wrangle_act

May 13, 2018

1 Project Wrangle and Analyze Data

1.0.1 *Author: Anthony T. O'Brien M.D.*

12th May 2018

2 Context

The context of this project is to wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations.

This dataset 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. WeRateDogs has over 4 million followers and has received international media coverage. More information on WeRateDogs can be found here.

3 Data

The WeRateDogs Twitter archive contains basic tweet data for 5000+ tweets. The data contains information on the tweet id, the timestamp of the tweet, the source of the tweet, the retweet status of the tweet, the rating in of the dog in the tweet in a ratio format (i.e. nominator and denominator), the name of the dog, and the dog's stage according to WeRateDogs criteria. Additionally we have access to data from Udacity which used neural networks to predict the type of dog in the tweets which posted jpegs. Finally we also have access to the number of retweets and favorites via the twitter API tweepy.

4 Abstract

We gather, assess and clean the above mentioned data and finally merge these data into one file. The file is then used to produce visualizations and insight into the data's contents.

5 Introduction

In this project, we go through the process of wrangling and analyzing data obtained from WeRateDogs. The data is derived from three sources, a .csv file, a url and from twitter's API. Data wrangling is a fundamental skill and involves the processes of gathering, assessing and cleaning data prior to any directed analysis. Data wrangling turns low quality and untidy data, into high quality and tidy (i.e. useful) data. By the end of this project these steps will be completed and then the data will be transfomred into meaningful insights.

6 Libraries to be installed

```
In [142]: import pandas as pd
          import numpy as np
          import requests
          import os
          import tweepy
          import json
          import io
          import matplotlib.pyplot as plt
          import plotly.plotly as py
          import seaborn as sns
          %matplotlib inline
          sns.set(style="ticks", color_codes=True)
```

7 Authentication

```
In [2]: #Please note that keys were removed prior to submission of project
        consumer_key = 'not_included_in_submitted_project'
        consumer_secret = 'not_included_in_submitted_project'
        access_token = 'not_included_in_submitted_project'
        access_secret = 'not_included_in_submitted_project'

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

        api = tweepy.API(auth)
```

8 1. Gathering data

```
In [3]: #1. Import csv for WeRateDogs twitter data provided by Udacity into a dataframe
        df_twitter = pd.read_csv("twitter-archive-enhanced.csv")
```

```
In [4]: #Import tsv for image prediction by Udacity

        #create folder to store tsv
        folder_name = "dogs_prediction"
        if not os.path.exists(folder_name):
            os.makedirs(folder_name)

        #use request to get tsv from url
        url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictio
```

```
        r = requests.get(url)

        #save tsv stored in jupyter notebook working memory into dogs_prediction folder
        with open(os.path.join(folder_name,
                               url.split('/')[-1]), mode='wb') as file:
            file.write(r.content)
```

```
In [5]: #2. Import prediction tsv into a dataframe
        df_predict = pd.read_csv("/home/workspace/dogs_prediction/image-predictions.tsv", sep="\
```

```
In [6]: #3. Create json file, convert to list, and load into dataframe
        deleted_tweets= []
        with open(os.path.join(os.getcwd(), 'tweet_json.txt'), mode = 'w') as file:
            for twt_id in df_twitter["tweet_id"]:
                try:
                    tweet = api.get_status(twt_id, tweet_mode = 'extended', wait_on_rate_limit=T
                    twt_sjson = json.dumps(tweet._json)
                    file.write(twt_sjson + '\n')
                except:
                    print('Tweet id was not found')
                    deleted_tweets.append(twt_id)

        print("Finished dumping tweets")

        #Append json file into a list so that I can extract relevant data
        tweets = []
        for line in open('tweet_json.txt', 'r'):
            tweets.append(json.loads(line))
```

```
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Tweet id was not found
Rate limit reached. Sleeping for: 722
Rate limit reached. Sleeping for: 724
Finished dumping tweets
```

```
In [8]: # Using the tweets list, I extract the id, retweet_count, and favorite_count
        # into a dataframe
        df_api = pd.read_json('tweet_json.txt', lines=True)[['id', 'retweet_count', 'favorite_co
```

9 2. Assesing Data

As a continuation of the previous section, I save the data frame (df_api) into a .csv so that in the future it will be easy to call upon without the need to re-run the tweepy API.

In [10]:
```
# Store the df_api dataframe into a .csv file so when I close project
# and reopen I can call upon the saved file instead of re-run the api
# which is time consuming. This only has to be run once then the following
# cell can be run to load the .csv into a df. Therefore I will "# it out"
# df_api.to_csv('twitter_api.csv', index=None)
```

In [11]:
```
# Once the API is run there is no need to run API again, for efficency.
# Therefore I load the data into df_api to save time
df_api= pd.read_csv('twitter_api.csv')
```

9.0.1 1. Visual assesment

In [12]:
```
#4. I visually assess the dataframes individually starting with the last
# data frame (df_api) created since it is the simplest
df_api.head(3)
```

Out[12]:

	id	retweet_count	favorite_count
0	892420643555336193	8610	38858
1	892177421306343426	6324	33280
2	891815181378084864	4195	25076

In [13]:
```
# Random sampling of dataframe
df_api.sample(5)
```

Out[13]:

	id	retweet_count	favorite_count
204	852226086759018497	7342	20950
857	761750502866649088	4402	0
845	764259802650378240	1697	6581
196	853760880890318849	6205	29829
1061	739979191639244800	6518	21398

Tidiness

- 1) Column name id does not match other data frames (should be tweet_id)

9.0.2 2. Visual assesment

In [14]:
```
# Now I visually assess the first data frame (df_twitter)
# Please note this was also done in excel according to class notes
df_twitter.tail(5)
```

Out[14]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id \
2351	666049248165822465	NaN	NaN
2352	666044226329800704	NaN	NaN

```
2353  666033412701032449                    NaN                  NaN
2354  666029285002620928                    NaN                  NaN
2355  666020888022790149                    NaN                  NaN

                      timestamp  \
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  \
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  \
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  \
2351                       NaN                        NaN
2352                       NaN                        NaN
2353                       NaN                        NaN
2354                       NaN                        NaN
2355                       NaN                        NaN

                                          expanded_urls  rating_numerator  \
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
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
```

In [16]:
```
# Random sampling of dataframe
df_twitter.sample(5)
```

```
Out[16]:                 tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
        1665  682750546109968385                    NaN                  NaN
        85    876120275196170240                    NaN                  NaN
        2072  671109016219725825                    NaN                  NaN
        424   821522889702862852                    NaN                  NaN
        1735  679729593985699840                    NaN                  NaN

                              timestamp  \
        1665  2016-01-01 02:29:49 +0000
        85    2017-06-17 16:52:05 +0000
        2072  2015-11-29 23:30:32 +0000
        424   2017-01-18 01:01:34 +0000
        1735  2015-12-23 18:25:38 +0000

                                                          source  \
        1665  <a href="http://twitter.com/download/iphone" r...
        85    <a href="http://twitter.com/download/iphone" r...
        2072  <a href="http://twitter.com/download/iphone" r...
        424   <a href="http://twitter.com/download/iphone" r...
        1735  <a href="http://twitter.com/download/iphone" r...

                                                            text  retweeted_status_id  \
        1665  Meet Taco. He's a speckled Garnier Fructis. Lo...                  NaN
        85    Meet Venti, a seemingly caffeinated puppoccino...                  NaN
        2072  This is Toby. He asked for chocolate cake for ...                  NaN
        424   This is Harlso. He has a really good idea but ...                  NaN
        1735  This is Hunter. He was playing with his ball m...                  NaN

              retweeted_status_user_id retweeted_status_timestamp  \
        1665                       NaN                        NaN
        85                         NaN                        NaN
        2072                       NaN                        NaN
        424                        NaN                        NaN
        1735                       NaN                        NaN

                                                   expanded_urls  rating_numerator  \
        1665  https://twitter.com/dog_rates/status/682750546...                 9
        85    https://twitter.com/dog_rates/status/876120275...                13
        2072  https://twitter.com/dog_rates/status/671109016...                 8
        424   https://twitter.com/dog_rates/status/821522889...                13
        1735  https://twitter.com/dog_rates/status/679729593...                 8

              rating_denominator    name doggo floofer pupper puppo
        1665                  10    Taco  None    None   None  None
        85                    10   Venti  None    None   None  None
        2072                  10    Toby  None    None   None  None
        424                   10  Harlso  None    None   None  None
        1735                  10  Hunter  None    None   None  None
```

Quality

- 1) Not all tweets are original (i.e. they are retweets).
- 2) Not all original tweets have dog ratings.
- 3) Not all original tweets have an image associated.
- 4) Some tweets are in reply to another tweet.
- 5) The source column url format is not useful.
- 6) The datetime column format is not useful.
- 7) The denominator on the rating_denominator is inconsistent.

Tidiness

- 2) Dog stages are not in one column (ie. not in tidy format)

9.0.3 3. Visual assesment

In [17]:
```
# Now I visually assess the second data frame (df_predict)
# Please note this was also done in excel according to class notes
df_predict.head(5)
```

Out[17]:
```
             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

   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

    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
```

In [18]:
```
# Random sampling of dataframe
df_predict.sample(5)
```

```
Out[18]:                  tweet_id                                            jpg_url  \
         977   707038192327901184    https://pbs.twimg.com/media/Cc_ney1W4AANuY3.jpg
         1858  841833993020538882  https://pbs.twimg.com/ext_tw_video_thumb/81742...
         1095  720043174954147842    https://pbs.twimg.com/media/Cf4bcm8XEAAX4xV.jpg
         244   670465786746662913    https://pbs.twimg.com/media/CU35E7VWEAAKYBy.jpg
         551   677557565589463040    https://pbs.twimg.com/media/CWcrAVQWEAA6QMp.jpg

               img_num         p1   p1_conf  p1_dog           p2   p2_conf  p2_dog  \
         977         1        pug  0.642426    True        llama  0.057306   False
         1858        1   ice_bear  0.336200   False      Samoyed  0.201358    True
         1095        1    Samoyed  0.954517    True   Eskimo_dog  0.029130    True
         244         1    axolotl  0.611558   False  tailed_frog  0.186484   False
         551         1  seat_belt  0.277257   False     Shih-Tzu  0.249017    True

                           p3   p3_conf  p3_dog
         977   French_bulldog  0.054186    True
         1858      Eskimo_dog  0.186789    True
         1095      white_wolf  0.004462   False
         244      common_newt  0.078694   False
         551         Pekinese  0.209213    True
```

Quality

- 8) Not all predictions correspond to a dog breed.
- 9) Not all the predictions are written uniformly, some are capitalized others not.

