Projet 2:Analyse des données nutritionnelles

Réalisé par: Taher Haggui

Problématique :

Aider le site La marmite à construire un générateur de recettes saines.

Comment ?

1.Recenser une base de données de milliers de produit.

2.Préparer les données .

3.Une analyse exploratoire des données.

Résultats attendues ?

1. L’analyse des différentes variables importantes.

2. L’analyse et l’étude de corrélation inter variables.

3. La recherche des modèles régissant la relation entre les variables.

4. Définition des features engineering.

Nettoyage des données :

1ére commande: état initial de la base de données

In [4]:
```
# Statics of data
data.describe()
```

Out[4]:

	no_nutriments	additives_n	ingredients_from_palm_oil_n	ingredients_from_palm_oil	ingredients_that_may_be_from_palm_oil_n	ingredients_that_may_
count	0.0	248939.000000	248939.000000	0.0	248939.000000	
mean	NaN	1.936024	0.019659	NaN	0.055246	
std	NaN	2.502019	0.140524	NaN	0.269207	
min	NaN	0.000000	0.000000	NaN	0.000000	
25%	NaN	0.000000	0.000000	NaN	0.000000	
50%	NaN	1.000000	0.000000	NaN	0.000000	
75%	NaN	3.000000	0.000000	NaN	0.000000	
max	NaN	31.000000	2.000000	NaN	6.000000	

8 rows × 106 columns

Nettoyage de données : Colonnes inutiles

We will drop each columns have more than 50% missing values.

In [6]:
```
def del_useless_columns(df,level):
    """ This function allow to delete columns, which have more than the accepted level of missing
        values.The accepted level is defined at first and introduced as argument.

         Args:

         df(DataFrame): the data,which the function will examinate.
         level(int): the level  of missing values that the function accept to save column.

         Returns:

         dt(DataFrame): DataFrame, which contain only columns that have less than 50% of missing
         values.
         """

    l=len(df)
    for c in df.columns:
        if c !="product_name":
            test=((df[c].count())/l)*100
            if test < level:
                del (df[c])
    return df
```

Nettoyage de données : Valeurs manquantes

Choix1 : Suppression des lignes dont le product_name est vide.

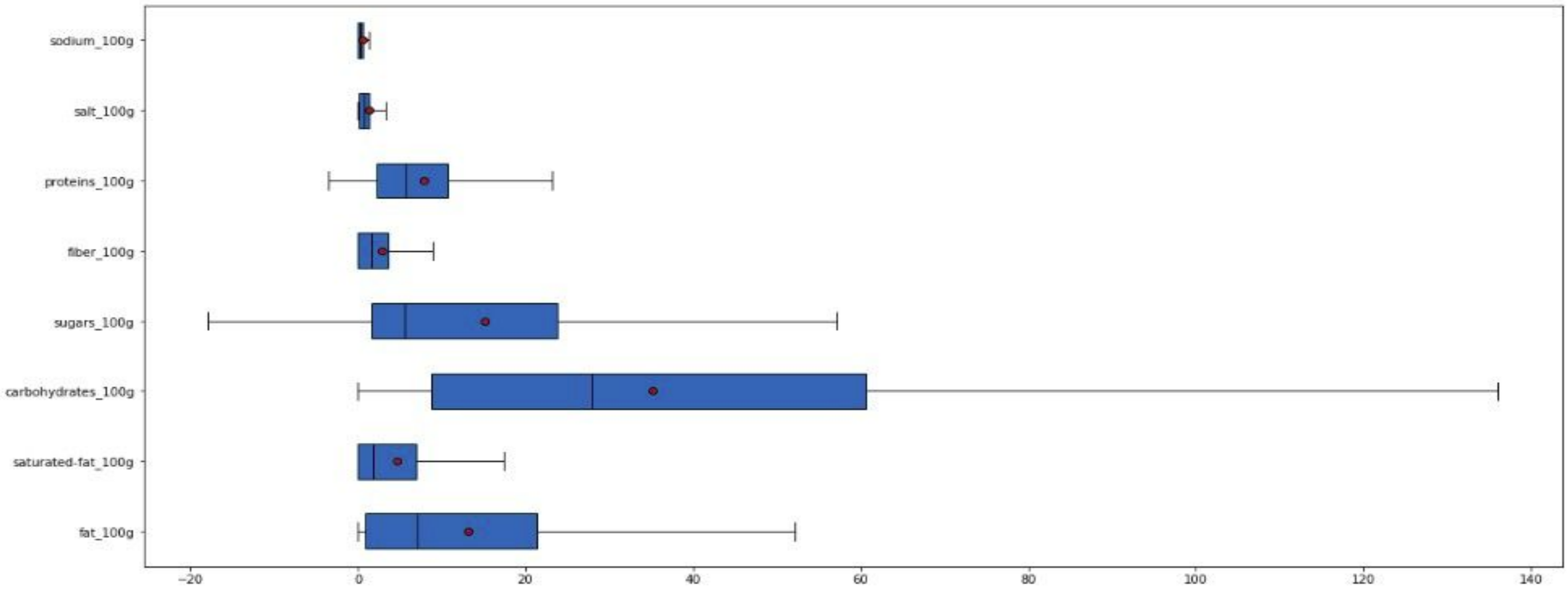
```
In [10]: #We will drop all rows which have missing product_name'value.

         data=data[data.product_name.notnull()]
```

