

PROJET 03 :

Préparez des données pour un organisme de santé publique

Mohamed A.

Regles et principes :

Ci contre quelques informations tirees du journal officiel de l'UE.

La loi prevoit la libre et juste circulation de l'information pour la protection du public et des fabricants.

(10) Le grand public s'intéresse à la corrélation entre l'alimentation et la santé ainsi qu'au choix d'un régime alimentaire approprié correspondant aux besoins individuels. Le livre blanc de la Commission du 30 mai 2007 intitulé *Une stratégie européenne pour les problèmes de santé liés à la nutrition, la surcharge pondérale et l'obésité* (ci-après dénommé «livre blanc de la Commission») indiquait que l'étiquetage nutritionnel était une méthode importante pour informer les consommateurs de la composition des denrées alimentaires et pour les aider à choisir en connaissance de cause. La communication de la Commission du 13 mars 2007 intitulée *Stratégie communautaire en matière de politique des consommateurs pour la période 2007-2013 – Responsabiliser le consommateur, améliorer son bien-être et le protéger efficacement* soulignait que cette possibilité pour le consommateur de choisir en connaissance de cause était essentielle pour assurer aussi bien une véritable concurrence que le bien-être des consommateurs. Une connaissance des principes de base de la nutrition et des informations nutritionnelles adéquates sur les denrées alimentaires aideraient de manière appréciable les consommateurs à faire de tels choix. Les campagnes d'éducation et d'information sont des mécanismes importants pour améliorer la compréhension, par le consommateur, de l'information relative aux denrées alimentaires.

- (20) La législation relative à l'information sur les denrées alimentaires devrait interdire d'utiliser des informations susceptibles d'induire en erreur le consommateur, en particulier en ce qui concerne les caractéristiques, les effets ou les propriétés des denrées alimentaires, ou d'attribuer aux denrées alimentaires des vertus médicinales. Pour être efficace, cette interdiction devrait également s'appliquer à la publicité faite à l'égard des denrées alimentaires et à leur présentation.
- (36) Le livre blanc de la Commission a mis en évidence certains éléments nutritionnels importants pour la santé publique, dont les acides gras saturés, les sucres ou le sodium. Il convient donc que les exigences régissant les informations nutritionnelles à fournir obligatoirement prennent en considération ces éléments.

Journal officiel de l'Union européenne

RÈGLEMENT (UE) N° 1169/2011 DU PARLEMENT EUROPÉEN ET DU CONSEIL
du 25 octobre 2011

concernant l'information des consommateurs sur les denrées alimentaires, modifiant les règlements (CE) n° 1924/2006 et (CE) n° 1925/2006 du Parlement européen et du Conseil et abrogeant la directive 87/250/CEE de la Commission, la directive 90/496/CEE du Conseil, la directive 1999/10/CE de la Commission, la directive 2000/13/CE du Parlement européen et du Conseil, les directives 2002/67/CE et 2008/5/CE de la Commission et le règlement (CE) n° 608/2004 de la Commission

(Texte présentant de l'intérêt pour l'EEE)

(37) Étant donné qu'un des objectifs du présent règlement est de fournir au consommateur final les bases pour décider en connaissance de cause, il importe de faire en sorte à cet égard que le consommateur final comprenne facilement les informations qui figurent sur l'étiquetage. Il est donc approprié que le terme «sel» soit utilisé sur l'étiquetage de préférence au terme correspondant au nutriment «sodium».

SITE REFERENCE POUR LE DATASET A UTILISER :

<https://fr.openfoodfacts.org/>

Le site propose toute une gamme d'informations nutritionnelles sur divers produits

The screenshot displays the Open Food Facts website. At the top, there's a donation banner asking for support, followed by a navigation bar with links like 'Recherche avancée', 'Graphiques et cartes', 'Découvrir', 'Contribuer', and 'Installz l'application mobile'. Below this is the main content area.

Open Food Facts - France

Découvrir

Open Food Facts est une base de données sur les produits alimentaires faite par tout le monde, pour tout le monde. Elle vous permet de faire des choix plus informés, et comme les données sont ouvertes (open data), tout le monde peut les utiliser pour tout usage.

→ En savoir plus sur Open Food Facts

Ajouter un produit

Pages : 1 2 3 4 ... 8844 8845 8846 Suivant (100 produits par page)

Classer les 100 produits ci-dessous suivant vos préférences

Faire un don

J'ai déjà fait un don ou je ne suis pas intéressé. Cacher la bannière.

Contribuer

Open Food Facts est un projet citoyen à but non lucratif créé par des milliers de volontaires à travers le monde. Vous pouvez commencer à contribuer en ajoutant un produit de votre cuisine, et nous avons plein de projets enthousiasmants auxquels vous pouvez participer de beaucoup de façons différentes.

→ Comment contribuer ?

Produits les plus scannés

884 569 produits

Modifier vos préférences alimentaires

Photo avec code-barres

Classement des produits :

- Eaux de sources - Cristaline - 1 l
- Prince Chocolat - LU - 300 g e
- Nutella pâte à tartiner aux noisettes et au cacao 1kg - Ferrero
- Sésame - Gerblé - 230 g
- Excellence 70% Cacao Noir Intense - Lindt - 100 g e
- Cruel Mélange de noix - QUAKER - 450 g
- Muesli Raisin, Figue, Abricot - Bjorg - 375 g
- Nocciolata Pâte à tartiner au cacao et noisettes - Rigoni di Asiago - 270 g
- Pur beurre de cacahuète - Léa Nature - 350 g
- Fournés Chocolat noir BIO - bjorg - 225 g
- 100% mie Complet - Harrys - 500 g
- Chocapic - Nestlé - 430 g
- Cristaline Eau de source - 50 cl
- Biscuit pomme noisette - Gerblé - 230 g
- Wasa tartine croustillante fibres - 230 g
- Special Muesli 30% fruits & noix - Jordans - 750 g
- Lindt Excellence 65% cacao - 100 g
- Coca-cola - bevande alla coca - 330 ml
- Flocons d'avoine - Bjorg - 500 g
- Granola - L'original - chocolat au lait - LU - 200 g e
- Coca Cola Sans sucres - Cacacola - 330 ml
- NESQUIK Cacao - Nestlé - 1000 g
- Volvic - 1,5 L
- Primevère Bio doux Tartine & Cuisson - 250 g
- S PELLEGRINO eau minérale naturelle gazeuse 1L - San Pellegrino - 1000 ml
- Biscottes 6 Céréales - Heudebert - 300 g e
- NESTLE DESSERT Noir 205g - 205 g
- Pains au lait - Pasquier - 350 g
- Tropicana 100% oranges pressées sans pulpe 1 L - 1000 ml
- HEPAR eau minérale naturelle 1L - 1000 ml

Exemple d'un affichage de données pour un produit choisi :

Yaourt soja végétal à la vanille - Alpro - 500 g

Code-barres: 541188103387 (EAN / EAN-13)

La page de ce produit n'est pas complète. Vous pouvez aider à la compléter en l'éditant et en ajoutant plus de données à partir des photos que nous avons, ou en prenant plus de photos à l'aide de l'application pour **Android** ou **iPhone / iPad**. Merci! X

Nutri-Score A
Très bonne qualité nutritionnelle



NOVA 4
Aliments ultra-transformés



NOVA 4
Aliments ultra-transformés



Éco-Score B
Faible impact environnemental



Choisissez quelles informations vous préférez voir en premier. [Modifier vos préférences alimentaires](#)

Caractéristiques du produit

Dénomination générique : Fermented soya product, vanilla, with added calcium and vitamins

Quantité : 500 g

Conditionnement : Pot, en:Card-sleeve, en:Pp-tub, Papier plastique

Marques : Alpro

Catégories : Aliments et boissons à base de végétaux, Aliments d'origine végétale, Desserts, Produits fermentés, Desserts végétaliens, Produits végétaux fermentés, Yaourts végétaux, Yaourts au soja, Desserts au soja, Dessert au soja aromatisé, Desserts de soja à la vanille

Labels, certifications, récompenses : Peu ou pas de matière grasse, Peu de matière grasse, Végétarien, Sans gluten, Végétalien, 100% végétal, Source de calcium, Contient du soja, Ne pas congeler, Riche en protéines, Sans OGM, Sans lactose, en:The Vegan Society, Sans ajout de produit laitiers



Lien vers la page du produit sur le site officiel du fabricant : <https://www.alpro.com/uk/products/plant-...>

Magasins : Magasins U, Delhaize, E.Leclerc, Monoprix, Sainsbury's, Carrefour, carrefour.fr

Pays de vente : Belgique, France, Allemagne, Hongrie, Italie, Pays-Bas, Portugal, Espagne, Suisse, Royaume-Uni



Exemple d'un affichage de données pour un produit choisi :

Ingrédients

→ Les ingrédients sont listés par ordre d'importance (quantité).

Liste des ingrédients:
Base de **soja** (eau, fèves de soja sans OGM décortiquées (9,9%), sucre, stabilisant (pectines), phosphate tricalcique, correcteurs d'acidité (acide citrique, citrates de sodium), extrait de carotte, sel marin de table, arôme naturel de vanille, arômes, vanille en poudre (0,03%), antioxygènes (extrait riche en tocophérols, esters d'acides gras de l'acide ascorbique), vitamines (B2, B12, D2), ferments de yaourt (*S. thermophilus, L. bulgaricus*)).

Substances ou produits provoquant des allergies ou intolérances : [Soja](#)

Traces éventuelles : [Fruits à coque](#)

Analyse des ingrédients :

 Pourrait contenir de l'huile de palme  Végétalien  Végétarien

→ L'analyse est basée uniquement sur les ingrédients listés et ne prend pas en compte les méthodes de fabrication.

[Détail de l'analyse des ingrédients » Nous avons besoin de votre aide !](#)

Additifs : <ul style="list-style-type: none">• E440 - Pectines• E330 - Acide citrique• E331 - Citrates de sodium• E306 - Extrait riche en tocophérols• E304 - Acide palmityle-6-L-ascorbique	Vitamines ajoutées : <ul style="list-style-type: none">• Riboflavine• Vitamine B12• Ergocalciférol	Minéraux ajoutés : <ul style="list-style-type: none">• Calcium• Phosphate de calcium
---	---	--

INGRÉDIENTS : base de **soja** (eau, fèves de **soja** sans OGM décortiquées (9,9%)), sucre, stabilisant (pectines), phosphate tricalcique, correcteurs d'acidité (acide citrique, citrates de sodium), extrait de carotte, sel marin de table, arôme naturel de vanille, arômes, vanille en poudre (0,03%), antioxygènes (extrait riche en tocophérols, esters d'acides gras de l'acide ascorbique), vitamines (B2, B12, D2), ferments de yaourt (*S. thermophilus, L. bulgaricus*)).

Peut contenir des traces de fruits à coque (pas d'arachides).
Sans produits laitiers, ni gluten. PO00787

Suggestion de... ©

Exemple d'un affichage de données pour un produit choisi :

NOVA
4

4 - Produits alimentaires et boissons ultra-transformés

Informations nutritionnelles

Note nutritionnelle de couleur NutriScore 



[Détail du calcul du Nutri-Score »](#)

⚠ Avertissement : Le taux de fruits, légumes et noix n'est pas indiqué sur l'étiquette, il a été estimé en fonction de la liste des ingrédients : 0%

Comparaison avec les valeurs moyennes des produits de même catégorie :

- Desserts de soja à la vanille (27 produits)
- Dessert au soja aromatisé (91 produits)
- Yaourts au soja (193 produits)
- Desserts au soja (210 produits)
- Yaourts végétaux (316 produits)
- Produits végétaux fermentés (326 produits)
- Desserts végétaliens (637 produits)
- Desserts (18720 produits)
- Produits fermentés (32664 produits)
- Aliments d'origine végétale (105416 produits)
- Aliments et boissons à base de végétaux (121579 produits)

Repères nutritionnels pour 100 g 

-  2.2 g Matières grasses en faible quantité
-  0.4 g Acides gras saturés en faible quantité
-  7.5 g Sucres en quantité modérée
-  0.2 g Sel en faible quantité

Valeurs nutritionnelles moyennes pour 100 g	
Energie	280 kJ / 66 kcal
Matières grasses	2,2 g
dont acides gras saturés	0,4 g
Glucides	7,5 g
dont sucres	7,5 g
Fibres alimentaires	0,9 g
Protéines	3,7 g
Sel	0,20 g
Vitamines :	
• D	0,75 µg*
• riboflavine (B2)	0,21 mg*
• B12	0,38 µg*
Sels minéraux : calcium	120 mg*
* = 15% des valeurs nutritionnelles de référence	

Exemple d'un affichage de données pour un produit choisi :

Détails du calcul de l'Éco-score »

Conditionnement

Instruction de recyclage et/ou informations d'emballage :
21 PAP, 41 ALU

Parties de l'emballage :

Nombre	Forme	Matière	Recyclage
Étui	Carton non ondulé		
Pot	Aluminium		
	PP - Polypropylène		

Sources de données

–

Produit ajouté le 11 mars 2016 à 10:09:56 CET par [openfoodfacts-contributors](#)

● Différence en % ○ valeur pour 100 g/ 100 ml
→ À noter : pour chaque nutriment, la moyenne n'est pas celle de tous les produits de la catégorie, mais des produits pour lesquels la quantité du nutriment est connue.

Informations nutritionnelles	Tel que vendu pour 100 g / 100 ml	Comparé à: Desserts de soja à la vanille
Énergie	280 kJ (66 kcal)	-24 %
Matières grasses	2,2 g	+24 %
Acides gras saturés	0,4 g	+28 %
Glucides	7,5 g	-48 %
Sucre	7,5 g	-27 %
Fibres alimentaires	0,9 g	+27 %
Protéines	3,7 g	+20 %
Sel	0,2 g	+47 %
Vitamine D	0,75 µg	
Vitamine B2 (Riboflavine)	0,21 mg	
Vitamine B12 (cobalamine)	0,38 µg	
Calcium	120 mg	-0 %
Fruits, légumes, noix et huiles de colza, noix et olive (estimation par analyse de la liste des ingrédients)	0 %	

Impact environnemental

Eco-score ●

L'Eco-Score est un score expérimental qui synthétise les impacts environnementaux des produits alimentaires.
→ La formule de l'Eco-Score est susceptible d'évoluer car elle est régulièrement améliorée pour la rendre plus précise.

Apercu et definition du Nutriscore :

How to calculate Nutriscore					Nutriscore = Total points N - total points P			
Negative Points					Positive Points			
Points	Energy (kJ/100g)	Saturated fat (g/100g)	Sugars (g/100g)	Sodium (mg/100g)	Points	Fruits, vegetables, pulses, nuts (g/100g) Method AOAC	Fibres (g/100g)	Proteins (g/100g)
0	≤ 335	≤ 1	≤ 4,5	≤ 90	0	≤ 40	≤ 0,9	≤ 1,6
1	> 335	> 1	> 4,5	> 90	1	> 40	> 0,9	> 1,6
2	> 670	> 2	> 9	> 160	2	> 60	> 1,9	> 3,2
3	> 1005	> 3	> 13,5	> 270	3	-	> 2,8	> 4,8
4	> 1340	> 4	> 18	> 360	4	-	> 3,7	> 6,4
5	> 1675	> 5	> 22,5	> 450	5	> 80	> 4,7	> 8
6	> 2010	> 6	> 27	> 540				
7	> 2345	> 7	> 31	> 630				
8	> 2680	> 8	> 36	> 720				
9	> 3015	> 9	> 40	> 810				
10	> 3350	> 10	> 45	> 900				

¹: la teneur en sodium correspond à la teneur en sel mentionnée sur la déclaration obligatoire d'au moins 2 %.

