Udacity WeRateDogs Project: Wrangle and Analyse Twitter Data

By Vincent Man

Date: July 28, 2018

Library

In [1]:
```
# Import all neccessary packages
import pandas as pd
import tweepy
import requests
import os
import json
from datetime import datetime, timedelta

import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')

import seaborn as sns
sns.set(style='darkgrid')
```

Gather Data

1.Twitter Archive File

In [2]:
```
# Import and read csv file into a dataframe that contains tweets
twitter_archive = pd.read_csv('twitter-archive-enhanced.csv', encoding=

twitter_archive.sort_values(by='tweet_id', inplace=True)
```

Tweet Image Predictions of Dog breeds

In [3]:
```
# Programmatically import the tsv data that contains predictions of do
img_pred_url = ('https://d17h27t6h515a5.cloudfront.net/topher/2017/'
                        'August/599fd2ad_image-predictions/image-predic
# Downloading file from a specified URL by using the request library
response = requests.get(img_pred_url)

# Saving data into a tsv file
with open('image-predictions.tsv',
          mode='wb') as file:
    file.write(response.content)

# Import data from TSV
img_pred_df = pd.read_csv('image-predictions.tsv',
                          sep='\t',
                          encoding='utf-8')
```

Tweets - Twitter API and JSON

In [5]:
```
# Details for Twitter API
consumer_key = 'YOUR CONSUMER KEY'
consumer_secret = 'YOUR CONSUMER SECRET'
access_token = 'YOUR ACCESS TOKEN'
access_secret = 'YOUR ACCESS SECRET'

# Establish connection to API
auth = tweepy.OAuthHandler(consumer_key,
                           consumer_secret)

auth.set_access_token(access_token,
                      access_secret)

api = tweepy.API(auth,
                 parser = tweepy.parsers.JSONParser(),
                 wait_on_rate_limit = True,
                 wait_on_rate_limit_notify = True)
```

In [6]:
```
# Append each Tweet id and store in a list
tweet_data = []
# Tweet id that cannot be found
failed_id = []

# Gather all of the tweets into twitter_archive dataframe
for tweet_id in twitter_archive['tweet_id']:
    try:
        tweet_status = api.get_status(tweet_id,
                                      tweet_mode = 'extended')

        # Save only the necessary columns into the dataframe
        favorite_count = tweet_status['favorite_count']
        retweet_count = tweet_status['retweet_count']

        tweet_data.append({'tweet_id': int(tweet_id),
                           'favorite_count': int(favorite_count),
                           'retweet_count': int(retweet_count)})

    # Tweets that might have been deleted
    except Exception:
        print("Error Occurred for Twitter ID: " + str(tweet_id))
        failed_id.append(tweet_id)
```

```
Rate limit reached. Sleeping for: 47

Error Occurred for Twitter ID: 754011816964026368
Error Occurred for Twitter ID: 770743923962707968
Error Occurred for Twitter ID: 775096608509886464
Error Occurred for Twitter ID: 802247111496568832

Rate limit reached. Sleeping for: 110

Error Occurred for Twitter ID: 827228250799742977
Error Occurred for Twitter ID: 837012587749474308
Error Occurred for Twitter ID: 842892208864923648
Error Occurred for Twitter ID: 845459076796616705
Error Occurred for Twitter ID: 861769973181624320
Error Occurred for Twitter ID: 866816280283807744
Error Occurred for Twitter ID: 869988702071779329
Error Occurred for Twitter ID: 873697596434513921
Error Occurred for Twitter ID: 888202515573088257
```

In [7]:
```
# Showing the number of tweets that are working
print("The number of found Tweets: "
      + str(len(tweet_data)))
# Showing the number of Tweets that cannot be found
print("The number of failed Tweets: "
      + str(len(failed_id)))
```

```
The number of found Tweets: 2343
The number of failed Tweets: 13
```

In [8]:
```
# Save the failed tweet ids that cannot be found and append them to th

for f in failed_id:
    try:
        favorite_count = tweet_status['favorite_count']
        retweet_count = tweet_status['retweet_count']

        tweet_data.append({'tweet_id': int(tweet_id),
                           'favorite_count': int(favorite_count),
                           'retweet_count': int(retweet_count)})
    except Exception:
        failed_id.append(f)
```

In [9]:
```
# Showing the total number of tweet ids in the list including the fail
print("Total number of Tweets after appending failed tweets to list: "
      + str(len(tweet_data)))
```

```
Total number of Tweets after appending failed tweets to list:
2356
```

In [10]:
```
# Create dataframes
tweet_json = pd.DataFrame(tweet_data,
                          columns = ['tweet_id', 'favorite_count', 're

# Export tweet_json dataframe into a CSV file
tweet_json.to_csv('tweet_json.txt',
                  encoding = 'utf-8',
                  index=False)
```

In [11]:
```
# Read the tweet_json csv file as a Pandas dataframe
tweet_json = pd.read_csv('tweet_json.txt',
                         encoding = 'utf-8')
```

Assess Data

Quality Issues:

twitter_archive dataset

1. Timestamp variable should be datetime instead of an object
2. Tweet_id should be a string
3. To keep original tweets without retweets and replies (remove in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp)
4. The source variable has html tags entries
5. Only keep the original ratings with images
6. The numerator and denominator columns have incorrect entries
7. The name variable does not have valid dog names such as 'a' and 'an'

img_pred_df dataset

8. The variables of P1, P2 and P3 do not consistently start with a capital letter and there are underscores
9. Confusing variable headers

tweet_json dataset

10. 13 duplications of Twitter ID '892420643555336193'

Tidyness Issues:

1. Melt the twitter_archive dataframe and have only one column instead of 4 ('doggo', 'floofer' ,'pupper' and 'puppo')
2. Consolidate and merge the 3 datasets: twitter_archive, img_pred_df and tweet_json into one as they are describing the same tweet

In [12]:
```
# List all of twitter_archive dataset variables
twitter_archive.columns
```

Out[12]:
```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', '
timestamp',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_us
er_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numera
tor',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', '
puppo'],
      dtype='object')
```

In [13]:
```
# Showing the first 5 rows of twitter_archive dataset
print(twitter_archive.head())
```

```
print(twitter_archive.head())
```

```
               tweet_id  in_reply_to_status_id  in_reply_to_user_id
\
2355  666020888022790149                    NaN                  NaN
2354  666029285002620928                    NaN                  NaN
2353  666033412701032449                    NaN                  NaN
2352  666044226329800704                    NaN                  NaN
2351  666049248165822465                    NaN                  NaN

                     timestamp  \
2355  2015-11-15 22:32:08 +0000
2354  2015-11-15 23:05:30 +0000
2353  2015-11-15 23:21:54 +0000
2352  2015-11-16 00:04:52 +0000
2351  2015-11-16 00:24:50 +0000

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

                                                 text  retweeted_s
tatus_id  \
2355  Here we have a Japanese Irish Setter. Lost eye...
NaN
2354  This is a western brown Mitsubishi terrier. Up...
NaN
2353  Here is a very happy pup. Big fan of well-main...
NaN
2352  This is a purebred Piers Morgan. Loves to Netf...
NaN
2351  Here we have a 1949 1st generation vulpix. Enj...
NaN

      retweeted_status_user_id retweeted_status_timestamp  \
2355                       NaN                         NaN
2354                       NaN                         NaN
2353                       NaN                         NaN
2352                       NaN                         NaN
2351                       NaN                         NaN

                                          expanded_urls  rating_nume
rator  \
2355  https://twitter.com/dog_rates/status/666020888...
(https://twitter.com/dog_rates/status/666020888...)
8
2354  https://twitter.com/dog_rates/status/666029285...
(https://twitter.com/dog_rates/status/666029285...)
7
2353  https://twitter.com/dog_rates/status/666033412...
(https://twitter.com/dog_rates/status/666033412...)
```

```
9
2352  https://twitter.com/dog_rates/status/666044226...
(https://twitter.com/dog_rates/status/666044226...)
6
2351  https://twitter.com/dog_rates/status/666049248...
(https://twitter.com/dog_rates/status/666049248...)
5
```

