Conception d’une application au service de la santé publique

Oumouni Mestapha

Projet 2 parcours IML, Openclassrooms

Plan

Présentation de l’application et le dataset

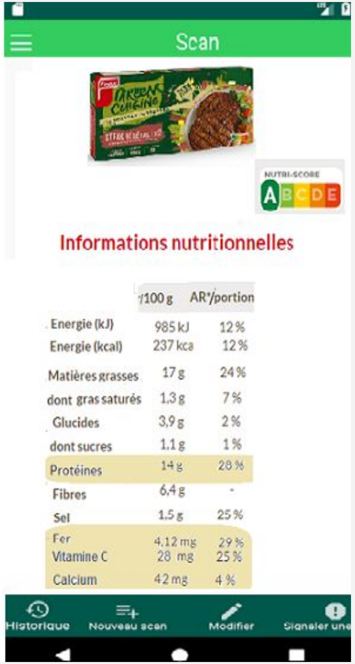
Nettoyage de jeu de données

Analyse univariée et bivariée

Analyse multi-variée

Application ***Nutri-vegan***

- Classement des produits vegan (végétarien et végétalien)
- Nutri-score du produit
- Liste des ingrédients
- Informations nutritionnelles (macro et micronutriments)
- Proposition des produits similairs



Données openFood

Les variables qualitatives

Les informations générales sur la fiche du produit : nom, date de création et de modification, les contributeurs les pays des produits et ses marques, ingredients, additives ...

Un ensemble de tags : catégorie du produit, localisation, origine...

Les variables quantitatives

Quantité en grammes d'un nutriment pour 100g du produit (macro et micro-nutriments)

Importation du jeu de données:

```
data_food =  pd.read_csv('fr.openfoodfacts.org.products.csv', sep='\t',low_memory=False)
```

```
data_food.shape
```

```
(320772, 162)
```

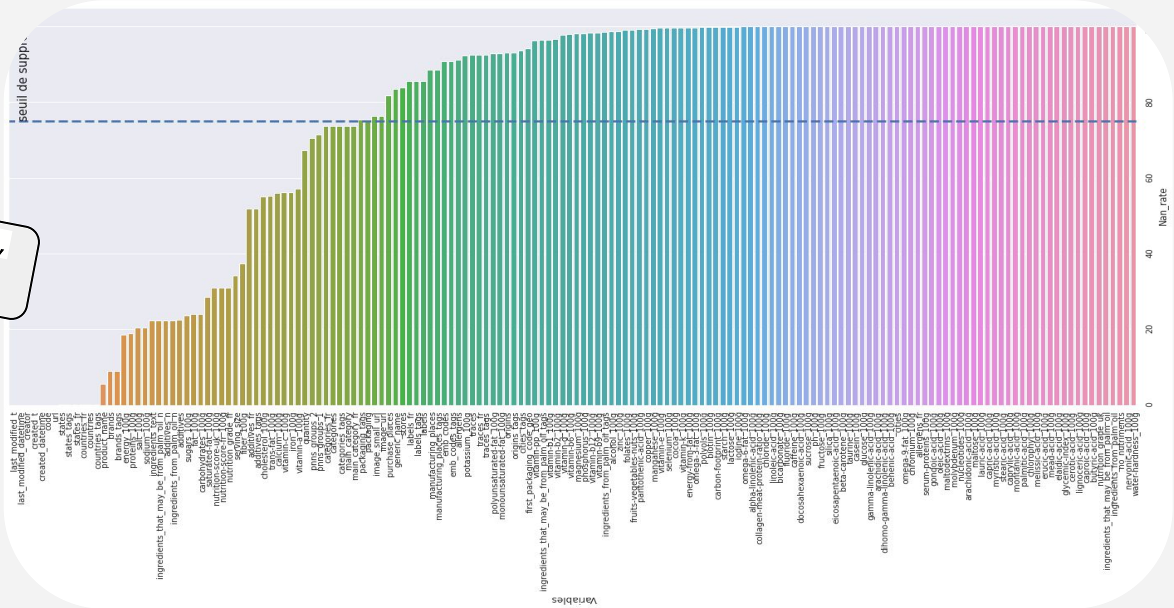
Le jeu de données compte 320772 lignes et 162 variables (colonnes)

```
data_food.dtypes.value_counts()
```

```
float64    106
object      56
```

Traitement de jeu de données

```
data_food.dropna(thresh=int(data_food.shape[0]/4),
axis=1)
```



data.shape : (320772, 50)
30 v. qualitatives et 20 v. quantitatives

```
data.duplicated().sum()
```

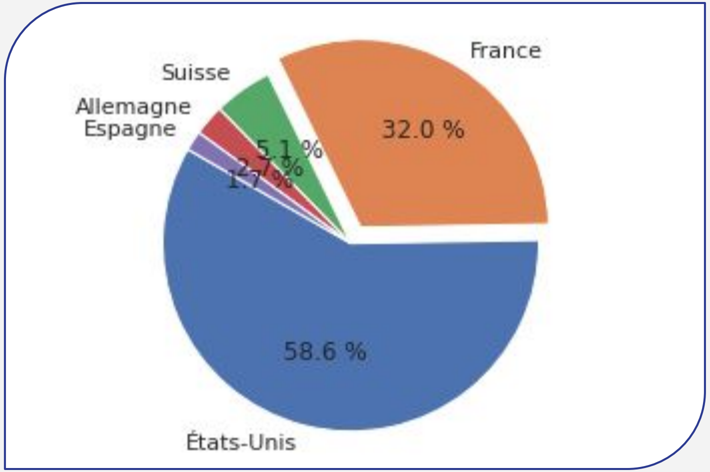
```
0
```

Variables qualitatives

```
countries_index=data['countries_fr'].value_counts().head(5).to_frame().index

data = data[data.countries_fr.isin(countries_index)]
```

