Projet Santé Publique

Efkan TUREDI

Intro

Nous voulons concevoir une application qui aura comme cahier des charges:

(i) Scanner les codes des aliments pour aller chercher le nutri-score

(ii) Donner une note aux aliments scannés et

(iii) Donner une note journalière à l'alimentation de l'utilisateur (health tracking)

Nous nous basons sur la base de données Open Food Facts

Choisissons les données utiles pour notre application

- Nous nommons les colonnes utiles dans une variable “used_features”, et utilisons cette variable avec pandas pour charger uniquement la database qui nous intéresse, car la database à beaucoup d’information
- Nous obtenons donc une database avec 13 colonnes et beaucoup de lignes avant nettoyage

[35]
```
filepath = '/Users/efkanturedi/Corteze/openfoodfacts.csv'
used_features = ['product_name','code','pnns_groups_1','nutriscore_grade','energy-kcal_100g','proteins_100g',
'carbohydrates_100g','sugars_100g','fat_100g','saturated-fat_100g','fiber_100g','sodium_100g',
'nutrition-score-fr_100g']
data = pd.read_csv(filepath, sep='\t', usecols=used_features)
```

```
/Users/efkanturedi/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3165: DtypeWarning: Columns
(0) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

[36]
```
pd.options.display.float_format = "{:.1f}".format
```

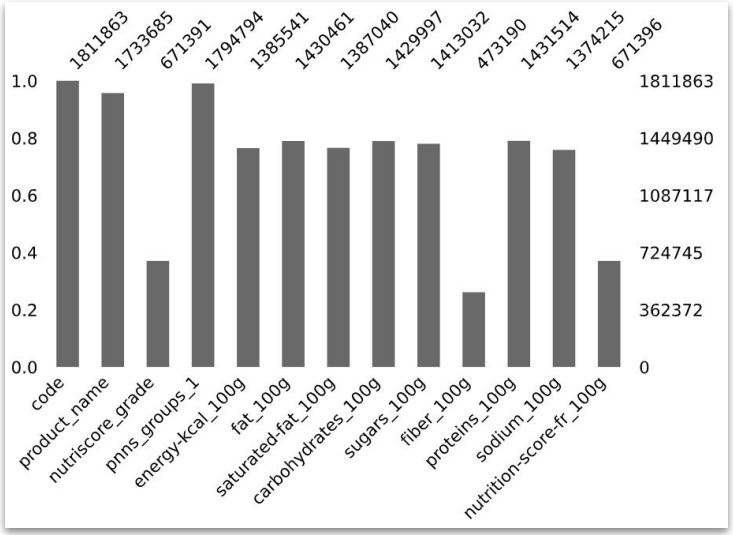
[37]
```
data.shape
```

```
(1811863, 13)
```

Un peu de nettoyage!

Nos données ont quelques soucis...

- Données abérrantes
- Absences de données ou NaN



	sugars_100g	fiber_100g	proteins_100g	sodium_100g
count	1413032.0	473190.0	1431514.0	1374215.0
mean	14.0	5.1	8.8	1.2
std	841.5	1453.7	146.2	345.6
min	-1.0	-20.0	-500.0	0.0
25%	0.6	0.0	1.3	0.0
50%	3.6	1.6	6.0	0.2
75%	18.0	3.6	12.3	0.6
max	999999.0	999999.0	173000.0	399999.6

...que nous avons résolus de la manière suivante

- Données abérrantes:
  - Remplacement avec la valeur maximale possible (0 ou 100g par exemple)
- Absences de données ou NaN:
  - Si un ligne a uniquement des NaNs, on la supprime

	sugars_100g	fiber_100g	proteins_100g	sodium_100g
count	1413032.0	473190.0	1431514.0	1374215.0
mean	13.3	3.0	8.6	0.6
std	19.6	5.0	9.8	2.9
min	0.0	0.0	0.0	0.0
25%	0.6	0.0	1.3	0.0
50%	3.6	1.6	6.0	0.2
75%	18.0	3.6	12.3	0.6
max	100.0	100.0	100.0	100.0

[9]
```
non_negative_capped_data = ['fat_100g','saturated-fat_100g','carbohydrates_100g','sugars_100g','fiber_100g',
'proteins_100g','sodium_100g']
numeric_data = ['energy-kcal_100g','fat_100g','saturated-fat_100g','carbohydrates_100g','sugars_100g',
'fiber_100g','proteins_100g','sodium_100g','nutrition-score-fr_100g']
```

[10]
```
for col in non_negative_capped_data:
    data_copy[col][data_copy[col] < 0] = 0
    data_copy[col][data_copy[col] > 100] = 100
data_copy['energy-kcal_100g'][data_copy['energy-kcal_100g'] > 1500] = 1500
```

Attention au piège des données “dupliquées”(1/2)

- A première vue il semble y avoir beaucoup de duplicats... Mais en fait ce sont surtout des aliments identiques mais de marques différentes.
- On montre ici l’exemple du steak haché pur boeuf

	code	product_name	brands	nutriscore_grade	pnns_groups_1	energy-kcal_100g	fat_100g	saturated-fat_100g
601200	3245414151017	Steak haché pur boeuf	Carrefour	a	Fish Meat Eggs	121.0	5.0	1.6
601457	3245415803748	Steak haché pur boeuf	NaN	NaN	Fish Meat Eggs	0.0	NaN	NaN
652988	3273230065355	Steak haché pur boeuf	Leader Price	b	Fish Meat Eggs	NaN	14.0	5.9
653084	3273230258658	Steak haché pur boeuf	NaN	c	Fish Meat Eggs	211.0	15.0	7.0
702409	3381590007806	Steak haché pur boeuf	oumaty	d	Fish Meat Eggs	252.0	20.0	8.6
916947	4056489361961	Steak haché pur boeuf	NaN	NaN	unknown	207.0	15.0	6.2
925778	40882048	Steak haché pur boeuf	L'Étal du Boucher,Lidl	a	Fish Meat Eggs	8.6	8.6	0.5
1081597	5901885541785	Steak haché pur boeuf	NaN	NaN	unknown	252.0	20.0	8.6
1469658	99999995	Steak haché pur boeuf	NaN	a	Fish Meat Eggs	131.0	5.0	2.3

	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name	abbreviated_product_name	generic_name
1366723	7340011495437	http://world-en.openfoodfacts.org/product/7340...	halal-app-chakib	1610378294	2021-01-11T15:18:14Z	1610393709	2021-01-11T19:35:09Z	NaN	NaN	NaN
1366724	7340011495437	http://world-en.openfoodfacts.org/product/7340...	halal-app-chakib	1610378294	2021-01-11T15:18:14Z	1610393709	2021-01-11T19:35:09Z	NaN	NaN	NaN

2 rows × 186 columns

Attention au piège des données “dupliquées”(2/2)

[10]

