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

Date : 15/06/2023

Sommaire

Santé publique France

1. Idée d'application

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

  Calcul automatique de nutriscore

OPEN FOOD FACTS

NUTRI-SCORE A B C D E

2. Nettoyage effectué - fonctions

Découpage du processus de nettoyage

- Contrôle des colonnes
- Plusieurs fonctions de nettoyage particulières
- Une fonction générale appliquant toutes les fonctions de nettoyage
- Capture d'exceptions 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 nutrition-score-uk_100g
- **Suppression des informations en doublon**
- **Titres des colonnes**

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

2. Nettoyage effectué - détail

Outliers avec la méthode d'interquartile

La méthode des 1% extrêmes consiste à identifier les valeurs qui se trouvent en dehors de la plage des 1% des valeurs les plus extrêmes de l'ensemble de données. Pour cela, on peut calculer les seuils inférieur et supérieur en utilisant la formule suivante :

seuil inférieur = quantile(ensemble de données, 0.05)

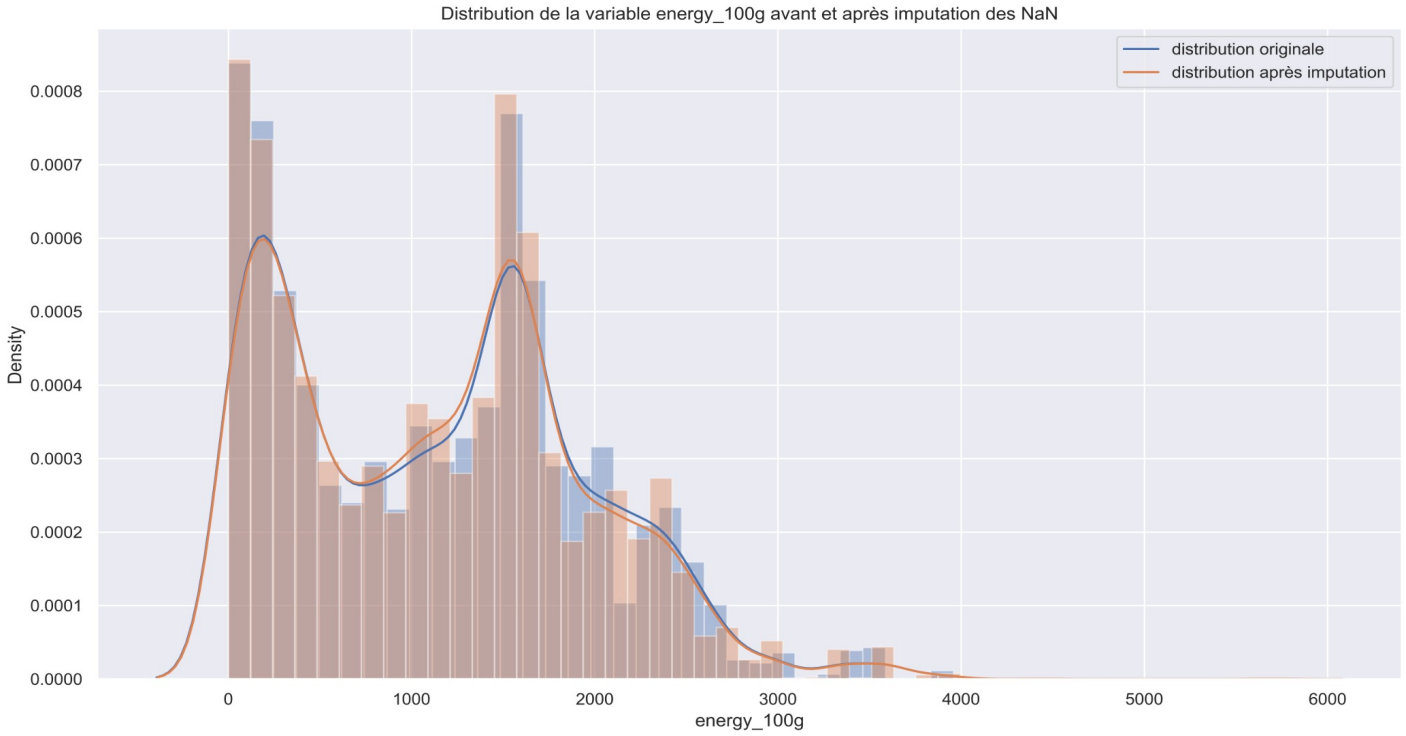
seuil supérieur = quantile(ensemble de données, 0.95)

```
def delete_outliers_interquartile(dataframe):
    '''Suppression des outliers avec la méthode de l'intervalle interquartile'''
    # Parcours des colonnes numériques de la dataframe
    for column in dataframe.select_dtypes(include = ['int8', 'float32','int32','float64']).columns.tolist() :
        # Calcul des quartiles
        q1 = dataframe[column].quantile(0.05)
        q3 = dataframe[column].quantile(0.95)
        # Calcul de l'écart interquartile
        iqr = q3 - q1
        # Calcul des bornes de l'intervalle interquartile
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        # Remplacement des valeurs en dehors de l'intervalle par NaN
        dataframe.loc[dataframe[column] < lower_bound, column] = np.nan
        dataframe.loc[dataframe[column] > upper_bound, column] = np.nan
        # Remplacement des valeurs négatives par 0
        dataframe.loc[dataframe[column] < 0, column] = 0
        # Remplacement des valeurs supérieures à 100 par 100
        dataframe.loc[dataframe[column] > 100, column] = 100
    # Retourne la dataframe modifiée
    return dataframe
```

2. Nettoyage effectué - détail

Traitement des NaN

- Suppression de colonne au delà d'un seuil préalablement fixé avec la fonction clean_nan()



2. Nettoyage effectué - détail

Traitement des NaN

Métodes KNN

```
from sklearn.impute import KNNImputer

def clean_dataframe_impute_knn(df):

#     return df_cleaned
    print('..... DEBUT De la fonction clean_dataframe_impute_knn  : \n.....')

    # Liste des colonnes avec des valeurs manquantes
    numeric_columns = df.select_dtypes(['int32','float64','int8','float32']).columns
    # Imputation des valeurs manquantes avec la stratégie "most_frequent"
    imputer = KNNImputer(n_neighbors=5)

    df[numeric_columns] = pd.DataFrame(imputer.fit_transform(df[numeric_columns]), columns=numeric_columns)

    print('..... Dans la fonction clean_dataframe_impute_knn df.shape : \n.....')
    print(df.shape)
    print('..... Dans la fonction clean_dataframe_impute_knn df.columns : \n.....')
    print(df.columns.tolist())
    print('..... Fin De la fonction clean_dataframe_impute_knn  : \n.....')
    return df
```

2. Nettoyage effectué - détail

Traitement des NaN

- Instruction pour remplacer les valeurs manquantes ou NaN par 0

```
# Remplacer les valeurs manquantes ou NaN par 0
df[columns_to_check] = df[columns_to_check].fillna(0)
```

