référence en Machine Learning, pour automatiser le traitement des données et l'extraction de connaissances. L'enjeu est de déterminer si une segmentation algorithmique peut isoler de manière pertinente les facteurs (calories, sucres, fibres, etc.) qui influencent la qualité nutritionnelle globale des produits.

L'étude s'articule autour de trois axes majeurs :

1. Le **prétraitement rigoureux** des données (imputation et normalisation).
2. L'**expérimentation comparative** entre deux modèles (k=3 et k=5).
3. L'**évaluation quantitative** de la qualité des clusters via les mesures de SSE (Inertie) et du score de Silhouette.

I-Compréhension des données

Le jeu de données utilisé contient 77 instances dans le fichier (cereals.csv), chacune représentant une céréale, décrite par 16 attributs, dont 13 attributs numériques réellement exploités par l'algorithme.

I-1-Attributs utilisés dans l'analyse

Les attributs pris en compte par WEKA sont :

- Calories : apport énergétique
- Protéines (protein) : teneur en protéines
- Lipides (fat) : teneur en matières grasses
- Sodium : teneur en sel
- Fibres (fiber) : apport en fibres alimentaires
- Glucides (carbo) : quantité totale de glucides
- Sucres (sugars) : part des glucides simples
- Potassium (potass) : teneur en minéraux
- Vitamines (vitamins) : enrichissement en vitamines
- Durée de conservation (shelf) : position sur l'étagère (indicateur commercial)
- Poids (weight) : masse d'une portion
- Contenance (cups) : volume d'une portion
- Note (rating) : appréciation globale des céréales

I-2-Attributs ignorés

Les attributs suivants ont été exclus car non numériques et donc non pertinents pour le clustering :

- name
- mfr
- type

Ainsi, la segmentation repose uniquement sur des critères mesurables et comparables, ce qui garantit la cohérence des résultats.

II-Prétraitement des données (Python)

Afin de garantir une analyse rigoureuse et comparable à celle effectuée sous WEKA, deux étapes cruciales de prétraitement ont été automatisées via la bibliothèque scikit-learn sur l'ensemble des 77 instances.

II-1 Gestion des valeurs manquantes : SimpleImputer

Tout comme le filtre *ReplaceMissingValues* de WEKA, nous avons utilisé la classe SimpleImputer pour traiter les données incomplètes.

- **Mécanisme** : Le script identifie les champs vides et les remplace systématiquement par la **moyenne** de l'attribut concerné (strategy="mean").
- **Objectifs garantis** :
    - **Conservation de l'intégrité** : Aucune des 77 céréales n'est supprimée du dataset.
    - **Fiabilité statistique** : L'imputation par la moyenne permet de maintenir la distribution globale des données sans introduire de biais majeur lors de la formation des clusters.

II-2 Normalisation des données : MinMaxScaler

Après l'imputation, l'étape de mise à l'échelle a été réalisée à l'aide de MinMaxScaler. Contrairement au *StandardScaler* (qui centre sur zéro), le MinMaxScaler reproduit exactement le comportement du filtre *Normalize* de WEKA.

- **Transformation** : Toutes les valeurs numériques sont compressées dans l'intervalle **[0 ; 1]**.
- **Avantages pour l'algorithme** :
    - **Homogénéité** : Elle rend comparables des attributs aux unités disparates (ex : le Sodium qui atteint 320 mg face aux Protéines qui ne dépassent pas 6 g).
    - **Équité des poids** : Elle empêche les attributs à forte échelle de dominer le calcul de la distance euclidienne.
    - **Stabilité** : Elle assure une convergence plus rapide de l'algorithme K-Means.

II-2. Le Code Source (voir clustering.py)

clustering.py > ...

```python
# IMPORT DES LIBRAIRIES
import pandas as pd
import numpy as np

from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# 1 CHARGEMENT DES DONNÉES
df = pd.read_csv("cereals.csv")

# Colonnes ignorées comme dans WEKA
df_numeric = df.drop(columns=["name", "mfr", "type"])

# 2 PRÉTRAITEMENT (IDENTIQUE WEKA)

# ReplaceMissingValues → moyenne
imputer = SimpleImputer(strategy="mean")
X_imputed = imputer.fit_transform(df_numeric)

# Normalize → [0,1]
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X_imputed)

X_scaled = pd.DataFrame(X_scaled, columns=df_numeric.columns)

# 3 FONCTION K-MEANS
def executer_kmeans(k):
    print(f"\n{'='*25} K-MEANS AVEC k = {k} {'='*25}")

    kmeans = KMeans(
        n_clusters=k,
        init="random",        # initialisation aléatoire comme WEKA
        max_iter=500,         # -I 500
        n_init=10,            # redémarrages multiples
        random_state=10       # -S 10
    )

    labels = kmeans.fit_predict(X_scaled)

    df_res = X_scaled.copy()
    df_res["Cluster"] = labels
```

