OPENCLASSROOMS

CentraleSupélec | université PARIS-SACLAY

Soutenance du Projet 3

Conception d'application d'aide à la décision pour Santé publique France

Par
Elisée TCHANA

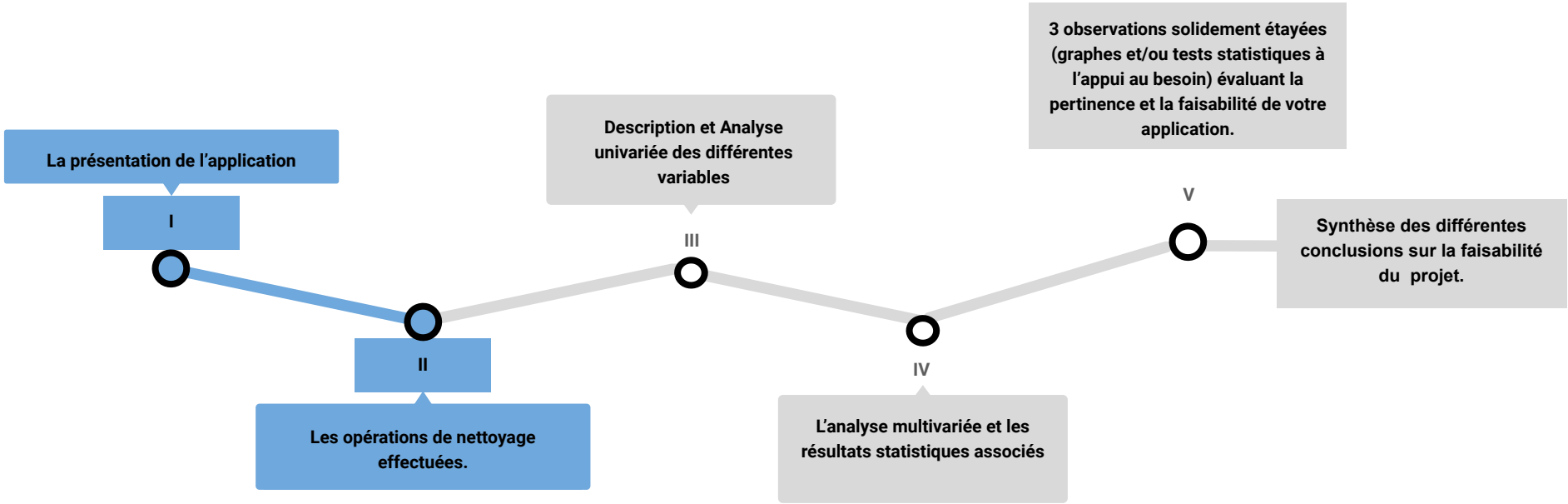
Mentor
Cyril MONTI

Mars 2021

OPENCLASSROOMS

CentraleSupélec | université PARIS-SACLAY

Plan de Soutenance



Objectifs du Projet

Répondre à l'appel à projets lancé par Santé publique France

Santé publique France

Objectifs

Trouver une idée innovante d'application en lien avec l'alimentation

Exploitation du jeu de données disponible sur Open Food Fact

OPEN FOOD FACTS
l'information alimentaire ouverte

Nutrition
NUTRI-SCORE
A B C D
Transformation
NOVA
1 2 3 4
Plus
Végan - Végétarien
Impact CO_2

Une application qui permet d'avoir une visualisation rapide du Nutri-Score pour un utilisateur n'ayant que quelques informations sur un produit pour une bonne alimentation.

Santé publique France

Le score varie de -15 à 40

OPENCLASSROOMS

CentraleSupélec | université PARIS-SACLAY

Data Analysis Workflow

Openfoodfacts

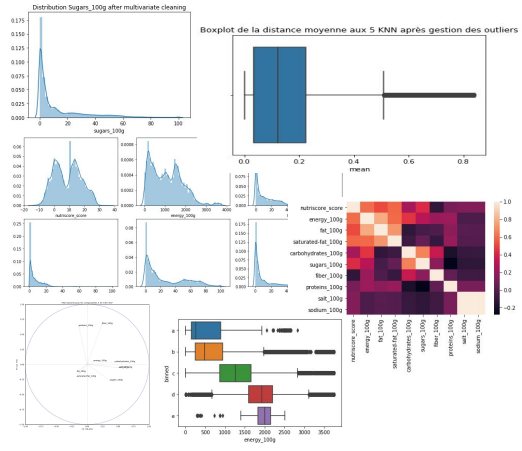
```
#import packages
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
import pickle
from sklearn.model_selection import train_test_split
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from scipy import spatial,stats

#import data
data= pd.read_csv("openfoodfacts.csv", sep='\t')
data.head()
```

