wrangle_act

June 28, 2022

0.1 import the required libraries.

[1]:
```
import configparser
import pandas as pd
import numpy as np
import requests
import tweepy
import json
import os
import re

from bs4 import BeautifulSoup
```

1 Data Gathering

1.1 Load the `twitter archive enhanced csv` file into the pandas dataframe

[2]:
```
Dog_tweets_df = pd.read_csv('Datasets/twitter-archive-enhanced.csv')
```

1.2 Download the tweet image predictions

[3]:
```
# # Check the if the required folder exists. If it does not exist, create it
# folder_name = 'Datasets'
# if not os.path.exists (folder_name):
#     os.makedirs(folder_name)

# # Use the request.get method to get access the data from the server.
# url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/
 ↪599fd2ad_image-predictions/image-predictions.tsv'
# response = requests.get(url)

# # Save the responses in a .tsv file
# with open (os.path.join(folder_name,url.split('/')[-1]), mode = 'wb') as file:
#     file.write(response.content)
```

1.2.1 Load the Image prediction file into the pandas dataframe

[4]:
```
image_prediction_df = pd.read_csv('Datasets/image-predictions.tsv', sep='\t')
```

1.3 Get additional data from the Twitter API

[5]:
```
# Read the configurations
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']
```

[6]:
```
# Authentication
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)

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

[7]:
```
# # Extract all the tweet IDs in the WeRateDogs Twitter archive and save them␣
 ↪in a python list.
# tweet_response_counts = []
# for tweet_id in Dog_tweets_df.tweet_id:
#     try:
#         tweet_info = api.get_status(tweet_id, tweet_mode='extended')
#     except Exception:
#         pass
#     finally:
#         tweet_id = tweet_info.id
#         retweet_count = tweet_info.retweet_count
#         favorite_count = tweet_info.favorite_count

#         tweet_response_counts.append({'tweet_id':tweet_id,
#                                       'retweet_count':retweet_count,
#                                       'favorite_count':favorite_count})
#         print(tweet_id)
# counts_df = pd.DataFrame(tweet_response_counts, columns = ['tweet_id',␣
 ↪'retweet_count', 'favorite_count'])
```

[8]:
```
# 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.TweepError as e:
#             print("Fail")
#             fails_dict[tweet_id] = e
#             pass
# end = timer()
# print(end - start)
# print(fails_dict)
```

[9]:
```
# counts_df.to_csv('Datasets/reaction_counts.csv', index=False)
```

1.3.1 Load the dataframe into the pandas dataframe

[10]:
```
reaction_counts_df = pd.read_csv('Datasets/reaction_counts.csv')
```

2 Assessing Data

2.1 1. Assessing the Dog_tweet dataframe

[11]:
```
# Display the dataframe
Dog_tweets_df
```

[11]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000
1	892177421306343426	NaN	NaN	
2	891815181378084864	NaN	NaN	
3	891689557279858688	NaN	NaN	
4	891327558926688256	NaN	NaN	
...	...	...	...	
2351	666049248165822465	NaN	NaN	
2352	666044226329800704	NaN	NaN	
2353	666033412701032449	NaN	NaN	
2354	666029285002620928	NaN	NaN	
2355	666020888022790149	NaN	NaN	

```
1     2017-08-01 00:17:27 +0000
2     2017-07-31 00:18:03 +0000
3     2017-07-30 15:58:51 +0000
4     2017-07-29 16:00:24 +0000
...                         ...
2351  2015-11-16 00:24:50 +0000
2352  2015-11-16 00:04:52 +0000
2353  2015-11-15 23:21:54 +0000
2354  2015-11-15 23:05:30 +0000
2355  2015-11-15 22:32:08 +0000

                                                 source  \
0     <a href="http://twitter.com/download/iphone" r...
1     <a href="http://twitter.com/download/iphone" r...
2     <a href="http://twitter.com/download/iphone" r...
3     <a href="http://twitter.com/download/iphone" r...
4     <a href="http://twitter.com/download/iphone" r...
...                                                 ...
2351  <a href="http://twitter.com/download/iphone" r...
2352  <a href="http://twitter.com/download/iphone" r...
2353  <a href="http://twitter.com/download/iphone" r...
2354  <a href="http://twitter.com/download/iphone" r...
2355  <a href="http://twitter.com/download/iphone" r...

