P6_02_notebook

September 8, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.graph_objects as go
import plotly.express as px
import seaborn as sns
import json
sns.set()
```

[2]:
```
users = []
with open('yelp_academic_dataset_review.json', encoding="utf8") as fl:
    for i, line in enumerate(fl):
        users.append(json.loads(line))
df = pd.DataFrame(users)
df.head()
print(df.shape)
```

```
(8021122, 9)
```

[3]:
```
df.head(2)
```

[3]:
```
                review_id                 user_id             business_id  \
0  xQY8N_XvtGbearJ5X4QryQ  OwjRMXRC0KyPrIlcjaXeFQ  -MhfebM0QIsKt87iDN-FNw
1  UmFMZ8PyXZTY2QcwzsfQYA  nIJD_7ZXHq-FX8byPMOkMQ  lbrU8StCq3yDfr-QMnGrmQ

   stars  useful  funny  cool  \
0    2.0       5      0     0
1    1.0       1      1     0

                                                text                 date
0  As someone who has worked with many museums, I…  2015-04-15 05:21:16
1  I am actually horrified this place is still in…  2013-12-07 03:16:52
```

[4]:
```
df.describe()
```

[4]:
```
              stars        useful         funny          cool
count  8.021122e+06  8.021122e+06  8.021122e+06  8.021122e+06
mean   3.703575e+00  1.322882e+00  4.596423e-01  5.745620e-01
```

```
std    1.490486e+00  3.550831e+00  2.188143e+00  2.476906e+00
min    1.000000e+00 -1.000000e+00  0.000000e+00 -1.000000e+00
25%    3.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%    4.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%    5.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00
max    5.000000e+00  1.122000e+03  9.760000e+02  5.020000e+02
```

```
[19]: df['date'].max()
```

```
[19]: '2019-12-13 15:51:19'
```

0.0.1 Extraction d’un échantillon pour analyse

```
[6]: sampled_index = np.random.choice(range(np.shape(df)[0]), 100000, replace=False)
     sample = df.iloc[sampled_index]
```

```
[7]: sample.describe()
```

```
[7]:                stars         useful         funny           cool
     count  100000.000000  100000.000000  100000.00000  100000.000000
     mean        3.699920       1.331750       0.47183       0.586520
     std         1.492901       3.465976       2.20653       2.549506
     min         1.000000       0.000000       0.00000       0.000000
     25%         3.000000       0.000000       0.00000       0.000000
     50%         4.000000       0.000000       0.00000       0.000000
     75%         5.000000       1.000000       0.00000       0.000000
     max         5.000000     133.000000     269.00000     129.000000
```

```
[25]: df['business_id'].value_counts().mean();
      df['business_id'].value_counts().std()
```

```
[25]: 127.46300694012055
```

1 Analyse exploratoire des données

1.0.1 Détection des valeurs manquantes

```
[8]: 1-sample.isna().mean()
```

```
[8]: review_id      1.0
     user_id        1.0
     business_id    1.0
     stars          1.0
     useful         1.0
```

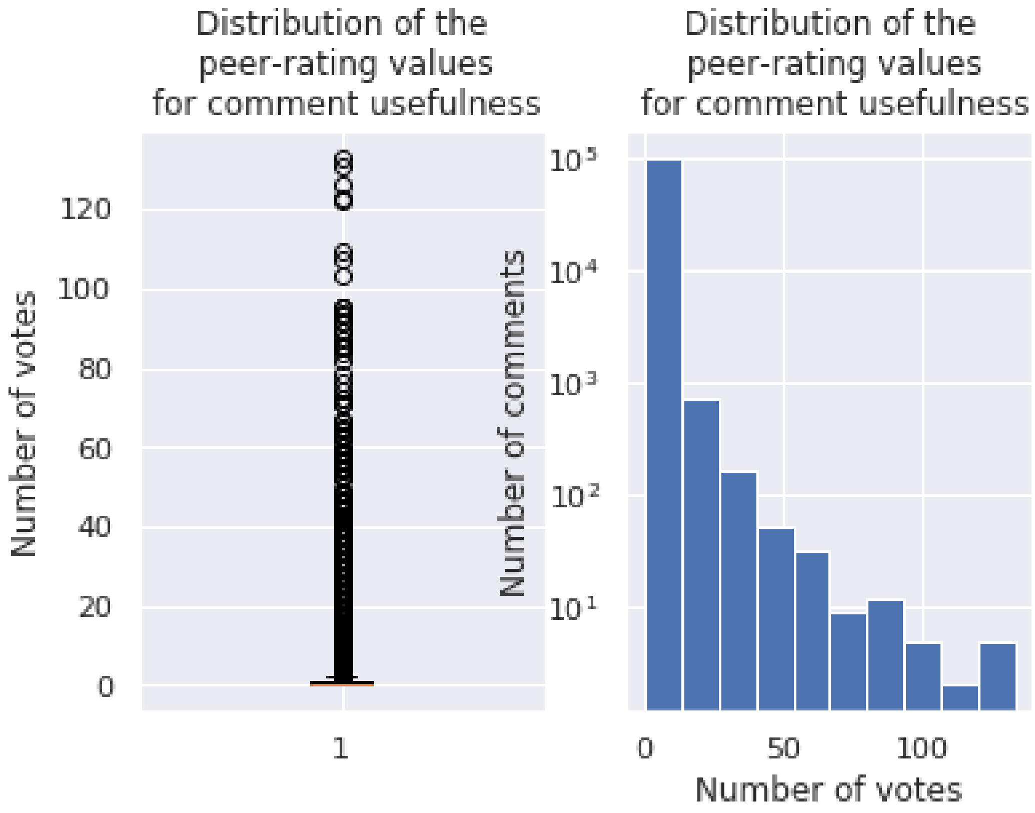
```
funny          1.0
cool           1.0
text           1.0
date           1.0
dtype: float64
```

1.0.2 Analyse univariée

Histogrammes des notes de qualité de commentaires - “useful”

[9]:

```
fig1 = plt.subplot(121);
plt.boxplot(sample['useful']);
plt.title("Distribution of the\n peer-rating values\n for comment usefulness");
plt.ylabel("Number of votes");
plt.subplot(122);
plt.hist(sample['useful']);
plt.title("Distribution of the\n peer-rating values\n for comment usefulness");
plt.xlabel("Number of votes");
plt.ylabel("Number of comments");
plt.semilogy();
plt.savefig('./Useful_histo.png', bbox_inches='tight')
```

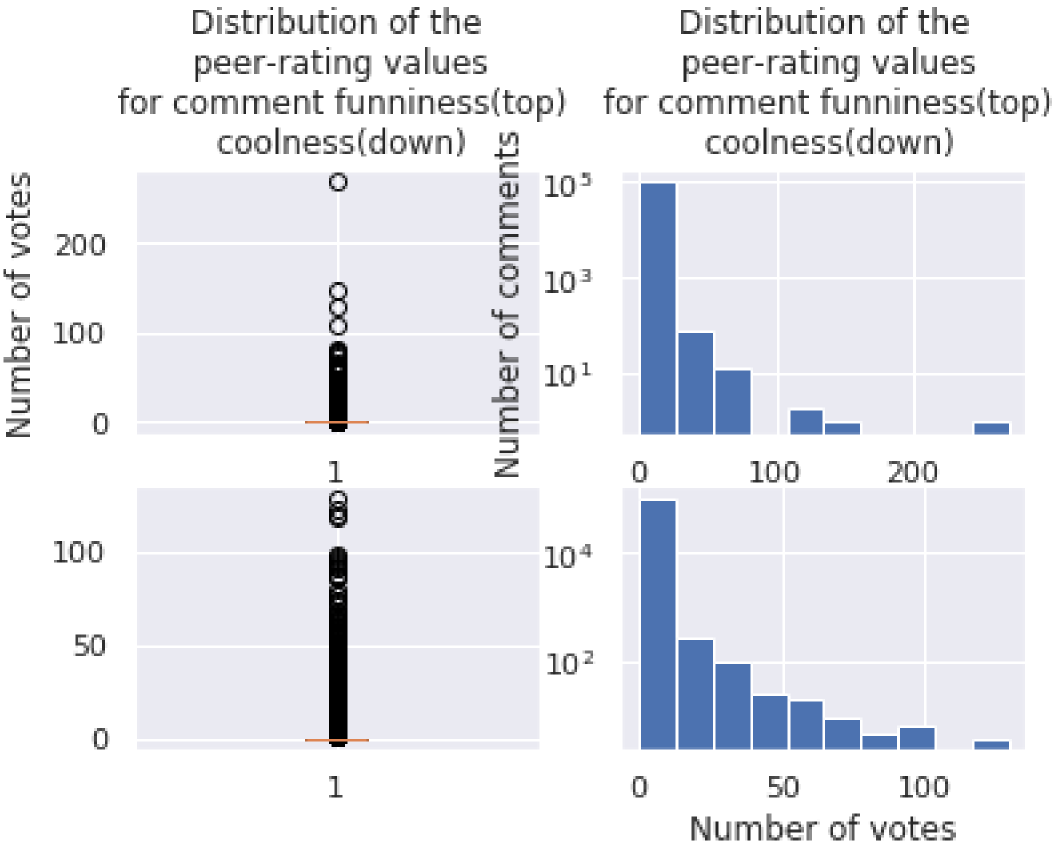


Histogrammes des notes de qualité de commentaires - “funny” & “cool”

[10]:

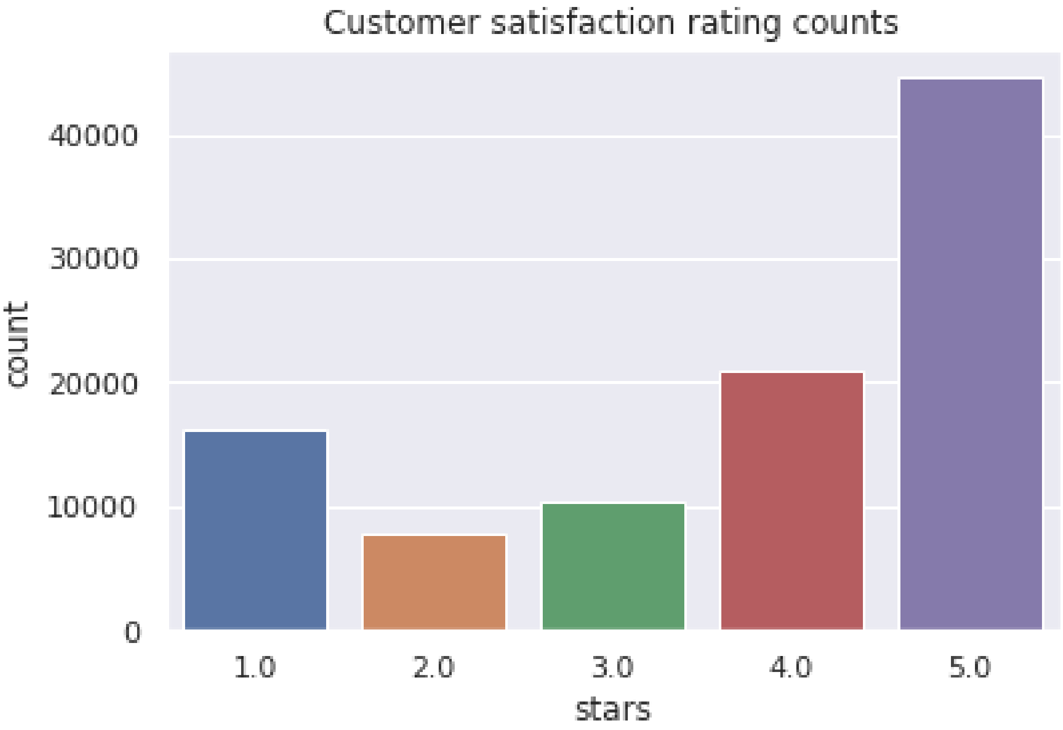
```
fig2 = plt.subplot(221);
plt.boxplot(sample['funny']);
plt.title("Distribution of the\n peer-rating values\n for comment␣
 ↪funniness(top)\n coolness(down)");
plt.ylabel("Number of votes");
plt.subplot(222);
plt.hist(sample['funny']);
plt.title("Distribution of the\n peer-rating values\n for comment␣
 ↪funniness(top)\n coolness(down)");
# plt.xlabel("Number of votes");
plt.ylabel("Number of comments");
plt.semilogy();

plt.subplot(223);
plt.boxplot(sample['cool']);
# plt.ylabel("Number of votes");
plt.subplot(224);
plt.hist(sample['cool']);
plt.xlabel("Number of votes");
# plt.ylabel("Number of comments");
plt.semilogy();
plt.savefig('./funny_cool_histo.png', bbox_inches='tight')
```



Repartition des notes de qualité de service - "stars"

```
[11]: uneliste = sample['stars'].value_counts().index.tolist()
      ax = sns.countplot(x='stars',
                         data=sample[sample['stars'].isin(uneliste)])
      plt.title('Customer satisfaction rating counts');
      plt.savefig('./satisfaction_rating.png', bbox_inches='tight')
```



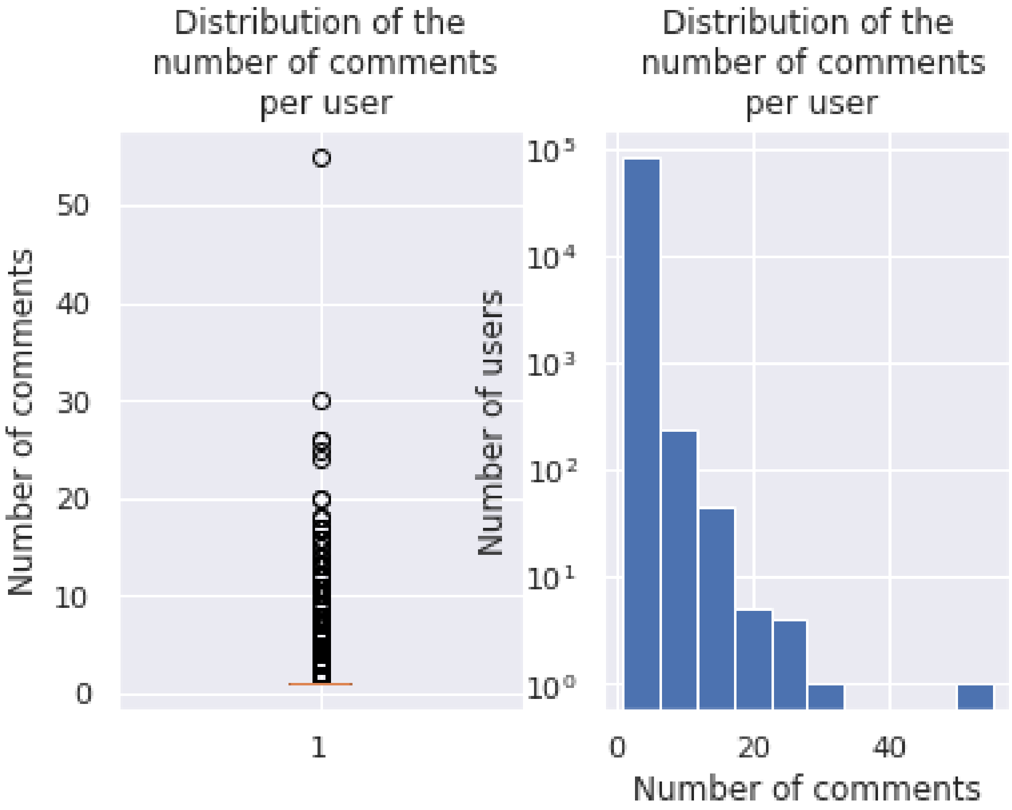
Histogramme du nombre de commentaires par utilisateurs

```
[12]: com_per_user = sample.groupby(['user_id']).count()
```

```
[13]: com_per_user['review_id'].value_counts();
```

```
[14]: fig2 = plt.subplot(121);
      plt.boxplot(com_per_user['review_id']);
      plt.title("Distribution of the\n number of comments\n per user");
      plt.ylabel("Number of comments");
      plt.subplot(122);
      plt.hist(com_per_user['review_id'])
      plt.title("Distribution of the\n number of comments\n per user");
      plt.xlabel("Number of comments");
      plt.ylabel("Number of users");
      plt.semilogy();
      # plt.subplot(111);
      # plt.figure(figsize=(6,6))
      # plt.pie(com_per_user['review_id'].value_counts().sort_values(ascending=False).
       ↪head(12))
      # plt.title("Distribution of the\n number of comments\n per user");
      # plt.legend(com_per_user['review_id'].value_counts().
       ↪sort_values(ascending=False).head(12).index.to_list())
```

