OPENCLASSROOMS

PROJET 3:

CONCEVEZ UNE APPLICATION AU SERVICE DE LA SANTÉ PUBLIQUE

Aymen Djiab

Sommaire :

1- Introduction

L'agence " Santé Publique France" a lancé un appel à projets pour trouver des idées innovantes d’applications en lien avec l'alimentation. On souhaite y participer et proposer une idée d’application.

Étant donné que j'aime le sport et j’en pratique énormément, il sera plus facile pour moi de comprendre les produits qui sont apte pour le sport, ainsi programmer une application qui permettra aux consommateurs qui cherchent à optimiser sa qualité nutritionnelle pourra le faire.

- Objectifs de l’application :

Le but de cette application, consiste à savoir le grade moyen des produits achetés.

Par exemple : si j’ai acheté 10 produits, j’aurais un niveau de nutri-grades moyen (de A à E) pour les 10 produits achetés, est cela indiquera si oui ou non cela est bon pour notre corps.

2- Nettoyage des données

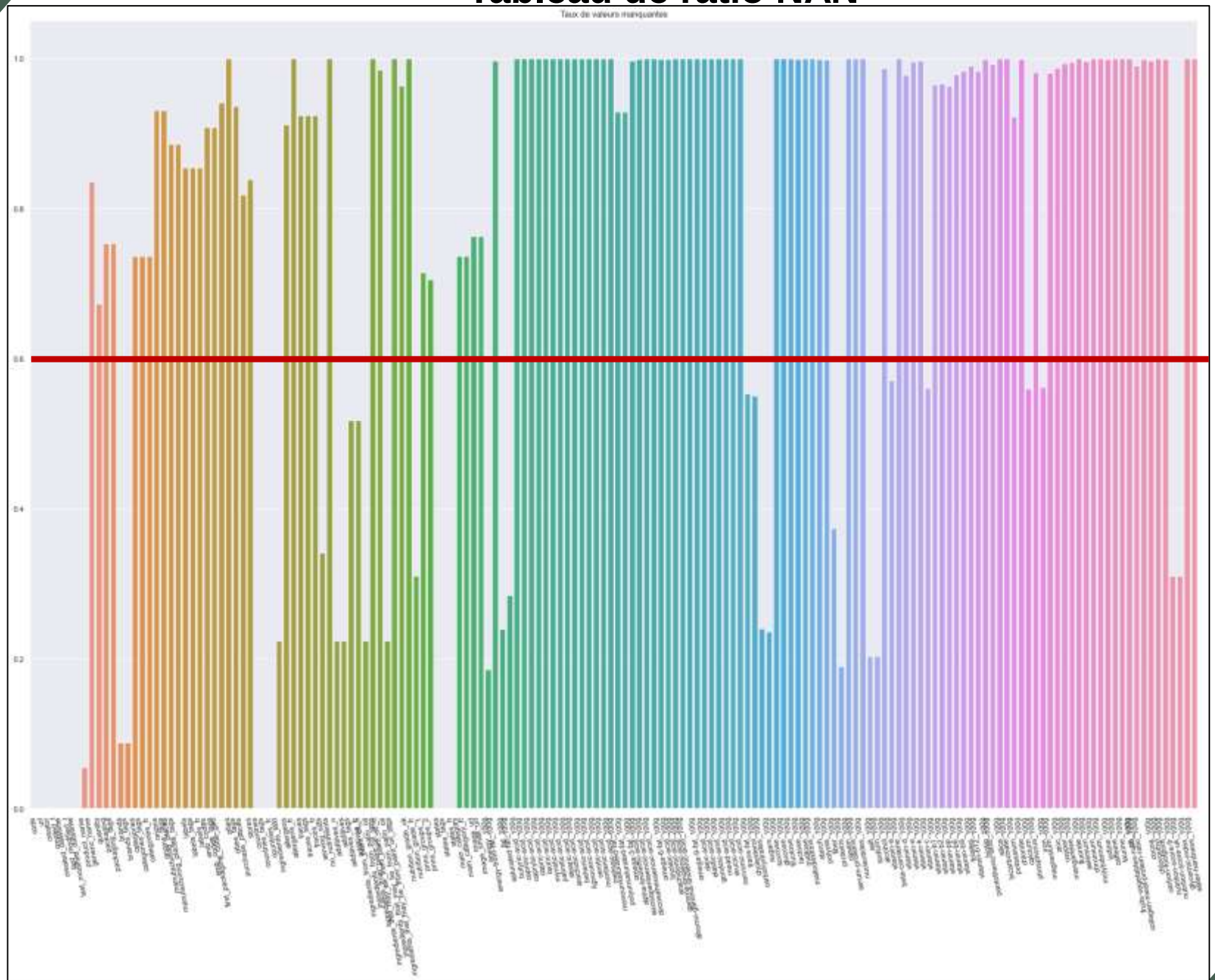
Description fichier & Méthode « Is null » / fonction NAN

fr.openfoodfacts.org.products.csv

Données sur les produits: pays de production dates de fabrication, apport caloriques…
320 772 lignes et 162 colonnes, pas de doublons

En mettant en évidence les éventuelles valeurs manquantes, avec au moins 3 méthodes de traitement adaptées aux variables concernées.

Tableau de ratio NAN



Sélection d'un seuil pour colonnes

On garde les colonnes qui sont inférieur à 60% de NAN

```
Out[16]: ['code',
          'url',
          'creator',
          'created_t',
          'created_datetime',
          'last_modified_t',
          'last_modified_datetime',
          'product_name',
          'brands',
          'brands_tags',
          'countries',
          'countries_tags',
          'countries_fr',
          'ingredients_text',
          'serving_size',
          'additives_n',
          'additives',
          'additives_tags',
          'additives_fr',
          'ingredients_from_palm_oil_n',
          'ingredients_that_may_be_from_palm_oil_n',
          'nutrition_grade_fr',
          'states',
          'states_tags',
          'states_fr',
          'energy_100g',
          'fat_100g',
          'saturated-fat_100g',
          'trans-fat_100g',
          'cholesterol_100g',
          'carbohydrates_100g',
          'sugars_100g',
          'fiber_100g',
          'proteins_100g',
          'salt_100g',
          'sodium_100g',
          'vitamin-a_100g',
          'vitamin-c_100g',
          'calcium_100g',
          'iron_100g',
          'nutrition-score-fr_100g',
```

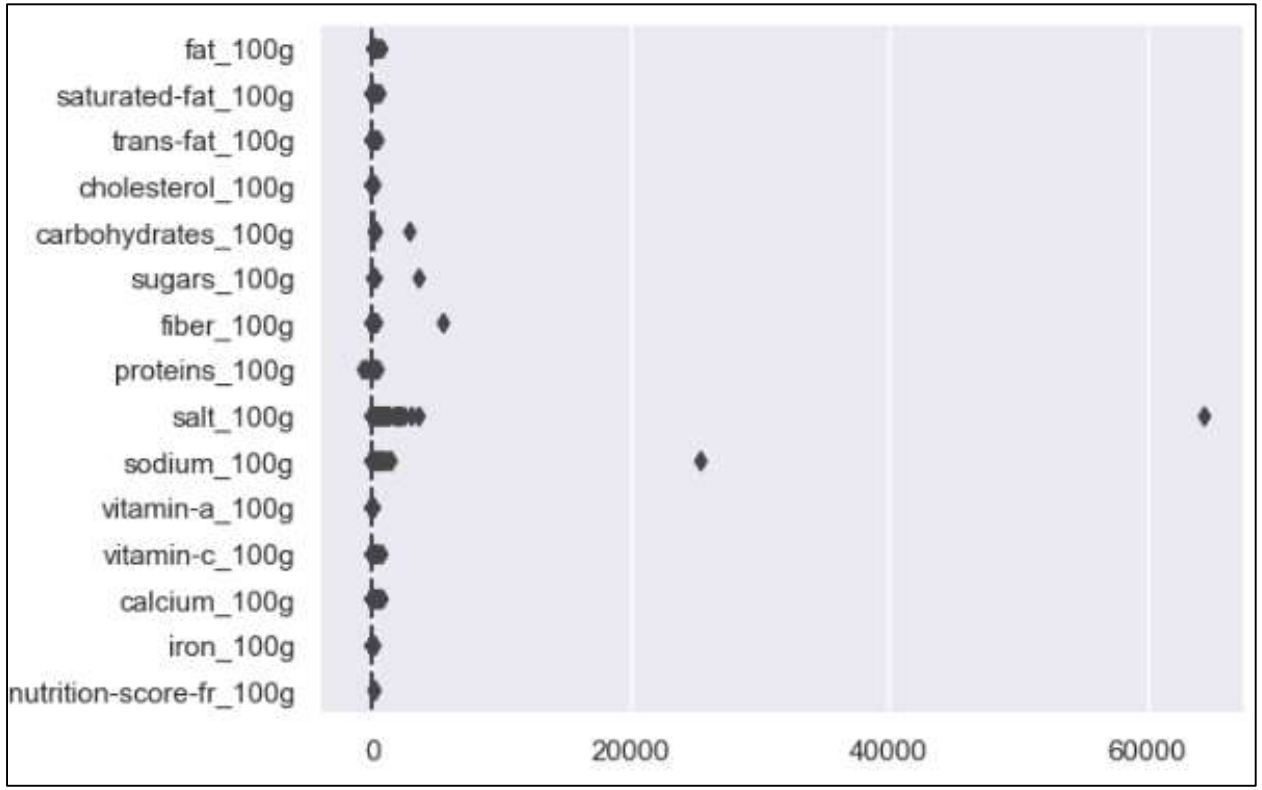
	created_datetime	last_modified_datetime
0	2016-09-17T09:17:46Z	2016-09-17T09:18:13Z
1	2017-03-09T14:32:37Z	2017-03-09T14:32:37Z
2	2017-03-09T14:32:37Z	2017-03-09T14:32:37Z
3	2017-03-09T10:35:31Z	2017-03-09T10:35:31Z
4	2017-03-09T10:34:13Z	2017-03-09T10:34:13Z
...	...	...
320767	2017-03-27T16:14:59Z	2017-04-03T18:34:58Z
320768	2017-03-09T11:31:16Z	2017-04-03T18:34:59Z
320769	2015-01-24T11:36:17Z	2017-04-03T18:34:59Z
320770	2017-04-16T10:54:49Z	2017-04-16T10:54:49Z
320771	2017-03-09T15:18:29Z	2017-04-03T18:34:59Z

