Projet 3 : Concevez une application au service de la santé publique

Date : 09/12/2019

Version 2

Jury : Walid AYADI

Sommaire

1. **Idée d'application**
2. **Nettoyage effectué**
3. **Analyse exploratoire**
4. **Faits pertinents pour l'application**
5. **Synthèse**

1. Idée d'application

Santé publique France

- Indicateur de nutriscore pour un utilisateur qui n'aurait que quelques informations élémentaires sur le produit (jeu de données réduit)

2. Nettoyage effectué - fonctions

Découpage du processus de nettoyage

- Contrôle des colonnes
- 9 fonctions de nettoyage particulières
- Une fonction générale appliquant toutes les fonctions de nettoyage
- Capture d'exeptions via try/except
- Sauvegarde d'un fichier nettoyé

2. Nettoyage effectué - détail

- **Correction des types / format des dates**

```
if column[-2:] == '_t':
    new_column = column[:-2]
    dataframe[new_column] = pd.to_datetime(dataframe[column],
                                           unit='s')
    dataframe = dataframe.drop(column, axis=1)
```

- **Traitement des colonnes tags :** mapping

```
#traces_tags
mapping = {'nuts' : 'arachides',
       'milk' : 'lait',
       'gluten' : 'gluten',
       'soybeans' : 'graines de soja',
       'peanuts' : 'arachides',
      'eggs' : 'oeufs'}
dataframe['traces_tags'] = dataframe['traces_tags'].apply(categorize, args=[mapping])
dataframe['traces_tags'] = dataframe['traces_tags'].astype('category')
print(dataframe['traces_tags'].unique())
```

2. Nettoyage effectué - détail

- **Pays d'origine**
  - France uniquement
  - Suppression nutriscore-UK
- **Suppression des informations en doublon**
- **Titres des colonnes**

```
for column in columns:
    if column[0] == '-':
        column = column[1:]
```

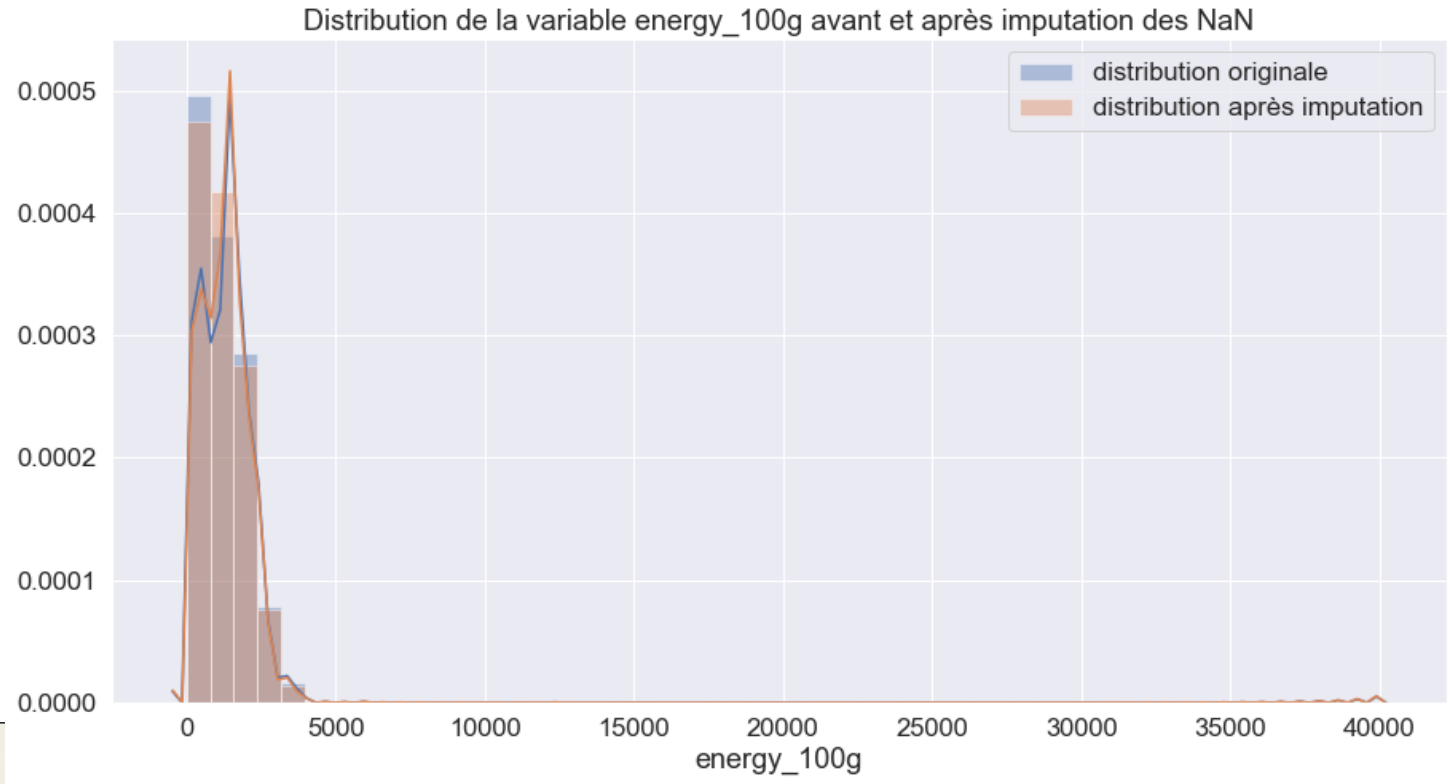
2. Nettoyage effectué - détail

- **Etude uni/multi-variée des outliers**
  - Outliers sur 1 dimension (1% extrême)
  - Outliers sur plusieurs dimensions (distance de Minkowski)

```
#outliers éloignés par rapport à leurs voisins
numeric_data = dataframe.select_dtypes(['int32', 'float64']).copy().dropna()
kdt = KDTree(numeric_data, leaf_size = 40, metric='minkowski')

dist, ind = kdt.query(numeric_data, k=3, return_distance=True)
numeric_data['3N_distance'] = np.sum(dist, axis=1)
numeric_data = numeric_data[numeric_data['3N_distance'] < numeric_data['3N_distance'].quantile(0.99)]
index_to_drop = numeric_data.index.tolist()

return dataframe.drop(index_to_drop, axis=0)
```