                                                   text  retweeted_status_id  \
0     This is Phineas. He's a mystical boy. Only eve...                  NaN
1     This is Tilly. She's just checking pup on you...                  NaN
2     This is Archie. He is a rare Norwegian Pouncin...                  NaN
3     This is Darla. She commenced a snooze mid meal...                  NaN
4     This is Franklin. He would like you to stop ca...                  NaN
...                                                 ...                  ...
2351  Here we have a 1949 1st generation vulpix. Enj...                  NaN
2352  This is a purebred Piers Morgan. Loves to Netf...                  NaN
2353  Here is a very happy pup. Big fan of well-main...                  NaN
2354  This is a western brown Mitsubishi terrier. Up...                  NaN
2355  Here we have a Japanese Irish Setter. Lost eye...                  NaN

      retweeted_status_user_id retweeted_status_timestamp  \
0                          NaN                         NaN
1                          NaN                         NaN
2                          NaN                         NaN
3                          NaN                         NaN
4                          NaN                         NaN
...                        ...                         ...
2351                       NaN                         NaN
2352                       NaN                         NaN
2353                       NaN                         NaN
```

```
2354                        NaN                         NaN
2355                        NaN                         NaN

                                              expanded_urls  rating_numerator  \
0     https://twitter.com/dog_rates/status/892420643...                 13
1     https://twitter.com/dog_rates/status/892177421...                 13
2     https://twitter.com/dog_rates/status/891815181...                 12
3     https://twitter.com/dog_rates/status/891689557...                 13
4     https://twitter.com/dog_rates/status/891327558...                 12
...                                                 ...                ...
2351  https://twitter.com/dog_rates/status/666049248...                  5
2352  https://twitter.com/dog_rates/status/666044226...                  6
2353  https://twitter.com/dog_rates/status/666033412...                  9
2354  https://twitter.com/dog_rates/status/666029285...                  7
2355  https://twitter.com/dog_rates/status/666020888...                  8

      rating_denominator      name doggo floofer pupper puppo
0                     10   Phineas  None    None   None  None
1                     10     Tilly  None    None   None  None
2                     10    Archie  None    None   None  None
3                     10     Darla  None    None   None  None
4                     10  Franklin  None    None   None  None
...                  ...       ...   ...     ...    ...   ...
2351                  10      None  None    None   None  None
2352                  10         a  None    None   None  None
2353                  10         a  None    None   None  None
2354                  10         a  None    None   None  None
2355                  10      None  None    None   None  None

[2356 rows x 17 columns]
```

[12]:
```
Dog_tweets_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
```

[13]:
```
# Check whether there are abnormal values in the rating numerator column
Dog_tweets_df.rating_numerator.describe()
```

[13]:
```
count    2356.000000
mean       13.126486
std        45.876648
min         0.000000
25%        10.000000
50%        11.000000
75%        12.000000
max      1776.000000
Name: rating_numerator, dtype: float64
```

Note that the maximum numerator rating is abnormally high. Let use view that particular record to gain more insight about it.

[14]:
```
Dog_tweets_df.query('rating_numerator == 1776.000000')
```

[14]:
```
               tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
979  749981277374128128                    NaN                  NaN

                    timestamp  \
979  2016-07-04 15:00:45 +0000

                                                source  \
979  <a href="https://about.twitter.com/products/tw...

                                                  text  retweeted_status_id  \
979  This is Atticus. He's quite simply America af...                  NaN

     retweeted_status_user_id retweeted_status_timestamp  \
979                       NaN                        NaN

                                       expanded_urls  rating_numerator  \
979  https://twitter.com/dog_rates/status/749981277...              1776

