Projet 3:
Conception d’application au service de la Santé Publique

Data Scientist
Xuefei ZHANG
07/2022

Sommaire

- Contexte et objectif
- Idée d’application
- Traitement et nettoyage de données
- Exploration de données: analyse univariée - bivariée - multivariée
- Faisabilité de la mise en production

Contexte

La Santé Publique a lancé un appel à projets pour trouver des idées innovantes d'applications en lien avec l'alimentation. On y participe et propose une idée d'application.

Objectif

À partir du jeu de données Open Food Facts et les informations de l'appel à projets:

1. Réfléchir une idée d'application et choisir les variables pertinentes conformément à l'application.
2. Nettoyage et traitement de données, et automatisation de processus de traitement
3. Faire une analyse multivariée pour les hypothèses et effectuer les tests statistiques associés
4. Justifier la faisabilité de l'application à partir des données Open Food Facts.
5. Pitch de l'idée

2. Comment peut-on valoriser les données ?

- Idée d’application de données Open Food Facts

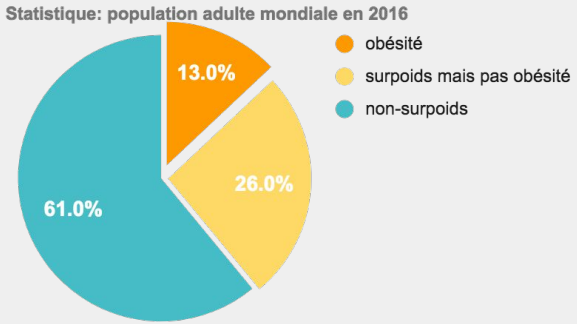
Réflexion

CE QU'ON A :

1. Jeu de données openfoodfacts.org contient des informations couvrant:
   - Les informations générales sur la fiche du produit: nom, date
   - Un ensemble de tags : catégorie, localisation, origine…
   - Les ingrédients et leurs additifs.
   - Informations nutritionnelles par 100g
2. Informations sur appels à projets (Santé Publique France)
3. Exemples d'application

SITUATION ACTUELLE :

Manger sain et léger prend le vogue avec l'augmentation du **taux d'obésité** provenant entre autres du **régime alimentaire**.



Raisons :

- une augmentation de la consommation d'**aliments très caloriques** riches en lipides
- une augmentation du manque d'activité physique

Quels indicateurs nous-seront utiles ?

IDÉE D'APPLICATION:

Dans l'hypothèse que les gens ont besoin de manger léger afin de se sortir de l'état surpoids ou prévenir l'éventualité de l'être, on a besoin d'un indicateur qui montre la légèreté des produits alimentaires.

On envisage une application capable d' évaluer le **niveau de légèreté** des produits alimentaires - **lightscore.**

En référant la littérature alimentaire, c'est clair que l'alimentation légère a certainement de caractéristiques ci-après:

- Bas en calorie
- Bas en grasse saturée
- Bas en sucre

En conséquence, on pré-sélectionne à ce titre les indicateurs ci-après:

- energy_100g,
- saturated-fat_100g
- sugars_100g
- fiber_100g (non-mandatoire)
- vitamins (non-mandatoire)
- nutrition-score-fr (non-mandatoire)

3. Traitement de jeux de données

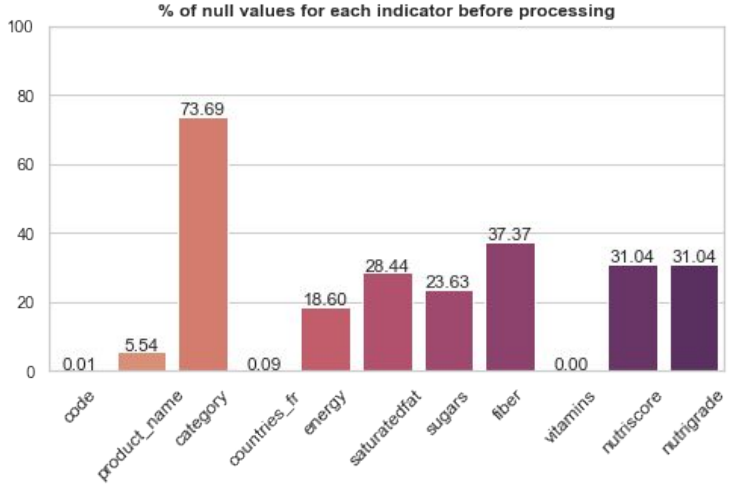
- Choix de variable-indicateurs pertinents
- Nettoyage et transformation
- Visualisation

1) Pré-sélection de variables et constitution de dataframe

Le choix des indicateurs doit prendre en compte à la fois :

A. A quel niveau l'indicateur est associé avec notre cible **(pertinence)**

B. Intégrité statistique des indicateurs **(% de null)**




```
RangeIndex: 320772 entries, 0 to 320771
Data columns (total 9 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   code          320749 non-null  object
 1   product_name  303010 non-null  object
 2   categories_fr 84411 non-null   object
 3   energy        261113 non-null  float64
 4   saturatedfat  229554 non-null  float64
 5   sugars        244971 non-null  float64
 6   fiber         200886 non-null  float64
 7   vitamins      320772 non-null  float64
 8   nutriscore    221210 non-null  float64
```

code	0.000072
product_name	0.055373
categories_fr	0.736850
energy	0.185986
saturatedfat	0.284370
sugars	0.236308
fiber	0.373742
vitamins	0.000000
nutriscore	0.310382

