Parcours Data Scientist : Projet 3

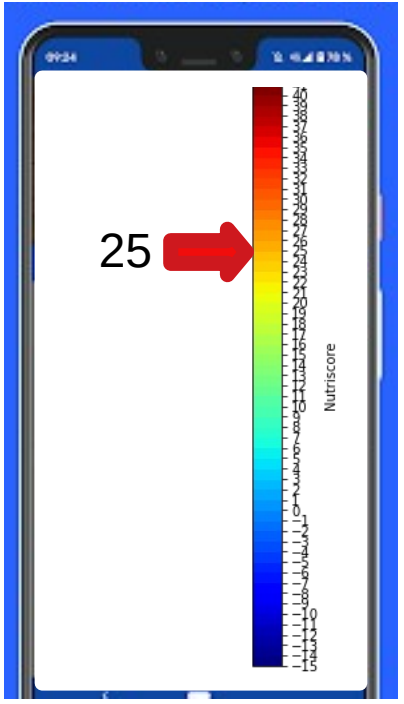
Concevez une application au service de la santé publique

Sommaire

- Idée d’application
- Nettoyage des données.
- Analyse uni-variée des différentes variables importantes
- Analyse multivariée
- Pertinence et faisabilité

Idée d’application

Grâce à la détection de code-barres, vous obtiendrez le Nutri-score du produit



Idée d’application

Principe :

- Des données numériques :
  - Des nutri-scores disponibles pour un grand jeu de donnée
  - Une grande partie de données numériques sur la composition et les valeurs énergétiques des produits
- Régression linéaires sur les données des aliments pour obtenir le nutri-score entre -15 et 40

Nettoyage des données

- Présentation des données
  - Général :

```
1 raw_data.shape
```

```
(1481675, 182)
```

  - Features :
    - Une dizaine de données générales : On va garder « code » et « nom de produit »
    - 25 catégories essentiellement des strings sur les origines des produits, leurs labels, le packaging…
    - 25 sur les ingrédients, les allergènes, les additifs… On va conserver les colonnes nutriscore-score et nutriscore-grade et le nova-group qui est une donnée chiffrée également
    - Les données nutritives (float) avec leurs poids pour 100g de produit

Nettoyage des données

- On commence par supprimer les colonnes qui ne nous seront pas utiles :

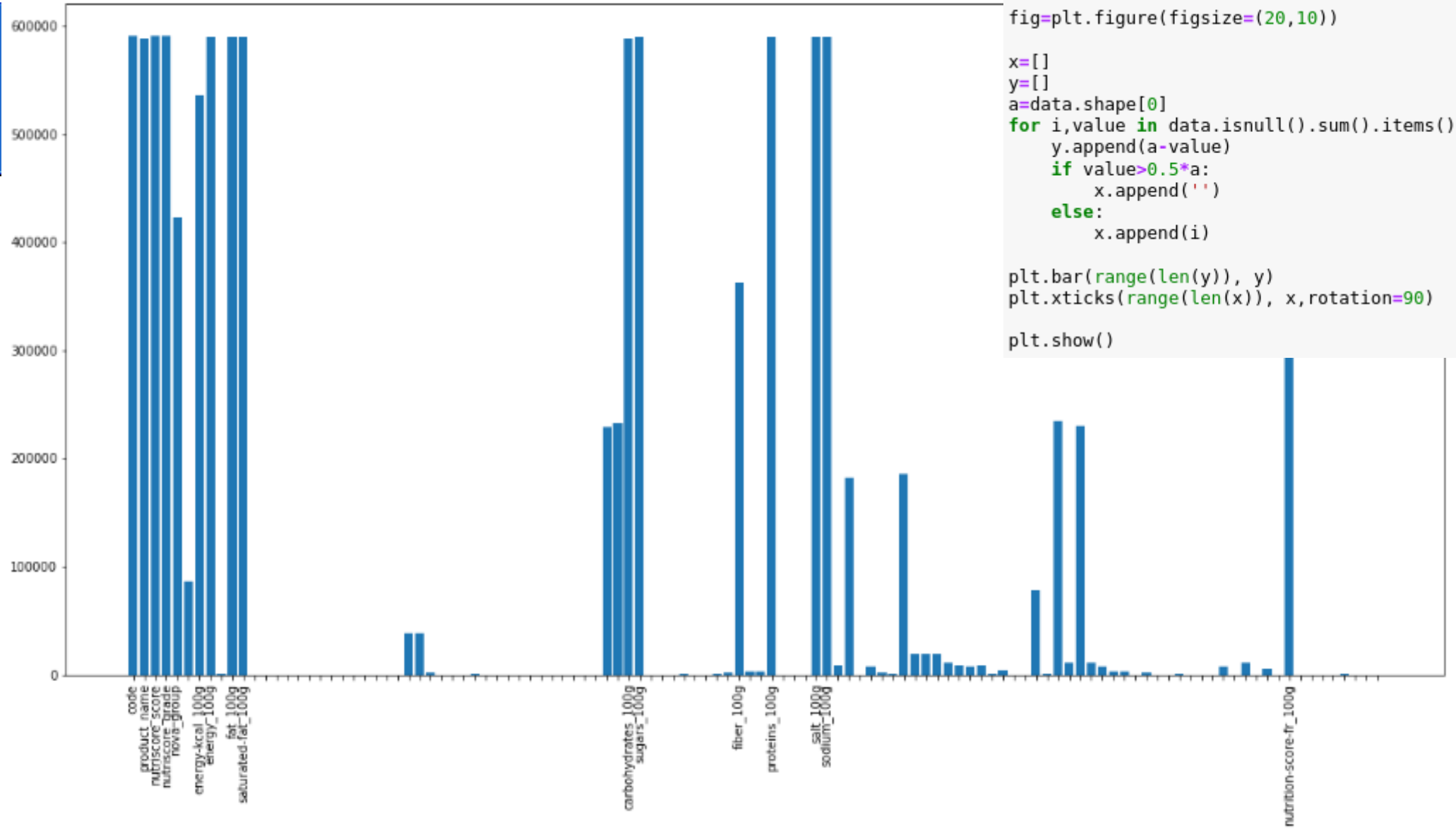
```
L=['code','product_name','nutriscore_score','nutriscore_grade','nova_group']
for i in raw_data.columns:
    if '_100g' in i and 'carbon-footprint' not in i:
        L.append(i)
data=raw_data[L]
```