     rating_denominator     name doggo floofer pupper puppo
979                  10  Atticus  None    None   None  None
```

It appears that the entry is not a problem. Perhaps, the person who rated the dog liked it that much.

[15]:
```
# Check whether there are abnormal values in the rating denominator column
Dog_tweets_df.rating_denominator.describe()
```

[15]:
```
count    2356.000000
mean       10.455433
std         6.745237
min         0.000000
25%        10.000000
50%        10.000000
75%        10.000000
max       170.000000
Name: rating_denominator, dtype: float64
```

[16]:
```
Dog_tweets_df.duplicated().sum()
```

[16]:
```
0
```

[17]:
```
Dog_tweets_df.isnull().sum()
```

[17]:
```
tweet_id                      0
in_reply_to_status_id      2278
in_reply_to_user_id        2278
timestamp                     0
source                        0
text                          0
retweeted_status_id        2175
retweeted_status_user_id   2175
retweeted_status_timestamp 2175
expanded_urls                59
rating_numerator              0
rating_denominator            0
name                          0
doggo                         0
floofer                       0
pupper                        0
puppo                         0
dtype: int64
```

2.2 2. Assessing the Image Predictions Dataframe

[18]:
```
# Display the dataframe
image_prediction_df
```

[18]:
```
             tweet_id                                       jpg_url  \
0  666020888022790149  https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg
1  666029285002620928  https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg
2  666033412701032449  https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg
```

```
3     666044226329800704  https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg
4     666049248165822465  https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg
...                  ...                                                ...
2070  891327558926688256  https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg
2071  891689557279858688  https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg
2072  891815181378084864  https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg
2073  892177421306343426  https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg
2074  892420643555336193  https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg

      img_num                     p1   p1_conf  p1_dog                 p2  \
0           1  Welsh_springer_spaniel  0.465074    True             collie
1           1                redbone  0.506826    True  miniature_pinscher
2           1        German_shepherd  0.596461    True            malinois
3           1    Rhodesian_ridgeback  0.408143    True             redbone
4           1     miniature_pinscher  0.560311    True          Rottweiler
...       ...                    ...       ...     ...                 ...
2070        2                 basset  0.555712    True    English_springer
2071        1            paper_towel  0.170278   False  Labrador_retriever
2072        1              Chihuahua  0.716012    True             malamute
2073        1              Chihuahua  0.323581    True             Pekinese
2074        1                 orange  0.097049   False                bagel

       p2_conf  p2_dog                           p3   p3_conf  p3_dog
0     0.156665    True             Shetland_sheepdog  0.061428    True
1     0.074192    True           Rhodesian_ridgeback  0.072010    True
2     0.138584    True                    bloodhound  0.116197    True
3     0.360687    True            miniature_pinscher  0.222752    True
4     0.243682    True                      Doberman  0.154629    True
...        ...     ...                           ...       ...     ...
2070  0.225770    True  German_short-haired_pointer  0.175219    True
2071  0.168086    True                       spatula  0.040836   False
2072  0.078253    True                        kelpie  0.031379    True
2073  0.090647    True                      papillon  0.068957    True
2074  0.085851   False                        banana  0.076110   False