	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name	abbrevia
0	0000000000017	http://world-en.openfoodfacts.org/produc09000...	kiliweb	1529059080	2018-06-15T10:38:00Z	1561463718	2019-06-25T11:55:18Z	Vitória crackers	
1	0000000000031	http://world-en.openfoodfacts.org/produc09000...	isagoofy	1539464774	2018-10-13T21:06:14Z	1539464817	2018-10-13T21:06:57Z	Cacao	
2	0000000000033327986	http://world-en.openfoodfacts.org/produc09000...	kiliweb	1574175736	2019-11-19T15:02:16Z	1574175737	2019-11-19T15:02:17Z	Filetes de pollo empanado	
3	0000000000100	http://world-en.openfoodfacts.org/produc09000...	del51	1444572561	2015-10-11T14:09:21Z	1444659212	2015-10-12T14:13:32Z	moutarde au moût de raisin	
4	0000000000111111111111	http://world-en.openfoodfacts.org/produc09000...	openfoodfacts-contributors	1560020173	2019-06-08T18:56:13Z	1560020173	2019-06-08T18:56:13Z	Sfudivn	

5 rows × 184 columns

```
sns.distplot(filled['sugars_100g'])
plt.title('Distribution Sugars_100g after multivariate cleaning')
```

```
Text(0.5, 1.0, 'Distribution Sugars_100g after multivariate cleaning')
```



Data Cleaning & Analysis

Data Sharing

Informations contenues dans le jeu de données

	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime		phylloquinone_100g	beta-glucan_100g	inositol_100g	carnitine_100g
0	0000000000017	http://world-en.openfoodfacts.org/product/0000...	kiliweb	1529059080	2018-06-15T10:38:00Z	1561463718	2019-06-25T11:55:18Z		NaN	NaN	NaN	NaN
1	0000000000031	http://world-en.openfoodfacts.org/product/0000...	isagoofy	1539464774	2018-10-13T21:06:14Z	1539464817	2018-10-13T21:06:57Z		NaN	NaN	NaN	NaN
2	000000000003327986	http://world-en.openfoodfacts.org/product/0000...	kiliweb	1574175736	2019-11-19T15:02:16Z	1574175737	2019-11-19T15:02:17Z	(...)	NaN	NaN	NaN	NaN
3	0000000000100	http://world-en.openfoodfacts.org/product/0000...	del51	1444572561	2015-10-11T14:09:21Z	1444659212	2015-10-12T14:13:32Z		NaN	NaN	NaN	NaN
4	0000000001111111111	http://world-en.openfoodfacts.org/product/0000...	openfoodfacts-contributors	1560020173	2019-06-08T18:56:13Z	1560020173	2019-06-08T18:56:13Z		NaN	NaN	NaN	NaN

```
print('lignes: ',data.shape[0],'\n colonnes: ',data.shape[1])
```

```
lignes:  1383645
 colonnes:  181
```

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.

Informations contenues dans le jeu de données

Santé publique France

POINTS ALIMENTS SOLIDES	POINTS BOISSONS	NUTRI-SCORE
-15 à -1	Eau	A
0 à 2	≤ 1	A B
3 à 10	2 à 5	A B C
11 à 18	6 à 9	A B C D
19 à 40	10 à 40	A B C D E

CALCUL DU NUTRI-SCORE

01 Attribution des points à différents facteurs nutritionnels

NUTRI-SCORE A B C D E — C'EST + FACILE DE MANGER MIEUX

Nutriments à limiter

Points N	Seuils pour les boissons				Seuils pour les matières grasses	
Points	Energie (kJ)	Sucres (g)	Energie (kJ)	Sucres (g)	Graisses saturées (g)	Graisses saturées (%)	Sodium (mg)
0	≤ 335	≤ 4,5	≤ 0	≤ 0	≤ 1	< 10	≤ 90
1	> 335	> 4,5	≤ 30	≤ 1,5	>1	< 16	> 90
2	> 670	> 9	≤ 60	≤ 3	>2	< 22	> 180
3	> 1005	> 13,5	≤ 90	≤ 4,5	>3	< 28	> 270
4	> 1340	> 18	≤ 120	≤ 6	>4	< 34	> 360
5	> 1675	> 22,5	≤ 150	≤ 7,5	>5	< 40	> 450
6	> 2010	> 27	≤ 180	≤ 9	>6	< 46	> 540
7	> 2345	> 31	≤ 210	≤ 10,5	>7	< 52	> 630
8	> 2680	> 36	≤ 240	≤ 12	>8	< 58	> 720
9	> 3015	> 40	≤ 270	≤ 13,5	>9	< 64	> 810
10	> 3350	> 45	> 270	> 13,5	>10	≥ 64	> 900
Gamme (points)	0 à 10	0 à 10	0 à 10	0 à 10	0 à 10	0 à 10	0 à 10
Total	Somme des points pour l'énergie, les sucres, les graisses saturées et le sodium						

