Nourriture application pour les enfants

Student: Azadeh POHIER

Mentor: Rim ROMDHANE

July 2021

Contents

Introduction:

Le service de la santé publique française recherche des idées innovantes d'applications en lien avec l'alimentation. Pour cela, nous avons à notre disposition la base de données Open Food Facts :

https://world.openfoodfacts.org/data

Nous effectuons une analyse de l'ensemble de données du site Web Open Food Facts. Open Food Facts est une base de données de produits alimentaires avec des ingrédients, des allergènes, des informations nutritionnelles et toutes les informations que nous pouvons trouver sur les étiquettes des produits.

Le Programme National Nutrition Santé utilise les données Open Food Facts pour valider la formule de son score de qualité nutritionnelle et de ses notes nutritionnelles.

L'objectif de notre analyse est d'écrire une application pour que les enfants aient une alimentation saine.

Les données :

Les champs sont séparés en quatre sections :

- Les informations générales sur la fiche du produit : nom, date de modification, etc.
- Un ensemble de tags : catégorie du produit, localisation, origine, etc.
- Les ingrédients composant les produits et leurs additifs éventuels.
- Des informations nutritionnelles : quantité en grammes d'un nutriment pour 100 grammes du produit.

Les données plus importantes pour notre application :

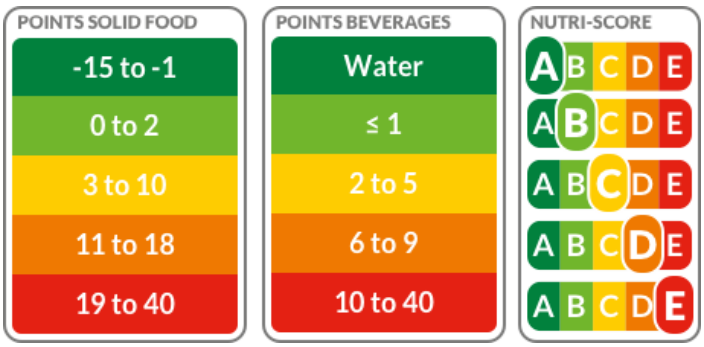
Nous voulons faire une application alimentaire pour les enfants de 3 à 12 ans.

L'alimentation plus importante pour la croissance des enfants :

- Les lipides
- Les protides
- les glucides (céréaliers, fruits et légumes, produits laitiers, sucre)
- Les vitamines (vitamines A, C, D, E, K et les vitamines B)
- Les minéraux (magnésium, calcium, fer, cuivre, zinc, sodium, sélénium,..)

Donc, nous essayons de faire le premier filtre sur le colon nutrition_score_fr _100g.

Méthode de calcul du score :



Le score est calculé par un système de points, le score le plus faible étant le meilleur.

Mais le score d'un produit acheté avec une cuisson incomplète et dont la cuisson est terminée par le consommateur, comme des frites précuites surgelées, sera nettement meilleur que celui qu'obtiendra le même produit après cuisson dans un bain d'huile de friture ; à l'inverse, un litre d'huile d'olive, même de très bonne qualité, sera coté D ou E, car cet aliment, comme toute autre huile alimentaire, est naturellement très gras.

Il faut donc analyser les produits et réfléchir avant de prendre une décision.

Quatre catégories avec des formules différentes sont mises en place :

Boissons, Fromages, Matières grasses, Autres aliments

Les résultats du calcul donnent une valeur comprise entre –15 et +40. La couleur verte correspondant à une valeur comprise entre –15 et –2, le vert clair de –1 à +3, le jaune de +4 à +11, l'orange de +12 à +16 et le rouge de +17 à +40.

Éléments défavorables au score

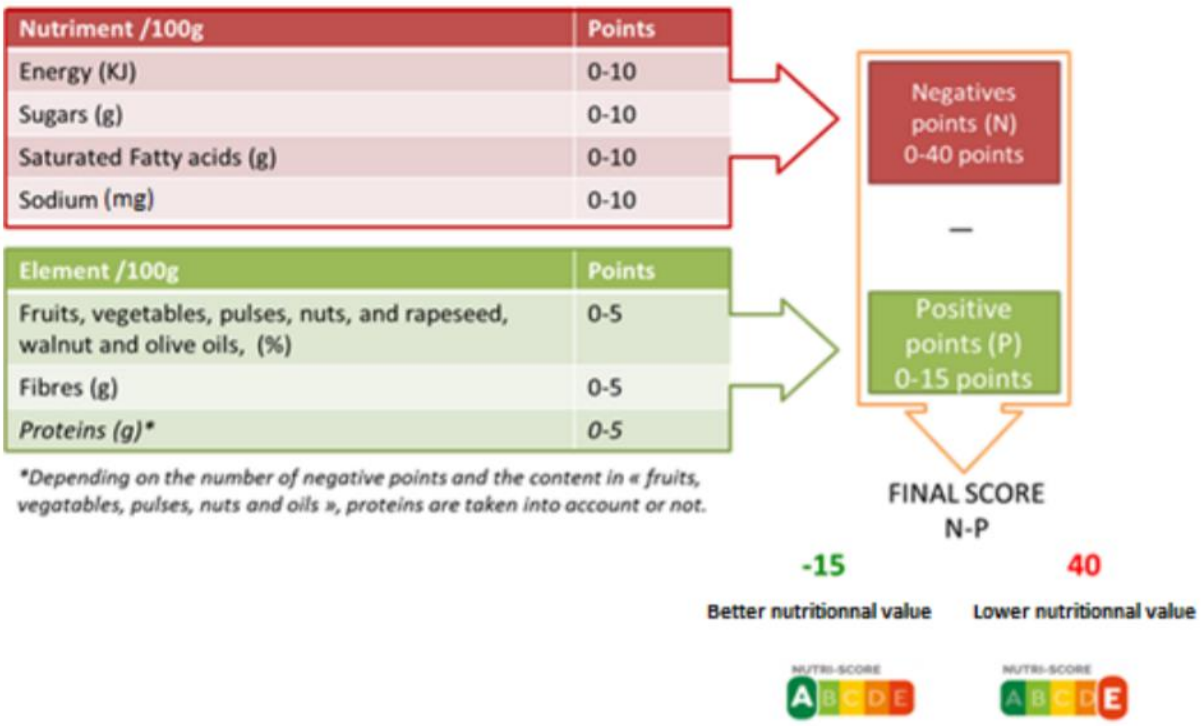
- Apport calorique pour cent grammes
- Teneur en sucre
- Teneur en graisses saturées
- Teneur en sel

Éléments favorables au score

- Teneur en fruits, légumes, légumineuses (dont les légumes secs), oléagineux, huiles de colza, de noix et d'olive.
- Teneur en fibres
- Teneur en protéines

Pour calculer la teneur de fruits et légumes, les féculents (tel que pomme de terre, patate douce, taro, manioc et tapioca) ne sont pas pris en compte.

Pour les fromages, la teneur en protéines est toujours prise en compte car celle-ci est liée à celle en calcium. Ceci améliore le nutri-score des fromages et la cohérence entre celui-ci et les recommandations nutritionnelles du Haut Conseil de la Santé Publique. Celles-ci recommandent en effet de consommer des produits laitiers plusieurs fois par jour.



Objective de la mission:

1) Traiter le jeu de données afin de repérer des variables pertinentes pour les traitements à venir. Automatiser ces traitements pour éviter de répéter ces opérations.