²: les fruits et légumes, légumineuses et fruits à coque comprennent de nombreuses vitamines (en particulier les vitamines E, C, B1, B2, B3, B6 et B9 ainsi que la provitamine A).

Class	Score limit	Color
A	Min à +1	Vert foncé
B	0 à 2	Vert clair
C	3 à 10	Orange clair
D	11 à 18	Orangé moyen
E	19 à Max	Orange foncé

Commentaires sur le site OpenFood :

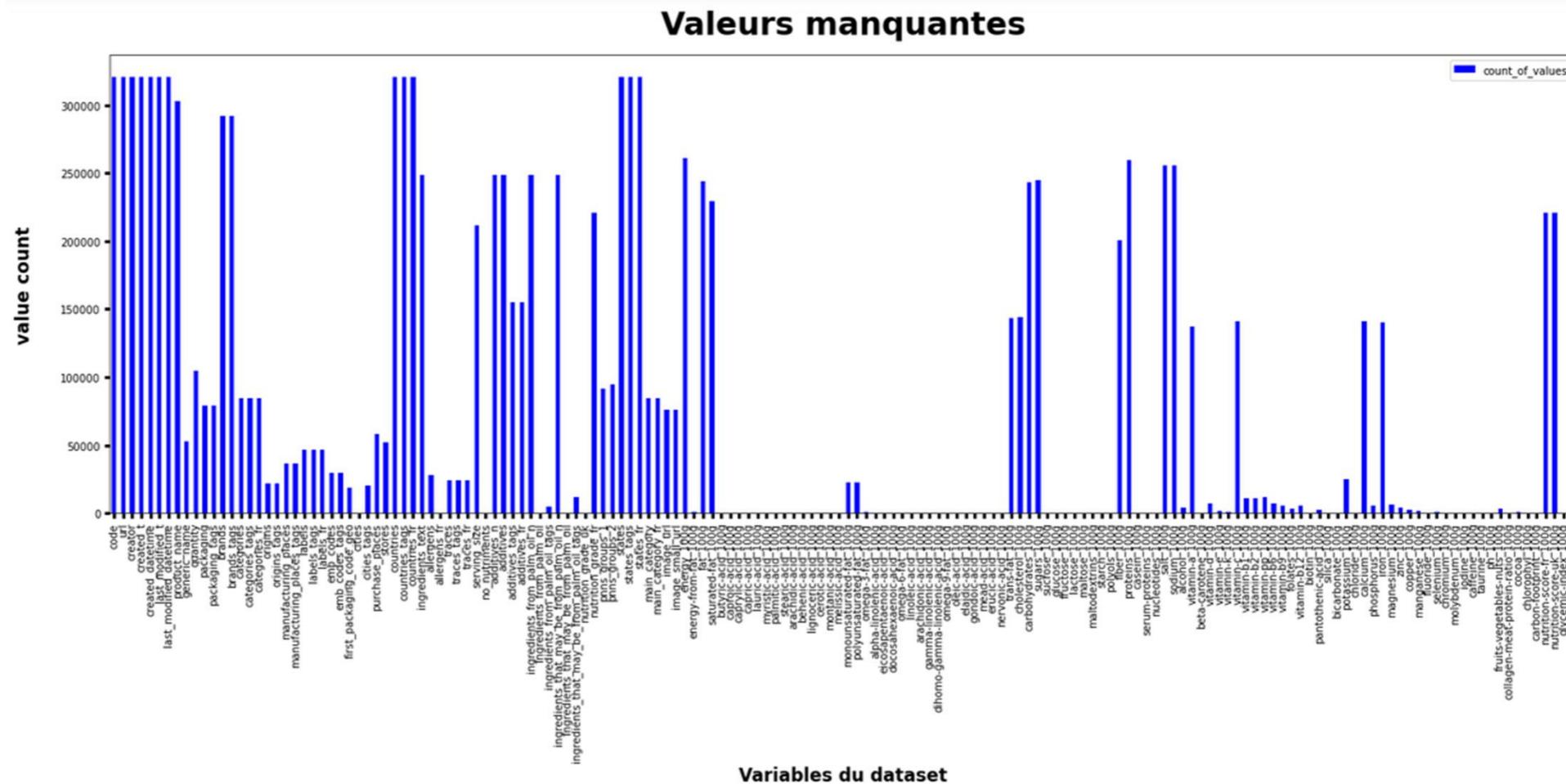
- Le site fourni un panel assez large d'informations nutritionnelles sur divers produits.
- Le site permet a toute source/personne d'introduire des donnees dans la database.
- Le processus de validation des donnees dans la database n'est pas specifie sur le site.
- Toute information et/ou produit non disponibles en mode input ne sont pas fourni au utilisateurs de l'application.
- Les valeurs du nutriscore et Nova sont calculees au niveau du dataset.

Methodologies et etapes de travail dans le projet :

- Chargement du dataset dans un notebook jupyter.
- Prise de connaissances du contenu et du type de l'ensemble des colonnes.
- Estimation des valeurs manquantes/aberrantes
- Elimination des colonnes nulles ou a faible ratio de disponibilite de data
- Fusionner les colonnes contenant des data similaires pour la reduction du nombre de colonnes
- Traitement des lignes en double

```
In [11]: # Determination du ratio de remplissage
nombre_data = 320772 * 162
Ratio = df.isna().sum().sum()/nombre_data
print('Ratio de remplissage du dataset = ',(1-Ratio)*100, '%')
```

Ratio de remplissage du dataset = 23.77842736804866 %



Apercu et details du dataset fourni pour le projet :

Col_num	variable	resultat notebook					ratio disponibilite	DATA	description data			Limites	
		count_of_values	Null_value	dtype	Zero_values				description	specificité	format	anomalie dans le dataset	min max
0	code	320749	23	object	0		99.9928	G	numero reference du produit	unique	string	doublon / valeur manquante	
1	url	320749	23	object	0		99.9928	G	adresse url du produit	unique	string	valeur manquante	
2	creator	320770	2	object	0		99.9994	G	nom du la source upload du produit	Not	string	valeur manquante	
3	created_t	320769	3	object	1		99.9991	G	date de creation	Not	float	valeur manquante	
4	created_datetime	320763	9	object	0		99.9972	G	date de creation	Not	Date	valeur manquante	
5	last_modified_t	320772	0	object	0		100.0000	S	date de modification	Not	float	RAS	
6	last_modified_datetime	320772	0	object	0		100.0000	S	date de modification	Not	Date	RAS	
7	product_name	303010	17762	object	0		94.4627	G	Nom du produit	Not	string	valeur manquante / differente langues / image	
8	generic_name	52795	267977	object	0		16.4587	S	Nom generic	Not	string	valeur manquante / differente langues	
9	quantity	104819	215953	object	0		32.6771	G	quantite du produit	Not	float / object	valeur manquante /	
10	packaging	78960	241812	object	0		24.6156	G	type d'emballage	Not	string	valeur manquante / differente langues	
11	packaging_tags	78961	241811	object	0		24.6159	S	type d'emballage	Not	string	valeur manquante / differente langues	
12	brands	292360	28412	object	0		91.1426	G	nom de marque	Not	string	valeur manquante / differente langues	
13	brands_tags	292352	28420	object	0		91.1401	S	nom de marque	Not	string	valeur manquante / differente langues	
14	categories	84410	236362	object	0		26.3146	G	type de produit	Not	string	valeur manquante / differente langues	
15	categories_tags	84389	236383	object	0		26.3081	S	type de produit	Not	string	valeur manquante / differente langues	
16	categories_fr	84411	236361	object	0		26.3150	S	type de produit	Not	string	valeur manquante / Traduction fr de la colonne au dessus	
17	origins	22190	298582	object	0		6.9177	G	pays d'origine	Not	string	valeur manquante / differente langues	
18	origins_tags	22153	298619	object	0		6.9062	S	pays ou lieu d'origine	Not	string	valeur manquante / differente langues	
19	manufacturing_places	36501	284271	object	0		11.3791	G	pays de confection	Not	string	valeur manquante / differente langues / adresse complete	
20	manufacturing_places_tags	36495	284277	object	0		11.3772	S	pays de confection	Not	string	valeur manquante / differente langues / adresse complete	
21	labels	46559	274213	object	0		14.5147	G	description specifique	Not	string	valeur manquante / differente langues	
22	labels_tags	46644	274128	object	0		14.5412	S	description specifique	Not	string	valeur manquante / differente langues	
23	labels_fr	46666	274106	object	0		14.5480	S	description specifique en fr	Not	string	valeur manquante / differente langues	
24	emb codes	29306	291466	object	0		9.1361	G	code emballeur officiel 'Insee'	Not	string	valeur manquante /	
25	emb_codes_tags	29303	291469	object	0		9.1351	S	code emballeur officiel 'Insee'	Not	string	valeur manquante /	
26	first_packaging_code_geo	18803	301969	object	0		5.8618	S	code geo / coordonnes	Not	string	valeur manquante /	
27	cities	23	320749	object	0		0.0072	G	valeurs 'a', 'b' et 'c' / sens ?	Not	string	Non utilisable	
28	cities_tags	20320	300452	object	0		6.3347	S	nom de commune /ville/ pays	Not	string	valeur manquante /	
29	purchase_places	58193	262579	object	0		18.1415	G	lieu de vente	Not	string	valeur manquante /	
30	stores	51722	269050	object	0		16.1242	G	point de vente	Not	string	valeur manquante /	
31	countries	320492	280	object	0		99.9127	G	possibilite : pays d'origine	Not	string	valeur manquante / differente langues	
32	countries_tags	320492	280	object	0		99.9127	S	possibilite : pays d'origine	Not	string	valeur manquante / differente langues	
33	countries_fr	320492	280	object	0		99.9127	S	possibilite : pays d'origine en francais	Not	string	valeur manquante /	
34	ingredients_text	248962	71810	object	0		77.6134	G	liste d'ingredients	Not	string	valeur manquante / differente langues	
35	allergens	28344	292428	object	0		8.8362	S	liste produit considere allergens	Not	string	valeur manquante / differente langues	
36	allergens_fr	19	320753	object	0		0.0059	S	url liste d'image par ligne	Not	string	Non utilisable	
37	traces	24353	296419	object	0		7.5920	S	liste d'ingredients	Not	string	valeur manquante / differente langues	
38	traces_tags	24329	296443	object	0		7.5845	S	liste d'ingredients	Not	string	valeur manquante / differente langues	

Col_num	variable	resultat notebook						ratio disponibilite	DATA	description data				Limites
		count_of_values	Null_value	dtype	Zero_values					description	specificité	format	anomalie dans le dataset	
39	traces_fr	24352	296420	object	0		7.5917	S	liste d'ingredients / en fr	Not	string	valeur manquante		
40	serving_size	211331	109441	object	0		65.8820	G	chiffre + unite + description	Not	string	valeur manquante / differente langues / non utilisable directement pour calcul		
41	no_nutriments	0	320772	float64	0		0.0000	S	colonne nulle					
42	additives_n	248939	71833	float64	94259		77.6062	S	nombre d'additifs	Not	float	valeur manquante		
43	additives	248905	71867	object	0		77.5956	S	Liste d'additifs par nom	Not	Liste	valeur manquante / differente langues		
44	additives_tags	154680	166092	object	0		48.2212	S	Liste d'additifs par code (e....)	Not	string	valeur manquante		
45	additives_fr	154680	166092	object	0		48.2212	S	Liste d'additifs par code & nom en fr	Not	string	valeur manquante		
46	Ingredients_from_palm_oil_n	248939	71833	float64	244104		77.6062	S	Nombre Ingeredients a partir d'huile de palme	Not	string	valeur manquante		
47	ingredients_from_palm_oil	0	320772	float64	0		0.0000	S	colonne nulle					
48	ingredients_from_palm_oil_tags	4835	315937	object	0		1.5073	S	Reference confirme a l'huile de palme	Not	string	valeur manquante		
49	ingredients_that_may_be_from_palm_oil_n	248939	71833	float64	237243		77.6062	S	Reference en nombre probable a l'huile de palme	Not	float	valeur manquante		
50	ingredients_that_may_be_from_palm_oil	0	320772	float64	0		0.0000	S	colonne nulle					
51	ingredients_that_may_be_from_palm_oil_tags	11696	309076	object	0		3.6462	S	Reference en nom probable a l'huile de palme	Not	string	valeur manquante		
52	nutrition_grade_uk	0	320772	float64	0		0.0000	S	colonne nulle					
53	nutrition_grade_fr	221210	99562	object	0		68.9618	S	nutriscore de a-b-c-d-e	Not	string	valeur manquante	-15 40	
54	pnns_groups_1	91513	229259	object	0		28.5290	G	categorie du produit	Not	string	valeur manquante / differente langues		
55	pnns_groups_2	94491	226281	object	0		29.4574	G	categorie du produit	Not	string	valeur manquante / differente langues		
56	states	320726	46	object	0		99.9857	G	statut de chargement de donnees	Not	string	valeur manquante / differente langues		
57	states_tags	320726	46	object	0		99.9857	G	statut de chargement de donnees	Not	string	valeur manquante / differente langues		
58	states_fr	320726	46	object	0		99.9857	G	statut de chargement de donnees en fr	Not	string	valeur manquante		
59	main_category	84366	236406	object	0		26.3009	G	type de produit	Not	string	valeur manquante / differente langues		
60	main_category_fr	84366	236406	object	0		26.3009	G	type de produit en fr	Not	string	valeur manquante		
61	image_url	75836	244936	object	0		23.6417	G	image produit	unique	string	valeur manquante		
62	image_small_url	75836	244936	object	0		23.6417	G	Image produit	unique	string	valeur manquante		
63	energy_100g	261113	59659	float64	8909		81.4014	G	valeur energetique	Not	float	valeur manquante	0 900	
64	energy-from-fat_100g	857	319915	float64	164		0.2672	S	valeur energetique	Not	float	valeur manquante	0 100	
65	fat_100g	243891	76881	float64	64504		76.0325	S	quantite de gras par 100g	Not	float	valeur manquante	0 100	
66	saturated-fat_100g	229554	91218	float64	68736		71.5630	S	quantite de gras saturee par 100 g	Not	float	valeur manquante	0 100	
67	butyric-acid_100g	0	320772	float64	0		0.0000	S					0 100	
68	caproic-acid_100g	0	320772	float64	0		0.0000	S					0 100	
69	caprylic-acid_100g	1	320771	float64	0		0.0003	S					0 100	
70	capric-acid_100g	2	320770	float64	0		0.0006	S					0 100	
71	lauric-acid_100g	4	320768	float64	0		0.0012	S					0 100	
72	myristic-acid_100g	1	320771	float64	0		0.0003	S					0 100	
73	palmitic-acid_100g	1	320771	float64	0		0.0003	S					0 100	
74	stearic-acid_100g	1	320771	float64	0		0.0003	S					0 100	
75	arachidic-acid_100g	24	320748	float64	0		0.0075	S					0 100	
76	behenic-acid_100g	23	320749	float64	0		0.0072	S					0 100	
77	lignoceric-acid_100g	0	320772	float64	0		0.0000	S					0 100	
78	cerotic-acid_100g	0	320772	float64	0		0.0000	S					0 100	
79	montanic-acid_100g	1	320771	float64	0		0.0003	S					0 100	
80	melissic-acid_100g	0	320772	float64	0		0.0000	S					0 100	
81	monounsaturated-fat_100g	22823	297949	float64	5757		7.1150	S	quantite de gras monosature par 100 g	Not	float	valeur manquante	0 100	
82	polyunsaturated-fat_100g	22859	297913	float64	6320		7.1262	S	quantite de gras polysature par 100 g	Not	float	valeur manquante	0 100	

