DataScientest

Challenge Rakuten

Rapport d'exploration et de data visualisation

Présenté par :

CHELOUAH Slimane
LOLO MVOUMBI Simplice
ZIDI Riadh
MORMICHE Nicolas

Table des matières

1 - Introduction du projet

1.1 - Contexte

Ce projet s'inscrit dans le cadre d'un challenge Rakuten qui porte sur le thème de la classification multimodale (texte et image) de produits à e-commerce à grande échelle où l'objectif est de prédire le code type de chaque produit tel que défini dans le catalogue de Rakuten France. D'un point de vue économique, le Multimodal Learning est une technique utile pour les e-commerces car il permet de catégoriser automatiquement des produits en fonction des images et des informations fournies par les commerçants.

1.2 - Périmètre

L'étude des classificateurs multimodales est adaptée à la détermination de catégorie de produit à partir d'image et de texte. On se limite ici aux données fournies par Rakuten concernant les produits et leurs images associées afin de vérifier son efficacité.

Nous prendrons comme base le modèle de référence fourni par Rakuten :

- Pour les données d'image, une version du modèle Residual Networks (ResNet)
- Pour les données texte, un classificateur CNN simplifié

1.3 - Objectifs

Nous devons fournir un fichier y_test.csv, dans le même format que le fichier y_train.csv, en associant chaque identifiant produit ('Unnamed: 0') au code type ('prdtypecode') associé :

1	Unnamed: 0	Un identifiant entier pour le produit. Cet identifiant permet d'associer le produit à son code type de produit correspondant.
2	prdtypecode	Code type de produit (catégorie)

On utilisera ici un classificateur multimodal pour rattacher un produit à son code type où les données à disposition comportent 27 catégories pour 84916 produits et 84916 images.

Rakuten ayant déjà réalisé une classification multimodale à l'aide de divers modèles, le but est de proposer une alternative en testant différents modèles (ex : CNN ou autre) afin d'obtenir un meilleur score F1 pondéré que celui proposé par Rakuten actuellement :

- Classification des textes avec CNN : 0.8113
- Classification des images avec ResNet50 : 0,5534

1.4 - Notions importantes

Le Multimodal Learning est une notion centrale du projet. Il existe plusieurs méthodes pour fusionner les informations textuelles et visuelles :

- Les réseaux de neurones multimodaux
  (modèle de fusion CNN / RNN / TNN)

- L'apprentissage par transfert
  (transférer des connaissances apprises d'une modalité à une autre)

- Les techniques de co-apprentissage ou d'apprentissage adversarial
  (représentation des données multimodales)

- Les réseaux de neurones auto-encodeurs
  (reconstruire les données d'entrée en passant par une représentation latente)

Concernant la classification des données multimodales (texte et image), nous aurons plusieurs choix de modèle à traiter afin de vérifier le score F1 pondéré de chacun :

- Classification des textes (ex : BERT ou CNN)

- Classification des images (ex : CNN, RNN ou TNN)

Pour former et entraîner un modèle de Multimodal Learning, on peut utiliser des techniques de Deep Learning tel que les réseaux de neurones où l'on utilise des encodeurs :

- Former un modèle multimodal à double encodeur (ex : BERT et ResNet-50)

- Entraîner un modèle multimodal à encodeur commun (ex : ALBEF)

2 - Exploration des données

Rakuten met à disposition un jeu de donnée et fourni X_train, y_train et X_test (sans les données y_test correspondantes). Nous avons donc décidé de fusionner X_train et y_train pour l'exploration des données.

- **Importer les fichiers CSV**

```
df_X = pd.read_csv("X_train.csv", sep = ',', index_col = 0)
df_y = pd.read_csv("Y_train.csv", sep = ',', index_col = 0)
```

- **Fusionner les DataFrames X (variables explicatives) et y (variable cible)**

```
fusion = pd.merge(df_X, df_y, left_index=True, right_index=True)
```

- **Afficher les premières lignes du DataFrame**

```
fusion.head()
```

- **Afficher les informations du DataFrame**

```
fusion.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 84916 entries, 0 to 84915
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   designation  84916 non-null  object
 1   description  55116 non-null  object
 2   productid    84916 non-null  int64 
 3   imageid      84916 non-null  int64 
 4   prdtypecode  84916 non-null  int64 
dtypes: int64(3), object(2)
memory usage: 5.9+ MB
```