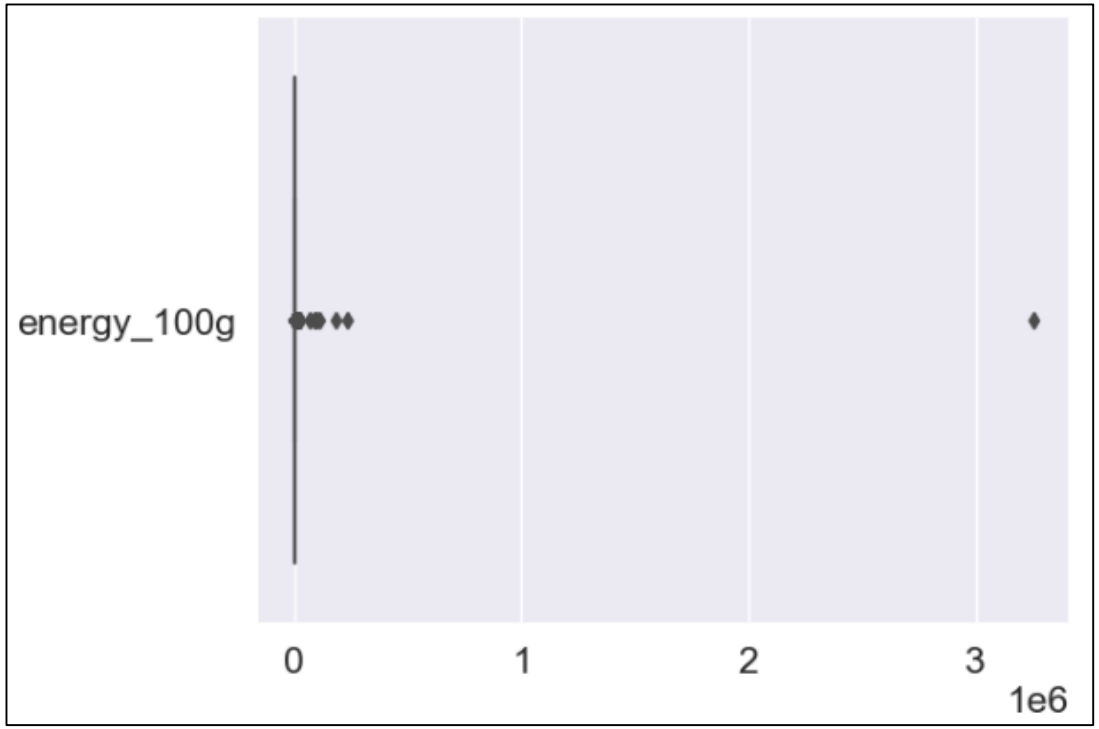
```
[26]: 1 cols_datetime.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 320772 entries, 0 to 320771
Data columns (total 2 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   created_datetime       320748 non-null  datetime64[ns, UTC]
 1   last_modified_datetime 320749 non-null  datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](2)
memory usage: 4.9 MB
```

	created_datetime	last_modified_datetime
0	2016-09-17 09:17:46+00:00	2016-09-17 09:18:13+00:00
1	2017-03-09 14:32:37+00:00	2017-03-09 14:32:37+00:00
2	2017-03-09 14:32:37+00:00	2017-03-09 14:32:37+00:00
3	2017-03-09 10:35:31+00:00	2017-03-09 10:35:31+00:00
4	2017-03-09 10:34:13+00:00	2017-03-09 10:34:13+00:00
...	...	...
320767	2017-03-27 16:14:59+00:00	2017-04-03 18:34:58+00:00
320768	2017-03-09 11:31:16+00:00	2017-04-03 18:34:59+00:00
320769	2015-01-24 11:36:17+00:00	2017-04-03 18:34:59+00:00
320770	2017-04-16 10:54:49+00:00	2017-04-16 10:54:49+00:00
320771	2017-03-09 15:18:29+00:00	2017-04-03 18:34:59+00:00

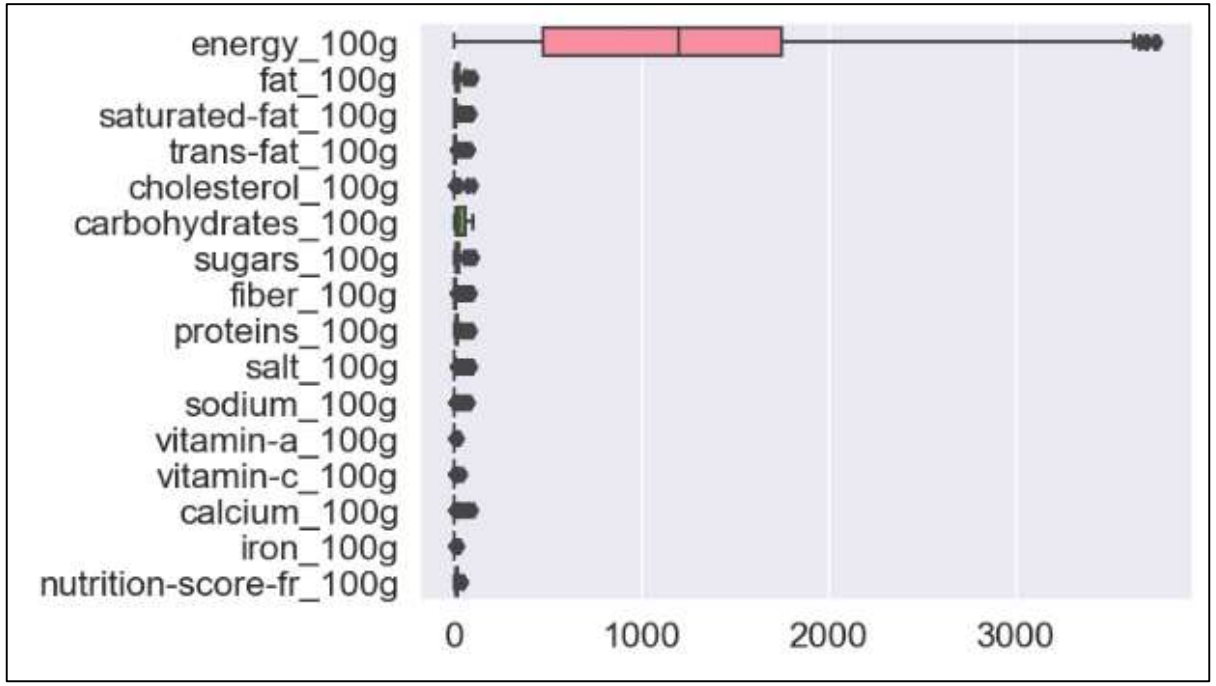
Analyse des Ingrédients 100g





Traitement adaptées aux variables concernées

Après avoir sélectionner les features numérique à un seuil de 60%, on passe aux aspects techniques



KNeighbourClassifier

```
nombre de voisins: K= 2 - pour une précision de : 73.42%
nombre de voisins: K= 3 - pour une précision de : 74.28%
nombre de voisins: K= 4 - pour une précision de : 73.58%
nombre de voisins: K= 5 - pour une précision de : 73.58%
nombre de voisins: K= 6 - pour une précision de : 72.97%
nombre de voisins: K= 7 - pour une précision de : 72.82%
nombre de voisins: K= 8 - pour une précision de : 72.46%
nombre de voisins: K= 9 - pour une précision de : 72.19%
nombre de voisins: K= 10 - pour une précision de : 71.67%
nombre de voisins: K= 11 - pour une précision de : 71.41%
nombre de voisins: K= 12 - pour une précision de : 71.12%
nombre de voisins: K= 13 - pour une précision de : 70.87%
nombre de voisins: K= 14 - pour une précision de : 70.55%
```

Traitement adaptées aux variables concernées

En utilisant l'algorithme KNeighbourClassifier, on obtient le meilleurs nombre de voisin qui est de K=3, donc en faisant le test on obtient un prédiction du score.

y_test: est égale à des chiffres mais c'est bien des grades chiffrés, ex: [0=A; 1=B; …; 4=E].

