Classifiez automatiquement des biens de consommation

Zaccaria Amillou

Sommaire

- Contexte
- Données
- EDA
- Traitement de texte
- Traitement image
- Modélisation
- Script

Contexte

Obejctifs

Automatiser la classification des images des produits

- Images
- Description

Données

```
In 13  1  # affichage sample df
       2  display(df.sample(5))
```

Executed at 2024.03.12 09:22:03 in 8ms

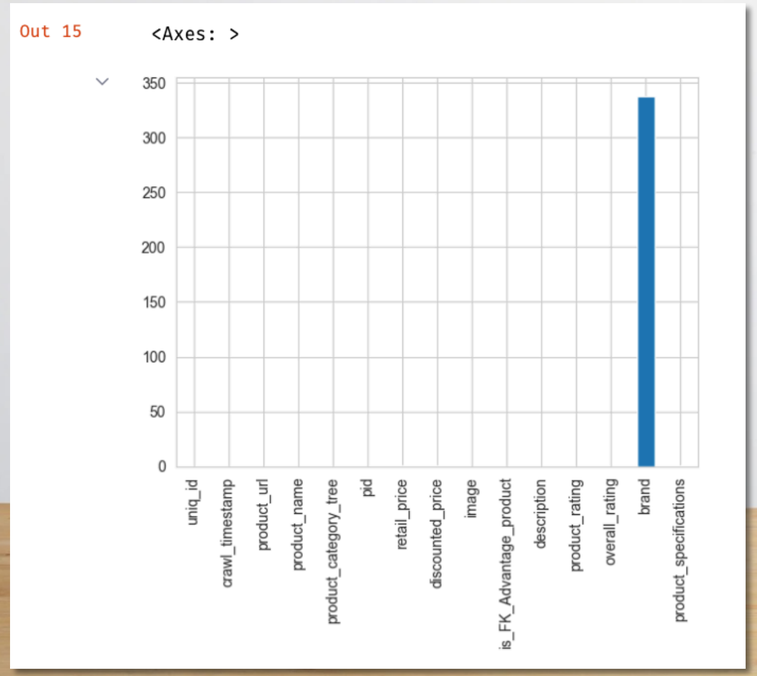
5 rows | 5 rows × 15 columns | Static Output

	uniq_id	crawl_timestamp	product_url	product_name	proc
71	f2658bad8c2b7d5b08984c6ac34267f	2016-04-19 05:00:32 +0000	http://www.flipkart.com/prime-printed-8-seater ..	Prime Printed 8 Seater Table Cover	["Ho
430	a54b94096938252901d3f7f8de97bffl	2016-03-11 06:55:07 +0000	http://www.flipkart.com/ocean-vglass0051/p/itm ..	ocean VGLASS0051	["Ki
306	7e11aaffb1d08f8091ea94598f7865a	2015-12-01 06:13:00 +0000	http://www.flipkart.com/lal-haveli-rajasthani- ..	Lal Haveli Rajasthani Handmade Decorative Wood ..	["Ho
809	1d1be744e491ed61e705c20e4a72320	2015-12-12 11:46:53 +0000	http://www.flipkart.com/oxyglow-lacto-bleach-f ..	Oxyglow Lacto Bleach & Fruit Massage Cream Wit ..	["Be
225	67d6b4a8aa19d57740b0440365b7762	2015-12-01 12:40:44 +0000	http://www.flipkart.com/lal-haveli-decorative- ..	Lal Haveli Decorative Dholak Musician Showpiec ..	["Ho

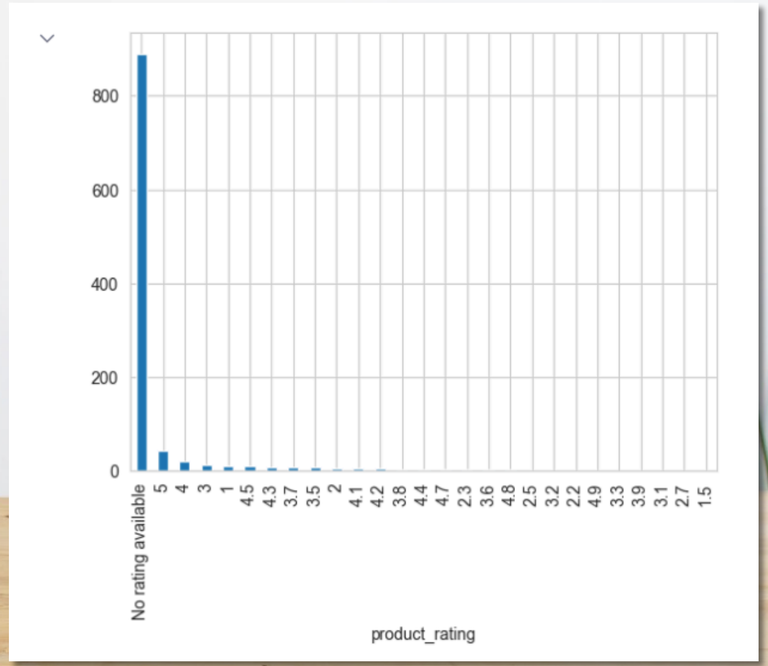
le df est composée par 1050 lignes et 15 colonnes