	rating_denominator	name	doggo	floofer	pupper	puppo
2355	10	None	None	None	None	None
2354	10	a	None	None	None	None
2353	10	a	None	None	None	None
2352	10	a	None	None	None	None
2351	10	None	None	None	None	None

In [14]:
```
# Showing the last 5 rows of twitter_archive dataset
print(twitter_archive.tail())
```

```
             tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
4  891327558926688256                    NaN                  NaN
3  891689557279858688                    NaN                  NaN
2  891815181378084864                    NaN                  NaN
1  892177421306343426                    NaN                  NaN
0  892420643555336193                    NaN                  NaN

                   timestamp  \
4  2017-07-29 16:00:24 +0000
3  2017-07-30 15:58:51 +0000
2  2017-07-31 00:18:03 +0000
1  2017-08-01 00:17:27 +0000
0  2017-08-01 16:23:56 +0000

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

                                                text  retweeted_stat
us_id  \
4  This is Franklin. He would like you to stop ca...
NaN
3  This is Darla. She commenced a snooze mid meal...
NaN
2  This is Archie. He is a rare Norwegian Pouncin...
NaN
1  This is Tilly. She's just checking pup on you....
NaN
0  This is Phineas. He's a mystical boy. Only eve...
NaN
```

```
   retweeted_status_user_id retweeted_status_timestamp  \
4                       NaN                         NaN
3                       NaN                         NaN
2                       NaN                         NaN
1                       NaN                         NaN
0                       NaN                         NaN

                                          expanded_urls  rating_numerat
or  \
4  https://twitter.com/dog_rates/status/891327558...
(https://twitter.com/dog_rates/status/891327558...)
12
3  https://twitter.com/dog_rates/status/891689557...
(https://twitter.com/dog_rates/status/891689557...)
13
2  https://twitter.com/dog_rates/status/891815181...
(https://twitter.com/dog_rates/status/891815181...)
12
1  https://twitter.com/dog_rates/status/892177421...
(https://twitter.com/dog_rates/status/892177421...)
13
0  https://twitter.com/dog_rates/status/892420643...
(https://twitter.com/dog_rates/status/892420643...)
13

   rating_denominator      name doggo floofer pupper puppo
4                  10   Franklin  None    None   None  None
3                  10      Darla  None    None   None  None
2                  10     Archie  None    None   None  None
1                  10      Tilly  None    None   None  None
0                  10    Phineas  None    None   None  None
```

In [15]:
```
# Showing 5 ramdom samples
print(twitter_archive.sample(5))
```

```
                tweet_id  in_reply_to_status_id  in_reply_to_user_id
\
1046  743545585370791937                    NaN                  NaN
1050  743210557239623680                    NaN                  NaN
144   863907417377173506                    NaN                  NaN
1056  742161199639494656                    NaN                  NaN
752   778990705243029504                    NaN                  NaN

                     timestamp  \
1046  2016-06-16 20:47:36 +0000
1050  2016-06-15 22:36:19 +0000
144   2017-05-15 00:02:33 +0000
1056  2016-06-13 01:06:33 +0000
752   2016-09-22 16:13:51 +0000

                                               source  \
1046  <a href="http://twitter.com/download/iphone" r...
1050  <a href="http://twitter.com/download/iphone" r...
```

```
144   <a href="http://twitter.com/download/iphone" r...
1056  <a href="http://twitter.com/download/iphone" r...
752   <a href="http://twitter.com/download/iphone" r...

                                                    text  retweeted_s
tatus_id  \
1046  Say hello to Bentley and Millie. They do every...
NaN
1050  Meet Kayla, an underground poker legend. Playe...
NaN
144   This is Albus. He's quite impressive at hide a...
NaN
1056  This is Doug. He's trying to float away. 12/10...
NaN
752   This is Jay. He's really h*ckin happy about th...
NaN

      retweeted_status_user_id retweeted_status_timestamp  \
1046                       NaN                         NaN
1050                       NaN                         NaN
144                        NaN                         NaN
1056                       NaN                         NaN
752                        NaN                         NaN

                                           expanded_urls  rating_nume
rator  \
1046  https://twitter.com/dog_rates/status/743545585...
(https://twitter.com/dog_rates/status/743545585...)
11
1050  https://twitter.com/dog_rates/status/743210557...
(https://twitter.com/dog_rates/status/743210557...)
10
144   https://twitter.com/dog_rates/status/863907417...
(https://twitter.com/dog_rates/status/863907417...)
13
1056  https://twitter.com/dog_rates/status/742161199...
(https://twitter.com/dog_rates/status/742161199...)
12
752   https://twitter.com/dog_rates/status/778990705...
(https://twitter.com/dog_rates/status/778990705...)
11

      rating_denominator     name doggo floofer pupper puppo
1046                  10  Bentley  None    None   None  None
1050                  10    Kayla  None    None   None  None
144                   10    Albus  None    None   None  None
1056                  10     Doug  None    None   None  None
752                   10      Jay  None    None   None  None
```

In [16]:
```
# Show the data types of each variable
twitter_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 2355 to 0
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: 331.3+ KB
```

In [17]:
```
# Check descriptive statistics of twitter_archive
twitter_archive.describe()
```

Out[17]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweeted
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	
std	6.856705e+16	7.582492e+16	1.252797e+17	6.236928e+16	
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	

In [18]:
```
#Show number of duplicated entries in twitter_archive dataset
print('Number of Duplicated entries in twitter_archive: '
      + '\n' +str(sum(twitter_archive.duplicated())))
```

```
Number of Duplicated entries in twitter_archive:
0
```

In [19]:
```
# Show the number of Null entries
print('Number of Null entries: ' + '\n')

print('in_reply_to_status_id: ' +
      str(sum(twitter_archive['in_reply_to_status_id'].isnull())))

print('in_reply_to_user_id: ' +
      str(sum(twitter_archive['in_reply_to_user_id'].isnull())))

print('retweeted_status_id: ' +
      str(sum(twitter_archive['retweeted_status_id'].isnull())))

print('retweeted_status_user_id: ' +
      str(sum(twitter_archive['retweeted_status_user_id'].isnull())))

print('retweeted_status_timestamp: ' +
      str(sum(twitter_archive['retweeted_status_timestamp'].isnull())))
```

```
Number of Null entries:

in_reply_to_status_id: 2278
in_reply_to_user_id: 2278
retweeted_status_id: 2175
retweeted_status_user_id: 2175
retweeted_status_timestamp: 2175
```