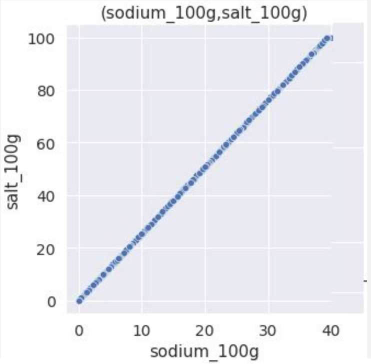
- code
- creator
- product_name
- brands
- categories_fr
- main_category_fr
- countries_fr
- ingredients_text
- additives_fr
- nutritions_grade_fr

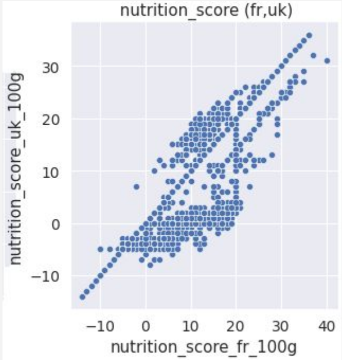


Variables quantitatives

forte corrélation
niveau d'importance
redondance

nutrition_score_fr_100g
energy_100g
fat_100g
saturated_fat_100g
cholesterol_100g
carbohydrates_100g
sugars_100g
fiber_100g
proteins_100g
salt_100g
vitamin_c_100g
calcium_100g
iron_100g





	energy_100g	fat_100g	saturated_fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g
count	2.510570e+05	234080.000000	221174.000000	233810.000000	236488.000000	195072.000000	250070.000000	246944.000000
mean	1.141427e+03	12.687791	5.102747	32.101850	16.005656	2.856944	7.080298	2.051322
std	6.572863e+03	17.569114	7.982191	29.782784	22.394071	13.033953	8.447826	130.468261
min	0.000000e+00	0.000000	0.000000	0.000000	-17.860000	-6.700000	-800.000000	0.000000
25%	3.770000e+02	0.000000	0.000000	6.000000	1.300000	0.000000	0.600000	0.063500
50%	1.100000e+03	5.000000	1.790000	20.750000	5.700000	1.500000	4.710000	0.589280
75%	1.674000e+03	20.000000	7.140000	58.330000	24.000000	3.600000	10.000000	1.391920
max	3.251373e+06	714.290000	550.000000	2916.670000	3520.000000	5380.000000	430.000000	64312.800000

Valeurs aberrantes

valeur négative	valeur absolue ou zéro
valeur maximale	100 (sucre et graisse) 3700 (énergie)

```
for c in col:
  upper_lim = data[c].quantile(0.96)
  ind = data.index[data[c] > upper_lim]
  data = data.drop(ind , axis=0)
```

	energy_100g	fat_100g	saturated_fat_100g	cholesterol_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g
count	233427.000000	217189.000000	204888.000000	129840.000000	216903.00000	220265.000000	179809.000000	232509.000000	229498.000000
mean	1100.123153	12.432798	4.914900	0.014447	31.93572	16.200523	2.352412	6.224411	1.616159
std	801.477035	17.422403	7.578051	0.026055	29.02987	21.373038	2.911432	6.600903	6.802467
min	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
25%	356.000000	0.000000	0.000000	0.000000	6.25000	1.500000	0.000000	0.500000	0.060000
50%	1046.000000	4.700000	1.790000	0.000000	20.00000	5.880000	1.400000	4.300000	0.561340
75%	1665.000000	20.000000	7.140000	0.018000	58.00000	24.240000	3.600000	9.250000	1.333500
max	3700.000000	100.000000	100.000000	0.107000	100.00000	100.000000	14.600000	28.000000	100.000000

Valeurs manquantes-KNN Imputation

KNN Imputer maintient la valeur, la variabilité de données avec plus de précision

```
# On divise les colonnes en deux ensembles
col = data.select_dtypes('float64').columns
col_macro = col.drop(['calcium_100g','energy_100g','cholesterol_100g','vitamin_c_100g', 'iron_100g'])
col_micro =['cholesterol_100g','vitamin_c_100g', 'calcium_100g', 'iron_100g']
```

```
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(data[col_macro], test_size=0.20)
model = KNNImputer(n_neighbors=5, missing_values=np.nan)
model.fit(X_train)
df_imp1 = model.transform(data[col_macro])
df_imp1 = pd.DataFrame(df_imp1, columns=col_macro)
```

Imputation de nutri-grade

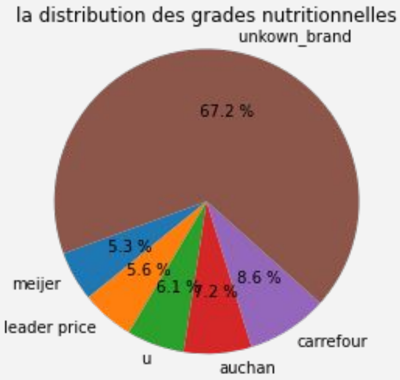
- Nan = string équivalent
- nutrition_grade_fr =nutrition_score_fr_100g

