

**Ecole Polytechnique de l’Université François Rabelais de Tours**

**Spécialité Informatique**

**64 avenue Jean Portalis**

**37200 Tours, France**

**Tél. +33 (0)2 47 36 14 14**

**[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr/)**

**Projet Libre 2019**

**Visualisation de la base de données complexe**

**Tuteurs académiques Étudiants**

*Gilles VENTURINI Alafate ABULIMITI (DI5)*

*Yuanyuan LI (DI5)*

# Table des matières

[**Table des matières**](#_qomfw23lgabx) **2**

[**1 Introduction**](#_79bcvseb89g2) **3**

[**2 Description générale**](#_ryfnlnwljwk6) **3**

[**3 Analyse et conception**](#_3qygnbsv8dlq) **4**

[3.1. OpenFood](#_3yts14dlwy1d) 4

[3.2. Choix de techenique](#_9yv9q27iow7t) 5

[3.2.1. Controleur de version du projet](#_4pwb70iuu4z3) 5

[3.2.2. Etat de l’art](#_gol13tx669kf) 6

[3.2.3. La methodé 2-Gram](#_n3xvpqyvi4p9) 7

[**4 Mise en oeuvre**](#_gugtln6okwgd) **7**

[4.1. Présentation de données](#_evdso0w91dfb) 7

[4.2. Traitement de données](#_gpuqkkcqrepg) 8

[4.2.1.Choix d’attributs](#_j711zqe2jhaj) 8

[4.2.2. Prétraitement de attributs](#_htwgi89nu9un) 9

[4.3. Visualisation interactive](#_w3i2oipp1cpx) 12

[4.3.1. Version 1.0 - Matplotlib](#_g93e1gt84dlt) 12

[4.2.2. Version 2.0 - Plotly.Scatter](#_7banosojk149) 13

[4.2.3. Version 3.0 - Plotly.Scattergl](#_wide35gv89y) 15

[4.4. Analyse de résultats](#_2k0ow4ydzk0p) 15

[4.4.1. Visualisation de résultats](#_ouf7ahxubfpk) 15

[4.4.2. La performance et la qualité](#_3bym2dkpn81f) 18

[**5 Bilan et conclusion**](#_al4jsufr31p3) **18**

[5.1. Faits](#_m84qxuqyazu4) 18

[5.2.Reste à faire](#_q3pyjlvqla3p) 18

[5.3. Bilan auto-critique](#_ao65t767cmt7) 18

# 1 Introduction

La visualisation des données est un domaine très important dans le domaine de la science des données et une méthode couramment utilisée en science de la décision. Dans ce projet, nous (Alafate ABULIMITI et Yuanyuan LI) utiliserons la méthode décrite dans [cet article](https://link.springer.com/content/pdf/10.1007%2Fs10619-015-7174-1.pdf) (On visualizing large multidimensional datasets with a multi-threaded radial approach) pour visualiser la base de données "[Open Food Fact](https://fr.openfoodfacts.org/)".

# 2 Description générale

Nous allons utiliser les outils ci-dessous pour visualiser la base de données comme décrit dans l'article.

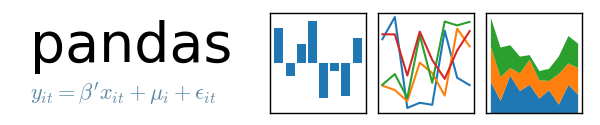
Dans ce projet, nous avons utilisé **Python** comme langage de développement.

**

*Python*

Parce que ce langage a une extension forte et dispose d'une riche bibliothèque d'outils de visualisation et de science des données.

**Pandas** en tant que bibliothèque de traitement et de nettoyage des données intègrent un grand nombre de bibliothèques et certains modèles de données standard, fournissant les outils nécessaires pour manipuler efficacement de grands ensembles de données. Pandas fournit un certain nombre de fonctions et de méthodes qui nous permettent de traiter les données rapidement et facilement.

**

*Pandas*

Nous utilisons **Matplotlib** comme notre bibliothèque pour la visualisation de données. Matplotlib est une bibliothèque de dessins 2D Python qui produit des graphiques de niveau qualité de publication dans une variété de formats de copie papier et d’environnements interactifs multi plates-formes.

Avec Matplotlib, les développeurs peuvent générer des tracés, des histogrammes, des spectres de puissance, des graphiques à barres, des tracés d'erreur, des diagrammes de dispersion, etc. avec seulement quelques lignes de code.



*Matplotlib*

Nous rencontrons des problèmes avec matplotlib, notamment:

1. Mauvaise interactivité: il ne peut y avoir d’expérience interactive fluide, telle que le zoom avant et arrière.

2. Le dessin est lent, même si vous utilisez la méthode parallèle. La vitesse est incrémentée du nombre de points.

Pour ces deux raisons, nous avons remplacé nos outils de dessin.

Nous avons utilisé l'outil **Plotly**. La bibliothèque graphique de Python de Plotly permet de créer en ligne des graphiques interactifs de qualité publication, tels que des tracés linéaires, des diagrammes de dispersion, des graphiques en aires, des graphiques à barres, des barres d'erreur, des diagrammes à barres, des histogrammes, des cartes sous-graphiques, des graphiques à axes multiples, des graphiques polaires, etc. Diagrammes à bulles.

