Projet de conception d'une application au service de la santé publique

Cyril REGAN

Base de données :
https://static.openfoodfacts.org/data/en.openfoodfacts.org.products.csv

10 mars 2020

Table of Contents

Problématique

Santé publique France

Appel à projets pour trouver des idées innovantes d'applications en lien avec l'alimentation.

Construire une fonction qui prédit le nutriscore en fonction de caractéristiques partielles

Présentation du jeu de données

Données brutes : Tableau de 1113279 lignes $\times$ 178 colonnes (2.3 Go).

le nutriscore est calculé suivant 10 variables :

Augmente le nutriscore :	Valeur énergétique	Acides gras saturés	Sucres	Sodium
Baisse le nutriscore :		Fibres	Fruits Légumes / Fruits à coque	Protéines

Table of Contents

Nettoyage des données

Les variables retenues sont :

product_name	nutriscore_score	energy_100g
saturated_fat_100g	sugars_100g	fiber_100g
proteins_100g	sodium_100g	~~Fruits~~(+ de 99% NaN)

X = NaN pour X $\in$ [saturated_fat_100g, sugars_100g, fiber_100g, proteins_100g, sodium_100g] tel que :

$X > \mathbf{100}$ **ou** $X < \mathbf{0}$

et energy_100g = NaN pour le 1^{er} et le dernier centile.

Entrainement Validation et Test

<table>
<tr><td>Nettoyage et outliers</td><td></td><td colspan="2">DATA
Data with NaN | Data complete</td></tr>
<tr><td>Analyse Statistique</td><td></td><td colspan="2">Data complete</td></tr>
<tr><td rowspan="2">Construction modèles =></td><td rowspan="2">Imputation non nutri
Prediction nutri</td><td>$NaN_{\notin nutri}$</td><td>Train + validation</td></tr>
<tr><td colspan="2">Train + validation</td></tr>
<tr><td>Test final =></td><td></td><td colspan="2">Test</td></tr>
</table>

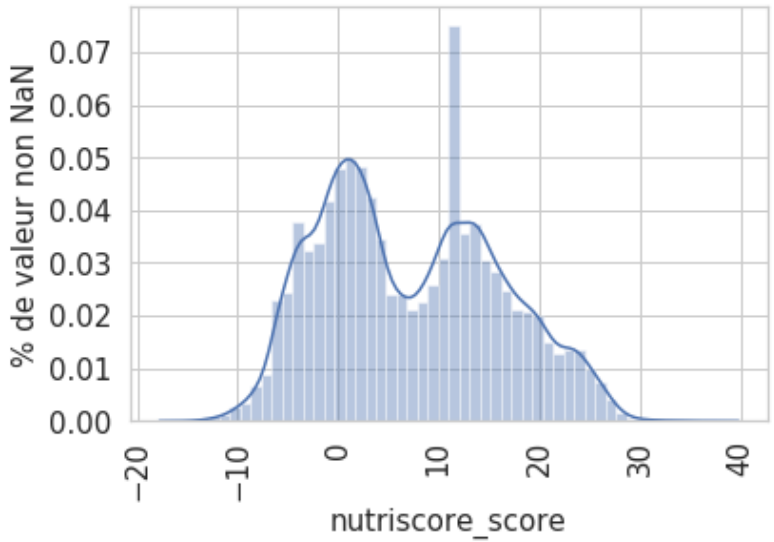
Table of Contents

Tests de Kolmogorov : normalité

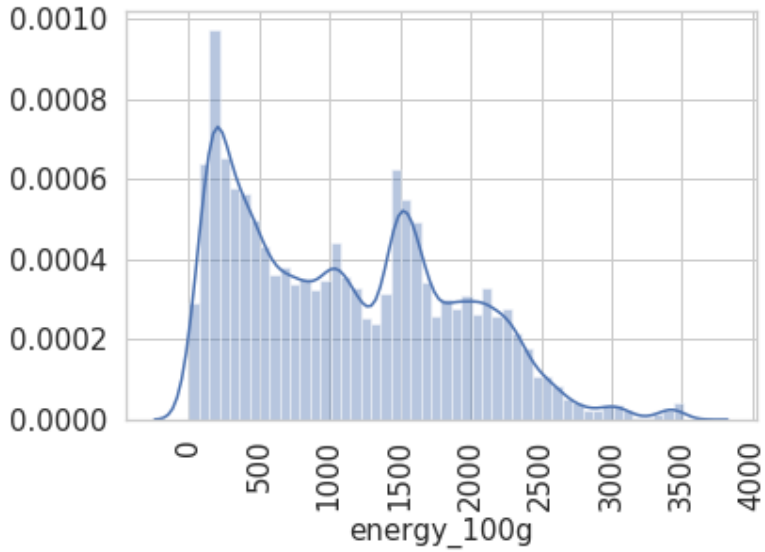
p-valeur : *plus petite valeur du niveau de test α conduisant au rejet de l'hypothèse nulle H0* :

$$\text{Rejet de H0 au niveau de test } \alpha \iff \text{p-valeur} < \alpha$$

=> **H0** : la distribution est normale



nutriscore_100g
p-valeur = 0 : H0 est rejetée.



energy_100g
p-valeur = 0 : H0 est rejetée.

