Introduction

wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information.

Key Points

Key points to keep in mind when data wrangling for this project:

- **You only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.**
- **Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling.**
- **the requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.**
- **Cleaning includes merging individual pieces of data according to the rules of tidy data.**
- **The fact that the rating numerators are greater than the denominators does not need to be cleaned. This unique rating system is a big part of the popularity of WeRateDogs.**

Part one: Data Wrangling:

a- Gathering Data

In [559]:

```
# package to be used in the project...
import pandas as pd
import numpy as np
import requests
import os
from PIL import Image
from io import BytesIO
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer
```

`First` **: Gathering data from "twitter-archive-enhanced.csv" which is download manually :**

`Second` **: Gathering data from "image_predictions.tsv" that will be downloaded programmatically using the Requests library:**

`Third` **: Gathering data from query the Twitter API for each tweet's JSON data using Python's Tweepy library and store each**

tweet's entire set of JSON data in a file called "tweet_json.txt" file:

b- Assessing Data:

`First: Assess Data for twitter_archive table`

`(1)`

View the first rows of the table.

```
twitter_archive.head()
```

View the last 5 rows of the table.

```
twitter_archive.tail()
```

`(2)`

view name of columns,them data type, count of non null values, counts of rows and columns.

```
twitter_archive.info()
```

`(3)`

view the count of each value in the name columns

```
dict(twitter_archive.name.value_counts())
```

`(4)`

view all names which isn't in title case.

- **df = twitter_archive[twitter_archive['name'].str.istitle()==False]**
- **df['name'].value_counts()**

`(5)`

Calculate values counts for doggo, floofer, pupper, and puppo columns.

- ***twitter_archive['doggo'].value_counts()***

```
None     2259
doggo      97
```

- ***twitter_archive['floofer'].value_counts()***

```
None       2346
floofer      10
```

- ***twitter_archive['pupper'].value_counts()***

```
None      2099
pupper     257
```

- ***twitter_archive['puppo'].value_counts()***

```
None     2326
puppo      30
```

`(6)`

View value_counts for source column.

```
twitter_archive.source.value_counts()
```

(7)

print a sample of text column(the first 10 values)

```
list(twitter_archive.text)[0:10]
```

(8)

Print the value counts for rating_numerator.

```
twitter_archive.rating_numerator.value_counts()
```

(9)

Print the unique values for rating_numerator.

```
twitter_archive.rating_numerator.unique()

array([  13,   12,   14,    5,   17,   11,   10,  420,  666,    6,   15,
        182,  960,    0,   75,    7,   84,    9,   24,    8,    1,   27,
          3,    4,  165, 1776,  204,   50,   99,   80,   45,   60,   44,
        143,  121,   20,   26,    2,  144,   88]
```

(10)

Print the value counts for rating_denominator.

```
twitter_archive.rating_denominator.value_counts()
```

(11)

print the count of rating_denominator not equal to 10.

```
Denominator_not_equal_10 = twitter_archive[twitter_archive['rating_denominator']!=1
0]
Denominator_not_equal_10.rating_denominator.size.

Its size = 23
```

(12)

print the unique values of rating_denominator.

```
twitter_archive.rating_denominator.unique()

array([ 10,   0,  15,  70,   7,  11, 150, 170,  20,  50,  90,  80,  40,
       130, 110,  16, 120,   2]
```

(13)

check for duplications

```
sum(twitter_archive.duplicated())

==> No Duplicate for twitter_archive
```

Second: Assess Data for twitter_archive table

(1)

View the first 5 rows of image_prediction table.

```
image_prediction.head()
```

(2)

Print image_prediction columns name, them data type, and count of rows and columns

```
image_prediction.info()
```

(3)

Check for duplicated values.

```
sum(image_prediction.duplicated())

==> Output = 0
```

(4)

Print the sum of duplicated value in jpg_url column.

```
sum(image_prediction['jpg_url'].duplicated())

==> Output = 66
```

(5)

Print value coumts for dog type of first prediction.

```
image_prediction.p1.value_counts().head(10)
```

(6)

print value coumts for dog type of second prediction.

```
image_prediction.p2.value_counts().head(10)
```

