SOUTENANCE DU PROJET 3

Concevez une application au service de la santé publique

Parcours Data scientist

Henrique DA COSTA

Openclassrooms – Mai 2021

Sommaire

1. Contexte et objectifs de la mission

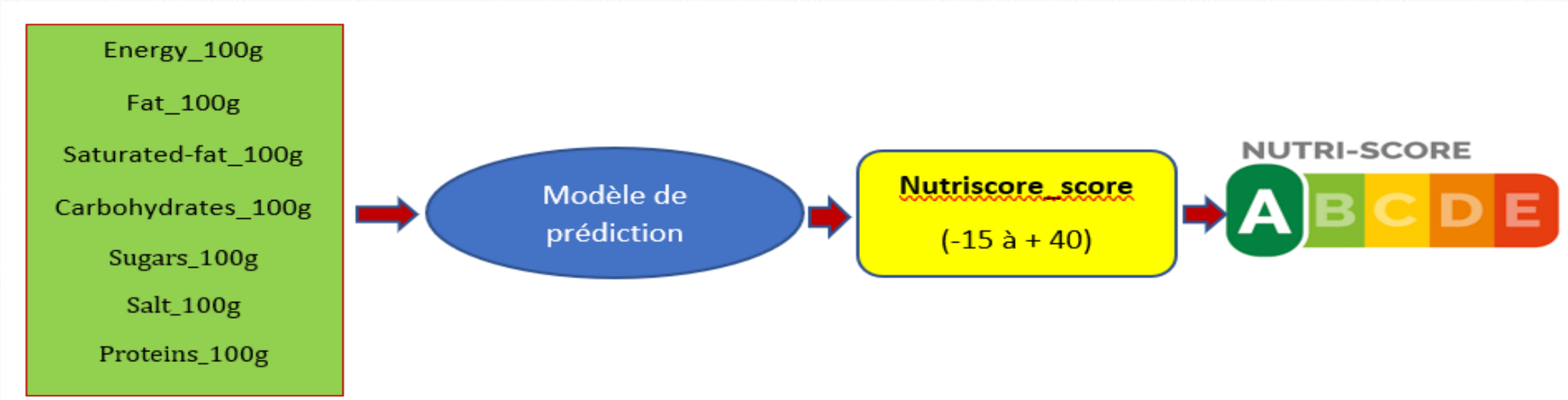
Contexte et objectifs de la mission

1. **Commanditaire de cette étude :** agence « Santé publique France »
2. **Appel à projets initié par l'agence "Santé publique France" :** trouver des idées innovantes d'applications en lien avec l'alimentation.
3. **Contenu de la mission :**
    - Traiter le jeu de données afin de ***repérer des variables pertinentes*** pour les traitements à venir. Automatiser ces traitements pour éviter de répéter ces opérations.
    - Produire des visualisations afin de mieux comprendre les données. Effectuer une ***analyse univariée*** pour chaque variable intéressante, afin de synthétiser son comportement.
    - Confirmer ou infirmer les hypothèses à l'aide d'une ***analyse multivariée***. Effectuer les tests statistiques appropriés pour vérifier la significativité des résultats.
    - ***Élaborer une idée d'application***. Identifier des arguments justifiant la faisabilité (ou non) de l'application à partir des données Open Food Facts. (https://world.openfoodfacts.org/)
    - Rédiger un rapport d'exploration et « pitcher » l'idée d'aplication.

2. Idée d’application

Idée d'application

- **Prédire le nutriscore à partir de certains éléments nutritionnels de base**



Score final	Couleur	Lettre
Entre - 15 et - 2	Vert	A
Entre - 1 et + 3	Vert clair	B
Entre + 4 et + 11	Jaune	C
Entre + 12 et + 16	Orange	D
Entre +17 et + 40	Rouge	E

Idée d'application

- Réaliser une application qui **aide le consommateur à améliorer ses habitudes de consommation et à contrôler ce qu'il mange,**
- Connaissant la teneur en nutriments à éviter et le taux de protéines (nutriment favorable à la santé et à la pratique du sport amateur) => l'idée est de permettre à tout un chacun et au sportif amateur en particulier de savoir si un aliment possède un score favorable à une bonne santé ou au contraire est à bannir.
- **Connaissance du nutriscore : modèle prédictif estimant le nutriscore de l'aliment à partir de la composition de ces quelques nutriments de base dans une fourchette de - 15 (aliment le plus sain) à + 40 (aliment le moins sain)** et de traduire cela par une lettre de A (le plus favorable) à E (à éviter absolument).
- Application légère permettant d'avoir une visualisation rapide du nutriscore pour manger de manière saine.
- Application ayant vocation à être déployé sur PC et sur mobile.

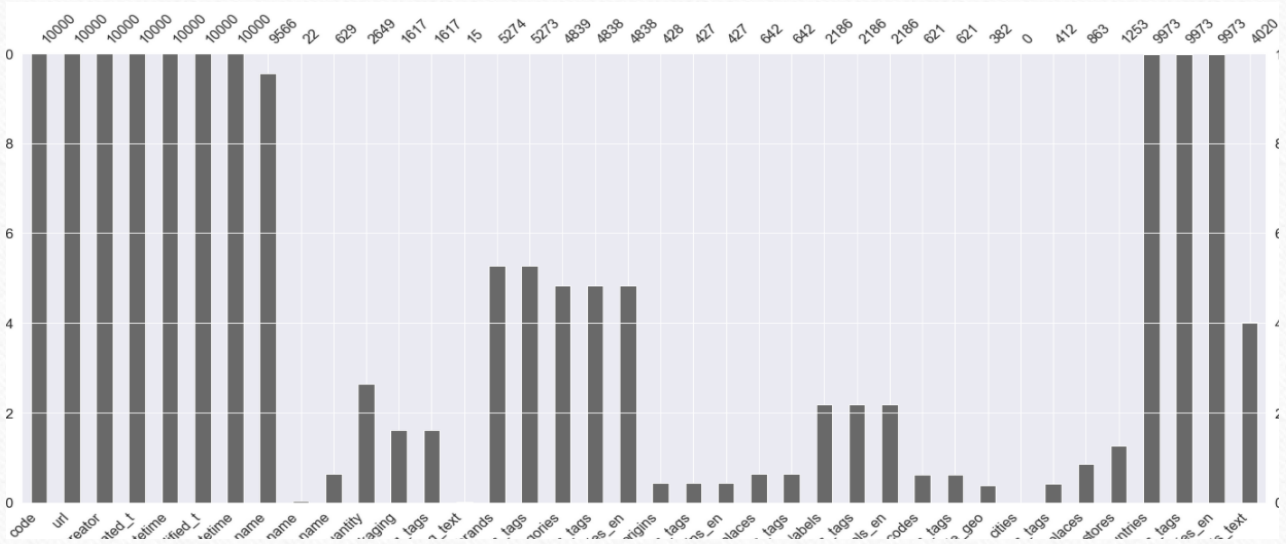
3. Nettoyage du jeu de données

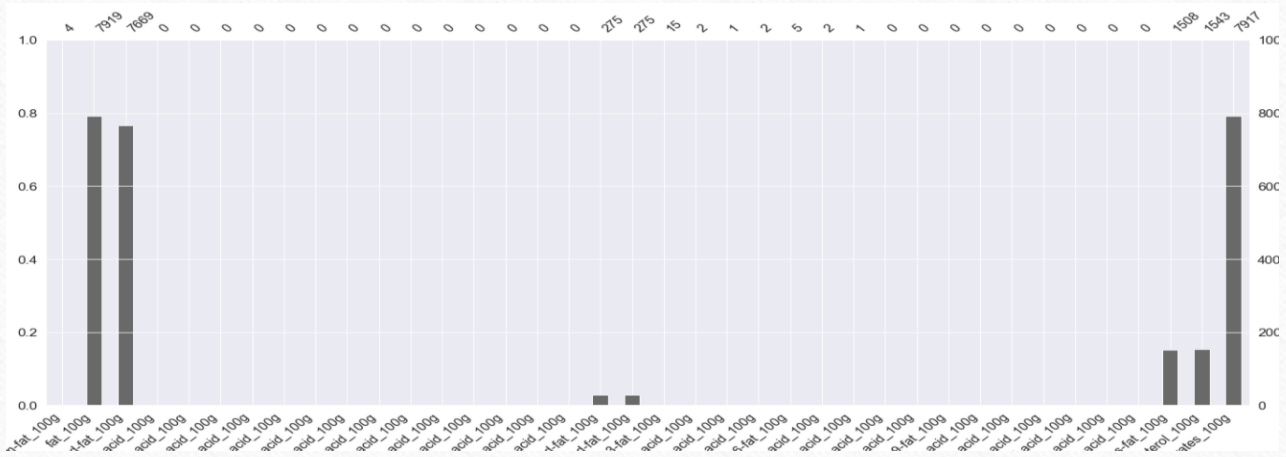
Nettoyage du jeu de données

Taille initiale du jeu : 1 724 219 lignes et 184 colonnes. Pèse 4 GB.

Après downcast des types : taille en mémoire ramené à 1,6 GB.