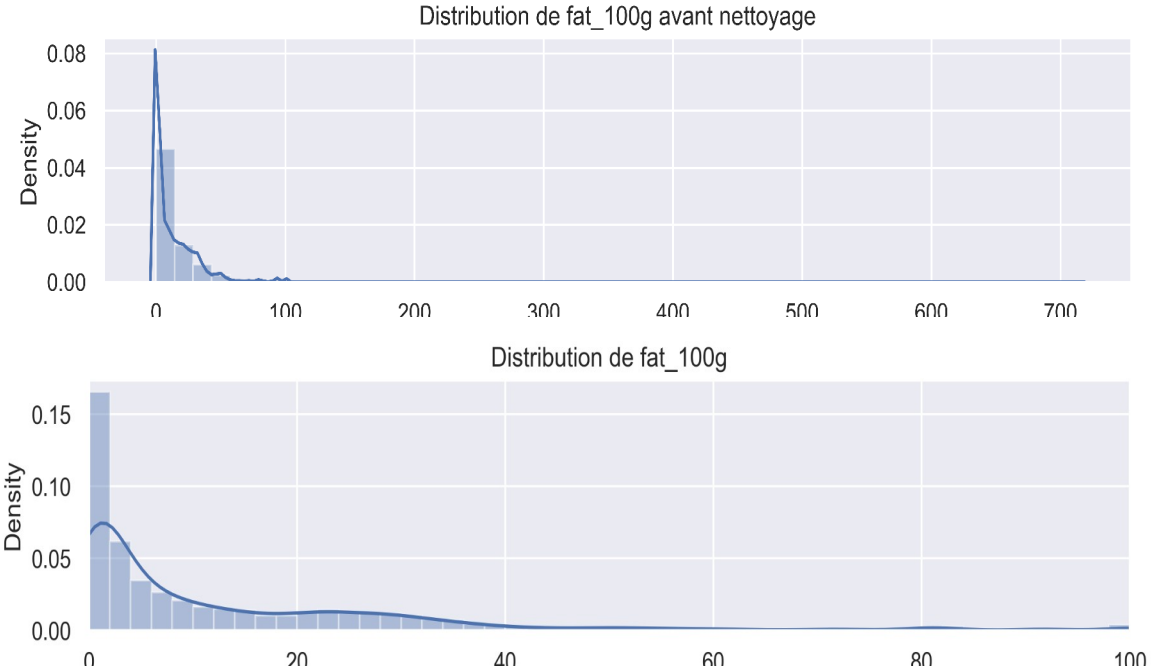
- Suppression des valeurs manquantes ou NaN par 0

```
print('.....nettoyage dropna')
try :
    # Suppression des lignes avec des valeurs manquantes
    dataframe = dataframe.dropna(axis=0)
except Exception as e:
    print(e)
    print('.....Erreur dropna')
```

2. Nettoyage effectué - détail

Etude uni/multi-variée des outliers - Exemple

Purge des outliers



Valeurs comprises dans l'intervalle [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 – bilan avant/après

- 320772 lignes réduites à 36906 lignes
- 165 colonnes réduites à 12 colonnes
- 75 % de NaN réduit à 0 %
- Fichier .csv passé de 1.7 Go à 193.6 Mo

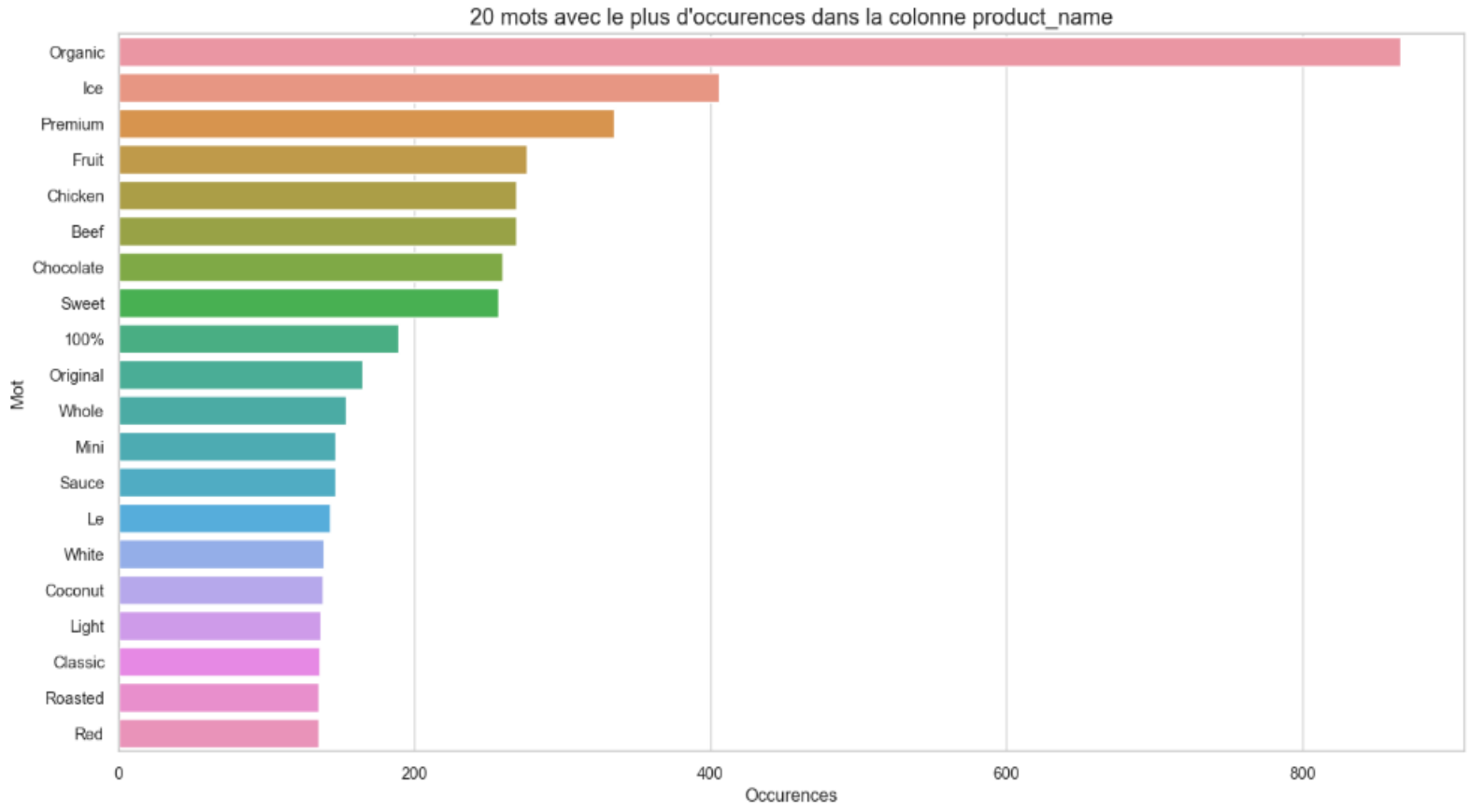
3. Analyse exploratoire – Connaissance des données :

Définition de certains champs :

- additives_n : nombre d'additifs alimentaires présents dans le produit.
- energy_100g : quantité d'énergie en kilojoules (kJ) pour 100g de produit.
- fat_100g : quantité de matières grasses en grammes (g) pour 100g de produit.
- saturated-fat_100g : quantité d'acides gras saturés en grammes (g) pour 100g de produit.
- carbohydrates_100g : quantité de glucides en grammes (g) pour 100g de produit.
- sugars_100g : quantité de sucres en grammes (g) pour 100g de produit.
- proteins_100g : quantité de protéines en grammes (g) pour 100g de produit.
- salt_100g : quantité de sel en grammes (g) pour 100g de produit.
- sodium_100g : quantité de sodium en grammes (g) pour 100g de produit.
- Le nutrition-score-fr_100g est un score nutritionnel qui a été développé pour évaluer la qualité nutritionnelle des aliments. Ce score prend en compte différents nutriments tels que les acides gras saturés, les sucres, le sodium, les fibres et les protéines.
- Le nutrition_grade_fr est une notation nutritionnelle qui va de A à E. Elle permet de classer les aliments en fonction de leur qualité nutritionnelle, A étant la meilleure note et E la moins bonne.
  Cette notation est basée sur le score nutritionnel mentionné ci-dessus.