Classification

Classification des variables qui augmentent le nutriscore selon

energy_100g	saturated_fat_100g	sugars_100g	sodium_100g
Classe 0			
1005	3	13.5	0.27
Classe 3			
2345	7	31	0.63
Classe 7			

Classification des variables qui diminuent le nutriscore selon

anses
French agency for food, environmental and occupational health & safety
Investigate, evaluate, protect

fiber_100g	proteins_100g
Classe 0	
1.9	3.0
Classe 2	
3.7	6.4
Classe 4	

Test de Khi2 : indépendance des classes proteins_100g et energy_100g

=> **H0** : les deux variables proteins_100g et energy_100g sont indépendantes.

proteins_100g energy_100g	0	2	4	Total
0	19398	16134	15039	50571
3	3922	11968	31677	47567
7	1492	619	4267	6378
Total	24812	28721	50983	104516

Tableau de contingence

La p-valeur = 0.0 : l'hypothèse nulle est rejetée

Test d'analyse de la variance (ANOVA)

=> **H0** (multifactoriel) : les distributions du **nutri_score** suivent la même loi normale suivant les modalités des facteurs **sugars_100g** et **fiber_100g** ou leur combinaison

Ensembles des groupes ci dessous :

sugars_100g	**fiber_100g**	N > **10**	pval Kolmogorov
0	0	26580	p = 0
	2	7328	p = 6.63272e-91
	4	4488	p = 4.82879e-85
3	0	15530	p = 0
	2	9731	p = 0
	4	14961	p = 0
7	0	14480	p = 0
	2	5016	p = 6.63272e-91
	4	6402	p = 4.82879e-85

Résultats de l'ANOVA :

	sum_sq	df	F	**PR(>F)**
C(sugars_100g)	3.525e+06	2.0	42159.100	**0.0**
C(fiber_100g)	5.348e+05	2.0	6395.308	**0.0**
C(sugars_100g) :C(fiber_100g)	8.343e+04	4.0	498.867	**0.0**
Residual	4.369e+06	104507.0	NaN	NaN

Sous l'hypothèse de normalité des groupes (non satisfaite) :

L'hypothèse nulle rejetée : le nutri_score n'a pas la même loi normale suivant les modalités des facteurs sugars_100g ou fiber_100g ou leur combinaison.

Test de Student : Comparaison de 2 moyennes

=> **HO** : les moyennes du nutri_score sont égales dans les deux groupes('0') et ('3') de energy_100g

Hypothèse pour le test de Student : Distribution normale des 2 groupes.

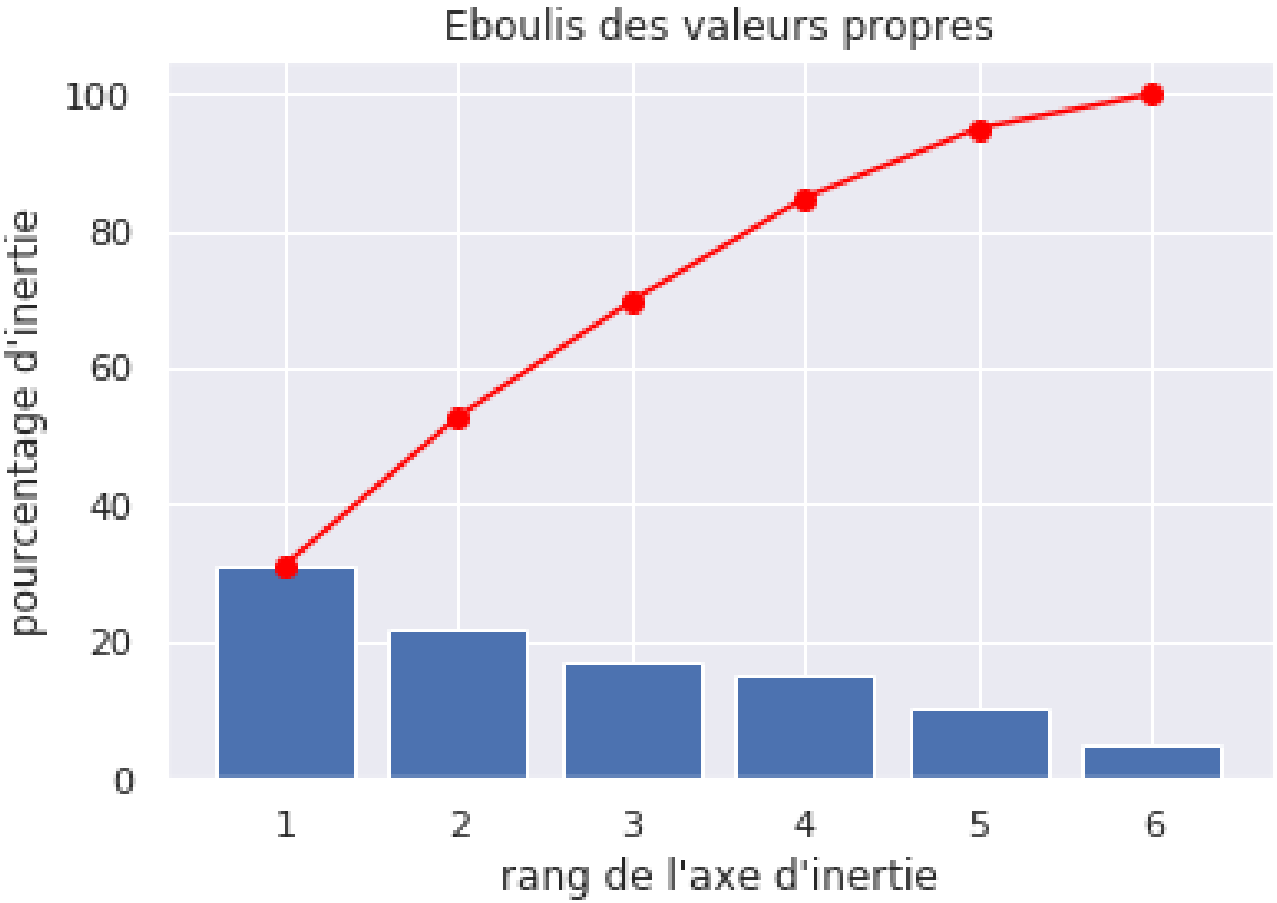
('0')	p = 0	HO Kolmogorov est rejetée
('3')	p = 0	HO Kolmogorov est rejetée