- On supprime les lignes qui ne sont pas labellisées :

```
1 data.dropna(subset=['nutriscore_score'],inplace=True)
2 data.shape
```

```
(590618, 114)
```

- On regarde la disponibilité des informations pour les variables restantes et on ne garde que celles disponibles pour plus de la moitié de l'échantillon:




```
fig=plt.figure(figsize=(20,10))

x=[]
y=[]
a=data.shape[0]
for i,value in data.isnull().sum().items():
    y.append(a-value)
    if value>0.5*a:
        x.append('')
    else:
        x.append(i)

plt.bar(range(len(y)), y)
plt.xticks(range(len(x)), x,rotation=90)

plt.show()
```

Nettoyage des données

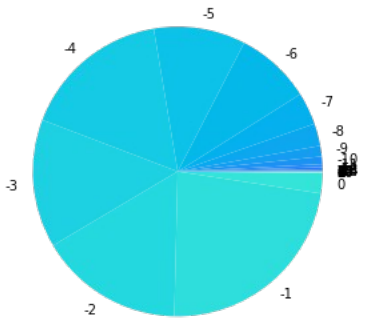
- On voit que dans les features « energy » les données sont dupliquées et seule la colonne « energy_100g » est importante (cf fichier Jupyter)
  - On ne garde donc que « energy_100g »

- On voit qu'il n'y a aucune corrélation directe entre le nutriscore_score et le nutriscore_grade ou entre le nutriscore_score et le nova_group ce sont des données probablement également corrélées aux données chiffrés d'ingrédients, on va donc les supprimer également.