2) Produire des visualisations, effectuer une analyse univariée, bivariée et multivariée.

3) Confirmer les hypothèses à l'aide d'une analyse multivariée. Effectuer les tests statistiques appropriés pour vérifier la significativité des résultats.

Nettoyage de données

Le code pour lire les 10 premières lignes du jeu de données:

```
data1 = pd.read_csv("C:/Users/azade/Desktop/OC/Projet 3/en.openfoodfacts.org.products.csv",
                    nrows=10,sep='\t',low_memory=False)
data1
```

Nous avons un jeu de de données volumineux, donc nous devons utiliser le paramètre chunksize pour lire notre jeu de de données en plusieurs parties.

```
data = pd.read_csv('C:/Users/azade/Desktop/OC/Projet 3/en.openfoodfacts.org.products.csv', chunksize=200000,sep='\t'
,low_memory=False)

for chunk in data:
    print(chunk)
```

```
1        2018-10-13T21:06:57Z                                Cacao
2        2019-11-19T15:02:17Z            Filetes de pollo empanado
3        2021-04-27T05:38:17Z          Hamburguesas de ternera 100%
4        2015-10-12T14:13:32Z           moutarde au moût de raisin
...                       ...                                  ...
199995   2020-04-23T15:29:29Z                    Red sweet peppers
199996   2020-10-11T14:44:35Z                         Simply pesto
199997   2020-04-23T07:59:33Z                   Pepperazzi Peppers
199998   2020-04-22T17:35:40Z          Green & red stuffing peppers
199999   2020-04-22T17:35:40Z  Mild stuffing peppers, green peepers.
```

Après avoir lu l'ensemble de données en plusieurs parties, nous devons utiliser la fonction concat pour les ajouter.

```
data = pd.read_csv('C:/Users/azade/Desktop/OC/Projet 3/en.openfoodfacts.org.products.csv', chunksize=200000,sep='\t'
,low_memory=False)
df= pd.concat(data)
```

La taille du jeu de données

```
df.info(memory_usage=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1829437 entries, 0 to 1829436
Columns: 186 entries, code to carnitine_100g
dtypes: float64(123), int64(2), object(61)
memory usage: 2.5+ GB
```

Ici, nous avons 1829437 observations et 186 valeurs donc il faut commencer à filtrer le jeu de données.

'nutrition-score-fr_100g' est une valeur plus importante pour nos analyses, donc on garde tous l'observation avec valeurs non-na.

```
df.dropna(subset=['nutrition-score-fr_100g'], inplace=True)
```

```
df.shape
```

```
(675587, 186)
```

On a diminuée l'observation de jeu donnée jusqu'à 675587. Etape suivent est diminuée les colons.

Vérifier la description de données

```
df.describe()
```

	created_t	last_modified_t	cities	allergens_en	serving_quantity	no_nutriments	additives_n	ingredients_from_palm_oil_n	ingredients_from_palm_oil	i
count	6.755870e+05	6.755870e+05	0.0	0.0	3.466980e+05	0.0	481124.000000	481124.000000	0.0	
mean	1.531626e+09	1.590905e+09	NaN	NaN	9.956275e+01	NaN	2.151406	0.019650	NaN	
std	5.255281e+07	2.539926e+07	NaN	NaN	2.795479e+03	NaN	3.025343	0.140698	NaN	
min	1.328021e+09	1.333873e+09	NaN	NaN	0.000000e+00	NaN	0.000000	0.000000	NaN	
25%	1.489094e+09	1.587577e+09	NaN	NaN	2.800000e+01	NaN	0.000000	0.000000	NaN	
50%	1.533028e+09	1.587667e+09	NaN	NaN	5.100000e+01	NaN	1.000000	0.000000	NaN	
75%	1.583670e+09	1.610461e+09	NaN	NaN	1.130000e+02	NaN	3.000000	0.000000	NaN	
max	1.623716e+09	1.623716e+09	NaN	NaN	1.001000e+06	NaN	39.000000	2.000000	NaN	

Apres examine de la partie 'count', il est clair qu'il existe beaucoup de valeurs manquantes. Donc, essaye de supprimer les colons avec plus de 50% valeurs manquant.

```
df= df.dropna(axis=1, thresh=338000)
```

On a réussi à diminuée les colons jusqu'à 49.

Observations en double :

Essaye de trouver les observations en double

```
df_food[df_food.code.duplicated(keep=False)].sort_values("code")
```

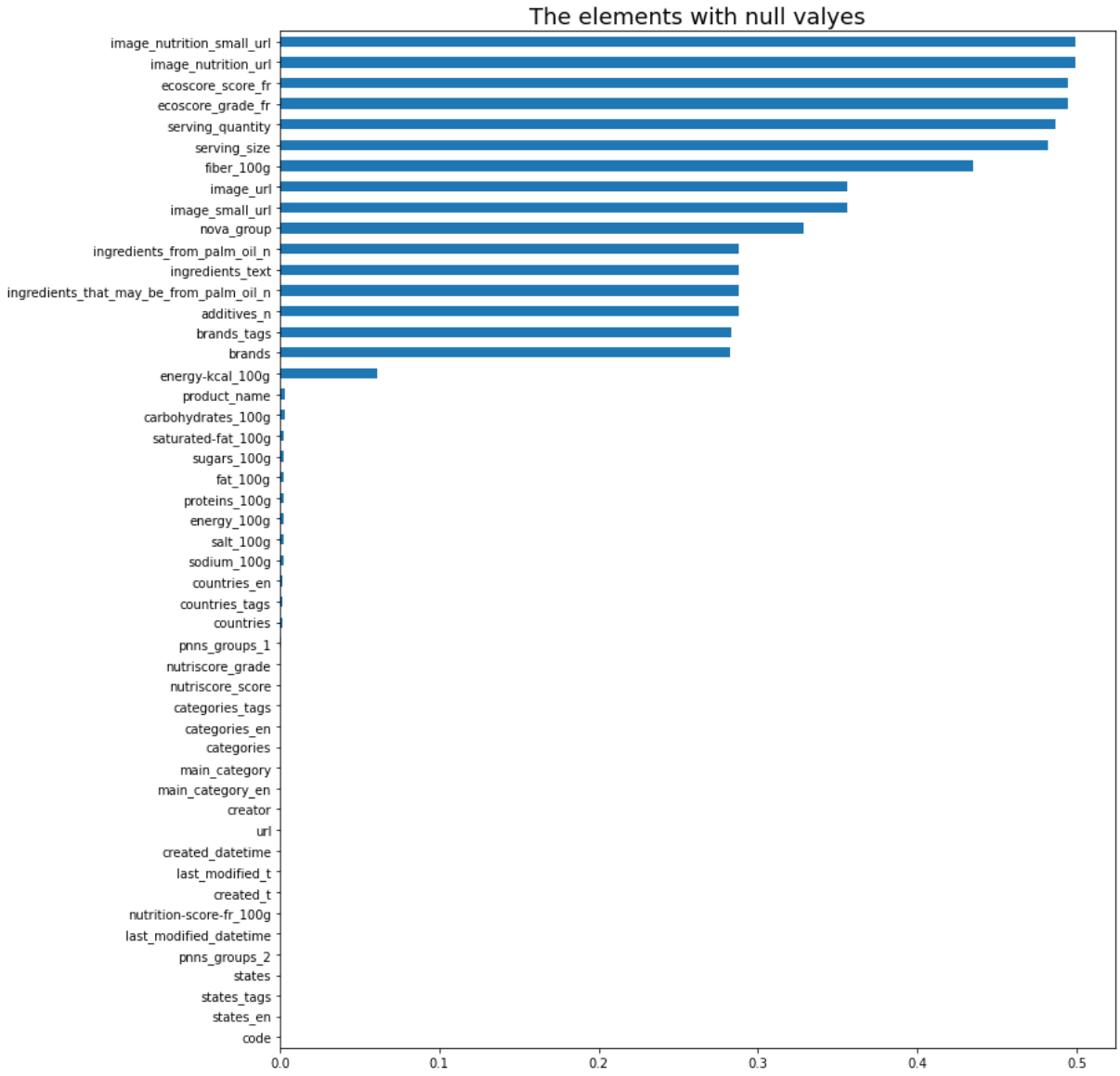
	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name	brands
311888	30383354190402	http://world-en.openfoodfacts.org/product/3038...	openfoodfacts-contributors	1608035756	2020-12-15T12:35:56Z	1610702480	2021-01-15T09:21:20Z	basilic	panzani
311889	30383354190402	http://world-en.openfoodfacts.org/product/3038...	openfoodfacts-contributors	1608035756	2020-12-15T12:35:56Z	1610702583	2021-01-15T09:23:03Z	basilic	panzani
422198	3560070278831	http://world-en.openfoodfacts.org/product/3560...	openfoodfacts-contributors	1381071983	2013-10-06T15:06:23Z	1618645457	2021-04-17T07:44:17Z	Pamplemousse rose, 100 % Pur Fruit Pressé	Carrefour,CMI (Carrefour Marchandises Internat...
422199	3560070278831	http://world-en.openfoodfacts.org/product/3560...	openfoodfacts-contributors	1381071983	2013-10-06T15:06:23Z	1621577199	2021-05-21T06:06:39Z	Pamplemousse rose, 100 % Pur Fruit Pressé	Carrefour,CMI (Carrefour Marchandises Internat...