- **Afficher les statistiques descriptives des variables quantitatives**

```
fusion.describe()
```

	productid	imageid	prdtypecode
count	8.491600e+04	8.491600e+04	84916.000000
mean	2.555468e+09	1.152691e+09	1773.219900
std	1.588656e+09	1.751427e+08	788.179885
min	1.839120e+05	6.728400e+04	10.000000
25%	6.760519e+08	1.056269e+09	1281.000000
50%	3.190506e+09	1.213354e+09	1920.000000
75%	3.995599e+09	1.275646e+09	2522.000000
max	4.252012e+09	1.328824e+09	2905.000000

- **Afficher les valeurs uniques prises par prdtypecode :** 27 catégories au total

```
fusion.prdtypecode.nunique()
```

- **Vérifier les doublons :** aucun doublon

```
fusion.duplicated().sum()
```

- **Vérifier les valeurs nulles :** description compte 29800 NaN sur 84916 soit 35%

```
fusion.isna().sum()
```

```
designation        0
description    29800
productid          0
imageid            0
prdtypecode        0
dtype: int64
```

- **Exploration des données**

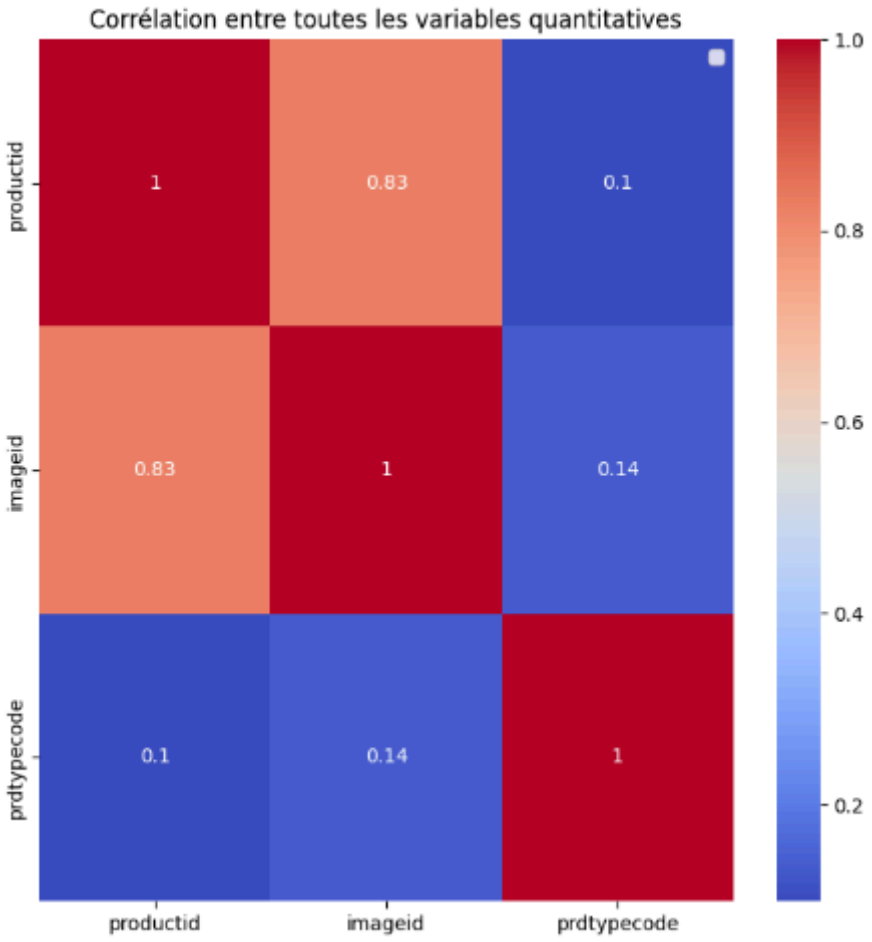
Vous trouverez le tableau d'exploration des données ici : [exploration des données]

- Onglet Fusion pour les données texte
- Onglet X_train_images pour les données image

