CentraleSupélec

# Projet 3 -70 heures

## Concevez une application au service de la santé publique

L'agence "Santé publique France" a lancé un appel à projet autour des problématiques alimentaires. Vous proposerez une application basée sur des données nutritionnelles.

Réalisé par : Sabrine OUANNES
Date de soutenance : 18 janvier  2022

# Sommaire

# Présentation du projet

L'agence "Santé publique France" a lancé un appel à projets pour trouver des idées innovantes d'applications en lien avec l'alimentation.

Le jeu de données Open Food Facts est disponible sur le site officiel

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.

Santé publique France

## Problématique

Concevoir une idée d'application au service de la santé publique en se basant sur le jeu de données Open Food Facts.

## Objectifs

- Réfléchir à une idée d'application.
- Nettoyer le jeu de données .
- Explorer le jeu de données.

# Sommaire

# Custom Diet

Plusieurs personnes suivront un régime alimentaire soit pour perdre du poids, soit parce qu'ils souffrent d'une certaine maladie ou allergie.
L'application « Custom Diet » permet au consommateur de suivre son régime lors de l'achat d'un produit et lors de sa consommation :

## Mon achat

- L'utilisateur sélectionne le régime alimentaire à suivre.
- L'utilisateur note son poids.
- En scannant le produit l'application enregistre le produit et rend :
  - les valeurs nutritionnelles adéquates au régime alimentaire sélectionné selon le poids.
  - le repère nutritionnel pour 100g du produit scanné
  - Le nutriscore
  - Proposer des produits similaires.
  - Enregistrer le produit acheté.

## Mon suivi quotidien

- L'utilisateur indique la quantité utilisée du produit acheté
- Calculer le reste des valeurs nutritionnelles à consommer par jour .

# Modèle de l'application (les entrées)

## Mon achat

1 - Sélectionner le régime

- Kéto
- Dukan
- Sans Lactose
- Équilibré
- Paléo
- Anti-cellulite
- Équilibré
- Hypocalorique

2- Entrer le poids (en Kg) → 70 Kg

3- scanner le produit →

4- Enregistrer le produit en cas d'achat →

## Mon suivi quotidien

Sélectionner le produit à enregistré à consommer →

Entrer le poids du produit (en Kg) → 30 g

L'application nous renvoie

# Mon achat

**Régime Kéto**

- Matières grasses en g : 210
- Sucres en g : 15
- Protéines  en g : 70

**Repère nutritionnels pour 100g**

- Matières grasses en g : 70
- Sucres en g : 2
- Protéines  en g : 10
- Apport calorifique en (Calories) : 678


**Produits similaires**




**Est-ce que vous allez acheter ce produit?**

OUI  NON

# Mon suivi quotidien

**Il vous reste ce jour :**

- Matières grasses en g : 170
- Sucres en g : 6
- Protéines  en g : 40

# Sommaire

# 3 - Présentation et Nettoyage du jeu données

Pour nettoyer le « dataset », nous avons utilisé Jupyter notebook comme outil.

Les grandes parties abordées sont :

- Suppression des lignes et des colonnes mal renseignées ou redondantes
- Gestion des valeurs aberrantes
- Imputation des valeurs manquantes

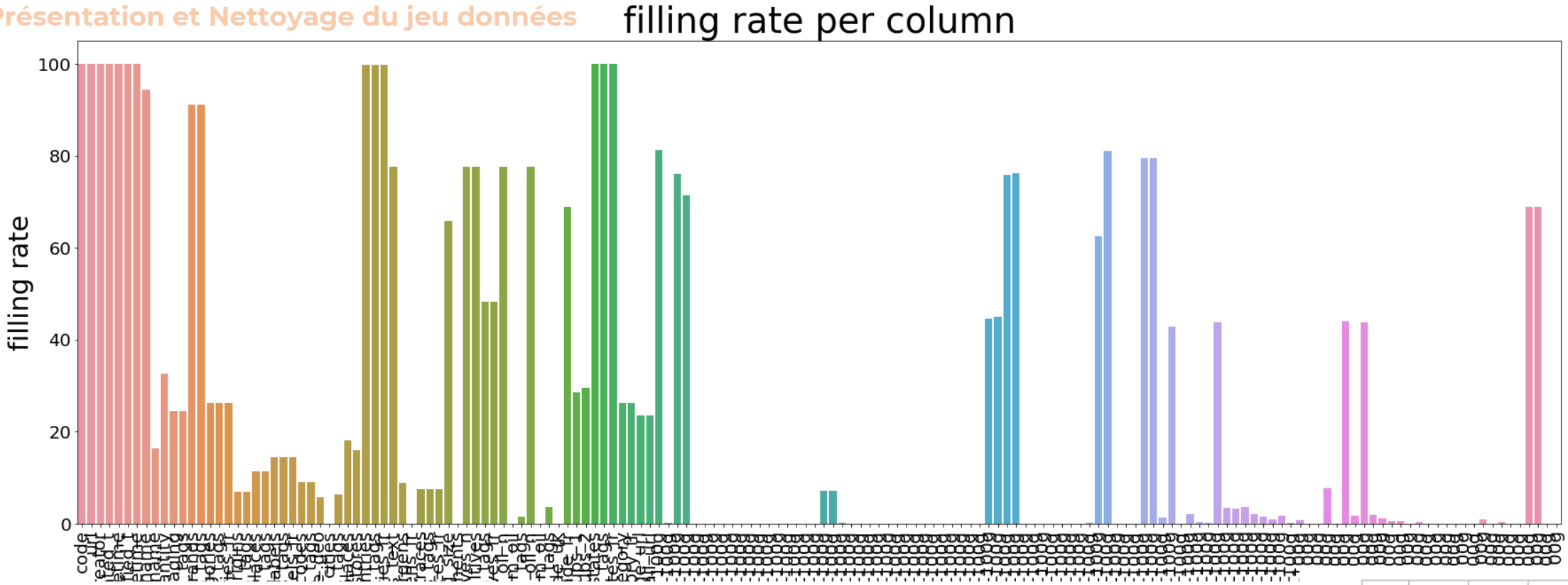
# Présentation du dataset

Le jeu de données est téléchargeable sur le site officiel de OPEN FOOD FACTS

**162 colonnes : 106 object et 56 float**

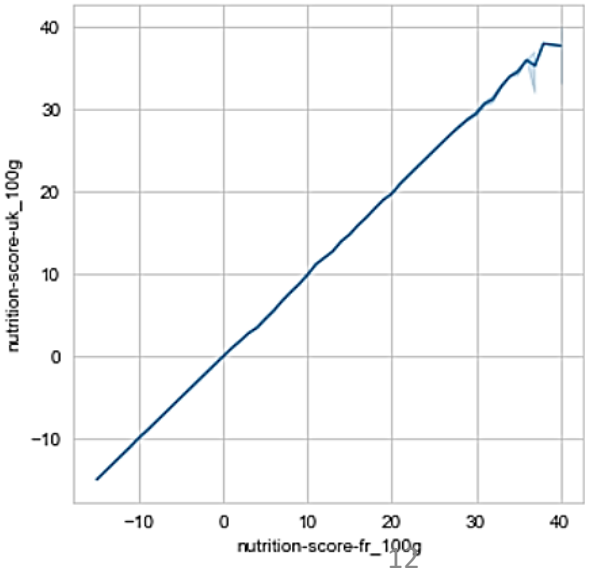
**320772 lignes**

| | additives_n | ingredients_that_may_be_from_palm_oil_n | energy_100g | nutri_score | fat_100g | saturated-fat_100g | carbohydrates_100g | sugars_100g | fiber_100g | pro |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 1 | 0.0 | 0.0 | 2243.0 | 14.0 | 28.57 | 28.57 | 64.29 | 14.29 | 3.6 | |
| 2 | 0.0 | 0.0 | 1941.0 | 0.0 | 17.86 | 0.00 | 60.71 | 17.86 | 7.1 | |
| 3 | 0.0 | 0.0 | 2540.0 | 12.0 | 57.14 | 5.36 | 17.86 | 3.57 | 7.1 | |
| 4 | 0.0 | 0.0 | 1552.0 | NaN | 1.43 | NaN | 77.14 | NaN | 5.7 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 320767 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 320768 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 | |
| 320769 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 320770 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 320771 | 7.0 | 0.0 | 2092.0 | NaN | 0.00 | NaN | 0.00 | 0.00 | NaN | |




- Des colonnes entièrement vides ou très peu renseignées
- Il y a des variables qui se répètent
- Il y a des variables redondantes comme : url ou states …

→
- Supprimer les colonnes ayant un taux de valeurs manquantes supérieur à 50 % afin de ne pas biaiser notre jeu de données.
- Supprimer les colonnes en double



## Gestion des valeurs aberrantes

En décrivant le jeu de données, les valeurs minimum et maximum de certaines variables sont aberrantes.
En effet, les seuils à respecter sont :

- Pour 100 g d'aliment les quantités de nutriments ne peuvent pas dépasser 100g.
- Saturated-fat 100g, sugars_ 100 g et sodium_100g ne peuvent pas dépasser fat_100g, carbohydrates_100g et salt_100 g respectivement.
- La valeur d'énergie ne peut pas excéder 3800 Calories pour 100 g.
- La somme des nutriments ne peut pas dépasser les 100g.

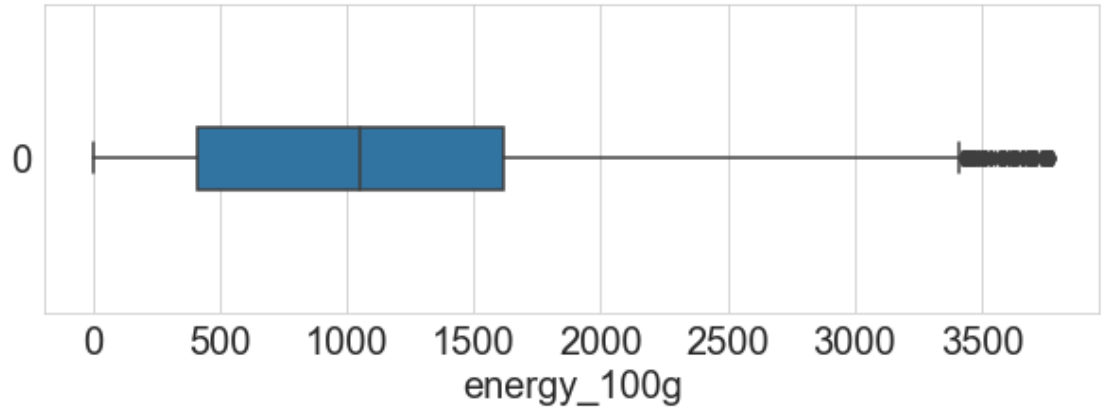
| | additives_n | ingredients_that_may_be_from_palm_oil_n | energy_100g | fat_100g | saturated-fat_100g | carbohydrates_100g | sugars_100g |
|---|---|---|---|---|---|---|---|
| count | 248939.00000000 | 248939.00000000 | 2.61113000e+05 | 243891.00000000 | 229554.00000000 | 243588.00000000 | 244971.00000000 |
| mean | 1.93602449 | 0.05524647 | 1.14191460e+03 | 12.73037857 | 5.12993230 | 32.07398108 | 16.00348355 |
| std | 2.50201947 | 0.26920744 | 6.44715409e+03 | 17.57874669 | 8.01423814 | 29.73171946 | 22.32728440 |
| min | 0.00000000 | 0.00000000 | 0.00000000e+00 | 0.00000000 | 0.00000000 | 0.00000000 | -17.86000000 |
| 25% | 0.00000000 | 0.00000000 | 3.77000000e+02 | 0.00000000 | 0.00000000 | 6.00000000 | 1.30000000 |
| 50% | 1.00000000 | 0.00000000 | 1.10000000e+03 | 5.00000000 | 1.79000000 | 20.60000000 | 5.71000000 |
| 75% | 3.00000000 | 0.00000000 | 1.67400000e+03 | 20.00000000 | 7.14000000 | 58.33000000 | 24.00000000 |
| max | 31.00000000 | 6.00000000 | 3.25137300e+06 | 714.29000000 | 550.00000000 | 2916.67000000 | 3520.00000000 |