Nettoyage – Vue d'ensemble du jeu de données

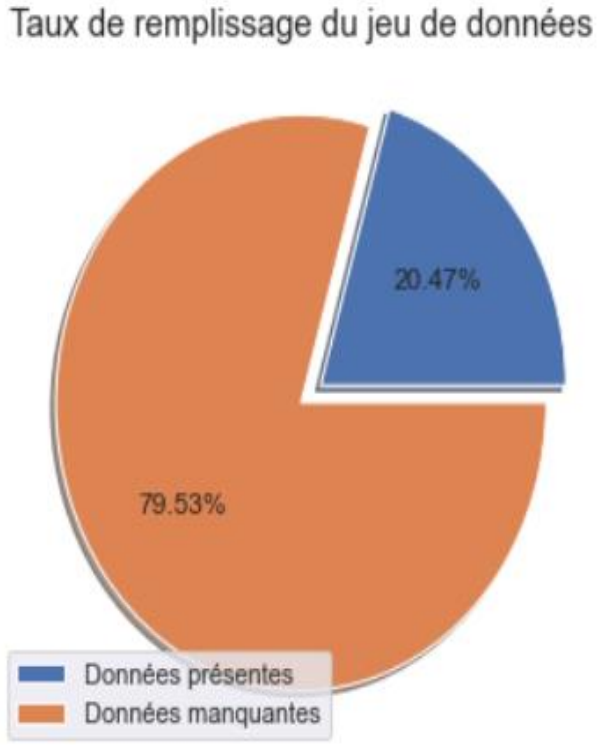




JEU DE DONNEES INITIAL

Le jeu de données a 1724219 lignes et 184 colonnes.

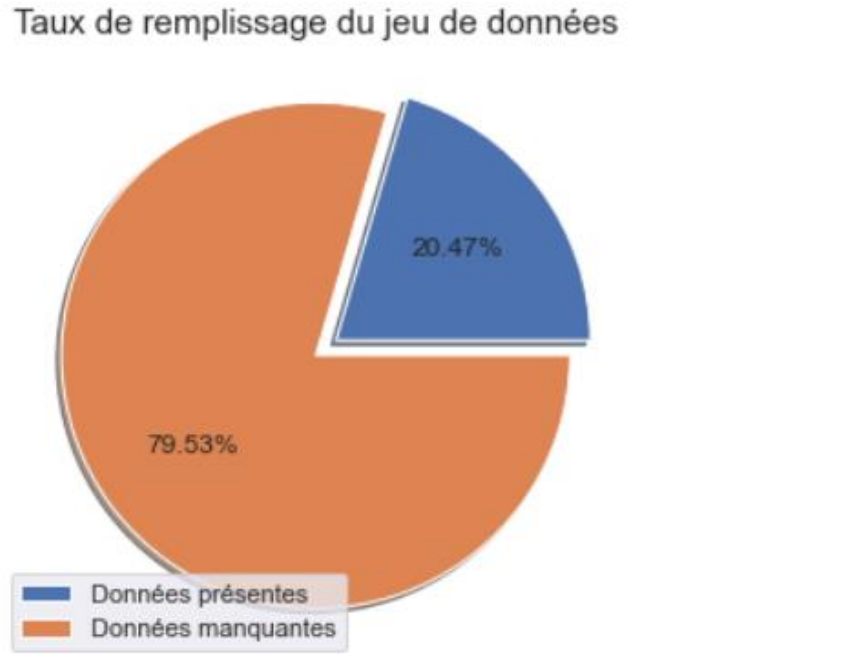
Le taux de remplissage global du jeu est de : 20.47%



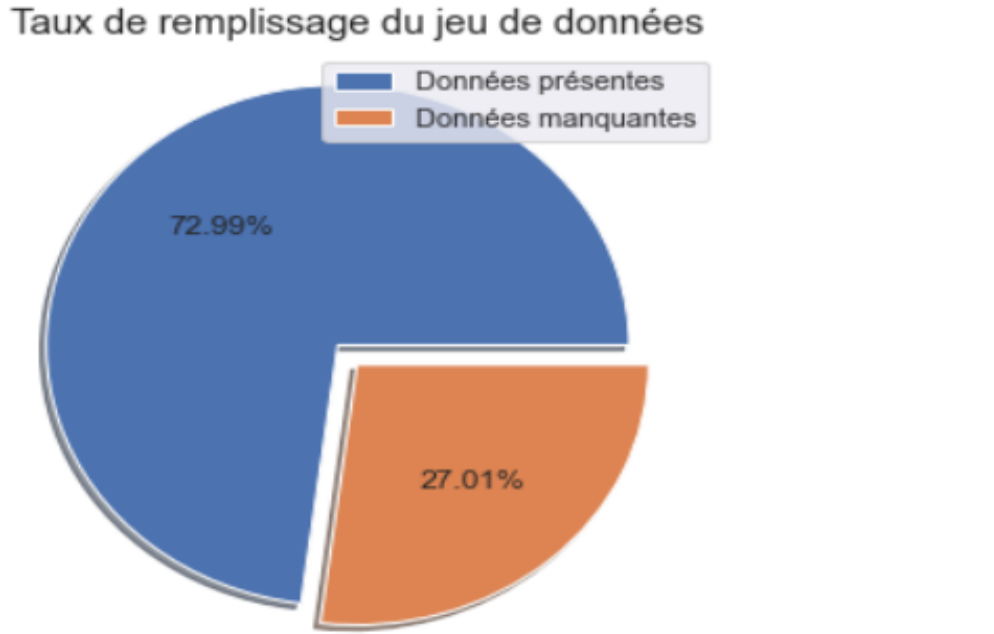
Nettoyage du jeu de données

- **Seuil de suppression des données manquantes à 90 % et filtres Pays = France et nutrition-score-fr_100g = renseigné.**

```
JEU DE DONNEES INITIAL
------------------------------------------------------
Le jeu de données a 1724219 lignes et 184 colonnes.
------------------------------------------------------
Le taux de remplissage global du jeu est de : 20.47%
------------------------------------------------------
```




```
JEU DE DONNEES RETRAITE
------------------------------------------------------
Le jeu de données a 198233 lignes et 66 colonnes.
------------------------------------------------------
Le taux de remplissage global du jeu est de : 72.99%
------------------------------------------------------
```



Nettoyage du jeu de données

- **On s'assure qu'il n'y a pas de doublons. => C'est la cas.**

Suppression des doublons

```
[53]: data.duplicated().value_counts()
```

```
[53]: False     183641
      dtype: int64
```

```
[55]: data.shape[0]
```

```
[55]: 183641
```

4. Choix des variables retenues au terme de la phase de nettoyage

Choix des variables

- **Choix des variables en lien avec notre application (valeurs nutritionnelles et nutriscore :**

energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	sodium_100g	pnns_groups_1	pnns_groups_2	nutriscore_grade	nutriscore_score

- **Validité des données produit**

on supprime les valeurs négatives et supérieures à 100 pour les teneurs d'indicateurs nutritionnels comprises entre 0 et 100 g.

```
data = data.query("~(fat_100g < 0) and ~(fat_100g > 100)")
data = data.query("~(carbohydrates_100g < 0) and ~(carbohydrates_100g  > 100)")
data = data.query("~(sugars_100g < 0) and ~(sugars_100g  > 100)")
data = data.query("~(fiber_100g < 0) and ~(fiber_100g  > 100)")
data = data.query("~(proteins_100g < 0) and ~(proteins_100g  > 100)")
data = data.query("~(salt_100g < 0) and ~(salt_100g  > 100)")
data = data.query("~(sodium_100g < 0) and ~(sodium_100g  > 100)")

data.shape
```

```
(183634, 66)
```

On supprime également d'éventuelles densités d'énergie aux valeurs négatives.

```
masque = data['energy_100g'] < 0.
indexDrop2 = data[masque].index
data = data.drop(indexDrop2)
print('Nb de produits aux densités d\'énergie négatives : ' + str(len(data[masque])))
```

```
Nb de produits aux densités d'énergie négatives : 0
```

Choix des variables

- **Vérification que les nutriscore_grade sont bien compris entre 'a' et 'e' et que les nutriscpre_score sont bien compris entre -15 et +40**

```
data['nutriscore_score'].value_counts(ascending=False)
```

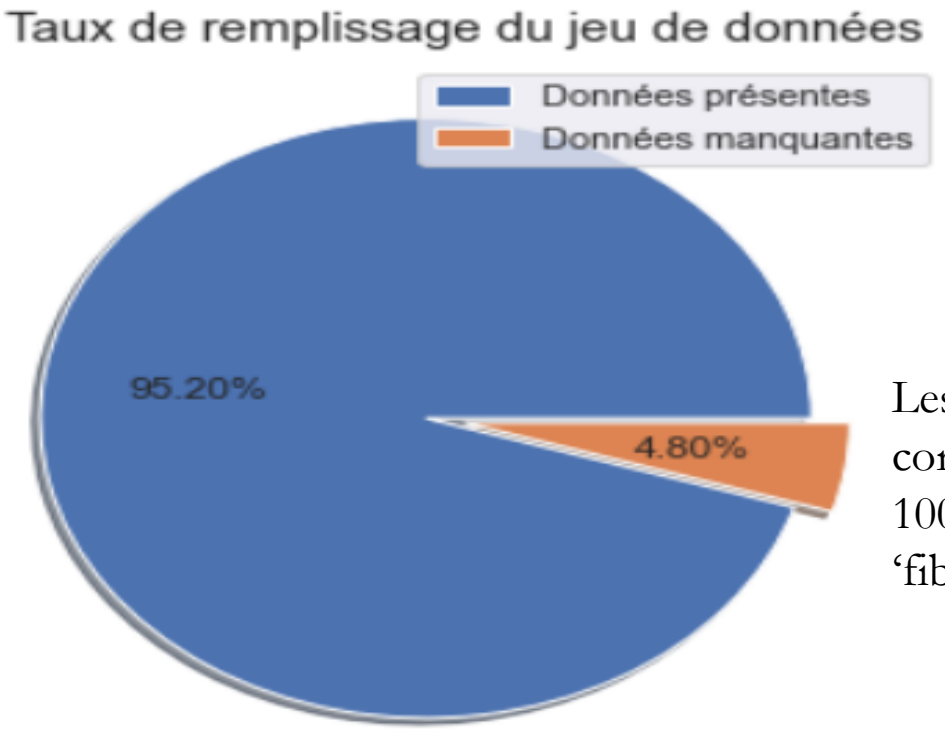
```
 14.0    9405
 3.0     8807
 11.0    8684
 0.0     8448
 13.0    8384
 2.0     8185
 12.0    7559
 4.0     7523
 15.0    7324
 1.0     7303
 5.0     6538
-1.0     6484
 16.0    6300
```

```
data['nutriscore_grade'].value_counts()
```

```
d    56701
c    40155
e    37180
a    26506
b    23092
Name: nutriscore_grade, dtype: int64
```

