

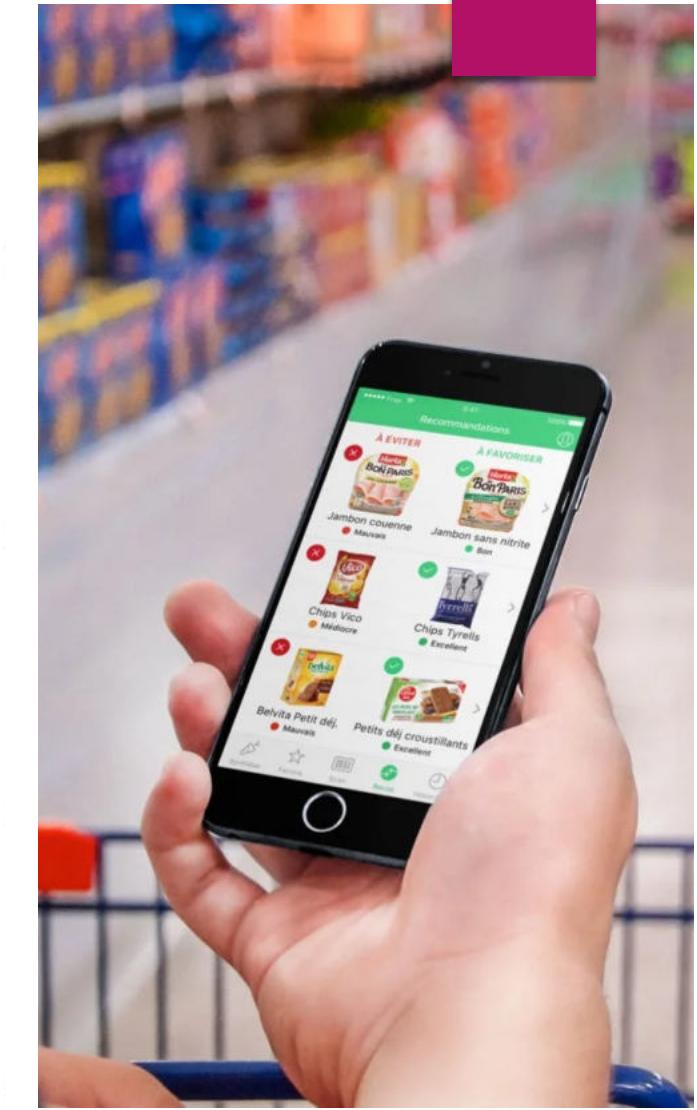
# Projet n°3 :

## Concevoir une application au service de la santé publique.

ALEXANDRE JACQUELINE – MARS 2022

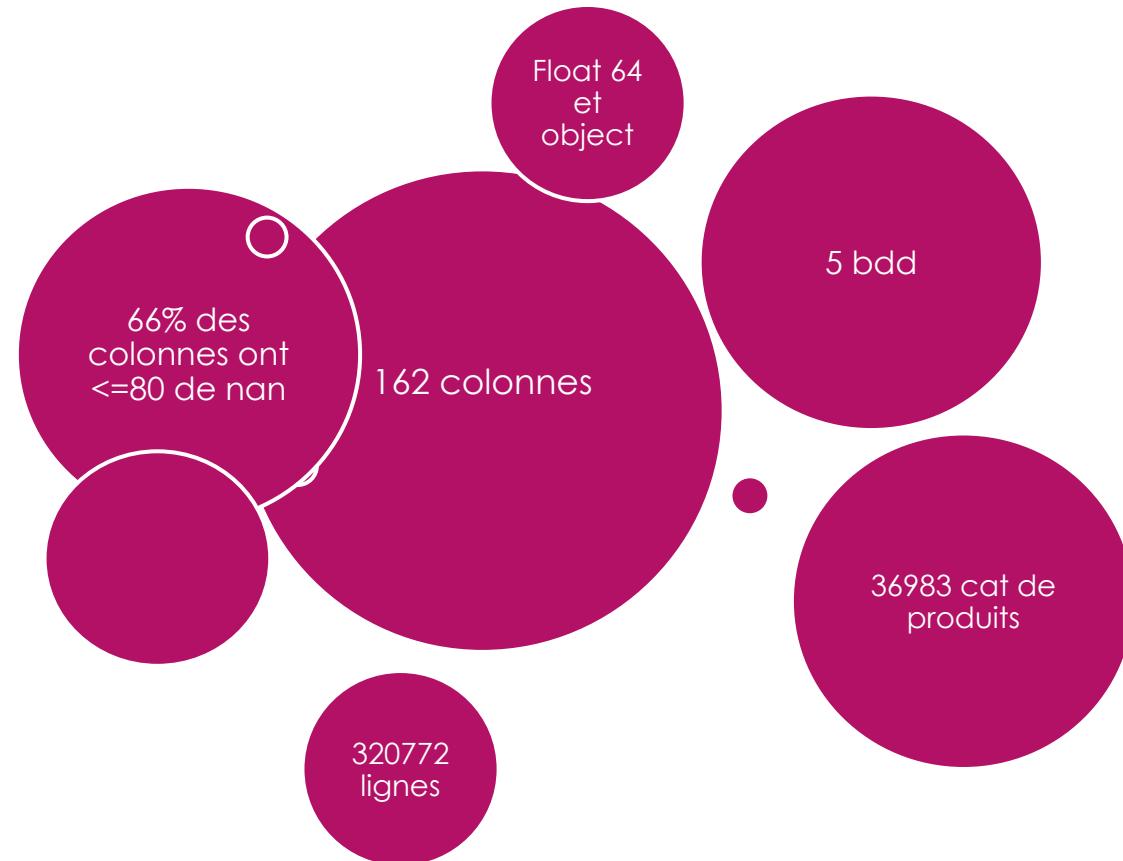
# Sommaire

- ▶ Présentation du jeux de données
- ▶ Opérations de nettoyage
- ▶ Analyse exploratoire
- ▶ Analyses multivariées
- ▶ ACP
- ▶ Conclusion