[2075 rows x 12 columns]
```

[19]:
```
# Check the datatypes of every column an whether any column has null entries
image_prediction_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
```

[20]:
```
# Check for duplicates
image_prediction_df.duplicated().sum()
```

[20]:
```
0
```

[21]:
```
# Check if the dataframe has abnormal numerical values.
image_prediction_df.describe()
```

[21]:

	tweet_id	img_num	p1_conf	p2_conf	p3_conf
count	2.075000e+03	2075.000000	2075.000000	2.075000e+03	2.075000e+03
mean	7.384514e+17	1.203855	0.594548	1.345886e-01	6.032417e-02
std	6.785203e+16	0.561875	0.271174	1.006657e-01	5.090593e-02
min	6.660209e+17	1.000000	0.044333	1.011300e-08	1.740170e-10
25%	6.764835e+17	1.000000	0.364412	5.388625e-02	1.622240e-02
50%	7.119988e+17	1.000000	0.588230	1.181810e-01	4.944380e-02
75%	7.932034e+17	1.000000	0.843855	1.955655e-01	9.180755e-02
max	8.924206e+17	4.000000	1.000000	4.880140e-01	2.734190e-01

2.3 3. Assessing the `likes and retweet` dataframe

[22]:
```
# Display the dataframe
reaction_counts_df
```

[22]:

	tweet_id	retweet_count	favorite_count
0	892420643555336193	7009	33815
1	892177421306343426	5302	29320
2	891815181378084864	3481	22050
3	891689557279858688	7219	36903
4	891327558926688256	7762	35311
...	...	...	...
2351	666049248165822465	37	89
2352	666044226329800704	115	247
2353	666033412701032449	36	100
2354	666029285002620928	39	112
2355	666020888022790149	423	2293

[2356 rows x 3 columns]

[23]:
```
# Check for null entries in the dataframe and the datatype of every column.
reaction_counts_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2356 non-null   int64
 1   retweet_count   2356 non-null   int64
 2   favorite_count  2356 non-null   int64
dtypes: int64(3)
memory usage: 55.3 KB
```

[24]:
```
# Check for abnormal numerical values.
reaction_counts_df.describe()
```

[24]:

	tweet_id	retweet_count	favorite_count
count	2.356000e+03	2356.000000	2356.00000
mean	7.427737e+17	2492.997878	7157.22708
std	6.857010e+16	4193.078114	11028.70282
min	6.660209e+17	1.000000	0.00000
25%	6.783989e+17	500.750000	1243.00000
50%	7.196279e+17	1186.000000	3116.00000
75%	7.993373e+17	2879.000000	8857.50000
max	8.924206e+17	70758.000000	144922.00000

3 Assessment Report

3.0.1 Quality Issues

Dog_tweet dataframe

1. The `in_reply_to_status_id` column has 2278 null entries
2. The `in_reply_to_user_id` column has 2278 null entries
3. The `retweeted_status_id` column has 2175 null entries
4. The `retweeted_status_user_id` column has 2175 null entries
5. The `retweeted_status_timestamp` column has 2175 null entries
6. The `expanded_urls` column has 59 null entries
7. Some of the entries in the `expanded_urls` column contains double entries that are duplicates.
8. `source column` contain a tags (<a/>).
9. Erroneous datatypes (timestamp, retweeted_status_timestamp columns, rating_denominator, rating_numerator columns).

image prediction dataframe

1. This dataframe has non-descriptive column names.

2. Comparing the number of entries in the image prediction dataframe and the dog tweets dataframe, it is clear that some of the tweets have no corresponding image predictions in the image prediction dataframe.

Favourite counts & retweet counts dataframe

- No issues detected

3.0.2 Tidiness Issues

Dog_tweet dataframe

1. `timestamp column` has two variable entries (Date & Time).
2. `retweeted_status_timestamp` has two variable entries (Date & Time).
3. `source column` has two variables (source name & source url).
4. `text column` has four variables in the same column (The tweet, rating_numerator, rating_denominator, Tweet link). **note: the tweet link in this column is similar to the link in the source column**
5. The `doggo`, `floofer`, `pupper`, `puppo` should be variables.

image prediction dataframe

- No isses detected

Favourite counts & retweet counts dataframe

1. We do not need this table because the information it holds (favourite counts and retweet counts) should appear alongside each respective tweet in the dog tweet dataframe.

4 Data Cleaning

[25]:
```
# Make copies of the datasets
tweet_clean = Dog_tweets_df.copy()
image_prediction_clean = image_prediction_df.copy()
reaction_count_clean = reaction_counts_df.copy()
```

4.1 Quality issues

4.2 `Dog Tweet dataframe`

4.2.1 Dealing with missing data

Define:

- The dog_tweet dataframe: The `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id`, `retweeted_status_user_id`, and `retweeted_status_timestamp` have many missing records, all of which we have no means to recover. Since we will not be needing then during analysis, we can drop them off.
- For the `expanded_urls` column, only few entries (59) are missing. However, we have no means of finding the data. So, we can hold it as it is for now.

code

[26]:
```
tweet_clean.drop(['in_reply_to_status_id',
                  'in_reply_to_user_id',
                  'retweeted_status_id',
                  'retweeted_status_user_id',
                  'retweeted_status_timestamp'],
                 axis=1,
                 inplace=True)
```

Test

[27]:
```
# If the columns were dropped successfully, the assert statement will run␣
 ↪without an assertion error.
dropped_columns = ['in_reply_to_status_id',
                   'in_reply_to_user_id',
                   'retweeted_status_id',
                   'retweeted_status_user_id',
                   'retweeted_status_timestamp']
for col in dropped_columns:
    assert col not in tweet_clean.columns
```

4.2.2 Dealing with duplicated records

Some of the entries in the `expanded_urls` column contains double entries that are duplicates. #### Define: For the entries with two URLs, only pick the first url and drop the others

Code

[28]:
```
separator = ','
for entry in tweet_clean.expanded_urls:
    if separator in str(entry):
        tweet_clean.expanded_urls = entry.split(',')[0]
```

Test

[29]:
```
# If the duplicated urls were removed successfully, this test will pass without␣
 ↪an assertion error
for url in tweet_clean.expanded_urls:
    assert separator not in url
```

4.2.3 `source column` contain a tags (<a/>).

Define Extract the `href` and `name of the source` from the a tag. After the extraction, delete the old source column

code

[30]:
```
for tag in tweet_clean.source:
    soup = BeautifulSoup(tag, "html.parser")
    tweet_clean['source_name'] = soup.find('a').contents[0]
    tweet_clean['source_url'] = soup.find('a').get('href')

