

Mohand Ou-Kaci Sari  
Giovanni Mangano  
Zaher Boudhaouia  
Jean-Charles Levy

# Transparence

**Présentation Finale :**

**ETUDE SUR LA TRANSPARENCE DES ALGORITHMES SUR LE  
CALCUL DES INFORMATIONS NUTRITIONNELLES DE  
PRODUITS ALIMENTAIRES**

**Partie 1 :**

**Élaboration des modèles de**

**Nutri-score basés sur**

**l'approche Electre Tri**

# Aliments

aliments	energy100g	saturatedfat100g	sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fruit
0	467	5,6	32	0,49	6,3	4	0
1	0	0	0	0,003	0	0	0
2	495	12	28	1,2	6	3,5	0
3	43	0	8,9	0	0,8	0,6	100
4	57	0,1	3,9	0,09	10	0	0
5	159	0,7	2,3	5,65	7,1	1,9	0
6	500	13	27,1	0,483	6,6	3,3	0
7	20	0	4,5	0,01	0	0	0
8	66	0,4	7,5	0,2	3,7	0,9	0
9	23	0	0,8	0,01	0,2	0	45
10	435	7,5	23	1,3	8	3	0
11	648	44	1,5	1,8	1,2	0,5	10
12	278	0,5	6,5	1,18	8,1	5	0
13	399	3,7	49	0,39	14,6	3,3	26
14	539	10,6	56	0,11	6,3	0	0
15	0	0	0	0,021	0	0	0
16	566	24	29	0,1	9,5	0	0
17	386	1,6	75,1	0,41	5,1	7,7	0
18	362	1,3	1,7	0,02	11	11	0
19	548	20,2	46,4	0,01	6	7,3	0
20	408	2,4	3,2	1,61	14	5,2	0
21	439	3,6	27	0,62	7,9	6,8	0
22	45	0,6	5,1	0,14	3,8	0	0
23	534	2,8	1,2	1,1	5,9	3,5	0
24	55	1,1	2,1	0,32	3,9	0,8	4,3

# Aliments avec Nova et Nutri-score

aliments	energy100g	saturatedfat1sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fr Nova	nutriscore	nom			
0	467	5,6	32	0,49	6,3	4	0	4 D	prince de lu		
1	0	0	0	0,003	0	0	0	1 A	volvic 1.5L		
2	495	12	28	1,2	6	3,5	0	4 E	granola		
3	43	0	8,9	0	0,8	0,6	100	1 C	Tropican Orange		
4	57	0,1	3,9	0,09	10	0	0	1 A	Skyr Danone		
5	159	0,7	2,3	5,65	7,1	1,9	0	2 C	amora Moutarde		
6	500	13	27,1	0,483	6,6	3,3	0	4 E	Kinder cereAlé		
7	20	0	4,5	0,01	0	0	0	1 D	Ice tea Peche		
8	66	0,4	7,5	0,2	3,7	0,9	0	1 A	AlPro Vanille		
9	23	0	0,8	0,01	0,2	0	45	1 C	Pulco citron		
10	435	7,5	23	1,3	8	3	0	4 E	Petit beurre de Lu		
11	648	44	1,5	1,8	1,2	0,5	10	4 E	Amora Mayonnaise		
12	278	0,5	6,5	1,18	8,1	5	0	4 A	Harris Pain de mie extra moelleux nature		
13	399	3,7	49	0,39	14,6	3,3	26	4 D	Lu Figolu		
14	539	10,6	56	0,11	6,3	0	0	4 E	Nutella		
15	0	0	0	0,021	0	0	0	1 B	Coca Zero		
16	566	24	29	0,1	9,5	0	0	4 E	Lindt Excellence 70% cacao noir		
17	386	1,6	75,1	0,41	5,1	7,7	0	4 E	Nesquick cacao nestlé		
18	362	1,3	1,7	0,02	11	11	0	4 A	Flocons d'avoine Bjorg		
19	548	20,2	46,4	0,01	6	7,3	0	4 E	Nestlé chocolat noir dessert		
20	408	2,4	3,2	1,61	14	5,2	0	4 C	Heudebert biscuits 6 céréales		
21	439	3,6	27	0,62	7,9	6,8	0	4 D	belvita petit dejeuner original chocolat		
22	45	0,6	5,1	0,14	3,8	0	0	3 A	Danone yaourt nature		
23	534	2,8	1,2	1,1	5,9	3,5	0	4 C	Pringles original		
24	55	1,1	2,1	0,32	3,9	0,8	4,3	1 B	Alpro Nature noix de coco		

# Critères

	energy100g	saturatedfat100g	sugars100g	sodium	proteins100g	fiber100g	teneur_en_fruit
poids	2	2	2	2	1	1	1
type_critere	min	min	min	min	max	max	max

# Critères 2

	energy100g	saturatedfat100g	sugars100g	sodium	proteins100g	fiber100g	teneur_en_fruit
poids	1	1	1	1	2	2	2
type_critere	min	min	min	min	max	max	max

$\pi^6$  : Borne supérieure des profils limites

$\pi^5$  : Profil limite entre A et B

$\pi^4$  : Profil limite entre B et C

$\pi^3$  : Profil limite entre C et D

$\pi^2$  : Profil limite entre D et E

$\pi^1$  : Borne inférieure des profils

TABLE 2 – Profils à déterminer

# Profils : Méthode 1

profil	energy100g	saturatedfat100g	sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fruit
b6	2500	5	50	2	0	0	0
b5	1700	3	30	1,5	7	2	0
b4	1300	2	20	1	14	4	0
b3	900	0,5	15	0,66	21	6	0
b2	500	1	10	0,33	26	8	0
b1	100	0,1	1	0	30	10	0

