

# Projet n°2 : Concevez une application au service de la santé publique

Soutenance du 30/09/21

The background features a series of vertical bars in red, yellow, green, and blue. Above the bars are stylized icons: a red building, a yellow wheat stalk, two yellow flowers, a green plant, and a black fish. The text 'Présentation de l'application' is centered in white.

# Présentation de l'application


OPEN FOOD FACTS

# Présentation de l'application **Glu'free**

L'application utile pour remplacer les produits qui contiennent en général du gluten



L'utilisateur entre un nom de produit qu'il souhaite substituer.

 pâtes



L'application trouve tous les produits correspondants sans gluten et l'utilisateur a la possibilité de les trier par nutriscore, nova-group, teneur en sel ...



# Présentation de l'application Glu'free



L'utilisateur accède de à toutes les informations nutritionnelles du produit qu'il sélectionne.



**Nom du produit:**

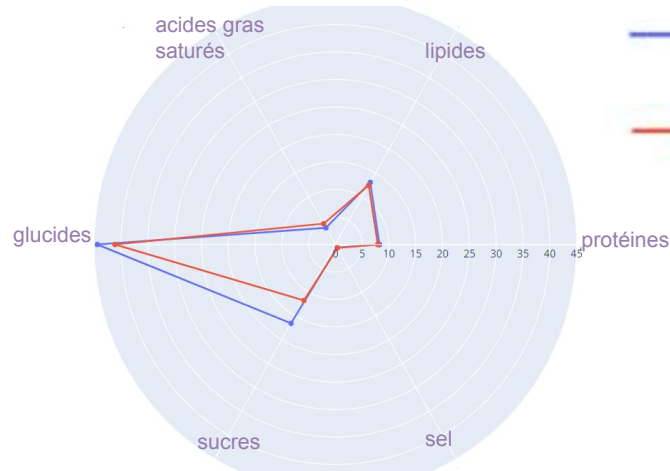
**Nutriscore:**

**Nova-goup:**

**Ecoscore:**

**Ingédients:**

## Valeurs nutritionnelles pour 100g



*Produit sélectionné*



*Valeurs nutritionnelles moyennes des pâtes avec gluten*

The background features a series of vertical bars of varying heights and colors: red, yellow, green, and blue. Above the bars are stylized food icons: a red cow, a yellow wheat stalk, two yellow flowers, a green plant with leaves, and a black fish. The text "Nettoyage du dataset" is centered in white.

Nettoyage du dataset

OPEN FOOD FACTS

# Nettoyage du dataset : première sélection

- Sur <https://world.openfoodfacts.org/data>, on a récupéré un dataset de 1 943 926 lignes et 186 colonnes

```
df=pd.read_pickle('en.openfoodfacts.org.products.pkl')
```

- On conserve les colonnes qui ont moins de 80 % de valeurs manquantes, il nous en reste 57.

```
#on conserve les colonnes qui possèdent moins de 80 % de valeurs manquantes  
mask=df.isna().mean()<0.8  
dg=df.loc[:,mask]
```

# Nettoyage du dataset : colonnes doublons

- Conversion des valeurs 'energy\_100g' (énergie en KJ) en kcal et sélection d'une seule colonne.

```
dg['energy-kcal_100g'].isna().mean()
```

```
0.23232880263960665
```

```
dg['energy_100g'].isna().mean()
```

```
0.20566317853663155
```

- On conserve la variable 'serving\_quantity' (type float64) par rapport à 'serving\_size' (type object)
- On conserve la variable brands\_tags par rapport à 'brands' qui est plus exploitable qui ne contient pas de caractères spéciaux, d'accents, et qui est en minuscule.



# Nettoyage du dataset: valeurs aberrantes

- Energie pour 100g :

```
max      8.693855e+12  
Name: energy-kcal_100g, dtype: float64
```

On supprime toutes les observations ayant une valeur supérieure à 1000 kcal pour 100g

```
#suppression des valeurs extrêmes de l'énergie pour 100g  
dg=dg[(dg['energy_kcal_100g']<=1000)&(dg['energy_kcal_100g']>=0)]
```



# Nettoyage du dataset: valeurs aberrantes

- Tailles de portions :

```
max      1.111111e+22  
Name: serving_quantity, dtype: float64
```

On supprime toutes les observations ayant une valeur supérieure 430g comme taille de portion.

```
dg=dg[(dg['serving_quantity']<=430)&(dg['serving_quantity']>=0)]
```

# Nettoyage du dataset: valeurs aberrantes

- Autres teneurs pour 100g

```
for c in ['proteins_100g', 'fat_100g', 'saturated-fat_100g', 'carbohydrates_100g', \
         'sugars_100g', 'salt_100g', 'sodium_100g', 'fiber_100g']:  
    dg=dg[(dg[c]<=100)&(dg[c]>=0)]
```

Avec suppression si la somme est >101 ( 1g d'erreur)

```
dg['sum_nutr']=dg['proteins_100g']+dg['fat_100g']+dg['fiber_100g']+dg['salt_100g']+dg['carbohydrates_100g']
```

```
dg=dg[dg['sum_nutr']<101]
```

- Pas d'autres valeurs aberrantes repérées pour les autres variables quantitatives et les variables catégorielles

# Nettoyage du dataset: Suppression des doublons

Le code EAN-13 identifie les produits alimentaires de façon unique. Certains codes produits sont internes à l'entreprise\*. Nous allons donc enlever :

- Les doublons sur les codes à 13 chiffres
- Les doublons sur les codes de produits de même marque

Dans les deux cas, nous garderons la ligne où il y a le plus de champs renseignés.

\*exemple <https://fr.openfoodfacts.org/produit/25279269/creme-brulee-aldi>

# Nettoyage du dataset: Suppression des doublons

*#Définition de la longueur du code*

```
dg['len_code']=dg['code'].apply(lambda x: len(str(x)))
```

*#Définition du nombre de valeurs renseignées*

```
dg['nb_values']=dg.notna().sum(axis=1)
```

*#On ordonne par nombre de valeurs renseignées*

```
dg=dg.sort_values('nb_values',ascending=False)
```

*#On enlève les doublons si le code a 13 chiffres*

```
dg=dg[dg['len_code']==13].drop_duplicates('code',keep='first')
```

*#On enlève les doublons si les codes et les marques sont identiques*

```
dg=dg.sort_values('nb_values',ascending=False)
```

```
dg=dg.drop_duplicates(['code','brands_tags'],keep='first')
```

```
dg.reset_index(drop=True,inplace=True)
```

# Nettoyage du dataset: création de la variable gluten-free

```
def nogluten(x):  
    '''x est une chaîne de caractères  
    La fonction renvoie 1 si x contient une mention sans gluten, 0 sinon '''  
    if re.search('(free.\s{0,4}gluten|gluten.\s{0,4}free|(sans|no|without).\s{0,4}gluten)',str(x),re.IGNORECASE):  
        return True  
    else:  
        return False
```

```
dg['gluten_free']=dg['labels'].apply(nogluten)
```

Utilisation d'une regex dans la colonne labels

# Nettoyage du dataset: sélection finale

```
dg.columns
```

```
Index(['code', 'url', 'brands_tags', 'product_name', 'additives_n',  
      'ingredients_from_palm_oil_n', 'nutriscore_grade', 'nutriscore_score',  
      'nova_group', 'pnns_groups_1', 'pnns_groups_2', 'proteins_100g',  
      'fat_100g', 'saturated-fat_100g', 'carbohydrates_100g', 'sugars_100g',  
      'fiber_100g', 'salt_100g', 'sodium_100g', 'serving_quantity', 'labels',  
      'ecoscore_grade_fr', 'ecoscore_score_fr', 'energy_kcal_100g',  
      'gluten_free'],  
      dtype='object')
```

```
dg.shape
```

```
(108652, 25)
```

The background features a series of vertical bars of varying heights and colors (red, yellow, green, blue) against a black backdrop. Above the bars are stylized icons: a red cow, a yellow wheat stalk, two yellow flowers, a green plant with leaves, and a black fish silhouette. The text 'Analyses exploratoires' is centered in white.

