2023 - APPASSAMY, CRAPPIER, GUILLAUD, LAMBERT

SAÉ Parcours C : Data

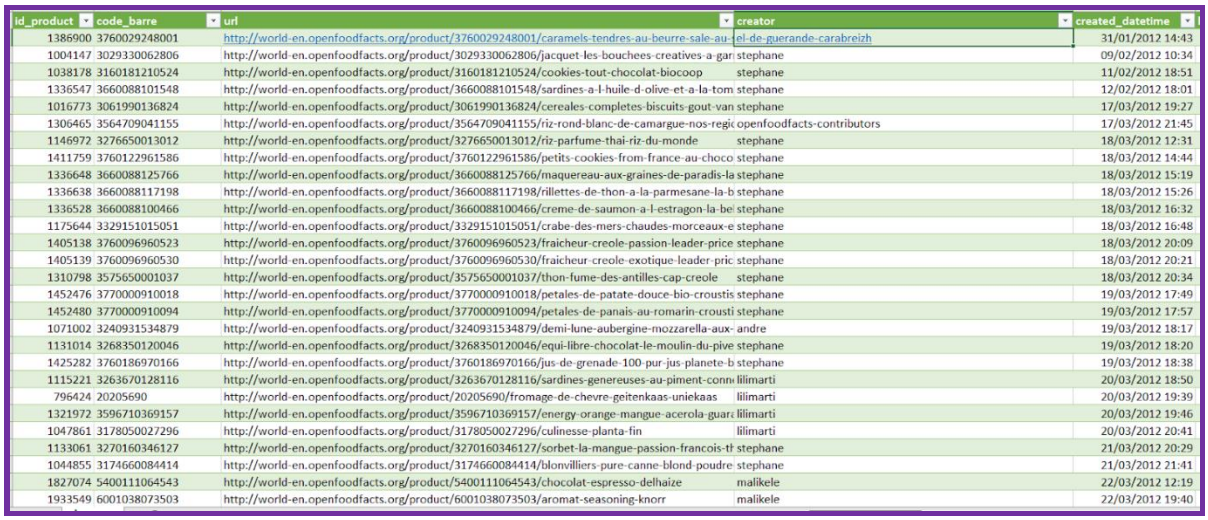
Développement avec une base de données et visualisation

Table des matières

Architecture et Base OLTP

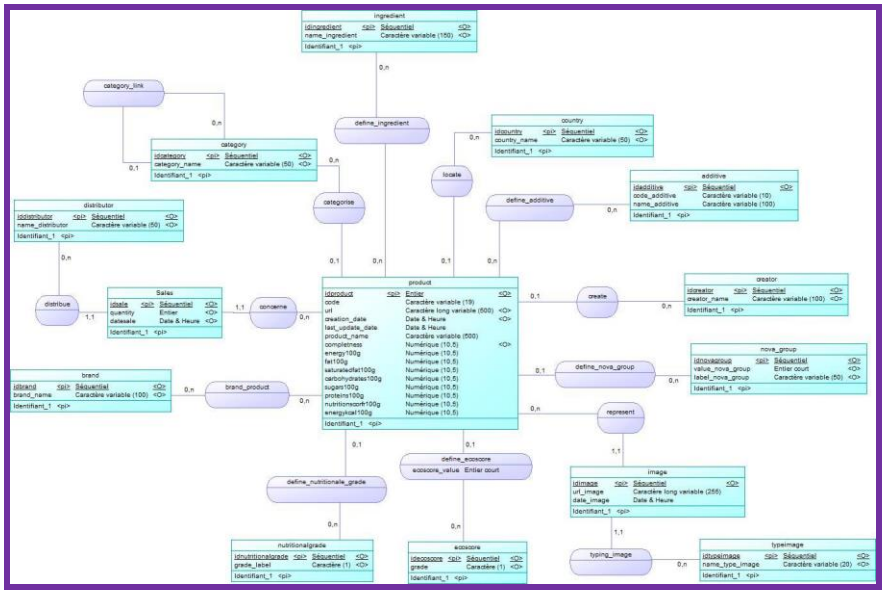
Conception

Pour la conception de la base de données, nous sommes partis du fichier .csv fourni. Nous l'avons ouvert avec l'outil l'Excel puis nous l'avons mis au propre en séparant chaque attributs dans une colonne. Nous avons également traité les données sur l'outil Power Query telles que : attribuer les bons types, renommer les colonnes, ajouter des colonnes conditionnelles, mettre les valeurs numériques au bon format etc.