```
data[data['url'].duplicated(keep=False)]
```

	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name	abbreviated_product_name
652037	30383354190402	http://world-en.openfoodfacts.org/product/3038...	openfoodfacts-contributors	1608035756	2020-12-15T12:35:56Z	1610702480	2021-01-15T09:21:20Z	basilic	NaN
652038	30383354190402	http://world-en.openfoodfacts.org/product/3038...	openfoodfacts-contributors	1608035756	2020-12-15T12:35:56Z	1610702583	2021-01-15T09:23:03Z	basilic	NaN
894243	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	NaN
894244	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	NaN
1041540	3770008983205	http://world-en.openfoodfacts.org/product/3770...	r-x	1614201389	2021-02-24T21:16:29Z	1614242412	2021-02-25T08:40:12Z	REMYX VODKA Aquatique	NaN
1041541	3770008983205	http://world-en.openfoodfacts.org/product/3770...	r-x	1614201389	2021-02-24T21:16:29Z	1614242412	2021-02-25T08:40:12Z	REMYX VODKA Aquatique	NaN
1366723	7340011495437	http://world-en.openfoodfacts.org/product/7340...	halal-app-chakib	1610378294	2021-01-11T15:18:14Z	1610393709	2021-01-11T19:35:09Z	NaN	NaN
1366724	7340011495437	http://world-en.openfoodfacts.org/product/7340...	halal-app-chakib	1610378294	2021-01-11T15:18:14Z	1610393709	2021-01-11T19:35:09Z	NaN	NaN
1434598	7798049540559	http://world-en.openfoodfacts.org/product/7798...	openfoodfacts-contributors	1615222625	2021-03-08T16:57:05Z	1615337559	2021-03-10T00:52:39Z	lentejas	NaN
1434599	7798049540559	http://world-en.openfoodfacts.org/product/7798...	openfoodfacts-contributors	1615222625	2021-03-08T16:57:05Z	1615337611	2021-03-10T00:53:31Z	lentejas	NaN

10 rows × 186 columns

[15]

```
data['url'].isna().sum()
```

```
0
```

Mais que faire des NaNs?

Il y en a énormément!

KNNImputer en solution!

- Uniquement valable sur les valeurs numériques.
- On sépare d'abord nos valeurs numériques dans un tableau, qu'on fusionne ensuite avec notre tableau de caractères, pour retrouver notre tableau complet, pour utilisation dans la partie analyse
- Toutefois lourd en temps de calcul!

```
[20]
secondary_data=new_data[:][['code','product_name','pnns_groups_1','nutriscore_grade']]
secondary_data = secondary_data.reset_index(drop=True)

[21]
full_num = new_data[numeric_data]

[22]
random_sample_full_num=full_num.sample(20000)
random_sample_full_num

[23]
imputer = KNNImputer(n_neighbors=3)
imputer.fit(random_sample_full_num)

KNNImputer(n_neighbors=3)

[24]
data_Knned = imputer.transform(full_num)

[25]
df = pd.DataFrame(data_Knned,columns = numeric_data)
```

Nutriscore et Nutrigrade (1/2)

- Le **Nutrigrade** est un indicateur à destination du consommateur, alors que le **Nutriscore** est l'indicateur permettant d'obtenir le nutrigrade.
- Le Nutrigrade va de A (meilleure note) à E (plus mauvaise note) et le Nutriscore va de -15 (meilleure note) à 40 (plus mauvaise note)
- Le tableau suivant indique la notation permettant d'obtenir le nutrigrade

Foods (points)	Beverages (points)	Colour
Min to -1	Water	Dark green
0 to 2	Min to 1	Light green
3 to 10	2 to 5	Yellow
11 to 18	6 to 9	Light orange
19 to max	10 to max	Dark orange

Source: Nutriscore calculations

Nutriscore et Nutrigrade (2/2)

- Les boissons ont une méthodologie différente de calcul de nutriscore
- On “Impute” les valeurs A, B, C, D ou E en fonction du nutriscore avec le code présenté

[29]