Jeu de données avant analyse exploratoire

```
JEU DE DONNEES FINAL AVANT ANALYSE EXPLORATOIRE
-----------------------------------------------------
Le jeu de données a 198233 lignes et 13 colonnes.
-----------------------------------------------------
Le taux de remplissage global du jeu est de : 95.20%
-----------------------------------------------------
```



Les données manquantes correspondent à prés de 100 % à la variable 'fiber_100g'

5. Analyse exploratoire univariée

Analyse exploratoire univariée

- **Remplacer les valeurs manquantes subsistant avec l'algorithme k Nearest Neighbors (k-NN) qui peut servir autant pour la régression que la classification**

```
df_knn.head()
```

	0	1	2	3	4	5	6	7	8	9
0	936.0	8.2	2.2	29.0	22.0	0.0	5.1	4.60	1.840	18.0
1	134.0	0.3	0.1	5.3	3.9	0.9	0.9	0.42	0.168	1.0
2	1594.0	22.0	15.5	27.3	21.9	4.4	4.6	0.10	0.040	14.0
3	657.0	0.0	0.0	36.0	27.0	3.6	0.6	0.00	0.000	-2.0
4	598.0	12.7	1.0	3.9	1.0	0.0	1.9	0.27	0.108	1.0

```
print(df_knn.apply(nombre_manquantes, axis=0))
```

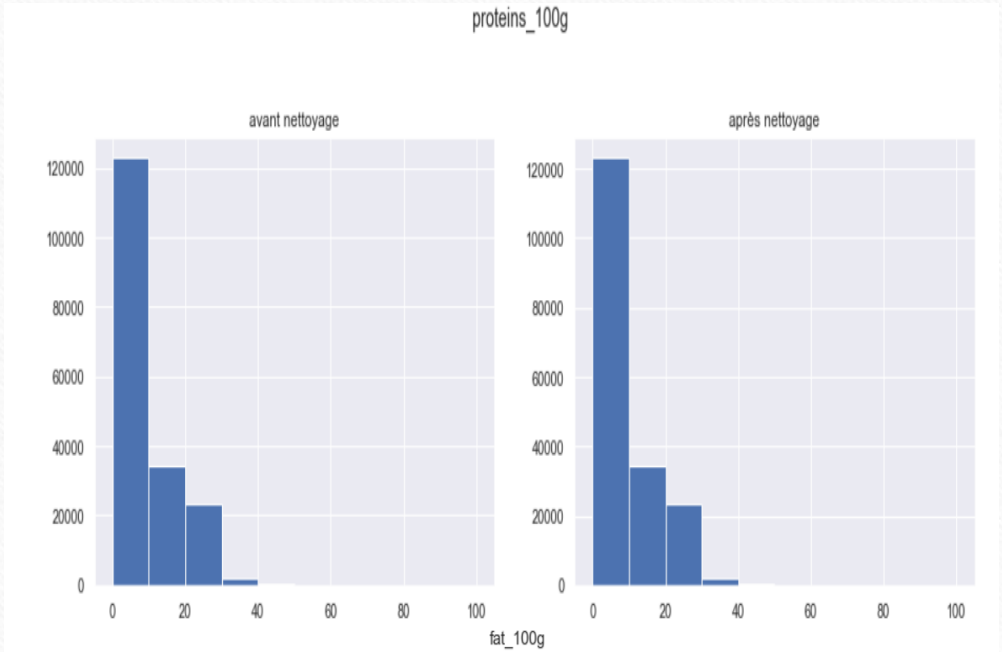
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
dtype: int64
```

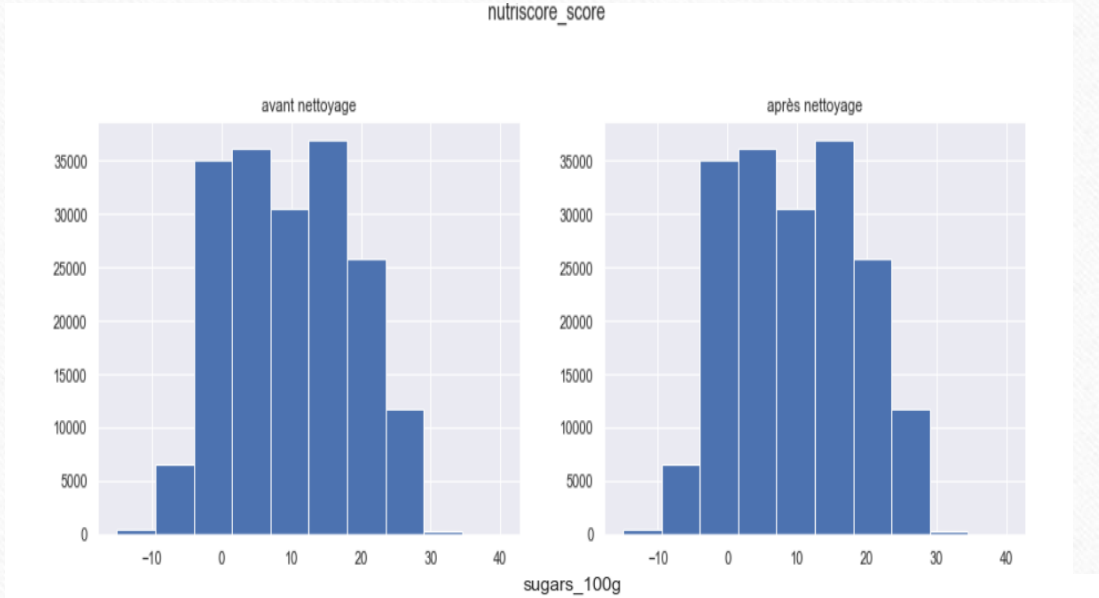
Analyse exploratoire univariée

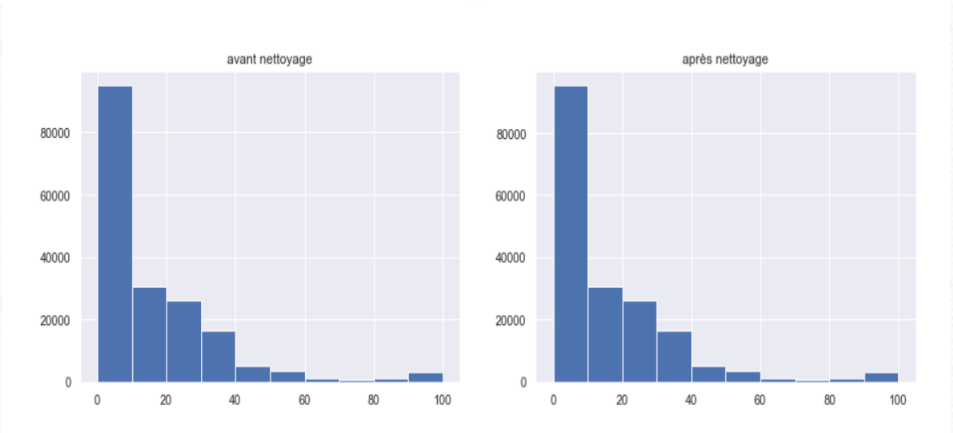
- Compte tenu du faible taux de valeurs manquantes pour l'ensemble des variables (hormis le 'fiber_100g'), les distributions et les différents indicateurs statistiques sont quasiment identiques après nettoyage k_NN. **S'agissant de 'fiber_100g' :**