3 - Visualisation des données

Heatmap : corrélation entre toutes les variables quantitatives

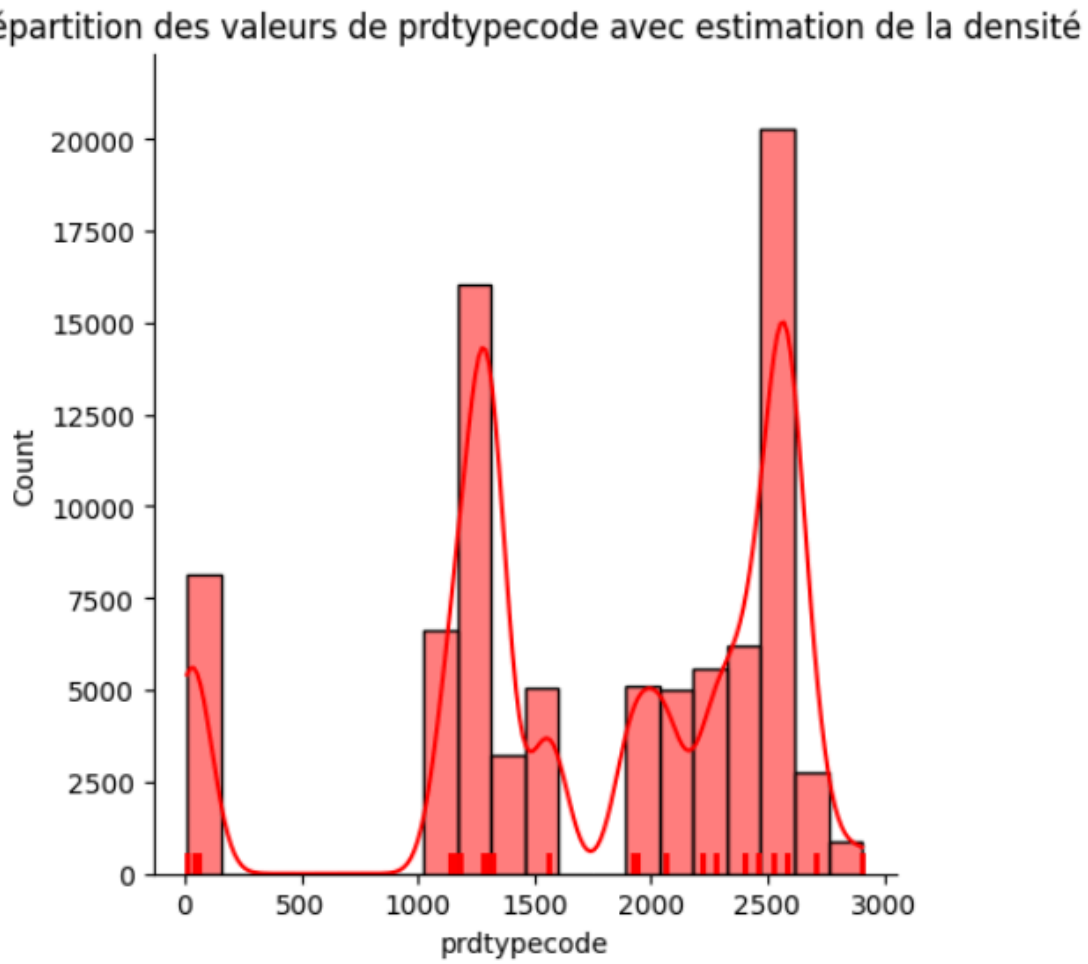
```
fusion = fusion.drop(['description'], axis=1)
fusion = fusion.drop(['designation'], axis=1)
fig, ax = plt.subplots(figsize = (8,8))
sns.heatmap(fusion.corr(), ax = ax, cmap = "coolwarm", annot=True)
plt.title('Corrélation entre toutes les variables quantitatives')
plt.savefig("reports/figures/heatmap.png", bbox_inches='tight')
```



Conclusion : on ne retrouve aucune corrélation intéressante entre les variables quantitative mis à part entre productid et imageid qui ne semble pas être significatif pour le projet