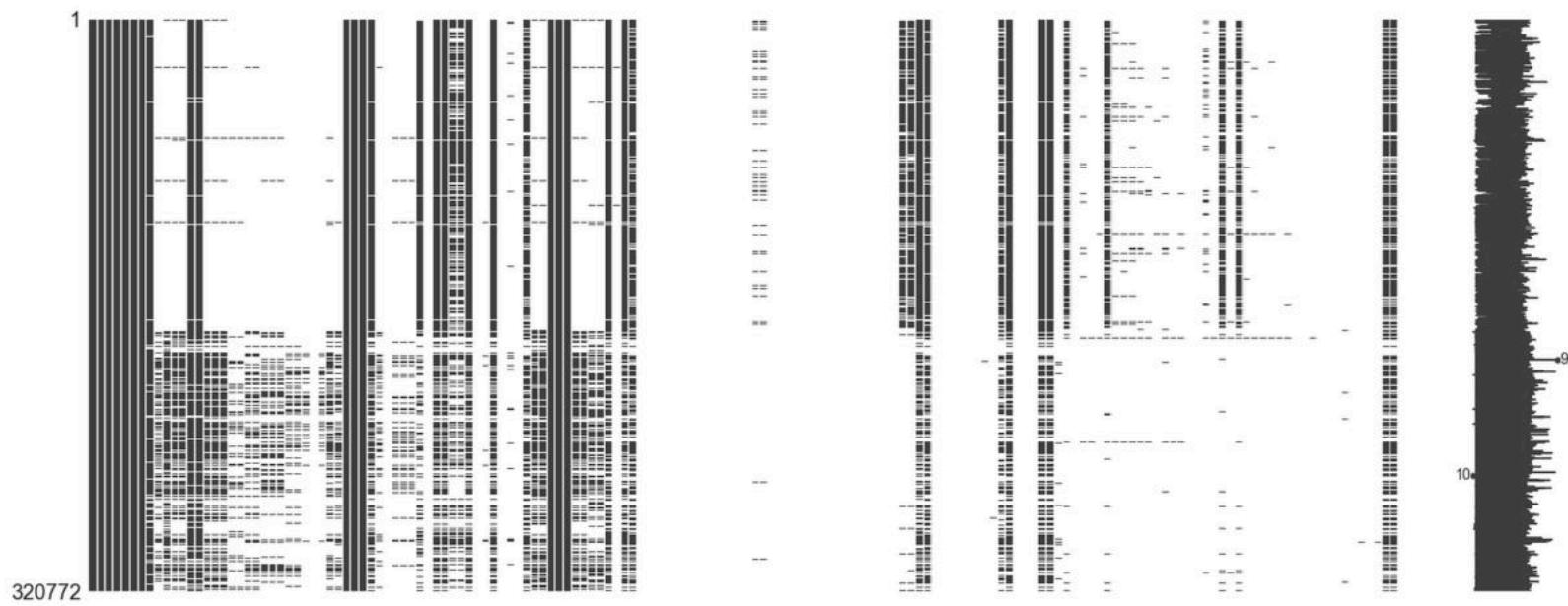


# Présentation du jeux de données

# Présentation du jeux de données



# Présentation du jeux de données



320772 X 162

# Opérations de nettoyage

# Opérations de nettoyage

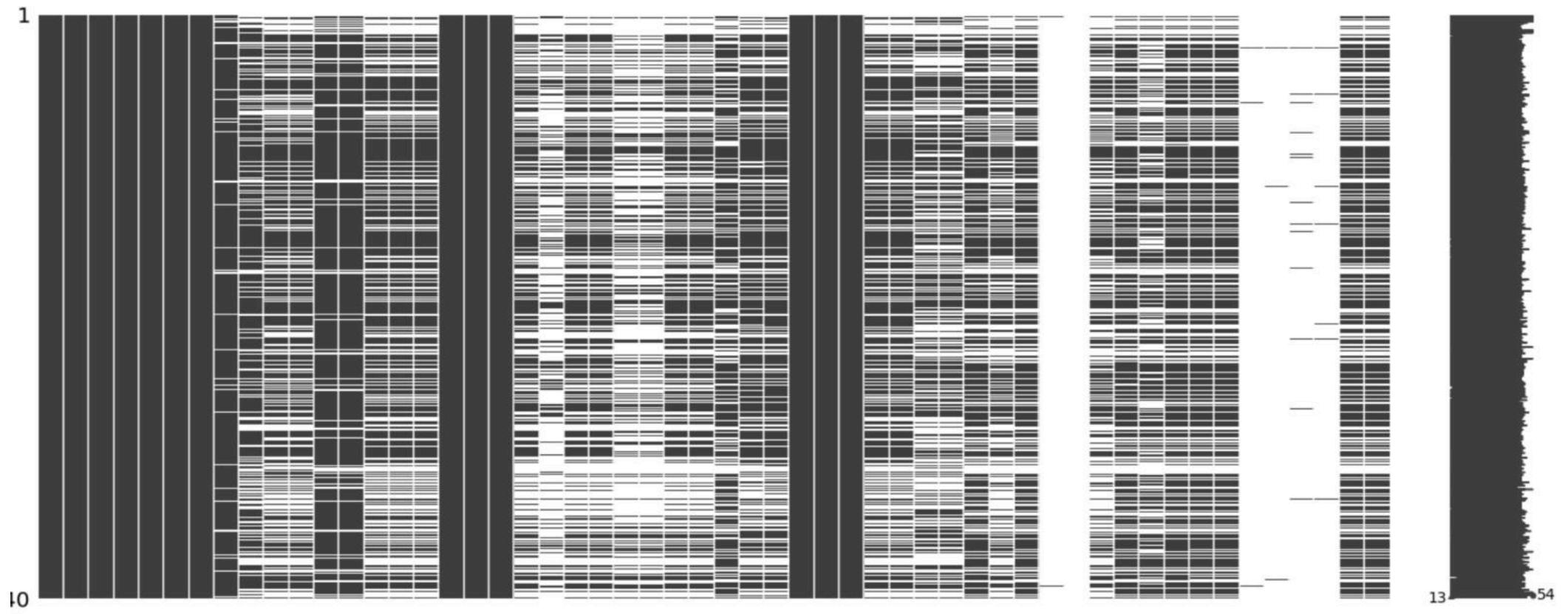
Opérations	Evolution	Notes:
Suppression des colonnes avec plus de 80% de valeur nulle	320772 X 54	Nous supprimons ces colonnes car impossible à compléter
Filtre sur les valeurs France	98440 X 54	Nous filtrons sur le pays France, afin d'avoir une vision plus fine de la BDD
Suppression des features avec un taux de valeurs NA sup à 50 %.	98440 X 48	Ces variables sont principalement des mesures physico-chimiques directes sur les produits
Analyse et supp des variables redondantes	98440 X 35	Certaines variables sont redondantes, suffixées par _tags ou _en qui ne font que reprendre d'autres features traduites ou simplifiées.
Ajout des valeurs manquantes avec méthode d'imputation et KNN.	59033 X 32	Utilisation de l'imputation par la médiane car rejets de l'hypothèse de normalité des distributions de ces variables

# Variable redondante

```
[59]: def search_redundant_col(df):
    redundant_columns = []
    for col in df.columns:
        if "_fr" in col:
            en = col.replace('_fr','')
            tags = col.replace('_fr','_tags')
            print("{:<20} 'Sans suffixe' -> {} ; 'Suffixe _tags' -> {}".format(col,
                                                                           en in df.columns, tags in df.columns))
        if en in df.columns :
            redundant_columns.append(en)
        if tags in df.columns :
            redundant_columns.append(tags)

        if '_tags' in col:
            tags_2 = col.replace('_tags','')
            print("{:<20} 'Suffixe _tags' -> {} ;".format(tags_2, tags_2 in df.columns))
            if tags_2 in df.columns :
                redundant_columns.append(col)

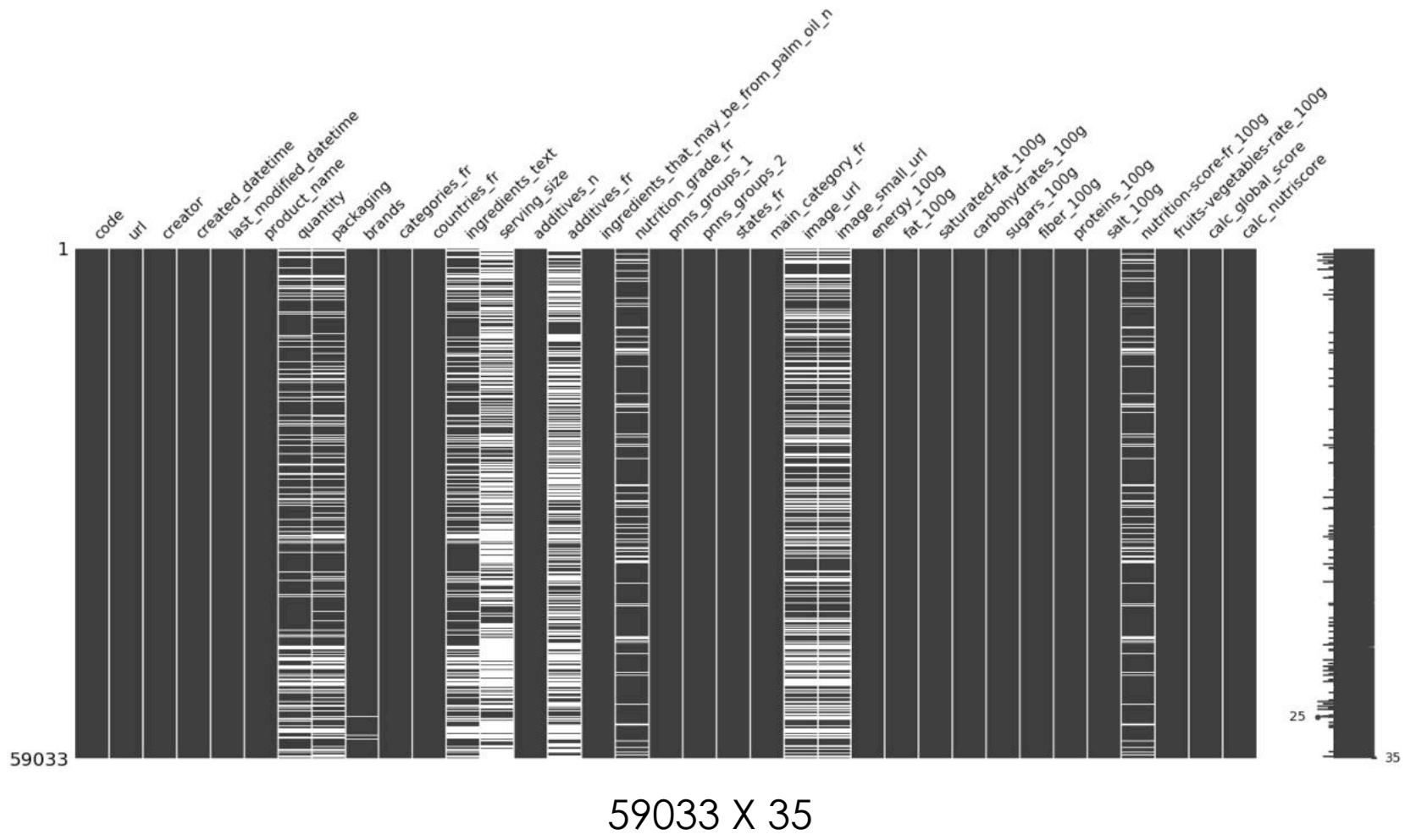
    return redundant_columns
```



