Projet P3

Conception d'une application au service de la Santé publique

OPENCLASSROOMS

Sommaire

1. Idée d'application
2. Nettoyage des données
3. Analyse pré-exploratoire
4. Conclusion

Idée Application

- Permettre au personne souffrant de diabètes de savoir si un aliment est sain ou pas dans leur cas

- Déterminer un modèles permettant de faire une approximation proche du nutriscore en utilisant les valeurs nutritionnelles les + pertinentes pour un diabétique

Nettoyage des données : Datasets

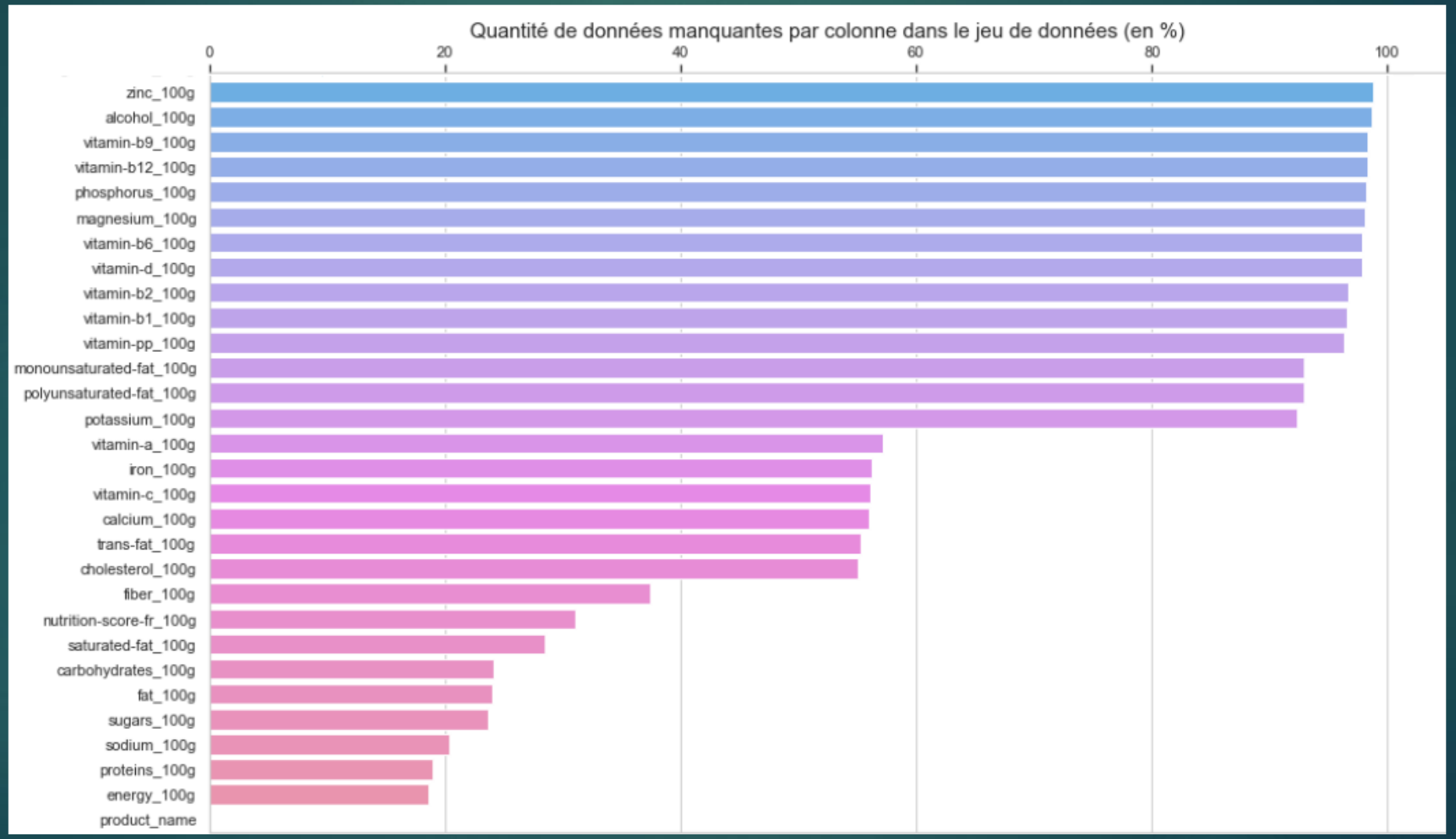
- Taille : 320772 × 162
- 90 variables nutritionnelles

	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name
0	3087	http://world-fr.openfoodfacts.org/produit/0000...	openfoodfacts-contributors	1474103866	2016-09-17T09:17:46Z	1474103893	2016-09-17T09:18:13Z	Farine de blé noir
1	4530	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489069957	2017-03-09T14:32:37Z	1489069957	2017-03-09T14:32:37Z	Banana Chips Sweetened (Whole)
2	4559	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489069957	2017-03-09T14:32:37Z	1489069957	2017-03-09T14:32:37Z	Peanuts
3	16087	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489055731	2017-03-09T10:35:31Z	1489055731	2017-03-09T10:35:31Z	Organic Salted Nut Mix
4	16094	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489055653	2017-03-09T10:34:13Z	1489055653	2017-03-09T10:34:13Z	Organic Polenta
...	...	...	...	...	...	...	...	...
320767	9948282780603	http://world-fr.openfoodfacts.org/produit/9948...	openfoodfacts-contributors	1490631299	2017-03-27T16:14:59Z	1491244498	2017-04-03T18:34:58Z	Tomato & ricotta
320768	99567453	http://world-fr.openfoodfacts.org/produit/9956...	usda-ndb-import	1489059076	2017-03-09T11:31:16Z	1491244499	2017-04-03T18:34:59Z	Mint Melange Tea A Blend Of Peppermint, Lemon ...
320769	9970229501521	http://world-fr.openfoodfacts.org/produit/9970...	tomato	1422099377	2015-01-24T11:36:17Z	1491244499	2017-04-03T18:34:59Z	乐吧泡菜味薯片
320770	9980282863788	http://world-fr.openfoodfacts.org/produit/9980...	openfoodfacts-contributors	1492340089	2017-04-16T10:54:49Z	1492340089	2017-04-16T10:54:49Z	Tomates aux Vermicelles
320771	999990026839	http://world-fr.openfoodfacts.org/produit/9999...	usda-ndb-import	1489072709	2017-03-09T15:18:29Z	1491244499	2017-04-03T18:34:59Z	Sugar Free Drink Mix, Peach Tea