quantite de divers acid

representativite extremement faible /
Les valeurs existantes restent utiles

Col_num	resultat notebook						ratio disponibilite	DATA	description data				Limites	
	variable	count_of_values	Null_value	dtype	Zero_values				description	specificité	format	anomalie dans le dataset	min	max
83	omega-3-fat_100g	841	319931	float64	5		0.2622	S					0	100
84	alpha-linolenic-acid_100g	186	320586	float64	1		0.0580	S					0	100
85	eicosapentaenoic-acid_100g	38	320734	float64	0		0.0118	S					0	100
86	docosahexaenoic-acid_100g	78	320694	float64	0		0.0243	S					0	100
87	omega-6-fat_100g	188	320584	float64	0		0.0586	S					0	100
88	linoleic-acid_100g	149	320623	float64	0		0.0465	S					0	100
89	arachidonic-acid_100g	8	320764	float64	0		0.0025	S					0	100
90	gamma-linolenic-acid_100g	24	320748	float64	0		0.0075	S					0	100
91	dihomo-gamma-linolenic-acid_100g	23	320749	float64	0		0.0072	S					0	100
92	omega-9-fat_100g	21	320751	float64	0		0.0065	S					0	100
93	oleic-acid_100g	13	320759	float64	0		0.0041	S					0	100
94	elaidic-acid_100g	0	320772	float64	0		0.0000	S					0	100
95	gondoic-acid_100g	14	320758	float64	0		0.0044	S					0	100
96	mead-acid_100g	0	320772	float64	0		0.0000	S					0	100
97	erucic-acid_100g	0	320772	float64	0		0.0000	S					0	100
98	nervonic-acid_100g	0	320772	float64	0		0.0000	S					0	100
99	trans-fat_100g	143298	177474	float64	140297		44.6729	S	quantite de gras non sature par 100 g	Not	float	valeur manquante / valeur negative	0	100
100	cholesterol_100g	144090	176682	float64	89441		44.9198	S	quantite de chlosterol par 100 g	Not	float	valeur manquante	0	100
101	carbohydrates_100g	243588	77184	float64	21607		75.9380	S	quantite de carbohydrates par 100 g	Not	float	valeur manquante	0	100
102	sugars_100g	244971	75801	float64	37077		76.3692	S	quantite de sucre par 100 g	Not	float	valeur manquante	0	100
103	sucrose_100g	72	320700	float64	8		0.0224	S					0	100
104	glucose_100g	26	320746	float64	4		0.0081	S					0	100
105	fructose_100g	38	320734	float64	4		0.0118	S					0	100
106	lactose_100g	262	320510	float64	121		0.0817	S					0	100
107	maltose_100g	4	320768	float64	0		0.0012	S	quantite de divers sucres			representativite extremement faible		
108	maltodextrins_100g	11	320761	float64	0		0.0034	S					0	100
109	starch_100g	266	320506	float64	24		0.0829	S					0	100
110	polyols_100g	414	320358	float64	14		0.1291	S					0	100
111	fiber_100g	200886	119886	float64	68833		62.6258	S	quantite de fibres par 100 g	Not	float	valeur manquante	0	100
112	proteins_100g	259922	60850	float64	53631		81.0301	S	quantite de protein par 100 g	Not	float	valeur manquante	0	100
113	casein_100g	27	320745	float64	0		0.0084	S					0	100
114	serum-proteins_100g	16	320756	float64	0		0.0050	S	quantite protein / nucleotide			representativite extremement faible		
115	nucleotides_100g	9	320763	float64	0		0.0028	S					0	100
116	salt_100g	255510	65262	float64	34174		79.6547	S	quantite de sel par 100 g	Not	float	valeur manquante	0	100
117	sodium_100g	255463	65309	float64	34131		79.6401	S	quantite de sodium par 100 g	Not	float	valeur manquante	0	100
118	alcohol_100g	4133	316639	float64	1555		1.2885	S	quantite d'alcool par 100 g	Not	float	valeur manquante	0	100
119	vitamin-a_100g	137554	183218	float64	78620		42.8822	S	quantite en vitamine par 100 g	Not	float	valeur manquante	0	100
120	beta-carotene_100g	34	320738	float64	2		0.0106	S	quantite precurseur de vitamine A par 100 g	Not	float	valeur manquante	0	100
121	vitamin-d_100g	7057	313715	float64	345		2.2000	S	quantite en vitamine par 100 g	Not	float	valeur manquante	0	100
122	vitamin-e_100g	1340	319432	float64	16		0.4177	S	quantite de vitamines			representativite extremement faible	0	100
123	vitamin-k_100g	918	319854	float64	46		0.2862	S					0	100

quantite divers acid et gras
(omega 3 et 9)

representativite extremement faible

quantite de divers sucres

representativite extremement faible

quantite protein / nucleotide

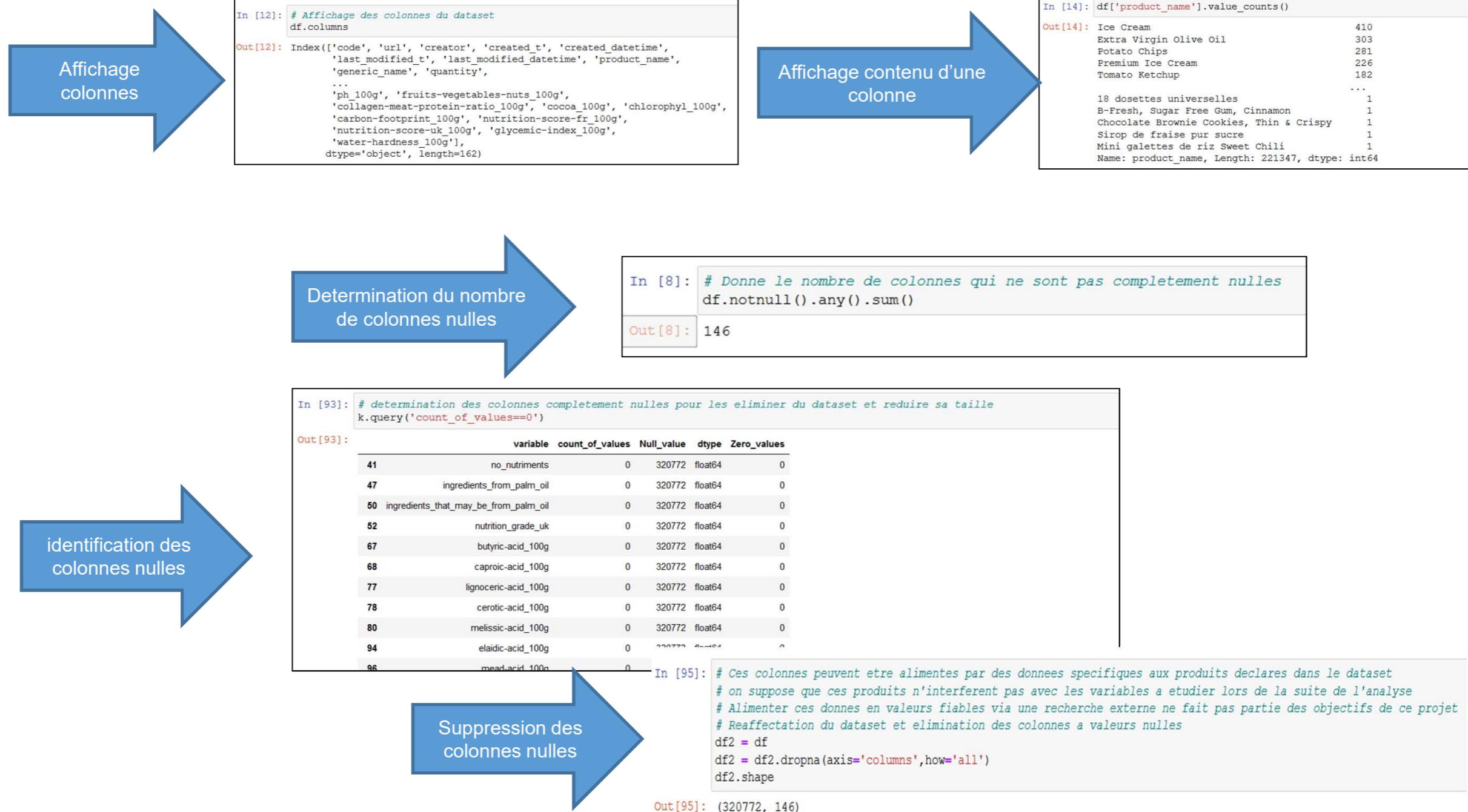
representativite extremement faible

quantite de vitamines

representativite extremement faible

Col_num	variable	resultat notebook					ratio disponibilite	DATA	description data				Limites
		count_of_values	Null_value	dtype	Zero_values				description	specificité	format	anomalie dans le dataset	
123	vitamin-k_100g	918	319854	float64	46		0.2862	S	quantite de vitamines			representativite extremement faible	0 100
124	vitamin-c_100g	140867	179905	float64	90500		43.9150	S	quantite en vitamin par 100 g				0 100
125	vitamin-b1_100g	11154	309618	float64	328		3.4772	S	quantite en vitamin par 100 g				0 100
126	vitamin-b2_100g	10815	309957	float64	361		3.3716	S	quantite en vitamin par 100 g				0 100
127	vitamin-pp_100g	11729	309043	float64	244		3.6565	S	quantite en vitamin par 100 g				0 100
128	vitamin-b6_100g	6784	313988	float64	139		2.1149	S	quantite en vitamin par 100 g				0 100
129	vitamin-b9_100g	5240	315532	float64	13		1.6336	S	quantite en vitamin par 100 g				0 100
130	folates_100g	3042	317730	float64	124		0.9483	S	Vitamin				0 100
131	vitamin-b12_100g	5300	315472	float64	107		1.6523	S	Vitamin				0 100
132	biotin_100g	330	320442	float64	3		0.1029	S	Vitamin (H)				0 100
133	pantothenic-acid_100g	2483	318289	float64	97		0.7741	S	Vitamin B5				0 100
134	silica_100g	38	320734	float64	0		0.0118	S	Additif				0 100
135	bicarbonate_100g	81	320691	float64	0		0.0253	S	Additif				0 100
136	potassium_100g	24748	296024	float64	775		7.7151	S	Sel principal				0 100
137	chloride_100g	158	320614	float64	0		0.0493	S	Sel principal				0 100
138	calcium_100g	141050	179722	float64	51048		43.9720	S	Sel principal				0 100
139	phosphorus_100g	5845	314927	float64	41		1.8222	S	Sel principal				0 100
140	iron_100g	140462	180310	float64	42678		43.7887	S	Oligo essentiel				0 100
141	magnesium_100g	6253	314519	float64	76		1.9494	S	Sel principal				0 100
142	zinc_100g	3929	316843	float64	73		1.2249	S	Oligo essentiel				0 100
143	copper_100g	2106	318666	float64	5		0.6565	S	Oligo essentiel				0 100
144	manganese_100g	1620	319152	float64	494		0.5050	S	Oligo essentiel				0 100
145	fluoride_100g	79	320693	float64	1		0.0246	S	Oligo essentiel				0 100
146	selenium_100g	1168	319604	float64	25		0.3641	S	Oligo essentiel				0 100
147	chromium_100g	20	320752	float64	0		0.0062	S	Oligo essentiel				0 100
148	molybdenum_100g	11	320761	float64	0		0.0034	S	Oligo essentiel				0 100
149	iodine_100g	259	320513	float64	3		0.0807	S	Oligo essentiel				0 100
150	caffeine_100g	78	320694	float64	10		0.0243	S	Stimulant				0 100
151	taurine_100g	29	320743	float64	0		0.0090	S	acid - amine soufre				0 100
152	ph_100g	49	320723	float64	1		0.0153	S	Potentiel Hydrogene				0 14
153	fruits-vegetables-nuts_100g	3036	317736	float64	962		0.9465	S	fruits & legumes a coques				0 100
154	collagen-meat-protein-ratio_100g	165	320607	float64	0		0.0514	S	ratio de collagene dans la viande				0 100
155	cocoa_100g	948	319824	float64	0		0.2955	S	quantite de cacao				0 100
156	chlorophyl_100g	0	320772	float64	0		0.0000	S	taux de chlorophyle				0 100
157	carbon-footprint_100g	268	320504	float64	20		0.0835	G	indice carbone				
158	nutrition-score-fr_100g	221210	99562	float64	12763		68.9618	G	nutri-score Fr				-15 40
159	nutrition-score-uk_100g	221210	99562	float64	13588		68.9618	G	nutri-score Uk				-15 40
160	glycemic-index_100g	0	320772	float64	0		0.0000	S	indice de glycemie				
161	water-hardness_100g	0	320772	float64	0		0.0000	S	durete eau				

representativite faible a extremement faible



Suppression colonnes
jugees non pertinentes

```
In [103]: # suppression de la colonne 'cities' apres examination de son contenu
df2.drop('cities', axis=1, inplace=True)

C:\Users\AMC\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
    return super().drop()

In [104]: # suppression de la colonne 'allergens_fr' apres examination de son contenu et l'existance de la colonne 'allergens'

In [105]: df2.drop('allergens_fr', axis=1, inplace=True)
```

Merge des colonnes a
donnees similaires

```
In [131]: df2['product_name'].fillna(df2['generic_name'], inplace=True)

C:\Users\AMC\anaconda3\lib\site-packages\pandas\core\series.py:4463: S
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-a-view-versus-a-copy
    return super().fillna()

In [132]: # suppression de la colonne 'generic_name'
df2.drop('generic_name',axis=1, inplace=True)

C:\Users\AMC\anaconda3\lib\site-packages\pandas\core\frame.py:4308: Se
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-a-view-versus-a-copy
    return super().drop()
```

Exemple similaire :

Traitement des colonnes 'countries'....et 'origins'

```
In [134]: df2[['countries','countries_tags','countries_fr','origins','origins_tags']]
Out[134]:
```

	countries	countries_tags	countries_fr	origins	origins_tags
0	en:FR	en:france	France	NaN	NaN
1	US	en:united-states	États-Unis	NaN	NaN

Suppression lignes à
produit non identifiable

```
In [192]: # Elimination des lignes avec des items non identifiables

In [193]: df3.dropna(subset=['code','url','creator','product_name'],inplace=True)

<ipython-input-193-6d6a633bbf89>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
    df3.dropna(subset=['code','url','creator','product_name'],inplace=True)
```

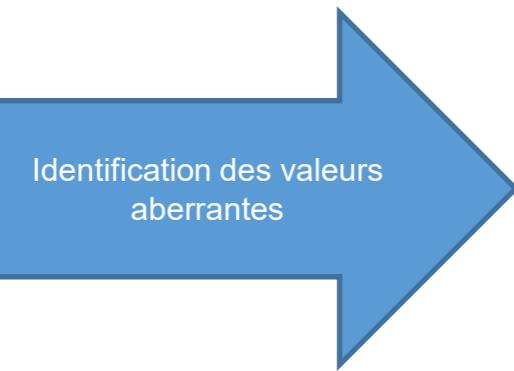
Elimination des doublons
pour le subset ['code']

```
In [203]: # suppression des doublons avec la configuration 'last'
# Un travail exhaustif impliquera la vérification de l'ensemble des data de chaque ligne avec comparaison aux data réel
<
>

In [204]: df3.drop_duplicates(subset=['code'],keep='last', inplace=True)

<ipython-input-204-5c075ca15e3d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
    df3.drop_duplicates(subset=['code'],keep='last', inplace=True)
```