Histogramme avec estimation de la densité : prdtypecode

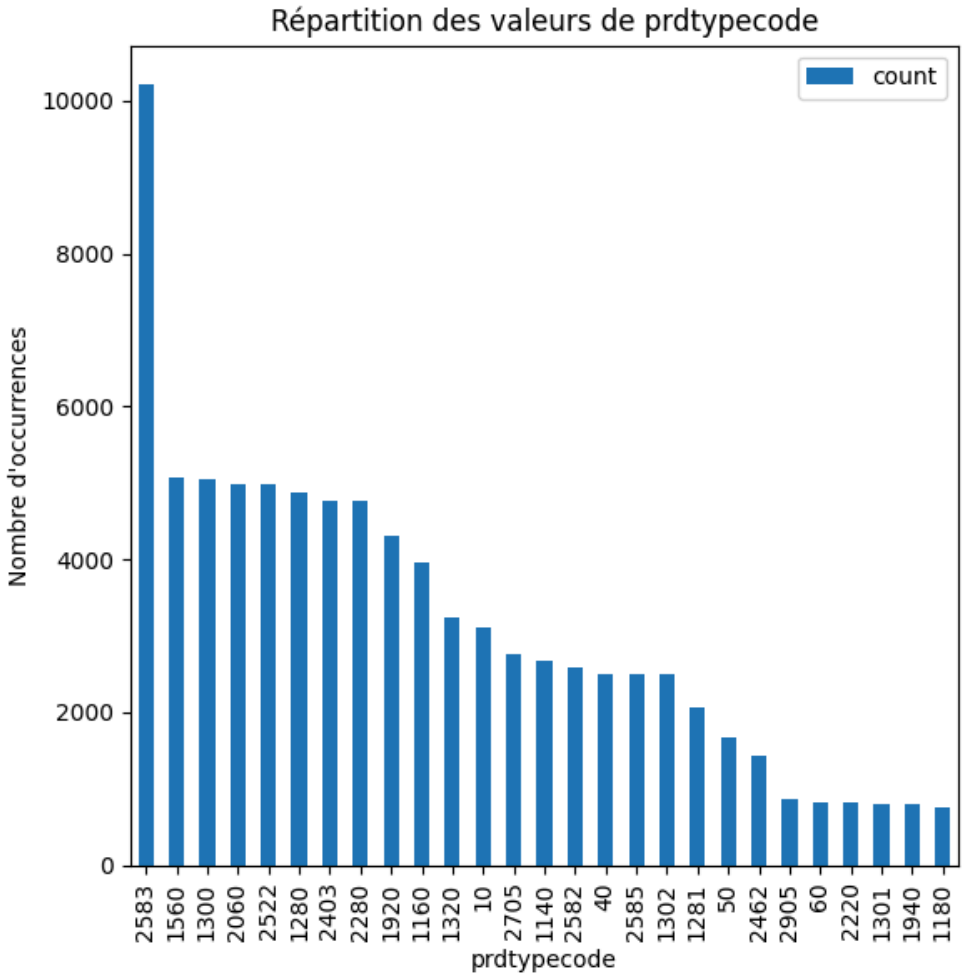
```
sns.displot(fusion.prdtypecode, bins=20, kde = True, rug=True,
color="red")
plt.title('Répartition des valeurs de prdtypecode avec estimation de la
densité')
plt.savefig("reports/figures/histogramme_avec_estimation_densite.png",
bbox_inches='tight')
```



Conclusion : on constate que les valeurs de codes type produits se répartissent sur 3 plages de valeurs principales.

