Etude de faisabilité d'une application basée sur un moteur de recommandation

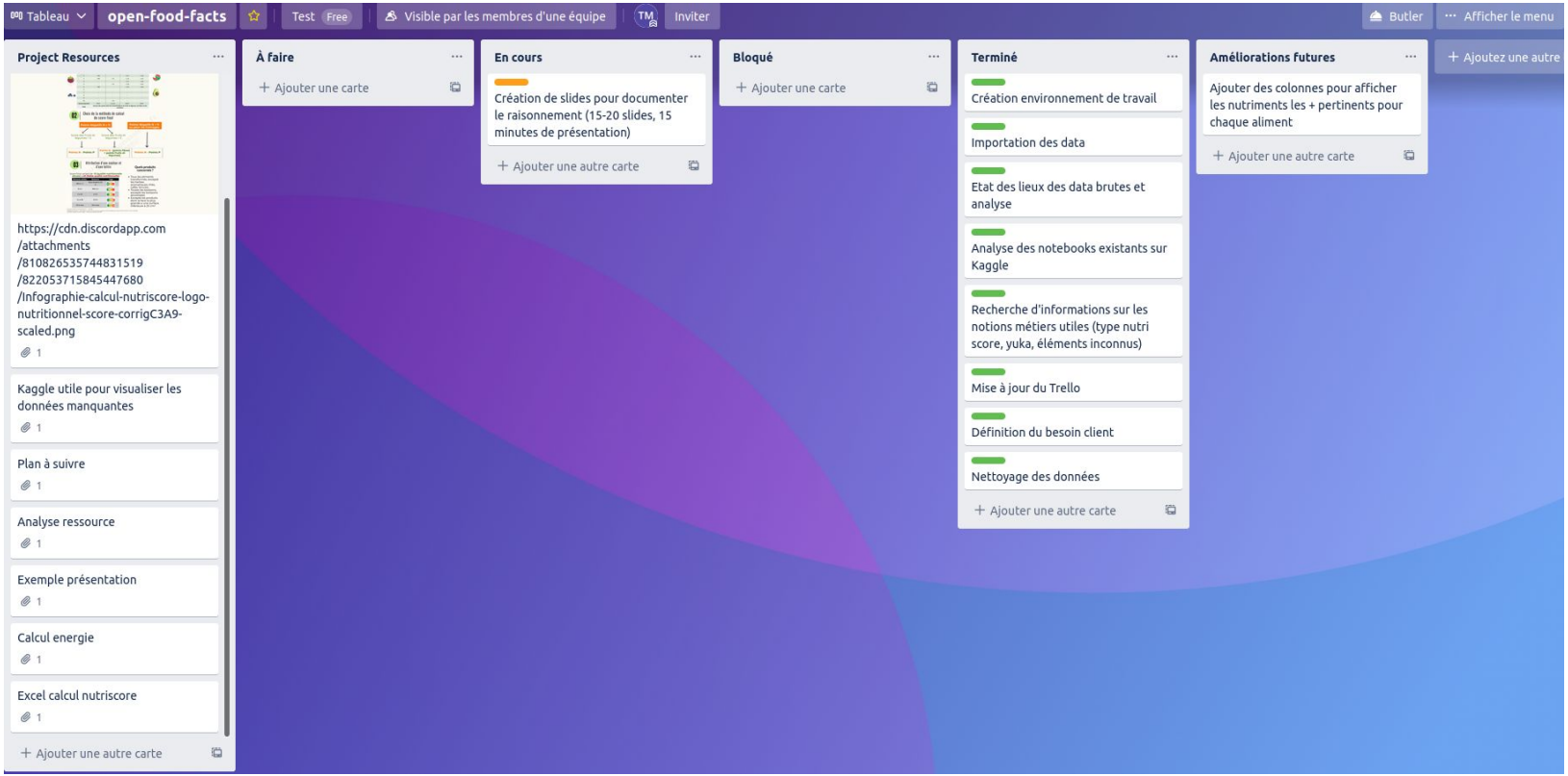
Tanguy Meyer

Sommaire

1. Contexte

- Foodflix : application permettant de recommander le meilleur produit à un utilisateur selon un mot clé ou un ensemble de mots clés
- Cahier des charges : Remonter les éléments liés au nutriscore
- Objectifs : Prouver que la donnée récupérée pourra être utilisée pour cette application
- Données : Open Food Facts, une base de données collaborative
- Démarche : Analyse et nettoyage de la donnée