# Now, delete the old column
tweet_clean.drop('source', axis=1, inplace=True)
```

Test

[31]:
```
# Assert that the new columns have been added to the dataframe and that the old␣
 ↪column has been downloaded.
# If the code above was successful, these assertion tests should pass without␣
 ↪assertion errors.
new_column = ['source_name', 'source_url']
for col in new_column:
    assert col in tweet_clean.columns

assert 'source' not in tweet_clean.columns
```

4.2.4 Erroneous datatypes (timestamp, retweeted_status_timestamp, rating_denominator, rating_numerator columns).

Define:

- Timestamp, retweeted_status_timestamp columns have some tidiness issues. therefore, we hold this columns for now.
- Change the datatype of rating_denominator, rating_numerator columns from float to integers

Code

[32]:
```
tweet_clean.rating_denominator = tweet_clean.rating_denominator.astype(int)
tweet_clean.rating_numerator = tweet_clean.rating_numerator.astype(int)
```

test

[33]:
```
# Check to see that the datatypes have been changed as required.
tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2356 non-null   int64
 1   timestamp           2356 non-null   object
 2   text                2356 non-null   object
```

```
 3   expanded_urls       2356 non-null   object
 4   rating_numerator    2356 non-null   int64
 5   rating_denominator  2356 non-null   int64
 6   name                2356 non-null   object
 7   doggo               2356 non-null   object
 8   floofer             2356 non-null   object
 9   pupper              2356 non-null   object
 10  puppo               2356 non-null   object
 11  source_name         2356 non-null   object
 12  source_url          2356 non-null   object
dtypes: int64(3), object(10)
memory usage: 239.4+ KB
```

4.3 Image Prediction dataframe

4.3.1 This dataframe has non-descriptive column names.

Define: Change: > - img_num –> number_of_images > - p1 –> prediction_1 > - p1_conf –> prediction_1_confidence > - p1_dog –> prediction_1_dog > - p2 –> prediction_2 > - p2_conf –> prediction_2_confidence > - p2_dog –> prediction_2_dog > - p3 –> prediction_3 > - p3_conf –> prediction_3_confidence > - p3_dog –> prediction_3_dog

code

[34]:
```
image_prediction_clean.columns = ['tweet_id', 'jpg_url', 'number_of_images',␣
↪'prediction_1', 'prediction_1_confidence', 'prediction_1_dog',␣
↪'prediction_2', 'prediction_2_confidence', 'prediction_2_dog',␣
↪'prediction_3', 'prediction_3_confidence', 'prediction_3_dog']
```

test

[35]:
```
image_prediction_clean.columns
```

[35]:
```
Index(['tweet_id', 'jpg_url', 'number_of_images', 'prediction_1',
       'prediction_1_confidence', 'prediction_1_dog', 'prediction_2',
       'prediction_2_confidence', 'prediction_2_dog', 'prediction_3',
       'prediction_3_confidence', 'prediction_3_dog'],
      dtype='object')
```

4.3.2 Comparing the number of entries in the image prediction dataframe and the dog tweets dataframe, it is clear that some of the tweets have no corresponding image predictions in the image prediction dataframe.

Define: Since there are fewer entries in the in the image prediction dataframe than in the dog tweet dataframe, it is evident that some of the dog tweets dont have image predictions. Since we have no means of obtaining the predictions, there is nothing we can do to address this issue.

4.4 Tidiness issues

4.5 Dog Tweet dataframe

4.5.1 Timestamp column has two variable entries (Date & Time).

Define: Separate the timestamp column to form the date column and the time columns > - first drop the '+0000' at the end of every entry > - since the date and time are separated using a space, separate them and assign the entry to the left to the date column and that in the right of the space to the time column

code

[36]:
```
for stamp in tweet_clean.timestamp:
    stamp = stamp[:-6]
    tweet_clean['date'], tweet_clean['time'] = stamp.split(' ', 1)

