Soutenance du Projet 3

Une application au service de la santé publique.

Parcours Data Scientist.

Présenté par Mr Dai TENSAOUT

Plan

Contexte

Idée d’application

État des lieux de la base de données

Filtrage et Nettoyage

Analyse Exploratoire (Exemples)

Exemple d’application

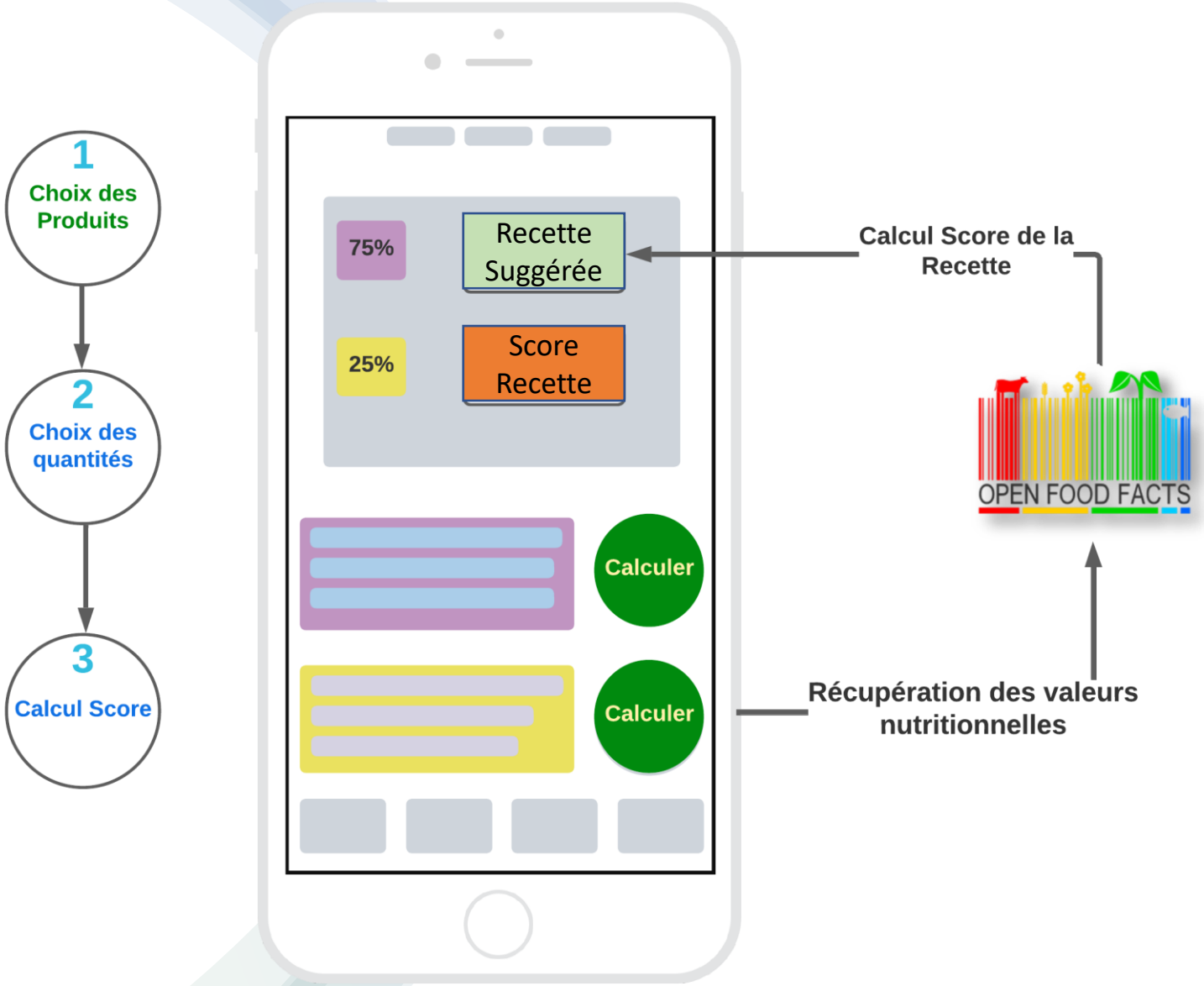
Contexte

Proposer une idée d’application en lien avec l’alimentation.

Jeu de données OPEN FOOD FACTS.

Faisabilité du projet et justifications.

Idée d’application



Etat des lieux de la base de données

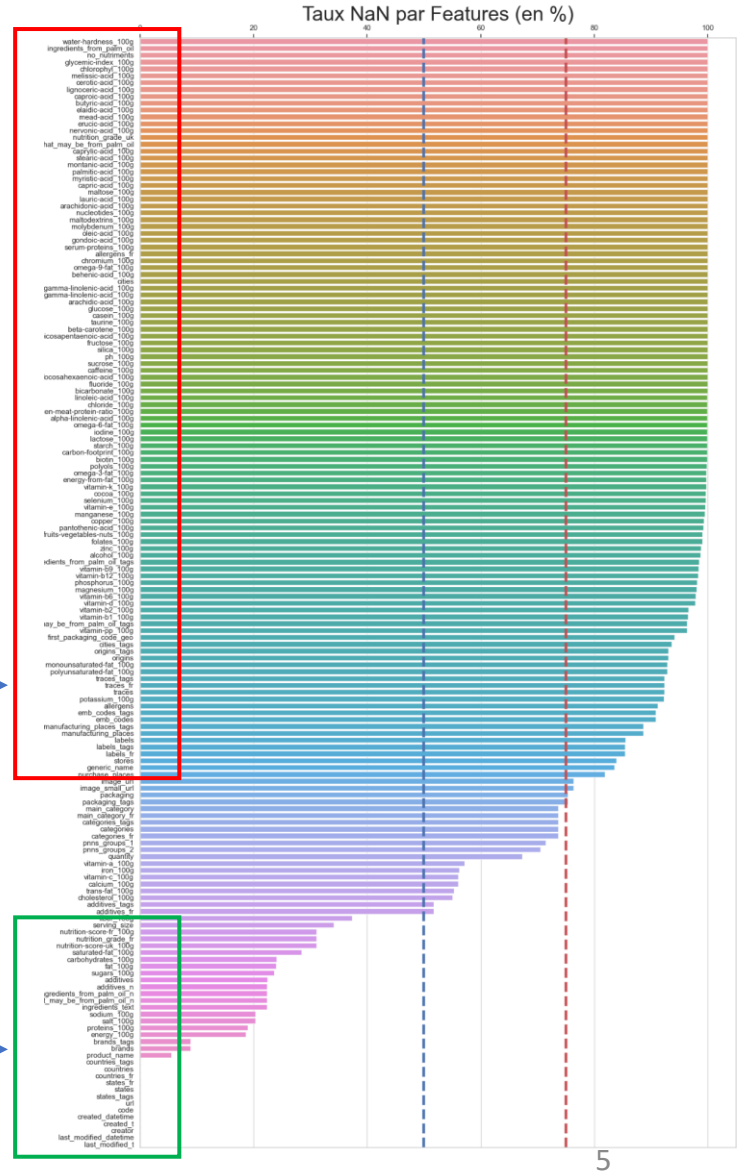
Taille:	320 722 Lignes 162 Colonnes
Taux de remplissage:	23,78 %
Doublon:	22
Colonnes vides	16