`(7)`

Print value coumts for dog type of third prediction.

```
image_prediction.p3.value_counts().head(10)
```

`Third: Assess Data for tweet_data table`

`(1)`

Print a sample of tweet_data table

```
tweet_data.sample(3)
```

`(2)`

Print column names and shape of tweet_data table

```
tweet_data.columns, tweet_data.shape

==> Output: print all columns name, its shape (2331, 32)
```

`(3)`

I need only three columns from tweet_data which are ('id','retweet_count', and 'favorite_count')

```
tweets_data=tweet_data.loc[ : ,['id','favorite_count','retweet_count']]
```

`(4)`

Print some statistics of tweet_data

```
tweets_data.describe()
```

`(5)`

Print tweet data column names , datatype and count of rows and columns

```
tweets_data.info()
```

`(6)`

Count of rows which contain zero value for the favorite_count column

```
print('count of rows which contain zero value for the favorite_count column: ',\
  tweets_data[tweets_data['favorite_count']==0].count()[1])

  ==> Output: 163
```

(7)

Count of rows which contain zero value for the retweet_count column

```
print('count of rows which contain zero value for the retweet_count column: ',\
  tweets_data[tweets_data['retweet_count']==0].count()[1])

  ==> Output: 0
```

(8)

check for duplicated values

```
sum(tweets_data.duplicated())

==> Output: 0
```

Quality

`twitter_archive` table

- **tweet_id is int instead of being a string.**
- **Each of(in_reply_to_status_id , and in_reply_to_user_id, retweeted_status_id,in_reply_to_status_id) is float not a string.**
- **"in_reply_to_status_id" and "in_reply_to_user_id" have only 78 non null values.**
- **"retweeted_status_id" and "retweeted_status_user_id" have only 181 non null values.**
- **Nulls represented as 'None' in each of doggo,floofer,pupper, and puppo.**
- **doggos has 2259 value equal to 'None'.**
- **floofer has 2346 value equal to 'None'.**
- **pupper has 2099 value equal to 'None'.**
- **puppo has 2326 value equal to 'None'.**
- **expanded_url column has 59 missing values.**
- **Nulls represented as 'None' in name column.**
- **lowercase names have unsuitable data like (all, the , a , an , by , old, not, his, this.....)**
- **Capitalize the rest of lowercase.**
- **name ='O' has to be replaced with Nan.**
- **some of rating_numerators and rating_denominators values are incorrect.**
- **rating_numerators and rating_denominators is int instead of float.**
- **timestamp is object instead of being datetime type.**

`image_prediction` table

- **tweet_id is int not a string.**
- **jpg_url column has duplicated photos (66 row are duplicated) have to be removed**
- **it includes 2074 row, there are some missing rows comparing to `twitter_archive` rows'counts(2356 row).**
- **rows with p1_dog, p2_dog and p3_dog with False value should be removed**

`tweet_data` table

- **id is int not a string.**
- **the 'id' column should be renamed to "tweet_id" to match the other 2 tables.**
- **it includes 2331 row, there are missing rows comparing to `twitter_archive` rows'counts (2356 row).**

Tidiness

`twitter_archive` table

- **Combine (Doggo, floofer, pupper, puppo) in one column named dog_type.**
- **Dropping uneeded columuns (in_reply_to_status_id' , 'in_reply_to_user_id','retweeted_status_id','retweeted_status_user_id','expanded_urls','source')**
- **Combine rating_numerator and rating_denominator one column in twitter_archive_clean.**

`image_prediction` table

- **The first True prediction for all image will be saved to new column (dog_type)with its confidence in another column named (confidence_degree).**
- **Dropping uneeded columns.**
-
  - **Combine `image_prediction` with `twitter_archive` using tweet_id.**

`tweets_data` table

- **Combine `tweets_data` with `twitter_archive` using tweet_id.**

c- Cleaning Data:

In [377]:

```
# make a copy of the original data frames.

twitter_archive_clean= twitter_archive.copy()
image_prediction_clean= image_prediction.copy()
tweet_data_clean=tweets_data.copy()
```