Test de Kolmogorov

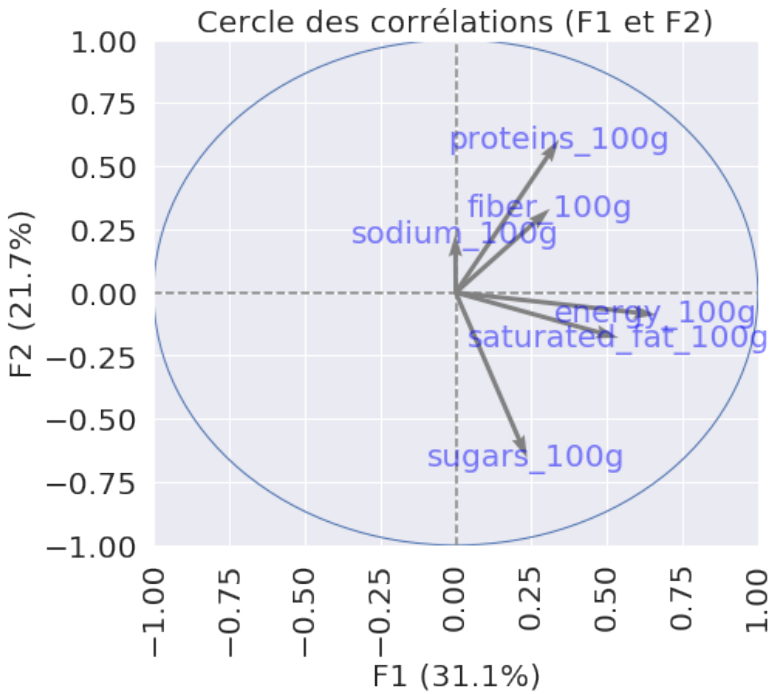
Sous l'hypothèse de normalité des groupes (non satisfaite) :

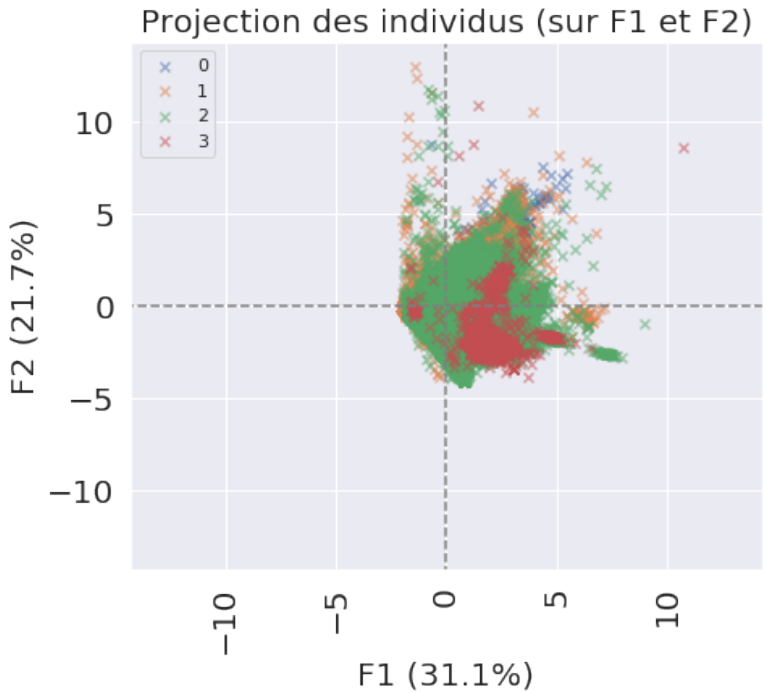
TTest	p = 0	H0 Student est rejetée.

Analyse par Composantes Principales (ACP) : Eboulis des valeurs propres

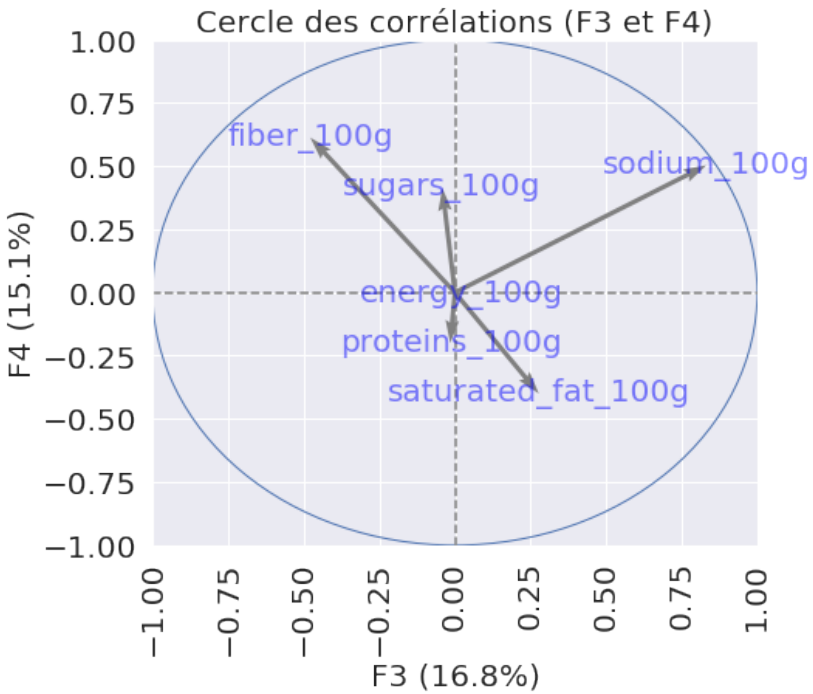


Cercles des corrélations et projection des individus sur (F1, F2)