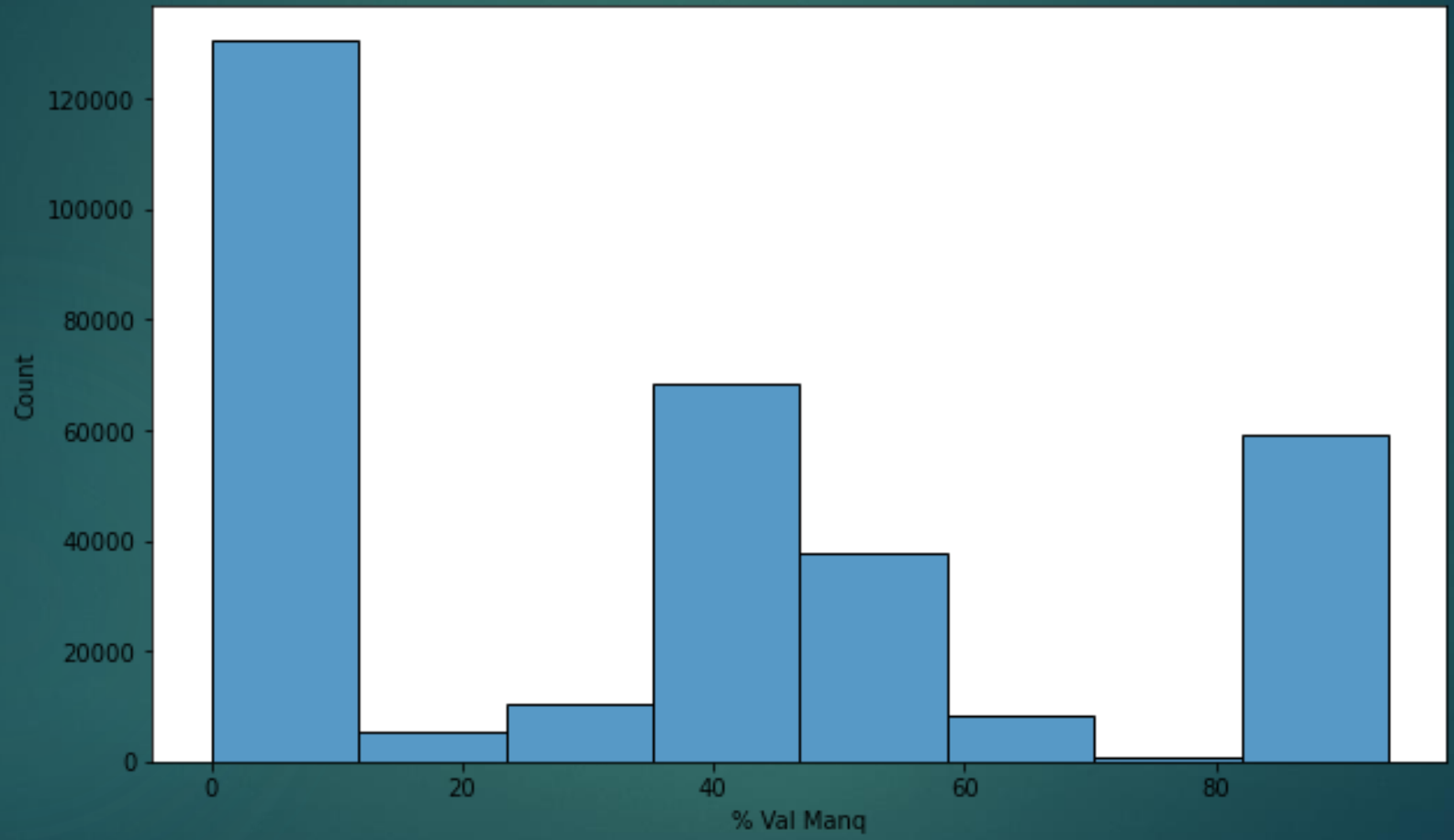
Nettoyage des données

1. Contrôle des colonnes en fonction du % de valeurs manquante
2. Contrôle des lignes en fonction du % des valeurs manquantes
3. Suppression des duplication et éléments non-ascii
4. Vérification des types
5. Vérification des valeurs aberrantes
6. Imputation via KNN imputer
7. Suppression des aliments avec une somme de macronutriment > 100g

Nettoyage des données : Contrôle des colonnes



Nettoyage des données : Contrôle des lignes



Titre : % de valeurs manquantes en fonction des lignes de notre dataset

Nettoyage des données : Traitement Outlier

Avant le traitement :

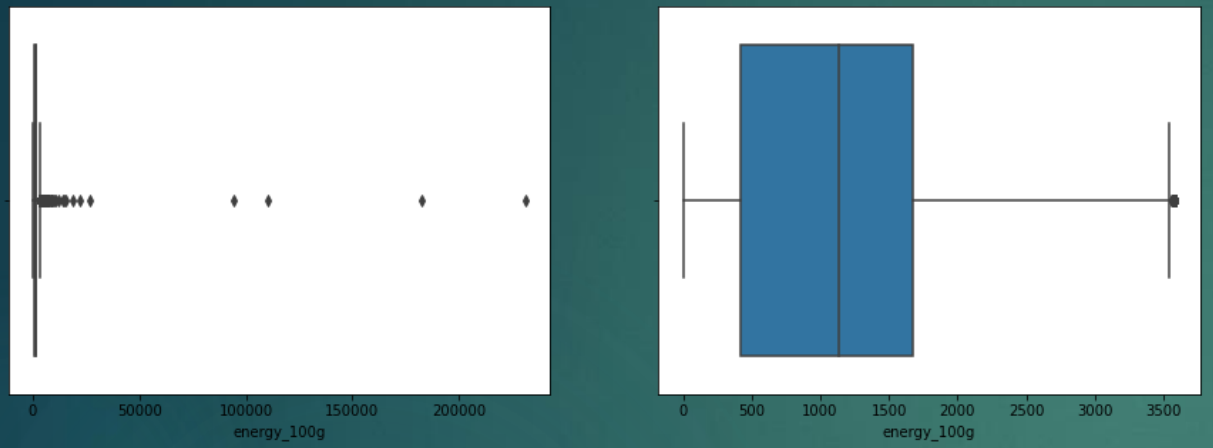
- 72576 élément dupliqué supprimés
- 94 éléments avec caractères non ascii