Identification des valeurs aberrantes

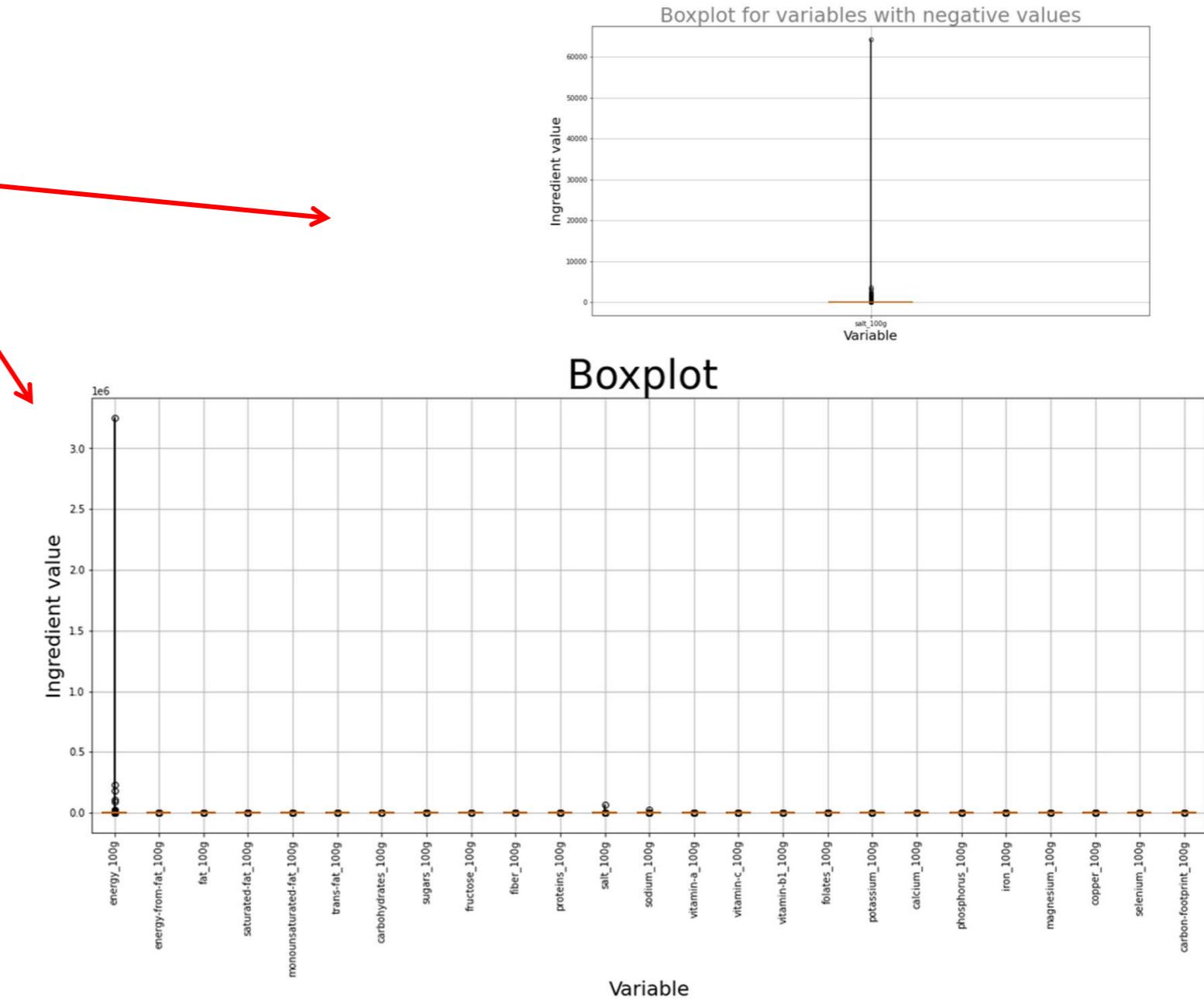


```
Out[20]: No issues      60
          excessive value  16
          Negative value   9
          Name: outliers type, dtype: int64
```

Out[18]:

	variable	outliers type	Min_	Mean	Max_	Outliers_count	dtype
0	additives_n	No issues	0.00	1.58	31.00	0	float64
1	ingredients_from_palm_oil_n	No issues	0.00	0.02	2.00	0	float64
2	ingredients_that_may_be_from_palm_oil_n	No issues	0.00	0.04	6.00	0	float64
3	energy_100g	excessive value	0.00	970.06	3251373.00	239877	float64
4	energy-from-fat_100g	excessive value	0.00	1.56	3740.00	541	float64
5	fat_100g	excessive value	0.00	10.08	714.29	4	float64
6	saturated-fat_100g	excessive value	0.00	3.83	550.00	3	float64
7	caprylic-acid_100g	No issues	0.00	0.00	7.40	0	float64
8	capric-acid_100g	No issues	0.00	0.00	6.20	0	float64
9	lauric-acid_100g	No issues	0.00	0.00	49.30	0	float64
10	myristic-acid_100g	No issues	0.00	0.00	18.90	0	float64
24	gamma-linolenic-acid_100g	No issues	0.00	0.00	0.10	0	float64
25	omega-9-fat_100g	No issues	0.00	0.00	75.00	0	float64
26	oleic-acid_100g	No issues	0.00	0.00	76.00	0	float64
27	trans-fat_100g	Negative value	-3.57	0.03	369.00	4	float64
28	cholesterol_100g	No issues	0.00	0.01	95.24	0	float64
29	carbohydrates_100g	excessive value	0.00	25.44	2916.67	18	float64
30	sugars_100g	Negative value	-17.86	12.77	3520.00	7	float64
31	sucrose_100g	No issues	0.00	0.00	92.80	0	float64
32	glucose_100g	No issues	0.00	0.00	23.00	0	float64
33	fructose_100g	excessive value	0.00	0.00	101.00	1	float64
34	lactose_100g	No issues	0.00	0.01	74.50	0	float64
35	maltose_100g	No issues	0.00	0.00	39.20	0	float64

Example d'une
représentation en
box plot et l'effet
des valeurs
outliers sur la
consistance des
données



Reduction du Dataset à traiter en choisissant uniquement les variables pertinentes :

```
In [7]: df_head_01.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303001 entries, 0 to 303000
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   code              303001 non-null   object  
 1   url               303001 non-null   object  
 2   creator            303001 non-null   object  
 3   created_datetime   303001 non-null   object  
 4   last_modified_datetime 303001 non-null   object  
 5   product_name       303001 non-null   object  
 6   brands             289910 non-null   object  
 7   countries          302804 non-null   object  
 8   main_category      83437 non-null   object  
 9   energy_100g        303001 non-null   float64 
 10  fat_100g           303001 non-null   float64 
 11  cholesterol_100g  303001 non-null   float64 
 12  carbohydrates_100g 303001 non-null   float64 
 13  sugars_100g        303001 non-null   float64 
 14  fiber_100g         303001 non-null   float64 
 15  proteins_100g     303001 non-null   float64 
 16  salt_100g          303001 non-null   float64 
 17  vitamin-c_100g    303001 non-null   float64 
 18  nutrition-score-fr_100g 218427 non-null   float64 
 19  nutrition-score-uk_100g 218427 non-null   float64 
dtypes: float64(11), object(9)
memory usage: 46.2+ MB
```

```
In [9]: # Changement du format des dates en mode 'time'
```

```
df['created_datetime'] = pd.to_datetime(df['created_datetime'])
df['last_modified_datetime'] = pd.to_datetime(df['last_modified_datetime'])
```

```
In [11]: # Arrangement de la casse de l'ensemble des string en mode 'lower'
```

```
In [12]: df['product_name'] = df['product_name'].str.lower()
```

```
In [13]: df['creator'] = df['creator'].str.lower()
```

```
In [14]: df['brands'] = df['brands'].str.lower()
```

```
In [15]: df['countries'] = df['countries'].str.lower()
```

```
In [16]: df['main_category'] = df['main_category'].str.lower()
```

```
In [17]: df.head(2)
```

	code	url	creator	created_datetime	last_modified_datetime	product_name	brands	countries	main_category	energy_100g
0	3087	http://world-fr.openfoodfacts.org/produit/0000...	openfoodfacts-contributors	2016-09-17 09:17:46+00:00	2016-09-17 09:18:13+00:00	farine de blé noir	ferme ty rnao	en.fr	Nan	0.0
1	4530	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	2017-03-09 14:32:37+00:00	2017-03-09 14:32:37+00:00	banana chips sweetened (whole)	Nan	us	Nan	2243.0

Diversité des données du Dataset :

Out[19]:	colonne	unique values
0	code	302956
1	url	303001
2	creator	3369
3	created_datetime	173106
4	last_modified_datetime	163884
5	product_name	214487
6	brands	53649
7	countries	1350
8	main_category	3451
9	energy_100g	3968
10	fat_100g	3371
11	cholesterol_100g	535
12	carbohydrates_100g	5414
13	sugars_100g	4061
14	fiber_100g	1013
15	proteins_100g	2491
16	salt_100g	5569
17	vitamin-c_100g	1166
18	nutrition-score-fr_100g	55
19	nutrition-score-uk_100g	55

Out[26]:	variable	count_of_values	Null_value	dtype	Zero_values
6	brands	289910	13091	object	0
7	countries	302804	197	object	0
8	main_category	83437	219564	object	0
18	nutrition-score-fr_100g	218427	84574	float64	12605
19	nutrition-score-uk_100g	218427	84574	float64	13423

In [27]: # Remplacement des valeurs manquantes :

```
In [29]: df['countries'].fillna('not specified', inplace=True)
df['main_category'].fillna('not specified', inplace=True)
df['brands'].fillna('not specified', inplace=True)
```

Out[30]:	variable	count_of_values	Null_value	dtype	Zero_values
0	code	303001	0	object	0
1	url	303001	0	object	0
2	creator	303001	0	object	0
3	created_datetime	303001	0	datetime64[ns, UTC]	0
4	last_modified_datetime	303001	0	datetime64[ns, UTC]	0
5	product_name	303001	0	object	0
6	brands	303001	0	object	0
7	countries	303001	0	object	0
8	main_category	303001	0	object	0
9	energy_100g	303001	0	float64	54158
10	fat_100g	303001	0	float64	126588
11	cholesterol_100g	303001	0	float64	248404
12	carbohydrates_100g	303001	0	float64	84239
13	sugars_100g	303001	0	float64	98018
14	fiber_100g	303001	0	float64	172879
15	proteins_100g	303001	0	float64	99816
16	salt_100g	303001	0	float64	84420
17	vitamin-c_100g	303001	0	float64	252767
18	nutrition-score-fr_100g	218427	84574	float64	12605
19	nutrition-score-uk_100g	218427	84574	float64	13423

Traitement des donnes aberrantes du Dataset :

Utilisation de la fonction 'abs' qui represente la valeur absolue pour changer le signe des valeurs tout en les sauvegardant

```
In [44]: # utilisation de la fonction abs(), valeur absolue pour l'elimination du signe (-) et la possibilite de garder les valeurs
df[negative_list]=df[negative_list].abs()
df.head(2)
```

Remplacement des valeurs superieurs a 100mg par la moyenne :

```
In [53]: # Traitement des outliers
# Remplacement de l'ensemble des valeurs superieures a 100 par la moyenne de chaque colonne
for i in df_num_01.columns :
    df.loc[df[i]>100] = df[i].mean()

In [54]: # Check par rapport a une variable
df.loc[df['salt_100g']>100]

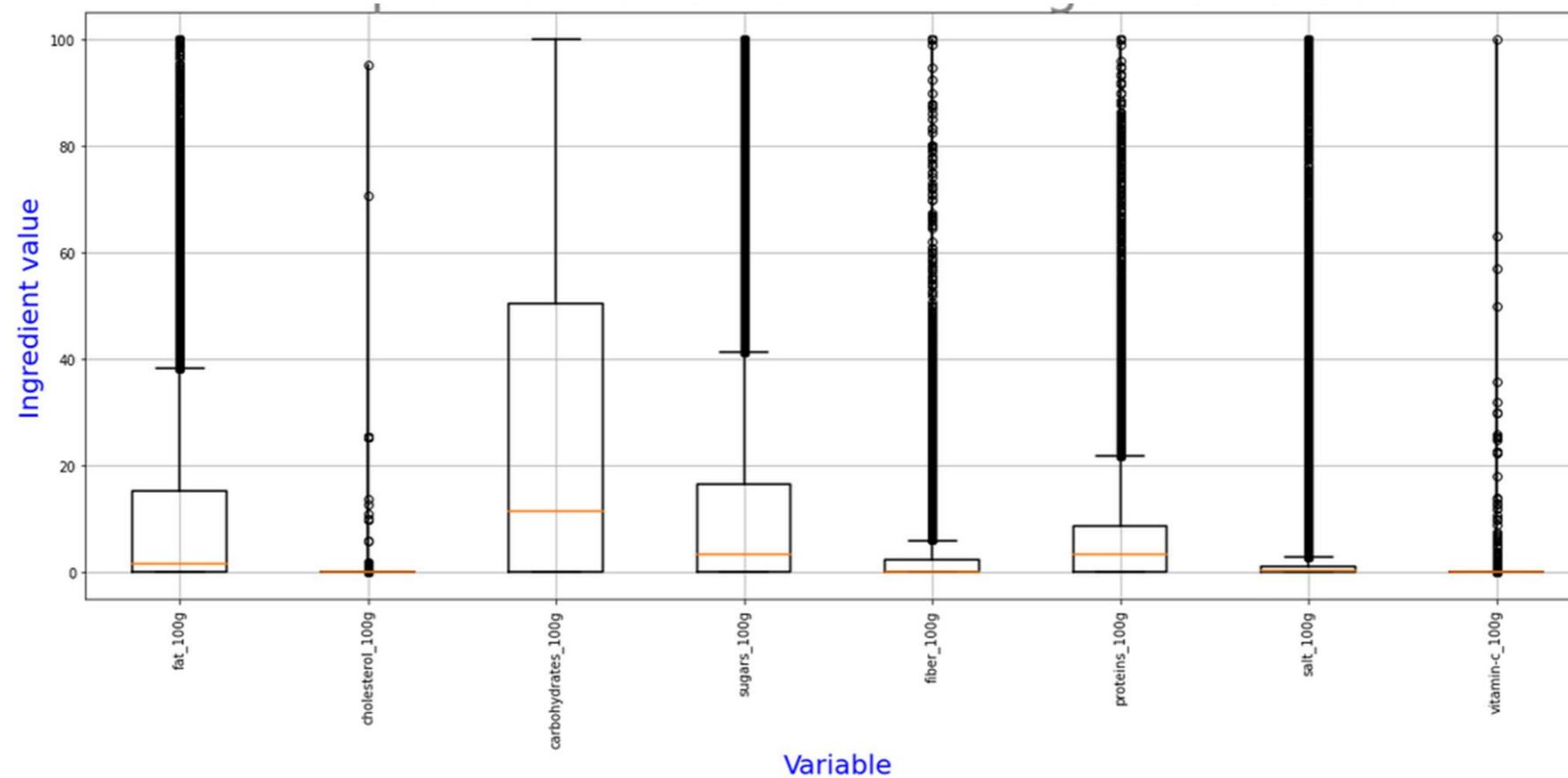
Out[54]:
code url creator created_datetime last_modified_datetime product_name brands countries main_category energy_100g fat_100g cholesterol_1
```

Traitement des donnes aberrantes du Dataset :

Un exemple d'affichage en boxplot apres le traitement effectue precedemment.

Aucune valeur negative ni celle qui depasse la valeur de 100 g

- Certaines valeurs restent incorrectes selon les ratios standards des aliments et seront traitees plus tard au besoin.