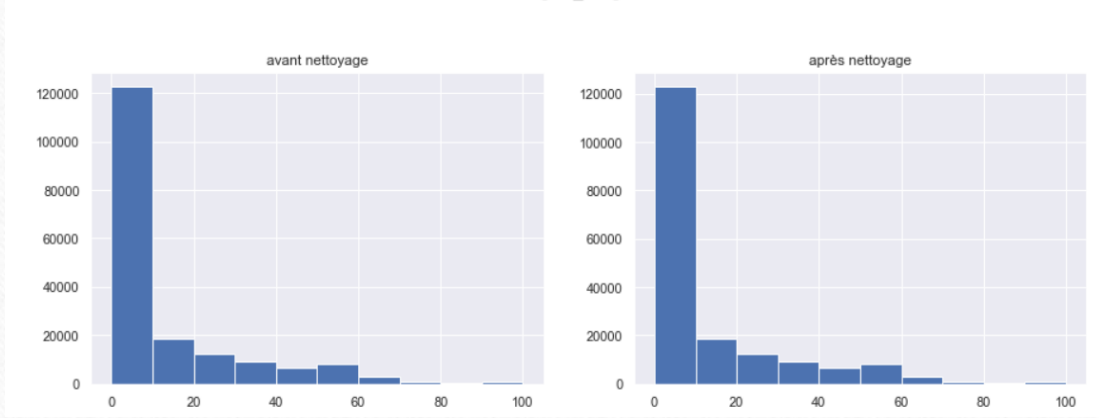
Indicateur	Avant traitement	Aprés traitement
Moyenne	2.61	1.78
Variance	15.25	9.10
Ecart-type	3.90	3.01
Skew	5.52	6.03
Kurtosis	63.93	81.15

Analyse exploratoire univariée – Comparaison des distributions avant et après nettoyage k-NN









Analyse exploratoire univariée

Synthèse de l'analyse univariée :

- Les distributions de chacune des variables ***ne semblent pas suivre une Loi normale***. Cela sera ***confirmée par des tests statistiques de normalité***. En effet, autant l'observation de la forme des distributions que les mesures de symétrie (Skew) et d'aplatissement (Kurtosis), de même que l'observation des dispersisons permises par les boxplots ou encore les diagrammes Q-Q Plot convergent vers la négation d'une Loi normale.

- On distingue 4 groupes distincts quand à l'allure de la distribution :

  - energy_100g, saturated-fat_100g, fiber_100g, salt_100g, sodium_100g.
  - fat_100g, carbohydrates_100g, sugars_100g.
  - proteins_100g.
  - nutriscore_100g.

- Les ***3 premiers groupes*** ont une ***distribution décalée vers la droite*** tandis que le dernier (***nutriscore***) présente une ***distribution bimodale.***

Analyse exploratoire univariée

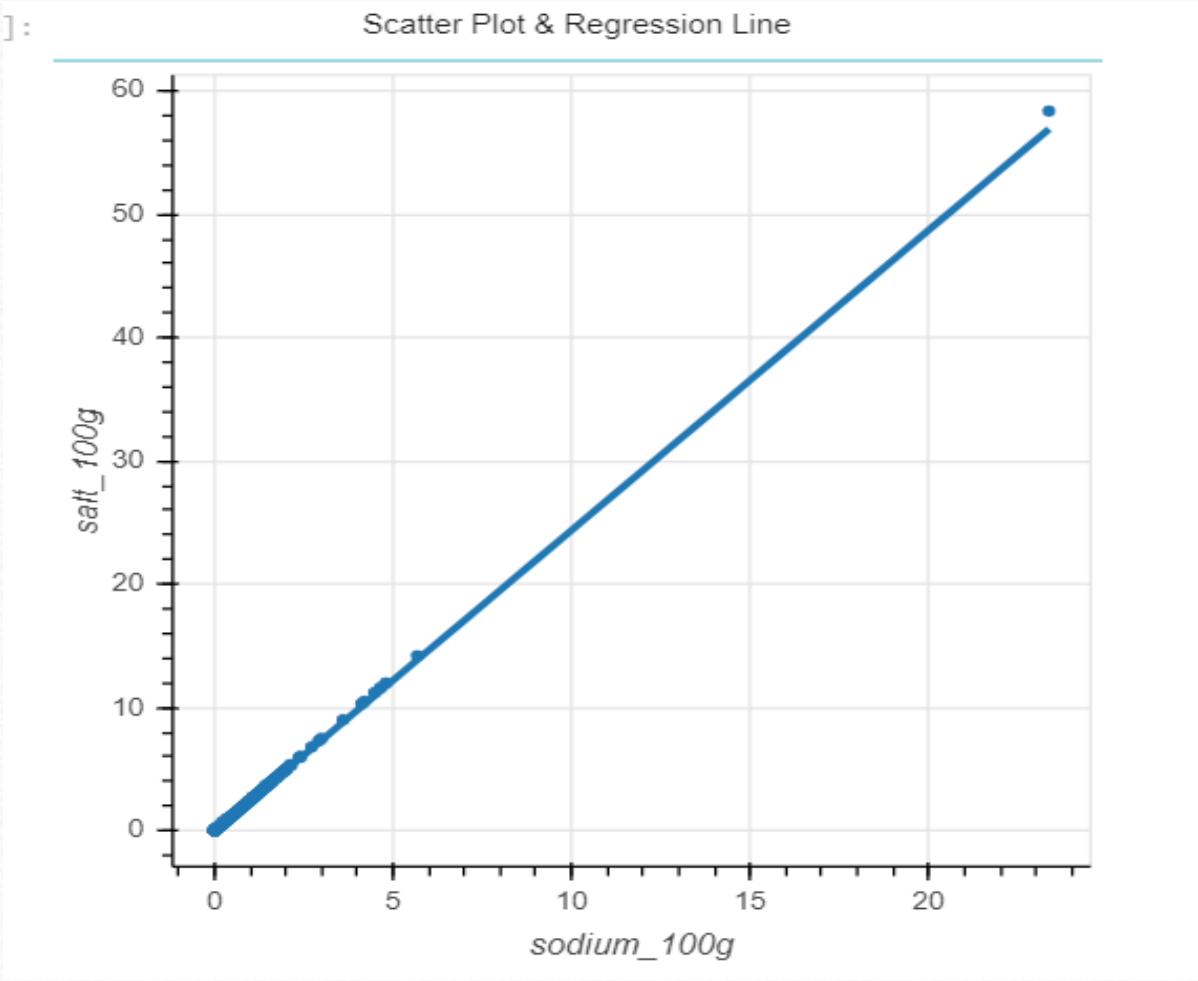
• **Les différents tests de normalité confirment que ces distributions ne suivent pas une Loi normale [**Test de Shapiro-Wilk, Test de Kolmogorov-Smirnov, Test K ^2 de D'Agostino-Pearson, Test d'Anderson-Darlin] => ***pvalue = 0 pour tous les tests.***