9.0.4 1. Programatic assesment

```
In [19]: # At this stage I also want to programatically review the data. I assess the data frame
         #First I review the data frame structure of df_api, and check for duplicates
         df_api.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2345 entries, 0 to 2344
Data columns (total 3 columns):
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 [20]: #Checking for duplicates in df_api
         sum(df_api.duplicated())
```

```
Out[20]: 0
```

1. Programatic assesment observations We observe that there are 2345 entries, that they are in integer format and there are no duplicate rows.

9.0.5 2. Programatic assesment

```
In [21]: # We check the structure and properties of df_twitter
         df_twitter.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 [58]: #Checking for duplicates in df_twitter
         sum(df_twitter.duplicated())
```

```
Out[58]: 0
```

```
In [57]: # Programatically confirming that the denominator variable is inconsistent.
         df_twitter['rating_denominator'].describe()
```

```
Out[57]: 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
```

2. Programatic assesment observations There are 2356 entries in df_twitter (more than df_api), also the data types are varied: 4 floats, 3 integers and 10 objects.

Quality

- 10) The timestamp variable is not in datetime format.
- 11) The dog rating in categries (i.e. doggo, floofer etc.) is an object instead of string.

9.0.6 3. Programatic assesment

```
In [24]: # We check the structure and properties of df_predict
         df_predict.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 [53]: #Checking for duplicates in df_predict
         sum(df_twitter.duplicated())
```

```
Out[53]: 0
```

3. Programatic assesment observations There are considerably less entries in df_twitter (n=2075). In ascending order of entries the data frames can be ordered as follows (smallest to biggest): df_twitter (n=2078) < df_api (n=2345) < df_twitter (n=2356). In terms of the data format there are 3 boolean formats, 3 float, 2 integers, and 4 objects. Of interest are the boolean formats (true/false) which will be leveraged to select the corresponding(i.e. True) dog breed to the jpg_url.

Quality

- 13) The column img_number is not of interest.

Tidiness

- 3) After merging the number of rows will have to be adjusted

10 Summary of data assesment

Quality In df_twitter: * 1) Not all tweets are original (i.e. they are retweets). * 2) Not all original tweets have dog ratings. * 3) Some tweets are in reply to another tweet. * 4) The source column url format is not useful. * 5) The denominator on the rating_denominator is inconsistent (mainly 10 but sometimes other numbers like 7). * 6) The timestamp variable is not in datetime format. * 7) The dog rating in categries (i.e. doggo, floofer etc.) is an object instead of string.

In df_predictions: * 10) Not all predictions correspond to a dog breed. * 11) Not all the predictions are written uniformly, some are capitalized others not. * 12) Not all original tweets have an image associated.

Tidiness In df_api: * 1) Column name id does not match other data frames (should be tweet_id)

In df_twitter: * 2) Dog stages are not in one column (ie. not in tidy format)

In final merge: * 3) After merging the number of rows will have to be adjusted since the dataframes are different dimensions

11 3. Cleaning Data - Part 1

1. Define

- 1) I am starting with the easiest scenario to clean:
- 2) Rename the column name id to tweet_id so it matched other dataframes

Code

```
In [22]: # Rename the column name id to tweet_id so it matched other dataframes
         df_api = df_api.rename(columns={'id': 'tweet_id'})
```

Test

```
In [23]: df_api.head(1)

Out[23]:              tweet_id  retweet_count  favorite_count
         0  892420643555336193           8610           38858
```

Create a copy of the dataframes At this stage, to prevent losing my original dataframes and having to rerun python, I create a copy of all the dataframes to work on.

```
In [24]: df_twitter_v2 = df_twitter.copy()

In [25]: df_predict_v2 = df_predict.copy()

In [26]: df_api_v2 = df_api.copy()
```

2. Define

- 1) Some tweets are in reply to another tweet.
- 2) Not all tweets are original (i.e. they are retweets).

Code

```
In [27]: # 1) Retain only rows with NaN in the 'in_reply_to_status_id' to remove tweets
         # which are in reply to another tweet
         df_twitter_v2 = df_twitter_v2[pd.isnull(df_twitter_v2['in_reply_to_status_id'])]


         # 2) Retain only row with NaN in the 'retweeted_status_id' to remove tweets
         # which are retweets
         df_twitter_v2 = df_twitter_v2[pd.isnull(df_twitter_v2['retweeted_status_id'])]
```

Test

```
In [28]: df_twitter_v2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2097 non-null int64
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2097 non-null object
source                        2097 non-null object
text                          2097 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                 2094 non-null object
rating_numerator              2097 non-null int64
rating_denominator            2097 non-null int64
name                          2097 non-null object
doggo                         2097 non-null object
floofer                       2097 non-null object
pupper                        2097 non-null object
puppo                         2097 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 294.9+ KB
```

3. Define Now that I have selected only the rows with original tweets I drop the following columns to clean the dataframe:

- 1) in_reply_to_status_id
- 2) in_reply_to_user_id
- 3) retweeted_status_id
- 4) retweeted_status_user_id
- 5) retweeted_status_timestamp

Code

```
In [29]: #Drop columns which do not provide any useful information
         df_twitter_v2 = df_twitter_v2.drop(['in_reply_to_status_id',
                         'in_reply_to_user_id',
                         'in_reply_to_user_id',
                         'retweeted_status_user_id',
                         'retweeted_status_timestamp',
                         'retweeted_status_id',
                         'expanded_urls',
                         'text',
                         'name'], axis=1);
```

Test

```
In [30]: df_twitter_v2.sample(2)

Out[30]:                 tweet_id                   timestamp  \
         783   775350846108426240  2016-09-12 15:10:21 +0000
         1746  679132435750195208  2015-12-22 02:52:45 +0000

                                                          source  rating_numerator  \
         783   <a href="http://vine.co" rel="nofollow">Vine -...                12
         1746  <a href="http://twitter.com/download/iphone" r...                10

               rating_denominator doggo floofer pupper puppo
         783                   10  None    None   None  None
         1746                  10  None    None   None  None
```

4. Define

- 1) I will standardize the rating_denominator column to only 10

Code

```
In [31]: #Replace any denominator that is not 10 with the value 10
         df_twitter_v2.loc[df_twitter_v2.rating_denominator != 10, 'rating_denominator'] = 10
```

Test

```
In [32]: df_twitter_v2['rating_denominator'].describe()

Out[32]: count    2097.0
         mean       10.0
         std         0.0
         min        10.0
         25%        10.0
         50%        10.0
         75%        10.0
         max        10.0
         Name: rating_denominator, dtype: float64
```

5. Define I will take the source column and categorize it. To do so first I want to know how many unique values are in the column and what are the values.

Code

```
In [33]: #Altough this is part of assesment I am iterating the Assesing code section here
         # to figure out how to categorize the source column
         df_twitter_v2['source'].unique()

Out[33]: array([ '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone
                '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
                '<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>',
                '<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck

In [34]: #Replace the above array as iphone, web, vine, and tweetdeck
         df_twitter_v2['source'] = df_twitter_v2['source'].replace({'<a href="http://twitter.com
                                                   '<a href="http://twitter.com" rel="nofollow">Twitter
                                                   '<a href="http://vine.co" rel="nofollow">Vine - Make
                                                   '<a href="https://about.twitter.com/products/tweetdec
```

Test

```
In [35]: df_twitter_v2.sample(5)

Out[35]:                 tweet_id                  timestamp source  rating_numerator  \
         497   813142292504645637  2016-12-25 22:00:04 +0000 iphone                13
         1280  708834316713893888  2016-03-13 01:57:25 +0000 iphone                10
         1391  700143752053182464  2016-02-18 02:24:13 +0000 iphone                10
         1672  682389078323662849  2015-12-31 02:33:29 +0000 iphone                 9
         1059  741743634094141440  2016-06-11 21:27:17 +0000 iphone                11

               rating_denominator doggo floofer  pupper puppo
         497                   10  None    None    None  None
         1280                  10  None    None    None  None
         1391                  10  None    None  pupper  None
         1672                  10  None    None    None  None
         1059                  10  None    None  pupper  None
```

6. Define

- 1) Convert timestamp into datetime for later use

Code - change timestamp format to datetime

```
In [36]: #From out assesment we know that timestamp is not in datetime format.
         #Therefore we convert to datetime format with pd.to_datetime
         df_twitter_v2['timestamp'] = pd.to_datetime(df_twitter_v2['timestamp'])
```

Test

```
In [37]: df_twitter_v2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 9 columns):
tweet_id              2097 non-null int64
timestamp             2097 non-null datetime64[ns]
source                2097 non-null object
rating_numerator      2097 non-null int64
rating_denominator    2097 non-null int64
doggo                 2097 non-null object
floofer               2097 non-null object
pupper                2097 non-null object
puppo                 2097 non-null object
dtypes: datetime64[ns](1), int64(3), object(5)
memory usage: 163.8+ KB
```

7. Define Create a ratio for the rating numerator and denominator. Due to the way WeRateDogs works I only standardized the denominator. I will not modify the numererator.

Code

```
In [38]: df_twitter_v2['score_ratio']= df_twitter_v2['rating_numerator']/df_twitter_v2['rating_d
```

Test

```
In [39]: df_twitter_v2.sample(2)
```

```
Out[39]:                  tweet_id           timestamp  source  rating_numerator  \
         1136  728387165835677696 2016-05-06 00:53:27  iphone                12
         190   855857698524602368 2017-04-22 18:55:51  iphone                13

               rating_denominator doggo floofer pupper puppo  score_ratio
         1136                  10  None    None   None  None          1.2
         190                   10  None    None   None  None          1.3
```

8. Define Convert the dog stages columns (i.e.doggo, floofer, pupper and puppo) into one column named dog_stages. To do this I will convert None to NaN and then combine the four columns into one by exlcuding any NaN values.