2. Nettoyage effectué - détail

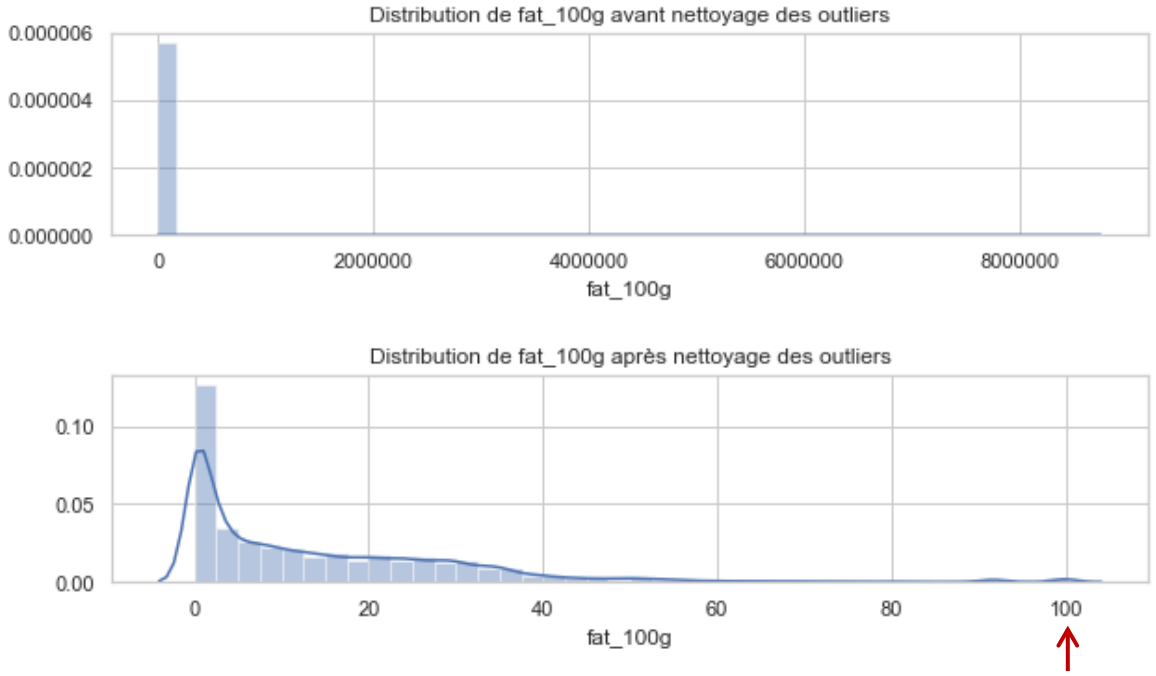
- **Traitement des NaN**
  - Suppression de colonne au delà d'un seuil préalablement fixé *(ajusté ici à 80 % maximum de taux de NaN)*
  - Imputation par la méthodes des K plus proches voisins



2. Nettoyage effectué - détail

- **Etude uni/multi-variée des outliers - Exemple**

Purge des outliers

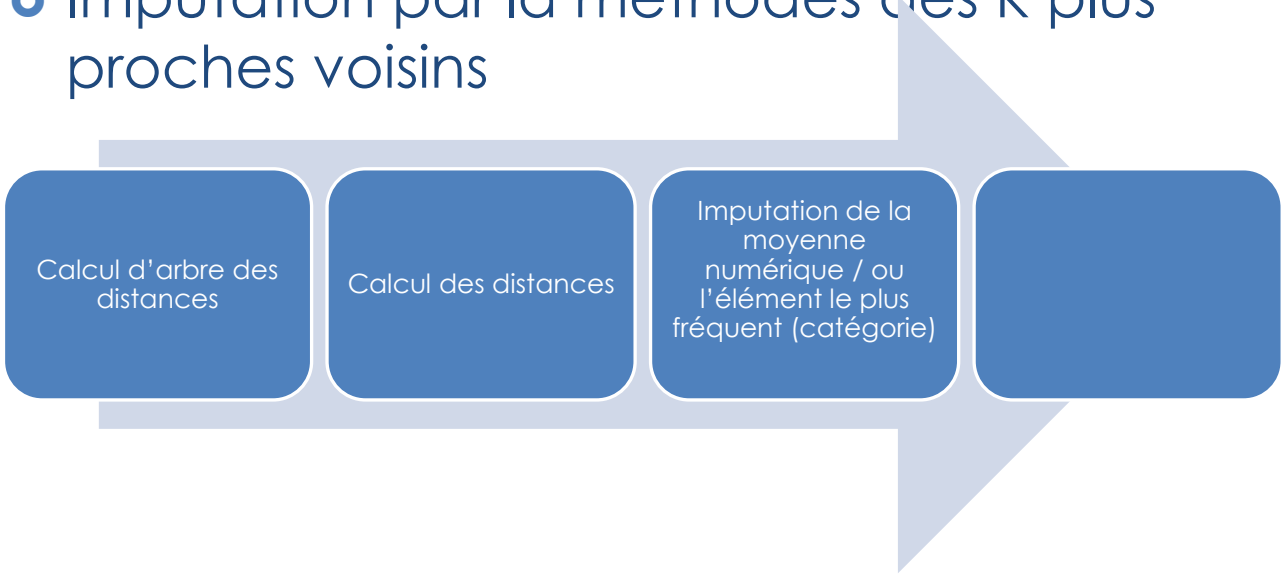


Valeurs comprises dans l'intervale [0 g ;100 g]

Inconvénient majeur : nombre d'outliers dépendant de la taille du jeu de données
*Alternative : outliers via distance à la moyenne supérieure à 2*std*

2. Nettoyage effectué - détail

- **Traitement des NaN**
  - Imputation par la méthodes des K plus proches voisins

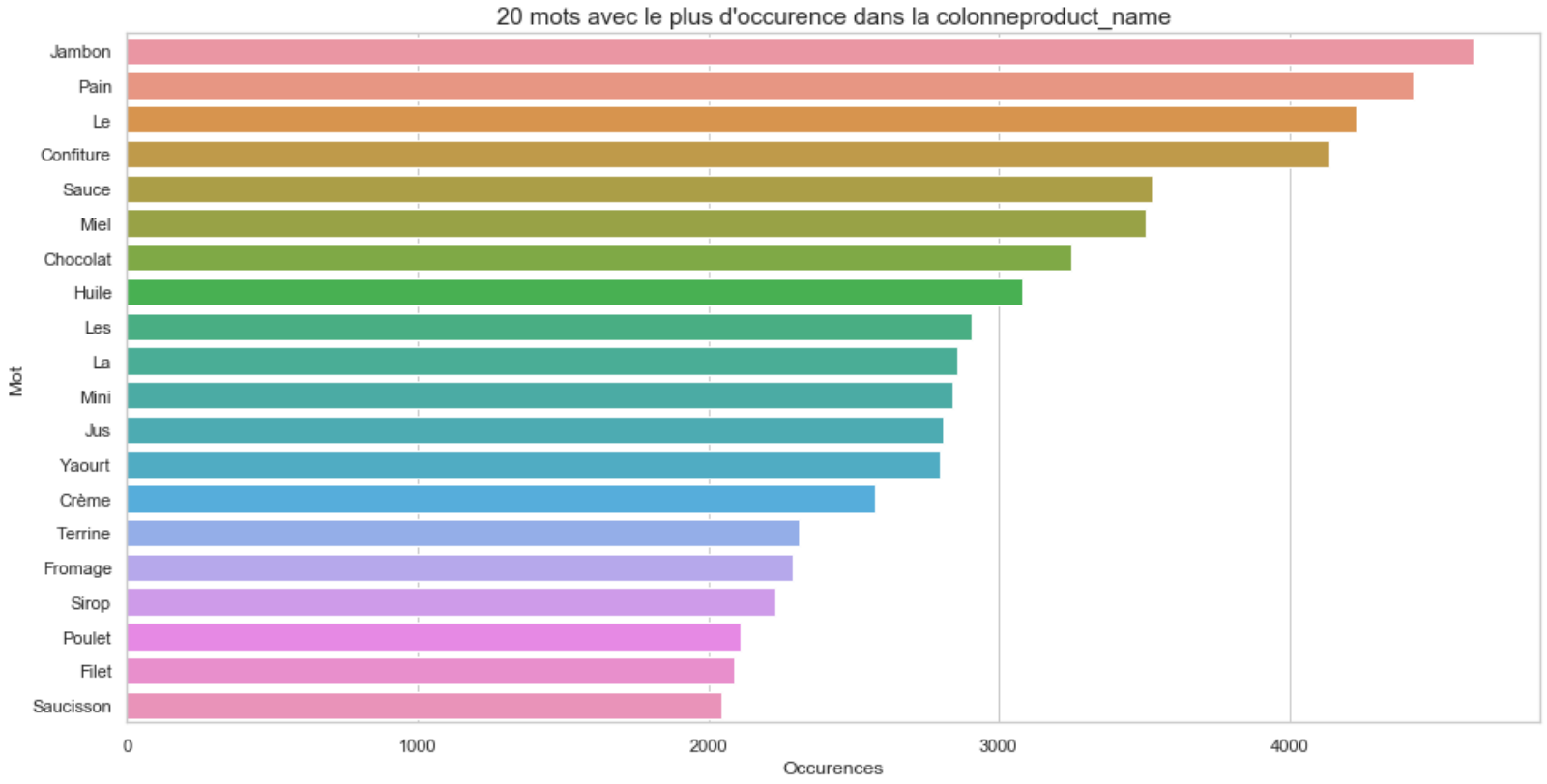


2. Nettoyage effectué – bilan avant/après

- 1 000 000 lignes réduites à 450 000 lignes
- 176 colonnes réduites à 48 colonnes
- 78 % de NaN réduit à 35 %
- Fichier .csv passé de 2 Go à 600 Mo