320772 X 54

# Compléter les valeurs manquantes

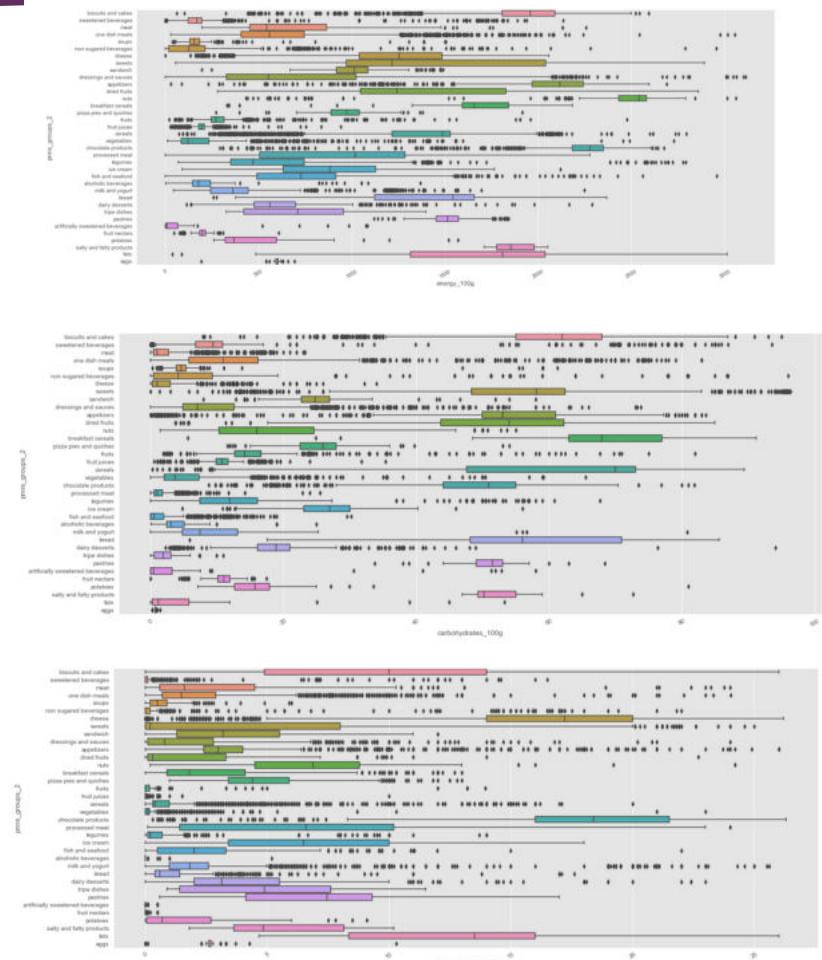
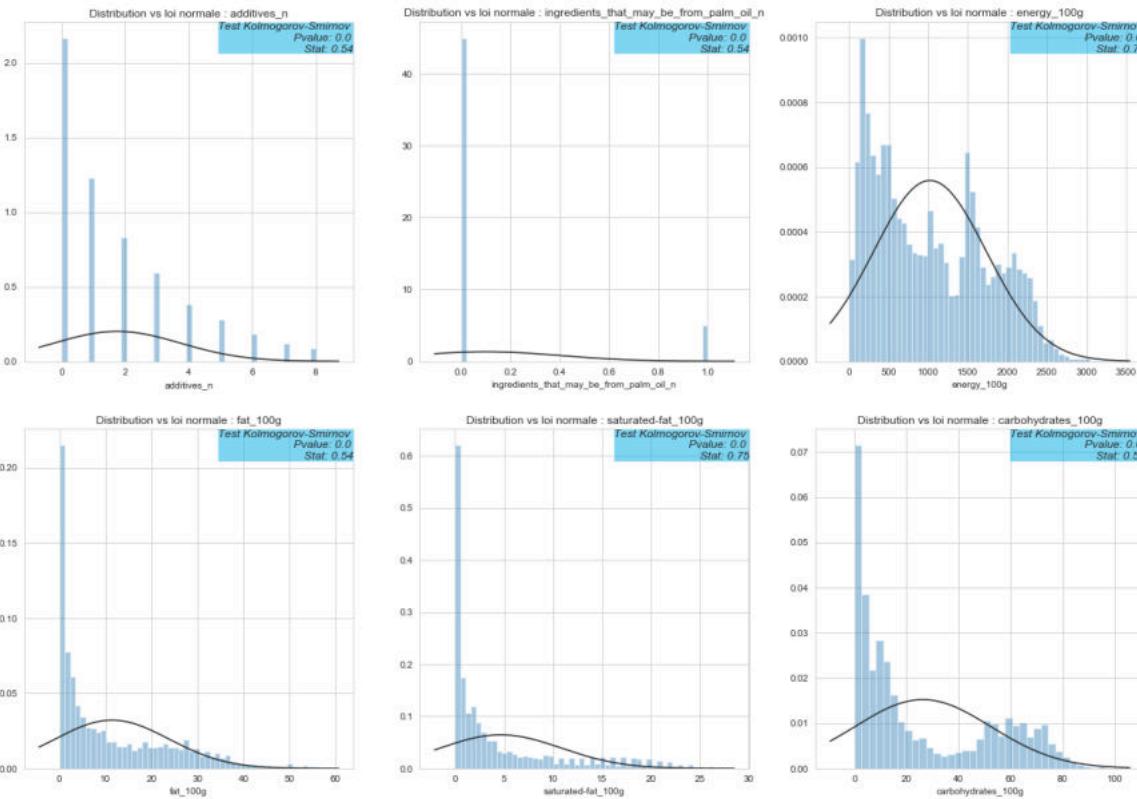
- Suppression des produits sans nom, ni catégorie = 29774 produits reste
- Suppression du nutri-score négatif
- Suppression des lignes dont les valeur numérique sont à 0 ou nulles ainsi que les valeurs négatives
- Pour les suffixes \_100g, suppression des variables de nutriments supérieurs au seuil de 100g
- Pour les variables restantes, suppression des outliers en se basant sur la médiane et l'écart-type
- Compléter les valeurs nulles par la médiane de la catégorie pnns\_group\_2
- Pour les autres variables ne suivent pas la loi gaussienne, nous allons imputer avec KNN





Complete les valeurs  
manquantes

# Imputation et KNN



# Nuage de mots des 100 meilleures catégories



[92]:

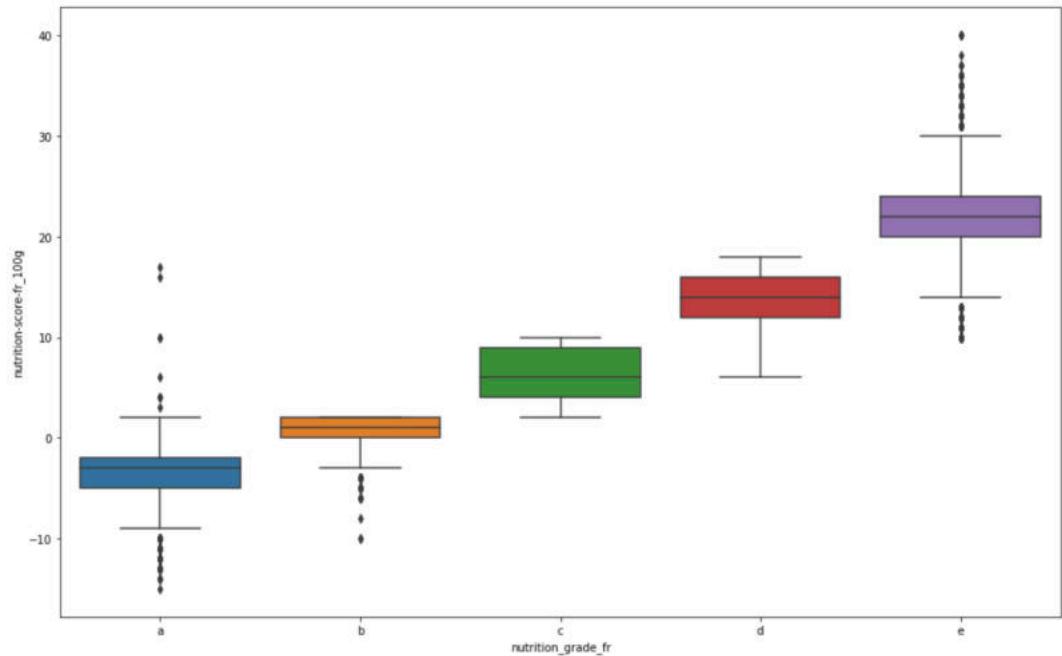
Keyword	count
0	unknown 36485
1	Aliments et boissons à base de végétaux 18675
2	Aliments d'origine végétale 15584
3	Snacks sucrés 8740
4	Boissons 8693
5	Produits laitiers 6563
6	Plats préparés 6432
7	Céréales et pommes de terre 5758
8	Aliments à base de fruits et de légumes 5559
9	Frais 4780

# Calcul du nutri-score manquants

# Le Nutriscore

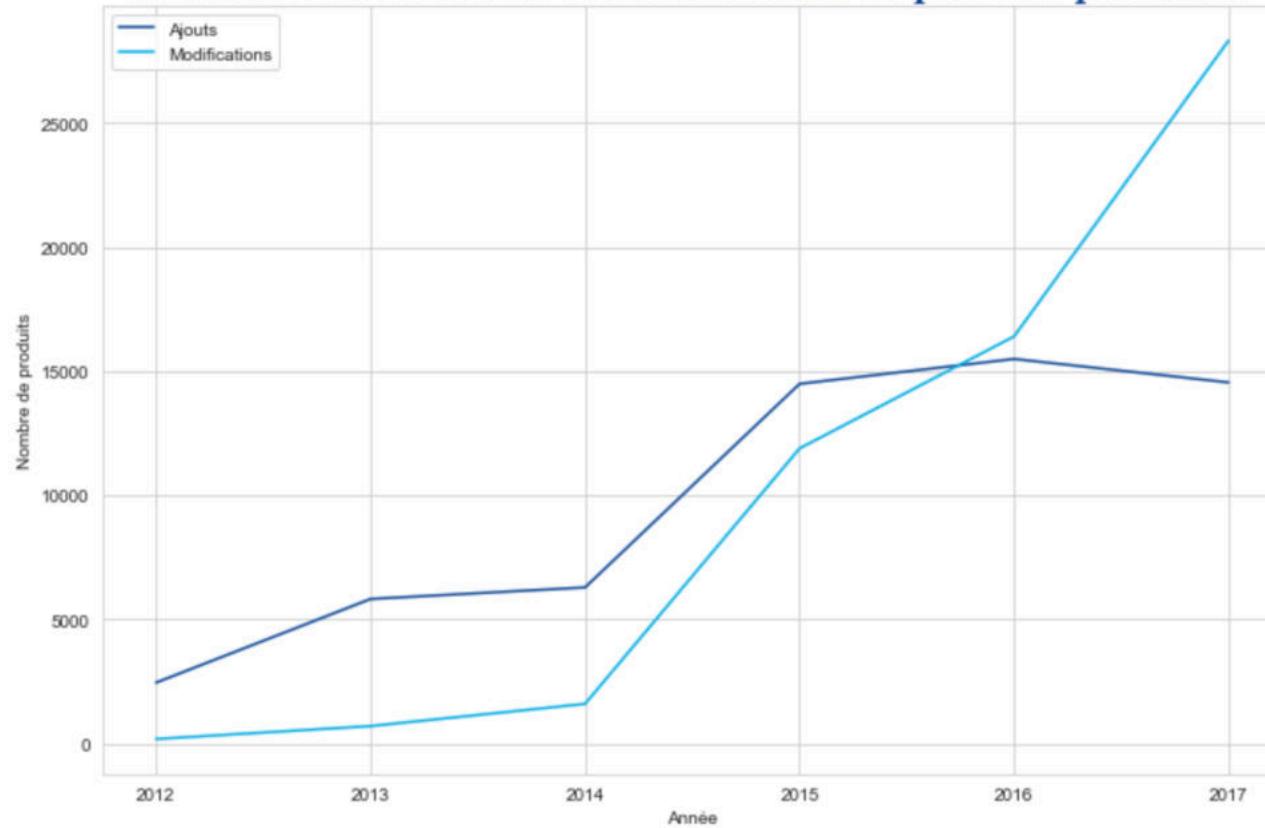
## Nutri-Score, c'est quoi ?