Detect and document at least eight (8) quality issues and two (2) tidiness issues

Quality Issues:

`twitter_archive` table :

- **change tweet_id data type to string.**
- **change rating_denominator, rating_nominator data type to float**
- **Replace all None name with Nan.**
- **Replace all 'None' values with Nan in each of doggo,floofer,pupper, and puppo**
- **Convert timestamp data type from object datetime.**
- **Fill unsuitable lowercase names with Nan.**
- **Replace name="O" with nan.**
- **Correct wrong values for denomirator_rate and numerator_rate.**
  - **=> Correct wrong values for denomirator_rate**
  - **=> Define function (view_rating) to view incorrect rates.**
  - **=> Define function (update_rating) to correct numinator and denominator rating for all rows that have rating_denominator not equal to 10.**
  - **=> Correct numerator and denomirator for all rows which rating_denomirator not equal 10.**
  - **=> Drop unrated rows.**

Finally we check the value_counts for rating_denominator equal to 10.0:

```
==> Output: all rows have value 10 for rating_denomirator
```

- **Correct the rest values of rating numerator.**
  - **=> View all text for all rows with rating_numerator greater or equal to 20**
  - **=> Update incorrect rating_numerator**
  - **=> Drop rows with wrong rating_numerator.**
  - **=> Drop rows with rating_numerator=0.**

Finally, check for all values for rating_numenator.

```
        ==> Output:
                    array([13.   , 12.   , 14.   ,   5.   , 17.   , 11.   , 10.   ,   6.   , 15
   .   ,
                           9.75,   7.   ,   9.   ,   8.   ,   1.   , 11.27,   3.   ,   4.   ,   2.
   ,11.26])
```

`image_prediction` ***table***

- **Change tweet_id data type to string.**
- **Drop duplicated rows where jpg_url values are duplicated (there are 66 duplicated image)**

```
==> Output: shape of image_prediction after deleting duplicates (2009, 12)
```

- **Drop rows with p1_dog, p2_dog and p3_dog with False**

```
==> Count of all rows after deleting all rows with false prediction for p1_dog,
p2_dog and p3_dog is:  1691
```

`tweet_data` ***table***

- **Change tweet_id data type to string.**
- **Rename (id) column to (tweet_id) to match the other tables**

Tidiness Issues:

`twitter_archive` ***table***

- **Drop unneeded columns which are columns relevant to retweets and replies, because we only want original tweets with images and most of values for these columns are Nan.**
- **Combine (Doggo, floofer, pupper, puppo) in one column named dog_type using melt function to merge these four columns in one columns named "dogs_stage"**

```
  ==> Output:Check for count values for dog stage in the new column dogs_stage
  pupper     245
  doggo       97
  puppo       29
  floofer      9
```

- **Create new column 'rate'.**

```
 =>Round up the rate value and convert it to integer.
 => Check for value counts in new column rate.
```

- **drop (text, rating_numerator, and rating_denomirator) from twitter_archive_clean.**

`image_prediction` ***table***

- **Create Two new column:**

```
One -->to store The first True prediction for image.
The other --> to store its confidence.
```

- **drop unneeded columns from image_prediction_clean.**

Finally

Finally

- **Combine (image_prediction_clean) and (tweets_data_clean) with (twitter_archive_clean) using 'tweet_id' in one table (df_master).**
- **check for all columns in the new table df_master.**

```
df_master.info()
==> Output:
1682 entries, 0 to 1681
 Data columns (total 11 columns):
 tweet_id            1682 non-null object
 timestamp           1682 non-null datetime64[ns]
 name                1178 non-null object
 dogs_stage          260 non-null object
 rate                1682 non-null int32
 jpg_url             1682 non-null object
 img_num             1682 non-null int64
 dog_type            1682 non-null object
 confidence_grade    1682 non-null float64
 favorite_count      1682 non-null int64
 retweet_count       1682 non-null int64
 dtypes: datetime64[ns](1), float64(1), int32(1), int64(3), object(5)
```

Saving df_master to twitter_archive_master.csv:

df_master.to_csv('twitter_archive_master.csv', index=False, encoding = 'utf-8')