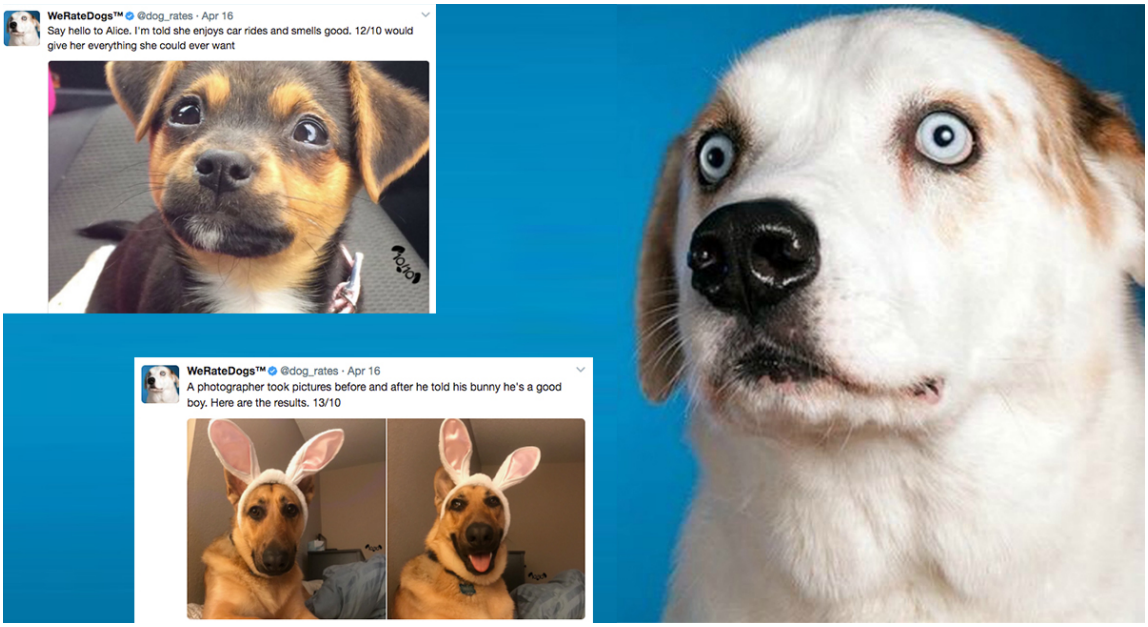


Project: Wrangling and Analyze Data- [WeRateDogs Tweet Data Wrangling]

Table of Contents

Introduction

Dataset Description

> **Présentation**: WeRateDogs est un compte Twitter qui évalue les chiens des gens avec un commentaire humoristique sur le chien. Le compte a été lancé en 2015 par l'étudiant Matt Nelson et a attiré l'attention des médias internationaux. Il partage des images de chiens et rédige un bref panégyrique sur le chien, puis laisse ses abonnés le noter en le mettant en favori. En demandant à WeRateDogs de partager avec nous certains de leurs tweets, ils l'ont fait. Ils ont partagé plus de 5000 de leurs tweets qui contiennent des données de base. Parfois, dans leur bref panégyrique, ils mentionnent la race du chien, et d'autres noms. Mais grâce à Udacity, ils ont effectué des procédures pour classer les chiens en fonction de leurs images partagées avec les tweets.

> **Motivation du projet**: Exploiter les données Twitter de WeRateDogs pour créer des analyses et des visualisations intéressantes et fiables. Les archives Twitter sont excellentes, mais elles ne contiennent que des informations très basiques sur les tweets. Une collecte supplémentaire, puis une évaluation et un nettoyage sont nécessaires pour des analyses et des visualisations « Wow ! »-dignes.
>
> **Données**: Dans le cadre de ce projet, nous travaillerons sur les trois ensembles de données suivants :
>
> - **Archives Twitter améliorées**: Les archives Twitter de WeRateDogs contiennent des données de base sur les 5 000 tweets et plus, mais pas sur tout. Une colonne de l'archive contient cependant : le texte de chaque tweet, qui peut être utilisé pour extraire la note, le nom du chien et les « étapes » du chien (cest-à-dire doggo, floofer, pupper et puppo) pour rendre cette archive Twitter « améliorée ».
> - **Données supplémentaires via l'API Twitter**: Revenons à l'aspect basique des archives Twitter : le nombre de retweets et le nombre de favoris sont deux des omissions notables de la colonne. Heureusement, ces données supplémentaires peuvent être collectées par n'importe qui à partir de l'API de Twitter. Enfin, « n'importe qui » ayant accès aux données des 3 000 tweets les plus récents, du moins.
> - **Fichier de prédictions d'images**: Autre chose intéressante : sont lancées toutes les images dans les archives Twitter de WeRateDogs à travers un réseau neural network capable de classer les races de chiens*. Les résultats : un tableau rempli de prédictions d'images (les trois premières seulement) avec l'ID de chaque tweet, l'URL de l'image et le numéro de l'image correspondant à la prédiction la plus sûre (numérotée de 1 à 4 puisque les tweets peuvent contenir jusqu'à quatre images).
>
> **Données externes** : Une **quatrième base de données** sera collectée sur wikipédia pour récupérer le nom complet des langues de tweet en utilisant le **web scraping** à partir d' ici.

Importation des librairies necessaires

In [584…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
import tweepy
from timeit import default_timer as timer
import json
import re
import missingno as msno
import seaborn as sns
from bs4 import BeautifulSoup
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
```

Data Gathering

Dans la cellule ci-dessous, nous rassemblons **les quatres** données pour ce projet et les chargeons dans le bloc-notes.
Remarque : les méthodes requises pour recueillir chaque donnée sont différentes.

---

1. Téléchargement directe des données d'archive Twitter de WeRateDogs (twitter_archive_enhanced.csv)

In [585…
```
archive_df = pd.read_csv("datasets/twitter-archive-enhanced.csv")
archive_df.head(5)
```

Out[585]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.cor
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.cor
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.cor
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.cor
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.cor

---

1. Nous utilisons ici la bibliothèque **Requests** pour télécharger la prédiction d'image de tweet (image_predictions.tsv)

In [586…
```
#URL = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-pre
#response = requests.get(URL)
#open("datasets/image-predictions.tsv", "wb").write(response.content)
```

In [587…
```
pred_df = pd.read_csv("datasets/image-predictions.tsv", sep='\t')
pred_df.head(5)
```

Out[587]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_spring
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature

---

1. Nous utilisez la bibliothèque Tweepy pour interroger des données supplémentaires via l'API Twitter (tweet_json.txt)

In [588…
```
bearer_token = '*****'
consumer_key = '*****'
consumer_secret = '*****'
access_token = '*****'
access_secret = '*****'
```

In [589…
```
#auth = tweepy.client(bearer_token=bearer_token,consumer_key =consumer_key, consume
#                     access_token = access_token, access_token_secret = access_sec
#
#api = tweepy.API(auth, parser=tweepy.parsers.JSONParser(), wait_on_rate_limit=True
```

In [590…
```
#for tweet_id in archive_df.tweet_id:
#   try:
#        tweet_json = api.get_status(tweet_id, tweet_mode = 'extented')
#        with open('datasets/tweet_json.txt', mode='a') as file:
#        json.dump(tweet_json, file)
#            file.write('\n')
#    except Exception as e:
#        pass
```

In [591…
```
# Liste vide pour stocker les dictionaries qui seront créés
lists = []
# Liste des colonnes sélectionnés pour les tweets
#mycolumns = ['id','full_text','retweet_count','is_quote_status','favorite_count',
with open('datasets/tweet_json.txt') as file:
    lines = file.read().splitlines()
    for line in lines:
        data = json.loads(line)
        lists.append(data)
tweet_df = pd.DataFrame(lists)#,columns=mycolumns)
```

In [592…
```
tweet_df.head(5)
```

Out[592]:

	created_at	id	id_str	full_text	truncated	display_text_rang
0	Tue Aug 01 16:23:56 +0000 2017	892420643555336193	892420643555336193	This is Phineas. He's a mystical boy. Only eve...	False	[0, 8
1	Tue Aug 01 00:17:27 +0000 2017	892177421306343426	892177421306343426	This is Tilly. She's just checking pup on you....	False	[0, 13
2	Mon Jul 31 00:18:03 +0000 2017	891815181378084864	891815181378084864	This is Archie. He is a rare Norwegian Pouncin...	False	[0, 12
3	Sun Jul 30 15:58:51 +0000 2017	891689557279858688	891689557279858688	This is Darla. She commenced a snooze mid meal...	False	[0, 7
4	Sat Jul 29 16:00:24 +0000 2017	891327558926688256	891327558926688256	This is Franklin. He would like you to stop ca...	False	[0, 13

5 rows × 31 columns

---

1. Nous utilisons beautifulsoup ici, un client http python pour collecter les données sur la page wikipédia (webscraping) des langues.

In [593…
```
# Obtenir la réponse sous forme de html
wikiurl="https://fr.wikipedia.org/wiki/Liste_des_codes_ISO_639-1"
table_class = 'wikitable sortable alternance jquery-tablesorter' # la class du tabl
response=requests.get(wikiurl) # reponse de la requete http
print(response.status_code) # Vérifier si le code marche (code 200)
```

```
200
```

In [594…
```
# Analyser les données du HTML dans un objet beautifulsoup
soup = BeautifulSoup(response.text, 'html.parser') # parser le html pour le manipul
langtable=soup.find('table',{'class':"wikitable"})
```

In [595…
```
# Stocker le tableau dans pandas dataframe
lang_df = pd.read_html(str(langtable))

# Convertir list en dataframe
lang_df=pd.DataFrame(lang_df[0])
lang_df.head(10)
```

Out[595]:

	639-1	639-2	639-3	Nom de la langue	Nom(dans la langue correspondante)	Nom en anglais	Commentaire
0	aa	aar	aar	Afar	Afaraf	Afar	NaN
1	ab	abk	abk	Abkhaze	Аҧсуа	Abkhazian	NaN
2	ae	ave	ave	Avestique	Avesta	Avestan	NaN
3	af	afr	afr	Afrikaans	Afrikaans	Afrikaans	NaN
4	ak	aka	aka + 2	Akan	Akan	Akan	NaN
5	am	amh	amh	Amharique	አማርኛ	Amharic	NaN
6	an	arg	arg	Aragonais	Aragonés	Aragonese	NaN
7	ar	ara	ara + 30	Arabe	العربية	Arabic	L'arabe standard est arb en ISO 639-3
8	as	asm	asm	Assamais	অসমীয়া	Assamese	NaN
9	av	ava	ava	Avar	авар мацI ; магIарул мацI	Avaric	NaN

Assessing Data

Dans cette section, détectont documentez au moins **huit (8) problèmes de qualité et deux (2) problèmes d'ordre**. Vous devez utiliser **les deux** évaluation visuelle évaluation programmatique pour évaluer les données.

Remarque : faites attention aux points clés suivants lorsque vous accédez aux données.

- Vous ne voulez que des notes originales (pas de retweets) qui ont des images. Bien qu'il y ait plus de 5000 tweets dans l'ensemble de données, tous ne sont pas des évaluations de chiens et certains sont des retweets.
- L'évaluation et le nettoyage complet de l'ensemble de données prendraient beaucoup de temps et il n'est pas nécessaire de pratiquer et de démontrer vos compétences en matière de traitement des données. Par conséquent, les exigences de ce projet consistent uniquement à évaluer et à éliminer au moins 8 problèmes de qualité et au moins 2 problèmes d'ordre dans cet ensemble de données.
- Le fait que les numérateurs de notation soient supérieurs aux dénominateurs n'a pas besoin d'être nettoyé. Ce [système de notation unique] (http://knowyourmeme.com/memes/theyre-good-dogs-brent) est une grande partie de la popularité de WeRateDogs.
- Vous n'avez pas besoin de collecter les tweets au-delà du 1er août 2017. Vous le pouvez, mais notez que vous ne pourrez pas collecter les prédictions d'images pour ces tweets puisque vous n'avez pas accès à l'algorithme utilisé.

Visual Assessment / Évaluation visuelle

Nous allons ici lancer arbitraire un échantillon de 100 observations de nos trois bases pour avoir un aperçu sur leurs compositions.

In [596…
```
archive_df.sample(100)
```

Out[596]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
726	782598640137187329	NaN	NaN	2016-10-02 15:10:30 +0000	href="http://twitte
2287	667177989038297088	NaN	NaN	2015-11-19 03:10:02 +0000	href="http://twitte
718	783466772167098368	NaN	NaN	2016-10-05 00:40:09 +0000	href="http://twitte
242	846153765933735936	NaN	NaN	2017-03-27 00:15:53 +0000	href="http://twitte
1818	676593408224403456	NaN	NaN	2015-12-15 02:43:33 +0000	
...	...	...	...	...	
232	847962785489326080	NaN	NaN	2017-04-01 00:04:17 +0000	href="http://twitte
2308	666817836334096384	NaN	NaN	2015-11-18 03:18:55 +0000	href="http://twitte
708	785264754247995392	NaN	NaN	2016-10-09 23:44:41 +0000	href="http://twitte
1776	677961670166224897	NaN	NaN	2015-12-18 21:20:32 +0000	
436	820314633777061888	NaN	NaN	2017-01-14 17:00:24 +0000	href="http://twitte

100 rows × 17 columns

In [597…
```
pred_df.sample(100)
```

Out[597]:

	tweet_id	jpg_url	img_num	
1187	739485634323156992	https://pbs.twimg.com/media/CkMuP7SWkAAD-2R.jpg	2	Walke
468	675109292475830276	https://pbs.twimg.com/media/CV54UQTXAAAGf-j.jpg	1	da
77	667437278097252352	https://pbs.twimg.com/media/CUM2qWaWoAUZ06L.jpg	1	pc
286	671151324042559489	https://pbs.twimg.com/media/CVBokRSWsAADuXx.jpg	1	Rc
1083	718460005985447936	https://pbs.twimg.com/media/Cfh7j6CWQAAndTd.jpg	1	
...	...	...	...	
249	670679630144274432	https://pbs.twimg.com/media/CU67jGSUkAAk_1Y.jpg	1	Ibizar
664	682697186228989953	https://pbs.twimg.com/media/CXltdtaWYAIuX_V.jpg	1	bal
50	666837028449972224	https://pbs.twimg.com/media/CUEUva1WsAA2jPb.jpg	1	tric
1562	793500921481273345	https://pbs.twimg.com/media/CwMU34YWIAAz1nU.jpg	2	golden_r
1053	714214115368108032	https://pbs.twimg.com/media/Cell8ikWIAACCJ-.jpg	1	

100 rows × 12 columns

In [598…