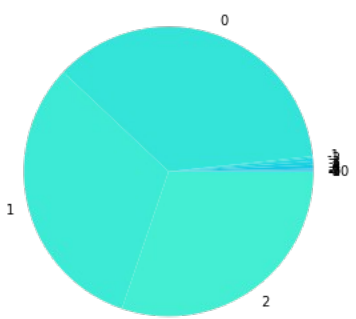
Nettoyage des données

Répartition des nutriscores par grade

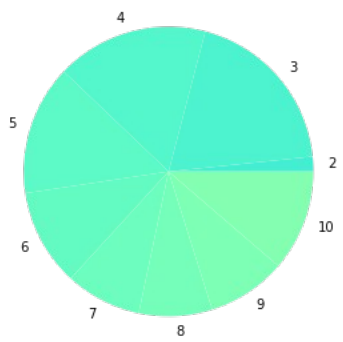
grade: a



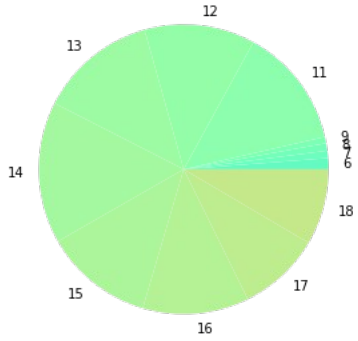
grade: b



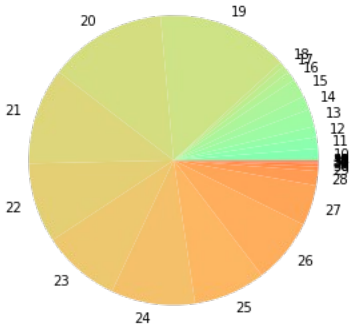
grade: c



grade: d

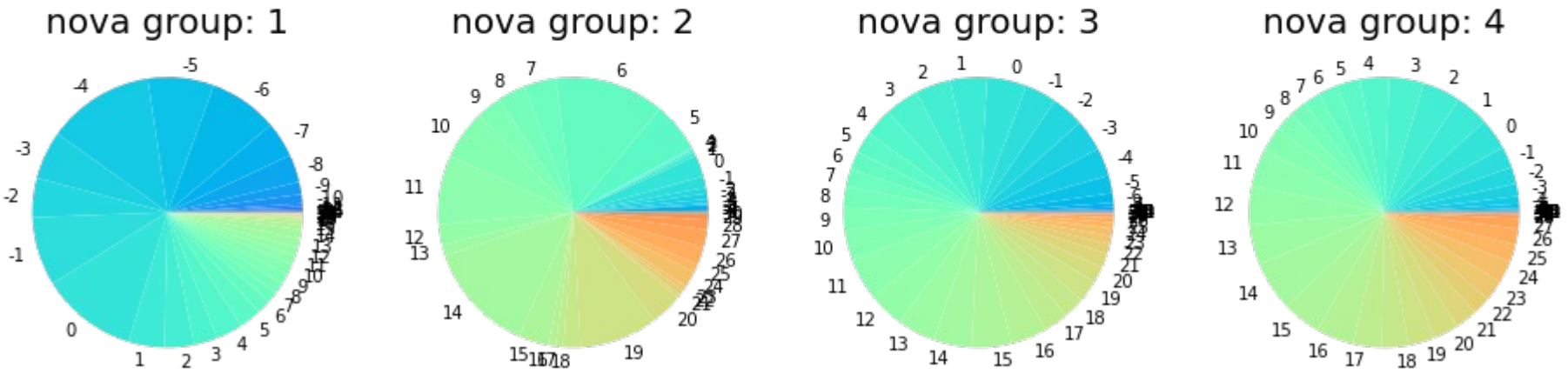


grade: e



Nettoyage des données

Répartition des nutriscores par nova group



On supprime donc le nutriscore_grade et le nova_group de notre jeu de données

Nettoyage des données

- Suppressions des doublons :

```
data3=data2.copy()
#Je cherche les produit ayant meme code barre
dupl=data3[data3.duplicated(subset='code',keep=False)].sort_values(by='code')
#Je complète les éventuelles cellules vides par les valeurs du doublon
for i in dupl.code.unique():
    na=data3[data3['code']==i].isnull()
    L=na.index.tolist()
    for k in na.columns:
        for j in range(2):
            if na.loc[L[j],k]:
                if not na.loc[L[(j+1)%2],k]:
                    data3.loc[L[j],k]=data3.loc[L[(j+1)%2], k]
#Je supprime les doublons sur les codes barres:
data3.drop duplicates(subset='code',inplace=True)
```

Nettoyage des données

- Gestion des outliers
  - On supprime tous les ingrédients qui n'ont aucune précisions en ingrédients :

```
data3.dropna(subset=data3.columns[3:],how='all',inplace=True)
```

  - On supprime toutes les valeurs au dessus de 100g et en dessous de 0g pour les ingrédients :

```
data4=data3.copy()
#Je considère que les Nan sont des 0
data4.fillna(0,inplace=True)
for i in data4.columns.tolist()[4:]:
    data4=data4[data4[i]>=0]
    data4=data4[data4[i]<=100]
```

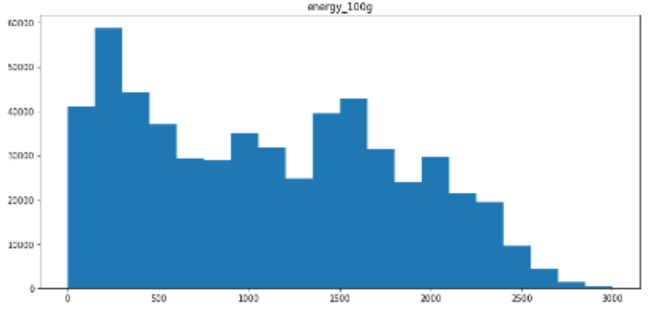
  - Maintenant, pour supprimer les autres valeurs extrêmes on va tracer les histogramme de ces variables et par tâtonnement établir un seuil pour chaque variable