Histogramme : prdtypecode

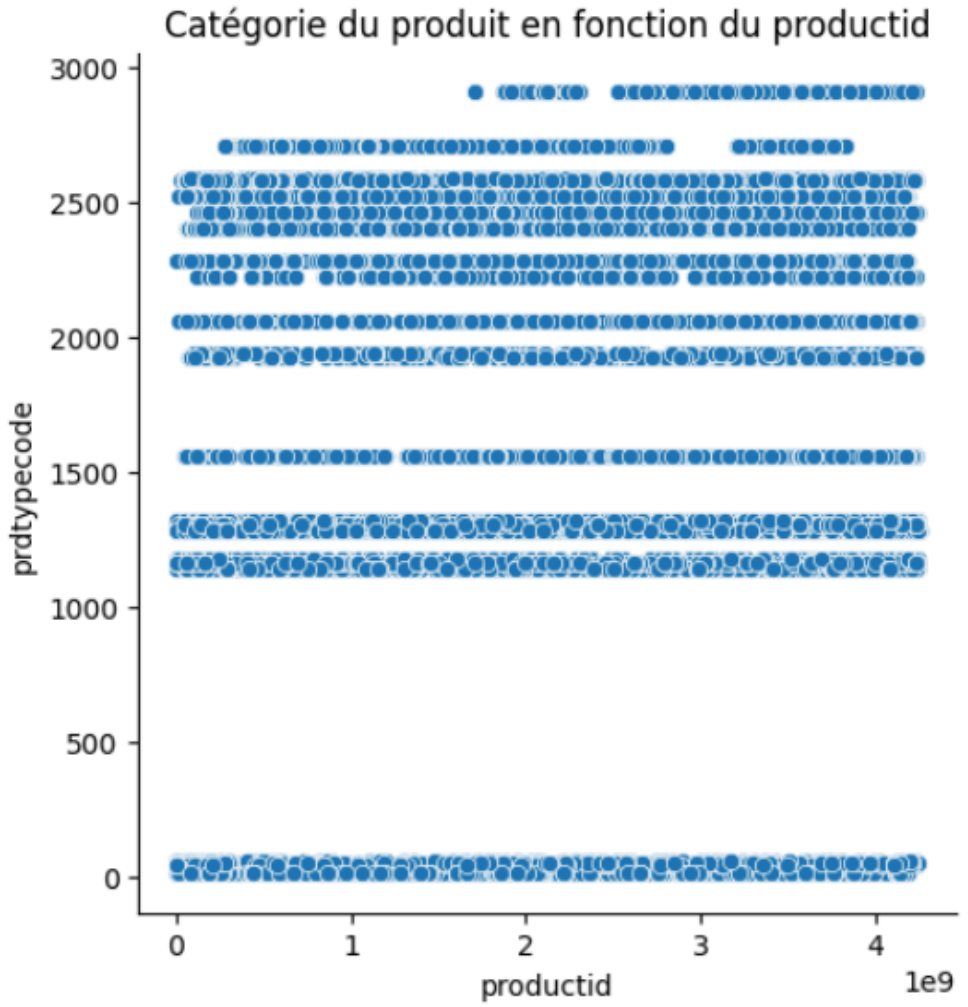
```
distribution = fusion['prdtypecode'].value_counts()
distribution_df = distribution.reset_index()
distribution_df.columns = ['prdtypecode', 'count']
distribution_df.plot(kind='bar', x='prdtypecode', y='count')
plt.xlabel('prdtypecode')
plt.ylabel('Nombre d\'occurrences')
plt.title('Répartition des valeurs de prdtypecode')
plt.savefig("reports/figures/historamme.png", bbox_inches='tight')
```



Conclusion : graphe qui reprend les représentations des 27 catégories de type produits ordonnés par ordre décroissants. La catégorie 2583 se détache fortement en terme de représentation.

Nuage de point : productid et prdtypecode

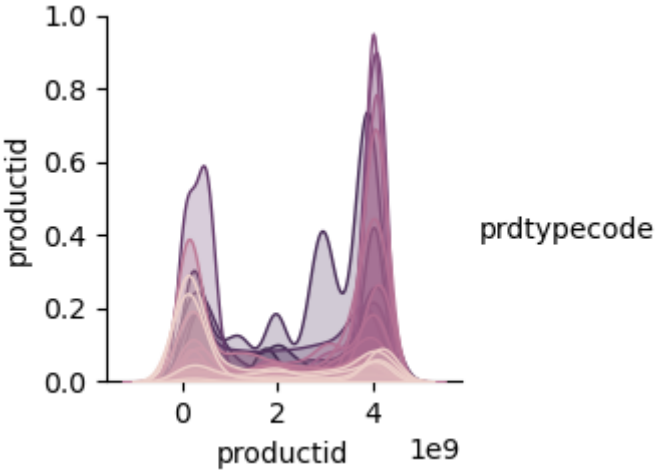
```
sns.relplot(x=fusion.productid, y=fusion.prdtypecode)
plt.title('Catégorie du produit en fonction du productid')
plt.savefig("reports/figures/scatterplot.png", bbox_inches='tight')
```