Choix2 : Suppression des lignes contenant des valeurs manquantes.

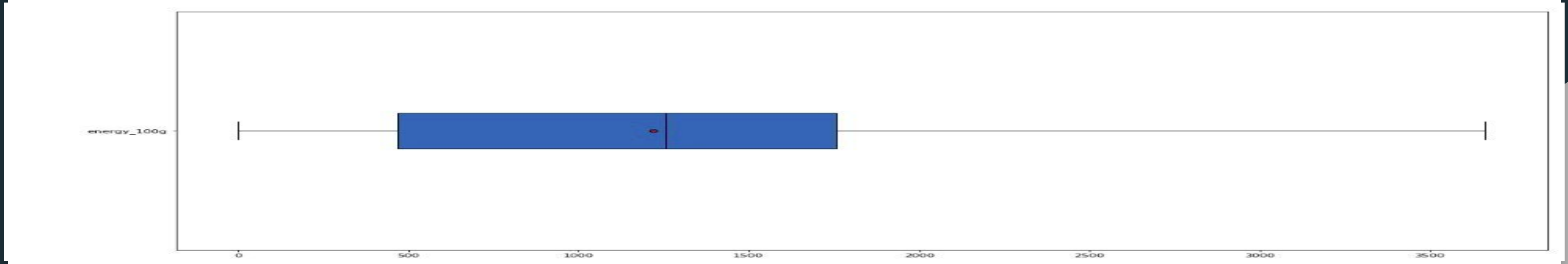
```
In [12]: #The data is large enough .So eliminate all rows, which contains missing values is more better
         #than replace them by estimated value.
         data=data.dropna(axis=0,how='any')
```

Valeurs aberrantes



Dispersion de la quantité de chaque nutriment dans 100grs de chaque produit

Valeurs aberrantes



Dispersion de la quantité d’énergie dans 100grs de chaque produit

Q1: is the middle value beteween the smallest number and the median of the dataset.

Q2: is the median of the dataset.

Q3: is the middle value between the median and the highest value.

IQR=Q3-Q1

Choix : garder que les valeurs comprises entre Q1-1.5IQR et Q3+1.5IQR.

Préparation des données

Action1: Organiser les colonnes par type.Mettre les types objects au début, puis les colonnes de type numérique.

In [4]:
```
#We will arrange the data by type, we will make object columns at first , then we make the
#numeric columns. This organisation can ease us the exploration of features .
list_obj=[]
list_num=[]
for c in data.columns:
    if data[c].dtypes==object:
        list_obj.append(c)
    else:
        list_num.append(c)
lis=list_obj+list_num
data=data[lis]
```

Action2: Suppression des colonnes dupliquées et non utiles.

After deleting of duplicate columns , we should mention that many columns are useless for our analyse. For example:
-We can mention that column of code and product_name can give the same information.So we will keep only product_name.
-The column namely url is not useful for our analyse. So we can get red of it.
-The columns namely created_datetime, last_modified_datetime are not useful for our odjectif.So we will get red of it.
-The column of creator of product , can not be interesting four our analyse. So we will get red of it.