**

*Plotly*

# 3 Analyse et conception

## 3.1. OpenFood

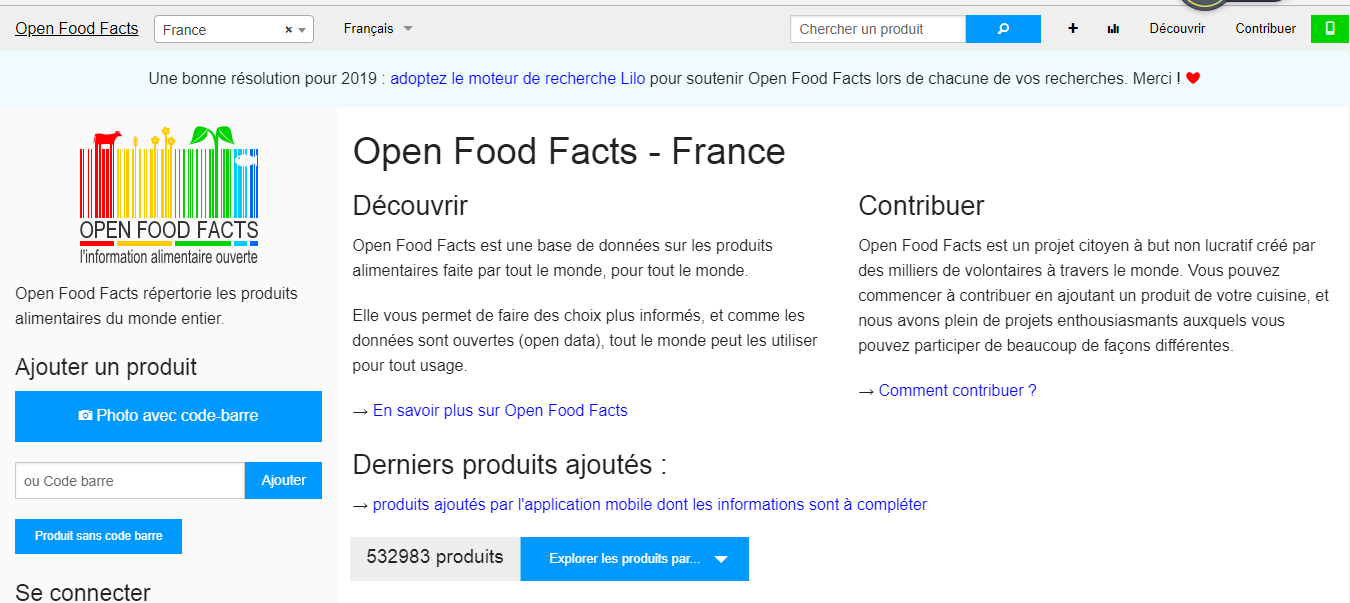
Open Food Facts est une base de données de produits alimentaires qui répertorie les ingrédients, les allergènes, la composition nutritionnelle et toutes les informations présentes sur les étiquettes des aliments.

Open Food Facts est une association à but non lucratif composée de volontaires.

Plus de 9000 contributeurs comme vous ont ajouté 600 000 produits de 200 pays en utilisant notre app Android, iPhone ou Windows Phone ou leur appareil photo pour scanner les codes barres et envoyer des photos des produits et de leurs étiquettes.

Les données sur la nourriture sont d'intérêt public et doivent être libres et ouvertes. Toute la base de données est publiée sous forme de données ouvertes (open data) qui peuvent être utilisées par tous et pour tous usages

La description spécifique des données est la suivante: <https://world.openfoodfacts.org/data/data-fields.txt>.



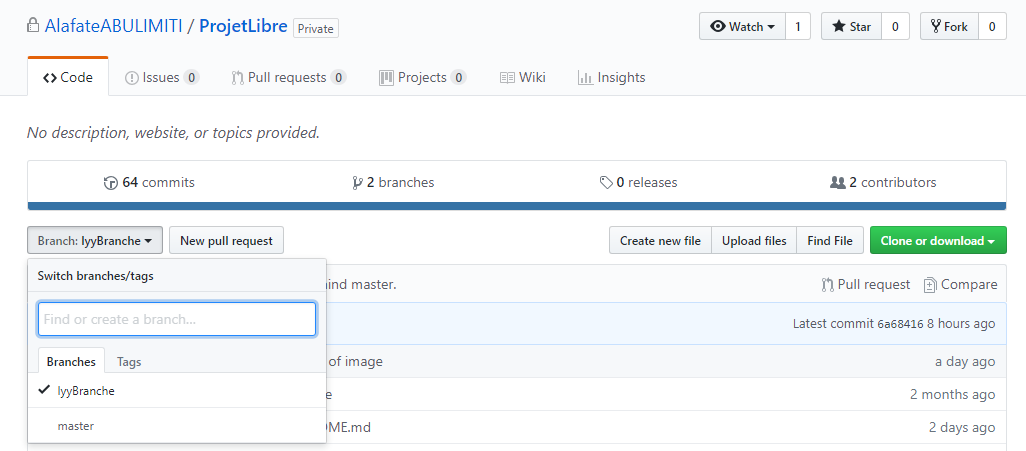
*OpenFood WebSite*

Ces données peuvent être grossièrement divisées en deux types: **types numériques et types de texte.**