id_product	code_barre	url	creator	created_datetime
1386900	3760029248001	http://world-en.openfoodfacts.org/product/3760029248001/caramels-tendres-au-beurre-sale-au-	el-de-guerande-carabreizh	31/01/2012 14:43
1004147	3029330062806	http://world-en.openfoodfacts.org/product/3029330062806/jacquet-les-bouchees-creatives-a-gar	stephane	09/02/2012 10:34
1038178	3160181210524	http://world-en.openfoodfacts.org/product/3160181210524/cookies-tout-chocolat-biocoop	stephane	11/02/2012 18:51
1336547	3660088101548	http://world-en.openfoodfacts.org/product/3660088101548/sardines-a-l-huile-d-olive-et-a-la-tom	stephane	12/02/2012 18:01
1016773	3061990136824	http://world-en.openfoodfacts.org/product/3061990136824/cereales-completes-biscuits-gout-van	stephane	17/03/2012 19:27
1306465	3564709041155	http://world-en.openfoodfacts.org/product/3564709041155/riz-rond-blanc-de-camargue-nos-regi	openfoodfacts-contributors	17/03/2012 21:45
1146972	3276650013012	http://world-en.openfoodfacts.org/product/3276650013012/riz-parfume-thai-riz-du-monde	stephane	18/03/2012 12:31
1411759	3760122961586	http://world-en.openfoodfacts.org/product/3760122961586/petits-cookies-from-france-au-choco	stephane	18/03/2012 14:44
1336648	3660088125766	http://world-en.openfoodfacts.org/product/3660088125766/maquereau-aux-graines-de-paradis-la	stephane	18/03/2012 15:19
1336638	3660088117198	http://world-en.openfoodfacts.org/product/3660088117198/rillettes-de-thon-a-la-parmesane-la-b	stephane	18/03/2012 15:26
1336528	3660088100466	http://world-en.openfoodfacts.org/product/3660088100466/creme-de-saumon-a-l-estragon-la-be	stephane	18/03/2012 16:32
1175644	3329151015051	http://world-en.openfoodfacts.org/product/3329151015051/crabe-des-mers-chaudes-morceaux-e	stephane	18/03/2012 16:48
1405138	3760096960523	http://world-en.openfoodfacts.org/product/3760096960523/fraicheur-creole-passion-leader-price	stephane	18/03/2012 20:09
1405139	3760096960530	http://world-en.openfoodfacts.org/product/3760096960530/fraicheur-creole-exotique-leader-pric	stephane	18/03/2012 20:21
1310798	3575650001037	http://world-en.openfoodfacts.org/product/3575650001037/thon-fume-des-antilles-cap-creole	stephane	18/03/2012 20:34
1452476	3770000910018	http://world-en.openfoodfacts.org/product/3770000910018/petales-de-patate-douce-bio-croustis	stephane	19/03/2012 17:49
1452480	3770000910094	http://world-en.openfoodfacts.org/product/3770000910094/petales-de-panais-au-romarin-crousti	stephane	19/03/2012 17:57
1071002	3240931534879	http://world-en.openfoodfacts.org/product/3240931534879/demi-lune-aubergine-mozzarella-aux-	andre	19/03/2012 18:17
1131014	3268350120046	http://world-en.openfoodfacts.org/product/3268350120046/equi-libre-chocolat-le-moulin-du-pive	stephane	19/03/2012 18:20
1425282	3760186970166	http://world-en.openfoodfacts.org/product/3760186970166/jus-de-grenade-100-pur-jus-planete-b	stephane	19/03/2012 18:38
1115221	3263670128116	http://world-en.openfoodfacts.org/product/3263670128116/sardines-genereuses-au-piment-conn	lilimarti	20/03/2012 18:50
796424	20205690	http://world-en.openfoodfacts.org/product/20205690/fromage-de-chevre-geitenkaas-uniekaas	lilimarti	20/03/2012 19:39
1321972	3596710369157	http://world-en.openfoodfacts.org/product/3596710369157/energy-orange-mangue-acerola-guar	lilimarti	20/03/2012 19:46
1047861	3178050027296	http://world-en.openfoodfacts.org/product/3178050027296/culinesse-planta-fin	lilimarti	20/03/2012 20:41
1133061	3270160346127	http://world-en.openfoodfacts.org/product/3270160346127/sorbet-la-mangue-passion-francois-th	stephane	21/03/2012 20:29
1044855	3174660084414	http://world-en.openfoodfacts.org/product/3174660084414/blonvilliers-pure-canne-blond-poudre	stephane	21/03/2012 21:41
1827074	5400111064543	http://world-en.openfoodfacts.org/product/5400111064543/chocolat-espresso-delhaize	malikele	22/03/2012 12:19
1933549	6001038073503	http://world-en.openfoodfacts.org/product/6001038073503/aromat-seasoning-knorr	malikele	22/03/2012 19:40

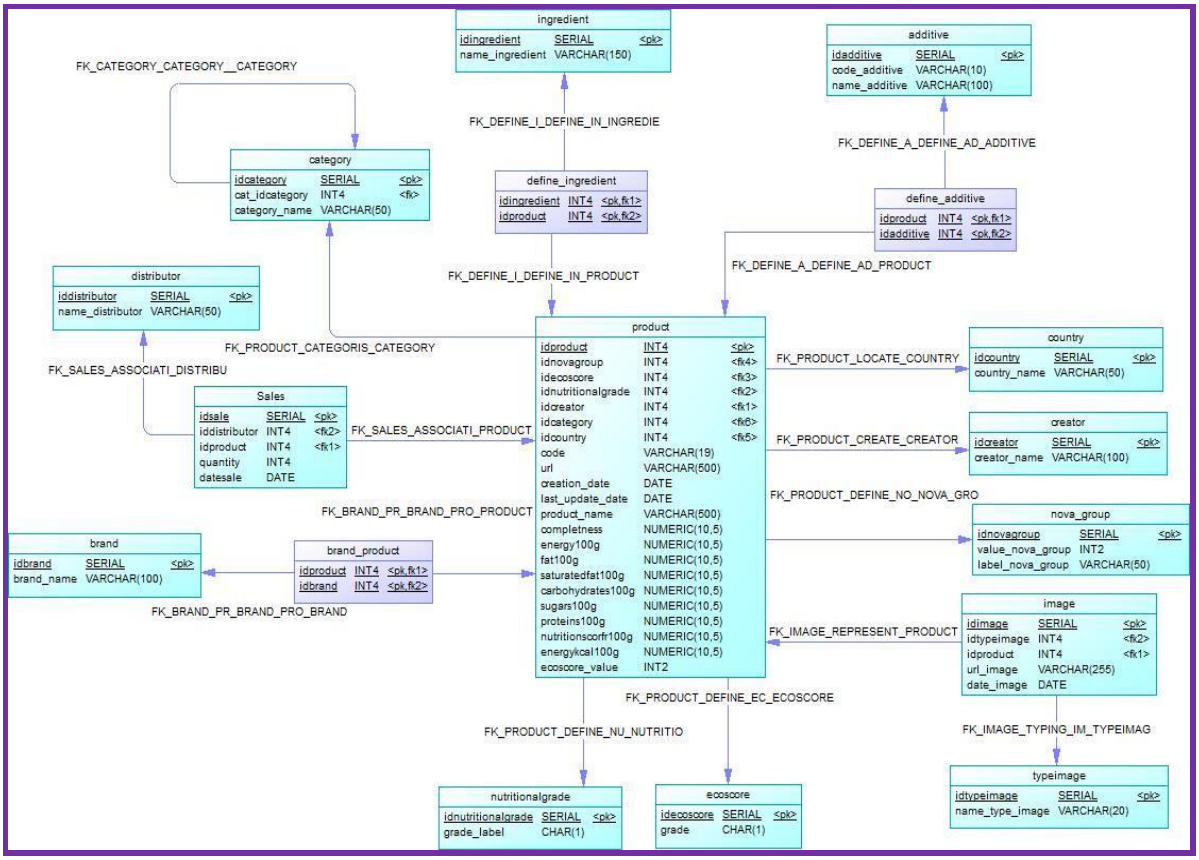
Une fois traité, nous avons toutes les informations nécessaires pour la conception de la nouvelle base de données. Ainsi nous avons établi un Modèle Conceptuel de Données (MCD) sur Power AMC. A la différence de la source de données d'origine constituée d'une table unique, nous avons plusieurs tables afin d'avoir une bonne architecture et faciliter la gestion de données. Pour répondre au besoin de la gestion des ventes, nous avons ajouté une partie dédiée dans le modèle, la table « sales ».