- Un logo apposé en face avant des emballages qui informe sur la qualité nutritionnelle des produits sous une forme simplifiée et complémentaire à la déclaration nutritionnelle obligatoire (fixée par la réglementation européenne)
- Basé sur une échelle de 5 couleurs : du vert foncé au orange foncé
- Associé à des lettres allant de A à E pour optimiser son accessibilité et sa compréhension par le consommateur



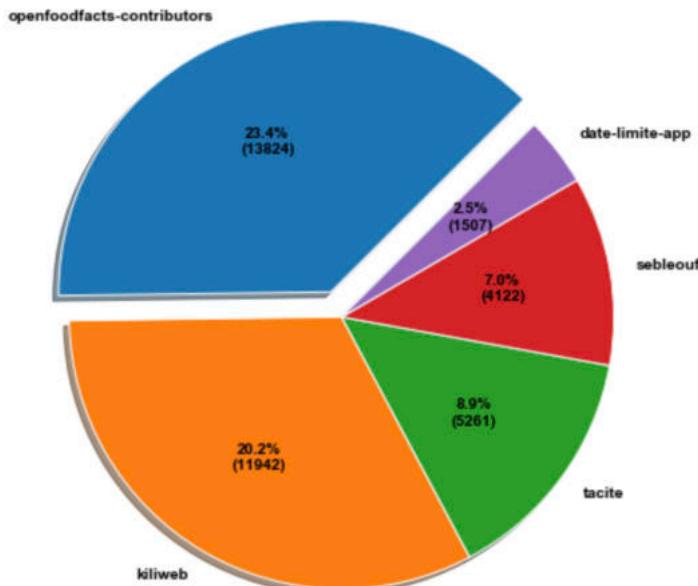
# Analyse exploratoire

### **Evolution des créations et modifications de produits par année**



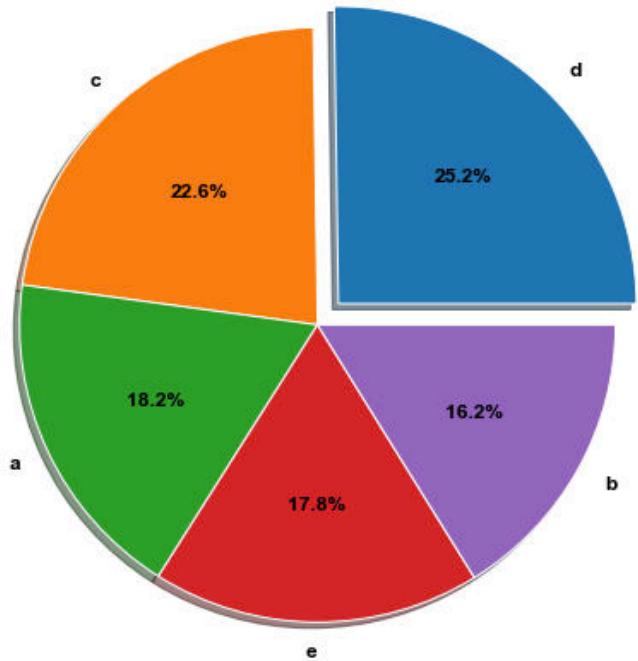
# Les contributeurs

Qui sont les 5 meilleurs contributeurs ?