In [20]:
```
#len(twitter_archive[twitter_archive.in_reply_to_status_id!=None])
```

In [21]:
```
# Show the number of entries for each unique element
# under the variable 'name'
print(twitter_archive['name'].value_counts())
```

```
None        745
a            55
Charlie      12
Lucy         11
Cooper       11
Oliver       11
Tucker       10
Lola         10
Penny        10
Bo            9
Winston       9
the           8
Sadie         8
Daisy         7
Bailey        7
Buddy         7
an            7
Toby          7
Leo           6
Jack          6
```

```
Oscar          6
Bella          6
Rusty          6
Koda           6
Stanley        6
Dave           6
Milo           6
Jax            6
Scout          6
Oakley         5
              ...
Maya           1
Lugan          1
Brockly        1
Eazy           1
Nimbus         1
Mona           1
light          1
Jomathan       1
Clarkus        1
Rizzo          1
Cupid          1
Barry          1
Puff           1
Timber         1
Peanut         1
Kloey          1
Kawhi          1
Winifred       1
Loomis         1
Stark          1
Zeus           1
Katie          1
Bloop          1
Milky          1
Tassy          1
Andy           1
Yoda           1
Jessifer       1
Snickers       1
life           1
Name: name, Length: 957, dtype: int64
```

In [22]:
```
# Show entries that are in lowercase under the variable 'name'
lowercase = twitter_archive[twitter_archive['name'].str.islower()]
lowercase.loc[:,['name']].sample(10)
```

Out[22]:

	name
2019	just
2311	a
2128	a
1878	a
1457	just
2222	a
2204	an
988	not
852	my
2304	a

In [23]:
```
# Show the number of entries for each unique element
# under the variable 'source'
print(twitter_archive['source'].value_counts())
```

```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter
for iPhone</a>     2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow
">TweetDeck</a>      11
Name: source, dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 2355 to 0
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: 331.3+ KB
```

In [25]:
```
# Show the number of entries for each unique element
# under the variable 'rating_numerator'
print(twitter_archive['rating_numerator'].value_counts())
```

```
12      558
11      464
10      461
13      351
9       158
8       102
7        55
14       54
5        37
6        32
3        19
4        17
1         9
2         9
420       2
0         2
15        2
75        2
80        1
20        1
24        1
26        1
44        1
50        1
60        1
165       1
84        1
88        1
144       1
182       1
143       1
666       1
960       1
1776      1
17        1
27        1
45        1
99        1
121       1
204       1
Name: rating_numerator, dtype: int64
```

In [26]:
```
# Show the number of entries for each unique element
# under the variable 'rating_denominator'
print(twitter_archive['rating_denominator'].value_counts())
```

```
10     2333
11        3
50        3
80        2
20        2
2         1
16        1
40        1
70        1
15        1
90        1
110       1
120       1
130       1
150       1
170       1
7         1
0         1
Name: rating_denominator, dtype: int64
```

In [27]:
```
# Show the number of entries for each unique element
# under the variable doggo, floofer, pupper, puppo.

print(twitter_archive['doggo'].value_counts())
print(twitter_archive['floofer'].value_counts())
print(twitter_archive['pupper'].value_counts())
print(twitter_archive['puppo'].value_counts())
```

```
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
```

In [28]:
```
# List all of img_pred_df dataset variables
img_pred_df.columns
```

Out[28]:
```
Index(['tweet_id', 'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog',
'p2',
       'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
      dtype='object')
```

In [29]:
```
# Showing the first 5 rows of img_pred_df dataset
print(img_pred_df.head())
```

```
             tweet_id                                                jpg_u
rl  \
0  666020888022790149  https://pbs.twimg.com/media/CT4udn0WwAA0aMy.j
pg (https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg)
1  666029285002620928  https://pbs.twimg.com/media/CT42GRgUYAA5iDo.j
pg (https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg)
2  666033412701032449  https://pbs.twimg.com/media/CT4521TWwAEvMyu.j
pg (https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg)
3  666044226329800704  https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.j
pg (https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg)
4  666049248165822465  https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.j
pg (https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg)

   img_num                     p1   p1_conf  p1_dog                  
p2  \
0        1  Welsh_springer_spaniel  0.465074    True                co
llie
1        1                redbone  0.506826    True  miniature_pins
cher
2        1         German_shepherd  0.596461    True              mali
nois
3        1     Rhodesian_ridgeback  0.408143    True               red
bone
4        1      miniature_pinscher  0.560311    True           Rottwe
iler

    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 [30]:
```
# Showing the last 5 rows of img_pred_df dataset
print(img_pred_df.tail())
```

```
                tweet_id                                                jp
g_url  \
2070  891327558926688256  https://pbs.twimg.com/media/DF6hr6BUMAAzZg
T.jpg (https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg)
2071  891689557279858688  https://pbs.twimg.com/media/DF_q7IAWsAEuuN
8.jpg (https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg)
2072  891815181378084864  https://pbs.twimg.com/media/DGBdLU1WsAANxJ
9.jpg (https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg)
2073  892177421306343426  https://pbs.twimg.com/media/DGGmoV4XsAAUL6
n.jpg (https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg)
2074  892420643555336193  https://pbs.twimg.com/media/DGKD1-bXoAAIAU
K.jpg (https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg)

      img_num           p1   p1_conf  p1_dog                   p2   p
2_conf  \
2070        2       basset  0.555712    True    English_springer  0.
225770
2071        1  paper_towel  0.170278   False  Labrador_retriever  0.
168086
2072        1    Chihuahua  0.716012    True             malamute  0.
078253
2073        1    Chihuahua  0.323581    True             Pekinese  0.
090647
2074        1       orange  0.097049   False                bagel  0.
085851

      p2_dog                          p3   p3_conf  p3_dog
2070    True  German_short-haired_pointer  0.175219    True
2071    True                     spatula  0.040836   False
2072    True                      kelpie  0.031379    True
2073    True                    papillon  0.068957    True
2074   False                      banana  0.076110   False
```

In [31]:
```
# Showing 5 ramdom samples
print(img_pred_df.sample(5))
```

