

Alexandre ROGUES

-

Parcours Data Scientist
Projet 3

Préparez des données pour un organisme de santé publique

-

Mars 2024



ANALYSE DATASET



open **FOOD** facts

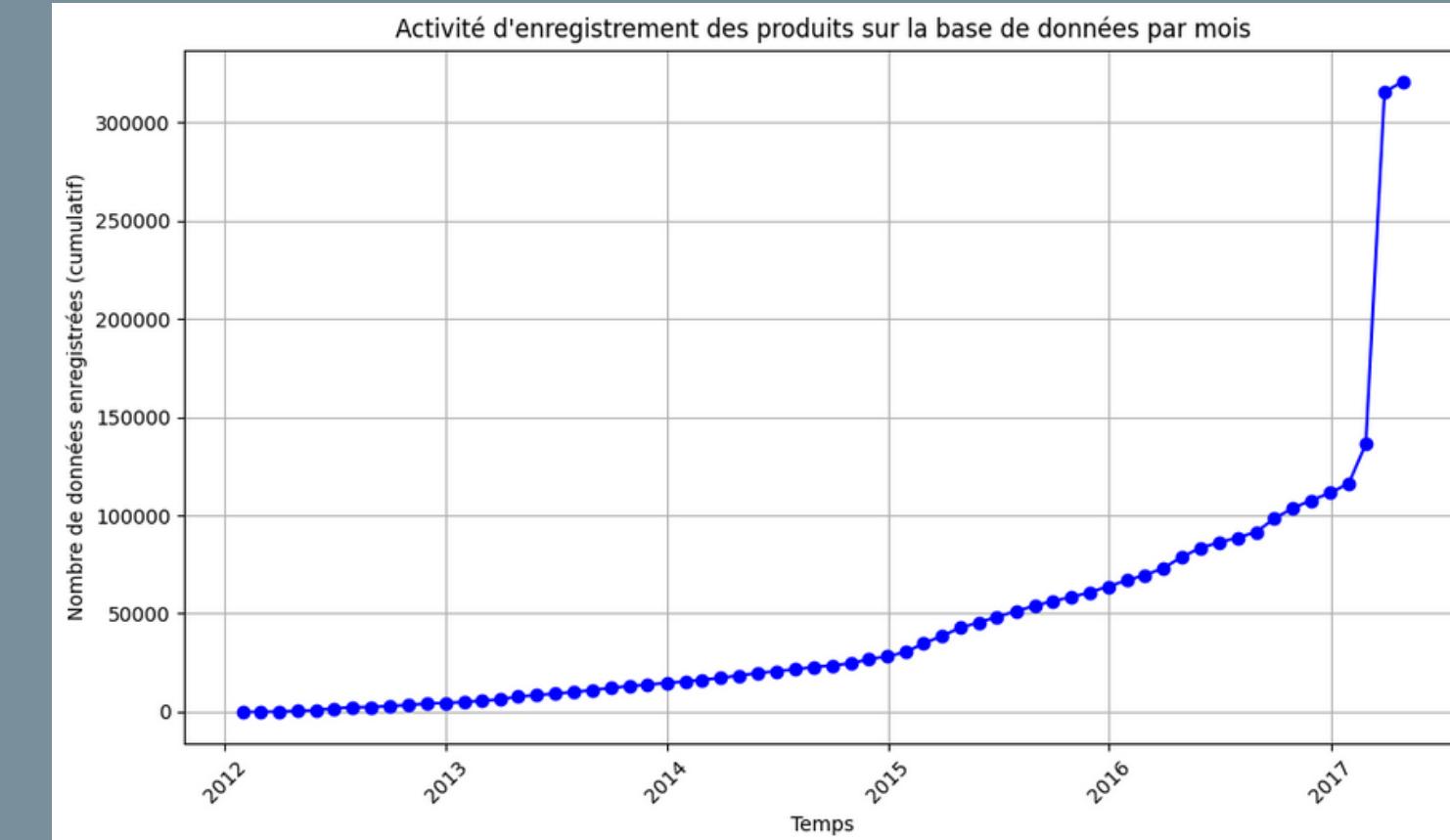
A PROPOS



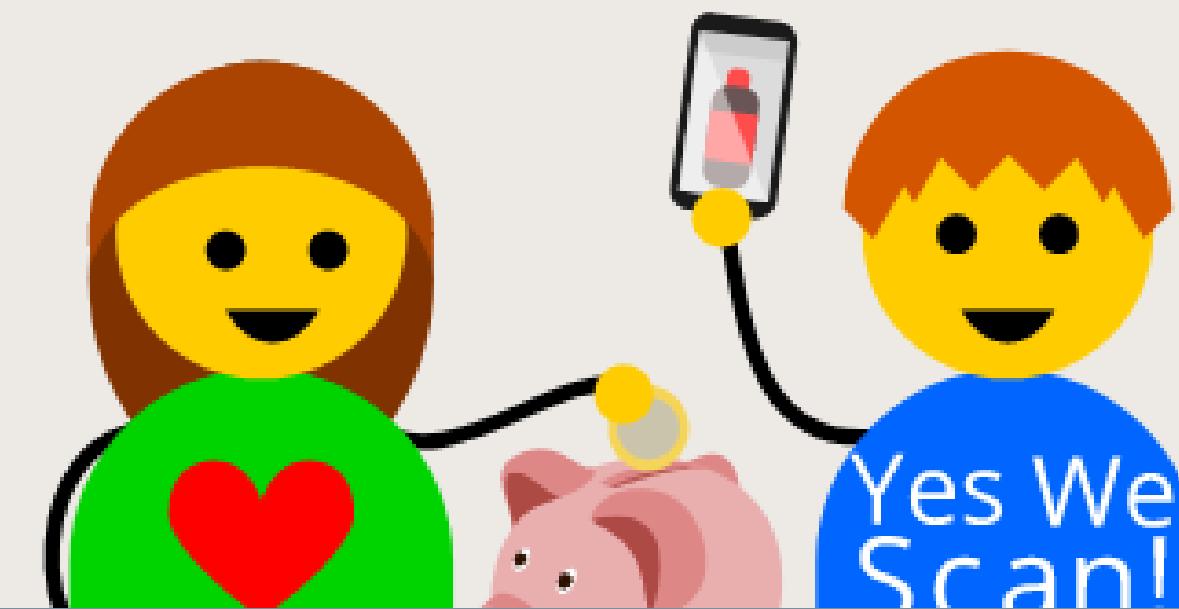
Organisation à but non lucratif, association Française loi 1901, créée en 2012

Objectif : informer les consommateurs du monde entier sur ce qu'ils mangent.

- Principe collaboratif
- Contributeurs particuliers et professionnels
- Site internet et appli mobile



PLUS D'INFOS



- Base de produits alimentaires, faite par tout le monde, pour tout le monde
- Décoder les étiquettes, trouver des produits, comparer et changer, explorer et découvrir
- Alimenter la réflexion, l'innovation et la recherche
- Projet citoyen, collaboratif et indépendant



- 170 000 contributeurs
- 600 000 produits

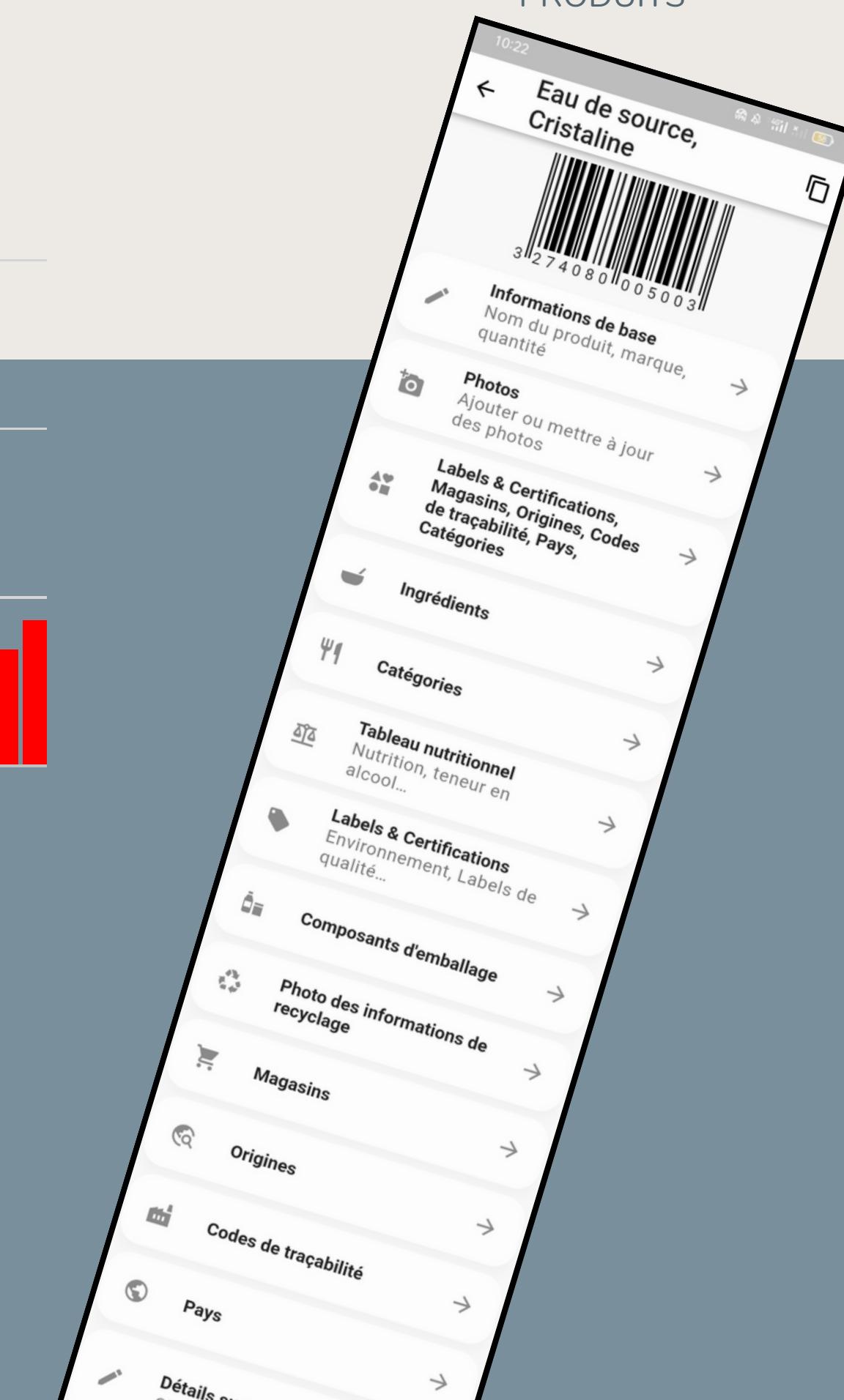
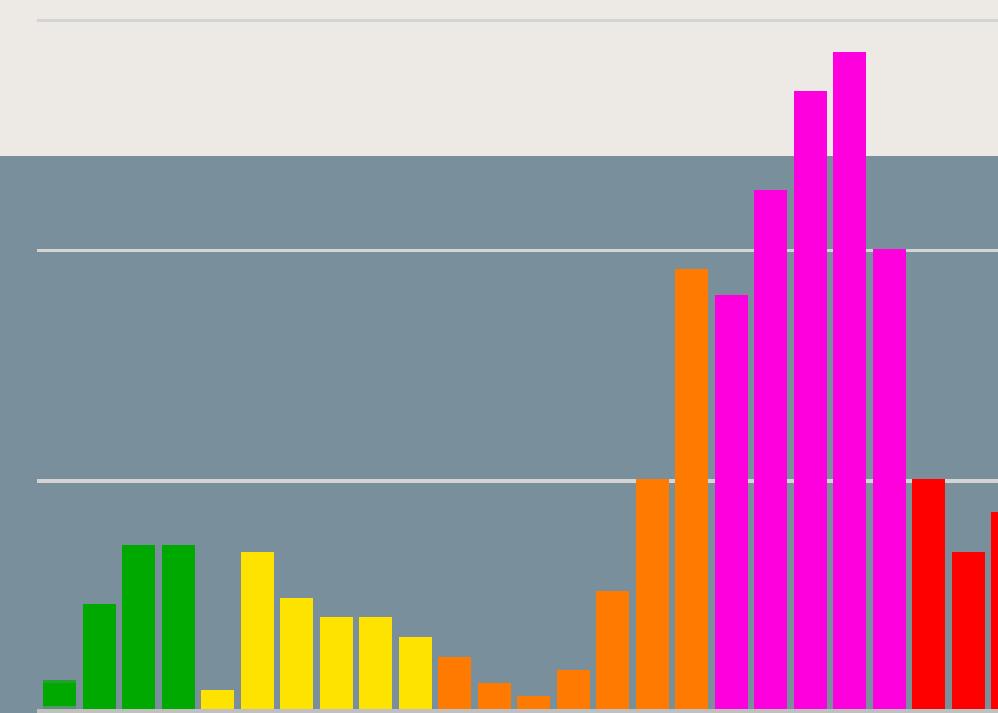
source : <https://www.openfoodfacts.org>