Possible clé primaire

2) nettoyage et transformation

A. Conserver seulement les observations qui ont lieu de vente ‘France’

```
df= df[df['countries_fr'].str.contains("France", na=False)]
```

```
0  code          98440 non-null  object
1  product_name  91247 non-null  object
2  category      61955 non-null  object
3  countries_fr  98440 non-null  object
4  energy        64593 non-null  float64
5  saturatedfat  62375 non-null  float64
6  sugars        62515 non-null  float64
7  fiber         45723 non-null  float64
8  vitamins      98440 non-null  float64
9  nutriscore    61415 non-null  float64
10 nutrigrade    61415 non-null  object
```

Suite à cette opération, on a les produits vendus en France (non-exclusive).

2) nettoyage et transformation

B. Vérification de null et doublons, enlèvement de null et dédoublonnage

```
df=df.dropna(subset=['code'])
df=df.drop_duplicates(subset=['code'])
```

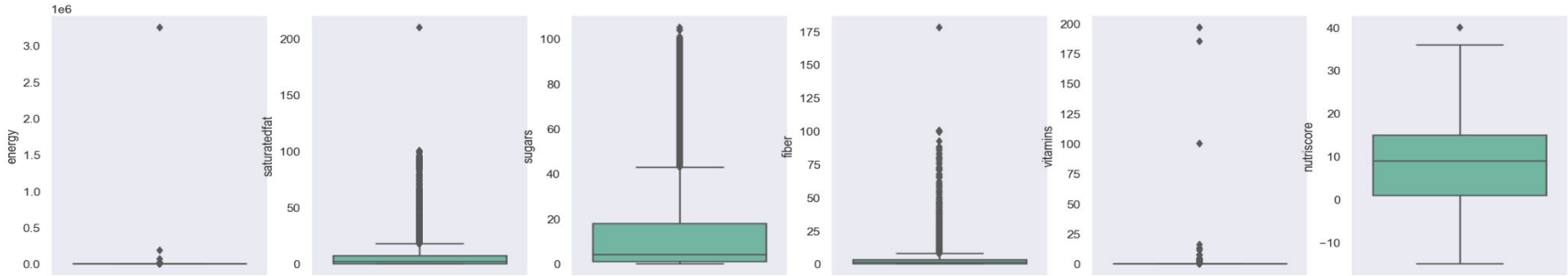
```
0  code          98436 non-null  object
1  product_name  91244 non-null  object
2  category      61952 non-null  object
3  countries_fr  98436 non-null  object
4  energy        64590 non-null  float64
5  saturatedfat  62372 non-null  float64
6  sugars        62512 non-null  float64
7  fiber         45720 non-null  float64
8  vitamins      98436 non-null  float64
9  nutriscore    61412 non-null  float64
10 nutrigrade    61412 non-null  object
```

Suite à cette opération, on obtient [code] unique, qualifié pour être clé primaire

Description de data

	energy	saturatedfat	sugars	fiber	vitamins	nutriscore
count	6.459000e+04	62372.000000	62512.000000	45720.000000	98436.000000	61412.000000
mean	1.171532e+03	5.423763	13.432775	2.559240	0.006676	8.683124
std	1.283620e+04	8.531269	19.087774	4.634929	0.923874	9.046211
min	0.000000e+00	0.000000	-0.100000	0.000000	0.000000	-15.000000
25%	4.270000e+02	0.300000	1.000000	0.000000	0.000000	1.000000
50%	1.035000e+03	2.000000	4.100000	1.365000	0.000000	9.000000
75%	1.649000e+03	7.400000	17.800000	3.200000	0.000000	15.000000
max	3.251373e+06	210.000000	105.000000	178.000000	196.707879	40.000000

- Présence de grand nombre de valeurs aberrantes dans **energy, saturatedfat, fiber, vitamins**
- Valeurs négatives dans **sugars** et **nutriscore.** Mais c'est normal pour nutriscore d'avoir valeur négative.



C. conversion de valeurs négatives en positives dans ‘sugars’
D. enlèvement de valeurs aberrantes dans ‘energy, saturatedfat, fiber, vitamins’