EDA
Exploration des données

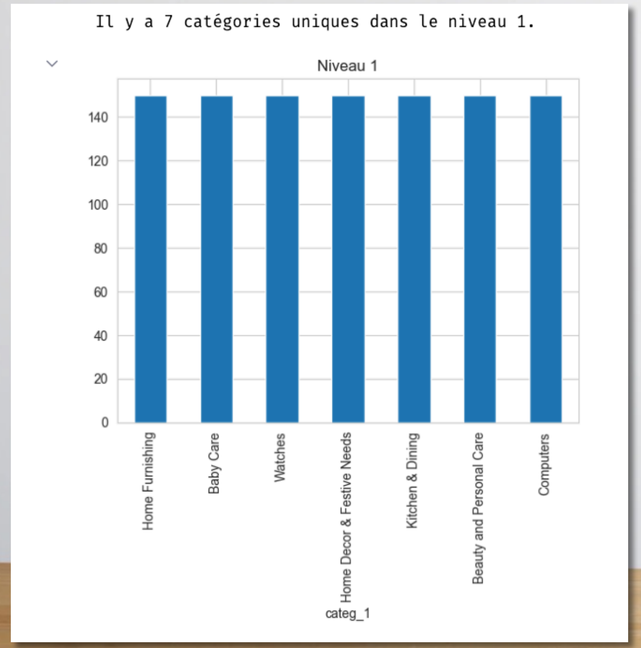
Valeurs nulles



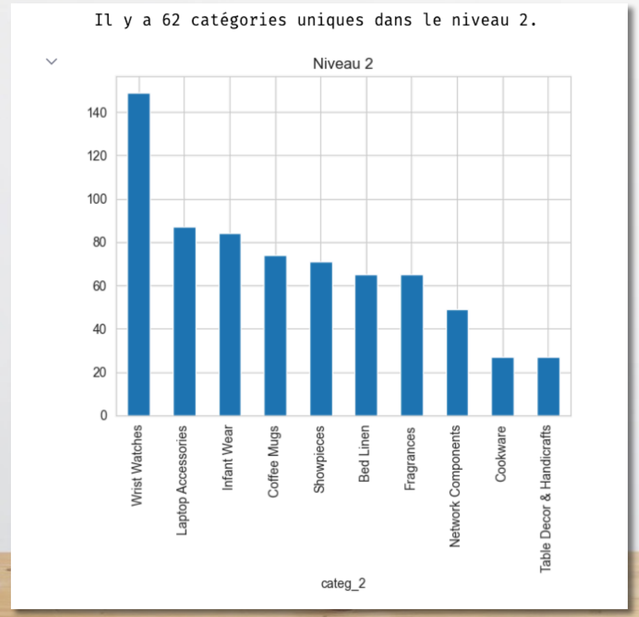
Contrôle notes clients



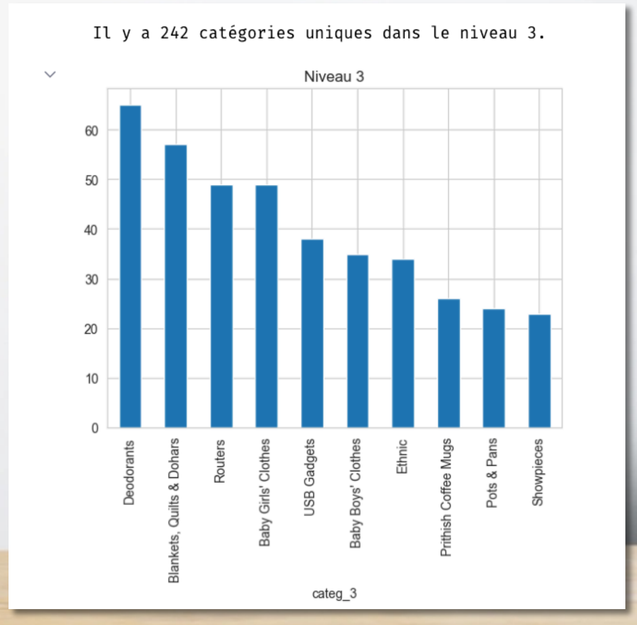
EDA
Exploration des données



Niveaux 1



Niveaux 2



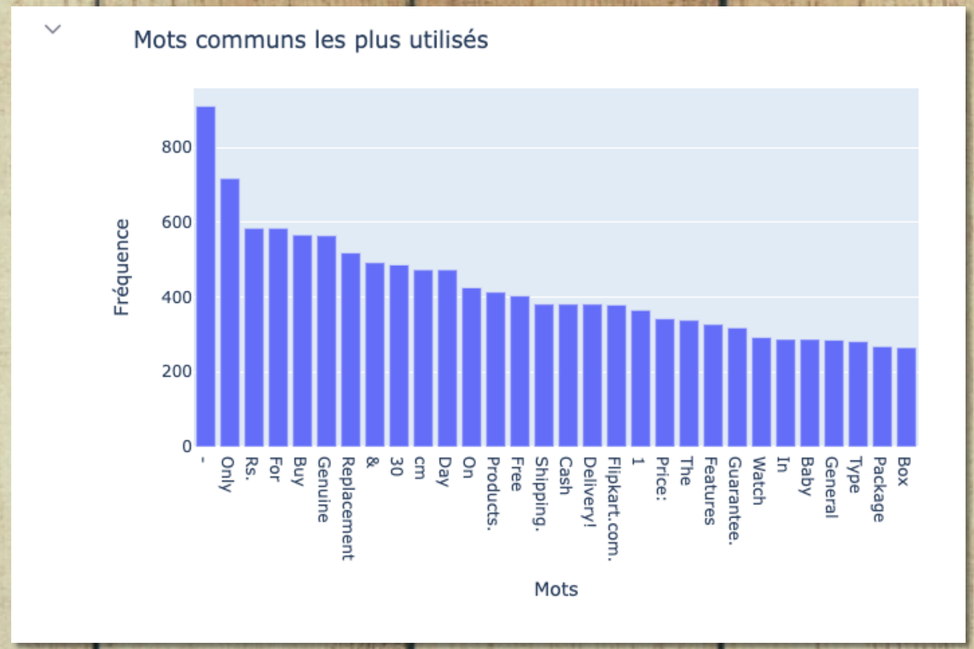
Niveaux 3

Traitement texte

10 rows 10 rows × 3 columns

	description	text_len	text_char_len
0	Key Features of Elegance Polyester Multicolor ...	230	1191
1	Specifications of Sathiyas Cotton Bath Towel (...	74	371
2	Key Features of Eurospa Cotton Terry Face Towe ...	215	1044
3	Key Features of SANTOSH ROYAL FASHION Cotton P ...	145	714
4	Key Features of Jaipur Print Cotton Floral Kin ...	214	984
5	Maserati Time R8851116001 Analog Watch - For ...	42	231
6	Camerii WM64 Elegance Analog Watch - For Men, ...	44	222
7	T STAR UFT-TSW-005-BK-BR Analog Watch - For B ...	131	649
8	Alfajr WY16B Youth Digital Watch - For Men, B ...	66	342
9	TAG Heuer CAU1116.BA0858 Formula 1 Analog Watc ...	63	365