# Profils quantiles : Méthode 2

profil	energy100g	saturatedfat100g	sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fruit
b6	1466	3,08	14,64	0,68	8	3	0
b5	998,5	1,75	8,45	0,48	6,65	2	0
b4	498	1,26	4,392	0,27	5,82	0,8	0
b3	388,1	0,6	3	0,1	4,1	0,07	0
b2	262,2	0,28	1,58	0,073	1,96	0	0
b1	145,5	0,095	0,5	0,01	0,3	0	0

# Calculs des indices de concordance

## Étape 1 : Détermination des indices de concordance partiels

```
[ ] c = calcul_indices_de_concordance_partiels(criteres = criteres, aliments = aliments, profils = profils)
c

{0: {'aliment_1': {'b6': 1, 'b5': 1, 'b4': 1, 'b3': 1, 'b2': 1, 'b1': 1},
      'b6': {'aliment_1': 0, 'aliment_2': 0},
      'b5': {'aliment_1': 0, 'aliment_2': 0},
      'b4': {'aliment_1': 0, 'aliment_2': 0},
      'b3': {'aliment_1': 0, 'aliment_2': 0},
      'b2': {'aliment_1': 0, 'aliment_2': 0},
      'b1': {'aliment_1': 0, 'aliment_2': 0},
      'aliment_2': {'b6': 1, 'b5': 1, 'b4': 1, 'b3': 1, 'b2': 1, 'b1': 1}},
 1: {'aliment_1': {'b6': 0, 'b5': 0, 'b4': 1, 'b3': 1, 'b2': 1, 'b1': 1},
      'b6': {'aliment_1': 1, 'aliment_2': 1},
      'b5': {'aliment_1': 1, 'aliment_2': 0},
      'b4': {'aliment_1': 1, 'aliment_2': 0},
      'b3': {'aliment_1': 0, 'aliment_2': 0},
      'b2': {'aliment_1': 0, 'aliment_2': 0},
      'b1': {'aliment_1': 0, 'aliment_2': 0},
      'aliment_2': {'b6': 0, 'b5': 1, 'b4': 1, 'b3': 1, 'b2': 1, 'b1': 1}},
 2: {'aliment_1': {'b6': 0, 'b5': 0, 'b4': 0, 'b3': 0, 'b2': 1, 'b1': 1},
      'b6': {'aliment_1': 1, 'aliment_2': 1},
      'b5': {'aliment_1': 1, 'aliment_2': 1},
      'b4': {'aliment_1': 1, 'aliment_2': 1},
      'b3': {'aliment_1': 1, 'aliment_2': 0},
      'b2': {'aliment_1': 0, 'aliment_2': 0},
      'b1': {'aliment_1': 0, 'aliment_2': 0},
      'aliment_2': {'b6': 0, 'b5': 0, 'b4': 1, 'b3': 1, 'b2': 1, 'b1': 1}},
 3: {'aliment_1': {'b6': 0, 'b5': 0, 'b4': 0, 'b3': 0, 'b2': 0, 'b1': 1},
      'b6': {'aliment_1': 1, 'aliment_2': 1},
      'b5': {'aliment_1': 1, 'aliment_2': 1},
      'b4': {'aliment_1': 1, 'aliment_2': 1},
      'b3': {'aliment_1': 1, 'aliment_2': 1},
      'b2': {'aliment_1': 1, 'aliment_2': 1},
      'b1': {'aliment_1': 0, 'aliment_2': 0},
      'aliment_2': {'b6': 0, 'b5': 0, 'b4': 0, 'b3': 0, 'b2': 0, 'b1': 1}},
```

## Étape 2 : Détermination des indices de concordance globaux

```
[ ] C = calcul_indices_de_concordance_globaux(n = len(criteres), indices_de_concordance_partiels = c, proids = poids)
C

{'aliment_1': {'b6': 0.18181818181818182,
  'b5': 0.18181818181818182,
  'b4': 0.36363636363636365,
  'b3': 0.36363636363636365,
  'b2': 0.5454545454545454,
  'b1': 1.0},
 'b6': {'aliment_1': 0.8181818181818182, 'aliment_2': 0.8181818181818182},
 'b5': {'aliment_1': 0.8181818181818182, 'aliment_2': 0.5454545454545454},
 'b4': {'aliment_1': 0.8181818181818182, 'aliment_2': 0.454545454545453},
 'b3': {'aliment_1': 0.6363636363636364, 'aliment_2': 0.2727272727272727},
 'b2': {'aliment_1': 0.454545454545453, 'aliment_2': 0.181818181818182},
 'b1': {'aliment_1': 0.09090909090909091, 'aliment_2': 0.0},
 'aliment_2': {'b6': 0.18181818181818182,
  'b5': 0.5454545454545454,
  'b4': 0.7272727272727273,
  'b3': 0.7272727272727273,
  'b2': 0.8181818181818182,
  'b1': 1.0}}
```

## · Étape 3 & 4 : Relation de surclassement & Procédures d'affectation

### · 1) pessimiste

```
[ ] categories_pessimist = PessimisticmajoritySorting(categories = categories, aliments = aliments, profils = profils, indices_de_concordance_globaux = C, seuil_de_majorite = seuil  
categories_pessimist  
{'aliment_1': 'E', 'aliment_2': 'B'}
```

### · 1) optimiste

```
[ ] categories_optimist = OptimisticmajoritySorting(categories = categories, aliments = aliments, profils = profils, indices_de_concordance_globaux = C, seuil_de_majorite = seuil_d  
categories_optimist  
{'aliment_1': 'A', 'aliment_2': 'A'}
```