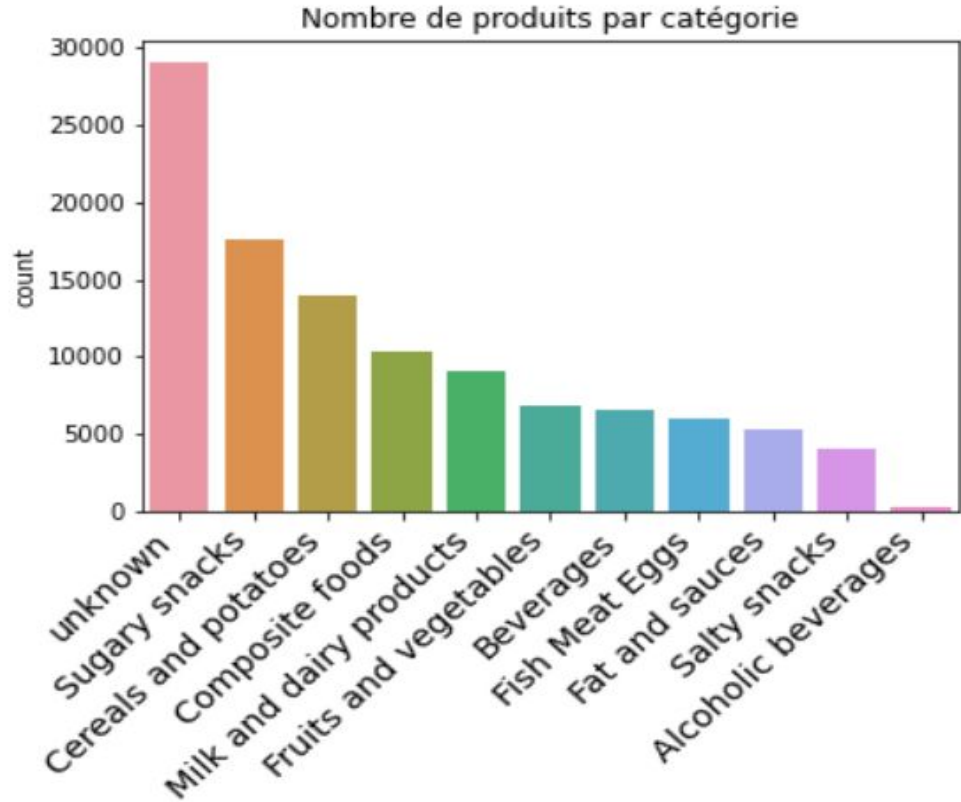
Analyses exploratoires

OPEN FOOD FACTS



# Catégories

27 % des produits n'ont pas de catégorie connue

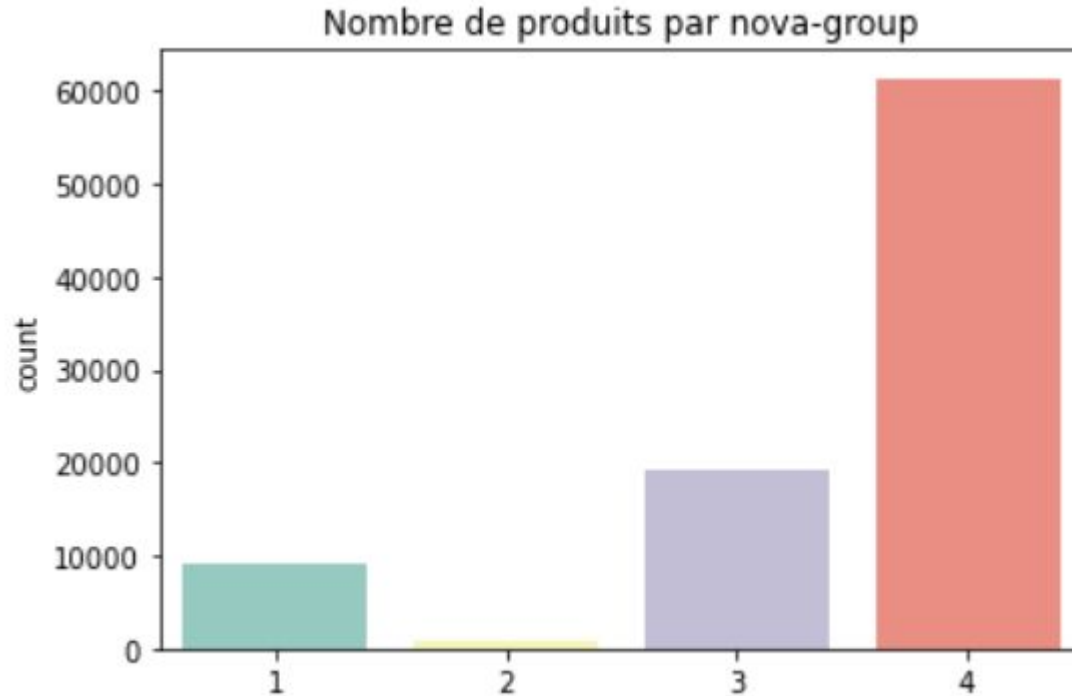


# Noms de produits

Mots apparaissant le plus dans les noms de produits :

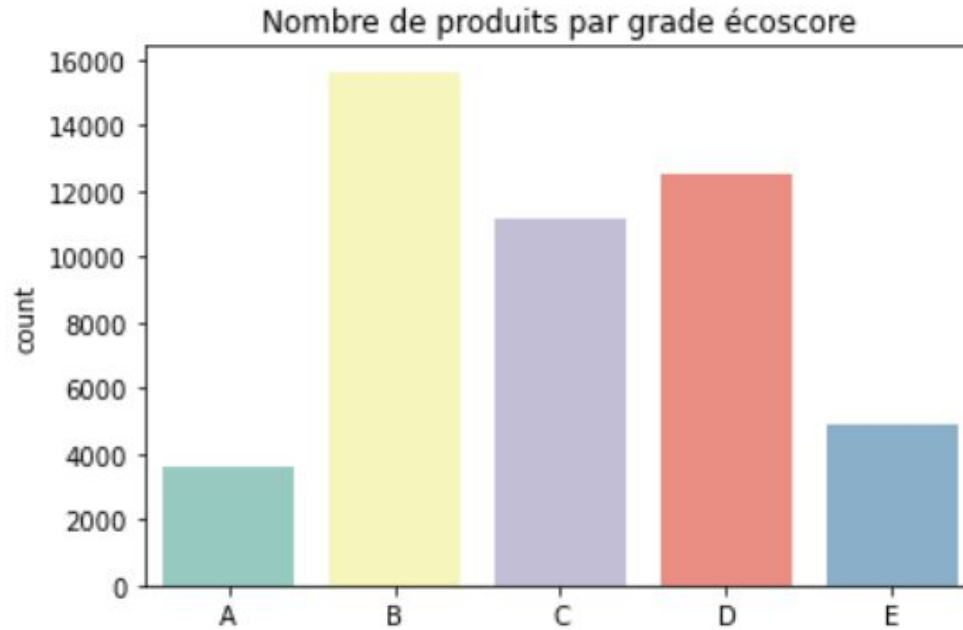
```
[('chocolate', 3943),  
 ('chocolat', 3441),  
 ('with', 2663),  
 ('sauce', 2627),  
 ('aux', 2565),  
 ('bio', 2400),  
 ('cheese', 2124),  
 ('organic', 1993),  
 ('lait', 1790),
```