```
def grader_food(x):
        if x<=-1:
            return 'a'
        elif (x>-1)&(x<=2):
            return 'b'
        elif (x>2)&(x<=10):
            return 'c'
        elif (x>10)&(x<=18):
            return 'd'
        else:
            return 'e'

def grader_beverages(x):
        if (x<=0):
            return 'a'
        elif (x>0)&(x<=1):
            return 'b'
        elif (x>1)&(x<=5):
            return 'c'
        elif (x>5)&(x<=9):
            return 'd'
        else:
            return 'e'
```

[30]

```
for ind in range(len(final.index)):
    if final['nutriscore_grade'].loc[ind] is np.nan:
        if final['pnns_groups_1'].loc[ind] == 'Beverages':
            final['nutriscore_grade'].loc[ind] = grader_beverages(final['nutrition-score-fr_100g'].loc[ind])
        else:
            final['nutriscore_grade'].loc[ind] = grader_food(final['nutrition-score-fr_100g'].loc[ind])
```

Notre database de sortie pour analyse

[31]

```
final
```

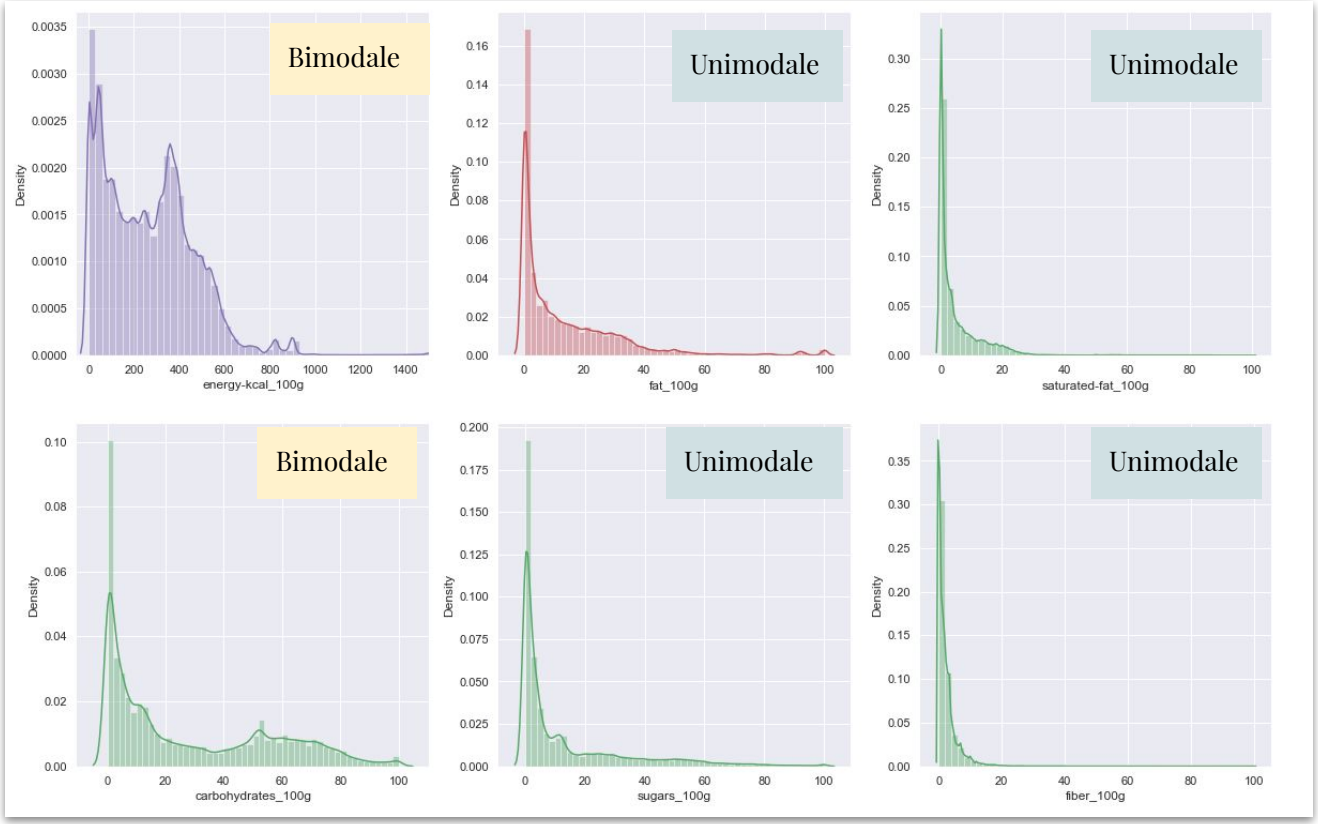
	code	product_name	pnns_groups_1	nutriscore_grade	energy-kcal_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	sodium_100g	nutrition-score-fr_100g
0	0000000000017	Vitória crackers	unknown	b	375.0	7.0	3.1	70.1	15.0	6.9	7.8	0.6	0.0
1	000000000004622327	Hamburguesas de ternera 100%	unknown	c	874.9	15.1	6.1	2.6	1.0	0.0	15.7	0.8	3.7
2	0000000000100	moutarde au moût de raisin	Fat and sauces	d	171.0	8.2	2.2	29.0	22.0	0.0	5.1	1.8	18.0
3	0000000000123	Sauce Sweety chili 0%	unknown	a	21.0	0.0	0.0	4.8	0.4	1.8	0.2	0.8	-3.0