4 rows × 49 columns

Nous avons 2 observations en double pour supprimer.

```
# Drop duplicated items
df_food.drop_duplicates(subset=['code'],keep='last', inplace=True)
```

Valeurs vides :



La prochaine étape: supprimer tous les colonnes avec suffixes ('_t', '_datetime', '_tags', '_serving', '_url', 'serving_size').

```
endwith_columns = ['_t', '_datetime', '_tags', '_serving', '_url', 'serving_size', '_en']

for col in df_food_no_dup.columns:

    for pattern in endwith_columns:

        if col.endswith(pattern):
            del df_food_no_dup[col]
            break
```

Filtre sur le pays:

En continu les nettoyages de données sur les produits vendus en France.

```
# Filtrage des produits vendus en france uniquement
contains_fr = df_food_no_dup.countries.str.contains("fr").fillna(False)
contains_France = df_food_no_dup.countries.str.contains("France").fillna(False)
df_food_fr = df_food_no_dup[ contains_France|contains_fr]
```

```
df_food_fr.shape
```

```
(268187, 32)
```

Apres comparaison de données on garde le values plus intéressant pour notre projet.

'code', 'product_name', 'categories', 'additives_n', 'nutriscore_grade', 'ecoscore_score_fr', 'ecoscore_grade_fr', 'energy-kcal_100g', 'energy_100g', 'fat_100g','saturated-fat_100g', 'carbohydrates_100g', 'sugars_100g', 'fiber_ 100g', 'proteins_100g', 'salt_100g', 'sodium_100g', 'nutrition_score_fr_ 100g.'

Valeurs atypiques et aberrantes :

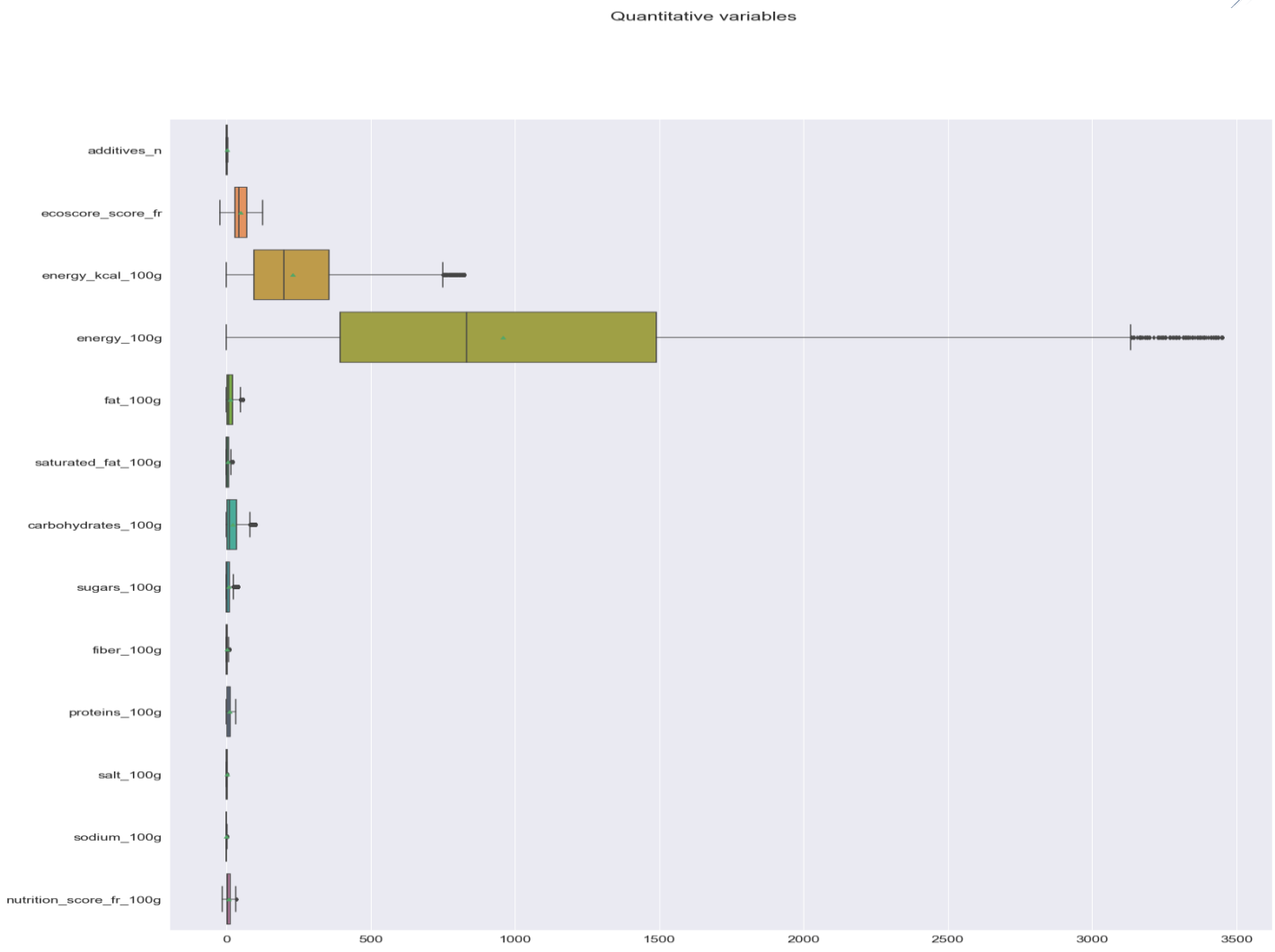
```
FR_food_filter2.loc[FR_food_filter2['product_name'].isin(['Sugar','sugar','Salt','salt'])]
```

code	product_name	categories	additives_n	nutriscore_grade	ecoscore_score_fr	ecoscore_grade_fr	energy_kcal_100g	energy_100g	fat_100g	saturated_fat_1