Conclusion : les code type produits se répartissent par plage discrètes de valeurs. D'où la répartition en lignes. Dans la majorité des cas, pour chaque code type produit, les codes produits s'étendent sur les plages entière des productid

Pairplot : corrélation entre productid et prdtypecode

```
sns.pairplot(fusion[['productid', 'prdtypecode']],
hue="prdtypecode",diag_kind="kde")
plt.title('Catégorie du produit en fonction du productid')
plt.savefig("reports/figures/pairplot.png", bbox_inches='tight')
```



Conclusion : il y a des distributions similaires de productid selon le code type produit : accumulation en début et fin de numérotations

4 - Compréhension et manipulation des données

4.1 - Cadre et pertinence

Le jeu de données fourni par Rakuten étant déjà divisé en deux ensembles d'entraînement et de test, on constate qu'il nous manque y_test. Les données test n'étant pas complet, nous avons décidé de ne garder que l'ensemble d'entraînement comme jeu de données. Nous avons donc procédé à un ré-échantillonage :

```
def re_echantillonage(X, y):
   X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=66)
   return X_train, X_test, y_train, y_test
```

On retrouve ainsi la variable cible 'prdtypecode' ainsi que les variables explicatives de notre futur modèle de Multimodal Learning :

- descriptif (texte)
- image_imageid_product_productid.jpg (image)

La particularité du jeu de données réside dans le Multimodal Learning où nous devrons mettre en relation texte et image pour determiner le prdtypecode correspondant. De plus, nous avons procédé à un traitement naturel du langage (NLP) pour la partie texte.

4.2 - Pre-processing et feature engineering

- Gestion des valeurs nulles

Etant donnée la variable ‘description’ qui contient 35% de valeurs nulles et pour conserver toutes les informations, nous avons fusionné les colonnes ‘designation’ et ‘description’ dans une nouvelle colonne ‘descriptif’ :

```
def fusion_description_designation():
    X = pd.read_csv('data/X_train.csv', index_col=0)
    y = pd.read_csv('data/Y_train.csv', index_col=0)
    X['descriptif'] = X['description'].astype(str).replace("nan", "") +
" " + X['designation'].astype(str)
    X = X.drop(['designation', 'description'], axis=1)
    return X, y
```

- Standardisation, discrétisation et dichotomisation

Le jeu de donnée étant vraisemblablement pré-traité côté Rakuten, on constate qu’il est épuré et qu’aucune standardisation ni discrétisation n’est utile. De plus, la dichotomisation semble également inutile car les seuls variables catégorielles sont une chaîne de caractère et la variable cible. Aucune nouvelle donnée n’est apportée pour enrichir le dataset car cela sort du cadre du Challenge Rakuten.

- Détection des langues

Nous avons analysé les langues présentes dans la colonnes ‘descriptif’ :

```
def traitement_lang(X):
    df_lang=pd.DataFrame(X['descriptif'].apply(detect_lang))
    print(df_lang.value_counts())
    print("nombre d'éléments=", len(df_lang))
    return df_lang
```

On retrouve un grand nombre de langues (français et anglais prédominant) :

descriptif

Français	62518	Portugais	344	Lituanien	81		
Anglais	12456	Suédois	274	Slovaque	79		
Italien	2476	Indonésien	247	Albanais	76		
Catalan	1084	Estonien	213	Swahili	73		
Roumain	886	Tagalog	199	Letton	49		
Espagnol	625	Croate	190	Tchèque	39		
Néerlandais	512	Finnois	182	Turc	33		
Norvégien	488	Slovène	177	Hongrois	31		
Afrikaans	440	Gallois	157	Erreur_langue		9	
Danois	409	Polonais	107	Vietnamien		4	
Allemand	375	Somali	83				

Après avoir créé un nouveau fichier CSV lors du nettoyage sur les mots, et lors de son import, le nouveau dataframe a transformé les chaînes de caractères vides en valeurs nulles, ce qui nous a permis de constater la présence de 9 modalités (ex : 9 ‘Erreur_langue’) où le commerçant n’a renseigné aucune description ni aucune designation sur son produit.