PROBLEMATIQUE

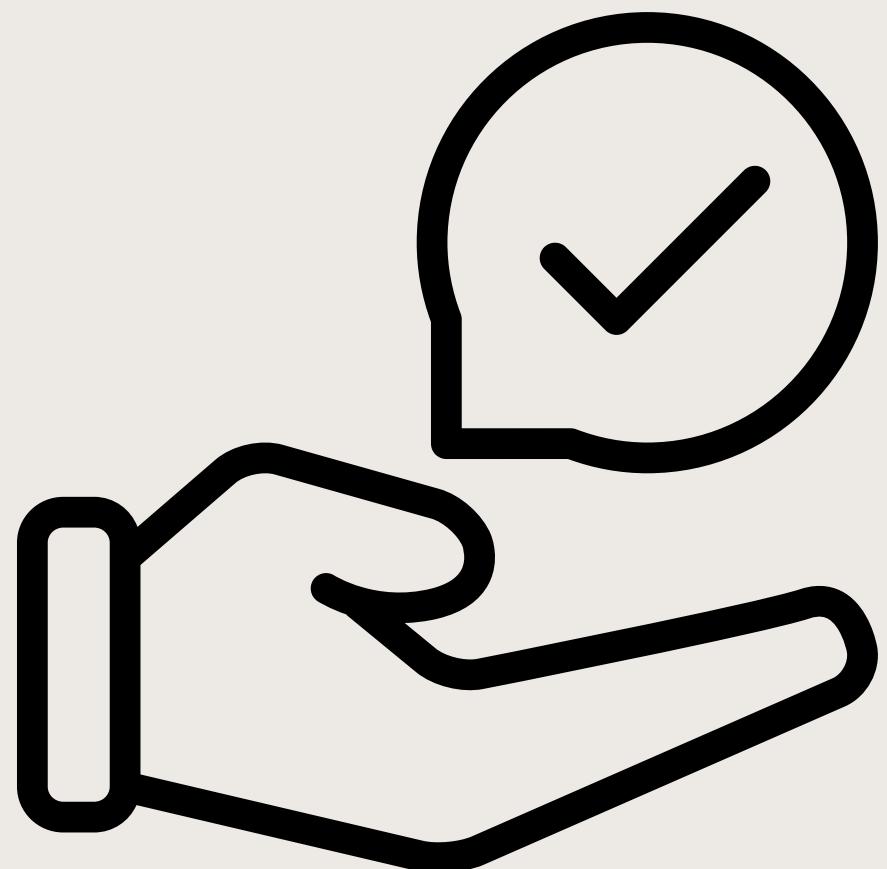
1 produit



162 champs à remplir



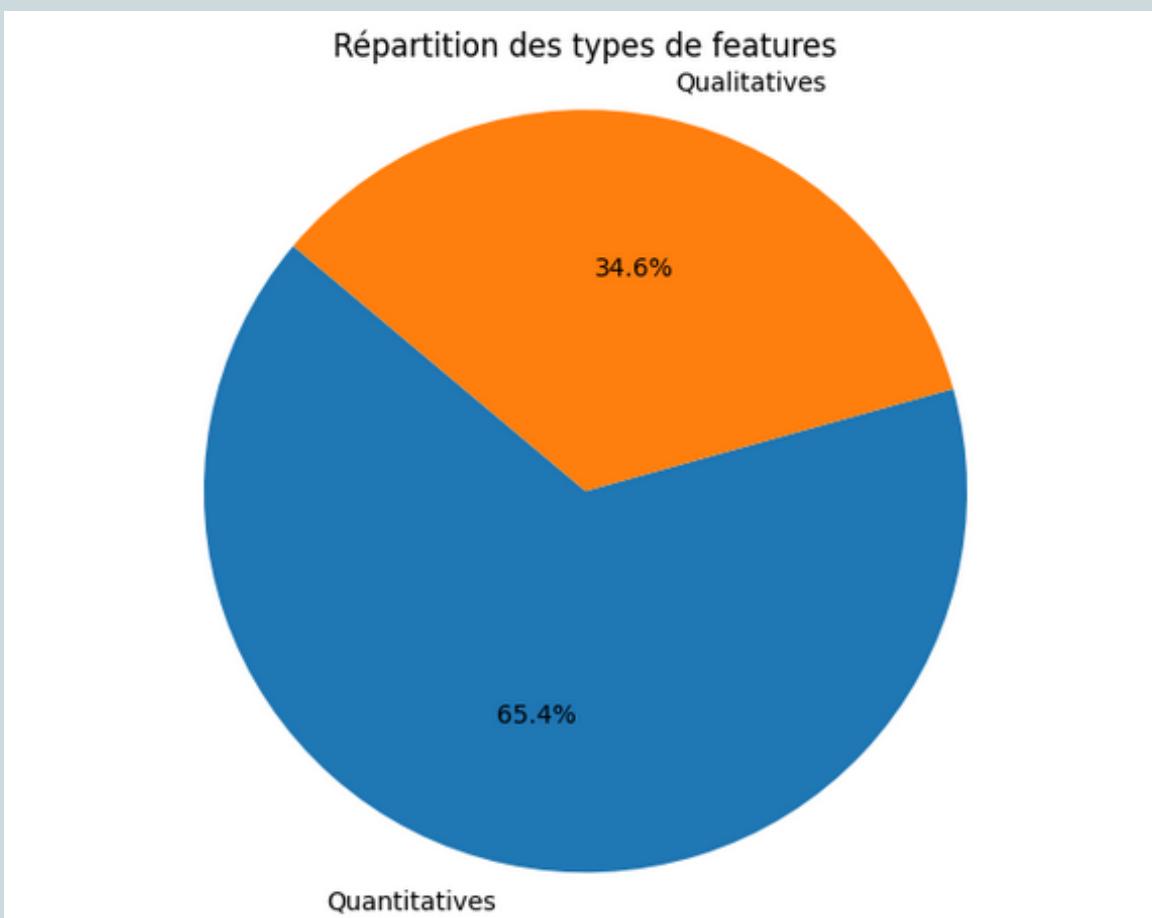
FACILITER LA CONTRIBUTION



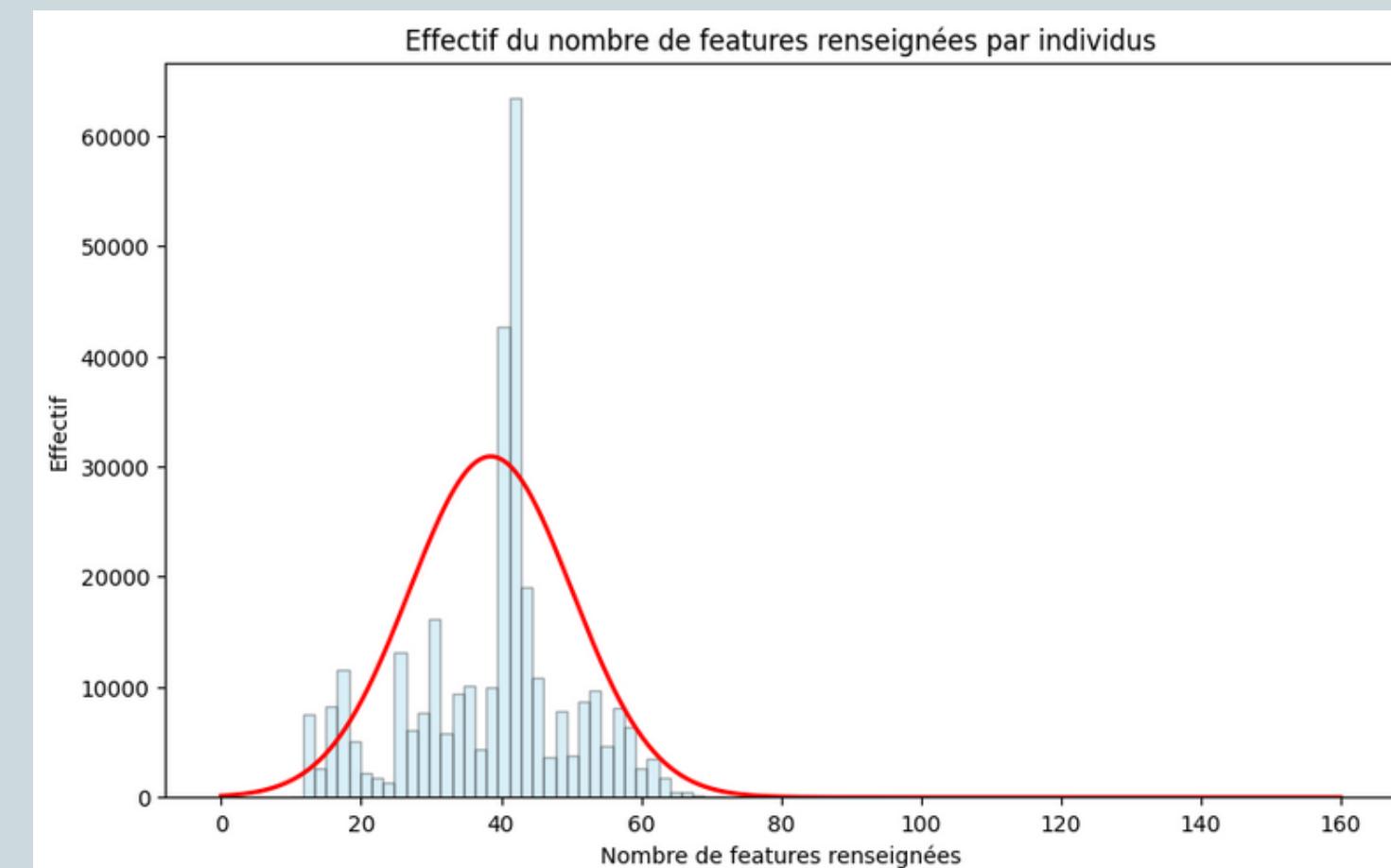
Question : est-il possible de suggérer les valeurs manquantes pour une variable dont plus de 50% des valeurs sont manquantes

NETTOYAGE DE DONNEES

Identification des features



Analyse du remplissage



NETTOYAGE DE DONNEES

Etude Nutriscore

Le calcul du nutriscore est basé sur plusieurs critères

- La catégorie du produit (colonnes)
- La présence d'un composant et sa quantité (lignes)

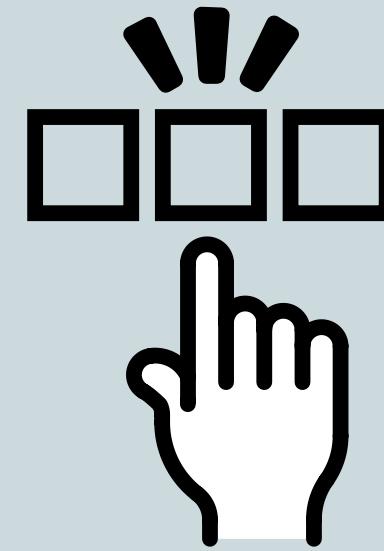
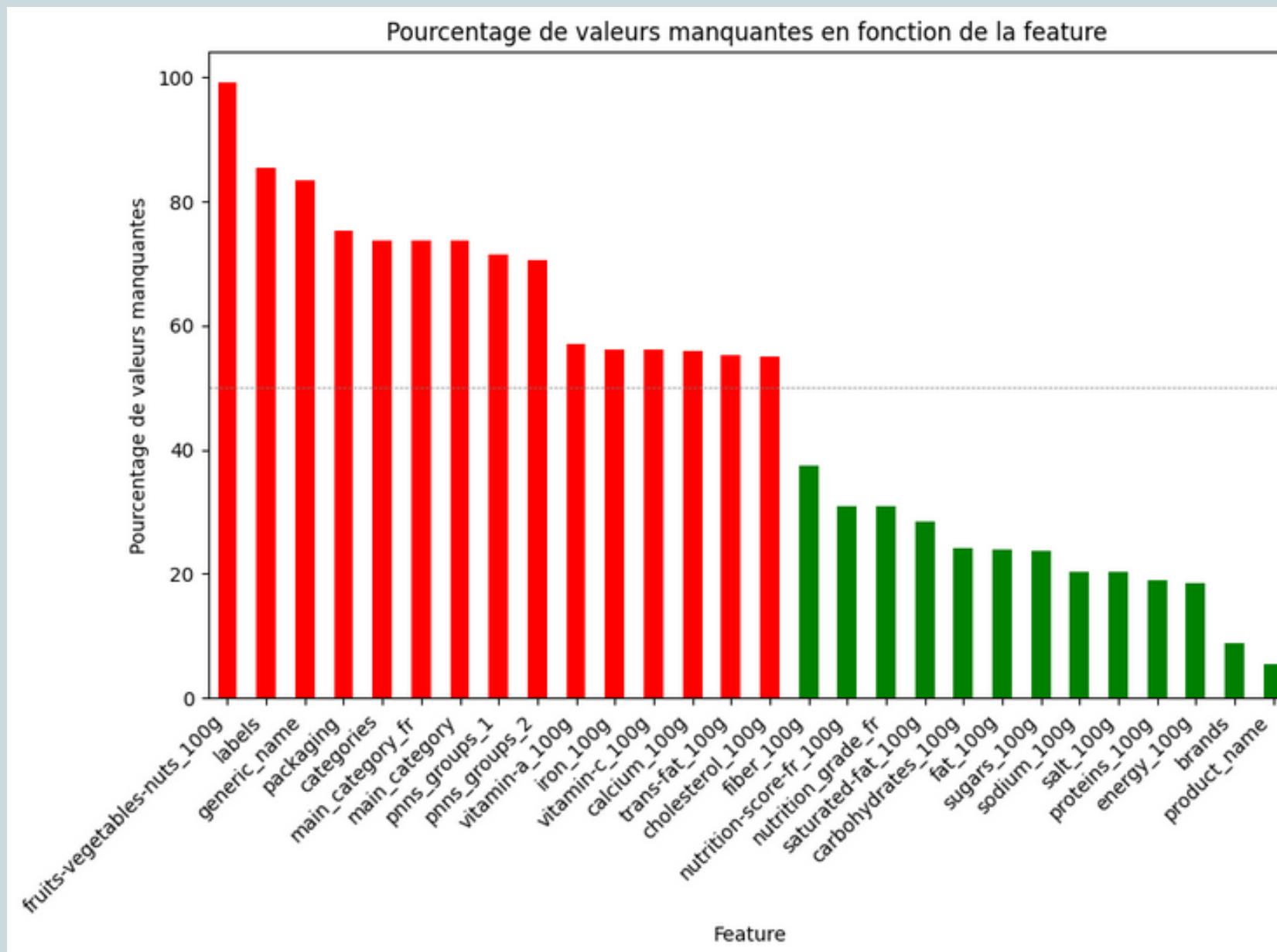
	Cas général	Viande rouge	Fromage	Mat grasse,fr à coque,graines	Boissons
Energie (kJ/100g)	-1	-1	-1	-1	-1
Sucres (g/100 g)	-1	-1	-1	-1	-1
Acides gras saturés (g/100 g)	-1	-1	-1	-1	-1
Sel (g/100 g)	-1	-1	-1	-1	-1
Présence d'édulcorant non-nutritif (OUI/NON)	0	0	0	0	-1
Fruits, légumes et légumes secs (%)	+1	+1	+1	0	+1
Fibres (g/100 g)	+1	+1	+1	+1	+1
Protéines (g/100 g)	+1	+1	+1	+1	+1
Huiles dérivées des fruits, légumes and légumes secs (%)	0	0	0	+1	0
Eau (sans aucune addition)	0	0	0	0	+1
Matière grasse (g/100 g)	0	0	0	-1	0

légende

- +1 indique que le composant est pris en compte positivement dans le calcul du nutriscore
- 0 indique que le composant n'est pas pris en compte dans le calcul du nutriscore
- -1 indique que le composant est pris en compte négativement dans le calcul du nutriscore

NETTOYAGE DE DONNEES

Présélection de features pertinentes



Features qualitatives

- product_name
- brands
- pnns_groups_1
- nutrition_grade_fr

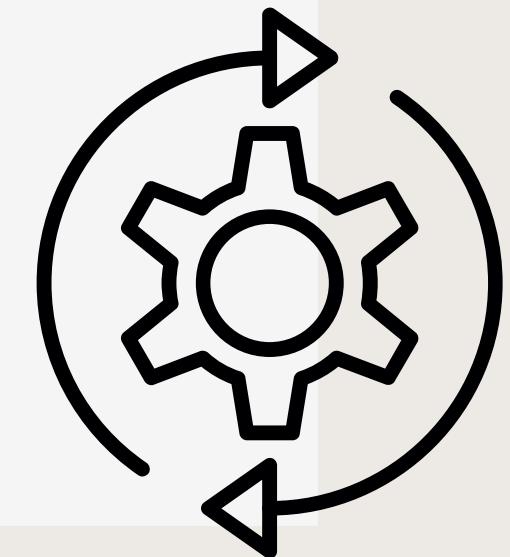
Features quantitatives

- energy_100g
- proteins_100g
- salt_100g
- sodium_100g
- sugars_100g
- carbohydrates_100g
- fat_100g
- saturated-fat_100g
- fiber_100g
- nutrition-score-fr_100g

AUTOMATISATION

Création d'une fonction de traitement automatique

```
#automatisation de la création du df de travail :  
  
#définition d'une fonction filtre qui prends en argument notre dataframe  
def filtre1(dataframe):  
  
    #liste des colonnes intéressantes  
    colonnes_interessantes = [  
  
        # valeurs qualitatives  
  
        'product_name', #name of the product  
        'brands',  
        'pnns_groups_1',  
        'nutrition_grade_fr',  
  
        # valeurs quantitatives  
  
        'energy_100g',  
        'proteins_100g',
```



1

Sélection des colonnes intéressantes

2

Suppression des colonnes sans product_name

3

Suppression des lignes sans valeurs renseignées

4

Suppression des doublons

VALEURS ABERRANTES

▼ 2 - Identification et traitement des valeurs aberrantes

Analyse colonne energy_100g

Analyse colonne proteins_100g

Analyse colonne salt_100g et sodium_100g

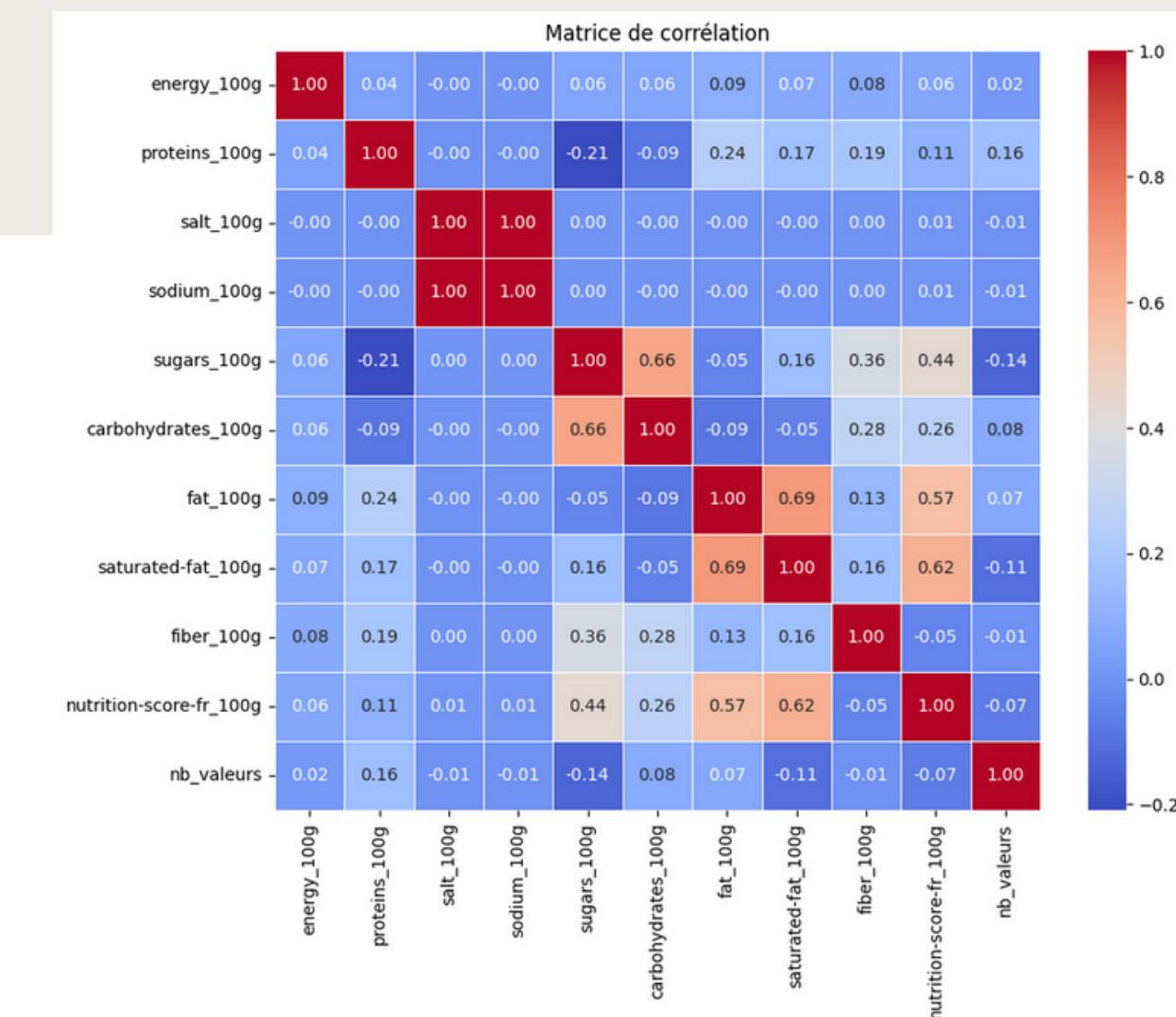
► Analyse colonne sugars_100g et carbohydrates_100g et fiber_100g