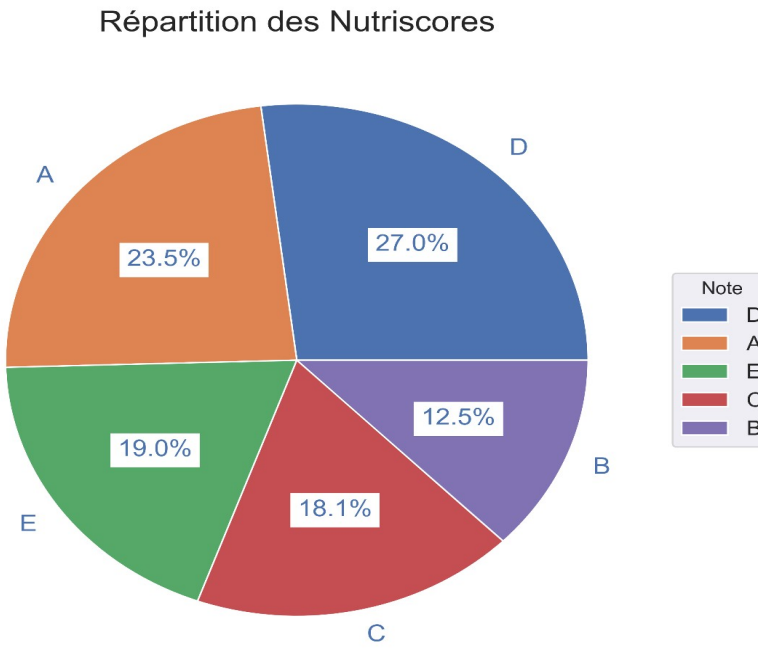
3. Analyse exploratoire – Connaissance des données :