Nutriments, aliments à encourager

Points P		Seuils pour les boissons		
Points	Fruits, légumes (%)	Fruits, légumes (%)	Fibres (g)	Protéines (g)
0	≤ 40	≤ 40	≤ 0,7	≤ 1,6
1	> 40	-	> 0,7	> 1,6
2	> 60	> 40	> 1,4	> 3,2
3	-	-	> 2,1	> 4,8
4	-	> 60	> 2,8	> 6,4
5	> 80		> 3,5	> 8,0
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	> 80	-	-
Gamme (points)	0 à 5	0 à 10	0 à 5	0 à 5
Total	Somme des points pour les consommations de fruits et légumes, les fibres et les protéines			

02 Choix de la méthode de calcul du score final

Points Négatifs N ≥ 11

- Score des fruits et légumes = 5 → **Points N - Points P**
- Score des fruits et légumes < 5 → **Points N - (points fibres + points fruits et légumes)**

Points Négatifs N < 11 ou pour les fromages → **Points N - Points P**

03 Attribution d'une couleur et d'une lettre

Score final variant de **-15 (qualité nutritionnelle élevée)** à **40 (faible qualité nutritionnelle)**

Aliments solides	Boissons	Logo
Min à -1	Eaux toujours en A	A
0 à 2	Min à 1	A B
3 à 10	2 à 5	A B C
11 à 18	6 à 9	A B C D
19 à max	10 à max	A B C D E

Quels produits concernés ?

- Tous les aliments transformés, excepté les herbes aromatiques, thés, cafés, levures...
- Toutes les boissons, excepté les boissons alcoolisées
- Excepté les produits dont la face la plus grande a une surface inférieure à 25 cm²

www.quotidiansmonassiette.fr T.Fiolet

Adapté de Julia C, Hercberg S (2017) Nutri-Score: evidence of the effective-ness of the French front-of-pack nutrition label. Ernahrungs. Umschau 64(12): 181–187

Nettoyage des données

Etude des individus ayant un Nutri-Score renseigné et un bon taux de remplissage général (ici arbitrairement moins de 80% de NaN)

```
df=df[df['nutriscore_score'].notnull()]
df.shape[0]
```

```
567328
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 567328 entries, 3 to 1383639
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   nutriscore_score     567328 non-null  float64
 1   energy_100g          565846 non-null  float64
 2   fat_100g             565805 non-null  float64
 3   saturated-fat_100g   565790 non-null  float64
 4   carbohydrates_100g   565544 non-null  float64
 5   sugars_100g          565796 non-null  float64
 6   fiber_100g           353943 non-null  float64
 7   proteins_100g        565808 non-null  float64
 8   salt_100g            566208 non-null  float64
 9   sodium_100g          566207 non-null  float64
dtypes: float64(10)
memory usage: 47.6 MB
```

On obtient une base de données avec un niveau de remplissage très satisfaisant

Séparation de notre jeu de données en 2 sets distincts :
base de données en 2 sets distincts :

- ***Un dataset "train" afin de pouvoir entraîner notre futur modèle de prédiction du nutriscore (70% des données)***
- ***Un dataset "test" afin de tester le niveau de précision de notre modèle en " mise en production" (30% des données)***

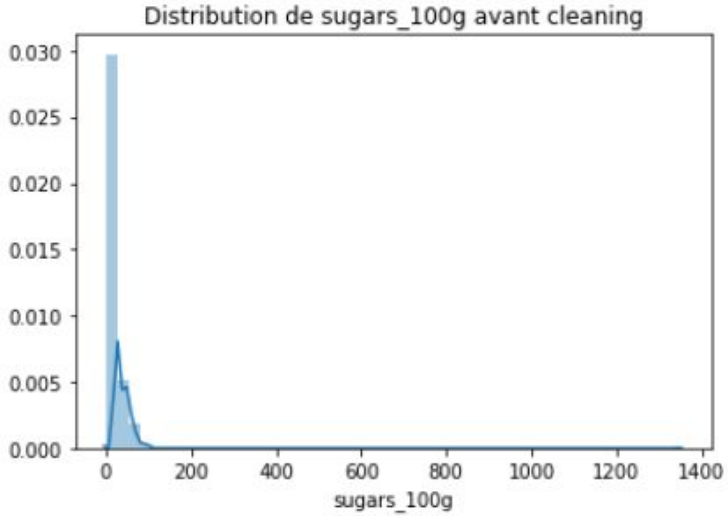
```
train, test = train_test_split(df, test_size=0.3)
train.shape[0]
```

