wrangle_report

January 27, 2018

1 WeRateDogs Data Wrangling

The dataset that we will wrangle consists of three sources:

The tweet archive of Twitter user @dog_rates.

The site is 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. Why? Because "they're good dogs Brent." WeRateDogs has over 4 million followers and has received international media coverage.

Additional Data via the Twitter API

Back to the basic-ness of Twitter archives: retweet count and favorite count are two of the notable column omissions. Fortunately, this additional data can be gathered by anyone from Twitter's API. We're going to query Twitter's API to gather this valuable data.

The Image Predictions File Thirdly, we have got access to results of a neural network that can classify breeds of dogs!

So, we will clean this data and at the prepare a dataset for visualization and statistical analysis.

1.1 Gather

First of all we will import all required libraries.

```
In [1]: #connect to the internet
        import requests

        #deal with data
        import numpy as np
        import pandas as pd

        #deal with datetime
        import datetime as dt
        import pytz

        #deal with visualization
        import seaborn as sns
        %matplotlib inline
        import matplotlib.pyplot as plt
```

```
#use padasql for SQL-query on dataframe
#http://blog.yhat.com/posts/pandasql-intro.html
from pandasql import sqldf
```

1.1.1 gather twitter-archive-enhanced.csv file

Read the locally stored file regarding the twitter archive

```
In [2]: df_twarchive = pd.read_csv('twitter-archive-enhanced.csv')
```

1.1.2 gather image-predictions.tsv file

```
In [3]: tgt_filename = 'image-predictions.tsv'
```

Download the image predictions file from the cloud

```
In [4]: url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictic
        r = requests.get(url)
        with open(tgt_filename, 'wb') as file:
            file.write(r.content)
```

```
In [5]: #read dataframe from file
        df_image = pd.read_csv(tgt_filename, sep = '\t')
        df_image.tweet_id.count()
```

```
Out[5]: 2075
```

1.1.3 gather tweed_json.txt

We gather each tweet's retweet count and favorite ("like") count using the **tweet IDs in the WeRateDogs Twitter archive.**

We woill 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. - tweet ID - retweet count - favorite count

```
In [7]: #refer to twitter application https://apps.twitter.com/app/14714948/keys
        #import authorization variables for security reasons
        import access.my_secret as xs
        import tweepy

        auth = tweepy.OAuthHandler(xs.consumer_key, xs.consumer_secret)
        auth.set_access_token(xs.access_token, xs.access_secret)
        api = tweepy.API(auth)
```

```
In [8]: #Extract tweet in json format from twitter api:
        def get_tweet(tweet_id):
            tweet = api.get_status(tweet_id, tweet_mode='extended', wait_on_rate_limit=True)
            return tweet._json
```

```
In [9]: # for testing
        # print(get_tweet(666082916733198337))
```

```
In [7]: #get the timer set up
        from timeit import default_timer as timer
        import datetime
        import json

        json_filename = 'tweet_json.txt'
```

```
In [22]: #get tweet
         #write json file line by line and log progress
         with open(json_filename, 'w') as file, open('log.txt','w') as log:
             for tweet_id in df_twarchive.tweet_id:
                 result = ''
                 start = timer()
                 try:
                     content = get_tweet(tweet_id)
                     #add newline is helpful to read the file line by line later
                     file.write(json.dumps(content) + '\n')
                     result = 'ok'
                 except tweepy.TweepError:
                     result = 'TweepError'
                 end = timer()
                 log.write(('%s\t%s\t%s\t%s\n') % (str(datetime.datetime.now()), result, str(twe
```

```
In [8]: #count number of tweets
        i = 0
        with open(json_filename, 'r') as file:
            for row in file:
                i += 1
        print("row count:", i)
```

```
row count: 2345
```

1.2 Assess

After gathering each of the above pieces of data, we assess them visually and programmatically for quality and tidiness issues. We detect and document at least **eight (8) quality issues and two (2) tidiness issues** in your wrangle_act.ipynb Jupyter Notebook. To meet specifications, the **issues that satisfy the Project Motivation must be assessed.**

1.2.1 assess twitter-archive-enhanced.csv file

Most of the quality issue have been detected in the twitter archvie file. Theese are typical isses. We will deal with them one at a time.

quality issues

1. derive clear categories for source(iPhone App, vine.co, Twitter Web Client, TweetDeck)
2. make this source column categorical
3. deal with a few invalid ratings where rating_numerator is 0 or rating_denominator is 0
    - 835246439529840640 rating has been changed to from 960/0 to 13/10 according to current twitter post
    - 835152434251116546 rating has been changed to from 0/10 to 11/10 according to current twitter post
    - 746906459439529985 no rating according to current twitter post -> filter out
4. take out ratings with a rating_denominator other than 10 as these are invalid ratings or rate multiple dogs at once.
5. take out ratings with rating_numerator > 15 they seem to be invalid
6. take out retweeds 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. (retweeted_status_id is NaN)
7. convert wrong dog names to Null: “a”, “the”, “an”, “n”, “None”

tidyness issues

1. four columns for the dog_stage can be put into one categorial variable (doggo,floofer,pupper,puppo) and turn the default value “None” into a Null or “unknown”