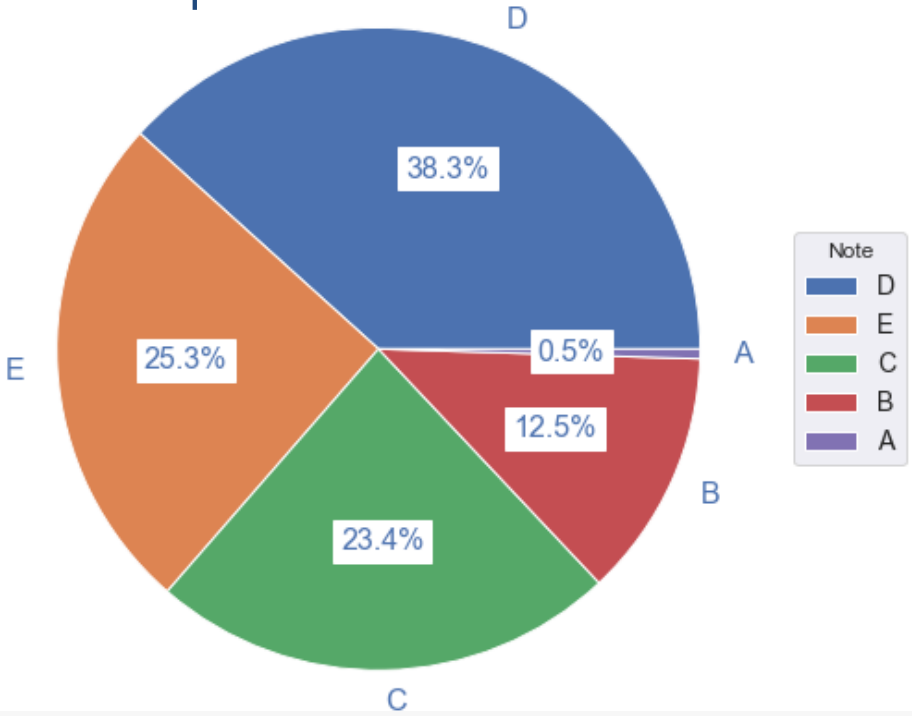
3. Analyse exploratoire – Connaissance des données :

- Occurence des mots dans les noms des produits



3. Analyse exploratoire – Connaissance des données

- Répartition des nutriscores

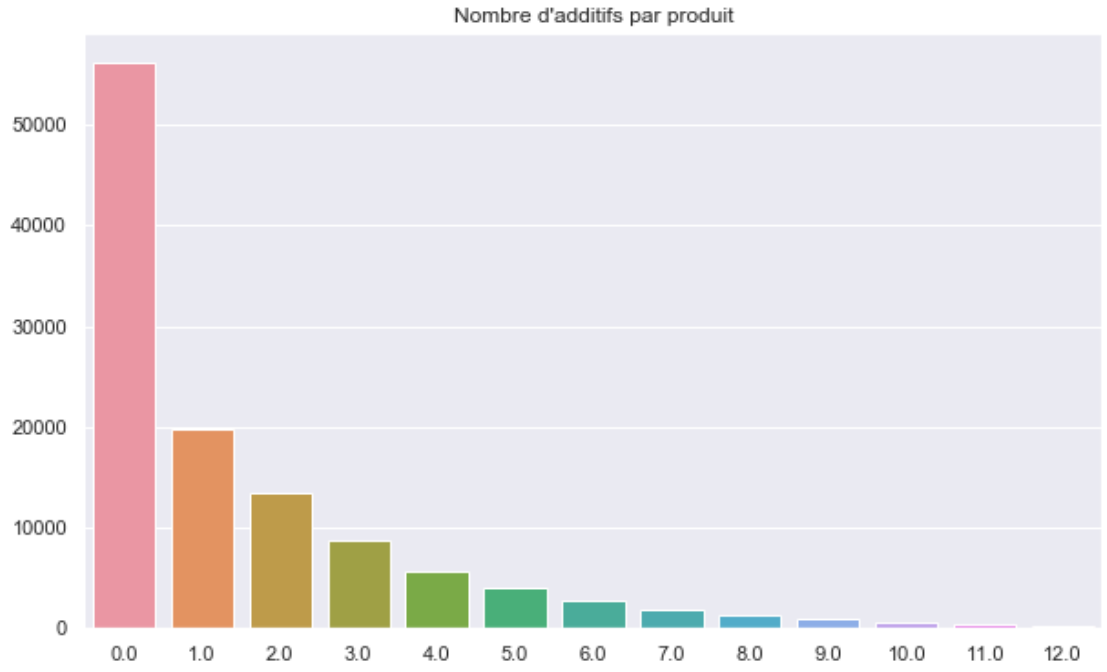



```
plt.title('Répartition des Nutriscores', size=20)
wedges, texts, autotexts = plt.pie(data.nutrition_grade_fr.value_counts().values,
        labels = data.nutrition_grade_fr.value_counts().index.str.upper(),
       autopct='%1.1f%%', textprops={'fontsize': 16,
                                     'color' : 'B',
                                     'backgroundcolor' : 'W',}
```

3. Analyse exploratoire – Connaissance des données

- Additifs

```
plt.title('Nombre d\'additifs par produit')
sns.barplot(x = data.additives_n.value_counts().index,
            y = data.additives_n.value_counts().values )
```