Code

```
In [40]: #The first 4 lines of this code is to replace in the None string in the
         #individual dog stage columns with NaN
         df_twitter_v2['doggo'].replace('None', np.nan, inplace=True)
```

```
df_twitter_v2['floofer'].replace('None', np.nan, inplace=True)
df_twitter_v2['pupper'].replace('None', np.nan, inplace=True)
df_twitter_v2['puppo'].replace('None', np.nan, inplace=True)

#The next line of code combines the dog stage columns and drops any NaN
df_twitter_v2['dog_stages'] = df_twitter_v2[['doggo','floofer','pupper', 'puppo']].fill

#The last line of code replaces any blank cells with the term unstaged
#which is intentional to not confuse with rating. Unstaged means that
#the dog did not get classified into a dog stage
df_twitter_v2['dog_stages'].replace('', 'unstaged', inplace=True)

#Then I drop the original columns because they are not needed
df_twitter_v2 = df_twitter_v2.drop(['doggo', 'floofer','pupper','puppo',],
                                    axis=1);
```

Test

```
In [41]: df_twitter_v2.sample(3)
```

```
Out[41]:                tweet_id           timestamp  source  rating_numerator  \
         1654  683449695444799489 2016-01-03 00:47:59  iphone                10
         945   752660715232722944 2016-07-12 00:27:52  iphone                10
         579   800513324630806528 2016-11-21 01:37:04  iphone                11

               rating_denominator  score_ratio dog_stages
         1654                  10          1.0   unstaged
         945                   10          1.0      doggo
         579                   10          1.1   unstaged
```

```
In [42]: df_twitter_v2['dog_stages'].unique()
```

```
Out[42]: array(['unstaged', 'doggo', 'puppo', 'pupper', 'floofer', 'doggopuppo',
                'doggofloofer', 'doggopupper'], dtype=object)
```

Observation (important note) There are 8 categories, in which the original exist but also there are combined categories eg. doggofloofer. It is not clear if the categories can be combined according to WeRateDogs. However this can be easily remediated by using the .replace function previously used. I will not do that because I am not sure if the combination of stages was intentional or an error in the data.

9. Define I am going to do some final cleaning up of the df_twitter_v2 data frame before copying it as v3. The final touch ups will be to drop the denominator and to change the rating_denominator column to score

Code

```
In [43]: #Drop rating denominator column
         df_twitter_v2 = df_twitter_v2.drop(['rating_denominator'], axis=1);

         #Rename column
         df_twitter_v2.rename(columns={'rating_numerator':'score'},inplace=True)
```

Test

```
In [44]: df_twitter_v2.sample(4)

Out[44]:                  tweet_id           timestamp  source  score  score_ratio  \
         626   795076730285391872 2016-11-06 01:33:58  iphone     11          1.1
         157   861288531465048066 2017-05-07 18:36:02  iphone     13          1.3
         2288  667176164155375616 2015-11-19 03:02:47  iphone      4          0.4
         454   818536468981415936 2017-01-09 19:14:36  iphone     11          1.1

              dog_stages
         626    unstaged
         157    unstaged
         2288   unstaged
         454    unstaged
```

12 3. Cleaning Data - Part 2

10. Define Not all predictions correspond to a dog breed. Therefore, the next code will have to: 1) use the prediciotn column boolean term True/False to take the name prediction and place it in a column. For example if the boolean prediciton is True the code will take the name column and place that name into another column. If it is False it will continue to the next prediction and so on.

Code

```
In [45]: #Create a function to run conditional statements to select the best prediction from the
         # best dog breed prediction and place into a separate column
         row=df_predict_v2

         def dog_breed (row):
             if row['p1_dog'] == True:
                 return row["p1"]
             elif row['p2_dog'] == True:
                 return row["p2"]
             elif row['p3_dog'] == True:
                 return row["p3"]
             else:
                 return "unpredictable"
         df_predict_v2['best_dog_prediction'] = df_predict_v2.apply (lambda row: dog_breed (row)


         def dog_conf (row):
```

```
    if row['p1_dog'] == True:
        return row["p1_conf"]
    elif row['p2_dog'] == True:
        return row["p2_conf"]
    elif row['p3_dog'] == True:
        return row["p3_conf"]
    else:
        return "NaN"
df_predict_v2['best_dog_pred_acc'] = df_predict_v2.apply (lambda row: dog_conf (row),ax
```

Test

```
In [46]: df_predict_v2.sample(5)
```

```
Out[46]:             tweet_id                                          jpg_url  \
         1050  713900603437621249  https://pbs.twimg.com/media/CehIzzZWQAEyHH5.jpg
         234   670421925039075328  https://pbs.twimg.com/media/CU3RLqfW4AE0pbA.jpg
         2065  890240255349198849  https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg
         616   680191257256136705  https://pbs.twimg.com/media/CXCGVXyWsAAAVHE.jpg
         1178  737826014890496000  https://pbs.twimg.com/media/Cj1I1fbWYAAOwff.jpg

               img_num                p1   p1_conf  p1_dog                      p2  \
         1050        1  golden_retriever  0.371816    True           cocker_spaniel
         234         1         Chihuahua  0.275793    True                     corn
         2065        1          Pembroke  0.511319    True                 Cardigan
         616         1  Brittany_spaniel  0.733253    True  Welsh_springer_spaniel
         1178        1            vizsla  0.990391    True      Rhodesian_ridgeback

                p2_conf  p2_dog                        p3   p3_conf  p3_dog  \
         1050  0.177413    True              Irish_setter  0.092725    True
         234   0.073596   False                    bolete  0.054905   False
         2065  0.451038    True                 Chihuahua  0.029248    True
         616   0.251634    True          English_springer  0.009243    True
         1178  0.005605    True  Chesapeake_Bay_retriever  0.002869    True

              best_dog_prediction best_dog_pred_acc
         1050    golden_retriever          0.371816
         234            Chihuahua          0.275793
         2065            Pembroke          0.511319
         616     Brittany_spaniel          0.733253
         1178              vizsla          0.990391
```

Note I remove the prediction columns and the img_number columns since I condier them untidy. Especially now that we have the best_prediction column

```
In [47]: df_predict_v2 = df_predict_v2.drop(['p1',
                         'p1_conf',
                         'p1_dog',
```

```
                        'p2',
                        'p2_conf',
                        'p2_dog',
                        'p3',
                        'p3_conf',
                        'p3_dog',
                        'img_num'], axis=1);
```

In [48]: df_predict_v2.sample(5)

```
Out[48]:                tweet_id                                           jpg_url  \
         158   668872652652679168  https://pbs.twimg.com/media/CUhQIAhXAAA2j7u.jpg
         144   668623201287675904  https://pbs.twimg.com/media/CUdtP1xUYAIeBnE.jpg
         1264  749064354620928000  https://pbs.twimg.com/media/CmU2DVWWgAArvp3.jpg
         304   671518598289059840  https://pbs.twimg.com/media/CVG2l9jUYAAwg-w.jpg
         314   671729906628341761  https://pbs.twimg.com/media/CVJ2yR2UwAAdCzU.jpg

               best_dog_prediction best_dog_pred_acc
         158   miniature_schnauzer         0.0355366
         144             Chihuahua          0.708163
         1264                  pug          0.985222
         304       Lakeland_terrier          0.428275
         314                kuvasz          0.431469
```

13 3. Cleaning Data - Part 3

I will make copies of my newly cleaned dataframes to then use for merging.

```
In [49]: df_twitter_v3 = df_twitter_v2.copy()

In [50]: df_predict_v3 = df_predict_v2.copy()

In [51]: df_api_v3 = df_api_v2.copy()
```

Then I run a quick sample of the new copies to as once last check before merging

```
In [52]: df_twitter_v3.sample()

Out[52]:                tweet_id            timestamp  source  score  score_ratio  \
         1130  729113531270991872 2016-05-08 00:59:46  iphone     10          1.0

              dog_stages
         1130   unstaged

In [53]: df_predict_v3.sample()

Out[53]:                tweet_id                                           jpg_url  \
         1222  744334592493166593  https://pbs.twimg.com/media/ClRoXGwWIAEVVzc.jpg

              best_dog_prediction best_dog_pred_acc
         1222             Samoyed          0.960543
```

```
In [54]: df_api_v3.sample()
```

```
Out[54]:                  tweet_id  retweet_count  favorite_count
         1001  747242308580548608           3180               0
```

11. Define The three dataframes all have valuable information and it is easier to merge them into the same dataframe for further analysis.

Code

```
In [55]: # Code to merge data
         df_merge1 = df_twitter_v3.merge(df_predict_v3, how='left', left_on=["tweet_id"], right_
         df_twitter_master= df_merge1.merge(df_api_v3, left_on=["tweet_id"], right_on=["tweet_id
```

Test

```
In [59]: # Quick test to see if data frames merged correctly
         df_merge1.sample(3)
```

```
Out[59]:                tweet_id           timestamp  source  score  score_ratio  \
         364  817056546584727552 2017-01-05 17:13:55  iphone     11          1.1
         983  716730379797970944 2016-04-03 20:53:33  iphone     12          1.2
         105  869227993411051520 2017-05-29 16:24:37  iphone     13          1.3

             dog_stages                                          jpg_url  \
         364   unstaged  https://pbs.twimg.com/media/C1bEl4zVIAASj7_.jpg
         983   unstaged                                              NaN
         105   unstaged  https://pbs.twimg.com/media/DBAePiVXcAAqHSR.jpg

             best_dog_prediction best_dog_pred_acc
         364              kelpie          0.864415
         983                 NaN               NaN
         105            Pembroke          0.664181
```

```
In [60]: # Quick test to see if data frames merged correctly
         df_twitter_master.sample(3)
```

```
Out[60]:                 tweet_id           timestamp  source  score  score_ratio  \
         1445  681891461017812993 2015-12-29 17:36:07  iphone     10          1.0
         1430  682662431982772225 2015-12-31 20:39:41  iphone     11          1.1
         74    876484053909872640 2017-06-18 16:57:37  iphone     13          1.3

              dog_stages                                          jpg_url  \
         1445     pupper  https://pbs.twimg.com/media/CXaQqGbWMAAKEgN.jpg
         1430   unstaged  https://pbs.twimg.com/media/CX1N1-EWMAQdwXK.jpg
         74     unstaged  https://pbs.twimg.com/media/DCnll_dUQAAkBdG.jpg

              best_dog_prediction best_dog_pred_acc  retweet_count  favorite_count
         1445           Chihuahua           0.20357            917            2646
         1430              beagle          0.413824           1179            3264
         74      golden_retriever          0.874566           2434           18821
```

Note I store df_twitter_master into a file so that I can easily access later, and also so that I can prep for storing into a master archive

```
In [61]: df_twitter_master.to_csv('twitter_master.csv', index=None)
         df_twitter_master = pd.read_csv('twitter_master.csv')
```