First of all we look at the data structure and sample data.

```
In [9]: df_twarchive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
```

```
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

```
In [10]: df_twarchive.sample(5)
```

Out[10]:

	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
2320	666437273139982337	NaN	NaN	2015-11-17 02:06:42 +0000	<a href="http://twitter.com/download/iphone" r...	Here we see a lone northeastern Cumberbatch. H...	NaN	NaN	NaN	https://twitter.com/dog_rates/status/666437273...	7
1943	673709992831262724	NaN	NaN	2015-12-07 03:45:53 +0000	<a href="http://twitter.com/download/iphone" r...	I know a lot of you are studying for finals. G...	NaN	NaN	NaN	https://twitter.com/dog_rates/status/673709992...	12
1676	682088079302213632	NaN	NaN	2015-12-30 06:37:25 +0000	<a href="http://vine.co" rel="nofollow">Vine -...	I'm not sure what this dog is doing but it's p...	NaN	NaN	NaN	https://vine.co/v/iqMjlxULzbn	12
1831	676215927814406144	NaN	NaN	2015-12-14 01:43:35 +0000	<a href="http://twitter.com/download/iphone" r...	This is Herm. He just wants to be like the oth...	NaN	NaN	NaN	https://twitter.com/dog_rates/status/676215927...	9
785	775085132600442880	NaN	NaN	2016-09-11 21:34:30 +0000	<a href="http://twitter.com/download/iphone" r...	This is Tucker. He would like a hug. 13/10 som...	NaN	NaN	NaN	https://twitter.com/dog_rates/status/775085132...	13

```
      rating_denominator    name doggo floofer pupper puppo
2320                  10    None  None    None   None  None
1943                  10    None  None    None   None  None
1676                  10    None  None    None   None  None
1831                  10    Herm  None    None   None  None
785                   10  Tucker  None    None   None  None
```

Analysing the twitter sources reveals that the site owner creates most of the tweets from his iPhone.

Only a few other sources are relvant. We will clean this up

```
In [11]: df_twarchive.source.value_counts()
```

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

The tweets range from End of 2015 to August 2017.

```
In [12]: #check the latest tweet
         print(df_twarchive.timestamp.min())
         print(df_twarchive.timestamp.max())
```

```
2015-11-15 22:32:08 +0000
2017-08-01 16:23:56 +0000
```

```
In [13]: df_twarchive.describe()
```

```
Out[13]:            tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
         count  2.356000e+03           7.800000e+01         7.800000e+01
         mean   7.427716e+17           7.455079e+17         2.014171e+16
         std    6.856705e+16           7.582492e+16         1.252797e+17
         min    6.660209e+17           6.658147e+17         1.185634e+07
         25%    6.783989e+17           6.757419e+17         3.086374e+08
         50%    7.196279e+17           7.038708e+17         4.196984e+09
         75%    7.993373e+17           8.257804e+17         4.196984e+09
         max    8.924206e+17           8.862664e+17         8.405479e+17

                retweeted_status_id  retweeted_status_user_id  rating_numerator  \
         count         1.810000e+02              1.810000e+02       2356.000000
         mean          7.720400e+17              1.241698e+16         13.126486
         std           6.236928e+16              9.599254e+16         45.876648
         min           6.661041e+17              7.832140e+05          0.000000
         25%           7.186315e+17              4.196984e+09         10.000000
         50%           7.804657e+17              4.196984e+09         11.000000
         75%           8.203146e+17              4.196984e+09         12.000000
```

```
max           8.874740e+17              7.874618e+17       1776.000000

       rating_denominator
count         2356.000000
mean            10.455433
std              6.745237
min              0.000000
25%             10.000000
50%             10.000000
75%             10.000000
max            170.000000
```

1.2.2 Analyse 0 values in the rating system

I've analysed zero ratings in the rating system as they are actually not allowed!

Once I've got the the twitte_id, I am able to check the current post on twitter.com

```
In [14]: df_twarchive.query('rating_numerator == 0 | rating_denominator == 0')
         #832464395298406 40 rating has been changed to from 960/0 to 13/10 according to current
         #831524342511165 46 rating has been changed to from 0/10 to 11/10 according to current
         #746906459439529985 no rating according to current twitter post -> filter out
```

```
Out[14]:                tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
         313   832464395298406 40          8.352460e+17         2.625958e+07
         315   831524342511165 46                   NaN                  NaN
         1016  746906459439529985          7.468859e+17         4.196984e+09

                              timestamp  \
         313   2017-02-24 21:54:03 +0000
         315   2017-02-24 15:40:31 +0000
         1016  2016-06-26 03:22:31 +0000

                                                    source  \
         313   <a href="http://twitter.com/download/iphone" r...
         315   <a href="http://twitter.com/download/iphone" r...
         1016  <a href="http://twitter.com/download/iphone" r...

