Préparez des données pour un organisme de santé publique

OpenClassrooms

Benjamin Demaille

Les données

Vue générale

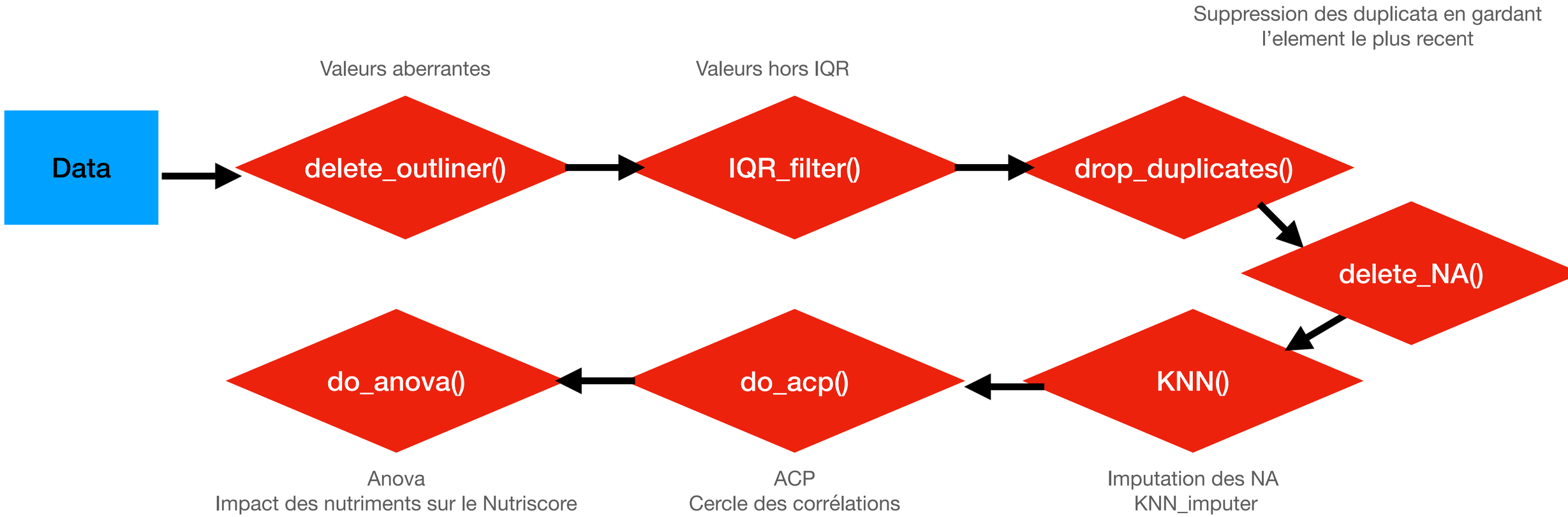
162 variables

320 749 entrées

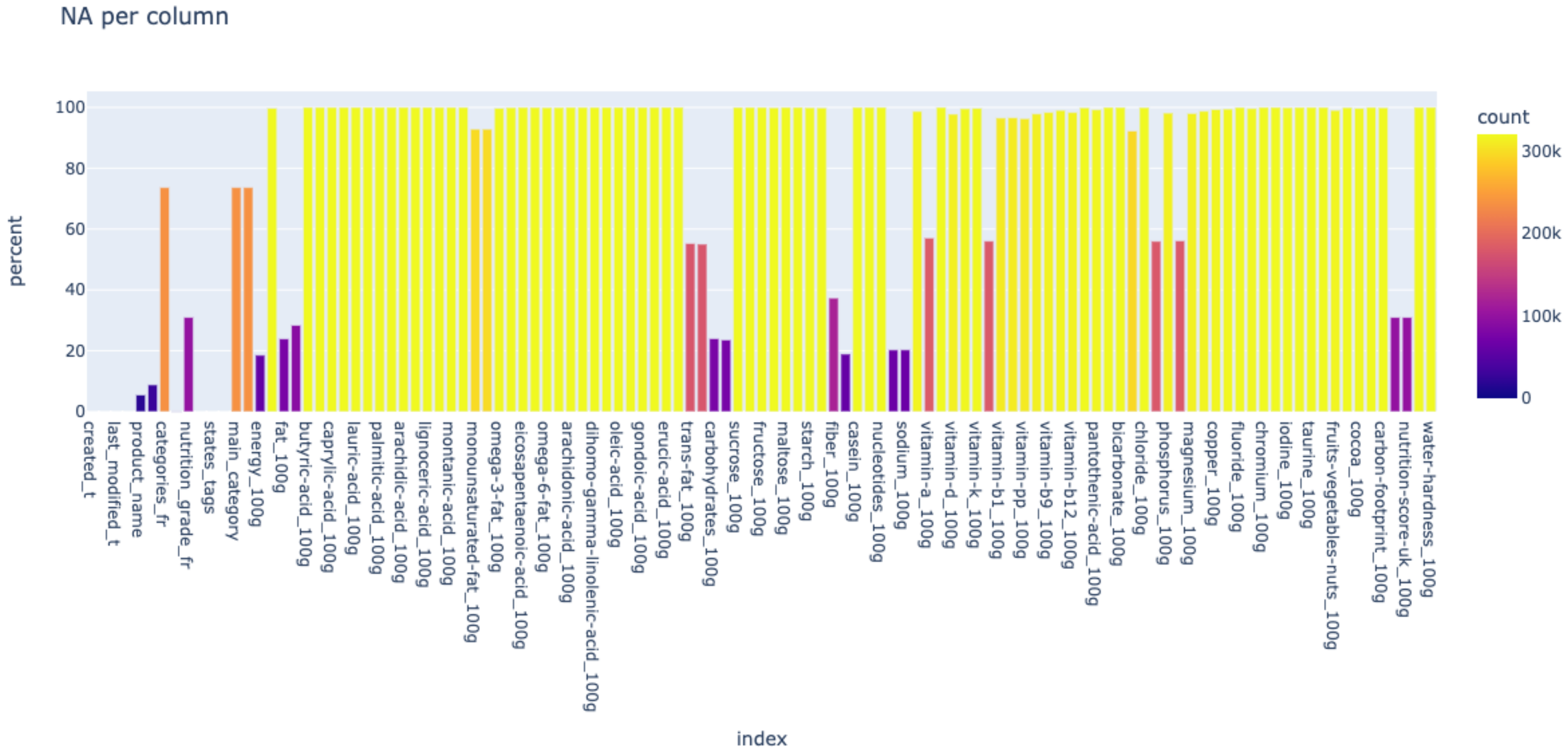
	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime	product_name	generic_name	quantity	...	ph_100g	fruits-vegetables-nuts_100g	collagen-meat-protein-ratio_100g	cocoa_100g	chlorophyl_100g	carbon-footprint_100g	nutrition-score-fr_100g	nutri s uk_
0	0000000003087	http://world-fr.openfoodfacts.org/produit/0000...	openfoodfacts-contributors	1474103866	2016-09-17T09:17:46Z	1474103893	2016-09-17T09:18:13Z	Farine de blé noir	NaN	1kg	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0000000004530	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)	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	14.0	
2	0000000004559	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489069957	2017-03-09T14:32:37Z	1489069957	2017-03-09T14:32:37Z	Peanuts	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	0.0	
3	0000000016087	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	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	12.0	
4	0000000016094	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489055653	2017-03-09T10:34:13Z	1489055653	2017-03-09T10:34:13Z	Organic Polenta	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
320744	9948282780603	http://world-fr.openfoodfacts.org/produit/9948...	openfoodfacts-contributors	1490631299	2017-03-27T16:14:59Z	1491244498	2017-04-03T18:34:58Z	Tomato & ricotta	NaN	1	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
320745	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 ...	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	0.0	
320746	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	Leba pickle flavor potato chips	50 g	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
320747	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	NaN	67g	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
320748	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	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

320749 rows × 162 columns

Pipeline


Suppression des duplicata en gardant l'element le plus recent
Valeurs aberrantes
Valeurs hors IQR
Data
delete_outliner()
IQR_filter()
drop_duplicates()
delete_NA()
KNN()
do_acp()
do_anova()
Anova
Impact des nutriments sur le Nutriscore
ACP
Cercle des corrélations
Imputation des NA
KNN_imputer