► Analyse colonne fat_100g et trans-fat_100g

Analyse conjointe des quantités de nutriments fondamentaux

Récapitulatif critères

Création d'une fonction de traitement automatique des valeurs aberrantes



```
# Expressions régulières pour détecter le mot "huile" dans différentes langues
motif_huile = r'huile|oil|aceite|olio|Öl|óleo|olie|масло|油|abura'

# Filtrer les lignes contenant les mots "huile" ou "oil" ou leurs traductions dans la colonne 'product_name'
filtre_huile = valeurs_aberrantes_energy['product_name'].str.contains(motif_huile, case=False, regex=True)

# Calculer le pourcentage d'individus contenant les mots "huile" ou "oil" ou leurs traductions
pourcentage_huile = (filtre_huile.sum() / len(valeurs_aberrantes_energy)) * 100

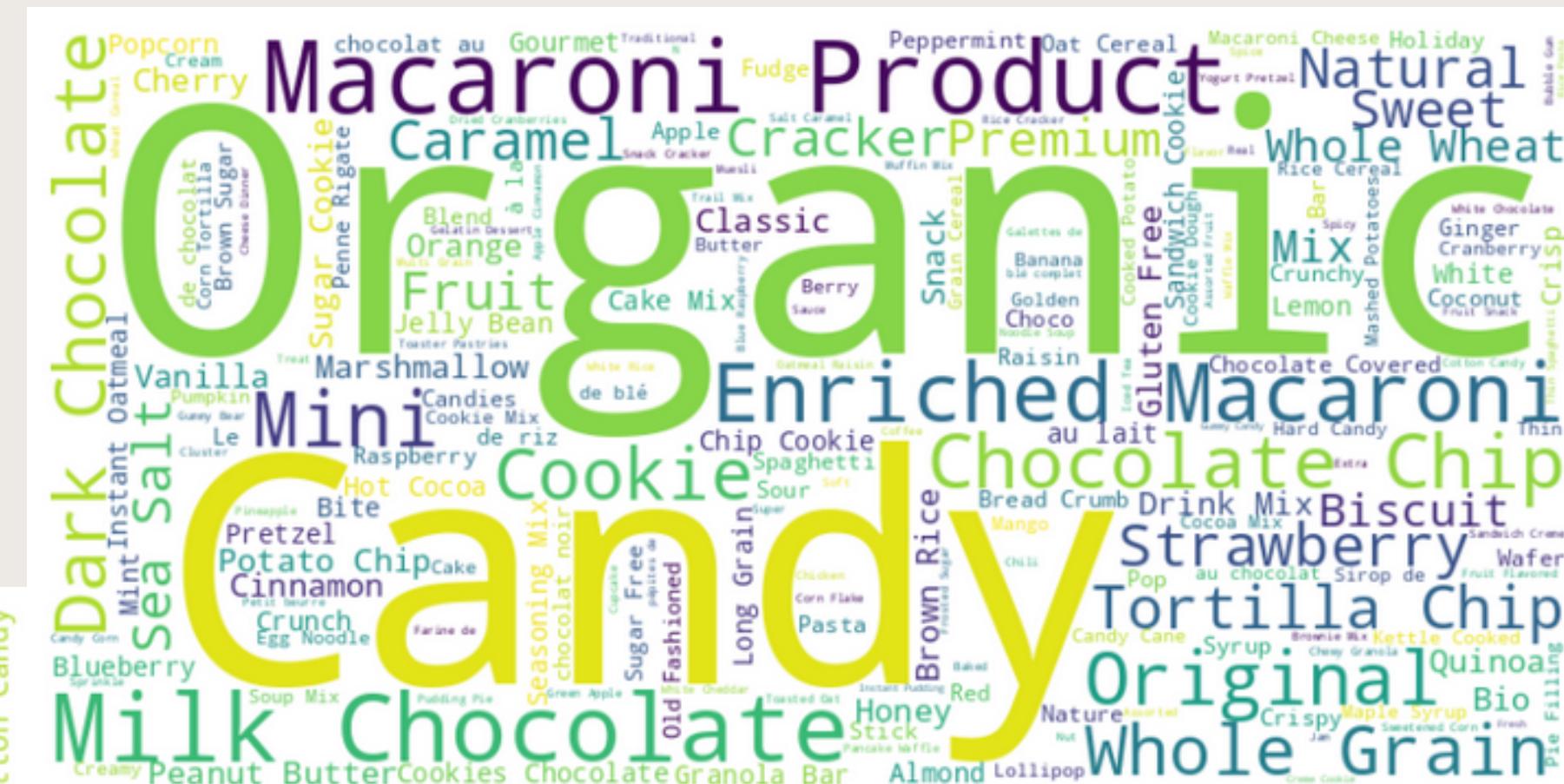
print(f"Le pourcentage d'individus contenant les mots 'huile' ou 'oil' avec une énergie en dehors des limites est de : {pourcentage_huile:.2f}%")
```

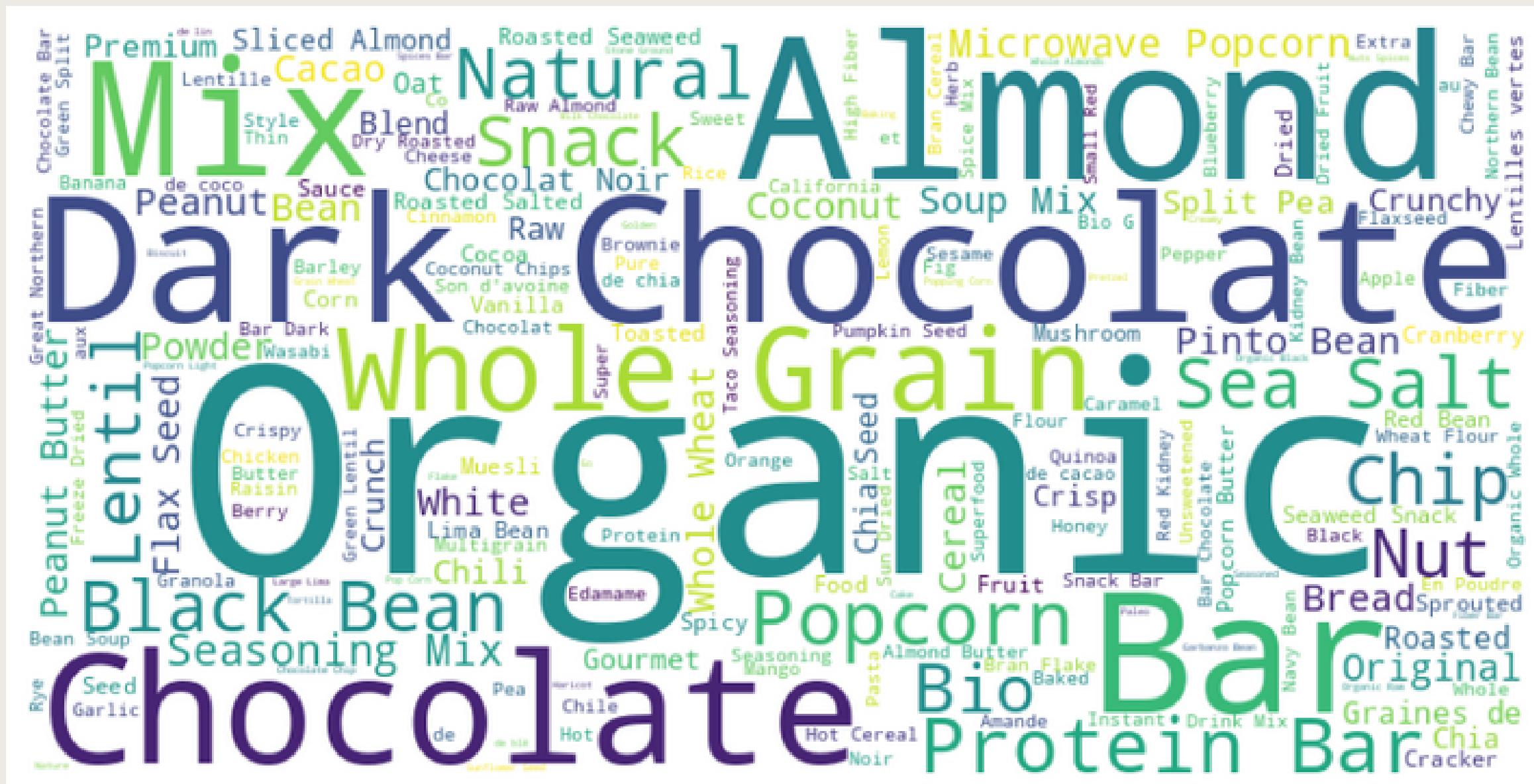
Le pourcentage d'individus contenant les mots 'huile' ou 'oil' avec une énergie en dehors des limites est de : 68.11%

L'huile est par définition composé de lipides, donc très énergétique. Ces valeurs sont à conserver.

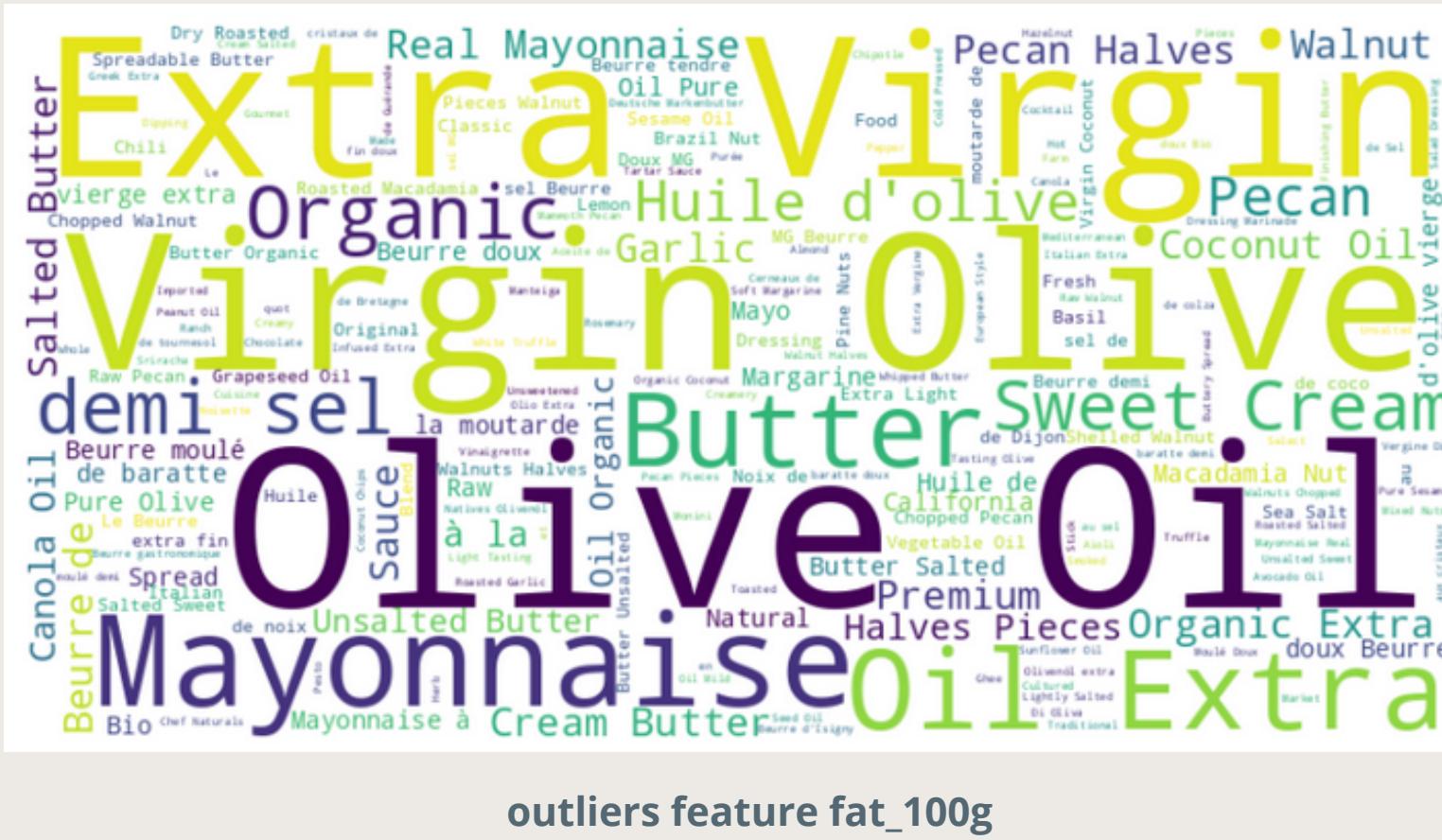


Nuage de mots des individus possédant plus de 25% de sel hors individus possédant le mot sel (ou ses traductions) dans son nom

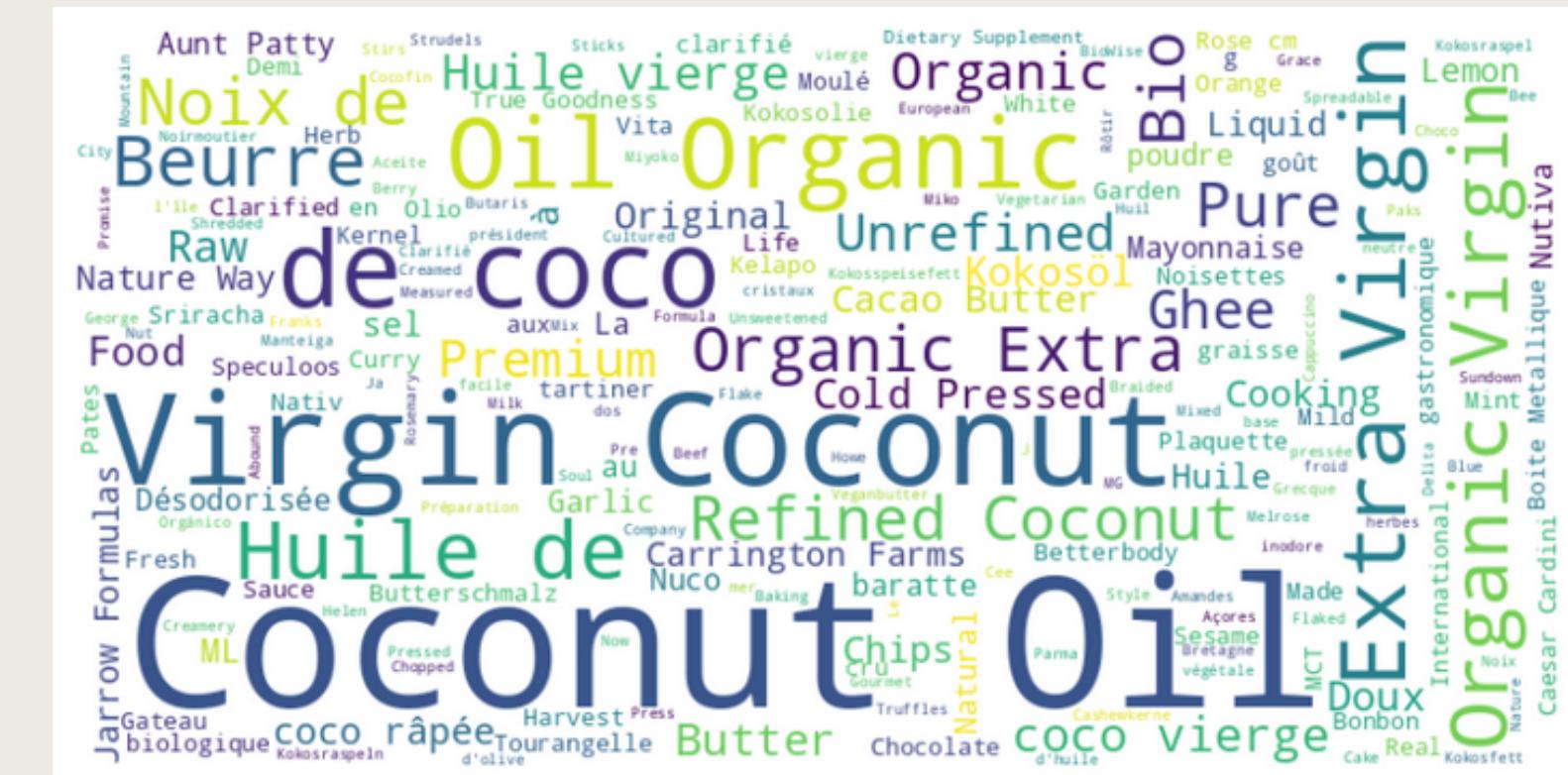




outliers feature fibers_100g



outliers feature fat_100g



outliers feature saturated_fat_100g

Critère feature unique :

$0 \leq \text{energy_100g} \leq 4000$

Critère sur feature unique identique :

$0 \leq \text{proteins_100g} \leq 100$

$0 \leq \text{salt_100g} \leq 100$

$0 \leq \text{sodium_100g} \leq 40$

$0 \leq \text{sugars_100g} \leq 100$

$0 \leq \text{carbohydrates_100g} \leq 100$

$0 \leq \text{fiber_100g} \leq 100$

$0 \leq \text{fat_100g} \leq 100$

$0 \leq \text{saturated-fat_100g} \leq 100$

```
def filtre2(dataframe):
    df_work = dataframe.copy()
    df_work = filtre1(df_work)
    #filtrage feature energy 100g (sans supprimer les nan)
    df_work = df_work[(df_work['energy_100g'].isna()) | (df_work['energy_100g'] <= 4000)]
    #filtrage >0 & <100 (suppression colonnes)
    columns_to_check = ['proteins_100g', 'salt_100g', 'sugars_100g', 'carbohydrates_100g', 'fiber_100g', 'fat_100g', 'saturated-fat_100g']
    for colonne in columns_to_check:
        df_work = df_work[(df_work[colonne].isna()) | ((df_work[colonne] >= 0) & (df_work[colonne] <= 100))]

    # filtrage features croisées
    # conservation des lignes où le sodium est inf à 45% salt
    df_work = df_work[(df_work['sodium_100g'].isna()) | (df_work['salt_100g'].isna()) | (df_work['sodium_100g'] <= 0.45 * df_work['salt_100g'])]
    #conservation lignes où le sucre est inf aux carbohydrate (glucides)
    df_work = df_work[(df_work['sugars_100g'].isna()) | (df_work['carbohydrates_100g'].isna()) | (df_work['sugars_100g'] <= df_work['carbohydrates_100g'])]

    # Calculer la somme des valeurs dans les colonnes energy_100g, proteins_100g et carbohydrates_100g
    sum_values = df_work[['fat_100g', 'proteins_100g', 'carbohydrates_100g']].sum(axis=1)

    # Filtrer les lignes où la somme est inférieure ou égale à 100 (en conservant les nans)
    df_work = df_work[(sum_values.isna()) | (sum_values <= 100)]

    return df_work
```

Critere features croisées :

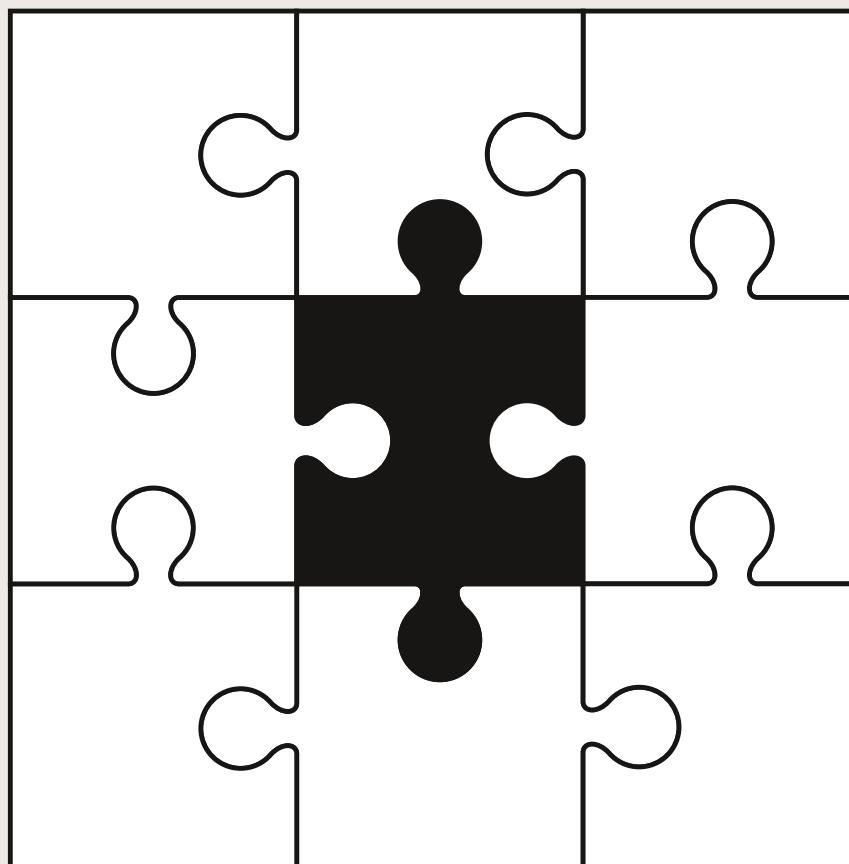
- $\text{sodium_100g} < 45\%(\text{salt_100g})$