]:

	energy_100g	fat_100g	saturated-fat_100g	trans-fat_100g	cholesterol_100g	carbohydrates_100g	sugars_
169337	247.0	0.3	0.1	0.3	0.0	12.4	
49799	1326.0	8.0	2.2	1.0	0.2	83.3	
54029	837.0	10.8	6.9	9.1	0.1	23.1	
39465	1766.0	11.1	10.0	1.0	0.1	55.6	
61931	418.0	0.9	0.6	5.2	0.0	20.0	
...	...	...	...	...	...	...	
4635	908.0	8.7	5.1	0.6	0.0	31.9	
122077	1623.0	16.4	0.9	1.2	0.2	46.5	
16577	803.0	11.5	1.9	0.3	0.1	3.9	
20378	1464.0	10.0	3.3	22.1	0.2	86.7	
89039	159.0	8.8	2.5	0.6	0.2	9.6	

131781 rows × 16 columns

56]:

```
# Prédiction de score compris entre 0 et 5, 0 = A , 1 = B , 2 = C, 3
# ici le 1er produit affiche 3 donc grade D, le 2eme produit grade E

y_test
```

]: array([1., 3., 3., ..., 3., 3., 1.])

```
# ici je test comme EXEMPLE un produit sélectionner au hasard avec des ingrédients au hasard pour pouvoir
# obtenir un numéro de grade compris entre 0 et 4.

test = pd.DataFrame([[1246.0, 11.8, 3.7, 0.3, 0.0, 65.0, 10.6, 5.0, 20.0, 20.0, 0.7, 0.0, 0.0, 0.0, 0.5, 0.0]], c
test
```

	energy_100g	fat_100g	saturated-fat_100g	trans-fat_100g	cholesterol_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	sodium_100g	vitamin-a_100g
0	1246.0	11.8	3.7	0.3	0.0	65.0	10.6	5.0	20.0	20.0	0.7	0.0

```
# ici mon produit test est de grade 1 sois grade B
model.predict(test)
```

```
array([2.])
```

En donnant un exemple, on test un produits avec des ingrédients au hasard et on peux voir qu'on obtient un score.

Ici c'est 2 sois grade C.

Bilan Nettoyage

Out[73]:

product_name	countries_fr	ingredients_text	serving_size	additives_n	additives	additive
					[bananas -> en:bananas] [vegetable-oil -> en:vegetable-oil] [oil -> en:oil] [coconut-oil -> en:coconut-oil] [oil -> en:oil] [corn-oil-and-or-palm-oil-sugar -> en:corn-oil-and-or-palm-oil-sugar] [oil-and-or-palm-oil-sugar -> en:oil-and-or-	
		Bananas,				

Entrée [75]:

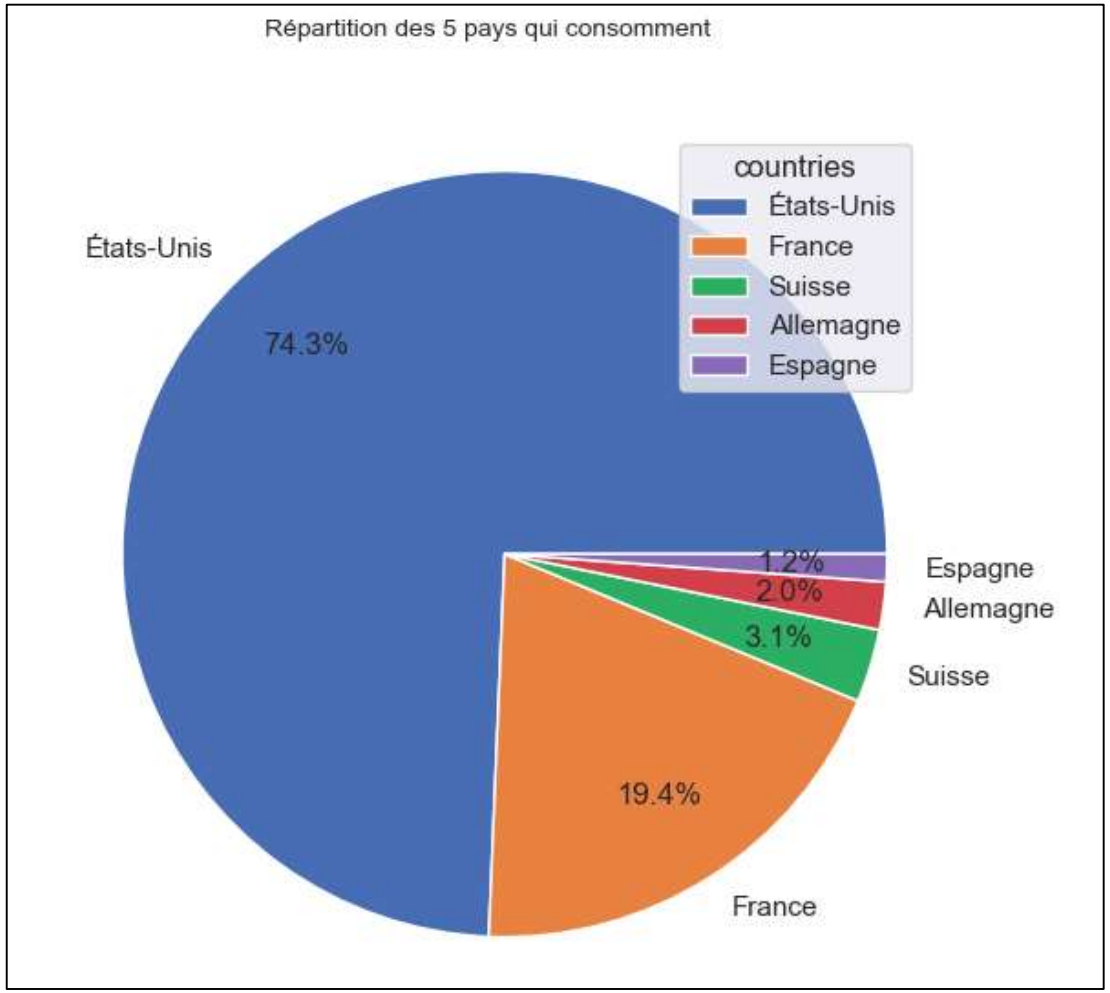
```
# ici le fichier cleaner finale avec toutes les colonnes nécessaires
df_clean.to_csv("fichier_cleaner.csv")
```

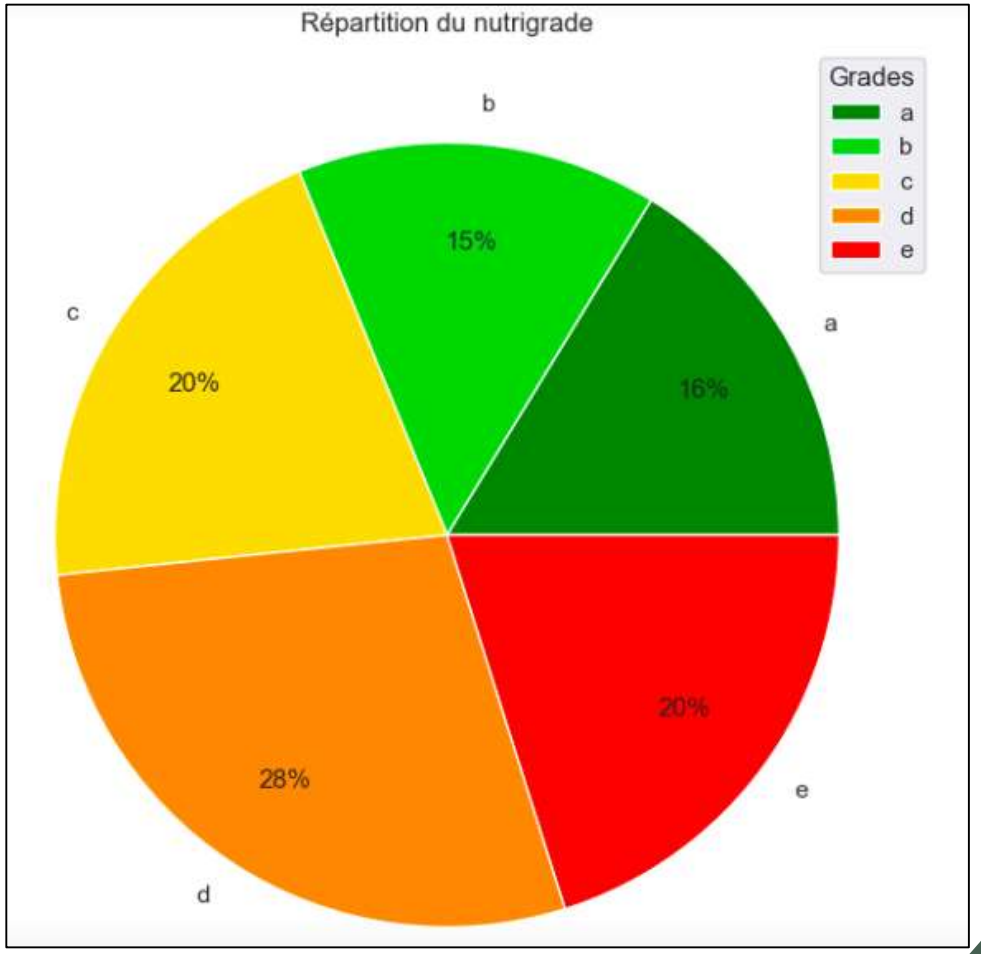
→ On à maintenant 188258 lignes et 28 colonnes