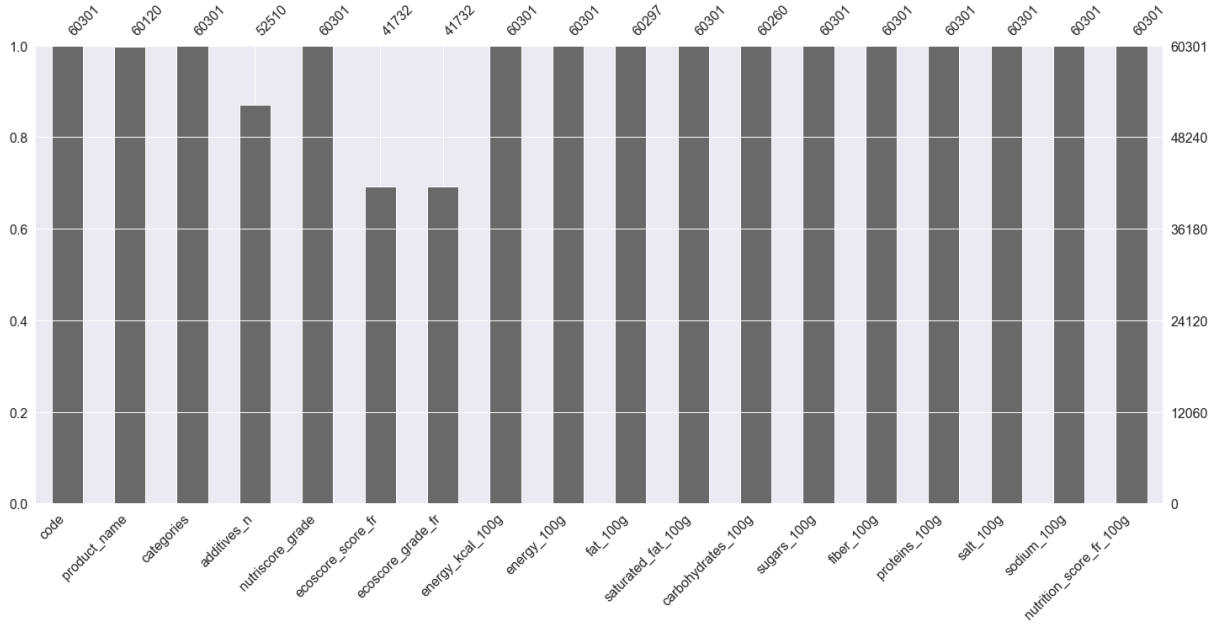
Il n'y a pas valeurs atypiques donc on va vérifier les valeurs aberrantes.

Chercher des valeurs aberrantes :

On va supprimer les fausses données pour les colons 'energy_100g' et 'energy_kcal_100g'. En suit on va supprimer les valeurs aberrantes à l'aide de **écart interquartile**.



Valeurs manquantes



Puisque notre application est destinée aux enfants, nous gardons juste le produit avec minimum additif (0, 1,2).

Nous avons réduit le data frame sur 76228 observations et 16 colons.

Maintenant essayer de faire une liste de groupe de la variable 'catégories' en 10 groups, 'Boissons', 'Snacks', 'Protides', 'Fruits & légumes', 'laitier', 'Biscuits & Desserts', 'Plats préparés', 'minéraux', 'Céréale', 'vitamines'.

```
df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('protéine|Viande|viande|poisson|Poisson|poulet|Produits de
df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('lait|Lait|yaourt|Yaourt|fromage|Fromage'), 'categories'] =
df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('Fruit|légume|végé|origen vegetal|Légume|Champignon|champig
df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('vitamine|Vitamine'), 'categories'] = 'vitamines'
df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('fer|Fer'), 'categories'] = 'minéraux'

df_food_fr_cat.loc[df_food_fr_cat['categories'].str.contains('noix|Noix|Graines|graines| '), 'categories'] = 'lipides'
```

```
categories_to_keep = ['Protides', 'laitier', 'Fruits & légumes','vitamines', 'minéraux','lipides']
df_food_fr_New = df_food_fr_cat[df_food_fr_cat.categories.isin(categories_to_keep)]
df_food_fr_New.shape
```

```
(23846, 15)
```

Après toute manipulation des ensembles de données, nous devons remplir les valeurs manquantes de 'carbohydrates_100g', 'fat_100g', 'nutriscore_grade', 'energy_100g'.

KNN Imputer

Utiliser l'algorithme d'apprentissage automatique knn pour remplir les valeurs manquantes.

```
from sklearn.impute import KNNImputer
```

```
imputer = KNNImputer(n_neighbors=3)
data_with_null = pd.DataFrame(imputer.fit_transform(data_with_null),columns = data_with_null.columns)
```

```
data_with_null.head()
```

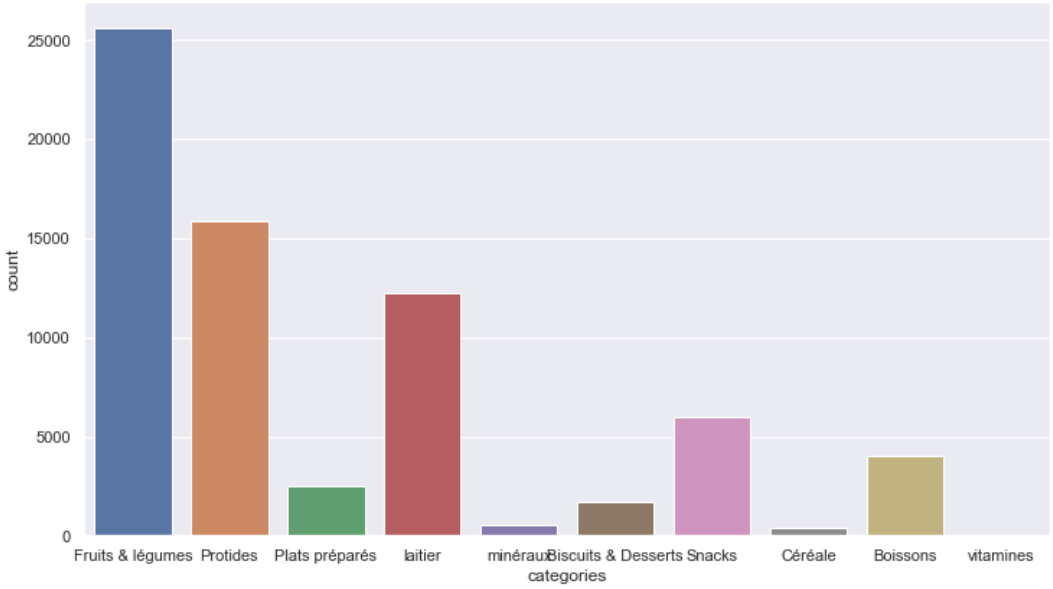
	carbohydrates_100g	fat_100g	nutriscore_grade	energy_100g
0	11.7	53.2	4.0	2318.0
1	17.0	37.0	4.0	1707.0
2	0.0	10.3	0.0	703.0
3	24.0	3.1	2.0	573.0
4	4.8	0.1	0.0	134.0