                                                      text  retweeted_status_id  \
         313   @jonnysun @Lin_Manuel ok jomny I know you're e...                  NaN
         315   When you're so blinded by your systematic plag...                  NaN
         1016  PUPDATE: can't see any. Even if I could, I cou...                  NaN

               retweeted_status_user_id retweeted_status_timestamp  \
         313                        NaN                        NaN
         315                        NaN                        NaN
         1016                       NaN                        NaN

                                             expanded_urls  rating_numerator  \
```

```
313                                                  NaN                960
315   https://twitter.com/dog_rates/status/835152434...                  0
1016  https://twitter.com/dog_rates/status/746906459...                  0

      rating_denominator  name doggo floofer pupper puppo
313                    0  None  None    None   None  None
315                   10  None  None    None   None  None
1016                  10  None  None    None   None  None
```

Furthermore the rating denomiator should be 10 according to wikipedia: https://en.wikipedia.org/wiki/WeRateDogs

We can also learn that the site owner is student at Campbell University in Buies Creek, **North Carolina**.

```
In [15]: df_twarchive[['tweet_id','rating_numerator','rating_denominator']].query('rating_denomi
```

Out[15]:

	tweet_id	rating_numerator	rating_denominator
313	835246439529840640	960	0
342	832088576586297345	11	15
433	820690176645140481	84	70
516	810984652412424192	24	7
784	775096608509886464	9	11
902	758467244762497024	165	150
1068	740373189193256964	9	11
1120	731156023742988288	204	170
1165	722974582966214656	4	20
1202	716439118184652801	50	50
1228	713900603437621249	99	90
1254	710658690886586372	80	80
1274	709198395643068416	45	50
1351	704054845121142784	60	50
1433	697463031882764288	44	40
1598	686035780142297088	4	20
1634	684225744407494656	143	130
1635	684222868335505415	121	110
1662	682962037429899265	7	11
1663	682808988178739200	20	16
1779	677716515794329600	144	120
1843	675853064436391936	88	80
2335	666287406224695296	1	2

```
In [16]: df = df_twarchive[['tweet_id','rating_numerator','rating_denominator']].query('rating_d
         df.rating_numerator.value_counts().value_counts()
         #df.query('rating_numerator > 15')
```

```
Out[16]: 1      6
         2      4
         54     2
         351    1
```

```
156    1
19     1
463    1
558    1
461    1
8      1
9      1
15     1
102    1
37     1
32     1
Name: rating_numerator, dtype: int64
```

The dogs on twitter have been categories by different stages.

Just to make sure we are all talking about the same there is a very good overview viedo on youtube:

https://www.youtube.com/watch?v=ah6fmNEtXFI

In the current data set we are talking about - doggo - floofer - pupper - pupppo

```
In [17]: #check if categories are clean
         categories = ['doggo', 'floofer', 'pupper', 'puppo']
         for category in categories:
             print(df_twarchive[category].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
```

We may want to analyse dog names an further. So we want to make sure they are proberperly cleaned up.

```
In [18]: df_twarchive.name.value_counts().head(20)
         #convert to Null: a, the, an
```

```
Out[18]: None       745
         a           55
         Charlie     12
         Oliver      11
         Cooper      11
         Lucy        11
         Tucker      10
         Lola        10
         Penny       10
         Winston      9
         Bo           9
         the          8
         Sadie        8
         Buddy        7
         Toby         7
         Daisy        7
         an           7
         Bailey       7
         Leo          6
         Milo         6
         Name: name, dtype: int64
```

1.3 Assess image.csvű

The tweet image predictions, i.e., what breed of dog (or other object, animal, etc.) is present in each tweet according to a neural network.

Example for the data set: - tweet_id is the last part of the tweet URL after "status/" a simple query for tweet id would be: (https://twitter.com/dog_rates/status/889531135344209921)

```
- p1 is the algorithm's #1 prediction for the image in the tweet  golden retriever
- p1_conf is how confident the algorithm is in its #1 prediction  95%
- p1_dog is whether or not the #1 prediction is a breed of dog  TRUE
- p2 is the algorithm's second most likely prediction  Labrador retriever
- p2_conf is how confident the algorithm is in its #2 prediction  1%
- p2_dog is whether or not the #2 prediction is a breed of dog  TRUE
- etc.
```

quality issues

- we have 66 duplicates and have to make sure they don't exists in the final data set

tidiyness issues

2. get the best true dog prediction in one column
3. make the dog prediction categorial

```
In [19]: df_image.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [20]:
```
df_image.sample(5)
```

Out[20]:

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog
1457	777684233540206592	https://pbs.twimg.com/media/CsrjryzWgAAZY00.jpg	1	cocker_spaniel	0.253442	True	golden_retriever	0.162850	True	otterhound	0.110921	True
790	690649993829576704	https://pbs.twimg.com/media/CZWugJsWYAIzVzJ.jpg	1	bighorn	0.215438	False	hyena	0.137928	False	Mexican_hairless	0.098171	True
928	702684942141153280	https://pbs.twimg.com/media/CcBwOnOXEAA7bNQ.jpg	1	golden_retriever	0.514085	True	Chesapeake_Bay_retriever	0.173224	True	Brittany_spaniel	0.118384	True
820	692919143163629568	https://pbs.twimg.com/media/CZ2-SRiWcAIjuM5.jpg	1	Saint_Bernard	0.612635	True	English_springer	0.269744	True	boxer	0.048666	True
964	706166467411222528	https://pbs.twimg.com/media/CczOp_OWoAAo5zR.jpg	1	Samoyed	0.430418	True	kuvasz	0.279600	True	Great_Pyrenees	0.117480	True

In [21]:
```
df_image.describe()
```

Out[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
```

In [22]:
```
df_image[df_image.duplicated(subset=['jpg_url'], keep='first')].head(5)
```

Out[22]:

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_conf	p3_dog
1297	752309394570878976	https://pbs.twimg.com/ext_tw_video_thumb/67535...	1	upright	0.303415	False	golden_retriever	0.181351	True	Brittany_spaniel	0.162084	True
1315	754874841593970688	https://pbs.twimg.com/media/CWza7kpWcAAdYLc.jpg	1	pug	0.272205	True	bull_mastiff	0.251530	True	bath_towel	0.116806	False
1333	757729163776290825	https://pbs.twimg.com/media/CWyD2HGUYAQ1Xa7.jpg	2	cash_machine	0.802333	False	schipperke	0.045519	True	German_shepherd	0.023353	True
1345	759159934323924993	https://pbs.twimg.com/media/CU1zsMSUAAAS0qW.jpg	1	Irish_terrier	0.254856	True	briard	0.227716	True	soft-coated_wheaten_terrier	0.223263	True
1349	759566828574212096	https://pbs.twimg.com/media/CkNjahBXAAQ2kWo.jpg	1	Labrador_retriever	0.967397	True	golden_retriever	0.016641	True	ice_bear	0.014858	False

1.3.1 assess tweet_json.txt

To assess the JSON object I refer to - https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object or - http://support.gnip.com/sources/twitter/data_format.html with more detail on - http://support.gnip.com/doing-more-with-140.html

tweet ID, retweet count, and favorite count

In [23]:
```
#read json file line by line
df_list=[]
with open(json_filename, 'r') as file:
    for row in file:
        data = json.loads(row)
        df_list.append({"tweet_id":data["id"],
                        "retweet_count":data["retweet_count"],
                        "favorite_count":data["favorite_count"]
                        })
#df_tweet = pd.DataFrame.from_dict(df_list)
df_tweet = pd.DataFrame(df_list, columns=["tweet_id", "retweet_count", "favorite_count"
```

In [24]:
```
df_tweet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2345 entries, 0 to 2344
Data columns (total 3 columns):
tweet_id          2345 non-null int64
retweet_count     2345 non-null int64
favorite_count    2345 non-null int64
dtypes: int64(3)
memory usage: 55.0 KB
```

```
In [25]: df_tweet.describe()
```

Out[25]:

	tweet_id	retweet_count	favorite_count
count	2.345000e+03	2345.000000	2345.000000
mean	7.423760e+17	3078.081876	8114.238380
std	6.836820e+16	5105.627191	12189.503552
min	6.660209e+17	0.000000	0.000000
25%	6.783802e+17	614.000000	1410.000000
50%	7.189719e+17	1437.000000	3574.000000
75%	7.987057e+17	3585.000000	10068.000000
max	8.924206e+17	78450.000000	142572.000000

```
In [26]: df_tweet[df_tweet.retweet_count>70000]
```

Out[26]:

	tweet_id	retweet_count	favorite_count
1029	744234799360020481	78450	129628

```
In [27]: df_twarchive[df_twarchive.tweet_id == 744234799360020481]
```

Out[27]:

	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
1039	744234799360020481	NaN	NaN	2016-06-18 18:26:18 +0000	<a href="http://twitter.com/download/iphone" r...	Here's a doggo realizing you can stand in a po...	NaN	NaN	NaN	https://twitter.com/dog_rates/status/744234799...	13	10	None	doggo	None	None	None

1.4 Assess new file twitter_archive_master.csv

In order to anaylse the data from all different sources we will have to merge them into one data set. #### tidyness issues - merge all three dataset into one - drop columns in the merged data set which are not required

1.5 Clean

Next we will clean each of the issues you documented while assessing. We will perform this cleaning in wrangle_act.ipynb as well. The result should be a high quality and tidy master pandas DataFrame.

Clean df_twarchive = pd.read_csv('twitter-archive-enhanced.csv')