```
------------------------------------------------------------
TEST DE KOLMOGOROV-SMIRNOV
------------------------------------------------------------
energy_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.9915385172853459  : Probablement non gaussien
fat_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.7022136547144213  : Probablement non gaussien
saturated-fat_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.5239496595402648  : Probablement non gaussien
carbohydrates_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.7331937564385029  : Probablement non gaussien
sugars_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.5658645554745689  : Probablement non gaussien
fiber_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.5  : Probablement non gaussien
proteins_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.73254028268092  : Probablement non gaussien
salt_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.5  : Probablement non gaussien
sodium_100g <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.5  : Probablement non gaussien
nutriscore_score <function kstest at 0x00001B2B9650DC0>
pvalue = 0.0 , statistic 0.7472271053825983  : Probablement non gaussien
------------------------------------------------------------
```

6. Analyse exploratoire bivariée

Analyse exploratoire bivariée

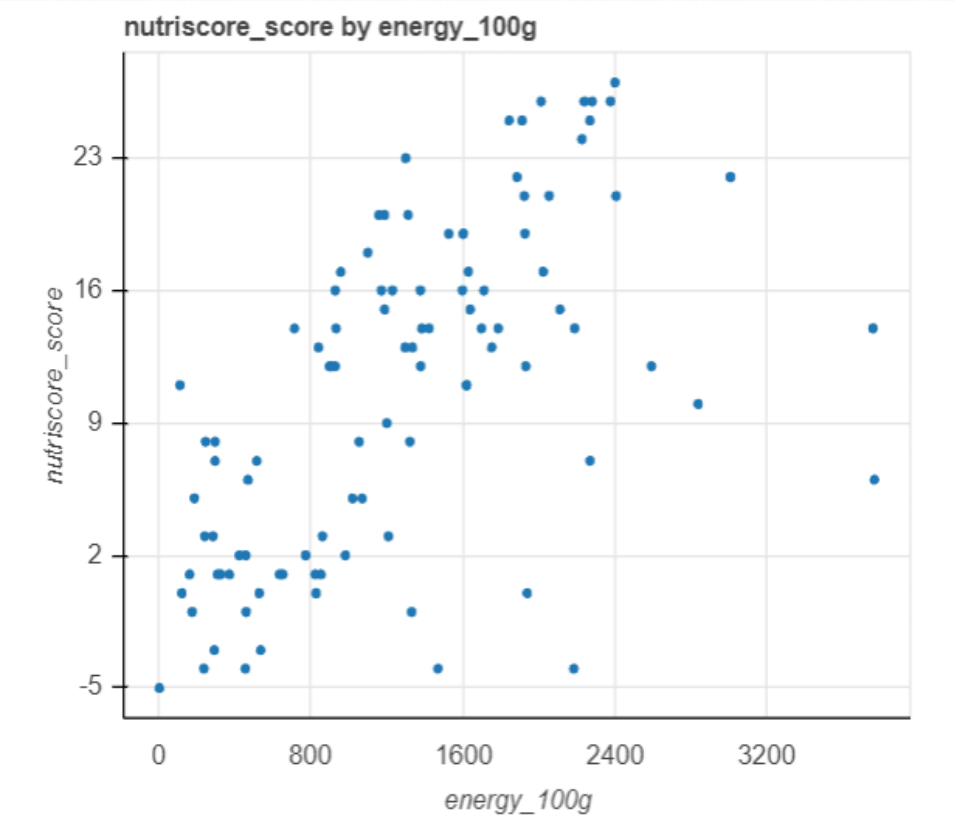
]:

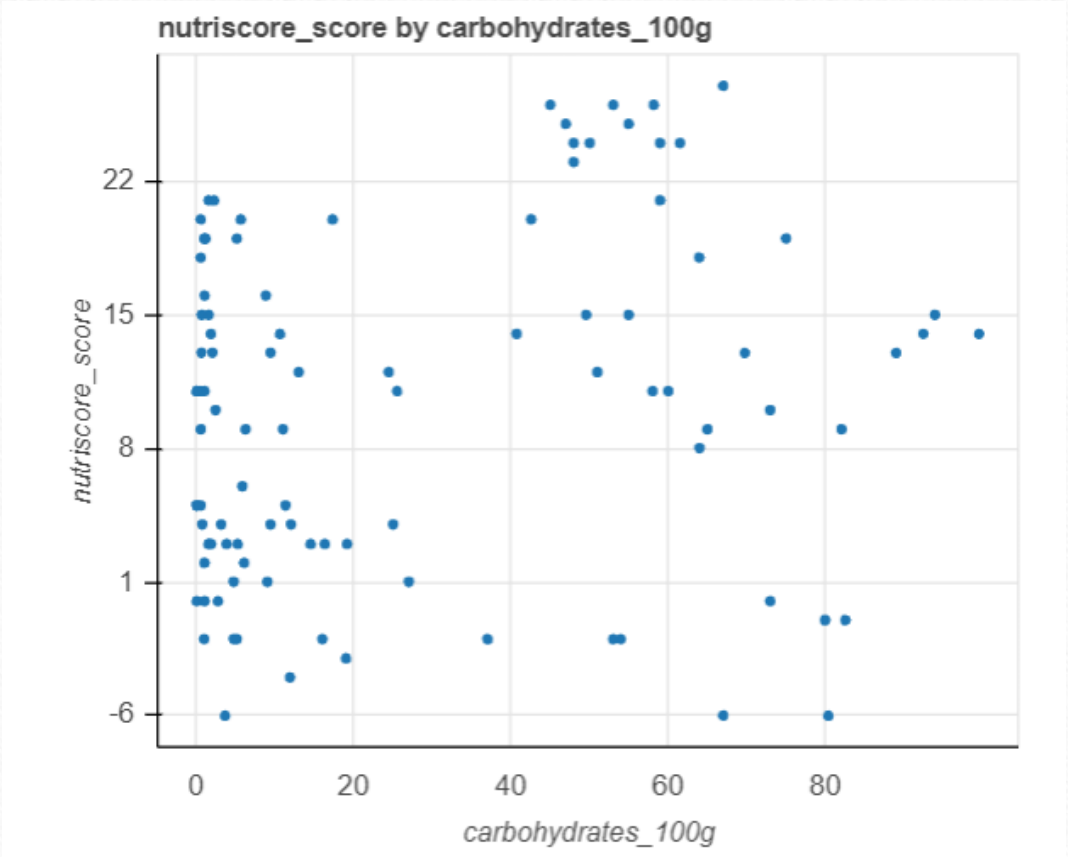


Comme l'on pouvait le présumer, 'salt' et 'sodium' sont parfaitement corrélés. On retirera donc la variable 'sodium' de notre jeu de données dans la suite de notre analyse.

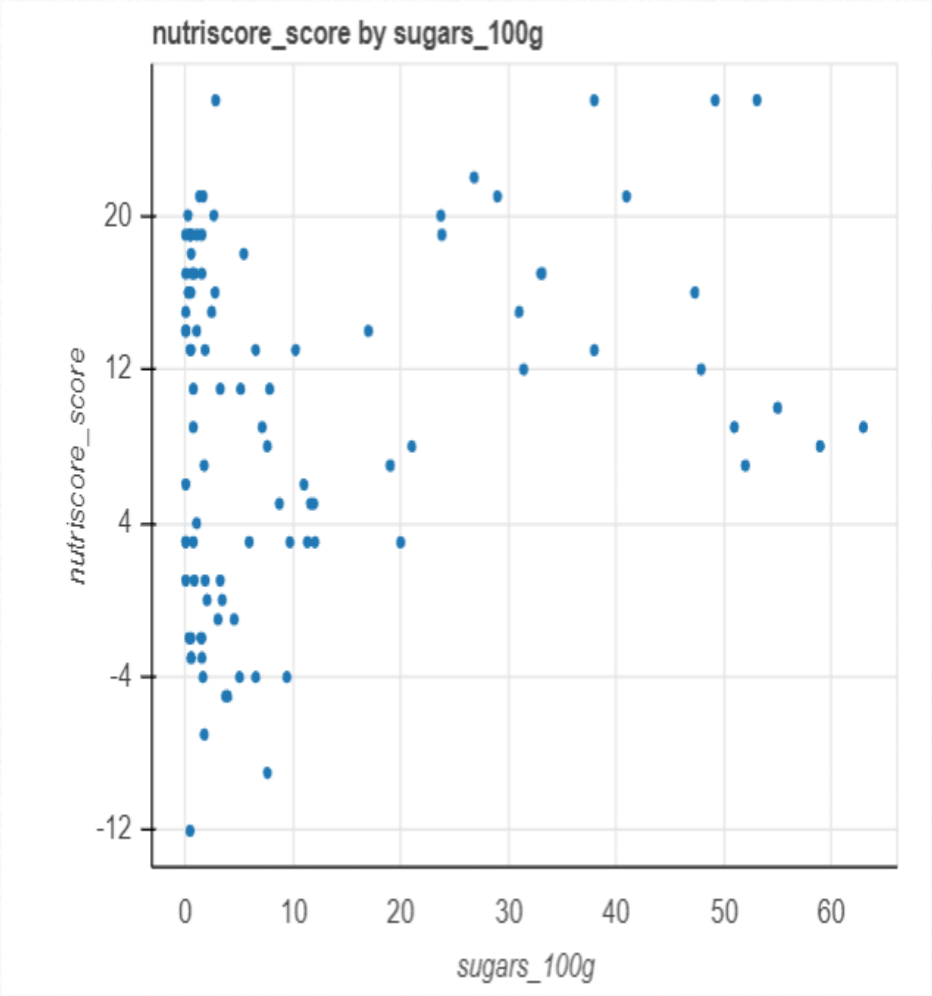
Analyse exploratoire bivariée

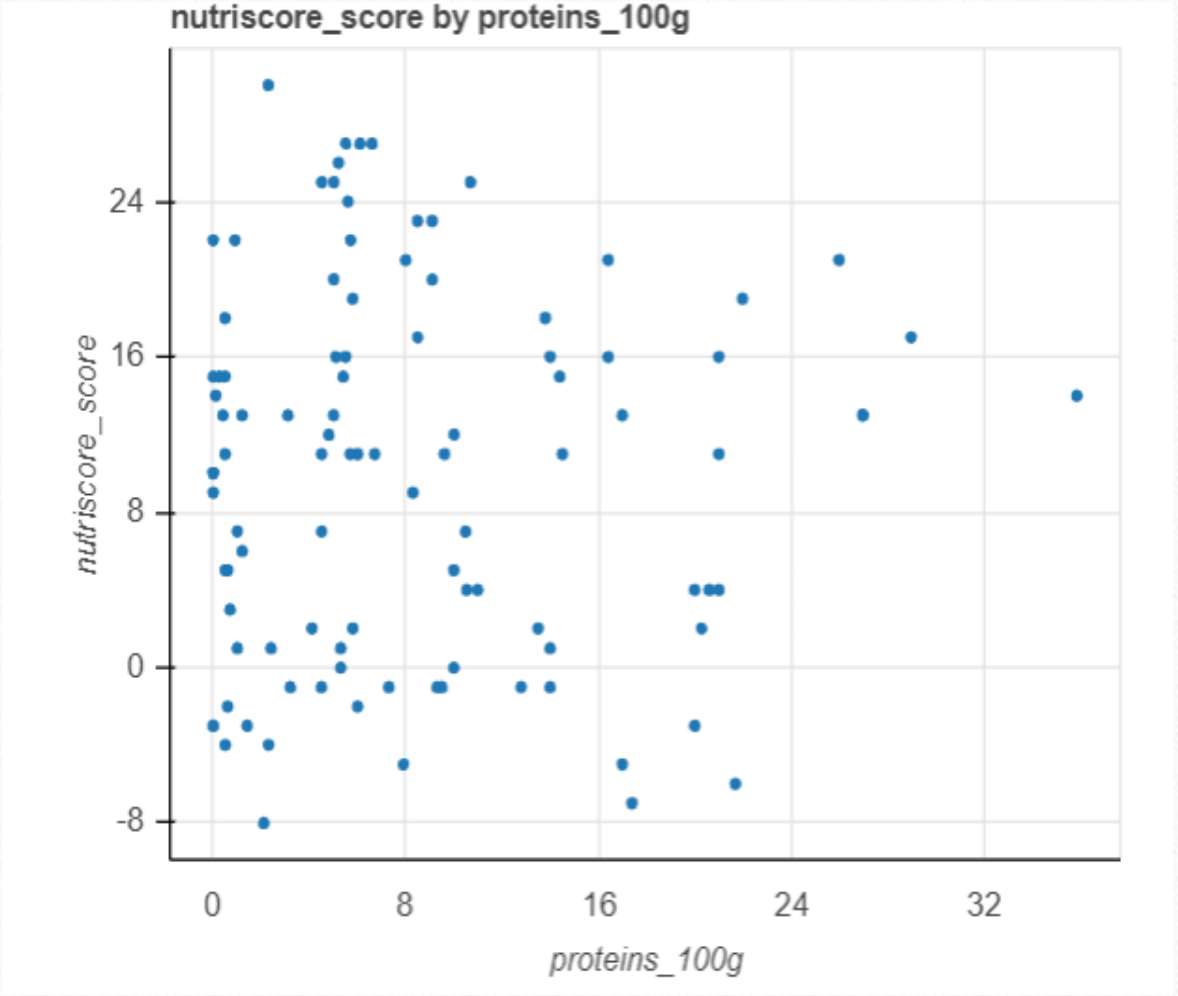
- Notre variable cible étant 'nutriscore_score', nous menons des ***analyses bivariées entre la cible et chacune des variables*** constituant les éléments nutritionnels de base.





Analyse exploratoire bivariée





Analyse exploratoire bivariée

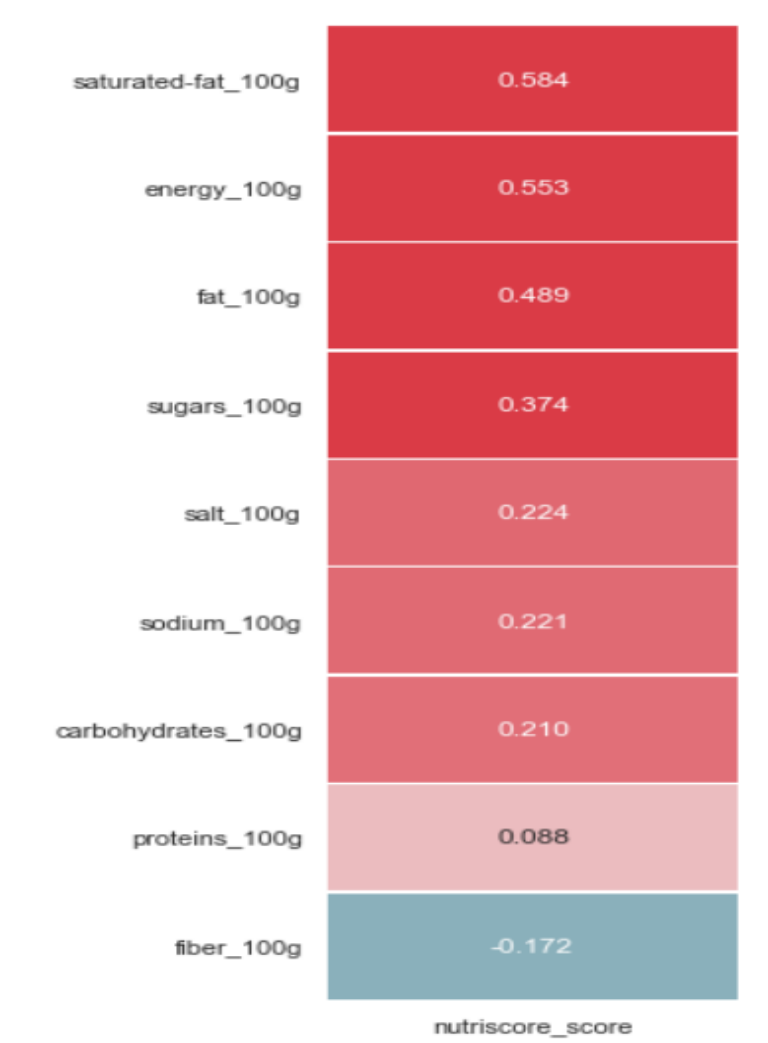
- L'observation des Scatter Plot ne permet pas de tirer des conclusions très tranchées quand à la nature des relations entre la variable cible (nutriscore_score) et chacune des variables explicatives.