Champs

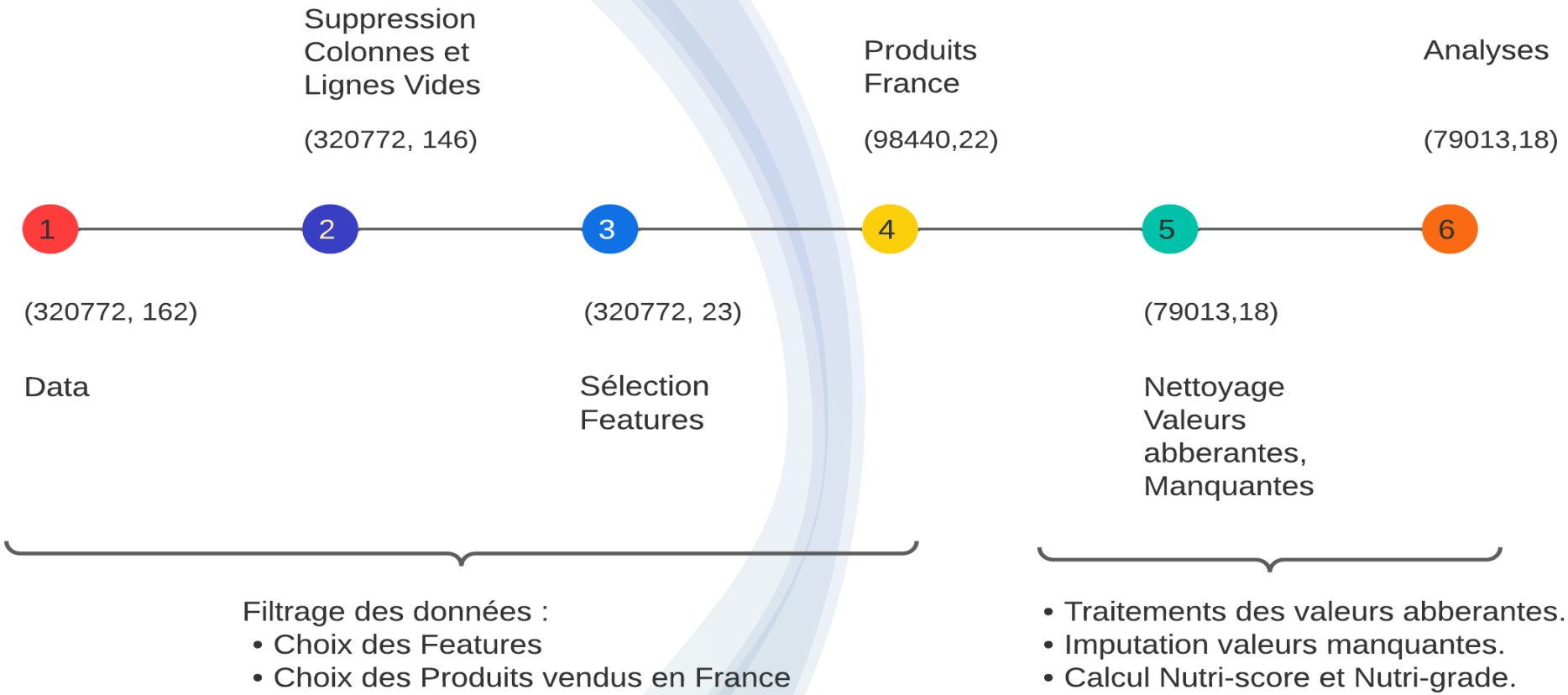
- Les informations générales sur la fiche du produit
- Un ensemble de tags
- Les ingrédients composant les produits
- Des informations nutritionnelles

Taux de NaN Sup à 70%

Taux de NaN Inf à 50%

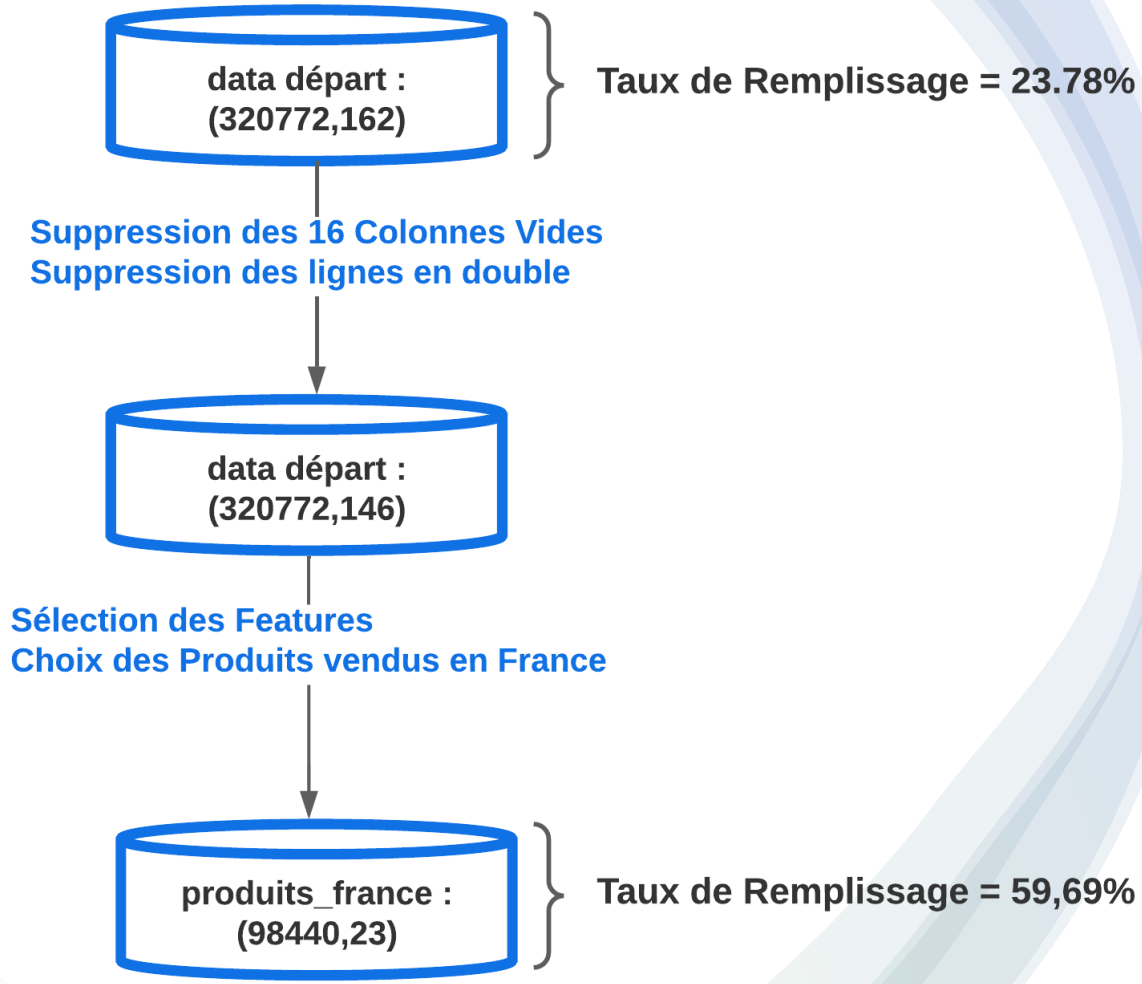


Filtrage et nettoyage



Filtrage et nettoyage

1 — 2 — 3 — 4 — 5 — 6



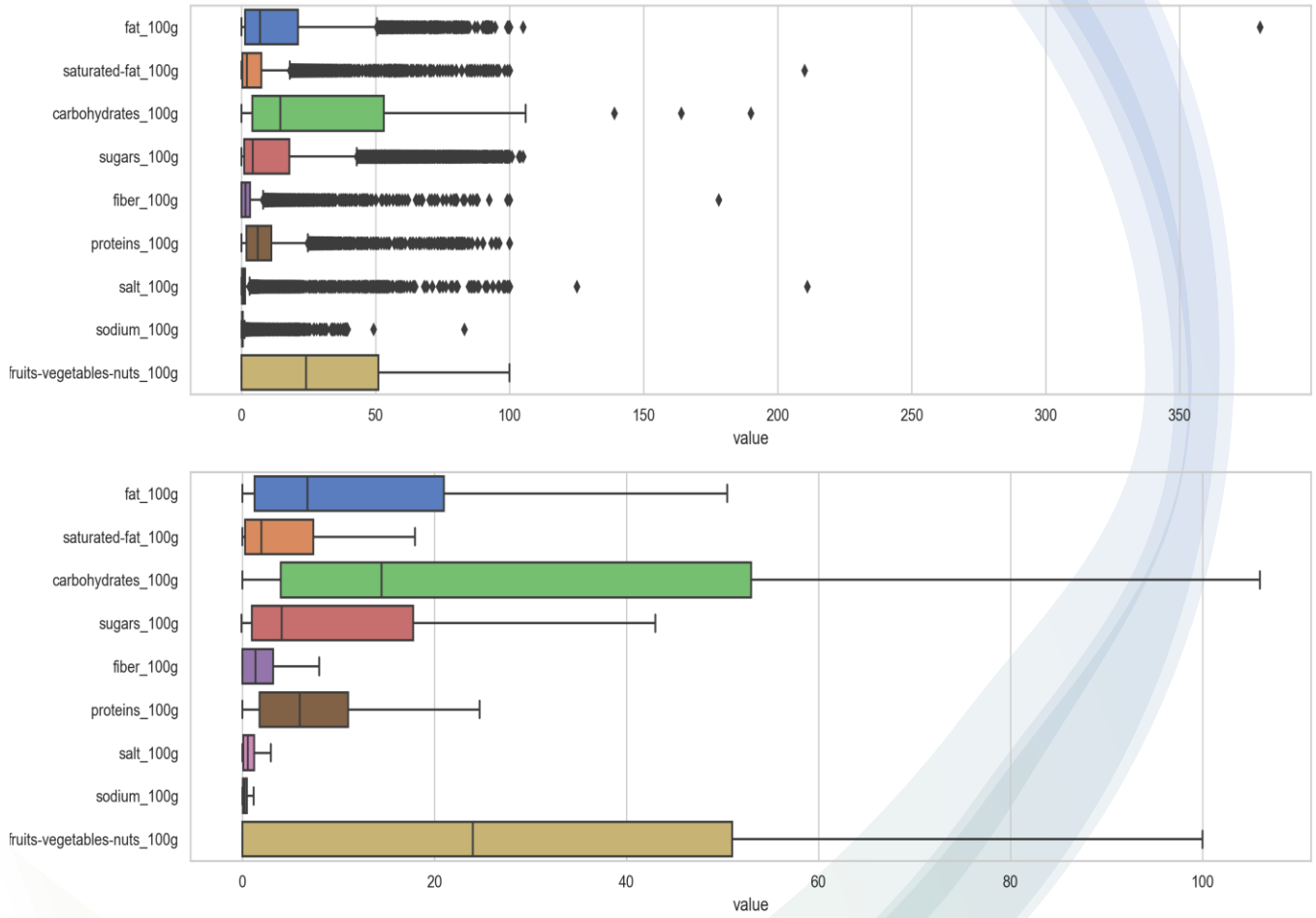

```
# Les taux de remplissage par colonnes :
produits_france.isna().mean().sort_values(ascending=False)
```

```
fruits-vegetables-nuts_100g    0.969819
additives_fr                   0.690614
fiber_100g                     0.535524
carbohydrates_100g             0.520408
fat_100g                       0.516030
additives_n                    0.456867
ingredients_from_palm_oil_n    0.456867
nutrition-score-fr_100g        0.376117
nutrition_grade_fr             0.376117
main_category_fr               0.370632
saturated-fat_100g             0.366365
sugars_100g                    0.364943
sodium_100g                    0.364374
salt_100g                      0.364344
proteins_100g                  0.346627
energy_100g                    0.343834
pnns_groups_1                  0.340908
pnns_groups_2                  0.318996
brands                         0.122044
product_name                   0.073070
countries_fr                   0.000000
code                           0.000000
dtype: float64
```