fichier_cleaner.csv	369 MB
fr.openfoodfacts.org.products.csv	847 MB

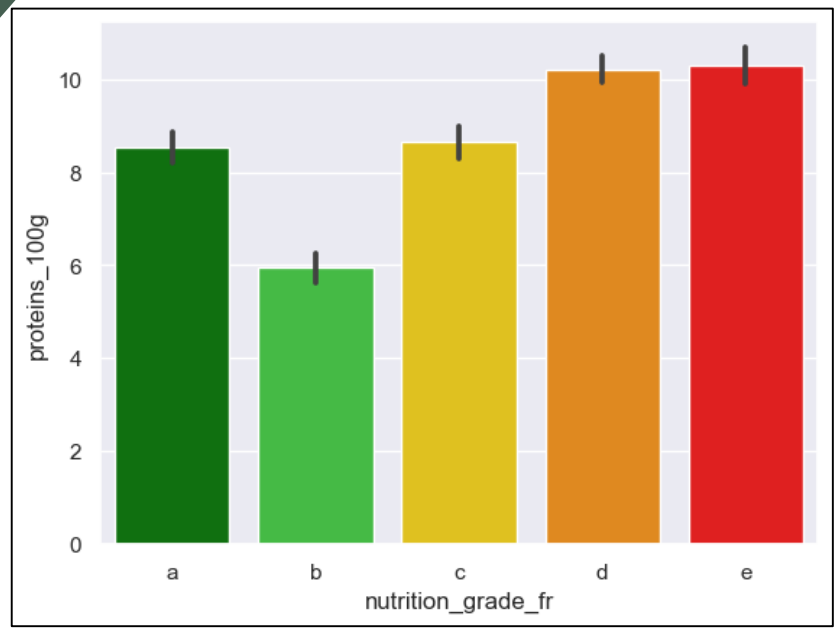
3 – Analyse exploratoire

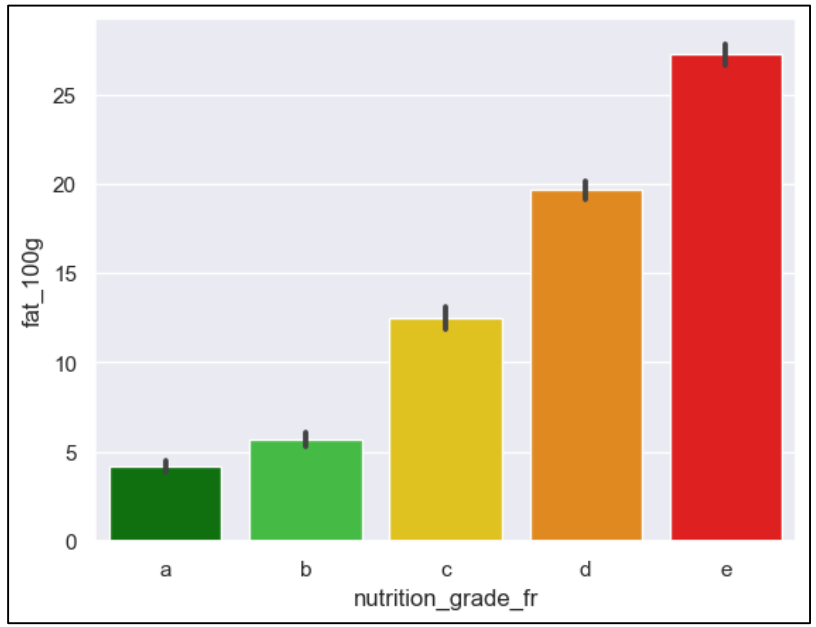
Pie plot

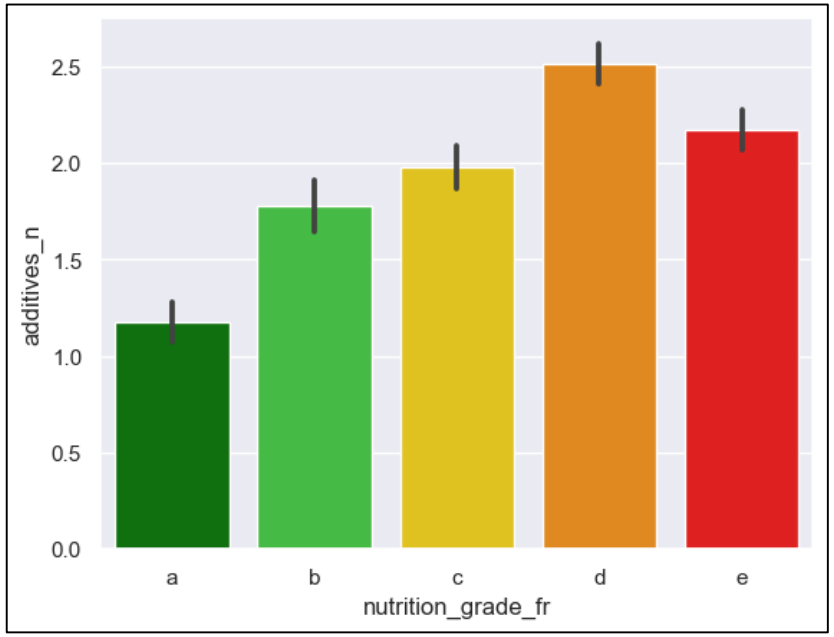

Répartition des 5 pays qui consomment
countries
États-Unis
France
Suisse
Allemagne
Espagne
États-Unis
74.3%
19.4%
France
3.1%
Suisse
2.0%
Allemagne
1.2%
Espagne