14 4. Iterate Assesing and Cleaning Data

14.0.1 4. Visual and programatic assesment

```
In [65]: df_twitter_master.head(100)
```

```
Out[65]:               tweet_id            timestamp  source  score  score_ratio  \
         0   892420643555336193  2017-08-01 16:23:56  iphone     13          1.3
         1   892177421306343426  2017-08-01 00:17:27  iphone     13          1.3
         2   891815181378084864  2017-07-31 00:18:03  iphone     12          1.2
         3   891689557279858688  2017-07-30 15:58:51  iphone     13          1.3
         4   891327558926688256  2017-07-29 16:00:24  iphone     12          1.2
         5   891087950875897856  2017-07-29 00:08:17  iphone     13          1.3
         6   890971913173991426  2017-07-28 16:27:12  iphone     13          1.3
         7   890729181411237888  2017-07-28 00:22:40  iphone     13          1.3
         8   890609185150312448  2017-07-27 16:25:51  iphone     13          1.3
         9   890240255349198849  2017-07-26 15:59:51  iphone     14          1.4
         10  890006608113172480  2017-07-26 00:31:25  iphone     13          1.3
         11  889880896479866881  2017-07-25 16:11:53  iphone     13          1.3
         12  889665388333682689  2017-07-25 01:55:32  iphone     13          1.3
         13  889638837579907072  2017-07-25 00:10:02  iphone     12          1.2
         14  889531135344209921  2017-07-24 17:02:04  iphone     13          1.3
         15  889278841981685760  2017-07-24 00:19:32  iphone     13          1.3
         16  888917238123831296  2017-07-23 00:22:39  iphone     12          1.2
         17  888804989199671297  2017-07-22 16:56:37  iphone     13          1.3
         18  888554962724278272  2017-07-22 00:23:06  iphone     13          1.3
         19  888078434458587136  2017-07-20 16:49:33  iphone     12          1.2
         20  887705289381826560  2017-07-19 16:06:48  iphone     13          1.3
         21  887517139158093824  2017-07-19 03:39:09  iphone     14          1.4
         22  887473957103951883  2017-07-19 00:47:34  iphone     13          1.3
         23  887343217045368832  2017-07-18 16:08:03  iphone     13          1.3
         24  887101392804085760  2017-07-18 00:07:08  iphone     12          1.2
         25  886983233522544640  2017-07-17 16:17:36  iphone     13          1.3
         26  886736880519319552  2017-07-16 23:58:41  iphone     13          1.3
         27  886680336477933568  2017-07-16 20:14:00  iphone     13          1.3
         28  886366144734445568  2017-07-15 23:25:31  iphone     12          1.2
         29  886258384151887873  2017-07-15 16:17:19  iphone     13          1.3
         ..                 ...                  ...     ...    ...          ...
         70  877316821321428993  2017-06-21 00:06:44  iphone     13          1.3
         71  877201837425926144  2017-06-20 16:29:50  iphone     12          1.2
         72  876838120628539392  2017-06-19 16:24:33  iphone     12          1.2
         73  876537666061221889  2017-06-18 20:30:39  iphone     14          1.4
```

74	876484053909872640	2017-06-18 16:57:37	iphone	13	1.3
75	876120275196170240	2017-06-17 16:52:05	iphone	13	1.3
76	875747767867523072	2017-06-16 16:11:53	iphone	13	1.3
77	875144289856114688	2017-06-15 00:13:52	iphone	13	1.3
78	875097192612077568	2017-06-14 21:06:43	iphone	13	1.3
79	875021211251597312	2017-06-14 16:04:48	iphone	12	1.2
80	874680097055178752	2017-06-13 17:29:20	iphone	12	1.2
81	874296783580663808	2017-06-12 16:06:11	iphone	13	1.3
82	874057562936811520	2017-06-12 00:15:36	iphone	12	1.2
83	874012996292530176	2017-06-11 21:18:31	iphone	13	1.3
84	873580283840344065	2017-06-10 16:39:04	iphone	13	1.3
85	873213775632977920	2017-06-09 16:22:42	iphone	12	1.2
86	872967104147763200	2017-06-09 00:02:31	iphone	12	1.2
87	872820683541237760	2017-06-08 14:20:41	iphone	13	1.3
88	872620804844003328	2017-06-08 01:06:27	iphone	13	1.3
89	872486979161796608	2017-06-07 16:14:40	iphone	12	1.2
90	872261713294495745	2017-06-07 01:19:32	iphone	13	1.3
91	872122724285648897	2017-06-06 16:07:15	iphone	12	1.2
92	871879754684805121	2017-06-06 00:01:46	iphone	13	1.3
93	871762521631449091	2017-06-05 16:15:56	iphone	12	1.2
94	871515927908634625	2017-06-04 23:56:03	iphone	12	1.2
95	871102520638267392	2017-06-03 20:33:19	iphone	14	1.4
96	871032628920680449	2017-06-03 15:55:36	iphone	13	1.3
97	870804317367881728	2017-06-03 00:48:22	iphone	11	1.1
98	870656317836468226	2017-06-02 15:00:16	iphone	13	1.3
99	870374049280663552	2017-06-01 20:18:38	iphone	13	1.3

	dog_stages	jpg_url	\
0	unstaged	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	
1	unstaged	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	
2	unstaged	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	
3	unstaged	https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg	
4	unstaged	https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg	
5	unstaged	https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg	
6	unstaged	https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg	
7	unstaged	https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg	
8	unstaged	https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg	
9	doggo	https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg	
10	unstaged	https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg	
11	unstaged	https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg	
12	puppo	https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg	
13	unstaged	https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg	
14	puppo	https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg	
15	unstaged	https://pbs.twimg.com/ext_tw_video_thumb/88927...	
16	unstaged	https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg	
17	unstaged	https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg	
18	unstaged	https://pbs.twimg.com/media/DFTH_O-UQAACu2O.jpg	
19	unstaged	https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg	

```
20   unstaged    https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg
21   unstaged  https://pbs.twimg.com/ext_tw_video_thumb/88751...
22   unstaged    https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg
23   unstaged  https://pbs.twimg.com/ext_tw_video_thumb/88734...
24   unstaged    https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg
25   unstaged    https://pbs.twimg.com/media/DE8yicJWOAAAvBJ.jpg
26   unstaged    https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg
27   unstaged    https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg
28     pupper    https://pbs.twimg.com/media/DEOBTnQUwAApKEH.jpg
29   unstaged    https://pbs.twimg.com/media/DEyfTG4UMAE4aE9.jpg
..        ...                                                ...
70   unstaged    https://pbs.twimg.com/media/DCza_vtXkAQXGpC.jpg
71   unstaged    https://pbs.twimg.com/media/DCxyahJWsAAddSC.jpg
72     pupper    https://pbs.twimg.com/media/DCsnnZsVwAEfkyi.jpg
73   unstaged                                                NaN
74   unstaged    https://pbs.twimg.com/media/DCnll_dUQAAkBdG.jpg
75   unstaged    https://pbs.twimg.com/media/DCiavj_UwAAcXep.jpg
76   unstaged    https://pbs.twimg.com/media/DCdH8YpUQAAiEbL.jpg
77   unstaged  https://pbs.twimg.com/ext_tw_video_thumb/87514...
78   unstaged                                                NaN
79   unstaged    https://pbs.twimg.com/media/DCSzF3NVoAAPzT4.jpg
80   unstaged    https://pbs.twimg.com/media/DCN85nGUwAAzG_q.jpg
81     pupper    https://pbs.twimg.com/media/DCIgSR0XgAANEOY.jpg
82   unstaged    https://pbs.twimg.com/media/DCFGtdoXkAEsqIw.jpg
83      puppo    https://pbs.twimg.com/media/DCEeLxjXsAAvNSM.jpg
84   unstaged    https://pbs.twimg.com/media/DB-UotKXkAEHXVi.jpg
85     pupper    https://pbs.twimg.com/media/DB5HTBGXUAE0TiK.jpg
86      doggo    https://pbs.twimg.com/media/DB1m871XUAAw5vZ.jpg
87   unstaged    https://pbs.twimg.com/media/DBzhx0PWAAEhl0E.jpg
88   unstaged    https://pbs.twimg.com/media/DBwr_hzXkAEnZBW.jpg
89   unstaged    https://pbs.twimg.com/media/DBuyRlTUwAAYhG9.jpg
90   unstaged    https://pbs.twimg.com/media/DBrlZk2UQAAfAkd.jpg
91   unstaged    https://pbs.twimg.com/media/DBpm-5UXcAUeCru.jpg
92   unstaged    https://pbs.twimg.com/media/DBmKAmBXUAE-pQ-.jpg
93     pupper    https://pbs.twimg.com/media/DBkfY58XcAEdzZy.jpg
94      doggo    https://pbs.twimg.com/media/DBg_HT9WAAEeIMM.jpg
95      doggo                                                NaN
96   unstaged    https://pbs.twimg.com/media/DBaHi3YXgAE6knM.jpg
97   unstaged    https://pbs.twimg.com/media/DBW35ZsVoAEWZUU.jpg
98   unstaged    https://pbs.twimg.com/media/DBUxSSTXsAA-Jn1.jpg
99   unstaged    https://pbs.twimg.com/media/DBQwlFCXkAACSkI.jpg

          best_dog_prediction  best_dog_pred_acc  retweet_count  \
0               unpredictable                NaN           8610
1                   Chihuahua           0.323581           6324
2                   Chihuahua           0.716012           4195
3          Labrador_retriever           0.168086           8723
4                      basset           0.555712           9489
```

5	Chesapeake_Bay_retriever	0.425595	3139
6	Appenzeller	0.341703	2093
7	Pomeranian	0.566142	19083
8	Irish_terrier	0.487574	4309
9	Pembroke	0.511319	7489
10	Samoyed	0.957979	7409
11	French_bulldog	0.377417	5016
12	Pembroke	0.966327	10158
13	French_bulldog	0.991650	4587
14	golden_retriever	0.953442	2256
15	whippet	0.626152	5487
16	golden_retriever	0.714719	4544
17	golden_retriever	0.469760	4388
18	Siberian_husky	0.700377	3617
19	French_bulldog	0.995026	3531
20	basset	0.821664	5446
21	unpredictable	NaN	11770
22	Pembroke	0.809197	18412
23	Mexican_hairless	0.330741	10501
24	Samoyed	0.733942	6015
25	Chihuahua	0.793469	7836
26	kuvasz	0.309706	3328
27	unpredictable	NaN	4512
28	French_bulldog	0.999201	3214
29	pug	0.943575	6355
..	...	...	...
70	Saluki	0.509967	5248
71	Pembroke	0.931120	5681
72	bloodhound	0.575751	3411
73	NaN	NaN	4676
74	golden_retriever	0.874566	2434
75	Bernese_mountain_dog	0.534327	4767
76	Labrador_retriever	0.799551	4360
77	Siberian_husky	0.245048	5001
78	NaN	NaN	6140
79	West_Highland_white_terrier	0.714319	4790
80	Labrador_retriever	0.836052	4718
81	cocker_spaniel	0.437216	4187
82	flat-coated_retriever	0.832177	4025
83	Cardigan	0.806674	10651
84	Newfoundland	0.678537	4023
85	vizsla	0.619782	1619
86	Labrador_retriever	0.476913	5508
87	pug	0.999120	3769
88	cocker_spaniel	0.513191	3784
89	Pembroke	0.931861	9191
90	Labrador_retriever	0.972019	6463
91	pug	0.066736	8302