NA avant traitement



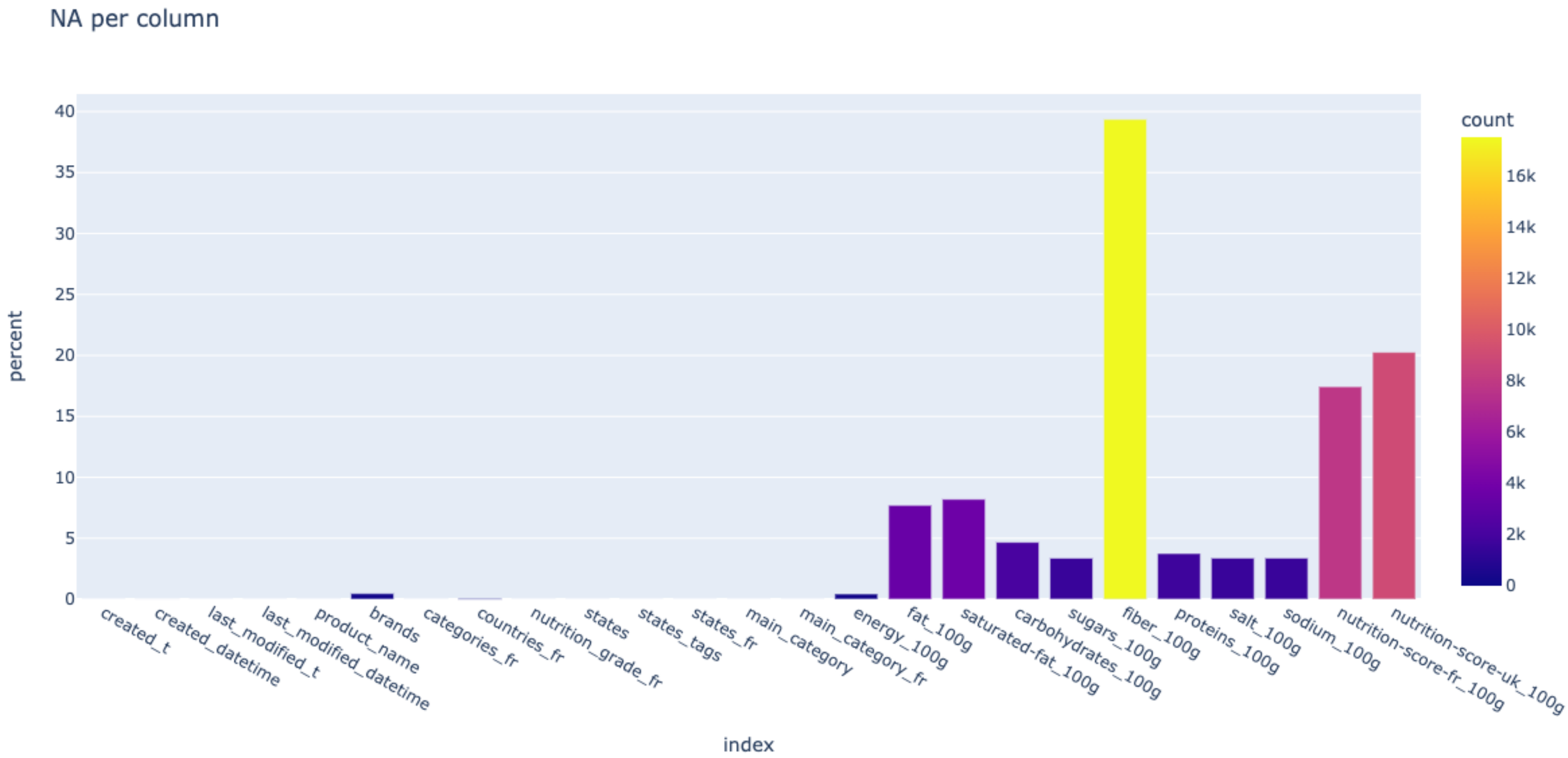
delete_outlier()

```
def delete_outliers(data: pd.DataFrame, Nutritive_colunms: list):
    for col in Nutritive_colunms:
        data.loc[:,col] = data[col].where(data[col] >= 0)
        data.loc[:,col] = data[col].where(data[col] <= 100)
    return(data)
```

IQR_filter()

```
def IQR_filter(s: pd.DataFrame, col_num: list, replace=np.nan):
    for col in col_num:
        Q1 = s[col].quantile(0.25)
        Q3 = s[col].quantile(0.75)
        IQR = Q3-Q1
        s.loc[:,col] = s[col].where((s[col] > (Q1 - 1.5 * IQR)) & (s[col] < (Q3 + 1.5 * IQR)))
    return(s)
```