2. Gestion de projet : Trello



3. Analyse du jeu de données

3.1. Etat des lieux

- Nombre de lignes : 356027
- Nombre de colonnes : 163
- Liste des colonnes :

'code', 'url', 'creator', 'created_t', 'created_datetime', 'last_modified_t', 'last_modified_datetime', 'product_name', 'generic_name', 'quantity', 'packaging', 'packaging_tags', 'brands', 'brands_tags', 'categories', 'categories_tags', 'categories_en', 'origins', 'origins_tags', 'manufacturing_places', 'manufacturing_places_tags', 'labels', 'labels_tags', 'labels_en', 'emb_codes', 'emb_codes_tags', 'first_packaging_code_geo','cities', 'cities_tags', 'purchase_places', 'stores', 'countries', 'countries_tags', 'countries_en', 'ingredients_text', 'allergens', 'allergens_en', 'traces', 'traces_tags', 'traces_en', 'serving_size', 'no_nutriments', 'additives_n', 'additives', 'additives_tags', 'additives_en', 'ingredients_from_palm_oil_n', 'ingredients_from_palm_oil', 'ingredients_from_palm_oil_tags', 'ingredients_that_may_be_from_palm_oil_n', 'ingredients_that_may_be_from_palm_oil', 'ingredients_that_may_be_from_palm_oil_tags', 'nutrition_grade_uk', 'nutrition_grade_fr', 'pnns_groups_1', 'pnns_groups_2', 'states', 'states_tags', 'states_en', 'main_category', 'main_category_en', 'image_url', 'image_small_url', 'energy_100g', 'energy-from-fat_100g', 'fat_100g', 'saturated-fat_100g', '-butyric-acid_100g', '-caproic-acid_100g', '-caprylic-acid_100g', '-capric-acid_100g', '-lauric-acid_100g', '-myristic-acid_100g', '-palmitic-acid_100g', '-stearic-acid_100g', '-arachidic-acid_100g', '-behenic-acid_100g', '-lignoceric-acid_100g', '-cerotic-acid_100g', '-montanic-acid_100g', '-melissic-acid_100g', 'monounsaturated-fat_100g', 'polyunsaturated-fat_100g', 'omega-3-fat_100g', '-alpha-linolenic-acid_100g', '-eicosapentaenoic-acid_100g', '-docosahexaenoic-acid_100g', 'omega-6-fat_100g', '-linoleic-acid_100g', '-arachidonic-acid_100g', '-gamma-linolenic-acid_100g', '-dihomo-gamma-linolenic-acid_100g', 'omega-9-fat_100g','-oleic-acid_100g', '-elaidic-acid_100g', '-gondoic-acid_100g', '-mead-acid_100g', '-erucic-acid_100g', '-nervonic-acid_100g', 'trans-fat_100g', 'cholesterol_100g', 'carbohydrates_100g', 'sugars_100g', '-sucrose_100g', '-glucose_100g', '-fructose_100g', '-lactose_100g', '-maltose_100g', '-maltodextrins_100g', 'starch_100g', 'polyols_100g', 'fiber_100g', 'proteins_100g', 'casein_100g', 'serum-proteins_100g','nucleotides_100g', 'salt_100g', 'sodium_100g', 'alcohol_100g', 'vitamin-a_100g', 'beta-carotene_100g', 'vitamin-d_100g', 'vitamin-e_100g', 'vitamin-k_100g', 'vitamin-c_100g', 'vitamin-b1_100g', 'vitamin-b2_100g', 'vitamin-pp_100g', 'vitamin-b6_100g', 'vitamin-b9_100g', 'folates_100g', 'vitamin-b12_100g', 'biotin_100g', 'pantothenic-acid_100g', 'silica_100g', 'bicarbonate_100g', 'potassium_100g', 'chloride_100g', 'calcium_100g', 'phosphorus_100g', 'iron_100g', 'magnesium_100g', 'zinc_100g', 'copper_100g', 'manganese_100g', 'fluoride_100g', 'selenium_100g', 'chromium_100g', 'molybdenum_100g', 'iodine_100g', 'caffeine_100g', 'taurine_100g', 'ph_100g', 'fruits-vegetables-nuts_100g', 'fruits-vegetables-nuts-estimate_100g', 'collagen-meat-protein-ratio_100g', 'cocoa_100g', 'chlorophyl_100g','carbon-footprint_100g', 'nutrition-score-fr_100g', 'nutrition-score-uk_100g', 'glycemic-index_100g', 'water-hardness_100g'

Nous allons classer les colonnes similaires pour rendre ça plus digeste !