```
tweet_df.sample(10)
```

Out[598]:

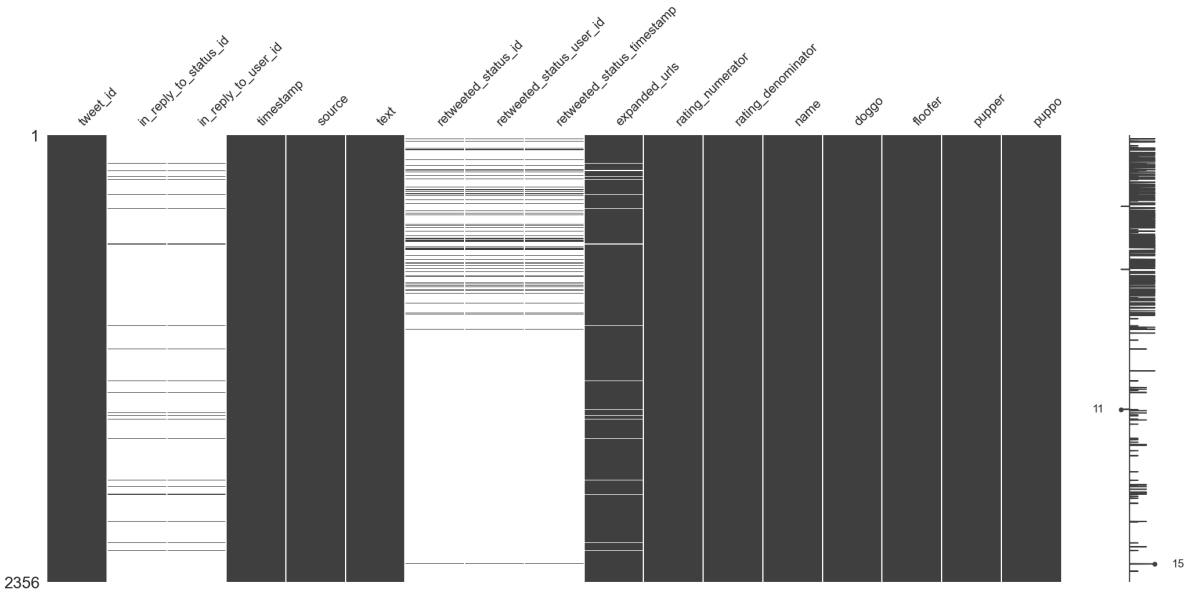
	created_at	id	id_str	full_text	truncated	display_tex
2099	Sat Nov 28 19:51:59 +0000 2015	670691627984359425	670691627984359425	This is Ester. He has a cocaine problem. This ...	False	
2260	Fri Nov 20 03:35:20 +0000 2015	667546741521195010	667546741521195010	Here is George. George took a selfie of his ne...	False	
88	Wed Jun 14 16:04:48 +0000 2017	875021211251597312	875021211251597312	Guys please stop sending pictures without any ...	False	
1581	Wed Jan 13 02:43:46 +0000 2016	687102708889812993	687102708889812993	Army of water dogs here. None of them know whe...	False	
746	Sun Sep 25 00:06:08 +0000 2016	779834332596887552	779834332596887552	This is Scout. He really wants to kiss himself...	False	
645	Mon Oct 31 18:00:14 +0000 2016	793150605191548928	793150605191548928	This is Nida. She's a free elf. Waited so long...	False	
1153	Tue Apr 26 15:29:30 +0000 2016	724983749226668032	724983749226668032	This is Fred-Rick. He dabbles in parkour. The ...	False	
792	Wed Sep 07 15:44:53 +0000 2016	773547596996571136	773547596996571136	This is Chelsea. She forgot how to dog. 11/10 ...	False	
1191	Wed Apr 06 02:21:30 +0000 2016	717537687239008257	717537687239008257	People please. This is a Deadly Mediterranean ...	False	
1285	Fri Mar 11 18:18:36 +0000 2016	708356463048204288	708356463048204288	This is Oliver. That is his castle. He protect...	False	

10 rows × 31 columns

Nous utilisons ici la librairie *missingno* pour observer les valeurs manquantes dans nos données.

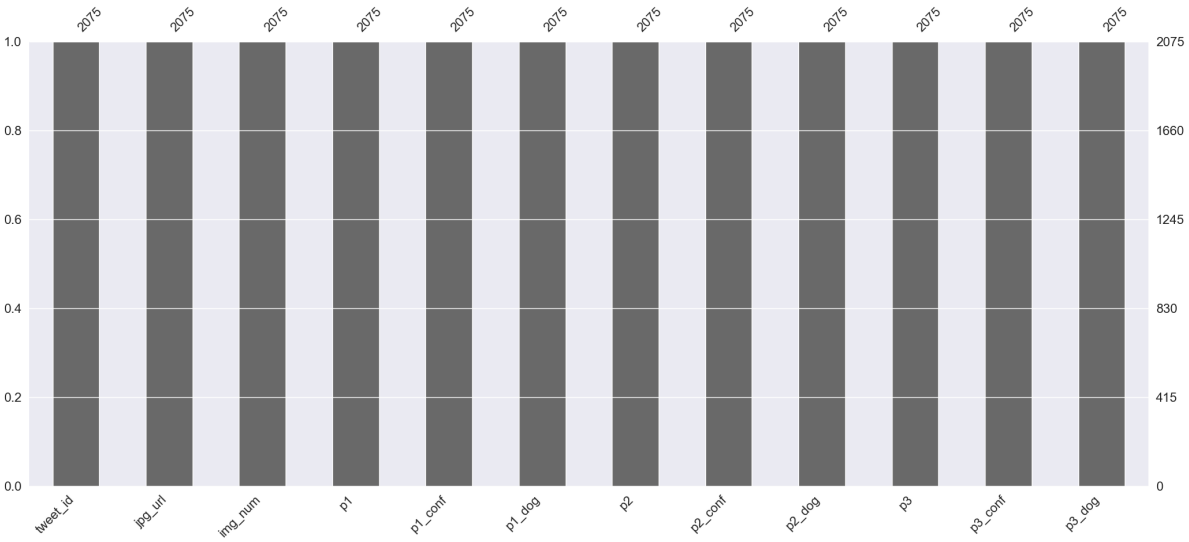
In [599…
```
# Visualiser les valeurs manquantes avec une matrice
msno.matrix(archive_df)
```

Out[599]:
```
<AxesSubplot: >
```



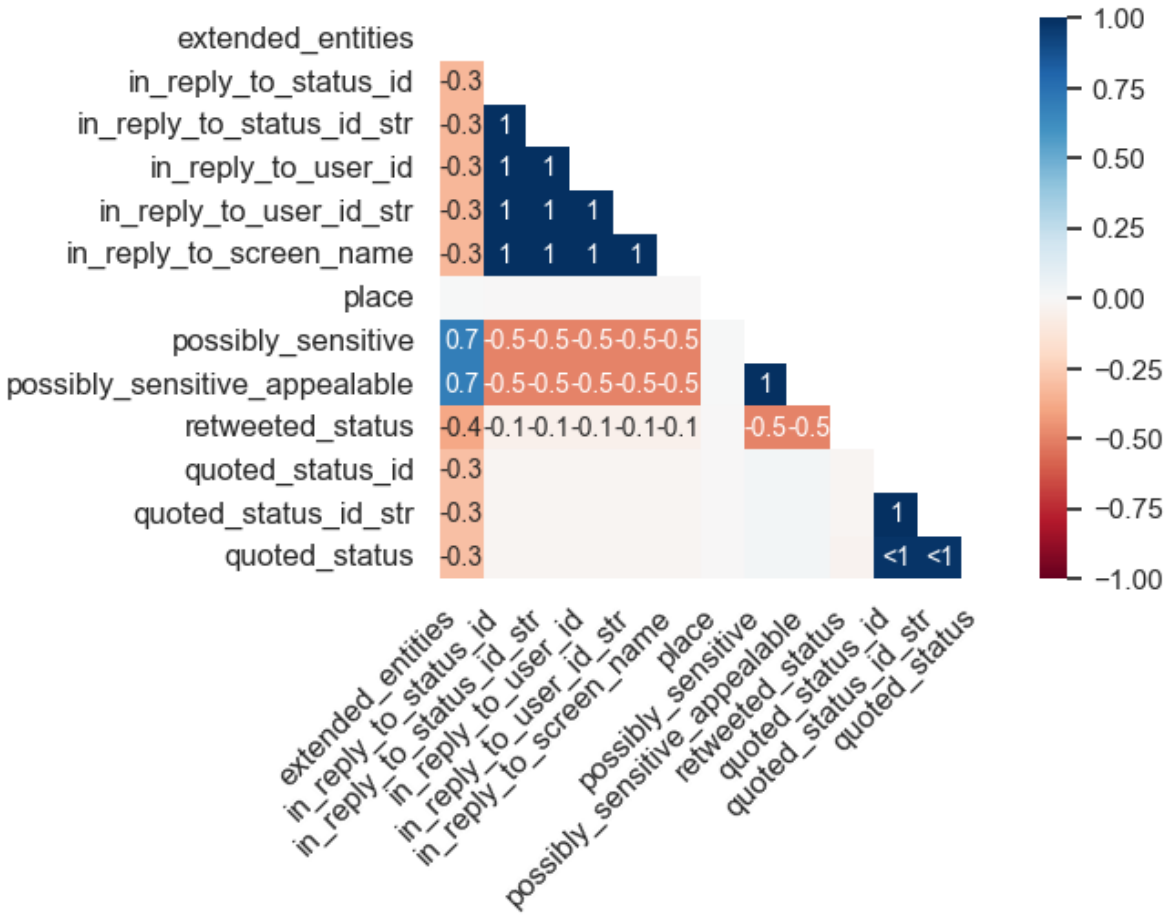
In [600…
```
# Visualisation des valeurs manquantes avec un graphique en barres
msno.bar(pred_df)
```

Out[600]:
```
<AxesSubplot: >
```



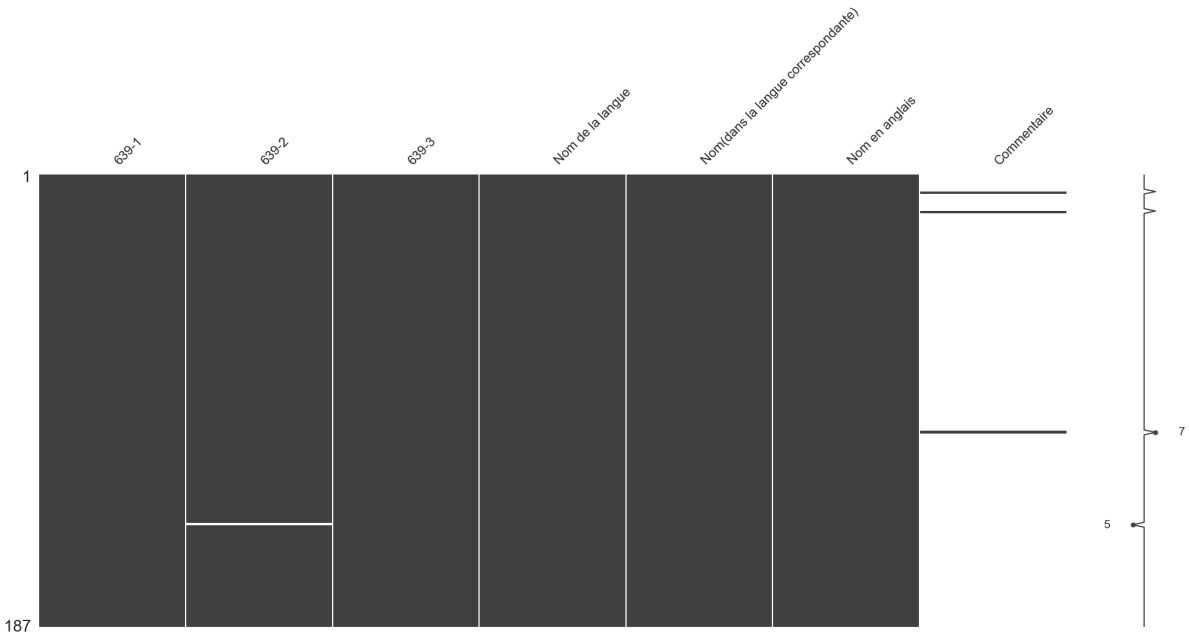
In [601…
```
# Visualisation de la corrélation entre le nombre de valeurs manquantes dans différ
msno.heatmap(tweet_df, figsize=(5,4), fontsize=12)
```

Out[601]:
```
<AxesSubplot: >
```



In [602…
```
msno.matrix(lang_df)
```

Out[602]:
```
<AxesSubplot: >
```



Programmatic Assessment / Évaluation programmatique

Dans cette partie nous procédons à l'évaluation par bouts de code pour dénicher les problèmes liés à nos données.

- `archive_df`

In [603…
```
archive_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

Problème 1 : Par le visuel et le code on constate que :

- Le type de la variable `timestamp` n'est pas adéquat pour déterminer le temps.*
- Nos trois variables `retweeted_status_*` et deux variables `in_reply_to_*` sont quasis vides donc n'apportent presque pas d'information à l'étude. Doivent normalement être écarté de l'étude.*
- `expanded_urls` contient aussi d'importantes valeurs manquantes mais pas autant que les dernières variables citées.*

Par ailleurs :

- Les variables `rating_numerator` et `rating_denominator` peuvent faire une seule variable au lieu de deux puisqu'il s'agit d'une note, même si on est en façe d'une notation assez particulière.*
- On ne remarque aussi qu'il n'y a pas de valeurs manquantes pour nos différentes stades de chien, ce qui n'est pas normal car un chien ne peut réunir toutes ces caractéristiques en même temps. Donc de fausses données sont insérées.*

In [604…
```
archive_df.rating_numerator.value_counts()
```

Out[604]:
```
12      558
11      464
10      461
13      351
9       158
8       102
7        55
14       54
5        37
6        32
3        19
4        17
2         9
1         9
75        2
15        2
420       2
0         2
80        1
144       1
17        1
26        1
20        1
121       1
143       1
44        1
60        1
45        1
50        1
99        1
204       1
1776      1
165       1
666       1
27        1
182       1
24        1
960       1
84        1
88        1
Name: rating_numerator, dtype: int64
```

In [605…
```
archive_df.rating_denominator.value_counts()
```

Out[605]:
```
10     2333
11        3
50        3
20        2
80        2
70        1
7         1
15        1
150       1
170       1
0         1
90        1
40        1
130       1
110       1
16        1
120       1
2         1
Name: rating_denominator, dtype: int64
```

Problème 2 :

- On est en face d'une notation pas du tout régulière, donc remise en cause.

In [606…
```
stades = ['doggo','floofer','pupper','puppo']
for stade in stades :
    print(archive_df[stade].value_counts())
    print('--------------------------------')
```

```
None     2259
doggo      97
Name: doggo, dtype: int64
--------------------------------
None       2346
floofer      10
Name: floofer, dtype: int64
--------------------------------
None      2099
pupper     257
Name: pupper, dtype: int64
--------------------------------
None     2326
puppo      30
Name: puppo, dtype: int64
--------------------------------
```

Problème 3 :

- On voit donc qu'en réalité les valeurs supposées vides était remplies par la valeur 'None' au lieu de `None` .

In [607…
```
list(archive_df.text)[0]
```

Out[607]:
```
"This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 1
3/10 https://t.co/MgUWQ76dJU"
```

Problème 4 :

- Dans la variable `text` on découvre une autre variable cachée qu'est le `lien` .

In [608…
```
list(archive_df.source.unique())
```

Out[608]:
```
['<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</
a>',
 '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
 '<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>',
 '<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck
</a>']
```

Problème 5 :

- La variable `source` affiche plus que la vraie source essentiel qu'est l'appareil ou le moyen d'acces de Twitter.

In [609…
```
list(archive_df.expanded_urls)[0]
```

Out[609]:
```
'https://twitter.com/dog_rates/status/892420643555336193/photo/1'
```

Rien d'interessant !

In [610…
```
pd.DataFrame([name for name in list(archive_df.name) if len(name)<=2]).value_counts
```

Out[610]:
```
a     55
Bo     9
an     7
Al     1
Ed     1
JD     1
Jo     1
Mo     1
O      1
by     1
my     1
dtype: int64
```

Problème 6 :

- On voit des noms de chiens incorectes comme *a* par exemple répété 55 fois et *an* 7 fois dans le jeu de donnée.