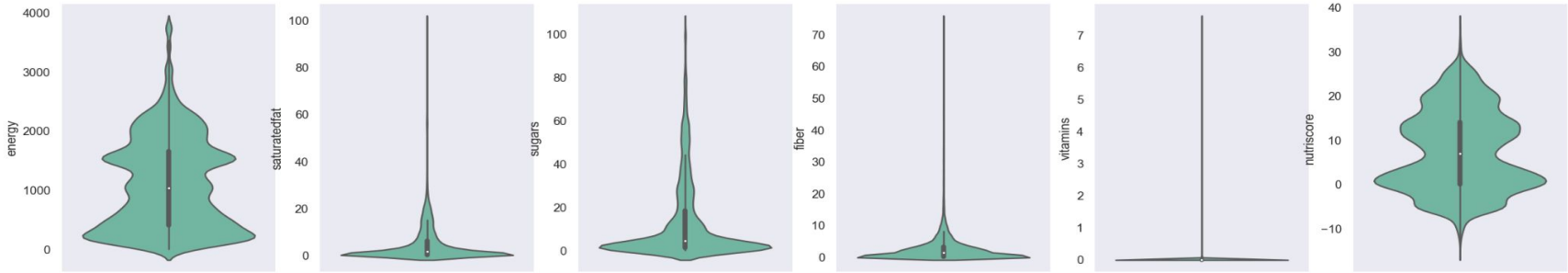
```
df.sugars = df.sugars.abs()
df = df.loc[(df.energy<=3770) & (df.saturatedfat<=100) &(df.fiber<=75) & (df.vitamins<=10)]
```

count	45286.000000	45286.000000	45198.000000	45286.000000	45286.000000	44677.000000
mean	1112.197626	4.918371	13.460465	2.505576	0.001928	7.588379
std	774.447680	8.211978	18.631677	4.131543	0.068608	9.036731
min	0.000000	0.000000	0.000000	0.000000	0.000000	-15.000000
25%	406.250000	0.300000	1.200000	0.000000	0.000000	0.000000
50%	1040.500000	1.600000	4.400000	1.330000	0.000000	7.000000
75%	1661.000000	6.200000	18.400000	3.200000	0.000000	14.000000
max	3770.000000	100.000000	104.000000	75.000000	7.596194	36.000000

Écart-type diminué

Min sugars positive

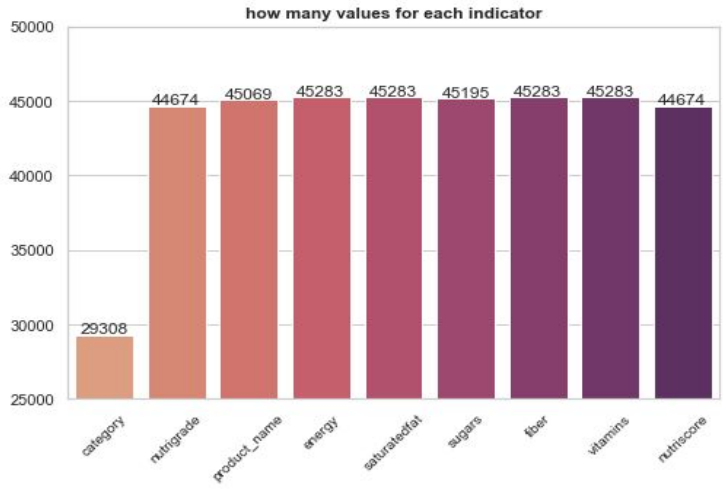
Max diminué



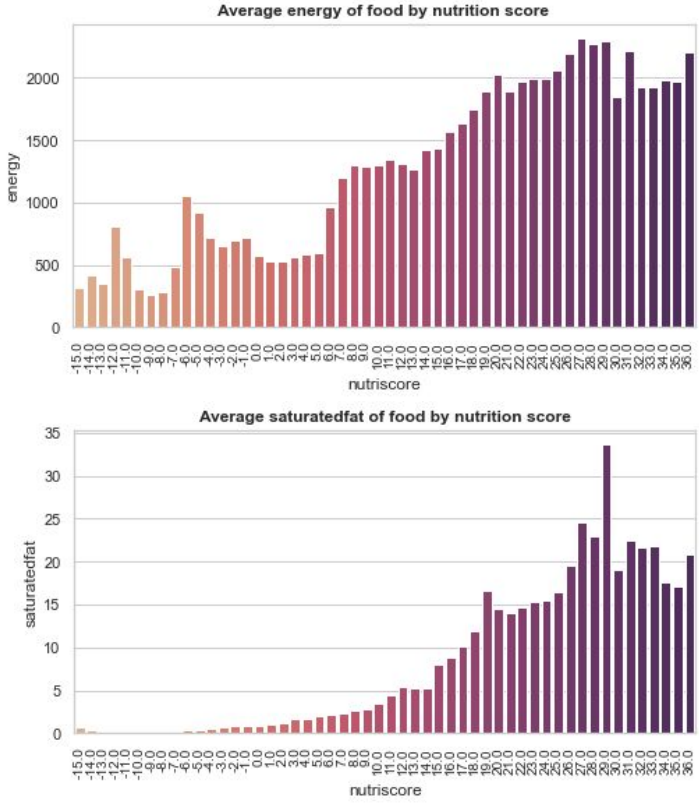
4. Exploration de données

- **Analyse univariée**
- **Analyse bivariée et multivariée:**
  - Corrélation
  - ANOVA - comment se varient le saturatedfat des produits de différents nutrigrade?
  - Régression linéaire - nutriscore peut être prédit par d'autres indicateurs?
  - Imputation de null
  - ACP
- **Scoring**