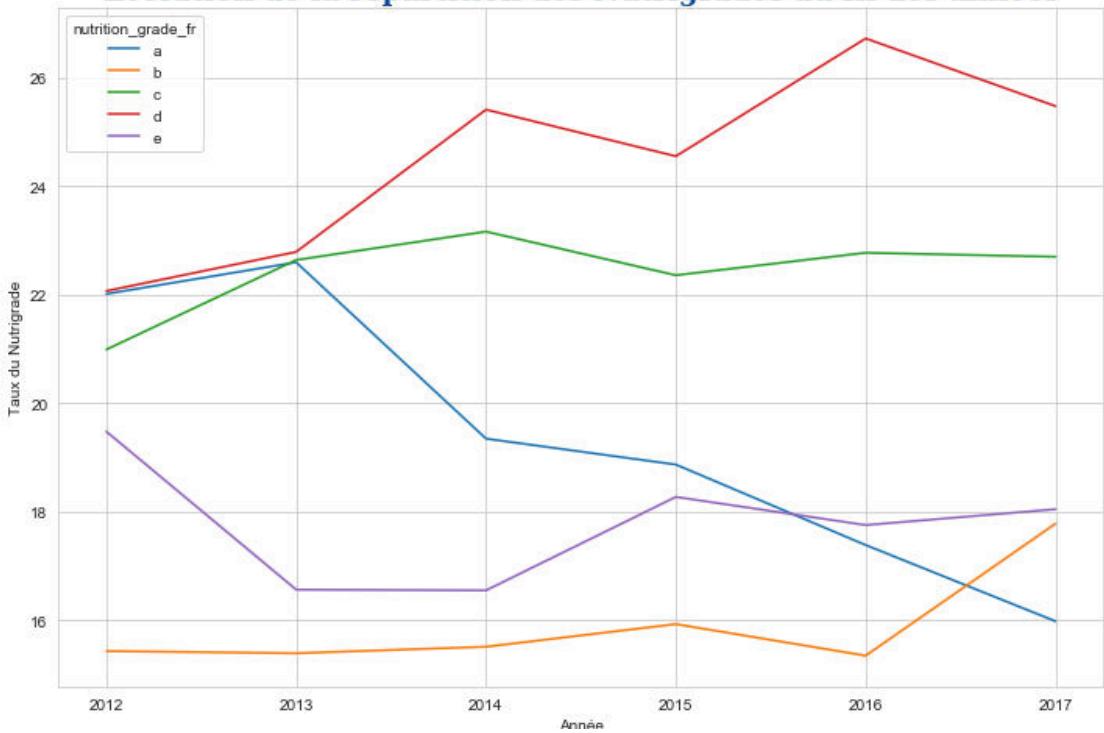


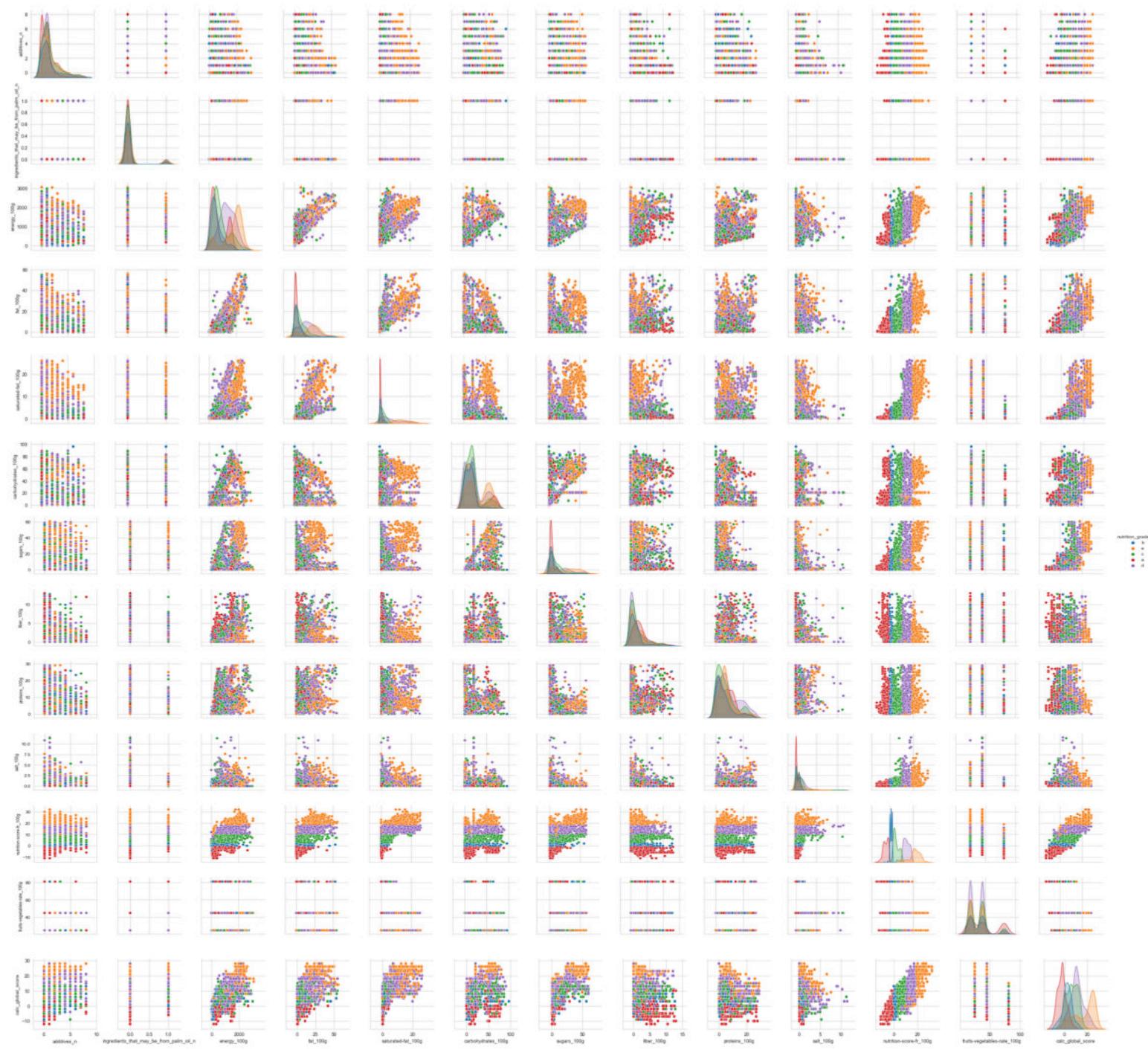
# Nutrigrades

Répartition des Nutrigrades

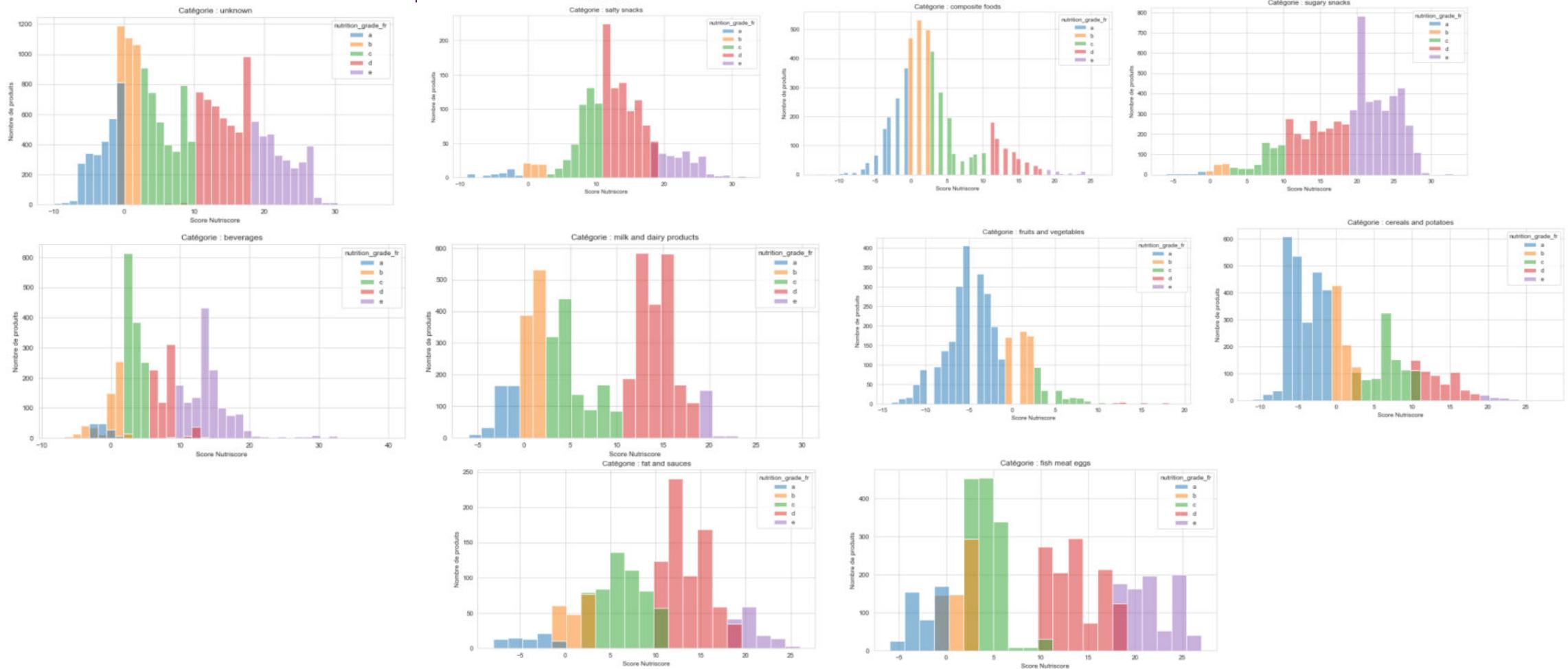


Evolution de la répartition des Nutrigrades au fil des années



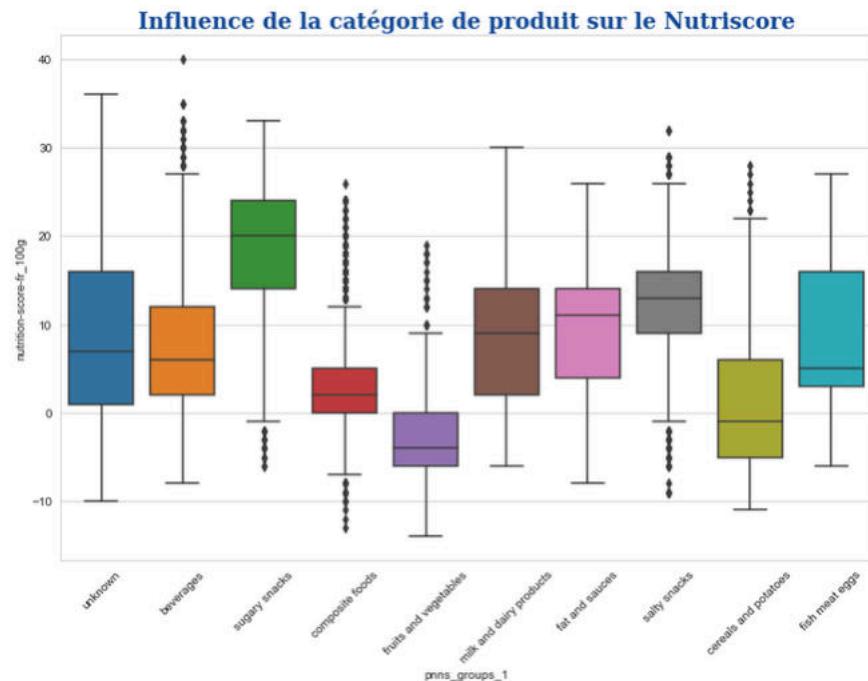


# Distribution des scores Nutriscore par catégorie



# Analyse de variance

# ANOVA (Analyse de la variance)



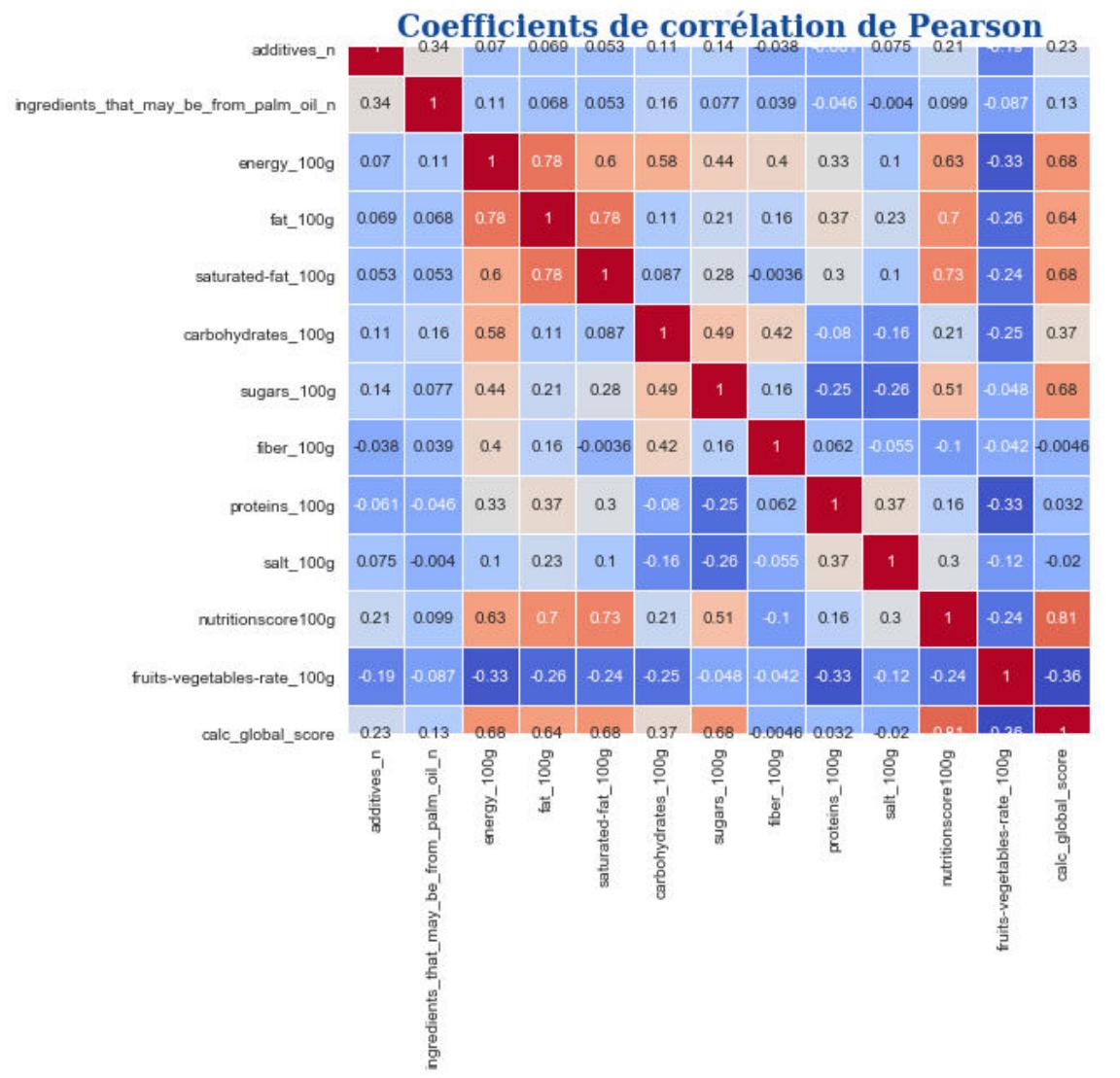
OLS Regression Results				
Dep. Variable:	nutritionscore100g	R-squared:	0.327	
Model:	OLS	Adj. R-squared:	0.327	
Method:	Least Squares	F-statistic:	2851.	
Date:	Fri, 01 Apr 2022	Prob (F-statistic):	0.00	
Time:	15:36:27	Log-Likelihood:	-1.8064e+05	
No. Observations:	52871	AIC:	3.613e+05	
Df Residuals:	52861	BIC:	3.614e+05	
Df Model:	9	Covariance Type:	nonrobust	
t	[0.025 0.975]	coef	std err	t
Intercept		7.3885	0.116	63.811
pnns_groups_1[T.cereals and potatoes]	7.162	7.615	-6.1204	0.158
pnns_groups_1[T.composite foods]	-6.430	-5.811	-4.3320	0.159
pnns_groups_1[T.fat and sauces]	-4.644	-4.020	2.3152	0.214
pnns_groups_1[T.fish meat eggs]	1.895	2.735	1.7212	0.161
pnns_groups_1[T.fruits and vegetables]	1.405	2.037	-10.6321	0.178
pnns_groups_1[T.milk and dairy products]	-10.981	-10.283	0.8107	0.158
pnns_groups_1[T.salty snacks]	0.502	1.120	5.5024	0.216
pnns_groups_1[T.sugary snacks]	5.079	5.926	10.9337	0.150
pnns_groups_1[T.unknown]	10.640	11.228	1.1655	0.128
pnns_groups_1[T.unknown]	0.915	1.416		9.103
Omnibus:	1612.748	Durbin-Watson:	1.258	
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1274.317	
Skew:	0.299	Prob(JB):	1.93e-277	
Kurtosis:	2.529	Cond. No.	13.1	