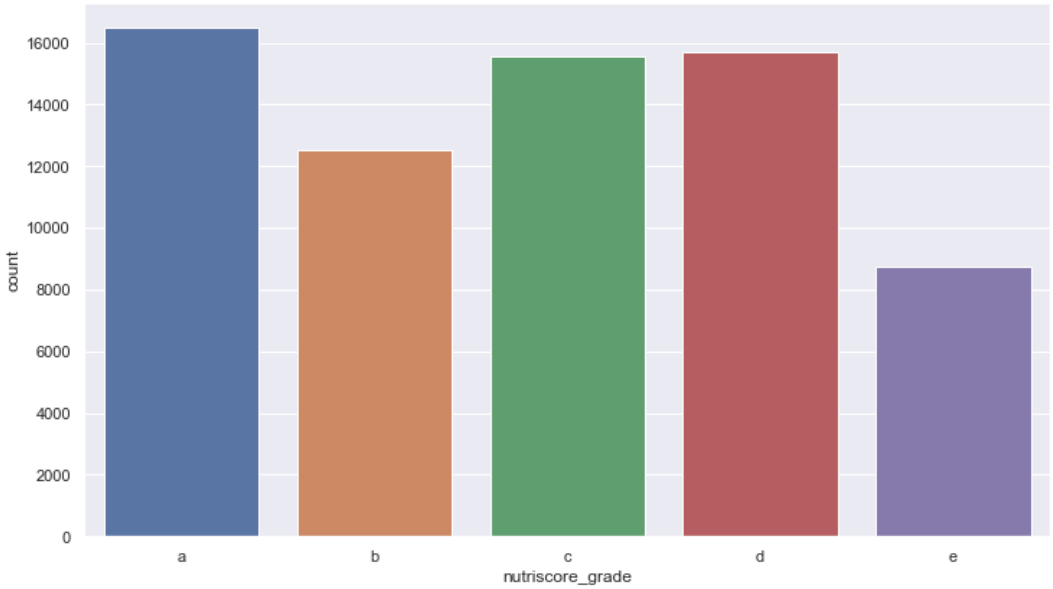
Maintenant, nous avons des ensembles de données propres afin que nous puissions commencer la partie exploratoire de l'analyse des données.

Analyse Exploratoire

Analyse Univariée

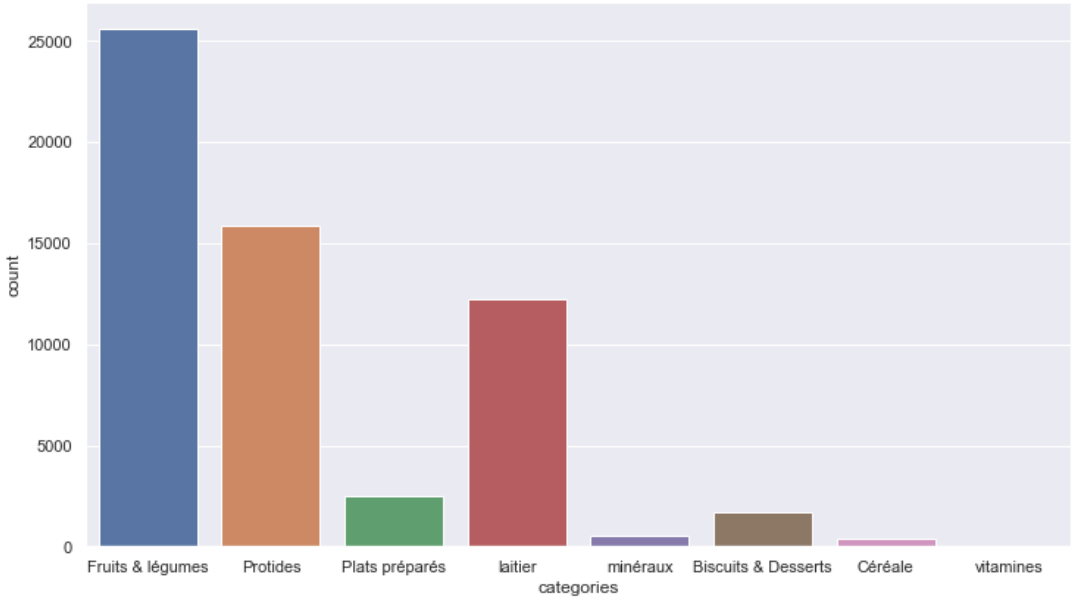
Valeurs qualitative

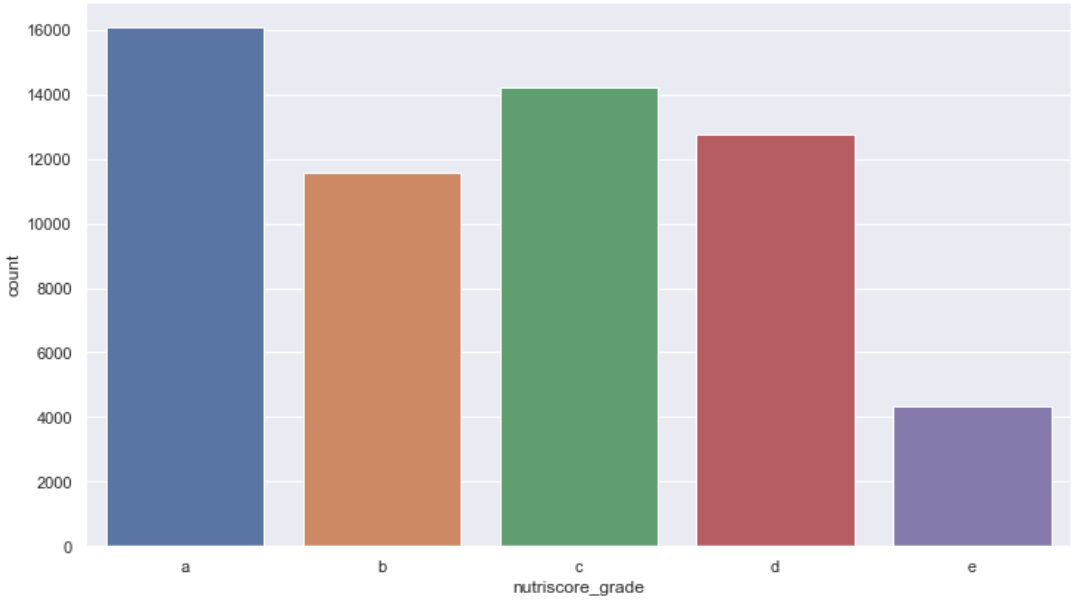




C'est clair dans toutes nos catégories que nous avons en colon 'categories', nutriscore_grade catégories 'e' est plus 8000 donc on supprime les catégories que sont pas très bon pour les enfants ('Boissons', 'Snacks').