Traitement texte



Pré-traitement texte

```
In 4
def process_text(text, min_len_word=3, force_is_alpha=True):

    tokenizer = RegexpTokenizer(r'\w+')
    lemmatizer = WordNetLemmatizer()

    # Convertit le texte en minuscules
    text = str(text).lower()
    raw_tokens_list = tokenizer.tokenize(text)
    cleaned_tokens_list = [w for w in raw_tokens_list if w not in stopwords.words('english')]
    non_rare_tokens = [w for w in cleaned_tokens_list if w not in list_rare_words]
    more_than_N = [w for w in non_rare_tokens if len(w) ⩾ min_len_word]

    if force_is_alpha:
        alpha_tokens = [w for w in more_than_N if w.isalpha()]
    else:
        alpha_tokens = more_than_N

    lemmatized_tokens = [lemmatizer.lemmatize(w) for w in alpha_tokens]

    if eng_words:
        english_tokens = [w for w in lemmatized_tokens if w in eng_words]
    else:
        english_tokens = lemmatized_tokens

    trans_text = ' '.join(english_tokens)

    return trans_text
```

Traitement texte

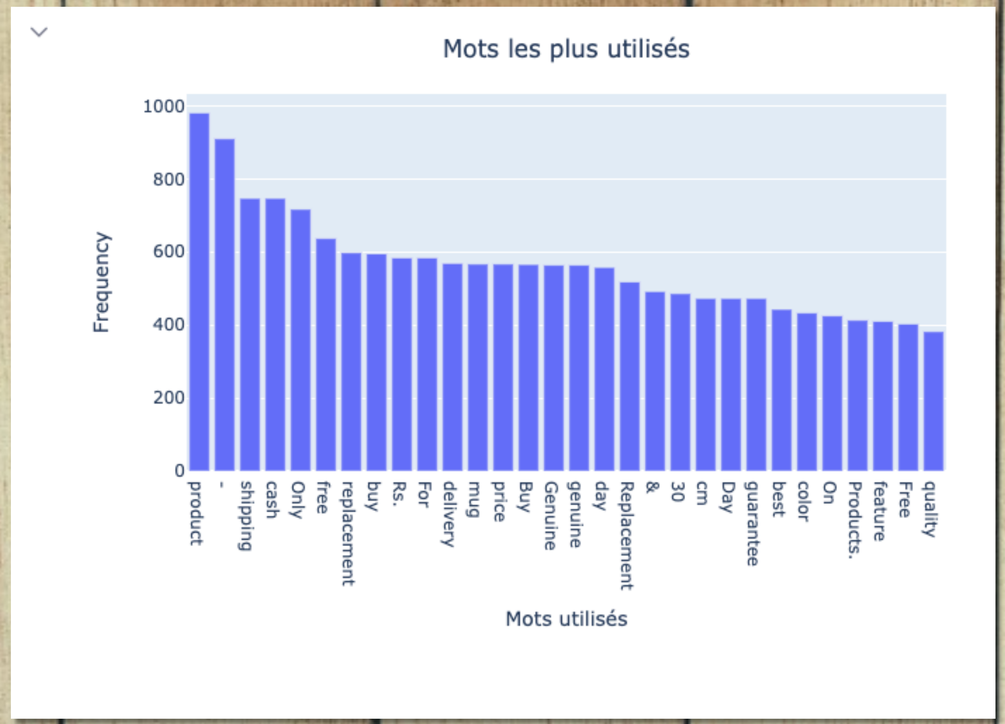
- Tokenization
- Stopwords
- Filtrage mots rares
- Filtrage longeueur
- Lemmatisation
- Filtrage mots anglais
- Consturction texte final

Traitement texte

10 rows | 10 rows × 6 columns | Static Output

	description	clean_text	text_len	text_len_clean	text_char_len	text_char_len_clean
305	Buy Pg handicrafts Beaded two drawers sofa set ...	buy handicraft beaded two drawer sofa set show ...	45	28	231	173
245	Buy Kalash Kadhai 11.5 L for Rs.1584 online. K ...	buy best price free shipping cash delivery gen ...	29	12	144	73
563	Fluid FS201-BL01 Analog-Digital Watch - For W ...	fluid digital watch woman buy fluid digital wa ...	46	27	264	158
126	Flipkart.com: Buy Svayam Premium Manicure Duo ...	buy premium manicure duo stainless steel price ...	29	16	169	101
514	Maxima 04615CMGY Gold Analog Watch - For Men ...	maximum gold watch men buy maximum gold watch ...	50	29	263	163
59	Specifications of Brillare Science Dandruff Co ...	specification science dandruff control shampoo ...	41	24	228	152
682	Flipkart.com: Buy VLCC Natural Sciences Oil Fr ...	buy natural science oil free gel pack price ge ...	37	17	188	97
1039	Key Features of Reiki Crystal Products Showpie ...	key feature crystal product showpiece singing ...	270	140	1336	877
497	Times 284TMS284 Party-Wedding Analog Watch - ...	time party wedding watch woman buy time party ...	42	23	237	134
69	Specifications of GAYATRI CREATIONS ORANGE KID ...	specification gayatri creation orange set cont ...	48	33	225	184