# Now, drop the old timestamp column
tweet_clean.drop('timestamp', axis=1, inplace=True)
```

Test

[37]:
```
# Display the columns of the dataframe to check that the timestamp column has␣
 ↪been created and that the date & time columns have been created
tweet_clean.columns
```

[37]:
```
Index(['tweet_id', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo',
       'source_name', 'source_url', 'date', 'time'],
      dtype='object')
```

Now that the timestamp has been separated as required, we can comfortably change the datatypes of the columns.

[38]:
```
tweet_clean.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   text                2356 non-null   object
 2   expanded_urls       2356 non-null   object
 3   rating_numerator    2356 non-null   int64
 4   rating_denominator  2356 non-null   int64
 5   name                2356 non-null   object
 6   doggo               2356 non-null   object
 7   floofer             2356 non-null   object
 8   pupper              2356 non-null   object
```

```
 9   puppo               2356 non-null   object
 10  source_name         2356 non-null   object
 11  source_url          2356 non-null   object
 12  date                2356 non-null   object
 13  time                2356 non-null   object
dtypes: int64(3), object(11)
memory usage: 257.8+ KB
```

code

[39]:
```
# change date and time columns to type datetime
tweet_clean.date = pd.to_datetime(tweet_clean.date)
tweet_clean.time = pd.to_datetime(tweet_clean.time)
```

Test

[40]:
```
tweet_clean.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   text                2356 non-null   object
 2   expanded_urls       2356 non-null   object
 3   rating_numerator    2356 non-null   int64
 4   rating_denominator  2356 non-null   int64
 5   name                2356 non-null   object
 6   doggo               2356 non-null   object
 7   floofer             2356 non-null   object
 8   pupper              2356 non-null   object
 9   puppo               2356 non-null   object
 10  source_name         2356 non-null   object
 11  source_url          2356 non-null   object
 12  date                2356 non-null   datetime64[ns]
 13  time                2356 non-null   datetime64[ns]
dtypes: datetime64[ns](2), int64(3), object(9)
memory usage: 257.8+ KB
```

4.5.2 source column has two variables (source name & source url).

Define: This column contained two sets of data within the 'a' tags. However, when extracting the content of the 'a' tags, this tidiness issue was solved since the source name and the source url were extracted and put into different columns.

4.5.3 Text column has four variables in the same column (The tweet, rating_numerator, rating_denominator, Tweet link). note: the tweet link in this column is similar to the link in the source column

Define: We already have the rating denominator, the rating numerator, and the tweet link in separate clean columns. Therefore, to clean this column, we only need to extract the message and drop all other entries.

code

[41]:
```
# Drop the ulrs in every cell
sep = '[0-9]'
pattern  = sep + ".*"
for text in tweet_clean.text:
    text = re.sub(pattern, '', text)
```

test

[42]:
```
# Assert that the links have been removed from the text. this should pass␣
 ↪without an assertion error.
for text in tweet_clean.text:
    assert 'http'or 'https' not in tweet_clean.text
```

4.5.4 The `doggo`, `floofer`, `pupper`, `puppo` should be variables under `affectionate_name` column

Define combine the columns and separate the entries with a dash (-) then delete the word `None` to remain with the affectionate names only. Lastly, drop the old unwanted columns

code

[43]:
```
tweet_clean["affectionate_name"] = tweet_clean['doggo']+ '-' +␣
 ↪tweet_clean['floofer'] + '-' + tweet_clean['pupper']+ '-' +␣
 ↪tweet_clean['puppo']
tweet_clean.affectionate_name = tweet_clean.affectionate_name.str.
 ↪replace('None', '')
tweet_clean.affectionate_name = tweet_clean.affectionate_name.str.strip('-')

# Replace the empty entries of the affectionate_name column with the name␣
 ↪(None)
for name in tweet_clean.affectionate_name:
    if name == '':
        tweet_clean.affectionate_name = 'None'

# Now drop the old columns
tweet_clean.drop(['doggo', 'floofer', 'pupper', 'puppo'], axis=1, inplace=True)
```

Test

[44]:
```
# Display a sample of the dataframe to see the changes in the columns
tweet_clean.head()
```

[44]:
```
             tweet_id                                               text  \
0  892420643555336193  This is Phineas. He's a mystical boy. Only eve...
1  892177421306343426  This is Tilly. She's just checking pup on you...
2  891815181378084864  This is Archie. He is a rare Norwegian Pouncin...
3  891689557279858688  This is Darla. She commenced a snooze mid meal...
4  891327558926688256  This is Franklin. He would like you to stop ca...