Analyse univariée - barplot



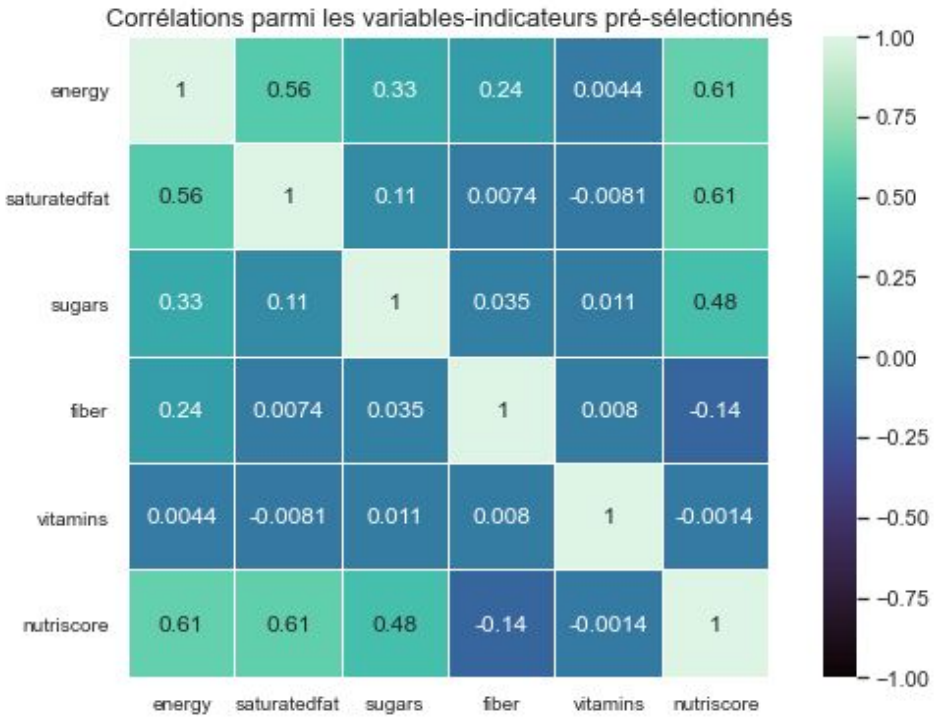
À part ‘category’, il n’existe pas de grand écart sur la quantité de données de chaque indicateur.



En générale, l’aliment est plus calorique et riche en saturatedfat avec l’augmentation de **nutricore**

Analyse bivariée

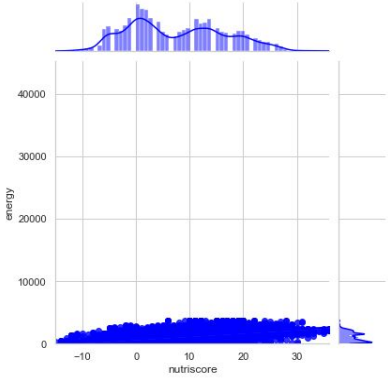
- corrélation: heatmap

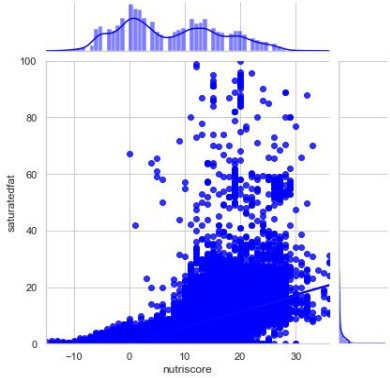


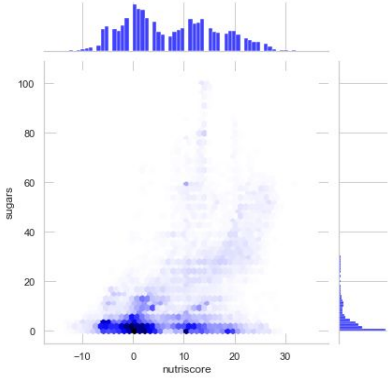
- Heatmap: nutriscore se corrèle fortement et positivement avec energy, saturatedfat et sugars
- Heatmap: il existe de très faible corrélation entre nutriscore et fiber & vitamins.

Analyse bivariée

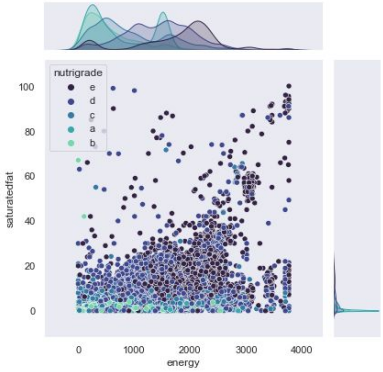
- distribution: jointplot (histogram+scatterplot)

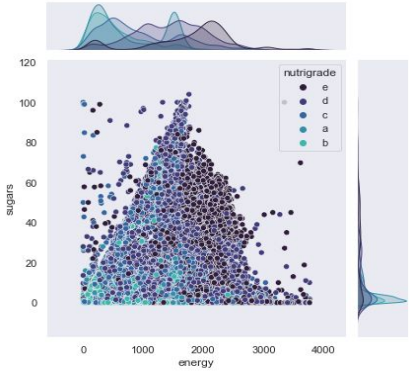






- Grande possibilité de faire régression linéaire **nutriscore-energy.**





- Energy n’a pas possibilité de faire régression linéaire avec ni saturatedfat ni sugars
- La distribution de ces 3 indicateurs varient en fonction de **nutrigrade**.