3. Analyse exploratoire – Connaissance des données

- Provenance des URL

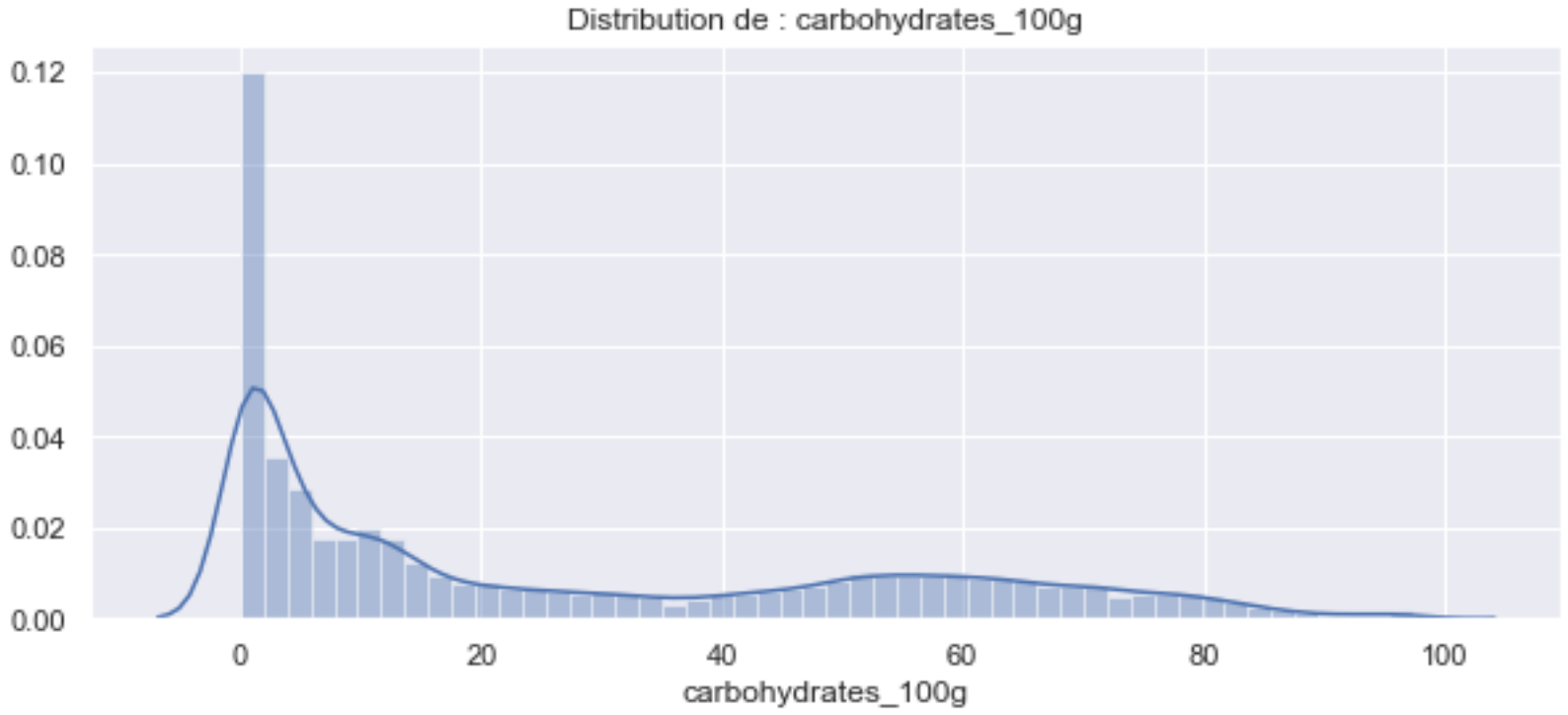
```
data.url.dropna().apply(lambda x: str(x).split('/')[2]).unique()
```

```
array(['world-en.openfoodfacts.org'], dtype=object)
```

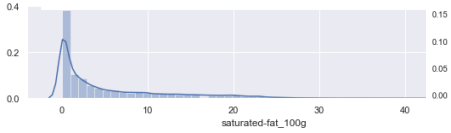
- Métriques des données numériques

	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g
count	328361.000000	325771.000000	326860.000000	325555.000000
mean	1153.429024	14.318785	5.412920	26.680424
std	768.761300	16.787356	6.899078	27.335283
min	0.000000	0.000000	0.000000	0.000000
25%	498.000000	1.100000	0.240000	2.200000
50%	1117.000000	8.800000	2.200000	13.710000
75%	1674.000000	23.000000	8.500000	52.000000
max	3766.000000	100.000000	47.200000	97.300000

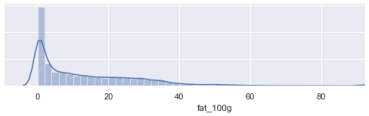
3. Analyse exploratoire – Analyse univariée : Distributions