```
In [28]: #preserve the raw data
         dfc_twarchive = df_twarchive.copy()
```

Define

- generate mapping dictionary to map to a short description with clear clear categories iPhone App, vine.co, Twitter Web Client, TweetDeck
- create function to change source
- call function
- change datatype to categorial

Code

```
In [29]: dfc_twarchive.source.value_counts()
```

```
Out[29]: <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
         <a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
         <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
         <a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>
         Name: source, dtype: int64
```

```
In [30]: ref_source = {'<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for
                       '<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>': 'vine.
                       '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>': 'Tw
                       '<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">Tw

         def clean_source(tweet):
             if tweet['source'] in ref_source.keys():
                 #return clean description
                 return ref_source[tweet['source']]
             else:
                 return dfc_twarchive.source

         dfc_twarchive.source = dfc_twarchive.apply(clean_source, axis=1)

         dfc_twarchive.source = dfc_twarchive.source.astype('category')
```

Test

```
In [31]: dfc_twarchive.source.value_counts()

Out[31]: iPhone App            2221
         vine.co                 91
         Twitter Web Client      33
         TweetDeck               11
         Name: source, dtype: int64

In [32]: dfc_twarchive.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                    2356 non-null int64
in_reply_to_status_id       78 non-null float64
in_reply_to_user_id         78 non-null float64
timestamp                   2356 non-null object
source                      2356 non-null category
text                        2356 non-null object
retweeted_status_id         181 non-null float64
retweeted_status_user_id    181 non-null float64
retweeted_status_timestamp  181 non-null object
expanded_urls               2297 non-null object
rating_numerator            2356 non-null int64
rating_denominator          2356 non-null int64
name                        2356 non-null object
doggo                       2356 non-null object
floofer                     2356 non-null object
pupper                      2356 non-null object
puppo                       2356 non-null object
dtypes: category(1), float64(4), int64(3), object(9)
memory usage: 297.1+ KB
```

Define

- deal with a few invalid ratings where rating_numerator is 0 or rating_denominator is 0
  - 835246439529840640 rating has been changed to from 960/0 to 13/10 according to current twitter post
  - 835152434251116546 rating has been changed to from 0/10 to 11/10 according to current twitter post
  - 746906459439529985 no rating according to current twitter post -> filter out

Code

```
In [33]: #update df cell
         #see https://stackoverflow.com/questions/12307099/modifying-a-subset-of-rows-in-a-panda
         dfc_twarchive.loc[dfc_twarchive.tweet_id == 835246439529840640, 'rating_numerator'] = 1
         dfc_twarchive.loc[dfc_twarchive.tweet_id == 835246439529840640, 'rating_denominator'] =

         dfc_twarchive.loc[dfc_twarchive.tweet_id == 835152434251116546, 'rating_numerator'] = 1

         dfc_twarchive = dfc_twarchive[dfc_twarchive.tweet_id != 746906459439529985]
```

Test

```
In [34]: dfc_twarchive.loc[dfc_twarchive.tweet_id.isin([835246439529840640, 835152434251116546,
```

```
Out[34]:              tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
         313  835246439529840640           8.352460e+17           26259576.0
         315  835152434251116546                    NaN                  NaN

                              timestamp      source  \
         313  2017-02-24 21:54:03 +0000  iPhone App
         315  2017-02-24 15:40:31 +0000  iPhone App

                                                    text  retweeted_status_id  \
         313  @jonnysun @Lin_Manuel ok jomny I know you're e...                  NaN
         315  When you're so blinded by your systematic plag...                  NaN

              retweeted_status_user_id retweeted_status_timestamp  \
         313                      NaN                         NaN
         315                      NaN                         NaN

                                              expanded_urls  rating_numerator  \
         313                                            NaN                13
         315  https://twitter.com/dog_rates/status/835152434...                11

              rating_denominator  name doggo floofer pupper puppo
         313                  10  None  None    None   None  None
         315                  10  None  None    None   None  None
```

Define

- take out ratings with a rating_denominator other than 10 as these are invalid ratings or rate multiple dogs at once.
- take out ratings with rating_numerator > 15 they seem to be invalid
- take out retweets 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. So **keep** all tweets with retweeted_status_id is NaN

Code

```
In [35]: #delete retweets
         dfc_twarchive = dfc_twarchive[dfc_twarchive.retweeted_status_id.isnull()]

In [36]: dfc_twarchive = dfc_twarchive[dfc_twarchive.rating_denominator == 10]
         dfc_twarchive = dfc_twarchive[dfc_twarchive.rating_numerator <= 15]
```

Test