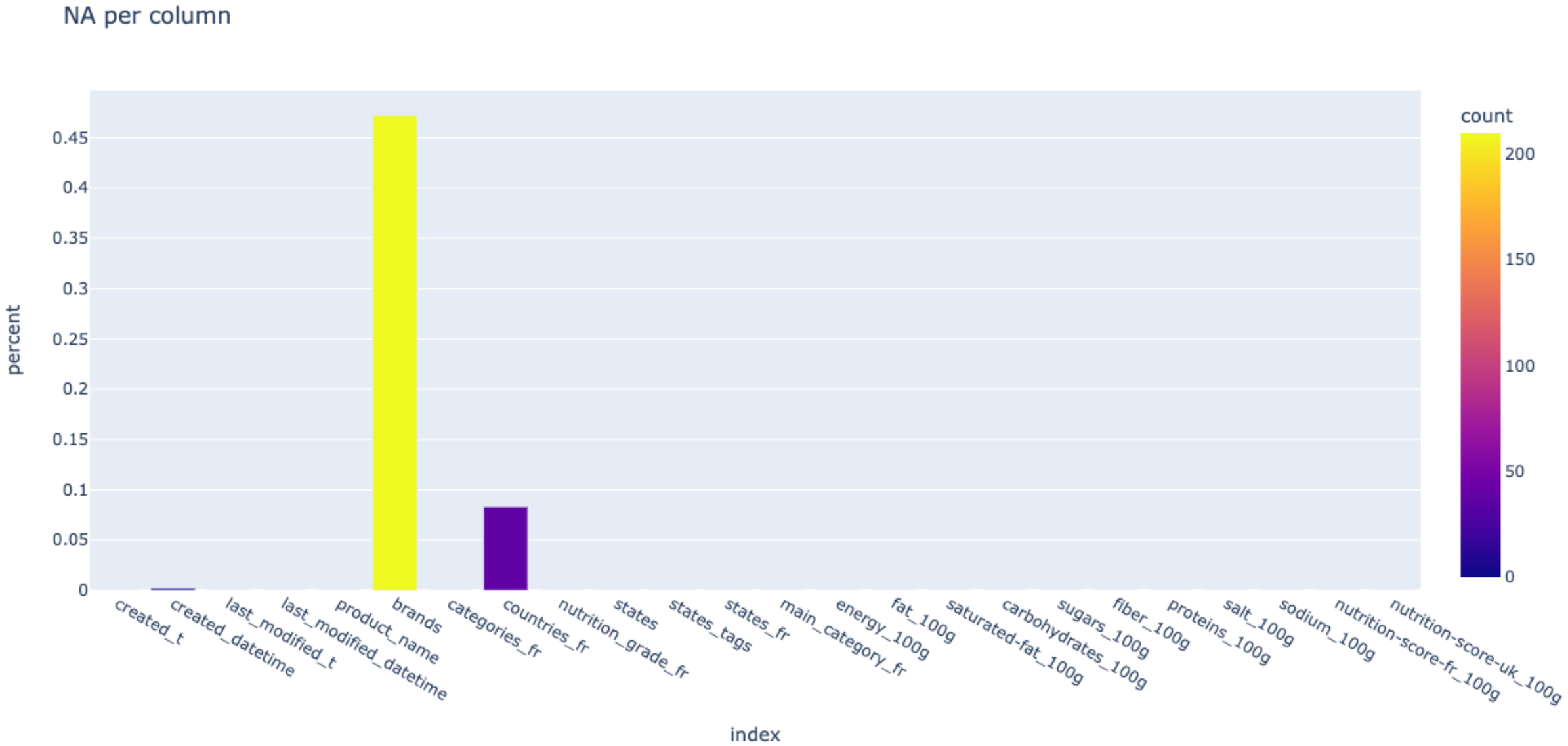
NA après filtre



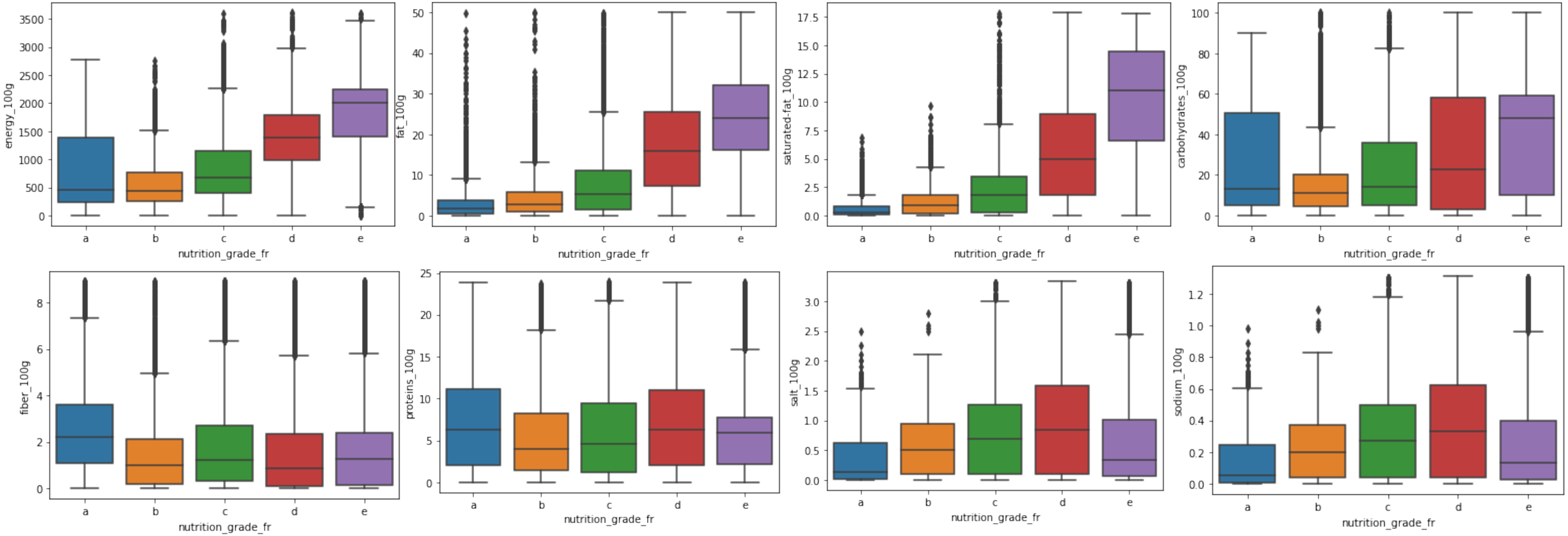
KNN()

```
def KNN(data):
    cat = list(['created_t', 'created_datetime',
       'last_modified_t', 'last_modified_datetime', 'product_name', 'brands',
       'categories_fr',
       'countries_fr', 'nutrition_grade_fr',
       'states', 'states_tags', 'states_fr',
       'main_category_fr'])
    dataKNN = pd.DataFrame(KNNImputer(n_neighbors=5).fit_transform(data.filter(regex='_100g')),
                           columns = data.filter(regex='_100g').columns)
    data = pd.concat([data[cat].reset_index(drop=True),
                      dataKNN.reset_index(drop = True)], axis=1)
    return(data)
```