- $\text{sugars_100g} < \text{carbohydrates_100g}$

- $\text{fat_100g} + \text{proteins_100g} + \text{carbohydrate_100g} \leq 100$

VALEURS MANQUANTES

Que faire des individus pour lesquels des features ne sont pas renseignées



Valeur constante

Valeur nulle, moyenne, médiane

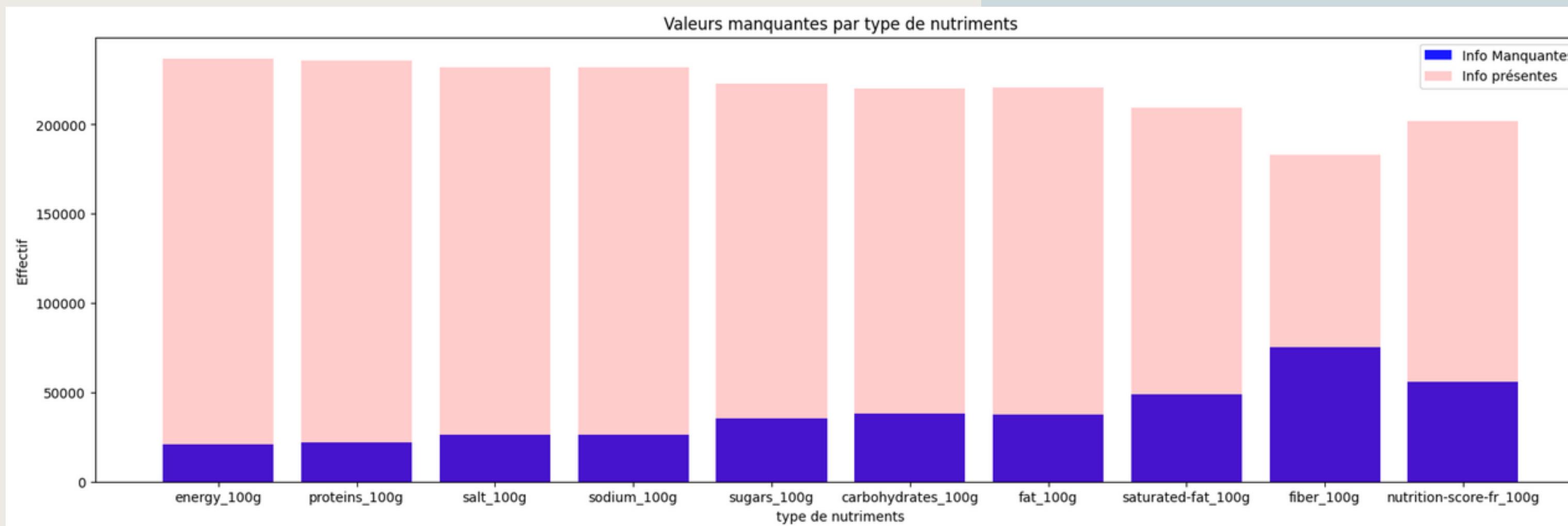
Iterative Imputing

Remplissage dynamique des valeurs manquantes par modèles statistiques

KNN

Algorithme de classification utilisant la proximité des voisins

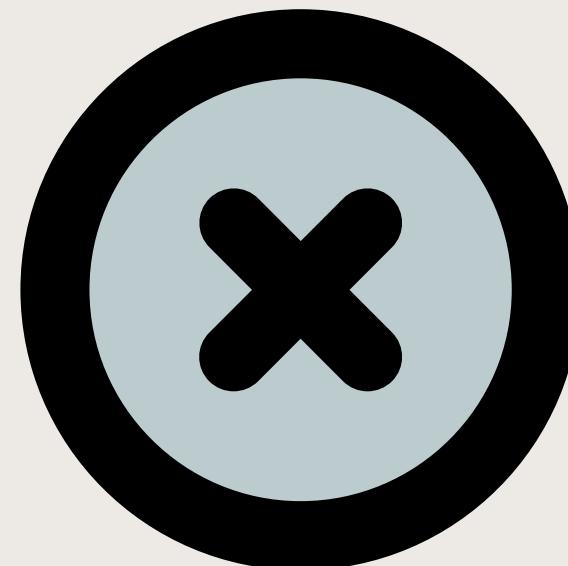
VALEURS MANQUANTES



VALEURS CONSTANTES



Simple à implémenter
Ne nécessite pas de calcul complexe
Pratique lorsque les valeurs manquantes sont aléatoires et peu nombreuses



Peut introduire des biais dans les données
Ne tient pas compte des relations entre les variables

VALEURS NULLES

	energy_100g	proteins_100g	salt_100g	sodium_100g	sugars_100g	carbohydrates_100g	fat_100g	saturated-fat_100g	fiber_100g	nutrition-score-fr_100g	nb_valeurs
count	237032.000000	235983.000000	232047.000000	232008.000000	222826.000000	220150.000000	220470.000000	209207.000000	183064.000000	202033.000000	258053.000000
mean	1118.079247	7.078799	1.605682	0.632263	15.733676	32.025061	12.537453	5.083836	2.823858	9.074013	8.505307
std	786.870195	8.073821	6.284254	2.474302	20.928711	29.045646	17.188183	7.903319	4.582908	9.041357	2.801647
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-15.000000	0.000000
25%	382.775000	0.700000	0.063500	0.025000	1.280000	6.000000	0.000000	0.000000	0.000000	1.000000	8.000000
50%	1096.000000	4.800000	0.591820	0.233000	5.500000	20.590000	5.000000	1.790000	1.500000	10.000000	10.000000
75%	1670.000000	10.000000	1.381760	0.544000	23.530000	58.140000	20.000000	7.140000	3.600000	16.000000	10.000000
max	4000.000000	100.000000	100.000000	39.370079	100.000000	100.000000	100.000000	100.000000	100.000000	40.000000	10.000000

La plus basse des moyennes est pour le sodium, mais celui ci est fortement corrélé à la quantité de sel



Pas de contexte dans lequel le remplacement par une valeur nulle serait sensé

VALEURS MOYENNES OU MEDIANES

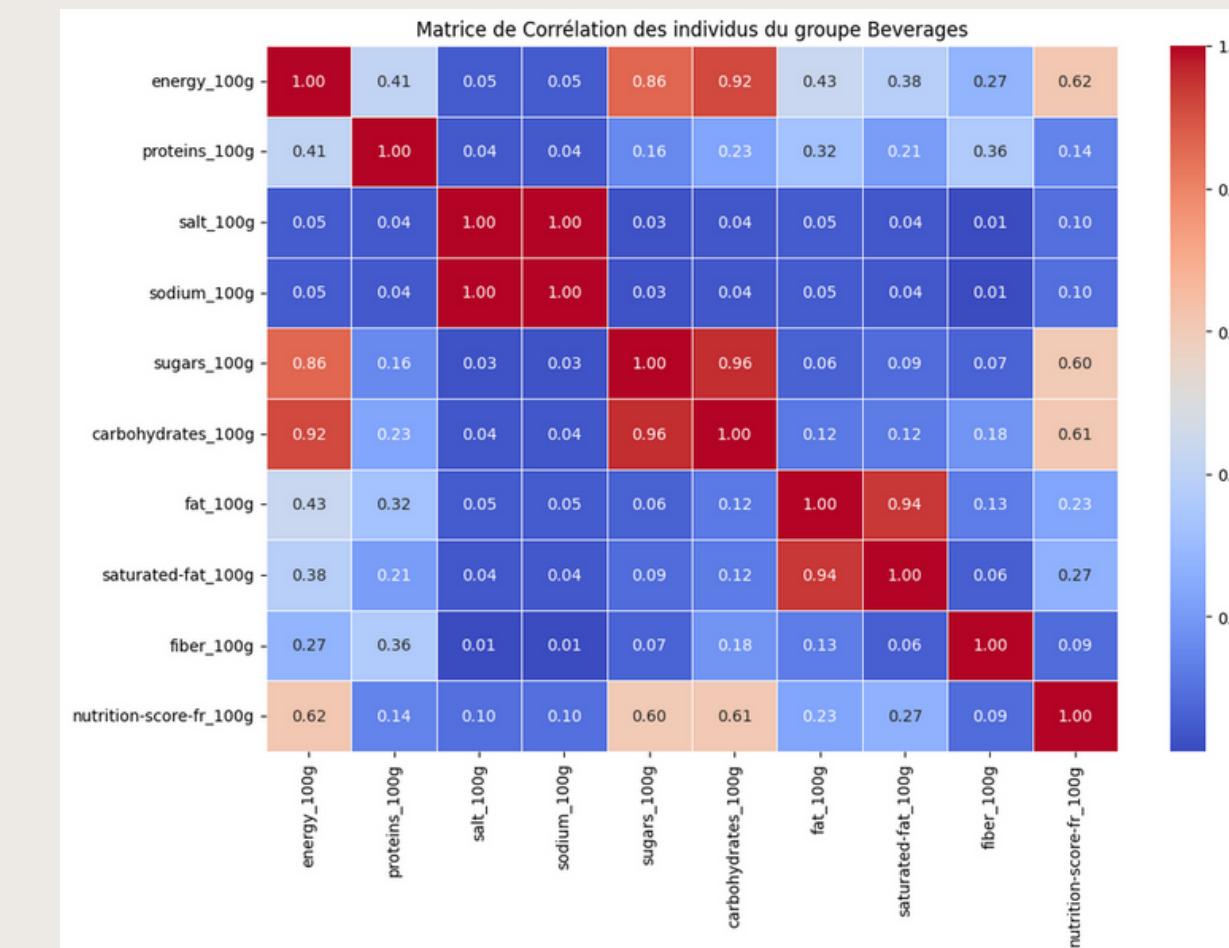
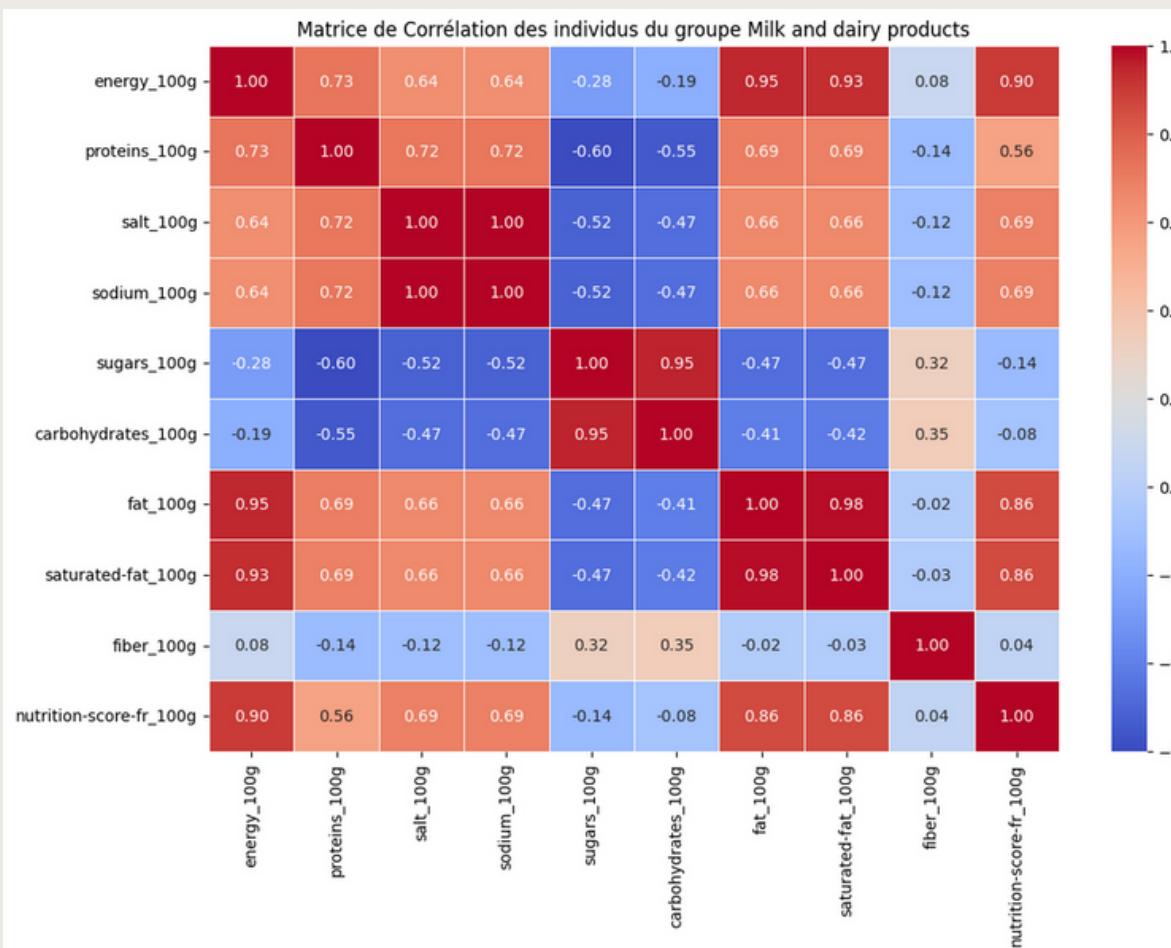
Tri des features par catégorie, moyenne, médiane et écart type

On recherche des couples feature-catégories dont l'écart type est significativement petit devant la moyenne ou la médiane

Les écart-types sont trop grands pour envisager un remplacement par une valeur moyenne ou une valeur médiane

	colonne	categorie	nb_indiv	moyenne	med	ecartype
33	sodium_100g	Beverages	8337	0.057681	0.003937	0.754042
23	salt_100g	Beverages	8337	0.146485	0.010000	1.915097
31	sodium_100g	Sugary snacks	12219	0.158394	0.094488	0.490830
30	sodium_100g	Fruits and vegetables	6578	0.215373	0.040000	0.853708
32	sodium_100g	Cereals and potatoes	8063	0.255149	0.169291	0.339987
34	sodium_100g	Milk and dairy products	8494	0.266830	0.060000	0.326449
21	salt_100g	Sugary snacks	12219	0.402276	0.240000	1.246149
84	fiber_100g	Milk and dairy products	8494	0.405435	0.000000	0.903144
36	sodium_100g	Composite foods	6568	0.445173	0.366142	0.668991
73	saturated-fat_100g	Beverages	8337	0.508119	0.000000	2.598258
20	salt_100g	Fruits and vegetables	6578	0.546944	0.101600	2.168169
39	sodium_100g	Salty snacks	2659	0.598333	0.572000	0.620158
83	fiber_100g	Beverages	8337	0.642190	0.100000	2.989621
22	salt_100g	Cereals and potatoes	8063	0.648073	0.430000	0.863526
24	salt_100g	Milk and dairy products	8494	0.677646	0.152400	0.829160
35	sodium_100g	Fat and sauces	4807	0.739193	0.500000	1.088525
70	saturated-fat_100g	Fruits and vegetables	6578	0.782108	0.100000	4.342494
88	fiber_100g	Fish Meat Eggs	7665	0.798647	0.100000	4.271956

VALEURS MOYENNES OU MEDIANES



Matrice de corrélation entre features par catégories

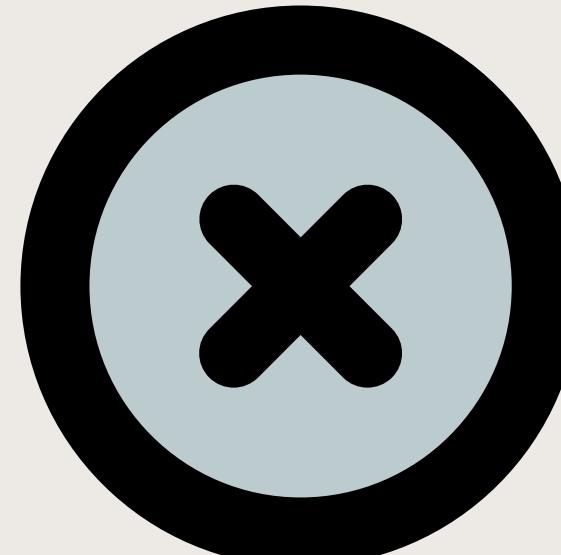
ITERATIVE IMPUTING



Tient compte des relations entre les variables en utilisant une méthode itérative

Peut être plus précis que le remplissage par valeurs constantes

Gère les valeurs manquantes de manière plus sophistiquée.



Plus complexe à mettre en œuvre

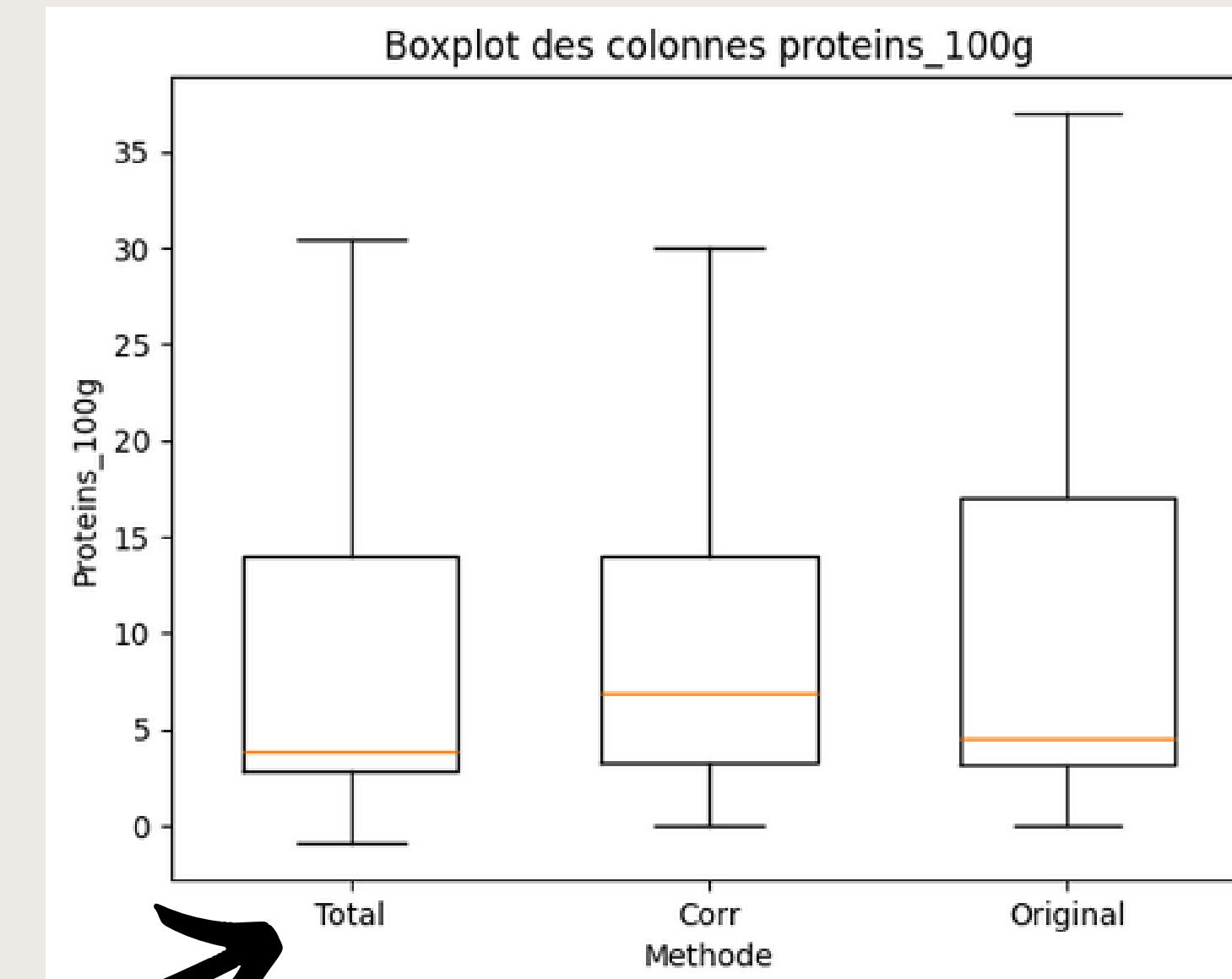
Plus de temps de calcul nécessaire

Sensible aux valeurs aberrantes et aux données non linéaires.

