@Alhassane AHMED

WHATS INSIDE YOUR EVERYDAY

F O O D ?

...

AWARE ADDITIV'

BE CONSCIOUS OF WHAT U EAT

1. Introduction
2. Initial data processing
3. Final data processing
4. Correlation Analysis
5. Use cases
6. Conclusion

INTRODUCTION

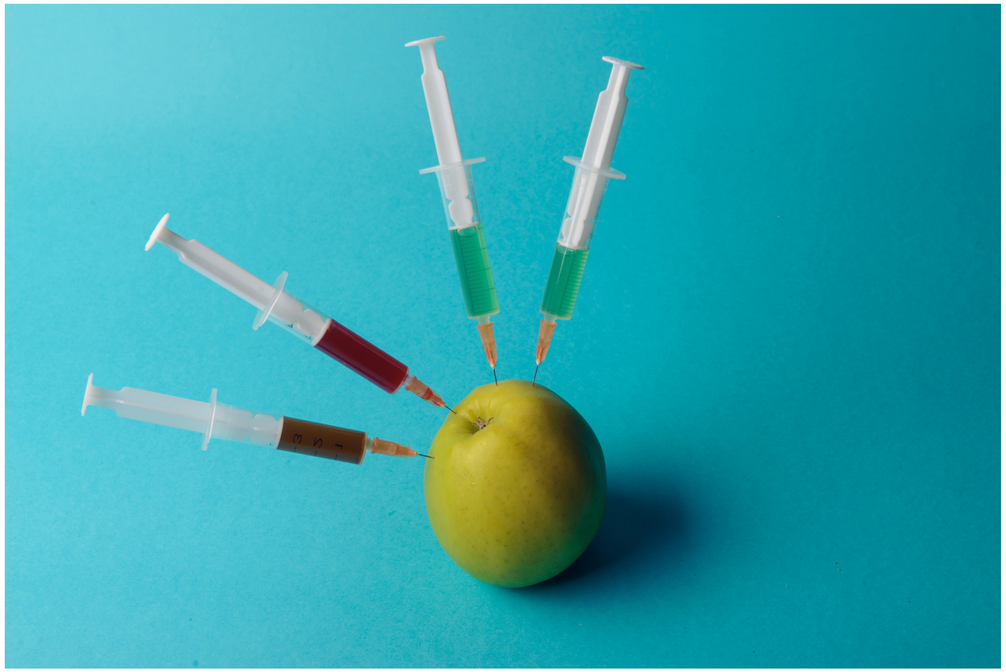
1. Origin
2. What's Aware Additiv' ?
3. What's an additive?

ONCE UPON A TIME....

open FOOD facts

IN DATA SCIENCE, **INSPIRATION COMES WITH EXPLORATION...**

WHAT IS AWARE ADDITIV'
?



- **Awareness tool** for supermarket food components that could be dangerous such as **additives**, and the risk they could probably present for health.

- **Recommandation tool** for the healthiest possible supermarket food based on additives.

WHAT IS AN ADDITIVE ?

Substance added directly to food during processing, as for preservation, coloring, or stabilization.

- **E100 : colorants**
- **E200 : conservateurs**
- **E300 : agents anti-oxygène**
- **E400 : agents de textures**

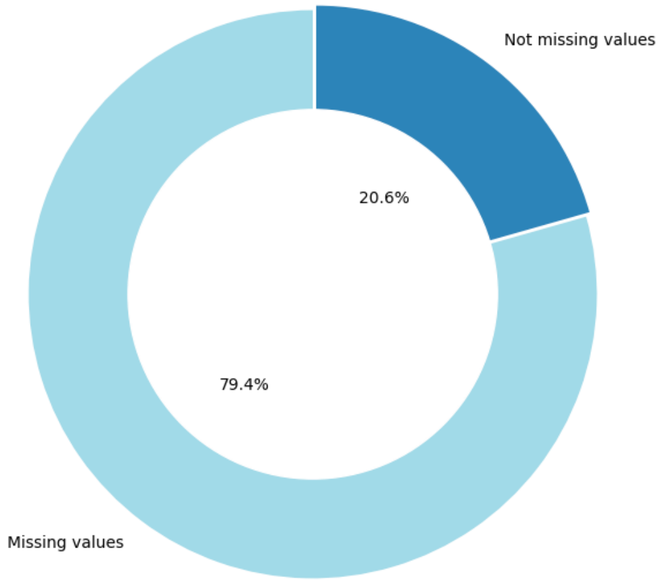
INITIAL DATA PROCESSING

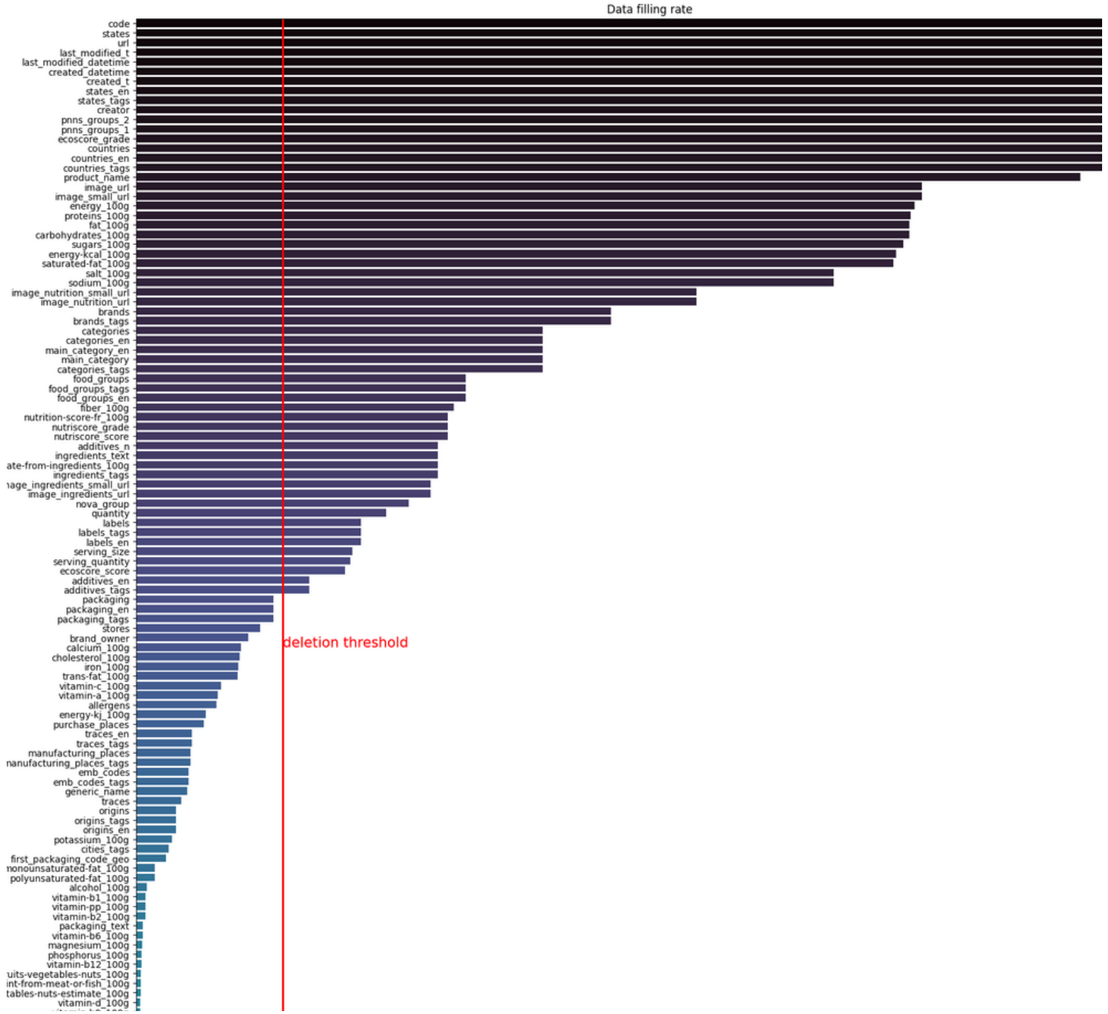
Explore and clean the initial dataset.