```
In [37]: dfc_twarchive.describe()
```

Out[37]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	\
count	2.144000e+03	6.900000e+01	6.900000e+01	
mean	7.370728e+17	7.403786e+17	2.276889e+16	
std	6.754280e+16	7.383460e+16	1.330848e+17	
min	6.660209e+17	6.658147e+17	1.185634e+07	
25%	6.766166e+17	6.753494e+17	1.198989e+09	
50%	7.095381e+17	7.030419e+17	4.196984e+09	
75%	7.895480e+17	8.131273e+17	4.196984e+09	
max	8.924206e+17	8.862664e+17	8.405479e+17	

	retweeted_status_id	retweeted_status_user_id	rating_numerator	\
count	0.0	0.0	2144.000000	
mean	NaN	NaN	10.630131	
std	NaN	NaN	2.171990	
min	NaN	NaN	1.000000	
25%	NaN	NaN	10.000000	
50%	NaN	NaN	11.000000	
75%	NaN	NaN	12.000000	
max	NaN	NaN	15.000000	

	rating_denominator
count	2144.0
mean	10.0
std	0.0
min	10.0
25%	10.0
50%	10.0
75%	10.0
max	10.0

Define

- convert wrong dog names to Null: “a”, “the”, “an”, “n”

Code

```
In [38]: # Deal with value=None
         # see https://stackoverflow.com/questions/17097236/how-to-replace-values-with-none-in-p
         dfc_twarchive.name = dfc_twarchive.name.replace(to_replace=['^(a|an|the|n|None)$'], val
```

Test

```
In [39]: dfc_twarchive.loc[dfc_twarchive.name.isin(['a', 'an', 'the', 'n', 'None'])].name.count(
```

```
Out[39]: 0
```

Define

tidyness issues

- four columns for the dog_stage can be put into one categorial variable (doggo,floofer,pupper,puppo) and turn the default value "None" into a Null or "unknown"

Code

```
In [40]: dfc_twarchive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2144 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2144 non-null int64
in_reply_to_status_id         69 non-null float64
in_reply_to_user_id           69 non-null float64
timestamp                     2144 non-null object
source                        2144 non-null category
text                          2144 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 2093 non-null object
rating_numerator              2144 non-null int64
rating_denominator            2144 non-null int64
name                          1419 non-null object
doggo                         2144 non-null object
floofer                       2144 non-null object
pupper                        2144 non-null object
puppo                         2144 non-null object
dtypes: category(1), float64(4), int64(3), object(9)
memory usage: 287.0+ KB
```

```
In [41]: #concat the stage columns doggo,floofer,pupper,puppo while replacing the None wordw wit
         dfc_twarchive['dog_stage'] = dfc_twarchive.doggo.replace('None','') + dfc_twarchive.flo
         dfc_twarchive['dog_stage'] = dfc_twarchive['dog_stage'].replace(to_replace=[''], value=
```

```
In [42]: dfc_twarchive['dog_stage'].value_counts()
```

```
Out[42]: pupper          223
         doggo            75
         puppo            24
         doggopupper      10
         floofer           9
         doggofloofer      1
         doggopuppo        1
         Name: dog_stage, dtype: int64
```

```
In [43]: # get rid of dog stages which are not defined
         # https://stackoverflow.com/questions/23330654/update-a-dataframe-in-pandas-while-itera
         for row in dfc_twarchive.itertuples():
             if dfc_twarchive.loc[row.Index, 'dog_stage'] in ['doggo','floofer','pupper','puppo'
                 next
             else:
                 dfc_twarchive.loc[row.Index, 'dog_stage'] = None
         dfc_twarchive['dog_stage'].value_counts()
```

```
Out[43]: pupper     223
         doggo       75
         puppo       24
         floofer      9
         Name: dog_stage, dtype: int64
```

```
In [44]: dfc_twarchive['dog_stage'] = dfc_twarchive['dog_stage'].astype('category')
```

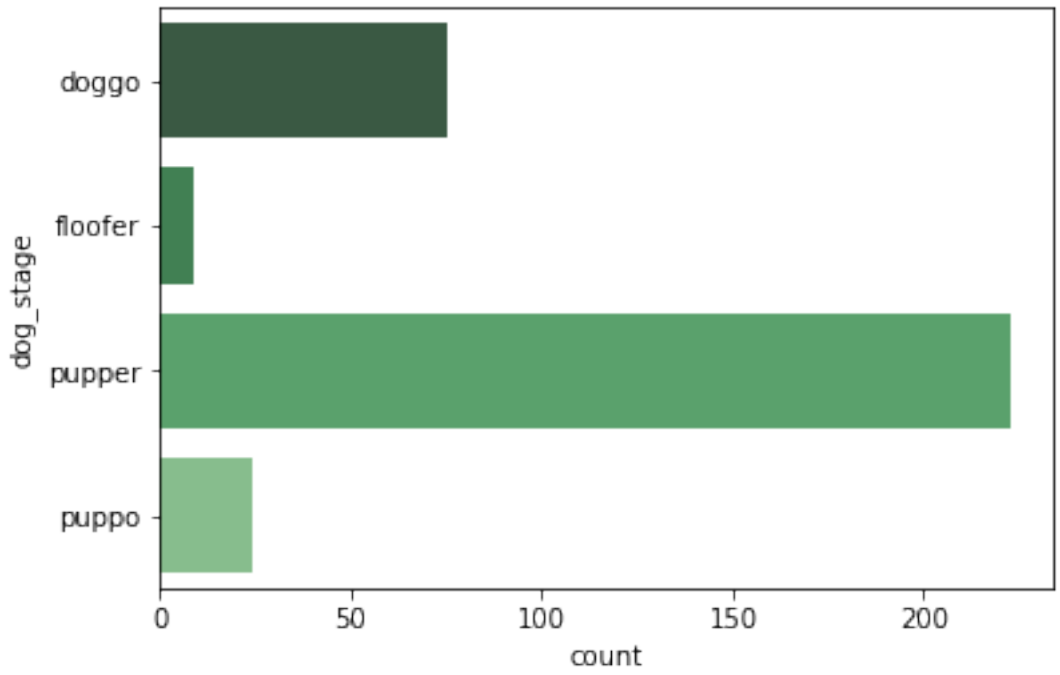
Test