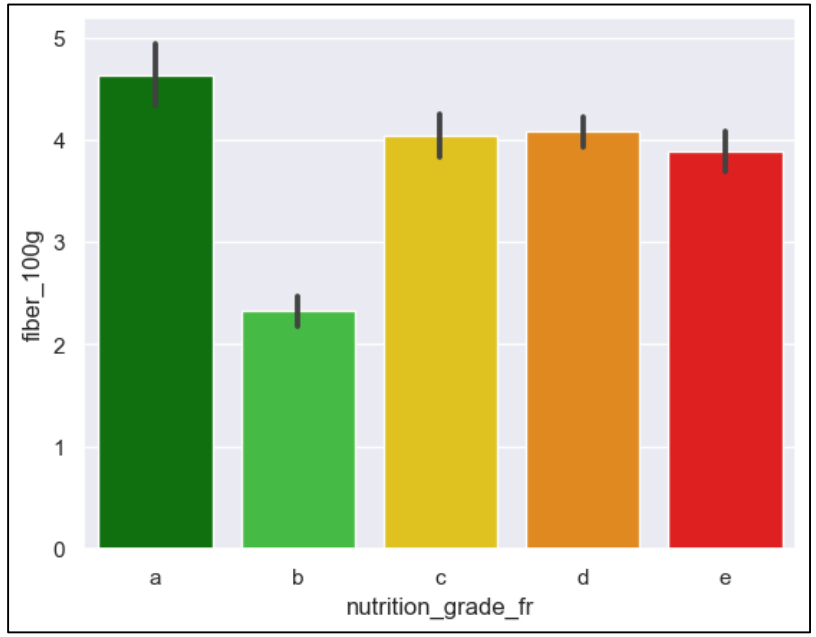

Répartition du nutrigrade
Grades
a
b
c
d
e
b
15%
c
20%
a
16%
20%
e
28%
d

Barplots

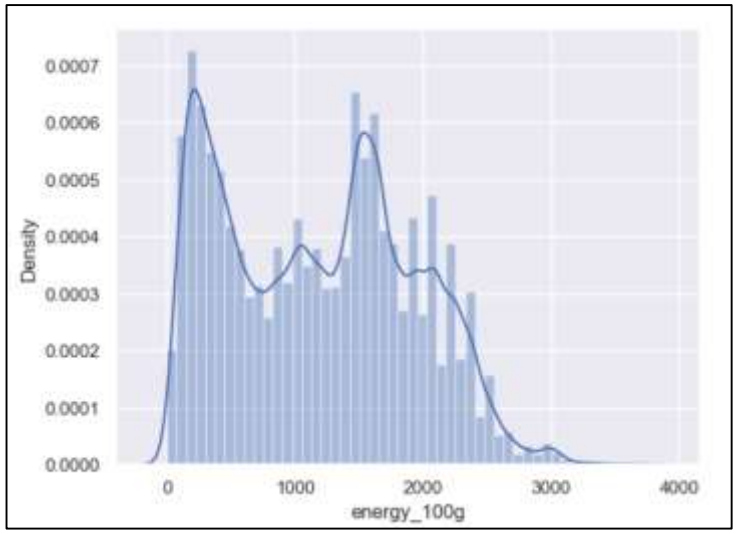

proteins_100g
0
2
4
6
8
10
a
b
c
d
e
nutrition_grade_fr

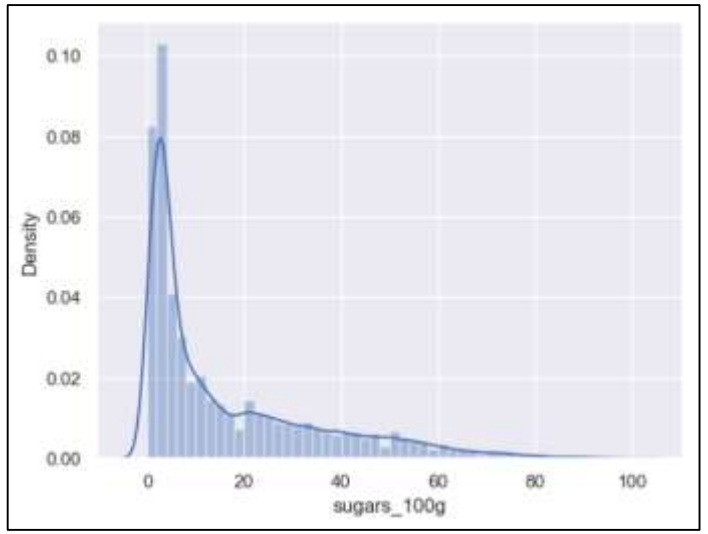

fat_100g
0
5
10
15
20
25
a
b
c
d
e
nutrition_grade_fr

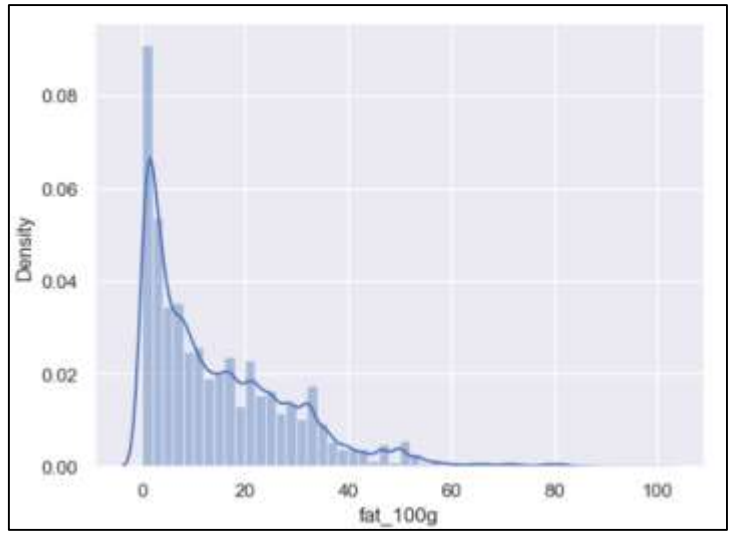

additives_n
0.0
0.5
1.0
1.5
2.0
2.5
a
b
c
d
e
nutrition_grade_fr

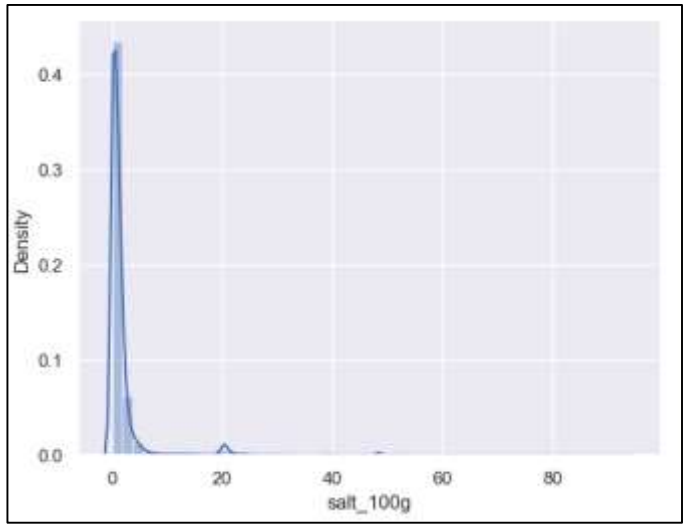

fiber_100g
0
1
2
3
4
5
a
b
c
d
e
nutrition_grade_fr