3.2 Classification des données

Les métadonnées

- Donne des informations sur la donnée en elle-même
- Pas utile pour l'application
- Liste des métadonnées :

'code', 'url', 'creator', 'created_t', 'created_datetime', 'last_modified_t', 'last_modified_datetime'

3. Analyse du jeu de données
3.2 Classification des données

Les informations logistiques

- Donne des informations sur l'origine et la dénomination du produit
- Partiellement utile pour l'application afin d'identifier le produit
- Liste des informations logistiques :

'product_name', 'generic_name', 'quantity', 'packaging', 'packaging_tags', 'brands', 'brands_tags', 'categories', 'categories_tags', 'categories_en', 'origins', 'origins_tags', 'manufacturing_places', 'manufacturing_places_tags', 'labels', 'labels_tags', 'labels_en', 'emb_codes', 'emb_codes_tags', 'first_packaging_code_geo', 'cities', 'cities_tags', 'purchase_places', 'stores', 'countries', 'countries_tags', 'countries_en', 'pnns_groups_1', 'pnns_groups_2', 'main_category', 'main_category_en'

Les colonnes finissant par _tags et _en reprennent les mêmes valeurs que les colonnes du même nom en modifiant la typographie ou la langue.

3. Analyse du jeu de données
3.2 Classification des données

Les valeurs nutritionnelles textuelles

- Donne des informations textuelles sur les valeurs nutritionnelles du produit
- Contient des infos importantes pour l'utilisateur de l'application
- Liste des valeurs nutritionnelles textuelles :

'ingredients_text', 'allergens', 'allergens_en', 'traces', 'traces_tags', 'traces_en', 'serving_size', 'no_nutriments', 'additives_n', 'additives', 'additives_tags', 'additives_en', 'ingredients_from_palm_oil_n', 'ingredients_from_palm_oil', 'ingredients_from_palm_oil_tags', 'ingredients_that_may_be_from_palm_oil_n', 'ingredients_that_may_be_from_palm_oil', 'ingredients_that_may_be_from_palm_oil_tags', 'nutrition_grade_uk', 'nutrition_grade_fr'

Les colonnes finissant par _tags et _en reprennent les mêmes valeurs que les colonnes du même nom en modifiant la typographie ou la langue

3. Analyse du jeu de données
3.2 Classification des données

Les valeurs nutritionnelles numériques

- Donne les valeurs nutritionnelles du produit pour 100 grammes
- Contient des infos importantes pour l'utilisateur et le nutriscore
- Liste des valeurs nutritionnelles numériques :