## ▼ Test sur des données venant des fichiers

Seuil de majorité à 0.5 et profils quantiles

```
[ ] profils_file = "profils_quantiles.xlsx"
criteres_file = "criteres2.xlsx"
aliments_file = "aliments.xlsx"
output_file_pesimite = "output_pesimite05.xlsx"
output_file_optimiste = "output_optimiste05.xlsx"
seuil_de_majorite = 0.5
```

```
[ ] criteres, poids = get_criteres_poids(file_path = criteres_file)
criteres, poids
```

```
([['energy100g', 'min'],
  ['saturatedfat100g', 'min'],
  ['sugars100g', 'min'],
  ['sodium', 'min'],
  ['proteins100g', 'max'],
  ['fiber100g', 'max'],
  ['teneur_en_fruit', 'max']],
 [1, 1, 1, 1, 2, 2, 2])
```

```
[ ] profils = get_profilis(file_path = profils_file)
profils
```

```
{'b6': [1466.0, 3.08, 14.64, 0.68, 8.0, 3.0, 0],
 'b5': [998.5, 1.75, 8.45, 0.48, 6.65, 2.0, 0],
 'b4': [498.0, 1.26, 4.392, 0.27, 5.82, 0.8, 0],
 'b3': [388.1, 0.6, 3.0, 0.1, 4.1, 0.07, 0],
 'b2': [262.2, 0.28, 1.58, 0.073, 1.96, 0.0, 0],
 'b1': [145.5, 0.095, 0.5, 0.01, 0.3, 0.0, 0]}
```

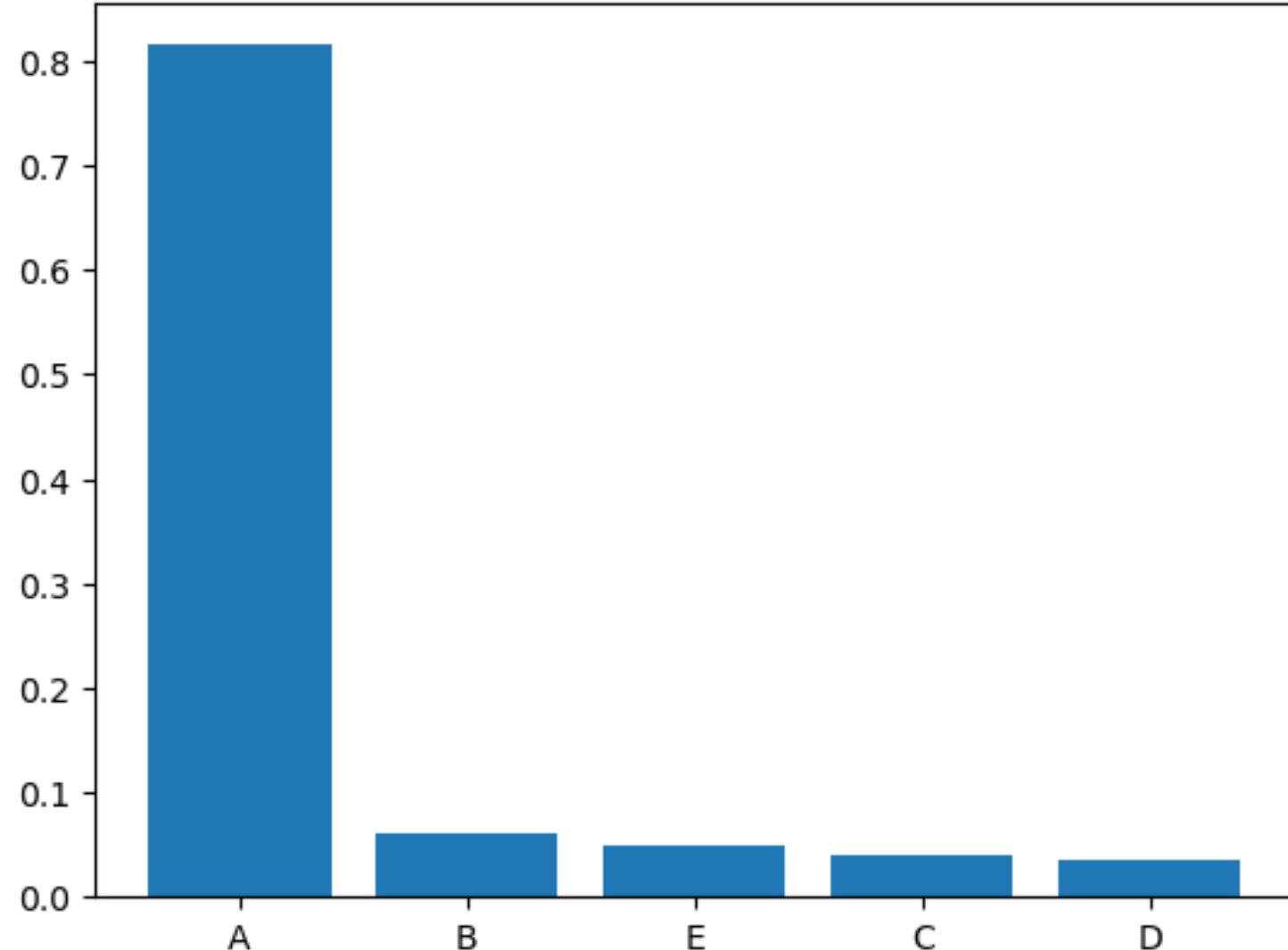
Etude des seuils  
majoritaires à 0.5, 0.6 et  
0.7.  
avec nos deux profils