ANOVA - analyse de variance des moyennes de groupes

```
Anova_grade1= ols('saturatedfat~nutrigrade', data=df2).fit()
Anova_grade1.summary()
```

]:

OLS Regression Results

Dep. Variable:	saturatedfat	**R-squared:**	0.349
Model:	OLS	**Adj. R-squared:**	0.349
Method:	Least Squares	**F-statistic:**	5978.
Date:	Sat, 23 Jul 2022	**Prob (F-statistic):**	0.00
Time:	13:17:04	**Log-Likelihood:**	-1.4799e+05
No. Observations:	44674	**AIC:**	2.960e+05
Df Residuals:	44669	**BIC:**	2.960e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5922	0.069	8.577	0.000	0.457	0.728
nutrigrade[T.b]	0.4917	0.104	4.719	0.000	0.287	0.696
nutrigrade[T.c]	1.7668	0.096	18.309	0.000	1.578	1.956
nutrigrade[T.d]	6.4345	0.094	68.658	0.000	6.251	6.618
nutrigrade[T.e]	13.7546	0.103	133.074	0.000	13.552	13.957

Variable catégorielle: nutrigrade
Donc on peut essayer de prendre nutrigrade pour variable explicative Anova

H0: égalité de moyenne des différents groupes (sous-population) pour une variable dépendante (saturatedfat).

- nutrigrade peut expliquer la variance de saturatedfat à **34.9 %**
- la variance de saturatedfat expliquée par les autres facteurs aléatoire: **65.1 %**

P-values <0.5%, rejet de H0, donc il exsite au moins une moyenne qui est très différente des moyennes d'autres groupes.
=> variance parmi les groupes sur saturatedfat peut être mesurée en fonction de différences en nutrisgrade.

Régression linéaire multiple & tests statistiques associés

reg_multi4 = smf.ols('**nutriscore~energy + saturatedfat + sugars + fiber**', data=df2).fit()

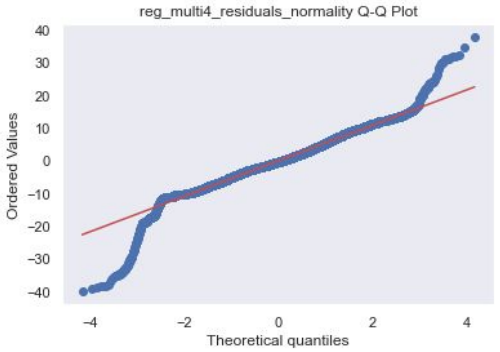
Dep. Variable:	nutriscore	**R-squared:**	0.633
Model:	OLS	**Adj. R-squared:**	0.633
Method:	Least Squares	**F-statistic:**	1.923e+04
Date:	Sat, 23 Jul 2022	**Prob (F-statistic):**	0.00
Time:	13:56:40	**Log-Likelihood:**	-1.3936e+05
No. Observations:	44674	**AIC:**	2.787e+05
Df Residuals:	44669	**BIC:**	2.788e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1681	0.047	3.599	0.000	0.077	0.260
energy	0.0040	4.44e-05	89.546	0.000	0.004	0.004
saturatedfat	0.4199	0.004	109.011	0.000	0.412	0.427
sugars	0.1614	0.001	108.995	0.000	0.159	0.164
fiber	-0.5063	0.007	-77.042	0.000	-0.519	-0.493

H0: variables explicatives n'ont pas d'impact sur variable dépendante.

P-values sont 0, => rejet de H0: tous ces 4 variables indépendantes ont de l'impact significatif sur nutriscore.

R-squared = 0.63,
ces 4 variables dans son ensemble arrivent à expliquer **63%** de variance dans nutriscore.



Normalité vérifiée

[('Lagrange multiplier statistic', 7854.536265
('p-value', 0.0),
('f-value', 2382.260935296986),
('f p-value', 0.0)]

Rejet de H0:
homoscétasticité vérifiée

imputation de null - identification de stratégie et 1ère imputation

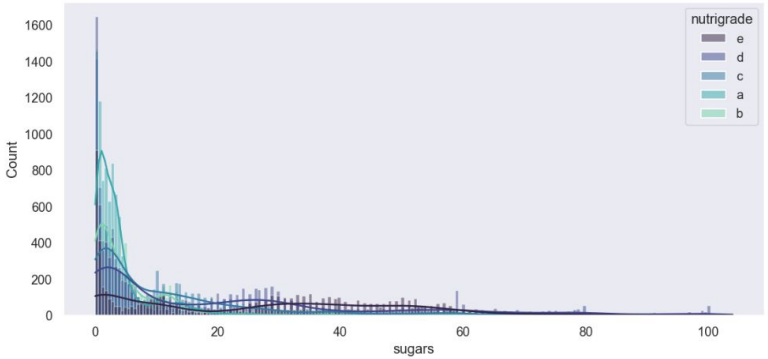
category	0.352781	category	29308
nutrigrade	0.013449	nutrigrade	44674
product_name	0.004726	energy	45283
energy	0.000000	saturatedfat	45283
saturatedfat	0.000000	sugars	45195
sugars	0.001943	fiber	45283
fiber	0.000000	vitamins	45283
vitamins	0.000000	nutriscore	44674
nutriscore	0.013449		