Pour le traitement :

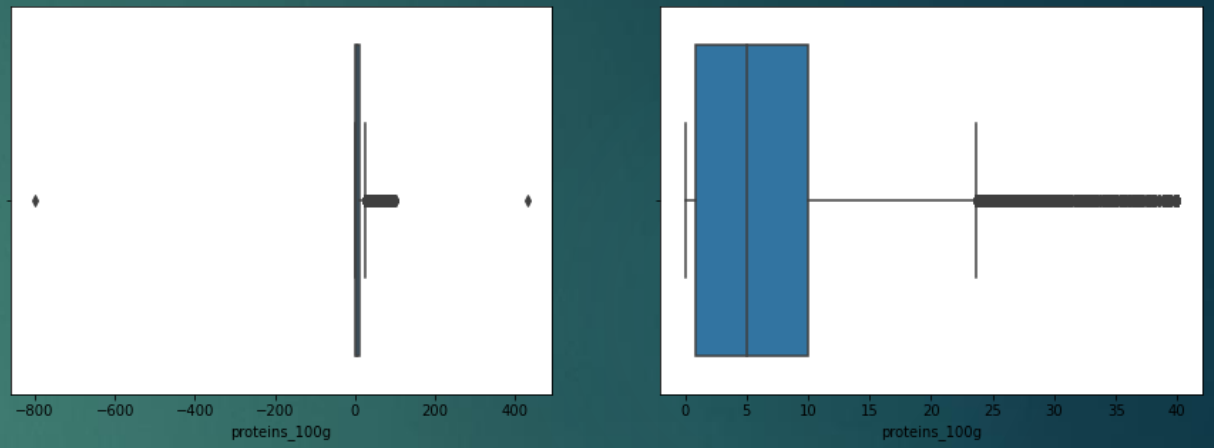
- Utilisation de la méthodes percentile
- Suppression des valeurs négatives (sauf nutriscore) et supérieurs à 100 (sauf energy)

Nettoyage des données : Traitement Outlier

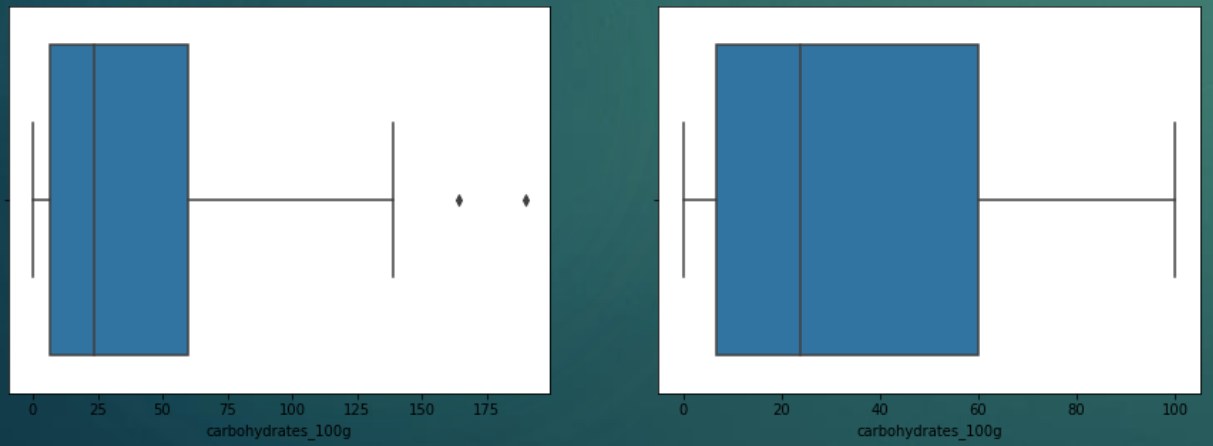
Boxplot avant et après traitement des outliers pour energy_100g



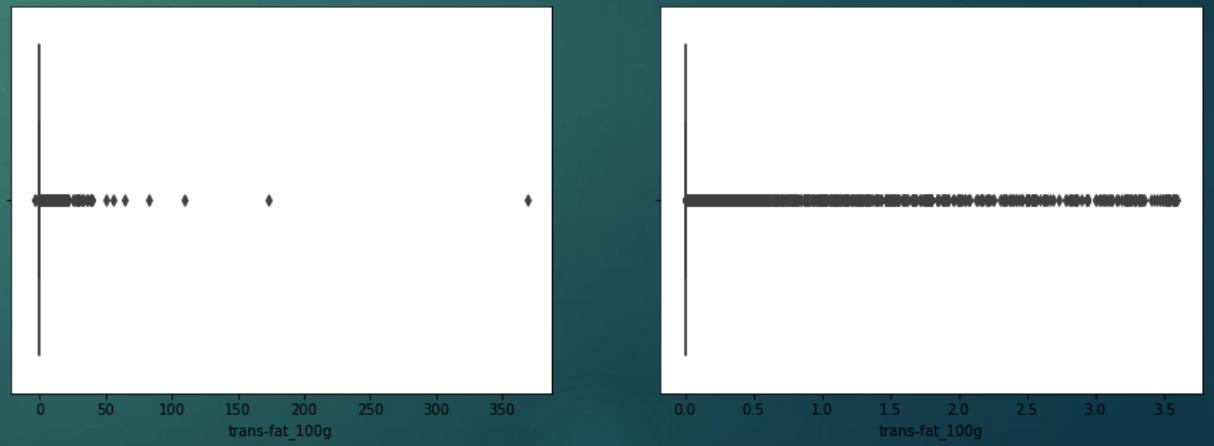
Boxplot avant et après traitement des outliers pour proteins_100g