'energy_100g', 'energy-from-fat_100g', 'fat_100g', 'saturated-fat_100g', '-butyric-acid_100g', '-caproic-acid_100g', '-caprylic-acid_100g', '-capric-acid_100g', '-lauric-acid_100g', '-myristic-acid_100g', '-palmitic-acid_100g', '-stearic-acid_100g', '-arachidic-acid_100g', '-behenic-acid_100g', '-lignoceric-acid_100g', '-cerotic-acid_100g', '-montanic-acid_100g', '-melissic-acid_100g', 'monounsaturated-fat_100g', 'polyunsaturated-fat_100g', 'omega-3-fat_100g', '-alpha-linolenic-acid_100g', '-eicosapentaenoic-acid_100g', '-docosahexaenoic-acid_100g', 'omega-6-fat_100g', '-linoleic-acid_100g', '-arachidonic-acid_100g', '-gamma-linolenic-acid_100g', '-dihomo-gamma-linolenic-acid_100g', 'omega-9-fat_100g', '-oleic-acid_100g', '-elaidic-acid_100g', '-gondoic-acid_100g', '-mead-acid_100g', '-erucic-acid_100g', '-nervonic-acid_100g', 'trans-fat_100g', 'cholesterol_100g', 'carbohydrates_100g', 'sugars_100g', '-sucrose_100g', '-glucose_100g', '-fructose_100g', '-lactose_100g', '-maltose_100g', '-maltodextrins_100g', 'starch_100g', 'polyols_100g', 'fiber_100g', 'proteins_100g', 'casein_100g', 'serum-proteins_100g', 'nucleotides_100g', 'salt_100g', 'sodium_100g', 'alcohol_100g', 'vitamin-a_100g', 'beta-carotene_100g', 'vitamin-d_100g', 'vitamin-e_100g', 'vitamin-k_100g', 'vitamin-c_100g', 'vitamin-b1_100g', 'vitamin-b2_100g', 'vitamin-pp_100g', 'vitamin-b6_100g', 'vitamin-b9_100g', 'folates_100g', 'vitamin-b12_100g', 'biotin_100g', 'pantothenic-acid_100g', 'silica_100g', 'bicarbonate_100g', 'potassium_100g', 'chloride_100g', 'calcium_100g', 'phosphorus_100g', 'iron_100g', 'magnesium_100g', 'zinc_100g', 'copper_100g', 'manganese_100g', 'fluoride_100g', 'selenium_100g', 'chromium_100g', 'molybdenum_100g', 'iodine_100g', 'caffeine_100g', 'taurine_100g', 'ph_100g', 'fruits-vegetables-nuts_100g', 'fruits-vegetables-nuts-estimate_100g', 'collagen-meat-protein-ratio_100g', 'cocoa_100g', 'chlorophyl_100g', 'carbon-footprint_100g', 'nutrition-score-fr_100g', 'nutrition-score-uk_100g', 'glycemic-index_100g', 'water-hardness_100g'

3.2 Classification des données

Les informations additionnelles

- Donne le statut de la ligne en question et un lien renvoyant vers une image du produit
- Le lien vers l'image peut être utile pour l'application
- Liste des informations additionnelles :

'states', 'states_tags', 'states_en', 'image_url', 'image_small_url'

3. Analyse du jeu de données

3.3 Choix des données

- Les données choisies pour être utilisées dans le nettoyage doivent :

  - Etre pertinentes pour l'utilisateur et l'application :

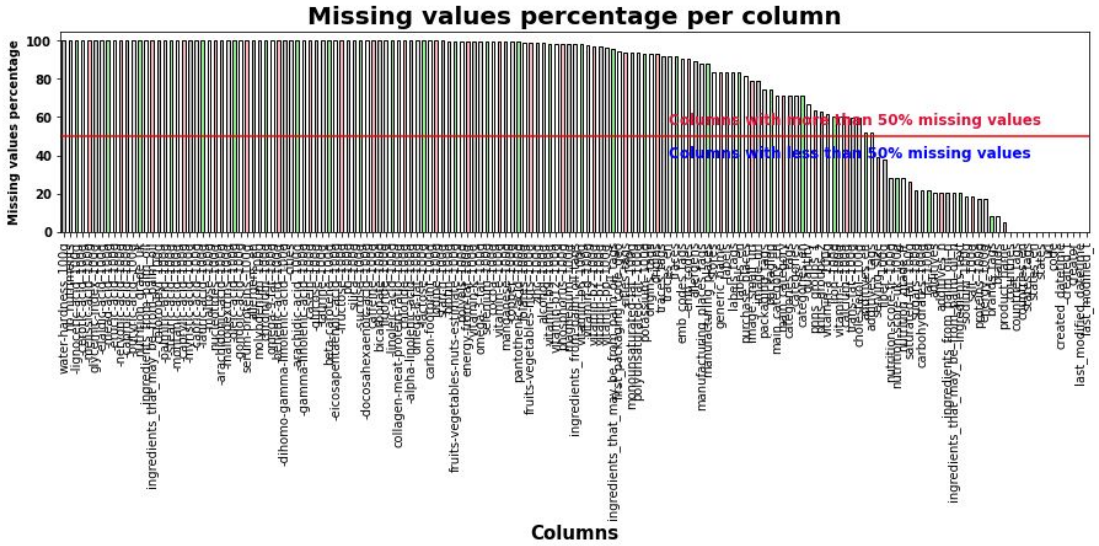
    Valeurs nutritionnelles les plus répandues, allergènes, présence d'huile de palme, nutriscore

  - Permettre de calculer le nutriscore

  - Permettre de compléter des valeurs manquantes (ces colonnes pourront être supprimées à la fin du nettoyage)