92	Shetland_sheepdog	0.969171	11475
93	Labrador_retriever	0.921393	3560
94	komondor	0.974781	3533
95	NaN	NaN	5613
96	kelpie	0.398053	3893
97	unpredictable	NaN	6376
98	Pembroke	0.945495	2716
99	golden_retriever	0.841001	26927

	favorite_count
0	38858
1	33280
2	25076
3	42239
4	40380
5	20248
6	11869
7	65652
8	27810
9	31997
10	30695
11	27835
12	48198
13	27219
14	15117
15	25341
16	29130
17	25635
18	19942
19	21796
20	30242
21	46324
22	69230
23	33748
24	30599
25	35217
26	12088
27	22470
28	21166
29	28015
..	...
70	27419
71	27214
72	20732
73	23508
74	18821
75	28006
76	25283

```
77          21994
78          27512
79          25566
80          27956
81          26159
82          22741
83          34852
84          24372
85           7301
86          27537
87          14799
88          20911
89          40947
90          34475
91          34679
92          38323
93          20416
94          20382
95          21083
96          22857
97          33956
98          12561
99          83523

[100 rows x 11 columns]
```

In [64]:
```
df_twitter_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2097 entries, 0 to 2096
Data columns (total 11 columns):
tweet_id               2097 non-null int64
timestamp              2097 non-null object
source                 2097 non-null object
score                  2097 non-null int64
score_ratio            2097 non-null float64
dog_stages             2097 non-null object
jpg_url                1971 non-null object
best_dog_prediction    1971 non-null object
best_dog_pred_acc      1666 non-null float64
retweet_count          2097 non-null int64
favorite_count         2097 non-null int64
dtypes: float64(2), int64(4), object(5)
memory usage: 180.3+ KB
```

Note The variable timestamp has reverted to its original format "object". If needed we can reuse the methods described in the "Clean Data" section to change back to datetime.

There are a total of 2097 entries. However only 1971 have jpg_url and since we coding data as NaN for best_dog_pred_acc there are only 1666 variables picked up (though this is not such a big deal).

As the objective of this project was to limit our data to data with jpeg images, we can repeat methods previously used to only keep data with jpeg images

12. Define Drop rows that do not have a jpeg associated to its data

Code

```
In [70]: # Drop rows with nan value in jpg_url column
         df_twitter_master = df_twitter_master.dropna(subset=['jpg_url'])
```

Test

```
In [82]: df_twitter_master.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 2096
Data columns (total 11 columns):
tweet_id               1971 non-null int64
timestamp              1971 non-null object
source                 1971 non-null object
score                  1971 non-null int64
score_ratio            1971 non-null float64
dog_stages             1971 non-null object
jpg_url                1971 non-null object
best_dog_prediction    1971 non-null object
best_dog_pred_acc      1666 non-null float64
retweet_count          1971 non-null int64
favorite_count         1971 non-null int64
dtypes: float64(2), int64(4), object(5)
memory usage: 264.8+ KB
```

Notes Now that we have the clean database, I start to iterate on assesment again

```
In [85]: df_twitter_master.sample(100)
```

```
Out[85]:             tweet_id            timestamp  source  score  score_ratio  \
         1260  692919143163629568  2016-01-29 03:56:12  iphone      9          0.9
         516   789137962068021249  2016-10-20 16:15:26  iphone     12          1.2
         1707  673580926094458881  2015-12-06 19:13:01  iphone      8          0.8
         1139  703041949650034688  2016-02-26 02:20:37  iphone     10          1.0
         1290  690728923253055490  2016-01-23 02:53:03  iphone      8          0.8
         1053  709409458133323776  2016-03-14 16:02:49  iphone      8          0.8
         1345  687460506001633280  2016-01-14 02:25:31  iphone     10          1.0
         914   728986383096946689  2016-05-07 16:34:32  iphone     11          1.1
```

1429	682697186228989953	2015-12-31 22:57:47	iphone	12	1.2
1817	671122204919246848	2015-11-30 00:22:57	iphone	4	0.4
1985	667915453470232577	2015-11-21 04:00:28	iphone	10	1.0
447	800751577355128832	2016-11-21 17:23:47	iphone	12	1.2
471	796031486298386433	2016-11-08 16:47:50	iphone	13	1.3
103	869702957897576449	2017-05-30 23:51:58	iphone	13	1.3
1144	702598099714314240	2016-02-24 20:56:55	iphone	11	1.1
838	742465774154047488	2016-06-13 21:16:49	iphone	14	1.4
2011	667509364010450944	2015-11-20 01:06:48	web	12	1.2
938	724983749226668032	2016-04-26 15:29:30	iphone	12	1.2
798	747594051852075008	2016-06-28 00:54:46	iphone	11	1.1
579	778748913645780993	2016-09-22 00:13:04	iphone	11	1.1
758	750383411068534784	2016-07-05 17:38:41	iphone	9	0.9
624	771500966810099713	2016-09-02 00:12:18	iphone	12	1.2
459	798209839306514432	2016-11-14 17:03:50	iphone	13	1.3
907	730427201120833536	2016-05-11 15:59:50	iphone	11	1.1
374	815390420867969024	2017-01-01 02:53:20	iphone	13	1.3
346	819347104292290561	2017-01-12 00:55:47	iphone	12	1.2
1891	669942763794931712	2015-11-26 18:16:16	iphone	11	1.1
476	794332329137291264	2016-11-04 00:15:59	iphone	12	1.2
959	719991154352222208	2016-04-12 20:50:42	iphone	10	1.0
953	720775346191278080	2016-04-15 00:46:48	iphone	10	1.0
...	...	...	...	...	...
1078	707776935007539200	2016-03-10 03:55:45	iphone	11	1.1
69	877556246731214848	2017-06-21 15:58:08	iphone	12	1.2
294	828372645993398273	2017-02-05 22:40:03	iphone	12	1.2
171	851591660324737024	2017-04-11 00:24:08	iphone	11	1.1
92	871879754684805121	2017-06-06 00:01:46	iphone	13	1.3
112	867051520902168576	2017-05-23 16:16:06	iphone	13	1.3
618	772152991789019136	2016-09-03 19:23:13	iphone	10	1.0
890	734787690684657664	2016-05-23 16:46:51	iphone	13	1.3
1005	714258258790387713	2016-03-28 01:10:13	iphone	10	1.0
400	811744202451197953	2016-12-22 01:24:33	iphone	13	1.3
1706	673583129559498752	2015-12-06 19:21:47	iphone	11	1.1
1887	670003130994700288	2015-11-26 22:16:09	iphone	10	1.0
1340	687704180304273409	2016-01-14 18:33:48	iphone	9	0.9
1471	680609293079592961	2015-12-26 04:41:15	iphone	9	0.9
1375	685641971164143616	2016-01-09 01:59:19	iphone	7	0.7
856	740214038584557568	2016-06-07 16:09:13	iphone	10	1.0
289	828650029636317184	2017-02-06 17:02:17	iphone	14	1.4
1109	705970349788291072	2016-03-05 04:17:02	iphone	12	1.2
782	748699167502000129	2016-07-01 02:06:06	iphone	11	1.1
118	865359393868664832	2017-05-19 00:12:11	iphone	13	1.3
1925	669000397445533696	2015-11-24 03:51:38	iphone	11	1.1
1416	683481228088049664	2016-01-03 02:53:17	iphone	11	1.1
1581	676582956622721024	2015-12-15 02:02:01	iphone	8	0.8
846	741303864243200000	2016-06-10 16:19:48	iphone	12	1.2
365	816816676327063552	2017-01-05 01:20:46	iphone	12	1.2

```
1955  668614819948453888  2015-11-23 02:19:29  iphone      7          0.7
1136  703356393781329922  2016-02-26 23:10:06  iphone      9          0.9
1902  669597912108789760  2015-11-25 19:25:57  iphone     10          1.0
1942  668815180734689280  2015-11-23 15:35:39  iphone      7          0.7
732   753294487569522689  2016-07-13 18:26:16  iphone     11          1.1

     dog_stages                                         jpg_url  \
1260   unstaged  https://pbs.twimg.com/media/CZ2-SRiWcAIjuM5.jpg
516    unstaged  https://pbs.twimg.com/media/CvOUw8vWYAAzJDq.jpg
1707   unstaged  https://pbs.twimg.com/media/CVkKRqOXIAEX83-.jpg
1139   unstaged  https://pbs.twimg.com/media/CcGO7BYW0AErrC9.jpg
1290   unstaged  https://pbs.twimg.com/media/CZX2SxaXEAEcnR6.jpg
1053   unstaged  https://pbs.twimg.com/media/CdhUIMSUIAA4wYK.jpg
1345   unstaged  https://pbs.twimg.com/media/CYpZrtDWwAE8Kpw.jpg
914    unstaged  https://pbs.twimg.com/media/Ch3hOGWUYAE7w0y.jpg
1429   unstaged  https://pbs.twimg.com/media/CXltdtaWYAIuX_V.jpg
1817   unstaged  https://pbs.twimg.com/media/CVBOFTLWwAAzlNi.jpg
1985   unstaged  https://pbs.twimg.com/media/CUTpj-GWcAATc6A.jpg
447    unstaged  https://pbs.twimg.com/media/CxzXOyBW8AEu_Oi.jpg
471    unstaged  https://pbs.twimg.com/media/CwwSaWJWIAASuoY.jpg
103    unstaged  https://pbs.twimg.com/media/DBHOOfOXoAABKlU.jpg
1144     pupper  https://pbs.twimg.com/media/CcAhPevW8AAoknv.jpg
838      pupper  https://pbs.twimg.com/media/Ck3EribXEAAPhZn.jpg
2011   unstaged  https://pbs.twimg.com/media/CUN4Or5UAAAa5K4.jpg
938    unstaged  https://pbs.twimg.com/media/Cg-o3w0WgAANXdv.jpg
798    unstaged  https://pbs.twimg.com/media/Cl_80k5WkAEbo9m.jpg
579    unstaged  https://pbs.twimg.com/media/Cs6r_-kVIAALh1p.jpg
758      pupper  https://pbs.twimg.com/media/CmnluwbXEAAqnkw.jpg
624    unstaged  https://pbs.twimg.com/media/CrTsCPHWYAANdzC.jpg
459    unstaged  https://pbs.twimg.com/media/CxPPnCYWIAAo_ao.jpg
907    unstaged  https://pbs.twimg.com/media/CiL_qh0W0AAu5VA.jpg
374    unstaged  https://pbs.twimg.com/media/C1DZQiTXgAUqgRI.jpg
346    unstaged  https://pbs.twimg.com/media/C17n1nrWQAIErU3.jpg
1891   unstaged  https://pbs.twimg.com/media/CUwdYL5UsAAP0XX.jpg
476    unstaged  https://pbs.twimg.com/media/CwYJBiHXgAQlvrh.jpg
959       doggo  https://pbs.twimg.com/media/Cf3sH62VAAA-LiP.jpg
953    unstaged  https://pbs.twimg.com/media/CgC1WqMW4AI1_N0.jpg
...         ...                                             ...
1078   unstaged  https://pbs.twimg.com/media/CdKHWimWoAABs08.jpg
69     unstaged  https://pbs.twimg.com/media/DC2OwEcW0AAf59m.jpg
294    unstaged  https://pbs.twimg.com/media/C374hb0WQAAIbQ-.jpg
171    unstaged  https://pbs.twimg.com/media/C9F2FG5WAAAJ0iN.jpg
92     unstaged  https://pbs.twimg.com/media/DBmKAmBXUAE-pQ-.jpg
112    unstaged  https://pbs.twimg.com/media/DAhiwb0XcAA8x5Q.jpg
618    unstaged  https://pbs.twimg.com/media/Crc9DEoWEAE7RLH.jpg
890    unstaged  https://pbs.twimg.com/media/CjJ9gQ1WgAAXQtJ.jpg
1005   unstaged  https://pbs.twimg.com/media/CemOGNjWQAEoN7R.jpg
400    unstaged  https://pbs.twimg.com/media/C0PlCQjXAAA9TIh.jpg
```

```
1706   unstaged  https://pbs.twimg.com/media/CVkMRUeWsAA9bMh.jpg
1887   unstaged  https://pbs.twimg.com/media/CUxUSuaW4AAdQzv.jpg
1340     pupper  https://pbs.twimg.com/media/CYs3TKzUAAAF9A2.jpg
1471   unstaged  https://pbs.twimg.com/media/CXICiB9UwAE1sKY.jpg
1375     pupper  https://pbs.twimg.com/media/CYPjvFqW8AAgiP2.jpg
856    unstaged  https://pbs.twimg.com/media/CkXEu2OUoAAs8yU.jpg
289    unstaged  https://pbs.twimg.com/media/C3_0yhCWEAETXj2.jpg
1109   unstaged  https://pbs.twimg.com/media/CcwcSS9WwAALE4f.jpg
782    unstaged  https://pbs.twimg.com/media/CmPp5pOXgAAD_SG.jpg
118    unstaged  https://pbs.twimg.com/media/DAJfxqGVoAAnvQt.jpg
1925   unstaged  https://pbs.twimg.com/media/CUjETvDVAAI8LIy.jpg
1416     pupper  https://pbs.twimg.com/media/CXw2jSpWMAAad6V.jpg
1581   unstaged  https://pbs.twimg.com/media/CWOOm8tUwAAB901.jpg
846    unstaged  https://pbs.twimg.com/media/Ckmj7mNWYAA4NzZ.jpg
365    unstaged  https://pbs.twimg.com/media/C1XqbhXXUAElpfI.jpg
1955   unstaged  https://pbs.twimg.com/media/CUdloW8WEAAxB_Y.jpg
1136   unstaged  https://pbs.twimg.com/media/CcLS6QKUcAAUuPa.jpg
1902   unstaged  https://pbs.twimg.com/media/CUrjvxiVEAA94dH.jpg
1942   unstaged  https://pbs.twimg.com/media/CUgb21RXIAAlff7.jpg
732    unstaged  https://pbs.twimg.com/media/CnQ9Vq1WEAEYP01.jpg

           best_dog_prediction  best_dog_pred_acc  retweet_count  \
1260              Saint_Bernard           0.612635            817
516                   Chihuahua           0.746135           3150
1707                     beagle           0.985062            284
1139              unpredictable                NaN          13764
1290                     kuvasz           0.422806            572
1053          Shetland_sheepdog           0.797450            764
1345                Boston_bull           0.223366            595
914                 Maltese_dog           0.952070            893
1429              unpredictable                NaN            393
1817                  Chihuahua           0.101228           2688
1985                      boxer           0.196655             58
447              cocker_spaniel           0.771984           3128
471            golden_retriever           0.893775           4177
103                    Pembroke           0.993449           6512
1144                     kelpie           0.219179           3603
838               unpredictable                NaN           4259
2011                     beagle           0.636169           2218
938            golden_retriever           0.675750           1429
798                     basenji           0.389136           1166
579  Staffordshire_bullterrier           0.351434           1502
758               Border_collie           0.672791           1261
624          Labrador_retriever           0.833952           2930
459                    Pekinese           0.524583           2882
907                  Eskimo_dog           0.682082           1143
374               unpredictable                NaN           4279
346                  Rottweiler           0.909106           1349
```