Filtrage et nettoyage

1 — 2 — 3 — 4 — 5 — 6

Distribution des valeurs nutritionnelles



Valeurs aberrantes

Visualisation des valeurs aberrantes graphiquement

Correction orthographes catégories pnns_groups 1 et 2

Suppression des valeurs aberrantes par différentes techniques :

- Remplacer les valeurs énergétiques >= 3800
- Les valeurs <0 et > 100 sont remplacer par des np.nan.
- Si sugars_100g = 100 alors carbohydrates_100g = 100.
- Si sugars_100g > carbohydrates_100g alors sugars_100g = **np.nan**

```
produits_france.loc[produits_france['proteins_100g']> 87.60, : ].shape
```

```
(13, 22)
```

Nous supprimerons directement ces produits.

Filtrage et nettoyage

1 — 2 — 3 — 4 — 5 — 6

Valeurs manquantes

Data_france_num : étant un data frame composé des colonnes **100_g**

```
missing_data_num_rows = check_missing_rows(data_france_num)
nombre_row_vides = missing_data_num_rows.loc[missing_data_num_rows['Percent'] == 100,:].shape[0]
print('Le Nombre de lignes vides est égal à :', nombre_row_vides, "Lignes")
```

```
Le Nombre de lignes vides est égal à : 19224 Lignes
```

```
#Taux de remplissage des features
produits_france.isna().mean().sort_values(ascending=False)*100
```

```
fruits-vegetables-nuts_100g    96.242390
additives_fr                   61.467100
fiber_100g                     42.250009
carbohydrates_100g             40.226292
fat_100g                       39.714984
pnns_groups_1                  33.703315
ingredients_from_palm_oil_n    32.357966
additives_n                    32.357966
pnns_groups_2                  30.973384
nutrition_grade_fr             22.293800
nutrition-score-fr_100g        22.293800
main_category_fr               21.830585
saturated-fat_100g             21.193981
sugars_100g                    21.033248
salt_100g                      20.829484
sodium_100g                    20.829484
proteins_100g                  18.622252
energy_100g                    18.388113
brands                          1.403567
product_name                    0.866946
code                            0.000000
Unnamed: 0                      0.000000
dtype: float64
```

Méthodes d'imputations

Variables qualitatives :
pnns_groups 1 & 2

Variables quantitatives :

- Méthode (*.fillna()*). *Médianes/pnns_2*
- *IterativeImputer.* *Features corrélés*
- Calcul : Energy_100g.
- Calcul Nutri Score.

```
# distribution des valeurs manquantes par colonne:
produits_france.isna().mean().sort_values(ascending=True)
```

```
code                           0.000000
sodium_100g                    0.000000
salt_100g                      0.000000
proteins_100g                  0.000000
fiber_100g                     0.000000
sugars_100g                    0.000000
carbohydrates_100g             0.000000
saturated-fat_100g             0.000000
fat_100g                       0.000000
energy_100g                    0.000000
pnns_groups_2                  0.000000
pnns_groups_1                  0.000000
ingredients_from_palm_oil_n    0.000000
additives_n                    0.000000
brands                         0.000000
product_name                   0.000000
fruits-vegetables-nuts_100g    0.000000
main_category_fr               0.218306
nutrition_grade_fr             0.222938
nutrition-score-fr_100g        0.222938
dtype: float64
```

Filtrage et nettoyage

1 — 2 — 3 — 4 — 5 — 6

Calcul Nutri Score

Mode de calcul[1] :

1. Attribution des points pour chaque composante.
2. Calcul des deux composantes P (+) et N (-).
3. Calcul du Nutri score final.
4. Cas Particulier : Boissons et Fromages,

```python
# Fonction pour calculer le nutriscore : (Entrée df) ----> (Sortie df)
def calcul_nutriscore(df):
    # Points Energie :
    conditions = [(df['energy_100g'] <= 335) ,(df['energy_100g'] > 335)&(df['energy_100g'] <= 670),
                  (df['energy_100g'] > 670)&(df['energy_100g'] <= 1005),(df['energy_100g'] > 1005)&(df['energy_100g'] <= 1340),
                  (df['energy_100g'] > 1340)&(df['energy_100g'] <=1675),(df['energy_100g'] > 1675)&(df['energy_100g'] <=2010),
                  (df['energy_100g'] > 2010)&(df['energy_100g'] <= 2345),(df['energy_100g'] > 2345)&(df['energy_100g'] <=2680),
                  (df['energy_100g'] > 2680)&(df['energy_100g'] <=3015),(df['energy_100g'] > 3015)&(df['energy_100g'] <=3350),
                  (df['energy_100g'] > 3350)]
    valeurs = [0,1,2,3,4,5,6,7,8,9,10]
    df['Point_Energie'] = np.select(conditions, valeurs)

    # Points Energie Boissons :
    conditions_b =[
        (df['pnns_groups_1'] == 'Beverages') & (df['energy_100g'] <=0),
        (df['pnns_groups_1'] == 'Beverages') & (df['energy_100g']  > 0) & (df['energy_100g'] <=30),
```

```python
# Fonction cas Particulier :

def cas_particulier_nutriscore(df):
    # Cas Particulier
    # 1- Boissons Fruité  :
    mask_nutriscore_boissons = (df['pnns_groups_1'] == 'Beverages') & ((df['N'] < 11) & (df['point_fruit'] == 10))
    df.loc[mask_nutriscore_boissons, 'nutri_score_calcule']   = df.loc[mask_nutriscore_boissons,'N'] - df.loc[mask_nutrisc

    # 2- Boissons
    mask_nutriscore_boissons = (df['pnns_groups_1'] == 'Beverages') & (df['N'] >= 11) & (df['point_fruit'] < 10)
       #mask_fruit_fibre_boissons = (df['point_fibre'] < 10) & (df['point_fruit'] < 10)
       #masque_boissons = mask_nutriscore_boissons &  mask_fruit_fibre_boissons

    # Recalcul pour les boissons :
    df.loc[mask_nutriscore_boissons,'nutri_score_calcule' ] = df.loc[mask_nutriscore_boissons,'N'] - (df.loc[mask_nutriscc

    #Cas Particuliers :

    #1 - Si : Fromages
```