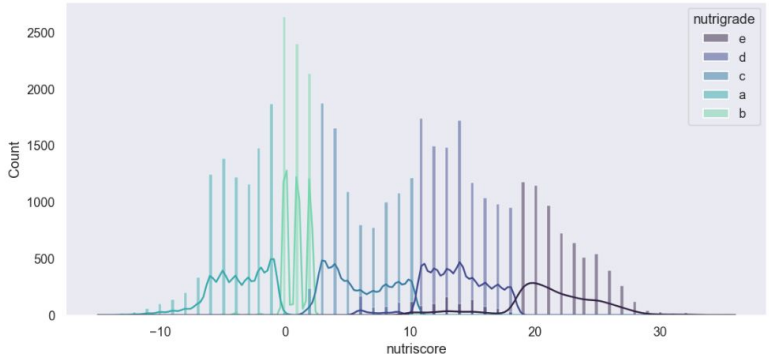
dtype: float64

Colonnes à imputer: **sugars, nutriscore**

Stratégie d'imputation

- imputer nutriscore selon **nutrigrade** où se trouvent les produits car nutrigrade se classe en fonction de nutriscore (Anova, histogramme) - par **moyenne ou médiane,** car il n'y a pas grand écart entre ces deux chiffres pour nutriscore, et qu'il n'y a pas bcp d'outliers
- imputer sugars selon **category** où se trouvent les produits - par **mediane** de chaque category: vu omniprésence des outliers et que la moyenne est non-robuste aux outliers





imputation de null - analyse et 2ème imputation

Suite à la 1ère imputation

category	29308
nutrigrade	44674
energy	45283
saturatedfat	45283
sugars	45261
fiber	45283
vitamins	45283
nutriscore	44674

Nombre de null énormément diminué

Nombre de null n'est pas changé

```
df_imputed.sugars = df_imputed.groupby(['category'],dropna=False)['sugars'].transform(lambda x: x.fillna(x.median()))
df_imputed.nutriscore = df_imputed.groupby(['nutrigrade'],dropna=False)['nutriscore'].transform(lambda x: x.fillna(x.mean()))
```

Cette opération n'a pas fonctionné sur nutriscore (44674 - 44674), et que nutrigrade a aussi 44674 valeurs, ctd. (45283 - 44674) individus n'ont ni de nutriscore ni nutrigrade. c'est ainsi que imputation groupby nutrigrade n'a pas marché.

Solution?
Groupby **category** pour nutriscore.

Suite à la 2ème imputation

category	29308
nutrigrade	44674
energy	45283
saturatedfat	45283
sugars	45261
fiber	45283
vitamins	45283
nutriscore	44899

Suite à l'imputation groupby category, le nombre de null est énormément diminué

Imputation de null - 3ème imputation par K-NN méthode

```
imputer = KNNImputer(n_neighbors=5).fit(X)
```

Imputation selon les 5 valeurs les plus proches

On a déjà groupby category et/ou nutriscore, donc c'est raisonnable d'opter pour K-NN méthode qui prend en compte les N valeurs plus proches vu la similarité des produits dans une même catégorie ou un même nutrigrade.

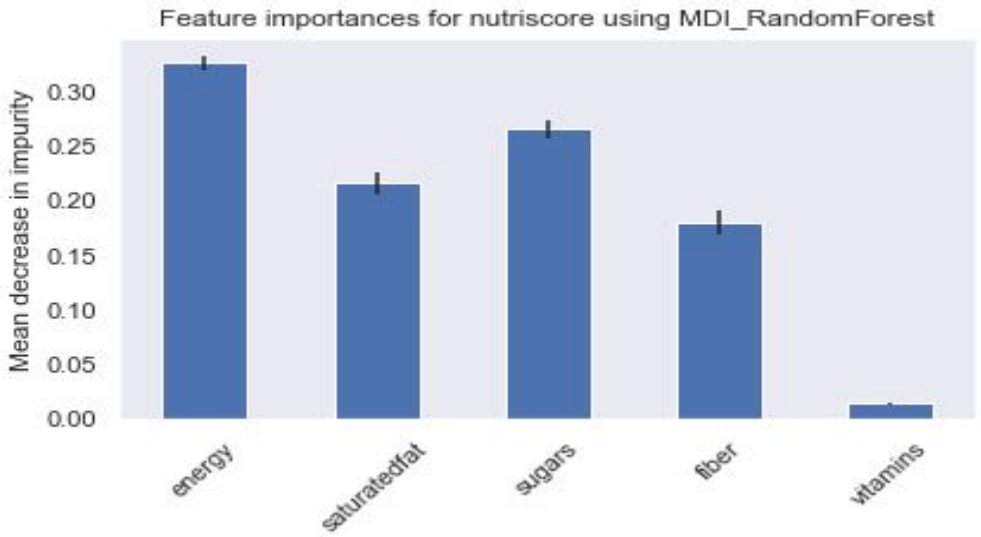
```
 0   energy        45283 non-null  float64
 1   saturatedfat  45283 non-null  float64
 2   sugars        45283 non-null  float64
 3   fiber         45283 non-null  float64
 4   vitamins      45283 non-null  float64
 5   nutriscore    45283 non-null  float64
```