- `pred_df`

In [611…
```
pred_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   int64  
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

Problème 7 :

- Pas de valeurs manquantes heureusement, mais nous avons 3 prédictions pour une seule image, faudra t-il chosir uniquement donc celle qui a la meilleure prédiction puisque nous ne sommes pas dans une étude d'apprentissage automatique.

In [612…
```
pred_df.p1.value_counts()
```

Out[612]:
```
golden_retriever      150
Labrador_retriever    100
Pembroke               89
Chihuahua              83
pug                    57
                     ...
pillow                  1
carousel                1
bald_eagle              1
lorikeet                1
orange                  1
Name: p1, Length: 378, dtype: int64
```

In [613…
```
pred_df.p2.value_counts()
```

Out[613]:
```
Labrador_retriever    104
golden_retriever       92
Cardigan               73
Chihuahua              44
Pomeranian             42
                     ...
medicine_chest          1
quail                   1
horse_cart              1
waffle_iron             1
bagel                   1
Name: p2, Length: 405, dtype: int64
```

Problème 8 :

- Le nom des races des chiens doit être redéfini.

- `tweet_df`

In [614…
```
tweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 31 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   created_at                     2354 non-null   object
 1   id                             2354 non-null   int64
 2   id_str                         2354 non-null   object
 3   full_text                      2354 non-null   object
 4   truncated                      2354 non-null   bool
 5   display_text_range             2354 non-null   object
 6   entities                       2354 non-null   object
 7   extended_entities              2073 non-null   object
 8   source                         2354 non-null   object
 9   in_reply_to_status_id          78 non-null     float64
 10  in_reply_to_status_id_str      78 non-null     object
 11  in_reply_to_user_id            78 non-null     float64
 12  in_reply_to_user_id_str        78 non-null     object
 13  in_reply_to_screen_name        78 non-null     object
 14  user                           2354 non-null   object
 15  geo                            0 non-null      object
 16  coordinates                    0 non-null      object
 17  place                          1 non-null      object
 18  contributors                   0 non-null      object
 19  is_quote_status                2354 non-null   bool
 20  retweet_count                  2354 non-null   int64
 21  favorite_count                 2354 non-null   int64
 22  favorited                      2354 non-null   bool
 23  retweeted                      2354 non-null   bool
 24  possibly_sensitive             2211 non-null   object
 25  possibly_sensitive_appealable  2211 non-null   object
 26  lang                           2354 non-null   object
 27  retweeted_status               179 non-null    object
 28  quoted_status_id               29 non-null     float64
 29  quoted_status_id_str           29 non-null     object
 30  quoted_status                  28 non-null     object
dtypes: bool(4), float64(3), int64(3), object(21)
memory usage: 505.9+ KB
```

In [615…
```
list(tweet_df.entities)[0]
```

Out[615]:
```
{'hashtags': [],
 'symbols': [],
 'user_mentions': [],
 'urls': [],
 'media': [{'id': 892420639486877696,
   'id_str': '892420639486877696',
   'indices': [86, 109],
   'media_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
   'media_url_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
   'url': 'https://t.co/MgUWQ76dJU',
   'display_url': 'pic.twitter.com/MgUWQ76dJU',
   'expanded_url': 'https://twitter.com/dog_rates/status/892420643555336193/photo/
1',
   'type': 'photo',
   'sizes': {'large': {'w': 540, 'h': 528, 'resize': 'fit'},
    'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
    'small': {'w': 540, 'h': 528, 'resize': 'fit'},
    'medium': {'w': 540, 'h': 528, 'resize': 'fit'}}}]}
```

In [616…
```
list(tweet_df.extended_entities)[0]
```

Out[616]:

```
{'media': [{'id': 892420639486877696,
   'id_str': '892420639486877696',
   'indices': [86, 109],
   'media_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
   'media_url_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
   'url': 'https://t.co/MgUWQ76dJU',
   'display_url': 'pic.twitter.com/MgUWQ76dJU',
   'expanded_url': 'https://twitter.com/dog_rates/status/892420643555336193/photo/
1',
   'type': 'photo',
   'sizes': {'large': {'w': 540, 'h': 528, 'resize': 'fit'},
    'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
    'small': {'w': 540, 'h': 528, 'resize': 'fit'},
    'medium': {'w': 540, 'h': 528, 'resize': 'fit'}}}]}
```


In [617…
```
list(tweet_df.user.sample(4))
```

Out[617]:


```
[{'id': 4196983835,
  'id_str': '4196983835',
  'name': 'WeRateDogs™ (author)',
  'screen_name': 'dog_rates',
  'location': 'DM YOUR DOGS, WE WILL RATE',
  'description': '#1 Source for Professional Dog Ratings | STORE: @ShopWeRateDogs
| IG, FB & SC: WeRateDogs MOBILE APP: @GoodDogsGame | Business: dogratingtwitter@g
mail.com',
  'url': 'https://t.co/N7sNNHAEXS',
  'entities': {'url': {'urls': [{'url': 'https://t.co/N7sNNHAEXS',
      'expanded_url': 'http://weratedogs.com',
      'display_url': 'weratedogs.com',
      'indices': [0, 23]}]},
   'description': {'urls': []}},
  'protected': False,
  'followers_count': 3201006,
  'friends_count': 104,
  'listed_count': 2812,
  'created_at': 'Sun Nov 15 21:41:29 +0000 2015',
  'favourites_count': 114031,
  'utc_offset': None,
  'time_zone': None,
  'geo_enabled': True,
  'verified': True,
  'statuses_count': 5288,
  'lang': 'en',
  'contributors_enabled': False,
  'is_translator': False,
  'is_translation_enabled': False,
  'profile_background_color': '000000',
  'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.pn
g',
  'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme
1/bg.png',
  'profile_background_tile': False,
  'profile_image_url': 'http://pbs.twimg.com/profile_images/861415328504569856/R2x
OOfwe_normal.jpg',
  'profile_image_url_https': 'https://pbs.twimg.com/profile_images/861415328504569
856/R2xOOfwe_normal.jpg',
  'profile_banner_url': 'https://pbs.twimg.com/profile_banners/4196983835/15011290
17',
  'profile_link_color': 'F5ABB5',
  'profile_sidebar_border_color': '000000',
  'profile_sidebar_fill_color': '000000',
  'profile_text_color': '000000',
  'profile_use_background_image': False,
  'has_extended_profile': True,
  'default_profile': False,
  'default_profile_image': False,
  'following': True,
  'follow_request_sent': False,
  'notifications': False,
  'translator_type': 'none'},
 {'id': 4196983835,
  'id_str': '4196983835',
  'name': 'WeRateDogs™ (author)',
  'screen_name': 'dog_rates',
  'location': 'DM YOUR DOGS, WE WILL RATE',
  'description': '#1 Source for Professional Dog Ratings | STORE: @ShopWeRateDogs
| IG, FB & SC: WeRateDogs MOBILE APP: @GoodDogsGame | Business: dogratingtwitter@g
mail.com',
  'url': 'https://t.co/N7sNNHAEXS',
  'entities': {'url': {'urls': [{'url': 'https://t.co/N7sNNHAEXS',
      'expanded_url': 'http://weratedogs.com',
```

```
       'display_url': 'weratedogs.com',
       'indices': [0, 23]}]},
    'description': {'urls': []}},
   'protected': False,
   'followers_count': 3200952,
   'friends_count': 104,
   'listed_count': 2805,
   'created_at': 'Sun Nov 15 21:41:29 +0000 2015',
   'favourites_count': 114031,
   'utc_offset': None,
   'time_zone': None,
   'geo_enabled': True,
   'verified': True,
   'statuses_count': 5288,
   'lang': 'en',
   'contributors_enabled': False,
   'is_translator': False,
   'is_translation_enabled': False,
   'profile_background_color': '000000',
   'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.pn
g',
   'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme
1/bg.png',
   'profile_background_tile': False,
   'profile_image_url': 'http://pbs.twimg.com/profile_images/861415328504569856/R2x
OOfwe_normal.jpg',
   'profile_image_url_https': 'https://pbs.twimg.com/profile_images/861415328504569
856/R2xOOfwe_normal.jpg',
   'profile_banner_url': 'https://pbs.twimg.com/profile_banners/4196983835/15011290
17',
   'profile_link_color': 'F5ABB5',
   'profile_sidebar_border_color': '000000',
   'profile_sidebar_fill_color': '000000',
   'profile_text_color': '000000',
   'profile_use_background_image': False,
   'has_extended_profile': True,
   'default_profile': False,
   'default_profile_image': False,
   'following': True,
   'follow_request_sent': False,
   'notifications': False,
   'translator_type': 'none'},
  {'id': 4196983835,
   'id_str': '4196983835',
   'name': 'WeRateDogs™ (author)',
   'screen_name': 'dog_rates',
   'location': 'DM YOUR DOGS, WE WILL RATE',
   'description': '#1 Source for Professional Dog Ratings | STORE: @ShopWeRateDogs
| IG, FB & SC: WeRateDogs MOBILE APP: @GoodDogsGame | Business: dogratingtwitter@g
mail.com',
   'url': 'https://t.co/N7sNNHAEXS',
   'entities': {'url': {'urls': [{'url': 'https://t.co/N7sNNHAEXS',
       'expanded_url': 'http://weratedogs.com',
       'display_url': 'weratedogs.com',
       'indices': [0, 23]}]},
    'description': {'urls': []}},
   'protected': False,
   'followers_count': 3200892,
   'friends_count': 104,
   'listed_count': 2786,
   'created_at': 'Sun Nov 15 21:41:29 +0000 2015',
   'favourites_count': 114031,
   'utc_offset': None,
   'time_zone': None,
```

```
  'geo_enabled': True,
  'verified': True,
  'statuses_count': 5288,
  'lang': 'en',
  'contributors_enabled': False,
  'is_translator': False,
  'is_translation_enabled': False,
  'profile_background_color': '000000',
  'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.pn
g',
  'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme
1/bg.png',
  'profile_background_tile': False,
  'profile_image_url': 'http://pbs.twimg.com/profile_images/861415328504569856/R2x
OOfwe_normal.jpg',
  'profile_image_url_https': 'https://pbs.twimg.com/profile_images/861415328504569
856/R2xOOfwe_normal.jpg',
  'profile_banner_url': 'https://pbs.twimg.com/profile_banners/4196983835/15011290
17',
  'profile_link_color': 'F5ABB5',
  'profile_sidebar_border_color': '000000',
  'profile_sidebar_fill_color': '000000',
  'profile_text_color': '000000',
  'profile_use_background_image': False,
  'has_extended_profile': True,
  'default_profile': False,
  'default_profile_image': False,
  'following': True,
  'follow_request_sent': False,
  'notifications': False,
  'translator_type': 'none'},
 {'id': 4196983835,
  'id_str': '4196983835',
  'name': 'WeRateDogs™ (author)',
  'screen_name': 'dog_rates',
  'location': 'DM YOUR DOGS, WE WILL RATE',
  'description': '#1 Source for Professional Dog Ratings | STORE: @ShopWeRateDogs
| IG, FB & SC: WeRateDogs MOBILE APP: @GoodDogsGame | Business: dogratingtwitter@g
mail.com',
  'url': 'https://t.co/N7sNNHAEXS',
  'entities': {'url': {'urls': [{'url': 'https://t.co/N7sNNHAEXS',
      'expanded_url': 'http://weratedogs.com',
      'display_url': 'weratedogs.com',
      'indices': [0, 23]}]},
   'description': {'urls': []}},
  'protected': False,
  'followers_count': 3200950,
  'friends_count': 104,
  'listed_count': 2805,
  'created_at': 'Sun Nov 15 21:41:29 +0000 2015',
  'favourites_count': 114031,
  'utc_offset': None,
  'time_zone': None,
  'geo_enabled': True,
  'verified': True,
  'statuses_count': 5288,
  'lang': 'en',
  'contributors_enabled': False,
  'is_translator': False,
  'is_translation_enabled': False,
  'profile_background_color': '000000',
  'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.pn
g',
  'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme
```

```
1/bg.png',
  'profile_background_tile': False,
  'profile_image_url': 'http://pbs.twimg.com/profile_images/861415328504569856/R2x
OOfwe_normal.jpg',
  'profile_image_url_https': 'https://pbs.twimg.com/profile_images/861415328504569
856/R2xOOfwe_normal.jpg',
  'profile_banner_url': 'https://pbs.twimg.com/profile_banners/4196983835/15011290
17',
  'profile_link_color': 'F5ABB5',
  'profile_sidebar_border_color': '000000',
  'profile_sidebar_fill_color': '000000',
  'profile_text_color': '000000',
  'profile_use_background_image': False,
  'has_extended_profile': True,
  'default_profile': False,
  'default_profile_image': False,
  'following': True,
  'follow_request_sent': False,
  'notifications': False,
  'translator_type': 'none'}]
```

In [618…
```
tweet_df[tweet_df.possibly_sensitive == tweet_df.possibly_sensitive_appealable].sha
```

Out[618]:
```
(2211, 31)
```

Problème 9 :

- Ce dataset a énormément de valeurs manquantes, contient très probablement des variables inutiles à notre étude etcertaines variables ont des valeurs uniques.

In [619…
```
tweet_df.iloc[:,13:].sample(5)
```

Out[619]:

	in_reply_to_screen_name	user	geo	coordinates	place	contributors	is_quote_sta
2141	None	{'id': 4196983835, 'id_str': '4196983835', 'na...	None	None	None	None	F
1000	None	{'id': 4196983835, 'id_str': '4196983835', 'na...	None	None	None	None	F
153	None	{'id': 4196983835, 'id_str': '4196983835', 'na...	None	None	None	None	F
1945	None	{'id': 4196983835, 'id_str': '4196983835', 'na...	None	None	None	None	F
1931	None	{'id': 4196983835, 'id_str': '4196983835', 'na...	None	None	None	None	F

In [620…