# Nova-group



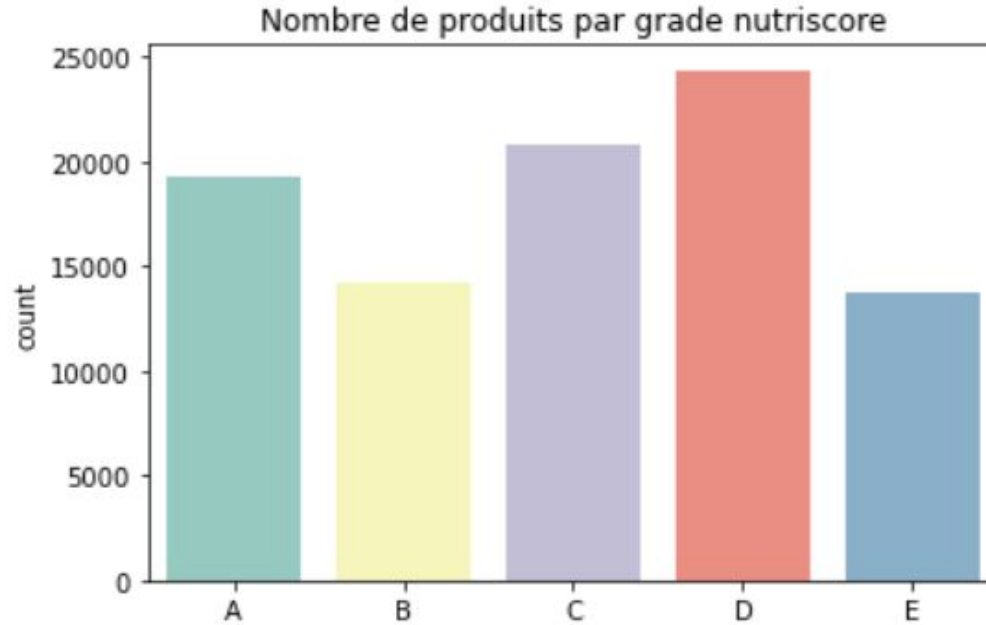
Majorité de produits ultras- transformés

# Ecoscores



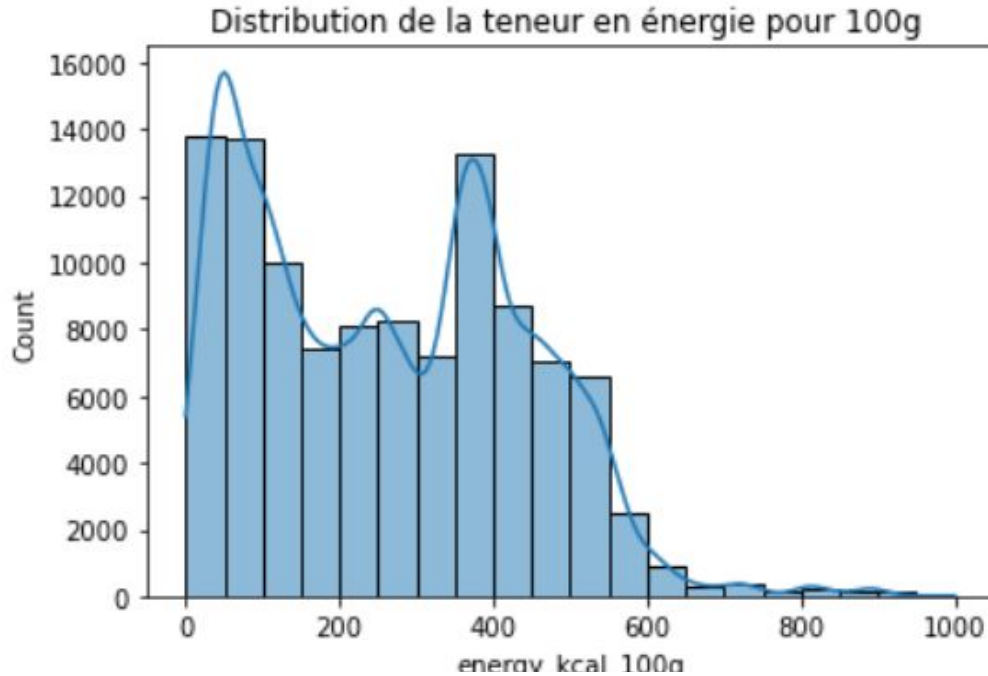
La catégorie B contient le plus grand effectif de produits

# Nutriscores



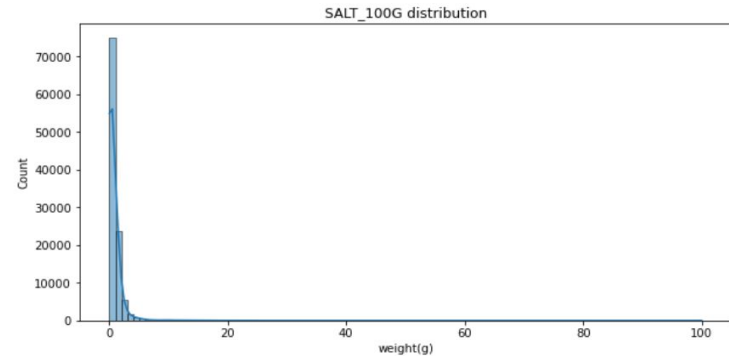
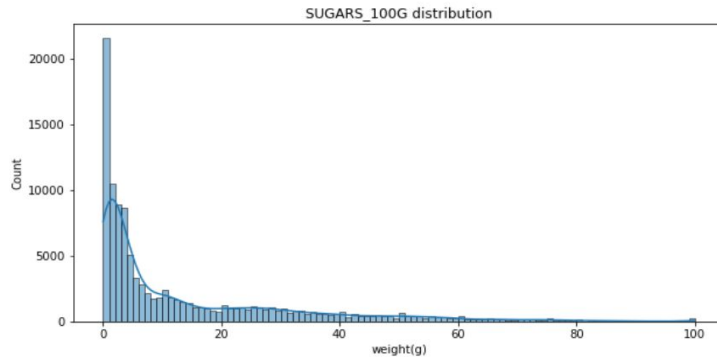
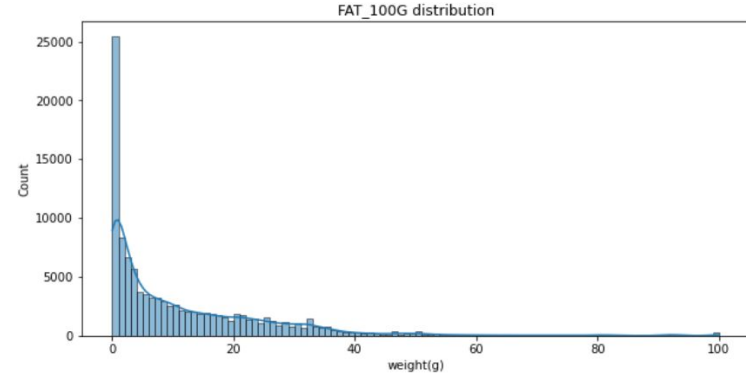
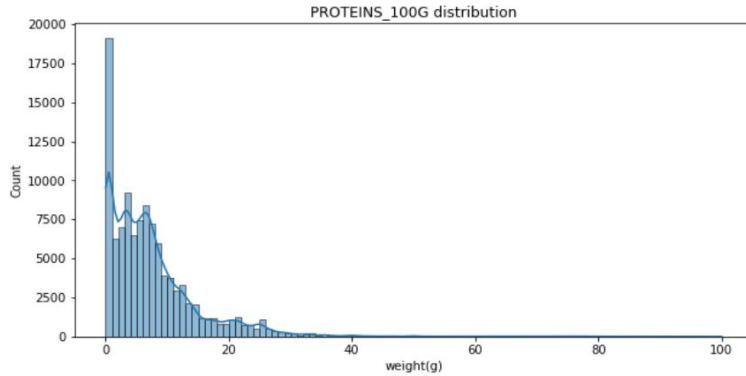
La catégorie D possède le plus grand effectif de produits