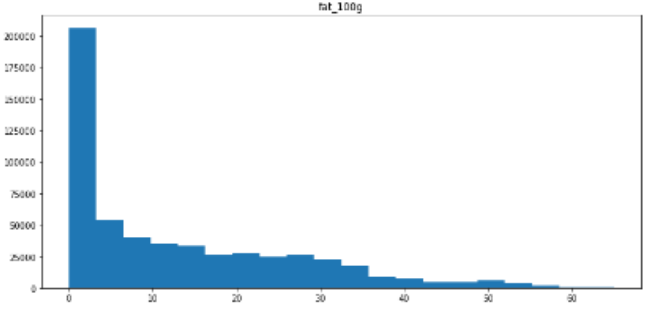
Nettoyage des données

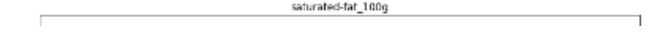
```
#Je commence par fixer mes seuils à au moins 10*Q3 et on verra ensuite en fonction des histogrammes
outliers={'energy_100g':3000,'fat_100g':65,'saturated-fat_100g':35,\
          'carbohydrates_100g':100,'sugars_100g':80,'fiber_100g':15,\
          'proteins_100g':50,'salt_100g':10,'sodium_100g':3}
data5=data4.copy()
for key,value in outliers.items():
    data5=data5[-(data5[key]>value)]
#Je vérifie que mon taux de valeurs supprimées n'est pas trop élevé
print(data5.shape[0]/data4.shape[0])
```

```
0.9435181665127899
```

```
fig=plt.figure(figsize=(28,56))
x=np.array(data5['nutriscore_score'])
k=1
for i in data5.columns.tolist()[3:]:
    fig.add_subplot(8,2,k)
    y=np.array(data5[i])
    plt.hist(y,bins=20)
    plt.title(i)
    k+=1
```

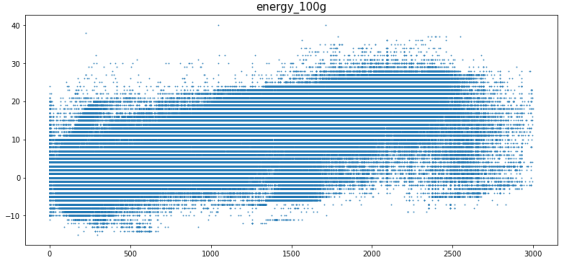


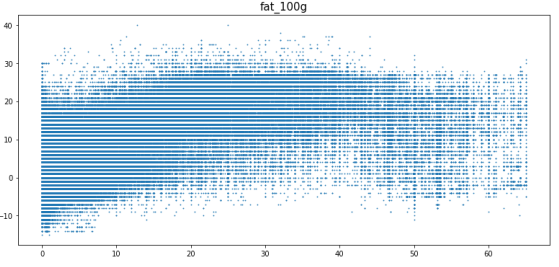


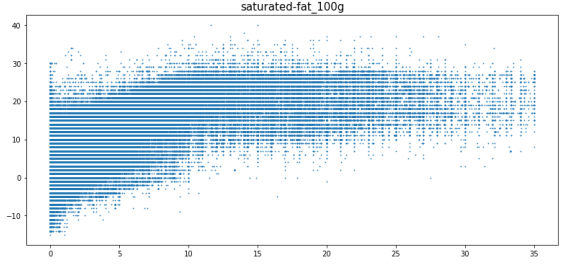


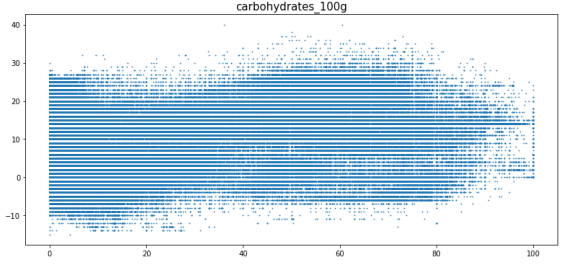
Analyse uni-variée des différentes variables importantes