```
plt.savefig('./comments_peruser.png', bbox_inches='tight')
```



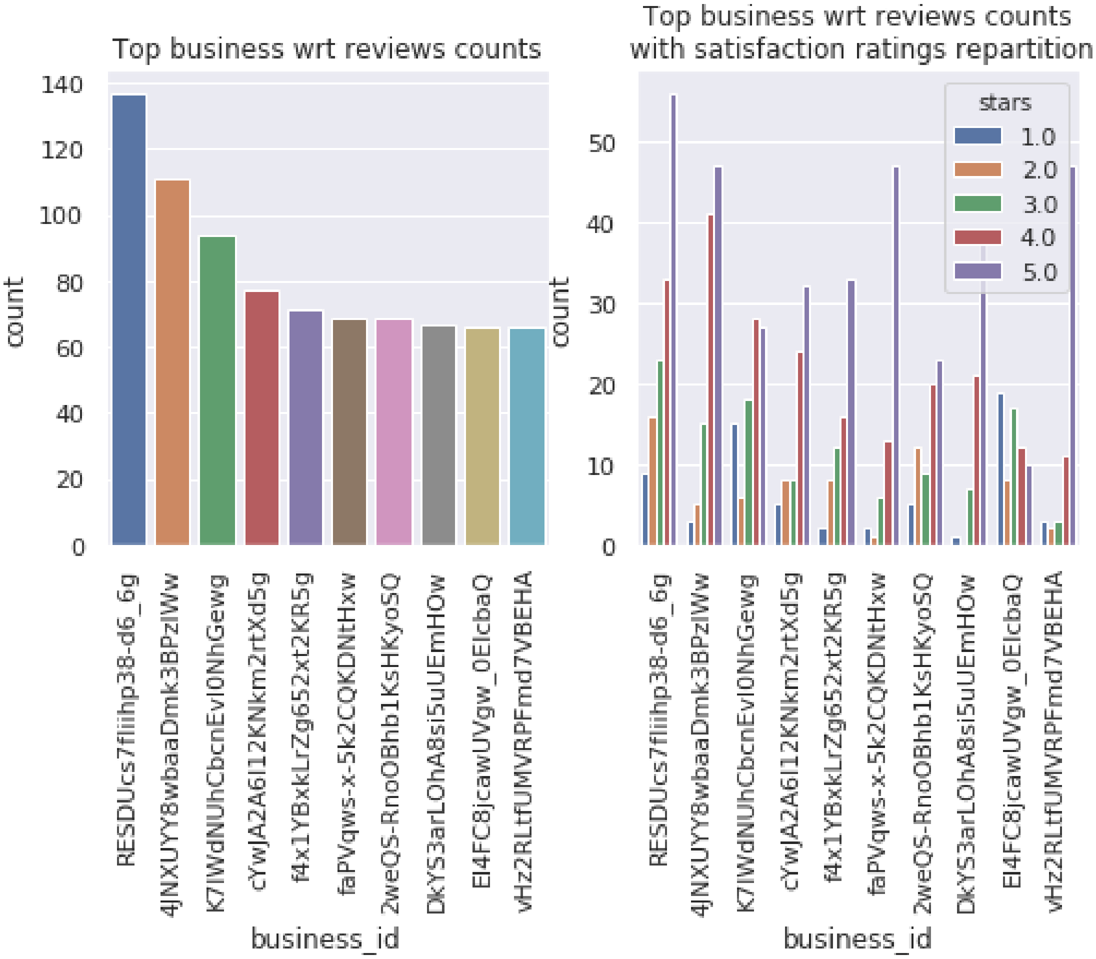
Répartition du nombre de commentaires par business

[15]:
```
com_per_business = sample[['review_id','business_id']].groupby(['business_id']).
 ↪count()
com_per_business.index.to_list();
```

[16]:
```
uneliste_combus = sample['business_id'].value_counts().
 ↪sort_values(ascending=False).head(10).index.to_list()

plt.figure(figsize=(8,4))
plt.subplot(121)
ax = sns.countplot(x='business_id',
                   data=sample[sample['business_id'].isin(uneliste_combus)],
                   order= uneliste_combus);
ax.set_xticklabels(ax.get_xticklabels(), rotation = 90);
plt.title('Top business wrt reviews counts');
plt.subplot(122)
ax = sns.countplot(x='business_id',
                    data=sample[sample['business_id'].isin(uneliste_combus)],
                    order= uneliste_combus,
                    hue='stars');
ax.set_xticklabels(ax.get_xticklabels(), rotation = 90);
plt.title('Top business wrt reviews counts\n with satisfaction ratings␣
 ↪repartition');

plt.savefig('./com_per_business.png', bbox_inches = 'tight')
```

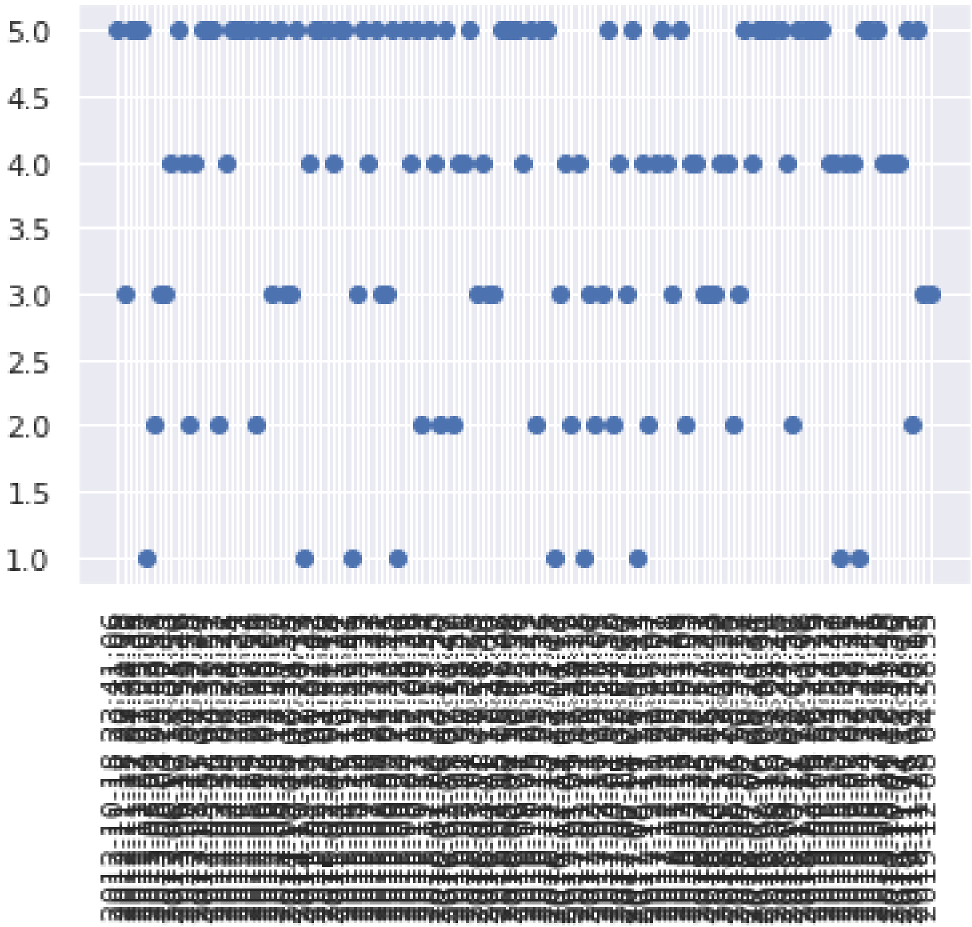


[28]:

```
top_reviewed_business = com_per_business.sort_values(by='review_id',␣
 ↪ascending=False).head(5).index.to_list()
```

[29]:

```
sorted_date = sample[sample['business_id']==top_reviewed_business[0]]['date'].
 ↪sort_values(ascending=True)
fig, ax = plt.subplots()
plt.plot_date(x= sorted_date,
              y=␣
↪sample[sample['business_id']==top_reviewed_business[0]]['stars'])
ax.xaxis.set_tick_params(rotation=90)
plt.show()
```



[30]:
```
def rating_time_series(business_id):
    df = sample[sample['business_id']==business_id][['date','stars']].
↪sort_values(ascending=True, by='date')
    df['date'] = df['date'].apply(lambda x:pd.to_datetime(x))
    df['date_s'] = (df['date']-df['date'].min()).apply(lambda x:x.
↪total_seconds())

    df.set_index('date')
    return df
```

[31]:
```
business1_ts = rating_time_series(top_reviewed_business[0])
# business1_ts.set_index('date');
```

```
business1_ts.iloc[0:50];
```

[36]:

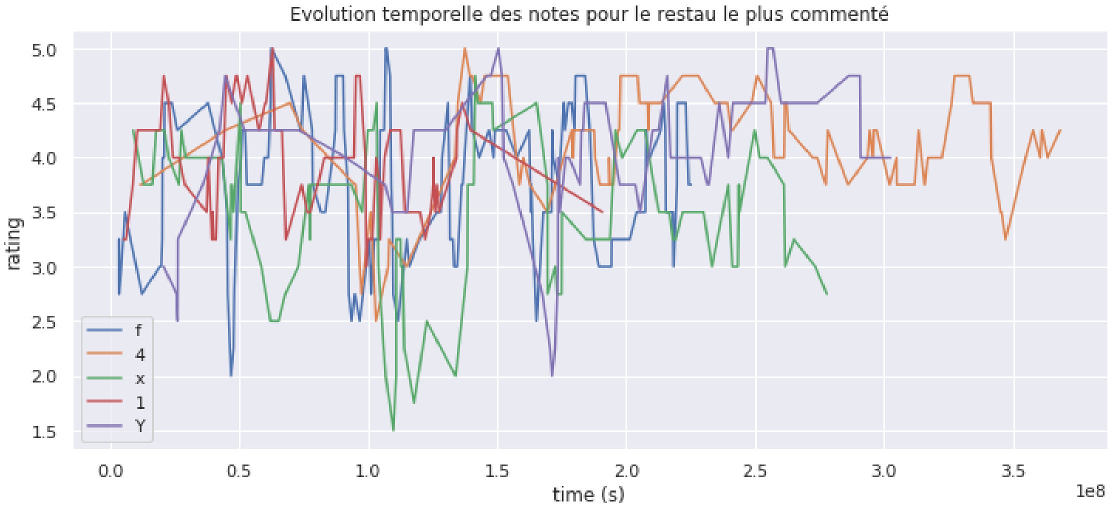
```
def plot_stars_series(business_ids):
    plt.figure(figsize=(12,5))
    plt.xlabel('time (s)')
    plt.ylabel('rating')
    plt.title('Evolution temporelle des notes pour le restau le plus commenté')
#     plt.legend(business_ids)
    for business_id in business_ids:
        business_ts = rating_time_series(business_id)
#         plt.plot(business_ts['date_s'], business_ts['stars']);
        plt.plot(business_ts['date_s'], business_ts.rolling(4,␣
 ↪on='stars')['stars'].mean());
        plt.legend(business_id)


    plt.savefig('./top5_rating_time.png', bbox_inches = 'tight')
    plt.show()


# plt.plot(business1_ts['date_s'], business1_ts['stars']);
```

[37]:

```
plot_stars_series(top_reviewed_business)
```



2 Traitement des commentaires utilisateurs

[1]:

```python
import json
import timeit
from itertools import combinations


import nltk
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.probability import FreqDist
import spacy
from langdetect import detect
from sklearn.feature_extraction.text import CountVectorizer;
from sklearn.feature_extraction.text import TfidfTransformer;
from sklearn.decomposition import NMF;
from sklearn.preprocessing import normalize;

import gensim as gsm
from gensim.models.word2vec import Word2Vec
from gensim.models import FastText
from gensim.models import nmf
from gensim.models.coherencemodel import CoherenceModel
from gensim import corpora

import pickle

import streamlit as st
```

```
[nltk_data] Downloading package wordnet to /home/erbadi/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /home/erbadi/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[2]:

```python
users = []
with open('yelp_academic_dataset_review.json', encoding="utf8") as fl:
    for i, line in enumerate(fl):
        users.append(json.loads(line))
df = pd.DataFrame(users)
df.head()
print(df.shape)
```

```
(8021122, 9)
```

La base de données review a été chargée en 57 secondes

[3]:

```python
df.head(2)
```

```
[3]:                review_id                 user_id             business_id  \
     0  xQY8N_XvtGbearJ5X4QryQ  OwjRMXRC0KyPrIlcjaXeFQ  -MhfebM0QIsKt87iDN-FNw
     1  UmFMZ8PyXZTY2QcwzsfQYA  nIJD_7ZXHq-FX8byPM0kMQ  lbrU8StCq3yDfr-QMnGrmQ

        stars  useful  funny  cool  \
     0    2.0       5      0     0
     1    1.0       1      1     0

                                                     text                 date
     0  As someone who has worked with many museums, I…  2015-04-15 05:21:16
     1  I am actually horrified this place is still in…  2013-12-07 03:16:52
```

```
[4]: df.describe()
```

```
[4]:               stars        useful         funny          cool
     count  8.021122e+06  8.021122e+06  8.021122e+06  8.021122e+06
     mean   3.703575e+00  1.322882e+00  4.596423e-01  5.745620e-01
     std    1.490486e+00  3.550831e+00  2.188143e+00  2.476906e+00
     min    1.000000e+00 -1.000000e+00  0.000000e+00 -1.000000e+00
     25%    3.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
     50%    4.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
     75%    5.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00
     max    5.000000e+00  1.122000e+03  9.760000e+02  5.020000e+02
```

```
[5]: df['useful'].value_counts()
```

```
[5]:  0      4337407
      1      1725147
      2       786918
      3       406057
      4       231336
               …
      225          1
      223          1
      222          1
      217          1
     -1            1
     Name: useful, Length: 266, dtype: int64
```

2.1 Sélection des commentaires négatifs

```
[6]: df_usef = df[df['stars']<=1]
```

```
[7]: df_usef.iloc[10:30]['text']
```

[7]:

```
53     Passed by here after we went to raijin ramen a…
58     I can't tell you how angry I am right now.  I …
59     Worst pedicure ever. First, waited over a half…
67     We drive by Fruits and Roots almost daily and …
69     I had an oil change at the 15515 N Scottsdale …
70     The absolute WORST apartment complex I have ev…
72     I ordered a pizza at 4:49. Got an email that s…
74     I went into this store yesterday and it was ho…
107    I went up too my storage unit at the Uhaul fac…
110    This place is clean and a great price the othe…
118    The worst hotel experience I have had…we cam…
128    I went to a Spring Training game here on Monda…
129    We got there by 5:30pm and most of the dishes …
138    I purchased the Groupon and I honestly shouldn…
149    I used to go this office, but as other reviewe…
152    After being a regular at Veggie House i though…
159    $28 for a 15 minutes pedicure with used tools?…
167    the most low quality catering in persian catag…
168    Table of 3 on Saturday March 25 around 8pm.. W…
171    While Cox maybe overprice and give crappy cust…
Name: text, dtype: object
```

[8]:

```python
print(df_usef.shape)
```

```
(1283897, 9)
```

[9]:

```python
df_usef.describe()
```

[9]:

```
           stars        useful         funny          cool
count  1283897.0  1.283897e+06  1.283897e+06  1.283897e+06
mean         1.0  1.932554e+00  5.484116e-01  2.172752e-01
std          0.0  4.632479e+00  2.049865e+00  1.161310e+00
min          1.0  0.000000e+00  0.000000e+00  0.000000e+00
25%          1.0  0.000000e+00  0.000000e+00  0.000000e+00
50%          1.0  1.000000e+00  0.000000e+00  0.000000e+00
75%          1.0  2.000000e+00  1.000000e+00  0.000000e+00
max          1.0  1.122000e+03  7.860000e+02  5.020000e+02
```

2.2 Sélection des commentaires en anglais

[10]:

```python
import os
import requests

def teledetected_lang(extract):
    #!/usr/bin/python3.7
    # -*-coding:utf-8 -*
    # pprint is used to format the JSON response
    from pprint import pprint

    #### Building the request body
    #Header values
    subscription_key = open("subkey.txt","rt").readline() #le fichier subkey.txt doit être créé dans le même répertoire que le script, avec comme une ligne unique contenant la clé d'abonnement
    endpoint = "https://ocproject.cognitiveservices.azure.com"
    language_api_url = endpoint + "/text/analytics/v3.0/languages"

    #Aggregating the request header
    headers = {"Ocp-Apim-Subscription-Key": subscription_key}

    ####Interacting with the user
    documents = {"documents": [{"id": "1", "text": extract[0:100]}
                              ]
                }
    #### Requesting and formatting the response
    response = requests.post(language_api_url, headers=headers, json=documents)
    languages = response.json()

    return languages

def detected_lang(comment):
    exclusive_common_words = [' the ', ' am ', ' and ', ' is ', ' more ', ' it ', ' in ']
    count = 0
    for string in exclusive_common_words:
        if string in comment:
            count=count+1

    if count >=2:  #S'il existe plus de 2 de ces mots, alors le commentaire est validé comme en anglais
        isenglish = True
    else:
        isenglish = False

    return isenglish
```