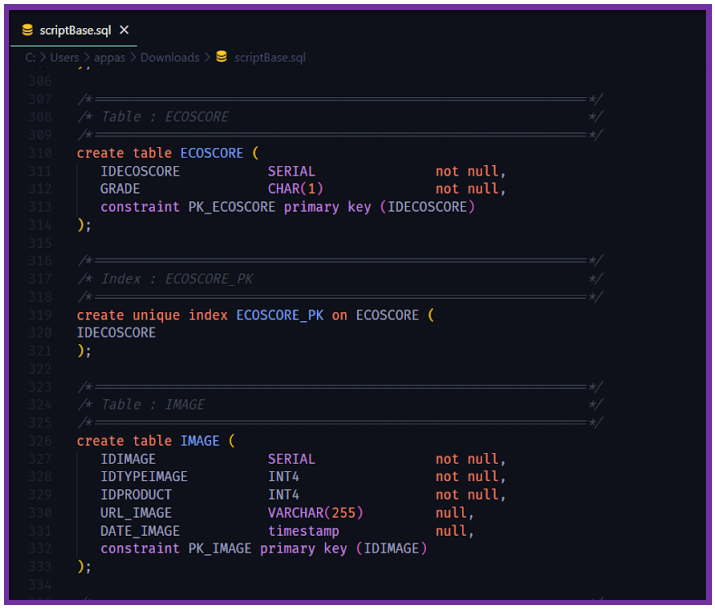
Capture d'écran :

Modèle Conceptuel de Données

Le MCD nous a permis par la suite de générer le Modèle Physique de Données (MDP), puis générer le script de création de la base de données. La base est optimisée grâce à la présence d'index sur les clés primaires des différentes tables, mais également sur les champs dont la recherche est fréquente.



Capture d'écran : Modèle Physique de Données



Capture d'écran : Extrait du script de création

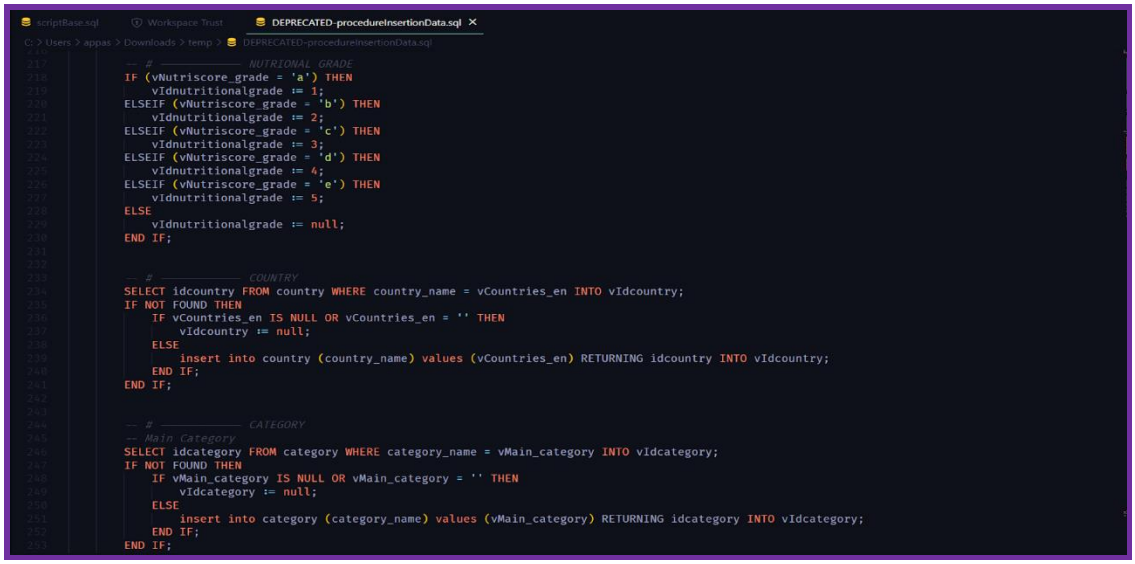
Insertion de données

Bien que le fichier Excel contient toutes les données, il est indispensable d'insérer ces milliers de lignes dans la base de données. Au cours de l'avancée du projet, nous avons procédé de plusieurs manières afin d'insérer les données. Ci-dessous le détail des différentes étapes jusqu'à la solution optimale que nous avons retenue.

I) Procédure Pl/pgSQL

Au départ, la solution la plus évidente était de créer une procédure stockée PL/PgSQL. Le principe est le suivant :

1. Exécuter le script créé afin de créer les tables du MPD
2. Créer une table temporaire ayant la même architecture que le fichier Excel
3. Convertir le fichier Excel en fichier csv
4. Exécuter la commande COPY afin d'insérer les données contenus dans le csv dans la table temporaire.
5. Exécuter la procédure qui parcourt chaque ligne de la table temporaire afin de les insérer dans les bonnes tables.



Capture d'écran : Extrait de la procédure d'insertion

L'inconvénient de cette pratique est le fait qu'il faut parcourir plus de 600 000 lignes, et pour chaque lignes faire une multitude de test (supprimer les doublons, ajout des étrangères correspondantes, …). Ces nombreux tests entraînent un temps d'exécution très important, de plus de 16 heures.

Ce rajoute à cela une procédure à part entière pour la création et insertion des jeux de données pour les ventes. Sachant que chaque produit à 100 ventes, il faut donc générer plus de 60 millions de lignes au total. Le temps d'exécution s'élevait à une vingtaine de minutes, ce qui est semble correct. Cependant le résultat ne nous était pas convenable, car cela ne générait pas de ventes réalistes, et donc en conséquence de mauvaises analyses de données par la suite. Nous avons définis que pour que les ventes soient réalistes il est nécessaire d'avoir une cohérence avec les nutriscores et les ecoscores. Typiquement un ecoscore et un nutriscore "a" aura tendance à être de plus en plus vendu au cours du temps. A l'inverse un produit correspondant à un nutriscore et un ecoscore "e" doit avoir tendance à être de moins en moins vendu car nous considérons que les distributeurs et les clients font de plus en plus attention à leur consommation.