# Simulation

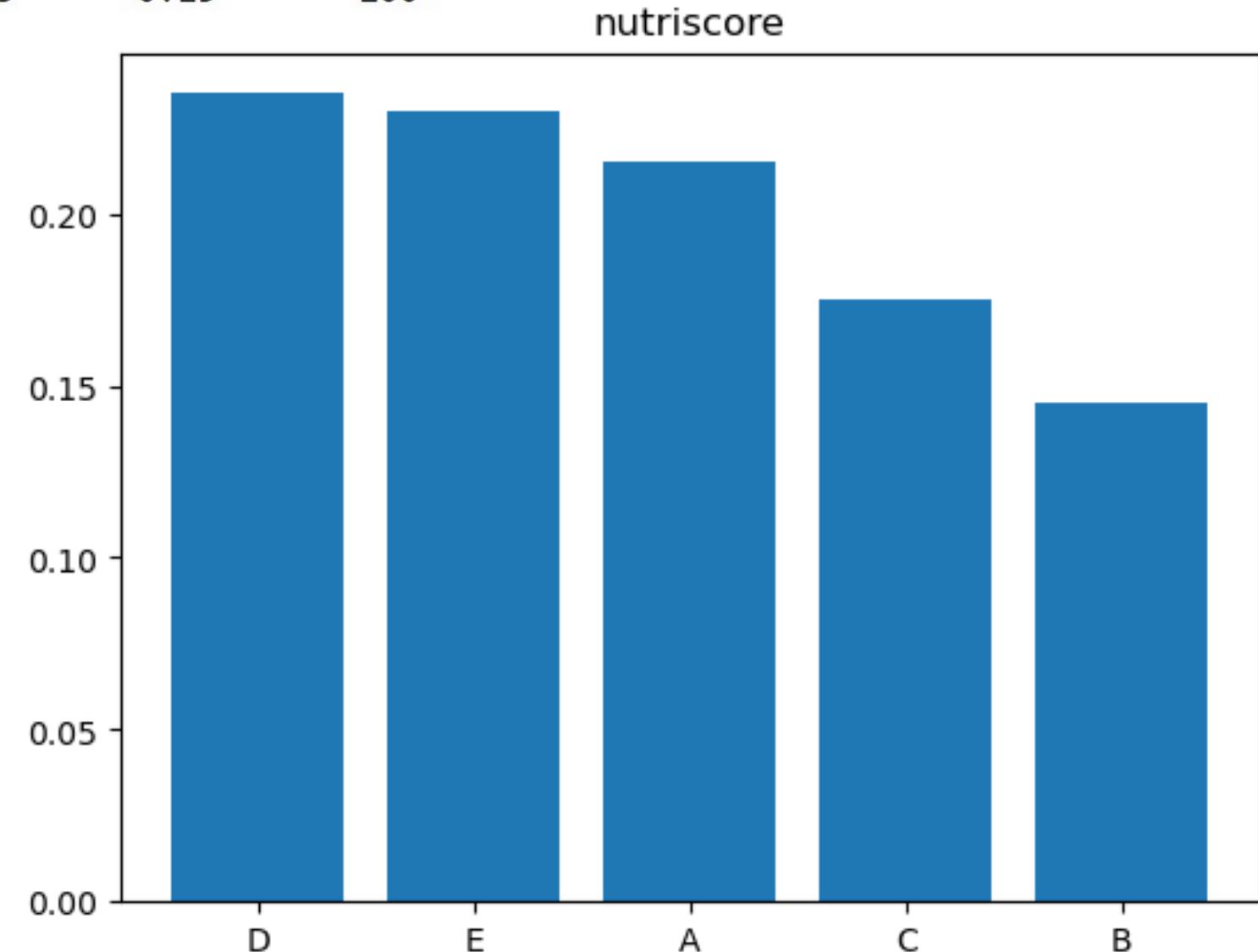
y=0.7

## pessimiste

categories



	precision	recall	f1-score	support
A	0.26	1.00	0.42	43
B	0.08	0.03	0.05	29
C	0.00	0.00	0.00	35
D	0.29	0.04	0.07	47
E	0.40	0.09	0.14	46
accuracy			0.25	200
macro avg	0.21	0.23	0.14	200
weighted avg	0.23	0.25	0.15	200



# **Partie 2 : le Super Nutri-Score**

# Élaboration d'un modèle Super-NutriScore

Nous allons créer notre propre système de notation basé sur le nutri-score et le score Nova, nous aurons 5 catégories :

Un aliment sera dans la catégorie A si :

- il a un score Nova de 1 et un nutri score de A
- il a un score Nova de 1 et un nutri score de B
- il a un score Nova de 2 et un nutri score de A

Un aliment sera dans la catégorie B si :

- il a un score Nova de 1 et un nutri score de C
- il a un score Nova de 2 et un nutri score de B
- il a un score Nova de 2 et un nutri score de C

Un aliment sera dans la catégorie C si :

- il a un score Nova de 1 et un nutri score de D
- il a un score Nova de 2 et un nutri score de D
- il a un score Nova de 3 et un nutri score de A
- il a un score Nova de 3 et un nutri score de B
- il a un score Nova de 4 et un nutri score de A

Un aliment sera dans la catégorie D si :

- il a un score Nova de 1 et un nutri score de E
- il a un score Nova de 2 et un nutri score de E
- il a un score Nova de 3 et un nutri score de C
- il a un score Nova de 3 et un nutri score de D
- il a un score Nova de 4 et un nutri score de B
- il a un score Nova de 4 et un nutri score de C