3. Analyse du jeu de données
3.3 Choix des données

Valeurs manquantes :



Beaucoup de valeurs pas renseignées

100% de valeurs manquantes pour :

'ingredients_that_may_be_from_palm_oil', 'ingredients_from_palm_oil', 'nutrition_grade_uk, ' '-buty,ric-acid_100g', '-caproic-acid_100g', '-nervonic-acid_100g', '-erucic-acid_100g', '-mead-acid_100g, '-elaidic-acid_100g', 'glycemic-index_100g', '-melissic-acid_100g, '-cerotic-acid_100g', '-lignoceric-acid_100g', 'no_nutriments' ,'water-hardness_100g'

3. Analyse du jeu de données

3.3 Choix des données

#	Code	Type	Description	Raison
1	**product_name**	Object	Nom du produit	Pour l'application
2	generic_name	Object	Nom alternatif	Pour le nettoyage
3	**brands**	Object	Marque du produit	Pour l'application
4	**countries_en**	Object	Pays de distribution	Pour l'application et le nettoyage
5	**pnns_groups_1**	Object	Catégorie du produit	Pour l'application et le nettoyage
6	**ingredients_text**	Object	Liste des ingrédients	Pour l'application
7	**allergens**	Object	Liste des allergènes	Pour l'application
8	**traces**	Object	Liste des traces d'allergènes	Pour l'application
9	**additives_en**	Object	Liste des additifs	Pour l'application

3. Analyse du jeu de données

3.3 Choix des données

#	Code	Type	Description	Raison
10	ingredients_from_palm_oil_n	Object	Présence d'huile de palme	Pour l'application
11	nutrition_grade_fr	Object	Nutrigrade	Pour le nettoyage
12	fat_100g	Float	Graisses pour 100g	Pour l'application
13	carbohydrates_100g	Float	Glucides pour 100g	Pour l'application
14	energy_100g	Float	Energie pour 100g	Pour le nutriscore
15	saturated-fat_100g	Float	Graisses saturées pour 100g	Pour le nutriscore
16	sugars_100g	Float	Sucre pour 100g	Pour le nutriscore
17	fiber_100g	Float	Fibres pour 100g	Pour le nutriscore
18	salt_100g	Float	Sel pour 100g	Pour le nutriscore