4	0000000000178	Mini coco	unknown	b	60.0	3.0	1.0	10.0	3.0	1.6	2.0	0.5	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1469657	9999999175305	Erdbeerkuchen 1019g tiefgefroren	Sugary snacks	d	46.7	7.6	4.8	35.0	24.0	2.0	2.6	0.1	12.0
1469658	99999995	Steak haché pur boeuf	Fish Meat Eggs	a	131.0	5.0	2.3	0.0	0.0	0.0	21.5	0.1	-2.0
1469659	9999999901	Scs	unknown	b	100.0	12.0	1.0	2.0	1.0	3.1	1.0	0.4	0.0
1469660	9999999990397	Fati	unknown	b	24.0	0.3	0.0	2.4	0.6	2.6	1.6	0.3	-0.7
1469661	9999999999994	Light & Free SKYR A BOIRE	unknown	b	0.0	0.2	0.1	8.0	7.8	0.5	5.5	0.1	0.0

1469662 rows × 13 columns

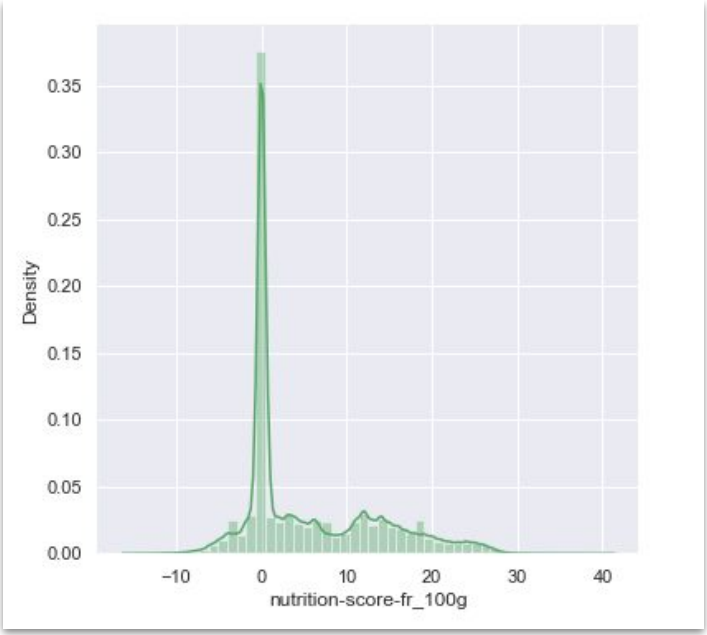
C’est parti pour l’analyse!

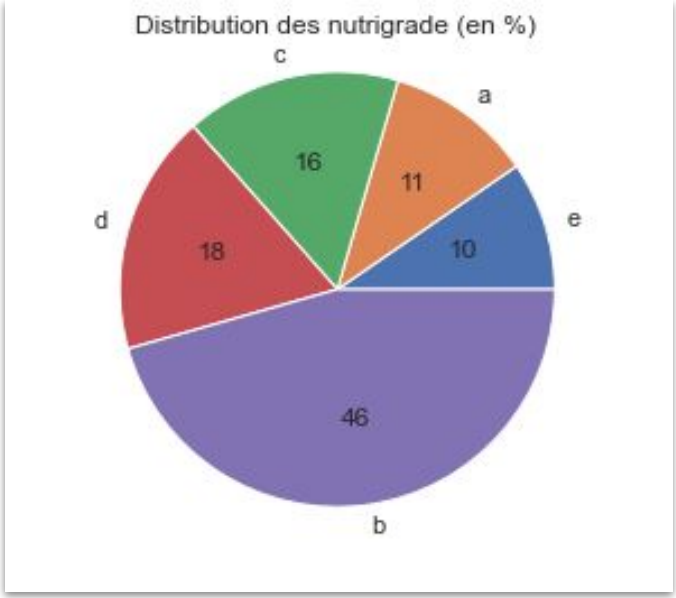