# Test de Fisher: tableau d'analyse de variance

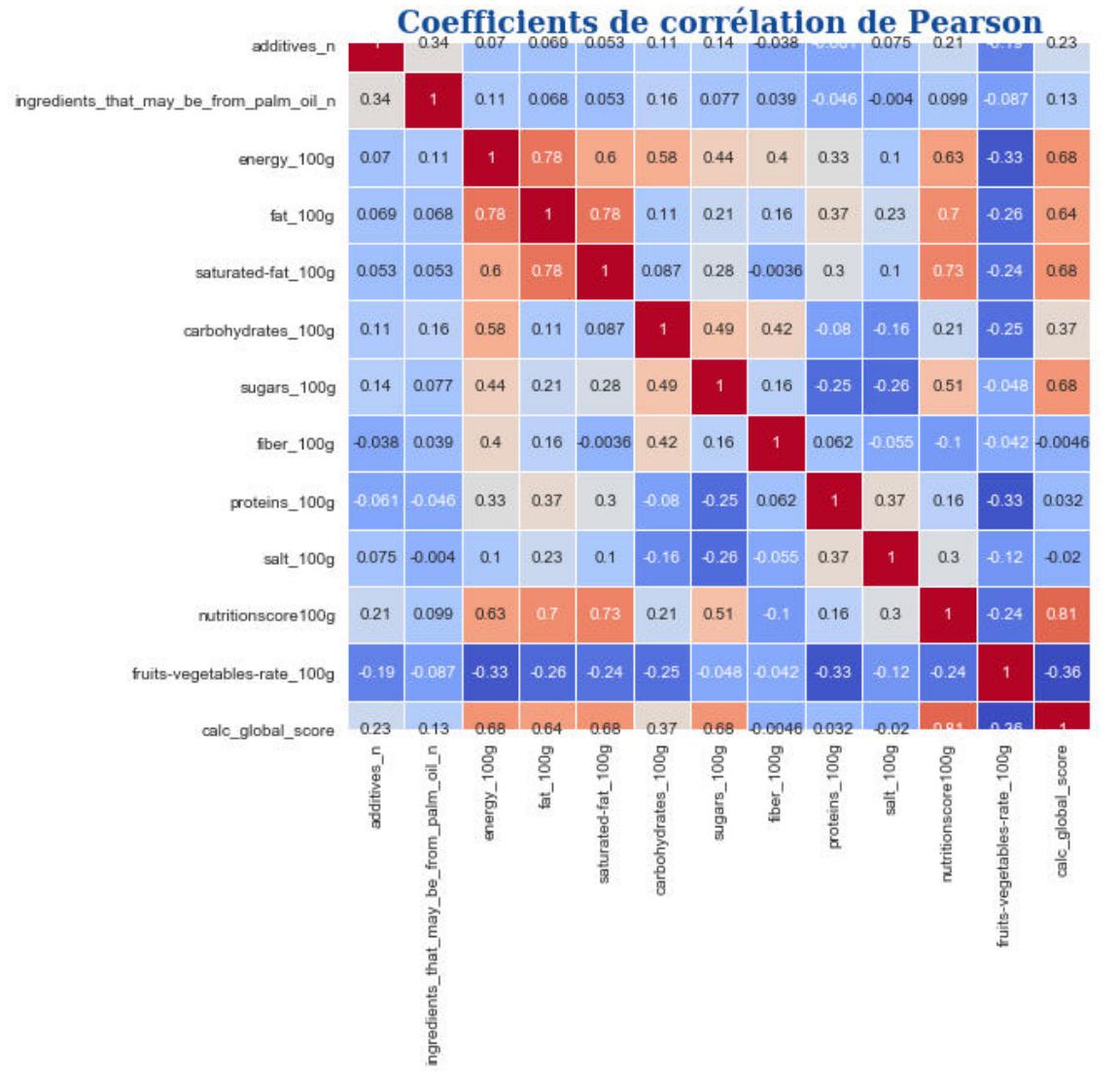
	sum_sq	df	F	PR(>F)
<b>pnns_groups_1</b>	1.394542e+06	9.0	2850.934604	0.0
<b>Residual</b>	2.873010e+06	52861.0	NaN	NaN

# Analyse des corrélations linéaires

# Coefficients de corrélation de Pearson



# Corrélation de Pearson avec le Nutriscore

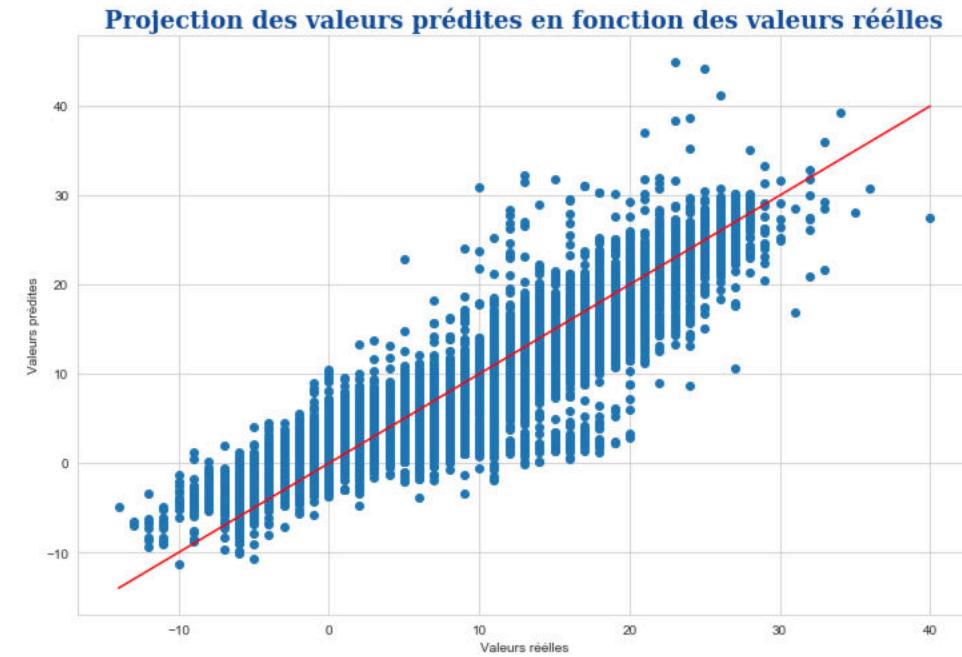


# Analyses multivariées

# Analyses multivariées : Régression linéaire multivariée

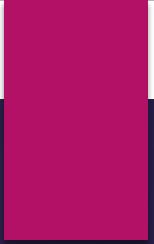
[34] :	Métrique	Baseline
0	MAE	7.844349
1	MSE	82.083512
2	RMSE	9.059995
3	R <sup>2</sup>	-0.000042

[36] :	Métrique	Baseline	LinearRegression
0	MAE	7.844349	2.717468
1	MSE	82.083512	13.176506
2	RMSE	9.059995	3.629946
3	R <sup>2</sup>	-0.000042	0.839468