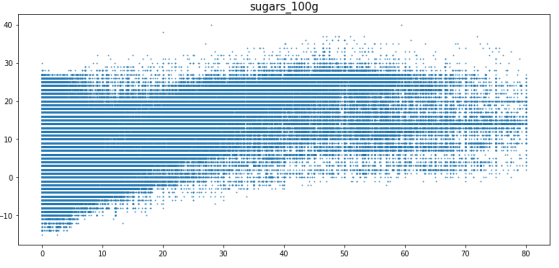
Nutriscore en fonction des compositions

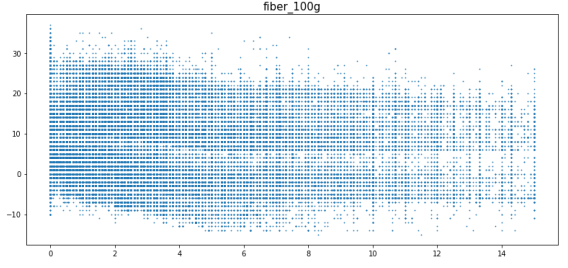


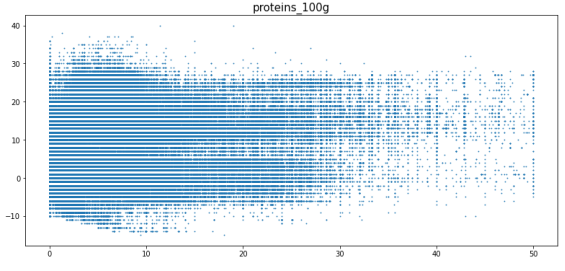


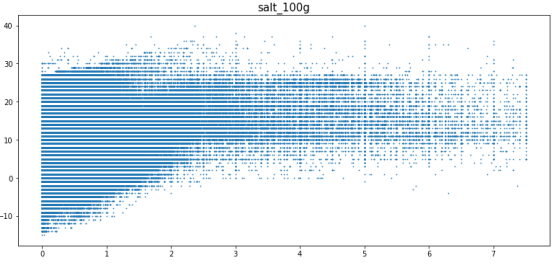


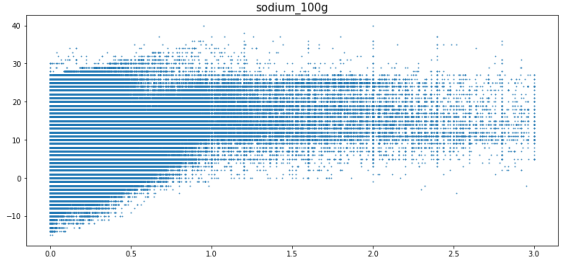












Analyse uni-variée

- Les courbes du sel et du sodium semblent également étrangement similaire. On calcule leur coefficient de corrélation et en effet elles sont extrêmement corrélées:

```
#Le potassium et les carbohydrates semblent inutiles
#Le sel et le sodium ont exactement les mêmes courbes:
import scipy.stats as st
st.pearsonr(data5['salt_100g'],data5['sodium_100g'])[0]
```

```
0.9999827946133235
```