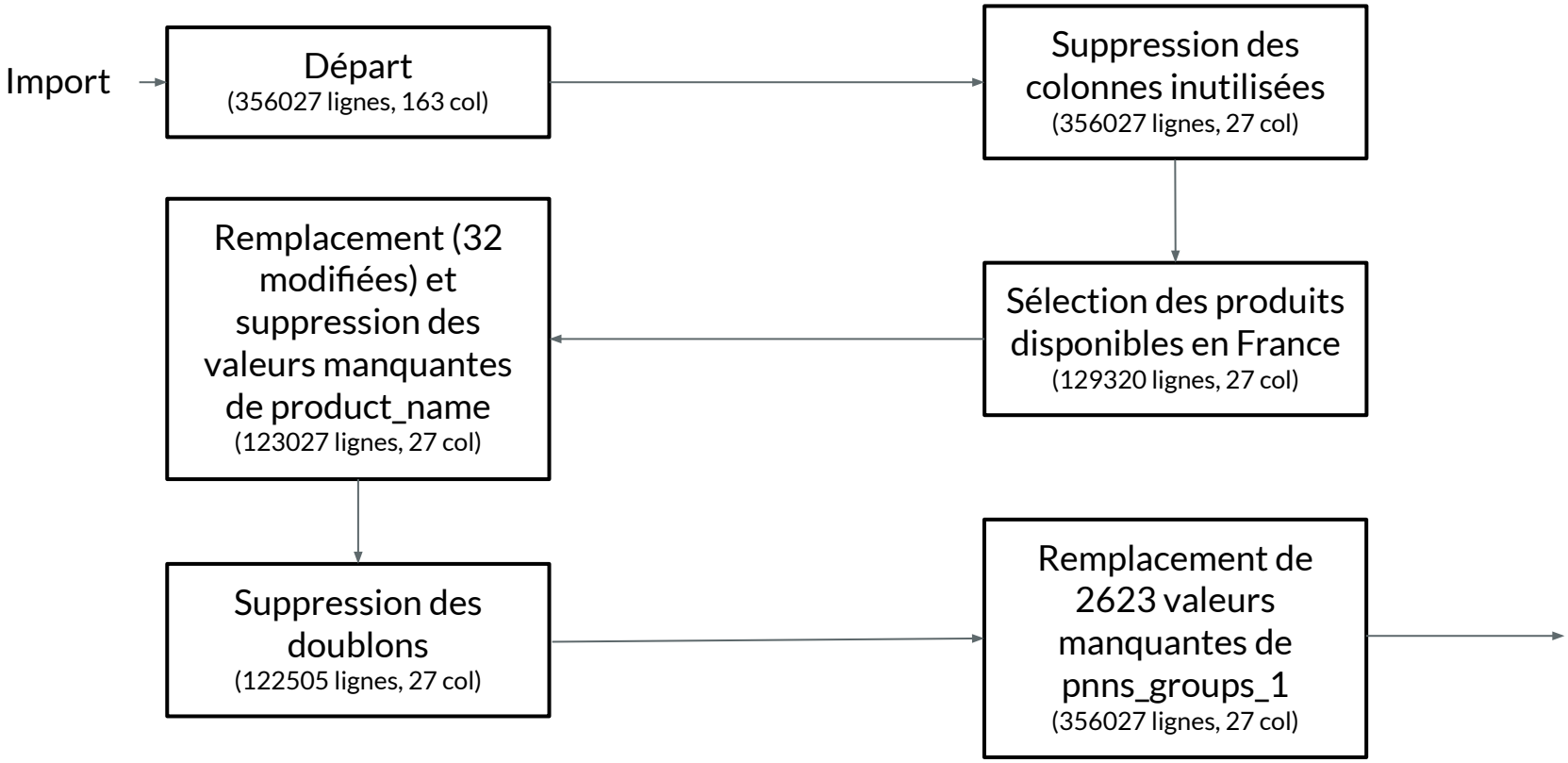
3. Analyse du jeu de données

3.3 Choix des données

#	Code	Type	Description	Raison
19	proteins_100g	Float	Protéines pour 100g	Pour le nutriscore
20	nutrition-score-fr_100g	Float	Valeur du nutriscore	Pour l'application
21	fruits-vegetables-nuts_100g	Float	Taux de fruits, légumes et noix	Pour le nutriscore
22	vitamin-d_100g	Float	Vitamine D pour 100g	Pour l'application
23	vitamin-c_100g	Float	Vitamine C pour 100g	Pour l'application
24	calcium_100g	Float	Calcium pour 100g	Pour l'application
25	iron_100g	Float	Fer pour 100g	Pour l'application
26	image_url	Object	Lien de l'image du produit	Pour l'application
27	pnns_groups_2	Object	Catégorie du produit alternative	Pour le nettoyage