- Traitement naturel du langage

Nous avons appliqué un nettoyage sur les mots en multi-langues qui nous a permis de réduire les données pour les futurs nuages de mot :

```
def clean_column_descriptif(texte):
    words = set(stopwords.words())
    texte = re.sub("\s{2,}", " ", texte)
    texte = texte.lower()
    texte = BeautifulSoup(texte, "html.parser").get_text()
    texte = re.sub('\d+', '', texte)
    texte = re.sub("[^a-zA-Z]", " ", texte)
    texte = ' '.join([
        word for word in texte.split() if word not in words
    ])
    texte = ' '.join([word for word in texte.split() if len(word) > 2])
    return texte
```

- Traitement des images

Concernant les images fournies par Rakuten, elles sont séparées en deux ensembles d'entraînement ('image_train') et de test ('image_test').

```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FC2466EF1C0>
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FC246605FD0>
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FC2466EF1C0>
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FC246605730>
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FC2466EF1C0>
```

On constate qu'elles n'ont besoin d'aucune modification ni aucun redimensionnement car elles sont toutes au format jpg, de taille 500x500 et de mode RGB :

```
def pre_processing_image():
    for filename in os.listdir("data/images/image_train"):
        image = Image.open("data/images/image_train/"+filename)
        if image.mode != "RGB":
            print("L'image {filename} n'est pas en mode RGB")
        if image.size != (500, 500):
            print("L'image {filename} n'est pas en 500x500 pixels")
        if image.format != "JPEG":
            print("L'image {filename} n'est pas au format JPEG")
        image.close()
    print("Les images sont en mode RGB, 500x500 pixels et JPEG")
```

4.3 - Visualisations et statistiques

L'étape du traitement du langage nous a permis de passer d'une liste de mot de 222906 à 137099 après nettoyage. Aussi, pour mieux comprendre la variable cible, nous avons réalisé une étude sur les mots les plus fréquents par prdtypecode en créant un DataFrame contenant le nombre d'occurence des mots présents dans chacunes des catégories.

```
def word_occurence_by_prdtypecode(X, y):
    df = pd.DataFrame()
    df['prdtypecode']= y['prdtypecode']
    df['descriptif'] = X['descriptif']
    df['mots'] = df['descriptif'].apply(nettoyer_et_separer)
    compteurs_par_classe = {}
    for classe, groupe in df.groupby('prdtypecode'):
        tous_les_mots = [
            mot for liste_mots in groupe['mots'] for mot in liste_mots
        ]
        compteurs_par_classe[classe] = Counter(tous_les_mots)

    df_result = pd.DataFrame(compteurs_par_classe).fillna(0).astype(int)
    return df_result
```

Nous avons ensuite pu réaliser un ensemble de 27 nuages de mot concernant les différentes catégories afin de se faire une idée plus claire sur le contenu de chacune. Ainsi, on visualise mieux à quoi correspond chaque prdtypecode, par exemple :

- 1920 : semble correspondre à la literie (oreiller, couverture, canapé, lit, maison etc)
- 1940 : semble correspondre à l'épicerie (chocolat, sucre, café, bio, fruit, etc)
- 2462 : semble correspondre aux jeux vidéos (jeux, xbox, playstation, manette, etc)

```
def nuage_de_mots(df_result):
    for classe in df_result.columns:
        wordcloud = WordCloud(width=800,
height=400).generate_from_frequencies(df_result[classe])
        plt.figure(figsize=(10, 5))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.title(f'Nuage de mots pour la classe {classe}')
        plt.savefig(f"reports/figures/nuage_de_mot/{classe}.png",
bbox_inches='tight')
```

- Analyse de la distribution des données

La nature du préprocessing ne modifie pas la distribution des données relevée durant la phase d'exploration. Concernant les outliers sur la colonne 'descriptif', il serait intéressant de vérifier le nombres de mots pour trouver des outliers (ex : avec moins de 3 mots), indiquant probablement une description inutile ou incomplète.