```
                tweet_id
jpg_url  \
528   676776431406465024  https://pbs.twimg.com/ext_tw_video_thumb/6
7677... (https://pbs.twimg.com/ext_tw_video_thumb/67677...)
691   684200372118904832    https://pbs.twimg.com/media/CX7EkuHWkAES
LZk.jpg (https://pbs.twimg.com/media/CX7EkuHWkAESLZk.jpg)
1432  773308824254029826    https://pbs.twimg.com/media/CrtYRMEWIAAU
kCl.jpg (https://pbs.twimg.com/media/CrtYRMEWIAAUkCl.jpg)
422   674051556661161984    https://pbs.twimg.com/media/CVq2UHwWEAAd
uMw.jpg (https://pbs.twimg.com/media/CVq2UHwWEAAduMw.jpg)
261   670786190031921152    https://pbs.twimg.com/media/CU8ceuxWUAAL
MEo.jpg (https://pbs.twimg.com/media/CU8ceuxWUAALMEo.jpg)

      img_num             p1   p1_conf  p1_dog                   p2
p2_conf  \
528         1        doormat  0.201346   False          dishwasher
0.191749
691         1          llama  0.681347   False                 ram
0.120142
1432        1  shopping_cart  0.572349   False  Labrador_retriever
0.151406
422         1       Shih-Tzu  0.179777    True              badger
0.160580
261         1          dingo  0.777124   False            Pembroke
0.127438

      p2_dog                p3   p3_conf  p3_dog
528    False         microwave  0.038110   False
691    False               hog  0.043686   False
1432    True   shopping_basket  0.107102   False
422    False  three-toed_sloth  0.132154   False
261     True          Cardigan  0.024007    True
```

In [32]:
```
# Show the data types of each variable
img_pred_df.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 [33]:
```
# Check descriptive statistics of img_pred_df
img_pred_df.describe()
```

Out[33]:

	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 [34]:
```
# Show number of duplicated entries in img_pred_df dataset
print('Number of Duplicated entries in img_pred_df: '
      + '\n' +str(sum(img_pred_df.duplicated())))
```

```
Number of Duplicated entries in img_pred_df:
0
```

In [35]:
```
# Show the number of entries for each unique element
# under the variables P1, P2, P3.

print('Value Counts for P1' + '\n')
print(img_pred_df['p1'].value_counts())
print('\n' + 'Value Counts for P2' + '\n')
print(img_pred_df['p2'].value_counts())
print('\n' + 'Value Counts for P3' + '\n')
print(img_pred_df['p3'].value_counts())
```

```
Value Counts for P1

golden_retriever             150
Labrador_retriever           100
Pembroke                      89
Chihuahua                     83
pug                           57
chow                          44
Samoyed                       43
toy_poodle                    39
Pomeranian                    38
malamute                      30
cocker_spaniel                30
French_bulldog                26
Chesapeake_Bay_retriever      23
miniature_pinscher            23
seat_belt                     22
Siberian_husky                20
German_shepherd               20
```

In [36]:
```
# List all of tweet_json dataset variables
tweet_json.columns
```

Out[36]:
```
Index(['tweet_id', 'favorite_count', 'retweet_count'], dtype='object
')
```

In [37]:
```
# Showing the first 5 rows of tweet_json dataset
tweet_json.head(3)
```

Out[37]:

	tweet_id	favorite_count	retweet_count
0	666020888022790149	2562	515
1	666029285002620928	130	47
2	666033412701032449	125	44

In [38]:
```
# Showing the last 5 rows of tweet_json dataset
tweet_json.tail(3)
```

Out[38]:

	tweet_id	favorite_count	retweet_count
2353	892420643555336193	38585	8529
2354	892420643555336193	38585	8529
2355	892420643555336193	38585	8529

In [39]:
```
# Showing 5 ramdom samples
tweet_json.sample(3)
```

Out[39]:

	tweet_id	favorite_count	retweet_count
2323	888078434458587136	21653	3486
1133	714258258790387713	3181	778
1441	757400162377592832	16293	7527

In [40]:
```
# Show the data types of each variable
tweet_json.info()
```

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

In [41]:
```
# Check descriptive statistics of tweet_json
tweet_json.describe()
```

Out[41]:

	tweet_id	favorite_count	retweet_count
count	2.356000e+03	2356.000000	2356.000000
mean	7.431053e+17	8212.123514	3016.802632
std	6.907537e+16	12379.509271	4951.470774
min	6.660209e+17	0.000000	0.000000
25%	6.783989e+17	1409.250000	604.000000
50%	7.196279e+17	3563.000000	1418.000000
75%	7.998353e+17	10115.250000	3514.250000
max	8.924206e+17	142530.000000	76826.000000

In [42]:
```
# Show number of duplicated entries in tweet_json dataset
print('Number of Duplicated entries in tweet_json: '
      + '\n' +str(sum(tweet_json.duplicated())))
```

```
Number of Duplicated entries in tweet_json:
13
```

In [43]:
```
# Find out what the duplicated entries are
tweet_json[tweet_json.duplicated()]
```

Out[43]:

	tweet_id	favorite_count	retweet_count
2343	892420643555336193	38585	8529
2344	892420643555336193	38585	8529
2345	892420643555336193	38585	8529
2346	892420643555336193	38585	8529
2347	892420643555336193	38585	8529
2348	892420643555336193	38585	8529
2349	892420643555336193	38585	8529
2350	892420643555336193	38585	8529
2351	892420643555336193	38585	8529
2352	892420643555336193	38585	8529
2353	892420643555336193	38585	8529
2354	892420643555336193	38585	8529
2355	892420643555336193	38585	8529

Clean Data

In [44]:
```
# Copy dataframes for the cleaning process
twitter_archive_clean = twitter_archive.copy()
img_pred_df_clean = img_pred_df.copy()
tweet_json_clean = tweet_json.copy()
```

Quality Issue #1

Define

Convert the data type for `timestamp` variable to datetime

Code

In [45]:
```
# Convert `timestamp` to datetime
twitter_archive_clean['timestamp'] = pd.to_datetime(twitter_archive_cl
```

In [46]:
```
# To remove the hours and minutes by applying lambda and use date()
twitter_archive_clean['timestamp'] = pd.to_datetime(twitter_archive_cl
```

Test

In [47]:
```
# Check datatypes for the following variables
twitter_archive_clean.dtypes
```

Out[47]:
```
tweet_id                      int64
in_reply_to_status_id       float64
in_reply_to_user_id         float64
timestamp                    object
source                       object
text                         object
retweeted_status_id         float64
retweeted_status_user_id    float64
retweeted_status_timestamp   object
expanded_urls                object
rating_numerator              int64
rating_denominator            int64
name                         object
doggo                        object
floofer                      object
pupper                       object
puppo                        object
dtype: object
```

Quality Issue #2

Define

Convert `tweet_id` to string

Code

In [48]:
```
# Convert of the tweet ids to string
twitter_archive_clean['tweet_id'] = twitter_archive_clean['tweet_id'].a
img_pred_df_clean['tweet_id'] = img_pred_df_clean['tweet_id'].astype(st
tweet_json_clean['tweet_id'] = tweet_json_clean['tweet_id'].astype(str
```

Test

In [49]:
```
print(twitter_archive_clean['tweet_id'].dtypes)
print(img_pred_df_clean['tweet_id'].dtypes)
print(tweet_json_clean['tweet_id'].dtypes)
```

```
object
object
object
```

Quality Issue #3

Define

Remove observations that have retweets and replies

Code

In [50]:
```
# Keep only the null values of retweets and replies and assign them to
not_retweet = twitter_archive_clean['retweeted_status_user_id'].isnull
not_reply = twitter_archive_clean['in_reply_to_user_id'].isnull()
twitter_archive_clean = twitter_archive_clean[not_retweet&not_reply]
```