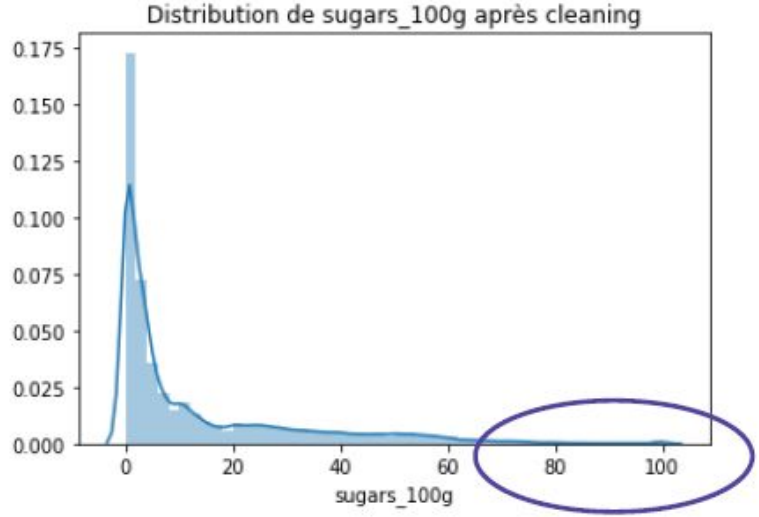
```
397129
```

Nettoyage des données

Liste des différentes étapes de cleaning :

1. Mise à NaN des valeurs < 0g et > 100g



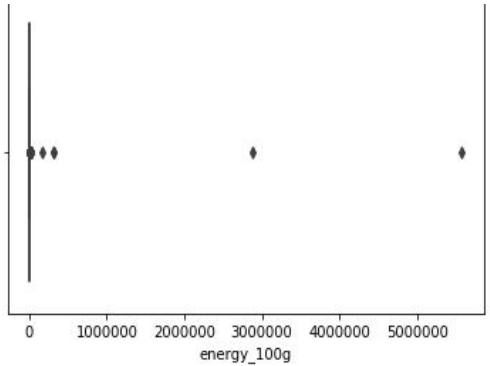


* Une partie de ces valeurs extrêmes devront être traitées via le cleaning cleaning d'outliers multivariés (partie 5)

Nettoyage des données

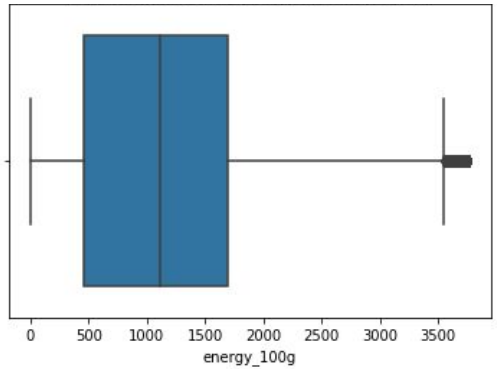
Mise à NaN des valeurs extrêmes pour l'énergie via la méthode des percentiles

Boxplot energy_100g avant cleaning

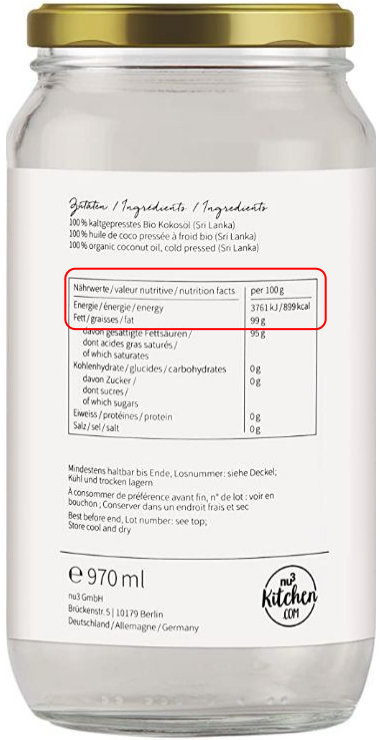


99,5% Percentile :
3761 Kj

Boxplot energy_100g après cleaning



0,995 percentile semble pertinent (3761 Kj) correspondent aux valeurs que l'on peut trouver sur l'huile de cuisson)