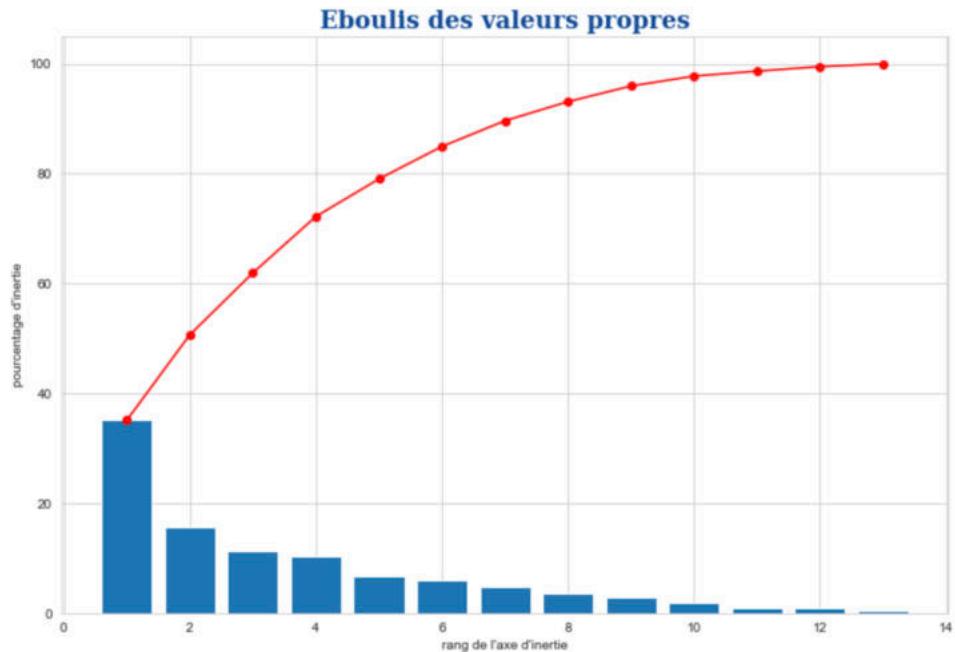
# Régression linéaire avec catégorie produits

[39] :	Métrique	Baseline	LinearRegression	LinearRegression cat
0	MAE	7.844349	2.717468	2.107047
1	MSE	82.083512	13.176506	7.641011
2	RMSE	9.059995	3.629946	2.764238
3	$R^2$	-0.000042	0.839468	0.906908



ACP

# Eboulis des valeurs propres



```
[45]: #Espace des composantes principales
pcs = pca.components_

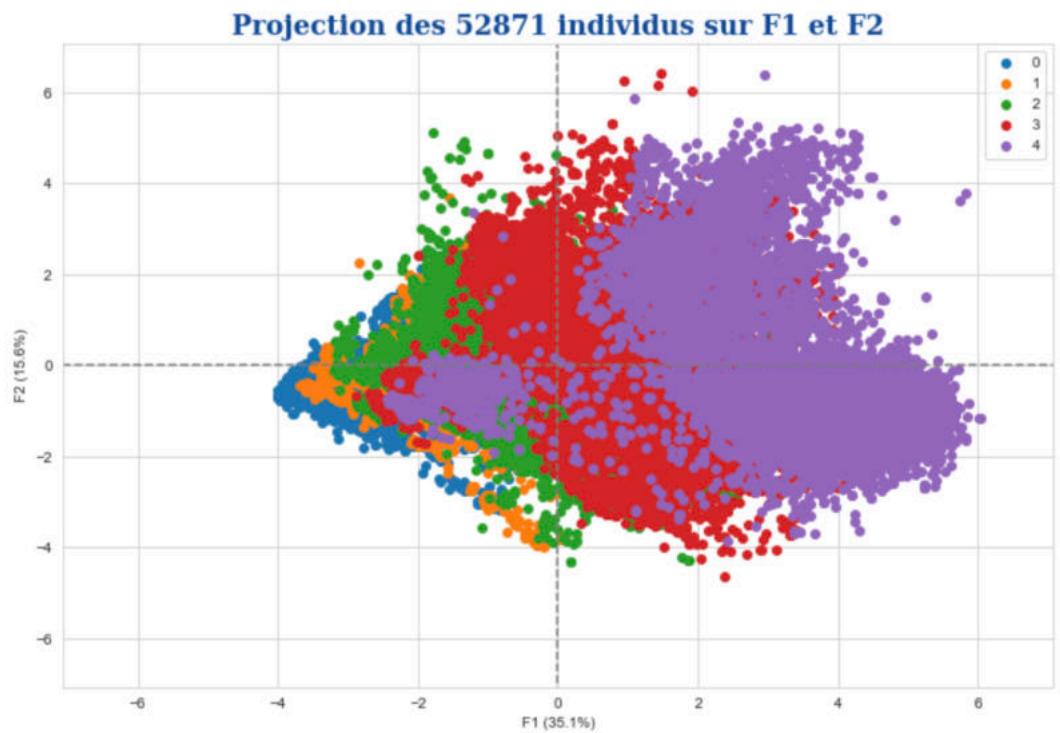
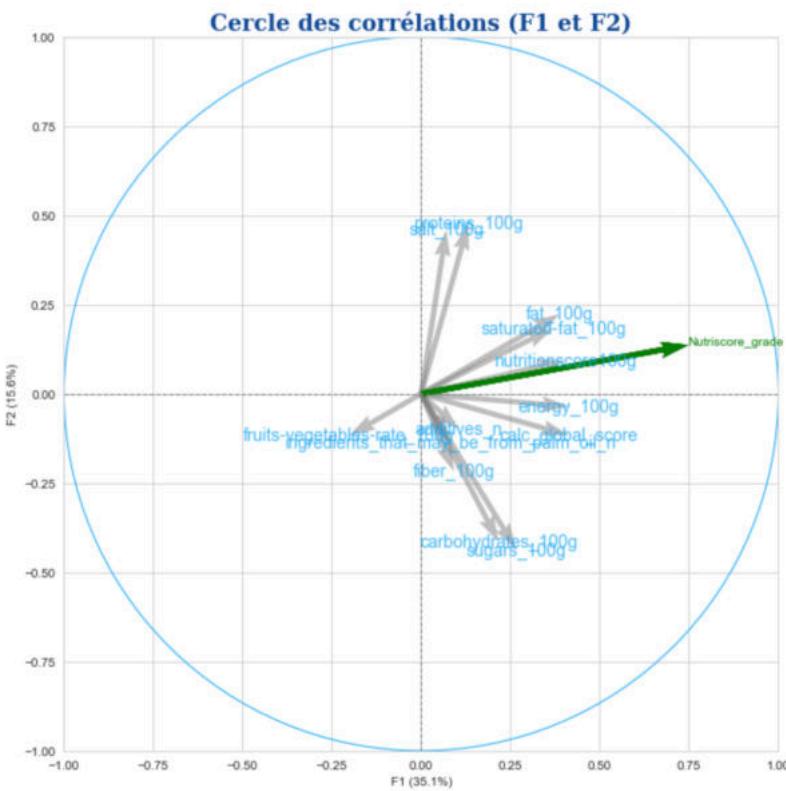
#Matrice des corrélations variables x facteurs
p = X.shape[1]
sqrt_valprop = np.sqrt(pca.explained_variance_)
corvar = np.zeros((p, p))
for dim in range(p):
    corvar[:,dim] = pcs[dim,:] * sqrt_valprop[dim]

#on affiche pour les deux premiers plans factoriels
corr_matrix = pd.DataFrame({'feature':X.columns,'CORR_F1':corvar[:,0],'CORR_F2':corvar[:,1],
                             'CORR_F3':corvar[:,2], 'CORR_F4':corvar[:,3]})

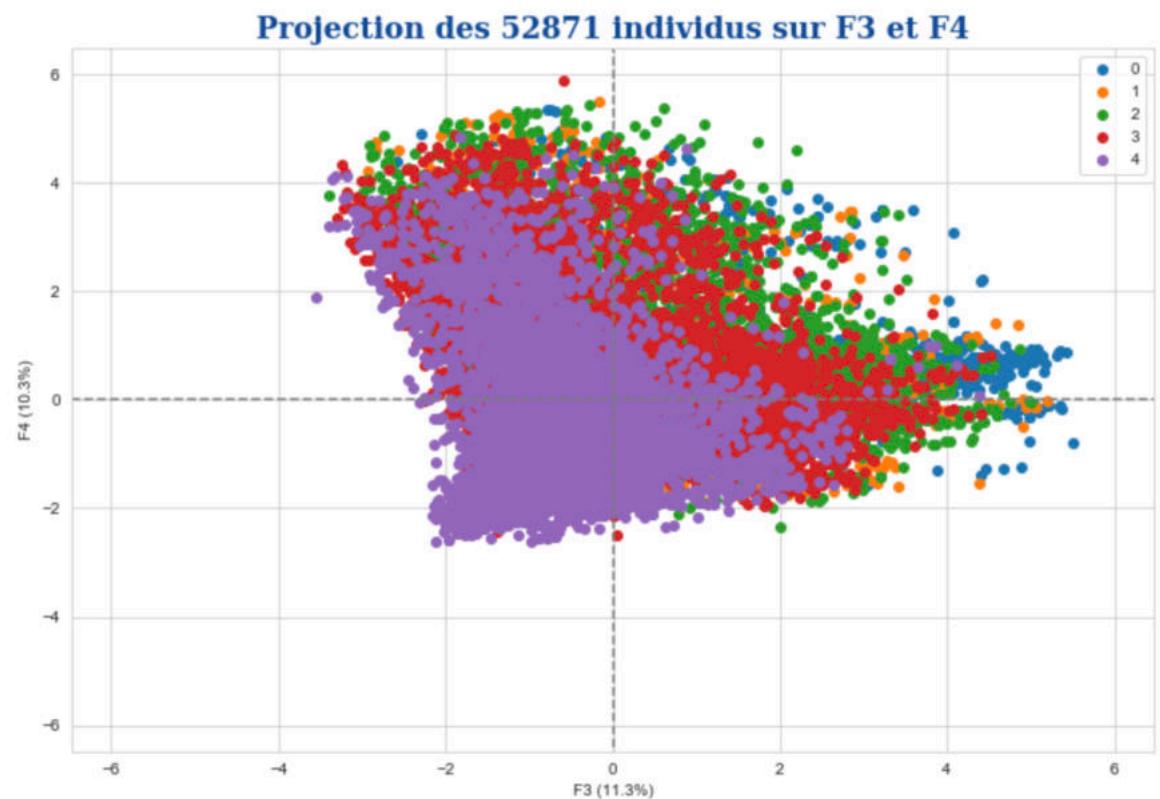
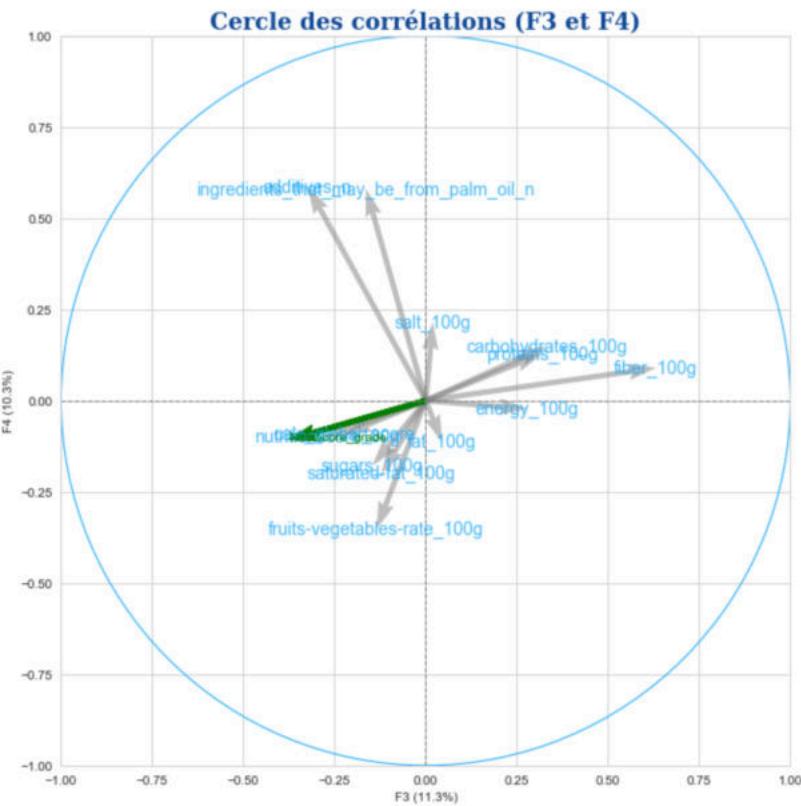
corr_matrix
```