In [51]:
```
# After the assignnment to twitter_archive you can now see the retweet
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 2355 to 0
Data columns (total 17 columns):
tweet_id                      2097 non-null object
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(2), object(11)
memory usage: 294.9+ KB
```

In [52]:
```
# Remove the retweet and reply variables now that there are 0 observat
twitter_archive_clean = twitter_archive_clean.drop(['in_reply_to_statu
                                                    'in_reply_to_user_
                                                    'retweeted_status_
                                                   'retweeted_status_u
                                                   'retweeted_status_t
                                                   axis=1)
```

Test

In [53]:
```
# The following variables have now been removed
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 2355 to 0
Data columns (total 12 columns):
tweet_id              2097 non-null object
timestamp             2097 non-null object
source                2097 non-null object
text                  2097 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: int64(2), object(10)
memory usage: 213.0+ KB
```

Quality Issue #3

Define

Remove html tags in `source` variable

Code

In [54]:
```
# Stripping out the HTML tags by using the regex expression
twitter_archive_clean['source'] = twitter_archive_clean['source'].str.
```

Test

In [55]:
```
# You can now see the following strings without the HTML tags
twitter_archive_clean['source'].value_counts()
```

Out[55]:
```
Twitter for iPhone     1964
Vine - Make a Scene      91
Twitter Web Client       31
TweetDeck                11
Name: source, dtype: int64
```

In [56]:
```
# See a sample of the twitter_archive dataframe
twitter_archive_clean.loc[:,['tweet_id','source']].sample(10)
```

Out[56]:

	tweet_id	source
1665	682750546109968385	Twitter for iPhone
2308	666817836334096384	Twitter for iPhone
1225	714141408463036416	Twitter for iPhone
1733	679777920601223168	Twitter for iPhone
1072	739979191639244800	Twitter for iPhone
619	796149749086875649	Twitter for iPhone
445	819227688460238848	Twitter for iPhone
93	874057562936811520	Twitter for iPhone
178	857263160327368704	Twitter for iPhone
174	858107933456039936	Twitter for iPhone

Quality Issue #4

Define

Remove tweets that do not contain images

Code

In [57]:
```
# Check to see how many observations are null values
sum(twitter_archive_clean['expanded_urls'].isnull())
```

Out[57]:
```
3
```

In [58]:
```
# Assign the variable 'expanded_urls' to twitter_archive_clean without
twitter_archive_clean = twitter_archive_clean.dropna(subset=['expanded
```

Test

In [59]:
```
# To make sure there are no more null values
sum(twitter_archive_clean['expanded_urls'].isnull())
```

Out[59]: 0

In [60]:
```
# The number of observations for 'expanded_urls' have gone down
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2094 entries, 2355 to 0
Data columns (total 12 columns):
tweet_id              2094 non-null object
timestamp             2094 non-null object
source                2094 non-null object
text                  2094 non-null object
expanded_urls         2094 non-null object
rating_numerator      2094 non-null int64
rating_denominator    2094 non-null int64
name                  2094 non-null object
doggo                 2094 non-null object
floofer               2094 non-null object
pupper                2094 non-null object
puppo                 2094 non-null object
dtypes: int64(2), object(10)
memory usage: 212.7+ KB
```

Quality Issue #5

Define

To convert the rating numerator and rating denominator variables to the datatype 'float'

Code

In [61]:
```
# Convert the variables to float and then assign them to the twitter_a
twitter_archive_clean[['rating_numerator', 'rating_denominator']] = tw
```

Test

In [62]:
```
# Check to see if they have been converted
twitter_archive_clean[['rating_numerator','rating_denominator']].dtypes
```

Out[62]:
```
rating_numerator      float64
rating_denominator    float64
dtype: object
```

In [63]:
```
# A sample of the twitter_archive_clean dataframe
twitter_archive_clean.loc[:,['tweet_id','text','rating_numerator','rat
```

Out[63]:

	tweet_id	text	rating_numerator	rating_denominator
1103	735256018284875776	This is Kellogg. He accidentally opened the fr...	8.0	10.0
578	800751577355128832	Say hello to Mauve and Murphy. They're rather ...	12.0	10.0
1388	700462010979500032	This is Murphy. He's a mini golden retriever. ...	6.0	10.0
2088	670792680469889025	This is Antony. He's a Sheraton Tetrahedron. S...	7.0	10.0
697	786595970293370880	This is Dale. He's a real spookster. Did me qu...	11.0	10.0

Quality Issue #6

Define

Remove any invalid dog names that are lowercase and replace them with 'None'

Code

In [64]:
```
# Search for all of the lowercase names and assign them to lowercase_n
lowercase_names = twitter_archive_clean[twitter_archive_clean['name'].s
```

In [65]:
```
# See the list of lowercase names
lowercase_names['name'].value_counts()
```

Out[65]:
```
a               55
the              8
an               6
one              4
very             4
quite            3
just             3
actually         2
not              2
getting          2
unacceptable     1
officially       1
old              1
this             1
my               1
light            1
infuriating      1
by               1
space            1
life             1
mad              1
his              1
incredibly       1
all              1
such             1
Name: name, dtype: int64
```

In [66]:
```
# Take out the key values of the lowercase names and assign them to a
lowercase_names = lowercase_names['name'].value_counts().keys().tolist
```

In [67]:
```
# All of the lowercase names in a list
lowercase_names
```

Out[67]:
```
['a',
 'the',
 'an',
 'one',
 'very',
 'quite',
 'just',
 'actually',
 'not',
 'getting',
 'unacceptable',
 'officially',
 'old',
 'this',
 'my',
 'light',
 'infuriating',
 'by',
 'space',
 'life',
 'mad',
 'his',
 'incredibly',
 'all',
 'such']
```

In [68]:
```
# Use a for loop to replace the lowercase names to 'None'
for dog_name in twitter_archive_clean['name']:
    if dog_name in lowercase_names:
        twitter_archive_clean[twitter_archive_clean['name'] == dog_name
```

Test

In [69]:
```
# We can now see all of the lowercase names have been changed to 'None
twitter_archive_clean['name'].value_counts()
```

Out[69]:
```
None         600
Charlie       11
Lucy          11
Oliver        10
Cooper        10
Tucker         9
Penny          9
Lola           8
Sadie          8
Winston        8
Toby           7
Daisy          7
```

```
Bailey        6
Oscar         6
Bella         6
Bo            6
Stanley       6
Koda          6
Jax           6
Rusty         5
Louis         5
Dave          5
Milo          5
Leo           5
Bentley       5
Chester       5
Scout         5
Buddy         5
Boomer        4
Reggie        4
             ...
Chloe         1
Ridley        1
Cedrick       1
Nida          1
Kane          1
Happy         1
Ace           1
Florence      1
Stormy        1
Logan         1
Cheesy        1
Hanz          1
Tonks         1
William       1
Joshwa        1
Striker       1
Shawwn        1
Jeffri        1
Tedders       1
Chubbs        1
Ester         1
Dug           1
Duddles       1
Kellogg       1
Cannon        1
Beemo         1
Blu           1
Pupcasso      1
Acro          1
Lilli         1
Name: name, Length: 930, dtype: int64
```

Quality Issue #7

Define

To capitalise the first letter and remove underscores in variables p1, p2 and p3.