# Distribution de l'énergie pour 100g



Distribution bimodale

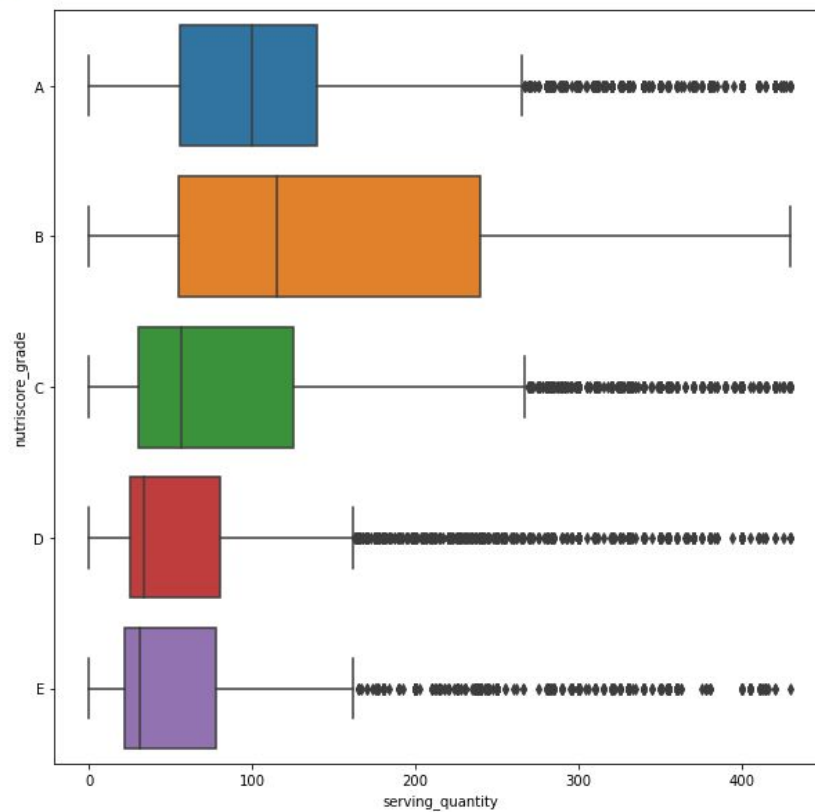
# Distributions d'autres valeurs nutritionnelles



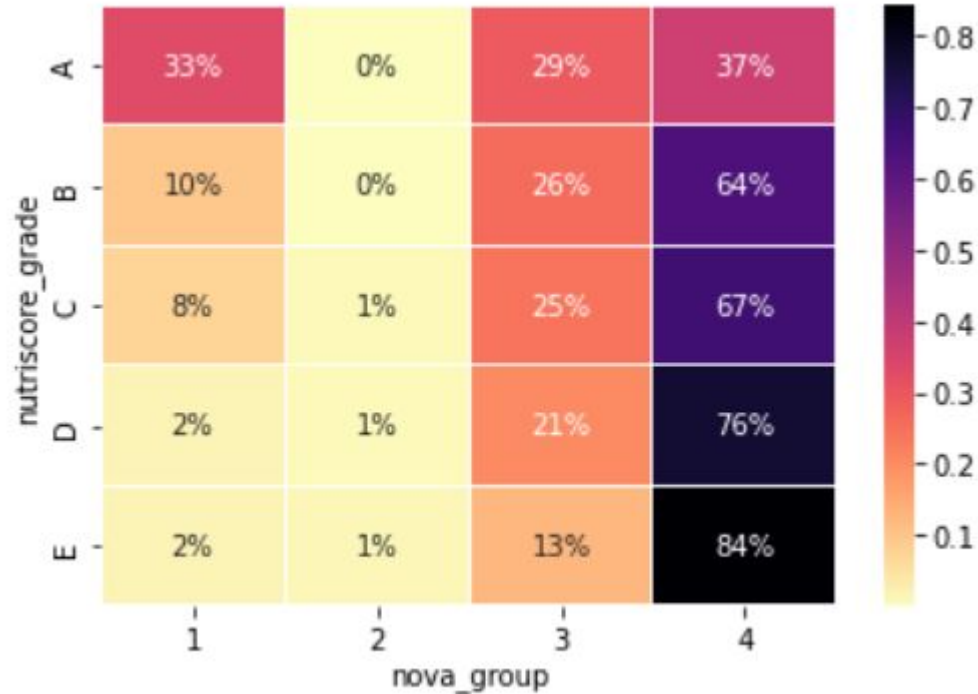
Distributions unimodales asymétriques, étalées à droite