1. Data Summary
2. Toolbox
3. Exploring phase
4. Main Idea
5. Main cleaning actions
6. NaN Imputations

DATA SUMMARY

```
********************************** Data infos **********************************
Nombre de colonnes :  186

Nombre de lignes :  2561947


 ********************************** Valeurs manquantes ***************************
```






```
# features to keep
to_keep = filling_features[filling_features['nan_rate']<85]
```

TOOLBOX

```python
# this aims at returning a dataframe containing columns with a filling rate equal or greater than a specific threshold
def null_factor(df, rate_threshold=80):
    nan_rate = ((df.isnull().sum() / df.shape[0])*100).sort_values(ascending=False).reset_index()
    nan_rate.columns = ['variable','nan_rate']
    nan_rate2 = nan_rate[nan_rate['nan_rate'] >= rate_thresho
    return nan_rate2
```

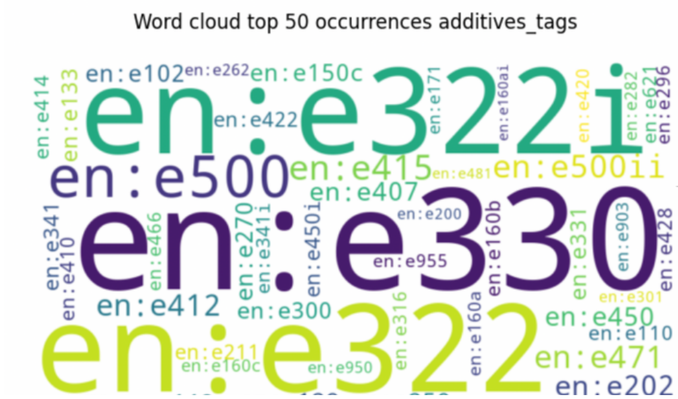
```python
def split_words(df, column = 'countries_en'):
    list_words = set()
    for word in df[column].str.split(','):
        if isinstance(word, float):
            continue
        list_words = set().union(word, list_words)
    return list_words
```

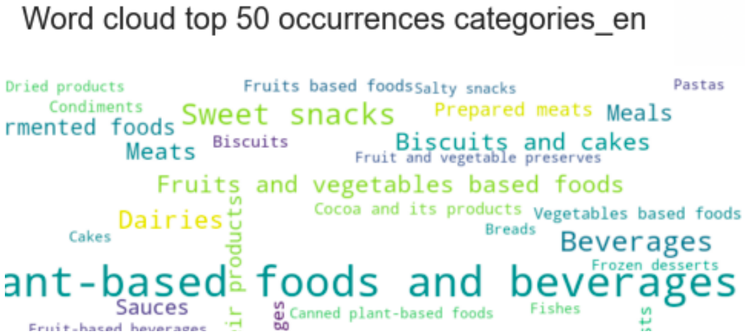
```python
def redundant_col(df):
    redundant_columns = []
    for col in df.columns:
        if "_en" in col:
            no_suffix = col.replace('_en','')
            tags = col.replace('_en','_tags')
            en = col.replace('_tags','_en')
            print("{:<20}|  no suffixe  : {}    |   suffixe _en : {}    |  suffixe _tags : {}".format(no_suffix,
no_suffix in df.columns, en in df.columns,tags

            if en in df.columns and no_suffix in df.columns:
                redundant_columns.append(no_suffix)
            if tags in df.columns and no_suffix or en in df.columns:
                redundant_columns.append(tags)

    return redundant_columns
```

```python
#this functiun aims at filtring data according to a suffix in column name
def search_componant(df, suffix):
    componants = [x for x in df.columns if suffix in x]
    return componants
```

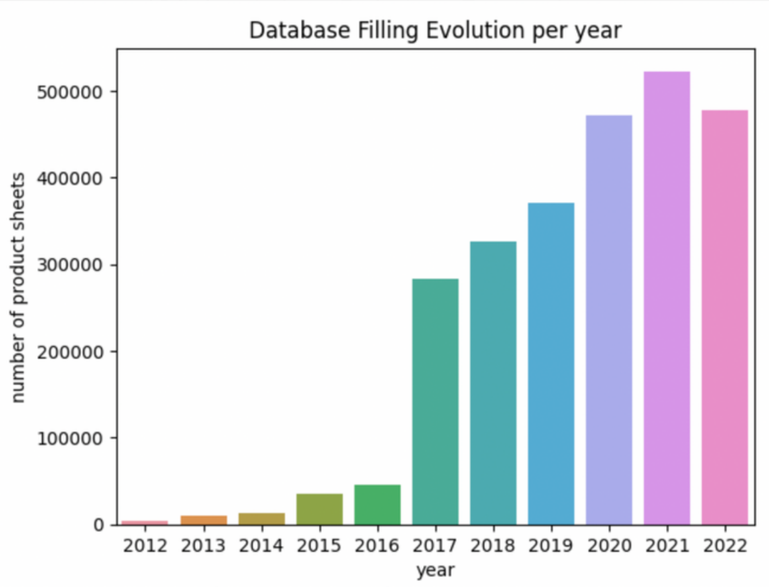
```python
def redundant_col(df):
    redundant_columns = []
    for col in df.columns:
        if "_en" in col:
            no_suffix = col.replace('_en','')
            tags = col.replace('_en','_tags')
            en = col.replace('_tags','_en')
```