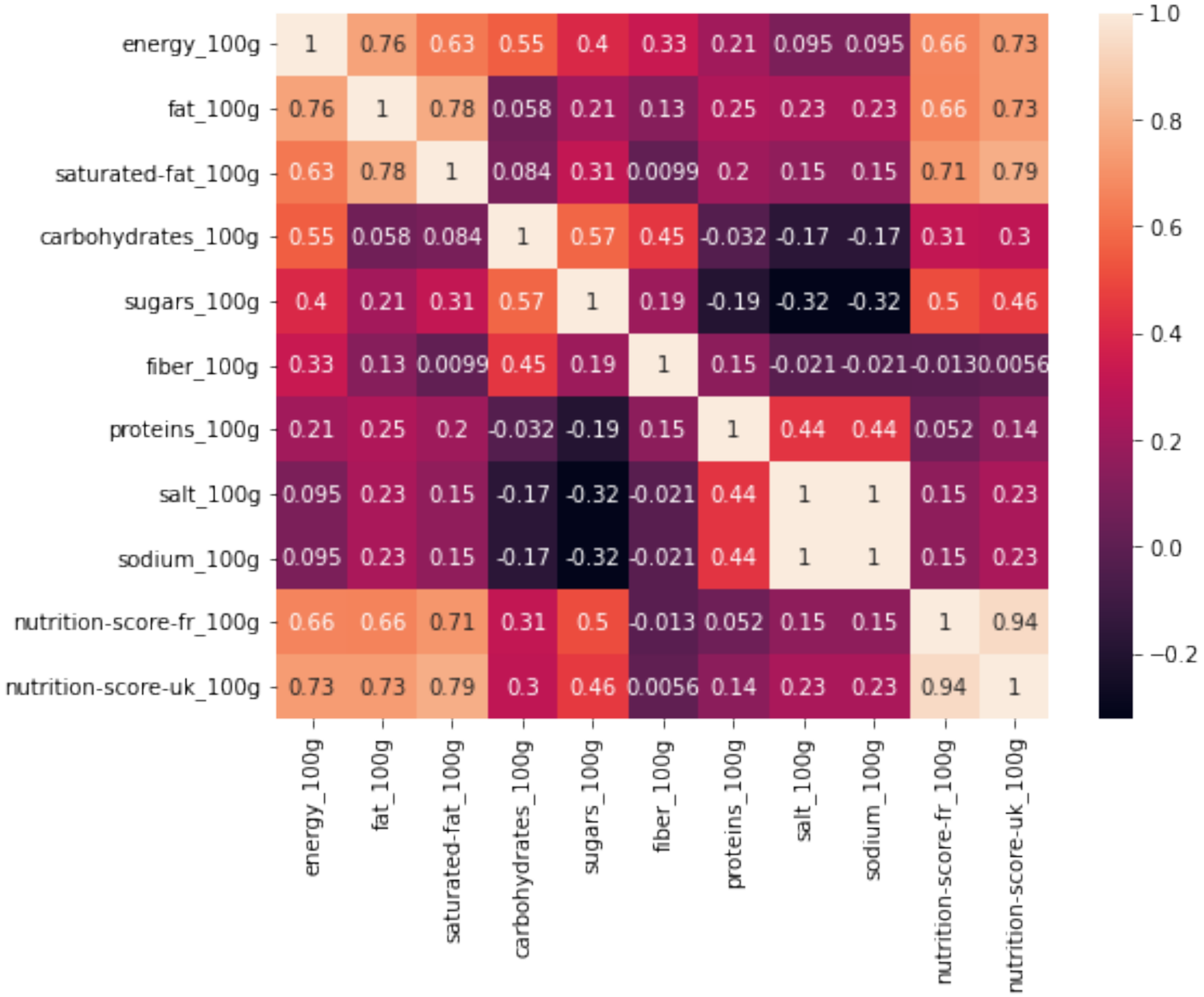
NA après KNN


NA per column
percent
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
count
200
150
100
50
0
created_t
created_datetime
last_modified_t
last_modified_datetime
product_name
brands
categories_fr
countries_fr
nutrition_grade_fr
states
states_tags
states_fr
main_category_fr
energy_100g
fat_100g
saturated-fat_100g
carbohydrates_100g
sugars_100g
fiber_100g
proteins_100g
salt_100g
sodium_100g
nutrition-score-fr_100g
nutrition-score-uk_100g
index