# Nutriscores & tailles des portions

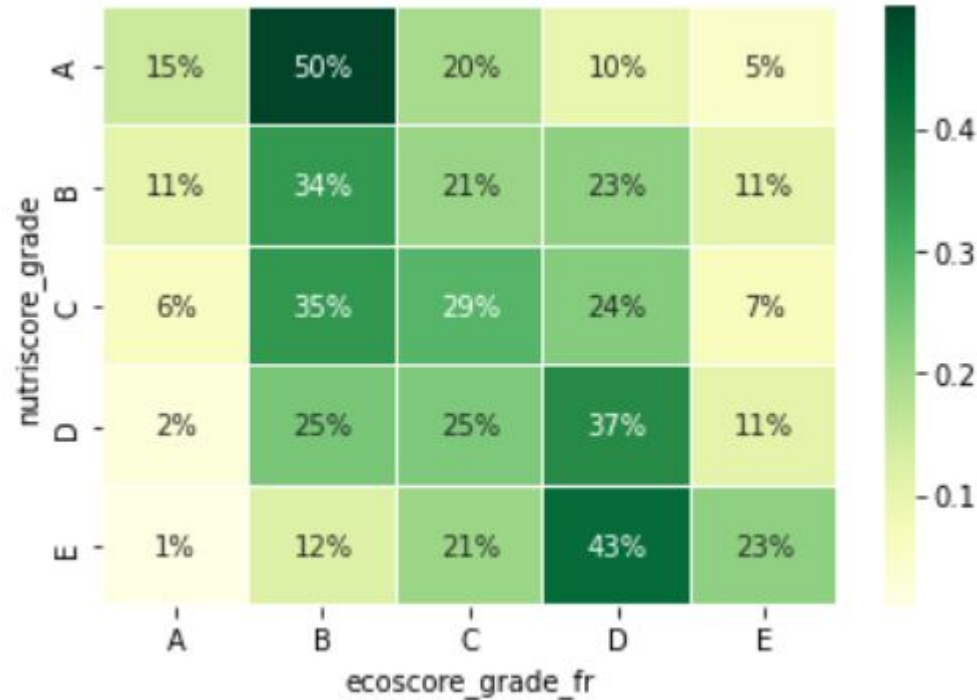


# Nutriscores & nova-group



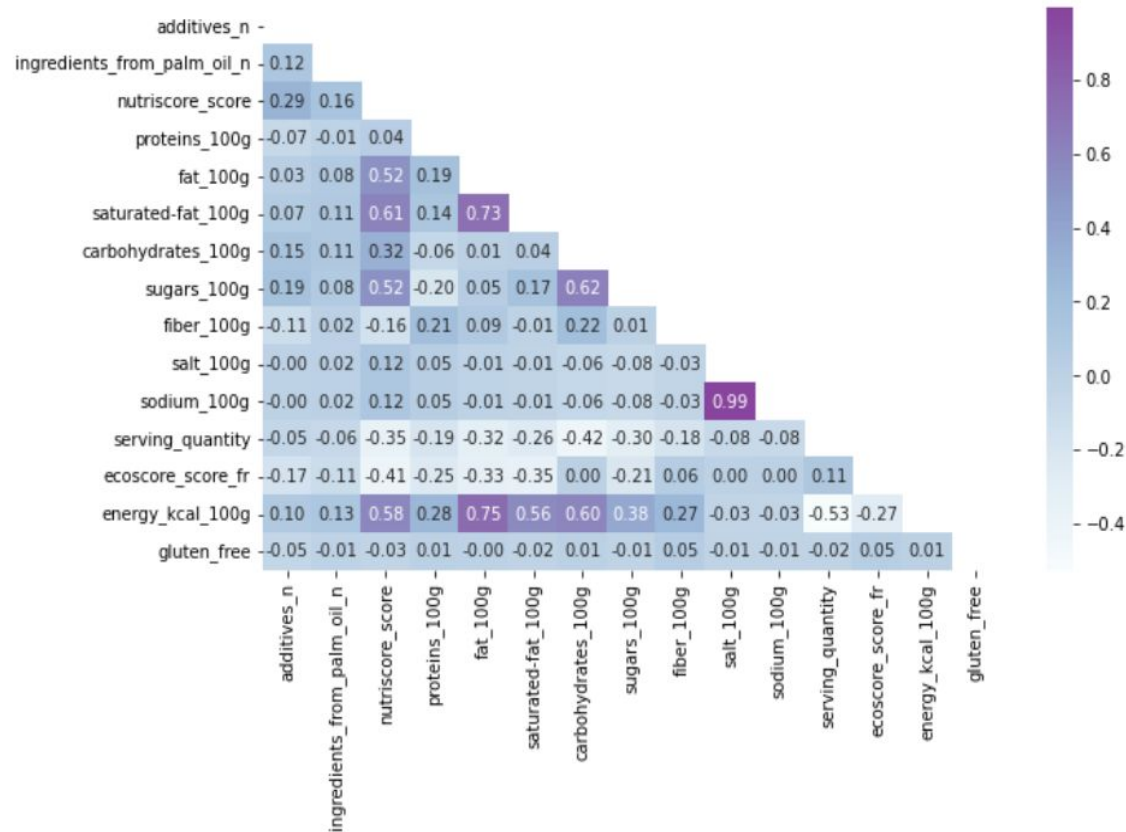
Lecture : Parmi les produits de nutriscore E, 84% sont ultra-transformés. Moins le nutriscore est sain, plus la proportion de produits ultra-transformés augmente.

# Nutriscores & écoscores



La tendance est que moins le nutriscore est sain, plus l'impact environnemental des produits augmente.

# Matrice de covariance



# Imputation du nutri-grade

On impute les 15 % de valeurs manquantes du nutri-grade grâce aux variables énergies, protéines, sel, glucides sucres fibres et lipides et à l'aide d'un modèle random-forest. On obtient une accuracy de 90%

## Distribution

avant imputation

```
dg_init.nutriscore_grade.value_counts(normalize=True)
```

```
D    0.263419  
C    0.225030  
A    0.208748  
B    0.153697  
E    0.149105
```

```
Name: nutriscore_grade, dtype: float64
```

après imputation

```
dg.nutriscore_grade.value_counts(normalize=True)
```

```
D    0.261808  
C    0.224524  
A    0.212716  
B    0.153785  
E    0.147167
```

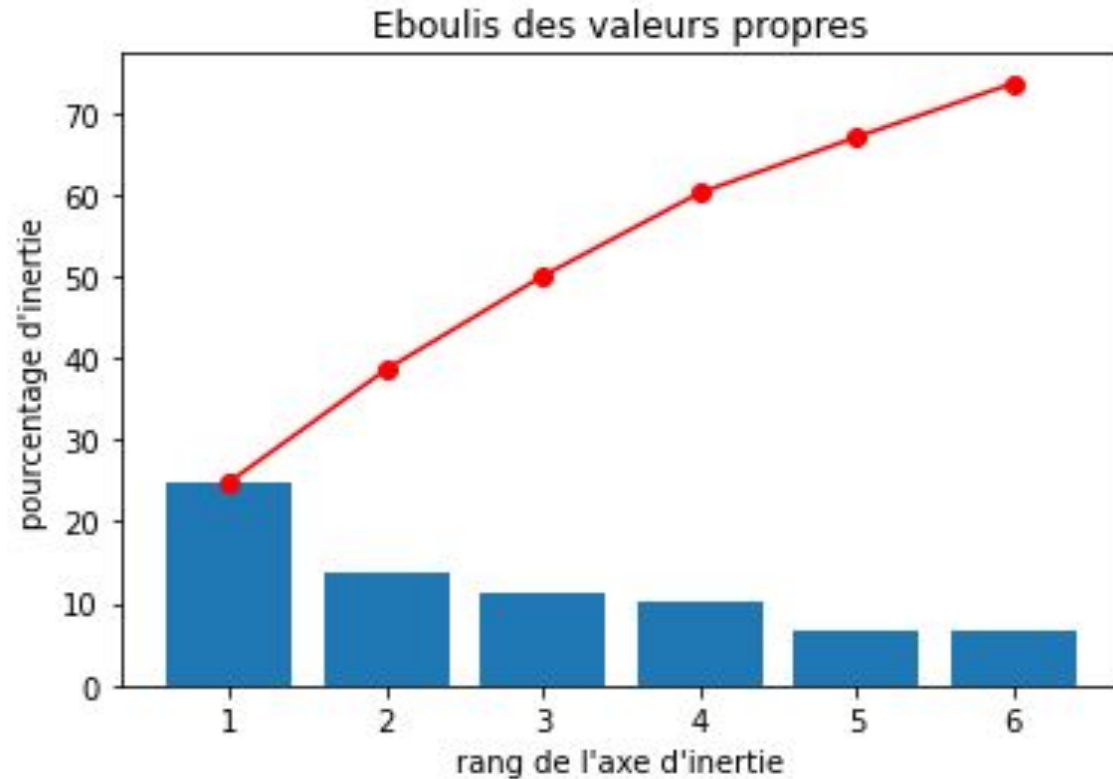
```
Name: nutriscore_grade, dtype: float64
```