1891	vizsla	0.743216	180
476	Samoyed	0.988307	2990
959	golden_retriever	0.605304	1914
953	Newfoundland	0.489970	743
...	...	...	...
1078	miniature_pinscher	0.890426	1038
69	basset	0.995368	3877
294	malamute	0.663047	3261
171	Cardigan	0.394507	3729
92	Shetland_sheepdog	0.969171	11475
112	Samoyed	0.471403	8160
618	golden_retriever	0.275318	1262
890	golden_retriever	0.883991	6908
1005	collie	0.176758	784
400	Pekinese	0.386082	1822
1706	golden_retriever	0.113946	391
1887	beagle	0.375313	98
1340	miniature_pinscher	0.956063	931
1471	French_bulldog	0.700764	785
1375	Lakeland_terrier	0.253839	855
856	Chesapeake_Bay_retriever	0.586414	2162
289	golden_retriever	0.649209	1495
1109	golden_retriever	0.776346	976
782	Pembroke	0.849029	1767
118	Chesapeake_Bay_retriever	0.832435	5207
1925	Pembroke	0.822940	6739
1416	keeshond	0.508951	1088
1581	Boston_bull	0.196307	305
846	Chihuahua	0.768156	3554
365	malamute	0.668164	2295
1955	unpredictable	NaN	334
1136	Border_collie	0.894842	423
1902	Eskimo_dog	0.595665	160
1942	redbone	0.461172	285
732	chow	0.194773	1156

	favorite_count
1260	2863
516	10645
1707	864
1139	28355
1290	2322
1053	2798
1345	2190
914	3393
1429	1409
1817	3643
1985	218

447	11482
471	11837
103	28584
1144	11091
838	7786
2011	6994
938	3961
798	3973
579	7536
758	4908
624	8991
459	11354
907	3712
374	11256
346	7855
1891	528
476	10473
959	5160
953	2641
...	...
1078	3507
69	22825
294	13519
171	17003
92	38323
112	32786
618	4105
890	13454
1005	3211
400	8245
1706	1241
1887	345
1340	2618
1471	2823
1375	3141
856	7174
289	10271
1109	3368
782	5110
118	27013
1925	21699
1416	2821
1581	1275
846	9426
365	10885
1955	635
1136	2042
1902	534

```
1942             596
732             3688

[100 rows x 11 columns]
```

In [83]:
```
df_twitter_master.describe()
```

Out[83]:
```
           tweet_id        score  score_ratio  best_dog_pred_acc  \
count  1.971000e+03  1971.000000  1971.000000        1666.000000
mean   7.360418e+17    12.223237     1.222324           0.551571
std    6.752810e+16    41.634034     4.163403           0.298923
min    6.660209e+17     0.000000     0.000000           0.000010
25%    6.758656e+17    10.000000     1.000000           0.305955
50%    7.088343e+17    11.000000     1.100000           0.550914
75%    7.880951e+17    12.000000     1.200000           0.822939
max    8.924206e+17  1776.000000   177.600000           0.999956

       retweet_count  favorite_count
count    1971.000000     1971.000000
mean     2740.205987     8917.240487
std      4724.045594    12641.240495
min        13.000000       79.000000
25%       611.000000     1949.500000
50%      1329.000000     4063.000000
75%      3146.000000    11214.000000
max     77488.000000   143577.000000
```

In [84]:
```
sum(df_twitter_master.duplicated())
```

Out[84]: 0

Notes

- 1) We see that there are extreme values in score (eg. the max value is 1776). However due to the nature of the scoring system it is not possibel to know what score was intended as scores range from any value greater than or equal to 0.
- 2) There does not seem to be any duplicate rows

In [87]:
```
# For the project purposed I reuse the previous code to reformat the timestamp.
# Run this code if you want to use timestamp in datetime format for further analysis.
df_twitter_master['timestamp'] = pd.to_datetime(df_twitter_master['timestamp'])
```

In [88]:
```
#A final test before saving the file as the master archive
df_twitter_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 2096
Data columns (total 11 columns):
```

```
tweet_id                1971 non-null int64
timestamp               1971 non-null datetime64[ns]
source                  1971 non-null object
score                   1971 non-null int64
score_ratio             1971 non-null float64
dog_stages              1971 non-null object
jpg_url                 1971 non-null object
best_dog_prediction     1971 non-null object
best_dog_pred_acc       1666 non-null float64
retweet_count           1971 non-null int64
favorite_count          1971 non-null int64
dtypes: datetime64[ns](1), float64(2), int64(4), object(4)
memory usage: 264.8+ KB
```

Note Now I will create the final master archive which we will use to perform visualizations. Prior to the creation of the master file I make a copy of the df_twitter_master data frame

```
In [90]: df_twitter_master_v2 = df_twitter_master.copy()
```

```
In [225]: #Create and load df_twitter_archive_master (i.e. final merged file of all three data f
          df_twitter_master_v2.to_csv('twitter_archive_master.csv', index=None)
          df_twitter_archive_master = pd.read_csv('twitter_archive_master.csv')
```

```
In [226]: # Quick test to see if data frame saved and loaded correctly
          df_twitter_archive_master.sample(5)
```