	feature	CORR_F1	CORR_F2	CORR_F3	CORR_F4
0	additives_n	0.225334	-0.141628	-0.391551	0.677803
1	ingredients_that_may_be_from_palm_oil_n	0.173622	-0.195862	-0.198124	0.670388
2	energy_100g	0.881994	-0.050285	0.335994	-0.025880
3	fat_100g	0.828716	0.317257	0.054516	-0.131337
4	saturated-fat_100g	0.792638	0.262407	-0.146941	-0.230567
5	carbohydrates_100g	0.464139	-0.590859	0.401176	0.171609
6	sugars_100g	0.569885	-0.621176	-0.177233	-0.206208
7	fiber_100g	0.196078	-0.309393	0.764072	0.103371
8	proteins_100g	0.281873	0.678593	0.387959	0.148913
9	salt_100g	0.150456	0.653896	0.023040	0.247228
10	nutritionscore100g	0.858805	0.131762	-0.327336	-0.112814
11	fruits-vegetables-rate_100g	-0.432351	-0.166875	-0.163276	-0.406485
12	calc_global_score	0.886077	-0.165561	-0.270559	-0.107369

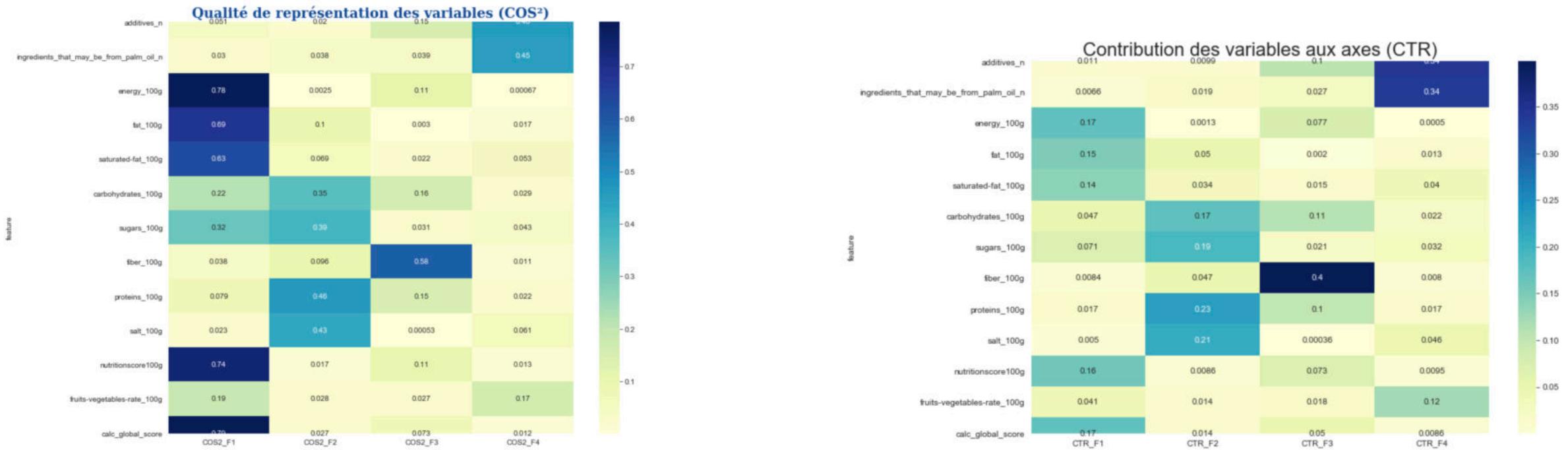
# F1 et F2



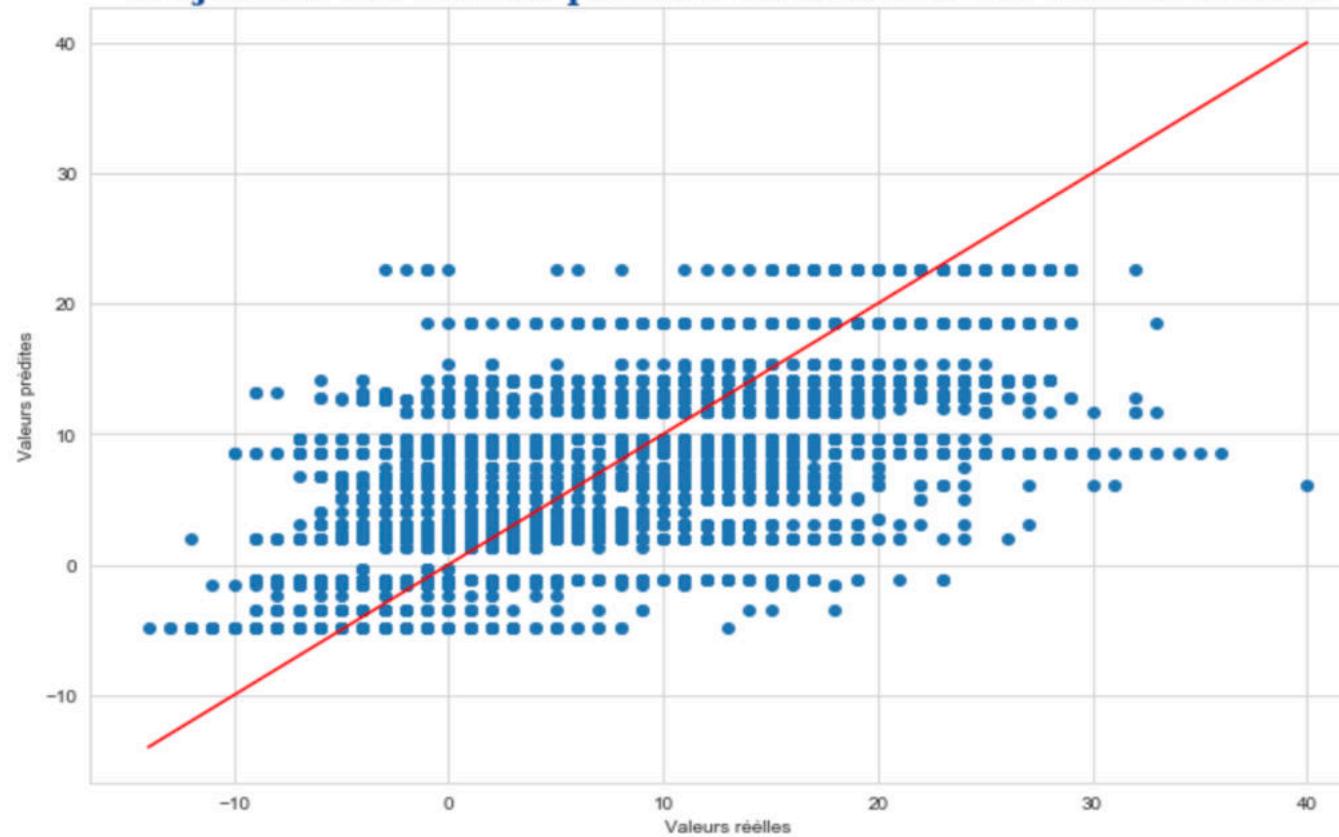
# F3 et F4



# Cercle,cos2 et contributions des variables aux axes



## Projection des valeurs prédictes en fonction des valeurs réélles



[55]:	Métrique	Baseline	LinearRegression	LinearRegression cat	LinearRegression PCA
0	MAE	7.844349	2.717468	2.107047	5.445194
1	MSE	82.083512	13.176506	7.641011	47.813385
2	RMSE	9.059995	3.629946	2.764238	6.914722
3	R <sup>2</sup>	-0.000042	0.839468	0.906908	0.417479

# Conclusion