# Imputation du nutriscore

On effectue l'imputation des points nutriscore en prenant la moyenne des points sur chaque grade :

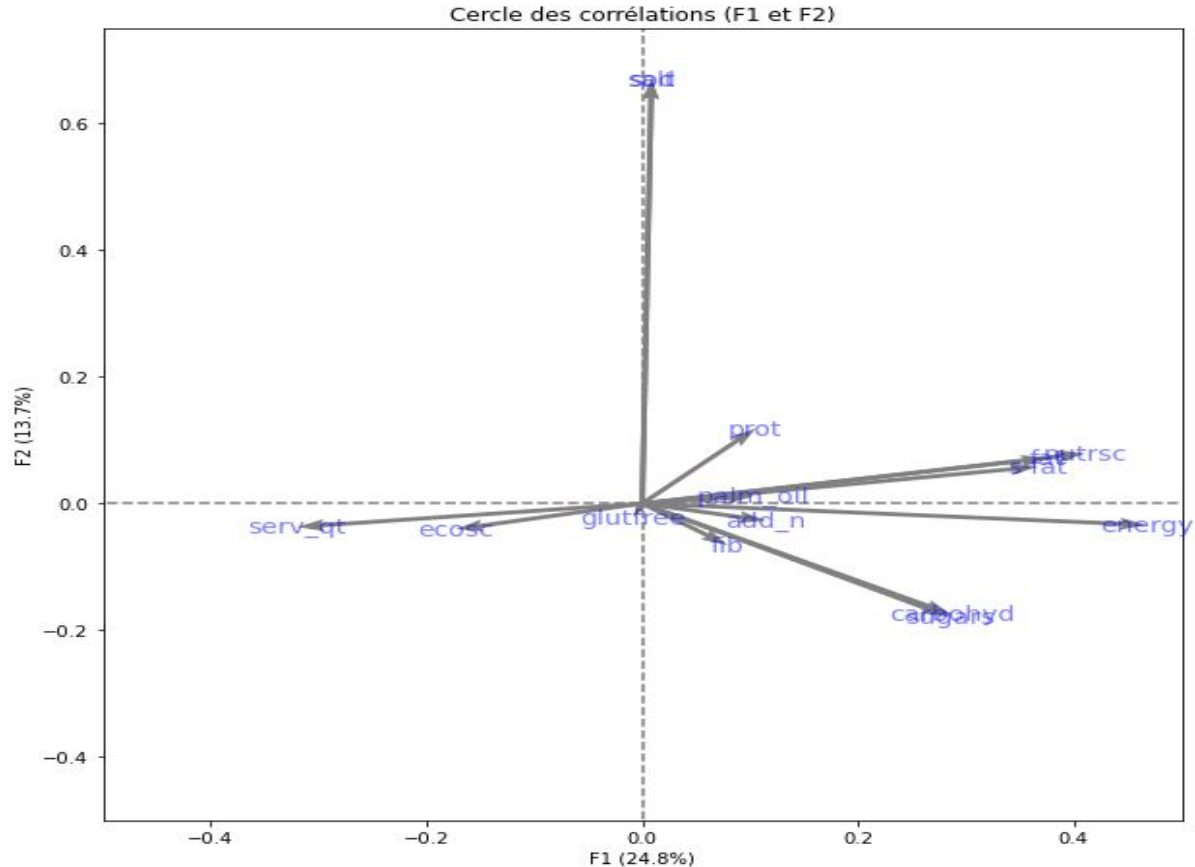
```
dg["nutriscore_score"] = dg.groupby("nutriscore_grade")["nutriscore_score"].transform(lambda x: x.fillna(x.mean()))
```