Pour combler ces deux problèmes, à savoir un temps d'exécution rapide sur n'importe quelle machine et la génération des données de ventes réalistes, nous avons changé notre outil de travail et passé au langage informatique Python.

II) Python

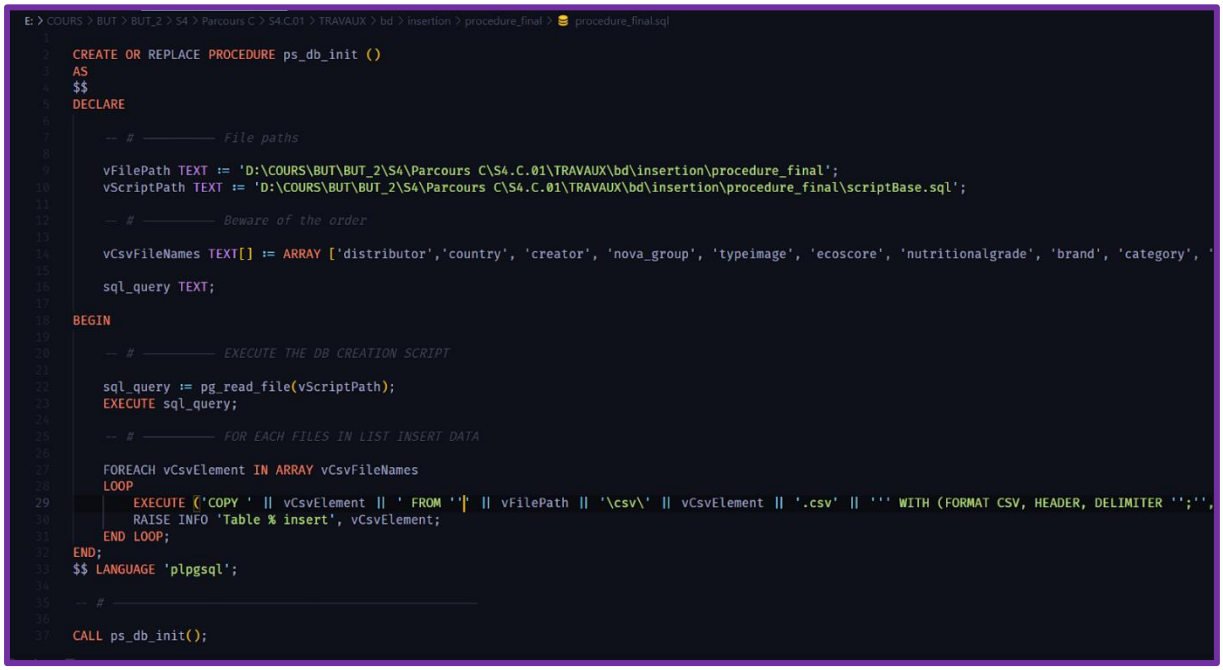
L'objectif de notre script Python est de créer l'ensemble des tables de notre base de données au format CSV. Nous pourrons ensuite facilement les insérer dans notre base de données Azure grâce aux commandes copy.

Nous avons principalement utilisé la bibliothèque Pandas. Celle-ci est très utilisée dans le monde de la data science et est incontournable. Elle est très performante, permet notamment la lecture et l'enregistrement de fichiers csv. La manipulation de dataframe (tableaux de données à deux dimensions) avec différentes fonctions.

Nous avons également optimisé le script python en passant d'un temps d'exécution de plus de 50 heures à moins de 8 minutes. Pour ce faire nous avons utilisé les fonctions pandas adapté et utilisé la bibliothèque multiprocessing pour la partie vente. Grâce à ça le script python utilise toute la puissance disponible de la machine.

Le code est disponible dans les rendues, il faut disposer dans un dossier input le fichier csv plat d'origine, créer un fichier output pour pouvoir enregistrer les fichiers csv et positionner le code python à la racine.

Une fois que le fichier csv principale a été séparé en plusieurs csv, soit un fichier par table (17 fichiers csv), nous avons créé la procédure finale qui insère toutes les données dans leurs tables correspondant grâce à la commande COPY.