EXPLORATION PHASE

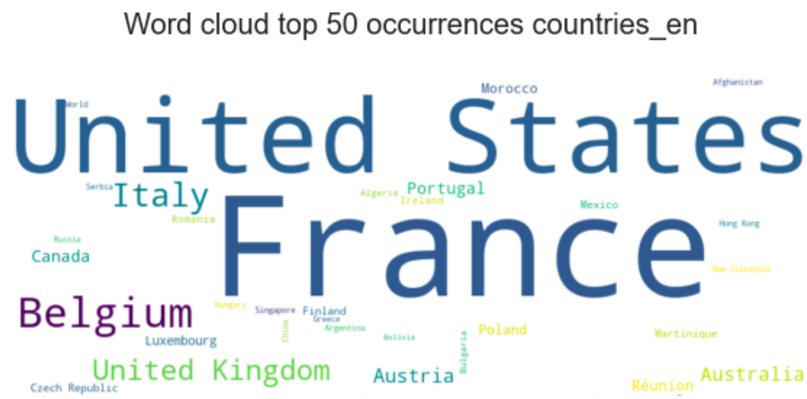


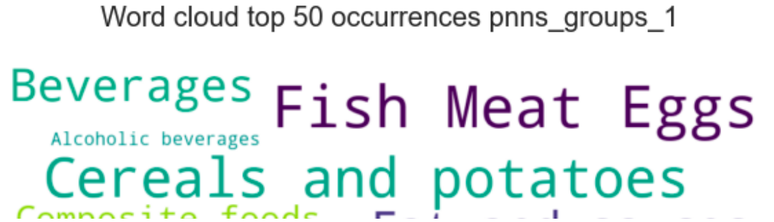



```
['0',
 '12',
 '19,99',
 '3-4',
 '9ш',
 'Afghanistan',
```




```
 'Afghanistan,France',
 'Afghanistan,France,French Guiana',
 'Afghanistan,France,Germany']

,'u200f\-فلسطين']
,'كل-البلدان'
,'ستانu200f\-كور'
,'ليبيا'
,'محمد-جمعه'
```



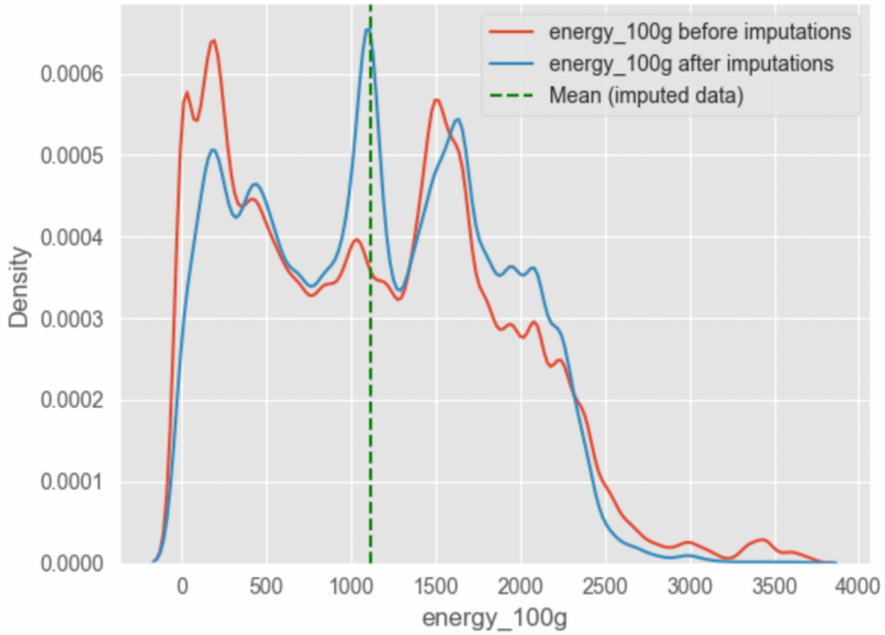


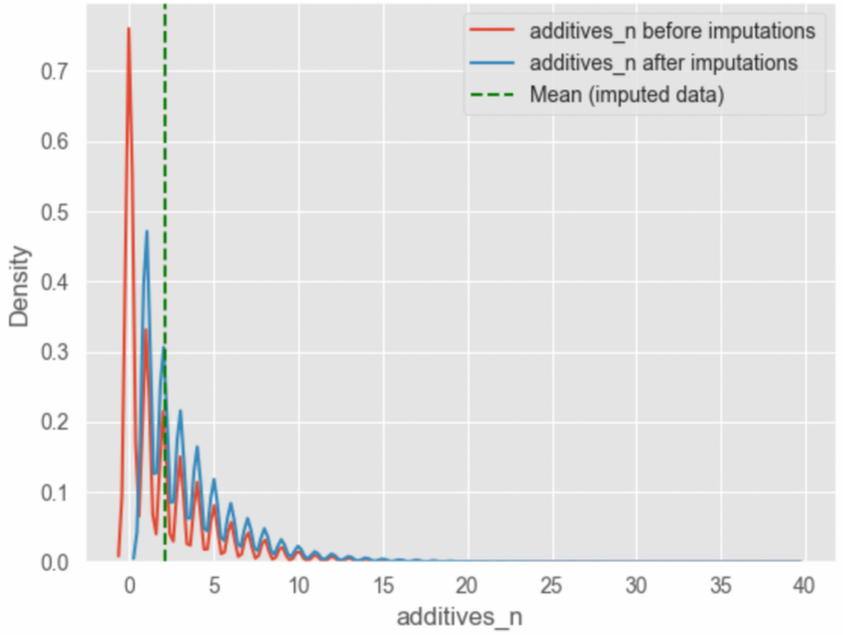
product_name	brands
울무차	Da Jung Co. Ltd
자연은 튼튼 (Jayeon-eun Teunteun)	자연은,Jayeon-eun
전복죽 Rice Porridge with Abalone	Bibigo
직화짜장	청정원
Deodorant	Old spice
burger sauce	***new***
🍇 Raisins sultanines	Rapunzel
🍚Riz au lait🥛	La Fermière
🐰 Lait du pays Alpin	Milka

NaN IMPUTING

```
datas_cleaned.corr()
```

	ecoscore_score	serving_quantity	nova_group	fruits-vegetables-nuts-estimate-from-ingredients_100g	additives_n	nutriscore_score	nutrition-score-fr_100g	fiber_100g	sodium_100g	salt_100g
sodium_100g	0.011883	0.000185	0.028614	-0.058954	-0.024272	0.128204	0.128203	-0.013828	1.000000	0.999648
salt_100g	0.011763	0.000185	0.028602	-0.058964	-0.024249	0.128335	0.128334	-0.013889	0.999648	1.000000