# Analyses en composantes principales

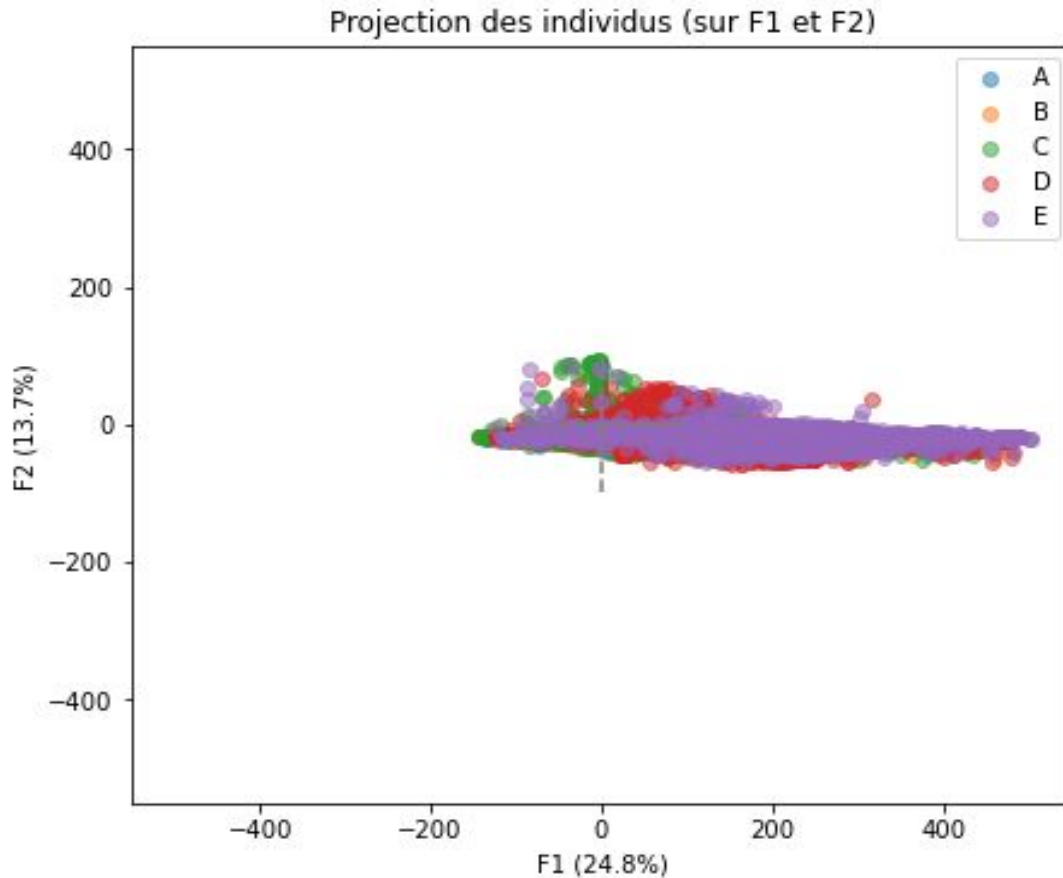




# ACP : projection des variables sur le 1er plan factoriel

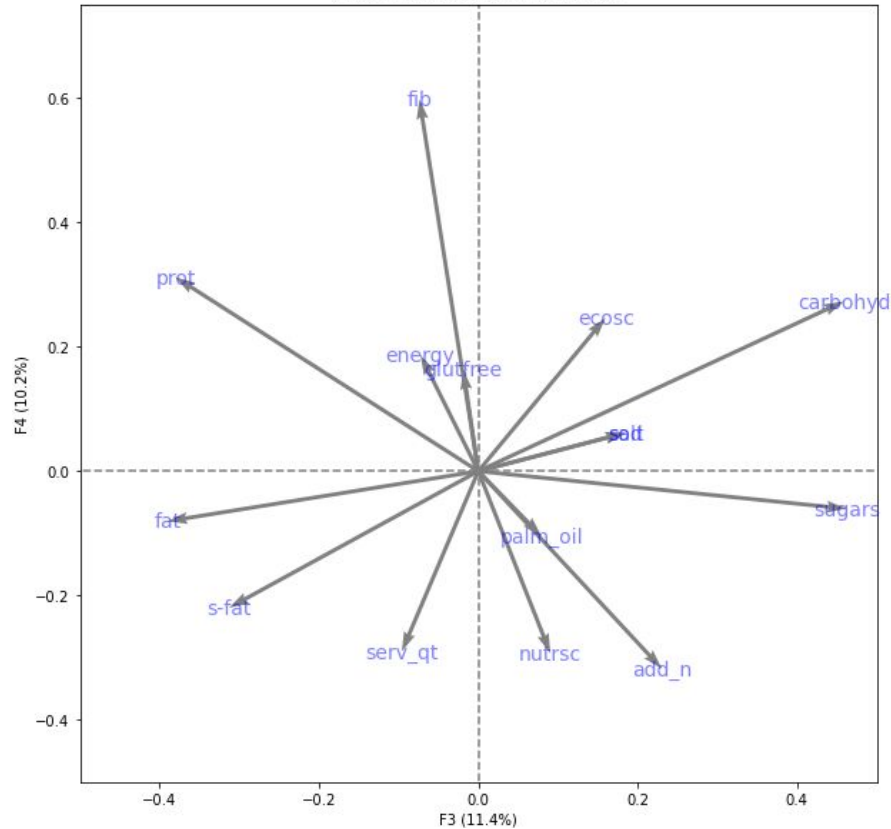


# ACP : projection des individus sur le 1er plan factoriel

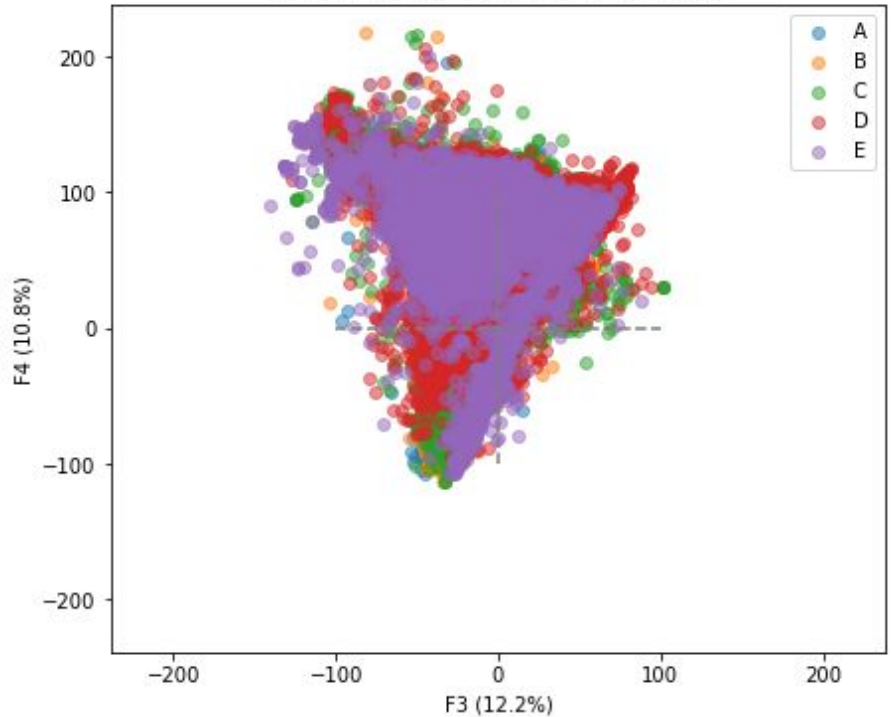


# ACP : Projections sur le 2e plan factoriel

Cercle des corrélations (F3 et F4)



Projection des individus (sur F3 et F4)

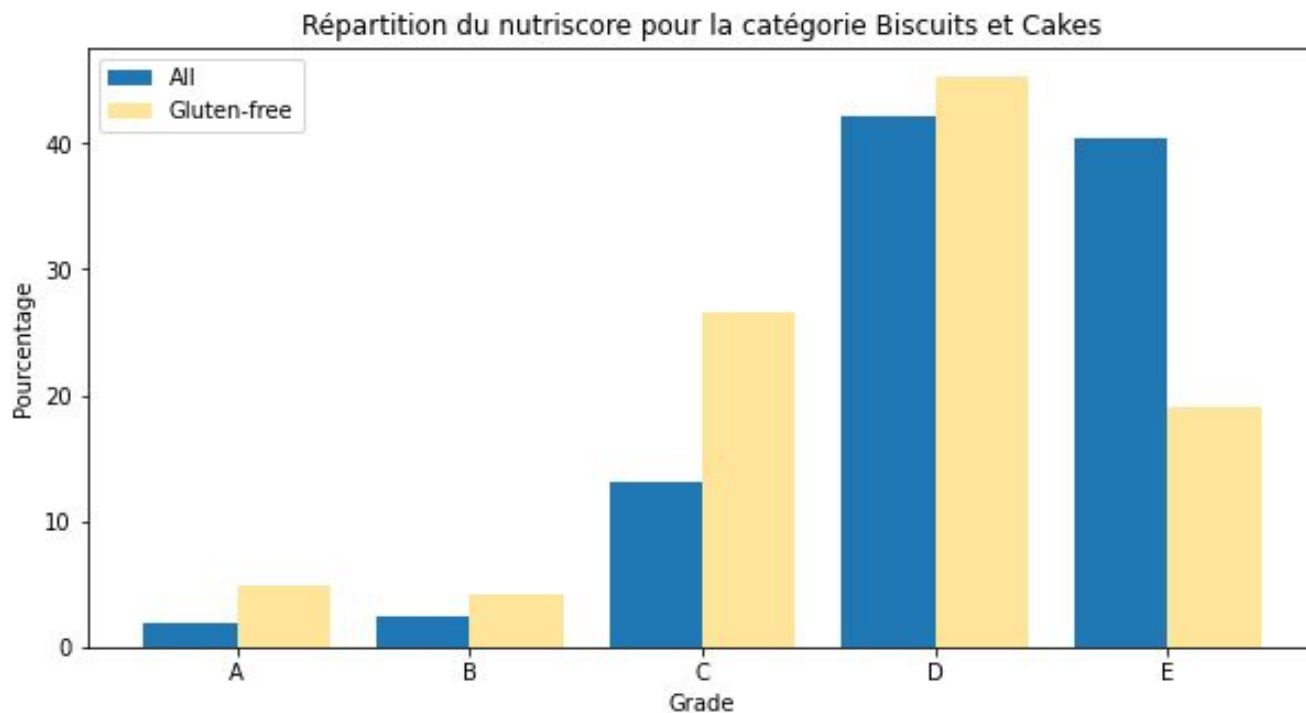


The background features a series of vertical bars in red, yellow, green, and blue. Above the bars are stylized icons: a red cow, a yellow wheat stalk, two yellow flowers, a green plant with leaves, and a black fish silhouette.