Cercles des corrélations et projection des individus sur (F3, F4)



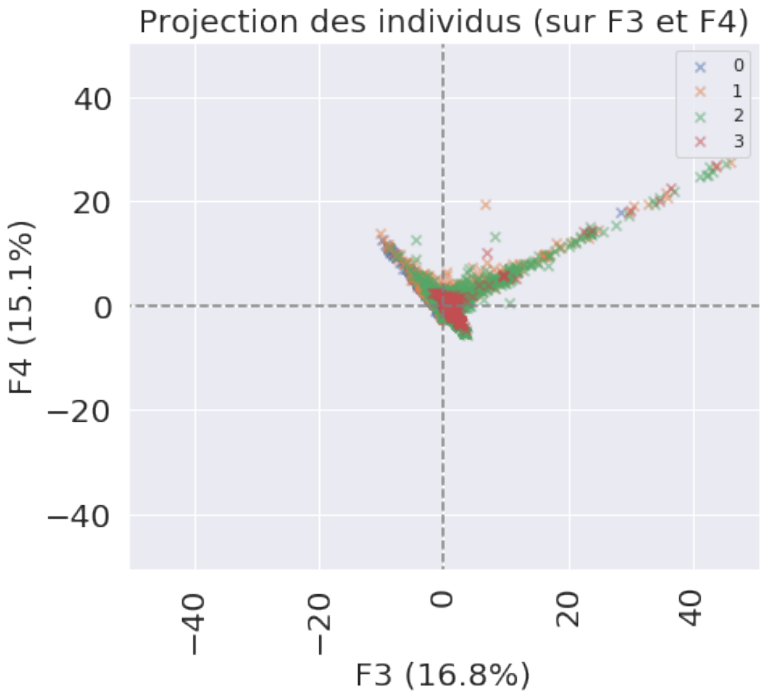
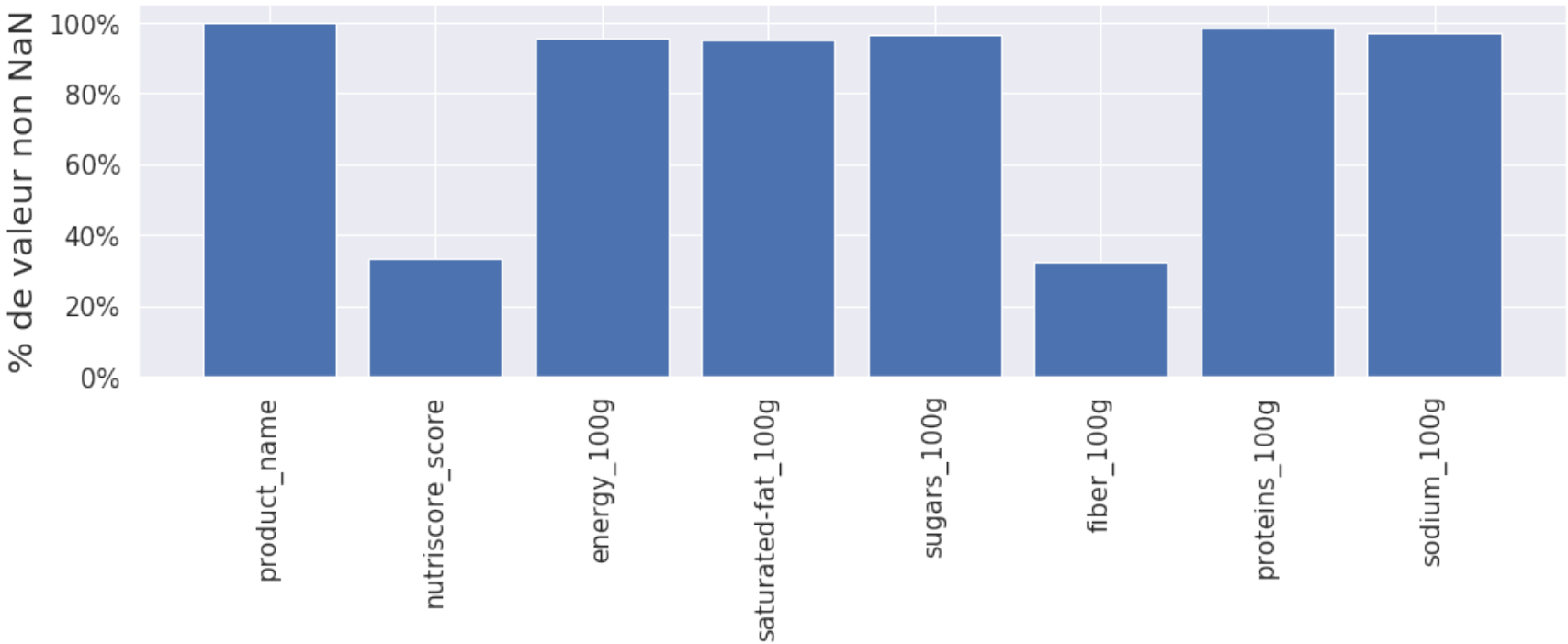


Table of Contents

Imputation des variables : Normalisation et réduction des variables [energy_100g : sodium_100g] par une moyenne = 0 et une variance = 1.