OTHER CLEANING ACTIONS

DUPLICATES

```
# delete duplicated rows based on code
data.drop_duplicates(subset ="code", keep = 'last', inplace=True)
```

```
# delete duplicated rows based on product name and brands at the same time
data = data[(~data.duplicated(["product_name","brands"],keep="last"))
      | ((data['product_name'].isnull()) & (data['brands'].isnull()))]
```

OUTLIERS

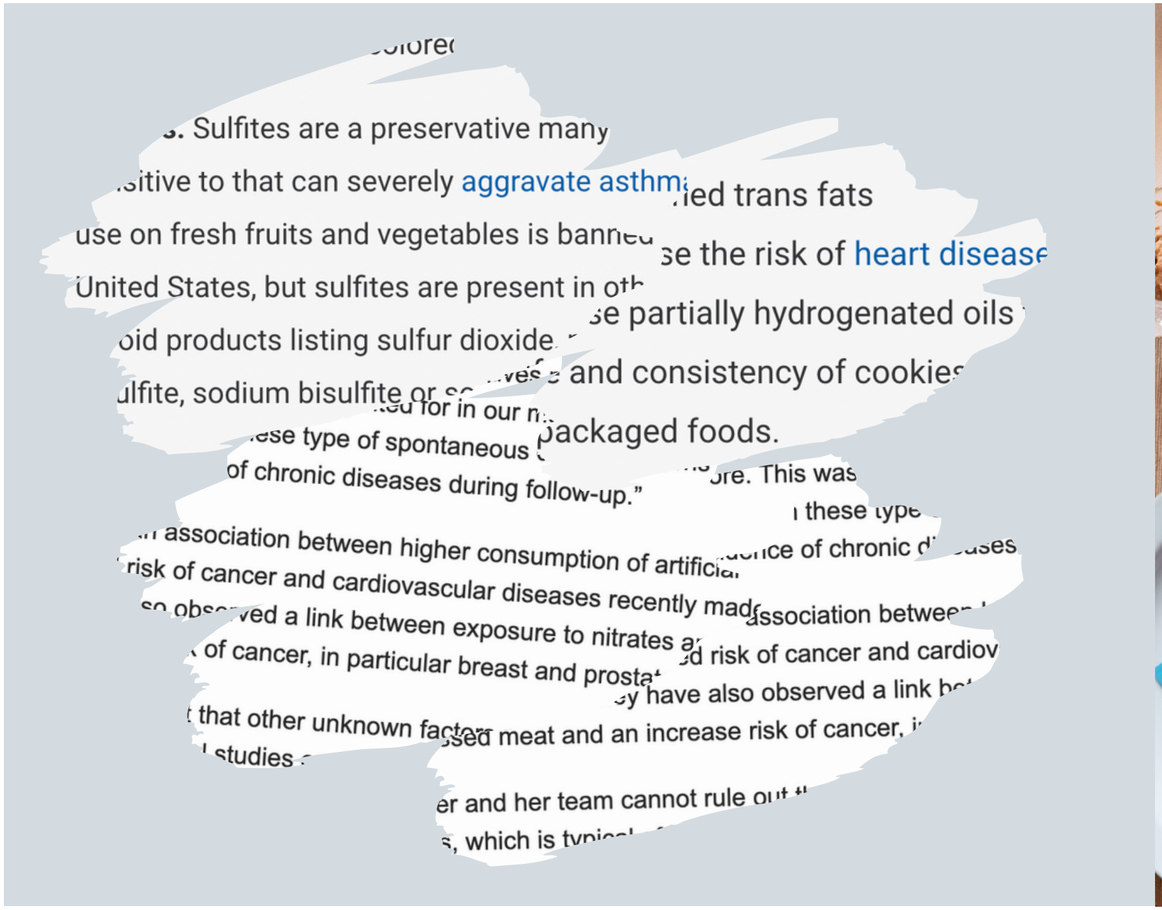
```
data3 = data2[~(data2[nutrients] < 0).any(axis=1)]
data3 = data3[~(data3[nutrients] > 100).any(axis=1)]

# acording to wikipedia : 3700 kj energy/100g => (https://en.wikipedia.org/wiki/Food_energy)
data3 = data3[~((data3['energy_100g'] > 3700) | (data3['energy-kcal_100g'] > 900))]
print('Total outliers deleted : ',len(data2)-len(data3))
```

MORE AND MORE

```
data["pnns_groups_1"] = data["pnns_groups_1"].str.replace('sugary-snacks', 'Sugary sna
for x in ['Pizza pies and quiche','Pizza pies and quichesssss','Pizza pies and quiches
    data["pnns_groups_2"] = data["pnns_groups_2"].str.replace(x,'Pizza pies and quiche
display(data["pnns_groups_1"].unique().tolist())
display(data["pnns_groups_2"].unique().tolist())
```

MAIN IDEA

ADDITIVES CAN BE DANGEROUS !



BE CONSCIOUS OF WHAT U EAT !