2.2.1 Detection of english comments

[11]:
```python
#tested_length = np.ceil(len(string)/2)
start_time = timeit.default_timer()
df_usef['isenglish'] = df_usef['text'].apply(lambda x:detected_lang(x))
elapsed = timeit.default_timer() - start_time
```

```
/home/erbadi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[12]:
```python
df_usef['isenglish'].value_counts()
```

[12]:
```
True     1180367
False     103530
Name: isenglish, dtype: int64
```

2.2.2 Selection of english comments

[13]:
```python
df_usef = df_usef[df_usef['isenglish']==True]
```

[14]:
```python
df_usef.shape
```

[14]:
```
(1180367, 10)
```

2.3 Binarisation des notes

Les notes sont binarisées en note négative (value = 0), et notes positives (value = 1)

[15]:
```python
df_usef['positif'] = df_usef['stars'].apply(lambda x: 1 if x>=3 else 0)
```

2.4 Nettoyage du texte

2.4.1 Tokenisation, stopwords removal and lemmatization

[16]:

```
from spacy.lang.en import English
from spacy.lang.en.stop_words import STOP_WORDS
```

[17]:

```
custom_stopwords = ['would', 'got', 'could', 'first']
```

[18]:

```
def cleaned_text(text):

    #Tokenization step
    tokenizer = nltk.RegexpTokenizer(r'\w+')
    tokenized_txt = tokenizer.tokenize(text)

    #Stopwords removal step
    stopwords = nltk.corpus.stopwords.words('english')
    stopwords = stopwords + custom_stopwords

    filtered_words = [word for word in tokenized_txt if word.lower() not in␣
 ↪stopwords]

    #Lemmatization step
    lemmatizer = nltk.stem.WordNetLemmatizer()
    lemmatized_txt = []
    for word in filtered_words:
        lemmatized_txt.append(lemmatizer.lemmatize(word))

#    res = [' '.join(token) for token in lemmatized_txt]
    res = [' '.join(lemmatized_txt)]
    return res
```

[19]:

```
test = cleaned_text('I am trying to test the function that I wrote to')
test
```

[19]:

```
['trying test function wrote']
```

[20]:

```
type(nltk.corpus.stopwords.words('english'))
```

[20]:

```
list
```

2.4.2 Extraction d'une partie des commentaires lemmatisés

[21]:

```
slice = df_usef.iloc[0:10000]
```

[22]:

```
start_time = timeit.default_timer()
slice['text_norm'] = slice['text'].apply(lambda x: cleaned_text(x))
elapsed = timeit.default_timer() - start_time
```

```
/home/erbadi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[23]:

```
elapsed
```

[23]:

```
4.358639608000885
```

2.4.3 Detection a posteriori de la langue

[24]:

```
langs = pd.DataFrame()
start_time = timeit.default_timer()
#langs['lang'] = slice.iloc[0:10000]['text'].apply(lambda x: detect(x))
langs['lang'] = slice['text'].apply(lambda x: detect(x))
time_of_detection = timeit.default_timer() - start_time
```

[25]:

```
langs['lang'].value_counts()
```

[25]:

```
en    9999
af       1
Name: lang, dtype: int64
```

[26]:

```
slice = slice[langs['lang']=='en']
```

2.5 Analyse de topics

2.5.1 Construction de la matrice des mots

[27]:

```
list_all_comments = [];
for ind,comment in slice['text_norm'].iteritems():
        list_all_comments=list_all_comments+ comment
```

[32]:

```
joined_comments = [''.join(token) for token in list_all_comments]
```

[33]:

```
#len(joined_comments)
len(list_all_comments)
```

[33]:

```
9999
```

[ ]:

```
vectorizer = CountVectorizer(analyzer='word', max_features=1000)
x_counts = vectorizer.fit_transform(joined_comments);
```

[35]:

```
transformer = TfidfTransformer(smooth_idf=False);
x_tfidf = transformer.fit_transform(x_counts);
```

[36]:

```
xtfidf_norm = normalize(x_tfidf, norm='l1', axis=1)
```

Saving the models

[37]:

```
vectfile = 'vectorizer.sav'
pickle.dump(vectorizer, open(vectfile, 'wb'))

transfofile = 'transformer.sav'
pickle.dump(transformer, open(transfofile, 'wb'))
```

2.5.2 Creation of the topic models

[38]:

```
# Create a list of the topic numbers we want to try
def generateTopicModels(topic_number_grid, xtfidf_norm):
#    topic_nums = list(np.arange(2, 10 + 1, 2))
    topic_models = []

    for num in topic_number_grid:
        print("Applying NMF for k=%d ..." % num )
        # run NMF
        model = NMF( init="nndsvd", n_components=num )
        W = model.fit_transform( xtfidf_norm )
        H = model.components_
        # store for later
        topic_models.append( (num,W,H) )

    return topic_models
```

[39]:

```
topic_models = generateTopicModels(list(np.arange(2, 10 + 1, 2)), xtfidf_norm)
```

```
Applying NMF for k=2 …
Applying NMF for k=4 …
Applying NMF for k=6 …
Applying NMF for k=8 …
Applying NMF for k=10 …
```

2.5.3 Détermination du meilleur nombre de topics

[44]:

```
nlp = spacy.load("en_core_web_md")
```

[45]:

```
# ## Version avec word2Vec de gensim
# def calculate_coherence( modelw2v, term_rankings ):
#     overall_coherence = 0.0
#     for topic_index in range(len(term_rankings)):
#         # check each pair of terms
#         pair_scores = []
#         for pair in combinations( term_rankings[topic_index], 2 ):
#             if (pair[0] in modelw2v.wv.vocab.keys() and pair[1] in modelw2v.
 ↪wv.vocab.keys()):
#                 pair_scores.append( modelw2v.similarity(pair[0], pair[1]) )

#         # get the mean for all pairs in this topic
#         topic_score = sum(pair_scores) / len(pair_scores)
#         overall_coherence += topic_score
#     # get the mean score across all topics
#     return overall_coherence / len(term_rankings)

## Version avec word2Vec de spacy
def calculate_coherence( nlp, term_rankings ):
    overall_coherence = 0.0
    for topic_index in range(len(term_rankings)):
        # check each pair of terms
        pair_scores = []
        for pair in combinations( term_rankings[topic_index], 2 ):
#            if (nlp(pair[0]).has_vector and nlp(pair[1]).has_vector):
            pair_scores.append( nlp(pair[0]).similarity(nlp(pair[1])) )

        # get the mean for all pairs in this topic
        topic_score = sum(pair_scores) / len(pair_scores)
        overall_coherence += topic_score
    # get the mean score across all topics
    return overall_coherence / len(term_rankings)
```

[ ]:

```
```

[46]:

```
import numpy as np
def get_descriptor( all_terms, H, topic_index, top ):
    # reverse sort the values to sort the indices
    top_indices = np.argsort( H[topic_index,:] )[::-1]
    # now get the terms corresponding to the top-ranked indices
    top_terms = []
    for term_index in top_indices[0:top]:
        top_terms.append( all_terms[term_index] )
    return top_terms
```

[ ]:

```

```

[47]:

```
k_values = []
coherences = []
for (k,W,H) in topic_models:
    # Get all of the topic descriptors - the term_rankings, based on top 10␣
 ↪terms
    term_rankings = []
    for topic_index in range(k):
        term_rankings.append( get_descriptor( list_all_comments, H,␣
 ↪topic_index, 10 ) )
    # Now calculate the coherence based on our Word2vec model
    k_values.append( k )
    coherences.append( calculate_coherence( nlp, term_rankings ) )
    print("K=%02d: Coherence=%.4f" % ( k, coherences[-1] ) )
```

```
K=02: Coherence=0.8588
K=04: Coherence=0.8497
K=06: Coherence=0.8608
K=08: Coherence=0.8655
K=10: Coherence=0.8541
```

[48]:

```
coherences.index (max (coherences))
n_components = k_values[coherences.index (max(coherences))]
n_components
```

[48]: 8

[49]:

```
n_components = k_values[coherences.index (max(coherences))]
#obtain a NMF model.
model_test = NMF(n_components=n_components, init='nndsvd');
#fit the model
model_test.fit(xtfidf_norm)
model_test
```

[49]:

```
NMF(alpha=0.0, beta_loss='frobenius', init='nndsvd', l1_ratio=0.0, max_iter=200,
    n_components=8, random_state=None, shuffle=False, solver='cd', tol=0.0001,
    verbose=0)
```

[ ]:

```

```

[50]:

```
nmf_file = 'model.sav'
pickle.dump(model_test, open(nmf_file, 'wb'))
```

```
[51]: num_topics = n_components
      def get_nmf_topics(model, n_top_words):

          #the word ids obtained need to be reverse-mapped to the words so we can␣
       ↪print the topic names.
          feat_names = vectorizer.get_feature_names()

          word_dict = {};
          for i in range(num_topics):

              #for each topic, obtain the largest values, and add the words they map␣
       ↪to into the dictionary.
              words_ids = model.components_[i].argsort()[:-n_top_words - 1:-1]
              words = [feat_names[key] for key in words_ids]
              word_dict['Topic # ' + '{:02d}'.format(i+1)] = words;

          return pd.DataFrame(word_dict);
```

```
[52]: topix = get_nmf_topics(model_test,10)
      topix
```

```
[52]:   Topic # 01  Topic # 02 Topic # 03 Topic # 04 Topic # 05  Topic # 06  \
      0       back        food    service      order      pizza         car
      1       said     chicken   customer     minute    ordered        wash
      2       told        good   horrible       wait      slice         oil
      3       call     ordered       rude       time     cheese       drive
      4      never  restaurant      worst      drink      sauce      change
      5       time         eat   terrible     waited    tasted  dealership
      6     called     quality       ever       hour      taste     vehicle
      7        day        cold        bad      table   delivery         get
      8        get        like      store      drive       like        tire
      9    company       taste       poor    waiting       wing        take

        Topic # 07 Topic # 08
      0       room      place
      1      hotel      worst
      2       stay       like
      3       desk       ever
      4      night     people
      5      check       star
      6        bed        bad
      7      front       even
      8   bathroom     really
      9     stayed       give
```

```
[53]: topix.to_csv('topics.csv')
      topix.to_excel('topics.xlsx')
```

```
df_usef['text'].to_csv('otherComments.csv')
```

```
[ ]:
```

def my_probable_topic(comment, df_topics): for

2.6 Pipeline

```
[54]: def my_first_pipeline(text):

          text_cleaned = cleaned_text(text);

          #loading the useful models
          vectorizer_model = pickle.load(open(vectfile, 'rb'))
          transformer_model = pickle.load(open(transfofile, 'rb'))
          nmf_model = pickle.load(open(nmf_file, 'rb'))

          x_counts = vectorizer_model.transform(text_cleaned);
          x_tfidf = transformer_model.transform(x_counts);
          xtfidf_norm = normalize(x_tfidf, norm='l1', axis=1)

          H = nmf_model.components_
          W = nmf_model.transform(xtfidf_norm)
          df = pd.DataFrame(data=W, index=["row1"], columns=topix.columns)

          res = df.columns[(df.loc['row1'] == df.loc['row1'].max())].values[0]
          return res
```

```
[55]: raw_text = df_usef.iloc[1000000]['text']
      raw_text
```

```
[55]: "The worst Jack Astor's in the city, and that's saying something. It's a place
      that's conveniently located for me, but whether for lunch or dinner, I reliably
      have a mediocre-at-best and infuriating-at-worst experience here. I got the pad
      thai once, and that was a tasteless mistake (probabaly my fault for ordering pad
      thai at Jack Astor's). \n\nYou'll have better luck with standard pub fare like
      chicken fingers, but first you have to wait eons to order, then they'll drop
      your food order and you'll have to ask for it again, when everyone else is
      already eating. Then you'll have to wait some more when you try to leave (why
      won't you just take my money???). The servers I've had were all pretty clueless
      -- not actively unpleasant but either they're way too understaffed or just not
      very capable (possibly both!)\n\nThey do have big TVs for watching the game. If
      you must come here, get the beer & cheddar soup and then ration out your beer
      b/c it might be a while before someone comes to see if you need another drink."
```

[56]:

```
pipelined_text = my_first_pipeline(raw_text)
pipelined_text
```

[56]:

```
'Topic # 04'
```

3 Traitement des photos utilisateurs

[20]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from os import listdir
import cv2
import timeit
import json
import pickle
```

3.1 Import du dataset

[21]:

```
path = "/home/erbadi/Documents/AI/OpenClassrooms/Projet6/photos/"
list_photos = [file for file in listdir(path)]
print(len(list_photos))
```

```
200000
```

[22]:

```
photos = []
with open('photos.json', encoding="utf8") as fl:
    for i, line in enumerate(fl):
        photos.append(json.loads(line))
df_pic = pd.DataFrame(photos)
df_pic.head()
print(df_pic.shape)
```

```
(200000, 4)
```

[23]:

```
df_pic.describe()
```

[23]:

```
                      photo_id             business_id caption   label
count                   200000                  200000  200000  200000
unique                  200000                   39830   72475       5
top     sfnVXwqu6jd9lhxndjbwUg  RESDUcs7fIiihp38-d6_6g            food
freq                         1                     652  107420  118597
```

[24]:
```
#type(df_pic['photo_id'].values)
df_pic['caption'].dtype
```

[24]:
```
dtype('O')
```

Taking a sample list to test the processing

[25]:
```
sample_size = 3000
sample_pict = list_photos[1:sample_size]
```

[26]:
```
sample_pict
```

[26]:
```
['a7TVhV_PxkmhaBS27IXZ-w.jpg',
 'nc6jn5XcmauB-axL_tpccA.jpg',
 'FWokWbVDIQMN2ns-DZ5SHQ.jpg',
 'OODQqQ-UjxbDMbSySXxeTQ.jpg',
 'H7KTvtONngZNKmpPqumF9A.jpg',
 'q515i-Y9WjsGknw_t4ELNw.jpg',
 'hRT7XLzMT2lTib8MDIoZuA.jpg',
 'zCZgkRyfyN3WjBg3RD7hHg.jpg',
 'vh_O-B2PUF3gN8oh-DvMqA.jpg',
 'JWpUavlxxbB50DdK0880nQ.jpg',
 'B-9_k5t6C9rMb-Vk5ocl7w.jpg',
 '5aikKan3ZUDH5z3mm3m6xw.jpg',
 '7OjrjPo0Usj5udb3bSWZPA.jpg',
 '9yCQ4GJBRSS26UqU_FxA6Q.jpg',
 'kuGsfUQFDxQW4QnUdoH5Wg.jpg',
 'ahDjx26yHuswYaPMYg0YJA.jpg',
 'IvNu0VjbM9jiXRpDFqj-5g.jpg',
 'APc4WvfnJ8j_1ik00pl2Fg.jpg',
 'bj3lr2RluYRSDCpLx2ja2w.jpg',
 'sirWaYr6004CFa_f6nk_Qw.jpg',
 'I93why2nR67ghwmHZ8szAw.jpg',
 'jRB108lFC_7fCuZbbkwGHg.jpg',
 'GakCmsqTJnIWGeh1pAAkxA.jpg',
 'Ei_tuLyZTTVLy0VSGKtluQ.jpg',
 'GZGiOlqptphRAnKqtt8TEA.jpg',
 'b5JaRfLDSDgWQI52_av39g.jpg',
 'izPeVLQ_EF4iA2qjFMri8w.jpg',
 'EoGtw9CdcHJOPzbZfA8-Ow.jpg',
 'P2A1UZRb5i3IJif_QJBsng.jpg',
 'g65wi2y4aS1Df-WEYyTbGQ.jpg',
 'qrz-RTeJg66YDFYQXXz1Pg.jpg',
 '8zr3da70dR1lmsw_W2kbMw.jpg',
 'heC_yhsQ8ilRMcD2VuyblQ.jpg',
 'Ag5uRGzkN2aONNW-ioHfjg.jpg',
 'Ysy2a4nhekE6Qsk2JJ67oA.jpg',
 'i1SIxcaA-zH4izdZKddLSQ.jpg',
 'oNGqccgbyFguFH7P2hrscg.jpg',
 '4f1VX_LTr8aaA9BqiCDuzw.jpg',
 'O3E7KsKSTLr_SQ7Yty0XMg.jpg',
 '757r9cWAZVI4A5lMdz9X6Q.jpg',
 'vrOKODgQljkXxbuJxNViZw.jpg',
 'tdiF6p9GFsy9EXW62AT8EA.jpg',
 'Euw5jYQdjwfUsKaXnCJulA.jpg',
 'mSqJBUCoDcxCLvg_5OjsBA.jpg',
 'rdqJUh2YATxdJJYlYQvhUQ.jpg',
 'wA24Ni6b4BMjflKLFWcNsg.jpg',
 'JFUQCXL-bH8xfH_D3iWqkg.jpg',
 'KVEUl_ZXYva4Z_t2VVtGTQ.jpg',
 'c0q907PJWzOFozh8Wvo_bQ.jpg',
 'KNhWQKS3ndqlj_OvzyG6Qg.jpg',
 'c82oR800Agle1QGadZF1bQ.jpg',
 'wjZHvQZYjgT4KsOgf60NeQ.jpg',
 'jldh6FdYpRI7wWYxHjG23A.jpg',
 'D1AFUQjgXd50y6729bRnfA.jpg',
 'yUYfOhqfAwwX9PGBhv7VWg.jpg',
 'ESXqaX-X-qw45b-vWMx0QQ.jpg',
 'RZpGIs9dobJx6NCG3jXFrw.jpg',
 'SPLuIA2OP2Z_jvI8Fzg8OA.jpg',
 'Jn7t0Ar7p1dbEjvN9VGd1w.jpg',
 'l31-agFyv4Pi9yvvWft1GA.jpg',
 'OUyd2LYw1Kg9OVYPOFCElQ.jpg',
 'khfxZZjPvMxxDDnRFbT0Ow.jpg',
 'IN99hD0lGhz6MHqtSooi7Q.jpg',
 'pYH27r6SSfpSpVhYYLC5AA.jpg',
 'Tqdvf50X32cbeiSvYmQDYw.jpg',
 '6axvlICpGy6hg-t-6P2jag.jpg',
 'MTvKXpWt64EYk5W_I4I2lA.jpg',
 'hWOLDBf4LNgg5uTlqRGS3A.jpg',
 'vP74nGpg8DGjAlDpOrA4Kg.jpg',
 'G_2s16gMH0EaFS55AYutQA.jpg',
 'YXlvfx3C6pNUv6ppCtiCOA.jpg',
 'gGjCH6YYZBXYNSOXDkkRPA.jpg',
 'UwHV_D5bB0hACCMLddji1A.jpg',
 'I-MgBBK_-QZJFWBZs1Jkgg.jpg',
 'waSCAy3XPiG-Bkp8ydhacw.jpg',
 'qnFErLme5mfHVgzWnOXTWg.jpg',
 'Yo2cEQQy8fPBs7Q_blSEUw.jpg',
 'KAJZfLoROfufa7g2ZlsVnQ.jpg',
 '3P82GYUH7K2XsVIwIIQRgg.jpg',
 'eC_rh6d52srOi4QvRKBg3A.jpg',
 'spyBZfGY-yh4E1VkAqQR0w.jpg',
 '2qqQCQCL2Nmn9iww7qdJ7g.jpg',
 '_Dm9FujtHXsC11GPWOeXVw.jpg',
 'OflHX4RtPZjDSGLjqWvVxQ.jpg',
 '_qkCc3i4wF6iMfmVpDO1lA.jpg',
 '42stoEvxf95MX3gWnQW7lA.jpg',
 'MC4GA4iUaQ1T5iqD--wFpQ.jpg',
 'rUROJ_O6j6CUqwtQHIvXPA.jpg',
 'w5VzBSVP78GfLcSOB_WcCA.jpg',
 'cWyiXLX2udyoqr7aN576Yg.jpg',
 'Hb7YqX7lZh8eWawjaZSqHA.jpg',
 '2-_3BpwVd1A2gfSuDQqS3g.jpg',
 'EGZbxb7XXc1DKoLW1IC7Jg.jpg',
 '2DKU0aG7qhT_Aun0w6MA0g.jpg',
 'o7WauYf7FBo1D9oDINdIRQ.jpg',
 '5xpieNxE6rHO7j8sijA5dA.jpg',
 'sjqvrBnnWlreND1qDc3yWQ.jpg',
 'K2rMuUly9W8_IEWVJVlcog.jpg',
 'G8BFLHAOxapTAbZ4NHBVWw.jpg',
 'zJcKE29G0eZ25sSJX1gM2w.jpg',
 '7Ifg5N5kpo8jmeMLsue_xg.jpg',
 '4rGUhlExqP63mVHL6XdGwg.jpg',
 'SBbcAvc5p2ZglqB2-mmPbQ.jpg',
 'HxK-qzrT_MHjsrTdmRgZ8A.jpg',
 '4_097gNBOM4PXm0mXpPPqg.jpg',
 'qaXNUdgwLJUhVBa0loF_Qg.jpg',
 'gaXJIYsMHGxbFnHJtrmjkg.jpg',
 'hGwNqvgmUvbRZxE7De_8Zg.jpg',
 'd0yh6YkS6Enri-nwp0nnlA.jpg',
 '2B_bLEf8deipFyoGH3oGCQ.jpg',
 'ikqYaG7PMvSBYm8BSCBDsQ.jpg',
 'VE0rVCVuJOjp9n-Pp19sSQ.jpg',
 'zVok1YtfVdNfP23x8OkbYw.jpg',
 'Rfz7cnEUm8iHlggdZY6IAw.jpg',
 'gDagbw-KjWA1wsHFHFEZ3A.jpg',
 'YTt7zsb8QLxeQ6Ty4rji1w.jpg',
 'BN0gOMCAMJ1YHjdTQSARTQ.jpg',
 'eAGuu9a_iZuskV8V16DCiw.jpg',
 '3pshI4tmFfbdPIL4ao9IuQ.jpg',
 '_T_o3Oq2VS40BYXezO76aw.jpg',
 'AtnllN8EecOiiTfvTXQQyg.jpg',
 'VAwOZrKkD5coGbOvQriUbg.jpg',
 'eV5YSO-dI9NhQE2YvtvDbg.jpg',
 'ddnDoxep4CfYex2IJcGHTQ.jpg',
 'xDolJO5E53zDmvG55XBMPQ.jpg',
 'XQCJ6NsKjaBsoE4D9dDKRw.jpg',
 '48aNDzXOxQbP4JOm1d0qfQ.jpg',
 'ebC7K8Z4UvWE8WUXAV40RA.jpg',
 'KXcTXsWg28i5kEBG1o9kFA.jpg',
 'CkCrKhwoOSCgPDyekinlDQ.jpg',
 'Gte4hY3oYziJRISK2JTlzA.jpg',
 '8k-VBEjxkA29QF3MfTTHcQ.jpg',
 'dVIv0Ffs7m24DhmGrCPpPA.jpg',
 'Gexqn0amKztubDHmvfa-TQ.jpg',
 'i_r-IXkWAOfN0QqapT2yYg.jpg',
 'o2RseXN9lzMoZy9JyL_HHg.jpg',
 'PRYeL5y1gj9z1Qw5Q3VBTA.jpg',
 'xOAG6tBTjYzvteV6bN8Scw.jpg',
 'APnZystGG1Ml7PBvkAxAsQ.jpg',
 'NxcmroC1J5NaIKMUsqzgxw.jpg',
 'T3hHuYbQyWks1FBPQNfNIw.jpg',
 'KHyEIBqZx26MLAIRfK-9og.jpg',
 'O3sTaz_9Fjv3riyIWTPopg.jpg',
 'F4376xDfNK2W4RJKYweokQ.jpg',
 '2BDYostevVhIhfF9gDsH6g.jpg',
 'cSUiCUA2GB2g7X9KAUMYWg.jpg',
 'CBMxevXj1fg32ZG5_7lQwg.jpg',
 'j_YVyuXGrkla-ZYi4cqv9g.jpg',
 '-HXPOkAgXcgeHTjrlQEGnQ.jpg',
 'W--iOyITviOJhdxjAwZsQw.jpg',
 'te1dskj2i3Mrn20nfDp3jw.jpg',
 '4dfUHuH7CiZc4wwXgOjR5A.jpg',
 'HzdIm7nUgnvV5Ezh-f-E5g.jpg',
 '62_Ev3h_mdYHRFOimSfmGQ.jpg',
 'wghfoIrMYCFienWusW-iNg.jpg',
 'OBJ4MMKArV50yfj9wApWMA.jpg',
 'ZbWyByy8o0kN6gglAEHGeg.jpg',
 'bSRJVGe4CYZjbjoC9a6EPw.jpg',
 'JWnGBlm-wuIrL3GFqokKrg.jpg',
 'WwQd0aBvLniNEdE8uTGbmQ.jpg',
 'xP1QFeC-Llz3kofnVAurCw.jpg',
 'pUZSayvIyDy-KQFhMEW8og.jpg',
 'RhyPIU0mME2AXW8oJOJusA.jpg',
 'n9mPN63pqLmM2wAxv0XPRw.jpg',
 'Xvv5ZXA6_pUp-toFp5t1tA.jpg',
 'MxNrbF_H4bNLRgMjgTsCCA.jpg',
 'OFp1bsbcnkkYEKc9Rmsljg.jpg',
 'UWNXe5O-PArMoqC4-ESV5w.jpg',
 'TlfE-iPspzkvFCZhgnXbeQ.jpg',
 'h-xgtzsjvPaWfHnd1k-RKw.jpg',
 'yMCC2TTNaPlvVr15qRXECg.jpg',
 'mfX4ifdS7iKHIXIp_hO75Q.jpg',
 'rJyHESxAzBy_o0TWMwO51w.jpg',
 'NB5_2i7IBdeAiCQSZ8FBnA.jpg',
 'S6FrBu6n2XO4SBTGO7bFhQ.jpg',
 '2LtOWZ0DozGDaDfKipVtIw.jpg',
 'W0XPgwhCsVSlwoWa7ShX3g.jpg',
 'zuQ6BQA6wJtXVbw1KO5NCA.jpg',
 '2UnLRTbdjJzx2Rp16UpZgg.jpg',
 'rUcrirjDljD38y7XRNMhbA.jpg',
 '63AIjC1Isw4lK8FOArUBMg.jpg',
 '8mzaqxtUIlzprv34jwv2kQ.jpg',
 'uyJsNGmXgfqi3UsZrLukdw.jpg',
 'EDI4fm51NbdBdTI3TRANGg.jpg',
 '-we_-ALjm09qSY_c8mgWSw.jpg',
 '9Kl7jMGD_djjPwv1gNmHVg.jpg',
 'hdScNkU8hHbHKFbU9R1Opg.jpg',
 'vai7cObrBYGpX1ztuXJlcQ.jpg',
 'RwaikbYoSwjSQpGFE3FVNw.jpg',
 'xtJy6pl8LrvAPkiCiCcg0Q.jpg',
 'hKaduGUFQy86Hov1_wr3dg.jpg',
 'Qku560jZ6bhxRvg5DfOi0g.jpg',
 'lGuvKGHocsTaGZUQny67Zw.jpg',
 'yo8eE_vycRS09zZ9WUxtFQ.jpg',
 'sTRQ7uePBFtdP_KwP9rakQ.jpg',
 'MlWeKErMIGHpU3pYS8381w.jpg',
 '6zCrm9IqEJjpaTB0T--fxw.jpg',
 'g_hUULMoAZYLBjEn1zjaQw.jpg',
 'hI57UN45RfHKNPC8G08ybQ.jpg',
 'DPn6SL1gnbwky4vXFMENlw.jpg',
 '9O8AWf60Dsw56cAL181QaA.jpg',
 'Q0LU58aAdi0Pch-wlkC84Q.jpg',
 'hQtl-H4w4jYL1SB1i1adfw.jpg',
 'h_Ke-ey6aOA9UcSY0A0r-Q.jpg',
 'QN4yhsOs55WIAy6HX90Lyg.jpg',
 'KwruAChQ3nJ1NgqyiOIuNw.jpg',
 'OfsE5zlDY9izl7bg-UO7nw.jpg',
 'anfTZCCcR0RGIVBNCCX8sw.jpg',
 'WnsnB8cGpyHRI2Kk4VxLtQ.jpg',
 '5XhqFfQalUl4aA4Lt7QnQQ.jpg',
 'hoJEfSCFk_ouU1GxRw2ELQ.jpg',
 '6gzW0yjVjeQhhZ0jqYsS7A.jpg',
 'EagXuyf7wEW1eMtmuFJzDw.jpg',
 '_OCo8VOe6BDegw8pSm9Ohw.jpg',
 'ZUVM07XCfpfx_ue6UCZOkQ.jpg',
 'mX82DczarhfqBg4saint8g.jpg',
 'Ac8Q5NFSSicWROtsAaJA4w.jpg',
 'xw4Xhj7nI-DsthXHKfeBCQ.jpg',
 '9diwCITzvsdavWVQbL3uZQ.jpg',
 'bvqg-LKp01cvNSUEXXbOyQ.jpg',
 '6F6Lq3cp3JzFWWa9x5C1gA.jpg',
 'Ut3zVvaRwJA492a-Oix4MA.jpg',
 'I4tczDUrgOqb4iOUmfpwLA.jpg',
 'gKaV9Eo8Pa2DX412FCM8tw.jpg',
 'm0Z00v5qnsCIm42oGmt0zw.jpg',
 'r_vX3QlgNkQvS-x_UIkycg.jpg',
 '387F_Bvf0-8PzUCadQAxLg.jpg',
 '4bYGEs6AILeUy39Cj7bI7Q.jpg',
 'B5OEUwzR3c3I_ib8V_qVFA.jpg',
 'GQbqVnct0i_yujn3SMoLGw.jpg',
 's92XUWPKzVrThlA4X4BmsQ.jpg',
 'iZ2qtutFzutvp1gRem__Bw.jpg',
 'ivhEhlAF_5hY0_w_Py0-UA.jpg',
 '13kZyQGTAGf-80ZVvkg3wQ.jpg',
 'BL7v9il92L_uuF-uEtdyAw.jpg',
 '2w-Ec3puIe5y9GtUM16htg.jpg',
 'MqFDASlFoZAQnL1yBakkxw.jpg',
 'O4DDylXuCDQfZ27U7Pc0Ng.jpg',
 '5namyi4JjSDRYWA6YUIBgA.jpg',
 '-RG8Ypy0kxNeSCSUcyUwSA.jpg',