- Voyons si la matrice de corrélation de Pearson peut nous aider à y voir plus clair,

	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g	salt_100g	sodium_100g	nutriscore_score
energy_100g	1.0	0.73	0.52	0.45	0.27	0.23	0.22	0.0018	0.00076	0.49
fat_100g	0.73	1.0	0.69	-0.081	-0.0074	0.061	0.13	0.052	0.049	0.48
saturated-fat_100g	0.52	0.69	1.0	-0.02	0.1	0.0021	0.11	0.019	0.018	0.57
carbohydrates_100g	0.45	-0.081	-0.02	1.0	0.6	0.26	-0.076	-0.14	-0.13	0.23
sugars_100g	0.27	-0.0074	0.1	0.6	1.0	0.054	-0.26	-0.18	-0.18	0.45
fiber_100g	0.23	0.061	0.0021	0.26	0.054	1.0	0.18	-0.046	-0.045	-0.17
proteins_100g	0.22	0.13	0.11	-0.076	-0.26	0.18	1.0	0.23	0.22	0.042
salt_100g	0.0018	0.052	0.019	-0.14	-0.18	-0.046	0.23	1.0	0.95	0.2
sodium_100g	0.00076	0.049	0.018	-0.13	-0.18	-0.045	0.22	0.95	1.0	0.19
nutriscore_score	0.49	0.48	0.57	0.23	0.45	-0.17	0.042	0.2	0.19	1.0

Nous écarterons 'fiber_100g' pour la suite de notre étude car cette variable n'est pas corrélée avec notre cible ajoutée au fait qu'elle présente plus de 61 % de valeurs manquantes avant traitement k-NN.

Analyse exploratoire bivariée – Vecteur de corrélation



En analysant ce vecteur de corrélation, nous pouvons constater que le score de nutrition est positivement corrélé à :

- saturated_fat_100g
- energy_100g
- fat_100g
- sugars_100g

Corrélé dans une moindre mesure à :

- salt_100g (sodium_100g)

Et pas corrélé avec :

- proteins_100g
- fiber_100g

Cela est cohérent avec le fait qu'un aliment est d'autant meilleur pour la santé que son score de nutrition est bas.

Analyse exploratoire bivariée – Tests d'indépendance entre la cible et les variables explicatives

- **Tests menés** :
  - Chi2 d'indépendance,
  - Test selon corrélation de Pearson,
  - Test selon corrélation de rang de Spearman,
  - Test selon corrélation de rang de Kendall,

- Les différents tests effectués nous donnent une assurance raisonnable que **les couples de variables** ('nutriscore_score', variable explicative) **sont dépendants.**

Analyse exploratoire bivariée – Tests d'indépendance entre la cible et les variables explicatives