In [9]:
```
del(data["code"])
del(data["url"])
del(data["created_datetime"])
del(data["last_modified_datetime"])
del(data["creator"])
list_obj.remove("code")
list_obj.remove("url")
list_obj.remove("created_datetime")
list_obj.remove("last_modified_datetime")
list_obj.remove("creator")
```

Exploration des données:Analyse univariée

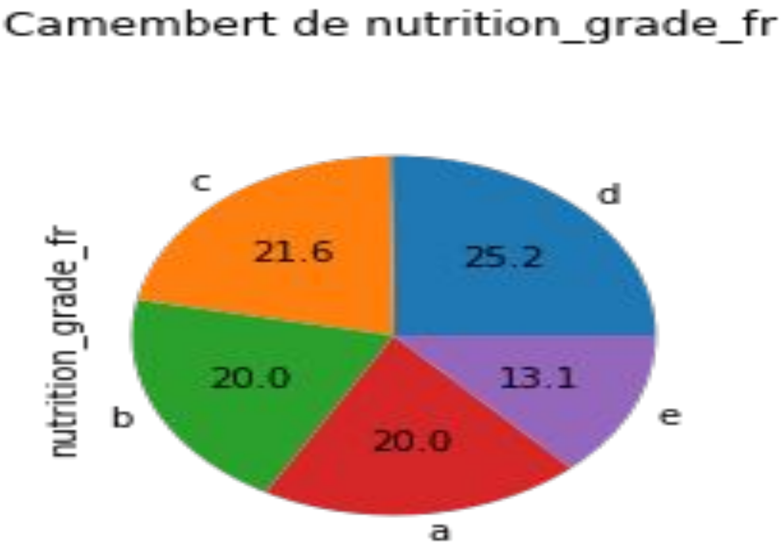
In [27]:
```
table={}
for c in list_num:
    table[c]=[data[c].mean(),data[c].var(ddof=1),data[c].std(ddof=1),np.median(data[c]),data[c].skew(),data[c].kurtosis()]
pd.DataFrame(table,index=["Mean","Variance","Standard deviation","Median","Symetrie measurement"," Flattening measurement"]
```

Out[27]:

	Unnamed: 0	additives_n	ingredients_from_palm_oil_n	ingredients_that_may_be_from_palm_oil_n	energy_100g	fat_100g	saturated-fat_100g	carb
Mean	1.132577e+05	2.164699	0.015855	0.053477	1014.706837	9.852145	3.190077	
Variance	5.946512e+09	6.881853	0.016181	0.066744	485771.359675	125.641417	17.730491	
Standard deviation	7.711363e+04	2.623329	0.127205	0.258349	696.972998	11.208988	4.210759	
Median	1.024675e+05	1.000000	0.000000	0.000000	962.000000	5.260000	1.200000	
Symetrie measurement	7.146788e-01	2.301467	8.168593	5.825033	0.289084	1.259932	1.420214	
Flattening measurement	-6.450198e-02	8.553923	68.825367	43.493935	-1.180603	1.023922	1.175880	

Tableau Synthétique des propriétés statistiques des variables quantitatives

Exploration des données:Analyse univariée



Exploration des données:Analyse multivariée

Liste des variables fortement corrélées

-Une forte corrélation positive de la variable energy_100g avec fat_100g et carbohydrates_100g.

-Une forte corrélation positive entre la variable fat_100g et saturated-fat_100g.

-Une forte correlation positive entre la variable carbohydrates_100g avec additives_n, sugars_100g, fiber_100g, proteins_100g.

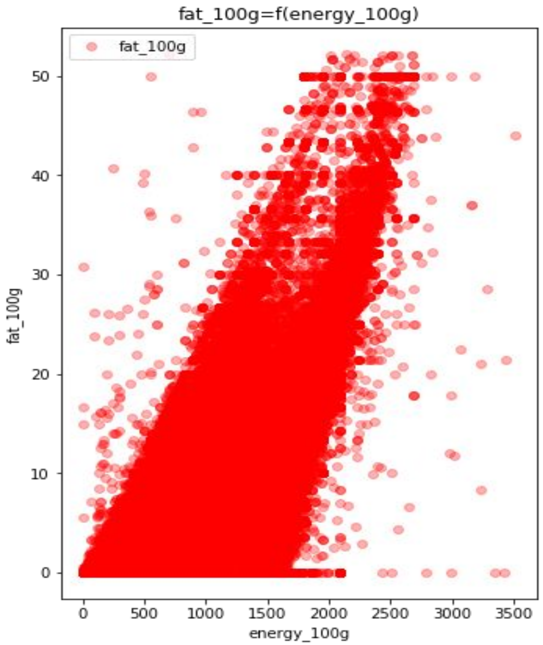
-Une forte correlation positive entre salt_100g et sodium_100g.