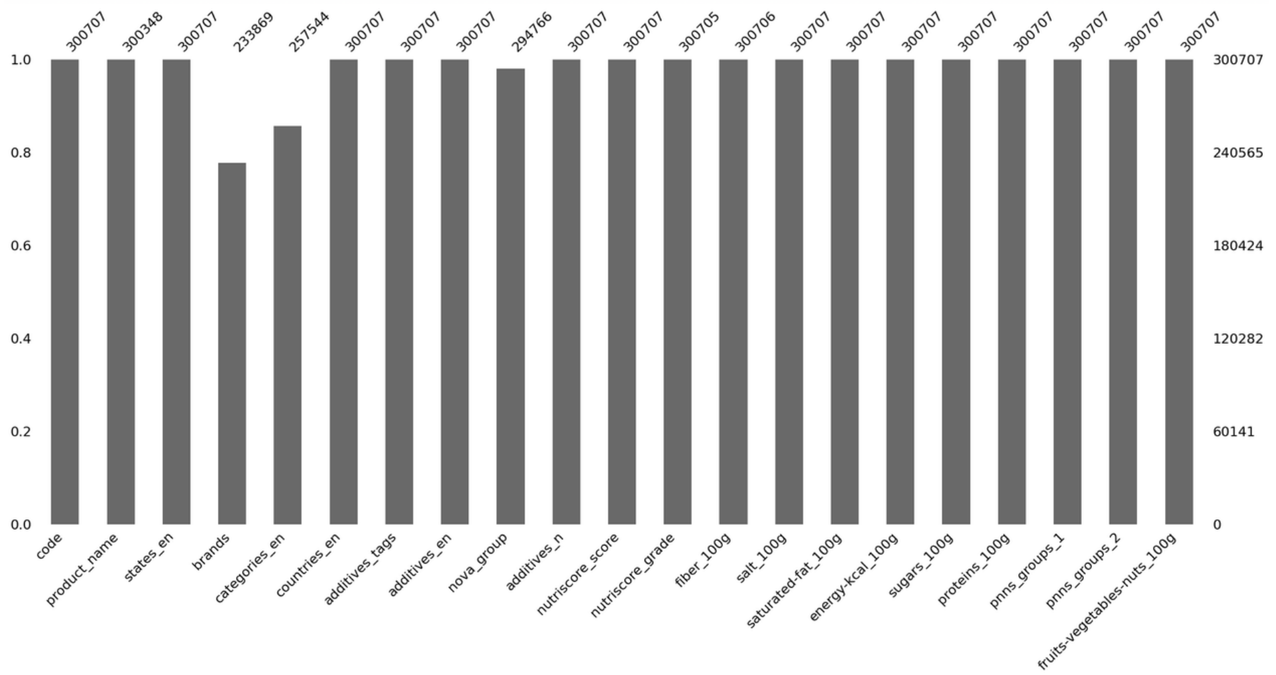
FINAL DATA PROCESSING

Explore the final dataset & analyze additives.

1. Data Summary
2. Toolbox
3. Univariate Analysis
4. Bivariate Analysis
5. Additives Analysis

DATA SUMMARY

```
********************************** Data infos **********************************
Nombre de colonnes :  21

Nombre de lignes :  300707

 ********************************** Valeurs manquantes ***************************
```




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300707 entries, 0 to 300706
Data columns (total 21 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   code                          300707 non-null  object
 1   product_name                  300348 non-null  object
 2   states_en                     300707 non-null  object
 3   brands                        233869 non-null  object
 4   categories_en                 257544 non-null  object
 5   countries_en                  300707 non-null  object
 6   additives_tags                300707 non-null  object
 7   additives_en                  300707 non-null  object
 8   nova_group                    294766 non-null  float64
 9   additives_n                   300707 non-null  float64
 10  nutriscore_score              300707 non-null  float64
 11  nutriscore_grade              300705 non-null  object
 12  fiber_100g                    300706 non-null  float64
 13  salt_100g                     300707 non-null  float64
 14  saturated-fat_100g            300707 non-null  float64
 15  energy-kcal_100g              300707 non-null  float64
 16  sugars_100g                   300707 non-null  float64
 17  proteins_100g                 300707 non-null  float64
 18  pnns_groups_1                 300707 non-null  object
 19  pnns_groups_2                 300707 non-null  object
 20  fruits-vegetables-nuts_100g   300707 non-null  float64
dtypes: float64(10), object(11)
memory usage: 48.2+ MB
```

TOOLBOX

```python
def chi2test(data, x, y, alpha=0.05):
    # Tableau de contingence
    cont = data[[x, y]].pivot_table(index=x, columns=y, aggfunc=len, margins=False)
    cont = cont.fillna(0)

    # Test du chi-2
    chi2, p, dof, expected = chi2_contingency(cont, alpha)

    print("=" * 100, "\n")
    print("""\t\t\t\t\t \033[1mTEST D'INDEPENDANCE DU CHI-2\033[0m \n""")
```

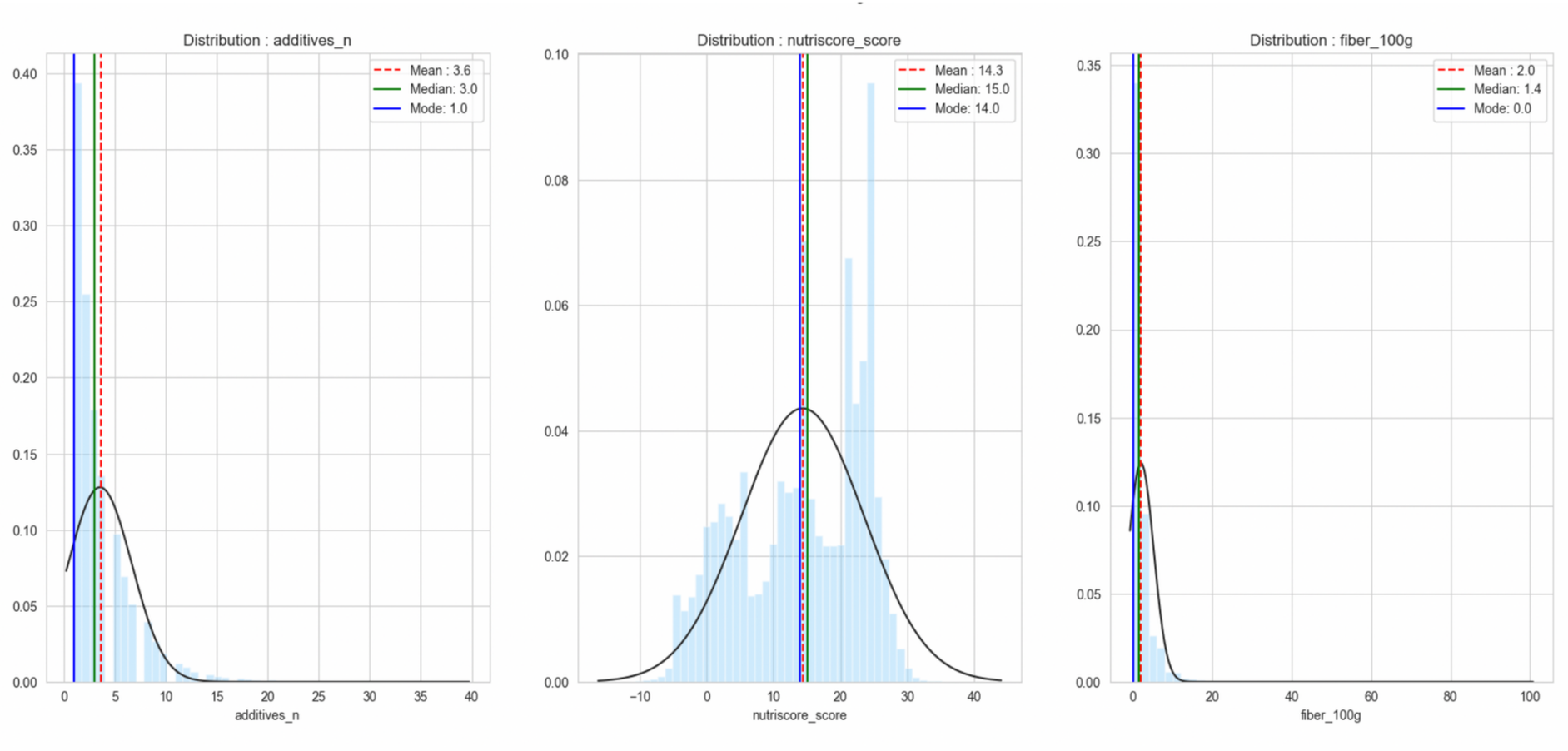
```python
def pearson_test(x, y, alpha=0.05):
    pvalue = st.pearsonr(x, y)[1]
    rs = st.pearsonr(x, y)[0]
    print("=" * 100, "\n")
    print("\t\t\t\t\t  \033[1mTEST D'
    print("=" * 100, "\n")
    print("""\t##### \033[1m0. Hypoth
    H0 : Les variables {0} sont indép
    H1 : Les variables {0} sont corré
    \t##### \033[1m1. Paramètre du te
    Variables aléatoires étudiées : \
    Indice de confiance : \033[1m{1}\
    Taille de l'échantillon : \033[1m
    \t #### \033[1m2. Résultat du tes
    coefficient de Spearman : \033[1m
    p-value associée au test de Spear
    \t #### \033[1m3. Conclusion du t