```
from scipy.stats import pearsonr
for col in data.select_dtypes('float64'):
    stat, p = pearsonr(data['nutriscore_score'], data[col])
    print('stat=%.3f, p=%.3f' % (stat, p))
    if p > 0.05:
        print('nutriscore_score / ', col, ' : Variables probablement indépendantes')
    else:
        print('nutriscore_score / ', col, ' : Variables probablement dépendantes')
```

```
stat=0.553, p=0.000
nutriscore_score /  energy_100g  : Variables probablement dépendantes
stat=0.489, p=0.000
nutriscore_score /  fat_100g  : Variables probablement dépendantes
stat=0.584, p=0.000
nutriscore_score /  saturated-fat_100g  : Variables probablement dépendantes
stat=0.210, p=0.000
nutriscore_score /  carbohydrates_100g  : Variables probablement dépendantes
stat=0.374, p=0.000
nutriscore_score /  sugars_100g  : Variables probablement dépendantes
stat=-0.087, p=0.000
nutriscore_score /  fiber_100g  : Variables probablement dépendantes
stat=0.088, p=0.000
nutriscore_score /  proteins_100g  : Variables probablement dépendantes
stat=0.224, p=0.000
nutriscore_score /  salt_100g  : Variables probablement dépendantes
stat=0.221, p=0.000
nutriscore_score /  sodium_100g  : Variables probablement dépendantes
stat=1.000, p=0.000
nutriscore_score /  nutriscore_score  : Variables probablement dépendantes
```

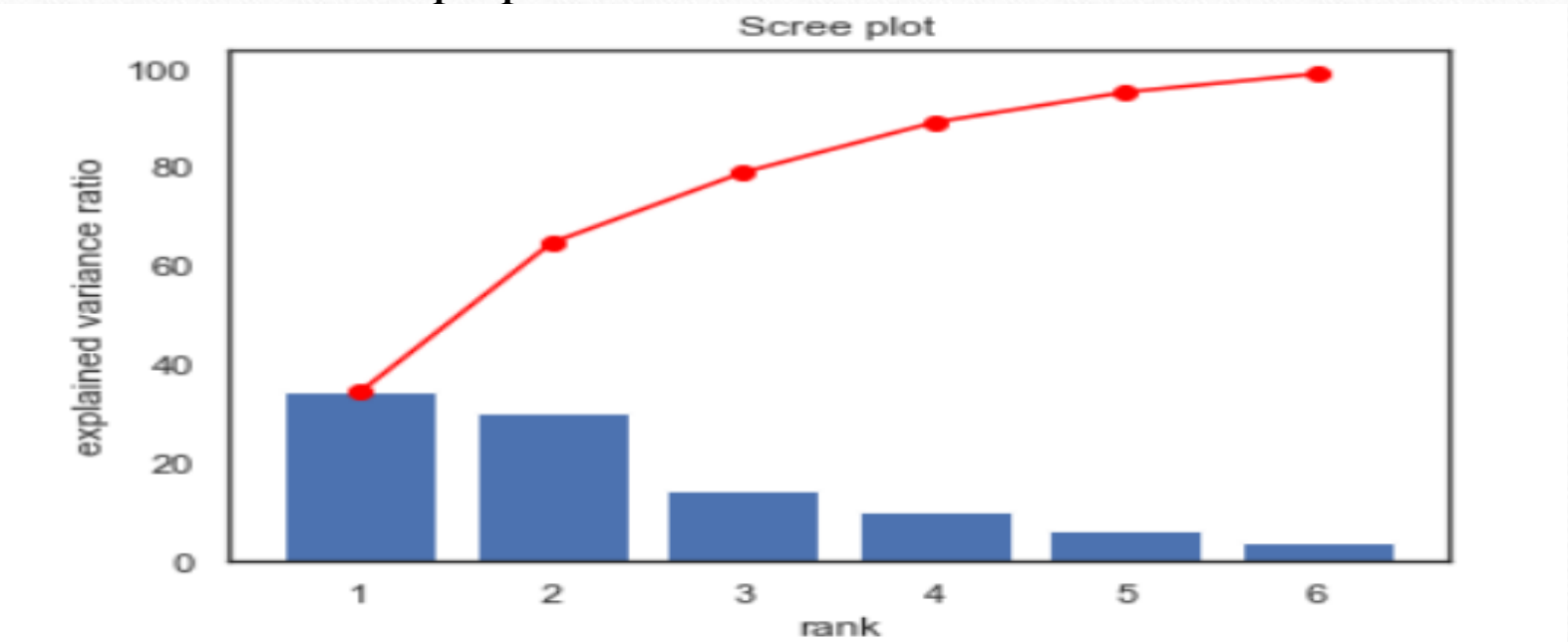
7. Analyse exploratoire multivariée - ACP

Analyse exploratoire multivariée - ACP

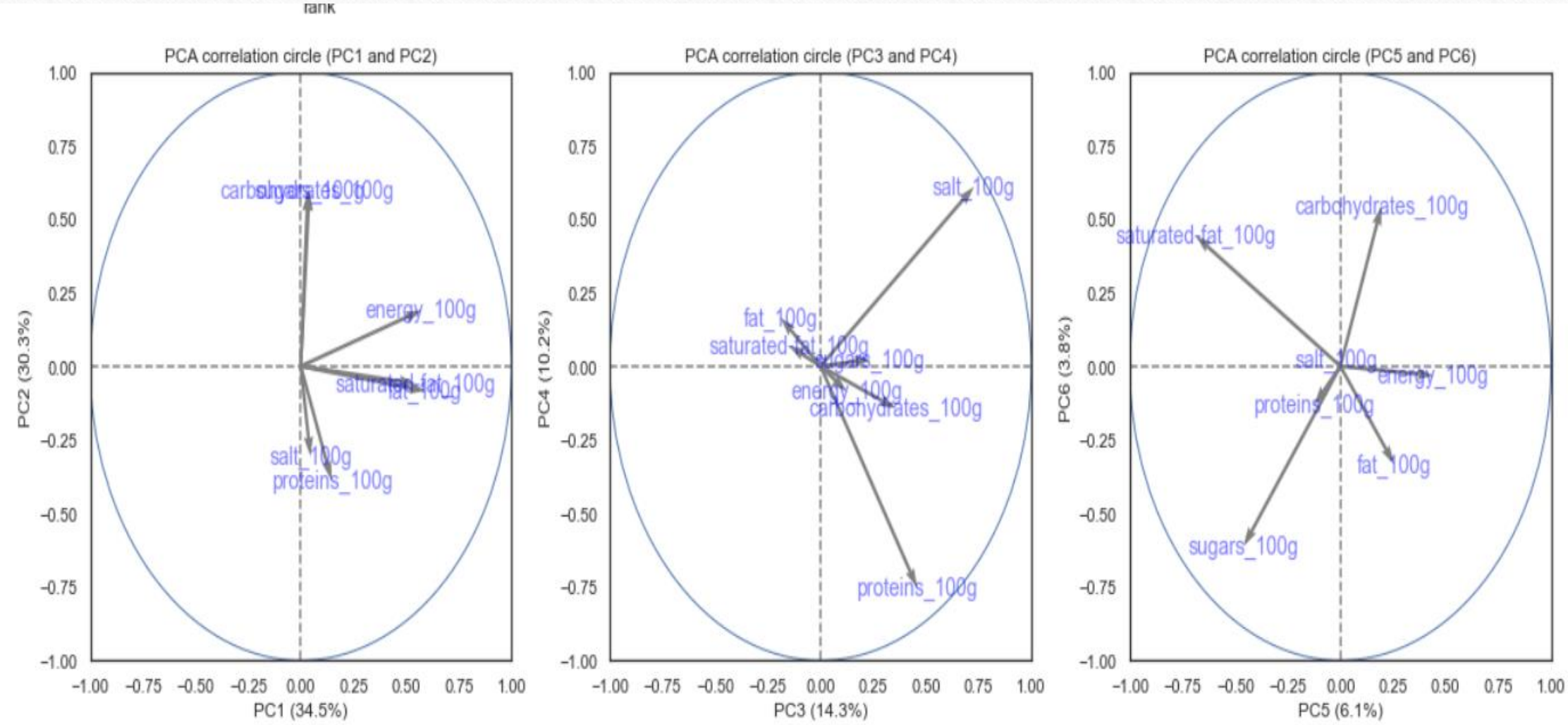
- **Sélection de 7 variables quantitatives pour l'ACP**

	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	proteins_100g	salt_100g
0	936.0	8.2	2.2	29.0	22.0	5.1	4.60
1	134.0	0.3	0.1	5.3	3.9	0.9	0.42

- **Projection des variables initiales sur 3 plans factoriels correspondant à 6 composantes principales**
- **Eboulis des valeurs propres**



Analyse exploratoire multivariée - ACP



Analyse exploratoire multivariée - ACP

Nous observons que :

- energy_100g et saturated_10g sont correctement positivement corrélés à l'axe PC1.
- carbohydrates_100g est positivement corrélé à l'axe PC2.
- salt_100g et proteins_100g sont faiblement corrélés négativement à l'axe PC2 mais correctement corrélés positivement sur l'axe PC3.

Choix du nombre de composantes :

- Valeurs propres associées aux axes factoriels :