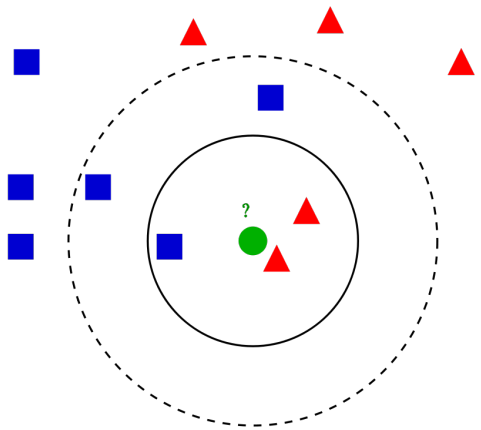
Schema =>

Combinaisons de knn

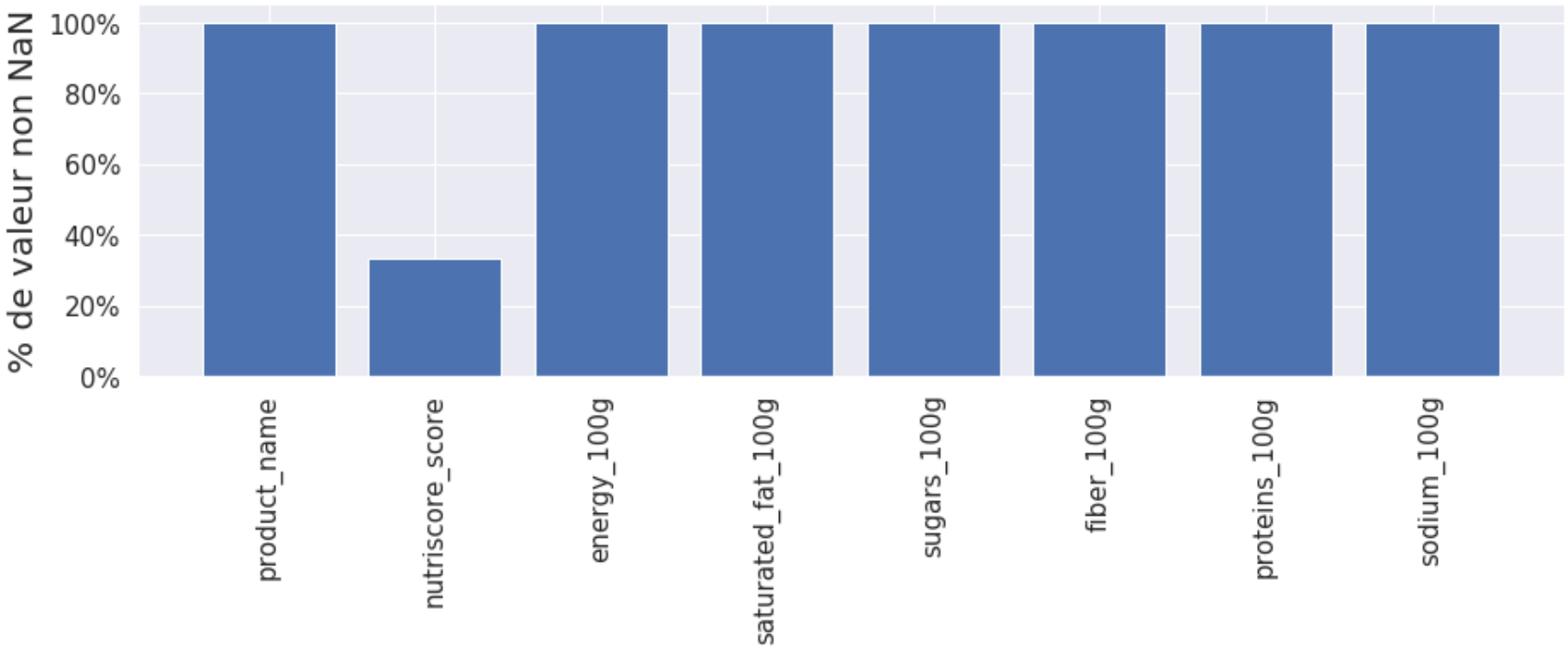
- L'imputation automatique par KNNImputer de sklearn impossible (Pb memoire).

$\Rightarrow$ Choix d'algorithme des k plus proches voisins (k-nearest neighbors algorithm : knn) successifs.

Pour chaque variable : imputation par knn (avec optimisation de lhyperparamètre k) en fonction des données completes des combinaisons des autres variables.



Valeurs présentes après imputation :



Construction du modèle nutriscore : détection des valeurs abhérantes multivariées :

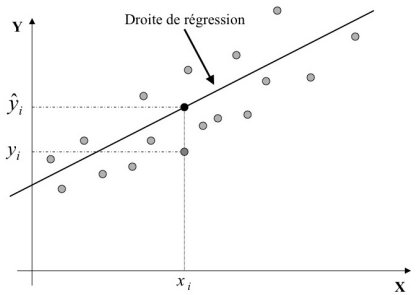
Si la masse de :

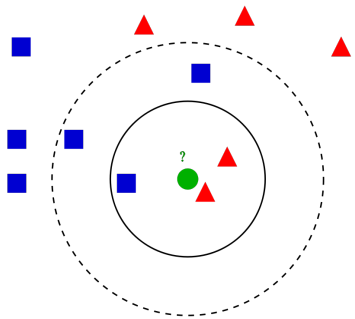
$$\sum (\text{saturated\_fat\_100g} + \text{sugars\_100g} + \text{fiber\_100g} + \text{proteins\_100g} + \text{sodium\_100g}) > \mathbf{100 \text{ ou } < 0}$$

$=>$ on supprime l'individu.