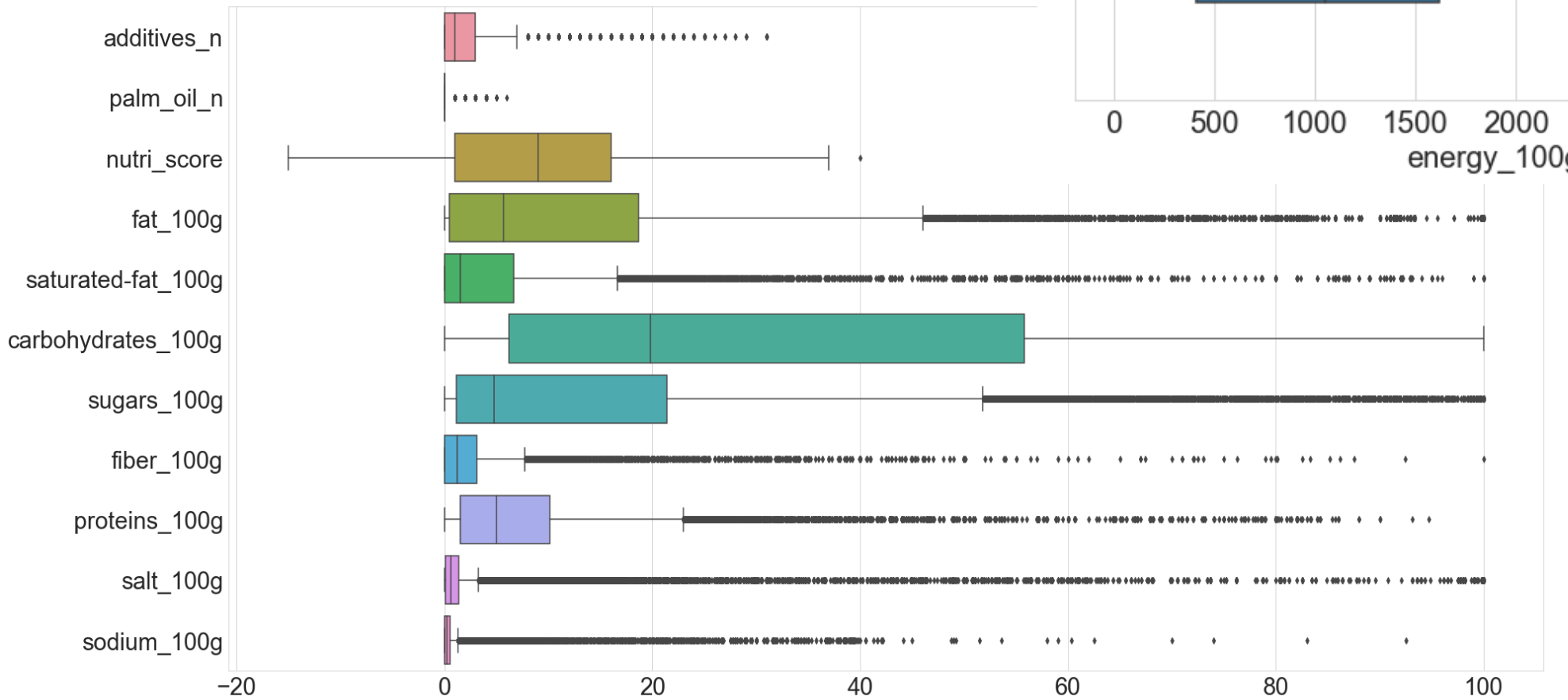
→ Remplacer les valeurs aberrantes par NaN dans les lignes.

Supprimer les lignes ayant plus de 50% de données nutritionnelles vides.

## Gestion des valeurs aberrantes

- À l'aide de ces boites à moustaches, nous avons pu vérifier qu'il n'y a plus de valeurs aberrantes pour les variables quantitatives.
- Les boxplots de distribution des différentes variables montrent que les médianes et les variances sont très différentes.
- Présence de beaucoup d'outliers.



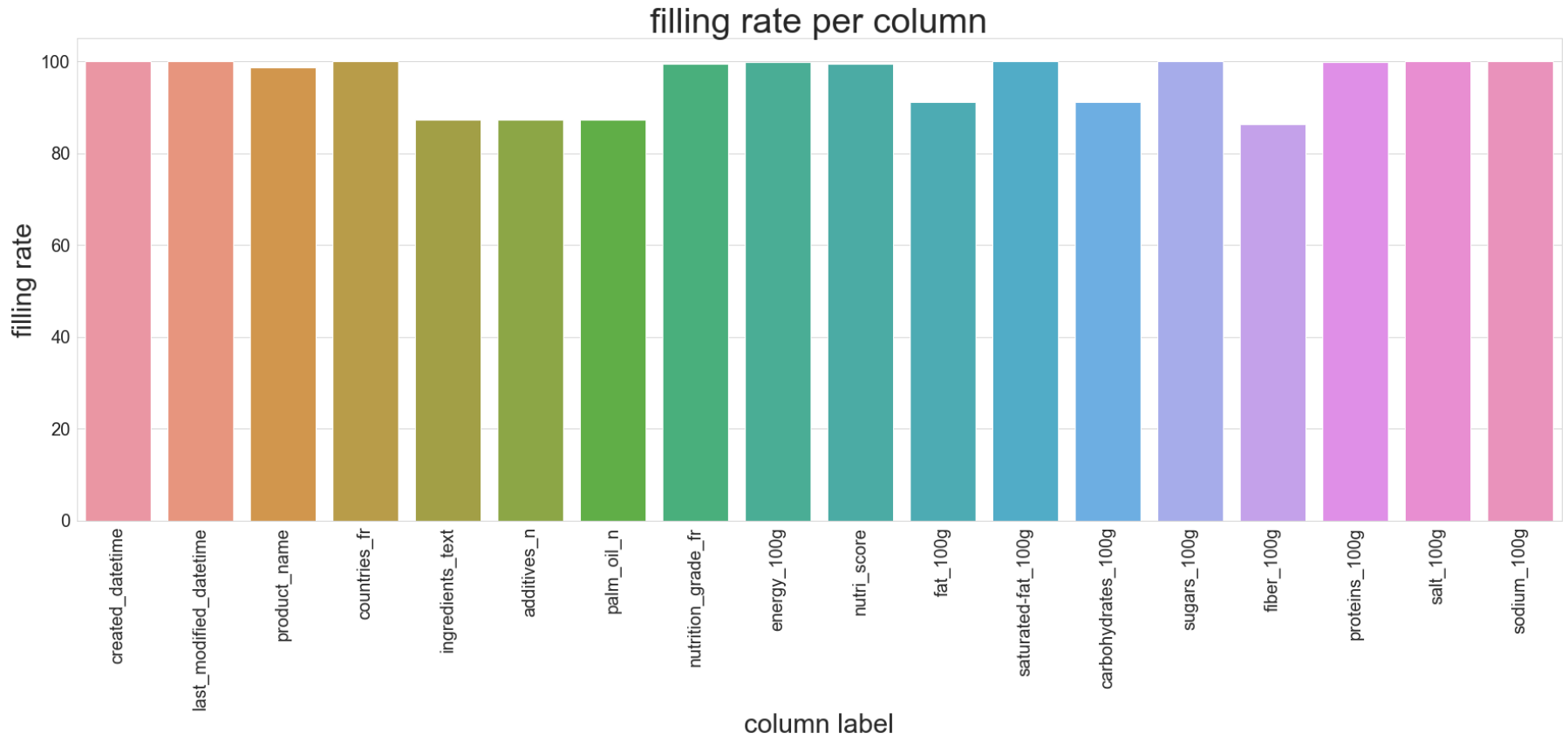



## Synthèse

- Supprimer les lignes et les colonnes mal renseignées
- Supprimer les doublons
- Supprime les variables redondantes
- Des nettoyages ont été également effectuées sur les types de variables

```
created_datetime  object                created_datetime         datetime64[ns, UTC]
last_modified_t   object                last_modified_datetime   datetime64[ns, UTC]
```

A ce niveau notre « dataset » contient 193708 lignes et 18 colonnes



## Imputation des valeurs manquantes

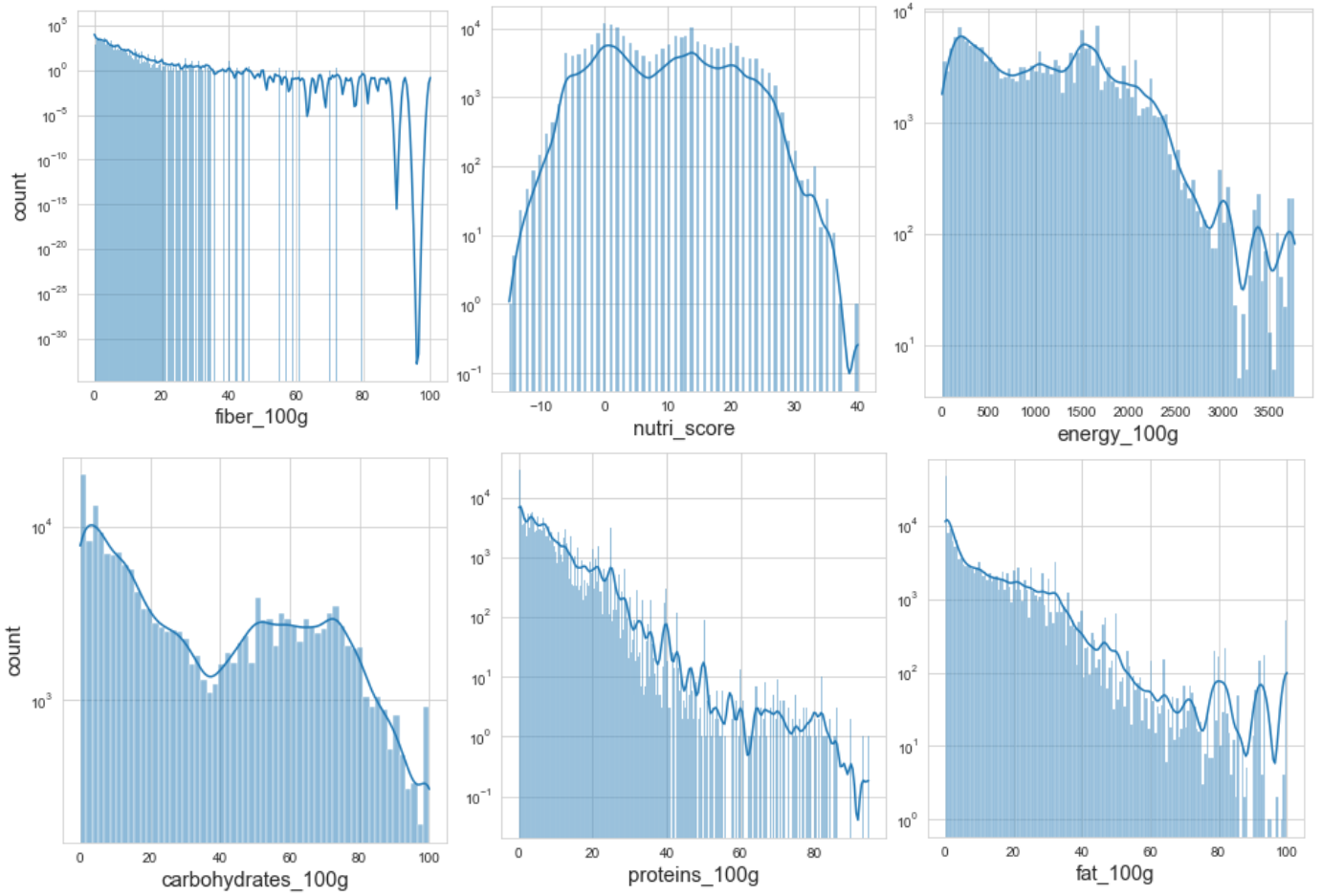
Il existe plusieurs méthodes pour imputer les valeurs manquantes. Pour choisir la méthode adéquate, il faut commencer par regarder la distribution des valeurs des variables à remplacer.