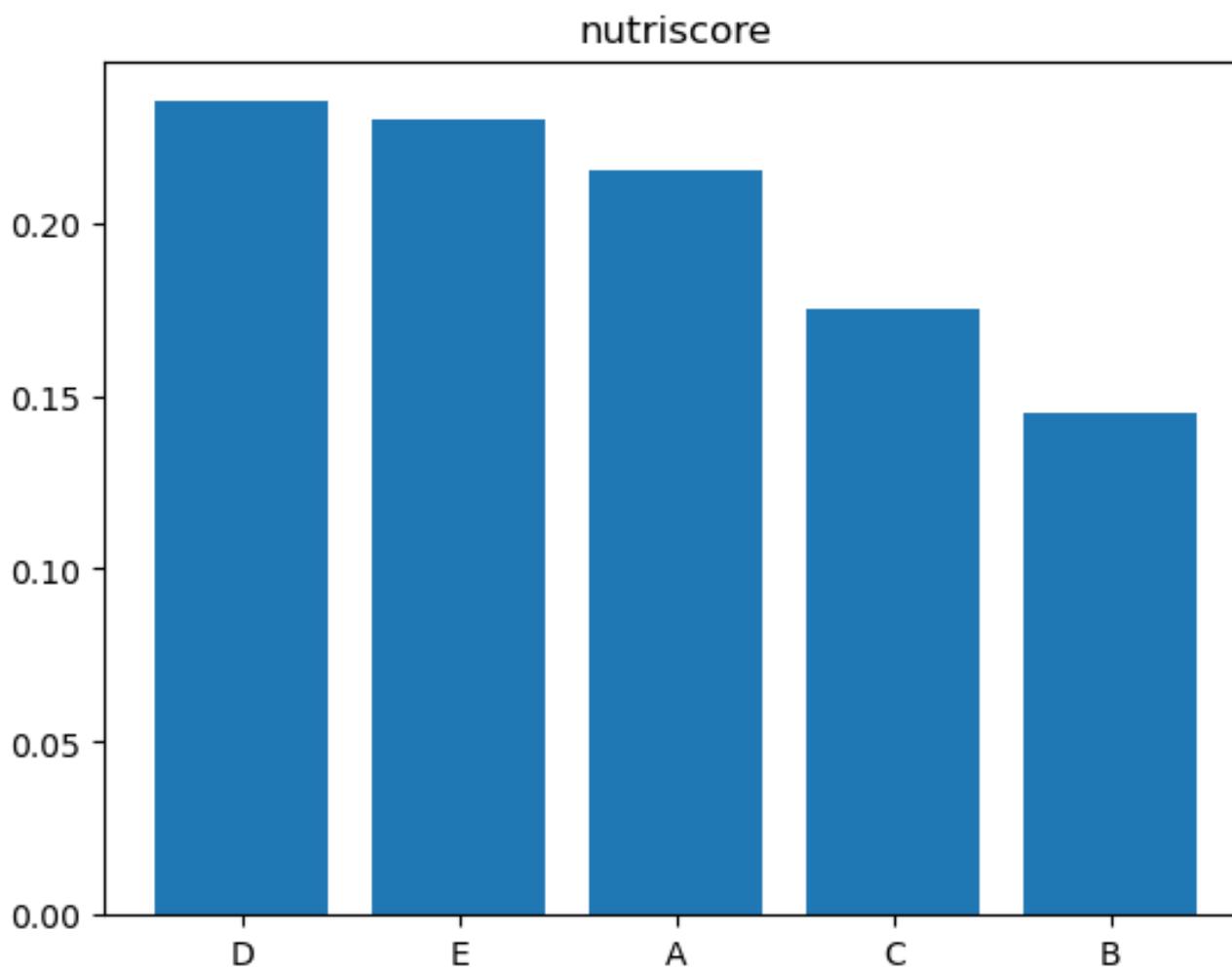
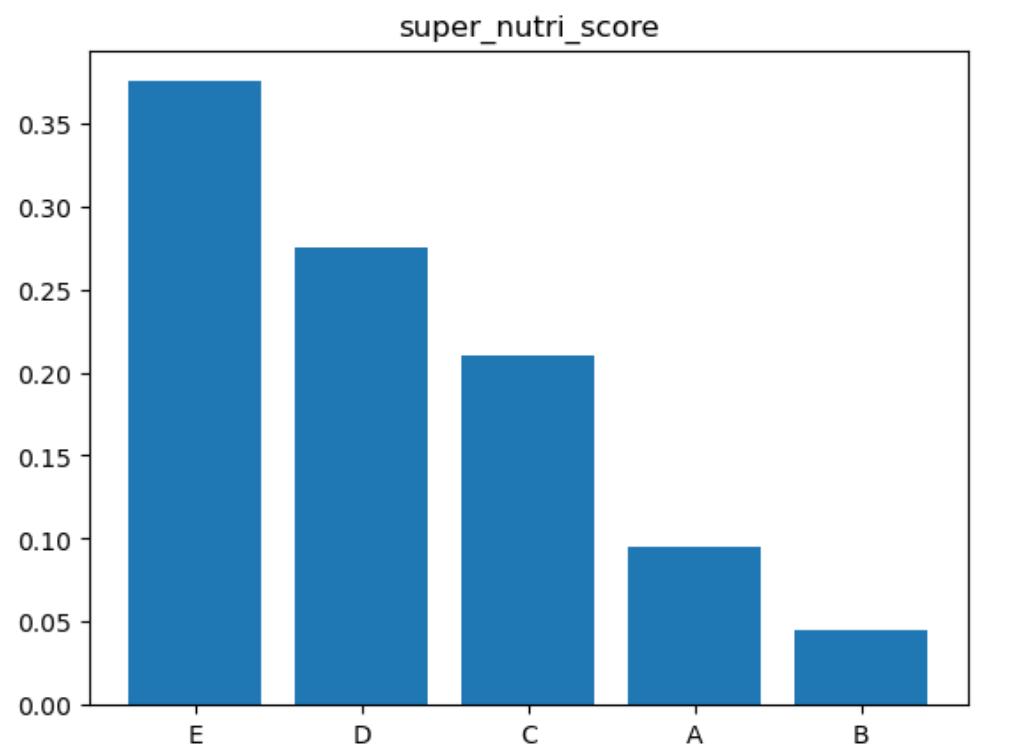
Un aliment sera dans la catégorie E si :