Analyse Univariée (1/2)

- La majorité de nos variables ont une distribution unimodale
- “Carborhydrates” et “energy-kcal”, et “nutriscore” ont des distributions bi-modale



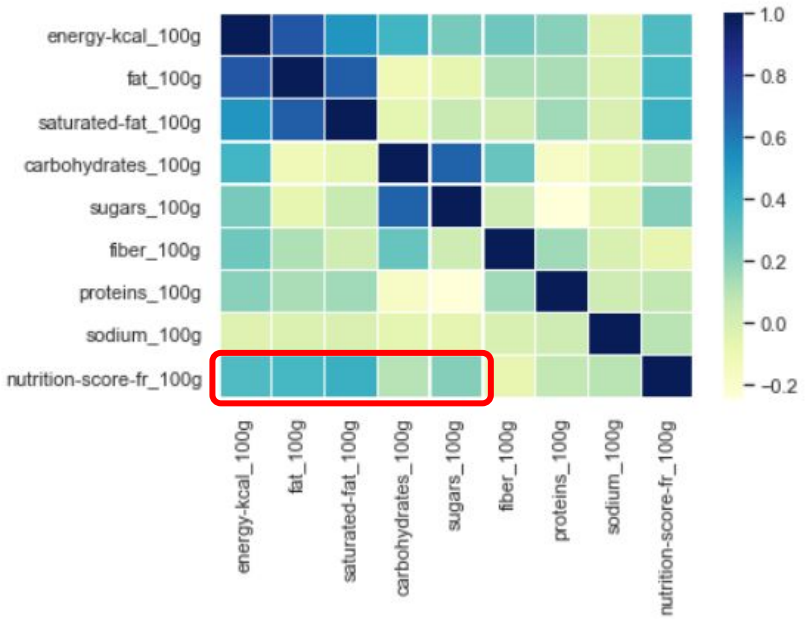
Analyse Univariée (2/2) - Nutriscore et Nutrigrade



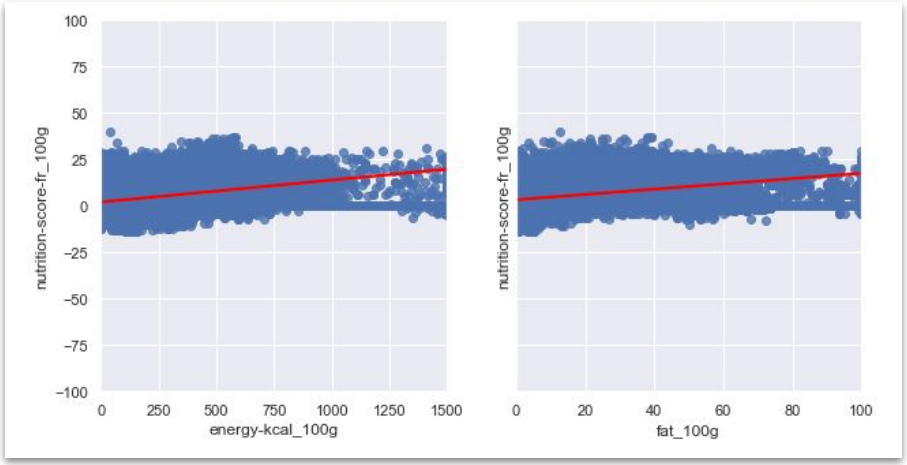


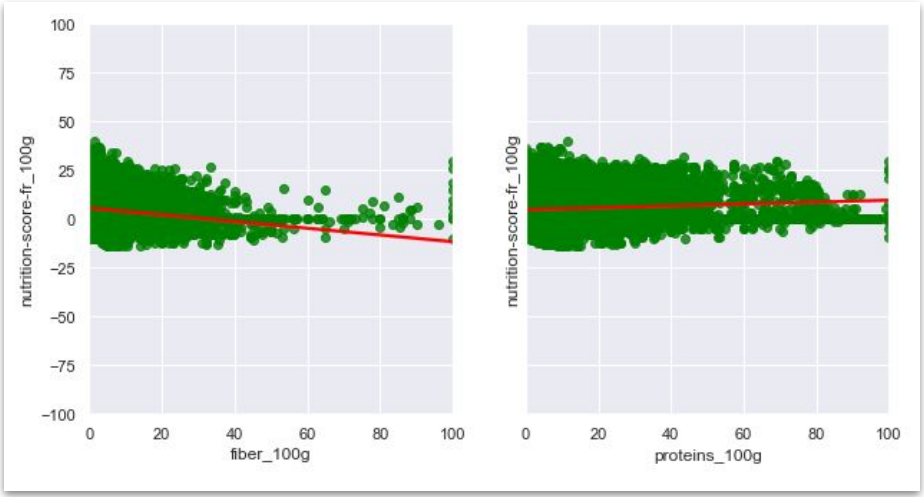
Matrice de corrélation

- Nous donne une bonne intuition concernant des dépendances entre variables
- 5 variables semblent impacter fortement le nutriscore: “energy_kcal”, “fat”, “saturated_fat”, “carbohydrates”, et “sugars”
- Proteins, fiber et sodium semblent moins importants

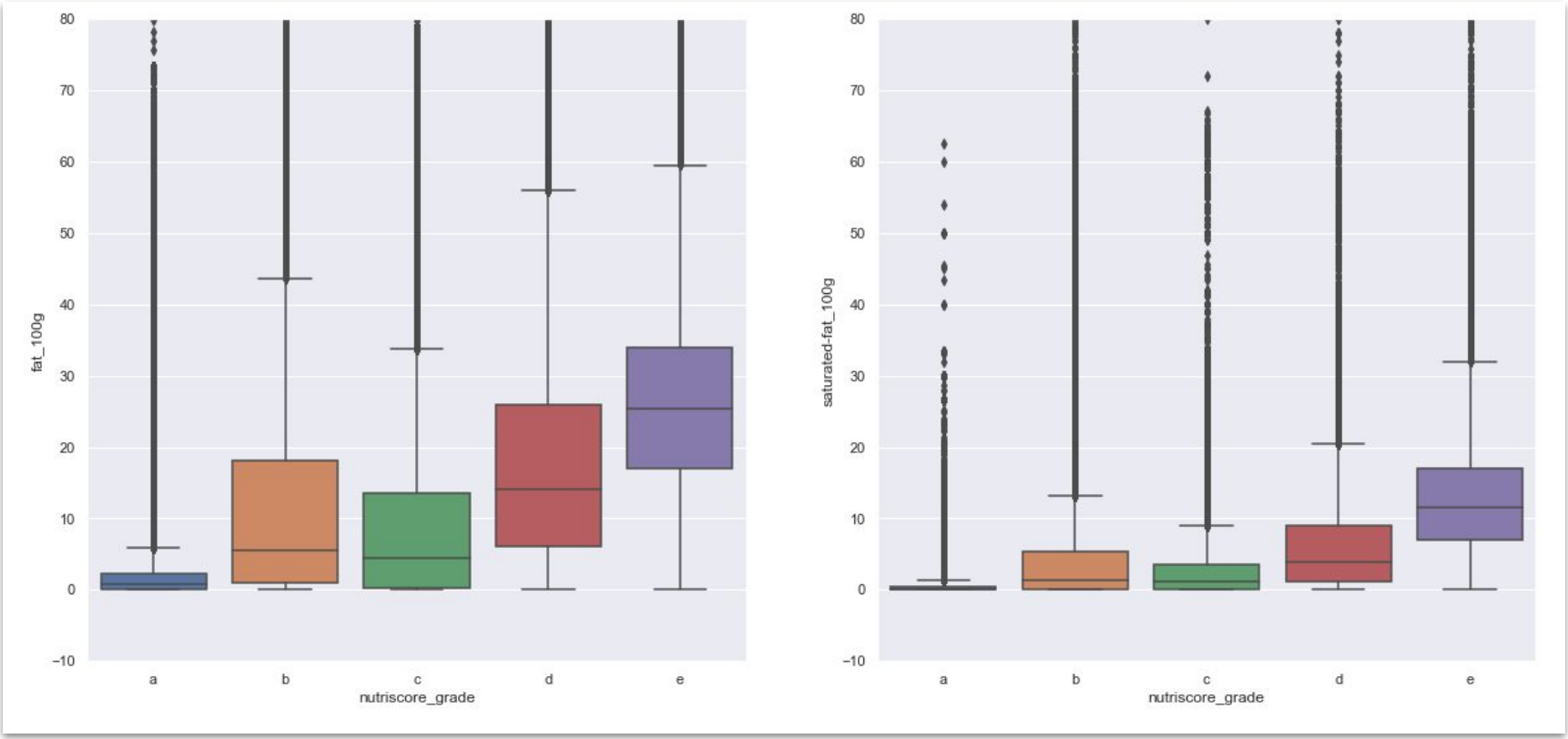


Analyse Bivariée: les scatterplots sont peu “tranchants”