D'après les histogrammes, on n'a pas de distribution normales pour les variables

Un test de normalité de **Kolmogorov-Smirnov** a été fait pour différentes variables.

L'hypothèse nulle H0 de la distribution normale a été rejetée puisque p value <0.05.

Il sera alors incorrect d'imputer les valeurs manquantes par la moyenne.




**Résultats Kolmogorov-Smirnov**

```
energy_100g KstestResult(statistic=0.9894646970200001, pvalue=0.0)
nutri_score KstestResult(statistic=0.6977570317129405, pvalue=0.0)
fat_100g KstestResult(statistic=0.6655217932746444, pvalue=0.0)
carbohydrates_100g KstestResult(statistic=0.879222944048664, pvalue=0.0)
fiber_100g KstestResult(statistic=0.5, pvalue=0.0)
proteins_100g KstestResult(statistic=0.7077590280172181, pvalue=0.0)
```

# Imputation des valeurs manquantes

| Simple imputer | Iterative imputer |
| --- | --- |
| Remplir les NaN de fiber_100g par « 0 » puisque la majorité des aliments ne contiennent pas de fibres | Pour des variables corrélées |

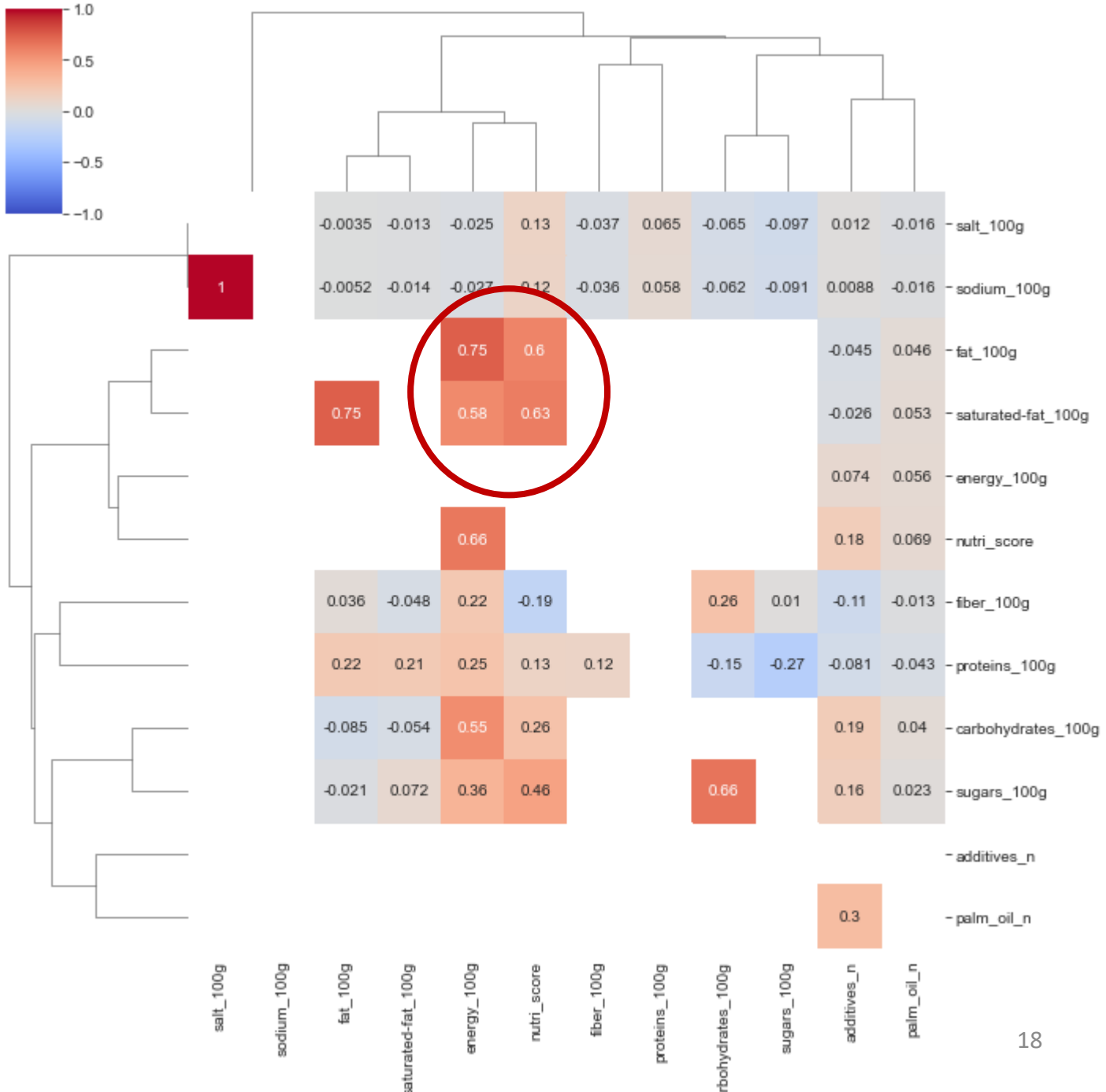
| kNN imputer | kNN classification |
| --- | --- |
| Pour des variables quantitatives | Pour des variables qualitatives |

## Iterative imputer

Cette méthode est utilisée pour imputer des variables qui sont corrélées entre elle.

D'après l'analyse du clusterheatmap avec les coefficients de Pearson affichés , on va essayer de remplacer les valeurs manquantes des variables quantitaives :

- fat_100g
- saturated_fat_100g
- energy_100g
- nutri_score

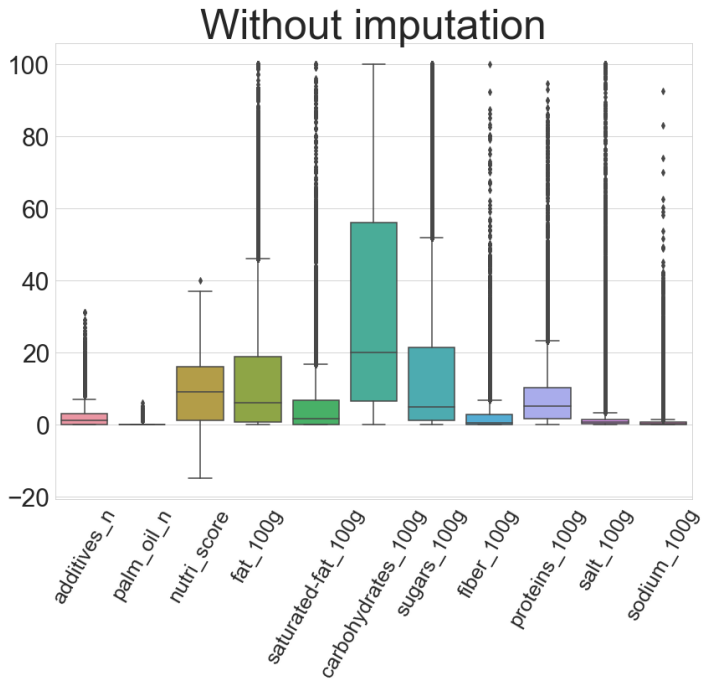


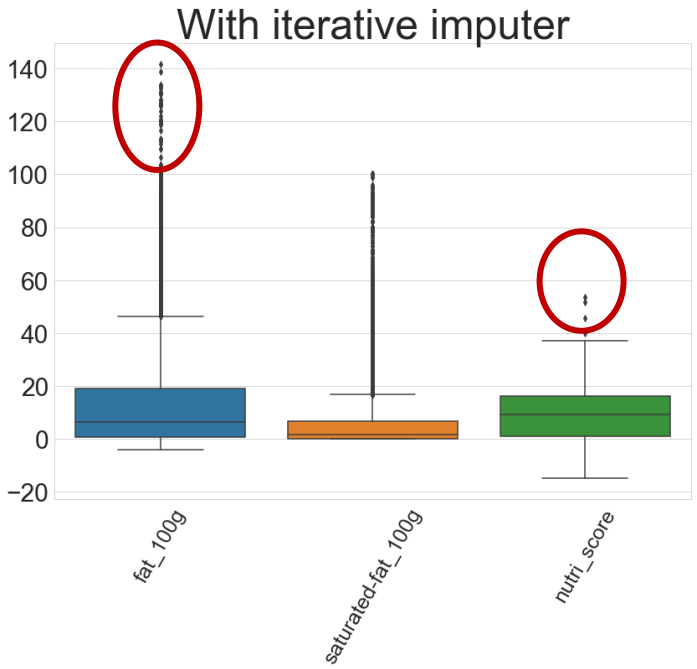
## Iterative imputer

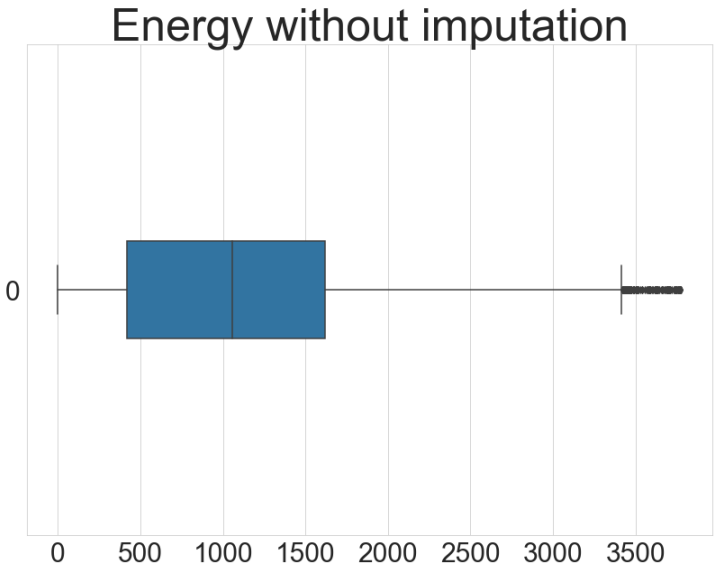
On remarque qu'après imputation des valeurs manquantes par la méthode « iterative imputer » des valeurs aberrantes apparaissent :

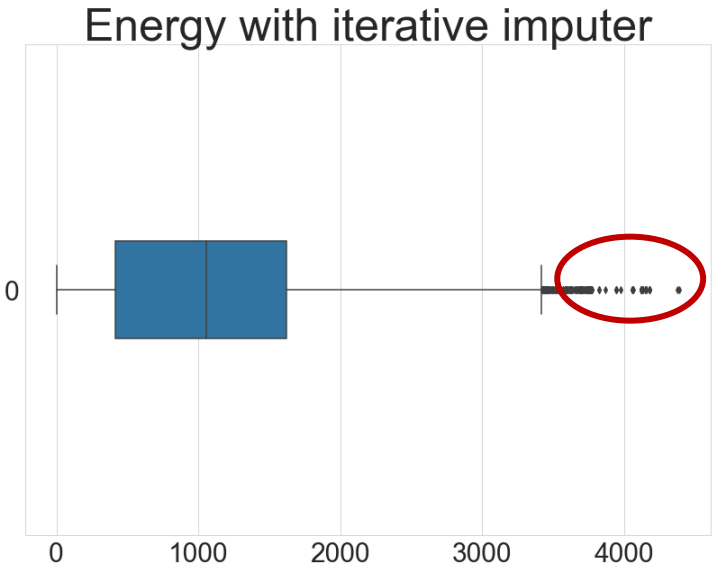
- Pour fat_100 g, on trouve des valeurs supérieures à 100 g
- Le nutriscore doit être compris entre -15 et 40
- La valeur de l'énergie ne doit pas dépasser 3800 kJ

Par conséquent, nous allons éliminer cette méthode.