```
In [45]: dfc_twarchive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2144 entries, 0 to 2355
Data columns (total 18 columns):
tweet_id                      2144 non-null int64
in_reply_to_status_id         69 non-null float64
in_reply_to_user_id           69 non-null float64
timestamp                     2144 non-null object
source                        2144 non-null category
text                          2144 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 2093 non-null object
rating_numerator              2144 non-null int64
rating_denominator            2144 non-null int64
name                          1419 non-null object
doggo                         2144 non-null object
floofer                       2144 non-null object
pupper                        2144 non-null object
```

```
puppo                         2144 non-null object
dog_stage                     331 non-null category
dtypes: category(2), float64(4), int64(3), object(9)
memory usage: 369.3+ KB
```

```
In [46]: sns.countplot(y="dog_stage", data=dfc_twarchive, palette="Greens_d");
```



Define drop all irrelevant columns

Code

```
In [47]: dfc_twarchive.drop(['in_reply_to_status_id',
                             'in_reply_to_user_id',
                             'retweeted_status_id',
                             'retweeted_status_user_id',
                             'retweeted_status_timestamp',
                             'doggo',
                             'floofer',
                             'pupper',
                             'puppo'], axis=1, inplace=True)
```

Test

```
In [48]: dfc_twarchive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2144 entries, 0 to 2355
Data columns (total 9 columns):
tweet_id              2144 non-null int64
timestamp             2144 non-null object
source                2144 non-null category
text                  2144 non-null object
expanded_urls         2093 non-null object
rating_numerator      2144 non-null int64
rating_denominator    2144 non-null int64
name                  1419 non-null object
dog_stage             331 non-null category
dtypes: category(2), int64(3), object(4)
memory usage: 218.6+ KB
```

1.5.1 clean image-predictions.tsv file

```
In [49]: dfc_image = df_image.copy()
```

Define

- Go through each line and look for true dog predictions from column 1 to column 3.
- Stop at the first occurence of a dog prediction
- Write that info into a separate field
- make that the dog prediction column categorial

Code

```
In [50]: dfc_image.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

```
In [51]: #make sure the priority is correct in the columns
         dfc_image.query('p1_conf < p2_conf | p2_conf < p3_conf | p1_conf < p3_conf')
```

```
Out[51]: Empty DataFrame
         Columns: [tweet_id, jpg_url, img_num, p1, p1_conf, p1_dog, p2, p2_conf, p2_dog, p3, p3_
         Index: []
```

```
In [52]: #merge thre dog columns into one according to priority
         df_p1 = dfc_image[dfc_image.p1_dog == True].filter(items=['tweet_id','jpg_url','p1','p1
         df_p1 = df_p1.rename(columns={'p1': 'dog', 'p1_conf': 'conf'})
         df_p2 = dfc_image.query('p1_dog == False & p2_dog == True').filter(items=['tweet_id','j
         df_p2 = df_p2.rename(columns={'p2': 'dog', 'p2_conf': 'conf'})
         df_p3 = dfc_image.query('p1_dog == False & p2_dog == False & p3_dog == True').filter(it
         df_p3 = df_p3.rename(columns={'p3': 'dog', 'p3_conf': 'conf'})
         dfc_image = pd.concat([df_p1, df_p2, df_p3])
```

```
In [53]: dfc_image.dog = dfc_image.dog.astype('category')
```

Test

```
In [54]: dfc_image.head(5)
```

```
Out[54]:              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

                           dog      conf
         0  Welsh_springer_spaniel  0.465074
         1                redbone  0.506826
         2        German_shepherd  0.596461
         3    Rhodesian_ridgeback  0.408143
         4     miniature_pinscher  0.560311
```

```
In [55]: dfc_image.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1751 entries, 0 to 2026
Data columns (total 4 columns):
tweet_id    1751 non-null int64
jpg_url     1751 non-null object
dog         1751 non-null category
conf        1751 non-null float64
dtypes: category(1), float64(1), int64(1), object(1)
memory usage: 62.3+ KB
```

```
In [56]: df_image.query('tweet_id == 666044226329800704')
```

```
Out[56]:              tweet_id                                          jpg_url  \
         3  666044226329800704  https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg

            img_num                   p1   p1_conf  p1_dog       p2   p2_conf  p2_dog  \
         3        1  Rhodesian_ridgeback  0.408143    True  redbone  0.360687    True

                            p3   p3_conf  p3_dog
         3  miniature_pinscher  0.222752    True
```