 'JlXrKybMcFV2gwvklLeFeA.jpg',
 'WuMrpmOV1_FGYGiBKFHXBA.jpg',
 'o1_bs8RHJzb4yBtpZOqrgA.jpg',
 'fz-eZtjNiRHwKCFcEANIXQ.jpg',
 'oaTupoxi9KuTzUAntxHaSA.jpg',
 'b7jWowGUoVWO8ztik3K4fQ.jpg',
 'fK3mnELLQFYXuf33w3uTbg.jpg',
 'Z35kyZBLDyiHWCpiPp_Q5A.jpg',
 'tfKprDFcYRsKXCL4SJDJWA.jpg',
 'UkuHBCxNeFIU_kEnf_0xGA.jpg',
 'lfYqyLN-gwCY99LYK7My4Q.jpg',
 'l-VhheGNdxfK6KCRfwCPjQ.jpg',
 'Lz_4LSrysYeJdLGswyN-kw.jpg',
 'ri2YIIpWowqdllr3AFEOtA.jpg',
 '2BXJlJ3vSXIpuLlH5O43Og.jpg',
 'IN_To0gJKyn8Szzt_uL11Q.jpg',
 'ga_lJ-KMHdHrgmfAcqWpUw.jpg',
 '4q8-8rhaVUFr7tMNA8iTmw.jpg',
 'JoRmO05xXDYoHkDHiOSM-Q.jpg',
 'SI_4ima5VkA85reMPSr-qw.jpg',
 '7UUv1QjWB4FXayGLVllx0g.jpg',
 's8DK0xvyu89Zt2sgf_VFLQ.jpg',
 '5P2mmg7n4HwsEDLHiFlXMA.jpg',
 'MwKt71YdEXPXXsXv5YLGqg.jpg',
 '4uDK3EFKphCN2EqhJ5gXZg.jpg',
 'uuZApgW2s4eenSBcNBOi1w.jpg',
 'VRCM46_DdI5CxQbUTctm3A.jpg',
 '2Qh518t0-MAUhx-LpzD7iA.jpg',
 'Cyuc94pr1ALljB-p0jx-EA.jpg',
 '2dW7SDGLcRQFRzBlZrCXCA.jpg',
 '5MwxFLnPK3z8Q5RMiJanww.jpg',
 'v7GKzi7zDHr5puONSVi8nQ.jpg',
 'VBW7KsbuGydW2_RbklSKSw.jpg',
 'kcc95ZTxNExSZoWBVGHSMQ.jpg',
 'T7DLlhP1MB4kUWcS8W_sdg.jpg',
 'Wm23nMMnFXnSsDfmXKmUbQ.jpg',
 'mg-2TI5yFA760h6ALAGd6w.jpg',
 'G7z1D7PfeJD03AdwjT34Gw.jpg',
 'Oy5NKo-pCjcRnZcLQRaBuA.jpg',
 'y0e3Cg3C0ZvxDlrrBk5skQ.jpg',
 '2RY52hoBk0tw0rFpYbFbhA.jpg',
 'OYM79PT3lQEX9Pec3IHRiQ.jpg',
 '9L8R2nHT4ROShRuysdJ5Rg.jpg',
 '90DSFSrF3IOr43dIV41NDg.jpg',
 'cob6OQZVawyGxwIKECisDQ.jpg',
 'ZXTgIaZ94UroCeAqtbjHNg.jpg',
 '0sq8p8TyUjSdlzb7aiX_Ng.jpg',
 'J8bMT5s8eT8XoSsplSglfg.jpg',
 'NoqgjgmssauXWefbntYTyA.jpg',
 't_SChh_fqwJLjQUb7SrQfQ.jpg',
 'KHtloWWivT1XSr6Kk68RSA.jpg',
 'w_zHOSSZD-9qKm--dY4Cbg.jpg',
 'vNdm45K7qYHY31l08DOsSg.jpg',
 'fNb-ygk7jLONRsImfiM1dg.jpg',
 'eKLTvor62fQEOh_GLvgdSg.jpg',
 'bgZgCIvGejCbHM3FULtDSg.jpg',
 '_GjyCX1OmEl_ha6NrsoQ1w.jpg',
 'cI_rVYgOwzLnuOeyqAzK0w.jpg',
 'Okg7A64t8ZhsUHVJxjzyeg.jpg',
 'SSMjqlfwz0JhBnmQGpS01A.jpg',
 'b8f5kpYBXuPtsLsuSHdNdw.jpg',
 'N53DXgcN3WcsAKfC851uFw.jpg',
 'ypT_UATOggAqgg5UE3waNw.jpg',
 'PhTA9eM-DPoD9AhrfcVnRA.jpg',
 'ihU7SeViZ9Jn5lzV0XAKnA.jpg',
 '2PNvdLwNquvzs6bOV1ZtkQ.jpg',
 'KJz_ernsgeS9vUKZ1GHw-Q.jpg',
 '3jGn_fZpuKzJGQg4itj8cg.jpg',
 'ywnS-xzXTRIkQtCZhnprvg.jpg',
 'HR_o9gV7XVf9Z1K46iqtLw.jpg',
 'W2aLac7BI76YDlX9DE3--w.jpg',
 'EH3YCGG-YfXYGQlM9X9Dgg.jpg',
 'EzBFg2Xp5fLyaDj90cRWgw.jpg',
 'mown4Dk6F2gXXnI6EcTQwA.jpg',
 'fwZec3gAH1Z3WyR5miprMQ.jpg',
 'oLxF7u7RqwSXWlyV7f8qrg.jpg',
 'ww40NoXhkzYcfzfHxLEKAg.jpg',
 'oInAVroGib0zJklDxulN-Q.jpg',
 '0PTgQ7zJmc9ZPTDH_EQsqQ.jpg',
 'UsiVqh6GjHXslI58nnZf2A.jpg',
 'I4LyTsHQ0aSUeLkFAIJVjQ.jpg',
 '_ezDthQJ1N4hmDnNhKHFMg.jpg',
 'td6SSFIkWBRj3jKobHDoLA.jpg',
 'FGdiMt1STT9Uh1zpQu_WVA.jpg',
 '9-F_--zp_WstZl6oUr0_cw.jpg',
 'kA64opFosjI12jkplUodEg.jpg',
 'TLrY3ZgkQKkTImqCFh4I2w.jpg',
 'NO1Q85MfrL-C1luQAxl4ZQ.jpg',
 'Pe4l6I4t3fb8bDv-ipaW8w.jpg',
 'ExLSl_AOm0sgKawcReRB6A.jpg',
 '4DtLFHu3BchdryTJEUgZrw.jpg',
 'k17q0eua5ebPrNbA5PTk1w.jpg',
 'PGimThOA-YahwUPwrcm4pw.jpg',
 'UdNrj89qlWO70gNkJ8lQpg.jpg',
 'poqlRqQY1xfbKyEIIOBC6g.jpg',
 '5R2ySBTVzY3ci5ZIsdRFGQ.jpg',
 '3SvI-pQdwEbETV9Kp0-sYg.jpg',
 'yxn8xej3MT8XHLGE0k7zsQ.jpg',
 'CRmaIElVIMPgGcgWS6qCwA.jpg',
 'asGvtD3I5aAnEZipTb9xyw.jpg',
 'OXhhY2fcR-K775S5je0B0A.jpg',
 '9VEYaty9OOVR5sL7nSMjog.jpg',
 'ndQuVNqFWhGcJMzVblgDMg.jpg',
 'Ton1wAOZyNRLNJs1TRQf-w.jpg',
 '_Tu_DigikkkaIvnLx5ZVxw.jpg',
 'e-OC-givVKKaLuwUlxsnyg.jpg',
 'SozzLrvQXL3hvgzKAmoNuw.jpg',
 'hBKBTcBcS7mTd0rvOYU3lA.jpg',
 'mJkyLun33Ewfktj6fVzk3w.jpg',
 'YnwC_PQpqskunSYbMSspIA.jpg',
 'O5ikTzO2-R8WwGzKn2cYkA.jpg',
 'qT8P9S7tS4j2oMqwrT6lkg.jpg',
 '3HtluZEsUW8mD_P-Vrz5fw.jpg',
 'IE61tD7VUqHVSRGgIj1J7g.jpg',
 'GRDNsT6ZYdYKsNzM-y5KYg.jpg',
 '5CbWUx_KWczDOU03yg-K_w.jpg',
 'xrKpFQKiQg5pqpbDVT4zIA.jpg',
 'LxrwhW-ttgOBGBbDJe9-Pw.jpg',
 'VgCPxfjz_VYLkwdCf3EGsw.jpg',
 'lqNk3_P_9AbMI1Q4e0eMXQ.jpg',
 'qCk-ZI0aA9CIUQScf2lFCQ.jpg',
 'qcTi2JBlJ-SCE9Vk1PNRWg.jpg',
 'Tp4Zyiqfdb30ies35-0i_Q.jpg',
 'ftEJBJpe4b3-TiCm6PP7SA.jpg',
 'slP7uRKvts7ZluAXV0UMpA.jpg',
 '9pehVjv9zCAL9BQM0fX0hQ.jpg',
 '-aaPDUfMxRPgxYyFnjqEXw.jpg',
 'JXsOL0SgyNOu7RMhBpNLJg.jpg',
 'ezOUKzl3gbEo1FaNFvnOHA.jpg',
 'Wp6qV_kvBEgQ1ZembFk6Vg.jpg',
 'qx-brxeIxjv74yQj12z7Og.jpg',
 'hicNcxVsh2L8fwdddGhSkg.jpg',
 '3MmfGCxFqKb9xVWQpCzmtw.jpg',
 'QPvjas89VC5xt4nQ3OH6Sw.jpg',
 'DHptbULGh5BFDI0kjW0l8g.jpg',
 'SCjGaQ_LEdYNp62t7RP0qw.jpg',
 'OyqLnIYmiNbMf1SWu9uK8Q.jpg',
 'IW9lXRV5_bpaflbRYg6I3Q.jpg',
 'lrk2GhWW0YU_dK__ME6YoA.jpg',
 'lug59Wu19yxwTeF0nADyUw.jpg',
 'BdaV-mf3JlOYtMgB9jEFiA.jpg',
 'hC6xf6-JkdAYZeNZI2P1GA.jpg',
 'D3o_sHBAnxkHNy7dA0RrlA.jpg',
 'VULsWO0fZy4WvzX1bB0iWw.jpg',
 'LF4QdVflKTxJEpbiDJmNNQ.jpg',
 'A_cqDbxR7aoQhFyToOBYBg.jpg',
 '68U6uqezKkWS_iaihJTOGA.jpg',
 '5heHdoU9iso-v8N6wwcA0A.jpg',
 'p4WmRoMxC0nuCqNwVCRMpw.jpg',
 'Ck58bEJLdNHTyfORN4o5ng.jpg',
 '10SunioJ36w2MPN1BFGgWw.jpg',
 'ws-yb5XJeD8ASXowZ5ef7w.jpg',
 'Yx2myRZqTHN_wg8EWUNTfQ.jpg',
 'I_m57fV2Y86fDgn4bWjaxQ.jpg',
 'vcxJvWcQ2Ql4j5x_fV2Zkg.jpg',
 'LXQ0Z-q8gIndJPzlJ0naPA.jpg',
 'Q3dbp9JKNYJXBp1w5TzLrQ.jpg',
 'ZubksoAfK7QHr8DisAoZzw.jpg',
 'iqeRwPZ_GE5EOfRu3ofdqA.jpg',
 'ECKh9X7ytx6TQ9U62MvklA.jpg',
 'BW3o_9UAZq27Glilj1yaOA.jpg',
 '3IcJyQlzmXS8HV3z6ZsM-g.jpg',
 'Q31OkIIt-HiHCHhJK9Sk3w.jpg',
 '9Bkpev4MFvYrmZ7988T9Iw.jpg',
 'oR3OpLY48bMPagUkw03WHw.jpg',
 'xFfdtPy-0m75BEyN9u7fEA.jpg',
 'wAUh-O3cu2ZON6dudXXhJQ.jpg',
 'vq_JnK-ZI2YBsK6ecnFYnQ.jpg',
 'ZR62aOOxiusALoSfUGwOhQ.jpg',
 'OvkEady2I8dplNmtKoz0dw.jpg',
 'nbE_NQH5jICLoeK5XZpjyg.jpg',
 '6CKRqed3N8CWeTs4edTljg.jpg',
 'S6Mz_On5pNNfEUb5NnlcZA.jpg',
 '4OUtC8XMA52fYIeEzDt0cw.jpg',
 'T5A7CKRpq9an9jrj_ZY_1w.jpg',
 '8g93kh0AKjmTCiLv1IYomA.jpg',
 '9HwlBk_9SJ4GKIDcVHcQSQ.jpg',
 'Db2d1-khMncU1FidVDBYsQ.jpg',
 'VpdLaAmbtpMhO2EaPozemA.jpg',
 'zo0jALO5DSSbQ2_wgAK_jA.jpg',
 '6SbM15Ip0H2cblKXlI-r1g.jpg',
 'temRgI01pRanlo1mFnyeRA.jpg',
 'ycj5Lpcc1ooPvw0JGJDXMQ.jpg',
 'ziGzZqINEojunU6Tamno-A.jpg',
 'U-5tv-nKx1navLQzzxdCGg.jpg',
 'zG-T9aBl0Ci_8DuWrAbH4g.jpg',
 'TiWn2H4vMg8rhGRrPi0mpw.jpg',
 '0SfewHYMTTLiWf6AyTS52g.jpg',
 'XE6QsNpo-Zh7bgFmzi29Fw.jpg',
 'ou0rnkVqhAopsWJK4IvX9g.jpg',
 'mWef_KcHN9dp2-61ufKhKw.jpg',
 'LklBr85LSDl4rIjY_QaqaA.jpg',
 'HRWyK8VqQbmCQioV6-y0NQ.jpg',
 'QVB_dQzG0ZNSAcbJPwTWag.jpg',
 'jXFuihgXdimzYoZFQedbLQ.jpg',
 'gxNnoG4D7rz3LwKasCdN3w.jpg',
 'kfrD85XNcu1qynjvN4b80Q.jpg',
 'YyPoiYvjWHf2t_I3bg03dA.jpg',
 '0Ix0F8zly6KxbzGkvnxzbA.jpg',
 'TYBMjxqV3mueFPcM0lAXzw.jpg',
 'xBSTzoapQzq46leRVKd57A.jpg',
 'J_Ij06-a205BIHnH6KRYFA.jpg',
 'mFGziDM7NzLVKrYKyW-bFA.jpg',
 'jzDme1nqGWqBzodRKRCoWg.jpg',
 'H30K0ASmaps5oWYxg5K0Lg.jpg',
 'qEbLgIjKkKZspmLPAs9Yjw.jpg',
 'hzogSCOPsXso3tYl_gPHjQ.jpg',
 'x4FZavl0kSKKjjj0V-n5bQ.jpg',
 '0K-nNIaw3jkiM5Ez_NBdmA.jpg',
 'hg51dr83AmqyLca0vRv2TA.jpg',
 'FDIjUis81R4pRmWIwMXopg.jpg',
 'htiHyut4vqCnf1dCkVGo8g.jpg',
 'S_C_MFQFPjh_jbr--EwFtA.jpg',
 'XxxUMEGe0npV6egFFCShiA.jpg',
 'X8An5Xyyj10XoWKjsoJo7w.jpg',
 'fonTCzGxmBuoMU7xNxGCpQ.jpg',
 'CsjYDE0BkxEubXayLyYbuw.jpg',
 'qeEYc_isdwzcaJflWLABiw.jpg',
 'wiN2X0Ez8tjg2B4nolqu5w.jpg',
 'YH8a85oFixDP_eqyu06PHQ.jpg',
 'c5aZ9oVWJai0qMlbups7Iw.jpg',
 'zZtKvfUIp2_lYgTpnJWYGQ.jpg',
 '492-M8FqUGKQ0y_C7kl3lQ.jpg',
 '4XjW3jMSTKwfo7iV9tmh2w.jpg',
 'epq3M8owkKSKIRUoPdGx_g.jpg',
 '8bfWMJ_xhKATnA-mplO9DQ.jpg',
 'LqYF6WCKlUZM01ZVUFlHAQ.jpg',
 'VO6_iuq-2WyR81KARvxv4g.jpg',
 'LNj6XzHjlICSemX5FXm8VQ.jpg',
 'I1HfDYQ8i-GWyJxi739qGA.jpg',
 'x4QbTHZAn38DbIWxwhjpug.jpg',
 '6HNNOJS853CzXyumwsmGXA.jpg',
 'mI8Bz-kDicPFaqyeJ3rhow.jpg',
 'mTpxf1NhFEwTwDgxrCpFRA.jpg',
 'IUHiaCiRVXbg9FUPXuUWRQ.jpg',
 'e_Ictq9Tfnug-B9CBlhPfw.jpg',
 'zwDQkZP4WJWdMB-Vb2PpVQ.jpg',
 'xnVDod6kMZoV8Cxgy6Wolw.jpg',
 '8WTqSEhKnq0bfGGLotcDqQ.jpg',
 'JA_gwS1O-SF71DQAJJPK5g.jpg',
 'W2jMmtF9GP9-HziKW-qB6Q.jpg',
 'dTFDL9YDVw7s3gMyaKA9TA.jpg',
 'WRz2wuXbyJuT4COORFQqyg.jpg',
 'lETz_hYaFzzQ7zPfWfFpjg.jpg',
 'eqXNNskeuWt8n9oMxcvq8A.jpg',
 'oNGTd2yCZyKoxNkb9QsVzg.jpg',
 'p3v0nPIOFk9a-m4Wq8vohQ.jpg',
 '6utrRpHumGxAFxzofnMZQw.jpg',
 'DafQbpH-9aIXgGzUfoNb1w.jpg',
 'NgQSK9DHBCYY7CS-gNM6kw.jpg',
 'XjHLHmFjChwVWiQpgl_TLg.jpg',
 'rZ8VbIFquYbZriklykOQkQ.jpg',
 'UBBb5gLlOaCIIGVxRdHOxQ.jpg',
 '19aCDVUyKyLLSlJ9_UUemg.jpg',
 '-1XxSmYXsIMaQyER2E6LQQ.jpg',
 '9b8B1g2y-p22ZdE9d7qx8g.jpg',
 'QmdXK5RDqXy0UKw0Ig6c1w.jpg',
 '39bz1HyQ9AQ987CyxYEQXw.jpg',
 'x55B4FZ3lvtVbLesMSx3XQ.jpg',
 '6ufuONOlhhMV5qYQVpZiTA.jpg',