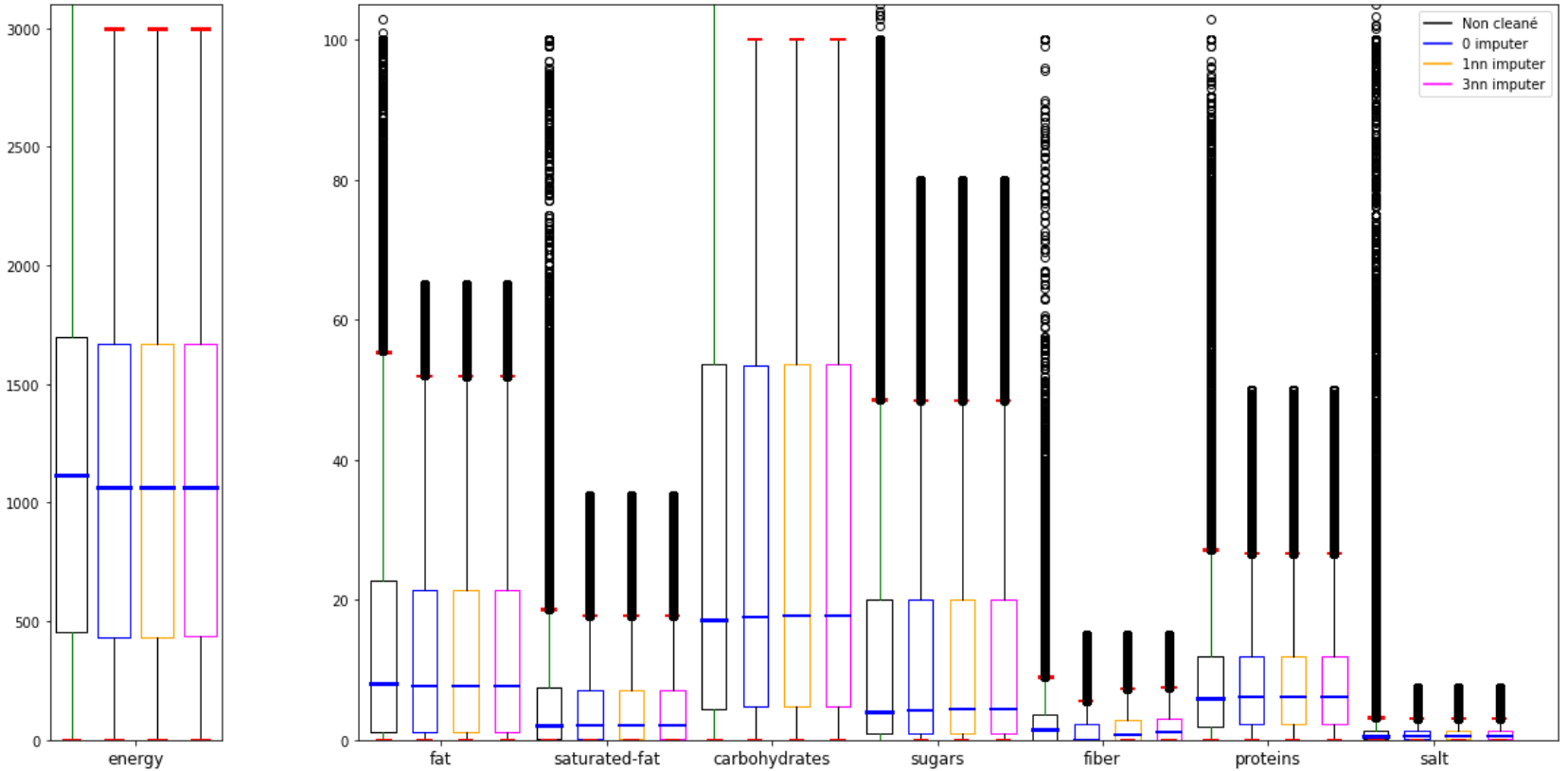
- On a deux variables bi-modales : les protéines et les fibres
- Le sel, les graisses saturées et le sucre sont plutot uni-modales
- Les autres variables sont plus dispersées.
- On voit une corrélation entre la valeur énergétique du produit et le nutriscore (plus le produit est énergétique plus le nutriscore est important)
- De même avec le sucre, le sel et les graisses
- Les fibres ont plutot tendances à faire baisser le nutriscore.
- Les carbohydrates ne semblent pas avoir d'impact sur le nutriscore.

Analyse uni-variée

- Avant de passer à une analyse multivariée, on va compléter les valeurs manquantes.
- On a utilisé plusieurs options :
  - Des zéros en admettant que si un ingrédient n'avait aucune valeur pour certains ingrédients mais pas pour d'autres c'était que les valeurs étaient nulles
  - Deux KNN imputers (en séparant ingrédient et énergie):
    - 1nn
    - 3nn
- On compare les données avant et après nettoyage/imputation :

Analyse uni-variée

Diagrammes en boite avant-après nettoyage



Analyse Multivariée

- On part sur le 3nn imputer, les données des fibres y sont moins écrasées que pour les autres.
- On regarde les coefficients de corrélation entre les différentes variables

Analyse multi-variée

```
col=final.columns.tolist()[2:]
corel=pd.DataFrame(col)
correlation=[]
for i in col:
    corele=[abs(st.pearsonr(final[i],final[j])[0]) for j in col if j not in correlation]
    corele=[0 for i in range(len(correlation))]+corele
    corel[i]=np.array(corele)
    correlation.append(i)
```

```
corel.set_index(0)
corel.style.background_gradient(cmap='Blues')
```