Code

In [70]:
```
# Capitalise the first letter of each observation
img_pred_df_clean['p1'] = img_pred_df_clean['p1'].str.title()
img_pred_df_clean['p2'] = img_pred_df_clean['p2'].str.title()
img_pred_df_clean['p3'] = img_pred_df_clean['p3'].str.title()
```

In [71]:
```
# Remove the underscore and replace them with spaces
img_pred_df_clean['p1'] = img_pred_df_clean['p1'].str.replace('_', ' '
img_pred_df_clean['p2'] = img_pred_df_clean['p2'].str.replace('_', ' '
img_pred_df_clean['p3'] = img_pred_df_clean['p3'].str.replace('_', ' '
```

Test

In [72]:
```
# We can see the first letters have been capitalised
img_pred_df_clean.loc[:, ['p1','p2','p3']].sample(10)
```

Out[72]:

	p1	p2	p3
1703	Tibetan Mastiff	Tibetan Terrier	Otterhound
150	Vacuum	Pug	Toilet Tissue
1291	Labrador Retriever	Golden Retriever	Dingo
1007	Borzoi	Wire-Haired Fox Terrier	English Setter
472	Labrador Retriever	Great Dane	Staffordshire Bullterrier
878	Pug	Pekinese	Sunglasses
1009	Labrador Retriever	Pomeranian	Golden Retriever
753	Sorrel	Horse Cart	Arabian Camel
1341	Chesapeake Bay Retriever	Vizsla	Weimaraner
1240	Traffic Light	Fountain	Space Shuttle

Quality Issue #8

Define

To rename confusing variable headers

Code

In [73]:
```
# Rename the current variables
img_pred_df_clean = img_pred_df_clean.rename(columns={'jpg_url': 'imag
                                                      'p1':'first_pred
                                                      'p1_dog':'first_
                                                      'p2':'second_pre
                                                      'p2_dog':'second_
                                                      'p3':'third_pred
                                                      'p3_dog':'third_
                                                     })
```

In [74]:
```
# Variable names are now easier to read and understand
img_pred_df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id                  2075 non-null object
image_url                 2075 non-null object
image_number              2075 non-null int64
first_prediction          2075 non-null object
first_pred_confidence     2075 non-null float64
first_pred_is_dog         2075 non-null bool
second_prediction         2075 non-null object
second_pred_confidence    2075 non-null float64
second_pred_is_dog        2075 non-null bool
third_prediction          2075 non-null object
third_pred_confidence     2075 non-null float64
third_pred_is_dog         2075 non-null bool
dtypes: bool(3), float64(3), int64(1), object(5)
memory usage: 152.1+ KB
```

Quality Issue #9

Define

Remove 13 duplications of Twitter ID '892420643555336193'

Code

In [75]:
```
# Remove the duplicated Twitter ID by using drop_duplicates
remove_dup_tweet_id = tweet_json.drop_duplicates(subset='tweet_id')
```

Test

In [76]:
```
# The duplicated entries have been removed
remove_dup_tweet_id[remove_dup_tweet_id['tweet_id']==892420643555336193
```

Out[76]:

	tweet_id	favorite_count	retweet_count
2342	892420643555336193	38585	8529

Tidyness Issue #1:

Define

Melt the twitter_archive dataframe and have only one column instead of 4 ('doggo', 'floofer' ,'pupper' and 'puppo')

Code

In [77]:
```
# Use Pandas Melt and combine the 4 variables('doggo', 'floofer' ,'pupp
twitter_archive_clean = pd.melt(twitter_archive_clean, id_vars=['tweet_
                                                                'times
                                                                'sourc
                                                                'text'
                                                                'expan
                                                                'ratin
                                                                'ratin
                                                                'name'
                                                               ,value_
```

In [78]:
```
print(twitter_archive_clean.sample(5))
```

```
                tweet_id   timestamp              source  \
3789  811985624773361665  2016-12-22  Twitter for iPhone
4006  849051919805034497  2017-04-04  Twitter for iPhone
8200  850380195714523136  2017-04-07  Twitter for iPhone
4685  675891555769696257  2015-12-13  Twitter for iPhone
7821  782598640137187329  2016-10-02  Twitter for iPhone

                                                  text  \
3789  Say hello to Ted. He accidentally opened the f...
4006  This is Kevin. Kevin doesn't give a single h*c...
8200  This is Leo. He's a personal triathlon coach. ...
4685  This is Donny. He's summoning the demon monste...
7821  This is Timmy. He's quite large. According to ...

                                          expanded_urls  rating_nume
rator  \
3789  https://twitter.com/dog_rates/status/811985624...
(https://twitter.com/dog_rates/status/811985624...)
11.0
4006  https://twitter.com/dog_rates/status/849051919...
(https://twitter.com/dog_rates/status/849051919...)
13.0
8200  https://twitter.com/dog_rates/status/850380195...
(https://twitter.com/dog_rates/status/850380195...)
13.0
4685  https://twitter.com/dog_rates/status/675891555...
(https://twitter.com/dog_rates/status/675891555...)
6.0
7821  https://twitter.com/dog_rates/status/782598640...
(https://twitter.com/dog_rates/status/782598640...)
11.0

      rating_denominator   name variable dog_stage
3789                10.0    Ted  floofer      None
4006                10.0  Kevin  floofer      None
8200                10.0    Leo    puppo      None
4685                10.0  Donny   pupper      None
7821                10.0  Timmy    puppo      None
```

In [79]:
```
# Drop 'variable' as it is no longer needed
twitter_archive_clean = twitter_archive_clean.drop('variable', axis=1)
```

In [80]:
```
# Sort values by 'dog_stage'
twitter_archive_clean = twitter_archive_clean.sort_values('dog_stage')
```

In [81]:
```
# Remove duplicates by 'tweet_id'
twitter_archive_clean = twitter_archive_clean.drop_duplicates('tweet_i
```

In [82]:
```
# Convert the variable 'dog_stage' into a category
twitter_archive_clean['dog_stage'] = twitter_archive_clean['dog_stage'
```

Test

In [83]:
```
# The variables 'doggo', 'floofer' ,'pupper' and 'puppo' have now been
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1991 entries, 2015 to 8354
Data columns (total 9 columns):
tweet_id              1990 non-null object
timestamp             1990 non-null object
source                1990 non-null object
text                  1990 non-null object
expanded_urls         1990 non-null object
rating_numerator      1990 non-null float64
rating_denominator    1990 non-null float64
name                  1990 non-null object
dog_stage             1990 non-null category
dtypes: category(1), float64(2), object(6)
memory usage: 142.1+ KB
```

In [84]:
```
len(twitter_archive_clean[twitter_archive_clean['dog_stage'] != None])
```

Out[84]:
```
1991
```

In [85]:
```
twitter_archive_clean['dog_stage'].value_counts()
```

Out[85]:
```
None       1669
pupper      217
doggo        70
puppo        24
floofer      10
Name: dog_stage, dtype: int64
```

Tidyness Issue #2:

Define

Consolidate and merge the 3 datasets: twitter_archive, img_pred_df and tweet_json into one as they are describing the same tweet

Code

In [86]:
```
# Merge twitter_archive_clean and img_pred_df_clean to create a new da
archive_img_df = pd.merge(twitter_archive_clean, img_pred_df_clean, on

# Consolidate all 3 tables and create a new dataframe 'twitter_final_d
twitter_final_df = pd.merge(archive_img_df, tweet_json_clean, on=['twe
```

Test

In [87]:
```
# All 3 tables' variables are in one dataframe under twitter_final_df
twitter_final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1886 entries, 0 to 1885
Data columns (total 22 columns):
tweet_id                  1886 non-null object
timestamp                 1886 non-null object
source                    1886 non-null object
text                      1886 non-null object
expanded_urls             1886 non-null object
rating_numerator          1886 non-null float64
rating_denominator        1886 non-null float64
name                      1886 non-null object
dog_stage                 1886 non-null category
image_url                 1886 non-null object
image_number              1886 non-null int64
first_prediction          1886 non-null object
first_pred_confidence     1886 non-null float64
first_pred_is_dog         1886 non-null bool
second_prediction         1886 non-null object
second_pred_confidence    1886 non-null float64
second_pred_is_dog        1886 non-null bool
third_prediction          1886 non-null object
third_pred_confidence     1886 non-null float64
third_pred_is_dog         1886 non-null bool
favorite_count            1885 non-null float64
retweet_count             1885 non-null float64
dtypes: bool(3), category(1), float64(7), int64(1), object(10)
memory usage: 287.5+ KB
```

Store

Data Analysis and Visualisation

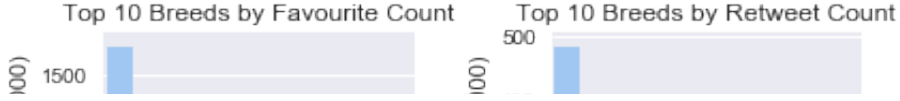
In [88]:
```
# Store the dataset into a csv file
twitter_final_df.to_csv('twitter_final_df.csv', encoding='utf-8')
```

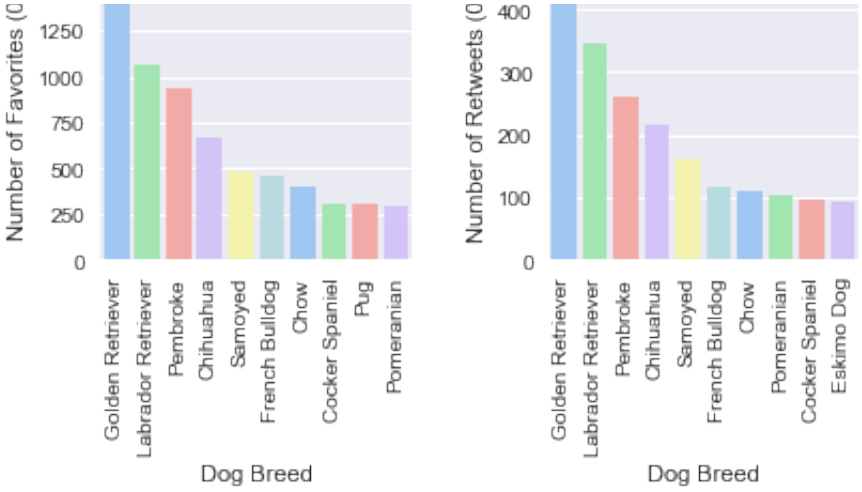
Analysis #1

Top 10 Most Popular Dog Breeds

In [89]:
```
# To filter out entries that are not identified as dogs
dog_breed = twitter_final_df[twitter_final_df['first_pred_is_dog']==Tr
```

In [97]:
```
# Create a subplot to show 2 plots: plot 1 - number of favourite tweet

plt.subplot(1,2,1)
# Show the top 10 dog breeds based on favourite tweets by using Groupb
top10breed_fav_counts = dog_breed.groupby(['first_prediction'])['favor
# Sort it in descending order
top10breed_fav_counts = top10breed_fav_counts.sort_values(ascending=Fa
# Divide by 1000 to make it easier for people to view
top10breed_fav_counts = top10breed_fav_counts/1000
# Remove decimal points by converting it to an integer
top10breed_fav_counts = top10breed_fav_counts.astype(int)

# Create a bar chart based on top10breed_fav_counts by using seaborn l
ax = top10breed_fav_counts.reset_index().pipe((sns.barplot, 'data'),
x = 'first_prediction', y='favorite_count', palette='pastel')
plt.xticks(rotation=90)
plt.ylabel('Number of Favorites (000)')
plt.xlabel('Dog Breed')
plt.title('Top 10 Breeds by Favourite Count')

plt.subplot(1,2,2)
# Show the top 10 dog breeds based on retweets by using Groupby on var
top10breed_retweet_counts = dog_breed.groupby(['first_prediction'])['r
# Sort it in descending order
top10breed_retweet_counts = top10breed_retweet_counts.sort_values(asce
# Divide by 1000 to make it easier for people to view
top10breed_retweet_counts = top10breed_retweet_counts/1000
# Remove decimal points by converting it to an integer
top10breed_retweet_counts = top10breed_retweet_counts.astype(int)

# Create a bar chart based on top10breed_retweet_counts by using seabo
ax = top10breed_retweet_counts.reset_index().pipe((sns.barplot, 'data'
x = 'first_prediction', y='retweet_count', palette='pastel')
plt.xticks(rotation=90)
plt.ylabel('Number of Retweets (000)')
plt.xlabel('Dog Breed')
plt.title('Top 10 Breeds by Retweet Count')

plt.tight_layout()
plt.show()
```