Define join together dfc_twarchive x df_tweet add image info if it exists from dfc_image

Code

```
In [57]: df_tweet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2345 entries, 0 to 2344
Data columns (total 3 columns):
tweet_id          2345 non-null int64
retweet_count     2345 non-null int64
favorite_count    2345 non-null int64
dtypes: int64(3)
memory usage: 55.0 KB
```

```
In [58]: j1 = pd.merge(dfc_twarchive, df_tweet, on='tweet_id')
         #filter out dogs where we don't know the type
         df_master = pd.merge(j1, dfc_image, on='tweet_id')
```

```
In [59]: df_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1664 entries, 0 to 1663
Data columns (total 14 columns):
tweet_id              1664 non-null int64
timestamp             1664 non-null object
source                1664 non-null category
text                  1664 non-null object
expanded_urls         1664 non-null object
rating_numerator      1664 non-null int64
rating_denominator    1664 non-null int64
name                  1203 non-null object
dog_stage             249 non-null category
retweet_count         1664 non-null int64
favorite_count        1664 non-null int64
jpg_url               1664 non-null object
dog                   1664 non-null category
```

```
conf                  1664 non-null float64
dtypes: category(3), float64(1), int64(5), object(5)
memory usage: 167.1+ KB
```

```
In [60]: df_master.sample(1)
```

```
Out[60]:                tweet_id                   timestamp      source  \
         857  709852847387627521  2016-03-15 21:24:41 +0000  iPhone App

                                                        text  \
         857  *lets out a tiny whimper and then collapses* ...

                                                expanded_urls  rating_numerator  \
         857  https://twitter.com/dog_rates/status/709852847...                12

              rating_denominator  name dog_stage  retweet_count  favorite_count  \
         857                  10  None       NaN           1320            3775

                                                  jpg_url        dog      conf
         857  https://pbs.twimg.com/media/CdnnZhhWAAEAoUc.jpg  Chihuahua  0.945629
```

```
In [61]: df_master[df_master.jpg_url.isnull()].count()
```

```
Out[61]: tweet_id              0
         timestamp             0
         source                0
         text                  0
         expanded_urls         0
         rating_numerator      0
         rating_denominator    0
         name                  0
         dog_stage             0
         retweet_count         0
         favorite_count        0
         jpg_url               0
         dog                   0
         conf                  0
         dtype: int64
```

Test

```
In [62]: #proove that there is no duplicate jpg anymore.
         df_master[df_master.duplicated(subset=['jpg_url'], keep='first')]
```

```
Out[62]: Empty DataFrame
         Columns: [tweet_id, timestamp, source, text, expanded_urls, rating_numerator, rating_de
         Index: []
```

Define According to wikipedia (https://en.wikipedia.org/wiki/WeRateDogs) the owner of the site is student in set timezone to EST according to wikipedia that's where the twitter account owner studies at Campbell University in Buies Creek, North Carolina. So we will have to **change the creation of tweets to local EST timezone**.

Code

```
In [63]: #convert object to datetime64
         #https://pandas.pydata.org/pandas-docs/stable/timeseries.html
         #localize to UTC and convert to EST
         #https://stackoverflow.com/questions/20689288/converting-pandas-columns-to-datetime64-i
         #df_master['timestamp'] =
         df_master.timestamp = pd.to_datetime(df_master.timestamp).dt.tz_localize('UTC').dt.tz_c
         #extract local hour
         df_master.create_HH24 = df_master.timestamp.dt.hour

In [64]: df_master['create_HH24'] = df_master.timestamp.dt.hour.astype('category')
```

Test

```
In [65]: df_master.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1664 entries, 0 to 1663
Data columns (total 15 columns):
tweet_id              1664 non-null int64
timestamp             1664 non-null datetime64[ns, EST]
source                1664 non-null category
text                  1664 non-null object
expanded_urls         1664 non-null object
rating_numerator      1664 non-null int64
rating_denominator    1664 non-null int64
name                  1203 non-null object
dog_stage             249 non-null category
retweet_count         1664 non-null int64
favorite_count        1664 non-null int64
jpg_url               1664 non-null object
dog                   1664 non-null category
conf                  1664 non-null float64
create_HH24           1664 non-null category
dtypes: category(4), datetime64[ns, EST](1), float64(1), int64(5), object(4)
memory usage: 169.5+ KB
```

Finally we store the intermediate result into a file for further analysis.

```
In [66]: df_master.to_csv('twitter_archive_master.csv',index=False, encoding='utf-8')
```