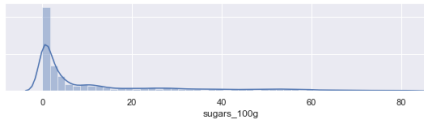
Saturated-fat_100g



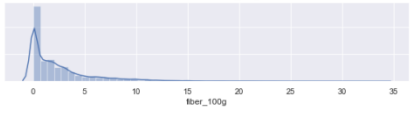
Fat_100g



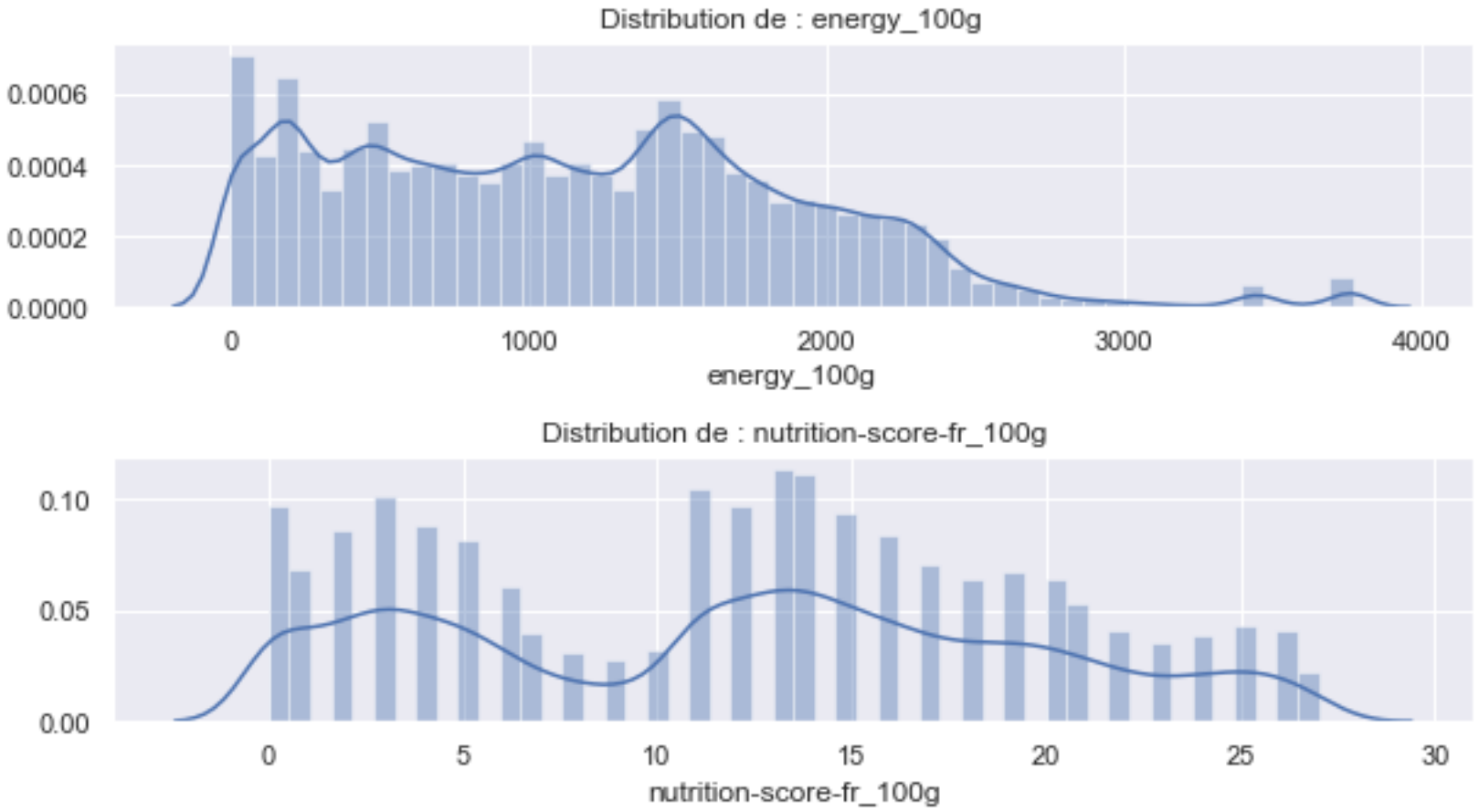
Sugars_100g



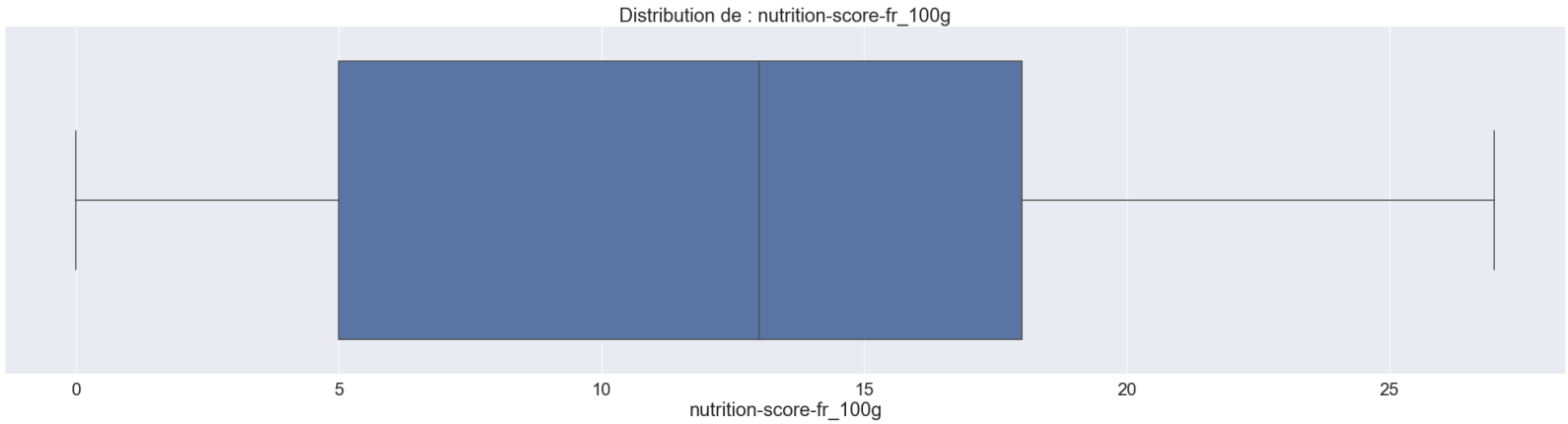
Fiber_100g

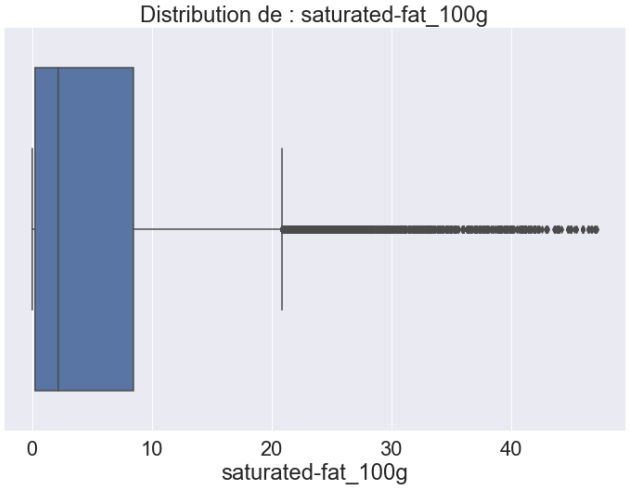


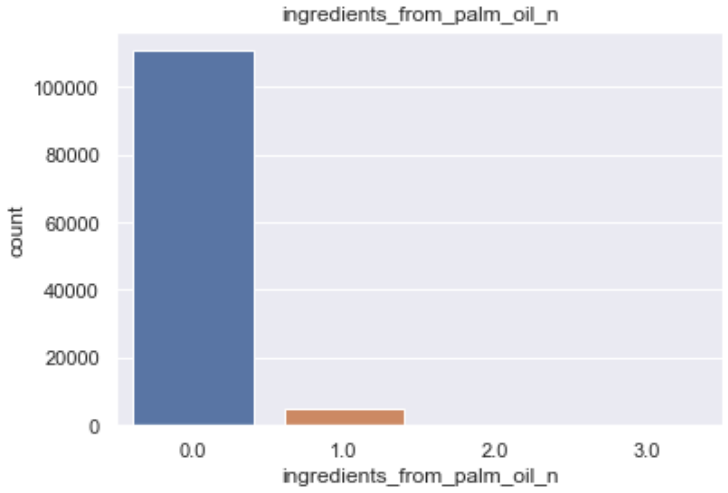
3. Analyse exploratoire – Analyse univariée : Distributions



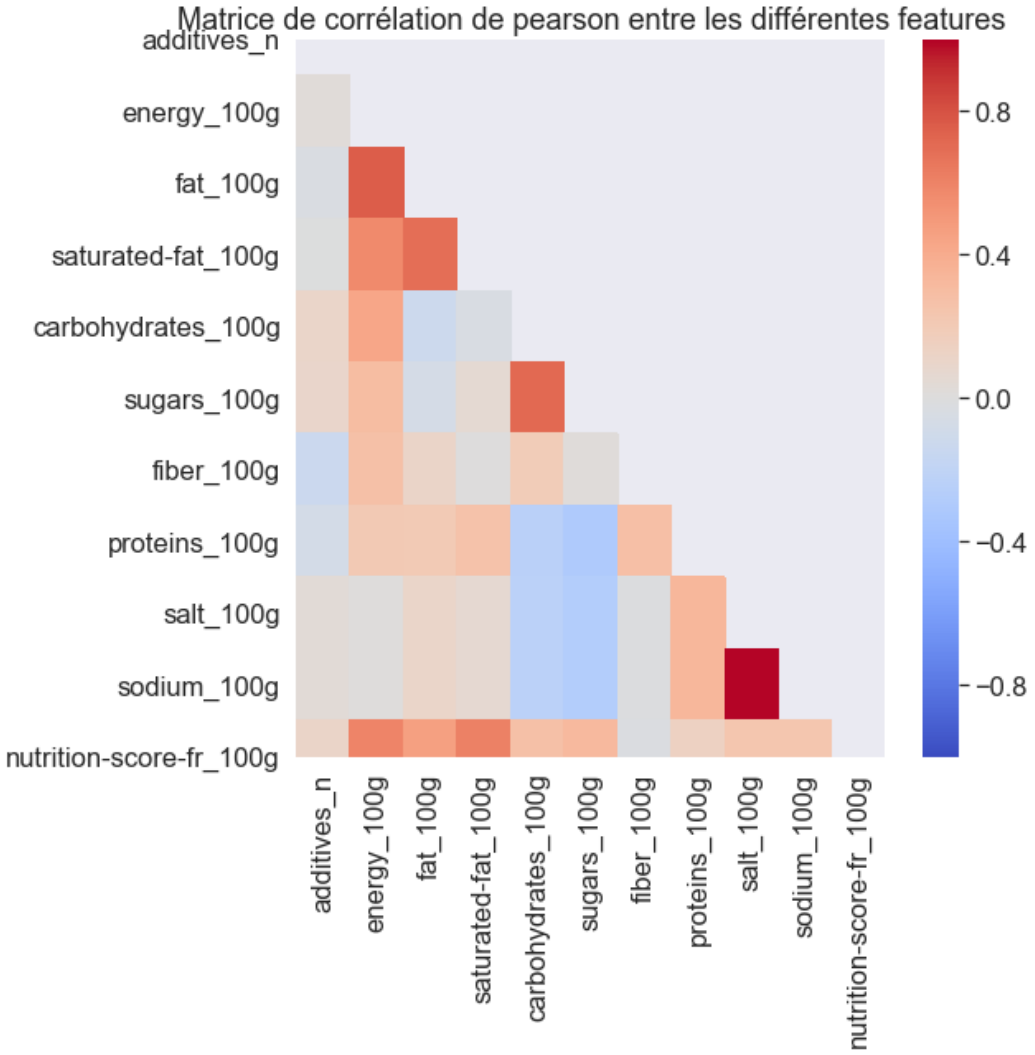
3. Analyse exploratoire – Analyse univariée : Exemple