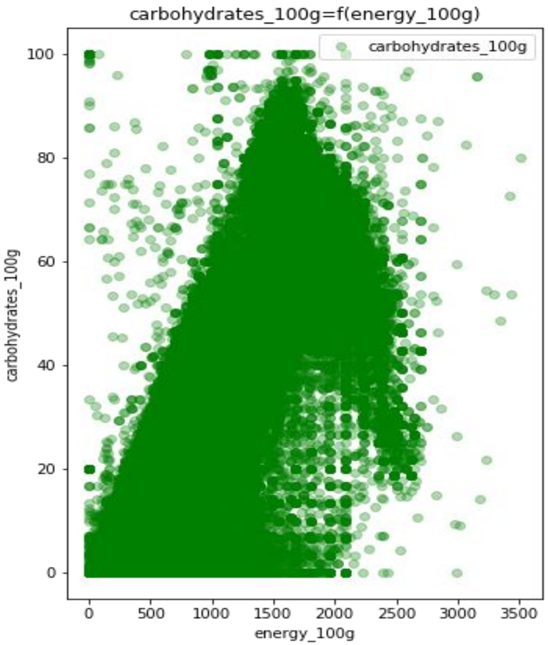
-Une forte correlation positive entre la variable nutrition-score-fr_100g avec energy_100g, fat_100g , saturated-fat_100g, sugars_100g et nutrition-score-uk_100g.

-Une forte correlation positive de la variable nutrition-score-uk_100g avec energy_100g, fat_100g , saturated-fat_100g, sugars_100g.

Exploration des données:Analyse multivariée

energy_100g

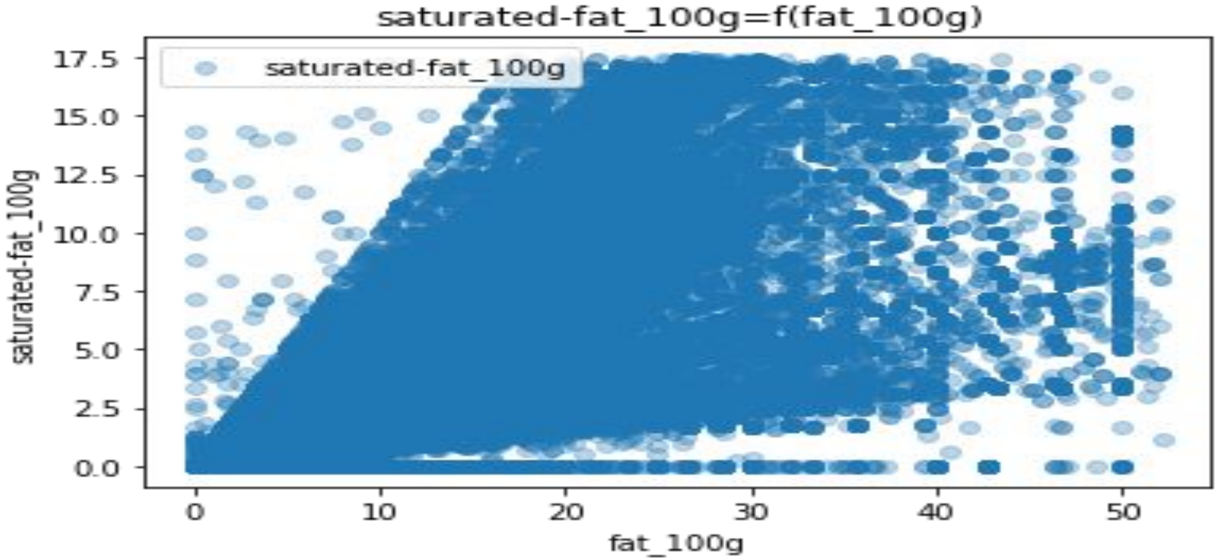




Ces graphiques ne permettent pas d'identifier aucun modèle qui peut régir la relation entre ces variables.