Traitement texte

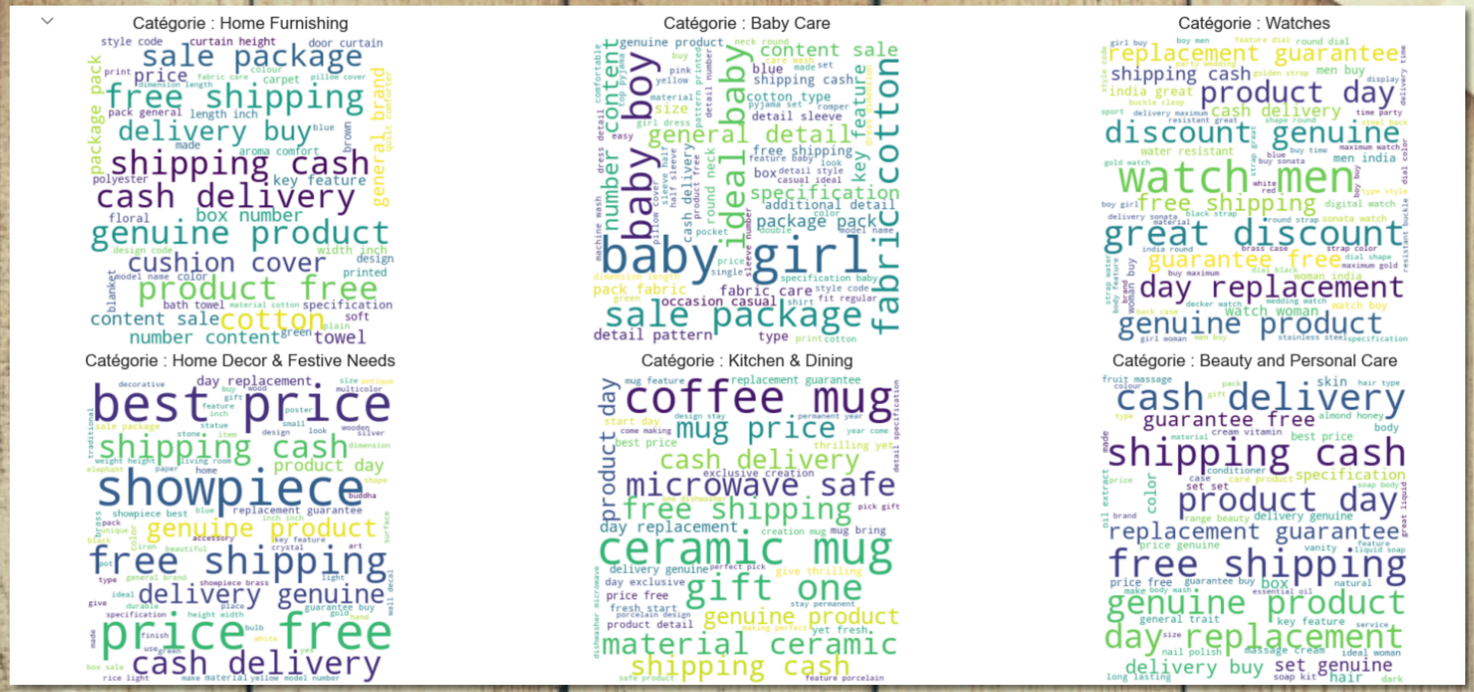


Traitement texte

Wordcloud

Traitement texte

Wordcloud

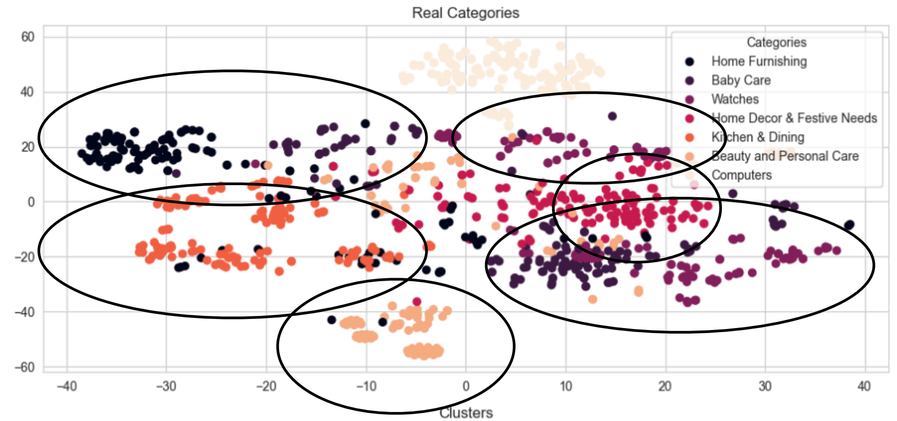


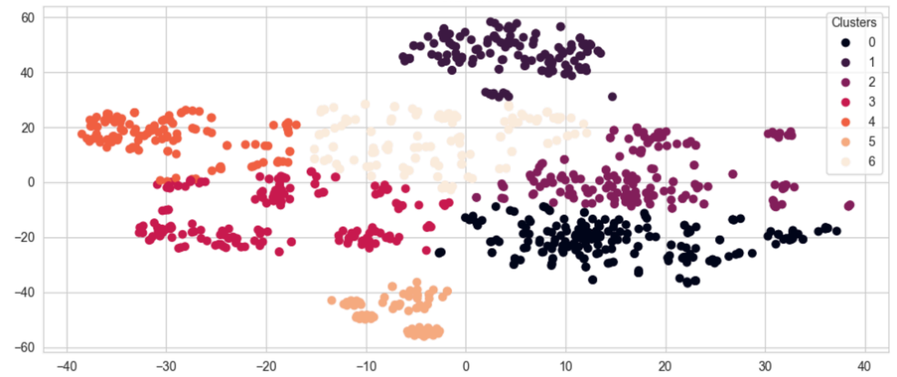
Traitement texte

Test Faisabilité

- TD-IDF
- Word2Vec
- BERT
- USE

```
Méthode TD-IDF
ARI :  0.4577140433392419 Temps d'exécution :  3.0
```