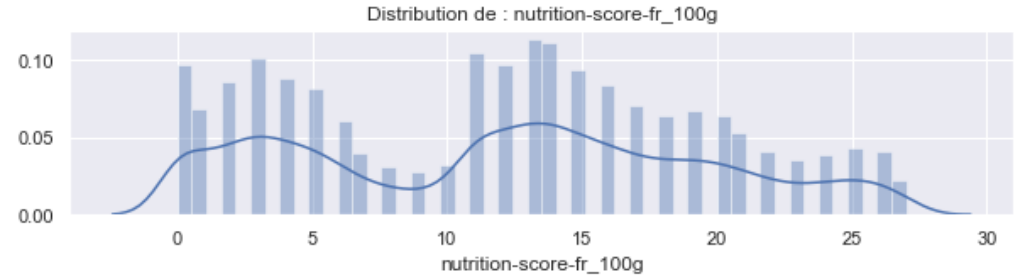


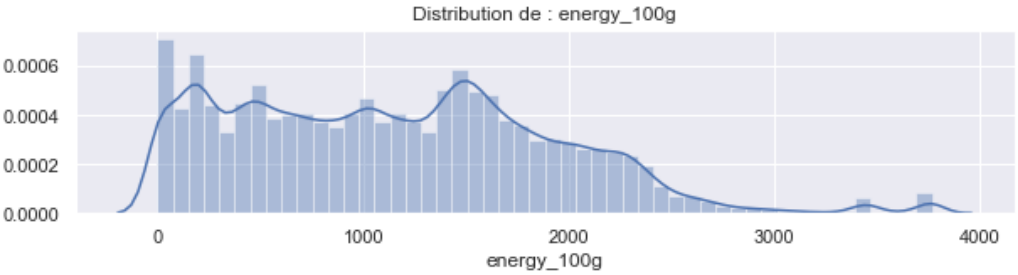
3. Analyse multivariée - corrélations

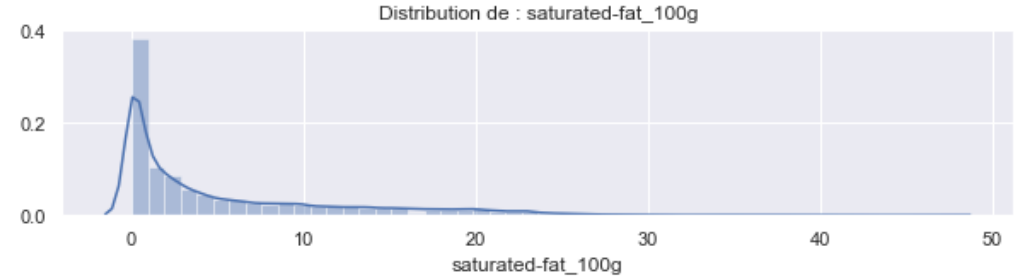


- **additives_n :** pas de correlation remarquable
- **energy_100g :** forte corrélation avec :
  - fat_100g
  - saturated-fat_100g
  - carbohydrates_100g
  - nutrition-score-fr_100g
- **fat_100g** et saturated-fat_100g fortement corrélés
- **sugars_100g** : forte correlation avec carbohydrates_100g
- **sodium_100g** correlation très forte avec salt_100g
- **nutrition-score-fr**_100g : forte corrélation avec:
  - energy_100g
  - saturated_fat_100g

3. Analyse multivariée - corrélations







- **additives_n :** pas de correlation remarquable
- **energy_100g :** forte corerlation avec :
  - fat_100g
  - saturated-fat_100g
  - carbohydrates_100g
  - nutrition-score-fr_100g
- **fat_100g** et saturated-fat_100g fortement corrélés
- **sugars_100g** : forte correlation avec carbohydrates_100g
- **sodium_100g** correlation très forte avec salt_100g
- **nutrition-score-fr**_100g : forte corrélation avec:
  - energy_100g
  - saturated_fat_100g

3. Analyse multivariée – indépendance des variables

- Test du CHI 2 :
  - Catégorisation des variables discrètes
  - Création de tableaux de contingences

packaging_tags	autre	carton	conserve	plastique	verre
ingredients_from_palm_oil_tags					
E304	101	6	1	38	0
autre	362811	4310	2220	30713	11155
huile-de-palme	3030	250	14	1589	25

  - Application du test du KHI2

```
#print('tableau de contingence :\n', pd.crosstab(serie1.array, serie2.array))
tab_contingence = pd.crosstab(serie1.array, serie2.array)
stat_chi2, p, dof, expected_table = chi2_contingency(tab_contingence.values)
print('chi2 : {},\np : {},\ndof : {}\n'.format(stat_chi2, p, dof))
#print('tableau de contingence : \n', tab_contingence)
```

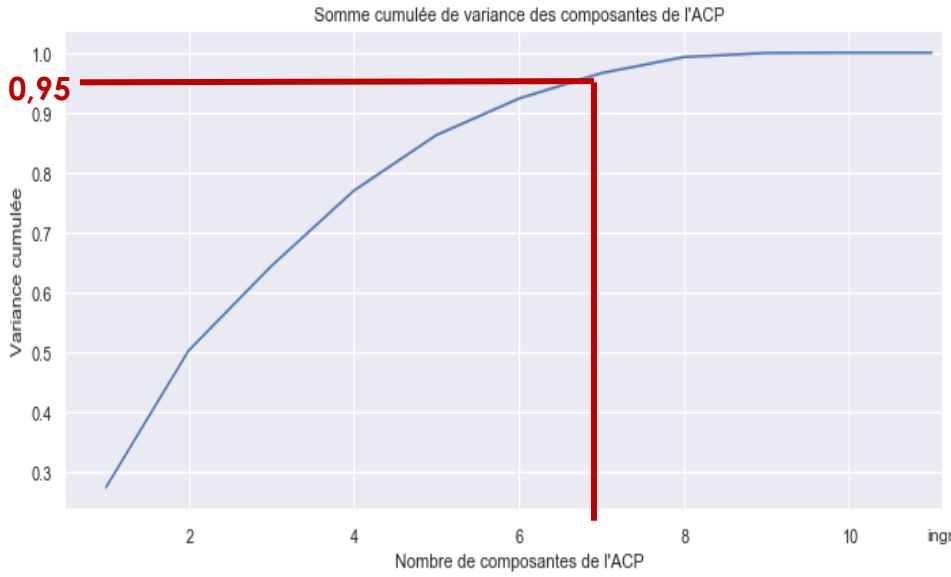
  - Conclusions

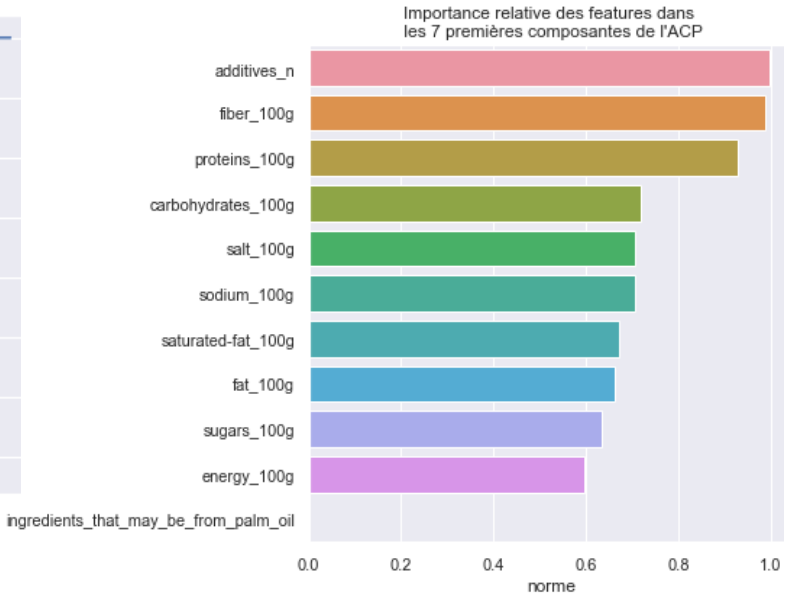
```
test d'indépendance nutriscore / fiber_100g
chi2 : 963.2662977794705,
p : 2.5328509986476587e-56,
dof : 361

Variables non indépendantes (H0 rejetée) car p = 2.5328509986476587e-56 <= alpha = 0.03
```