Traitement de la localisation des produits du dataset :

Un traitement pour une estimation de la distribution des produits par pays a été réalisé et on peut voir ci-dessous l'étendue des données disponibles et le niveau de difficulté pour les traiter dans le cadre de ce projet.

```
In [4]: # Evaluation de la distribution géographique des variables du dataset
# Le traitement de cette colonne pour une uniformisation qui faciliterait l'utilisation de la localisation geo
# prendrait un temps supérieur à celui alloué dans le cadre de ce projet
df['countries'].value_counts(sort=True)
```

```
Out[4]: us
169552
france
76029
en:fr
11322
suisse
9184
deutschland
5827

...
belgique,france,suisse, en:germany
1
tunisie,france
1
global market
1
dánia,finnország,franciaország,németország,magyarország,olaszország,hollandia,norvégia,portugália,oroszország,szlovénia,spa
nyolország,svédország,nagy-britannia
en:an, en:france
1
Name: countries, Length: 1357, dtype: int64
```

```
In [5]: df['countries'].nunique()
```

```
Out[5]: 1357
```

```
In [6]: dg=df.loc[df['countries'].str.contains('france')]
```

```
In [7]: dg['countries'].value_counts()
```

```
Out[7]: france
76029
france, suisse
499
belgique,france
217
france,suisse
209
uk,france
124
...
france,belgique,danemark
1
saint pierre and miquelon, en:france, en:belgium
1
royaume-uni,france, en:denmark, suisse
1
france,belgique,allemande,suisse,autriche,pays-bas,danemark,suède
1
china, en:france
1
Name: countries, Length: 559, dtype: int64
```

Definition du dataset relatif au produit Ice-cream :

Le produit ice-cream (ou glaces) a ete choisi pour etre utilise dans l'application web finale.

Il faudra traiter la variable energy qui presente des valeurs incorrectes

	variable	count_of_values	Null_value	dtype	Zero_values	min	Mean	max	Outliers > 100	Outliers-excess > 900
0	energy_100g	3195	0	float64	19	0.0	975.589671	2720.0000	3168	1994
1	fat_100g	3195	0	float64	118	0.0	11.748213	52.9400	0	0
2	cholesterol_100g	3195	0	float64	211	0.0	0.058705	70.5880	0	0
3	carbohydrates_100g	3195	0	float64	25	0.0	28.947061	96.4300	0	0
4	sugars_100g	3195	0	float64	44	0.0	21.751577	91.6700	0	0
5	fiber_100g	3195	0	float64	2001	0.0	0.714870	38.7000	0	0
6	proteins_100g	3195	0	float64	77	0.0	3.656973	24.0000	0	0
7	salt_100g	3195	0	float64	47	0.0	0.257351	54.3560	0	0
8	vitamin-c_100g	3195	0	float64	2559	0.0	0.000692	0.0508	0	0

Il faudra traiter la variable 'energy' qui presente des valeurs incorrectes (excessives)

Definition du dataset relatif au produit Ice-cream :

Le produit ice-cream (ou glaces) a ete choisi pour etre utilise dans l'application web finale.

Il faudra traiter la variable 'energy' qui presente des valeurs incorrectes (excessives)

```
In [27]: # On se propose de recalculer l'apport energetique par aliments pour evaluer l'écart des valeurs
# inscrites dans le dataset
# Energie = protéines + lipides + glucides
# On utilisera les coefficients suivants :
# 1g de protéine = 4 kcal // 1g de lipide = 9 kcal // 1g de Glucides = 4 kcal
```

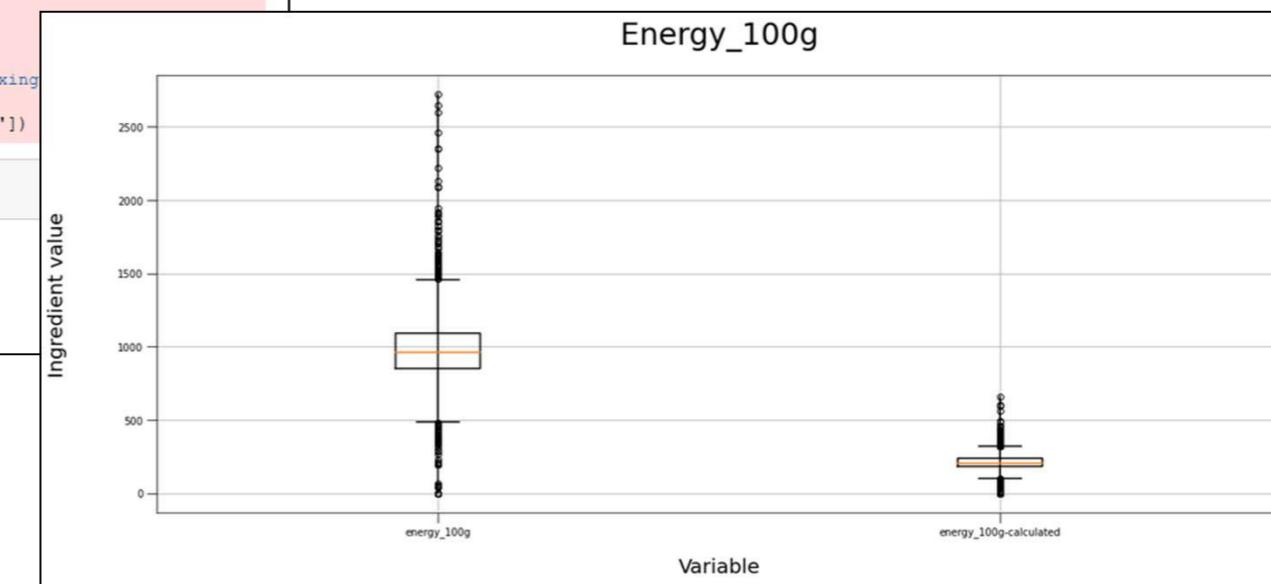
```
In [29]: # Formule de calcul énergétique pour les aliments
df2['energy_100g-calculated'] = 9*(df2['fat_100g'])+4*(df2['proteins_100g'])+4*(df2['sugars_100g'])

<ipython-input-29-5e19d970a989>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#versus-a-copy
df2['energy_100g-calculated'] = 9*(df2['fat_100g'])+4*(df2['proteins_100g'])+4*(df2['sugars_100g'])

In [30]: energy_calc = df2[['energy_100g', 'energy_100g-calculated']]
energy_calc.head(2)

Out[30]:
   energy_100g  energy_100g-calculated
280      883.0              180.31
281     1000.0              218.29
```



Definition du dataset relatif au produit Ice-cream :

On procede a un dernier traitement des variables relatives au nouveau dataset ‘ice-cream’ :

```
In [27]: # On se propose de recalculer l'apport energetique par aliments pour evaluer l'écart des valeurs
# inscrites dans le dataset
# Energie = protéines + lipides + glucides
# On utilisera les coefficients suivants :
# 1g de protéine = 4 kcal // 1g de lipide = 9 kcal // 1g de Glucides = 4 kcal
```

```
In [29]: # Formule de calcul energetique pour les aliments
df2['energy_100g-calculated'] = 9*(df2['fat_100g'])+4*(df2['proteins_100g'])+4*(df2['sugars_100g'])

<ipython-input-29-5e19d970a989>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df2['energy_100g-calculated'] = 9*(df2['fat_100g'])+4*(df2['proteins_100g'])+4*(df2['sugars_100g'])
```

```
In [30]: energy_calc = df2[['energy_100g','energy_100g-calculated']]
energy_calc.head(2)
```

```
Out[30]:      energy_100g  energy_100g-calculated
280        883.0            180.31
281       1000.0            218.29
```

Il faudra traiter la variable ‘energy’ qui presente des valeurs incorrectes (excessives)

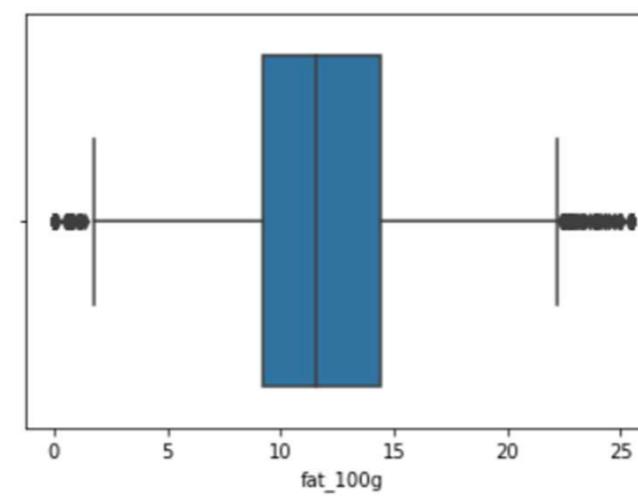
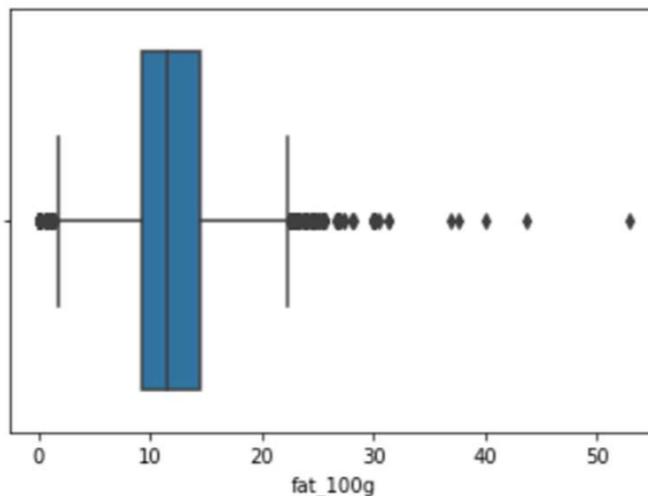
Utilisation de la méthode ‘iterative imputer’ pour le traitement des valeurs manquantes :

Traitement des variables restantes :

```
In [44]: # on se propose un traitement des valeurs aberrantes pour les colonnes 'fat' , 'carbohydrates', 'cholesterol', 'salt'  
# 'sugars et 'fiber' , 'proteins'  
# en doit établir des valeurs limites pour chaque variable pour établir un nettoyage clair et efficace
```

```
In [48]: fat_upper = df2['fat_100g'].quantile(.99)  
fat_upper
```

```
Out[48]: 25.49899999999999
```



Definition du dataset relatif au produit Ice-cream :

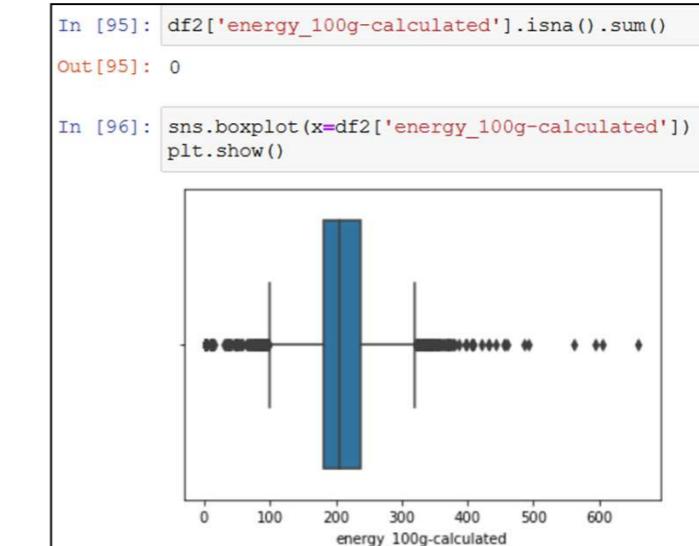
Utilisation de la methode 'iterative imputer' pour remplacer les valeurs manquantes de la variable 'energy-calculated' :

Situation initiale :

Module ML

```
In [89]: df2['energy_100g-calculated'].isna().sum()  
Out[89]: 38
```

```
In [90]: from sklearn.experimental import enable_iterative_imputer  
In [91]: from sklearn.impute import IterativeImputer  
  
In [92]: # Iterative imputer method  
# remplacement des valeurs manquantes de la colonne 'energy'- calculated  
# Le principe de 'ierative imputer' serait d'utiliser les correlations possibles entre la valeur la colonne ou il y'a  
# valeurs manquantes et les autres colonnes du dataset dans une approche multivariee  
# sikitlearn entraine un modele a regression utilisant les colonnes considerees comme 'features' et pour les lignes  
# ou la cible est manquante, les data 'features' des lignes seront utilisees pour predire la valeur manquante cible  
# On va utiliser les colonnes 'sugars' et 'fat' comme colonnes features  
# le tunning et l'evaluation du modele choisit ne sera pas traite dans ce projet  
  
In [93]: # estimator default=BayesianRidge()  
itr = IterativeImputer()  
  
In [94]: df2[['sugars_100g','fat_100g','energy_100g-calculated']] = itr.fit_transform(df2[['sugars_100g','fat_100g','energy_100g-']])
```



Statstiques et graphiques :

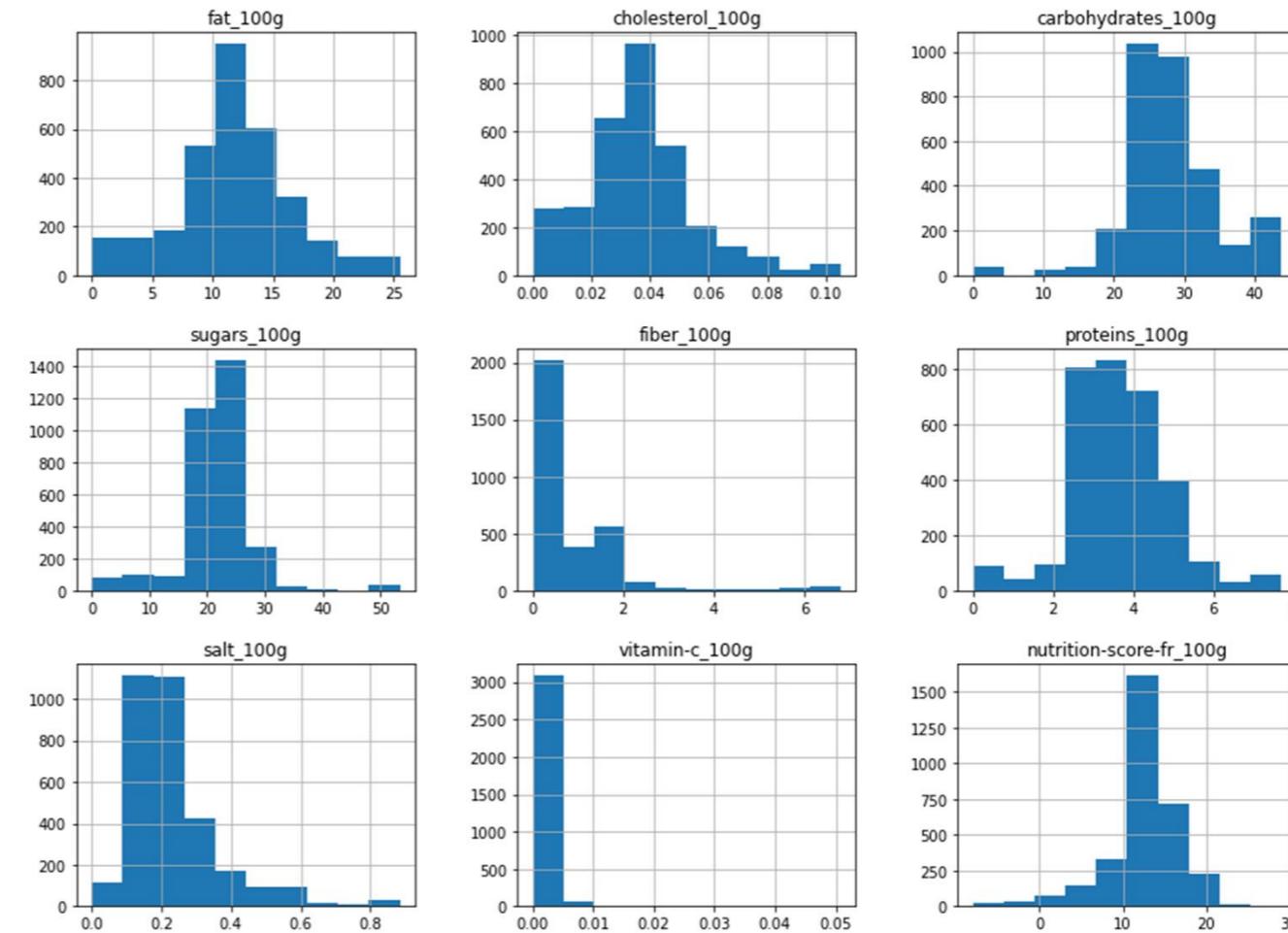
Principales donnees stats :