```
tweet_df[tweet_df['is_quote_status'] == True]
```

Out[620]:

	created_at	id	id_str	full_text	truncat
31	Sat Jul 15 02:45:48 +0000 2017	886054160059072513	886054160059072513	RT @Athletics: 12/10 #BATP https://t.co/WxwJmv...	Fa
34	Thu Jul 13 15:19:09 +0000 2017	885518971528720385	885518971528720385	I have a new hero and his name is Howard. 14/1...	Fa
41	Mon Jul 10 03:08:17 +0000 2017	884247878851493888	884247878851493888	OMG HE DIDN'T MEAN TO HE WAS JUST TRYING A LIT...	Fa
71	Sat Jun 24 13:24:20 +0000 2017	878604707211726852	878604707211726852	Martha is stunning how h*ckin dare you. 13/10 ...	Fa
82	Sun Jun 18 20:30:39 +0000 2017	876537666061221889	876537666061221889	I can say with the pupmost confidence that the...	Fa
87	Wed Jun 14 21:06:43 +0000 2017	875097192612077568	875097192612077568	You'll get your package when that precious man...	Fa
109	Sat Jun 03 20:33:19 +0000 2017	871102520638267392	871102520638267392	Never doubt a doggo 14/10 https://t.co/AbBLh2FZCH	Fa
132	Mon May 22 18:21:28 +0000 2017	866720684873056260	866720684873056260	He was providing for his family 13/10 how dare...	Fa
189	Sat Apr 22 18:55:51 +0000 2017	855857698524602368	855857698524602368	HE'S LIKE "WAIT A MINUTE I'M AN ANIMAL THIS IS...	Fa
191	Sat Apr 22 16:18:34 +0000 2017	855818117272018944	855818117272018944	I HEARD HE TIED HIS OWN BOWTIE MARK AND HE JUS...	Fa
240	Mon Mar 27 23:35:28 +0000 2017	846505985330044928	846505985330044928	THIS WAS NOT HIS FAULT HE HAD NO IDEA. 11/10 S...	Fa
242	Sun Mar 26 23:20:02 +0000 2017	846139713627017216	846139713627017216	SHE DID AN ICY ZOOM AND KNEW WHEN TO PUT ON TH...	Fa

	created_at	id	id_str	full_text	truncat
255	Tue Mar 21 00:22:10 +0000 2017	843981021012017153	843981021012017153	HE WAS DOING A SNOOZE NO SHAME IN A SNOOZE 13/...	Fa
268	Mon Mar 13 16:08:50 +0000 2017	841320156043304961	841320156043304961	We don't rate penguins, but if we did, this on...	Fa
282	Tue Mar 07 03:22:35 +0000 2017	838952994649550848	838952994649550848	SHE MISPLACED HER HOOMAN 13/10 MISTAKES HAPPEN...	Fa
322	Tue Feb 21 17:18:39 +0000 2017	834089966724603904	834089966724603904	DOGGO ON THE LOOSE I REPEAT DOGGO ON THE LOOSE...	Fa
333	Fri Feb 17 20:05:43 +0000 2017	832682457690300417	832682457690300417	Prosperous good boy 13/10 socioeconomic af htt...	Fa
403	Tue Jan 24 22:44:42 +0000 2017	824025158776213504	824025158776213504	"I wish we were dogs" 14/10 for @BadlandsNPS h...	Fa
457	Sun Jan 08 01:40:55 +0000 2017	817908911860748288	817908911860748288	Looks like he went cross-eyed trying way too h...	Fa
538	Wed Dec 07 19:09:37 +0000 2016	806576416489959424	806576416489959424	Hooman catch successful. Massive hit by dog. F...	Fa
550	Fri Dec 02 00:02:45 +0000 2016	804475857670639616	804475857670639616	HE'S TRYING TO BE HIS OWN PERSON LET HIM GO 13...	Fa
576	Tue Nov 22 00:17:10 +0000 2016	800855607700029440	800855607700029440	RT @Lin_Manuel: 11/10 would recommend. https:/...	Fa
619	Tue Nov 08 23:01:49 +0000 2016	796125600683540480	796125600683540480	#ImWithThor 13/10\nhttps://t.co/a18mzkhTf6	Fa
675	Fri Oct 21 03:56:25 +0000 2016	789314372632018944	789314372632018944	HE WAS JUST A LIL SLEEPY FROM BEING SUCH A GOO...	Fa

	created_at	id	id_str	full_text	truncat
806	Sat Sep 03 03:13:29 +0000 2016	771908950375665664	771908950375665664	Doggo will persevere. 13/10\nhttps://t.co/yOVz...	Fa
891	Sat Jul 30 17:51:13 +0000 2016	759446261539934208	759446261539934208	No no no this is all wrong. The Walmart had to...	Fa
1038	Sat Jun 18 17:41:06 +0000 2016	744223424764059648	744223424764059648	This is actually a pupper and I'd pet it so we...	Fa
1152	Wed Apr 27 22:57:10 +0000 2016	725458796924002305	725458796924002305	Pup had to be removed cuz it wouldn't have bee...	Fa
1198	Sun Apr 03 20:53:33 +0000 2016	716730379797970944	716730379797970944	There has clearly been a mistake. Pup did noth...	Fa
1235	Tue Mar 22 16:06:19 +0000 2016	712309440758808576	712309440758808576	Reminder that we made our first set of sticker...	Fa
1322	Sat Mar 05 17:26:40 +0000 2016	706169069255446529	706169069255446529	He was doing his best. 12/10 I'll be his lawye...	Fa

31 rows × 31 columns

In [621…
```
tweet_df.is_quote_status.value_counts()
```

Out[621]:
```
False    2323
True       31
Name: is_quote_status, dtype: int64
```

In [622…
```
tweet_df.lang.value_counts()
```

Out[622]:
```
en     2336
und       7
in        3
nl        3
eu        1
es        1
tl        1
ro        1
et        1
Name: lang, dtype: int64
```

Problème 10 :

- Le nom des langues dans la variable `lang` a été abrégée, elle doit être explicite !

- `lang_df`

In [623…
```
lang_df.sample(5)
```

Out[623]:

	639-1	639-2	639-3	Nom de la langue	Nom(dans la langue correspondante)	Nom en anglais	Commentaire
68	ii	iii	iii	Yi	ꆈꌠ	Sichuan Yi	NaN
42	fa	per/fas	fas + 2	Persan	فارسی	Persian	NaN
162	ti	tir	tir	Tigrigna	ትግርኛ	Tigrinya	NaN
56	he	heb	heb	Hébreu	עברית	Hebrew	NaN
105	mn	mon	mon + 2	Mongol	Монгол	Mongolian	NaN

In [624…
```
lang_df[lang_df.eq('en').any(1)]
```

Out[624]:

	639-1	639-2	639-3	Nom de la langue	Nom(dans la langue correspondante)	Nom en anglais	Commentaire
37	en	eng	eng	Anglais	English	English	NaN

Pour récupérer le nom de la langue nous aurons besoin des variables `Nom de la langue` pour le nom de langue et `639-1` pour la jointure.

> Après cette longue évaluation accrue sur nos données, nous conviendrons que nous avons un peu perdu le fil des différents problèmes survenus. Une bonne pratique est de, comme appris en salle de cours *Udacity*, lister nos différentes problèmes liés à l'évaluation des données. Pour rappel quand on évalue les données, c'est pour réveler les problèmes liés soient à la qualité (contenu, valeurs manquantes, incorrectes, incohérentes et dupliquées) de nos données elles-même, soient avec leurs structures ou ordres (où *chaque variable forme une colonne*, *chaque observation forme une ligne* et *chaque type d'unité d'observation forme un tableau*).

En résumé nous avons :

Quality issues

Données `archive_df`

1. Le type de la variable `timestamp` n'est pas adéquat pour déterminer le temps.
2. Nos trois variables `retweeted_status_*` et deux variables `in_reply_to_*` sont quasis vides et donc doivent être écartés de l'étude..

3. `expanded_urls` contient aussi d'importantes valeurs manquantes mais pas autant que les dernières variables citées.

4. Notation pas du tout régulière, donc remise en cause.

5. Dans la variable `text` on découvre une autre variable cachée qu'est le `lien`.

6. La variable `source` affiche plus que la vraie source essentiel qu'est l'appareil ou le moyen d'acces de Twitter.

7. On voit des noms de chiens incorrects comme *a* par exemple répété 55 fois et *an* 7 fois dans le jeu de donnée.

8. Les valeurs supposées vides sont remplies par la valeur 'None' au lieu de `None`.

Données `pred_df`

1. Pas de valeurs manquantes heureusement, mais nous avons 3 prédictions pour une seule image, faudra t-il chosir uniquement donc celle qui a la meilleure prédiction puisque nous ne sommes pas dans une étude d'apprentissage automatique.

2. Le noms des races des chiens doit être redéfini.

Données `tweet_df`

1. Ce dataset a énormément de valeurs manquantes, contient très probablement des variables inutiles à notre étude et certaines variables ont des valeurs uniques.

2. Le nom des langues dans la variable `lang` a été abrégée, elle doit être explicite !

Tidiness issues

1. Les variables `rating_numerator` et `rating_denominator` peuvent faire une seule variable au lieu de deux puisqu'il s'agit d'une note, même si on est en façe d'une notation assez particulière.

2. Dénomination mal géré pour les étapes des chiens.

3. Les 3 premiers datasets ne doivent former qu'une seule unité d'observation qui axée sur l'analyse des tweets sur les chiens. Cette étape sera effectué un plus tard que les deux premières. On sous entend qu'on aura déjà récupéré le nom complet de la langue des tweets auprès de la 4ième base externes (`lang_df`).

Cleaning Data

In this section, clean **all** of the issues you documented while assessing.

Note: Make a copy of the original data before cleaning. Cleaning includes merging individual pieces of data according to the rules of tidy data. The result should be a high-quality and tidy master pandas DataFrame (or DataFrames, if appropriate).

In [625…
```
archive_df_copy = archive_df.copy()
pred_df_copy = pred_df.copy()
tweet_df_copy = tweet_df.copy()
lang_df_copy = lang_df.copy()
```

> Souvent il est plus facile de nettoyer des données déjà ordonnés que des données désordonnées, nous réglerons d'abord les problèmes liés à l'ordre afin d'aborder ceux liés à la qualité de ces mêmes données. **Néanmoins les problèmes 1 et 3 de qualité sur les donénes seront abordés un peu plus tard pour des raisons de logique de traitement de données et de commodités**. On sait qu'en traitement de données il n'y a jamais un raisonnement mécanique ou unanime pour tous les cas de data wrangling.

Problèmes d'ordre

Issue #2: Dénomination mal gérée pour les étapes des chiens.

Define:

- Remplacer les valeurs `None` par une chaîne vide.
- Regrouper les colonnes concernant les étapes des chiens pour créer une seule colonne `stade_chien`.
- Séparer les noms de stades de chien par une , dans la colonne `stade_chien`.

Code

In [626…
```
archive_df_copy.columns
```

Out[626]:
```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
      dtype='object')
```

In [627…
```
# Remplacer les valeurs None par une chaîne vide.
for column in ['doggo', 'floofer', 'pupper', 'puppo']:
    archive_df_copy[column].replace('None', '', inplace=True)
```

In [628…
```
# Jeter un coup d'œil à la disposition des stades de chien
archive_df_copy.groupby(["doggo", "floofer", "pupper", "puppo"]).size().reset_inde
```

Out[628]:

	doggo	floofer	pupper	puppo	0
0					1976
1				puppo	29
2			pupper		245
3		floofer			9
4	doggo				83
5	doggo			puppo	1
6	doggo		pupper		12
7	doggo	floofer			1

Sur la première ligne on voit qu'il y a 1976 chaines vides, mais elles seront traités plus tard au même titre que les valeurs 'None'.

In [629…

```
# Ajout de toutes les étapes et création de la colonne `stade_chien`
archive_df_copy['stade_chien'] = archive_df_copy.doggo +  archive_df_copy.floofer +

# Séparer les étapes du chien par une ','
archive_df_copy.loc[archive_df_copy.stade_chien == 'doggopupper', 'stade_chien'] =
archive_df_copy.loc[archive_df_copy.stade_chien == 'doggopuppo', 'stade_chien'] =
archive_df_copy.loc[archive_df_copy.stade_chien == 'doggofloofer', 'stade_chien'] =

# Supprimer les anciennes colonnes d'étape de chien
archive_df_copy.drop(columns=['doggo', 'floofer', 'pupper', 'puppo'], inplace=True)
```

Test

In [630…

```
archive_df_copy.stade_chien.value_counts()
```

Out[630]:

```
                  1976
pupper             245
doggo               83
puppo               29
doggo, pupper       12
floofer              9
doggo, puppo         1
doggo, floofer       1
Name: stade_chien, dtype: int64
```

In [631…

```
archive_df_copy.columns
```

Out[631]:

```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'stade_chien'],
      dtype='object')
```

Problèmes de qualité

- Données `archive_df`

Issue #1: Le type de la variable `timestamp` n'est pas adéquat pour déterminer le temps.

Define:

- Le type de la variable `timestamp` doit être une pandas *datetime*.
- La variable `timestamp` doit être renommée en `date`, ce qui semble plus logique.

Code

In [632…
```
archive_df_copy['timestamp'] = pd.to_datetime(archive_df_copy['timestamp'],format='
archive_df_copy = archive_df_copy.rename({'timestamp' : 'date'},axis='columns')
```

Test

In [633…
```
archive_df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    2356 non-null   int64
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   date                        2356 non-null   datetime64[ns, UTC]
 4   source                      2356 non-null   object
 5   text                        2356 non-null   object
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object
 9   expanded_urls               2297 non-null   object
 10  rating_numerator            2356 non-null   int64
 11  rating_denominator          2356 non-null   int64
 12  name                        2356 non-null   object
 13  stade_chien                 2356 non-null   object
dtypes: datetime64[ns, UTC](1), float64(4), int64(3), object(6)
memory usage: 257.8+ KB
```

Issue #2: Nos trois variables `retweeted_status_*` et deux variables `in_reply_to_*` sont quasis vides et donc doivent être écartés de l'étude.

Define

- Afficher le pourcentage de valeurs null dans le dataFrame est après supprimer ces 5 variables.

Code

In [634…
```
bad_df = archive_df_copy[archive_df_copy.columns[pd.Series(archive_df_copy.columns
```

In [635…
```
archive_df_copy = archive_df_copy.drop(columns=list(bad_df.columns))
```

Test

In [636…
```
percent_missing = bad_df.isnull().sum() * 100 / len(bad_df)
```

```
pd.DataFrame({"pourcentage de valeurs null (%)": percent_missing})
```

Out[636]:

	pourcentage de valeurs null (%)
in_reply_to_status_id	96.689304
in_reply_to_user_id	96.689304
retweeted_status_id	92.317487
retweeted_status_user_id	92.317487
retweeted_status_timestamp	92.317487

In [637…
```
archive_df_copy.columns
```

Out[637]:
```
Index(['tweet_id', 'date', 'source', 'text', 'expanded_urls',
       'rating_numerator', 'rating_denominator', 'name', 'stade_chien'],
      dtype='object')
```

Issue #3: `expanded_urls` contient aussi d'importantes valeurs manquantes .

Define

- Explorer la variable et voir si elle vaut d'être conservée.

Code

In [638…
```
np.random.seed(20)
archive_df_copy.loc[np.random.randint(0, len(archive_df_copy.index)), 'expanded_ur
```

Out[638]:
```
'https://twitter.com/dog_rates/status/841077006473256960/photo/1'
```

La variable en question contient aucun information utile pour la suite de l'analyse, nous prenons donc la liberté de la supprimer.

In [639…
```
archive_df_copy = archive_df_copy.drop(columns='expanded_urls')
```

Test

In [640…
```
archive_df_copy.columns
```

Out[640]:
```
Index(['tweet_id', 'date', 'source', 'text', 'rating_numerator',
       'rating_denominator', 'name', 'stade_chien'],
      dtype='object')
```

Issue #4: Notation pas du tout régulière, donc remise en cause.

Define

- Explorer la variable et voir si elle vaut d'être conservée.

Code

In [641…
```
def remplacement_hierarchique(df: pd.DataFrame, ref_col: str, hierarchical_cols: l

    """
    Parcourir itérativement les colonnes hiérarchiques et remplacez les valeurs `re

    Arguments :
        df (pd.DataFrame) : une trame de données sur laquelle nous voulons travail
        ref_col (str) : une référence (une colonne) pour référencer et attribuer ur
        hierarchical_cols (list) : la liste des colonnes hiérarchiques
        hierarchical_values (list) : liste des colonnes de valeurs hiérarchiques
    """
    for clef, col in enumerate(hierarchical_cols):
        filtered = df[(df[ref_col].isna()) & (df[col].notnull())]
        df.loc[filtered.index, ref_col] = filtered[hierarchical_values[clef]]
```

In [642…
```
# Trouver toutes les notes, y compris les valeurs fractionnaires
archive_df_copy['notes'] = archive_df_copy.text.str.findall(r"\d*\.?\d+\/\d+").str
archive_df_copy[['premier', 'second', 'notes']] = archive_df_copy.notes.str.split(

# Réinitialisation de l'index
archive_df_copy.reset_index(inplace=True)

# recursively replace the values
remplacement_hierarchique(archive_df_copy, 'notes', ['second', 'premier'], ['secon

# creating new numerator and denominator
archive_df_copy.ratings = archive_df_copy.notes.str.strip('.')
archive_df_copy[['new_numerator', 'new_denominator']] = np.round(archive_df_copy.n
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_5336\1246797570.py:12: UserWarning: P
andas doesn't allow columns to be created via a new attribute name - see https://p
andas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  archive_df_copy.ratings = archive_df_copy.notes.str.strip('.')
```

In [643…
```
# Remplacer les dénominateurs corrects ayant la base 10 avec les numérateurs
filtered = archive_df_copy[archive_df_copy.rating_denominator != archive_df_copy.n
archive_df_copy.loc[filtered.index, 'rating_numerator'] = filtered['new_numerator'
archive_df_copy.loc[filtered.index, 'rating_denominator'] = filtered['new_denomina
```

In [644…
```
# Remplacer les dénominateurs corrects ayant la base 10 avec les numérateurs
filtered = archive_df_copy.query('((new_numerator > rating_numerator) | (rating_nu
archive_df_copy.loc[filtered.index, 'rating_numerator'] = archive_df_copy.new_nume
```

In [645…
```
# Porter différentes plages de valeurs à 10
filtered = archive_df_copy.query('(new_numerator == rating_numerator) & (rating_de
archive_df_copy.loc[filtered.index, 'rating_numerator'] = np.round(archive_df_copy
archive_df_copy.loc[filtered.index, 'rating_denominator'] = 10
```

In [646…
```
# Remplacer les valeurs extrêmes par la moyenne
archive_df_copy.loc[archive_df_copy.rating_numerator > 15, 'rating_numerator'] = i
```

In [647…
```
# Supprimer les colonnes inutiles
archive_df_copy.drop(columns=['premier', 'second', 'notes', 'new_numerator', 'new_
```

Test

In [648…
```
print(archive_df_copy.rating_numerator.unique(), archive_df_copy.rating_denominato
```