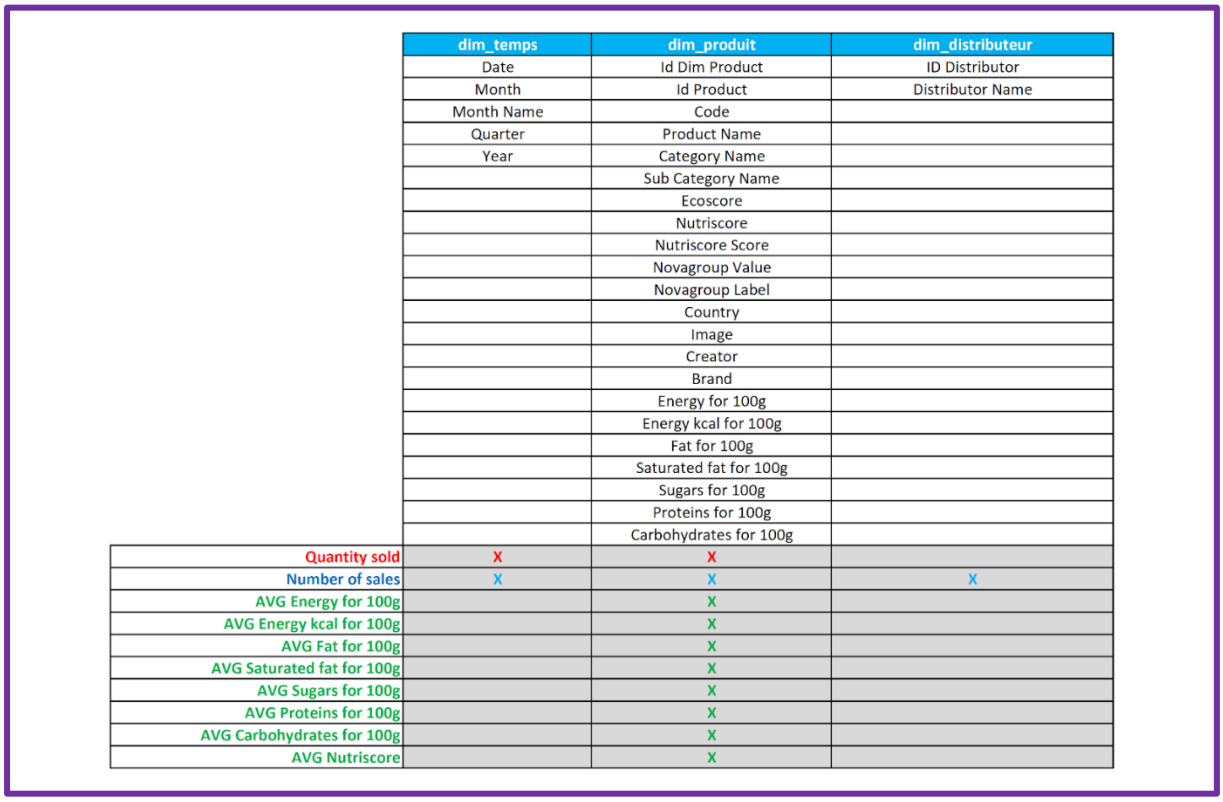


Capture d'écran : Procédure de création et d'insertion

Modèle en étoile

Tableau d'analyse

Pour commencer, nous avons étudié les différentes mesures dont nous aurons besoin pour les rapports analytiques PowerBi. Pour cela nous avons élaboré un tableau d'analyse regroupant les mesures ainsi que les différentes dimensions dont nous avons besoin.



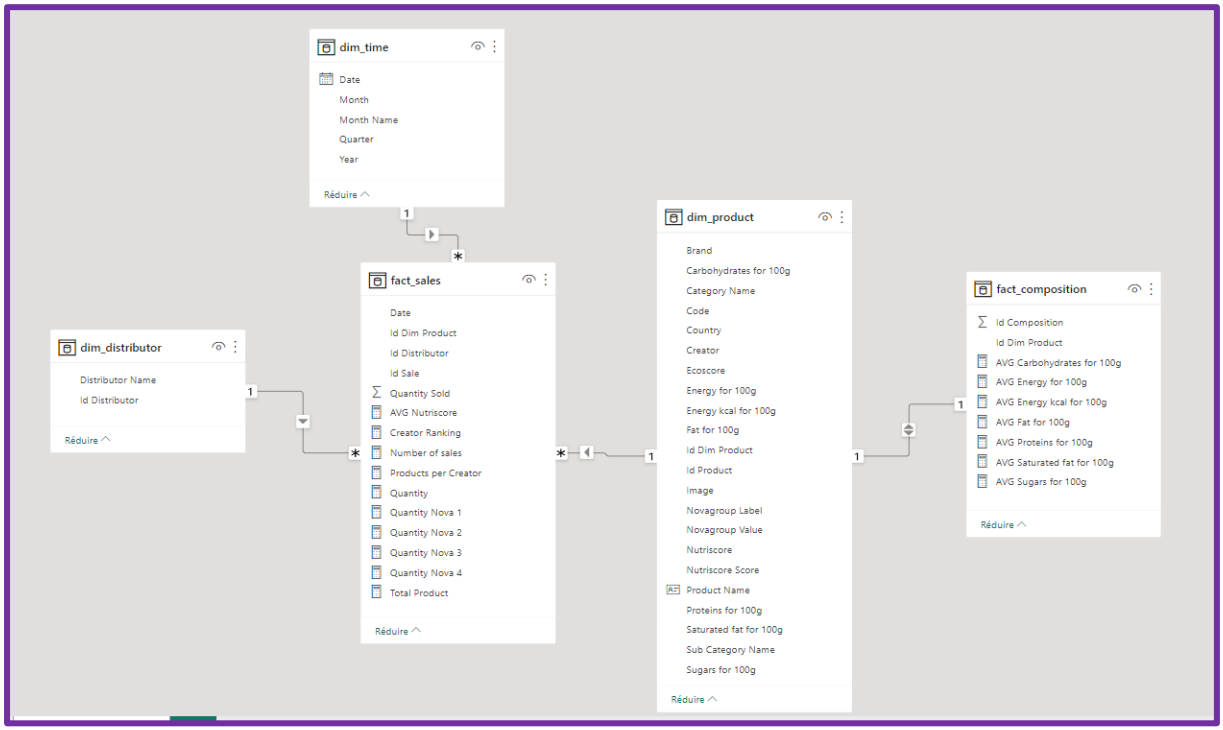
	dim_temps	dim_produit	dim_distributeur
	Date	Id Dim Product	ID Distributor
	Month	Id Product	Distributor Name
	Month Name	Code	
	Quarter	Product Name	
	Year	Category Name	
		Sub Category Name	
		Ecoscore	
		Nutriscore	
		Nutriscore Score	
		Novagroup Value	
		Novagroup Label	
		Country	
		Image	
		Creator	
		Brand	
		Energy for 100g	
		Energy kcal for 100g	
		Fat for 100g	
		Saturated fat for 100g	
		Sugars for 100g	
		Proteins for 100g	
		Carbohydrates for 100g	
Quantity sold	X	X	
Number of sales	X	X	X
AVG Energy for 100g		X	
AVG Energy kcal for 100g		X	
AVG Fat for 100g		X	
AVG Saturated fat for 100g		X	
AVG Sugars for 100g		X	
AVG Proteins for 100g		X	
AVG Carbohydrates for 100g		X	
AVG Nutriscore		X	

Capture d'écran : Tableau d'analyse

Grâce à ce dernier nous avons 3 dimensions : product, time et distributor et 2 tables de faits : sales et composition. En ce qui concerne les mesures, nous avons décidé d'étudier les ventes par rapport aux quantités de produits vendues, le nombre de ventes, ainsi que des valeurs moyennes sur la composition des différents produits.