In [111]:	df3[list_no_code].describe()									
Out[111]:	fat_100g	cholesterol_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	vitamin-c_100g	nutrition-score-fr_100g	energy_100g-calculated
count	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000	3195.000000
mean	11.698954	0.036359	28.099186	21.575555	0.697437	3.624883	0.235657	0.000692	12.536490	207.390639
std	4.931286	0.019410	6.822569	6.427879	1.230215	1.238744	0.136872	0.002811	4.351003	60.480847
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-8.000000	1.893475
25%	9.230000	0.025000	24.490000	19.230000	0.000000	2.990000	0.157480	0.000000	11.000000	181.605000
50%	11.540000	0.037000	27.360000	21.800000	0.000000	3.410000	0.200660	0.000000	13.000000	205.820000
75%	14.450000	0.044000	30.930000	24.405000	1.300000	4.440000	0.274320	0.000000	15.000000	237.340000
max	25.499000	0.105000	43.807000	53.361200	6.800000	7.690000	0.885088	0.050800	29.000000	660.000000

Statstiques et graphiques :

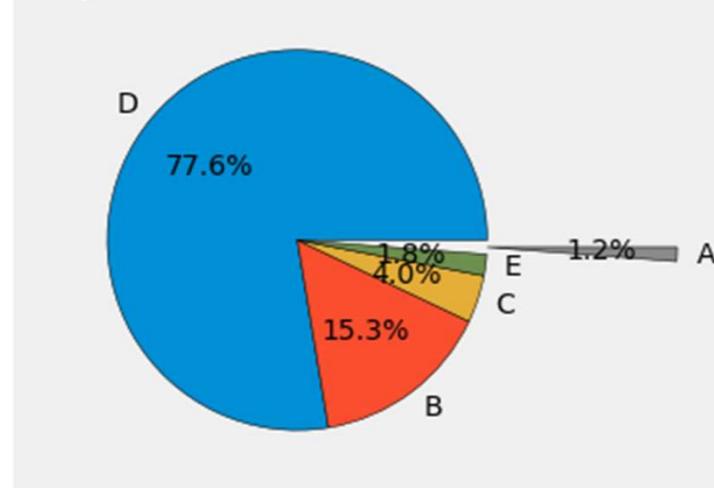
Principales donnees stats :

Les barplots ci contre montrent la fréquence de distribution de chaque variable.

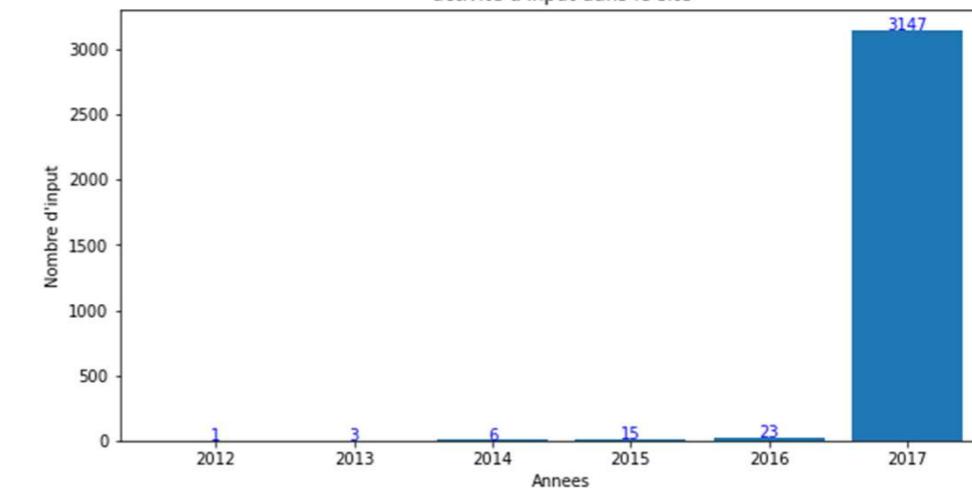


Statistiques et graphiques : Ci dessous quelques formes graphiques

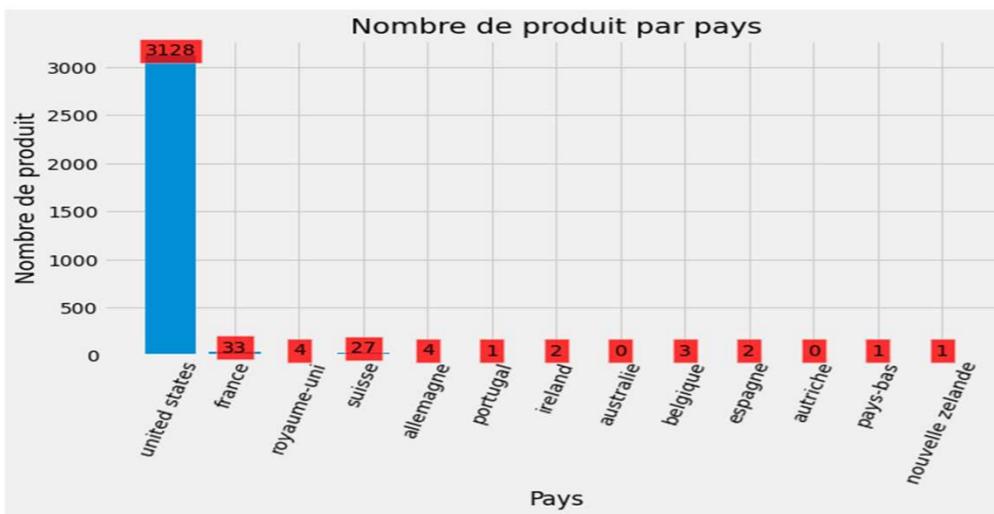
Repartition des nutri-score



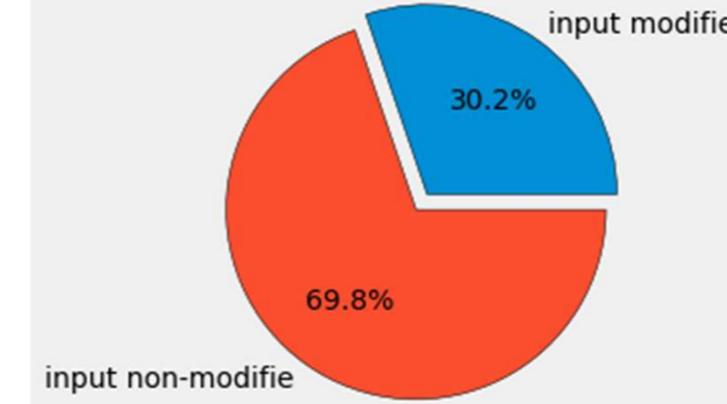
activite d'input dans le site



Nombre de produit par pays

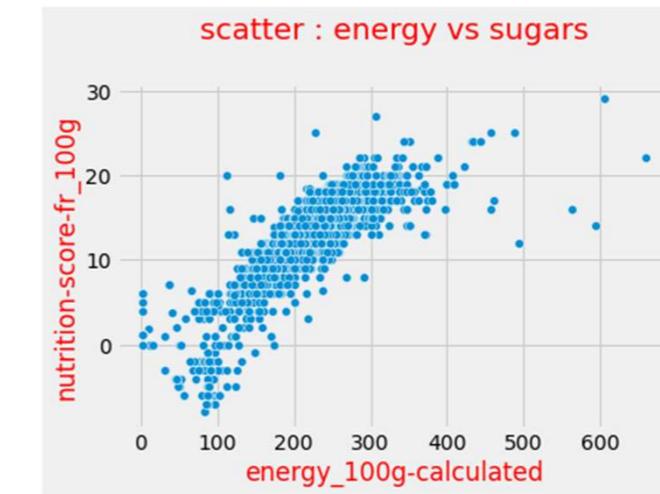
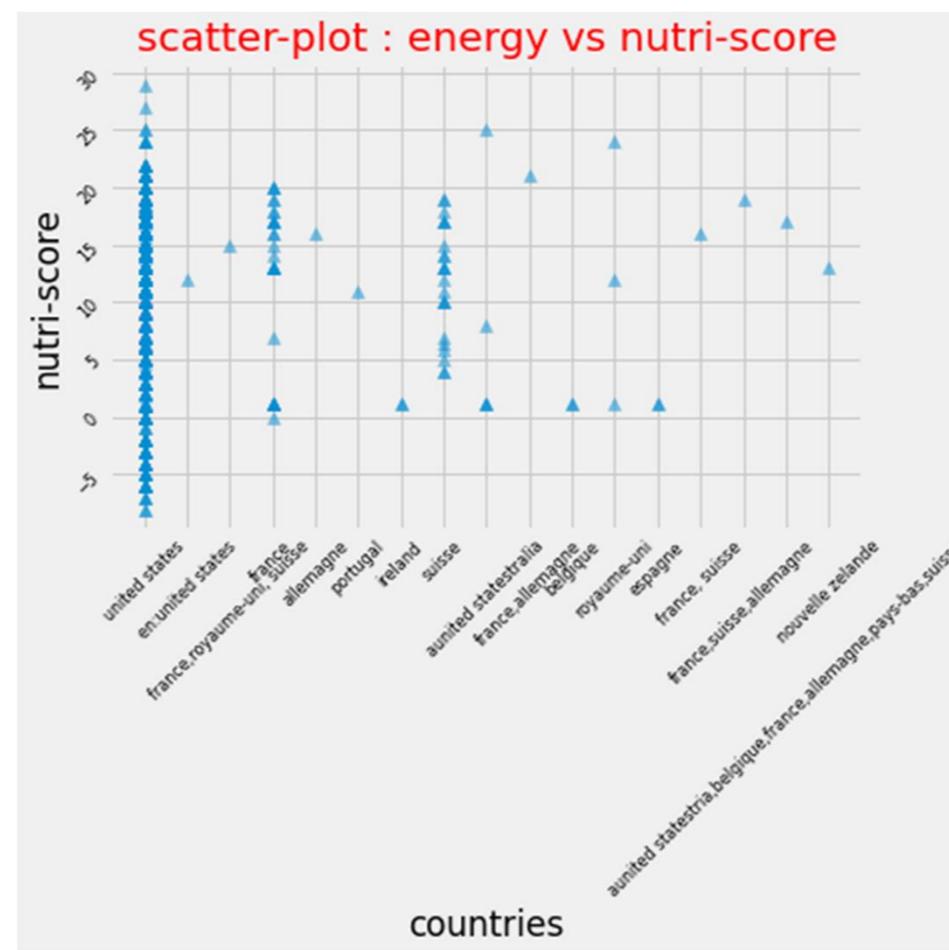
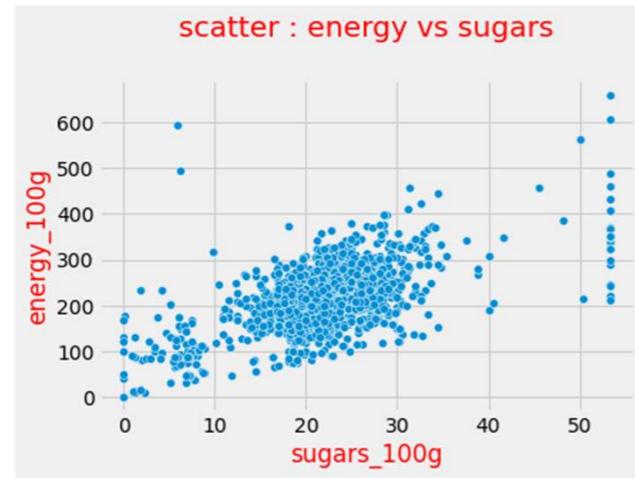


ratio de modification des input



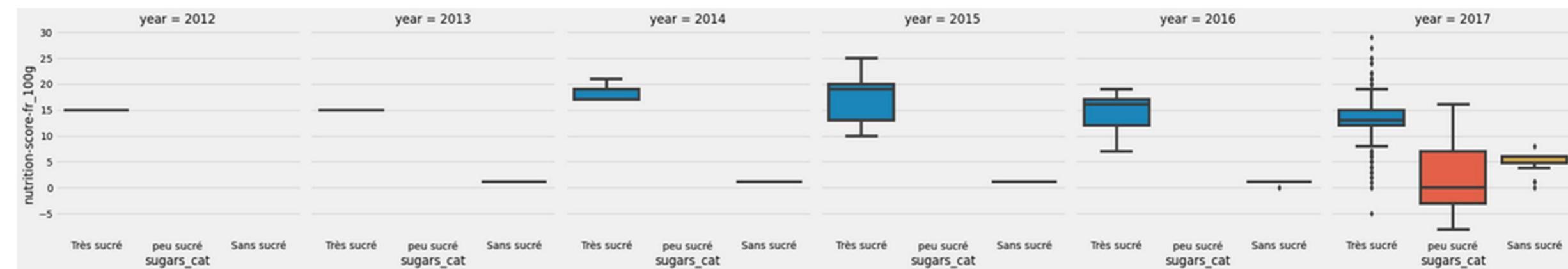
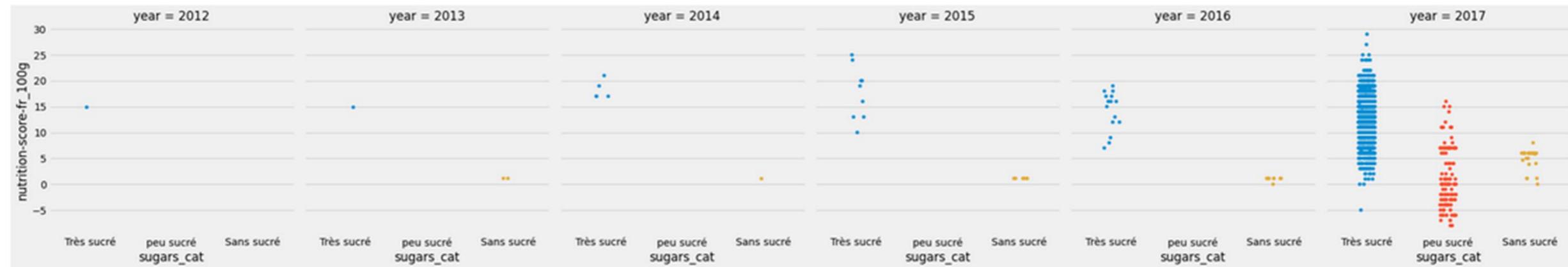
Statistiques et graphiques :

Analyse bivariee :