ITERATIVE IMPUTING

Remplissage total du
dataframe par iterative
imputer

On remarque que les valeurs
imputés par iterative
imputer sont distribuées de
manière relativement
similaires



Remplissage d'une
colonne particulière d'une
catégorie particulière

ITERATIVE IMPUTING

df_work2_ter.describe()											
	energy_100g	proteins_100g	salt_100g	sodium_100g	sugars_100g	carbohydrates_100g	fat_100g	saturated-fat_100g	fiber_100g	nutrition-score-fr_100g	nb_valeurs
count	7236.000000	8494.000000	6747.000000	6746.000000	6717.000000	6902.000000	7283.000000	6753.000000	3128.000000	6644.000000	8494.000000
mean	815.345522	9.653637	0.677646	0.266830	7.634871	8.918180	13.967858	9.155958	0.405435	8.145545	9.476454
std	496.699274	8.078891	0.829160	0.326449	8.001201	9.656374	11.826579	8.052547	0.903144	6.811766	3.783322
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-6.000000	1.000000
25%	372.000000	3.300000	0.101600	0.040000	0.800000	1.500000	3.000000	1.900000	0.000000	2.000000	11.000000
50%	676.500000	6.900000	0.152400	0.060000	4.700000	4.800000	10.000000	6.000000	0.000000	8.000000	11.000000
75%	1243.000000	14.000000	1.300000	0.511811	13.000000	14.000000	25.000000	17.000000	0.500000	14.000000	12.000000
max	2448.000000	52.200000	9.652000	3.800000	91.000000	95.000000	60.000000	43.100000	24.000000	30.000000	12.000000
df_work2.describe()											
	energy_100g	proteins_100g	salt_100g	sodium_100g	sugars_100g	carbohydrates_100g	fat_100g	saturated-fat_100g	fiber_100g	nutrition-score-fr_100g	nb_valeurs
count	237032.000000	235983.000000	232047.000000	232008.000000	222826.000000	220150.000000	220470.000000	209207.000000	183064.000000	202033.000000	258053.000000
mean	1118.079247	7.078799	1.605682	0.632263	15.733676	32.025061	12.537453	5.083836	2.823858	9.074013	8.505307
std	786.870195	8.073821	6.284254	2.474302	20.928711	29.045646	17.188183	7.903319	4.582908	9.041357	2.801647
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-15.000000	0.000000
25%	382.775000	0.700000	0.063500	0.025000	1.280000	6.000000	0.000000	0.000000	0.000000	1.000000	8.000000
50%	1096.000000	4.800000	0.591820	0.233000	5.500000	20.590000	5.000000	1.790000	1.500000	10.000000	10.000000
75%	1670.000000	10.000000	1.381760	0.544000	23.530000	58.140000	20.000000	7.140000	3.600000	16.000000	10.000000
max	4000.000000	100.000000	100.000000	39.370079	100.000000	100.000000	100.000000	100.000000	100.000000	40.000000	10.000000

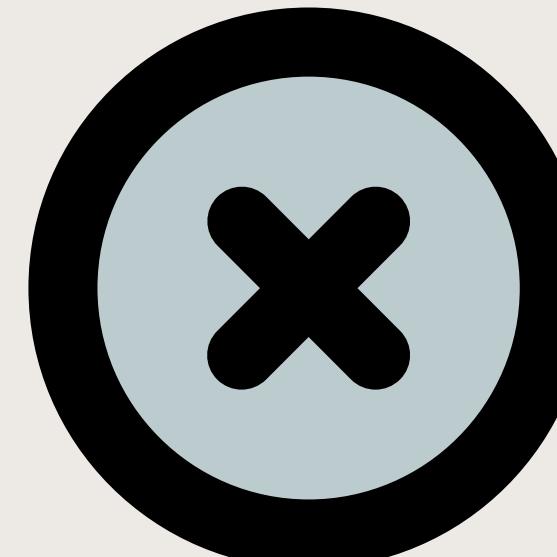
En revanche, les moyennes, min, max sont fortement modifiées pour les autres features, ce qui semble assez inquiétant quant à la qualité des données imputées

KNN



Utilise l'information des voisins les plus proches pour estimer les valeurs manquantes.

Peut être plus précis que d'autres méthodes, surtout si les données sont bien regroupées.



Sensible à la dimensionnalité élevée des données

Plus de temps de calcul nécessaire, surtout avec de grands ensembles de données

Nécessite de définir le nombre de voisins K.

KNN

imputation des toutes les valeurs de chaque feature du dataframe

Les valeurs sont largement moins modifiées que selon la méthode d'iterative imputer

KNN

Scoring du KNN sur la prédiction de la valeur pour la feature ‘nutrition_grade_fr’

La précision est assez basse et nécessiterait un travail plus approfondi pour l'améliorer.

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Diviser le dataframe en features (X) et la cible (y)
X = df_score[['energy_100g', 'proteins_100g', 'salt_100g', 'sodium_100g',
               'sugars_100g', 'carbohydrates_100g', 'fat_100g', 'saturated-fat_100g',
               'fiber_100g', 'nutrition-score-fr_100g', 'nb_valeurs']]
y = df_score['nutrition_grade_fr']

# Diviser le jeu de données en ensembles d'entraînement et de test (80% pour l'entraînement et 20% pour le test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialiser et entraîner le modèle KNN
knn = KNeighborsClassifier(n_neighbors=5) # Vous pouvez choisir le nombre de voisins (k) que vous souhaitez
knn.fit(X_train, y_train)

# Faire des prédictions sur l'ensemble de test
y_pred = knn.predict(X_test)

# Calculer la précision du modèle
accuracy = accuracy_score(y_test, y_pred)
print("Précision du modèle KNN : {:.2f}%".format(accuracy * 100))
```

Précision du modèle KNN : 87.66%

KNN

Une imputation de valeurs pour une feature qualitative a été réalisée par la méthode KNN

```
# Création d'un dictionnaire pour mapper les valeurs de 'nutrition_grade_fr' vers les valeurs numériques
grade_mapping = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