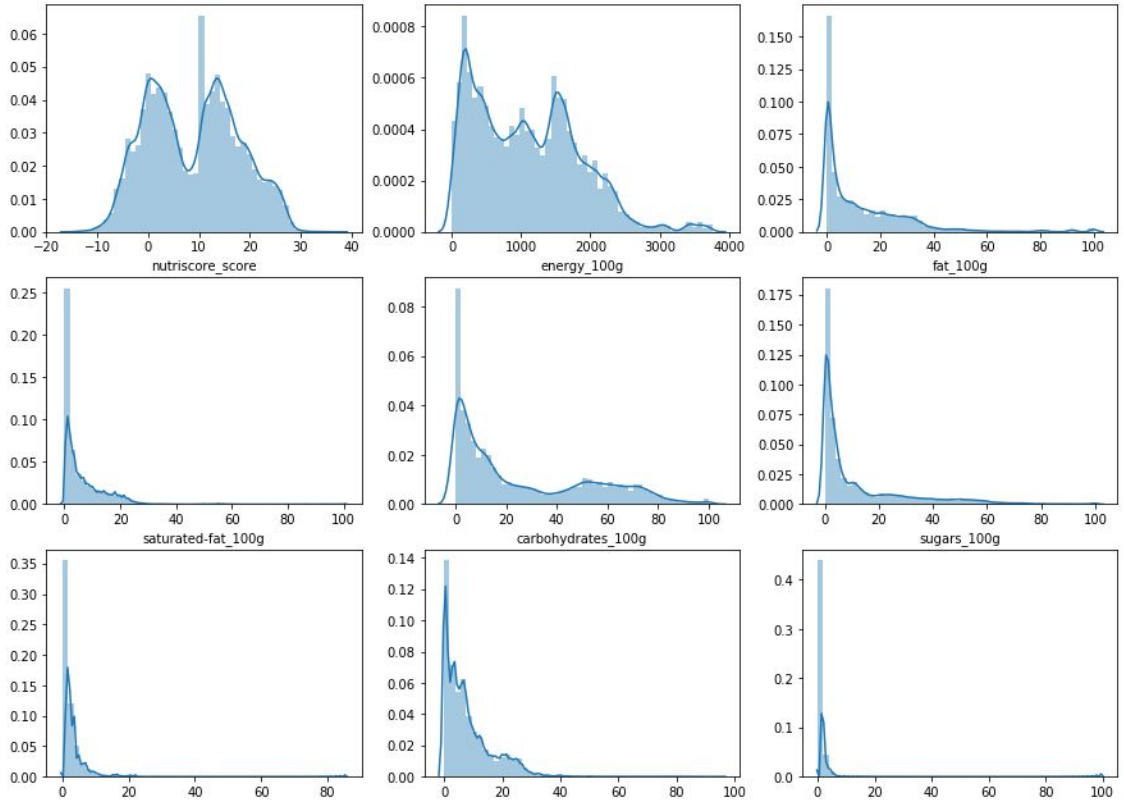


Nettoyage des données

Liste des différentes étapes de cleaning :

1. Mise à NaN des valeurs < 0g et > 100g
2. Mise à NaN des valeurs extrêmes pour l'énergie via la méthode des percentiles
3. Suppression des individus ayant plus de 2 NaN par ligne
4. Imputation des NaN restants via KNN Imputer
5. Suppression des valeurs aberrantes par somme (somme supérieure à 100g)
6. Suppression des outliers multivariés par distance euclidienne (KDTree)

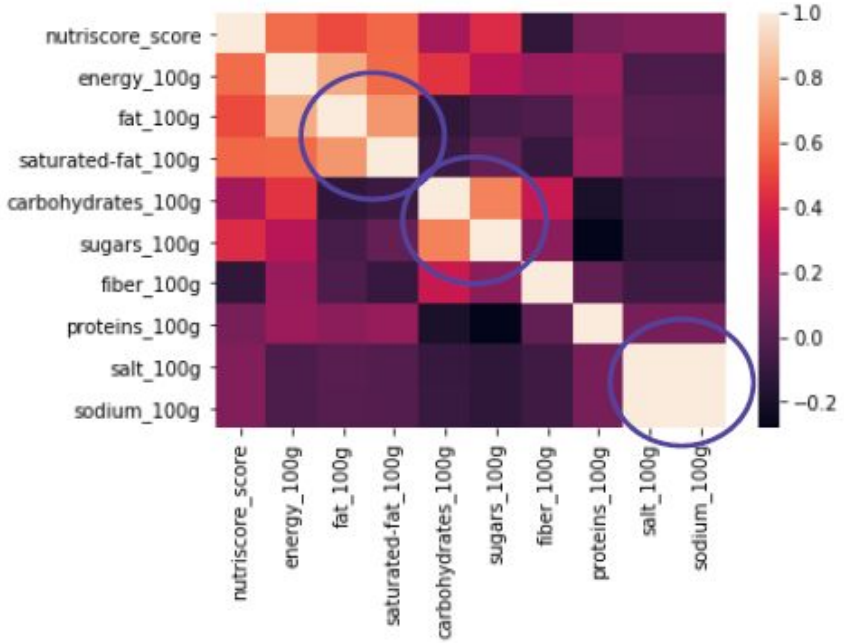
Nettoyage des données



Distribution de chaque feature du train dataset

Analyse et exploitation

Matrice de corrélation (coefficient de Pearson)