Analyse univariée



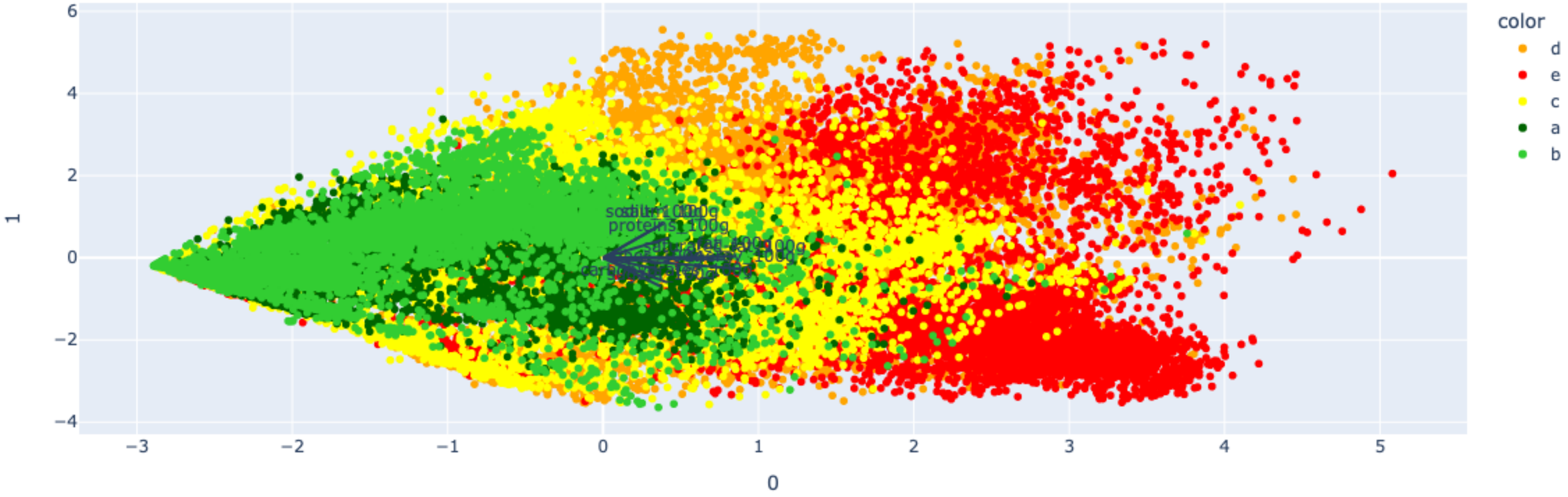
Corrélation



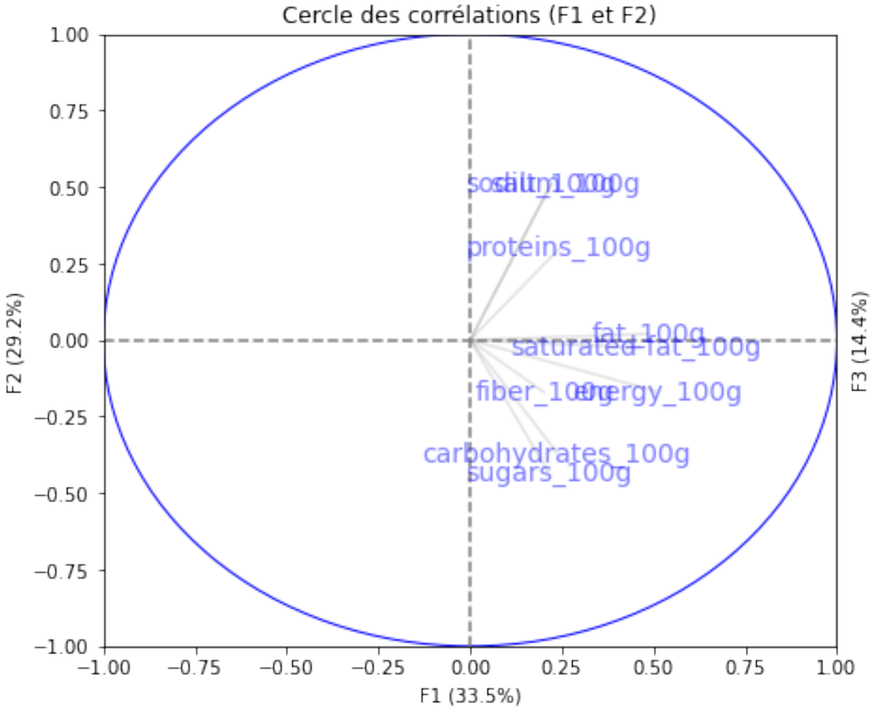
do_acp()

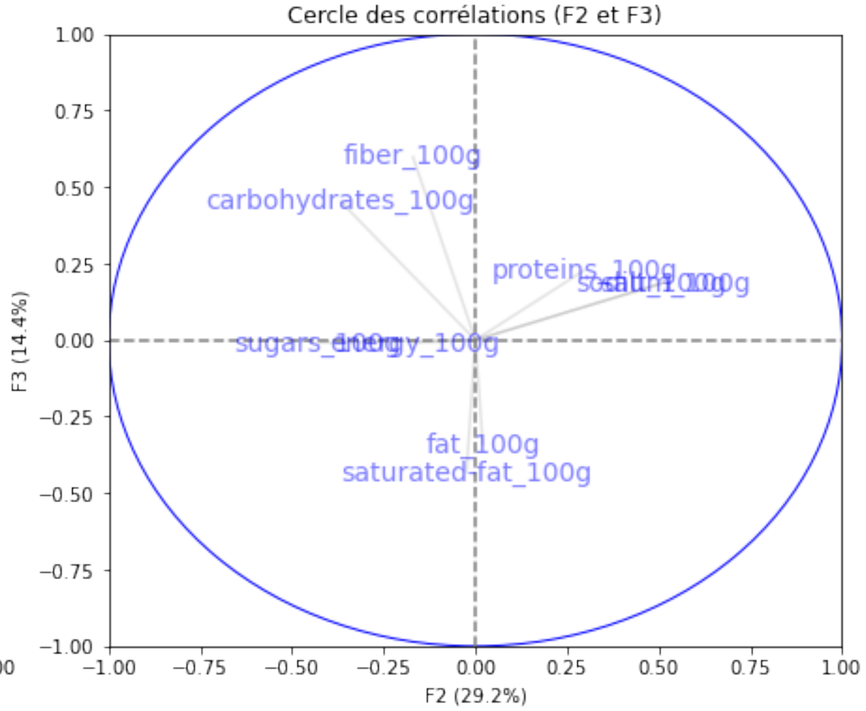
```
def do_pca(data):
    data2  = data.filter(regex='_100g').drop(['nutrition-score-fr_100g','nutrition-score-uk_100g'], axis=1)
    data2 = StandardScaler().fit_transform(data2)
    acp =  PCA(n_components=data2.shape[1]).fit_transform(data2)
    #3D  PCA
    n_comp = data2.shape[1]
    fig = px.scatter_3d(
        acp, x=0, y=1, z=2, color=data.nutrition_grade_fr, color_discrete_map= col,
        title= "3D PCA",
        labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'})
    fig.show()
    # Explained variance
    pca = PCA()
    pca.fit(data2)
    pcs = pca.components_
    exp_var_cumul = np.cumsum(pca.explained_variance_ratio_)
    px.area(
        x=range(1, exp_var_cumul.shape[0] + 1),
        y=exp_var_cumul,
        labels={"x": "# Components", "y": "Explained Variance"})
    loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
    features = data.filter(regex='_100g').drop(['nutrition-score-fr_100g','nutrition-score-uk_100g'], axis=1).columns
    fig = px.scatter(acp, x=0, y=1, color=data['nutrition_grade_fr'], color_discrete_map= col)
    for i, feature in enumerate(features):
        fig.add_shape(
            type='line',
            x0=0, y0=0,
            x1=loadings[i, 0],
            y1=loadings[i, 1]
        )
        fig.add_annotation(
            x=loadings[i, 0],
            y=loadings[i, 1],
            ax=0, ay=0,
            xanchor="center",
            yanchor="bottom",
            text=feature,
        )
    fig.show()
    #Cercle des corelations
    display_circles(pcs, n_comp, pca, [(0,1), (1,2), (3,4)], labels=features)
```