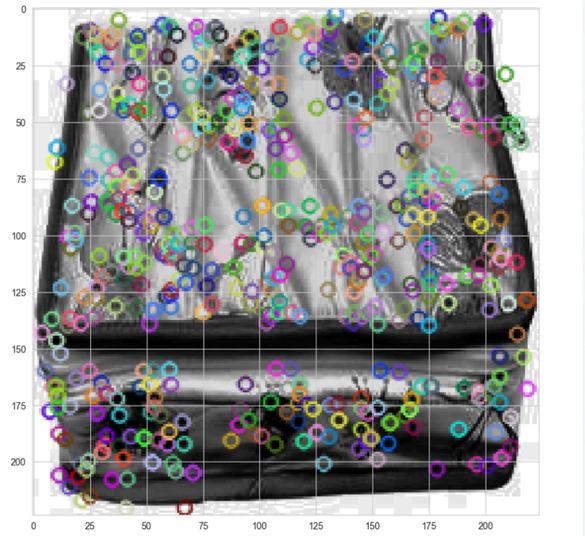


Traitement images

Extractions des features

Sift

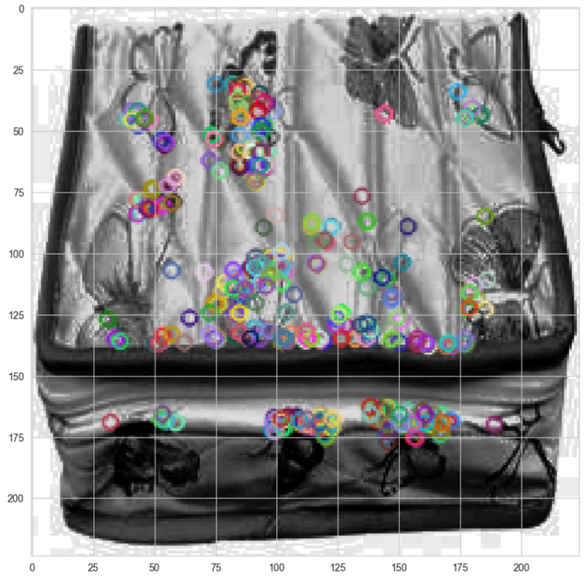
```
In 61  1  # test méthode SIFT
       2  test_img('sift',100)
Executed at 2024.03.12 09:26:01 in 157ms
```




```
Le nombre des features extraits est de :  (593, 128)
```

ORB

```
In 62  1  # test méthode ORB
       2  test_img('orb',100)
Executed at 2024.03.12 09:26:01 in 323ms
```



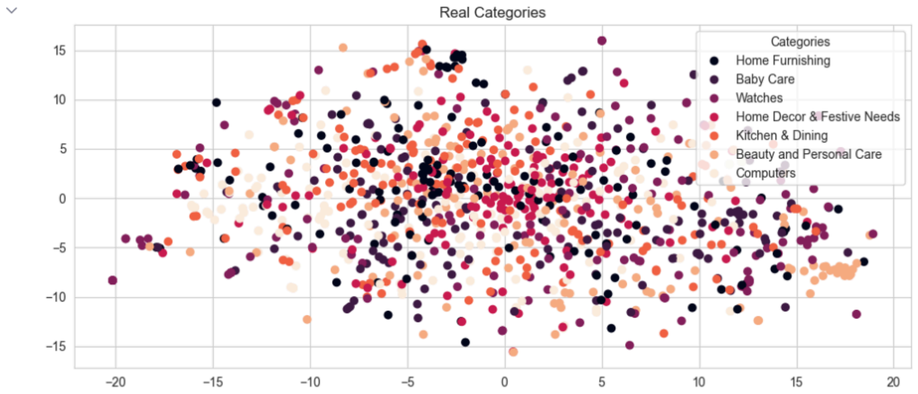

```
Le nombre des features extraits est de :  (410, 32)
```

Traitement images

Extractions des features

Sift

ARI : 0.02754513474539411 Temps d'exécution : 2.0



Traitement images

VGG19

```
ARI :  0.5523022007630232 Temps d'exécution :  2.0
```

Modélisation

Data Generator

In 16
```python
# Créer un générateur d'images
datagen = ImageDataGenerator(rotation_range=20,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True,
                             rescale=1./255)
```
Executed at 2024.03.12 09:32:11 in 1ms

In 17
```python
# création df pour train, test et validation
df_train = pd.DataFrame({"image": X_train, "label_categ": y_train})
df_test = pd.DataFrame({"image": X_test, "label_categ": y_test})
df_val = pd.DataFrame({"image": X_val, "label_categ": y_val})
```
Executed at 2024.03.12 09:32:11 in 2ms