- il a un score Nova de 3 et un nutri score de E
- il a un score Nova de 4 et un nutri score de D
- il a un score Nova de 4 et un nutri score de E

```
[ ] df.head(10)
```

	aliments	energy100g	saturatedfat100g	sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fruit	Nova	nutriscore	nom	super_nutri_score
0	0	467.0	5.6	32.0	0.490	6.3	4.0	0.0	4	D	prince de lu	E
1	1	0.0	0.0	0.0	0.003	0.0	0.0	0.0	1	A	volvic 1.5L	A
2	2	495.0	12.0	28.0	1.200	6.0	3.5	0.0	4	E	granola	E
3	3	43.0	0.0	8.9	0.000	0.8	0.6	100.0	1	C	Tropican Orange	B
4	4	57.0	0.1	3.9	0.090	10.0	0.0	0.0	1	A	Skyr Danone	A
5	5	159.0	0.7	2.3	5.650	7.1	1.9	0.0	2	C	amora Moutarde	B
6	6	500.0	13.0	27.1	0.483	6.6	3.3	0.0	4	E	Kinder cereAlé	E
7	7	20.0	0.0	4.5	0.010	0.0	0.0	0.0	1	D	Ice tea Peche	C
8	8	66.0	0.4	7.5	0.200	3.7	0.9	0.0	1	A	AlPro Vanille	A
9	9	23.0	0.0	0.8	0.010	0.2	0.0	45.0	1	C	Pulco citron	B

	precision	recall	f1-score	support
A	0.79	0.35	0.48	43
B	0.00	0.00	0.00	29
C	0.00	0.00	0.00	35
D	0.15	0.17	0.16	47
E	0.57	0.93	0.71	46
accuracy				200
macro avg	0.30	0.29	0.27	200
weighted avg	0.34	0.33	0.30	200



Si on note  $\text{Nutri-Score}(x)$  (respectivement  $F(x)$ ) le score de l'aliment  $x$  obtenu par le calcul du Nutri-score (respectivement par application de la fonction  $F$  de l'Équation 4), alors on a le résultat suivant :

$$\text{Nutri-Score}(x) = 40 - F(x) \quad (5)$$

Cette Équation 5 montre que l'algorithme de calcul du Nutri-score est équivalent à une simple somme pondérée, en l'occurrence la fonction  $F$  de l'Équation 4. Ce qui suppose que les 7 critères utilisés doivent être indépendants entre eux. Or la formule de l'énergie ci-dessus montre que c'est un critère dépendant des acides gras, du sucre, des protéines et des fibres. Cette dépendance contredit l'utilisation de la somme pondérée.

$$\text{Energy} = (9 \times \text{fat}) + (7 \times \text{alcohol}) + (4 \times \text{protein}) + (4 \times \text{sugar}) + (2.4 \times \text{organic acids}) + (2.4 \times \text{polyols}) + (2 \times \text{fibers})$$

Pour résoudre ce problème de la dépendance des critères, nous allons considérer dans ce projet les deux fonctions suivantes :

1. La composante énergie n'est pas prise en compte dans le calcul du score nutritionnel de l'aliment

$$x = (x'_{en}, x'_{su}, x'_{sa}, x'_{so}, x'_{pr}, x'_{fi}, x'_{fr}) :$$

$$N_{eo}(x'_{su}, x'_{sa}, x'_{so}, x'_{pr}, x'_{fi}, x'_{fr}) = x'_{su} + x'_{sa} + x'_{so} + \frac{1}{2}(x'_{pr} + x'_{fi} + x'_{fr}) \quad (6)$$

2. Les 4 composantes "acides gras", "sodium", "protéines" et "fibres" ne sont pas pris en compte dans le calcul du score nutritionnel de l'aliment

$$x = (x'_{en}, x'_{su}, x'_{sa}, x'_{so}, x'_{pr}, x'_{fi}, x'_{fr}) :$$

$$N_{ei}(x'_{en}, x'_{so}, x'_{fr}) = x'_{en} + x'_{so} + \frac{1}{2}x'_{fr} \quad (7)$$

# Le Nutri-score vu comme un problème d'Aide Multicritères à la Décision

```

# Fonction pour affecter une classe Nutri-Score en fonction de la somme pondérée calcul
def calculate_nutri_score_from_sum_6(sum_6):
    if sum_6 < 15:
        return 'A'
    elif sum_6 >= 15 and sum_6 < 25:
        return 'B'
    elif sum_6 >= 25 and sum_6 < 35:
        return 'C'
    elif sum_6 >= 35 and sum_6 < 45:
        return 'D'
    else:
        return 'E'

# Fonction pour affecter une classe Nutri-Score en fonction de la somme pondérée calcul
def calculate_nutri_score_from_sum_7(sum_7):
    if sum_7 < 30:
        return 'A'
    elif sum_7 >= 30 and sum_7 < 100:
        return 'B'
    elif sum_7 >= 100 and sum_7 < 200:
        return 'C'
    elif sum_7 >= 200 and sum_7 < 500:
        return 'D'
    else:
        return 'E'
import pandas as pd

# Chargement des données à partir du fichier Excel
df = pd.read_excel('aliments avec nova et nutri score.xlsx')

# Ajout d'une colonne vide pour la classe Nutri-Score
df['nutri_score_equation'] = None