# Ajout de la colonne 'nutrition_grade_fr_encoded' en appliquant le mapping
df_work2_imputed['nutrition_grade_fr_encoded'] = df_work2_imputed['nutrition_grade_fr'].map(grade_mapping)
```

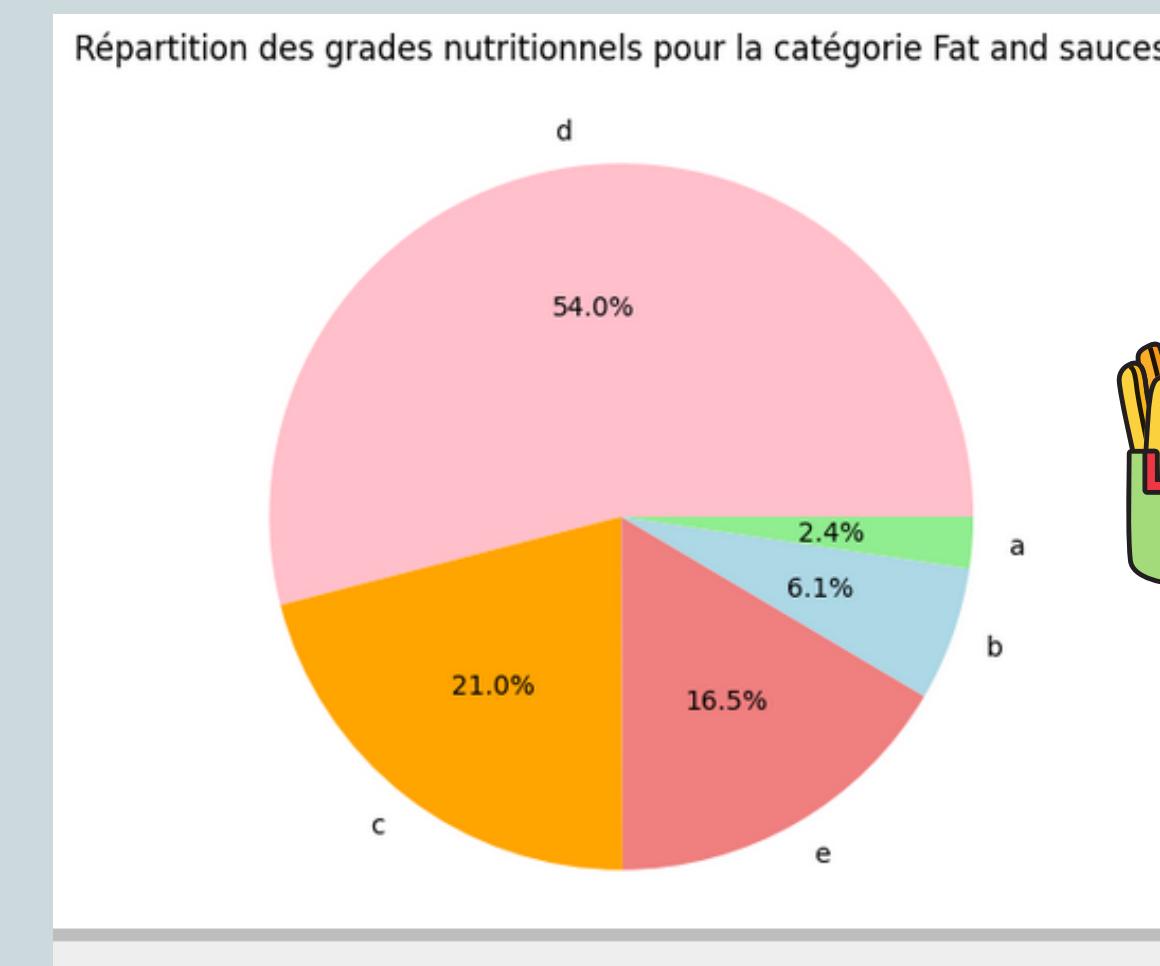
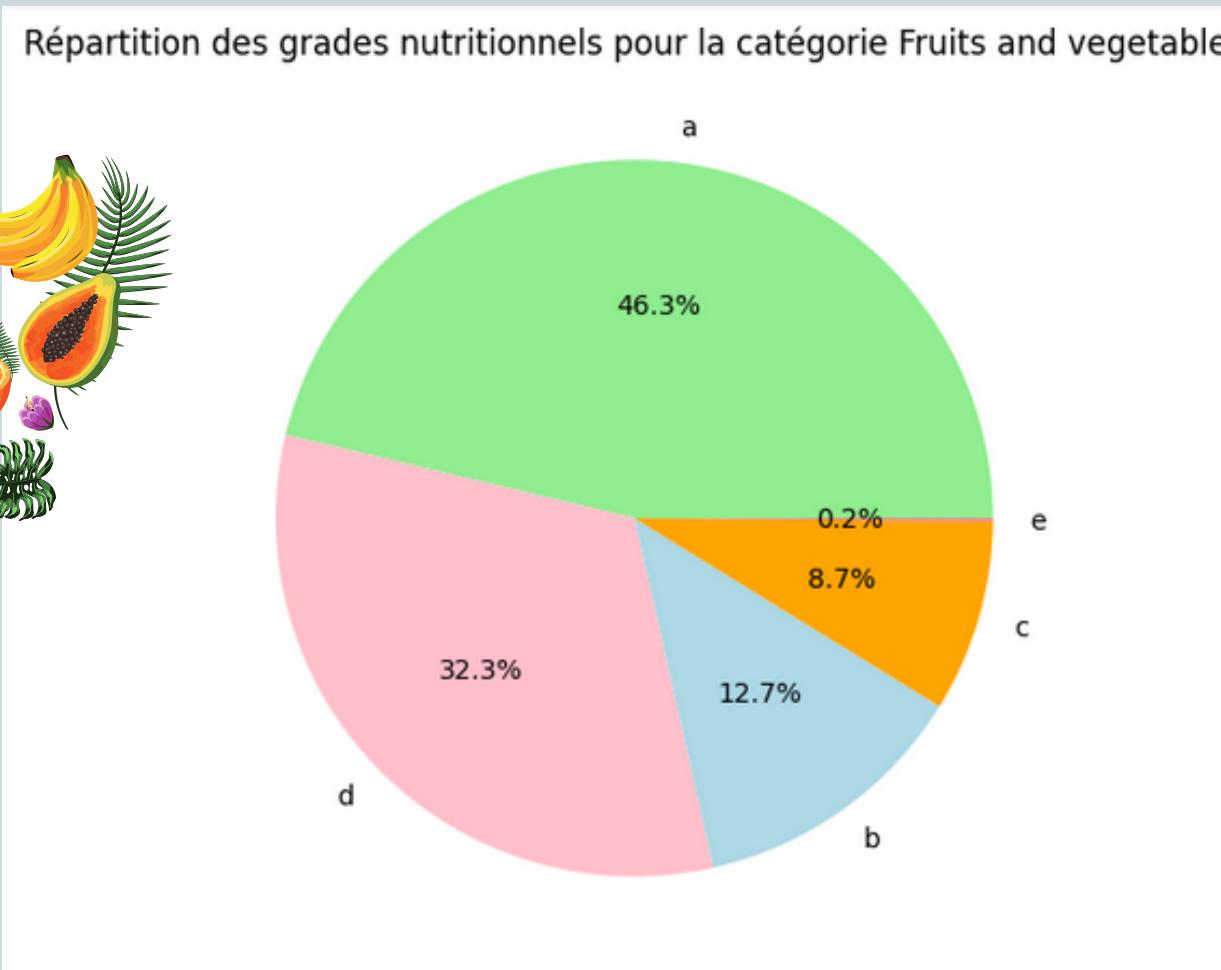
EXPLORATION DES DONNÉES

Le dataset est désormais nettoyé et complet

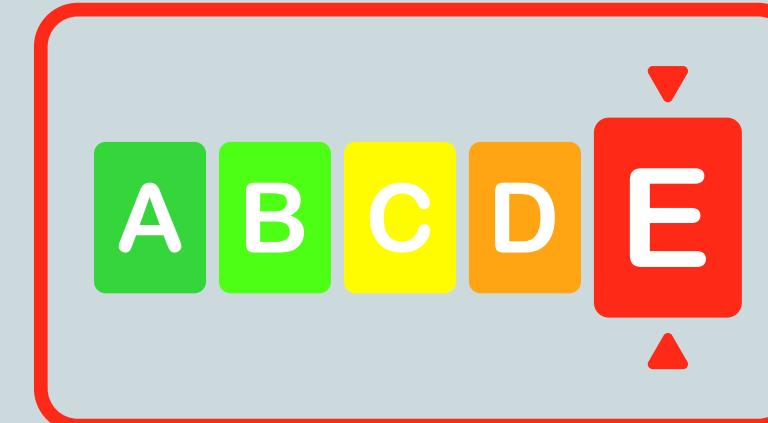
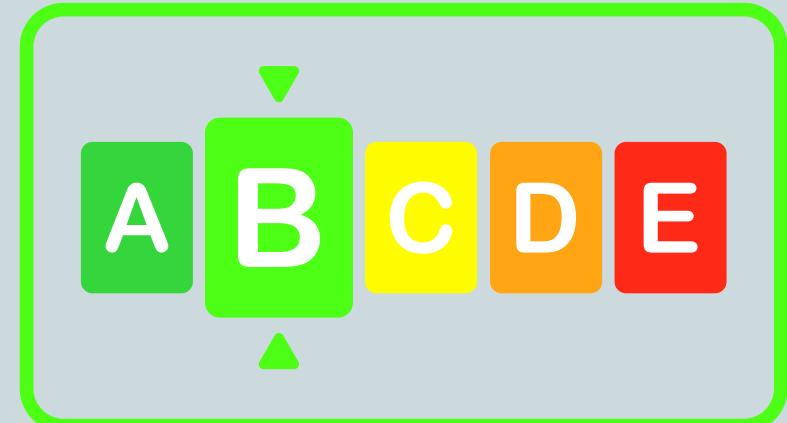
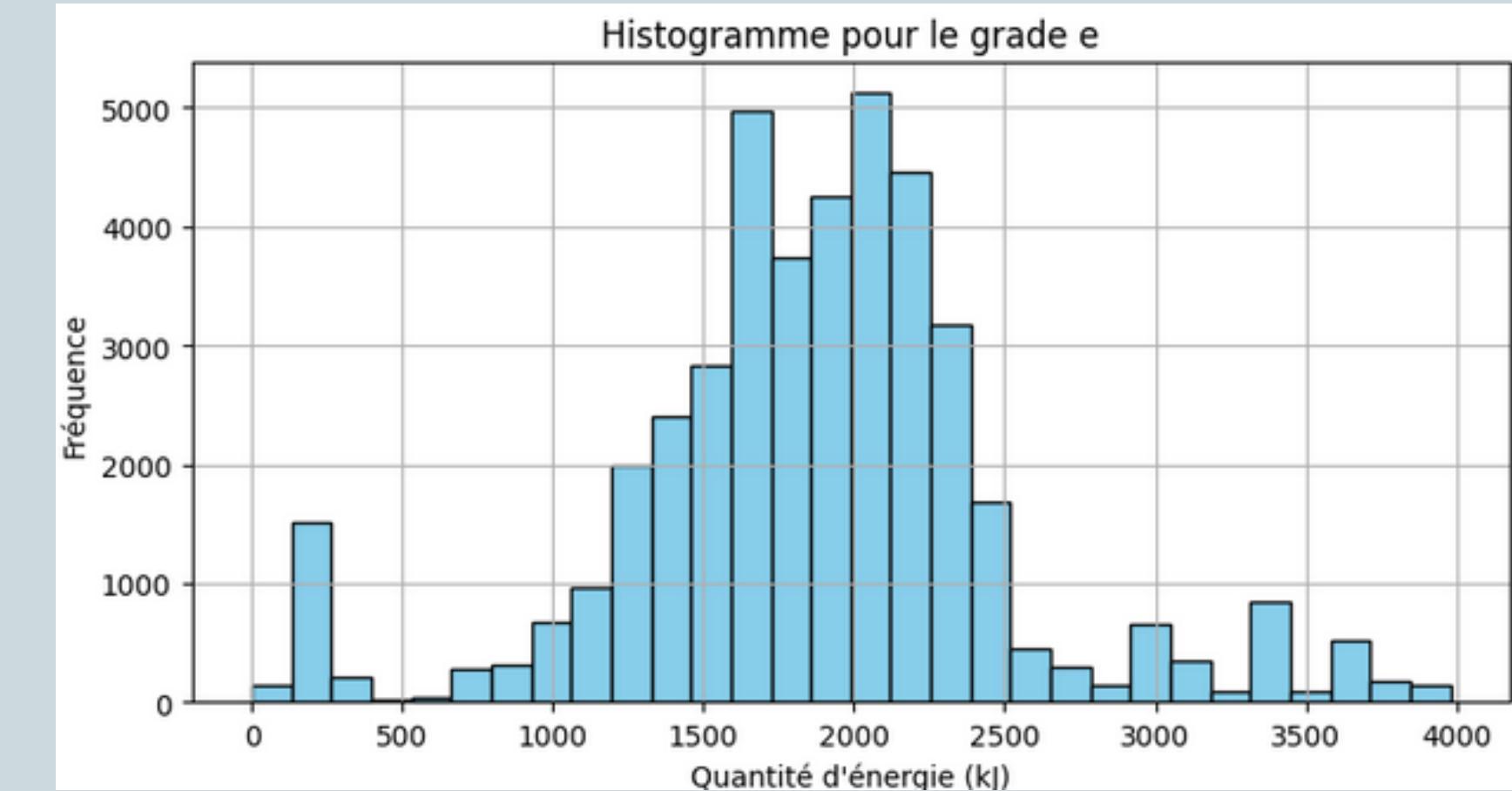
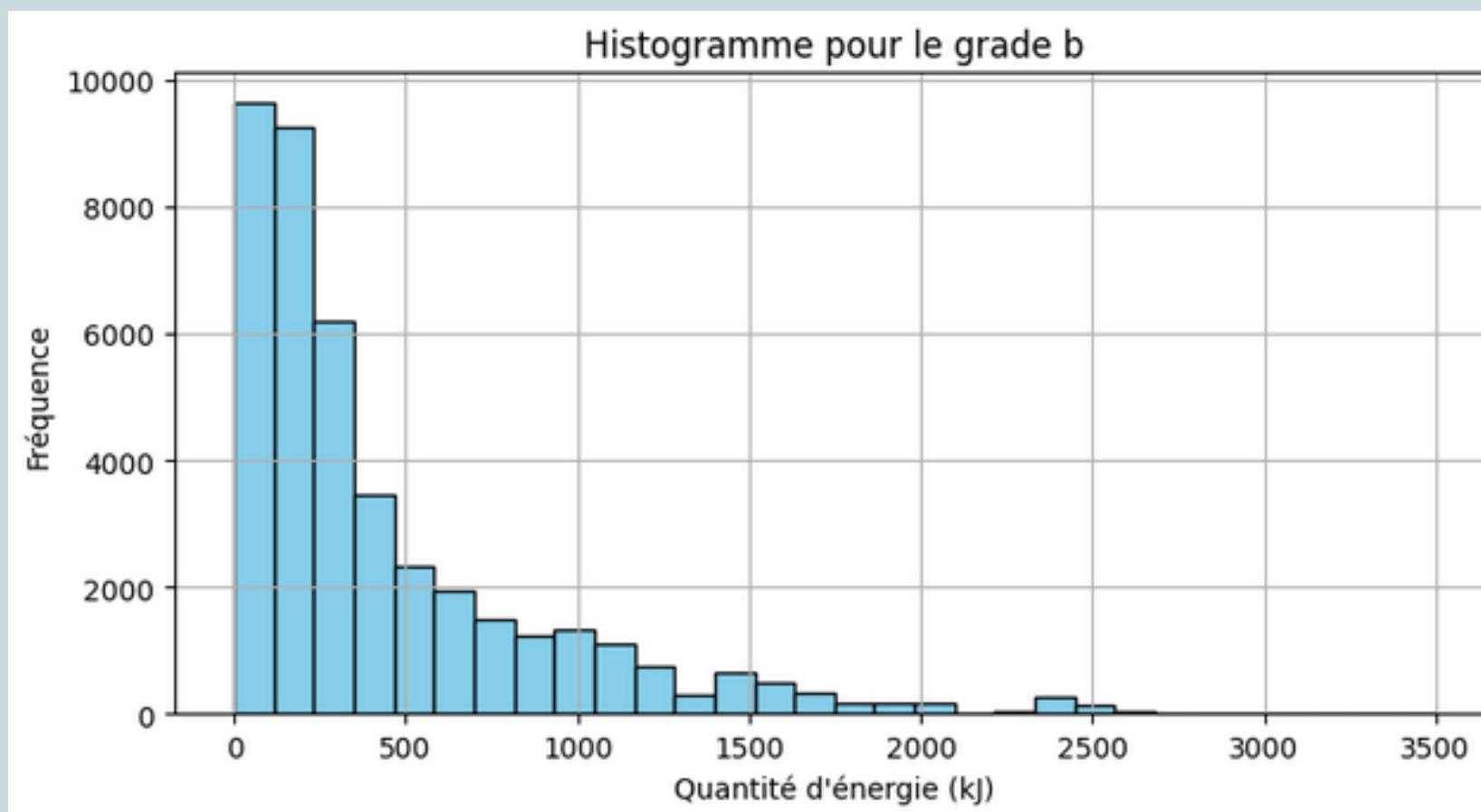
- 1 Analyses Uni-variées
- 2 Analyse Bi-Variées
- 3 Analyses Multi-Variées



ANALYSE UNIVARIÉE



ANALYSE UNIVARIÉE

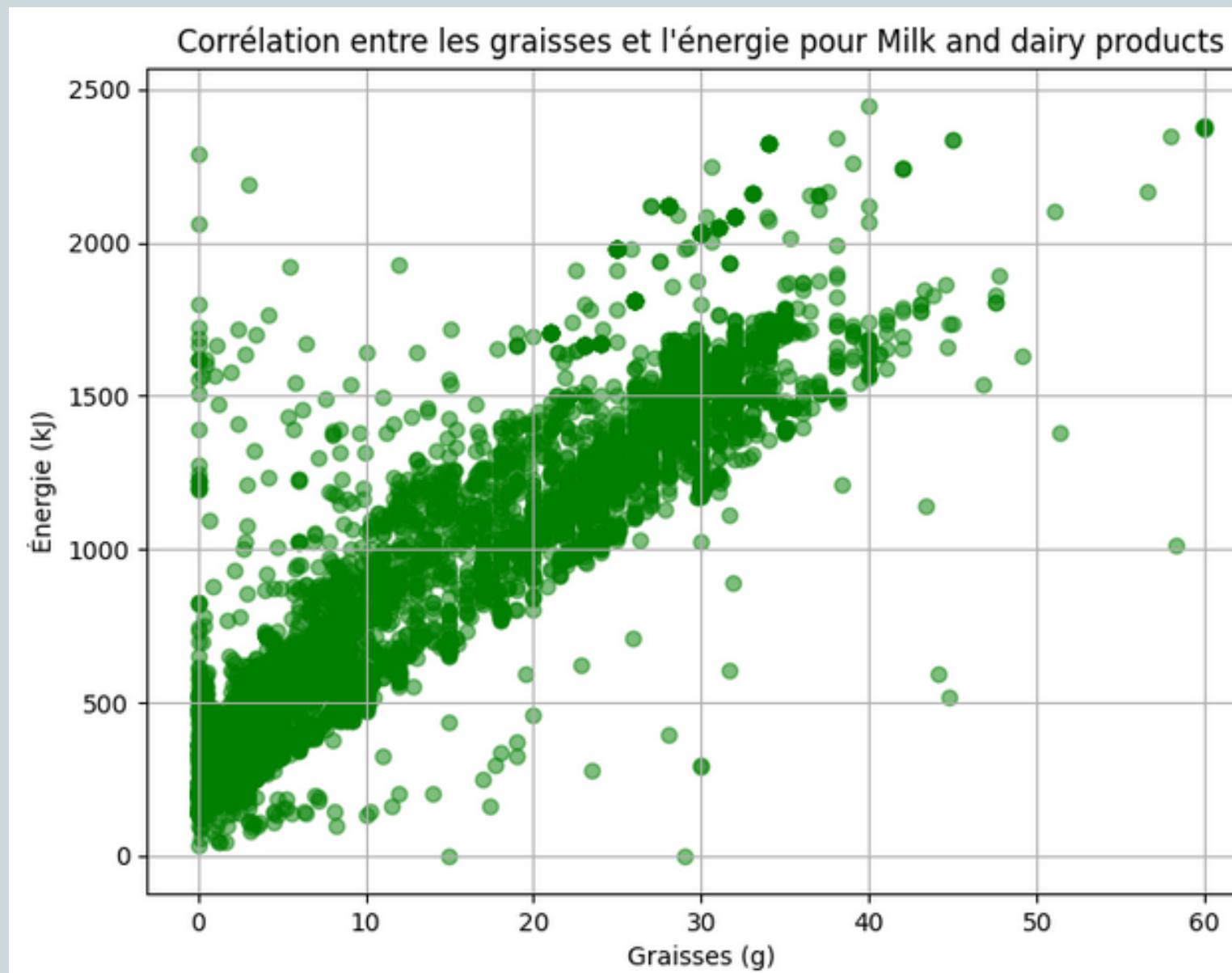


ANALYSE UNIVARIÉE

Deux features qui nous offrent des différenciations entre produits

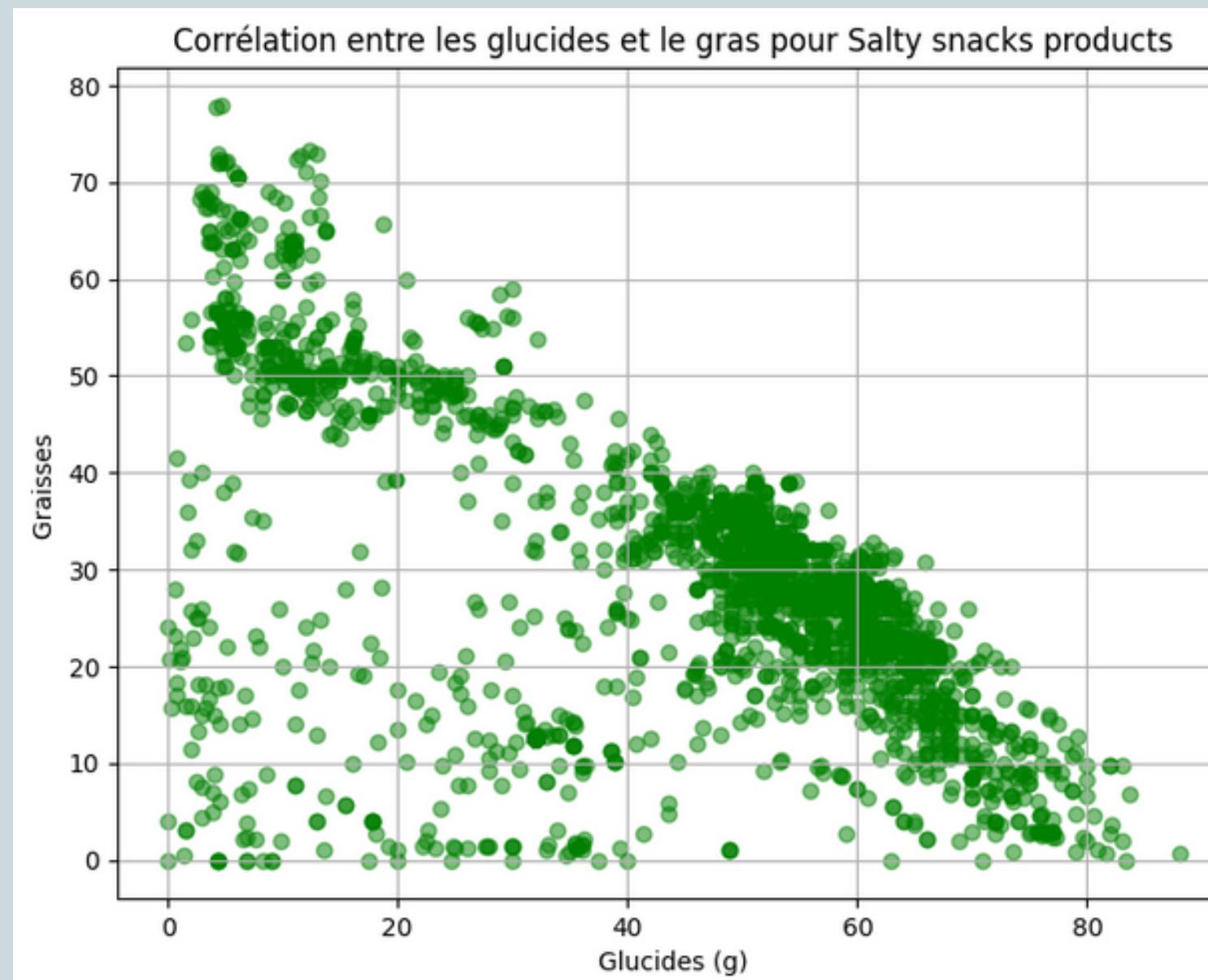
- Le nutriscore
- La catégorie

ANALYSE BI-VARIÉE



Corrélation positive

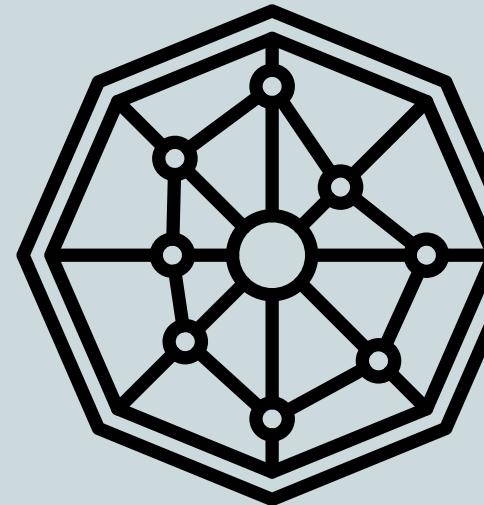
ANALYSE BI-VARIÉE



Corrélation négative

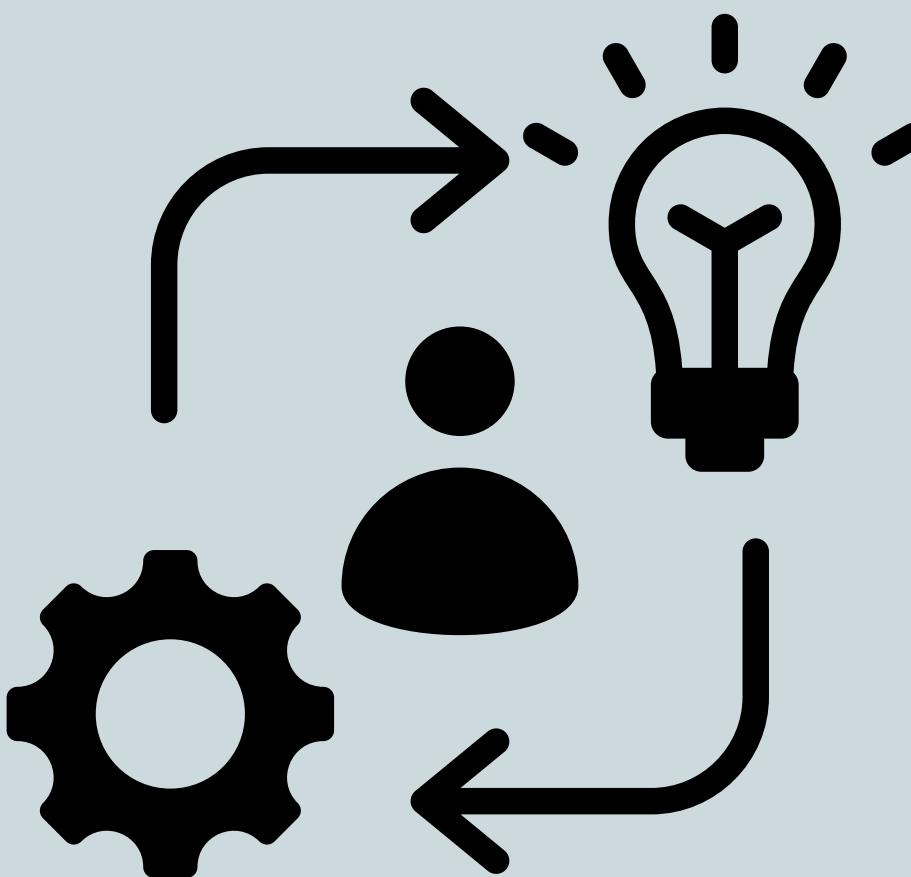


ANALYSE MULTI-VARIÉE



Deux méthodes d'analyse multi-variée

- Analyse en Composante Principale
- ANalyse Of Variance



ANALYSE MULTI-VARIÉE

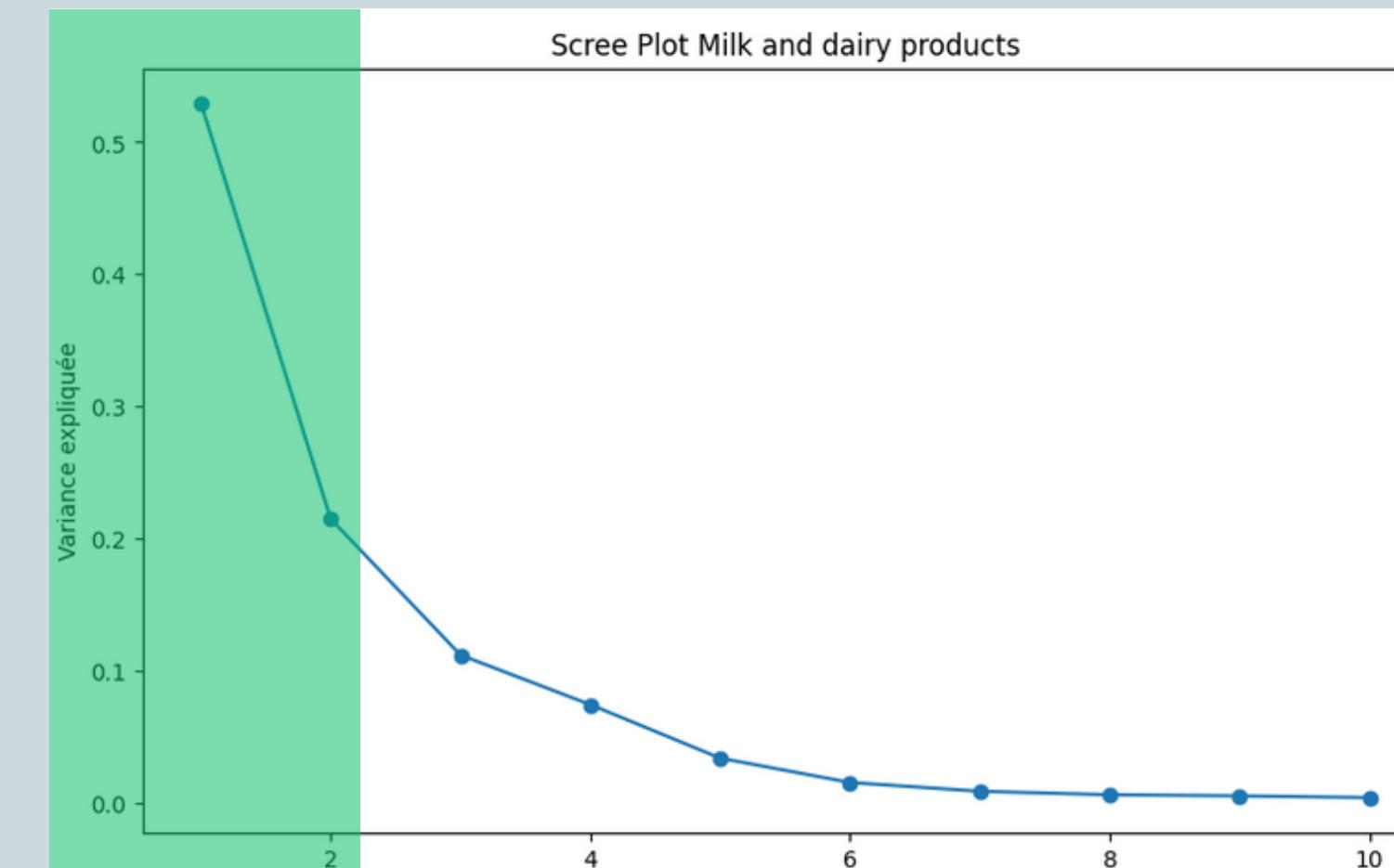
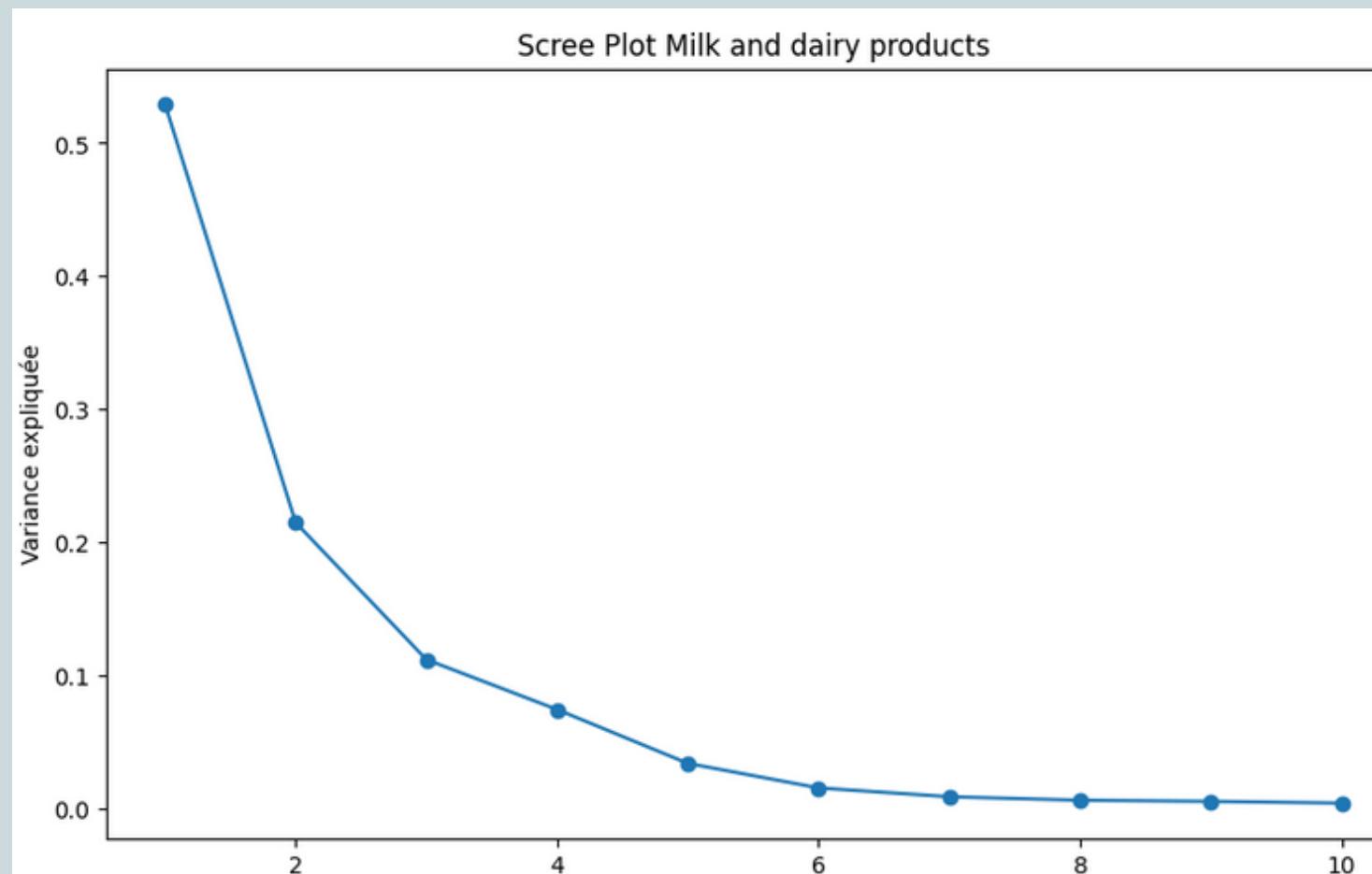
ACP

Méthode mathématiques de réduction de dimension basée sur la maximisation de la variance