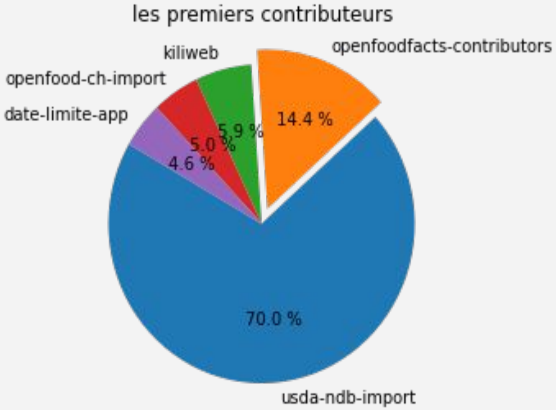
Aliments solides (points)	Boissons (points)	Nutriscore (lettre)
-15 à -1	Eau	A
0 à 2	≤ 1	B
3 à 10	2 à 5	C
11 à 18	6 à 9	D
19 à 40	10 à 40	E

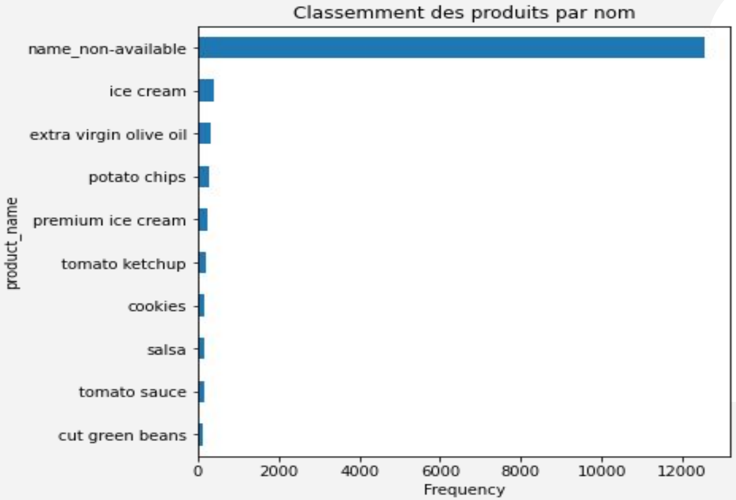
```
data.info()
```

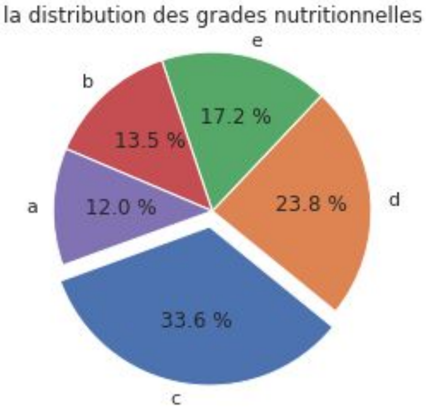
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 261852 entries, 0 to 320771
Data columns (total 23 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   code                     261852 non-null  object
 1   creator                  261852 non-null  object
 2   product_name             261852 non-null  object
 3   brands                   261852 non-null  object
 4   categories_fr            261852 non-null  object
 5   countries_fr             261852 non-null  object
 6   ingredients_text         261852 non-null  object
 7   additives_fr             261852 non-null  object
 8   nutrition_grade_fr       261852 non-null  object
 9   main_category_fr         261852 non-null  object
 10  energy_100g              261852 non-null  float64
 11  fat_100g                 261852 non-null  float64
 12  saturated_fat_100g       261852 non-null  float64