## 3.2. Choix de techenique

### 3.2.1. Controleur de version du projet

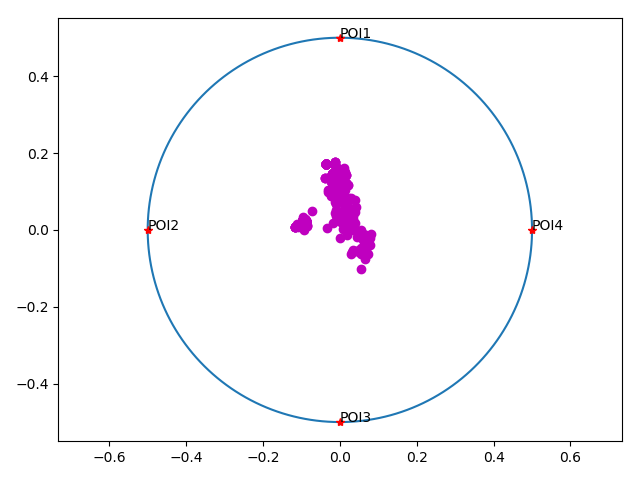
Nous gérons la version de notre projet par ‘Git’. Et pour chaque version d'implémentation, on crée une branche (2 branches totales, comme l’image ci-dessous).

**

*Git*

### 3.2.2. Etat de l’art

Notre tâche principale est de mettre en œuvre la méthode de visualisation des données dans cet article: [On visualizing large multidimensional datasets with a multi-threaded radial approach.](https://link.springer.com/article/10.1007/s10619-015-7174-1)

**

Les étapes de visualisation des données de ce document sont les suivantes:

1. Le point de données sélectionné est utilisé comme point d’intérêt.

2. Calculer la distance entre les autres points et les points d’intérêt.

3. Calculer les coordonnées du point.

4. Visualiser les données comme indiqué.

### 3.2.3. La methodé 2-Gram

Pour les données de deux types de texte, nous ne pouvons pas déduire directement leur distance directe. Nous devons d’abord vectoriser les données de texte.

Nous avons besoin d'une méthode 2-grams pour la vectorisation du texte. La méthode spécifique est la suivante:

1. Épissage des données de type texte

2. Séparer les données toutes les deux lettres

3. Compter le nombre d'occurrences de deux combinaisons de lettres

4. Placer dans un vecteur 26 \* 26.

Exemple : tu es beau

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | aa | ab | ... | au | ... | be | ... | es | ... | tu | ... | zy | zz |
| tu es beau | 0 | 0 | ... | 1 | ... | 1 |  | 1 |  | 1 |  | 0 | 0 |
| ... |  |  |  |  |  |  |  |  |  |  |  |  |  |

Dans cet exemple, nous divisons cette phrase en quatre parties: tu es beau.Pour le vecteur 26 \* 26, nous pouvons en placer 1 pour les colonnes au, be, es, tu. Ceci représente une occurrence.

# 4 Mise en oeuvre

## 4.1. Présentation de données

Le fichier (‘.csv’) de Open Food données est très grand (755252 lignes, 173 colonnes, la mémoire est plus de 2 GB). Donc, nous d’abord récupérons une partie de ces données pour visualiser et traiter facilement (aussi pour tester). Une partie de données est comme l’image ci-dessous:



openfood csv fichier (une partie - 500 lignes)

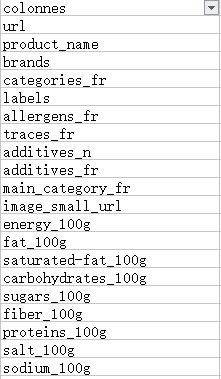
Il contient 173 colonnes et 500 lignes. c’est à dire, pour chaque produit, on peut utiliser 173 attributs pour le décrire. Parce que ce fichier contient beaucoup d'informations qui ne sont pas très importantes. En plus, certaines descriptions d'informations sont confuses et le format n'est pas uniforme, par exemple, sur la colonne ‘brand’.Et ‘Auchan’ et ‘auchan’ sont des description de la marque ‘Auchan’. En conséquence, on doit d’abord nettoyer ces données.

## 4.2. Traitement de données

### 4.2.1.Choix d’attributs

Pour visualiser les données,d’abord on doit déterminer avec quelles attributs on analyse.

Le critère de choix d’attributs: l’attribut représentatif et important, aussi, si dans une colonne, il n’y a pas beaucoup de contenus, on ne choisit pas cet attribut non plus.

**

*Choix de attributs (colonnes)*

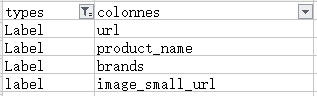
Les attributs principaux:

* url: l’url de produit.
* product\_name: le nom de produit.
* brands: la marque de produit.
* categories\_fr: la categorie de produit.
* additives: l’additif de l’alimentation.
* protein: le pourcentage de la protéine.

### 4.2.2. Prétraitement de attributs

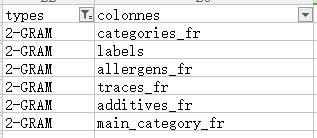
1. Nous divisons ces attribut à 3 groupes:

* **Groupe 1**: ces attributs peuvent être utilisées comme le label.

**

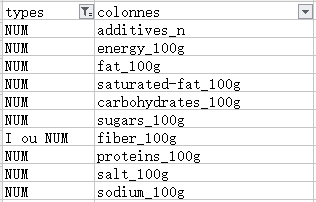
*L’attribut comme le label*

* **Groupe 2**: quand on calcule la similitude des produits, ces attributs peuvent être utilisées la méthode 2-gram à calculer.