Exploration des données:Analyse multivariée

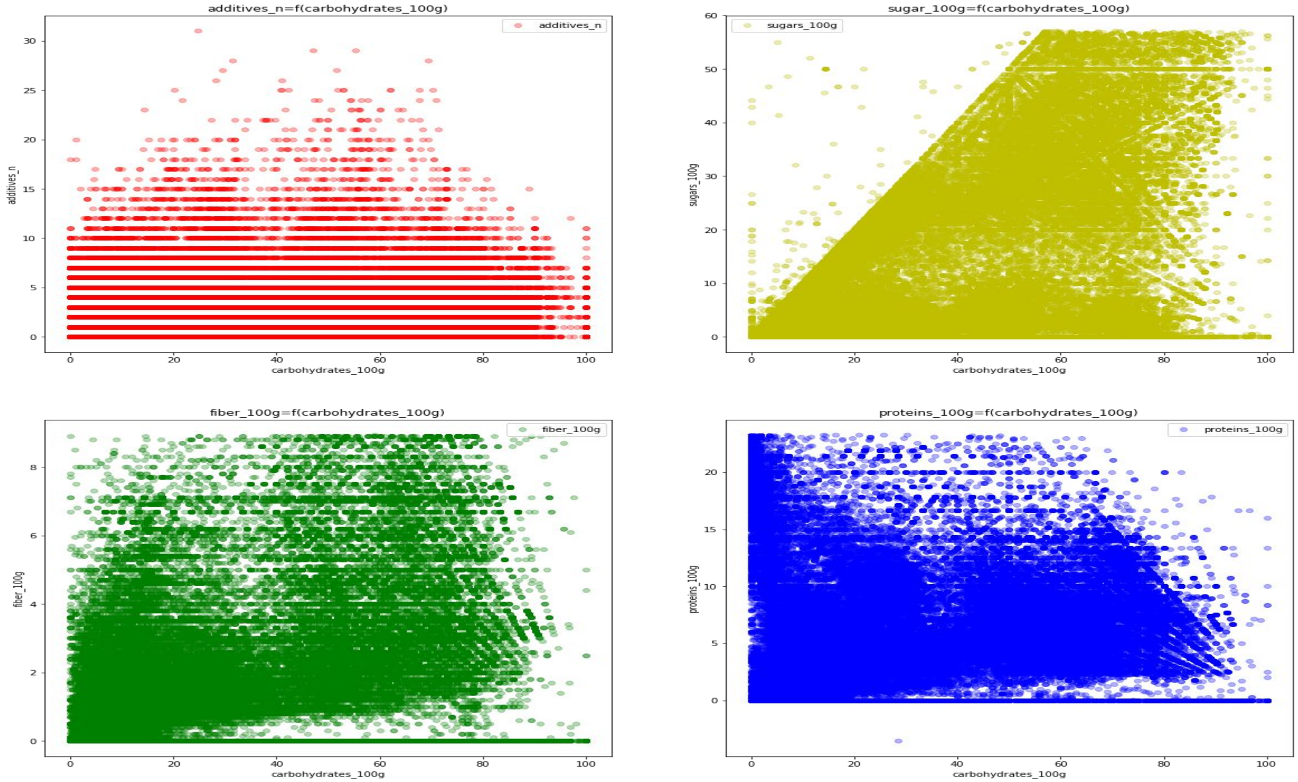
fat_100g



Le graphique ne permet pas d'identifier aucun modèle ,régissant la relation entre ces deux variables.

Exploration des données:Analyse multivariée

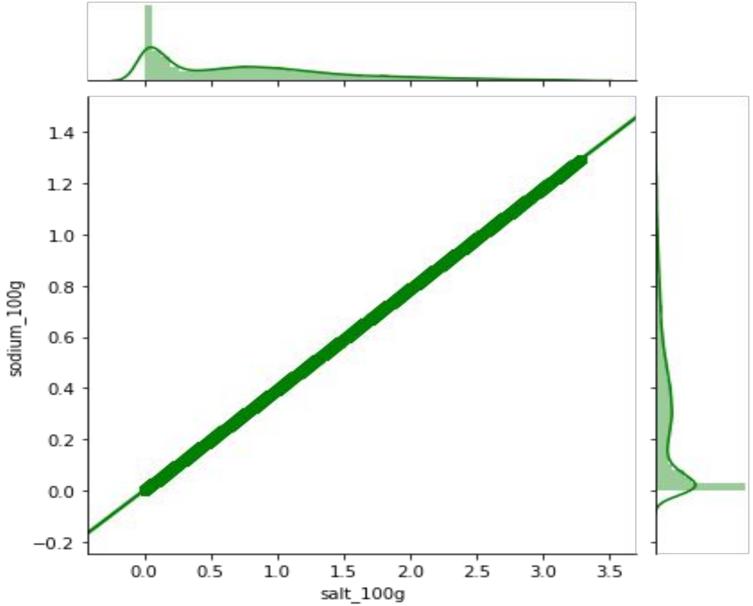
carbohydrates_100g



La corrélation de carbohydrates_100g avec ces différents variables ne pourrait être régie par un modéle standard.