Identification des modèles d'imputation du nutriscore :

Modèle d'apprentissage **supervisé** de **regression**





Sans ACP : [energy_100g : sodium_100g]

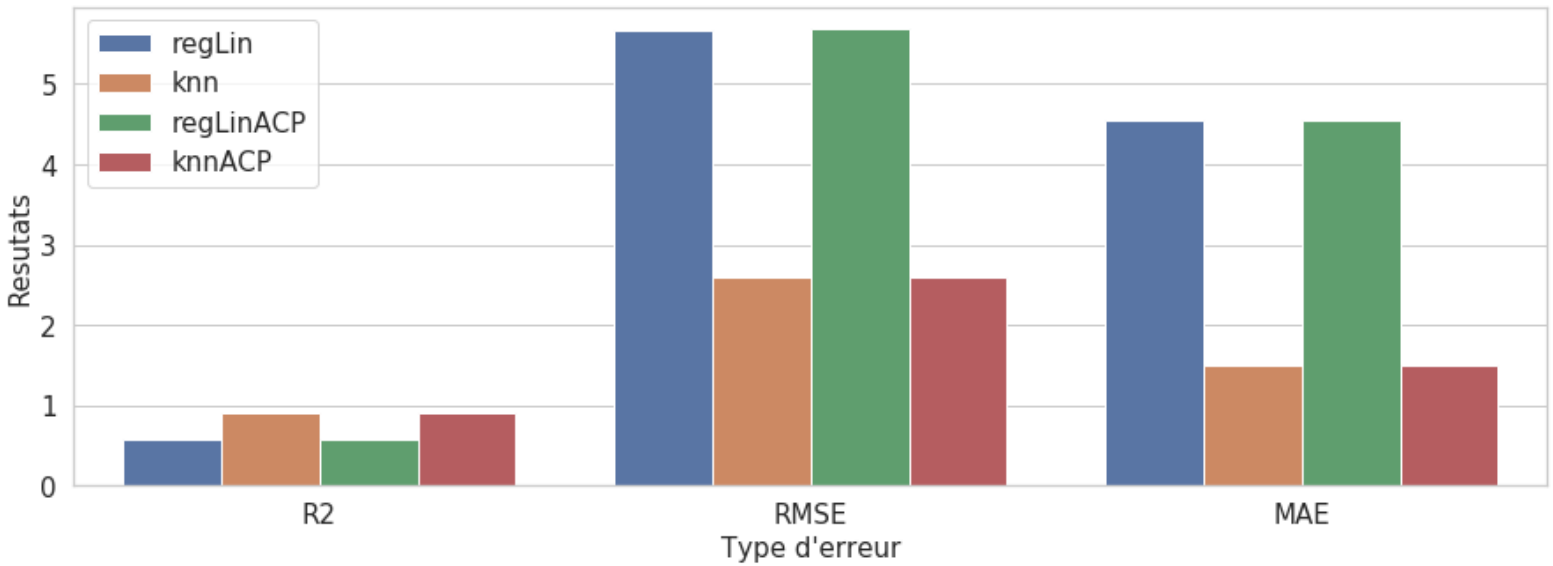
ou

avec ACP : [F1 :F6]

Coéficient de détermination : R2 = $1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

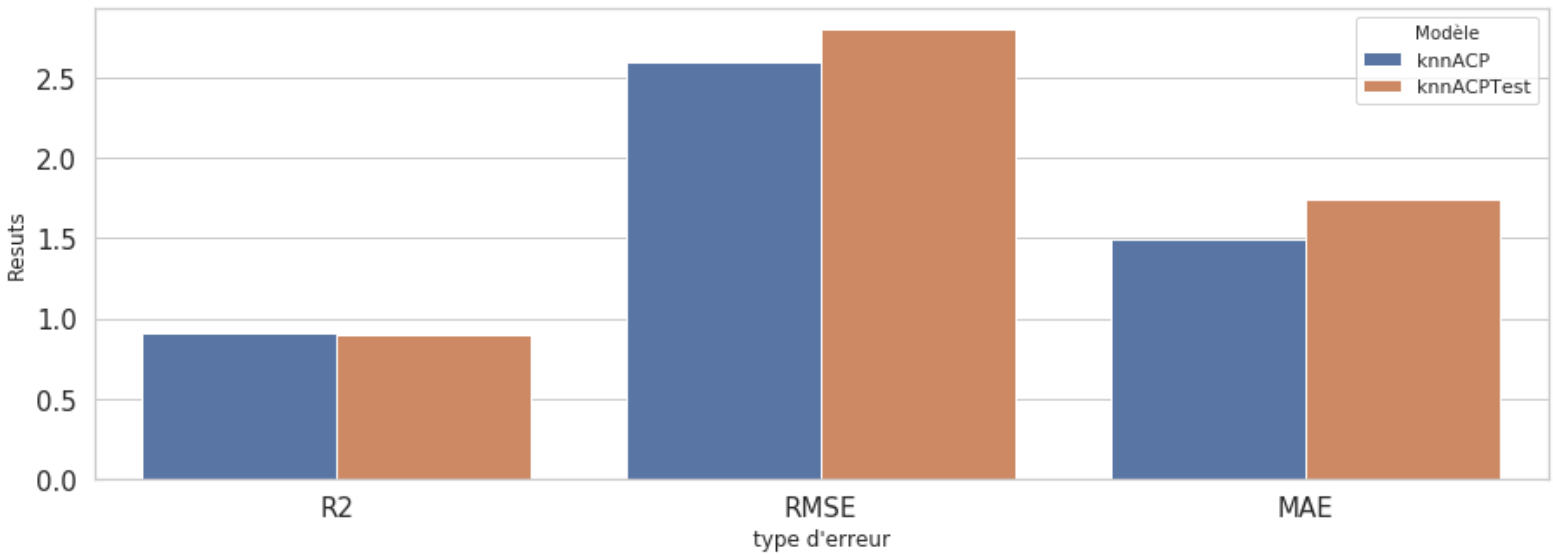
Root Mean Square Error RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

Mean Absolute Error : MAE = $\frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|$.