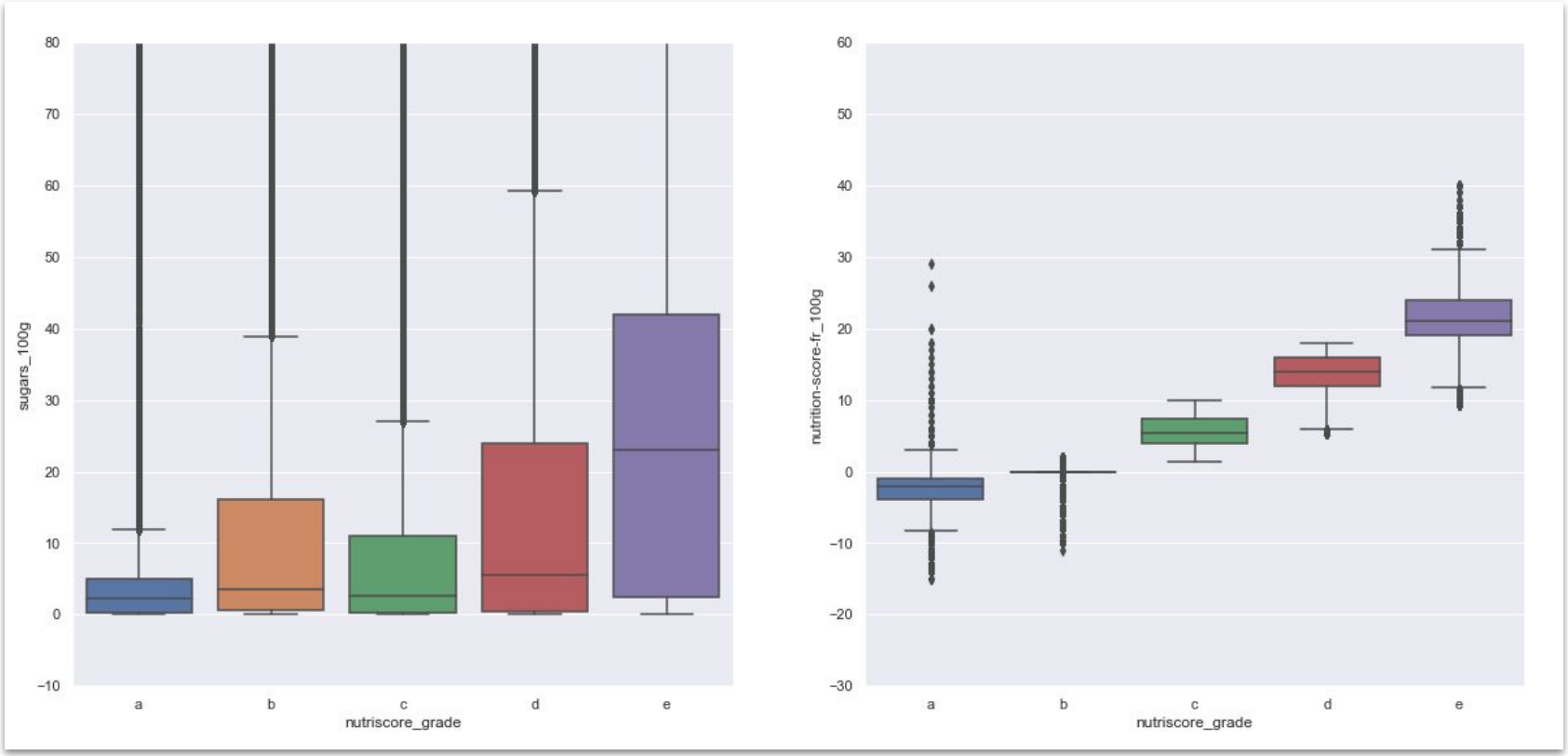




Analyse Bivariée: les Boxplots confirment nos intuitions (1/2)



Analyse Bivariée: les Boxplots confirment nos intuitions (2/2)



Conclusions intermédiaires:

(i) Plus “*energy_kcal*”, “*fat*”, “*saturated_fat*”, “*carbohydrates*”, et “*sugars*” sont élévés, plus le nutrigrade est mauvais

(ii) Plus le “*nutriscore*” est bon, plus le nutrigrade est bon

Analyse ANOVA: Confirme nos conclusions intermédiaires

- Nous voulons étudier la corrélation entre notre variable qualitative “Nutrigrade” et les variables quantitative définis dans notre variable “item_list”

- 3 variables explicatives semblent se détacher: “energy_kcal”, “fat”, “saturated_fat” (Eta² >=0.1)

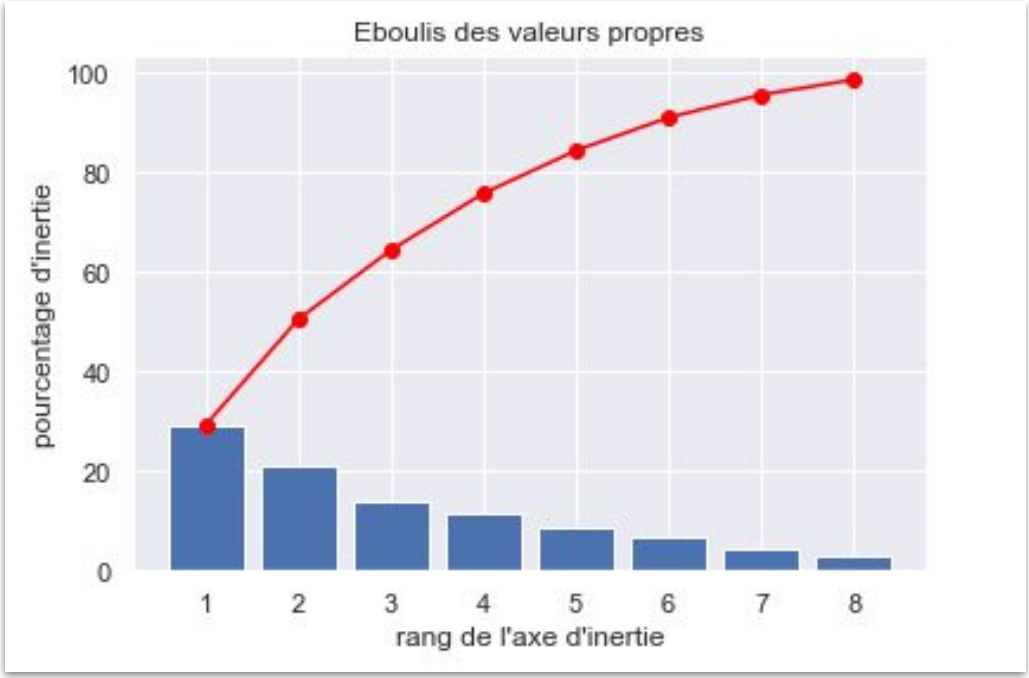
[25]