Objectif : simplifier les données en identifiant des tendances importantes

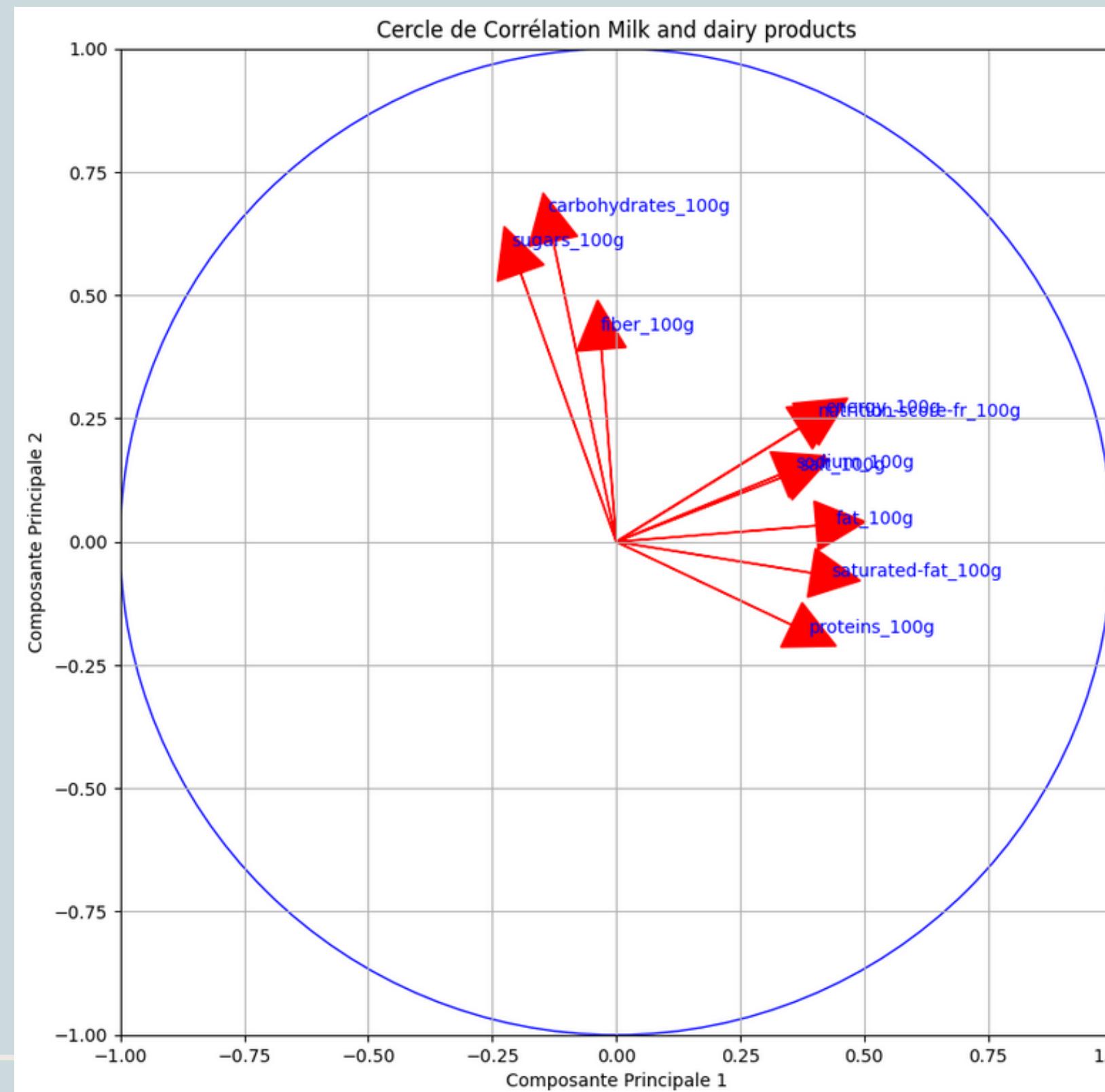


ANALYSE MULTI-VARIÉE



+70% de variance expliquée sur les 2 premières composantes

ANALYSE MULTI-VARIÉE



On voit apparaître assez clairement sur le cercle des corrélations deux nouvelles composantes

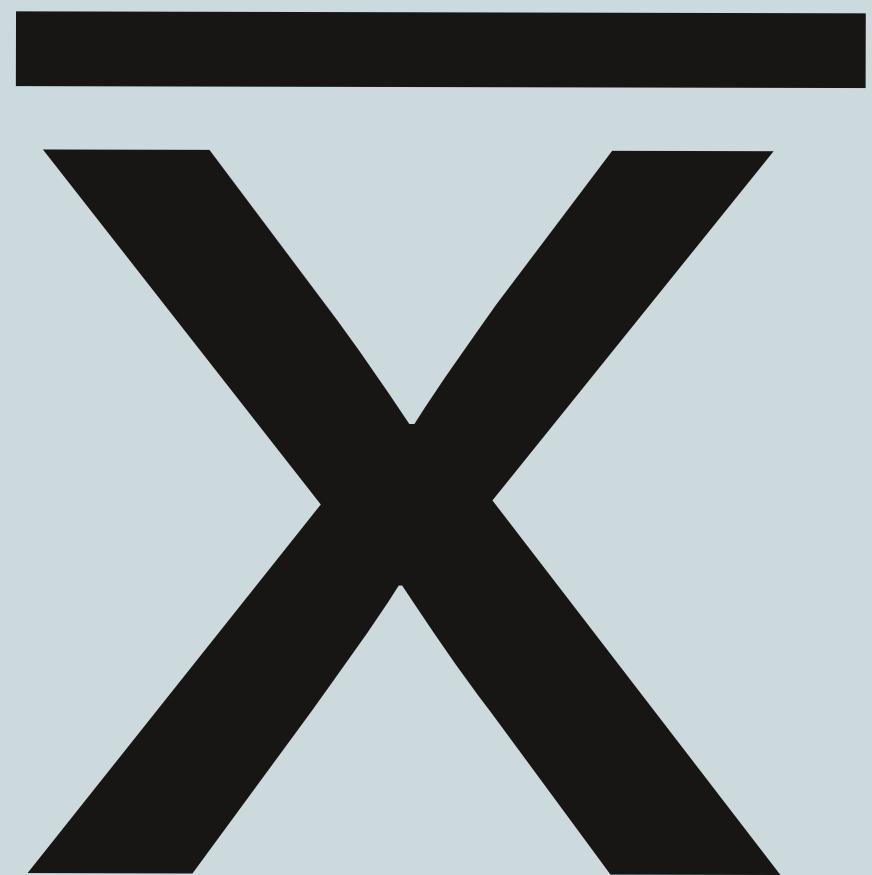
une dimension regroupant :

- fat, saturated_fat, sodium, salt, proteins

et une autre :

- glucides, sucre, fibres

ANALYSE MULTI-VARIÉE



ANOVA

méthode statistique comparant les moyennes de plusieurs groupes pour détecter les différences significatives.

ANALYSE MULTI-VARIÉE

Les résultats de l'ANOVA montrent que pour chaque variable nutritionnelle ('energy_100g', 'proteins_100g', 'salt_100g', 'sodium_100g', 'sugars_100g', 'carbohydrates_100g', 'fat_100g', 'saturated-fat_100g', 'fiber_100g'), la p-value est extrêmement faible (0.0). Cela signifie qu'il existe des différences significatives entre les différentes catégories de nutriscore pour chacune de ces variables nutritionnelles.

En d'autres termes, les contenus nutritionnels des aliments, tels que l'énergie, les protéines, le sel, le sucre, etc., varient considérablement en fonction du nutriscore attribué à chaque aliment.