                                        expanded_urls  rating_numerator  \
0  https://twitter.com/dogratingrating/status/667...                13
1  https://twitter.com/dogratingrating/status/667...                13
2  https://twitter.com/dogratingrating/status/667...                12
3  https://twitter.com/dogratingrating/status/667...                13
4  https://twitter.com/dogratingrating/status/667...                12

   rating_denominator      name         source_name  \
0                  10   Phineas  Twitter for iPhone
1                  10     Tilly  Twitter for iPhone
2                  10    Archie  Twitter for iPhone
3                  10     Darla  Twitter for iPhone
4                  10  Franklin  Twitter for iPhone

                           source_url       date                time  \
0  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
1  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
2  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
3  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
4  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08

  affectionate_name
0              None
1              None
2              None
3              None
4              None
```

[45]:
```
tweet_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2356 non-null   int64
 1   text                2356 non-null   object
 2   expanded_urls       2356 non-null   object
```

```
 3   rating_numerator    2356 non-null   int64
 4   rating_denominator  2356 non-null   int64
 5   name                2356 non-null   object
 6   source_name         2356 non-null   object
 7   source_url          2356 non-null   object
 8   date                2356 non-null   datetime64[ns]
 9   time                2356 non-null   datetime64[ns]
 10  affectionate_name   2356 non-null   object
dtypes: datetime64[ns](2), int64(3), object(6)
memory usage: 202.6+ KB
```

4.6 Favourite counts & retweet counts dataframe

4.6.1 We do not need this table because the information it holds (favourite counts and retweet counts) should appear alongside each respective tweet in the dog tweet dataframe.

Define: Combine/merge the reactions count dataframe to the tweet dataframe on the tweet id

code

[46]:
```
tweet_clean = pd.merge(tweet_clean, reaction_count_clean,
                       on='tweet_id')
```

test

[47]:
```
tweet_clean.head()
```

[47]:
```
             tweet_id                                               text  \
0  892420643555336193  This is Phineas. He's a mystical boy. Only eve...
1  892177421306343426  This is Tilly. She's just checking pup on you...
2  891815181378084864  This is Archie. He is a rare Norwegian Pouncin...
3  891689557279858688  This is Darla. She commenced a snooze mid meal...
4  891327558926688256  This is Franklin. He would like you to stop ca...

                                        expanded_urls  rating_numerator  \
0  https://twitter.com/dogratingrating/status/667...                13
1  https://twitter.com/dogratingrating/status/667...                13
2  https://twitter.com/dogratingrating/status/667...                12
3  https://twitter.com/dogratingrating/status/667...                13
4  https://twitter.com/dogratingrating/status/667...                12

   rating_denominator      name         source_name  \
0                  10   Phineas  Twitter for iPhone
1                  10     Tilly  Twitter for iPhone
2                  10    Archie  Twitter for iPhone
3                  10     Darla  Twitter for iPhone
4                  10  Franklin  Twitter for iPhone

                        source_url       date                time  \
```

```
0  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
1  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
2  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
3  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
4  http://twitter.com/download/iphone 2015-11-15 2022-06-28 22:32:08
```

	affectionate_name	retweet_count	favorite_count
0	None	7009	33815
1	None	5302	29320
2	None	3481	22050
3	None	7219	36903
4	None	7762	35311

5 Storing Data

[48]:
```
# Check the if the required folder exists. If it does not exist, create it
folder_name = 'Datasets/Clean_data'
if not os.path.exists (folder_name):
    os.makedirs(folder_name)

tweet_clean.to_csv('Datasets/Clean_data/twitter_archive_master.csv',␣
 ↪index=False)
image_prediction_clean.to_csv('Datasets/Clean_data/image_prediction_master.
 ↪csv', index=False)
```

6 Analyzing and Visualizing Data

[ ]:
```

```