clustering.py > ...

```python
def executer_kmeans(k):
    df_res = X_scaled.copy()
    df_res["Cluster"] = labels

    # Moyennes par cluster (centroïdes finaux)
    moyennes = df_res.groupby("Cluster").mean().T
    print("\nMoyennes des attributs par cluster :")
    print(moyennes)

    # Répartition des instances
    print("\nRépartition des instances :")
    counts = df_res["Cluster"].value_counts().sort_index()
    for c, n in counts.items():
        print(f"Cluster {c} : {n} instances ({n/len(df_res)*100} %)")

    # SSE (équivalent WEKA)
    print("\nWithin-cluster sum of squared errors (SSE) :")
    print(kmeans.inertia_)

    # Silhouette (qualité globale)
    silhouette = silhouette_score(X_scaled, labels)
    print(f"\nSilhouette score (k={k}) : {silhouette}")

    return silhouette, kmeans.inertia_

# 4 EXÉCUTION
sil_k5, sse_k5 = executer_kmeans(5)
sil_k3, sse_k3 = executer_kmeans(3)

# 5 COMPARAISON FINALE
print("\n========== COMPARAISON FINALE ==========")
print(f"k=5 → Silhouette : {sil_k5} | SSE : {sse_k5}")
print(f"k=3 → Silhouette : {sil_k3} | SSE : {sse_k3}")

if sil_k5 > sil_k3:
    print("La solution k=5 est la plus satisfaisante.")
else:
    print("La solution k=3 est la plus satisfaisante.")
```

III-Implémentation du K-Means en Python

III-1-Algorithme k-mean avec k=5

Pour assurer la cohérence avec l'analyse WEKA, nous utilisons les bibliothèques pandas pour la donnée et scikit-learn pour l'intelligence artificielle.

III-1-1. Paramétrage du script (Identique WEKA)

Paramètre	Valeur Python	Correspondance WEKA
Algorithme	KMeans	SimpleKMeans
Nombre de clusters	n_clusters=5	k = 5
Initialisation	init="random"	Random Initialisation
Distance	Euclidienne	Default DistanceFunction
Graine Aléatoire	random_state=10	-S 10
Itérations Max	max_iter=500	-I 500

IV-3. Analyse détaillée des résultats (k=5)

```
========================= K-MEANS AVEC k = 5 =========================

Moyennes des attributs par cluster :
Cluster          0         1         2         3         4
calories  0.503497  0.318182  0.121212  0.595041  0.560606
protein   0.353846  0.360000  0.600000  0.375758  0.077778
fat       0.138462  0.060000  0.133333  0.278788  0.200000
sodium    0.764423  0.029687  0.552083  0.524621  0.512153
fiber     0.101648  0.147857  0.785714  0.186147  0.029762
carbo     0.790064  0.637500  0.319444  0.679293  0.556713
sugars    0.317308  0.150000  0.291667  0.520833  0.802083
potass    0.215663  0.248943  0.939577  0.370503  0.124706
vitamins  0.250000  0.075000  0.250000  0.378788  0.250000
shelf     0.038462  0.400000  1.000000  0.969697  0.388889
weight    0.500000  0.383000  0.500000  0.604545  0.500000
cups      0.582769  0.504800  0.109333  0.391515  0.516889
rating    0.341919  0.590476  0.737511  0.309109  0.127523

Répartition des instances :
Cluster 0 : 13 instances (16.883116883116884 %)
Cluster 1 : 10 instances (12.987012987012985 %)
Cluster 2 : 3 instances (3.896103896103896 %)
Cluster 3 : 33 instances (42.857142857142854 %)
Cluster 4 : 18 instances (23.376623376623375 %)

Within-cluster sum of squared errors (SSE) :
24.099423163102873

Silhouette score (k=5) : 0.2774123994252421
```

Répartition des clusters :

D'après les résultats de l'exécution Python :