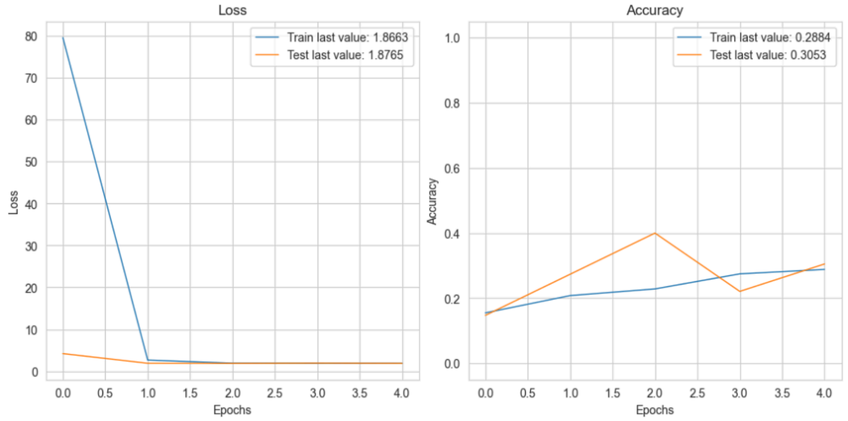
In 18
```python
# génération images pour set enrainement
train_generator = datagen.flow_from_dataframe(dataframe=df_train,
                                              x_col="image",
                                              y_col="label_categ",
                                              target_size=(224, 224),
                                              batch_size=32,
                                              class_mode="categorical")

# Générer des images pour le set de test
test_generator = datagen.flow_from_dataframe(dataframe=df_test,
                                             x_col="image",
                                             y_col="label_categ",
                                             target_size=(224, 224),
                                             batch_size=32,
                                             class_mode="categorical")

# générer images pour set de validation
val_generator = datagen.flow_from_dataframe(dataframe=df_val,
                                            x_col="image",
                                            y_col="label_categ",
                                            target_size=(224, 224),
                                            batch_size=32,
                                            class_mode="categorical")
```
Executed at 2024.03.12 09:32:11 in 11ms

Modélisation

CNN

```
In 20
# Créer le modèle
model_cnn = Sequential()
model_cnn.add(Conv2D(64, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model_cnn.add(MaxPooling2D((2, 2)))
model_cnn.add(Flatten())
model_cnn.add(Dense(128, activation='relu'))
model_cnn.add(Dense(7, activation='softmax'))

# Compiler le modèle
model_cnn.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

# Afficher un résumé du modèle
model_cnn.summary()

checkpoint = ModelCheckpoint("../models/best_model_cnn.h5", monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)

# Entraîner le modèle sur les données d'entraînement
history_cnn = model_cnn.fit(train_generator,
                            batch_size=32,
                            epochs=epochs,
                            validation_data=val_generator,
                            validation_batch_size=32,
                            callbacks=[early_stop, checkpoint])

# Charger le meilleur modèle
best_model_cnn = load_model("../models/best_model_cnn.h5")

# Temps d'exécution
end_time = time.time()
execution_time = end_time - start_time
print("Temps d'exécution de la cellule : ", execution_time, " secondes")
```

```
Validation Accuracy: 0.2842
Test Accuracy: 0.3000
```



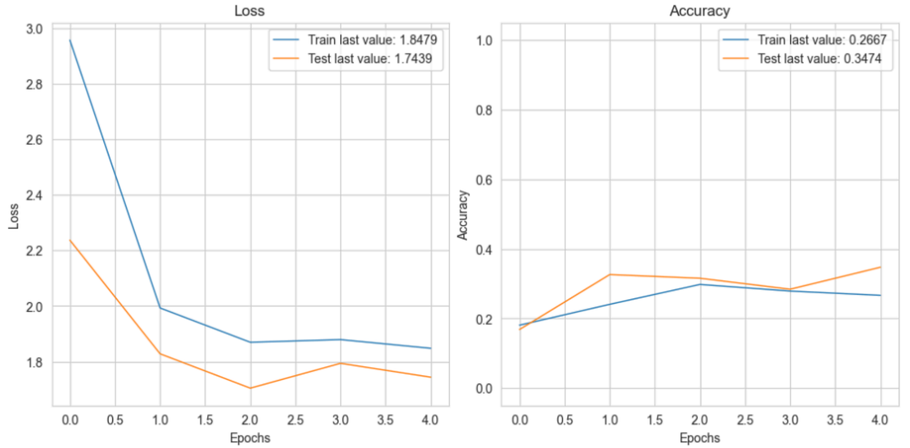
Modélisation

RESNET 50

```
In 24
```

```python
# Définir le tenseur d'entrée
inputs = Input(shape=(224, 224, 3))

# Charger le modèle ResNet50 pré-entraîné avec des poids pré-entraînés
resnet_base = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs)

# Geler les couches pré-entraînées
for layer in resnet_base.layers:
    layer.trainable = False

# Ajouter des couches supérieures personnalisées pour ma tâche spécifique
x = MaxPooling2D(pool_size=(2, 2))(resnet_base.output)  # Couche de mise en commun
x = Flatten()(x)  # Couche de dépliement
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)  # Couche de sortie avec 7 neurones et activation softmax

# Créer le modèle final
model_resnet = Model(inputs=inputs, outputs=predictions)

# Compiler le modèle
model_resnet.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Afficher un résumé du modèle
model_resnet.summary()

# Point de contrôle pour sauvegarder le meilleur modèle
checkpoint = ModelCheckpoint("../models/best_model_resnet50.h5", monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)
# Entraîner le modèle sur les données d'entraînement
history_resnet = model_resnet.fit(train_generator,
                                  batch_size=32,
                                  epochs=epochs,
                                  validation_data=val_generator,
                                  validation_batch_size=32,
                                  callbacks=[early_stop, checkpoint])

# Charger le meilleur modèle basé sur la précision de validation
best_model_resnet = load_model("../models/best_model_resnet50.h5")
# Calculer et afficher le temps d'exécution
end_time = time.time()
execution_time = end_time - start_time
print("Temps d'exécution de la cellule : ", execution_time, " secondes")
```

```
Validation Accuracy: 0.3158
Test Accuracy: 0.3182
```