Modèle supervisé de regression retenu : **knn avec ACP**

Evaluation du modèle retenu avec l'échantillon test Schéma => :



Construction de la fonction nutriscore :

Utilisation :

- Entrée : [energy_100g :sodium_100g]
- Sortie : le nutriscore

Fonctionnement :

- Imputation des variables manquantes avec KNNImputer de sklearn
- Réduction et normalisation des variables
- Projection sur les composantes principales
- Prediction du nutriscore avec le modèle retenu (knn avec ACP)

Test de la fonction sur des aliments de nutriscore élevé et faible.
Données issues de l'échantillon de test :

product_name	nutriscore_score	energy_100g	saturated_fat_100g	sugars_100g	fiber_100g	proteins_100g	sodium_100g
White Chocolate	**28**	2406.	25.	50.	0.0	7.5	0.1016
Fat Free Skim Milk	**-1**	138.0	0.0	4.58	0.0	3.33	0.0508

White Chocolate de nutriscore = **28** :

calculNutriACP(2406.0, 25.0, 50.0, 0.0, 7.5, 0.1016)	=	**28**
calculNutriACP(2406.0, 25.0, 50.0, 0.0, 7.5, **np.nan**)	=	**28**
calculNutriACP(2406.0, 25.0, 50.0, 0.0, **np.nan**, **np.nan**)	=	**28**
calculNutriACP(2406.0, 25.0, 50.0, **np.nan**, **np.nan**, **np.nan**)	=	**27**
calculNutriACP(2406.0, 25.0, **np.nan**, **np.nan**, **np.nan**, **np.nan**)	=	**23**
calculNutriACP(2406.0,**np.nan**, **np.nan**, **np.nan**, **np.nan**, **np.nan**)	=	**20**

Fat Free Skim Milk = **-1** :

calculNutriACP(138.0, 0.0, 4.58, 0.0, 3.33, 0.0508)	=	**-1**
calculNutriACP(138.0, 0.0, 4.58, 0.0, 3.33, **np.nan**)	=	**-1**
calculNutriACP(138.0, 0.0, 4.58, 0.0, **np.nan**, **np.nan**)	=	**-1**
calculNutriACP(138.0, 0.0, 4.58, **np.nan**, **np.nan**, **np.nan**)	=	**-1**
calculNutriACP(138.0, 0.0, **np.nan**, **np.nan**, **np.nan**, **np.nan**)	=	**0**
calculNutriACP(138.0, **np.nan**, **np.nan**, **np.nan**, **np.nan**, **np.nan**)	=	**4**

Table of Contents

Conclusion

- **Analyse des données** : échantillons $\neq$ loi normale

$\Rightarrow$ tous les tests (Khi2/ANOVA/Student) rejettent l'hypothèse nulle d'indépendance des variables

- **Projection sur les composantes principales (ACP)**

$\Rightarrow$ Corrélations entre (energy_100g et saturated_fat_100g),

$\Rightarrow$ Anticorrélation entre (proteins_100g et sugars_100g) et (entre fiber_100g, sodium_100g).

- **Construction du modèle du nutriscore**

$\Rightarrow$ knn avec ACP est retenu. L'ACP améliore peu les résultats (dû au fait du faible nombre de variables).

- **Fonction calcul du nutriscore avec données partielles**

$\Rightarrow$ Résultats cohérents pour produits de nutriscore élevé et faible

Perspectives

- Contacter les developpeurs d'appli comme Yuka pour améliorer notre outil et potentiellement leur service.