    if abs(rs) < .10:
        qual = 'négligeable (ou nulle
```

```python
def spearman_test(x, y, alpha=0.05):
    pvalue = st.spearmanr(x, y)[1]
    rs = st.spearmanr(x, y)[0]
    print("=" * 100, "\n")
    print("\t\t\t\t\t  \033[1mTEST D'INDEPE
    print("=" * 100, "\n")
    print("""\t##### \033[1m0. Hypothèse du
    H0 : Les variables {0} sont indépendant
    H1 : Les variables {0} sont corrélées\n
    \t##### \033[1m1. Paramètre du test\033
    Variables aléatoires étudiées : \033[1m
    Indice de confiance : \033[1m{1}\033[0m
    Taille de l'échantillon : \033[1m{2}\03
    \t #### \033[1m2. Résultat du test\033[
    coefficient de Spearman : \033[1m{3}\03
    p-value associée au test de Spearman : '
    \t #### \033[1m3. Conclusion du test\03

    if abs(rs) < .10:
        qual = 'négligeable (ou nulle)'
    elif abs(rs) < .20:
```

```python
def shapiro_test(x, alpha=0.05):
    x1, pval1 = shapiro(x)
    print("=" * 100, "\n")
    print("\t\t\t\t\t TEST DE LA NORMALITE (TE
    print("=" * 100, "\n")
    print("""
    \t##### \033[1m0. Hypothèse du test\033[0m
    H0 : \033[1m{0}\033[0m suit une loi normal
    H1 : \033[1m{0}\033[0m ne suit pas une loi
    \t##### \033[1m1. Paramètre du test de Sha
    Variable aléatoire étudiée : \033[1m{0}\03
    Indice de confiance : \033[1m{1}\033[0m\n
    Taille de l'échantillon : \033[1m{2}\033[0
    \t #### \033[1m2. Résultat du test\033[0m
    p-value de shapiro : \033[1m{3}\033[0m\n
    coefficient de shapiro : \033[1m{4}\033[0m
    \t #### \033[1m3. Conclusion du test\033[0
    if pval1 < alpha:
        print("L'hypothèse nulle est rejetée \
    else:
        print("On ne peut pas rejeter l'hypoth
    print()
    print("=" * 100, "\n")
```

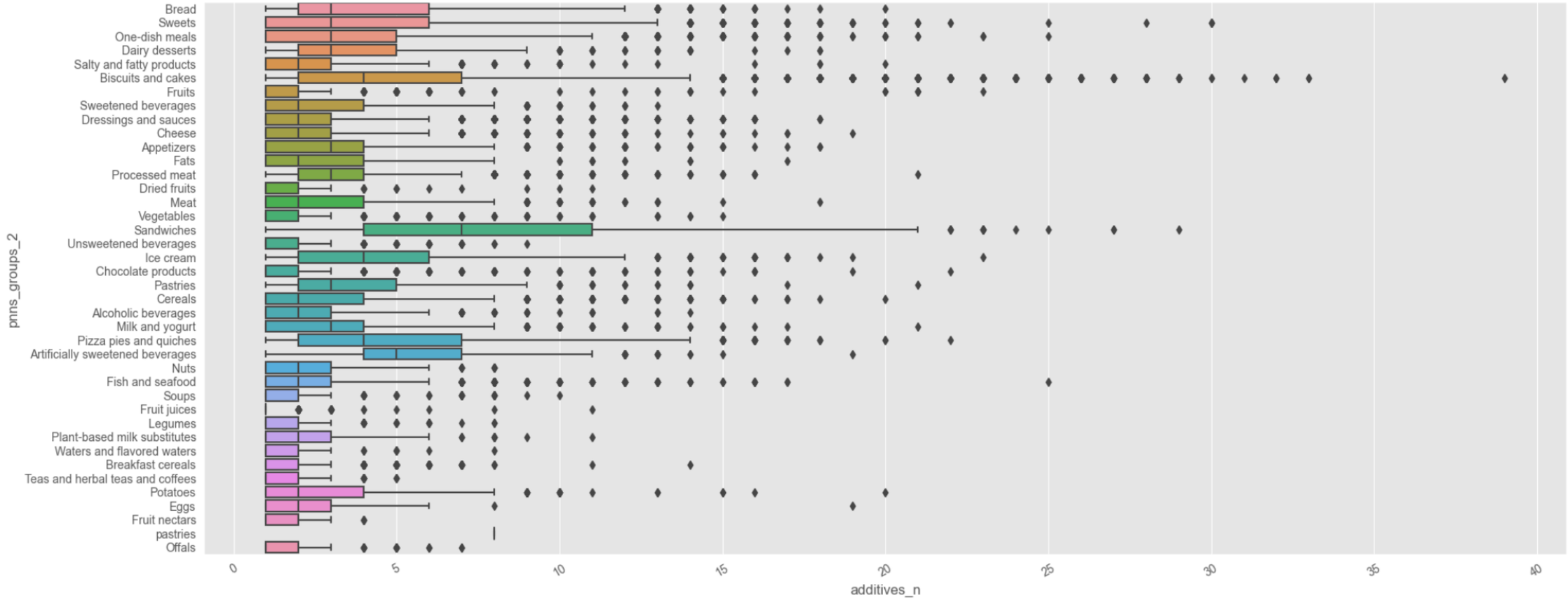
UNIVARIATE ANALYSIS

3 Numerical Values examples



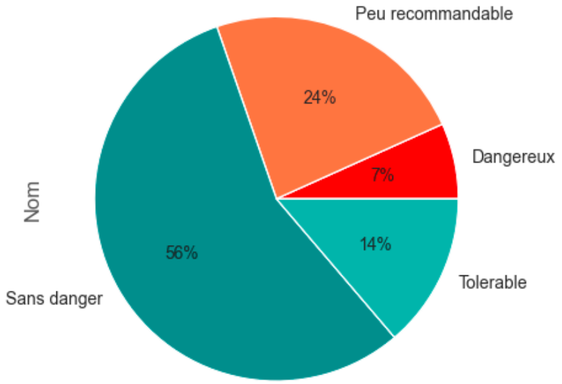
BIVARIATE ANALYSIS

pnns_groups vs additives_n example

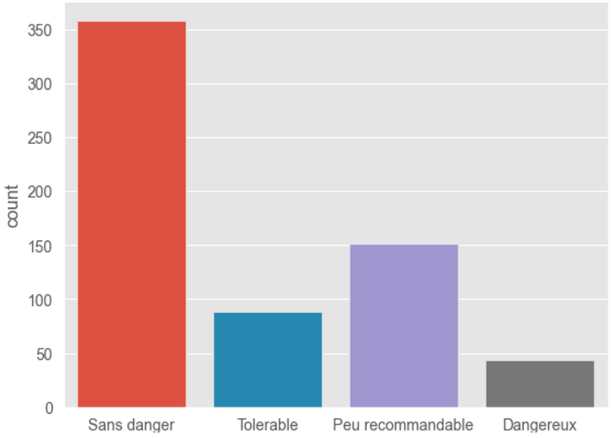


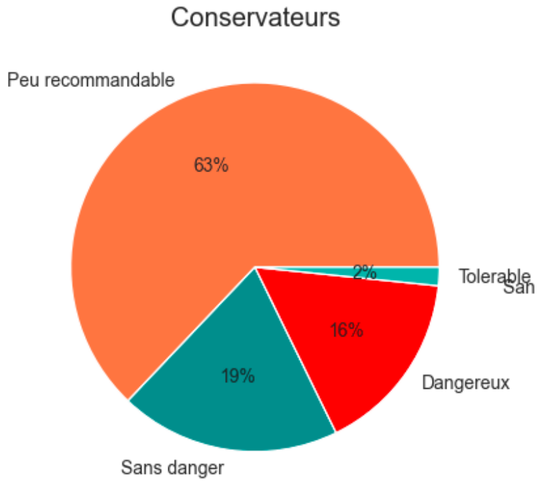
ADDITIVES ANALYSIS

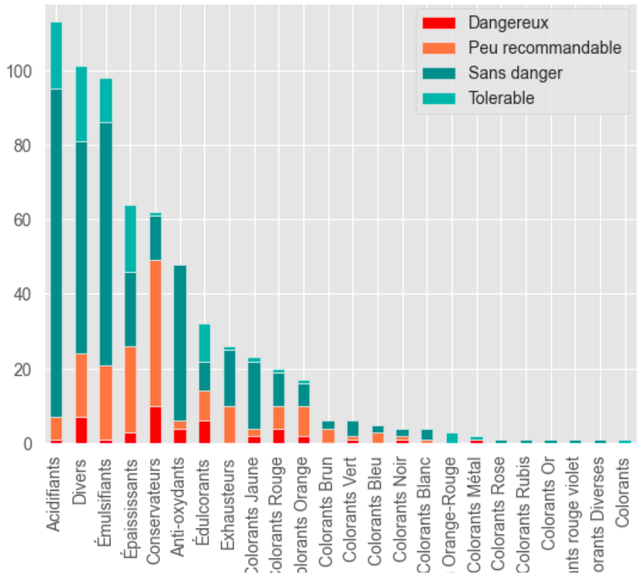
Source : https://www.les-additifs-alimentaires.com/liste-tous-danger.php

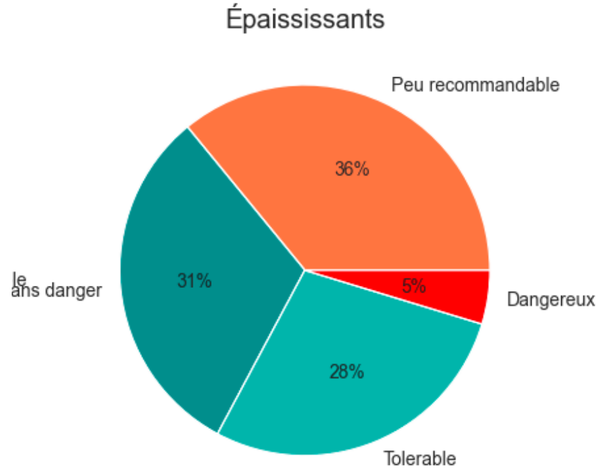


	Numero	Nom	Famille	Note
0	e100	Curcumine	Colorants Jaune	Sans danger
1	e100i	Curcumine	Colorants Jaune	Sans danger
2	e100ii	Curcuma	Colorants Jaune	Sans danger
3	e101	Vitamine G	Colorants Jaune	Sans danger
4	e101i	Riboflavin	Colorants Jaune	Sans danger









CORRELATION ANALYSIS

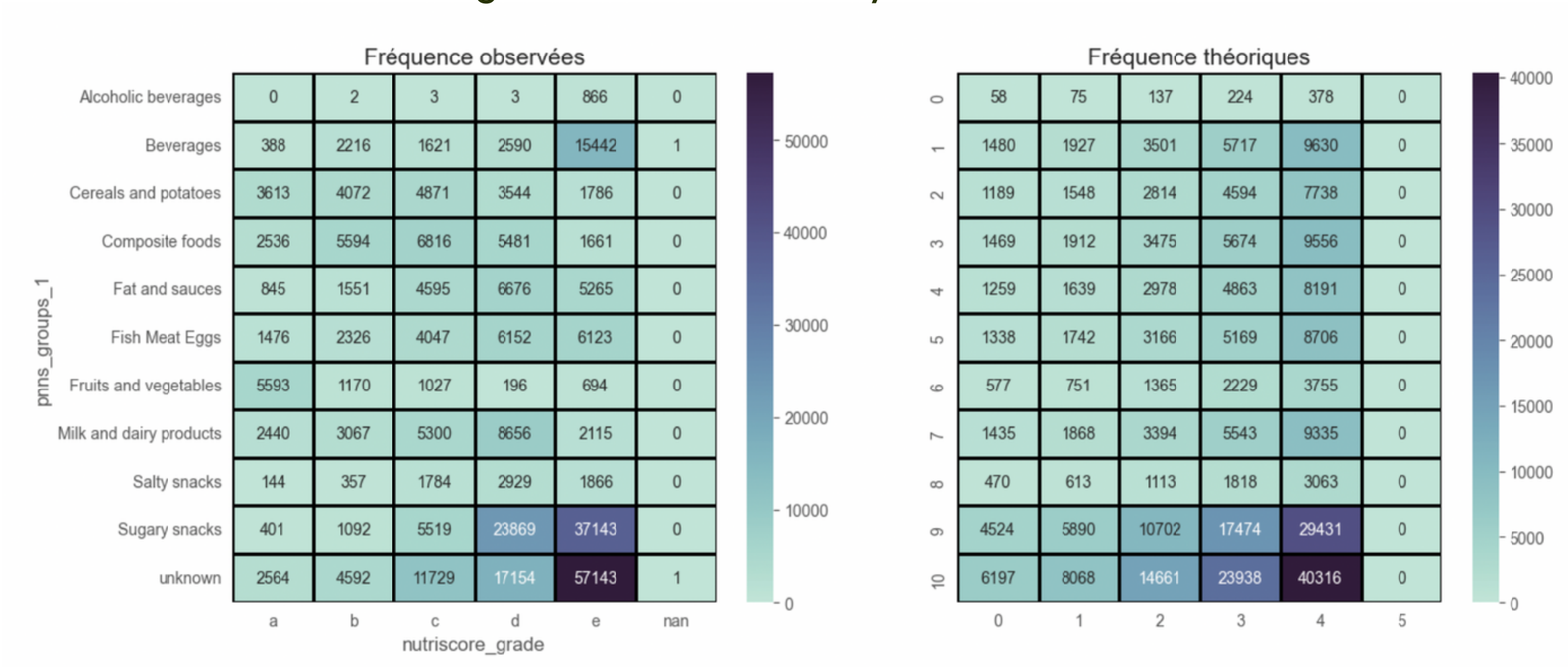
Statistical Tests on meaningfull features

1. Nutriscore_grade vs pnns_group_1
2. additives_n vs nutrients
3. Anova

NUTRISCORE_GRADE VS PNNS_GROUP_1

khi-2 test

This test aims at checking wether if there is any relation between the features below



2. Résultat du test du Qui-2

Coefficient du qui-2 : **133011.4694093311**

p-value calculée : **0.0**

H0 est rejetée : **pnns_groups_1** et **nutriscore_grade** sont corrélées significativement

Le coefficient de Cramer est de : **0.2974**

L'intensité du lien entre les variables est **moyen**

ADDITIVES_N VS NUTRISCORE_SCORE

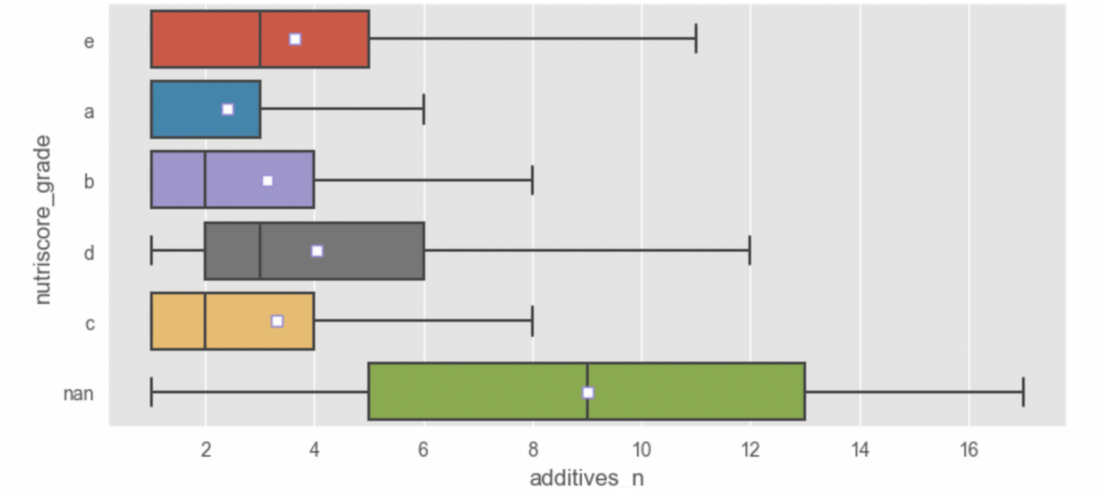
Spearman test

This test aims at checking whether the number of additives changes according to the others numerical components