Modèle en étoile

Le tableau d'analyse précédent nous ramènes au modèle en étoile suivant :



Capture d'écran : Modèle en étoile

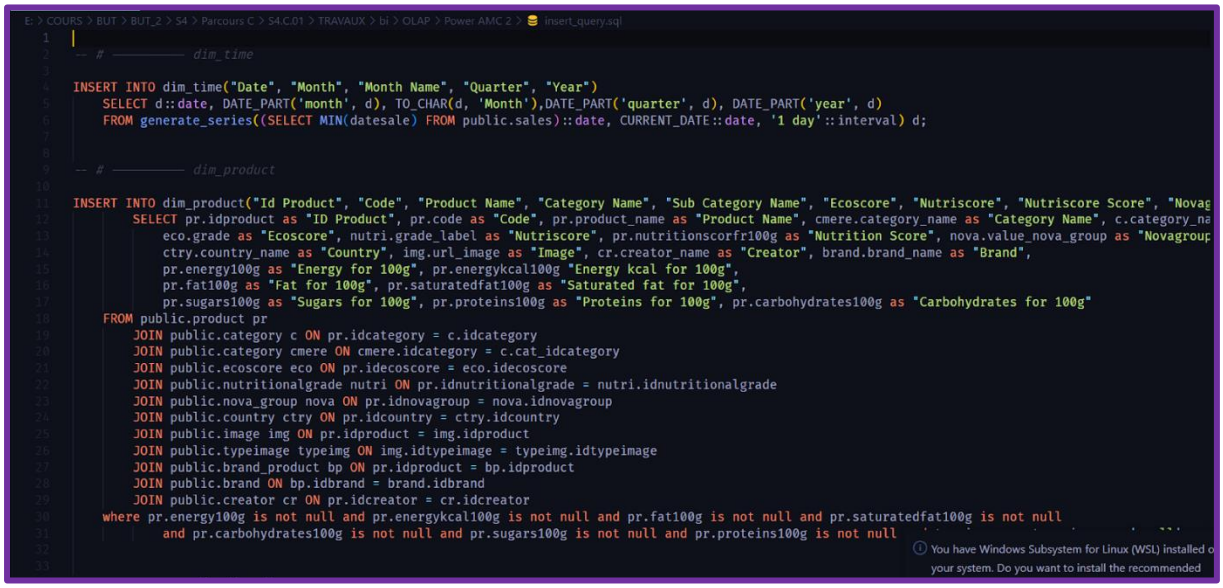
Architecture

Pour ce qui est de l'architecture de base, nous avons opté pour l'architecture OLAP. En effet, le modèle OLAP est le plus adapté car nous avons une grande quantité de données à analyser. Dans le cas du choix de l'architecture OLTP, il faut faire des requêtes pour créer des vues et ensuite connecter ces derniers à Powebi.

En revanche pour mettre en place l'architecture OLAP nous avons créé un nouveau schéma dans la base de données nommé "olap". Une fois créé nous avons ajouté à ce schéma, les tables nécessaires, autrement dit les tables de faits et de dimensions issues du modèle en étoile.

Afin d'alimenter ces nouvelles tables avec les nombreuses données existantes, on a exécuté des requêtes SQL qui prennent les données des tables du schéma public. Nous avons utilisé des jointures de type « inner » afin de ne pas avoir des clés étrangères vides. Nous avons aussi ajouté des conditions dans la clause where, afin d'avoir des compositions non vides. En effet, il est important d'avoir un jeu de données dans lequel les données ne sont pas vides, tout

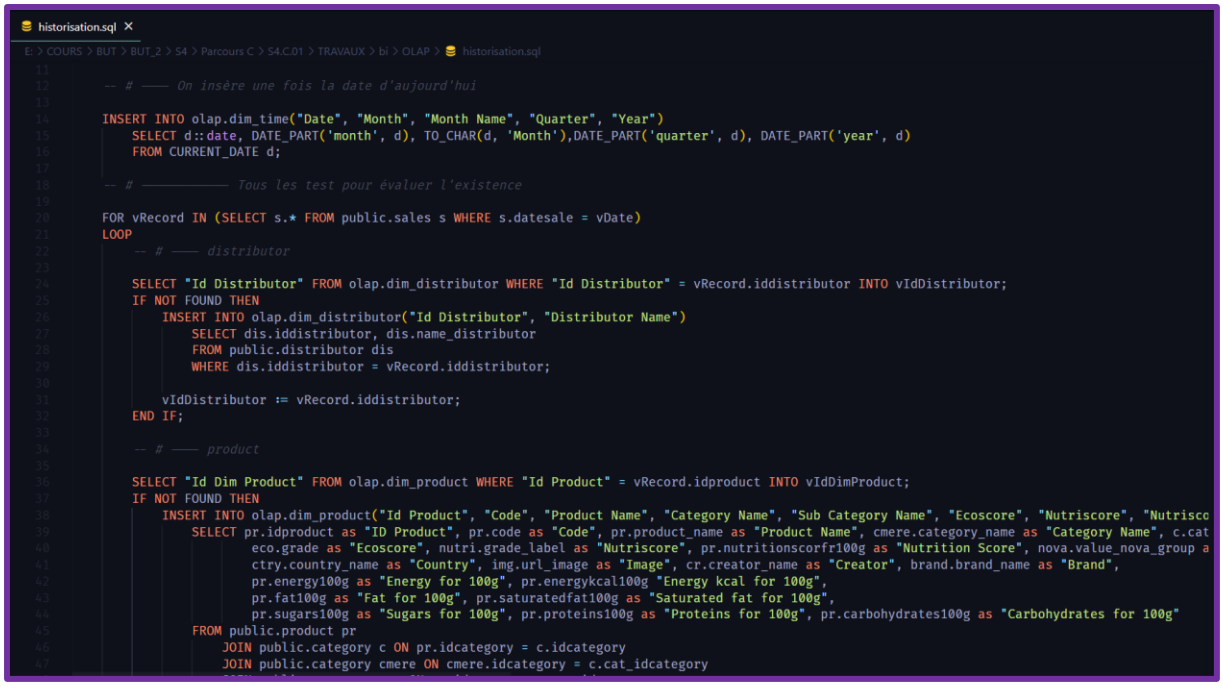
simplement pour ne pas fausser les résultats. Ce traitement a été assuré directement lors de l'insertion des données afin de ne pas traiter les données sur Power Query, en conséquences des chargement rapide et fichier moins lourd.



```sql
-- # ———————— dim_time

INSERT INTO dim_time("Date", "Month", "Month Name", "Quarter", "Year")
    SELECT d::date, DATE_PART('month', d), TO_CHAR(d, 'Month'),DATE_PART('quarter', d), DATE_PART('year', d)
    FROM generate_series((SELECT MIN(datesale) FROM public.sales)::date, CURRENT_DATE::date, '1 day'::interval) d;


-- # ———————— dim_product

INSERT INTO dim_product("Id Product", "Code", "Product Name", "Category Name", "Sub Category Name", "Ecoscore", "Nutriscore", "Nutriscore Score", "Novag
        SELECT pr.idproduct as "ID Product", pr.code as "Code", pr.product_name as "Product Name", cmere.category_name as "Category Name", c.category_na
            eco.grade as "Ecoscore", nutri.grade_label as "Nutriscore", pr.nutritionscorfr100g as "Nutrition Score", nova.value_nova_group as "Novagroup
            ctry.country_name as "Country", img.url_image as "Image", cr.creator_name as "Creator", brand.brand_name as "Brand",
            pr.energy100g as "Energy for 100g", pr.energykcal100g "Energy kcal for 100g",
            pr.fat100g as "Fat for 100g", pr.saturatedfat100g as "Saturated fat for 100g",
            pr.sugars100g as "Sugars for 100g", pr.proteins100g as "Proteins for 100g", pr.carbohydrates100g as "Carbohydrates for 100g"
    FROM public.product pr
        JOIN public.category c ON pr.idcategory = c.idcategory
        JOIN public.category cmere ON cmere.idcategory = c.cat_idcategory
        JOIN public.ecoscore eco ON pr.idecoscore = eco.idecoscore
        JOIN public.nutritionalgrade nutri ON pr.idnutritionalgrade = nutri.idnutritionalgrade
        JOIN public.nova_group nova ON pr.idnovagroup = nova.idnovagroup
        JOIN public.country ctry ON pr.idcountry = ctry.idcountry
        JOIN public.image img ON pr.idproduct = img.idproduct
        JOIN public.typeimage typeimg ON img.idtypeimage = typeimg.idtypeimage
        JOIN public.brand_product bp ON pr.idproduct = bp.idproduct
        JOIN public.brand ON bp.idbrand = brand.idbrand
        JOIN public.creator cr ON pr.idcreator = cr.idcreator
    where pr.energy100g is not null and pr.energykcal100g is not null and pr.fat100g is not null and pr.saturatedfat100g is not null
        and pr.carbohydrates100g is not null and pr.sugars100g is not null and pr.proteins100g is not null
```