1 Source : Règlement d'usage Nutri-score Fichier PDF à télécharger

N = *P_Densité.Énergétique + P_Graisse.Sat + P_Sucres + P_Sodium*

P = *P_Fruit.Légumes + P_Fibres + P_Protéines*

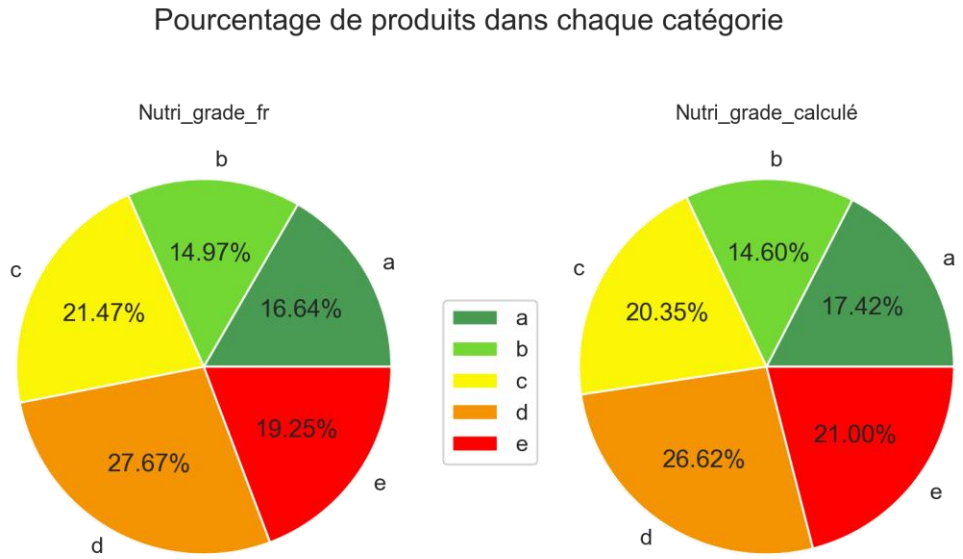
Filtrage et nettoyage

1 — 2 — 3 — 4 — 5 — 6

Calcul Nutri Score et imputation *NaN*

Attribution des *Nutri grade* :

1. Attribution des *Nutri grade* (*aliments solides + boissons*).
2. Imputations des *NaN par les valeurs calculées: Nutri score & Nutri grade.*
3. Sauvegarde pour analyse exploratoire.



	nutrition-score-fr_100g	nutri_score_calcule
1	22.0	22
9	14.0	14
10	14.0	14
11	13.0	13
12	15.0	15
...	...	...

Analyse exploratoire

1 2 3 4 5 6

Analyse Exploratoire

Statistiques descriptives et exploration :

- Affichage data set
- Types et info()
- Value_counts()

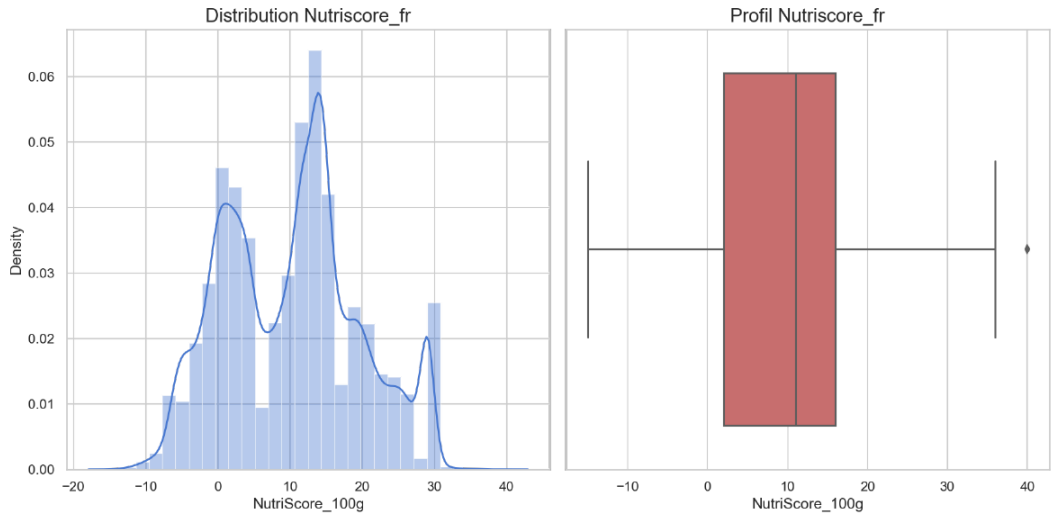
Analyses Uni variées	
Variables catégorielles	
Variables numériques	

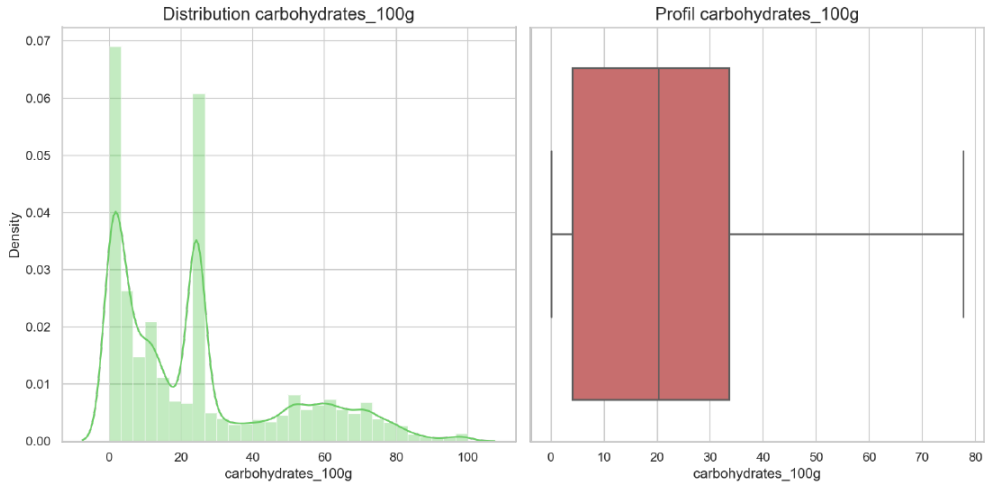
Analyses Bi variées	
Numériques Vs Numériques	**Corrélations**
Numériques Vs Catégories	
Catégories Vs Catégories	**Table de contingence**

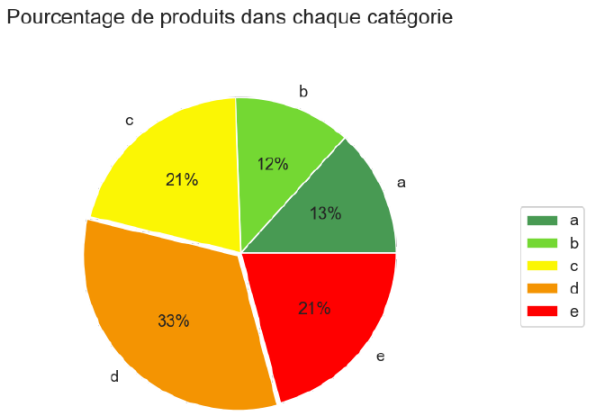
Analyses Multi variées	
ACP : réduction de dimension	
ANOVA : Analyse de la varriance	