```
[13 12 14 11 10  6 15  0  7  9  8  1  5  3  4  2] [10]
```

In [649…
```
archive_df_copy['rating_denominator']
```

Out[649]:
```
0       10
1       10
2       10
3       10
4       10
       ..
2351    10
2352    10
2353    10
2354    10
2355    10
Name: rating_denominator, Length: 2356, dtype: int64
```

Issue #5: Dans la variable `text` on découvre une autre variable cachée qu'est le `lien`

Define

- Vérifiez d'abord de quoi il s'agit.
- Supprimer les liens de tweet de la colonne `text`.

Code

In [650…
```
# Voir un exemple
np.random.seed(35)
archive_df_copy.loc[np.random.randint(0, len(archive_df_copy.index)), 'text']
```

Out[650]:
```
"Guys this really needs to stop. We've been over this way too many times. This is
a giraffe. We only rate dogs.. 7/10 https://t.co/yavgkHYPOC"
```

In [651…
```
# On sait mainteant de quoi il s'agit, les liens sont inscrits en fin de texte.
# Donc nous pouvons les remplacer par une chaîne vide ou un point en utilisant rege
archive_df_copy.text = archive_df_copy.text.str.replace("(https://[\w\/\.]+$)", ".'
archive_df_copy.text = archive_df_copy.text.str.strip()
```

Test

In [652…
```
np.random.seed(54)
archive_df_copy.loc[np.random.randint(0, len(archive_df_copy.index)), 'text']
```

Out[652]:
```
'This is Lilli Bee &amp;amp; Honey Bear. Unfortunately, they were both born with no ey
es. So heckin sad. Both 11/10 .'
```

Issue #6: La variable `source` affiche plus que la vraie source essentiel qu'est l'appareil ou le moyen d'acces de Twitter.

Define

- Voir de quoi il s'agit
- Extraire la vraie source en les balises html et par le texte correspondant

Code

In [653…
```
# Aperçu du problème
archive_df_copy.groupby('source').source.count()
```

Out[653]:
```
source
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a
>      11
Name: source, dtype: int64
```

In [654…
```
# Extraire la source cachée avec regex encore
#.text.str.replace(
maliste = []
for s in list(archive_df_copy.source) :
    maliste.append(re.search(r'\">(.*?)</a>', s).group(1))
archive_df_copy.source = maliste
```

Test

In [655…
```
archive_df_copy.groupby('source').source.count()
```

Out[655]:
```
source
TweetDeck                11
Twitter Web Client       33
Twitter for iPhone     2221
Vine - Make a Scene      91
Name: source, dtype: int64
```

Issue #7: On voit des noms de chiens incorrects dans le jeu de données.

Define

- Remplacer les noms ne commençant par une majuscule par NaN.
- Supprimer les noms constituant une seule lettre.

Code

In [656…
```
# Remplacer les noms ne commençant par une majuscule par NaN.
archive_df_copy['name'] = archive_df_copy['name'].apply(lambda x: x if str(x).lowe
```

In [657…
```
# Supprimer les noms constituant une seule lettre.
archive_df_copy['name'] = archive_df_copy['name'].apply(lambda x: x if len(str(x))
```

Test

In [658…
```
archive_df_copy['name'].unique()
```

Out[658]:
```
array(['Phineas', 'Tilly', 'Archie', 'Darla', 'Franklin', 'None', 'Jax',
       'Zoey', 'Cassie', 'Koda', 'Bruno', 'Ted', 'Stuart', 'Oliver',
       'Jim', 'Zeke', 'Ralphus', 'Canela', 'Gerald', 'Jeffrey', nan,
       'Maya', 'Mingus', 'Derek', 'Roscoe', 'Waffles', 'Jimbo', 'Maisey',
       'Lilly', 'Earl', 'Lola', 'Kevin', 'Yogi', 'Noah', 'Bella',
       'Grizzwald', 'Rusty', 'Gus', 'Stanley', 'Alfy', 'Koko', 'Rey',
       'Gary', 'Elliot', 'Louis', 'Jesse', 'Romeo', 'Bailey', 'Duddles',
       'Jack', 'Emmy', 'Steven', 'Beau', 'Snoopy', 'Shadow', 'Terrance',
       'Aja', 'Penny', 'Dante', 'Nelly', 'Ginger', 'Benedict', 'Venti',
       'Goose', 'Nugget', 'Cash', 'Coco', 'Jed', 'Sebastian', 'Walter',
       'Sierra', 'Monkey', 'Harry', 'Kody', 'Lassie', 'Rover', 'Napolean',
       'Dawn', 'Boomer', 'Cody', 'Rumble', 'Clifford', 'Dewey', 'Scout',
       'Gizmo', 'Cooper', 'Harold', 'Shikha', 'Jamesy', 'Lili', 'Sammy',
       'Meatball', 'Paisley', 'Albus', 'Neptune', 'Quinn', 'Belle',
       'Zooey', 'Dave', 'Jersey', 'Hobbes', 'Burt', 'Lorenzo', 'Carl',
       'Jordy', 'Milky', 'Trooper', 'Winston', 'Sophie', 'Wyatt', 'Rosie',
       'Thor', 'Oscar', 'Luna', 'Callie', 'Cermet', 'George', 'Marlee',
       'Arya', 'Einstein', 'Alice', 'Rumpole', 'Benny', 'Aspen', 'Jarod',
       'Wiggles', 'General', 'Sailor', 'Astrid', 'Iggy', 'Snoop', 'Kyle',
       'Leo', 'Riley', 'Gidget', 'Noosh', 'Odin', 'Jerry', 'Charlie',
       'Georgie', 'Rontu', 'Cannon', 'Furzey', 'Daisy', 'Tuck', 'Barney',
       'Vixen', 'Jarvis', 'Mimosa', 'Pickles', 'Bungalo', 'Brady',
       'Margo', 'Sadie', 'Hank', 'Tycho', 'Stephan', 'Indie', 'Winnie',
       'Bentley', 'Ken', 'Max', 'Maddie', 'Pipsy', 'Monty', 'Sojourner',
       'Odie', 'Arlo', 'Sunny', 'Vincent', 'Lucy', 'Clark', 'Mookie',
       'Meera', 'Buddy', 'Ava', 'Rory', 'Eli', 'Ash', 'Tucker', 'Tobi',
       'Chester', 'Wilson', 'Sunshine', 'Lipton', 'Gabby', 'Bronte',
       'Poppy', 'Rhino', 'Willow', 'Orion', 'Eevee', 'Smiley', 'Logan',
       'Moreton', 'Klein', 'Miguel', 'Emanuel', 'Kuyu', 'Dutch', 'Pete',
       'Scooter', 'Reggie', 'Kyro', 'Samson', 'Loki', 'Mia', 'Malcolm',
       'Dexter', 'Alfie', 'Fiona', 'Mutt', 'Bear', 'Doobert', 'Beebop',
       'Alexander', 'Sailer', 'Brutus', 'Kona', 'Boots', 'Ralphie',
       'Phil', 'Cupid', 'Pawnd', 'Pilot', 'Ike', 'Mo', 'Toby', 'Sweet',
       'Pablo', 'Nala', 'Balto', 'Crawford', 'Gabe', 'Mattie', 'Jimison',
       'Hercules', 'Duchess', 'Harlso', 'Sampson', 'Sundance', 'Luca',
       'Flash', 'Finn', 'Peaches', 'Howie', 'Jazzy', 'Anna', 'Bo',
       'Seamus', 'Wafer', 'Chelsea', 'Tom', 'Moose', 'Florence', 'Autumn',
       'Dido', 'Eugene', 'Herschel', 'Strudel', 'Tebow', 'Chloe', 'Betty',
       'Timber', 'Binky', 'Dudley', 'Comet', 'Larry', 'Levi', 'Akumi',
       'Titan', 'Olivia', 'Alf', 'Oshie', 'Bruce', 'Chubbs', 'Sky',
       'Atlas', 'Eleanor', 'Layla', 'Rocky', 'Baron', 'Tyr', 'Bauer',
       'Swagger', 'Brandi', 'Mary', 'Moe', 'Halo', 'Augie', 'Craig',
       'Sam', 'Hunter', 'Pavlov', 'Maximus', 'Wallace', 'Ito', 'Milo',
       'Ollie', 'Cali', 'Lennon', 'Major', 'Duke', 'Reginald', 'Sansa',
       'Shooter', 'Django', 'Diogi', 'Sonny', 'Philbert', 'Marley',
       'Severus', 'Ronnie', 'Anakin', 'Bones', 'Mauve', 'Chef', 'Doc',
       'Sobe', 'Longfellow', 'Mister', 'Iroh', 'Baloo', 'Stubert',
       'Paull', 'Timison', 'Davey', 'Pancake', 'Tyrone', 'Snicku', 'Ruby',
       'Brody', 'Rizzy', 'Mack', 'Butter', 'Nimbus', 'Laika', 'Dobby',
       'Juno', 'Maude', 'Lily', 'Newt', 'Benji', 'Nida', 'Robin',
       'Monster', 'BeBe', 'Remus', 'Mabel', 'Misty', 'Happy', 'Mosby',
       'Maggie', 'Leela', 'Ralphy', 'Brownie', 'Meyer', 'Stella', 'Frank',
       'Tonks', 'Lincoln', 'Oakley', 'Dale', 'Rizzo', 'Arnie', 'Pinot',
       'Dallas', 'Hero', 'Frankie', 'Stormy', 'Mairi', 'Loomis', 'Godi',
       'Kenny', 'Deacon', 'Timmy', 'Harper', 'Chipson', 'Combo', 'Dash',
       'Bell', 'Hurley', 'Jay', 'Mya', 'Strider', 'Wesley', 'Solomon',
       'Huck', 'Blue', 'Finley', 'Sprinkles', 'Heinrich', 'Shakespeare',
       'Fizz', 'Chip', 'Grey', 'Roosevelt', 'Gromit', 'Willem', 'Dakota',
       'Dixie', 'Al', 'Jackson', 'Carbon', 'DonDon', 'Kirby', 'Lou',
       'Nollie', 'Chevy', 'Tito', 'Louie', 'Rupert', 'Rufus', 'Brudge',
       'Shadoe', 'Colby', 'Angel', 'Brat', 'Tove', 'Aubie', 'Kota', 'Eve',
       'Glenn', 'Shelby', 'Sephie', 'Bonaparte', 'Albert', 'Wishes',
       'Rose', 'Theo', 'Rocco', 'Fido', 'Emma', 'Spencer', 'Lilli',
       'Boston', 'Brandonald', 'Corey', 'Leonard', 'Chompsky', 'Beckham',
```

```
'Devón', 'Gert', 'Watson', 'Rubio', 'Keith', 'Dex', 'Carly', 'Ace',
'Tayzie', 'Grizzie', 'Fred', 'Gilbert', 'Zoe', 'Stewie', 'Calvin',
'Lilah', 'Spanky', 'Jameson', 'Piper', 'Atticus', 'Blu',
'Dietrich', 'Divine', 'Tripp', 'Cora', 'Huxley', 'Keurig',
'Bookstore', 'Linus', 'Abby', 'Shaggy', 'Shiloh', 'Gustav',
'Arlen', 'Percy', 'Lenox', 'Sugar', 'Harvey', 'Blanket', 'Geno',
'Stark', 'Beya', 'Kilo', 'Kayla', 'Maxaroni', 'Doug', 'Edmund',
'Aqua', 'Theodore', 'Chase', 'Rorie', 'Simba', 'Charles', 'Bayley',
'Axel', 'Storkson', 'Remy', 'Chadrick', 'Kellogg', 'Buckley',
'Livvie', 'Terry', 'Hermione', 'Ralpher', 'Aldrick', 'Rooney',
'Crystal', 'Ziva', 'Stefan', 'Pupcasso', 'Puff', 'Flurpson',
'Coleman', 'Enchilada', 'Raymond', 'Rueben', 'Cilantro', 'Karll',
'Sprout', 'Blitz', 'Bloop', 'Lillie', 'Ashleigh', 'Kreggory',
'Sarge', 'Luther', 'Ivar', 'Jangle', 'Schnitzel', 'Panda',
'Berkeley', 'Ralphé', 'Charleson', 'Clyde', 'Harnold', 'Sid',
'Pippa', 'Otis', 'Carper', 'Bowie', 'Alexanderson', 'Suki',
'Barclay', 'Skittle', 'Ebby', 'Flávio', 'Smokey', 'Link',
'Jennifur', 'Ozzy', 'Bluebert', 'Stephanus', 'Bubbles', 'Zeus',
'Bertson', 'Nico', 'Michelangelope', 'Siba', 'Calbert', 'Curtis',
'Travis', 'Thumas', 'Kanu', 'Lance', 'Opie', 'Kane', 'Olive',
'Chuckles', 'Staniel', 'Sora', 'Beemo', 'Gunner', 'Lacy', 'Tater',
'Olaf', 'Cecil', 'Vince', 'Karma', 'Billy', 'Walker', 'Rodney',
'Klevin', 'Malikai', 'Bobble', 'River', 'Jebberson', 'Remington',
'Farfle', 'Jiminus', 'Clarkus', 'Finnegus', 'Cupcake', 'Kathmandu',
'Ellie', 'Katie', 'Kara', 'Adele', 'Zara', 'Ambrose', 'Jimothy',
'Bode', 'Terrenth', 'Reese', 'Chesterson', 'Lucia', 'Bisquick',
'Ralphson', 'Socks', 'Rambo', 'Rudy', 'Fiji', 'Rilo', 'Bilbo',
'Coopson', 'Yoda', 'Millie', 'Chet', 'Crouton', 'Daniel', 'Kaia',
'Murphy', 'Dotsy', 'Eazy', 'Coops', 'Fillup', 'Miley', 'Charl',
'Reagan', 'Yukon', 'CeCe', 'Cuddles', 'Claude', 'Jessiga',
'Carter', 'Ole', 'Pherb', 'Blipson', 'Reptar', 'Trevith', 'Berb',
'Bob', 'Colin', 'Brian', 'Oliviér', 'Grady', 'Kobe', 'Freddery',
'Bodie', 'Dunkin', 'Wally', 'Tupawc', 'Amber', 'Edgar', 'Teddy',
'Kingsley', 'Brockly', 'Richie', 'Molly', 'Vinscent', 'Cedrick',
'Hazel', 'Lolo', 'Eriq', 'Phred', 'Oddie', 'Maxwell', 'Geoff',
'Covach', 'Durg', 'Fynn', 'Ricky', 'Herald', 'Lucky', 'Ferg',
'Trip', 'Clarence', 'Hamrick', 'Brad', 'Pubert', 'Frönq', 'Derby',
'Lizzie', 'Ember', 'Blakely', 'Opal', 'Marq', 'Kramer', 'Barry',
'Gordon', 'Baxter', 'Mona', 'Horace', 'Crimson', 'Birf', 'Hammond',
'Lorelei', 'Marty', 'Brooks', 'Petrick', 'Hubertson', 'Gerbald',
'Oreo', 'Bruiser', 'Perry', 'Bobby', 'Jeph', 'Obi', 'Tino',
'Kulet', 'Sweets', 'Lupe', 'Tiger', 'Jiminy', 'Griffin', 'Banjo',
'Brandy', 'Lulu', 'Darrel', 'Taco', 'Joey', 'Patrick', 'Kreg',
'Todo', 'Tess', 'Ulysses', 'Toffee', 'Apollo', 'Asher', 'Glacier',
'Chuck', 'Champ', 'Ozzie', 'Griswold', 'Cheesy', 'Moofasa',
'Hector', 'Goliath', 'Kawhi', 'Emmie', 'Penelope', 'Willie',
'Rinna', 'Mike', 'William', 'Dwight', 'Evy', 'Rascal', 'Linda',
'Tug', 'Tango', 'Grizz', 'Jerome', 'Crumpet', 'Jessifer', 'Izzy',
'Ralph', 'Sandy', 'Humphrey', 'Tassy', 'Juckson', 'Chuq', 'Tyrus',
'Karl', 'Godzilla', 'Vinnie', 'Kenneth', 'Herm', 'Bert', 'Striker',
'Donny', 'Pepper', 'Bernie', 'Buddah', 'Lenny', 'Arnold', 'Zuzu',
'Mollie', 'Laela', 'Tedders', 'Superpup', 'Rufio', 'Jeb', 'Rodman',
'Jonah', 'Chesney', 'Henry', 'Bobbay', 'Mitch', 'Kaiya', 'Acro',
'Aiden', 'Obie', 'Dot', 'Shnuggles', 'Kendall', 'Jeffri', 'Steve',
'Mac', 'Fletcher', 'Kenzie', 'Pumpkin', 'Schnozz', 'Gustaf',
'Cheryl', 'Ed', 'Leonidas', 'Norman', 'Caryl', 'Scott', 'Taz',
'Darby', 'Jackie', 'Jazz', 'Franq', 'Pippin', 'Rolf', 'Snickers',
'Ridley', 'Cal', 'Bradley', 'Bubba', 'Tuco', 'Patch', 'Mojo',
'Batdog', 'Dylan', 'Mark', 'JD', 'Alejandro', 'Scruffers', 'Pip',
'Julius', 'Tanner', 'Sparky', 'Anthony', 'Holly', 'Jett', 'Amy',
'Sage', 'Andy', 'Mason', 'Trigger', 'Antony', 'Creg', 'Traviss',
'Gin', 'Jeffrie', 'Danny', 'Ester', 'Pluto', 'Bloo', 'Edd',
'Willy', 'Herb', 'Damon', 'Peanut', 'Nigel', 'Butters', 'Sandra',
'Fabio', 'Randall', 'Liam', 'Tommy', 'Ben', 'Raphael', 'Julio',
```

