In this project, the dataset that is undergoing data wrangling process is the tweet archive of Twitter user **"@dog_rates"** also known as **WeRateDogs.** WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc.

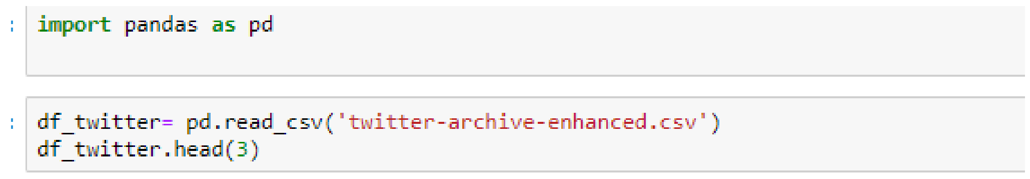
Data wrangling is the process of gathering data from multiple sources, assessing and cleaning the said data in order to get insights out of it. To dive into it, I will explain each step of the wrangling process below:

Step 1: Data Gathering

There were three datasets that were supposed to be gathered,

a. Twitter-archive-enhanced.csv

This dataset was already provided for us in the Udacity classroom. WeRateDogs downloaded their Twitter archive and sent it to Udacity via email exclusively for to be used in this project. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017. I downloaded it and then read it using pandas. To read the data in pandas, I imported pandas as pd in my notebook and then read the csv file.

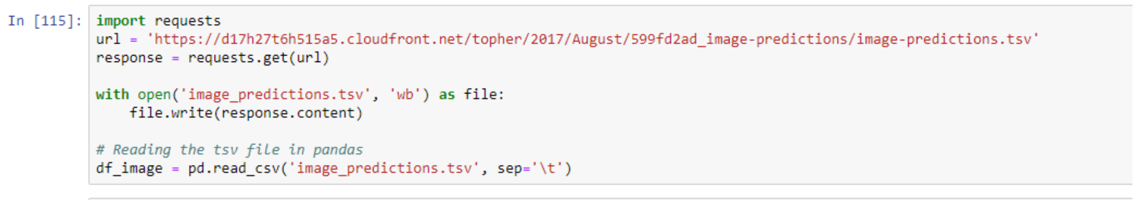


```
: import pandas as pd
```

```
: df_twitter= pd.read_csv('twitter-archive-enhanced.csv')
  df_twitter.head(3)
```

b. Image-predictions.tsv

This file (image_predictions.tsv) is present in each tweet according to a neural network. There was a URL link provided in the Udacity classroom https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv and I used the requests library (*Requests will allow you to send HTTP/1.1 requests using Python)* to download the tweet image prediction as shown in the image below.



```
In [115]: import requests
          url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
          response = requests.get(url)

          with open('image_predictions.tsv', 'wb') as file:
              file.write(response.content)

          # Reading the tsv file in pandas
          df_image = pd.read_csv('image_predictions.tsv', sep='\t')
```

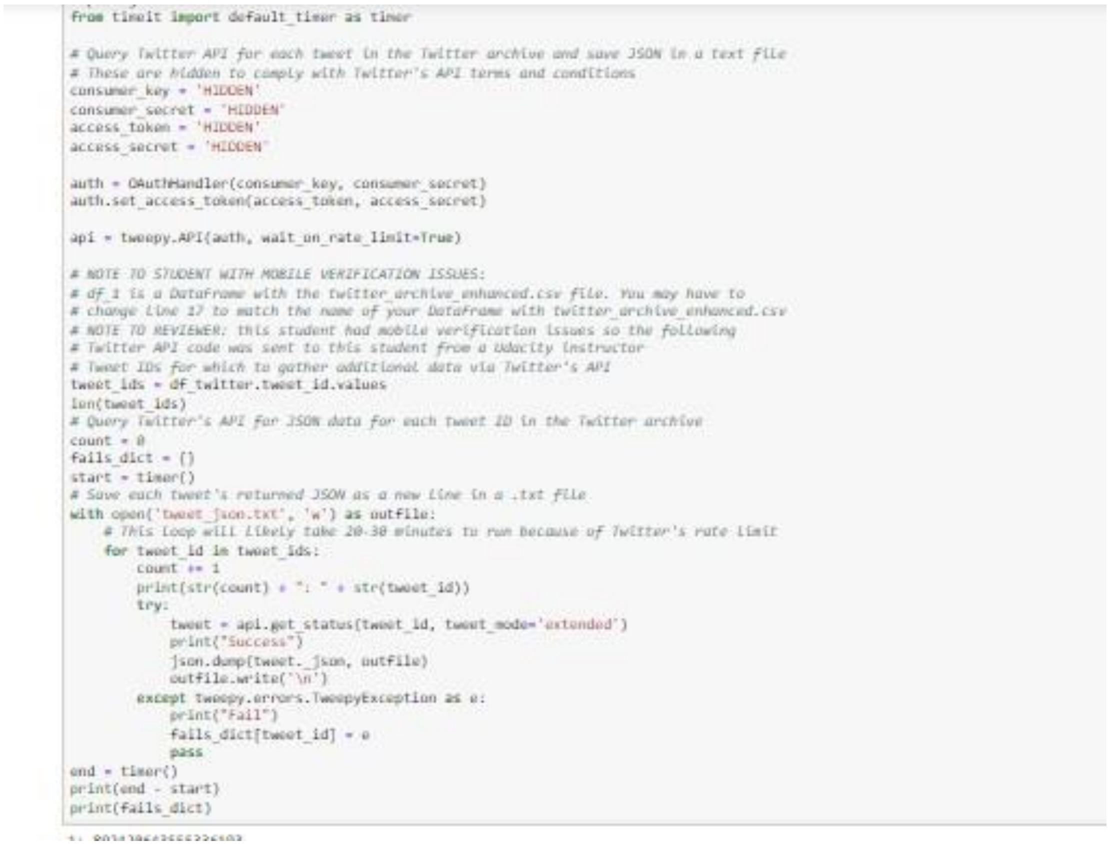
Next, since I'm interested in only getting "image-predictions.tsv', I used the with open method and 'wb' which means "write in binary' to open the file.