```
ANOVA for energy_100g:  
F-statistic: 40593.76178260351  
p-value: 0.0
```

```
-----  
ANOVA for proteins_100g:  
F-statistic: 2642.4524791807794  
p-value: 0.0
```

```
-----  
ANOVA for salt_100g:  
F-statistic: 1631.667867149198  
p-value: 0.0
```

```
-----  
ANOVA for sodium_100g:  
F-statistic: 1616.1676663900823  
p-value: 0.0
```

```
-----  
ANOVA for sugars_100g:  
F-statistic: 14605.137473480041  
p-value: 0.0
```

```
-----  
ANOVA for carbohydrates_100g:  
F-statistic: 6466.289722246683  
p-value: 0.0
```

```
-----  
ANOVA for fat_100g:  
F-statistic: 25170.786683899285  
p-value: 0.0
```

```
-----  
ANOVA for saturated-fat_100g:  
F-statistic: 42757.93475239021  
p-value: 0.0
```

```
-----  
ANOVA for fiber_100g:  
F-statistic: 3824.1605463401443  
p-value: 0.0
```

```
-----
```

RESPECT DU RGPD

OpenFoodFacts

REGLEMENTATION

1. Principe de légalité, loyauté et transparence :

- Seules les données nécessaires afin d'atteindre les objectifs définis par le projet sont collectées.
- Les utilisateurs sont informés de manière claire et transparente sur la collecte et l'utilisation de leurs données.

2. Principe de limitation des finalités :

- Les données collectées ne sont utilisées que dans le cadre spécifique de ce projet de recherche.
- Elles ne sont pas utilisées à d'autres fins sans le consentement explicite des utilisateurs.

3. Principe de minimisation des données :

- Éviter la collecte de données excessives ou non pertinentes.

4. Principe d'exactitude :

- Prendre des mesures pour rectifier les données inexactes ou incomplètes dès que leur existence est constatée.

5. Principe de conservation des données :

- Mettre en place des politiques de conservation des données pour garantir qu'elles sont supprimées ou anonymisées dès qu'elles ne sont plus nécessaires.



Politique de confidentialité d'Open Food Facts

En tant qu'organisation à but non lucratif, nous dépendons de vos dons pour continuer à informer...

 openfoodfacts.org



Conditions d'utilisation, de contribution et de réutilisation

En tant qu'organisation à but non lucratif, nous dépendons de vos dons pour continuer à informer...

 openfoodfacts.org

Le projet respecte pleinement les principes du RGPD en matière de protection des données personnelles. Les mesures mises en place assurent la légalité, la transparence et la sécurité des données, tout en respectant les droits et les préoccupations des utilisateurs.

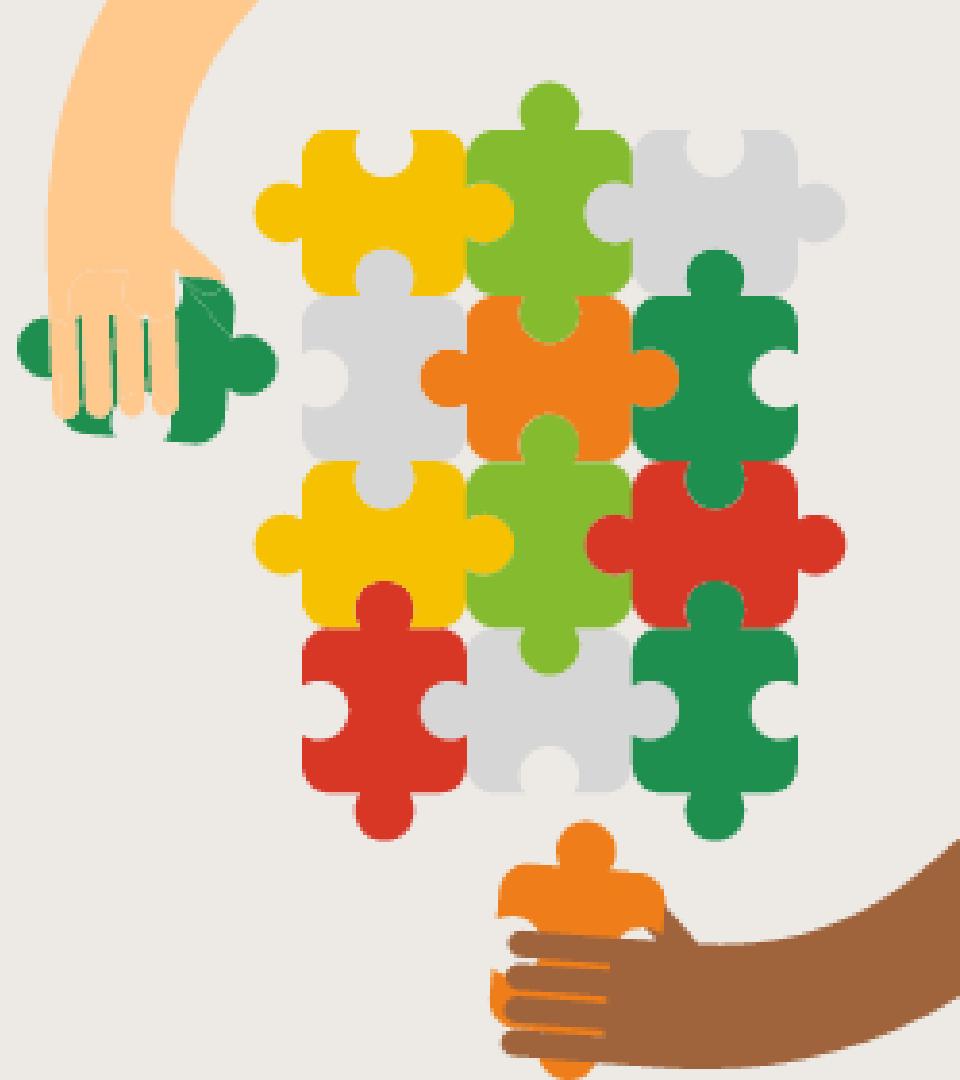
CONCLUSION

Des données nombreuses et significatives

Le dataset fourni par OpenFoodFacts possède les caractéristiques nécessaires pour constituer une base permettant de prédire des valeurs pour des valeurs cibles

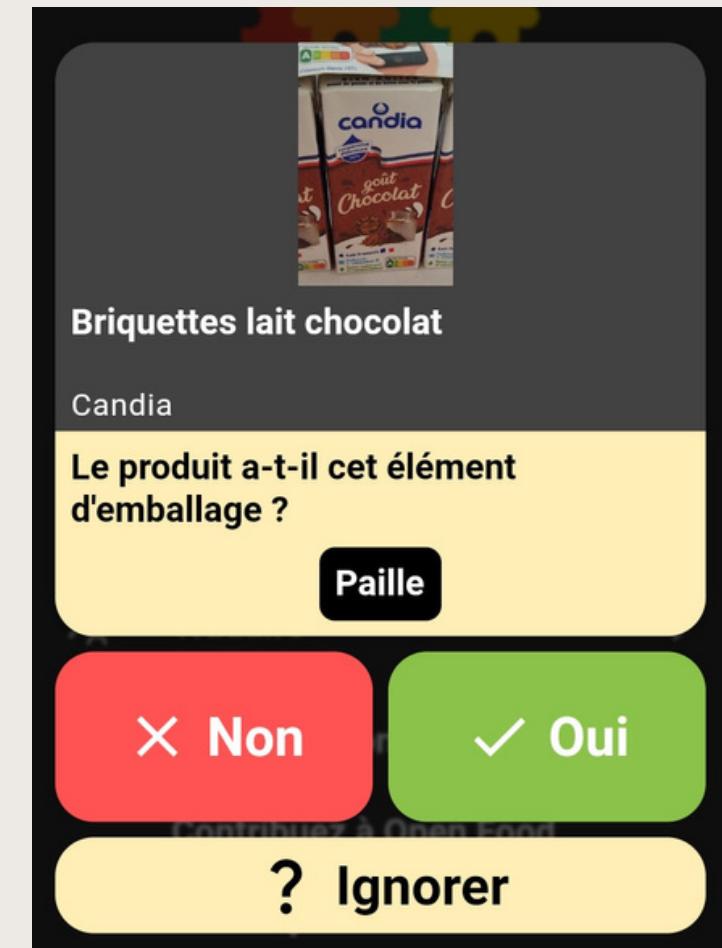
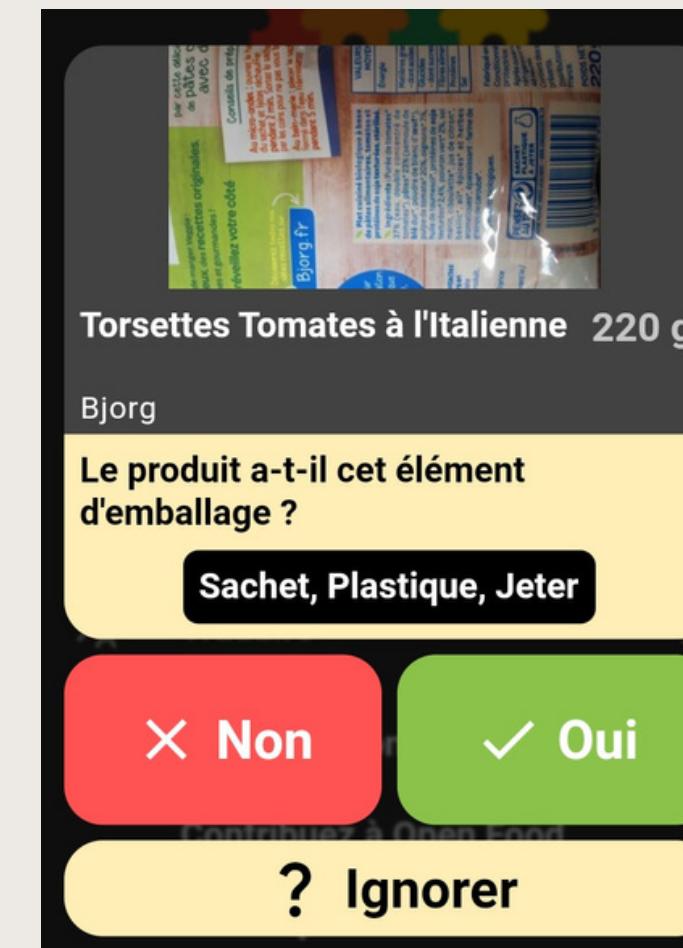
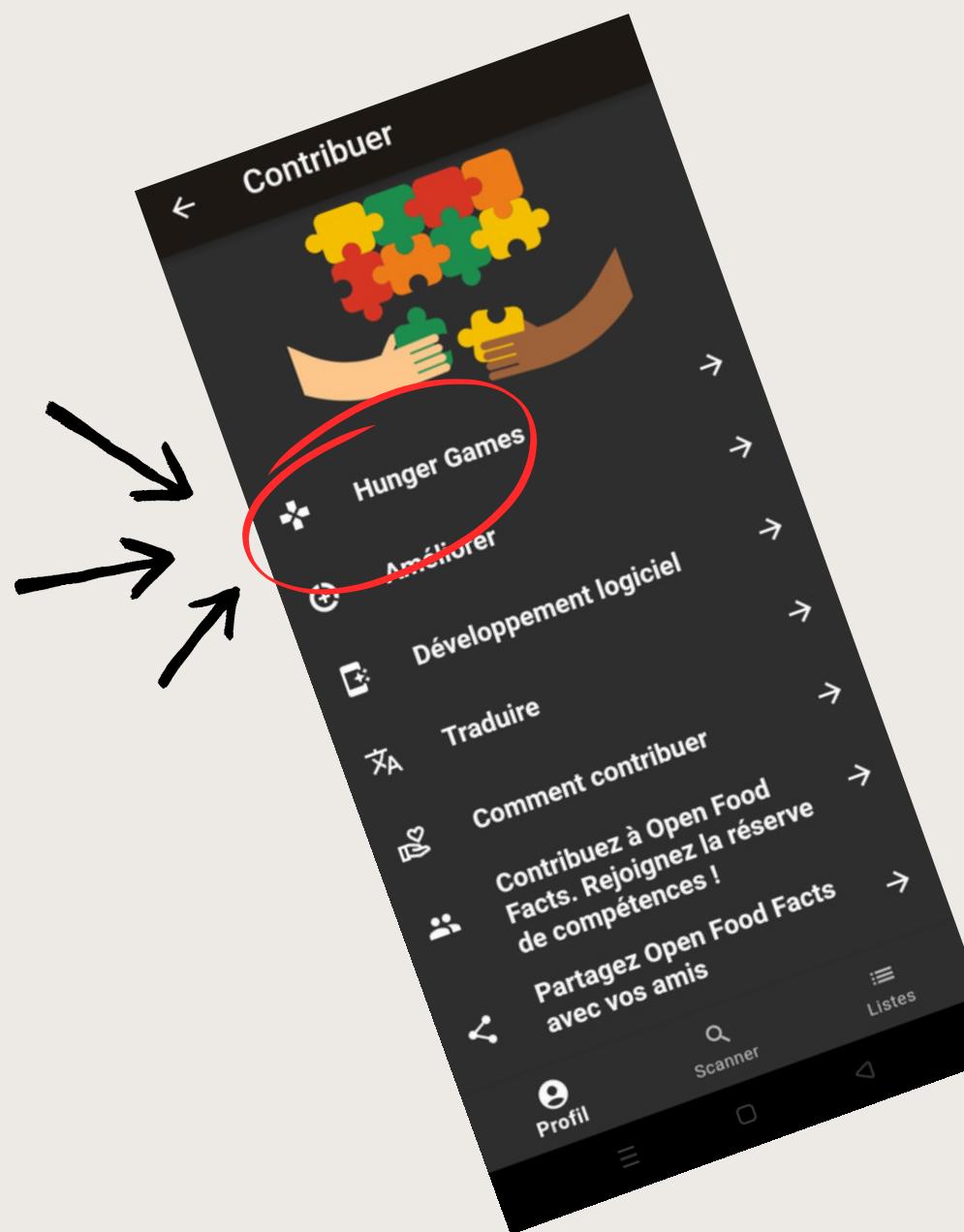
En se basant sur un ensemble de données nettoyées, débarrassé de ses valeurs aberrantes, en sélectionnant des features on peut prévoir des valeurs en croisant les informations

L'analyse en composante principale nous montre qu'il est possible de trier les individus selon de nouvelles dimensions et l'ANOVA nous démontre que la catégorie à laquelle appartient un produit est une donnée essentielle.



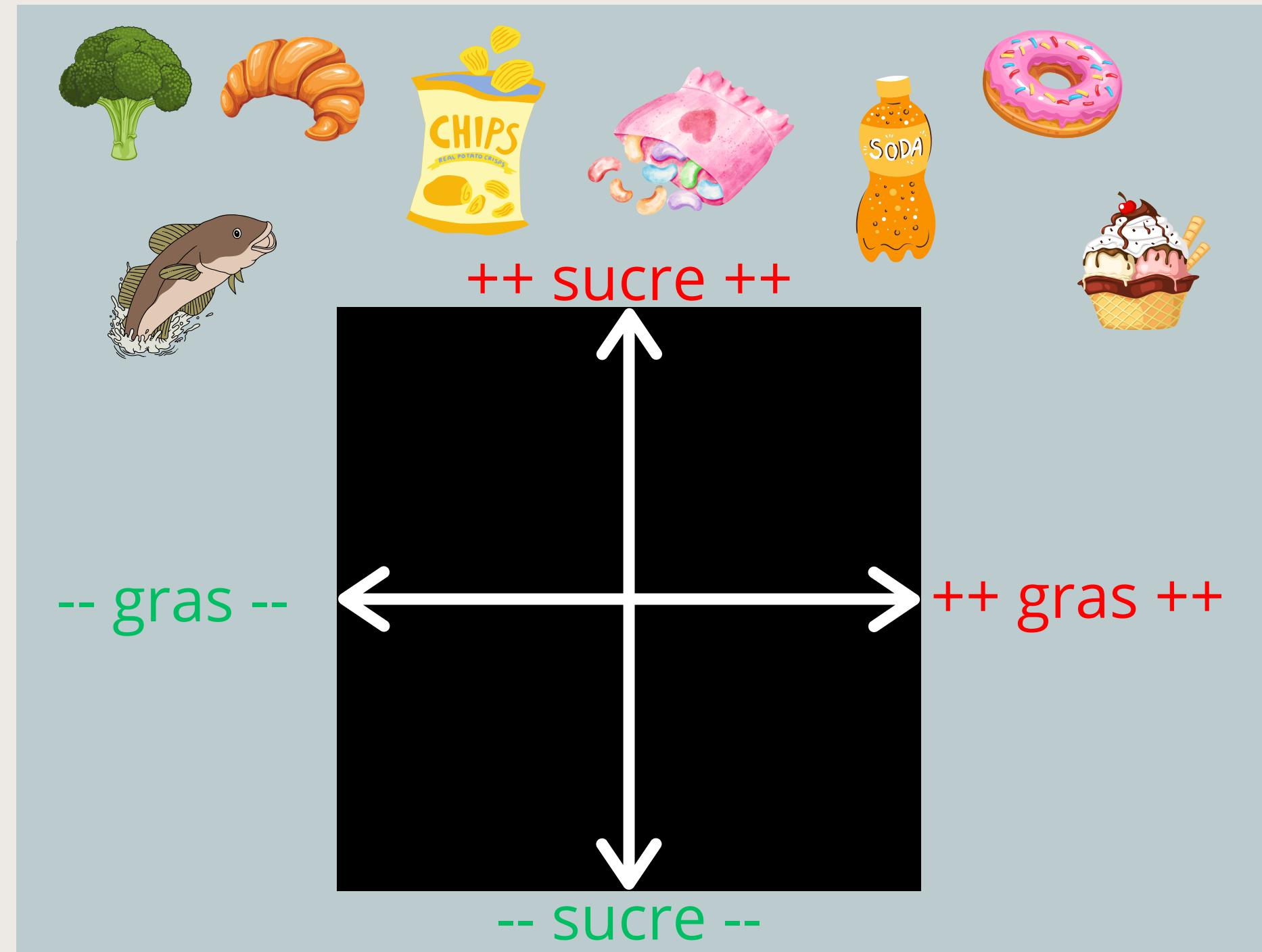
LUDIFICATION DE LA CONTRIBUTION

OpenFoodFacts a mis en place une interface ludique afin de permettre aux utilisateurs de participer à la construction des informations



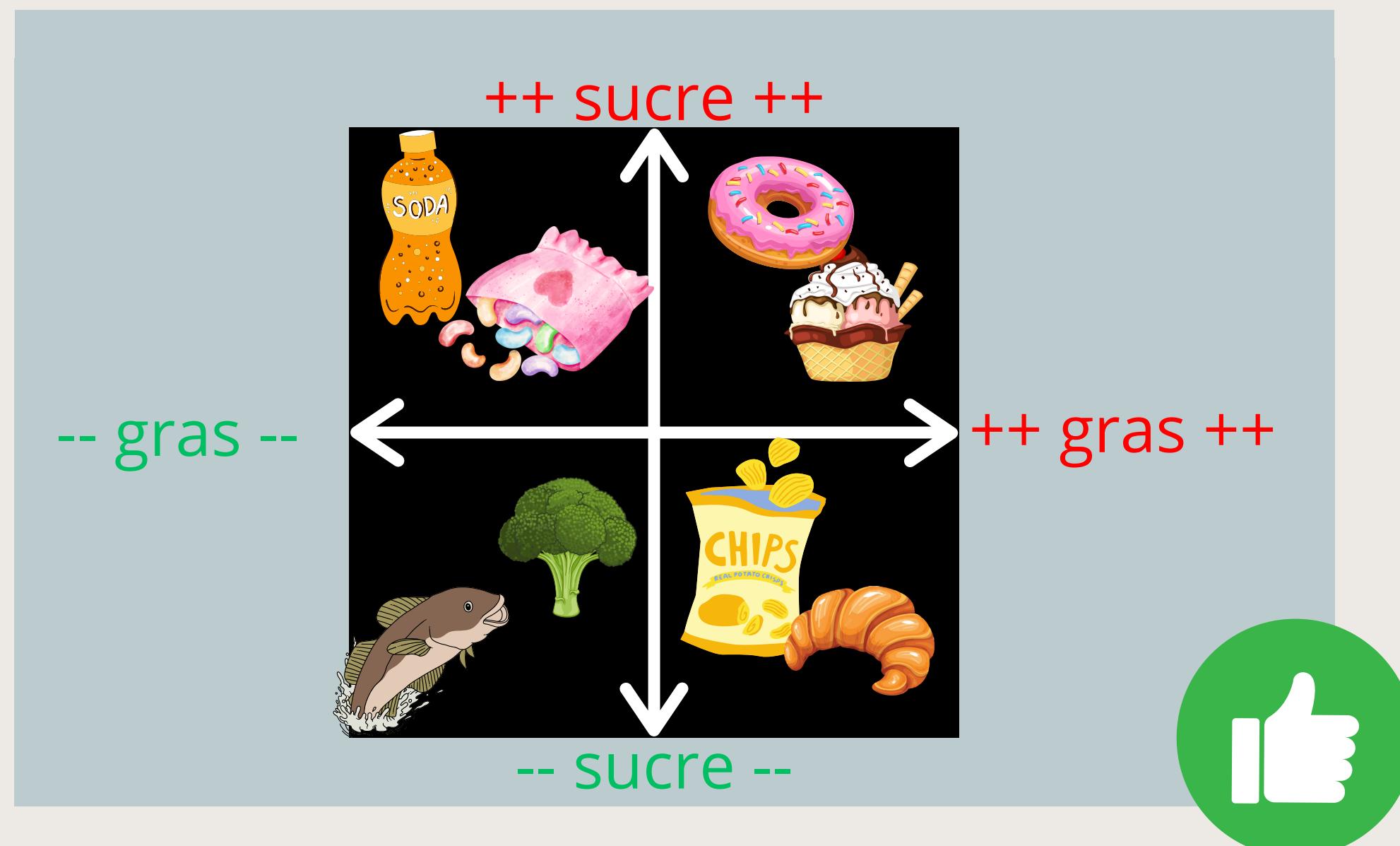
IDEE LUDIFICATION

Envisageons un jeu permettant à l'utilisateur d'interagir avec les données



IDEE LUDIFICATION

Envisageons un jeu permettant à l'utilisateur d'interagir avec les données



OpenFoodFacts

MERCI