Nombre de null réduit à 0

Nombre de null réduit à 0

Référence - nutriscore: importances des indicateurs

Identifier l'importance de chaque indicateur-variable via **RandomForest**.



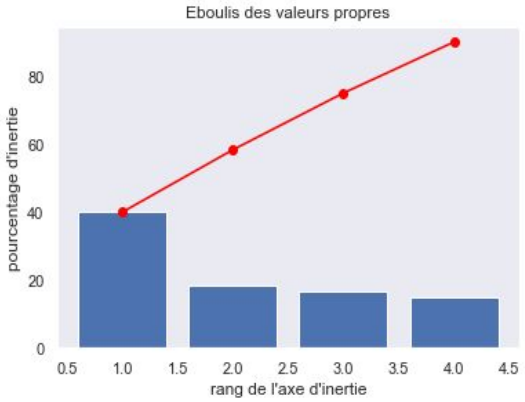
Conclusion:

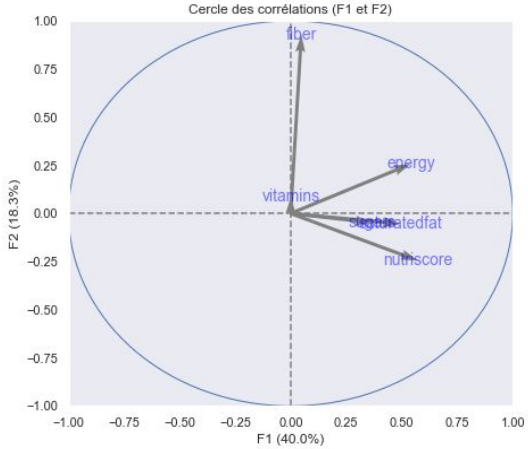
- Il exsite de grande différence en terme de l'impact (poids) de ces 5 variables pour nutriscore (model).
- il ne faut pas négliger facilement les 4 première variables, et vitamins n'est pas forcément à prendre en compte.
- Autrement dit, ces 4 variables sont bien pertinentes, alors que vitamins à voir.

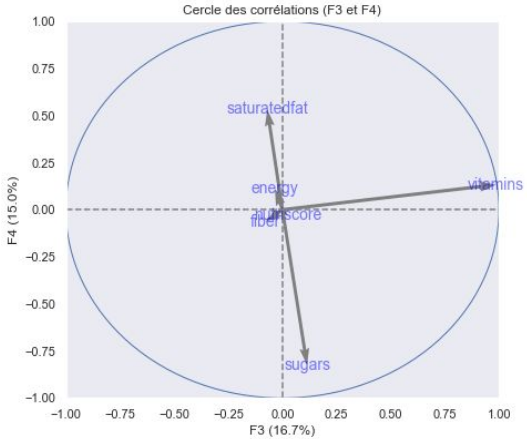
ACP sur dataframe imputé

ACP consiste à:

- identifier la pertinence de chaque variable-indicateur sur le model
- chercher la possibilité de réduire le nombre de variables en entrée (réduction de dimension)







- **On peut réduire jusqu'à 4 dimensions (environ 90% inertie - 10% d'info perdue)** ;
- Les inerties de chacune ces 4 dimensions ne se varient pas bcp;
- Toutes les 6 variable-indicateurs sont bien représentées par l'ensemble de ces 4 dimensions (90% inertie) => on peut réduire la dimensionnalité, mais on peut pas enlever aucune de ces variables;
- **=> ces variables pré-sélectionées sont bien pertinentes pour notre algorithme.**

Calculer lightscore -
en fonction de résultat RandomForest et à base de **df_imputed2**

- Std représente les importances proportionnelles des variable-indicateurs obtenues via RandomForest

:

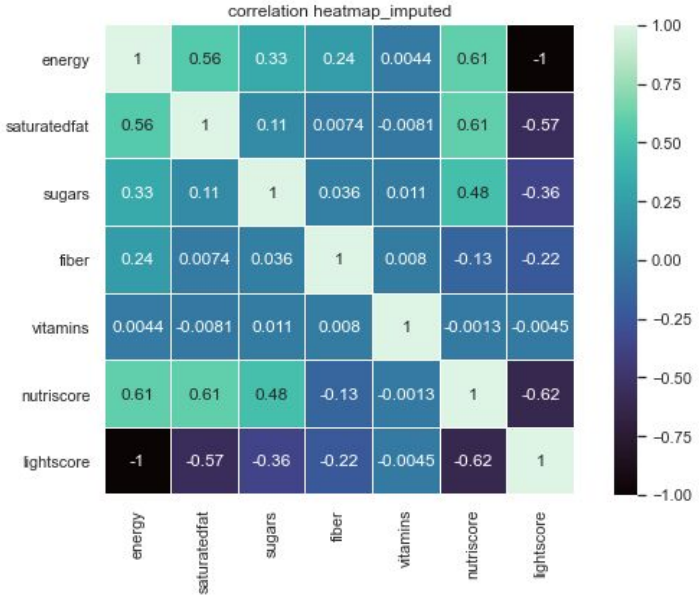
nutrigrade	energy	saturatedfat	sugars	fiber	vitamins	nutriscore	lightscore
e	3766.0	100.0	0.0	0.0	0.0	20.0	-25.07
e	3766.0	95.0	0.0	0.0	0.0	20.0	-25.02
d	3766.0	95.0	0.0	0.0	0.0	15.0	-25.02
e	3766.0	95.0	0.0	0.0	0.0	20.0	-25.02
e	3766.0	94.0	0.0	0.0	0.0	20.0	-25.01

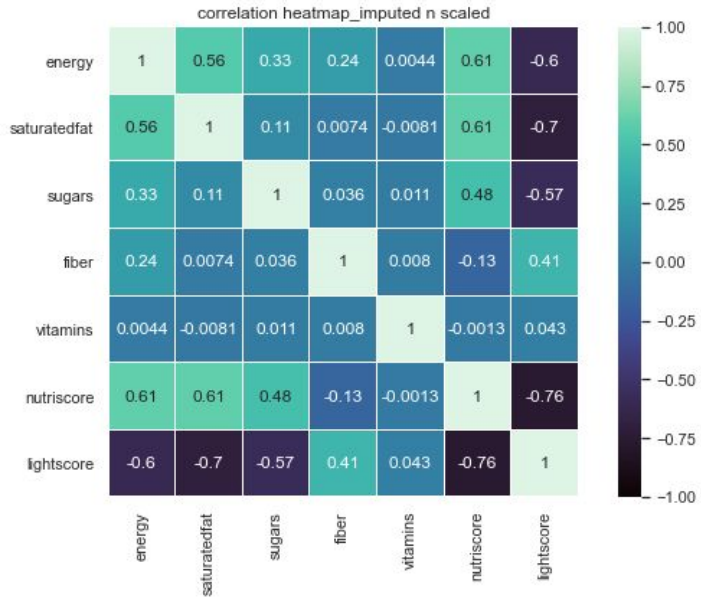
Calculer lightscore -
en fonction de résultat RandomForest et à base de **df_scaled**

- df_scaled = df_imputed2 centré et réduit via **StandardScaler** dans ACP
- **std[i]** représente les importances proportionnelles des variable-indicateurs obtenues via RandomForest

nutrigrade	energy	saturatedfat	sugars	fiber	vitamins	nutriscore	lightscore
e	3.426776	11.578176	-0.721928	-0.60643	-0.028103	1.379578	-0.14
e	0.600221	10.116924	3.037013	-0.60643	-0.028103	2.709964	-0.14
e	0.465930	9.873382	2.392624	-0.60643	-0.028103	1.823040	-0.13
d	3.419029	10.482237	-0.721928	-0.60643	-0.028103	0.825251	-0.13
e	3.426776	10.725779	-0.721928	-0.60643	-0.028103	1.379578	-0.13

Comparaison de 2 résultats lightscore





- Il semble que le résultat lightscore basé sur df_scaled est plus raisonnable que celui de df_imputed.
- Normalement, lightscore devrait être négativement corrélée avec les 3 premiers indicateurs et nutriscore, positivement avec fiber et vitamins.

5. Faisabilité de la mise en production

En synthétisant les conclusions ci-dessus:

CRITÈRES DE SÉLECTION D'INDICATEUR (de l'ordre séquentielle 1 à 4):

1. **Pertinence** d'indicateur
2. **Poids** d'indicateur dans lightscore (si pertinent)
3. Facilité de **récupération** de données (% de null)
4. Facilité de **traitement** de données

1 & 2 : En règle générale, sauf vitamins, les 5 indicateurs sont bien pertinents.

Et leur poids dans lightscore:

- Energy - 0.6
- Saturated-fat - 0.7
- Sugars - 0.57
- Fiber 0.41
- Vitamins 0.043
- Nutriscore - 0.76

3. Pour la facilité de récupération, on fait référence au % null dans le jeu de données original:

- Energy 18.6%
- Saturated-fat 28.4%
- Sugars 23.6%
- Fiber 37.4%
- Vitamins NaN ou 100%
- Nutriscore 31%

4. il vaut mieux qu'on ignore vitamins, vu que dans la pratique, de divers types de vitamins ne se constatent pas par la somme mais séparément. Ça pénalise le traitement.

=> Ainsi on prend le reste 5 indicateurs dans notre application.

C'est faisable d'effectuer notre application à travers les données OpenFoodFacts.

A vos questions !