After opening the file, I read the file in pandas and since it was a tsv file, I had to include theseparation as “\t” for the file to be read successfully.

```
#reading the file in pandas
df_image = pd.read_csv("image-predictions.tsv", sep='\t')
df_image.head()
```

c. tweet_json.txt

This is an additional data obtained from the twitter API. I used the Tweepy library provided in the classroom and queried each tweet’s retweet count and favorite(“like”) count. I did not get access from Twitter, so, I did not run that cell as the output brought out a lot of fails rather than success.



```
from timeit import default_timer as timer

# Query Twitter API for each tweet in the Twitter archive and save JSON in a text file
# These are hidden to comply with Twitter's API terms and conditions
consumer_key = 'HIDDEN'
consumer_secret = 'HIDDEN'
access_token = 'HIDDEN'
access_secret = 'HIDDEN'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth, wait_on_rate_limit=True)

# NOTE TO STUDENT WITH MOBILE VERIFICATION ISSUES:
# df_1 is a DataFrame with the twitter_archive_enhanced.csv file. You may have to
# change line 17 to match the name of your DataFrame with twitter_archive_enhanced.csv
# NOTE TO REVIEWER: this student had mobile verification issues so the following
# Twitter API code was sent to this student from a Udacity instructor
# Tweet IDs for which to gather additional data via Twitter's API
tweet_ids = df_twitter.tweet_id.values
len(tweet_ids)
# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
count = 0
fails_dict = {}
start = timer()
# Save each tweet's returned JSON as a new line in a .txt file
with open('tweet_json.txt', 'w') as outfile:
    # This loop will likely take 20-30 minutes to run because of Twitter's rate limit
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Success")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except tweepy.errors.TweepyException as e:
            print("Fail")
            fails_dict[tweet_id] = e
            pass
end = timer()
print(end - start)
print(fails_dict)
```

After querying the Twitter API, each tweet’s entire set of JSON data is stored as “tweet_json.txt” file. When I ran my codes, I got a lot of fails more than what is expected, so I had to use the json file provided in the classroom.
The file led to a link 'https://video.udacity-data.com/topher/2018/November/5be5fb7d_tweet-json/tweet-json.txt' so I opened it programmatically using requests and later read the file in pandas.

```
In [117]: url= 'https://video.udacity-data.com/topher/2018/November/5be5fb7d_tweet-json/tweet-json.txt'
          response = requests.get(url)

          with open('tweet-json.txt', 'wb') as file:
              file.write(response.content)

          #Reading the Json file into a pd dataframe
          df_json = pd.read_json('tweet-json.txt', lines='True', orient = 'records')[:-1]
          df_json.head()
```