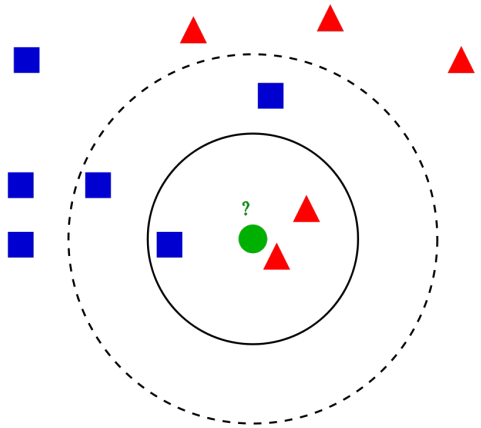
Merci de votre attention

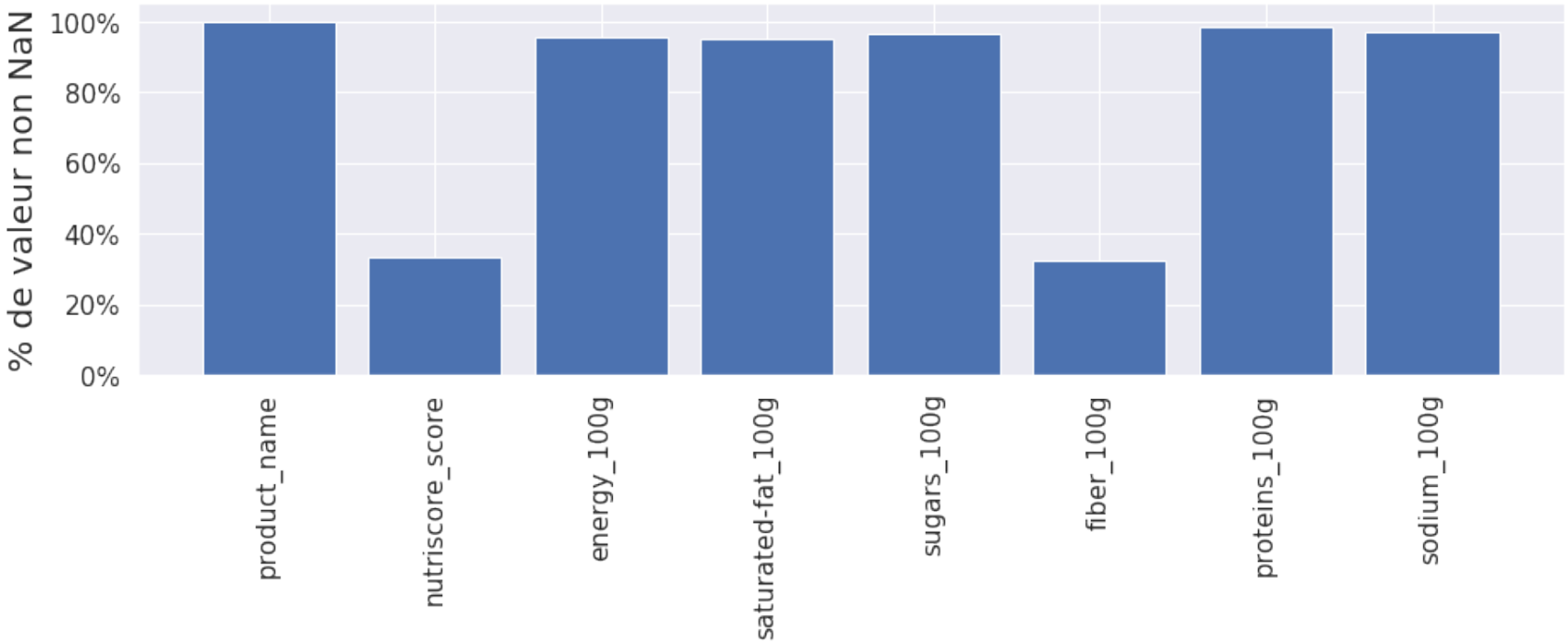
ANNEXES

Combinaisons de knn

- L'imputation automatique par KNNImputer de sklearn impossible (Pb memoire).

$\Rightarrow$ Choix d'algorithme des k plus proches voisins (k-nearest neighbors algorithm : knn) successifs :




% de valeur non NaN
100%
80%
60%
40%
20%
0%
product_name
nutriscore_score
energy_100g
saturated-fat_100g
sugars_100g
fiber_100g
proteins_100g
sodium_100g

Exemple : imputation des **40059** NaN de **energy_100g**

1. Imputation avec les variables d'entrainement et de test :
   trainTest =
   (1) **saturated_fat_100g**
   (2) **sugars_100g**
   (3) **fiber_100g**
   (4) **proteins_100g**
   (5) **sodium_100g**

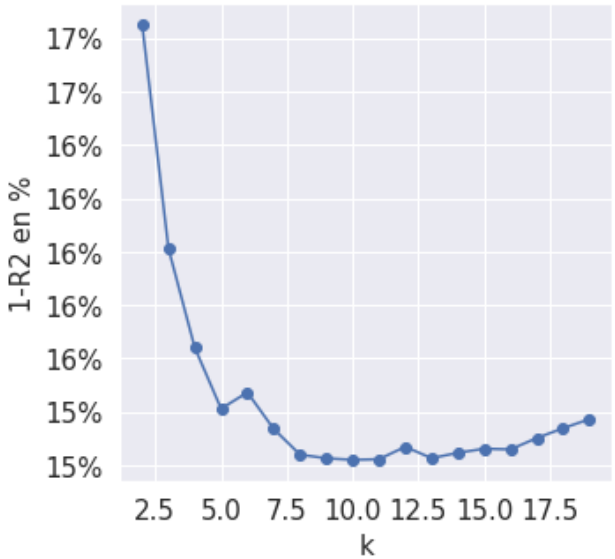
- Echantillon d'entrainement et de test :

  [**fit**] = dropNaN(**energy_100g** + **trainTest**) (264486 ind)

- Echantillon de prédiction :

  [Prediction] = dropNaN(**trainTest**) - dropNaN(**energy_100g** + **trainTest**) (5257 ind)

- si [Prediction] $\neq$ [] : On choisi k tel que knn.score([**fit**]) soit le plus petit possible :



- On **impute** [Prediction].**energy_100g** avec knn.predict()

$\Rightarrow$ Le nombre de NaN de la variable **energy_100g** restant = **34802** : L'imputation est incomplete

2. Imputation d'**energy_100g** avec les variables d'entrainement et de test **(1) (2) (3) (4)** :
   - **(1)** saturated_fat_100g
   - **(2)** sugars_100g
   - **(3)** fiber_100g
   - **(4)** proteins_100g

$\Rightarrow$ NaN d'**energy_100g** restant = **34734**

3. Imputation avec **[(1) (2) (3) (5)]**, **[(1) (2) (4) (5)]**, ...
4. Imputation avec **[(1) (2) (3)]**, **[(1) (2) (4)]**, ..., **[(1) (2)]**, ..., **[(1)]**, ...

30 knn plus tard... **energy_100g** est completement imputée.
Le même algo est appliquée pour les autres variables. Infine :