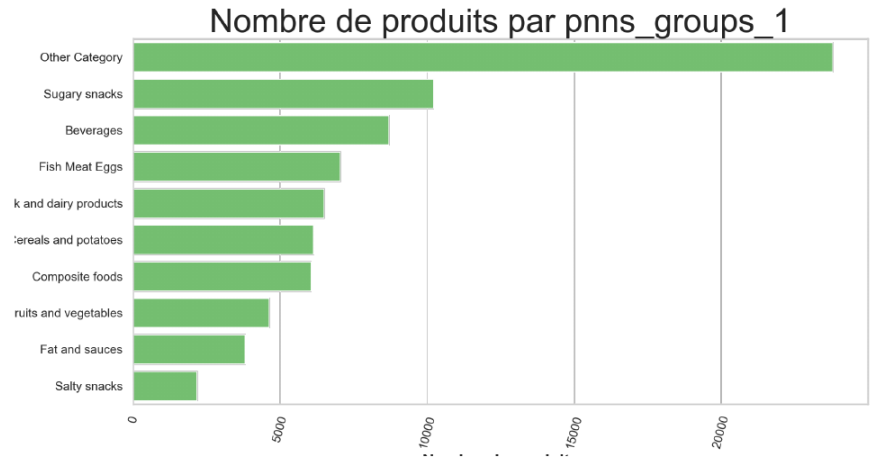
Exemples d’analyses

Analyses Uni variées



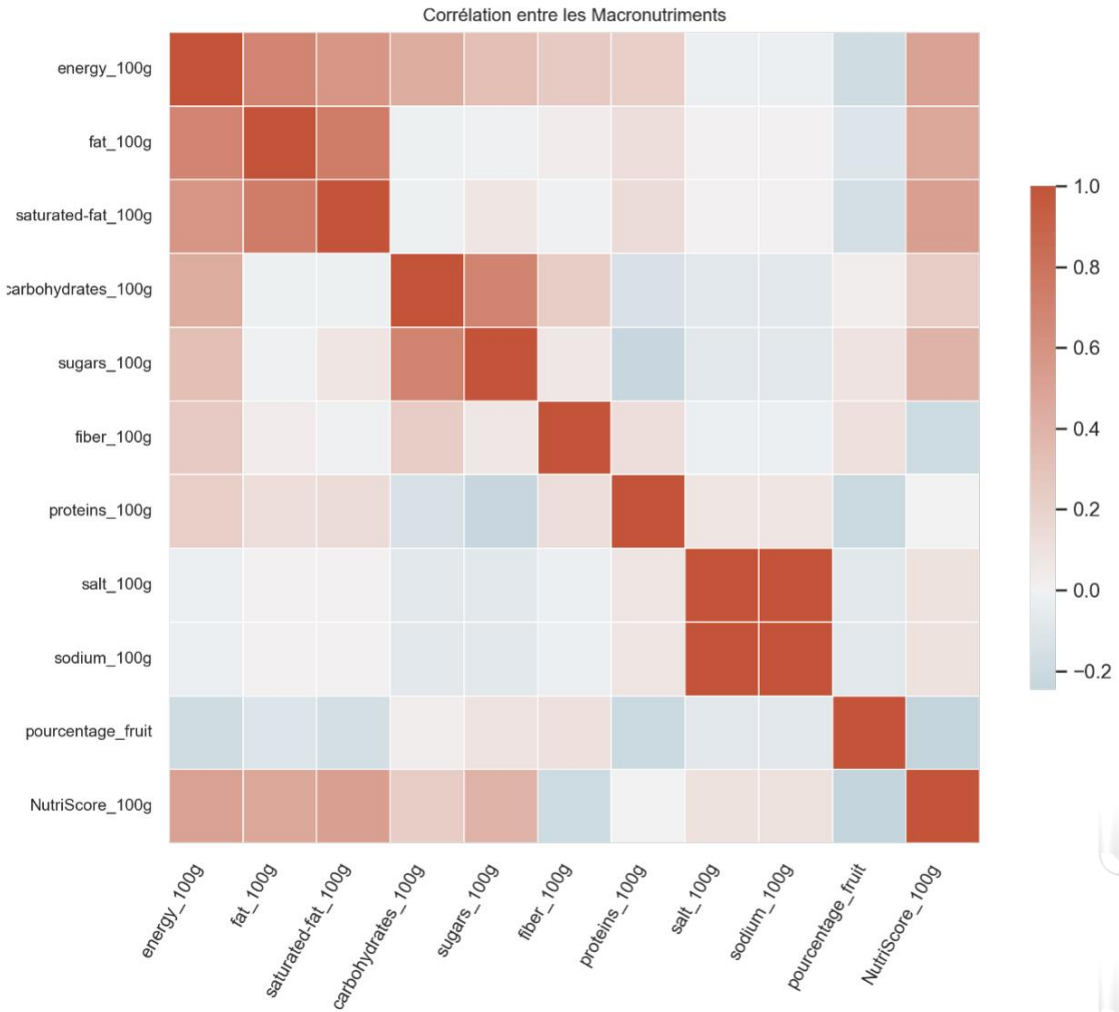


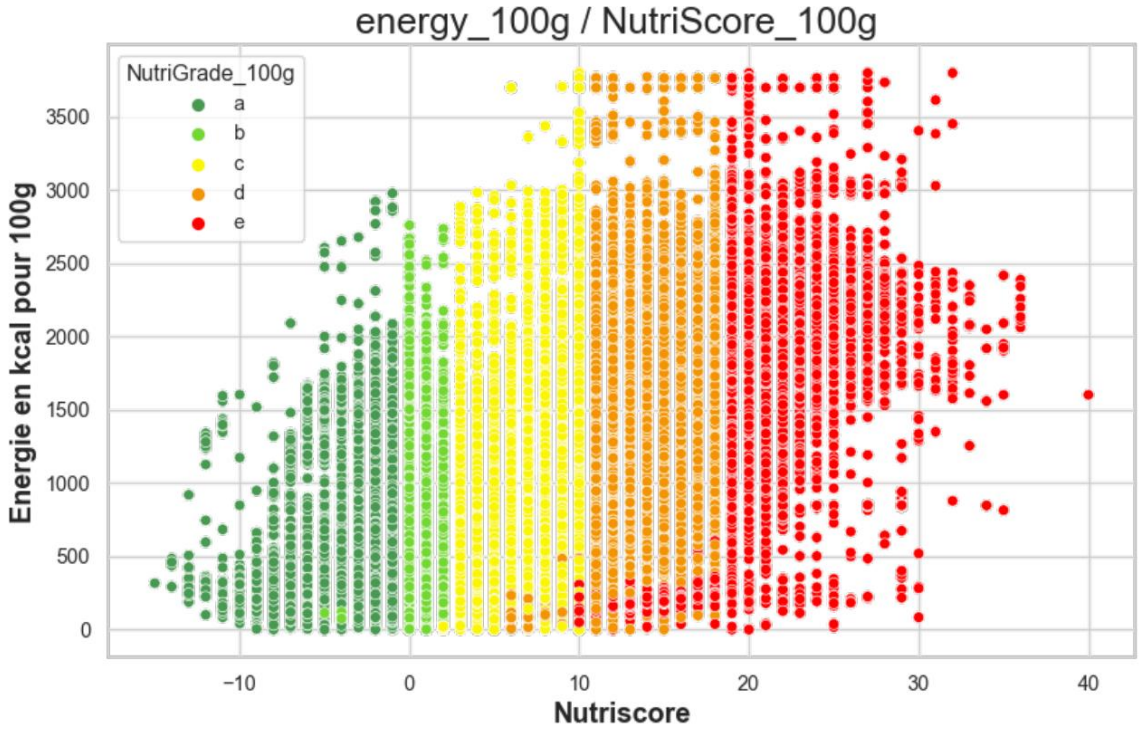




Exemples d'analyses

Analyses Bi variées



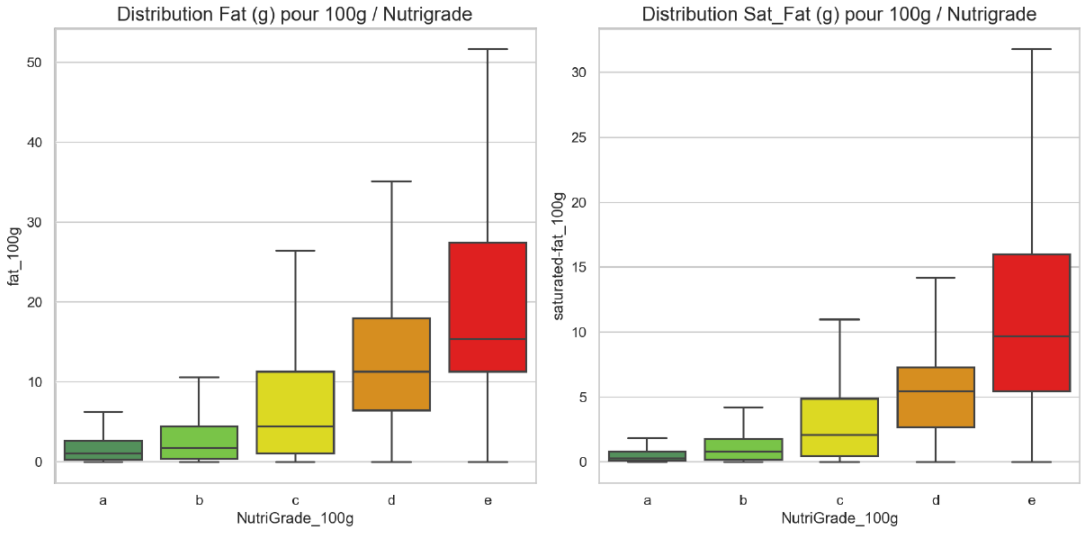


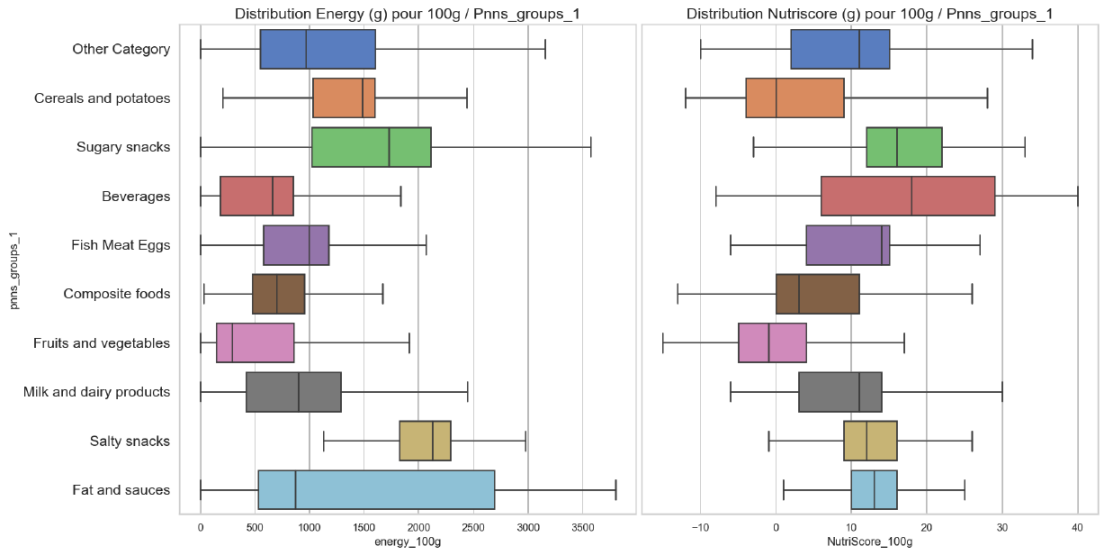
Energy vs (fat & satu_fat & carbohydrates & Nutriscore) : corrélation positive et significative.