 13  cholesterol_100g         261852 non-null  float64
 14  carbohydrates_100g       261852 non-null  float64
 15  sugars_100g              261852 non-null  float64
 16  fiber_100g               261852 non-null  float64
 17  proteins_100g            261852 non-null  float64
 18  salt_100g                261852 non-null  float64
 19  vitamin_c_100g           261852 non-null  float64
 20  calcium_100g             261852 non-null  float64
 21  iron_100g                261852 non-null  float64
 22  nutrition_score_fr_100g  261852 non-null  float64
dtypes: float64(13), object(10)
memory usage: 47.9+ MB
```

Analyse exploratoire (v. qualitative)

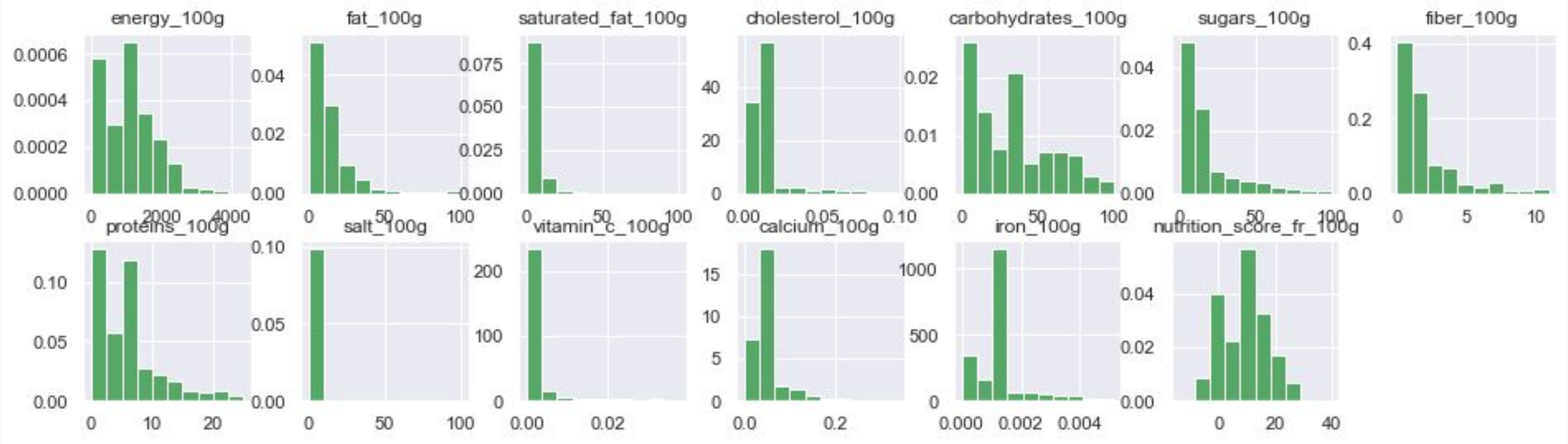




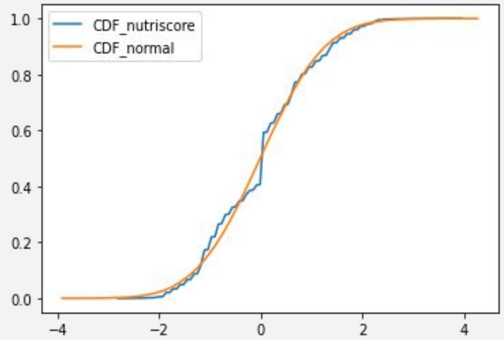




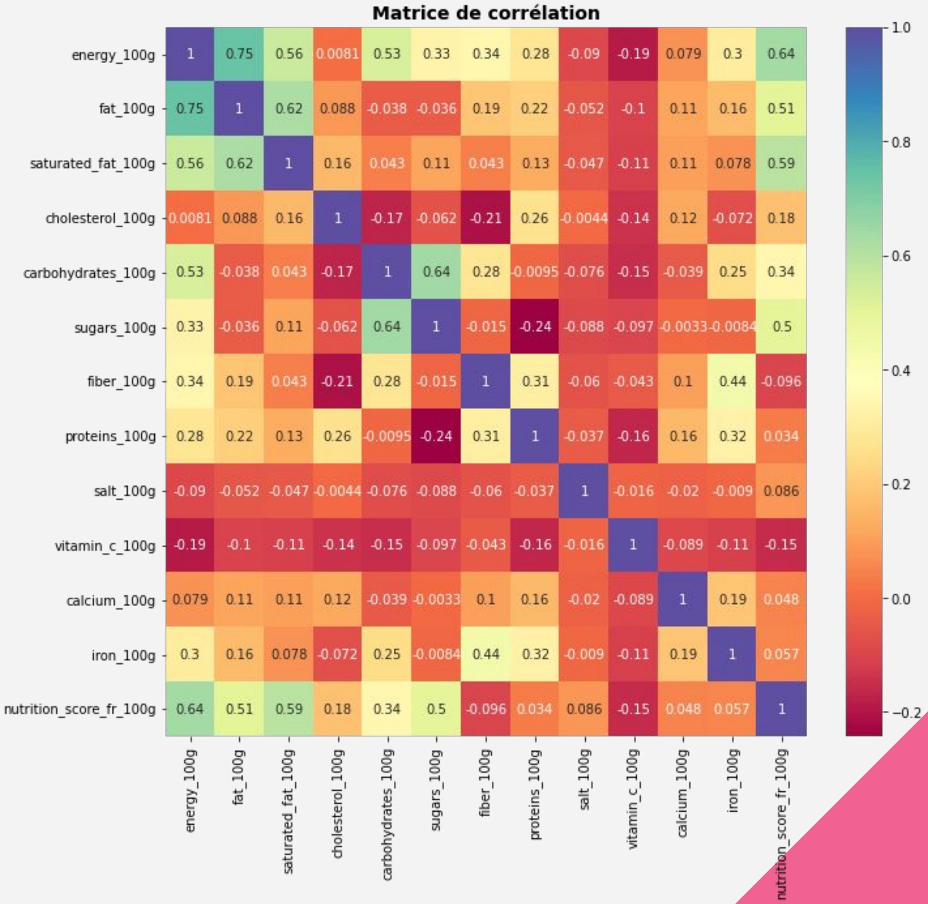
Analyse exploratoire (v. quantitative)



distribution non-gaussienne



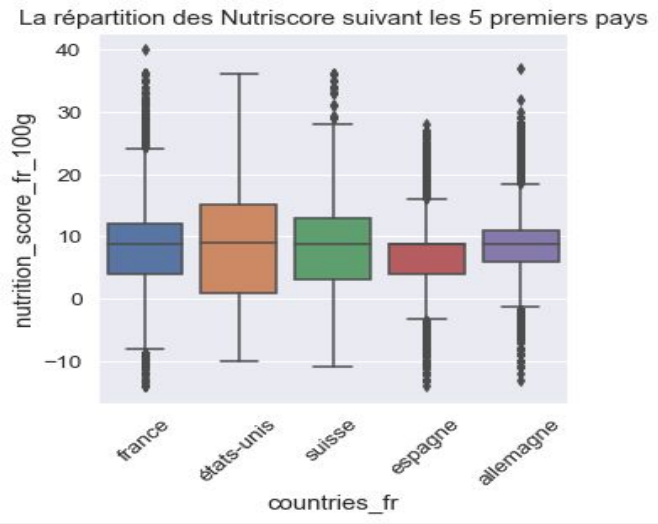
Matrice de corrélation



Corrélation significative entre les variables énergétiques et le nutri-score.

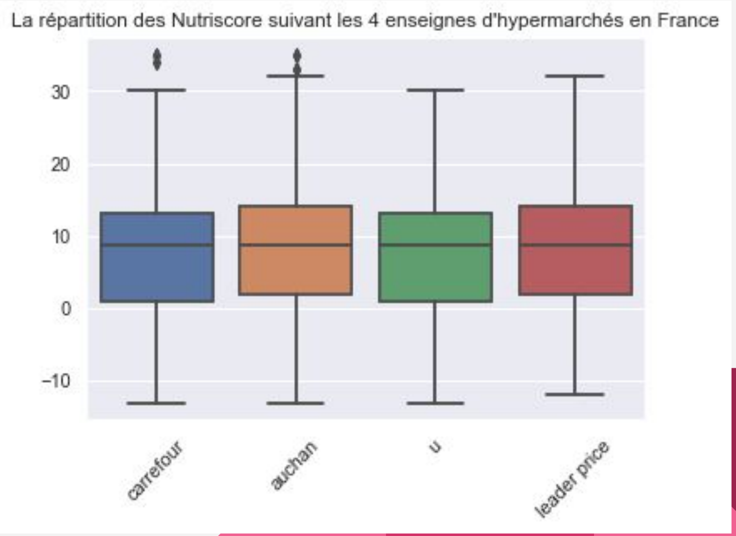
Absence d'une corrélation négative.

Analyse ANOVA