Boxplot avant et après traitement des outliers pour carbohydrates_100g



Boxplot avant et après traitement des outliers pour trans-fat_100g



Nettoyage des données : Imputation KNN

```
col=list(dataset_reduit5.columns)
df_Av_imputation = dataset_reduit5.copy()[col[1:]]
```

executed in 29ms, finished 18:40:51 2022-05-01

```
imputer = KNNImputer(n_neighbors=3)
imputed = imputer.fit_transform(df_Av_imputation)
df_Ap_imputation = pd.DataFrame(imputed, columns=df_Av_imputation.columns)
```

executed in 27m 5s, finished 19:07:57 2022-05-01

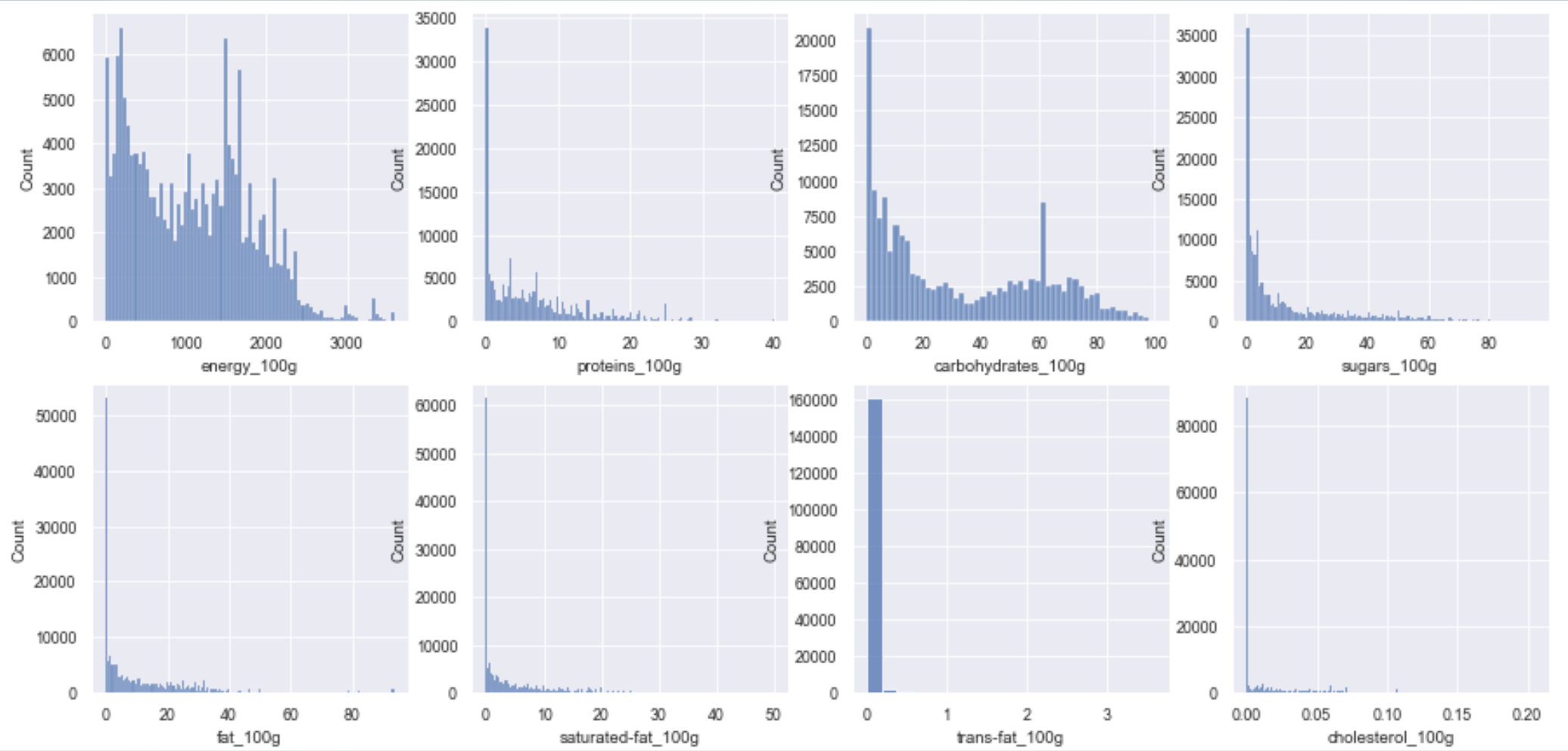
```
produit_name = dataset_reduit5['product_name'].tolist()
df_name=pd.DataFrame(produit_name,columns=['product_name'])
dataset_reduit6 = pd.concat((df_name,df_Ap_imputation),axis=1)
```