**

*L’attribut utilisant 2-gram à calculer*

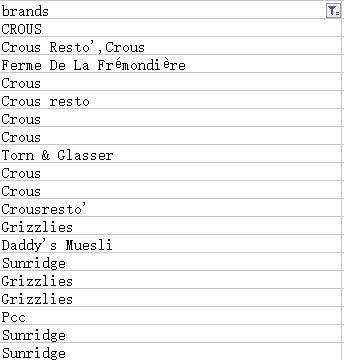
* **Groupe 3**: quand on calcule la similitude des produits, le contenu de ces attributs sont numériques, donc, nous pouvons utiliser la ‘Distance Euclidienne’ directement.

**

*L’attribut numerique*

2. Traitement de l’attribut **‘brands’**:

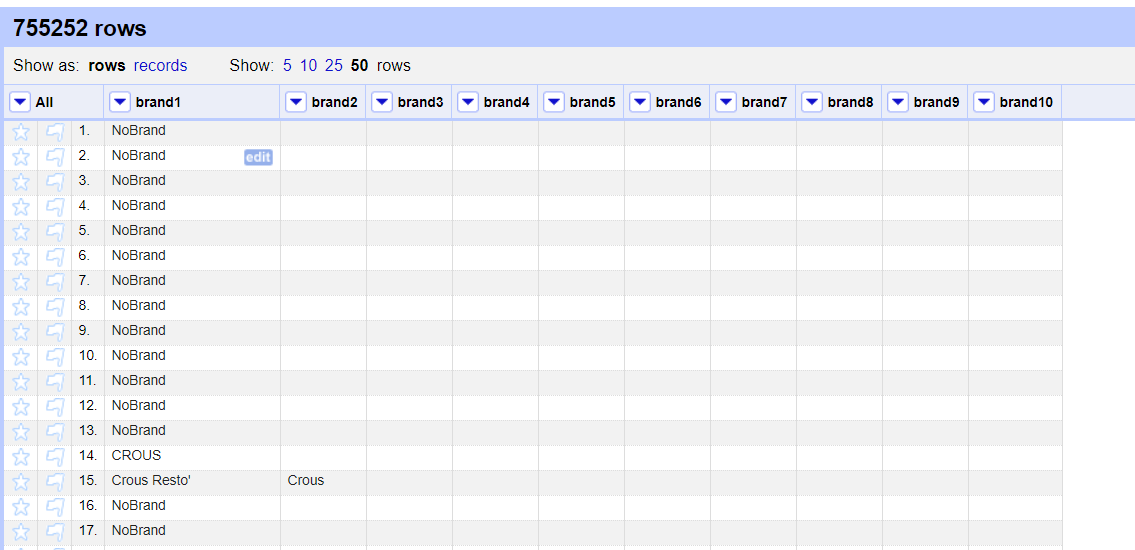
Nous observons la colonne ‘brands’, on trouve qu’il utilise plusieurs types de descriptions pour une marque, par exemple, en effet CROUS = Crous. En plus, quelques produits ont plusieurs marques labels.

**

*L’attribut ‘brand’*

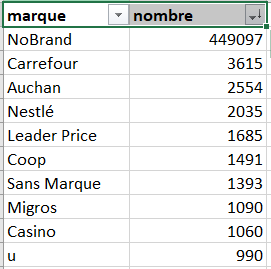
Donc, nous voulons nettoyer cette colonne. Parce que le fichier complet est très grand, on juste récupère la colonne ‘brands’.

(1). D’abord, on utilise ‘OpenRefine’ pour diviser ces marques pour chaque lignes. Aussi, on remplit toutes les ligne Null avec le mot - ‘NoBrand’.

**

*La division de ‘brands’*

(2).On juste utilise la première marque pour labelliser chaque produit. Et on veut utiliser la fonctionnalité ‘Facet’ et ‘Cluster’ de ‘OpenRefine’ pour nettoyer la colonne ‘brand1’. Mais, il y a plus de 90,000 types de descriptions, il a besoins beaucoup de mémoire pour exécuter et il a toujour crash en cours d'exécution. Donc, on utilise ‘Excel’ traitant avec ‘Top 10 des marques avec le plus de produits’.

**

*Top 10 des marques avec le plus de produits*

Après nettoyage:



*Top 10 des marques avec le plus de produits - après nettoyage*

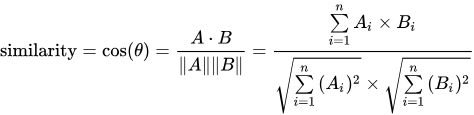
Jusqu'à cette étape, on peut commencer à calculer la similitude entre des produits.

(1). D’abord, on choisit 5 produits par la méthode ‘K-Means’. Ces 5 points sont comme Point d'intérêt, noté comme POIs

(2). Suivante, on calcule la similitude entre chaque point et POIs.

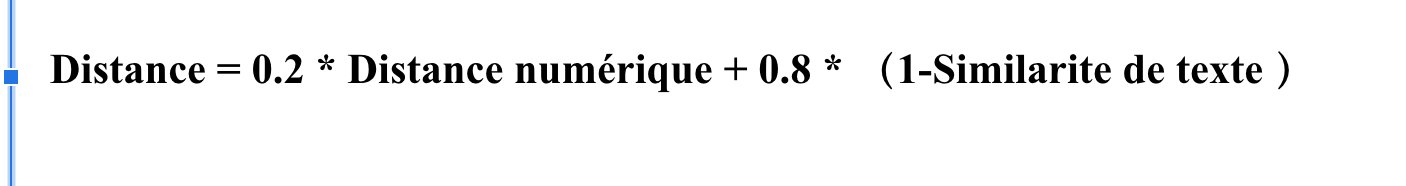
Nous allons calculer la distance entre les deux données en utilisant la distance euclidienne plus la similarité entre les mots. La formule correspondante est la suivante:





Mais ces deux valeurs ne sont pas sur une échelle et la distance euclidienne sera supérieure à la similarité. Nous devons donc le faire la pondération.