```
'Andru', 'Kloey', 'Shawwn', 'Skye', 'Kollin', 'Ronduh', 'Billl',
'Saydee', 'Dug', 'Tessa', 'Sully', 'Kirk', 'Ralf', 'Clarq',
'Jaspers', 'Samsom', 'Harrison', 'Chaz', 'Jeremy', 'Jaycob',
'Lambeau', 'Ruffles', 'Amélie', 'Bobb', 'Banditt', 'Kevon',
'Winifred', 'Hanz', 'Churlie', 'Zeek', 'Timofy', 'Maks',
'Jomathan', 'Kallie', 'Marvin', 'Spark', 'Gòrdón', 'Jo', 'DayZ',
'Jareld', 'Torque', 'Ron', 'Skittles', 'Cleopatricia', 'Erik',
'Stu', 'Tedrick', 'Filup', 'Kial', 'Naphaniel', 'Dook', 'Hall',
'Philippe', 'Biden', 'Fwed', 'Genevieve', 'Joshwa', 'Bradlay',
'Clybe', 'Keet', 'Carll', 'Jockson', 'Josep', 'Lugan',
'Christoper'], dtype=object)
```

Issue #8: Les valeurs supposées vides sont remplies par la valeur `'None'` au lieu de `None`.

Define

- Remplacer les valeurs `'None'` par `NaN`
- En profiter pour emplacer aussi les chaînes vides (`''`) par `NaN`

Code

In [659…
```
archive_df_copy.name.replace('None', np.nan, inplace=True)
for col in list(archive_df_copy.columns) :
    archive_df_copy[col].replace("", np.nan, inplace=True)
```

Test

In [660…
```
archive_df_copy[archive_df_copy.eq('None').any(1)]
```

Out[660]:

index	tweet_id	date	source	text	rating_numerator	rating_denominator	name	stade_chien

In [661…
```
archive_df_copy[archive_df_copy.eq('').any(1)]
```

Out[661]:

index	tweet_id	date	source	text	rating_numerator	rating_denominator	name	stade_chien

Tidiness issues #2: Les variables `rating_numerator` et `rating_denominator` peuvent faire une seule variable au lieu de deux puisqu'il s'agit d'une note, même si on est en façe d'une notation assez particulière.

Define

- Créer une nouvelle colonne appelé `single_rate` (une seule note)
- Et `single_rate` sera égal à `rating_numerator` / `rating_denominator`
- Supprimer les variables `rating_numerator` et `rating_denominator` reste optionnel comme le fait de créer une variable `single_rate`, puisque que tous les

dénominateurs sont déjà remis à 10 (normalisation).

Code

In [662…
```
archive_df_copy['single_rate'] = archive_df_copy.rating_numerator / archive_df_copy
```

Test

In [663…
```
archive_df_copy[['single_rate','rating_numerator','rating_denominator']].sample(5)
```

Out[663]:

	single_rate	rating_numerator	rating_denominator
469	1.2	12	10
1728	1.0	10	10
1352	1.0	10	10
761	1.1	11	10
186	1.4	14	10

- Données `pred_df`

Issue #9: Trois prédictions pour une seule image

Define

Rappel:Les prédictions sont hiérarchiques, c'est-à-dire que `p1_conf` est le plus élevé et `p2_conf` est le second et ainsi de suite.

- Créer une colonne `race_chien`
- Filtrer les points de données avec des valeurs `True` pour `p1_dog`
- Attribuer la valeur de `p1` à la colonne `race_chien` pour les points de données filtrés ci-dessus
- Répéter les étapes 2 et 3 pour `p2_dog` et `p3_dog`, si la prédiction pour le précédent n'est pas dog
- Et enfin supprimer les colonnes inutiles

Code

In [664…
```
# Creéer une colonne Nan
pred_df_copy['race_chien'] = np.nan

# Remplacer les valeurs False par NaN
for column in ['p1_dog', 'p2_dog', 'p3_dog']:
    pred_df_copy[column].replace(False, np.nan, inplace=True)

# Remplacer itérativement les valeurs
remplacement_hierarchique(pred_df_copy, 'race_chien', ['p1_dog', 'p2_dog', 'p3_dog

# Supprimer les colonnes inutiles
pred_df_copy.drop(columns=['img_num','p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf', '
```

Test

In [665…
```
pred_df_copy.columns
```

Out[665]:
```
Index(['tweet_id', 'jpg_url', 'race_chien'], dtype='object')
```

In [666…
```
pred_df_copy.race_chien.head()
```

Out[666]:
```
0    Welsh_springer_spaniel
1                   redbone
2            German_shepherd
3       Rhodesian_ridgeback
4         miniature_pinscher
Name: race_chien, dtype: object
```

Issue #10: Le nom des races des chiens doit être redéfini.

Define

- Mettre en majuscule toutes les petites lettres de départ

Code

In [667…
```
# Mise en Majscule
pred_df_copy.race_chien = pred_df_copy.race_chien.str.capitalize()
```

Test

In [668…
```
np.random.seed(45)
pred_df_copy.race_chien.sample(5)
```

Out[668]:
```
737          Borzoi
815         Samoyed
1011    Maltese_dog
782         Samoyed
417      Pomeranian
Name: race_chien, dtype: object
```

- Données `tweet_df`

Issue #11: Ce dataset a énormément de valeurs manquantes, contient très probablement des variables inutiles

Define

- Revoir de quoi en est-il !
- Supprimes les variables inutles ou conserver celles utiles uniquement.

Code

In [669…
```
# Pourcetage de valeurs null
dfpercent = pd.DataFrame(tweet_df_copy.isnull().sum()/len(tweet_df_copy)*100,columı
dfpercent
```

Out[669]:

	valeurs manquantes (%)
created_at	0.000000
id	0.000000
id_str	0.000000
full_text	0.000000
truncated	0.000000
display_text_range	0.000000
entities	0.000000
retweeted	0.000000
source	0.000000
favorited	0.000000
favorite_count	0.000000
retweet_count	0.000000
is_quote_status	0.000000
lang	0.000000
user	0.000000
possibly_sensitive_appealable	6.074766
possibly_sensitive	6.074766
extended_entities	11.937128
retweeted_status	92.395922
in_reply_to_screen_name	96.686491
in_reply_to_user_id	96.686491
in_reply_to_status_id_str	96.686491
in_reply_to_status_id	96.686491
in_reply_to_user_id_str	96.686491
quoted_status_id	98.768054
quoted_status_id_str	98.768054
quoted_status	98.810535
place	99.957519
coordinates	100.000000
contributors	100.000000
geo	100.000000

In [670…

```
# Conserver celles qui ont moins de 20% de valeurs manquantes
dfpercent = dfpercent[dfpercent.lt(20)].dropna()
print(dfpercent)
tweet_df_copy = tweet_df_copy[list(dfpercent.index)]
```

```
                                valeurs manquantes (%)
created_at                                    0.000000
id                                            0.000000
id_str                                        0.000000
full_text                                     0.000000
truncated                                     0.000000
display_text_range                            0.000000
entities                                      0.000000
retweeted                                     0.000000
source                                        0.000000
favorited                                     0.000000
favorite_count                                0.000000
retweet_count                                 0.000000
is_quote_status                               0.000000
lang                                          0.000000
user                                          0.000000
possibly_sensitive_appealable                 6.074766
possibly_sensitive                            6.074766
extended_entities                            11.937128
```

Nous avons également vu un plus tôt dans la partie collecte de données que certaines variables n'apportaient pas d'informations significative, d'autres contiennent des valeurs uniques et d'autres se répètent.

In [671…
```
tweet_df_copy = tweet_df_copy.drop(columns =['possibly_sensitive_appealable', 'disp
                                             'possibly_sensitive','extended_entitie
```

Egalement d'autres variables existent déjà, donc inutiles pour la fusion sachant qu'ils ne sont pas des identifiants.

In [672…
```
tweet_df_copy = tweet_df_copy.drop(columns = ['source','created_at','full_text'])
```

Test

In [673…
```
tweet_df_copy.sample(5)
```

Out[673]:

	id	retweeted	favorited	favorite_count	retweet_count	is_quote_status
825	769212283578875904	False	False	5980	1969	False
2100	670679630144274432	False	False	799	315	False
1166	721001180231503872	False	False	2748	686	False
98	872967104147763200	False	False	28031	5669	False
1327	705898680587526145	False	False	2597	643	False

Issue #12: Le nom des langues dans la variable `lang` a été abrégée, elle doit être explicite !

Define

- Revoir de quoi en est-il !
- Utiliser uniquement les colonnes nécessaires à la récupération du nom complet de langue à partir `lang_df_copy`.

- Joindre les bases `tweet_df_copy` et `lang_df_copy` pour récupérer le nom de la langue.
- Supprimer les colonnes inutiles et renommer si nécessaire.

Code

In [674…
```
# Revoir de quoi en est-il !
tweet_df_copy.lang.value_counts()
```

Out[674]:
```
en     2336
und       7
in        3
nl        3
eu        1
es        1
tl        1
ro        1
et        1
Name: lang, dtype: int64
```

In [675…
```
# Utiliser uniquement les colonnes nécessaires à la récupération du nom complet de
lang_df_copy = lang_df_copy[['639-1','Nom de la langue']] # Voir partie collecte d
```

In [676…
```
# Joindre les bases `tweet_df_copy` et `lang_df_copy` pour récupérer le nom de la
joinlang_df = tweet_df_copy.merge(lang_df_copy, how='inner', left_on='lang', right_
joinlang_df
```

Out[676]:

	id	retweeted	favorited	favorite_count	retweet_count	is_quote_status
0	892420643555336193	False	False	39467	8853	False
1	892177421306343426	False	False	33819	6514	False
2	891815181378084864	False	False	25461	4328	False
3	891689557279858688	False	False	42908	8964	False
4	891327558926688256	False	False	41048	9774	False
...	...	...	...	...	...	...
2339	674790488185167872	False	False	1180	278	False
2340	667435689202614272	False	False	326	89	False
2341	668967877119254528	False	False	161	25	False
2342	667550882905632768	False	False	0	34	False
2343	667192066997374976	False	False	414	115	False

2344 rows × 9 columns

In [677…
```
# Supprimer les colonnes inutiles
joinlang_df = joinlang_df.drop(columns = ['lang','639-1'])
joinlang_df.sample(5)
```

Out[677]:

	id	retweeted	favorited	favorite_count	retweet_count	is_quote_status
213	850753642995093505	False	False	33348	10352	False
1983	672523490734551040	False	False	696	189	False
640	793150605191548928	False	False	6909	1984	False
1677	681320187870711809	False	False	2918	863	False
2003	672125275208069120	False	False	2578	1253	False

In [678…

```
# Le nom de la variable (Nom de la langue) est trop longue pour être réutilisé.
joinlang_df.rename(columns = {'Nom de la langue':'langue'}, inplace = True)
joinlang_df.sample(5)
```

Out[678]:

	id	retweeted	favorited	favorite_count	retweet_count	is_quote_status
379	827199976799354881	False	False	11659	2579	False
664	789986466051088384	False	False	10369	2704	False
705	784517518371221505	False	False	10039	2970	False
918	754874841593970688	False	False	0	9193	False
1383	700002074055016451	False	False	3627	1529	False

Test

In [679…

```
tweet_df_copy = joinlang_df.copy()
tweet_df_copy
```

Out[679]:

	id	retweeted	favorited	favorite_count	retweet_count	is_quote_status
0	892420643555336193	False	False	39467	8853	False
1	892177421306343426	False	False	33819	6514	False
2	891815181378084864	False	False	25461	4328	False
3	891689557279858688	False	False	42908	8964	False
4	891327558926688256	False	False	41048	9774	False
...	...	...	...	...	...	...
2339	674790488185167872	False	False	1180	278	False
2340	667435689202614272	False	False	326	89	False
2341	668967877119254528	False	False	161	25	False
2342	667550882905632768	False	False	0	34	False
2343	667192066997374976	False	False	414	115	False

2344 rows × 7 columns

Tidiness issues #3: Les 3 premiers datasets ne doivent former qu'une seule unité d'observation.

Define

- Vérifier les tailles avant jointure finale
- Vérifier la taille finale

Code

In [680…
```
# Vérifier les tailles avant jointure finale
archive_df_copy.shape, pred_df_copy.shape, tweet_df_copy.shape
```

Out[680]:
```
((2356, 10), (2075, 3), (2344, 7))
```

In [681…
```
# Vérifier la taille finale
all_data = archive_df_copy.merge(pred_df_copy, how='inner', on='tweet_id').merge(tw
all_data.shape
```

Out[681]:
```
(2071, 19)
```

Test

In [682…
```
all_data.head(5)
```

Out[682]:

	index	tweet_id	date	source	text	rating_numerator	rating_denc
0	0	892420643555336193	2017-08-01 16:23:56+00:00	Twitter for iPhone	This is Phineas. He's a mystical boy. Only eve...	13	
1	1	892177421306343426	2017-08-01 00:17:27+00:00	Twitter for iPhone	This is Tilly. She's just checking pup on you....	13	
2	2	891815181378084864	2017-07-31 00:18:03+00:00	Twitter for iPhone	This is Archie. He is a rare Norwegian Pouncin...	12	
3	3	891689557279858688	2017-07-30 15:58:51+00:00	Twitter for iPhone	This is Darla. She commenced a snooze mid meal...	13	
4	4	891327558926688256	2017-07-29 16:00:24+00:00	Twitter for iPhone	This is Franklin. He would like you to stop ca...	12	

Storing Data

Enregistrer l'ensemble de données principal collecté, évalué et nettoyé dans un fichier CSV nommé "twitter_archive_master.csv".

In [683…
```
# Supprimer les colonnes index,id qui sont inutiles
all_data.drop(columns=['index', 'id'], inplace=True)