 'i3qSrUWnrWUNe2-k2bda_g.jpg',
 '5HWOR8Y50WoIHtB84h4l6g.jpg',
 'IMAOfiHAXjoGxruB3IRVGg.jpg',
 '1nwWshFgLBHFDHjGSjhTWA.jpg',
 's3dgEDQJMuQT-8v90ESx2g.jpg',
 'wEtihXY6jPVxs5Y85LZ-gg.jpg',
 'xoesNO5fC5q1a3S-gJHvnQ.jpg',
 'yfuDI2XagE8iU5KMMKz5vQ.jpg',
 'I5XIk28o4oAapxJtl702cA.jpg',
 'BWF9G0zg97--cUN1XNuXAQ.jpg',
 'F54lu1733mX1OfvOBCaoWA.jpg',
 'jXNIMxqSRkmFJhcGS4s0Gg.jpg',
 'd1G2zInRB3tmsHpqpRjDFg.jpg',
 'bLhDOE7oPUlIJxW-FZcstQ.jpg',
 'wA9hJQV7xRyC5I98a3u_6w.jpg',
 'NXUc9RGKwwwBfl22J8yvxA.jpg',
 'OtkAAJO4nJZaSwusJhApEw.jpg',
 '5qJ9C-SHIIeoILqkFU6tIw.jpg',
 'doL2hTksZmX_jyKR9iBPFA.jpg',
 'C3d3dA5FW9FuFTeUGecACw.jpg',
 'M1dcYi0UFtle5pyNE8EWMQ.jpg',
 'byufCCMhz1p_vKiwcOTnGw.jpg',
 '4Pnw4rq5w8GWiyDQ5FvcQA.jpg',
 'r8k1my-2OGaxCvePsB2CTA.jpg',
 '1ikh_irLpX74hCnXRrHe0A.jpg',
 'oCesvsuwZIAU89gpgWNUFA.jpg',
 'BCtBaUlY6VNtlgRpL-wflQ.jpg',
 '9iTokAQim8nmoC9VQYezZQ.jpg',
 'g35O85-ZvqNwcXNb-ZOPTg.jpg',
 '5wiKAtXPOn6DWnjGvgnLlw.jpg',
 'iN87jUCb8ERJ8iA43p1jiw.jpg',
 '9iX2Q3u4iBrc8Fw7vGBR0A.jpg',
 'Vmy_wEGyuXbNdnufXx3B3Q.jpg',
 'j_s2YLESIvdeYH4WTIODiw.jpg',
 'EF-bTcaTzrqxqRms5v_t5A.jpg',
 'C9xnaI-PejlqsQ351As8PA.jpg',
 'BBOKTZ3Q_RpRsNQnu8nVhg.jpg',
 '6aoQ54LTDfzQGAQ8EtyJPQ.jpg',
 'l3XctYDNYmraqWlgZXBlhg.jpg',
 'dWuQqwrI2TdOHjcOyXC1Ww.jpg',
 'JtjwHGoKO55uNNtYPMGghQ.jpg',
 'kJuaizqi1EQby8reWHUUzA.jpg',
 'JtZXLEHg_P0o_yZKJW-KBw.jpg',
 '25s_uHWFxNJpFMSZxhUvYg.jpg',
 'q1vzzAw27Zwd2kEPQhOyUg.jpg',
 '2RCJd2RLn0DPUYEBJfjs4A.jpg',
 'pucEBB_95l07ES_c1fr_DA.jpg',
 'qNlZsDYiG3dcQxFlW-ft9A.jpg',
 'qAXDTGtpDTX8cBWkTRwXbQ.jpg',
 '-U5al-KGf3bQ91fic-neoQ.jpg',
 'l4xG1q6QlBAmAoxEHoKl7g.jpg',
 'QupBHF2ETQbexK0_LZyRLA.jpg',
 'HbGyTe2Qz0pDGYc2jLr1EA.jpg',
 'zohE4GrPwa48pe6XOzyFHA.jpg',
 'IfSbjKnuExrFFeoxhXhb7g.jpg',
 'Dz-Kh6NK5Qye20MPrEhknw.jpg',
 'YPIhX9RQ_7Lqpzr0bKai8g.jpg',
 'WGr-0JbK86KNunfScdDO0A.jpg',
 'xiOgfmeKVya9UUpbmByoDg.jpg',
 'hCQ1AnQSAfAJsVAhRSv45Q.jpg',
 'NFBNPbZQ5bQsqnP0O3H8zg.jpg',
 '9ELmIyZ0O8q4sMWDKKz7JA.jpg',
 '7_a8diWtJPUv8uf0A5MbQg.jpg',
 'aqFgA2D1kYjUCg6kZFs_-g.jpg',
 '8htRg1J0WiErWOL-kr1hLg.jpg',
 'bIRNdaTanDe1eKyG45aXxg.jpg',
 'KFnyK7CGUt6kfGFxNYZI_g.jpg',
 'GhgErW7XwY_y4kcwj4LCsA.jpg',
 'm8XK6sTm89E7wc3kJVyJqA.jpg',
 'FuNVwsfTrL2sNLbZ_Mi7qg.jpg',
 '33rjYQY-vTENf9j1mUztSQ.jpg',
 'seX5i-ugjV7EqD7SIfYepg.jpg',
 'gIE5dkU7ynCTa79dJlDqKg.jpg',
 'WvMrNwcv82pH_r9Hq_6zrg.jpg',
 'qljd5WlEHtv8G-qhePIsGw.jpg',
 '6PArPoiT8TZ7XVsILhA-JQ.jpg',
 'kbpFEPbXJC0Bp2UppTNOLw.jpg',
 'y4aYq051LcKidAeo2MqPjg.jpg',
 'UDOL2Jh97GMwB3RuZKjEiA.jpg',
 'cHXhhBQBvJv4v1rT6UJpOg.jpg',
 't8ZHPPi0b-8sJjbWsAF03A.jpg',
 'l1qZCjaCScSgLbOuI5JGmA.jpg',
 'evgvR5bjtrOKSZ9BHsuEQQ.jpg',
 'rDMiwfG8mgEvB0Ky174Udw.jpg',
 'OfgNPrbFmgbmybUK_ucZLQ.jpg',
 '1VL2rEm6mrfysAPWo5uaAg.jpg',
 'Ry9E0hXIAbqhRjZiCEgMqg.jpg',
 '8JJ5a0VQSFK0npVrWOEpsQ.jpg',
 'lxr9akjavvJf8d4AmgUN8A.jpg',
 'z98y6xn3lsLyhiM7KYx_rw.jpg',
 'h0s2G3NdYXkYiZaXwPiFvQ.jpg',
 'AO2QvYE6hO72FZ6k1q0-QA.jpg',
 'D1FiEGBDDPwTExN_LlgZjg.jpg',
 'x0Cn3GHK9UUfuE-KhUQj4w.jpg',
 'a-ts2dG1EvtoecEIvkjhKg.jpg',
 'ygv6pPC-x-kqPFRP77P3dQ.jpg',
 'WqfGWV-ZLKIVHeojfkGw6Q.jpg',
 'sh3dHcOFiYuKcBtnDUrL9g.jpg',
 'OmfuJhj4u6T9jyLqjJqtng.jpg',
 'OzRrbQdqUqxRUvEx3OBNqw.jpg',
 'sBxno97Mggiz_IM8tTbMMQ.jpg',
 'UcJtsEH-4X6St6gdzXs0Jg.jpg',
 'N-3HDH2sAEWIjCii0YROAQ.jpg',
 'IkId3MfAGkwiXHodN9yl8A.jpg',
 'DsgJhzPufisCj_H4eZMhKg.jpg',
 'ANifQhtwllLjx4PyOO7w4A.jpg',
 'J-M70x7pk8aCCascwKFThw.jpg',
 'MUSvLb_msby76tJRKqhNNw.jpg',
 'YVRaFT0RZdju1GPRPSZh-w.jpg',
 '1f-gzbnfO0mcVsdlU-O5Kw.jpg',
 'G6bK6OZIx6epDwa8njIcXw.jpg',
 '6pqsa8IFjeFGLMZUBl6S6A.jpg',
 'GIKekaQz4AWC5CsModIpeA.jpg',
 'u88AYush5R5BsobPscxZPA.jpg',
 'l4HKDlwL6hYQO0GJXkMvCg.jpg',
 'yZp3V-iKRCwQH27e47Omzw.jpg',
 'XUwl-i8aYH1vlUKdNgq5aw.jpg',
 'B9rzff1dAL15uakk2sY3HQ.jpg',
 'MklXpHM6LXd7D4VPt5r79A.jpg',
 'IUr0OKfGglr9o3IL2Pg0MQ.jpg',
 'Yf1zWjKMpPtUI5oCI5FSyQ.jpg',
 '3sRRz_Tvxh-D30ReAWPJZA.jpg',
 'z6DKBdwbiDTy8m7kIbj7_w.jpg',
 '5whbVMF4Svq1b7LBc80wlw.jpg',
 'egQcSLN5Zot7h5mDN0C7PQ.jpg',
 'L6Hgk3uS4E98InIAa8JuFw.jpg',
 '_3GxSmv75TesUqUYXg3mdQ.jpg',
 'kP_V02koaxEPiJJ95Vqv3A.jpg',
 'RG2RzJNLvnxxBVZBHKhvhA.jpg',
 'ksRPdfbfZQjaKYkmfgetgA.jpg',
 'dxJik8WeaXImnftk2RrnWg.jpg',
 'D8h2A5kkemblHiQcj-tk-w.jpg',
 'Dpbo0gRjGS8_sTkV-xIf4g.jpg',
 'WXjXcJ37UX3Zil5BTig6vw.jpg',
 'UTQxzKiVUOGdkGukCCyEFg.jpg',
 'UNAuZREkCRGeUiiH65qLkw.jpg',
 'UeUxdQXrArRm0Y06ioK8hw.jpg',
 'N4PukgRMKwqb3SDo6OilPQ.jpg',
 'qMLj-kS7LYQ9-S3SrH6NmA.jpg',
 'escncn7t3rGJcPJePFD9ag.jpg',
 'UnqUy6eKa6Cd0GxVL_jkaQ.jpg',
 'Ls0xi_2W3BusPzZO02e_OA.jpg',
 'IS_QuvqQeGOFPgnhVyXPmA.jpg',
 'DAjLBZIpGBKTKhGSfZf3Rw.jpg',
 'KcwAkqXu0qw2ndXnkyokjA.jpg',
 'kMH9nMvG46qNlEjF0y5sFQ.jpg',
 'cZqCbbaAV0wQ8kDH2xPbJw.jpg',
 'EiMnuoTfv0lV0S8wKa0I_g.jpg',
 'LHE3uO37JK8N8KYXpC5Ncw.jpg',
 '1eU-r6GIjzpc1noqQseC-Q.jpg',
 'inmUgupvdGq0cvEBLzqQOA.jpg',
 'ZVeKSfWuYzPIfl-rEbfXsw.jpg',
 'czIV0nEp3SX-Z8ebEJFYWA.jpg',
 'qHQ6cwruMsMVpjtLFDfCWQ.jpg',
 'TEvLbdBwiupcF6ucUhK9Ew.jpg',
 'IqqjYCyx6g6YZmI37mtcvQ.jpg',
 'Hwi5b7mTwXe_7aPMLZC6ng.jpg',
 'NrVxAK8elDNXiEphiRFE6A.jpg',
 'GjTgBi6EJjK5zm8L7EjKkw.jpg',
 'zUwS5RcSxUSZ43pl5ybyWQ.jpg',
 '-ytb_bVi5f64FjtZYeGyHg.jpg',
 '1RQEqNO6pk6GtRVeegdBXg.jpg',
 'aNFeqeu4WA8kI-NrjEsweQ.jpg',
 '1NGH0NYp0RA9jw0AS0KC0Q.jpg',
 '7ao7FRzY-mpzuPDjcKh_lw.jpg',
 'HTXaXXwP2VwqvP5iS6Erxw.jpg',
 'usEJ127gLkSaxZMvdbjBaA.jpg',
 'QqhRHzMoLNgCy4YWTfrvRg.jpg',
 'zM-pOBOHUoXsaQAZtJZ7Aw.jpg',
 'eeT7ZwWD7-7Xgc-69QVNdg.jpg',
 '8YZ1rtrDICOZP_LyHtMrDQ.jpg',
 'S4KxO3EZvV3BdgWoO1evOA.jpg',
 'g0q8oUDIFRcVrHI6q8nHBw.jpg',
 'cgNYzfE3uPhOoyq-QE-Syg.jpg',
 'rFZ8intyx8egmqD3Alp0zg.jpg',
 'rRVPYbima0SiYzNG4gKMMg.jpg',
 '7vjJTH4jeN_j0eW8zVJcUg.jpg',
 '_p3RAdLLWd7FmoRqTnK8_w.jpg',
 'yZLQlEIL4VFY7eE2ip-ILw.jpg',
 'UUhULyRFJ52uo9xDez0j5A.jpg',
 'I3fUdZFtTZN6RJElTRi4Xw.jpg',
 '48ykpPnW2XJMueikpwpdMQ.jpg',
 'HUme3xJY-UwpMtg5ABgRKA.jpg',
 'T2YTF3kn26Sp2Gz7Prnb8A.jpg',
 'kUBRPU_EyKd4VNQfXhexIg.jpg',
 '2WWrb6ocT4kB_V7a4iDtbg.jpg',
 'qf1E7pgk6v5BBWtinBsYdQ.jpg',
 'CY-nbyhpnyuwOmHPlqZ7LA.jpg',
 'jtpskFy0i-wP5k9vq1NSEQ.jpg',
 'HKJoqcW9aWfM-LecxgnfIw.jpg',
 'J3ARFzgLgco30BXy2X6yIQ.jpg',
 '62SXu_Nb3HO5tDkFneH-dw.jpg',
 'wK8FjS2cocrUdpBKM9iXnw.jpg',
 '6yr0l1iJodJ0k1HDSPclXg.jpg',
 'NUPQbYgCFC-7ZaaGp52IXA.jpg',
 'E5zzrZQVcnnVngbzEtpgXQ.jpg',
 'dZqkJWI7ofhdiwPRWkFFEA.jpg',
 'w1N7OHtb1mwsQpAqthDqfQ.jpg',
 'grTEr2YhkVQ-w9TgY2x-7w.jpg',
 'kSe-Mc9ORVgr_PPAKARkjA.jpg',
 '3oWal0PJqmrzlDBT5tvt3g.jpg',
 'gQVuCucQ_i49DMTbs_k0CQ.jpg',
 '_tpqtB5dnAqqmeoKFMwJCQ.jpg',
 'JO1WQexvBuiESkwnejordg.jpg',
 '8rAGtyZwFscUHQ6CWXI7cQ.jpg',
 'aWurnfh9N0UWBNxKeY24IQ.jpg',
 'rKqTXzV2F4ndArKcWCmKUg.jpg',
 'Q2lyblE_FBe72MoS77cSEg.jpg',
 'A3XTQywhK5pNBB8ItHxO-w.jpg',
 'Gm8aelmkB2YLWj04DdA9Zg.jpg',
 'uPTz7_zn8eF-IV0NUuxrqA.jpg',
 '8cYqBF9HCEtSCTn0DBESNg.jpg',
 'erWeuwBarkp5m12dRuRHFA.jpg',
 'SFco08-_862TPbMvhiNvdA.jpg',
 'trZIjqDjnurjbiwvyCO0eQ.jpg',
 'uSXH4RgqxVD0Nz4iwt51Aw.jpg',
 'YZRpy8agZ88kk0VYnSUeuQ.jpg',
 'BvSltD7-hruAfACSrxXGRA.jpg',
 'xNn-SlSQ4U2fjFU1TF25Qw.jpg',
 'tJf7_8KpgFvCzmraq7iNtQ.jpg',
 'M7wNjbYHBUcv4vOEA1OYcg.jpg',
 '3AKEC3eZ61V05vssoENkXQ.jpg',
 '0X7RV67cJV4wyLXUSO6aMQ.jpg',
 '7wrB3Zcdb_-BAUXDeb_mjg.jpg',
 'f1iwleFJalYcER3fvHvXsg.jpg',
 'tRx2k4MSxNrYT7mWBNhe_A.jpg',
 'DUw_F7RgpPOoibvbwiHC5g.jpg',
 '7n-yPwfxFca8iAdYd7ONRw.jpg',
 'nDPavdr-MeqlU0ZEC9f0vg.jpg',
 '17F1yb_F2QFNuEOtczlQlQ.jpg',
 'FV1D5bxmJUfTgbJthRIZAA.jpg',
 'vUgd-NZPUUSvtQtQQOQoCQ.jpg',
 '-HvXD2dM1tOmrfu7t8dw3Q.jpg',
 '9moXRHzEyZe-NJqv-F2K3Q.jpg',
 'e_ObEugJQ_t3MhRpoHfOmQ.jpg',
 'WWf-l_wmnqSD8y0G5tUzdg.jpg',
 'qnV2dYe9v0Seza3UTVCydg.jpg',
 '1vLIspMspBtBOdnvj5k3ZA.jpg',
 'v43c-MO4q71qgLqgRc5woA.jpg',
 'KPbxgo0CO3Pc6XCIQjYIQA.jpg',
 'SiKPMkM3bzSyUlQ214n0tA.jpg',
 '3Ehqt5KVi5KDYg8-Kn4h_w.jpg',
 'CLTkJSm_W8Vq3iUmHqIHrw.jpg',
 'bAGJB3ePoOFNV4DjKS5KEw.jpg',
 'VLFND9PKAIsw3mJCmS-WvA.jpg',
 'DL62G0qw_XNtMb2i8zNUNg.jpg',
 'Q2sthnaUGnpQIAu9uCXveQ.jpg',
 'nj0KQknjjfEwp0ckqh1jLg.jpg',
 '39TF2w0nylYS2_Dn1Cje2A.jpg',
 'Y_v7aSaduf5CBsRWTl8mFQ.jpg',
 'EGaLSaHo8tF6-YKj3x69yw.jpg',
 'wOjll2dsDcZMdRJcLOpwXA.jpg',
 'tXyRdypwC2jDbtdAv7SKFA.jpg',
 'dq5QD0Xn0xXGXXgE4D4k4A.jpg',
 'StbPDU7RogeDdEUlawm_iw.jpg',
 'qP5B3h0uPULP6BxIjOp2QQ.jpg',
 'dEFQWjMqcjb-RidjP813iw.jpg',
 'ol_OPyeAe8CsCRmF1DOfyQ.jpg',
 'TJKZvkmN_w3eGq_cwzqx5A.jpg',
 'Tw5RcyoJSEA-5z1q7Tb6Dg.jpg',