```python
# reshape  dataframe db suitable for ANOVA test
data_brand = pd.melt(db, id_vars=[], value_vars=top_brand)
data_brand.columns=['brands','nutrition_score_fr_100g']
anova_category = smf.ols('nutrition_score_fr_100g~brands', data=data_brand).fit()
sm.stats.anova_lm(anova_category)
```

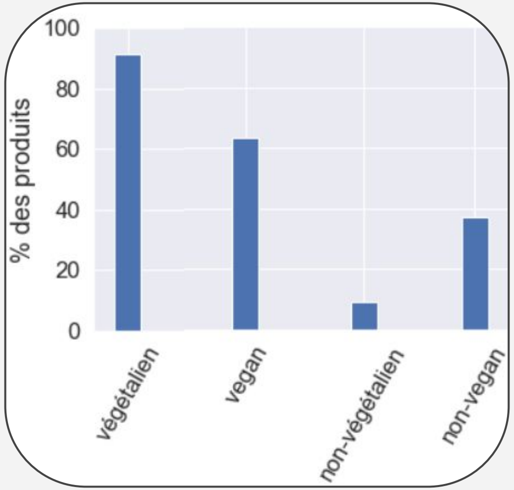
	df	sum_sq	mean_sq	F	PR(>F)
brands	3.0	412.838594	137.612865	2.042609	0.105656
Residual	7120.0	479682.288036	67.371108	NaN	NaN

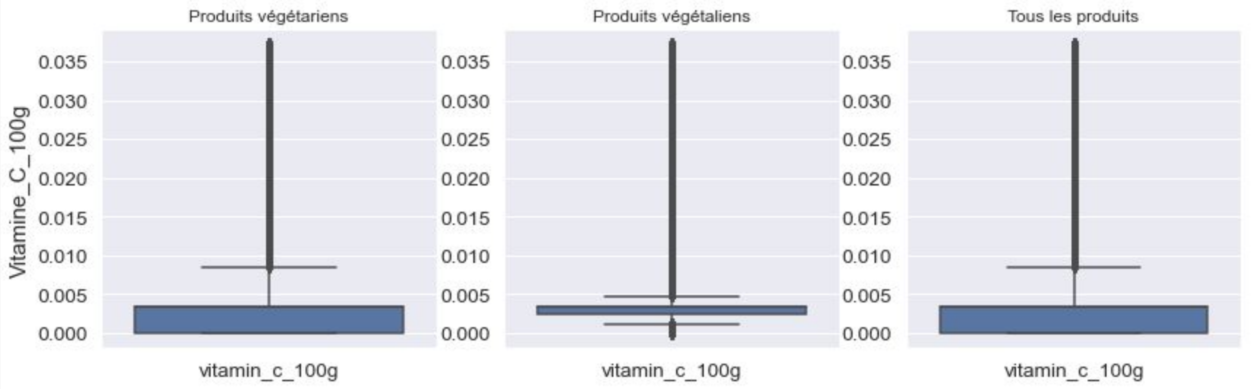