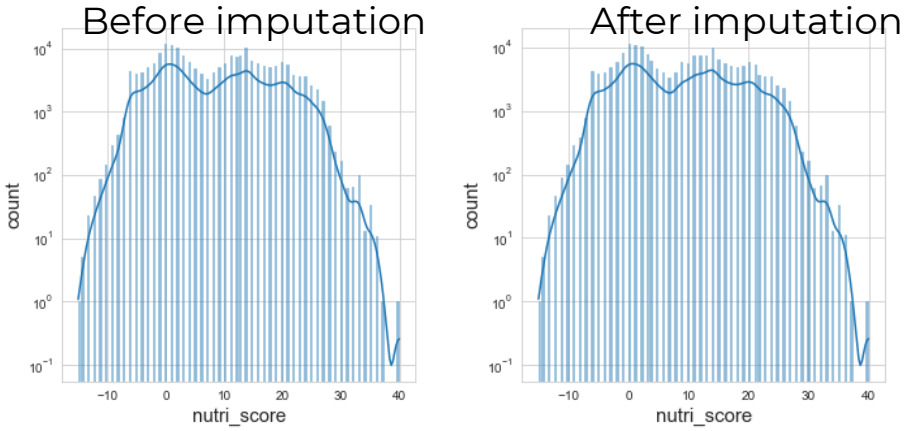



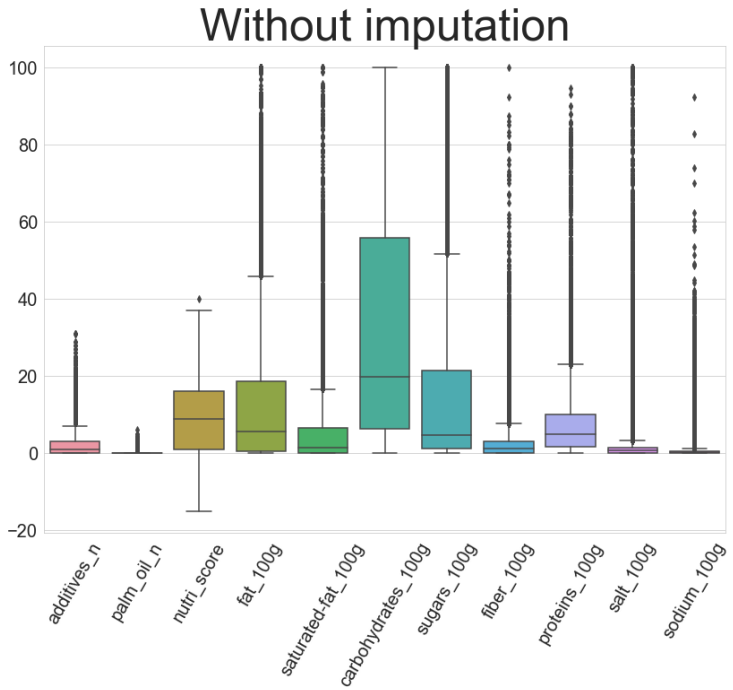



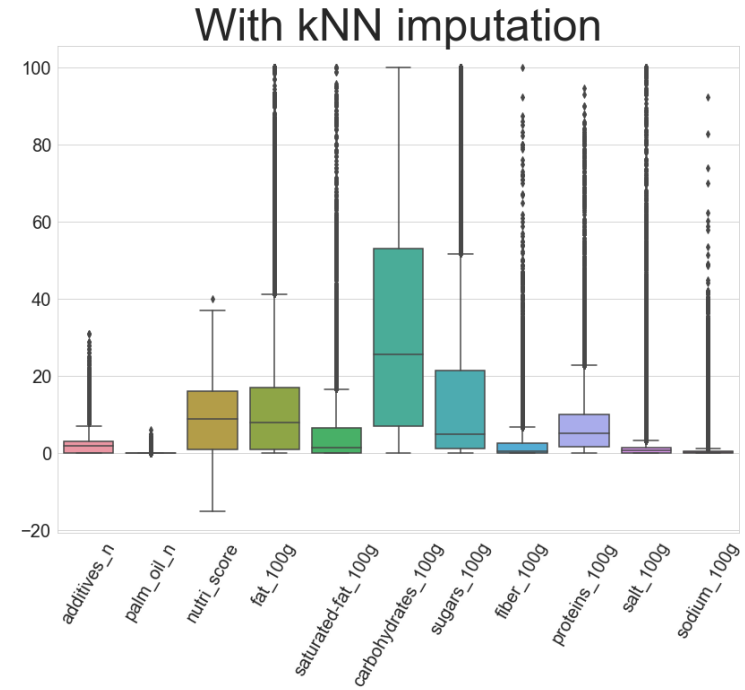
# kNN imputation

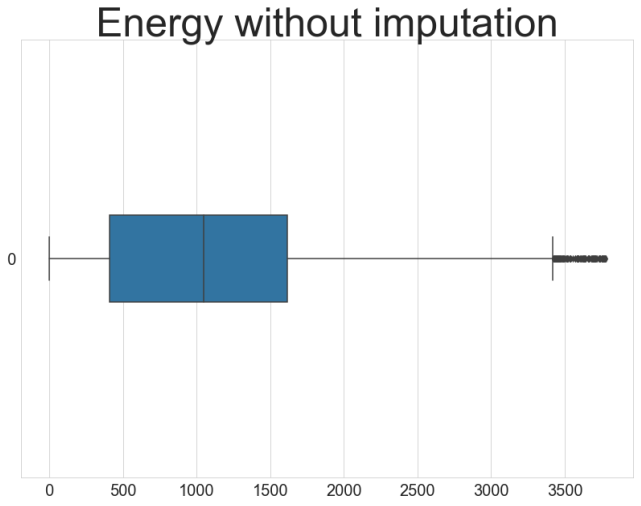
- Un algorithme de Machine Learning qui appartient à la classe des algorithmes d'apprentissage supervisé. Les valeurs manquantes de chaque échantillon sont imputées à l'aide de la valeur moyenne des k_voisins les plus proches.

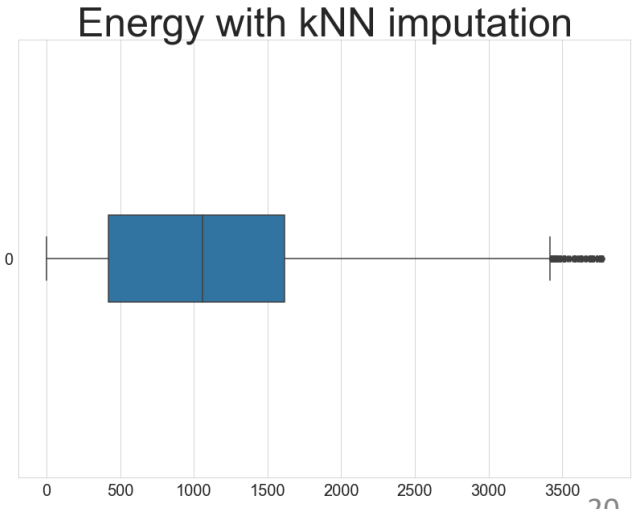
- Les boites à moustaches et les histogrammes montrent que les distributions des différentes variables nutritionnelles après imputation n'ont pas été modifiées comparativement à celles d'origine.



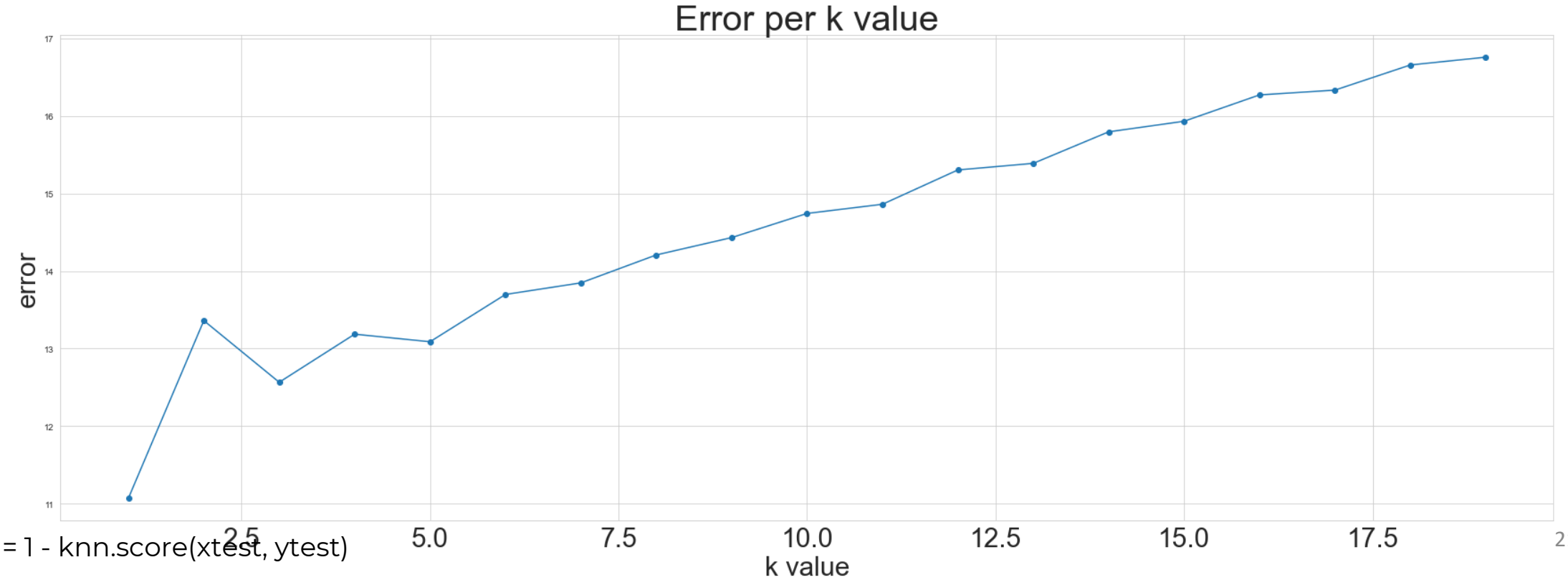



# kNN Classification

Ce modèle est utilisé pour remplacer les valeurs manquantes des variables qualitatives, dans notre cas le nutrigrade classé de A à E.

Nous testons à présent l'erreur de notre classifieur. La méthode *score* permet de tester les performances de prédiction d'un classifieur dans lequel on passe un jeu de données annoté. Il renvoie ainsi le pourcentage de prédiction véridique trouvée par le classifieur.