- **Cluster 0 : 13 instances (16.88 %)**
- **Cluster 1 : 10 instances (12.99 %)**
- **Cluster 2 : 3 instances (3.90 %)**
- **Cluster 3 : 33 instances (42.86 %)**
- **Cluster 4 : 18 instances (23.38 %)**

Analyse détaillée des clusters (k = 5)

Cluster 0 (16.88 %) :

- **Profil nutritionnel : calories = 0.503497, protein = 0.353846 , carbo = 0.790064, sugars = 0.317308, shelf = 0.038462, rating = 0.341919**
- **Interprétation : Ce groupe se concentre sur l'apport en énergie.**
    1. **Force en Glucides : Il possède le taux de Carbo le plus élevé (0.7900).**
    2. **Marketing : Placé quasi exclusivement sur l'étagère du bas (shelf = 0.0384).**
- **Conclusion : Ce sont les céréales "Énergétiques".**

Cluster 1 (12.99 %) :

Profil nutritionnel : calories = 0.318182, sodium = 0.029687, sugars = 0.150000, shelf = 0.400000, rating = 0.590476

- **Interprétation : Avec un Rating de 0.5904, c'est le deuxième groupe le plus sain.**
    1. **Pureté : Il détient le record du Sodium le plus bas (0.0296) et du Sucre le plus bas (0.1500) de toute l'analyse.**
    2. **Positionnement : Placé sur l'étagère intermédiaire (shelf = 0.40).**
- **Conclusion : Ce segment regroupe les céréales "Sans additifs". Idéal pour les consommateurs surveillant leur tension et leur glycémie.**

Cluster 2 (3.90 %) :

Profil nutritionnel : calories = 0.121212, protein = 0.600000 , fat = 0.133333 , sodium = 0.552083, fiber = 0.785714 , carbo = 0.319444, sugars = 0.291667, potass = 0.939577, vitamins = 0.250000, rating = 0.737511

- **Interprétation : C'est le groupe le plus performant. Il possède le Rating le plus élevé (0.7375).**

1. **Richesse naturelle : Il affiche les taux les plus élevés en Fibres (0.7857) et en Potassium (0.9395).**
2. **Légèreté : C'est le cluster le moins calorique (0.1212).**

- **Conclusion : Ce segment regroupe les céréales dites "Saines".**

<u>Cluster 3 (42.86 %) :</u>

- **Profil nutritionnel : calories = 0.595041, protein = 0.375758 , vitamins = 0.378788, shelf = 0.969697, rating = 0.309109**
- **Interprétation : C'est le segment le plus vaste du marché (33 instances).**
    1. **Enrichissement : Taux de Vitamines élevé (0.3787) pour compenser un profil moyen.**
    2. **Visibilité : Placé sur les étagères du haut (shelf = 0.9696).**
- **Conclusion : Ce sont les "Céréales Moyennes"**

<u>Cluster 4 (23.38 %) :</u>

Profil nutritionnel : calories = 0.560606, protein = 0.077778, fiber = 0.029762, sugars = 0.802083, rating = 0.127523

- **Interprétation : Le groupe le moins bien noté par l'algorithme.**
    1. **Excès de Sucre : Record de Sugars (0.8020).**
    2. **Carences : Taux de Protéines (0.0777) et de Fibres (0.0297) les plus bas.**
- **Conclusion : Ce segment regroupe les céréales "Moins saines". Leur faible qualité nutritionnelle (Rating 0.1275) est une conséquence directe de l'omniprésence du sucre.**

IV-2-Algorithme k-mean avec k=3

Résultats pour k = 3

```
======================== K-MEANS AVEC k = 3 ========================

Moyennes des attributs par cluster :
Cluster          0         1         2
calories  0.375758  0.558442  0.542088
protein   0.466667  0.388571  0.118519
fat       0.120000  0.262857  0.170370
sodium    0.248958  0.528571  0.599537
fiber     0.150952  0.238776  0.044974
carbo     0.658333  0.652381  0.641975
sugars    0.187500  0.503571  0.655093
potass    0.263646  0.421580  0.143449
vitamins  0.133333  0.371429  0.250000
shelf     0.300000  0.985714  0.277778
weight    0.422000  0.598571  0.500000
cups      0.514667  0.368914  0.538667
rating    0.534972  0.344358  0.184488

Répartition des instances :
Cluster 0 : 15 instances (19.480519480519483 %)
Cluster 1 : 35 instances (45.45454545454545 %)
Cluster 2 : 27 instances (35.064935064935064 %)

Within-cluster sum of squared errors (SSE) :
32.86544487443294

Silhouette score (k=3) : 0.24181364039112196
```

```
========== COMPARAISON FINALE ==========
k=5 → Silhouette : 0.2774123994252421 | SSE : 24.099423163102873
k=3 → Silhouette : 0.24181364039112196 | SSE : 32.86544487443294
La solution k=5 est la plus satisfaisante.
```