```

```

# Calcul de la somme pondérée pour chaque aliment
for index, row in df.iterrows():
    # Récupération des valeurs des critères pour l'aliment en cours
    sugar = row['sugars100g']
    saturated_fat = row['saturatedfat100g']
    sodium = row['sodium100g']
    proteins = row['proteins100g']
    fiber = row['fiber100g']
    fruit = row['teneur_en_fruit']
    energy = row['energy100g']

    # Calcul de la somme pondérée en utilisant l'équation (6)
    sum_6 = (sugar * 1) + (saturated_fat * 1) + (sodium * 1) + (proteins * 0.5) + (fiber * 0.5)

    # Calcul de la somme pondérée en utilisant l'équation (7)
    sum_7 = (energy * 1) + (sodium * 1) + (fruit * 0.5)

    # Détermination de la classe Nutri-Score en fonction de la somme pondérée la plus élevée
    nutri_score = None
    if sum_6 < sum_7:
        nutri_score = calculate_nutri_score_from_sum_7(sum_7)
    else:
        nutri_score = calculate_nutri_score_from_sum_6(sum_6)

    # Ajout de la classe Nutri-Score au DataFrame
    df.loc[index, 'nutri_score_equation'] = nutri_score

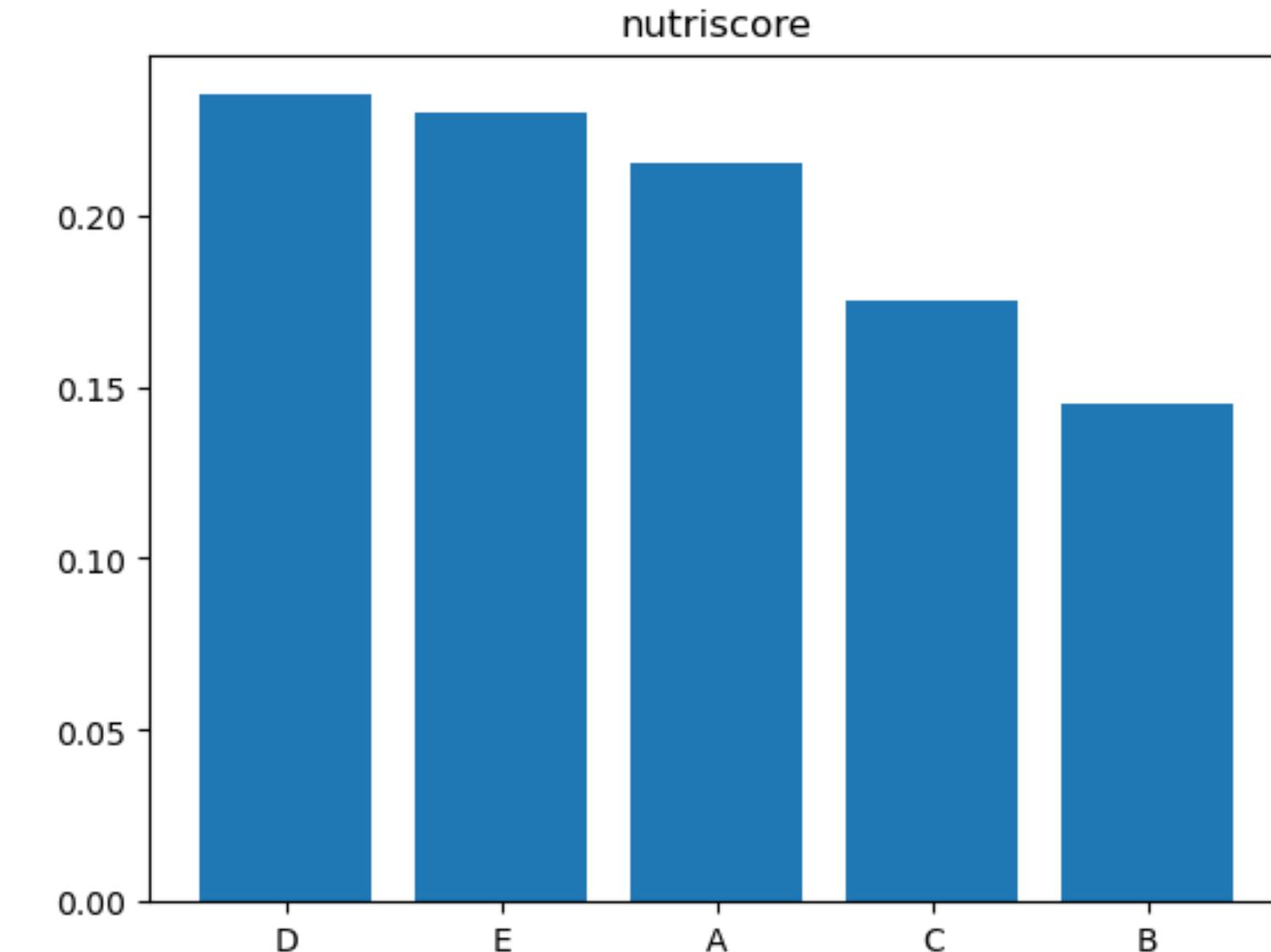
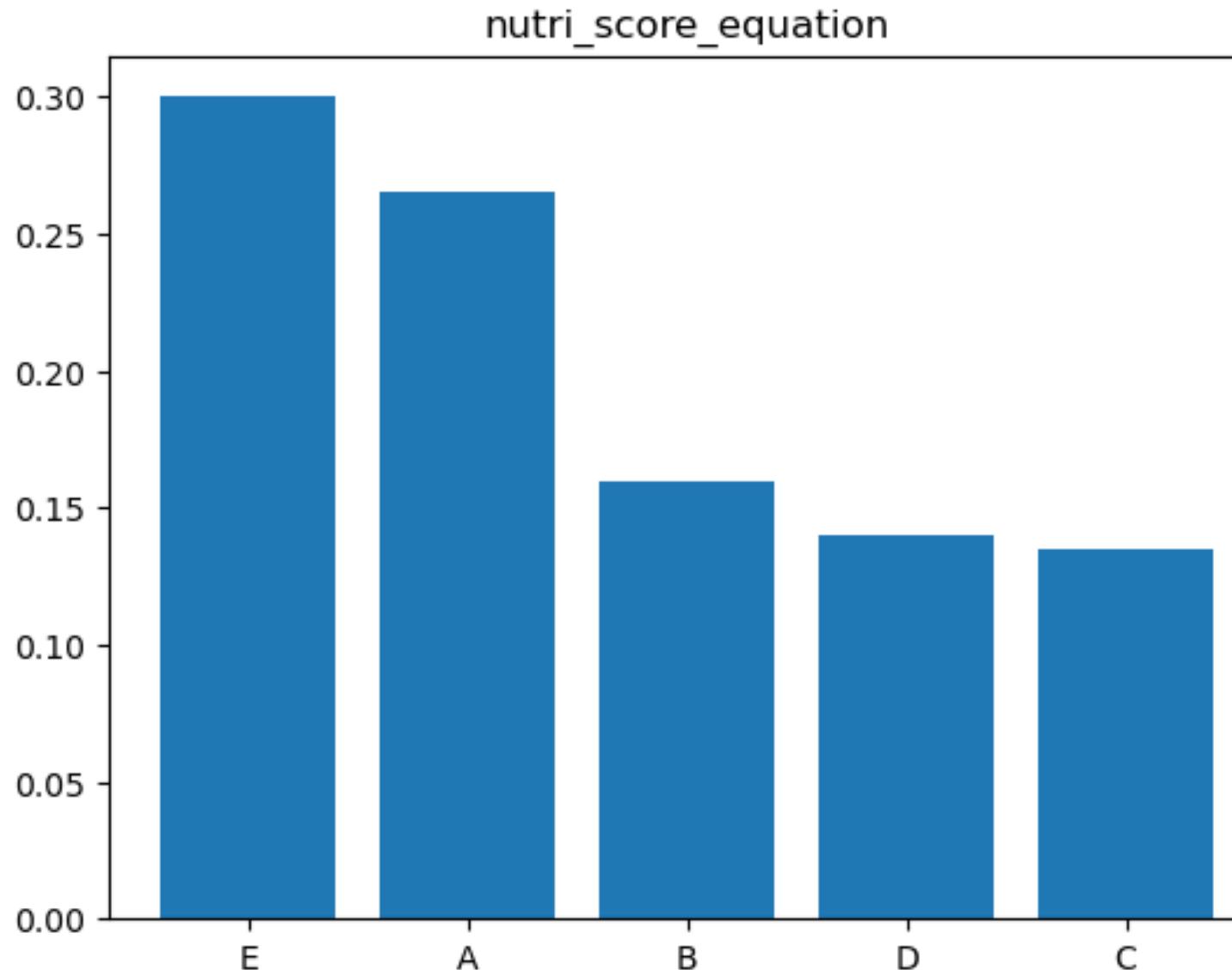
```

```
[ ] df.head(10)
```