- Occurrence des mots dans les noms des produits



3. Analyse exploratoire – Connaissance des données

Répartition des nutriscores

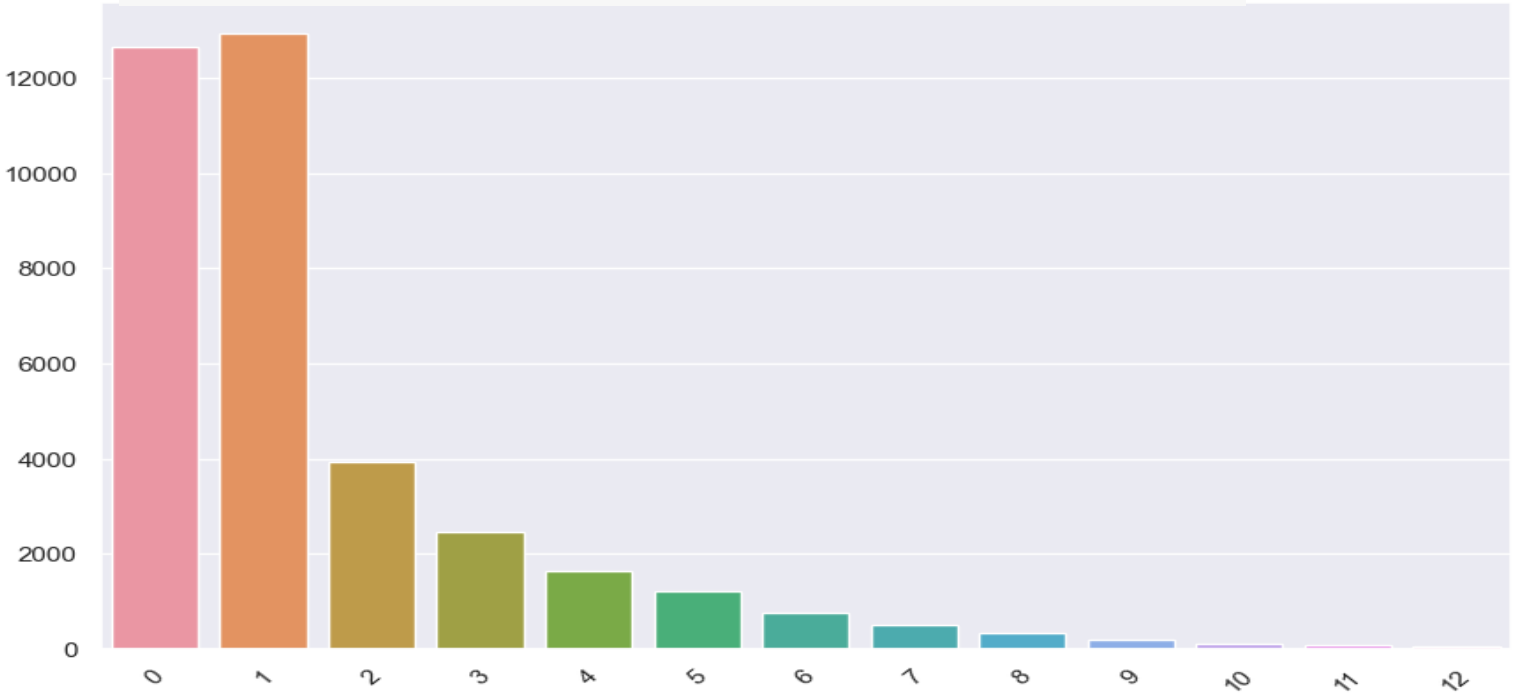



```
plt.title('Répartition des Nutriscores', size=20)
wedges, texts, autotexts = plt.pie(data.nutrition_grade_fr.value_counts().values,
        labels = data.nutrition_grade_fr.value_counts().index.str.upper(),
       autopct='%1.1f%%', textprops={'fontsize': 16,
                                    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

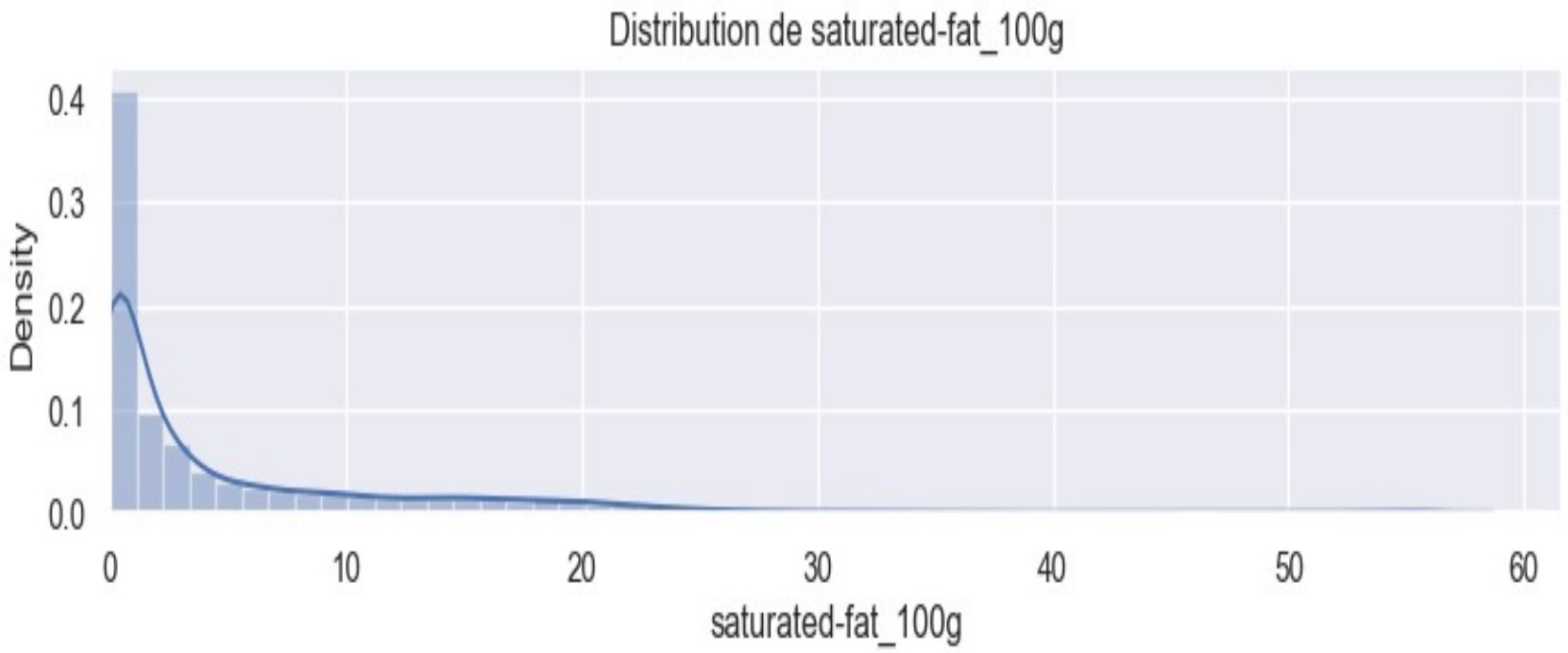
3. Analyse exploratoire – Connaissance des données

Additifs

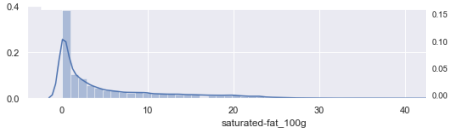
```
plt.title('Nombre d\'additifs par produit')
sns.barplot(x = data.additives_n.value_counts().index,
            y = data.additives_n.value_counts().values )
```



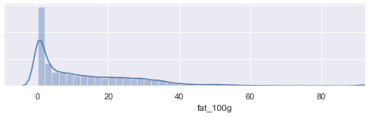
3. Analyse exploratoire – Analyse univariée : Distributions



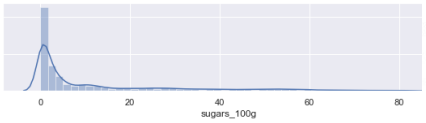
Saturated-fat_100g



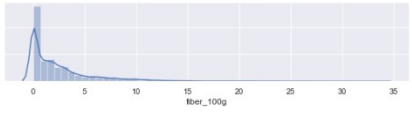
Fat_100g



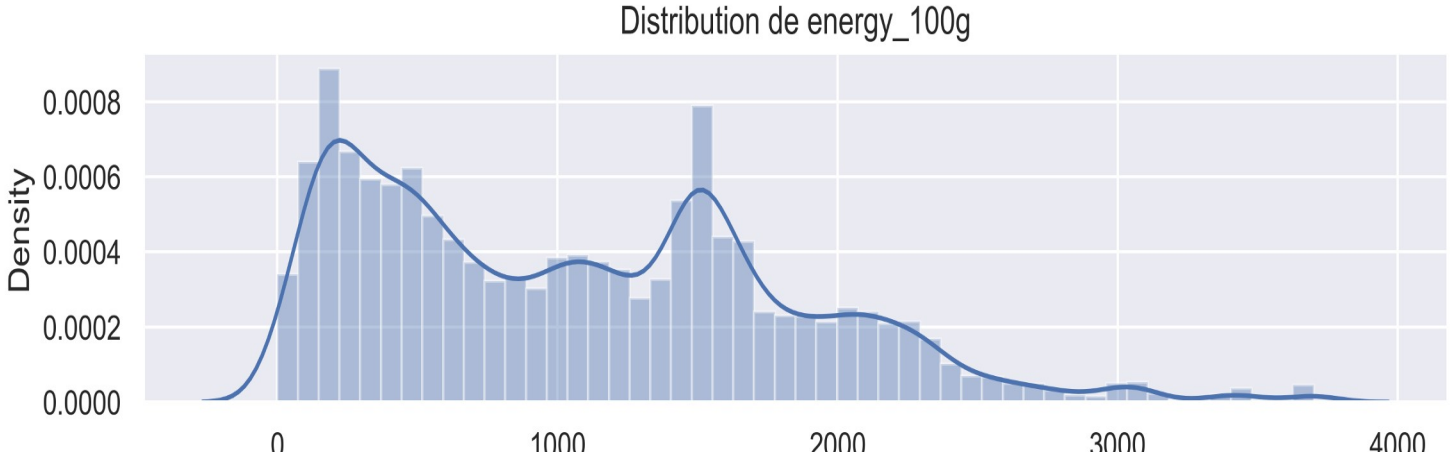
Sugars_100g

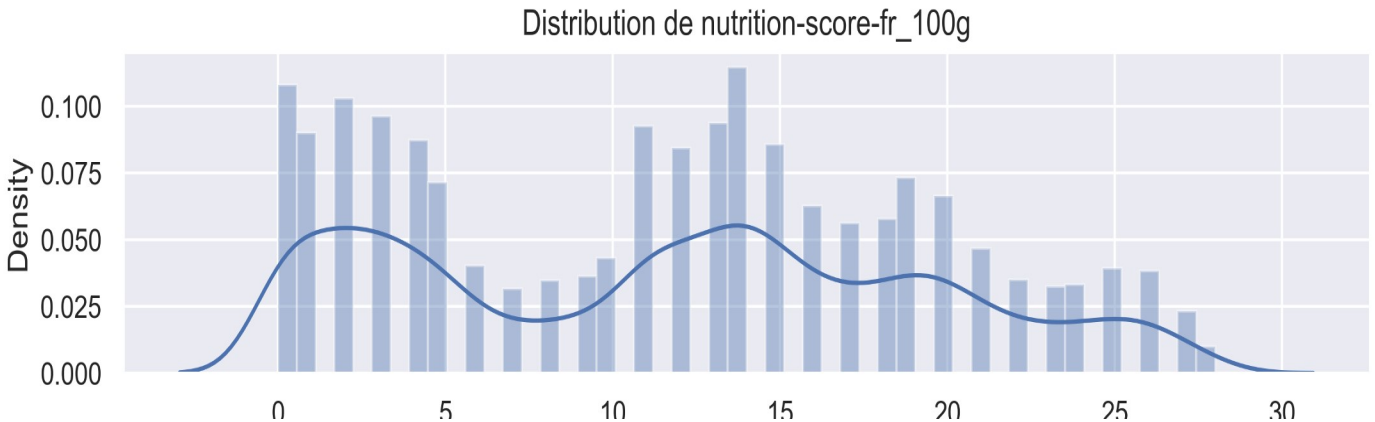


Fiber_100g

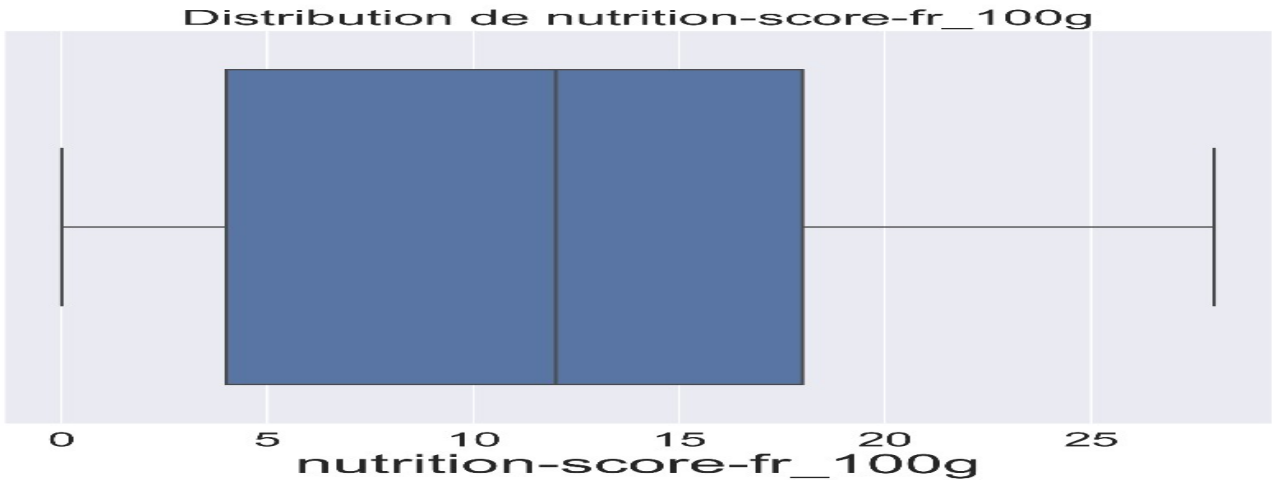


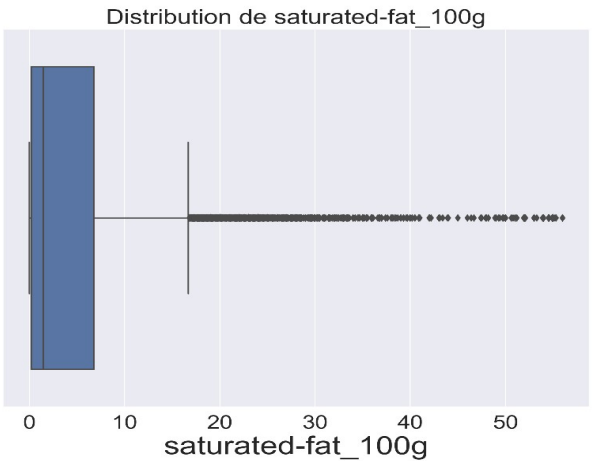
3. Analyse exploratoire – Analyse univariée : Distributions

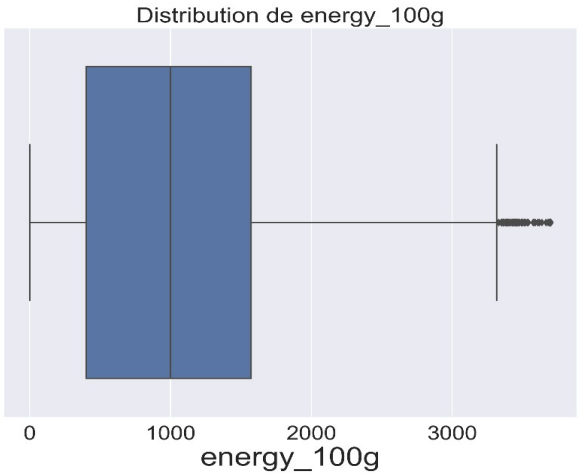




3. Analyse exploratoire – Analyse univariée : Exemple







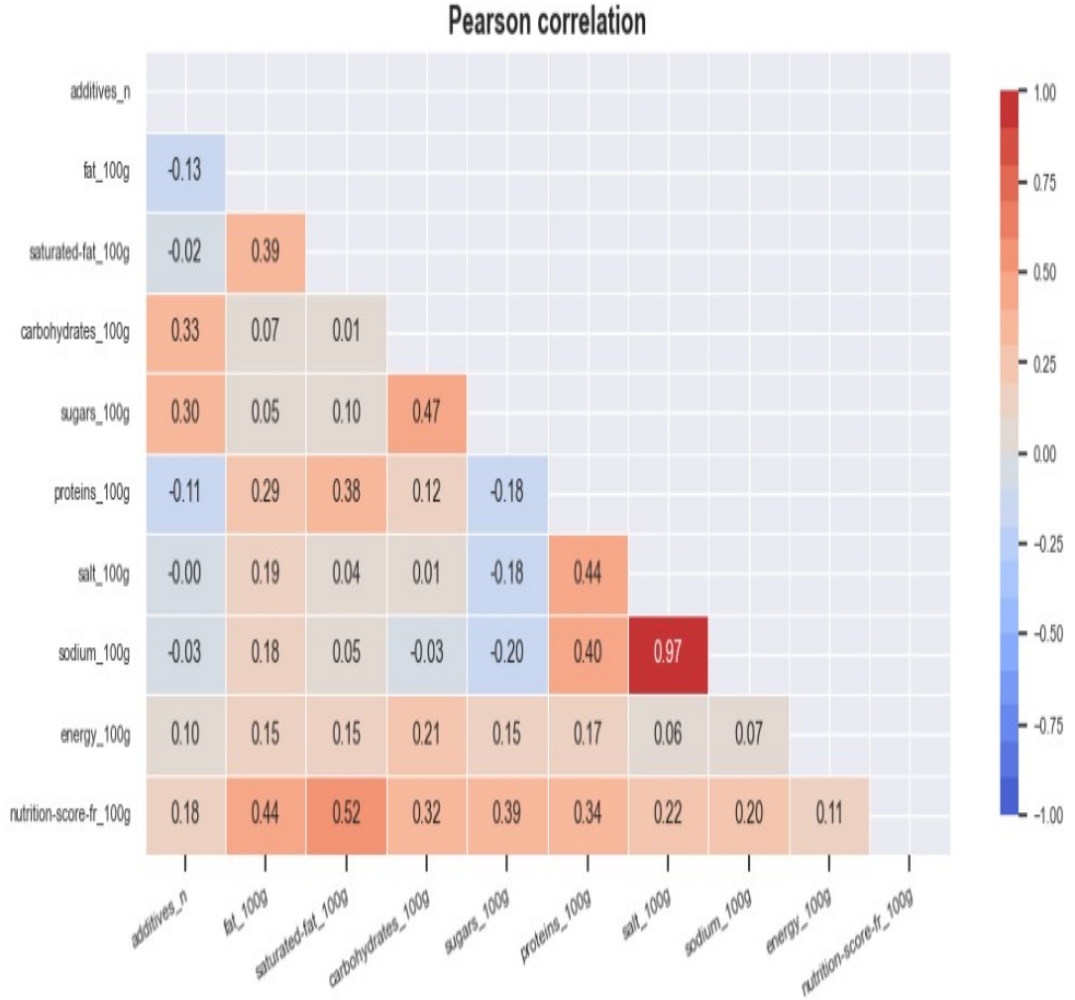
3. Analyse exploratoire – Analyse univariée : Anova