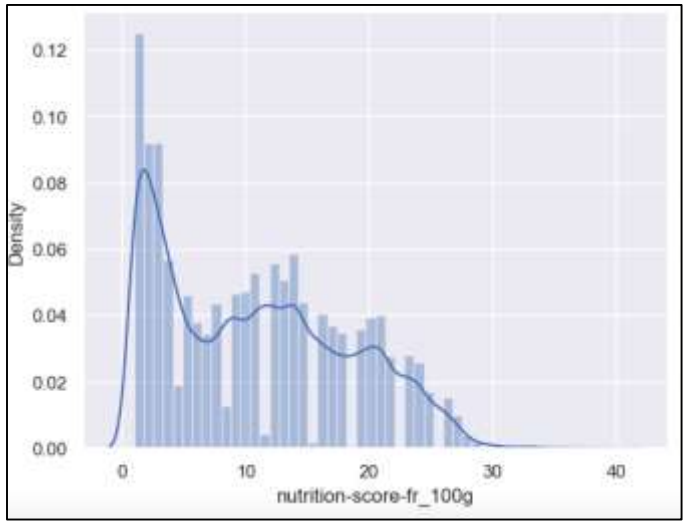
Analyse Univariée de distribution

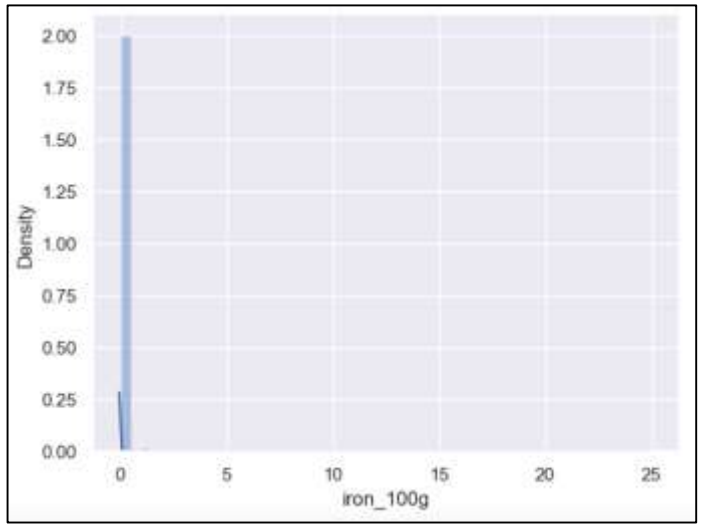




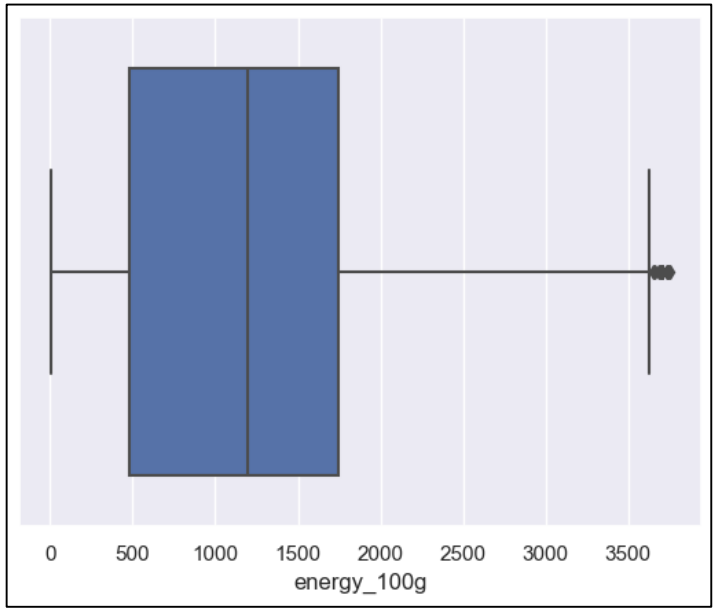


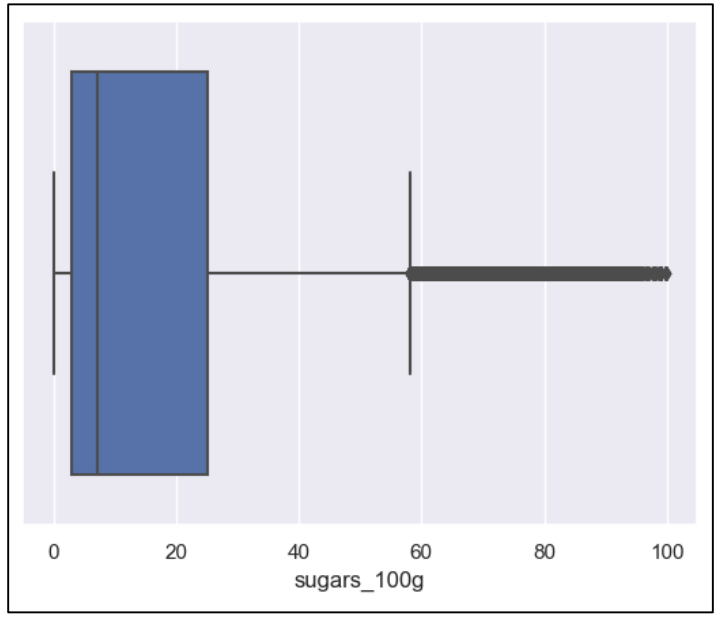


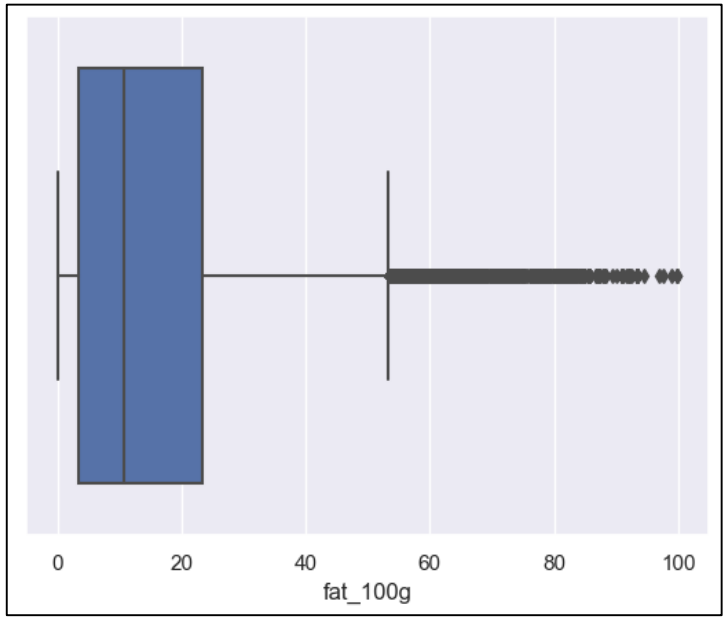


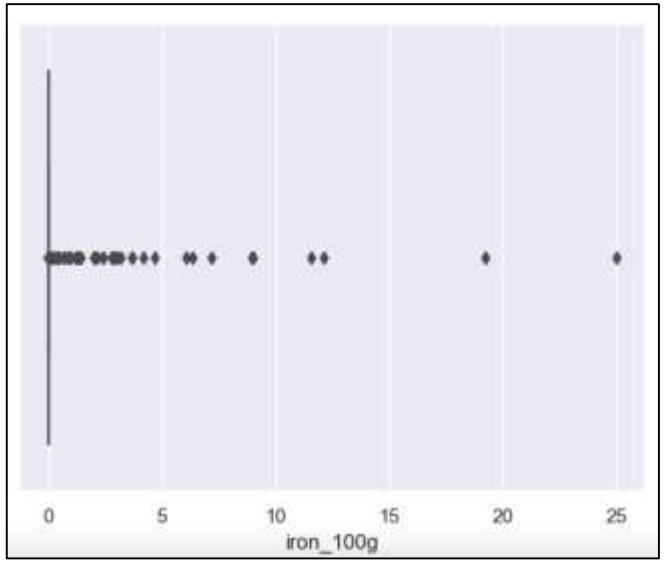


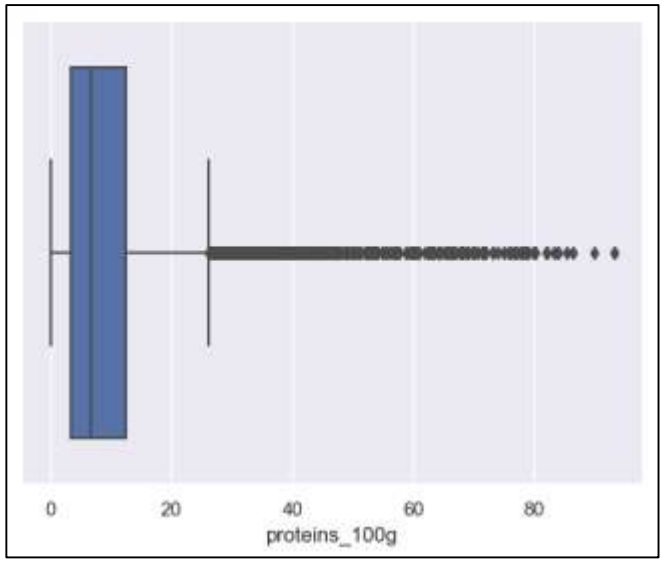
Analyse Univariée de distribution / Boxplots

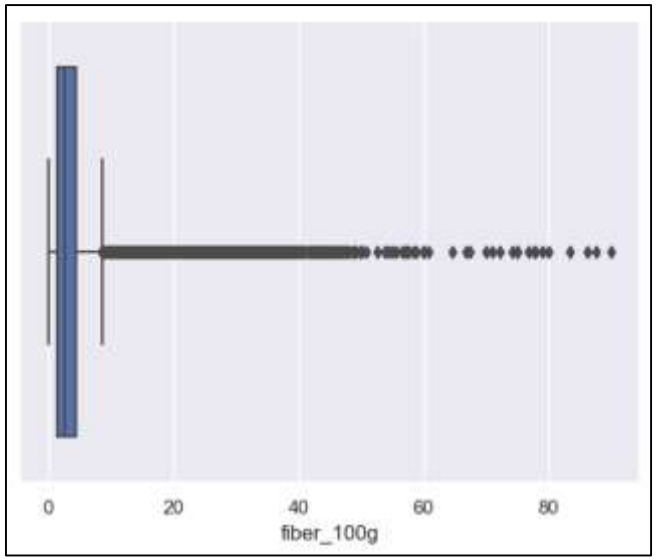




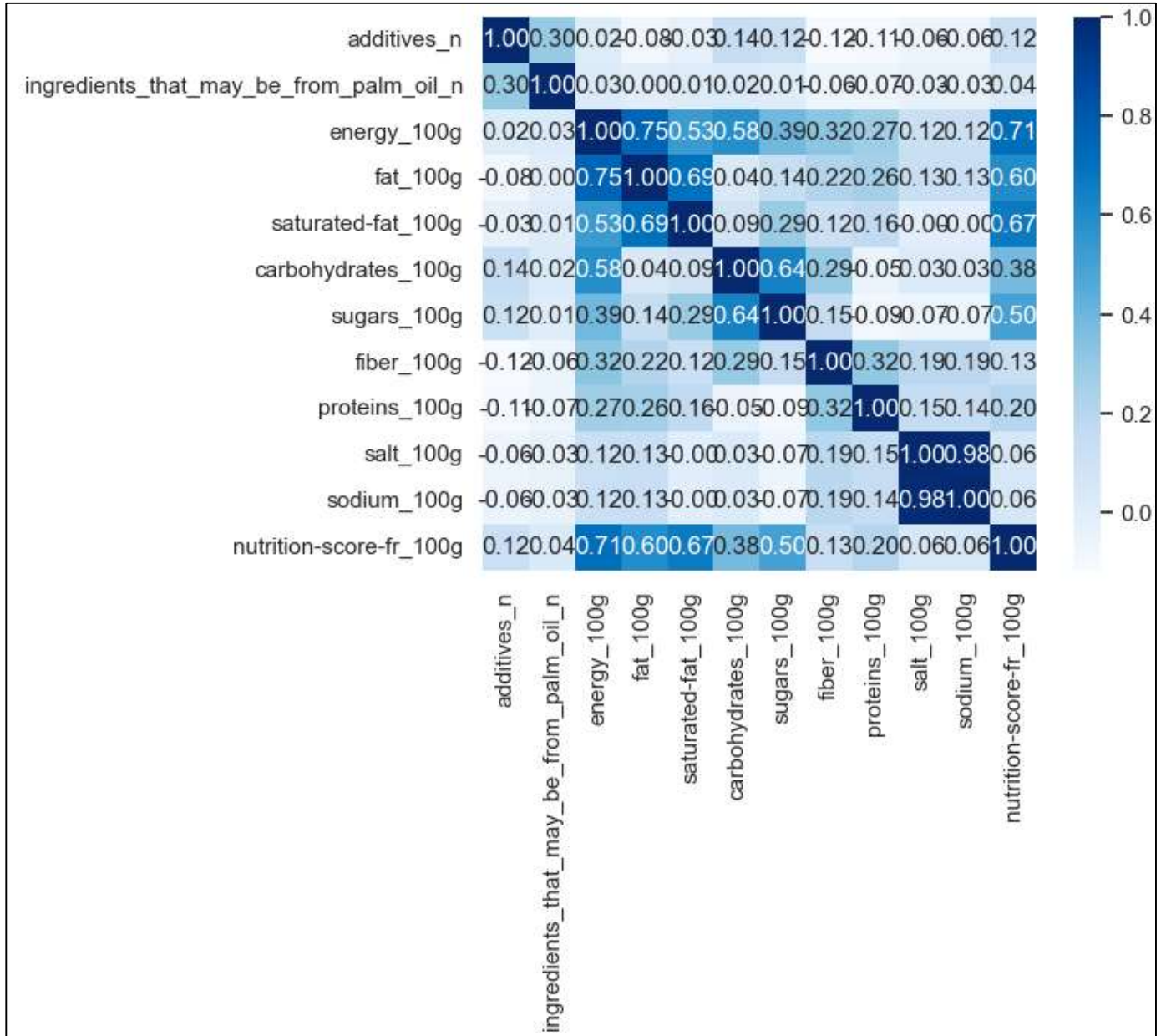








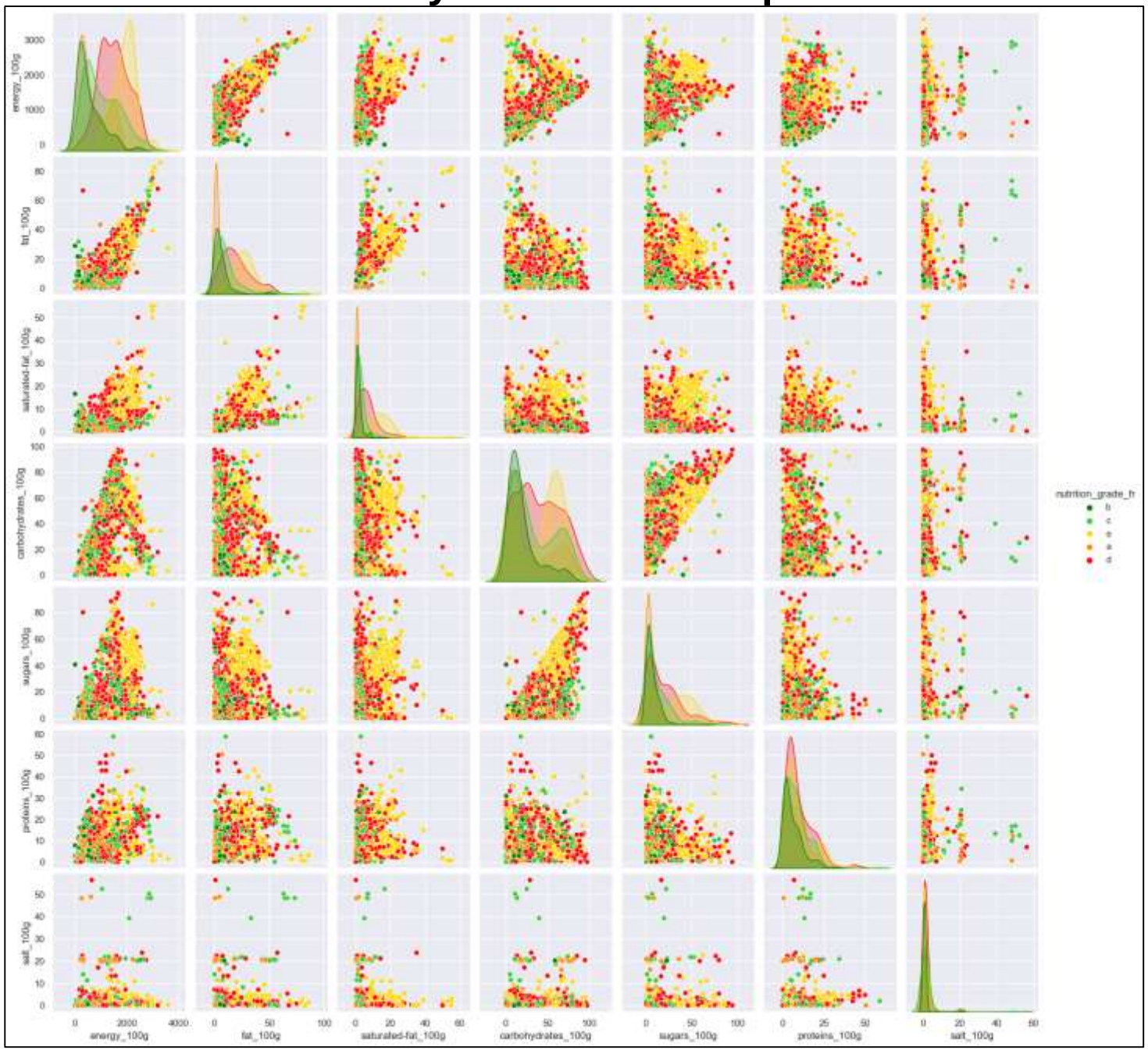
Analyse Bivarié / Matrice de corrélation