Dans cette partie, on cherche à optimiser l'hyper-paramètre k pour minimiser l'erreur sur les données test. Pour trouver le k optimal, on va simplement tester le modèle pour k allant de 1 à 15.
K=1 représente le maximum de score, ainsi on va prendre cette valeur pour impute le nutrigrade.




error = 1 - knn.score(xtest, ytest)

# kNN Classification

Afin d'évaluez cet  algorithme de classification, on a recours au classification report.
À ce niveau, on va lire ce rapport d'une façon très superficielle et dire que la précision (precision) et le rappel (recall) sont proches de 90% dans chaque classe alors on peut donner confiance à ce modèle avec :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.91 | 0.92 | 6539 |
| 1 | 0.84 | 0.86 | 0.85 | 6360 |
| 2 | 0.87 | 0.85 | 0.86 | 8009 |
| 3 | 0.89 | 0.91 | 0.90 | 10690 |
| 4 | 0.93 | 0.92 | 0.92 | 7269 |
| accuracy | | | 0.89 | 38867 |
| macro avg | 0.89 | 0.89 | 0.89 | 38867 |
| weighted avg | 0.89 | 0.89 | 0.89 | 38867 |

# Sommaire

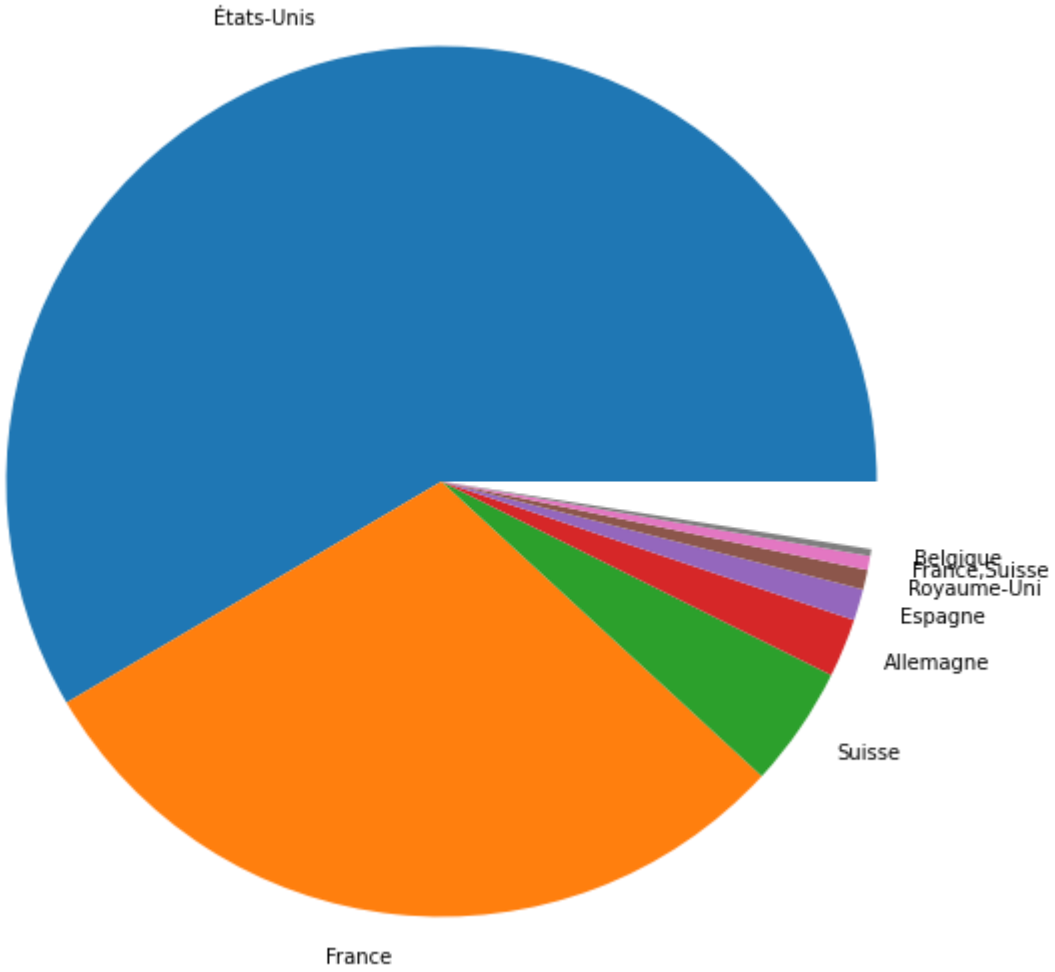
## Analyse exploration des données (EDA)

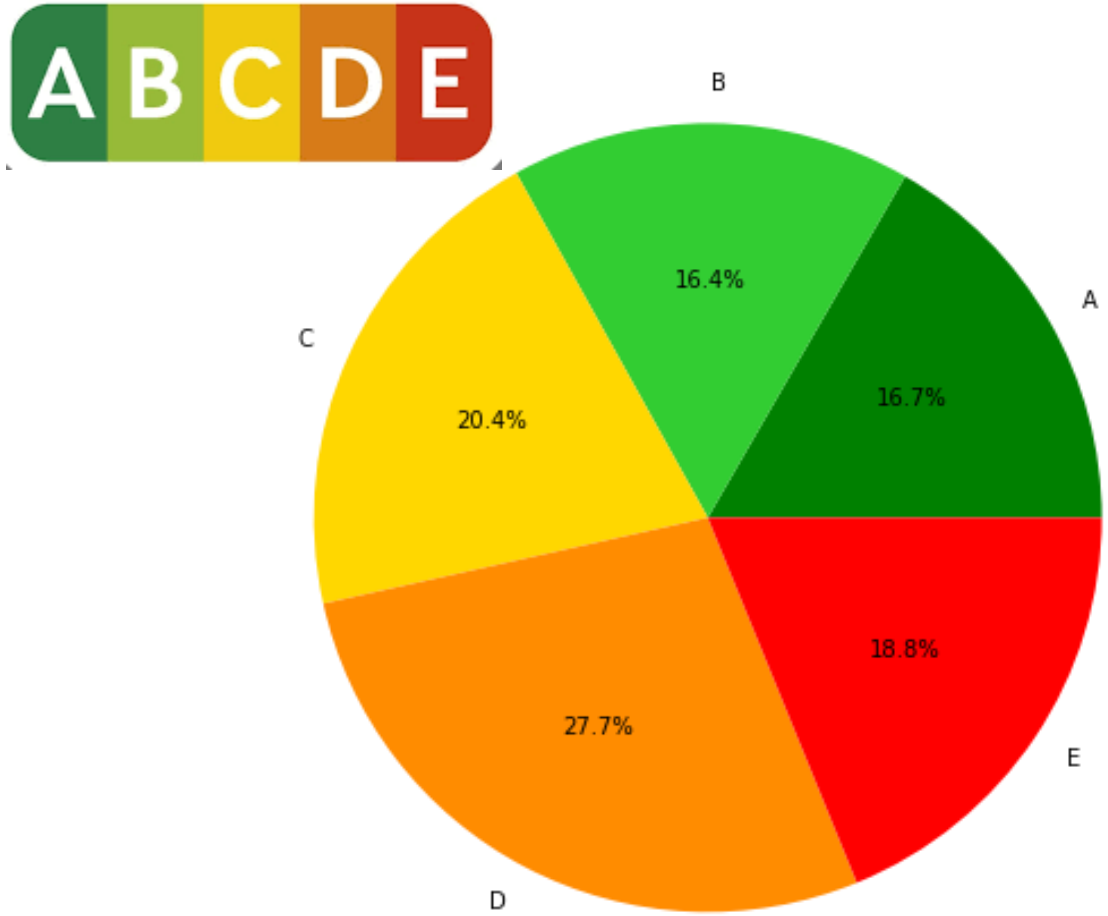
Après avoir nettoyer notre jeu de données , il est temps de l'explorer. Dans cette partie, on va faire :

- Une Analyse univariée
- Une Analyse multivariée
- Une ANOVA
- Une Analyse des Composantes Principales (ACP)

# Analyse univariée

Les Etats Unis et la France représentent plus de variété dans les produits alimentaires



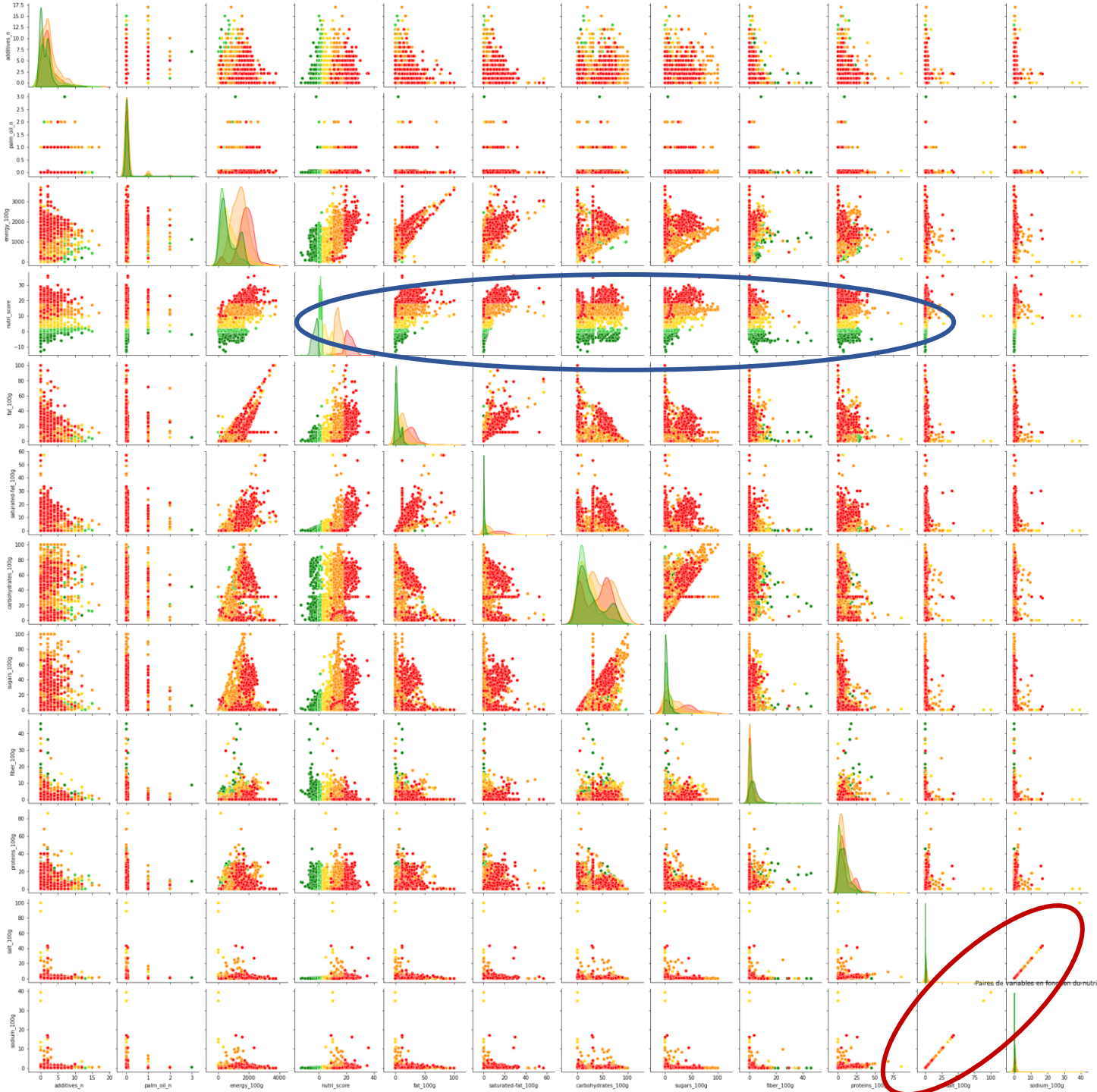




La catégorie **D** des nutri_grade est majoritairement représentée par 27.7% suivi par la catégorie C avec 20,4%.
Les aliments classés **A** qui sont les plus favorables sur le plan nutritionnel représentent un taux faible.