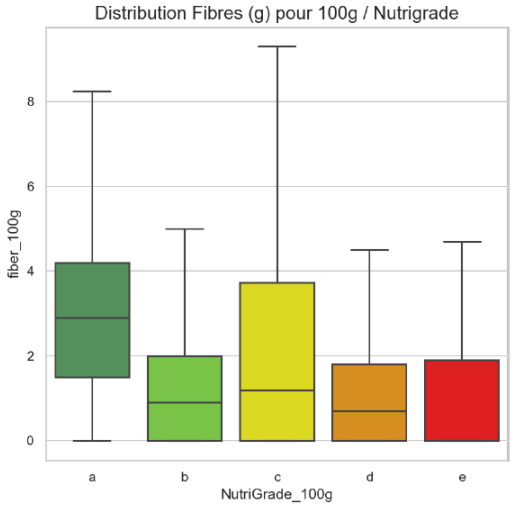
Une corrélation positive et forte entre (Fat & satu_fat) et (carbohydrates & sugars) et (Sodium et Sel)

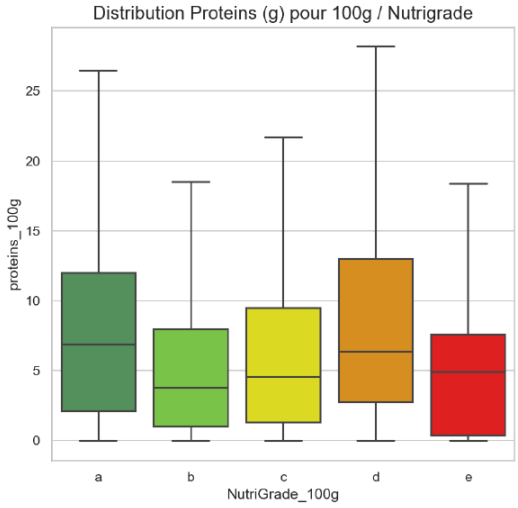
Exemples d'analyses

Analyses Bi variées









Nous remarquons que la distribution du Nutri-grade selon les macronutriments suit bien la logique de calcul de celui-ci.

Les produits riches en Fat et Saturated_Fat sont plutôt classés C, D et E.

Contrairement au produits riches en Fibres et Protéines, souvent Classés A et B

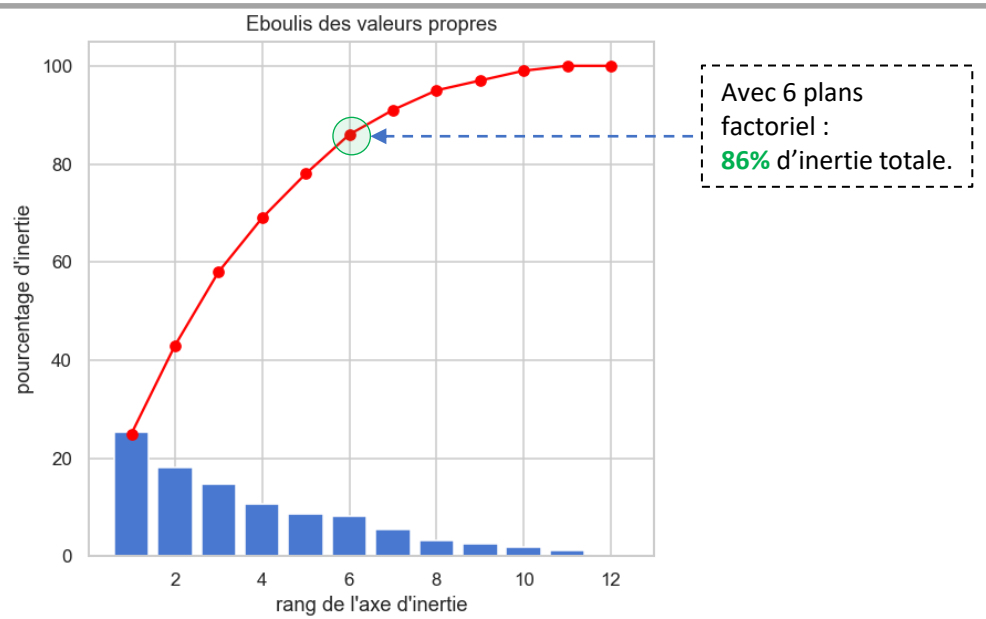
Exemples d'analyses

ACP & ANOVA

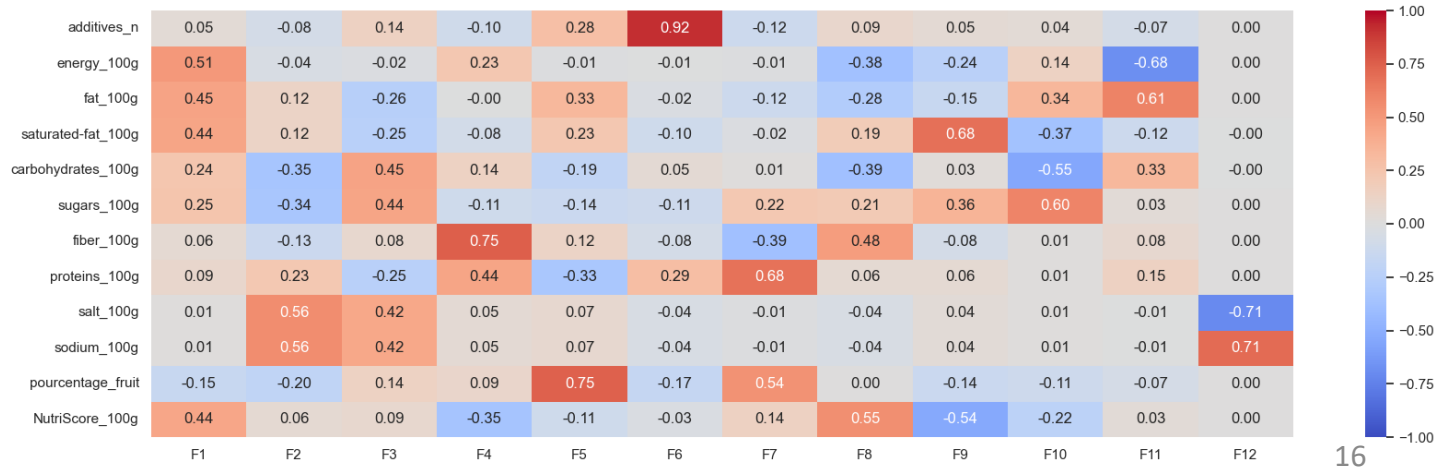
1. *Scaler* les données.
2. Matrice Covariance.
3. Eblouis des valeurs propres.
4. Contributions des variables.
5. Graphiques des corrélations.
6. Projection des individus.

```
scaler = StandardScaler()

# On transforme :
scaled_X = scaler.fit_transform(data_acp)
scaled_X
```

```
array([[-5.51190536e-01, -1.99088114e-01, -2.51276635e-16, ...,
        -3.73519100e-17, -5.91460348e-01,  4.08141955e-01],
       [-1.12396176e-01,  1.13112664e+00,  6.15515432e-01, ...,
        -2.84349131e-01, -5.91460348e-01,  1.26546765e+00],
       [-5.51190536e-01,  5.62403024e-01, -5.04110154e-01, ...,
        -3.73519100e-17, -5.91460348e-01,  4.08141955e-01],
       ...,
       [-5.51190536e-01, -3.31365871e-01, -2.51276635e-16, ...,
        -3.73519100e-17, -5.91460348e-01,  2.01562763e+00],
       [-5.51190536e-01, -1.50311062e+00, -7.84900652e-01, ...,
        -3.03490999e-01, -1.67620937e-01, -8.77846585e-01],
       [-5.51190536e-01,  1.40023955e-01, -2.51276635e-16, ...,
        -3.73519100e-17, -5.91460348e-01,  5.15307666e-01]])
```