Une telle normalisation est propice à une discrimination accrue des deux données.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Le nombre total de points (à calculer la coordonnée)** | **Le nombre de processeurs** | **Le nombre de task** | **Le temps moyen pour chaque task** | **Le temps total moyen** |
| 1,000 | 4 | 4 | environs 1 minutes | environs 1 minutes |
| 10,000 | 4 | 16 | environs 2 minutes | environs 8 minutes |
| 100,000 | 4 | 16 | environs 22.5 minutes | environs 90 minutes |
| 755,252 | 3 | 64 | en virons 25 minutes | environs 8 heures |
| 40 | 160 | environs 20 minutes | environs 1 heure |

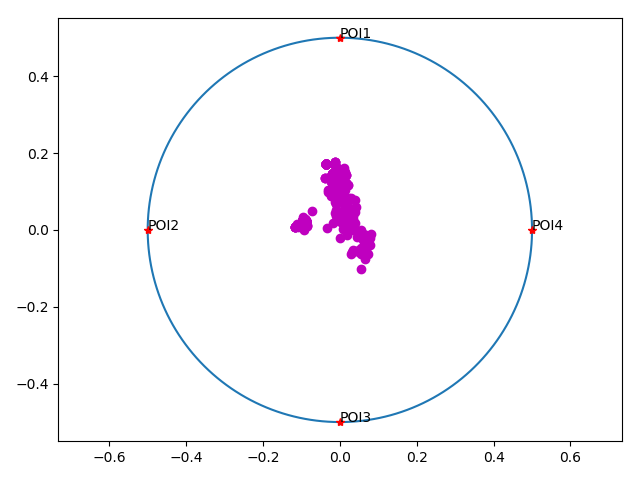
*Le temps de calculer la coordonnée*

## 4.3. Visualisation interactive

### 4.3.1. Version 1.0 - Matplotlib

Après on obtient les coordonnées pour chaque point, on peut positionner ces points sur l’image ou la figure.

Au début, nous utilisons la librairie ‘Matplotlib’ pour faire la visualisation, et on essaie avec 1000 points (4 POIs). Le résultat est comme l’image ci-dessous. Quand on clique sur un point, il affichera le label avec l’information de produit (le nom et la marque).

**

*La visualisation avec 500 points (Matplotlib)*

Quand on augmente le nombre de points, on trouve que le temps d'exécution augmente plus.

|  |  |  |  |
| --- | --- | --- | --- |
| **Le nombre total de points (à positionner)** | **Le nombre de points (à positionner) augmenté** | **Le temps augmenté pour positionner (créer la figure)** | **Le temps total pour positionner (créer la figure)** |
| 10,000 | 10000 | 2 minutes | 2 minutes |
| 20,000 | 10000 | 3 minutes | 5 minutes |
| 30,000 | 10000 | 5 minutes | 10 minutes |
| 40,000 | 10000 | 8 minutes | 18 minutes |
| …... | …... | …... | …... |
| 750,000 | 10000 | …... | plus de 1 jour |

*Le temps pour* créer *la figure- Matplotlib*

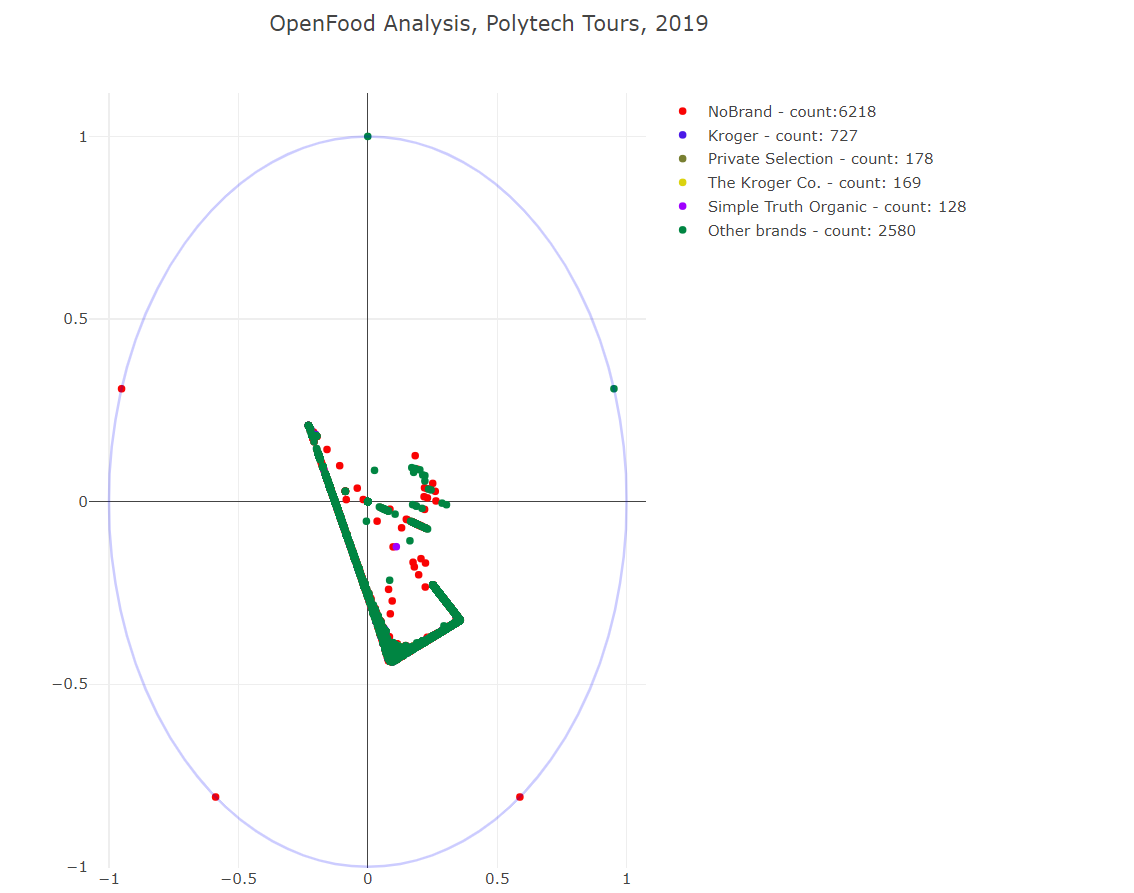
Vers ce tableau, nous pouvons observer que si on veut positionner 750,000 points sur la figure, ce prendra plus de 1 jour.