```python
anova_category = smf.ols('nutrition_score_fr_100g~countries_fr', data=data_food).fit()
sm.stats.anova_lm(anova_category)
```

	df	sum_sq	mean_sq	F	PR(>F)
countries_fr	4.0	1.446847e+04	3617.117573	58.239486	3.205471e-49
Residual	261847.0	1.626270e+07	62.107650	NaN	NaN

Produit Vegan et variables nutritionnelles

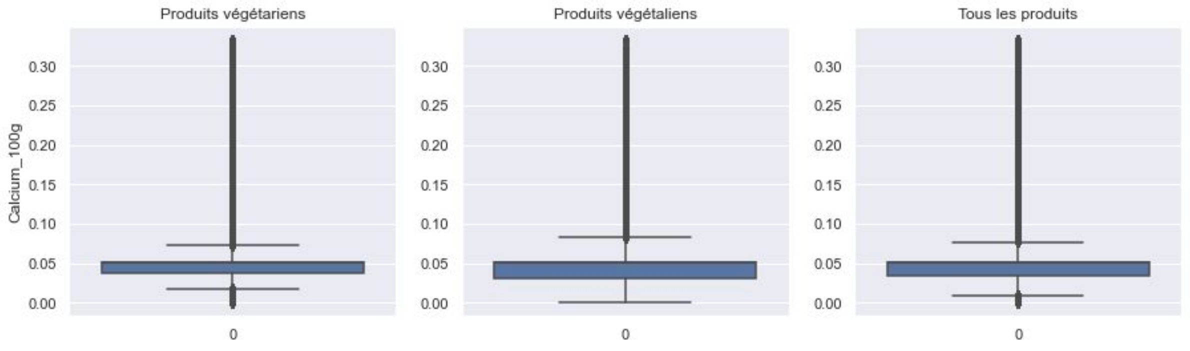


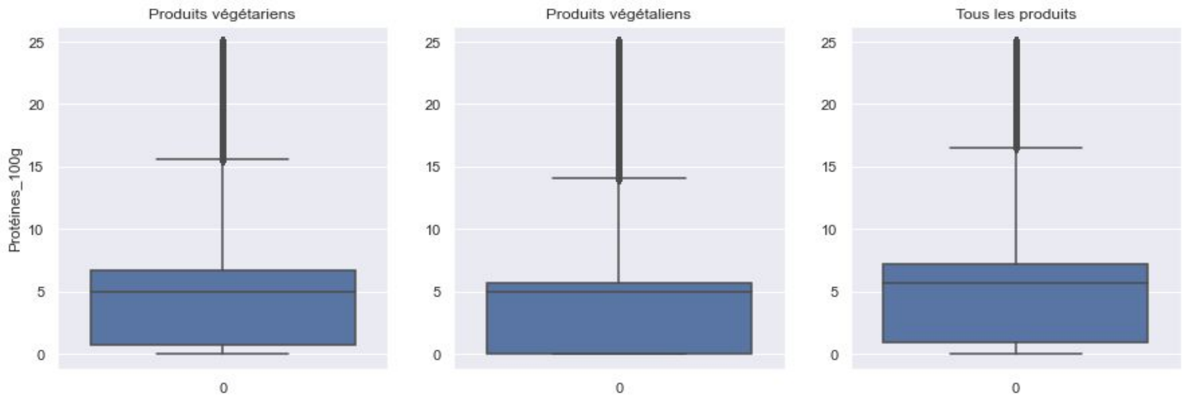


Produit Vegan et variables nutritionnelles

La répartition des protéines au sein des trois groupes est déséquilibré

Les étiquettes se contentent d'afficher l'information calorique des produits: les protéines, graisses et sucres.

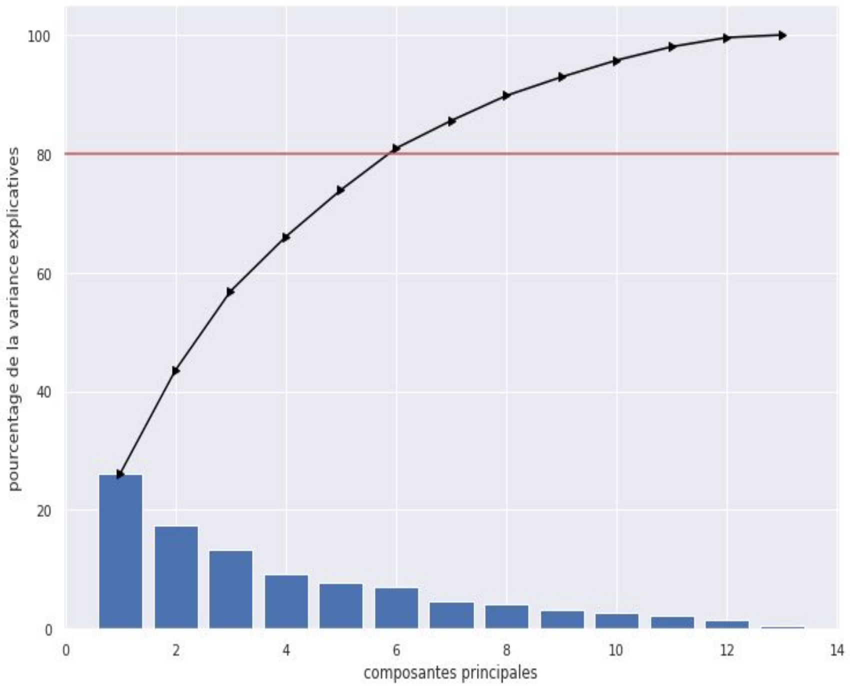




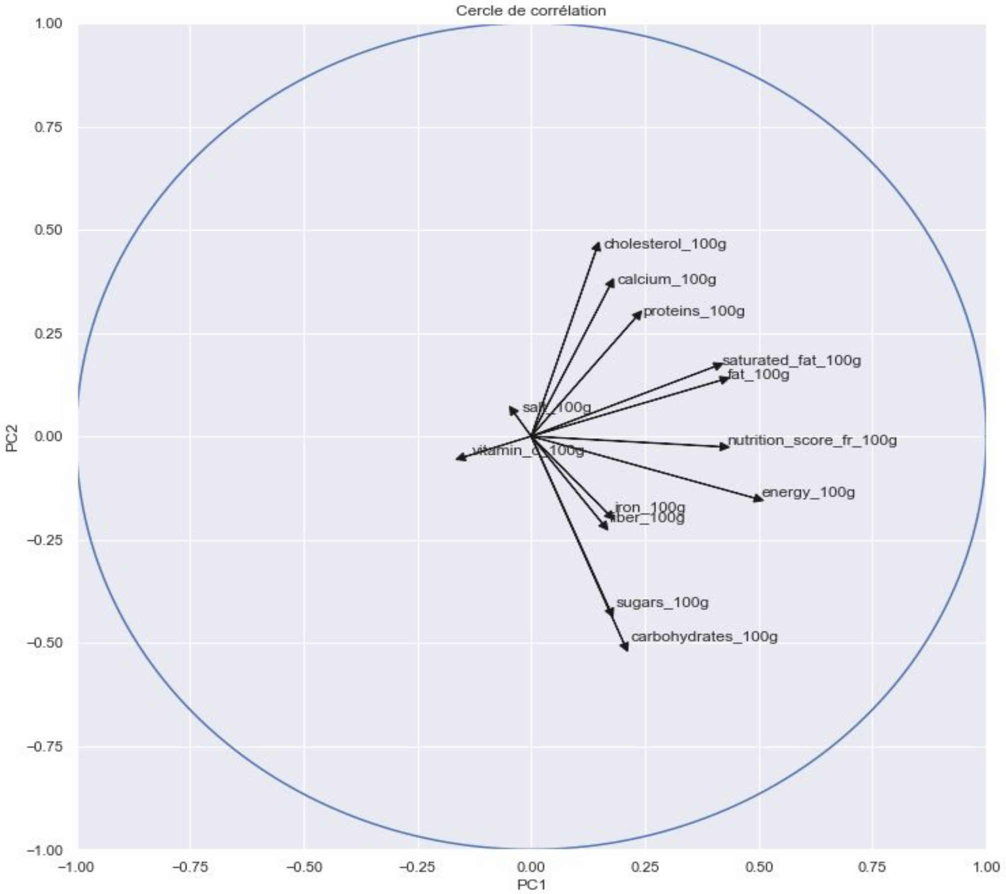