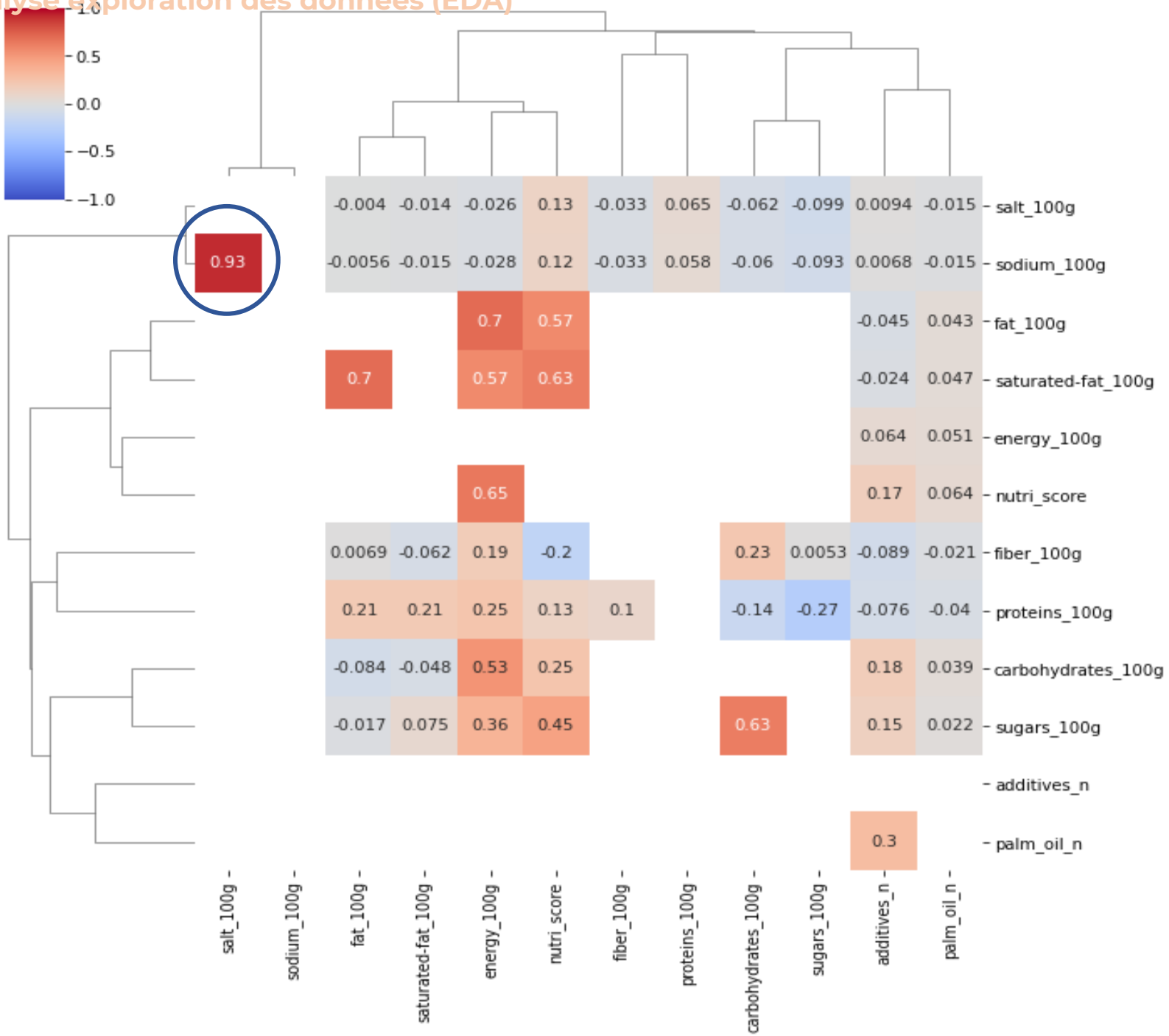
# Analyse multivariée

L'analyse des variables deux à deux montre que :

- La séparation par nutri_grade est remarquables entre les variables nutritionnelles et le score.
- Il y a des relations linéaires entre sel et sodium. Ces relations peuvent par exemples être vérifiées par un heatmap.
- À priori, sur la diagonale, on remarque que les groupes des nutrigardes sont séparer, on va vérifier ça ultérieurement par une ANOVA.

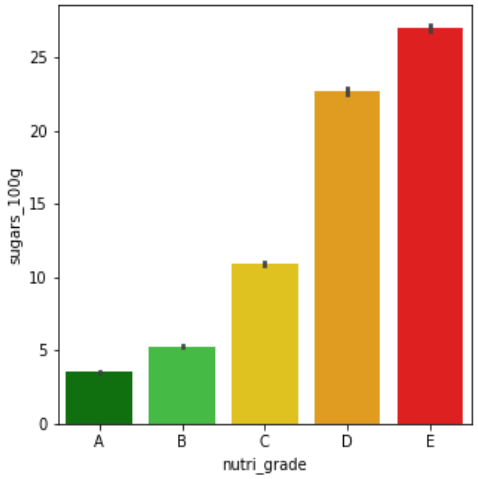


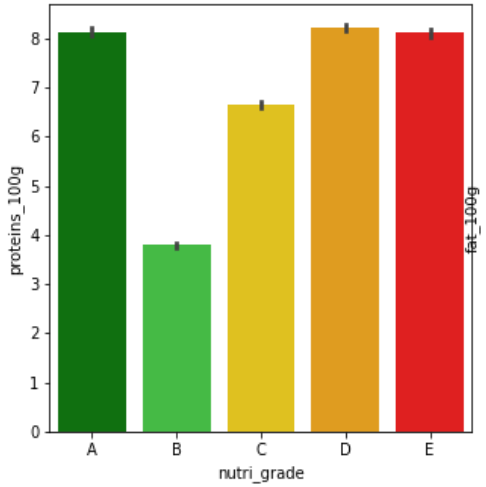

# Analyse multivariée

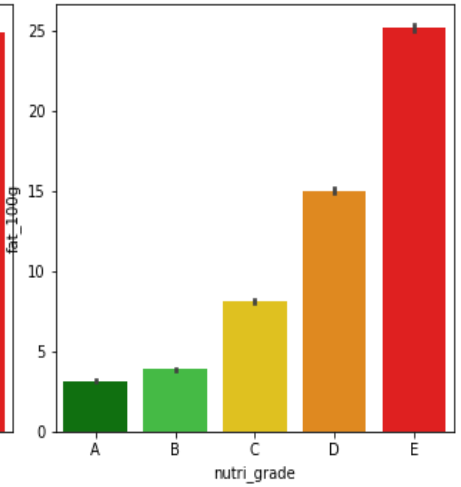


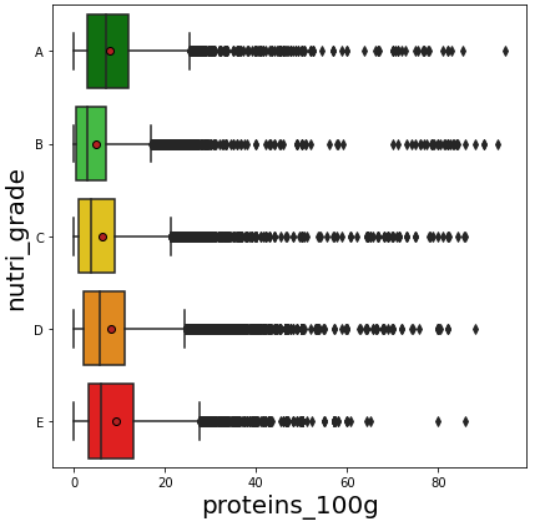

Nous avons vu dans le pairplot des relations linéaires entre certaines variables. On peut vérifier ça par un *clusterheatmap*. Par exemple entre le sel et le sodium le coefficient de Pearson est presque égal à 1