Exploration des données:Analyse multivariée

salt_100g



Une parfaite relation linéaire entre salt_100g et sodium_100g.

Exploration des données:Analyse multivariée

salt_100g

---> estimation des paramètres de la relation linéaire

```
In [32]: #We will estimate the coefficient of the best linear regression between sodium_100g and salt_100g.
         a=np.cov(data['sodium_100g'],data['salt_100g'])[1,0]/data['salt_100g'].var(ddof=1)
         b=data['sodium_100g'].mean()-a*data['salt_100g'].mean()
         print("a={}".format(a))
         print("b={}".format(b))
```

```
a=0.3937004591053414
b=2.7587459594524688e-08
```

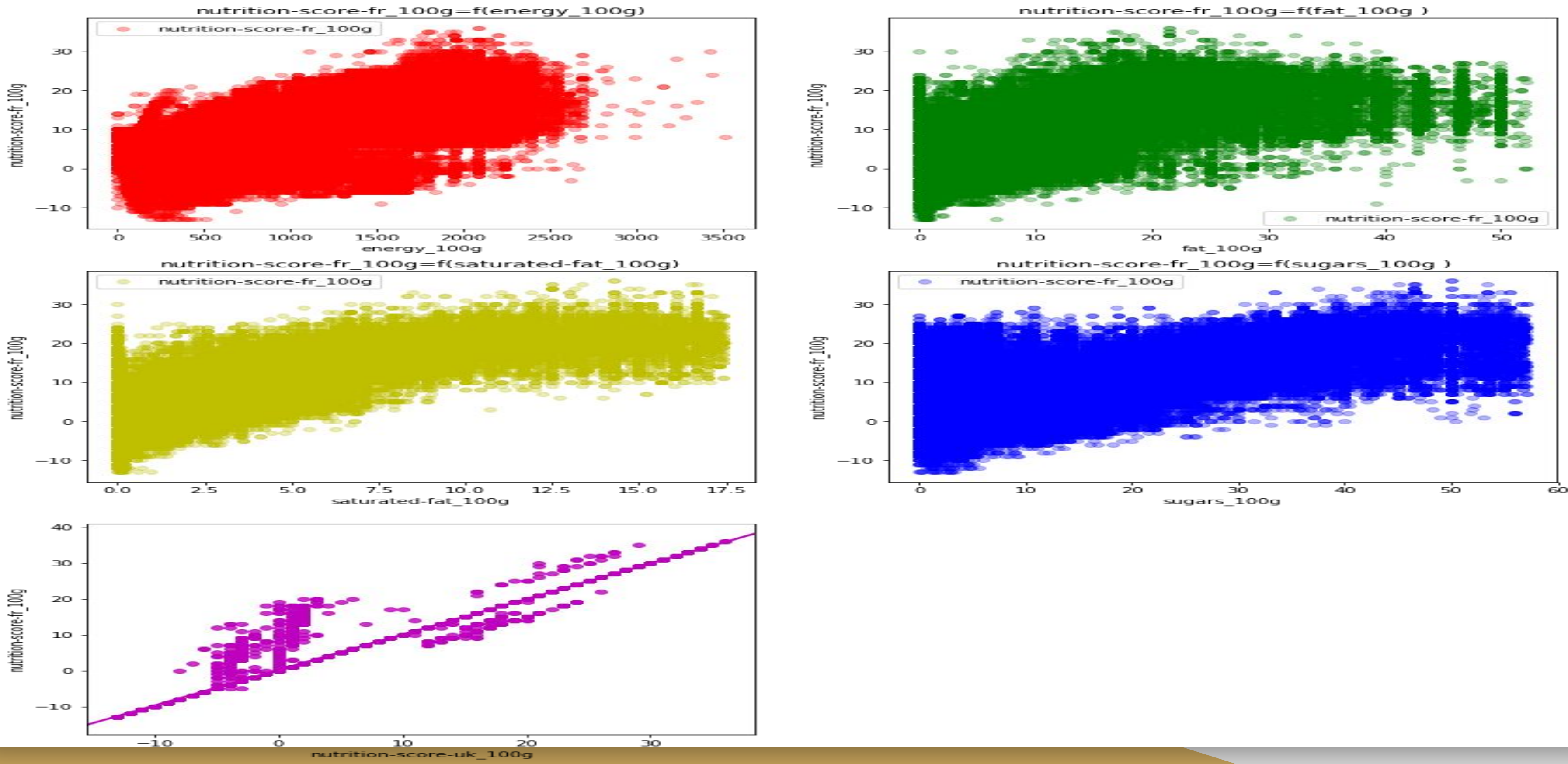
We can now write sodium_100g in function of salt_100g with the following lineair function:

sodium_100g=a*salt_100g+b

sodium_100g = 0.4*salt_100g+2.75

Exploration des données:Analyse multivariée

nutrition_score-fr_100g



Exploration des données: Analyse multivariée

nutrition_score-fr_100g

Le graphes de nutrition_score-fr_100g en fonction séparément de variables energy_100g, fat_100g, saturated-fat_100g,sugars_100g montre:

➔ La fonction de nutrition_score-fr_100g en fonction de l'un de ces variables, n'est pas une application.

➔ Tous les graphes ont quasiment la même forme.

➔ Une très forte corrélation linéaire entre nutrition_score-fr_100g et chacune de ces variables.

⇨ **Il faux chercher une relation ,permet de lier nutrition_score-fr_100g avec simultanément energy_100g, fat_100g , saturated-fat_100g et sugars_100g.**

Exploration des données:Analyse multivariée

nutrition_score-fr_100g

Recherche d'un modèle linéaire

```
In [34]: from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import train_test_split
         X=data[['energy_100g','fat_100g','saturated-fat_100g','sugars_100g']]
         Y=data['nutrition-score-fr_100g']
         x_train,x_test,y_train,y_test=train_test_split(X,Y,train_size=0.5)
         regressor=LinearRegression()
         regressor.fit(x_train,y_train)
```

```
/home/taher/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning: From versi
on 0.21, test_size will always complement train_size unless both are specified.
  FutureWarning)
```

```
Out[34]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [35]: #We will calculate les coefficients de regression linéaire de
         #nutrition-score-fr_100g=f('energy_100g','fat_100g','saturated-fat_100g','sugars_100g')
         df_coefficient=pd.DataFrame(regressor.coef_,X.columns,columns=["coefficients"])
         df_coefficient
```

Out[35]:

	coefficients
energy_100g	0.000903
fat_100g	0.182105
saturated-fat_100g	0.842614
sugars_100g	0.217660

Exploration des données:Analyse multivariée

nutrition_score-fr_100g

Robustesse de modèle

In [37]:
```
#We will now test the robustness of the model.
from sklearn import metrics
regression_score=metrics.r2_score(y_test,y_pred)
explained_variance_score=metrics.explained_variance_score(y_test,y_pred)
pd.DataFrame([regression_score,explained_variance_score],index=["regression_score","explained_variance_score"],colum
```

Out[37]:

	robustesse
regression_score	0.751228
explained_variance_score	0.751238

So we can consider that the model , which defined by:

0.000971 *energy_100g+0.177920* fat_100g+0.842 *saturated-fat_100g+0.216* sugars_100g
as a model which can allow us to predict the value of nutrition-score-fr_100g of one product, from values of simultaneously energy_100, fat_100g, saturated-fat_100g and sugars_100g.

nutrition_score-fr = 0.001*energy+0.178*fat+0.842*saturated-fat+0.216*sugars

Exploration des données:Analyse multivariée

nutrition_score-fr_100g

Exploration de la relation linéaire entre nutrition_score-fr_100g et nutrition_score-uk_100g.

```
In [38]: #Estimate the parametrs of the best linear regression between nutrition-score-fr and nutrition-score-uk_100g.
         c=np.cov(data["nutrition-score-fr_100g"],data["nutrition-score-uk_100g" ])[1,0]/data["nutrition-score-uk_100g"].var(
         β=data["nutrition-score-fr_100g"].mean()-c*data["nutrition-score-uk_100g" ].mean()
         print("c={}".format(c))
         print("β={}".format(β))

         c=0.9897134702535476
         β=0.16229073987685005
```

We can now write nutrition-score-fr_100g in function of nutrition-score-uk_100g with the following lineair function:
nutrition-score-fr_100g= c * nutrition-score-uk_100g + β

nutrition_score-fr_100g = 0.99*nutrition_score-uk_100g+0.162

Feature engineering

Une nouvelle variable , appelée n_ingredients, compte le nombre des ingrédients utilisées pour chaque produit.