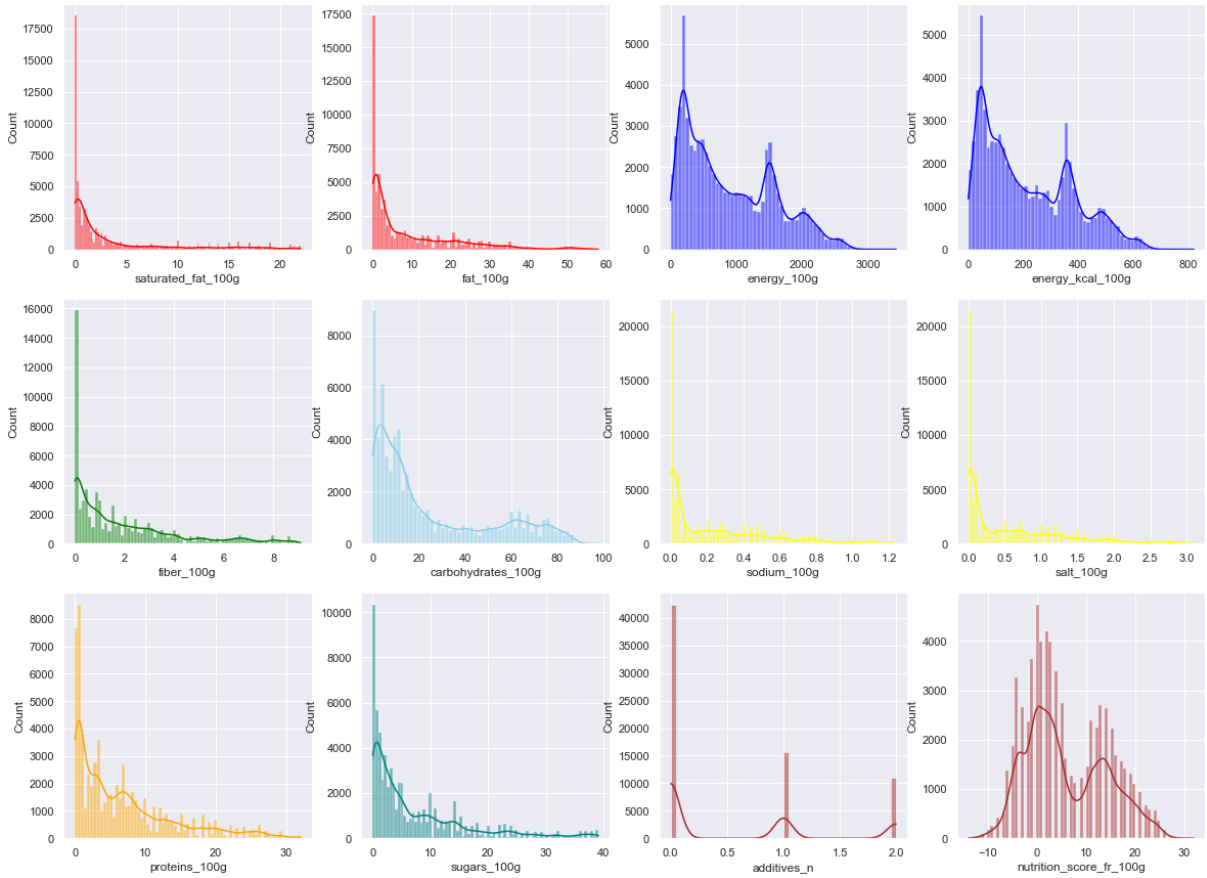
Note : les Boisson ici considéré comme un catégorise mauvaise parce que consiste tout type de boissons (coca, alcooliques et non acholiques et …)





Ici c'est clair que les 2 catégories 'Boissons', 'Snacks' sont catégorisé comme mauvaise produit pour la santé.

Valeurs quantitative

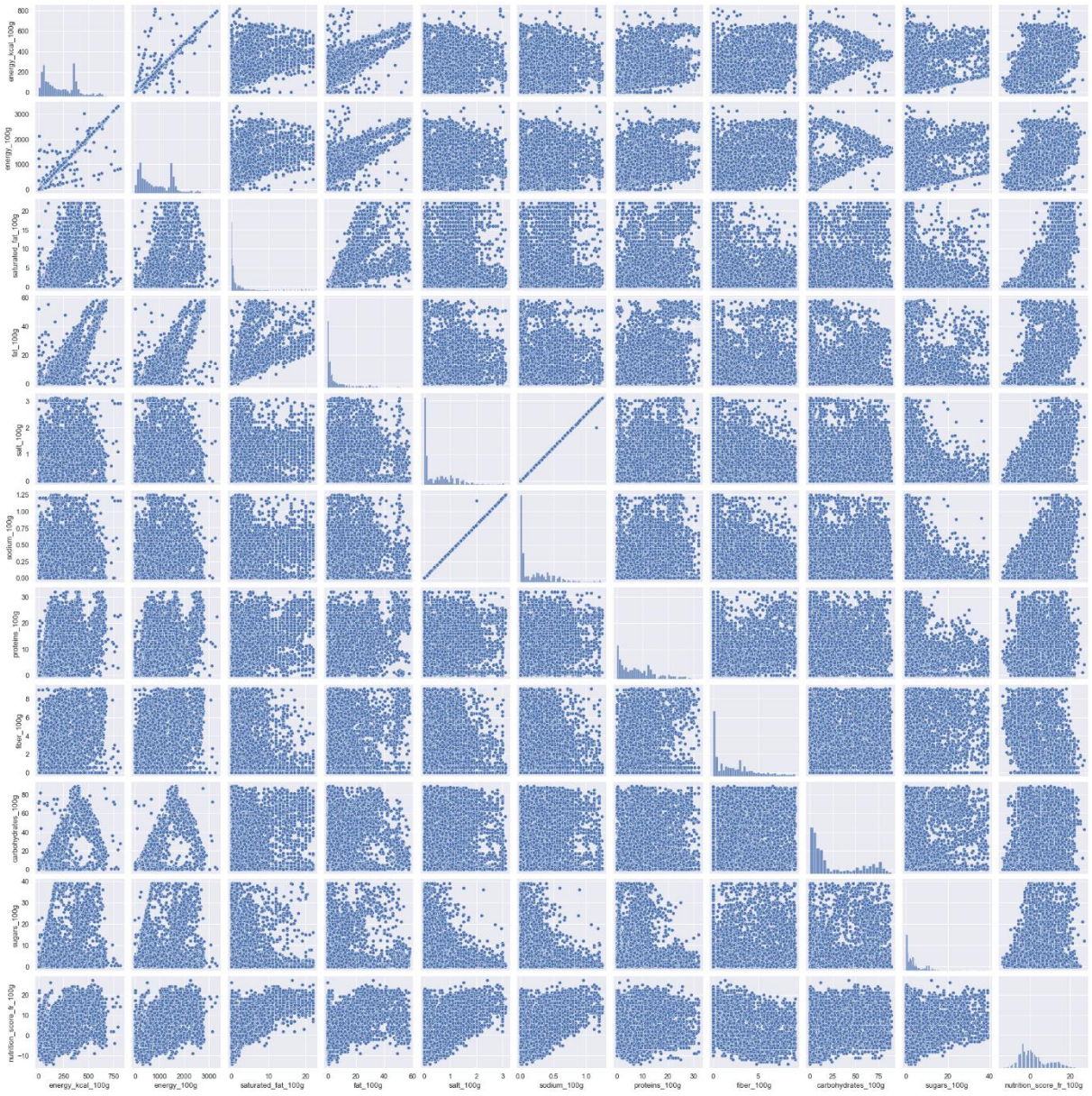


Comme nous pouvons le voir, la plupart de votre graphique a une distribution asymétrique.