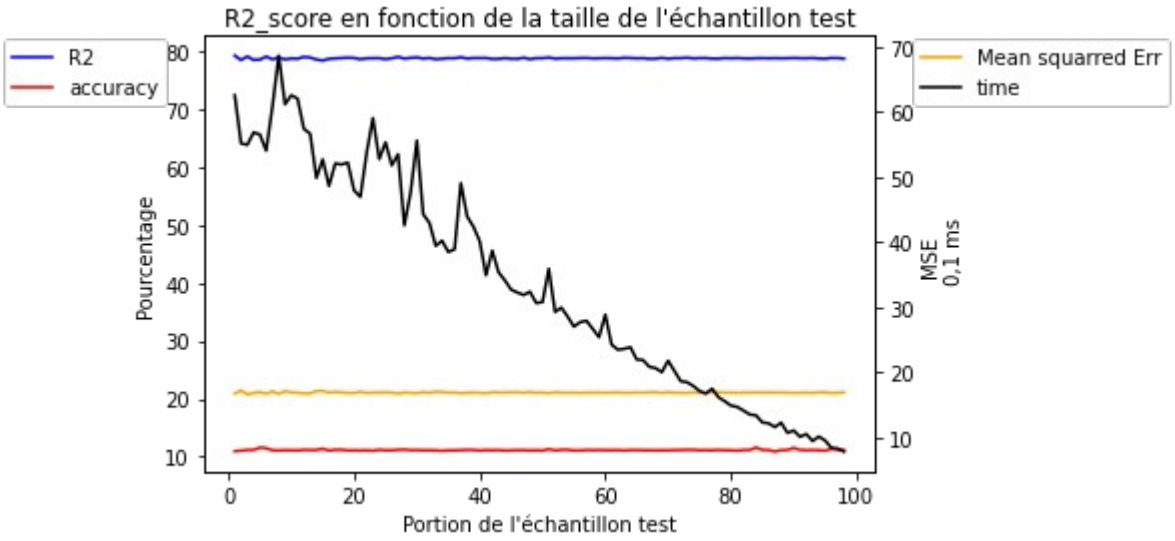
0	nutriscore_score	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g
nutriscore_score	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
energy_100g	0.628983	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
fat_100g	0.600338	0.771090	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
saturated-fat_100g	0.666869	0.583960	0.747319	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
carbohydrates_100g	0.279391	0.577551	0.003733	0.000527	1.000000	0.000000	0.000000	0.000000	0.000000
sugars_100g	0.466238	0.374054	0.048953	0.156138	0.642175	1.000000	0.000000	0.000000	0.000000
fiber_100g	0.103952	0.401660	0.210517	0.007838	0.384202	0.081319	1.000000	0.000000	0.000000
proteins_100g	0.121183	0.323396	0.373075	0.304915	0.197114	0.300756	0.100274	1.000000	0.000000
salt_100g	0.329705	0.105241	0.194505	0.101140	0.170973	0.282526	0.064406	0.377972	1.000000

Analyse multi-variée

- On avait pas vu de lien entre les carbohydrates et le nutriscore et ces derniers semblent relativement corrélés avec les sucres.
- De même le lien graisses-nutriscores étaient beaucoup moins prononcés que les lien graisses saturées et nutri-scores. Et graisses et graisses saturées sont bien corrélés ainsi que graisses et énergie.
- Conclusion :
  - Pour éviter un sur-apprentissage on va effectuer la regression linéaire sur les données en ne prennant pas en compte ni les graisses ni les carbohydrates

Régressions linéaires

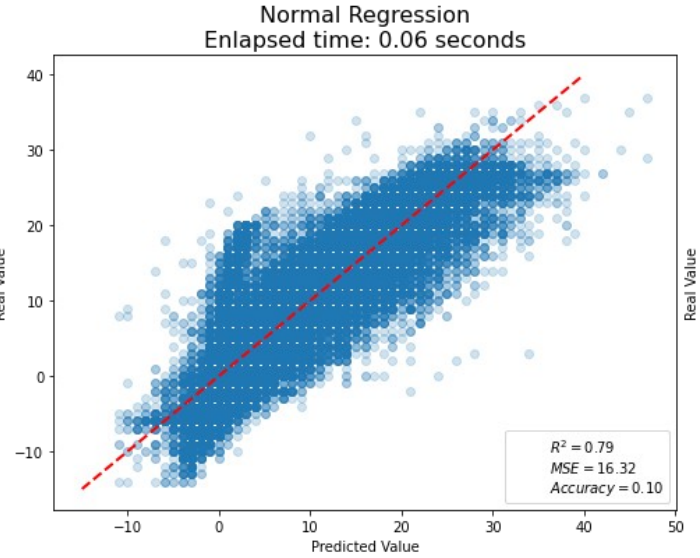
- Estimation de la taille de l'échantillon pour une simple régression linéaire :