Et aussi, on teste avec tous les points (755252 points), on trouve qu’il ne peut pas afficher le résultat de la figure dans 1 heure. Donc, nous changeons la libraire pour faire la visualisation.

### 4.2.2. Version 2.0 - Plotly.Scatter

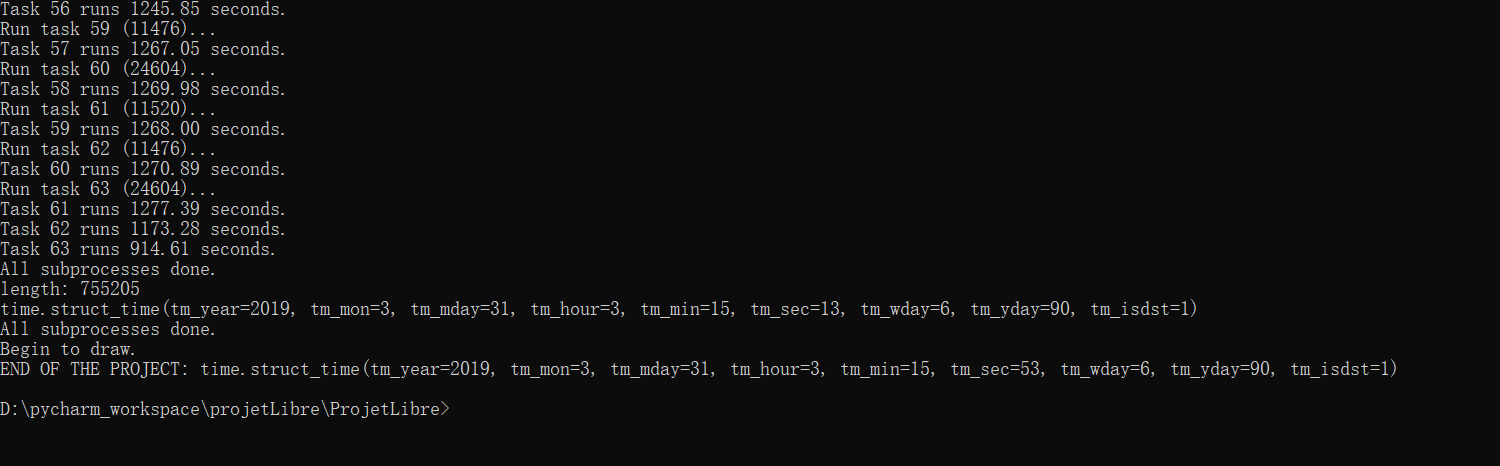
La libraire ‘Plotly’ permet de créer de figures, diagrammes, images, etc. Aussi, il a une variété de fonctionnalités interactives, par exemple: zoomer, afficher le label, afficher la légende. Et il peut sauvegarder tous les donnees de figures dans un fichier ‘.html’.

Nous ajoutons la légende avec ‘Top 5 des marques avec le plus de produits’, le type de la figure est ‘SVG’.

**

*La visualisation avec 10000 points (Plotly.Scatter)*

Et le temps d'exécution est plus court, on teste avec tous les points (755252 points), il juste prend 40 seconds (comme l’image ci-dessous).

**

*Le temps pour* créer *la visualisation - Plotly.Scatter*

Mais, on trouve l’autre problème, la charge du fichier ‘.html’ et le rendu de l’image nécessitent pleines ressources de navigateur. Donc, quand on charge le fichier ‘.html’ (le résultat de la visualisation), si le nombre de points plus grande, le temps d’affiche sera plus long.