Analyse Bivariée

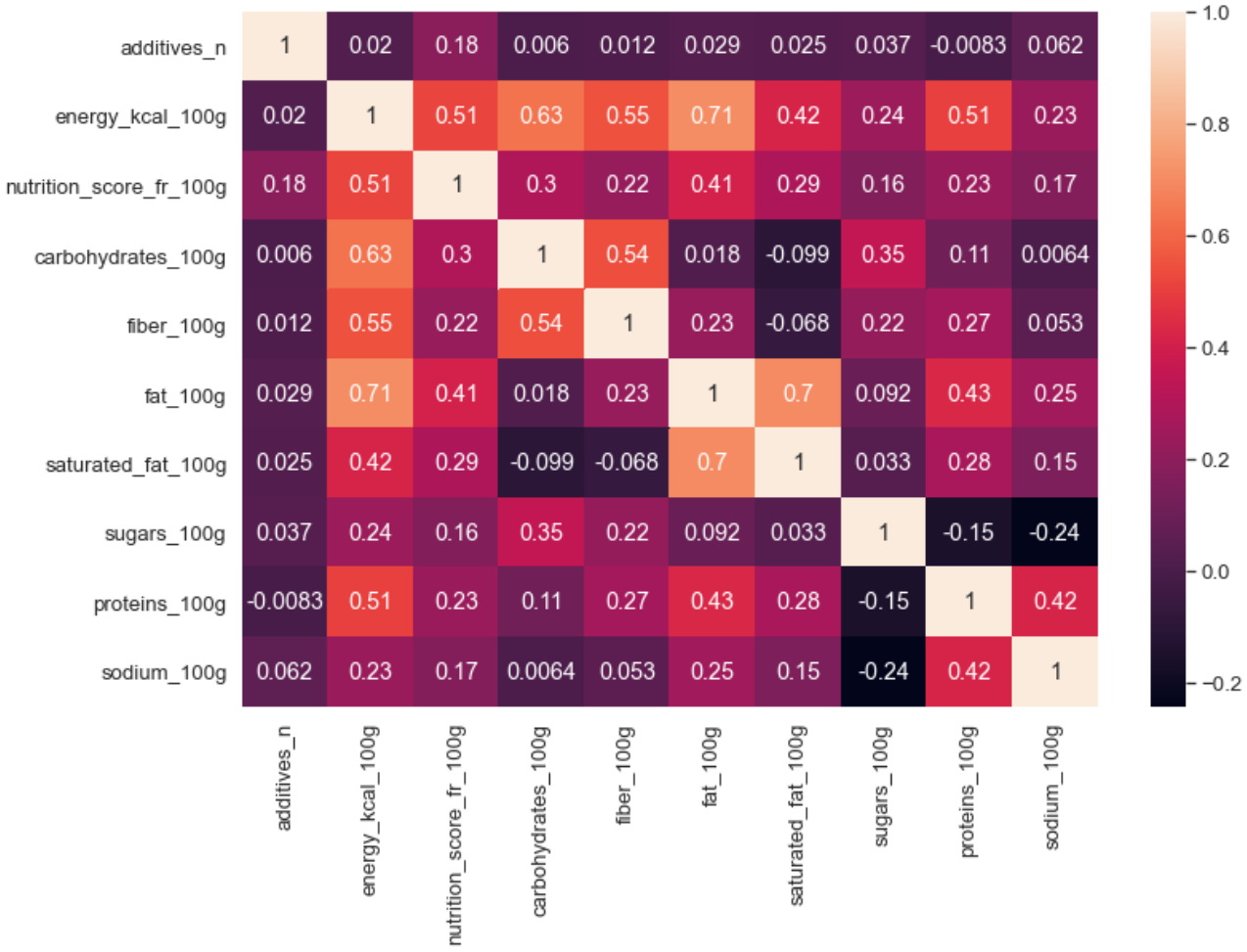
L'analyse bivariée, comme son nom l'indique, a pour objectif d'analyser le lien qui peut exister entre deux variables. Nous utilisons **Pairplot** et **Heatmap** pour montrer la relation entre chacune des deux variables de notre base de données.

Pairplot



On peut constater que le sel et le sodium représente une droite parfaite. Cela veut dire que nous pouvons supprimer un des deux indicateurs. Et on peut faire le même constat entre les gras et les gras saturés.

Corrélation Heatmap



C'est clair qu'il existe une forte corrélation positive entre 'fat_100g' et 'energy_kcal_100g' avec coefficient de corrélation de : 0.71

Et pareil entre 'fat_100g' et 'saturated_fat_100g' avec coefficient de corrélation de : 0.7

Et antre de 'carbohydrates_100g' et 'energy_kcal_100g' avec coefficient de corrélation de : 0.63

'fiber_100g' et 'carbohydrates_100g' ont une corrélation moyenne avec le coeff : 0.54

'fiber_100g' et 'energy_kcal_100g' ont une corrélation moyenne avec le coeff : 0.55

'proteins_100g' et 'energy_kcal_100g' ont une corrélation moyenne avec le coeff : 0.51

Analyse ACP

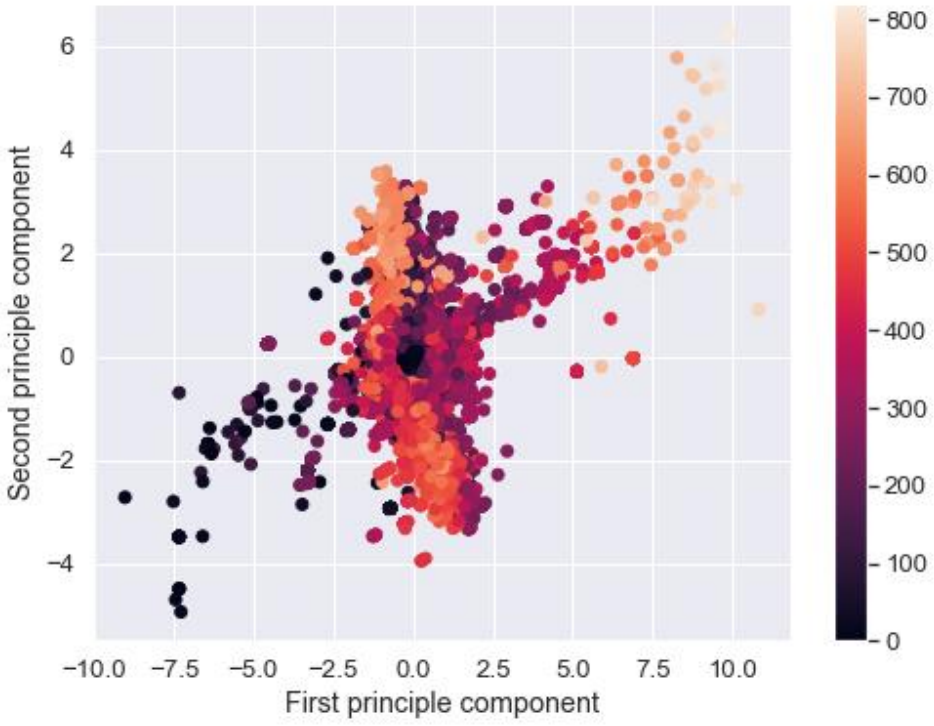
L'ACP doit être utilisé principalement pour les variables fortement corrélées. Si la relation est faible entre les variables, l'ACP ne fonctionne pas bien pour réduire les données. Référez-vous à la matrice de corrélation pour déterminer. En général, si la plupart des coefficients de corrélation sont inférieurs à 0,3, l'ACP n'aidera pas.

```
from sklearn.decomposition import PCA,TruncatedSVD
```

```
pca = PCA(n_components=10, whiten='True')
x = pca.fit(df).transform(df)
```