# Stocker des données
all_data.to_csv('datasets/twitter_archive_master.csv', index=False)
```

Analyzing and Visualizing Data

Dans cette section, nous allons analyser et visualiser nos données mélangées. Pour la résussite de cette partie, nous devons produire au moins **trois (3) aperçus et une (1) visualisation.**

Getting Insights

- Quelle est la valeur moyenne des notes ?
  - Pour les notes supérieures à 10
  - Pour les notes inférieures à 10
  - Pour chaque année
- Quel sont les ratios pour les sources des tweets ?
- Quel est le nom de chien le plus populaire ?
- Quelle est la race de chien la plus populaire ?
- Quel est le ratio des stades de chien ?
- Quel est le tweet le plus retweeté ?
  - Qu'en est il du chiffre de retweet Annuel ?
- Le nombre de retweets augmente-t-il avec le temps ?
- Quel est le chien le plus favorisé ou préféré ?
- Quelle est la langue la plus utilisée dans les tweets ?

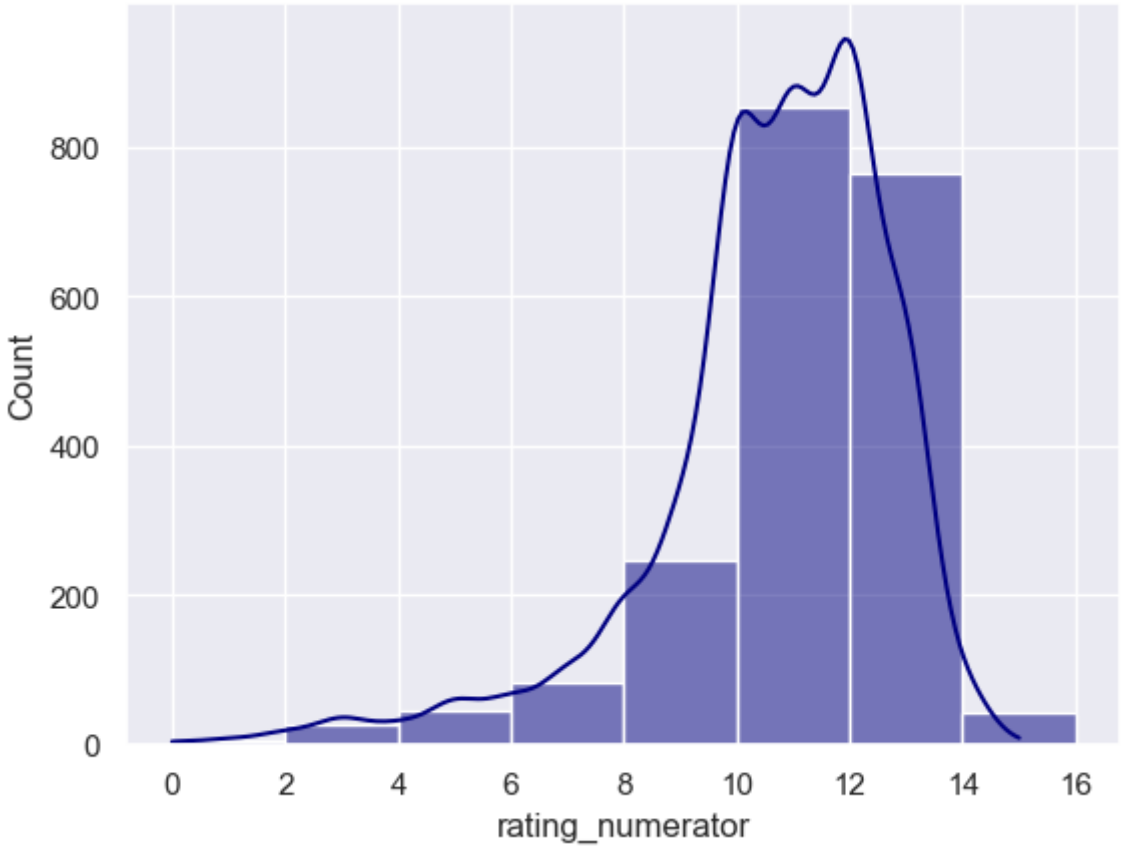
Insights:

Q#1:

- La Moyenne générale des notes est de *11* tandis que les Moyennes des notes > 10 et < 10 sont respectivement *12.0* et *7.0*.
- *2017* est l'année qui a obtenu le plus grande moyenne, avec une tendance croissante au fil du temps.

In [684…
```
# Aperçu de l'histograme de notes
sns.set(style="darkgrid")
sns.histplot(data=all_data.rating_numerator,color='navy', binwidth = 2, kde=True)
```

Out[684]:
```
<AxesSubplot: xlabel='rating_numerator', ylabel='Count'>
```

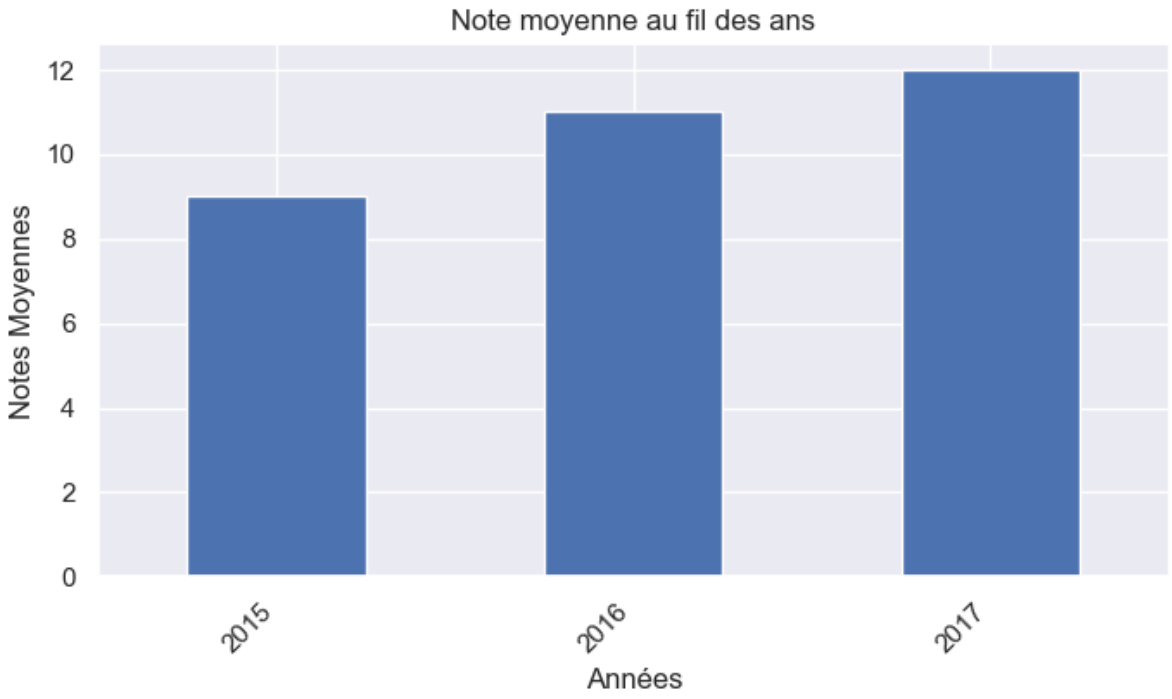


In [685…
```
# Regroupement par année
group_an = all_data.groupby(all_data.date.dt.year)
```

In [686…
```
# Note moyenne générale
print("Moyenne générale des notes : ",np.round(all_data.rating_numerator.mean(), 0
# plus grand que 10
print("Moyenne des notes > 10 : ", np.round(all_data[all_data.rating_numerator > 1(
# plus petit que 10
print("Moyenne des notes < 10 : ", np.round(all_data[all_data.rating_numerator < 1(
```

```
Moyenne générale des notes :  11
Moyenne des notes > 10 :  12.0
Moyenne des notes < 10 :  7.0
```

In [687…
```
# Note à l'année
notes_an = np.round(group_an.rating_numerator.mean()).astype(int)

plt.figure(figsize=(8, 4))
notes_an.plot(kind='bar')
plt.title("Note moyenne au fil des ans")
plt.xlabel("Années")
plt.ylabel("Notes Moyennes")
plt.xticks(rotation=45, horizontalalignment='right');
```

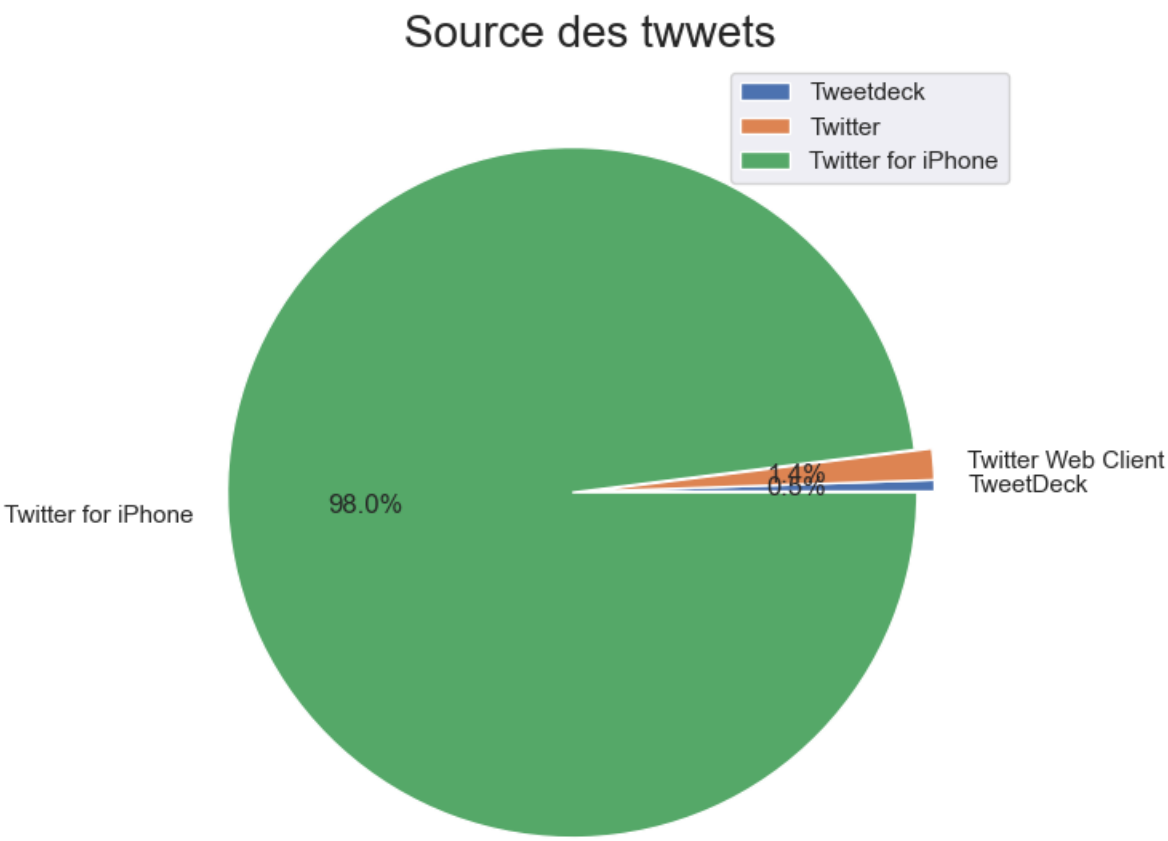


Q#1:

- *98.0 %* des tweets sont postés à partir de l'application Tweeter pour *Iphone*,cela est dû au fait que les tweets eux-même proviennent du compte @WeRateDodgs. Donc le seul utilisateur de ce compte préfère utiliser l'appli sur son iphone pour poster ses tweets.

In [688…
```
tweet_sources = all_data.groupby('source').count()[['tweet_id']]
tweet_sources.rename(columns={'tweet_id': 'source_count'}, inplace=True)
tweet_sources['source_percentage'] = tweet_sources.source_count / tweet_sources.sou
tweet_sources['source_percentage'].plot.pie(figsize=(7,7), autopct='%1.1f%%',
        explode=(0,0,0.05))
plt.title("Source des twwets", {'fontsize': 20})
plt.legend(["Tweetdeck", "Twitter", "Twitter for iPhone"])
plt.ylabel("  ")
```

Out[688]:
```
Text(0, 0.5, '  ')
```

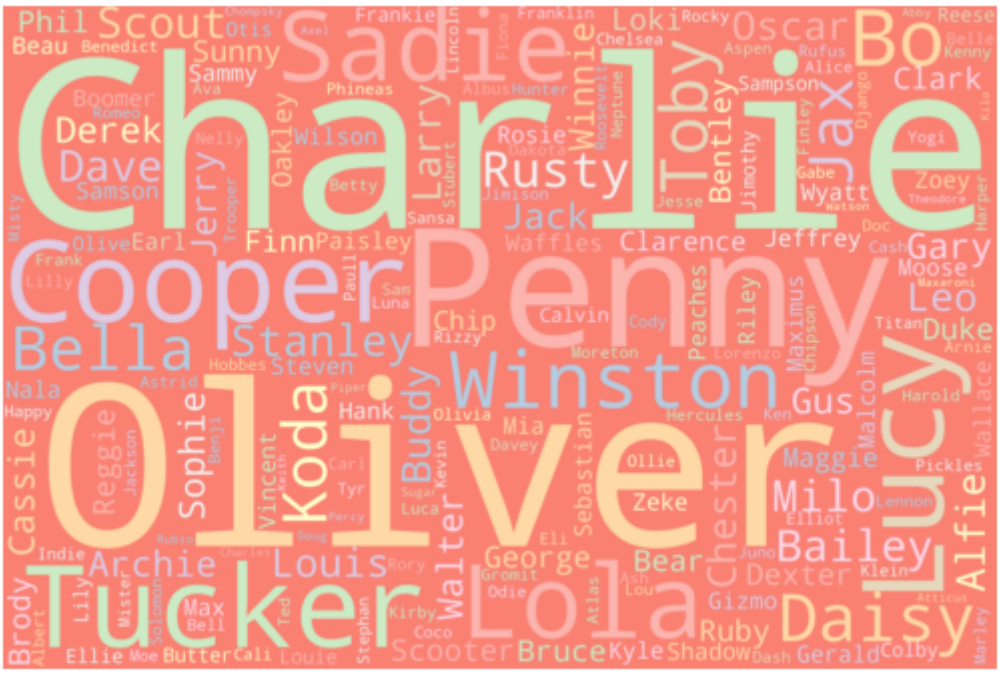


Q#3:

- Pas besoin de chiffres ou de graphes pour savoir que *Charlie* est le nom de chien le plus utilisé, comme on le dit souvent "une image vaut mieux que milles mots". Même si dans *WordCloud* c'est les mots qui forment l'image, mdr... ref

In [689…

```
# Générer du text avec le nom des chiens
text = " ".join(name for name in list(all_data.name.dropna()))

# Générer un WordCloud sur le text précédent
word_cloud = WordCloud(
    width=3000,
    height=2000,
    random_state=1,
    background_color="salmon",
    colormap="Pastel1",
    collocations=False,
    stopwords=STOPWORDS,
    max_font_size=1500,
    ).generate(text)

# sauvegrder l'image
word_cloud.to_file("img/Noms_de_chien_les_plus_populaires.png")

# Afficher le Word Cloud généré
plt.imshow(word_cloud,interpolation="bilinear")
plt.axis("off")
plt.show()
```

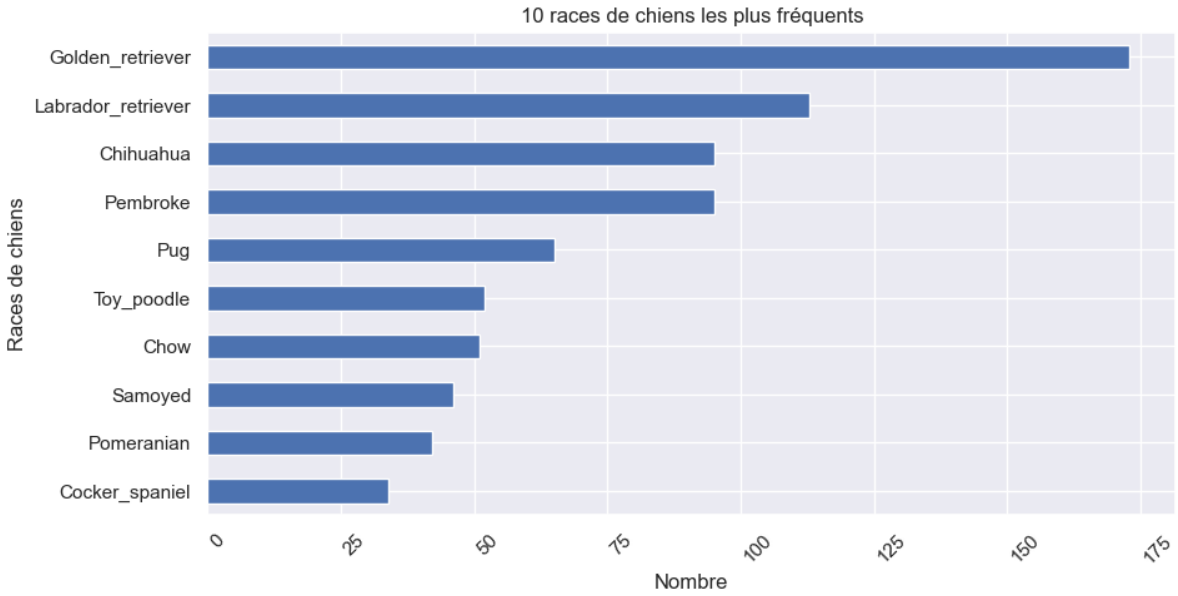


Q#4:

- Le *Golden retriever* occupe la première place des races de chiens les plus fréquents sur les tweets.

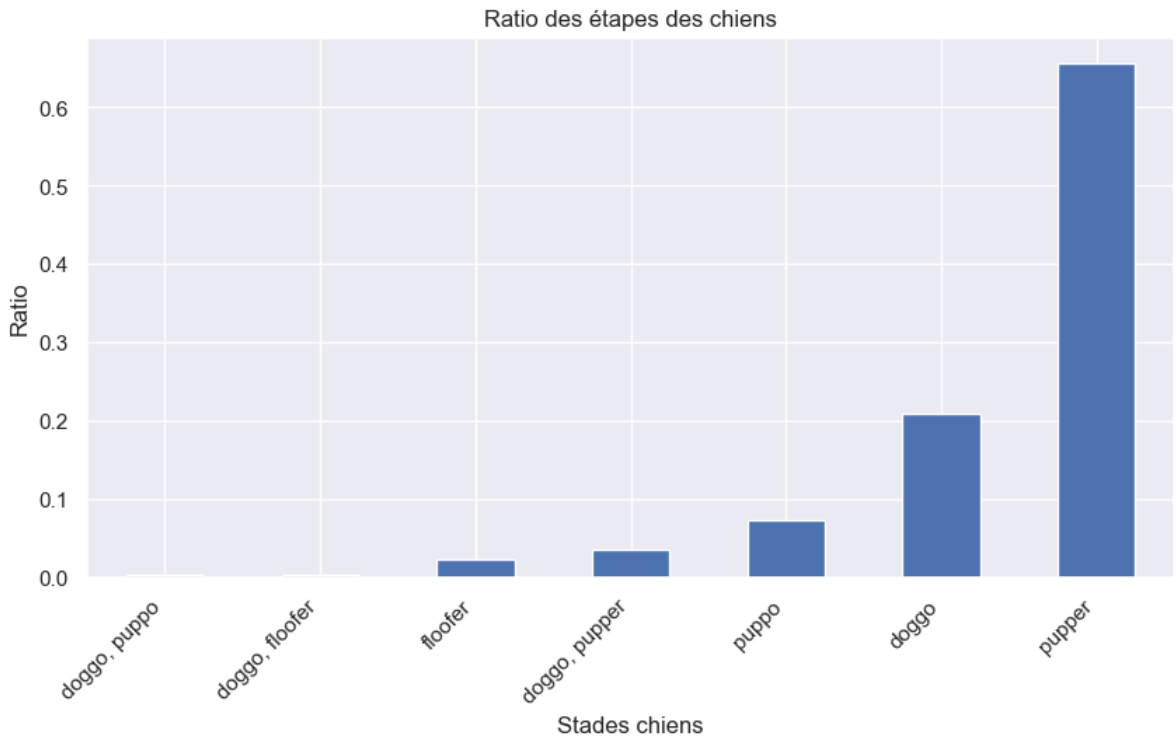
In [690…