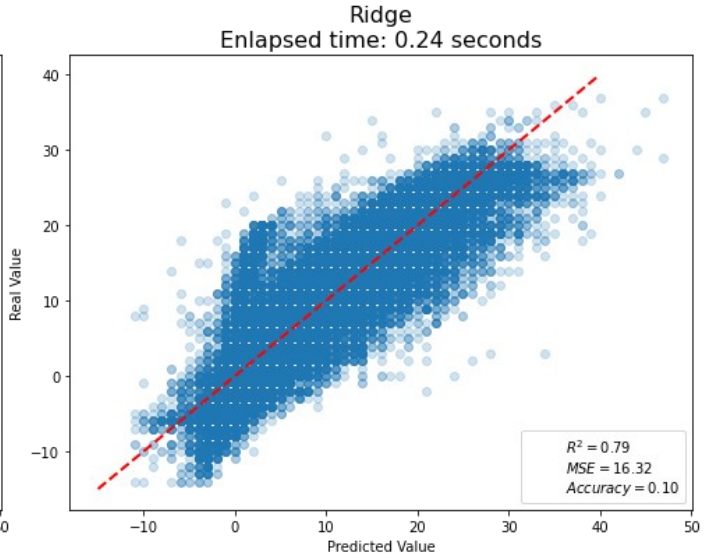


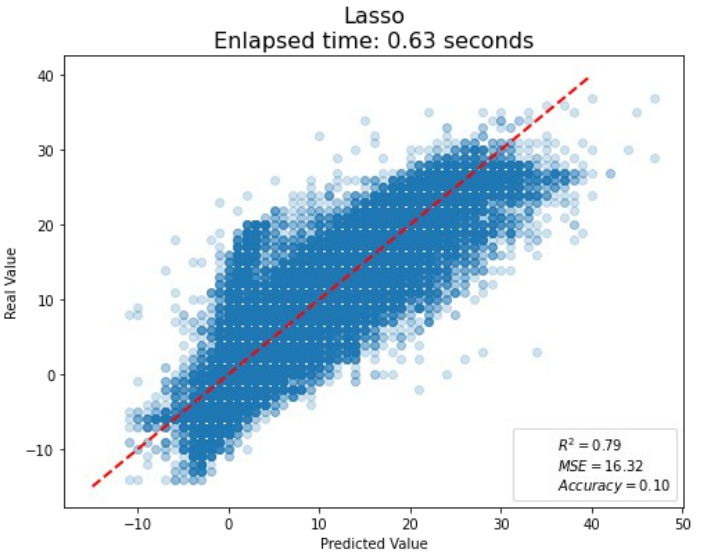

```
xtrain, xtest, ytrain, ytest = train_test_split(final[features], final[['nutriscore_score']],test_size=0.2)
```

Régressions linéaires

Comparaison avec des régressions Ridge et Lasso sur 5 folds







On a aucune différence on va donc prendre la régression linéaire simple

Bilan

- Pas de sur-apprentissage, la régression normale est donc plus adaptées.
- Manque de précision pour une application. Donne simplement des indications sur le score-nutritionnel. Une erreur moyenne de plus de 4 points.
- Besoin de plus de données pour affiner les résultats. Peut-être une prise en compte de données non chiffrées sur des catégories de produits.

Présentation de l’application

1 : On scanne le produit

2 : Une application obtient le code barre

3 : On récupère les informations produit dans la base de donnée :

```
produit=raw_data[raw_data.code=code_barre]
```

4 :On remplit les données manquante et on effectue le calcul du nutri-score

```
#Je selectionne uniquement les colonnes qui m'intéresse:
ingredients=['saturated-fat_100g','sugars_100g','fiber_100g','proteins_100g','salt_100g']
product=produit[['energy_100g']+ingredients]
#Je regarde si des cases sont vides:
if product.isnull().any().any():
#si que des nan
    if product.isnull().any().all():
        print("impossible à calculer, aucune valeur d'ingrédient ni valeur énergétique disponible")
    else:
#Je regarde si j'ai qq chose qui manque dans les ingrédients et effectue si besoin le knn imputer:
        if product[ingredients].isnull().any().any():
            df=final[ingredients]
            df=pd.concat([df,product[ingredients]])
            #j'effectue le knn imputer
            prod=imputer3.fit_transform(df)[-1,:]
            energy=np.array([product.iloc[0].values[0]]+list(prod))
        else:
            energy=product.values
#Je regarde si j'ai qq chose qui manque en énergie et effectue si besoin le knn imputer:
        if np.isnan(energy).any():
            df=final[['energy_100g']+ingredients]
            df=pd.concat([df,energy])
            #j'effectue le knn imputer
            produitplein=imputer3.fit_transform(df)[-1,:]
        else:
            produitplein=energy

#J'ai maintenant dans produitplein toutes les valeurs dont j'ai besoin.
#J'effectue ma régression linéaire et renvois le résultat
print(linreg.predict([produitplein]).round()[0][0])
```

```
19.0
```

5 : Une application affiche ce code proprement