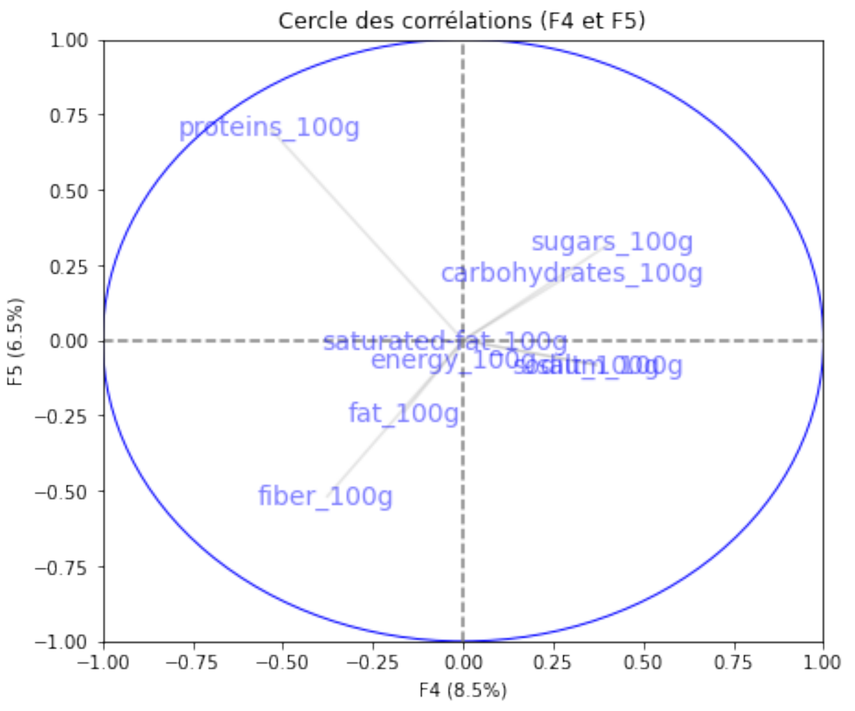
ACP



ACP


Cercle des corrélations (F1 et F2)
proteins_100g
fat_100g
saturated-fat_100g
fiber_100g
energy_100g
carbohydrates_100g
sugars_100g
F1 (33.5%)
F2 (29.2%)


Cercle des corrélations (F2 et F3)
fiber_100g
carbohydrates_100g
proteins_100g
fat_100g
saturated-fat_100g
F2 (29.2%)
F3 (14.4%)


Cercle des corrélations (F4 et F5)
proteins_100g
sugars_100g
carbohydrates_100g
saturated-fat_100g
fat_100g
fiber_100g
F4 (8.5%)
F5 (6.5%)

do_anova()

```
def do_anova(data):
    X = "nutrition_grade_fr" #qualitative
    Y = data.filter(regex='_100g').columns #quantitative
    anova = []
    nutriment = []
    for y in Y:
        anova.append(eta_squared(data[X],data[y]))
        nutriment.append(y)
    return(pd.DataFrame(np.column_stack([nutriment, anova]), columns=['nutriment', 'anova']))
```

ANOVA

nutriment	anova
energy_100g	0.3163253651456699
fat_100g	0.36039822889719414
saturated-fat_100g	0.4405205482601115
carbohydrates_100g	0.05626378061387536
sugars_100g	0.2256717021893017
fiber_100g	0.04266394894237031
proteins_100g	0.011313202357831198
salt_100g	0.08407891418818202
sodium_100g	0.08407815628274008

Conclusion

- Erreur de séparateur
- Énormément de données manquante calculable si on a la formule
- Analyse par catégorie ou marque ou pays d’origine pourrai nous en apprendre plus