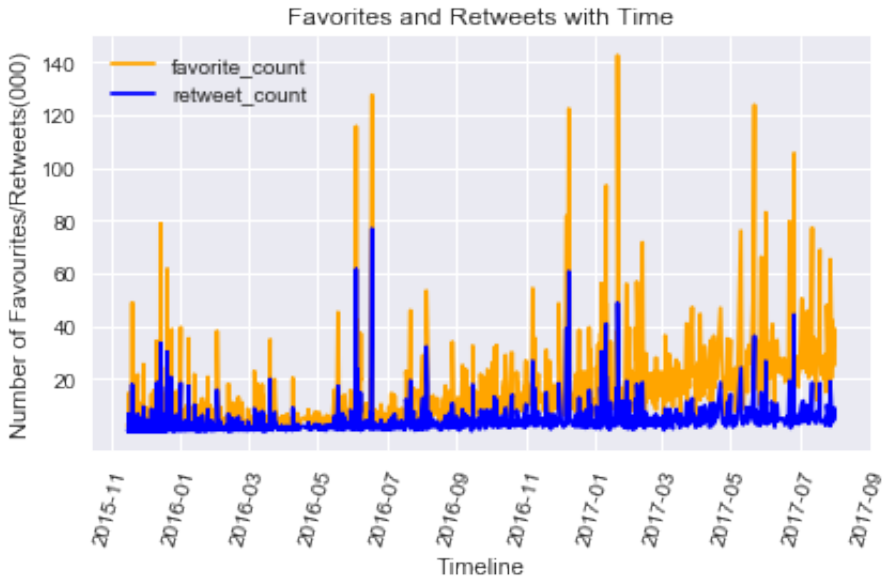




Analysis #2

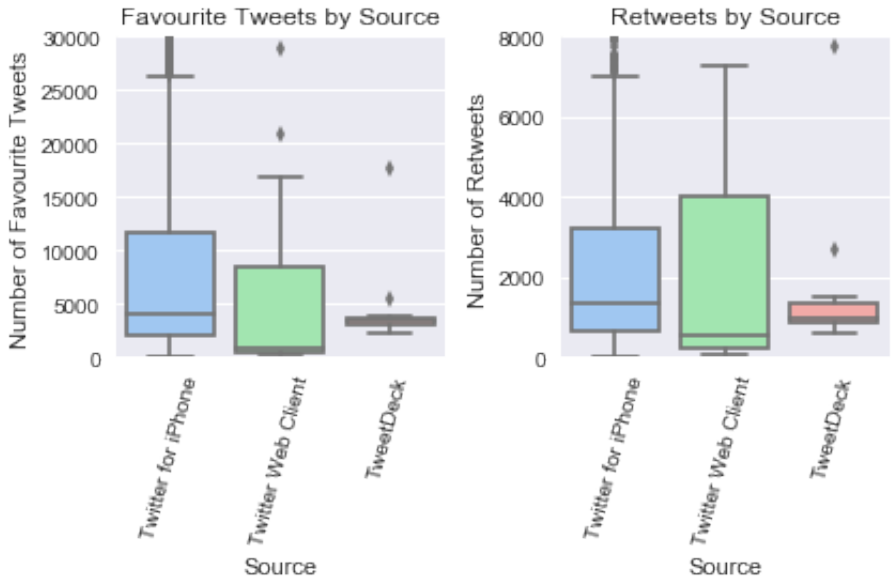
In [91]:
```
# Set the index to datetime using the variable `timestamp`
twitter_final_df.set_index('timestamp', inplace=True)
```

In [92]:
```
# To create a timeline for the number favourite tweets and retweets in

# To make it easier for people read the labels by removing the the 3 z
y_time = [20000, 40000, 60000, 80000, 100000, 120000, 140000]
y_time_labels = [20, 40, 60, 80, 100, 120, 140]

# Create a line plot using favourite count and retweet count using col
twitter_final_df[['favorite_count', 'retweet_count']].plot(color=['ora
plt.title('Favorites and Retweets with Time')
plt.xlabel('Timeline')
plt.yticks(y_time, y_time_labels)
plt.xticks(rotation=75)
plt.ylabel('Number of Favourites/Retweets(000)')

plt.tight_layout()
plt.show()
```



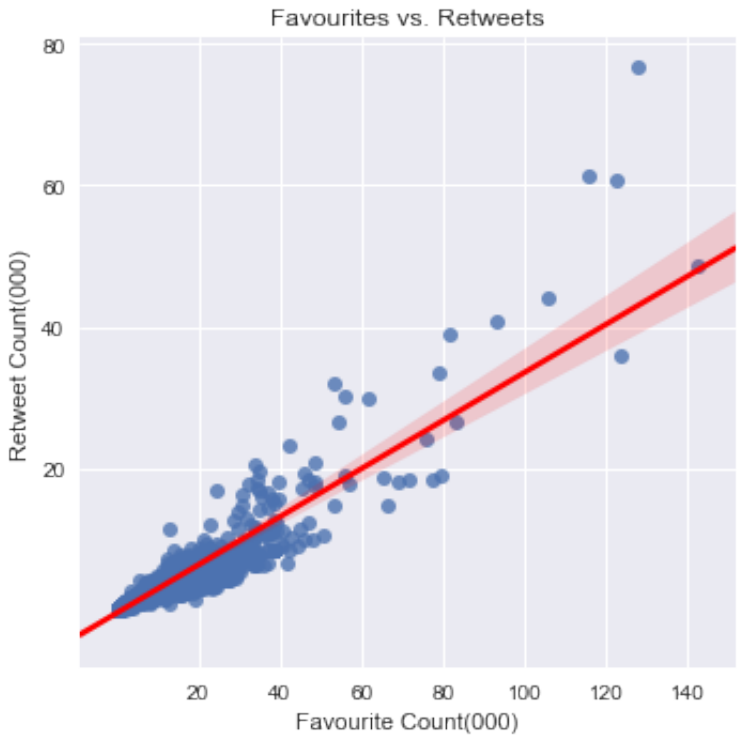
Analysis #3

In [93]:
```
# Create a subplot to show 2 plots based on number of favourite tweets

plt.subplot(1,2,1)
# Create a boxplot on `favourite count` and `source`
sns.boxplot(data=twitter_final_df, x='source',y='favorite_count', pale
plt.title('Favourite Tweets by Source')
plt.xticks(rotation=75)
plt.xlabel('Source')
plt.ylabel('Number of Favourite Tweets')
# Limiting the number of favourite counts to 3,000 to make it easier t
plt.ylim(0,30000)

plt.subplot(1,2,2)
# Create a boxplot on `retweet count` and `source`
sns.boxplot(data=twitter_final_df, x='source', y='retweet_count', pale
plt.title('Retweets by Source')
plt.xticks(rotation=75)
plt.xlabel('Source')
plt.ylabel('Number of Retweets')
# Limiting the number of retweet counts to 8,000 to make it easier to
plt.ylim(0,8000)

plt.tight_layout()
plt.show()
```



Analysis #4

In [94]:
```
# To show the relationship between favourite tweets amd retweets

# Remove the 3 zeros to make it easier to view
x = [20000, 40000, 60000, 80000, 100000, 120000, 140000]
y = [20000, 40000, 60000, 80000]
labels = [20, 40, 60, 80, 100, 120, 140]

# Show only the specified the categories of the dog stages
hue_order = ['pupper', 'doggo', 'puppo', 'floofer']

# Create a scatter plot and show relationship between 2 variables and
sns.lmplot(data=twitter_final_df, x='favorite_count', y='retweet_count
plt.xticks(x, labels)
plt.yticks(y, labels)
plt.xlabel('Favourite Count(000)')
plt.ylabel('Retweet Count(000)')
plt.title('Favourites vs. Retweets')

plt.tight_layout()
plt.show()
```



In [95]:
```
# To investigate further by adding another variable for analysis

x = [20000, 40000, 60000, 80000, 100000, 120000, 140000]
y = [20000, 40000, 60000, 80000]
labels = [20, 40, 60, 80, 100, 120, 140]
hue_order = ['pupper', 'doggo', 'puppo', 'floofer']

# Using variable `dog_stage` to findout the spread of each dog stage
sns.lmplot(data=twitter_final_df, x='favorite_count', y='retweet_count
           hue='dog_stage', hue_order= hue_order)

plt.title('Favourites vs. Retweets by Dog Stage')
plt.xticks(x, labels)
plt.xlabel('Favourite Count(000)')
plt.ylabel('Retweet Count(000)')
plt.yticks(y, labels)

plt.show()
```