Analyse du tableau :

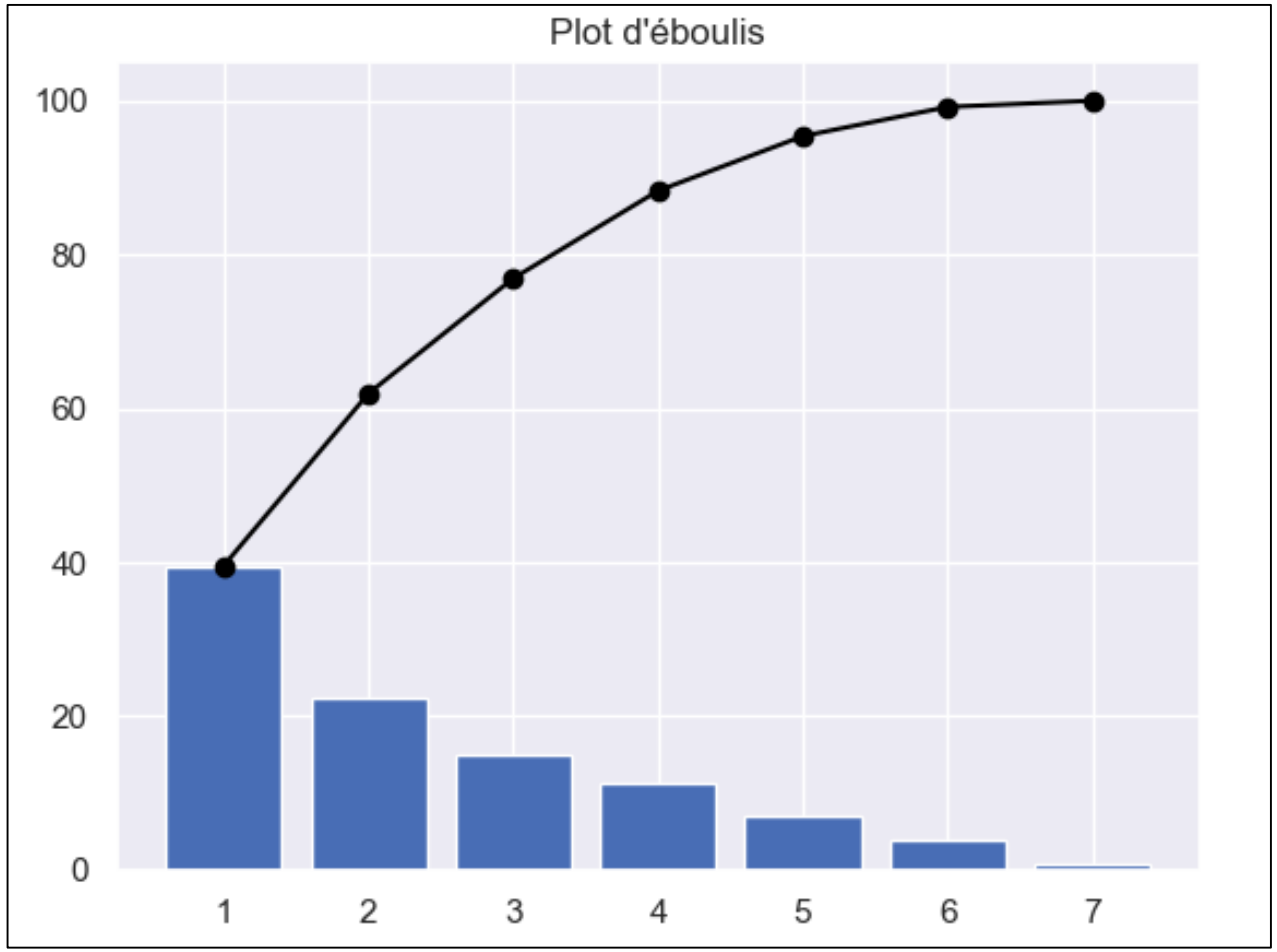
- additives_n : pas de correlation forte
- ingredients_may_be..._palm_oil : pas de correlation forte
- energy_100g : forte corrélation avec: fat_100g, saturated-fat_100g, carbohydrates_100g, nutrition-score-fr_100g
- fat_100g: fortement corrélés avec: saturated-fat_100g
- nutrition-score-fr_100g : forte corrélation avec: saturated_fat_100g, energy_100g, fat_100g
- sugars_100g : fortement corrélés avec: carbohydrates_100g
- sodium_100g: fortement corrélés avec: salt_100g