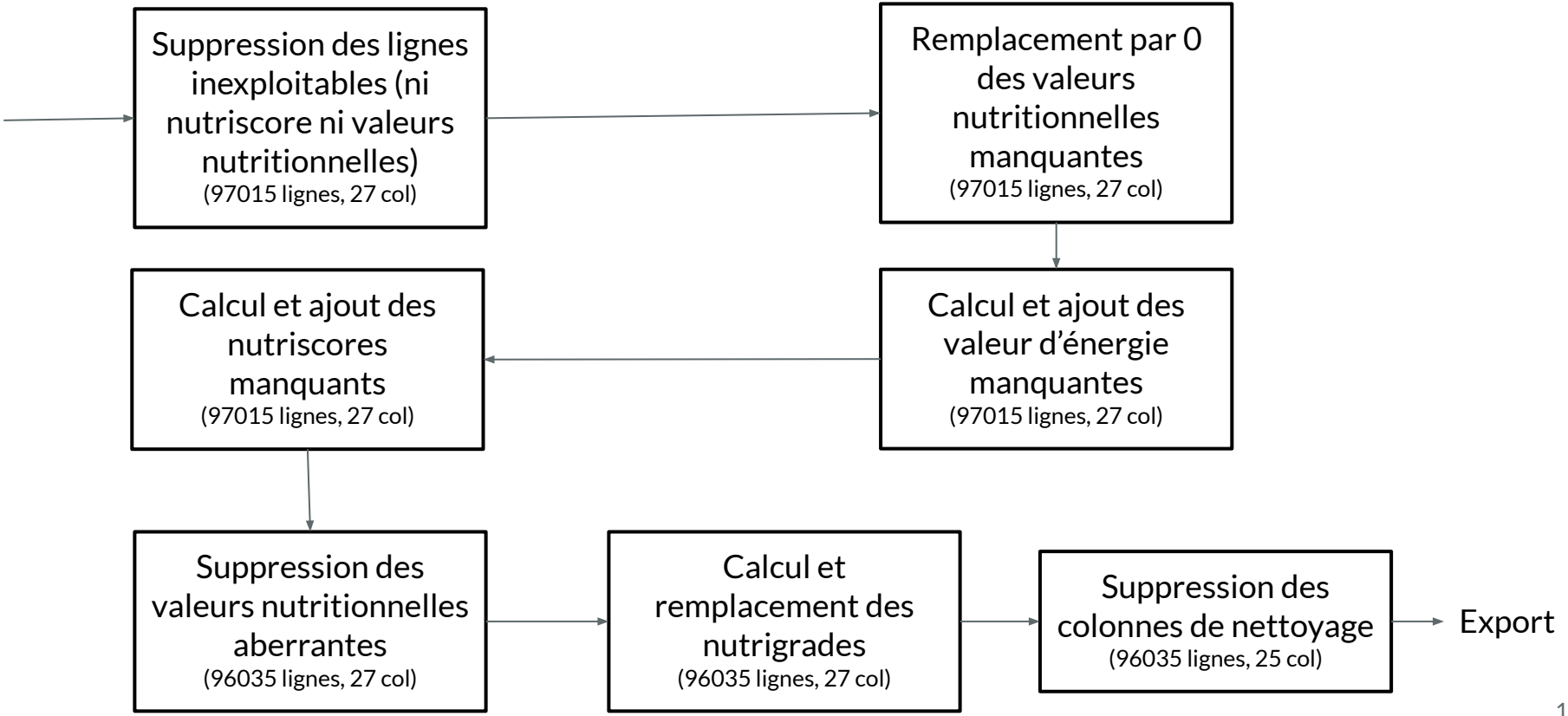
4. Nettoyage du jeu de données

4.1 Processus de nettoyage



4. Nettoyage du jeu de données

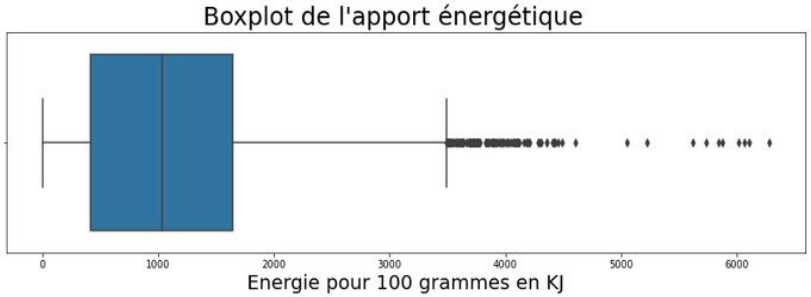
4.1 Processus de nettoyage



4. Nettoyage du jeu de données
4.2 Méthodes de nettoyage et de calcul

- Energie :

```
#  E KJ = (37 x lipides) + (17 x protéines) + (17 x glucides) + (8 x fibres)
# On calcule l'énergie suivant la formule ci-dessus quand la valeur est manquante
df["energy_100g"] = df.apply(
    lambda row: 37*row['fat_100g'] + 17*row['proteins_100g'] + 17*row['carbohydrates_100g'] + 8*row['fiber_100g']
    if np.isnan(row["energy_100g"])
    else row["energy_100g"],
    axis=1)
```



- Nutrigrade :

```
# Liste des valeurs de nutrigrade
grad=['a', 'b', 'c', 'd', 'e']
# Valeurs limites de nutriscore pour les solides
scoS=[-1, 2, 10, 19]
# Valeurs limites de nutriscore pour les liquides
scoL=[1, 5, 9]
```

```
def nutrigrade(catg, nutriscore):
    """Calcule le nutrigrade à partir du nutriscore"""
    # Les critères ne sont pas les mêmes pour les liquides et les solides
    if "beverage" in str(catg).lower():
        # L'eau a toujours un nutrigrade de a
        if "eau" in str(catg).lower():
            return grad[0]
        return grad[calcul_points(nutriscore, scoL) + 1]
    return grad[calcul_points(nutriscore, scoS)]
```

```
# On calcule le nutrigrade pour chaque valeur manquante
df["nutrition_grade_fr"] = df.apply(
    lambda row: nutrigrade(row["pnns_groups_1"], row["nutrition-score-fr_100g"]),
    axis=1)
```