We have the column "ingredients_text", describe the list of ingredient which used each product. This column gives the different ingredients used, seperated by ",".So we will define a function, which can underline the number of ingredients used in a product from a text.

In [39]:

```
# define function,which can underline a number of string seperated by the symbol",".
def n_str(chaine,c):
    """This function ,allow to calculate the number of string seperated by the symbol c.
       The symbol c introduced as a parameter in the function.

       Args:

       chaine(str): This is the string that the function will examinate to calculate the number of substrings
                    seperated by the symbol introduced as a parametre of the function.
       c(str): this the symbol , which seperate substrings in the introduced string.

       Returns:

       numbre(int): the number of substrings, which seperated by the introduced symbole  in the introduced string.
    """
    li=chaine.split(c)
    return len(li)
```

In [40]:

```
data["n_ingredients"]=[n_str(c,",")for c in data['ingredients_text']] #introduce the variable of n_ingredients
```

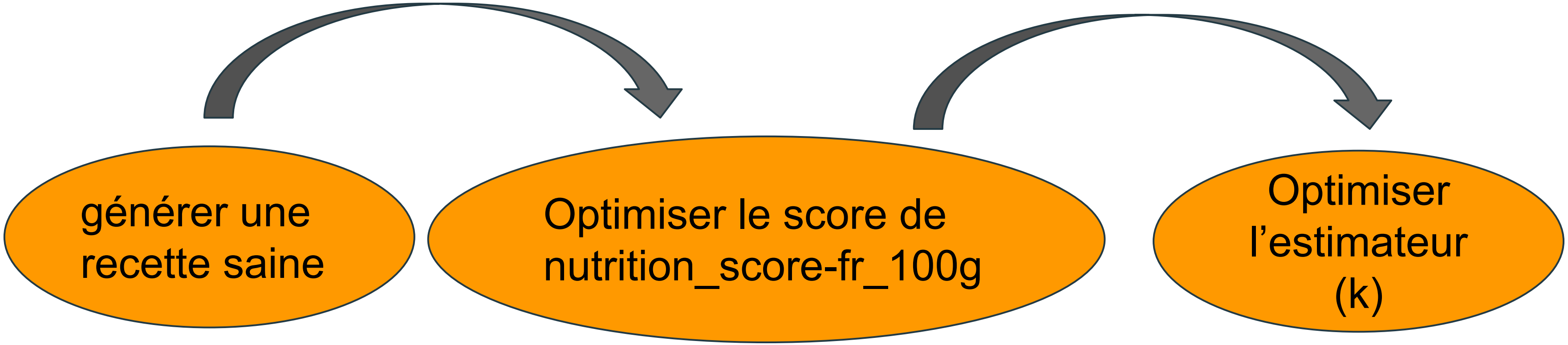
In [41]:

```
data["n_ingredients"].head()# the first values of the column n_ingredients.
```

Out[41]:

```
0    13
1    25
2    12
3    22
4    13
Name: n_ingredients, dtype: int64
```

Propositions pour le site La marmite



(k):

nutrition_score-fr = 0.001*energy+0.178*fat+0.842*saturated-fat+0.216*sugars

Conclusion

- 80% des produits commercialisées ne sont pas très sains.
- La quantité d'énergie dans un produit alimentaire est fortement corrélée avec séparément la quantité de graisse et la quantité de glucides.
- La quantité de graisse dans un produit alimentaire est fortement corrélée avec la quantité de gras saturé.
- La quantité de glucides dans un produit alimentaire est fortement corrélée avec séparément la quantité de sucres, fibres, protéines et le nombre des additifs utilisés dans ce produit .
- La quantité de sel dans un produit alimentaire est reliée avec la quantité de sodium par une relation linéaire.
- Nous pouvons inférer l'aspect sain d'un produit tout en calculant le score déterminé par: ***0.001*energy+0.178*fat+0.842*saturated-fat+0.216*sugars***