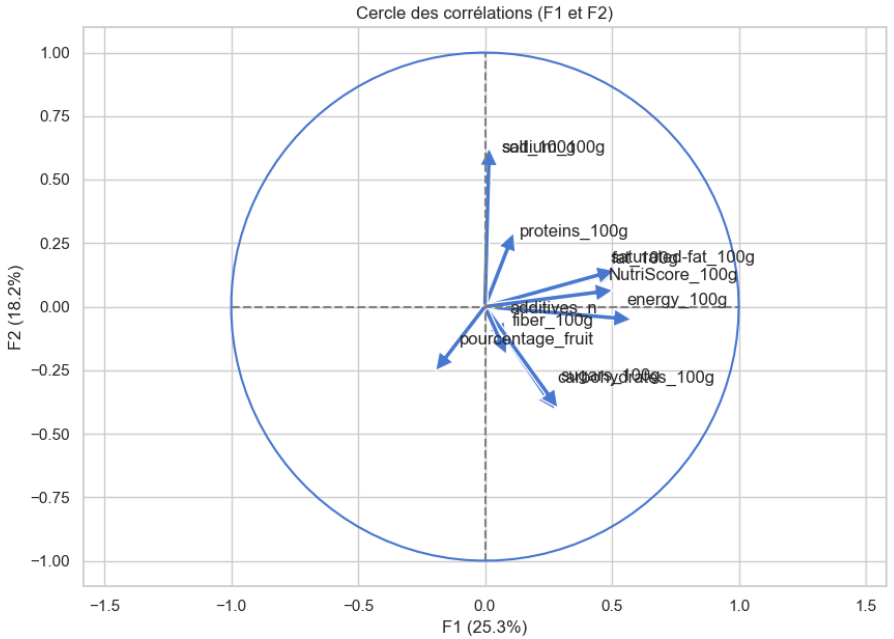
Avec 6 plans factoriel : **86%** d'inertie totale.



Exemples d'analyses

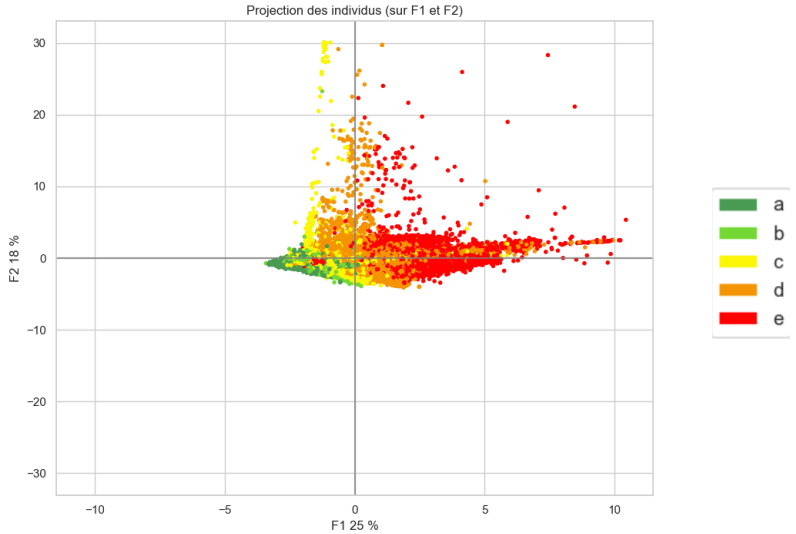
ACP & ANOVA

1. Scaler les données.
2. Matrice Covariance.
3. Eblouis des valeurs propres.
4. Contributions des variables.
5. Graphiques des corrélations.
6. Projection des individus.




```
# F1 et F2
x_y = (0,1)
correlation_graph(pca, x_y, fetures_acp)
```

```
x_y = [0,1]
labels = Nutrigrade_individus
Nutrigrade_color_dict = {'a':"#499A53", 'b':"#74D834", 'c':"#FBF605",'d': "#F49402",'e': "#FF0000"}
clusters = [Nutrigrade_color_dict[label] for label in labels]
display_factorial_planes(X_proj, x_y, clusters= clusters, pca=pca)
```

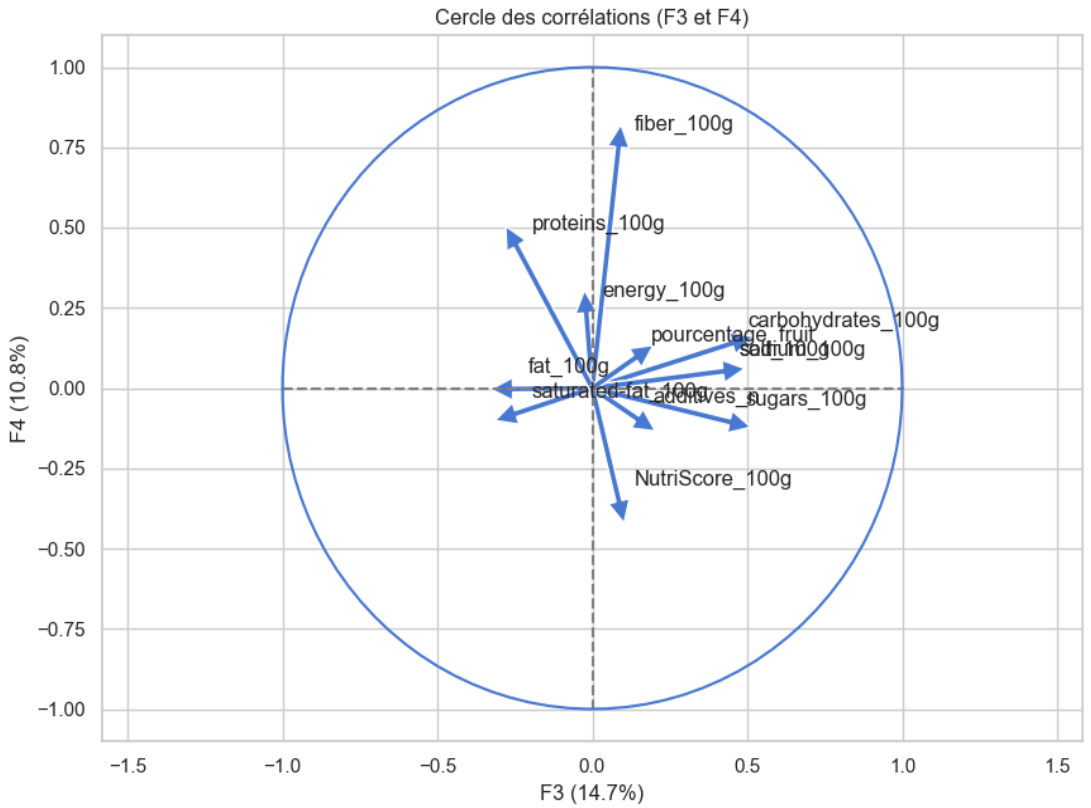


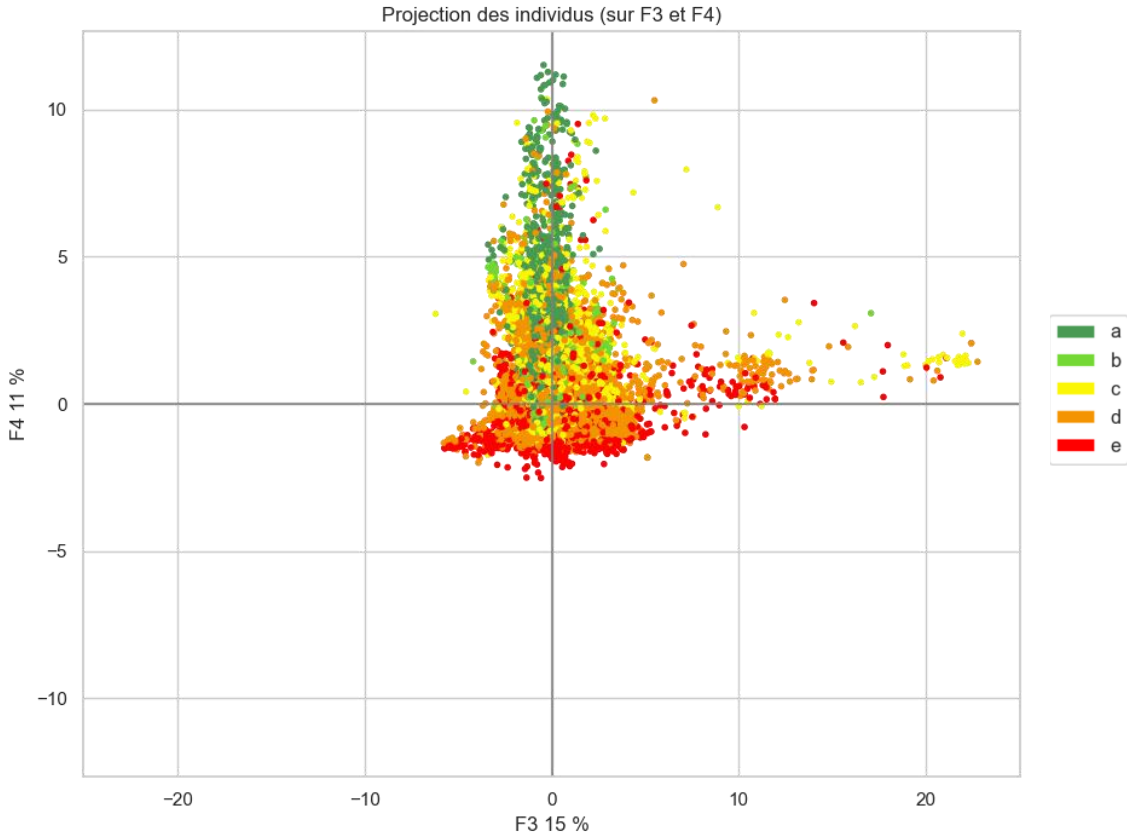
On peut voir que la première composante **F1** est corrélée avec les variables **energy_100g** **fat_100g** et saturated-fat_100g, cette composante est donc liée aux matières grasses.