 'D6odU_UZBHdeZmOL8d4ikw.jpg',
 'OoLW2Wm5W63nZf8cP_DtBQ.jpg',
 'sqB5U2k751SeJ7kigXu2ng.jpg',
 'DFPzuP6AGyyepAXszE1YQA.jpg',
 'pC072jFBo2Xu2ayJcrUFtQ.jpg',
 'ldWQro5_enRvqo-GaMHY5g.jpg',
 'FrhKryVrPmfaUg2T22l-MA.jpg',
 'HrzNhpFgFJPwT-hQVs5Y9w.jpg',
 'z-R5g-5kVsWccI6EMBW6rw.jpg',
 'k6hR6Kd83vi6tQBHrHOFqg.jpg',
 'aVJMePFLS9qUwAK8sVU5Qw.jpg',
 'pGvtsJwAm1wIf7Bbxjmhfw.jpg',
 'FUzS4qBJPUB5aV4OOCD9TQ.jpg',
 'd1uWdNacXcs7pmmW8tUWlg.jpg',
 'oJyA-OGYRosU8oWU9lK05w.jpg',
 '1zRqMUBwRVmLnpbU2cJcew.jpg',
 '_CaFZBwW8E_jhkSYFAhXAw.jpg',
 'AlNJn6HICRSy37xWzkZZEQ.jpg',
 'AqM4ioAdnlblA6ED8G_jDQ.jpg',
 'GFnOEgFfvqlzimpCsDNZEg.jpg',
 'cnpuFQ5MVvsWFBoZa3mxSw.jpg',
 'ZvjtvQXqRx6Ch2AGffDxSw.jpg',
 'tHvKI3UTO0xibqW3X3377A.jpg',
 'UyRN6BsLRn51GhsSpE-Evg.jpg',
 'hW2pVJOb1Ab3LvYPwWLBsg.jpg',
 'KrTANhcDYKhlt3WDp_KN2g.jpg',
 'UGznH39DS9LQQ43ZPpfTxw.jpg',
 'ELMO-auPmexrubYc6OeJ9w.jpg',
 'vqcdBYrbCWgUIyPtLSJWUw.jpg',
 'yhuM4lVZaCBaMiZoR8BTdQ.jpg',
 'kwGuzmq-35352EwiRKkZ8A.jpg',
 '-lO_BAglgu_kqoVLk9QD8w.jpg',
 '8l-l8jG2t0WJySMG9vT_IQ.jpg',
 'ibHOUmLUYcwgF2EIUglS9A.jpg',
 'VpraZO9PcB7oBES5fvnG4A.jpg',
 'ryW-wB-dZxEUMvRz2InUVw.jpg',
 'VTb0djEULo0WH4J_ykc3DQ.jpg',
 'meVSMeO4rPtG3mAGteWoHw.jpg',
 'ARt_rfRUqE3O-v_jLGtvbA.jpg',
 'fo3_JIfh8ZUyENjzbBvOMw.jpg',
 'ZeDGPb7VtwKRFFr7L6VNRA.jpg',
 'i3qWGbH3FhIJ1J3eb8ITEw.jpg',
 'HA_755-RofdycEsIMVuIbg.jpg',
 'mRU-CQZLUHFmONOpgEKL-w.jpg',
 'dWlDDibOredeDUjhFoWleg.jpg',
 'nRnr_l4PKIsPCR6_xuK6nQ.jpg',
 'yMJFSXpoX8hy9ryOxZ-G_g.jpg',
 's-z2cM5lZLtgsX4HruU8Xg.jpg',
 'yCE9tZpkhxXSvkxgHzDFbQ.jpg',
 'DKQA6caL6iri9uAmzpMOhw.jpg',
 'mZ_MgExGOpIfMDgySVdHTQ.jpg',
 'ysBeaVRWIbldlWSqvUbfwA.jpg',
 'VnxcAq-4dRgzySne-qbV5Q.jpg',
 'ipF8W51CRYjbHWxRY7t6zQ.jpg',
 '65uaPrlXFh5yvdSrRkKyzA.jpg',
 'T8z9LQMfNOHq5BZ4HRPU2w.jpg',
 '44fPAY9REcmmAnLgnpLR_g.jpg',
 'tsXEwXmgXZwzV6lpp8TPTg.jpg',
 'BofNAIkTwlJQP4cvQGg9AQ.jpg',
 'Zk4sc-JM9Fz9CKj6miSTjQ.jpg',
 'dAnBaEtqzt7-0XtlCRyRGw.jpg',
 'yYbCZpI1WLAe_ypQaZYTKg.jpg',
 'LdOJy4qtlEuyZtgHkWqMeQ.jpg',
 'MPti8l7P2CPafSdvKfT2vA.jpg',
 'gZeF_Hc5PPDwDbI65fMQcQ.jpg',
 'w7YgcqWPcobW4_zaVxrSRQ.jpg',
 'smusA1_vOHJDZo1MK0lGpA.jpg',
 'jePcCSPAch4n-jVCg6_5eA.jpg',
 'IYO3hQDR6TGE4PHyf7x22Q.jpg',
 'zjcf4QemzeHDS1N1dY2j1w.jpg',
 '6qc0wCm_8cfCvIWFKZARRQ.jpg',
 'PhIPD5fo6K0LcaVq0Sf96A.jpg',
 'dlhQRDoUHv2BwHnrpL4C9Q.jpg',
 'AJWjvE86_hM7necXxB42zg.jpg',
 '3yjOnYX1IKfwCK32U1S0vw.jpg',
 'p5koVtAF0CxtPFxshzFiiw.jpg',
 'O2cPKBQLBrS58k-tygxLtg.jpg',
 '7iP7ZOX6tppoqQYR3NXLDA.jpg',
 '6BY4zutf2kFrrdBSwrtOxg.jpg',
 '3EanA-NeybfaDLNUqe0sZw.jpg',
 'j6wJR691ZltTwO6y-3g4cA.jpg',
 'S3EPJb7FN2wUmOyQeRIDXw.jpg',
 'kOkKo_6Hae4OKSIDyscdyQ.jpg',
 'YBAOtOxqnlJyymStLt3_Hw.jpg',
 'JNr2l3LXSvRgEYwHGXyjvg.jpg',
 'yyIV6-1qRDA5StG28foZ3Q.jpg',
 'JqJmHMPfPM1r32xD-FBLrw.jpg',
 '-qlpcMl-yFFy8-JxcQ4waA.jpg',
 'JcdDkPnc30esIAmgtLQEVw.jpg',
 '7O_qIOm5RD2v3k6j281g9Q.jpg',
 'r_RgqJK9zleCRo2iDBv6vg.jpg',
 'Y_NZz-2-4py4sCRkow6u4A.jpg',
 's2kbnsuCeSkKnMJlixmyvg.jpg',
 'SUw4qTTla7FQbOOpMIOt_Q.jpg',
 'X7zyEff-cGX7oC2x5JZQ3w.jpg',
 'KOyMWW-h3VYVtEDej9kh4A.jpg',
 'RObFfqaVUZqJ9s3BrjeASg.jpg',
 'crEknZ3lKYeSGYJ71I8_0g.jpg',
 'FBbirU0FChLFrIsoKwjKjw.jpg',
 'IiJrVg7jCb3D0SW3jKhxkg.jpg',
 '66HuF-7N6ru-aw4nL02wgA.jpg',
 'PyiJwGwCex3EgYPXbFxbCQ.jpg',
 'dD34TfuDZVC44MJh_RTxlA.jpg',
 'awZoHac4746e2Eamr2hjIg.jpg',
 'QZUq3UOpUxjVj7v97QHQZg.jpg',
 'yU86PX1r5KHcovZmztohVQ.jpg',
 'p3ctFtguKEV7WeKubiFH4Q.jpg',
 'H5OFYZ_P2q6Ui_2d319okA.jpg',
 'QEwNSi4u_xuNfMlezT63NQ.jpg',
 'fghHmGW5OLFVjwpNySEdrA.jpg',
 'P-aEf4_N3GqVyEn6S7viBA.jpg',
 'kk64Db37wI8TIfXP61tgkA.jpg',
 'Vo5qQ63jx8D_VswMN5x_aw.jpg',
 '703AlMZtCL-S8RFLcW3e9Q.jpg',
 'L7-vQNheftTxmSuIE9UXHg.jpg',
 'holUhhPczyxlU3iA1v3zsQ.jpg',
 'Xp2E4tbu3Y222PPN-xie1g.jpg',
 'ELO1J9czjUede_Jr0TfcRw.jpg',
 'y07fx01vXfrq9rcUZQvd5A.jpg',
 'g04Ffdke-Ce_yQ6Ncw9pvw.jpg',
 'p3lg6WYWHWtXqlJX-xCGIg.jpg',
 'Db5TWRM-Td8VtUnwoIISzQ.jpg',
 'vMrQlBoze4FbSdjwkcCAWA.jpg',
 'poavBAadJ80IqIHXTWe1wQ.jpg',
 'JOO7VtJ-TilhYgIJ9ScUoQ.jpg',
 '-FxqhyHgf8B_ng2hpXcdhw.jpg',
 'gow8vY6C3wgXB-B77im9qQ.jpg',
 'ZM9mu6vZeUKLaWMN4KASzQ.jpg',
 'A_RzHL_2Em7o4NvUMPyarA.jpg',
 'ByOBHQ-Ap89aF5kQXEa9Aw.jpg',
 'ezgj6RD0EZBI4eoIIESrVQ.jpg',
 'o9ITH_n0xER6whUKqfKpmw.jpg',
 'gneMa6st2yscYy0N3f5rxg.jpg',
 'A_4iro0ZjX7u0p8Dgk6kiQ.jpg',
 '4KsdP5ZY9qse-cqhPSULQg.jpg',
 'jsz2ya0oztu-fnBol10T7Q.jpg',
 'r_UTJnwHgWSyrN1hhfudoA.jpg',
 '-BK-qRMCNCS3juVi2M_RRQ.jpg',
 'GPfmJSdD_wkX-InaHENNhg.jpg',
 'C0JT9QjaouEmdBjrvk4YJg.jpg',
 'OEEVtt9USTAOjYo-2N0jDw.jpg',
 'Kei_VvIck-vg1uSNzkz_bA.jpg',
 'iv3wJqHuLx1WsbJjRzJiyw.jpg',
 'jY9uOLp06vVD4fWibNJpwA.jpg',
 'RRCCdb01QzkJAJEKcr45Vw.jpg',
 '44sBbT_vMn3HZMooAtS22w.jpg',
 'F-A3Z5Hi3HbyCHx-tkRWlg.jpg',
 '3mN11sRI5_hMLTqixrjbBA.jpg',
 'BJPmyRH-K_ZJKhX3WLRGKA.jpg',
 'VuLvXPp7lB3Gli3qCbYwNA.jpg',
 'p26C0N1FHvuK2e8LzzrEeQ.jpg',
 'FtL_a9fcC3T3llwURI9rWw.jpg',
 '7snD0EkqXy_DdRvEz13wUA.jpg',
 'an0kc_yrQdqyuz9_uIWY6w.jpg',
 'vSbtm-xLMk8BRJpPiBlUyw.jpg',
 'H4whAchT9CQztZ0XC_RObA.jpg',
 '2Fv5N1XAMpzXdrR2-XBvpg.jpg',
 'GeS55QIb-s8IfUQ-xdgTUg.jpg',
 'I-qNH2fTbFjFvxw5p-oVPA.jpg',
 'vsT2JOv-SOXh3C4o26Oj_Q.jpg',
 'g7ogTtAk21Q7pxzxM0FX9w.jpg',
 'WH1l4-S3i8GKSpE5y1ao7Q.jpg',
 '8Rb7sAODAW9NKUA1snZbBQ.jpg',
 'wlXXHjgIjJ9k__4jRC-2XQ.jpg',
 'MSb4Sg4DedTO0t4SFqLb3Q.jpg',
 'RHTJE85rjZUjr9BuBvGqbA.jpg',
 'Jx7DH0ObLjD9xiZU5CcQ3g.jpg',
 '5S_i6PQXMYN3jDTY_Lpl0Q.jpg',
 'VOsFQcwTPfoordreR1aXtA.jpg',
 'Z8Dww4NRPfTehdzhcxdXLA.jpg',
 'a0Gj9KDp52JZ5r-QIFlrbg.jpg',
 '33tAkMv6Y2ToY7Td06v7DA.jpg',
 'tLcvNqRpMk5R5IkLE3fPuQ.jpg',
 'Qf2wJQaTkD07Ur8k8hnpXQ.jpg',
 'A15M3otwaIgiIkrQloQ5Uw.jpg',
 'wFkLARTyM9wzAse-xffWhw.jpg',
 'tFNqMx1EhYKWIE4vVGZbPA.jpg',
 'uNMozkwaRa-4rsF2ql0gFQ.jpg',
 'EWJEiR8yl-Seqa3vzovkeQ.jpg',
 'C7esExIVrJov3WcMG993Nw.jpg',
 'bclLZdPITQi2PGO5VOVZ8A.jpg',
 'Bw-YTosctwR1pwcLxUJX2g.jpg',
 'eGyqoYAdz9Fc93kONAC9Ug.jpg',
 'jDZ8qlzLOPI4OrserSY-_Q.jpg',
 'fpmyHXBa3pPwtFVRNz3XLA.jpg',
 'B1g8hpa0CGG10YcHBy2pcw.jpg',
 'o7xLznrbQypA3CJT_cJFJw.jpg',
 'KOwhDZluy3ftF5hm5BWShQ.jpg',
 'suw3rJ7ujmXhmYr9cC93cQ.jpg',
 'usT0qv1OXsE2ohE_9M_6mA.jpg',
 'l7cihOdW8wHqQqd1g2bh0Q.jpg',
 'zhZw9fw8jh84BIXNnXop5A.jpg',
 '1mYchmq-1eLQFERNd1q8MQ.jpg',
 'MBRGJEIFawqb4NZ_nUopBQ.jpg',
 '2HNvJ5rQ6EOCJm74mpQNIQ.jpg',
 'cJDF7Nv40LtTbxyiuc8b_g.jpg',
 'OiGPipylFnaBEmCcs-Y25w.jpg',
 'na04dVEWaiQ8mDenR58iTQ.jpg',
 'bvS2Zy2zxofalTjr8ZKtDA.jpg',
 'WdaKNQlIBnpKzvEmhj9W8w.jpg',
 'x1p4Lc6oQL8prveBz_vAQw.jpg',
 'UtRU8O8HLtmh-sBpwOmliA.jpg',
 'rRppvxfj84ydIospXpeb_A.jpg',
 '-HXGYZ7AEY716p8vrG9fWg.jpg',
 'hNc1E8pWAUvCYYJB4UjkmQ.jpg',
 'X0xSuvPQ9907YY9WhBnYHw.jpg',
 'Qh6WN3drPL2DNcyoKJzCRA.jpg',
 'Ofl9XmQR3dMLQUKUmS9wsg.jpg',
 'wuPO4FDJvpEzsliixLWI2g.jpg',
 'xI-ixFjy4cQuGkoySvodXQ.jpg',
 'U3CcLGz-vpKXEpBSGWD1Jg.jpg',
 'SpRRrxFvJUo4_Gy8cR043g.jpg',
 'lRSZffDkawz7ezkNj0aFfA.jpg',
 'gkz2d7pAOWrgQ_6q8ISKhA.jpg',
 'CupJBNjGGBVNphHWeDoFyQ.jpg',
 'DyqEled6AkMI-qAVW7D6Lg.jpg',
 'INWi0ja-eoozNYhthMJcew.jpg',
 'bwR06Uj0VfmdUkw7Q2IgvQ.jpg',
 'PzJLuVXDmfqxABI6740VTw.jpg',
 'Lp5A2NkSBgGMC6cAbPf9Dw.jpg',
 'VuZfMVFaKARq_85IhXVLZA.jpg',
 'lPQwjWm-mgO5XY9njTOPhg.jpg',
 'Nb7HfDjTR1MvqdgD2zJyMg.jpg',
 'MRMLG1aJEybyzUcvlGC0gw.jpg',
 '6lPURNnuquoyOMt3jGT21g.jpg',
 'qg14eL3mnJ1wlSjwIpKLHQ.jpg',
 'Zu2OiQ9iA3661iOAddEKVQ.jpg',
 's9Ov3dLHQZgsQ9wjvGuFRA.jpg',
 '4NS7phpwTi8CgES_mK4Qmg.jpg',
 'TOn9BSDk4baNUgKcNTfniA.jpg',
 '38ydUvD98lBYn5qvpVtLdg.jpg',
 'tmmxO5PInBw3_QeS5DFTDA.jpg',
 'VZEIETENP7yh8Z9Pp8pl4g.jpg',
 'hLFCzL69kD2hCGwoVcXdZg.jpg',
 'hd8oEHx-xjSp5f-Bm4d3tA.jpg',
 'zHhHKUjeg9tYDuWrsUl0bQ.jpg',
 'gJrHZ8Qf9euZztc2ZZBTng.jpg',
 'cHIhpaNpqy8tWoIn2C_TPg.jpg',
 'oEw2XHqV-UoFssvigIqn5Q.jpg',
 'ENm89bWyMLdFhrThb_RcAA.jpg',
 …]
```

3.2 Détermination et affichage des descripteurs SIFT

Voir os.path.join pour rendre propre la concatenation du chemin et du nom de fichier

[27]:
```python
def build_des(pict):
    sift = cv2.xfeatures2d.SIFT_create(500)
    image = cv2.imread(path+pict,0) # convert in gray
    image = cv2.equalizeHist(image)   # equalize image histogram
    kp, des = sift.detectAndCompute(image, None)
    img=cv2.drawKeypoints(image,kp,image)
#     plt.imshow(img)
#     plt.show()
#     print("Descripteurs : ", des.shape)
#     print()
#     print(des)
    return des
```

3.3 Créations des descripteurs de chaque image

[28]:

```
sample_pic = df_pic[['photo_id','label']]
list_pic = sample_pic['photo_id'].to_list()[0:sample_size];
list_labels = sample_pic['label'].to_list()[0:sample_size];

# Saving df_pic into a csv file
sample_pic.to_csv('pictures.csv')

sift_keypoints = []
for id in list_pic:
    sift_keypoints.append(build_des(id+'.jpg'))

sift_keypoints_by_img = np.asarray(sift_keypoints)
sift_keypoints_all = np.concatenate(sift_keypoints_by_img, axis=0)
```

[29]:

```
print()
print("Nombre de descripteurs : ", sift_keypoints_all.shape)
```

```

Nombre de descripteurs :  (1469040, 128)
```

3.4 Création des clusters de descripteurs

[30]:

```
from sklearn import cluster, metrics

n_categ=5
# Determination number of clusters
start_time = timeit.default_timer()

k = int(round(np.sqrt(len(sift_keypoints_all)),0))
# k = 10*n_categ

print("Nombre de clusters estimés : ", k)
print("Création de",k, "clusters de descripteurs ...")

# Clustering
kmeans = cluster.MiniBatchKMeans(n_clusters=k, init_size=3*k, random_state=0)
kmeans.fit(sift_keypoints_all)

elapsed = timeit.default_timer() - start_time
print("temps de traitement kmeans : ", "%15.2f" % elapsed, "secondes")
```

```
Nombre de clusters estimés :  1212
Création de 1212 clusters de descripteurs …
temps de traitement kmeans :           208.58 secondes
```

Saving the fitted kmeans model

[31]:

```
clusteringfile = 'kmeans.sav'
pickle.dump(kmeans, open(clusteringfile, 'wb'))
```

3.5 Création des features des images

[32]:

```
# Creation of histograms (features)
start_time = timeit.default_timer()

def build_histogram(kmeans, des, image_num):
    res = kmeans.predict(des)
    hist = np.zeros(len(kmeans.cluster_centers_))
    nb_des=len(des)
    if nb_des==0 : print("problème histogramme image  : ", image_num)
    for i in res:
        hist[i] += 1.0/nb_des
    return hist


# Creation of a matrix of histograms
hist_vectors=[]

for i, image_desc in enumerate(sift_keypoints_by_img) :
    if i%100 == 0 : print(i)
    hist = build_histogram(kmeans, image_desc, i) #calculates the histogram
    hist_vectors.append(hist) #histogram is the feature vector

im_features = np.asarray(hist_vectors)

elapsed = timeit.default_timer() - start_time
print("temps de création histogrammes : ", "%15.2f" % elapsed, "secondes")
```

```
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
```

```
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
2300
2400
2500
2600
2700
2800
2900
temps de création histogrammes :            20.86 secondes
```

[33]:
```
im_matrixfile = 'im_matrix.sav'
pickle.dump(im_features, open(im_matrixfile, 'wb'))
```

3.6 Réductions de dimension

3.6.1 Réduction de dimension PCA

[34]:
```
from sklearn import manifold, decomposition

print("Dimensions dataset avant réduction PCA : ", im_features.shape)
pca = decomposition.PCA(n_components=0.60) #30.04
feat_pca= pca.fit_transform(im_features)
print("Dimensions dataset après réduction PCA : ", feat_pca.shape)
```

```
Dimensions dataset avant réduction PCA :  (3000, 1212)
Dimensions dataset après réduction PCA :  (3000, 141)
```

3.6.2 Réduction de dimension T-SNE

[35]:
```
from sklearn import manifold, decomposition

tsne = manifold.TSNE(n_components=2, perplexity=30,
                     n_iter=2000, init='random', random_state=6)
X_tsne = tsne.fit_transform(feat_pca)

df_tsne = pd.DataFrame(X_tsne[:,0:2], columns=['tsne1', 'tsne2'])
```

```
df_tsne["class"] = df_pic["label"]
print(df_tsne.shape)
```

```
(3000, 3)
```

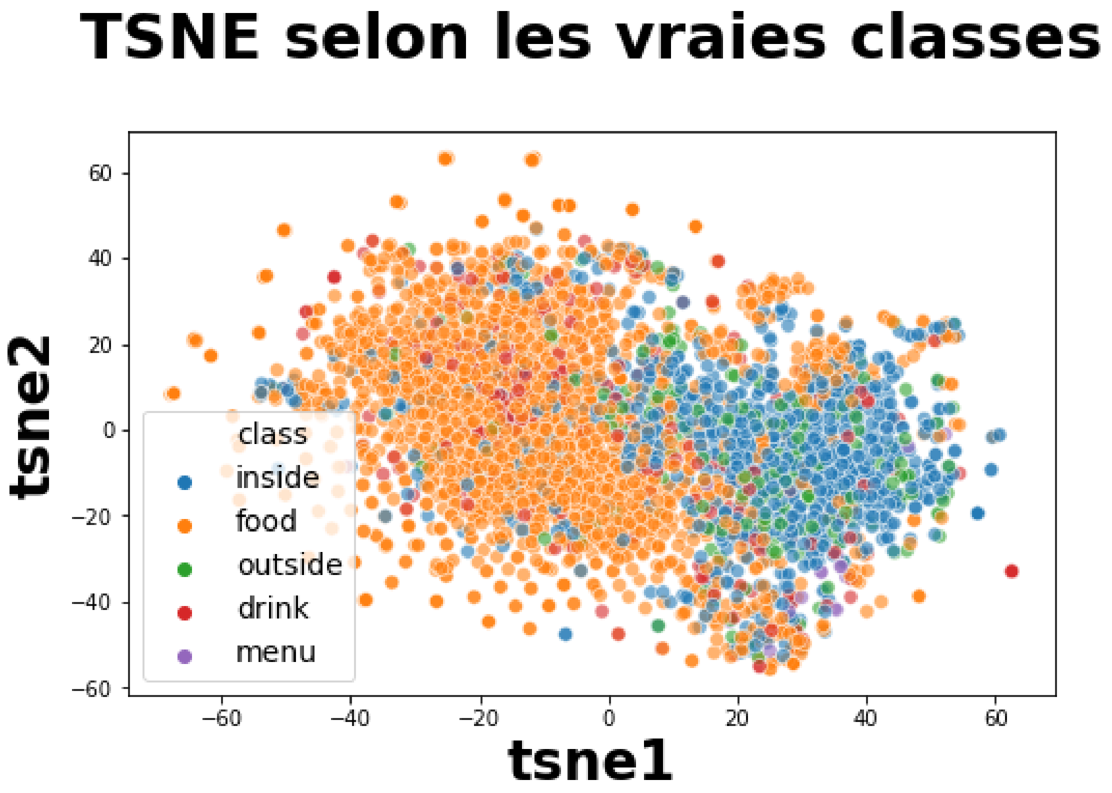
3.7 Analyse visuelle : affichage T-SNE selon catégories d'images

```
[36]: plt.figure(figsize=(8,5))
sns.scatterplot(
    x="tsne1", y="tsne2", hue="class", data=df_tsne, legend="brief",
    palette=sns.color_palette('tab10', n_colors=5), s=50, alpha=0.6)

plt.title('TSNE selon les vraies classes', fontsize = 30, pad = 35, fontweight␣
 ↪= 'bold')
plt.xlabel('tsne1', fontsize = 26, fontweight = 'bold')
plt.ylabel('tsne2', fontsize = 26, fontweight = 'bold')
plt.legend(prop={'size': 14})

plt.savefig('./img_classes_pca.png', bbox_inches='tight')

plt.show()
```



3.8 Analyse mesures : similarité entre catégories et clusters

3.8.1 Création de clusters à partir du PCA

[37]:

```
from sklearn import cluster, metrics

cls = cluster.KMeans(n_clusters=10, random_state=6)
cls.fit(feat_pca)

# Saving the feat_pca
pcafile = 'pca.sav'
pickle.dump(pca, open(pcafile, 'wb'))

df_tsne["cluster"] = cls.labels_
print(df_tsne.shape)
```

```
(3000, 4)
```

3.8.2 Affichage des images selon clusters et calcul ARI de similarité catégories images / clusters

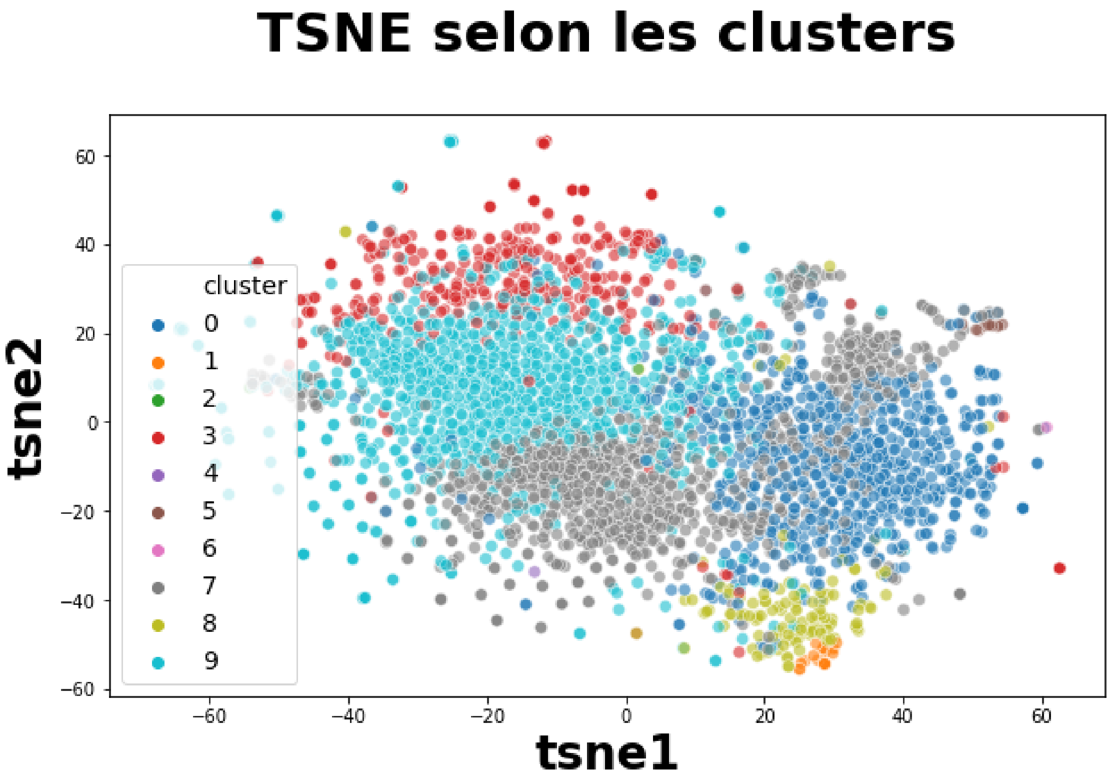
[38]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(
    x="tsne1", y="tsne2",
    hue="cluster",
    palette=sns.color_palette('tab10', n_colors=10), s=50, alpha=0.6,
    data=df_tsne,
    legend="brief")

plt.title('TSNE selon les clusters', fontsize = 30, pad = 35, fontweight =␣
 ↪'bold')
plt.xlabel('tsne1', fontsize = 26, fontweight = 'bold')
plt.ylabel('tsne2', fontsize = 26, fontweight = 'bold')
plt.legend(prop={'size': 14})

plt.savefig('./img_clusters_pca.png', bbox_inches='tight')
plt.show()

labels = df_tsne['class'] #!!!!!!!
print("ARI : ", metrics.adjusted_rand_score(labels, cls.labels_))
```



```
ARI :  0.15324705160191057
```

3.9 Training a classification model based on the histogram

```
[20]: refrom sklearn import metrics
      # classifier: this argument consists of the call to the classifier including␣
       ↪its arguments. Ex: linear_model.LogisticRegression(solver = 'liblinear')

      def apprentissage(classifier, classifier_name, X_train, y_train, X_test,␣
       ↪y_test, param, cross_val=5):


          #Initializing the grid search
          classifier_gs = model_selection.GridSearchCV(classifier, param_grid =␣
       ↪param, cv=cross_val, n_jobs=4)
      #     classifier_gs = model_selection.GridSearchCV(classifier, cv=cross_val,␣
       ↪n_jobs=4)

          #Training the model with a capture of elapsed time
          start_time = timeit.default_timer()
          classifier_gs.fit(X_train, y_train)
```

```
    elapsed = timeit.default_timer() - start_time

#    print('The best parameters as rendered by the gridSearch process are: \n',␣
 ↪classifier_gs.best_params_)
    print("The computation time for training the model is {:.2f}s".
 ↪format(elapsed))


    #Initializing the prediction over the test labels
    if classifier_name == 'SVM':
        y_pred = classifier_gs.predict(X_test)
    else:
        y_prob = classifier_gs.predict_proba(X_test)[:,1]
    #     print(np.shape(y_prob))
        y_pred = np.where(y_prob > 0.5, 1, 0)
    #     fpr, tpr, thr = metrics.roc_curve(y_test, y_prob)
    #     print(np.shape(y_pred))

#     roc_auc = metrics.auc(fpr, tpr)
    f1_sc = metrics.f1_score(y_test, y_pred, average='macro')
    rcall = metrics.recall_score(y_test, y_pred, average='macro')
#     cmatrx = metrics.confusion_matrix(y_test, y_pred)
    time = elapsed

#    print(roc_auc)
#    print(rcall)
#    print(f1_sc)
#    print(cmatrx)
#    print(time)

    res={}
    res['best_estimator']= classifier_gs.best_estimator_
    for string in param.keys():
        res[string]=param[string]
    for string in classifier_gs.best_params_.keys():
        res[string+'_opt']=classifier_gs.best_params_[string]

    #Building the outpout
    res.update({'cv':cross_val, 'time':time, 'recall':rcall, 'f1_score':f1_sc})
    return res



def affichage_resultats(apprentissage, df, classifier_name):
    line = pd.Series(apprentissage, name=classifier_name)
    df = df.append(line)
    return df
```

[21]:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
bin_class = le.fit_transform(df_tsne['class'])
```

[22]:

```
from sklearn import model_selection
X_train, X_test, y_train, y_test = model_selection.
 ↪train_test_split(feat_pca,bin_class, test_size=0.3)

# # liste labels de colonnes
# feat = list(X_train.columns)

std_scale = preprocessing.StandardScaler().fit(X_train)
X_train_std = std_scale.transform(X_train)
X_test_std = std_scale.transform(X_test)

#Saving the scaler in a file
scalerfile = 'standardScaler.sav'
pickle.dump(std_scale, open(scalerfile, 'wb'))
```

[32]:

```
from sklearn import ensemble
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC

num_trees = 200
seed = 9
scoring = 'f1_score'

models = []
models.append(('LR', LogisticRegression(random_state=seed), {'C': np.
 ↪logspace(-7, 7, 12) , 'penalty':['l1','l2']}))
models.append(('LDA', LinearDiscriminantAnalysis(), {}))
models.append(('KNN', KNeighborsClassifier(), {'n_neighbors':np.arange(5,10)}))
models.append(('CART', DecisionTreeClassifier(random_state=seed), {'max_depth':
 ↪np.arange(5,10)}))
models.append(('RF', RandomForestClassifier(n_estimators=num_trees,␣
 ↪random_state=seed), {'max_depth':np.arange(5,10)}))
models.append(('NB', GaussianNB(), {}))
models.append(('SVM', LinearSVC(random_state=seed), {'penalty':['l1','l2'],␣
 ↪'multi_class':['crammer_singer', 'ovr'], 'C': np.logspace(0.1, 7, 10)}))

names = []
for name, model, param in models:
    computation = apprentissage(model,name,
              X_train_std, y_train, X_test_std, y_test,
              param
             )
    names.append(name)
    msg_f1 = "%s: %f" % (name, computation['f1_score'])
    msg_rec = "%s: %f" % (name, computation['recall'])

    print(msg_f1)
    print(msg_rec)
```

```
The computation time for training the model is 1.92s
LR: 0.206622
LR: 0.269881
The computation time for training the model is 0.17s
LDA: 0.202987
LDA: 0.263333
The computation time for training the model is 0.99s
KNN: 0.179367
KNN: 0.220357
The computation time for training the model is 1.00s
CART: 0.192372
CART: 0.243214
The computation time for training the model is 11.30s
RF: 0.194826
RF: 0.240952
The computation time for training the model is 0.02s
NB: 0.046520
NB: 0.204762
The computation time for training the model is 2035.99s
SVM: 0.506648
SVM: 0.499040
```

```
/home/erbadi/anaconda3/lib/python3.7/site-packages/sklearn/svm/_base.py:947:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[29]:
```
computation
```

[29]:
```
{'best_estimator': LinearSVC(C=1.2589254117941673, class_weight=None, dual=True,
           fit_intercept=True, intercept_scaling=1, loss='squared_hinge',
           max_iter=1000, multi_class='ovr', penalty='l2', random_state=9,
           tol=0.0001, verbose=0),
 'penalty': ['l1', 'l2'],
 'multi_class': ['crammer_singer', 'ovr'],
 'C': array([1.25892541]),
 'C_opt': 1.2589254117941673,
 'multi_class_opt': 'ovr',
 'penalty_opt': 'l2',
 'cv': 5,
 'time': 103.99945970499539,
 'recall': 0.4990396825396825,
 'f1_score': 0.5066477529145651}
```

[30]:

```
classif_model = computation['best_estimator']
```

Sauvegarde du modèle de classification

[31]:

```
classifierfile = 'classifier.sav'
pickle.dump(classif_model, open(classifierfile, 'wb'))
```