```
print(pca.explained_variance_)
```

```
[2.41229873 2.11882704 0.99808758 0.71382644 0.42961821 0.26274947]
```

- Selon la règle de Kaiser, **un axe est intéressant si sa valeur propre est supérieur à 1.**
- Sur l'éboulis des valeurs propres, on observe un **décrochement sur l'axe 3.**
- L'application de ces 2 critères nous conduit à **retenir les axes PC1 et PC2 qui expliquent à eux deux prés de 65 % de l'inertie totale. Ajoutons** néanmoins **les 2 axes suivants** pour expliquer un **taux d'inertie plus important à 89.3 %.** Ce faisant, nous réduisons nos dimensions de 7 variables initiales à 4 composantes principales.

8. Prédiction du nutriscore à partir des composantes principales

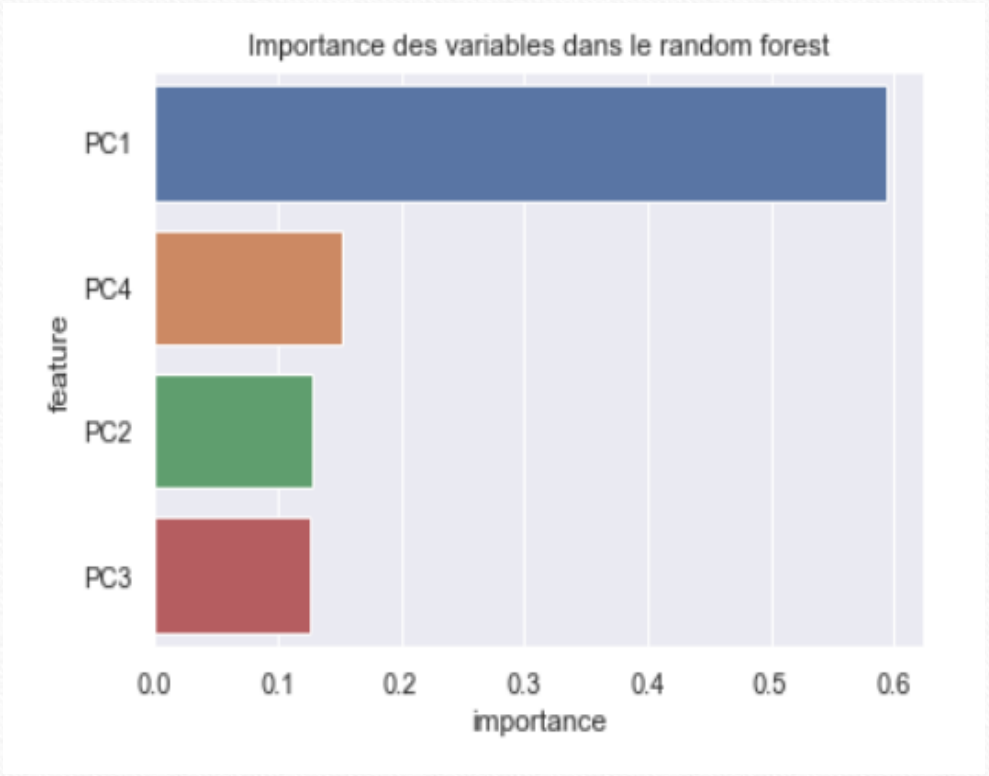
Prédiction du nutriscore à partir des composantes principales

- La bibliothèque Pycaret nous aide à mieux cibler les modèles d'apprentissage les plus adaptés à notre jeu de données. Dans notre application, **les modèles Extra Trees Regressor et Random Forest Regressor semblent les plus appropriés.** Extra Trees et Random Forest sont en fait deux méthodes d'ensemble très similaires. Les méthodes d'ensemble d' arbres sont meilleures que les simples arbres de décision.

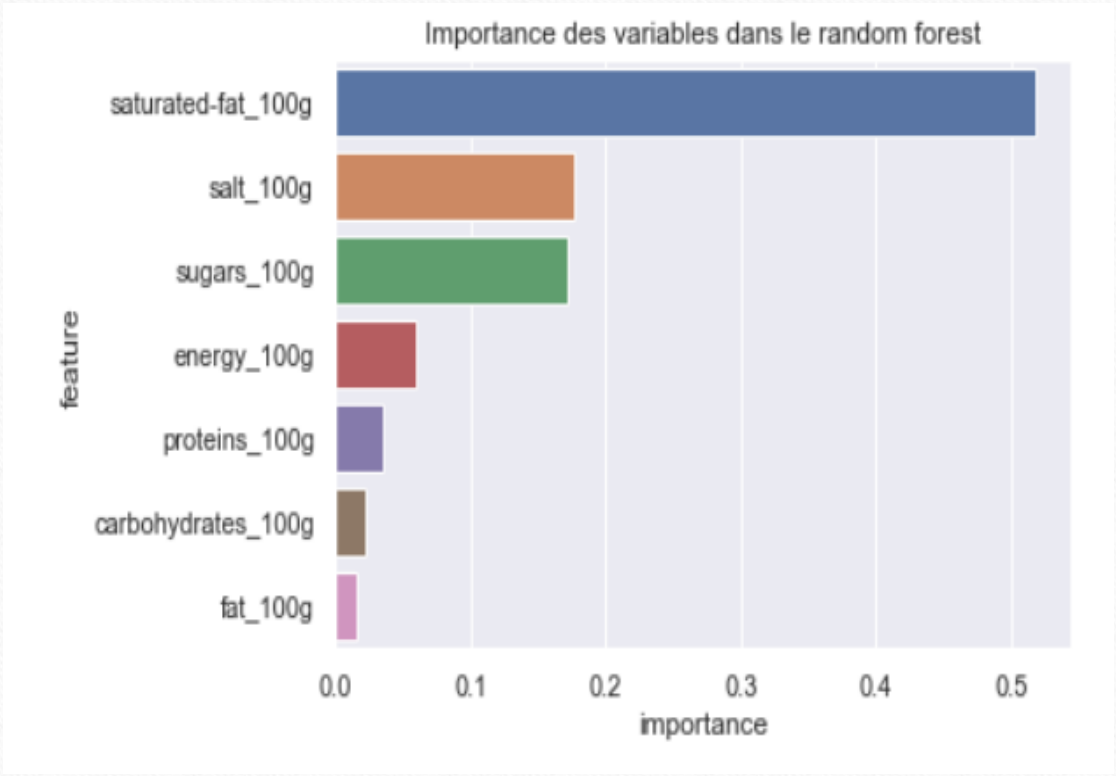
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	1.6677	8.2155	2.8661	0.8954	0.4175	0.3681	3.5770
rf	Random Forest Regressor	1.7373	8.3419	2.8881	0.8937	0.4207	0.3811	9.3480
knn	K Neighbors Regressor	1.8410	9.2198	3.0362	0.8826	0.4366	0.4038	0.1220
catboost	CatBoost Regressor	2.0720	9.4989	3.0818	0.8790	0.4518	0.4419	7.9760
lr	Linear Regression	4.8279	38.8824	6.2338	0.5046	0.6962	0.9976	0.4920
lar	Least Angle Regression	4.8279	38.8824	6.2338	0.5046	0.6962	0.9976	0.0240
br	Bayesian Ridge	4.8280	38.8824	6.2338	0.5046	0.6961	0.9976	0.0250
ridge	Ridge Regression	4.8279	38.8824	6.2338	0.5046	0.6962	0.9976	0.0240
lasso	Lasso Regression	5.3017	42.0288	6.4824	0.4645	0.7045	1.1457	0.0240

Prédiction du nutriscore à partir des composantes principales

Composantes principales ACP



Variables initiales



Prédiction du nutriscore à partir des composantes principales

2 indicateurs pour porter un jugement sur la préférence entre Random Forest et Extra Trees :

- Le coefficient de détermination R2. Plus R2 sera proche de 1, meilleure sera la prédiction. => $R^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$

- Le RMSE (Erreur quadratique moyenne). Cet indice fournit une indication par rapport à la dispersion ou la variabilité de la qualité de la prédiction. => $RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$

Indicateur	Random Forest	Extra Trees
RMSE	2.82	2.81
R2 entraînement	0.98	0.99
R2 test	0.89	0.90

- Ces deux modèles sont effectivement extrêmement proches avec une ***très légère préférence pour Extra Trees Regressor.***

Prédiction du nutriscore à partir des composantes principales

- **Comparaison effectuée sur le modele Random Forest Regressor entre une prédiction fondée sur les 4 premières composantes principales et les 7 variables initiales :**

Indicateur	Composantes principales	Variables initiales
RMSE	2.82	2.34
R2 entraînement	0.98	0.98
R2 test	0.89	0.93

- Perte légère de performance avec les composantes principales sur le R2 en test.

9. Faits pertinents pour l'application

Faits pertinents pour l'application

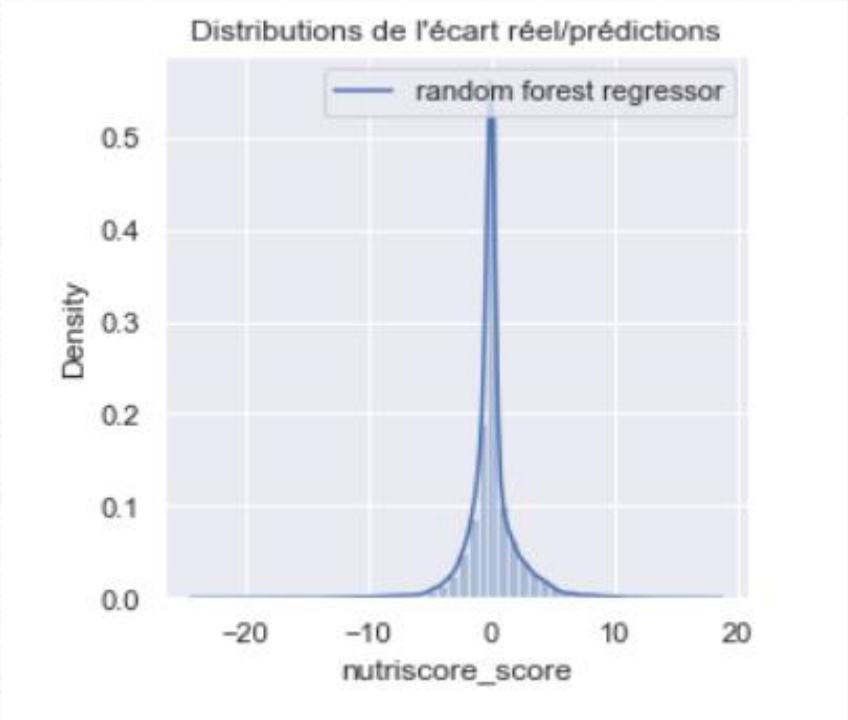
4 observations :

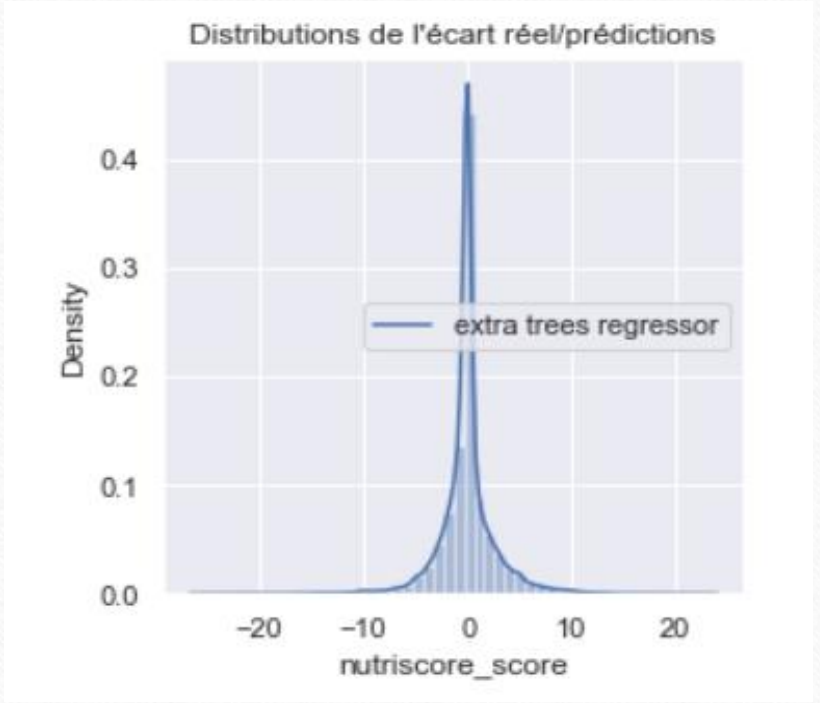
- Dépendances des variables.
- Bonne corrélation de certaines variables avec le nutriscore_score.
- Réduction des dimensions (de 7 variables initiales à 4 composantes principales) avec peu de pertes de performances.
- Résultats des régressions.

Faits pertinents pour l'application

Application d'un algorithme d'ensemble (Random Forest et/ou Extra Trees) :

- R^2 = 0,90 à 0,93 sur le jeu de données de test.
- RMSE = 2,34 à 2,82





10. Synthèse

Synthèse

- Modèles Random Forest ou Extra Trees concluants.
- Résultats cohérents avec les principes nutritionnels (graisses saturées, sucres, …).
- Possibilité de proposer un algorithme qui donne une indication de nutriscore approché ($R^2 = 0{,}90$) avec 7 variables réduites à 4 composantes principales.
- **=> Faisabilité de l'application**

MERCI

Avez-vous des questions ?