Analyse Bivarié / Pairplot



Études de l’Analyse / ACP

Méthode ACP : L'analyse en composantes principales

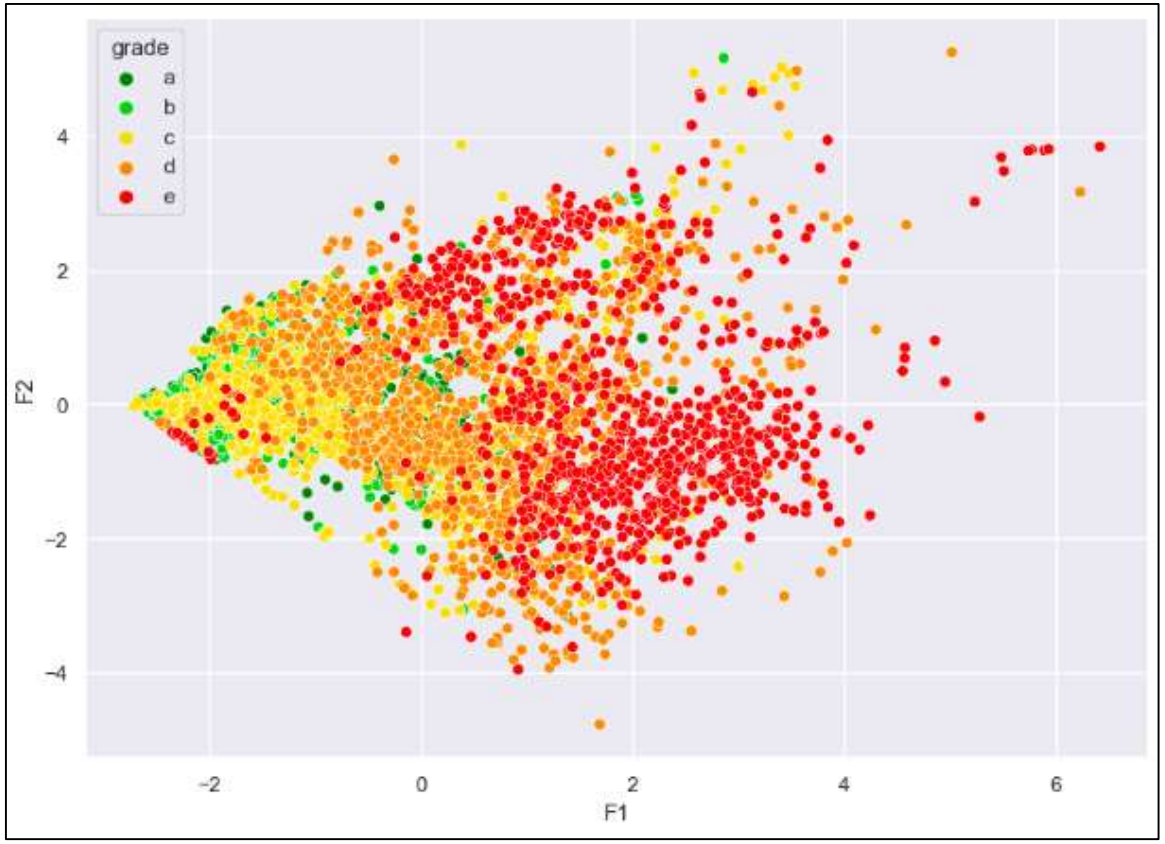


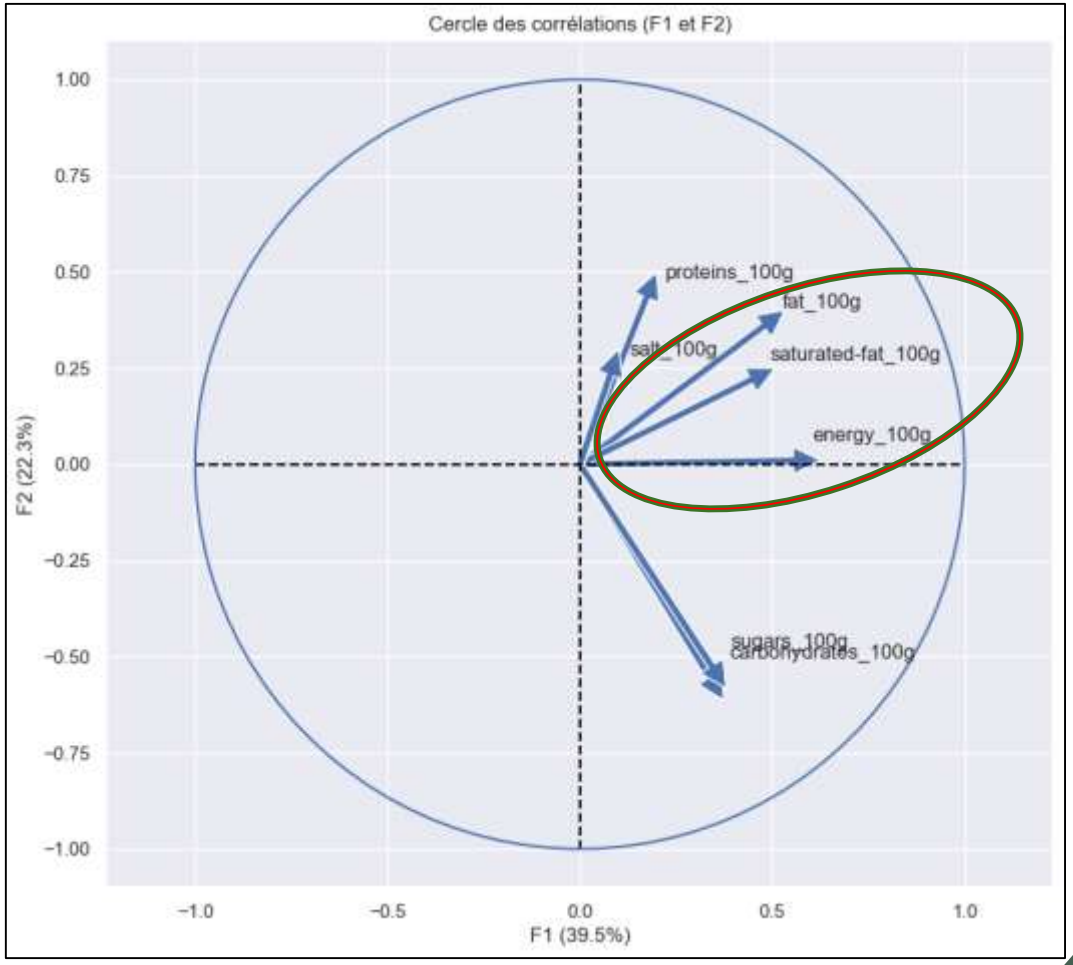
	F1	F2	F3	F4	F5	F6	F7	grade
151337	-1.924951	0.043820	-0.342626	0.225513	-0.020419	0.026985	-0.046839	b
119286	1.034917	0.969419	-0.865456	0.691196	-0.123825	-1.092270	-0.023888	d
65110	-0.601981	1.213600	-1.442624	1.274541	0.753528	0.355731	0.933226	b
173763	3.022742	-1.085063	-0.891574	0.427127	0.763448	-0.172825	-0.284309	e
86517	0.269521	-0.393035	-0.713606	0.529203	0.248662	-0.065810	-0.062370	d
...	...	...	...	...	...	...	...	...
88319	2.050901	2.313918	-0.289838	-0.635085	-1.476566	-1.430973	0.246256	c
97781	3.129252	-0.016908	-0.887918	-0.077503	0.607640	0.075432	-0.181360	e
150951	-0.350389	1.746122	-0.026074	-0.166726	0.273415	-0.068148	-0.113433	e
131876	-0.937930	1.515839	0.240570	-0.599249	0.359991	-0.125944	-0.149552	d
158803	-0.070659	1.926344	-0.371214	-0.514469	0.052794	-0.203533	-0.096356	d

91052 rows × 8 columns

Études de l'Analyse / ACP

Méthode ACP : L'analyse en composantes principales





Études de l'Analyse / ANOVA paramétriques.

- **Test de Kolmogorov-Smirnov**: sert à tester la normalité des distributions.

- **Méthode Levene:** Vérifie l'homogénéité pour les distribution normales.

Méthode Levene:

```
LeveneResult(statistic=5.232007406910179, pvalue=0.02815667801478735)
LeveneResult(statistic=1.8810221693852305, pvalue=0.1787064177717)
```

Test de Kolmogorov-Smirnov :

```
...... energy_100g
-----------> p = 0.0
...... fat_100g
-----------> p = 0.0
...... saturated-fat_100g
-----------> p = 0.0
...... trans-fat_100g
-----------> p = 0.0
...... cholesterol_100g
-----------> p = 0.0
...... carbohydrates_100g
-----------> p = 0.0
...... sugars_100g
-----------> p = 0.0
...... fiber_100g
-----------> p = 0.0
...... proteins_100g
-----------> p = 0.0
...... salt_100g
-----------> p = 0.0
```

Pvalues< 0.05 une fois de plus l'hypothèse nulle sera écartée.

On peux voir que l'hypothèse nulle est exclue pour les variables, car H0 est rejetée : Pour tout les variables car ce n'est pas des distributions normale

Études de l’Analyse / ANOVA NON-Paramétrique

- **Méthode Kruskall-Wallis :** C'est une version non paramétrique de l'Anova.

Il teste l'hypothèse nulle selon laquelle la médiane de la population de tous les groupes est égale.

Méthode Kruskall-Wallis :

```
----------------------------------------------------------
energy_100g
H = KruskalResult(statistic=74159.85002895546, pvalue=0.0)
fat_100g
H = KruskalResult(statistic=80814.05231630483, pvalue=0.0)
saturated-fat_100g
H = KruskalResult(statistic=95177.93258567923, pvalue=0.0)
carbohydrates_100g
H = KruskalResult(statistic=14721.995624240564, pvalue=0.0)
sugars_100g
H = KruskalResult(statistic=35843.59127816292, pvalue=0.0)
proteins_100g
H = KruskalResult(statistic=8422.424155741985, pvalue=0.0)
salt_100g
H = KruskalResult(statistic=16784.279605020198, pvalue=0.0)
----------------------------------------------------------

Test de Kruskal-Wallis pour energy_100g
Grade A et grade B : H= 1.0465523574004937e-272
Grade A et grade C : H= 0.0
Grade A et grade D : H= 0.0
Grade A et grade E : H= 0.0
Grade B et grade C : H= 0.0
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 0.0
Grade C et grade E : H= 0.0
Grade D et grade E : H= 0.0

Test de Kruskal-Wallis pour fat_100g
Grade A et grade B : H= 4.5634230004307923e-240
Grade A et grade C : H= 0.0
Grade A et grade D : H= 0.0
Grade A et grade E : H= 0.0
Grade B et grade C : H= 0.0
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 0.0
Grade C et grade E : H= 0.0
Grade D et grade E : H= 0.0
```

```
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 0.0
Grade C et grade E : H= 0.0
Grade D et grade E : H= 5.3177954681129696e-45

Test de Kruskal-Wallis pour sugars_100g
Grade A et grade B : H= 2.654232675523432e-196
Grade A et grade C : H= 0.0
Grade A et grade D : H= 0.0
Grade A et grade E : H= 0.0
Grade B et grade C : H= 0.0
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 0.0
Grade C et grade E : H= 0.0
Grade D et grade E : H= 0.0

Test de Kruskal-Wallis pour proteins_100g
Grade A et grade B : H= 0.0
Grade A et grade C : H= 0.2976411229387067
Grade A et grade D : H= 3.636311510856052e-246
Grade A et grade E : H= 2.7208398310226466e-79
Grade B et grade C : H= 0.0
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 6.211298636524605e-278
Grade C et grade E : H= 1.0393892304750784e-92
Grade D et grade E : H= 2.832550607877299e-43

Test de Kruskal-Wallis pour salt_100g
Grade A et grade B : H= 5.542659007945473e-166
Grade A et grade C : H= 0.0
Grade A et grade D : H= 0.0
Grade A et grade E : H= 0.0
Grade B et grade C : H= 0.0
Grade B et grade D : H= 0.0
Grade B et grade E : H= 0.0
Grade C et grade D : H= 1.4342598352418783e-76
Grade C et grade E : H= 1.2654757198800974e-39
Grade D et grade E : H= 1.4880704236431443e-05
```

•Dans ces résultats, les estimations des médianes pour les échantillons des 7 groupes. L'hypothèse nulle veut que les médianes de population de ces groupes soient toutes égales. Les deux valeurs de p étant inférieures au seuil de signification de 0,05.On peux rejeter l'hypothèse nulle et conclure que les médianes ne sont pas égales.

•La p-value nous indique que la probabilité de rejeter l'hypothèse nulle alors qu'elle serait vraie est inférieure à 0.05. Dans ce cas, on peut rejeter en toute confiance l'hypothèse nulle

•La p-value du test étant en majorité égal à 0. On peut donc rejeter l'hypothèse nulle (selon laquelle les données proviennent d'une distribution normale)

4 – Synthèse

Donc pour conclure à la fin de notre nettoyage de données et d’avoir fait des Analyses statistiques, avec cette base de données. On peut bel est bien crée l’application, qui consistes pour rappel, à savoir le grade moyen des produits achetées.

Par exemple : si j’ai acheté 10 produits, j’aurais un niveau de nutri-grades moyen (de A à E) pour les 10 produits achetés, est cela indiquera si oui ou non cela est bon pour notre corps.

Merci