```
==========================================================================================

        ##### 0. Hypothèse du test #####

    H0 : Les variables ('additives_n', 'nutriscore_score') sont indépendantes

    H1 : Les variables ('additives_n', 'nutriscore_score') sont corrélées

        ##### 1. Paramètre du test #####

    Variables aléatoires étudiées : ('additives_n', 'nutriscore_score')

    Indice de confiance : 0.05

    Taille de l'échantillon : 300707

         #### 2. Résultat du test ####

    coefficient de Spearman : 0.07046939314441944

    p-value associée au test de Spearman : 0.0

         #### 3. Conclusion du test ####


 --> ('additives_n', 'nutriscore_score') présentent significativement une négligeable (ou nulle) corrélation positiv
e.
==========================================================================================
```

ADDITIVES_N VS NUTRISCORE_GRADE

Anova test




```
Test ANOVA entre additives_n et nutriscore_grade
        Hypoyhèse nulle H0 :            Le nombre d'additifs n'affecte pas le nutriscore_grade (mêmes moyennes).
        Hypoyhèse alternative H1 :      Au moins un nutriscore_grade a une moyenne d'additives_n significativement d
ifférente.

Table ANOVA :
```

	sum_sq	df	F	PR(>F)
nutriscore_grade	1848.685716	4.0	49.799338	1.459960e-41
Residual	92760.337884	9995.0	NaN	NaN

```
p-value : 1.459960293710079e-41
L'hypothèse nulle est rejetée    ==> H1: Au moins un nutriscore_grade a une moyenne d'additives_n significativement d
ifférente.
```

USES CASES

HOW THE TOOL CAN BE USED

- Awareness system
- Recommandation system

Use case 1 : Awareness system

When a user scan a product, Aware Additiv' alerts whether if that product contains additives that dans are in 'dangereux' or 'peu recommandable' categories

.....

Use case 2 : Recommandation system

If a scanned product contains at least 1 additives from 'dangereux' or 'peu recommandable' categories, Aware Additiv' will recomand a similar healthier product

CONCLUSION

OpenFood est une BBD riche utilisée dans tous les pays du monde, ce qui est une force mais aussi une faiblesse à la fois. Le fait que celle-ci soit alimentée par l'utilisateur rend son exploration fastidieuse du fait de la non rigueur (coté users) et du manque de test au niveau de l'ingestion.

Du fait de sa richesse, il existe des dizaines voire des centaines de possibilité de projets en lien avec les produits de supermarché. Notre choix s'est porté dans la sensibilisation sur les additifs et la la prévention de certaines maladies qui pourraient s'avérer mortelles à termes.