Nous pouvons observer une corrélation significative entre les caractéristiques suivantes:

- **graisses / graisses saturées (fat/ saturated-fat)**
- **glucides / sucres (carbohydrates/ sugars)**
- **sel / sodium (salt/sodium)**

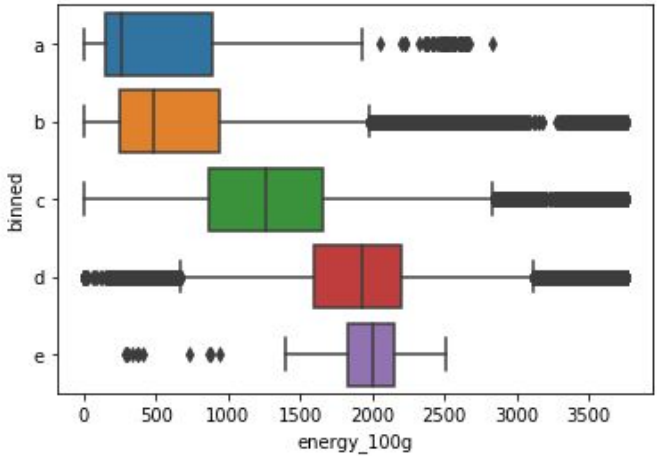
Analyse et exploitation

Test de normalité normalité (KosmogorovKosmogorov-Smirnov)

```
====================
null hypothesis : studied feature follow a normal distribution.
 ====================


D= 0.718031325990802
P-value= 0.0
P-value is lower than alpha (0.05) we can reject the null hypothesis
 => nutriscore_score does not follow normal distribution
----------------

D= 0.9882169779727689
P-value= 0.0
P-value is lower than alpha (0.05) we can reject the null hypothesis
 => energy_100g does not follow normal distribution
----------------

D= 0.6546795418476797
P-value= 0.0
P-value is lower than alpha (0.05) we can reject the null hypothesis
 => fat_100g does not follow normal distribution
----------------
```

Analyse et exploitation

Test ANOVA



	F	P_value
energy_100g	48402.018888	0.0
fat_100g	29275.871175	0.0
saturated-fat_100g	47963.338632	0.0
carbohydrates_100g	6747.897986	0.0
sugars_100g	18350.684481	0.0
fiber_100g	3257.267355	0.0
proteins_100g	1461.444236	0.0
salt_100g	2663.101043	0.0
sodium_100g	2623.958628	0.0

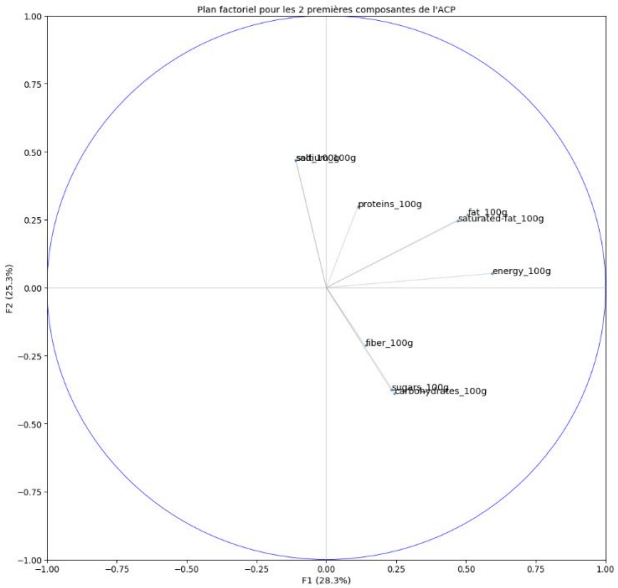
Objectif : le test d'ANOVA permet d'étudier le comportement d'une variable quantitative à une ou plusieurs variables qualitatives.

P Value= 0 (on peut rejeter l'hypothèse H0 avec 0% de risque, les catégories ont donc une influence sur la distribution des variables étudiées)