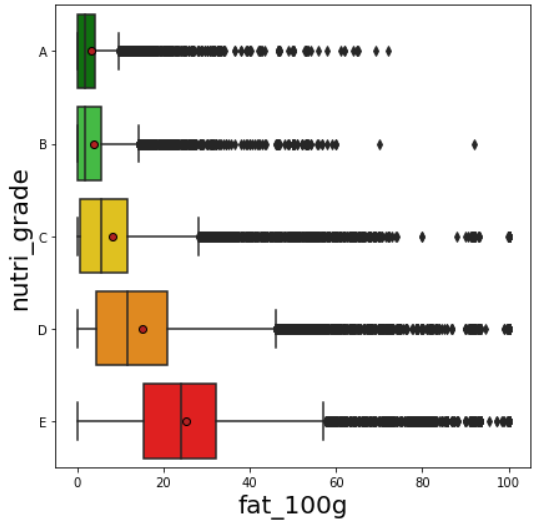
# Analyse multivariée

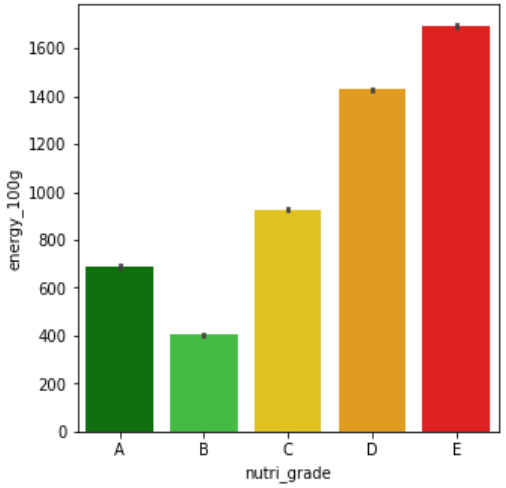


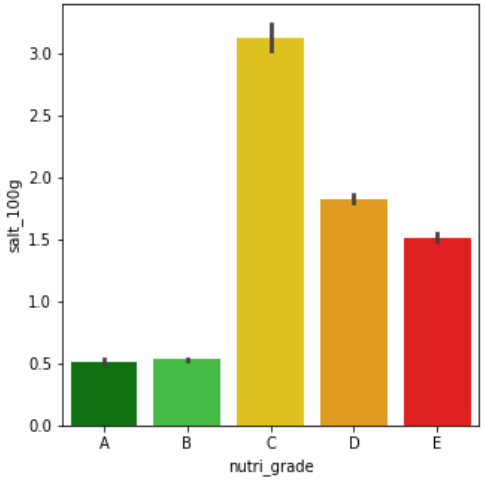



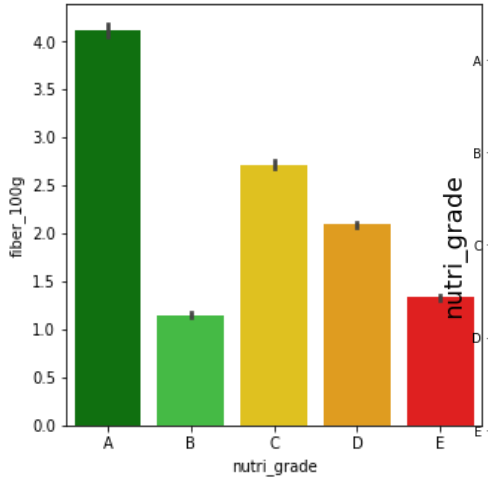



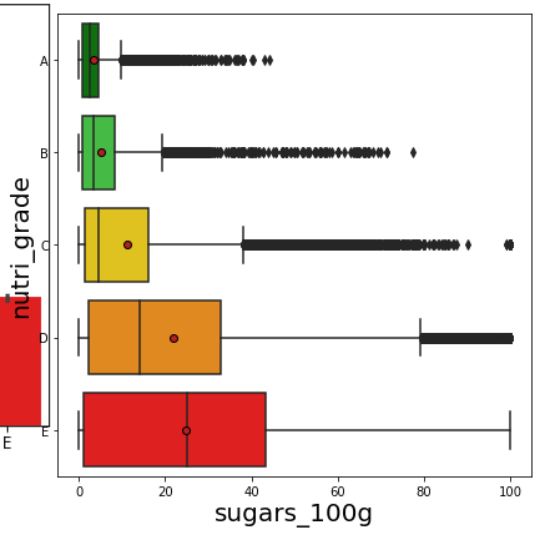



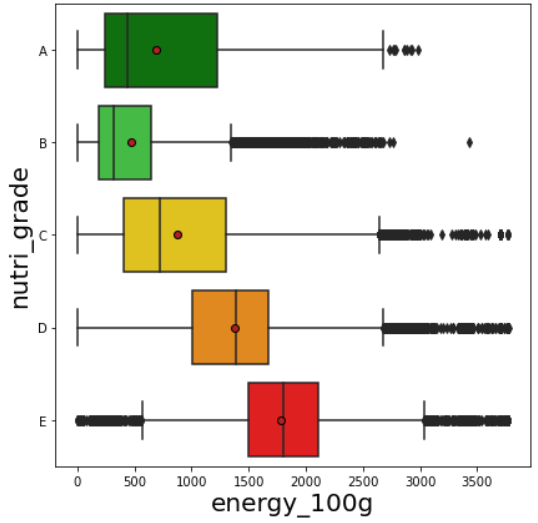




L'analyse des *barplots* et des *boxplots* des différentes valeurs nutritionnelles en fonction du grade nutritionnel montre une grande variation. Par exemple, la catégorie E est la plus riche en sucres et lipides et donc fournit l'énergie la plus élevée en opposition avec le groupe A qui est riche protéines et fibres.
À première vue On peut dire que les valeurs nutritionnelle sont très différentes d'une catégorie à une autre. Il reste à prouver.

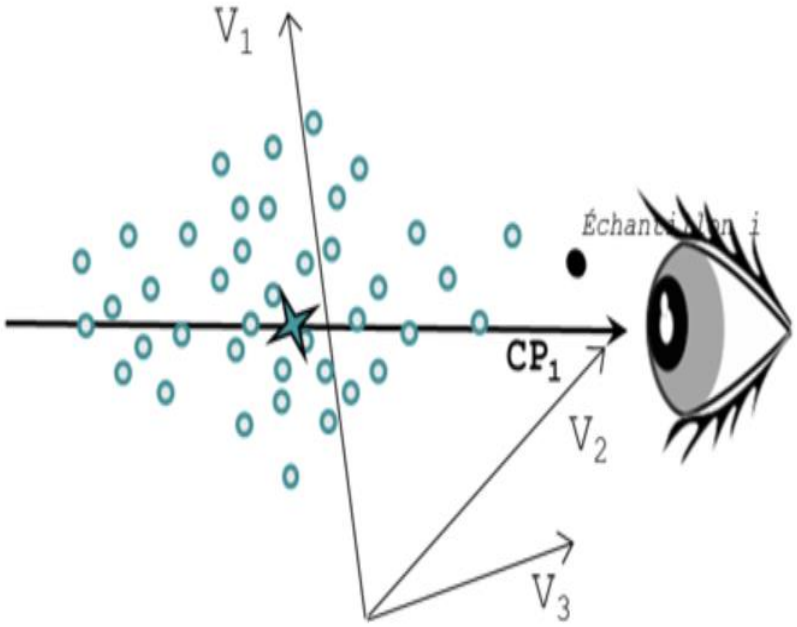
# Analyse en Composantes principals (ACP)

L'ACP est une méthode d'analyse des données multivariées bien connue de réduction de dimension qui va permettre de transformer des variables très corrélées en nouvelles variables décorrélées les unes des autres.

Le principe est simple : Il s'agit en fait de résumer l'information qui est contenue dans une large base de données en un certain nombre de variables synthétiques appelées : Composantes principales.

Les étapes ACP :

- Chercher les plans factoriels pour visualizer le maximum d'informations
- Tracer les cercles de correlation sur les différents plans factoriels
- Projection sur les plans factoriels



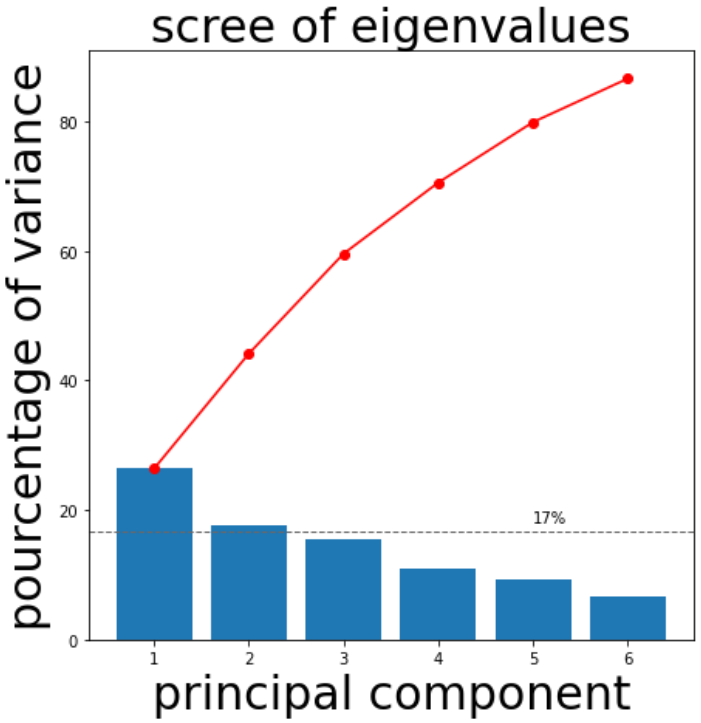
## ACP

## Inertie cumulée sur les axes des plans factoriels

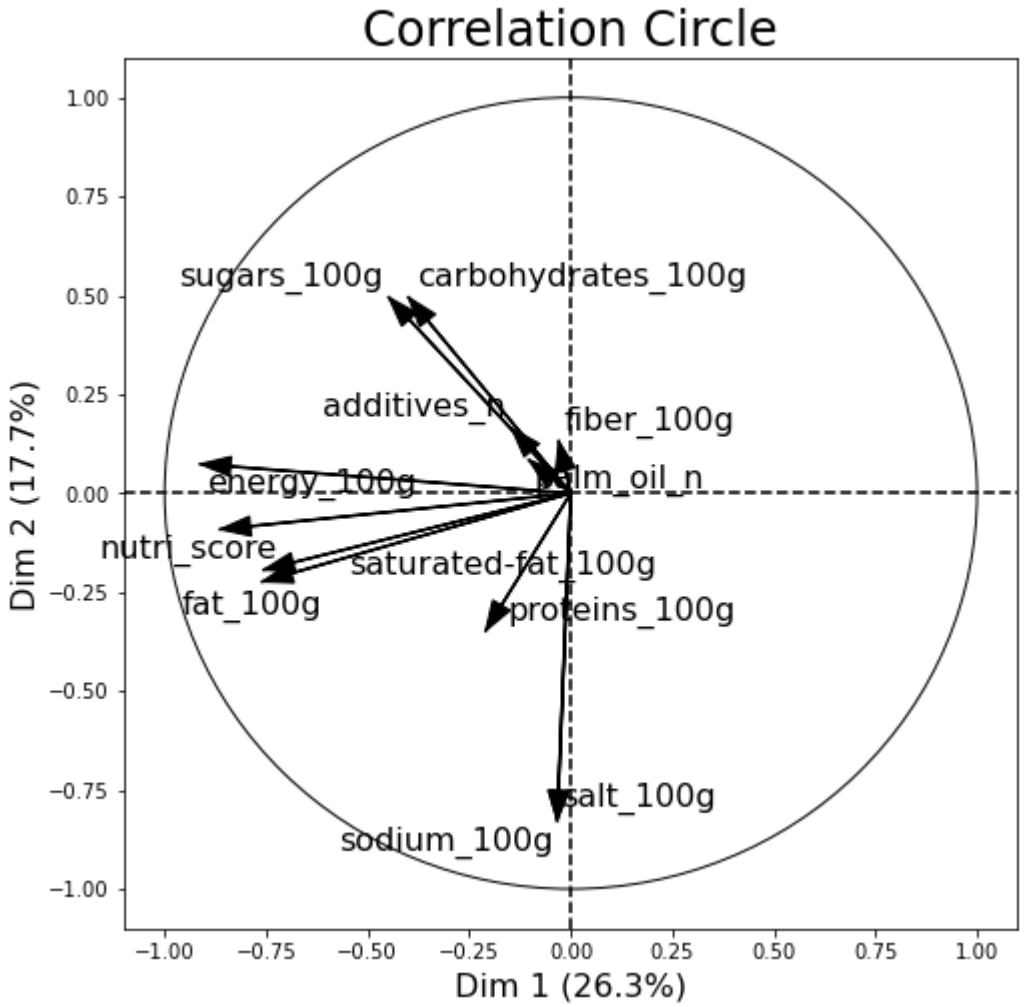
- Afin de déterminer le nombre de composantes nécessaires à l'analyse, nous projetons les données sur les axes principaux d'inertie qui sont ordonnés selon l'inertie du nuage projeté de la plus grande à la plus petite : c'est l'éboulis des valeurs propres.