	features	p	bool_test
0	additives_n	0.004080	True
1	energy_100g	0.000041	True
2	fat_100g	0.240923	False
3	saturated-fat_100g	0.571011	False
4	carbohydrates_100g	0.362897	False
5	sugars_100g	0.801849	False
6	proteins_100g	0.100276	False
7	salt_100g	0.072309	False
8	sodium_100g	0.180924	False
9	nutrition-score-fr_100g	0.002083	True

On trouve que la distribution n'est pas normal.
Après une standardisation, je fais un test anova pour mieux comprendre.
On pourrait interpréter les résultats comme suit :

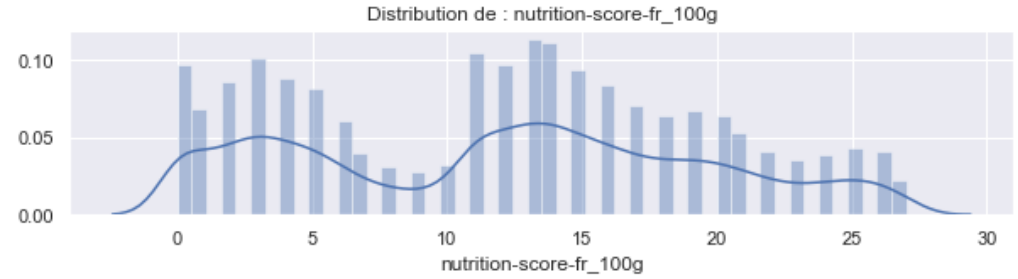
Les variables additives_n, energy_100g et nutrition-score-fr_100g ont des valeurs de p inférieures à 0,05, ce qui indique qu'elles ont un impact significatif sur votre variable cible.
Les variables fat_100g, saturated-fat_100g, carbohydrates_100g, sugars_100g, proteins_100g, salt_100g et sodium_100g ont des valeurs de p supérieures à 0,05, ce qui suggère qu'elles n'ont pas un impact significatif sur votre variable cible le nutrition_grade_fr. .

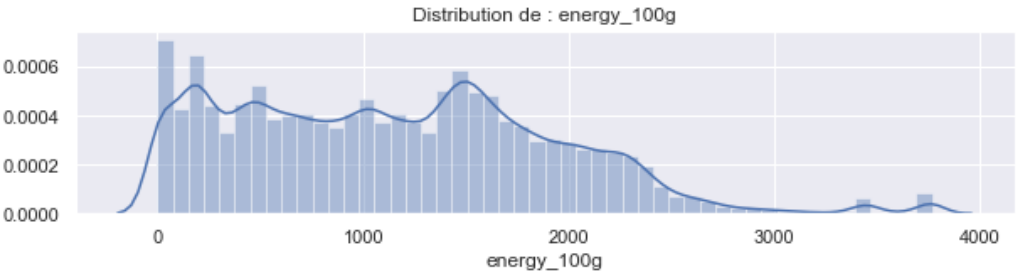
3. Analyse multivariée - corrélations

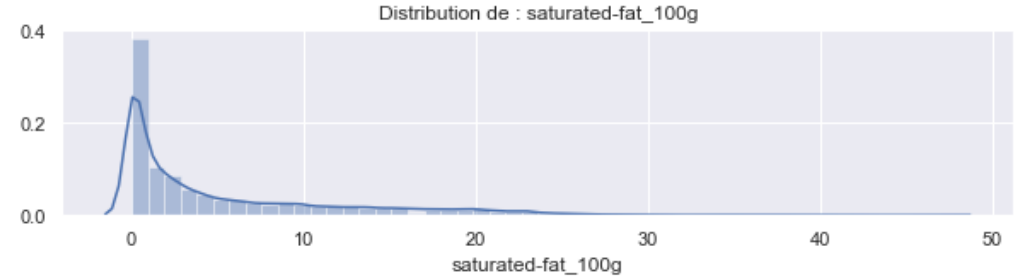


- **additives_n** pas de corrélation remarquable
- **energy_100g** faible corrélation avec :
  - fat_100g
  - saturated-fat_100g
  - carbohydrates_100g
  - nutrition-score-fr_100g
- **fat_100g** et saturated-fat_100g fortement corrélés
- **sugars_100g** forte corrélation avec carbohydrates_100g
- **sodium_100g** corrélation très forte avec salt_100g
- **nutrition-score-fr**_100g : corrélation avec:
  - fat_100g
  - saturated_fat_100g
  - carbohydrates_100g
  - sugars_100g
  - protein_100g

3. Analyse multivariée - corrélations





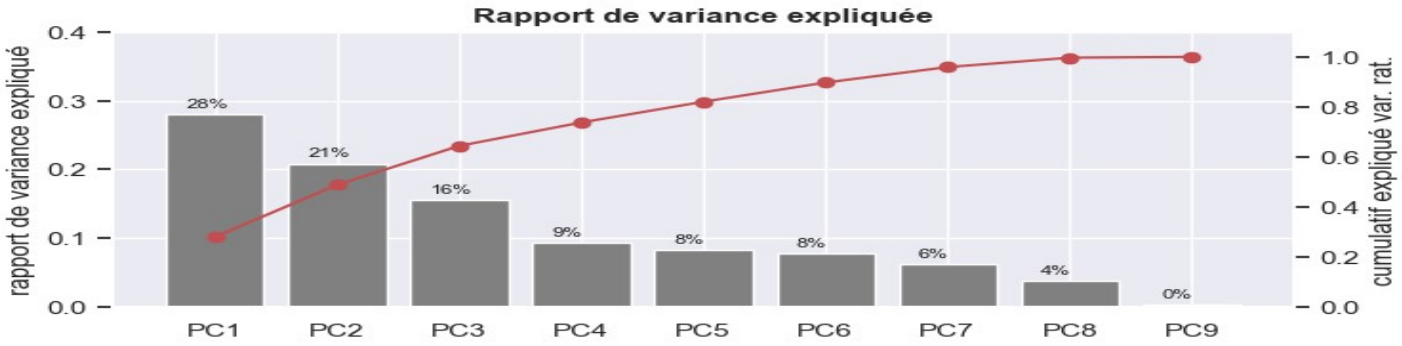


- **additives_n :** pas de correlation remarquable
- **energy_100g :** forte corerlation avec :
  - fat_100g
  - saturated-fat_100g
  - carbohydrates_100g
  - nutrition-score-fr_100g
- **fat_100g** et saturated-fat_100g fortement corrélés
- **sugars_100g** : forte correlation avec carbohydrates_100g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

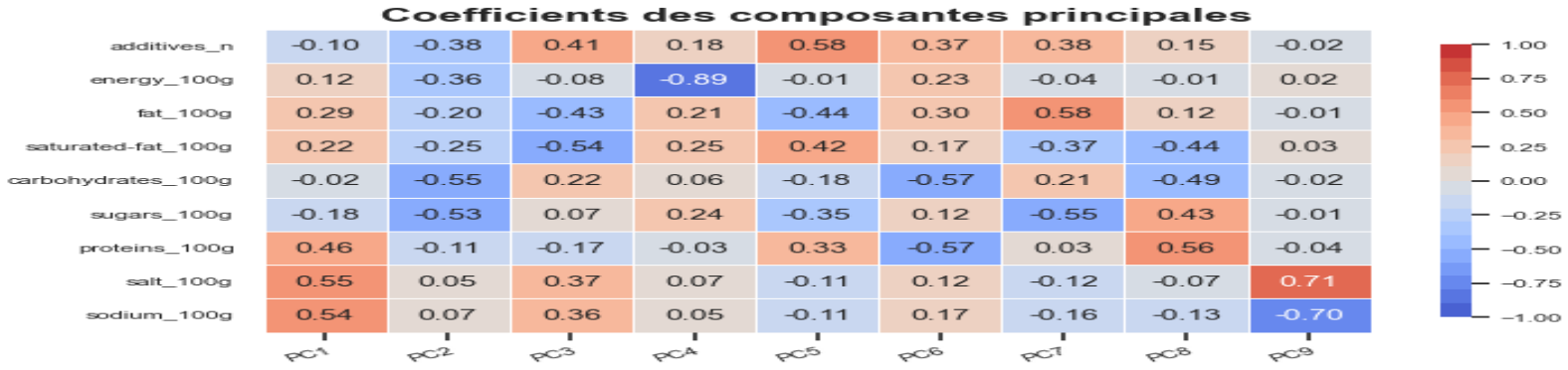
3. Réduction de dimension par Analyse par Composantes Principales

Au départ, 9 variables quantitatives pour l'ACP.

L'allure de la courbe de rapport expliqué cumulé ne présente pas de coude.
Dans notre cas, nous garderons de PC1 à PC4 qui expliquent collectivement 74% de la variance totale.



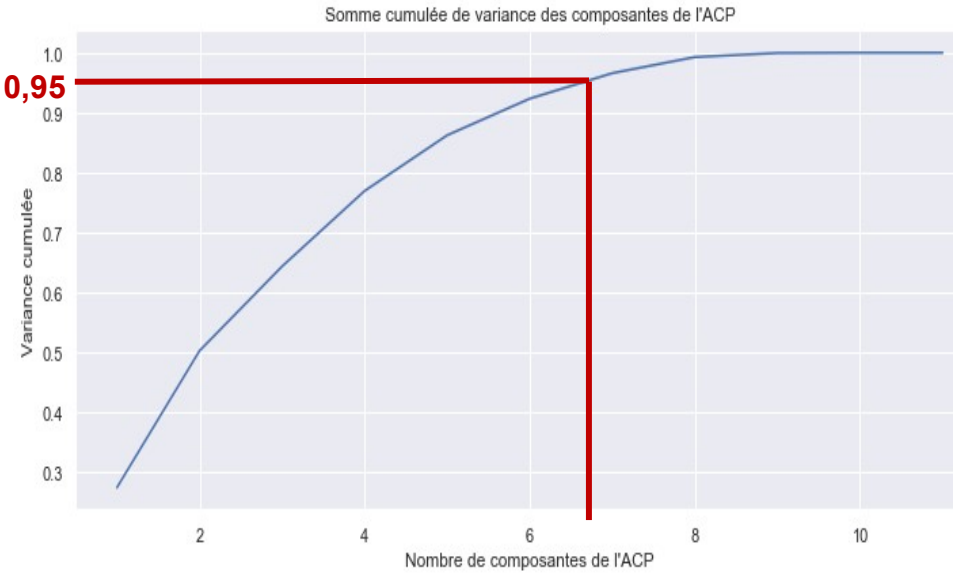
Le rapport de variance expliqué est une mesure de la quantité de variation dans les données qui est expliquée par un modèle statistique ou une méthode d'analyse. Il indique la proportion de la variance totale qui est expliquée par chaque composante principale dans le cas de l'analyse en composantes principales (PCA). Plus le rapport de variance expliquée est élevé pour une composante, plus cette dernière est importante pour expliquer les variations dans les données.

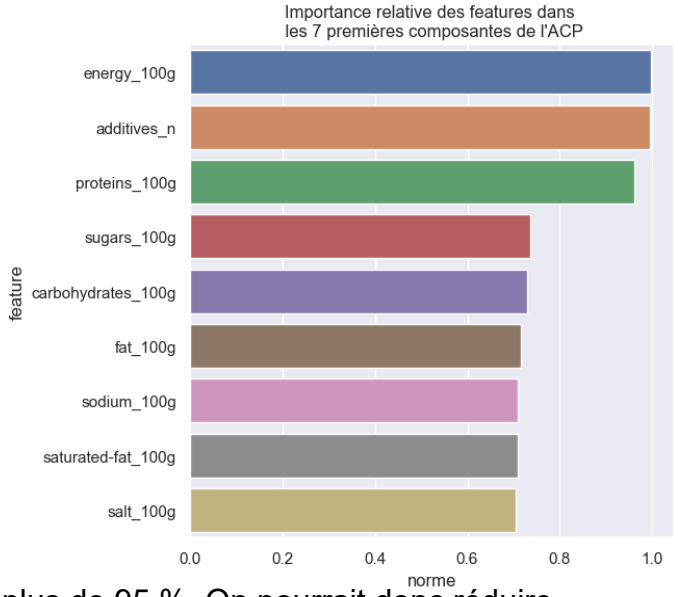


3. Réduction de dimension par Analyse par Composantes Principales