# Analyses des produits gluten-free

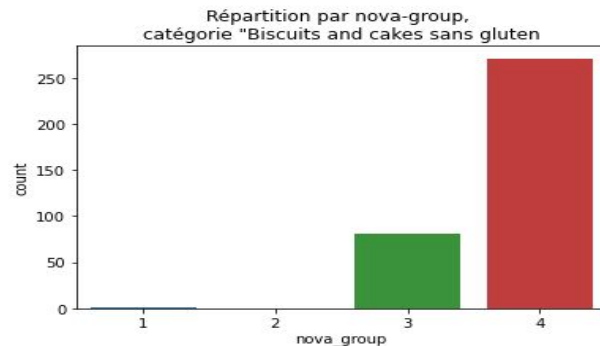
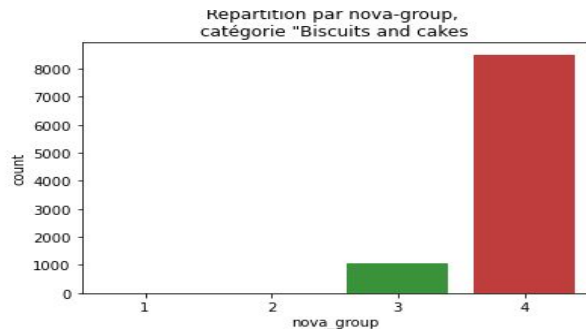
OPEN FOOD FACTS

# Catégorie biscuits and cakes



# Catégorie biscuits and cakes

Influence du label gluten free sur le **nova-group** ?



On réalise un test du chi-2. H0 "Parmi les produits de la catégorie "biscuits and cakes", le label gluten-free et le nova group sont indépendants"

p-valeur

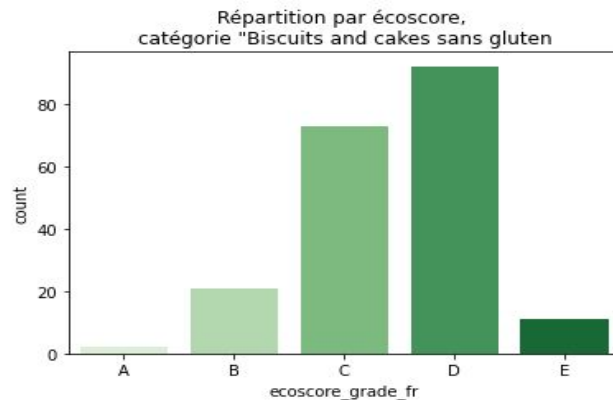
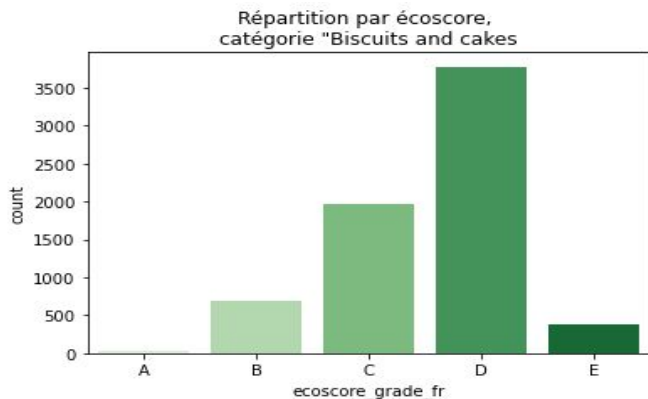
```
chi2_contingency(dh)[1]
```

4.379576119501591e-28

on rejette H0

# Catégorie biscuits and cakes

Influence du label gluten free sur l'**éco-score**?



On réalise un test du chi-2.

H0 "Parmi les produits de la catégorie "biscuits and cakes", le label gluten-free et l'éco-score sont indépendants"

p-valeur

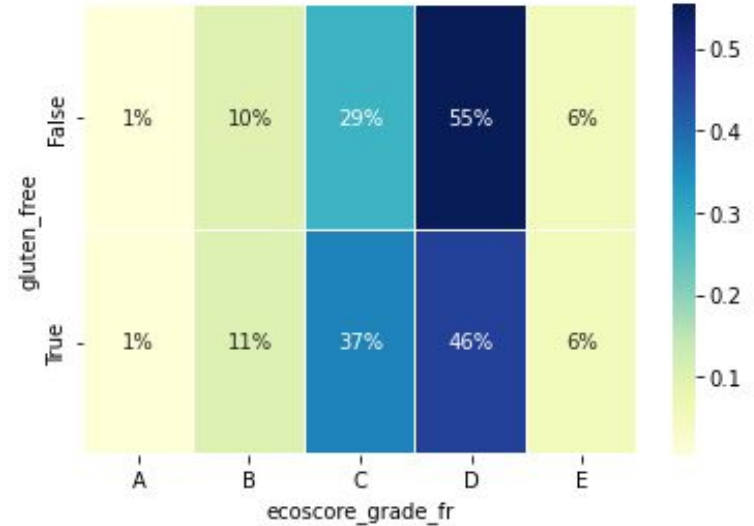
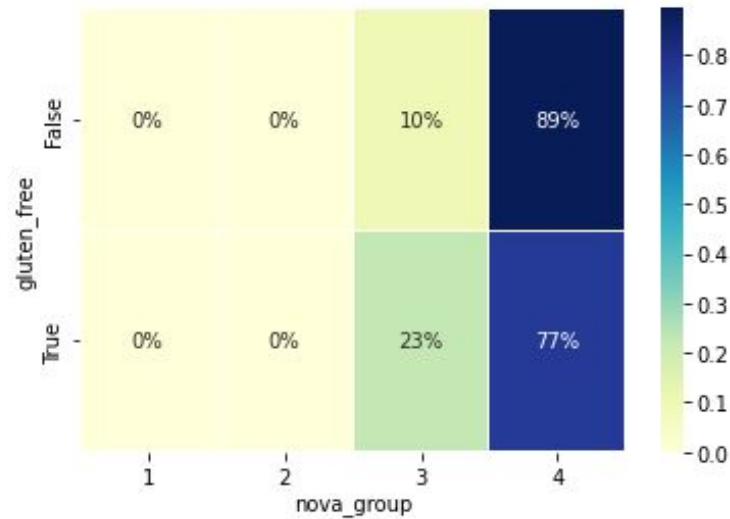
```
chi2_contingency(dh)[1]
```

0.08916072974777293

on accepte H0 au seuil 5%



# Catégorie biscuits and cakes



Les écarts sont un peu plus resserrés pour l'éco-score

# Répartitions moyennes par type d'aliments

graphiques interactifs

# Réflexions pour l'application

- Analyser la liste d'ingrédients pour repérer la totalité des produits sans gluten du dataset
- Améliorer la recherche par mot clé (si l'utilisateur mets "pâtes" alors tous les types de pâtes seront trouvés)