Répartition des clusters :

D'après les résultats de l'exécution Python pour $k=3$:

- **Cluster 0** : 15 instances (19,48 %)
- **Cluster 1** : 35 instances (45,45 %)
- **Cluster 2** : 27 instances (35,06 %)

Analyse détaillée des clusters (k = 3)

Cluster 0 (19,48 %) :

- **Profil nutritionnel** : calories = 0,375758, protein = 0,466667, fat = 0,120000, sodium = 0,248958, sugars = 0,187500, rating = 0,534972
- **Interprétation** : C'est le groupe le plus sain de cette configuration.
    1. **Qualité nutritionnelle** : Il possède le **Rating le plus élevé (0,5349)** de la solution à 3 clusters.
    2. **Atouts** : Il affiche le taux de sucre le plus bas (0,1875) et le taux de sodium le plus bas (0,2489).
    3. **Construction** : Un apport en protéines solide (0,4666), le plus élevé des trois groupes.
- **Conclusion** : Ce cluster regroupe les céréales dites **"Bien-être"**.

Cluster 1 (45,45 %) :

- **Profil nutritionnel** : calories = 0,558442, fiber = 0,238776, sugars = 0,503571, vitamins = 0,371429, shelf = 0,985714, rating = 0,344358
- **Interprétation** : C'est le segment le plus volumineux, représentant presque la moitié du dataset.
    1. **Profil Énergétique** : Des calories plus élevées (0,5584) compensées par un bon taux de fibres (0,2387) et de potassium (0,4215).
    2. **Marketing** : Positionné presque exclusivement sur l'étagère du haut (**shelf = 0,9857**).
- **Conclusion** : Ce segment regroupe les **"Energétiques"**. Ce sont des produits enrichis (vitamines = 0,3714).

Cluster 2 (35,06 %) :

- **Profil nutritionnel** : calories = 0,542088, protein = 0,118519, fiber = 0,044974, sugars = 0,655093, rating = 0,184488
- **Interprétation** : Ce groupe rassemble les céréales les moins bien notées.
    1. **Déséquilibre** : Il possède le **taux de sucre le plus élevé (0,6550)** et le taux de protéines le plus bas (0,1185).
    2. **Carences** : Les fibres sont quasi-absentes (0,0449).
- **Conclusion** : Ce segment regroupe les céréales **"Moins saines"**. Leur faible **Rating (0,1844)** reflète une transformation industrielle importante privilégiant le sucre au détriment des nutriments essentiels.

V- Comparaison finale et choix de la solution la plus satisfaisante (Python)

Comparaison quantitative

Pour évaluer la performance des modèles en Python, nous utilisons le **SSE** (équivalent au WCSS) et le **Silhouette Score**.

- **k = 5**
    - **SSE** = 24.099423163102873
    - **Silhouette Score** = 0.2774123994252421
    - **Observation** : Les clusters sont plus compacts et mieux définis mathématiquement.
- **k = 3**
    - **SSE** = 32.86544487443294
    - **Silhouette Score** = 0.24181364039112196
    - **Observation** : L'erreur globale est plus élevée (augmentation de ~36%), ce qui traduit une plus grande dispersion des données au sein des groupes.

Comparaison qualitative

- **k = 5 : Précision et Cohérence**
    - **Séparation fine** : Le modèle distingue clairement les céréales "Ultra-Saines" (Cluster 2).
    - **Profils cohérents** : Chaque groupe répond à un profil marketing réel :
        - Céréales diététiques (fibres/potassium élevés).
        - Céréales naturelles (très pauvres en sodium/sucre).
        - Céréales énergétiques (riches en glucides complexes).
        - Céréales standards enrichies (vitamines élevées).
        - Céréales "Plaisir" (taux de sucre maximal).
- **k = 3 : Généralisation excessive**
    - **Groupes trop larges** : Le Cluster 1 regroupe à lui seul 45% des instances.
    - **Perte d'information** : Les nuances entre les produits énergétiques et les produits simplement sucrés disparaissent.
    - **Analyse moins précise** : Moins efficace pour une recommandation diététique ciblée.