Aussi, quand on clique sur l’image, il peut être apparaît le crash. Et on essaie avec plusieur navigateurs, le résultat est comme le tableau ci-dessous:

|  |  |  |  |
| --- | --- | --- | --- |
| **Le nombre total de points (à positionner)** | **Le navigateur** | **Le temps d’affiche** | **Crash ou non** |
| 1,000 | Chrome | moins de 5 seconds | Non |
| 10,000 | Chrome | moins de 5 seconds | Non |
| 100,000 | Chrome | plus de 5 seconds | Crash beaucoup |
| 755,252 | Chrome | ne peut pas affichier | / |
| FireFox | ne peut pas affichier | / |
| IE | ne peut pas affichier | / |
| Microsoft Edge | plus de 15 seconds | ne peut pas etre interactif |

*Le temps d’affichier la figure ‘SVG’*

Vers le tableau ci-dessous, on peut trouver qu’il ne peut pas afficher la figure dans le navigateur. Donc, nous voulons utiliser d’autre type de l’image pour afficher le résultat.

### 4.2.3. Version 3.0 - Plotly.Scattergl

Quand on lit API de la libraire ‘Plotly’, on trouve l’autre type de l’image ‘WebGL’ qui permet d’afficher millions points. Donc, on change le type de la figure, et on relance notre projet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Le nombre total de points (à positionner)** | **Le navigateur** | **Le temps d’affiche** | **Crash ou non** |
| 1,000 | Chrome | moins de 3 seconds | Non |
| 10,000 | Chrome | moins de 3 seconds | Non |
| 100,000 | Chrome | moins de 5 seconds | Non |
| 755,252 | Chrome | moins de 5 seconds | Non |

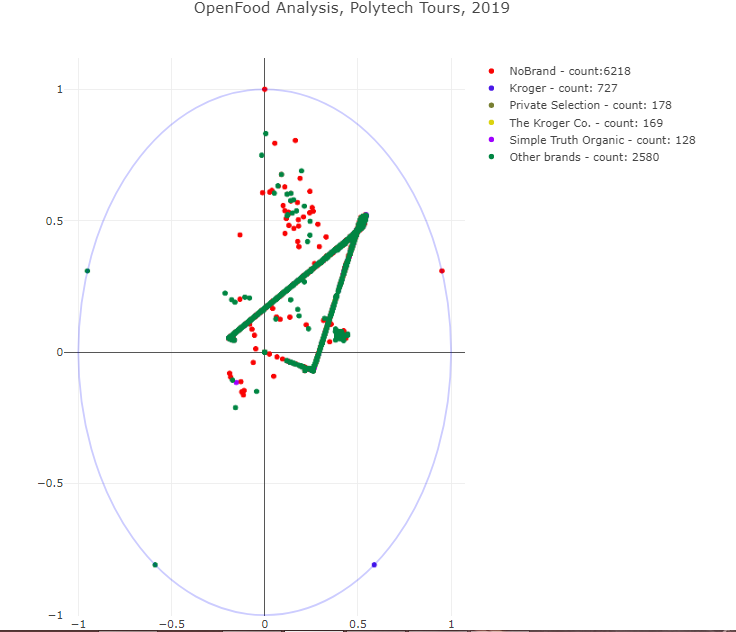
*Le temps d’affichier la figure ‘WebGL’*

## 4.4. Analyse de résultats

### 4.4.1. Visualisation de résultats

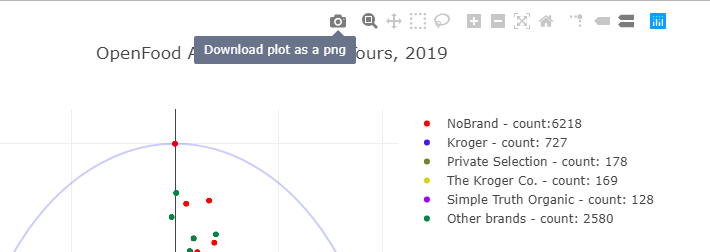
(1). Visualiser avec 10000 points

Vers l’image ci-dessous, on peut regarder la titre de la figure, la légende de la figure. Aussi, on peut savoir que dans ces 10000 points, les marques qui ont des plus produits sont: ‘Kroger’, ’Private Selection’, ‘The Kroger Co.’ et ’Simple Truth Organic’.

**

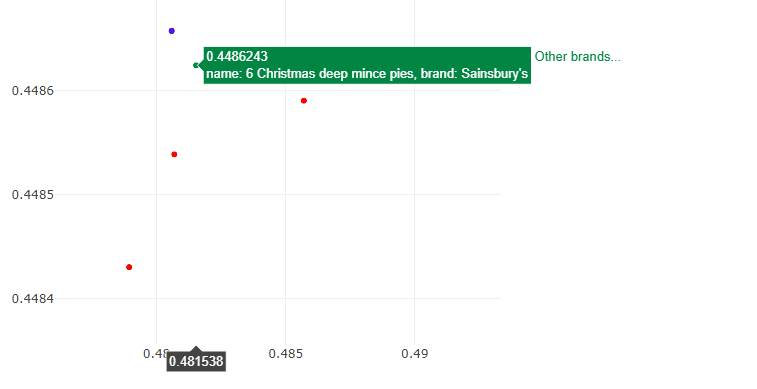
*La visualisation avec 10000 points*

Au-dessus de la figure, il y a plusieurs fonctionnalités interactives: par exemple, sauvegarder la figure comme une l’image au format ‘PNG’, zoomer + ou -,