executed in 81ms, finished 19:12:12 2022-05-01

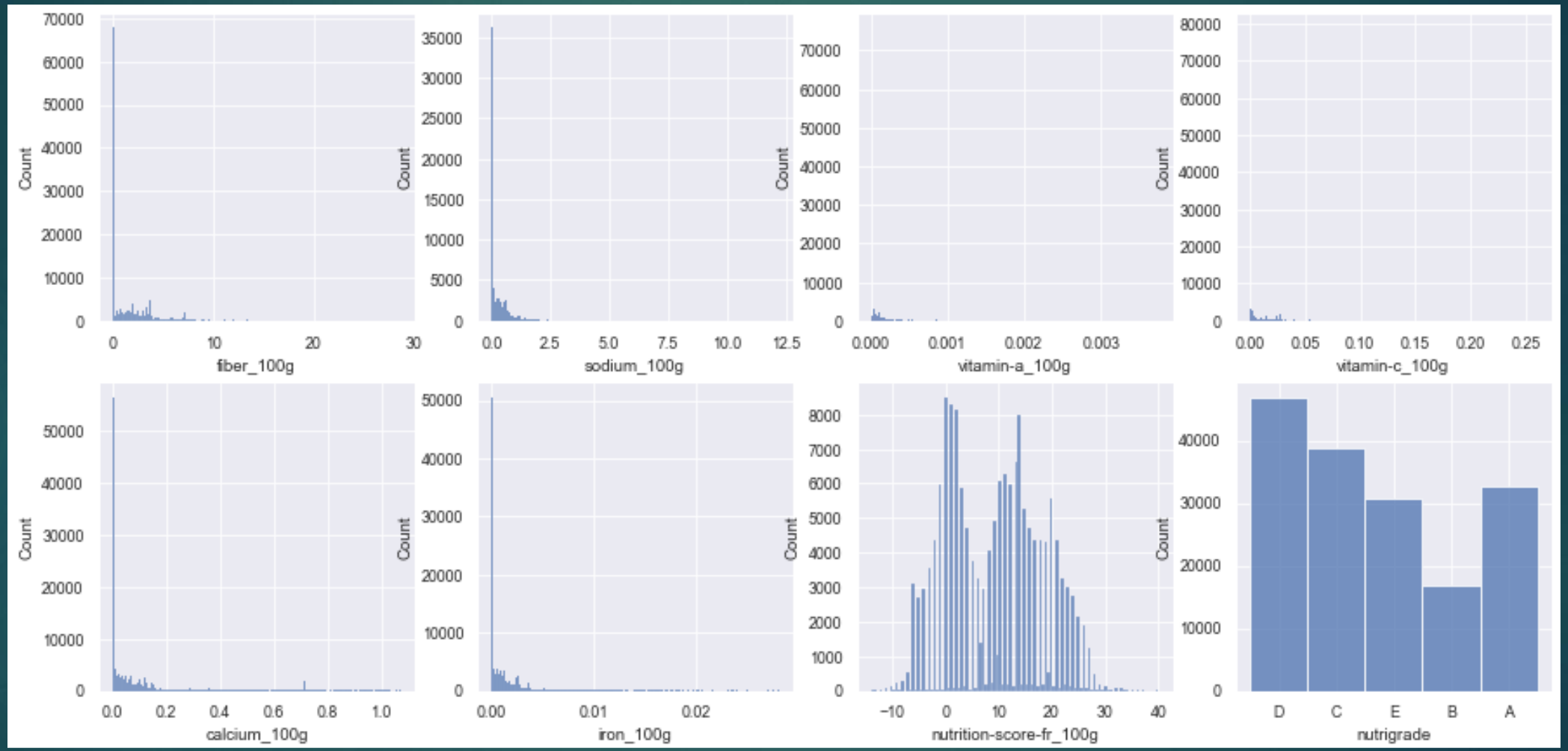
```
dataset_reduit6
```

executed in 25ms, finished 19:12:14 2022-05-01

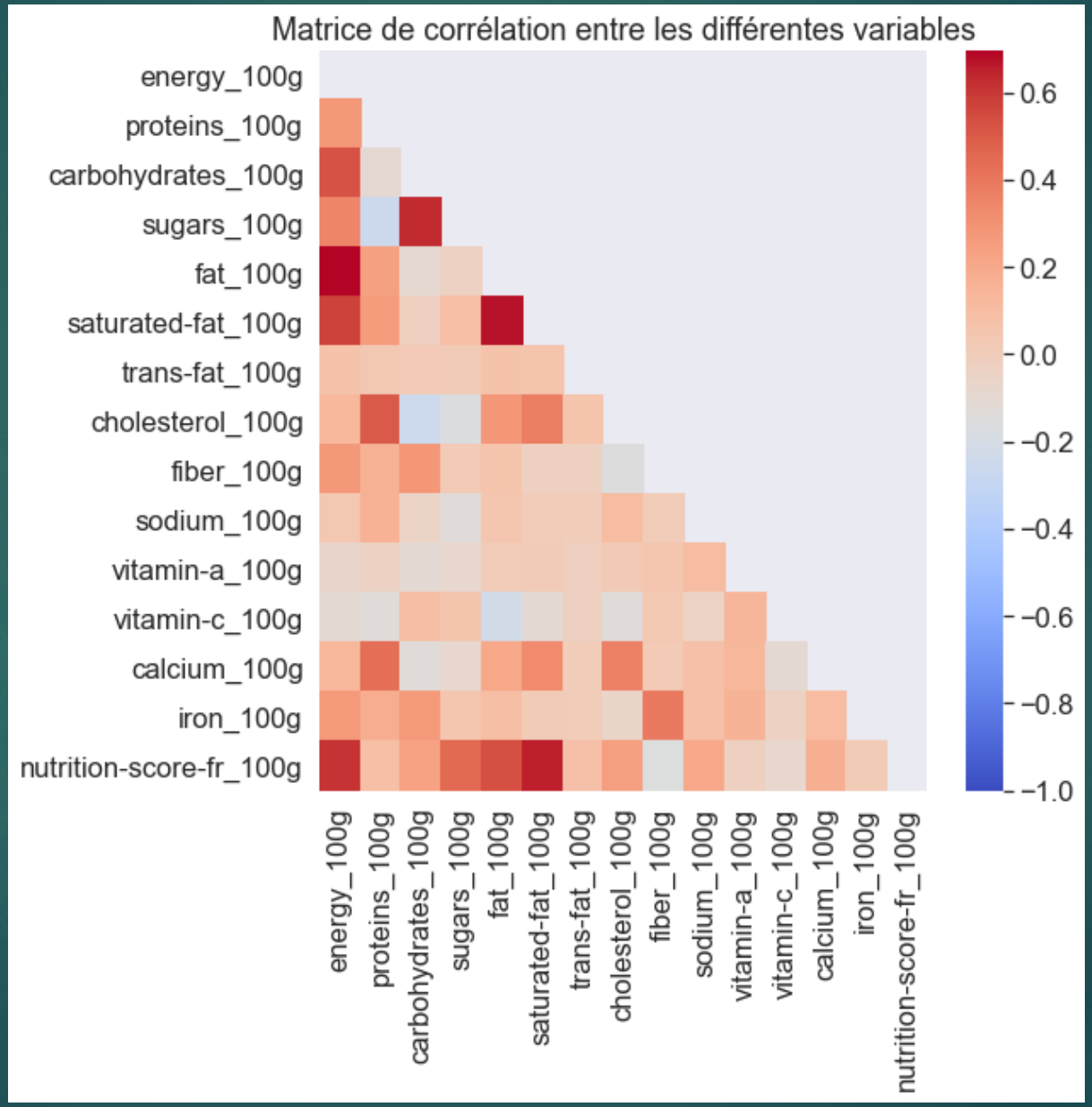
Analyse des données : Analyse Univariés



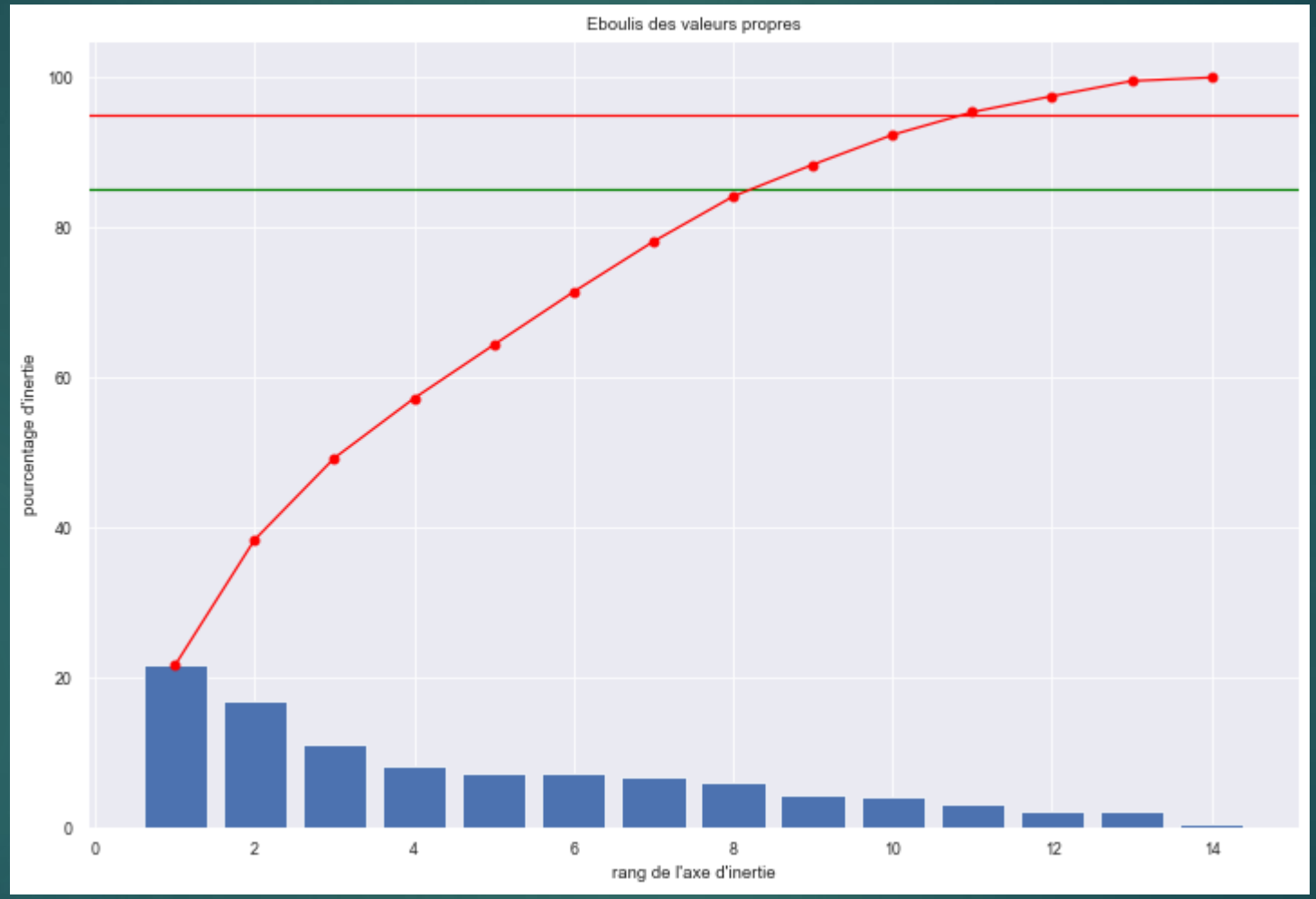
Analyse des données : Analyse Univariés



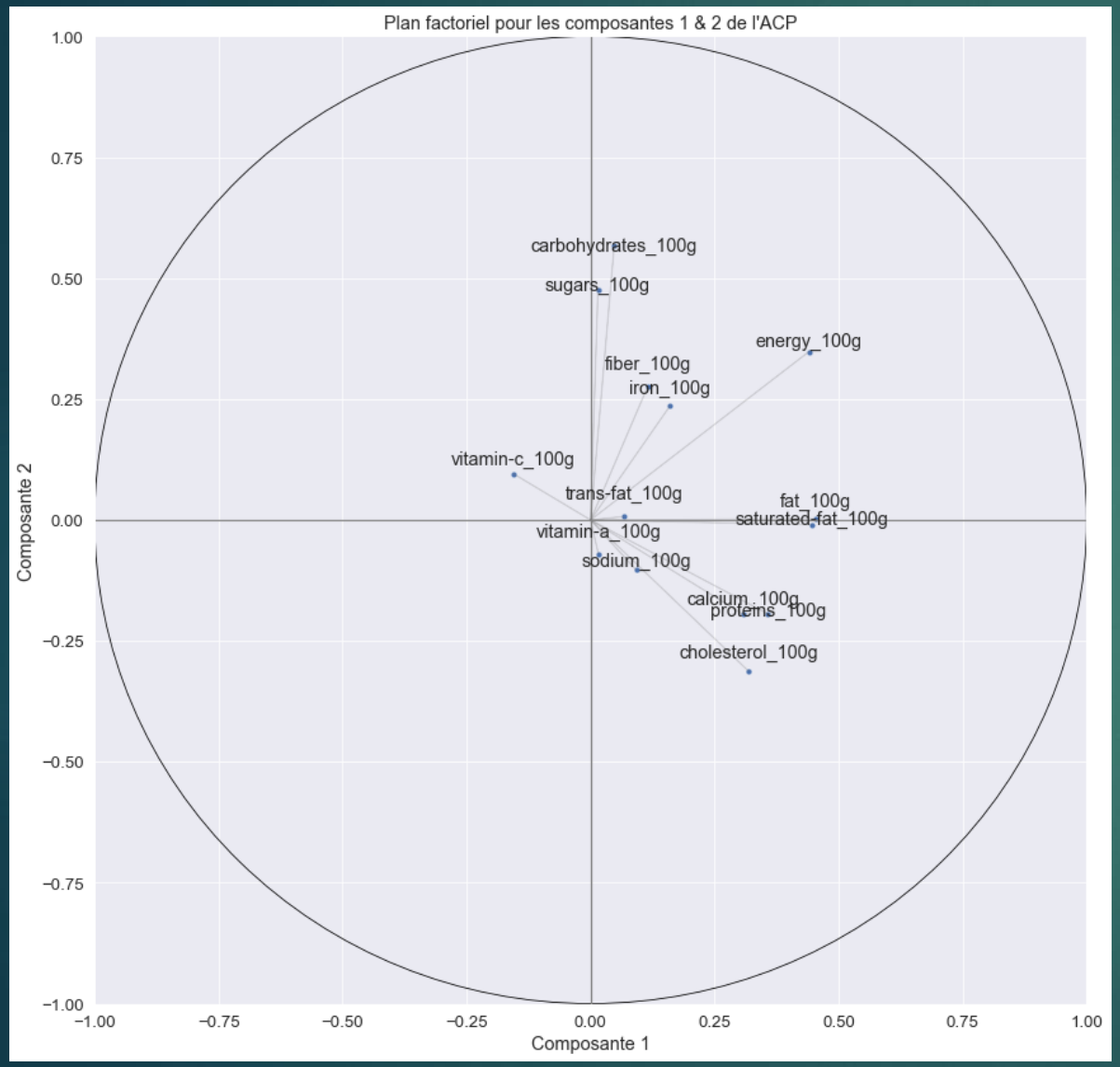
Analyse des données : Analyse Bivariés

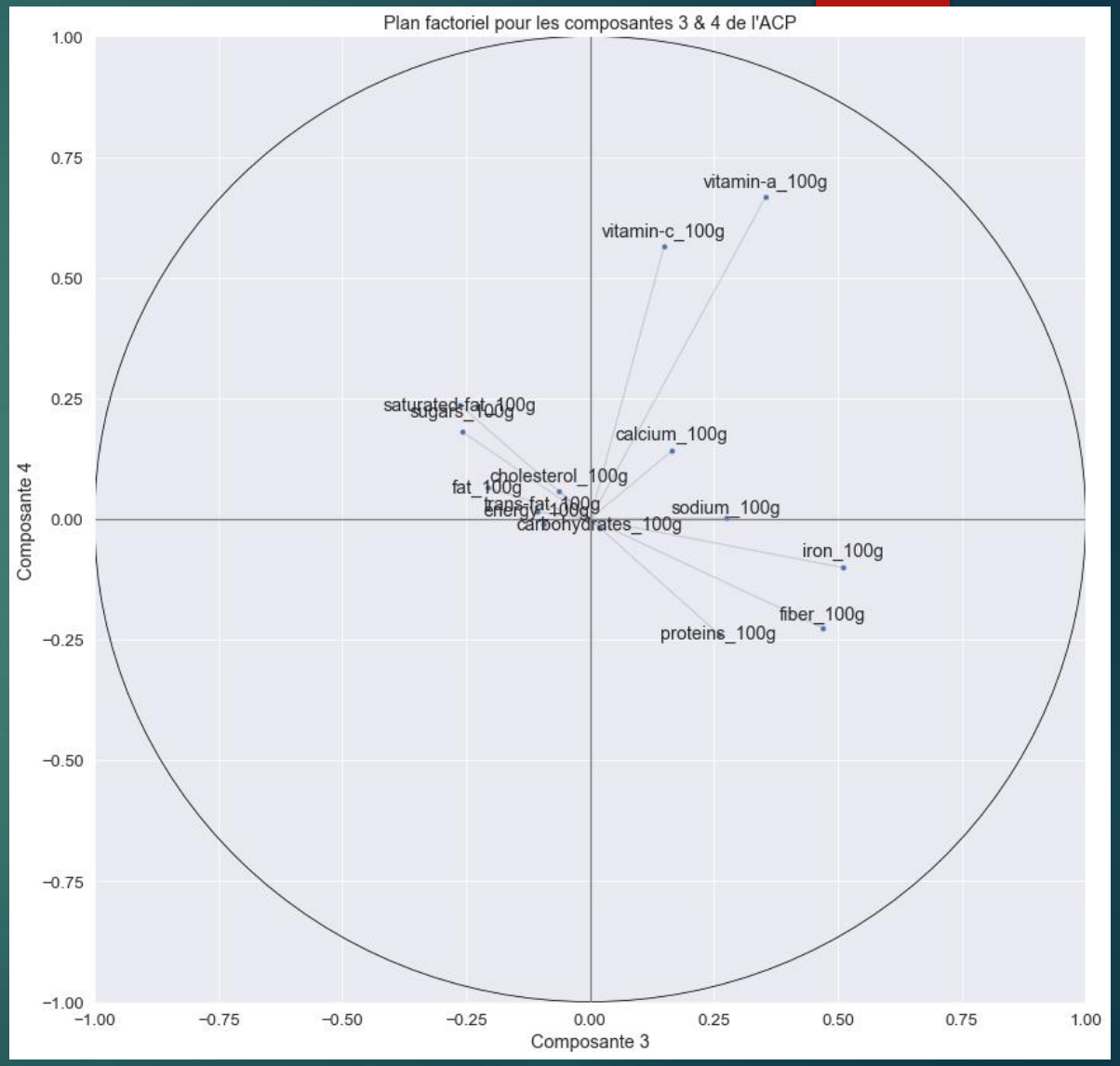


Analyse des données : ACP



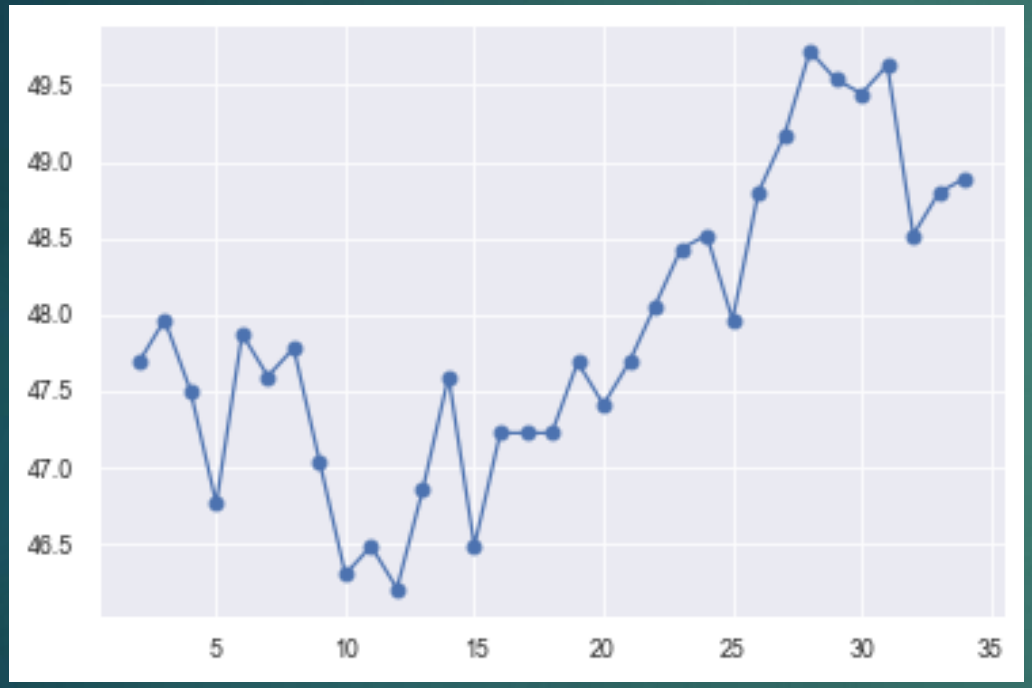
Analyse des données : ACP





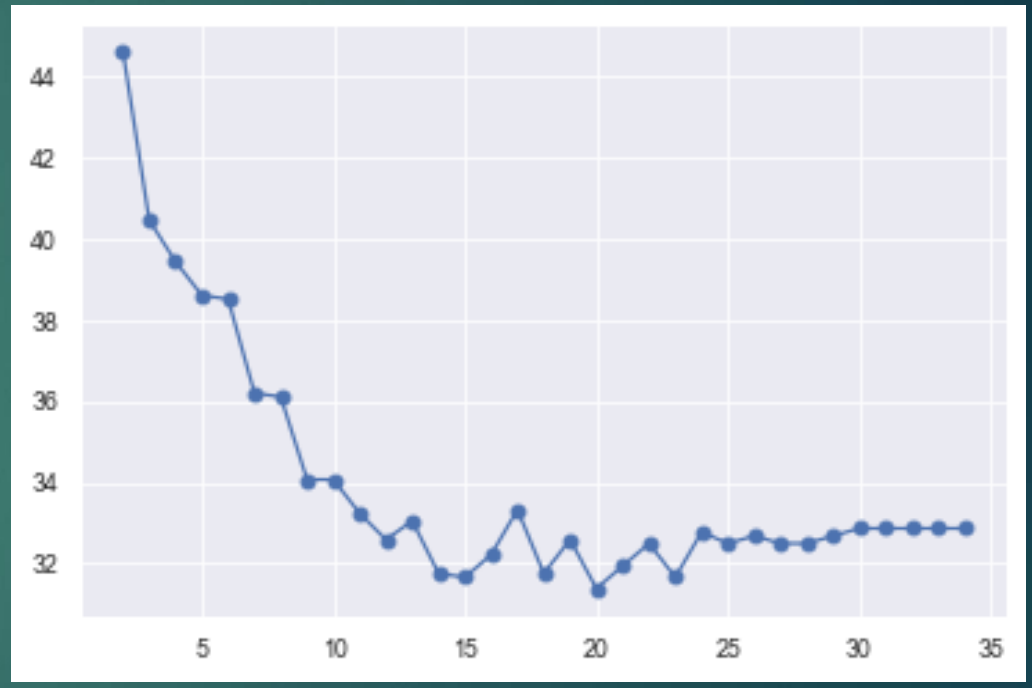
Analyse des données : Modèle d'approximation du nutriscore

Modèle kNN



Evolution du taux d'erreur en fonction du nombre de voisin