Capture d'écran : Extrait des requêtes d'insertion

Cependant nous ne pouvons pas exécuter ces requêtes à chaque fois que l'on souhaite mettre à jour les modèles OLAP. En effet, il faudrait automatiser une procédure qui alimente tous les soirs le modèle OLAP avec les nouvelles données ajoutées. L'execution de la procédure peut être automatisé grâce à des langages systèmes, telle que le bash par exemple.



```sql
-- # ——— On insère une fois la date d'aujourd'hui

INSERT INTO olap.dim_time("Date", "Month", "Month Name", "Quarter", "Year")
    SELECT d::date, DATE_PART('month', d), TO_CHAR(d, 'Month'),DATE_PART('quarter', d), DATE_PART('year', d)
    FROM CURRENT_DATE d;

-- # ———————— Tous les test pour évaluer l'existence

FOR vRecord IN (SELECT s.* FROM public.sales s WHERE s.datesale = vDate)
LOOP
    -- # ——— distributor

    SELECT "Id Distributor" FROM olap.dim_distributor WHERE "Id Distributor" = vRecord.iddistributor INTO vIdDistributor;
    IF NOT FOUND THEN
        INSERT INTO olap.dim_distributor("Id Distributor", "Distributor Name")
            SELECT dis.iddistributor, dis.name_distributor
            FROM public.distributor dis
            WHERE dis.iddistributor = vRecord.iddistributor;

        vIdDistributor := vRecord.iddistributor;
    END IF;

    -- # ——— product

    SELECT "Id Dim Product" FROM olap.dim_product WHERE "Id Product" = vRecord.idproduct INTO vIdDimProduct;
    IF NOT FOUND THEN
        INSERT INTO olap.dim_product("Id Product", "Code", "Product Name", "Category Name", "Sub Category Name", "Ecoscore", "Nutriscore", "Nutrisco
            SELECT pr.idproduct as "ID Product", pr.code as "Code", pr.product_name as "Product Name", cmere.category_name as "Category Name", c.cat
                    eco.grade as "Ecoscore", nutri.grade_label as "Nutriscore", pr.nutritionscorfr100g as "Nutrition Score", nova.value_nova_group a
                    ctry.country_name as "Country", img.url_image as "Image", cr.creator_name as "Creator", brand.brand_name as "Brand",
                    pr.energy100g as "Energy for 100g", pr.energykcal100g "Energy kcal for 100g",
                    pr.fat100g as "Fat for 100g", pr.saturatedfat100g as "Saturated fat for 100g",
                    pr.sugars100g as "Sugars for 100g", pr.proteins100g as "Proteins for 100g", pr.carbohydrates100g as "Carbohydrates for 100g"
            FROM public.product pr
                JOIN public.category c ON pr.idcategory = c.idcategory
                JOIN public.category cmere ON cmere.idcategory = c.cat_idcategory
```

Capture d'écran : Extrait de la procédure d'historisation

Modèle IA Knime

Mise en place des données sur Knime

Avant toutes choses, nous nous sommes posés la question : Que faire des données manquantes ou erronées ? Sachant que nous n'avons pas accès à une expertise, nous partons du principe que les données sont non-imputables. C'est-à-dire que nous ne chercherons pas à « réparer » la donnée.

Nous sommes partis d'un fichier au format CSV comprenant les 605 000 lignes fournies. En premier lieu, nous avons filtré les colonnes jugées "peu pertinentes". De ce fait, nous avons utilisés un column filter qui nous a enlevé les colonnes suivantes: id_product, code_barre, url, creator, created_datetime, last_modified_datetime, product_name, sub_category, completeness, last_image_datetime, image_url, image_small_url, nutrition_score_fr_100g, ecoscore_score, ingredients_tags, additives_tags, brands_tags.

Pour continuer dans la filtration des données, nous avons utilisé un Rule-Based Row Filter. Cela nous a permis premièrement d'éliminer les lignes avec des valeurs nulles. Deuxièmement, de vérifier que les valeurs nutritionnelles sur 100g soient bien comprises entre 0 et 100. Et enfin une ultime vérification de la conformité des nutri scores (de a à e). Pour faciliter la manipulation des données l'ecoscore est passé d'une notation en lettres (de a à e) à une notation en chiffres (de 1 à 5).

Exploration de données

Nous avons débuté par l'utilisation d'une matrice de corrélation pour déterminer quelles sont les données qui peuvent être intéressantes pour découvrir une potentielle synergie entre elles. Cela nous a permis de découvrir que l'énergie et le gras sont reliés entre elles. Mais aussi que le gras et les glucides sont négativement liés. Les protéines et les sucres sont eux aussi négativement liés. Mais le plus intéressant est que le nutri-scores est lié à la catégorie des produits.