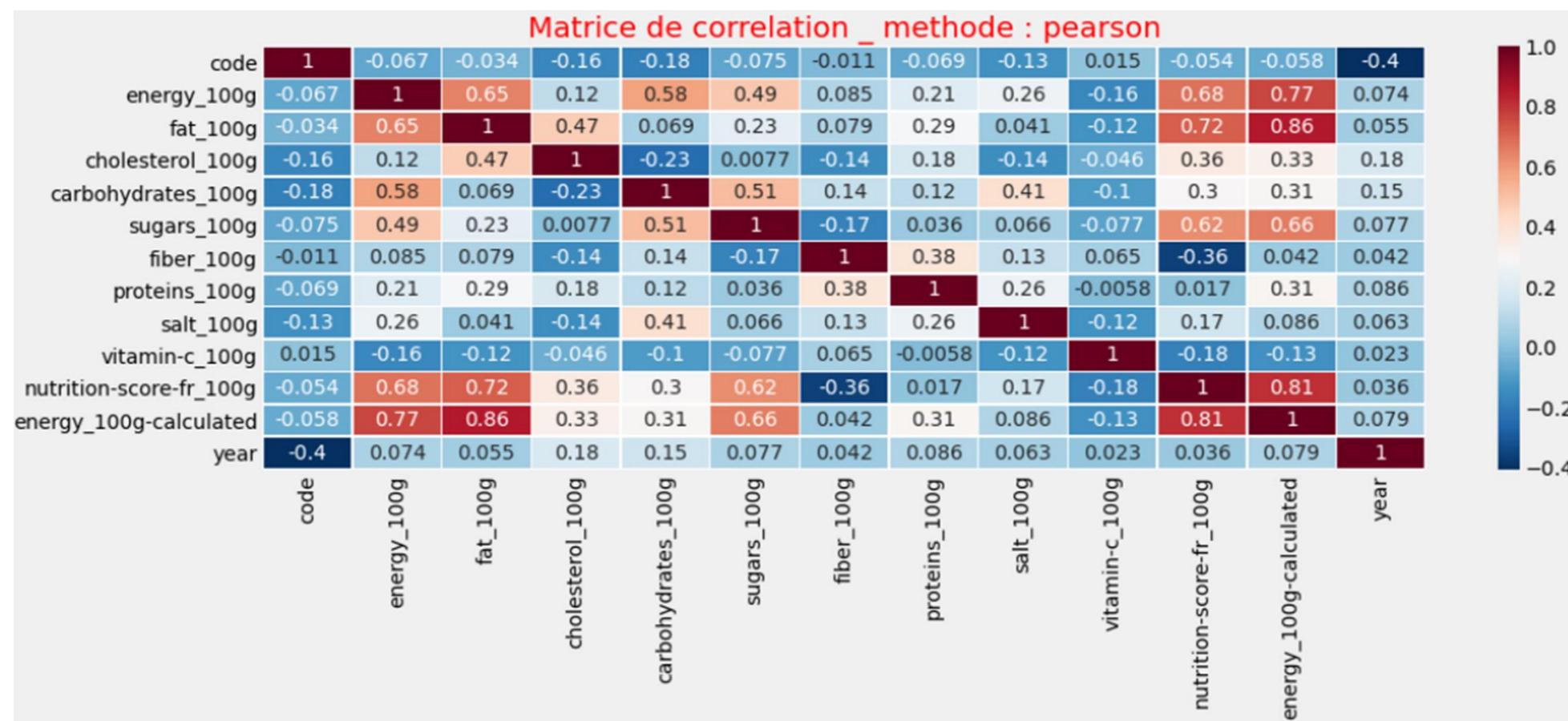
Statistiques et graphiques :

Analyse bivariee :



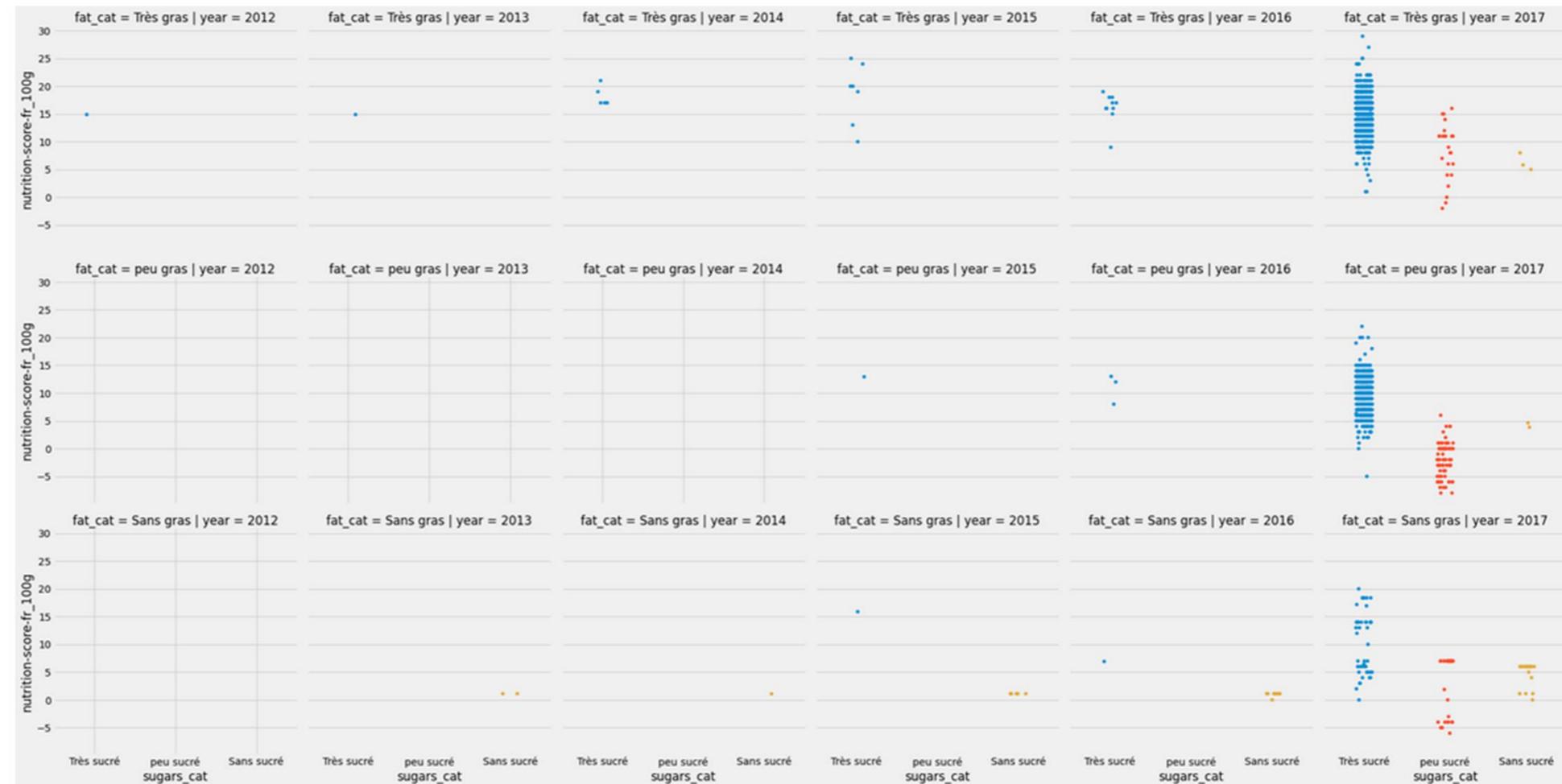
Statstiques et graphiques :

Analyse bivariee :



Statstiques et graphiques :

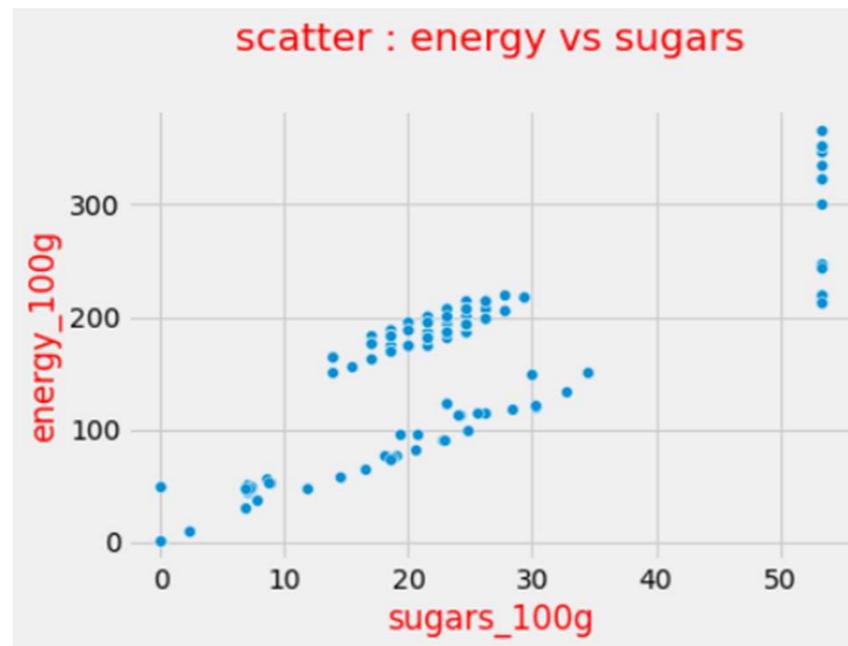
Analyse multivariée :



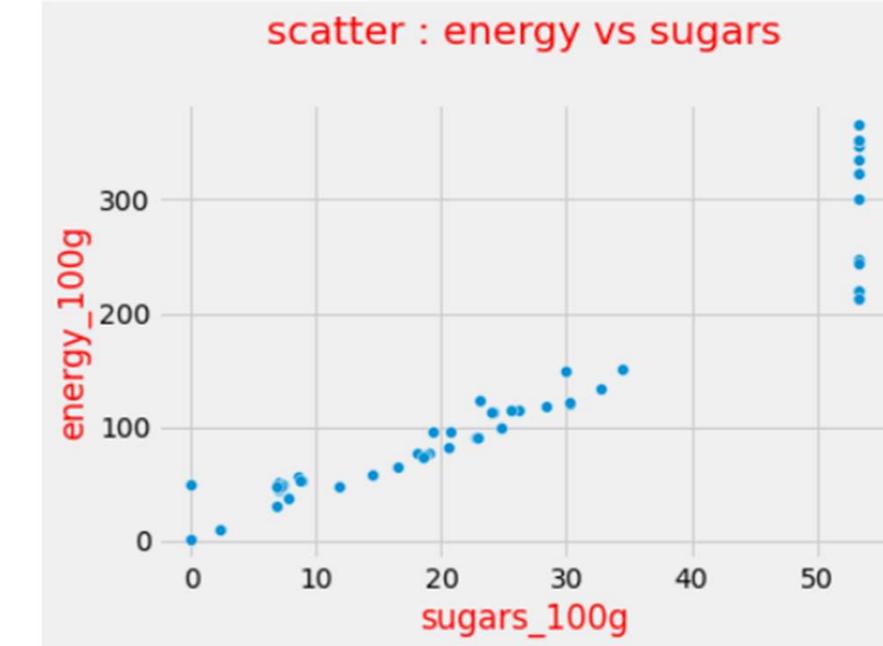
Statstiques et graphiques :

Analyse multivarreee :

```
df_fat_fixe = df3.loc[(df3['fat_100g']==10.77) | (df3['fat_100g']==0) | (df3['fat_100g']==9.23)]
```



```
In [165]: df_fat_nulle = df3.loc[(df3['fat_100g']==0)]
```



Statstiques et graphiques :

Methode ANOVA:

```
In [168]: import statsmodels.api as sm  
from statsmodels.formula.api import ols
```

```
In [174]: mod2 = ols('sugars_100g~countries', data=df3).fit()
```

```
In [176]: aov2 = sm.stats.anova_lm(mod2, type=2)
```

```
In [177]: print(aov2)
```

	df	sum_sq	mean_sq	F	PR(>F)
countries	16.0	4015.430878	250.964430	6.233261	6.859404e-14
Residual	3178.0	127953.085947	40.262142	NaN	NaN

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [184]: # Elimination de la colonne cible qui n'est pas prise en compte dans l'analyse ACP  
target_df = pd.DataFrame([],columns=['nutrition-score-fr_100g'])  
target_df['nutrition-score-fr_100g']=df3['nutrition-score-fr_100g']  
df3
```

```
In [193]: # Etape de standardisation des donnees pour eviter les effets de difference d'echelle  
# definition d'une fonction pour ce calcul  
def standardscalar(data):  
    "axis=0 means along the column, axis=1 means working along the row"  
    scaled_df = (data - np.mean(data, axis=0))/np.std(data, axis=0)  
    return scaled_df
```

```
scaled_df = standardscalar(df_pca)  
scaled_df = np.around(scaled_df,3)  
scaled_df
```

	energy_100g	fat_100g	cholesterol_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	vitamin-c_100g	nutrition-score-fr_100g	energ	ca
280	-0.361449	-0.087	0.651343	-0.814951	-0.727501	-0.567011	-0.649859	-0.68262	-0.246307	-0.123322	-0	
281	0.095292	0.768895	0.290648	-0.814951	-0.946892	0.571179	0.488569	0.765085	-0.246307	-0.123322	-0	
1601	-0.115511	0.026579	0.908982	-0.493907	-0.346289	-0.567011	0.545087	-0.719741	0.216207	0.106546	-0	
1602	-0.115511	0.243594	1.991067	-0.650764	-0.512777	-0.567011	0.545087	-0.515577	0.216207	0.106546	-0	
1603	0.993159	0.967657	0.23912	0.709641	0.853361	1.384172	0.876119	-0.08869	0.251785	0.336414	1	
...
301493	-3.808476	-2.372766	-1.873522	-4.119208	-3.357084	-0.567011	-2.926716	-1.721998	-0.246307	-2.623991	-0	
301511	-3.808476	-2.372766	-1.873522	-4.119208	-3.357084	-0.567011	-2.926716	-1.721998	-0.246307	-2.623991	-0	
302072	-1.548194	-1.500646	-1.873522	-0.718198	-0.011756	-0.567011	-1.796361	-1.369352	-0.246307	-1.042794	-1	
302470	-0.314603	-0.054549	-0.379214	-0.140612	-0.690158	-0.567011	-0.617563	-0.66406	2.208575	-0.123322	-0	
302677	-0.361449	-0.121479	-1.873522	-0.512964	0.050483	-0.567011	-0.423788	-0.886783	-0.246307	0.106546	-0	

3195 rows × 13 columns

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [194]: # Calcul de la matrice de variance  
# N-dimensional dataset genere une matrice de covariance d'ordre N*N  
# Si une variable tend a augmenter alors que les autres decroient alors le coefficient est negatif
```

```
In [195]: def compute_covariance_matrix(data):  
    len_data = data.shape[0]  
    covariance = data.T.dot(data)/len_data  
    return covariance  
  
cov_mat = compute_covariance_matrix(scaled_df)  
cov_mat = np.round(cov_mat, 3)  
cov_mat
```

	energy_100g	fat_100g	cholesterol_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	vitamin-c_100g	nutri-score-fr
energy_100g	1.0	0.64767	0.120117	0.582712	0.485606	0.085366	0.214966	0.260114	-0.156572	0.6
fat_100g	0.64767	1.0	0.468596	0.068683	0.228172	0.079039	0.288101	0.040628	-0.124467	0.7
cholesterol_100g	0.120117	0.468596	1.0	-0.234973	0.007653	-0.142881	0.180219	-0.138617	-0.045656	0.
carbohydrates_100g	0.582712	0.068683	-0.234973	1.0	0.512017	0.135681	0.11585	0.407793	-0.100415	0.3
sugars_100g	0.485606	0.228172	0.007653	0.512017	1.0	-0.170352	0.035619	0.065529	-0.077284	0.6
fiber_100g	0.085366	0.079039	-0.142881	0.135681	-0.170352	1.0	0.384854	0.134433	0.0647	-0.3
proteins_100g	0.214966	0.288101	0.180219	0.11585	0.035619	0.384854	1.0	0.258824	-0.005787	0.0
salt_100g	0.260114	0.040628	-0.138617	0.407793	0.065529	0.134433	0.258824	1.0	-0.12236	0.1
vitamin-c_100g	-0.156572	-0.124467	-0.045656	-0.100415	-0.077284	0.0647	-0.005787	-0.12236	1.0	-0.1
nutrition-score-fr_100g	0.680889	0.719595	0.36289	0.300504	0.622807	-0.357586	0.017265	0.165971	-0.175611	
energy_100g-calculated	0.765919	0.856088	0.333252	0.30635	0.655568	0.04223	0.313835	0.08555	-0.129513	0.8
year	0.074161	0.055057	0.183076	0.153801	0.076684	0.041674	0.086284	0.063309	0.022902	0.0
delta_time	-0.001928	-0.013292	-0.114725	-0.082976	-0.023402	-0.027167	-0.027014	-0.033835	-0.01243	0.0