Modèle Random Forest



Evolution du taux d'erreur en fonction de ma profondeur de l'arbre

Analyse des données : Modèle d'approximation du nutriscore

Points	energy_100g	saturated-fat_100g	sugars_100g	sodium_100g
0	0	0	0.0	0
1	335	1	4.5	0.90
2	670	2	9.0	1.80
3	1005	3	13.5	2.70
4	1340	4	18.0	3.60
5	1675	5	22.5	4.50
6	2010	6	27.0	5.40
7	2345	7	31.0	6.30
8	2680	8	36.0	7.20
9	3015	9	40.0	8.10
10	3350	10	45.0	9.00

Tableau N : Point negative calcul nutricore

Points	fiber_100g	proteins_100g
0	0.0	0.0
1	0.9	1.6
2	1.9	3.2
3	2.8	4.8
4	3.7	6.4
5	4.7	8.0

Tableau P : Point positive calcul nutricore

Calcul nutriscore = N-P

Analyse des données : Modèle d'approximation du nutriscore

```python
def calcul_nutriscore (df,tab_n,tab_p):
    dataset=df.copy()
    col_n = ['energy_100g','sugars_100g','saturated-fat_100g']
    col_p = ['fiber_100g','proteins_100g']
    dataset['nutriscore_predit']=0

    for j in dataset.index :
        for i in col_p :
            p=tab_p[tab_p[i]<=float(dataset[i][j])].sort_values(by=i,ascending=False).index[0]
            dataset['nutriscore_predit'][j]-=tab_p['Points'][p]

        for k in col_n :
            n=tab_n[tab_n[k]<=dataset[k][j]].sort_values(by=k,ascending=False).index[0]
            dataset['nutriscore_predit'][j]+=tab_n['Points'][n]

    dataset['nutrigrade_predit'] = 'E'
    dataset.loc[dataset['nutriscore_predit']<19,'nutrigrade_predit']='D'
    dataset.loc[dataset['nutriscore_predit']<11,'nutrigrade_predit']='C'
    dataset.loc[dataset['nutriscore_predit']<3,'nutrigrade_predit']='B'
    dataset.loc[dataset['nutriscore_predit']<=0,'nutrigrade_predit']='A'

    return dataset
```

Avec ce calcul on arrive à un taux d'erreur de 56%

Conclusion

- Nutriscore corrélé positivement avec le sucres, l'énergie, le gras saturés
- Variables intéressant pour les personnes diabétiques
- Le modèle est pertinente car le calcul avec la formule du nutriscore ne permet pas une bonne prédiction avec les variables d'intérêt
- La prédiction du modèles est correcte mais reste perfectible (faire varier d'autres paramètres)

Merci de votre attention