La composante **F2** est donc liée aux prdouits salés, car corrélée avec **salt_100g** et **sodium_100g** , nous remarquons ici que ces variables apporte la même information.

Exemples d'analyses

ACP & ANOVA





La composante **`F3`** est corrélée avec **`carbohydrates_100g`** et **`sugars_100g`**, elle est donc liée au produits sucrés.

La composante **`F4`** quand à elle, est liée au produits riches en **`fibres et en proteines`**.

Exemples d'analyses

ACP & ANOVA

9.2 ANOVA Pnns_groups_1 et Energie

Ronald Fisher : L'ANOVA (analyse de la variance) est une méthode statistique pour analyser la relation entre plus de deux groupes indépendants d'une variable (en comparant leurs moyennes) et son effet sur la variable dépendante numérique.

Le but principale de l'ANOVA ici, et d'étudier la relation entre une variable qualitative `pnns_groups_1` avec une variable quantitative `energy_100g` , nous poserons les hypothèses suivantes:

- H0 : La distribution des données est similaire dans les différents groupe pnns_groups_1.
- H1 : Une ou plusieurs distribution sont différentes.

Avec un risque d'erreur `alpha = 0.05` **Si** `P_Value` **inf ou égale à alpha** : Les différences entre certaines des moyennes sont statistiquement significatives, on `rejette alors l'hypothèse nulle` .

Si `P_value` est **sup à alpha** Les différences entre les moyennes ne sont pas statistiquement significatives, dans ce cas on `accepte l'hypothèse nulle` .

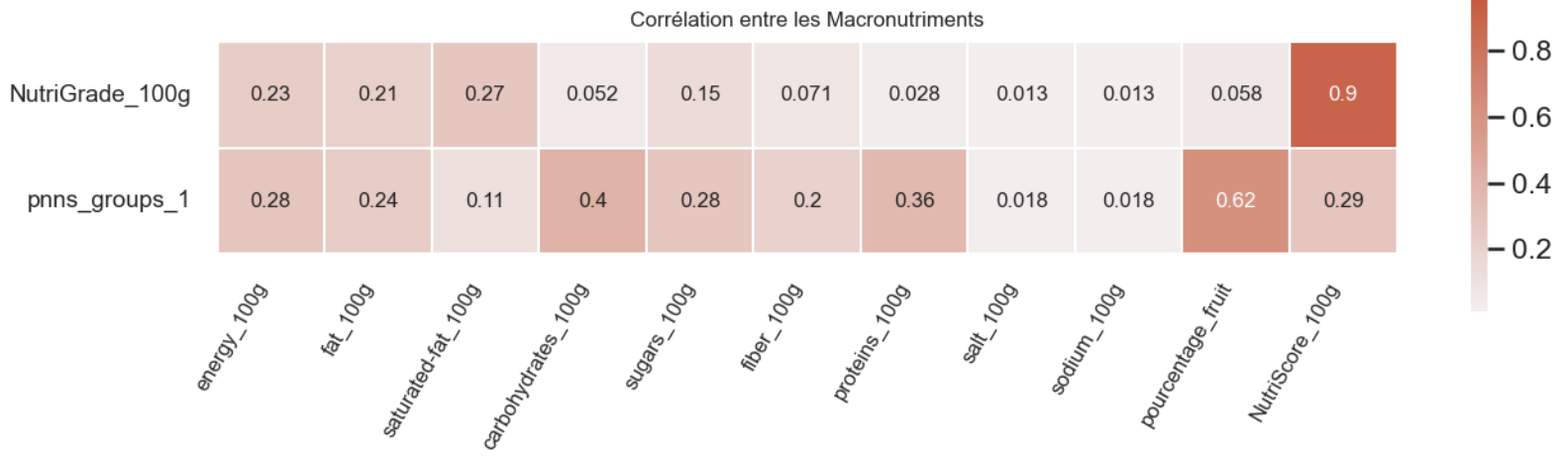
```
from scipy import stats
def anova(var_quali, var_quanti, data=df, affiche=None):
```

```
# Variable Quanti Vs Quali
anova(var_quali='pnns_groups_1', var_quanti='carbohydrates_100g', data=df, affiche=True)
```

```
La statistique de Fisher =  5926.892853841597
La P-Value =  0.0
eta_squared =  0.40305278855874543
```

Exemples d'analyses

ACP & ANOVA




```
round(p_value_matrix(df_anova),3)
```

	NutriGrade_100g	pnns_groups_1
energy_100g	0.0	0.0
fat_100g	0.0	0.0
saturated-fat_100g	0.0	0.0
carbohydrates_100g	0.0	0.0
sugars_100g	0.0	0.0
fiber_100g	0.0	0.0
proteins_100g	0.0	0.0
salt_100g	0.0	0.0
sodium_100g	0.0	0.0
pourcentage_fruit	0.0	0.0
NutriScore_100g	0.0	0.0

Une forte relation entre le `Nutri grade` et le `Nutri score`, ce qui est logique.

Une forte relation aussi entre le (`pnns_groups_1` & `Nutri grade`) et `pourcentage_fruit` .

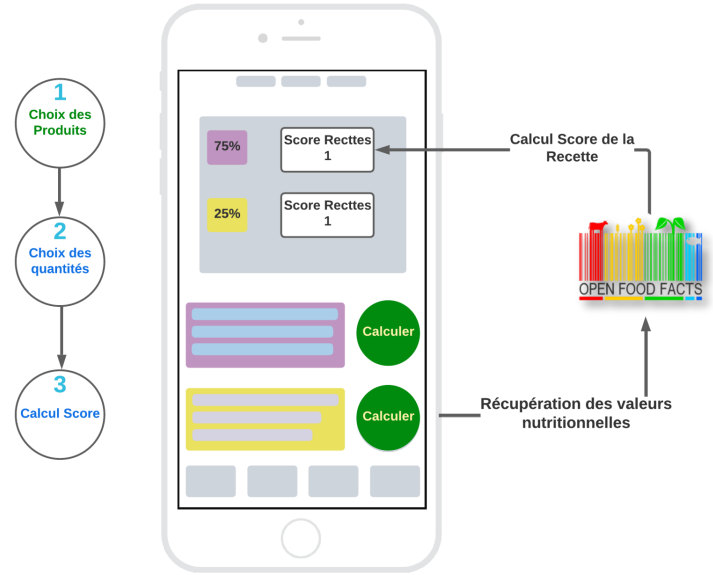
Toutes les p_values sont inférieures à 0.05 ce qui laisse penser qu'ils exsistent des différence significatives entre les groupes `pnns_groups_1` est les variables dépendantes dans notre cas les valeurs nutritionnelles des produits et le Nutriscore.

Exemple de notre idée d'application

```python
#Liste d'ingrédients :
liste_ingredient = ['Filet de bœuf','lentilles vertes ','Barbecue Sauce','Sauce aux piments','Boisson au soja',
                    'Beurre gastronomique doux','Riz']

#Composition de la rectte :
recette = df_recette.loc[(df_recette['product_name'] == 'Tortillas') ,:].iloc[:1]

for ingredient in liste_ingredient:
        p = df_recette.loc[(df_recette['product_name'] == ingredient) ,:].iloc[:1]
        recette = pd.concat([recette, p])

#Quantité de chauqe ingrédient:
Quantites_produits =  [250, 550,400,100,60,20,350,120]
recette['Quantite_produit'] = Quantites_produits

recette
```

```python
# Exemple Avec une rectte de 4 Ingrédients
score_recette(recette)
```

```
Votre rectte a un score de  28.000000000000004 %, Votre recette est  moyennement diététique
```



Remarques :

- Le Score prend en compte que les composantes (Sucres, Fat et Energie), une amélioration serait d'ajouter D'autres critères pour catégoriser les recettes.

- Après avoir calculer le score, nous pourrons imaginer une autre fonction qui suggère une autre recette avec des Produits similaires pour obtenir un bon score diététique.

Une application au service de la santé publique.

Parcours Data Scientist.

Merci de votre attention

Présenté par Mr Dai TENSAOUT