Composantes PCA

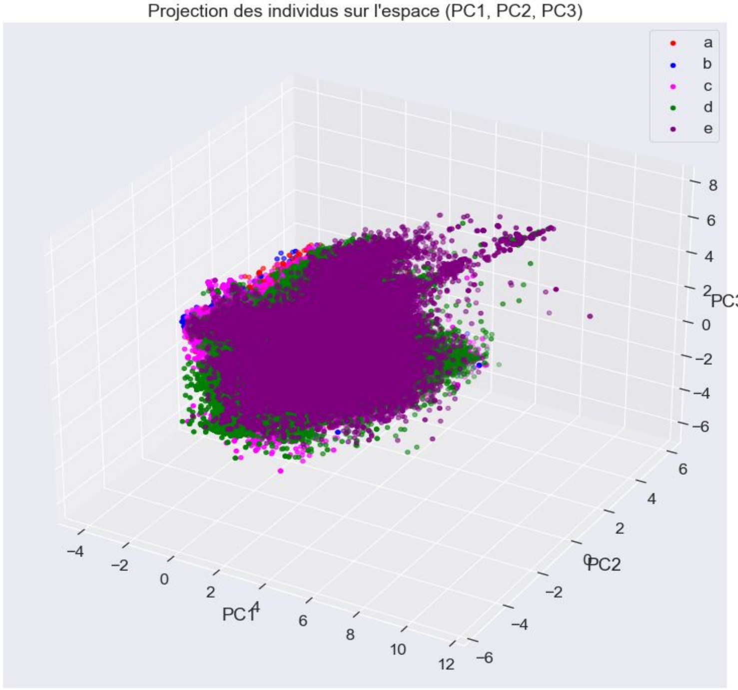
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# removing the mean and scaling to unit variance
scaler = StandardScaler()
X = data_food.select_dtypes('float64')
X_scaled = scaler.fit_transform(X)
#Instanciation de l'ACP
n = data_food.select_dtypes('float64').shape[1]
space_pca = PCA(n_components=n)
nutri_pca = space_pca.fit_transform(X_scaled)
#Explained variance in %
var_exp = space_pca.explained_variance_ratio_*100
print(var_exp.cumsum())
```

```
[ 26.13465762  40.77487013  54.35517329  63.54409163  71.41898725
  78.48797618  84.65082737  89.01634972  92.66052365  95.71636878
  97.99768929  99.54518255 100.        ]
```