```
#ANOVA method
def eta_squared(x,y):
    moyenne_y = y.mean()
    classes = []
    for classe in x.unique():
        yi_classe = y[x==classe]
        classes.append({'ni': len(yi_classe),
                        'moyenne_classe': yi_classe.mean()})
    SCT = sum([(yj-moyenne_y)**2 for yj in y])
    SCE = sum([c['ni']*(c['moyenne_classe']-moyenne_y)**2 for c in classes])
    return SCE/SCT

item_list = ['energy-kcal_100g','fat_100g', 'saturated-fat_100g','sugars_100g','fiber_100g','proteins_100g']

for item in item_list:
  print(eta_squared(data_clean['nutriscore_grade'],data_clean[item]))
```

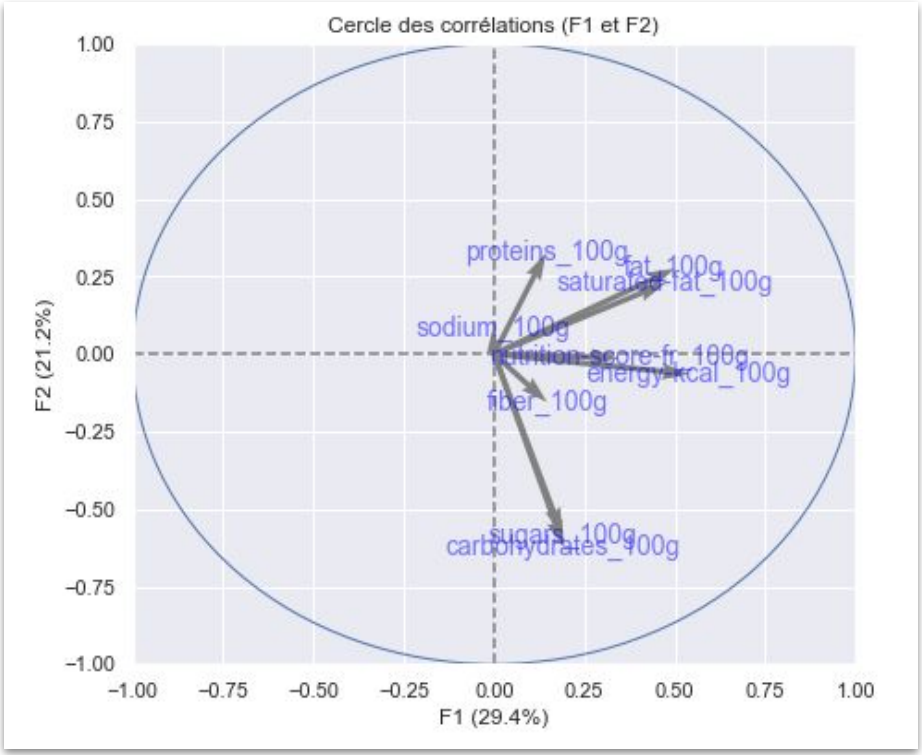
```
0.10022598442634192
0.10074471974003048
0.1632578746355588
0.06077799496265686
0.010183762623987251
0.00885793319826313
```

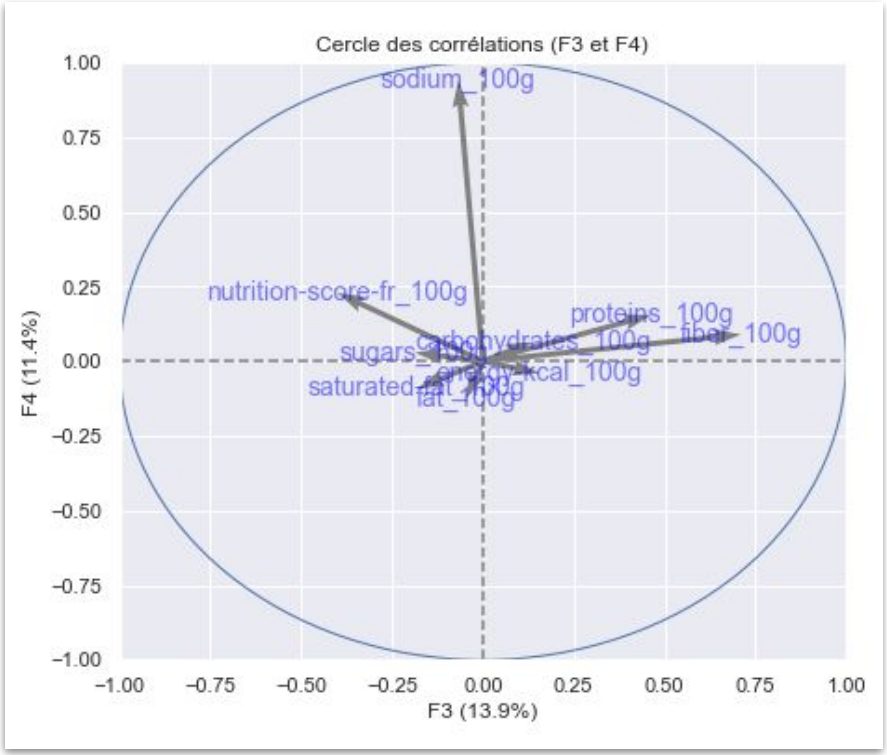
Analyse ACP (1/2): F1 et F2 sont les axes majeurs

- Le premier et le deuxième sont les axes qui conservent le plus d'inertie: à eux ils conservent plus de 50% de l'inertie
- Nous ferons les cercles de corrélation uniquement sur les 4 premiers axes (75% de l'inertie)



Analyse ACP (2/2): Des résultats cohérents nos conclusions





Conclusions finales:

(i) Les conclusions intermédiaires sont validées

(ii) Les cercles de corrélations mettent en avant des interdépendances entre nos variables

(iii) Le nutrigrade est avant tout déterminé par la présence de facteurs “pénalisants” (ceux qui dégradent la note)

Pour aller plus loin

Nous avons maintenant tous les éléments en main pour évaluer la nutrition de notre utilisateur.

La note journalière donnée par notre application, peut être obtenu de plusieurs manières. Une approche simple avec le mode peut être une approche facile et claire à implémenter.

Une bonne alimentation est la clé de la santé! Vive les aliments et boissons nutrigrade A!

=

Merci de votre attention!