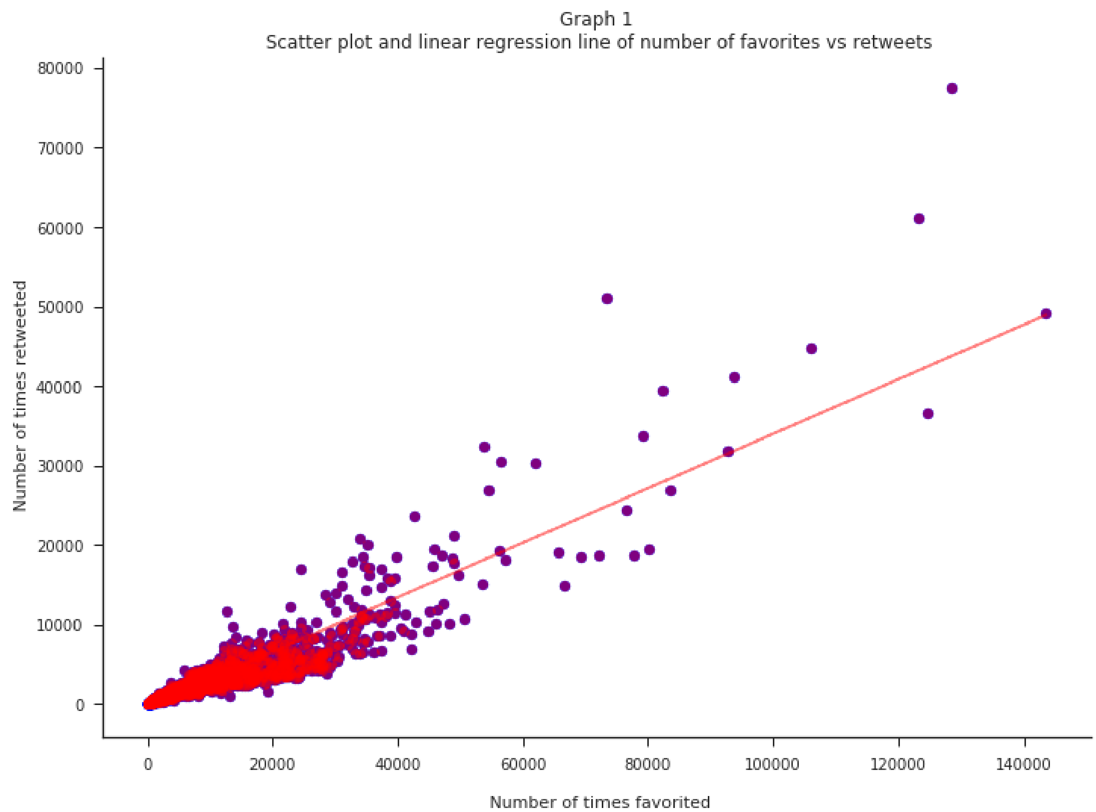
```
Out[226]:                 tweet_id            timestamp  source  score  score_ratio  \
          376   812466873996607488  2016-12-24 01:16:12  iphone     12          1.2
          268   829141528400556032  2017-02-08 01:35:19  iphone     12          1.2
          124   861383897657036800  2017-05-08 00:54:59  iphone     13          1.3
          1641  672160042234327040  2015-12-02 21:06:56  iphone      8          0.8
          116   863907417377173506  2017-05-15 00:02:33  iphone     13          1.3

               dog_stages                                          jpg_url  \
          376    unstaged  https://pbs.twimg.com/media/C0Z2T_GWgAAxbL9.jpg
          268    unstaged  https://pbs.twimg.com/media/C4GzztSWAAA_qi4.jpg
          124    unstaged  https://pbs.twimg.com/media/C_RAFTxUAAAbXjV.jpg
          1641     pupper  https://pbs.twimg.com/media/CVP9_beUEAAwURR.jpg
          116    unstaged  https://pbs.twimg.com/media/C_03NPeUQAAgrMl.jpg

               best_dog_prediction  best_dog_pred_acc  retweet_count  favorite_count
          376           Great_Dane           0.078205           2174            8733
          268     golden_retriever           0.573140           8259           26430
          124             Cardigan           0.771008          11184           37002
          1641                 pug           0.561027            382             905
          116        unpredictable                NaN           4315           21075
```

15 5. Analysis and visualization

Note As part of this project we must provide at least three separate insights, of which at least one must be a visualization. In the following section I will do so.

Visual relationship between number of retweets and favorites

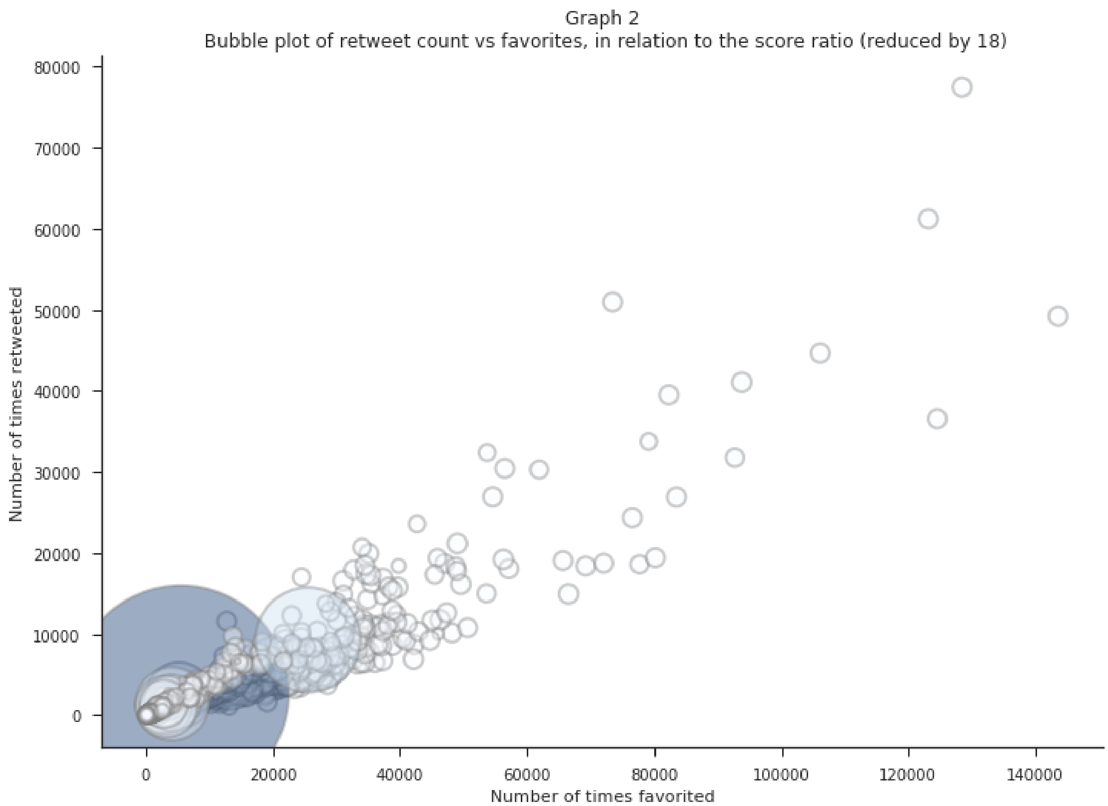
In [296]:
```
# Set plot to visually pleasing dimensions
a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
#Asign x and y axis values
y=df_twitter_archive_master['retweet_count']
x=df_twitter_archive_master['favorite_count']
plt.scatter(x, y, color="blue", alpha=1.0, label='');
#Label x and y axis
plt.xlabel('\n Number of times favorited')
plt.ylabel(' \n Number of times retweeted')
# Draw the same scatter plot but with a regression line and overlay so scatter and reg
m,b = np.polyfit(x, y, 1)
fit = np.polyfit(x,y,1)
fit_fn = np.poly1d(fit)
plt.plot(x,y, 'yo', x, fit_fn(x), '--k', color="red", alpha=0.5, label="Lnear regressi
#axis removal
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(True)
#plot label and graph
ax.set_title("Graph 1\n Scatter plot and linear regression line of number of favorites
# Save plot
plt.savefig('graph1.png')
#Plot graph
plt.show();
```



Note We see here a predictable relationship between the number of times a tweet is favorited in relation to the number of times it is retweeted. In general the relationship is poistively linear.

```
In [295]: # Set plot to visually pleasing dimensions
          a4_dims = (11.7, 8.27)
          fig, ax = plt.subplots(figsize=a4_dims)
          # Define variables to be included in plot
          x = df_twitter_archive_master['favorite_count']
          y = df_twitter_archive_master['retweet_count']
          z = df_twitter_archive_master['score_ratio']/18
          # Change color with c and alpha. I map the color to the z axis value.
          plt.scatter(x, y, s=z*2000, c=z, cmap="Blues", alpha=0.4, edgecolors="grey", linewidth
          #axis removal
          ax.spines['top'].set_visible(False)
          ax.spines['right'].set_visible(False)
          ax.spines['bottom'].set_visible(True)
          # Add titles (main and on axis)
          plt.xlabel("Number of times favorited")
          plt.ylabel("Number of times retweeted")
          plt.title("Graph 2\n Bubble plot of retweet count vs favorites, in relation to the sco
          # Save plot
```