Solution la plus satisfaisante (Python) :

Après étude des résultats fournis par l'algorithme K-Means (implémentation scikit-learn) sur le dataset des céréales, nous préconisons la solution à **k = 5** comme étant la plus pertinente pour les raisons suivantes :

1. Une meilleure finesse de segmentation (Précision)

Alors que la solution à k = 3 se contente de séparer le marché en trois blocs massifs (Santé, Moyen c'est à dire Energétiques, Moins sains), la solution à k = 5 permet d'isoler des comportements nutritionnels beaucoup plus subtils :

- **Distinction des profils négatifs** : Elle sépare les céréales "Moins saines" (Cluster 4, Sugars = 0.80) des céréales plus "Moyennes" (Cluster 3, Sugars = 0.52). Dans la version k=3, ces nuances sont partiellement fusionnées, masquant la sévérité du taux de sucre de certains produits.
- **Isolement du "pôle d'excellence"** : Elle isole un groupe très spécifique (Cluster 2, environ 4% du dataset) qui présente des taux de fibres (0.78) et de potassium (0.93) exceptionnels. Ces produits "de santé" sont totalement dilués dans le bloc généraliste de la solution à k=3.

2. Une corrélation plus précise avec la variable Rating

L'objectif de l'analyse était de comprendre ce qui détermine la qualité nutritionnelle (score de Rating).

- **Hiérarchie claire** : Avec k = 5, nous observons une hiérarchie de Ratings extrêmement nette et progressive : **0.73, 0.59, 0.34, 0.30, et 0.12**.
- **Identification des causes** : Cette décomposition prouve que la note de qualité ne chute pas uniquement à cause du sucre, mais aussi à cause de la carence en protéines (0.07 dans le Cluster 4) ou de l'excès de calories, des nuances que seule la segmentation à 5 clusters met en évidence avec cette précision.

3. Cohérente avec la réalité du marché

La solution à k = 5 sous Python correspond plus fidèlement à la segmentation réelle en grande distribution :

- **Diététique** (Cluster 2) : Riche en fibres et potassium.
- **Santé Naturelle** (Cluster 1) : Très pauvre en sodium et sucre.
- **Énergétique** (Cluster 0) : Apport maximal en glucides complexes.
- **Moyen** (Cluster 3) : Céréales enrichies en vitamines, étagère du haut.
- **Moins Sains** (Cluster 4) : Très sucrés, pauvres en nutriments essentiels.

Conclusion

Bien que la solution à k = 3 offre des segments plus larges et donc plus stables statistiquement, elle demeure trop généraliste pour une analyse nutritionnelle fine. La solution à **k = 5** apporte une valeur explicative supérieure. Elle transforme les données numériques brutes en une véritable connaissance

métier, permettant d'identifier précisément quels nutriments influencent la perception de qualité des céréales.

Bibliographie

Téléchargement du dataset (cereals.csv) https://www.kaggle.com/crawford/datasets?query=cereal

Comprendre le k-mean avec Python: https://www.kaggle.com/code/prashant111/k-means-clustering-with-python

Conclusion

Au terme de cette analyse réalisée sous Python, l'application de l'algorithme K-Means a permis de transformer une base de données brute en une cartographie nutritionnelle précise.

La comparaison des modèles démontre que la solution à **k=5** est la plus satisfaisante. Sur le plan quantitatif, elle affiche une meilleure cohésion avec un **SSE de 24.10** et une séparation des groupes plus nette confirmée par un **score de Silhouette de 0.277**. Sur le plan qualitatif, cette configuration permet d'isoler des niches spécifiques, comme le "pôle d'excellence" (Cluster 2) dont le **Rating de 0.73** témoigne d'une densité exceptionnelle en fibres et potassium.

En conclusion, ce projet illustre la puissance des outils de Machine Learning en Python pour la segmentation de données. L'algorithme a su identifier de manière autonome les déséquilibres nutritionnels (notamment l'impact négatif du sucre dans le Cluster 4) et valider mathématiquement une classification qui fait sens d'un point de vue diététique. Ce modèle constitue une base solide pour le développement d'outils d'aide à la décision ou de systèmes de recommandation nutritionnelle automatisés.