```
scaler = StandardScaler()
data_pca = scaler.fit_transform(data_pca)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

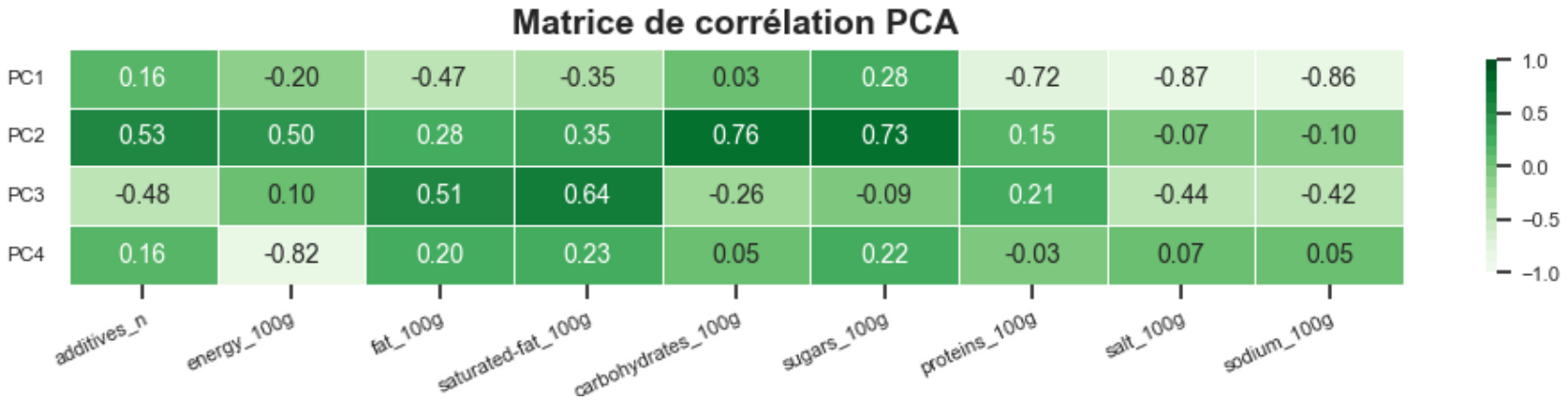
En résumé, l'ACP est une méthode qui permet de transformer un ensemble complexe de données en un ensemble de variables plus simples et compréhensibles, tout en préservant autant que possible l'information contenue dans les données initiales
L'ACP, a deux objectifs principaux, permet d'étudier :
la variabilité entre les individus, c'est-à-dire quelles sont les différences et les ressemblances entre individus ;
les liaisons entre les variables : y a-t-il des groupes de variables très corrélées entre elles, qui peuvent être regroupées en de nouvelles variables synthétiques ?