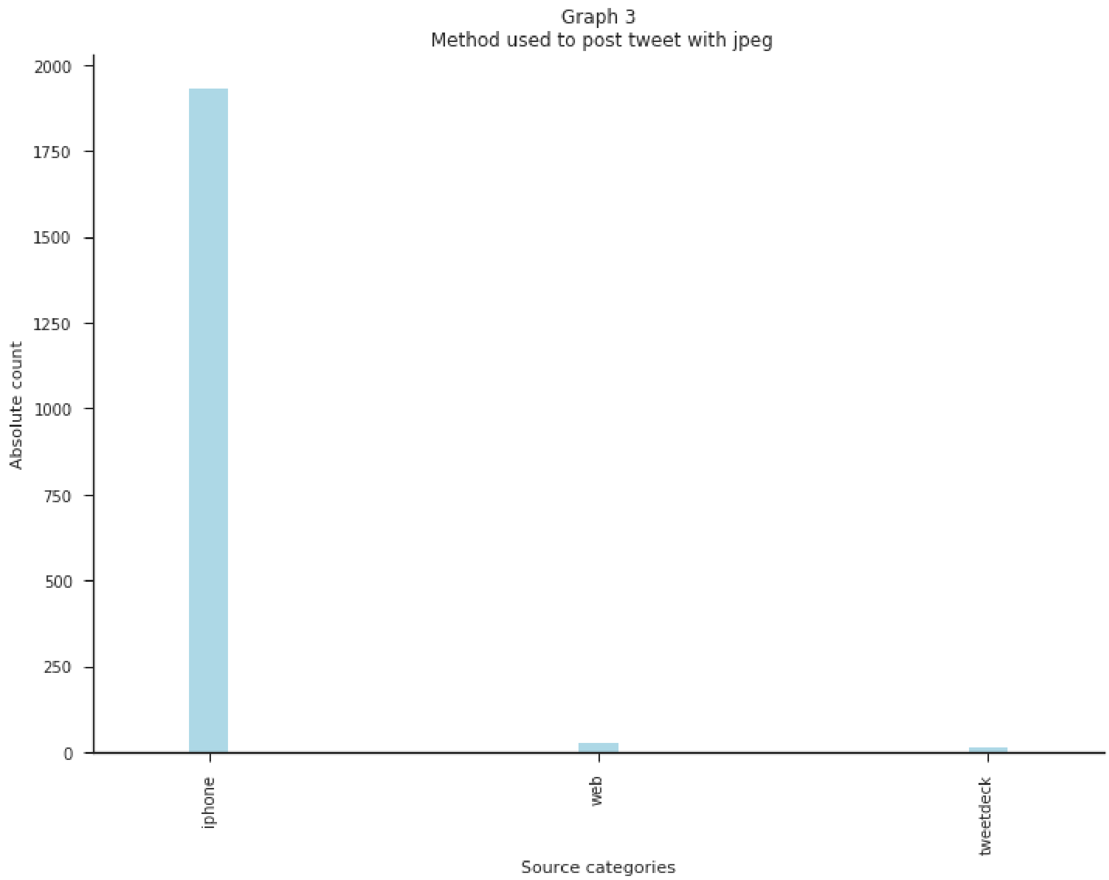
```
plt.savefig('graph2.png')
#Plot graph
plt.show()
```



Note To drive the message home about the score attributed to the dogs, I ran the same scatter plot but sized each plot point in relationship to the score_ratio/18. It is pretty clear that the score_ratio does not impact the number of times that the tweet is favorited or retweeted.

```
In [294]: # Set plot to visually pleasing dimensions
          a4_dims = (11.7, 8.27)
          fig, ax = plt.subplots(figsize=a4_dims)
          #Bar plot of categories in source
          df_twitter_archive_master['source'].value_counts().plot(kind='bar', color='lightblue',
          #axis removal
          ax.spines['top'].set_visible(False)
          ax.spines['right'].set_visible(False)
          ax.spines['bottom'].set_visible(True)
          # Add titles (main and on axis)
          plt.xlabel("Source categories")
          plt.ylabel("Absolute count")
          plt.title("Graph 3\n Method used to post tweet with jpeg")
```

```
# Save plot
plt.savefig('graph3.png')
#Plot graph
plt.show();
```



Note The method most used to post tweets with jpegs from this dataset to WeRateDogs is overwhelmingly directly from the iPhone app, followed by twitter's webclient and finaly tweetdeck.

```
In [293]: # To get an idea of the type of dog breed most frequently posted in these tweets I mak
          display(df_twitter_archive_master['best_dog_prediction'].value_counts())
```

```
unpredictable                   305
golden_retriever                156
Labrador_retriever              106
Pembroke                         94
Chihuahua                        90
pug                              62
toy_poodle                       50
chow                             48
```

Samoyed	42
Pomeranian	41
malamute	33
Chesapeake_Bay_retriever	31
French_bulldog	31
cocker_spaniel	30
miniature_pinscher	24
Eskimo_dog	22
German_shepherd	21
Cardigan	21
Siberian_husky	20
Staffordshire_bullterrier	20
beagle	20
Shih-Tzu	20
Maltese_dog	19
Shetland_sheepdog	18
Rottweiler	18
basset	17
kuvasz	17
Italian_greyhound	17
Lakeland_terrier	17
West_Highland_white_terrier	16
American_Staffordshire_terrier	16
Great_Pyrenees	15
soft-coated_wheaten_terrier	14
Pekinese	14
Old_English_sheepdog	14
kelpie	13
schipperke	13
vizsla	13
dalmatian	12
Boston_bull	12
Airedale	12
Border_collie	12
collie	11
boxer	11
standard_poodle	11
Norwegian_elkhound	11
malinois	11
whippet	11
Great_Dane	11
Bernese_mountain_dog	11
English_springer	10
Blenheim_spaniel	10
borzoi	10
Yorkshire_terrier	10
Doberman	9
basenji	9

German_short-haired_pointer	8
Brittany_spaniel	8
miniature_poodle	8
English_setter	8
flat-coated_retriever	8
bloodhound	7
Dandie_Dinmont	7
Newfoundland	7
Saint_Bernard	7
Border_terrier	7
Mexican_hairless	7
papillon	7
Norfolk_terrier	6
Bedlington_terrier	6
redbone	6
Irish_terrier	6
Lhasa	5
bull_mastiff	5
Norwich_terrier	5
Walker_hound	5
miniature_schnauzer	5
Irish_setter	4
Tibetan_mastiff	4
Weimaraner	4
Gordon_setter	4
Ibizan_hound	4
Scottish_deerhound	4
Tibetan_terrier	4
Welsh_springer_spaniel	4
Saluki	4
Rhodesian_ridgeback	4
keeshond	4
bluetick	4
briard	3
cairn	3
komondor	3
Brabancon_griffon	3
giant_schnauzer	3
Irish_water_spaniel	3
curly-coated_retriever	3
Leonberg	3
Greater_Swiss_Mountain_dog	3
Afghan_hound	3
toy_terrier	3
black-and-tan_coonhound	2
groenendael	2
wire-haired_fox_terrier	2
Appenzeller	2

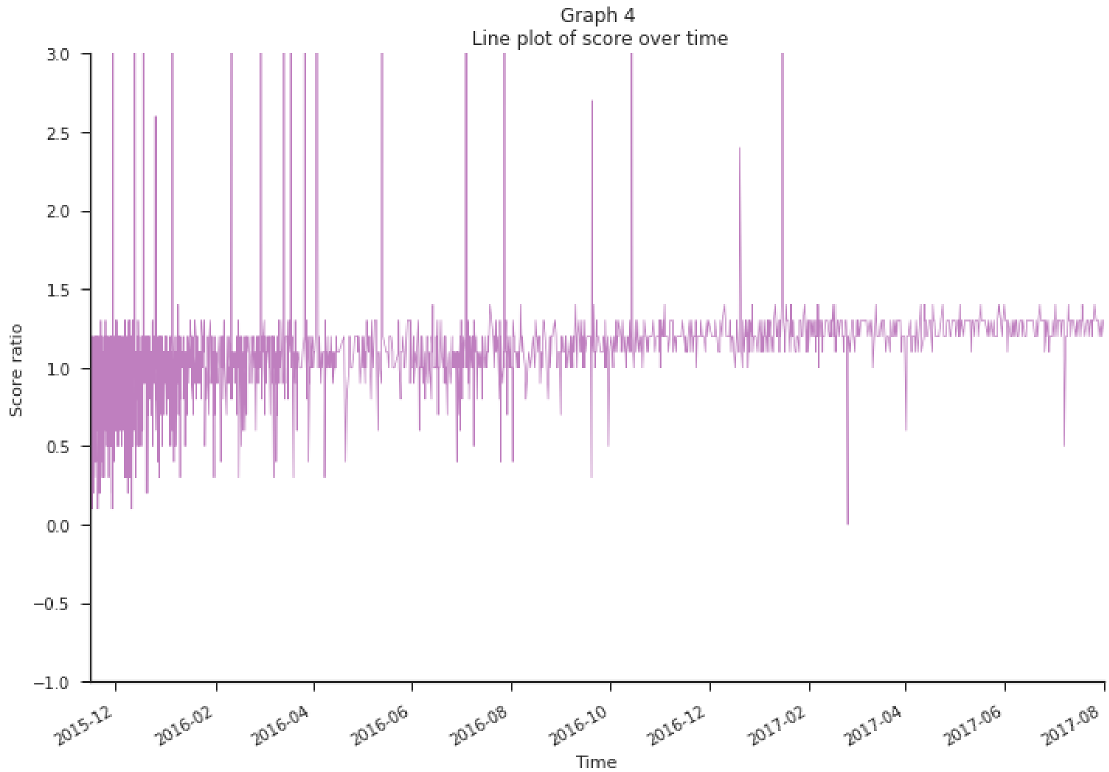
```
Sussex_spaniel                     2
Australian_terrier                 2
Scotch_terrier                     1
Irish_wolfhound                    1
Japanese_spaniel                   1
clumber                            1
EntleBucher                        1
Bouvier_des_Flandres               1
silky_terrier                      1
standard_schnauzer                 1
Name: best_dog_prediction, dtype: int64
```

Note Very quickly we observe that most jpegs were not deciferable by the neural network. After this the top three dog breeds tweeted in this data set were the Golden Retriever, then the Labrador and the Pembroke

Note Finally to make use of the timestamp column I run a line plot of the score_ratio.

```
In [231]: #To utilize date time without damaging the archive_master dataframe I create a time da
          df_time_graph = df_twitter_archive_master
          #As before convert the timestamp into a datetime
          df_time_graph['timestamp'] = pd.to_datetime(df_time_graph['timestamp'])
          #Set the timestamp as the index for df_time_graph data frame
          df_time_graph.set_index('timestamp', inplace=True)
```

```
In [292]: # Set plot to visually pleasing dimensions
          a4_dims = (11.7, 8.27)
          fig, ax = plt.subplots(figsize=a4_dims)
          #Plot graph
          df_time_graph['score_ratio'].plot(alpha=0.5,aa=True, c='purple', lw=0.7)
          # Limit the y axis focus to only consider ratios between -1 and 3
          plt.ylim(-1, 3)
          #axis removal
          ax.spines['top'].set_visible(False)
          ax.spines['right'].set_visible(False)
          ax.spines['bottom'].set_visible(True)
          # Add titles (main and on axis)
          plt.xlabel("Time")
          plt.ylabel("Score ratio")
          plt.title("Graph 4\n Line plot of score over time")
          # Save plot
          plt.savefig('graph4.png')
          #Plot graph
          plt.show();
```



Note We observe that initially the score tends to be below 1. However, as time progresses the scores become closer towards 1- in this sense they "regress towards the mean" of 1. Additionally the density of post diminishes (this is observed by the color becoming lighter as time progresses).