3. Réduction de dimension par Analyse par Composantes Principales

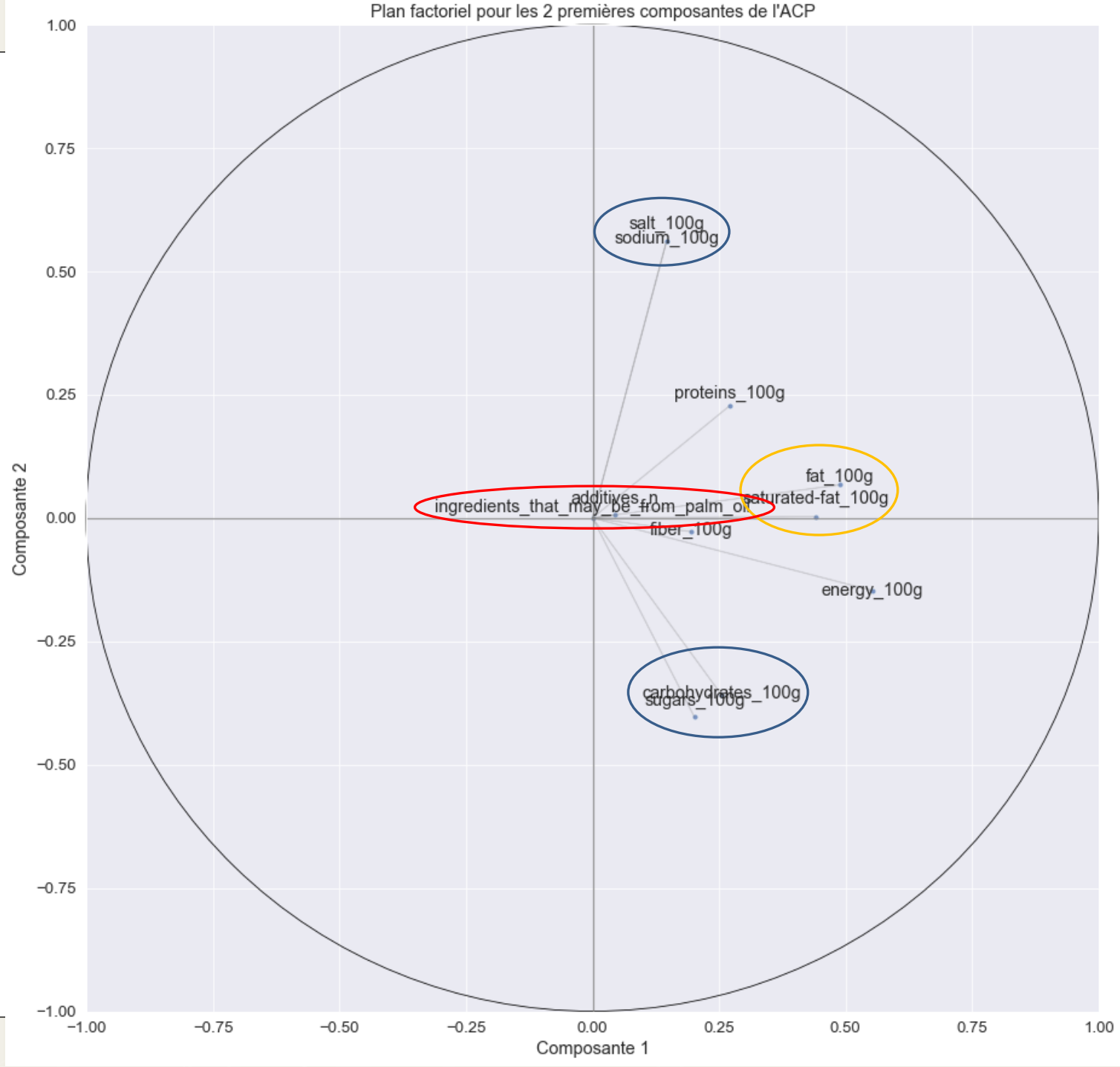
```
scaler = StandardScaler()
data_pca = scaler.fit_transform(data_pca)
```

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```





Réduction à 7 dimensions possible (initialement 12 features)


Plan factoriel pour les 2 premières composantes de l'ACP
Composante 1
Composante 2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
salt_100g
sodium_100g
proteins_100g
fat_100g
saturated-fat_100g
additives_n
ingredients_that_may_be_from_palm_oil
fiber_100g
energy_100g
carbohydrates_100g
sugars_100g

4. Faits pertinents pour l'application

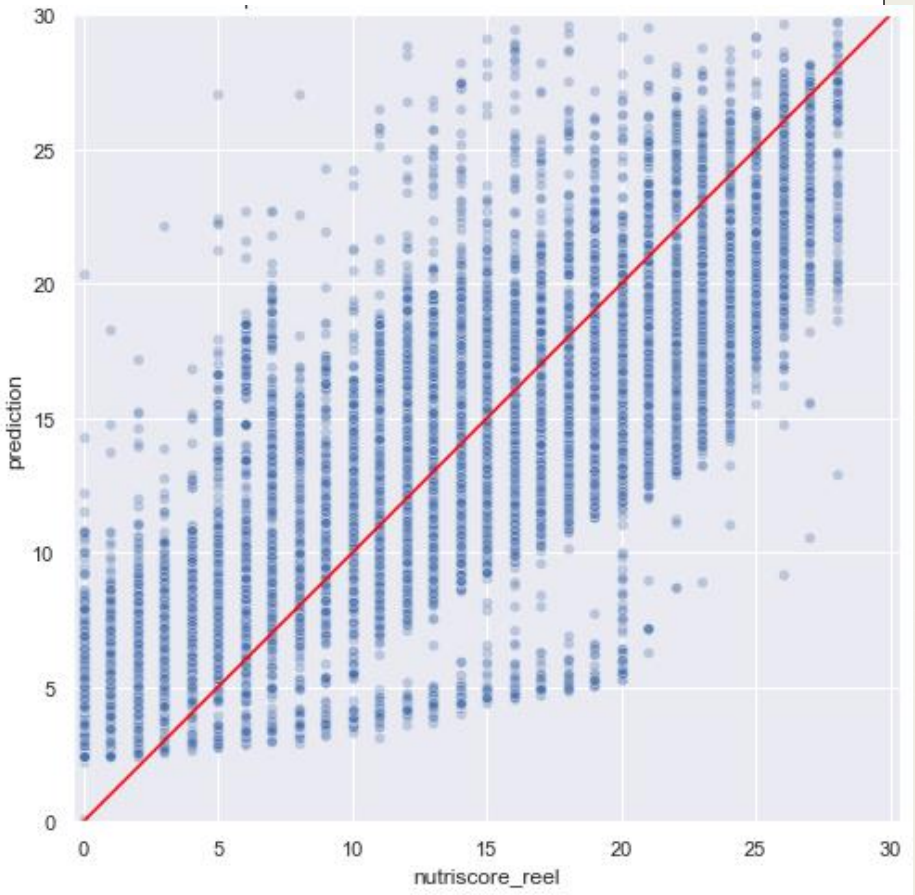
3 observations :