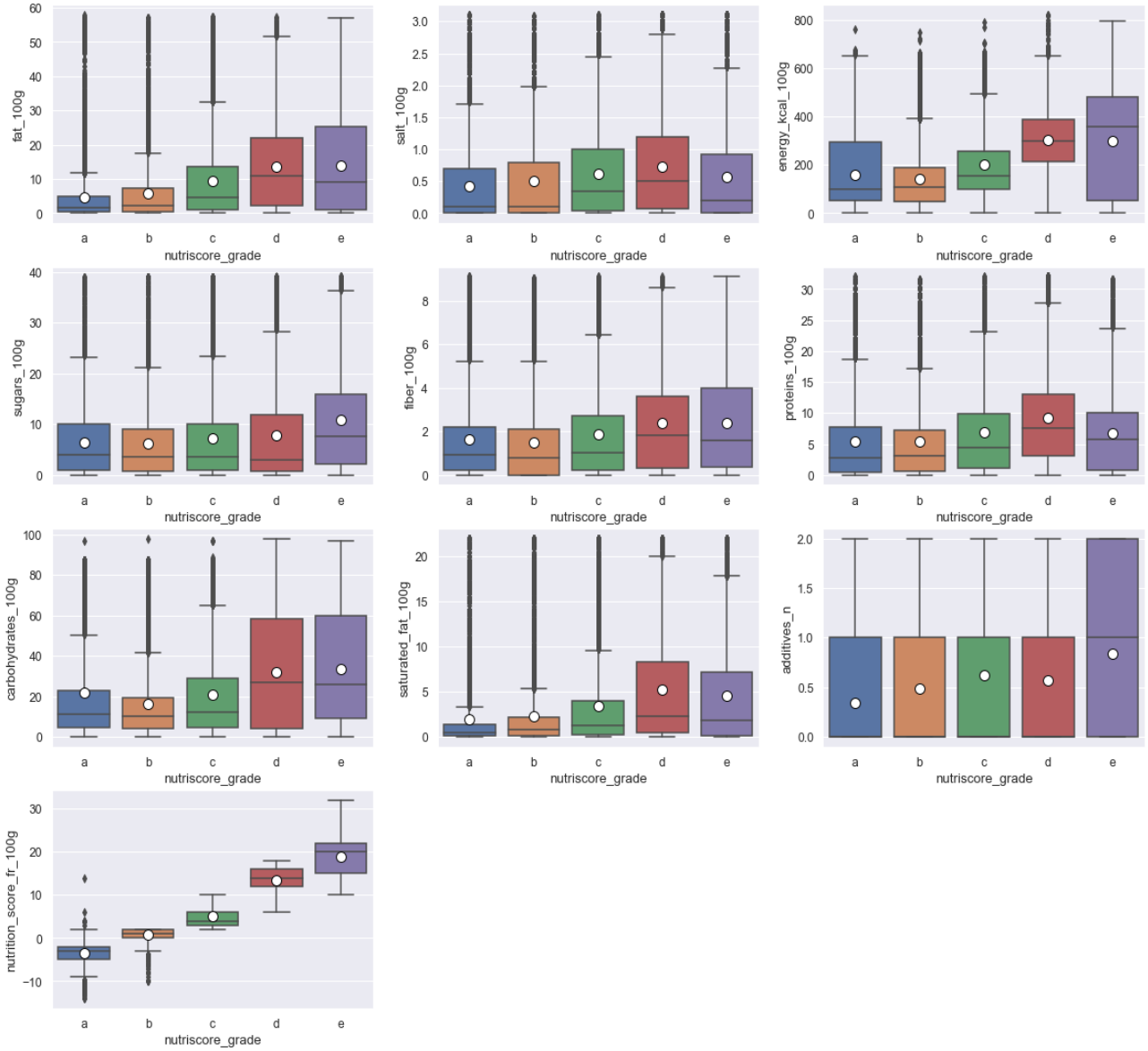
```
pca = PCA(n_components=2, whiten='True')
x_pca=x = pca.fit(x).transform(x)
```

Ici nous choisissons 'energy_kcal_100g' car il a une forte corrélation avec d'autres variables.



ANOVA

La méthode ANOVA est une analyse de la covariance entre deux variables (quantitative et qualitative).



L'illustration parfaite de l'ANOVA est le graphique du Nutriscore.

On constate que l'énergie, les gras, les gras saturés, les sucres et les fibres semblent dépendre du Nutriscore. Par contre, les féculents, les protéines et le sel ne semblent pas avoir de lien avec le Nutriscore.

Ces différentes analyses nous indiquent que pour construire un modèle de prédiction, nous pourrons nous appuyer sur les indicateurs suivants :

L'énergie, les gras, les gras saturés, les sucres et les fibres.

```
model = ols('nutriscore_grade ~ saturated_fat_100g', data=df_sort).fit()
aov_table = sm.stats.anova_lm(model, typ=2)
aov_table
```

	sum_sq	df	F	PR(>F)
saturated_fat_100g	6211.227301	1.0	3550.672895	0.0
Residual	120775.855182	69042.0	NaN	NaN

	sum_sq	df	F	PR(>F)
fat_100g	12501.019232	1.0	7538.868446	0.0
Residual	114486.063252	69042.0	NaN	NaN

	sum_sq	df	F	PR(>F)
energy_kcal_100g	17398.695846	1.0	10961.387383	0.0
Residual	109588.386638	69042.0	NaN	NaN

	sum_sq	df	F	PR(>F)
fiber_100g	2801.106371	1.0	1557.29328	0.0
Residual	124185.976113	69042.0	NaN	NaN

	sum_sq	df	F	PR(>F)
sugars_100g	2356.286533	1.0	1305.317306	3.531005e-283
Residual	124630.795951	69042.0	NaN	NaN

L'hypothèse nulle pour une ANOVA est qu'il n'y a pas de différence significative entre les groupes. Ici, la valeur p est inférieure à alpha donc rejet de l'hypothèse nulle. Donc, on conclut que les moyennes de tous les groupes ne sont pas égales.

Conclusion :

- La 'nutriscore_grade' a une relation positive avec les gras, les gras saturés, les sucres et les fibres. cela signifie que tous les produits qui ont un nombre élevé de ces 4 éléments sont classés pour un niveau élevé de nutriscore (d ou e). Donc il faut faire attention à la quantité de consommation de ces éléments

- Tous les produits avec additif égal à 2 sont dans la catégorie e de nuriscore. Le nutriscore est calculé pour les adultes mais pour les enfants le calcul doit être différent. Mais nous sommes sûrs que le produit avec additif plus de 1 est dangereux pour les enfants.