4. Nettoyage du jeu de données

4.2 Méthodes de nettoyage et de calcul

- Nutriscore :

```
# Mise en place des listes contenant les limites pour chaque valeur nutritionnelle
# Limites spécifiques aux solides
ptsNrjS=[335, 670, 1005, 1340, 1675, 2010, 2345, 2680, 3015, 3350]
ptsSucS=[4.5, 9, 13.5, 18, 22.5, 27, 31, 36, 40, 45]
ptsLegS=[40,60,80,80,80]

# Limites partagées entre solides et liquides
ptsSod=[0.090, 0.180, 0.270, 0.360, 0.450, 0.540, 0.630, 0.720, 0.810, 0.900]
ptsGra=[1, 2.1, 3.2, 4.3, 5.4, 6.5, 7.6, 8.7, 9.8, 10.9]
ptsFib=[0.9, 1.9, 2.8, 3.7, 4.7]
ptsPro=[1.6, 3.2, 4.8, 6.4, 8.0]

# Limites spécifiques aux liquides
ptsNrjL=[0, 30 , 60, 90, 120, 150, 180, 210, 240, 270]
ptsSucL=[0, 1.5, 3, 4.5, 6, 7.5, 9, 10.5, 12, 13.5]
ptsLegL=[40, 40, 60, 80, 80, 80, 80, 80, 80, 80]
```

```
def calcul_points(val, pts):
    """Donne le nombre de points de la valeur val par rapport à son placement relatif à la liste pts"""
    points=0
    # On parcourt parcourt la liste des limites jusqu'à trouver une limite majorant notre valeur nutritionnelle
    for points in range(len(pts)):
        if val <= pts[points]:
            return points
    return points+1


def nutriscore(catg, nrj, suc, gra, sel, leg, fib, pro):
    """Calcule le nutriscore à partir des valeurs nutritionnelles"""
    # Compteur de points positifs
    p=0
    # Compteur de points négatifs
    n=0
    p = calcul_points(fib, ptsFib)
    n = n + calcul_points(gra, ptsGra) + calcul_points(sel/2.54, ptsSod)
    # Cas où le produit est un liquide
    if "beverage" in str(catg).lower():
        p = p + calcul_points(leg, ptsLegL)
        n = n + calcul_points(nrj, ptsNrjL) + calcul_points(suc, ptsSucL)
    # Cas où le produit est un solide
    else:
        p = p + calcul_points(leg, ptsLegS)
        n = n + calcul_points(nrj, ptsNrjS) + calcul_points(suc, ptsSucS)
    # Cas où les points des protéines ne sont pas pris en comptes
    if n >= 11 and leg < 100 and "fromage" not in str(catg).lower():
        return(n-p)
    # Cas où les points des protéines sont pris en compte
    p = p + calcul_points(pro, ptsPro)
    return (n-p)
```

```
# On calcule les nutriscores qui ne sont pas renseignés
df["nutrition-score-fr_100g"] = df.apply(
    lambda row: nutriscore(row["pnns_groups_1"], row["energy_100g"],
                           row["sugars_100g"], row["saturated-fat_100g"],
                           row["salt_100g"], row["fruits-vegetables-nuts_100g"],
                           row["fiber_100g"], row["proteins_100g"])
    if np.isnan(row["nutrition-score-fr_100g"])
    else row["nutrition-score-fr_100g"],
    axis=1)
```

5. Next step

- Appliquer et étudier le modèle de régression linéaire appliqué au nutriscore
- Prendre davantage de valeurs nutritionnelles pour pouvoir donner à l'utilisateur les valeurs nutritionnelles les plus pertinentes suivant le produit
- Utiliser un autre modèle que le modèle de régression linéaire pour calculer l' énergie de façon plus juste
- Utiliser un modèle différent pour le calcul de la graisse à partir de la graisse saturée et le calcul des glucides à partir des sucres pour éviter de dépasser les valeurs supérieures à 100, par exemple un modèle logarithmique