Composantes PCA





Régression multiples

$$y = \sum_{i=1}^{n} a_i X_i + a_0$$

```
columns = data_food.select_dtypes('float64').columns
# variable cible
y = data_food['nutrition_score_fr_100g']
 #(variables explicative
X = data_food[columns.drop(['nutrition_score_fr_100g'])]
```

```
from sklearn.preprocessing import StandardScaler

# removing the mean and scaling to unit variance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
#split data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=0)
#fit model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
#prediction
y_pred = model.predict(X_test)
```

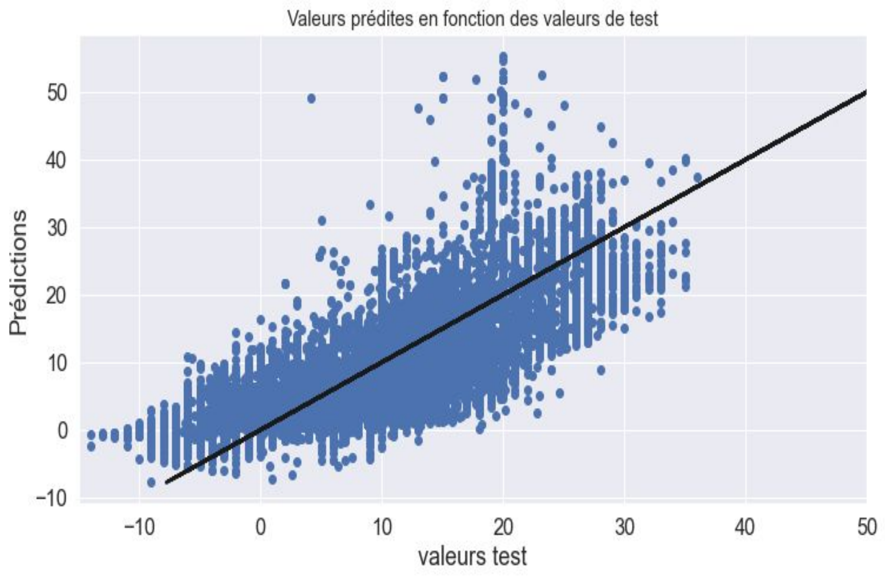
Régression multiples

model.coef_

variables importantes

variables moins importantes

variable	coefficient
energy_100g	2.778804
salt_100g	1.260009
fat_100g	0.995227
saturated_fat_100g	2.165956
cholesterol_100g	0.766248
sugars_100g	2.972427
carbohydrates_100g	-0.069229
fiber_100g	- 1.767607
proteins_100g	0.079393
vitamin_c_100g	0.001284
calcium_100g	-0.098990
iron_100g	0.147771
l'intercept	9.0671




```
from sklearn import metrics
print("l'Erreur absolue (L1)          :", metrics.mean_absolute_error(y_test, y_pred))
print("l'erreur quadratique (L2)      :", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("coefficient de détermination R2:", metrics.r2_score(y_test, y_pred))
```

```
l'Erreur absolue (L1)          : 3.1349504014936582
l'erreur quadratique (L2)      : 4.412398124608607
coefficient de détermination R2: 0.6858701921302498
```

Régression multiple

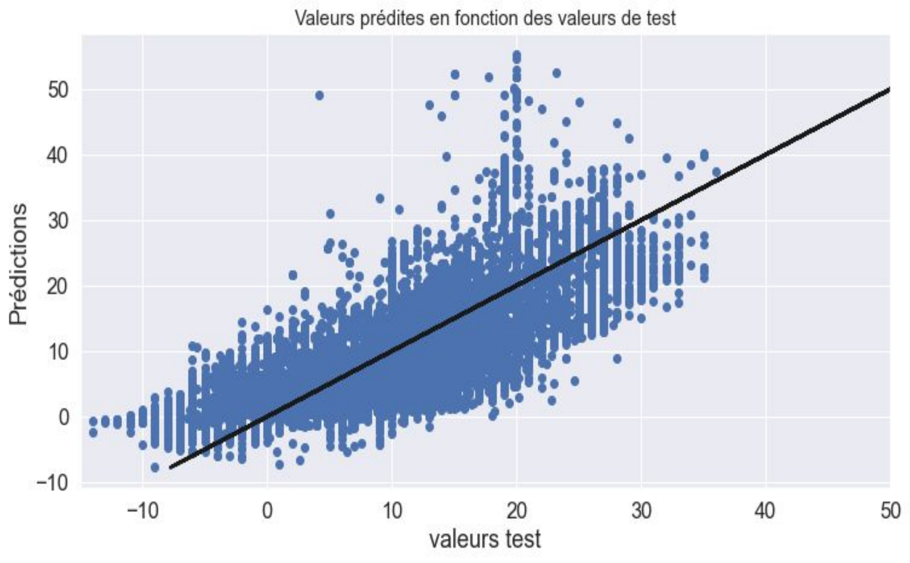
Erreur quadratique approche **10%** du Max de **Nuti-score**

Le modèle peut faire des prédictions raisonnablement bonnes

Amélioration du modèle

↕

Qualité et la quantité des données



Données OpenFood assez éparses

Données contenant beaucoup d’erreur

La qualité de nettoyage affect les prédictions