- Non indépendance des données
- Corrélation forte de certaines variables avec le nutriscore
- Résultats des premières régressions

4. Faits pertinents pour l'application - suite

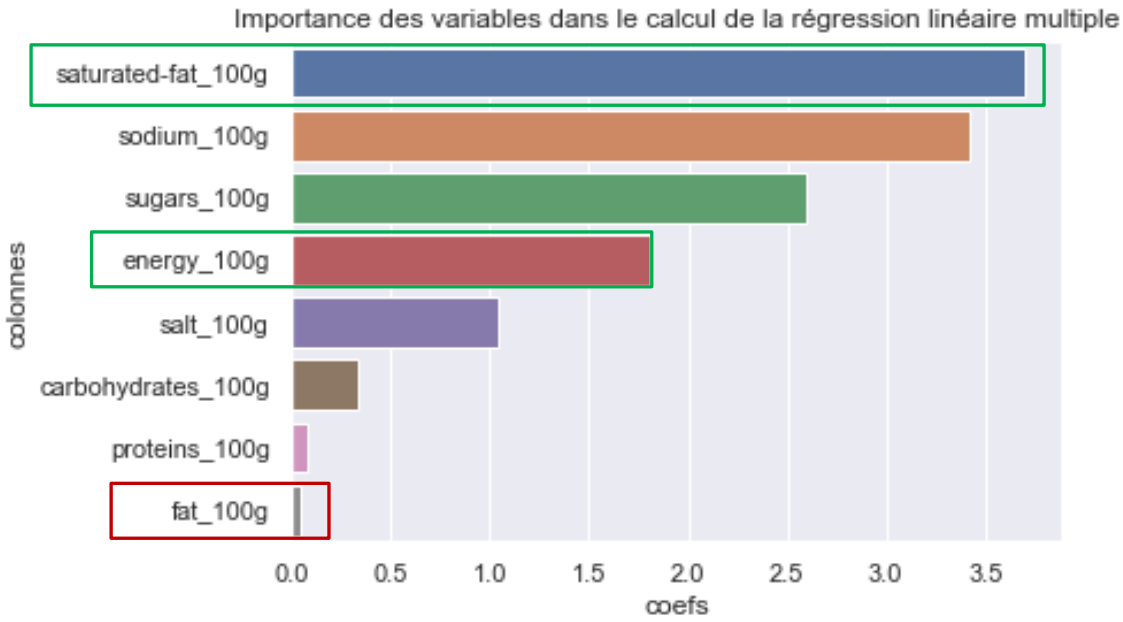
- Premières régressions
  - Jeu de données réduit à 9 variables
  - Séparation X/y
  - Séparation entrainement / test
  - Standard Scaler (μ = 0 / σ = 1)
  - Modèles : linéaire / Ridge / Lasso / Elasticnet
  - Mesure : RMSE
  - Optimisation des paramètres (Ridge / Lasso)
  - R^2 = 0,63 (Linéaire / Ridge / Lasso / Elasticnet)
  - RMSE ≈ 4,5
    (NB : nutriscore compris entre 0 et 26)

Comparaison des nutriscores estimés (y) et réels (x)



4. Faits pertinents pour l'application - suite

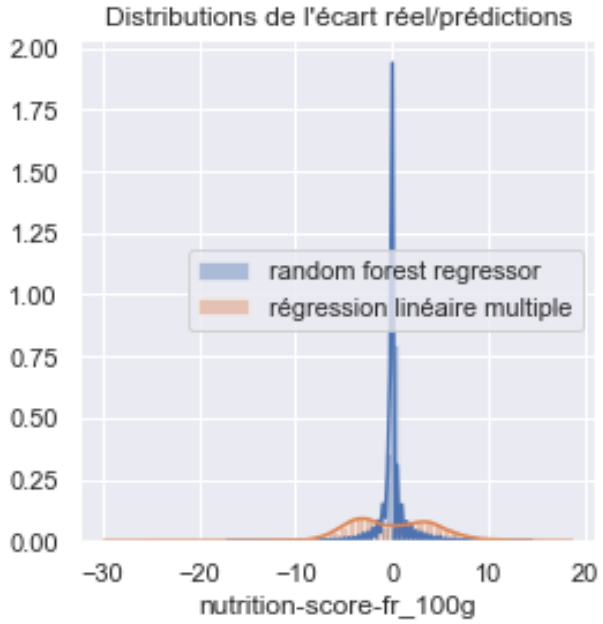
- Vérification : comparaison de l'importance des variables avec corrélation de Pearson

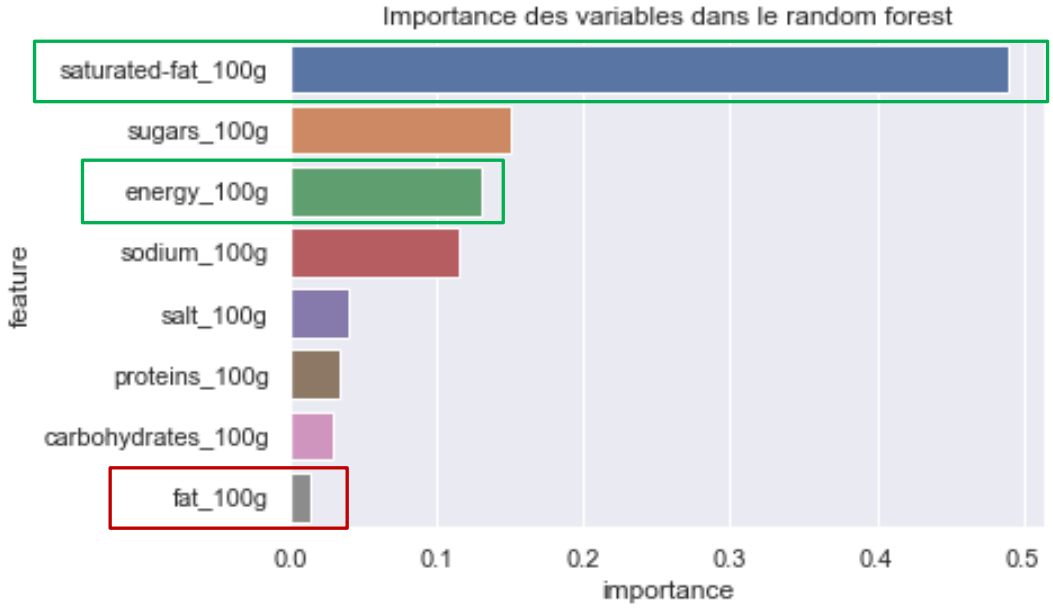


NB : La Forte corrélation de fat_100g avec nutrition-score ne se retrouve pas ici

4. Faits pertinents pour l'application - suite

- Application d'un algorithme d'ensemble (forêts aléatoires)
  - *R^2 = 0,94 sur le jeu de données de test*
  - *RMSE = 1,85 (progrès d'un facteur 2,5 x)*





5. Synthèse

- Forêts aléatoires concluantes sur un jeu de données réduit *(NB : Vrai algorithme du nutriscore non linéaire : cohérent)*
- Résultats cohérents avec les principes nutritionels (graisses saturées, sucres)
- Possibilité de proposer un algorithme qui donne une indication de nutriscore approché ($R^2 = 0,85$) avec 8 variables : faisabilité de l'application

Merci de votre attention