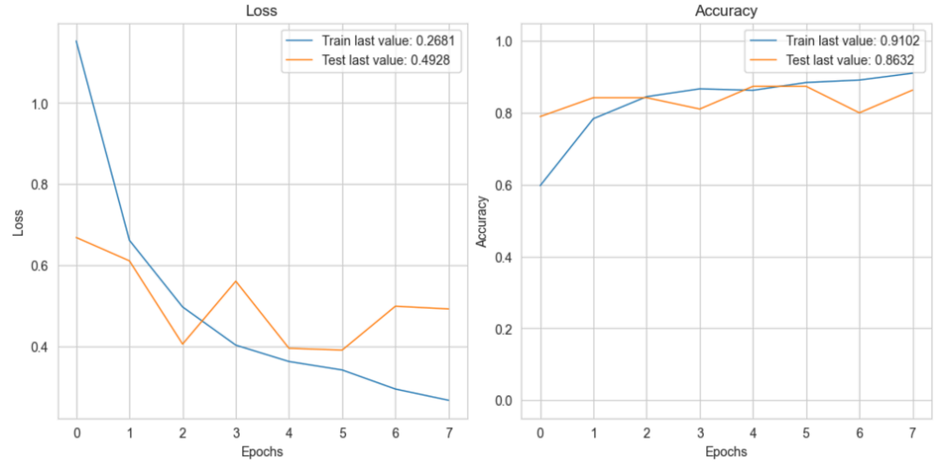


Modélisation

Inception

```
In 22
# Définir le tenseur d'entrée
inputs = Input(shape=(224, 224, 3))

# Charger le modèle InceptionV3 pré-entraîné avec les poids pré-entraînés
inception_base = InceptionV3(weights='imagenet', include_top=False, input_tensor=inputs)

# Geler les couches pré-entraînées
for layer in inception_base.layers:
    layer.trainable = False

# Ajouter des couches supérieures personnalisées pour votre tâche spécifique
x = GlobalAveragePooling2D()(inception_base.output)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)  # Couche de sortie avec 7 neurones et activation softmax

# Créer le modèle final
model_inception = Model(inputs=inputs, outputs=predictions)

# Compiler le modèle
model_inception.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Afficher un résumé du modèle
model_inception.summary()

checkpoint = ModelCheckpoint("../models/best_model_inception.h5", monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)

# Entraîner le modèle sur les données d'entraînement
history_inception = model_inception.fit(train_generator,
                                        batch_size=32,
                                        epochs=epochs,
                                        validation_data=val_generator,
                                        validation_batch_size=32,
                                        callbacks=[early_stop, checkpoint])

# Charger le meilleur modèle basé sur la précision de validation
best_model_inception = load_model("../models/best_model_inception.h5")

# Calculer et afficher le temps d'exécution
end_time = time.time()
execution_time = end_time - start_time
print("Temps d'exécution de la cellule : ", execution_time, " secondes")
```

```
Validation Accuracy: 0.8737
Test Accuracy: 0.8182
```



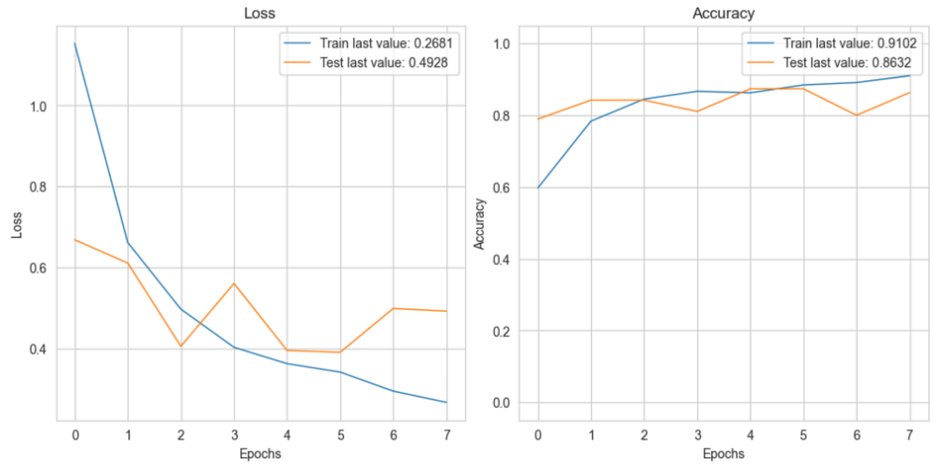
Modélisation

Xception

```
In 26
```

```python
# Définir le tenseur d'entrée
inputs = Input(shape=(224, 224, 3))

# Charger le modèle Xception pré-entraîné avec les poids pré-entraînés
xception_base = Xception(weights='imagenet', include_top=False, input_tensor=inputs)

# Geler les couches pré-entraînées
for layer in xception_base.layers:
    layer.trainable = False

# Ajouter des couches supérieures personnalisées pour votre tâche spécifique
x = GlobalAveragePooling2D()(xception_base.output)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)

predictions = Dense(7, activation='softmax')(x)  # Couche de sortie avec 7 neurones et activation softmax

# Créer le modèle final
model_xception = Model(inputs=inputs, outputs=predictions)

# Compiler le modèle
model_xception.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Afficher un résumé du modèle
model_xception.summary()

checkpoint = ModelCheckpoint("../models/best_model_xception.h5", monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)

# Entraîner le modèle sur les données d'entraînement
history_xception = model_xception.fit(train_generator,
                                      batch_size=32,
                                      epochs=epochs,
                                      validation_data=val_generator,
                                      validation_batch_size=32,
                                      callbacks=[early_stop, checkpoint])

# Charger le meilleur modèle basé sur la précision de validation
best_model_xception = load_model("../models/best_model_xception.h5")

# Calculer et afficher le temps d'exécution
end_time = time.time()
execution_time = end_time - start_time
print("Temps d'exécution de la cellule : ", execution_time, " secondes")
```

```
Validation Accuracy: 0.8737
Test Accuracy: 0.8318
```



Modélisation

Xception

Inception

```
In 28  1  plot_bad_images(best_model_inception, val_generator, original_labels)
Executed at 2024.03.12 09:42:47 in 3s 211ms

3/3 [==============================] - 3s 632ms/step
```


```
In 29  1  plot_bad_images(best_model_xception, val_generator, original_labels)
Executed at 2024.03.12 09:42:52 in 4s 346ms

3/3 [==============================] - 4s 1s/step
```