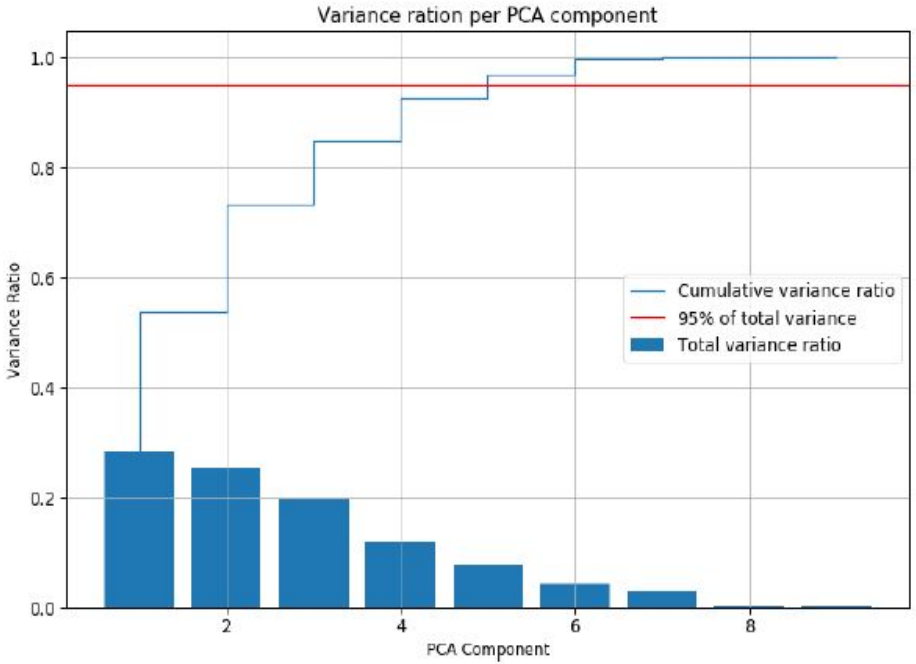
***Résultat du test à relativiser, on ne respecte pas la condition normalité de nos données (nécessaire pour un test ANOVA)**

Analyse et exploitation

Test ACP



On retrouve les mêmes corrélations que dans notre matrice (on pourra les prendre en compte pour la partie prédiction)



D'après cette vue, on pourrait représenter ~95% de nos données uniquement avec 5 features.
Ici, notre nombre de features est déjà suffisamment petit.

Analyse et exploitation

Exploitation :
Création d'une fonction de cleaning similaire au cleaning du train :
Liste des différentes étapes de cleaning:

1. Mise à NaN des valeurs < 0g et > 100g
2. Mise à NaN des valeurs extrêmes pour l'énergie via la méthode des percentiles
3. ~~Suppression des individus ayant plus de 2 NaN par ligne~~
4. Imputation des NaN restants via KNN Imputer
5. Suppression des valeurs aberrantes par somme (somme supérieure à 100g)
6. Suppression des outliers multivariés par distance euclidienne (KDTreeKDTree)

Fitté sur le train dataset

Analyse et exploitation


```
======================
Début de la phase de cleaning
======================
Gestion des valeurs inférieures à 0g et supérieures à 100g:
138  valeurs aberrantes mises à NaN

_____________________________
Gestion des outliers pour la variable: energy_100g:
850  valeurs aberrantes mises à NaN

_____________________________
Remplissage des NaNs via KNN Imputer (fitté sur le train)
Les Nans du dataframe ont été imputés via KNNImputer

_____________________________
Suppression des lignes avec une somme des macronutriments supérieure à 100g:
11106  lignes supprimées

_____________________________
Suppression outliers multivariés avec KDTree:
2351  lignes supprimées

_____________________________
```

Analyse et exploitation

b) Exploitation : Prédiction par régression linear multiple

Score de la prédiction (R^2) sur train: 0.621~

Score de la prédiction (R^2): **0.626~**

	prédiction	nutriscore
count	159037.000000	148610.000000
mean	9.220771	9.030186
std	7.111163	8.893977
min	-44.044231	-15.000000
25%	3.660006	1.000000
50%	7.293254	10.000000
75%	13.987662	16.000000
max	58.235002	36.000000

Poids de chaque variable dans la régression:

	Feature	Coef
0	energy_100g	3.099802
1	fat_100g	0.010714
2	saturated-fat_100g	2.877117
3	carbohydrates_100g	-0.724187
4	sugars_100g	3.840426
5	fiber_100g	-1.707929
6	proteins_100g	0.554542
7	salt_100g	2.751464
8	sodium_100g	-1.190483

Analyse et exploitation

Prédiction via régression KNN Regressor :

Ici, l'optimisation a été plus poussée

1) GridSearchCV va également nous permettre de chercher les paramètres optimum pour notre modèle (ici uniquement le nombre de voisins)