```
# filtrer le dataset
race_chien = all_data[all_data.race_chien.notna()]

# Graphe des dix races de chiens les plus fréquents
most_frequent = race_chien.race_chien.value_counts(ascending=True)[-10:]

plt.figure(figsize=(10, 5))
most_frequent.plot(kind='barh')
plt.title("10 races de chiens les plus fréquents")
plt.ylabel("Races de chiens")
plt.xlabel("Nombre")
plt.xticks(rotation=45, horizontalalignment='left');
plt.savefig("img/top_races_chiens")
```



Q#5: Les chiens les plus tweetés sont au niveau *pupper*.

In [691…
```
stades_chien = all_data[all_data.stade_chien.notna()]
stades_chien = stades_chien.stade_chien.value_counts(normalize=True, ascending=Tru

plt.figure(figsize=(10, 5))
stades_chien.plot(kind='bar')
plt.title("Ratio des étapes des chiens")
plt.xlabel("Stades chiens")
plt.ylabel("Ratio")
plt.xticks(rotation=45, horizontalalignment='right');
```



Q#6:

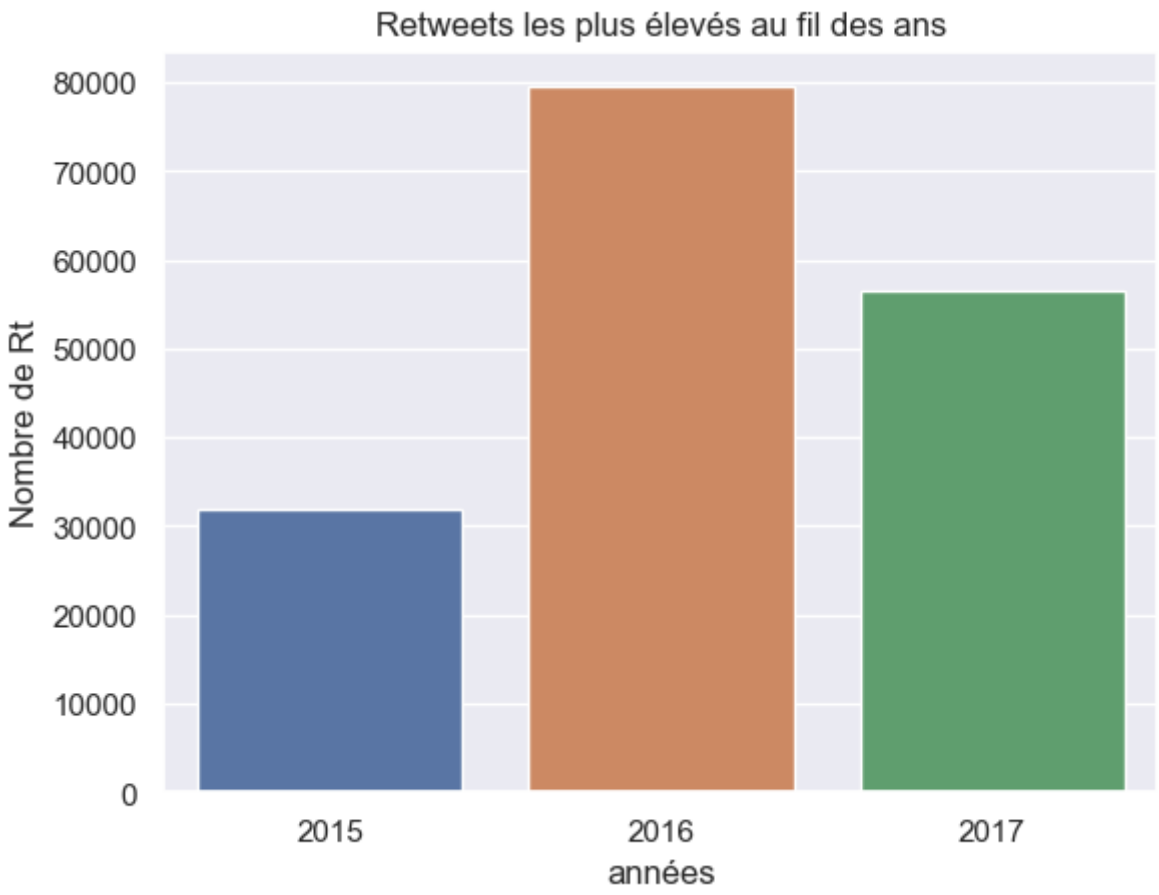
- Le tweet le plus retweeté est celui de l'id *744234799360020481* à la date du *2016-06-18* et concerne un *Labrador_retriever*.
- *2016* est l'année où il y a eu le plus de Retweet (Rt).

In [692…
```
# Tweet le plus retweeté
all_data.loc[all_data.retweet_count.idxmax()]
```

Out[692]:
```
tweet_id                                            744234799360020481
date                                           2016-06-18 18:26:18+00:00
source                                                  Twitter for iPhone
text                  Here's a doggo realizing you can stand in a po...
rating_numerator                                                      13
rating_denominator                                                    10
name                                                                 NaN
stade_chien                                                        doggo
single_rate                                                          1.3
jpg_url               https://pbs.twimg.com/ext_tw_video_thumb/74423...
race_chien                                            Labrador_retriever
retweeted                                                         False
favorited                                                         False
favorite_count                                                   131075
retweet_count                                                     79515
is_quote_status                                                   False
langue                                                          Anglais
Name: 850, dtype: object
```

In [693…
```
# Retweets sur l'année
highest_retweets = group_an.retweet_count.max()
df = pd.DataFrame(highest_retweets)
df.reset_index(inplace=True)
df.rename(columns={'retweet_count':'Nombre de Rt', 'date':'années'}, inplace=True)
sns.barplot(data=df, x="années", y="Nombre de Rt").set(title='Retweets les plus élé
```

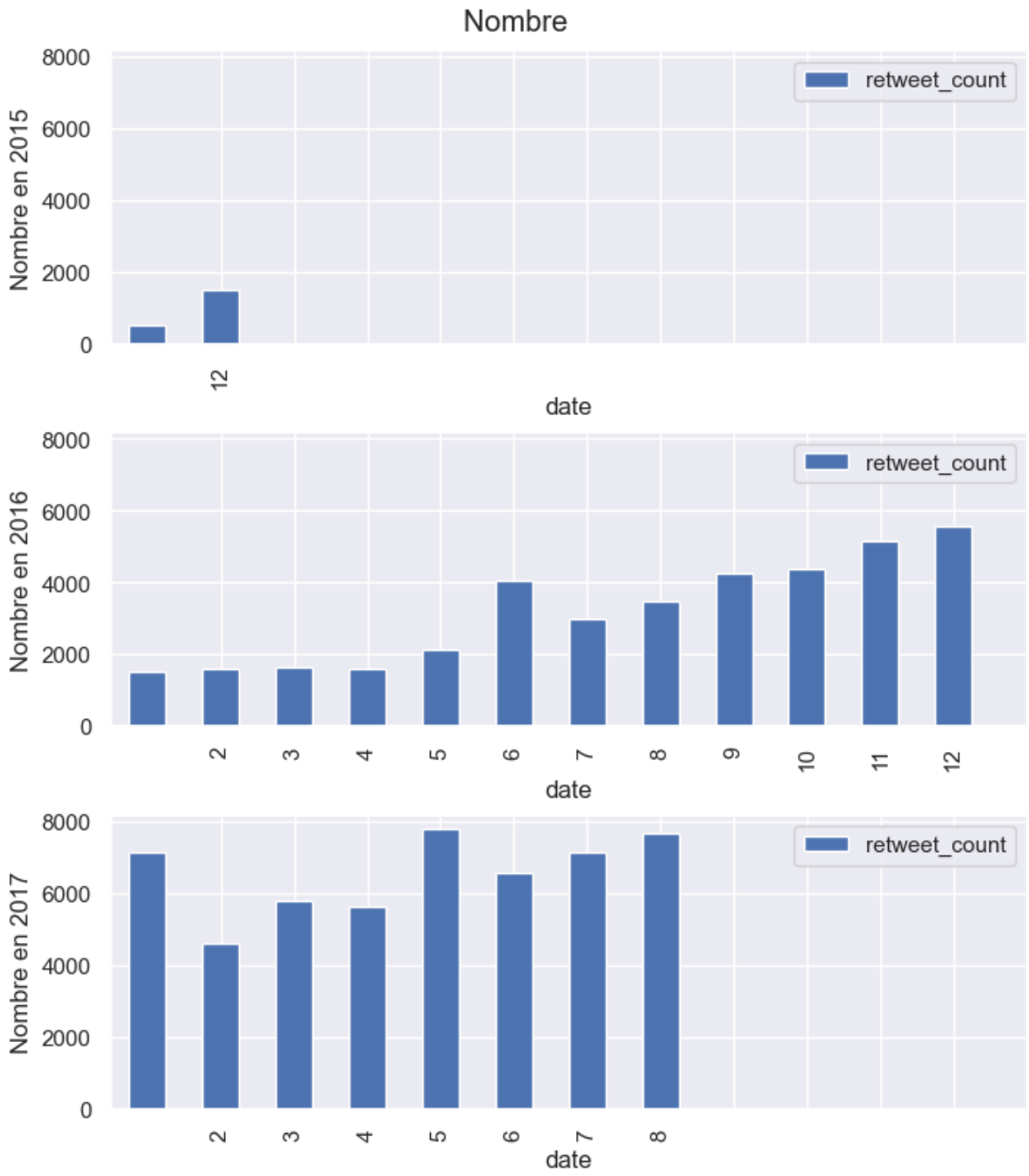
Out[693]:
```
[Text(0.5, 1.0, 'Retweets les plus élevés au fil des ans')]
```



Q#7:

- Le nombre de Retweets (Rt) augmente au fil du temps chaque années, même si *2017* présente une certaine irrégularité.

In [694…
```
years_data = all_data.groupby([all_data.date.dt.year, all_data.date.dt.month]).agg(

# Plotting
fig, axis = plt.subplots(nrows=3, ncols=1, figsize=(7, 8), constrained_layout=True)
years_data.xs(2015).plot(kind='bar', ax=axis[0])
years_data.xs(2016).plot(kind='bar', ax=axis[1])
years_data.xs(2017).plot(kind='bar', ax=axis[2])

axis[0].set_ylabel('Nombre en 2015')
axis[1].set_ylabel('Nombre en 2016')
axis[2].set_ylabel('Nombre en 2017')

plt.setp(axis, xticks=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]);
plt.suptitle("Nombre ");
```



Q#8:

- Chien le mieux aimé est celui illustré dans le tweet ayant pour id 822872901745569793 à la date du 2017-01-21. Il est de race *Lakeland Terrier*, de niveau *pupo* et ce présente sur cette image.
- Voici ce qui a été dit dans ce fameux tweet : "*Here's a super supportive puppo participating in the Toronto #WomensMarch today. 13/10* .".
- Malheureusement on connait pas nom !

In [695…
```
# Chien le mieux aimé
pd.DataFrame(all_data.loc[all_data.favorite_count.idxmax()])
```

Out[695]:

	329
tweet_id	822872901745569793
date	2017-01-21 18:26:02+00:00
source	Twitter for iPhone
text	Here's a super supportive puppo participating ...
rating_numerator	13
rating_denominator	10
name	NaN
stade_chien	puppo
single_rate	1.3
jpg_url	https://pbs.twimg.com/media/C2tugXLXgAArJO4.jpg
race_chien	Lakeland_terrier
retweeted	False
favorited	False
favorite_count	132810
retweet_count	48265
is_quote_status	False
langue	Anglais

In [696…
```
# Contenu du tweet
print("Contenu du tweet : ",all_data.loc[all_data.favorite_count.idxmax()].text)
```

```
Contenu du tweet :  Here's a super supportive puppo participating in the Toronto
#WomensMarch today. 13/10 .
```

Q#9:

- Sur *2071* tweets où les langues ont été reconnues *2065* sont écrites en *Anglais*. Ce qui montre que montre que l'utilisateur du compte est soit originaire d'un pays anglophone soit il veut cibler plus de public avec ses tweets puisque l'anglais est la langue la plus parlé au monde.

In [697…
```
tweet_lang = all_data[all_data.langue.notna()]
tweet_lang = tweet_lang.langue.value_counts(ascending=True)
pd.DataFrame(tweet_lang)
```

Out[697]:

	langue
Basque	1
Roumain	1
Estonien	1
Néerlandais	3
Anglais	2065

Visualization

Bonus (Word Cloud avec les Tweets)

In [698…
```
# D'abord créer une liste avec tous les mots qui ont été tweetés dans notre DataFr
tweets = np.array(all_data.text)
my_list = []
for tweet in tweets:
    my_list.append(tweet.replace("\n",""))
```

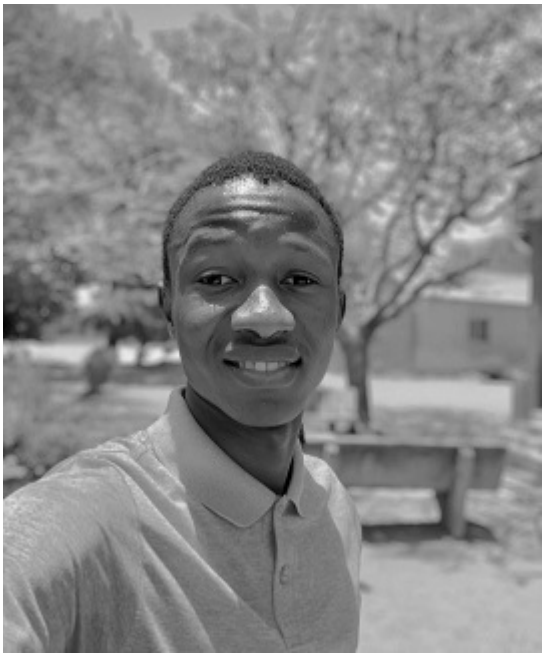
In [699…
```
# Ensuite télécharger une image d'une empreinte de patte sur Internet pour l'utili
# des tweet avec wordcloud.
mask = np.array(Image.open(requests.get('https://clipartix.com/wp-content/uploads/
text = my_list
def gen_wc(text, mask):
    word_cloud = WordCloud(width = 500, height = 500, background_color='white', ma
    plt.figure(figsize=(10,8),facecolor = 'white', edgecolor='red')
    plt.imshow(word_cloud)
    plt.axis('off')
    plt.tight_layout(pad=0)
    plt.show()
```

Ce script utilisé ci-dessus a été inspiré de ce blog à propos de "*Comment générer un nuage de mots de n'importe quelle forme en Python*".

In [700…
```
gen_wc(text, mask)
```

Référence : B.ALEMU

Auteur



Nom : **DIATTA**

Prénom : **Arona Ben Cherif**

Niveau : **Master 2 | Statistiques et Informatique Décisionnelle (UADB)**

Nom de classe : **ARONA_ADND**

Promotion : **Cohorte 2 ALX-Udacity**

Nom de classe : **ARONA_ADND**

Programme : **Nanodegree d'analyste de données**