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [200]: # Determination des valeurs et vecteurs propres
# on importe eig a partir de linalg
eig_values, eig_vectors = np.linalg.eig(cov_mat)
print('\nNumber of Eigenvectors : ', len(eig_vectors))
print('\nEigenvectors : ', eig_vectors)
print('\nEigenvalues : ', eig_values)
```

```
Number of Eigenvectors : 13
Eigenvectors :
[s  [[ 0.42534053 -0.16238904 -0.00465057 -0.00266766 -0.04834236  0.08844206
-0.30130115  0.12473735 -0.39634138 -0.12188973  0.62798163  0.33066123
0.02333109]
[ 0.39423911  0.20179066  0.09756649 -0.29664979  0.01003928  0.05743346
-0.34466616 -0.20595297 -0.05523788 -0.53716938 -0.34969309 -0.24075307
-0.26433973]
[ 0.17029291  0.5099614   0.0459481  -0.25147759  0.13693453 -0.2371161
0.15384147  0.70051213  0.17892731  0.00555801  0.04268381  0.13003498
-0.07383131]
[ 0.24877784 -0.44525122 -0.20757813  0.26489945 -0.07717233  0.03432543
0.0231472   0.51527549 -0.296794   0.0720699  -0.48235561 -0.14006129
-0.08864118]
[ 0.33228351 -0.11279691  0.05533518  0.34793507 -0.25830052  0.13204017
0.51696131 -0.03678162  0.44947056 -0.40607115  0.14307045  0.05679696
-0.09682704]
[-0.01354152 -0.29752962 -0.28449959 -0.50643114 -0.15291064  0.41609563
-0.2039099  0.17759775  0.469617   0.0099473  -0.00238253  0.02754651
0.28781804]
[ 0.15031201 -0.13084058 -0.21392543 -0.5393993  0.02623221 -0.16699988
0.62421395 -0.21363031 -0.36914127 -0.03884397 -0.03343722  0.02966786
0.14121239]
[ 0.12715032 -0.39978142 -0.21348194 -0.01626432  0.36593485 -0.66212491
-0.17876911 -0.10481521  0.36329749 -0.02321201  0.08725908  0.01598749
-0.15004203]
[-0.10479804  0.05047621 -0.0573777 -0.07721619 -0.85409656 -0.47345402
-0.15107341  0.00669725 -0.01773492  0.00172448 -0.00957516  0.01721219
0.00136149]
[ 0.43282211  0.13103919  0.18642262  0.17021783  0.03573662 -0.16771355
-0.08725737 -0.11556579  0.07308371  0.13255503 -0.19069793 -0.03121341
0.78812434]
[ 0.4650253   0.06170602  0.07478148 -0.1041509  -0.1299111  0.12928997
0.0256814   -0.21398015  0.14830013  0.71138244 -0.00532968 -0.03932762
-0.39221341]
[ 0.08258588  0.28755946 -0.60754194  0.16692522  0.00220102  0.02864128
-0.00593059 -0.00459754 -0.03470297  0.01773518  0.32171834 -0.63436198
0.07210941]
[-0.04502998 -0.3003205   0.59763998 -0.1902203  -0.02277161 -0.06917962
0.06825269  0.1925382  -0.00966427  0.01932087  0.2822578  -0.62114574
0.03782389]]]

Eigenvalues :
[s  [4.10302268 1.79627682 1.90208103 1.55088822 1.01891912 0.79024935
0.62365219 0.44510242 0.3696318  0.01426756 0.16118015 0.14906124
0.07570399]
```

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [201]: # Conversion de l'array des valeurs propres en un dataframe pandas pour une meilleure visualisation
```

```
In [202]: eigenvc_df = pd.DataFrame(eig_vectors)
eigenvc_df.columns = ['eigvec_'+str(i+1) for i in range(0, len(eig_vectors))]
eigenvc_df.T
```

```
Out[202]:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
eigvec_1	0.425341	0.394239	0.170293	0.248778	0.332284	-0.013542	0.150312	0.127150	-0.104798	0.432822	0.465025	0.082586	-0.045030
eigvec_2	-0.162389	0.201791	0.509961	-0.445251	-0.112797	-0.297530	-0.130841	-0.399781	0.050476	0.131039	0.061706	0.287559	-0.300320
eigvec_3	-0.004651	0.097566	0.045948	-0.207578	0.055335	-0.284500	-0.213925	-0.213482	-0.057378	0.186423	0.074781	-0.607542	0.597640
eigvec_4	-0.002668	-0.296650	-0.251478	0.264899	0.347935	-0.506431	-0.539399	-0.016264	-0.077216	0.170218	-0.104151	0.166925	-0.190220
eigvec_5	-0.048342	0.010039	0.136935	-0.077172	-0.258301	-0.152911	0.026232	0.365935	-0.854097	0.035737	-0.129911	0.002201	-0.022772
eigvec_6	0.088442	0.057433	-0.237116	0.034325	0.132040	0.416096	-0.167000	-0.662125	-0.473454	-0.167714	0.129290	0.028641	-0.069180
eigvec_7	-0.301301	-0.344666	0.153841	0.023147	0.516961	-0.203910	0.624214	-0.178769	-0.151073	-0.087257	0.025681	-0.005931	0.068253
eigvec_8	0.124737	-0.205953	0.700512	0.515275	-0.036782	0.177598	-0.213630	-0.104815	0.006697	-0.115566	-0.213980	-0.004598	0.192538
eigvec_9	-0.396341	-0.055238	0.178927	-0.296794	0.449471	0.469617	-0.369141	0.363297	-0.017735	0.073084	0.148300	-0.034703	-0.009664
eigvec_10	-0.121890	-0.537169	0.005558	0.072070	-0.406071	0.009947	-0.038844	-0.023212	0.001724	0.132555	0.711382	0.017735	0.019321
eigvec_11	0.627982	-0.349693	0.042684	-0.482356	0.143070	-0.002383	-0.033437	0.087259	-0.009575	-0.190698	-0.005330	0.321718	0.282258
eigvec_12	0.330661	-0.240753	0.130035	-0.140061	0.056797	0.027547	0.029668	0.015987	0.017212	-0.031213	-0.039328	-0.634362	-0.621146
eigvec_13	0.023331	-0.264340	-0.073831	-0.088641	-0.096827	0.287818	0.141212	-0.150042	0.001361	0.788124	-0.392213	0.072109	0.037824

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [204]: indexes = eig_values.argsort()[:-1]
eig_values = eig_values[indexes]
eig_vectors = eig_vectors[:,indexes]
sorted_eig_pairs = [(np.around(np.abs(eig_values[i]),3),eig_vectors[:,i]) for i in range(len(eig_values))]
print('\nSorted Eigen-pairs (descending order) :\n')

sorted_eigenValues= []
for i in range(0, len(sorted_eig_pairs)):
    print("eigenvec_{0} :{1}".format(str(i+1),str(sorted_eig_pairs[i][0])))
    sorted_eigenValues.append(sorted_eig_pairs[i][0])

print("\nTotal variance (sum of eigenvalues) : ", round(sum(sorted_eigenValues),3))
```

Sorted Eigen-pairs (descending order) :

eigenvec_1 :4.103
eigenvec_2 :1.902
eigenvec_3 :1.796
eigenvec_4 :1.551
eigenvec_5 :1.019
eigenvec_6 :0.79
eigenvec_7 :0.624
eigenvec_8 :0.445
eigenvec_9 :0.37
eigenvec_10 :0.161
eigenvec_11 :0.149
eigenvec_12 :0.076
eigenvec_13 :0.014
n\Total variance (sum of eigenvalues) : 13.0

```
In [205]: # Choix du nombre 'K'( le nombre optimal des composantes principales)
# on utilise la methode de 'Kaise's stopping rule' qui consiste a choisir les colonnes avec des valeurs propres > 1.

In [206]: best_eig_pairs = [sorted_eig_pairs[i][0] for i in range(0, len(sorted_eig_pairs)) if sorted_eig_pairs[i][0]>1.0]
print("\nAccording to Kaiser's stopping rule :")
print("Number of PCs to be considered in PCA is : :{}".format(str(len(best_eig_pairs))))
print("Their eigenValues are : {}".format(str(len(best_eig_pairs))))
```

According to Kaiser's stopping rule :
Number of PCs to be considered in PCA is : :5
Their eigenValues are : 5

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [207]: # Construction de la matrice de projection a partir du choix des PC = 5
In [208]: K=5
projection_matrix = np.array([list(np.hstack(i[1].reshape(13,1))) for i in sorted_eig_pairs[:]])
projection_matrix = projection_matrix[:,K]
print('\nProjection-Matrix Dimension ....',projection_matrix.shape)
projection_matrix
```

Projection-Matrix Dimension (5, 13)

```
Out[208]: array([[ 0.42534053,  0.39423911,  0.17029291,  0.24877784,  0.33228351,
       -0.01354152,  0.15031201,  0.12715032, -0.10479804,  0.43282211,
       0.4650253 ,  0.08258588, -0.04502998],
      [-0.00465057,  0.09756649,  0.0459481 , -0.20757813,  0.05533518,
       -0.28449959, -0.21392543, -0.21348194, -0.0573777 ,  0.18642262,
       0.07478148, -0.60754194,  0.59763998],
      [-0.16238904,  0.20179066,  0.5099614 , -0.44525122, -0.11279691,
       -0.29752962, -0.13084058, -0.39978142,  0.05047621,  0.13103919,
       0.06170602,  0.28755946, -0.3003205 ],
      [-0.00266766, -0.29664979, -0.25147759,  0.26489945,  0.34793507,
       -0.50643114, -0.5393993 , -0.01626432, -0.07721619,  0.17021783,
       -0.1041509 ,  0.16692522, -0.1902203 ],
      [-0.04834236,  0.01003928,  0.13693453, -0.07717233, -0.25830052,
       -0.15291064,  0.02623221,  0.36593485, -0.85409656,  0.03573662,
       -0.1299111 ,  0.00220102, -0.02277161]])
```

```
In [209]: # Transformation du dataset original vers le nouveau feature espace
```

```
In [210]: scaled_df.shape
transformedData = (scaled_df).dot(projection_matrix.T)
transformedData.columns = ['PC'+str(i+1) for i in range(0,K)]
transformedData
```

```
Out[210]:
```

	PC1	PC2	PC3	PC4	PC5
280	-0.922088	0.447664	1.341016	0.118972	0.442452
281	0.107134	-0.329111	0.252756	-1.401217	0.740237
1601	-0.13673	0.218175	1.202583	-0.438777	-0.069914
1602	0.105854	0.275446	1.810655	-0.887289	0.198766
1603	2.060053	-0.537415	-0.56881	-1.084677	-0.848952
...
301493	-9.044743	6.155315	-0.485151	-0.755273	0.984496
301511	-9.041861	6.117066	-0.46593	-0.743099	0.985953
302072	-3.518442	2.760926	-1.038062	1.355428	-0.267887
302470	-1.115063	0.116788	0.62483	0.348319	-1.856859
302677	-1.421905	5.138635	-2.330225	-0.308375	-0.247838

Retour au
dataframe
initial

3195 rows × 5 columns

Statstiques et graphiques :

Methode ACP (Analyse en composante principale) :

```
In [211]: # on remarque que le data set transforme contient les 3195 lignes mais seulement 5 colonnes au lieu de 13
# Ce nouveau dataset regroupe le maximum de variance du dataset original

In [212]: df3.columns

Out[212]: Index(['energy_100g', 'fat_100g', 'cholesterol_100g', 'carbohydrates_100g',
       'sugars_100g', 'fiber_100g', 'proteins_100g', 'salt_100g',
       'vitamin-c_100g', 'nutrition-score-fr_100g', 'energy_100g-calculated',
       'year', 'delta_time'],
      dtype='object')

In [213]: # la matrice de projection au format dataframe

In [214]: projection_matrix_df = pd.DataFrame(projection_matrix)
projection_matrix_df.columns = [str(i+1) for i in range(0, len(eig_vectors))]
projection_matrix_df = projection_matrix_df.set_index([pd.Index(['PC1','PC2','PC3','PC4','PC5'])])
projection_matrix_df

Out[214]:
   1    2    3    4    5    6    7    8    9    10   11   12   13
PC1  0.425341  0.394239  0.170293  0.248778  0.332284 -0.013542  0.150312  0.127150 -0.104798  0.432822  0.465025  0.082586 -0.045030
PC2 -0.004651  0.097566  0.045948 -0.207578  0.055335 -0.284500 -0.213925 -0.213482 -0.057378  0.186423  0.074781 -0.607542  0.597640
PC3 -0.162389  0.201791  0.509961 -0.445251 -0.112797 -0.297530 -0.130841 -0.399781  0.050476  0.131039  0.061706  0.287559 -0.300320
PC4 -0.002668 -0.296650 -0.251478  0.264899  0.347935 -0.506431 -0.539399 -0.016264 -0.077216  0.170218 -0.104151  0.166925 -0.190220
PC5 -0.048342  0.010039  0.136935 -0.077172 -0.258301 -0.152911  0.026232  0.365935 -0.854097  0.035737 -0.129911  0.002201 -0.022772

In [215]: # On reconnaît le nom des colonnes à considérer selon la combinaison 'classement/ valeur'
# PC1 = energy_100g
# PC2 = delta_time
# PC3 = cholesterol
# PC4 = carbohydrates_100g
# PC5 = salt_100g

In [216]: # Le résultat est conséquent vu que la méthode a éliminé l'ensemble des colonnes liées entre elles
```

Application via voila:

Une application via voila est disponible avec ce projet et qui permet de visualiser sur une page web :

- Contenu du dataset d'une maniere personnalisable.
- Choix et visualisation des caracteristiques d'un produit en fournissant son code bar.
- L'ajout d'un produit au dataset

Conclusion :

Pour une meilleure prise charge et amélioration et diversification des données OPENFOOD :

- Prevoir une option efficace de correction/verification par rapport à chaque produit introduit dans le dataset et qui se base sur les points suivants :
 - Données fournies par la partie 'creator'
 - Données fournies par les images du produit.
 - Croisement des données avec l'archive ou bien avec des données issues de sources différentes.
- Prevoir une campagne de sensibilisation /marketing dans les pays autres que les USA pour avoir plus d'inputs dans le dataset et donc plus de diversification.