On s'est servi d'un box plot pour visualiser les nombreux outliers. Cependant, les données n'étant pas sur les même échelle, certaines comprises entre 1 et 4 et d'autres entre 0 et plusieurs milliers. Nous avons utilisé un normalizer paramètres en normalisation min, max (entre 0 et 1) pour que les données soit plus facilement analysables sur les différents jeux de données. Cela a permis d'enlever une grande partie des outliers présents dans la base. Ainsi nous avons pu avoir des analyses d'avantages précises sur le futur.

Histogramme

Pour continuer l'exploration nous avons choisi d'utiliser un histogramme accompagné d'un color manager qui a permis de différencier par différentes couleurs les catégories de produits entre elles. Il a permis de décrire les différentes proportions de produits, ainsi que celles des outliers dans les différentes catégories. Donc on peut voir que la catégorie reine est celle des "sugary snacks". Et la catégorie Fat and sauces qui possède le plus grand pourcentage d'outliers par rapport à son jeu de données.

Clustering

Dans le cadre de la partie clustering du jeu de données nous avons utilisé les 3 principaux outils de clustering (k-Means, Hierarchical Clustering et le DBSCAN). Avant de les utiliser nous avons enlevé les outliers avec un Numeric Outilers permettant d'enlever automatiquement et de manière plus précise les outliers présents. De plus, l'avantage du Numeric Outliers est que les paramètres sont conservés quand nous relançons le Workflow. Dans le cas du Hierarchical Clustering et du DBSCAN nous nous sommes servis du Row Sampling. Nous avons sampler à 1% pour des raisons de puissances de traitement.

Et nous avons stratifié sur les nutri-scores dans le but de vouloir trouver un nouveaux modèles de scores.

K-Means

Nous avons utilisé k-Means pour des raisons de simplicité et d'efficacité. Celui-ci permet l'utilisation de l'intégralité du jeu de données sans avoir besoin de sampler. Après avoir utilisé des boucles pour tester les différents nombres de clusters (de 3 à 10 clusters car au-dessus de 10 cela n'a pas vraiment de pertinence de même pour la borne inférieur à 3). Nous avons pu déterminer que les meilleurs résultats étaient pour 6 clusters avec les paramètres suivants: fat_100g, saturated_fat_100g, sugar_100g, nova_group, ecoscore_grade. Pour mesurer la qualité de nos résultats, nous utilisons un entropy scorer mesurant donc l'entropie, la taille du cluster, l'entropy normalisé (l'entropie qui nous intéressent), ainsi que la qualité (elles aussi intéressante pour le clustering). Nous avons relevé une entropie normalisée de 0,732 avec une qualité de 0,268. Les résultats ne paraissent pas spécialement bons, mais ce sont les meilleurs résultats que l'algorithme peut nous proposer avec le jeu de données.

Hierarchical clustering

Pour repartir sur les mêmes bases nous gardons les colonnes précédemment utilisées ainsi que 6 clusters. De plus, nous ajoutons les paramètres de la fonction de distance en euclidienne et une mesure en moyenne.

Grâce à la fonction distance de l'algorithme, le choix de 6 clusters était assez pertinent. Nous avons mesuré l'entropie. Nous obtenons les résultats suivants , une entropie normalisée de 0,834, avec une qualité de 0,166. Le hierarchical clustering apporte des résultats moins bons. Cependant, cette fluctuation est possiblement liée au sampling de 1%.

DBSCAN

Nous procédons de la même manière que le hierarchical clustering à la différence que nous passons par un nœud numeric distance pour paramétrer le DB Scan. Pour le nœud numeric distance, nous gardons une configuration euclidienne avec les mêmes colonnes sélectionnées. La configuration du DBSCAN se fait avec un epsilon de 0.7 et un minimum de points de 5. Nous obtenons en résultat une entropie normalisée de 0,916 et une qualité 0,084. Ce sont des résultats très médiocre par rapport aux autres méthodes de clustering.

Conclusion

Grâce à ces tests et résultats nous pouvons en conclure que l'algorithme K-means semble être le plus propice pour l'attribution des nouveaux scores aux différents produits.

Classification

Pour réaliser la classification, nous utiliser un arbre de décision avec un Decision tree learner et tree predictor. Après de nombreuses essais nous choisissons de paramétrer le Desicion tree learner de la manière suivante, nous paramétrons avec la colonne nutri score grade, en gini index et no pruning et un nombre minimum d'éléments de 20. Envoie cette méthode dans un tree predictor qui est paramétré à 70 000 lignes. Avec l'aide d'un scorer, nous pouvons voir que la moyenne du recall est de 0,7992, une précision moyenne de 0,802 et une accuracy de 0,811. Ce sont des résultats corrects pour le jeu de données. Pour tester le taux d'erreur moyen de cette fonction nous avons le choix d'utiliser une cross validation. Dans le meilleur des cas, nous sommes à un peu moins de 17% d'erreurs et dans le pire des cas, nous nous trouvons à un peu moins de 19% d'erreurs.

Pour tester la fiabilité, nous comparons la colonne nutri score et la prédiction dans un scorer.

Attribution du nouveau score

Le score que nous avons créé se base sur les colonnes de fat_100g, saturated_fat_100g, sugar_100g, nova_group, ecoscore. Ce dernier prend en compte les apports nutritifs des aliments, mais aussi le niveau de transformation et son impact écologique. Le but de ce score est dans un sens de créer des produits qui soit bon pour nous , tout en respectant l'environnement.

Nous avons mesuré l'attribution des produits sur ce nouveau score. Avec un arbre de décision composée d'un desicion tree learner paramétré en gini index, no pruning avec en class de référence le nouveau score. Quant au tree predictor, il est à 70 000 lignes pour respecter le jeu de données.

Nous avons appliqué cette méthode sur les différentes méthodes de clustering fait précédemment:

k-Means on obtient un taux d'erreur de 0,175% et une accuracy de 99,825%

Hierarchical Clustering on obtient un taux d'erreur de 0,409% et une accuracy de 99,591%

DBSCAN on obtient un taux d'erreur de 0,647% et une accuracy de 99,353%

Nous pouvons conclure que k-Means est l'algorithme le plus pertinent pour créer un nouveau score qui soit attribuable à tous les produits.