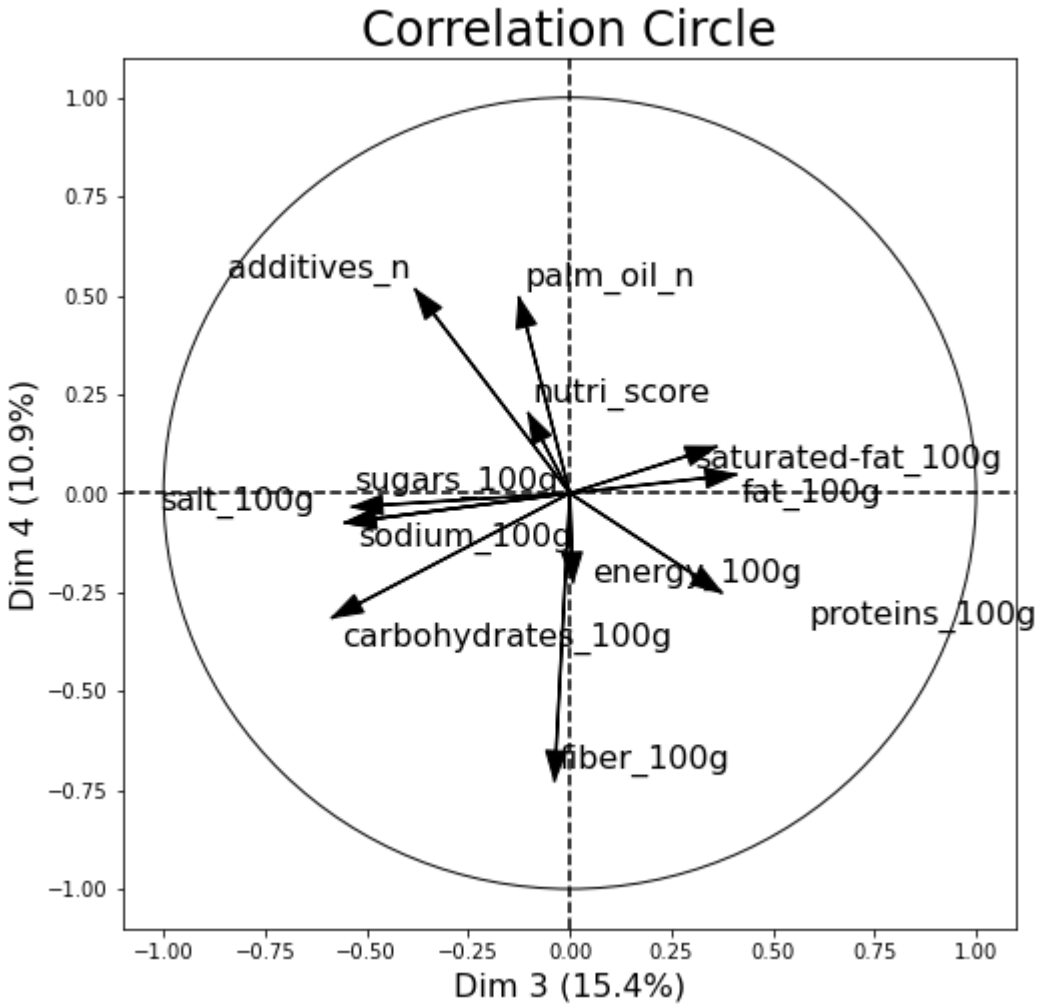
- On constate que le premier plan factoriel couvre une inertie de 44%.

- On remarque que les 4 premières composante c'est-à-dire le premier et le deuxième plan factoriel couvre 71% d'informations



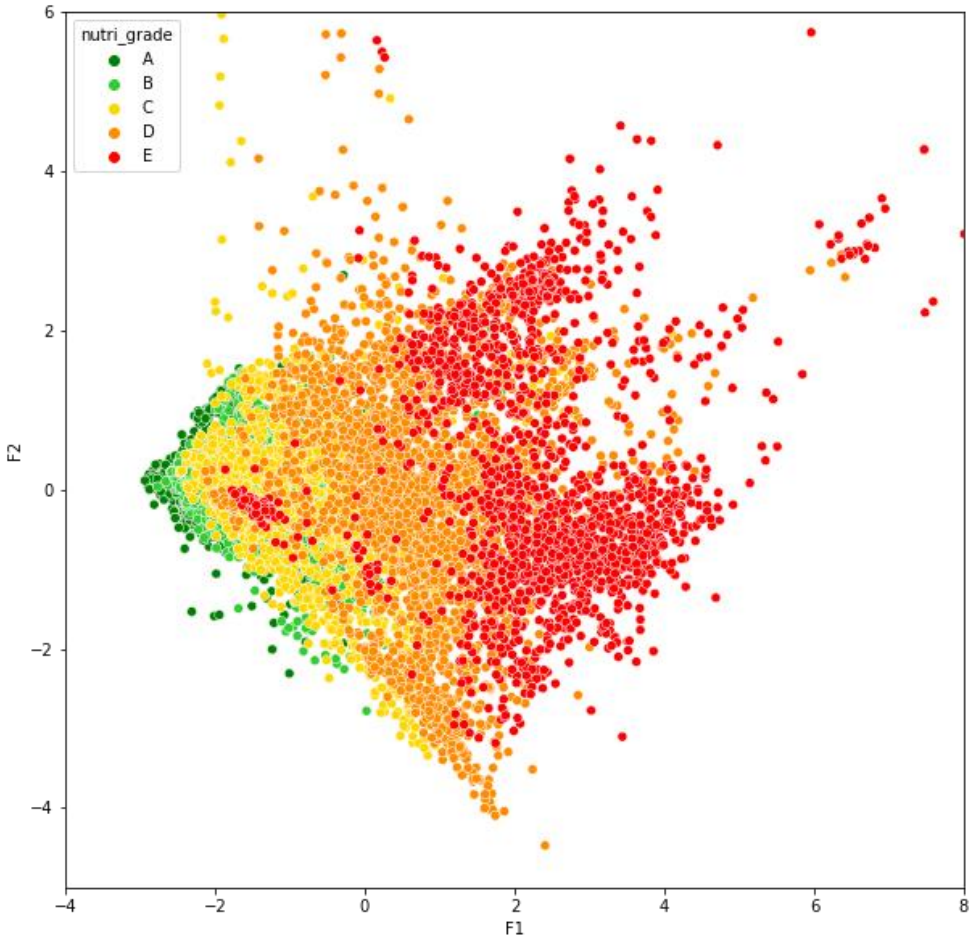

| | Dimension | Variance expliquée | % variance expliquée | % cum. var. expliquée |
|---|---|---|---|---|
| 0 | Dim1 | 3.16617865 | 26.0 | 26.0 |
| 1 | Dim2 | 2.12923662 | 18.0 | 44.0 |
| 2 | Dim3 | 1.85098951 | 15.0 | 60.0 |
| 3 | Dim4 | 1.31804804 | 11.0 | 71.0 |
| 4 | Dim5 | 1.11949243 | 9.0 | 80.0 |
| 5 | Dim6 | 0.80632025 | 7.0 | 87.0 |

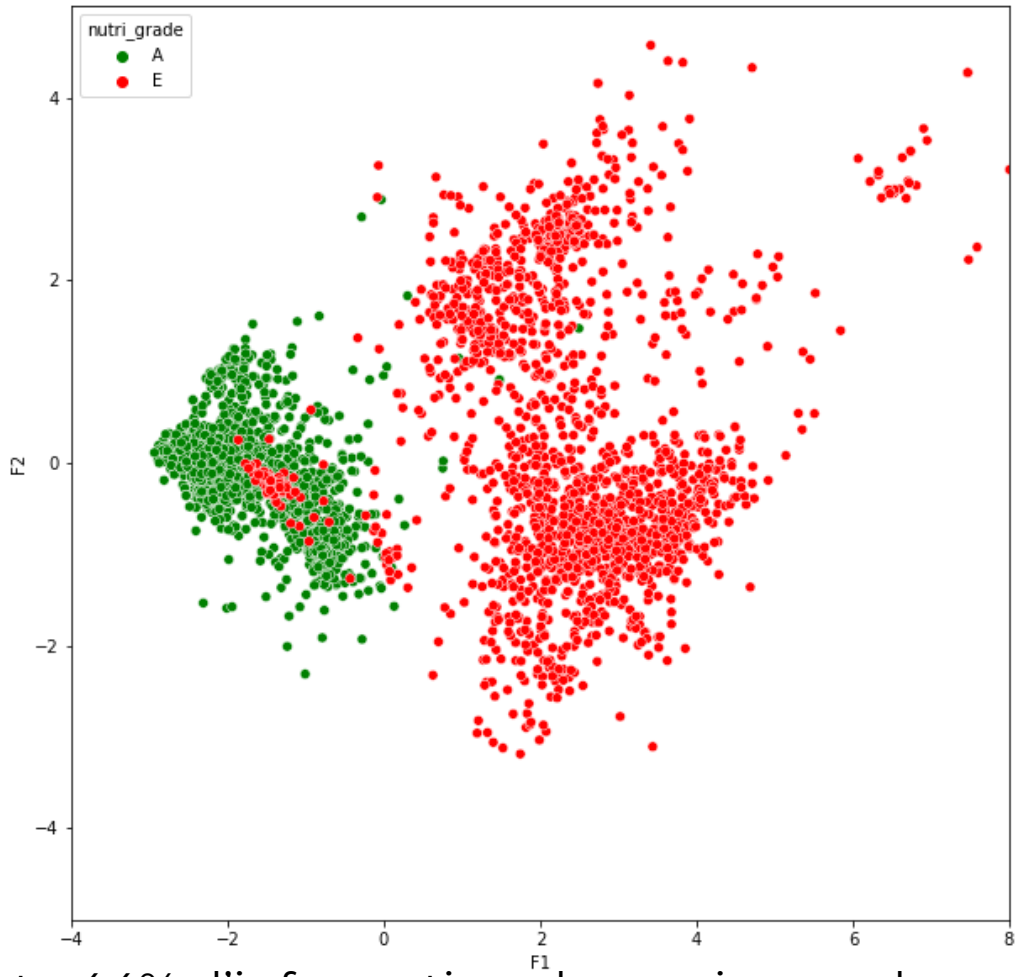



- L'axe F1 est fortement corrélé avec le caractère énergétique et le nutriscore (corrélés négativement).
- Le caractère salé est fortement corrélé avec F2.
- La variable fibre est anticorrélée avec F4.
- Fat et saturated fat sont fortement corrélés puisque l'angle entre les deux fléches tend vers 0 et ces deux dernières ont la même pointe.
- Sugars et proteins ne sont pas corrélés car l'angle est droit entre les flèches.

# Projection sur les plans factoriels




On voit bien sur le premier plan factoriel qui représente 44% d'informations le maximum de points projetés que les classes sont séparés en comparaison avec le pairplot où on a des graphes avec les catégories très mélangées

# ANOVA (ANalysis Of VAriance)

Méthode d'analyse qui permet d'étudier la dépendance d'une variable quantitative à une variable qualitative.
Dans notre cas la variable quantitative est F1 issue de l'ACP et la variable qualitative est le nutrigrade.

On va tester si la moyenne de la variable quantitative est elle homogène sur l'ensemble des modalités de la variable qualitative.
Par l'analyse de la variance, on va savoir si on va rejeter l'hypothèse nulle  Ho qui stipule qu'il n'y a pas de différence entre les moyennes des 5 catégories.
Le test de l'hypothèse réalisé est le test de Fisher. En utilisant le package pingouin de python, on a obtenu les résultats suivants :

| | Source | SS | DF | MS | F | p-unc | np2 |
|---|---|---|---|---|---|---|---|
| 0 | nutri_grade | 36860.52666156 | 4 | 9215.13166539 | 9416.50554545 | 0.0 | 0.70899383 |
| 1 | Within | 15129.38476586 | 15460 | 0.97861480 | NaN | NaN | NaN |

```
f_statistic result : (9416.505545446207, 0.0)
LeveneResult(statistic=219.46052111605024, pvalue=1.6361206449380882e-183)
 F1 eta squared is : 0.7089938345637928
```

Les résultats du test de Fisher nous indique une valeur de p value de 0 qui est inférieure au seuil de 5% et une valeur de F >> 1. Donc l'hypothèse nulle est rejetée et on retient l'hypothèse alternative qui dit que la variable F1 ( fortement corrélée avec l'énergie et les lipides ) dépend du grade nutritionnel.

# Sommaire

# Faisabilité du projet

On peut utiliser le dataset OPEN FOOD FACTS pour développer l'application **Custom Diet** pour certains types de régimes comme le régime cétogène ou Dukan qui sont basés essentiellement sur les quantités de protéines, lipides, sucres, fibres et l'énergie.

Pour d'autres types de régimes comme **Végétarien** ou **Végétalien** qui sont basés sur les quantités de vitamines et de sels minéraux , nous avons beaucoup de données manquantes. Ainsi, la base de données OPEN FOOD FACTS est insuffisante.

Pour avoir plus de données, il est nécessaire d'avoir d'autres dataset pour compléter les informations manquantes comme celles de FAO (Food and Agriculture Organisation ) l'Organisation des Nations unies pour l'alimentation et l'agriculture.

# Merci pour votre attention