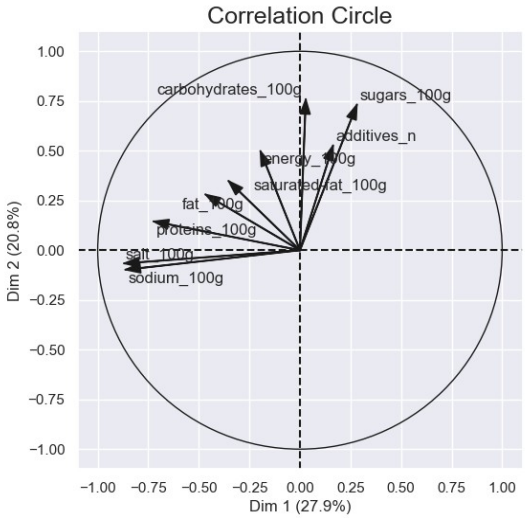
On voit qu'à partir de 7 caractéristiques on a une variance cumulée de plus de 95 %. On pourrait donc réduire notre jeu de données à 7 dimensions si on souhaitait gagner en temps de calcul / volume de données.

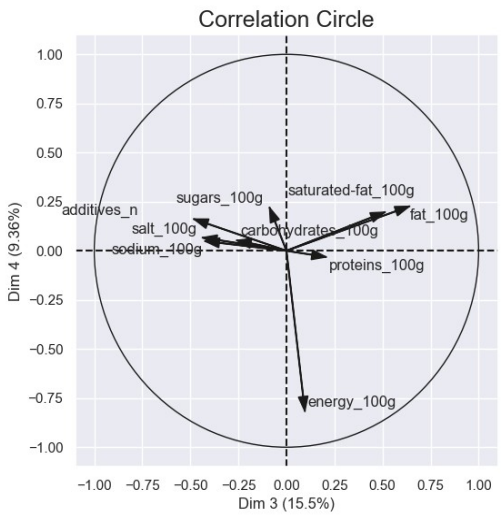
3. Réduction de dimension par Analyse par Composantes Principales



une matrice de corrélation en PCA aide à comprendre comment les différentes variables d'un ensemble de données sont liées les unes aux autres. Elle mesure la force et la direction de la relation linéaire entre chaque paire de variables. Cette information est utile pour identifier les variables qui ont le plus d'impact sur les données et pour réduire la dimensionnalité des données en identifiant les composantes principales qui capturent le maximum de variance.

3. Réduction de dimension par Analyse par Composantes Principales





- **energy_100g** corrélation avec :
- saturated-fat_100g
- fat_100g
- carbohydrates_100g

- **fat_100g** et saturated-fat_100g corrélés
- **sugars_100g** corrélation avec : carbohydrates_100g
- **sodium_100g** corrélation salt_100g

- Analyse des composantes
- Composantes 1

Les variables quantitatives corrélées à cette composantes sont le taux de sel et de sodium.
Cette composante représente donc l'apport en sel.

- Composantes 2

Les variables quantitatives contre-corrélées à cette composantes sont le taux de graisses, de graisses saturées et l'apport calorique.
Cette composante représente donc l'apport en graisse et l'apport en énergie.

- Composantes 3

Les variables quantitatives corrélées à cette composantes sont dans un sens le taux de graisses, de graisses saturées et dans l'autre sens, le taux de glucide et de sucre.
Cette composante représente donc le côté sucré ou salé de l'aliment.

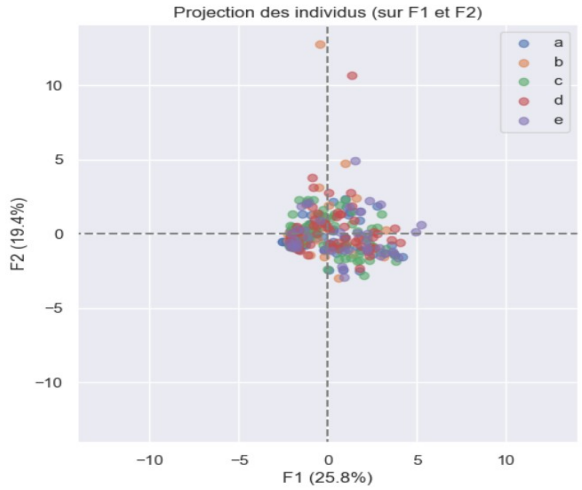
- Composantes 4

Les variables quantitatives anti corrélées à l'apport en énergie.

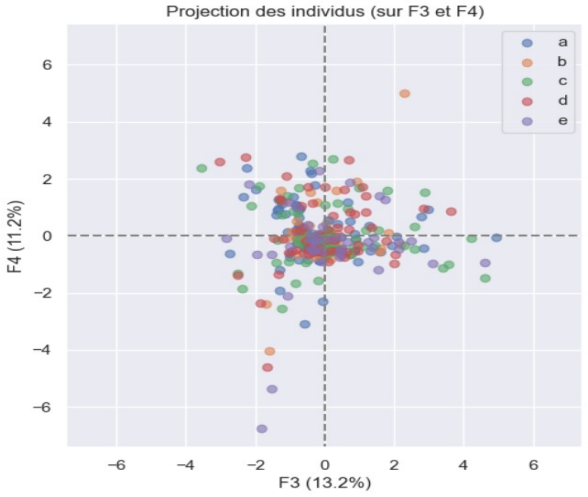
- Analyse des nutriscores projetés sur les composantes principales

3. Réduction de dimension par Analyse par Composantes Principales

Graphique fait avec échantillon aléatoire de 300 échantillons à partir des données transformées par l'analyse en composantes principales (PCA) et normalisées à l'aide de la StandardScaler.



Les aliments se répartissent équitablement sur tous les axes, il n'y a pas de corrélation entre le nutriscore et les apports énergétiques, les taux de graisse, de sel ou
de sucre. Le premier graphique n'est pas significatif.



Nutriscore A : plus les aliments sont salés, moins ils doivent contenir d'additifs pour être dans cette catégorie.

Nutriscore B : les aliments sont principalement plutôt sucré avec essentiellement peu d'additifs.

Nutriscore C : même si on retrouve encore beaucoup d'aliments vers le centre, donc,
ni trop sucré, ni trop salé, et avec peu d'additifs, on a également dans cette catégorie des aliments qui s'éloigne beaucoup du centre.

Nutriscore D : les aliments se répartissent sur une très large gamme de valeur sans corrélation
Avec les composantes sucré/salé ou présence d'additifs.

Nutriscore E : les aliments sont essentiellement sucrés et sans corrélation avec le nombre d'additifs.

4. Faits pertinents pour l'application - suite

La synthèse des différentes conclusions sur la faisabilité de votre projet.

- Exemple de code pour prédire un nutriscore

```
# Préparer les caractéristiques de l'aliment
nouvel_aliment = [...]  # Liste des caractéristiques de l'aliment

# Charger le modèle
model = RandomForestRegressor()  # Exemple avec un modèle de régression forestière
model.load_model("chemin/vers/le/model")  # Charger le modèle entraîné à partir du fichier

# Effectuer la prédiction
prediction = model.predict([nouvel_aliment])

# Afficher la prédiction
print("La prédiction du Nutri-Score pour le nouvel aliment est :", prediction)
```

4. Faits pertinents pour l'application

3 observations :

- Dépendance des données
- Corrélation forte de certaines variables avec le nutriscore
- • Le nutriscore dépend principalement de l'énergie et de la teneur en graisses

5. Synthèse

La synthèse des différentes conclusions sur la faisabilité de votre projet
Mon projet le calcul du nutriscore :

• Le nutriscore calculé avec les règles d'aujourd'hui ne prend pas assez en compte ni les apports énergétiques ni les apports en sucre.

• De plus, même dans les produits au nutriscore A et B, les additifs sont déjà présents.

• Il serait donc pertinent de recalculer un nouveau nutriscore prenant en compte ces critère afin d'améliorer sensiblement l'alimentation des populations.

Merci de votre attention