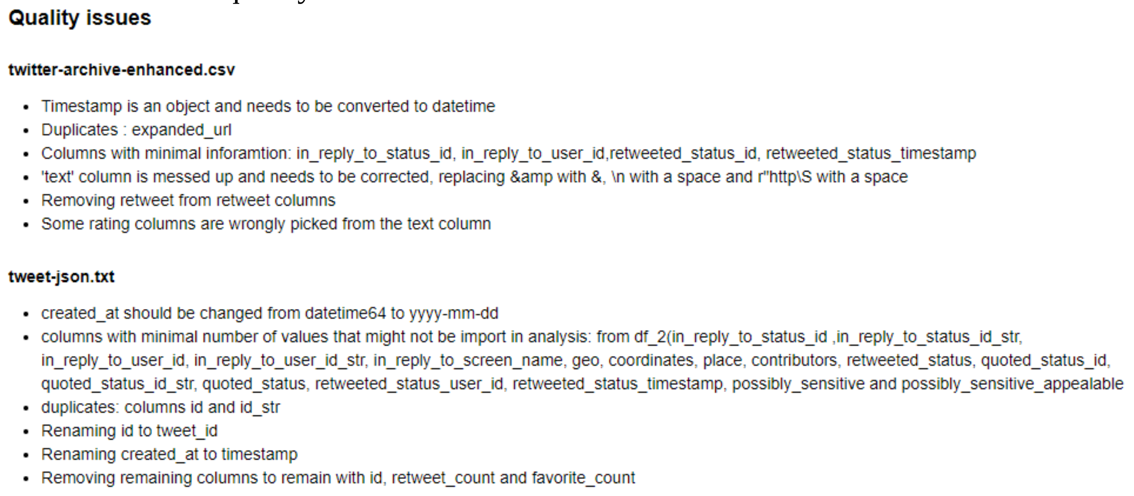
Step 2: Assessing Data

After gathering the three datasets and reading them in pandas, the next step is to assess the data.

There are two methods used for assessing: visually and programmatically. Visual assessment involves use of Excel, google sheets, etc and programmatic assessment involves use of codes such as .info( ) etc.

The dimensions for data quality are: completeness, validity, accuracy and consistency while the dimension for tidiness are that, each variable forms a column, each observation forms a row and each type of observational unit forms a table.

I identified 12 quality issues and 2 tidiness issues.



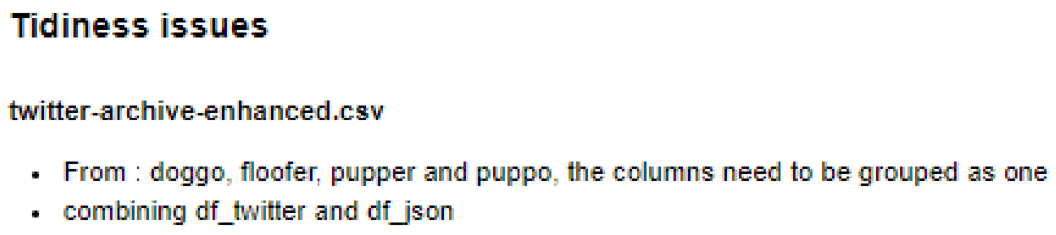
Quality issues

twitter-archive-enhanced.csv

- Timestamp is an object and needs to be converted to datetime
- Duplicates : expanded_url
- Columns with minimal inforamtion: in_reply_to_status_id, in_reply_to_user_id,retweeted_status_id, retweeted_status_timestamp
- 'text' column is messed up and needs to be corrected, replacing &amp with &, \n with a space and r"http\S with a space
- Removing retweet from retweet columns
- Some rating columns are wrongly picked from the text column

tweet-json.txt

- created_at should be changed from datetime64 to yyyy-mm-dd
- columns with minimal number of values that might not be import in analysis: from df_2(in_reply_to_status_id ,in_reply_to_status_id_str, in_reply_to_user_id, in_reply_to_user_id_str, in_reply_to_screen_name, geo, coordinates, place, contributors, retweeted_status, quoted_status_id, quoted_status_id_str, quoted_status, retweeted_status_user_id, retweeted_status_timestamp, possibly_sensitive and possibly_sensitive_appealable
- duplicates: columns id and id_str
- Renaming id to tweet_id
- Renaming created_at to timestamp
- Removing remaining columns to remain with id, retweet_count and favorite_count



Tidiness issues

twitter-archive-enhanced.csv

- From : doggo, floofer, pupper and puppo, the columns need to be grouped as one
- combining df_twitter and df_json

Step 3: Cleaning data

From the quality and tidiness issues that I identified, I cleaned all of them using codes because codes are meant to make work easier unlike cleaning manually. I did the cleaning process in such a way that important information wouldn’t be lost in the process.

Step 4: Step 4: Storing data

I saved gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".