Modélisation

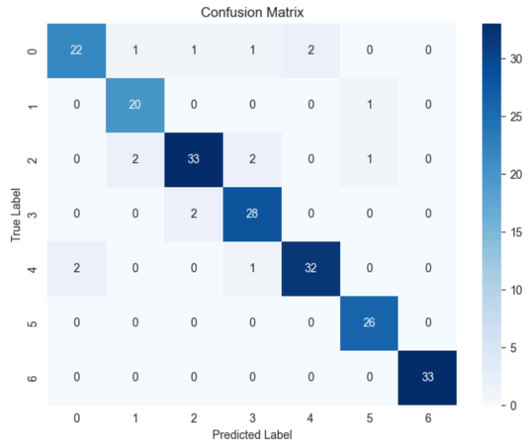
Modèle combiné

```
In 30  1  train_predict_evaluate(data, best_model_xception, path,classifier_type='random_forest')
Executed at 2024.03.12 09:44:05 in 1m 13s 189ms
```

Matrice de confusion :

Rapport de classification :

	precision	recall	f1-score	support
0	0.92	0.81	0.86	27
1	0.87	0.95	0.91	21
2	0.92	0.87	0.89	38
3	0.88	0.93	0.90	30
4	0.94	0.91	0.93	35
5	0.93	1.00	0.96	26
6	1.00	1.00	1.00	33
accuracy			0.92	210
macro avg	0.92	0.93	0.92	210
weighted avg	0.92	0.92	0.92	210

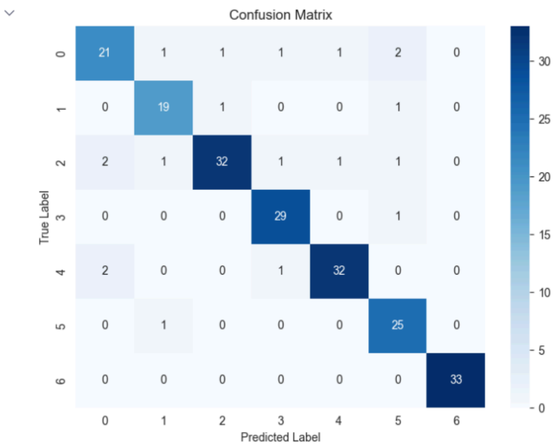



```
In 31  1  train_predict_evaluate(data, best_model_xception,path, classifier_type='xgboost')
Executed at 2024.03.12 09:45:15 in 1m 10s 582ms
```

Matrice de confusion :

Rapport de classification :

	precision	recall	f1-score	support
0	0.84	0.78	0.81	27
1	0.86	0.90	0.88	21
2	0.94	0.84	0.89	38
3	0.91	0.97	0.94	30
4	0.94	0.91	0.93	35
5	0.83	0.96	0.89	26
6	1.00	1.00	1.00	33
accuracy			0.91	210
macro avg	0.90	0.91	0.91	210
weighted avg	0.91	0.91	0.91	210



Script

```python
# URL de l'API
url = "https://edamam-food-and-grocery-database.p.rapidapi.com/api/food-database/v2/parser"

# Paramètres pour l'API
querystring = {"ingr":"champagne"}

# En-têtes de la requête pour l'API
headers = {
    "X-RapidAPI-Key": "279d3e1367msh23732bcf30a5598p1ec127jsn4ba6087d5aed",
    "X-RapidAPI-Host": "edamam-food-and-grocery-database.p.rapidapi.com"
}

# requête pour l'API et réponse
response = requests.request( method: "GET", url, headers=headers, params=querystring)


# Charger la réponse en format JSON
json_data = json.loads(response.text)


# Extraire les données pertinentes et les structurer en une liste de dict
csv_data = []
for item in json_data['hints']:
    row = {
        'foodId': item['food']['foodId'],
        'label': item['food']['label'],
        'category': item['food']['category'],
        'foodContentsLabel': item['food'].get('foodContentsLabel'),
        'image': item['food'].get('image')
    }
    csv_data.append(row)
```

Script

```python
# sauvegarde des données CSV dans un fichier
with open('output.csv', 'w', newline='') as csv_file:
    fieldnames = ['foodId', 'label', 'category', 'foodContentsLabel', 'image']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()
    for row in csv_data:
        writer.writerow(row)

# Charger les données dans un dataframe et sélectionner colonnes souhaitées
df = pd.read_csv( filepath_or_buffer: 'output.csv', nrows=10, usecols=['foodId', 'label', 'category', 'foodContentsLabel', 'image'])

# Exporter le dataframe en fichier CSV
df.to_csv( path_or_buf: 'output_2.csv', index=False)

# Constat du fichier
df
```

10 rows | 10 rows × 5 columns

	foodId	label	category	foodConten ...	image
	Missing 0 Count 10 10 Unique values	Missing 0 Count 10 7 Unique values	Missing 0 Count 10 Packaged … 50% Generic m… 40% Generic f… 10%	Missing 1 Count 9 10 Unique values	Missing 7 Count 3 nan 70% https://w… 10% Other 20%
0	food_a656mk2a5dmq…	Champagne	Generic foods	NaN	https://www.edama…
1	food_b753ithamdb8…	Champagne Vinaigr…	Packaged foods	OLIVE OIL; BALSAM…	NaN
2	food_b3dyababjo54…	Champagne Vinaigr…	Packaged foods	INGREDIENTS: WATE…	https://www.edama…
3	food_a9e0ghsamvoc…	Champagne Vinaigr…	Packaged foods	CANOLA AND SOYBEA…	NaN
4	food_an4jjueaucpu…	Champagne Vinaigr…	Packaged foods	WATER; CANOLA AND…	NaN
5	food_bmu5dmkazwuv…	Champagne Dressin…	Packaged foods	SOYBEAN OIL; WHIT…	https://www.edama…
6	food_alpl44taoyv1…	Champagne Butterc…	Generic meals	sugar; butter; sh…	NaN

Merci pour votre attention