	aliments	energy100g	saturatedfat100g	sugars100g	sodium100g	proteins100g	fiber100g	teneur_en_fruit	Nova	nutriscore	nom	nutri_score_equation
0	0	467.0	5.6	32.0	0.490	6.3	4.0	0.0	4	D	prince de lu	D
1	1	0.0	0.0	0.0	0.003	0.0	0.0	0.0	1	A	volvic 1.5L	A
2	2	495.0	12.0	28.0	1.200	6.0	3.5	0.0	4	E	granola	E
3	3	43.0	0.0	8.9	0.000	0.8	0.6	100.0	1	C	Tropicana Orange	E
4	4	57.0	0.1	3.9	0.090	10.0	0.0	0.0	1	A	Skyr Danone	A
5	5	159.0	0.7	2.3	5.650	7.1	1.9	0.0	2	C	amora Moutarde	A
6	6	500.0	13.0	27.1	0.483	6.6	3.3	0.0	4	E	Kinder cereAlé	E
7	7	20.0	0.0	4.5	0.010	0.0	0.0	0.0	1	D	Ice tea Peche	A
8	8	66.0	0.4	7.5	0.200	3.7	0.9	0.0	1	A	AlPro Vanille	A
9	9	23.0	0.0	0.8	0.010	0.2	0.0	45.0	1	C	Pulco citron	B

# Simulation Nutri-score équation

	precision	recall	f1-score	support
A	0.40	0.49	0.44	43
B	0.16	0.17	0.16	29
C	0.11	0.09	0.10	35
D	0.36	0.21	0.27	47
E	0.55	0.72	0.62	46
accuracy			0.36	200
macro avg	0.31	0.34	0.32	200
weighted avg	0.34	0.36	0.34	200



# **Conclusion :**

Quelle analyse faites-vous de la transparence de vos modèles élaborés ci-dessus, à travers les cinq propriétés vues en cours (loyauté, équité, responsabilité, explicabilité, intelligibilité).

D'autres modèles de décision, comme les arbres de décision, peuvent-ils être plus transparents que les modèles étudiés ci-dessus ? Pourquoi ?

# **Autres approches envisagées ?**