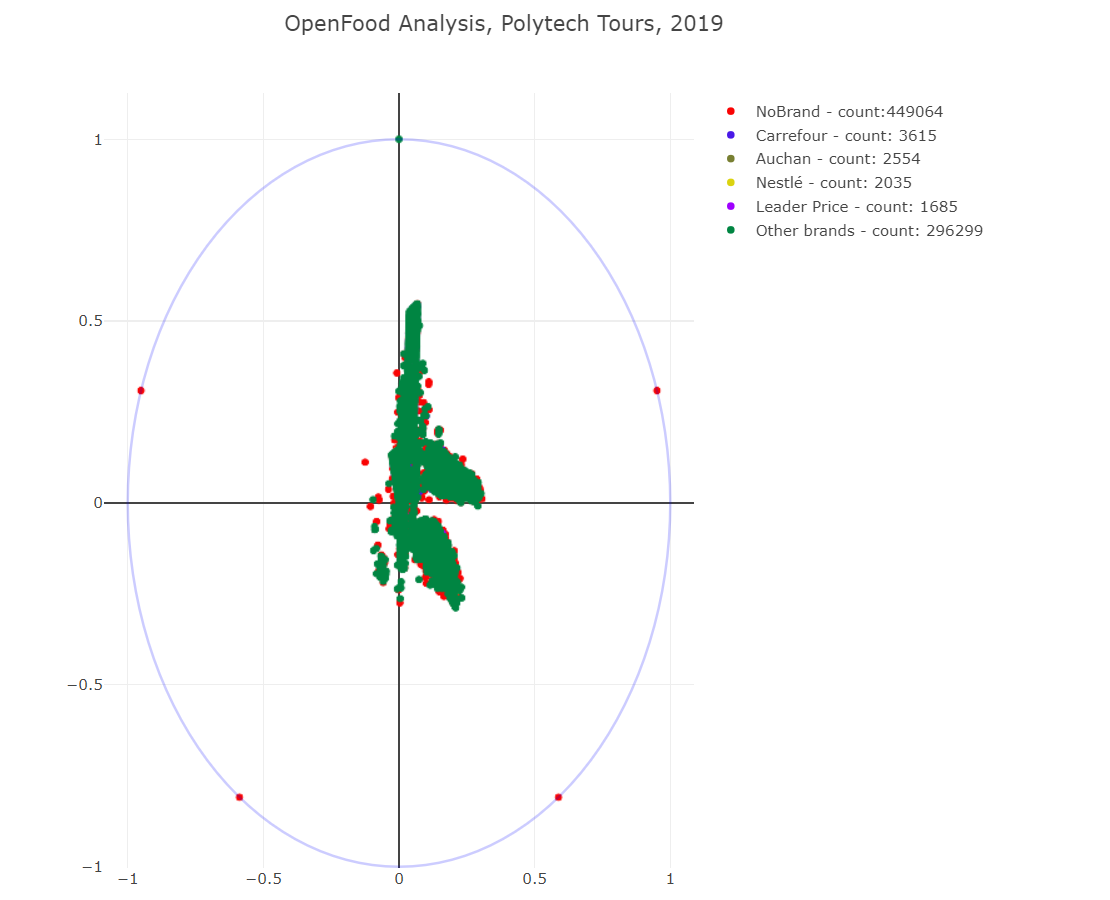
**

*Les fonctionnalites interactives*

Nous pouvons choisir quelque région pour grandir, et quand on met la souris sur un point, il affichera la coordonnée de ce point, le nom de produit et la marque de produit (comme l’image ci-dessous).

**

*Zoomer et affichier l’information d’un point*



*La visualisation avec 755,252 points*

### 4.4.2. La performance et la qualité

Le temps d’éxecution totale est comme le tableau ci-dessous:

|  |  |  |
| --- | --- | --- |
| **Le nombre de points** | **Le nombre de processeurs** | **Le temps d’éxecution total (moyen)** |
| 100 | 4 | moins de 1 minute |
| 1,000 | 4 | moins de 5 minutes |
| 10,000 | 4 | 9 minutes |
| 100,000 | 3 | 90 minutes |
| 755,252 | 3 | 8 heures |
| 40 | environs 1 heure |

*Le tableau de temps d’execution total*

# 5 Bilan et conclusion

## 5.1. Faits

Pendant notre projet libre, notre faits sont comme ci-dessous:

* Récuperer et prétraiter les donnees de websit ‘OpenFood’.
* Lire et etudier le document de la visualisation.
* Calculer la distance (la similitude) entre des produits (multi processus, 755,252 produits).
* Réaliser la visualisation interactive (WebGl, 755,252 points).
* Ecrire le rapport.
* Preparer tous les document et code pour la soutenance.

## 5.2.Reste à faire

Parce que nous juste formalisons les description de quelques marques. Donc, pour plus loin, nous pouvons formaliser et nettoyer toutes les descriptions de marques.

## 5.3. Bilan auto-critique

Pendant le projet libre, nous avons revisé et étudié des connaissances et des méthodes des traitement de données, par exemple, la méthode ‘2-Gram’. Aussi, on a révisé et étudié des outils pour traiter ou nettoyer les données, par exemple, ‘Excel’, ‘OpenRefine’.

En plus, nous avons étudié les libraires pour visualiser les données, par exemple, la librairie ‘Matplotlib’, la libraire ‘Plotly’.

Vers ce projet, nous avons aussi étudié qu’il important de chercher et choisir les techniques pour programme. Par exemple, bien que la figure ‘SVG’ satisfait 10000 points, il ne peut pas bien être chargé avec 10000 points ou plus.

Bien que nous rencontrons plusieurs problèmes, par exemple, les méthode de traitement de données, la machine pour exécuter le projet. Avec l’aide de M.Gilles, ces problèmes sont résolues. Nous voulons remercier de l’aide de M.Gilles.