2) Etude des features "Très corrélées" (non-concluante)

```
for i in range(len(scoring)):

    grid= GridSearchCV(KNeighborsRegressor(),param_grid,cv=5,scoring=scoring[i])
    grid.fit(xtrain,ytrain)

    print('Methode de Scoring: ',scoring[i])
    print('Meilleur score :',grid.best_score_)
    print('Meilleurs Paramètres :',grid.best_params_,'\n_____________________')

    if i == 0 :
        model_1 = grid.best_estimator_
    elif i == 1 :
        model_2 = grid.best_estimator_
    else:
        model_3 = grid.best_estimator_
```

```
Methode de Scoring:  r2
Meilleur score : 0.9485970162173958
Meilleurs Paramètres : {'n_neighbors': 5}
_____________________
Methode de Scoring:  neg_mean_absolute_error
Meilleur score : -0.921002810035702
Meilleurs Paramètres : {'n_neighbors': 1}
_____________________
Methode de Scoring:  neg_mean_squared_error
Meilleur score : -4.062249569922076
Meilleurs Paramètres : {'n_neighbors': 5}
_____________________
```

Analyse et exploitation

2) Etude des features “trop corrélées” (non-concluante):

Suppression pas à pas des variables très corrélées entre elles

Rappel score avec toutes les variables

```
Methode de Scoring:  r2
Meilleur score : 0.9485970162173958
Meilleurs Paramètres : {'n_neighbors': 5}
```

	col	r2	neg_mean_absolute_error	neg_mean_squared_error
sans saturated fat	saturated-fat_100g	0.923408	-1.259147	-6.052823
sans sugars	sugars_100g	0.932598	-1.124096	-5.326558
sans sodium	sodium_100g	0.946746	-0.957152	-4.208524
sans saturated fat & sugars	[saturated-fat_100g, sugars_100g]	0.897964	-1.509248	-8.063685
sans saturated fat & sodium	[saturated-fat_100g, sodium_100g]	0.9221	-1.277486	-6.156246
sans sugars et sodium	[sugars_100g, sodium_100g]	0.931532	-1.144292	-5.410799
sans saturates fat & sugars & sodium	[saturated-fat_100g, sugars_100g, sodium_100g]	0.89735	-1.519647	-8.112165

Analyse et exploitation

b. Exploitation :

Prédiction via KNN Regressor : Score final sur le test dataset (R^2): 0,93

Ici, l'optimisation a été plus poussée

•GridSearchCV va également nous permettre de chercher les paramètres optimum pour notre modèle (ici uniquement le nombre de voisins)

•Etude des features ayant une forte corrélation

OPENCLASSROOMS
Analyse et exploitation

b) Exploitation : Prédiction par régression linear multiple

Score de la prédiction (R^2) sur train: 0.621
Score de la prédiction (R^2): **0.626**

	prédiction	nutriscore
count	159037.000000	148610.000000

Poids de chaque variable dans la régression:

	Feature	Coef
0	energy_100g	3.099802

Analyse et exploitation

	nutriscore_score	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	sodium_100g
0	-6.0	444.0	0.59	0.0	22.35	9.41	2.4	3.53	0.22	0.088
1	-6.0	444.0	0.59	0.0	22.35	NaN	2.4	3.53	0.22	0.088
2	-6.0	444.0	0.59	NaN	NaN	9.41	2.4	3.53	0.22	0.088
3	-6.0	444.0	0.59	0.0	NaN	NaN	2.4	3.53	NaN	0.088
4	-6.0	NaN	0.59	0.0	NaN	9.41	2.4	NaN	NaN	0.088
5	-6.0	NaN	NaN	NaN	22.35	NaN	2.4	3.53	NaN	0.088
6	-6.0	NaN	NaN	NaN	22.35	9.41	2.4	NaN	NaN	NaN
7	-6.0	NaN	NaN	NaN	NaN	9.41	NaN	NaN	NaN	0.088

	nutriscore_score	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	sodium_100g	predict
0	-6.0	444.000000	0.590000	0.000000e+00	22.350000	9.41	2.400	3.530000	0.220000	0.088	-6.0
1	-6.0	444.000000	0.590000	0.000000e+00	22.350000	8.43	2.400	3.530000	0.220000	0.088	-6.0
2	-6.0	444.000000	0.590000	-8.881784e-16	22.350000	9.41	2.400	3.530000	0.220000	0.088	-6.0
3	-6.0	444.000000	0.590000	0.000000e+00	22.350000	8.43	2.400	3.530000	0.000000	0.088	-6.0
4	-6.0	185.800000	0.590000	0.000000e+00	12.626667	9.41	2.400	1.546667	0.483483	0.088	-3.8
6	-6.0	444.000000	0.590000	-8.881784e-16	22.350000	9.41	2.400	3.530000	0.000000	0.000	-6.0
7	-6.0	638.666667	7.951667	4.581667e+00	17.390000	9.41	0.926	5.280000	0.217333	0.088	3.8

Conclusion

Après le nettoyage des données et l'exploration, certaines variables sont nécessaires pour prédire un Nutri-Score automatique :

- **L'Energy**
- **Le Sucre**
- **La Graisse**
- **La Graisse Saturée**

Santé publique France

Merci pour votre attention !!